Big Data Architecture

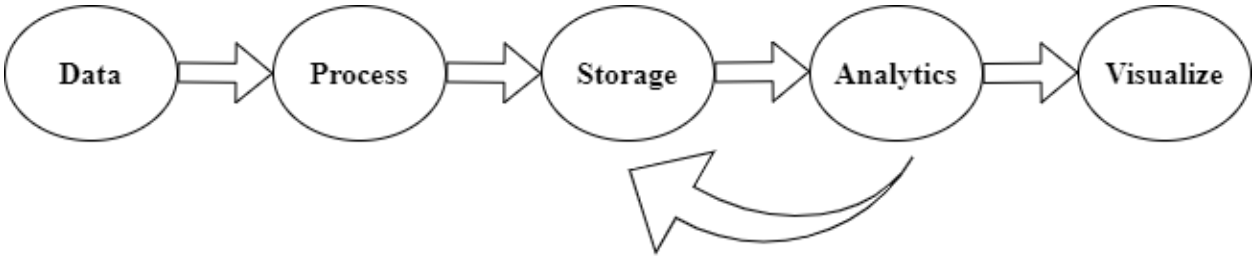
- **Overview**
    - Choosing the right architecture is critical to your big data project
    - Unlike software engineering architecture, there is no standard architectures for big data that have been used for years.
    - You need to identify your problem statement, before start designing your architecture.
    - It refers to the logical and physical structure that dictates how big data are processed, stored, managed, and accessed.
    - It is the foundation for big data analytics
    - An architecture design document is the key technical document used to determine whether the critical system requirements are met.
    - Selection attributes to choose the right architecture:
        - Type of data: batch or stream
        - Scalability
        - Are you keeping the extracted raw data?

- **Definitions**
    - "Big Data architecture is the **logical and/or physical** layout/structure of how Big Data will be **stored**, **accessed** and **managed** within a Big Data or IT environment" Techopedia

- **Layered Architecture**
    - It works well whenever system requirements call for independent tasks organized hierarchically.
    - Architecture layers have been used in databases, operating, computer-to-computer communications, and now in big data.
    - Definitions:

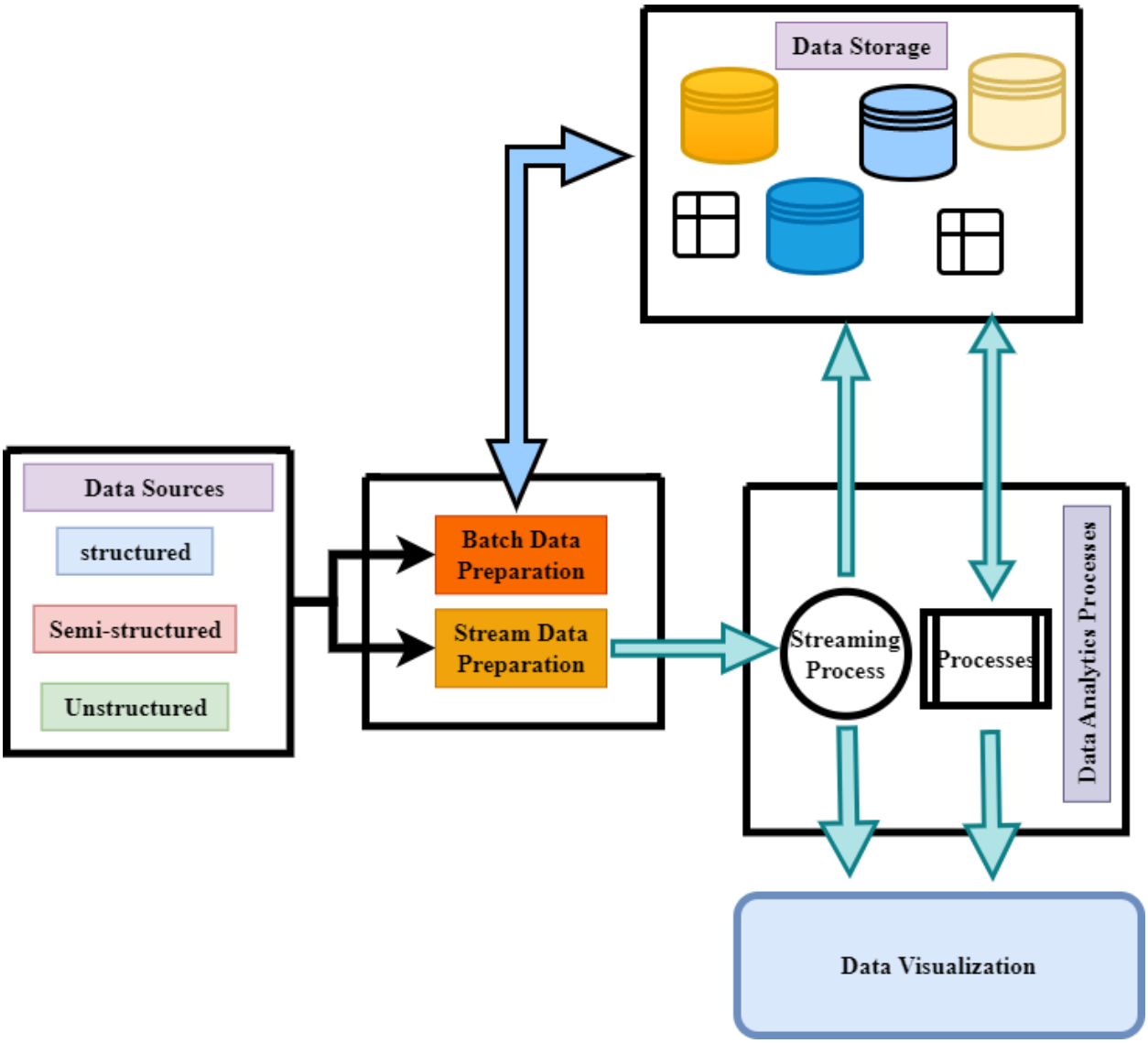
- A layered architecture is designed as a hierarchy of client-server processes that minimizes interaction between layers.
- Each layer acts as a client for the module above it and acts as a server for the module below it in an architecture layer.

