

Università degli Studi di Roma “Tor Vergata”

Dipartimento di Ingegneria Civile e Ingegneria Informatica

Hadoop Distributed File System

A.A. 2017/18

Matteo Nardelli

Laurea Magistrale in
Ingegneria Informatica - II anno

The reference Big Data stack

High-level Interfaces

Data Processing

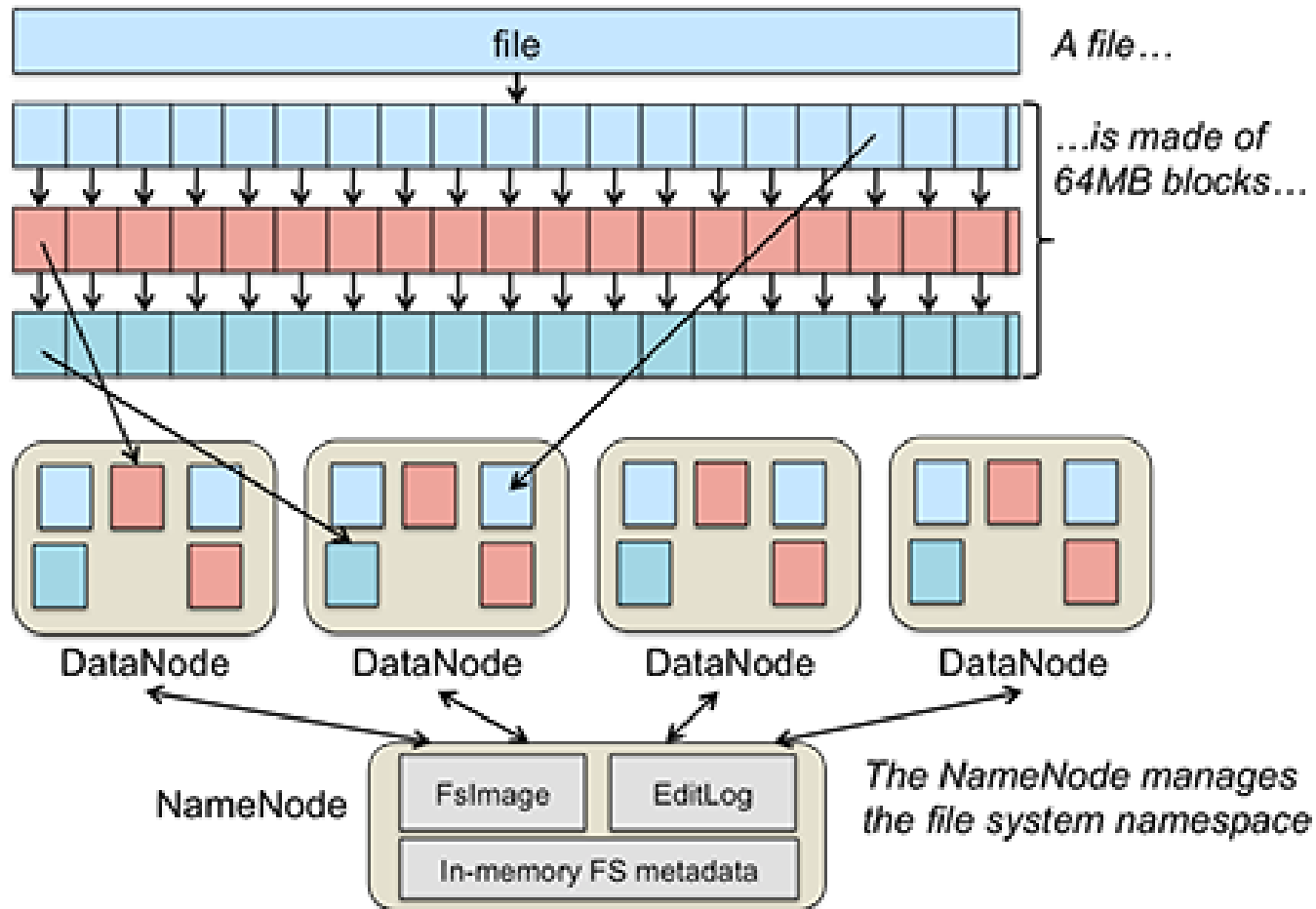
Data Storage

Resource Management

