# Introduction

MapR is a Hadoop distribution that includes HBase, Pig, Hive, Mahout, Cascading, Sqoop, Flume and more. MapR’s distribution is 100% API compatible with Hadoop (MapReduce, HDFS and HBase). MapR Technologies makes Big Data analytics easy, dependable, and fast.

There are various factors regarding data that influence the systems handling them. Some of them are

**Data Volume**: The size of available data has been growing at an increasing rate. A text file is a few kilo bytes, a sound file is a few mega bytes while a full length movie is a few giga bytes. More sources of data are added on continuous basis. Peta byte data sets are common these days and Exabyte is not far away.

**Data Velocity** :Initially, companies analyzed data using a batch process. One takes a chunk of data, submits a job to the server and waits for delivery of the result. That scheme works when the incoming data rate is slower than the batch processing rate and when the result is useful despite the delay. With the new sources of data such as social and mobile applications, the batch process breaks down. The data is now streaming into the server in real time, in a continuous fashion and the result is only useful if the delay is very short.

**Data Variety** : Pure text, photo, audio, video, web, GPS data, sensor data, relational data bases, documents, SMS, pdf, flash, etc. One no longer has control over the input data format. Structure can no longer be imposed like in the past in order to keep control over the analysis. As new applications are introduced new data formats come to life.

**Veracity** refers to the messiness or trustworthiness of the data. With many forms of big data, quality and accuracy are less controllable (just think of Twitter posts with hash tags, abbreviations, typos and colloquial speech as well as the reliability and accuracy of content) but big data and analytics technology now allows us to work with these type of data. The volumes often make up for the lack of quality or accuracy.

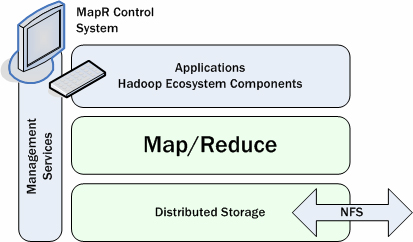
**Value**: Then there is another V to take into account when looking at Big Data: Value! It is all well and good having access to big data but unless we can turn it into value it is useless. So you can safely argue that 'value' is the most important V of Big Data. It is important that businesses make a business case for any attempt to collect and leverage big data. It is so easy to fall into the buzz trap and embark on big data initiatives without a clear understanding of costs and benefits.

Hadoop was introduced to deal with such data in batch processing, real time, streaming in a distributed architecture.

MapR specifically improves the performance and robustness of the distributed file system, eliminating the Namenode. The MapR distribution for Hadoop supports continuous read/write access, improving data load and unload processes.

To reiterate, **MapR Hadoop does not use Namenodes**.

The diagram below illustrates the services surrounding the basic Hadoop idea of Map and Reduce operations performed across a distributed storage system. Some services provide management and others run at the application level.



* MapReduce services: JobTracker, TaskTracker
* Storage services: CLDB, FileServer, HBase RegionServer, NFS
* Management services: HBase Master, Webserver, ZooKeeper

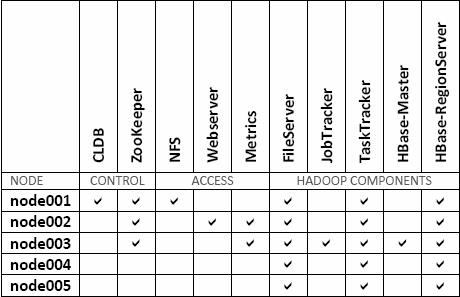
Warden : A daemon called the warden runs on every node to make sure that the proper services are running (and to allocate resources for them). The only service that the warden doesn’t control is the ZooKeeper. Part of the ZooKeeper’s job is to have knowledge of the whole cluster; in the event that a service fails on one node, it is the ZooKeeper that tells the warden to start the service on another node.

## ZooKeeper: The MapR cluster uses Apache ZooKeeper to coordinate services and enables high availability (HA) and fault tolerance for MapR clusters. The Warden will not start any services unless ZooKeeper is reachable and more than half of the configured ZooKeeper nodes (a quorum) are live.

Service Layout:

Before installing services, it’s important to determine which services to run and where to run them. A lot of the factors that weigh on this decision have to do with the size of your cluster, the anticipated load, and the kinds of jobs you plan to run. Here are a few tips to help you understand service layout.

