Whitepaper

Data Mesh on **Databricks**

Get more out of distributed data with improved transparency, end-to-end compliance, flexibility & independence, faster access, and accurate data delivery.

Author:

Abhishek Patel

Head of Databricks Center of Excellence - LTIMindtree

Table of Contents

Introduction

Data warehouse, though dominant analytical paradigm, often creates challenges due to data volume, velocity, and variety. Centralized IT teams tend to become bottlenecks trying to cope with various business units' data hosting and processing demands. Often, this leads to delay in go-to-market strategy and stale analytical product due to the time it takes for the IT team to understand and develop the analytical solution. To mitigate these challenges and accelerate the creation and sharing of data-as-a- product, organizations are looking for a shift in how analytical platforms are built, served, and governed.

The answer to the many issues and roadblocks in the traditional organizational setup and the monolithic architectures is addressed by a Data Mesh that focuses on building modern distributed architecture at scale by paving the way for modern architecture designed for scalability and flexibility and federated computational governance

What is Data Mesh?

The concept of Data Mesh, a term coined by Zhamak Dehghani, encompasses data, technology, processes, and organization. On a conceptual level, it's a democratized approach to managing data where various domains operationalize their data, relieving the Central IT team from designing and developing data products. Instead, IT teams focus on providing and governing IT resources using a self-service platform.

Data Mesh challenges the idea of conventional centralization of data. Rather than looking at data as one huge repository, Data Mesh considers the decomposition of independent data products. This shift is backed by a modern and self-service data platform, which is typically designed using cloud-native technologies from centralized to federated ownership. Overall, Data Mesh implementation is managed by federated computational governance through catalogs and policies.

Data Mesh Architecture Principles

Borrowing the concept from distributed computing, Data Mesh focuses on disseminated data products aligned around enterprise domains and possessed by cross-functional teams with data engineers and product owners, utilizing standard foundational infrastructure as a platform to host, prep, and serve their data-driven information.

Data Mesh architecture adheres to 4 principles. They are -

01

Decentralized data domain

Fundamental units that represent each functional domain and host multiple data products about that functional domain. Domains are provisioned and governed by a centralized IT team but owned and maintained by individual domain-specific teams.

02

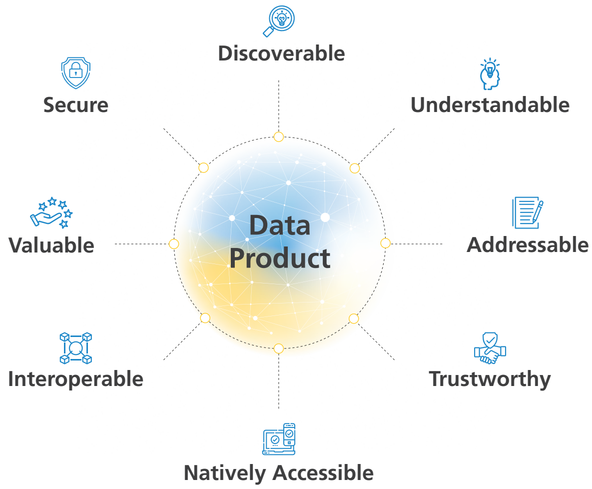
Self-service data infrastructure platform

This platform is implemented using infrastructureasacode, provisioning fast yet consistent domains and required services using best practices and best-of-breed technologies.

03

Data as a product

Data Mesh approaches data as an asset and offers the same as a product. Data Product works on logical architecture called a data quantum that controls and encapsulates all the structural components to share data as a product - data, metadata, code, policies, etc. Data Product should abide by the below-mentioned baseline usability attributes.



04

Federated computational governance

Due to the decentralized nature of Data Mesh, it is of prime importance to impose consistent and computational governance. It maintains required standards across different data domains and is required to be centrally managed and governed.

Data Mesh Architecture Pattern

Data Mesh, in essence, is an architecture paradigm where each functional data domain is represented as nodes and is interconnected, managed, and governed by a centralized IT/Governance node. Each data domain can host multiple data products that can be shared across different data domains using the same centralized IT/governance mode. The conceptual architecture of a Data Mesh is depicted as shown below.

