An Evolution of Data Platform Architectures in Azure

Lambda, Kappa, Delta, Data Mesh

λ κ δ

Paul Andrew | Technical Architect in Azure CoE

Microsoft MVP Most Valuable Professional

avanade

Mr Paul Andrew Consulting Ltd

@MrPaulAndrew In/MrPaulAndrew MrPaulAndrew.com c/MrPaulAndrew

What is the answer to life, the universe and everything?

Answer:
42 ✗

Answer:
It depends! ✓

What is big data?

Answer:
It depends! ✓

Answer:
"Any data that you cannot process
in the time that you have/want
using the technology you have." ✓

- Buck Woody

@BuckWoodyMSFT

Volume
Velocity
Variety
Veracity
Value

What is the goal of our data solutions?

Data
Collection

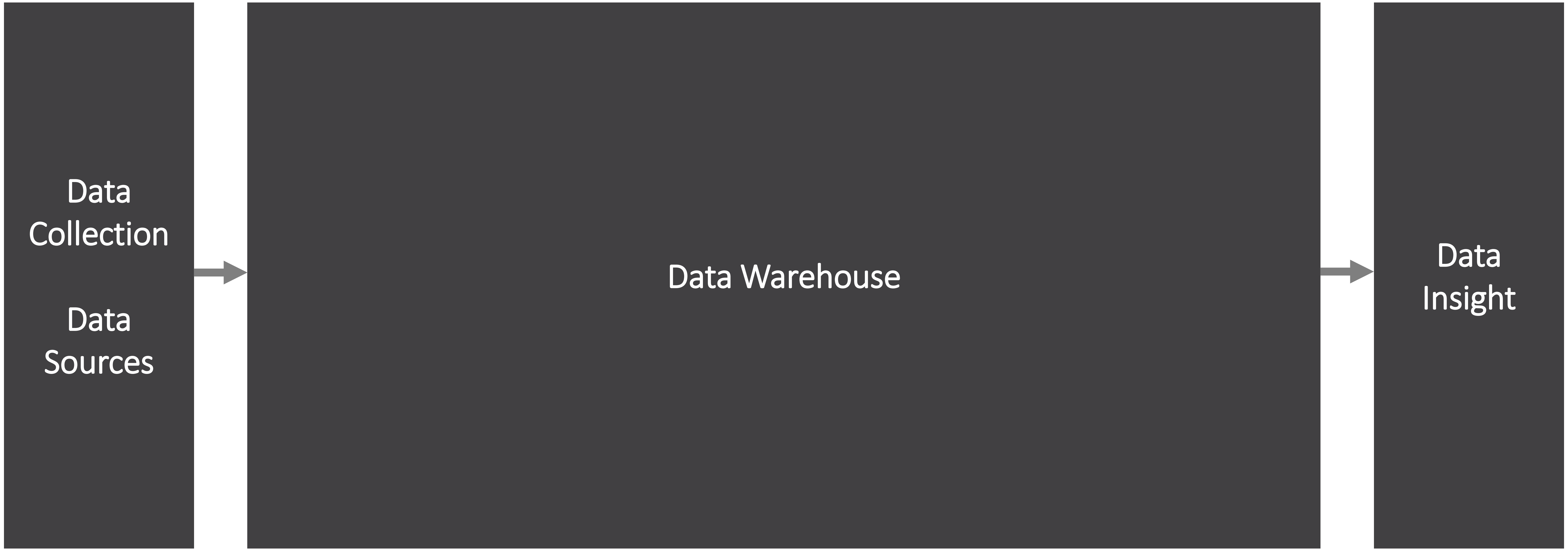
Data
Sources

Paul's Magic Box -
From the Hogwarts School of Witches & Wizardry

Data
Insight

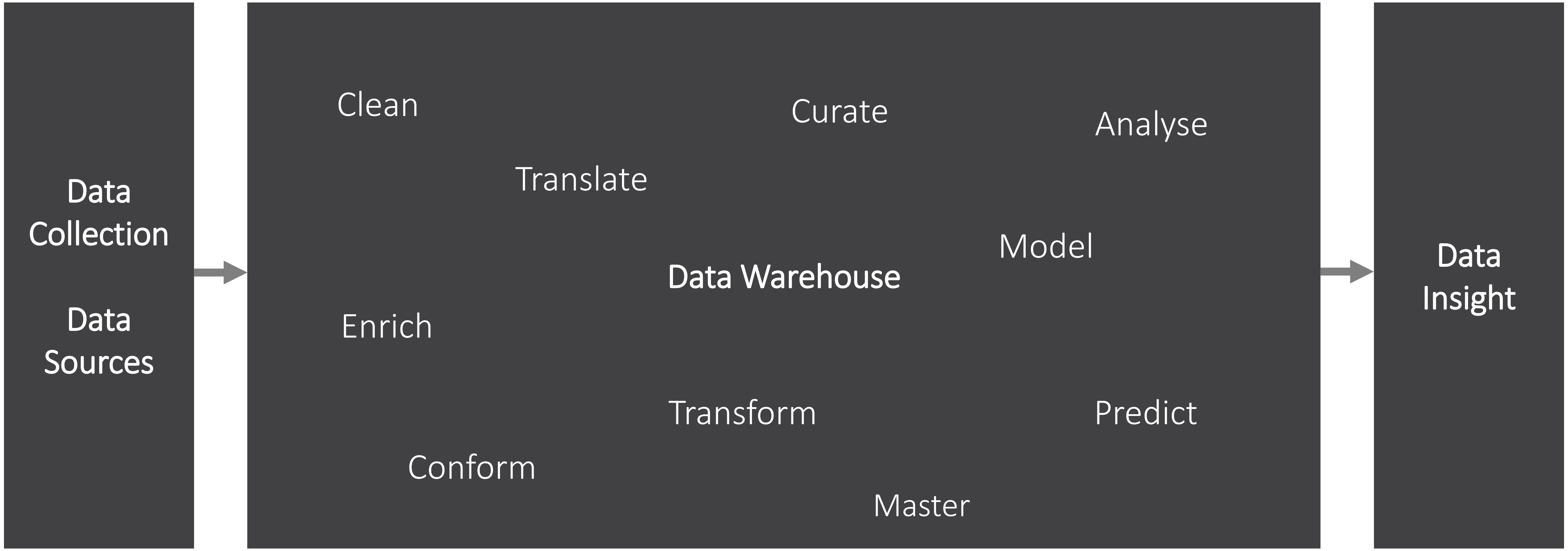
Data = Information = Knowledge = Power

What is the goal of our data solutions?



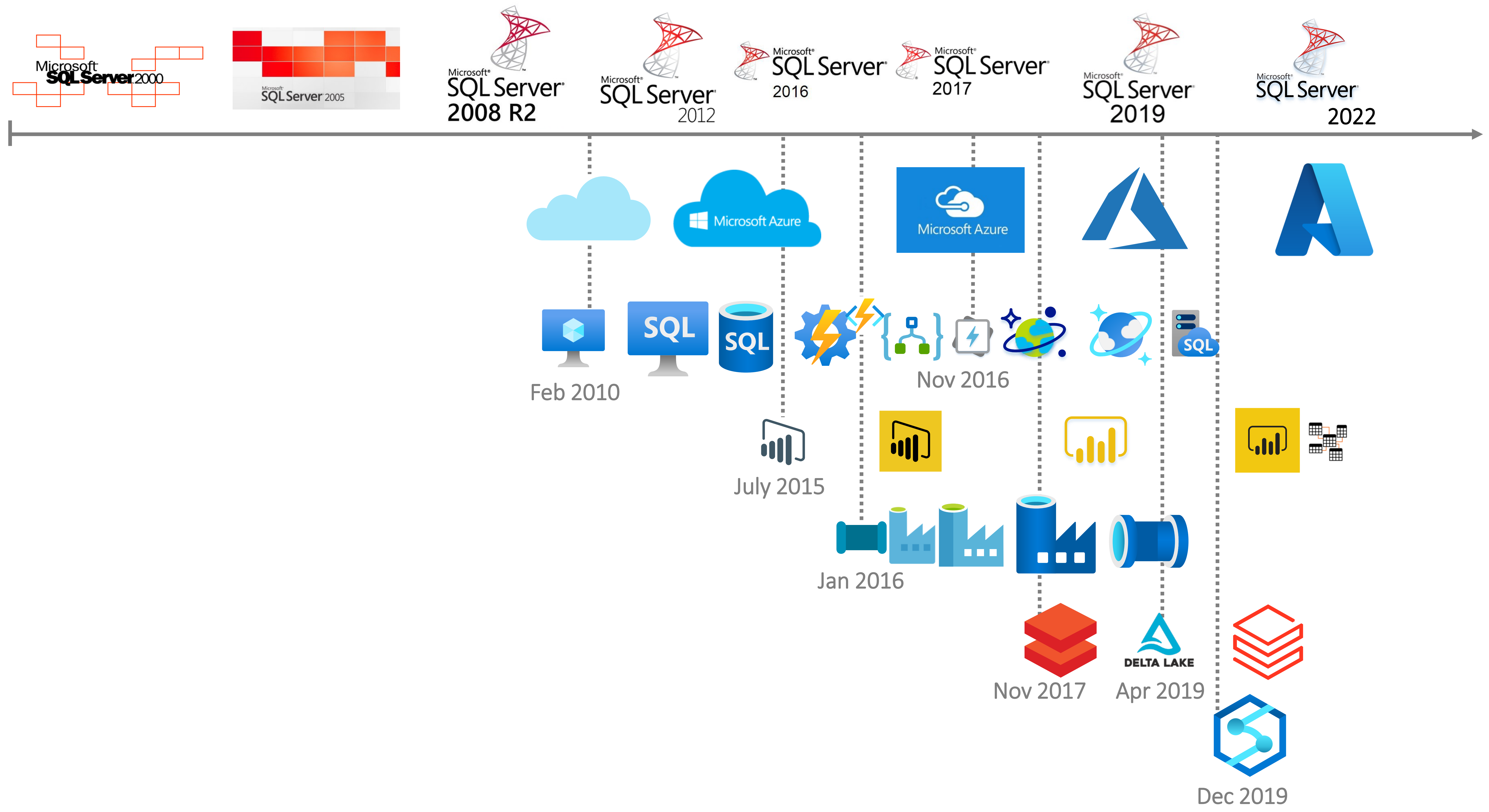
Data = Information = Knowledge = Power/Insights

What is the goal of our data solutions?

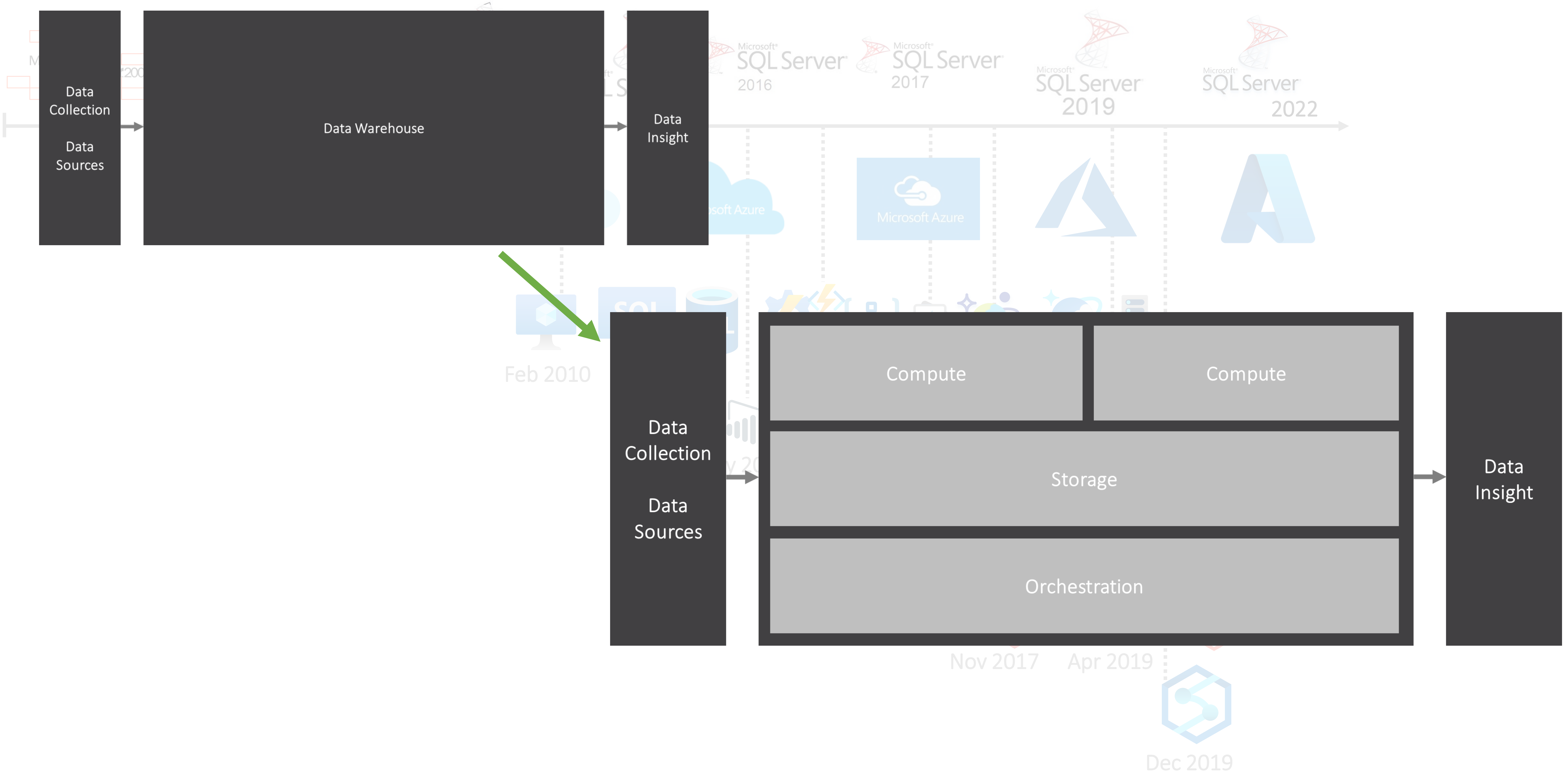


Data = Information = Knowledge = Power/Insights

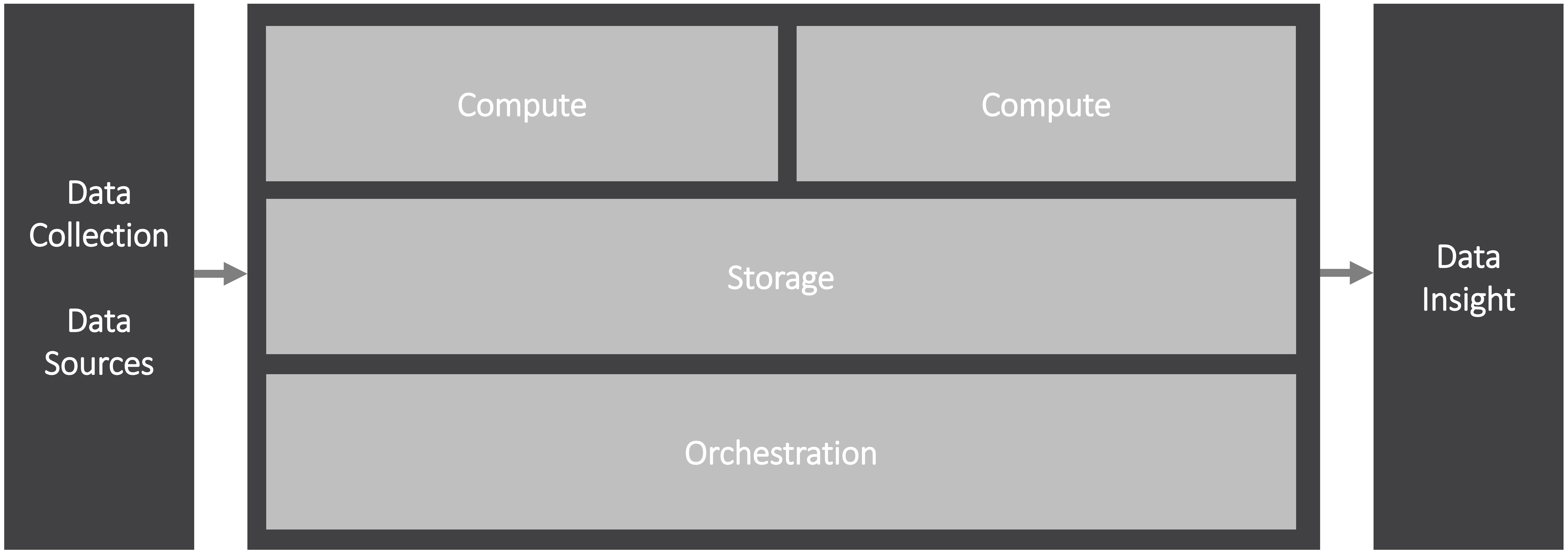
An Evolution of Data Platforms


Microsoft SQL Server 2000
Microsoft SQL Server 2005
Microsoft SQL Server 2008 R2
Microsoft SQL Server 2012
Microsoft SQL Server 2016
Microsoft SQL Server 2017
Microsoft SQL Server 2019
Microsoft SQL Server 2022
Microsoft Azure
Microsoft Azure
SQL
SQL
Feb 2010
Nov 2016
SQL
July 2015
Jan 2016
Nov 2017
DELTA LAKE
Apr 2019
Dec 2019

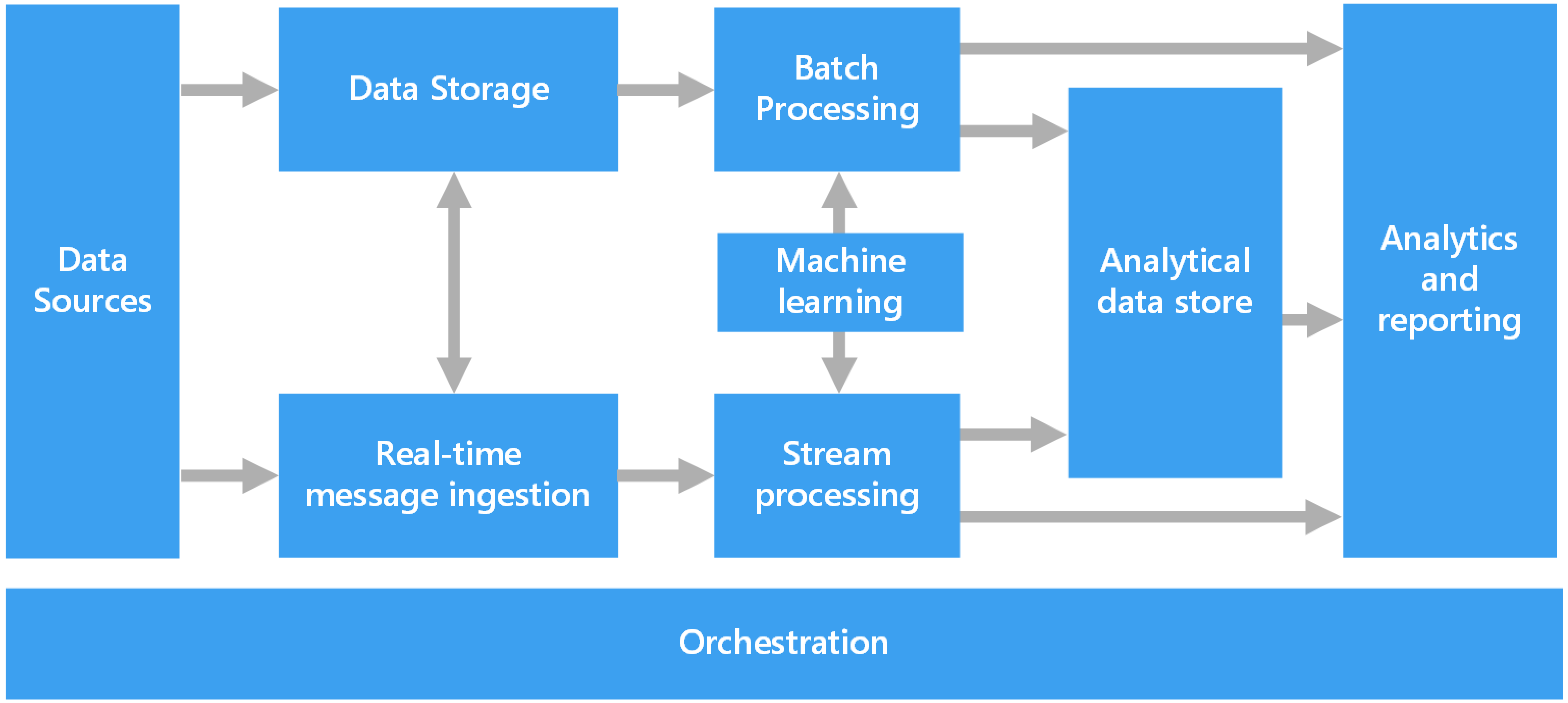
An Evolution of Data Platforms



A Reference Architecture?

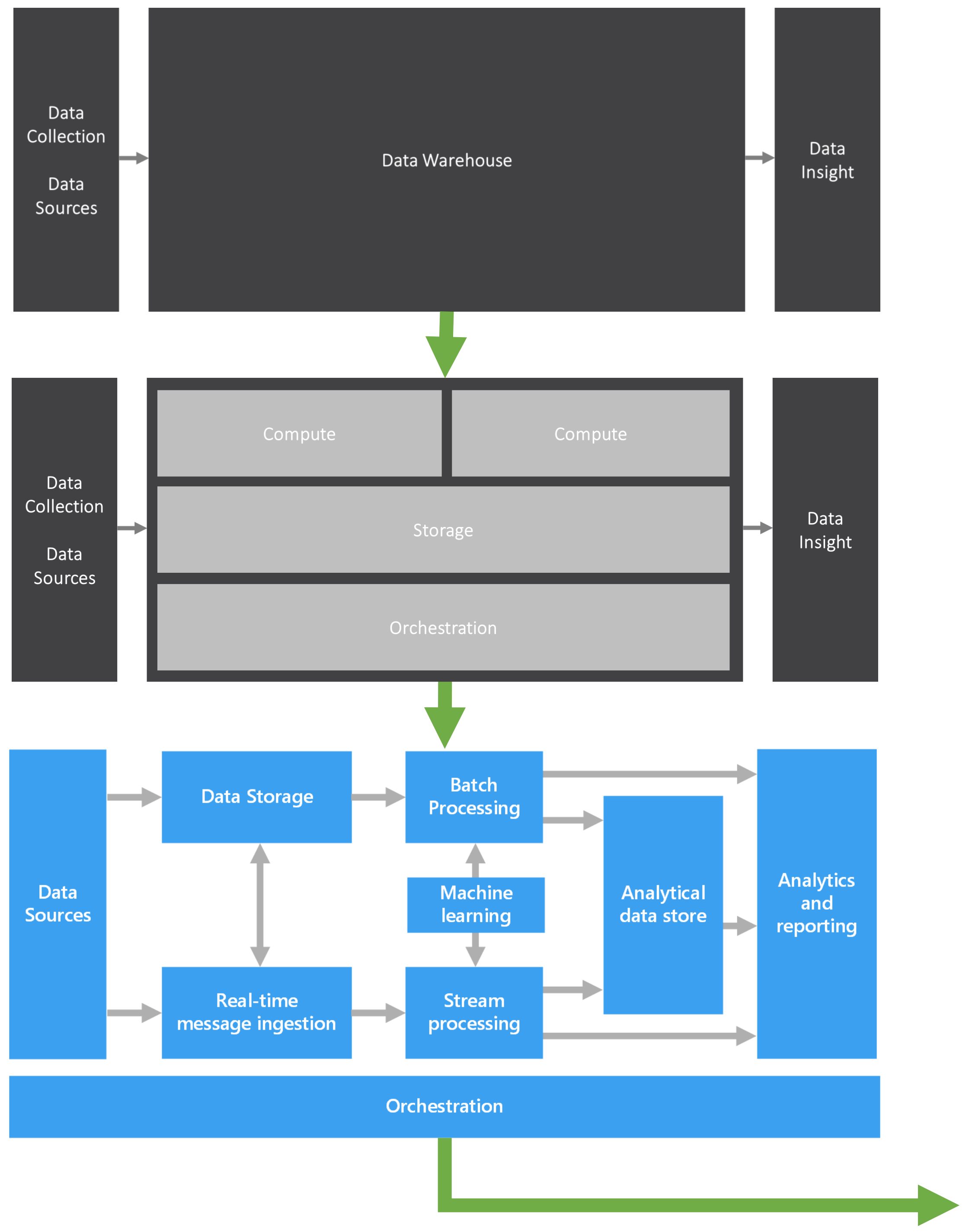


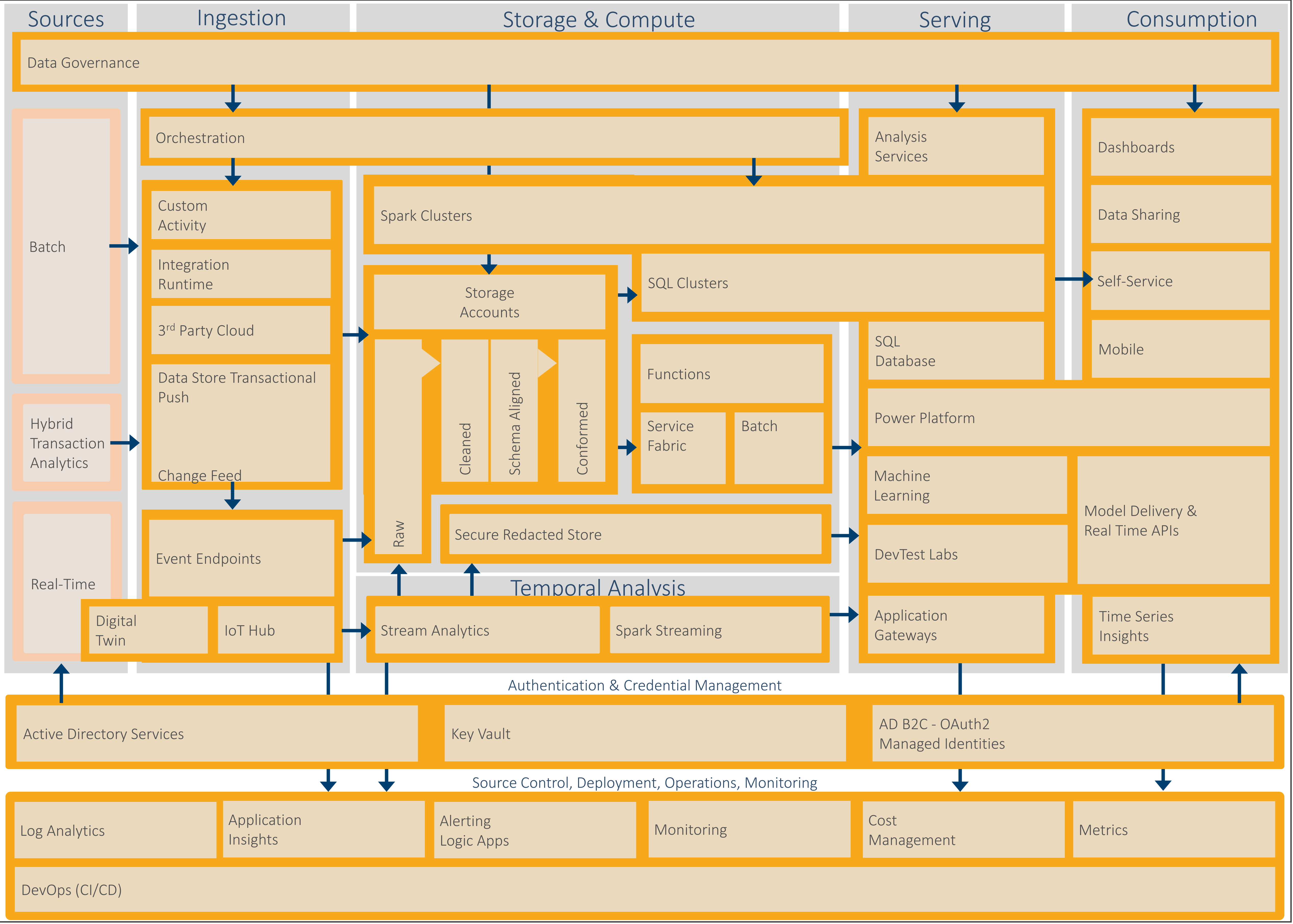
Microsoft's Components of a ~~Big~~ Data Architecture



