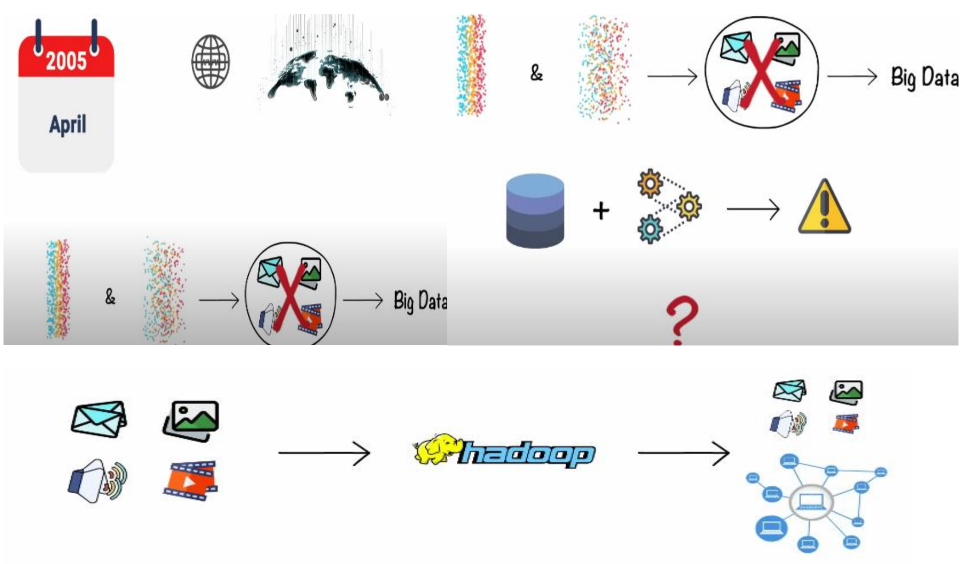
Hadoop

- Open-source framework designed to process and store large datasets in a distributed computing environment.
- Based on the principle of scalability and fault tolerance.

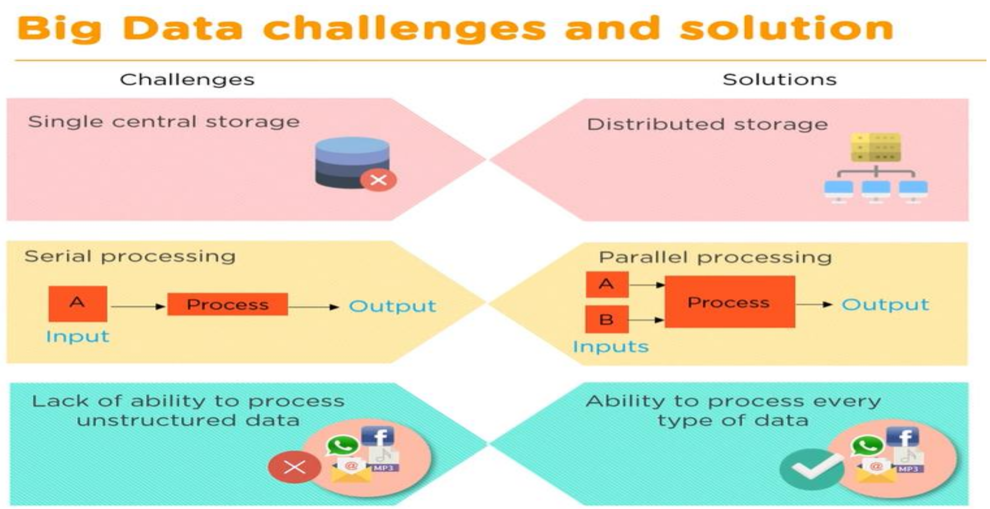
Key Components

1. HDFS (Hadoop distributed file system)
    - Distributed file system that stores data across multiple machines while ensuring reliability and fault tolerance.
    - Files are split into blocks (default size: 128 MB) and distributed across nodes.
2. **MapReduce:**
    - A programming model for distributed data processing.
    - Consists of two main tasks:
        - ✓ **Map:** Processes input data and converts it into key-value pairs.
        - ✓ **Reduce:** Aggregates or reduces the intermediate output from the Map step to generate the final result.
3. **YARN (Yet Another Resource Negotiator):**
    - Manages resources and schedules tasks across the Hadoop cluster.
    - Ensures efficient utilization of cluster resources.
4. **Hadoop Common:**
    - A collection of utilities and libraries that support other Hadoop components.

Visual Explanations:

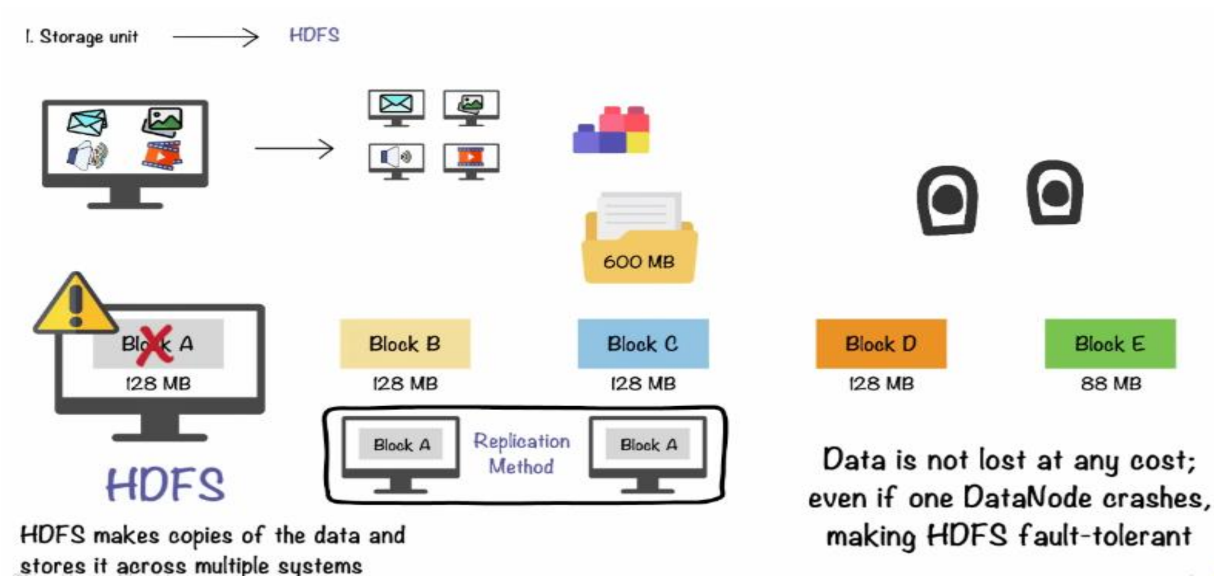


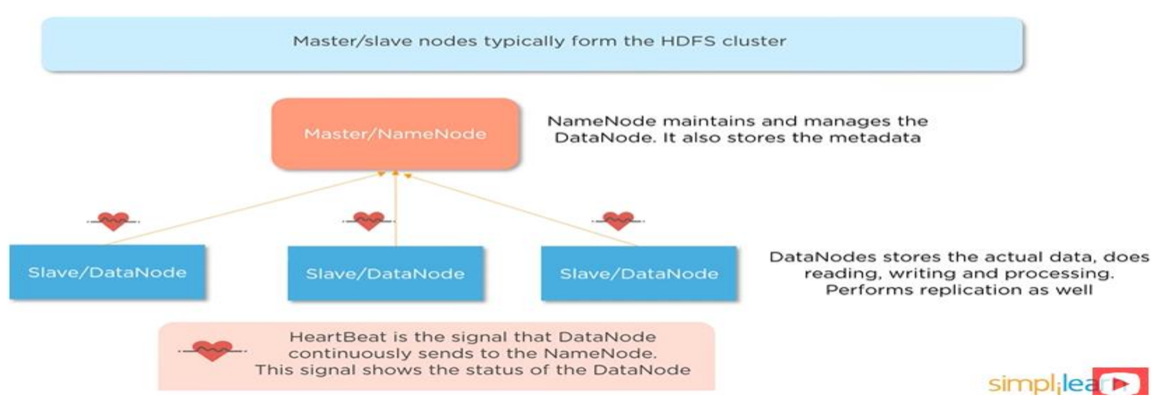
The Internet gave rise to Big Data and the conventional storage processor unit was not enough hence the need for a solution. (solution: multiple storage units and processor which was incorporated in the framework of Hadoop using a cluster of commodity Hardware).

