From SQL to Lakehouse

Unpacking DBT, Trino & Iceberg

19.10.2024

Agenda

Why Migrate from SQL Warehouses to a Lakehouse?

Lakehouse Architecture

- Why Parquet?
- Why Iceberg?
- Why Trino?
- Why DBT?

The Medallion Architecture

Code Walkthrough

- DBT repo setup & deployment
- How to read from different sources in DBT
- Sql with incremental update in DBT
- How to define a macro with DBT
- How to add description & tests in DBT

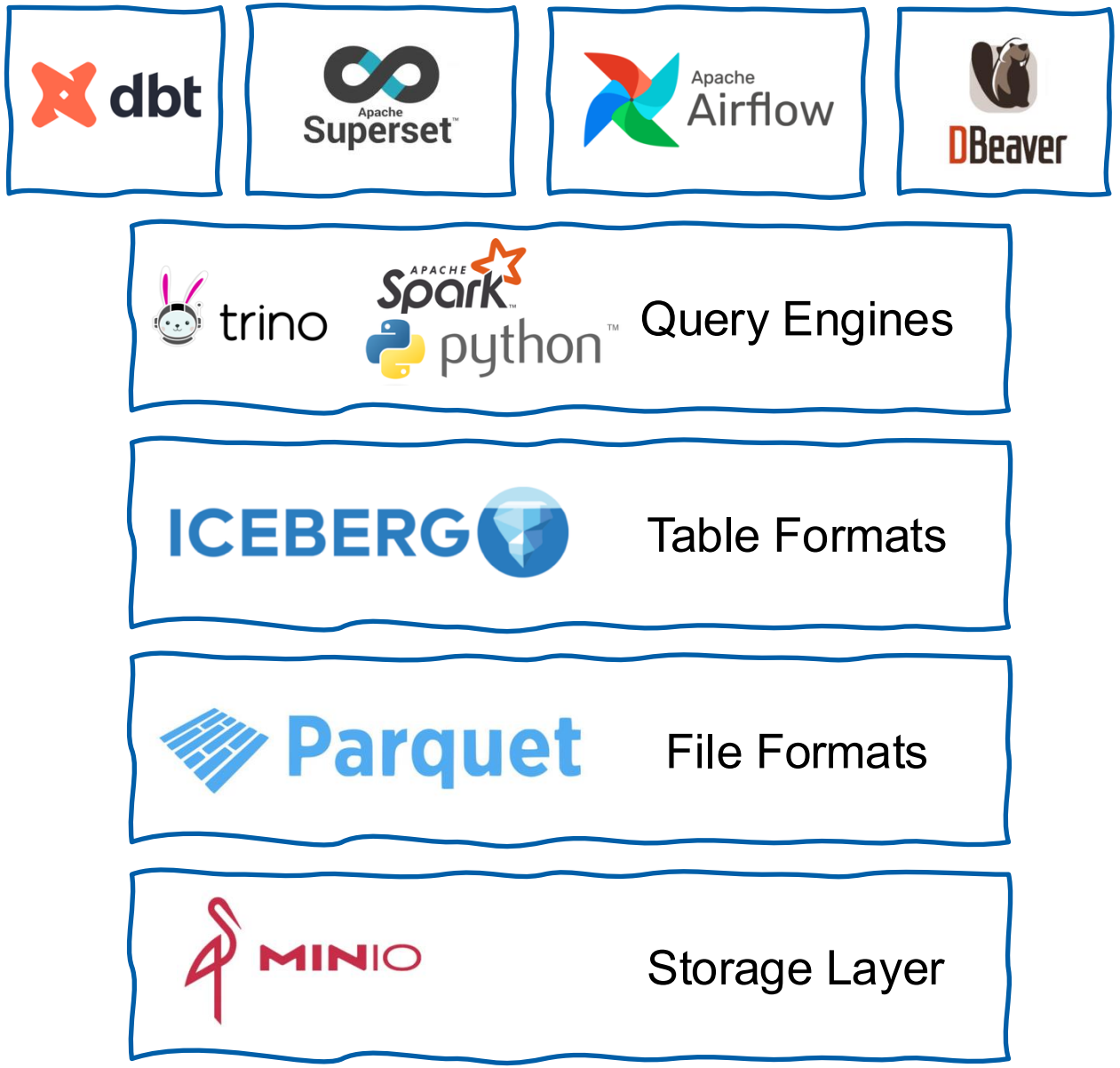
Some Challenges

Why Migrate from SQL Warehouses to a Lakehouse?

