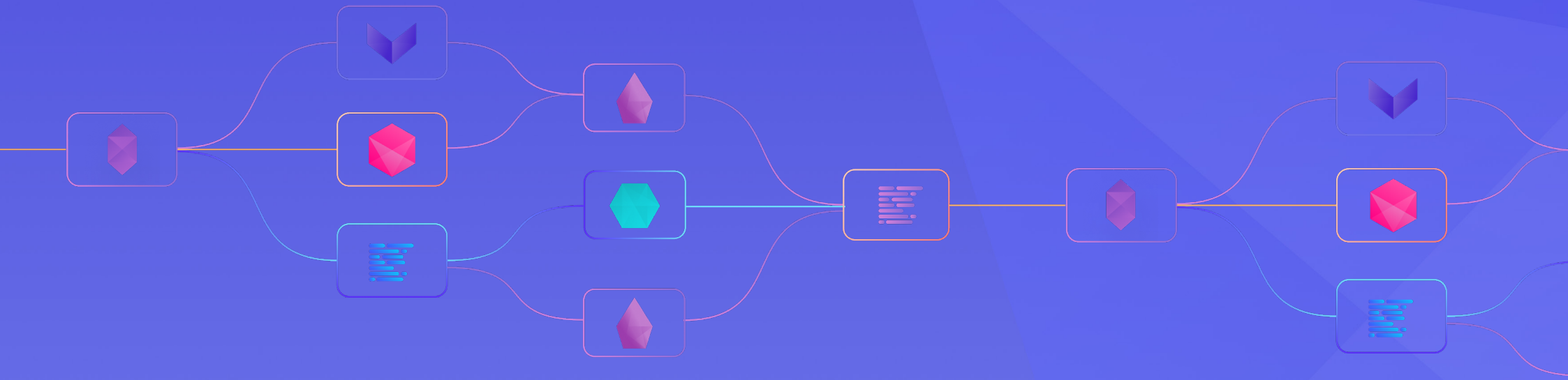
Implement Data Mesh with Self-Serve

Get business teams started with a self-serve platform to build data products with speed, standards & quality on the Lakehouse



Prophecy

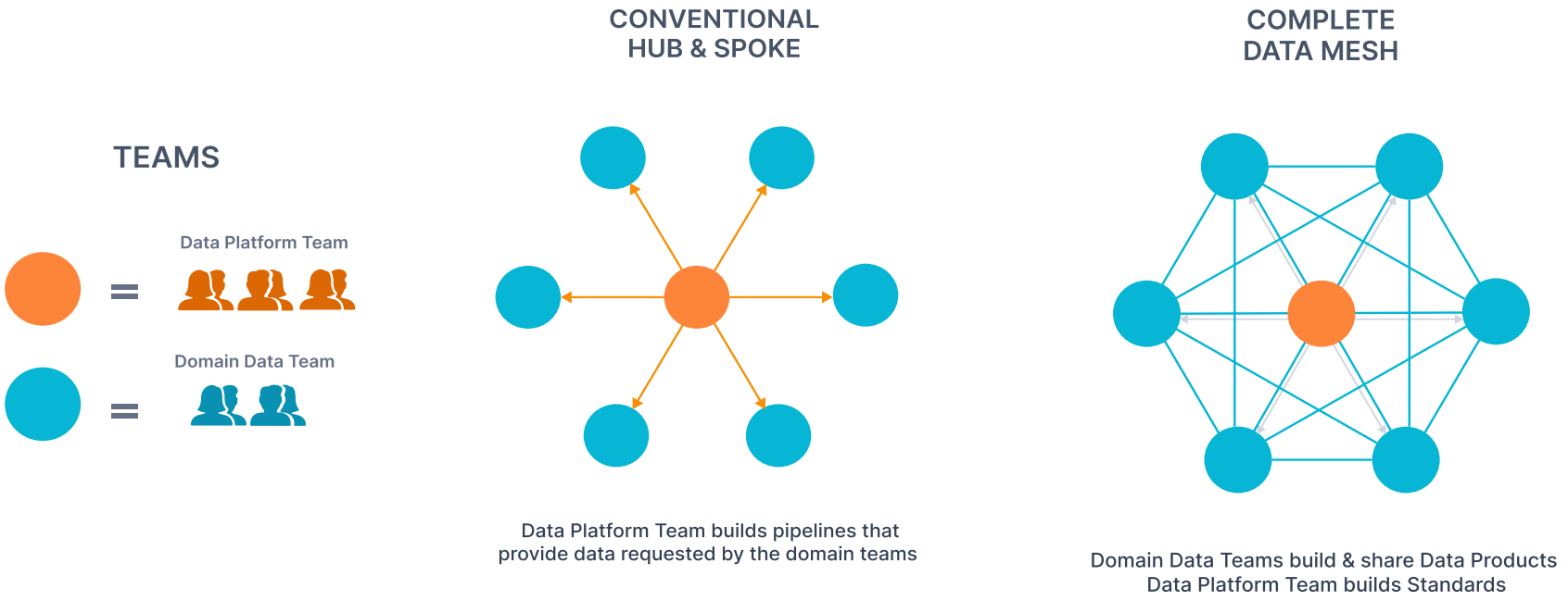
Author: **Raj Bains**

Introduction to Data Mesh

Analytics and Machine Learning promise to be the new source of value and strategic differentiation in the market. Most companies however, continue to find it hard to derive adequate value from their data - and the challenge of supplying analytics with relevant, high-quality and timely data has remained unsolved for over a decade. Data Mesh and other architectural patterns have emerged to solve this and hold promise.

Data mesh is fundamentally about putting more of the data into the hands of domain experts who understand the data, instead of completely relying on a single data platform team to clean, process, combine and summarize data for every domain team. There is clear business value to be derived from this change (described in the following sections), but the move in this direction involves organizational and technical changes. The leaders of companies are the experts in navigating their organization, so we'll focus on the technological pieces that enable this change.

Data mesh is described well as a high-level concept by Zhamak Dehghani in her blogs on Thoughtworks and her book Data Mesh. One should read this for a high-level understanding of the concept. We will focus on an interpretation that is much closer to the ground - a nuts and bolts approach that can be implemented in a few months. For the rest of this paper, we'll explain a more practical approach, the benefits and a step-by-step guide to achieve this.



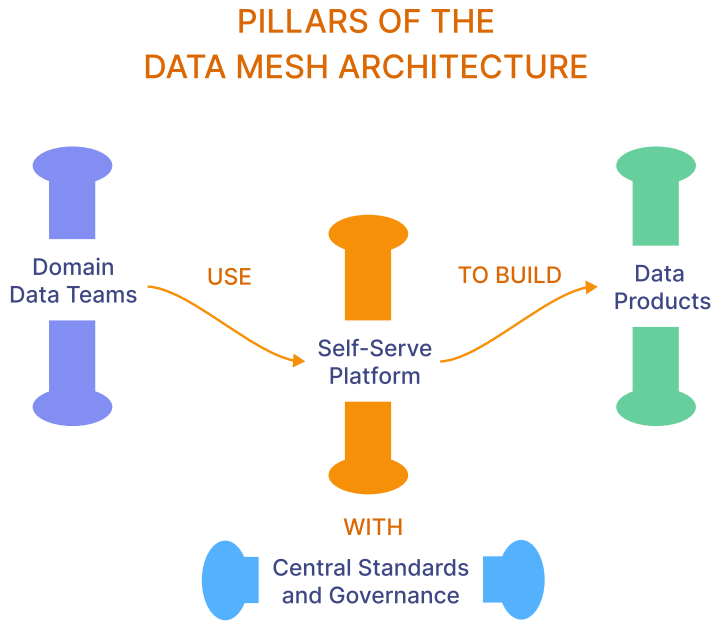
The Basics of the Data Mesh

The Data Mesh is focused on enabling the domain data teams to build their own data products from operational sources and publish them across the organization.

