Published in Towards Data Science

Eric Broda Follow

Feb 4, 2022 · 6 min read · ✨ · Listen

Data Mesh Patterns: Immutable Change / Audit Log

Let's review how the "Immutable Change/Audit Log" pattern is used by the Enterprise Data Mesh.

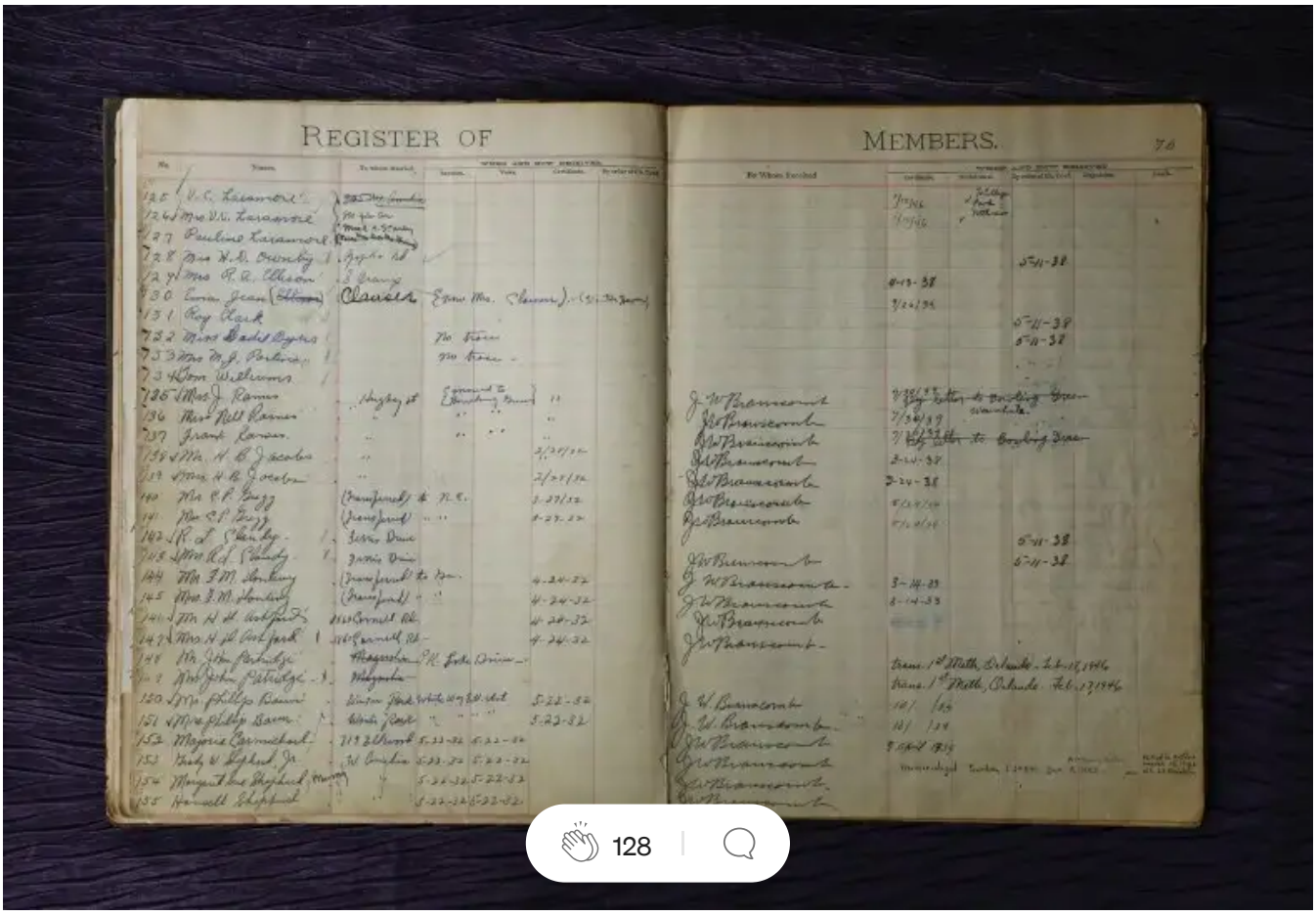

Photo by Mick Haupt on Unsplash

Immutable Change/Audit Log: A Foundational Data Mesh Pattern

This article discusses the Immutable Change/Audit Log and is the third in a series of articles on Foundational Data Mesh Patterns. I will be summarizing the purpose of the pattern, its problem domain and business context, how the pattern works, and candidate vendors that enable this pattern.

This article assumes that you have a high-level understanding of Data Mesh. The following articles should give you some background if you need it:

- Data Mesh principles (more information is available here)
- Data Mesh architecture (more information is available here)
- Data Mesh patterns (more information is available here)

A list of the full series of foundational Data Mesh pattern articles is provided at the end of this article.

