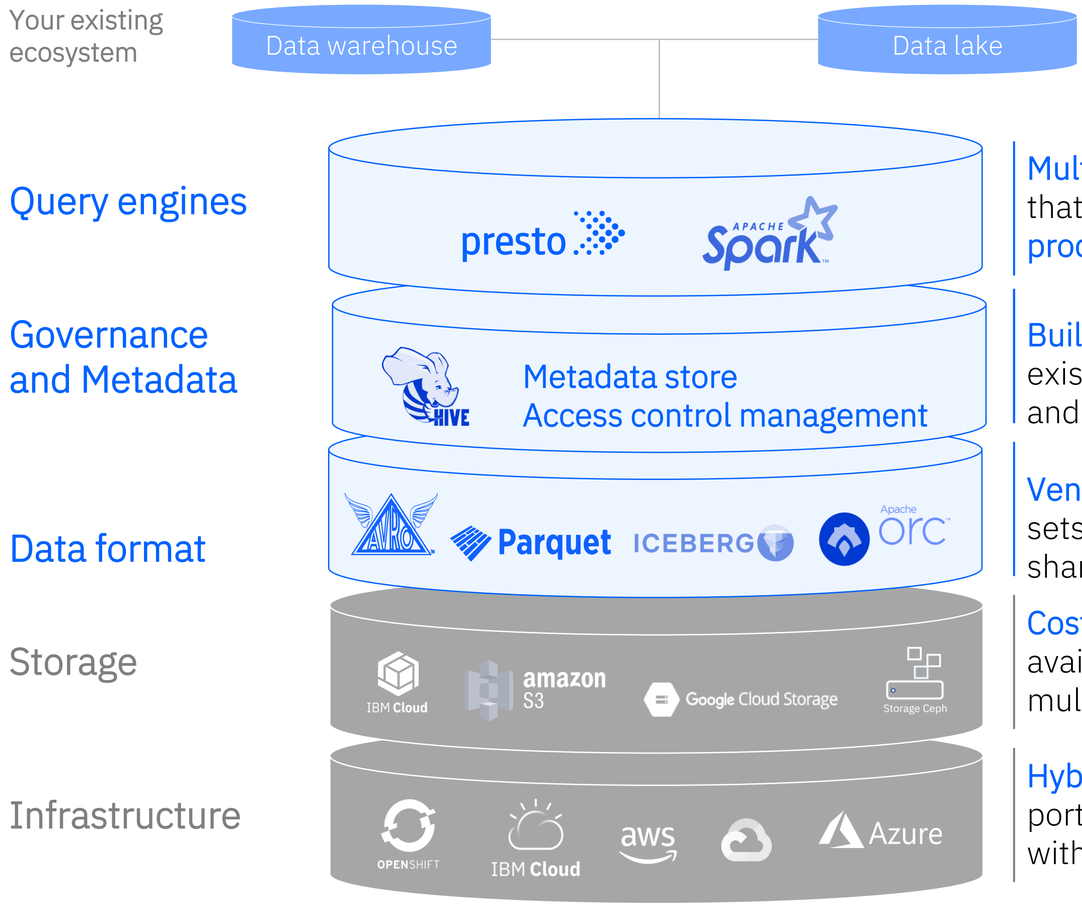
Introduction to Data Lakehouse Open-Source Technologies

Kelly Schlamb
WW Technology Sales Enablement
kschlamb@ca.ibm.com

IBM **watsonx.data** – the next generation data lakehouse

Completely open.
No lock-in!

Built on a foundation of industry-embraced open-source technologies.



watsonx.data

Multiple engines including Presto and Spark that provide **fast, reliable, and efficient processing of big data** at scale

Built-in governance that is compatible with existing solutions such as watsonx.governance and IBM Knowledge Catalog

Vendor agnostic open formats for analytic data sets, allowing different engines to access and share the same data, at the same time

Cost effective, simple object storage available across hybrid-cloud and multicloud environments

Hybrid-cloud deployments and workload portability across hyperscalers and on-prem with Red Hat OpenShift

