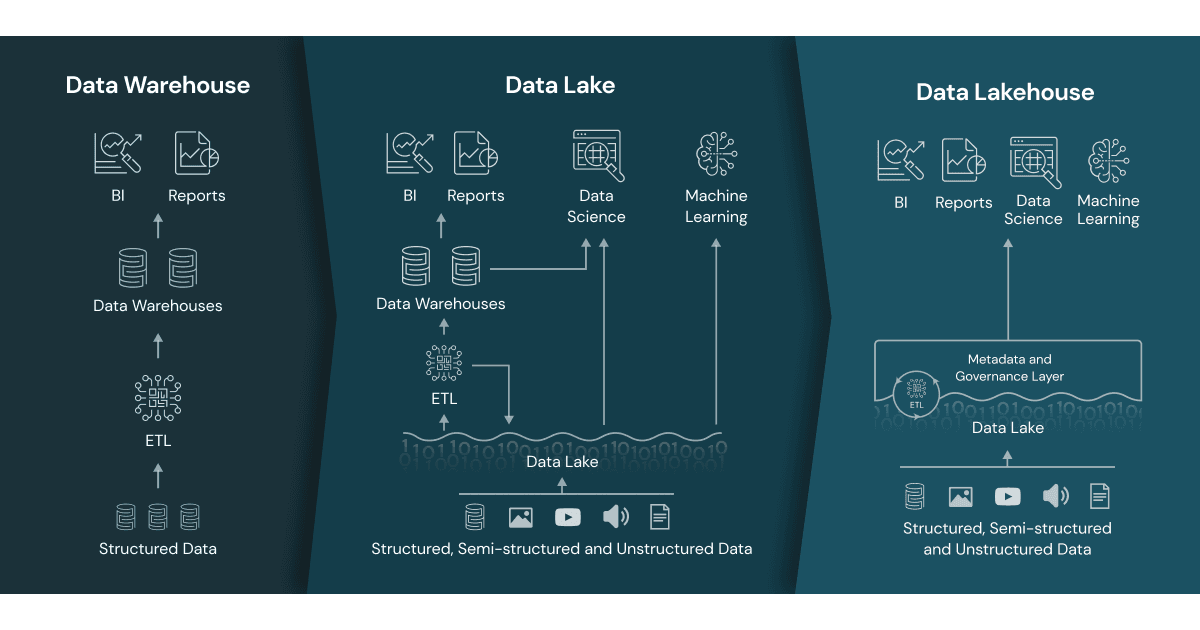
### **What is a Data Lakehouse?**

A data lakehouse is a new, open data management architecture that combines the flexibility, cost-efficiency, and scale of [**data lakes**](https://databricks.com/discover/data-lakes/introduction) with the data management and ACID transactions of data warehouses, enabling business intelligence (BI) and machine learning (ML) on all data.

### **Data Lakehouse: Simplicity, Flexibility, and Low Cost**

Data lakehouses are enabled by a new, open system design: implementing similar data structures and data management features to those in a data warehouse, directly on the kind of low-cost storage used for data lakes. Merging them together into a single system means that data teams can move faster as they are able to use data without needing to access multiple systems. Data lakehouses also ensure that teams have the most complete and up-to-date data available for data science, machine learning, and business analytics projects. 

### **Key Technology Enabling the Data Lakehouse**

There are a few key technology advancements that have enabled the data lakehouse:

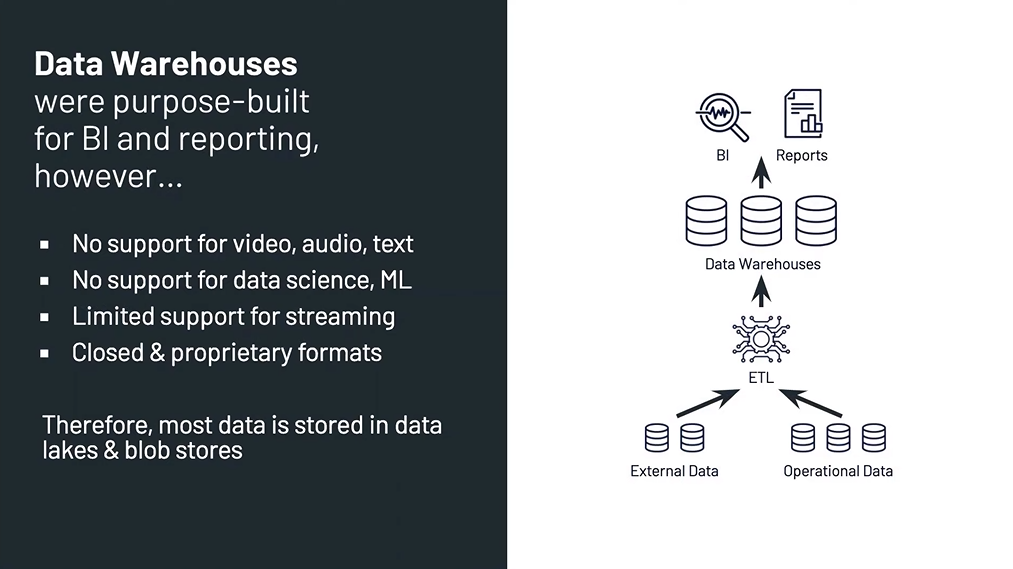
* metadata layers for data lakes
* new query engine designs providing high-performance SQL execution on data lakes
* optimized access for data science and machine learning tools.

**The Fundamentals of Data Warehouse + Data Lake = Lake House**

# Introduction

With the evolution of Data Warehouses and Data Lakes, they have certainly become more specialized yet siloed in their respective landscapes over the last few years. Both data management technologies each have their own identities and are best used for certain tasks and needs, however they also struggle in providing some important abilities. Data Warehouse advantages are focused around analyzing structured data, OLTP, schema-on-write, SQL, and delivering ACID-compliant database transactions. Data Lake advantages are focused around analyzing all types of data (structured, semi-structured, unstructured), OLAP, schema-on-read, API connectivity, and low-cost object storage systems for data in open file formats (i.e. Apache Parquet).

Notably, Data Warehouses particularly struggle with support for advanced data engineering, data science, and machine learning. For example, their inability to store unstructured data (i.e. text, images, video, feature engineering vectors, etc.) for machine learning development. In addition, proprietary Data Warehouse software are expensive and struggle with integrating open source + cloud platform data science and data engineering tools (i.e. Python, Scala, Spark, SageMaker, Anaconda, DataRobot, SAS, R, etc.) for exploratory data analysis via notebooks, distributed compute processing, hosting deployed models, and storing inference pipeline results. System integration, data movement costs, and data staleness will even become more challenging to address in a hybrid on-premise cloud environment.



On the flip side, unfortunately, Data Lakes sometimes notoriously struggle with data quality, transactional support, data governance, and query performance issues. Data Lakes built without vital skills, key capabilities, and specialized technologies will inevitably over time turn into “Data Swamps”. This can be a tough situation to revert especially if the data volume and velocity continue to increase. Avoiding this dilemma is absolutely critical for achieving data-driven value and providing customer satisfaction to users who are dependent on having reliable fast data retrieval to perform their downstream analytics job duties for their stakeholders.

# Databricks Lakehouse

**Fundamentals**

Databricks Lakehouse is centered around a technology named Delta Lake, an open source project managed by the Linux Foundation. Delta Lake is a storage layer via Apache Parquet format that provides ACID-compliant transactions and additional benefits to Data Lakes. Databricks mentions 9 common Data Lake challenges Delta Lake can help address. They are:



* Hard to append data
* Jobs fail mid-way
* Modifications of existing data is difficult
* Real-time operations
* Costly to keep historical versions of data
* Difficult to handle large metadata
* “Too many files” problems
* Hard to get great performance
* Data quality issues