https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/

A Logical Data Architecture





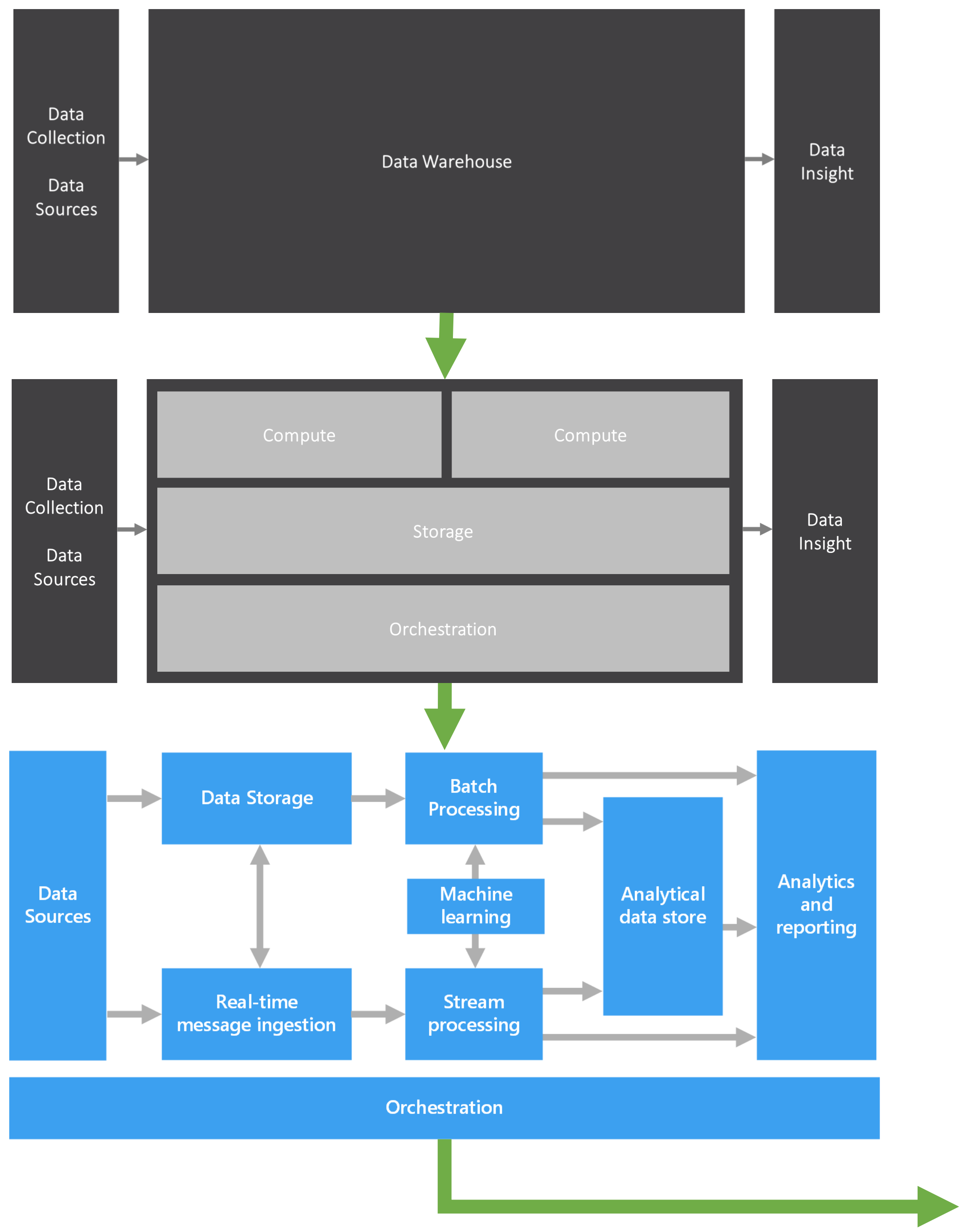
▨ Concepts

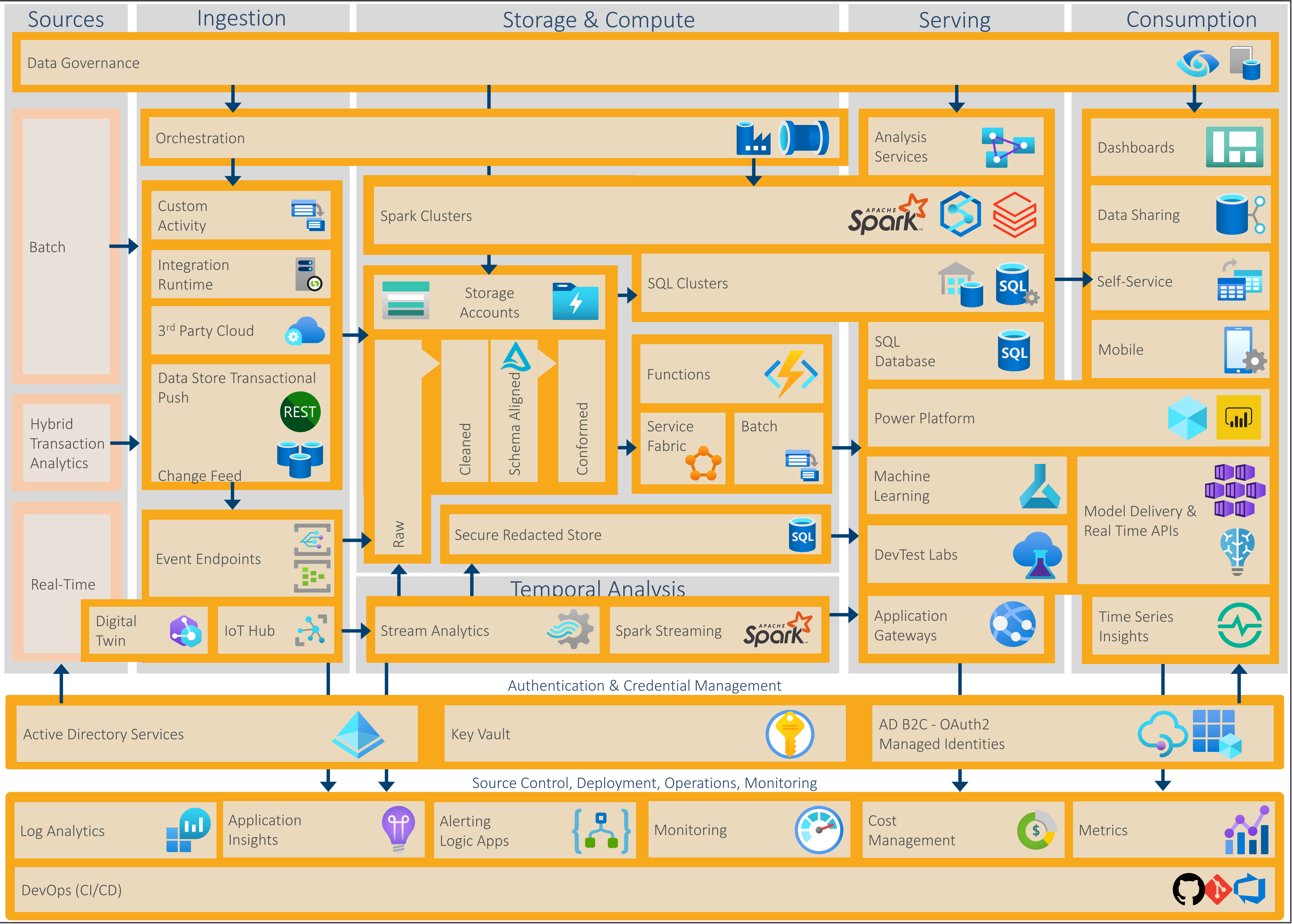
🟧 Resources

▨ Layers

avanade

A Logical Data Architecture





🟧 Concepts
🟧 Resources
⬜ Layers

avanade

Delta* Lake

Delta.io

* We are not talking about the delta of changed records since our data processing last ran.

Databases

DataBase

Management

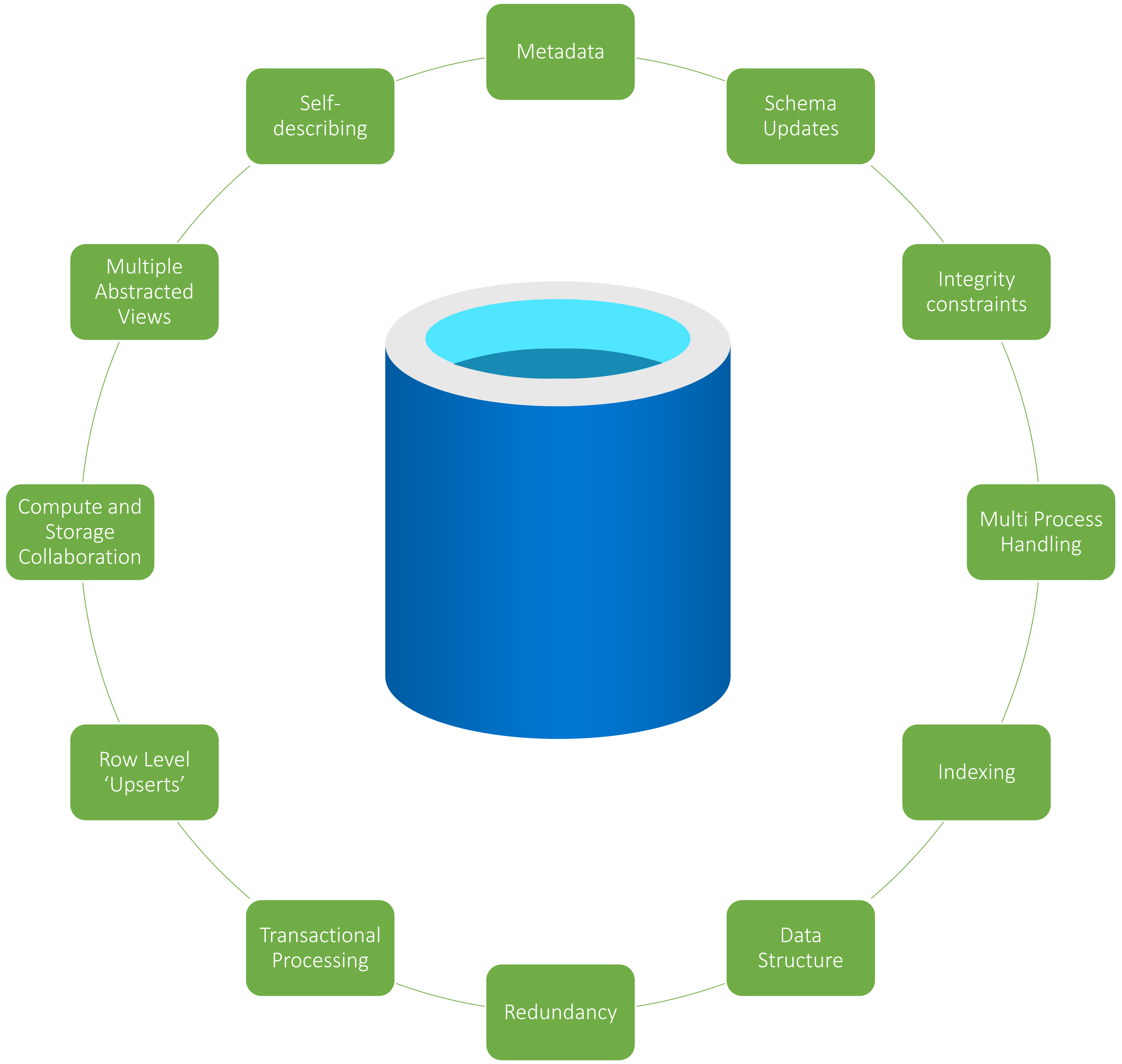
System

Atomicity

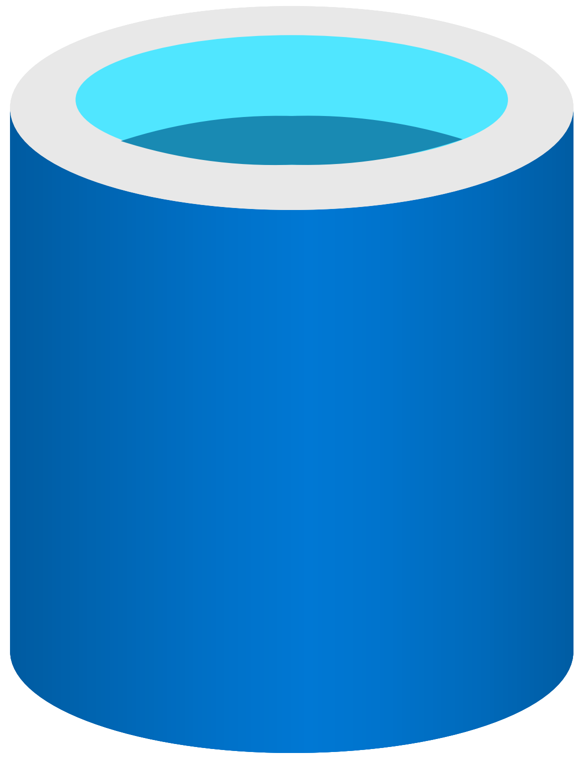
Consistency

Isolation

Durability



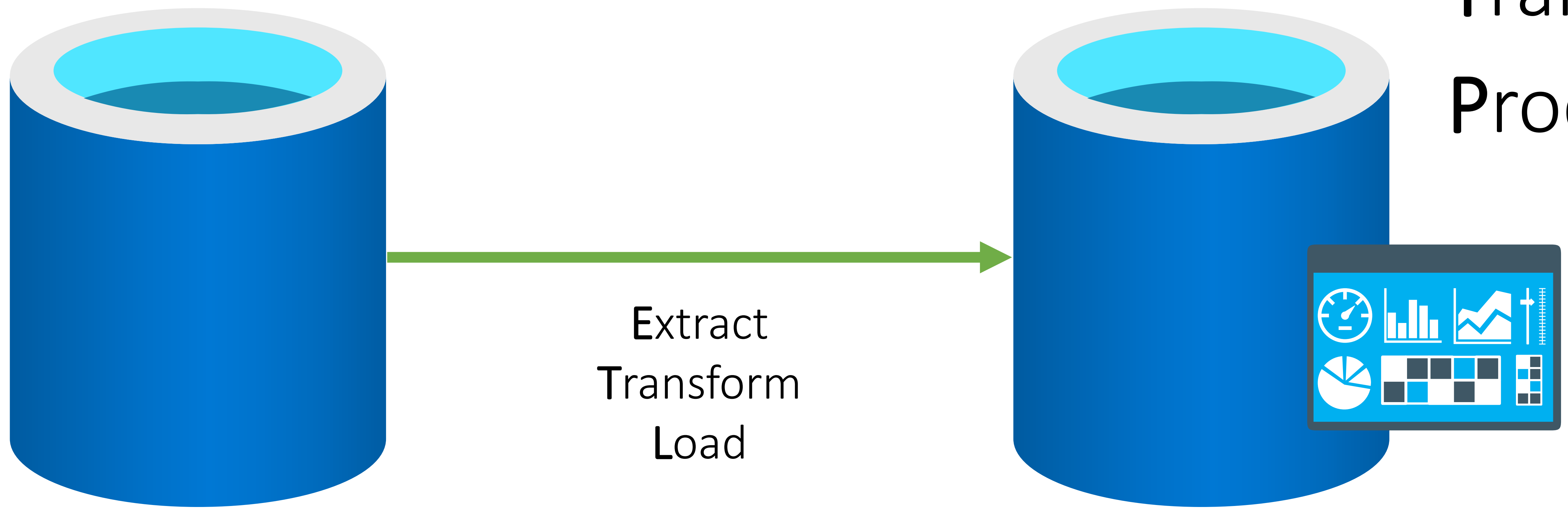
Databases



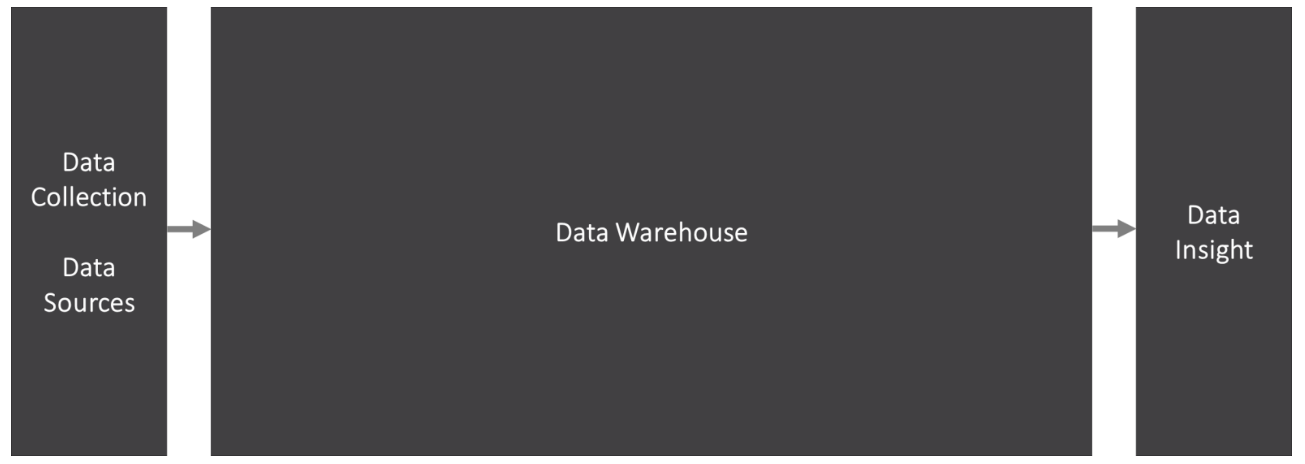
Creating a Data Warehouse

Online
Line
Transactional
Processing

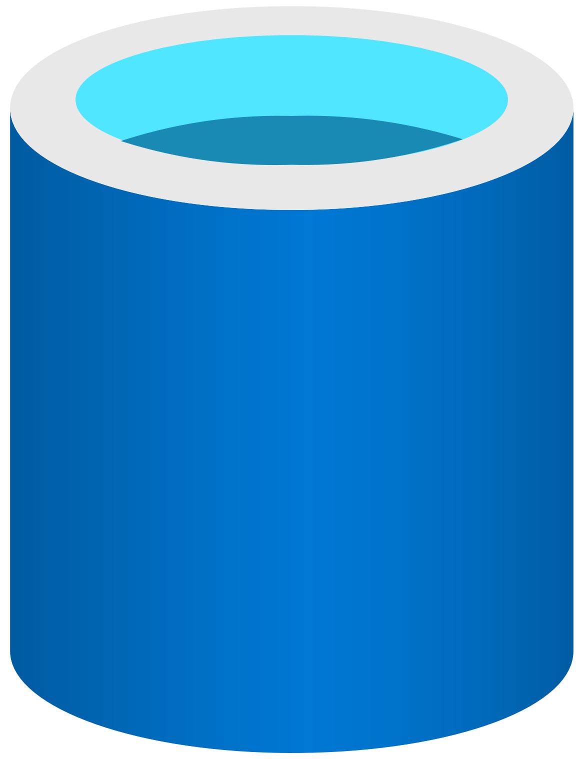
Offline
Analytical
Transactional
Processing



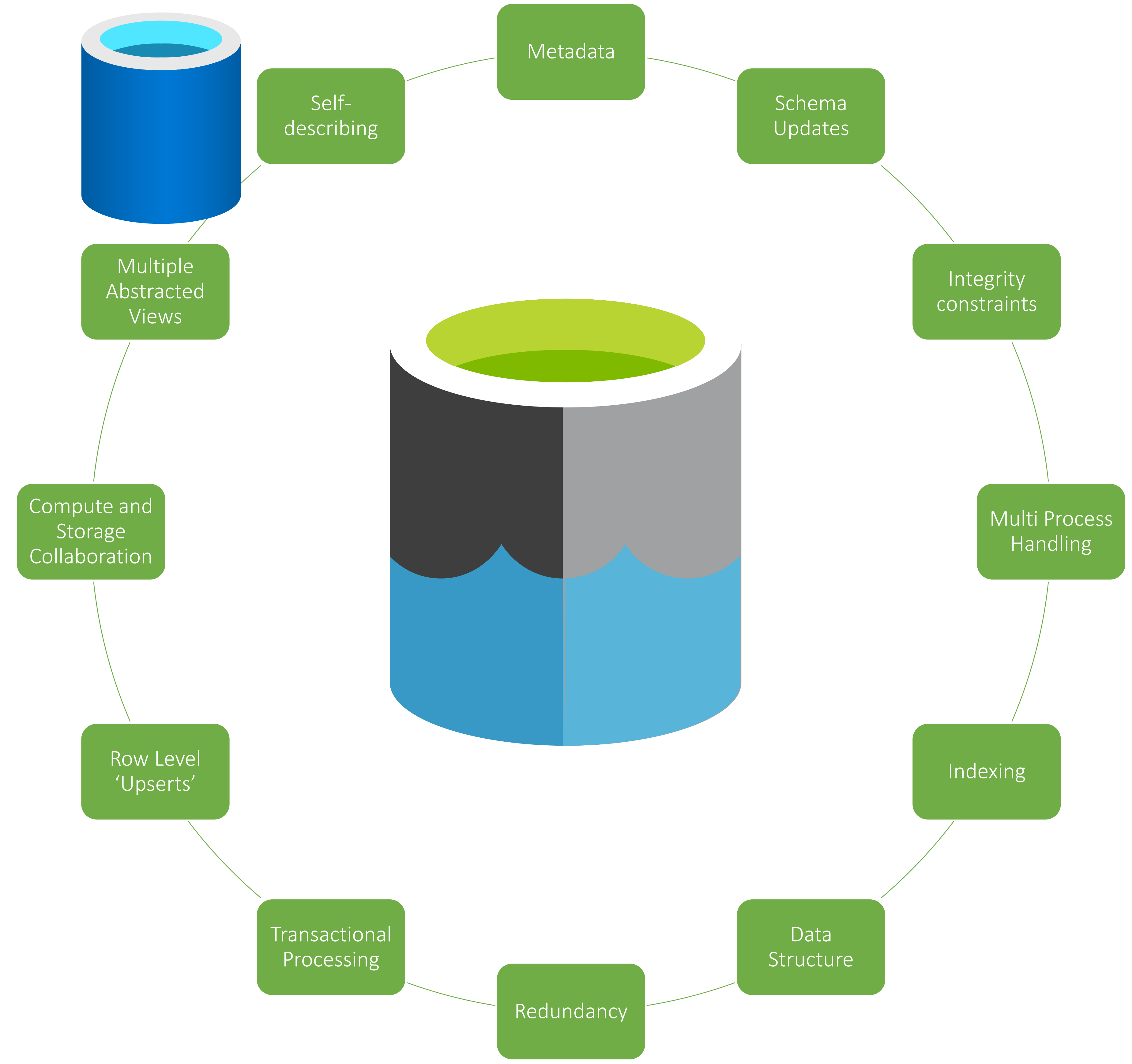
Application
Data



Databases

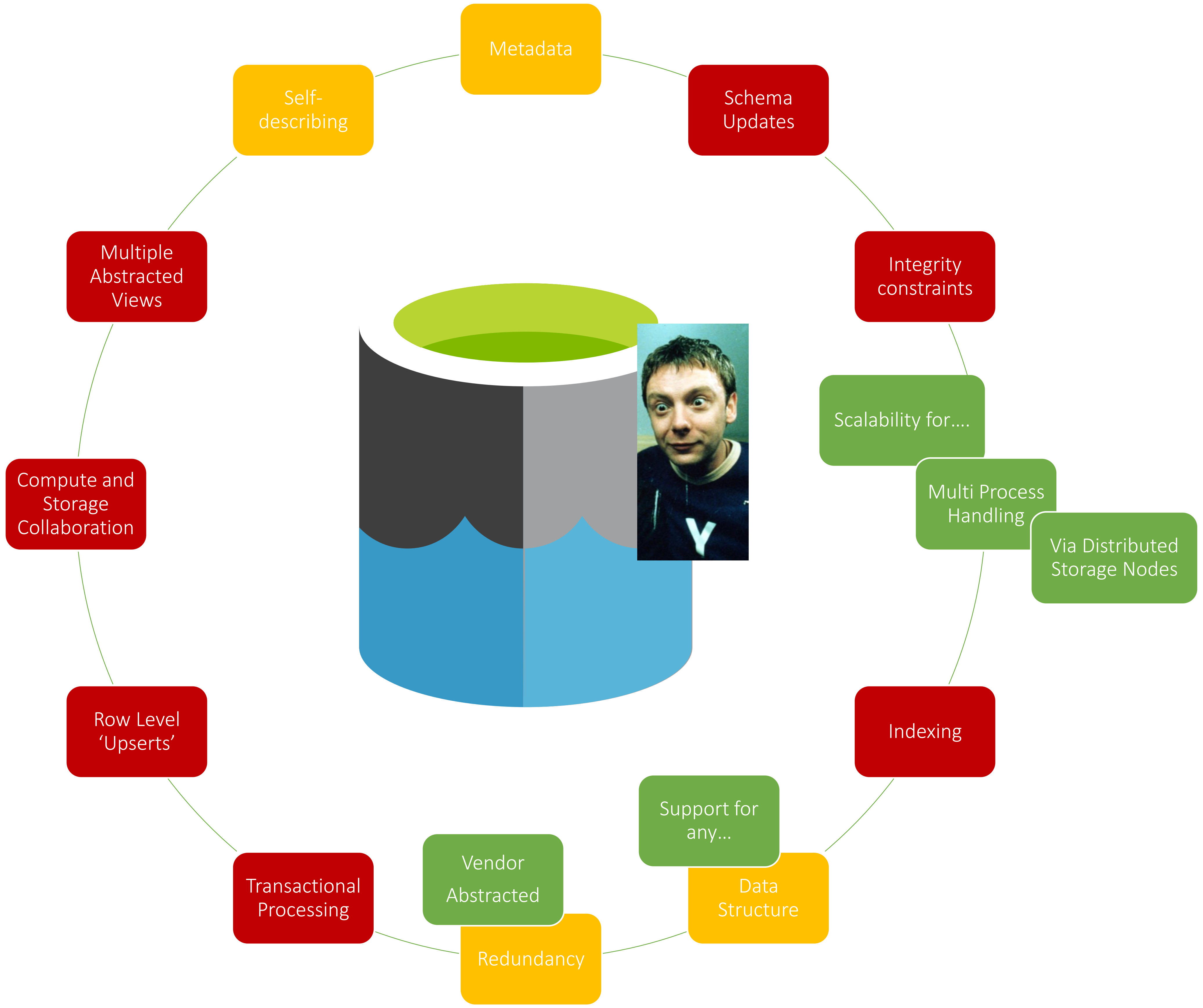


Data Lakes



Volume
Velocity
Variety
Veracity
Value

Data Lakes

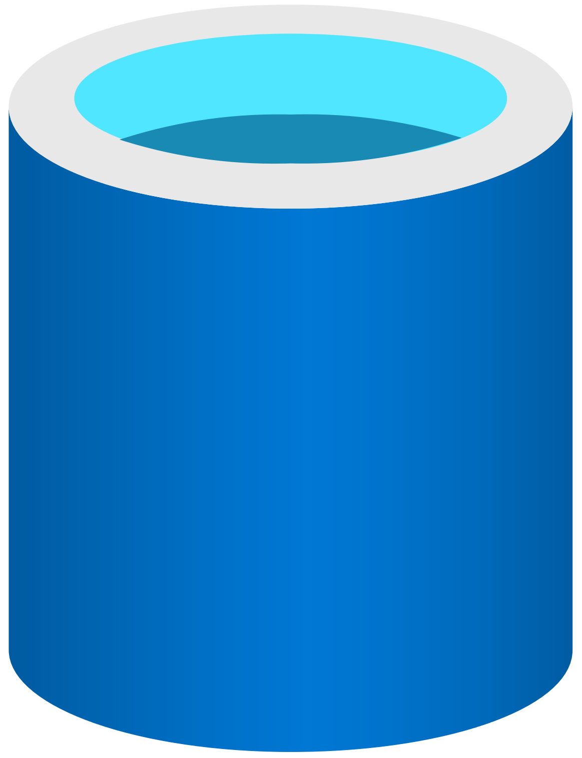


Volume
Velocity
Variety
Veracity
Value

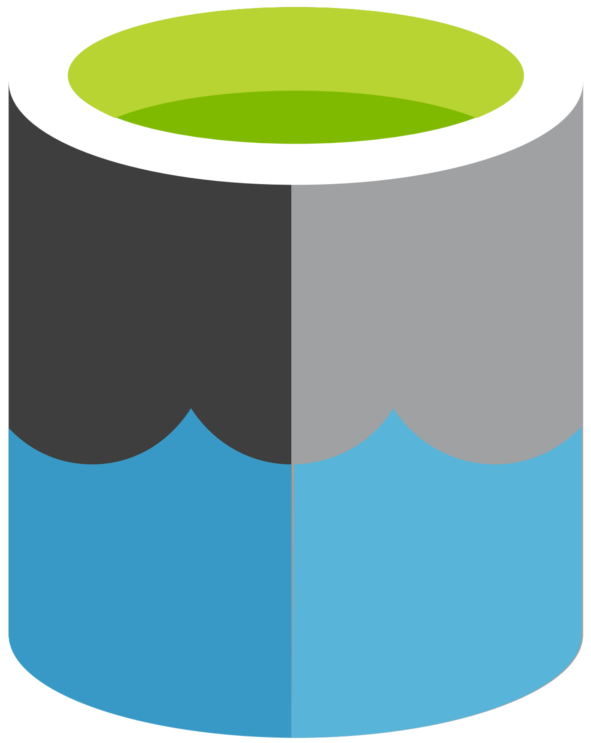
Problem Summary

Data Lakes are good, but they still lack some of the basic ACID functionality needed for data processing.

