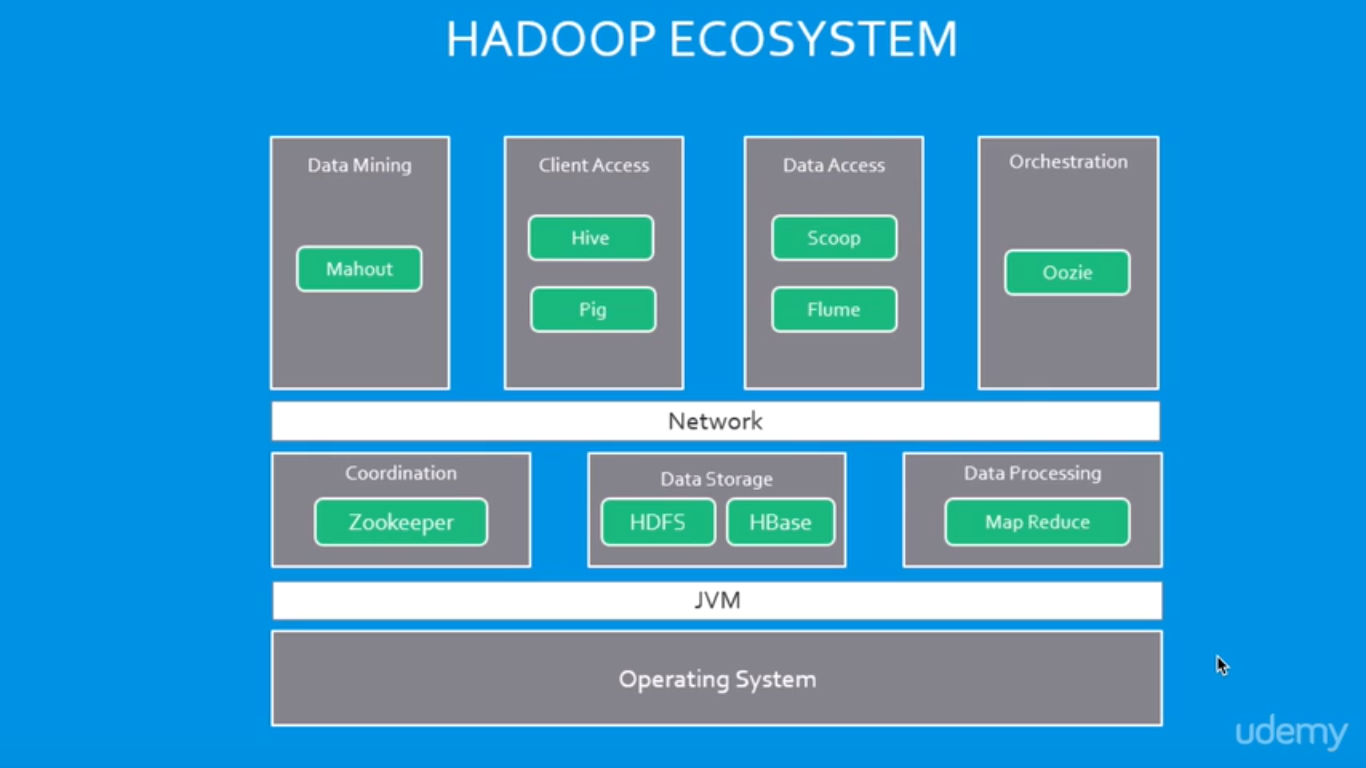
**HADOOP ECOSYSTEM**

The Hadoop ecosystem includes both official Apache open source projects and a wide range of commercial tools and solutions. Some of the best-known open source examples include Spark, Hive, Pig, Oozie and Sqoop. Commercial Hadoop offerings are even more diverse and include platforms and packaged distributions from vendors such as Cloudera, Hortonworks, and MapR, plus a variety of tools for specific Hadoop development, production, and maintenance tasks.

Most of the solutions available in the Hadoop ecosystem are intended to supplement one or two of Hadoop’s four core elements (HDFS, MapReduce, YARN, and Common). However, the commercially available framework solutions provide more comprehensive functionality. The sections below provide a closer look at some of the more prominent components of the Hadoop ecosystem, starting with the Apache projects.



