Name: Umer Farooq

Assignment: 1

Topic Title: Linux Commands

❖ Linux Commands:

A Linux command is a program or utility that runs on the command line. A command line is an interface that accepts lines of text and processes them into instructions for your computer. Any graphical user interface (GUI) is just an abstraction of command-line programs.

Here’s what a Linux command’s general syntax looks like:

```
CommandName [option(s)] [parameter(s)]
```

Here the **command name** is the rule you want to perform.

[Options] it modifies the command operation. To invoke it, use hyphen(-).

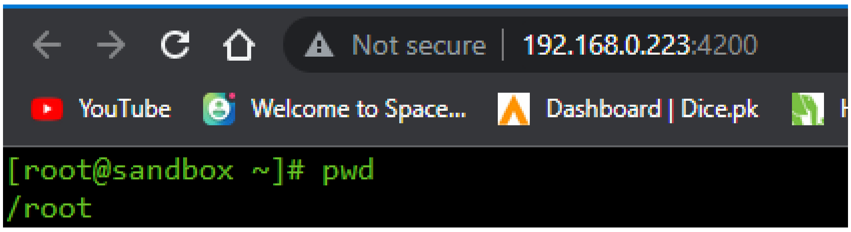
❖ 15 Linux Commands:

Here is the list of fundamental/basic linux commands:

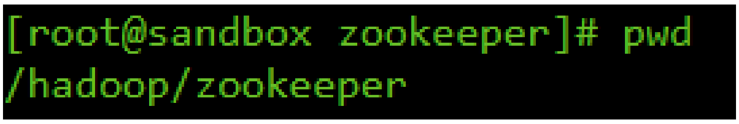
1. pwd command
2. ls command
3. cd command:
4. mkdir command:
5. rmdir command:
6. df command:
7. du command:
8. head command:
9. tail command:
10. jobs command:
11. ping command:
12. top command:
13. history command:
14. man command:
15. echo command:

1.pwd command:

Use the **pwd command** to find the path of your **current working directory**. Simply entering pwd will return the full current path – a path of all the directories that starts with a forward slash (/).



```
[root@sandbox ~]# pwd
/root
```

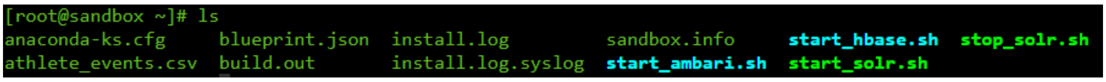


```
[root@sandbox zookeeper]# pwd
/hadoop/zookeeper
```

2.ls command:

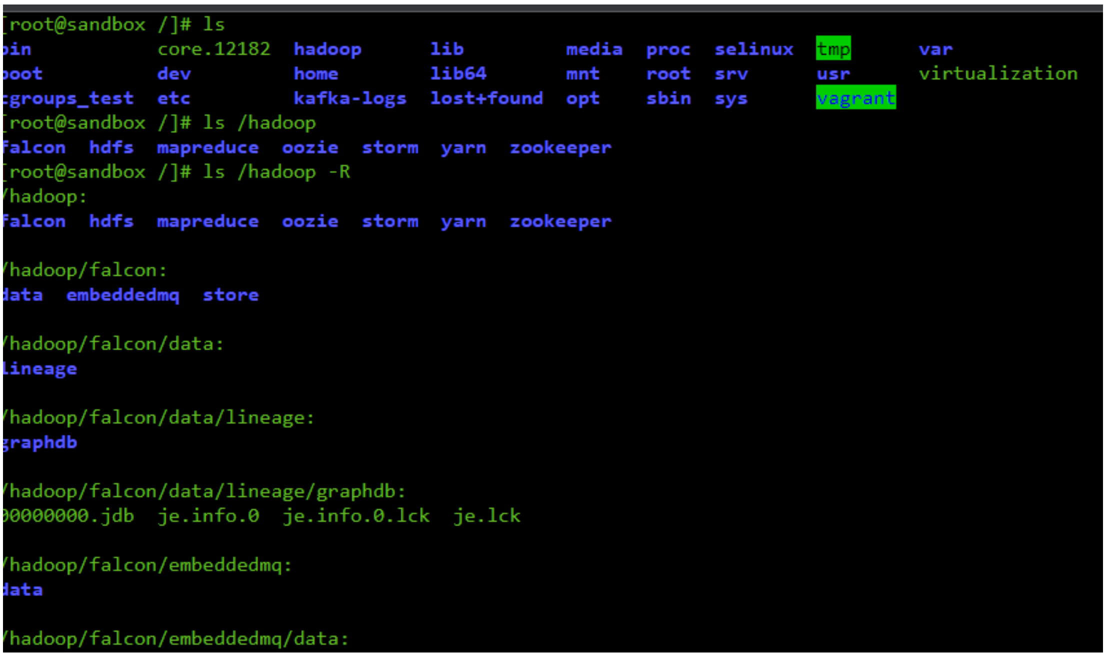
The ls command lists files and directories within a system.

