This documentation gives complete information about user and admin commands of HDFS.

Hadoop fs is the command to execute file system operations like directory creation, renaming files, copying files etc.

Cmd> hadoop fs –fs [local | <file system URI>]

Specify the file system to use. If not specified, the current configuration is used, taken from the following, in increasing precedence: core-default.xml inside the hadoop jar file, core-site.xml in configuration directory in hadoop home conf/

'Local' means use the local file system as your DFS.

<File system URI> specifies a particular file system to contact.

Example:-

- **hadoop fs -fs local ---> it will use the file system as local file system.**
- **hadoop fs -fs localhost: 9000 ---> It will use the file system as Hadoop distributed file system.**

There is no need to specify, by default it chooses based on configuration files [core-site, hdfs-site, mapred-site xml files]

Cmd> hadoop fs -ls <path>:

List the contents that match the specified file pattern. If path is not specified, the contents of /user/<currentUser> will be listed. Directory entries are of the form dirName full path/<dir> and file entries are of the form fileName(full path) <r n> size where n is the number of replicas specified for the file and size is the size of the file, in bytes.

Example:-

- **hadoop fs -ls / ---> This command lists all the files under the root directory of HDFS**

Cmd> hadoop fs -lsr <path>:

Recursively list the contents that match the specified file pattern. Behaves very similarly to hadoop fs -ls, except that the data is shown for all the entries in the subtree.

Example:-

- **hadoop fs -ls / ---> This command lists all the files Recursively under the root directory of HDFS**

Cmd> hadoop fs -du <path>:

Show the amount of space, in bytes, used by the files that match the specified file pattern. Equivalent to the unix command "du -sb <path>/*" in case of a directory, and to "du -b <path>" in case of a file. The output is in the form name(full path) size (in bytes)

Example:

- **hadoop fs -du / ---> This gives size in bytes and file name (Individual files).**

Cmd> hadoop fs -dus <path>:

Show the amount of space, in bytes, used by the files that match the specified file pattern. Equivalent to the unix command "du -sb" The output is in the form name(full path) size (in bytes)

Example:

- **hadoop fs -dus / ---> This gives the filename and size in bytes (all files)**

Cmd> hadoop fs -mv <src> <dst>:

Move files that match the specified file pattern <src> to a destination <dst>. When moving multiple files, the destination must be a directory. Here the both src and dst are HDFS locations only.

Example:

- **hadoop fs -mv /data/groups /groups ---> This moves the file groups from /data to / and deletes the groups in /data.**

Cmd> hadoop fs -cp <src> <dst>:

Copy files that match the file pattern <src> to a destination. When copying multiple files, the destination must be a directory.

Example:

- **hadoop fs -cp /data/groups /kanna/groups ---> This copies the file groups from /data to /kanna/.**

Cmd> hadoop fs -rm [-skipTrash] <src>:

Delete all files that match the specified file pattern. Equivalent to the Unix command "rm <src>" -skipTrash option bypasses trash, if enabled, and immediately deletes <src>

Example:

- **hadoop fs -rm /groups --> Removes the file name groups from the specified directory.**

Cmd> hadoop fs -rmr [-skipTrash] <src>:

Remove all directories which match the specified file pattern. Equivalent to the Unix command "rm -rf <src>" -skipTrash option bypasses trash, if enabled, and immediately deletes <src>

Example:

- **hadoop fs -rmr /data --> Removes the specified directory.**

Cmd> hadoop -put <localsrc> ... <dst>:

Copy files from the local file system into HDFS.

Example:

- **hadoop fs -put groups /data --> Copy the data from local system to HDFS.**

Cmd> hadoop -copyFromLocal <localsrc> ... <dst>:

Identical to the -put command.

Example:

- **hadoop fs -put groups /data --> Copy the data from local system to HDFS.**

Cmd> hadoop fs -moveFromLocal <localsrc> ... <dst>:

Same as -put, except that the source is deleted after it's copied.

