# Hadoop

**What is Hadoop?**

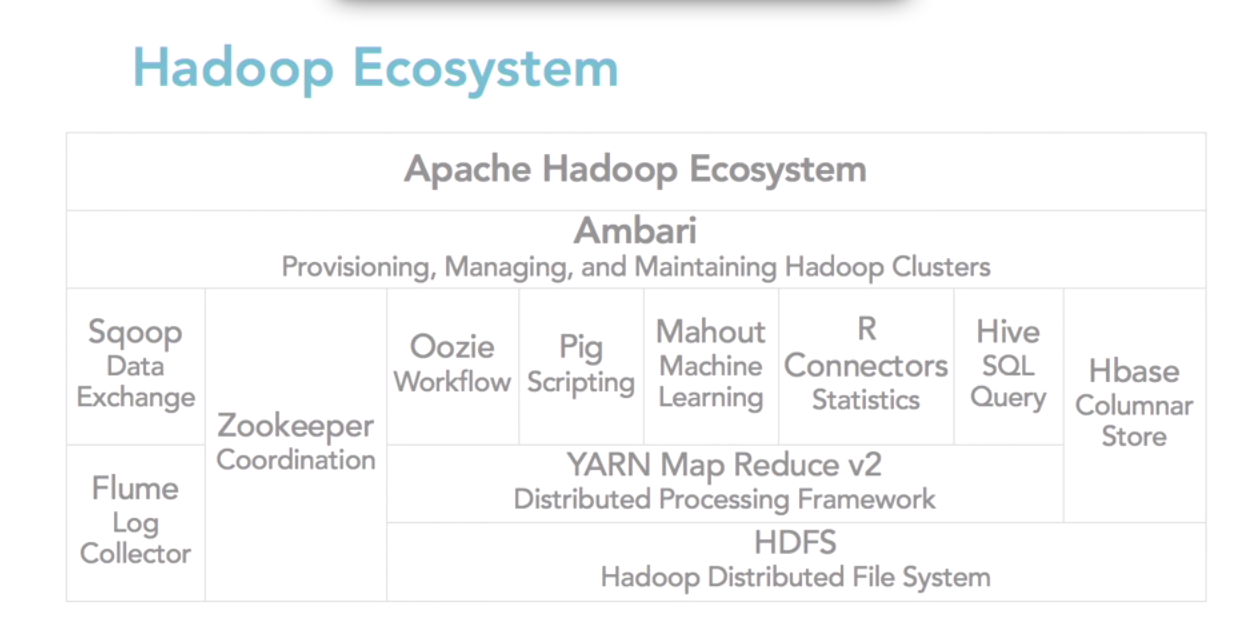
The Hadoop ecosystem is designed to solve a different set of data problems than those of relational databases.

It consists of two components, and oftentimes deployed with other projects as well.

These two components are

1. Open-Source data storage or **HDFS** (Hadoop File System).
2. Processing API that is called **MapReduce**.

Other projects or libraries that commonly used along with Aboe two are Hbase, Hive and Pig.



**What is HDFS?**

The Hadoop Distributed File System (HDFS) is the primary storage system used by Hadoop applications.