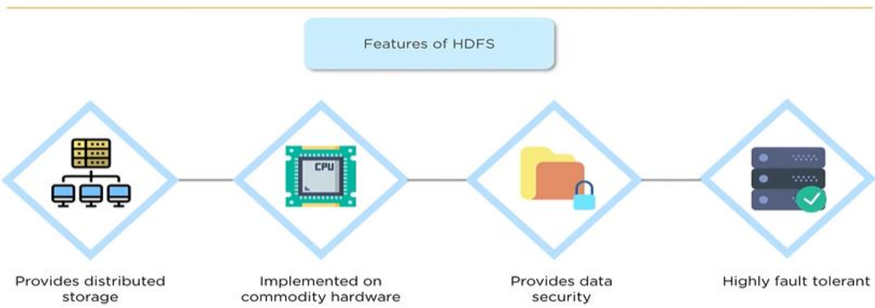


Hadoop Framework Consisted of three components which were basically designed to work on big data.

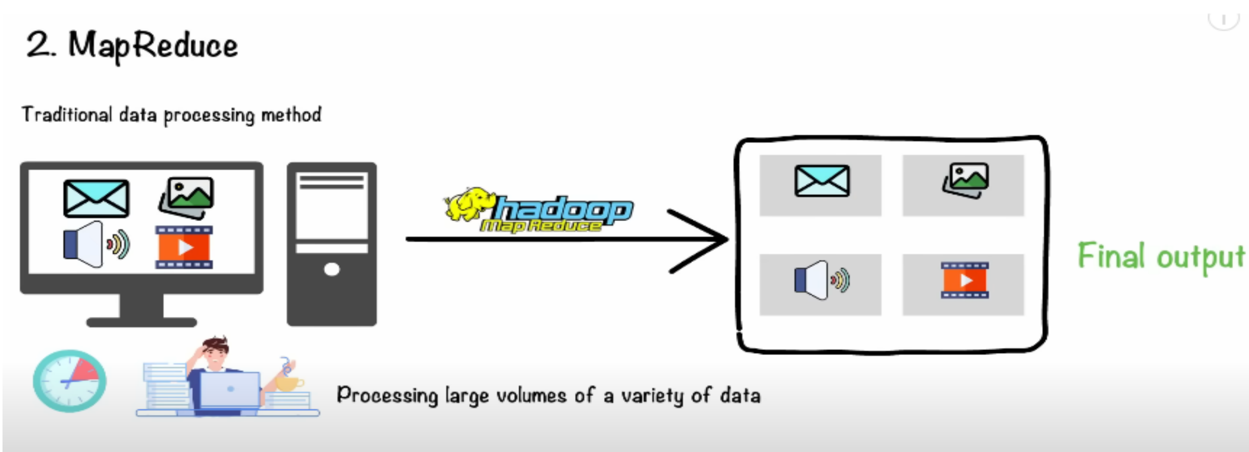
- Storage Unit (HDFS)
  Data is stored in multiple nodes and reliability ensured thru' fault tolerance (Replication)



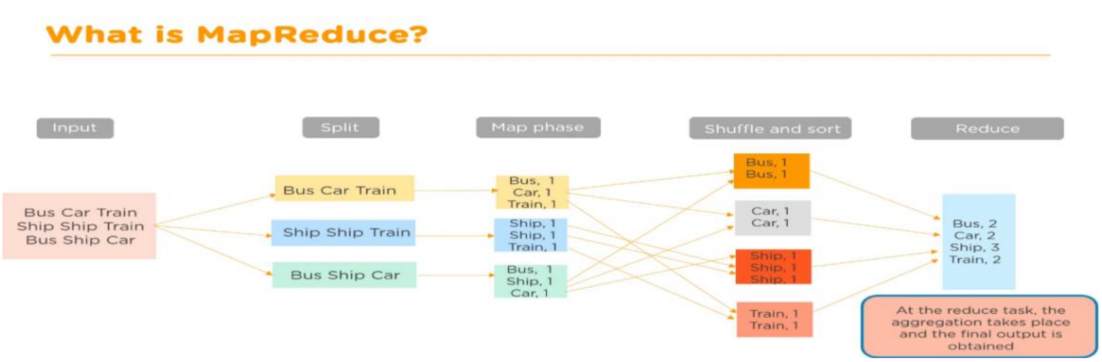




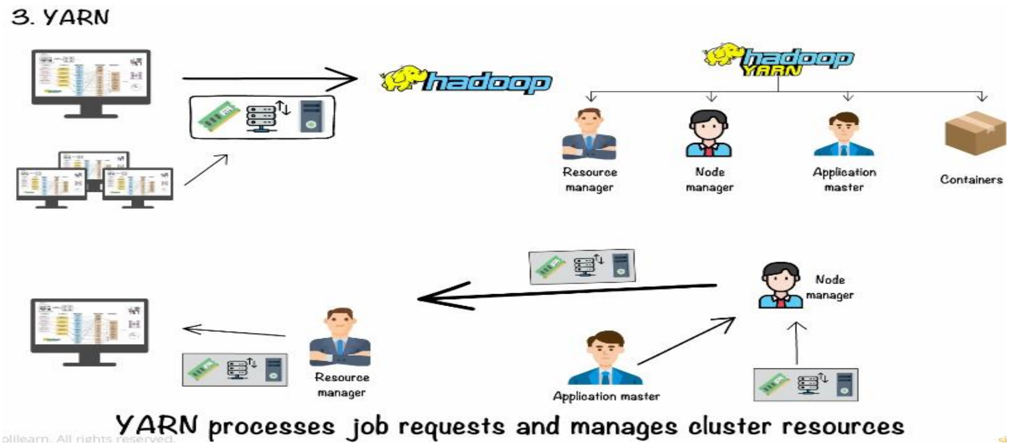
- Processing Unit (Map Reduce)
  - ✓ Traditionally Entire data would be processed on a single machine having a single processor which consumed time and space (Inefficient: Large Volumes of a variety of data).
  - ✓ To Overcome, Map Reduce splits data into parts and processes each of them separately on different data nodes. Individual results are then aggregated to give final output.



Example: Counting the Number of occurrences of words.



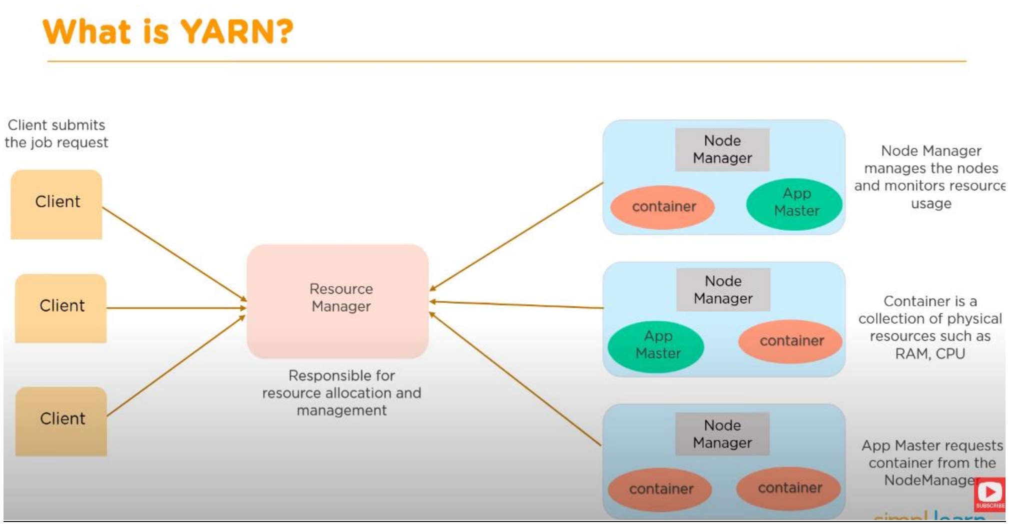
- **Resource Manager Unit (YARN)**
  - ✓ Once the Map Reduce job is ready, it is run down the Hadoop cluster which is done by a set of resources (RAM, Network Bandwidth and CPU)
  - ✓ To efficiently Manage these Resources, YARN Comes into play.



