# Module - 02
# Introduction to Big Data Analytics

COMPONENTS OF HADOOP

ANALYZING BIG DATA WITH HADOOP, DESIGN OF HDFS

DEVELOPING A MAP REDUCE APPLICATION.

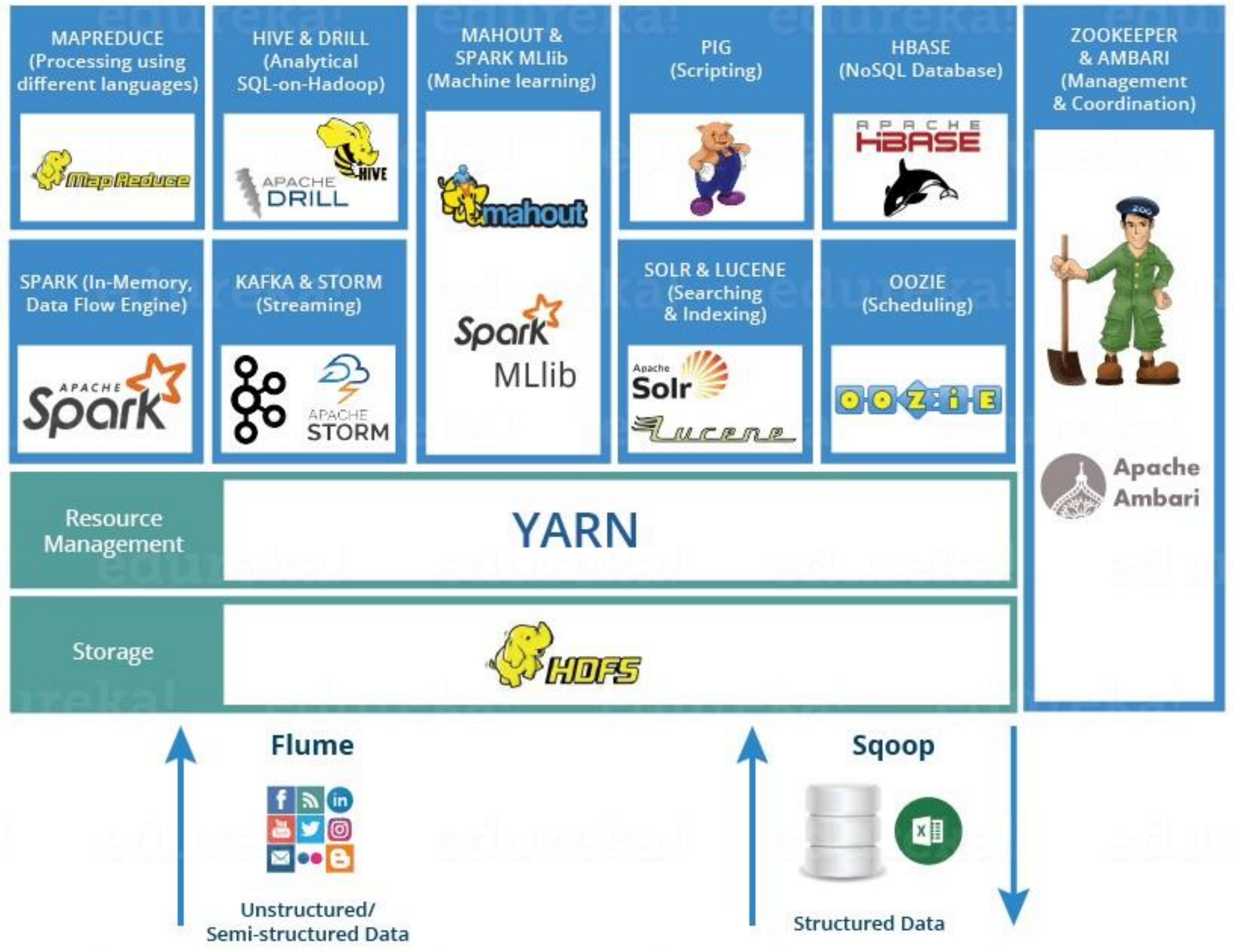
# Why hadoop ..?

- **Problem with Traditional Systems** (Relational Databases and Data Warehouses).
- Traditional systems have been designed to handle only structured data (well-designed rows and columns).
- Relations Databases are vertically scalable which means we need to add more processing, memory, storage to the same system. It is very expensive.
- Most of the data generated today are semi-structured or unstructured and are stored in different silos.
- Google File System (GFS) has overcome many issues of the traditional systems.

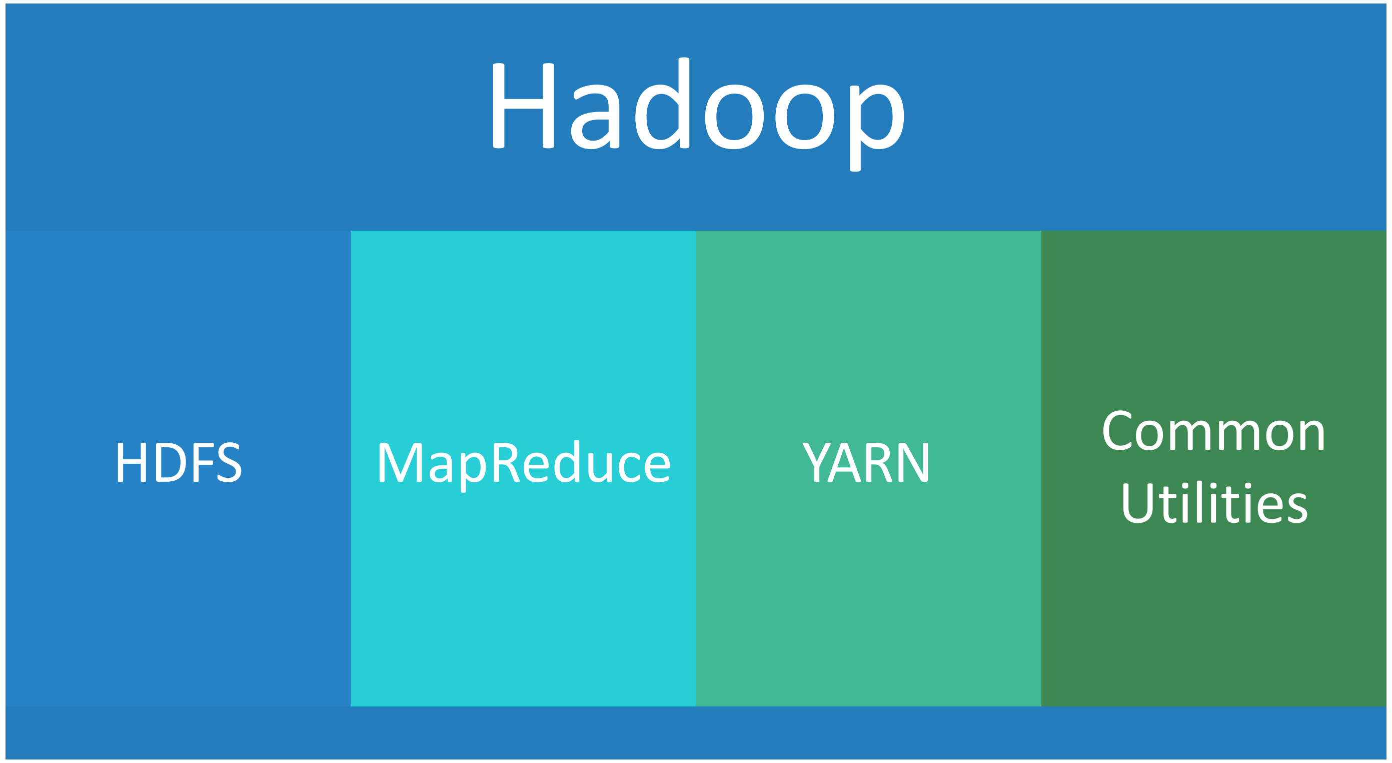
# Introduction

- **Hadoop** is an open source framework from Apache based on **GFS**
- It is used to store process and analyze data which are very huge   in volume.
- It can **handle any type of data**
- Hadoop is **highly scalable** because it handles data in a distributed manner
- Hadoop offers **horizontal scaling**
- It creates and saves replicas of data making it **fault-tolerant**
- Hadoop utilizes the **data locality concept**

