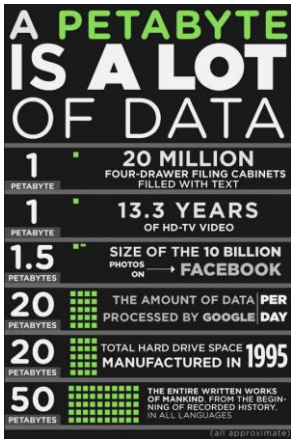
Introduction – Unit 1

How much data do the following use?

	Facebook	NSA	Google
Current Storage	300 petabytes	~5 exabytes	15 exabytes
Processed per day	600 terabytes	30 petabytes	100 petabytes
Users per month	1 billion		
Likes per day	2.7 billion	NSA touches 1.6% of internet traffic per day Web searches, websites visited, phone calls, credit/debit card transactions, financial and health information	
Photos uploaded per day	300 million		
Unique search users per month			>1 billion
Searched per second			2.3 million
Number of pages indexed			60 trillion

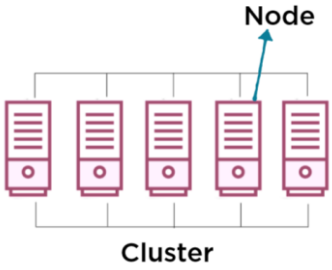


Big Data System requirements

Raw Data	Extract useful information	Accommodate changing needs
Store massive amounts of data	Process it in a timely manner	Scale easily as data grows
Store	Process	

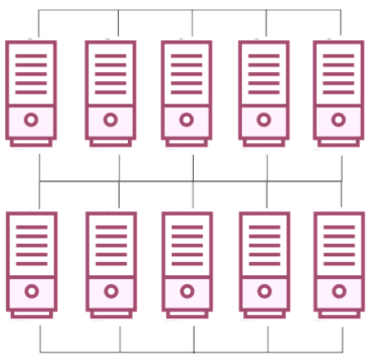
- The infrastructure needs to keep up with the growing size of data
- Distributed Computing Frameworks like Hadoop were developed for exactly this

Two Ways to Build a System

Monolithic	Distributed
	
A single powerful server	Many small and cheap computers come together to act as a single entity
2x Expense	Such a system can scale linearly
< 2x Performance	

The way to build a system

- Distributed Systems

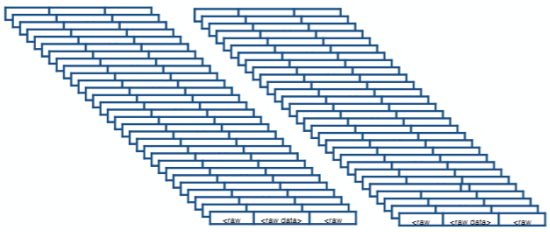
	2x Nodes 2x Storage ~ 2x Speed

Server farms

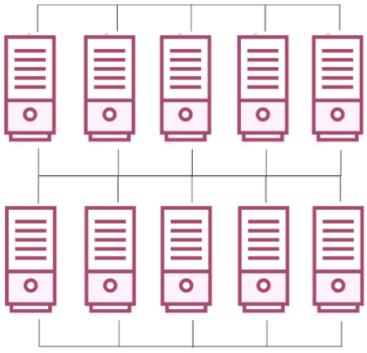
- Companies like Facebook, Google, Amazon are building vast server farms
- These farms have 100s of 1000s of servers working in tandem to process complex data
- All of these servers need to be coordinated by a single piece of software

Single Co-ordinating Software

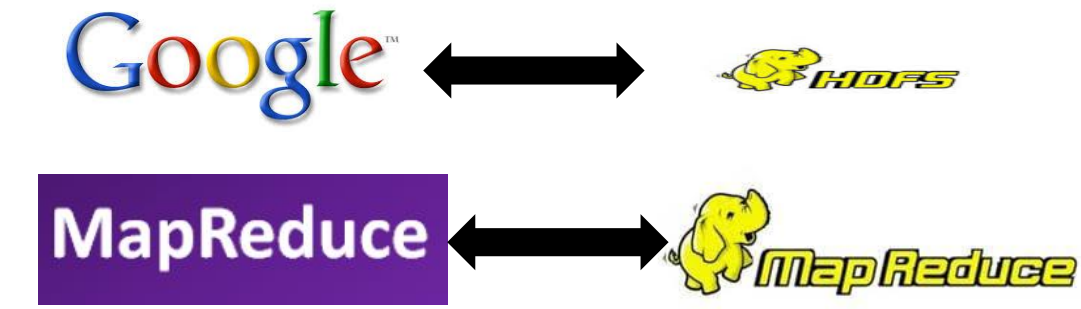
- Should
  - Partition data
  - Co-ordinate computing tasks
  - Handle fault tolerance and recovery
  - Allocate capacity to processes
- Back in the early 2000s Google realized that web search requires something completely new
- Google developed proprietary software to run on these distributed systems



- STEP 1
  - Store millions of records on multiple machines



- STEP 2
  - Run processes on all these machines to crunch data

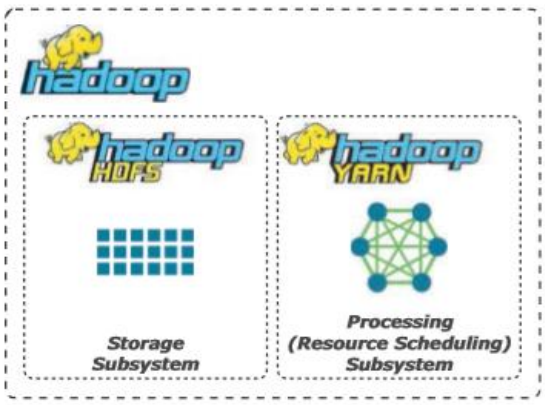


- A file system to manage the storage of data
- A framework to process data across multiple servers

Hadoop

<table>
<tr><td rowspan="2">  </td><td colspan="2">- MapReduce was broken into two separate parts
 
  </td></tr>
<tr><td>A framework to define a data processing task</td><td>A framework to run the data processing task</td></tr>
</table>

Hadoop core system



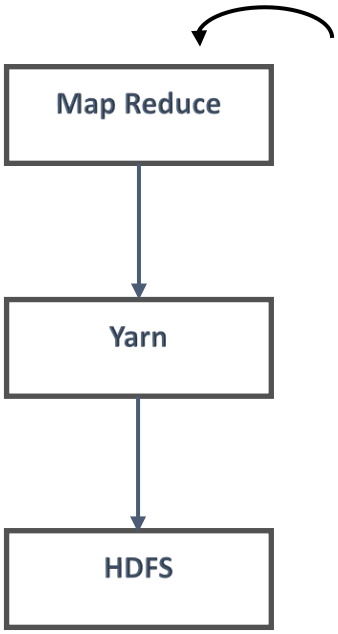
Hadoop - history

- Hadoop was created by Doug Cutting, the creator of Apache Lucene, the widely used text search library.
- Hadoop has its origins in Apache Nutch, an open source web-search engine, itself a part of the Lucene project.