- Resource Manager: Assigns Resources
- Node Manager: Handle the Nodes & Monitor the Resource Uses in the node
- Containers: Hold a collection of physical Resources

<u>Example: To process the Map_Reduce Job:</u>

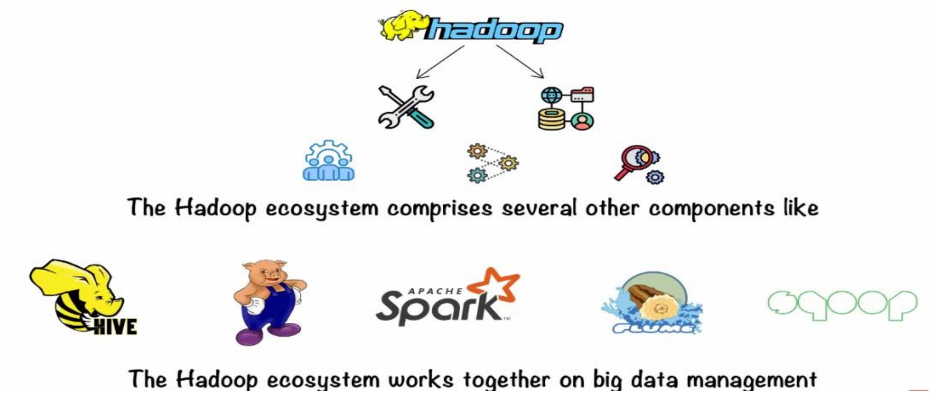
The **Application Master** Requests the container from the **Node Manager**, Once the **Node Manager** gets the **Resources**, it sends them to the **Resource Manager**.



<u>N/B:</u>

In addition to these systems, Hadoop ecosystem has various other big data tools and frameworks, dedicated to managing, processing and analyzing data.

- ✓ Hive
- ✓ Pig
- ✓ Apache Spark
- ✓ Flume
- ✓ Sqoop
- ✓ …



Examples of Hadoop in Action

Example 1: Log Processing

- **Problem:** A website generates terabytes of log data every day. The company wants to analyze user activity patterns.
- **Solution:**
    - ✓ Logs are stored in HDFS.
    - ✓ A MapReduce job processes the logs to find trends such as peak traffic times, most visited pages, etc.
    - ✓ Data is then visualized using tools like Tableau or integrated into a data warehouse.

Example 2: Fraud Detection

- **Problem:** A bank wants to identify fraudulent credit card transactions in real time.
- **Solution:**
    - ✓ Transaction data is streamed into the Hadoop ecosystem (e.g., using Apache Kafka).
    - ✓ Machine learning models are trained on historical data stored in HDFS.
    - ✓ Spark (integrated with Hadoop) processes new transactions in real time, flagging suspicious activities.

Example 3: Retail Analytics

- **Problem:** An e-commerce company wants to optimize inventory by predicting future demand.
- **Solution:**
    - ✓ Sales and inventory data are ingested into Hadoop.
    - ✓ MapReduce jobs calculate historical sales trends.
    - ✓ Insights are used to optimize stock levels and improve supply chain efficiency.

Example 4: Social Media Sentiment Analysis

- **Problem:** A brand wants to analyze customer sentiment from millions of tweets and Facebook posts.
- **Solution:**
  - ✓ Data from social media platforms is collected and stored in HDFS.
  - ✓ Text mining and natural language processing (NLP) are performed using Apache Mahout or Spark MLlib.
  - ✓ The sentiment analysis results help improve customer engagement strategies.

Advantages of Hadoop

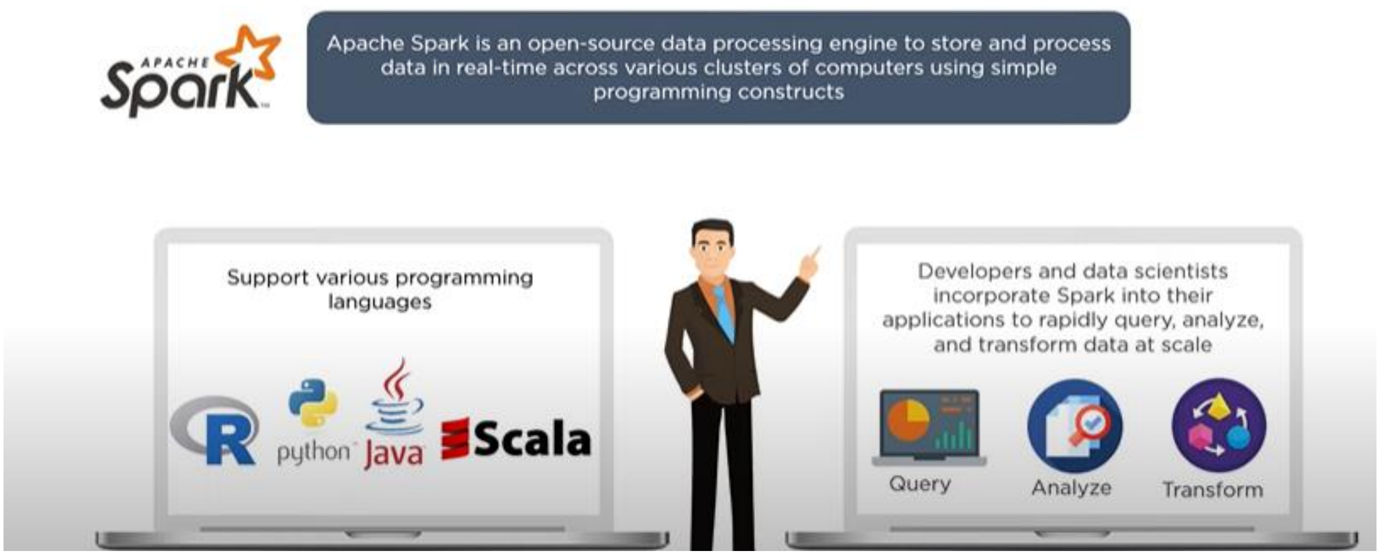
1. **Scalability:** Easily scales to accommodate petabytes of data.
2. **Fault Tolerance:** Replicates data across nodes to prevent data loss in case of hardware failures.
3. **Cost-Effective:** Works with inexpensive commodity hardware.
4. **Flexibility:** Supports structured, semi-structured, and unstructured data.
5. **High Throughput:** Efficiently processes large volumes of data.

Real-World Users

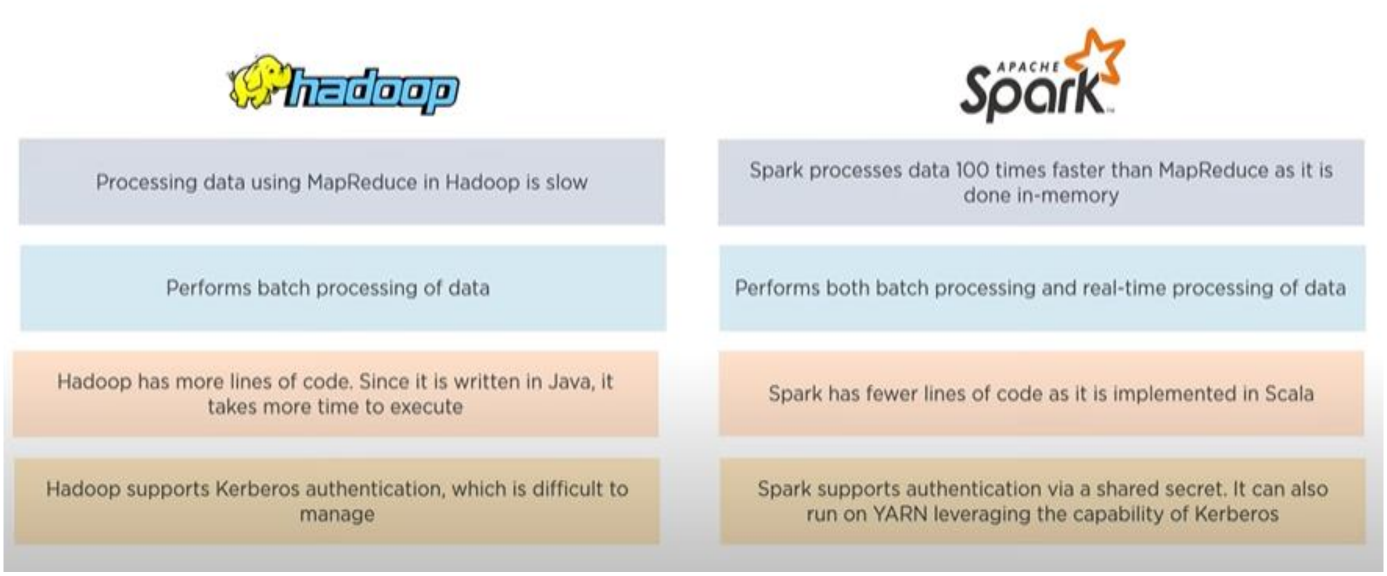
- **Facebook:** Analyzing user interactions to improve social recommendations.
- **Netflix:** Personalizing movie recommendations.
- **eBay:** Optimizing search results and fraud detection.
- **Yahoo!:** Analyzing massive web search indexes.

