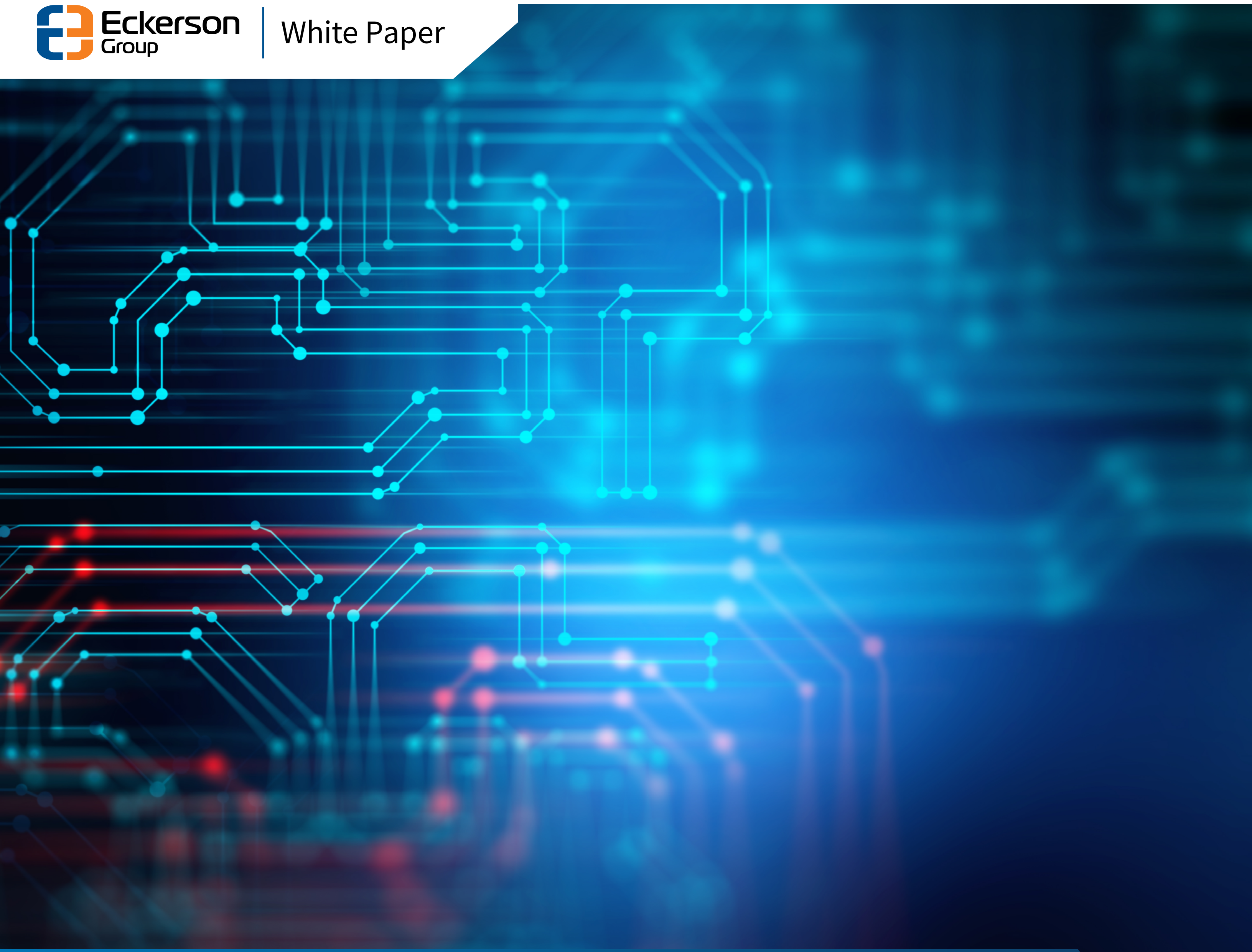
Eckerson Group | White Paper



The Definitive Guide to the Data Lakehouse

Must-Have Characteristics and Components

BY KEVIN PETRIE

JUNE 2023

RESEARCH SPONSORED BY DREMIO

dremio

About the Author

Kevin Petrie is the VP of Research at Eckerson Group, where he manages the research agenda and writes about topics such as data integration, data observability, machine learning, and cloud data platforms. For 25 years Kevin has deciphered what technology means to practitioners, as an industry analyst, instructor, marketer, services leader, and tech journalist. He launched a data analytics services team for EMC Pivotal in the Americas and EMEA and ran field training at the data integration software provider Attunity (now part of Qlik). A frequent public speaker and co-author of two books about data management, Kevin most loves helping startups educate their communities about emerging technologies.

