**Hadoop**

**Overview**

Hadoop is a high-performance distributed data storage and processing system.

Its two major subsystems are:

* HDFS for storage
* Mapreduce for parallel data processing.
* It can store any kind of data from any source, inexpensively and at very large scale, and it can do very sophisticated analysis of that data easily and quickly.
* Hadoop automatically detects and recovers from hardware, software and system failures.
* Hadoop provides scalable, reliable and fault tolerant services for data storage and analysis at very low cost.



**Hadoop Components**

**Management**

* Cloudera
* Oozie
* Chukwa

**Data Access**

* Hive
* Pig
* Avro

**Data Processing**

* Map Reduce

**Storage**

* Hbase
* HDFS

**Cloudera**

Cloudera. is a [Palo Alto](http://en.wikipedia.org/wiki/Palo_Alto)-based American enterprise software company that provides [Apache Hadoop](http://en.wikipedia.org/wiki/Apache_Hadoop)-based software, support and services, and training to data driven enterprises.

Cloudera's open-source Apache Hadoop distribution, CDH (Cloudera Distribution Including Apache Hadoop), targets enterprise-class deployments of that technology. Cloudera says that more than 50% of its engineering output is donated upstream to the various Apache-licensed open source projects (Apache Hive, Apache Avro, Apache HBase, and so on) that combine to form the Hadoop platform. Cloudera is also a sponsor of the [Apache Software Foundation](http://en.wikipedia.org/wiki/Apache_Software_Foundation). The preferred [demonym](http://en.wikipedia.org/wiki/Demonym) for an employee of Cloudera is "Clouderan.

**Oozie**

Oozie is a workflow scheduler system to manage [Hadoop](http://en.wikipedia.org/wiki/Apache_Hadoop) jobs. It is a server based Workflow Engine specialized in running workflow jobs with actions that run Hadoop Map/Reduce and Pig jobs. Oozie is a Java Web-Application that runs in a Java servlet-container. For the purposes of Oozie, a workflow is a collection of actions (i.e. Hadoop Map/Reduce jobs, Pig jobs) arranged in a control dependency DAG (Direct Acyclic Graph). "control dependency" from one action to another means that the second action can't run until the first action has completed. The workflow actions start jobs in remote systems (i.e. Hadoop, Pig). Upon action completion, the remote systems callback Oozie to notify the action completion, at this point Oozie proceeds to the next action in the workflow. Oozie workflows contain control flow nodes and action nodes. Control flow nodes define the beginning and the end of a workflow ( start , end and fail nodes) and provide a mechanism to control the workflow execution path ( decision , fork and join nodes).Action nodes are the mechanism by which a workflow triggers the execution of a computation/processing task. Oozie provides support for different types of actions: Hadoop map-reduce, Hadoop file system, Pig, SSH, HTTP, eMail and Oozie sub-workflow. Oozie can be extended to support additional type of actions.

**Chukwa**

Chukwa is a Hadoop subproject devoted to large-scale log collection and analysis. Chukwa is built on top of the Hadoop distributed filesystem (HDFS) and [MapReduce](http://wiki.apache.org/hadoop/MapReduce) framework and inherits Hadoop’s scalability and robustness. Chukwa also includes a ﬂexible and powerful toolkit for displaying monitoring and analyzing results, in order to make the best use of this collected data.

**Hive**

Hive is designed to enable easy data summarization, ad-hoc querying and analysis of large volumes of data. It provides a simple query language called Hive QL, which is based on SQL and which enables users familiar with SQL to do ad-hoc querying, summarization and data analysis easily. At the same time, Hive QL also allows traditional map/reduce programmers to be able to plug in their custom mappers and reducers to do more sophisticated analysis that may not be supported by the built-in capabilities of the language.

**Pig**

Pig is a data flow platform for writing Hadoop operations in a language called *Pig Latin.*It adds a layer of abstraction on top of Hadoop to simplify its use by giving a SQL-like interface to process data on Hadoop and thus help the programmer focus on business logic and help increase productivity. It supports a variety of data types and the use of user-defined functions (UDFs) to write custom operations in Java, Python and JavaScript. Due its simple interface,  support for doing complex operations such as joins and filters, Pig is popular for performing query operations in hadoop.

Difference between Hive and Pig

<http://www.larsgeorge.com/2009/10/hive-vs-pig.html>

**Avro**

Apache Avro is a data serialization system.

Avro provides:

* Rich data structures.
* A compact, fast, binary data format.
* A container file, to store persistent data.
* Remote procedure call (RPC).
* Simple integration with dynamic languages. Code generation is not required to read or write data files nor to use or implement RPC protocols. Code generation as an optional optimization, only worth implementing for statically typed languages.

