databricks

What is a data lakehouse?

History of data management

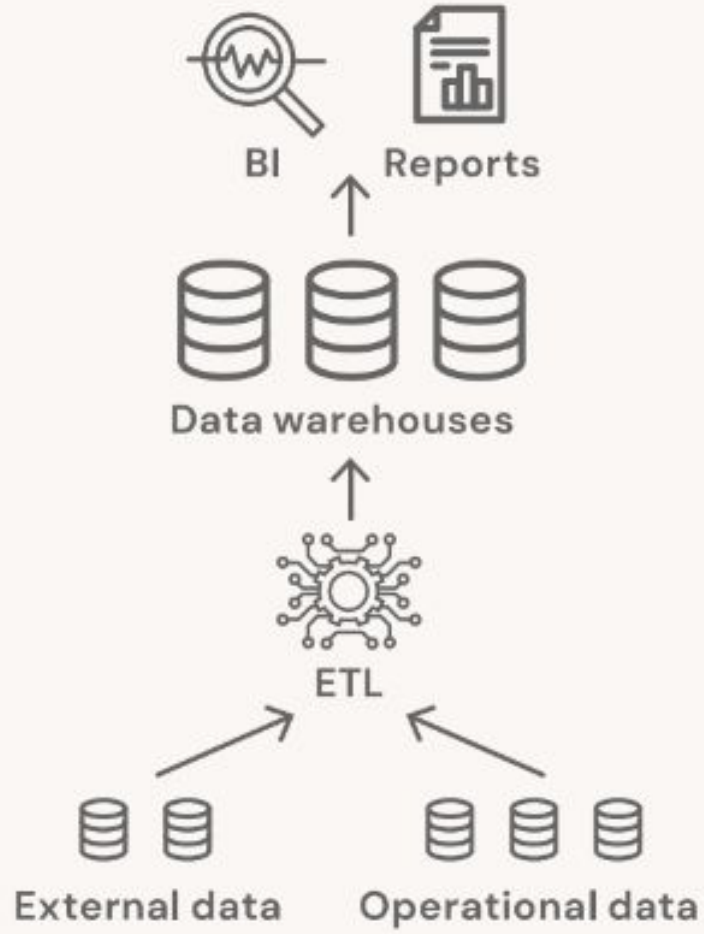
Learning objectives

- Describe the origin and purpose of the data lakehouse.
- Explain the challenges of managing and using big data.

The history of data management and analytics

1980s Businesses need more than relational databases

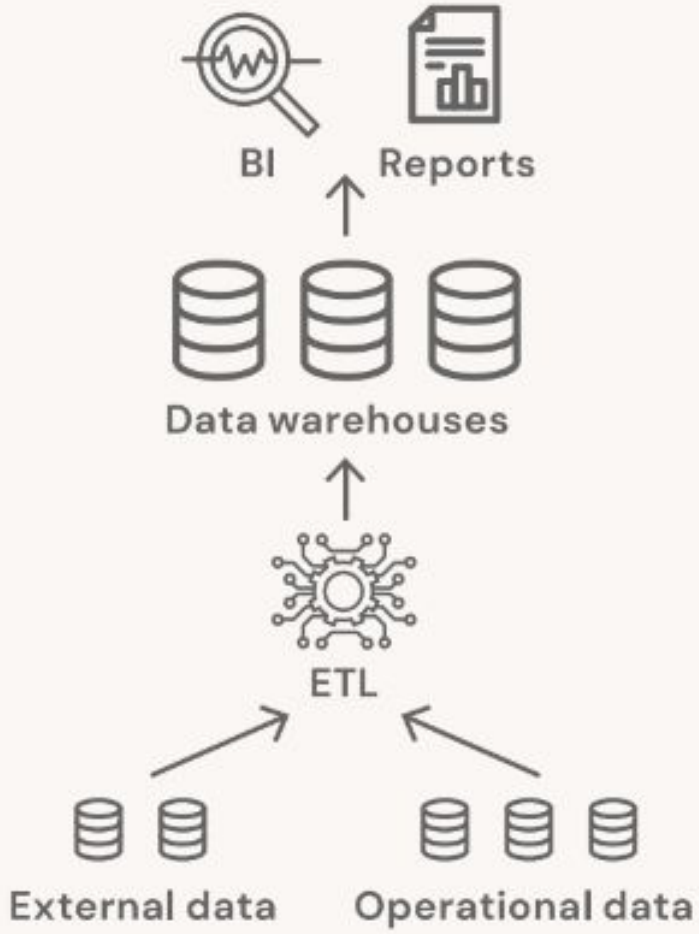
Data warehouse



Pros:

- Business intelligence (BI)
- Analytics
- Structured & clean data
- Predefined schemas

Data warehouse



Cons:

- No support for semi or unstructured data
- Inflexible schemas
- Struggled with volume and velocity upticks
- Long processing time

2000s Big Data explosion

Data Lakes

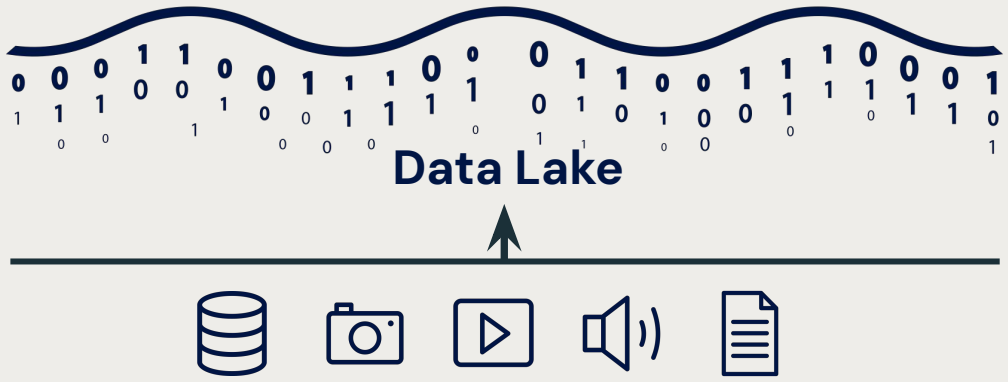
Pros:

- Flexible data storage
- Streaming support
- Cost efficient in the cloud
- Support for AI and Machine Learning

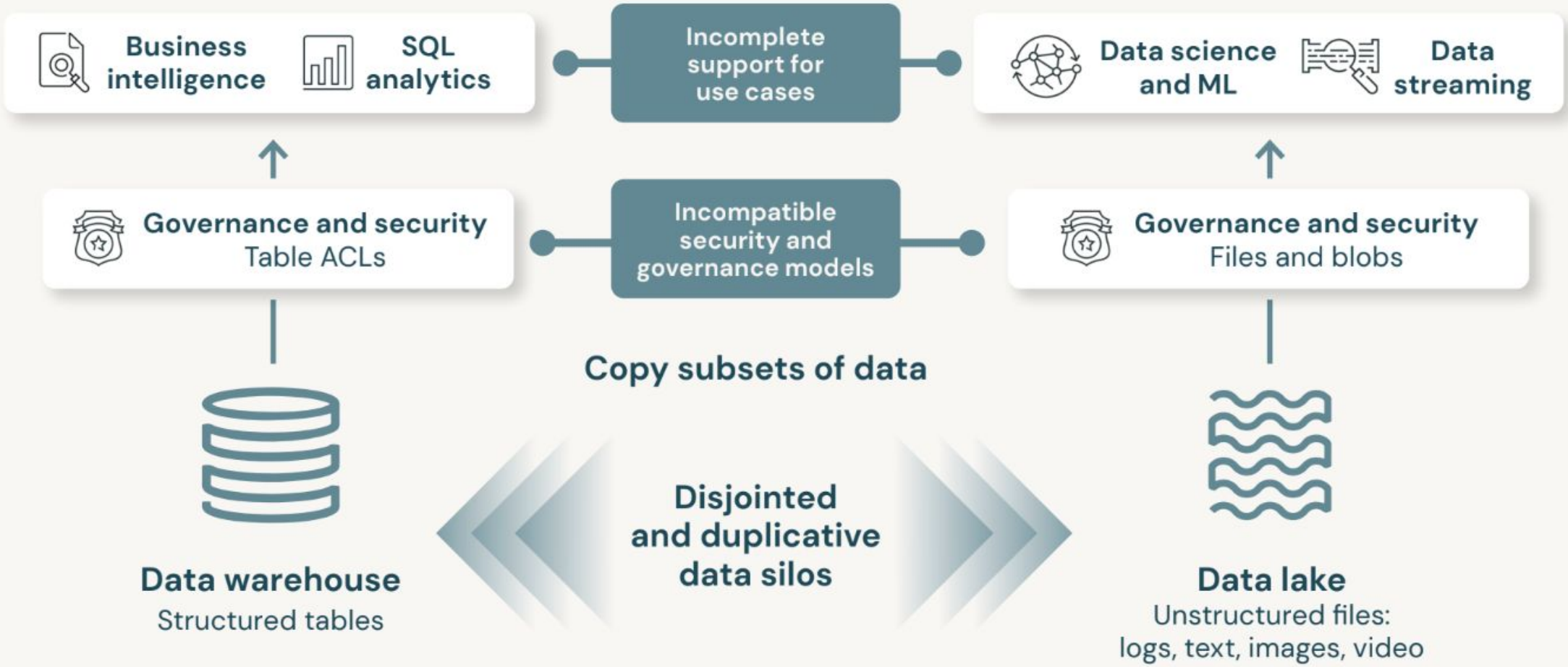
Data Lakes

Cons:

- No transactional support
- Poor data reliability
- Slow analysis performance
- Data governance concerns
- Data warehouses still needed



Business required two disparate, incompatible data platforms

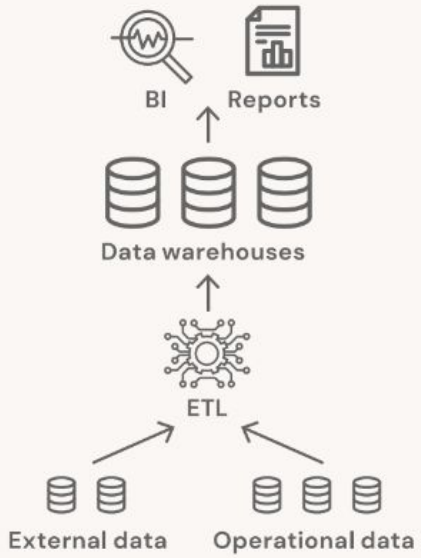


Companies reporting measurable value from data

Data lakehouse

One platform to unify all your data, analytics and AI workloads

Data warehouse



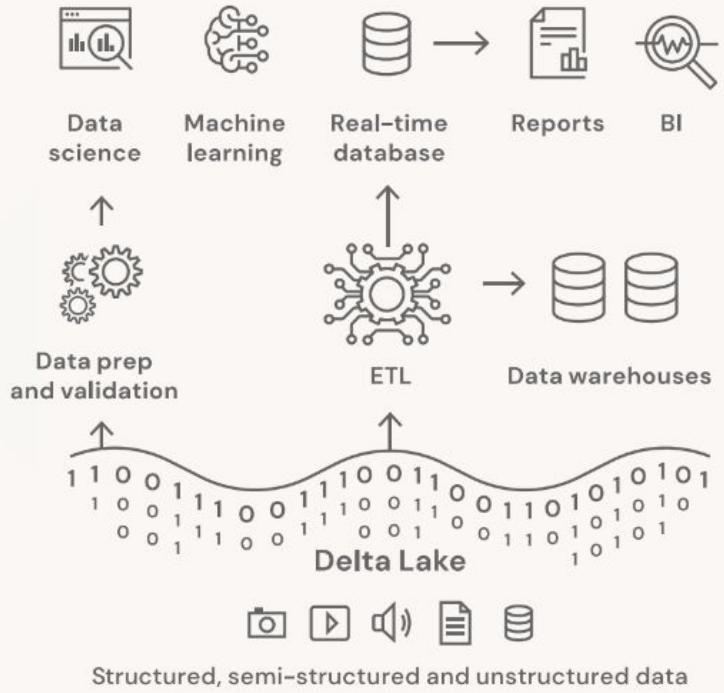
Lakehouse Platform

All machine learning, SQL, BI, and streaming use cases

One security and governance approach for all data assets on all clouds

An open and reliable data platform to efficiently handle all data types

Delta Lake



Key features of a data lakehouse:

- **Transaction support**
- **Schema enforcement and governance**
- **Data governance**
- **BI Support**
- **Decoupled storage from compute**

- **Open storage formats**
- **Support for diverse data types**
- **Support for diverse workloads**
- **End-to-end streaming**

databricks

What is the Databricks Lakehouse Platform?

Databricks and the Data Lakehouse Platform

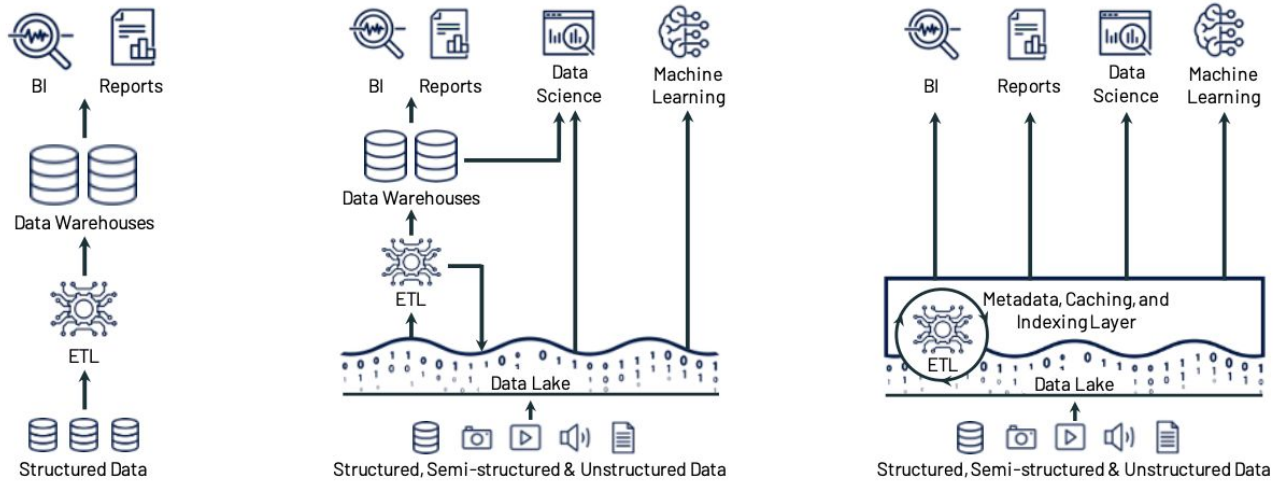
Learning objectives

- Recall the origins of Databricks and the Databricks Lakehouse Platform.
- Define the Databricks Lakehouse Platform.
- Give examples of how the Databricks Lakehouse Platform solves big data challenges.
- Describe how the Databricks Lakehouse Platform benefits Data Engineers, Data Analysts, and Data Scientists.

APACHE Spark™

DELTA LAKE

mlflow™



(a) First-generation platforms. **(b) Current two-tier architectures.** **(c) Lakehouse platforms.**

Figure 1: Evolution of data platform architectures to today's two-tier model (a-b) and the new Lakehouse model (c).

Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics

Michael Armbrust[1], Ali Ghodsi[1,2], Reynold Xin[1], Matei Zaharia[1,3]
[1]Databricks, [2]UC Berkeley, [3]Stanford University

Abstract

This paper argues that the data warehouse architecture as we know it today will wither in the coming years and be replaced by a new architectural pattern, the Lakehouse, which will (i) be based on open direct-access data formats, such as Apache Parquet, (ii) have first-class support for machine learning and data science, and (iii) offer state-of-the-art performance. Lakehouses can help address several major challenges with data warehouses, including data staleness, reliability, total cost of ownership, data lock-in, and limited use-case support. We discuss how the industry is already moving toward Lakehouses and how this shift may affect work in data management. We also report results from a Lakehouse system using Parquet that is competitive with popular cloud data warehouses on TPC-DS.

1 Introduction

This paper argues that the data warehouse architecture as we know it today will wane in the coming years and be replaced by a new architectural pattern, which we refer to as the Lakehouse, characterized by (i) open direct-access data formats, such as Apache Parquet and ORC, (ii) first-class support for machine learning and data science workloads, and (iii) state-of-the-art performance.

The history of data warehousing started with helping business leaders get analytical insights by collecting data from operational databases into centralized warehouses, which then could be used for decision support and business intelligence (BI). Data in these warehouses would be written with schema-on-write, which ensured that the data model was optimized for downstream BI consumption. We refer to this as the first generation data analytics platforms.

A decade ago, the first generation systems started to face several challenges. First, they typically coupled compute and storage into an on-premises appliance. This forced enterprises to provision and pay for the peak of user load and data under management, which became very costly as datasets grew. Second, not only were datasets growing rapidly, but more and more datasets were completely unstructured, e.g., video, audio, and text documents, which data warehouses could not store and query at all.

To solve these problems, the second generation data analytics platforms started offloading all the raw data into *data lakes*: low-cost storage systems with a file API that hold data in generic and usually open file formats, such as Apache Parquet and ORC [8, 9]. This approach started with the Apache Hadoop movement [5], using the Hadoop File System (HDFS) for cheap storage. The data lake was a schema-on-read architecture that enabled the agility of storing any data at low cost, but on the other hand, punted the problem of data quality and governance downstream. In this architecture, a small subset of data in the lake would later be ETLed to a downstream data warehouse (such as Teradata) for the most important decision support and BI applications. The use of open formats also made data lake data directly accessible to a wide range of other analytics engines, such as machine learning systems [30, 37, 42].

From 2015 onwards, cloud data lakes, such as S3, ADLS and GCS, started replacing HDFS. They have superior durability (often >10 nines), geo-replication, and most importantly, extremely low cost with the possibility of automatic, even cheaper, archival storage, e.g., AWS Glacier. The rest of the architecture is largely the same in the cloud as in the second generation systems, with a downstream data warehouse such as Redshift or Snowflake. This two-tier data lake + warehouse architecture is now dominant in the industry in our experience (used at virtually all Fortune 500 enterprises).

This brings us to the challenges with current data architectures. While the cloud data lake and warehouse architecture is ostensibly cheap due to separate storage (e.g., S3) and compute (e.g., Redshift), a two-tier architecture is highly complex for users. In the first generation platforms, all data was ETLed from operational data systems directly into a warehouse. In today's architectures, data is first ETLed into lakes, and then again ELTed into warehouses, creating complexity, delays, and new failure modes. Moreover, enterprise use cases now include advanced analytics such as machine learning, for which *neither* data lakes nor warehouses are ideal. Specifically, today's data architectures commonly suffer from four problems:

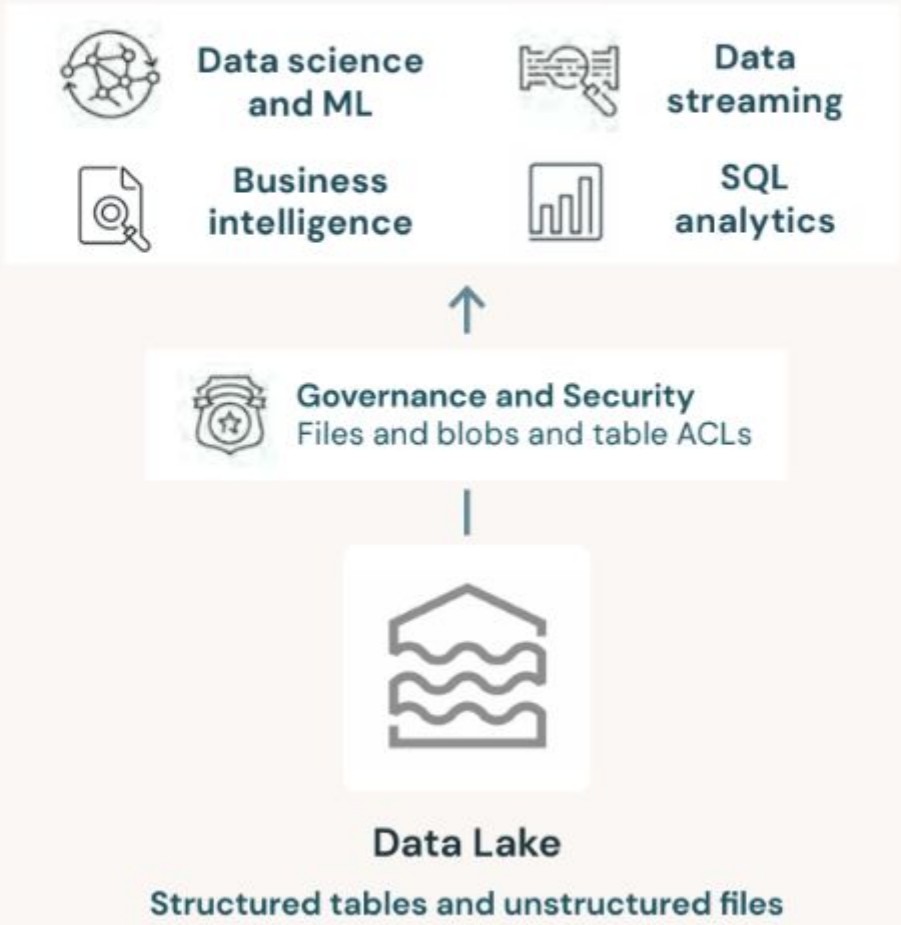
Reliability. Keeping the data lake and warehouse consistent is difficult and costly. Continuous engineering is required to ETL data between the two systems and make it available to high-performance decision support and BI. Each ETL step also risks incurring failures or introducing bugs that reduce data quality, e.g., due to subtle differences between the data lake and warehouse engines.

Data staleness. The data in the warehouse is stale compared to that of the data lake, with new data frequently taking days to load. This is a step back compared to the first generation of analytics systems, where new operational data was immediately available for queries. According to a survey by Dimensional Research and Fivetran, 86% of analysts use out-of-date data and 62% report waiting on engineering resources numerous times per month [47].

