Building Reliable Data Lakes at Scale With Delta Lake

Contents

Drivers

Data Engineering Drivers

Data engineering professionals are needing to respond to several different drivers. Chief among the drivers they face are:

Rise of Advanced Analytics — Advanced analytics, including methods based on machine learning techniques, have evolved to such a degree that organizations seek to derive far more value from their corporate assets.

Widespread Adoption — Once the province of leading edge, high-tech companies, these advanced approaches are being adopted across a multitude of industries from retail to hospitality to healthcare and across private as well as public sector organizations. This is further driving the need for strong data engineering practices.

Regulation — With the growth of data generation and data collection, there is increased interest in how the data is protected and managed. Regulatory regimes such as GDPR (General Data Protection Regulation) from the EU and other jurisdictions mandate very specific ways in which data must be managed.

Drivers

Technology Innovation — The move to cloud-based analytics architectures that is now well underway is being propelled further by innovations such as analytics-focused chipsets, pipeline automation and the unification of data and machine learning. All these offer data professionals new approaches for their data initiatives.

Financial Scrutiny — With a growth in investment, analytics initiatives are also subject to increasing scrutiny. There is also a greater understanding of data as a valuable asset. Deriving value from data must be done in a manner that is financially responsible and actually value adding to the enterprise and meeting ROI hurdles.

Role Evolution — Reflecting the importance of managing the data and maximizing value extraction, the Chief Data Officer (CDO) role is becoming more prominent and newer roles such as Data Curator are emerging. They must balance the needs of governance, security and democratization.

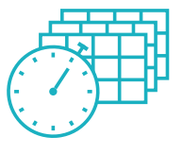
Key Goals

Data Pipeline Key Goals

Making quality data available in a reliable manner is a major determinant of success for data analytics initiatives be they regular dashboards or reports, or advanced analytics projects drawing on state-of-the-art machine learning techniques. Data engineers tasked with this responsibility need to take account of a broad set of dependencies and requirements as they design and build their data pipelines.

Three primary goals that data engineers typically seek to address as they work to enable the analytics professionals in their organizations are:

Deliver quality data in less time — When it comes to data, quality and timeliness are key. Data with gaps or errors (which can arise for many reasons) is "unreliable," can lead to wrong conclusions, and is of limited value to downstream users. Equally well, many applications require up-to-date information (who wants to use last night's closing stock price or weather forecast) and are of limited value without it.



Enable faster queries — Wanting fast responses to queries is natural enough in today's "New York minute," online world. Achieving this is particularly demanding when the queries are based on very large data sets.

Simplify data engineering at scale — It is one thing to have high reliability and performance in a limited, development or test environment. What matters more is the ability to have robust, production data pipelines at scale without requiring high operational overhead.

