Databricks Simple guide

What is Databricks?

As companies have started to collect large amounts of data from many different sources, there is a growing need to have a single system to store it

Making images, sounds and other unstructured data easily accessible for training ML models requires a different architectural approach

Data bricks is simple to use fast data execution and collaborative Apache Spark-based Centralized data processing and analytics platform built on the cloud system.

The data is distributed and parallel processed in memory of multiple nodes in an exceeding cluster because it's supported Spark execution Engine.

It has support for all the Spark use cases machine learning, instruction execution, stream processing, and advanced analytics, etc., and similar like spark data bricks also supports All the languages like Scala, Python, SQL, R, or Java.

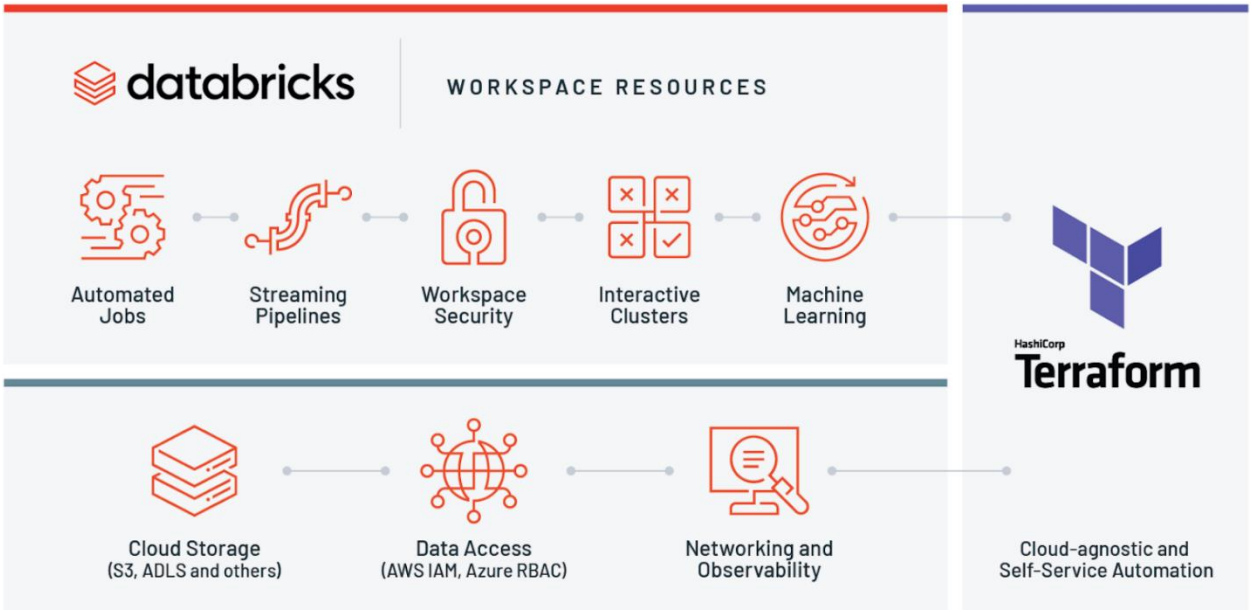

Figure 1: Databricks workspace

Different Components of Databricks

1. Cluster

Since data in Spark is processed in parallel, having more worker nodes may help in faster processing. And driver node, which is responsible for making the request,

Distributing the task to worker nodes, and coordinating the execution.

There are two types of clusters you can create in Databricks, an interactive cluster that allows multiple users to interactively explore and analyze the data and a job cluster that is used to run fast and automated jobs.

2. Workspace

A workspace is a place where you can manage all the data or files in a folder format, which can be notebooks, different libraries, Visualization dashboards, or ML experiments, etc.

3. Notebook

Notebook is the tool where you can write and execute the code or perform different data transformations on data with Spark-supported languages.

On the other hand, you can build the workflow of data with end to end system by invoking the notebooks with another one this can help in creating end-to-end workflows.

4. Jobs.

Jobs allow the execution of a notebook, or if you have an external JAR file that you would like to execute on a Spark cluster, you can do that using jobs. A job can run immediately, or it can be scheduled.

5. Databases and tables

The table has a structure, it has columns, and columns have a datatype. This table is equivalent to a DataFrame because a DataFrame also has a structure.This means any operation that you can perform on DataFrame, you can do the same on a table. A table is created using the file present on the storage.

Data warehouses, lakes…and lake houses?

- Humble data warehouse has some pretty obvious limitations:
- There's usually no built in capabilities for supporting ML use cases — that is the ability to ask more predictive questions using data.
- They lack flexibility and scalability when it comes to storing and retrieving new, complex and unstructured data formats.
- Sure, the emergence of modern Cloud Data Warehouses (CDWs) like Snowflake and Redshift have helped to address the 'scalability' limitations of on premise offerings.