Example:

- **hadoop fs - moveFromLocal groups /data --> Copy the data from local system to HDFS and removes the file in local system.**

Cmd> hadoop fs -get [-ignoreCrc] [-crc] <src> <localdst>:

Copy files that match the file pattern <src> to the local name. <src> is kept. When copying multiple, files, the destination must be a directory.

Example:

- **hadoop fs -get /data/groups groups --> This copies the groups file from HDFS to local file system.**

Cmd> hadoop fs -getmerge <src> <localdst>:

Get all the files in the directories that match the source file pattern and merge and sort them to only one file on local fs. <src> is kept.

Example:

- **hadoop fs -getmerge /data/groups groups --> This copies the all the files from data directory in HDFS to local file system.**

Cmd> hadoop fs -cat <src>:

Fetch all files that match the file pattern <src> and display their content on stdout.

Example:

- **hadoop fs -cat /data/groups --> It displays the content of groups file in stdout**

Cmd> hadoop fs -copyToLocal [-ignoreCrc] [-crc] <src> <localdst>:

Identical to the -get command.

Example:

- **hadoop fs - copyToLocal /data/groups groups --> This copies the groups file from HDFS to local file system**

Cmd> hadoop fs -moveToLocal <src> <localdst>:

Not implemented yet

Cmd> hadoop fs -mkdir <path>:

Create a directory in specified location.

Example:

- **hadoop fs -mkdir /training --> it creates a directory training in HDFS.**

Cmd> hadoop fs -setrep [-R] [-w] <rep> <path/file>:

Set the replication level of a file. The -R flag requests a recursive change of replication level for an entire tree.

Example:

- **hadoop fs -setrep 2 /training/input --> It changes replication factor of input file to 2 from the old replication factor.**
- **hadoop fs -setrep -R 2 /training/ --> It changes replication factor of all files to 2 from the old replication factor.**

Cmd> hadoop fs -tail [-f] <file>:

Show the last 1KB of the file. The -f option shows apended data as the file grows.

Example:

- **hadoop fs -tail /training/input --> It displays the last 1KB content of input file**

Cmd> hadoop fs -touchz <path>:

It is used to create an empty file in specified directory.

Example:

- **hadoop fs -touchz /training/name --> It creates an empty file name in training directory.**

Cmd> hadoop fs -test -[ezd] <path>:

If file { exists, has zero length, is a directory then return 0, else return 1.

Example:

- **hadoop fs -test -e /training/name**

Cmd> hadoop fs -text <src>:

Takes a source file and outputs the file in text format. The allowed formats are zip and TextRecordInputStream.

Example:

- **hadoop fs -text /training/binaryfile --> It prints the binary file in text format.**

Cmd> hadoop fs -stat [format] <path>:

Print statistics about the file/directory at <path> in the specified format. Format accepts filesize in blocks (%b), filename (%n), block size (%o), replication (%r), modification date (%y, %Y)

Example:

- **hadoop fs -stat [%b,%n,%o,%r] /training --> It prints the stats about the file/directory based on the format.**

Cmd> hadoop fs -chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...

Changes permissions of a file. This works similar to shell's chmod with a few exceptions.

-R modifies the files recursively. This is the only option currently supported.

MODE Mode is same as mode used for chmod shell command.

Only letters recognized are 'rwxX'. E.g. a+r,g-w,+rwx,o=r

OCTALMODE Mode specifed in 3 digits. Unlike shell command, this requires all three digits.

E.g. 754 is same as u=rwx,g=rx,o=r

If none of 'augo' is specified, 'a' is assumed and unlike

shell command, no umask is applied.

Example:

- **hadoop fs -chmod -R 777 /data/daily --> It changes the permissions to 777 means read/write for owner, group, and rest also.**

Cmd> hadoop fs -chown [-R] [OWNER][:[GROUP]] PATH...

Changes owner and group of a file.

This is similar to shell's chown with a few exceptions.

-R modifies the files recursively. This is the only option currently supported.

If only owner or group is specified then only owner or group is modified.

