1. Bridging
2. ***MYsqld for hive and ntpd for network***
3. Ip setting
4. Proxy setting
5. Hostname change in /etc/hosts and /etc/sysconfig/networks
6. Disable selinux in vi /etc/sysconfig/selinux

service iptables save  
service iptables stop  
chkconfig iptables off

service ip6tables save  
service ip6tables stop  
chkconfig ip6tables off

1. Restart server and check ping
2. cd /etc/yum.repos.d

http\_proxy=http://10.10.87.131:3128/ wget --proxy-user=test --proxy-password=opus@123 <http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.4.3.38/ambari.repo>

http\_proxy=http://10.10.87.131:3128/ wget --proxy-user=test --proxy-password=opus@123 <http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.5.1/ambari.repo>

http\_proxy=http://10.10.87.131:3128/ wget --proxy-user=test --proxy-password=opus@123 <http://public-repo-1.hortonworks.com/ARTIFACTS/jdk-6u31-linux-x64.bin>

1. Download repository from <http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.4.3.38/ambari.repo>
2. save it on /etc/yum.repos.d
3. install ambari server by yum install -y ambari-server
4. set up ambari server by  ambari-server setup
5. install ambari agent by  yum install -y ambari-agent
6. ambari-server start
7. Configure the ambari-agent

vi /etc/ambari-agent/conf/ambari.ini

[server]

hostname=localhost

1. ambari-agent start
2. start ntpd service by service ntpd start

pig setting for hive

/usr/bin/pig file:

Replace to: exec /usr/lib/pig/bin/pig -useHCatalog “$@”

-----------------------------------------------------------------------------------------------------------------------------------------

1)set port numer

           export HIVE\_PORT=10000

2) Check which services is listening

     sudo lsof -i -P | grep -i "listen"

3)if there is process relevant to port 10000 kill it

     kill -9 pid

4) Start hive server

         $HIVE\_HOME/bin --service hiveserver

If it not work go to step 2 and start server again

Hadoop Configuration: /etc/hadoop/conf

Hadoop Log : /var/log/hadoop/hdfs

Ambari log : /var/log/ambari-server

Error :

**[root@localhost RPMS]# yum update**  
rpmdb: Thread/process 7977/139764163434432 failed: Thread died in Berkeley DB library  
error: db3 error(-30974) from dbenv->failchk: DB\_RUNRECOVERY: Fatal error, run database recovery  
error: cannot open Packages index using db3 -  (-30974)  
error: cannot open Packages database in /var/lib/rpm  
CRITICAL:yum.main:

Error: rpmdb open failed

Solution :

Run this command:

**[root@localhost RPMS]# rm /var/lib/rpm/\_\_db\***

Output:

**rm: remove regular file `/var/lib/rpm/\_\_db.001′? y**  
**rm: remove regular file `/var/lib/rpm/\_\_db.002′? y**  
**rm: remove regular file `/var/lib/rpm/\_\_db.003′? y**  
**rm: remove regular file `/var/lib/rpm/\_\_db.004′? y**

Now  try  the  Update  again :

vi /var/run/hadoop/hdfs/hadoop-hdfs-datanode.pid to clear the pid from agent

Clear cache:

sudo sh -c "sync; echo 3 > /proc/sys/vm/drop\_caches"

import by query

sqoop import --connect "jdbc:sqlserver:// 74.208.67.53:1433;username=sqoop;password=gYY66YfX;database=EDW" --query 'select top 100 \* from dbo.ATMTxnDetails where $CONDITIONS' --split-by Settlementdate  --hive-table atm100 --target-dir /user/root/atmtxn100 -m 1 --hive-import

sqoop import --connect "jdbc:sqlserver:// 74.208.67.53:1433;username=sqoop;password="gYY66YfX";database=EDW" --table ATMTxnDetails   --split-by settlementdate --target-dir /user/root/ATMTxnDetails\_settlement  -m 10 --hive-import

sqoop import --connect "jdbc:sqlserver:// 74.208.67.53:1433;username=sqoop;password="gYY66YfX";database=EDW" --query 'select top 100 \* from  dbo.coop\_terminals where $CONDITIONS'  --split-by terminalid --target-dir /user/root/ATMTxnDetails\_test1  -m 10 --hive-import

sqoop import --connect "jdbc:sqlserver:// 74.208.67.53:1433;username=sqoop;password="gYY66YfX";database=EDW" --table ATMTxnDetails  --incremental lastmodified --check-column SettlementDate --last-value "2014-03-27 00:00:00.0" --target-dir /user/root/ATMTxnDetails\_inc  -m 10 --hive-import

sqoop import –connect

sqoop import --connect "jdbc:sqlserver:// 100.67.20.29:1433;username=test;password=test;database=EDW" --query ' select top 100 \* from dbo.coop\_terminals where $CONDITIONS' --split-by Atminternalid   --target-dir /user/mongodbtw/coop1 -m 1 --hive-import

