Safemode for Namenode is a read-only mode for the HDFS cluster, where it does not allow any modifications to file system or Blocks HDFS cluster is in safemode state during start up because the cluster needs to validate all the blocks and their locations.
Once validated, safemode is then disabled.

Other possible reasons for namenode going to safemode are-

1) Unreported datanodes which will cause missing blocks and if the missing blocks are greater than a limit, namenode will go to safemode
2) If the storage of some datanodes got cleared accidently and if no blocks are available in the cluster for those data, namenode will go to safemode, because the metadata corresponding to those blocks will be present in the namenode and it will wait for those blocks to report. Till that time it can't provide this data to users, so it will be in safemode
3) If the storage of namenode is full. Then namenode will go to safemode
4) If the namenode lacks physical memory, it will enter into safemode
5) If the cluster storage is full, namenode will enter into safemode

Switch to hdfs user using below command –
su -hdfs
Execute below commands as hdfs user –
Command to enter safe mode –
`hdfs dfsadmin -safemode enter`
Command to leave safe mode –
`hdfs dfsadmin -safemode leave`
Command to get safe mode –
`hdfs dfsadmin -safemode get`

Safe Mode in Hadoop is a maintenance state of Name Node during which Name Node doesn’t allow any changes to the file system. During safe mode Hadoop is read only and doesn’t allow the replication or deletion of blocks.

• 1.Name Node automatically enters into safe mode when its startup.

• 2 . To leave safe mode , name node need to collect reports for at least a specified threshold percentage of blocks and these blocks should satisfy min replication condition.

• 3. Once threshold is reached , safe mode will be extended for a configured amount of time to get remaining data nodes to check in. so that name node can starts replicating missing data blocks or deleting over replicated blocks.

• 4. Once all the blocks reports are collected and namenode done with block replication and adjustment it will leave the safe mode automatically.

Even though Name node can also be enter manually to do some administration activities by using below HDFS command:
1. Command to enter into safe mode :-

```
hdfs dfsadmin -safemode enter
```

2. Command to leave safe mode :-

```
hdfs dfsadmin –safemode leave
```

3. Command to get the current status of safemode:-

```
hdfs dfsadmin -safemode get
```

HDFS Commands

•

HDFS is the primary or major component of the Hadoop ecosystem which is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files. To use the HDFS commands, first you need to start the Hadoop services using the following command:

```
sbin/start-all.sh
```

To check the Hadoop services are up and running use the following command:

```
jps
```

Commands:

1. **ls:** This command is used to list all the files. Use *lsr* for recursive approach. It is useful when we want a hierarchy of a folder.
   Syntax:
   ```
   bin/hdfs dfs -ls  <path>
   ```
   Example:
   ```
   bin/hdfs dfs -ls /
   ```
   It will print all the directories present in HDFS. bin directory contains executables so, *bin/hdfs* means we want the executables of hdfs particularly *dfs*(Distributed File System) commands.

2. **mkdir**: To create a directory. In Hadoop *dfs* there is no home directory by default. So let’s first create it.
   Syntax:
   ```
   bin/hdfs dfs -mkdir <folder name>


   creating home directory:


   hdfs/bin -mkdir /user

   hdfs/bin  -mkdir  /user/username  ->  write  the  username  of  your
   computer
   ```
   Example:
   ```
   bin/hdfs dfs -mkdir  /punam  =>  '/' means absolute path

   bin/hdfs dfs -mkdir  punam2  =>   Relative path -> the folder will
   be

                                     created  relative  to  the  home
   directory.
   ```

3. **touchz**: It creates an empty file.
   Syntax:
   ```
   bin/hdfs dfs  -touchz  <file_path>
   ```
   Example:
   ```
   bin/hdfs dfs -touchz  /punam/myfile.txt
   ```

4. **copyFromLocal (or) put:** To copy files/folders from local file system to hdfs store. This is the most important command. Local filesystem means the files present on the OS.
   Syntax:
   ```
   bin/hdfs dfs -copyFromLocal <local file path>  <dest(present on hdfs)>
   ```
   Example: Let's suppose we have a file *AI.txt* on Desktop which we want to copy to folder *punam* present on hdfs.
   ```
   bin/hdfs dfs -copyFromLocal ../Desktop/AI.txt /punam


   (OR)


   bin/hdfs dfs -put ../Desktop/AI.txt /punam
   ```