To do this, the Domain Data Teams need a Self-Serve Platform that meets them at their level of technical expertise, and enables them to build and publish Data Products, with support from a Data Platform team that provides them with Central Standards and Governance. These are therefore the core pillars of the Data Mesh Architecture.

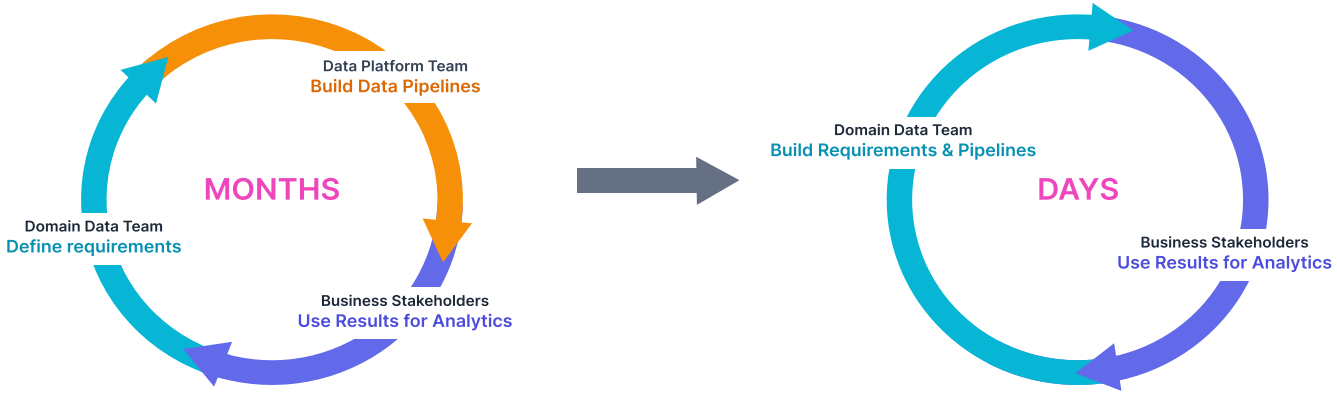
There are two kinds of changes:

- Organizational changes such as enablement, responsibility and accountability given to the domain data teams
- Technological changes such as a Self-Serve Platform with the accessbility and capabilities needed can be available as an off-the-shelf product.



The Value of Self-Serve

A self-serve platform is one that enables the domain data teams to build data pipelines & data products themselves without having the data platform team in the initial path. It requires the technology to meet the team at their current level of technical expertise. It puts the power in the hands of domain experts enabling them to iterate in days instead of months (that is the norm when the central data platform team is in the loop). This is the only path that is scalable and flexible enough to meet the promise of data.

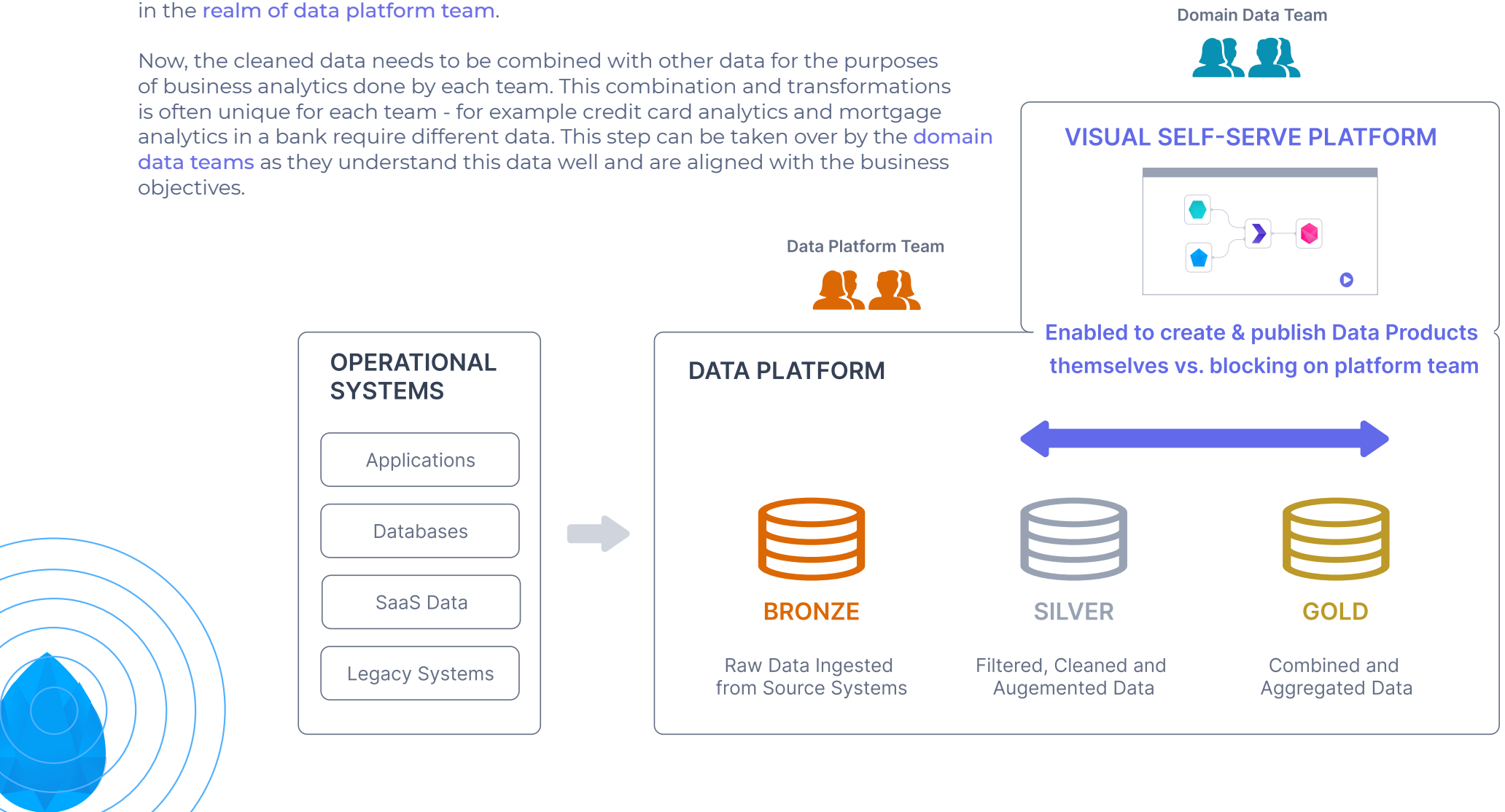


The Practical Data Mesh

Data Mesh is very abstractly defined in the community, that includes suggestions such as domain data teams setting up micro-services to publish data. This is aspirational and we'll take a more practical approach that can deliver much of the value quickly. Our customers often talk in terms of a more concrete implementation, where the domain data teams can build their own pipelines via a self-serve platform, and are able to publish their refined datasets for the rest of the organization to consume.

Many organizations talk in terms of Bronze, Silver and Gold dataset layers. The bronze layer has an exact copy of your data as it arrives from operational systems. The Silver layer starts with cleaning and augmenting the data. This can stay initially in the realm of data platform team.