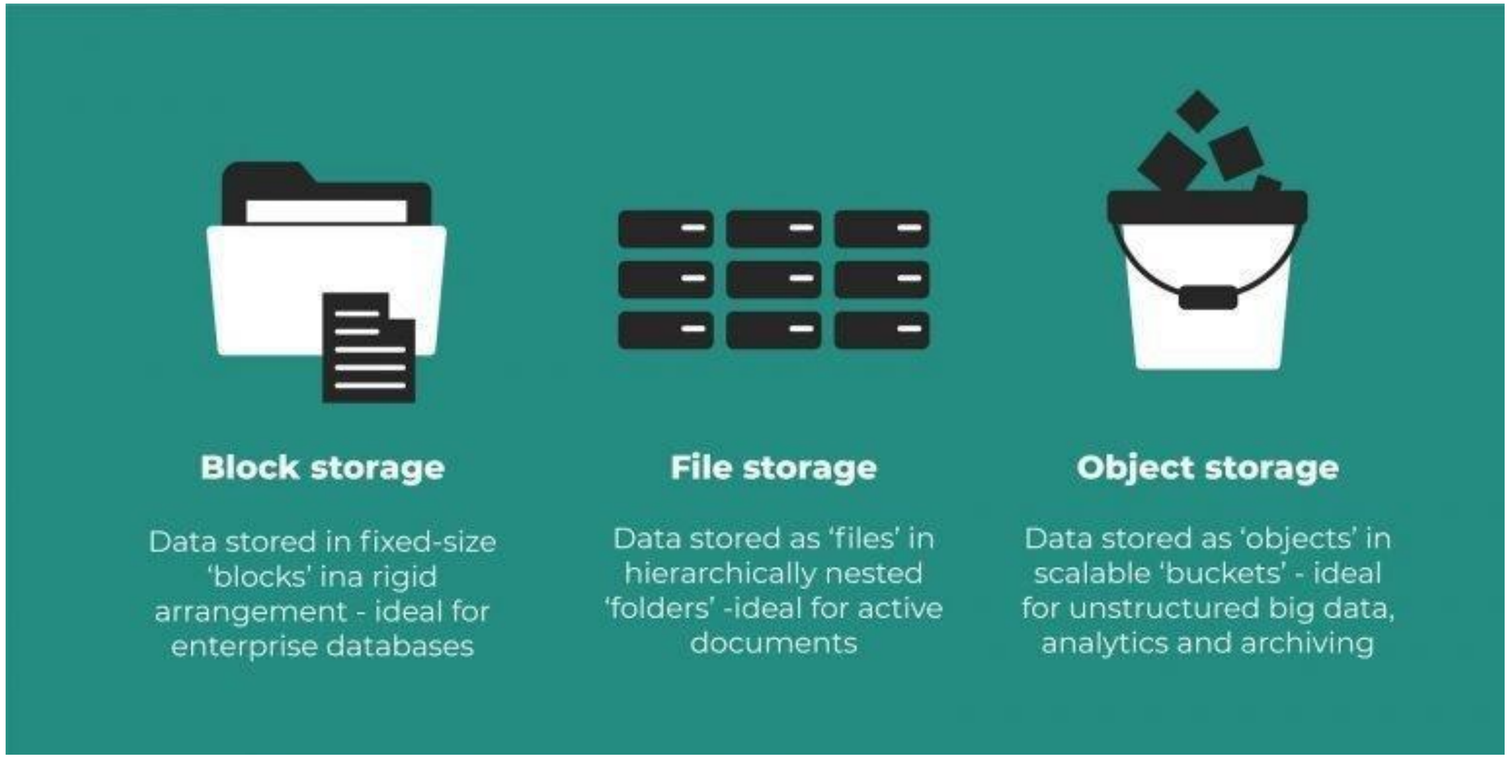
Core watsonx.data functionality

Ecosystem infrastructure

Let’s get familiar with some common open-source technologies used in watsonx.data and data lakehouse architectures in general…

Storage,
data file formats,
and table formats

What is object storage?



Object storage:

- Low cost
- Near unlimited scalability
- Extreme durability & reliability (99.999999999%)
- High throughput
- High latency (but can be compensated for)
- Basic units are *objects*, which are organized in *buckets*

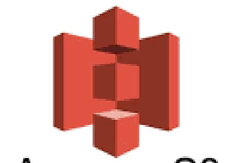
- Most notable provider for object storage is Amazon S3 (Simple Storage Service)
- Other vendors offer S3-compatible object storage

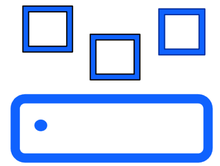
amazon S3
IBM Spectrum Fusion
Storage Ceph
DELL EMC ISILON
Cloud Object Storage
NetApp

