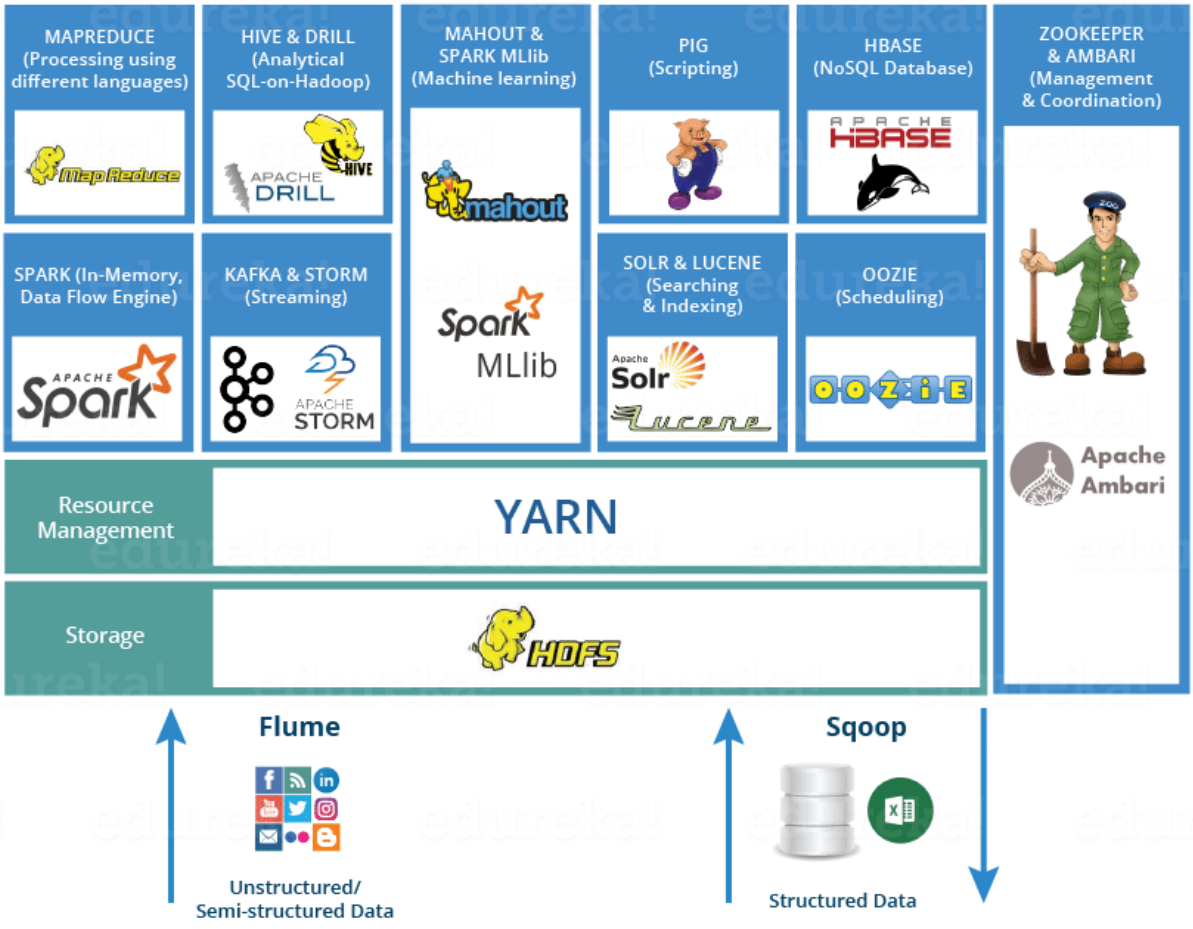
HADOOP ECOSYSTEM

Hadoop Ecosystem is neither a programming language nor a service, it is a platform or framework which solves big data problems. You can consider it as a suite which encompasses a number of services (ingesting, storing, analyzing and maintaining) inside it.