- **Big Data Generic Architecture:**
  - Big Data uses a multi-layer architecture

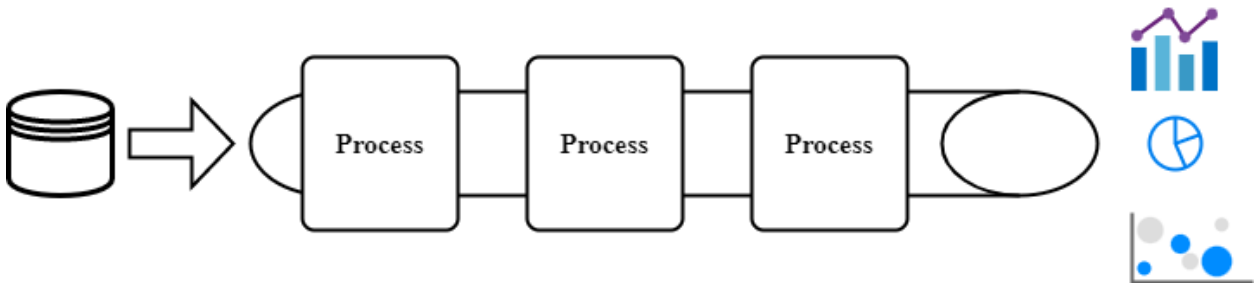


  - Big Data Architecture Layers:
    - Big Data **Sources Layer**:
      - Big data include the both batch and stream data:
        - Structured data
        - Semi-structured data
        - Unstructured data
    - Big Data **Storage Layer**:
      - Depending of the big data project, data can be stored in HDFS, NoSQL database, etc.
    - Big Data **Preparation Layer**:
      - It processes data from the source, converts the data into a format ready for both storage and analysis.

- Big Data **Analytics Layer**:
  - It handles both batch and stream data analytics.
- **Visualization Layer**:
  - It receives results from the big data analysis layer and visualize them in dashboard, charts, etc.

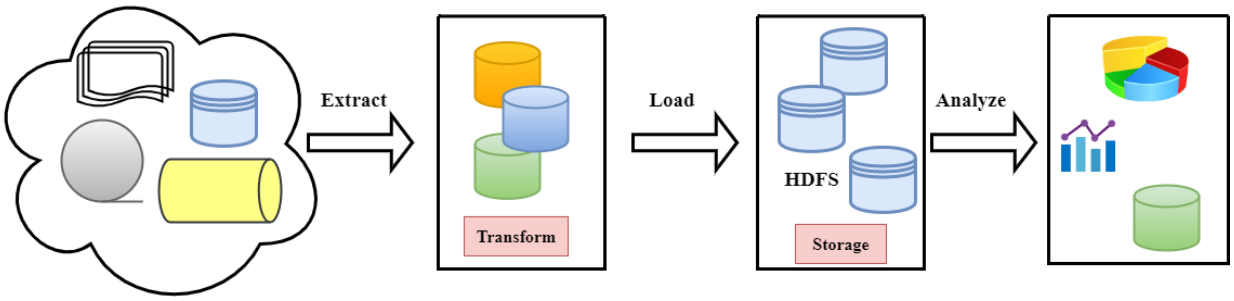


- **Data Pipeline Architecture**
    - This architecture style is modeled after assembly lines in manufacturing plants.
    - The pipe consists of stages. When one stage completes its processing, its outputs become the inputs of the following stage.



    - There are different data pipelines for different types of big data:
        - Streaming Data pipeline supports data with high velocity (**Velocity**):
            - Data can be captured, processed at a higher speed to make actionable intelligence
            - To process ATM transactions and store them in a big data storage.
        - Batch data pipeline (**Volume**):
            - To process significant volumes of data concurrently.
            - To process large e-commerce web log data and store it in a big data storage such as HSFS or data warehouse.
            -
        - Different data format pipeline (**Variety**):
            - Structured, unstructured, and semi-structured.

- ETL
  - ETL pipeline is a type of data pipeline.



  - ETL process consists of the following steps:
    - **Extract**: Data is first extracted from data sources
    - **Transform**:
      - Extracted data is processed and converted into a data format suitable for the target application.
      - Data Filtering
      - Data Mapping: Takes one data input and map it to another equivalent format.
        - Currency conversion
        - Align time zone in multiple data
      - Data Deduplication
      - Joining data from multiple sources
      - Aggregating data
      - Splitting data
      - Handling Missing data
    - **Load**: Transformed data in loaded in a data storage.