About Eckerson Group

Eckerson Group RESEARCH CONSULTING ADVISORY GET • MORE • VALUE • FROM • YOUR • DATA

Eckerson Group is a global research and consulting firm that helps organizations get more value from data. Our experts think critically, write clearly, and present persuasively about data analytics. They specialize in data strategy, data architecture, self-service analytics, master data management, data governance, and data science. Organizations rely on us to demystify data and analytics and develop business-driven strategies that harness the power of data. **Learn what Eckerson Group can do for you!**

About This Report

This report is sponsored by Dremio, who has exclusive permission to syndicate its content.

Table of Contents

Executive Summary

The data lakehouse has captured the hopes of modern enterprises that seek to combine the best of the data warehouse with the best of the data lake. Like a data warehouse, it transforms and queries data at high speed. Like a data lake, it consolidates multi-structured data in flexible object stores. Together these elements can support both business intelligence (BI) and data science workloads. While early in the adoption cycle, many enterprises implement the data lakehouse to streamline their architectures, reduce cost, and assist the governance of self-service analytics. Common use cases include data mesh support, a unified access layer for analytics, data warehouse consolidation, data modernization for the hybrid cloud, departmental lakehouses, and support for FinOps programs.

This report explores use cases and case studies, then defines the must-have characteristics of the data lakehouse: unified, simple, accessible, fast, economic, governed, and open. It also examines the architectural layers of the data lakehouse environment, including the object store, a data layer, processing layer, semantic layer, communication layer, and client layer. Data teams that select the right elements for their environments and establish the right points of integration can modernize their data architecture for analytics and BI.

Take the following steps to build and execute the right strategy to modernize your open data stack.

- ***Prioritize your business use cases.*** *Define the highest priority business use cases for your data lakehouse. Identify which of these can get you a "quick win," and prioritize the architectural characteristics required to support them: unified, simple, accessible, high performance, economic, governed, or open.*
- ***Tackle your first project.*** *Plan and execute the first project to support your highest priority use case(s). Assemble the right stakeholders, then create and implement a roadmap for incrementally changing your environment.*
- ***Expand your lakehouse.*** *When you demonstrate business value with your first project, this "quick win" can give you the budget, executive support, and architectural platform to expand. Your goal is to chalk up a sequence of incremental, achievable projects that each demonstrate the ROI of your lakehouse.*

Rise of the Data Lakehouse

Enterprises have a new option to unify the worlds of business intelligence and data science: the data lakehouse.

The data lakehouse seeks to combine the structure and performance of a data warehouse with the flexibility of a data lake. It is a type of data architecture that uses data warehouse commands, often in structured query language (SQL), to query data lake object stores, on premises or in the cloud, at high speed. The data lakehouse supports both business intelligence (BI) and data science workloads by running queries against both relational data and multi-structured data stored as files. Some lakehouse platforms also provide a semantic layer that–among other things–consolidates virtual views of the underlying physical data, or even a communication and client layer.

Enterprises adopt the **data lakehouse** to simplify how they meet exploding business demand for analytics. They see two **primary benefits.**

> **Simpler architecture.** By running fast queries directly on the object store within the data lake, enterprises no longer have to copy or move data to meet BI performance requirements. This can reduce the need for data extracts and a data warehouse, which minimizes the pain of managing multiple copies and streamlines cost. This also can improve agility to support changing business requirements.

> **Workload consolidation.** By supporting both BI and data science, the data lakehouse can enable enterprises to consolidate workloads because they no longer need two distinct platforms. They still maintain an open architecture and open formats to interoperate with other tools.

Lakehouse platforms that consolidate data views into a semantic layer also can help simplify data access, enable self-service, and assist governance. Data analysts and data scientists can prototype new analytics approaches without having to copy or move data. This can reduce the burden on data engineers, freeing up their time for more innovative work.

Lakehouse platforms that consolidate data views into a semantic layer can help simplify data access and enable self-service

Architectural Overview

The lakehouse itself includes three core architectural layers: the object store, data layer, and processing layer.

> **The object store,** as the name suggests, comprises multi-structured data objects that reside in on-premises, hybrid, cloud, or multi-cloud infrastructure.

> **The data layer** includes data management (e.g., transformation and querying), table formats, file formats, and meta stores that centralize metadata.

> **The processing layer** includes functions, such as columnar processing, parallel processing, and caching, which help optimize queries.

Lakehouse environments also have various types of semantic, communication, and client layers. Some lakehouse platforms offer these layers natively. For example, Dremio offers a semantic layer and Snowflake and Databricks offer client tools such as ML libraries.

