**DATA PLATFORMS, DATA STORES AND SECURITY**

**ARCHITECTING THE DATA PLATFORM**

A data platform architecture. A layer represents functional components that perform a specific set of tasks in the data platform. The layers that we’re going to explore, include:

Data Ingestion or Data Collection Layer, Data Storage and Integration Layer, Data Processing Layer, and Analysis and User Interface Layer.

**The Data Collection Layer** is responsible for connecting to the source systems and bringing the data from these systems into the data platform. This layer performs the following key tasks: Connect to data sources. Transfer data from these data sources to the data platform in streaming, batch, or both modes. Maintain information about the data collected in the metadata repository. For example, how much data was ingested in a batch, data source, and other descriptive information. Google Cloud DataFlow, IBM Streams, IBM Streaming Analytics on Cloud, Amazon Kinesis, and Apache Kafka are some of the tools used for data ingestion, supporting both batch and streaming modes. Once data is ingested, it needs to be stored and integrated.

**The Storage and Integration layer** in a data platform needs to: Store data for processing and long-term use. Transform and merge extracted data, either logically or physically. Make data available for processing in both streaming and batch modes. The storage layer needs to be reliable, scalable, high-performing, and also cost-efficient. IBM DB2, Microsoft SQL Server, MySQL, Oracle Database, and PostgreSQL are some of the popular relational databases. Cloud-based relational databases, also referred to as Database-as-a-Service, have gained great popularity over the recent years. Such as IBM DB2 on Cloud, Amazon Relational Database Service (RDS), and Google Cloud SQL, and SQL Azure. In the NoSQL, or non-relational database systems on the cloud, we have IBM Cloudant, Redis, MongoDB, Cassandra, and Neo4J. Tools for integration include IBM’s Cloud Pak for Data and Cloud Pak for Integration; Talend’s Data Fabric and Open Studio. Open-source tools such as Dell Boomi and SnapLogic are also very popular integration tools. There are a number of vendors offering cloud-based Integration Platform as a Service (or iPaaS). For example, Adeptia Integration Suite, Google Cloud's Cooperation 534, IBM's Application Integration Suite on Cloud, and Informatica's Integration Cloud. Once the data has been ingested, stored, and integrated, it needs to be processed.

**The processing layer** should be able to: Read data in batch or streaming modes from storage and apply transformations. Support popular querying tools and programming languages. Scale to meet the processing demands of a growing dataset. Provide **a way for analysts and data scientists to work with data in the data platform**. Some of the transformation tasks that occur in this layer include: **Structuring,** essentially, actions that change the form and schema of the data. This change may be as simple as changing the order of fields within a record or dataset or as complex as combining fields into complex structures using joins and unions. **Normalization**, which focuses on cleaning the database of unused data and reducing redundancy and inconsistency. **Denormalization**, which combines data from multiple tables into a single table so that it can be queried more efficiently for reporting and analysis. And **Data Cleaning,** which fixes irregularities in data to provide credible data for downstream applications and uses. There are a host of tools available for performing these transformations on data, selected based on the **data size**, **structure**, and **specific capabilities of the tool**. Such as **spreadsheets, OpenRefine, Google DataPrep, Watson Studio Refinery, and Trifacta Wrangler.** Python and R also offer several libraries and packages that are explicitly created for processing data. **It’s important to note that storage and processing may not always be performed in separate layers.** For example, in relational databases, storage and processing can occur in the same layer, while in Big Data systems, data can be first stored in the Hadoop File Distribution System, or HDFS, and then processed in a data processing engine like Spark. And, **the data processing layer can also precede the data storage layer, where transformations are applied before the data is loaded, or stored, in the database.**

**The Analysis and User Interface Layer** delivers processed data to data consumers. Data consumers can include: Business Intelligence Analysts and business stakeholders who consume this data through interactive visual representations, such as dashboards and analytical reports. Data Scientists and Data Analytics that further process this data for specific use cases. Other applications and services that may need this data as input for further use. The Analysis and UI Layer needs to support: **Querying tools and programming languages**. For example, SQL for querying relational databases and SQL-like querying tools for non-relational databases, such as CQL for Cassandra, Programming languages such as Python, R, and Java, **APIs** that can be used to run reports on data for both online and offline processing. **APIs** that can consume data from the storage in real-time for use in other applications and services. **Dashboarding and Business Intelligence applications**. For example, IBM Cognos Analytics, Tableau, Jupyter Notebooks, Python and R libraries, and Microsoft Power BI.

Overlaying the Data Ingestion, Data Storage and Integration, and Data Processing layers is the **Data Pipeline layer** with the Extract, Transform, and Load tools. This layer is responsible for implementing and maintaining a continuously flowing data pipeline. There are a number of data pipeline solutions available, most popular among them being Apache Airflow and DataFlow.

**Factors for selecting and Designing Data Stores.**