**Building Blocks**

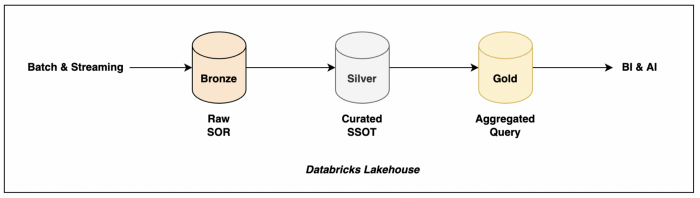
Databricks Lakehouse powered by Delta Lake contains some key internals designed to ensure data reliability and consistency. They are:

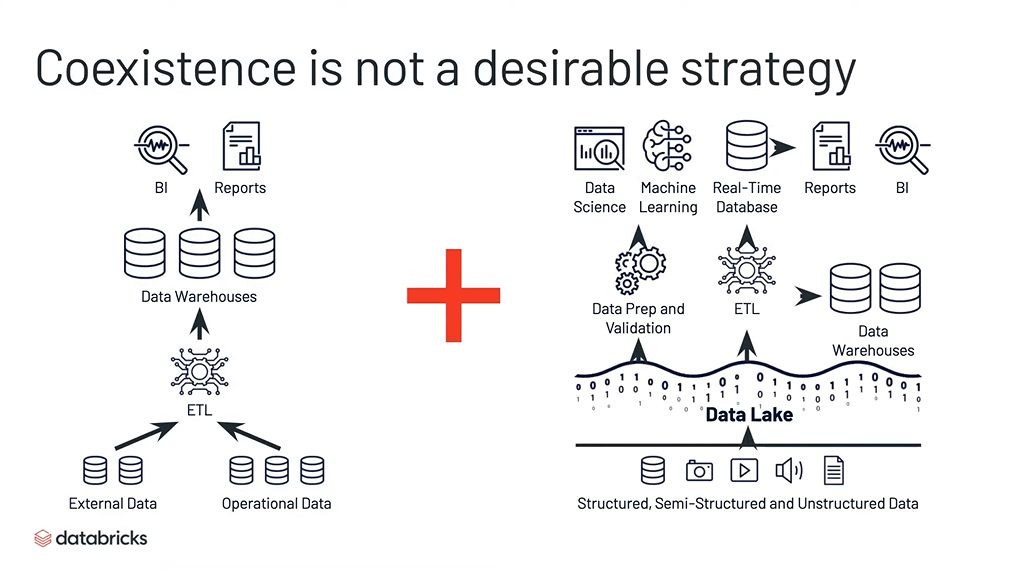
* Delta tables
* Delta files
* Delta transaction log
* Delta engine
* Delta storage layer