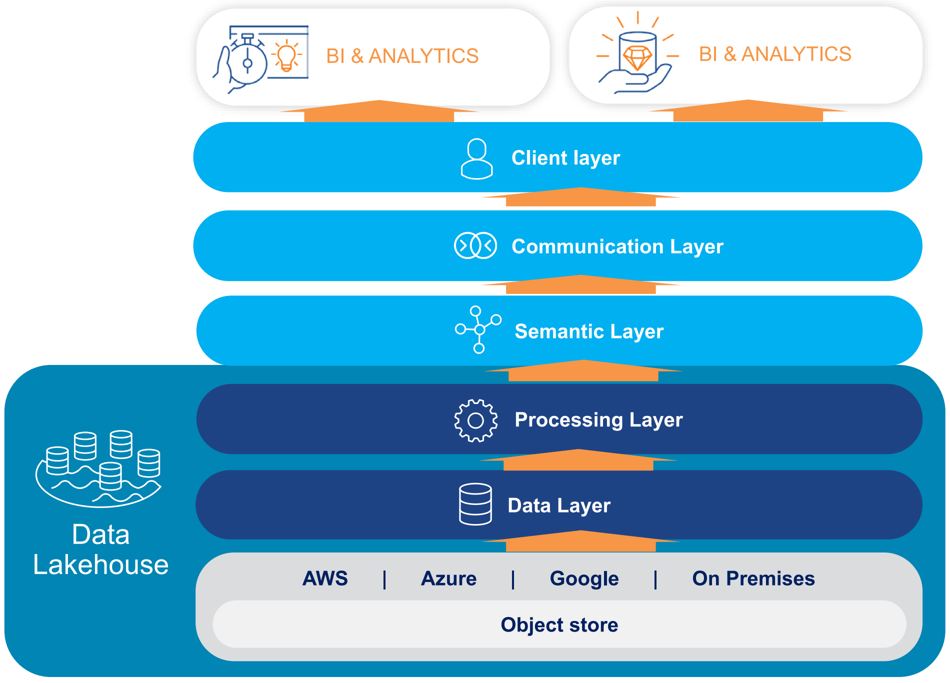
> **The semantic layer** combines capabilities such as virtual data views, a catalog, lineage, and other functions that help govern data.

> **The communication layer** comprises APIs such as REST or ODBC, as well as communication frameworks that client tools use to interact with the semantic layer.

> **The client layer** includes BI products, machine learning libraries, and other tools that analysts use to manipulate and visualize query results.

Figure 1 illustrates these architectural layers, which we'll explore in more detail later in this report.

Figure 1. The Data Lakehouse Environment



Designing an effective data lakehouse and integrating it into a heterogeneous enterprise environment requires careful planning and assessment of the elements involved. This report explores use cases and case studies, then defines the must-have characteristics and architectural components of the data lakehouse.

Architectural Use Cases

Enterprise data teams rely on the data lakehouse to address a variety of scenarios. Common use cases include a unified access layer for data mesh, data warehouse consolidation, data modernization for the hybrid cloud, and departmental data lakehouses.

Unified access layer for data mesh

The lakehouse offers a unified access layer for distributed datasets, helping analytics teams find and query data as part of a data mesh. This access layer remains consistent even as elements change, helping increase agility and simplify data management. The ecommerce company eMAG, for example, uses a lakehouse based on Dremio to help its business intelligence team access data within HDFS, Microsoft SQL Server, MySQL, and Amazon S3.

In fact, the data lakehouse supports all four pillars of the data mesh: domain ownership, data as a product, self-service, and federated governance. Domain owners can share data as a product for other data analysts and scientists to discover, inspect, and consume through a governed semantic layer. They can discover version-controlled data products in the catalog, inspect their lineage, then query virtual views of those products based on business context, all within the guardrails of role-based access controls. For example, **Shell** forecasts electricity demand by querying and applying ML techniques to different data products in a distributed data mesh, all accessed by a Dremio lakehouse platform.

Data warehouse consolidation

The data lakehouse offers an alternative to the sprawling data copies inherent to many data lake environments. Rather than replicating data from the lake into satellite data warehouses, BI extracts, or OLAP cubes for performance, enterprises can use the lakehouse to accelerate queries on the lake itself. Consolidating their environment in this fashion enables data teams to reduce duplicative copies, improving efficiency and assisting compliance. They can support more analytics projects and queries with fewer resources. The consumer-packaged goods firm Henkel implemented a lakehouse platform on Microsoft **Azure Data Lake Storage** (ADLS) to eliminate duplicative data silos, improving data quality to support supply chain analytics.