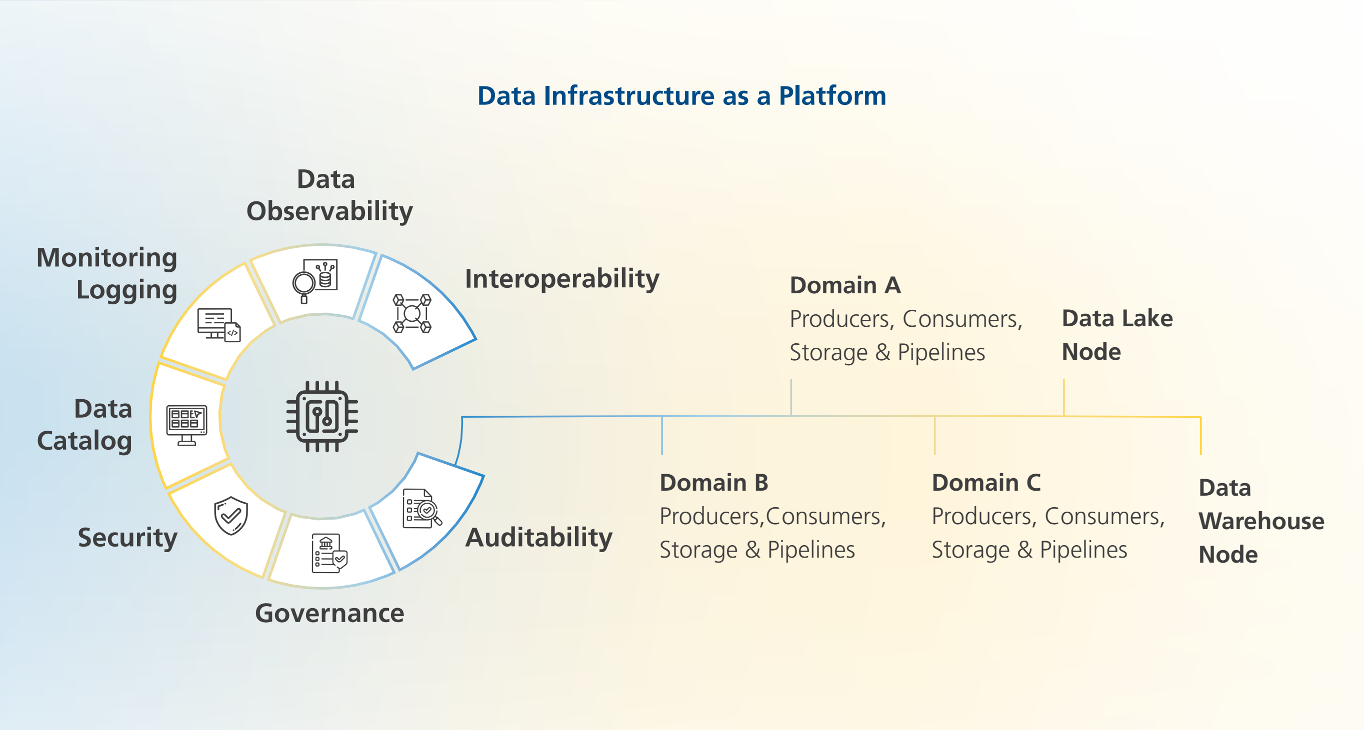


Figure 2 - Data Mesh Architecture Pattern

Preparing For a Data Mesh?

Although a technical concept, Data Mesh demands a change in work culture and a different perspective to observe and share the data. Hence, Data Mesh implementation needs to be approached from technical and behavioral aspects. Before defining any technical architecture or framework, it is essential to visualize the organization as different functional domains to define and fulfill the first principle of Data Mesh, i.e., domain oriented decentralized data ownership. Data Mesh can be approached in below linear steps.

01

Define personas

Data Mesh implementation needs different teams to collaborate and collectively owns the architecture. And hence, having clearly define personas and responsibilities of each persona is critical for the consistency of Data Mesh architecture. Mainly there are four personas. Domain Provisioner, Domain Owner, Developer, and Consumer. Each of these personas is explained in detail in subsequent sections.

02

Identify and leverage DataOps platform

For the consistent and accelerated implementation of Data Mesh, a well-established and approved DataOps platform needs to be deployed and leveraged e.g., dbt. DataOps platform should provide a consistent methodology for developing, deploying, and testing data products.

03

Develop data marketplace

Well-defined data marketplace can help easy publishing and consumption of data products. Data consumers can browse, preview, and subscribe for the data products they are interested in using a common marketplace.

04

Develop self-service infrastructure provisioning

Establish a centralized platform to help self-service infrastructure provisioning for faster domain and data product creation. This centralized platform will govern data mesh as per defined governance policies. A self-service infrastructure platform provides a user-friendly, quick, and compliant environment, taking away all the complexity of the underlying technical platform.

05

Establish org structure and identify domains and data products

This is a very important and intensive work that needs a thorough understanding of business and its modularity. Defining domains with clear ownership and data products requirements can eventually produce Data Mesh with high-performing data products and seamless governance. Data products should imbibe all eight characteristics as defined in an earlier section.

06

Define a unified governance setup process

Ideally, a governance committee will be established and represented by participants from different business units like security, compliance, legal, IT, and data domains. This team aims to identify, define, and impose governance policies that adhere to organizational and regional /government guidelines/policies. Having centralized federated computational governance can establish consistent and uniform policies across different data domains.

Why Data Mesh on Databricks?

Databricks provides many features, including data ingestion, data transformation, SQL, AI/ML and many more, making it a complete unified data platform. It takes away complexity involved with multiple tools/services and interoperability between them. This unified platform nature of Databricks makes it an ideal platform to implement Data Mesh architecture that demands heterogenous data types, use cases and data delivery methods. Below diagram depicts the match between Datamesh requirements and how Databricks can support the same.