The rise of cloud object storage for data lakes and lakehouses

Cloud object storage technology is displacing HDFS as de facto storage technology for data lakes

	Object Storage	HDFS	Object Storage vs. HDFS
Elasticity	Yes (decoupled)	No	S3 is more elastic
Cost/TB/Month	$23	$206	10X
Performance	20MB/s/core	90MB/s/core	2x better price/perf
Availability	99.99%	99.9% (estimated)	10X
Durability	99.999999999%	99.9999% (estimated)	10X+
Transactional writes	Most technologies now provide strong consistency	Yes	Comparable





Common data file formats

Computer systems and applications store data in files

Data can be stored in binary or text format

File formats can be open or closed (proprietary/lock-in)

Open formats (Parquet, ORC, and Avro) are commonly used in data lakes and lakehouses

CSV

- Human-readable text
- Each row corresponds to a single data record
- Each record consists of one or more fields, delimited by commas

{ JSON }

- Human-readable text
- Open file and data interchange format
- Consists of attribute-value pairs and arrays
- JSON = JavaScript Object Notation

Parquet

- Open-source
- Binary columnar storage
- Designed for efficient data storage and fast retrieval
- Highly compressible
- Self-describing

Apache orc

- Open-source
- Binary columnar storage
- Designed and optimized for Hive data
- Self-describing
- Similar in concept to Parquet

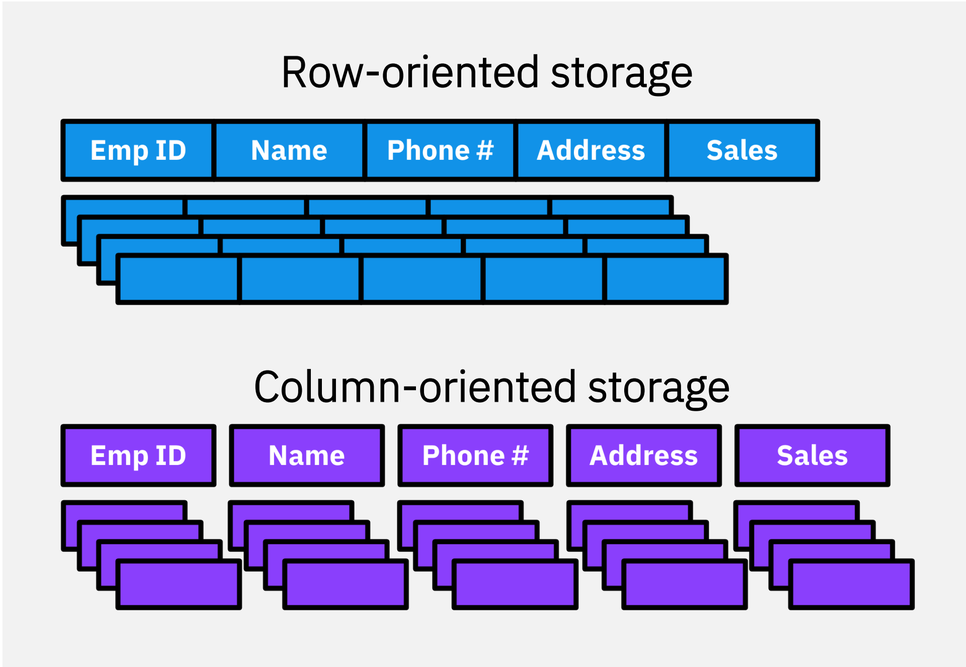
AVRO

- Open-source
- Row-oriented data format and serialization framework
- Robust support for schema evolution
- Mix of text/binary

Apache Parquet

Parquet is designed to support fast data processing for complex data

- Open-source
- Columnar storage
- Highly compressible with configurable compression options and extendable encoding schemas by data type
- Self-describing: schema and structure metadata is included
- Schema evolution with support for automatic schema merging

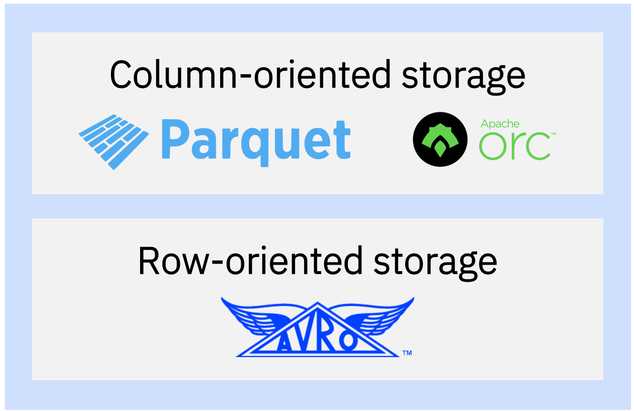


Why do these things matter in a lakehouse?