# Hadoop Ecosystem


MAPREDUCE (Processing using different languages)
HIVE & DRILL (Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib (Machine learning)
PIG (Scripting)
HBASE (NoSQL Database)
ZOOKEEPER & AMBARI (Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM (Streaming)
SOLR & LUCENE (Searching & Indexing)
OOZIE (Scheduling)
Resource Management
YARN
Storage
HDFS
Apache Ambari
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data

## Hadoop core components



## Hadoop - Installation

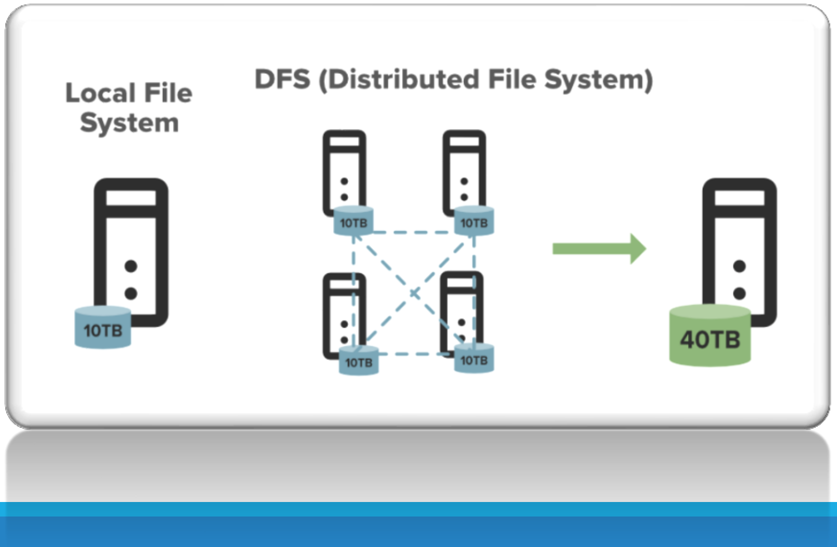
For Hadoop installation from tar ball on the UNIX environment you need

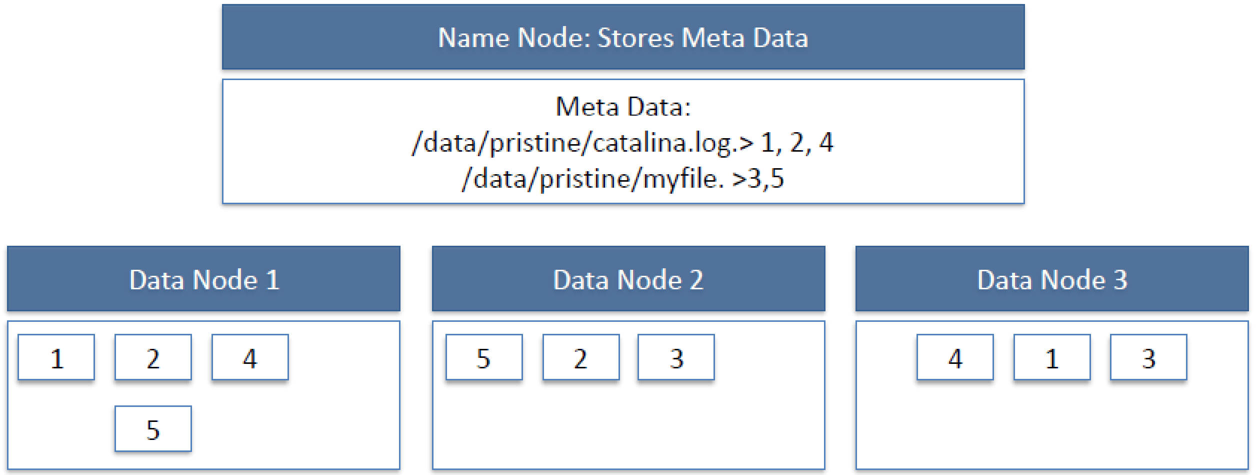
1. Java Installation
2. SSH installation
3. Hadoop Installation and File Configuration