We are/were trying to use Data Lakes for everything (to replace Databases).



VS

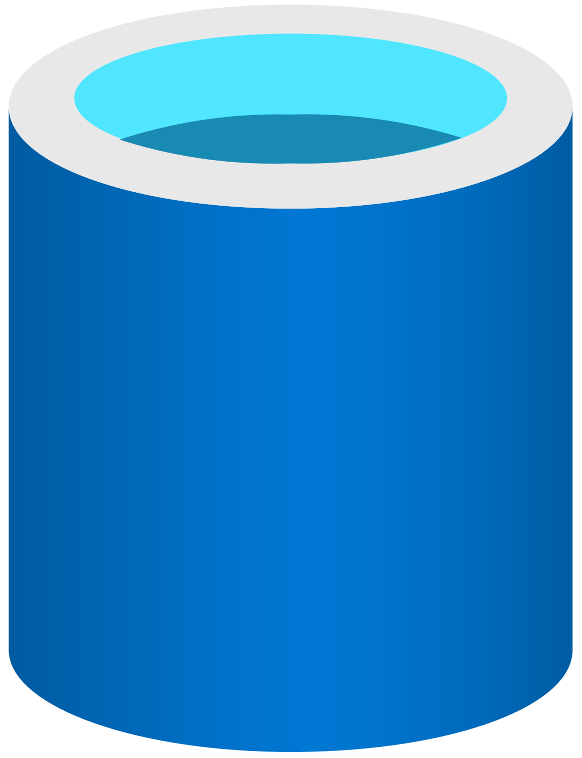


Scales Up	Scales Out
Natural Home for Structured Data	Any Data Structure
Storage Limits	No Storage Limits
Transactional Resilience	No Transactional Handling
Storage & Compute Coupled	Storage & Compute Decoupled

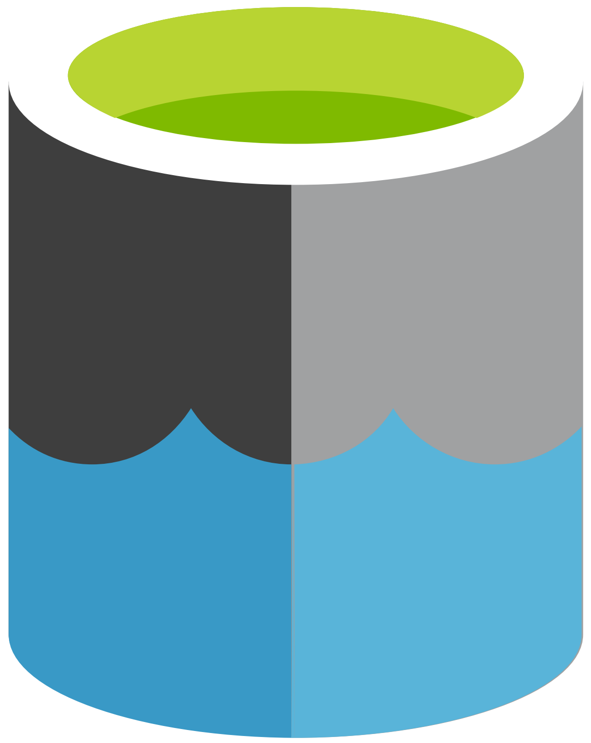
Problem Summary

Data Lakes are good, but they still lack some of the basic ACID functionality needed for data processing.

We are/were trying to use Data Lakes for everything (to replace Databases).



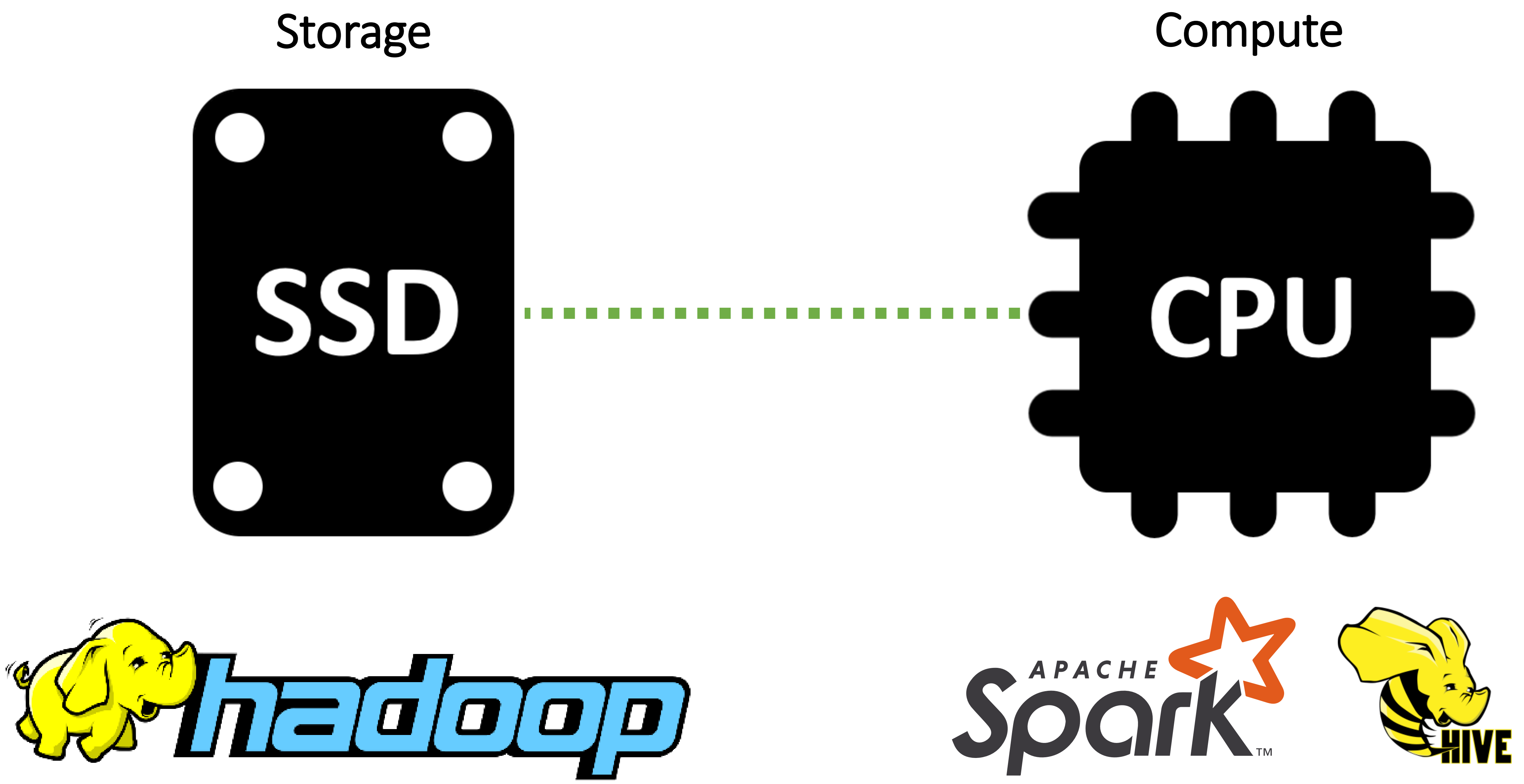
VS



Scales Up	Scales Out
Natural Home for Structured Data	Any Data Structure
Storage Limits	No Storage Limits
Transactional Resilience	No Transactional Handling
Storage & Compute Coupled	Storage & Compute Decoupled

Solution

Just enable ACID transactional support for Data Lakes...



Storage & Compute ~~Decoupled~~ Working Together Again As Friends!

ACID Data Frameworks for Data Lakes

DELTA LAKE™ ← databricks®

February 2019

ICEBERG ← NETFLIX

Apache hudi ← Uber

What is Delta Lake?

DELTA LAKE™ ← databricks®

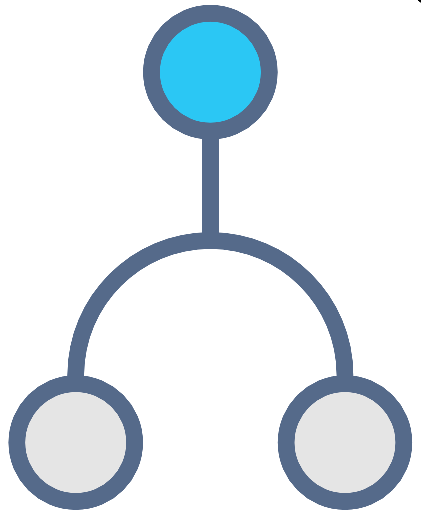
What is Delta Lake?

DELTA LAKE™

https://delta.io

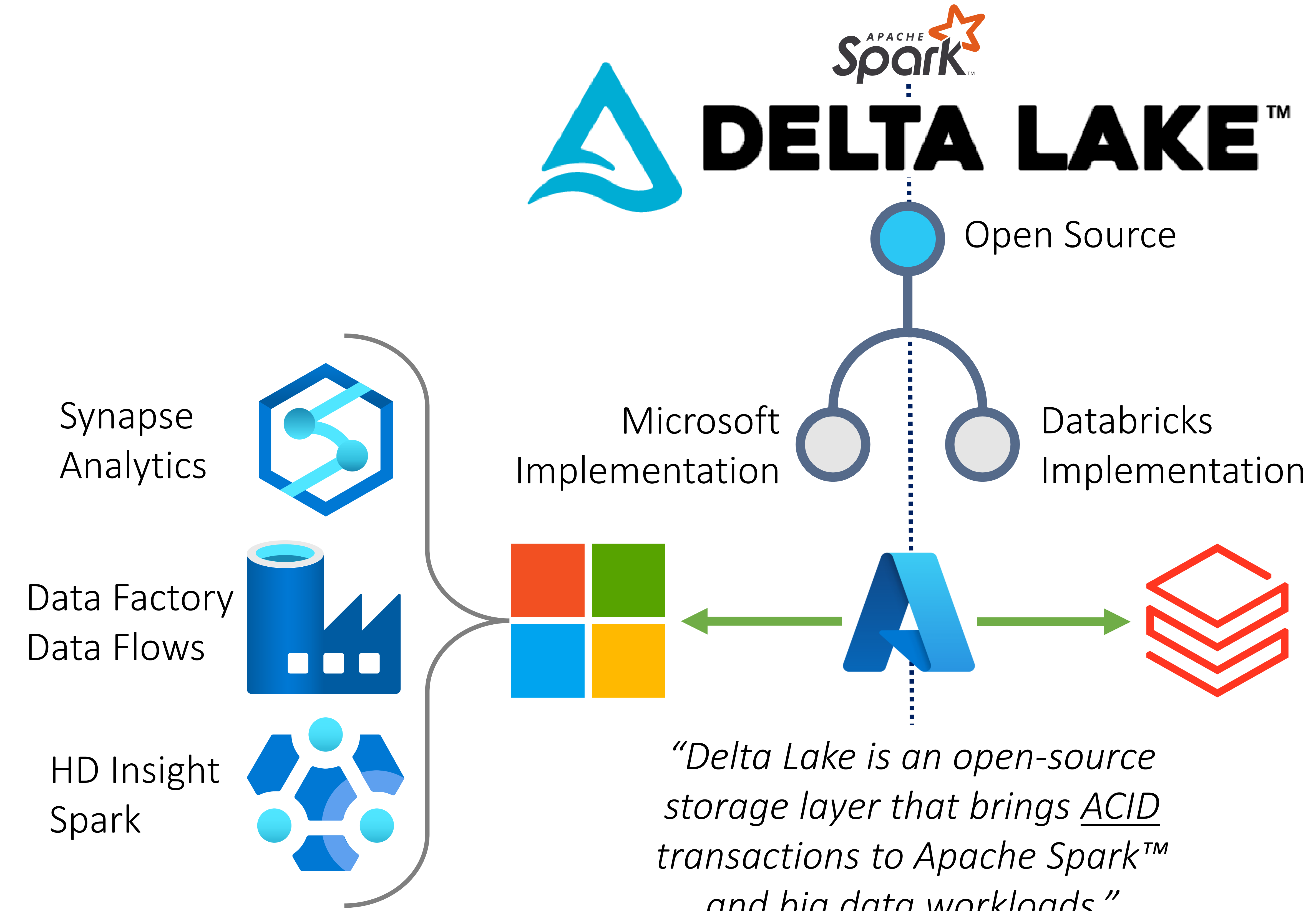
https://github.com/delta-io/delta

April 2019



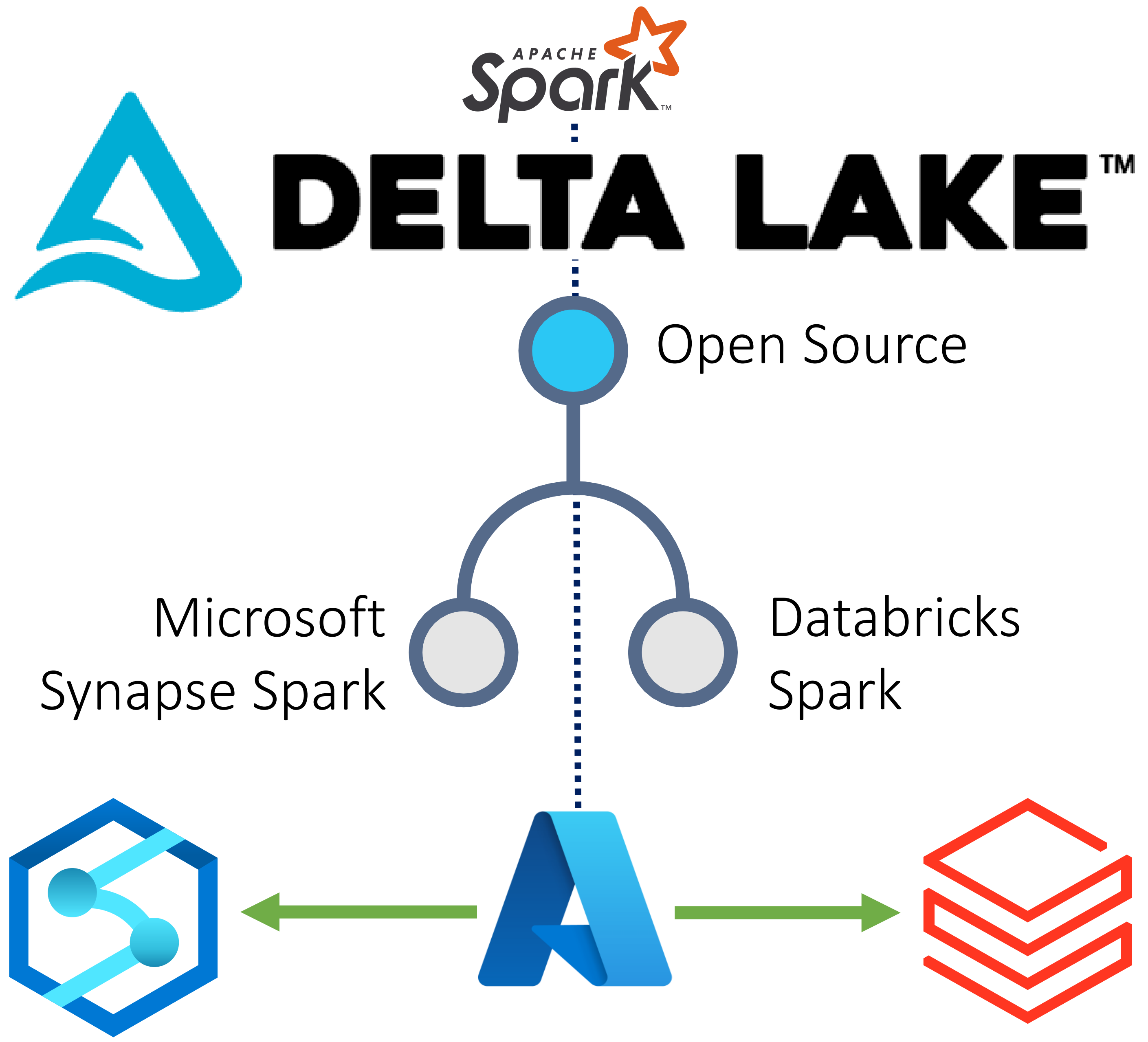
databricks®

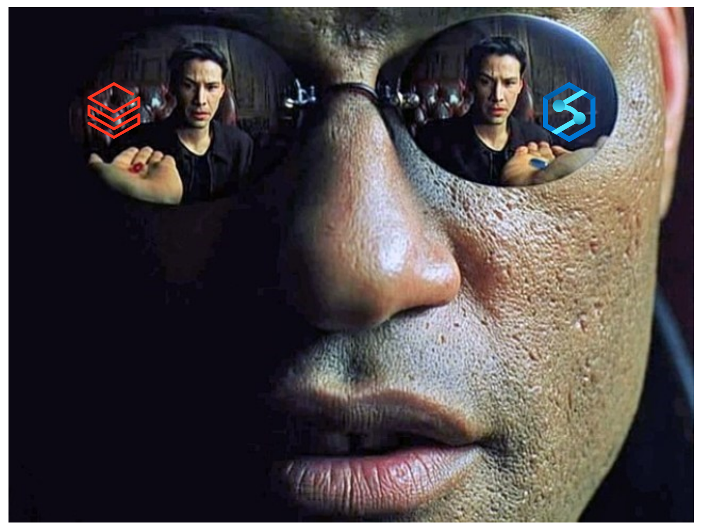
What is Delta Lake?



"Delta Lake is an open-source storage layer that brings ACID transactions to Apache Spark™ and big data workloads."

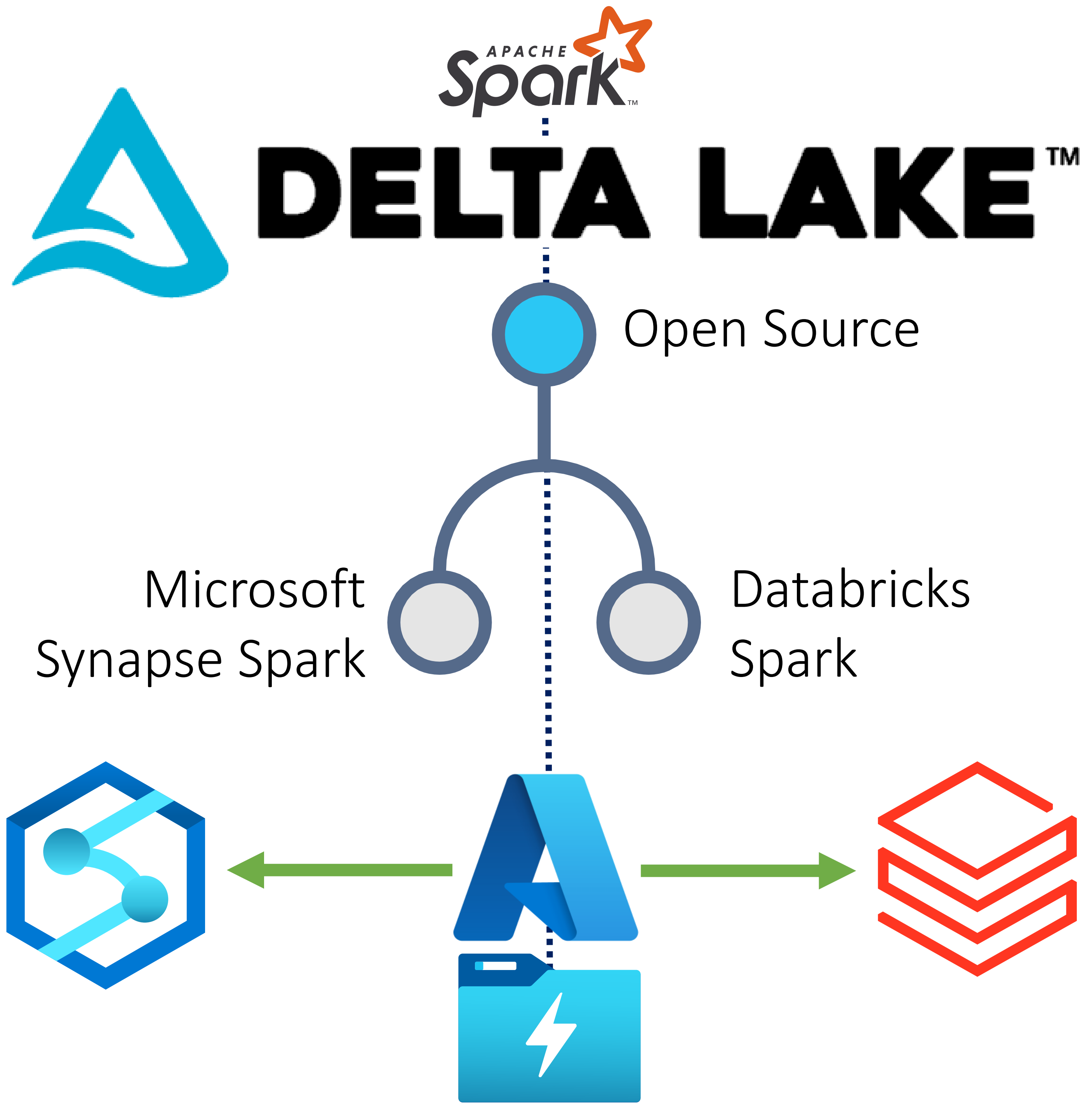
Which Spark Implementation is Better?



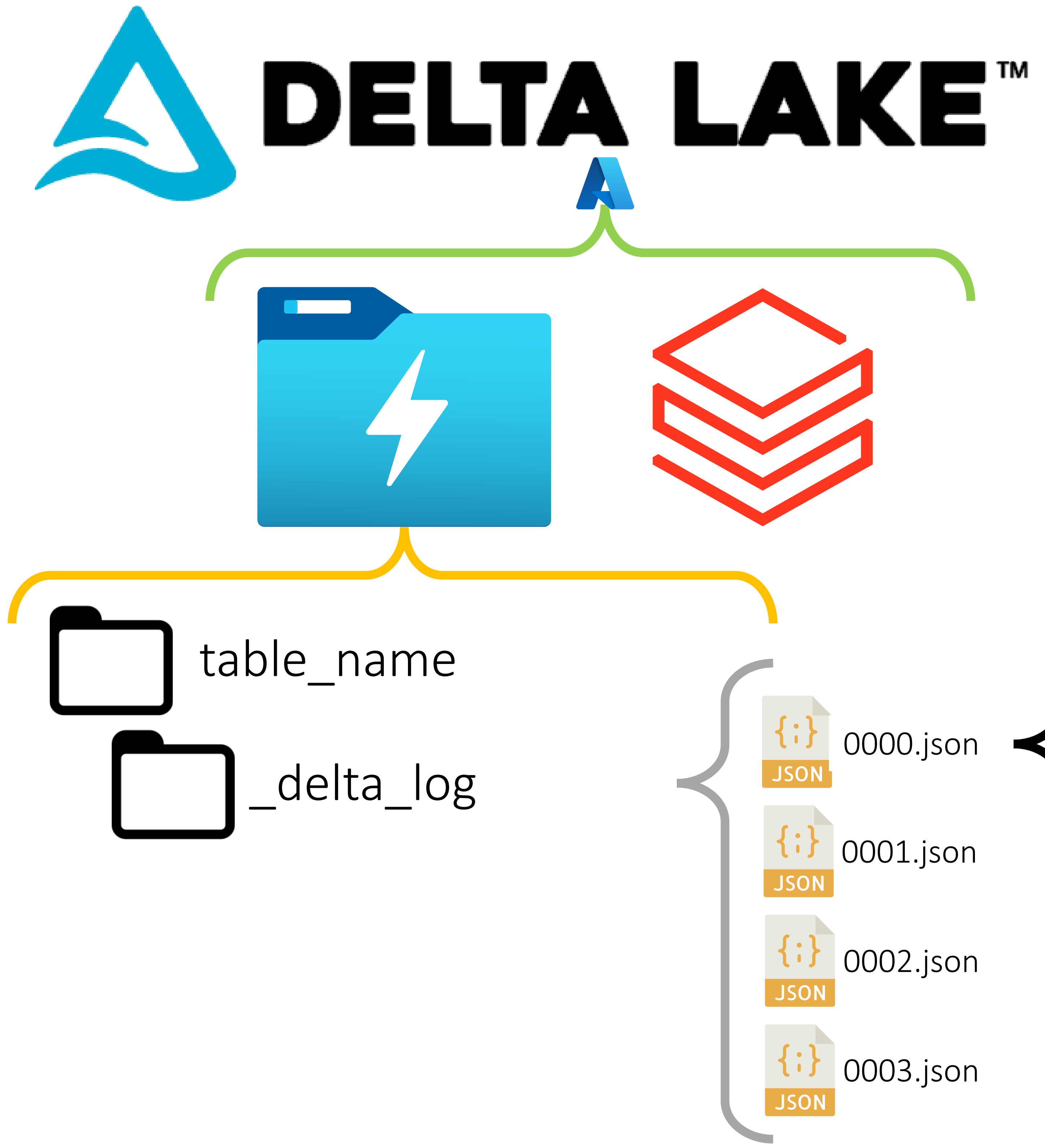


"Delta Lake is an open-source storage layer that brings <u>ACID</u> transactions to Apache Spark™ and big data workloads."

Which Spark Implementation is Better?