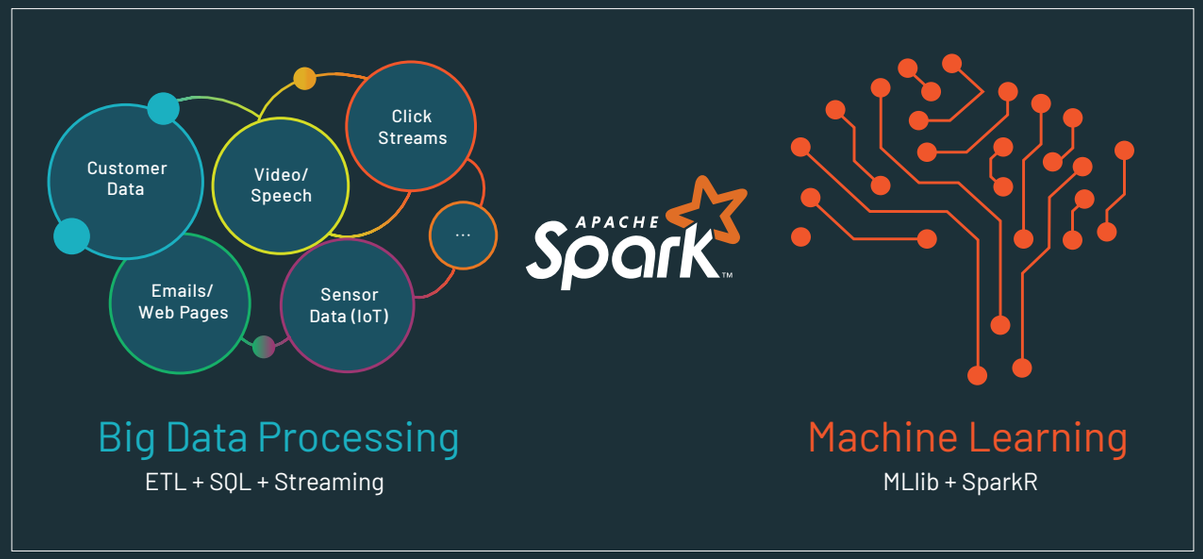
Apache Spark™

DELTA LAKE

Apache Spark™: The First Unified Analytics Engine

Originally developed at UC Berkeley in 2009, Apache Spark can be considered the first unified analytics engine. Uniquely bringing data and AI technologies together, Spark comes packaged with higher-level libraries, including support for SQL queries, streaming data, machine learning and graph processing. These standard libraries increase developer productivity and can be seamlessly combined to create complex workflows.

Since its release, Apache Spark, has seen rapid adoption by enterprises across a wide range of industries. Internet powerhouses such as Netflix, Yahoo and eBay have deployed Spark at massive scale, collectively processing multiple petabytes of data on clusters of over 8,000 nodes making it the de facto choice for new analytics initiatives. It has quickly become the largest open source community in big data, with over 1000 contributors from 250+ organizations.



While Spark has had a significant impact in taking data analytics to the next level, practitioners continue to face data reliability and performance challenges with their data lakes.

Data Reliability Challenges With Data Lakes

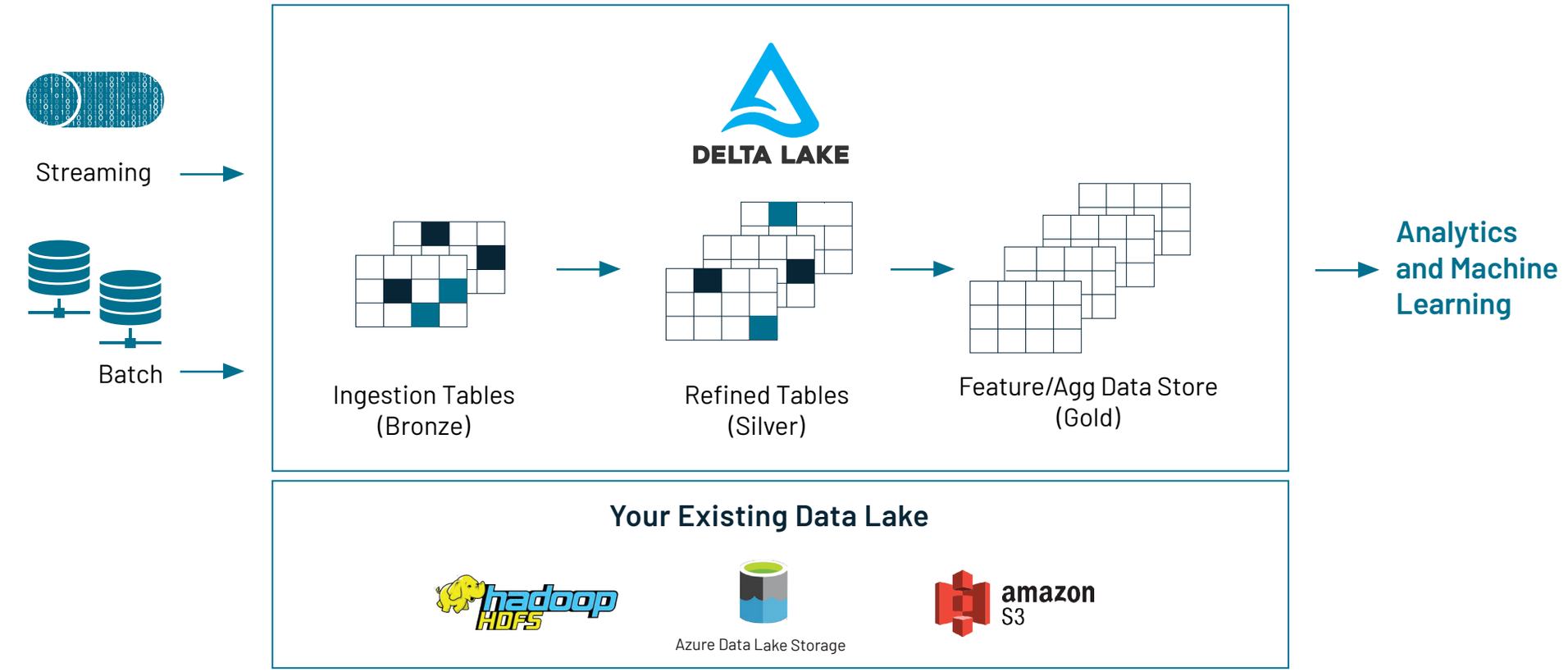
Failed Writes — If a production job that is writing data experiences failures which are inevitable in large distributed environments, it can result in data corruption through partial or multiple writes. What is needed is a mechanism that is able to ensure that either a write takes place completely or not at all (and not multiple times, adding spurious data). Failed jobs can impose a considerable burden to recover to a clean state.

