Azure Fabric Data Lakehouse

Christian Cote

WhoAmI

Christian Cote

Canada

14 years in the program

Most Valuable Professionals

Data Engineer/Solution Architect

On-Prem ETL development using various ETL tools:

DTS / SSIS, Hummungbird Genio, Informatica, Datastage

DW Experience in various domains:

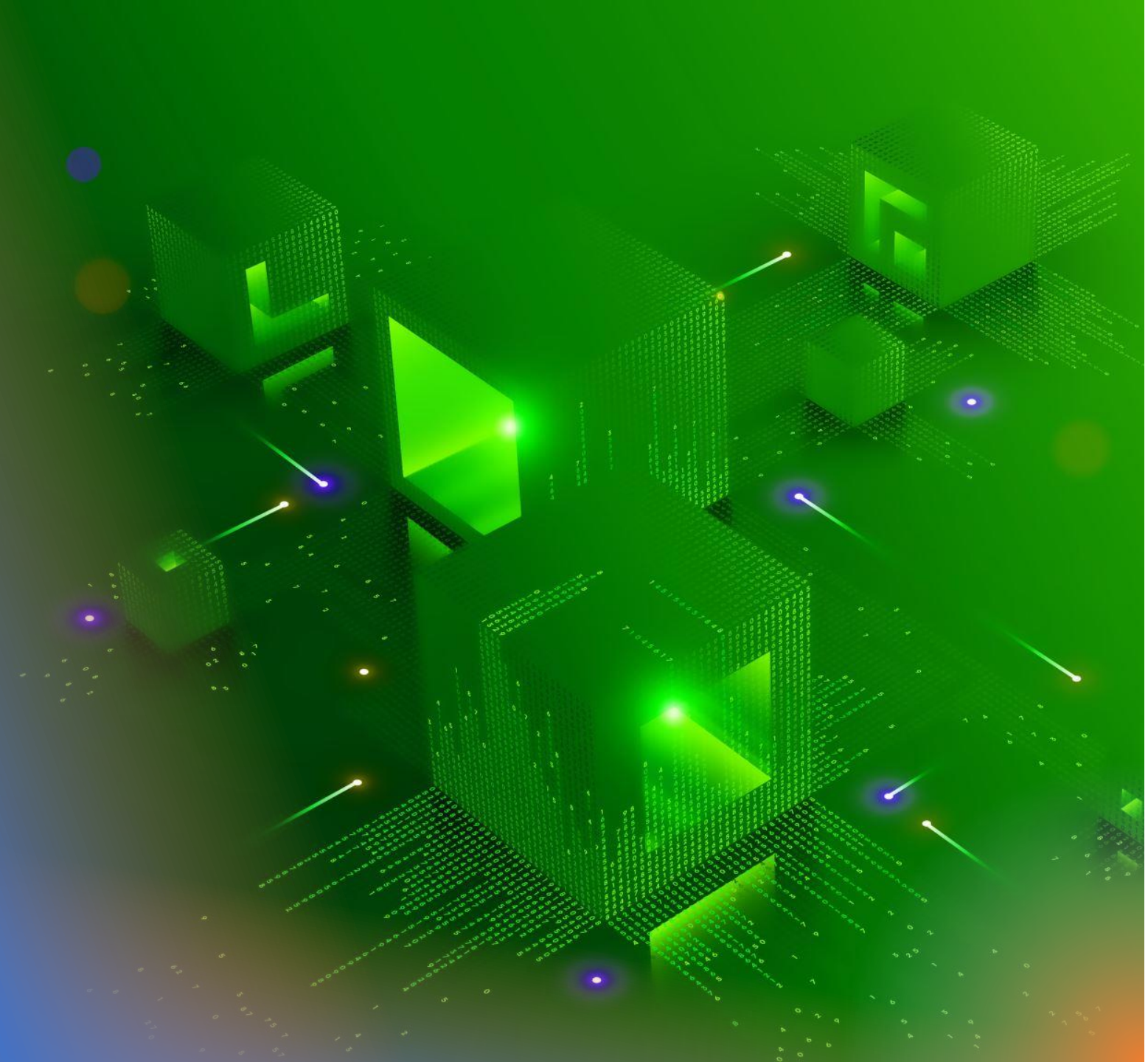
Pharmaceutical, finance, insurance, manufacturing and education

Specialized in Data warehouse/BI/Big Data

Writer of several books on data integration

Microsoft Data Platform Most Valuable Professional (MVP)

Montreal Data Platform User group leader



Agenda

- Data Lakehouse
  - What lead to it
  - Why we need it
- Data Lakehouse in Microsoft Fabric
  - Medallion Architecture
  - One Lake
  - Implementation


Data Lakehouse
Natural evolution of data lake and data warehouse

Evolution

1990's and 2000's

Data warehouses

- Structured or semi-structured data sets
- Not very flexible in term of data types
- Well organized
- Very performant

2010's

Data Lakes

- Any data types
- Any structures
- Not necessarily well organized
- Query performance varies with the toolset used

Data Warehouse

Pros:

Data is very well organized, cleaned and validated

Ease of consumption via simplified data model

Queries return in sub seconds

Many performances optimization possibilities

- Indexes
- Partitioning
- Distribution

Challenges:

Require a great deal of effort to do a data model easy to query

Requires lots of Data Engineering (ELT/ETL)

Schema on write – might lead to complex data pipelines

Data type limitation

Store limited amount of data – less history

Not the best place to archive data.