- Transformation Example:
  - Joining data from multiple sources
  - Given the following two tables:

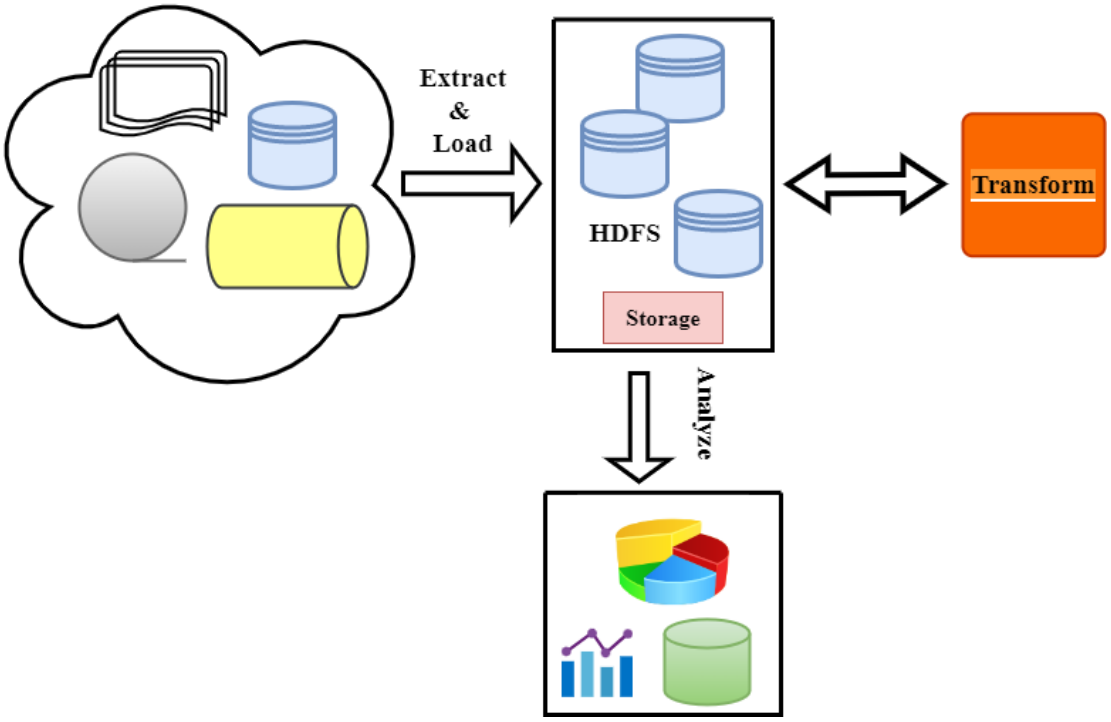
E-commerce Sites	Customer	Product	Cost	Date
S1	Name1	P1	C1	D1
S2	Name2	P2	C2	D2
S1	Name3	P1	C1	D3

Product	Original Cost	Sales Cost	Details
P1	C1	S1	Details1
P2	C2	S2	Details2

E-commerce Site	Products Sales	profits	Year
S1			
S2			

- ETL Tools:
  - Apache Nifi (Free)
  - Informatica
  - Talend
  - Bods

- ELT:
  - ELT pipelines (extract, load, transform)
  - Unlike ETL, ELT does not require data transformations to take place before the loading process.

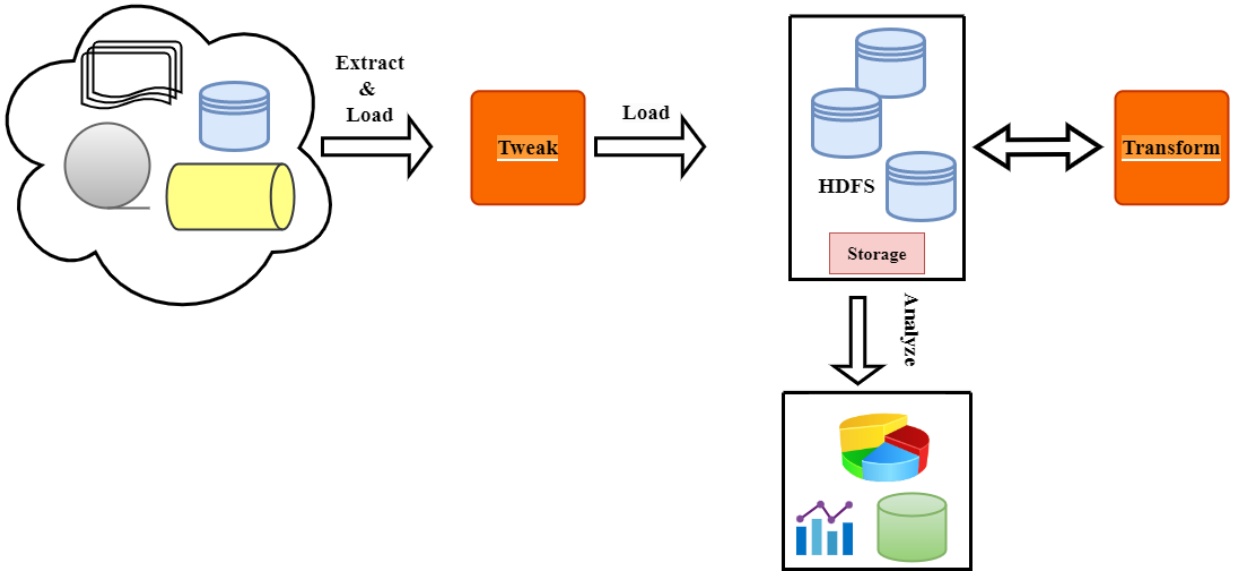


  - It reverses the steps, allowing for a quick load of raw data that will be later transformed and analyzed.
  - ELT transform data within the data storage itself.
  - Raw data is stored indefinitely in the data storage, allowing for multiple transformations.