Support / Integration

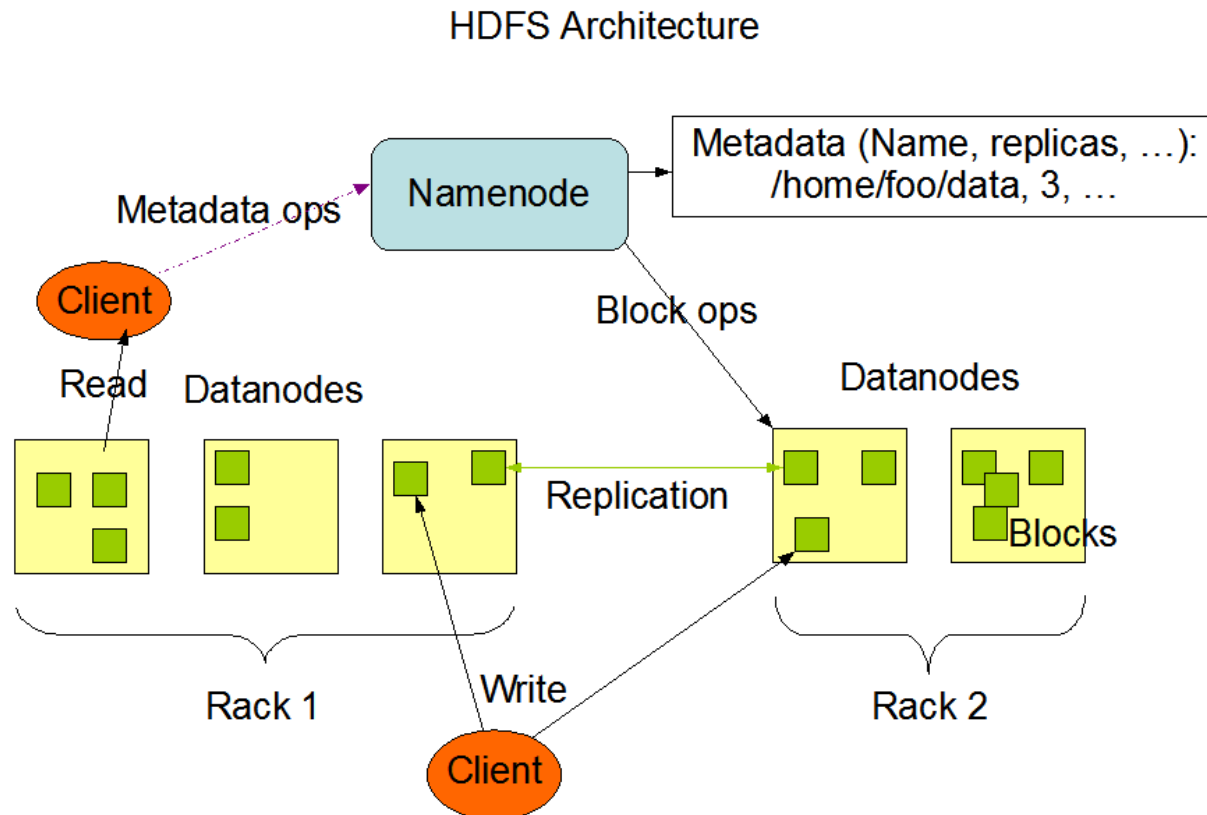
HDFS: a very short summary

A file is split into one or more **blocks** and these blocks are stored in a set of storing nodes (named DataNodes)



HDFS: a very short summary

- An HDFS cluster has two types of nodes:
 - One master, called **NameNode**
 - Multiple workers, called **DataNodes**