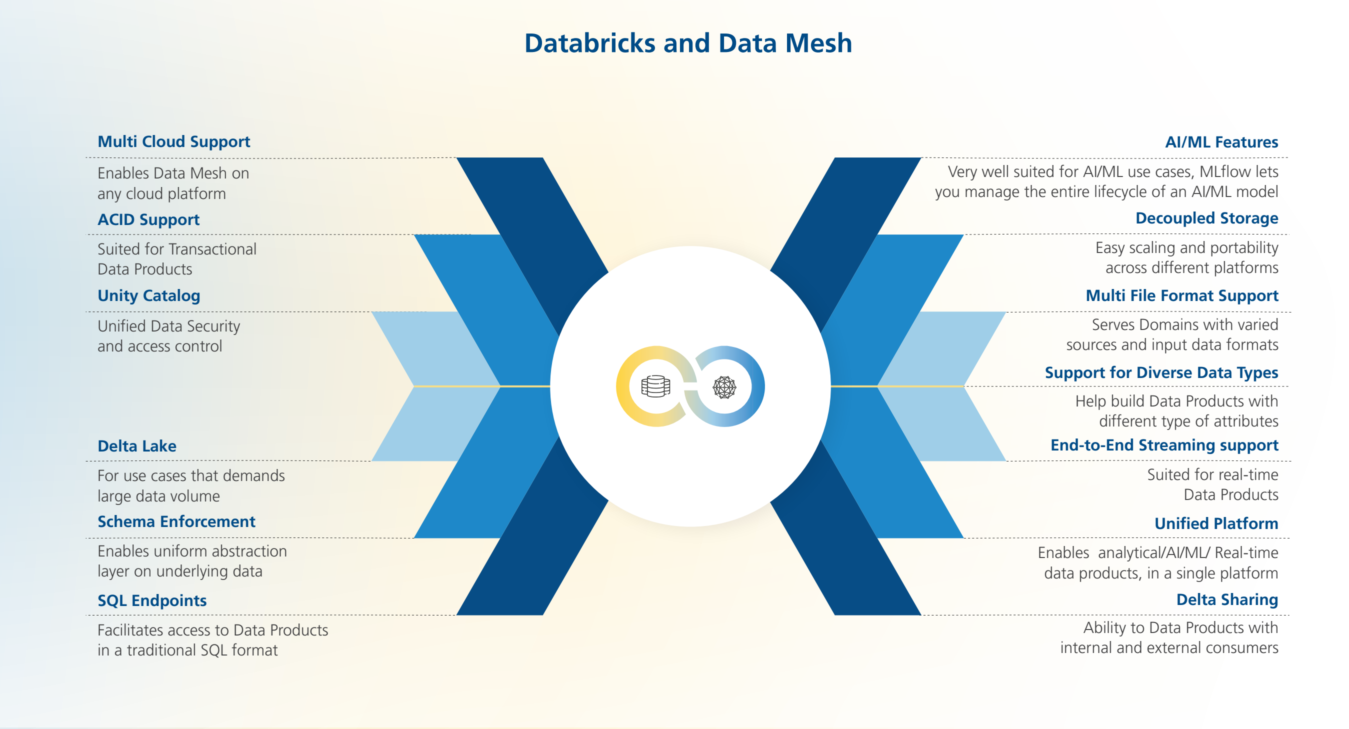


Figure 3- Data Mesh and Databricks Compatibility

Operationalizing Data Mesh on Databricks

Below diagram shows the logical architecture that comprises different modules of Data Mesh and their relevance within Data Mesh.

