Oct 11, 2022 · 6 min read

Active Metadata: Tracking the Life Cycle and Data Trails of Data Products with Data Lineage

Updated: Dec 9, 2022

To find where are your data from and where exactly did them go - by letting themselves reporting to you.

fst network
Active Metadata
Tracking the Life Cycle and Data Trails of Data Products with Data Lineage

What is Metadata?

In Zhamak Dehghani's works about data mesh, *metadata* is part of a data product (a API-like data pipeline running on some infrastructure) and was described as "information about data" which can make data useful.

In the world of analytical data (which is the focus of many concepts and practises, including data mesh, data fabric and business intelligence), metadata is just as valuable - for it provides the *context* of how data is used, and from which you can extract insights for key business decisions or improvements. Metadata can also be used for machine learning, helping to make AI models *explainable* and *responsible*.

There are three major types of metadata:

- *Technical* or *structural* metadata: technical description of a data object - database table schema, data type or model, etc. which decides how a data can be operated and transformed.
- *Process* or *administrative* metadata: how and when a data is generated and used, as well as its access control and performance, etc. These details are for stewardship and maintenance. The history of the data usage is referred as its *lineage*.
- *Business* or *descriptive* metadata: the value that a data have to its organisation and users, in the forms like names, labels, annotations - which supports discoverability and categorisation.

Active Metadata and Data Lineage

One of the most common metadata tools are **data catalogues**, and one key purpose of them is **data discovery**, which enables users to quickly search and get data across separated data silos. This goal is overlapped with *data virtualisation* and *data fabric*. It also eliminates the need to duplicate data around your company and waste more resources.

However, this is not the only thing you can do with metadata. What if you actively generate new meanings from them? Metadata are, after all, *data*.

In 2021, Gartner had dropped metadata management from their Magic Quadrant report and replaced with *active metadata management*. It is defined as below:

> "Active" metadata attempts to recalibrate this idea by making metadata a computational problem rather than a managerial one. It shifts the process of metadata management from being a passive, static operation to an active, continuous one.
> – Forbes, Active Metadata: What It Is, And Why It Matters (2022)

There are, in fact, many interpretations about how exactly is an "active" metadata. Some believe machine learning has to be leveraged - but in most cases any automated processing and analysing can count as active metadata management.

On many data catalogue tools, one common form of active metadata is **data lineages**. A data lineage indicates what and where the data source is, where data moves to over time, and who with what privilege used them, etc. Along with *audit trails*, you can validate how trustworthy a data is and was a data usage in compliance with your data policy.

Depending on how metadata is generated, it can also be classified as *self-contained* or by *external parsing*. The major data catalogue platforms are naturally the latter.

Types of Data Lineage

There are also three major types of data lineages:

- *Pattern*-based lineage: the lineage is based on structural metadata, for example, database tables, without any codes involved. This lineage reads the relationship between how data are stored.
- *Code*- or *logic*-based lineage: this lineage is based on code logic - for example, SQL scripts or workflow definitions - to understand how they would transform or manipulate data.
- *Log*- or *runtime*-based lineage: this lineage is based on metadata generated by executing data logic code, which can be used to find out how the code are actually executed and what data pipelines have been invoked by users.

Data Mesh: Shifting Lineage to Runtime

Many existing data catalogue tools relies on centralised ETL (Extract-Transform-Load) processes or data streaming services. This is a good model for the traditional data warehouse model or separate data silos. However, as we explained in the data mesh introduction, the scale of (big) data has caused bottlenecks in such architecture. Data mesh is about moving the data pipelines toward domain teams and making them into "data products".

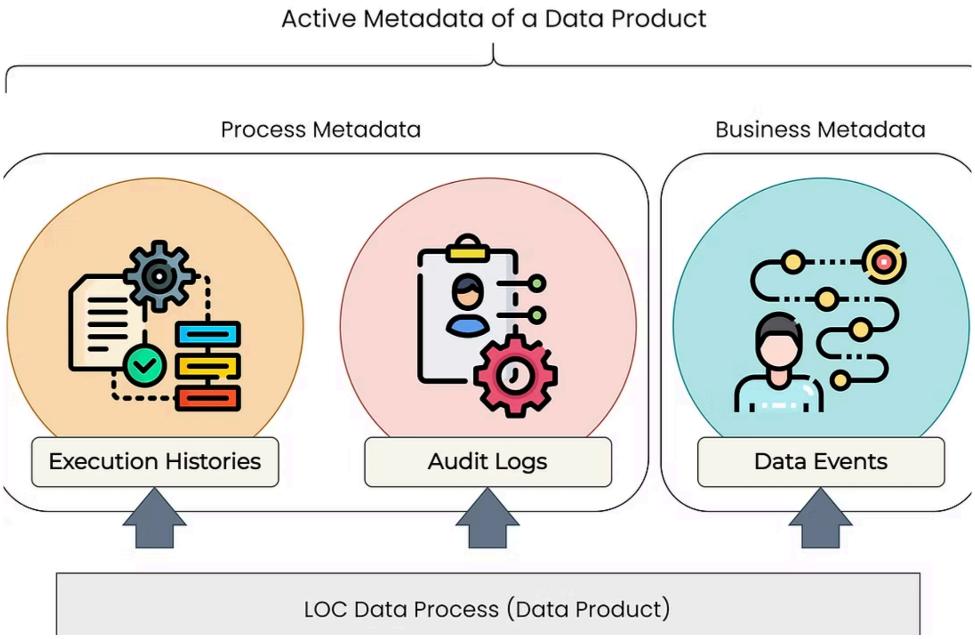
This means in this new decentralised, distributed paradigm, we can no longer only focus on pattern- or code-based lineages: for data products (which are like callable APIs) exists in a virtual plane (data fabric) are the only thing the users want. They would like to know what happened when you invoke a data process, and have it passed anything you want to somewhere.