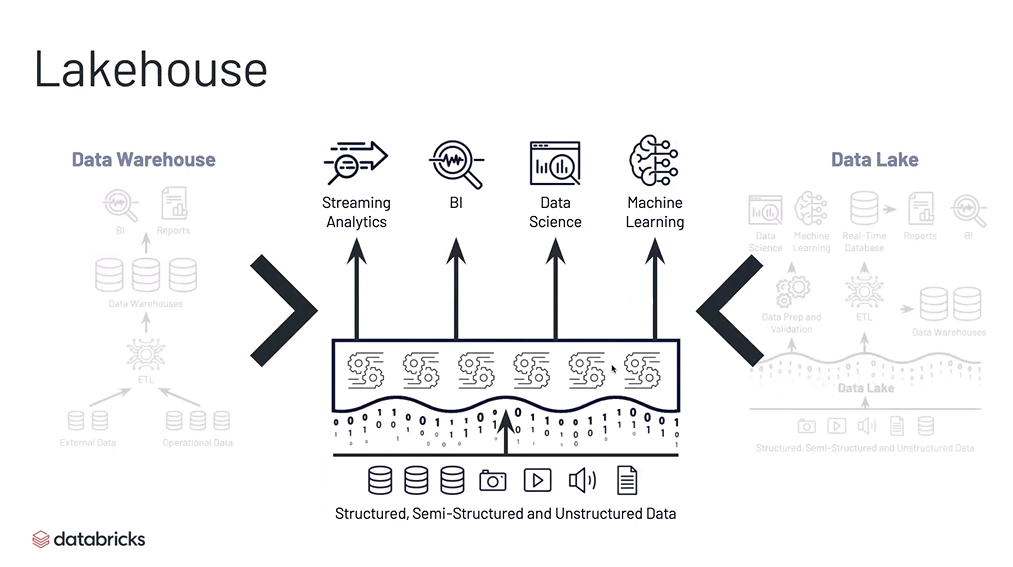
**Solutions Architecture**

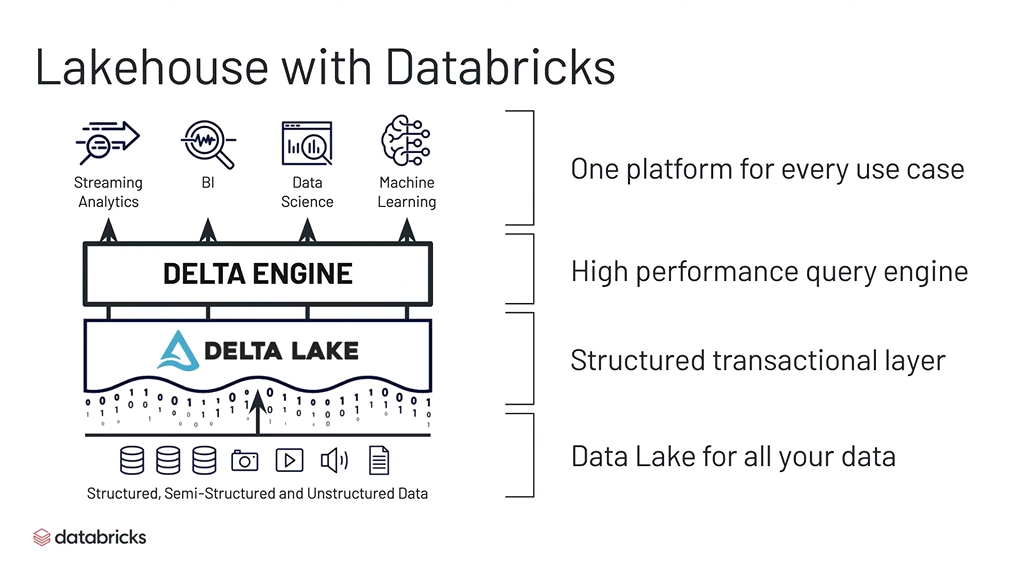
Databricks Lakehouse follows a design pattern architecture delivering multi-layers of data quality and curation via a 3 table tier nomenclature. They are:

* Bronze
* Silver
* Gold









# Amazon Web Services Lake House

**Fundamentals**

AWS Lake House is focused around using many of the AWS Analytics services in tandem. Specifically, integrating these specialized services to build seamless interaction between Data Lake, Data Warehouse, and the data movement between systems. Hence, AWS Lake House stresses how important it is to ensure data movement between AWS services become easier by design. Therefore, AWS describes the 3 data movement scenarios as:

* Inside-out
* Outside-in
* Around-the-perimeter

For example, inside-out refers to collecting data from an internal Data Lake and copying it over to another system.

Similarly, outside-in refers to the opposite direction of transferring data from an external Data Warehouse to an internal Data Lake or file system (i.e. performing feature engineering in Amazon S3 & Amazon SageMaker). Sometimes, external data stores replicate data around-the-perimeter to scale better performance (i.e. Amazon RDS to Amazon Redshift cluster migration) and or improve the user experience. There are many customer use cases for all 3 of these data movement scenarios.

AWS refers to data growth and data movement difficulty as “data gravity”, which is basically related to the “Big Data” phrase where data presence and increasing volumes occur across many IT systems. Resulting in the need for a data architecture capable of meeting customer demand, infrastructure management costs, data locality performance, and data & system integration. AWS documents their Well-Architected framework of 5 pillars as:

* Operational Excellence
* Security
* Reliability
* Performance Efficiency
* Cost Optimization

These pillars are the foundation to achieving a data architecture that provides continuous scalable performance, flexible service tool choices, data & system integration, secure data protection, and infrastructure elasticity. A Lake House architecture starts with adopting these fundamental best practices.