Data modernization for the hybrid cloud

The data lakehouse supports data modernization by providing fast and consolidated access to distributed data wherever it resides—and wherever it goes. Enterprises often modernize by migrating data from legacy data warehouses or Hadoop file systems (HDFS) to cloud object stores. They also might migrate to object storage on premises to meet security or sovereignty requirements that rule out cloud platforms. In either case, the lakehouse can support both the sources and targets of such migrations. It provides consistent query capabilities with higher performance than legacy engines such as Apache **Hive** or **Impala.**

Departmental data lakehouses

Individual business teams or units also can spin up departmental data lakehouses without replicating or migrating datasets. Marketing, trading, supply-chain management, ecommerce, or other types of divisions can create virtual views of shared datasets within the enterprise object store. They build their own business insights and support their own projects, while using the same query interface and semantic layer as other teams. For example, **AP Intego** (now part of Next Insurance), offers distinct views of consolidated datasets to its business users, engineers, and customers.

7 Must-Have Characteristics of the Data Lakehouse

These use cases require a data lakehouse with certain must-have characteristics. It must be unified, simple, accessible, high performance, economic, governed, and open. Let's explore these seven characteristics before defining the architectural components.

Unified

The data lakehouse must support both BI and data science use cases. For example, a data scientist might devise an ML model that predicts market prices and give that model to the data analyst so they can forecast future revenue scenarios. This requires a unified repository for data engineers, data analysts, and data scientists. They need to share views of data from sources such as operational databases, applications, IoT sensors, and social media feeds. They need to run multiple concurrent workloads on the same copy of data and share tools and outputs with one another.

Data engineers, data analysts, and data scientists need to share views of data.

Simple

The data lakehouse should automate the configuration and management of its various components to help data teams execute projects with less effort. This includes a graphical interface that helps data engineers and data analysts discover, transform, curate, and query data. Data analysts should be able to serve themselves rather than waiting on data engineers. They might need to peruse files within the data store, select one, and preview its contents before filtering or reformatting it for analytics—all via mouse clicks, drop down menus and popup windows rather than manual scripting. A managed service can further simplify things by minimizing software implementation and administration work.

Accessible

The data lakehouse should enable data analysts and data scientists to access data themselves rather than relying on data engineers. Self-service like this requires a catalog with intuitive views of metadata, including file attributes, lineage, and usage history. In addition, data analysts and data scientists need to access these data views without tripping over one another. That is, they need to create consistent data views that rely on the same underlying physical copy of data. A finance report might tabulate paid invoices to measure quarterly revenue and a sales report might tabulate signed contracts to measure bookings. Those reports must derive their distinct numbers from the same consistent records.

High performance

The data lakehouse should meet rigorous Service Level Agreements (SLAs) for key performance metrics. These include low latency to support short query response times, high throughput to query high volumes of data, and high concurrency to support many workloads. A dashboard for retail sales on Cyber Monday might require low latency to help sales leaders adjust market prices. A root-cause analysis of manufacturing defects, meanwhile, might require high throughput to analyze sufficient volumes of IoT sensor data. And both these projects might need to serve many concurrent users.

Economic

Like any other cloud data architecture, the data lakehouse needs to help enterprises control cost by using resources wisely and supporting FinOps objectives for cloud analytics projects. It should profile workloads prior to execution so users know how many compute cycles they will require, then automatically adjust processing methods along the way to streamline those workloads. It should provide the ability to scale elastic resources up or down, consuming storage and compute capacity as needed to meet changing workloads requirements. And that scalability must be economic. Like a hybrid car that goes quiet at the stoplight, the data lakehouse should avoid unnecessary compute cycles. Enterprises need economic capabilities like these because most analytics projects, ranging from quarterly financial reports to ad-hoc customer 360 analyses, entail bursts of processing. They should not have to pay for peak performance all the time.

Governed

Enterprises must govern data usage in order to reduce risks to data quality and ensure compliance with regulations such as the **General Data Protection Regulation (GDPR), California Consumer**

Privacy Act, and **Health Insurance Portability and Accountability Act (HIPAA).** This means avoiding unnecessary data copies that might undermine a "single source of truth." It means controlling user actions with role-based access controls, masking sensitive data, and tracking lineage. Finally, it means recording user actions in a comprehensive audit log. Guardrails like these enable data analysts to generate accurate reports and compliance officers to protect personally identifiable information (PII).

Open

The data lakehouse should integrate with the ecosystem of data stores, formats, processors, tools, APIs, and libraries that modern data teams need to innovate. It also should complement and interoperate with alternative cloud data architectures such as Azure Synapse Analytics without the risk of lock-in. Data teams need an open architecture in which they can change these elements as business requirements evolve. They should not have to write custom scripts in order to move data between object stores, switch processing engines, or import an ML algorithm.

An open architectural approach helps data analysts take iterative steps and provides them with the flexibility to bring multiple engines and their choice of tools directly to the data, leveraging open formats, and not the other way around. This minimizes the need for complex, insecure, or risky data moves and data copy proliferation. For example, analysts might enrich their BI projects with new datasets, query the enriched dataset, and then see whether they can speed performance by swapping in a different query engine—all while retaining the governance integrity of the datasets.