**Hadoop Distributed File System, or HDFS**

HDFS is the storage system for a Hadoop cluster. When data arrives at the cluster, the HDFS software breaks it into pieces and distributes those pieces among the different servers participating in the cluster. Each server stores just a small fragment of the complete data set, and each piece of data is replicated on more than one server.

**Hbase**

**Distributed data processing framework called MapReduce**

Because Hadoop stores the entire dataset in small pieces across a collection of servers, analytical jobs can be distributed, in parallel, to each of the servers storing part of the data. Each server evaluates the question against its local fragment simultaneously and reports its results back for collation into a comprehensive answer. MapReduce is the plumbing that distributes the work and collects the results.

**Mahout**

Recommendation mining takes users' behavior and from that tries to find items users might like. Clustering takes e.g. text documents and groups them into groups of topically related documents. Classification learns from exisiting categorized documents what documents of a specific category look like and is able to assign unlabelled documents to the (hopefully) correct category. Frequent itemset mining takes a set of item groups (terms in a query session, shopping cart content) and identifies, which individual items usually appear together.

**History**

2003.2 First MapReduce library written at Google

2003.10 Google File System paper published

2004.12 Google MapReduce paper published

2005.7 Doug Cutting reports that Nutch now use new MapReduce implementation

2006.2 Hadoop code moves out of Nutch into new Lucene sub-project

2006.11 Google Bigtable paper published

2007.2 First HBase code drop from Mike Cafarella

2007.4 Yahoo! Running Hadoop on 1000-node cluster

2008.1 Hadoop made an Apache Top Level Project

**Why Hadoop**

**Store anything**: Hadoop stores data in its native format, exactly as it arrives at the cluster. Translating data on arrival so that it fits into a fixed data warehouse schema destroys information. Because Hadoop stores data without forcing that transformation, no information is lost. Downstream analyses run with no loss of fidelity. Of course it is always possible to digest, analyze and transform data, but Hadoop allows the data analyst to choose how and when to do that.

**Control costs**: Hadoop is open source software that runs on commodity hardware. That combination means that the cost per terabyte, for both storage and processing, is much lower than on older proprietary systems. Adding or removing storage capacity is simple. You can dedicate new hardware to a cluster incrementally, as required, and can retire nodes.

**Proven at scale**: You may not have petabytes of data that you need to analyze today. However, you can deploy Hadoop with confidence because companies like Facebook,Yahoo! and others run very large Hadoop instances managing enormous amounts of data. When you adopt a platform for data management and analysis, you are making a commitment that you will have to live with for years. The success of the biggest Web companies in the world demonstrates that Hadoop can grow as your business does.

**What is Hadoop used for**

* Searching/ Text mining
* Log processing
* Recommendation systems
* Business Intelligence/Data Warehousing
* Video and Image analysis
* Archiving
* Graph creation and analysis
* Pattern recognition
* Risk assessment
* Sentiment Analysis

**Characteristics**

**Very large files**: files that are hundreds of megabytes, gigabytes,or terabytes in size.

**Streaming data access:** HDFS is built around the idea that the most efficient data processing pattern is a write-once, read-many-times pattern,time to read the whole dataset is more important than the latency in reading the first record

**Commodity hardware:** Hadoop doesn’t require expensive, highly reliable hardware to run on. It’s designed to run on clusters of commodity hardware (commonly available hardware available from multiple vendors)

**HDFS Components**

A HDFS cluster has two types of node operating in a master-worker pattern:

* namenode (the master)
* number of datanodes (workers)



**HDFS Architecture**



**Blocks**

* Files in HDFS are broken into block-sized chunks, which are stored as independent units.
* Default is 64mb
* Blocks are replicated to different data nodes depending on the value of replication factor.
* Block placement algorithm is rack-aware

**Configuration Scripts**

**Default ports used**



**Web Interface**

The NameNode web interface will be available via HTTP on port 50070 [http://master:50070](http://master:50070/)



![](data:None;base64,)

Hadoop is not a good fit for…

\_ Low-latency data access - Applications that require low-latency access to data,

in the tens of milliseconds range, will not work well with HDFS.

\_ Lots of small files - Since the namenode holds filesystem metadata in memory,

the limit to the number of files in a filesystem is governed by the amount of

memory on the namenode.

\_ Multiple writers, arbitrary file modifications - Files in HDFS may be written to

by a single writer. Writes are always made at the end of the file. There is no

support for multiple writers, or for modifications at arbitrary offsets in the file

Mapreduce Basics

What is Mapreduce

\_ MapReduce is a programming model designed for processing large volumes of data in

parallel by dividing the work into a set of independent tasks.

Map Phase

\_ The first phase of a MapReduce program is called mapping. A list of data elements are

provided, one at a time, to a function called the Mapper, which transforms each

element individually to an output data element.



Reduce Phase

Reducing lets you aggregate values together. A reducer function receives an iterator of

input values from an input list. It then combines these values together, returning a

single output value.