- Performance of queries directly impacted by size and amount of file(s) being read
- Ability to read/write data to an open format from multiple runtime engines enables collaboration
- Size of data stored, amount of data scanned, and amount of data transported affect the charges incurred in using a lakehouse (depending on the pricing model)

Apache ORC

- Open-source, **columnar storage** format
  - Similar in concept to Parquet, but different design
  - Parquet considered to be more widely used than ORC
- Highly compressible, with multiple compression options
  - Considered to have higher compression rates than Parquet
- Self-describing and type-aware
- Support for schema evolution
- Built-in indexes to enable skipping of data not relevant to a query
- Excellent performance for read-heavy workloads
  - ORC generally better for workloads involving frequent updates or appends
  - Parquet generally better for write-once, read-many analytics



Apache Avro

- Open-source, **row-based** storage and serialization format
  - Can be used for file storage or message passing
- Beneficial for write-intensive workloads
- Format contains a mix of text and binary
  - Data definition: Text-based JSON
  - Data blocks: Binary
- Robust support for schema evolution
  - Handles missing/added/changed fields
- Language-neutral data serialization
  - APIs included for Java, Python, Ruby, C, C++, and more

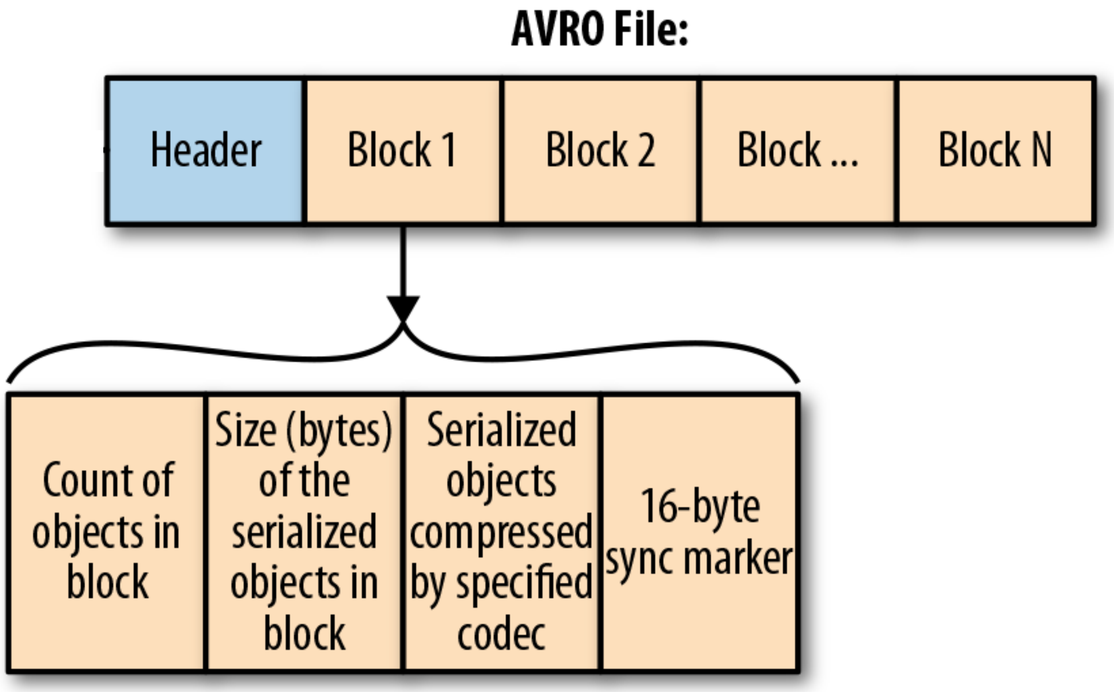


Table management and formats

Sits “above” the data file layer

Organizes and manages table metadata and data

Typically supports multiple underlying disk file formats (Parquet, Avro, ORC, etc.)