https://www.javatpoint.com/hadoop-installation

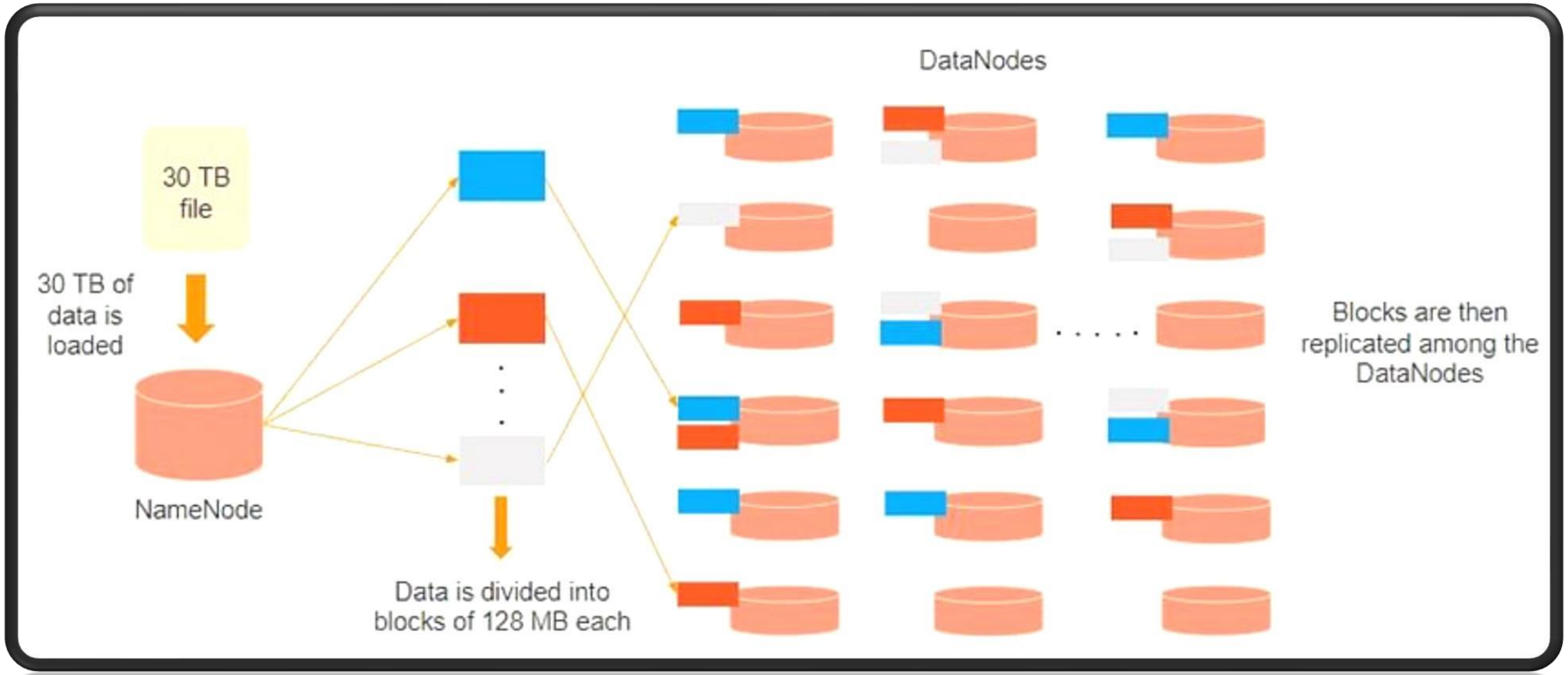
# Hadoop HDFS

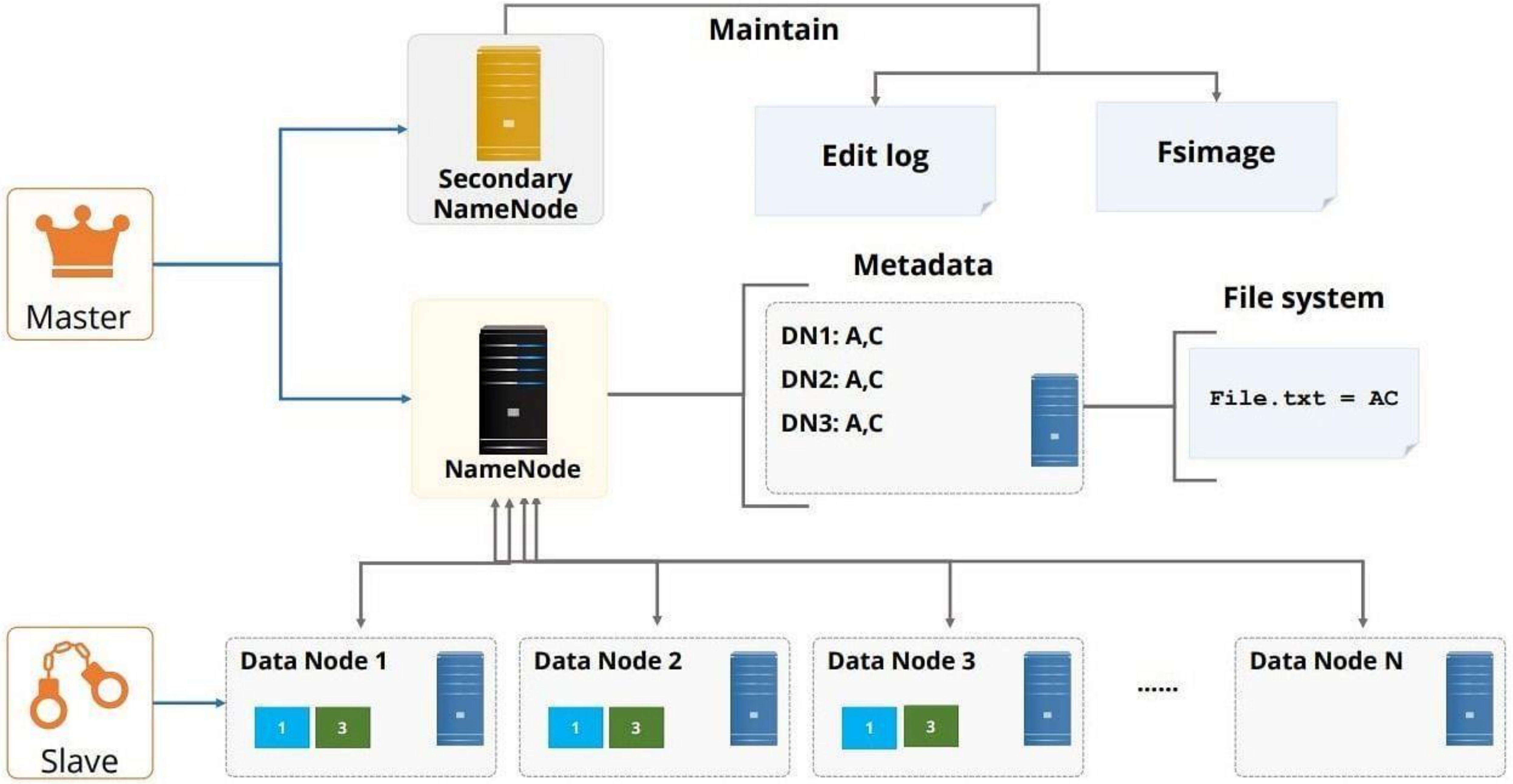
- Hadoop Distributed File System (HDFS) is the storage unit of Hadoop.
- Stores data in the form of files and each file is divided into blocks (size 128 mb).
- Two components of HDFS - name node and data node.




Name Node: Stores Meta Data
Meta Data:
/data/pristine/catalina.log.> 1, 2, 4
/data/pristine/myfile. >3,5
Data Node 1
1
2
4
5
Data Node 2
5
2
3
Data Node 3
4
1
3

# Hadoop HDFS


DataNodes
30 TB file
30 TB of data is loaded
NameNode
Data is divided into blocks of 128 MB each
Blocks are then replicated among the DataNodes


Maintain
Secondary NameNode
Edit log
Fsimage
Master
Metadata
File system
DN1: A,C
DN2: A,C
DN3: A,C
File.txt = AC
NameNode
Slave
Data Node 1
1
3
Data Node 2
1
3
Data Node 3
1
3
......
Data Node N

## Hadoop HDFS components

- Namenode
- Secondary Namenode
- File system
- Metadata
- Datanode

## Hadoop HDFS components

**Namenode**

It is the core component of an HDFS cluster

It maintains and executes the file system namespace operation such as opening, closing, and renaming of files and directories, which are present in HDFS.

**Secondarynode**

It maintains the edit log and namespace image information in sync with the NameNode server