**Hadoop Single Node Setup**

* **Pre-requisites**

**Java 1.6.X**

Java needs to be installed in your node, in order to run Hadoop as it is a java based framework. We can check the Java installation of node by,

global @VM-GMM-t62b:~$ java –version

java version "1.6.0\_23"

Java(TM) SE Runtime Environment (build 1.6.0\_23-b05)

Java HotSpot(TM) Server VM (build 19.0-b09, mixed mode)

If it is not giving an intended output, we need to install Java by running,

global @VM-GMM-t62b:~$ sudo apt-get install sun-java6-jdk

If java is already installed in your machine and need to find out location by using command

global@VM-GMM-t62A:/etc/alternatives$ which java

/usr/bin/java

global@VM-GMM-t62A:/usr/bin$ ls -l java

lrwxrwxrwx 1 root root 22 Jul 19 10:09 java -> /etc/alternatives/java

global@VM-GMM-t62A:/usr/bin$ cd /etc/alternatives/

global@VM-GMM-t62A:/etc/alternatives$ ls -l java

lrwxrwxrwx 1 root root 46 Jul 19 10:09 java -> /usr/lib/jvm/java-6-openjdk-amd64/jre/bin/java

**Create a Hadoop User**

At this step we create a user account dedicated for Hadoop installation. This not a "must do" step, but recommended for security reasons and ease of managing the nodes. So we create a group called 'hadoop' and add a new user to the group called 'hpuser'(the names can be of our choice) with the following commands.

global @VM-GMM-t62b:~$ sudo addgroup hadoop

global @VM-GMM-t62b:~$ sudo adduser --ingroup hadoop myhduser

global @VM-GMM-t62b:~$ su myhduser

myhduser @VM-GMM-t62b:~$

**Enable SSH Access**

Hadoop requires SSH(Secure Shell) access to the machines it uses as nodes. This is to create a secured channel to exchange data. So even in single node the localhost need SSH access for the hpuser in order to exchange data for Hadoop operations. Refer this documentation if you need more details on SSH.

We can install SSH with following command.

myhduser @VM-GMM-t62b:~$ sudo apt-get install ssh

Now let's try to SSH the localhost without a pass-phrase.

myhduser @VM-GMM-t62b:~$ ssh localhost

Linux VM-GMM-t62b 2.6.32-33-generic #72-Ubuntu SMP Fri Jul 29 21:08:37 UTC 2011 i686 GNU/Linux

Ubuntu 10.04.4 LTS

Welcome to Ubuntu!

................................................

If it does not give something similar to the above, we have to enable SSH access to the localhost as following.

Generate a RSA key pair without a password. (We do not use a password only because then it will prompt us to provide password in each time Hadoop communicate with the node.

myhduser @VM-GMM-t62b:~$ ssh-keygen -t rsa -P ""

Then we need to concatenate the generated public key in the authorized keys list of the localhost. It is done as follows. Then make sure 'ssh localhost' is successful.

myhduser @VM-GMM-t62b:~$ cat $HOME/.ssh/id\_rsa.pub >> $HOME/.ssh/authorized\_keys

Now we are ready to move onto Hadoop. Use the latest stable release from [Hadoop site](http://hadoop.apache.org/releases.html" \l "Download) and I used hadoop-1.1.2.

$ cd /home/myhduser

$ sudo tar xzf hadoop-1.1.2.tar.gz

Now on we will work as myhduser, with the last command.

Command for editing file.

$ sudo vi filename.txt

If it display file name is read only then use following command for save file

Press ‘ESC’ then ‘:’

:w !sudo tee %

Press enter then

:q!

Or

$ sudo gedit filename.txt

* **Hadoop Configurations**

**(Check proper path for java\_home home in .bashrc)(find exact locatoin)**

Set JAVA\_HOME and HADOOP\_HOME. For that open the same file ($HOME/.bashrc) and add the following two lines at the end.

export JAVA\_HOME=/usr/lib/jvm/java-6-openjdk-amd64

export HADOOP\_HOME=/home/myhduser/hadoop-1.1.2

export PATH=$PATH:$HADOOP\_HOME/bin

**Disable IPv6**

As there will not be any practical need to go for IPv6 addressing inside Hadoop cluster, we are disabling this to be less error-prone. Open the file HADOOP\_HOME/conf/hadoop-env.sh and add the following line.

export HADOOP\_OPTS=-Djava.net.preferIPv4Stack=true

**Set Paths and Configuration**

In the same HADOOP\_HOME/conf/hadoop-env.sh file add the JAVA\_HOME too.

export JAVA\_HOME=/usr/lib/jvm/java-6-openjdk-amd64

Then we need create a directory to be used as HDFS (Hadoop Distributed File System).Make directory at a place of your choice and make sure the owner of the directory is 'myhduser'. I ll refer this as 'tmp'. We can do it by,

I personally prefer to create tmp directory under /home/myhduser/hadoop-1.1.2

myhduser @VM-GMM-t62b:~$ sudo mkdir tmp

myhduser @VM-GMM-t62b:~$ sudo chown myhduser:hadoop /home/myhduser/hadoop-1.1.2/tmp

Now we add the following property segments in the relevant file inside the <configuration>......</configuration> tags.

