LEARNING

hadoop

Table of Contents

About

Chapter 1: Getting started with hadoop

Remarks

What is Apache Hadoop?

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

Apache Hadoop includes these modules:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS)**: A distributed file system that provides high-throughput access to application data.
- **Hadoop YARN**: A framework for job scheduling and cluster resource management.
- **Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

Reference:

Apache Hadoop

Versions

Version	Release Notes	Release Date
3.0.0-alpha1		2016-08-30
2.7.3	Click here - 2.7.3	2016-01-25
2.6.4	Click here - 2.6.4	2016-02-11
2.7.2	Click here - 2.7.2	2016-01-25
2.6.3	Click here - 2.6.3	2015-12-17
2.6.2	Click here - 2.6.2	2015-10-28
2.7.1	Click here - 2.7.1	2015-07-06

Examples

Installation or Setup on Linux

A Pseudo Distributed Cluster Setup Procedure

Prerequisites

- Install JDK1.7 and set JAVA_HOME environment variable.
- Create a new user as "hadoop".

  `useradd hadoop`

- Setup password-less SSH login to its own account

  ```
  su - hadoop
  ssh-keygen
  << Press ENTER for all prompts >>
  cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
  chmod 0600 ~/.ssh/authorized_keys
  ```

- Verify by performing `ssh localhost`
- Disable IPV6 by editing `/etc/sysctl.conf` with the followings:

  ```
  net.ipv6.conf.all.disable_ipv6 = 1
  net.ipv6.conf.default.disable_ipv6 = 1
  net.ipv6.conf.lo.disable_ipv6 = 1
  ```

- Check that using `cat /proc/sys/net/ipv6/conf/all/disable_ipv6`

  (should return 1)

Installation and Configuration:

- Download required version of Hadoop from Apache archives using `wget` command.

  ```
   cd /opt/hadoop/
   wget http:/addresstoarchive/hadoop-2.x.x/xxxxx.gz
   tar -xvf hadoop-2.x.x.gz
   mv hadoop-2.x.x.gz hadoop
  (or)

   ln -s hadoop-2.x.x.gz hadoop
   chown -R hadoop:hadoop hadoop
  ```

- Update `.bashrc`/`.kshrc` based on your shell with below environment variables

  ```
    export HADOOP_PREFIX=/opt/hadoop/hadoop
    export HADOOP_CONF_DIR=$HADOOP_PREFIX/etc/hadoop
    export JAVA_HOME=/java/home/path
  ```

```
export PATH=$PATH:$HADOOP_PREFIX/bin:$HADOOP_PREFIX/sbin:$JAVA_HOME/bin
```

- In `$HADOOP_HOME/etc/hadoop` directory edit below files
    - core-site.xml

      ```
      <configuration>
          <property>
              <name>fs.defaultFS</name>
              <value>hdfs://localhost:8020</value>
          </property>
      </configuration>
      ```

    - mapred-site.xml

      Create `mapred-site.xml` from its template

      `cp mapred-site.xml.template mapred-site.xml`

      ```
       <configuration>
           <property>
               <name>mapreduce.framework.name</name>
               <value>yarn</value>
           </property>
       </configuration>
      ```

    - yarn-site.xml

      ```
      <configuration>
          <property>
              <name>yarn.resourcemanager.hostname</name>
              <value>localhost</value>
          </property>
          <property>
              <name>yarn.nodemanager.aux-services</name>
              <value>mapreduce_shuffle</value>
          </property>
      </configuration>
      ```

    - hdfs-site.xml

      ```
      <configuration>
          <property>
              <name>dfs.replication</name>
              <value>1</value>
          </property>
          <property>
              <name>dfs.namenode.name.dir</name>
              <value>file:///home/hadoop/hdfs/namenode</value>
          </property>
          <property>
              <name>dfs.datanode.data.dir</name>
              <value>file:///home/hadoop/hdfs/datanode</value>
          </property>
      </configuration>
      ```

Create the parent folder to store the hadoop data

```
mkdir -p /home/hadoop/hdfs
```

- Format NameNode (cleans up the directory and creates necessary meta files)

```
hdfs namenode -format
```

- Start all services:

```
start-dfs.sh && start-yarn.sh
mr-jobhistory-server.sh start historyserver
```

Instead use start-all.sh (deprecated).

- Check all running java processes

```
jps
```

- Namenode Web Interface: http://localhost:50070/
- Resource manager Web Interface: http://localhost:8088/
- To stop daemons(services):

```
stop-dfs.sh && stop-yarn.sh
mr-jobhistory-daemon.sh stop historyserver
```

Instead use stop-all.sh (deprecated).