The data lakehouse should integrate with the ecosystem of data stores, formats, processors, tools, APIs, and libraries that modern data teams need to innovate.

Architecture and Components

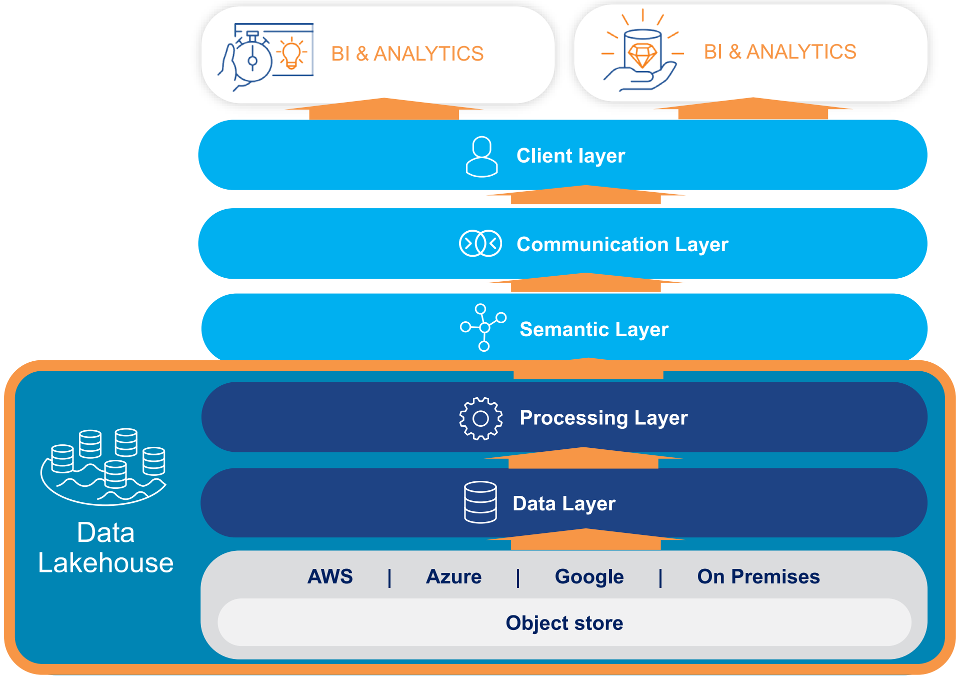
Now let's explore the architectural layers of a data lakehouse environment and elements that support those must-have characteristics. We'll work from the bottom up in our earlier architecture diagram.

Data Lakehouse

As mentioned earlier, the core architectural layers of the lakehouse are the object store, data layer, and processing layer.

Object Store

This stores all types of data objects—tables, flat files containing plain text, images, you name it—and classifies them for querying with identifiers and metadata. Object stores find a natural home on the cloud, whose elastic storage capacity enables enterprises to scale up or down as needed. Many enterprise data teams now favor object stores such as Amazon Simple Storage Service **(Amazon S3), Azure Blob Storage, Google Cloud Storage,** and **Dell ECS** because they offer a unified and economic



repository for structured, semi-structured, and unstructured data. Enterprise data teams should consider consolidating data onto cloud object stores as part of any data modernization initiative, provided they select open formats to avoid lock-in.

Data Layer

The data layer includes data management, file formats, table formats, and metastores.

Data management

The data layer receives commands (often in SQL) from various client tools to perform data management tasks such as transformation, querying, and quality checks. It might transform data by reformatting it, merging columns or tables, or structuring it with a schema. The data layer also queries data and presents views to BI tools, and performs data quality checks (accuracy, completeness, consistency, etc.) to support data observability tools such as Monte Carlo or Acceldata.

File formats

Many common file formats, such as comma-separated values (CSV), optimized row columnar (ORC) (for Apache Hive data), **Apache Parquet,** Avro, and JavaScript object notation **(JSON),** interoperate with various tools, processors, and data stores to enable an open architecture. By selecting these formats, enterprise data teams can maintain data portability and flexibility to adapt with the needs of the business. They should avoid converting to proprietary formats that cloud infrastructure providers recommend. While those proprietary formats might enable platform-specific enhancements such as higher performance, they risk locking enterprises into that provider's infrastructure, thereby reducing future portability.

Table formats

The **Apache Iceberg** open table format helps the data lakehouse run fast, concurrent queries on large data volumes. Iceberg maintains metadata that helps datasets evolve and support multiple workloads without having them interfere with one another. For example, it adapts to schema changes such as added or dropped columns. Iceberg partitions data, and updates those partitions as datasets evolve, to help speed queries by avoiding unnecessary reads. It also enables time travel so users can roll back versions to correct issues or run queries on historical snapshots. For example, a data analyst might want to compare query results for 2019 and 2022 to gauge the impact of COVID on customer spending habits.

