**HADOOP AND SPARK**

To understand the concept of the application and usage of HADOOP we first need to understand what BIG DATA is.

**BIG DATA:**

Everything on and off the Internet is BIG DATA.

BIG DATA is the collection of data which is huge in volume and is still excessively growing with every passing second.

In simpler words, we can say data generated in the form of files, photos, videos, audios, emails and what not are a part of big data.

We can take your mobile phone as an example, which contains every form of data and when multiplied to 5 million users making this data highly complex.

This big data is huge and complex which makes it impossible to process or manage for traditional systems.

WE CAN CLASSIFY DATA AS BIG DATA based on following concepts, called the 5 Vs:

1. Volume
2. Velocity
3. Variety
4. Veracity
5. Value

**Hospital example.**

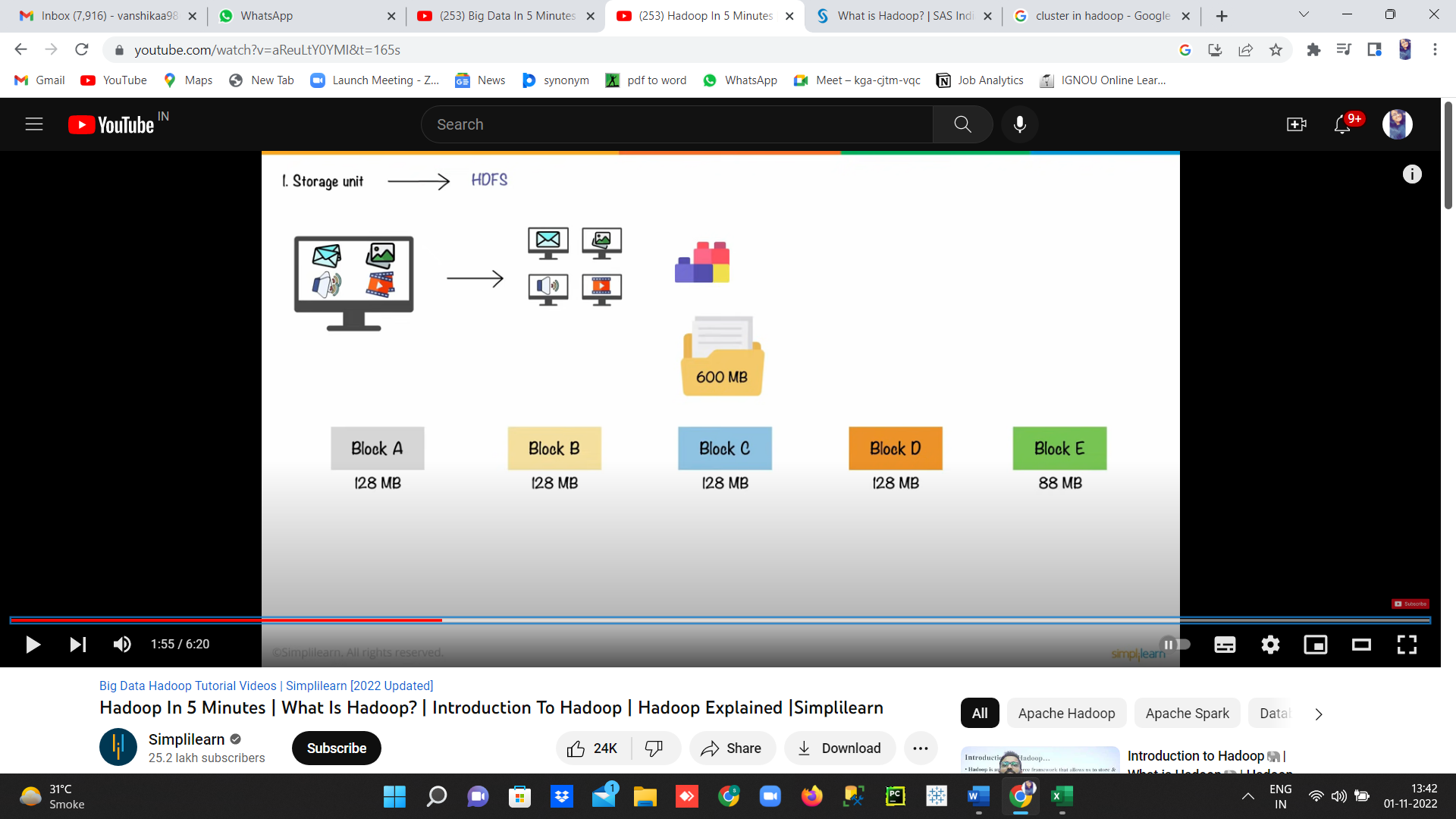
This briefly defines what BIG DATA is.

Now, lets understand what is the role of HADOOP in managing this massive data.

**HADOOP is basically an open-source framework for managing and processing big data using a cluster of commodity hardware.**

HADOOP has **three components** which were specifically designed to work with big data.

1. Storage Unit: **HDFS (HADOOP DISTRIBUTED FILE SYSTEM)**

HDFS stores data in many computers or systems into different blocks.

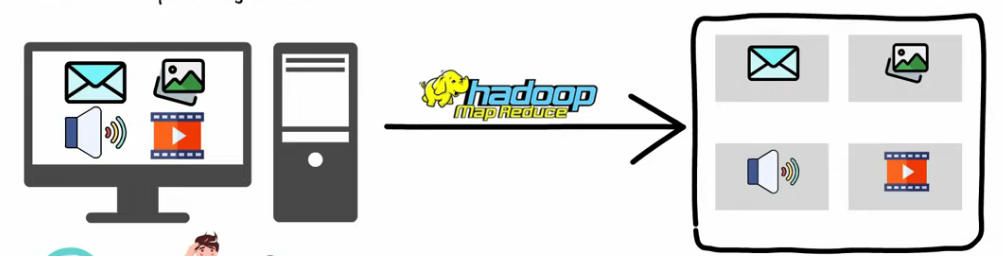
Example

HDFS stores copies of data and stores it across multiple computer systems which makes it **fault-tolerant.**