Change Datatype in Hive

ALTER TABLE atmtxn CHANGE settlementdate settlementdate date AFTER terminalsequencenumber;

start postgresql

/etc/init.d/postgresql start

Change url of name node UI

sudo vi /etc/hadoop/conf/core-site.xml

And change this line:

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

to

<value>hdfs://**[your IP]**:54310</value>

127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4

::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

10.10.8.139 hadoopsrv.poc.com

10.10.26.27 agent1.poc.com

10.10.26.45 agent2.poc.com

~

sudo -u hdfs hadoop dfsadmin –refreshNodes

sudo -u hdfs hadoop dfsadmin –report

kill application in yarn:

yarn application -kill application\_1396447312343\_0004

start ganglia

# service gmond restart

# service gmetad restart

You can also edit the firewall configuration in /etc/sysconfig/iptables and add a line with the port you need to open.  
i.e. below the rule for SSH:  
-A INPUT -p tcp -m state --state NEW -m tcp --dport 45454 -j ACCEPT  
This will add the port to the firewall whenever you reboot the host or restart the firewall.

sudo -u hdfs hadoop dfsadmin –report

sudo -u hdfs hadoop dfsadmin –refreshNodes

Hadoop dfsadmin -safemode leave

2--> then check id the safemode s off or not.

    hadoop dfsadmin -safemode get

3--> if you will get the output safe is on, then check the name space ID for DD and NN. I had faced similar kind off issue, then I found Name space ID of NN and DD are not same..

in that case you need to format the NN, caution: formatting the namenode will result into creation os new HDFS and privious Data will be lost.

Find disk space

df -T –h

/var/run/postmaster.PORT.pid

/var/lib/pgsql/pgstartup.log

ambari-server reset

python /usr/lib/python2.6/site-packages/ambari\_agent/HostCleanup.py --silent --skip=users

**5.7. Disable PackageKit**

On the RHEL/CentOS installation host, open /etc/yum/pluginconf.d/refresh-packagekit.conf with a text editor and make this change:

enabled=0

u17556398.onlinehome-server.com

python yarn-utils.py -c 32 -m 64 -d 2 -k True

sudo -u hdfs hadoop fs -mkdir /user/admin

sudo -u hdfs hadoop fs -chown root:root  /apps/hive/warehouse/terminaldetails

## HADOOP FS SHELL COMMANDS EXAMPLES - TUTORIALS

Hadoop file system (fs) shell commands are used to perform various file operations like copying file, changing permissions, viewing the contents of the file, changing ownership of files, creating directories etc.   
  
The syntax of fs shell command is

hadoop fs <args>

All the fs shell commands takes the path URI as arguments. The format of URI is sheme://authority/path. The scheme and authority are optional. For hadoop the scheme is hdfs and for local file system the scheme is file. IF you do not specify a scheme, the default scheme is taken from the configuration file. You can also specify the directories in hdfs along with the URI as hdfs://namenodehost/dir1/dir2 or simple /dir1/dir2.   
  
The hadoop fs commands are almost similar to the unix commands. Let see each of the fs shell commands in detail with examples: 

### Hadoop fs Shell Commands

**hadoop fs ls**:   
  
The hadoop ls command is used to list out the directories and files. An example is shown below:

> hadoop fs -ls /user/hadoop/employees

Found 1 items

-rw-r--r--   2 hadoop hadoop 2 2012-06-28 23:37 /user/hadoop/employees/000000\_0

The above command lists out the files in the employees directory.

> hadoop fs -ls /user/hadoop/dir

Found 1 items

drwxr-xr-x   - hadoop hadoop  0 2013-09-10 09:47 /user/hadoop/dir/products

The output of hadoop fs ls command is almost similar to the unix ls command. The only difference is in the second field. For a file, the second field indicates the number of replicas and for a directory, the second field is empty.   
  
**hadoop fs lsr**:   
  
The hadoop lsr command recursively displays the directories, sub directories and files in the specified directory. The usage example is shown below:

> hadoop fs -lsr /user/hadoop/dir

Found 2 items

drwxr-xr-x   - hadoop hadoop  0 2013-09-10 09:47 /user/hadoop/dir/products

-rw-r--r--   2 hadoop hadoop    1971684 2013-09-10 09:47 /user/hadoop/dir/products/products.dat

The hadoop fs lsr command is similar to the ls -R command in unix.   
  