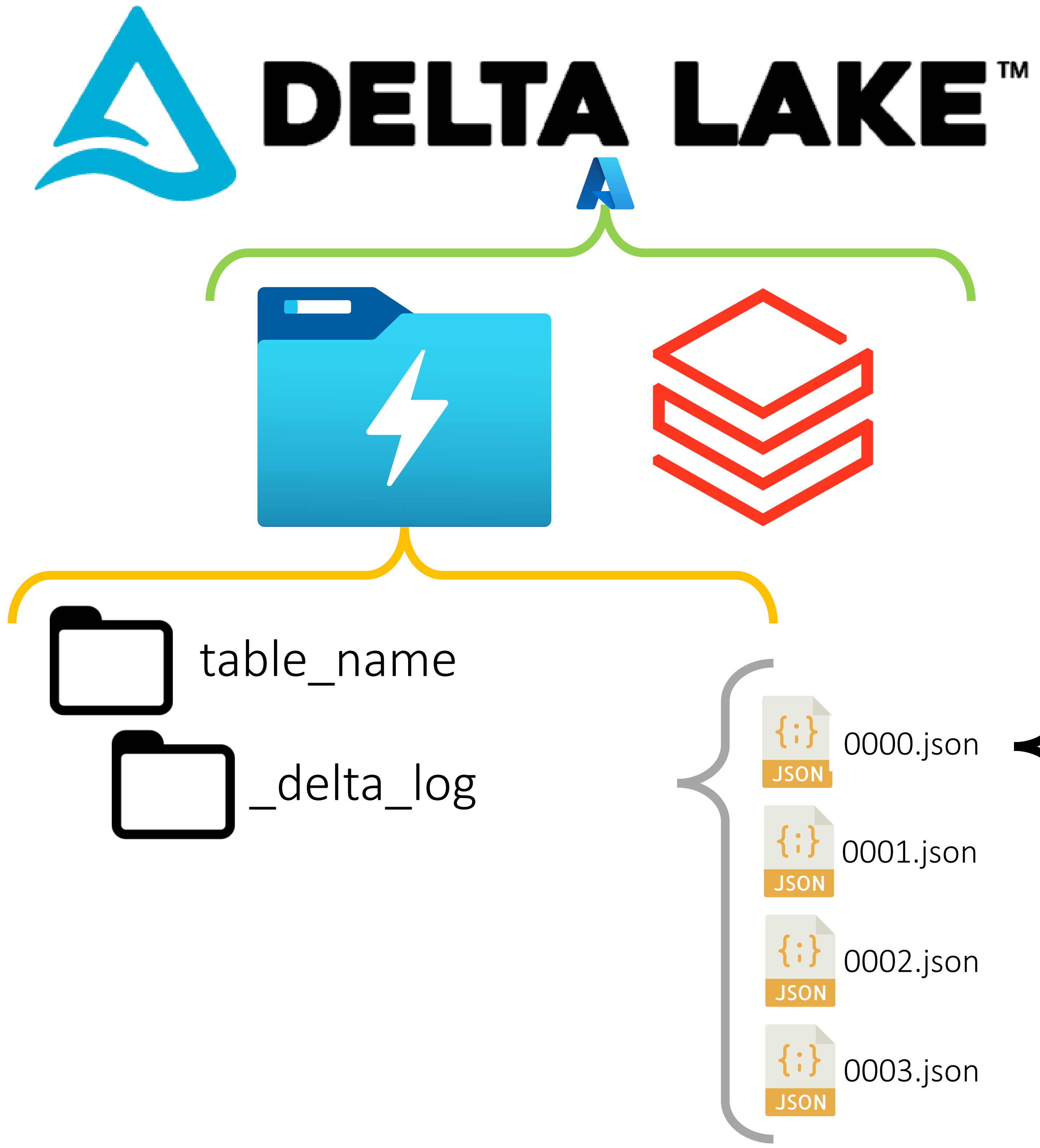


Delta Tables – On Disk



```
{
    "commitInfo":
    {
        "timestamp":1628596034417,
        "operation":"WRITE",
        "operationParameters":
        {
            "mode":"ErrorIfExists",
            "partitionBy":"[]"
        },
        "isBlindAppend":true,
        "operationMetrics":
            {
                "numFiles":"6",
                "numOutputBytes":"2407",
                "numOutputRows":"5"
            }
    }
}
{"protocol":{"minReaderVersion":1,"minWriterVersion":2}}
{
    "metaData":
    {
    "id":"58e5de01-de72-4d5b-a208-d0b4ae919efe",
    "format":
    {
        "provider":"parquet",
        "options":{}
    },
    "schemaString":
                "{\"type\":\"struct\",\"fields\":[{\"name\":\"id\",\"type\":\"long\",
                 \"nullable\":true,\"metadata\":{}}]}",
    "partitionColumns":[],
    "configuration":{},
    "createdTime":1628596029470
    }
}
{"add":{"path":"part-00000.snappy.parquet","size":262,"modificationTime":1628596034000}}
{"add":{"path":"part-00001.snappy.parquet","size":429,"modificationTime":1628596034000}}
{"txn":{"appId":"731b2c96-bf64-445c-8ca8-cd6cad6735e2","lastUpdated":1628596094191}}
{"add":{"path":"part-00000.snappy.parquet","size":429,"modificationTime":1628596094000}}
{"add":{"path":"part-00001.snappy.parquet","size":429,"modificationTime":1628596094000}}
{"remove":{"path":"part-00150.snappy.parquet","deletionTimestamp":1628596098597}}
{"remove":{"path":"part-00128.snappy.parquet","deletionTimestamp":1628596098597}}
```

Delta Tables – On Disk



Breaking Down Transactions Into Atomic Commits

Whenever a user performs an operation to modify a table (such as an INSERT, UPDATE or DELETE), Delta Lake breaks that operation down into a series of discrete steps composed of one or more of the actions below.

Add file – adds a data file.

Remove file – removes a data file.

Update metadata – Updates the table's metadata (e.g., changing the table's name, schema or partitioning).

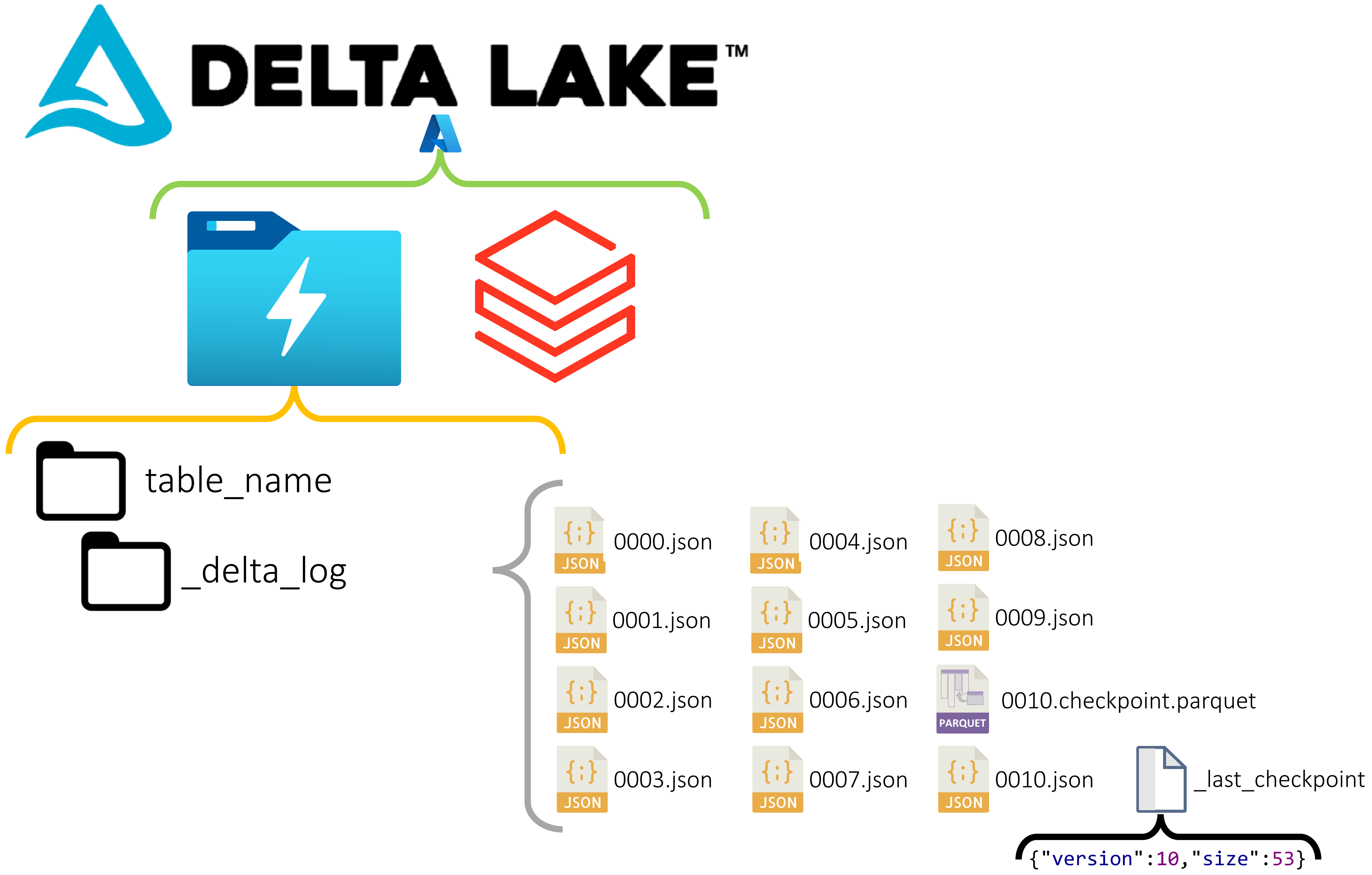
Set transaction – Records that a structured streaming job has committed a micro-batch with the given ID.

Change protocol – enables new features by switching the Delta Lake transaction log to the newest software protocol.

Commit info – Contains information around the commit, which operation was made, from where and at what time.

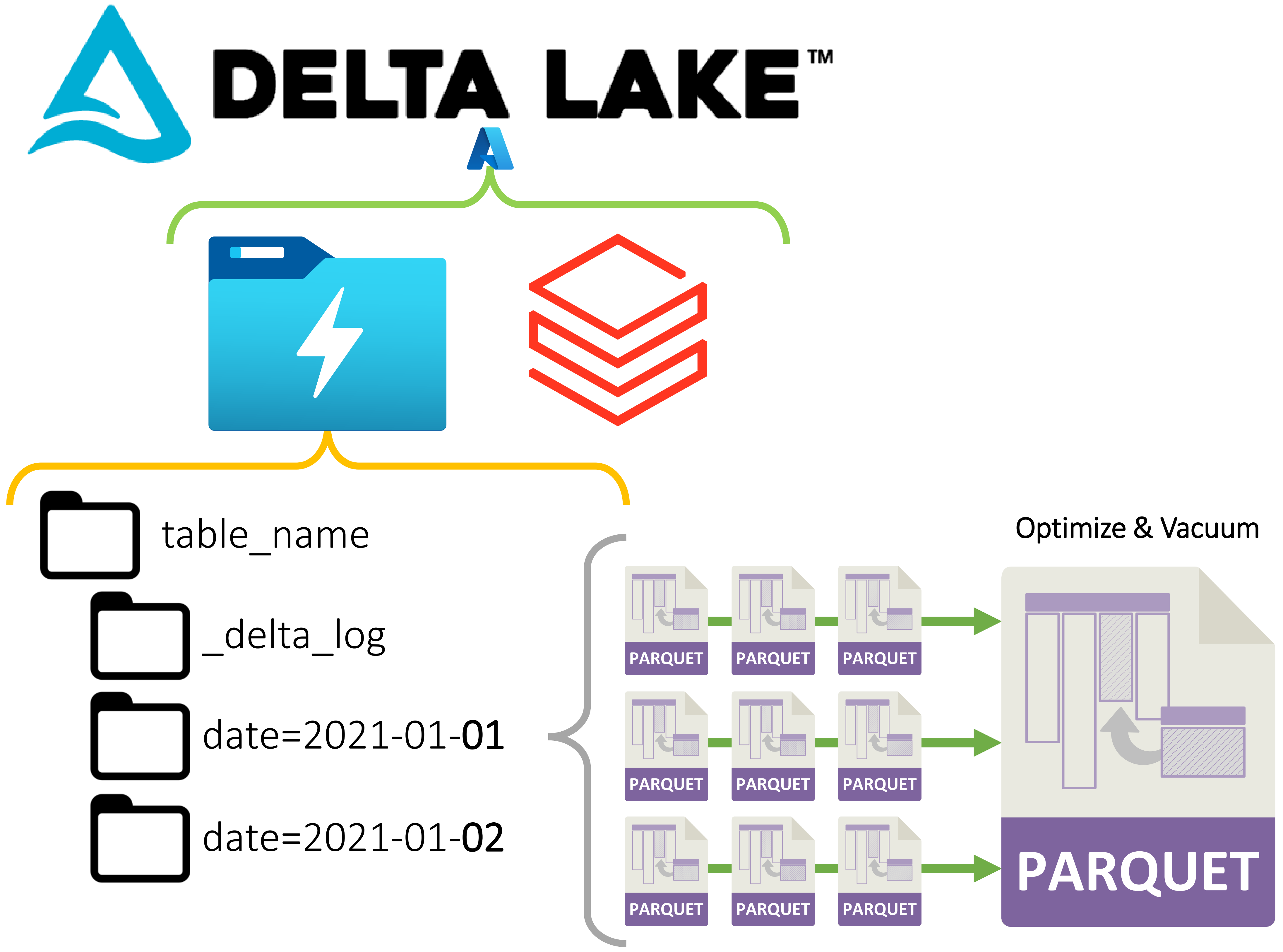
Source: https://databricks.com/blog

Delta Tables – On Disk

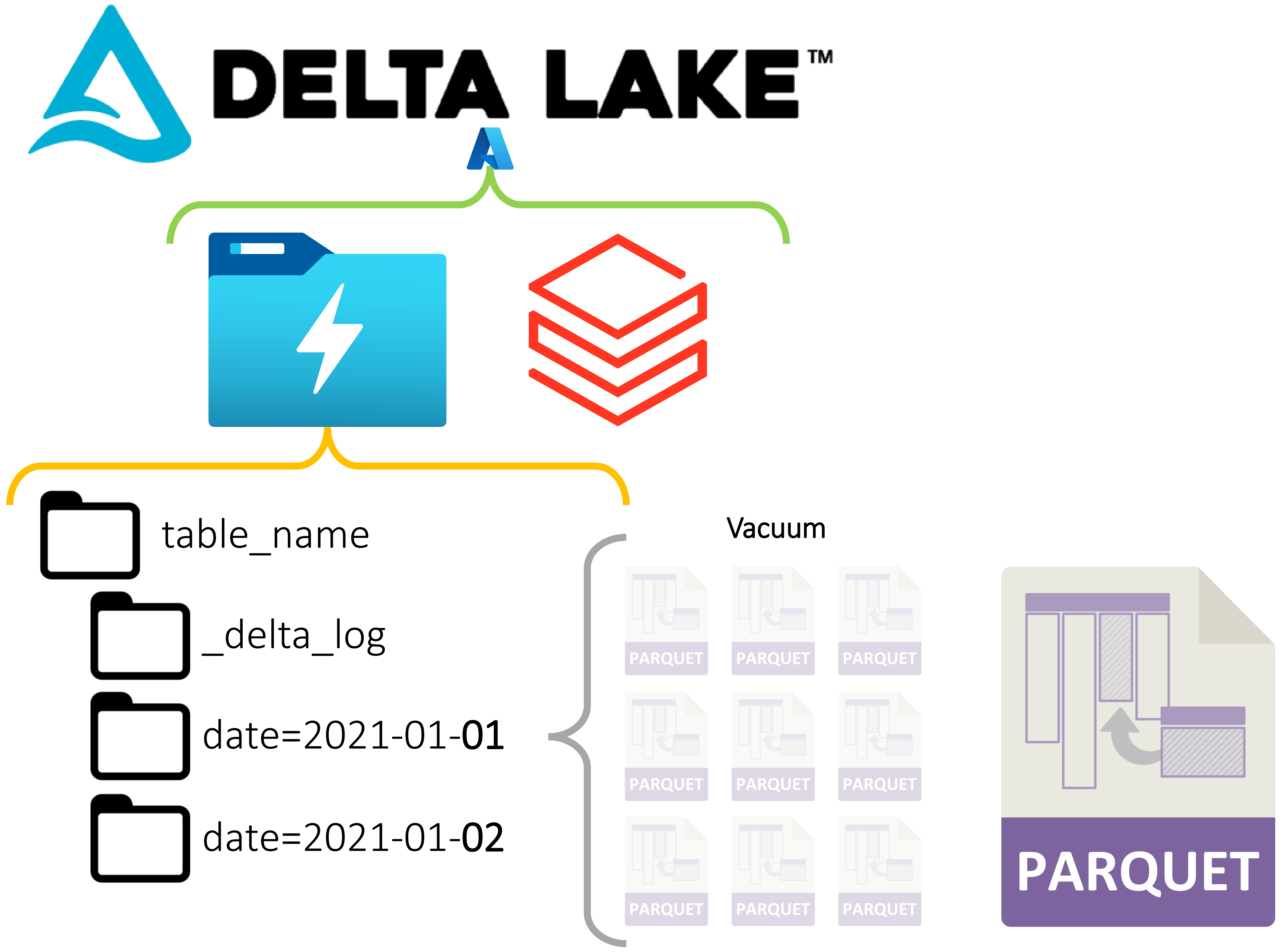


```
{"version":10,"size":53}
```

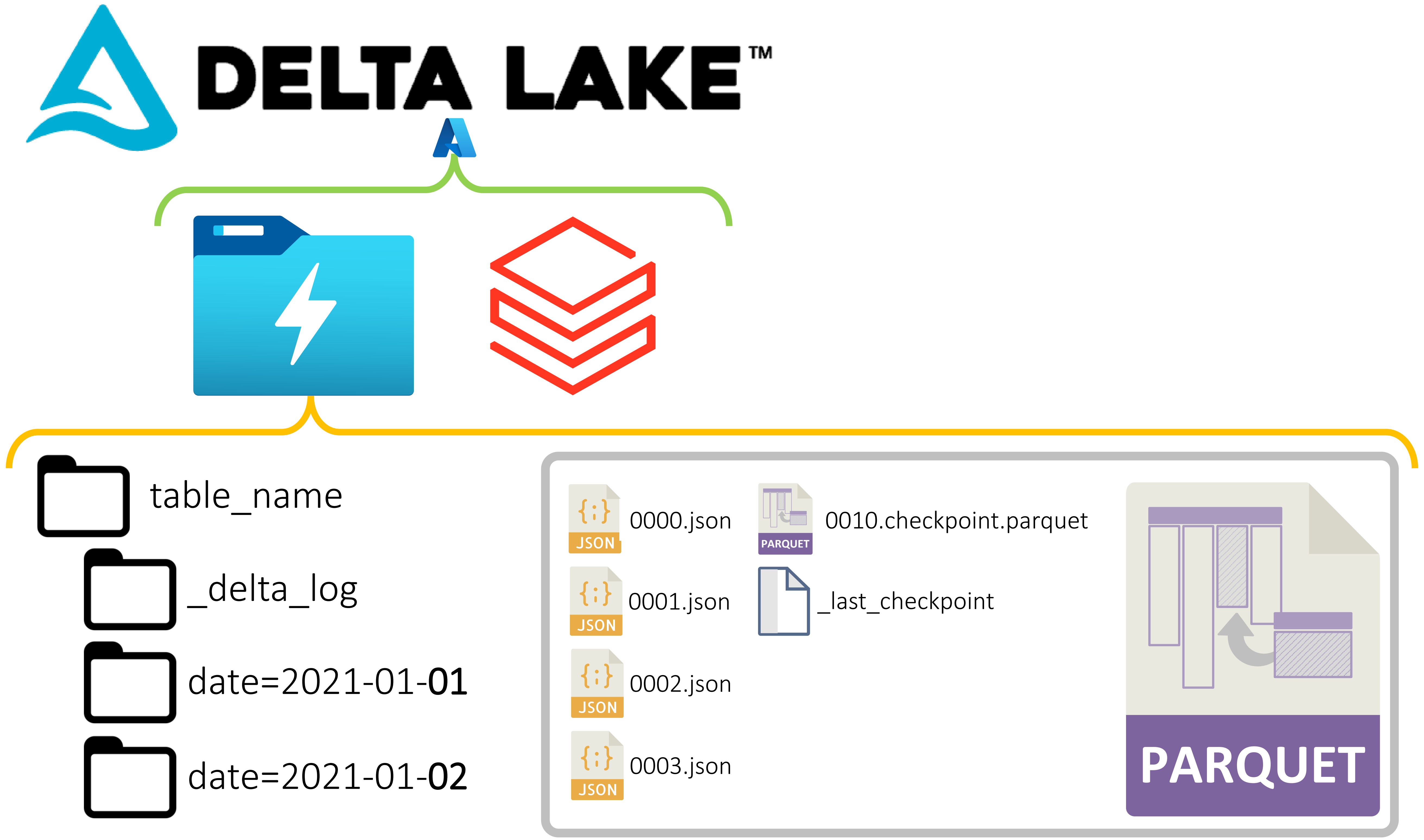
Delta Table - Transaction Log



Delta Tables – On Disk



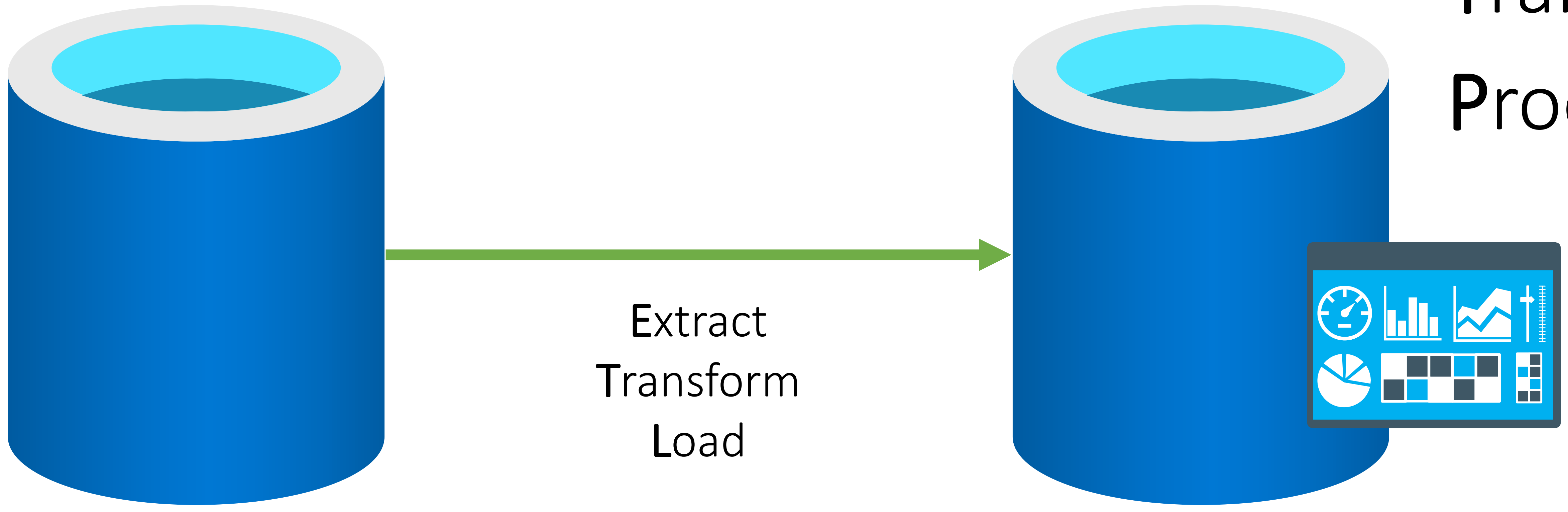
Delta Tables – On Disk



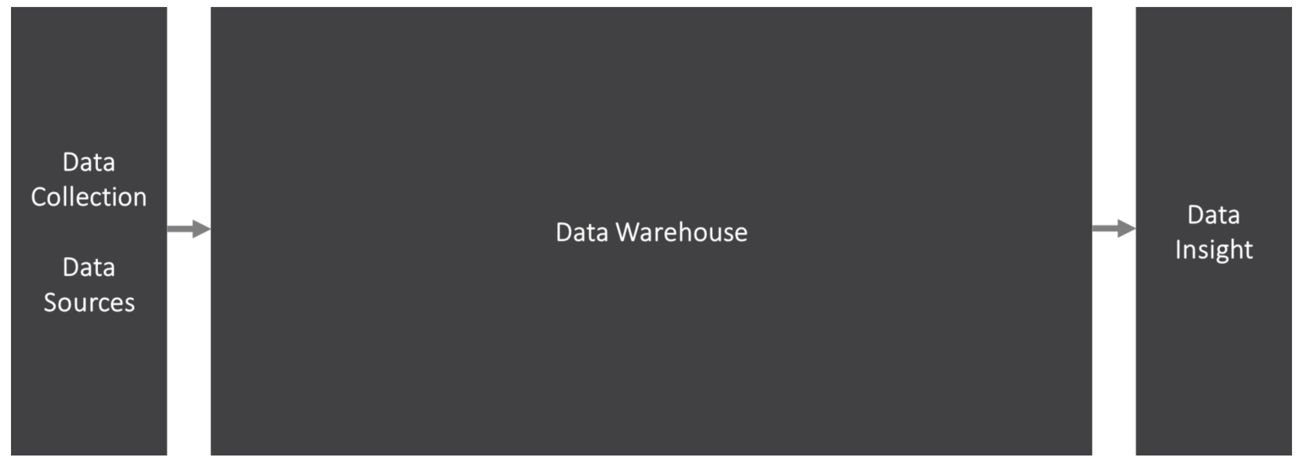
Data Warehouse

Online
Line
Transactional
Processing

Offline
Analytical
Transactional
Processing



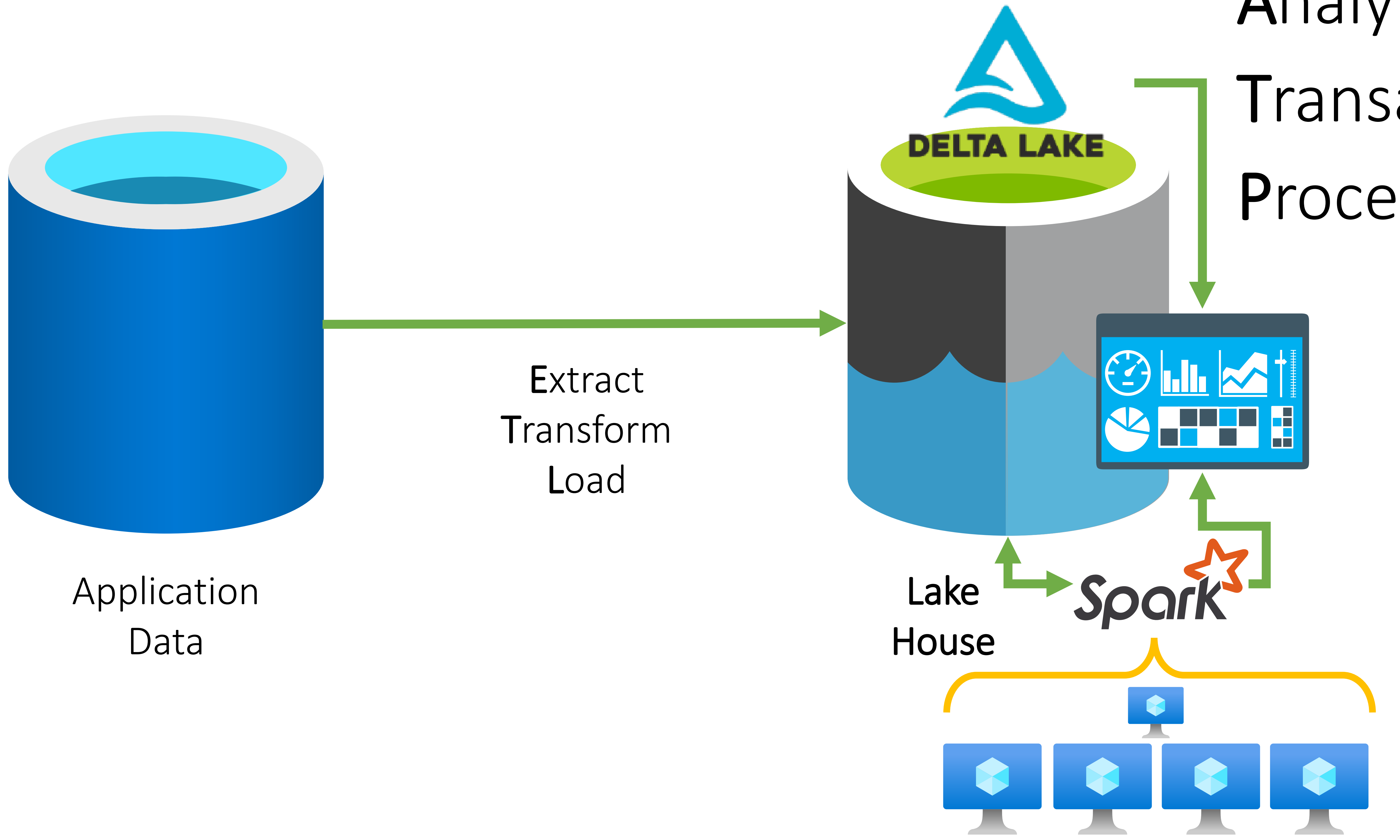
Application
Data



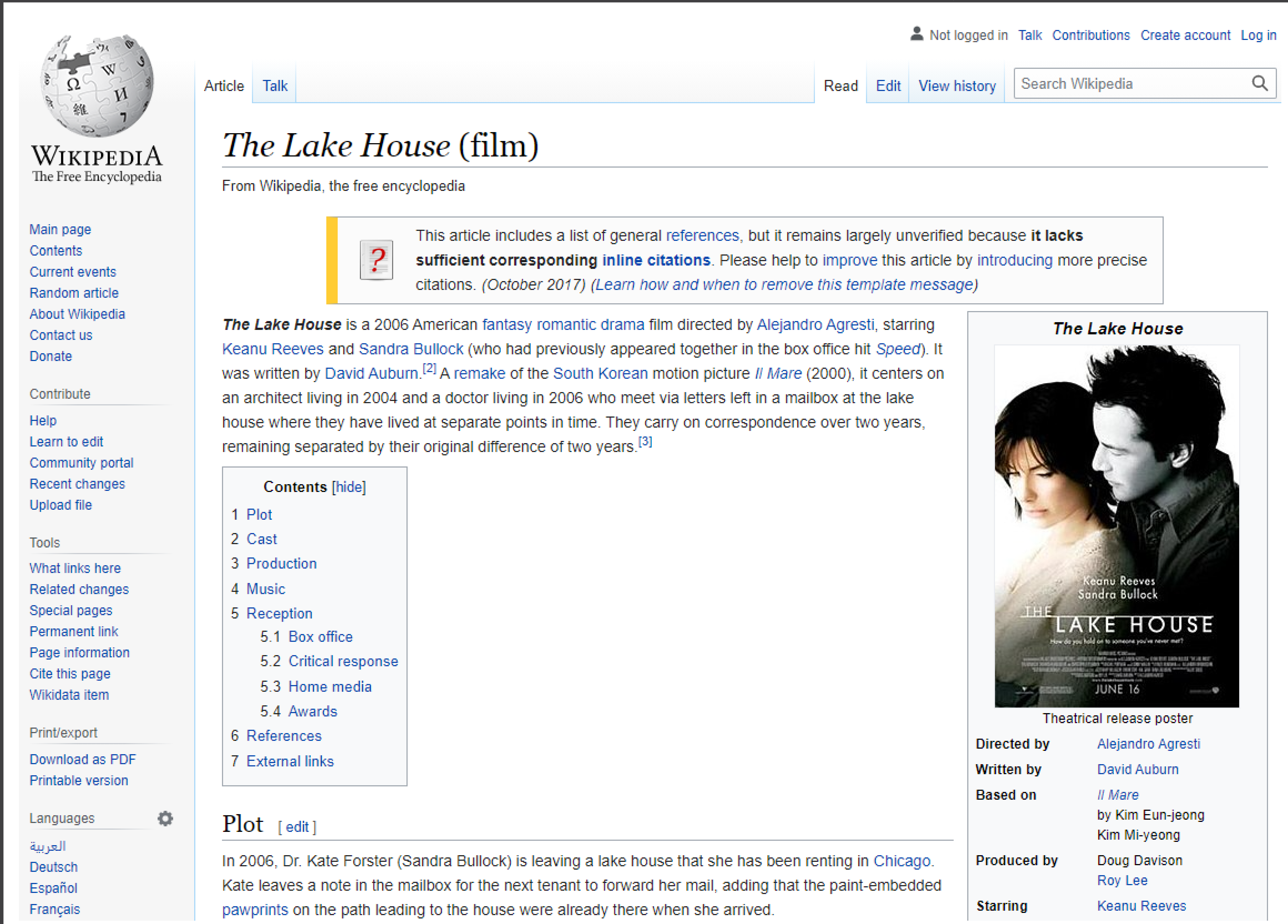
Lake House

Online
Line
Transactional
Processing

Offline
Analytical
Transactional
Processing



Lake House



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF
Printable version

Languages
العربية
Deutsch
Español
Français

The Lake House (film)