Data Lakes

Pros:

File based storage

Can store any file types

Can store all history (large volumes of files)

Right place to archive data

File based: easier to load via stream or batch data

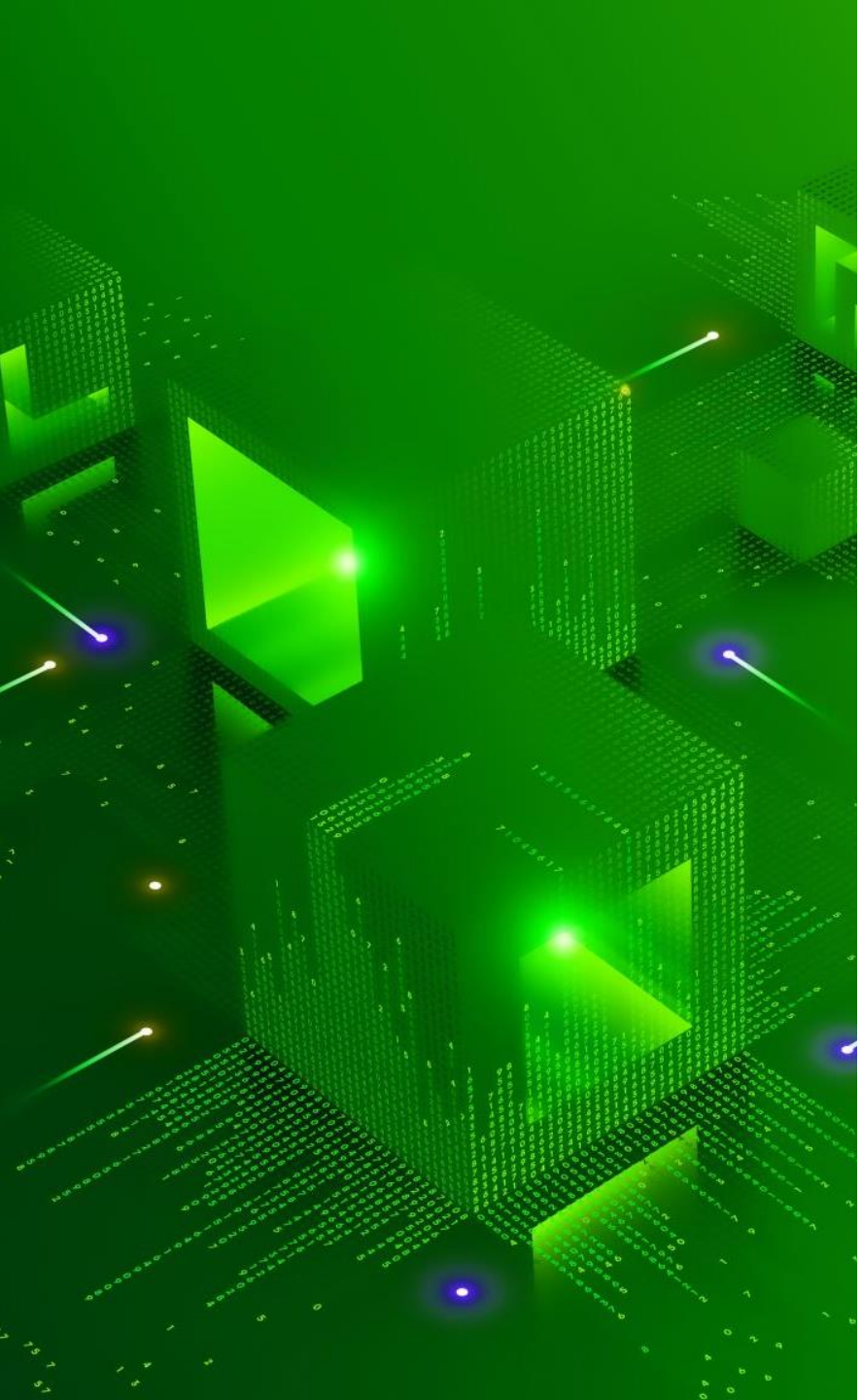
Challenges:

Queries in seconds rather that sub seconds.

