# Modern Data Architecture in the Cloud Era

How cloud-native design, decentralized ownership, and metadata-driven automation are shaping tomorrow's data platforms

# About Me

## Sjoukje Zaal

### CTO Data & AI Europe

Insights & Data @Capgemini

# Agenda

01

## Cloud-Native Architecture

Leveraging elastic computing and modular design principles

02

## Data Mesh

Decentralizing ownership with domain-oriented thinking

03

## Data Fabric

Connecting systems through metadata and automation

04

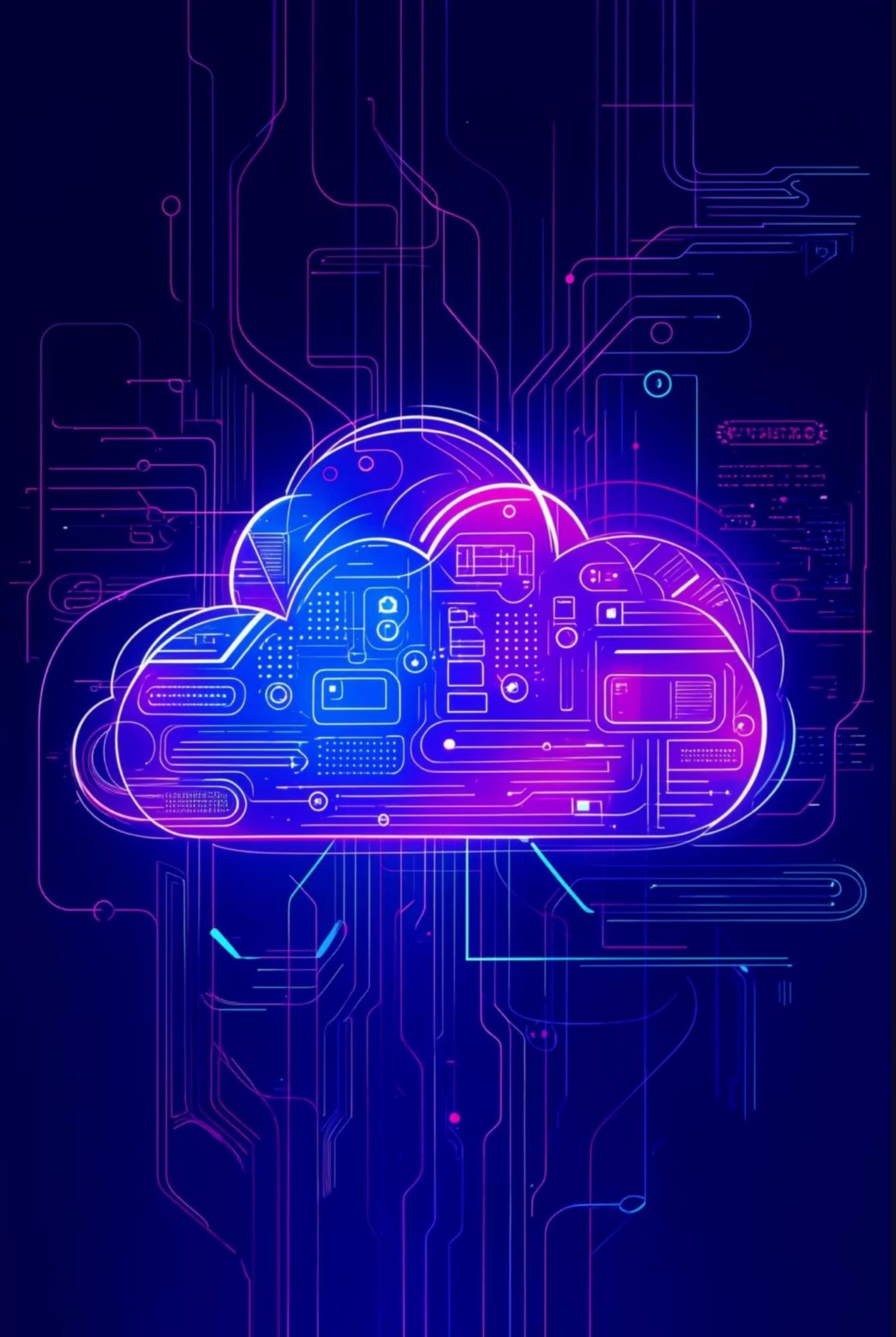
## Data Lakehouse

Unifying analytics and data science workloads

05

## Future Trends

Emerging patterns shaping tomorrow's data landscape



# Cloud-Native Architecture

# What is Cloud-Native? (Data Context)

Cloud-native data architecture means designing data platforms to take **full advantage of cloud elasticity, automation, and modularity** — from day one.