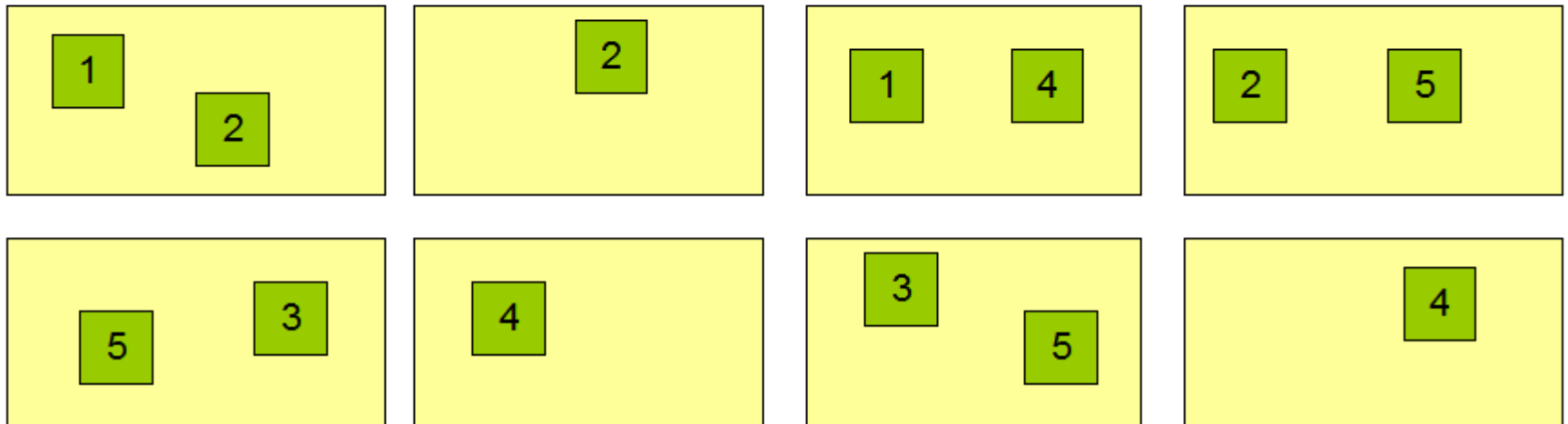
HDFS: a very short summary

Namenode

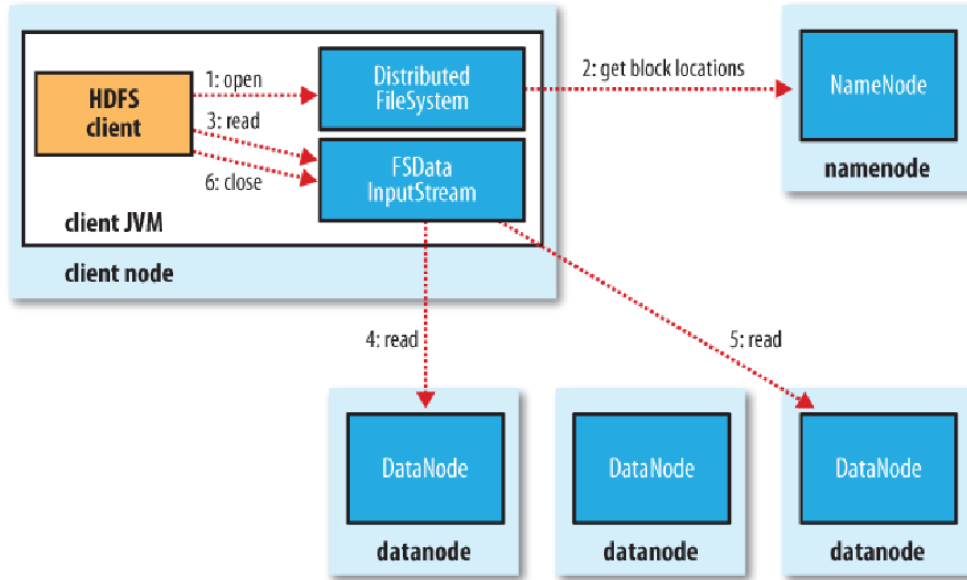
Block Replication

Namenode (Filename, numReplicas, block-ids, ...)
/users/sameerp/data/part-0, r:2, {1,3}, ...
/users/sameerp/data/part-1, r:3, {2,4,5}, ...