/home/myhduser/hadoop-1.1.2/conf/core-site.xml

<property>

<name>hadoop.tmp.dir</name>

<value>/home/myhduser/hadoop-1.1.2/tmp</value>

<description>Location for HDFS.</description>

</property>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:54310</value>

<description>The name of the default file system. A URI whose

scheme and authority determine the FileSystem implementation. </description>

</property>

/home/myhduser/hadoop-1.1.2/conf/mapred-site.xml

<property>

<name>mapred.job.tracker</name>

<value>localhost:54311</value>

<description>The host and port that the MapReduce job tracker runs

at. </description>

</property>

/home/myhduser/hadoop-1.1.2/conf/hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default number of block replications.

</description>

</property>

The above value needs to be decided on the priority on speed, space and fault tolerance factors.

**Format HDFS**

This operation is needed every time we create a new Hadoop cluster. If we do this operation on a running cluster all the data will be lost.What this basically do is creating the Hadoop Distributed File System over the local file system of the cluster.

myhduser @VM-GMM-t62b:~$/home/myhduser/hadoop-1.1.2/bin/hadoop  namenode –format

12/09/20 14:39:56 INFO

namenode.NameNode: STARTUP\_MSG:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

STARTUP\_MSG: Starting NameNode

STARTUP\_MSG: host = pushpalanka-laptop/127.0.1.1

STARTUP\_MSG: args = [-format]

STARTUP\_MSG: version = 1.0.3

STARTUP\_MSG: build = https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.0 -r

1335192; compiled by 'hortonfo' on Tue May 8 20:31:25 UTC 2012

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

...

12/09/20 14:39:57 INFOnamenode.NameNode: SHUTDOWN\_MSG:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

SHUTDOWN\_MSG: Shutting down NameNode at pushpalanka-laptop/127.0.1.1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

If the output is something as above HDFS is formatted successfully. Now we are almost done and ready to see the action.

**Running the Single Node Cluster**

myhduser @VM-GMM-t62b:~/hadoop-1.1.2/bin$ ./start-all.sh

starting namenode, logging to /home/myhpuser/hadoop-1.1.2/libexec/../logs/hadoop-myhduser-namenode- VM-GMM-t62b.out

localhost: starting datanode, logging to /home/myhpuser/hadoop-1.1.2/libexec/../logs/hadoop-myhduser-datanode- VM-GMM-t62b.out

localhost: starting secondary namenode, logging to /home/myhduser/hadoop-1.1.2/libexec/../logs/hadoop-myhduser-secondary namenode- VM-GMM-t62b.out

starting jobtracker, logging to /home/myhpuser/hadoop-1.1.2/libexec/../logs/hadoop-myhduser-jobtracker- VM-GMM-t62b.out

localhost: starting tasktracker, logging to /home/myhduser/hadoop-1.1.2/libexec/../logs/hadoop-myhduser-tasktracker- VM-GMM-t62b.out

which starts several nodes and trackers. We can observe these Hadoop processes using jps tool from Java.

myhduser@VM-GMM-t62A:~/hadoop-1.1.2/bin$ jps

18217 NameNode

19085 Jps

18688 SecondaryNameNode

18778 JobTracker

18446 DataNode

19007 TaskTracker

Also we can check the ports Hadoop is configured to listen on,

hpuser@ VM-GMM-t62A:~/hadoop-1.1.2/bin$ sudo netstat -plten | grep java

[sudo] password for myhduser:

tcp 0 0 0.0.0.0:33793 0.0.0.0:\* LISTEN 1001 175415 8164/java

....

tcp 0 0 0.0.0.0:60506 0.0.0.0:\* LISTEN 1001 174566 7767/java

tcp 0 0 0.0.0.0:50075 0.0.0.0:\* LISTEN 1001 176269 7962/java

There are several web interfaces available to observe inside behavior, job completion, memory consumption etc. as follows.

http://localhost:50070/– web UI of the NameNode

http://localhost:50030/– web UI of the JobTracker

http://localhost:50060/– web UI of the TaskTracker

At the moment this will not show any information as we have not put any job for execution to observe the progress.

We can stop the cluster at any time with the following command.

hpuser@ VM-GMM-t62A:~/hadoop-1.1.2/bin$ ./stop-all.sh

stopping jobtracker

localhost: stopping tasktracker

stopping namenode

localhost: stopping datanode

localhost: stopping secondarynamenode

Now we have completed setting up a single node cluster with Apache Hadoop.