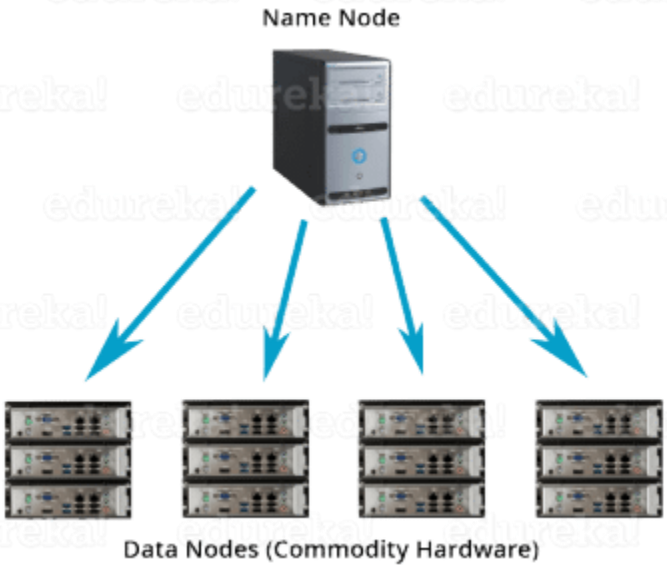
Below are the Hadoop components, that together form a Hadoop ecosystem.

- **HDFS** -> Hadoop Distributed File System
- **YARN** -> Yet Another Resource Negotiator
- **MapReduce** -> Data processing using programming
- **Spark** -> In-memory Data Processing
- **PIG, HIVE**-> Data Processing Services using Query (SQL-like)
- **HBase** -> NoSQL Database
- **Mahout, Spark MLlib** -> Machine Learning
- **Apache Drill** -> SQL on Hadoop
- **Zookeeper** -> Managing Cluster
- **Oozie** -> Job Scheduling
- **Flume, Sqoop** -> Data Ingesting Services
- **Solr& Lucene** -> Searching & Indexing
- **Ambari** -> Provision, Monitor and Maintain cluster



HDFS

- Hadoop Distributed File System is the core component or you can say, the backbone of Hadoop Ecosystem.
- HDFS is the one, which makes it possible to store different types of large data sets (i.e. structured, unstructured and semi structured data).
- HDFS creates a level of abstraction over the resources, from where we can see the whole HDFS as a single unit.
- It helps us in storing our data across various nodes and maintaining the log file about the stored data (metadata).
- HDFS has two core components, i.e. **NameNode and DataNode**.
    1. The **NameNode** is the main node and it doesn't store the actual data. It contains metadata, just like a log file or you can say as a table of content. Therefore, it requires less storage and high computational resources.
    2. On the other hand, all your data is stored on the **DataNodes** and hence it requires more storage resources. These DataNodes are commodity hardware (like your laptops and desktops) in the distributed environment. That's the reason, why Hadoop solutions are very cost effective.
    3. You always communicate to the NameNode while writing the data. Then, it internally sends a request to the client to store and replicate data on various DataNodes.



YARN

Consider YARN as the brain of your Hadoop Ecosystem. It performs all your processing activities by allocating resources and scheduling tasks.

- It has two major components, i.e. **Resource Manager and Node Manager**.
    1. **Resource Manager** is again a main node in the processing department.
    2. It receives the processing requests, and then passes the parts of requests to corresponding Node Managers accordingly, where the actual processing takes place.
    3. **Node Managers** are installed on every Data Node. It is responsible for execution of task on every single Data Node.

-
    1. **Schedulers:** Based on your application resource requirements, Schedulers perform scheduling algorithms and allocates the resources.
    2. **Applications Manager:** While Applications Manager accepts the job submission, negotiates to containers (i.e. the Data node environment where process executes) for executing the application specific Application Master and monitoring the progress. ApplicationMasters are the deamons which reside on DataNode and communicates to containers for execution of tasks on each DataNode.
    3. ResourceManager has two components: Schedulers and application manager

MAPREDUCE

It is the core component of processing in a Hadoop Ecosystem as it provides the logic of processing. In other words, MapReduce is a software framework which helps in writing applications that processes large data sets using distributed and parallel algorithms inside Hadoop environment.

- In a MapReduce program, **Map() and Reduce()** are two functions.
    1. The **Map function** performs actions like filtering, grouping and sorting.
    2. While **Reduce function** aggregates and summarizes the result produced by map function.
    3. The result generated by the Map function is a key value pair (K, V) which acts as the input for Reduce function.

Student	Department	Count	(Key, Value), Pair
Student 1	D1	1	(D1, 1)
Student 2	D1	1	(D1, 1)
Student 3	D1	1	(D1, 1)
Student 4	D2	1	(D2, 1)
Student 5	D2	1	(D2, 1)
Student 6	D3	1	(D3, 1)
Student 7	D3	1	(D3, 1)

Let us take the above example to have a better understanding of a MapReduce program.