The Apache Iceberg open table format helps the data lakehouse run fast, concurrent queries on large data volumes.

Delta Lake, also an open-source table format, provides similar capabilities to enable fast, concurrent query workloads on a large dataset. Delta Lake maintains transactional consistency, supports schema evolution, and enables time travel. **Apache Hudi** also offers similar capabilities, along with stream processing and data pipeline management. While each of these open table formats has a slightly different workload focus, enterprise data teams can choose any of them today and still swap out in the future. These open-source table formats help make the lakehouse unified, accessible, and fast.

Metastores

The metastore centralizes the metadata that data engineers need to ingest, transform, and reconcile different versions of data to ensure data quality. For example, the open-source **Project Nessie,** on which Dremio Arctic is based, offers a metastore that helps data engineers apply DataOps techniques such as code branching and version control to datasets. Based on the metadata Nessie centralizes, data engineers can view, branch, and merge datasets in an approach akin to GitHub for software development. Nessie supports "extract, transform, and load" (ETL) or ELT sequences for both batch and streaming data pipelines.

Project Nessie centralizes metadata to help data engineers apply DataOps techniques such as code branching and version control to datasets.

The **AWS Glue** data pipeline also offers a metastore to help data engineers organize and view data so they can ingest and transform it in ETL pipelines that feed Amazon S3 data lakes. In addition, the **Apache Hive** metastore can find and manage metadata for similar purposes in a Hive data warehouse. Of these options, Nessie and Hive metastores offer the more open approaches. Metastores help simplify data management and make data accessible to various users while still governing it.

Processing Layer

The processing layer includes a mix of functions that focus precious compute cycles on the data that matters for a given query. The price of cloud compute makes streamlined processing mandatory for any hybrid, cloud, or multi-cloud environment. Lakehouse platforms offer processing capabilities such as the following.

> **Columnar processing.** By arranging tabular data in columns rather than rows, the lakehouse platform can read only the attributes that relate to a query. Many enterprises use **Apache Arrow,** for example, to support columnar processing. The **Apache Spark** distribution now includes Arrow, as do the **MATLAB** data science tool and **Dremio** lakehouse platform. Other types of cloud data architectures, such as **Google BigQuery** and **Vertica,** also support columnar processing.

> **Caching.** Some platforms identify frequently-queried tables or records and pre-fetch those into cache to prepare for the next query.

> **Query pre-computation.** Some platforms prepare for frequent queries by pre-assembling columns and pre-computing aggregate values.

> **Workload isolation.** Isolating a workload on one or more compute nodes minimizes resource contention, making that workload faster and more predictable.

> **Elastic engines.** The platform should provision flexible "engines," or allotments of parallel compute nodes that scale up or down to support a workload within configurable thresholds.

> **Cost-based optimization.** By profiling workloads and automatically adjusting how it handles tasks—

such as the order in which it joins tables—the platform can streamline compute cycles and thereby reduce costs. It also should measure these cycles by workload and user group to support chargeback.

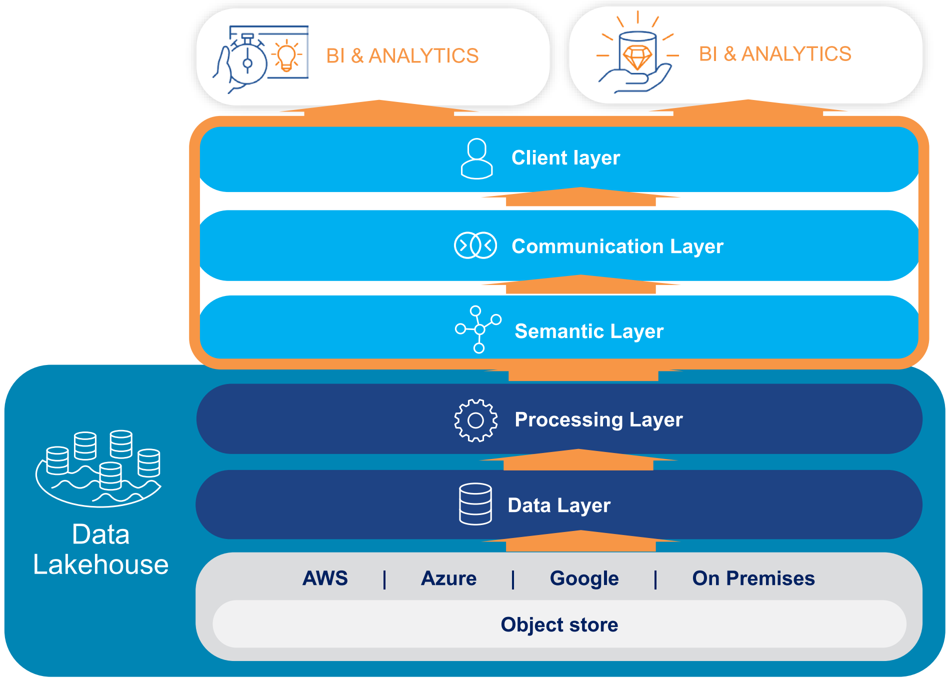
> **HW optimized processing.** Some platforms optimize tasks such as data sorting and filtering for specific runtime environments. For example, **Apache Gandiva** helps do this for highly parallel workloads on modern CPUs.