**Multi node set up**

**Pre-Requisites**

Minimum two machines are required for configure Hadoop Multi Node cluster. First check all machines are in same network by executing below command.

myhduser @VM-GMM-t62b:~$ifconfig 172.16.6.152

It’s better to create same username in same location to all slave machines and Setup all same configurations to all salve machines as suggested for single node. It’s better to copy from one configured machine “/home/myhduser/hadoop-1.1.2” folder to all slave Machines in same location by executing in command in Terminal.

myhduser @VM-GMM-t62b:~$Sudo scp -r /home/myhduser/hadoop-1.1.2 [myhduser@172.16.6.152:/home/myhduser](mailto:myhduser@172.16.6.152:/home/myhduser)

scp [[user@]from-host:]source-file [[user@]to-host:][destination-file]

Then we need to add these in '/etc/hosts' file of each machine as follows.

172.16 . 6.67 master

172.16 . 6.152 slave

Note: The addition of more slaves should be updated here in each machine using unique names for slaves (eg: slave01, slave02).

**Enable SSH Access**

We did this step in single node set up for each machine to create a secured channel between the localhost and myhduser. Now we need to make the myhduser in master, is capable of connecting to the myhduser account in slave via a password-less SSH login. We can do this by adding the public SSH key of myhduser in master to the authorized\_keys of myhduser in slave. Following command from myhduser at master will do the work .

myhduser@master :~ $ ssh - copy - id - i $HOME /. ssh / id\_rsa . pub myhduser@slave

Note: If more slaves are present this needs to be repeated for them. This will prompt for the password of myhduser of slave and once given we are done. To test we can try to connect from master to master and master to slave as per our requirement as follows.

myhduser@master :~ $ ssh slave

The authenticity of host 'slave (172.16.6.152)' can 't be established.

RSA key fingerprint is ............................................................

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added ' slave (172.16.6.152) ' (RSA) to the list of known hosts.

myhduser@slave' s password :

Welcome to Ubuntu 11.10 ( GNU / Linux 3.0 . 0 - 12 - generic i686 )

....................................

If a similar kind of output is given for 'ssh master' we can proceed to next steps.

**Hadoop Configurations**

We have to do the following modifications in the configuration files.

In master machine

1. conf/masters

master

This file is defining in which nodes are the secondary NameNodes are starting, when bin/start-dfs.sh is run. The duty of secondary NameNode is to merge the edit logs periodically and keeping the edit log size within a limit.

2. conf/slaves

master

slave

This file lists the hosts that act as slaves processing and storing data. As we are just having two nodes we are using the storage of master too.

Note: If more slaves are present those should be listed in this file of all the machines.

**In all machines**

1. conf/core-site.xml

<property>

<name> fs.default.name </name>

<value> hdfs://master:54310 </value>

<description> ..... </description>

</property>

We are changing the 'localhost' to master as we can now specifically mention to use master as NameNode.

2. conf/mapred-site.xml

<property>

<name> mapred.job.tracker </name>

<value> master:54311 </value>

<description> The host and port that the MapReduce job tracker runs

at. </description>

</property>

We are changing the 'localhost' to master as we can now specifically mention to use master as JobTracker.

3. conf/hdfs-site.xml

<property>

<name> dfs.replication </name>

<value> 2 </value>

<description> Default number of block replications.

</description>

</property>

It is recommended to keep the replication factor not above the number of nodes. We are here setting it to 2.

**Format HDFS from the NameNode**

Initially we need to format HDFS as we did in the single node set up too.

myhduser@master :~/ hadoop - 1.1. 2 $ bin / hadoop namenode - format

........................................

12 / 11 / 02 23 : 25 : 54 INFO common . Storage : Storage directory /home/myhduser/tmp/dfs/ name has been successfully formatted .

12 / 11 / 02 23 : 25 : 54 INFO namenode . NameNode : SHUTDOWN\_MSG :

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

If the output for the command ended up as above we are done with formatting the file system and ready to run the cluster.

Running the Multi Node Cluster

Starting the cluster is done in an order that first starts the HDFS daemons(NameNode, Datanode) and then the Map-reduce daemons(JobTracker, TaskTracker).

Also it is worth to notice that we can observe what is going on in slaves when we run commands in master from the logs directory inside the HADOOP\_HOME of the slaves.

1. Start HDFS daemons - bin/start-dfs.sh in master

myhduser@master :~/ hadoop - 1.1 .2 $bin/start - all.sh

starting namenode , logging to ../ bin /../ logs / hadoop - myhduser - namenode - master . out

slave : Ubuntu

slave : starting datanode , logging to .../ bin /../ logs / hadoop - myhduser - datanode - slave . out

master : starting datanode , logging to ..../ bin /../ logs / hadoop - myhduser - datanode - master . out

master : starting secondarynamenode , logging to .../ bin /../ logs / hadoop - myhduser - secondarynamenode - master . out

This will get the HDFS up with NameNode and DataNodes listed in conf/slaves.

