Hadoop Shell Commands

Table of contents

1. DFShell

The HDFS shell is invoked by `bin/hadoop dfs <args>`. All the HDFS shell commands take path URIs as arguments. The URI format is *scheme://autority/path*. For HDFS the scheme is *hdfs*, and for the local filesystem the scheme is *file*. The scheme and authority are optional. If not specified, the default scheme specified in the configuration is used. An HDFS file or directory such as */parent/child* can be specified as *hdfs://namenode:namenodeport/parent/child* or simply as */parent/child* (given that your configuration is set to point to *namenode:namenodeport*). Most of the commands in HDFS shell behave like corresponding Unix commands. Differences are described with each of the commands. Error information is sent to *stderr* and the output is sent to *stdout*.

2. cat

`Usage: hadoop dfs -cat URI [URI …]`

Copies source paths to *stdout*.

Example:

- `hadoop dfs -cat hdfs://host1:port1/file1 hdfs://host2:port2/file2`
- `hadoop dfs -cat file:///file3 /user/hadoop/file4`

Exit Code:
`Returns 0 on success and -1 on error.`

3. chgrp

`Usage: hadoop dfs -chgrp [-R] GROUP URI [URI …]`

Change group association of files. With `-R`, make the change recursively through the directory structure. The user must be the owner of files, or else a super-user. Additional information is in the Permissions User Guide.

4. chmod

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

Change the permissions of files. With `-R`, make the change recursively through the directory structure. The user must be the owner of the file, or else a super-user. Additional information

is in the Permissions User Guide.

5. chown

```
Usage: hadoop dfs -chown [-R] [OWNER][:[GROUP]] URI [URI ]
```

Change the owner of files. With `-R`, make the change recursively through the directory structure. The user must be a super-user. Additional information is in the Permissions User Guide.

6. copyFromLocal

```
Usage: hadoop dfs -copyFromLocal <localsrc> URI
```

Similar to **put** command, except that the source is restricted to a local file reference.

7. copyToLocal

```
Usage: hadoop dfs -copyToLocal [-ignorecrc] [-crc] URI
<localdst>
```

Similar to **get** command, except that the destination is restricted to a local file reference.

8. cp

```
Usage: hadoop dfs -cp URI [URI …] <dest>
```

Copy files from source to destination. This command allows multiple sources as well in which case the destination must be a directory.
Example:

- `hadoop dfs -cp /user/hadoop/file1 /user/hadoop/file2`
- `hadoop dfs -cp /user/hadoop/file1 /user/hadoop/file2 /user/hadoop/dir`

Exit Code:

```
Returns 0 on success and -1 on error.
```

9. du

```
Usage: hadoop dfs -du URI [URI …]
```

Displays aggregate length of files contained in the directory or the length of a file in case its

just a file.
Example:
`hadoop dfs -du /user/hadoop/dir1 /user/hadoop/file1 hdfs://host:port/user/hadoop/dir1`
Exit Code:
`Returns 0 on success and -1 on error.`

10. dus

`Usage: hadoop dfs -dus <args>`

Displays a summary of file lengths.

11. expunge

`Usage: hadoop dfs -expunge`

Empty the Trash. Refer to HDFS Design for more information on Trash feature.

12. get

`Usage: hadoop dfs -get [-ignorecrc] [-crc] <src> <localdst>`

Copy files to the local file system. Files that fail the CRC check may be copied with the `-ignorecrc` option. Files and CRCs may be copied using the `-crc` option.

Example:

- `hadoop dfs -get /user/hadoop/file localfile`
- `hadoop dfs -get hdfs://host:port/user/hadoop/file localfile`

Exit Code:

`Returns 0 on success and -1 on error.`

13. getmerge

`Usage: hadoop dfs -getmerge <src> <localdst> [addnl]`

Takes a source directory and a destination file as input and concatenates files in src into the destination local file. Optionally `addnl` can be set to enable adding a newline character at the end of each file.