From Wikipedia, the free encyclopedia

This article includes a list of general references, but it remains largely unverified because **it lacks sufficient corresponding inline citations**. Please help to improve this article by introducing more precise citations. *(October 2017) (Learn how and when to remove this template message)*

The Lake House is a 2006 American fantasy romantic drama film directed by Alejandro Agresti, starring Keanu Reeves and Sandra Bullock (who had previously appeared together in the box office hit *Speed*). It was written by David Auburn.[2] A remake of the South Korean motion picture *Il Mare* (2000), it centers on an architect living in 2004 and a doctor living in 2006 who meet via letters left in a mailbox at the lake house where they have lived at separate points in time. They carry on correspondence over two years, remaining separated by their original difference of two years.[3]

Contents [hide]

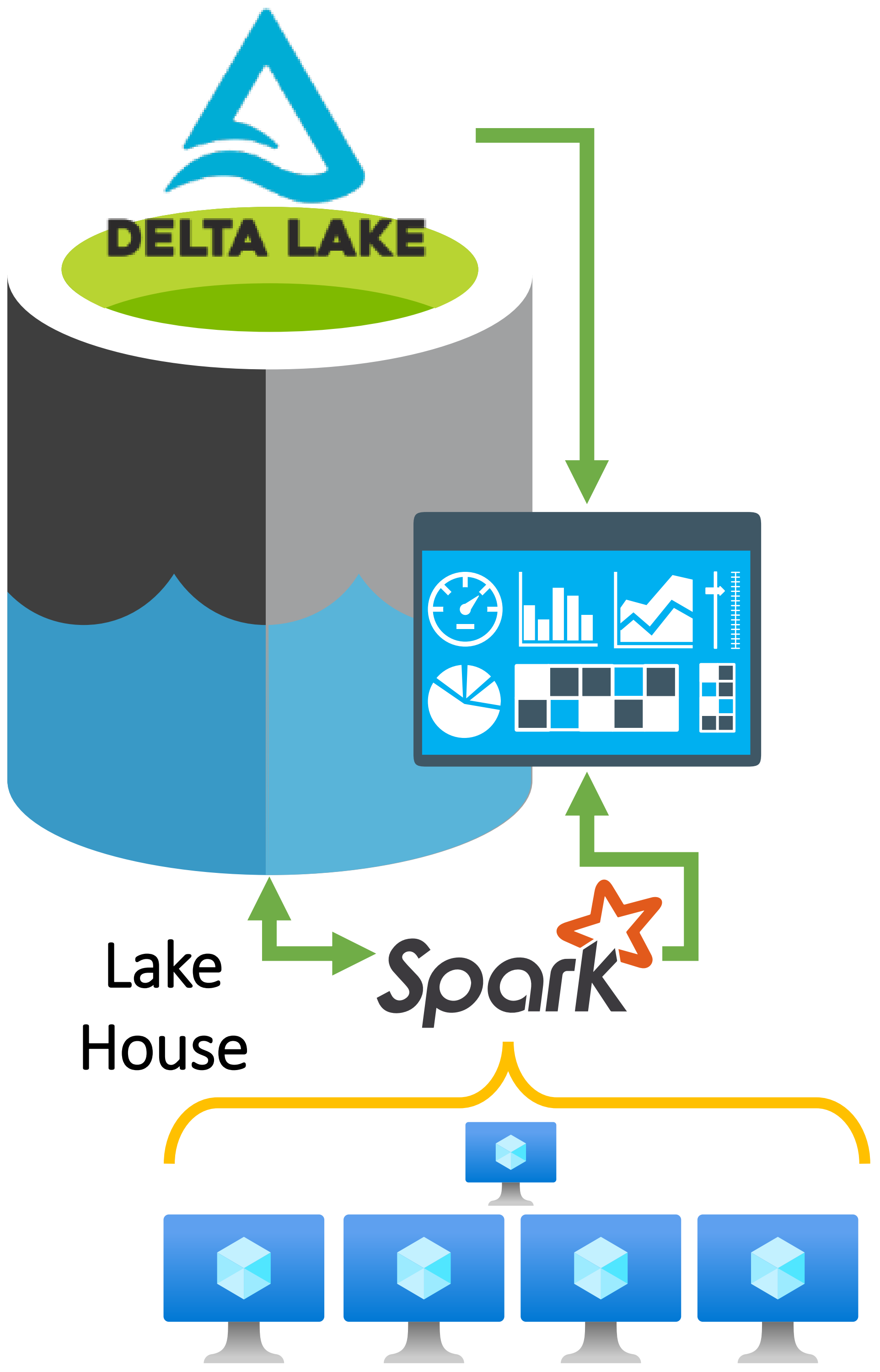
Plot [edit]

In 2006, Dr. Kate Forster (Sandra Bullock) is leaving a lake house that she has been renting in Chicago. Kate leaves a note in the mailbox for the next tenant to forward her mail, adding that the paint-embedded pawprints on the path leading to the house were already there when she arrived.

The Lake House

Theatrical release poster

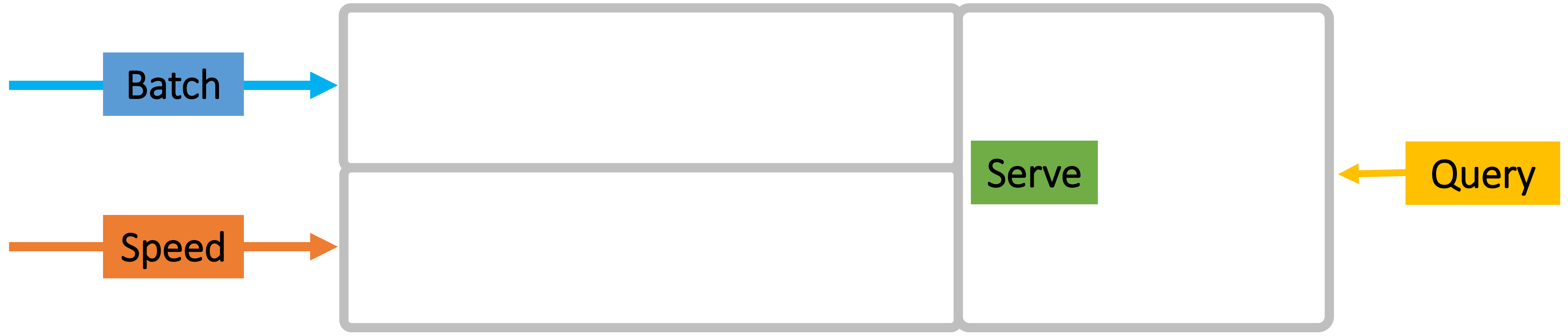
Directed by	Alejandro Agresti
Written by	David Auburn
Based on	*Il Mare* by Kim Eun-jeong Kim Mi-yeong
Produced by	Doug Davison Roy Lee
Starring	Keanu Reeves

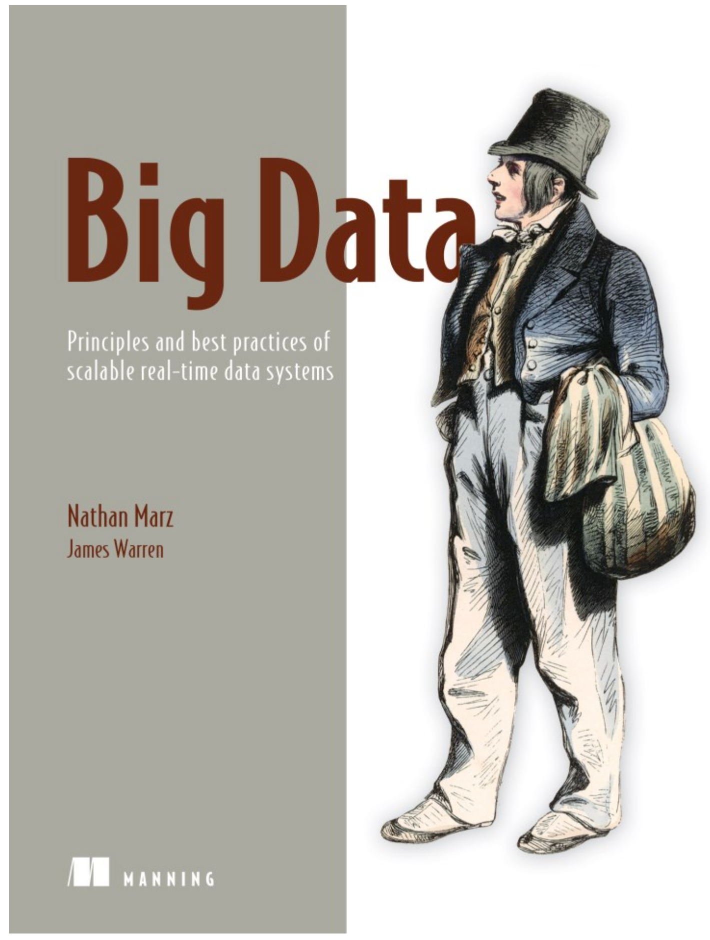


Lambda* & Kappa

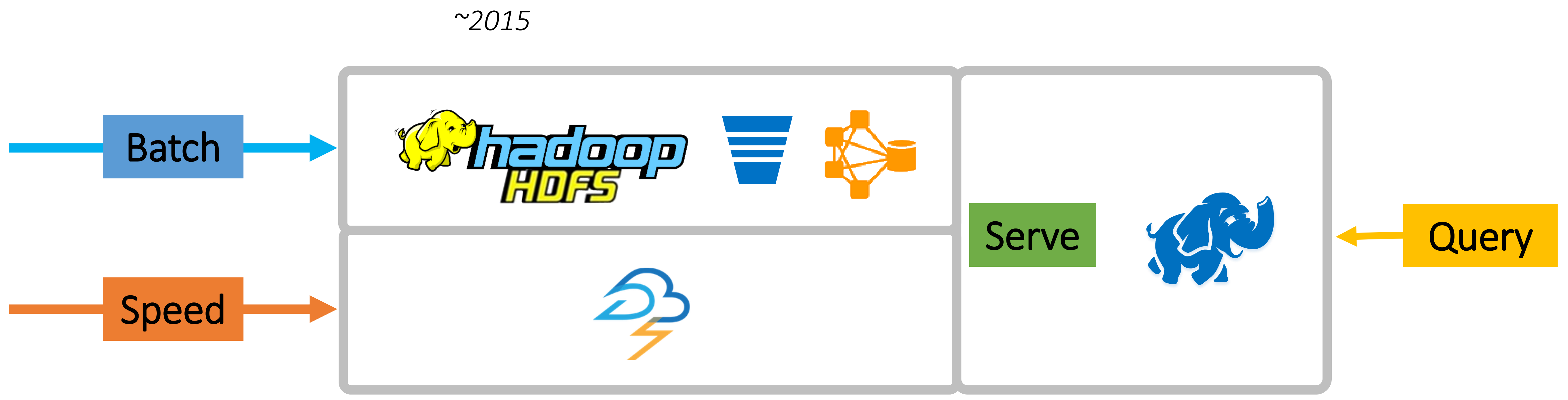
* We are not talking about the computer game Half-Life.

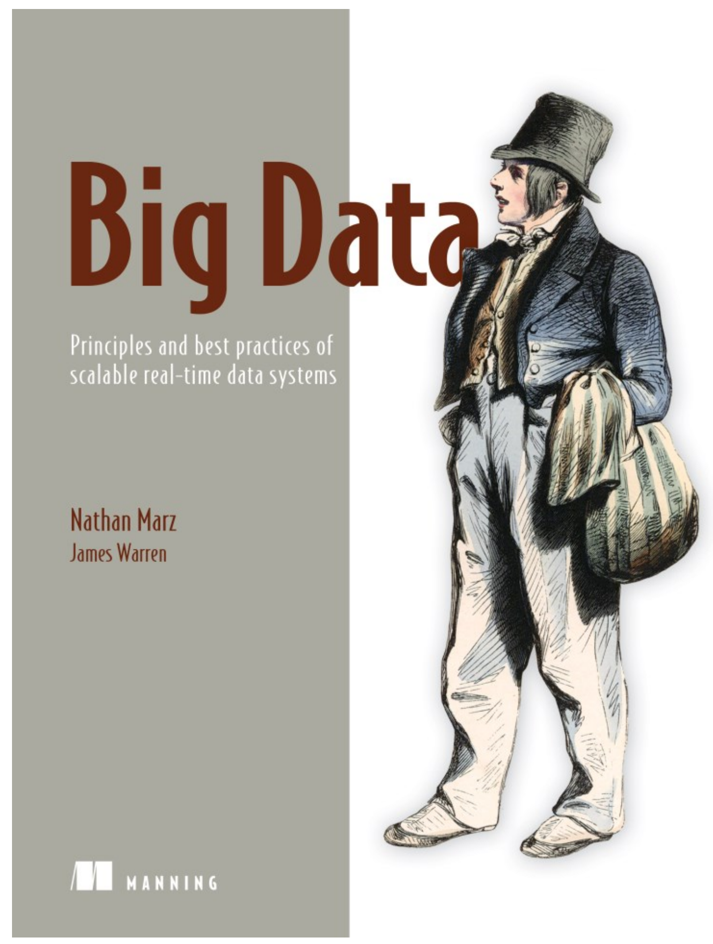
Lambda & Kappa Architectures



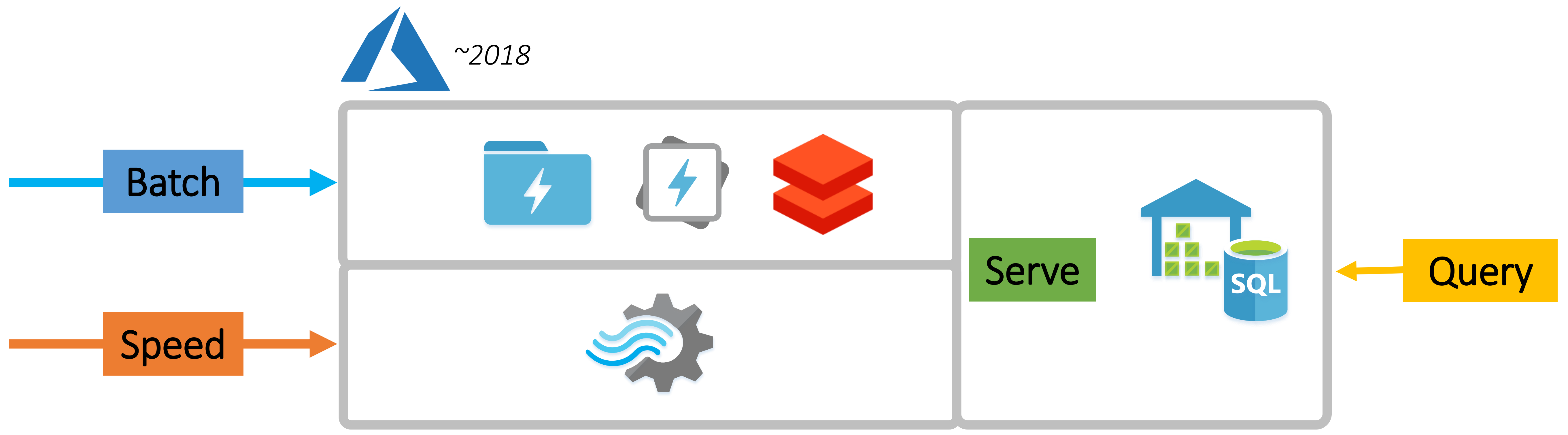


Lambda & Kappa Architectures



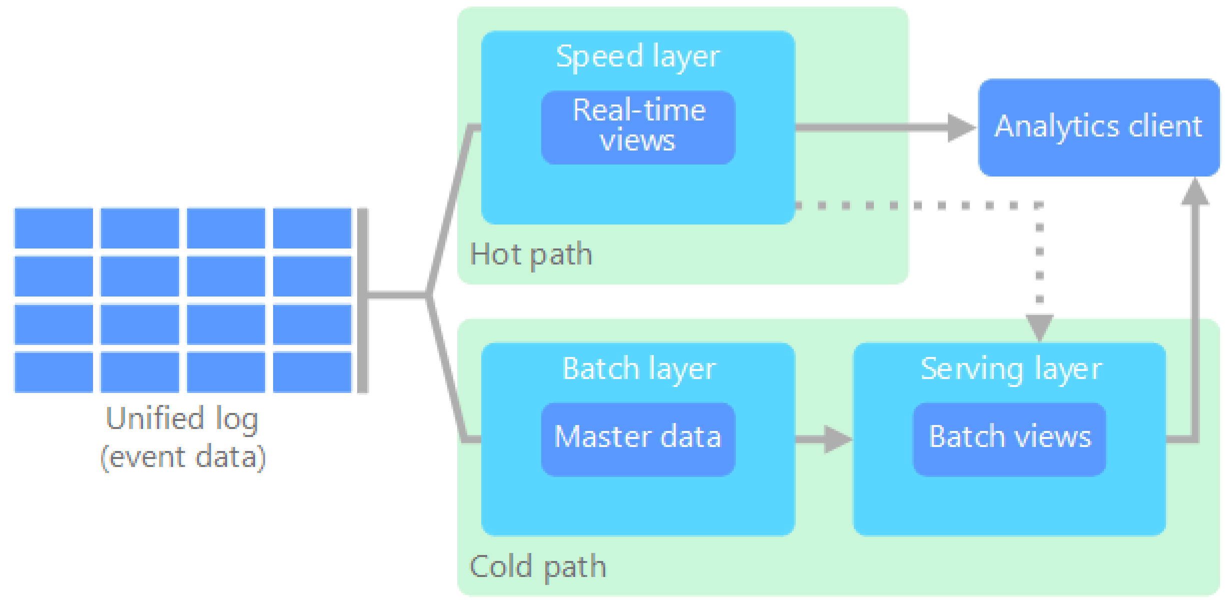

Big Data
Principles and best practices of scalable real-time data systems
Nathan Marz
James Warren
MANNING

Lambda & Kappa Architectures



Lambda & Kappa Architectures

Lambda

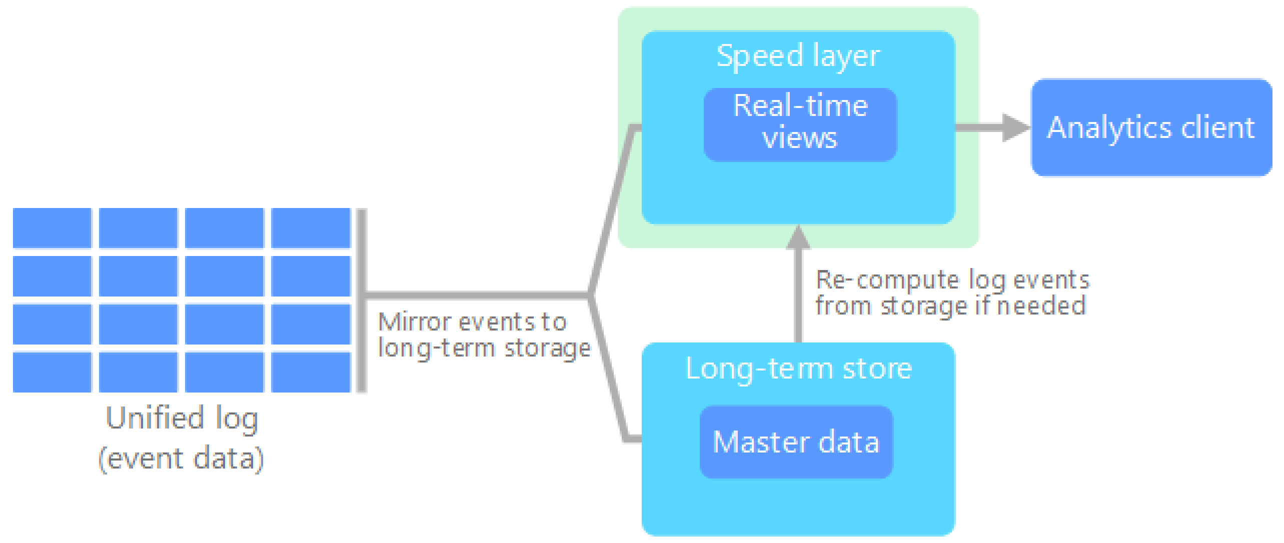


"The **lambda architecture**, first proposed by Nathan Marz, addresses this problem by creating two paths for data flow. All data coming into the system goes through these two paths:

A **batch layer** (cold path) stores all of the incoming data in its raw form and performs batch processing on the data. The result of this processing is stored as a **batch view**.

A **speed layer** (hot path) analyzes data in real time. This layer is designed for low latency, at the expense of accuracy."

Kappa

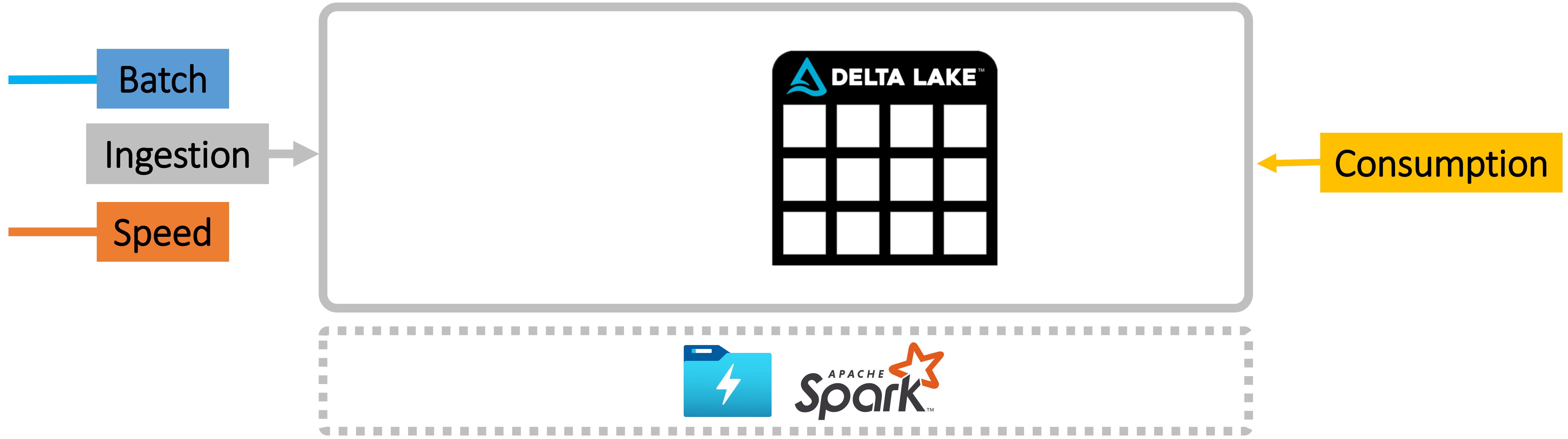


"A drawback to the lambda architecture is its complexity. Processing logic appears in two different places — the cold and hot paths — using different frameworks. This leads to duplicate computation logic and the complexity of managing the architecture for both paths.