Hadoop – In a nutshell

- Hadoop is a data storage and processing platform, based upon a central concept: data locality.
- Data locality refers to the processing of data where it resides by bringing the computation to the data, rather than the typical pattern of requesting data from its location (for example, a database management system) and sending this data to a remote processing system or host.
- With Internet-scale data - "Big Data" - it is no longer efficient, or even possible in some cases, to move the large volumes of data required for processing across the network at compute time.
- Hadoop enables large datasets to be processed locally on the nodes of a cluster using a shared nothing approach, where each node can independently process a much smaller subset of the entire dataset without needing to communicate with one another.
- Hadoop is schema-less with respect to its write operations (it is what's known as a schema-on-read system).
  - This means that it can store and process a wide range of data, from unstructured text documents, to semi-structured JSON (JavaScript Object Notation) or XML documents, to well-structured extracts from relational database systems.
- Schema-on-read systems are a fundamental departure from the relational databases that is used, which are, in contrast, broadly categorized as schema-on-write systems, where data is typically strongly typed and a schema is predefined and enforced upon INSERT, UPDATE, or UPSERT operations.
- NoSQL platforms (such as Apache HBase or Cassandra) are typically classified as schema-on-read systems as well.
- Because the schema is not interpreted during write operations to Hadoop, there are no indexes, statistics, or other constructs typically employed by database systems to optimize query operations and filter or reduce the amount of data returned to a client.
- Hadoop is designed to find needles in haystacks.
  - It does so by dividing and conquering a large problem into a set of smaller problems applying the concepts of data locality and shared nothing.

Co-ordination Between Hadoop Blocks



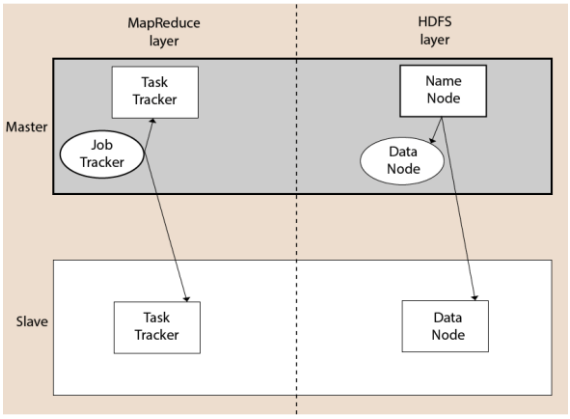
User defines map and reduce tasks using the MapReduce API

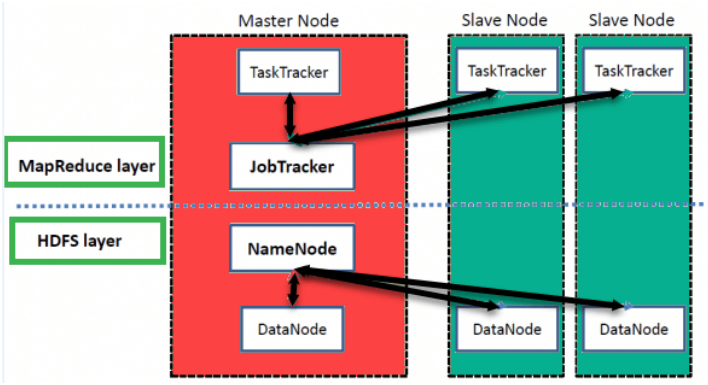
A job is triggered on the cluster

YARN figures out where and how to run the job, and stores the result in HDFS

Hadoop Architecture

- The Hadoop architecture is a package of the file system, MapReduce engine and the HDFS (Hadoop Distributed File System).
  - The MapReduce engine can be MapReduce/MR1 or YARN/MR2.
- A Hadoop cluster consists of a single master and multiple slave nodes.
- The master node includes Job Tracker, Task Tracker, NameNode, and DataNode.
- The slave node includes DataNode and TaskTracker.





- NameNode
  - It is a single master server exist in the HDFS cluster.
  - As it is a single node, it may become the reason of single point failure.
  - It manages the file system namespace by executing an operation like the opening, renaming and closing the files.
  - It simplifies the architecture of the system.
- DataNode
  - The HDFS cluster contains multiple DataNodes.
  - Each DataNode contains multiple data blocks.
  - These data blocks are used to store data.
  - It is the responsibility of DataNode to read and write requests from the file system's clients.
  - It performs block creation, deletion, and replication upon instruction from the NameNode.
- Job Tracker
  - The role of Job Tracker is to accept the MapReduce jobs from client and process the data by using NameNode.
  - In response, NameNode provides metadata to Job Tracker.
- Task Tracker
  - It works as a slave node for Job Tracker.
  - It receives task and code from Job Tracker and applies that code on the file.
  - This process can also be called as a Mapper.
- **Master node**
  - The master node allows you to conduct parallel processing of data using Hadoop MapReduce.
- **Slave node**
  - The slave nodes are the additional machines in the Hadoop cluster which allows you to store data to conduct complex calculations.
  - Moreover, all the slave node comes with Task Tracker and a DataNode.

- ∞ This allows you to synchronize the processes with the NameNode and Job Tracker respectively.

ϖ **MapReduce Layer**

- ∞ The MapReduce comes into existence when the client application submits the MapReduce job to Job Tracker.
- ∞ In response, the Job Tracker sends the request to the appropriate Task Trackers.
- ∞ Sometimes, the TaskTracker fails or time out.
  ⇒ In such a case, that part of the job is rescheduled.

Advantages of Hadoop

ϖ **Fast**

- ∞ In HDFS the data distributed over the cluster and are mapped which helps in faster retrieval.
- ∞ Even the tools to process the data are often on the same servers, thus reducing the processing time.
- ∞ It is able to process terabytes of data in minutes and Peta bytes in hours.