## Hadoop HDFS components

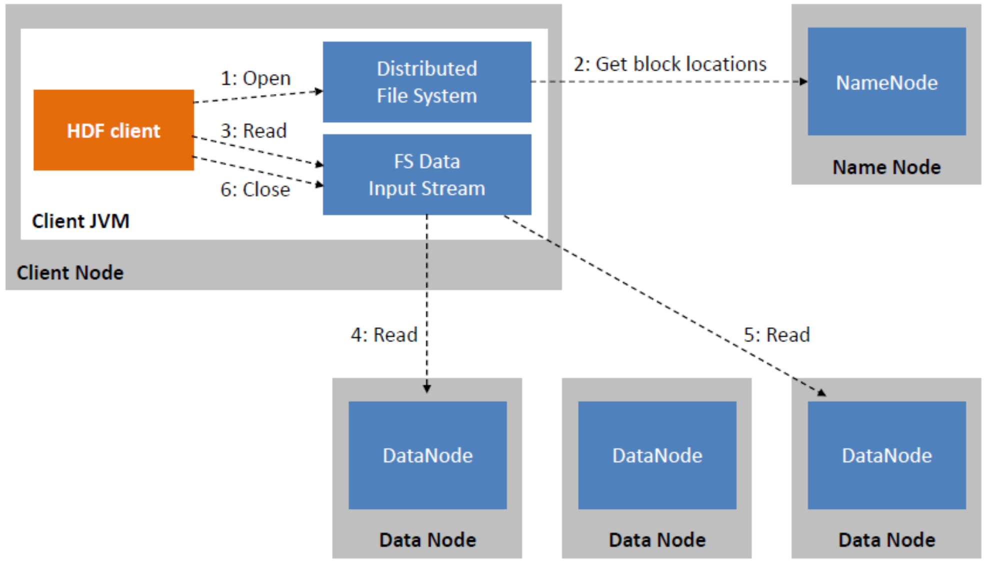
**File System**

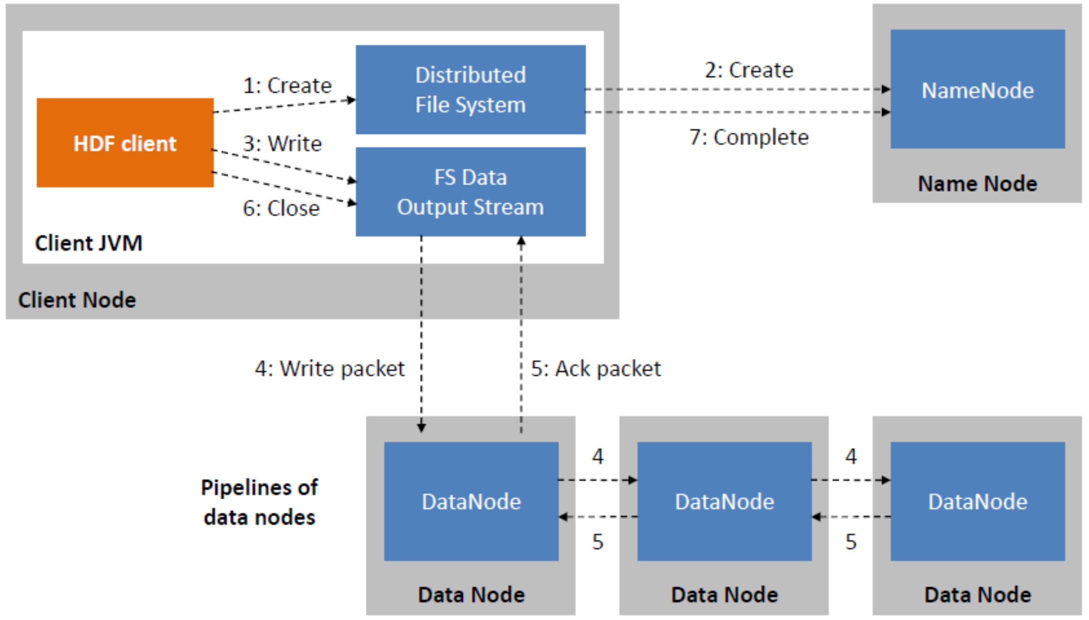
Editlog – used to maintain current transactions

FsImage – used to maintain all transactions from creation of node.

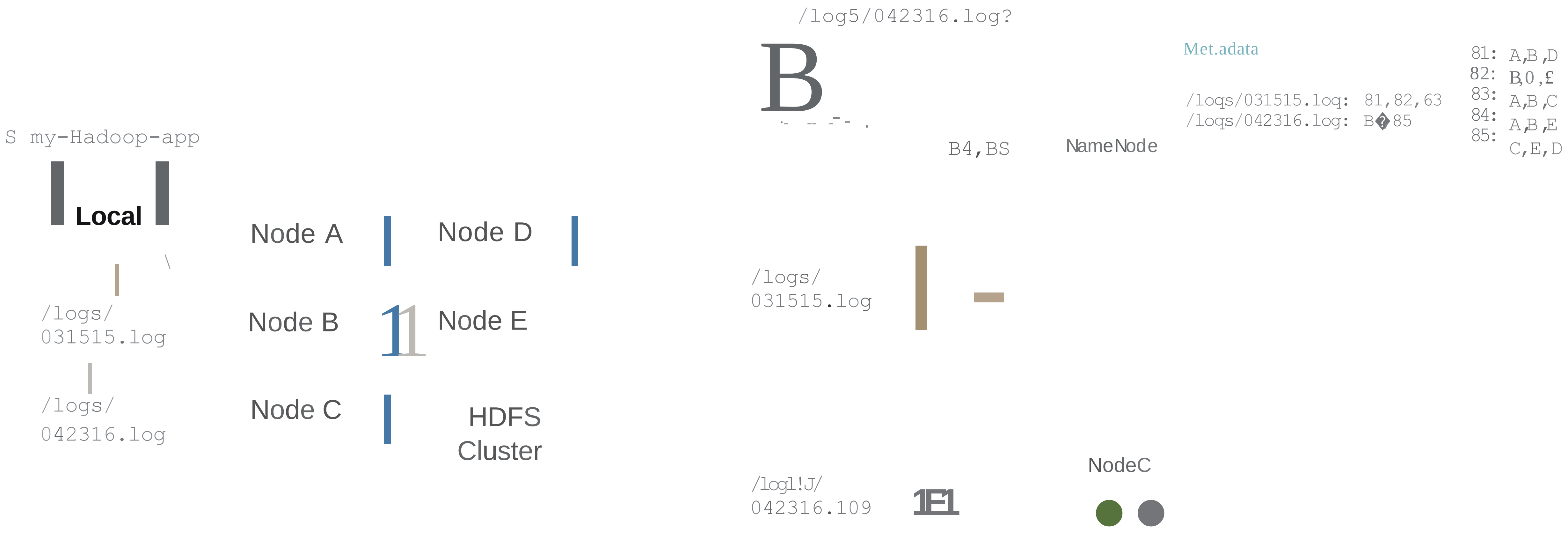
The entire file system namespace including mapping of blocks, files, and file system properties is stored in FsImage. This is also stored in the NameNode local file system.