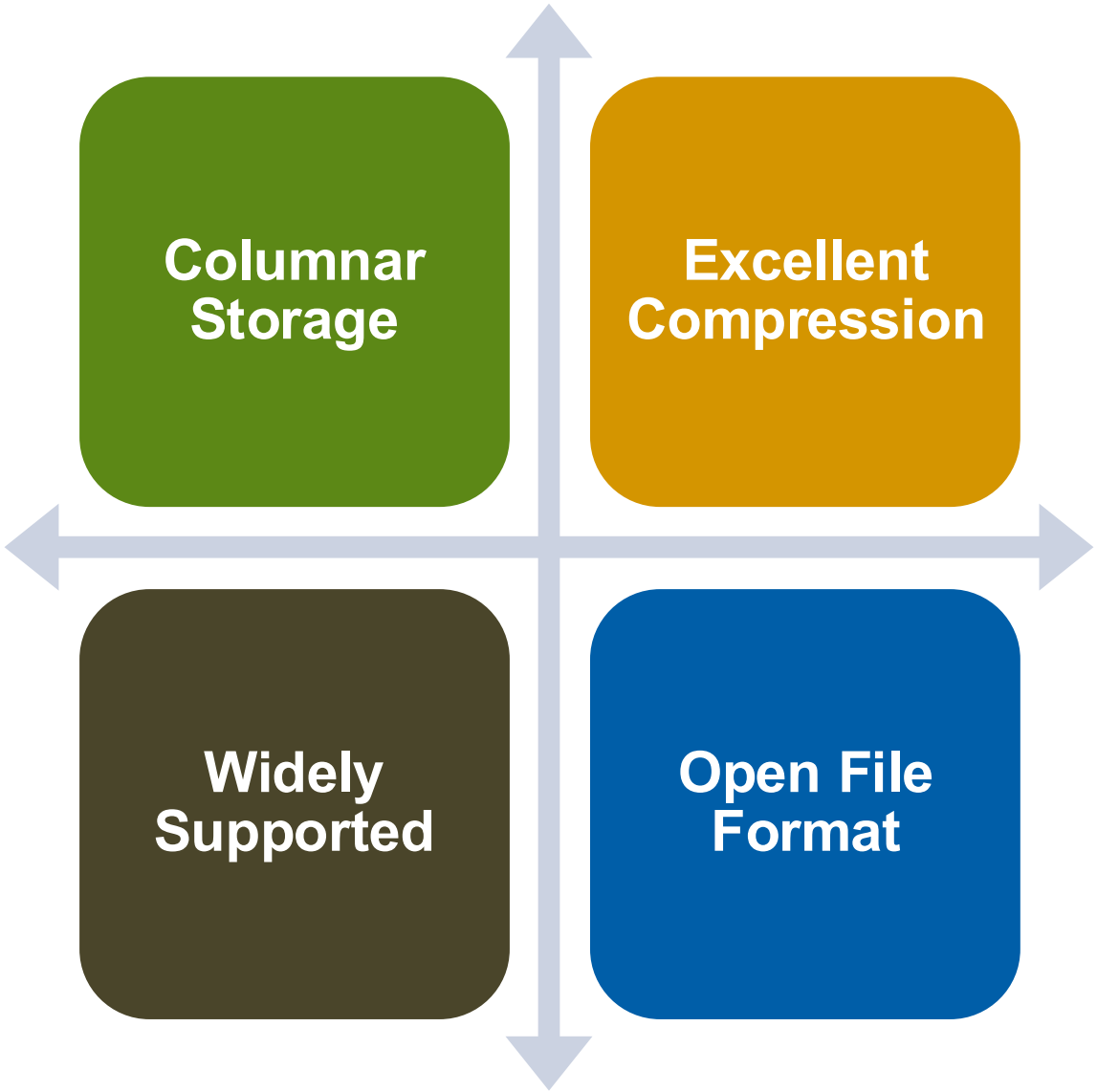
Traditional Data Warehouse	Lakehouse Architecture
Mostly Structured data	Both structured and unstructured
Expensive storage	Cheap object storage (S3/MinIO)
Proprietary formats	Open formats (Parquet + Iceberg)
Compute & storage are coupled	Compute & storage decoupled
Single query engine	Multi-engine workflows (Trino, Spark, Python, etc.)