Performance tuning is achieved mainly via partitioning

Consistency of the data may vary

Schema on read: can be harder to query compared to an optimized relational model

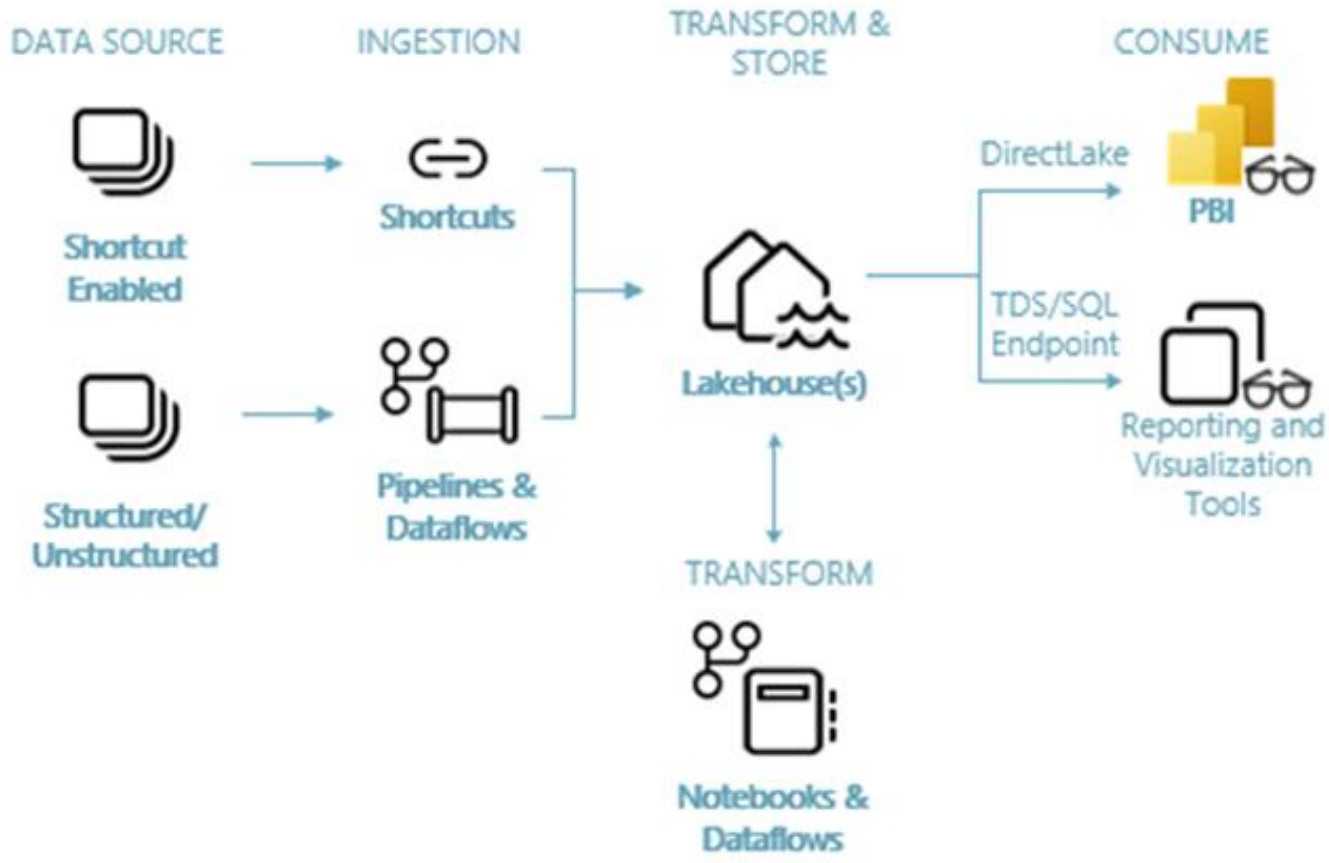


Data Lakehouse

- Combines both previous architectures
  - Data Lake
    - Storage and data type flexibility; Give access to various type of data
    - Unlimited storage space to store all history and archive data
  - Data Warehouse
    - Easier data querying with good performance
    - Cleaned and validated data
- Easier to get started
  - Build a data lake first
  - Build cleaner tables for specific purposes as needs arise
  - Cleaned tables are later integrated into a data warehouse if needed