**hadoop fs cat**:   
  
Hadoop cat command is used to print the contents of the file on the terminal (stdout). The usage example of hadoop cat command is shown below:

> hadoop fs -cat /user/hadoop/dir/products/products.dat

cloudera book by amazon

cloudera tutorial by ebay

**hadoop fs chgrp**:   
  
hadoop chgrp shell command is used to change the group association of files. Optionally you can use the -R option to change recursively through the directory structure. The usage of hadoop fs -chgrp is shown below:

hadoop fs -chgrp [-R] <NewGroupName> <file or directory name>

**hadoop fs chmod**:   
  
The hadoop chmod command is used to change the permissions of files. The -R option can be used to recursively change the permissions of a directory structure. The usage is shown below:

hadoop fs -chmod [-R] <mode | octal mode> <file or directory name>

**hadoop fs chown**:   
  
The hadoop chown command is used to change the ownership of files. The -R option can be used to recursively change the owner of a directory structure. The usage is shown below:

hadoop fs -chown [-R] <NewOwnerName>[:NewGroupName] <file or directory name>

**hadoop fs mkdir**:   
  
The hadoop mkdir command is for creating directories in the hdfs. You can use the -p option for creating parent directories. This is similar to the unix mkdir command. The usage example is shown below:

> hadoop fs -mkdir /user/hadoop/hadoopdemo

The above command creates the hadoopdemo directory in the /user/hadoop directory.

> hadoop fs -mkdir -p /user/hadoop/dir1/dir2/demo

The above command creates the dir1/dir2/demo directory in /user/hadoop directory.   
  
**hadoop fs copyFromLocal**:   
  
The hadoop copyFromLocal command is used to copy a file from the local file system to the hadoop hdfs. The syntax and usage example are shown below:

Syntax:

hadoop fs -copyFromLocal <localsrc> URI

Example:

Check the data in local file

> ls sales

2000,iphone

2001, htc

Now copy this file to hdfs

> hadoop fs -copyFromLocal sales /user/hadoop/hadoopdemo

View the contents of the hdfs file.

> hadoop fs -cat /user/hadoop/hadoopdemo/sales

2000,iphone

2001, htc

**hadoop fs copyToLocal**:   
  
The hadoop copyToLocal command is used to copy a file from the hdfs to the local file system. The syntax and usage example is shown below:

Syntax

hadoop fs -copyToLocal [-ignorecrc] [-crc] URI <localdst>

Example:

hadoop fs -copyToLocal /user/hadoop/hadoopdemo/sales salesdemo

The -ignorecrc option is used to copy the files that fail the crc check. The -crc option is for copying the files along with their CRC.   
  
**hadoop fs cp**:   
  
The hadoop cp command is for copying the source into the target. The cp command can also be used to copy multiple files into the target. In this case the target should be a directory. The syntax is shown below:

hadoop fs -cp /user/hadoop/SrcFile /user/hadoop/TgtFile

hadoop fs -cp /user/hadoop/file1 /user/hadoop/file2 hdfs://namenodehost/user/hadoop/TgtDirectory

**hadoop fs -put**:   
  
Hadoop put command is used to copy multiple sources to the destination system. The put command can also read the input from the stdin. The different syntaxes for the put command are shown below:

Syntax1: copy single file to hdfs

hadoop fs -put localfile /user/hadoop/hadoopdemo

Syntax2: copy multiple files to hdfs

hadoop fs -put localfile1 localfile2 /user/hadoop/hadoopdemo

Syntax3: Read input file name from stdin

hadoop fs -put - hdfs://namenodehost/user/hadoop/hadoopdemo

**hadoop fs get**:   
  
Hadoop get command copies the files from hdfs to the local file system. The syntax of the get command is shown below:

hadoop fs -get /user/hadoop/hadoopdemo/hdfsFileName localFileName

**hadoop fs getmerge**:   
  
hadoop getmerge command concatenates the files in the source directory into the destination file. The syntax of the getmerge shell command is shown below:

hadoop fs -getmerge <src> <localdst> [addnl]

The addnl option is for adding new line character at the end of each file.   
  