The owner and group names may only cosists of digits, alphabet,

and any of '-_.@/' i.e. [-_.@/a-zA-Z0-9]. The names are case sensitive.

WARNING: Avoid using '.' to separate user name and group though

Linux allows it. If user names have dots in them and you are

using local file system, you might see surprising results since

shell command 'chown' is used for local files.

Example:

- **hadoop fs -chown -R naga:supergroup /data/daily --> This will change the user and group of the file /data/daily**

Cmd> hadoop fs -chgrp [-R] GROUP PATH...

This is equivalent to -chown ... :GROUP ...

Cmd> hadoop fs -count[-q] <path>:

Count the number of directories, files and bytes under the paths that match the specified file pattern. The output columns are: DIR_COUNT FILE_COUNT CONTENT_SIZE FILE_NAME

Example:

- **hadoop fs -count /data/**

Cmd> hadoop fs -help [cmd]:

Displays help for given command or all commands if none is specified.

Example:

- **hadoop fs -help ls --> it gives the usage information about command ls.**

Cmd> hadoop balancer [-threshold <threshold>] percentage of disk capacity

Example:

- **hadoop balancer --> It does the cluster balancing**

Cmd> hadoop fsck <path> [-move | -delete | -openforwrite] [-files [-blocks [-locations | -racks]]]

<path> start checking from this path

-move move corrupted files to /lost+found

-delete delete corrupted files

-files print out files being checked

-openforwrite print out files opened for write

-blocks print out block report

-locations print out locations for every block

-racks print out network topology for data-node locations

By default fsck ignores files opened for write, use –open for write to report such files. They are usually tagged CORRUPT or HEALTHY depending on their block allocation status

Example:

- **hadoop fsck / --> It provides the healthiness of the hadoop cluster.**

Cmd> hadoop distcp [OPTIONS] <srcurl>* <desturl>

OPTIONS:

-p[rbugp] Preserve status

r: replication number

b: block size

u: user

g: group

p: permission

-p alone is equivalent to -prbugp

-i Ignore failures

-log <logdir> Write logs to <logdir>

-m <num_maps> Maximum number of simultaneous copies

-overwrite Overwrite destination

-update Overwrite if src size different from dst size

-skipcrccheck Do not use CRC check to determine if src is

different from dest. Relevant only if -update is specified

-f <urilist_uri> Use list at <urilist_uri> as src list

-filelimit <n> Limit the total number of files to be <= n

-sizelimit <n> Limit the total size to be <= n bytes

-delete Delete the files existing in the dst but not in src

-mapredSslConf <f> Filename of SSL configuration for mapper task

Example:

- **hadoop distcp -p -overwrite "hdfs://localhost:9000/training/bar" "hdfs://anotherhostname:8020/user/foo" --> It loads data from one cluster to another cluster in parallel. It over writes if file is there.**
- **hadoop distcp -p -update "file:///filename" "hdfs://localhost:9000/training/filename" --> It loads data from local system to hadoop cluster in parallel. It updates if file is there.**

Cmd> hadoop archive -archiveName NAME -p <parent path> <src>* <dest>

Example: hadoop archive –archiveName myfile.har –p /data /data/files /archive/

---

hadoop dfsadmin is the command to execute DFS administrative commands.

Example:

- **hadoop dfsadm <command name>**

Cmd> hadoop dfsadmin -report:

Reports basic filesystem information and statistics.

Example:

- **hadoop dfsadmin -report**

Cmd> hadoop dfsadmin -safemode <enter|leave|get|wait>:

Safe mode maintenance command.

Safe mode is a Namenode state in which it

1. does not accept changes to the name space (read-only)
2. does not replicate or delete blocks.

Safe mode is entered automatically at Namenode startup, and leaves safe mode automatically when the configured minimum percentage of blocks satisfies the minimum replication condition. Safe mode can also be entered manually, but then it can only be turned off manually as well.