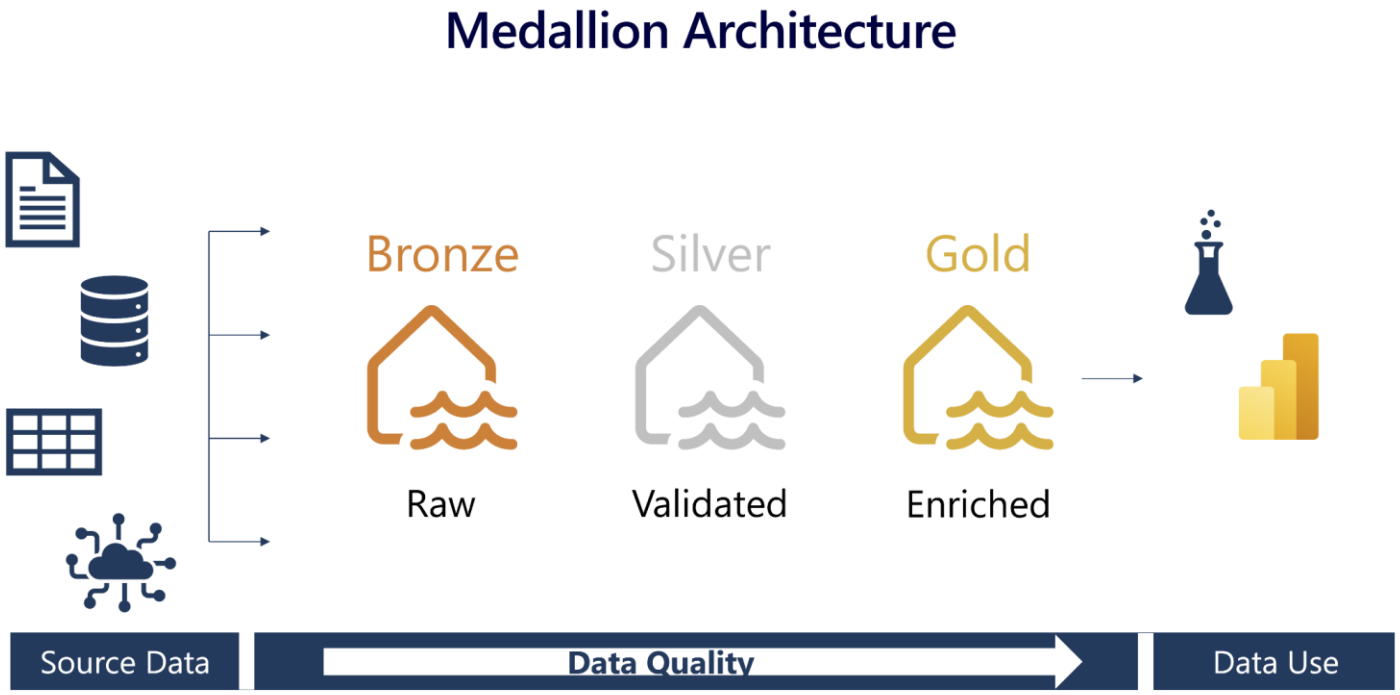
Microsoft Fabric Lakehouse

Overview


DATA SOURCE
INGESTION
TRANSFORM & STORE
CONSUME
Shortcut Enabled
Structured/ Unstructured
Shortcuts
Pipelines & Dataflows
Lakehouse(s)
TRANSFORM
Notebooks & Dataflows
DirectLake
TDS/SQL Endpoint
PBI
Reporting and Visualization Tools