# Hadoop HDFS (read / write)



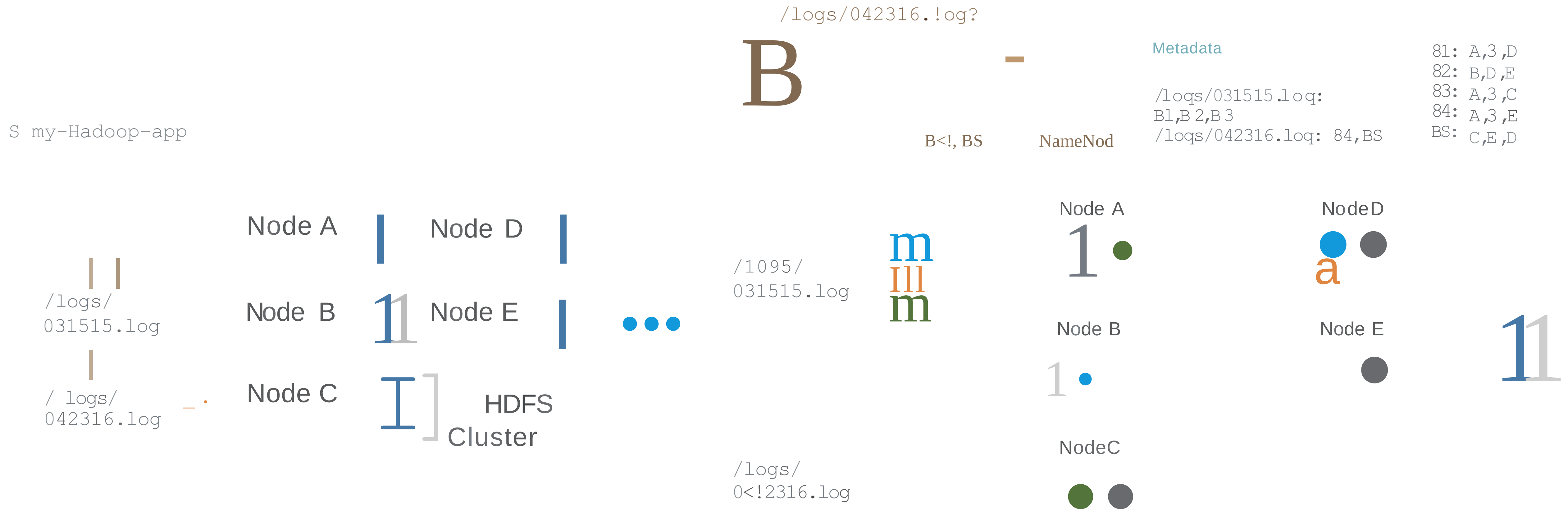


# Hadoop HDFS



/log5/042316.log?
B
Met.adata
81: A,B,D
82: B,0,£
83: A,B,C
84: A,B,E
85: C,E,D
/loqs/031515.loq: 81,82,63
/loqs/042316.log: B�85
S my-Hadoop-app
B4,BS
NameNode
Local
Node A
Node D
Node B
Node E
Node C
HDFS Cluster
/logs/ 031515.log
/logs/ 042316.log
/logs/ 031515.log
/logl!J/ 042316.109
1E1
NodeC

# Hadoop HDFS


/logs/042316.!og?
B
Metadata
/loqs/031515.loq:
Bl,B 2,B 3
/loqs/042316.loq: 84,BS
81: A,3 ,D
82: B,D ,E
83: A,3 ,C
84: A ,3 ,E
BS: C,E ,D
B<!, BS
NameNod
S my-Hadoop-app
/logs/
031515.log
/ logs/
042316.log
Node A
Node D
Node B
Node E
Node C
HDFS
Cluster
/1095/
031515.log
/logs/
0<!2316.log
Node A
NodeD
Node B
Node E
NodeC

## Start of HDFS

The HDFS should be formatted initially and then started in the distributed mode. Commands are given below.

To Format **$ hadoop namenode -format**

To Start **$ start-dfs.sh**

## HDFS Basic File Operations

**Putting data to HDFS from local file system**

- First create a folder in HDFS where data can be put form local file system.

```
$ hadoop fs -mkdir /user/test
```

- Copy the file "data.txt" from a file kept in local folder /usr/home/Desktop to HDFS folder /user/ test

```
$ hadoop fs -copyFromLocal /usr/home/Desktop/data.txt /user/test
```

- Display the content of HDFS folder

```
$ Hadoop fs -ls /user/test
```

## HDFS Basic File Operations

**Copying data from HDFS to local file system**

```
$ hadoop fs -copyToLocal /user/test/data.txt /usr/bin/data_copy.txt
```

**Compare the files and see that both are same**

```
$ md5 /usr/bin/data_copy.txt /usr/home/Desktop/data.txt
```

"<path>" means any file or directory name.

"<path>..." means one or more file or directory names.

"<file>" means any filename.

"<src>" and "<dest>" are path names in a directed operation.

"<localSrc>" and "<localDest>" are paths as above, but on the local file system

# Hadoop HDFS

**Features**

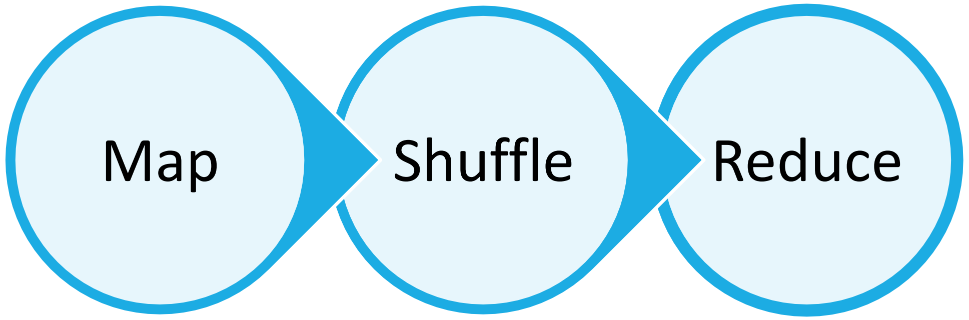
- Data replication
- Fault tolerance and reliability
- High availability
- Scalability
- High throughput
- Data locality

**Advantages**

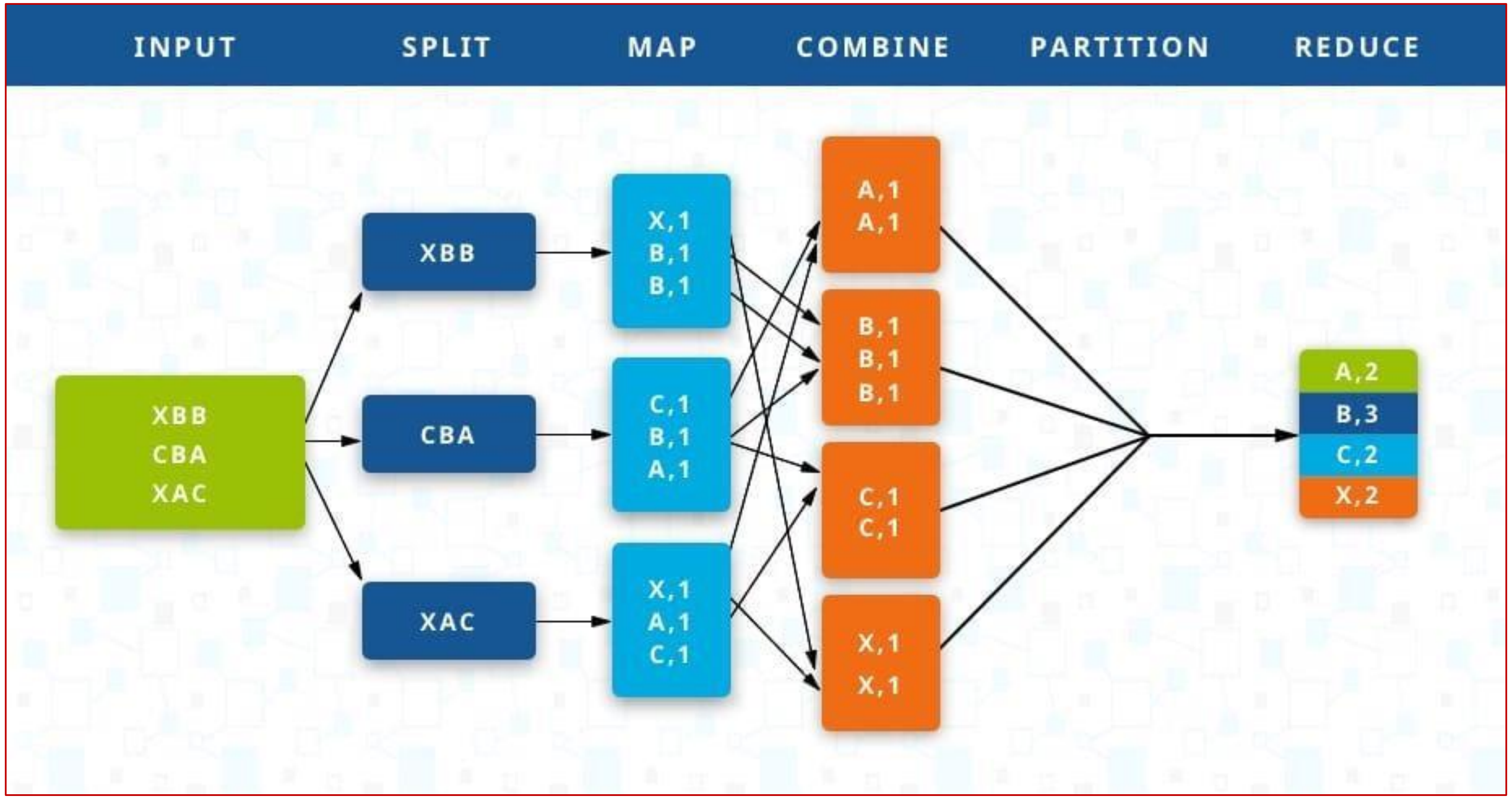
- Cost effectiveness
- Large data set storage.
- Fast recovery from hardware failure
- Portability
- Streaming data access