May offer transactional concurrency, I/U/D, indexing, time-based queries, and other capabilities

ICEBERG

- Open-source
- Designed for large, petabyte (PB)-scale tables
- ACID-compliant transaction support
- Capabilities not traditionally available with other table formats, including schema evolution, partition evolution, and table version rollback – all without re-writing data
- Advanced data filtering
- Time-travel queries let you see data at points in the past

Apache hudi

- Open-source
- Manages the storage of large datasets on HDFS and cloud object storage
- Includes support for tables, ACID transactions, upserts/ deletes, advanced indexes, streaming ingestion services, concurrency, data clustering, and asynchronous compaction
- Multiple query options: snapshot, incremental, and read-optimized

DELTA LAKE

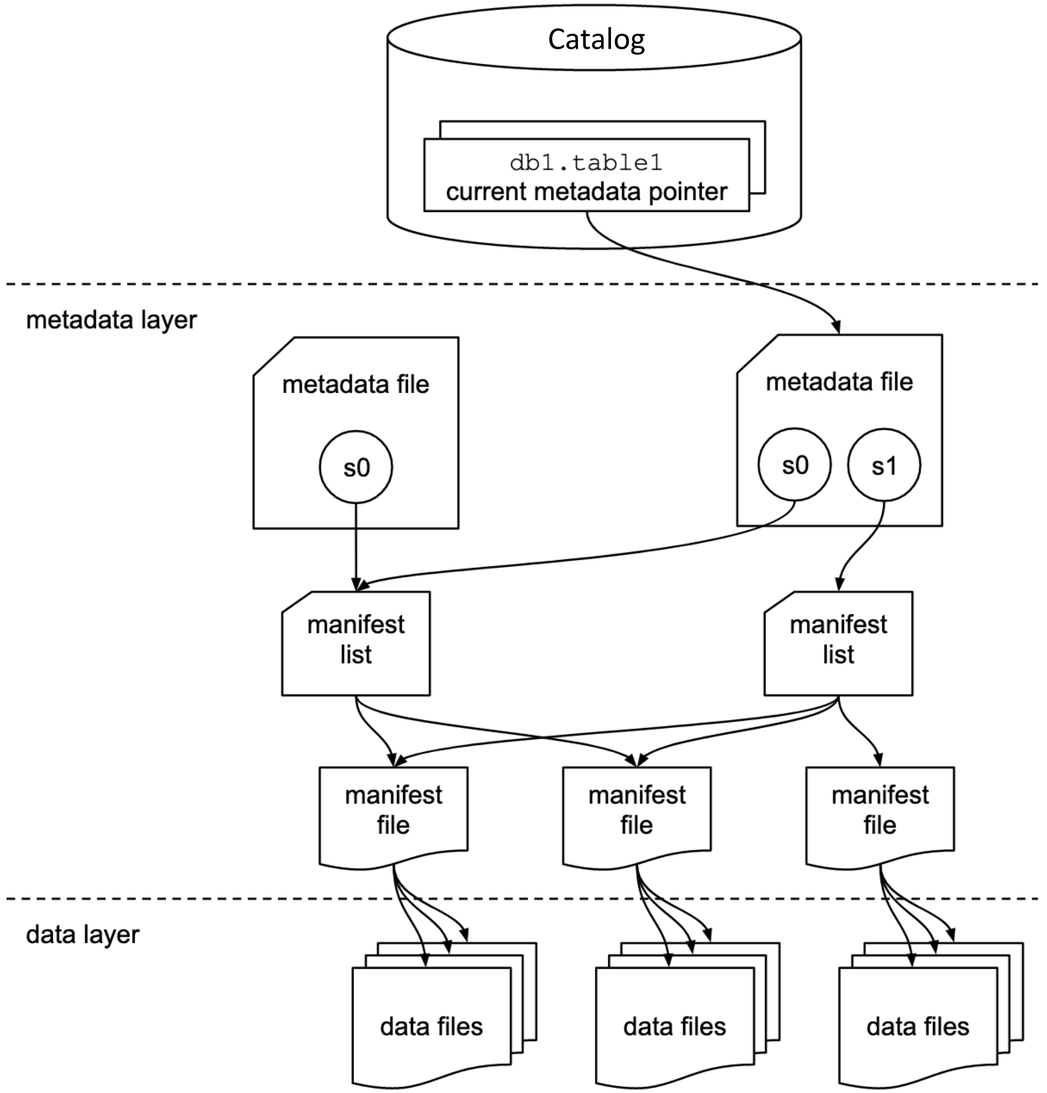
- Open-source, but Databricks is primary contributor and user, and controls all commits to the project – so “closed”
- Foundation for storing data in the Databricks Lakehouse Platform
- Extends Parquet data files with a file-based transaction log for ACID transactions and scalable metadata handling
- Capabilities include indexing, data skipping, compression, caching, and time-travel queries
- Designed to handle batch as well as streaming data

Why Apache Iceberg for data lakehouses?

