Delta Lake and Apache Spark

Joshua Cook

**What is Delta Lake?**

Delta Lake is a technology for building robust data lakes and is a component of building your cloud data platform.

Delta Lake is a storage solution specifically designed to work with Apache Spark and is read from and written to using Apache Spark.

A data lake built using Delta Lake is ACID compliant, meaning that the data stored inside of the data lake has guaranteed consistency.

Due to this guaranteed data consistency, Delta Lake is considered to be a robust data store, whereas a traditional data lake is not.

**Elements of Delta Lake**

Delta Lake is comprised of the following elements:

Delta tables

–

A Delta table is a collection of data kept using the Delta Lake technology and consists of three things:

1. the **Delta files** containing the data and kept in object storage
2. a **Delta table** registered in the [Metastore](https://docs.databricks.com/data/metastores/index.html#metastores)
3. the **Delta**[**Transaction Log**](https://databricks.com/blog/2019/08/21/diving-into-delta-lake-unpacking-the-transaction-log.html) kept with the Delta files in object storage

*Delta Files*

Delta Lake, by design, uses Parquet files (sometimes referred to as Delta files) to store an organization’s data in their object storage.

Parquet files are a state-of-the-art file format for keeping tabular data. They are faster and considered more powerful than traditional methods for storing tabular data because they store data using columns as opposed to rows.

Delta files leverage all of the technical capabilities of working with Parquet files but also track data versioning and metadata, and store transaction logs to keep track of all commits made to a table or object storage directory to provide ACID transactions.

Delta optimization engine

–

Because Delta Lake is specifically designed to be used with Apache Spark, reads and writes made to Delta tables benefit from the inherent massively parallel processing capabilities of Apache Spark.

When Apache Spark code is run to read and write to Delta Lake, the following optimizations are available:

* File management optimizations including compaction, data skipping, and localized data storage
* Auto optimized writes and file compaction
* Performance optimization via Delta caching

Delta Lake storage layer

–

Building with Delta Lake, we store data using Delta Lake and then access the data via Apache Spark. With this pattern, organizations have a highly performant persistent storage layer built on low-cost, easily scalable object storage (Azure Data Lake Storage/ADLS, Amazon Web Services Simple Storage Service/S3).

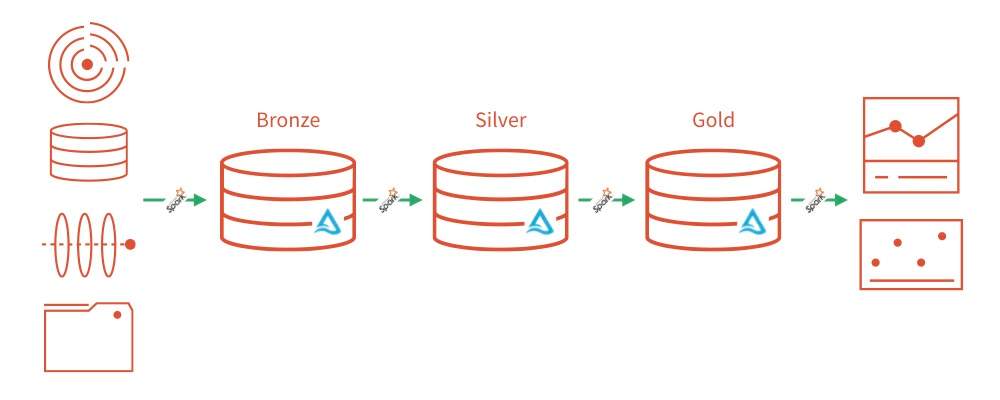
Keeping all of your data in files in object storage is the central design pattern that defines a **data lake**.

Using the Delta Lake storage layer design pattern ensures consistency of data **and** allows the flexibility of working with a data lake.