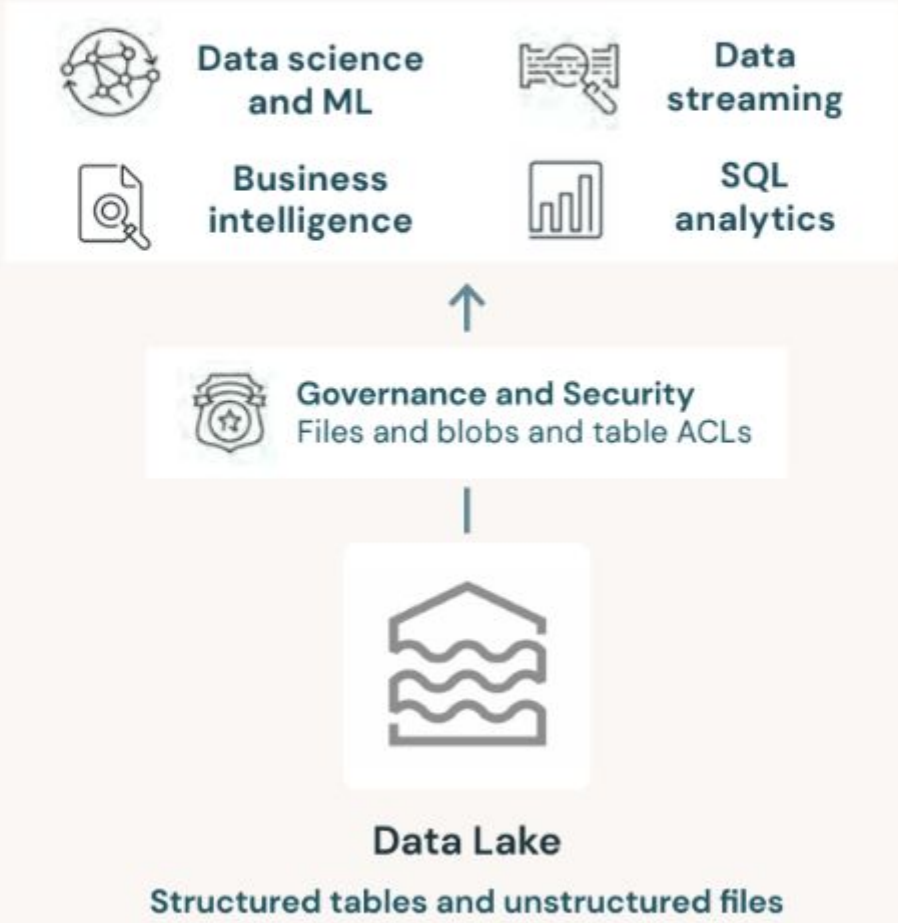
Limited support for advanced analytics. Businesses want to ask predictive questions using their warehousing data, e.g., "which customers should I offer discounts to?" Despite much research on the confluence of ML and data management, none of the leading machine learning systems, such as TensorFlow, PyTorch and XGBoost, work well on top of warehouses. Unlike BI queries, which extract a small amount of data, these systems need to process large datasets using complex non-SQL code. Reading this data via ODBC/JDBC is inefficient, and there is no way to directly access the internal

Lakehouse: A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics. M. Armbrust, A. Ghodsi, R. Xin, M. Zaharia. 11th Annual Conference on Innovative Data Systems Research (CIDR '21), January 11–15, 2021, Online.

This is the lakehouse paradigm



This is the lakehouse paradigm

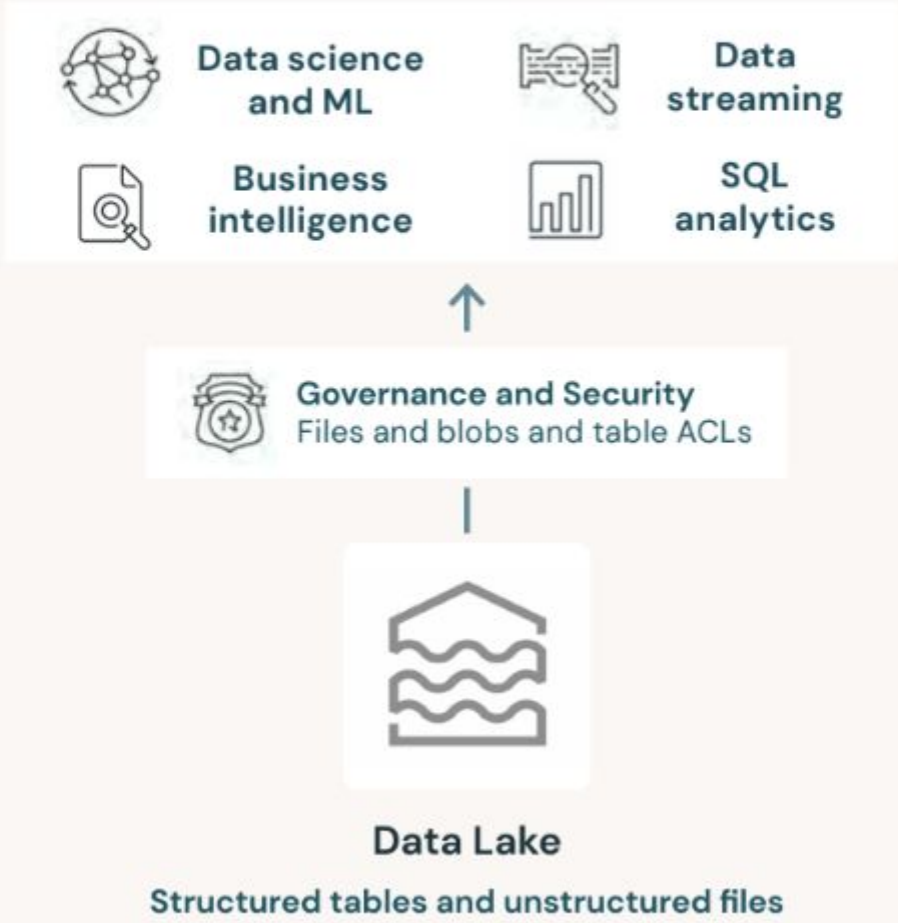


One security and governance approach for all data assets on all clouds

This is the lakehouse paradigm

One security and governance approach for all data assets on all clouds

A reliable data platform to efficiently handle all data types

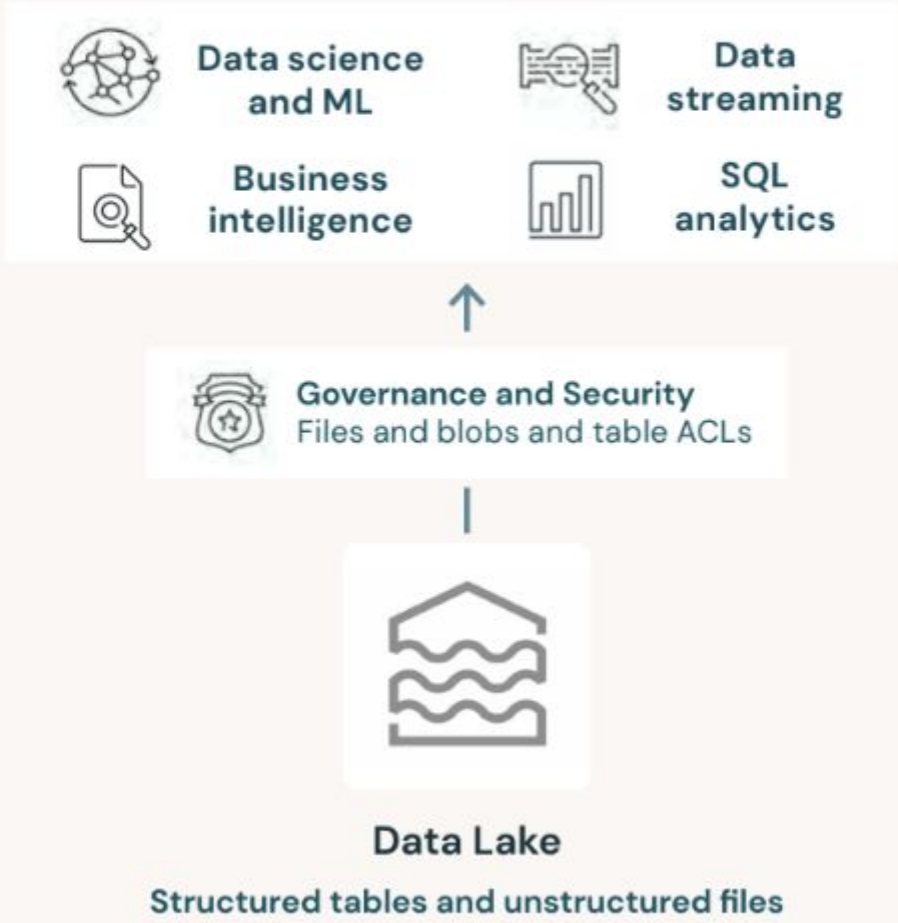


This is the lakehouse paradigm

All ML, SQL, BI,
and streaming use cases

One security and governance
approach for all data assets
on all clouds

A reliable data platform
to efficiently handle
all data types

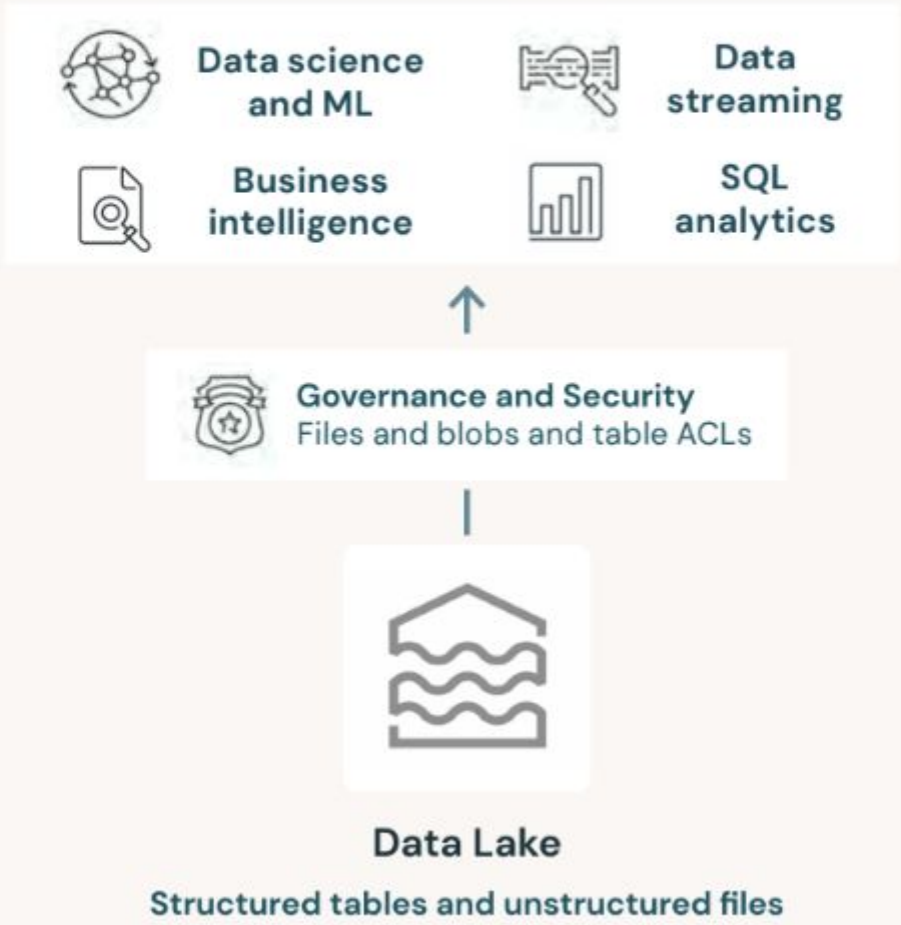


This is the lakehouse paradigm

All ML, SQL, BI,
and streaming use cases

One security and governance
approach for all data assets
on all clouds

A reliable data platform
to efficiently handle
all data types

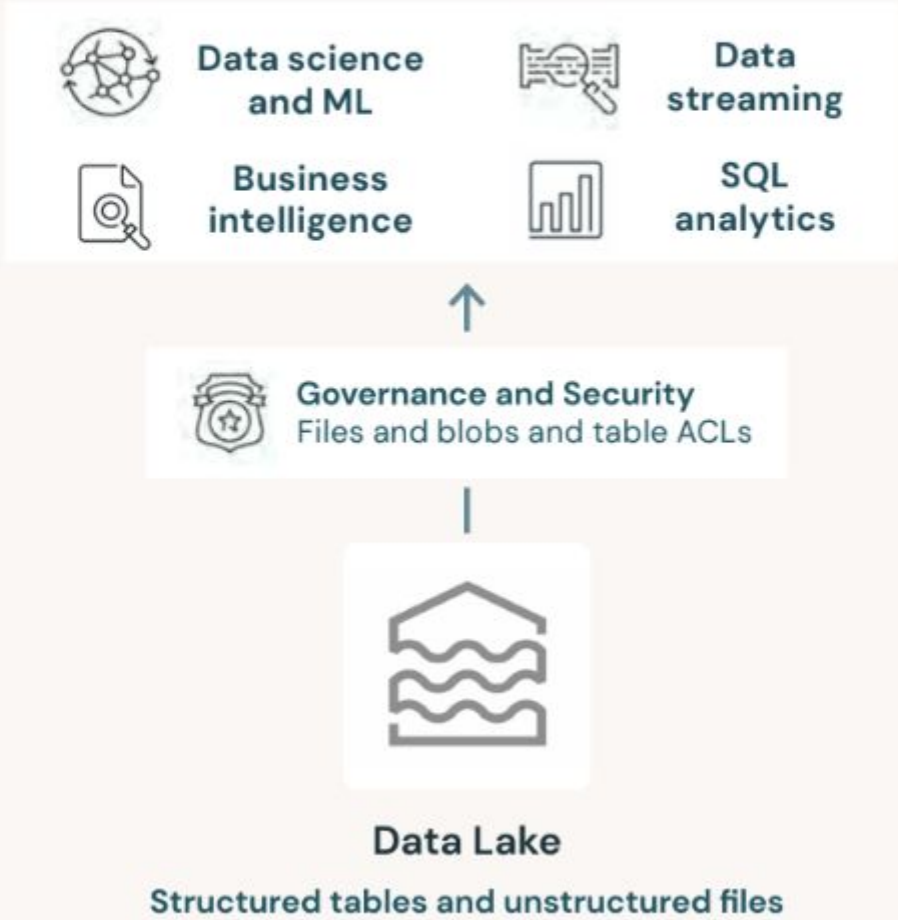


This is the lakehouse paradigm

All ML, SQL, BI,
and streaming use cases

One security and governance
approach for all data assets
on all clouds

A reliable data platform
to efficiently handle
all data types



Realized on Databricks

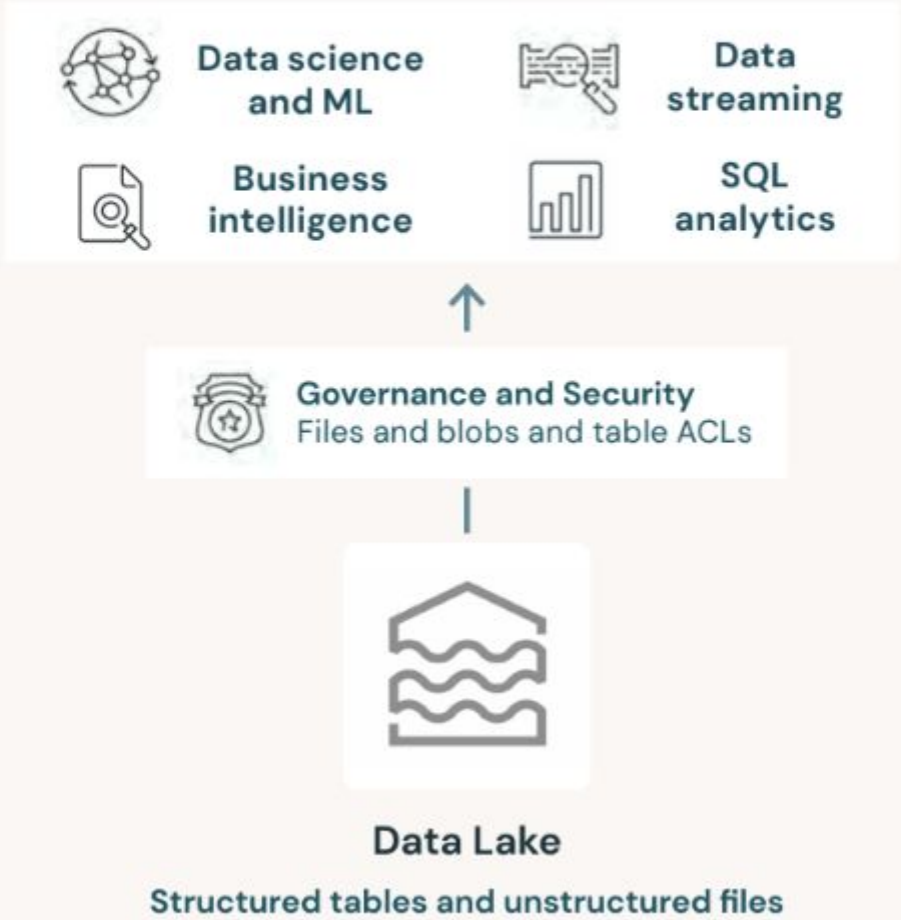
Delta Lake
Data reliability and performance

This is the lakehouse paradigm

All ML, SQL, BI,
and streaming use cases

One security and governance
approach for all data assets
on all clouds

A reliable data platform
to efficiently handle
all data types



Realized on Databricks

Unity Catalog
Fine-grained governance
for data and AI

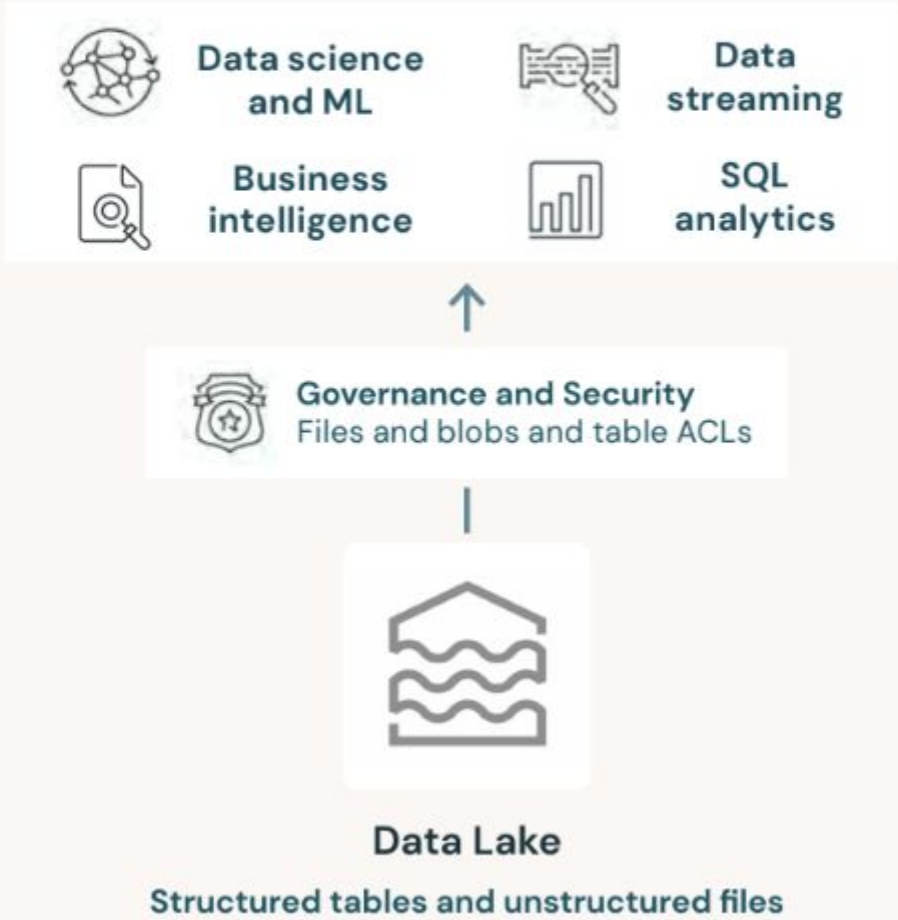
Delta Lake
Data reliability and performance