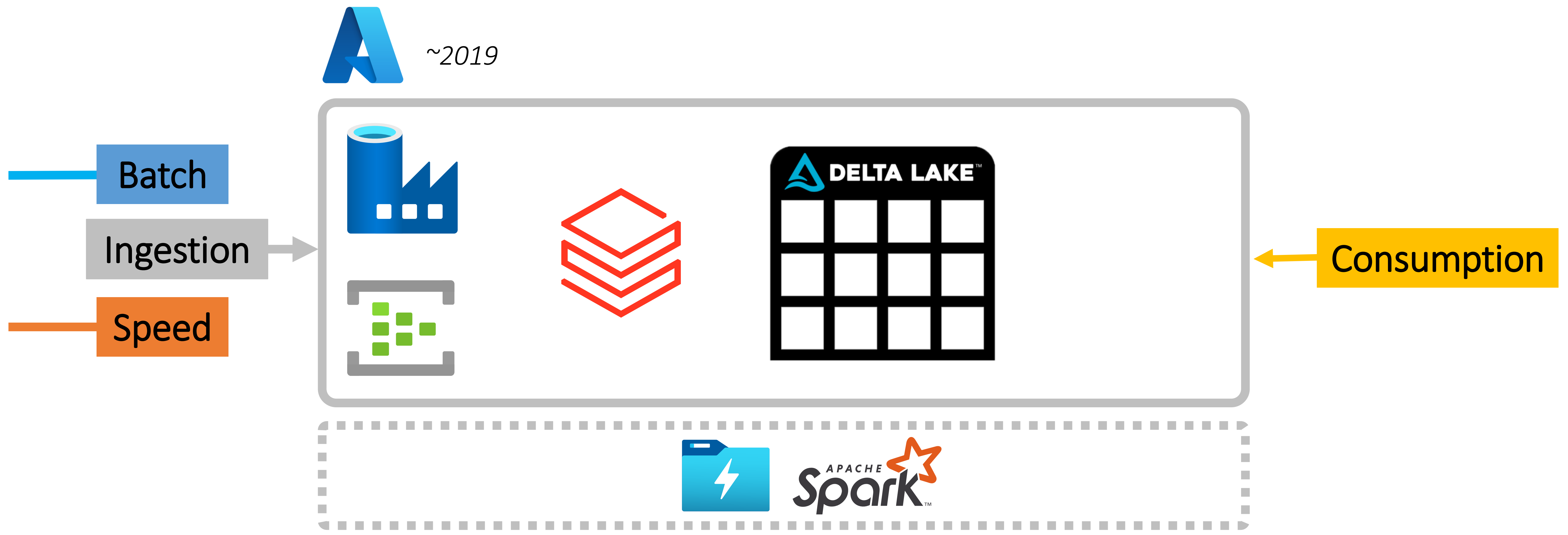
The **kappa architecture** was proposed by Jay Kreps as an alternative to the lambda architecture. It has the same basic goals as the lambda architecture, but with an important distinction: All data flows through a single path, using a stream processing system."

https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/

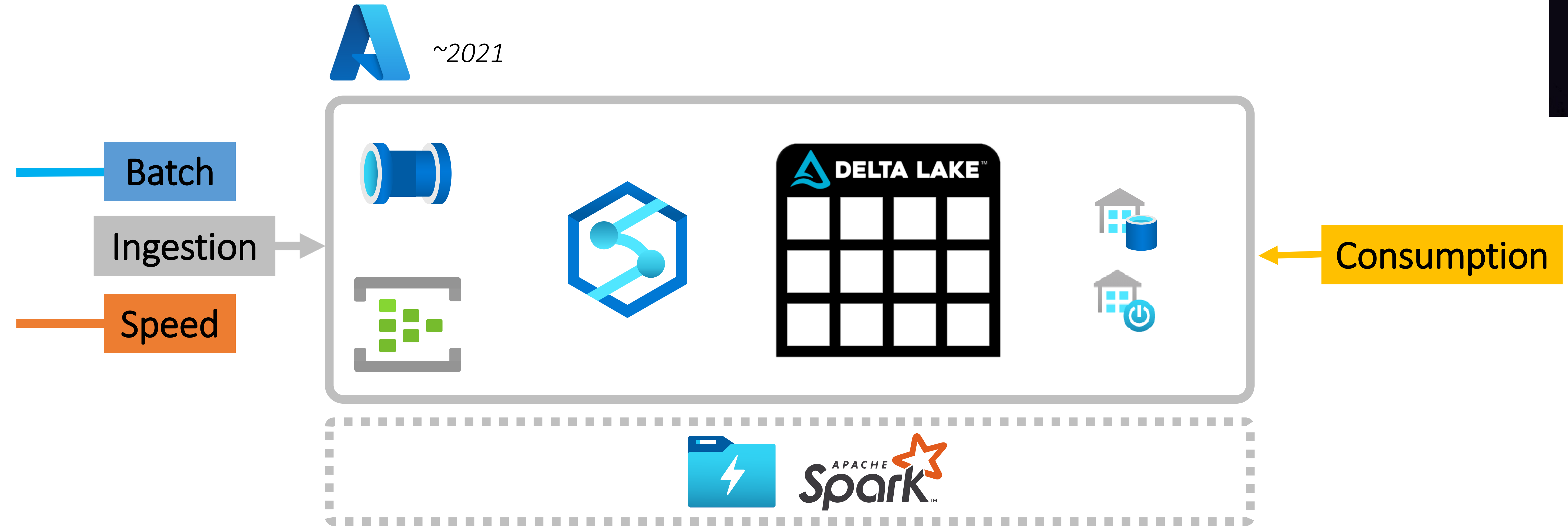
Lambda & Kappa Architectures



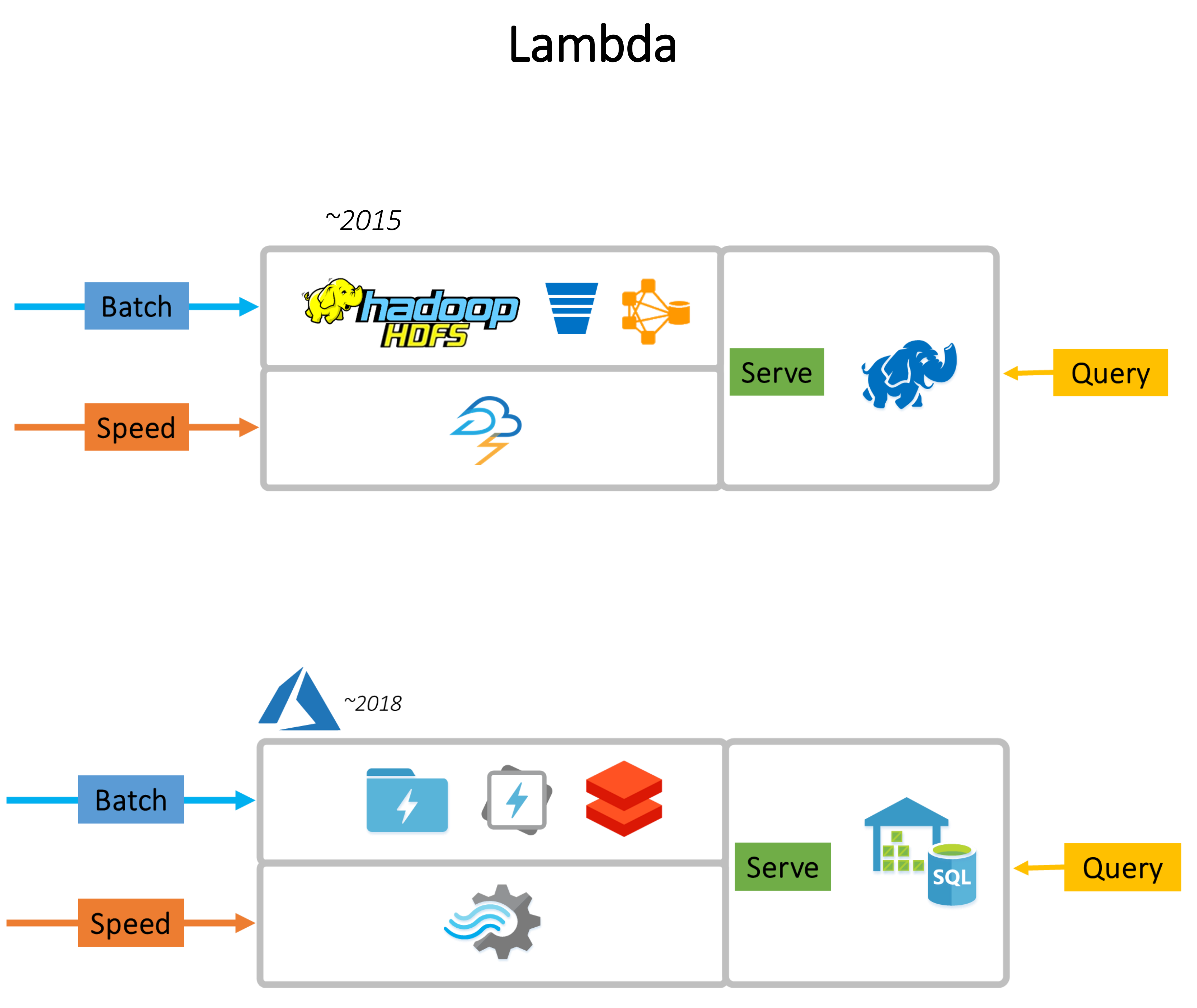
Lambda & Kappa Architectures

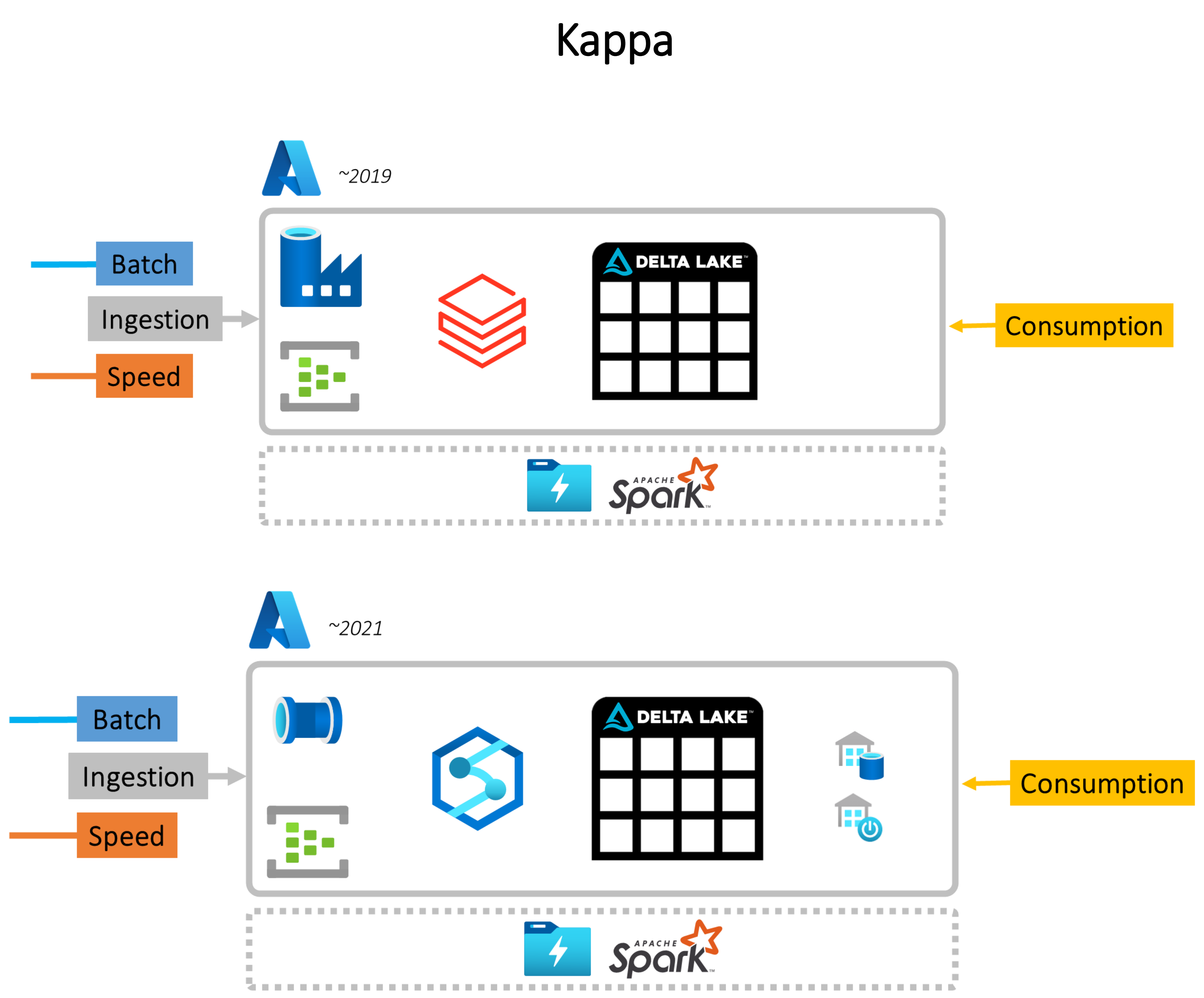


Lambda & Kappa Architectures



Delta Lake in the Context of Lambda & Kappa

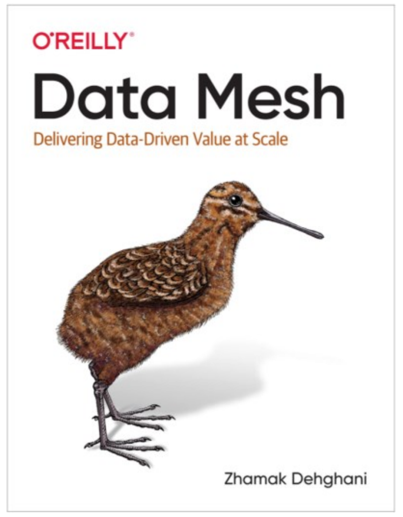


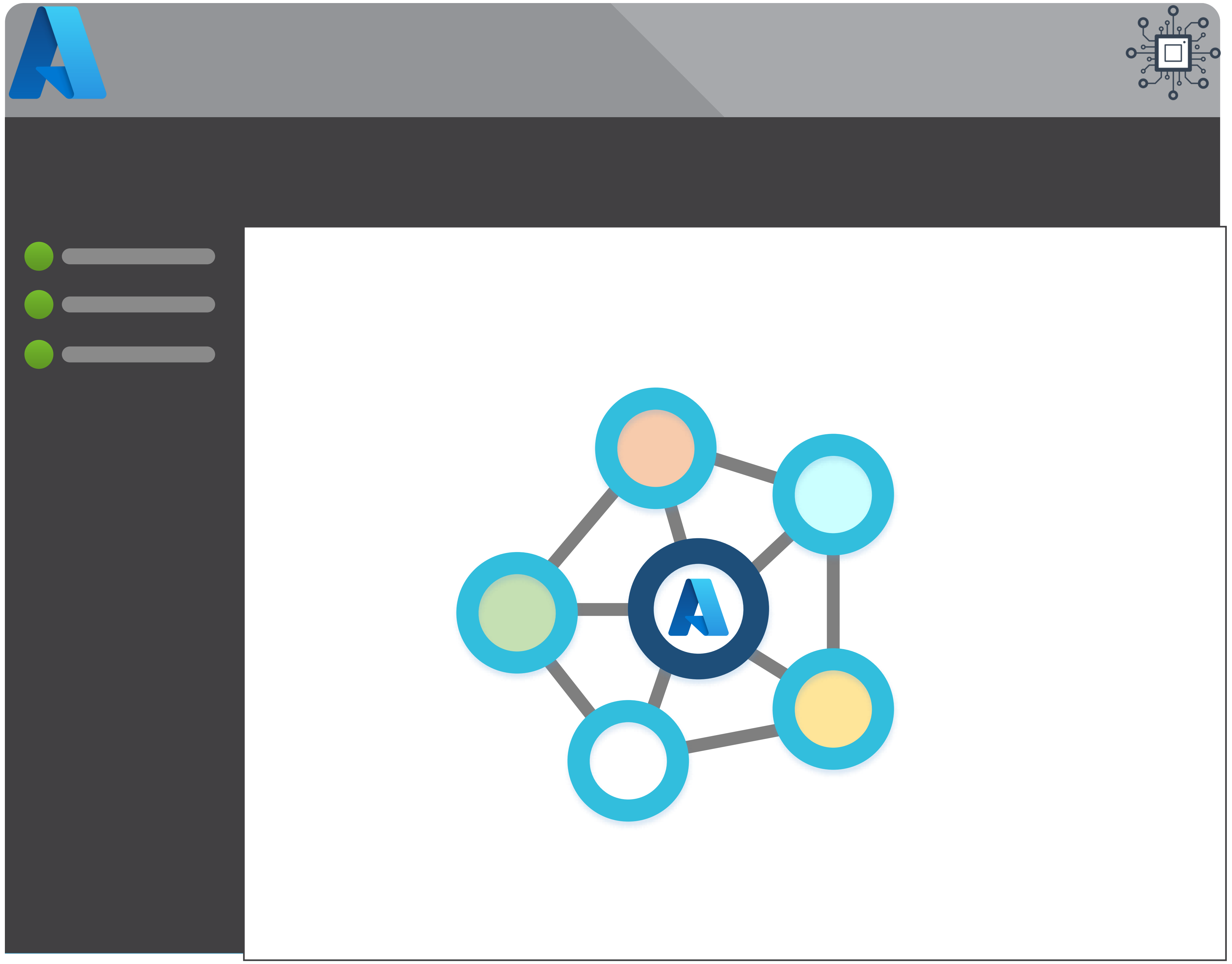


Data Mesh

- Zhamak Dehghani

@zhamakd

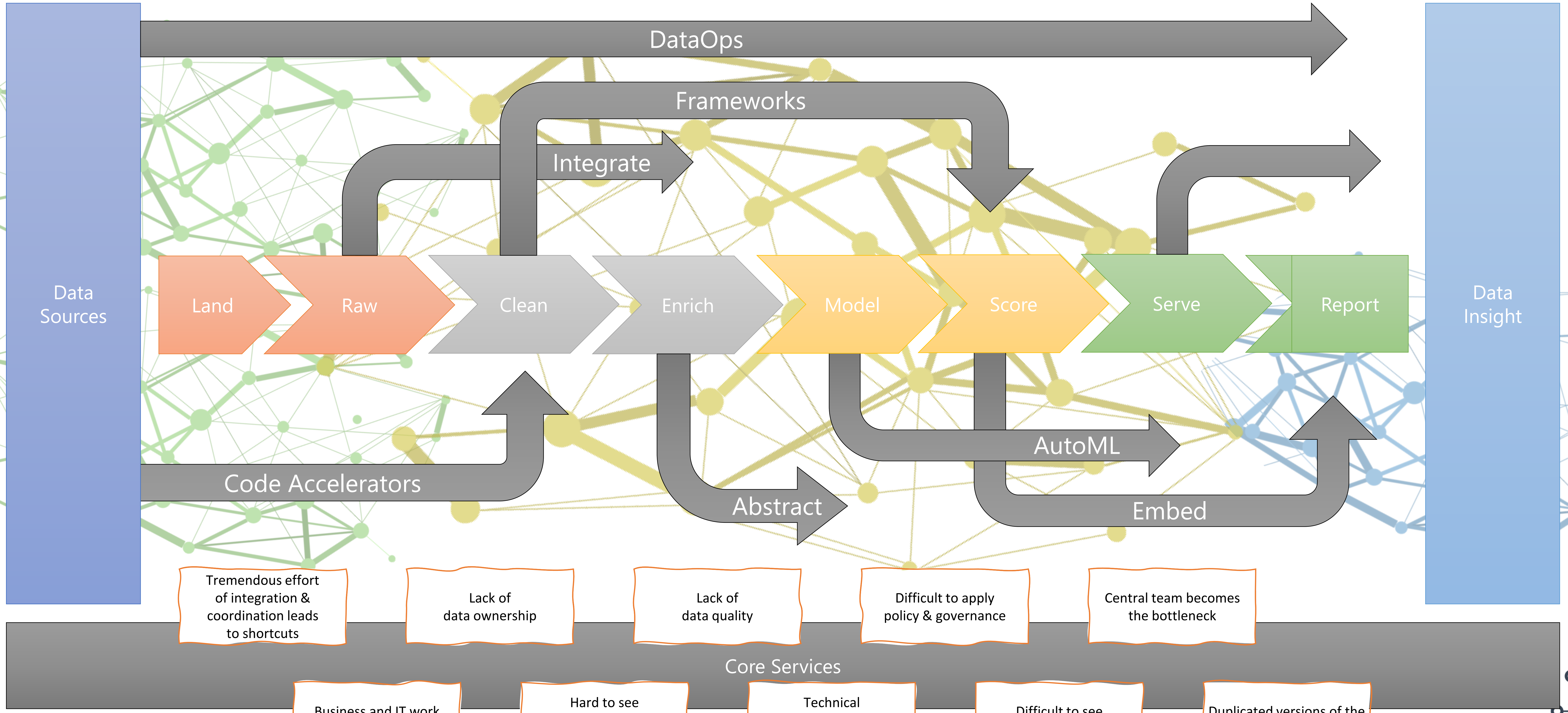




1. Domain-oriented decentralised data ownership and architecture.
2. Data as a product.
3. Self-serve data infrastructure as a platform.
4. Federated computational governance.

Data Mesh – *Why should we build it?*

Using a **traditional centralised approach**, enhanced with cloud scale technologies to create a modern data analytics platform.



Tremendous effort of integration & coordination leads to shortcuts

Lack of data ownership

Lack of data quality

Difficult to apply policy & governance

Central team becomes the bottleneck

Business and IT work in silos

Hard to see dependencies (technical debt)

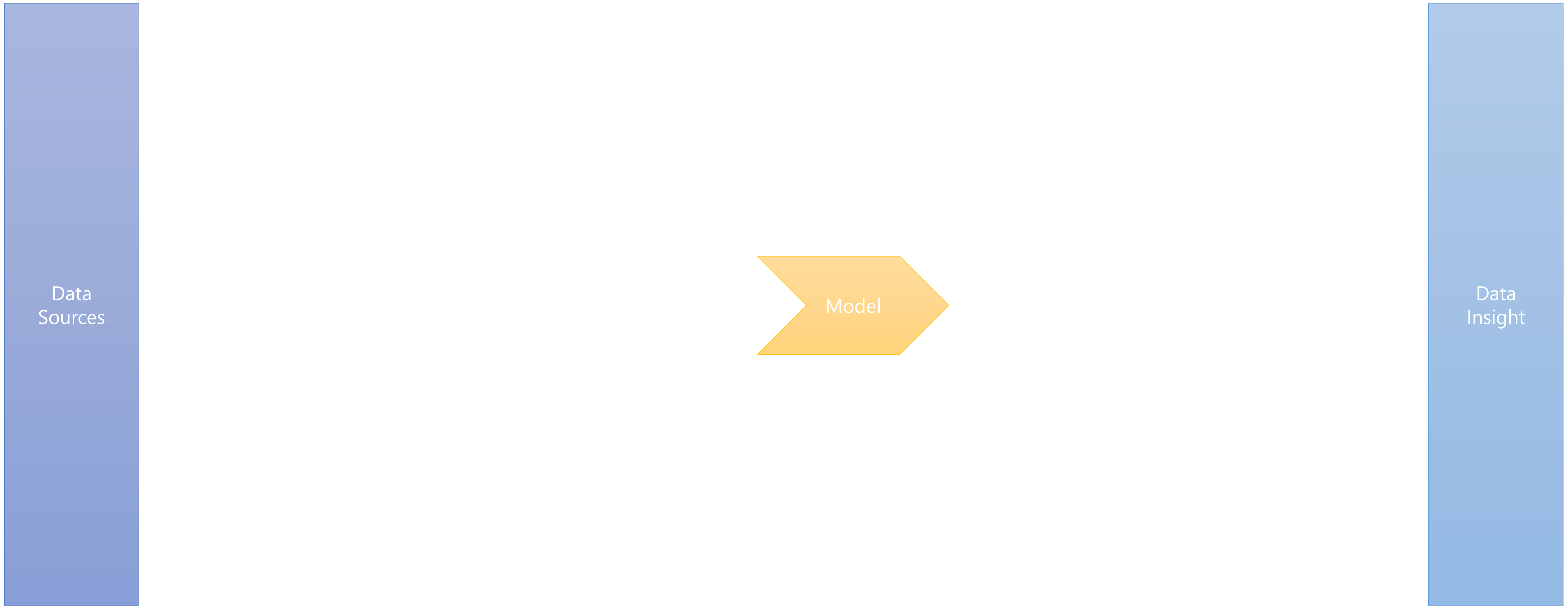
Technical ownership, rather than data ownership

Difficult to see interdependencies

Duplicated versions of the truth

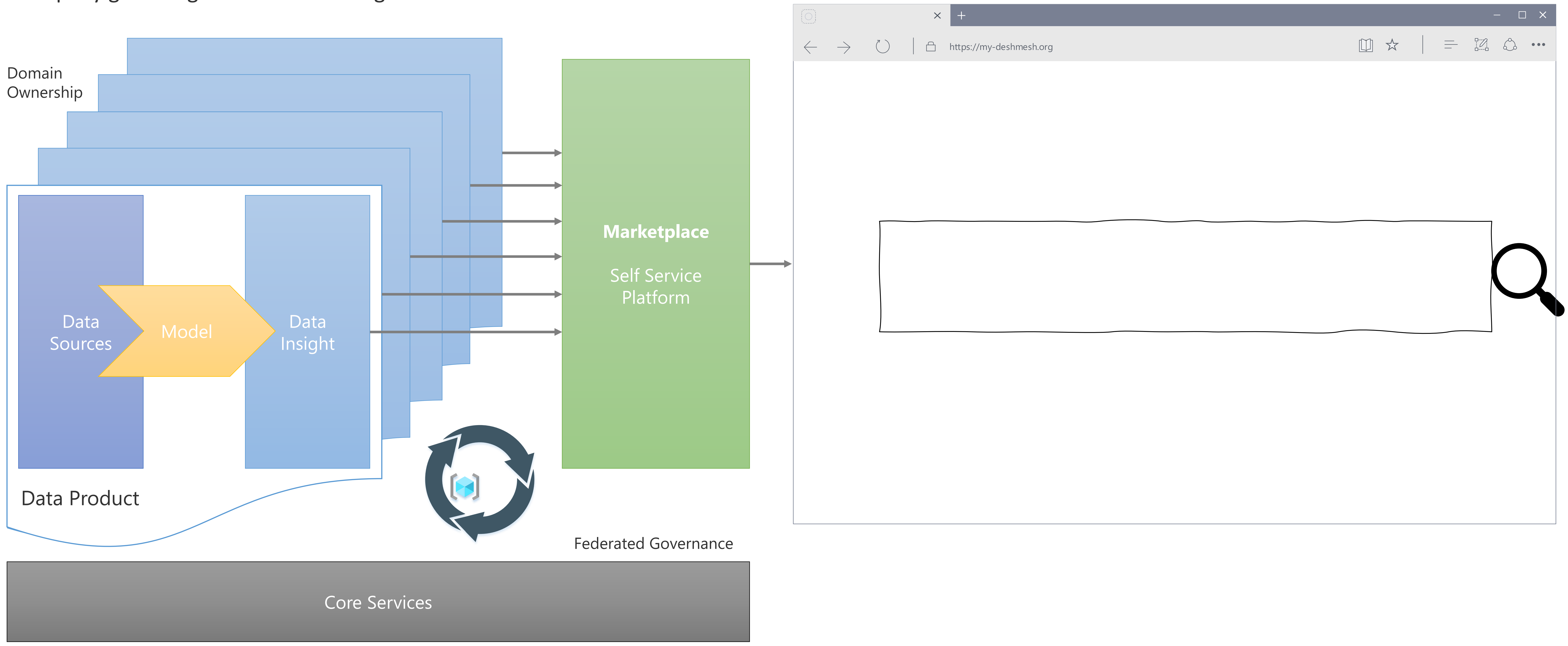
Data Mesh – *Why should we build it?*

Using a **traditional centralised** approach, enhanced with cloud scale technologies to create a modern data analytics platform.



Data Mesh – *Why should we build it?*

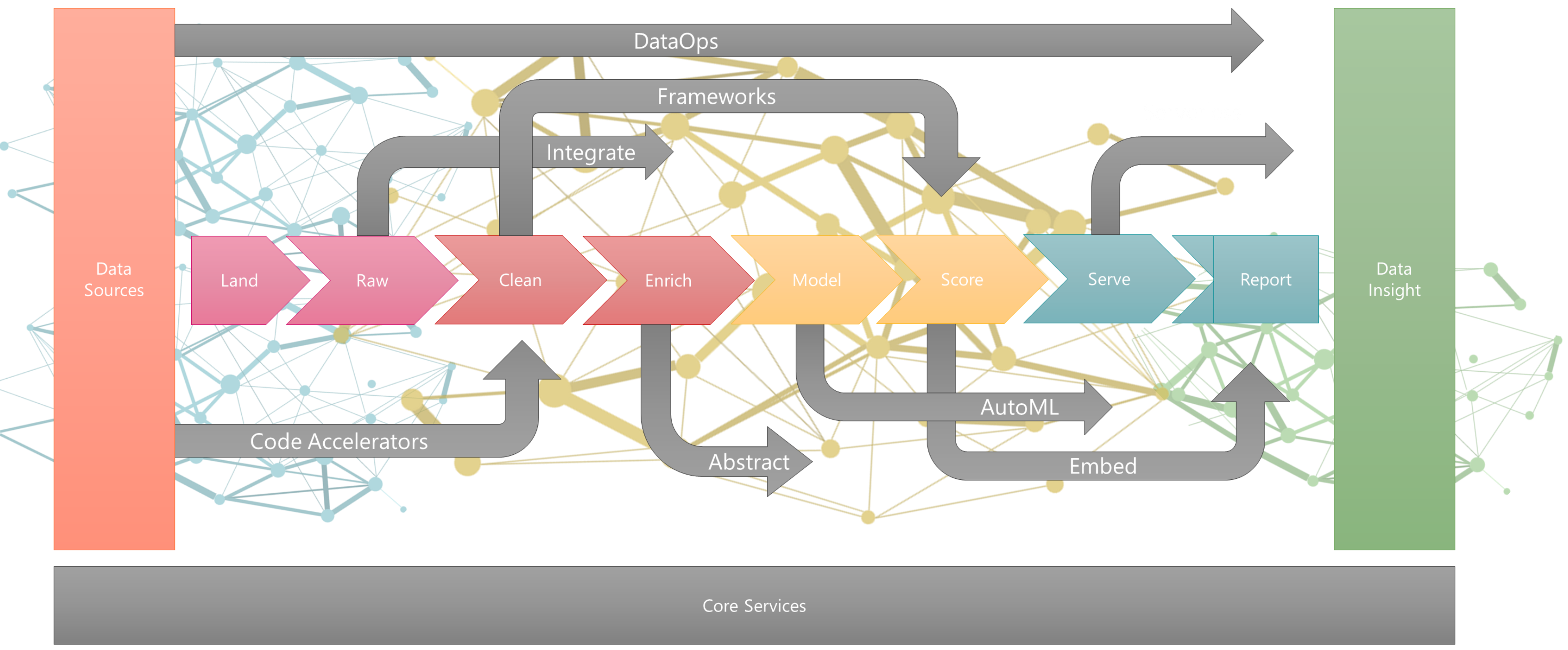
Using a **de-centralised** approach to cloud scale analytics, empowering users to rapidly gain insights to make strategic business decisions.