ϖ **Scalable**

- ∞ Hadoop cluster can be extended by just adding nodes in the cluster.

ϖ **Cost Effective**

- ∞ Hadoop is open source and uses commodity hardware to store data so it really cost effective as compared to traditional relational database management system.

ϖ **Resilient to failure**

- ∞ HDFS has the property with which it can replicate data over the network, so if one node is down or some other network failure happens, then Hadoop takes the other copy of data and use it.
- ∞ Normally, data are replicated thrice but the replication factor is configurable.

How Does Hadoop Work?

ϖ It is quite expensive to build bigger servers with heavy configurations that handle large scale processing.

ϖ As an alternative, tie together many commodity computers with single-CPU, as a single functional distributed system.

ϖ The clustered machines can read the dataset in parallel and provide a much higher throughput.

ϖ Moreover, it is cheaper than one high-end server.

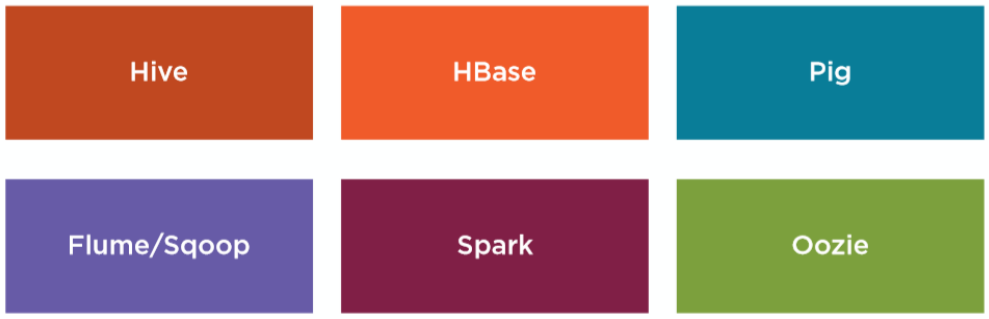
ϖ Hadoop runs code across a cluster of computers.

ϖ This process includes the following core tasks that Hadoop performs –

- ∞ Data is initially divided into directories and files.
  ⇒ Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).
- ∞ These files are then distributed across various cluster nodes for further processing.
- ∞ HDFS, being on top of the local file system, supervises the processing.
- ∞ Blocks are replicated for handling hardware failure.
- ∞ Checking that the code was executed successfully.
- ∞ Performing the sort that takes place between the map and reduce stages.
- ∞ Sending the sorted data to a certain computer.
- ∞ Writing the debugging logs for each job.

HADOOP ECO System

ϖ An ecosystem of tools has sprung up around this core piece of software.



	∞ Provides an SQL interface to Hadoop ∞ The bridge to Hadoop for folks who don't have exposure to OOP in Java
	∞ A database management system on top of Hadoop Integrates with your application just like a traditional database
	∞ A data manipulation language. ∞ Transforms unstructured data into a structured format ∞ Query this structured data using interfaces like Hive
	∞ A distributed computing engine used along with Hadoop ∞ Interactive shell to quickly process datasets ∞ Has a bunch of built in libraries for machine learning, stream processing, graph processing etc.

	∞ A tool to schedule workflows on all the Hadoop ecosystem technologies
 	∞ Tools to transfer data between other systems and Hadoop

Hadoop Installation - Modes and Methods

Installation Modes

- Hadoop has three installation modes
  - Standalone
  - Pseudo-distributed
  - Fully Distributed
- Standalone
  - The default mode in which Hadoop runs.
  - Runs on a single node
  - A single JVM process
  - Local File System for Storage
  - HDFS and YARN do not run
  - Used to test MapReduce programs before running them on a cluster
- Pseudo-Distributed
  - Runs on a single node
  - 2 JVM processes to simulate 2 nodes
  - HDFS for storage
  - YARN for managing tasks
  - Used as a fully-fledged test environment
- Fully Distributed
  - Runs on a cluster of machines
    - ⇒ Linux servers in a data center
    - ⇒ VMs requisitioned on a cloud service
  - Manual configuration of a cluster is complicated
  - Usually use enterprise editions
    - ⇒ Cloudera, MapR, Hortonworks

HDFS

- HDFS is a file system designed for storing very large files with streaming data access patterns, running on clusters of commodity hardware.

- Hadoop Distributed File System
- It is a file system that is spread across multiple machines.
- 
- Built on commodity hardware
- Highly fault tolerant, hardware failure is the norm
- Suited to batch processing - data access has high throughput rather than low latency
- Supports very large data sets

HDFS – Data Storage

<table>
<tr>
<td>

Any data that is stored in HDFS is split across multiple storage disks.
Each disk is present on a different machine in a cluster.
The file system's responsibility is to manage all the machines and all the storage space.
This is done by master-slave architecture.

</td>
<td>  </td>
</tr>
<tr>
<td>

It randomly sets up one machine as master node.
The master node is responsible for coordinating storage across all other nodes on the machine which are slave nodes.

</td>
<td>  </td>
</tr>
<tr>
<td colspan="2">

On the master node HDFS runs a Java Process that receives all requests that are made to the cluster and then forwards these requests to the slave nodes in the cluster.
This process is called the name node.

The node itself is designated as name node.

</td>
</tr>
</table>

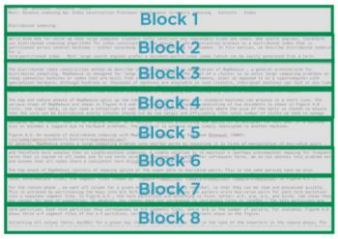
- On all the other nodes HDFS runs the data node processes.
  - All other machines are designated as data node.
- One name node per cluster and many data nodes depending on the number of machines in the cluster.

HDFS – Analogy

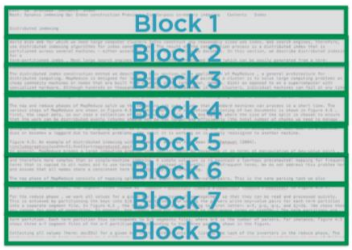
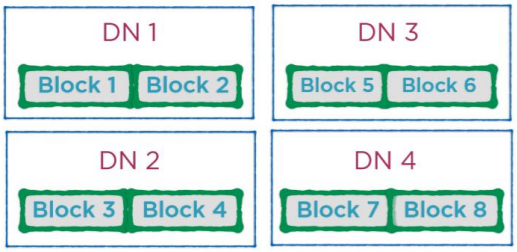
	If the data in the distributed file system is a book
Name node 	The name node is the table of contents. Manages the overall file system No data is stored in the name node Stores ⇒ The directory structures. ⇒ Metadata of the files
Data nodes 	The data nodes hold the actual text in each page Physically stores the data in the files. This is usually unformatted and unstructured data

Storing File in HDFS

	Data stored in HDFS is normally a large text file of data that is in peta bytes.