This is the lakehouse paradigm

All ML, SQL, BI, and streaming use cases

One security and governance approach for all data assets on all clouds

A reliable data platform to efficiently handle all data types

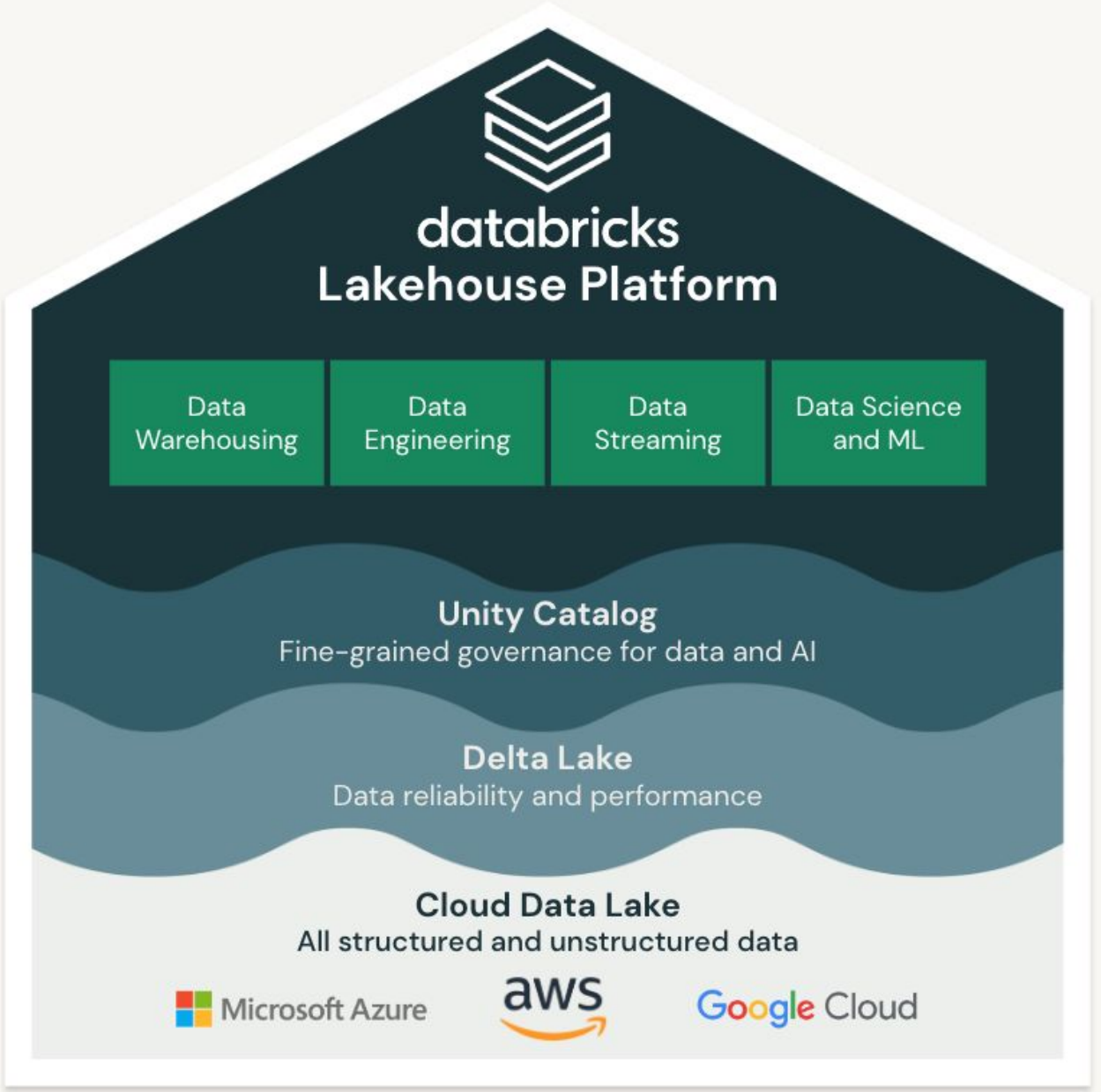


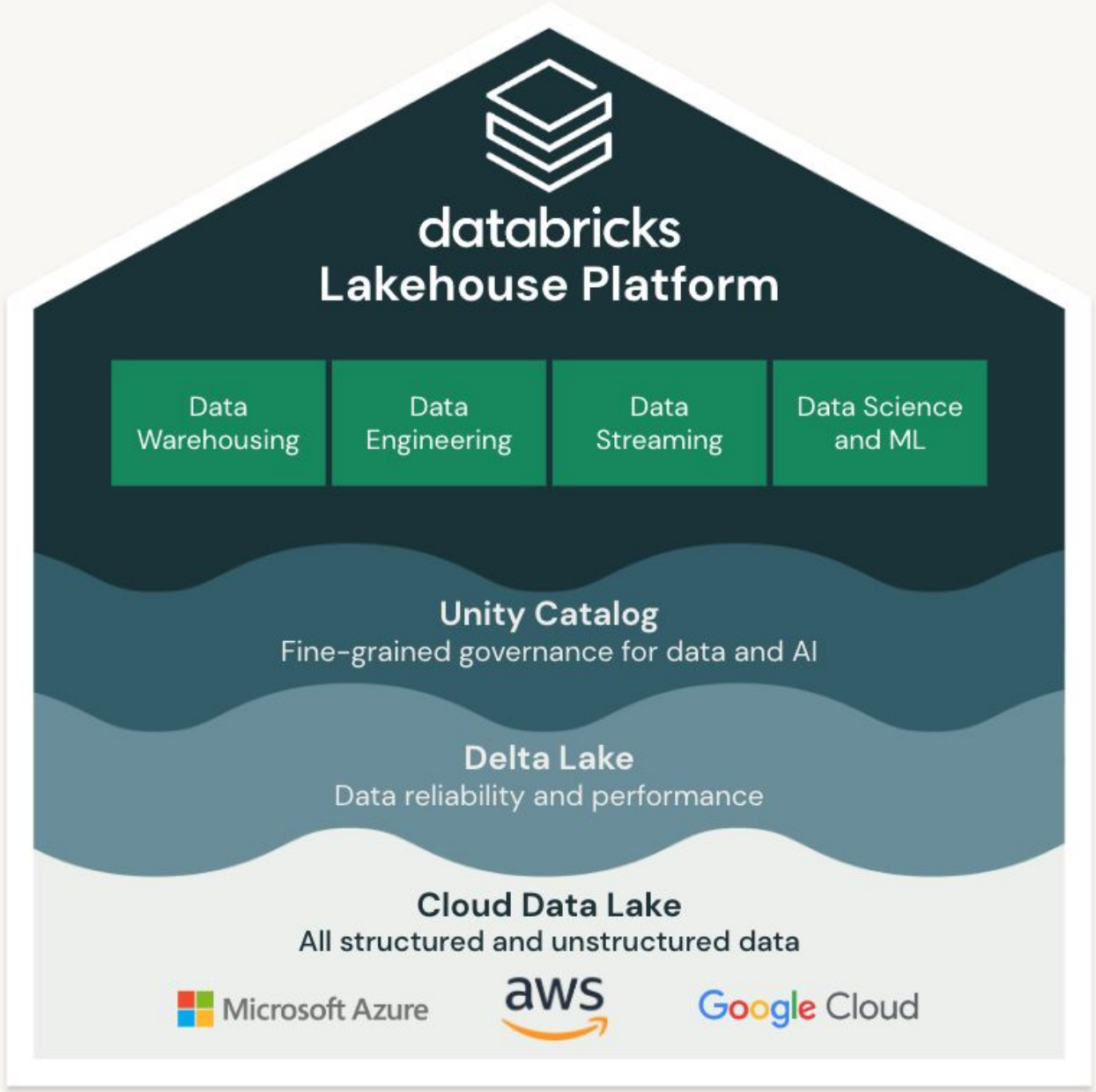
Realized on Databricks

Persona-based use cases

Unity Catalog
Fine-grained governance for data and AI

Delta Lake
Data reliability and performance





Databricks Lakehouse Platform

Simple

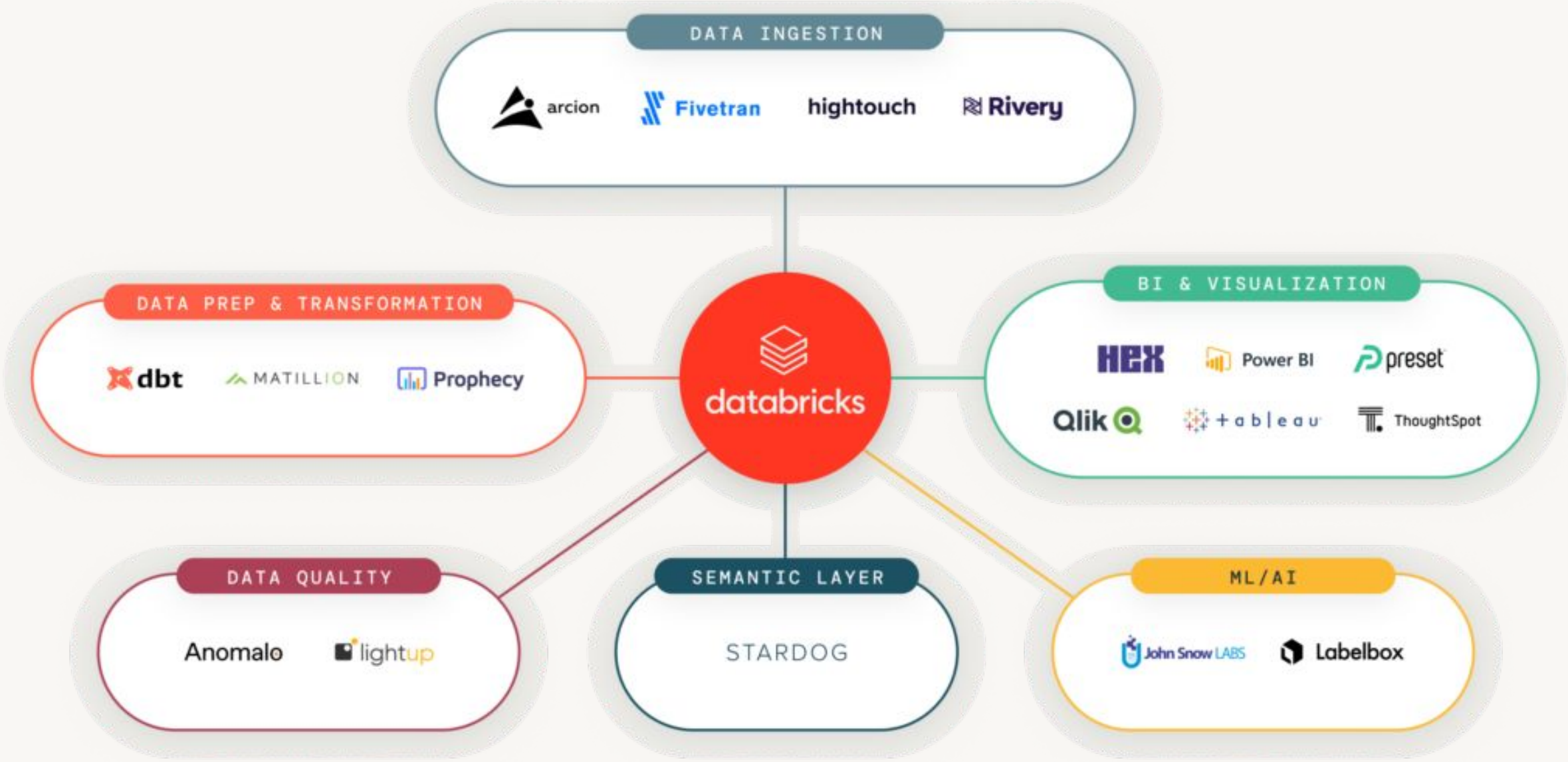
Unify your data warehousing and AI use cases on a single platform

Open

Built on open source and open standards

Multicloud

One consistent data platform across clouds



databricks

Databricks Lakehouse Platform Architecture and Security Fundamentals

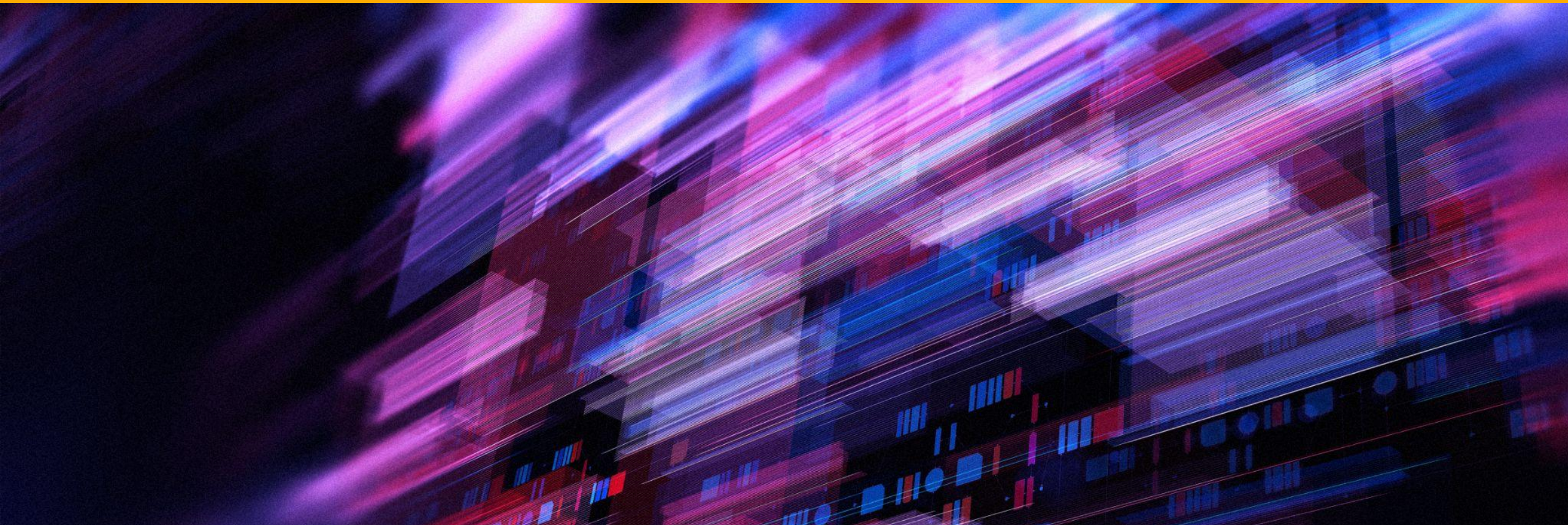
Data reliability and performance

Learning objectives

- Explain the importance of data reliability and performance on platform architecture.
- Define Delta Lake and its features.
- Describe how Photon improves performance of the Databricks Lakehouse Platform.

Why is data reliability and performance important?

bad data in = bad data out



Problems encountered when using data lakes

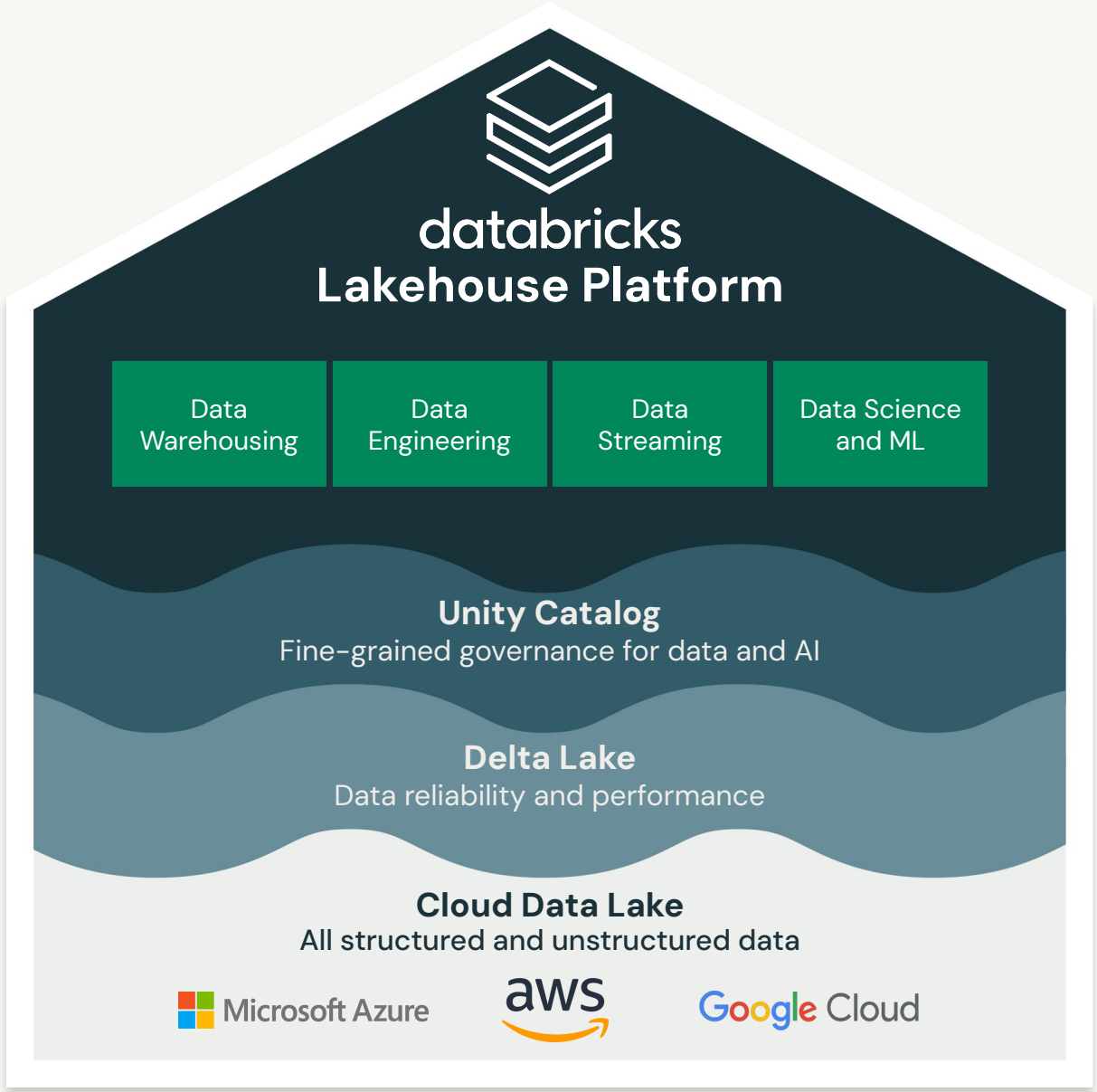
- Lack of ACID transaction support
- Lack of schema enforcement
- Lack of integration with a data catalog
- Ineffective partitioning
- Too many small files

Databricks Lakehouse Platform

Photon

Delta Lake

- ACID transaction guarantees
- Scalable data and metadata handling
- Audit history and time travel
- Schema enforcement and schema evolution
- Support for deletes, updates, and merges
- Unified streaming and batch data processing



Additional points:

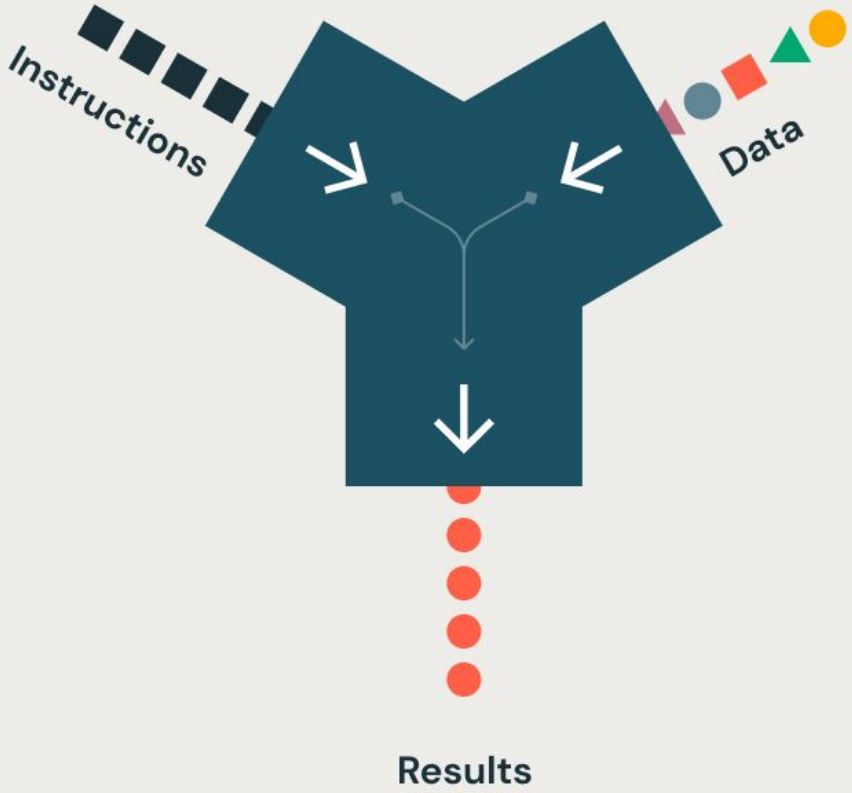
- Compatible with Apache Spark™
- Uses Delta Tables
- Has a transaction log
- Is an open-source project

Delta Lake integrates with all major analytics tools

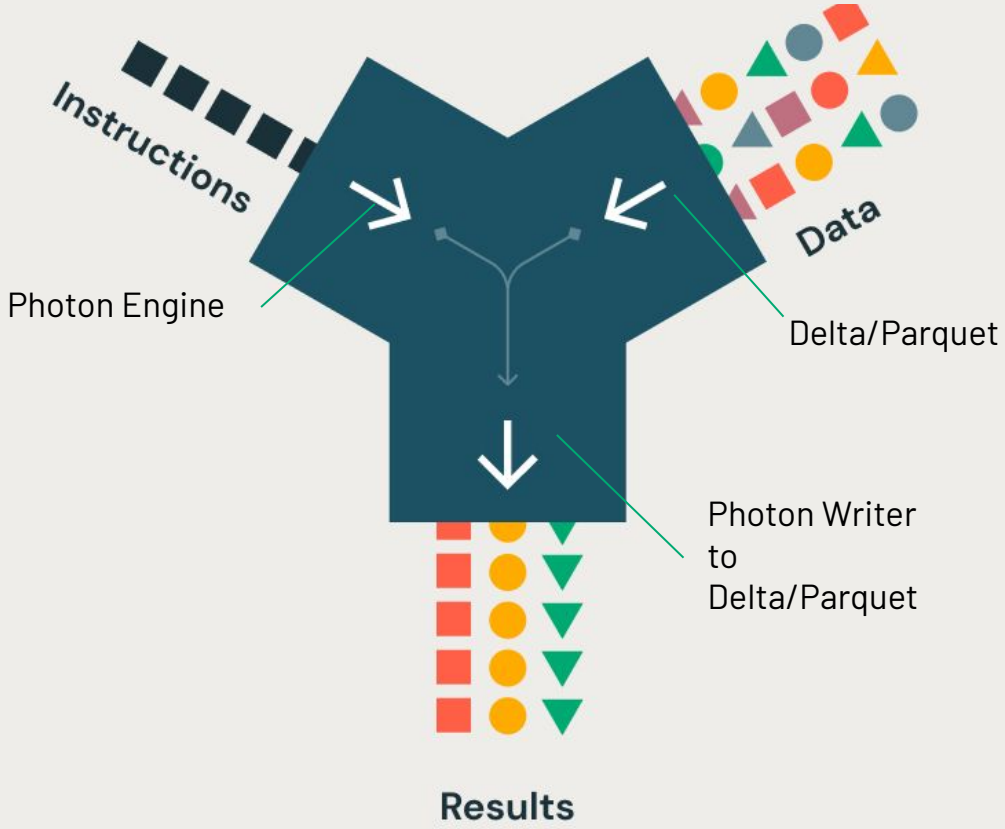
Eliminates unnecessary data movement and duplication

What is Photon?