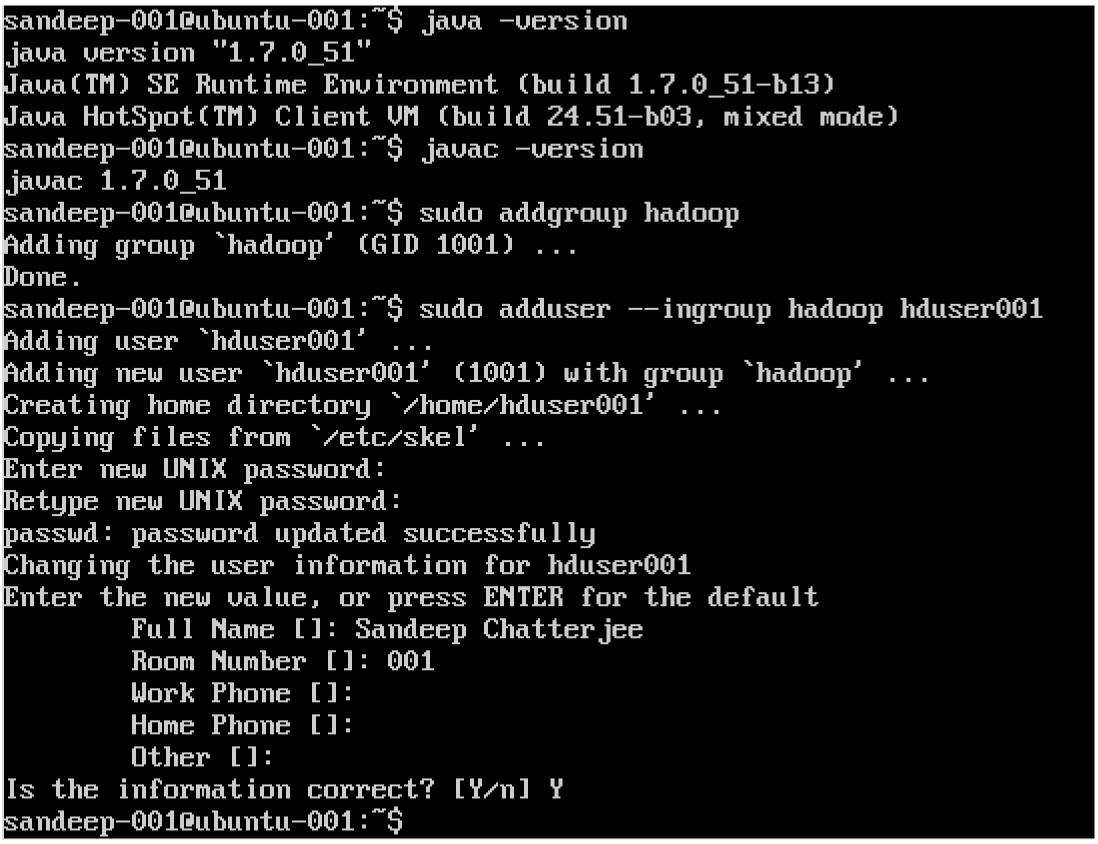
Installation of Hadoop on ubuntu

Creating Hadoop User:

```
sudo addgroup hadoop
```

Adding a user:

```
sudo adduser --ingroup hadoop hduser001
```

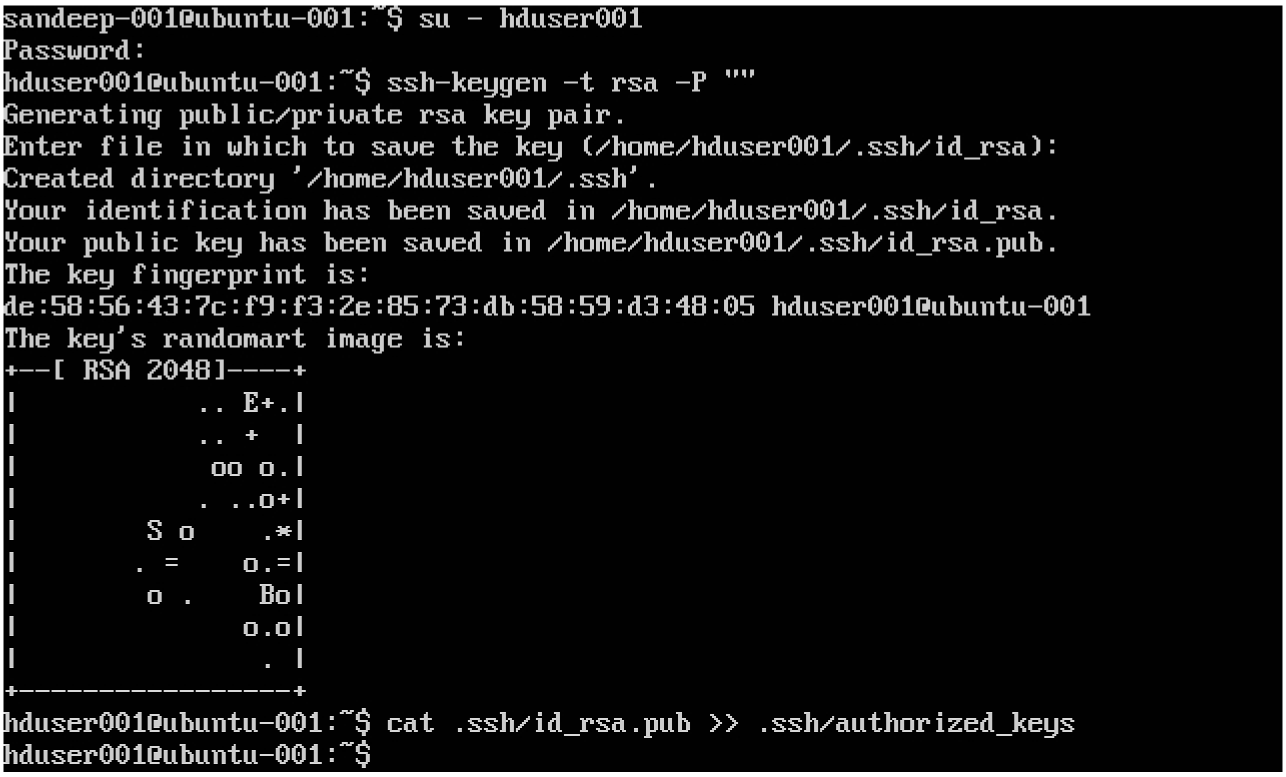


```
sandeep-001@ubuntu-001:~$ java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) Client VM (build 24.51-b03, mixed mode)
sandeep-001@ubuntu-001:~$ javac -version
javac 1.7.0_51
sandeep-001@ubuntu-001:~$ sudo addgroup hadoop
Adding group `hadoop' (GID 1001) ...
Done.
sandeep-001@ubuntu-001:~$ sudo adduser --ingroup hadoop hduser001
Adding user `hduser001' ...
Adding new user `hduser001' (1001) with group `hadoop' ...
Creating home directory `/home/hduser001' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser001
Enter the new value, or press ENTER for the default
        Full Name []: Sandeep Chatterjee
        Room Number []: 001
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
sandeep-001@ubuntu-001:~$
```

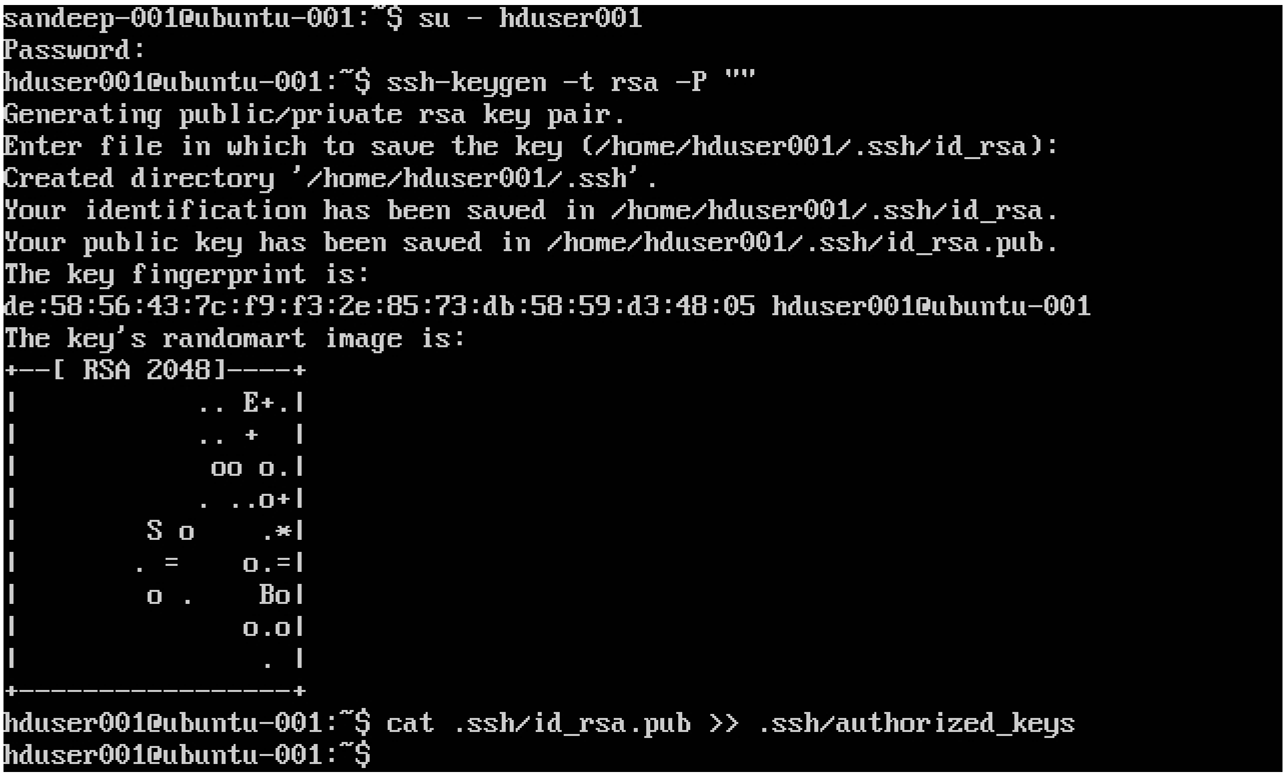
Configuring SSH:

```
su -hduser001
ssh-keygen -t rsa -P ""
cat .ssh/id rsa.pub >> .ssh/authorized_keys
```