Running it without a flag or parameter will show the current working directory's content.



```
[root@sandbox ~]# ls
anaconda-ks.cfg      blueprint.json  install.log         sandbox.info     start_hbase.sh  stop_solr.sh
athlete_events.csv   build.out       install.log.syslog  start_ambari.sh  start_solr.sh
```

To see other directories' content, type ls followed by the desired path. For example, to view files in the hadoop folder, **enter: ls /hadoop**

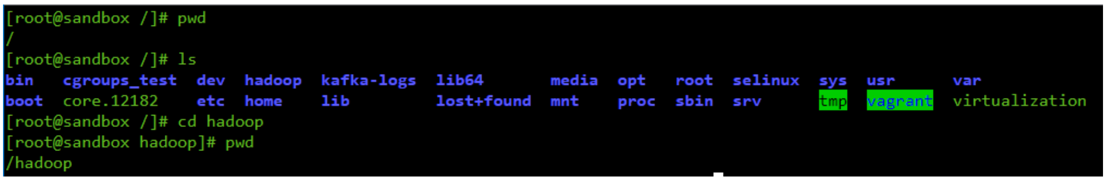


```
[root@sandbox /]# ls
bin           core.12182  hadoop      lib         media  proc  selinux  tmp      var
boot          dev         home        lib64       mnt    root  srv      usr      virtualization
cgroups_test  etc         kafka-logs  lost+found  opt    sbin  sys      vagrant
[root@sandbox /]# ls /hadoop
falcon  hdfs  mapreduce  oozie  storm  yarn  zookeeper
[root@sandbox /]# ls /hadoop -R
/hadoop:
falcon  hdfs  mapreduce  oozie  storm  yarn  zookeeper

/hadoop/falcon:
data  embeddedmq  store

/hadoop/falcon/data:
lineage

/hadoop/falcon/data/lineage:
graphdb

/hadoop/falcon/data/lineage/graphdb:
00000000.jdb  je.info.0  je.info.0.lck  je.lck

/hadoop/falcon/embeddedmq:
data

/hadoop/falcon/embeddedmq/data:
```

ls /hadoop -R: lists all the files in the subdirectories.

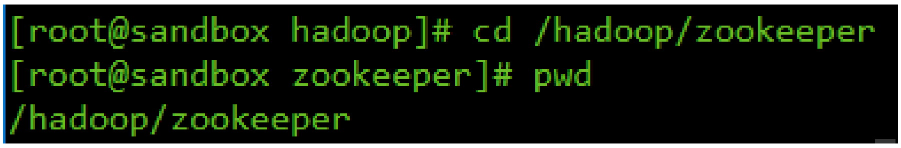
3.cd command:

To navigate through the files and directories, use the **cd command**. Depending on your current working directory, it requires either the full path or the directory name.



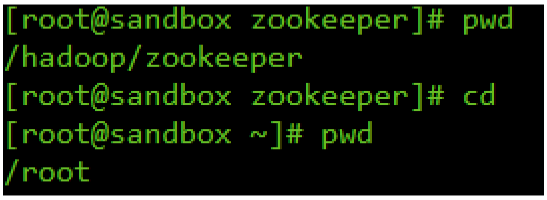
```
[root@sandbox /]# pwd
/
[root@sandbox /]# ls
bin   cgroups_test  dev  hadoop  kafka-logs  lib64       media  opt   root  selinux  sys  usr      var
boot  core.12182    etc  home    lib         lost+found  mnt    proc  sbin  srv      tmp  vagrant  virtualization
[root@sandbox /]# cd hadoop
[root@sandbox hadoop]# pwd
/hadoop
```

If you want to move to zookeeper, a sub category, then you can enter command like this:



```
[root@sandbox hadoop]# cd /hadoop/zookeeper
[root@sandbox zookeeper]# pwd
/hadoop/zookeeper
```

If you execute cd only, you will be moved to the home directory, in this case its root.