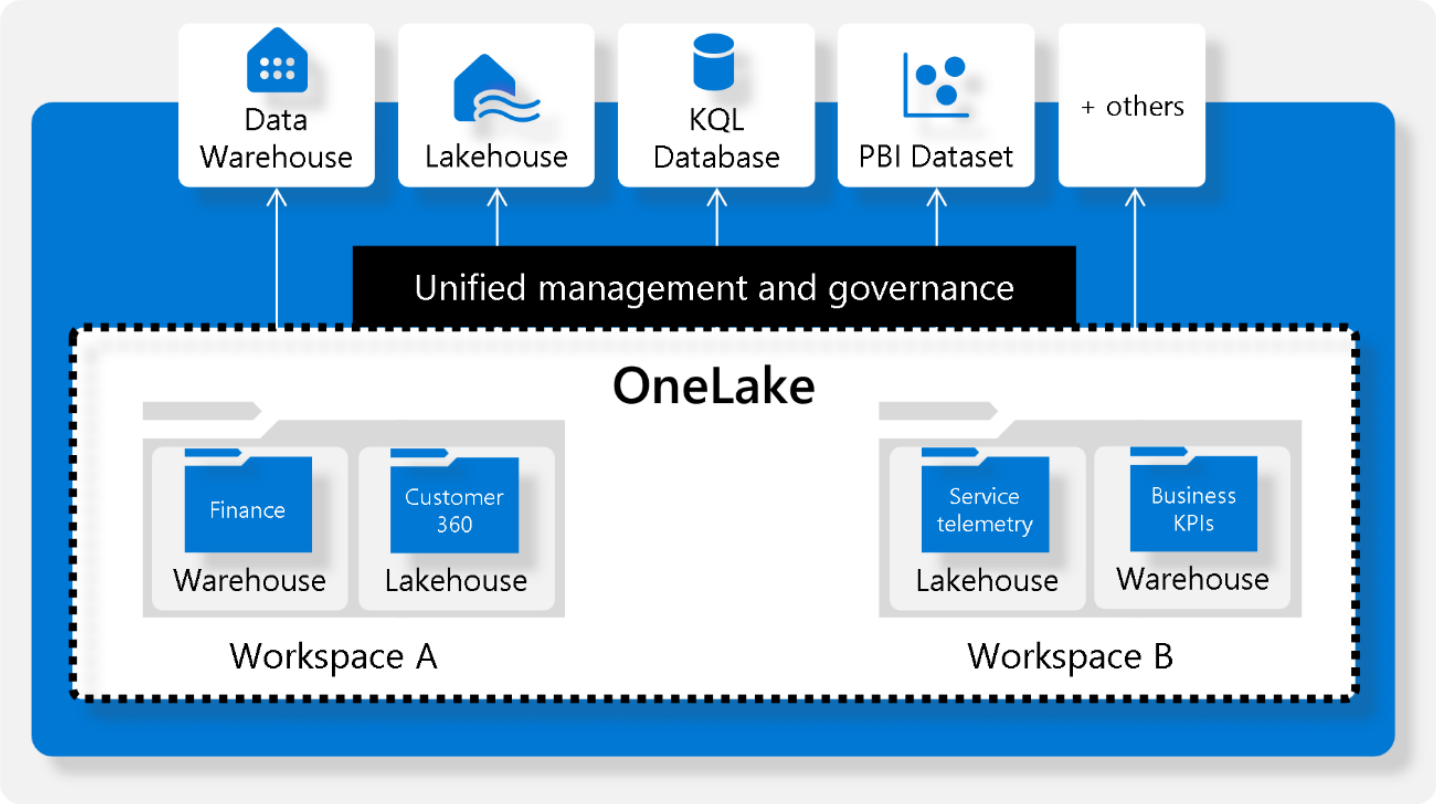
Lakehouse architecture

- **Bronze Zone**:
  - Data is ingested as is from sources systems
- **Silver Zone**:
  - Data is organized to facilitate querying
  - Some system data cleansing is applied
- **Gold zone**:
  - Data is aligned to answer most queries: business based
  - Can be stored in a database for better performances