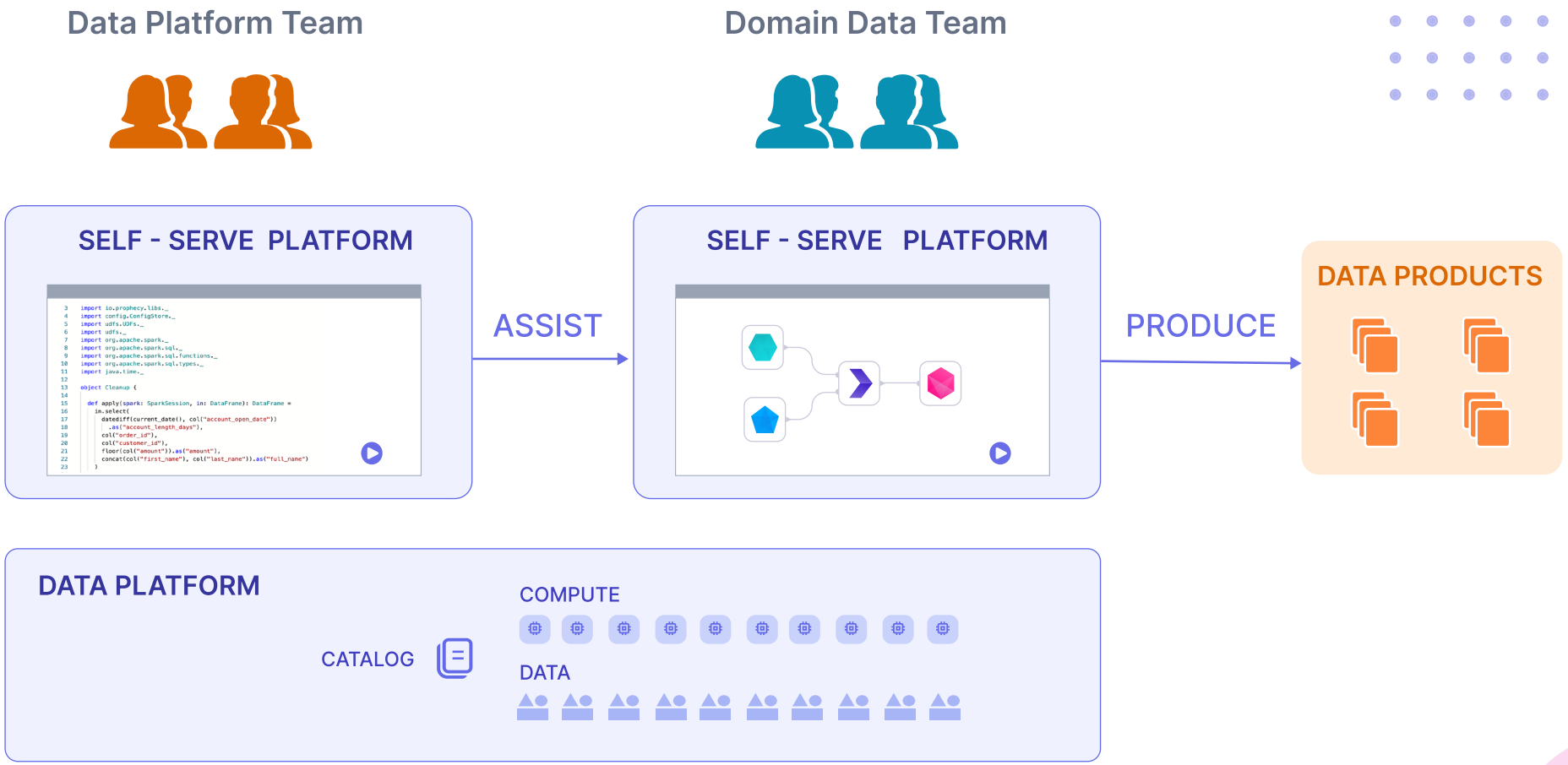
Now, the cleaned data needs to be combined with other data for the purposes of business analytics done by each team. This combination and transformations is often unique for each team - for example credit card analytics and mortgage analytics in a bank require different data. This step can be taken over by the domain data teams as they understand this data well and are aligned with the business objectives.



The Teams

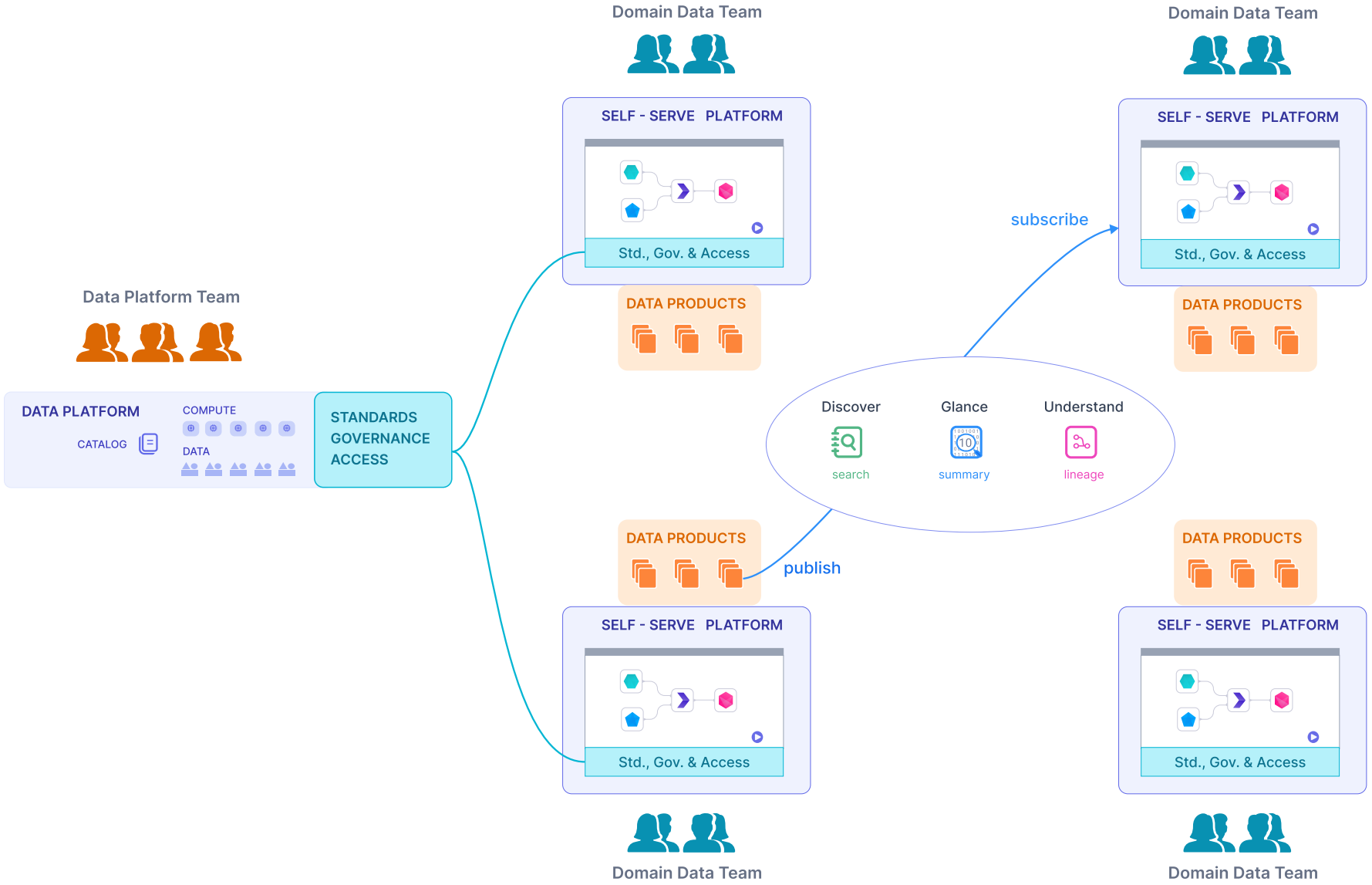
The Data Platform Team is often a separate team in larger organizations that is responsible for the underlying data platform and governance. This team also assists in developing standards and templates to ensure that the Domain Data Teams are following the best practices and are productive on the data platform.