This doesn't mean just "lifting" your data warehouse into the cloud. It means building:

- Scalable **compute clusters** that spin up/down as needed
- **Serverless data services** that react to events
- Modular components connected via APIs or event streams

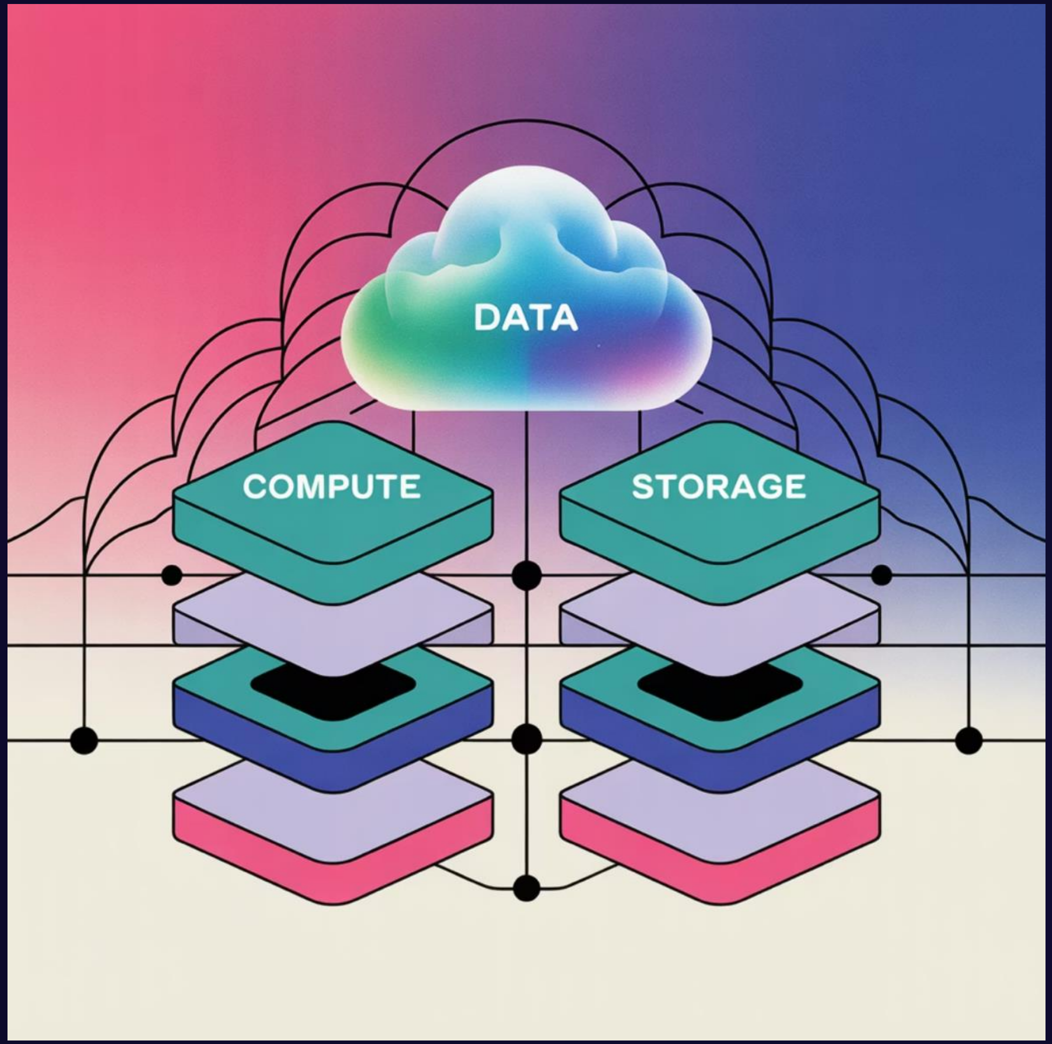
# Benefits of Decoupling Compute & Storage

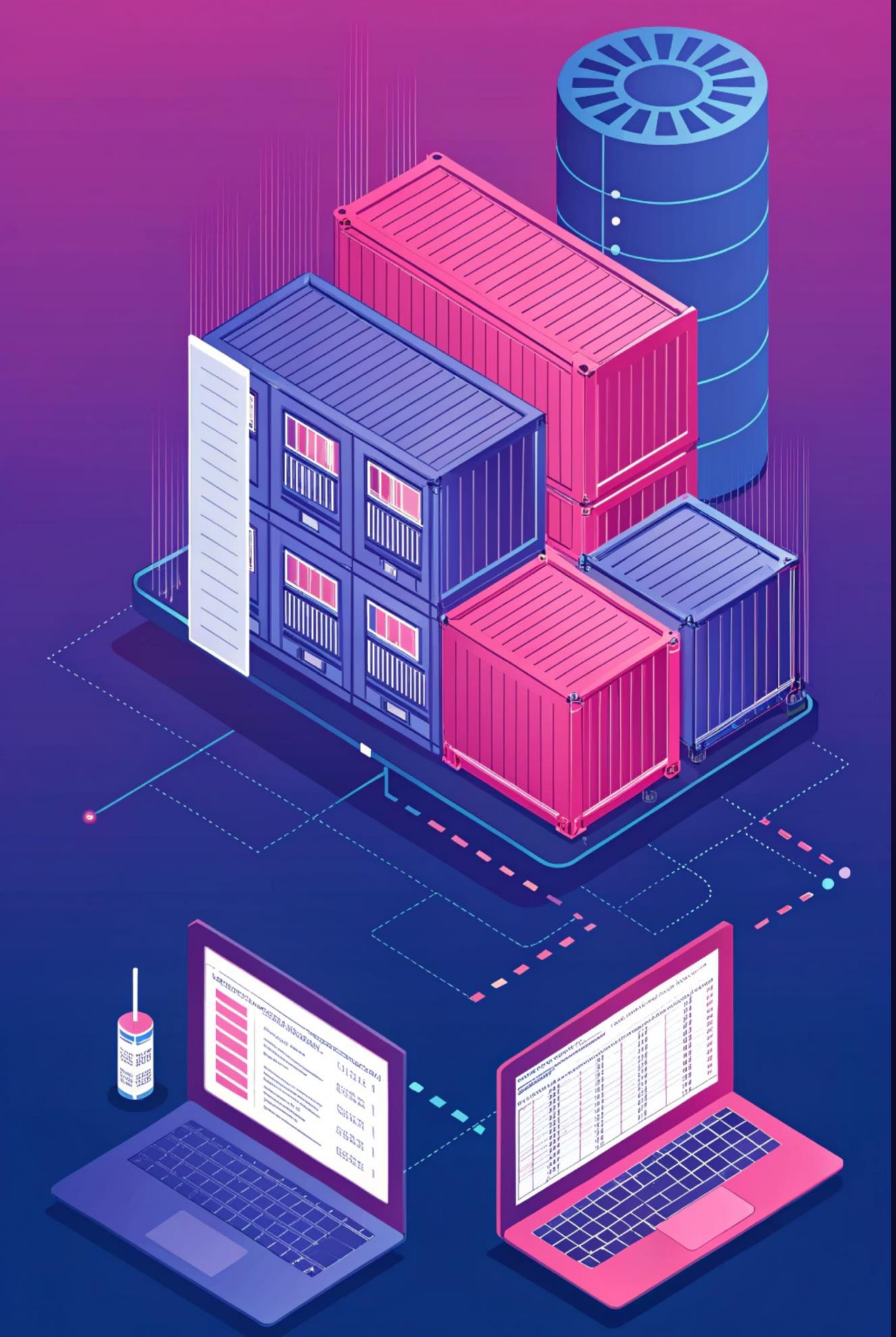
Cloud-native architectures separate:

- **Storage** (low-cost, high-availability object storage like S3, ADLS, GCS)
- **Compute** (on-demand engines like Spark, Presto, BigQuery, Synapse)

This enables you to:

- Run analytics across huge datasets **without copying data**
- Scale compute workloads independently — based on load, team, or SLA
- Save cost by **shutting down idle compute**





# Real-World Example – Containerized Data Platform

## The Challenge

Large organization migrating from on-prem Hadoop to a cloud-native data platform

## The Solution

- Containerized **Spark jobs** for batch ETL
- Kafka and Event Hubs for real-time ingestion
- ML pipelines running in Kubernetes
- Central data lake on object storage

## The Results

- Cut data processing costs by 40%
- Reduced pipeline deployment from days to minutes
- Enabled **isolated, parallel workloads** on shared data

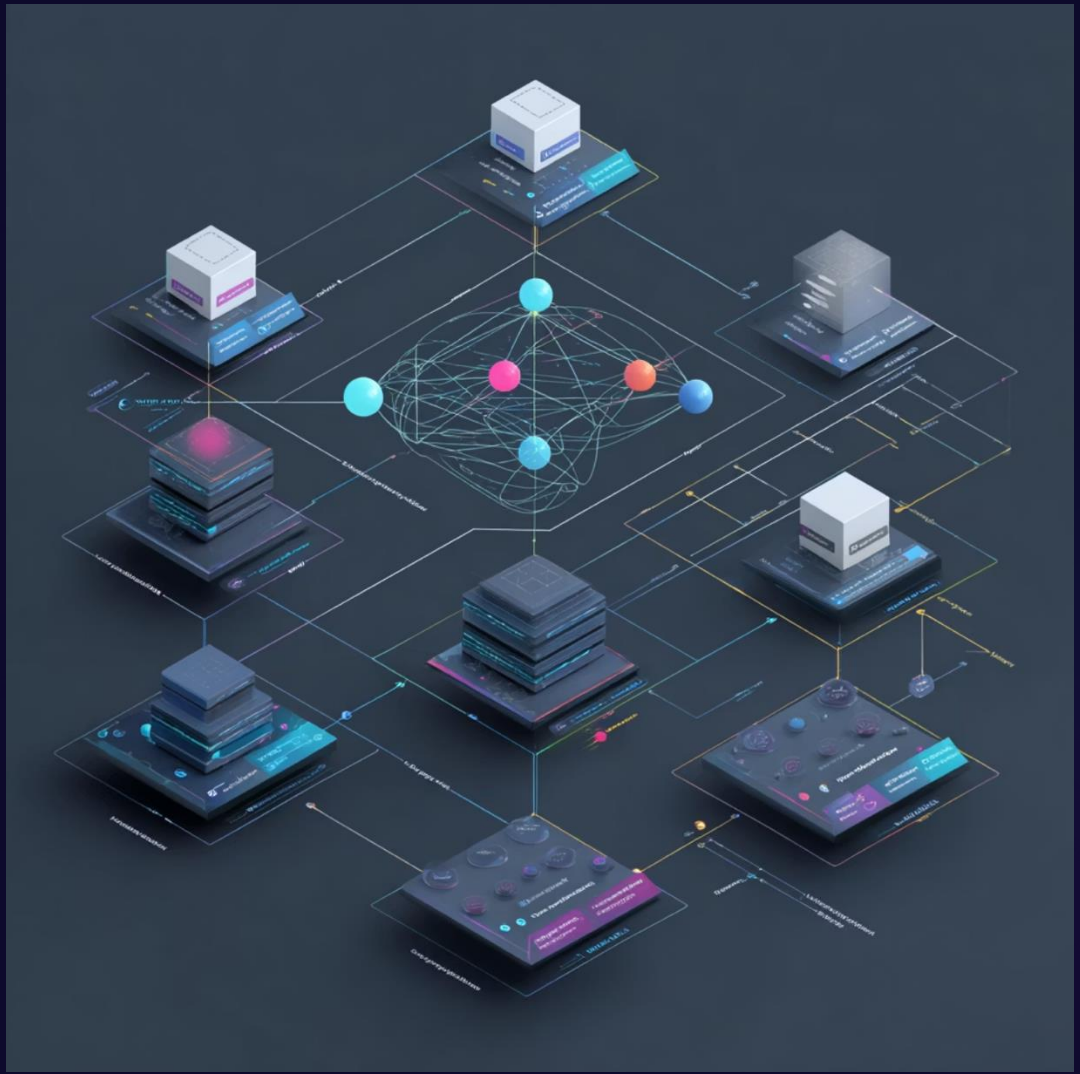
# Microservices + Service Mesh for Data Services