> **Compression.** By modifying data structures to consume less disk space, compression frees up valuable capacity, which improves efficiency and reduces cost.

Capabilities like these enable enterprises to maintain or improve query response times while restricting the number of on-premises servers or cloud compute nodes they need underneath. Data teams should seek out and apply these capabilities to ensure they can support use cases such as rapid reporting and ad-hoc querying of large data volumes. They need a lakehouse platform that is both fast and economic.

Additional Architectural Layers

The additional architectural layers of the lakehouse environment are the semantic, communication, and client layers.



Semantic Layer

The semantic layer presents business-oriented data views to data analysts and data scientists, helping them serve themselves and reduce the burden on data engineers. It consolidates these views of one or more object stores –or relies on semantic layers within BI tools–helping ensure everyone has the same version of the truth. The semantic layer includes capabilities such as a catalog, virtual data views, lineage, role-based access controls, and data masking. Here is what this looks like as part of a lakehouse platform.

> **Catalog.** The catalog centralizes metadata to create an indexed, searchable inventory of data assets, including tables, files, schema, and queries. This enables data analysts to find the assets they need, while still helping data curators and compliance officers manage data quality and control data access.

> **Virtual data views.** Because users often need many different views of the same physical dataset, the semantic layer presents virtual views that are transformations of physical data underneath.

> **Lineage.** The lineage of a dataset describes its origins, who touches it, and how it changes over time, so users can validate data quality and identify the cause of issues.

> **Role-based access controls.** As with any enterprise software platform, the semantic layer authenticates the identities of users and authorizes the actions they take, according to user type, data asset, and category of action. It also integrates with third-party identity management tools.

> **Data masking.** Finally, the semantic layer masks sensitive data such as customers' personally identifiable information (PII). Masking replaces selected data with dummy values that support queries while still preventing bad actors from compromising real data.

These capabilities of the semantic layer make the lakehouse more unified, accessible, and governed. Data engineers can transform and organize multi-sourced data in the object store. Data analysts can serve themselves by creating virtual data views and running queries to feed their BI projects. Data stewards and compliance officers, meanwhile, can control and oversee teams' activities. Data teams should be sure to implement capabilities like these, within either the lakehouse platform or a larger enterprise governance offering such as Privacera or BigID.

Communication Layer

To interact with various client tools, the lakehouse relies on a communication layer that comprises APIs and data transfer frameworks.

Application programming interfaces (APIs)

Data analysts, data scientists, and data engineers access data through common APIs, such as representational state transfer (REST), SQL, Java database connectivity (JDBC), and open database connectivity (ODBC). For example, a data analyst might use the REST API to connect their BI tool to the data lakehouse, submit queries, and view results. A data scientist, meanwhile, might use a JDBC API to export query results from the data lakehouse to a data science platform. In addition, various other components, such as Amazon S3 and Apache Iceberg, use their own APIs to interoperate with one another.

Enterprises should seek out APIs like these in order to make their data lakehouse open and accessible. They should avoid proprietary APIs—for example, from cloud infrastructure providers—that offer provider-specific enhancements but reduce data portability. There is also the risk of "creeping lock-in," whereby vendors offer APIs that become less open over time.

Data transfer frameworks

Data teams also need to maintain performance levels as they apply various processors to a large dataset or transfer high volumes of data between repositories. Data transfer frameworks, most notably **Apache Arrow Flight,** can help. Arrow Flight offers a high-speed alternative to APIs such as ODBC and JDBC by eliminating the need to serialize and deserialize data as it passes between architectural components. For example, a data analyst might use Arrow Flight to present higher volumes of data to their BI tool or client than would otherwise be possible. A data scientist, meanwhile, might use Arrow Flight to transfer higher volumes of data from the data lakehouse to a data science platform such as **Domino Data Lab** or **DataRobot.** The communication framework, especially Arrow Flight, helps unify and accelerate the data lakehouse.

Arrow Flight offers a high-speed alternative to APIs such as ODBC and JDBC by eliminating the need to serialize and deserialize data as it passes between components of a data architecture

Client Layer

The client layer includes tools for BI projects and data science projects.