We have a sample case of students and their respective departments. We want to calculate the number of students in each department. Initially, Map program will execute and calculate the students appearing in each department, producing the key value pair as mentioned above. This key value pair is the input to the Reduce function. The Reduce function will then aggregate each department and calculate the total number of students in each department and produce the given result.

Department	Total Student
D1	3
D2	2
D3	2

APACHE PIG

- PIG has two parts: **Pig Latin**, the language and **the pig runtime,** for the execution environment. You can better understand it as Java and JVM.
- It supports *pig latin* language, which has SQL like command structure.

10 line of pig latin = approx. 200 lines of Map-Reduce Java code

But don't be shocked when I say that at the back end of Pig job, a map-reduce job executes.

- The compiler internally converts pig latin to MapReduce. It produces a sequential set of MapReduce jobs, and that's an abstraction (which works like black box).
- PIG was initially developed by Yahoo.

- It gives you a platform for building data flow for ETL (Extract, Transform and Load), processing and analyzing huge data sets.

How Pig works?

In PIG, first the load command, loads the data. Then we perform various functions on it like grouping, filtering, joining, sorting, etc. At last, either you can dump the data on the screen or you can store the result back in HDFS.

APACHE HIVE

- Facebook created HIVE for people who are fluent with SQL. Thus, HIVE makes them feel at home while working in a Hadoop Ecosystem.
- Basically, HIVE is a data warehousing component which performs reading, writing and managing large data sets in a distributed environment using SQL-like interface.

HIVE + SQL = HQL

- The query language of Hive is called Hive Query Language(HQL), which is very similar like SQL.
- It has 2 basic components: **Hive Command Line and JDBC/ODBC driver**.
- The **Hive Command line** interface is used to execute HQL commands.
- While, Java Database Connectivity (**JDBC**) and Object Database Connectivity (**ODBC**) is used to establish connection from data storage.

- Secondly, Hive is highly scalable. As, it can serve both the purposes, i.e. large data set processing (i.e. Batch query processing) and real time processing (i.e. Interactive query processing).
- It supports all primitive data types of SQL.
- You can use predefined functions, or write tailored user defined functions (UDF) also to accomplish your specific needs.

APACHE MAHOUT

Now, let us talk about Mahout which is renowned for machine learning. Mahout provides an environment for creating machine learning applications which are scalable.

Machine learning algorithms allow us to build self-learning machines that evolve by itself without being explicitly programmed. Based on user behaviour, data patterns and past experiences it makes important future decisions. You can call it a descendant of Artificial Intelligence (AI).

What Mahout does?

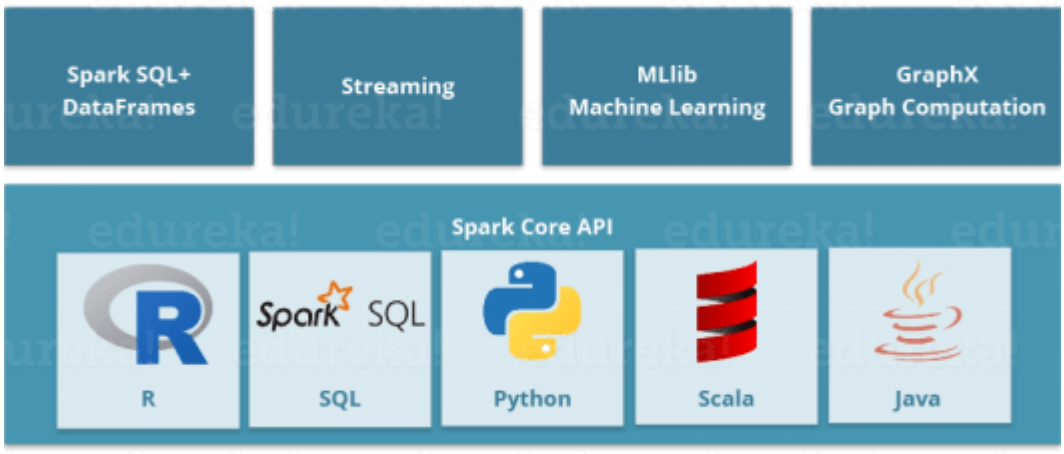
It performs **collaborative filtering, clustering and classification.** Some people also consider **frequent item set missing** as Mahout's function. Let us understand them individually:

1. **Collaborative filtering:** Mahout mines user behaviors, their patterns and their characteristics and based on that it predicts and make recommendations to the users. The typical use case is E-commerce website.
2. **Clustering:** It organizes a similar group of data together like articles can contain blogs, news, research papers etc.
3. **Classification**: It means classifying and categorizing data into various sub-departments like articles can be categorized into blogs, news, essay, research papers and other categories.
4. **Frequent item set missing**: Here Mahout checks, which objects are likely to be appearing together and make suggestions, if they are missing. For example, cell phone and cover are brought together in general. So, if you search for a cell phone, it will also recommend you the cover and cases.