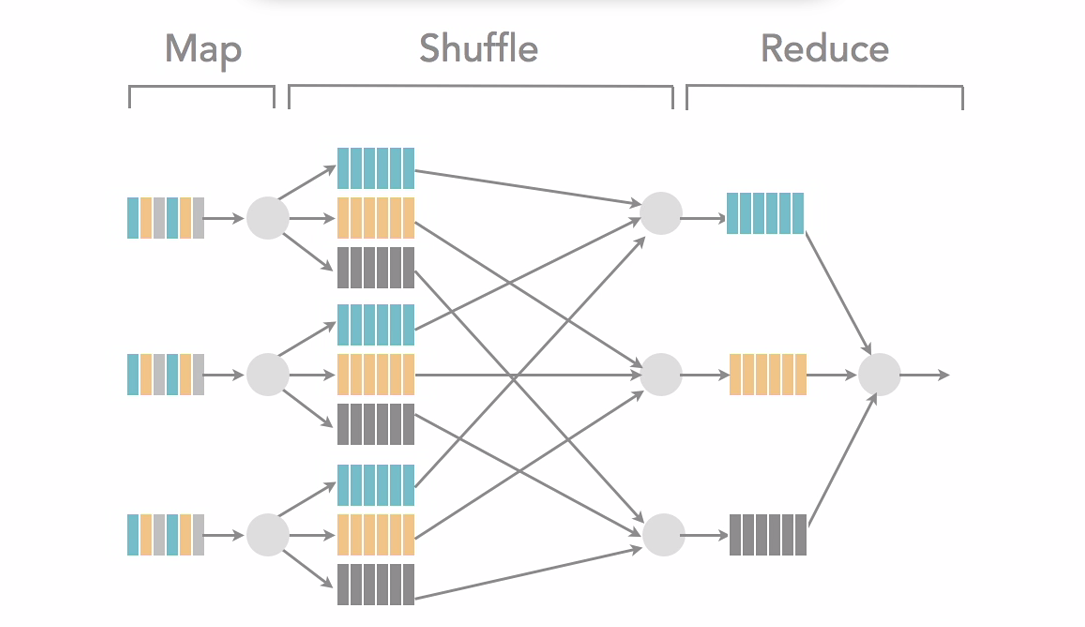
HDFS is a distributed file system that provides high-performance access to data across Hadoop clusters. Like other Hadoop-related technologies, HDFS has become a key tool for managing pools of big data and supporting big data analytics applications.

Because HDFS typically is deployed on low-cost commodity hardware, server failures are common. The file system is designed to be highly fault-tolerant, however, by facilitating the rapid transfer of data between compute nodes and enabling Hadoop systems to continue running if a node fails. That decreases the risk of catastrophic failure, even in the event that numerous nodes fail.

When HDFS takes in data, it breaks the information down into separate pieces and distributes them to different nodes in a cluster, allowing for **parallel processing**. The file system also copies each piece of data multiple times and distributes the copies to individual nodes, placing at least one copy on a different server rack than the others. As a result, the data on nodes that crash can be found elsewhere within a cluster, which allows processing to continue while the failure is resolved.

**What is MapReduce?**

The MapReduce algorithm contains two important tasks, namely **Map** and **Reduce**. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, Reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.



**Database Choices:**

Databases

* File System Database
  + HDFS (Hadoop Distributed File System)
* Databases
  + NoSQL Databases

(KeyValue Pair, ColumnStore, DocumentType etc)

* + RDBMS (Oracle, MySQL)

**Different types of NoSQL Databases**

1. **Document-Oriented Databases**: CouchDB, Couchbase, Mongo, Riak
2. **In-Memory Databases**: Memcached, Redis, Riak, VoltDB, Hazelcast?
3. **Graph Database**s: InfiniteGraph, Neo4J, OrientDB
4. **Column Store Databases**: Apache Hbase, Cassandra, and Google’s BigTable

**Hadoop Vs HBase**

**Hadoop** is based on technology that was developed originally at Google to index the entire Internet, which is called the GFS or the Google File System. What Google did about 10 years ago is they wrote a whitepaper on how they created this file system and the open-source community took the information from this whitepaper and it was part of the basis of Hadoop.So if you hear GFS and HDFS, they are very, very similar implementations.

**HBase** is a column-oriented database management system that runs on top of **HDFS**.

**HBase** is a NoSQL database that is very commonly used with Hadoop solutions. It is a wide columnstore database. And what that means is it's a database that consists of a key and then one to n number of values. Width of the **column** varies depending on the amount of information that's inserted. And that's called a wide columnstore.

(**HBase** is an open source, non-relational, distributed database modeled after Google's BigTable and written in Java. It is developed as part of Apache Software Foundation's Apache Hadoop project and runs on top of **HDFS** (Hadoop Distributed Filesystem), providing BigTable-like capabilities for Hadoop.)

**CAP Theory**

Consistency 🡺 Transaction

Availability🡺up time

Partitioning🡺 scalability

Traditional RDBMS systems are known for having **consistency** and **availability**, and for having some difficulty, as I mentioned earlier, at the highest levels of partitioning.