Delta architecture design pattern

–

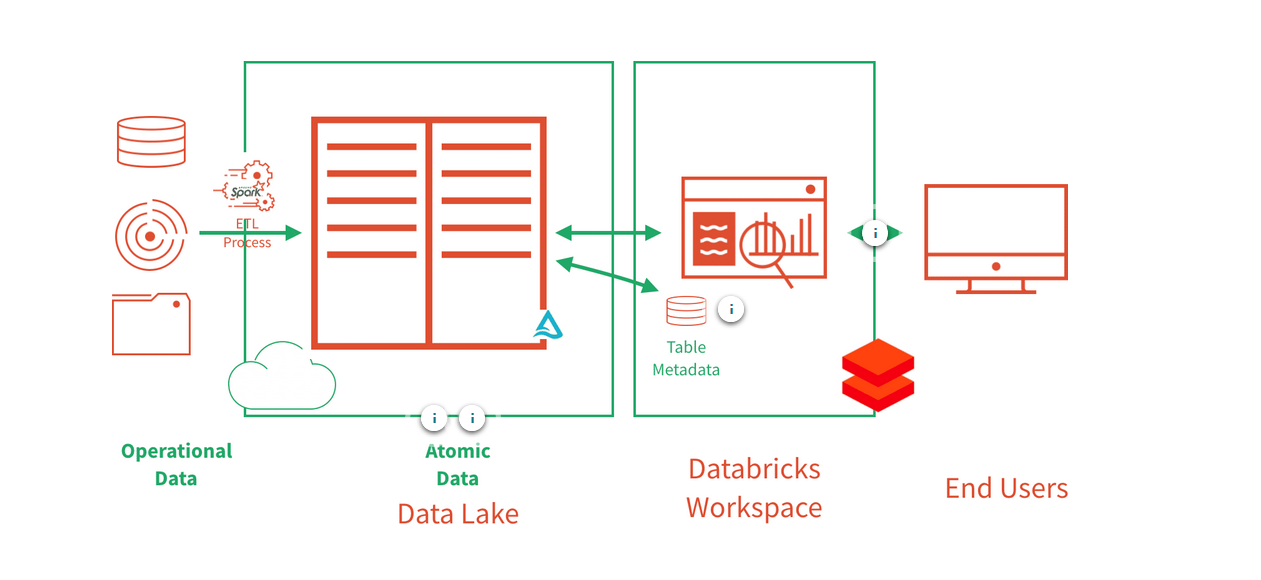
The Delta architecture design pattern consists of landing data in successively cleaner Delta Lake tables from raw (Bronze) to clean (Silver) to aggregate (Gold), as shown in the graphic below.



**Key Concepts: Delta Tables**

As discussed above, Delta tables consist of three things:

1. the **Delta files** containing the data and kept in object storage
2. a **Delta table** registered in the [Metastore](https://docs.databricks.com/data/metastores/index.html#metastores)
3. the **Delta**[**Transaction Log**](https://databricks.com/blog/2019/08/21/diving-into-delta-lake-unpacking-the-transaction-log.html) kept with the Delta files in object storage



**Key Concepts: Delta Lake Storage Layer**

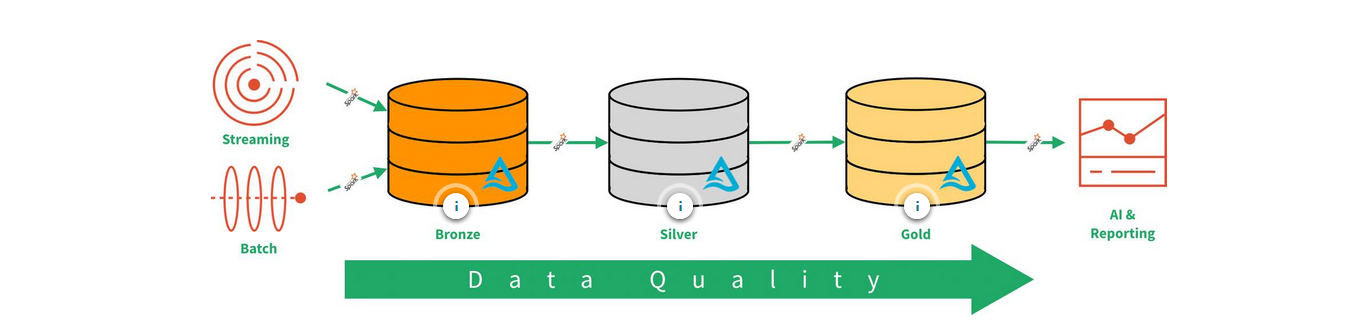
As discussed above, the Delta Lake storage layer consists of two parts:

1. the Delta files containing the data, kept in object storage
2. a Delta table registered in the [Metastore](https://docs.databricks.com/data/metastores/index.html#metastores)

**Key Concepts: Delta Architecture**

Throughout our Delta Lake discussions, we'll often refer to the concept of Bronze/Silver/Gold tables. These levels refer to the state of data refinement as data flows through a processing pipeline.

These levels are conceptual guidelines, and implemented architectures may have any number of layers with various levels of enrichment. Below are some general ideas about the state of data in each level.



**The Power of Delta Lake**

The true power of Delta Lake emerges when it is combined with Apache Spark. Benefits of combining Delta Lake with Apache Spark include:

[Compute/Storage SeparationReliable Data LakeStructured Streaming Support](https://scorm.servicerocket.io/packages/dd645d44-9818-4079-be7a-b55d32fb7d8e/scormcontent/index.html)

**Separation of compute and storage**

A powerful paradigm in modern data storage and processing is the **separation of compute and storage**.

Building systems with decoupled compute and storage has benefits associated with scalability, availability, and cost.

In this system, Apache Spark loads and performs **computation** on the data. It does not handle permanent storage. Apache Spark works with Delta Lake, the first **storage** solution specifically designed to do so.