## **Apache open source Hadoop ecosystem elements**

The Apache Hadoop project actively supports multiple projects intended to extend Hadoop’s capabilities and make it easier to use. There are several top-level projects to create development tools as well as for managing Hadoop data flow and processing. Many commercial third-party solutions build on the technologies developed within the Apache Hadoop ecosystem.

Spark, Pig, and Hive are three of the best-known Apache Hadoop projects. Each is used to create applications to process Hadoop data. While there are a lot of articles and discussions about whether Spark, Hive or Pig is better, in practice many organizations do not only use a single one because each is optimized for specific functions.

#### [**Spark**](http://www.bmcsoftware.in/guides/hadoop-apache-spark.html)

Spark is both a programming model and a computing model. It provides a gateway to in-memory computing for Hadoop, which is a big reason for its popularity and wide adoption. Spark provides an alternative to MapReduce that enables workloads to execute in memory, instead of on disk. Spark accesses data from HDFS but bypasses the MapReduce processing framework, and thus eliminates the resource-intensive disk operations that MapReduce requires. By using in-memory computing, Spark workloads typically run between 10 and 100 times faster compared to disk execution.

Spark can be used independently of Hadoop. However, it is used most commonly with Hadoop as an alternative to MapReduce for data processing. Spark can easily coexist with MapReduce and with other ecosystem components that perform other tasks.

Spark is also popular because it supports SQL, which helps overcome a shortcoming in core Hadoop technology. The Spark programming environment works interactively with Scala, Python, and R shells. It has been used for data extract/transform/load (ETL) operations, stream processing, machine learning development and with the Apache GraphX API for graph computation and display. Spark can run on a variety of Hadoop and non-Hadoop clusters, including Amazon S3.

#### **Hive**

Hive is data warehousing software that addresses how data is structured and queried in distributed Hadoop clusters. Hive is also a popular development environment that is used to write queries for data in the Hadoop environment. It provides tools for ETL operations and brings some SQL-like capabilities to the environment. Hive is a declarative language that is used to develop applications for the Hadoop environment, however it does not support real-time queries.

Hive has several components, including:

* HCatalog – Helps data processing tools read and write data on the grid. It supports MapReduce and Pig.
* WebHCat – Lets you use an HTTP/REST interface to run MapReduce, Yarn, Pig, and Hive jobs.
* HiveQL – Hive’s query language intended as a way for SQL developers to easily work in Hadoop. It is similar to SQL and helps both structure and query data in distributed Hadoop clusters.

Hive queries can run from the Hive shell, JDBC, or ODBC. MapReduce (or an alternative) breaks down HiveQL statements for execution across the cluster.

Hive also allows MapReduce-compatible mapping and reduction software to perform more sophisticated functions. However, Hive does not allow row-level updates or support for real-time queries, and it is not intended for OLTP workloads. Many consider Hive to be much more effective for processing structured data than unstructured data, for which Pig is considered advantageous.

#### [**Pig**](http://www.bmcsoftware.in/guides/hadoop-apache-pig.html)

Pig is a procedural language for developing parallel processing applications for large data sets in the Hadoop environment. Pig is an alternative to Java programming for MapReduce, and automatically generates MapReduce functions. Pig includes Pig Latin, which is a scripting language. Pig translates Pig Latin scripts into MapReduce, which can then run on YARN and process data in the HDFS cluster. Pig is popular because it automates some of the complexity in MapReduce development.

Pig is commonly used for complex use cases that require multiple data operations. It is more of a processing language than a query language. Pig helps develop applications that aggregate and sort data and supports multiple inputs and exports. It is highly customizable, because users can write their own functions using their preferred scripting language. Ruby, Python and even Java are all supported. Thus, Pig has been a popular option for developers that are familiar with those languages but not with MapReduce. However, SQL developers may find Hive easier to learn.

#### [**HBase**](http://www.bmcsoftware.in/guides/hadoop-hbase.html)

HBase is a scalable, distributed, NoSQL database that sits atop the HFDS. It was designed to store structured data in tables that could have billions of rows and millions of columns. It has been deployed to power historical searches through large data sets, especially when the desired data is contained within a large amount of unimportant or irrelevant data (also known as sparse data sets). It is also an underlying technology behind several large messaging applications, including Facebook’s.

