Get unlimited access to the best of Medium for less than $5/week. Become a member

Alchemesh console: The core concepts

Alex Alexandre Gaudin · Following
Published in Alchemesh · 10 min read · Jun 5, 2024

The announcement of the launch of our framework to support the Data Mesh has been made, and we can now begin our new adventure together!

The idea is to share with you, as we develop, our reflections and the technical choices we make.

The goal is to share through these articles our representation of the Data Mesh, present our development approach, get feedback on our choices, and most importantly, try to think together about the challenges of implementing the Data Mesh.

Alchemesh console: Standardize interfaces to facilitate socialization and understanding

As we have introduced, one of the objectives of the framework, and particularly the console, is to provide support and a framework to help different stakeholders understand, interact with, and adapt the Data Mesh.

The console should be a means to convey the concepts of the Data Mesh. This is a major challenge for us, especially with as ambitious as we intend to be with the Data Mesh.

Many concepts come into play: data product, data domain, data contract, polysemy, addressability, truthfulness, ownership, autonomy, etc.

Many questions arise: What are the interactions between the different concepts? Which component should carry which information? And so on.

In such a context, it is difficult to ensure that everyone shares a common minimum understanding and to minimize the risk of non-interoperability or misunderstanding between different teams. It is important to define clear and standardized interfaces and scopes to enable teams to understand and adapt strong proposals through feature requests.

For us, it was a natural choice to address these questions through the console by standardizing the definition of core Data Mesh concepts and their interactions, all translated into the interface.

Alchemesh: Core concepts modeling

The first step was to ensure we have a corresponds to what we defined during the MVP design phase. It is necessarily subject to change as we develop and implement new features 🔥.

Alchemesh console: Core concepts modeling

Users

Users are at the heart of the players who will interact in the mesh. In our framework, we distinguish between several personas:

- **Data product developer**: Considering a wide spectrum of skills and - from generalist developers with general programming skills to specialist data engineers.
- **Data product consumers**: Covers multiple roles that have one thing in common, they need to access and use data to do their job (e.g. data scientists, data analysts, application developers).
- **Data product owner**: Responsible for delivering and evangelizing successful data products for their specific domain.
- **Data platform developer**: Responsible for delivering the platform services as a product, with the best user experience.
- **Data platform owner**: Build an operate the data platform as well as you X, Data platform developers who work on the data product experience plane services.

Alchemesh users

Data domains

Domains data ownership is the foundation of the scale in a complex system like enterprises today. Scrum@ Scrum Scrum design OKR strategy embraces modeling based on multiple models each contextualized to a particular domain, called a bounded context.

A bounded context is "the delimited applicability of a particular model [that] gives team members a clear and shared understanding of what has to be consistent and what can develop independently".

We support 3 types of data domains:

- **Source aligned domain**: Analytical data reflecting the business facts generated by the operational systems, organized by domains, keeping the traits of their business domains as source-aligned domain data.
- **Aggregated domain**: Analytical data that is an aggregate of multiple upstream domains.
- **Consumer aligned domain**: Analytical data transformed to fit the needs of one or multiple specific use cases. This is also called fit for purpose.

In addition to clarifying the role of a domain in relation to the data products it owns, we need to ensure that the domain's responsibilities are clear. Data governance policies is a priority given the mesh is a - underlying is to ensure that decision data products from a source aligned domain that do not rely on any source system- or assist in prioritizing the reorganization of domains.

Data domain view

Technical teams

Depending on the size of certain data domains, an organization may decide to define multiple cross-functional teams to manage sets of data products. To address this need, we decided to have a concept of a technical team, bringing together people contributing to the same scope within a domain.

We distinguish between several kind of teams:

- **Data product team**: Stream aligned team, It is responsible for end to end delivery of activities (ingestion, consumption, discovery, etc) of a specific data product.
- **Platform team**: Stream aligned team, is to enable stream aligned to deliver their work with minimal interaction.
- **Governance group**: Enabling team, its key role is to facilitate decision making around global policies. These policies then get implemented computationally and adopted by the product teams.

Technical teams

Source system

In the case of source-aligned data domains, the operational and analytical worlds are unified within the same domain and this is reflected in the cross functional teams. It is important that the source system maintains this connection.