Pattern Summary

The Immutable Change/Audit Log pattern tracks data lineage in the Enterprise Data Mesh. It does this by establishing a log of data changes, typically fed from the Enterprise Data Mesh's Change Data Capture (CDC) system, which can be aggregated, analyzed, and sliced-and-diced to support audit and federated governance needs.

Context and Business Problem

Most large enterprises are confounded by seemingly simple problems. Where does my data come from? What happened to it as it moved from one application and database to another? And, who made the changes to the data?

But why is this even important? Well, first, at its most simple, is that basic "good hygiene" required by data governance makes it useful and in many cases required in large enterprises. But more importantly, and especially in regulated industries such as financial services and health care, it is mandated by regulators.

For example, in financial services regulators demand that "important" AI/Machine Learning models (ie. those that directly impact people, or financial risk/position)

are reproducible, traceable, and verifiable. Addressing this need means that an enterprise's data lineage must be well established, understood, and available/visualized at a moments notice.

Unfortunately, enterprises have had to cobble together somewhat bespoke solutions with each targeted to specific applications or database profiles. So, not surprisingly, enterprises are left with an inconsistent set of tools and gaps in capabilities, which, ultimately, hinder agility and time to market.

Solution

The Immutable Change/Audit Log lets an enterprise understand, visualize, and report on the lifecycle — or lineage — of a unit of data (note that I use "unit of data" in the broadest sense, as it could be a data element, data row, data table, depending on the granularity required by your specific situation.)

The Immutable Change/Audit Log listens (on the Event Streaming Backbone) for data change events. It is notified of data change events and automatically captures changes in a data unit (using the Change Data Capture pattern), renders the data unit change (and meta-data) as a "data lineage event", stores the event in an append-only persistent log, and publishes the data lineage event (using the Event Streaming Backbone pattern) so other interested downstream parties can be consume data lineage events.

It also provides the tools to transform data change events into actionable insights. Low-level data change events are grouped and aggregated to view an individual data unit's lifecycle and visualization tools make it easy to find and view a data unit's lifecycle and lineage.

Note that it is a common mistake to conflate this pattern with the popular "event sourcing" pattern (for your convenience, some fantastic information about event sourcing is available here, here, and here). But there are a few important differences.

Where event sourcing provides a persistent log of events, the Immutable Change/Audit log ties data units change — before and after images of the data — and events that caused the data change.

And where the primary entity in event sourcing is the event and ***not*** the data unit upon which it acted upon, the primary entity in the Immutable Change/Log pattern

is both the data unit ***and*** the event that acted upon it.