In modern data platforms, we break up monolithic ETL or orchestration engines into **small, composable services**:

- Ingestion services
- Transformation engines
- Feature serving
- Lineage tracking
- Cataloging and discovery

A **service mesh** adds structure to this growing web of services:

- Secure, encrypted communication between sensitive data services
- Fine-grained routing for A/B testing or blue/green pipeline rollouts
- Centralized telemetry and policy enforcement



# Cloud Independence – Do's and Don'ts for Data Platforms

In data architecture, cloud independence doesn't mean avoiding cloud services — it means designing for **flexibility and portability** when needed.

## DO

- Store data in **open formats** (Parquet, Delta, Iceberg)
- Use **containerized runtimes** for ML and pipelines
- Keep **metadata and governance layers separate** from vendor tools

## DON'T

- Build your data model around proprietary features
- Lock critical business logic into vendor-specific tools
- Sacrifice strategic flexibility for short-term convenience

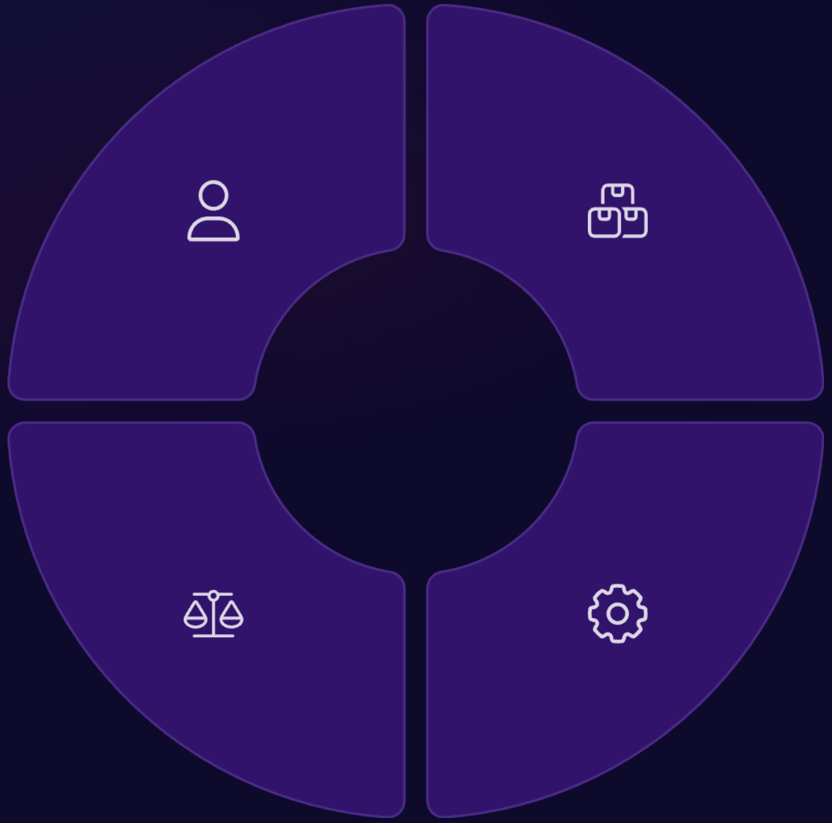
The goal is not "vendor avoidance" — it's making **smart bets** with a clear exit plan.


# Data Mesh

# What is a Data Mesh?

Data Mesh is a response to a problem we've all seen: Centralized platforms become bottlenecks.

Instead of routing all data through a central team or platform, we shift to a **domain-oriented approach** — where teams **own and operate their own data products**.



### Domain Ownership

Business-aligned teams own their data end-to-end

### Data as a Product

Each dataset is treated like a product with documentation, quality, and lifecycle