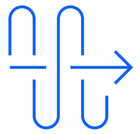
Open-source data table format that helps simplify data processing on large dataset stored in data lakes

People love it because it has:

- SQL — Use it to build the data lake and perform most operations without learning a new language
- Data Consistency — ACID compliance (not just append data operations to tables)
- Schema Evolution — Add/remove columns without distributing underlying table structure
- Data Versioning — Time travel support that lets you analyze data changes between update and deletes
- Cross Platform Support — Supports variety of storage systems and query engines (Spark, Presto, Hive, +++)



ACID transactions



ACID refers to a set of properties of database transactions intended to **guarantee data validity** despite errors, power failures, and other mishaps

Atomicity

Guarantees that each transaction is a single event that either succeeds or fails completely; there is no half-way state.

Consistency

Ensures that data is in a consistent state when a transaction starts and when it ends, guaranteeing that data is accurate and reliable.

Isolation

Allows multiple transactions to occur at the same time without interfering with each other, ensuring that each transaction executes independently.

Durability

Means that data is not lost or corrupted once a transaction is submitted. Data can be recovered in the event of a system failure, such as a power outage.

Metastore

What is a metastore?

- Manages metadata for the tables in the lakehouse, including:
  - Schema information (column names, types)
  - Location and type of data files
- Similar in principle to the system catalogs of a relational database
- Shared metastore ensures query engines see schema and data consistently
- May be a built-in component of a larger integration/governance solution

HMS used by watsonx.data

- Hive metastore (HMS) is a component of Hive, but can run standalone
- Open-source
- Manage tables on HDFS and cloud object storage
- Pervasive use in industry

AWS Glue Data Catalog

- Component of AWS Glue integration service
- Inventories data assets of AWS data sources
- Includes location, schema, and runtime metrics

Microsoft Purview Data Catalog

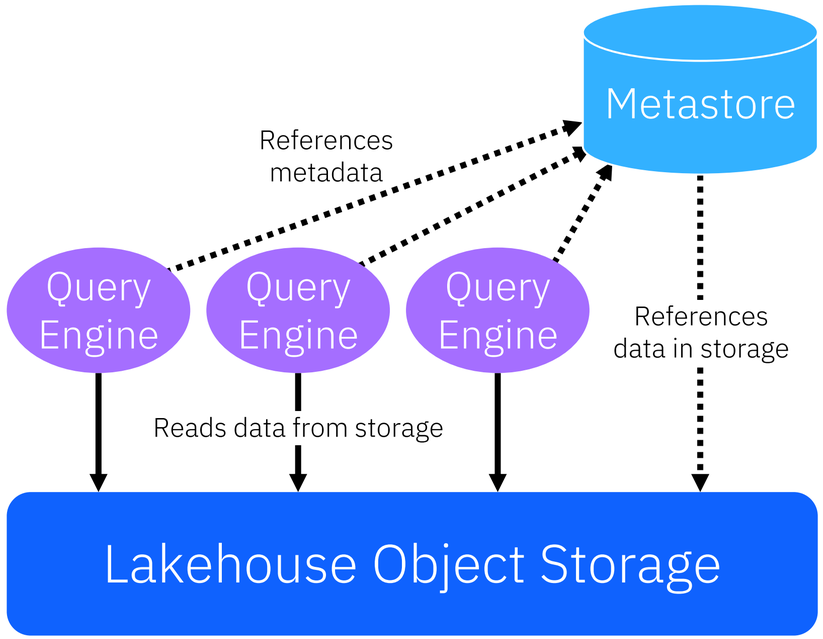
- Component of Microsoft Purview data governance solution
- Helps manage on-premises, multicloud, and SaaS data
- Offers discovery, classification, and lineage

Databricks Unity Catalog

- Provides centralized access control, auditing, lineage, and data discovery across a Databricks lakehouse
- Contains data and AI assets including files, tables, machine learning models, and dashboards