Datanodes



HDFS: a very short summary

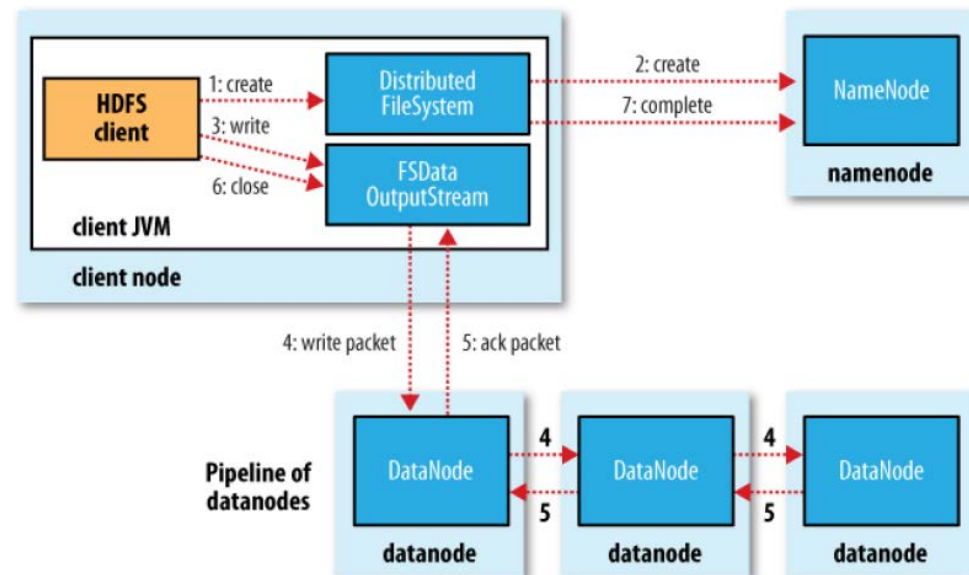


Read

- NameNode used to get block location

Write

- Clients ask NameNode for a list of suitable DataNodes
- This list forms a pipeline: first DataNode stores a copy of a block, then forwards it to the second, and so on



Installation and Configuration of HDFS

(step by step)

Apache Hadoop 2: Configuration

Download

<http://hadoop.apache.org/releases.html>

Configure environment variables

In the **.profile** (or **.bash_profile**) export all needed environment variables

```
$ cd  
$ nano .profile
```



```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle/jre  
export HADOOP_HOME=/usr/local/hadoop-X.Y.Z  
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin
```

(on a Linux/Mac OS system)

Apache Hadoop 2: Configuration

Allow remote login

- Your system should accept connection through SSH (i.e., run a SSH server, set your firewall to allow incoming connections)
- Enable login without password and a RSA key
- Create a new RSA key and add it into the list of authorized keys

```
$ ssh-keygen -t rsa -P ""
```

```
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

(on a Linux/Mac OS system)

Apache Hadoop 2: Configuration

Hadoop Configuration

in \$HADOOP_HOME/etc/hadoop:

- **core-site.xml**: common settings for HDFS, MapReduce, and YARN
- **hdfs-site.xml**: configuration settings for HDFS daemons (i.e., namenode, secondary namenode, and datanodes)
- **mapred-site.xml**: configuration settings for MapReduce (e.g., job history server)
- **yarn-site.xml**: configuration settings for YARN daemons (e.g., resource manager, node managers)

By default, Hadoop runs in a non-distributed mode, as a single Java process. We will configure Hadoop to execute in a pseudo-distributed mode

More on the Hadoop configuration: <https://hadoop.apache.org/docs/current/>

Apache Hadoop 2: Configuration

core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
|
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
|
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

Apache Hadoop 2: Configuration

mapred-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
|
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

yarn-site.xml

```
<?xml version="1.0"?>
|<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

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

<http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/>

Installation and Configuration of HDFS

(our pre-configured Docker image)