Apache Spark

- ✓ Def1: Open-source data processing engine to store and process data in real-time across various clusters of computers using simple programming constructs.
- ✓ Def2: Open-source, unified analytics engine designed for large-scale data processing. It offers high-level APIs in Java, Scala, Python, and R, and an optimized engine that supports general execution graphs. (Main Language is Scala but supports all the others)
- ✓ Main Reason for Its development (2009) was to address the Limitations of Hadoop (2005)
- ✓ In memory processing makes it 100X faster than hadoop



Comparison between Hadoop and Spark

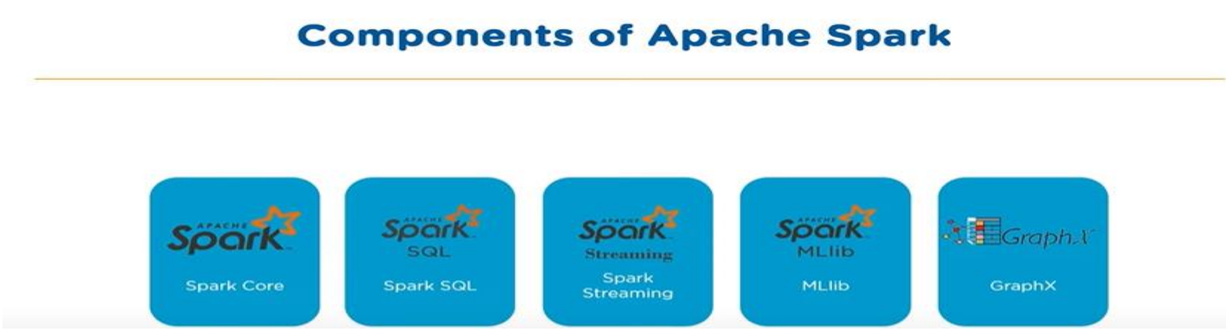


Spark Features/Advantages

- **Fast Processing**: Spark Contains Resilient Distributed Datasets (RDD) which saves time taken in reading, and writing operations and hence, it runs almost 10 to 100 times faster than Hadoop.
- **In-Memory Computing**: In spark data is stored in the RAM, so it can access the data quickly and accelerate the speed of analytics.
- **Flexible**: Spark supports **multiple languages** and allows the developers to write applications in Java, Scala, R, or Python.
- **Fault Tolerance**: Spark contains Resilient Distributed Datasets (RDD) that are designed to handle the failure of any worker node in the cluster. Thus, it ensures that the loss of data reduces to zero. RDDs (Execution logic/Temporary datasets: Data is loaded into them when execution happens) are distributed across multiple nodes.
- **Better Analytics:** Has a rich set of SQL Queries, Machine Learning Algorithms, Complex Analytics etc. (With all functionalities, Analysis can be performed better).

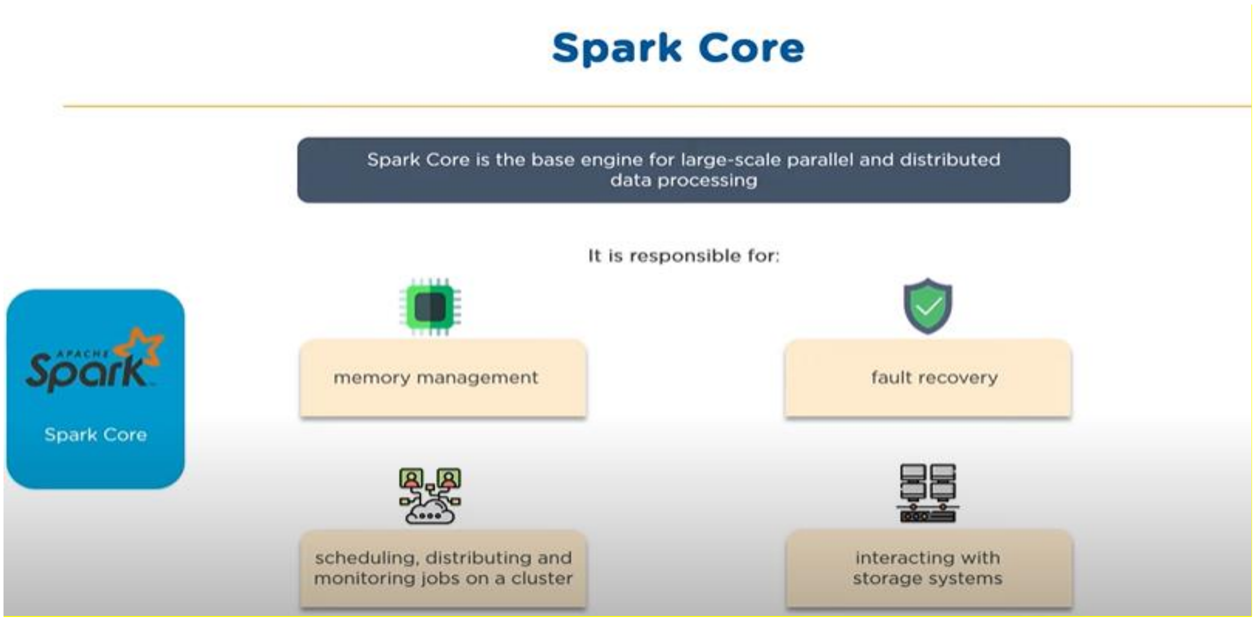
Components of Spark

- **Sparks Core: Core engine which has RDDs**
- **Spark SQL: Working with structured data (Has Data Frame Features)**
- **Spark Streaming: Allows for creation of spark streaming Applications which not only works on data being streamed in or constantly generated but you could also transform/analyze/process the data as it comes in smaller chunks.**
- **Spark MLlib: Allows Data Scientist to build ML algorithms for predictive/Descriptive or Preemptive analytics or Recommendation Systems.**
- **GraphX: I.e. LinkedIn & Twitter where the data naturally has like a network kind of flow (Graph in this context refer to network related data). Data that can be networked together (Some kind of relationship) can be processed using Graph Base Processing thru' Spark GraphX.**