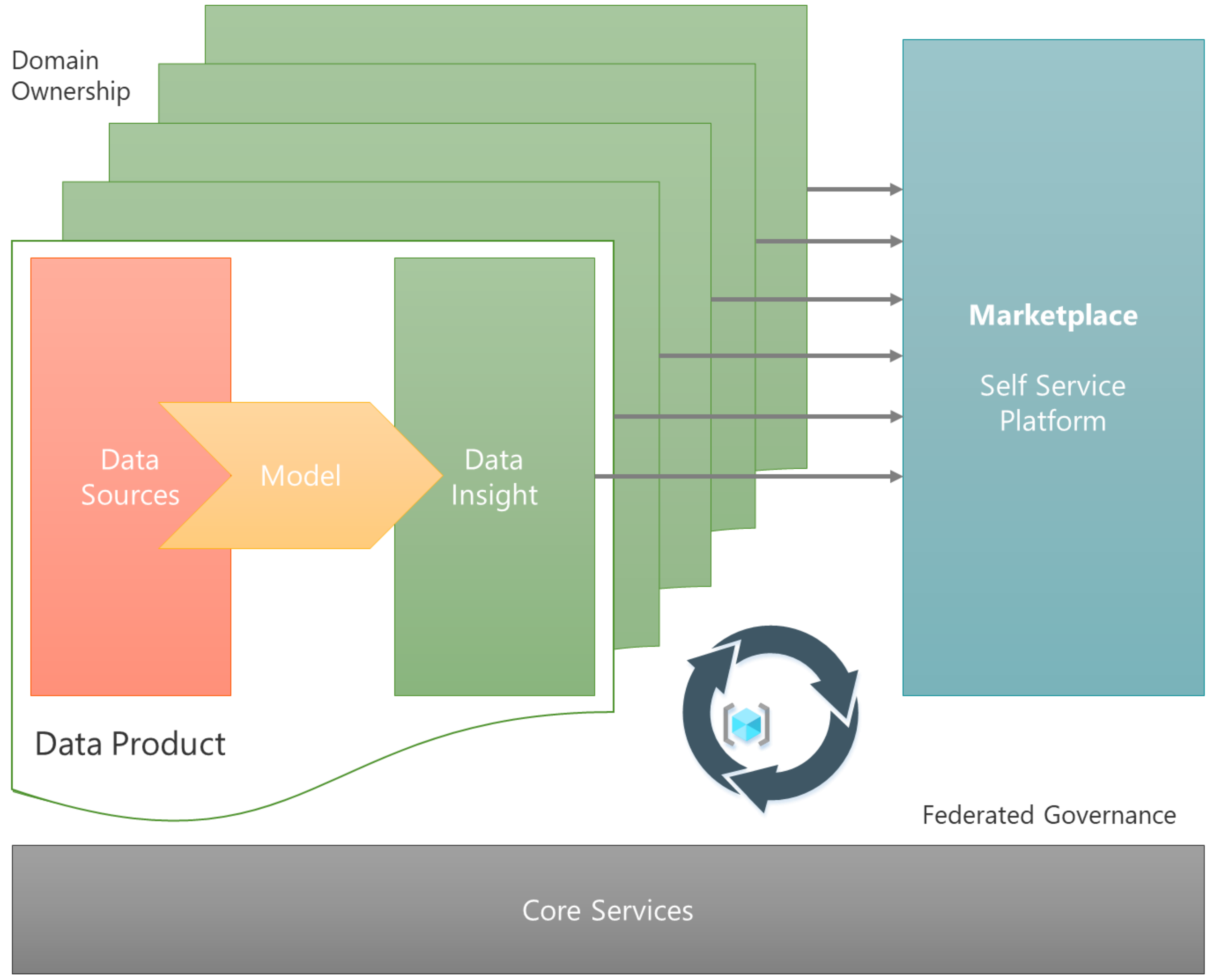


Time to Insight

Using a **traditional centralised approach**, enhanced with cloud scale technologies to create a modern data analytics platform.

Using a **de-centralised** approach to cloud scale analytics, empowering users to rapidly gain insights to make strategic business decisions.



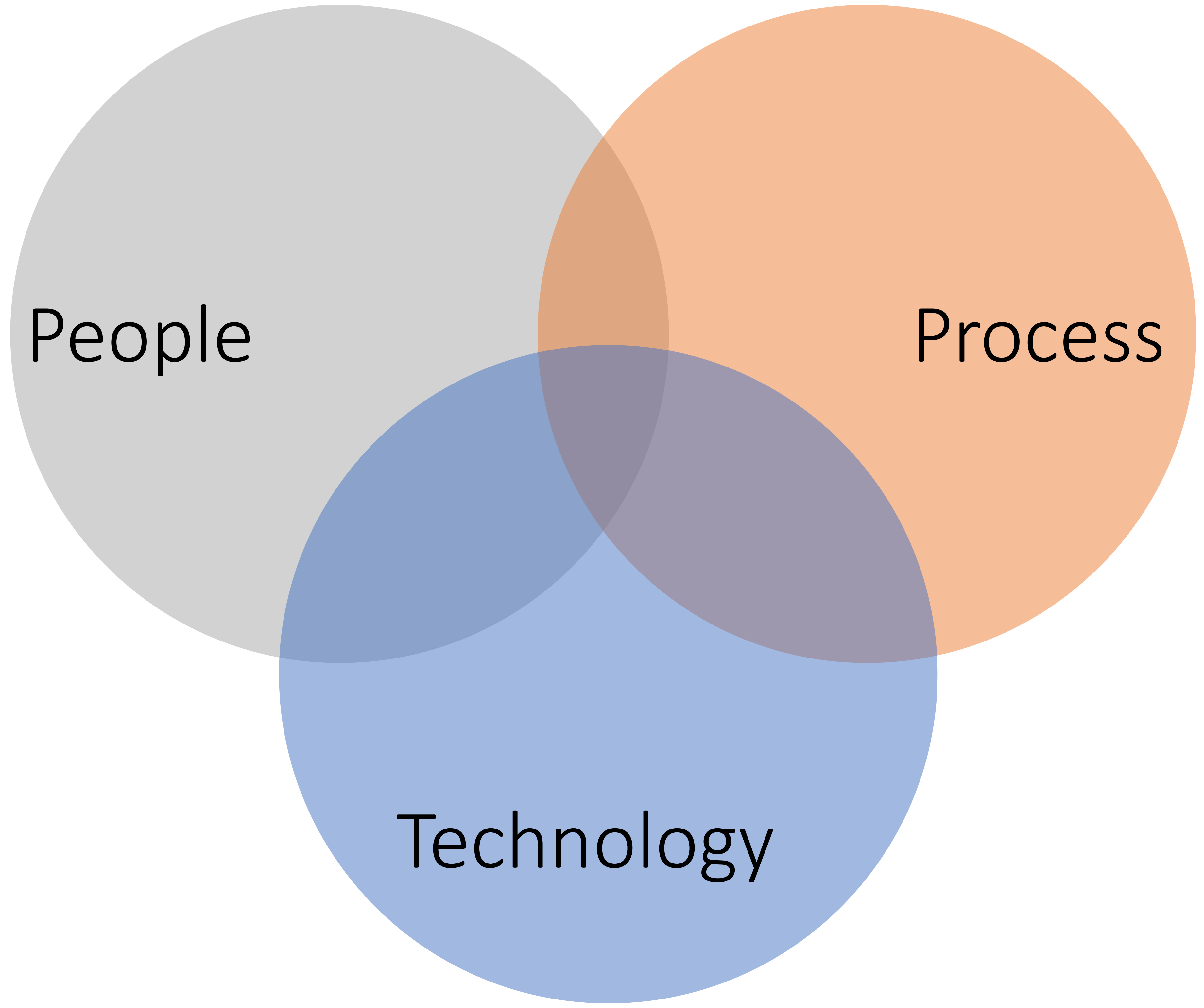


... Weeks/Months

... Hours/Days

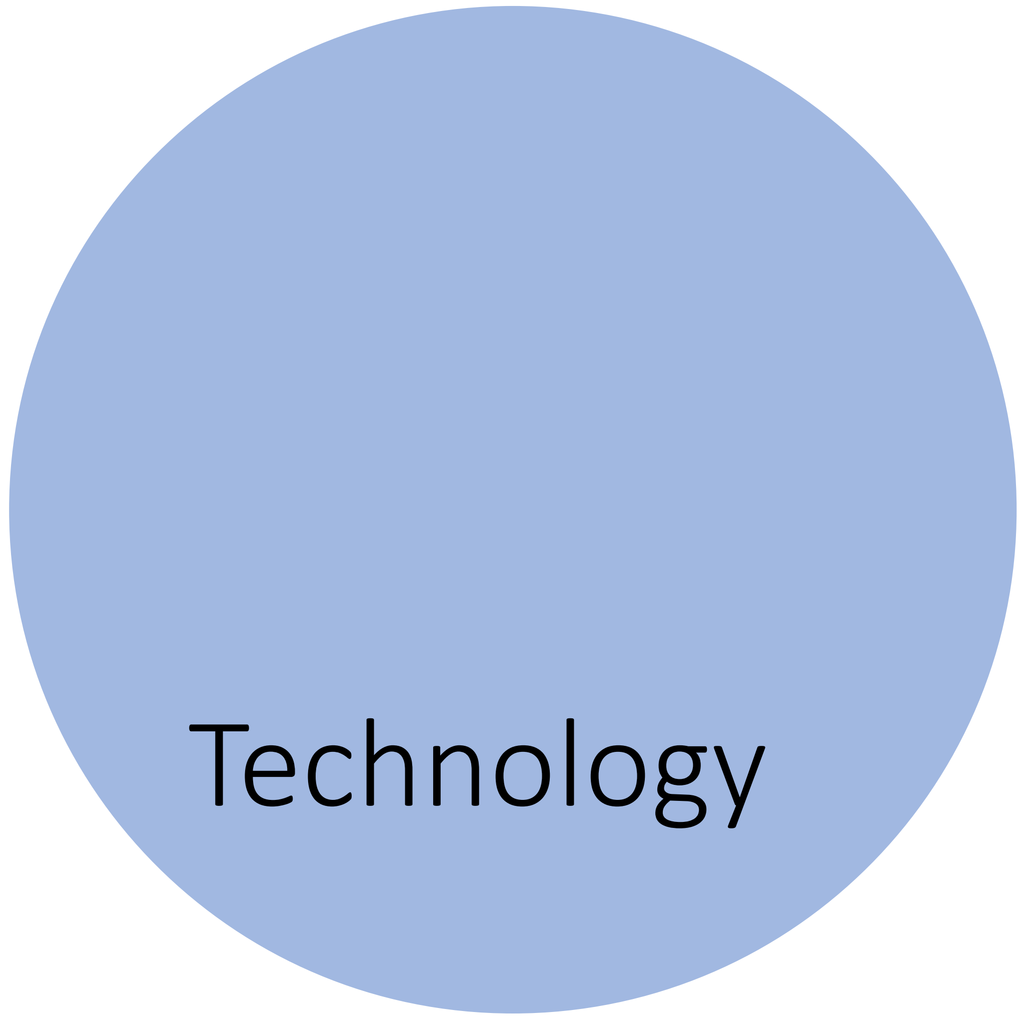
Introducing the Data Mesh

- Culture
- Personas
- Structure
- Owners
- Teams
- Skills
- Operations
- Guilds

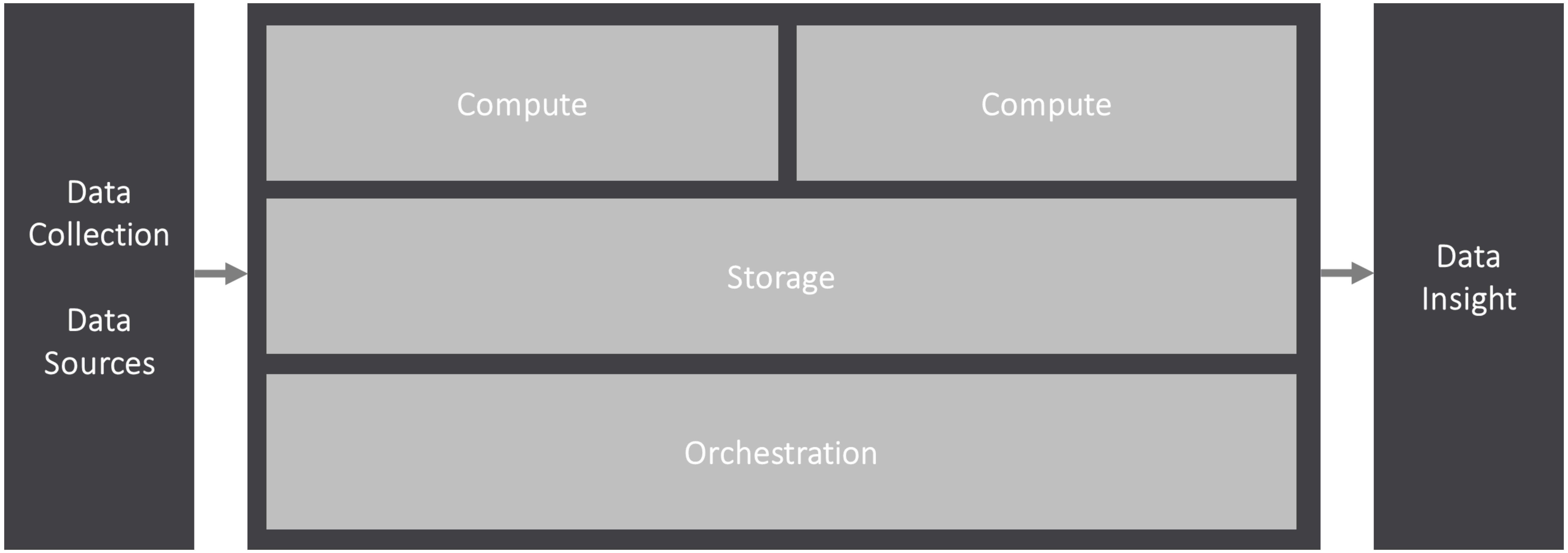


- Works of Working
- Agility
- Governance
- Workloads
- Requirements Gathering
- Maturity
- Frameworks
- Time Zones