HDFS with Dockers

```
$ docker pull matnar/hadoop
```

- create a small network named **hadoop_network** with one namenode (**master**) and 3 datanode (**slave**)

```
$ docker network create --driver bridge hadoop_network
```

```
$ docker run -t -i -p 50075:50075 -d --  
network=hadoop_network --name=slave1 matnar/hadoop
```

```
$ docker run -t -i -p 50076:50075 -d --  
network=hadoop_network --name=slave2 matnar/hadoop
```

```
$ docker run -t -i -p 50077:50075 -d --  
network=hadoop_network --name=slave3 matnar/hadoop
```

```
$ docker run -t -i -p 50070:50070 --  
network=hadoop_network --name=master matnar/hadoop
```

HDFS with Dockers

How to remove the containers

- stop and delete the namenode and datanodes

```
$ docker kill slave1 slave2 slave3
```

```
$ docker rm master slave1 slave2 slave3
```

- remove the network

```
$ docker network rm hadoop_network
```

HDFS: initialization and operations

Apache Hadoop 2: Configuration

At the first execution, the HDFS needs to be initialized

```
$ hdfs namenode -format
```

- this operation **erases the content of the HDFS**
- it should be executed only during the initialization phase

HDFS: Configuration

Start HDFS:

```
$ $HADOOP_HOME/sbin/start-dfs.sh
```

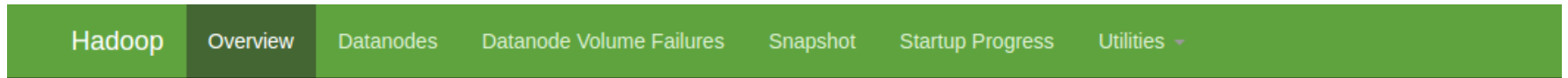
Stop HDFS:

```
$ $HADOOP_HOME/sbin/stop-dfs.sh
```

HDFS: Configuration

When the HDFS is started, you can check its WebUI:

- <http://localhost:50070/>



Overview 'localhost:9000' (active)

Started:	Sat May 21 20:08:25 CEST 2016
Version:	2.7.2, rb165c4fe8a74265c792ce23f546c64604acf0e41
Compiled:	2016-01-26T00:08Z by jenkins from (detached from b165c4f)
Cluster ID:	CID-b9be02fd-bef9-471c-97a5-2fae5eb7f36d
Block Pool ID:	BP-602852293-127.0.1.1-1463854100552

Obtain basic filesystem information and statistics:

```
$ hdfs dfsadmin -report
```

HDFS: Basic operations

ls: for a file ls returns stat on the file; for a directory it returns list of its direct children

```
$ hdfs dfs -ls [-d] [-h] [-R] <args>
```

- d: Directories are listed as plain files
- h: Format file sizes in a human-readable fashion
- R: Recursively list subdirectories encountered

mkdir: takes path uri's as argument and creates directories

```
$ hdfs dfs -mkdir [-p] <paths>
```

- p: creates parent directories along the path.

<http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html>

HDFS: Basic operations

mv: moves files from source to destination. This command allows multiple sources in which case the destination needs to be a directory. Moving files across file systems is not permitted

```
$ hdfs dfs -mv URI [URI ...] <dest>
```

put: copy single src, or multiple srcs from local file system to the destination file system

```
$ hdfs dfs -put <localsrc> ... <dst>
```

Also reads input from stdin and writes to destination file system

```
$ hdfs dfs -put - <dst>
```

HDFS: Basic operations

append: append single or multiple files from local file system to the destination file system

```
$ hdfs dfs -appendToFile <localsrc> ... <dst>
```

get: copy files to the local file system; files that fail the CRC check may be copied with the `-ignorecrc` option

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

cat: copies source paths to stdout

```
$ hdfs dfs -cat URI [URI ...]
```

HDFS: Basic operations

rm: Delete files specified as args

```
$ hdfs dfs -rm [-f] [-r | -R] [-skipTrash] URI [URI ...]
```

- f: does not display a diagnostic message (modify the exit status to reflect an error if the file does not exist)
- R (or -r): deletes the directory and any content under it recursively
- skipTrash: bypasses trash, if enabled

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

```
$ hdfs dfs -cp [-f] [-p | -p[topax]] URI [URI ...] <dest>
```

- f: overwrites the destination if it already exists.
- p: preserves file attributes [topx] (timestamps, ownership, permission, ACL, XAttr). If -p is specified with no arg, then preserves timestamps, ownership, permission.