	ϖ These files are normally broken in to smaller chunks of information called blocks. ϖ Each of these blocks are stored on different nodes in a cluster. ϖ The entire file is not stored in one node.

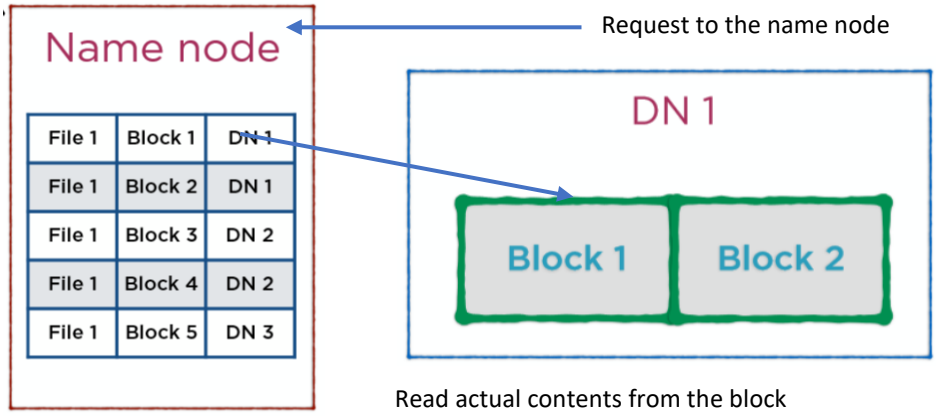
ϖ Each of these blocks are of equal size.
  ∞ Different length files are treated in the same way.
  ∞ Equal block size ensures equal amount of processing time for all queries.
ϖ Storage is simplified
  ∞ A block is the unit for replication and fault tolerance
    ⇒ Multiple copies of blocks of data are kept and not multiple copies of the entire data.
ϖ The blocks are of an optimal size of 128 MB
ϖ Block size is a tradeoff.
  ∞ If block size is increased, parallelism is reduced.
    ⇒ Fewer chunks of data.
    ⇒ Fewer processes.
  ∞ If block size is too small, increases overheads.
    ⇒ One file will have many hundreds of splits
    ⇒ Requires hundreds of splits
ϖ Time taken to read a block of data from disk is divided into
  ∞ Seek time
    ⇒ Time taken to seek that position
    ⇒ Is roughly 1% of transfer time
  ∞ Read time / Transfer time
    ⇒ Time taken to read the block

	ϖ Store the blocks across the data nodes
	ϖ Each node contains a partition or a split of data

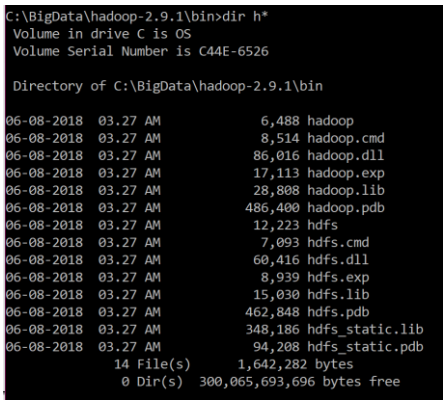
Reading a file in HDFS

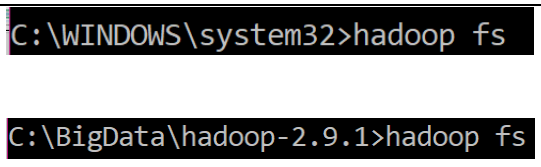
ϖ Use metadata in the name node to look up block locations

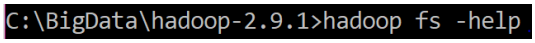
ϖ Read the blocks from respective locations

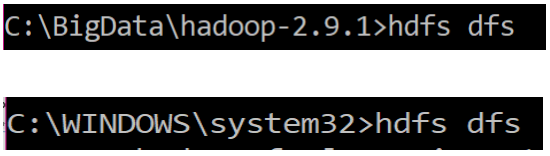


Interacting with the HDFS command line interface

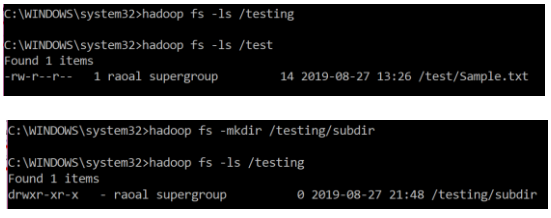
<table>
<tr>
<td>
ϖ Go to the bin directory of Hadoop.
ϖ In ubuntu type ls –l
ϖ In windows type dir
ϖ hadoop.cmd and hdfs.cmd are the commands for interaction with file system.
</td>
<td>


</td>
</tr>
<tr>
<td>
ϖ The following command will start the namenode as well as the data nodes as cluster.
ϖ start-dfs.sh or start-all.cmd
</td>
<td></td>
</tr>
<tr>
<td colspan="2">ϖ In the terminal or command prompt type Hadoop fs.</td>
</tr>
<tr>
<td>ϖ Allows to manipulate file system.</td>
<td>


</td>
</tr>
<tr>
<td>ϖ Allows to open the man page for the commands</td>
<td>


</td>
</tr>
<tr>
<td>ϖ Allows to manipulate the Hadoop Distributed file system</td>
<td>


</td>
</tr>
<tr>
<td>ϖ Create a directory at the root level.</td>
<td>


</td>
</tr>
<tr>
<td>ϖ To check the contents of the directory</td>
<td>


</td>
</tr>
</table>

```
C:\WINDOWS\system32>hadoop fs -ls /testing
Found 2 items
-rw-r--r--   1 raoal supergroup       4969 2019-08-27 22:10 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir
```

File Permissions | Number of file replicas. Set to 1 since that is the replication factor | Owner and group of files | File Size | Last Modified Time | File name and path