# Hadoop MapReduce

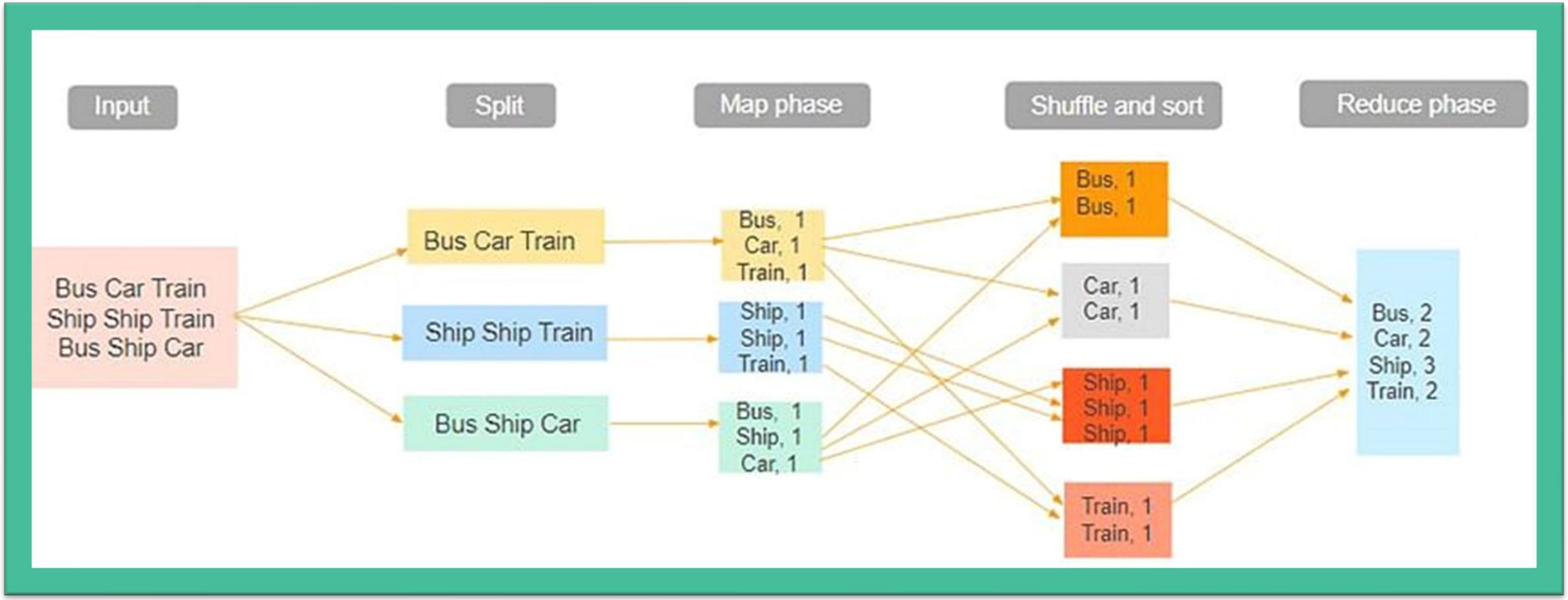
- MapReduce is the processing unit of Hadoop.
- It essentially divides a single task into multiple tasks and processes them on different machines.
- It operates exclusively on **<key, value>** pairs
- **<k1, v1> -> Map() -> list(<k2, v2>)**
  **<k2, list(v2)> -> Reduce() -> list(<k3, v3>)**
- It is easy to scale data processing over multiple computing nodes.
- Under the MapReduce model, the data processing primitives are called **mappers** and **reducers.**



## Hadoop MapReduce



# Hadoop MapReduce



## Hadoop MapReduce

- Input Files
- Input Format
- Input Split
- Record Reader
- Mapper
- Combiner
- Partitioner
- Shuffling & Sorting
- Reducer
- Record Writer
- Output Format

## Hadoop MapReduce

**Mapper Code**

- Write Mapper logic
- How map task will process the data to produce the key-value pair to be aggregated.

**Reducer Code**

- Write reducer logic
- Combines the intermediate key-value pair generated by the mapper.