What the CAP theory says is that database systems can really only meet two of the three aspects of CAP theory. This is where Hadoop comes into play because as I mentioned earlier the data that's becoming available for businesses and other companies is growing larger and larger and larger, so partitioning and the complexities around partitioning and the expense is causing companies to look at database solutions that support that aspect, and maybe they don't have a need for the other two aspects to be as fully implemented.

**Hadoop Distribution**

In order to understand core components of Hadoop it is imporant to understand Hadoop Distribution.

* **Open Source:** Apache Hadoop
* **Commercial:** Cloudera, Horton works, MapR
* **Cloud Distribution:** AWS (Apache Hadoop, MapR), Microsoft Azure

**Hadoop File Systems**

 Hadoop Ecosystem let's look more closely at the File Systems that can be associated with the Hadoop Distribution.

1. **HDFS** (default):  The HDFS File System has two modes for implementation.

* **Fully distributed:** which will give you the three copies
* **Pseudo-distributed:** which will use the HDFS File System but is designed for testing and will be implemented in a single node on a single machine

1. **Regular File System:** This is called the standalone mode.You’re reducing the complexity by just working with your regular file system.
2. **Cloud File system:** In Amazon it is S3 File System and In Azure BLOB Storage. Which is similar to the Standalone mode in that you are not using HDFS, you’re using a regular file system but choosing a cloud based file system.

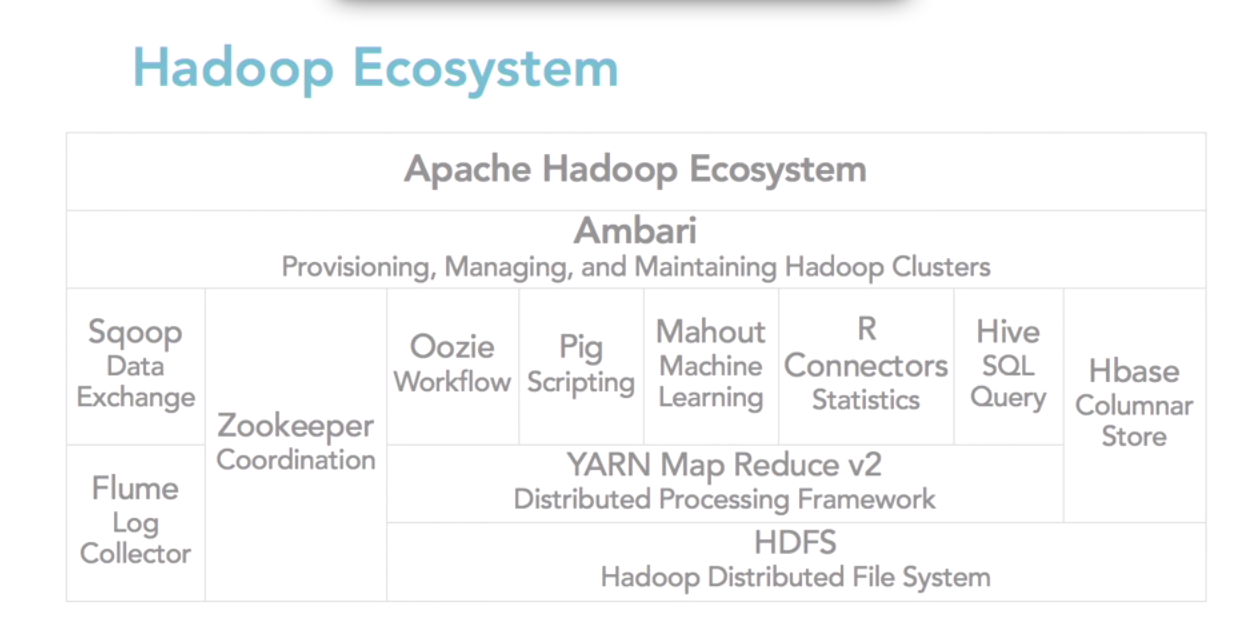
Another consideration is to understand how the file system of choice for your hadoop implementation and JVM interact

🡺 If you deploy Hadoop in Single node you're going to use the Local file system and a single JVM for all the Hadoop processes.

🡺 If you deploy in Pseudo-distributed mode you’re going to use HDFS and Java Deamons are going to run all the processes on a single machine.