The Domain Data Team is the team that uses a self-serve platform to build and consume data products.



Data Mesh in Action!

Here we show the primary actors and the systems participating in a data mesh - the roles and responsibilities of various teams, and the affordances that the self-serve platform must provide to enable these teams to achieve their goals.



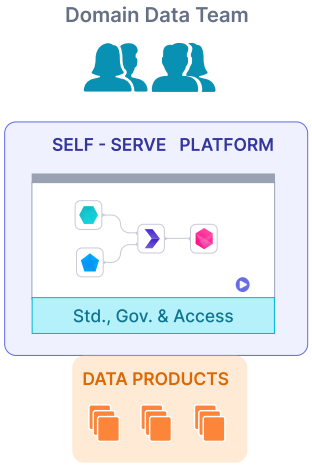
1. Domain Data Teams must own data products

The team that is closest to the business or the domain, such as a marketing team or a credit risk analytics team, is the right place for the ownership and production of high-quality data products for their business area. This has multiple reasons:

- This team has domain expertise and understands the meaning of the data the best, and will be the right steward of it.

- There is a strong alignment of incentives - the domain teams must meet business priorities, ensure fast time to market and are responsible for producing business value.

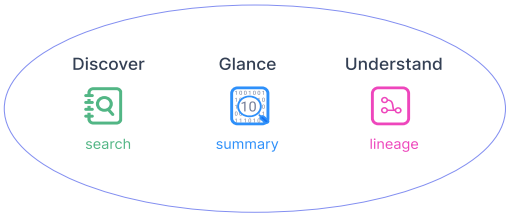
- Business priorities will change often and only the teams closest to the business can be flexible and adapt quickly to ensure alignment with the business value.



2. Self-Serve Platform to build, publish and find data products

Technology must be designed to make the business teams productive with their current skillsets, ensuring that the data products are built based on business knowledge and priorities rather than being centered around technical skills.

Obviously, we need a self-serve platform for the domain data teams that enables them to develop and manage data pipelines. The teams will combine and transform the incoming data to produce the datasets that capture the right business information. It must also be easy to put the data pipelines into production (possibly using templates provided by the data platform team) to keep the datasets updated on a regular cadence.



The datasets that are designed to be used by other teams (or are ready to be used by other members of the same team for analytics) should be published explicitly. How does the user evaluate if this dataset is right for them? The user should be able to find and visit the dataset page and evaluate the fitness of these datasets.

3. Standards by the Data Platform Team

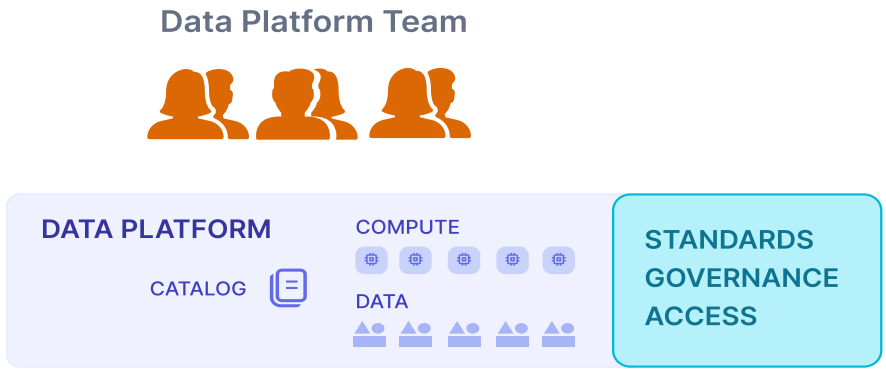
The data platform team can be a key enabler of the various domain data teams, not be metaphorically catching the fish - but by providing the tools to the domain data teams to fish themselves. The data platform team can help in the following ways:

- Create standards for development, such as the standard visual components to be used in the self-serve platform for encryption and anonymization of the data, and publish these standards to the domain teams.

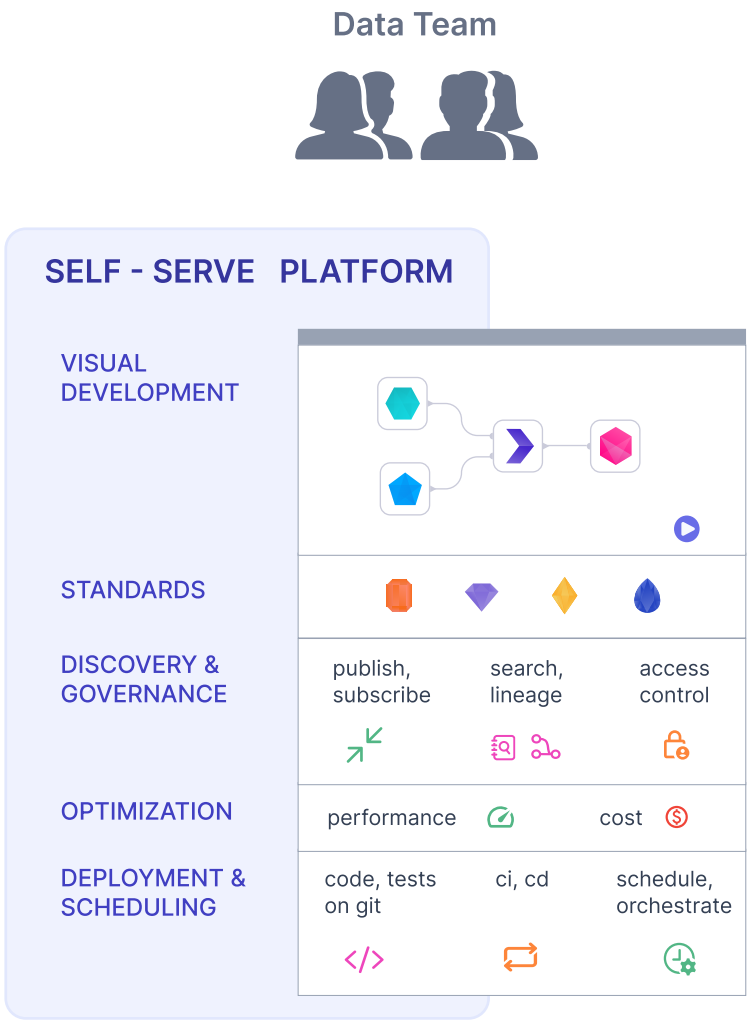
- Create pipeline templates guiding best practices like authorization, notifications, error handling and logging.

- The performance and cost of data pipelines must also be made explicit to the user. The central team can define and share performance guidelines and create a plan to handle exceptions. This enables the business team to work with the data platform team when they are exceeding limits or need help with performance.

The self-serve platform has to enable the data platform teams to create and publish these standards to the various teams. They will have to write code for creating the standards, but the use of the standards must be done visually and simply.