Spark Instructions

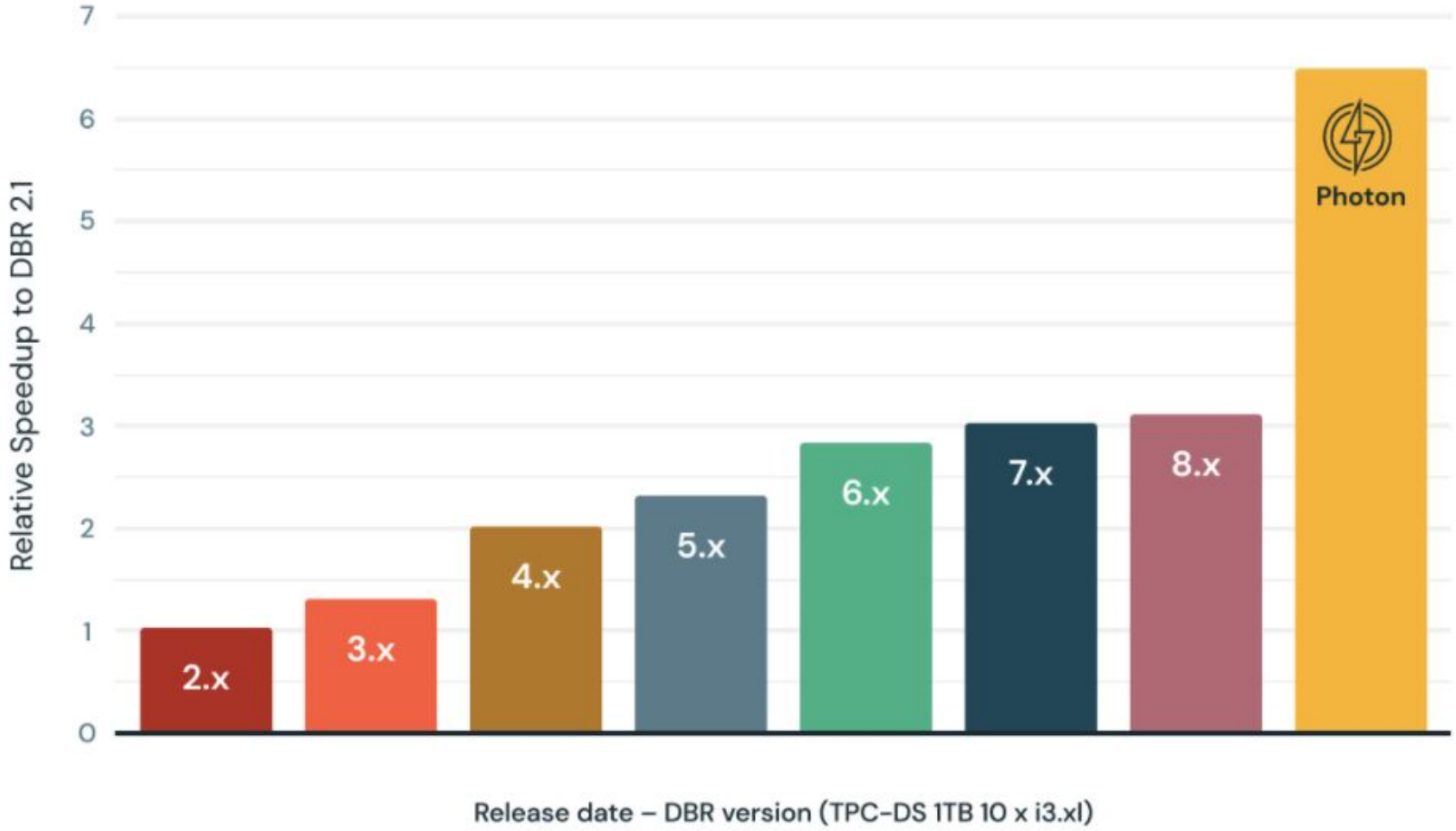


Photon Instructions



Relative Speedup to DBR 2.1 by DBR version

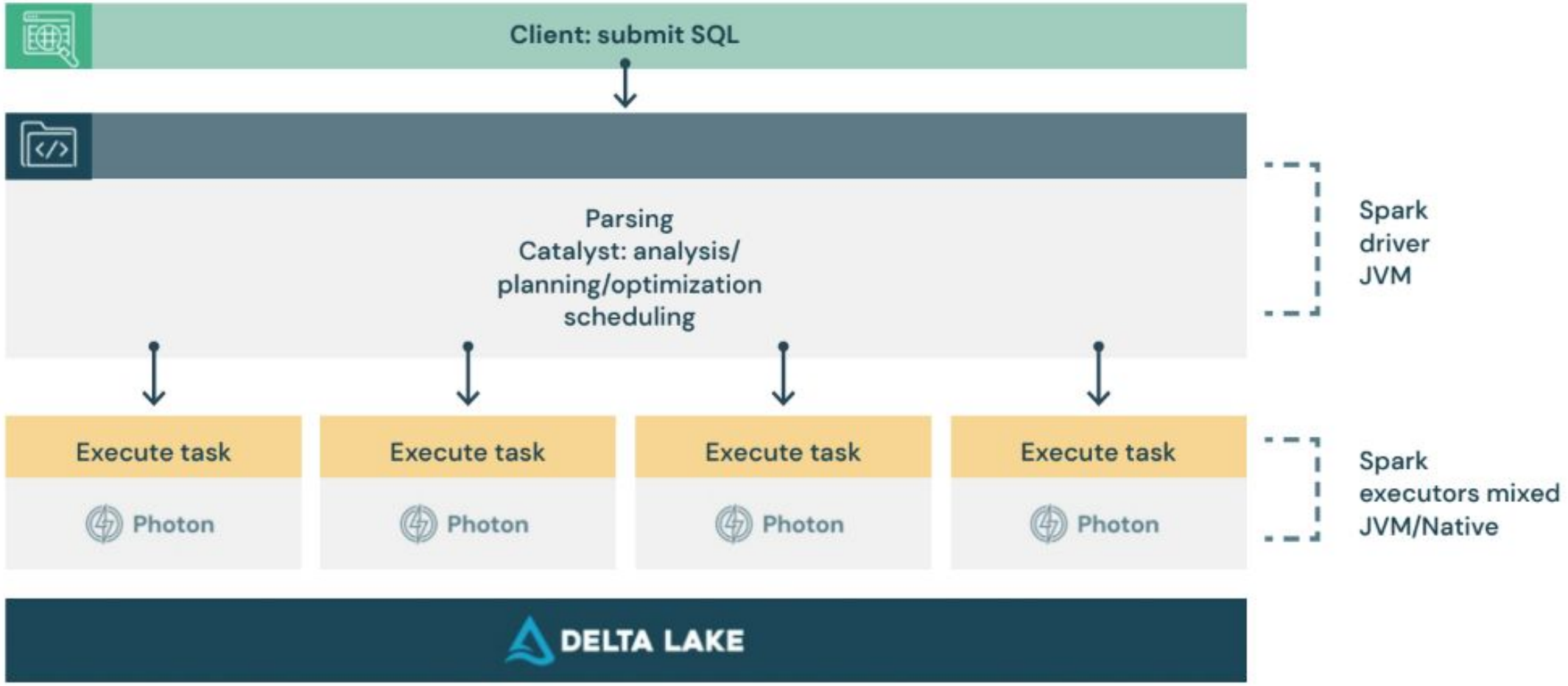
Higher is better



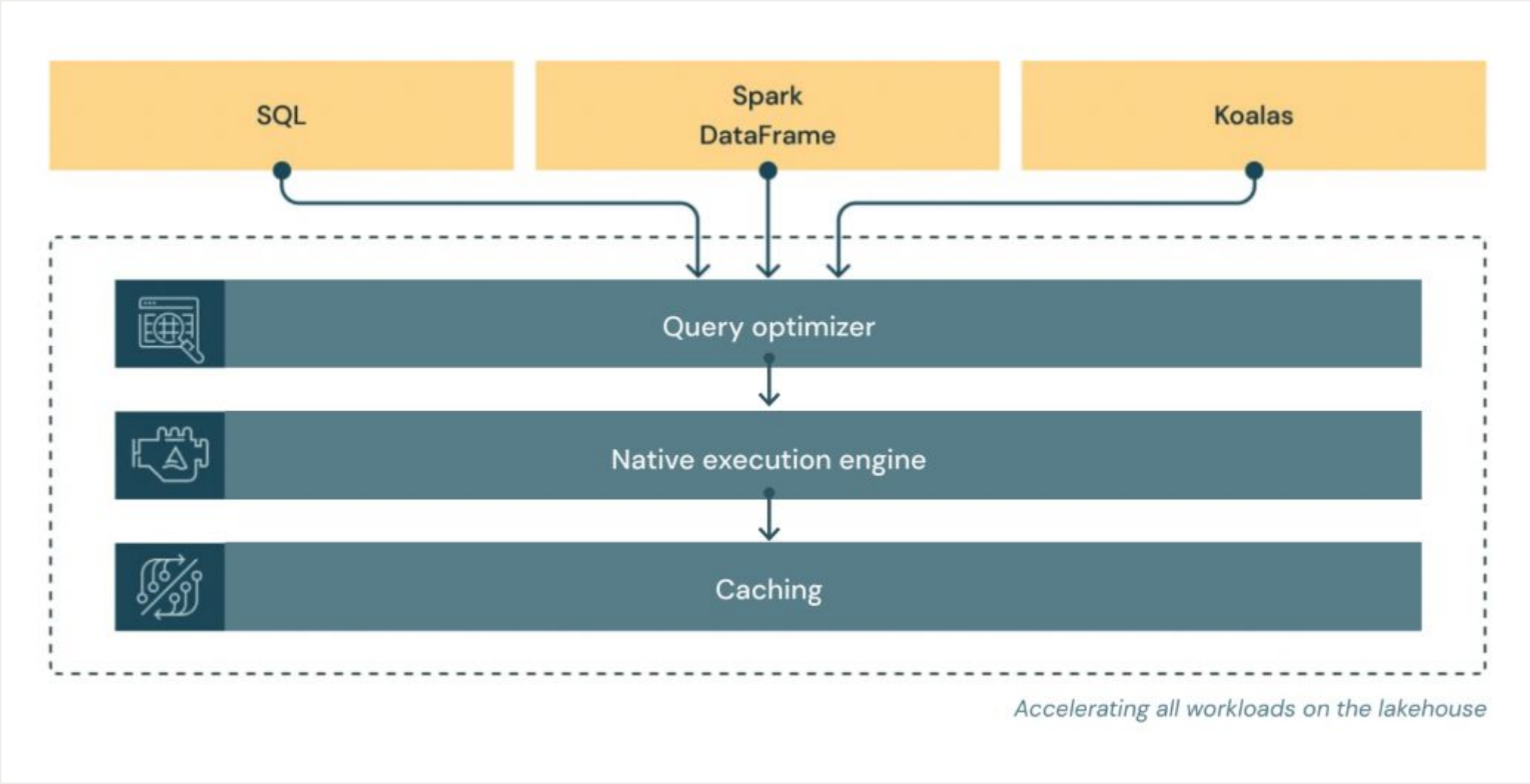
Reported workloads impacted by Photon

- SQL-based jobs
- IoT use cases
- Data privacy and compliance
- Loading data into Delta and Parquet

Photon in the Databricks Lakehouse Platform



Lifecycle of a Photon query



databricks

Databricks Lakehouse Platform Architecture and Security Fundamentals

Unified governance and security

Learning objectives

- Explain how important unified governance and security is to platform architecture.
- Recognize the security features of the Databricks Lakehouse Platform.
- Explain how Unity Catalog and Delta Sharing are used.
- Differentiate between the control plane and the data plane in the Databricks Lakehouse Platform architecture.

Why is a unified governance and security structure important?

Challenges to data and AI governance

- Diversity of data and AI assets
- Using two disparate and incompatible data platforms
- Rise of multi-cloud adoption
- Fragmented tool usage for data governance

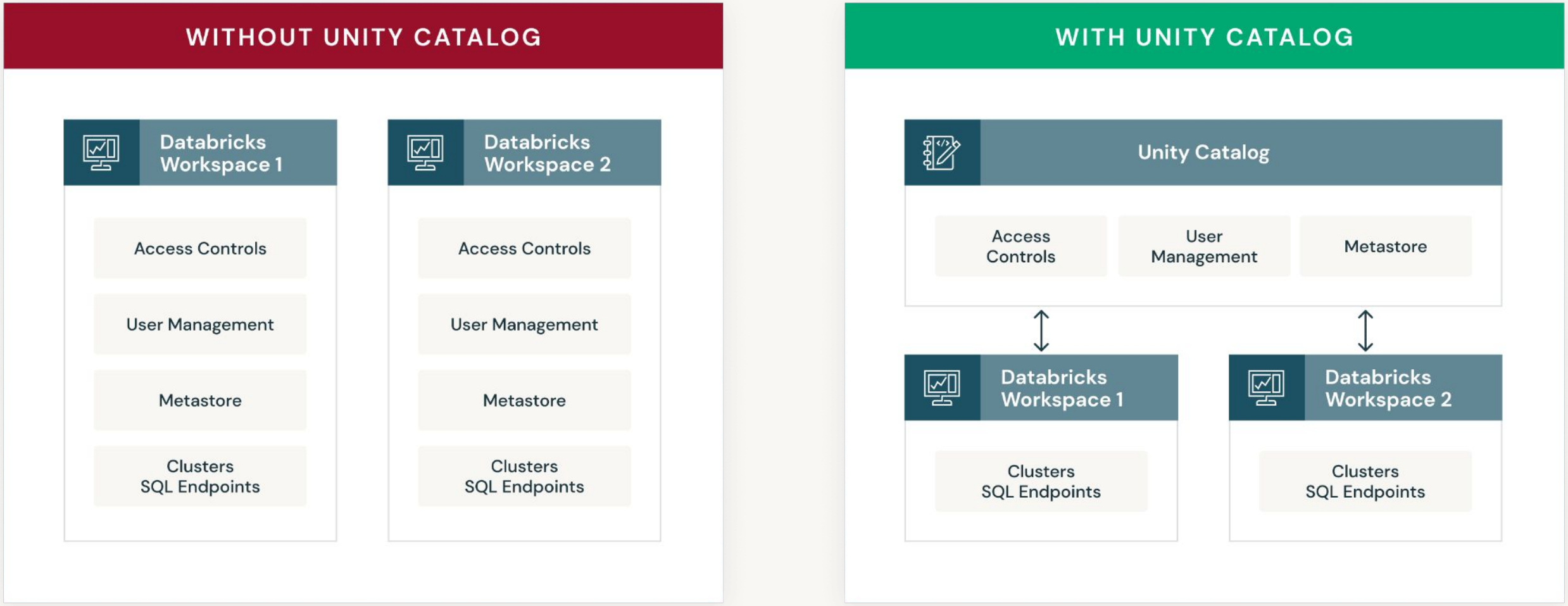
How the Databricks Lakehouse Platform solves data and AI governance challenges

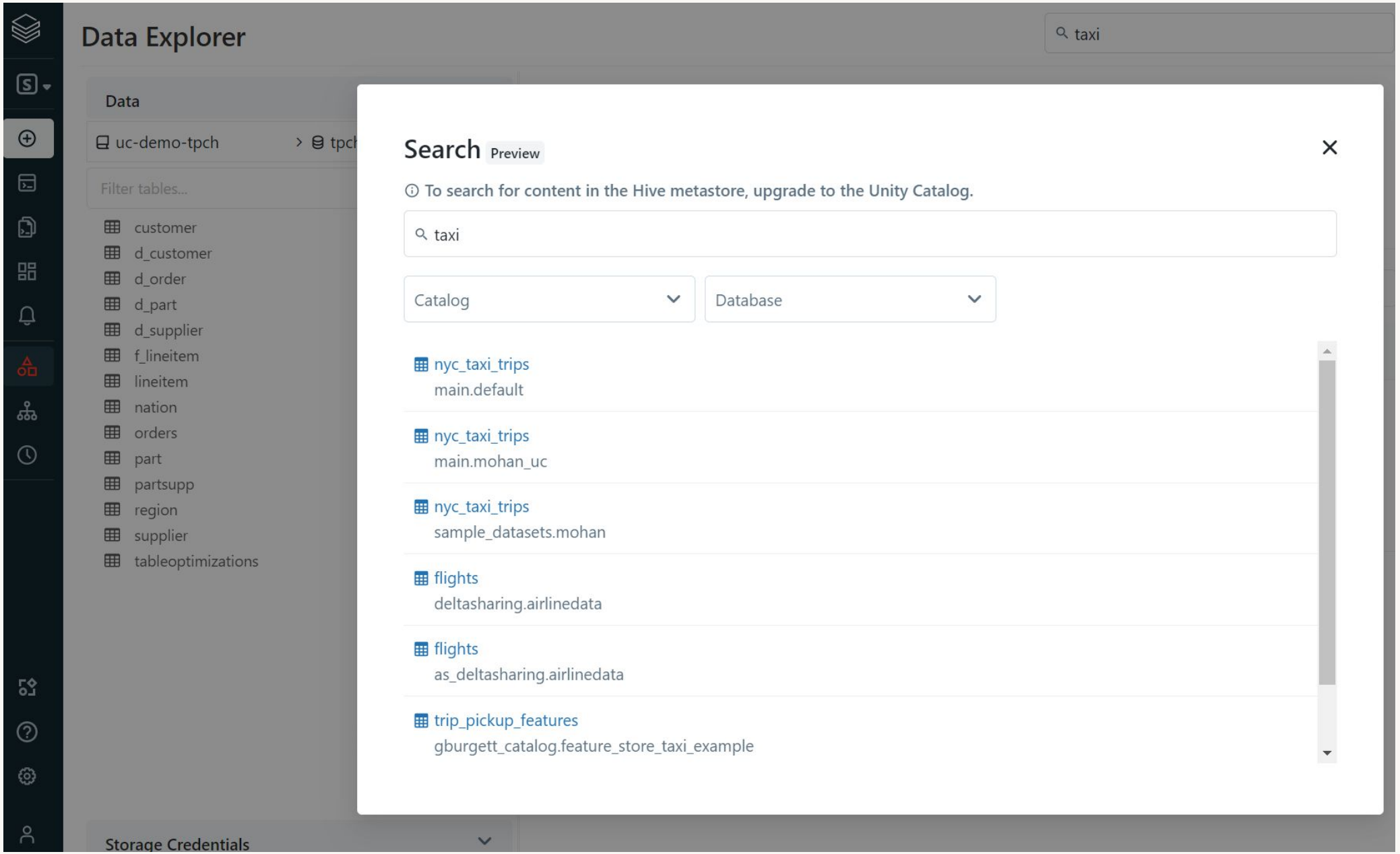
Unity Catalog

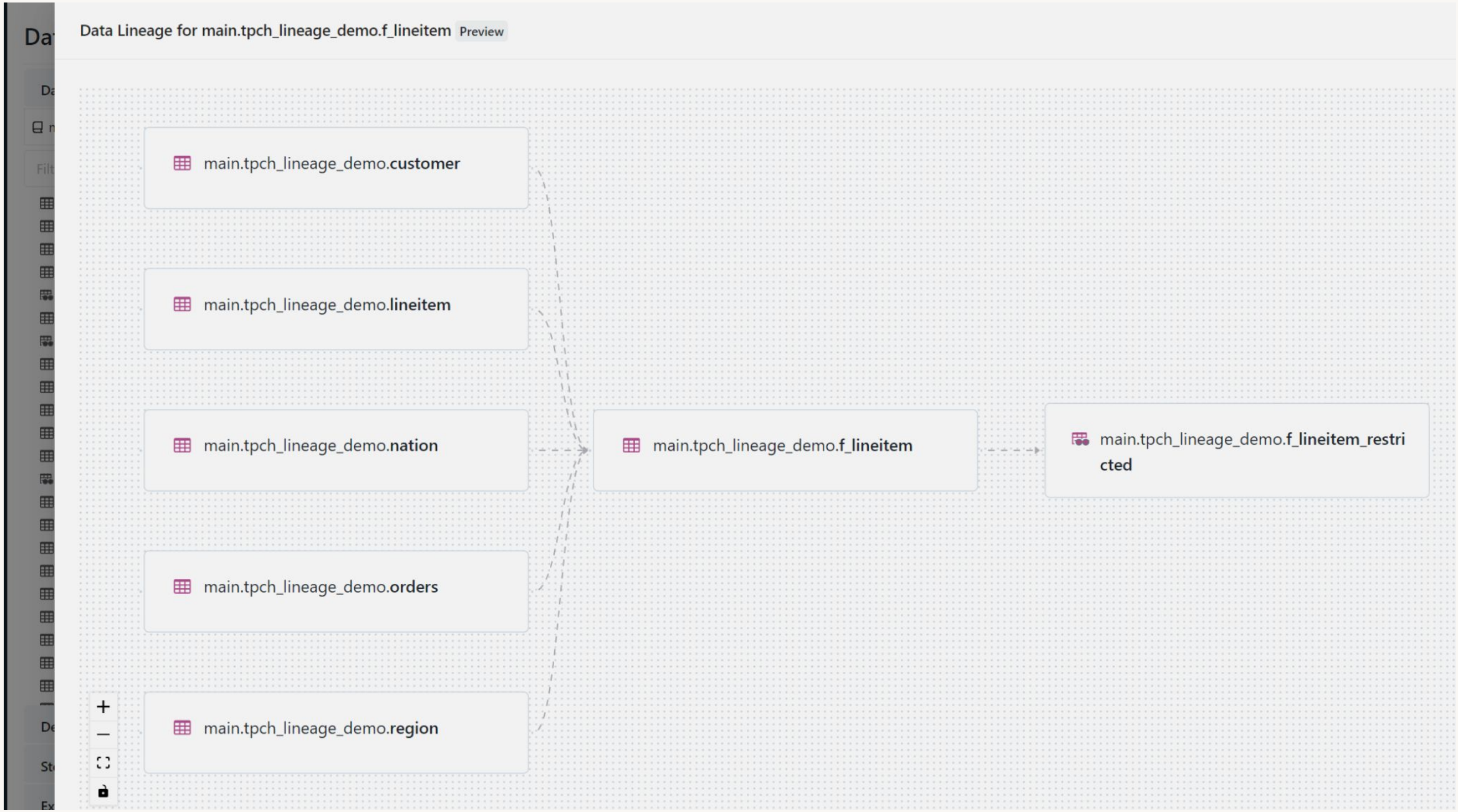
Control plane

Data plane

Unity Catalog







Data and cloud storage

Data governance and catalog partners

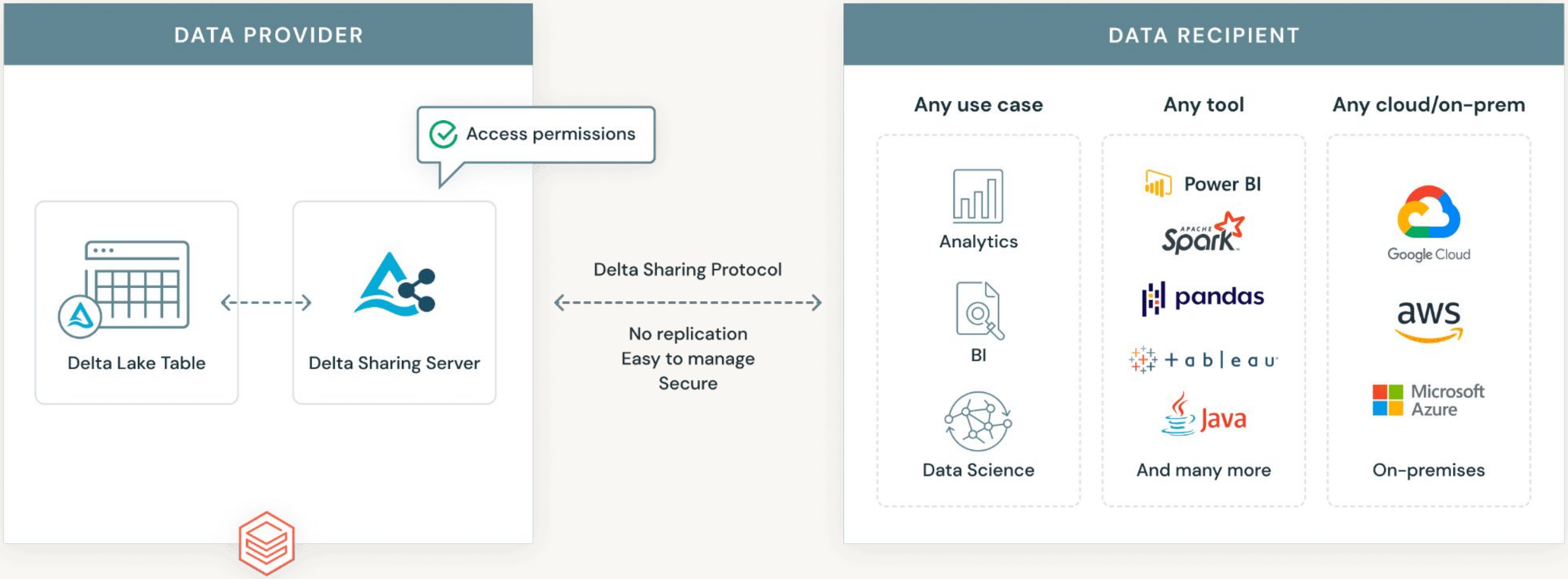
Data sharing with Delta Sharing

	Proprietary vendor solutions	SFTP	Cloud object store
Secure	✓	✓	✓
Cheap		✓	✓
Vendor agnostic		✓	
Multicloud		✓	
Open source		✓	
Table/DataFrame abstraction	✓		
Live data	✓		
Predicate pushdown	✓		
Object store bandwidth			✓
Zero compute cost			✓
Scalability			✓