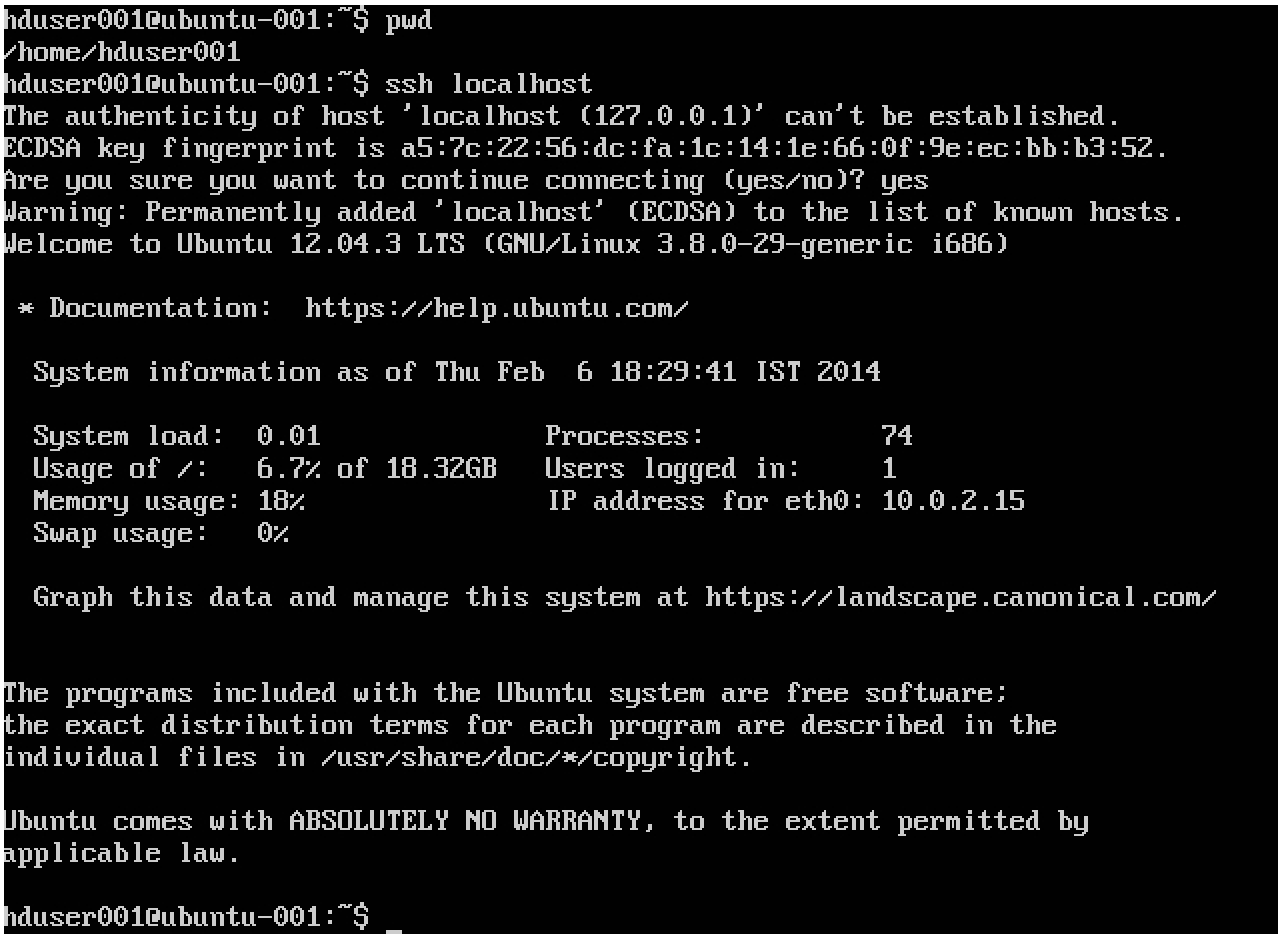
Note: If you get errors [*bash: .ssh/authorized_keys: No such file or directory*] whilst writing the authorized key. Check here.



```
sandeep-001@ubuntu-001:~$ su - hduser001
Password:
hduser001@ubuntu-001:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser001/.ssh/id_rsa):
Created directory '/home/hduser001/.ssh'.
Your identification has been saved in /home/hduser001/.ssh/id_rsa.
Your public key has been saved in /home/hduser001/.ssh/id_rsa.pub.
The key fingerprint is:
de:58:56:43:7c:f9:f3:2e:85:73:db:58:59:d3:48:05 hduser001@ubuntu-001
The key's randomart image is:
+--[ RSA 2048]----+
|           .. E+.|
|           .. +  |
|            oo o.|
|           . ..o+|
|       S o    .*|
|      . =    o.=|
|       o .    Bo|
|             o.o|
|               . |
+-----------------+
hduser001@ubuntu-001:~$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
hduser001@ubuntu-001:~$
```



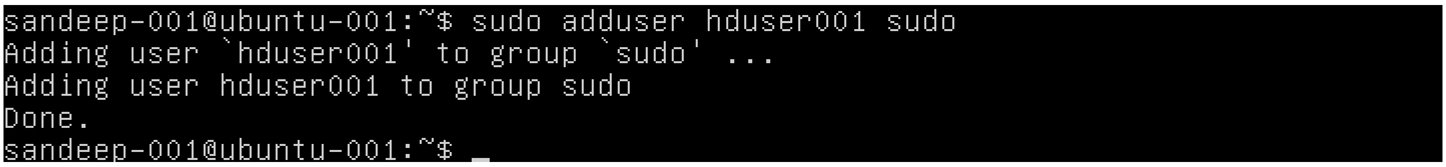
```
sandeep-001@ubuntu-001:~$ su - hduser001
Password:
hduser001@ubuntu-001:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser001/.ssh/id_rsa):
Created directory '/home/hduser001/.ssh'.
Your identification has been saved in /home/hduser001/.ssh/id_rsa.
Your public key has been saved in /home/hduser001/.ssh/id_rsa.pub.
The key fingerprint is:
de:58:56:43:7c:f9:f3:2e:85:73:db:58:59:d3:48:05 hduser001@ubuntu-001
The key's randomart image is:
+--[ RSA 2048]----+
|           .. E+.|
|           .. +  |
|            oo o.|
|           . ..o+|
|       S o    .*|
|      . =    o.=|
|       o .    Bo|
|             o.o|
|               . |
+-----------------+
hduser001@ubuntu-001:~$ cat .ssh/id_rsa.pub >> .ssh/authorized_keys
hduser001@ubuntu-001:~$
```



```
hduser001@ubuntu-001:~$ pwd
/home/hduser001
hduser001@ubuntu-001:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is a5:7c:22:56:dc:fa:1c:14:1e:66:0f:9e:ec:bb:b3:52.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 12.04.3 LTS (GNU/Linux 3.8.0-29-generic i686)

 * Documentation:  https://help.ubuntu.com/

  System information as of Thu Feb  6 18:29:41 IST 2014

  System load:  0.01              Processes:           74
  Usage of /:   6.7% of 18.32GB   Users logged in:     1
  Memory usage: 18%               IP address for eth0: 10.0.2.15
  Swap usage:   0%

  Graph this data and manage this system at https://landscape.canonical.com/


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

hduser001@ubuntu-001:~$ _
```

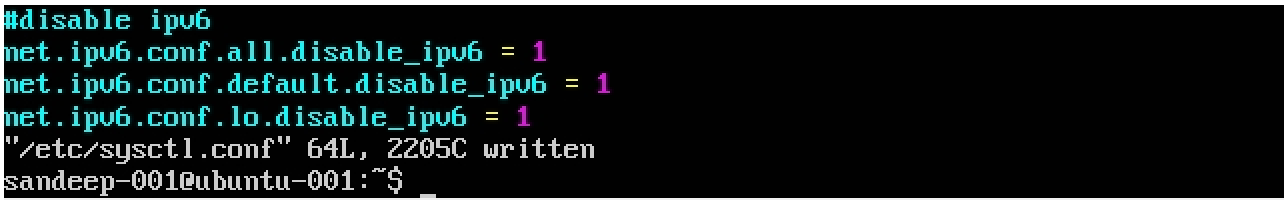
Add hadoop user to sudoer's list:

```
sudo adduser hduser001 sudo
```

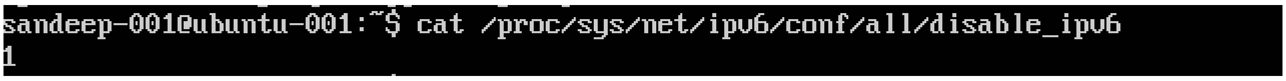


```
sandeep-001@ubuntu-001:~$ sudo adduser hduser001 sudo
Adding user `hduser001' to group `sudo' ...
Adding user hduser001 to group sudo
Done.
sandeep-001@ubuntu-001:~$ _
```

Disabling IPv6:



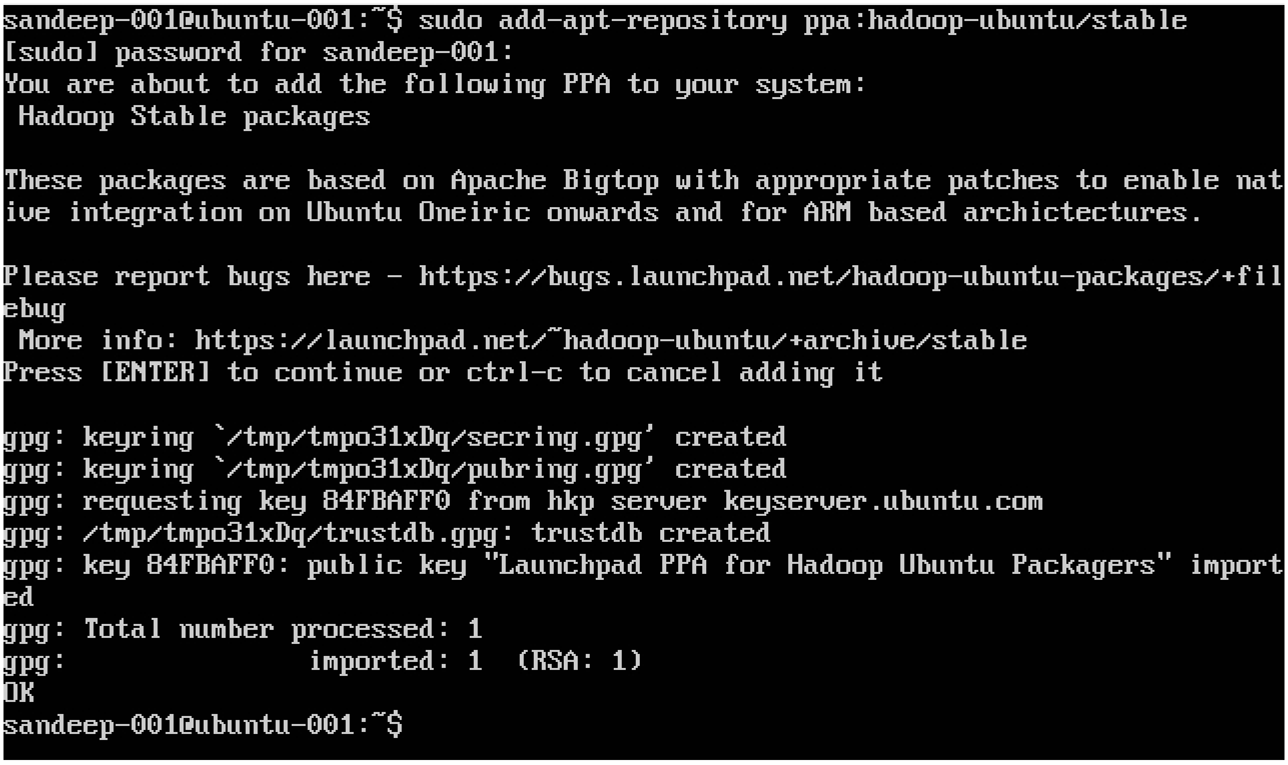
```
#disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
"/etc/sysctl.conf" 64L, 2205C written
sandeep-001@ubuntu-001:~$ _
```



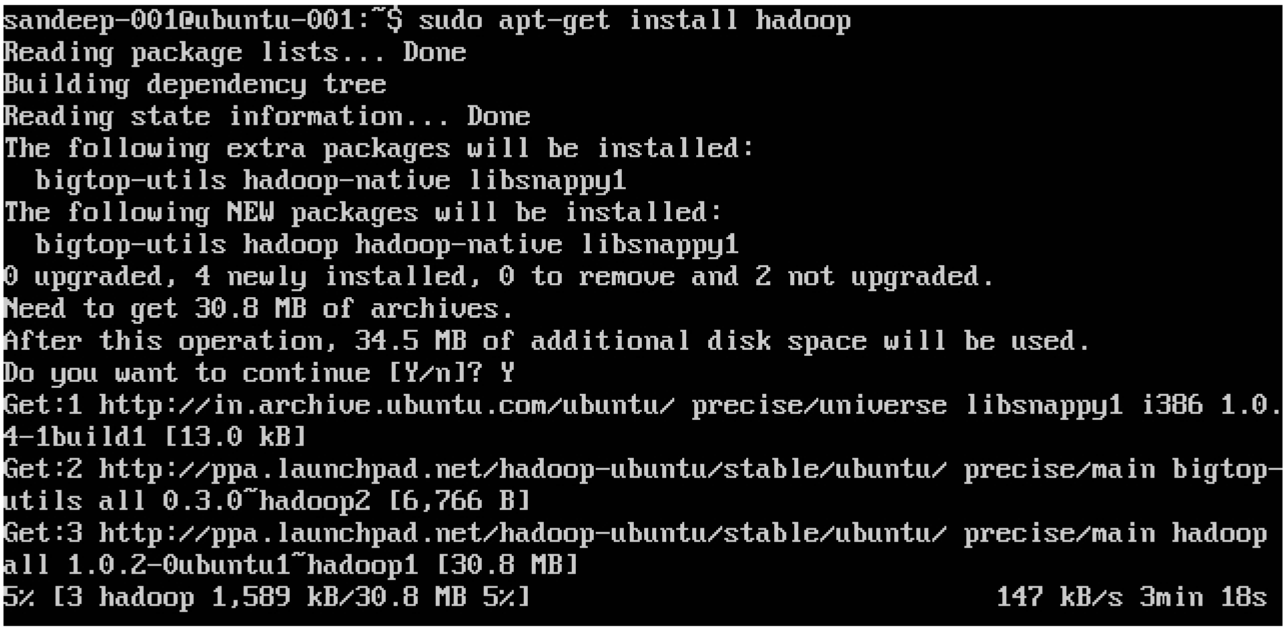
```
sandeep-001@ubuntu-001:~$ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
1
```

Installing Hadoop:

```
sudo add-apt-repository ppa:hadoop-ubuntu/stable
sudo apt-get install hadoop
```



```
sandeep-001@ubuntu-001:~$ sudo add-apt-repository ppa:hadoop-ubuntu/stable
[sudo] password for sandeep-001:
You are about to add the following PPA to your system:
 Hadoop Stable packages

These packages are based on Apache Bigtop with appropriate patches to enable nat
ive integration on Ubuntu Oneiric onwards and for ARM based archictectures.

Please report bugs here - https://bugs.launchpad.net/hadoop-ubuntu-packages/+fil
ebug
 More info: https://launchpad.net/~hadoop-ubuntu/+archive/stable
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmpo31xDq/secring.gpg' created
gpg: keyring `/tmp/tmpo31xDq/pubring.gpg' created
gpg: requesting key 84FBAFF0 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmpo31xDq/trustdb.gpg: trustdb created
gpg: key 84FBAFF0: public key "Launchpad PPA for Hadoop Ubuntu Packagers" import
ed
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
OK
sandeep-001@ubuntu-001:~$
```



```
sandeep-001@ubuntu-001:~$ sudo apt-get install hadoop
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  bigtop-utils hadoop-native libsnappy1
The following NEW packages will be installed:
  bigtop-utils hadoop hadoop-native libsnappy1
0 upgraded, 4 newly installed, 0 to remove and 2 not upgraded.
Need to get 30.8 MB of archives.
After this operation, 34.5 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise/universe libsnappy1 i386 1.0.
4-1build1 [13.0 kB]
Get:2 http://ppa.launchpad.net/hadoop-ubuntu/stable/ubuntu/ precise/main bigtop-
utils all 0.3.0~hadoop2 [6,766 B]
Get:3 http://ppa.launchpad.net/hadoop-ubuntu/stable/ubuntu/ precise/main hadoop
all 1.0.2-0ubuntu1~hadoop1 [30.8 MB]
5% [3 hadoop 1,589 kB/30.8 MB 5%]                               147 kB/s 3min 18s
```

Hadoop overview and HDFS

Hadoop is an open-source software framework for storage and large-scale processing of data-sets in a distributed computing environment. It is sponsored by Apache Software Foundation. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

History

- Hadoop was created by Doug Cutting and Mike Cafarella in 2005.
- Cutting, who was working at Yahoo! at the time, named it after his son's toy elephant.
- It was originally developed to support distribution for the search engine project.