* General Tips  
  If you are using MapR Metrics, avoid running the MySQL server that supports the MapR Metrics service on a CLDB node. Consider running the MySQL server on a machine external to the cluster to prevent the MySQL server’s resource needs from affecting services on the cluster. For the same reason, avoid running the webserver on CLDB nodes. Queries to the MapR Metrics service can impose a bandwidth load that reduces CLDB performance.   
  In general, the FileServer should run on all nodes.
* Large Clusters  
  In a large cluster, the JobTracker, CLDB, and ZooKeeper nodes are likely to get a lot of requests. With the ZooKeeper especially, latency is a bad thing. The ZooKeeper is not particularly resource-hungry, but if it gets slowed down waiting for some other process to relinquish CPU cycles, it can introduce problems to the cluster. For this reason, you should avoid putting the ZooKeeper on the same node as the CLDB or JobTracker in a large cluster. You should even consider isolating the ZooKeeper completely—that is, on ZooKeeper nodes, it’s best not to run any other services at all. On clusters over 250 nodes, isolate the JobTracker service on a dedicated node.  
    
  If possible, you should run three or five ZooKeepers on different racks. ZooKeeper maintains the integrity of its information through majority consensus: if more than half the ZooKeeper nodes agree on something, it is considered true. If fewer than half the ZooKeeper nodes are present, or able to agree, then ZooKeeper stops the cluster until the problem is resolved. This means that if you run three ZooKeeper nodes, you can tolerate one ZooKeeper failure (two is more than half of three). If you run five ZooKeeper nodes, you can tolerate two ZooKeeper failures (three is more than half of five). There’s no point in running an even number of ZooKeeper nodes—if you run four, for example, then you can only tolerate one ZooKeeper failure since two is not more than half of four.
* Small Clusters  
  On very small clusters of just a few nodes, it’s impractical to isolate services on dedicated nodes. One layout approach is to run one CLDB and one ZooKeeper on the same node, leaving the other nodes free to run the TaskTracker. All nodes should run the FileServer. If you need HA in a small cluster, you will end up running the CLDB and ZooKeeper on additional nodes.

Here is a sample layout:  


# MapR Cluster Installation

The MapR cluster can be installed either manually or using a MapR Installer ( UI).

**Prerequisites for Installation**

1. Plan the cluster
2. Decide the number of nodes. This can be decided on the data volume

Eg: If there is a need to set up MapR cluster for 100TB of data,

Data sizing should consider 3 times replication + temp space + 25%

Cluster size = (100TB \* 3 + 100 TB + 25% of 100TB ) / ( 12 \* 2TB disks)

= 17.3 …so it should be 20 nodes

1. Check the OS dependencies. For Redhat and Centos, below packages are must

nss,python-pycurl,openssh-clients ,openssh-server,openssl ,sshpass,

sudo,wget,which,httpd,bash,rpcbind,dmidecode,glibc,hdparm,,initscripts,

Iputils,irqbalance,libgcc,libstdc++,redhat-lsb-core,rpm-libs,sdparm,shadow-utils,syslinux,unzip,zip,Ccyrus-sasl-gssapi,g++

1. Java : OpenJDK 1.8 (It has to be openjdk). Set the JAVA\_HOME to point to the correct path where OpenJdk is installed
2. Create uid and gid for mapr user which should be consistent on all the nodes

Example:

* groupadd –g 5000 mapr ( This will add group called mapr with id 5000)
* useradd   –u 5000 –g 5000 mapr ( This will add user called mapr with id 5000 and group id 5000)

adding password for mapr

* passwd mapr ( this will ask for a password)

1. In /etc/hosts of each node , add the IP and hostname of all the other nodes

# Installation using UI - MapR Installer ( without Internet)

1. Create a local repository for both OS and MapR
   * 1. **Creating a local repository for MapR**
   1. Create the directory /var/www/html/yum/base on the node through which installation will be done
   2. Download the below files on a machine having internet from **package.mapr.com**

* mapr-v5\*.rpm.tgz ( eg: mapr-latest-201602270100.rpm.tar.gz)
* mapr-ecosystem-\*.rpm.tgz
* mapr-installer-\*.noarch.rpm
* mapr-installer-definitions-\*.noarch.rpm
* mapr-setup.sh

and place the files in /var/www/html/yum/base directory

* 1. Go to the directory /etc/yum.repos.d , Create a file maprtech.repo in this path if not exists and edit the file as below