Mahout provides a command line to invoke various algorithms. It has a predefined set of library which already contains different inbuilt algorithms for different use cases.

APACHE SPARK

- Apache Spark is a framework for real time data analytics in a distributed computing environment.
- The Spark is written in Scala and was originally developed at the University of California, Berkeley.
- It executes in-memory computations to increase speed of data processing over Map-Reduce.
- It is 100x faster than Hadoop for large scale data processing by exploiting in-memory computations and other optimizations. Therefore, it requires high processing power than Map-Reduce.



As you can see, Spark comes packed with high-level libraries, including support for R, SQL, Python, Scala, Java etc. These standard libraries increase the seamless integrations in complex workflow. Over this, it also allows various sets of services to integrate with it like MLlib, GraphX, SQL + Data Frames, Streaming services etc. to increase its capabilities.

. Apache Spark best fits for real time processing, whereas Hadoop was designed to store unstructured data and execute batch processing over it. When we combine, Apache Spark's ability, i.e. high processing speed, advance analytics and multiple integration support with Hadoop's low cost operation on commodity hardware, it gives the best results.

That is the reason why, Spark and Hadoop are used together by many companies for processing and analyzing their Big Data stored in HDFS.

APACHE HBASE

- HBase is an open source, non-relational distributed database. In other words, it is a NoSQL database.
- It supports all types of data and that is why, it's capable of handling anything and everything inside a Hadoop ecosystem.
- It is modelled after Google's BigTable, which is a distributed storage system designed to cope up with large data sets.
- The HBase was designed to run on top of HDFS and provides BigTable like capabilities.
- It gives us a fault tolerant way of storing sparse data, which is common in most Big Data use cases.
- The HBase is written in Java, whereas HBase applications can be written in REST, Avro and Thrift APIs.

For better understanding, let us take an example. You have billions of customer emails and you need to find out the number of customers who has used the word complaint in their emails. The request needs to be processed quickly (i.e. at real time). So, here we are handling a large data set while retrieving a small amount of data. For solving these kind of problems, HBase was designed.

APACHE DRILL

Apache Drill is used to drill into any kind of data. It's an open source application which works with distributed environment to analyze large data sets.

- It is a replica of Google Dremel.
- It supports different kinds NoSQL databases and file systems, which is a powerful feature of Drill. For example: Azure Blob Storage, Google Cloud Storage, HBase, MongoDB, MapR-DB HDFS, MapR-FS, Amazon S3, Swift, NAS and local files.

So, basically the main aim behind Apache Drill is to provide scalability so that we can process petabytes and exabytes of data efficiently (or you can say in minutes).

- The main power of Apache Drill lies in ***combining a variety of data stores just by using a single query.***

- Apache Drill basically follows the ANSI SQL.
- It has a powerful scalability factor in supporting millions of users and serve their query requests over large scale data.

APACHE ZOOKEEPER

- Apache Zookeeper is the coordinator of any Hadoop job which includes a combination of various services in a Hadoop Ecosystem.
- Apache Zookeeper coordinates with various services in a distributed environment.

Before Zookeeper, it was very difficult and time consuming to coordinate between different services in Hadoop Ecosystem. The services earlier had many problems with interactions like common configuration while synchronizing data. Even if the services are configured, changes in the configurations of the services make it complex and difficult to handle. The grouping and naming was also a time-consuming factor.

Due to the above problems, Zookeeper was introduced. It saves a lot of time by performing **synchronization, configuration maintenance, grouping and naming.**

Although it's a simple service, it can be used to build powerful solutions.

APACHE OOZIE

Consider Apache Oozie as a clock and alarm service inside Hadoop Ecosystem. For Apache jobs, Oozie has been just like a scheduler. It schedules Hadoop jobs and binds them together as one logical work.