How It Works

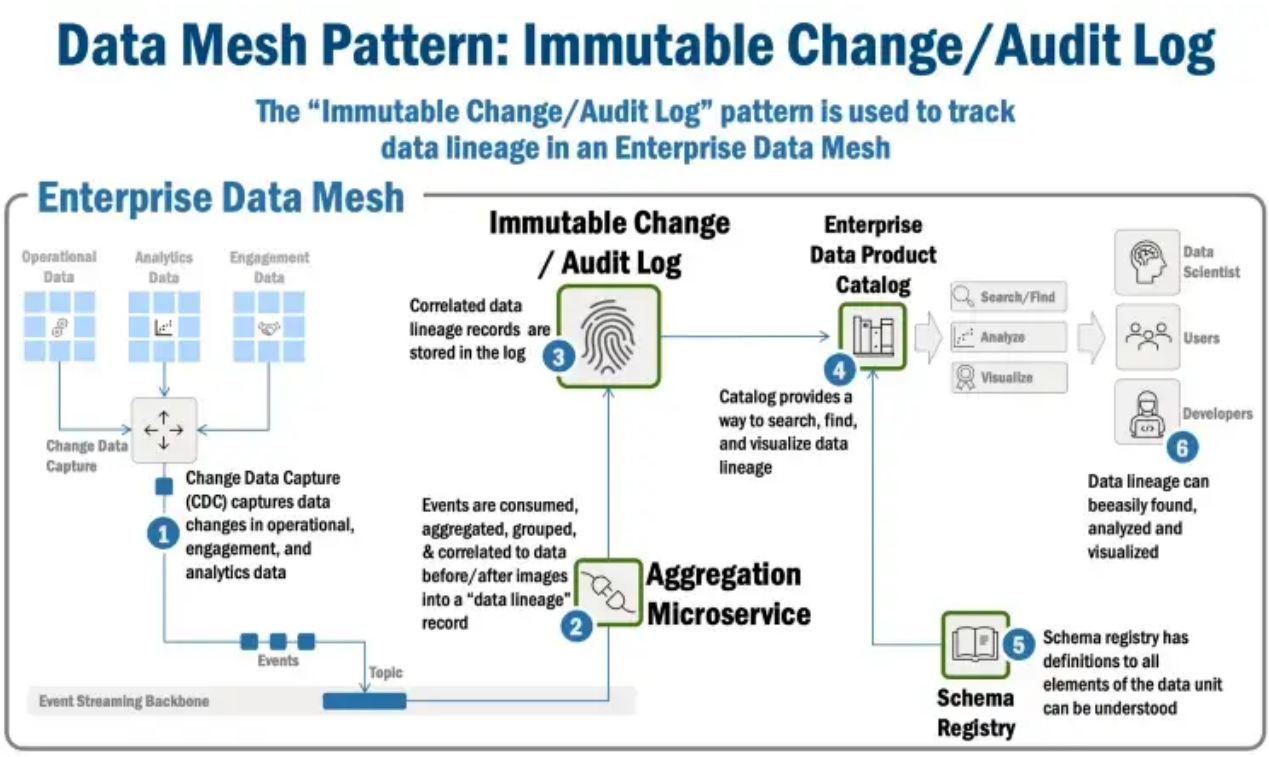

Figure 1, Data Mesh Pattern: Immutable Change/Audit Log

1. Change Data Capture (CDC) captures data changes from a database transaction log from an operational, engagement, or analytics database, formats them as events, and publishes the events (using the Event Streaming Backbone) for consumption by downstream systems.

2. One of the downstream services is an "Aggregation Microservices" that receives the CDC data change event and creates a **data lineage record** that correlates the "before" (prior to the data change) & "after" (the final state of the data) images of the data, and the event that caused the data change.

3. The data lineage record is stored in immutable change/audit log; Note that the "immutable" characteristic is important — this typically implemented as an "append-only" log and hence provides a persistent and unchangeable historical record of a data unit's lifecycle.

4. Data lineage records are forward to an Enterprise Data Product Catalog; The catalog allows data lineage records (as well as many other attributes) to be searched and visualized on demand.

5. A Schema Registry provides definitions for all events as well as data lineage records (and the event information they contain) providing the necessary structure to enable sophisticated data lineage analysis.

6. Users across the enterprise — data scientists, governance team members, and developers can consume data lineage records as needed.

Usage Scenarios

Data scientists use this pattern to understand the lineage of training data to enable reproducibility, traceability, and verifiability of their models, which is required as AI/Machine Learning supports increasingly mission-critical decision making.

Data governance teams also use this pattern to understand data lineage and related consumption patterns across the enterprise.

Vendor Landscape

There are several components that interact to capture, catalog, and visualization of the data lineage records stored by the Immutable Change/Audit Log pattern:

- **Schema Registry**: Since this component provides the definition (typically JSON Schema or AVRO) of events it is tightly coupled to the Event Streaming Backbone. I have had good experiences with the Confluent version of the Kafka Schema Registry.

- **Enterprise Data Product Catalog**: So far, I have not found a specific vendor that address this and have had to build my own bespoke systems that link the Schema Registry, aggregation microservices, event visualization, and a searchable catalog. With this in mind, I usually start with a flexible static open source content management service such as Docusaurus or Hugo that come with solid search/find and content storage features and then layer on aggregation and visualization capabilities. I realize there are probably more sophisticated solutions out there, but this works well and can be quickly implemented.

- **Immutable Log**: Any robust database that can store unstructured data will do here; Also, while the capability is different than event sourcing the persistent log kept by Kafka can be the basis for an immutable change/audit log, although additional capability and configuration probably will need to be implemented.

*Full disclosure: I have **no financial interest** in recommending any of the above products — I am highlighting these products because I have some experience with them, and they have worked well for me.*

Other Articles in this Series

The full series of articles in this series on foundational Data Mesh patterns is list below.

- **Change Data Capture (CDC) pattern**, which tracks changes in a database and captures them as "events" (available here).
- **Event Streaming Backbone pattern**, used by CDC and other applications to publish and subscribe/receive events in an Enterprise Data Mesh (available here).
- **Immutable Change / Audit Log pattern**, which retains logs and tracks data lineage within the Enterprise Data Mesh for future audit and governance purposes (this article).
- **Enterprise Data Product Catalog pattern**, which is a catalog/repository that contains meta-data about Data Products in the Enterprise Data Mesh (coming soon).

Concluding Thoughts

The Immutable Change/Audit Log is a foundational Data Mesh pattern. Coupled with a Schema Registry, and an Enterprise Data Product Catalog, it provides the underlying capability to implement the data lineage required by regulators, auditors, and data governance professionals while implementing a core Data Mesh principle of "federated computational governance").

***

All images in this document except where otherwise noted have been created by Eric Broda (the author of this article). All icons used in the images are stock PowerPoint icons and are free from copyrights.

Data Mesh Data Science Artificial Intelligence Machine Learning

Data Governance