- ETL vs ELT: What's the Difference (and Which is Better)?

	ETL	ELT
Extraction	Raw data is extracted from Data Sources	Raw data is extracted from Data Sources
Transformation	Raw data is transformed on and moved to a data storage or staging area	Data transformation is done within Data Storage
Loading	Data is loaded in a Data Storage for analysis.	Raw data is loaded directly into the Data Storage.

- EtLT:
  - ETLT combines the best of ETL and ELT



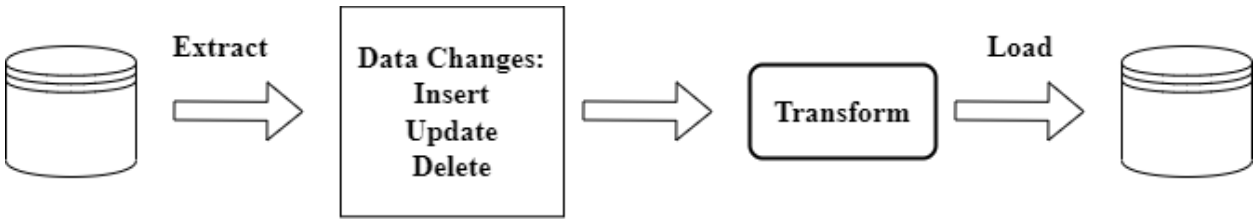
  - ETLT:
    - E: Extract
      - Extract data from data sources
    - t: Tweak
      - It is light process to clean and prepare data to help reduce latency and leave the most

complex transformation for later in the pipeline.
        - Example of tweaks:
            - Remove duplicate,
            - Prepare data to be compliant with the General Data Protection Regulation (GDPR) to protect personal and sensitive information such as CC information, customer personal information, etc.:
                - Mask/Encrypt critical information
    - L: Load
        - Load data in the Data Storage
    - T: Transform
        - Tansform data with Data Storage

- Change Data Capture (CDC) pipeline
    - CDC is a process that detects changes in data and apply those changes to a target data. Otherwise, the data analytics will be outdated.
    - It is generally used to sync source data and target data.
    - Example of data changes in a database are stored sin database logs:
        - Insert, update, and delete
    - There are two types of CDC implementation:
        - Pull-based: The target data requests changes from the source data.
        - Push-based: The source data sends change to the target data source.
    - There are three methods to implement change data capture:

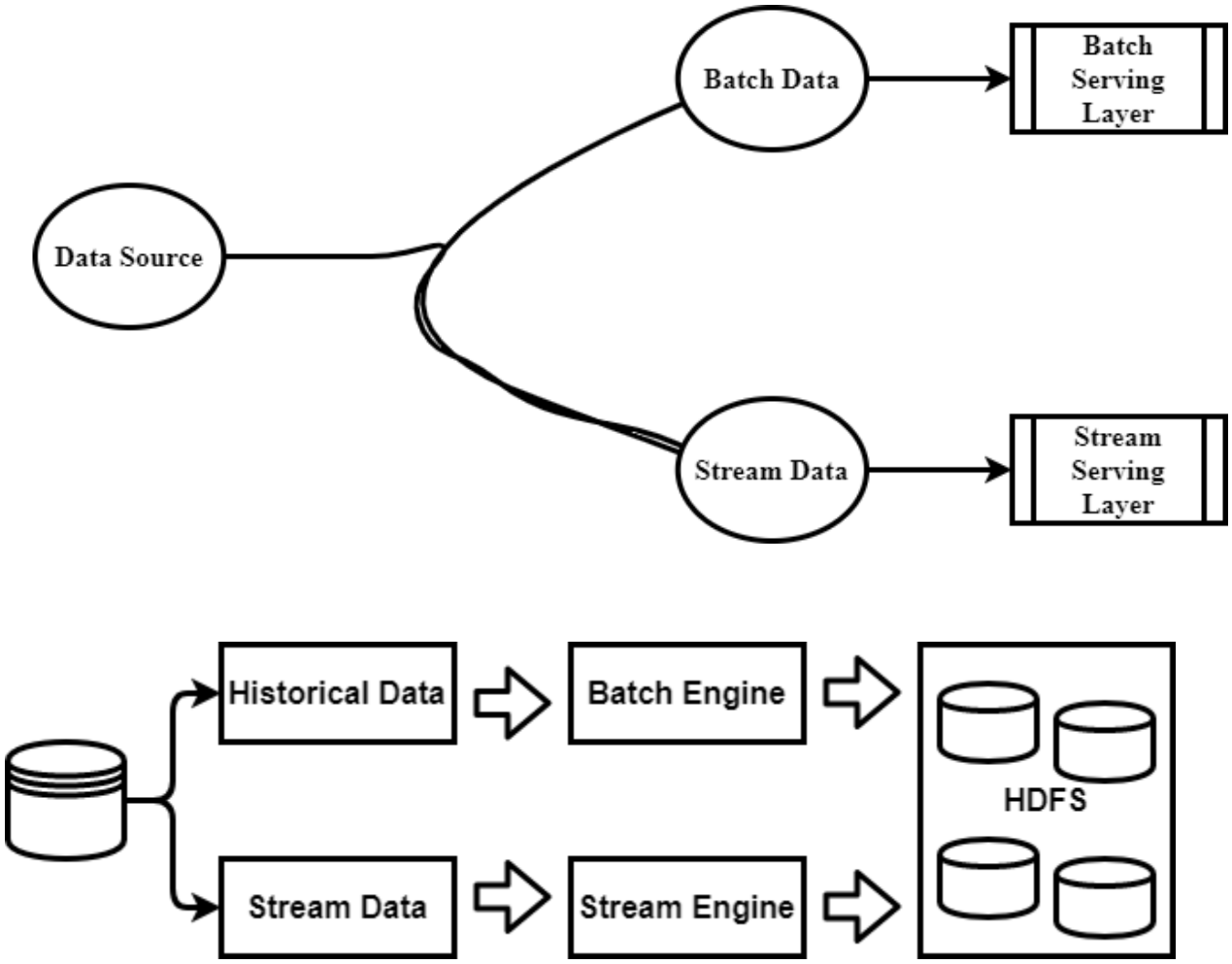
- Log-based:
    - This method accesses the database log file for any changes.
    - Log-based CDC is most efficient since it does involve the database resources in the notification of data change.
- Query-based:
    - This method requires the database to store the changes in a separate database table. The changes are then queried.
- Trigger-based:
    - In this method, the data source triggers a notification when data changes occur within the source database. This method uses a timestamp or other attributes that indicate a data change.
-