At this moment, java processes running on master and slaves will be as follows.

myhduser@master :~/ hadoop - 1.1.2 $ jps

18217 NameNode

19085 Jps

18688 SecondaryNameNode

18778 JobTracker

18446 DataNode

19007 TaskTracke

myhduser@slave :~/ hadoop - 1.1.2 $ jps

1390 Jps

1142 DataNode

1335 TaskTracker

60 Hadoop Interview Question

http://www.pappupass.com/Hadoop\_Interview\_Question.pdf

<http://hadoop.apache.org/common/docs/r0.20.2/mapred_tutorial.html>

<http://hadoop.apache.org/hdfs/>

for changing host name

?sudo gedit /etc/hostname

type master

check your host name now

?hostname

Not changed(need to restart host name server)

?Sudo service hostname start

Now check the host name

?hostname

Name would be changed as master

Now change the host configuration

?sudo gedit /etc/hosts

To get ipaddress type

?ifconfig

File will be look like for master

![](data:None;base64,)

Check ssh

? which ssh

It will display the path

Or you can install ssh

?Sudo apt-get install sshd

![](data:None;base64,)inter ip address of slave in etc/host

Cntrl+h (show hidden files)

If you want to copy file from one machine to other machin

? scp –r hadoop-1,2,1 username@machiname/path(path is where we to copy)

Sudo scp -r /home/myhduser/hadoop-1.1.2 myhduser@172.16.6.152:/home/myhduser

**Setting Pig on Hodoop**

After confguring Hadoop, dwonload pig file([pig-0.11.1.tar.gz](http://apache.mesi.com.ar/pig/stable/pig-0.11.1.tar.gz)

) from [http://apache.](http://apache.mesi.com.ar/pig/stable/)

`[.com.ar/pig/stable/](http://apache.mesi.com.ar/pig/stable/)

**tar xzf pig-0.11.1.tar.gz**

It’s convenient to add Pig’s binary directory to your command-line path. For example:

% **export PIG\_INSTALL=**

% **export PATH=$PATH:$PIG\_INSTALL/bin**

### Sqoop installation tutorial

Sqoop is a tool which is used to import / export data from RDBMS to HDFS

It can be downloaded from the apache website. As of writing this post the Sqoop is in incubation project with apache , but it would come as full project in the near future.

Sqoop is a client tool , you are not required to install it to all nodes of Cluster. The best practice is to just install it on client ( or edge node of the cluster) . The data transfer is direct between Cluster and Database , incase you are worried for traffic between machine where you install Sqoop and Database.

Installation steps

You can download the latest version of sqoop from apache website  
<http://sqoop.apache.org/>

The installation is fairly simple to start off for development purpose with Sqoop

Download the latest sqoop binary file

Extract it in some folder

Specify the SQOOP\_HOME and add Sqoop path variable so that we can directly run the sqoop commands

For example i downloaded sqoop in following directory and my environment variables look like this  
export SQOOP\_HOME="/home/hadoop/software/sqoop-1.4.3"

export PATH=$PATH:$SQOOP\_HOME/bin

Sqoop can be connected to various types of databases .

For example it can talk to mysql , Oracle , Postgress databases. It uses JDBC to connect to them. JDBC driver for each of databases is needed by sqoop to connect to them.

JDBC driver jar for each of the database can be downloaded from net. For example mysql jar is present at link below

<http://dev.mysql.com/downloads/connector/j/>

Download the mysql j connector jar and store in lib directory present in sqoop home folder.

Thats it.

Just test your installation by typing

$ sqoop help

You should see the list of commands with there use in sqoop

Happy sqooping :)

<http://localhost:50070/> – web UI of the NameNode daemon

* <http://localhost:50030/> – web UI of the JobTracker daemon
* <http://localhost:50060/> – web UI of the TaskTracker daemon

**Configuring Flume**

**download apache flume from** [**http://apache.tradebit.com/pub/flume/1.4.0/apache-flume-1.4.0-bin.tar.gz**](http://apache.tradebit.com/pub/flume/1.4.0/apache-flume-1.4.0-bin.tar.gz)