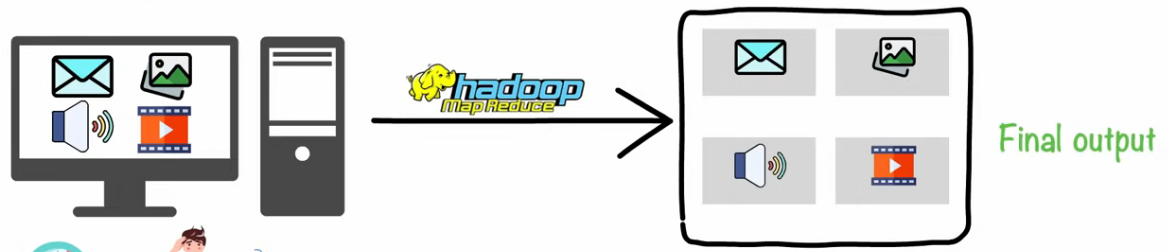


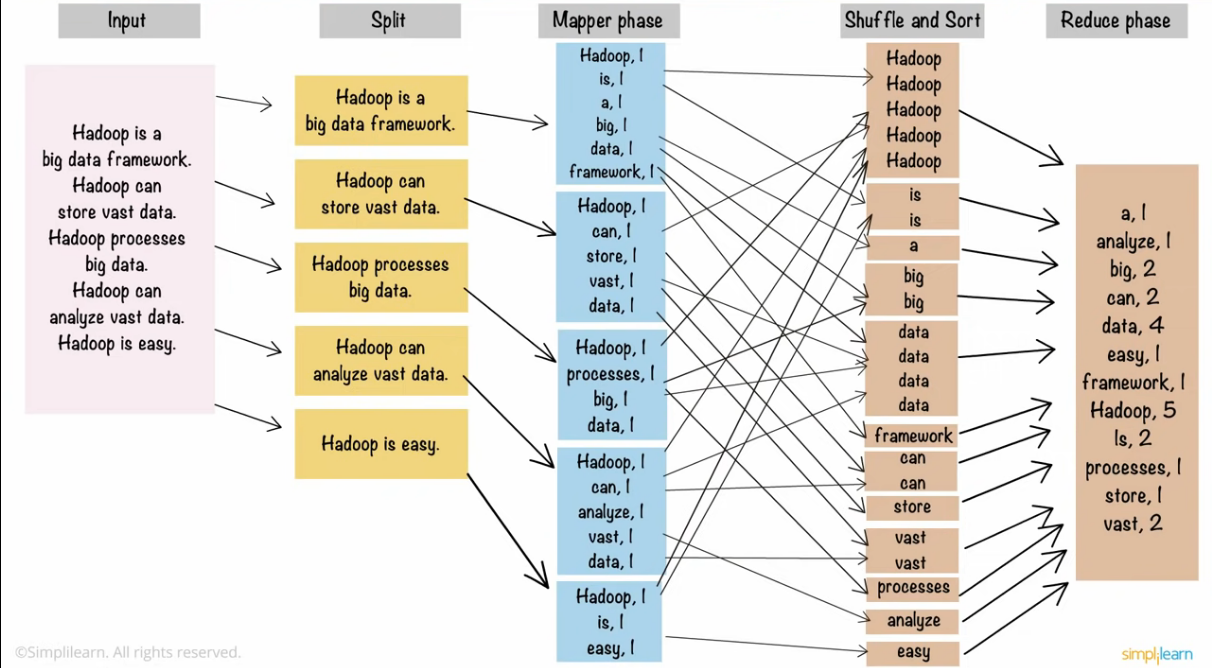
Once the data is stores successfully then comes the part of processing this huge data and here enters the use of second component of HADOOP which is

1. Processing Unit - MAPREDUCE: Hadoop basically processes different types of data into **different nodes**. In other words, MapReduce **splits data** into different parts and process them separately to make the process more **efficient and easier.**

****

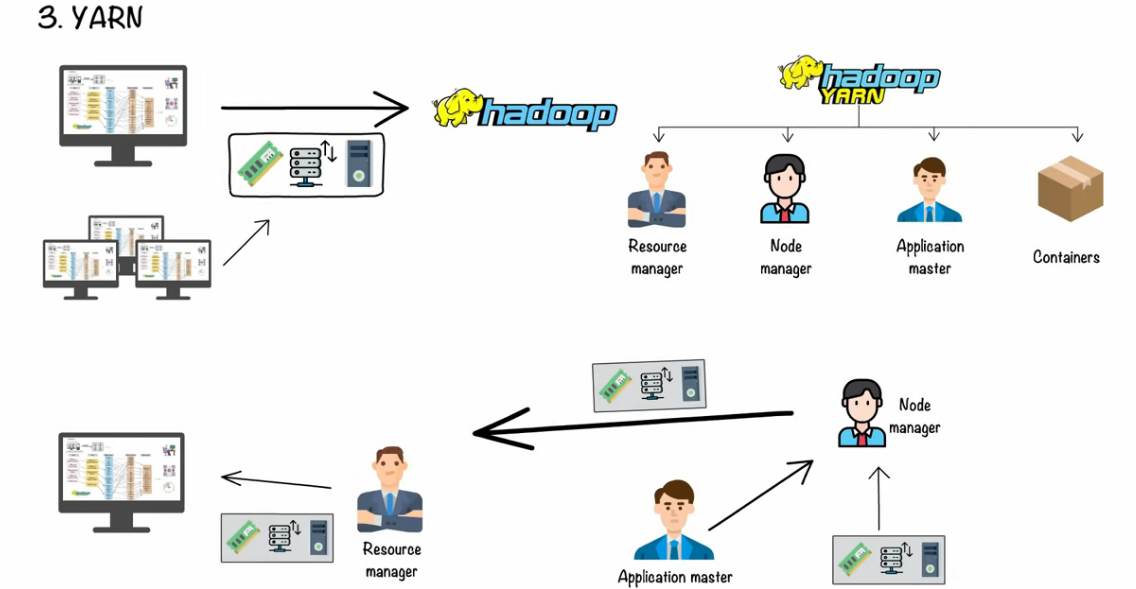
**This separated data is used to derive required results and these results are aggregated to give a final output.**