The intention is clearly not to manage operational tasks within the data mesh platform, but it is vital to maintain this connection to bridge the gap between the two worlds beyond just doing an organizational.

Data product

With domain ownership (supported by a technical team), the domain oriented data is shared as product directly with data users.

Data as a product introduces a new unit of logical architecture called data quanum, controlling and incorporating all the structural components needed to share data as a product.

By adopting a product approach, we will communicate the state of our offering:

- **Lifecycle state**: Refers to the product's lifecycle - whether it is in development, in discovery, stable, or being decommissioned.
- **Maturity level**: A product considered stable but with little historical usage data is a different proposition than one that has been used for years, has been and by many consumers for several years.

Input ports

In the context of source-aligned data products, data will need to be ingested from an operational system or made it available to input for the internal processing pipeline of the data product. This integration will be done via input ports (a platform component dedicated to this integration provided by the platform or implemented by the domain teams).

To give a concrete example, suppose that operational data is available in a Kafka topic, the input port could be a Kafka connector. In the case of a batch input could involve provisioning a AWS bucket and a AWS platform that ingests data from the Kafka topic.

Semantic model

On Mesh for semantic models up to data product we offer.

Structure and human readable model definition that captures the domain model of the data. Since the data product models the domain, what types of entities the data includes, the properties of the entities, etc.

Output ports

These models will be exposed as assets through an output port. Simply put, an output port is a port consisting of a storage connector (object storage, relational table, streaming topic, etc.) associated with a group that allows access via different protocols and languages (SQL, REST API, GraphQL, etc.).

One of our concerns on this matter is that an output port will not necessarily expose all the models managed by the data product.

Code

This is the core work of a data product developer, who is often two detached from the data that produces to impact data tools and characteristics. The data mesh places the code that creates the value of a data product at its centre, and that is naturally what we do. This logic allows data engineers to improve or generate the output assets.

In the data product, it is the responsibility of data product developers to properly define their data product, consume and expose data via the standard ports, and maintain the related artefacts.

In return, everything that happens inside (like code) is completely left to the discretion of the teams (Airflow? Big ol'A, see Airflow Mesh, Airflow DAG, a simple Python job in a Lambda… The data product owner responsibility lie with the owner (this is what we call autonomy).

Infrastructure

A data product may depend on infrastructure to be provisioned to carry out its processing, such as object storage, an intermediate dataset, etc., which are not related to how the code is executed, data is ingested, or data is exposed. This interface allows specifying to the platform what the data product needs for this.

Metadata

Lineal

We consider an asset to be the instantiation of a data product model (as an output port).

Since the data product is deployed and functional, the code must maintain certain status information to inform its consumers of its state:

- **Overall state**: operational, in incident, down
- **State of the assets**: their technical data quality (accuracy, completeness, timeliness, validity) and their freshness.

Data product

Data contract

We have our data product (or our data domain, owned by a technical team, or a platform component) that an operational system has ingested and exposing the output value of the data product generated by the code via output ports. *Lineal*

But before consuming this data product, I, as a consumer, would like to know what I am consuming (s), and as a producer, who contents to consuming from me? This is where data contracts come into play.

Output Port

A data contract applies to an output port of a data product, not to the entire data product. There are several reasons for this:

- Expectations differ between a streaming flow and an object stored in a data lake in terms of response time, update frequency, etc.
- Not all output ports carry the same models, or the commitment to consumption is not the same.

Access type

Depending on the nature of the data product, access to it will not be authorized in the same way. We have three types:

- **Restricted access**: This means that the owner of this data product must review and validate any access request.
- **Internal access**: This means that all requests from within the same domain are auto-approved, otherwise, they require the owner's validation.
- **Public access**: This means that all requests are automatically approved without review or validation from the owner.

Versioning and Lifecycle State

Data contracts are versioned and have a lifecycle state to inform about their status and to provide warnings in case of deprecation or changes.

Service Level Agreements

A data contract is a commitment to a service that we will provide and, more specifically, how we will provide it. Currently, we define the following commitments:

- Uptime
- Update frequency
- Response time

Terms

It is also a commitment on how the data product will be consumed in terms of:

- Usage
- Billing
- Notice period to adapt your consumption

Data quality tests

As you may have seen in the cases within the data product, we distinguish between **data quality tests that we call technical** and those **we call business**. The former has a purely technical meaning, regardless of consumer expectations, and is defined by the technical teams.