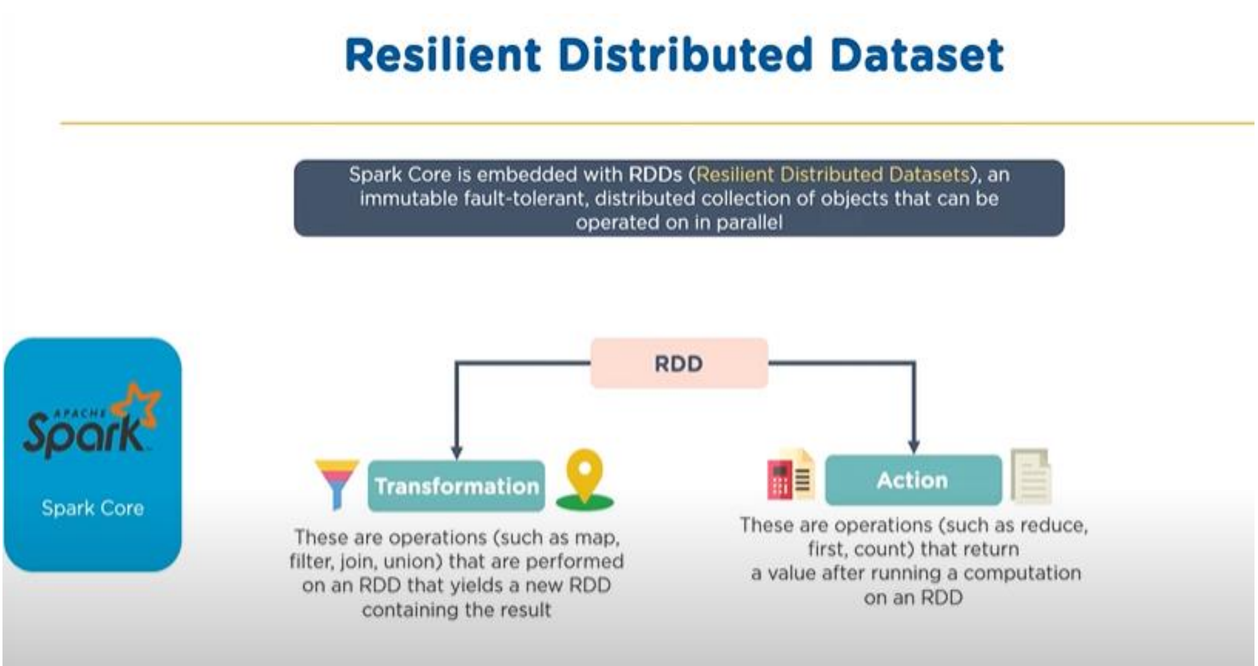


N/B

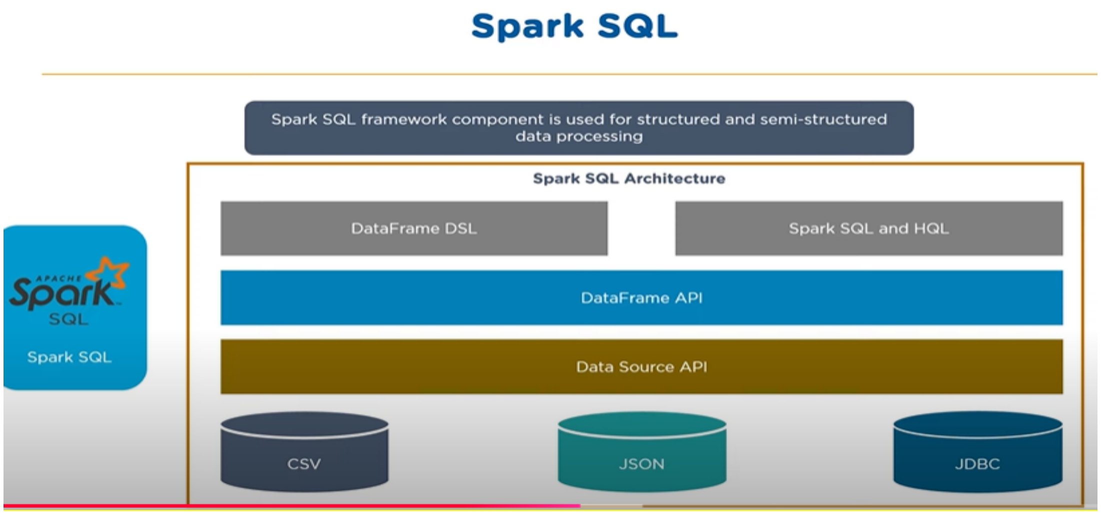
Spark by itself does not have its own storage. It relies on storage. The storage could be: HDFS, MySQL, Cassandra (NoSQL) etc. You just connect your Spark then Fix the Data, Extract the data Process it and analyze.



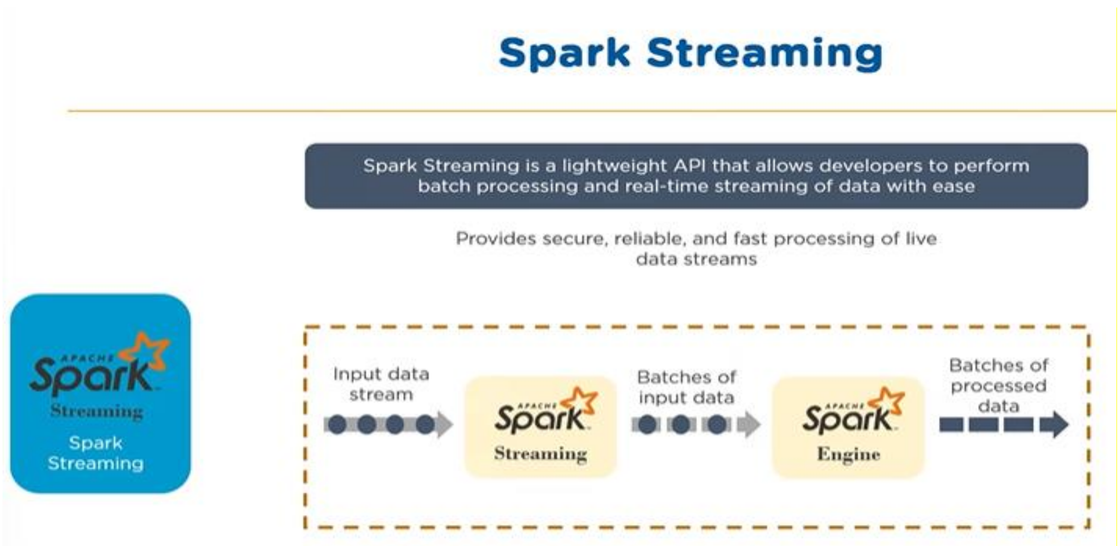
Resilient Distributed Dataset (RDD)