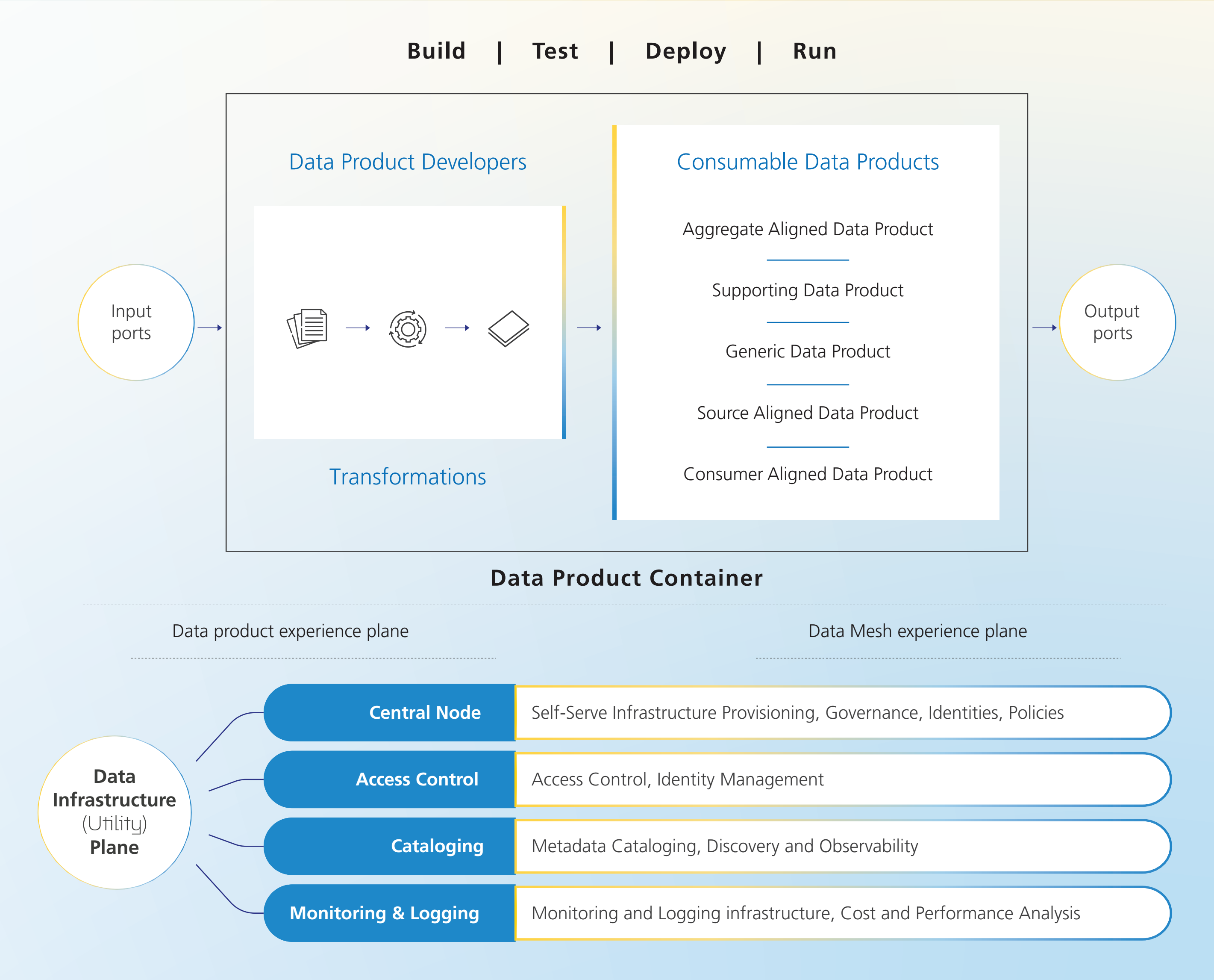


Figure 4 - Logical Data Mesh Architecture

Architecture Components	Input Ports	Output Ports	Data Product Container
Explanation	Represents ports to pull data from different sources. These sources could be native to Databricks or external, e.g., on-premises.	Represents ports to expose/share data products. These can be native, e.g., SQL Endpoints or API based.	It encompasses all the structural units required to create data products, such as services to ingest data, transform data and host final consumable data products.

Architecture Components	Data Product Experience Plane	Data Mesh Experience Plane	Data Infrastructure Plane
Explanation	This is a higher-level abstraction built using the infrastructure plane to build, maintain, and consume data products.	A marketplace where consumers can browse and subscribe to data products. This plane will enlist all relevant metadata for a given Data Product.	Self-service platform to define, manage and govern Data Mesh.

In subsequent sections, we'll provide an implementation perspective and elaborate on the approach to implement Data Mesh on Databricks.

Domain-oriented and decentralized data ownership

Data ownership / independent domain nodes can be built-in Databricks at the workspace level. Creating domain at a workspace level will provide required domain atomicity while reducing the data domain management complexity, anything lower than workspace e.g. repos etc. will result into more complex data domain governance and maintenance.

Consider below guidelines while creating each domain.

- Attach Cloud storage to each domain
- Create DevOps Repos and attach to each domain
- It is necessary to leverage Active Directory integrated with the cloud for user access management
- Workspaces can be further created in a logical cloud container such as Azure Resource group
- Databricks Delta tables will be used as a data product that can be published and consumed, using multi-model access, as explained in subsequent sections
- Leverage unity catalog for cataloging of all domains

Data as a Product

As described earlier, data product quantum refers to all structural units that present a data product with the final aim to produce a consumable data product that can be subscribed to and accessed by data product consumers. At the simplest form, Databricks tables/views can be used as a data product and access could be controlled using Unity catalog for internal consumers. Delta sharing is an excellent feature that can be leveraged to share the Data Products externally. As the Data Mesh and overall Databricks capabilities mature, objects such as AI/ML models and

ReDash reports/dashboards could also be made available as a data product. However, in all scenarios it is necessary to address each characteristic of the data product as discussed earlier. Below table summarizes these characteristics and potential Databricks features that can address those requirements. Services external to Databricks can also be considered as appropriate.

Data Product Characteristic	Databricks	Other Options
Discoverable	Unity Catalog	External Data Catalog Data Marketplace
Addressable	Delta Share ODBC/JDBC	BI Tools Data Science
Understandable	Custom Metadata Tagging	External Data Catalog
Truthful	Robust Testing, Data Lineage Quality Endorsement	Data Marketplace – Endorsement Tracking, Data Lineage Tools
Natively Accessible	Custom APIs, SQL Endpoints Databricks SQL	External tools using SQL Endpoints, ODBC/JDBC
Interoperable	Well-Defined Data Standards SQL and cross domain Joins	Abstraction/Semantic Layer
Valuable	Custom metadata Usage Metrics	Data Marketplace – Ratings and Views
Secure	Encrypted Access Tokens RBAC/ABAC Dynamic Data Masking	External tools such as Immuta External Tokenization Secret and Keys
Cost	DBU Usage and Cost Monitoring Budget Alerts Policies to create compliant clusters	Chargebacks to respective Domains Finops for cost optimization
Compliance	Unity Catalog – Masking PII PCI data	Compliance Governing committee Audit and approval of Data products

Figure 5 - Data Product Characteristics and Implementation Approaches

Once Data products are created, the next important aspect is to make them easily available to the prospective consumers. While Data Marketplace can provide the way of listing the data products, each domain needs to identify the method to share the respective data products once they are subscribed by a consumer. For a better reach and consumption, multi-model access approach is recommended as described below.