- To copy from local machine to Hadoop, use hadoop fs –copyFromLocal
- The first argument is source files.
- The second argument is the destination folder.
- The destination folder should exist.

```
C:\WINDOWS\system32>hadoop fs -copyFromLocal C:\BigData\hadoop-2.9.1\
etc\hadoop\hadoop-env.sh /testing
```

- To copy from local machine to Hadoop use –put
- The first argument is source files.
- The second argument is the destination folder.
- The destination folder should exist.

```
C:\WINDOWS\system32>hadoop fs -put C:\BigData\hadoop-2.9.1\etc\hadoop\core-site.xml
/testing
```

```
C:\WINDOWS\system32>hadoop fs -ls /testing
Found 3 items
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup       4969 2019-08-27 22:10 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir
```

```
C:\WINDOWS\system32>hadoop fs -ls /
Found 2 items
drwxr-xr-x   - raoal supergroup          0 2019-08-27 13:26 /test
drwxr-xr-x   - raoal supergroup          0 2019-08-28 14:00 /testing
```

- Use cp command to copy between directories on HDFS.

```
C:\WINDOWS\system32>hadoop fs -cp /testing/* /test/
```

```
C:\WINDOWS\system32>hadoop fs -ls /testing
Found 3 items
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup       4969 2019-08-27 22:10 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir
```

```
C:\WINDOWS\system32>hadoop fs -ls /test
Found 4 items
-rw-r--r--   1 raoal supergroup         14 2019-08-27 13:26 /test/Sample.txt
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:11 /test/core-site.xml
-rw-r--r--   1 raoal supergroup       4969 2019-08-28 14:11 /test/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-28 14:11 /test/subdir
```

- Create a directory on your local machine.

```
C:\WINDOWS\system32>mkdir fromhdfs
```

- To copy from HDFS to local directory use copyToLocal.
- The first argument is source files on HDFS.
- The second argument is the destination folder on local machine.
- The destination folder should exist.

```
C:\WINDOWS\system32>hadoop fs -copyToLocal /test/* fromhdfs
```

- Check whether it is copied or not

```
C:\WINDOWS\system32>dir fromhdfs
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\WINDOWS\system32\fromhdfs

28-08-2019  02.15 PM    <DIR>          .
28-08-2019  02.15 PM    <DIR>          ..
28-08-2019  02.15 PM               901 core-site.xml
28-08-2019  02.15 PM             4,969 hadoop-env.sh
28-08-2019  02.15 PM                14 Sample.txt
28-08-2019  02.15 PM    <DIR>          subdir
               3 File(s)          5,884 bytes
               3 Dir(s)  299,288,395,776 bytes free
```

- Use the hadoop get command to copy from HDFS to local directory

```
C:\Windows\System32>hadoop fs -get /test/* fromhdfs
```

```
C:\Windows\System32>dir fromhdfs
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\Windows\System32\fromhdfs

28-08-2019  02.59 PM    <DIR>          .
28-08-2019  02.59 PM    <DIR>          ..
28-08-2019  02.59 PM               901 core-site.xml
28-08-2019  02.59 PM             4,969 hadoop-env.sh
28-08-2019  02.59 PM                14 Sample.txt
28-08-2019  02.59 PM    <DIR>          subdir
               3 File(s)          5,884 bytes
               3 Dir(s)  299,298,496,512 bytes free
```

- To see the contents of a file in HDFS use -cat.

```
C:\Windows\System32>hadoop fs -cat /test/Sample.txt
Hello World!!!
```

- To delete a file on HDFS use -rm.

```
C:\Windows\System32>hadoop fs -rm -r /test/core-site.xml
Deleted /test/core-site.xml
```

- -f If the file does not exist, do not display a diagnostic message or modify the exit status to reflect an error.
- -[rR] Recursively deletes directories.
- -skipTrash option bypasses trash, if enabled and immediately deletes <src>
- -safely option requires safety confirmation, if enabled, requires confirmation before deleting large directory with more than <hadoop.shell.delete.limit.num.files> files.
  - Delay is expected when walking over large directory recursively to count the number of files to be deleted before the confirmation.

```
C:\WINDOWS\system32>hadoop fs -rm -f /test/core-site.xml

C:\WINDOWS\system32>hadoop fs -rm /test/core-site.xml
rm: `/test/core-site.xml': No such file or directory
```

```
C:\WINDOWS\system32>hadoop fs -rm -safely /test/Samp.txt
Deleted /test/Samp.txt
```

- To know the disk usage of all files in a particular folder use –du.
- It gives the result in bytes.
- To give the same as a summary use –dus
- To get the usage in human readable format use –h option in –du.

```
C:\Windows\System32>hadoop fs -du /testing
901   /testing/core-site.xml
4969  /testing/hadoop-env.sh
0     /testing/subdir
```

```
C:\Windows\System32>hadoop fs -dus /testing
dus: DEPRECATED: Please use 'du -s' instead.
5870  /testing
```

```
C:\Windows\System32>hadoop fs -du -s /testing
10853  /testing
```

```
C:\Windows\System32>hadoop fs -du -h /testing
14     /testing/Sample.txt
901    /testing/core-site.xml
4.9 K  /testing/hadoop-e.sh
4.9 K  /testing/hadoop-env.sh
0      /testing/subdir

C:\Windows\System32>hadoop fs -du -s -h /testing
10.6 K  /testing
```

- To move a file from local directory to HDFS and delete the same from local directory use the command –moveFromLocal.
- It requires two parameters.
- The first argument is source files.
- The second argument is the destination folder.

```
C:\Windows\System32>hadoop fs -moveFromLocal fromhdfs/Sample.txt /testing/
```

```
C:\Windows\System32>dir fromhdfs
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\Windows\System32\fromhdfs

28-08-2019  09.40 PM    <DIR>          .
28-08-2019  09.40 PM    <DIR>          ..
28-08-2019  02.59 PM               901 core-site.xml
28-08-2019  02.59 PM             4,969 hadoop-env.sh
28-08-2019  02.59 PM    <DIR>          subdir
               2 File(s)          5,870 bytes
               3 Dir(s)  298,893,316,096 bytes free
```

- To move a file or directory within different HDFS directories

```
C:\Windows\System32>hadoop fs -mv /test/hadoop-env.sh /testing/
mv: `/testing/hadoop-env.sh': File exists

C:\Windows\System32>hadoop fs -mv /test/hadoop-env.sh /testing/hadoop-e.sh
```

```
C:\Windows\System32>hadoop fs -ls /testing
Found 5 items
-rw-r--r--   1 raoal supergroup         14 2019-08-28 21:40 /testing/Sample.txt
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup       4969 2019-08-28 14:11 /testing/hadoop-e.sh
-rw-r--r--   1 raoal supergroup       4969 2019-08-27 22:10 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir
```