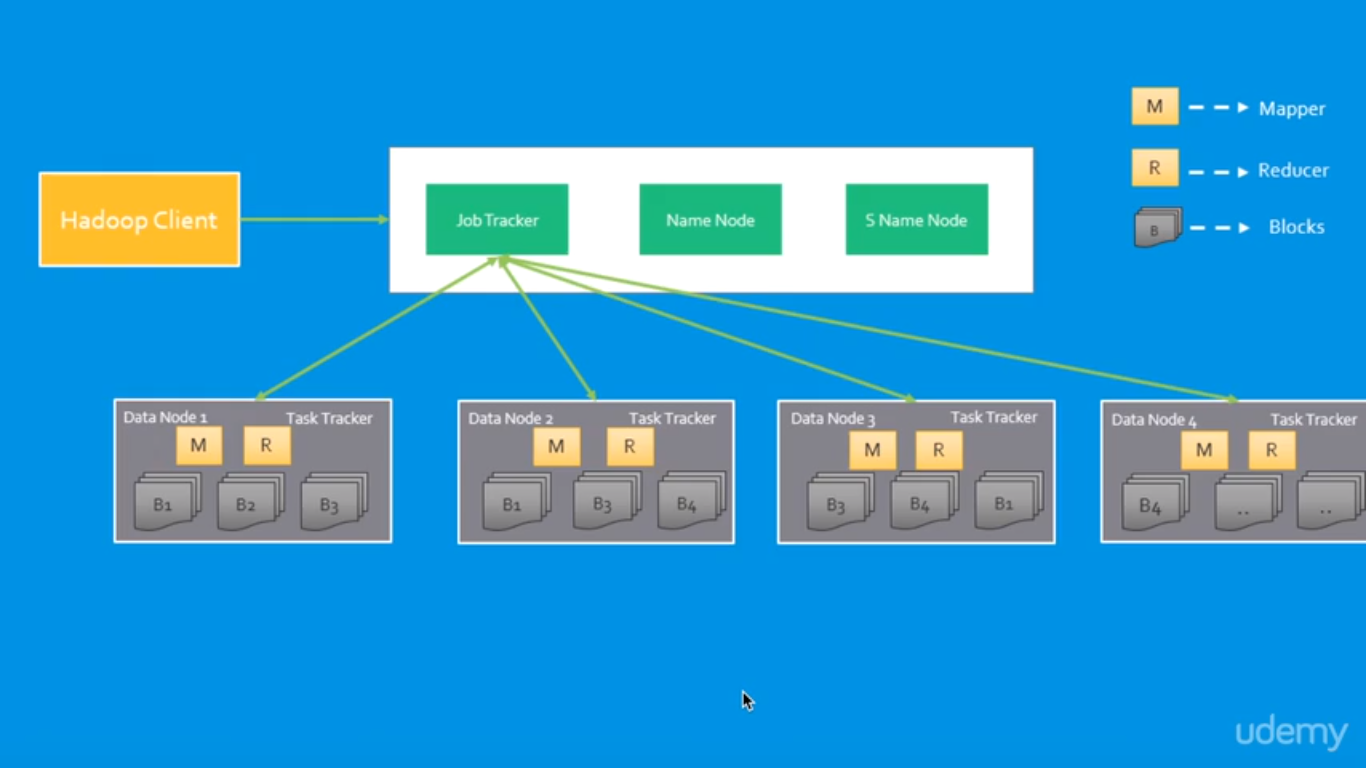
HBase is not a relational database and wasn’t designed to support transactional and other real-time applications. It is accessible through a Java API and has ODBC and JDBC drivers. HBase does not support SQL queries, however there are several SQL support tools available from the Apache project and from software vendors. For example, Hive can be used to run SQL-like queries in HBase.