**and extract it in your favorite directory**

**setup the path**

**export PIG\_FLUME=/home/myhduser/hadoop-1.2.1/apache-flume-1.4.0-bin**

**export PATH=$PATH:$PIG\_INSTALL/bin**

**$>flume-ng**

**configure the source and sink location under apache-flume-1.4.0-bin/conf/flume.conf**

**note: if you see flume-conf.properties.tamplate then rename to flume.conf name**

**Add following details in flume.conf file to upload system log file into HDFS**

**# Define a memory channel on agent called memory-channel.**

**agentsyslog.channels.memory-channel.type = memory**

**# Define a source on agent and connect to channel memory-channel.**

**agentsyslog.sources.tail-source.type = exec**

**agentsyslog.sources.tail-source.command = cat /var/log/bootstrap.log**

**agentsyslog.sources.tail-source.channels = memory-channel**

**# Define a sink that outputs to logger.**

**agentsyslog.sinks.log-sink.channel = memory-channel**

**agentsyslog.sinks.log-sink.type = logger**

**# Define a sink that outputs to hdfs.**

**agentsyslog.sinks.hdfs-sink.channel = memory-channel**

**agentsyslog.sinks.hdfs-sink.type = hdfs**

**agentsyslog.sinks.hdfs-sink.hdfs.path = hdfs://localhost:54310/dataByFlume/system.log/**

**agentsyslog.sinks.hdfs-sink.hdfs.fileType = DataStream**

**# Finally, activate.**

**agentsyslog.channels = memory-channel**

**agentsyslog.sources = tail-source**

**agentsyslog.sinks = log-sink hdfs-sink**

**--------------------------------------------------ennd-----------------------------------------------**

**agentsyslog.sources.tail-source.command = tail -F /var/log/bootstrap.log(this is the location from where file will be upload into HDFS )**

**agentsyslog.sinks.hdfs-sink.hdfs.path = hdfs://localhost:54310/dataByFlume/system.log/(this the location where the files will be reside in HDFS)**

**Below find command to upload file from source to sink(destination means HDFS)**

**bin/flume-ng agent --conf ./conf/ -f conf/flume.conf -Dflume.root.logger=DEBUG,console -n agentsyslog**

**once you will execute about commend in you terminal. You will able to see uploaded log files in HDFS**

**via hadoop ls command or navigating below namenode URL**

<http://localhost:50070/> – web UI of the NameNode daemon

**Note that at the end of cammand “agentsyslog” use thbin/flume-ng agent --conf ./conf/ -f conf/flume.conf -Dflume.root.logger=DEBUG,console -n agentsyslog e name exactly same in flume.conf file. First word of key value. So flume.conf may contain many source configration details with unique key name.**

Configure Ozee

link to apache oozie quick start  <http://oozie.apache.org/docs/3.3.1/DG_QuickStart.html>  
and apache hadoop [http://hadoop.apache.org/docs/r0.23.0](http://hadoop.apache.org/docs/r0.23.0/)  
My testing environment

* 4 node cluster ( 1 master , 3slaves )
* Apache Hadoop 0.23.0
* Apache oozie 3.3.1
* java 1.6.0\_26
* Maven 3.0.4

Oozie server installation

* Download oozie 3.3.1.tar.gz  from the nearest mirror site [apache/oozie/3.3.1](http://mirror.nus.edu.sg/apache/oozie/3.3.1/) ( i downloaded from nus.edu.sg mirror)
* Unpack the oozie-3.3.1 tar.gz file under some /home/srikanth
* The following two properties are required in Hadoop core-site.xml:

<!-- OOZIE -->

<property>

<name>hadoop.proxyuser.[OOZIE\_SERVER\_USER].hosts</name>

<value>[OOZIE\_SERVER\_HOSTNAME]</value>

</property>

<property>

<name>hadoop.proxyuser.[OOZIE\_SERVER\_USER].groups</name>

<value>[USER\_GROUPS\_THAT\_ALLOW\_IMPERSONATION]</value>

</property>

[ I replaced [OOZIE\_SERVER\_USER] with my srikanth and [OOZIE\_SERVER\_HOSTNAME] with \* [USER\_GROUPS\_THAT\_ALLOW\_IMPERSONATION] with \*  
Replace the capital letter sections with specific values and then restart Hadoop.  
*(For this blog purpose is set /home/srikanth/oozie-3.3.1 as my $OOZIE\_HOME)*

* Download ExtJS (e.g. [extjs-2.2.zip](http://extjs.com/deploy/ext-2.2.zip)). Do not unzip it right away. At this point, the instructions in the Quick Start Guide ( a.k.a. QSG) jump to calling bin/setup.sh, which will not work. Also, the path to setup.sh is incorrectly specified in the QSG.
* Building a Distro
  + *> cd $OOZIE\_HOME*
  + *> ./bin/mkdistro.sh -DskipTests*  
           If successful, you will see the following print out:

*Oozie distro created, DATE[2013.01.02-01:38:05GMT] VC-REV[unavailable], available at [/home/srikanth/oozie-3.3.1/distro/target*]