Major modules of hadoop

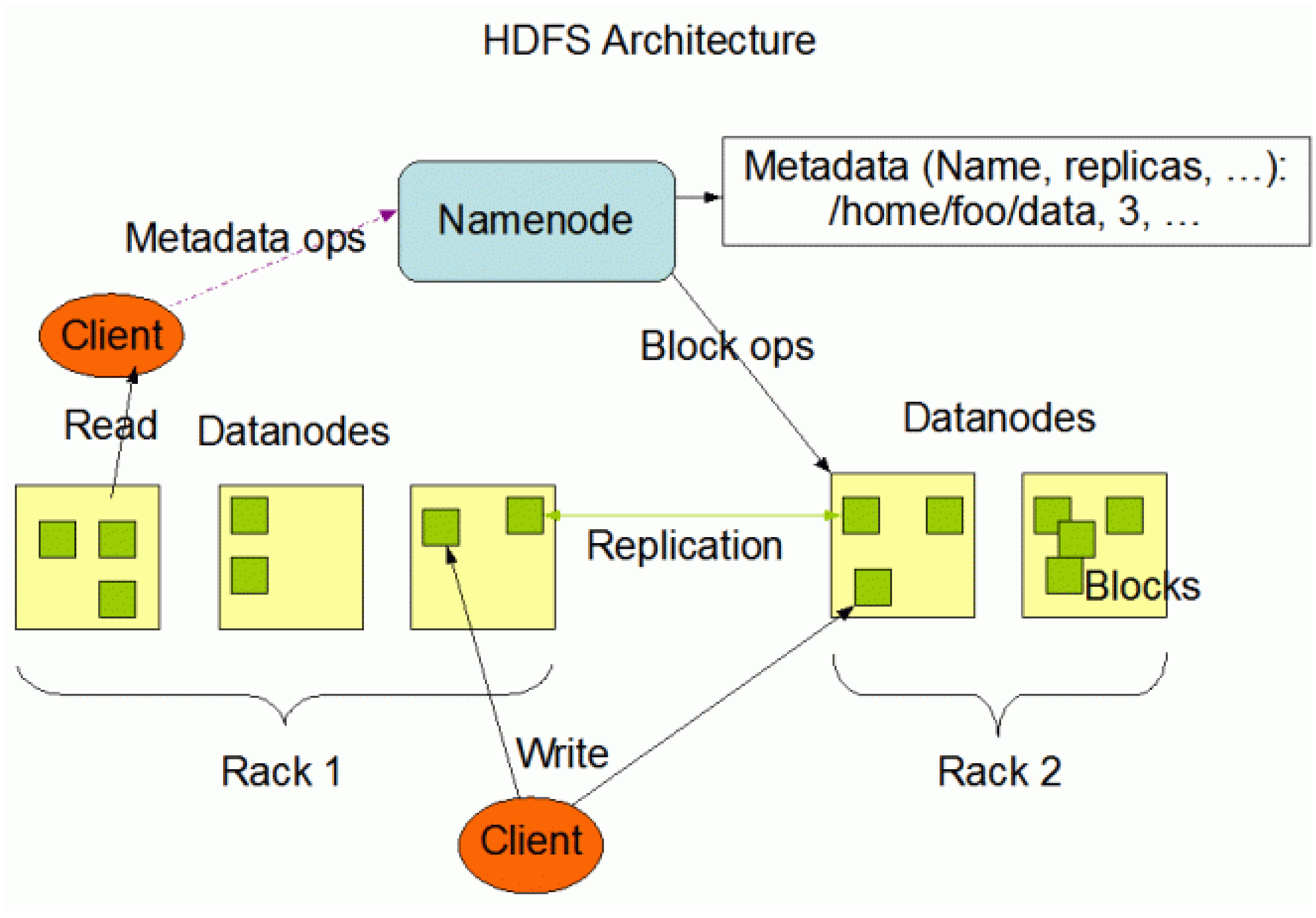
Hadoop Distributed File System (HDFS): A distributed file system that provides high-throughput access to application data. Hadoop MapReduce: A software framework for distributed processing of large data sets on compute clusters.

Hadoop File System Basic Features

Highly fault-tolerant. High throughput. Suitable for applications with large data sets. Can be built out of commodity hardware.

Namenode and Datanodes

Master/slave architecture. HDFS cluster consists of a single Namenode, a master server that manages the file system namespace and regulates access to files by clients. The DataNodes manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files. A file is split into one or more blocks and set of blocks are stored in DataNodes. DataNodes: serves read, write requests, performs block creation, deletion, and replication upon instruction from Namenode.

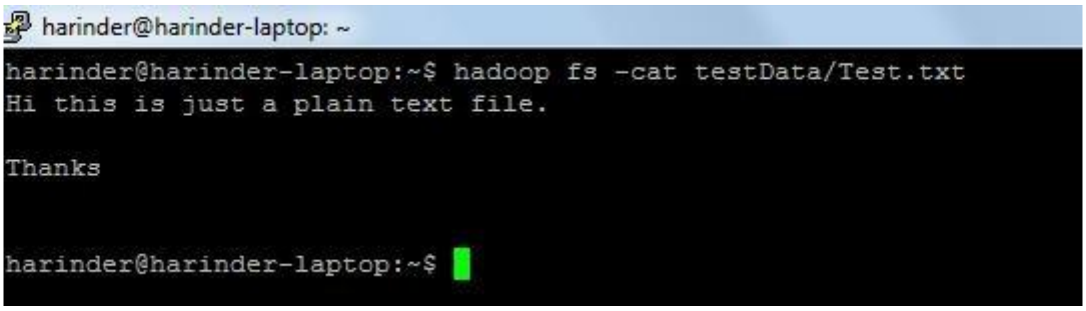


HDFS is designed to store very large files across machines in a large cluster. Each file is a sequence of blocks. All blocks in the file except the last are of the same size. Blocks are replicated for fault tolerance. The Namenode receives a Heartbeat and a BlockReport from each DataNode in the cluster. BlockReport contains all the blocks on a Datanode.

Hadoop Shell Commands

Common commands used:-

ls Usage: **hadoop fs –ls Path**(dir/file path to list). **Cat** Usage: **hadoop fs -cat PathOfFileToView**



harinder@harinder-laptop: ~

```
harinder@harinder-laptop:~$ hadoop fs -cat testData/Test.txt
Hi this is just a plain text file.

Thanks


harinder@harinder-laptop:~$
```

Link for hadoop shell commands:- https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

Read Getting started with hadoop online: https://riptutorial.com/hadoop/topic/926/getting-started-with-hadoop

Chapter 2: Debugging Hadoop MR Java code in local eclipse dev environment.

Introduction

The basic thing to remember here is that debugging a Hadoop MR job is going to be similar to any remotely debugged application in Eclipse.

A debugger or debugging tool is a computer program that is used to test and debug other programs (the “target” program). It is greatly useful specially for a Hadoop environment wherein there is little room for error and one small error can cause a huge loss.

Remarks

That is all you need to do.

Examples

Steps for configuration

As you would know, Hadoop can be run in the local environment in 3 different modes :

1. Local Mode
2. Pseudo Distributed Mode
3. Fully Distributed Mode (Cluster)

Typically you will be running your local hadoop setup in Pseudo Distributed Mode to leverage HDFS and Map Reduce(MR). However you cannot debug MR programs in this mode as each Map/Reduce task will be running in a separate JVM process so you need to switch back to Local mode where you can run your MR programs in a single JVM process.

Here are the quick and simple steps to debug this in your local environment:

1. Run hadoop in local mode for debugging so mapper and reducer tasks run in a single JVM instead of separate JVMs. Below steps help you do it.

2. Configure HADOOP_OPTS to enable debugging so when you run your Hadoop job, it will be waiting for the debugger to connect. Below is the command to debug the same at port 8080.

(export HADOOP_OPTS=”-agentlib:jdwp=transport=dt_socket,server=y,suspend=y,address=8008“)

3. Configure fs.default.name value in core-site.xml to file:/// from hdfs://. You won’t be using hdfs in local mode.

4. Configure mapred.job.tracker value in mapred-site.xml to local. This will instruct Hadoop to run MR tasks in a single JVM.
5. Create debug configuration for Eclipse and set the port to 8008 – typical stuff. For that go to the debugger configurations and create a new Remote Java Application type of configuration and set the port as 8080 in the settings.
6. Run your hadoop job (it will be waiting for the debugger to connect) and then launch Eclipse in debug mode with the above configuration. Do make sure to put a break-point first.