- CDC and ETL:
    - CDC makes ETL more efficient.

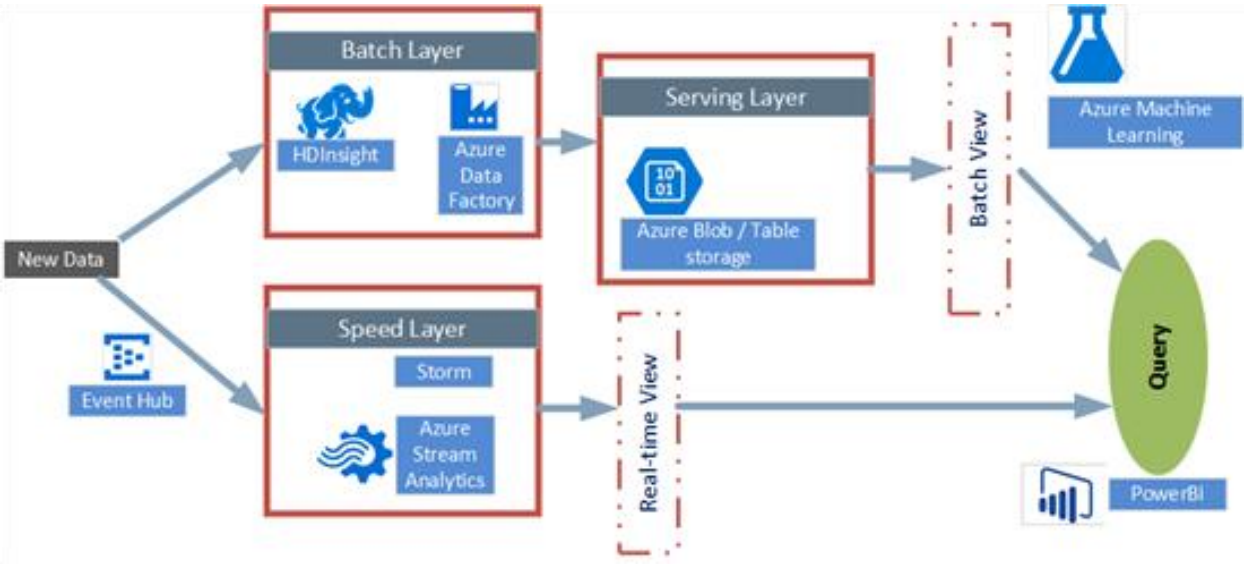


    - ETL does not need to extract the entire data from the data source, it only extracts the data changes and update the target data.

- **Lambda Architecture**
    - It is not the Python anonymous lambda function.
    - It is designed by Nathan Marz (Twitter)
    - Case Studies:
        - Fraud Detection:
            - Current transaction
            - Transaction history
        - Loan Processing:
            - Credit Report
            - Loan application
            - Recommendation
    - Lambda consists of the following layers:
        - Batch Layer:
            - It reads, processes, and combines the entire historical raw data (i.e., web log) to produce clean data ready for serving layer.
            - The batch processing will periodically run or start a new run right after completion of the last batch.
            - Common technologies used to implement this layer include Hadoop map-reduce, Spark, or Hive.
        - Contains the immutable, constantly growing master dataset.
        - Speed Layer:
            - The focus of this layer is to analyze the incoming streaming data in near real-time.
        - Serving Layer:
            - Loads and presents data for analytics.
            - This layer queries data from the most recent batch analytics and combines that data with the output from the speed layer to produce a unique data view for the data.