Self-Serve via Low-Code



A self-service platform provides capabilities to both domain data teams and the data platform team. It enables the domain data team to become self-sufficient and succeed in rapidly building, publishing and consuming data products -

- **Low-Code Development** - Visual development has some obvious benefits and some more subtle ones. Here are the primary things to consider:

  - Visually developed data pipelines can produce the same quality of code as the top data engineers (Prophecy can be reached for a deeper dive here). The visual pipelines make common tasks such as reading from sources, doing common transforms and inserting and merging new data with existing data simple.

  - Visual development enables the business to access a talent pool that is at least 10 times larger and includes the data practitioners already in the business teams.

  - Visual development enables every data practitioner to be at least 10 times more productive than when programming the same data pipeline since it is easy to see data structure and samples after each step.

- **Visual Development Standards** - Standards are reusable business or execution process logic that is shared across teams. The self-serve platform must come with built-in standards that cover common cases, and enable extensions - the creation and reuse of user defined standards for business specific cases.

- **Deployment & Scheduling** - Once the data pipeline is built, it needs to be deployed to production and run regularly to keep the data up to date. This needs a scheduled run to designed visually that might include running one or more pipelines at a certain cadence. For releasing changes from development to production, we should use the techniques built by software engineers that include - development with git for versioned source code, test with high coverage, and continuous integration & deployment.

- **Discovery and Governance** - Domain data teams need to produce and share datasets, and they need to consume datasets produced by other teams. This requires affordances in the underlying platform to publish & subscribe to datasets, and to search, and find correct datasets. When searching for datasets, there needs to be the information required to evaluate which datasets are the right ones to consume. This evaluation requires information along multiple dimensions - the structure, quality, freshness of the dataset and number of other consumers. This information is stored in the low-code platform as well.

- **Optimization and Cost Management** - As the business consumption of the underlying data platforms such as Spark and Data Warehouses increases, the performance and its most impactful consequence - the cost - must be managed. Making the cost of producing every dataset explicit during the development of data products is a first good step. Next, a data platform team might have experts who can look at the few troublesome pipelines and see if it can be helped. Ideally, there will be guidance in the self-serve platform as well, but we must walk before we can run.

- **Infrastructure Setup** - The infrastructure of the data platform and the self-serve platform is done by the data platform team.

The self-serve platform adds a large number of features and functionalities that are not present in the underlying data platforms and are essential to making a self-serve team succeed and be more productive on your existing cloud data platforms - Apache Spark platforms such as Databricks, and Cloud Data Warehouses such as Databricks, Google BigQuery or Snowflake.

Build Data Products - Development & Deployment

Building Data Products is the source of value to the business. We should enable the largest number of data users to build data products - and build them quickly. These are the primary affordances provided by the self-serve platform

- Visual drag-and-drop data pipelines should be used to create data products. This only requires some SQL or Excel level skills to write expressions, and enables the largest set of users to build data products. Since the users can see the results after every step - interactively running the pipeline after each change - building working pipelines is quick.

- Visual pipelines are built out of standard visual-blocks that are known to be high quality and run at scale - so that another team or user does not ever have to rewrite these pipelines. The data pipeline development and execution happens inside the data platforms such as Spark or a Data Warehouse - not in the tool.

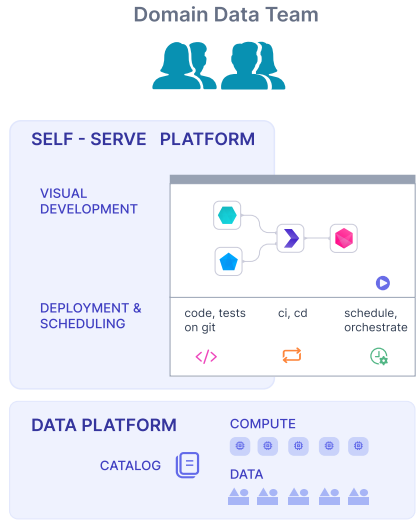
- The data users must be able to schedule the pipelines visually, so that they run regularly and keep the data up to date. This might require running multiple connected pipelines in order, where some pipelines consume the results from the previous ones.

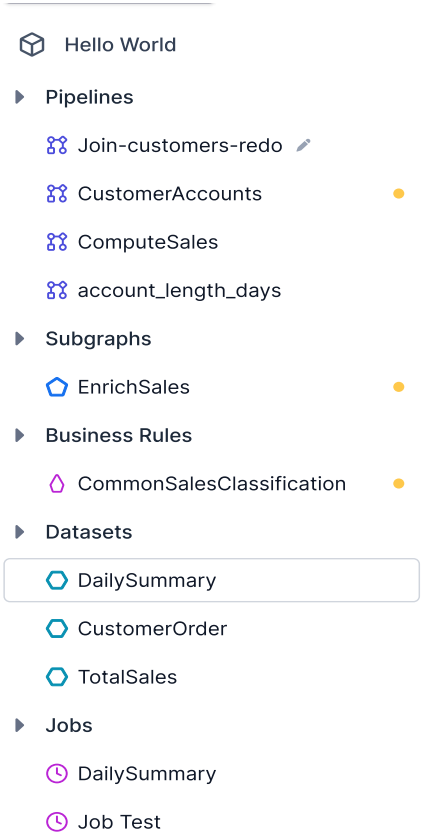
- There must be a robust development process, and this is a problem software engineering has cracked. Let's look at this in the context of data pipelines:

  - The visual pipelines should turn to high-quality code on git (such as GitHub)

  - The data users must develop pipelines add tests in their git branch, and when ready - commit their change to git main. This ensures that for every subsequent change, all the tests are run and what's working once is not broken by later changes. It also ensures that if a change introduces an error, one can go back to previous working version immediately, and not break other consumers of data.

  - The pipelines in production should have monitoring and alerting built-in to enable a dashboard to check if all is well, and inform the user if an error has occurred with some guidance on how to fix it.





Data Projects to package & manage data work

Data Projects are the physical representation of the work done by the domain data team. So, data project is the context in which the self-serve product must provide all the capabilities to make the data mesh work. Projects must be the primary unit in the Low-Code environment for developing data products, and should be reflected in code on Git, so as to be the source of truth. A data project contains:

- Data Pipelines or Data Models that contain the business logic to transform data. These are built visually and stored as 100% open-source code in Spark (Python / Scala) or SQL formats.

- Subgraphs and Business rules are the reusable pieces of business logic that can be reused within a project or across projects

- Datasets are the intermediate (internal) and final data (published) produced by the project

- Jobs are the scheduled runs that run the data pipelines regularly to ensure that the datasets are kept up to date with newly arriving data it.

Publish Data Products from Projects

Once a domain data team has built a data product, this must become the source of truth for the organization. This will avoid other teams recomputing the same information - spending extra development time, expensive cloud resources and the likelihood of mistakes since other teams will often not be expert in this domain. Unfortunately, recomputing the same data products with low-quality is the norm in the industry.