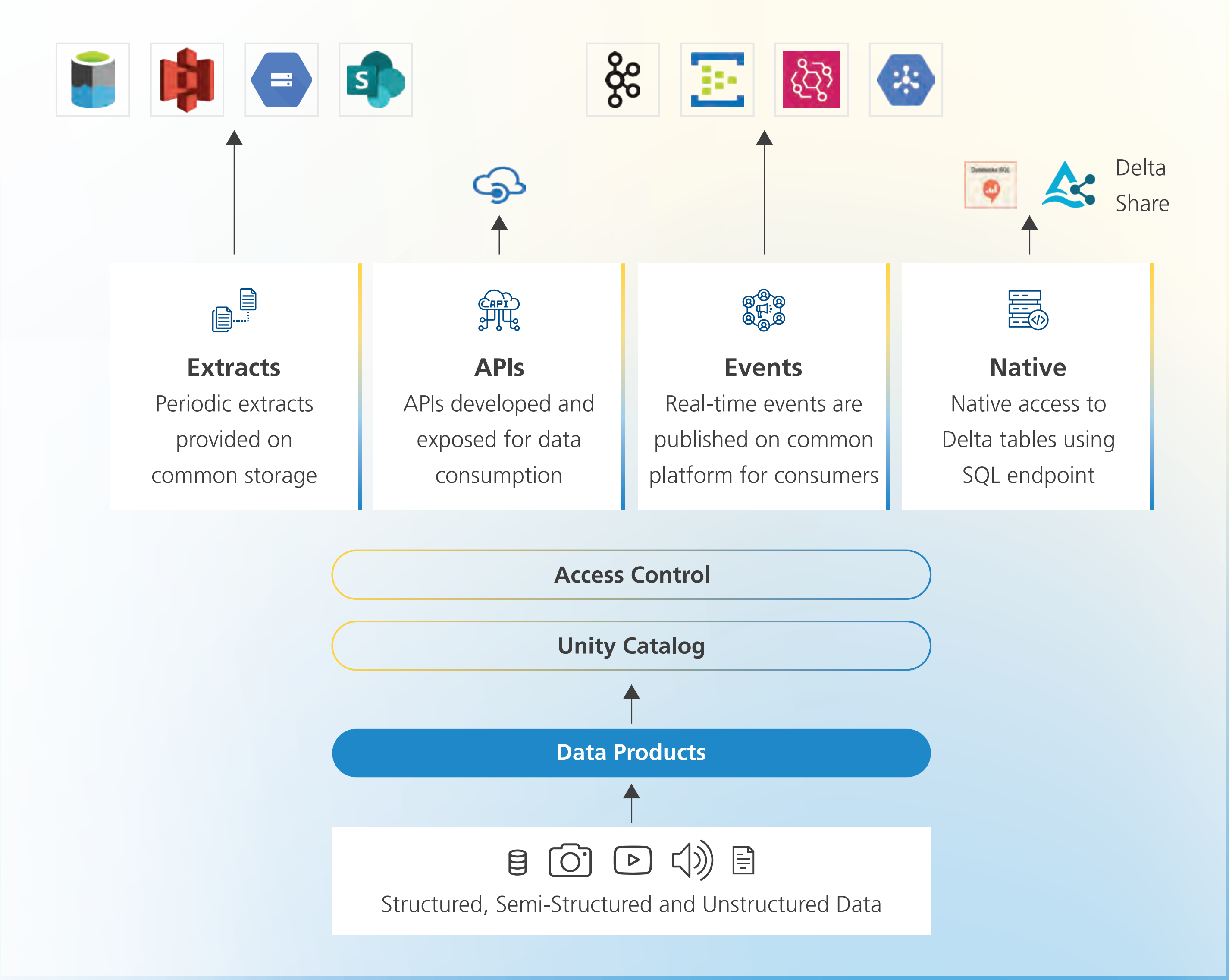


Figure 6 - Data Product Access Model

There are four typical access methods that can be implemented in Databricks.

Extract

Suitable for data products that can refreshed periodically. New/updated Data products can be extracted periodically on a common storage location such as ADLS/S3, from where consumers can access and ingest or their purpose.

Events

In case of real or near-real time events, data products can publish these events on an external service such as Event Hub or Kafka. With proper access management, consumers can be provided access to these events, and they can integrate their application to ingest the data.

APIs

Additionally, API layer can be built to expose Data product elements. e.g., Customer details. Consumers can use these APIS to access relevant data of the data product.

Native

SQL Endpoint can also be used to provide access to the data products, tables in this case, natively. This type of access method is suitable for only internal customers for security reasons.

User Personas

Domain Provisioner

- Creates domain and grants access to domain owner
- Owns Self-service infrastructure provisioning platform
- Integrates and enforces computational governance on Domains
- Part of Central IT team

Domain Developer

- Creates data products based on use cases
- Manages end-to-end development cycles
- Part of Domain team

Domain Owner

- Manages domain
- Grants access to developers
- Publishes Data Products and approves access to consumers
- Part of Domain team

Domain Consumers

- Leverages Data Marketplace to search for Data products
- Subscribe and consumes data products
- Can be part of any team in the organization

Figure 7

Data Mesh User Personas

Domain provisioner persona belongs to Central IT team, while domain owner and developer personas belong to a particular data domain. Anyone can be a domain consumer, internal and/or even external to the organization.

Federated computational governance

For a consistent behavior of the Data Mesh and to be compliant with the organizational policies, it is essential to implement governance in a federated computational manner, i.e., it should be governed in an automated fashion and enforced automatically as per the configuration of policies and scope. Also, governance manages automated metadata capture to facilitate searchability and observability of data products.

A few essential governance aspects of being considered are:

- Data Catalog
- Data Lineage
- Data Classification
- Policies
- Monitoring
- Logging and Alerting
- Identity and Security

Data Catalog

Here are the features of a Data Catalog:

- Automatic technical metadata collection.
- Business metadata via predefined tag templates.
- Default all datasets across domains published.
- Domains can restrict based on the sensitivity.
- A local catalog with entire data set attributes can be leveraged by the technical team for analysis and development.
- Enterprise catalog with published data products that are intended for end-user consumption. This catalog will be integrated with the marketplace to provide consumers with user-friendly navigation of published data products.

Unity Catalog can be used to implement data cataloguing features

Data Lineage

Here are the features of Data Lineage.

- Provide end-to-end lineage for given data products.
- It should be easily accessible in a user-friendly way.

Unity catalog can be integrated with Enterprise cataloging solution such as Azure Purview and Collibra to implement the Data Lineage feature. Or external tools/DataOps platform such as dbt can be leveraged.

Data Classification

Data Product attributes should be classified based on organizational data security policies.