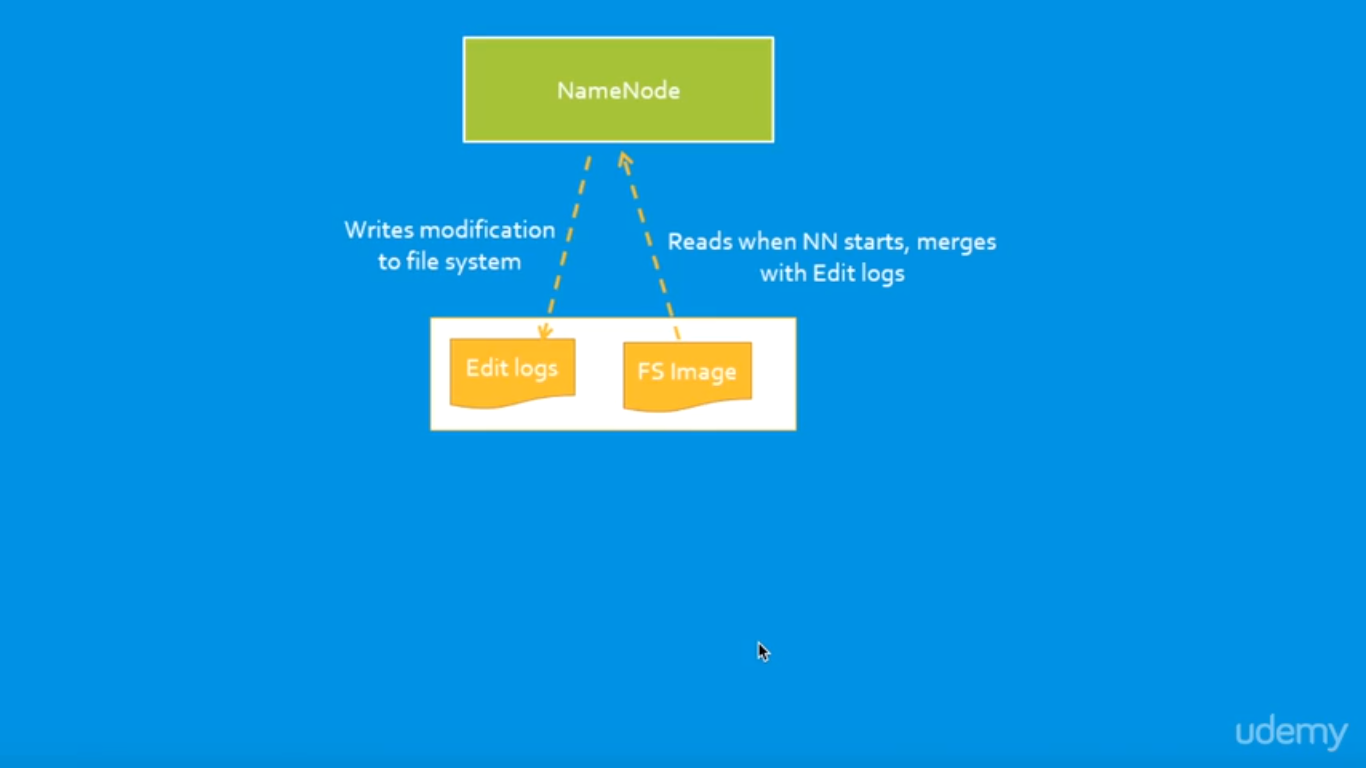
#### **Oozie**

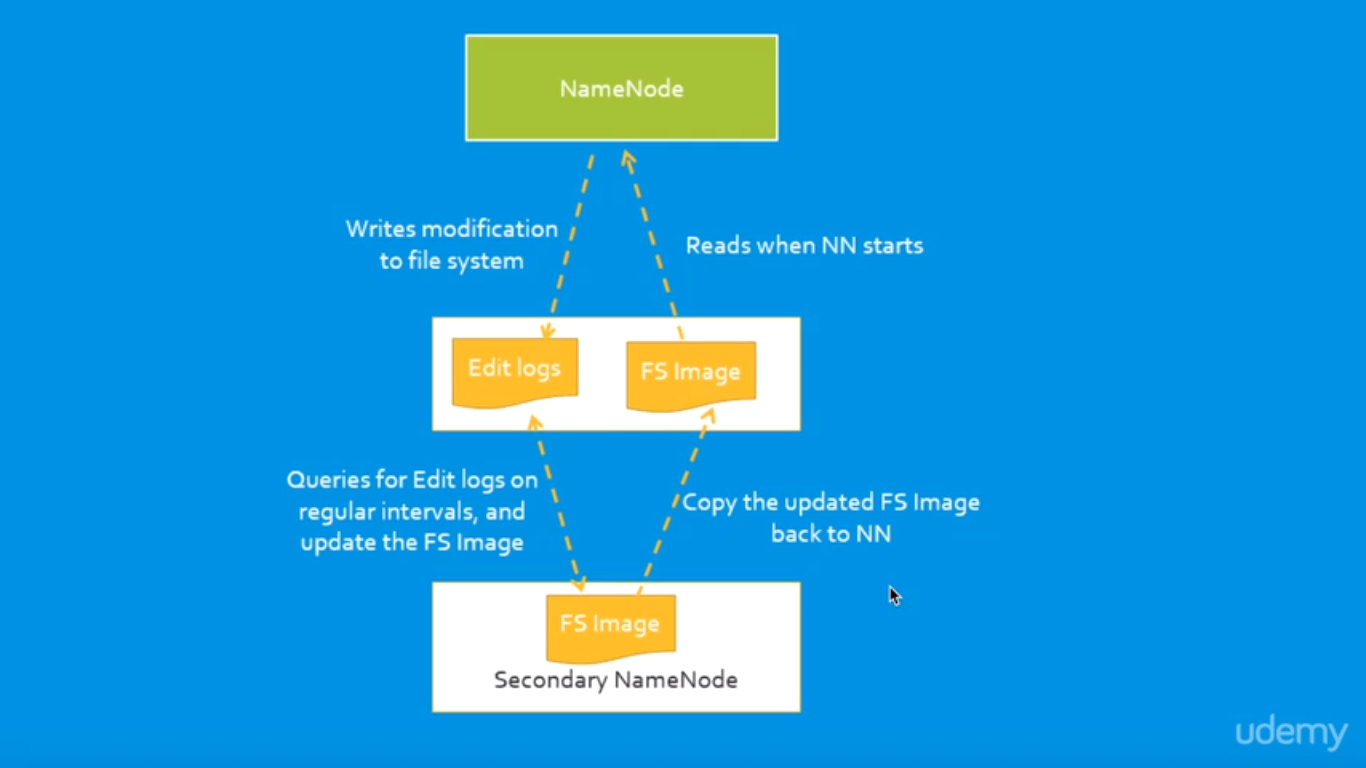
Oozie is the workflow scheduler that was developed as part of the Apache Hadoop project. It manages how workflows start and execute, and also controls the execution path. Oozie is a server-based Java web application that uses workflow definitions written in hPDL, which is an XML Process Definition Language similar to [JBOSS JBPM](http://www.jboss.org/jbossjbpm/) jPDL. Oozie only supports specific workflow types, so other workload schedulers are commonly used instead of or in addition to Oozie in Hadoop environments.