- To merge n number of files in the HDFS distributed file system and put it into a single file in local file system use the command getmerge.
- The first set of parameters are the source files.
- The last parameter is the destination folder with file name.

```
C:\Windows\System32>hadoop fs -getmerge /testing/hadoop-e.sh /testing/hadoop-env.sh fromhdfs/hadoop.txt
```

```
C:\Windows\System32>hadoop fs -du  /testing
14    /testing/Sample.txt
901   /testing/core-site.xml
4969  /testing/hadoop-e.sh
4969  /testing/hadoop-env.sh
0     /testing/subdir
```

```
C:\Windows\System32>dir fromhdfs
 Volume in drive C is OS
 Volume Serial Number is C44E-6526

 Directory of C:\Windows\System32\fromhdfs

29-08-2019  09.52 AM    <DIR>          .
29-08-2019  09.52 AM    <DIR>          ..
29-08-2019  09.52 AM                88 .hadoop.txt.crc
28-08-2019  09.52 PM                12 .sampleq.txt.crc
28-08-2019  02.59 PM               901 core-site.xml
28-08-2019  02.59 PM             4,969 hadoop-env.sh
29-08-2019  09.52 AM             9,938 hadoop.txt
28-08-2019  09.52 PM                28 sampleq.txt
28-08-2019  02.59 PM    <DIR>          subdir
               6 File(s)         15,936 bytes
               3 Dir(s)  299,261,620,224 bytes free
```

- To print information about path use stat command.
- Format is a string which accepts file size in blocks (%b), filename (%n), block size (%o), replication (%r), and modification date (%y, %Y).

```
C:\Windows\System32>hadoop fs -stat /testing
2019-08-29 04:19:21
```

- To show the last 1KB of a file in HDFS on stdout use -tail command.
- -f Shows appended data as the file grows.

```
C:\WINDOWS\system32>hadoop fs -tail -f /testing/hadoop-e.sh
ters
###
# Specify the JVM options to be used when starting the HDFS Mover.
# These options will be appended to the options specified as HADOOP_OPTS
# and therefore may override any similar flags set in HADOOP_OPTS
#
# export HADOOP_MOVER_OPTS=""
```

- Appends the contents of all the given local files to the given dst file.
- The dst file will be created if it does not exist.
- If <localSrc> is -, then the input is read from stdin.

```
C:\Windows\System32>hadoop fs -appendToFile - /testing/Sample.txt
Hello, How are you?

^X
^Z

C:\Windows\System32>hadoop fs -cat  /testing/Sample.txt
Hello World!!!Hello, How are you?
```

- To Count the number of directories, files and bytes under the paths that match the specified file pattern use –count.
- The output columns are:
- DIR_COUNT FILE_COUNT CONTENT_SIZE PATHNAME
- With –q option, the output columns are
- QUOTA REM_QUOTA SPACE_QUOTA REM_SPACE_QUOTA DIR_COUNT FILE_COUNT CONTENT_SIZE PATHNAME

- The -h option shows file sizes in human readable format.
- The -v option displays a header line.
- The -x option excludes snapshots from being calculated.
- The -t option displays quota by storage types.
- It should be used with -q or -u option, otherwise it will be ignored.
- If a comma-separated list of storage types is given after the -t option, it displays the quota and usage for the specified types.
- Otherwise, it displays the quota and usage for all the storage types that support quota.
- The list of possible storage types(case insensitive):
  - ram_disk, ssd, disk and archive.
- It can also pass the value '', 'all' or 'ALL' to specify all the storage types.
- The -u option shows the quota and the usage against the quota without the detailed content summary.

```
C:\WINDOWS\system32>hadoop fs -count /test
           2            1                 14 /test

C:\WINDOWS\system32>hadoop fs -ls /test
Found 2 items
-rw-r--r--   1 raoal supergroup         14 2019-08-27 13:26 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2019-08-28 14:11 /test/subdir
```

```
C:\WINDOWS\system32>hadoop fs -count /testing
           2            4              10879 /testing

C:\WINDOWS\system32>hadoop fs -ls /testing
Found 5 items
-rw-r--r--   1 raoal supergroup         40 2019-08-29 10:12 /testing/Sample.txt
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup       4969 2019-08-28 14:11 /testing/hadoop-e.sh
-rw-r--r--   1 raoal supergroup       4969 2019-08-27 22:10 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir
```

```
C:\WINDOWS\system32>hadoop fs -count -q /test
       none           inf           none           inf            2            1               14 /test
```

```
C:\WINDOWS\system32>hadoop fs -count -v /test
   DIR_COUNT   FILE_COUNT       CONTENT_SIZE PATHNAME
           2            1                 14 /test
```

```
C:\WINDOWS\system32>hadoop fs -count -h /test
           2            1                 14 /test
```

```
C:\WINDOWS\system32>hadoop fs -count -u /test
        none             inf            none             inf /test
```

```
C:\WINDOWS\system32>hadoop fs -count -u -v /test
       QUOTA       REM_QUOTA     SPACE_QUOTA REM_SPACE_QUOTA PATHNAME
        none             inf            none             inf /test
```

```
C:\WINDOWS\system32>hadoop fs -count -q -v /test
       QUOTA       REM_QUOTA     SPACE_QUOTA REM_SPACE_QUOTA    DIR_COUNT   FILE_COUNT       CONTENT_SIZE PATHNAME
        none             inf            none             inf           2            1                 14 /test
```

- To dump checksum information for files that match the file pattern <src> to stdout use –checksum command.
- This requires a round-trip to a datanode storing each block of the file, and thus is not efficient to run on a large number of files.
- The checksum of a file depends on its content, block size and the checksum algorithm and parameters used for creating the file.

```
C:\WINDOWS\system32>hadoop fs -checksum /test/Sample.txt
/test/Sample.txt        MD5-of-0MD5-of-512CRC32C        000002000000000000000000b1519b12e40c2c708d46777956749a00
```

