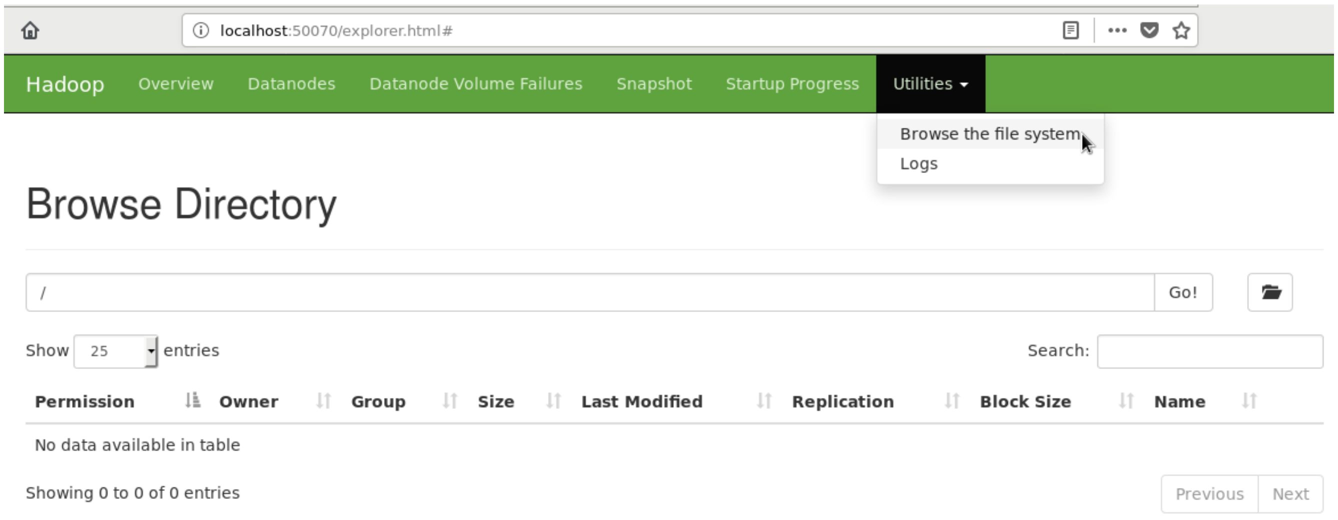
Experiment - 3
Lab Manual

Aim : "Interfacing to Distributed File System – HDFS commands and Web GUI file system browser usage"

Hadoop monitoring and HDFS Web GUI can be accessed by the URL http://localhost:50070/. Explore this Web GUI.

Go to 'Browse the file system' under 'Utilities' section. If no data is dumped in HDFS yet, it should display the same message.



Browse Directory

/

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
No data available in table							

Showing 0 to 0 of 0 entries

Documentation/Help

Seek help using hdfs dfs –help ls

will provide help about ls command and options. Just to see documentation about command you don't need to have hdfs daemons started. Of course to practice, you will first need to have hdfs started and verified that the NameNode, DataNode and SecondaryNameNode daemons are up and running.

Sample command usage:

Create a directory named 'user' and create another directory named 'hadoop' inside 'user' directory:
To verify

```
[hadoop@hadoop-clone hadoop]$ hdfs dfs -ls /user/hadoop
```

To create complete path

```
[hadoop@hadoop-clone hadoop]$ hdfs dfs -mkdir -p /user/hadoop
```

Browse the file system again to see these directories. Both CLI and GUI ways.

Know that name of the user here assumed 'hadoop' and hence above location will be considered hdfs home location of user named 'hadoop'. You may use relative path to hdfs user home as third approach to referring files or directories. Ideally hadoop adminstrator shall do this job of creating home for all users. The user home concept is similar to Linux user home but the location is like /user/nameofuser.