The latter, defined within a data contract, aim to have a business meaning that validates the value we introduce and commit to for consumers (a duplication of lines may have a technical impact on storage costs and compute time without necessarily impacting the value we deliver).

Status

The data contract is responsible for verifying its own state to allow the system to compare it with the commitments. It maintains the state of:

- SLAs
- Usage
- Billing
- Data quality test results

Data contract

Data contract access request

The data contract is ready, now, it's time to request access to subscribe to it! This is the role of the access request, which will include:

- **Who wants to consume?**: Data product, Technical team, Single user, or Data domain
- **What is the purpose?**

Access request

Platform components

I won't go into detail on this part, not because it's uninteresting, but because it deserves a dedicated article in my opinion.

The important thing here is that we want to use these resources to offer interfaces between data product developers and platform teams (Data Product Experience Plane and Infrastructure Utils Plane) to support the provision of a self-service platform, ensuring developer autonomy while offering decentralization through platform components implemented and provided by the platform (our famous LEGO).

Conclusion

There you have it — we've covered the core concepts that the console will support to enable teams to implement their data mesh. Let's not forget one thing: we are still at the very beginning of development, aiming for an MVP with the basic concepts to start introducing the data mesh! Many concepts essential for a data mesh at scale and in the long run, such as polysemes, feedback loops, computational policies, etc., are still missing. We'll get there!

The concepts are in place; the next step is the north star architecture of Alchemesh!

Data Mesh · Alchemesh · Data Contract · Data

52 · 1

Written by Alexandre Guitton

138 Followers · Editor for Alchemesh

Lead Data Ops @Doctolib

Following

More from Alexandre Guitton and Alchemesh

Alexandre Guitton

A Kubernetes Operator for Apache NiFi

Successfully manage your data processing at scale with NiFi on kubernetes using NiFiKop.

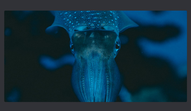
Nov 19, 2022 · 42

Alexandre Guitton

Alchemesh: Data mesh implementation on GCP (theory)

We're deploying our Data Mesh on GCP, using Dataplex for governance and Dagster...

2d ago · 52



Alexandre Guitton

[Notes 1/2] - Dagster: Data orchestrator hands-on

Follow this hands-on tutorial to implement your Job in a Data orchestrator with Dagster.

Mar 19, 2023 · 14 · 4

Alexandre Guitton in Dev Genius

[Part 0] - Journey to a modern data architecture (in 2022)

Start a journey with me to implement a modern data architecture in 2022.

Nov 24, 2022 · 62 · 1

See all from Alexandre Guitton · See all from Alchemesh

Recommended from Medium

Jagadesh Jamjala

Essential Data Engineering Tools in 2024

Are you keeping up with the latest tools in data space?

May 26 · 129 · 1

Shawn Gordon

What The Heck is Apache Polaris?

Introduction

Sep 12 · 7

Lists

General Coding Knowledge
20 stories · 1620 saves

data science and AI
40 stories · 256 saves

Predictive Modeling w/ Python
20 stories · 1574 saves

ChatGPT
21 stories · 821 saves

Pavan Emani in Art of Data Engineering

Mastering Architecture Diagrams and Technical Presentations: A Da...

How to create impactful technical presentations and diagrams that effectively...

Sep 20 · 110 · 1

Alireza Sadeghi

The History and Evolution of Open Table Formats

From Hive to High Performance: A Journey Through the Evolution of Data Managemen...

Aug 23 · 63

Joe Procopio in Entrepreneurship Handbook

The Slow, Painful Death Of Agile and Jira

Agile has stopped being agile, so it's time for Agile to go and take Jira with it

Sep 26 · 2.2K · 101

Ajay Parmar in Top Python Libraries

11-Python Packages That Turn Hundreds of Lines of Code into One

🚀 11 Python Libraries That Turn 100 Lines of Code Into One: A Fascinating Journey

Sep 28 · 565 · 4

See more recommendations