**hadoop fs moveFromLocal**:   
  
The hadoop moveFromLocal command moves a file from local file system to the hdfs directory. It removes the original source file. The usage example is shown below:

> hadoop fs -moveFromLocal products /user/hadoop/hadoopdemo

**hadoop fs mv**:   
  
It moves the files from source hdfs to destination hdfs. Hadoop mv command can also be used to move multiple source files into the target directory. In this case the target should be a directory. The syntax is shown below:

hadoop fs -mv /user/hadoop/SrcFile /user/hadoop/TgtFile

hadoop fs -mv /user/hadoop/file1 /user/hadoop/file2 hdfs://namenodehost/user/hadoop/TgtDirectory

**hadoop fs du**:   
  
The du command displays aggregate length of files contained in the directory or the length of a file in case its just a file. The syntax and usage is shown below:

hadoop fs -du hdfs://namenodehost/user/hadoop

**hadoop fs dus**:   
  
The hadoop dus command prints the summary of file lengths

> hadoop fs -dus hdfs://namenodehost/user/hadoop

hdfs://namenodehost/user/hadoop 21792568333

**hadoop fs expunge**:   
  
Used to empty the trash. The usage of expunge is shown below:

hadoop fs -expunge

**hadoop fs rm**:   
  
Removes the specified list of files and empty directories. An example is shown below:

hadoop fs -rm /user/hadoop/file

**hadoop fs -rmr**:   
  
Recursively deletes the files and sub directories. The usage of rmr is shown below:

hadoop fs -rmr /user/hadoop/dir

**hadoop fs setrep**:   
  
Hadoop setrep is used to change the replication factor of a file. Use the -R option for recursively changing the replication factor.

hadoop fs -setrep -w 4 -R /user/hadoop/dir

**hadoop fs stat**:   
  
Hadoop stat returns the stats information on a path. The syntax of stat is shown below:

hadoop fs -stat URI

> hadoop fs -stat /user/hadoop/

2013-09-24 07:53:04

**hadoop fs tail**:   
  
Hadoop tail command prints the last kilobytes of the file. The -f option can be used same as in unix.

> hafoop fs -tail /user/hadoop/sales.dat

12345 abc

2456 xyz

**hadoop fs test**:   
  
The hadoop test is used for file test operations. The syntax is shown below:

hadoop fs -test -[ezd] URI

Here "e" for checking the existence of a file, "z" for checking the file is zero length or not, "d" for checking the path is a directory or no. On success, the test command returns 1 else 0.   
  
**hadoop fs text**:   
  
The hadoop text command displays the source file in text format. The allowed source file formats are zip and TextRecordInputStream. The syntax is shown below:

hadoop fs -text <src>

**hadoop fs touchz**:   
  
The hadoop touchz command creates a zero byte file. This is similar to the touch command in unix. The syntax is shown below:

hadoop fs -touchz /user/hadoop/filename

NameNode java heap size  1 GB (1 GB per Million **blocks**)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Value** | **Notes** |
| mapred.map.parallel.copies | 50 | Higher number of parallel copies run by map |
| mapred.reduce.parallel.copies | 50 | Higher number of parallel copies run by reduce |
| mapred.job.reuse.jvm.num.tasks | -1 | Reuses JAVA JVM over and over |
| mapred.compress.map.output | TRUE | Compress map output |
| mapred.output.compress | TRUE | Compress output |
| mapred.output.compression.type | RECORD | Compression type is RECORD |
| dfs.replication | 1 | Replication = 1 |
| io.sort.mb | 600 | The total amount of buffer memory to use while sorting files, in megabytes. |
| mapred.tasktracker.map.tasks.maximum | 50 | The maximum number of map tasks that will be run simultaneously by a task tracker. |
| mapred.tasktracker.reduce.tasks.maximum | 50 | The maximum number of reducers tasks that will be run simultaneously by a task tracker. |
| mapred.map.tasks.speculative.execution | FALSE | Multiple instances of some map tasks may be executed in parallel. |
| mapred.reduce.tasks.speculative.execution | FALSE | Multiple instances of some reduce tasks may be executed in parallel. |
| io.sort.factor | 100 | The number of streams to merge at once while sorting files. |
| mapred.inmem.merge.threshold | 0 | The threshold, in terms of the number of files for the in-memory merge process. |
| hive.map.aggr | TRUE | Aggregation at map side |
| hive.exec.parallel | TRUE | Allow parallel execution |
| hive.exec.compress.intermediate | TRUE | Intermediate compression |
| hive.auto.convert.join | TRUE | Auto convert join |
| tasktracker.http.threads | 40 | The number of worker threads that for the http server. This is used for map output fetching |
| dfs.block.size | 2147483648 | Specifies the size of data blocks in which the input data set is split |
| mapred.max.split.size | 2415919104 | Maximum size of the Mapper |
| mapred.child.java.opts | -Xmx2048m | Java opts for the task tracker child processes |
| io.sort.spill.percent | 0.99 | Thread will begin to spill the contents to disk in the background once reached |
| io.file.buffer.size | 131072 | Size of read/write buffer used in sequence files |
| mapred.reduce.slowstart.completed.maps | 0 | Starts the reducer before the mapper is finished |