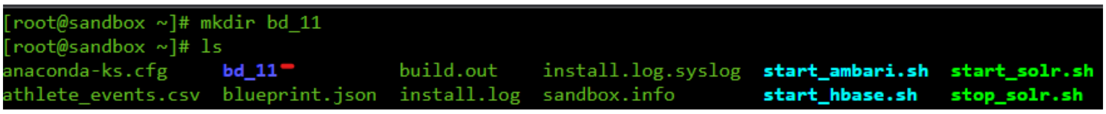
```
[root@sandbox zookeeper]# pwd
/hadoop/zookeeper
[root@sandbox zookeeper]# cd
[root@sandbox ~]# pwd
/root
```

cd .. moves one directory up.

4.<u>mkdir command:</u>

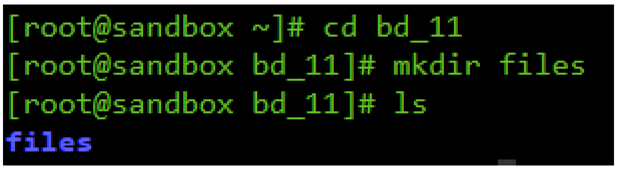
Use the mkdir command to create directories at once and set permissions for each of them.

The user executing this command must have the **privilege to make a new folder** in the parent directory, or they may receive a **permission denied error**.



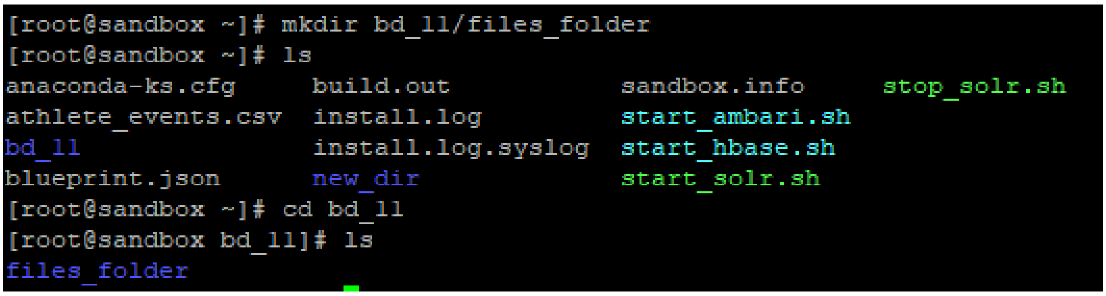
```
[root@sandbox ~]# mkdir bd_11
[root@sandbox ~]# ls
anaconda-ks.cfg     bd_11           build.out    install.log.syslog  start_ambari.sh  start_solr.sh
athlete_events.csv  blueprint.json  install.log  sandbox.info        start_hbase.sh   stop_solr.sh
```

Creating a subdirectory within bd_11:



```
[root@sandbox ~]# cd bd_11
[root@sandbox bd_11]# mkdir files
[root@sandbox bd_11]# ls
files
```

Or we can also create subdirectory as:

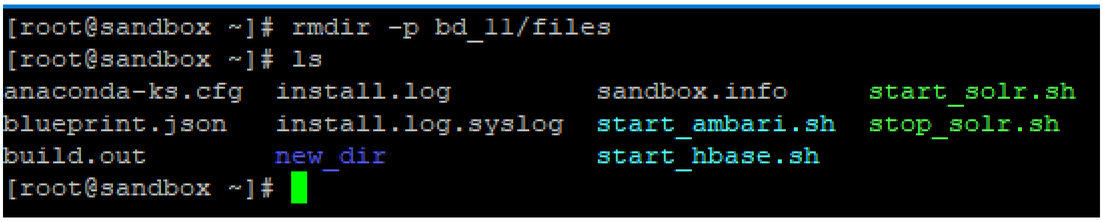


```
[root@sandbox ~]# mkdir bd_11/files_folder
[root@sandbox ~]# ls
anaconda-ks.cfg     build.out           sandbox.info     stop_solr.sh
athlete_events.csv  install.log         start_ambari.sh
bd_11               install.log.syslog  start_hbase.sh
blueprint.json      new_dir             start_solr.sh
[root@sandbox ~]# cd bd_11
[root@sandbox bd_11]# ls
files_folder
```

5.rmdir command:

To permanently delete an empty directory, use the rmdir command.