	Proprietary vendor solutions	SFTP	Cloud object store	Delta Sharing
Secure	✓	✓	✓	✓
Cheap		✓	✓	✓
Vendor agnostic		✓		✓
Multicloud		✓		✓
Open source		✓		✓
Table/DataFrame abstraction	✓			✓
Live data	✓			✓
Predicate pushdown	✓			✓
Object store bandwidth			✓	✓
Zero compute cost			✓	✓
Scalability			✓	✓

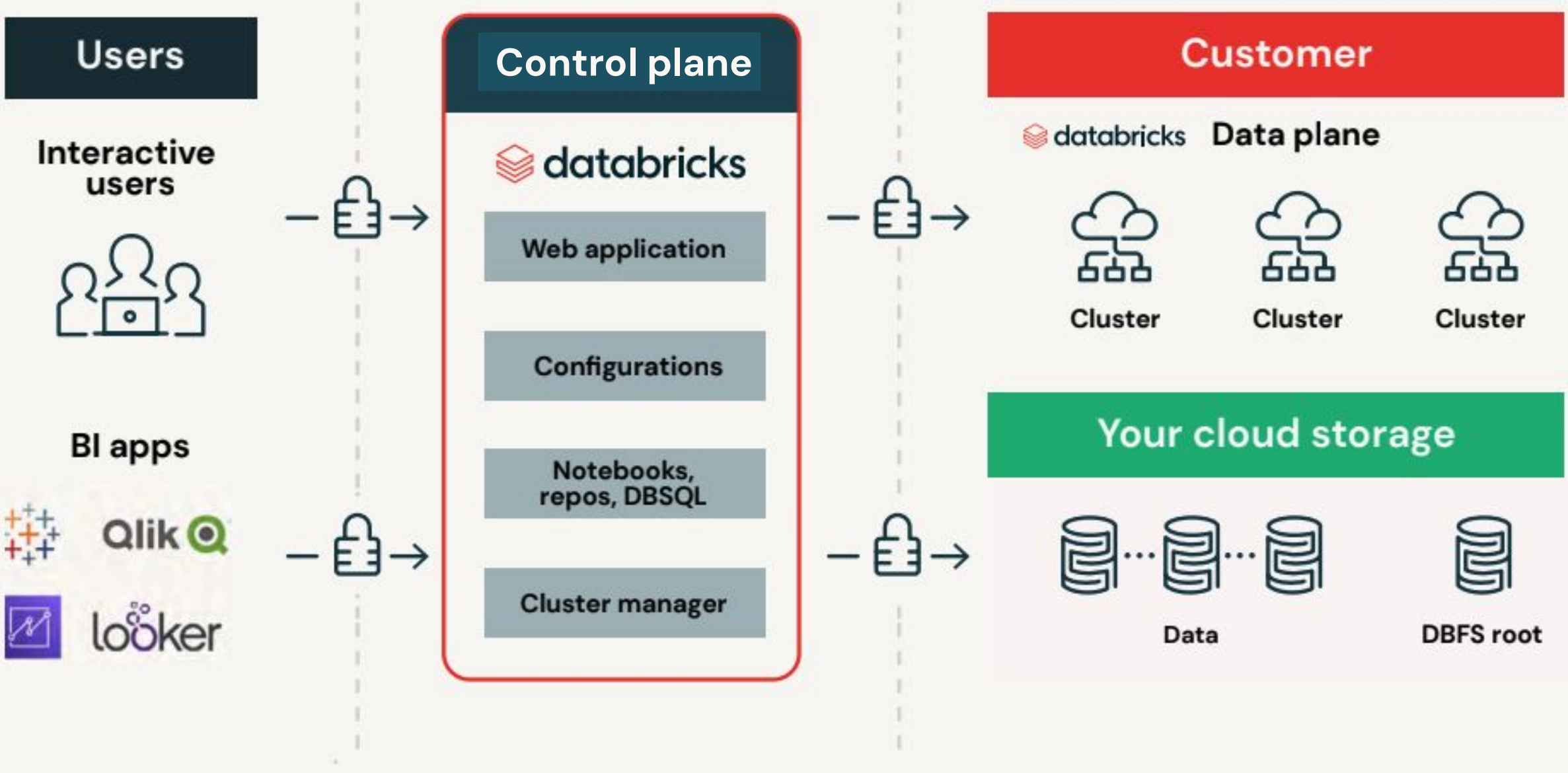
Benefits of Delta Sharing

- Open cross-platform sharing
- Share live data without copying it
- Centralized administration and governance
- Marketplace for data products
- Privacy-safe data clean rooms



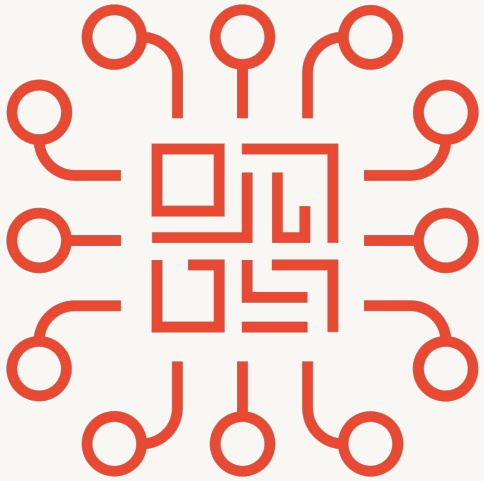
Divided security architecture

The control plane and the data plane



Security of the data plane

Networking



Servers

Databricks

databricks

User identity and access

- Table ACLs feature
- IAM instance profiles
- Securely stored access key
- The Secrets API

Data security

Databricks encryption capabilities are in place both at rest and in motion

For data-at-rest encryption:

- Control plane is encrypted
- Data plane supports local encryption
- Customers can use encrypted storage buckets
- Customers at some tiers can configure customer-managed keys for managed services

For data-in-motion encryption:

- Control plane <-> data plane is encrypted
- Offers optional intra-cluster encryption
- Customer code can be written to avoid unencrypted services (e.g., FTP)

Compliance

- SOC 2 Type II
- ISO 27001
- ISO 27017
- ISO 27018
- FedRAMP High
- HITRUST
- HIPAA
- PCI

GDPR and CCPA ready

databricks

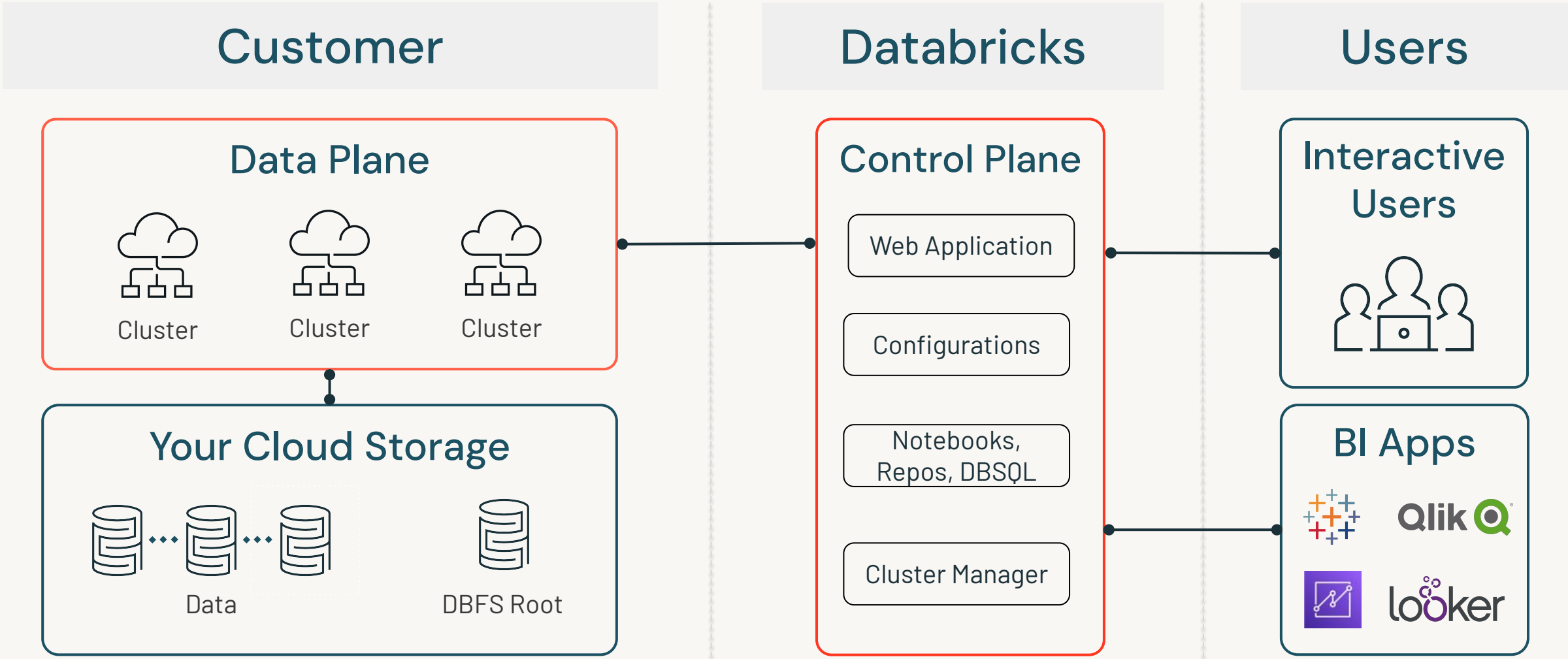
Databricks Lakehouse Platform Architecture and Security Fundamentals

Instant compute and serverless

Learning objectives

- Describe compute resources for your Databricks Lakehouse Platform.
- Define serverless compute.
- Explain the benefits of using Databricks Serverless SQL.

Classic data plane

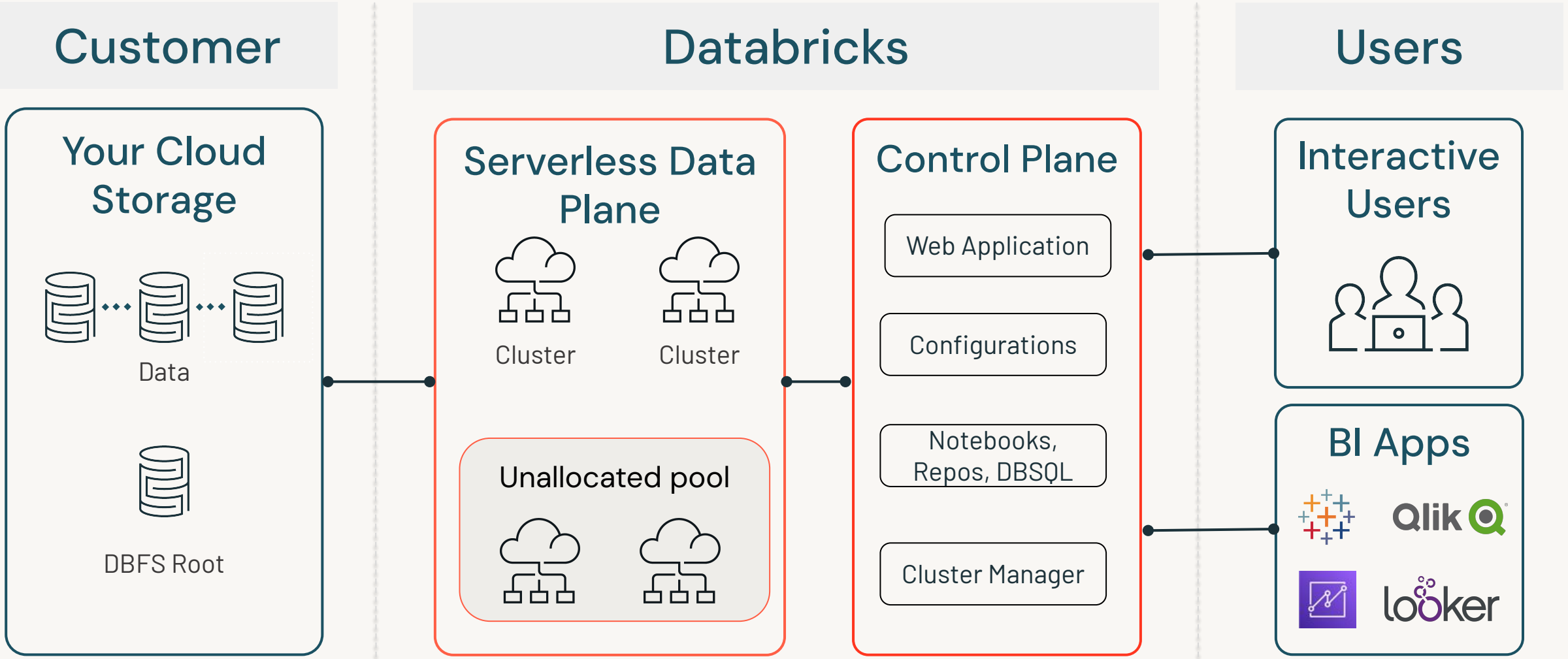


Compute resource challenges

- Cluster creation is complicated
- Environment startup is slow
- Business cloud account limitations and resource options

- Long running clusters
- Over provisioning of resources
- Higher resource costs
- High admin overhead
- Unproductive users

Serverless data plane



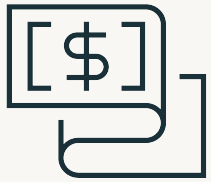
Users

Increase productivity
Instant query execution
Built-in connectors

Databricks
Serverless SQL

Admins

Reduce effort
Databricks optimally configures the cluster, manages updates to the VMs

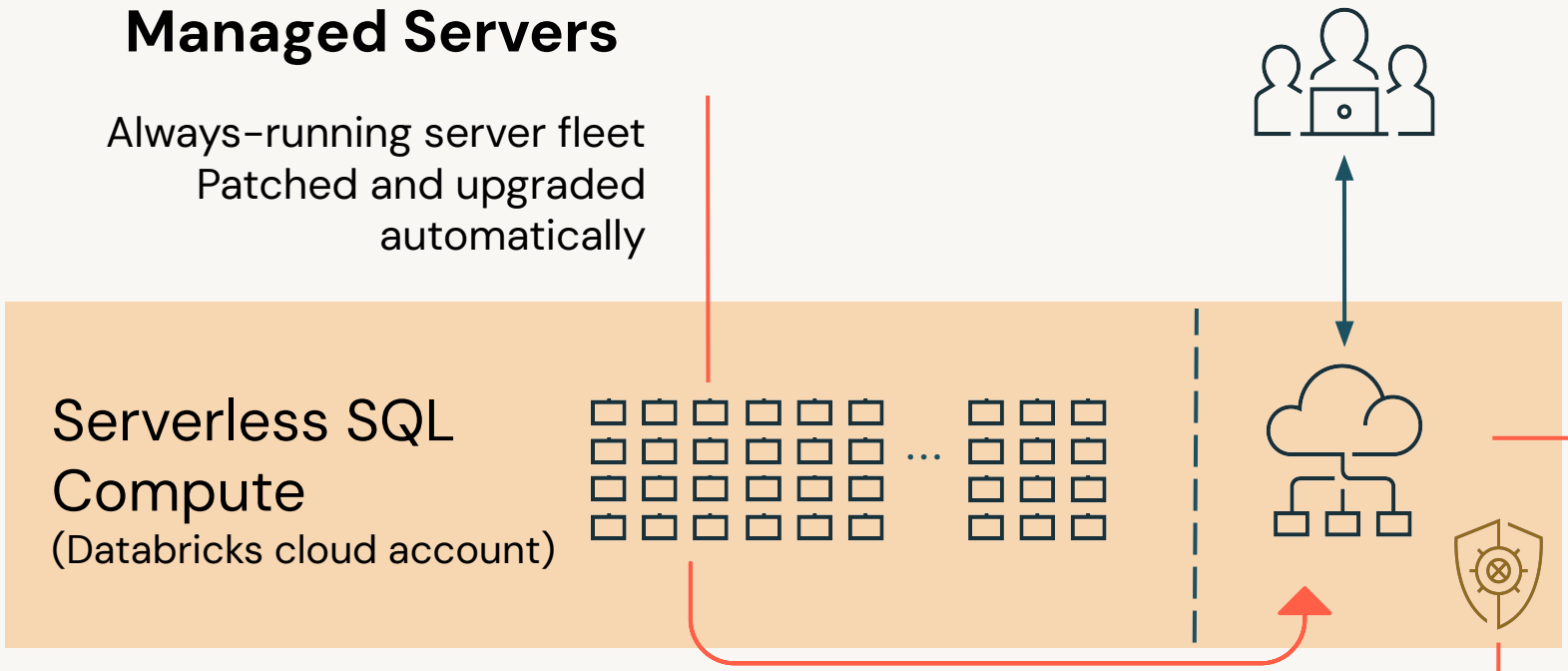


Finance

Lower cost
Reduce idle time
No over-provisioning

Managed Servers

Always-running server fleet
Patched and upgraded automatically



Elastic

Scale up and down automatically

Instant Compute

Allocation in seconds

Secure

Three layer isolation with data encryption

databricks

Databricks Lakehouse Platform Architecture and Security Fundamentals

Introduction to lakehouse data management terminology

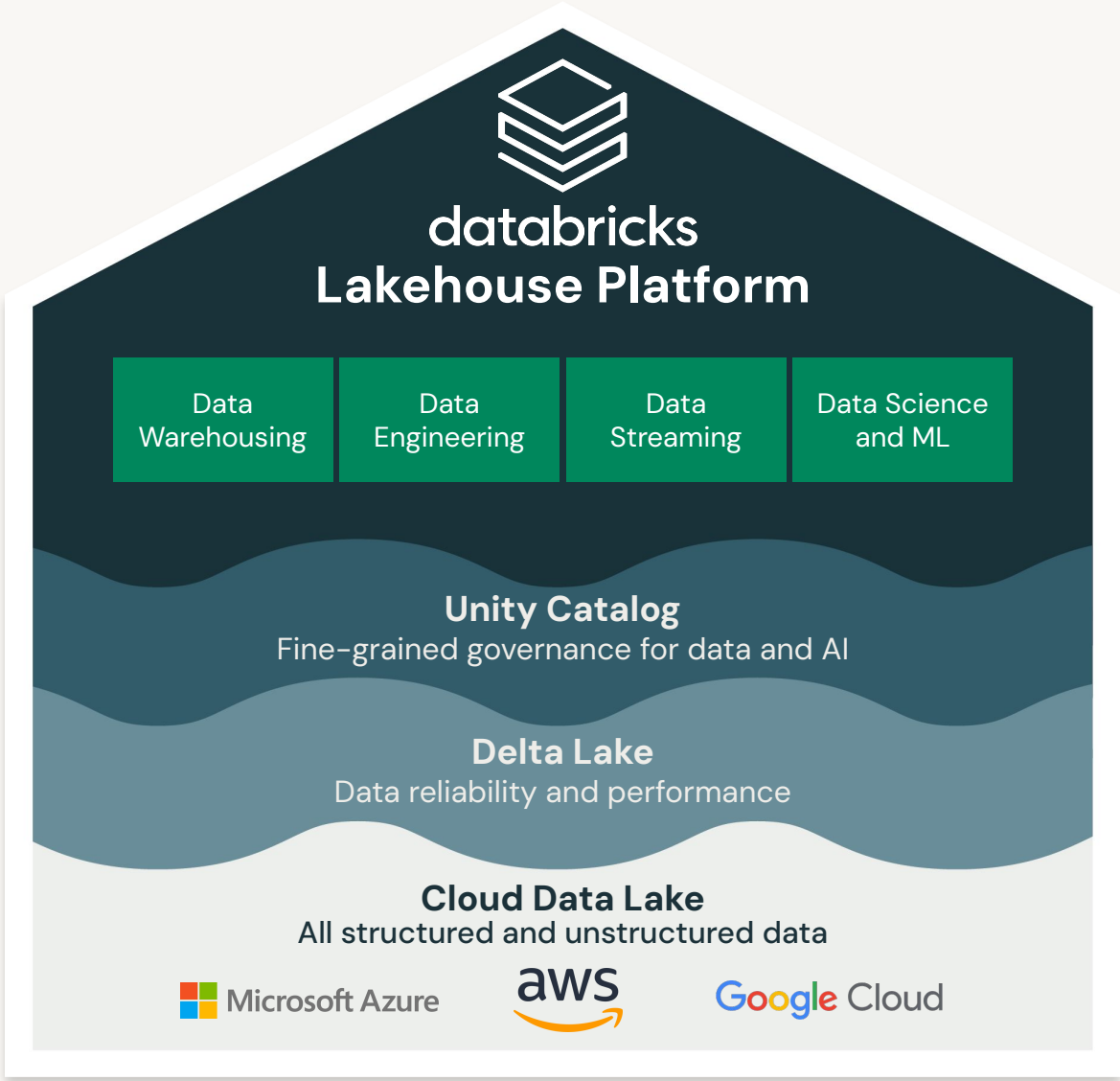
Learning objectives

- Define the terms metastore, catalog, schema, table, view, and function.
- Describe how these terms relate to data management in the Databricks Lakehouse Platform.