****

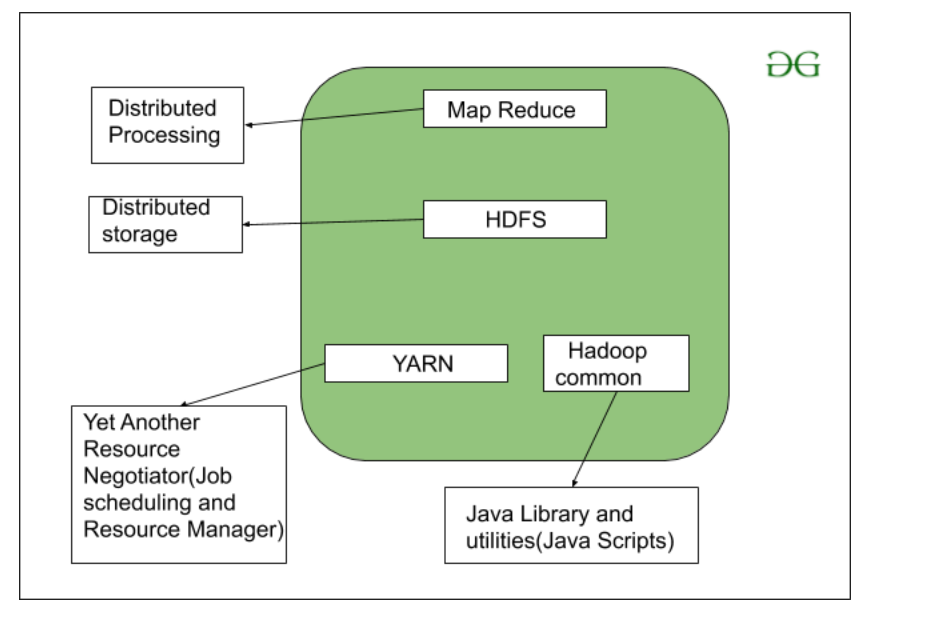
****

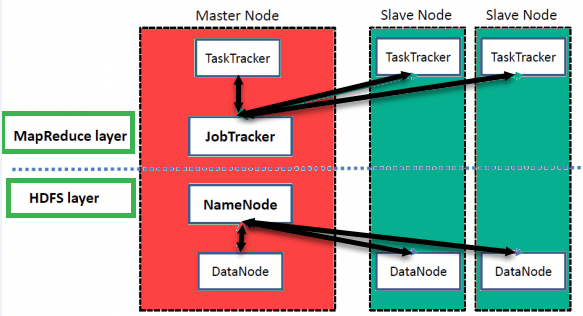
**This improves load balances and saves a considerable amount of time.**

**Once we have the separated and understandable data, we need to manage the resources.**

1. **Resource negotiator: YARN:** This consists of resource manager, node manager, application manager and container.

**CLUSTER IS THE COLLECTION OF NODES.**



**HADOOP ARCHITECTURE**

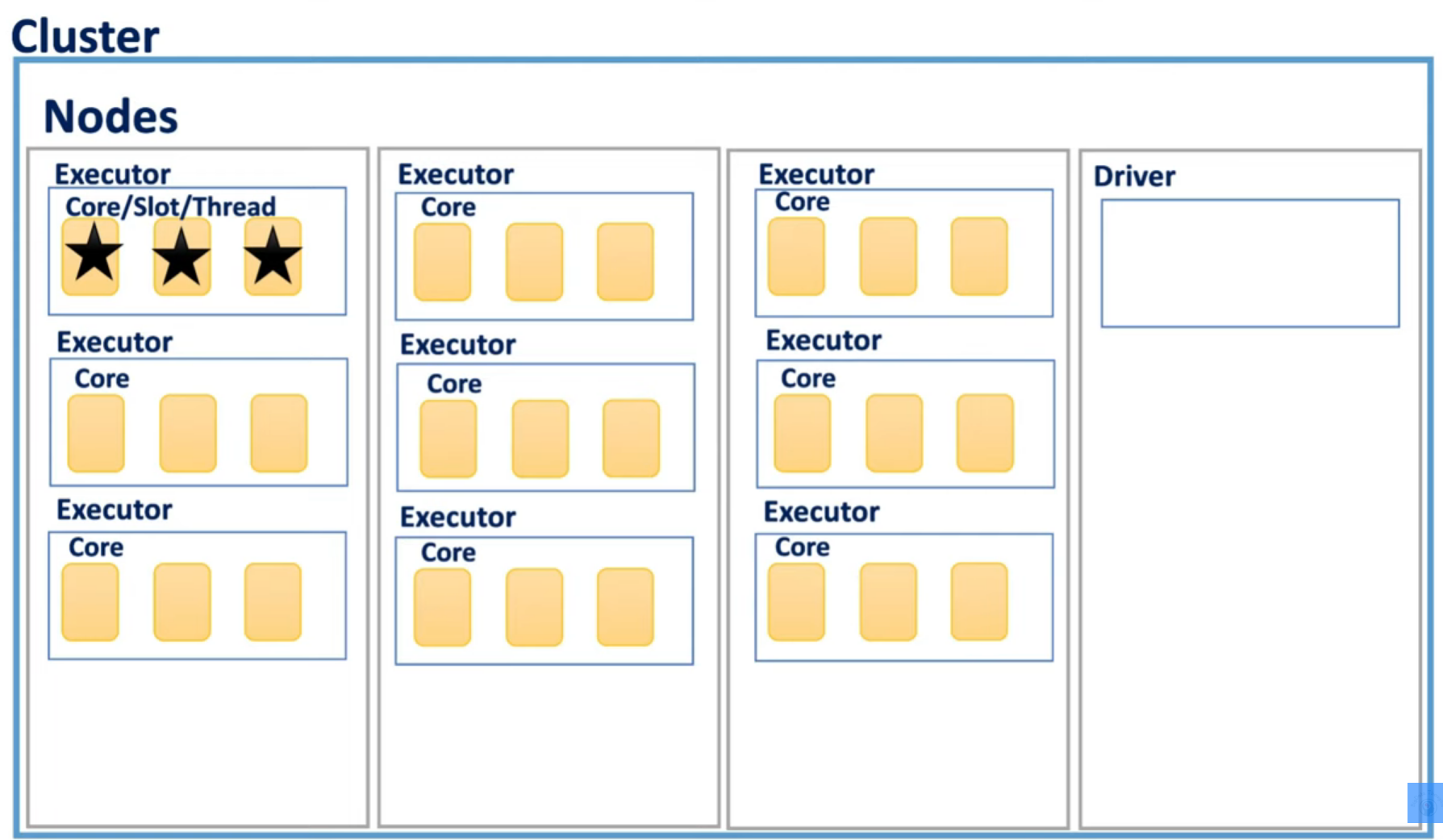
**It uses DISTRIBUTED STORAGE.** Distributed storage is when the data is stored in different computer systems rather than storing whole data at a single place.