For example, you want to remove an empty subdirectory named files_folder and its main folder bd_11:



```
[root@sandbox ~]# rmdir -p bd_11/files
[root@sandbox ~]# ls
anaconda-ks.cfg  install.log         sandbox.info     start_solr.sh
blueprint.json   install.log.syslog  start_ambari.sh  stop_solr.sh
build.out        new_dir             start_hbase.sh
[root@sandbox ~]# 
```

6.rm command:

The rm command is used to delete files within a directory. Make sure that the user performing this command has write permissions.

Remember the directory's location as this will remove the file(s) and you can't undo it.

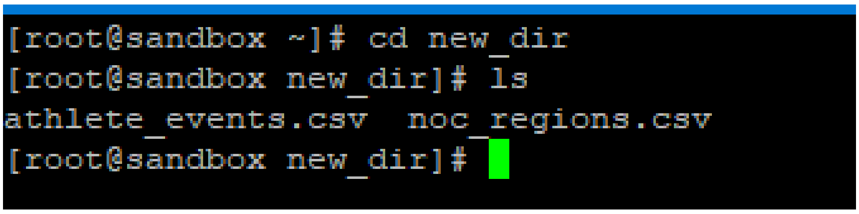
- **Here's the general syntax:**

rm filename

- **To remove multiple files**, enter the following command:

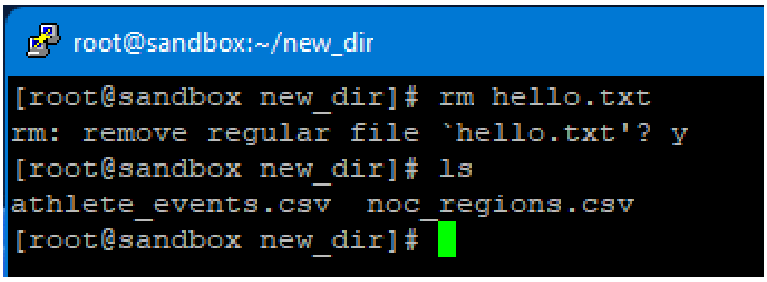
rm filename1 filename2 filename3

I have a file in new_dir, athlethe_events.csv & noc.csv.



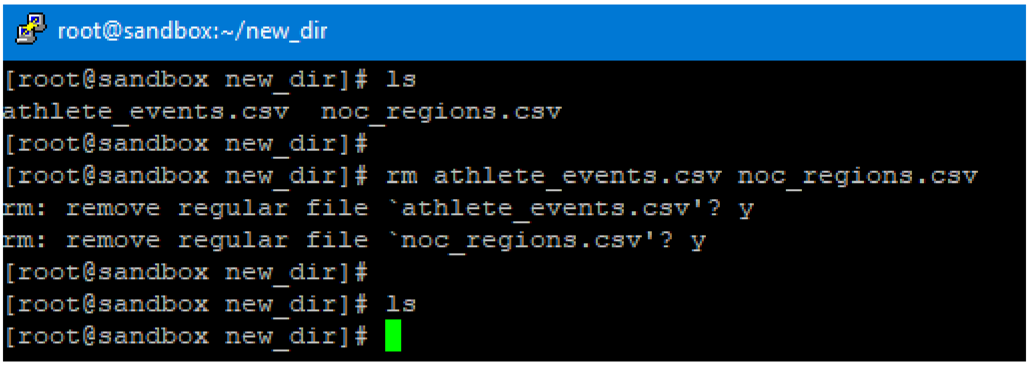
```
[root@sandbox ~]# cd new_dir
[root@sandbox new_dir]# ls
athlete_events.csv  noc_regions.csv
[root@sandbox new_dir]#
```

Deleting a single file:

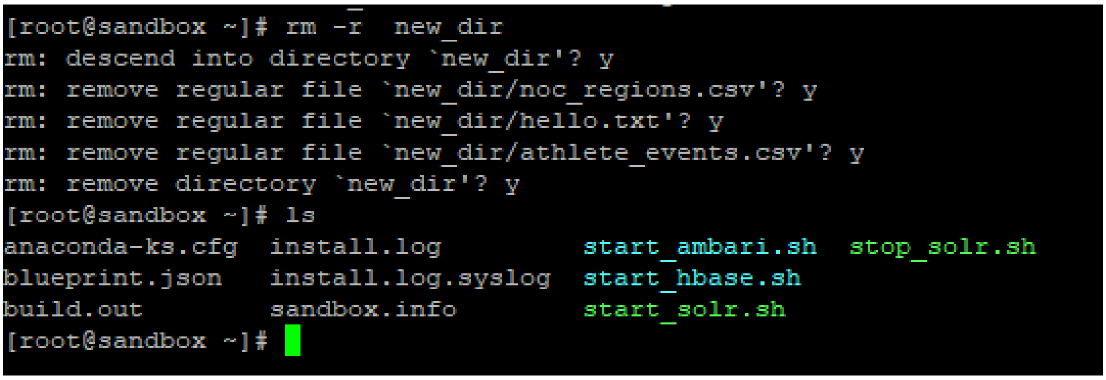


```
[root@sandbox new_dir]# rm hello.txt
rm: remove regular file `hello.txt'? y
[root@sandbox new_dir]# ls
athlete_events.csv  noc_regions.csv
[root@sandbox new_dir]#
```