Tools for BI projects

Data teams access and manipulate data for BI projects using three types of client tools. First, data analysts and some data scientists use BI tools such as **Tableau, Looker,** and **Qlik** to query the virtual data views within the semantic layer and visualize the results. They also use open-source tools such as **Apache Superset** for data exploration and visualization. Second, data analysts, developers, and data scientists use client tools such as **DbVisualizer** and **DataGrip** to write and manage commands. Third, developers use workflow tools such as **Airflow** to orchestrate analytical and operational workflows that integrate with query results.

A given BI project might include all three types of tools. A developer might write a complex query in DbVisualizer and pass it over to a data analyst who runs that query and visualizes the results in Tableau. They feed the results back to the developer, who integrates them into application workflows using Airflow. This type of scenario would support various use cases, such as the personalization of web pages for customers based on their most recent purchases and social media posts.

Tools for data science projects

Data scientists use two types of client tools as they manage data science projects. First, they download algorithms from AI/ML libraries such as **PyTorch** or **TensorFlow,** or use tools within those libraries to develop and train models. Second, they transform data, visualize data, and develop models in AI/ML notebooks such as **Jupyter.** AI/ML libraries and notebooks must both integrate easily with the data lakehouse as data scientists move code or data back and forth. Data science projects, iterative by nature, need these tools to interoperate and exchange data with minimal effort.

Conclusion - Defining Your Approach

The data lakehouse can simplify how enterprises meet business demands, foster analyst collaboration, and reduce effort. Data teams that select the right components for their environment and establish the right points of integration can start to redefine how their business uses analytics. They can support new and existing use cases on time and within budget. Of course, this means building and executing on the right plan. As a data leader, you should take the following steps to increase your odds of success.

> **Prioritize your business use cases.** Define the highest priority business use cases for your data lakehouse. These business use cases might include periodic reporting; interactive reports and dashboards; ad-hoc queries, 360-degree customer views; or artificial intelligence/machine learning (AI/ML). Identify which of these can get you a “quick win,” and prioritize the architectural characteristics required to support them: unified, simple, accessible, high performance, economic, governed, or open.

> **Tackle your first project.** Plan and execute the first project to support your highest priority use case(s). Assemble the right stakeholders, including an executive sponsor, data analyst or scientist, data engineer, architect, and governance manager. Then have this team create and implement a roadmap for incrementally changing your environment. For example, to support 360-degree customer views, your team might migrate semi-structured customer data from HDFS to a cloud object store in the target lakehouse.

> **Expand your lakehouse.** When you demonstrate business value with your first project, this “quick win” gives you the budget, executive support, and architectural platform to expand. You might plan and execute a second project that migrates other functional data—perhaps finance or supply-chain records—from HDFS to your lakehouse. Alternatively, you might extend the unified access layer to support legacy databases on premises. This can support a data mesh in which business domain owners publish data products to self-service users throughout the business. In these or other scenarios, your goal is to chalk up a sequence of incremental, achievable projects that each demonstrate the ROI of your lakehouse.

About Eckerson Group

Wayne Eckerson, a globally-known author, speaker, and consultant, formed **Eckerson Group** to help organizations get more value from data and analytics. His goal is to provide organizations with expert guidance during every step of their data and analytics journey.

Eckerson Group helps organizations in three ways:

> **Our thought leaders** publish practical, compelling content that keeps data analytics leaders abreast of the latest trends, techniques, and tools in the field.

> **Our consultants** listen carefully, think deeply, and craft tailored solutions that translate business requirements into compelling strategies and solutions.

> **Our advisors** provide one-on-one coaching and mentoring to data leaders and help software vendors develop go-to-market strategies.

Eckerson Group is a global research and consulting firm that focuses solely on data and analytics. Our experts specialize in data governance, self-service analytics, data architecture, data science, data management, and business intelligence.

Our clients say we are hard-working, insightful, a   nd humble. It all stems from our love of data and our desire to help organizations turn insights into action. We are a family of continuous learners, interpreting the world of data and analytics for you.

Get more value from your data. Put an expert on your side. **Learn what Eckerson Group can do for you!**

Eckerson Group

RESEARCH CONSULTING ADVISORY

GET • MORE • VALUE • FROM • YOUR • DATA

About Dremio

Dremio is the open and easy data lakehouse, providing self-service SQL analytics, data warehouse performance and functionality, and data lake flexibility across all of your data. Hundreds of organizations, including 3 of the Fortune 5, use Dremio to deliver mission-critical BI on the data lake. As the original creator of Apache Arrow, Dremio is on a mission to reinvent SQL for data lakes and meet customers where they are in their cloud journey. Dremio was founded in 2015 and is headquartered in Santa Clara, CA.