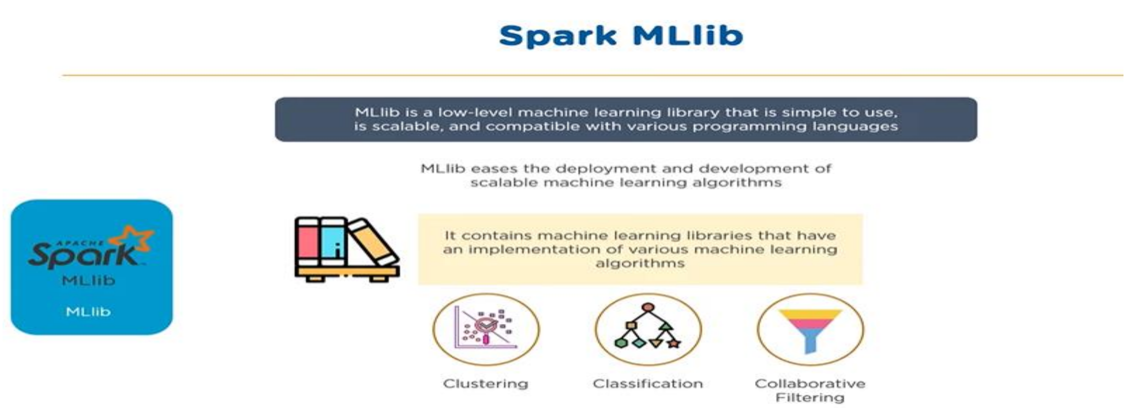
Spark SQL



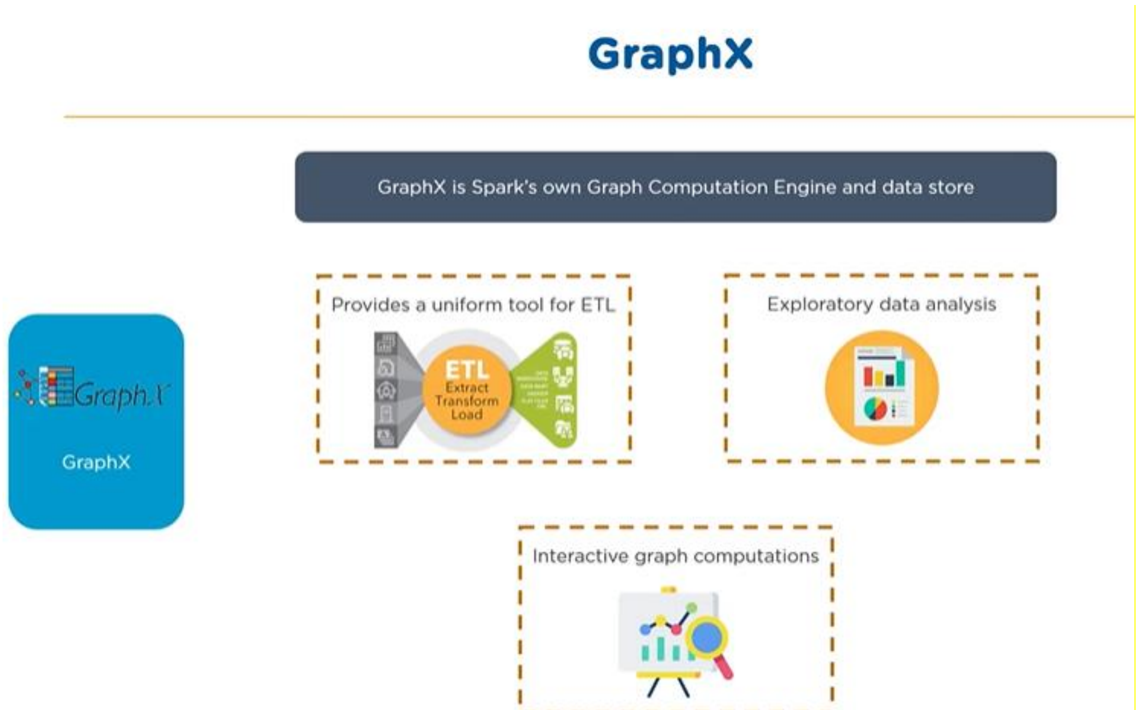
Spark Streaming



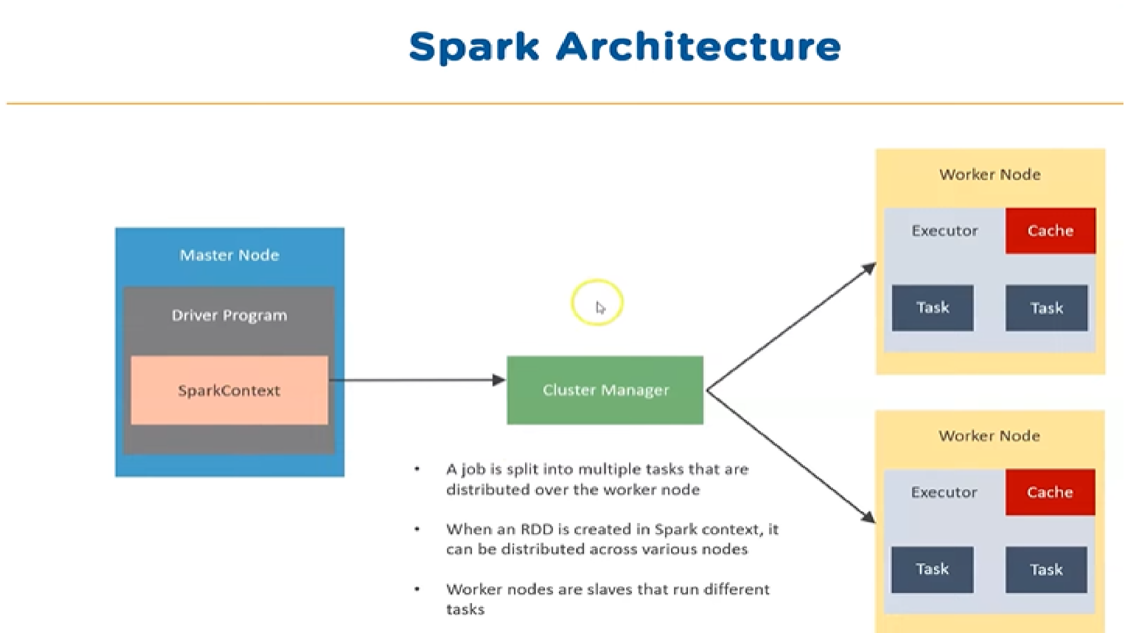
Spark MLlib



GraphX

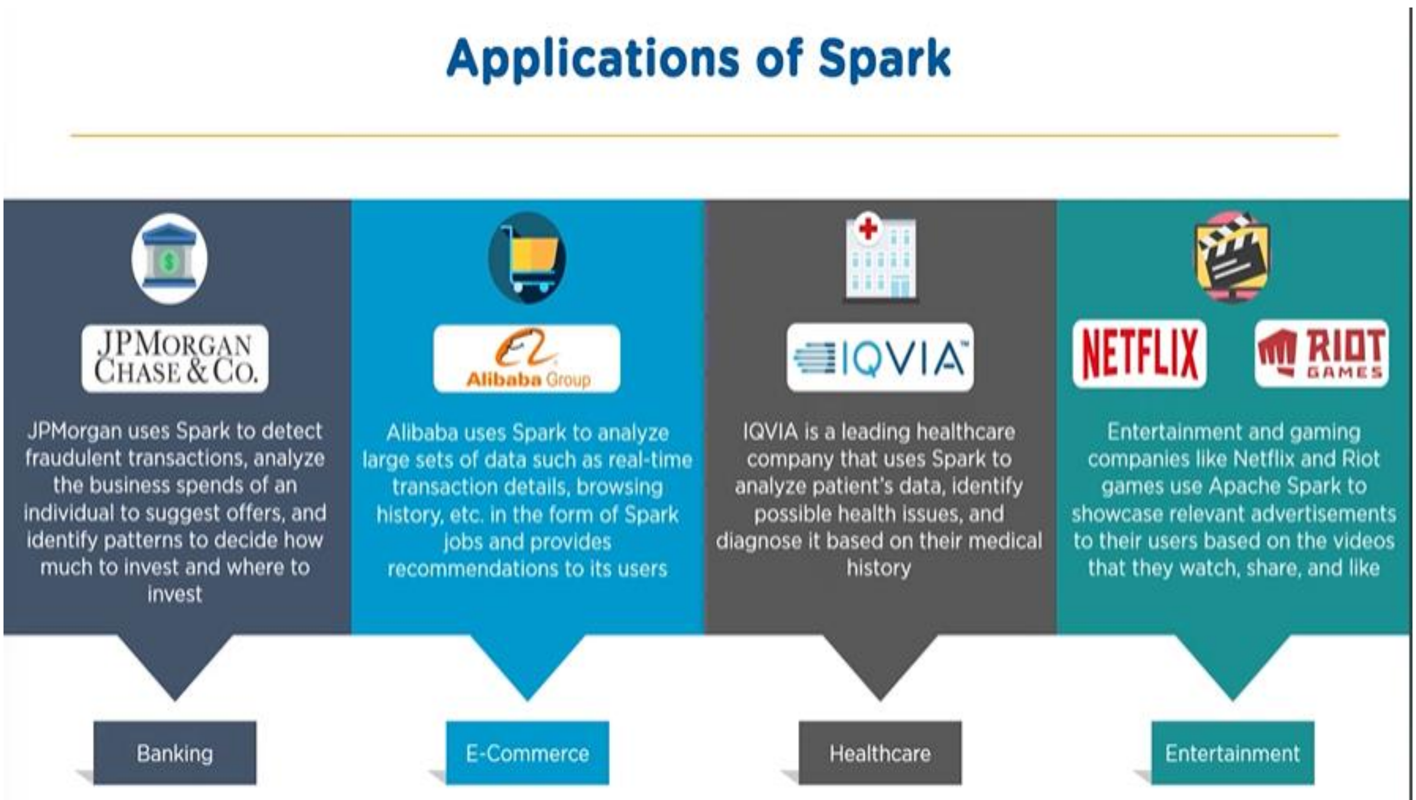


SPARK ARCHITECTURE



Spark Can also work in Stand Alone without Hadoop

Application of Spark



Apache Kafka

- Kafka is a high-performance, real-time messaging system. It is an open-source tool and part of Apache projects.

Characteristics are:

- ✓ It is a distributed and partitioned Messaging System
- ✓ It is highly fault-tolerant
- ✓ It is highly Scalable
- ✓ It can process and send millions of messages per second to several receivers.

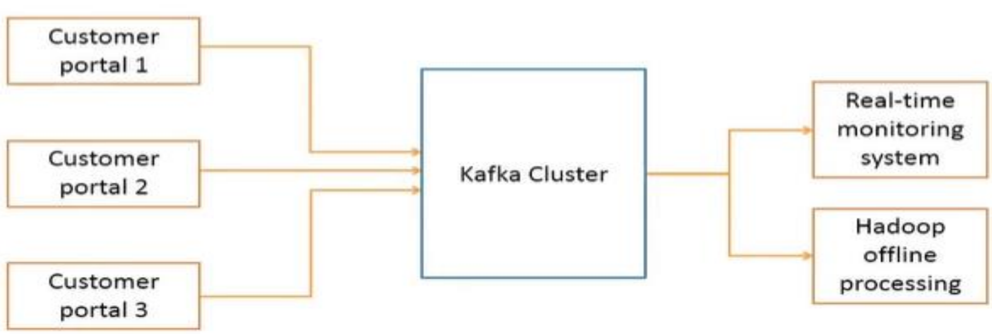
- Originally developed by LinkedIn and later handed over to the open-source community in early 2011.

Kafka Use Cases

Kafka can be used for various purposes in an organization, such as:

Messaging service	Millions of messages can be sent and received in real-time, using Kafka.
Real-time stream processing	Kafka can be used to process a continuous stream of information in real-time and pass it to stream processing systems such as Storm.
Log aggregation	Kafka can be used to collect physical log files from multiple systems and store them in a central location such as HDFS.
Commit log service	Kafka can be used as an external commit log for distributed systems.
Event sourcing	A time ordered sequence of events can be maintained through Kafka.

Kafka can be used to aggregate user activity data such as clicks, navigation, and searches from different websites of an organization; such user activities can be sent to a real-time monitoring system and hadoop system for offline processing.