* **NameNode**: NameNode is basically a master node that acts like a monitor and supervises operations performed by DataNodes.
* **Secondary NameNode:** A Secondary NameNode plays a vital role in case there is some technical issue in the NameNode.
* **DataNode:** DataNode is the slave node that stores all files and processes.
* **Mapper:** Mapper maps data or files in the DataNodes. It will go to every DataNode and run a particular set of codes or operations in order to get the work done.
* **Reducer:** While a Mapper runs a code, a Reducer is required for getting the result from each Mapper.
* **JobTracker:** JobTracker is a master node used for getting the location of a file in different DataNodes. It is a very important service in Hadoop as if it goes down, all the running jobs will get halted.
* **TaskTracker:** TaskTracker is a reference for the JobTracker present in the DataNodes. It accepts different tasks, such as map, reduces, and shuffle operations, from the JobTracker. It is a key player performing the main MapReduce functions.
* **Block:** Block is a small unit wherein the files are split. It has a default size of 64 MB and can be increased as needed.
* **Cluster:** A cluster is a set of machines such as DataNodes, NameNodes, Secondary NameNodes, etc.

**SPARK:**

Spark is basically a member of the HADOOP ecosystem, which is designed to overcome the loopholes and drawbacks of HADOOP.

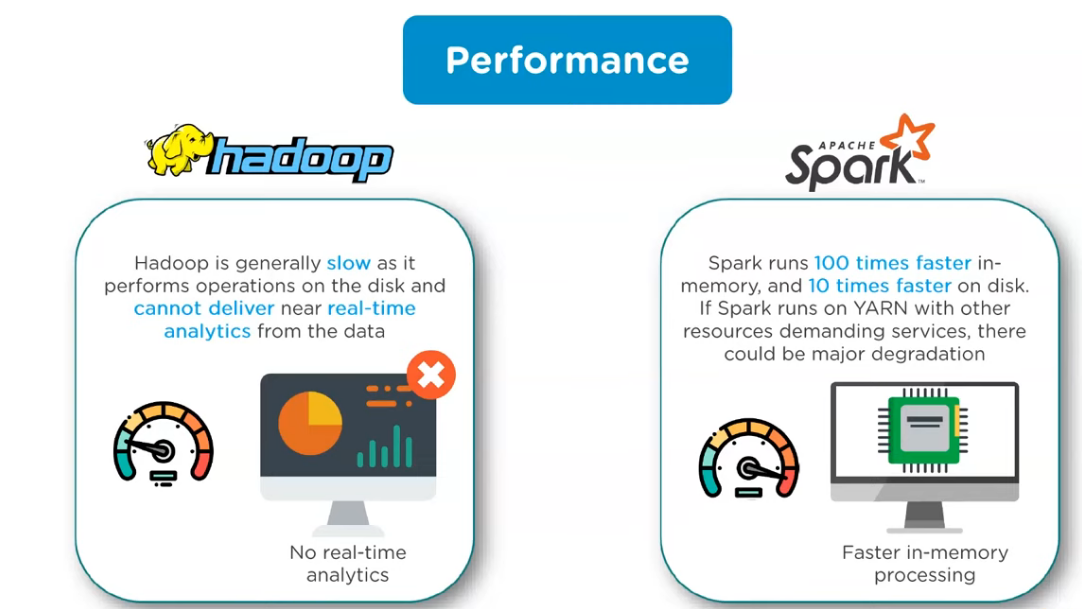
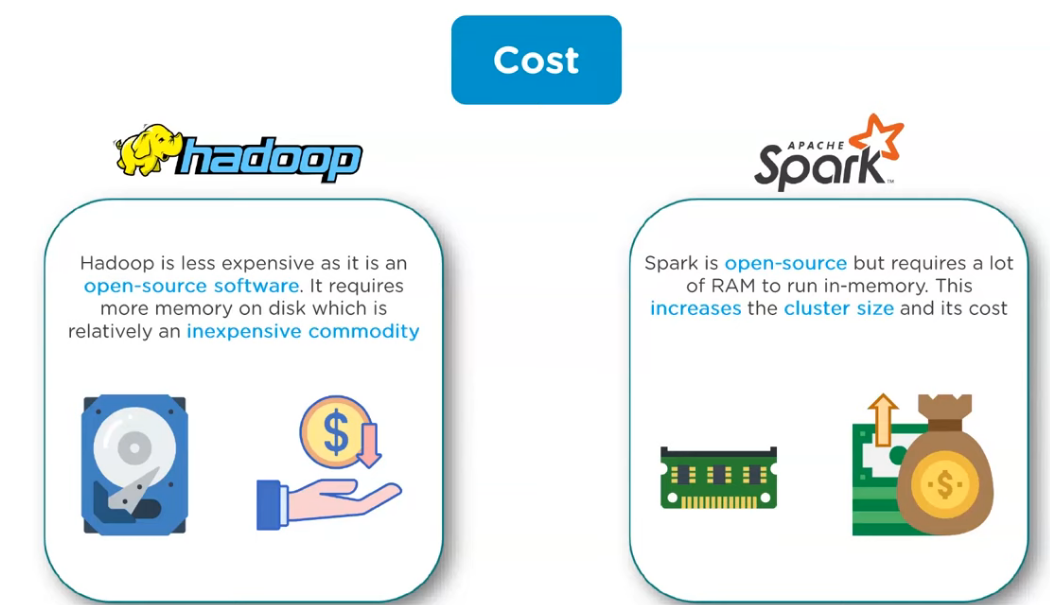
Spark is an open-source framework focused on **interactive query, machine learning, and real-time workloads**. It does not have its own storage system, but runs analytics on other storage systems like HDFS, or other popular stores like Amazon Redshift, Amazon S3, Couchbase, Cassandra, and others.