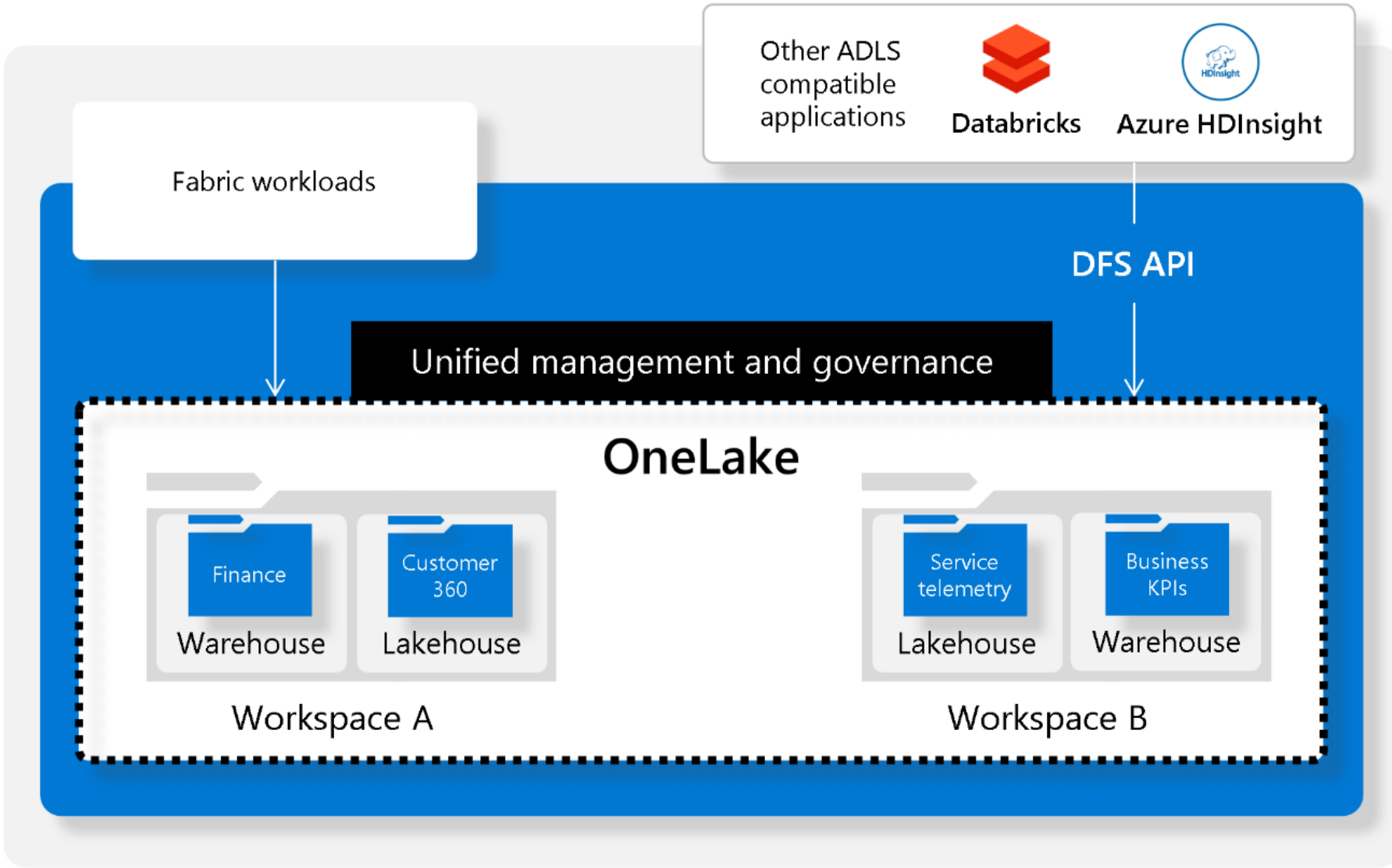


OneLake

- Data architecture platform that allows you to store, manage, and analyze both structured and unstructured data in a single location
- Helps eliminate the need for data movement and data copies
- Provides immediate access to data from various sources such as ADLS, S3, data warehouses, KQL databases, and other Lakehouses
- Integrates with various tools and frameworks in Fabric, such as Spark, SQL, Dataset, Power BI, etc.

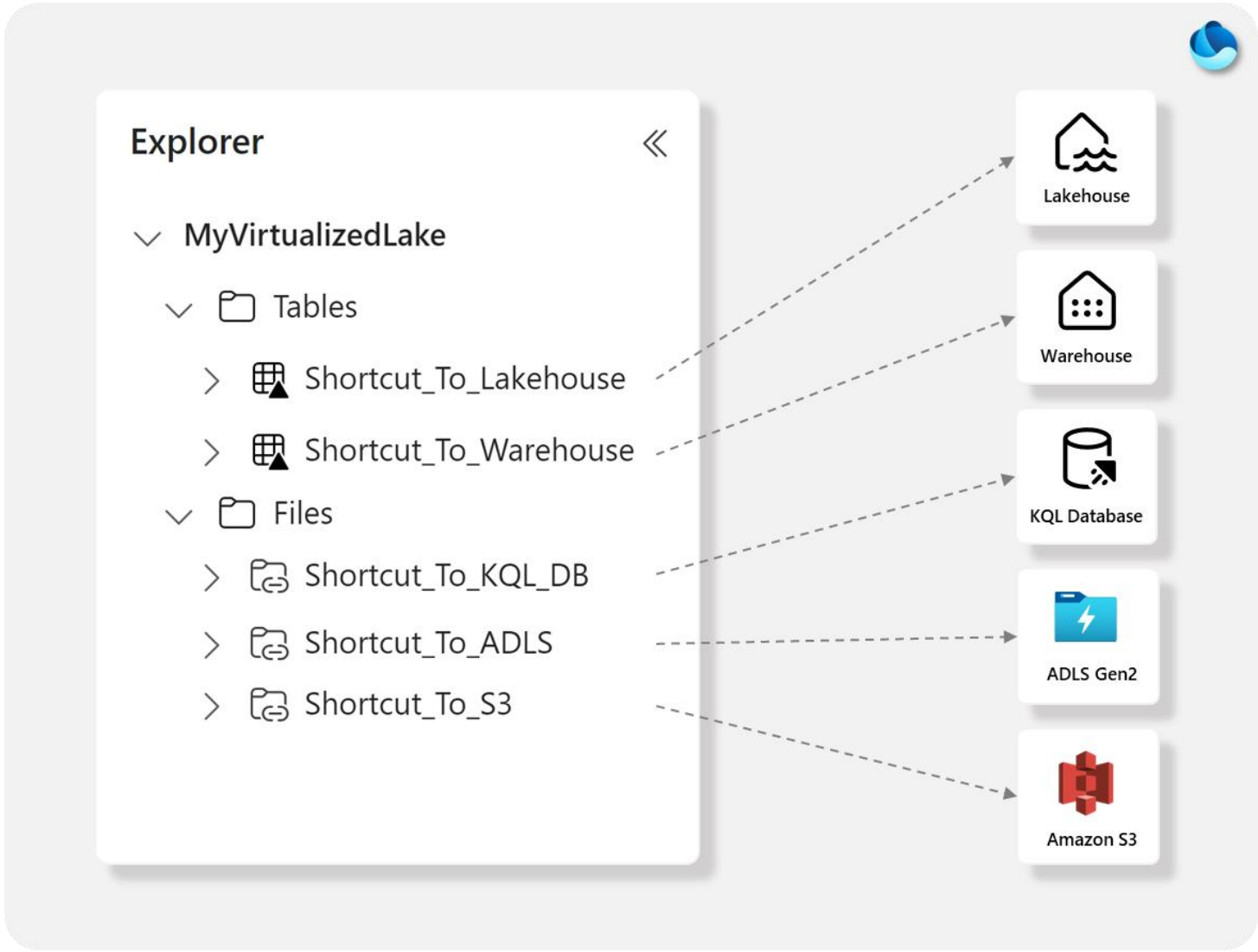


OneLake – open API for other transformation tools



Onelake shortcuts - One copy of data

- Data is not copied over OneLake, it is consumed at the source
- Shortcuts can lead to egress (network) costs