### Federated Governance

Standards are enforced across domains, not above them

### Self-serve Platform

Reusable tooling and infrastructure, offered as a service

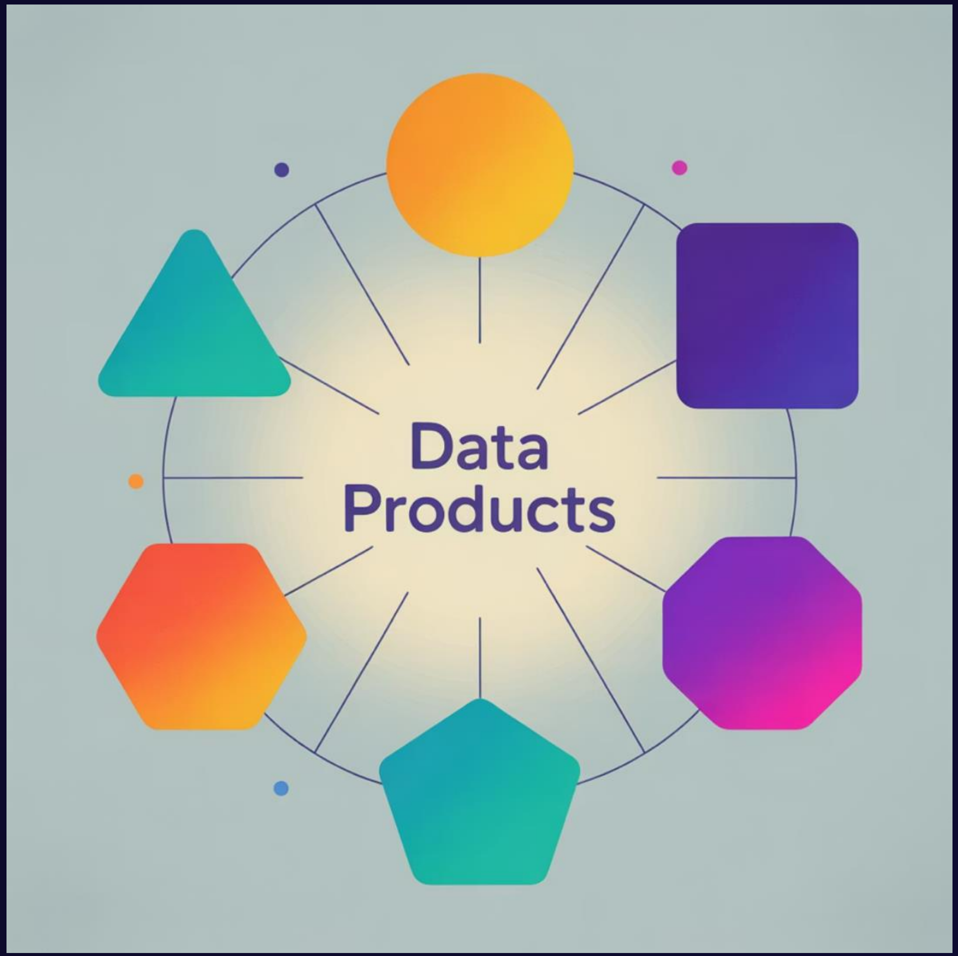
It's not a tool. It's an operating model.

# Centralized vs Decentralized Model

## Centralized Model

- One data team handles ingestion, pipelines, governance, and reporting
- Bottlenecks grow as data demand scales
- Business teams depend on others to get insights

## Decentralized (Mesh) Model




- Each domain owns the full lifecycle of their data products
- Central teams build shared infrastructure
- Data is closer to those who understand it best

# Org Structure & Governance Implications

A Data Mesh will fail without **organizational alignment**.

## Team Structure

- **Autonomous data teams** aligned to business domains
- Accountable for quality, discoverability, and SLAs
- Blend of business and technical skills

## Governance Model

- Central governance becomes **federated**
- Standards enforced through automated checks
- Enabling teams rather than blocking them

## Required Elements

- Clear **roles and ownership** across domains
- Defined **interfaces** for metadata, policies, contracts
- Incentives for producing reusable data assets

Without this structure, it becomes chaos. With it, you get scale.

# Product Thinking Applied to Data

What does "data as a product" mean in practice?

Think about how we build digital products:

- There's a **target audience**
- Defined **interfaces and documentation**
- A clear **lifecycle** and support model

We apply the same thinking to data:

- Who is this data for?
- Is it reliable, versioned, and well-described?
- Can others discover and use it without the original team?

# The Cultural Shift

This is where most transformations fail — **the culture**.

## From Service Provider

- Reactive to business requests
- Focused on completing tickets
- Success = pipeline works

## To Product Owner

- Owning outcomes, not just pipelines
- Thinking in SLAs and user experience
- Success = others get value from your data

A fundamental mindset shift: From "How do we centralize and control?" To "How do we scale and enable?"

# Data Fabric

# Data Fabric vs Data Mesh

## Data Mesh

An **organizational model** that:

- Distributes ownership to domain teams
- Encourages product thinking for datasets
- Pushes responsibility to business domains

**The "who" and "how"**

## Data Fabric

A **technology approach** that:

- Connects data across environments
- Discovers and manages metadata
- Automates governance and lineage

**The "what" and "where"**

They're not competing concepts — they solve different parts of the problem. You can (and should) use both together.

# Metadata & Active Metadata

## Passive Metadata

Static information about your data:

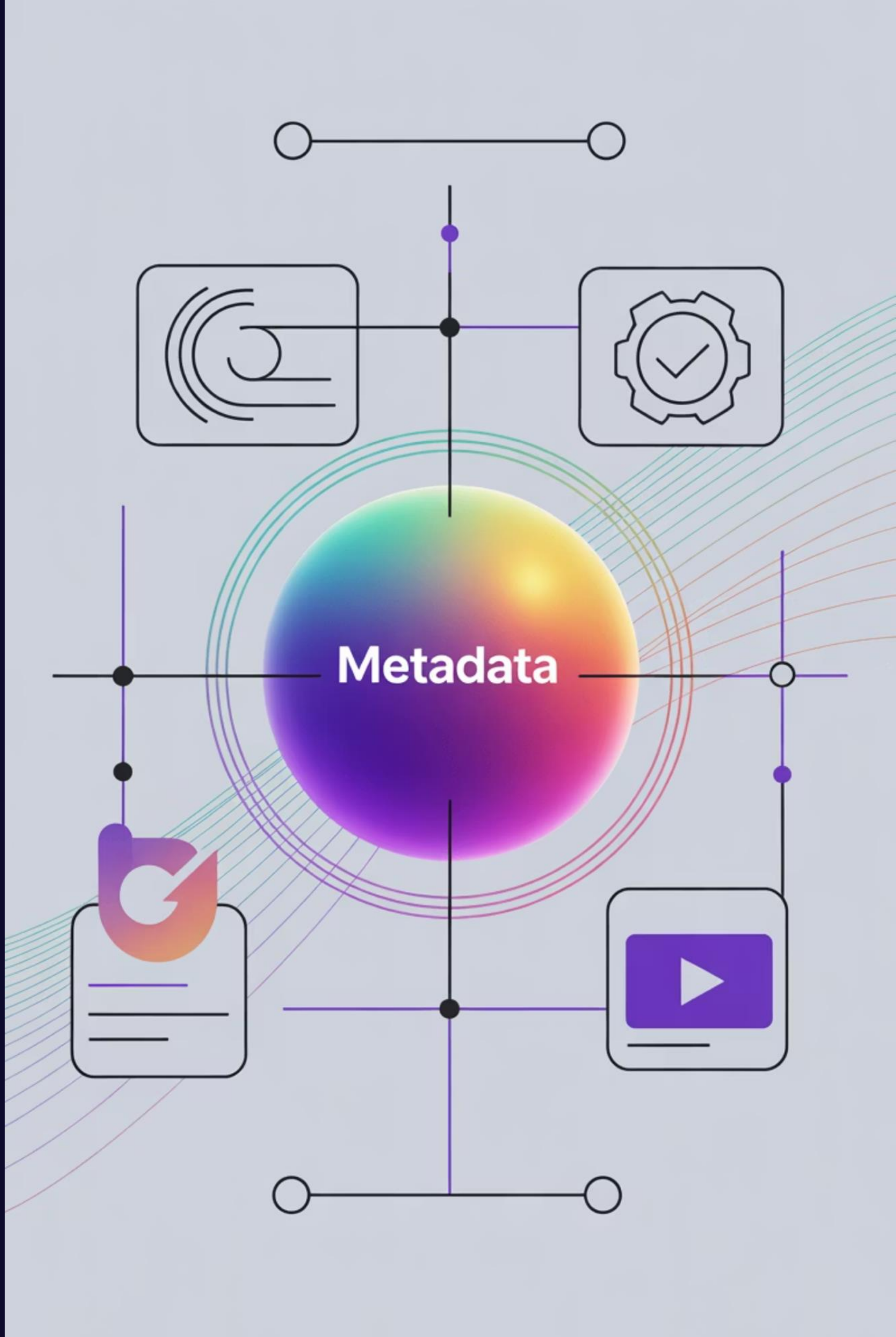
- Schema definitions
- Data types
- Source systems
- Business glossary terms

## Active Metadata

Dynamic, real-time signals:

- Usage patterns and popularity
- Quality scores and freshness
- Processing history
- Access patterns and data flows

Active metadata turns your platform from a directory into a **smart, responsive system**.



# Governance & Lineage Automation

Manual governance doesn't scale. In a modern fabric architecture, governance is **automated and embedded**.

### Real-time Enforcement

Policies applied at ingestion — not weeks later

### Automatic Protection

Sensitive data auto-tagged with access restrictions

### End-to-end Lineage

Traceability from dashboard to raw source

# Reusable Data Products via Composition

Data Fabric helps accelerate innovation by making data **modular and reusable**.

Instead of building each dataset from scratch:

- Compose new data products by linking existing ones
- Think like LEGO blocks — combining sales data with customer
- profiles and churn predictions

This works because:

- Metadata describes how pieces fit together
- The fabric enforces dependencies and access rules

# Data Lakehouse

# What is a Lakehouse?

The **Lakehouse** is an architectural pattern that combines the best of both worlds:

## Data Lake Benefits

- Flexibility and scale
- Support for all data types
- Cost-effective storage
- ML-ready format

## Data Warehouse Benefits

- ACID transactions
- Schema enforcement
- Time travel capabilities
- Fine-grained governance

This means you can run traditional BI, real-time streaming, and machine learning — all from one place, without copying data across systems.

# Warehouse vs Lake vs Lakehouse

| Feature | Data Warehouse | Data Lake | Data Lakehouse |
|---|---|---|---|
| Schema | Strict (predefined) | Flexible or none | Enforced on write/read |
| Cost | High | Low | Lower |
| Data Types | Structured only | All types | All data types |
| BI Support | Strong | Weak | Strong |
| ML Support | Limited | Strong | Strong |
| Governance | Built-in | Requires tooling | Native + open formats |

The Lakehouse gives you structure where you need it, and flexibility where you don't — without the overhead of running two platforms in parallel.

# Delta Lake, Apache Iceberg, Apache Hudi

Lakehouse architectures rely on **open table formats** that bring structure and performance to object storage.

### Delta Lake (Databricks)

- Strong ACID support
- Optimized for Spark
- Popular in enterprise setups

### Apache Iceberg (Netflix, Apple)

- Engine-agnostic
- Open community-driven format
- Rich metadata capabilities

### Apache Hudi (Uber)

- Optimized for real-time ingestion
- Supports upserts
- Used in high-velocity environments

Each has strengths — but they all bring **warehouse-like reliability** to your lake.