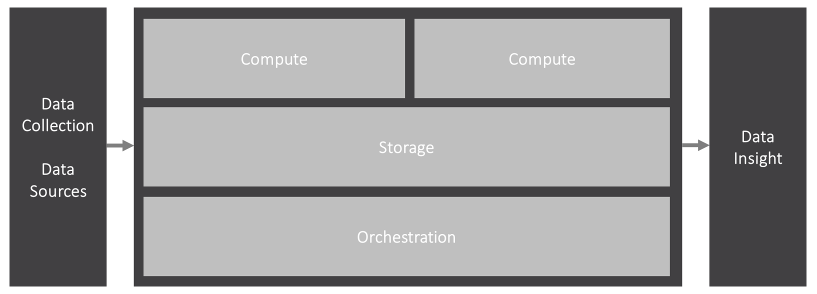
Introducing the Data Mesh

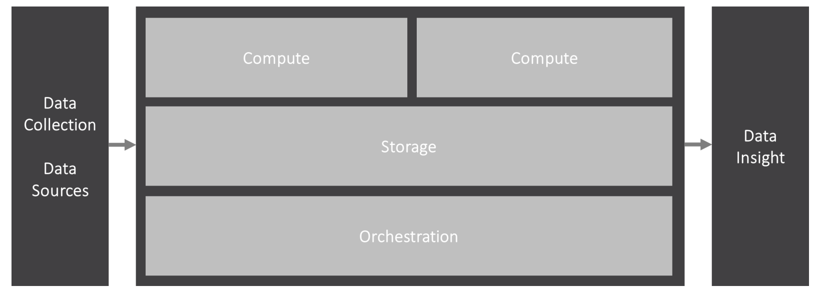


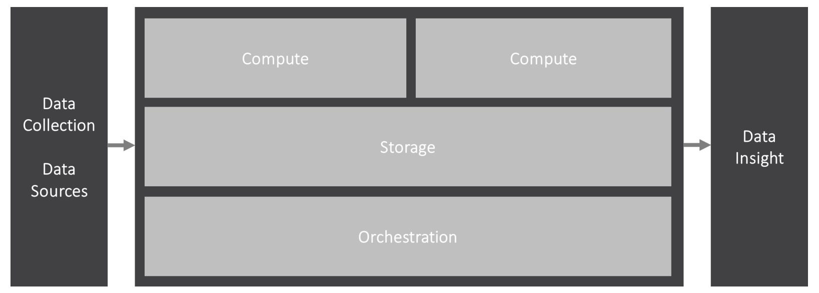
Data Products

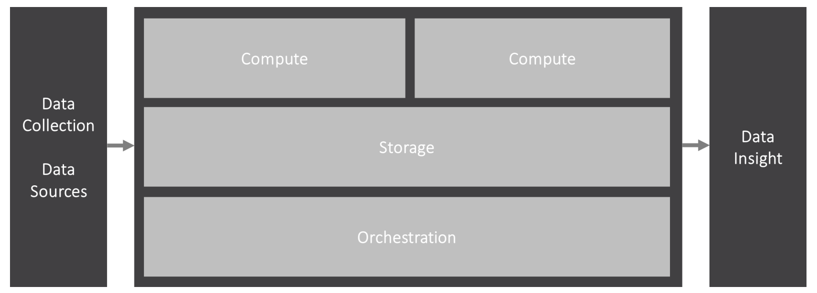


Data Products in Azure

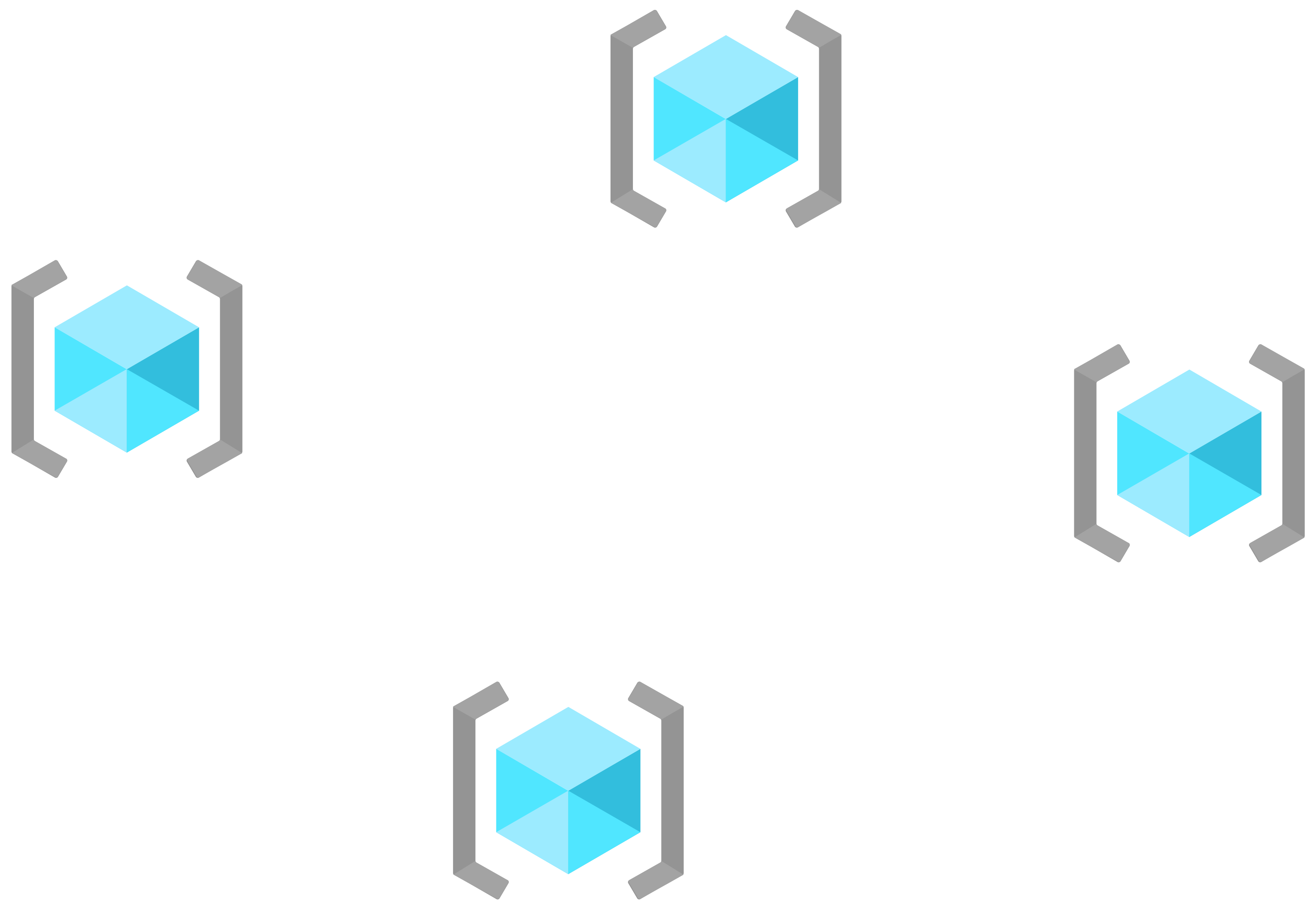




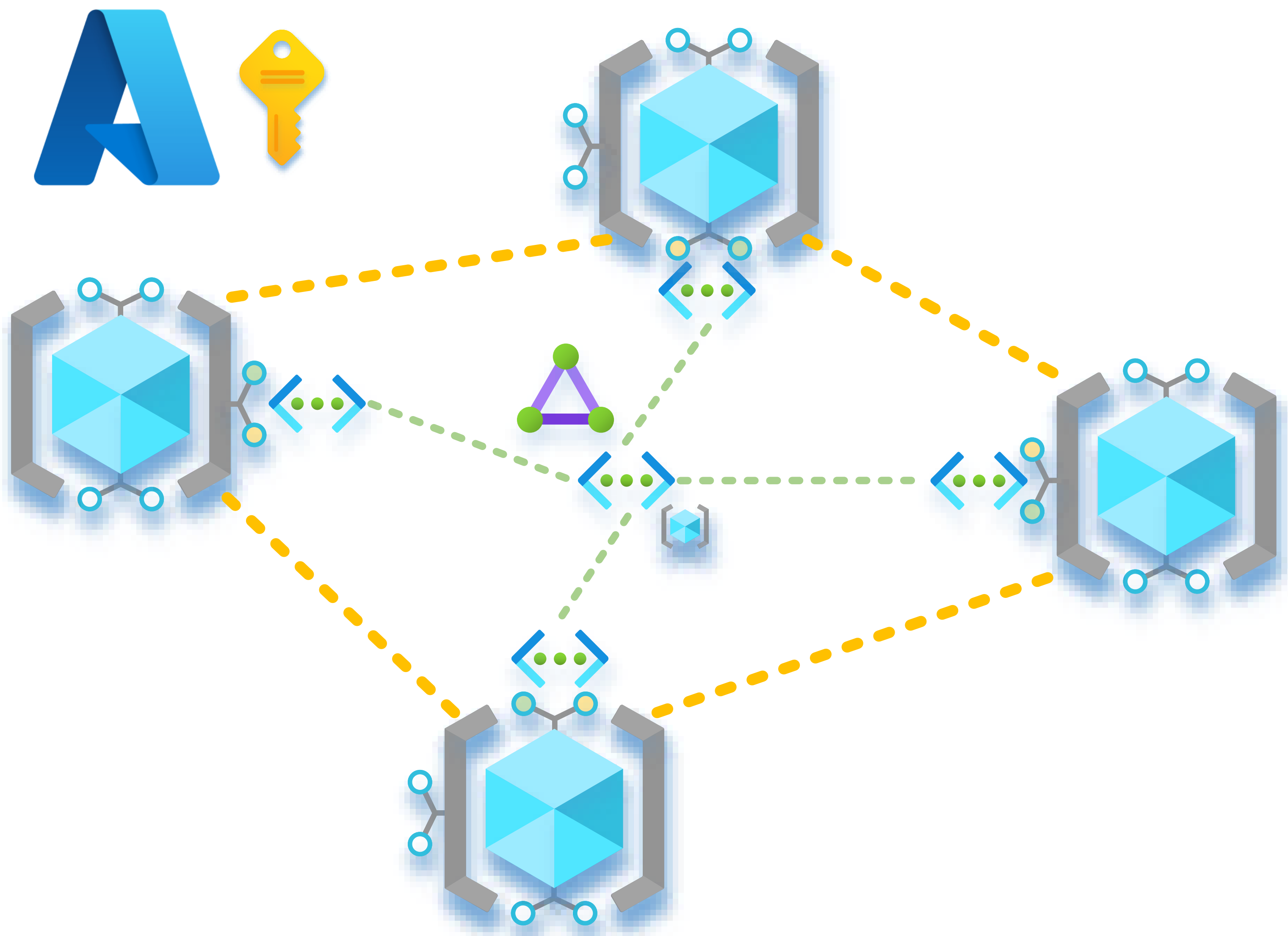




Data Products in Azure



Data Products in Azure with Interfaces

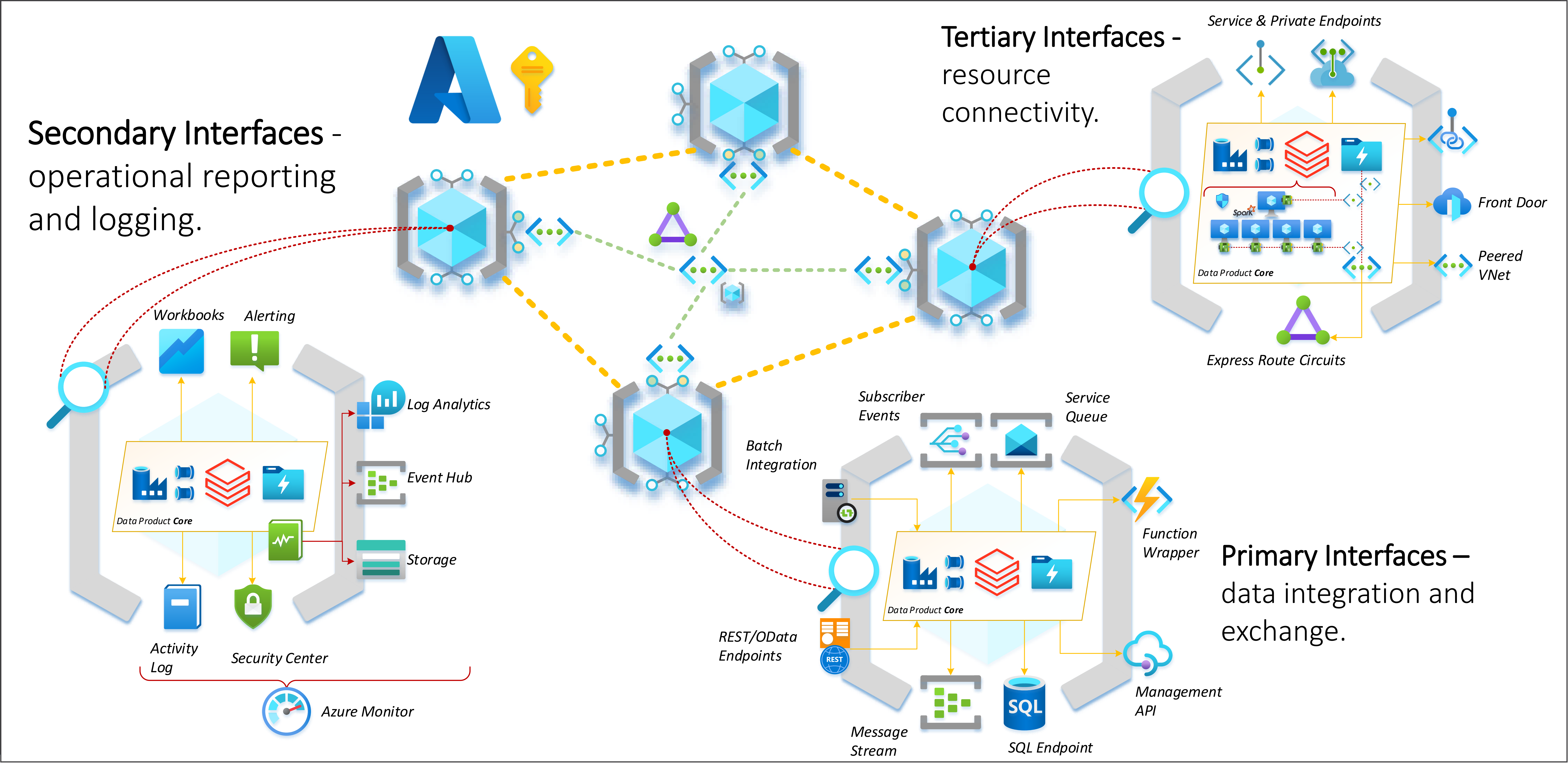


Data Products in Azure with Interfaces

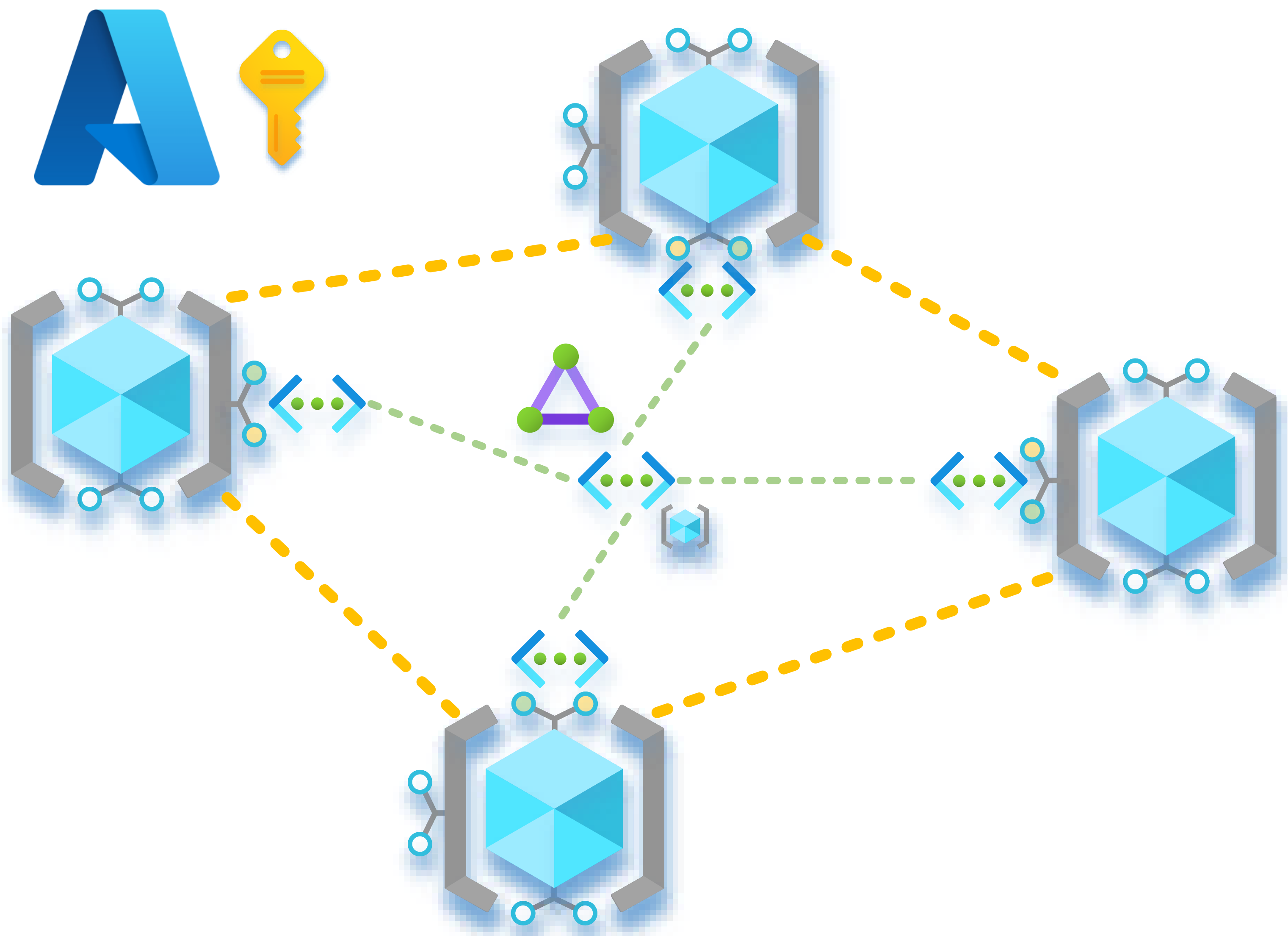
Secondary Interfaces - operational reporting and logging.

Tertiary Interfaces - resource connectivity.

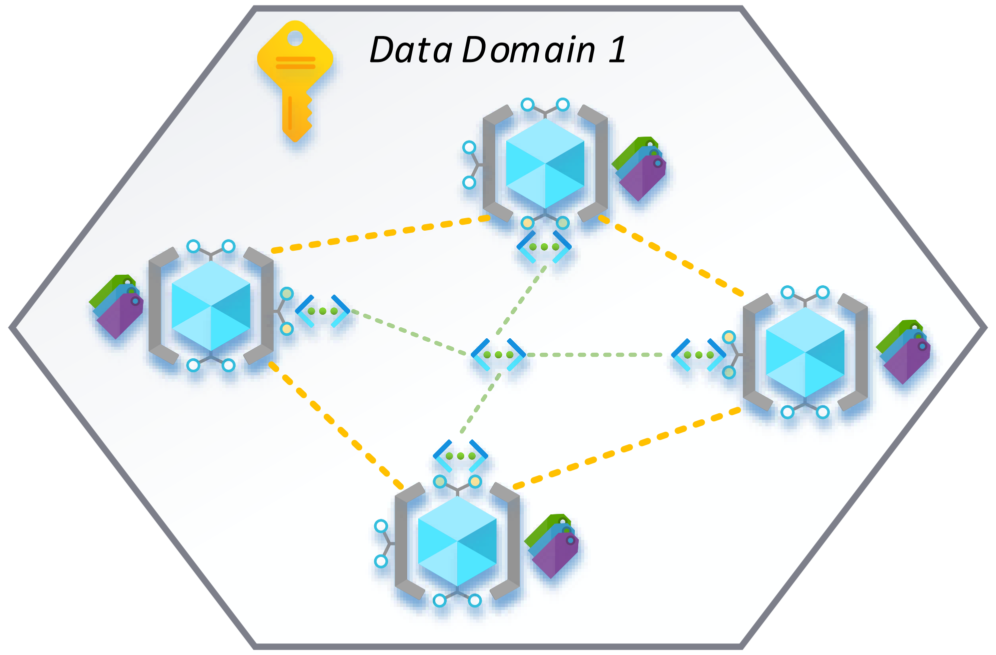
Primary Interfaces – data integration and exchange.



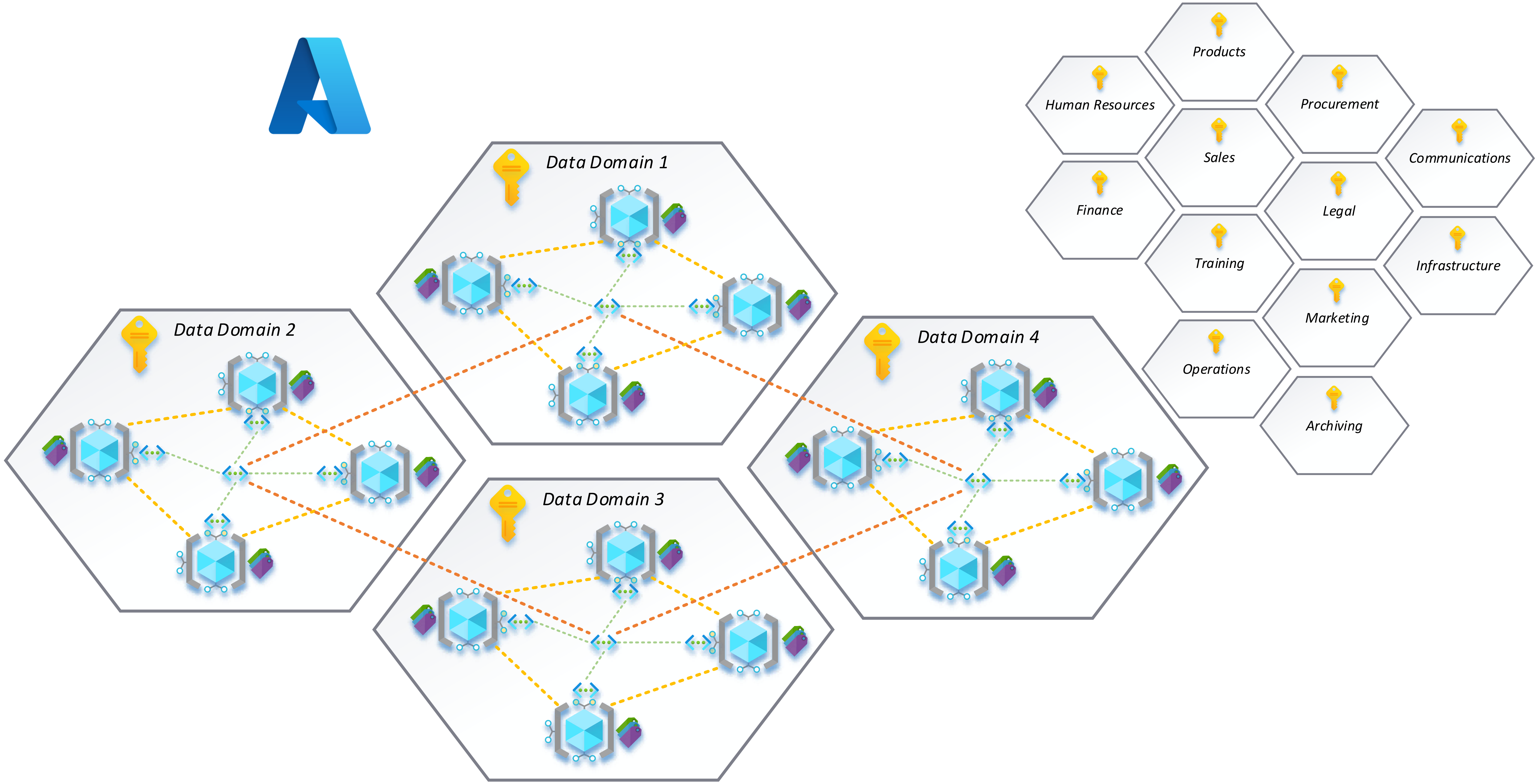
Data Domains in Azure



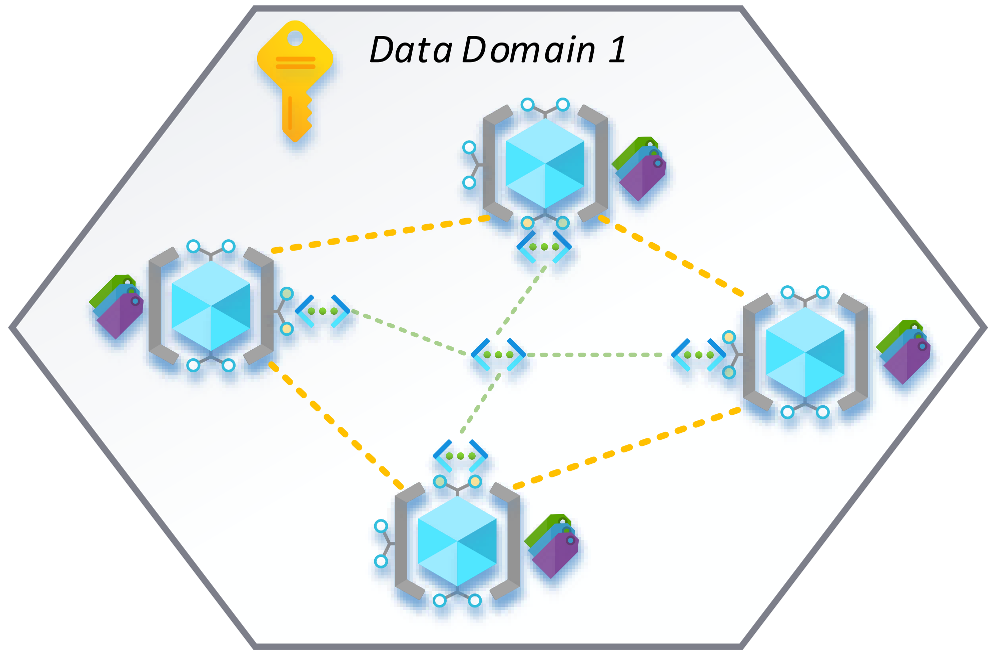
Data Domains in Azure

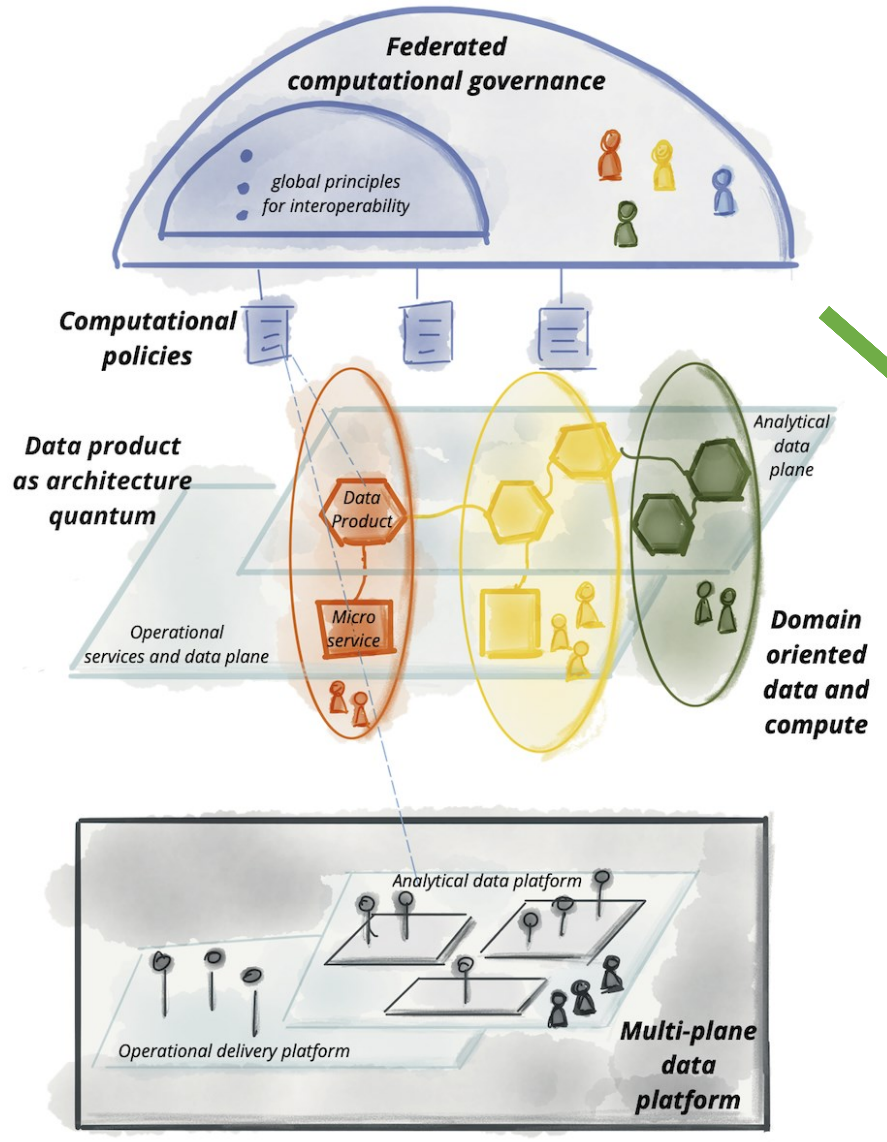


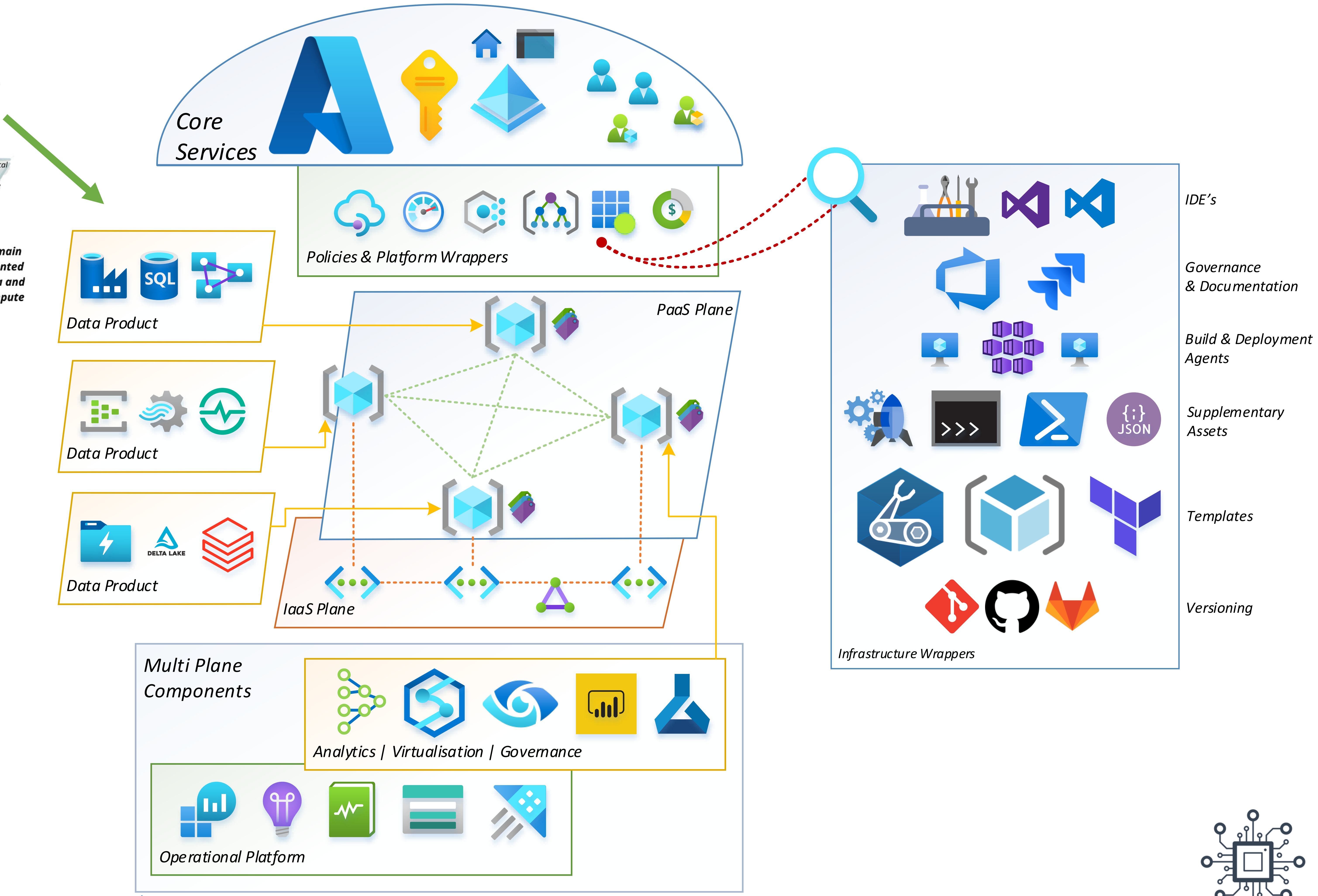
Data Domains in Azure



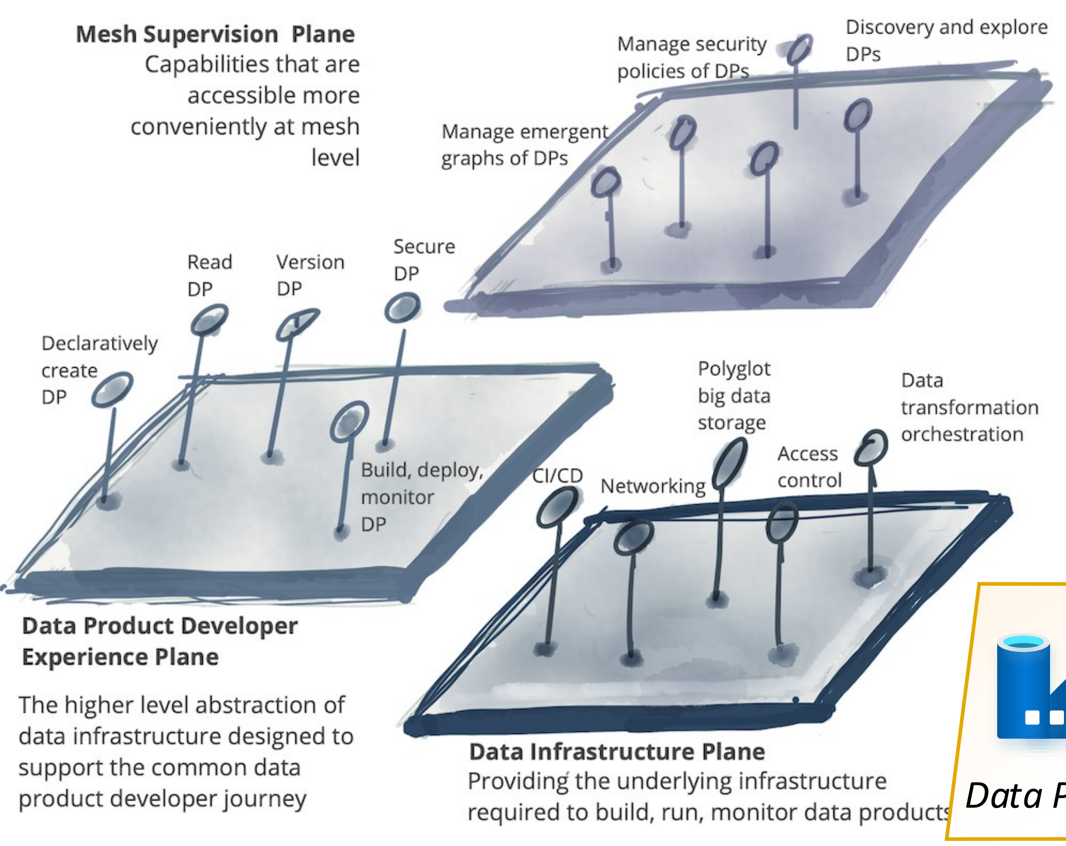
Data Domains in Azure

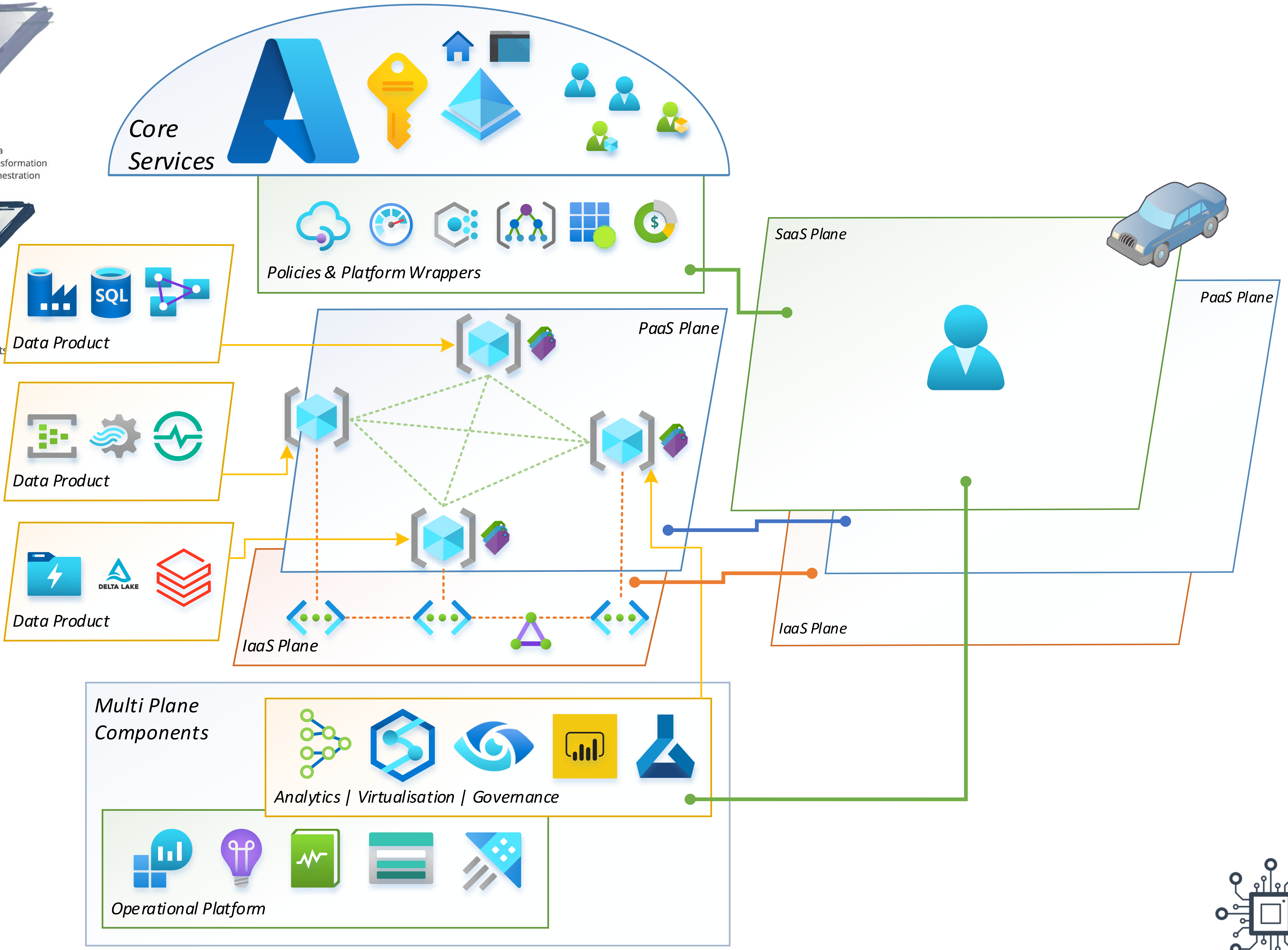



Federated computational governance
global principles for interoperability
Computational policies
Data product as architecture quantum
Data Product
Micro service
Analytical data plane
Operational services and data plane
Domain oriented data and compute
Analytical data platform
Operational delivery platform
Multi-plane data platform



https://mrpaulandrew.com/tag/data-mesh-vs-azure/


Mesh Supervision Plane
Capabilities that are accessible more conveniently at mesh level
Manage security policies of DPs
Discovery and explore DPs
Manage emergent graphs of DPs
Read DP
Version DP
Secure DP
Declaratively create DP
Build, deploy, monitor DP
Data Product Developer Experience Plane
The higher level abstraction of data infrastructure designed to support the common data product developer journey
CI/CD
Networking
Polyglot big data storage
Access control
Data transformation orchestration
Data Infrastructure Plane
Providing the underlying infrastructure required to build, run, monitor data products


Core Services
Policies & Platform Wrappers
SQL
Data Product
Data Product
DELTA LAKE
Data Product
PaaS Plane
IaaS Plane
SaaS Plane
PaaS Plane
IaaS Plane
Multi Plane Components
Analytics | Virtualisation | Governance
Operational Platform

In Summary

An Evolution of Data Platform Architectures in Azure

Lambda, Kappa, Delta, Data Mesh

λ κ δ