HDFS: Basic operations

stat: Print statistics about the file/directory at <path> in the specified format

```
$ hadoop fs -stat [format] <path> ...
```

Format accepts

%b Size of file in bytes

%F Will return "file", "directory", or "symlink" depending on the type of inode

%g Group name

%n Filename

%o HDFS Block size in bytes (128MB by default)

%r Replication factor

%u Username of owner

%y Formatted mtime of inode

%Y UNIX Epoch mtime of inode

An example

```
$ echo "File content" >> file
$ hdfs dfs -put file /file
$ hdfs dfs -ls /
$ hdfs dfs -mv /file /democontent
$ hdfs dfs -cat /democontent
$ hdfs dfs -appendToFile file /democontent
$ hdfs dfs -cat /democontent
$ hdfs dfs -mkdir /folder01
$ hdfs dfs -cp /democontent /folder01/text
$ hdfs dfs -ls /folder01
$ hdfs dfs -rm /democontent
$ hdfs dfs -get /folder01/text textfromhdfs
$ cat textfromhdfs
$ hdfs dfs -rm -r /folder01
```

HDFS: Snapshot

Snapshots

- read-only point-in-time copies of the file system
- can be taken on a sub-tree or the entire file system

Common use cases:

data backup, protection against user errors, and disaster recovery.

The implementation is **efficient**:

- the creation is instantaneous
- blocks in datanodes are not copied (it operates on metadata only)
- a snapshot does not adversely affect regular HDFS operations
 - changes are recorded in reverse chronological order so that the current data can be accessed directly
 - the snapshot data is computed by subtracting the modifications from the current data.

<https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-hdfs/HdfsSnapshots.html>

HDFS: Snapshot

Declare a folder where snapshot operations are allowed

```
$ hdfs dfsadmin -allowSnapshot <folder>
```

Create a snapshot

```
$ hdfs dfs -createSnapshot <folder> <snapshot-name>
```

Listing the snapshots

```
$ hdfs dfs -ls <folder>/.snapshot
```

Delete a snapshot

```
$ hdfs dfs -deleteSnapshot <folder> <snapshot-name>
```

Disable snapshot operations within a folder

```
$ hdfs dfsadmin -disallowSnapshot <folder>
```

An example

```
$ hdfs dfs -mkdir /snap
$ hdfs dfs -cp /debs /snap/debs
$ hdfs dfsadmin -allowSnapshot /snap
$ hdfs dfs -createSnapshot /snap snap001
$ hdfs dfs -ls /snap/.snapshot
$ hdfs dfs -ls /snap/.snapshot/snap001
$ hdfs dfs -cp -ptopax /snap/.snapshot/snap001/debs /debs
$ hdfs dfs -deleteSnapshot /snap snap001
$ hdfs dfsadmin -disallowSnapshot /snap
$ hdfs dfs -rm -r /snap
```

HDFS: Replication

setrep: change the replication factor of a file. If path is a directory then the command recursively changes the replication factor of all files under the directory tree rooted at path.

```
$ hdfs dfs -setrep [-w] <numReplicas> <path>
```

-w: requests that the command wait for the replication to complete; this can potentially take a very long time

An example

```
$ hdfs dfs -put file /norepl/file
$ hdfs dfs -ls /norepl
$ hdfs dfs -setrep 1 /norepl
$ hdfs dfs -ls /norepl
$ hdfs dfs -put file /norepl/file2
$ hdfs dfs -ls /norepl
$ hdfs dfs -setrep 1 /norepl/file2
# also check block availability from webUI
```