5. **cat:** To print file contents.
   Syntax:
   ```
   bin/hdfs dfs -cat <path>
   ```
   Example:
   ```
   // print the content of AI.txt present

   // inside punam folder.

   bin/hdfs dfs -cat /punam/AI.txt ->
   ```

6. **copyToLocal (or) get:** To copy files/folders from hdfs store to local file system.
   Syntax:
   ```
   bin/hdfs dfs -copyToLocal  <<srcfile(on hdfs)> <local file dest>
   ```
   Example:
   ```
   bin/hdfs dfs -copyToLocal  /punam   ../Desktop/hero


   (OR)


   bin/hdfs dfs -get /punam/myfile.txt  ../Desktop/hero
   ```
   myfile.txt from *punam* folder will be copied to folder *hero* present on *Desktop*.

   Note: Observe that we don't write *bin/hdfs* while checking the things present on local filesystem.

7. **moveFromLocal:** This command will move file from local to hdfs.
   Syntax:
   ```
   bin/hdfs dfs -moveFromLocal <local src>   <dest(on hdfs)>
   ```
   Example:
   ```
   bin/hdfs dfs -moveFromLocal  ../Desktop/cutAndPaste.txt   /punam
   ```

8. **cp:** This command is used to copy files within hdfs. Lets copy folder *punam* to *punam_copied*.
   Syntax:
   ```
   bin/hdfs dfs -cp  <src(on hdfs)>  <dest(on hdfs)>
   ```
   Example:
   ```
   bin/hdfs -cp /punam  /punam_copied
   ```

9. **mv:** This command is used to move files within hdfs. Lets cut-paste a file *myfile.txt* from *punam* folder to *punam_copied*.
   Syntax:
   ```
   bin/hdfs dfs -mv  <src(on hdfs)> <src(on hdfs)>
   ```
   Example:
   ```
   bin/hdfs  -mv  /punam/myfile.txt  /punam_copied
   ```

10. **rmr:** This command deletes a file from HDFS *recursively*. It is very useful command when you want to delete a *non-empty directory*.

Syntax:

```
bin/hdfs dfs -rmr <filename/directoryName>
```

Example:

```
bin/hdfs dfs -rmr  /punam_copied -> It will delete all the content inside the

                                    directory then the directory itself.
```

11. **du:** It will give the size of each file in directory.

Syntax:

```
bin/hdfs dfs -du  <dirName>
```

Example:

```
bin/hdfs dfs -du /punam
```

12. **dus:**: This command will give the total size of directory/file.

Syntax:

```
bin/hdfs dfs -dus  <dirName>
```

Example:

```
bin/hdfs dfs -dus /punam
```

13. **stat:** It will give the last modified time of directory or path. In short it will give stats of the directory or file.

Syntax:

```
bin/hdfs  dfs -stat    <hdfs file>
```

Example:

```
bin/hdfs dfs -stat /punam
```

14. **setrep:** This command is used to change the replication factor of a file/directory in HDFS. By default it is 3 for anything which is stored in HDFS (as set in hdfs *core-site.xml*).

    Example 1: To change the replication factor to 6 for *punam.txt* stored in HDFS.