[maprtech]

Name = MapR Technologies

baseurl=file:/// var/www/html/yum/base/

enabled=1

gpgcheck=0

[maprecosystem]

Name = MapR Technologies

baseurl=file:/// var/www/html/yum/base/

enabled=1

gpgcheck=0

* 1. Execute the command

Createrepo / var/www/html/yum/base/

* + 1. **Create a local repos for OS ( here eg of Centos local repos)**
* Copy CentOS-6.5-x86\_64-bin-DVD1.iso into /home/mapr
* Create a directory called dvd in root
* Mount the iso in dvd. Give the below command to mount the same.

mount –t iso9660 –o loop /home/mapr/CentOS-6.5-x86\_64-bin-DVD1.iso /dvd

* Go to /yum/repos.d, rename all the .repo files to remove .repo except CentOS-Media.repo , maprtech.repo , mapr-installer.repo and mapr-ecosystem.repo
* Edit the above CentOS-Media.repo files to point baseurl to the path <file:///dvd>
* Edit the other Mapr\*.repo files to point baseurl to the path

file:/// var/www/html/yum/base/

* + 1. **MapR Installer**

Run the below command using root to open the MapR Installer UI

**bash ./mapr-setup.sh –a mapr-v5.1.0GA.rpm.tgz**

**This will ask few parameters, keep all the default parameters and you will be asked to open the URL as https://<ip>:9443**

**The browser based installation will give navigation and screens**

# Manual Installation

* 1. Create the directory /var/www/html/yum/base on all the node which are a part of MapR clusrer
  2. Download the below files on all the nodes from **package.mapr.com**
* mapr-v5\*.rpm.tgz ( eg: mapr-latest-201602270100.rpm.tar.gz)
* mapr-ecosystem-\*.rpm.tgz

and untar the files in /var/www/html/yum/base directory

* + - tar -xvzf mapr-v5-\*.rpm.tgz
    - tar -xvzf mapr-ecosystem-\*.rpm.tgz
  1. Execute the individual MapR packages. Identify the control node and data nodes and accordingly install the packages on them

rpm -ivh mapr-core-internal-5.0.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-hadoop-core-2.7.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-mapreduce1-0.20.2.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-mapreduce2-2.7.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-core-5.0.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-fileserver-5.0.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-cldb-5.0.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-zk-internal-5.0.0.32987.GA.v3.4.5-1.x86\_64.rpm

rpm -ivh mapr-zookeeper-5.0.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-nfs-5.0.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-webserver-5.0.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-tasktracker-5.0.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-resourcemanager-2.7.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-nodemanager-2.7.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-jobtracker-5.0.0.32987.GA-1.x86\_64.rpm

rpm -ivh mapr-historyserver-2.7.0.32987.GA-1.x86\_64.rpm

* 1. Check if cldb and fileserver roles have been created

Cd /opt/mapr/roles.

* 1. Add disks
     + Create a file disks.txt in /tmp directory
     + In Disks.txt, add /dev/sdb
     + /opt/mapr/server/disksetup -F /tmp/disks.txt
  2. Run configure.sh command to add services

/opt/mapr/server/configure.sh –C <ip1>,<1p2> –Z <ip1>,<ip2>,<ip3> –N MapRCluster

* 1. Check if the services have started

Service mapr-zookeeper qstatus

Service mapr-cldb status

Check if cldb has started

>maprcli node cldbmaster

1. Permissions on the cluster

The users and groups should be given permissions to login and do other administrative jobs.

Once the cluster has been setup, give full control access to mapr user

>maprcli acl edit –type cluster –user mapr:fc

1. Security using ACL’s and ACE on Volumes & MapRDB tables

 A volume is a logical unit that allows you to apply policies to a set of files, directories, and sub-volumes. A well-structured volume hierarchy is an essential aspect of your cluster's performance. As your cluster grows, keeping your volume hierarchy efficient maximizes your data's availability. Without a volume structure in place, your cluster's performance will be negatively affected.

Volumes should be used to enforce disk usage limits, set replication levels, establish ownership and accountability, and measure the cost generated by different projects or departments. Create a volume for each user, department, or project. You can mount volumes under other volumes to build a structure that reflects the needs of your organization. The volume structure defines how data is distributed across the nodes in your cluster. Create multiple small volumes with shallow paths at the top of your cluster's volume hierarchy to spread the load of access requests across the nodes.