Unity Catalog can be integrated with Enterprise cataloging solution such as Azure Purview and Collibra to implement Data Classification features.

Monitoring

Databricks monitor with built-in/pre-configured charts/statistics can be used for monitoring purposes.

Logging and alerting

Native Databricks Logging and alerting can be used. Alternatively, specialized application like FinOps can be built to implement centralized logging, monitoring, alerting and optimization for all the Data Domains i.e., Databricks Workspaces.

Policies

For consistent enforcement of data compliance, it is necessary to enforce policies, global and local, on the data domain and data products. There are two steps, policy definition, and policy enforcement. Databricks policies can be leveraged. Alternatively, open-source like Open Policy Agent can also be used to create a framework for policy definition and enforcement. The governance committee is primarily responsible for defining and enforcing policies.

There are three types of policies that can be considered.

Type1: Policy related to code

Example: Table names should be in the upper code.
Enforcement: CI/CD pipeline.
Implementation: DataOps platform.

Type2: Policy related to Infra

E.g., Warehouse size cannot be more than XL.
Enforcement: Self-service provisioner.
Implementation: IaaC script/Databricks Policies

Type3: Policy related to Data

E.g., Emp_Num must be positive.
Enforcement: Data pipeline.
Implementation: DataOps platform.

Identity and security

Azure AD should be used as a single service to manage identities and access to the resources. Please note that specific elevated AD access like Directory Reader will have to be enabled for managing resource accesses through a self-service portal.

Implementing federated governance is the most critical and complex process. A well-established governance guidelines and designated accountability needs to be identified and implemented for a comprehensive coverage of relevant organizational and geographical compliance requirement. Below diagram describes potential structure of Data Governance committee and their respective accountabilities.

Governance and Critical features

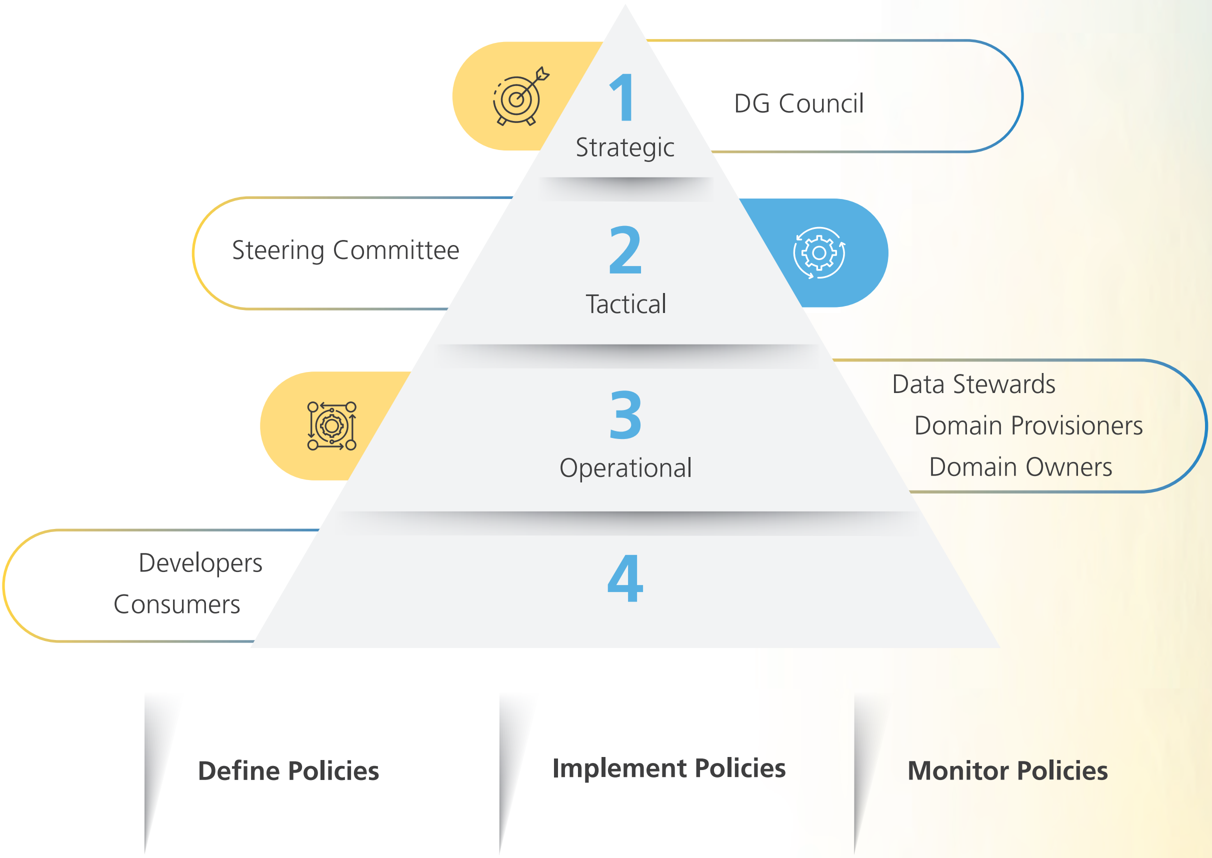


Figure 8 – Data Governance Council

Self-service data infrastructure

Portal Architecture

A self-service platform can be created using preferred technologies like HTML, CSS, Python, etc. However, it is advisable to follow industry-recognized 3-layer architecture to isolate the front, back, and database tiers into different networks. Microservice-based architecture is recommended for better control and reusability. All microservices about a service functionality, e.g., creating a workspace or repo, can be written using Databricks APIs, and leveraged from Web Tier.