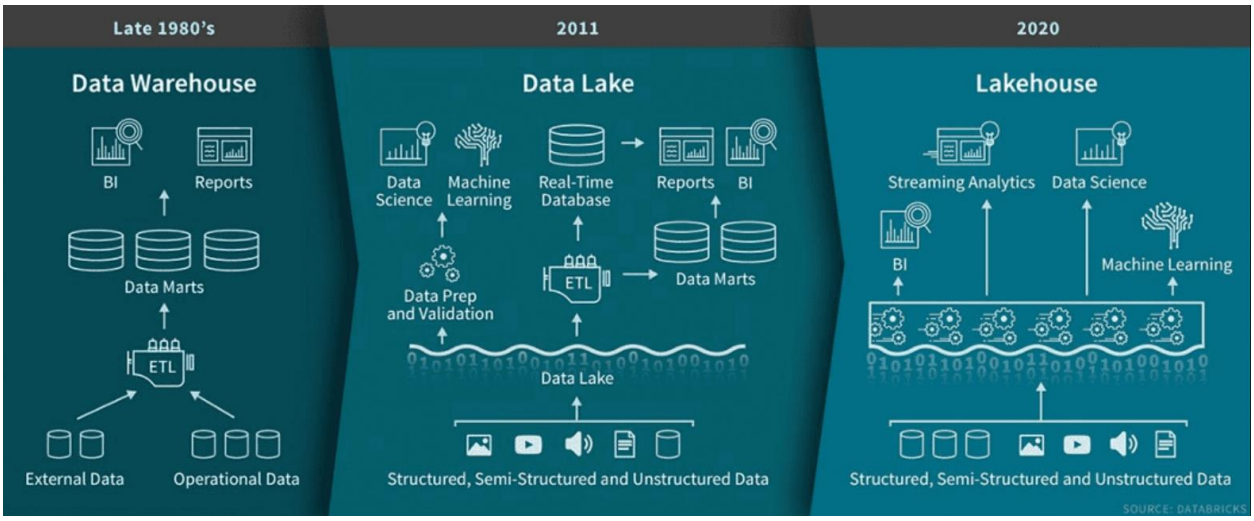

Figure 2: The evolution from Data warehouse to Lakehouse

Continuing along the evolutionary scale in Figure 1, Data lakes came next. They provide a way for storing large amounts of structured, semi-structured, and unstructured data in their raw formats.

The problem is that data lakes have a tendency to become a dumping ground, because they:

- Lack the functionality to easily curate and govern the data stored in them; some folks even refer to them as 'data swamps'!
- Don't support the concept of a transaction or enforce any sort of data quality standards.

Enter the Data Lakehouse

To address the challenges we've discussed so far relating to data warehouses and lakes, the more recent 'lakehouse' concept adds several key advancements:

- **Metadata layers for data lakes** — this is a way to track things like table versions, descriptions of what the data is and enforce data validation standards.
- **New query engine designs** providing high-performance SQL execution on data lakes.
- **Optimized access** for data science and machine learning tools — this is making processed data available in open data formats suitable for ML, enabling faster experimentation and development of models.