Deleting multiple files:



```
[root@sandbox new_dir]# ls
athlete_events.csv  noc_regions.csv
[root@sandbox new_dir]#
[root@sandbox new_dir]# rm athlete_events.csv noc_regions.csv
rm: remove regular file `athlete_events.csv'? y
rm: remove regular file `noc_regions.csv'? y
[root@sandbox new_dir]#
[root@sandbox new_dir]# ls
[root@sandbox new_dir]#
```



```
[root@sandbox ~]# rm -r  new_dir
rm: descend into directory `new_dir'? y
rm: remove regular file `new_dir/noc_regions.csv'? y
rm: remove regular file `new_dir/hello.txt'? y
rm: remove regular file `new_dir/athlete_events.csv'? y
rm: remove directory `new_dir'? y
[root@sandbox ~]# ls
anaconda-ks.cfg  install.log         start_ambari.sh  stop_solr.sh
blueprint.json   install.log.syslog  start_hbase.sh
build.out        sandbox.info        start_solr.sh
[root@sandbox ~]#
```

7

-r deletes files and directories recursively.

7.df command:

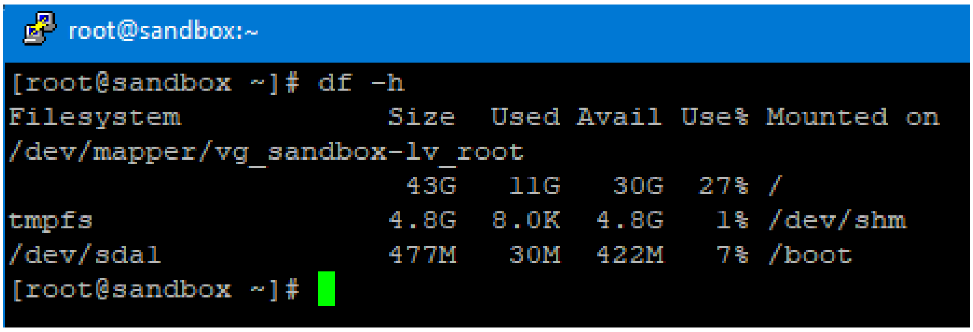
Use the df command to report the system's disk space usage.

Here's the general syntax:

df [options] [file]

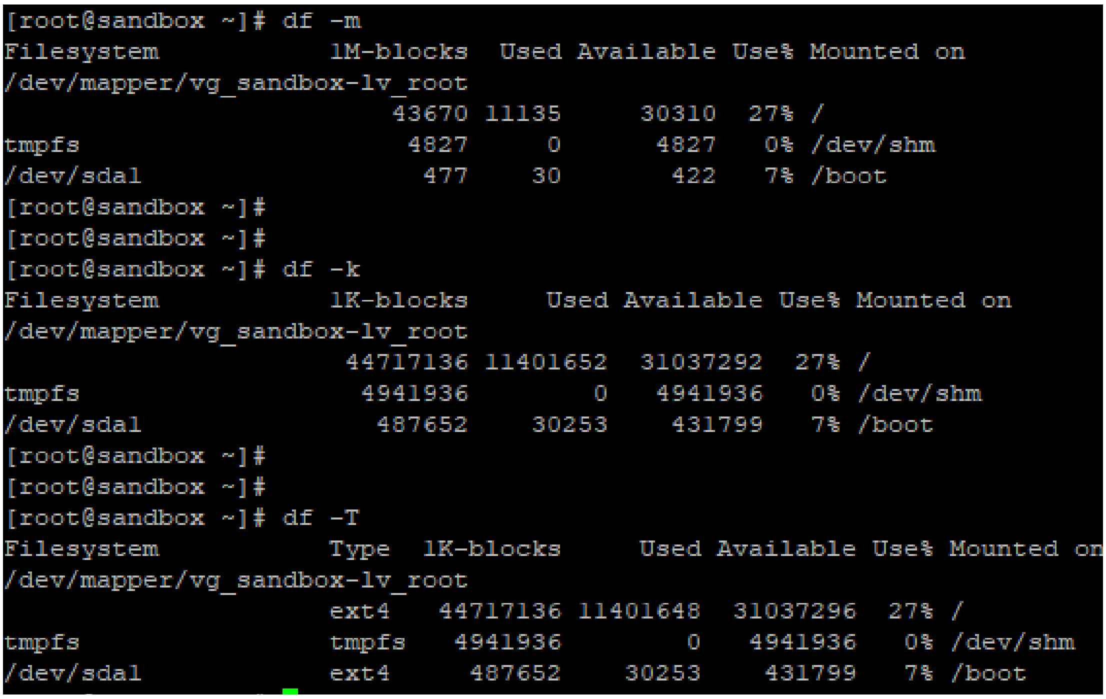
For example, enter the following command if you want to see the current directory's system disk space usage in a human-readable format:

df -h



```
[root@sandbox ~]# df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/vg_sandbox-lv_root
                       43G   11G   30G  27% /
tmpfs                 4.8G  8.0K  4.8G   1% /dev/shm
/dev/sda1             477M   30M  422M   7% /boot
[root@sandbox ~]# 
```

There are different options we can use with df such as -m, -k, & -T:

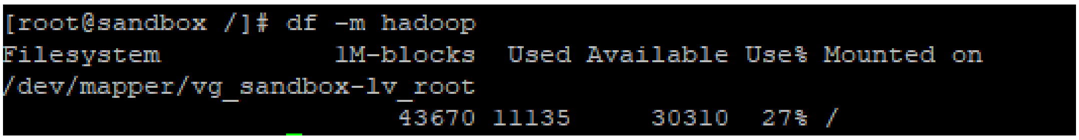


```
[root@sandbox ~]# df -m
Filesystem           1M-blocks  Used Available Use% Mounted on
/dev/mapper/vg_sandbox-lv_root
                         43670 11135     30310  27% /
tmpfs                     4827     0      4827   0% /dev/shm
/dev/sda1                  477    30       422   7% /boot
[root@sandbox ~]#
[root@sandbox ~]#
[root@sandbox ~]# df -k
Filesystem           1K-blocks     Used Available Use% Mounted on
/dev/mapper/vg_sandbox-lv_root
                      44717136 11401652  31037292  27% /
tmpfs                  4941936        0   4941936   0% /dev/shm
/dev/sda1               487652    30253    431799   7% /boot
[root@sandbox ~]#
[root@sandbox ~]#
[root@sandbox ~]# df -T
Filesystem           Type   1K-blocks     Used Available Use% Mounted on
/dev/mapper/vg_sandbox-lv_root
                     ext4    44717136 11401648  31037296  27% /
tmpfs                tmpfs    4941936        0   4941936   0% /dev/shm
/dev/sda1            ext4      487652    30253    431799   7% /boot
```

df -m displays information on the file system usage in MBs.

df -k displays file system usage in KBs.

df -T shows the file system type in a new column.

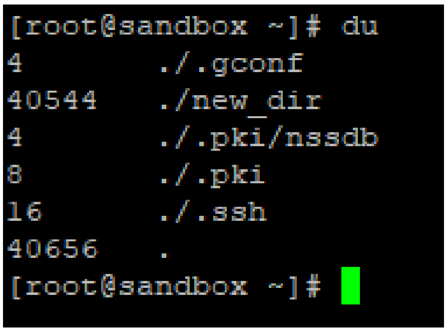


```
[root@sandbox /]# df -m hadoop
Filesystem           1M-blocks  Used Available Use% Mounted on
/dev/mapper/vg_sandbox-lv_root
                         43670 11135     30310  27% /
```

8.du command:

If you want to check how much space a file or a directory takes up, use the du command.