There are two kinds of Oozie jobs:

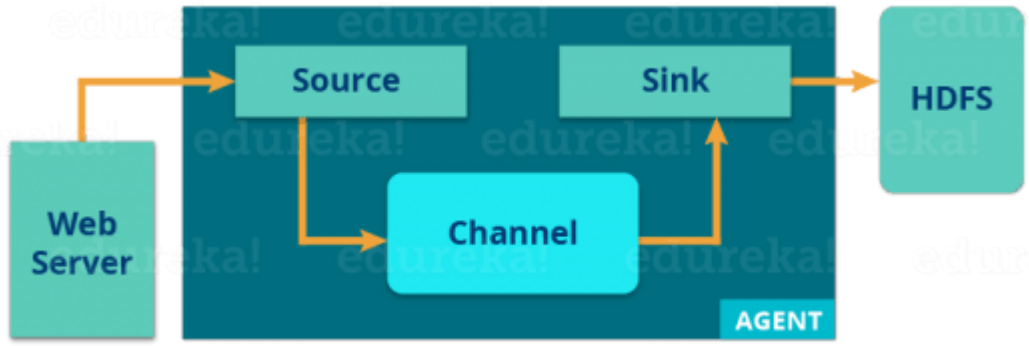
1. **Oozie workflow**: These are sequential set of actions to be executed**.** You can assume it as a relay race. Where each athlete waits for the last one to complete his part.
2. **Oozie Coordinator**: These are the Oozie jobs which are triggered when the data is made available to it. Think of this as the response-stimuli system in our body. In the same manner as we respond to an external stimulus, an Oozie coordinator responds to the availability of data and it rests otherwise.

APACHE FLUME

Ingesting data is an important part of our Hadoop Ecosystem.

- The Flume is a service which helps in ingesting unstructured and semi-structured data into HDFS.
- It gives us a solution which is reliable and distributed and helps us in **collecting, aggregating** and **moving large amount of data sets**.
- It helps us to ingest online streaming data from various sources like network traffic, social media, email messages, log files etc. in HDFS.

Now, let us understand the architecture of Flume from the below diagram:



There is a **Flume agent** which ingests the streaming data from various data sources to HDFS. From the diagram, you can easily understand that the web server indicates the data source. Twitter is among one of the famous sources for streaming data.

The flume agent has 3 components: **source, sink and channel**.

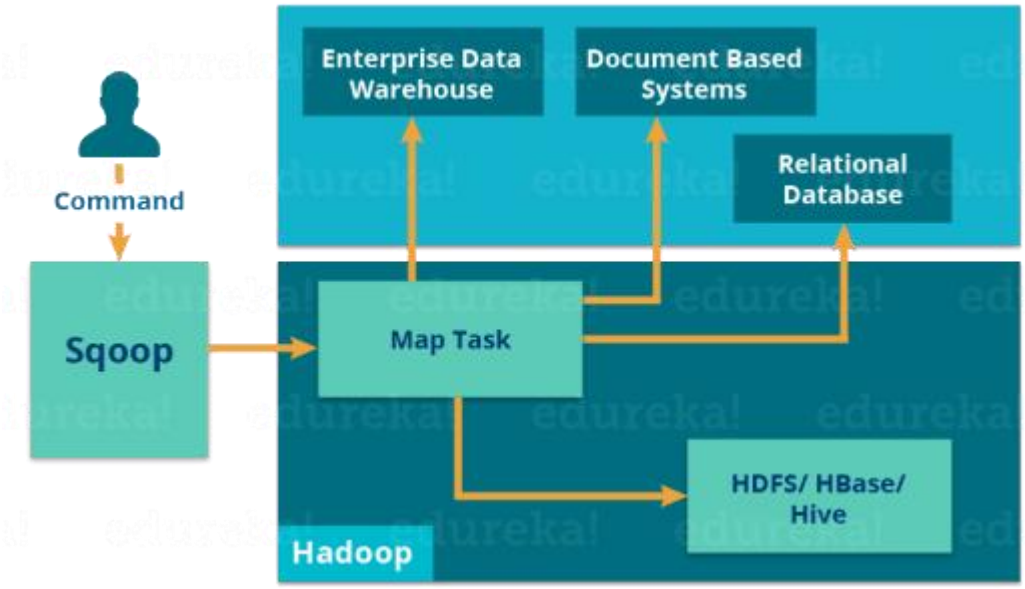
1. **Source**: it accepts the data from the incoming streamline and stores the data in the channel.
2. **Channel:** it acts as the local storage or the primary storage. A Channel is a temporary storage between the source of data and persistent data in the HDFS.
3. **Sink:** Then, our last component i.e. Sink, collects the data from the channel and commits or writes the data in the HDFS permanently.