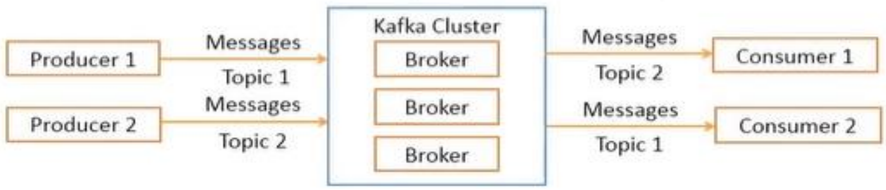
Kafka Data Model

Kafka Data Model

simpl

The Kafka data model consists of messages and topics.

- Messages represent information such as, lines in a log file, a row of stock market data, or an error message from a system.
- Messages are grouped into categories called topics. Example: LogMessage and StockMessage.
- The processes that publish messages into a topic in Kafka are known as producers.
- The processes that receive the messages from a topic in Kafka are known as consumers.
- The processes or servers within Kafka that process the messages are known as brokers.
- A Kafka cluster consists of a set of brokers that process the messages.

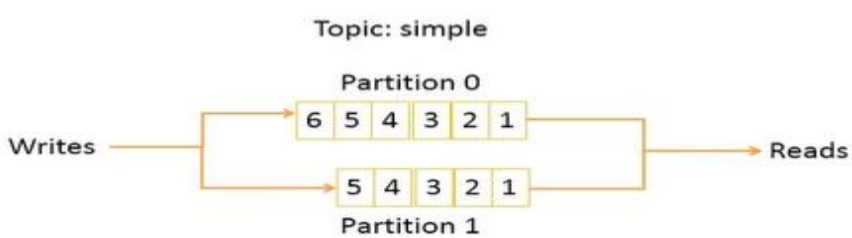


Topics in Kafka

Topics

A topic is a category of messages in Kafka.

- The producers publish the messages into topics.
- The consumers read the messages from topics.
- A topic is divided into one or more partitions.
- A partition is also known as a commit log.
- Each partition contains an ordered set of messages.
- Each message is identified by its offset in the partition.
- Messages are added at one end of the partition and consumed at the other.



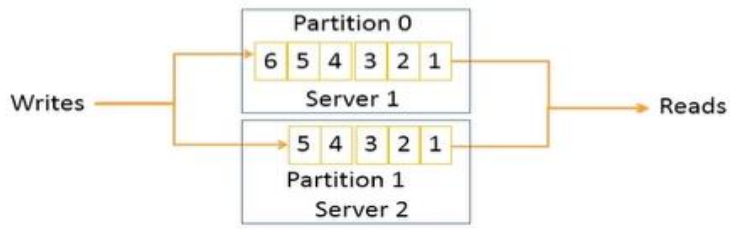
- Topics are divided into partitions in Kafka

Topic Partitions

Partition Distribution

Partitions can be distributed across the Kafka cluster.

- Each Kafka server may handle one or more partitions.
- A partition can be replicated across several servers for fault-tolerance.
- One server is marked as a leader for the partition and the others are marked as followers.
- The leader controls the read and write for the partition, whereas, the followers replicate the data.
- If a leader fails, one of the followers automatically become the leader.
- Zookeeper is used for the leader selection.



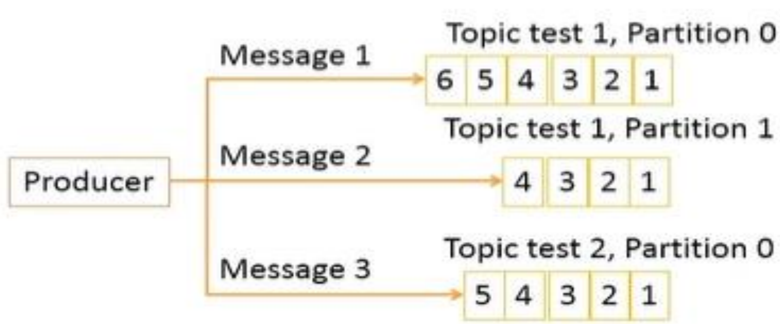
- Producers are the creators of message in Kafka

Producers

Producers

The producer is the creator of the message in Kafka.

- The producers place the message to a particular topic.
- The producers also decide which partition to place the message into.
- Topics should already exist before a message is placed by the producer.
- Messages are added at one end of the partition.



Consumers

Consumers

The consumer is the receiver of the message in Kafka.

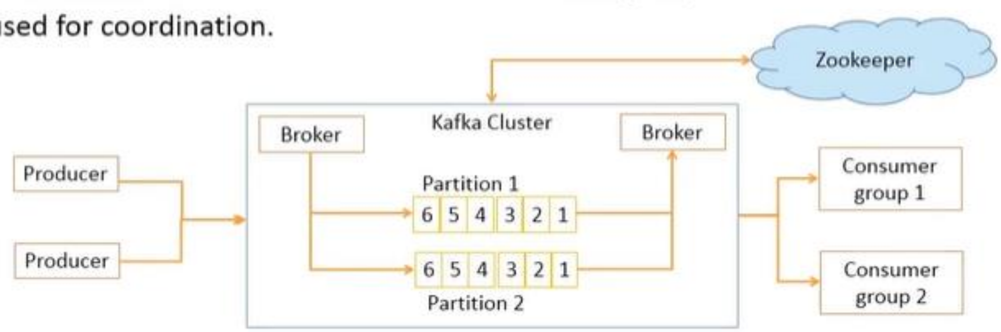
- Each consumer belongs to a consumer group.
- A consumer group may have one or more consumers.
- The consumers specify what topics they want to listen to.
- A message is sent to all the consumers in a consumer group.
- The consumer groups are used to control the messaging system.

KAFKA ARCHITECTURE

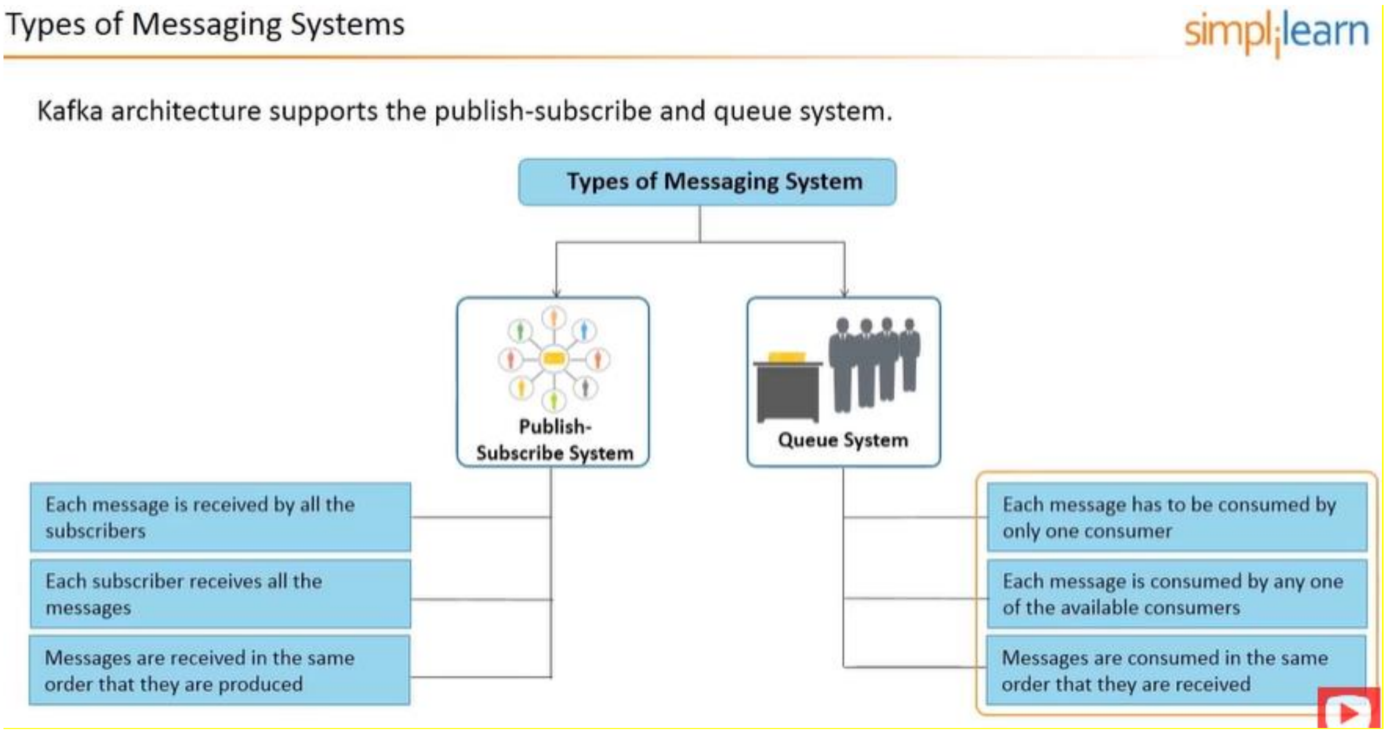
Kafka Architecture

Kafka architecture consists of brokers that take messages from the producers and add to a partition of a topic. Brokers provide the messages to the consumers from the partitions.