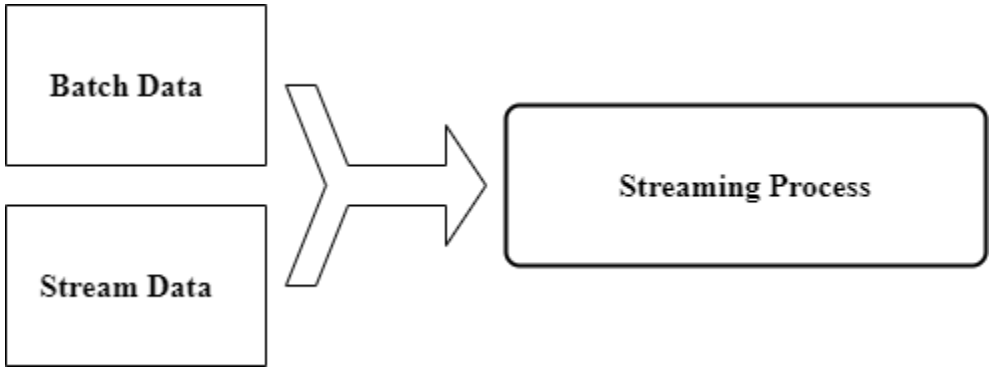


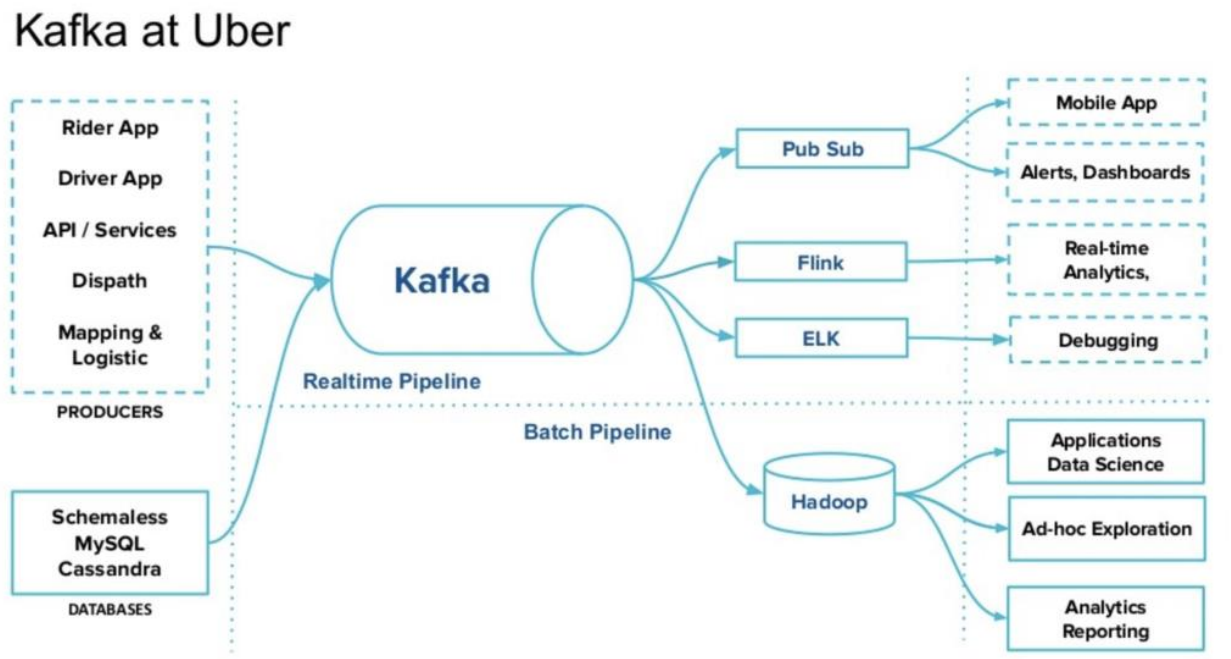
- Implementation:
  - Hadoop HDFS can be used to store batch data,
  - Spark (or Storm) can be used for speed layer,
  - HBase (or Cassandra) can be the serving layer
  - Hive creates views that can be queried.



CourtseyMicrosofot.com

- **Kappa Architecture**
    - It is a simplification of the Lambda architecture.
    - Batch data operations are a subset of streaming operations ➔ Everything can be treated as a stream.
    - Kappa eliminates the batch layer leaving only the streaming layer.
    - Data is considered as an event and processed by a single code application: batch & stream data.
    - Kappa maintains a single flow, the code, the maintenance, and the system update are considerably reduced.
    - Technologies of distributed streaming systems:
        - Amazon Kinesis
        - Apache Flink
        - Apache Spark
        - Apache Storm
        - Kafka Streams