Databricks Lakehouse Platform

Unity Catalog

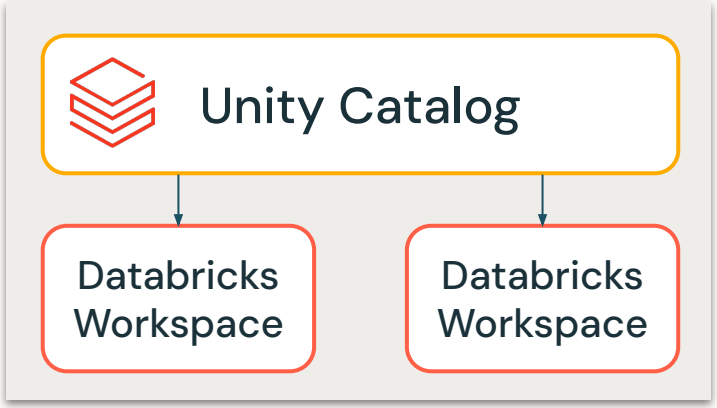
DELTA LAKE



Databricks Unity Catalog

Unified governance for all data and AI assets

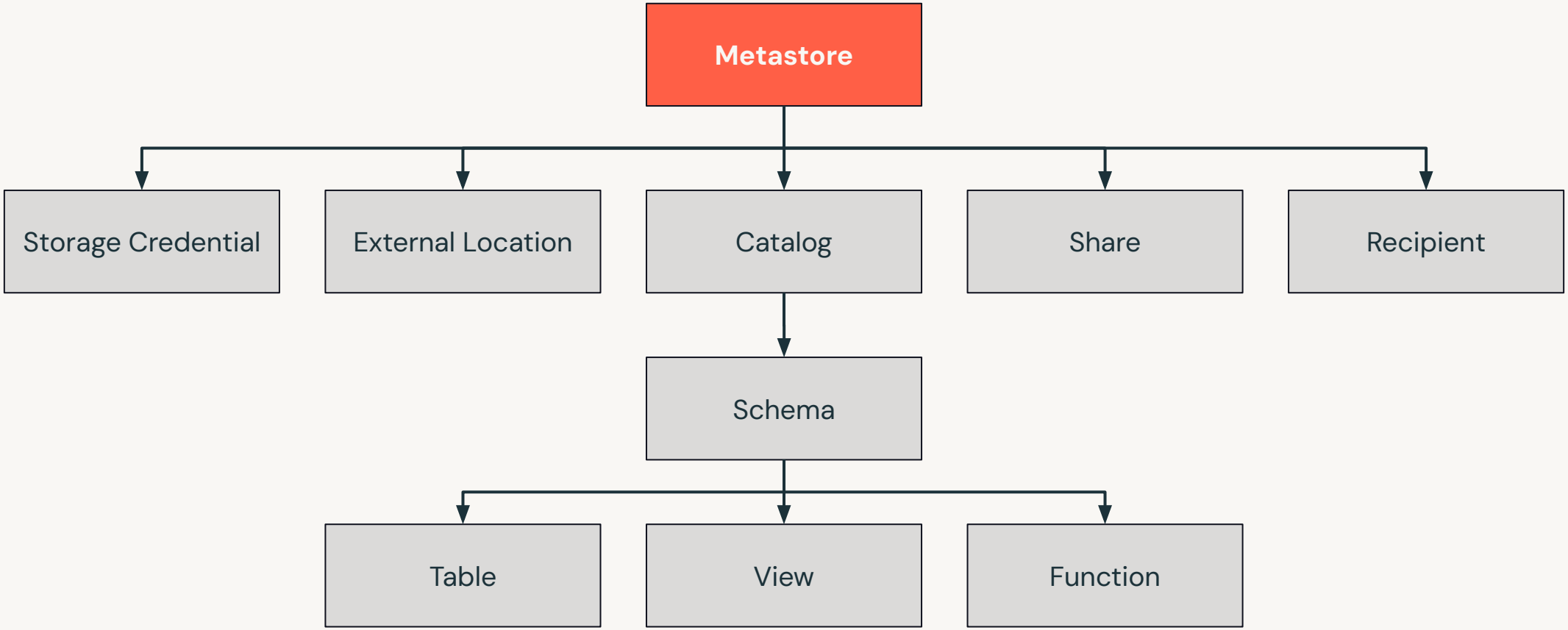
- Centralized governance for data and AI
- Built-in data search and discovery
- Performance and scale
- Automated lineage for all workloads
- Integrated with your existing tools



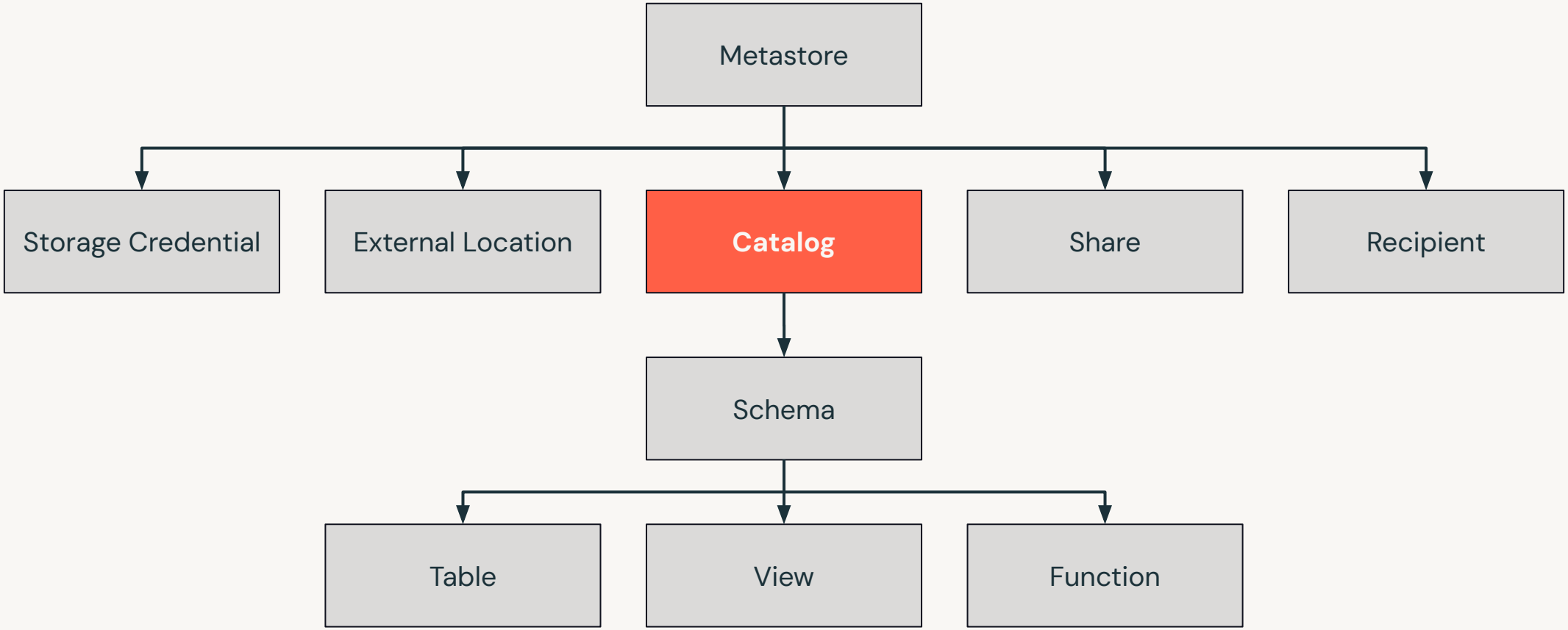

```
GRANT ... ON ... TO ...
REVOKE ... ON ... FROM ...
```

Catalogs, Schemas, Tables, Views, Storage credentials, External locations

Metastore

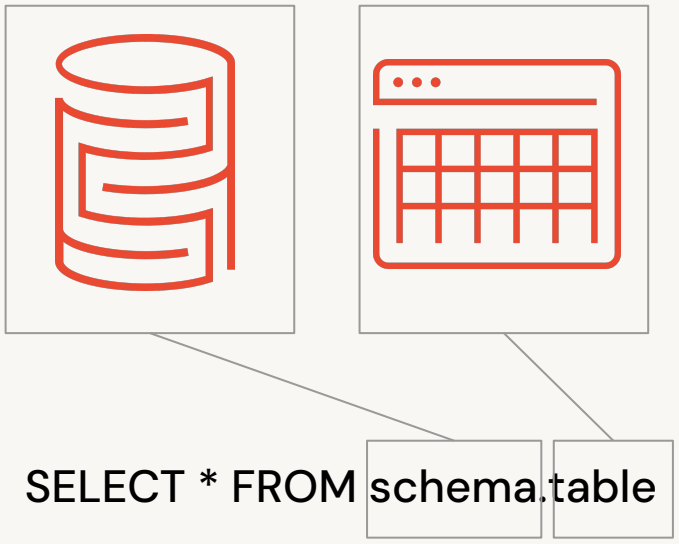


Catalog

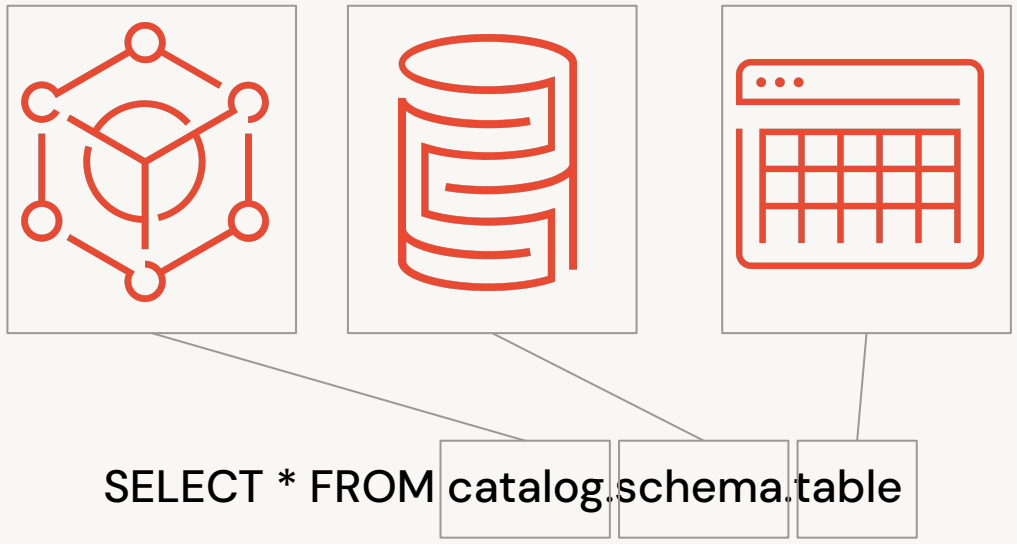


Three-level namespace

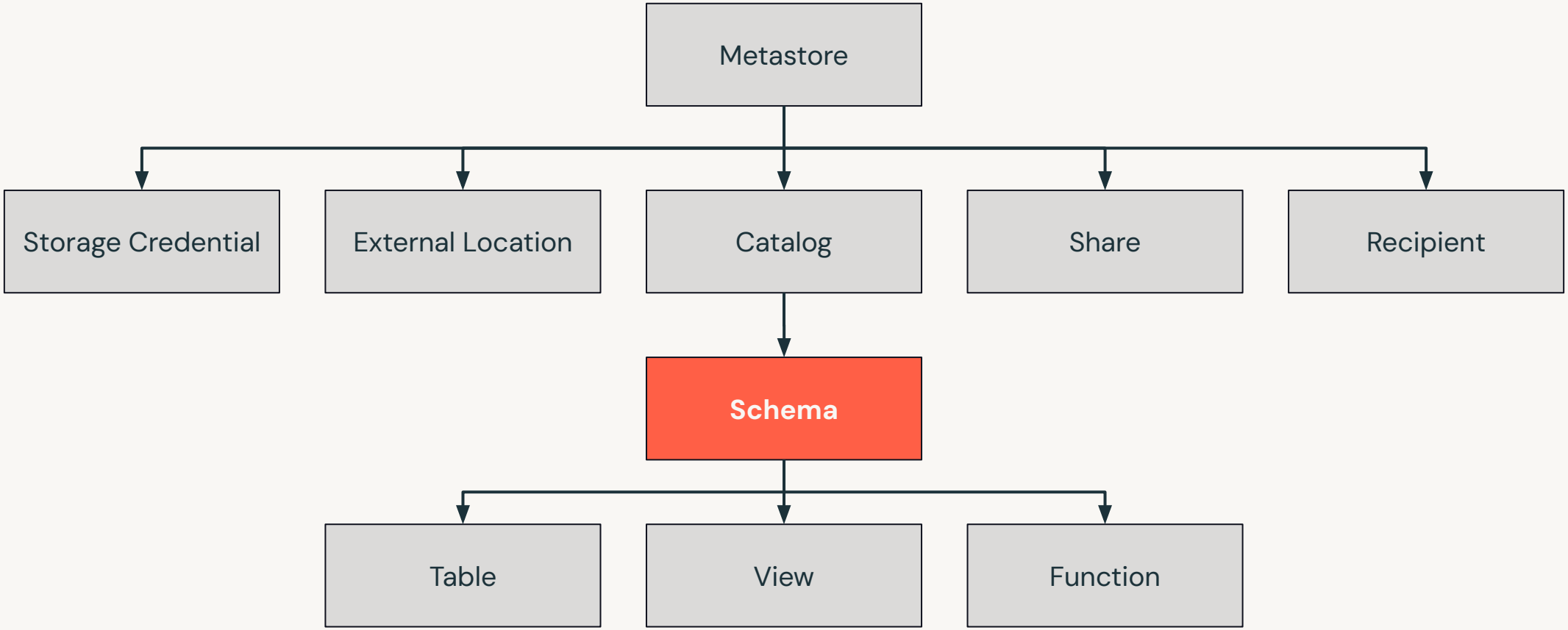
Traditional SQL two-level namespace



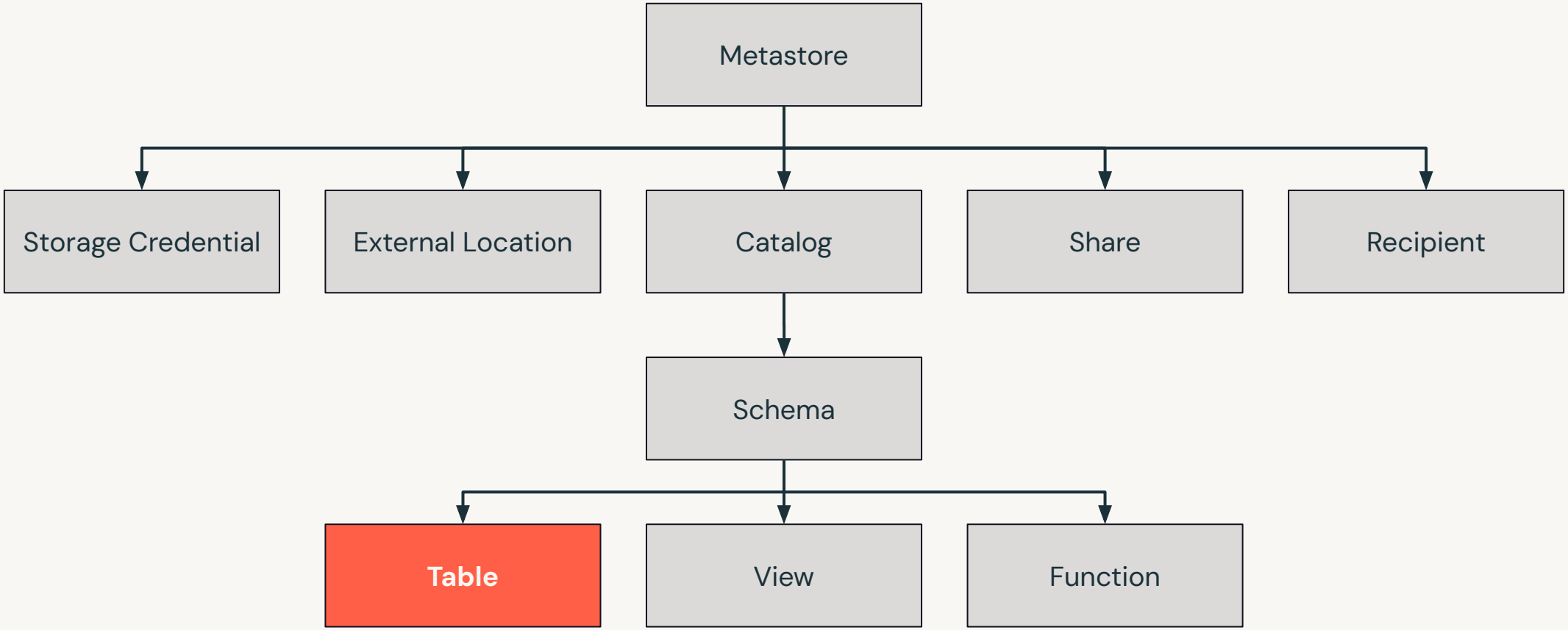
Unity Catalog three-level namespace



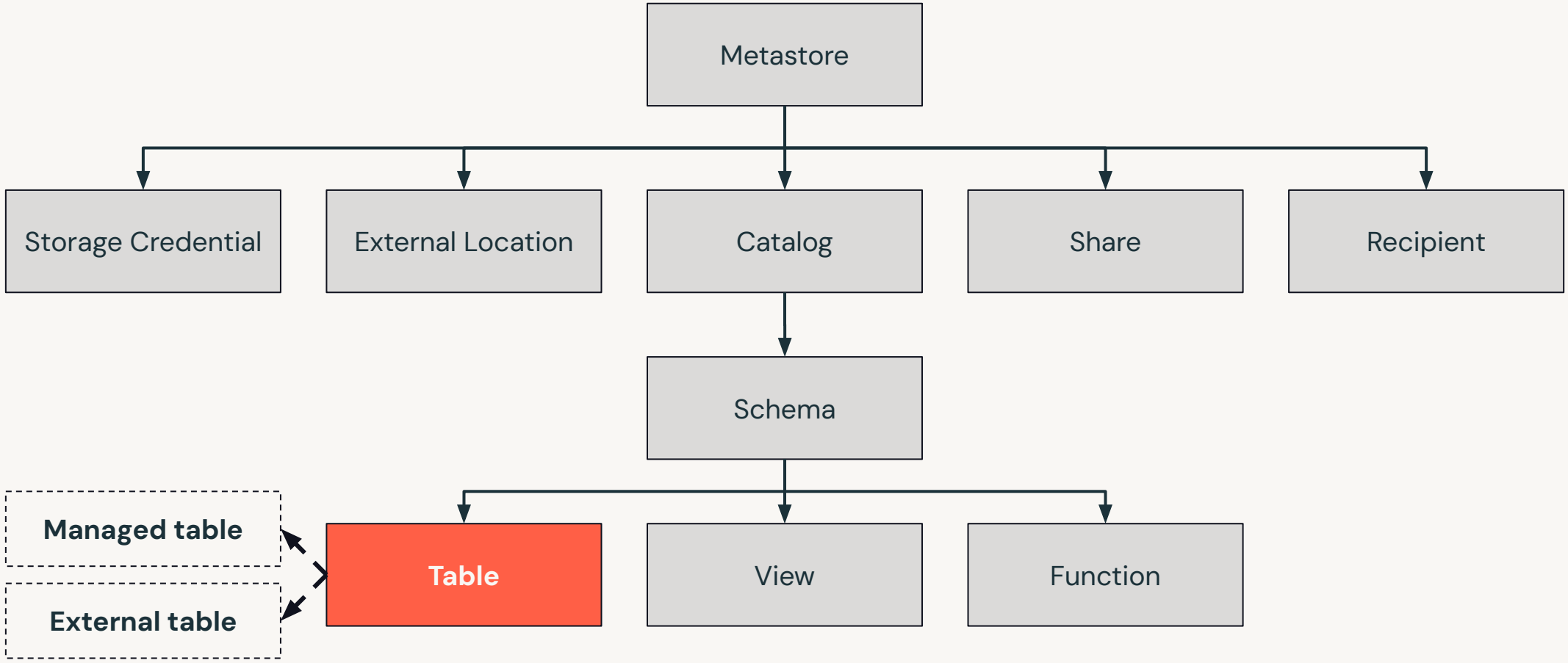
Schema (Database)