Data bricks Architecture

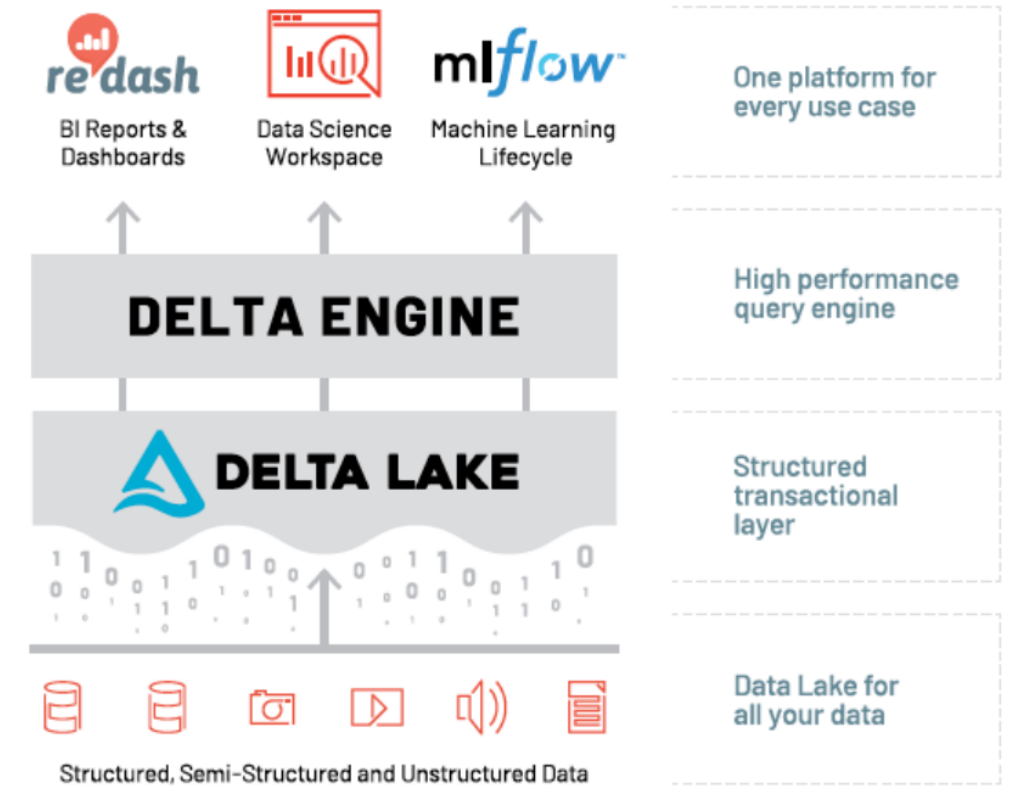

Figure 3: Databricks Architecture

Let’s have a look at the architecture of Databricks. It is divided into three important layers, the Cloud Service, the Runtime, and the Workspace. Let’s understand each layer with components.

1. Cloud Service.

Databricks highly use for three main cloud-based platforms like AWS, Microsoft Azure, and Google cloud, etc, using virtual machine environments to execute tasks.

Databricks File system

- It is one of the prominent features of Databricks with native support of a distributed file system and it is required to persist the data.
- Whenever you create a cluster in Databricks, it mostly comes with preinstalled Databricks File System called DBFS.
- The important point to note is that DBFS is just an abstraction layer, and it uses Azure Blob storage at the back end to persist the data.
- Azure Storage or blob storage is an azure cloud account that you see here is mounted to DBFS by default, you can connect multiple Azure Data Lake Store or Azure Storage.

2. Databricks Runtime

This is the second and most important layer is the Databricks called data bricks

Runtime which is the collection of different data bricks services like Apache spark,

Delta Lake, data bricks I/O, serverless cluster, etc.

Apache Spark

Each Runtime version of apache spark comes with a specific bundled, with some additional set of advancement and optimizations.

In Azure, Databricks runs on the Ubuntu Operation system that means runtime comes with system libraries of Ubuntu, and All the languages with their corresponding libraries are preinstalled.

Databricks I/O

It Is also called DBIO in data bricks which is a module that brings different levels of optimizations on top of Spark related to caching, file decoding, disk read/write, etc. and you can control these optimizations also.

The significant point is that workloads running on Databricks can perform 10 times faster than vanilla Spark deployments.

Now even though you can create multiple clusters in Databricks, doing so adds to cost, so you would want to maximize the usage of the clusters.

Databricks Serverless clusters

It is also called high concurrency clusters, has an automatically managed shared pool of resources that enables multiple users and workloads to use it simultaneously.

But you might think, what if a large workload like ETL consumes a lot of resources and blocks other users' short and interactive queries? Your question is very valid

That's why each user in a serverless cluster gets a fair share of resources, complete isolation, and security from other processes without doing any manual configuration or tuning.

This improves cluster utilization and provides another 10x performance improvement over native Spark deployments.

To use Databricks Serverless, you will have to create a high concurrency cluster instead of a standard one, which you will see in the next module.

Databricks Runtime ML

Databricks also provide native support for various machine learning frameworks via Databricks Runtime ML.It is built on top of Databricks Runtime, so whenever you want to enable machine learning, you need to select Databricks Runtime ML while creating the cluster.

The cluster then comes preinstalled with libraries like TensorFlow, PyTorch, Keras, GraphFrames, and more. And it also supports third-party libraries that you can install on the cluster, like scikit-learn, XGBoost, DataRobot, etc.

Delta Lake

In data bricks, the very interesting component is Delta Lake which is built by the Databricks team and it was Databricks Delta, but now this component is open-source called Delta Lake.

Now most of the teams still using Data Lakes, but they struggle to manage them as the files in Data Lake and they don't have the great features of relational tables.

Delta Lake is an open-source storage layer that gives features to Data Lake, which is very close to relational databases and tables, and much beyond that, like ACID transaction support where multiple users can work with the same files and get ACID guarantees.

You can do different DML operations like insert, update, delete, and merge, and also some time travel operations like keeping snapshots of data enabling audits and rollbacks, etc.

3. Databricks Workspace.

In the data bricks workspace, two-part have been created which handle the workspace and production of spark execution jobs.The first one is an interactive workspace and the second one is the data bricks production let's check each one separately in the details.

Interactive workspace

In this environment, you can explore and analyze the data interactively just like you open an Excel file, apply the formula, and see the results immediately.

In the same way, you can do complex calculations and interactively see the results in the workspace. You can also render and visualize the data in the form of charts.

In Databricks Workspace, you get a collaborative environment. Multiple people can write code in the same notebook, track the changes to the code, and push them to source control when done.

And datasets that you have processed can be put together on a dashboard. It could be for the end-users, or these dashboards can also be used to monitor the system. You will learn about these components that enable these features in just a minute.

Databricks Production

After done with data exploration, you can now build end-to-end workflows by orchestrating the notebooks. These workflows can then be deployed as Spark jobs

Similarly, in the same workspace, you cannot just interactively explore the data but you can also take it to production with very little effort.