Managed tables

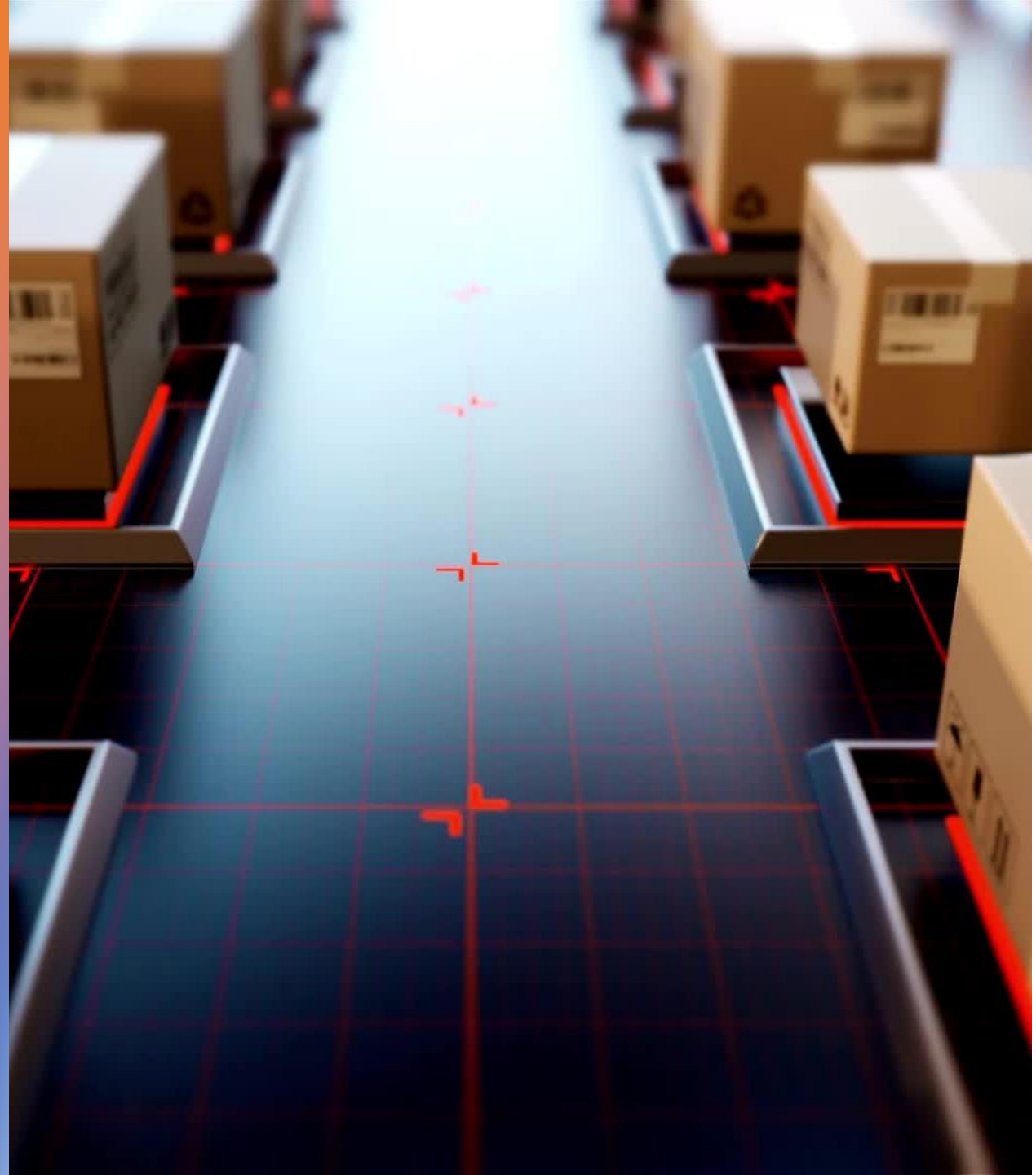
- Store data and metadata in Delta Lake
- Use CREATE TABLE statement only
- Optimized by Microsoft Fabric
- Data and metadata are deleted with DROP TABLE
- Best tables to use for data consumption

Delta format is the standard for data access in OneLake

- Open format storage layer
- ACID transactions on Spark: Serializable isolation levels ensure that readers never see inconsistent data.
- Scalable metadata handling: Leverages Spark's distributed processing power to handle all the metadata for petabyte-scale tables with billions of files at ease.
- Streaming and batch unification: A table in Delta Lake is a batch table as well as a streaming source and sink. Streaming data ingest, batch historic backfill, interactive queries all just work out of the box.
- Schema enforcement: Automatically handles schema variations to prevent insertion of bad records during ingestion.
- Time travel: Data versioning enables rollbacks, full historical audit trails, and reproducible machine learning experiments.