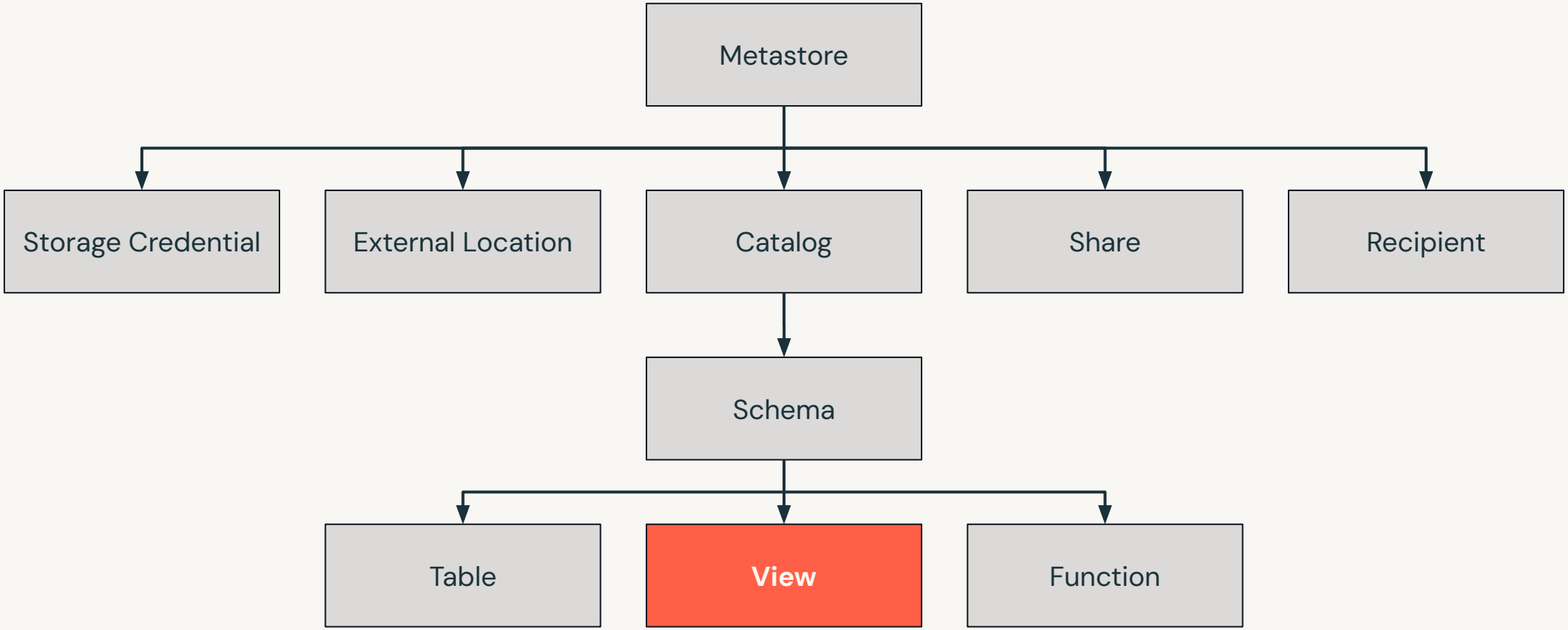
Table



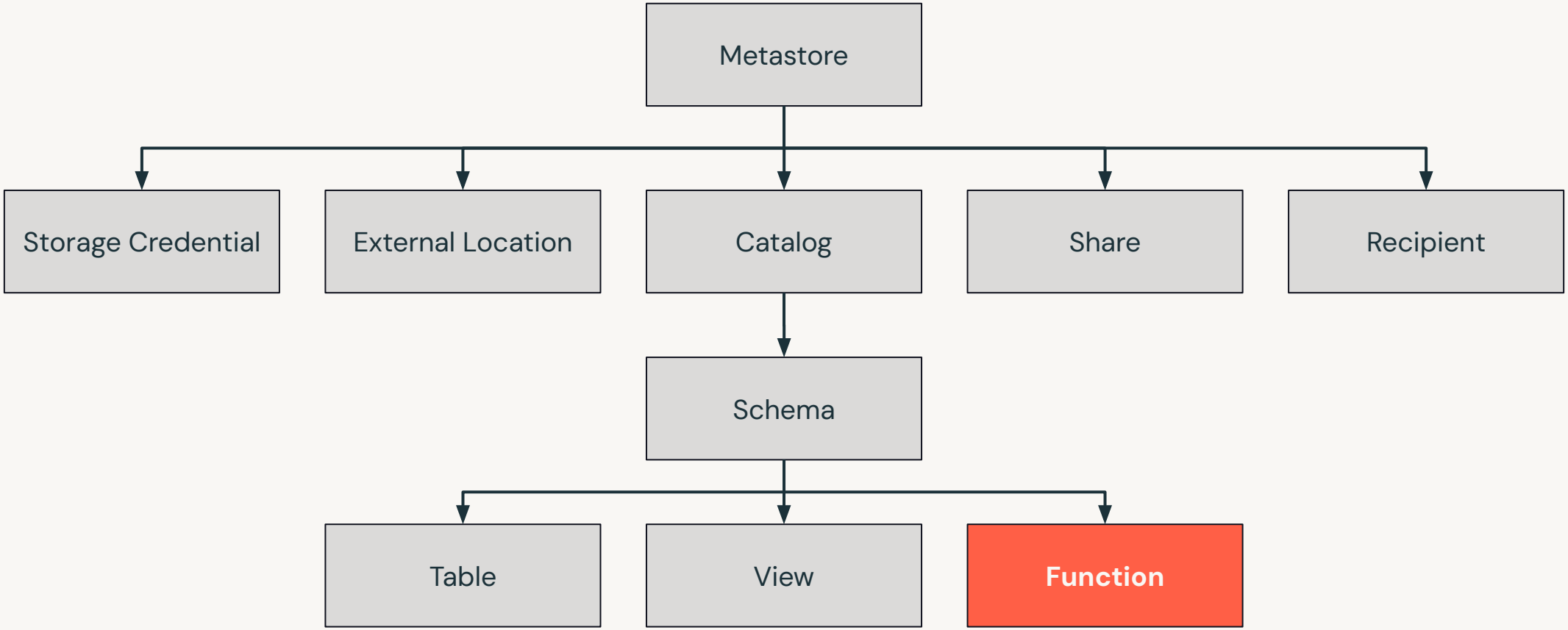
Managed and external tables



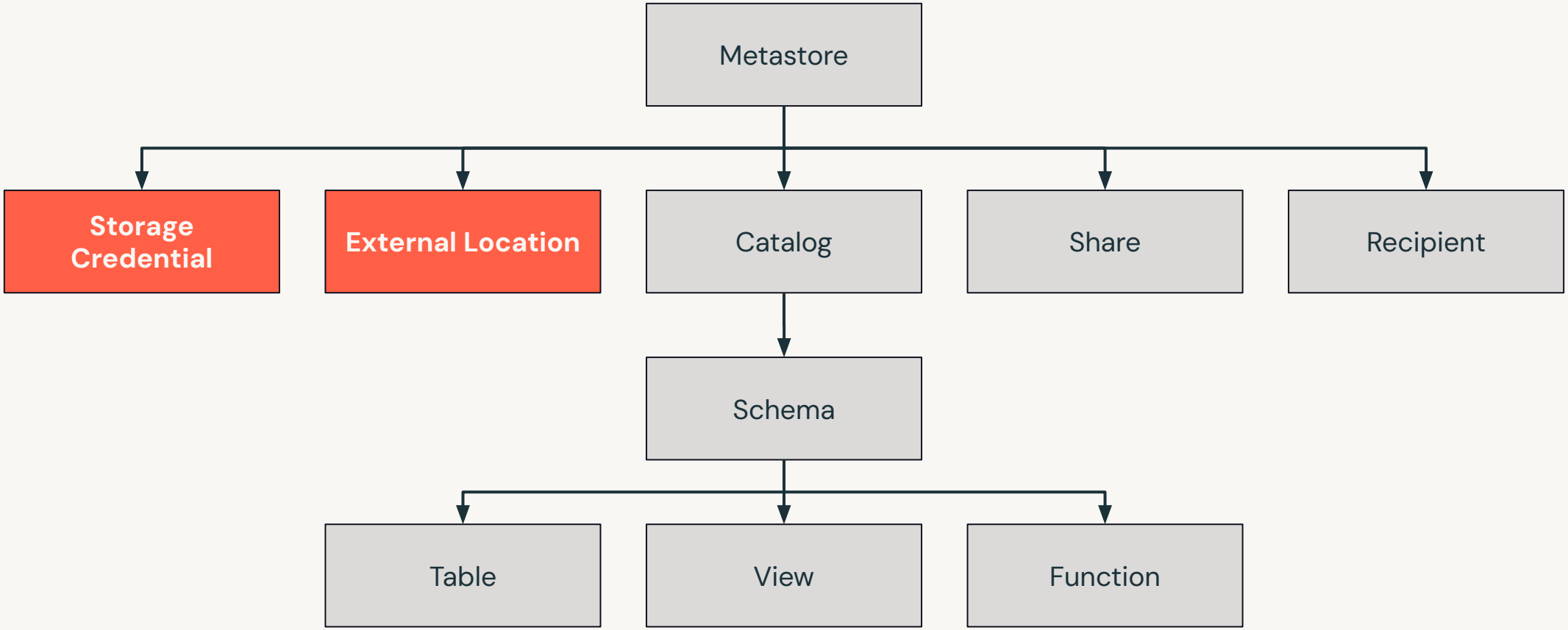
View



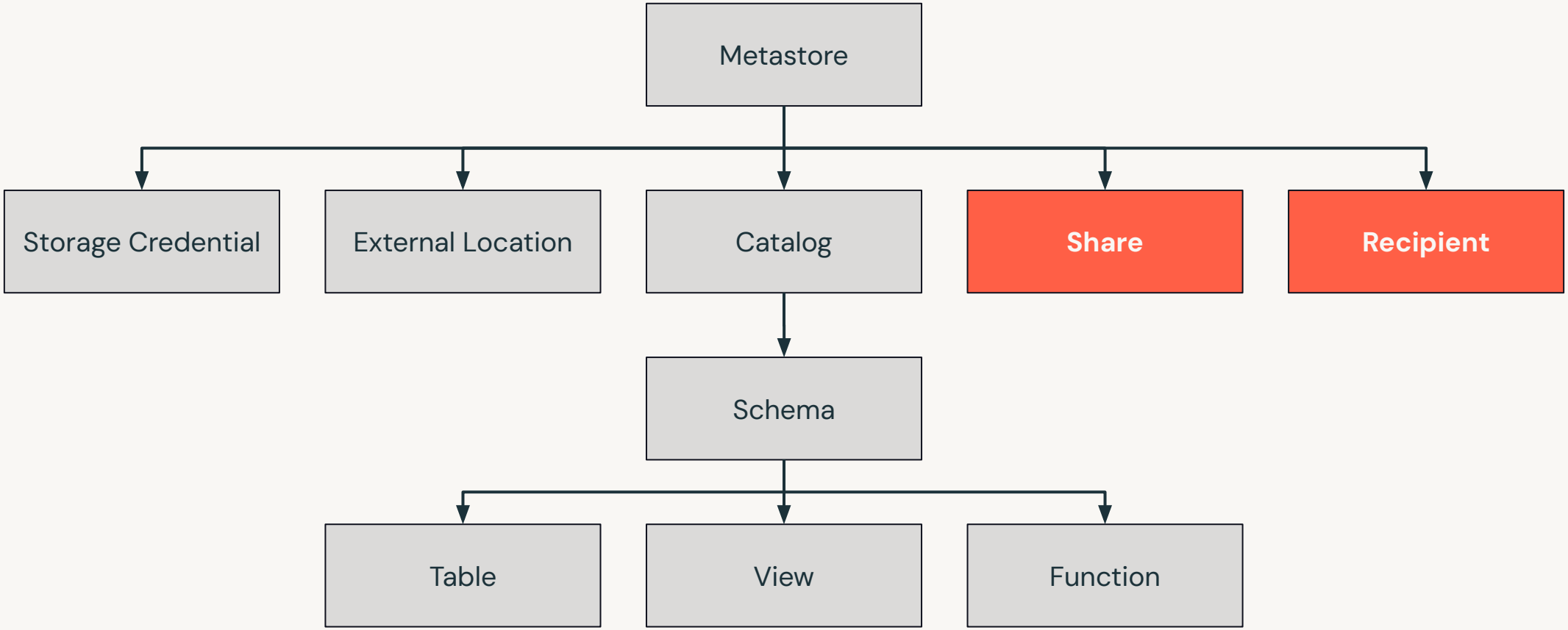
Function



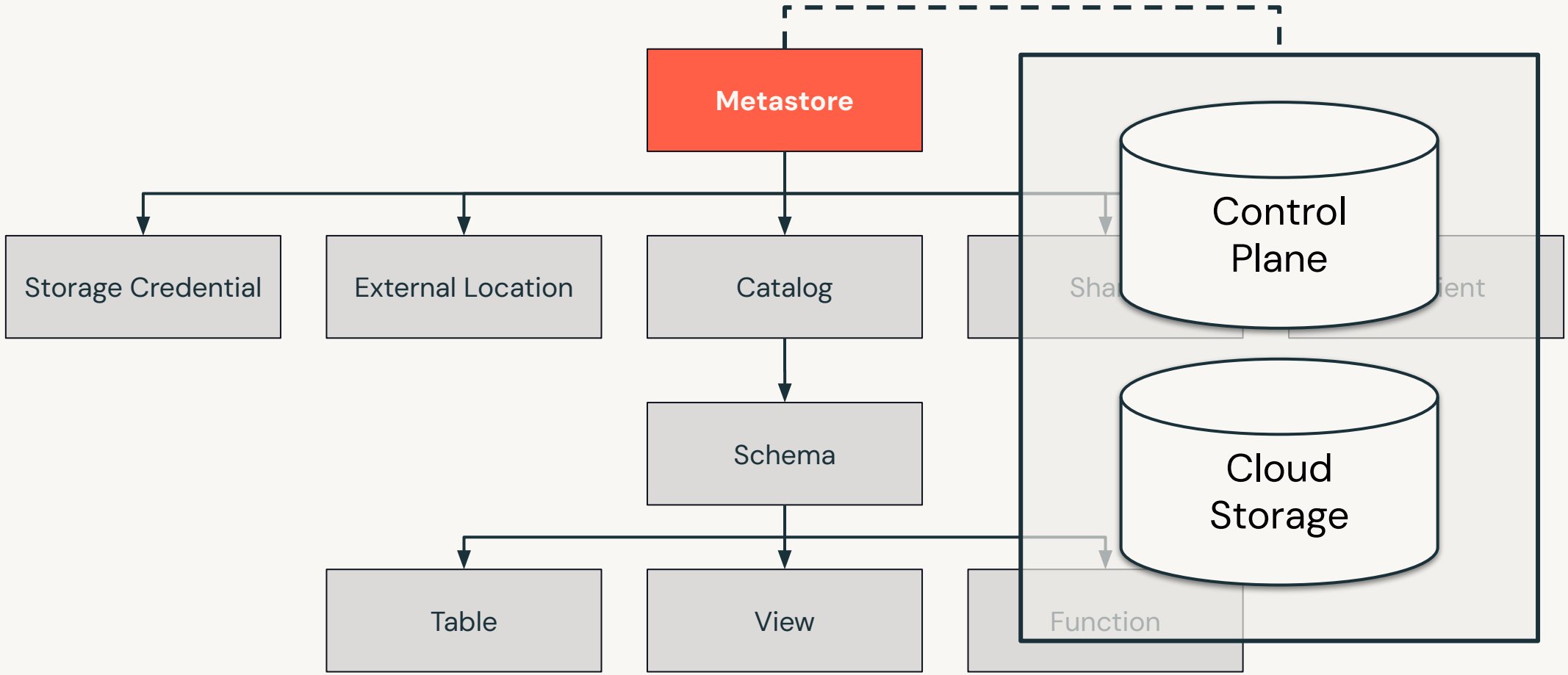
Storage credentials and External locations



Delta Sharing



Metastore data storage



databricks

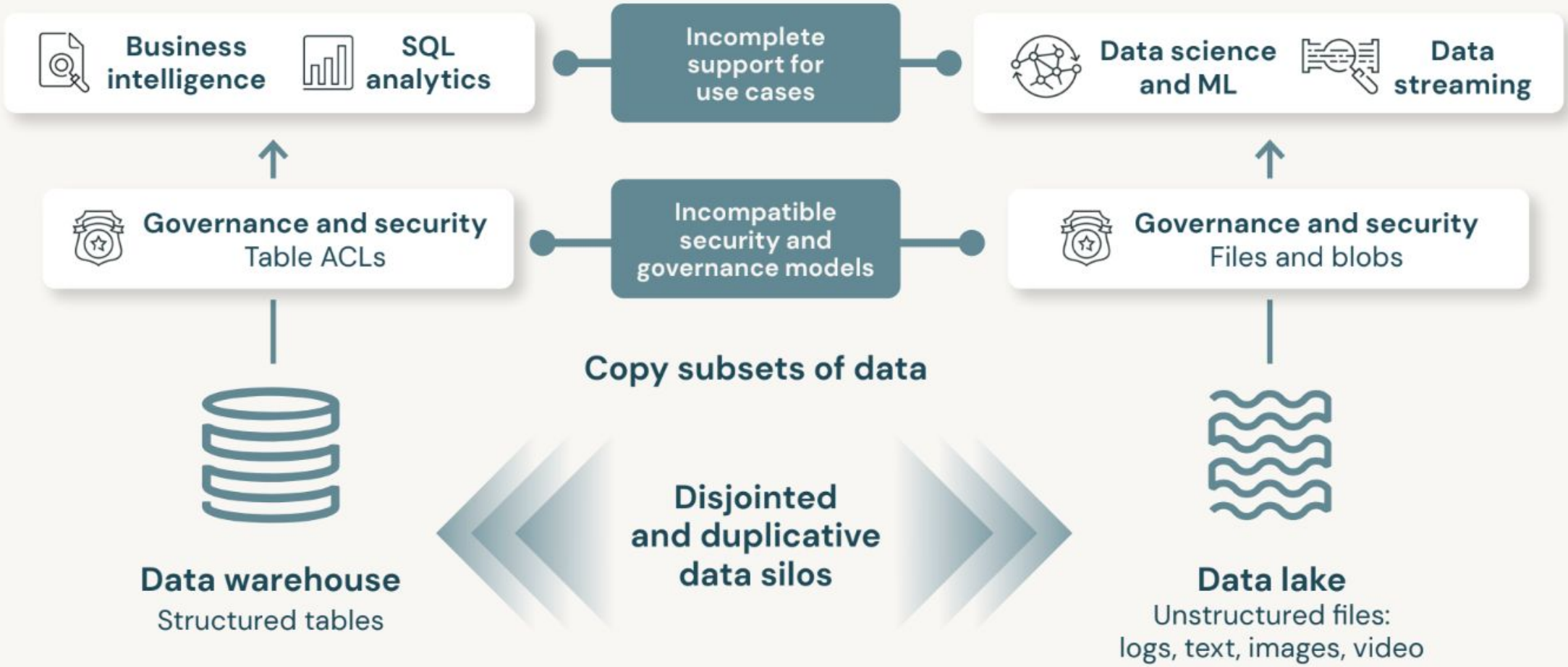
Supported Workloads on the Databricks Lakehouse Platform

Data warehousing

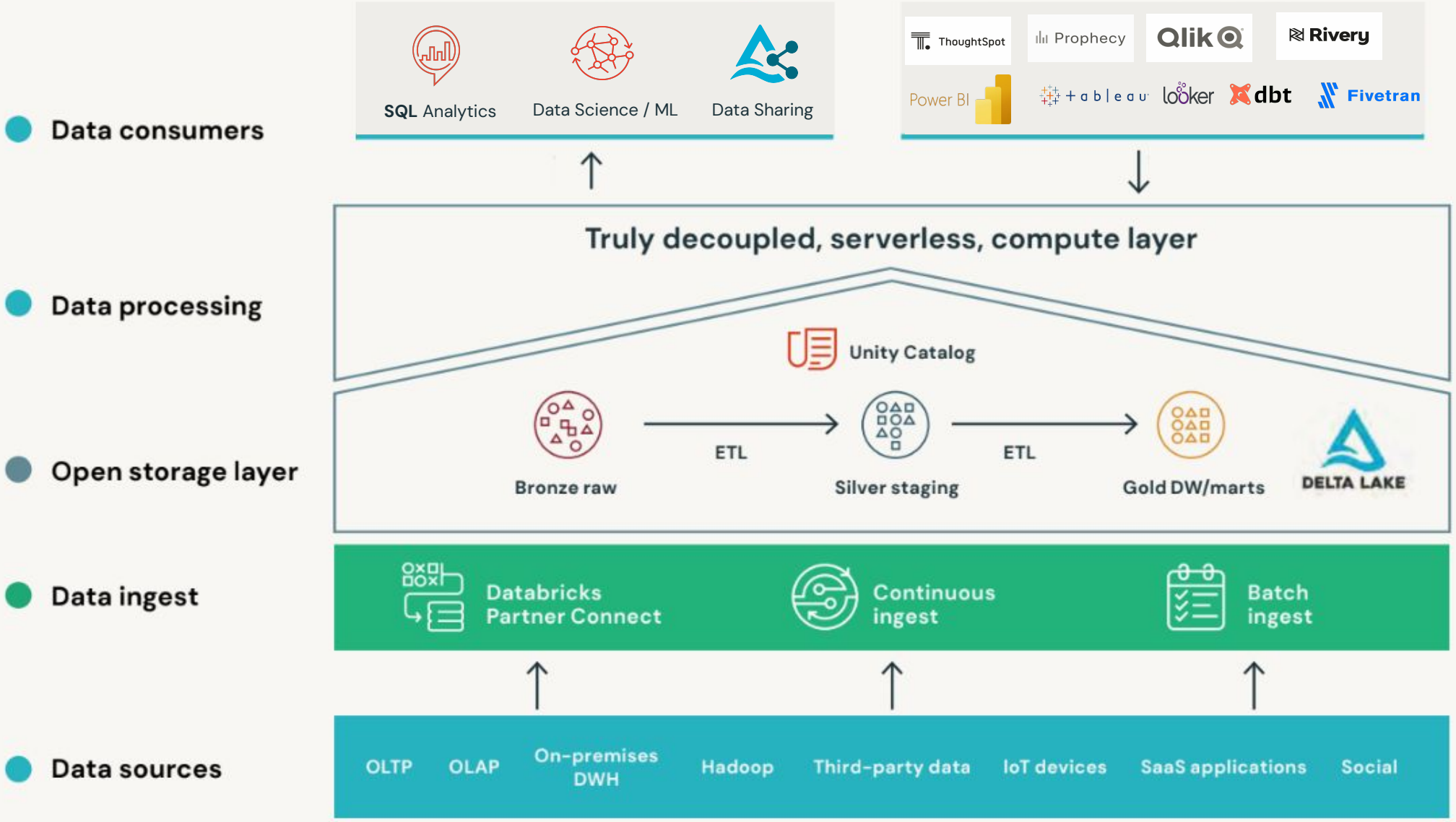
Learning objectives

- Recognize how the Databricks Lakehouse platform supports data warehousing with Databricks SQL.
- Describe the benefits of data warehousing with the Databricks Lakehouse Platform.

Two disparate, incompatible data platforms



Data warehousing on Databricks



Key benefits of data warehousing with the Databricks Lakehouse Platform

- Best price/ performance
- Built-in governance
- A rich ecosystem
- Break down silos

databricks

Supported Workloads on the Databricks Lakehouse Platform

Data engineering

Learning objectives

- Explain why data quality is important for data engineering.
- Describe how the Databricks Lakehouse Platform benefits the data engineer.
- Define what Delta Live Tables is.
- Explain how Databricks Workflows support data orchestration.

Data is a valuable business asset.