* In $OOZIE\_HOME a directory called  \*hadooplibs/\* directory will be created containing the Hadoop JARs for the versions of Hadoop that the Oozie distribution supports.
* Create a **libext/** directory in the directory where Oozie was expanded.
* If using a version of Hadoop bundled in Oozie **hadooplibs/** , copy the corresponding Hadoop JARs from **hadooplibs/** to the **libext/** directory.
* If using a different version of Hadoop, copy the required Hadoop JARs from such version in the **libext/** directory. ( since hadoop 0.23.0 is not in the versions given in **hadooplibs/**, i copied the required jars from hadoop-0.23.0 to **libext**/. )
  + *Note :  I did the following steps to copy the hadoop jars into libext/ directory. Hope this will give some idea what jars to select and copy*  
                     > cd $HADOOP\_HOME/lib  
                         > cp \*.jar ../../oozie-3.3.1/libext/  
                         > cd ..   
                         > cp \*.jar ../../oozie-3.3.1/libext/  
                         > cp hadoop-mapreduce-tools-0.23.0.jar hadoop-mapreduce-tools-0.23.0-sources.jar hadoop-mapreduce-test-0.23.0.jar hadoop-mapreduce-test-0.23.0-sources.jar hadoop-mapreduce-examples-0.23.0.jar hadoop-mapreduce-examples-0.23.0-sources.jar hadoop-mapreduce-0.23.0.jar hadoop-mapreduce-0.23.0-sources.jar hadoop-0.23.0-streaming.jar hadoop-0.23.0-gridmix.jar /home/srikmvm/oozie-3.3.1/libext/
* Building with ExtJS library for Oozie Web Console
  + Copy the extjs-2.2.zip over to $OOZIE\_HOME/webapp/src/main/webapp/
  + Unzip it
* Run oozie-setup.sh to create an oozie.war file
  + *> cd $OOZIE\_HOME*

*> ./distro/target/oozie-3.3.1-distro/oozie-3.3.1/bin/oozie-setup.sh -hadoop 0.23.0 /home/srikanth/hadoop-0.23.0 -extjs $OOZIE\_HOME/webapp/src/main/webapp/ext-2.2.zip*

* + if the command is successfully run ,
  + *New Oozie WAR file will be created by injecting ‘Hadoop JARs, ExtJS library’ at /home/srikanth/oozie-3.3.1/distro/target/oozie-3.3.1-distro/oozie-3.3.1/oozie-server/webapps/oozie.war*

*INFO: Oozie is ready to be started*

* Copy the newly minted oozie.war file to your Tomcat deployment directory

*> cd $OOZIE\_HOME*

*> cp ./distro/target/oozie-3.3.1-distro/oozie-3.3.1/oozie-server/webapps/oozie.war   ./webapp/src/main/webapp/oozie.war*

* Make a configuration change before starting the Oozie Tomcat App
  + *> cd $OOZIE\_HOME*
  + *> vi ./distro/target/oozie-3.3.1-distro/oozie-3.3.1/conf/oozie-site.xml*Change the property below to true as shown. It is originally false, which causes startup to fail.

<property>

        <name>oozie.service.JPAService.create.db.schema</name>

        <value>true</value>

        <description>

            Creates Oozie DB.

            If set to true, it creates the DB schema if it does not exist. If the DB schema exists is a NOP.

            If set to false, it does not create the DB schema. If the DB schema does not exist it fails start up.

        </description>

    </property>

* Run the following command

*> cd $OOZIE\_HOME*

*> ./distro/target/oozie-3.3.1-distro/oozie-3.3.1/bin/ooziedb.sh create -sqlfile oozie.sql -run*

            if the command is successfully run, we can the following

DONE

Check DB schema does not exist

DONE

Check OOZIE\_SYS table does not exist

DONE

Create SQL schema

DONE

DONE

Create OOZIE\_SYS table

DONE

Oozie DB has been created for Oozie version '3.3.1'

* Start the Oozie Server

> *cd $OOZIE\_HOME*

*> ./distro/target/oozie-3.3.1-distro/oozie-3.3.1/bin/oozie-start.sh*

* To start Oozie as a foreground process run:

 > cd $OOZIE\_HOME

> ./distro/target/oozie-3.3.1-distro/oozie-3.3.1/bin/oozie-run.sh

Check the Oozie log file  *./distro/target/oozie-3.3.1-distro/oozie-3.3.1/logs/oozie.log* to ensure Oozie started properly.  
  
Using the Oozie command line tool check the status of Oozie:  
$ *./distro/target/oozie-3.3.1-distro/oozie-3.3.1/bin/oozie admin -oozie http://[ip address of namenode]:11000/oozie -status*

* Using a browser go to the Oozie web console, Oozie status should be **NORMAL**

Reference

<http://srikanthayalasomayajulu.blogspot.sg/2013/02/apache-oozie-331-installation-on-apache.html>

<http://practicalcloudcomputing.com/post/26337621577/installing-and-running-apache-oozie-3-2-x-and-possibly?543b50f0>