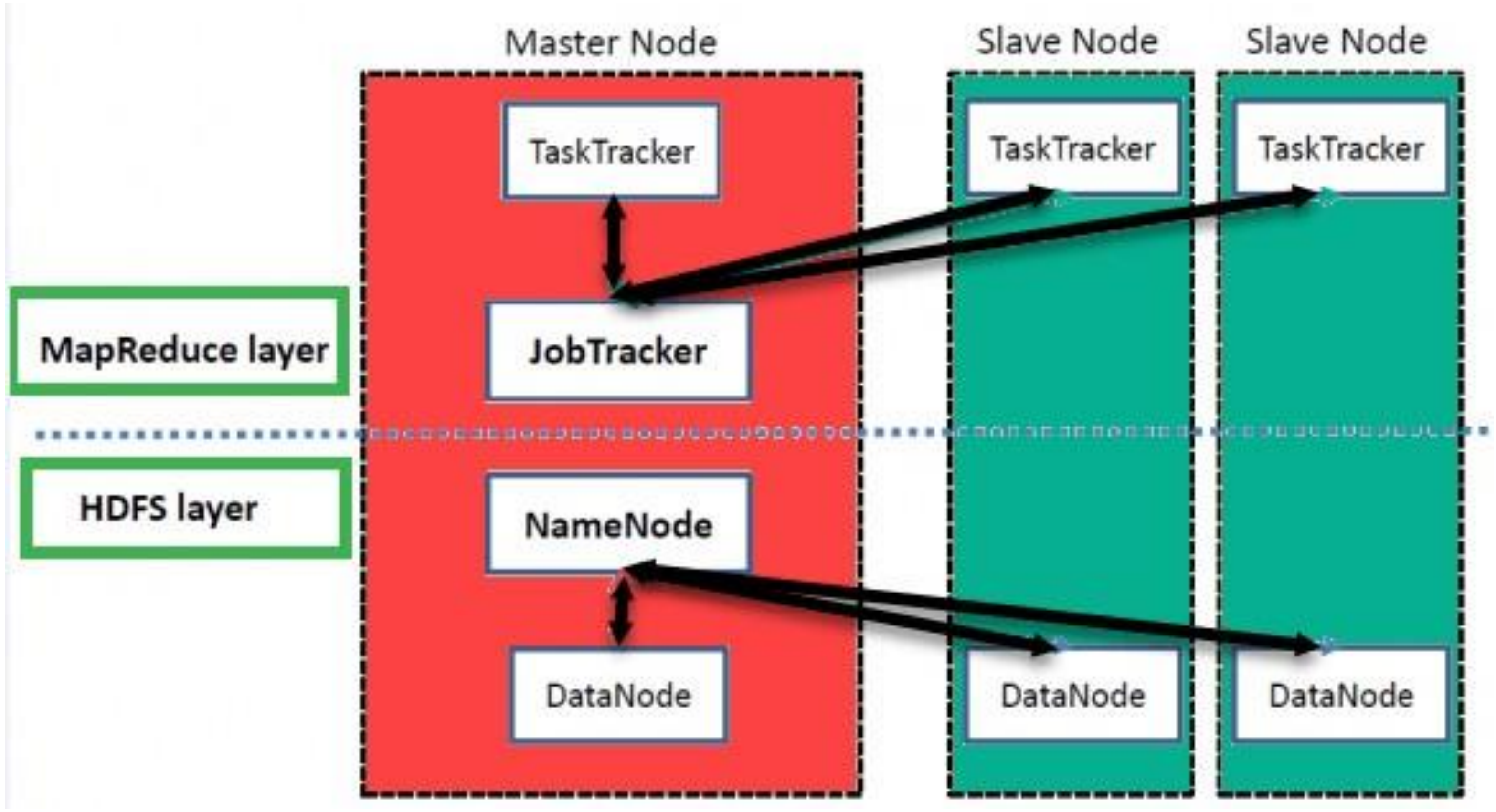
**Driver Code**

- Specify the job configurations (job name, i/o path)

## Hadoop MapReduce

- Example – Counting the word occurrences (frequencies) in a text file (or set of files).
- https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

## Hadoop Architecture



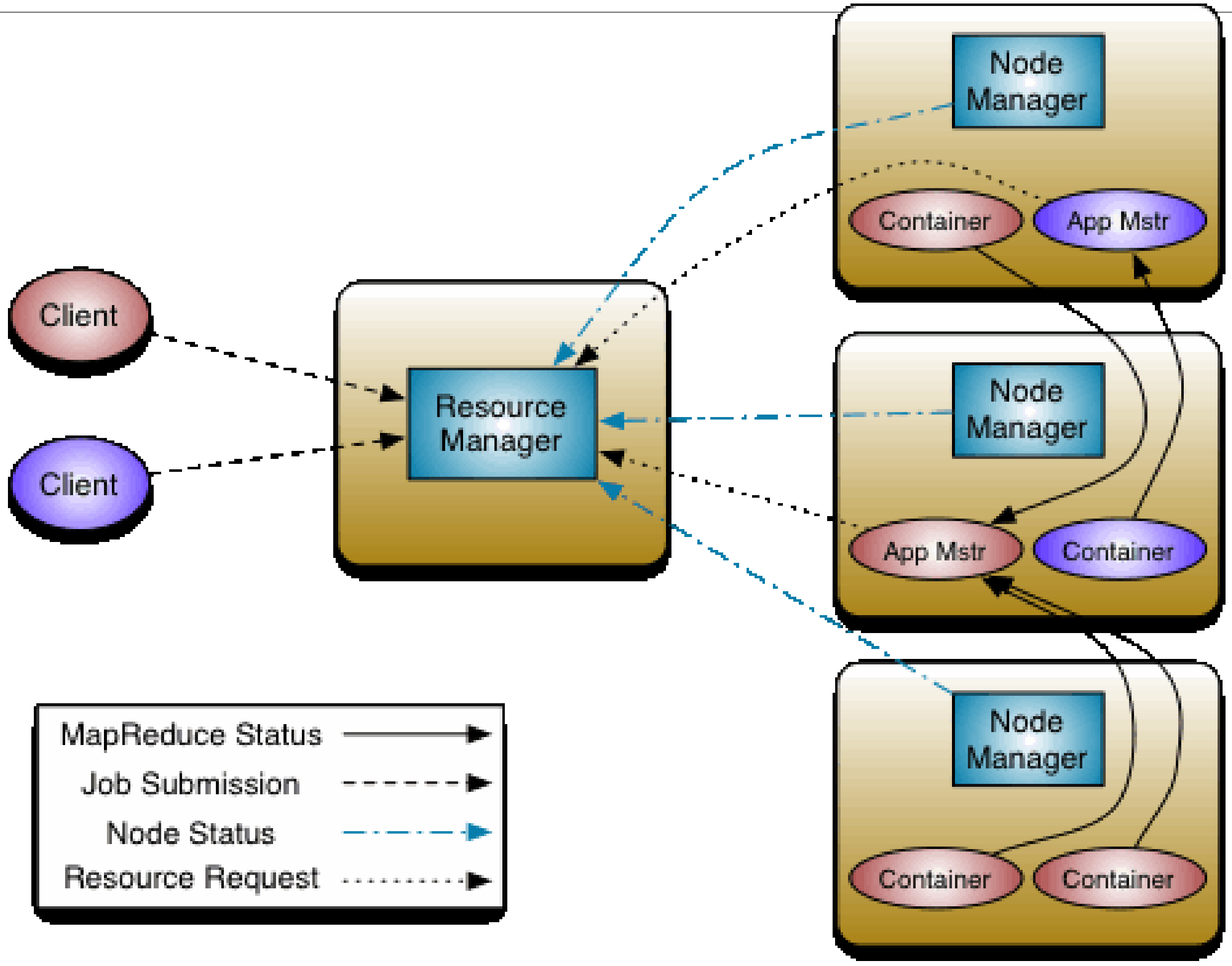
## Hadoop YARN

- Yet Another Resource Negotiator (YARN) is a resource management unit.
- YARN was introduced in Hadoop version 2.0 in the year 2012 by Yahoo and Hortonworks.
- It is a file system that is built on top of HDFS.
- Responsible for allocating system resources to the various applications running in a Hadoop cluster.
- Performs job scheduling.