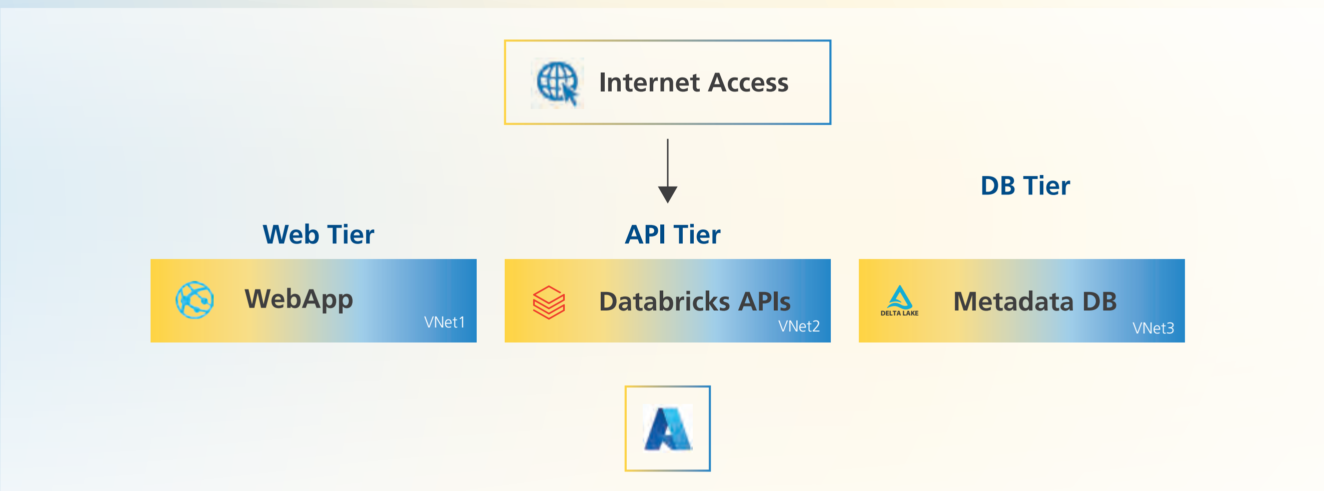


Figure 9 – Self-Service Portal – 3 Layer Architecture

As shown, only Web Tier can be accessed from the public internet. API and DB Tier are isolated from the internet.

Web Tier: Front end for the self-service portal, accessed from the internet.

API Tier: Backbone of the self-service portal. Hosts microservices for different Databricks APIs.

Metadata DB: Lean metadata layer to store functional details of a data domain and data products. Also, it stores particulars about the accessibility of a data product and consumers.

Web Tier

Front end for the self-service portal, accessed from the internet.

API Tier

Backbone of the self-service portal. Hosts microservices for different Databricks APIs.

Metadata DB

Lean metadata layer to store functional details of a data domain and data products. Also, it stores particulars about the accessibility of a data product and consumers.

ARM template-based framework

Creating a generalized framework to execute different Databricks related requests is necessary. E.g., creating a workspace, cataloging a DB, or granting access to a data product. Azure ARM templates are one probable solution to make such a framework. All templates can be stored in the DevOps repository and executed using Azure pipelines and parameters as depicted in the below diagram. Alternatively, direct Databricks APIs can be used to submit different requests. However, such architecture will become more rigid and complex to manage due to embedded functionality within codes.

