Building a Distributed Data Mesh from a Centralized Data Lake

In an endeavour to share data at scale, many enterprises are investing in their next-generation data lakes in the hopes that these efforts will potentially enable automated intelligent decision-making. The data lake architecture's typical failure modes in data platforms cause promises to be unfulfilled at scale. We need to forsake the centralised paradigm of a lake or its forerunner data warehouse in order to address these failure modes. We must embrace a different paradigm that is motivated by modern distributed architecture, wherein domains are regarded as first-class concerns, self-serve data infrastructure is designed using platform thinking, and data is regarded as a product.

The First Generation:

proprietary business intelligence and enterprise data warehouse platforms; solutions with high costs that have left businesses with high levels of technical debt; Technical debt results in an underappreciated beneficial impact on the business due to thousands of unmaintainable ETL tasks, tables, and reports that only a small group of specialist individuals understand.

The Second Generation:

Big data ecosystems that view a data lake as the panacea; intricate big data ecosystems and ongoing batch jobs managed by a centralised group of hyper-specialized data engineers have produced data lake monsters that, at best, have enabled a few isolated areas of R&D analytics; they have been over-promised and under-delivered.

Third or Current Generation:

With a modern twist toward (a) streaming for real-time data availability with architectures like Kappa, (b) combining batch and stream processing for data transformation with frameworks like Apache Beam, and (c) fully embracing cloud based managed services for storage, data pipeline execution engines, and machine learning platforms, the third and current generation of data platforms are more or less similar to the previous generation. It is clear that the third generation data platform is filling in some of the gaps left by the previous generations, including those related to real-time data analytics and the expense of operating big data infrastructure. However it suffers from many of the underlying characteristics that led to the failures of the previous generations.

**Issues with some of Architectures:**

**Monolithic Data Platform**

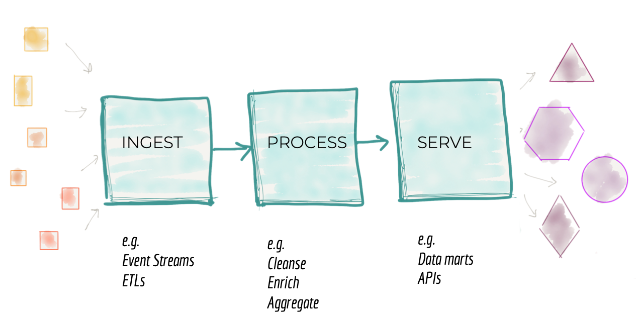
A picture containing diagram

Description automatically generated

While this centralized model can work for organizations that have a simpler domain with smaller number of diverse consumption cases, it fails for enterprises with rich domains, a large number of sources and a diverse set of consumers.

There are two pressure points on the architecture and the organizational structure of a centralized data platform that often lead to its failure:

**Pipeline decomposition**



This paradigm has an inherent drawback that hinders the delivery of features even though it adds some scale by assigning teams to various pipeline stages. To produce an independent feature or value, the pipeline's phases are highly coupled. It is broken down orthogonally to the direction of change.

**Silos Ownership**

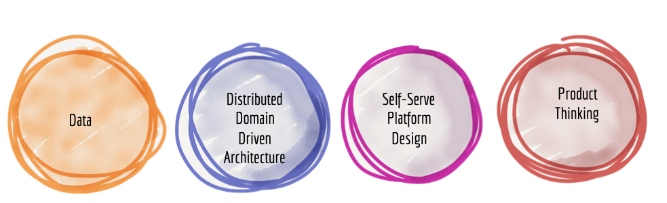
Text, whiteboard

Description automatically generated

The third failure mode of today's data platforms has to do with the organisation of the teams responsible for creating and maintaining the platform. When we focus in on the lives of those who create and manage data platforms, we discover a team of highly specialised data engineers who are isolated from the operational parts of the company, where the data is generated or used to drive decisions. The organisational structure of the data platform engineers is compartmentalised, but they are also divided into teams based on their technical proficiency with big data tooling, frequently lacking in business and domain understanding.

**Data Mesh : The Next Generation Data Platform Architecture**

The next enterprise data platform architecture is in the convergence of Distributed Domain Driven Architecture, Self-serve Platform Design, and Product Thinking with Data.

****

**Distributed pipelines**

**Diagram

Description automatically generated**

The requirement for preparing, collecting, and serving data persists, as does the necessity for data pipelines, even when ownership of the datasets has been transferred from the central platform to the domains. A data pipeline is treated internally within the data domain in this architecture and is only an internal complexity and implementation of the data domain. As a result, each domain will receive a distribution of the data pipeline steps.

For their domain events to be consumed by other domains without duplication of cleansing, the source domains must include cleansing, deduplication, and enrichment. Each domain dataset is required to specify Service Level Objectives for the accuracy and timeliness of the data it supplies.

**Data domain as a Product**

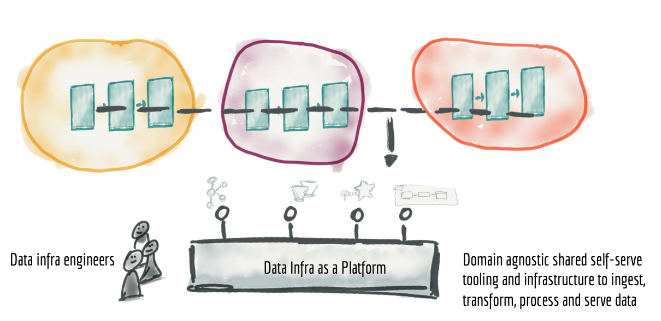
Diagram

Description automatically generated

The capabilities that operational domains offer to the rest of the company now incorporate product thinking. The rest of the organization's developers can use these capabilities, which are provided by domain teams as APIs, as building blocks to generate functionality and value of a higher level. For their domain APIs, the teams work to provide the greatest developer experience possible. This includes API test sandboxes, discoverable and intelligible API documentation, and constantly monitored quality and adoption KPIs.

Domain data teams must apply product thinking with the same rigour to the datasets they supply if they want a distributed data platform to succeed. They must think of their data assets as products and the rest of the organization's data scientists, ML experts, and data engineers as customers.

**Self-Serve Platform Design**



The duplication of labour and expertise needed to operate the data pipelines technology stack and infrastructure in each domain is one of the key issues with distributing the ownership of data to the domains. Fortunately, creating common infrastructure as a platform is a well-known and well-solved topic; nonetheless, it must be acknowledged that the data ecosystem's tooling and methodologies are not as advanced.

The necessity for duplicating the effort of setting up data pipeline engines, storage, and streaming infrastructure is eliminated by harvesting and extracting domain agnostic infrastructure features into a data infrastructure platform. In order for the domains to acquire, process, store, and deliver their data products, a data infrastructure team might own and offer the required technology.

**Data Mesh as a Platform**

Diagram

Description automatically generated

It took a while to read. Let's combine everything. We examined some of the fundamental traits of the present data platforms, including their centralization, monolithic structure, highly connected pipeline architecture, and silos of highly specialised data engineers operating them. We introduced distributed data products, which are the platform-level building blocks of a pervasive data mesh. These products are owned by independent cross-functional teams with embedded data engineers and product owners, and they use common data infrastructure as a platform to host, prepare, and serve their data assets.

The data mesh platform is a purposefully planned distributed data architecture that is supported by a shared, standardised self-serve data infrastructure and is governed centrally for interoperability. I hope it's obvious that it's not a landscape.