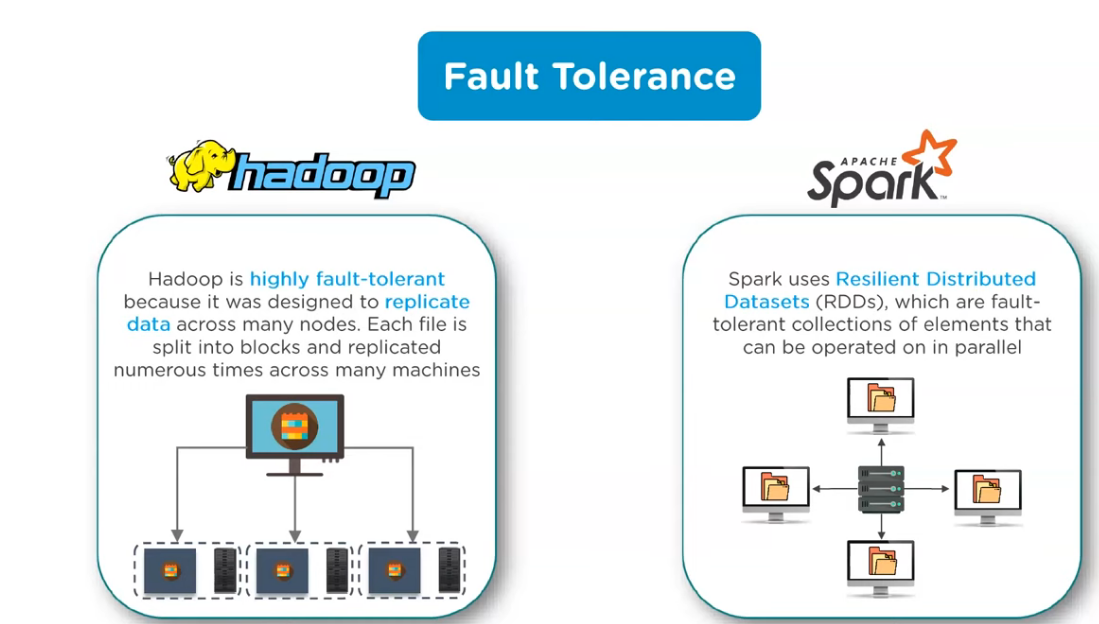
**WORKING OF SPARK**

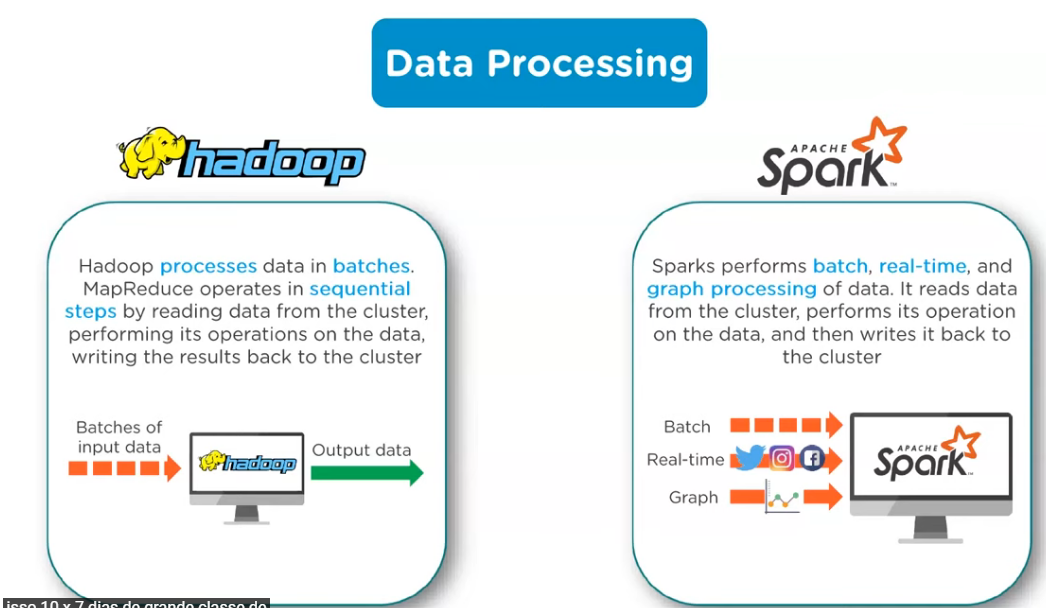
**JOB is the actually high-level task to be performed on the organised data set.**

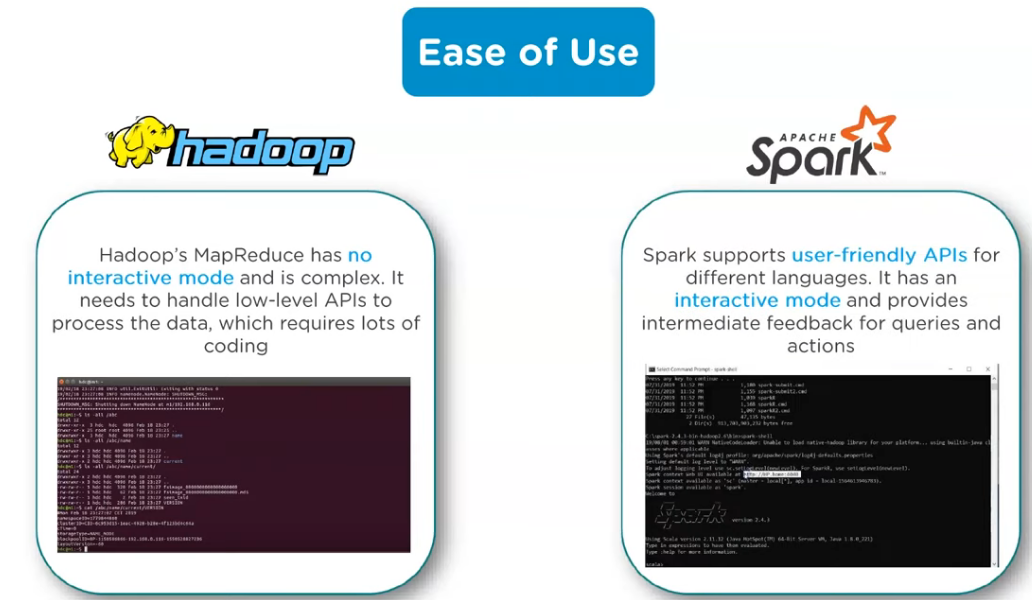
**And then it is divided into multiple stages we want the operations we want to group by.**