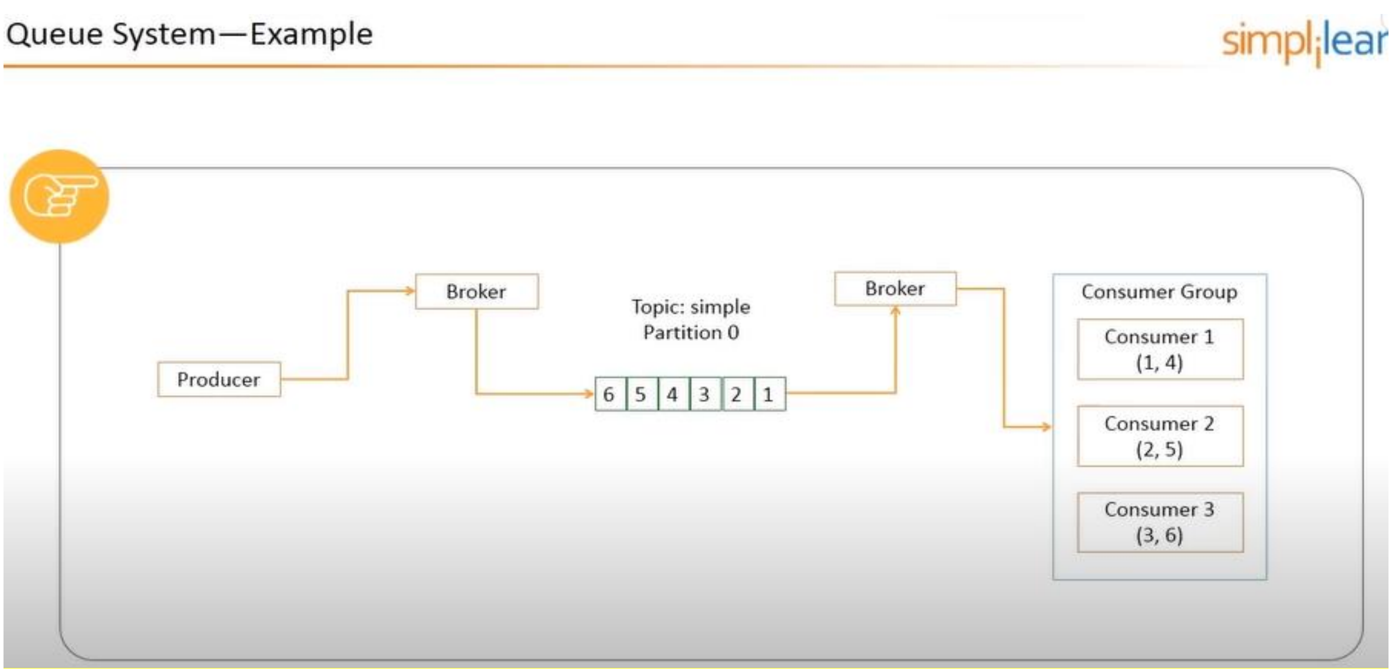
- A topic is divided into multiple partitions.
- The messages are added to the partitions at one end and consumed in the same order.
- Each partition acts as a message queue.
- Consumers are divided into consumer groups.
- Each message is delivered to one consumer in each consumer group.
- Zookeeper is used for coordination.



Type of Messaging

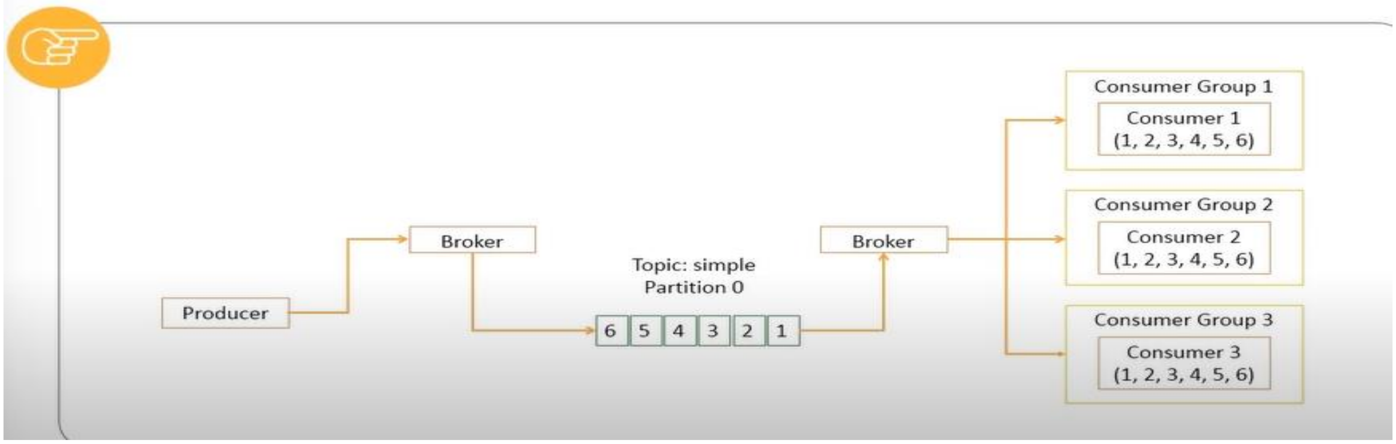


Queue System



Publish-Subscribe System

Publish-Subscribe System—Example



Brokers

Brokers

Brokers are the Kafka processes that process the messages in Kafka.

- Each machine in the cluster can run one broker.
- They coordinate among each other using Zookeeper.
- One broker acts as a leader for a partition and handles the delivery and persistence, whereas, the others act as followers.
- Brokers receive the message from the producer and send it to consumer groups.

Kafka Guarantees

Kafka Guarantees

Kafka guarantees the following:

1. Messages sent by a producer to a topic and a partition are appended in the same order
2. A consumer instance gets the messages in the same order as they are produced
3. A topic with replication factor N, tolerates upto N-1 server failures

Application

Kafka at LinkedIn

Kafka is used by LinkedIn to manage streams of information.

Some of the uses of Kafka at LinkedIn are as follows:

Monitoring
- Collect metrics
- Create monitoring dashboards

Messaging
- Used for message queues in content feeds
- As publish-subscribe system for searches and content feeds

Analytics
- Collection of page views and clicks
- Store into a central hadoop-based analytics system

A building block for distributed applications
- For distributed databases
- For distributed log systems

Replication in Kafka

Replication in Kafka

Kafka uses the primary-backup method of replication.

- One machine (one replica) is called a leader and is chosen as the primary; the remaining machines (replicas) are chosen as the followers and act as backups.
- The leader propagates the writes to the followers.
- The leader waits until the writes are completed on all the replicas.
- If a replica is down, it is skipped for the write until it comes back.
- If the leader fails, one of the followers will be chosen as the new leader; this mechanism can tolerate n-1 failures if the replication factor is ‘n’.

Persistence in kafka

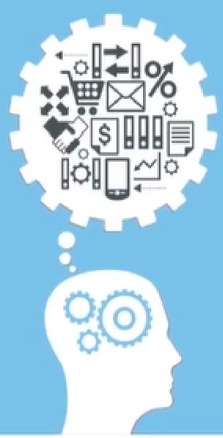
Persistence in Kafka

Kafka uses the Linux file system for persistence of messages.

- Persistence ensures no messages are lost.
- Kafka relies on the file system page cache for fast reads and writes.
- All the data is immediately written to a file in file system.
- Messages are grouped as message sets for more efficient writes.
- Message sets can be compressed to reduce network bandwidth.
- A standardized binary message format is used among producers, brokers, and consumers to minimize data modification.

Summary

Here is a quick recap of what we have learned in this lesson:



- Kafka is a high-performance, real-time messaging system.
- Kafka can be used as an external commit log for distributed systems.
- Kafka data model consists of messages and topics.
- Kafka architecture consists of brokers that take messages from the producers and add to a partition of a topic.
- Kafka architecture supports two types of messaging system called publish-subscribe and queue system.
- Brokers are the Kafka processes that process the messages in Kafka.

Cassandra