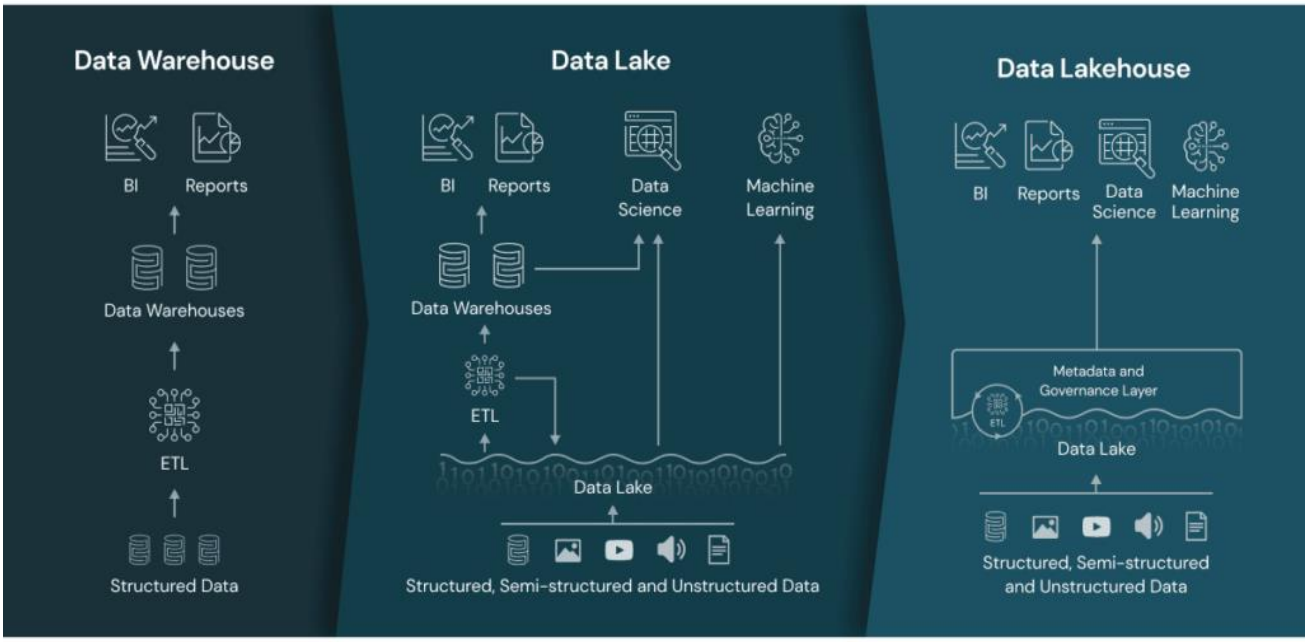


Courtesy: Uber

- **Lambda & Kappa Architectures Comparison**

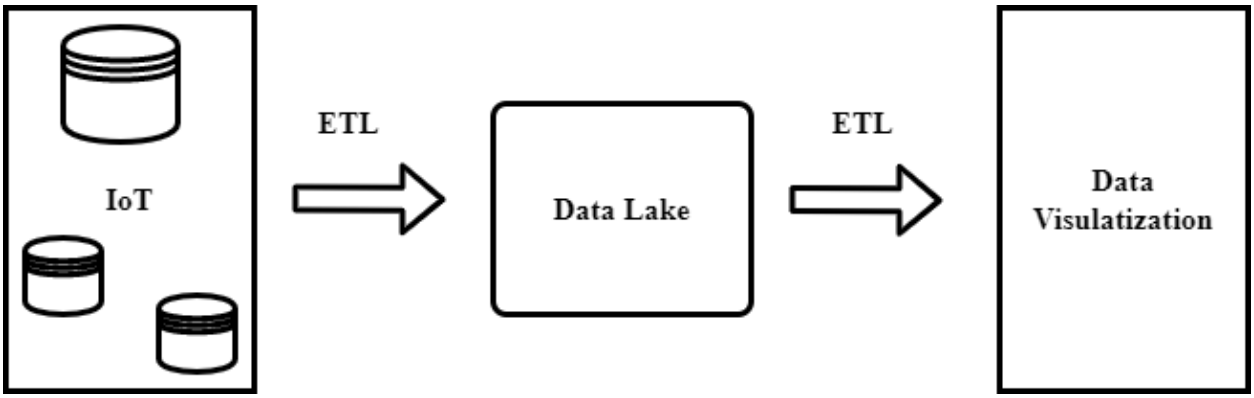
	Lambda	Kappa
Code Duplication	Yes: Batch & Stream	Single code
Implementation	Simple	Complex
Maintenance	Not easy as you need to maintain two different systems	Easy

- **Data Lake**
    - A Data Lake is an extensive storage repository that can store large amount of raw unprocessed data structured, semi-structured, and unstructured data.
    - It is a place to store every type of data in its native format with no fixed limits on account size or file. It offers high data quantity to increase analytic performance and native integration.
        - Data Lake could include clickstream, and real-time data.
        - Data Lake is like a large container which is very similar to real lake and rivers.



Courtesy: Databricks.

- **Data Lake Architecture Layers**



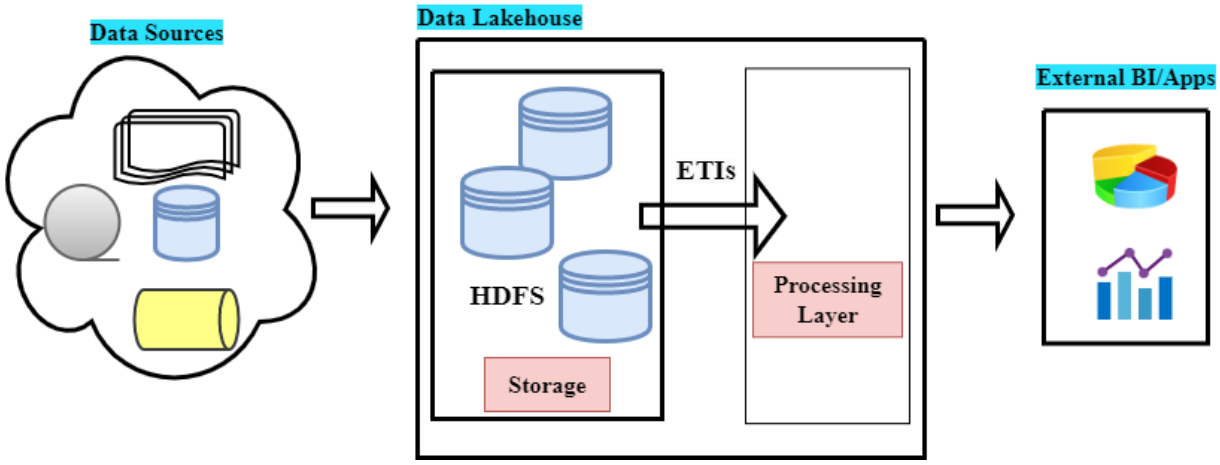
- **What are the benefits of data lakes?**
  - It stores all data in one single place, e.g., cloud based.
  - Data can be accessed by different users based on their needs, e.g., AI applications.
  -
- **What are the Drawbacks of Data Lakes?**
  - **Unreliable data:**
    - Generally, the incoming data is not validated and therefore the quality of data may not be consistent.
    - This may yield to incorrect data analysis.
  - **Data swamps:**
    - It happens when data is expired, or the lake include poor quality data.
  - **Slow performance:**
    - As the size of the data in a data lake increases, the performance of traditional query engines has traditionally gotten slower. Some of the