# Unified Storage = Fewer Copies, Fresher Data

## Traditional Problem

Data sprawl across multiple systems:

- One copy in the lake for ML
- Another in the warehouse for reporting
- A third in a database for operational use

Leading to:

- Sync delays
- Inconsistent results
- High storage and processing costs

## Lakehouse Solution

Multiple engines access the same data:

- BI tools query with SQL
- ML pipelines use Spark
- Dashboards stay fresh automatically


# Use Case – BI + ML on One Platform

1

## Retail Company Challenge

Analyze customer behavior across online and in-store purchases

2

## Multiple Use Cases

- **Dashboards** for marketing teams
- **Churn prediction models** for data scientists
- **Streaming updates** from transactions

3

## Lakehouse Implementation

- Data lands in Delta/Iceberg tables in object storage
- BI analysts use Power BI directly on that table
- Data scientists train models on the same data
- Real-time updates refresh both simultaneously

No duplication. No silos. One platform serving multiple teams.


# What's Next?

# AI in DataOps – Monitoring and Remediation

As data systems grow more complex, manual monitoring doesn't cut it anymore.

**AI-driven DataOps** is emerging for:

- **Detecting anomalies** in pipelines or data quality
- **Predicting failures** before they impact users
- **Auto-remediating issues** — restarting jobs, flagging broken schemas

Use cases:

- Flagging unusual drops in daily ingestion
- Auto-blocking downstream jobs if quality degrades
- Suggesting root causes for failed loads


It doesn't replace engineers — it frees them up to focus on improvement instead of firefighting.

# Real-Time Architecture Patterns

### Event-driven Ingestion

Using Kafka, Event Hubs, or Kinesis to capture data changes as they happen

### Stream Processing

Spark Structured Streaming, Flink, or Materialize for continuous transformation

### Micro-batch Updates

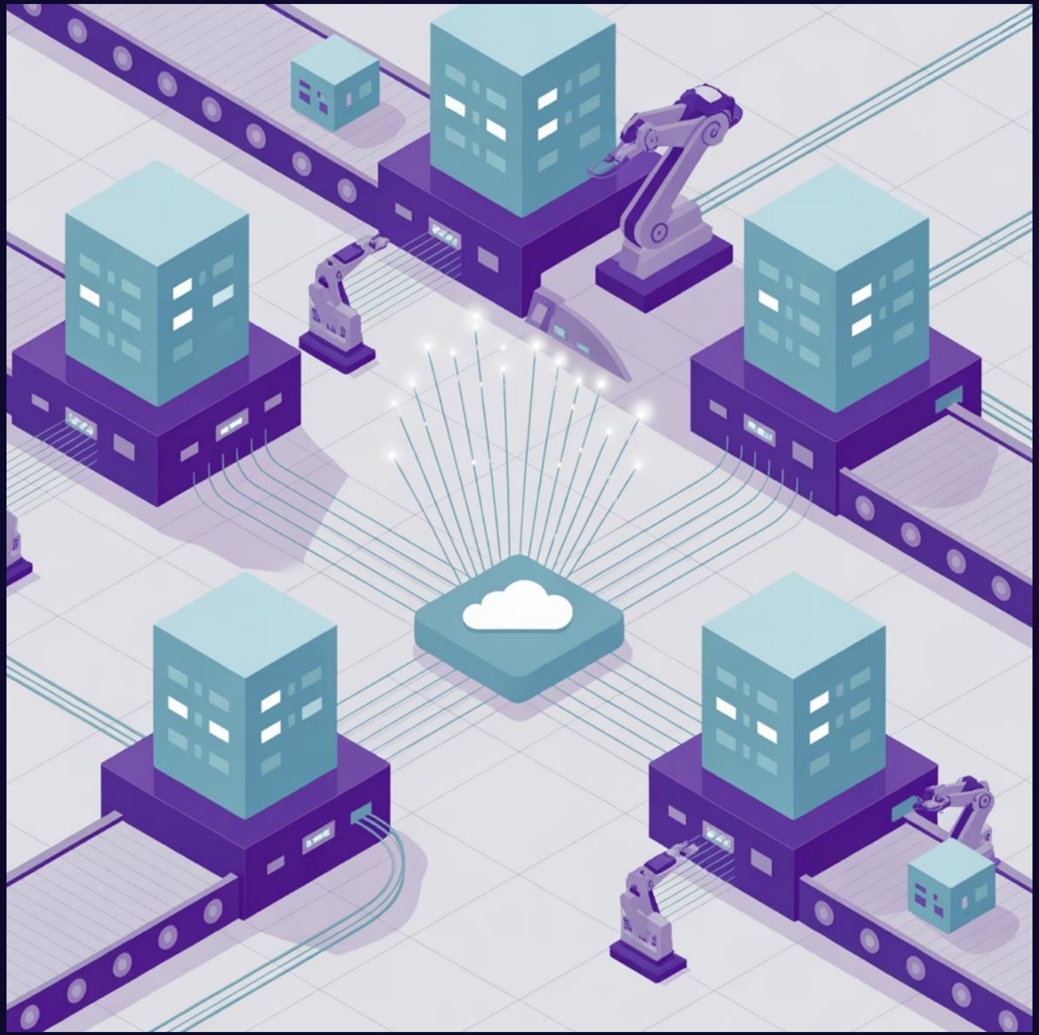
For dashboards or ML feature stores needing near-real-time refresh

### Low-latency APIs

Serving processed results directly to applications and users

Architectures are shifting from hourly jobs to **millisecond updates** — especially in areas like fraud detection, recommendations, and customer scoring.