Hive Metastore (HMS)

- Open-source **Apache Hive** was built to provide an SQL-like query interface for data stored in Hadoop
- **Hive Metastore (HMS)** is a component of Hive that stores metadata for tables, including schema and location
- HMS can be deployed standalone, without the rest of Hive (often needed for lakehouses, like watsonx.data)
- Query engines use the metadata in HMS to optimize query execution plans
- The metadata is stored in a traditional relational database (PostgreSQL in the case of watsonx.data)
- In watsonx.data, IBM Knowledge Catalog integrates with HMS to provide policy-based access and governance



Query engines

Presto

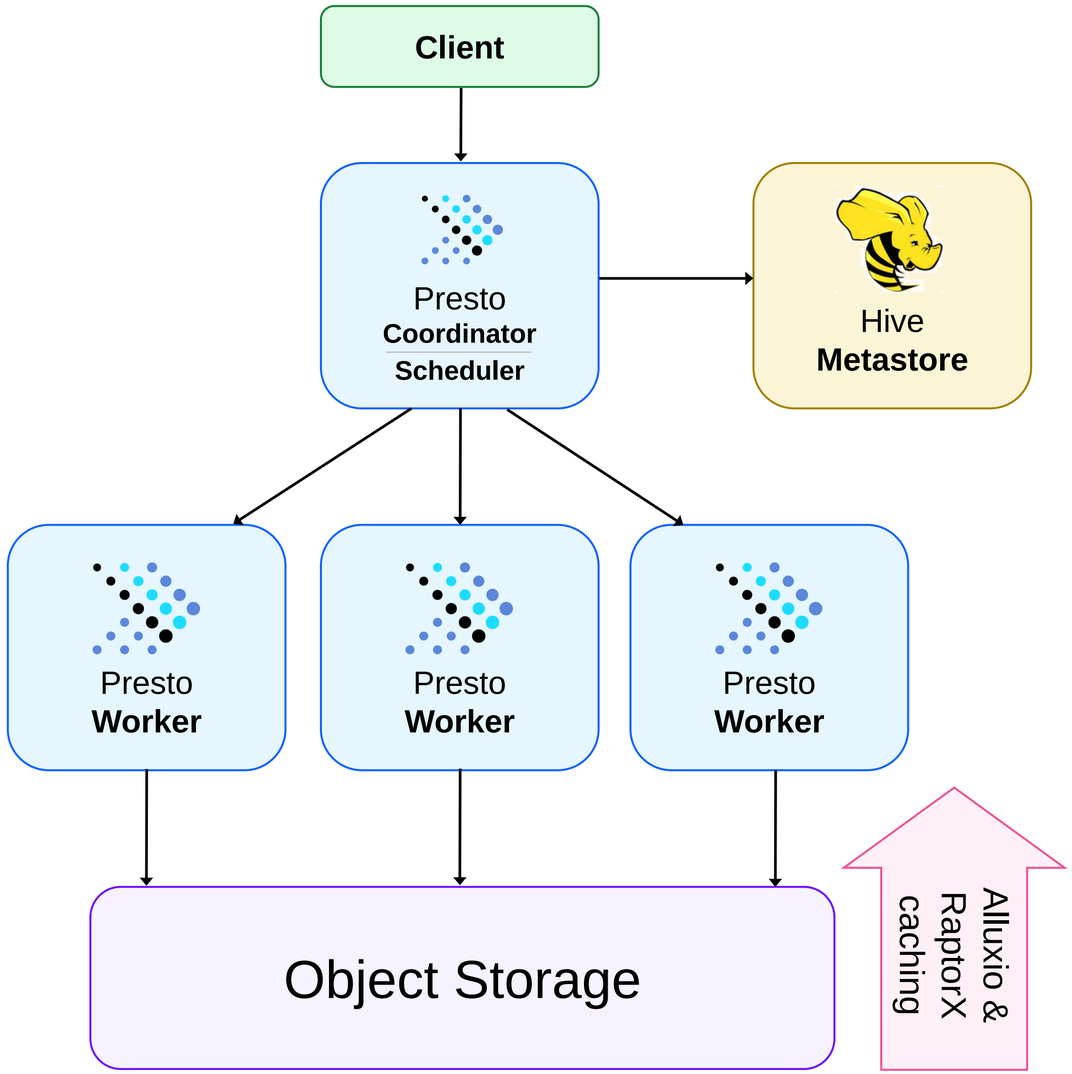
- Presto is an open-source distributed SQL engine suitable for querying large amounts of data
- Supports both relational and non-relational sources
- Easy to use with data analytics and business intelligence tools
- Supports both interactive and batch workloads
- In watsonx.data, spin up one or more Presto compute engines of various sizes – cost effective, in that engines are ephemeral and can be spun up and shut down as needed