The File System (FS) shell includes various shell-like commands that directly interact with the Hadoop Distributed File System (HDFS) as well as other file systems that Hadoop supports, such as Local FS, HFTP FS, S3 FS, and others. The FS shell is invoked by 'hdfs dfs <args>'.

All FS shell commands take path URIs as arguments. The URI format is scheme://authority/path. For HDFS the scheme is hdfs, and for the Local FS 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://namenodehost/parent/child or simply as /parent/child (given that your configuration is set to point to hdfs://namenodehost). Most of the commands in FS shell behave like corresponding Unix commands.

1. mkdir

Usage: hdfs dfs -mkdir [-p] <paths>

Takes path uri’s as argument and creates directories.

Options:

- The -p option behavior is much like Unix mkdir -p, creating parent directories along the path.

Example:

- hdfs dfs -mkdir /user/hadoop/dir1 /user/hadoop/dir2
- hdfs dfs -mkdir hdfs://nn1.example.com/user/hadoop/dir hdfs://nn2.example.com/user/hadoop/dir

Exit Code:

Returns 0 on success and -1 on error.

2. ls

Usage: hdfs dfs -ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] <args>

Options:

- -C: Display the paths of files and directories only.
- -d: Directories are listed as plain files.
- -h: Format file sizes in a human-readable fashion (eg 64.0m instead of 67108864).
- -q: Print ? instead of non-printable characters.
- -R: Recursively list subdirectories encountered.
- -t: Sort output by modification time (most recent first).
- -S: Sort output by file size.
- -r: Reverse the sort order.
- -u: Use access time rather than modification time for display and sorting.

For a file ls returns stat on the file with the following format:

permissions number_of_replicas userid groupid filesize modification_date modification_time filename

For a directory it returns list of its direct children as in Unix. A directory is listed as:

permissions userid groupid modification_date modification_time dirname

Files within a directory are order by filename by default.

Example:

- hdfs dfs -ls /user/hadoop/file1

Exit Code:

Returns 0 on success and -1 on error.

3. put

Usage: hdfs dfs -put [-f] [-p] [-l] [-d] [- | <localsrc1> ..]. <dst>

Copy single src, or multiple srcs from local file system to the destination file system. Also reads input from stdin and writes to destination file system if the source is set to "-"

Copying fails if the file already exists, unless the -f flag is given.

Options:

- -p : Preserves access and modification times, ownership and the permissions. (assuming the permissions can be propagated across filesystems)
- -f : Overwrites the destination if it already exists.
- -l : Allow DataNode to lazily persist the file to disk, Forces a replication factor of 1. This flag will result in reduced durability. Use with care.
- -d : Skip creation of temporary file with the suffix ._COPYING_.

Examples:

- hdfs dfs -put localfile /user/hadoop/hadoopfile
- hdfs dfs -put -f localfile1 localfile2 /user/hadoop/hadoopdir
- hdfs dfs -put -d localfile hdfs://nn.example.com/hadoop/hadoopfile
- hdfs dfs -put - hdfs://nn.example.com/hadoop/hadoopfile Reads the input from stdin.

Exit Code:

Returns 0 on success and -1 on error.

4. rm

Usage: hdfs dfs -rm [-f] [-r |-R] [-skipTrash] [-safely] URI [URI ...]

Delete files specified as args.

If trash is enabled, file system instead moves the deleted file to a trash directory. Currently, the trash feature is disabled by default. User can enable trash by setting a value greater than zero for parameter fs.trash.interval (in core-site.xml).

Options:

- The -f option will not display a diagnostic message or modify the exit status to reflect an error if the file does not exist.

- The -R option deletes the directory and any content under it recursively.
- The -r option is equivalent to -R.
- The -skipTrash option will bypass trash, if enabled, and delete the specified file(s) immediately. This can be useful when it is necessary to delete files from an over-quota directory.
- The -safely option will require safety confirmation before deleting directory with total number of files greater than hadoop.shell.delete.limit.num.files (in core-site.xml, default: 100). It can be used with -skipTrash to prevent accidental deletion of large directories. Delay is expected when walking over large directory recursively to count the number of files to be deleted before the confirmation.