Example:

- **hadoop dfsadmin -safemode get --> To get status of safe mode**
- **hadoop dfsadmin -safemode enter --> To enter safe mode**
- **hadoop dfsadmin -safemode leave --> To leave the safemode**

Cmd> hadoop dfsadmin -saveNamespace:

Save current namespace into storage directories and reset edits log.Requires superuser permissions and safe mode.

Cmd> hadoop dfsadmin -refreshNodes:

Updates the set of hosts allowed to connect to namenode.

Re-reads the config file to update values defined by dfs.hosts and dfs.host.exclude and reads the entires (hostnames) in those files.

Each entry not defined in dfs.hosts but in

dfs.hosts.exclude is decommissioned. Each entry defined

in dfs.hosts and also in dfs.host.exclude is stopped from

decommissioning if it has aleady been marked for decommission.

Entires not present in both the lists are decommissioned.

Cmd> hadoop dfsadmin -finalizeUpgrade:

Finalize upgrade of HDFS. Datanodes delete their previous version working directories, followed by Namenode doing the same. This completes the upgrade process.

Cmd> hadoop dfsadmin -upgradeProgress <status|details|force>:

Request current distributed upgrade status, a detailed status or force the upgrade to proceed.

Cmd> hadoop dfsadmin -metasave <filename>:

Save Namenode's primary data structures

to <filename> in the directory specified by hadoop.log.dir property.

<filename> will contain one line for each of the following

1. Datanodes heart beating with Namenode
2. Blocks waiting to be replicated
3. Blocks currrently being replicated
4. Blocks waiting to be deleted

Cmd> hadoop dfsadmin -setQuota <quota> <dirname>...<dirname>:

Set the quota <quota> for each directory <dirName>.

The directory quota is a long integer that puts a hard limit

on the number of names in the directory tree

Best effort for the directory, with faults reported if

1. N is not a positive integer, or

2. user is not an administrator, or

3. the directory does not exist or is a file, or

Cmd> hadoop dfsadmin -clrQuota <dirname>...<dirname>:

Clear the quota for each directory <dirName>.

Best effort for the directory. with fault reported if

1. the directory does not exist or is a file, or

2. user is not an administrator.

It does not fault if the directory has no quota.

Cmd> hadoop dfsadmin -setSpaceQuota <quota> <dirname>...<dirname>:

Set the disk space quota <quota> for each directory <dirName>.

The space quota is a long integer that puts a hard limit

on the total size of all the files under the directory tree.

The extra space required for replication is also counted. E.g.

a 1GB file with replication of 3 consumes 3GB of the quota.

Quota can also be speciefied with a binary prefix for terabytes,

petabytes etc (e.g. 50t is 50TB, 5m is 5MB, 3p is 3PB).

Best effort for the directory, with faults reported if

1. N is not a positive integer, or

2. user is not an administrator, or

3. the directory does not exist or is a file, or

Cmd> hadoop dfsadmin -clrSpaceQuota <dirname>...<dirname>:

Clear the disk space quota for each directory <dirName>.

Best effort for the directory. with fault reported if

1. the directory does not exist or is a file, or

2. user is not an administrator.

It does not fault if the directory has no quota.

Cmd> hadoop dfsadmin -refreshServiceAcl:

Reload the service-level authorization policy file. Namenode will reload the authorization policy file.

Cmd> hadoop dfsadmin -refreshUserToGroupsMappings:

Refresh user-to-groups mappings

Cmd> hadoop dfsadmin -refreshSuperUserGroupsConfiguration:

Refresh superuser proxy groups mappings

Cmd> hadoop dfsadmin -setBalancerBandwidth <bandwidth>:

Changes the network bandwidth used by each datanode during

HDFS block balancing.

<bandwidth> is the maximum number of bytes per second

that will be used by each datanode. This value overrides

the dfs.balance.bandwidthPerSec parameter.

NOTE: The new value is not persistent on the DataNode

Cmd> hadoop -help [cmd]:

Displays help for the given command or all commands if none is specified.

Example:

- **hadoop -help namenode --> It gives the complete information about the command namenode.**