- To list the contents that match the specified file pattern use –ls command.
- If path is not specified, the contents of /user/<currentUser> will be listed.
- For a directory a list of its direct children is returned (unless -d option is specified).
- Directory entries are of the form
  - permissions - userId groupId sizeOfDirectory(in bytes) modificationDate(yyyy-MM-dd HH:mm) directoryName
- File entries are of the form
  - permissions numberOfReplicas userId groupId sizeOfFile(in bytes) modificationDate(yyyy-MM-dd HH:mm) fileName
- -C --- Display the paths of files and directories only.
- -d --- Directories are listed as plain files.
- -h --- Formats the sizes of files in a human-readable fashion rather than a number of bytes.
- -q --- Print ? instead of non-printable characters.
- -R --- Recursively list the contents of directories.
- -t --- Sort files by modification time (most recent first).
- -S --- Sort files by size.
- -r --- Reverse the order of the sort.
- -u --- Use time of last access instead of modification for display and sorting.

```
C:\WINDOWS\system32>hadoop fs -ls -C /test
/test/Sample.txt
/test/subdir
```

```
C:\WINDOWS\system32>hadoop fs -ls -d /test
drwxr-xr-x   - raoal supergroup          0 2019-08-29 09:49 /test

C:\WINDOWS\system32>hadoop fs -ls -d /testing
drwxr-xr-x   - raoal supergroup          0 2019-08-29 09:49 /testing
```

```
C:\WINDOWS\system32>hadoop fs -ls -h /testing
Found 5 items
-rw-r--r--   1 raoal supergroup         40 2019-08-29 10:12 /testing/Sample.txt
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup      4.9 K 2019-08-28 14:11 /testing/hadoop-e.sh
-rw-r--r--   1 raoal supergroup      4.9 K 2019-08-27 22:10 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir

C:\WINDOWS\system32>hadoop fs -ls -h -d /testing
drwxr-xr-x   - raoal supergroup          0 2019-08-29 09:49 /testing
```

```
C:\WINDOWS\system32>hadoop fs -ls -h -d /testing
drwxr-xr-x   - raoal supergroup          0 2019-08-29 09:49 /testing

C:\WINDOWS\system32>hadoop fs -ls -h -d /test
drwxr-xr-x   - raoal supergroup          0 2019-08-29 09:49 /test
```

```
C:\WINDOWS\system32>hadoop fs -ls -S /testing
Found 5 items
-rw-r--r--   1 raoal supergroup       4969 2019-08-28 14:11 /testing/hadoop-e.sh
-rw-r--r--   1 raoal supergroup       4969 2019-08-27 22:10 /testing/hadoop-env.sh
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup         40 2019-08-29 10:12 /testing/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir
```

```
C:\WINDOWS\system32>hadoop fs -ls -q /test
Found 2 items
-rw-r--r--   1 raoal supergroup         14 2019-08-27 13:26 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2019-08-28 14:11 /test/subdir

C:\WINDOWS\system32>hadoop fs -ls -q /testing
Found 5 items
-rw-r--r--   1 raoal supergroup         40 2019-08-29 10:12 /testing/Sample.txt
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup       4969 2019-08-28 14:11 /testing/hadoop-e.sh
-rw-r--r--   1 raoal supergroup       4969 2019-08-27 22:10 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir
```

```
C:\WINDOWS\system32>hadoop fs -ls -t /testing
Found 5 items
-rw-r--r--   1 raoal supergroup         40 2019-08-29 10:12 /testing/Sample.txt
-rw-r--r--   1 raoal supergroup       4969 2019-08-28 14:11 /testing/hadoop-e.sh
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup       4969 2019-08-27 22:10 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir
```

```
C:\WINDOWS\system32>hadoop fs -ls -u /testing
Found 5 items
-rw-r--r--   1 raoal supergroup         40 2019-08-29 10:12 /testing/Sample.txt
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup       4969 2019-08-31 09:35 /testing/hadoop-e.sh
-rw-r--r--   1 raoal supergroup       4969 2019-08-29 09:52 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 1970-01-01 05:30 /testing/subdir
```

- To find all files that match the specified expression and apply selected actions to them use –find command.
- If no <path> is specified then defaults to the current working directory.
- If no expression is specified then defaults to -print.
- The following primary expressions are recognized
  - -name pattern
  - -iname pattern
- Evaluates as true if the basename of the file matches the pattern using standard file system globbing.
- If -iname is used then the match is case insensitive.
  - -print
  - -print0
- Always evaluates to true.
- Causes the current pathname to be written to standard output followed by a newline.
- If the -print0 expression is used then an ASCII NULL character is appended rather than a newline.
- The following operators are recognized:
  - expression -a expression
  - expression -and expression
  - expression expression
    - ⇒ Logical AND operator for joining two expressions.
    - ⇒ Returns true if both child expressions return true.
    - ⇒ Implied by the juxtaposition of two expressions and so does not need to be explicitly specified.
    - ⇒ The second expression will not be applied if the first fails.

```
C:\WINDOWS\system32>hadoop fs -find /test/S*
/test/Sample.txt

C:\WINDOWS\system32>hadoop fs -find /test/*.sh
find: `/test/*.sh': No such file or directory

C:\WINDOWS\system32>hadoop fs -find /testing/*.sh
/testing/hadoop-e.sh
/testing/hadoop-env.sh
```

- To display the usage for given command or all commands if none is specified use –usage.

```
C:\WINDOWS\system32>hadoop fs -usage test
Usage: hadoop fs [generic options] -test -[defsz] <path>
```

- To create a file of zero length at <path> with current time as the timestamp of that <path> use -touchz.
- An error is returned if the file exists with non-zero length

```
C:\WINDOWS\system32>hadoop fs -touchz /test/Samp.txt

C:\WINDOWS\system32>hadoop fs -ls /test
Found 3 items
-rw-r--r--   1 raoal supergroup          0 2019-09-02 22:11 /test/Samp.txt
-rw-r--r--   1 raoal supergroup         14 2019-08-27 13:26 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2019-08-28 14:11 /test/subdir
```

```
C:\WINDOWS\system32>hadoop fs -touchz /test/Sample.txt
touchz: `/test/Sample.txt': Not a zero-length file
```