Read Debugging Hadoop MR Java code in local eclipse dev environment. online: https://riptutorial.com/hadoop/topic/10063/debugging-hadoop-mr-java-code-in-local-eclipse-dev-environment-

Chapter 3: Hadoop commands

Syntax

- Hadoop v1 commands: `hadoop fs -<command>`
- Hadoop v2 commands: `hdfs dfs -<command>`

Examples

Hadoop v1 Commands

1. Print the Hadoop version

```
hadoop version
```

2. List the contents of the root directory in HDFS

```
hadoop fs -ls /
```

3. Report the amount of space used and

available on currently mounted filesystem

```
hadoop fs -df hdfs:/
```

4. Count the number of directories,files and bytes under

the paths that match the specified file pattern

```
hadoop fs -count hdfs:/
```

5. Run a DFS filesystem checking utility

```
hadoop fsck – /
```

6. Run a cluster balancing utility

```
hadoop balancer
```

7. Create a new directory named “hadoop” below the

/user/training directory in HDFS. Since you’re

currently logged in with the “training” user ID,

/user/training is your home directory in HDFS.

```
hadoop fs -mkdir /user/training/hadoop
```

8. Add a sample text file from the local directory

named “data” to the new directory you

created in HDFS

during the previous step.

```
hadoop fs -put data/sample.txt /user/training/hadoop
```

9. List the contents of this new directory in HDFS.

```
hadoop fs -ls /user/training/hadoop
```

10. Add the entire local directory called “retail” to the

/user/training directory in HDFS.

```
hadoop fs -put data/retail /user/training/hadoop
```

11. Since /user/training is your home directory in HDFS,

any command that does not have an absolute path is

interpreted as relative to that directory. The next

command will therefore list your home directory, and

should show the items you’ve just added there.

```
hadoop fs -ls
```

12. See how much space this directory occupies in HDFS.

```
hadoop fs -du -s -h hadoop/retail
```

13. Delete a file ‘customers’ from the “retail” directory.

```
hadoop fs -rm hadoop/retail/customers
```

14. Ensure this file is no longer in HDFS.

```
hadoop fs -ls hadoop/retail/customers
```

15. Delete all files from the “retail” directory using a wildcard.

```
hadoop fs -rm hadoop/retail/*
```

16. To empty the trash

```
hadoop fs -expunge
```

17. Finally, remove the entire retail directory and all

of its contents in HDFS.

```
hadoop fs -rm -r hadoop/retail
```

18. List the hadoop directory again

```
hadoop fs -ls hadoop
```

19. Add the purchases.txt file from the local directory

named “/home/training/” to the hadoop directory you created in HDFS

```
hadoop fs -copyFromLocal /home/training/purchases.txt hadoop/
```

20. To view the contents of your text file purchases.txt

which is present in your hadoop directory.

```
hadoop fs -cat hadoop/purchases.txt
```

21. Add the purchases.txt file from “hadoop” directory which is present in HDFS directory

to the directory “data” which is present in your local directory

```
hadoop fs -copyToLocal hadoop/purchases.txt /home/training/data
```

22. cp is used to copy files between directories present in HDFS

```
hadoop fs -cp /user/training/*.txt /user/training/hadoop
```

23. ‘-get’ command can be used alternaively to ‘-copyToLocal’ command

```
hadoop fs -get hadoop/sample.txt /home/training/
```

24. Display last kilobyte of the file “purchases.txt” to stdout.

```
hadoop fs -tail hadoop/purchases.txt
```

25. Default file permissions are 666 in HDFS

Use ‘-chmod’ command to change permissions of a file

```
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chmod 600 hadoop/purchases.txt
```

26. Default names of owner and group are training,training

Use ‘-chown’ to change owner name and group name simultaneously

```
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chown root:root hadoop/purchases.txt
```

27. Default name of group is training

Use ‘-chgrp’ command to change group name

```
hadoop fs -ls hadoop/purchases.txt
sudo -u hdfs hadoop fs -chgrp training hadoop/purchases.txt
```

28. Move a directory from one location to other

```
hadoop fs -mv hadoop apache_hadoop
```

29. Default replication factor to a file is 3.

Use ‘-setrep’ command to change replication factor of a file

```
hadoop fs -setrep -w 2 apache_hadoop/sample.txt
```

30. Copy a directory from one node in the cluster to another

Use ‘-distcp’ command to copy,

-overwrite option to overwrite in an existing files

-update command to synchronize both directories

```
hadoop fs -distcp hdfs://namenodeA/apache_hadoop hdfs://namenodeB/hadoop
```

31. Command to make the name node leave safe mode

```
hadoop fs -expunge
sudo -u hdfs hdfs dfsadmin -safemode leave
```

32. List all the hadoop file system shell commands

```
hadoop fs
```

33. Get hdfs quota values and the current count of names and bytes in use.

```
hadoop fs -count -q [-h] [-v] <directory>...<directory>
```

34. Last but not least, always ask for help!

```
hadoop fs -help
```

Hadoop v2 Commands

appendToFile: Append single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and appends to destination file system. Keep the as -

```
 hdfs dfs -appendToFile [localfile1 localfile2 ..] [/HDFS/FILE/PATH..]
```

cat: Copies source paths to stdout.

```
 hdfs dfs -cat URI [URI …]
```

chgrp: Changes the group association of files. With -R, makes the change recursively by way of the directory structure. The user must be the file owner or the superuser.

```
 hdfs dfs -chgrp [-R] GROUP URI [URI …]
```

chmod: Changes the permissions of files. With -R, makes the change recursively by way of the directory structure. The user must be the file owner or the superuser

```
 hdfs dfs -chmod [-R] <MODE[,MODE]... | OCTALMODE> URI [URI …]
```

chown: Changes the owner of files. With -R, makes the change recursively by way of the directory structure. The user must be the superuser.

```
 hdfs dfs -chown [-R] [OWNER][:[GROUP]] URI [URI ]
```

copyFromLocal: Works similarly to the put command, except that the source is restricted to a local file reference.

```
hdfs dfs -copyFromLocal <localsrc> URI
```

copyToLocal: Works similarly to the get command, except that the destination is restricted to a local file reference.

```
hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>
```

count: Counts the number of directories, files, and bytes under the paths that match the specified file pattern.

```
hdfs dfs -count [-q] [-h] <paths>
```

cp: Copies one or more files from a specified source to a specified destination. If you specify multiple sources, the specified destination must be a directory.

```
hdfs dfs -cp URI [URI …] <dest>
```

du: Displays the size of the specified file, or the sizes of files and directories that are contained in the specified directory. If you specify the -s option, displays an aggregate summary of file sizes rather than individual file sizes. If you specify the -h option, formats the file sizes in a "human-readable" way.

```
hdfs dfs -du [-s] [-h] URI [URI …]
```

dus: Displays a summary of file sizes; equivalent to hdfs dfs -du –s.

```
hdfs dfs -dus <args>
```

expunge: Empties the trash. When you delete a file, it isn't removed immediately from HDFS, but is renamed to a file in the /trash directory. As long as the file remains there, you can undelete it if you change your mind, though only the latest copy of the deleted file can be restored.

```
hdfs dfs –expunge
```

get: Copies files to the local file system. Files that fail a cyclic redundancy check (CRC) can still be copied if you specify the -ignorecrc option. The CRC is a common technique for detecting data transmission errors. CRC checksum files have the .crc extension and are used to verify the data integrity of another file. These files are copied if you specify the -crc option.

```
hdfs dfs -get [-ignorecrc] [-crc] <src> <localdst>
```

getmerge: Concatenates the files in src and writes the result to the specified local destination file. To add a newline character at the end of each file, specify the addnl option.

```
hdfs dfs -getmerge <src> <localdst> [addnl]
```

ls: Returns statistics for the specified files or directories.

```
hdfs dfs -ls <args>
```