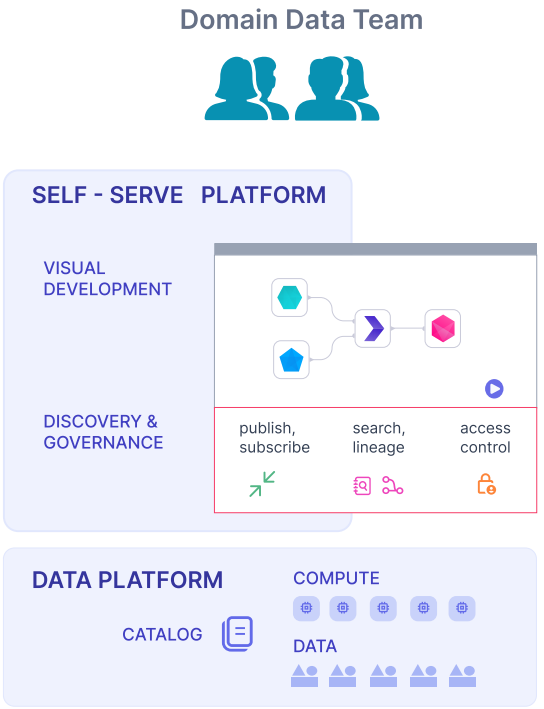
A data project can contain many data products - datasets, transformations, business rules and perhaps reports and dashboards - some for internal use and as the final results to be published. We will focus primarily on the datasets, but the concepts will apply to all data products.

For the publisher of the data product, the following affordances must be available

- There must the ability to mark a dataset as published, at which time it becomes searchable by other data projects. If another data project subscribes to this dataset and becomes a downstream user, there must be the ability to approve or deny the subscription request.

- The producer must be able to add other information to the dataset page in the low-code providing information that helps the downstream consumers evaluate the fitness of this dataset. This information will go from the basics - including the structure of data and its freshness, to things that help evaluate the dataset for usage and includes data quality and the meaning of each column (how each column was computed usually provided by column-level lineage with an additional description field for further guidance)

- The producer needs the information to handle change management. Say, an existing column in a dataset is to be modified or dropped, she'll want to know which consumers are dependent on it, and how they're using the column. Based on the information about the use of the column, it might be ok to change the column or add a second modified column with a nudge to downstream users to migrate. This kind of impact analysis is critical to avoiding breaking downstream consumers and causing outages in the data factory.



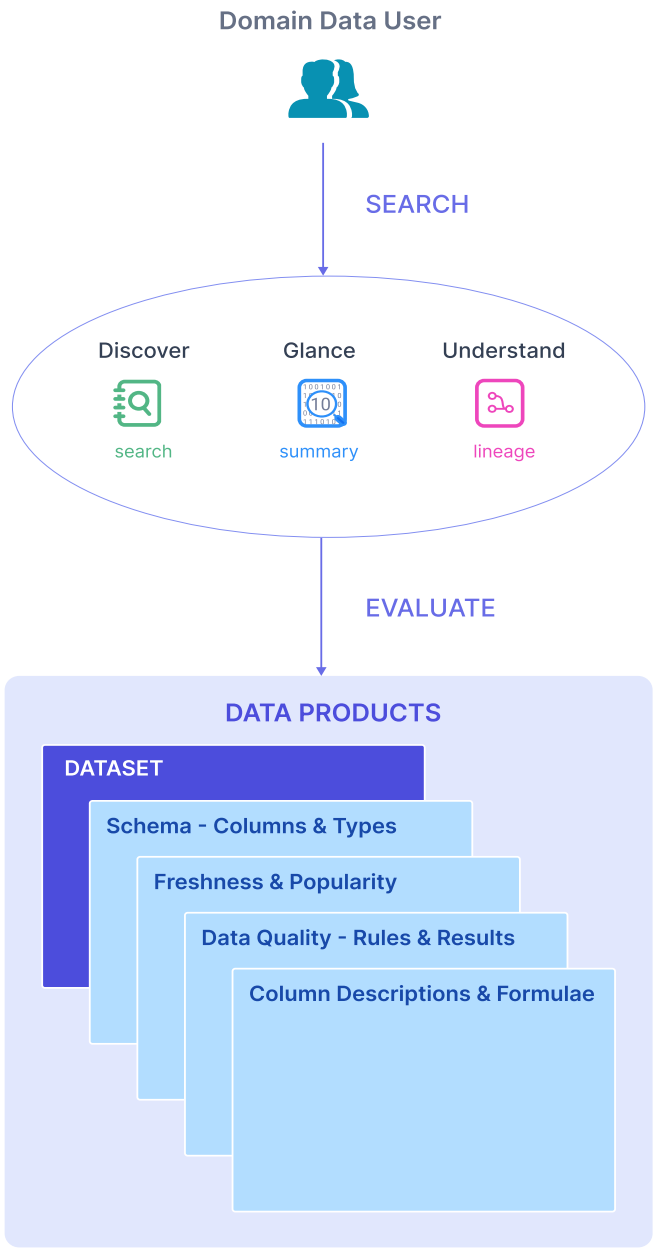
Search, Evaluate and Subscribe to Data Products

To get high reuse of datasets (or other data products) across the organization, to minimize duplication, minimize effort and cost, and to ensure that the datasets used are the highest quality, there should be a high degree of reuse. Whenever possible, the first instinct has to be - has someone else computed the dataset that I want already. This will only happen if other datasets are made easy to find, easy to evaluate, easy to use and do not break my pipelines over time.

Search has to give you a summary view of all the published datasets that might meet the criteria, with rankings that put highly used datasets with high quality scores on the top.

Once a dataset has been identified, the user has to be able to evaluate the fitness across multiple dimensions. These include the basic structure of data, freshness, popularity, and quality. The data quality can be understood by looking at the rules that run on the data product and how well the data product does on these rules over time. Finally the consumer has to understand the meaning of every single column - using a mixture of producer supplied description and the auto-evaluated formula coming from the column level lineage.

If the dataset seems fit, the user can subscribe to it, and it should inform the producer that a new user has started using the dataset. This will ensure the producer will take into account the consumers when making changes. This will ensure that downstream pipelines are not broken when the upstream producer makes changes.



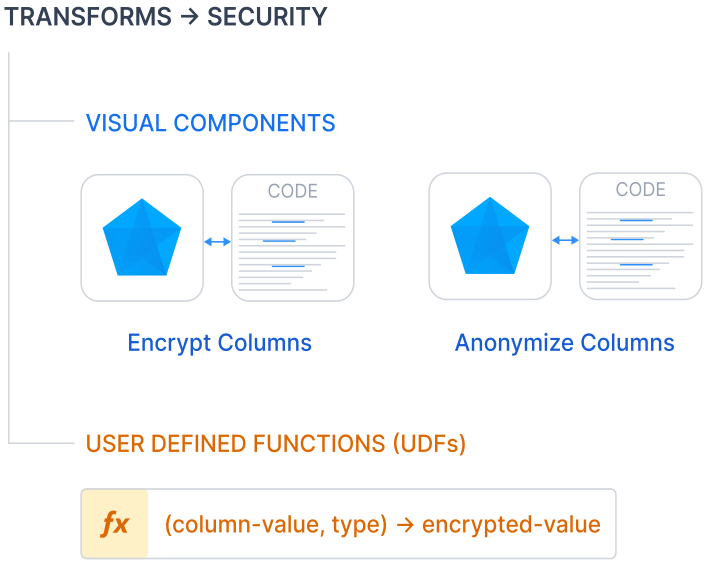
Build & Publish Standards

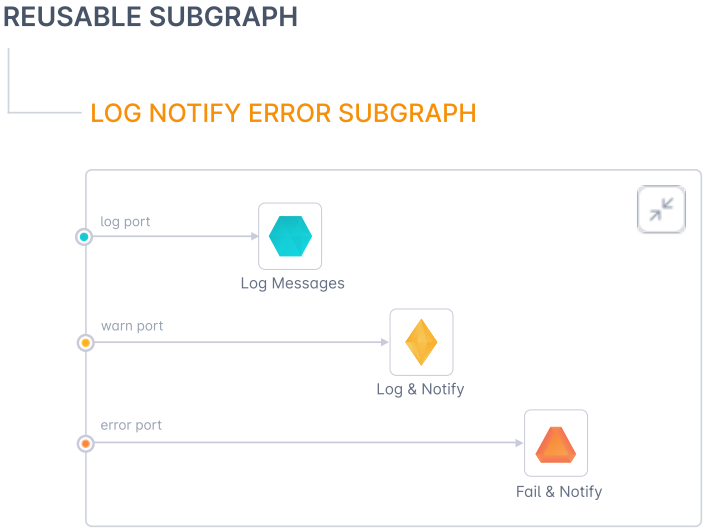
Data platform teams will often want to build utilities and standards to help teams make progress faster and for standards across organizations. The domain teams can build reusable business logic as well. These reusable pieces can be packaged into a library and published across the organization for multiple teams to use. Here are some common ones:

- Standardization of commonly used primitives is very important and these for the basic blocks used across a very high number of pipelines. Here, there are often new visual components that can be created - for example on the right, we can see Encryption & Anonymization components that apply these operations to multiple columns in a dataset. These visual components are associated with templated code where some values are inserted from the UI. Also, there can be user defined functions (udfs) that operate at one column at a time and can be used in an expression.

- Within the business, there will be commonly used terminology, such as the standard classification of customers into sales regions that all the analyses should share so that when different teams look at charts and graphs about sales, the numbers agree. These can be transforms written by the domain team and shared for multiple teams to use as reusable business rules.

- Some standards also need to be set often for how the data pipelines are operationalized, how error handling will be done and if the pipeline fails in production who must be notified and what is the severity of the event and the expected SLA for getting the pipeline back to running state. Such operational logic can be packaged as reusable subgraph that is made part of the standard template that business teams start with when developing a new pipeline.

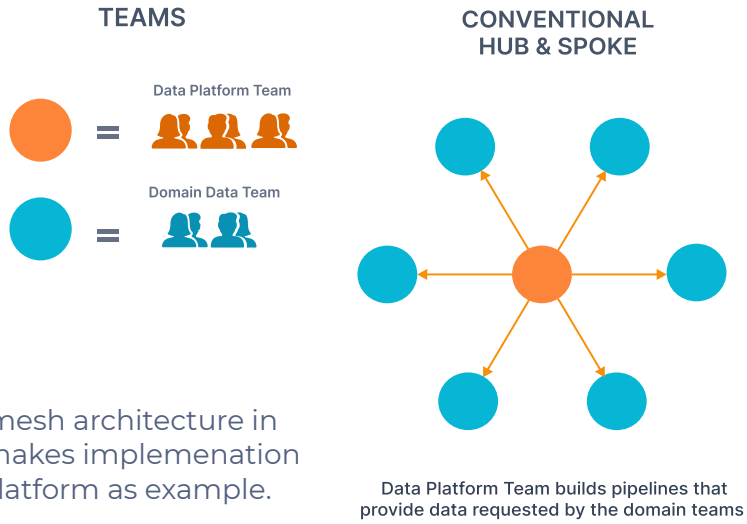




Data Mesh Implementation

Most organizations start from a hub-and-spoke model where the hub is represented by the data platform team. This team has the skills to build data pipelines, and produces the data that is consumed by the line of business or domain data teams.

Often this data cannot be called a Data Product since it lacks the necessary information to be consumed as a product - it does not come with a freshness or quality report, it lacks the definitions and formulae required to understand the columns and other information that makes the data usable off the shelf.



Data Platform Team builds pipelines that provide data requested by the domain teams

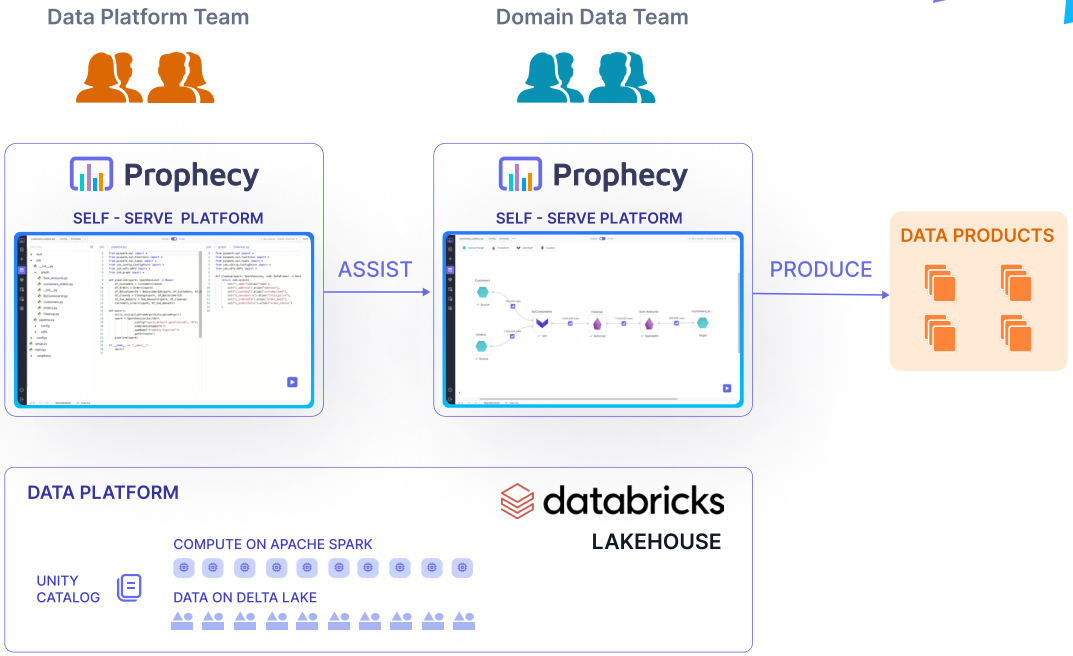
Let's look at the steps required to move from this architecture toward a data mesh architecture in small manageable steps. The largest missing piece is having a platform that makes implemenation of data mesh easy. So, let's make things more concrete by picking a specific platform as example.

1. Choosing the Self-Serve Platform and the Data Platform

The platform we are choosing includes:

- Prophecy as the Self-Serve Platform that meets the line of business users where they are - making them productive with their skillset and enabling them to focus on deriving business value rather than focusing on technology. Prophecy has the mechanisms required to make the data mesh work.

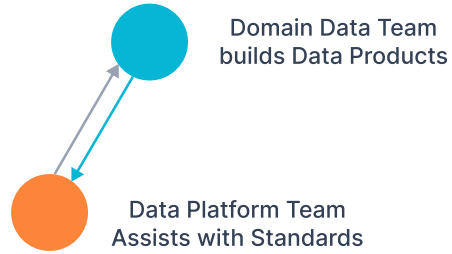
- Databricks Lakehouse as the Data Platform since it can run at scale required for the most demanding Enterprise workloads, supports data pipelines and analytics whether you're doing business intelligence or machine learning. With Databricks, you cannot go wrong and in the long term the platform will never be found wanting.



2. Transition the First Team to Self-Serve

We must identify the teams and the work for the transition. Following are the steps

FIRST TEAM TO SELF-SERVE



- Identify a line of business (LOB) team that is a good candidate to be the first team to use the self-serve platform to build data pipelines and data products.

- Identify a project that has clear and limited scope, perhaps one that was a candidate to be moved from this team to the data platform team for implementation. Now, this team will do it themselves. Ideally, this project can be finished in a few weeks.