On a cluster with an M5 or M7 license, you can create a special type of volume called a [mirror](http://doc.mapr.com/display/MapR3/Working+with+Mirror+Volumes), a local or remote read-only copy of an entire volume. Mirrors are useful for load balancing or disaster recovery. You can also create a [snapshot](http://doc.mapr.com/display/MapR3/Working+with+Snapshots), an image of a volume at a specific point in time. Snapshots are useful for rollback to a known data set. You can create snapshots and synchronize mirrors manually or using a [schedule](http://doc.mapr.com/display/MapR3/Schedules).

Create Directories and create volumes within the directories . give different user access permissions to work on volumes.

* Create different groups and users and give different access control permissions to access the directories and volumes created within .

Create everything though root

groupadd –g 6000 mkt

useradd   –u 601 –g 6000 sharon

passwd Sharon (---to set password for the user)

useradd -u 602 -g 6000 mktreadonly

passwd mktreadonly (---to set password for the user)

groupadd –g 7000 rnd

useradd   –u 701 –g 7000 jenn

passwd jenn (---to set password for the user)

* Change the group for the directory path

hadoop fs -chgrp -R mkt /Mastek/Marketing

hadoop fs -chgrp -R rnd /Mastek/RnD

* Eg: Create a directory Mastek. Create 2 different directories called Marketting and RnD in the Projects

hadoop fs –mkdir /Mastek

hadoop fs –mkdir /Mastek/Marketing

hadoop fs –mkdir /Mastek/RnD

* Create volumes within DevRnD and ProdRnD and give access to different groups and users using ACE and ACL’s.

maprcli volume create -name DevRnD -path /Mastek/RnD/DevRnD

maprcli volume create -name ProdRnD -path /Mastek/RnD/ProdRnD

maprcli volume create -name ProdMkt -path /Mastek/RnD/ProdMkt

* Setting the permission for volumes using maprcli,

maprcli acl set -type volume -name ProdMkt -user sharon:fc -group mkt:dump,restore,m

* Create MapRDB table using maprcli command

maprcli table create -path /Mastek/Marketing/ProdMkt/table1

maprcli table cf create -path /Mastek/Marketing/ProdMkt/table1 -cfname cf1

* Give ACE’s to maprdb table table1 : access control expressions to give permissions on the object using AND or OR operations

(eg: give create/rename column family permission to group mapr and mkt but restrict access to user Sharon that belongs to mkt group)

maprcli table create –path /Mastek/Marketing/ProdMkt/table1 –createrenamefamilyperm ‘g:mapr|(g:mkt & !u:Sharon)’

* Deleteing the MaprDB table using maprcli

maprcli table delete -path /Mastek/Marketing/ProdMkt/table1

* Deting column family

maprcli table cf delete -path /Mastek/Marketing/ProdMkt/table1 -cfname cf1

* Loading data in maprdb table

hadoop jar /opt/mapr/hbase/hbase-0.98.12/lib/hbase-server-0.98.12-mapr-1506.jar importtsv -Dimporttsv.separator=, -Dimporttsv.columns=cf1:CAMPAIGN\_CODE,HBASE\_ROW\_KEY,cf1:PRODUCT\_CODE,cf1:ELIGIBILITY,cf1:OFFER\_AMOUNT,cf1:START\_DATE,cf1:END\_DATE,cf1:DESCRIPTION,cf1:FLAG,cf1:ACCOUNT\_ID,cf1:RACK\_RATE,cf1:INTEREST\_RATE,cf1:PROC\_FEES,cf1:CSA\_YESNO,cf1:PRE\_CLOSURE,cf1:AL\_SEGMENT,cf1:CC\_SEGMENT,cf1:CARD\_TYPE1,cf1:PRICING1,cf1:CARD\_TYPE2,cf1:PRICING2,cf1:CARD\_TYPE3,cf1:PRICING3,cf1:TENURE,cf1:RACK\_PROCESSING\_FEE,cf1:TAX '/Mastek/Marketing/ProdMkt/table1' /Mastek/offer\_wo\_header.csv

# MapR Client Installation ( on windows)

* Add the cldb server’s IP and name in etc/hosts file

The /etc/hosts file is present in C:\Windows\System32\drivers\etc

Add a mapping between the CLDB nodes in the cluster and the IP addresses of those nodes. For example, add the IP address 10.10.82.22 and CLDB node name centos22 on the Mac OSX where you installed the client:

Eg: the mapR client is to my laptop which has windows 7. The MapR cluster is a 2 node cluster 172.16.243.120 and 172.16.243.121. the cldb service is hosted on 172.16.243.120

* + - 1. ind-mhp1v29lnx11
* Install JAVA and set JAVA\_HOME correctly

This path is set in the environment variable.