lsr: Serves as the recursive version of ls; similar to the Unix command ls -R.

```
hdfs dfs -lsr <args>
```

mkdir: Creates directories on one or more specified paths. Its behavior is similar to the Unix mkdir -p command, which creates all directories that lead up to the specified directory if they don't exist already.

```
hdfs dfs -mkdir <paths>
```

moveFromLocal: Works similarly to the put command, except that the source is deleted after it is copied.

```
hdfs dfs -moveFromLocal <localsrc> <dest>
```

mv: Moves one or more files from a specified source to a specified destination. If you specify multiple sources, the specified destination must be a directory. Moving files across file systems isn't permitted.

```
hdfs dfs -mv URI [URI …] <dest>
```

put: Copies files from the local file system to the destination file system. This command can also read input from stdin and write to the destination file system.

```
hdfs dfs -put <localsrc> ... <dest>
```

rm: Deletes one or more specified files. This command doesn't delete empty directories or files. To bypass the trash (if it's enabled) and delete the specified files immediately, specify the -skipTrash option.

```
hdfs dfs -rm [-skipTrash] URI [URI …]
```

rm r: Serves as the recursive version of –rm.

```
hdfs dfs -rm -r [-skipTrash] URI [URI …]
```

setrep: Changes the replication factor for a specified file or directory. With -R, makes the change recursively by way of the directory structure.

```
hdfs dfs -setrep <rep> [-R] <path>
```

stat: Displays information about the specified path.

```
hdfs dfs -stat URI [URI …]
```

tail: Displays the last kilobyte of a specified file to stdout. The syntax supports the Unix -f option, which enables the specified file to be monitored. As new lines are added to the file by another process, tail updates the display.

```
hdfs dfs -tail [-f] URI
```

test: Returns attributes of the specified file or directory. Specifies -e to determine whether the file or directory exists; -z to determine whether the file or directory is empty; and -d to determine whether the URI is a directory.

```
hdfs dfs -test -[ezd] URI
```

text: Outputs a specified source file in text format. Valid input file formats are zip and TextRecordInputStream.

```
hdfs dfs -text <src>
```

touchz: Creates a new, empty file of size 0 in the specified path.

```
hdfs dfs -touchz <path>
```

Read Hadoop commands online: https://riptutorial.com/hadoop/topic/3870/hadoop-commands

Chapter 4: Hadoop load data

Examples

Load data into hadoop hdfs

STEP 1: CREATE A DIRECTORY IN HDFS, UPLOAD A FILE AND LIST CONTENTS

Let’s learn by writing the syntax. You will be able to copy and paste the following example commands into your terminal:

hadoop fs -mkdir:

Takes the path URI’s as an argument and creates a directory or multiple directories.

Usage:

```
# hadoop fs -mkdir <paths>
```

Example:

```
hadoop fs -mkdir /user/hadoop
hadoop fs -mkdir /user/hadoop/dir1 /user/hadoop/dir2 /user/hadoop/dir3
```

hadoop fs -put:

Copies single src file or multiple src files from local file system to the Hadoop Distributed File System.

Usage:

```
# hadoop fs -put <local-src> ... <HDFS_dest_path>
```

Example:

```
hadoop fs -put popularNames.txt /user/hadoop/dir1/popularNames.txt
```

hadoop fs -copyFromLocal:

Copies single src file or multiple src files from local file system to the Hadoop Distributed File System.

Usage:

```
    # hadoop fs -copyFromLocal <local-src> ... <HDFS_dest_path>
```

Example:

```
    hadoop fs -copyFromLocal popularNames.txt /user/hadoop/dir1/popularNames.txt
```

hadoop fs -moveFromLocal:

Similar to put command, except that the source localsrc is deleted after it’s copied.

Usage:

```
    # hadoop fs -moveFromLocal <local-src> ... <HDFS_dest_path>
```

Example:

```
    hadoop fs -moveFromLocal popularNames.txt /user/hadoop/dir1/popularNames.txt
```

SQOOP DATA TRANSFER TOOL:

We can also load data into HDFS directly from Relational databases using Sqoop(a command line tool for data transfer from RDBMS to HDFS and vice versa).

Usage:

```
$ sqoop import --connect CONNECTION_STRING --username USER_NAME --table TABLE_NAME
```

Example:

```
$ sqoop import --connect jdbc:mysql://localhost/db --username foo --table TEST
```

Read Hadoop load data online: https://riptutorial.com/hadoop/topic/3846/hadoop-load-data

Chapter 5: hue

Introduction

Hue is an User Interface to connect and work with most of the commonly used Bigdata technologies like HDFS, Hive, Spark, Hbase, Sqoop, Impala, Pig, Oozie etc. Hue also supports running queries against Relational databases.

Hue, a django web application, was primarily built as a workbench for running Hive queries. Later the functionality of Hue increased to support different components of Hadoop Ecosystem. It is available as open source software under Apache License.

Examples

Setup process

Instalation Dependencies

Hue installation process details are not available for most operating systems, so depending on the OS, there might be variations on the dependencies you need to install prior to executing the install script provided in the installation package:

CentOS

```
sudo yum install ant
sudo yum install python-devel.x86_64
sudo yum install krb5-devel.x86_64
sudo yum install krb5-libs.x86_64
sudo yum install libxml2.x86_64
sudo yum install python-lxml.x86_64
sudo yum install libxslt-devel.x86_64
sudo yum install mysql-devel.x86_64
sudo yum install openssl-devel.x86_64
sudo yum install libgsasl-devel.x86_64
sudo yum install sqlite-devel.x86_64
sudo yum install openldap-devel.x86_64
sudo yum install -y libffi libffi-devel
sudo yum install mysql-devel gcc gcc-devel python-devel
sudo yum install rsync
sudo yum install maven
wget https://bootstrap.pypa.io/ez_setup.py -O - | sudo python
```

1. GMP

- CentOS > 7.x
  `sudo yum install libgmp3-dev`

- CentOS < 6.x
  `sudo yum install gmp gmp-devel gmp-status`

Hue Installation in Ubuntu

This installation assumes `hadoop` to be pre-installed under `hadoop` user.

Prerequisites:

Hue depends on these following packages

1. gcc
2. g++
3. libxml2-dev
4. libxlst-dev
5. libsasl2-dev
6. libsasl2-modules-gssapi-mit
7. libmysqlclient-dev
8. python-dev
9. python-setuptools
10. libsqlite3-dev
11. ant
12. libkrb5-dev
13. libtidy-0.99-0
14. libldap2-dev
15. libssl-dev
16. libgmp3-dev

Installing all the packages

```
sudo apt-get update
sudo apt-get install gcc g++ libxml2-dev libxslt-dev libsasl2-dev libsasl2-modules-gssapi-mit
libmysqlclient-dev python-dev python-setuptools libsqlite3-dev ant libkrb5-dev libtidy-0.99-0
libldap2-dev libssl-dev libgmp3-dev
```

Installation and Configuration

Performing installation as `hadoop` user.

```
su - hadoop
```

1. Download Hue from gethue.com (this link is an example obtained from Hue website)

   ```
   wget https://dl.dropboxusercontent.com/u/730827/hue/releases/3.9.0/hue-3.9.0.tgz
   ```

2. Extract the downloaded tarball

   ```
   tar -xvf hue-3.9.0.tgz
   ```

3. Execute install command

   ```
   cd hue-3.9.0
   PREFIX=/home/hadoop/ make install
   ```

4. Once the above process is completed,

   Update `~/.bashrc` file,

   ```
   export HUE_HOME=/home/hadoop/hue
   export PATH=$PATH:$HUE_HOME/build/env/bin
   ```

   source after adding the entries, source ~/.bashrc

5. Configure Hue (3 files to edit)

`cd $HUE_HOME/desktop/conf`

- hue.ini

  ```
  [desktop]
   server_user=hadoop
   server_group=hadoop
   default_user=hadoop
   default_hdfs_superuser=hadoop
  ```

`cd $HADOOP_CONF_DIR`

- core-site.xml

  ```
  <property>
      <name>hadoop.proxyuser.hadoop.hosts</name>
      <value>*</value>
  </property>
  <property>
      <name>hadoop.proxyuser.hadoop.groups</name>
      <value>*</value>
  </property>
  ```

- hdfs-site.xml

  ```
  <property>
      <name>dfs.webhdfs.enabled</name>
      <value>true</value>
  </property>
  ```

6. Start Hue (Start Hadoop daemons if not already started)

   `nohup supervisor &`

7. Login to Hue Web Interface: http://localhost:8888

   username: hadoop

   password: user_choice

Read hue online: https://riptutorial.com/hadoop/topic/6133/hue

Chapter 6: Introduction to MapReduce

Syntax

- To run the example, the command syntax is:

  ```
  bin/hadoop jar hadoop-*-examples.jar wordcount [-m <#maps>] [-r <#reducers>] <in-dir>
  <out-dir>
  ```

- To copy data into HDFS(from local):

  ```
  bin/hadoop dfs -mkdir <hdfs-dir> //not required in hadoop 0.17.2 and later
  bin/hadoop dfs -copyFromLocal <local-dir> <hdfs-dir>
  ```

Remarks

Word Count program using MapReduce in Hadoop.