Lack of Consistency — In a complex big data environment, one may be interested in considering a mix of both batch and streaming data. Trying to read data while it is being appended to provides a challenge since on the one hand there is a desire to keep ingesting new data while on the other hand anyone reading the data prefers a consistent view. This is especially an issue when there are multiple readers and writers at work. It is undesirable and impractical, of course, to stop read access while writes complete or stop write access while reads are in progress.

Schema Mismatch — When ingesting content from multiple sources, typical of large, modern big data environments, it can be difficult to ensure that the same data is encoded in the same way i.e., the schema matches. A similar challenge arises when the formats for data elements are changed without informing the data engineering team. Both can result in low quality, inconsistent data that requires cleaning up to improve its usability. The ability to observe and enforce schema would serve to mitigate this.

Delta Lake: A New Storage Layer

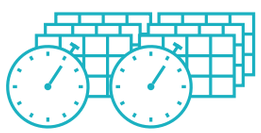
Delta Lake is an open source storage layer that brings reliability to data lakes. Delta Lake provides ACID transactions, scalable metadata handling, and unifies streaming and batch data processing. Delta Lake runs on top of your existing data lake and is fully compatible with Apache Spark APIs. Raw data is ingested from various batch and streaming input sources. Simple, reliable data pipelines help create a curated data lake containing tables of differing degrees of refinement based on business needs. The data in these tables is then made available via the standard Spark APIs or special connectors for various use cases such as machine learning, SQL analytics or feeding to a data warehouse.



Delta Lake: Key Features

ACID Transactions — Data lakes typically have multiple data pipelines reading and writing data concurrently, and data engineers have to go through a tedious process to ensure data integrity, due to the lack of transactions. Delta Lake brings ACID transactions to your data lakes. It provides serializability, the strongest level of isolation level.

Scalable Metadata Handling — In big data, even the metadata itself can be "big data." Delta Lake treats metadata just like data, leveraging Spark's distributed processing power to handle all its metadata. As a result, Delta Lake can handle petabyte-scale tables with billions of partitions and files at ease.

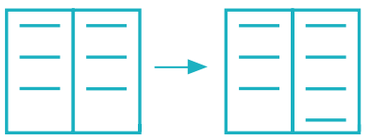


Time Travel (data versioning) — Delta Lake provides snapshots of data enabling developers to access and revert to earlier versions of data for audits, rollbacks or to reproduce experiments. For further details, please see this documentation.

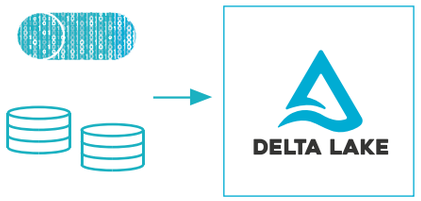
Schema Enforcement — Delta Lake provides the ability to specify your schema and enforce it. This helps ensure that the data types are correct and required columns are present, preventing bad data from causing data corruption.

Delta Lake: Key Features

Open Format — All data in Delta Lake is stored in Apache Parquet format, enabling Delta Lake to leverage the efficient compression and encoding schemes that are native to Parquet.



Schema Evolution — Big data is continuously changing. Delta Lake enables you to make changes to a table schema that can be applied automatically, without the need for cumbersome DDL.



Unified Batch and Streaming Source and Sink — A table in Delta Lake is both a batch table, as well as a streaming source and sink. Streaming data ingest, batch historic backfill, and interactive queries all just work out of the box.

100% Compatible With Apache Spark API — Developers can use Delta Lake with their existing data pipelines with minimal change as it is fully compatible with Spark, the commonly used big data processing engine.

Getting Started With Delta Lake

Getting started with Delta Lake is easy. Specifically, to create a Delta table simply specify Delta instead of using Parquet.

Instead of parquet...

```
dataframe
.write
.format("parquet")
.save("/data")
```

... simply say delta

```
dataframe
.write
.format("delta")
.save("/data")
```

Learn more about Delta Lake:

- Delta Lake Blogs
- Delta Lake Tutorials
- Delta Lake Integrations

For more information, please refer to the documentation.