How LOC Utilise Data Lineage

As a self-serve, serverless platform deeply influenced by the data mesh architecture, FST Network's Logic Operating Centre (LOC) is all about building data pipelines - data processes - between data sources, sometimes not unlike overpasses or highways.

And unlike other data catalogue tools, LOC does not generate lineages from
these sources. Instead, all the lineages are self-contained and would be generated along with the life cycle of data processes. They can help you to manage and improve your data products better.

We will take a look at some data lineages currently supported in LOC.



Execution Histories

Triggers are external events to invoke LOC data processes (which is an **execution** containing one or more **tasks**), including HTTP routes, message queue topics and schedulers. They are deployed by users to pair with specific data processes via their ID.

A trigger itself defines what data processes should be invoked, which is a form of code-based lineage. However, a **execution history** can tell you what really happened during an actual execution, including where did it failed:

- When did the execution happened
- Which trigger invoked this execution
- What payload did the trigger carry
- Tasks in this execution
- Data operations performed in each task

The execution histories are a form of process metadata.

Audit Logs

In a platform that implements the data mesh architecture, *computational governance* is necessary. LOC's **audit logs** - audit trails - show a user's action of managing data processes:

- Who create/delete a data process at when
- Who update a data process (revision) at when

- Who deploy/undeploy a data process at when

This captures part of the data processes life cycle and is crucial for the organization’s **data governance** purposes.

The audit logs are a form of process metadata.

Data Events

LOC also have something different up in its sleeve: **data events** (or simply *events*). These are not the same with triggers we’ve mentioned above or *Event-Driven Architecture (EDA)* events, but a form of runtime metadata generated by the users' code.

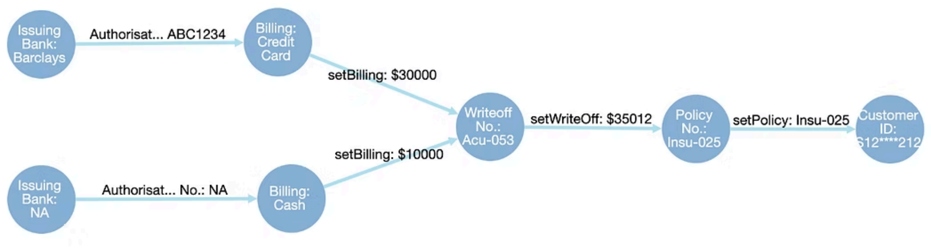
In every data process, users can use so-called *event store agent* to emit or query customised data events:

- Event name
- Event source
- Event target
- Additional payload or metadata fields

Each source (data producer) or target (data consumer) is a logical node that can also be other events’ source or target. Together these nodes can form a longer lineage across data processes, which indicates *data flow* or *data trail* or an actual business process. This also helps users to track data issues outside separate data pipelines.

These events are a form of business metadata, which can only be defined manually - not exactly self-contained - but once the code is in place, they will be able to generate active metadata by task. This also allows users to capture their business logic much better than pattern- or code-based metadata.

The image below is an actual data lineage displayed in LOC’s user interface, LOC Studio:



An emitted event also contains other metadata:

- Date/time of the event (useful for time-series analysis)

- The data process and task that emitted the event

The events can also be used for exchanging data between data processes or triggering other data processes - but that will be a topic for another day.

LOC also offers data discovery for triggers and data processes, which is more similar to traditional data catalogues. Triggers themselves are in fact code-based lineages, since they contain the list of data processes that should to be invoked.

Then again, a perfectly fine task may still fail due to various runtime issues - for example, incorrect data payloads or downed databases. Runtime lineages are more dynamic and thus more important to capture insights on a living, running data platform, especially one that pursues the vision of bringing data mesh into your organisation.

FST Network aims to revolutionise the data landscape by creating world's first cloud-native and serverless Data Mesh platform. Logic Operating Centre (LOC) is fully capable to productise and eventify your data, with easy-to-deploy data pipelines that seamlessly connects everything across your organisation, bringing in agile, accessible and trustworthy business insights. The data race is constantly changing and fiercer than ever - but we are there to give you a winning edge.

Icon credits: flaticon.com