🡺 If you run in Fully distributed mode you’re going to use HDFS, it’s going to be **triple replicated** and the daemons are going to run in various locations depending on where you choose to place them.

**Hadoop Ecosystem**



**HDFS:** Cloud implementations is actually a more standard file system like Amazon S3 or Azure blobs depending on the cloud vendor.

But it is common to use HDFS.

**Map Reduce:** Second core part of a Hadoop implementation is Map Reduce. Map reduce version 2 (v2) also called YARN (Yet Another Resource Negotiator).

**Hbase:** Column based database

**Hive:** which is HQL or SQL –like query language that is used to query Hbase.

**Pig**: Scripting language that is used for ETL like processes.

**Mahout**: which is for machine learning or predictive analytics.

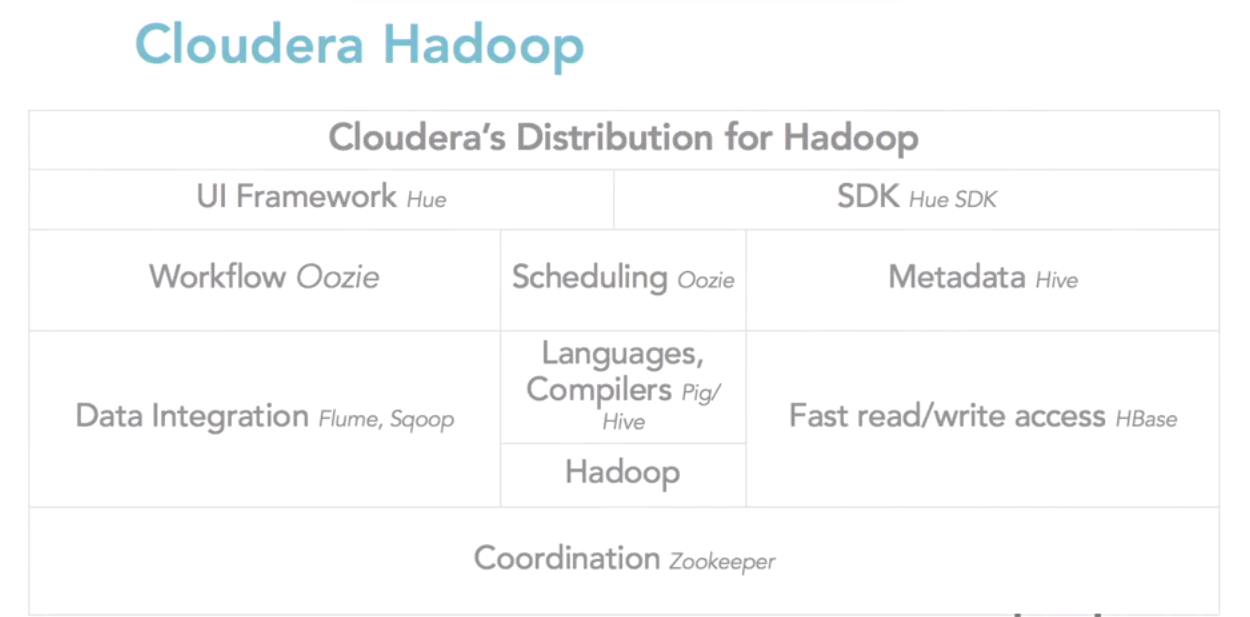
**Oozie**: which is workflow or coordination of jobs.

**Zookeeper**: which is coordination of groups of jobs

**Sqoop**: data exchange in between other systems particularly relation systems like SQL server and Hadoop.

**Flume**: is a log collector because Hadoop jobs produce a large amount of log information about job process because the jobs are running batch, so they take time to run.

**Ambari**: which is provisioning, managing and monitoring Hadoop clusters.



HDFS is a filesystem designed for large-scale distributed data processing under frameworks such as MapReduce. You can store a big data set of (say) 100 TB as a single file in HDFS, something that would overwhelm most other filesystems.

It replicates the data for availability and distributes it over multiple machines to enable parallel processing.

 HDFS abstracts these details away and gives you the illusion that you’re dealing with only a single file.

Basic Hadoop Command format

$/>hadoop fs -cmd <args>