Unmanaged tables

- Store data externally and only reference it in metadata.
- Use CREATE TABLE USING with a data source.
- Need user optimization.
- Only metadata is deleted with DROP TABLE, data is kept.
- Used mainly for data integration



Delta table performance – V-Order

- Proprietary technology developed by Microsoft.
- Write time optimization for parquet files that enables faster reads by Microsoft Fabric compute engines.
- It applies special sorting, row group distribution, dictionary encoding and compression on parquet files to reduce resource consumption.
- Sorting increases write time by 15% but improves compression by up to 50%.
- Compatible with Power BI, SQL, Spark and other Microsoft Fabric compute engines.
- Provides cost efficiency and performance benefits for data analysis and processing.
- **Makes Delta tables 10-100X faster than standard parquet files**

Lake House Implementation

- Lakehouse
  - Explore data via files and folders
  - Load to table used to create managed tables
  - Read/Write
- SQL Endpoint
  - Query Delta tables using TSQL
  - Schemas, Stored procedure, etc.
  - Read-only
  - Row level security

Row Level security

- Row-Level Security (RLS) simplifies the design and coding of security in your application.
- RLS helps you implement restrictions on data row access.
- Access rules are kept in the database, and are enforce whenever anyone tries to access data from any part of the system

Demo

Fabric Lakehouse

Data engineering decision guide

	Pipeline copy activity	Dataflow Gen 2	Spark
Use case	Data lake and data warehouse migration, data ingestion, lightweight transformation	Data ingestion, data transformation, data wrangling, data profiling	Data ingestion, data transformation, data processing, data profiling
Primary developer persona	Data engineer, data integrator	Data engineer, data integrator, business analyst	Data engineer, data scientist, data developer
Primary developer skill set	ETL, SQL, JSON	ETL, M, SQL	Spark (Scala, Python, Spark SQL, R)
Code written	No code, low code	No code, low code	Code
Data volume	Low to high	Low to high	Low to high
Development interface	Wizard, canvas	Power query	Notebook, Spark job definition
Sources	30+ connectors	150+ connectors	Hundreds of Spark libraries
Destinations	18+ connectors	Lakehouse, Azure SQL database, Azure Data explorer, Azure Synapse analytics	Hundreds of Spark libraries
Transformation complexity	Low: lightweight - type conversion, column mapping, merge/split files, flatten hierarchy	Low to high: 300+ transformation functions	Low to high: support for native Spark and open-source libraries

Data warehouse and lakehouse properties

	Data warehouse	Lakehouse	Power BI Datamart	KQL Database
Data volume	Unlimited	Unlimited	Up to 100 GB	Unlimited
Type of data	Structured	Unstructured,semi-structured,structured	Structured	Unstructured, semi-structured, structured
Primary developer persona	Data warehouse developer, SQL engineer	Data engineer, data scientist	Citizen developer	Citizen Data scientist, Data engineer, Data scientist, SQL engineer
Primary developer skill set	SQL	Spark(Scala, PySpark, Spark SQL, R)	No code, SQL	No code, KQL, SQL
Data organized by	Databases, schemas, and tables	Folders and files, databases, and tables	Database, tables, queries	Databases, schemas, and tables
Read operations	Spark,T-SQL	Spark,T-SQL	Spark,T-SQL,Power BI	KQL, T-SQL, Spark, Power BI
Write operations	T-SQL	Spark(Scala, PySpark, Spark SQL, R)	Dataflows, T-SQL	KQL, Spark, connector ecosystem
Multi-table transactions	Yes	No	No	Yes, for multi-table ingestion. See update policy.
Primary development interface	SQL scripts	Spark notebooks, Spark job definitions	Power BI	KQL Queryset, KQL Database
Security	Object level (table, view, function, stored procedure, etc.), column level, row level, DDL/DML	Row level, table level (when using T-SQL), none for Spark	Built-in RLS editor	Row-level Security
Access data via shortcuts	Yes (indirectly through the lakehouse)	Yes	No	Yes
Can be a source for shortcuts	Yes (tables)	Yes (files and tables)	No	Yes
Transaction performance - updates to a few rows	Good	Slow	--	--
Query across items	Yes, query across lakehouse and warehouse tables	Yes, query across Lakehouse and warehouse tables; query across Lakehouses (including shortcuts using Spark)	No	Yes, query across KQL Databases, lakehouses, and warehouses with shortcuts
Advanced analytics				Time Series native elements, Full geospatial storing and query capabilities
Advanced formatting support				Full indexing for free text and semi-structured data like JSON

To sum up

- Data lake houses combine the best of both data lakes and data warehouse capabilities
- Fabric OneLake is unifying needs for all personas
- There are many ways to load and transform data in a LakeHousewith Fabric
  - Copy data
  - Shortcuts
  - Dataflows
  - Spark
- Spark Clusters are managed by Azure Fabric, no need to estimate the size beforehand

Questions?

Thank you!