Challenges for the data engineering workload:

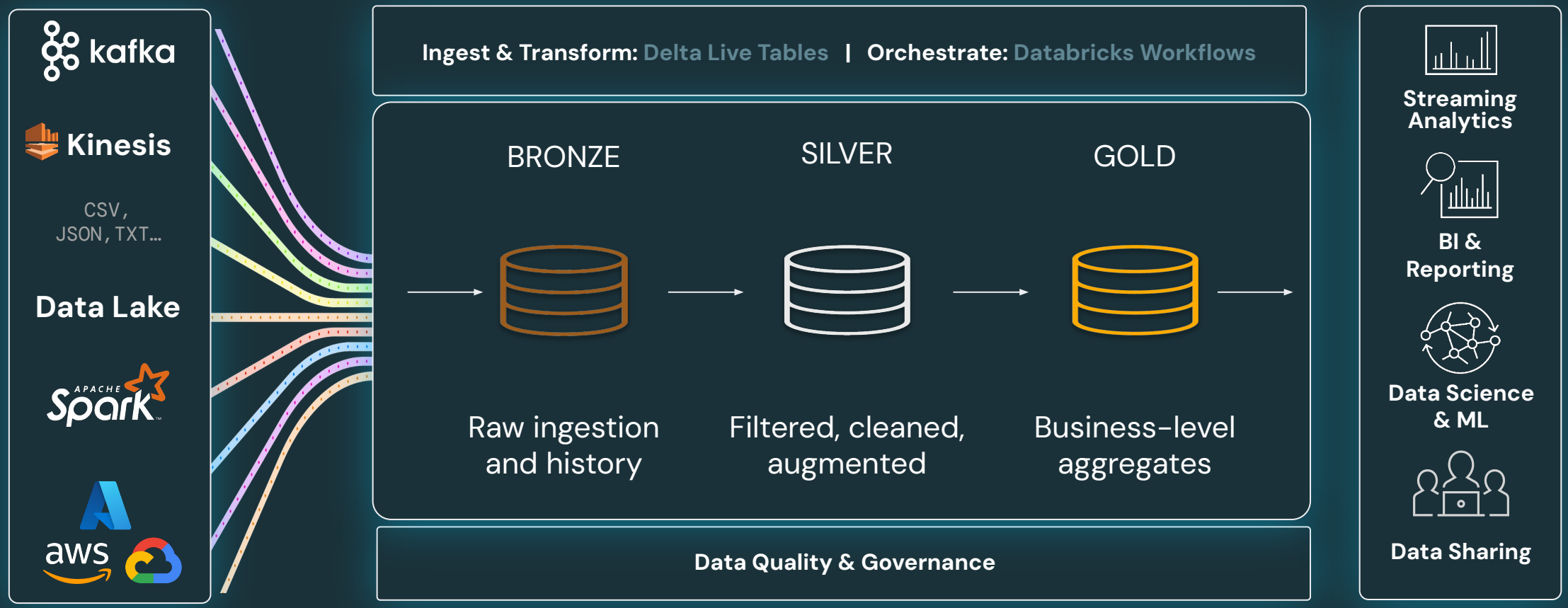
- Complex data ingestion methods
- Support for data engineering principles
- Third-party orchestration tools
- Pipeline and architecture performance tuning
- Inconsistencies between data warehouse and data lake providers

Key capabilities of data engineering on the lakehouse:

- Easy data ingestion
- Automated ETL pipelines
- Data quality checks
- Batch and streaming tuning
- Automatic recovery
- Data pipeline observability
- Simplified operations
- Scheduling and orchestration

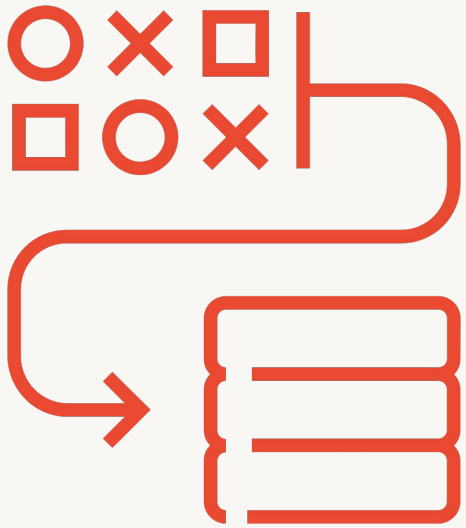
Data Engineering on the Lakehouse

Reliable data, analytics and AI

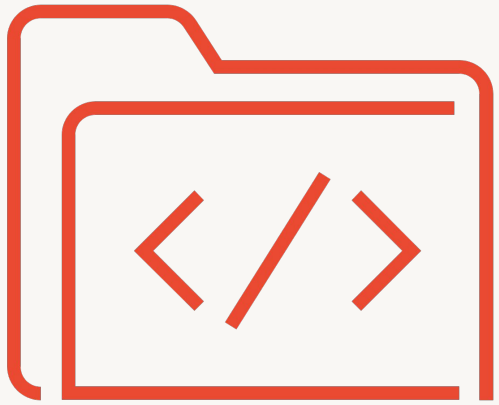


Data ingestion

Auto Loader



COPY INTO



Data transformation

```
CREATE LIVE TABLE raw_data as SELECT * FROM json.`…`
CREATE LIVE TABLE clean_data as SELECT … FROM LIVE.raw_data
```

Repos

Unity Catalog*

Databricks Workflows

Delta Live Tables

Full refresh

Dependency management

Expectations

APACHE Spark™

DELTA LAKE

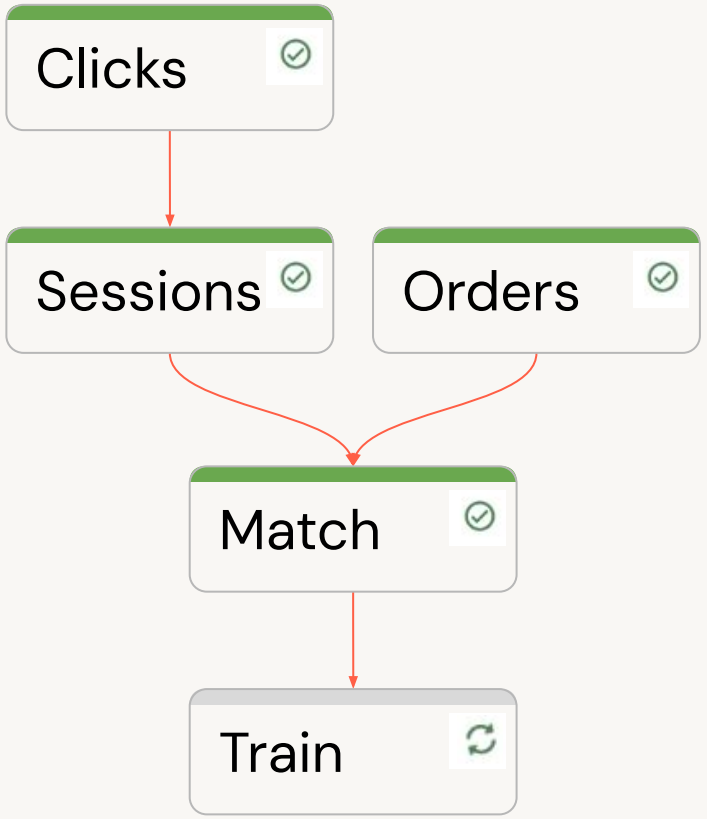
Incremental computation*

Checkpointing and retries

*Coming soon

Databricks Workflows

Orchestrate... **...any task...** **...across any platform**



Delta Live Tables

Koalas

Auto Loader

mlflow

Java

Spark

SQL

non-Spark

dbt

+more

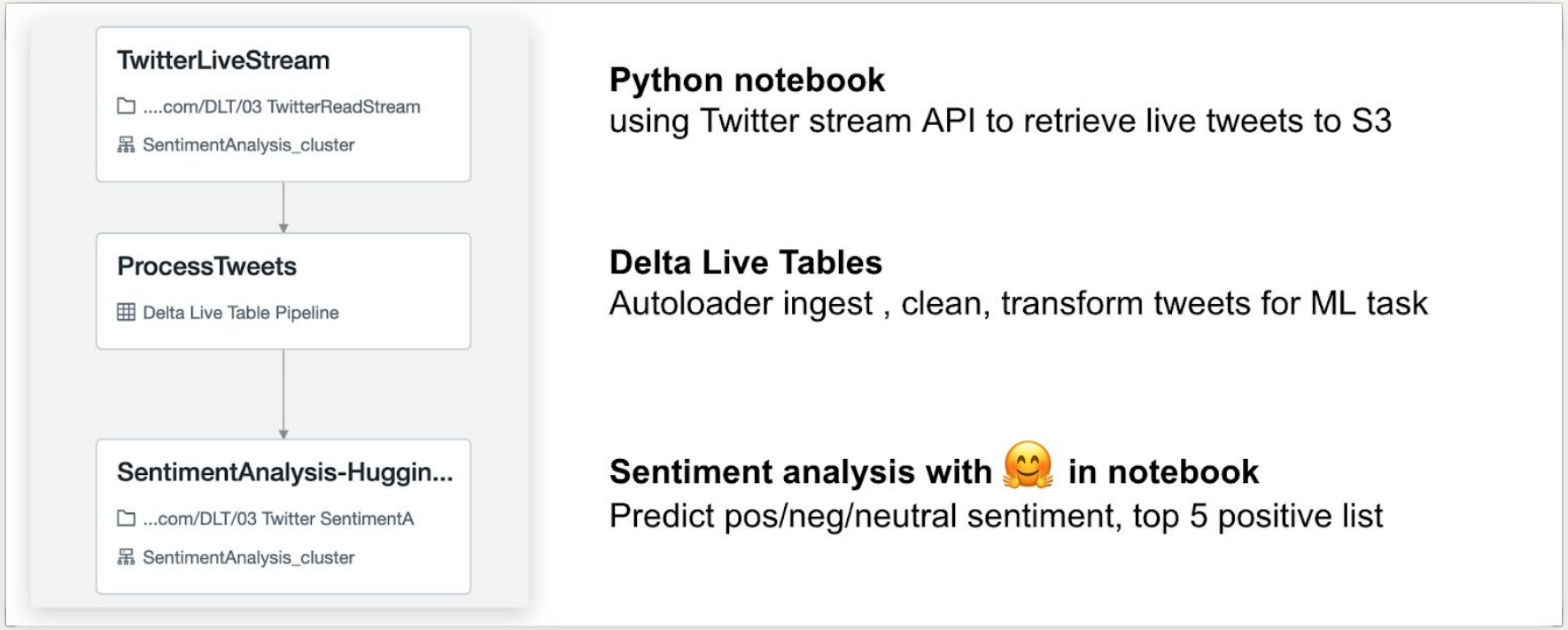
Google Cloud

talend

Fivetran

+more

Orchestrate anything



databricks

Supported Workloads on the Databricks Lakehouse Platform

Data streaming

Learning objectives

- Explain what streaming data is.
- Explain how the Databricks Lakehouse Platform supports data streaming.

New opportunities from real-time data

Every organization generates vast amounts of real-time data

Transactional records
Point of sale (POS)
Banking transactions
Airline reservations
Call center records

Third-party
News feeds
Weather
Market data
Real-time traffic

Interactions
Web clicks
Social posts
Emails
Instant messages

IoT events
Sensors
Geolocation
Machine logs
Mobile devices

Creating opportunities for new kinds of real-time applications

Fraud detection

Personalized offer

Vaccine distribution

Smart pricing

In-game analytics

Connected cars and smart devices

Predictive maintenance

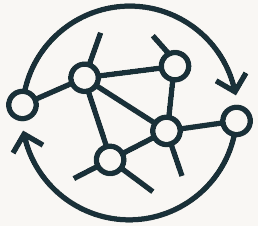
Content recommendations

Data Streaming Use Cases

Real-Time Analytics

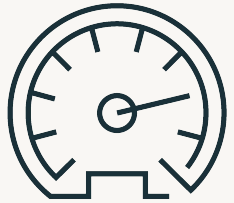
Analyze streaming data for instant insights and faster decisions.

Real-Time Machine Learning



Train models on the freshest data. Score in real-time.

Real-Time Applications



Embed automatic and real-time actions into business applications.

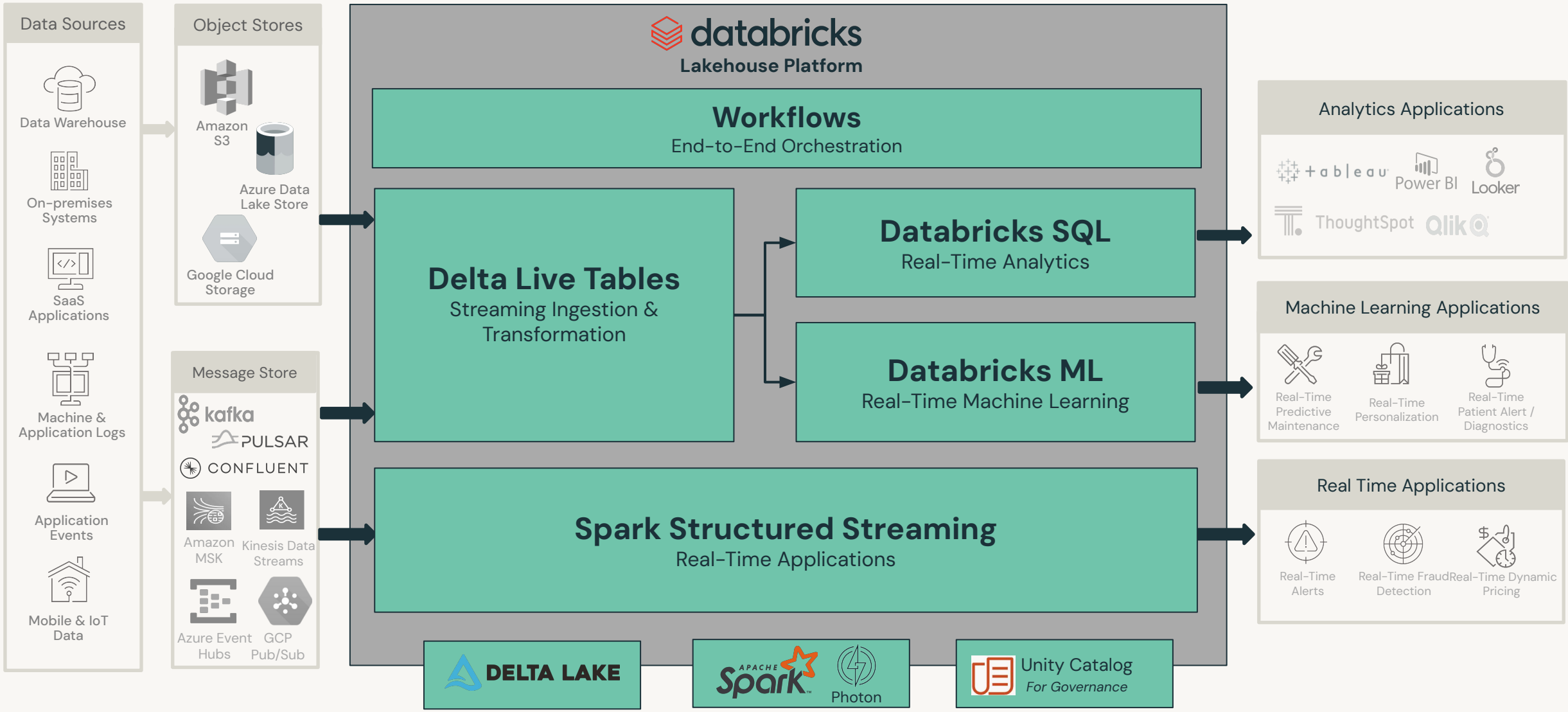
Industry Specific Use Cases

- Retail
- Industrial automation
- Healthcare
- Financial Institutions
- and many more!

Top 3 differentiating capabilities for data streaming on the lakehouse

1. Build streaming pipelines and applications faster
2. Simplify operations with automation
3. Unified governance for real time and historical data

Reference architecture for streaming use cases



databricks

Supported Workloads on the Databricks Lakehouse Platform

Data science and machine learning

Learning objectives

- Explain the challenges of harnessing machine learning and AI.
- Describe how the Databricks Lakehouse Platform supports the data science and machine learning workload.

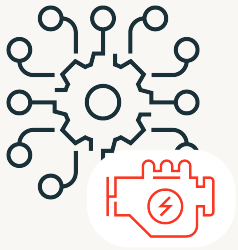
Challenges to successful machine learning and AI endeavors

- Siloed and disparate data systems
- Complex experimentation environments
- Getting models to production

- Multiple tools available
- Experiments are hard to track
- Reproducing results is difficult
- ML is hard to deploy

Compute Platform

Any ML workload optimized and accelerated.



Databricks Machine Learning Runtime

- Optimized and preconfigured ML Frameworks
- Turnkey distributed ML
- Built-in AutoML
- GPU support out of the box

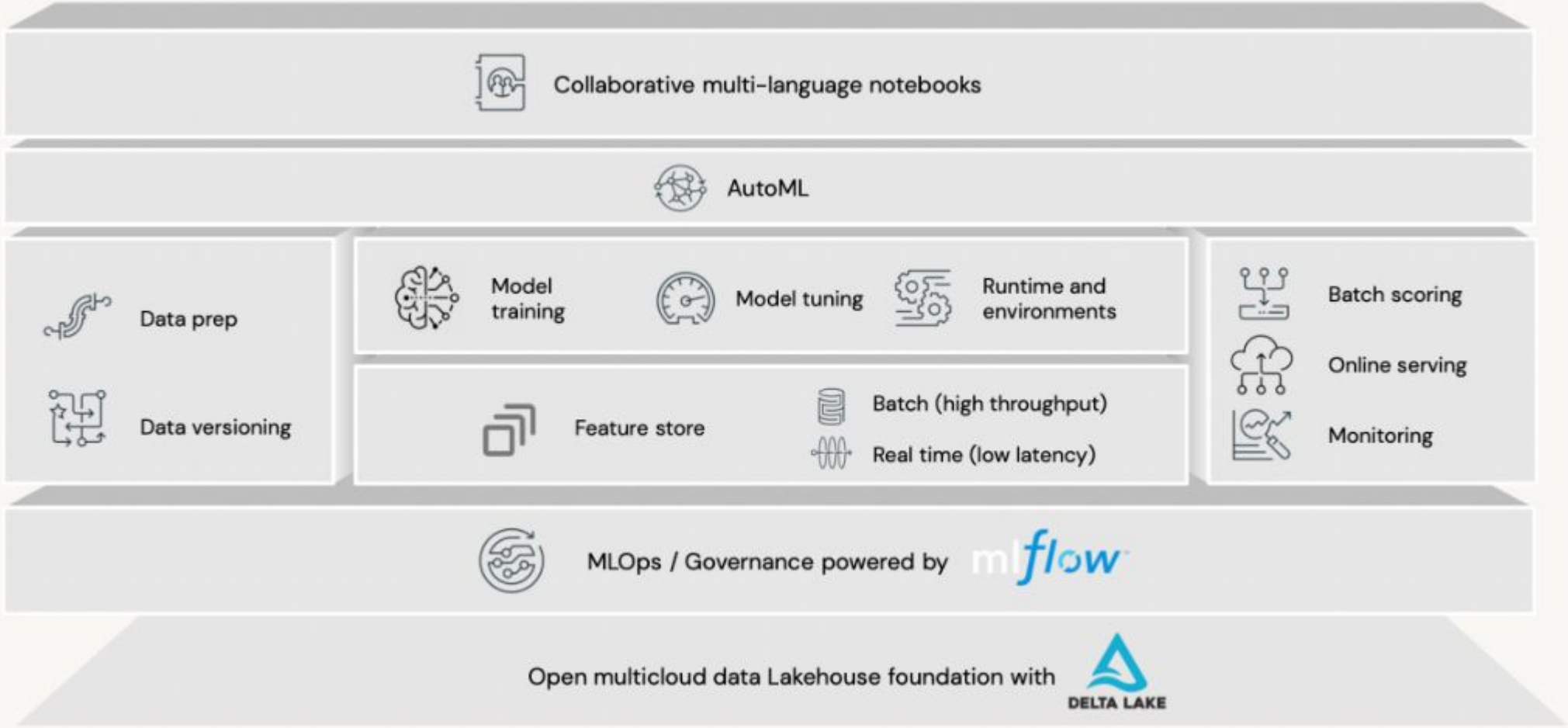
Built-in **ML Frameworks** and **model explainability**

Built-in support for **distributed training**

Built-in support for **AutoML** and **hyperparameter tuning**

Built-in support for **hardware accelerators**

Databricks Machine Learning



Experiments > Configure AutoML experiment

Configure AutoML experiment

Provide feedback

1 Configure — 2 Train — 3 Evaluate

AutoML Experiment Configuration

* Cluster (ML runtime 8.3 or above)

ML Runtime 10.4 LTS

* ML problem type

Classification

* Dataset

Browse | yxiong.titanic_train

* Prediction target

Survived

* Experiment name

Survived_titanic_train-2022_04_20-12_49

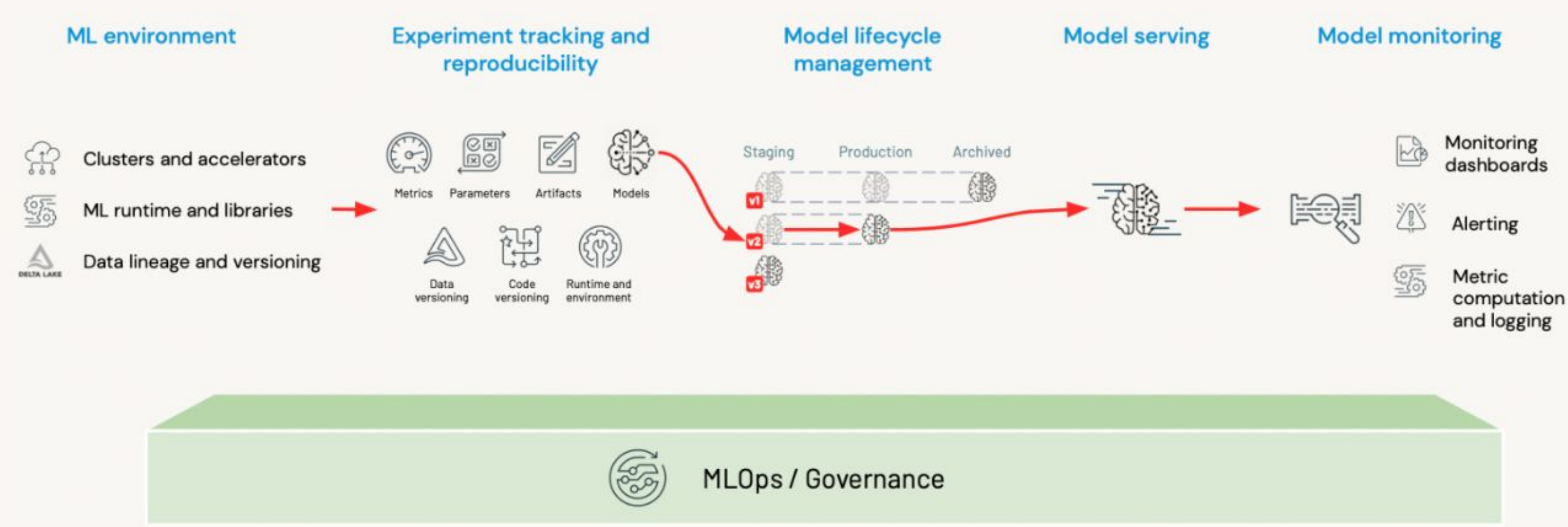
Advanced Configuration (optional)

Start AutoML

Schema:

Include	Column name	Data type	Impute with
☑	PassengerId	int	Auto
✓	Survived	int	Auto
☑	Pclass	int	Auto
☑	Name	string	Auto
☑	Sex	string	Auto
☑	Age	double	Auto
☑	SibSp	int	Auto
☑	Parch	int	Auto
☑	Ticket	string	Auto
☑	Fare	double	Auto
☑	Cabin	string	Auto
☑	Embarked	string	Auto

12 total columns.



Databricks Machine Learning