Example:

- hdfs dfs -rm hdfs://nn.example.com/file /user/hadoop/emptydir

Exit Code:

Returns 0 on success and -1 on error.

5. rmdir

Usage: hadoop fs -rmdir [--ignore-fail-on-non-empty] URI [URI ...]

Delete a directory.

Options:

- --ignore-fail-on-non-empty: When using wildcards, do not fail if a directory still contains files.

Example:

- hdfs dfs -rmdir /user/hadoop/emptydir

6. copyFromLocal

Usage: hdfs dfs -copyFromLocal <localsrc> URI

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

Options:

- -p : Preserves access and modification times, ownership and the permissions. (assuming the permissions can be propagated across filesystems)
- -f : Overwrites the destination if it already exists.
- -l : Allow DataNode to lazily persist the file to disk, Forces a replication factor of 1. This flag will result in reduced durability. Use with care.

- -d : Skip creation of temporary file with the suffix ._COPYING_.

7. copyToLocal

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

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

8. cp

Usage: hdfs dfs -cp [-f] [-p | -p[topax]] 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.

'raw.*' namespace extended attributes are preserved if (1) the source and destination filesystems support them (HDFS only), and (2) all source and destination pathnames are in the /.reserved/raw hierarchy. Determination of whether raw.* namespace xattrs are preserved is independent of the -p (preserve) flag.

Options:

- The -f option will overwrite the destination if it already exists.
- The -p option will preserve file attributes [topx] (timestamps, ownership, permission, ACL, XAttr). If -p is specified with no *arg*, then preserves timestamps, ownership, permission. If -pa is specified, then preserves permission also because ACL is a super-set of permission. Determination of whether raw namespace extended attributes are preserved is independent of the -p flag.

Example:

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

Exit Code:

Returns 0 on success and -1 on error.

9. get

Usage: hdfs dfs -get [-ignorecrc] [-crc] [-p] [-f] <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:

- hdfs dfs -get /user/hadoop/file localfile

- hdfs dfs -get hdfs://nn.example.com/user/hadoop/file localfile

Exit Code:

Returns 0 on success and -1 on error.

Options:

- -p : Preserves access and modification times, ownership and the permissions. (assuming the permissions can be propagated across filesystems)
- -f : Overwrites the destination if it already exists.
- -ignorecrc : Skip CRC checks on the file(s) downloaded.
- -crc: write CRC checksums for the files downloaded.

10. moveFromLocal

Usage: hdfs dfs -moveFromLocal <localsrc> <dst>

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

11. mv

Usage: hdfs 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 file systems is not permitted.

Example:

- hdfs dfs -mv /user/hadoop/file1 /user/hadoop/file2
- hdfs dfs -mv hdfs://nn.example.com/file1 hdfs://nn.example.com/file2 hdfs://nn.example.com/file3 hdfs://nn.example.com/dir1

Exit Code:

Returns 0 on success and -1 on error.

References :

1. https://hadoop.apache.org

2. https://hadoop.apache.org/docs/r2.9.2/hadoop-project-dist/hadoop-common/FileSystemShell.html

Exercises:

1. Create a directory named 'Test' from HDFS web UI in '/user/hadoop'. It should give error message. Read this message and solve the error.
2. Create a text file in local file system and put the same file in HDFS using appropriate command.
3. Display the contents of this file of HDFS on terminal.
4. Download this file of HDFS on local machine and change its contents. Upload this modified file again on HDFS and verify its contents. Also make sure that this file should be deleted from local machine automatically.
5. Find out a CLI way to display all file system commands supported by hdfs. Record the list.

p.s. “hadoop fs ...” and “hdfs dfs ...” cab be used interchangeably but the “hadoop fs ...” is deprecated.

Last modified 20-July-2020.