# Edge Computing and Hybrid Cloud



As more data is generated outside traditional data centers, **edge and hybrid** architectures are becoming essential.

## Key Drivers

- **Latency** — when decisions need to happen instantly
- **Bandwidth** — when streaming everything to the cloud is prohibitive
- **Compliance** — when data needs to stay local for regulatory reasons

## What's Changing

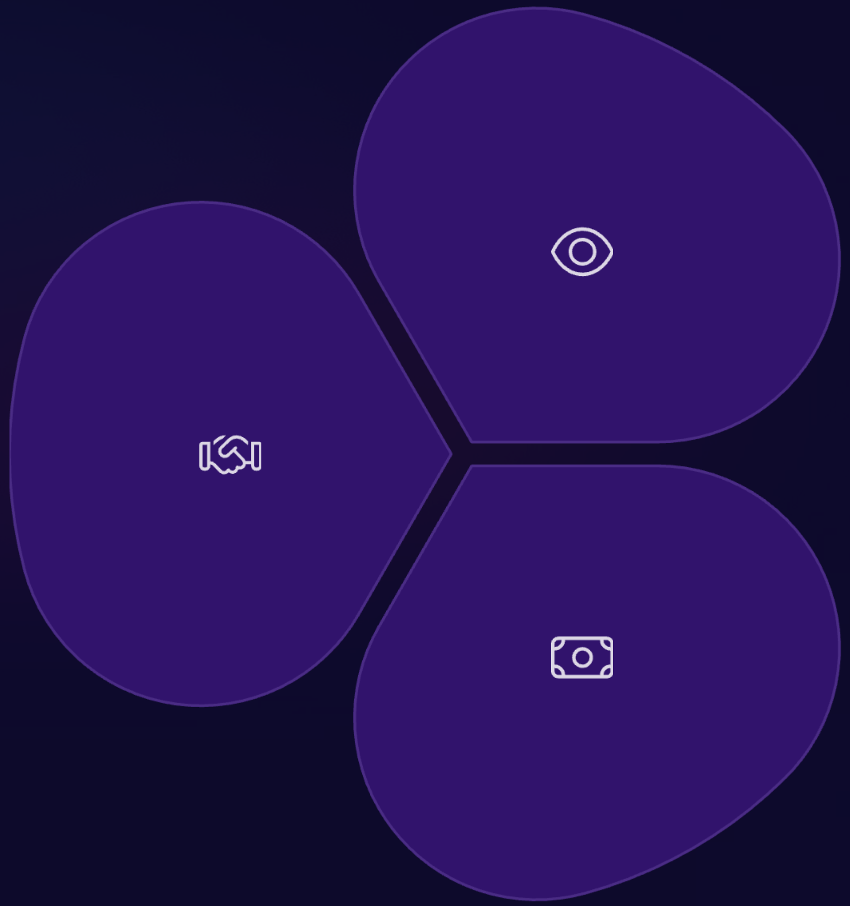
- AI models pushed to the edge
- Metadata synced centrally, processing happens locally
- Hybrid platforms manage policy and identity across locations

Cloud doesn't mean "centralized" anymore. It means "coordinated."

# Trends – Data Contracts, Observability, FinOps

## Data Contracts

- Agreements between producers and consumers
- Define schema, quality expectations, update policies
- Help prevent broken pipelines and "silent failures"



## Data Observability

- Extends monitoring to include freshness and completeness
- Combines metrics, metadata, and pipeline health
- Helps teams debug and trust data at every step

## FinOps for Data

- Visibility into cost per query, pipeline, or dashboard
- Push for better cost/performance balance
- Reduces waste and makes consumption transparent

These aren't just trends — they're becoming **baseline expectations** for modern data platforms.

# How to Get Ready – Start Small, Design for Scale

## Start Small

- Pick one domain team to pilot data product ownership
- Choose one use case to apply active metadata or contracts
- Build incremental value to gain organizational buy-in

## Treat Your Platform as a Product

- Invest in onboarding and self-service
- Track adoption, not just pipeline uptime
- Gather and incorporate user feedback

## Design for Scale

- Use open standards and loosely coupled services
- Build around trust, reuse, and observability
- Document architectural decisions and patterns

## Balance Ambition with Pragmatism

- Don't rebuild everything — modernize selectively
- Prioritize patterns over specific tools
- Focus on business outcomes, not technical purity

We're not just building systems. We're shaping how organizations work with data — starting with smart choices today.

# Key Takeaways

Modern data architecture isn't about a single tool or platform — it's about how we **design for change**.

### Think Cloud-Native

Not just cloud-hosted – Build for scale, automation, and portability

### Decentralize with Purpose

Data Mesh isn't chaos — it's intentional ownership and product thinking

### Automate with Metadata

Let your platform handle governance, discovery, and quality behind the scenes

### Simplify with Lakehouse

One source of truth that supports BI, ML, and real-time use cases

### Prepare for What's Next

Real-time, AI-assisted, edge-aware platforms are becoming the new normal

Modern data platforms don't just support business needs — they help **drive** them.

# What You Can Do Next

## Review Architecture

Where are the bottlenecks? What can be automated? What's still dependent on manual handoffs?

## Identify Domain Team

Empower them to take ownership of a data product — with the right support and standards.

## Invest in Metadata

Start making data easier to find and trust through cataloging, lineage, or observability.

This isn't a big bang transformation. It's a set of smart steps that make your architecture more adaptive, more useful, and more aligned to your business.