**Building Blocks**

AWS Lake House has 5 elements of data architecture. They are:

* Scalable data lakes
* Purpose-built data services
* Seamless data movement
* Unified governance
* Performant and cost-effective

Each element serves a purpose via specialized AWS services to deliver a resilient, scalable, elastic, secure, and flexible architecture. AWS Lake Formation is the staple for building secure and scalable Data Lakes via Amazon S3. Some new Lake Formation features include ACID transactions and governed tables to address faster query performance via file compaction methods. These are very similar features like Delta Lake.

**Architecture**

AWS Lake House follows an ecosystem architecture via 5 layers that address data gravity using specialized AWS services stationed on the periphery of a centralized Data Lake. These flexible layers are:

* Ingestion layer
* Storage layer
* Catalog layer
* Processing layer
* Consumption layer

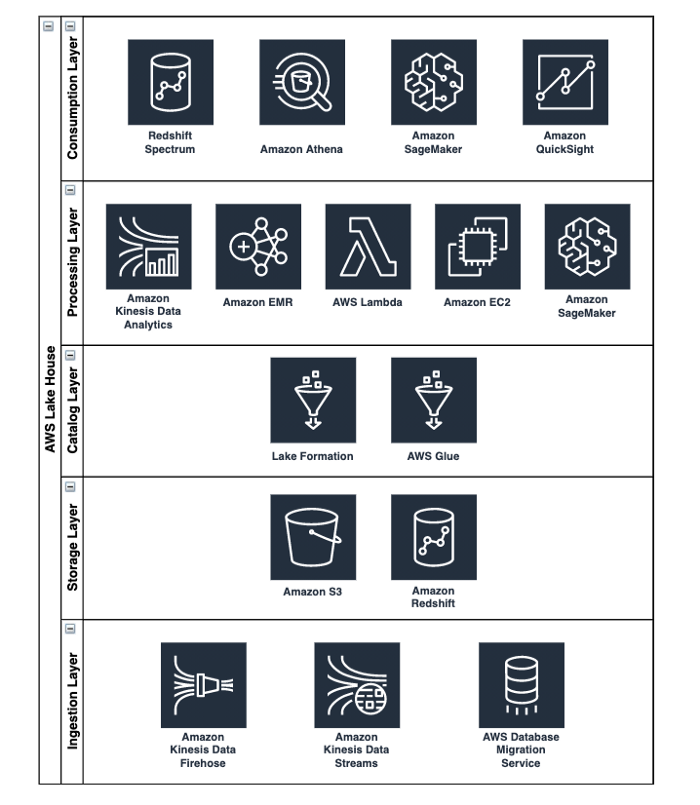


Image by Author

Each of these 5 layers play their own part to enabling a seamless flow of data movement within a Lake House. The key is the technical flexibility available to iteratively scale and adjust accordingly by using the right service for the right job at an affordable (pay for what you use) price.

# Conclusion

Overall, both architectures provide very similar solutions. From a cultural perspective Databricks Lakehouse and AWS Lake House are revolutionizing better ways to manage, store, process, and consume data. From a technical perspective Databricks Lakehouse utilizes Delta Lake, Apache Parquet, and Apache Spark; AWS Lake House utilizes a plethora of AWS services leveraging a lot of the open source Apache projects mentioned throughout this blog. Some of the main technical focal points include:

1. Separation of compute and storage — elastic compute, scalable infrastructure, cheap storage, and highly available reusable data
2. Reliability & consistency — improved data quality, less corrupt & duplicate data, file size optimization & compaction, less schema/metadata mismatches, and guaranteed complete transactions & rollback options
3. Real-time streaming support — continuous/immediate incremental delta updates & changes as new data arrives while maintaining great query performance, joining of batch and streaming data across lines of business, removal of complex workflows (i.e. Lambda Architecture → parallel data pipelines of a streaming job that continuously appends + batch job that re-partitions, updates, and overwrites files per some cadence .