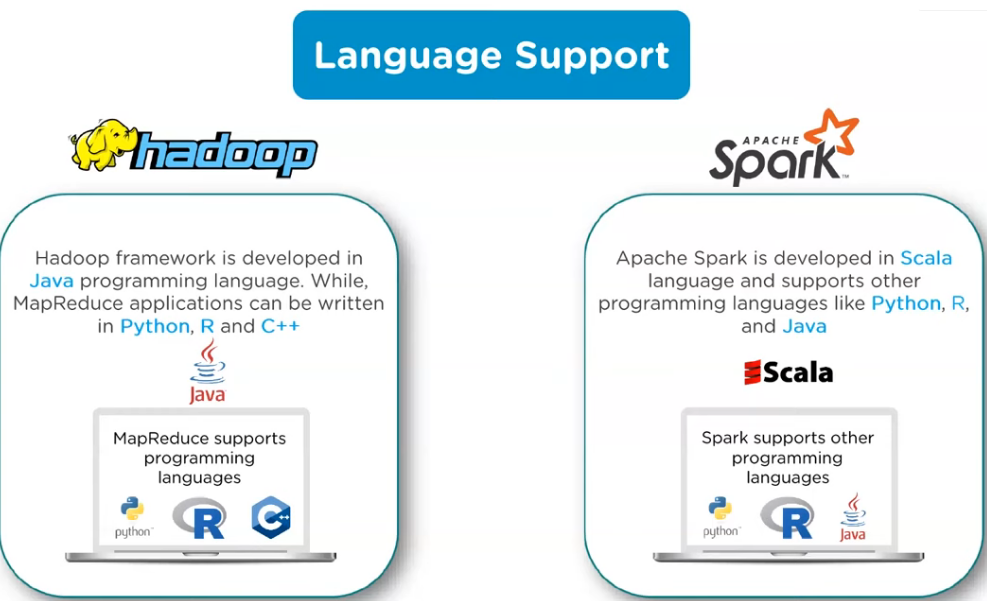
**Stages are further divided into number of tasks.**

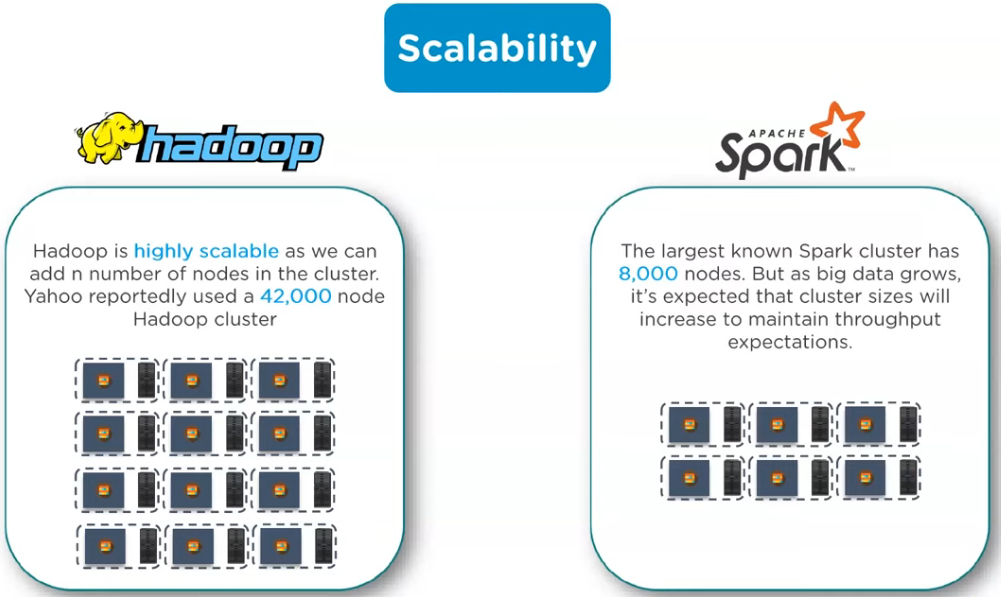
**DIFFERENCE BETWEEN SPARK AND HADOOP**

****

****

****

****





| Basis | Hadoop | Spark |
| --- | --- | --- |
| Processing Speed | Hadoop’s MapReduce model reads and writes from a disk, thus slowing down the processing speed. | Spark reduces the number of read/write cycles to disk and stores intermediate data in memory, hence faster-processing speed. |
| Usage | Hadoop is designed to handle batch processing efficiently. | Spark is designed to handle real-time data efficiently. |
| Latency | Hadoop is a high latency computing framework, which does not have an interactive mode. | Spark is a low latency computing and can process data interactively. |
| Data | With Hadoop MapReduce, a developer can only process data in batch mode only. | Spark can process real-time data, from real-time events like Twitter, and Facebook. |
| Cost | Hadoop is a cheaper option available while comparing it in terms of cost | Spark requires a lot of RAM to run in-memory, thus increasing the cluster and hence cost. |
| Algorithm Used | The PageRank algorithm is used in Hadoop. | Graph computation library called GraphX is used by Spark. |
| Fault Tolerance | Hadoop is a highly fault-tolerant system where data is replicated across the nodes and used the data in case of any issue. | Spark uses a DAG to rebuild the data across the nodes. |
| Security | Hadoop supports LDAP, ACLs, SLAs, etc and hence it is extremely secure. | Spark is not secure, it relies on the integration with Hadoop to achieve the necessary security level. |
| Machine Learning | Data fragments in Hadoop can be too large and can create bottlenecks. Thus, it is slower than Spark. | Spark is much faster as it uses MLib for computations and has in-memory processing. |
| Performance | Hadoop has a slower performance as it uses disk for storage and depends upon disk read and write operations. | It has fast performance with reduced disk reading and writing operations. |
| Scalability | Hadoop is easily scalable by adding nodes and disk for storage. It supports tens of thousands of nodes. | It is quite difficult to scale as it relies on RAM for computations. It supports thousands of nodes in a cluster. |
| Language support | It uses Java or Python for MapReduce apps. | It uses Java, R, Scala, Python, or Spark SQL for the APIs. |
| User-friendliness | It is more difficult to use. | It is more user-friendly. |
| Resource Management | YARN is the most common option for resource management. | It has built-in tools for resource management. |

**CONCLUSION:**

One should try to store data in the HADOOP HDFS and process the data in SPARK.

**MOST USED PROGRAMMING LANGUAGES FOR Processing and managing BIG DATA**

1. PYTHON
2. JAVA
3. R
4. SCALA

**SCALA**

Scala **allows interaction between distributed databases and empowers parallel data processing to reduce time**. The language is known for big data processing and containing large data into scalable volumes to make decisions.

SCALA BECOME SO POPULA WHEN JAVA CODES STARTED BECOMING COMPEX AND LENTHY.

Scala minimised the code length and the execution time.

RUBY ALSO STARTED BECOMING INEFFECTIVE WHEN HUGE AMOUNT OF DATA STARTED FLOWING IN.