bottlenecks include metadata management, improper data partitioning and others.

- **Security:**
  - It is hard to secure data lakes since data cannot be deleted or updated.

- **Data Lakehouse Architecture**
  - It is an open architecture that combines the best of data lakes and data warehouses

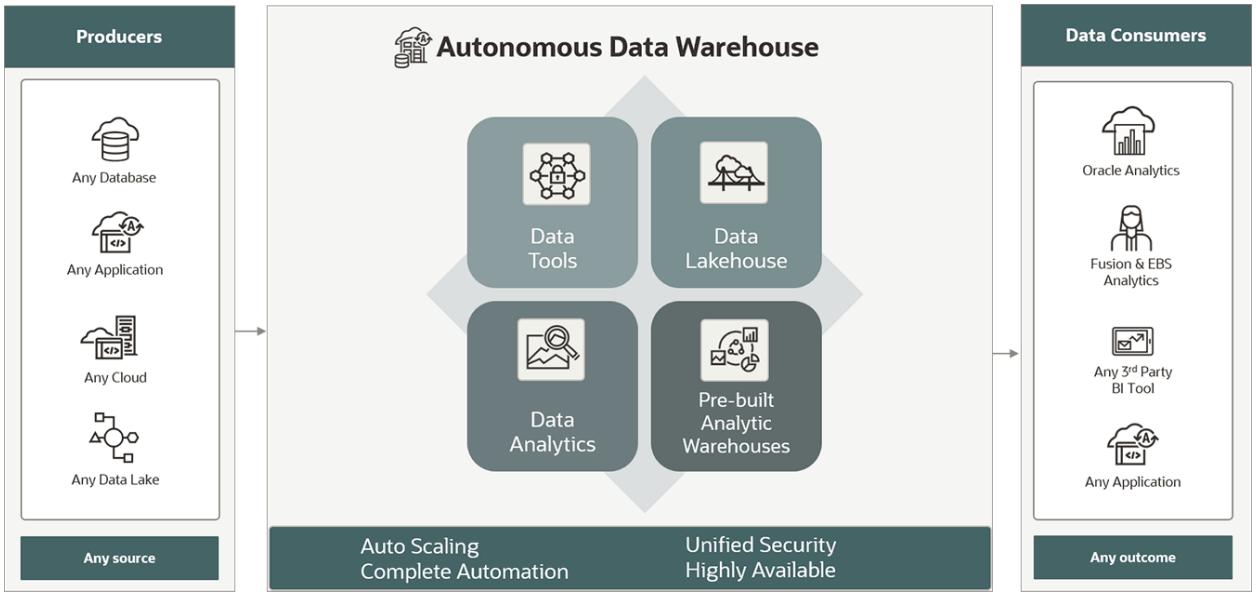
Data Lake + **Data warehouse** = **Lakehouse**



  - Lakehouse Characteristics**:**
    - **Transaction support**:
      - Multiple users concurrently accessing data in a lakehouse using for examples ELT's.
      - Lakehouse provides support for **ACID** (Atomicity, Consistency, Isolation, Durability)

transactions to ensure consistency when multiple users are concurrently accessing data.

- **Schema enforcement and governance**:
  - It supports the data warehouse schema such as star/snowflake schemas.
  - It also has robust governance and auditing mechanisms.
- **BI support**:
  - Lakehouse allows BI tools to directly access data sources without data duplication.
- **Storage is decoupled from compute:**
  - Data are stored in different clusters and decoupled from processing ➔ scalability of users and data
- **Openness:**
  - Lakehouse support multiple data formats such as Avro (Row-based storage format) and Parquet (Columnar-based storage format).
  - Provides API so other tools can directly access data.
- **Support for diverse data and workload:**
  - Lakehouse can be used to store, refine, analyze, and access multiple data types needed by multiple workloads including data mining, other analytics.
- **End-to-end streaming:**
  - Support for stream processing.


Producers
Any Database
Any Application
Any Cloud
Any Data Lake
Any source
Autonomous Data Warehouse
Data Tools
Data Lakehouse
Data Analytics
Pre-built Analytic Warehouses
Auto Scaling
Complete Automation
Unified Security
Highly Available
Data Consumers
Oracle Analytics
Fusion & EBS Analytics
Any 3rd Party BI Tool
Any Application
Any outcome

Oracle.com