The Lakehouse Architecture – Our data platform (Ocean) design

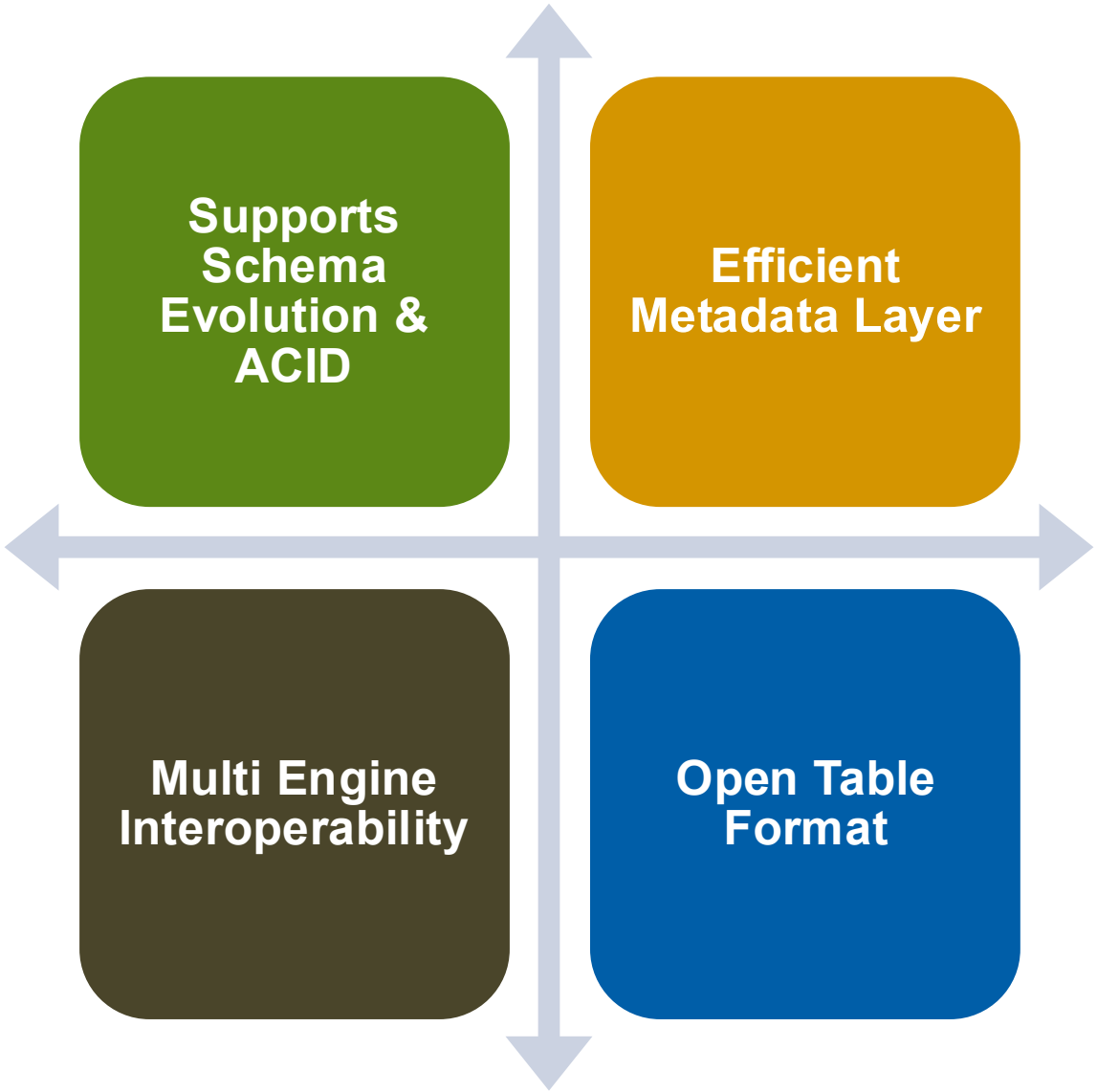
- **Storage Layer:** Durable, S3-compatible object storage for all raw and processed data.
- **File Format:** Columnar storage enabling fast analytical queries.
- **Table Format:** It brings database guarantees (ACID, schema evolution) to data stored as files.
- **Query Engines:** Required for reading/writing data to iceberg table format.
- **DBT:** It is a transformation framework that allows data teams to write modular SQL.