So, **now SCALA BECOME THE PROGRAMMING LANGUAGE FOR THE DEVELOPERS TO MANAGE TASKS AND PROCESSING DATA USING HADOOP AND SPARK.**

**NOTE: 1. HADOOP CAN WORK WITH BOTH SCALA AND JAVA.**

**2. BUT SPARK IS ONLY DESIGNED TO WORK WITH SCALA.**

**MAJOR ADVANTAGE OF SCALA IS THE JVM.**

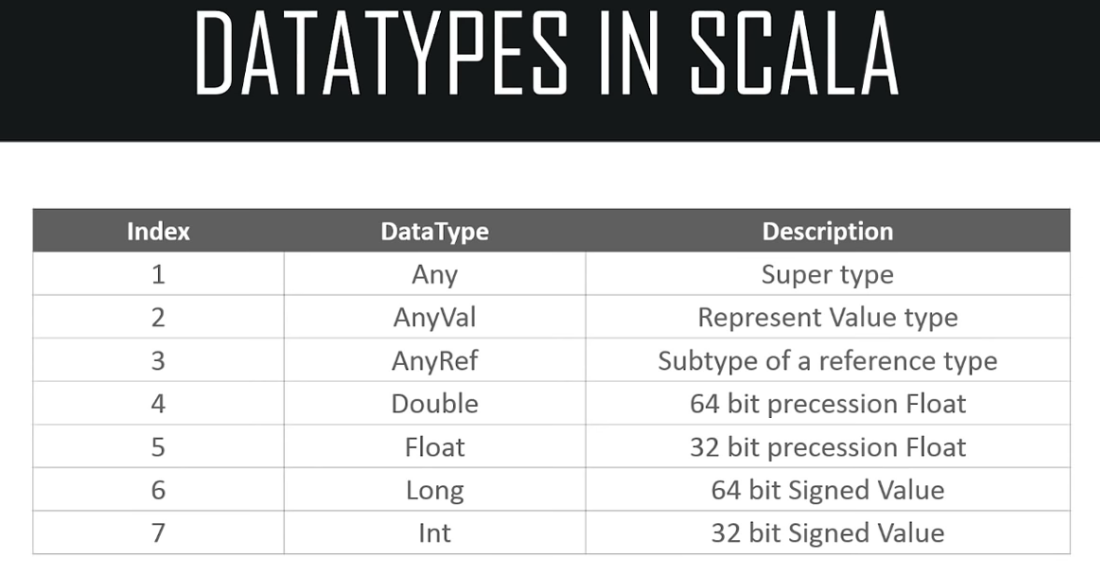
**Scaler code is first compiled by the scaler complier and bytecode is generated which is then transferred to the JAVA VIRTUAL MACHINE to generate the output.**

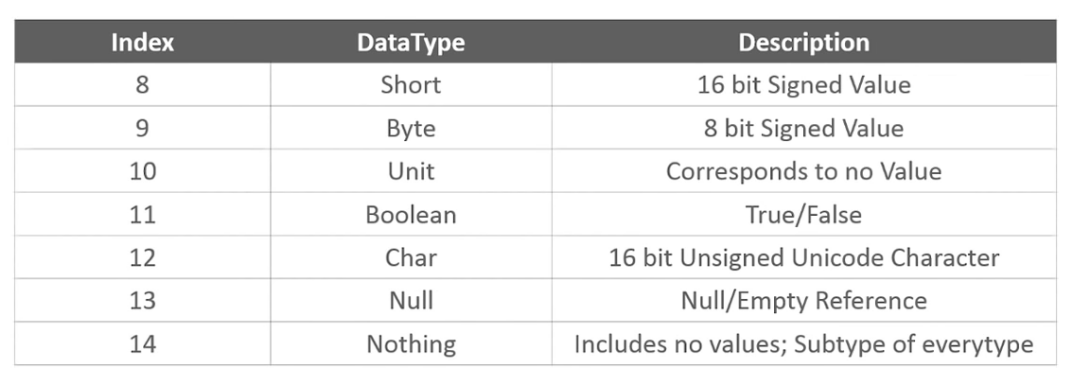
**WHY SCALA**

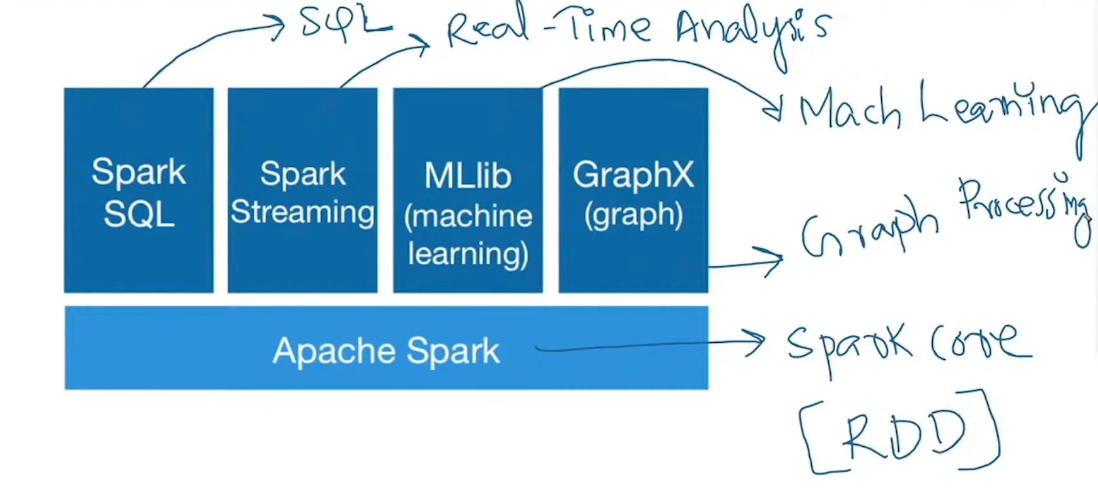
1. Distributed data analysis: SCALA can work on the data which is in distributed data and can support multiple libraries for parallel processing.
2. Immutable language: SCALA uses immutable data and supports high order functions which means, one function can be passed as a parameter in a whole new function.
3. Less down time: SCALA eliminates all the boiler plate codes which saves time and reduces the line of code. It handles APIs in a better way than other conventional languages.
4. Support for type construction: which enables programmers to work with containers with ease.