APACHE SQOOP

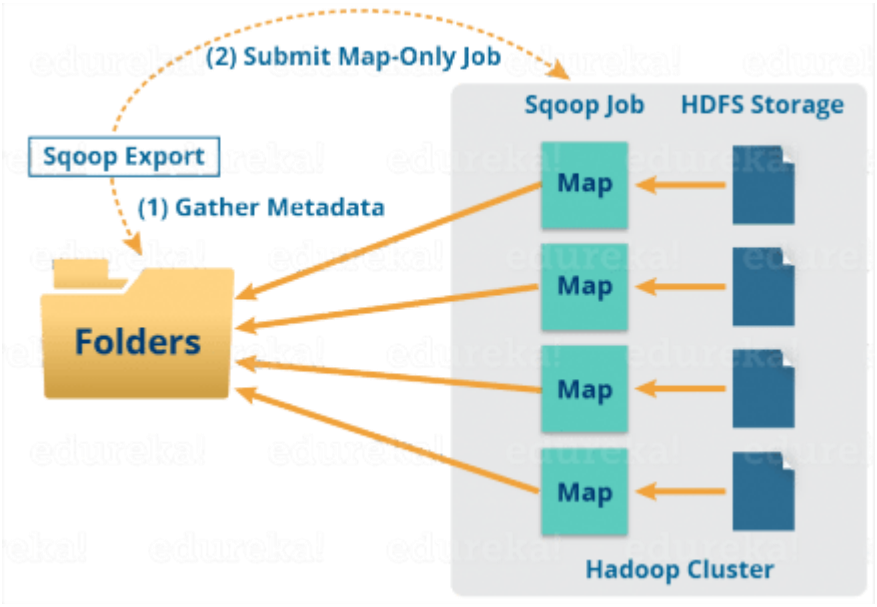
The major difference between Flume and Sqoop is that:

- Flume only ingests unstructured data or semi-structured data into HDFS.
- While Sqoop can import as well as export structured data from RDBMS or Enterprise data warehouses to HDFS or vice versa.

Let us understand how Sqoop works using the below diagram:



When we submit Sqoop command, our main task gets divided into sub tasks which is handled by individual Map Task internally. Map Task is the sub task, which imports part of data to the Hadoop Ecosystem. Collectively, all Map tasks imports the whole data.



Export also works in a similar manner.

When we submit our Job, it is mapped into Map Tasks which brings the chunk of data from HDFS. These chunks are exported to a structured data destination. Combining all these exported chunks of data, we receive the whole data at the destination, which in most of the cases is an RDBMS (MYSQL/Oracle/SQL Server).

APACHE SOLR & LUCENE

Apache Solr and Apache Lucene are the two services which are used for searching and indexing in Hadoop Ecosystem.

- Apache Lucene is based on Java, which also helps in spell checking.
- If Apache Lucene is the engine, Apache Solr is the car built around it. Solr is a complete application built around Lucene.

- It uses the Lucene Java search library as a core for search and full indexing.

APACHE AMBARI

Ambari is an Apache Software Foundation Project which aims at making Hadoop ecosystem more manageable.

It includes software for **provisioning, managing and monitoring** Apache Hadoop clusters.

The Ambari provides:

1. **Hadoop cluster provisioning**:
    - It gives us step by step process for installing Hadoop services across a number of hosts.
    - It also handles configuration of Hadoop services over a cluster.
2. **Hadoop cluster management:**
    - It provides a central management service for starting, stopping and re-configuring Hadoop services across the cluster.
3. **Hadoop cluster monitoring:**
    - For monitoring health and status, Ambari provides us a dashboard.
    - The **Amber Alert framework** is an alerting service which notifies the user, whenever the attention is needed. For example, if a node goes down or low disk space on a node, etc.

At last, I would like to draw your attention on three things importantly:

1. Hadoop Ecosystem owes its success to the whole developer community, many big companies like Facebook, Google, Yahoo, University of California (Berkeley) etc. have contributed their part to increase Hadoop's capabilities.
2. Inside a Hadoop Ecosystem, knowledge about one or two tools (Hadoop components) would not help in building a solution. You need to learn a set of Hadoop components, which works together to build a solution.
3. Based on the use cases, we can choose a set of services from Hadoop Ecosystem and create a tailored solution for an organization.