**Components of Hadoop 2.x** :

* Hadoop 2.x has some common Hadoop API which can easily be integrated with any third party applications to work with Hadoop
* It has some new Java APIs and features in HDFS and MapReduce which are known as HDFS2 and MR2 respectively
* New architecture has added the architectural features like HDFS High Availability and HDFS Federation
* Hadoop 2.x not using Job Tracker and Task Tracker daemons for resource management now on-wards, it is using YARN (Yet Another Resource Negotiator) for Resource Management.

In addition., to handle failure of name node below concepts have been introduced:

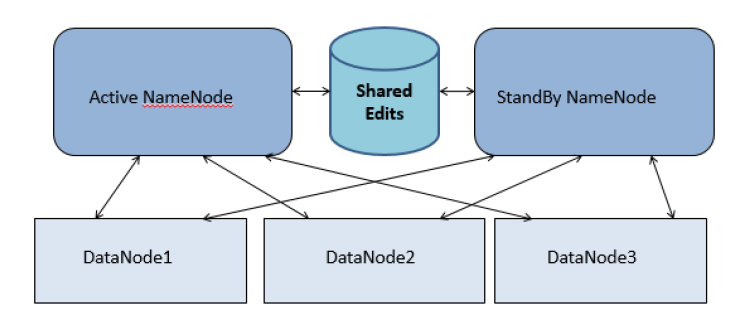
* HDFS High Availability (HA)
* HDFS Federation (HF)

**HDFS High Availability (HA)**

**Problem:**  As you know in Hadoop 1.x architecture Name Node was a single point of failure, which means if your Name Node daemon is down somehow, you don’t have access to your Hadoop Cluster than after. How to deal with this problem?

**Solution:**  Hadoop 2.x is featured with Name Node HA which is referred as HDFS High Availability (HA).

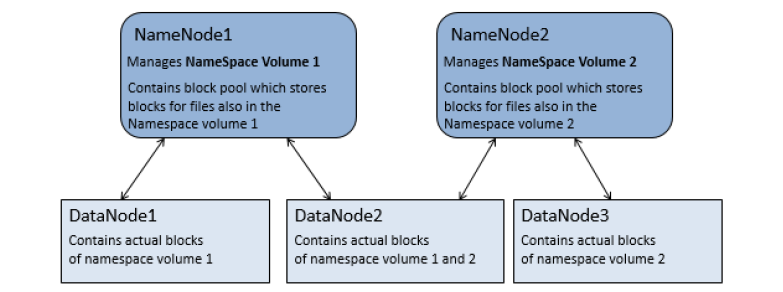
* Hadoop 2.x supports two Name Nodes at a time one node is active and another is standby node
* Active Name Node handles the client operations in the cluster
* Standby Name Node manages metadata same as Secondary Name Node in Hadoop 1.x
* When Active Name Node is down, Standby Name Node takes over and will handle the client operations then after
* HDFS HA can be configured by two ways
  + Using Shared NFS Directory
  + Using Quorum Journal Manager



**HDFS Federation**

**Problem:**  HDFS uses namespaces for managing directories, file and block level information in cluster. Hadoop 1.x architecture was able to manage only single namespace in a whole cluster with the help of the Name Node (which is a single point of failure in Hadoop 1.x). Once that Name Node is down you loose access of full cluster data. It was not possible for partial data availability based on name space.

**Solution:**  Above problem is solved by HDFS Federation in Hadoop 2.x Architecture which allows to manage multiple namespaces by enabling multiple Name Nodes. So on HDFS shell you have multiple directories available but it may be possible that two different directories are managed by two active Name Nodes at a time.

HDFS Federation

HDFS Federation by default allows single Name Node to manage full cluster (same as in Hadoop 1.x)

**Hadoop 2.x Architecture In Detail**

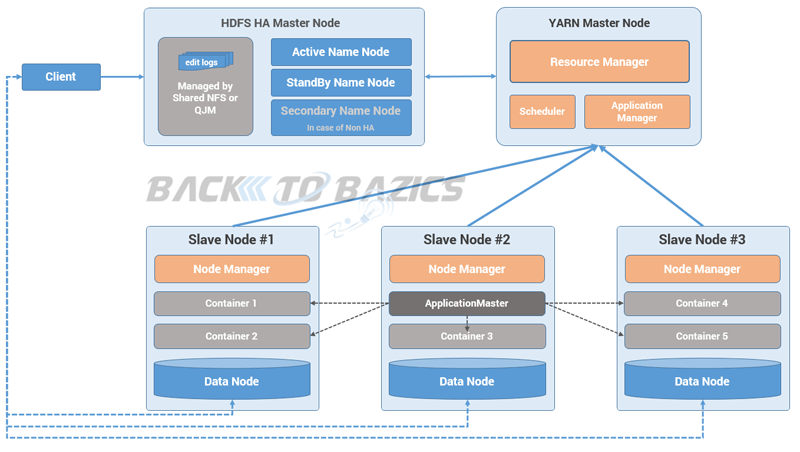
Hadoop2 Architecture has mainly 2 set of daemons

* **HDFS 2.x Daemons:**  Name Node, Secondary Name Node (not required in HA) and Data Nodes
* **MapReduce 2.x Daemons (YARN):**  Resource Manager, Node Manager

**HDFS 2.x Daemons**

The working methodology of HDFS 2.x daemons is same as it was in Hadoop 1.x Architecture with following differences.

* Hadoop 2.x allows Multiple Name Nodes for HDFS Federation
* New Architecture allows HDFS High Availability mode in which it can have Active and StandBy Name Nodes (No Need of Secondary Name Node in this case)
* Hadoop 2.x Non HA mode has same Name Node and Secondary Name Node working same as in Hadoop 1.x architecture

[](http://backtobazics.com/wp-content/themes/twentyfourteen/images/hadoop/hadoop2-architecture.png)Hadoop 2.x Architecture

**MapReduce 2.x Daemons (YARN)**

MapReduce2 has replace old daemon process Job Tracker and Task Tracker with YARN components Resource Manager and Node Manager respectively. These two components are responsible for executing distributed data computation jobs in Hadoop 2(Refer my post on [***YARN Architecture***](http://backtobazics.com/big-data/yarn-architecture-and-components/) for further understanding).

**Resource Manager**

* This daemon process runs on master node (may run on the same machine as name node for smaller clusters)
* It is responsible for getting job submitted from client and schedule it on cluster, monitoring running jobs on cluster and allocating proper resources on the slave node
* It communicates with Node Manager daemon process on the slave node to track the resource utilization
* It uses two other processes named *Application Manager*and *Scheduler*for MapReduce task and resource management

**Node Manager**

* This daemon process runs on slave nodes (normally on HDFS Data node machines)
* It is responsible for coordinating with Resource Manager for task scheduling and tracking the resource utilization on the slave node
* It also reports the resource utilization back to the Resource Manager
* It uses other daemon process like Application Master and Container for MapReduce task scheduling and execution on the slave node