- Presto connectors allow access to data in-place, allowing for **no-copy data access and federated querying**
- Consumers are abstracted from the physical location of data
- A wide variety of data sources are supported, including:

Presto architecture

The structure of Presto is similar to that of classical MPP database management systems.

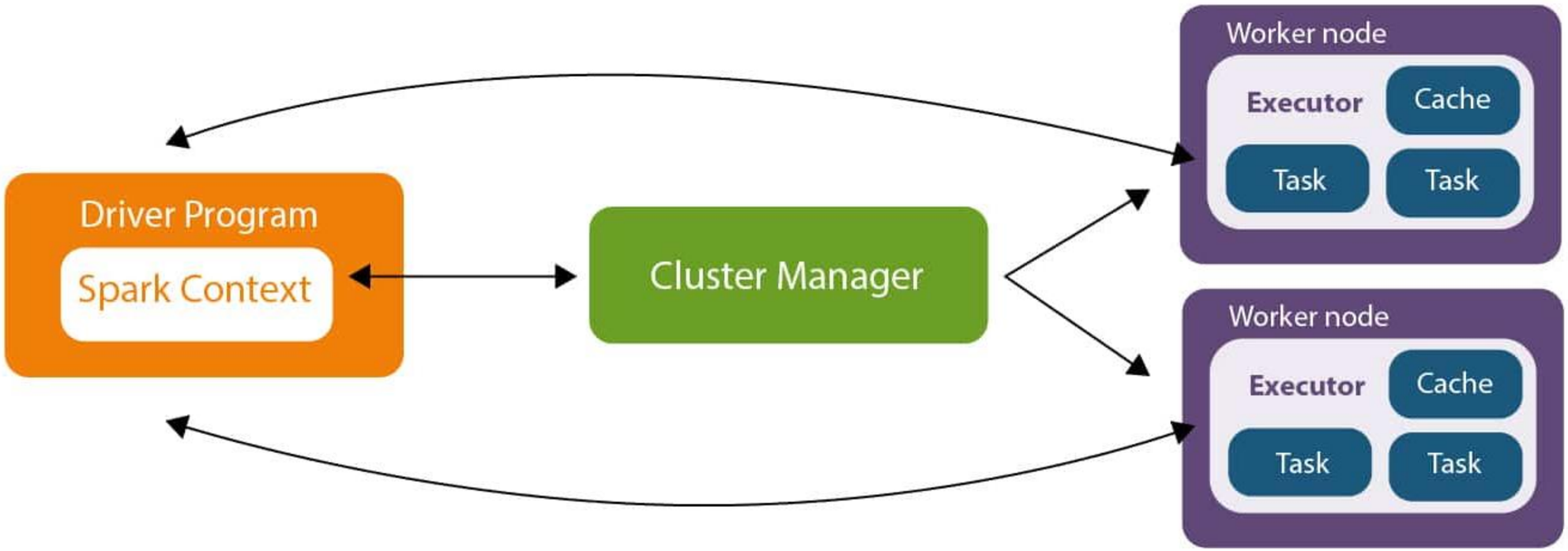
- **Client:** Issues user query and receives final result.
- **Coordinator:** Parses statement, plans query execution, and manages worker nodes. Gets results from workers and returns final result to client.
- **Workers:** Execute tasks and process data.
- **Connectors:** Integrate Presto with external data sources like object stores, relational databases, or Hive.
- **Caching:** Accelerated query execution through metadata and data caching (provided by Alluxio and RaptorX).



Apache Spark

Apache Spark is an open-source data-processing engine for large data sets. It is designed to deliver the computational speed, scalability, and programmability required for *big data*, specifically for streaming data, graph data, ML, and AI applications.

The basic Apache Spark architecture diagram:



Apache Spark and machine learning

Spark has libraries that extend the capabilities to ML, AI, and stream processing.

- Apache Spark MLlib
- Spark Streaming
- Spark SQL
- Spark GraphX