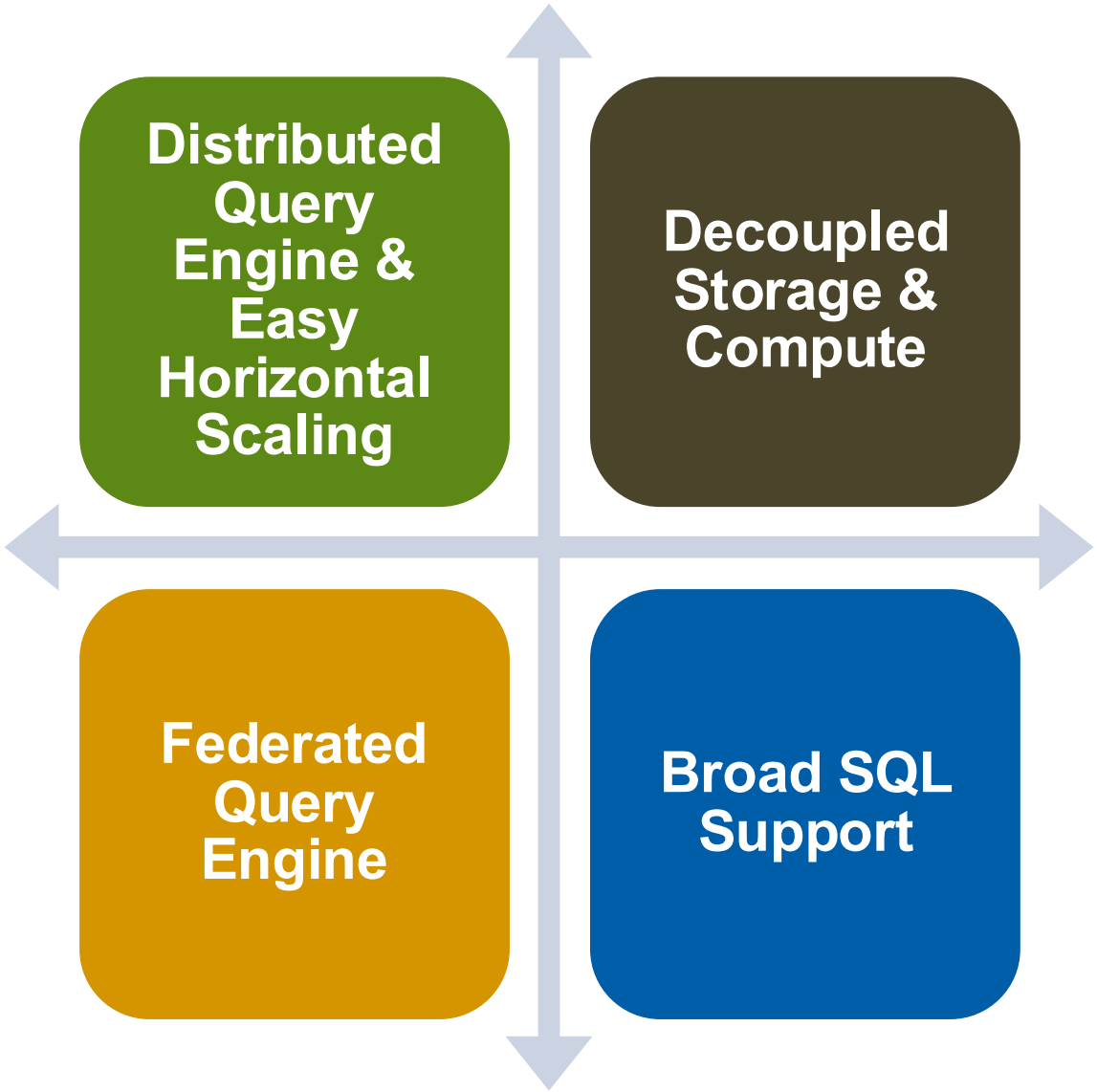
Why Parquet?