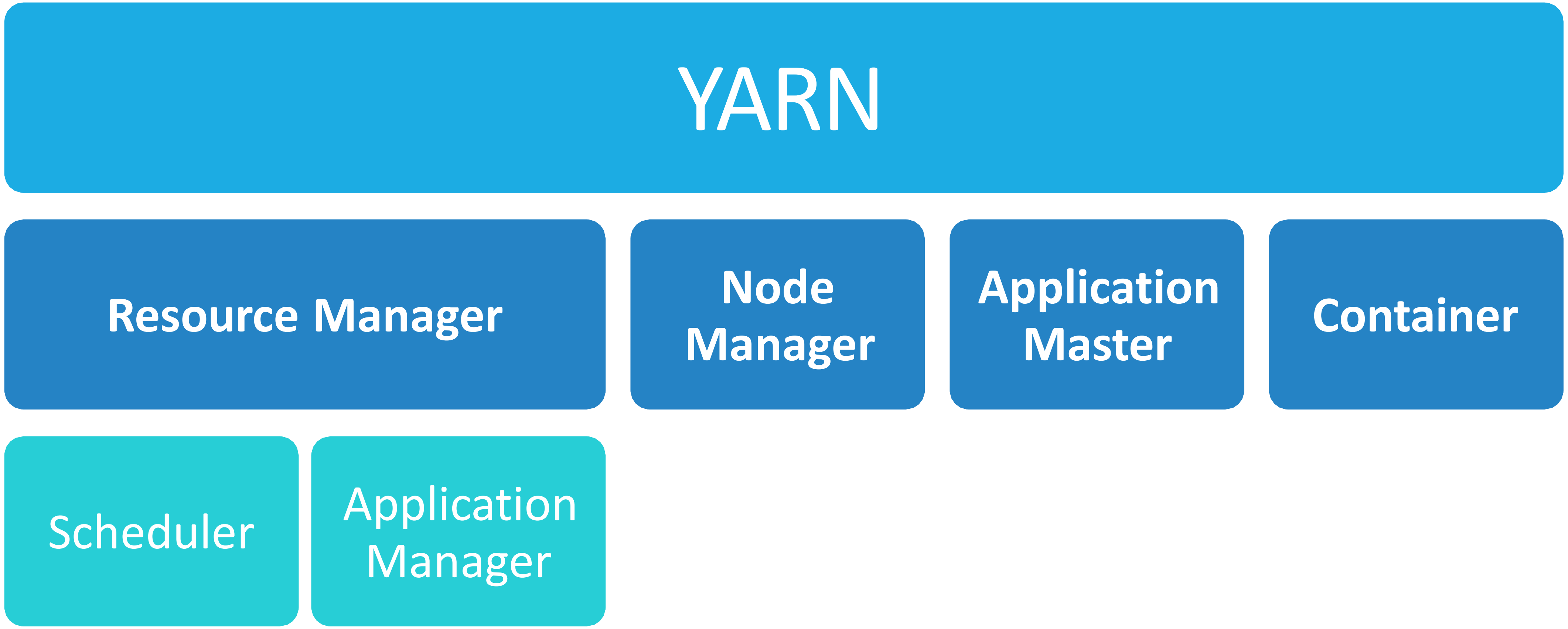
## Hadoop YARN

- YARN allows different data processing methods like graph processing, interactive processing, stream processing as well as batch processing to run and process data stored in HDFS.
- YARN enabled the users to perform operations as per requirement by using a variety of tools like ***Spark*** for real-time processing, ***Hive*** for SQL, ***HBase*** for NoSQL and others.

# Hadoop YARN Architecture



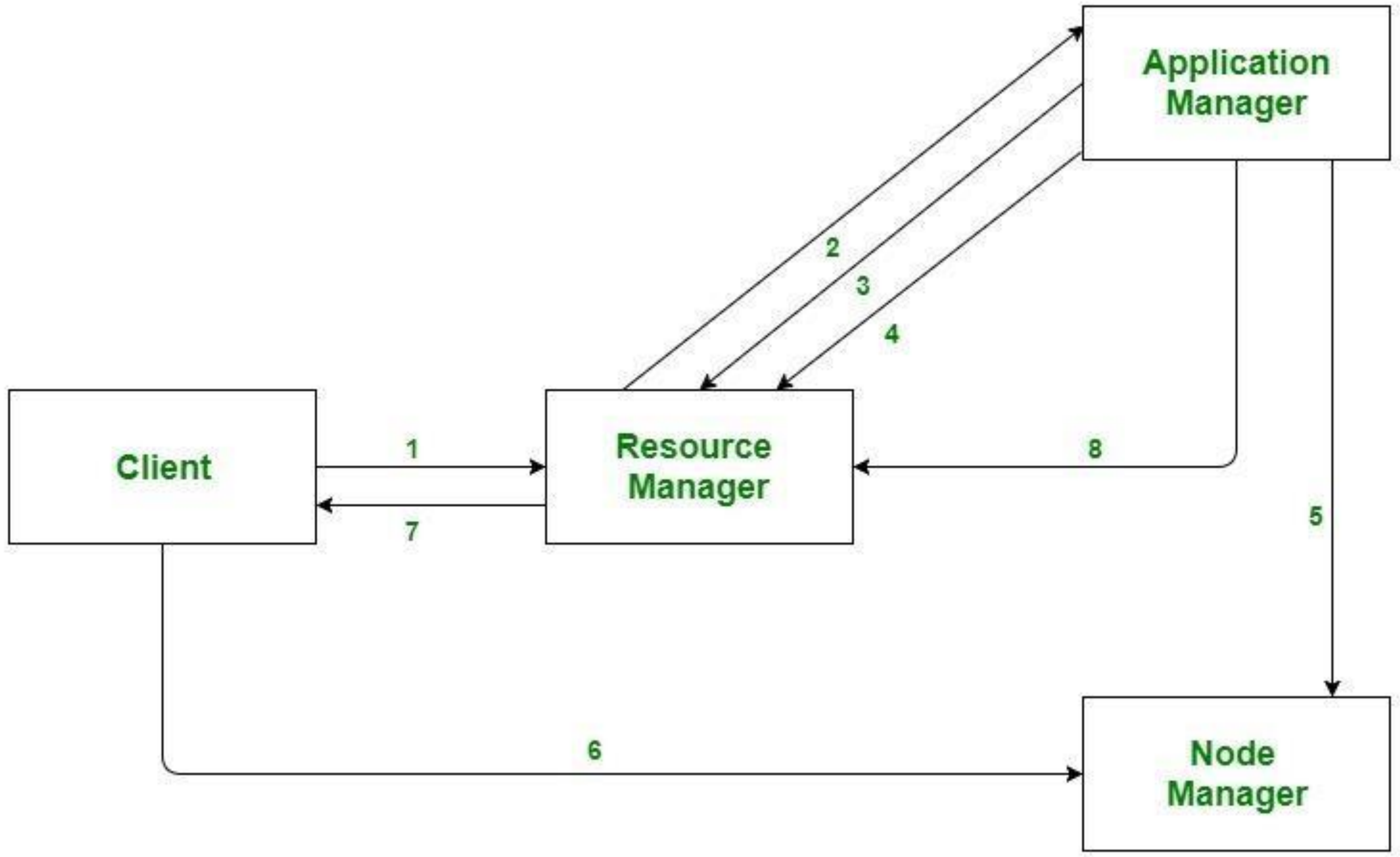
## Hadoop YARN - Components



## Hadoop YARN - Components

- **Resource Manager:** It is the master daemon of YARN and is responsible for resource assignment and management among all the applications.
- **Node Manager:** It take care of individual node on Hadoop cluster and manages application and workflow and that particular node. Its primary job is to keep-up with the Resource Manager.
- **Application Master:** An application is a single job submitted to a framework. The application master is responsible for negotiating resources with the resource manager, tracking the status and monitoring progress of a single application.
- **Container:** It is a collection of physical resources such as RAM, CPU cores and disk on a single node.

## Hadoop YARN - Workflow



## References

https://www.javatpoint.com/what-is-hadoop

https://www.analyticsvidhya.com/blog/2020/10/introduction-hadoop-ecosystem/

https://www.guru99.com/learn-hadoop-in-10-minutes.html

https://www.geeksforgeeks.org/hadoop-ecosystem/

https://www.edureka.co/blog/every-hadoop-component/

https://www.simplilearn.com/tutorials/hadoop-tutorial/what-is-hadoop

https://www.techtarget.com/searchdatamanagement/definition/Hadoop-Distributed-File-System-HDFS

https://www.simplilearn.com/tutorials/hadoop-tutorial/hdfs

https://www.talend.com/resources/what-is-mapreduce/

https://www.projectpro.io/article/learn-to-build-big-data-apps-by-working-on-hadoop-projects/344

https://techvidvan.com/tutorials/how-mapreduce-works/

## References

https://hadoop.apache.org/docs/stable/hadoop-yarn/hadoop-yarn-site/YARN.html

https://data-flair.training/blogs/hadoop-yarn-tutorial/

https://www.edureka.co/blog/hadoop-yarn-tutorial/

https://www.techtarget.com/searchdatamanagement/definition/Apache-Hadoop-YARN-Yet-Another-Resource-Negotiator

https://www.geeksforgeeks.org/hadoop-yarn-architecture/