- The data platform team should identify members that are dedicated to helping with this project for the duration of a few weeks. This team will work with Prophecy and Databricks to ensure that the platforms are setup for the line of business team to use. This team will also be available to assist the domain data team.

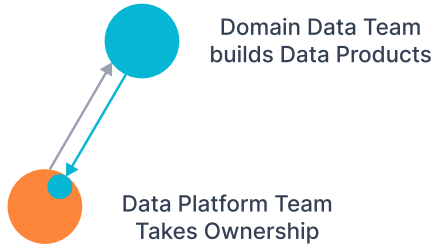
- Onboarding the LOB team where each member will build the first pipeline with guidance from Prophecy and data platform team - this is usually a couple of days.

- The LOB team members will build data pipelines themselves to produce the data products using the visual drag-and-drop interface in Prophecy, and during this the data platform team will be available to assist. As an example, if a common transformation or connector is needed in addition to the standard connectors and transforms already available, the data platform team can quickly build that by writing Spark code for a new visual component (gem) in Prophecy. The Prophecy team will be present and can assist or play the part of data platform team if needed.

- The LOB team members will deploy pipelines to production & publish datasets, deciding on the standard process to do this. Along with the central data platform team, they can figure out the permissions and the process for maintainence of these data products.

3. Transition reveals unresolved issues

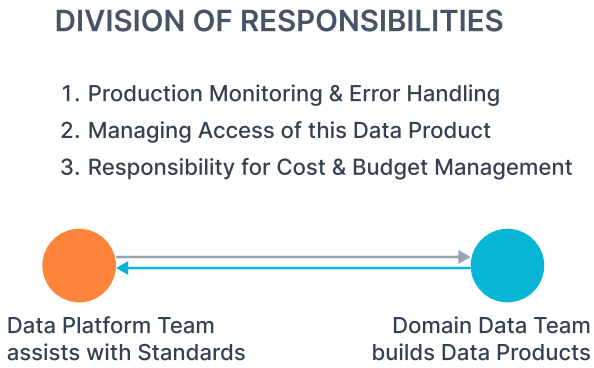
PROXY HELP FROM DATA PLATFORM



As soon as you put the first data pipelines into production, that are producing data products that might be shared across the organization, a set of unresolved issues will show up. There will be responsibility of operations, of budget, of access and a few other things where it will not be clear who the owner should be. For some of these issues a pattern is available where the data platform team can take responsibilities that might move to domain teams later, but that makes the transition faster.

Some common decision points are:

- Production monitoring and error handling - Let's say that there is a data project that is now owned by the domain data team who built a data pipeline visually, and scheduled it to run every day in the morning. The resulting dataset will now be used by this team and soon other teams. There are also quality rules and results added that show the fitness of the dataset. Here are the scenarios to solve for



- Pipeline Job Failure - If the pipeline processing job fails in production (perhaps some upstream data that it was relying on, changed or wasn't produced), then who will handle this scenario? It requires certain amount of production support that cannot be built into each line of business team. Perhaps the data platform team can be the first team that gets notified, and if there is indeed an error in the business logic - only then the domain data team comes into the picture. This will require a template from the data platform team to be included in each pipeline (see Build & Publish Standards section above), so this can be done in a standard way.
- Soft Failure (Quality) - If the data quality of the pipeline falls below the standards set by the domain team - this needs to be looked at (perhaps upstream data has a new pattern not seen before). However, tthis does not need to stop the processing and there must be an effective way for the domain team to be notified and for them to be able to promptly put a fix in.

- Managing Access - The central data platform team usually owns access. Here, there may be role based access control or attribute based access control used in an organization. Whenever a teams asks to subscribe to a data product, the central team can get the approval request. If the company is using an attribute based access control system, then the task is simpler. An example is that when a data product is moved to production and published, it goes through a process where the central team ensures that it has the right tags - such as PII tags on columns to indicate that a column includes personally identifying information, and ContainsPII tag on the dataset. Then rest of the access control can be taken care of by rules applied based on tags.

- Cost & Budget Management - The first step here is that the self-serve / low-code platform that you are using mush assign a cost to pipeline development and every pipeline run. This cost can be precise in machine units (such as Databricks DBUs), but approximate in dollar terms (it will be subject to contracts and discounts). If the cost associated with each pipeline and dataset is shown in the platform and rolled up to projects and teams - the hard part has been done by the product. Now, which team is responsible for the budget is more of an organizational question.

Other such issues will be revealed during this process and each company has to ensure that they come out of here with decisions and process largely agreed upon, and a few issues in the list to be handled later. However, with good progress here we should be ready to expand the usage to more teams.

4. Expand Self-Serve to Second Team

The goal here is to ensure that the playbook created with the first team can be applied, and to establish a producer-consumer relationship between two domain teams that is independent of the data platform team.

Here, we will establish the process by which the data product published by the first team can be discovered and subscribed to by the second team. The datasets published in Prophecy will be visible to the second team and will be read-only for them. They will be able to search and subscribe to the datasets.

5. Unlock Complete Data Mesh

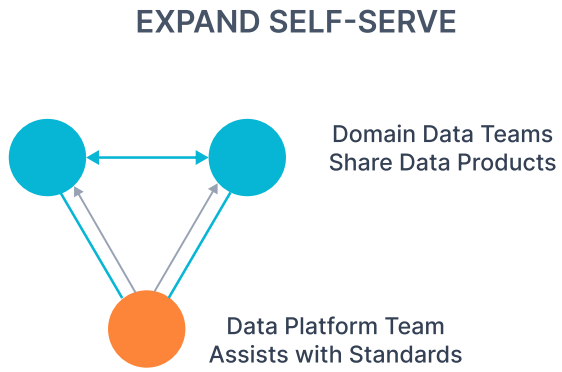
The move to complete data mesh is now unlocked - the process is working and the playbook is in place. However, different LOB teams will have their own priorities and deliverables and often will move gradually to the data mesh architecture as each team sees the benefits and witnesses other teams succeeding. Remember that one can accelerate the transition and work around this by the central data platform team acting as the proxy - taking ownership over from the LOB team for some of the tasks.

As a summary, the key points to review in this process are:

- The process is gradual and the different pace will work for each organization

- The process is not one-size fits-all and allows some teams to completely own data products, and others to lean on the data platform team temporarily (or permanently) once data products are in production.

- The business value will start accruing as soon as the first team has moved to the new process accelerating the analytics in the team that transitions. This means that the process enables the driving executive to get a quick first win and show value. This will make it much simpler to show an example to the rest of the organization and make it easier to get more teams on board.



Domain Data Teams build & share Data Products
Data Platform Team builds Standards

A word of caution is that the first project will build a new muscle for the organization. To ensure success one should keep the scope of first project small, keep the timeline short, and have executive oversight. It is also important that adequate resources are committed so that the project can be a quick win. This will enable the organization to get into a positive cycle to iteratively move further toward a complete data mesh.

We showed why a data mesh is a good architecture for many organizations, the benefits it provides and a very practical path to gradually adopt this architecture, getting incremental value at each step.

How can I try Prophecy?

Prophecy makes it easy to try and use the product, here are some options:
- Go to https://app.prophecy.io and sign up to run Prophecy with your Databricks account or Enterprise Trial on Prophecy account
- Go to your Databricks UI and use Partner Connect for a single-click setup of Prophecy with your Databricks account.
- Search Prophecy on cloud marketplaces, you can use Prophecy as Public or Private SaaS as a free trial before buying.

SIGN UP FOR A DEMO

Wherever you are in your cloud journey,
we're the product partner for your success!

Prophecy