Columnar Storage
Excellent Compression
Widely Supported
Open File Format

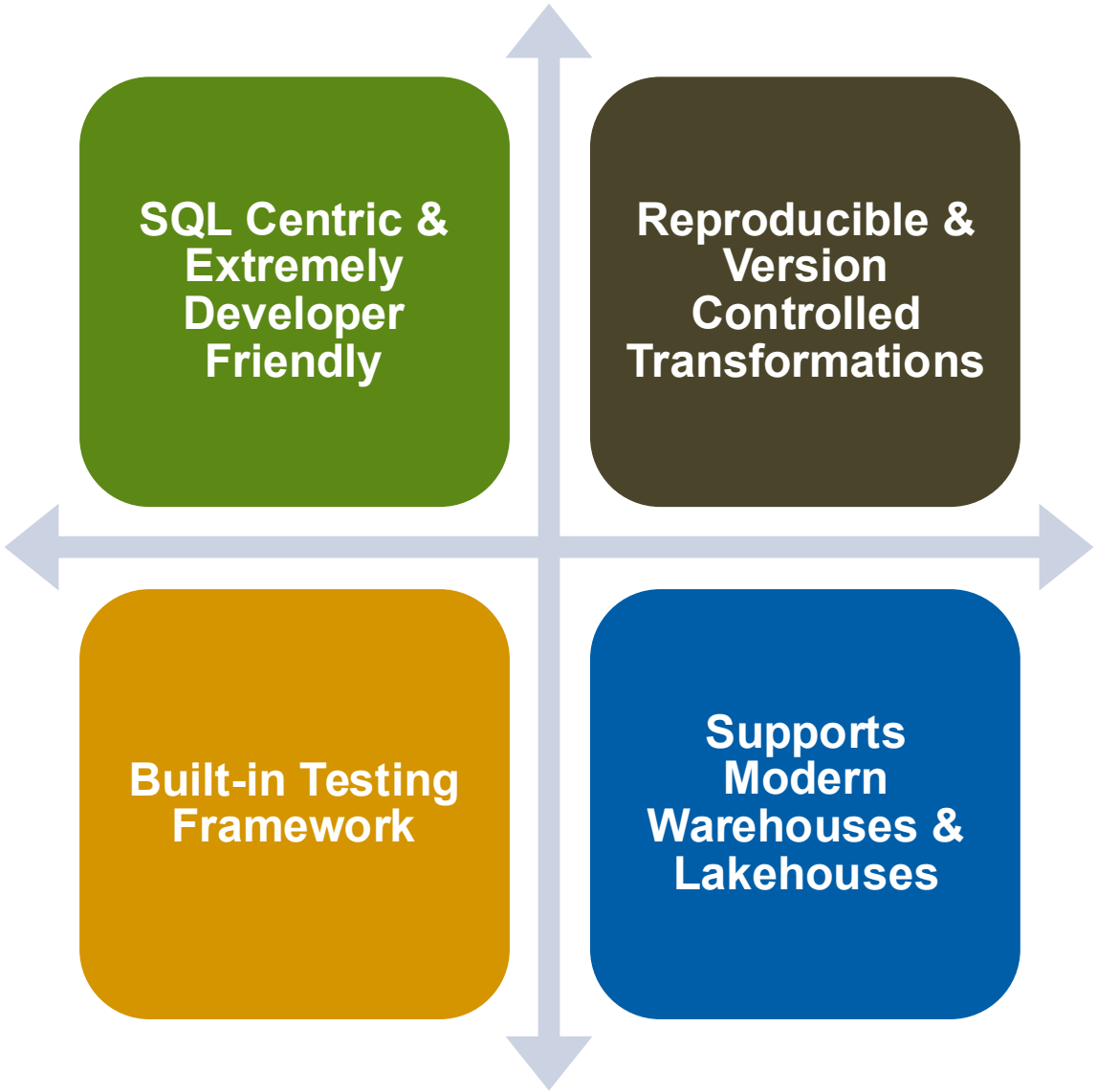
Why Iceberg?


Supports Schema Evolution & ACID
Efficient Metadata Layer
Multi Engine Interoperability
Open Table Format

Why Trino?


Distributed Query Engine & Easy Horizontal Scaling
Decoupled Storage & Compute
Federated Query Engine
Broad SQL Support

Why DBT?