14. ls

```
Usage: hadoop dfs -ls <args>
```

For a file returns stat on the file with the following format:
`filename <number of replicas> filesize modification_date modification_time permissions userid groupid`
For a directory it returns list of its direct children as in unix. A directory is listed as:
`dirname <dir> modification_time modification_time permissions userid groupid`
Example:
`hadoop dfs -ls /user/hadoop/file1 /user/hadoop/file2 hdfs://host:port/user/hadoop/dir1 /nonexistentfile`
Exit Code:
`Returns 0 on success and -1 on error.`

15. lsr

```
Usage: hadoop dfs -lsr <args>
```
Recursive version of `ls`. Similar to Unix `ls -R`.

16. mkdir

```
Usage: hadoop dfs -mkdir <paths>
```

Takes path uri's as argument and creates directories. The behavior is much like unix mkdir -p creating parent directories along the path.

Example:

- `hadoop dfs -mkdir /user/hadoop/dir1 /user/hadoop/dir2`
- `hadoop dfs -mkdir hdfs://host1:port1/user/hadoop/dir hdfs://host2:port2/user/hadoop/dir`

Exit Code:

`Returns 0 on success and -1 on error.`

17. movefromLocal

```
Usage: dfs -moveFromLocal <src> <dst>
```

Displays a "not implemented" message.

18. mv

```
Usage: hadoop dfs -mv URI [URI …] <dest>
```

Moves files from source to destination. This command allows multiple sources as well in which case the destination needs to be a directory. Moving files across filesystems is not permitted.
Example:

- `hadoop dfs -mv /user/hadoop/file1 /user/hadoop/file2`
- `hadoop dfs -mv hdfs://host:port/file1 hdfs://host:port/file2 hdfs://host:port/file3 hdfs://host:port/dir1`

Exit Code:

```
Returns 0 on success and -1 on error.
```

19. put

```
Usage: hadoop dfs -put <localsrc> <dst>
```

Copy src from local file system to the destination filesystem. Also reads input from stdin and writes to destination filesystem.

- `hadoop dfs -put localfile /user/hadoop/hadoopfile`
- `hadoop dfs -put localfile hdfs://host:port/hadoop/hadoopfile`
- `hadoop dfs -put - hdfs://host:port/hadoop/hadoopfile`
  Reads the input from stdin.

Exit Code:

```
Returns 0 on success and -1 on error.
```

20. rm

```
Usage: hadoop dfs -rm URI [URI …]
```

Delete files specified as args. Only deletes non empty directory and files. Refer to rmr for recursive deletes.
Example:

- `hadoop dfs -rm hdfs://host:port/file /user/hadoop/emptydir`

Exit Code:

```
Returns 0 on success and -1 on error.
```

21. rmr

```
Usage: hadoop dfs -rmr URI [URI …]
```

Recursive version of delete.
Example:

- `hadoop dfs -rmr /user/hadoop/dir`
- `hadoop dfs -rmr hdfs://host:port/user/hadoop/dir`

Exit Code:

```
Returns 0 on success and -1 on error.
```

22. setrep

```
Usage: hadoop dfs -setrep [-R] <path>
```

Changes the replication factor of a file. -R option is for recursively increasing the replication factor of files within a directory.

Example:

- `hadoop dfs -setrep -w 3 -R /user/hadoop/dir1`

Exit Code:

```
Returns 0 on success and -1 on error.
```

23. stat

```
Usage: hadoop dfs -stat URI [URI …]
```

Returns the stat information on the path.

Example:

- `hadoop dfs -stat path`

Exit Code:
```
Returns 0 on success and -1 on error.
```

24. tail

```
Usage: hadoop dfs -tail [-f] URI
```

Displays last kilobyte of the file to stdout. -f option can be used as in Unix.

Example:

- `hadoop dfs -tail pathname`

Exit Code:
`Returns 0 on success and -1 on error.`

25. test

```
Usage: hadoop dfs -test -[ezd] URI
```

Options:
-e check to see if the file exists. Return 0 if true.
-z check to see if the file is zero length. Return 0 if true
-d check return 1 if the path is directory else return 0.

Example:

- `hadoop dfs -test -e filename`

26. text

```
Usage: hadoop dfs -text <src>
```

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

27. touchz

```
Usage: hadoop dfs -touchz URI [URI …]
```

Create a file of zero length.

Example:

- `hadoop -touchz pathname`

Exit Code:
`Returns 0 on success and -1 on error.`