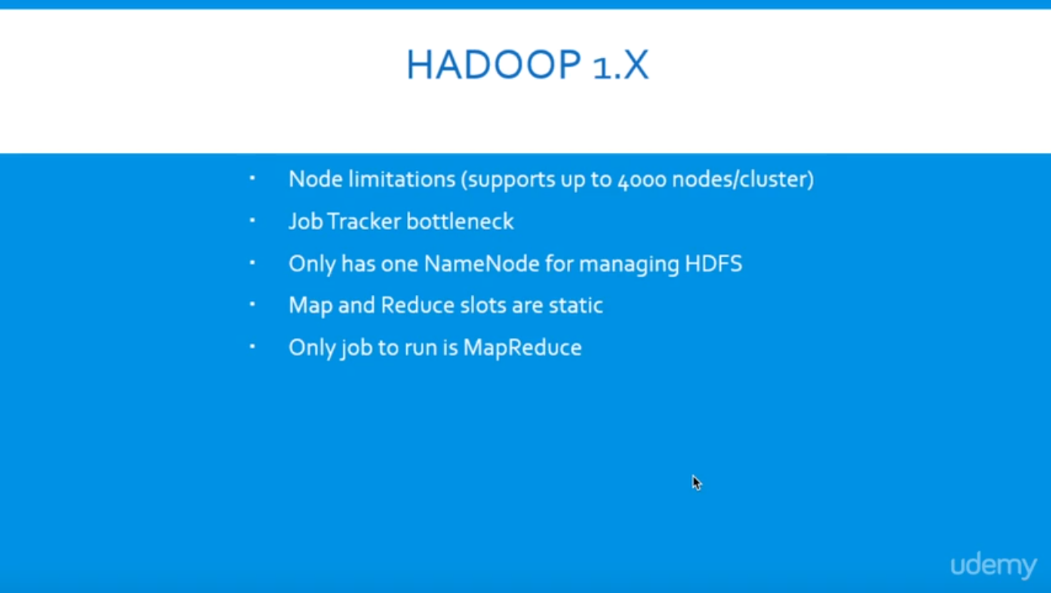
#### **Sqoop**

Think of Sqoop as a front-end loader for big data. Sqoop is a command-line interface that facilitates moving bulk data from Hadoop into relational databases and other structured data stores. Using Sqoop replaces the need to develop scripts to export and import data. One common use case is to move data from an enterprise data warehouse to a Hadoop cluster for ETL processing. Performing ETL on the commodity Hadoop cluster is resource efficient, while Sqoop provides a practical transfer method.