    ```
    bin/hdfs dfs -setrep -R -w 6 punam.txt
    ```

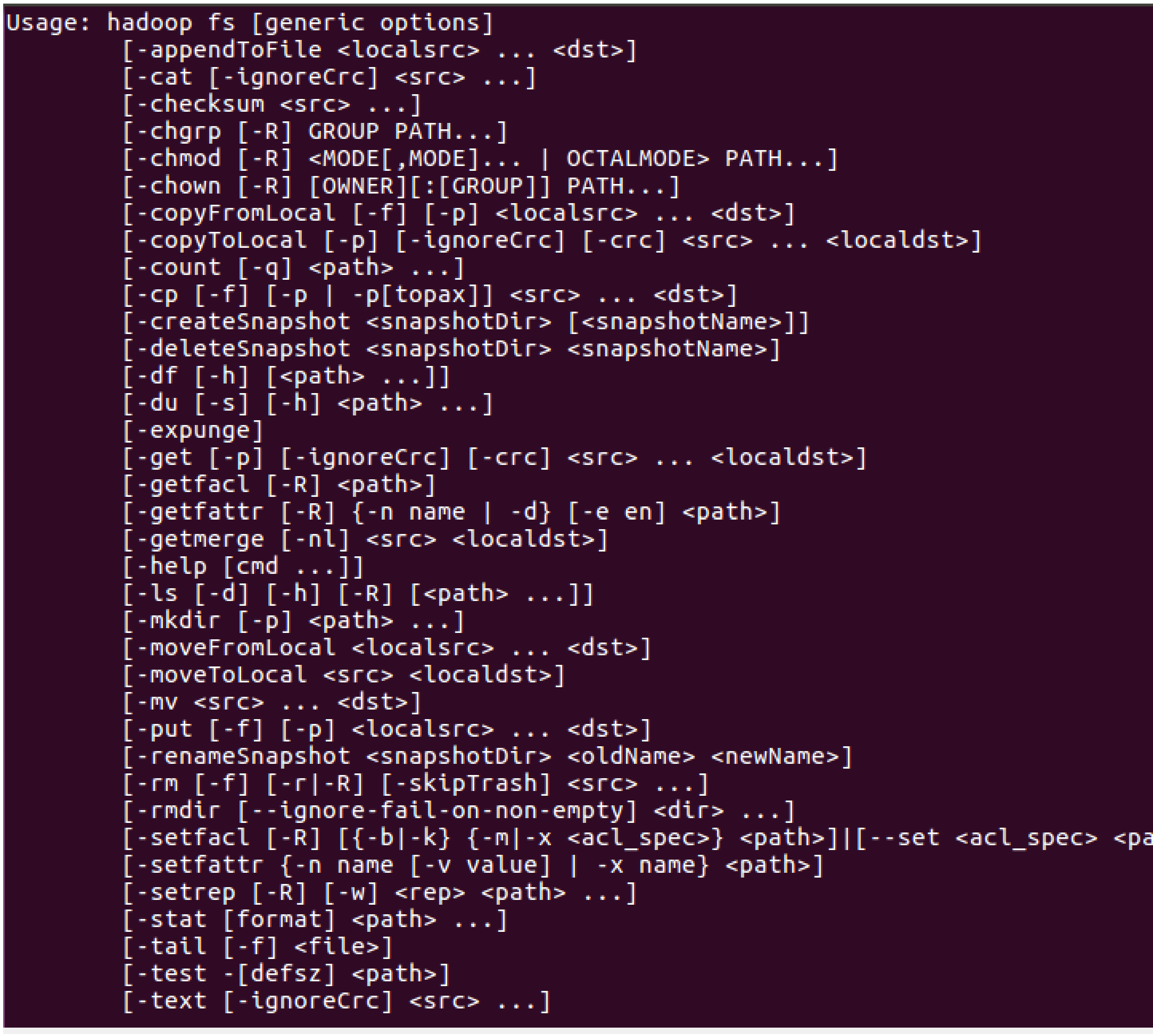
    Example 2: To change the replication factor to 4 for a directory *punamInput* stored in HDFS.

    ```
    bin/hdfs dfs -setrep -R  4 /punam
    ```

    Note: The **-w** means wait till the replication is completed. And **-R** means recursively, we use it for directories as they may also contain many files and folders inside them.

You can check out the list of *dfs* commands using the following command:

```
bin/hdfs dfs
```



```
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
        [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-count [-q] <path> ...]
        [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] <path> ...]
        [-expunge]
        [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-getfacl [-R] <path>]
        [-getfattr [-R] {-n name | -d} [-e en] <path>]
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-d] [-h] [-R] [<path> ...]]
        [-mkdir [-p] <path> ...]
        [-moveFromLocal <localsrc> ... <dst>]
        [-moveToLocal <src> <localdst>]
        [-mv <src> ... <dst>]
        [-put [-f] [-p] <localsrc> ... <dst>]
        [-renameSnapshot <snapshotDir> <oldName> <newName>]
        [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
        [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
        [-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <acl_spec> <pa
        [-setfattr {-n name [-v value] | -x name} <path>]
        [-setrep [-R] [-w] <rep> <path> ...]
        [-stat [format] <path> ...]
        [-tail [-f] <file>]
        [-test -[defsz] <path>]
        [-text [-ignoreCrc] <src> ...]
```