- To truncate all files that match the specified file pattern to the specified length use –truncate.
- The length has to be specified.
  - -w --- Requests that the command wait for block recovery to complete, if necessary.

```
C:\WINDOWS\system32>hadoop fs -truncate 10 /test/Sample.txt
Truncating /test/Sample.txt to length: 10. Wait for block recovery to complete before furth
er updating this file.
```

```
C:\WINDOWS\system32>hadoop fs -ls /test
Found 3 items
-rw-r--r--   1 raoal supergroup          0 2019-09-02 22:11 /test/Samp.txt
-rw-r--r--   1 raoal supergroup         10 2019-09-03 08:26 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2019-08-28 14:11 /test/subdir
```

```
C:\WINDOWS\system32>hadoop fs -cat /test/Sample.txt
Hello Worl
```

```
C:\WINDOWS\system32>hadoop fs -truncate -w 8 /test/Sample.txt
Waiting for /test/Sample.txt ...
Truncated /test/Sample.txt to length: 8
```

```
C:\WINDOWS\system32>hadoop fs -cat /test/Sample.txt
Hello Wo
C:\WINDOWS\system32>hadoop fs -ls /test
Found 3 items
-rw-r--r--   1 raoal supergroup          0 2019-09-02 22:11 /test/Samp.txt
-rw-r--r--   1 raoal supergroup          8 2019-09-03 08:29 /test/Sample.txt
drwxr-xr-x   - raoal supergroup          0 2019-08-28 14:11 /test/subdir
```

- To take a source file and outputs the file in text format use –text.
- The allowed formats are zip and TextRecordInputStream and Avro.

```
C:\WINDOWS\system32>hadoop fs -text /testing/core-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
   <property>
     <name>fs.default.name</name>
     <value>hdfs://0.0.0.0:19000</value>
   </property>
</configuration>
```

- To set the replication level of a file use -setrep command.
- If <path> is a directory then the command recursively changes the replication factor of all files under the directory tree rooted at <path>.
  - -w --- It requests that the command waits for the replication to complete.
    - ⇒ This can potentially take a very long time.
  - -R --- It is accepted for backwards compatibility.
    - ⇒ It has no effect.

```
C:\WINDOWS\system32>hadoop fs -setrep 2 /testing/Sample.txt
Replication 2 set: /testing/Sample.txt

C:\WINDOWS\system32>hadoop fs -ls /testing
Found 5 items
-rw-r--r--   2 raoal supergroup         40 2019-08-29 10:12 /testing/Sample.txt
-rw-r--r--   1 raoal supergroup        901 2019-08-28 14:00 /testing/core-site.xml
-rw-r--r--   1 raoal supergroup       4969 2019-08-28 14:11 /testing/hadoop-e.sh
-rw-r--r--   1 raoal supergroup       4969 2019-08-27 22:10 /testing/hadoop-env.sh
drwxr-xr-x   - raoal supergroup          0 2019-08-27 21:48 /testing/subdir
```

```
C:\WINDOWS\system32>hadoop fs -setrep -w 2 /testing/Sample.txt
Replication 2 set: /testing/Sample.txt
Waiting for /testing/Sample.txt ..............................
```

```
C:\WINDOWS\system32>hadoop fs -setrep -w 2 /testing/Sample.txt
Replication 2 set: /testing/Sample.txt
Waiting for /testing/Sample.txt ........................................
```

```
C:\WINDOWS\system32>hadoop fs -setrep -w 2 /testing/Sample.txt
Replication 2 set: /testing/Sample.txt
Waiting for /testing/Sample.txt ..........................................................
.......................................
```

- To set an extended attribute name and value for a file or directory use –setfattr command.
- -setfattr {-n name [-v value] | -x name} <path> :
  - -n name --- The extended attribute name.
  - -v value --- The extended attribute value.
- There are three different encoding methods for the value.
  - If the argument is enclosed in double quotes, then the value is the string inside the quotes.
  - If the argument is prefixed with 0x or 0X, then it is taken as a hexadecimal number.
  - If the argument begins with 0s or 0S, then it is taken as a base64 encoding.
- -x name --- Remove the extended attribute.
- <path> --- The file or directory.

```
C:\WINDOWS\system32>hadoop fs -setfattr -n user.encoding -v "UTF-8" /test/Sample.txt
```

- To display the extended attribute names and values (if any) for a file or directory use –getfattr command.
- -getfattr [-R] {-n name | -d} [-e en] <path>
  - -R --- Recursively list the attributes for all files and directories.
  - -n name --- Dump the named extended attribute value.
  - -d --- Dump all extended attribute values associated with pathname.
  - -e <encoding> --- Encode values after retrieving them.
    - ⇒ Valid encodings are "text", "hex", and "base64".
    - ⇒ Values encoded as text strings are enclosed in double quotes ("), and values encoded as hexadecimal and base64 are prefixed with 0x and 0s, respectively.
- <path> --- The file or directory.

```
C:\WINDOWS\system32>hadoop fs -getfattr -n user.encoding /test/Sample.txt
# file: /test/Sample.txt
user.encoding="UTF-8"
```

```
C:\WINDOWS\system32>hadoop fs -getfattr -R -n user.encoding /test/Sample.txt
# file: /test/Sample.txt
user.encoding="UTF-8"
```

```
C:\WINDOWS\system32>hadoop fs -getfattr -d /test/Samp.txt
# file: /test/Samp.txt
```

- To remove the directory entry specified by each directory argument, provided it is empty use –rmdir command.
- -rmdir [--ignore-fail-on-non-empty] <dir>

```
C:\WINDOWS\system32>hadoop fs -rmdir /test/subdir/subdir2/subdir3
rmdir: `/test/subdir/subdir2/subdir3': Directory is not empty

C:\WINDOWS\system32>hadoop fs -rmdir --ignore-fail-on-non-empty /test/subdir/subdir2/subdir3

C:\WINDOWS\system32>hadoop fs -ls /test/subdir/subdir2/subdir3
Found 1 items
-rw-r--r--   1 raoal supergroup          0 2019-09-03 10:05 /test/subdir/subdir2/subdir3/Samp.txt
```

- To show the capacity, free and used space of the filesystem use –df command.
- If the filesystem has multiple partitions, and no path to a particular partition is specified, then the status of the root partitions will be shown.
  - -h --- Formats the sizes of files in a human-readable fashion rather than a number of bytes.

```
C:\WINDOWS\system32>hadoop fs -df /test
Filesystem                    Size   Used     Available  Use%
hdfs://0.0.0.0:19000  502703058944  11342  302387425280    0%
```

```
C:\WINDOWS\system32>hadoop fs -df -h /test
Filesystem                Size    Used  Available  Use%
hdfs://0.0.0.0:19000  468.2 G  11.1 K    281.6 G    0%
```