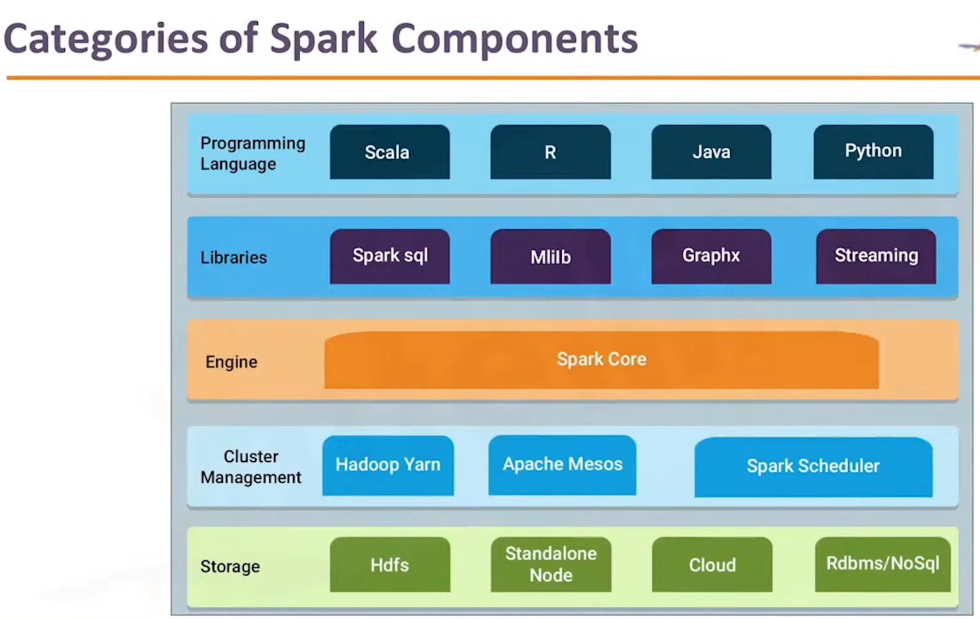
**FEATURES OF SCALA**

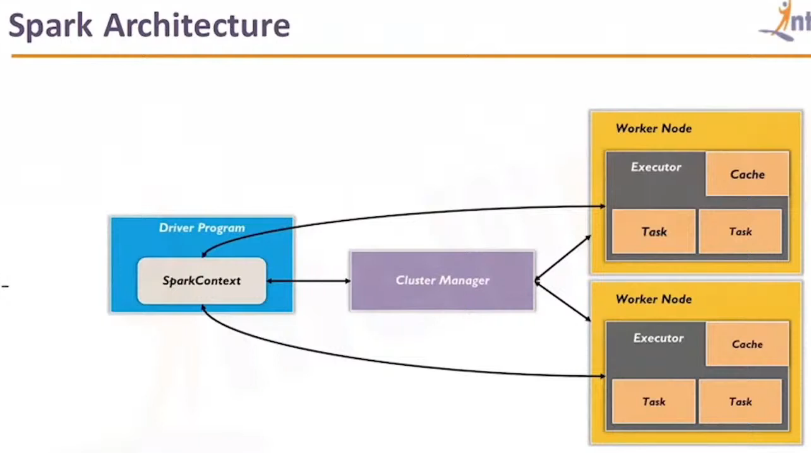
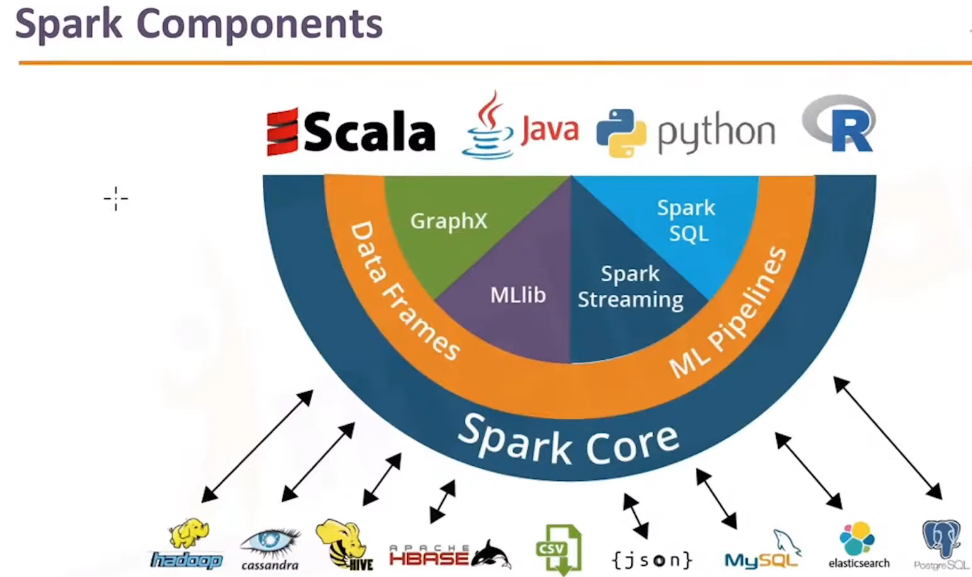
1. OOPS
2. Extensible Programming Language
3. Statically Types Programming Language: it binds the datatype to the variable throughout the scope of code.
4. Functional Programming Language
5. Interoperable



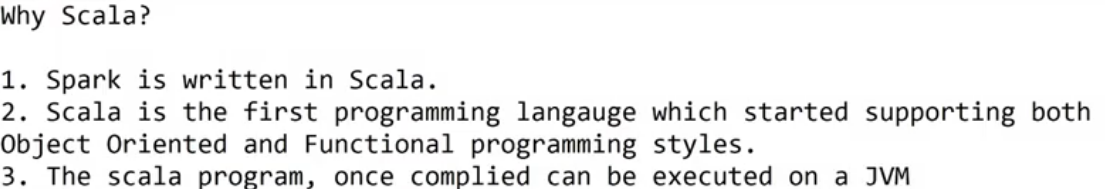








SCALA



SCALA automatically assigns datatype to the input or anything you provide: Dynamic type inference.



There are two types of variables:

1. Immutable variable: the variables which can not be changed. We use keywork ‘VAL’ to declare such variables.
2. Mutable variable: the variables which can be changed. We use keywork ‘VAR’ to declare such variables.

BLOCK EXPRESSION:

Last part of the block/code will be executed.

Lazy variable:

The variable will not be initialized prior, it will initialize when the variable will be called or printed.

IDE (INTEGRATED DEVELOPMENT ENVIRONMENT) – ECLIPSE

2parts of your program

1. Driver: contains the settings which are used to run the program.

TYPES OF FUNCTIONS

Regular functions

Pass by value

Pass by reference

Higher order functions

Anonymous functions

ARRAY (FIXED LENGTH ARRAY)

COLLECTIONS

Df.filter{rowname and condition}.show()