Eg : in advanced setting of the computer,

JAVA HOME variable is set to

C:\Program Files\Java\jdk1.8.0\_45

This will not work, as spaces are not considered

Put the folder jdk1.8.0\_45 in c:/opt/

And change the env.bat present in \opt\mapr\conf\ as below

In env.bat file

set JAVA\_PATH=C:\opt\jdk1.8.0\_45

set JAVA\_HOME=%JAVA\_PATH%

set MAPR\_HOME=C:\opt\mapr

* Create a directory in c: as /opt/mapr

On command prompt,

mkdir c:\opt\mapr

* Create another environment variable called MAPR\_HOME and set as c:\opt\mapr
* Extract the mapR client packages from package.mapr.com and place it in c:\opt\mapr
* Restart the client m/c to take the values of environment variables

**RUN cmd.exe as administraor**

* Run the below :

Cldb node is ind-mhp1v29lnx11 and port is 7222

History server is on ind-mhp1v29lnx12

server\configure.bat -N MapRCluster  -c -C ind-mhp1v29lnx11:7222 -HS ind-mhp1v29lnx12

eg: this runs but run cmd.exe as administrator

C:\opt\mapr\server>configure.bat -N MapRCluster -c -C 172.16.243.120:7222 -HS 172.16.243.121

* Edit the core-site.xml file in

C:\opt\mapr\hadoop\hadoop-2.7.0\etc\hadoop\core-site.xml

<property>

<name>hadoop.spoofed.user.uid</name>

<value>{uid}</value>

</property>

<property>

<name>hadoop.spoofed.user.gid</name>

<value>{GID}</value>

</property>

<property>

<name>hadoop.spoofed.user.username</name>

<value>{id of user who has UID}</value>

</property>

Eg: on 2 node cluster of TIS

<configuration>

<property>

<name>hadoop.spoofed.user.uid</name>

<value>5000</value>

</property>

<property>

<name>hadoop.spoofed.user.gid</name>

<value>5000</value>

</property>

<property>

<name>hadoop.spoofed.user.username</name>

<value>mapr</value>

</property>

</configuration>

* Add the below in mapred-site.xml in C:\opt\mapr\hadoop\hadoop-2.7.0\etc\hadoop\mapred-site.xml ( set this to submit mapreduce jobs)

<property>

<name>mapreduce.app-submission.cross-platform</name>

<value>true</value>

</property>

* Test if the commands are running using client

The below command will list all the files inside the folder /Mastek in MaprFS

C:\opt\mapr\hadoop\hadoop-0.20.2\bin>hadoop fs -ls /Mastek

C:\opt\jdk1.8.0\_45

C:\opt\mapr

SLF4J: Class path contains multiple SLF4J bindings.

SLF4J: Found binding in [jar:file:/C:/opt/mapr/hadoop/hadoop-0.20.2/lib/slf4j-lo

g4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]

SLF4J: Found binding in [jar:file:/C:/opt/mapr/lib/slf4j-log4j12-1.7.10.jar!/org

/slf4j/impl/StaticLoggerBinder.class]

SLF4J: See http://www.slf4j.org/codes.html#multiple\_bindings for an explanation.

SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]

Found 4 items

drwxr-xr-x - uid\_5000 gid\_5000 1 2016-08-10 18:13 /Mastek/Marketing

drwxr-xr-x - uid\_5000 gid\_7000 2 2016-08-10 14:14 /Mastek/RnD

-rw-r--r-- 3 uid\_5000 gid\_5000 1764 2016-08-10 16:17 /Mastek/offer\_wo\_he

ader.csv

-rwxr-xr-x 3 uid\_5000 gid\_5000 210 2016-08-11 12:23 /Mastek/personal\_wo

\_header.csv