Examples

Word Count Program(in Java & Python)

The word count program is like the "Hello World" program in MapReduce.

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.

A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks.

Word Count Example:

WordCount example reads text files and counts how often words occur. The input is text files and the output is text files, each line of which contains a word and the count of how often it occured, separated by a tab.

Each mapper takes a line as input and breaks it into words. It then emits a key/value pair of the word and each reducer sums the counts for each word and emits a single key/value with the word and sum.

As an optimization, the reducer is also used as a combiner on the map outputs. This reduces the amount of data sent across the network by combining each word into a single record.

Word Count Code:

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount {

 public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()) {
            word.set(tokenizer.nextToken());
            context.write(word, one);
        }
    }
 }

 public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        context.write(key, new IntWritable(sum));
    }
 }

 public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

        Job job = new Job(conf, "wordcount");

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);
 }

}
```

To run the example, the command syntax is:

```
bin/hadoop jar hadoop-*-examples.jar wordcount [-m <#maps>] [-r <#reducers>] <in-dir> <out-
dir>
```

All of the files in the input directory (called in-dir in the command line above) are read and the counts of words in the input are written to the output directory (called out-dir above). It is assumed that both inputs and outputs are stored in HDFS.If your input is not already in HDFS, but is rather in a local file system somewhere, you need to copy the data into HDFS using a command like this:

```
bin/hadoop dfs -mkdir <hdfs-dir> //not required in hadoop 0.17.2 and later
bin/hadoop dfs -copyFromLocal <local-dir> <hdfs-dir>
```

Word Count example in Python:

mapper.py

```
import sys
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        print '%s\t%s' % (word, 1)
```

reducer.py

```
import sys
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    # remove leading and trailing whitespaces
    line = line.strip()
    # parse the input we got from mapper.py
    word, count = line.split('\t', 1)
    # convert count (currently a string) to int
    try:
        count = int(count)
    except ValueError:
        # count was not a number, so silently
        # ignore/discard this line
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print '%s\t%s' % (current_word, current_count)
        current_count = count
        current_word = word
if current_word == word:
    print '%s\t%s' % (current_word, current_count)
```

The above program can be run using `cat filename.txt | python mapper.py | sort -k1,1 | python reducer.py`

Read Introduction to MapReduce online: https://riptutorial.com/hadoop/topic/3879/introduction-to-mapreduce

Chapter 7: What is HDFS?

Remarks

A good explanation of HDFS and how it works.

Syntax should contain the commands which maybe use in HDFS.

Examples

HDFS - Hadoop Distributed File System

Hadoop Distributed File System (HDFS) is a Java-based file system that provides scalable and reliable data storage that is designed to span large clusters of commodity servers. HDFS, MapReduce, and YARN form the core of Apache™ Hadoop®.

HDFS is designed to be highly fault-tolerant, which is achieved by saving multiple copies(3 by default) of a given data block across multiple nodes.

Finding files in HDFS

To find a file in the Hadoop Distributed file system:

```
hdfs dfs -ls -R / | grep [search_term]
```

In the above command,

`-ls` is for listing files

`-R` is for recursive(iterate through sub directories)

`/` means from the root directory

`|` to pipe the output of first command to the second

`grep` command to extract matching strings

`[search_term]` file name to be searched for in the list of all files in the hadoop file system.

Alternatively the below command can also be used find and also apply some expressions:

`hadoop fs -find / -name test -print`

Finds all files that match the specified expression and applies selected actions to them. If no path is specified then defaults to the current working directory. If no expression is specified then defaults to -print.

The following primary expressions are recognised:

- `name pattern`
- `iname pattern`

Evaluates as true if the basename of the file matches the pattern using standard file system globbing. If -iname is used then the match is case insensitive.

- `print`
- `print0Always`

Evaluates to true. Causes the current pathname to be written to standard output. If the `-print0` expression is used then an ASCII NULL character is appended.

The following operators are recognised:

```
expression -a expression
expression -and expression
expression expression
```

Blocks and Splits HDFS

1. **Block Size and Blocks in HDFS** : HDFS has the concept of storing data in blocks whenever a file is loaded. Blocks are the physical partitions of data in HDFS (or in any other filesystem, for that matter).

   Whenever a file is loaded onto the HDFS, it is splitted physically (yes, the file is divided) into different parts known as blocks. The number of blocks depend upon the value of `dfs.block.size` in `hdfs-site.xml`

   Ideally, the block size is set to a large value such as 64/128/256 MBs (as compared to 4KBs in normal FS). The default block size value on most distributions of Hadoop 2.x is 128 MB. The reason for a higher block size is because Hadoop is made to deal with PetaBytes of data with each file ranging from a few hundred MegaBytes to the order of TeraBytes.

   Say for example you have a file of size 1024 MBs. if your block size is 128 MB, you will get 8 blocks of 128MB each. This means that your namenode will need to store metadata of `8 x 3 = 24` files (3 being the replication factor).

   Consider the same scenario with a block size of 4 KBs. It will result in `1GB / 4KB = 250000` blocks and that will require the namenode to save the metadata for `750000` blocks for just a 1GB file. Since all these metadata related information is stored in-memory, larger block size is preferred to save that bit of extra load on the NameNode.

   Now again, the block size is not set to an extremely high value like 1GB etc because, ideally, 1 mapper is launched for each block of data. So if you set the block size to 1GB, you might lose parallelism which might result in a slower throughput overall.

2.) **Split Size in HDFS** : Splits in Hadoop Processing are the logical chunks of data. When files are divided into blocks, hadoop doesn't respect any file bopundaries. It just splits the data depending on the block size. Say if you have a file of 400MB, with 4 lines, and each line having 100MB of data, you will get 3 blocks of `128 MB x 3` and `16 MB x 1`. But when input splits are calculated while the prceossing of data, file/record boundaries are kept in mind and in this case we will have 4 input splits of 100 MB each, if you are using, say, `NLineInputFormat`.

Split Size can also be set per job using the property `mapreduce.input.fileinputformat.split.maxsize`

A very good explanation of Blocks vs Splits can be found in this SO Answer/

Read What is HDFS? online: https://riptutorial.com/hadoop/topic/3845/what-is-hdfs-

Credits

S. No	Chapters	Contributors
1	Getting started with hadoop	Ani Menon, Community, franklinsijo, Harinder, ItayB, Sandeep Chatterjee, Shailesh Kumar Dayananda, sunkuet02, Udeet Solanki, Venkata Karthik
2	Debugging Hadoop MR Java code in local eclipse dev environment.	Manish Verma
3	Hadoop commands	Ambrish, Ani Menon, jedijs, philantrovert
4	Hadoop load data	Ani Menon, Backtrack, BruceWayne, NeoWelkin, Tejus Prasad
5	hue	andriosr, franklinsijo
6	Introduction to MapReduce	Ani Menon, Arduino_Sentinel, Tejus Prasad, Udeet Solanki, user3335966
7	What is HDFS?	Ani Menon, NeoWelkin, neuromouse, philantrovert, Suraj Kumar Yadav, Tejus Prasad