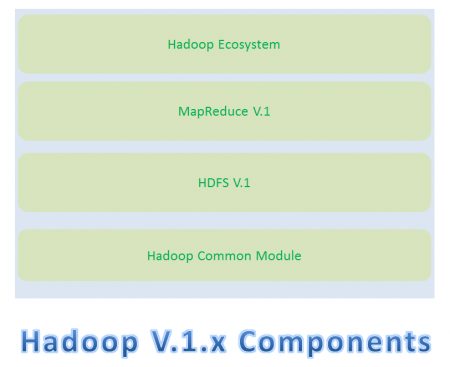




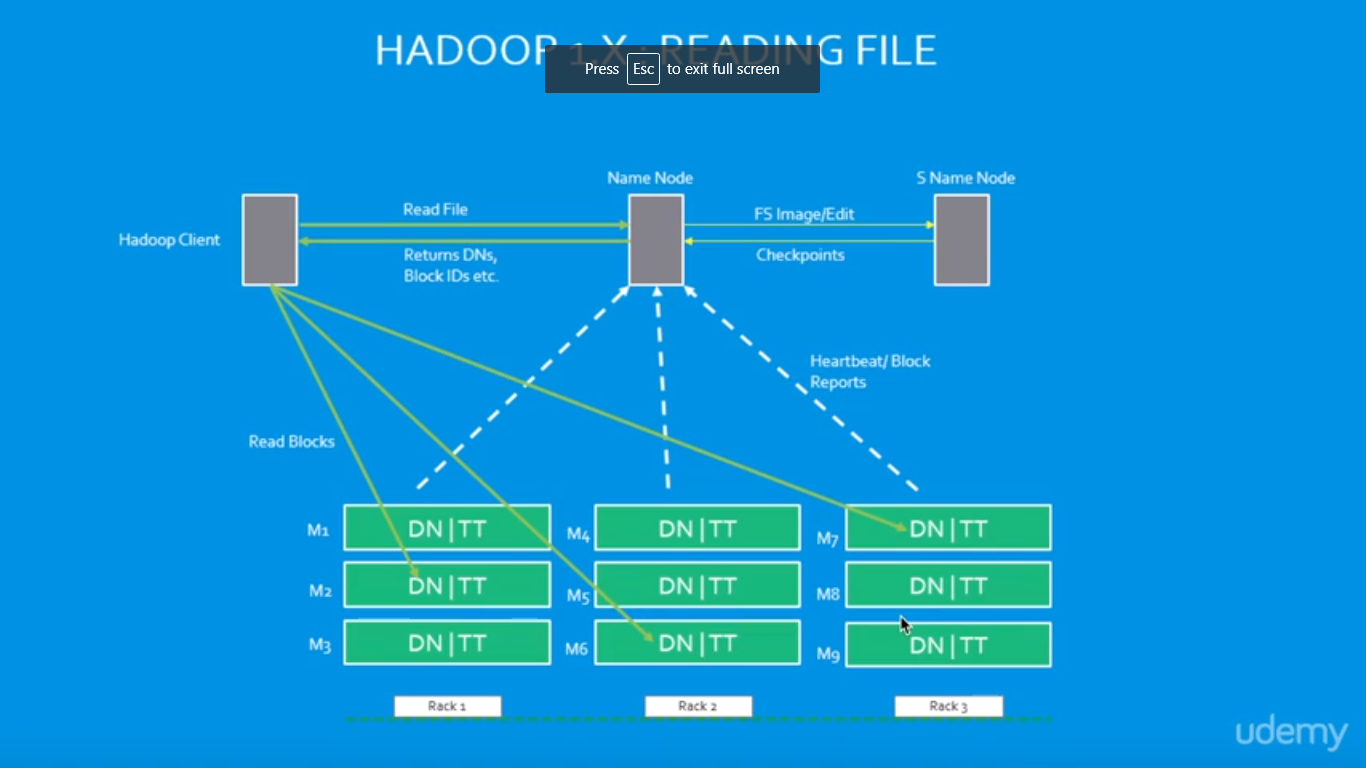
**HADOOP 1.X**

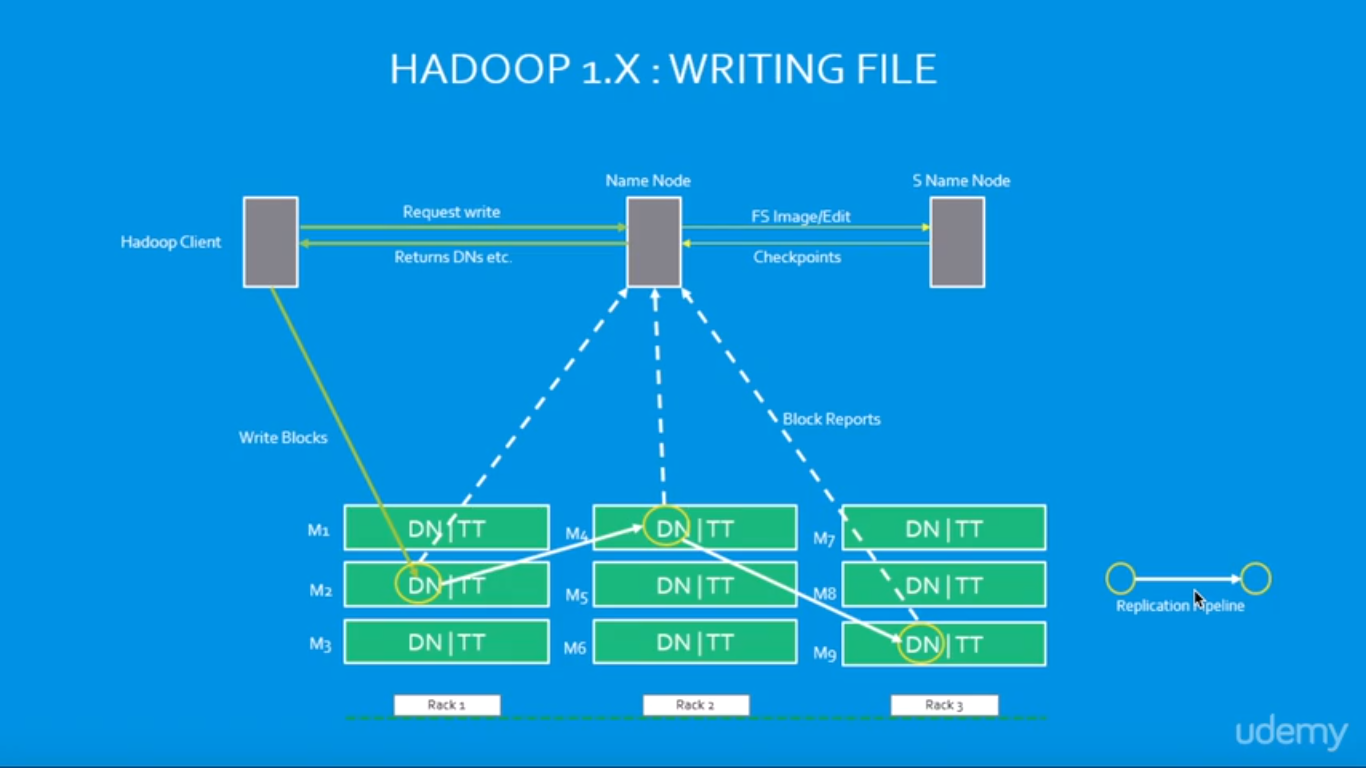
### **Hadoop 1.x Architecture**

Apache Hadoop 1.x or earlier versions are using the following Hadoop Architecture. It is a Hadoop 1.x High-level Architecture. We will discuss in-detailed Low-level Architecture in coming sections.

[](https://cdn.journaldev.com/wp-content/uploads/2015/08/hadoop1.x-components.png)If you don’t understand this Architecture at this stage, no need to worry. Read next sections in this post and also coming posts to understand it very well.

* Hadoop Common Module is a Hadoop Base API (A Jar file) for all Hadoop Components. All other components works on top of this module.
* HDFS stands for Hadoop Distributed File System. It is also know as HDFS V1 as it is part of Hadoop 1.x. It is used as a Distributed Storage System in Hadoop Architecture.
* MapReduce is a Batch Processing or Distributed Data Processing Module. It is built by following Google’s MapReduce Algorithm. It is also know as “MR V1” or “Classic MapReduce” as it is part of Hadoop 1.x.
* Remaining all Hadoop Ecosystem components work on top of these two major components: HDFS and MapReduce. We will discuss all Hadoop Ecosystem components in-detail in my coming posts.





### **Hadoop 1.x Major Components**

* Hadoop 1.x Major Components components are: HDFS and MapReduce. They are also know as “Two Pillars” of Hadoop 1.x.

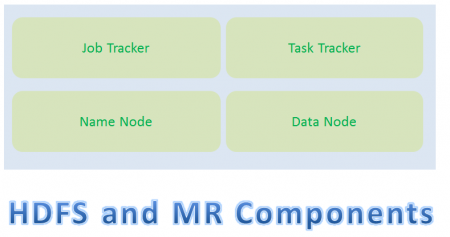
**HDFS:**  
HDFS is a Hadoop Distributed FileSystem, where our BigData is stored using Commodity Hardware. It is designed to work with Large DataSets with default block size is 64MB (We can change it as per our Project requirements).