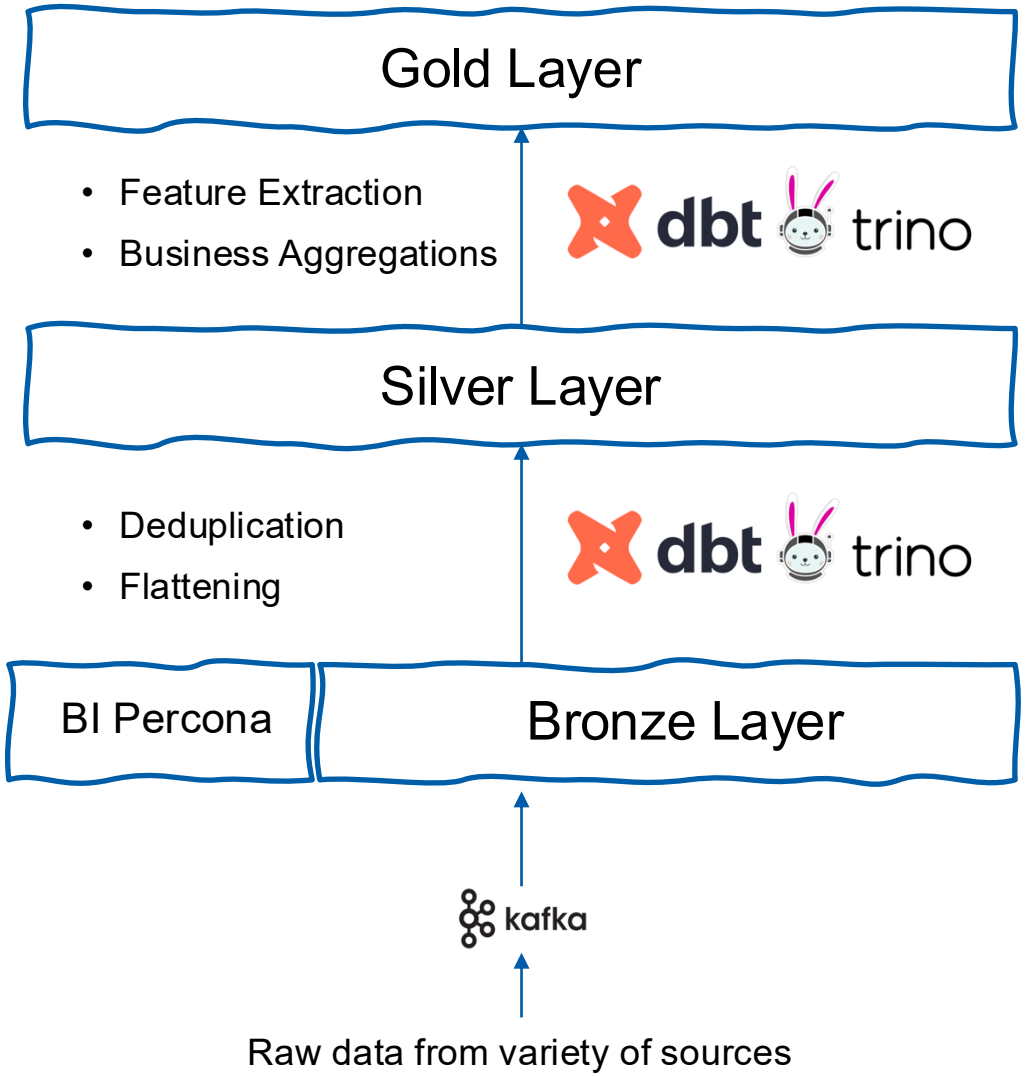

SQL Centric & Extremely Developer Friendly
Reproducible & Version Controlled Transformations
Built-in Testing Framework
Supports Modern Warehouses & Lakehouses

The Medallion Architecture - Data design pattern for organizing data

Medallion Architecture organizes data into three layers - **Bronze**, **Silver**, and **Gold**—to ensure clean, reliable, and optimized data flow.

- **Bronze Layer**: Raw, ingested data stored as-is. Kafka Connect writes raw data into the Bronze layer using Apache Iceberg on S3.
- **Silver Layer**: Cleansed and filtered data with transformations applied.
  DBT moves and transforms data from Bronze to Silver, leveraging Trino as the query engine.
- **Gold Layer**: Aggregated, business-ready datasets for model training, analytics and reporting.
  DBT further refines Silver data into Gold for final consumption.

Spark is used for optimizing tables across layers, ensuring performance and efficient storage.



Code Walkthrough

Some Challenges

Operational complexity

- Iceberg requires managing extra things like file compaction, snapshots, and an external Catalog, which adds setup and maintenance overhead.

Incremental logic can be tricky

- DBT incremental models rely on the developer to correctly define filters and update logic, which can be error-prone and requires careful handling.

Scaling tests in DBT can be tricky

- Built-in DBT tests run on the **entire table** with limited filtering options, so targeted validation is difficult. Custom tests help by allowing more precise, flexible checks.