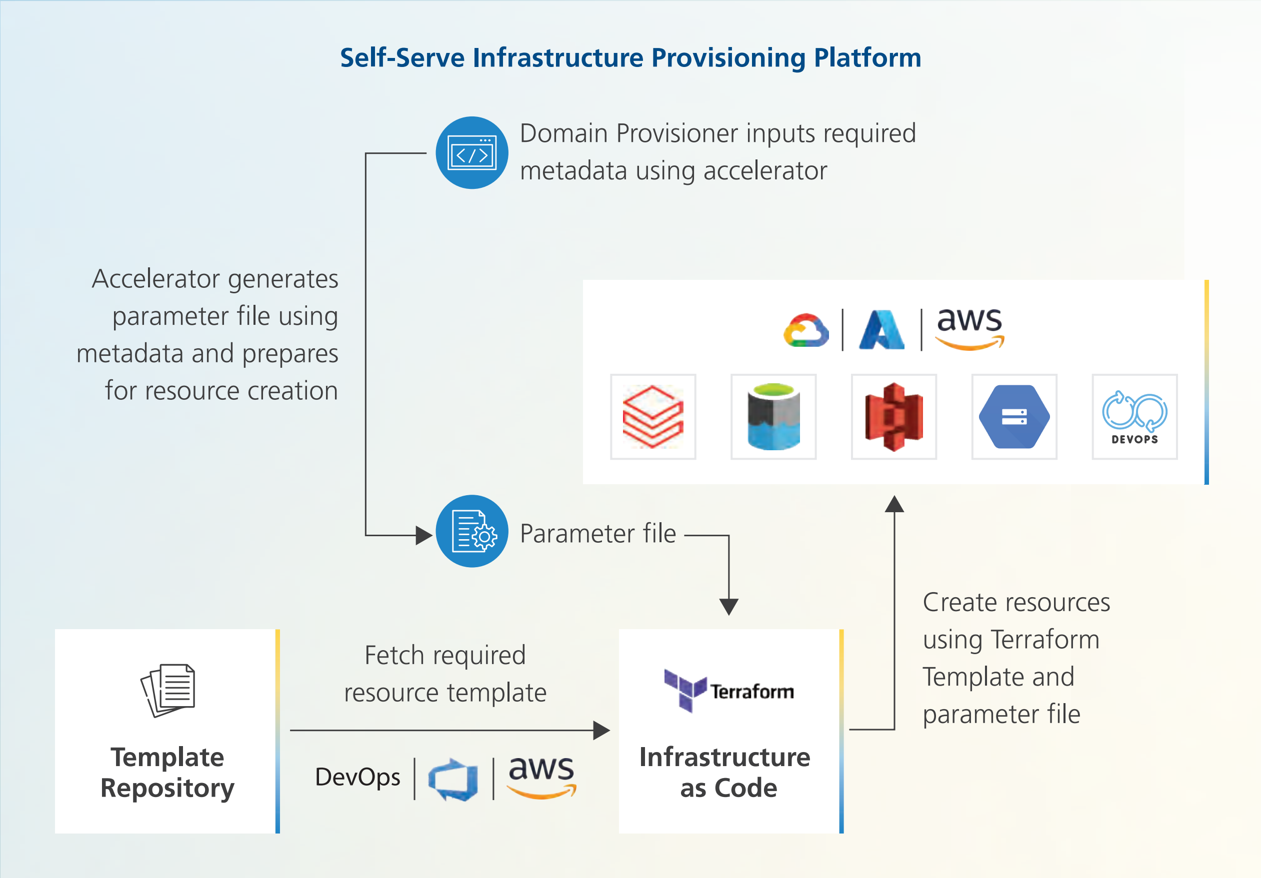


Figure 10 - ARM Template based Framework

Advantages of template-based Framework:

- Faster development due to decoupling of accelerator and resource creation processes.
- DevOps CI/CD processes can be leveraged for development and enhancement.
- Centralized storage and management of ARM Templates.
- Lean and modularize architecture for Accelerator.
- Easy-to-deploy accelerator as PaaS service.

DataOps Platform

DataOps - For Development
Essential Components

Develop
Develop modular codes consistency and faster

Test
Test before migrating to production and share test results

Deploy
Maintain versions and facilitate CI/CD

dbt — DBT is one of the options for the DataOps platform and provides direct integration with Databricks

DataOps Capabilities

Speed | Versioning | Quality | Automation | Deployment | Collaboration | Governance

Figure 11 - DataOps Platform

Due to the decentralized nature of Data Mesh architecture and multiple teams leveraging the same, it is essential to establish a collaborative data management practice to improve the communication, integration and automation of data operations and data product development. DataOps platform need to support at least below mentioned features.

Develop: Users should be able to quickly and easily develop data pipelines to create new data products and manage existing ones.

Test: Tool should have the ability to execute test cases against a given data pipeline to test and certify the accuracy of the data product. It should provide the automated way to execute these test cases and provide detail view of the outcome for better analysis and rectification of any issues.

Deploy: Tool should support CI/CD feature required for collaborative development environment.

Data product marketplace

To successfully adopt Data Mesh, it is essential to create an easy-to-use and intuitive data product marketplace for consumers to browse and subscribe to required data products. The marketplace platform leverages catalogs and metadata generated by the self-service infrastructure platform and creates a functional view of underlying data products for easy consumption. It also facilitates the approval request workflow to raise and grant requests for a particular data product.

Please note that only the enterprise catalog qualifies and integrates with marketplace. Integrating native catalogs can lead to too many unwanted data products and potentially create confusion and improper data product selection.

Also, the marketplace should track and enlist statistics around 8 attributes of a data product, as mentioned in an earlier section. E.g., it should capture the number of subscriptions, consumers, and access to a data product to understand the value and discoverability of that data product. Data products that are not used or have poor user ratings could be considered for enhancement or decommissioning.

Below are the critical features needed for an effective and productive data product marketplace.

Search

Provide search based on keyword, domain etc.

Explore

Bookmarks, download, favorites

Access

Ability to request and grant access

Collaborate

Ability to share and publish data products

Personalize

Recently viewed, recommendations for you like features

Personalize

Recently viewed, recommendations for you like features

Preview

Ability to preview data products and features

Insights

Track and present data product performance

Conclusion

Data Mesh is a new way of creating and sharing 'Data-as-a-Product' by decentralizing data ownership and accountability. However, extensive research, thinking and decision making are required to define proper data domain strategy as a pre-requisite to Data Mesh implementation. Next critical business task is to define trustworthy and accurate data products that can be easily discovered and shared across organization.

Important yet complex task is to create self-service data infrastructure that executes technical requirements in the form of Data Mesh functionality. Databricks' unified platform enables simple and effective implementation of Data Mesh in cloud. LTI's Databricks Data Mesh accelerator simplifies the centralized platform infrastructure and integrates federated computational governance in easy-to-use UI portal, therein reducing overall cost and efforts to implement Data Mesh.

References

Data Mesh – Delivering Data-Driven Value at Scale by Zhamak Dehghani

Article by Paul Andrew: https://mrpaulandrew.com/

https://www.openpolicyagent.org/

LTIMindtree

About the Author

Abhishek Patel

Head of Databricks Center of Excellence - LTIMindtree

He has over 21 years of experience helping customers build data platforms and analytical systems. He has architected multi-year large-scale projects on data modernization and migration to cloud data platforms like Azure and Databricks. His expertise lies in data modernization & cloud transformation for customers across banking and financial services, manufacturing & engineering, and healthcare industries, with core competencies in data strategy development, architecture design, application development, data warehousing, data integration, and cloud modernization. He holds a bachelor's degree in Computer Science.

About LTIMindtree