* HDFS component is again divided into two sub-components:
* Name Node
* Name Node is placed in Master Node. It used to store Meta Data about Data Nodes like “How many blocks are stored in Data Nodes, Which Data Nodes have data, Slave Node Details, Data Nodes locations, timestamps etc” .
* Data Node
* Data Nodes are places in Slave Nodes. It is used to store our Application Actual Data. It stores data in Data Slots of size 64MB by default.

**MapReduce:**  
MapReduce is a Distributed Data Processing or Batch Processing Programming Model. Like HDFS, MapReduce component also uses

Commodity Hardware to process “High Volume of Variety of Data at High Velocity Rate” in a reliable and fault-tolerant manner.

* MapReduce component is again divided into two sub-components:
* Job Tracker
* Job Tracker is used to assign MapReduce Tasks to Task Trackers in the Cluster of Nodes. Sometimes, it reassigns same tasks to other Task Trackers as previous Task Trackers are failed or shutdown scenarios.
* Job Tracker maintains all the Task Trackers status like Up/running, Failed, Recovered etc.
* Task Tracker
* Task Tracker executes the Tasks which are assigned by Job Tracker and sends the status of those tasks to Job Tracker.

[](https://cdn.journaldev.com/wp-content/uploads/2015/08/hadoop1.x-hdfs-mr-components.png)

* We will discuss these four sub-component’s responsibilities and how they interact each other to perform a “Client Application Tasks” in detail in next section.
* How Hadoop 1.x Major Components Works
* Hadoop 1.x components follow this architecture to interact each other and to work parallel in a reliable and fault-tolerant manner.