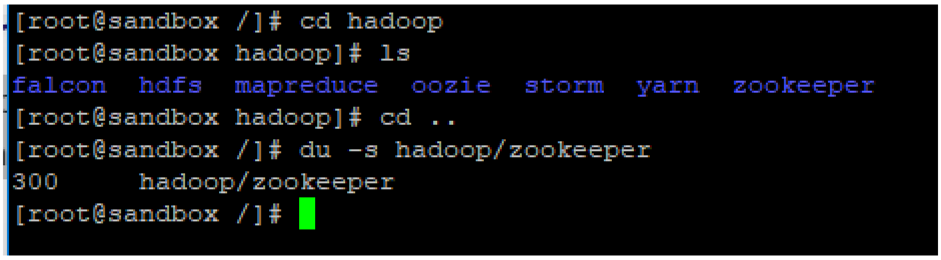
You can run this command to identify which part of the system uses the storage excessively.



```
[root@sandbox ~]# du
4       ././.gconf
40544   ./new_dir
4       ././.pki/nssdb
8       ././.pki
16      ././.ssh
40656   .
[root@sandbox ~]#
```

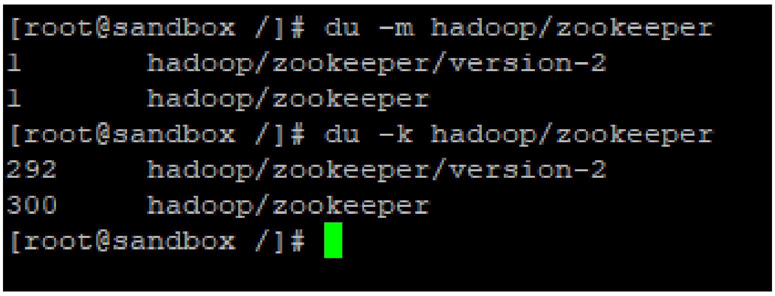
Checking the total disk usage by hadoop/zookeeper:

-s offers the total size of a specified folder.



```
[root@sandbox /]# cd hadoop
[root@sandbox hadoop]# ls
falcon  hdfs  mapreduce  oozie  storm  yarn  zookeeper
[root@sandbox hadoop]# cd ..
[root@sandbox /]# du -s hadoop/zookeeper
300     hadoop/zookeeper
[root@sandbox /]#
```

We can also check the size of folder in MBs & KBs:



```
[root@sandbox /]# du -m hadoop/zookeeper
1       hadoop/zookeeper/version-2
1       hadoop/zookeeper
[root@sandbox /]# du -k hadoop/zookeeper
292     hadoop/zookeeper/version-2
300     hadoop/zookeeper
[root@sandbox /]#
```

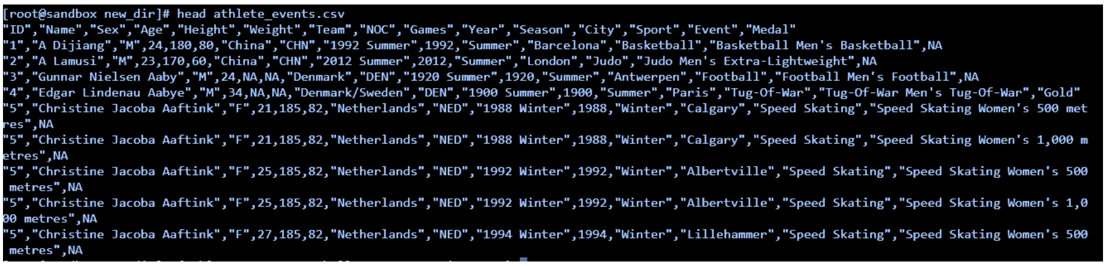
9.head command:

The head command allows you to view the first ten lines of a text.

Adding an option lets you change the number of lines shown. The head command is also used to output piped data to the CLI.

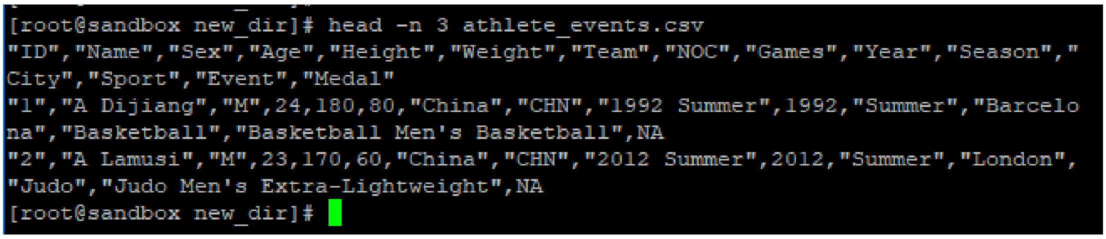
Here’s the general syntax:

head [option] [file]



```
[root@sandbox new_dir]# head athlete_events.csv
"ID","Name","Sex","Age","Height","Weight","Team","NOC","Games","Year","Season","City","Sport","Event","Medal"
"1","A Dijiang","M",24,180,80,"China","CHN","1992 Summer",1992,"Summer","Barcelona","Basketball","Basketball Men's Basketball",NA
"2","A Lamusi","M",23,170,60,"China","CHN","2012 Summer",2012,"Summer","London","Judo","Judo Men's Extra-Lightweight",NA
"3","Gunnar Nielsen Aaby","M",24,NA,NA,"Denmark","DEN","1920 Summer",1920,"Summer","Antwerpen","Football","Football Men's Football",NA
"4","Edgar Lindenau Aabye","M",34,NA,NA,"Denmark/Sweden","DEN","1900 Summer",1900,"Summer","Paris","Tug-Of-War","Tug-Of-War Men's Tug-Of-War","Gold"
"5","Christine Jacoba Aaftink","F",21,185,82,"Netherlands","NED","1988 Winter",1988,"Winter","Calgary","Speed Skating","Speed Skating Women's 500 met
res",NA
"5","Christine Jacoba Aaftink","F",21,185,82,"Netherlands","NED","1988 Winter",1988,"Winter","Calgary","Speed Skating","Speed Skating Women's 1,000 m
etres",NA
"5","Christine Jacoba Aaftink","F",25,185,82,"Netherlands","NED","1992 Winter",1992,"Winter","Albertville","Speed Skating","Speed Skating Women's 500
 metres",NA
"5","Christine Jacoba Aaftink","F",25,185,82,"Netherlands","NED","1992 Winter",1992,"Winter","Albertville","Speed Skating","Speed Skating Women's 1,0
00 metres",NA
"5","Christine Jacoba Aaftink","F",27,185,82,"Netherlands","NED","1994 Winter",1994,"Winter","Lillehammer","Speed Skating","Speed Skating Women's 500
 metres",NA
```

Lets print first 3 rows of athlete_events.csv



```
[root@sandbox new_dir]# head -n 3 athlete_events.csv
"ID","Name","Sex","Age","Height","Weight","Team","NOC","Games","Year","Season","
City","Sport","Event","Medal"
"1","A Dijiang","M",24,180,80,"China","CHN","1992 Summer",1992,"Summer","Barcelo
na","Basketball","Basketball Men's Basketball",NA
"2","A Lamusi","M",23,170,60,"China","CHN","2012 Summer",2012,"Summer","London",
"Judo","Judo Men's Extra-Lightweight",NA
[root@sandbox new_dir]#
```

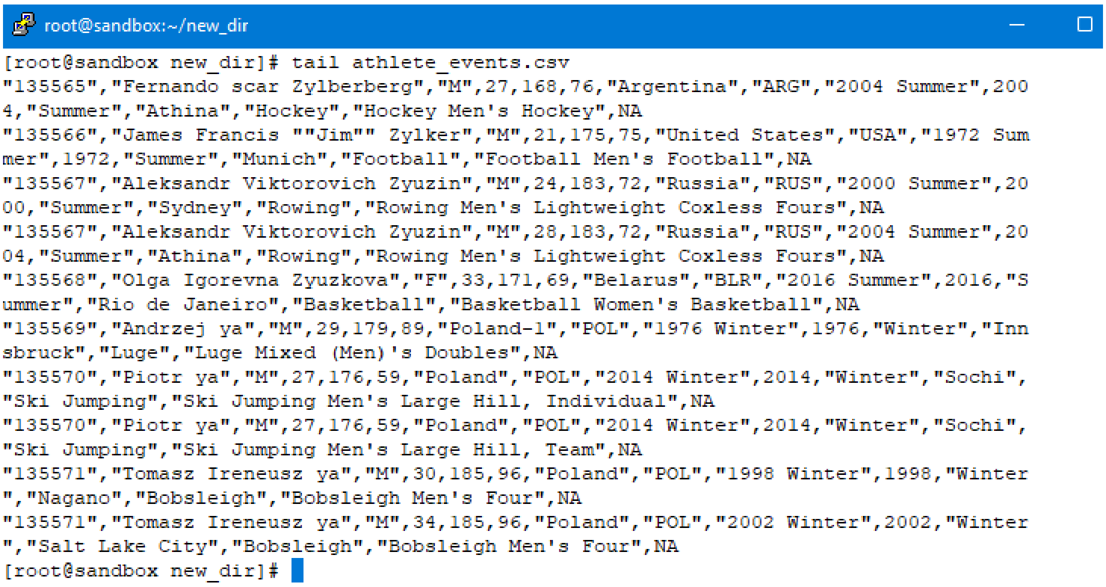
10. tail command:

The tail command displays the last ten lines of a file. It allows users to check whether a file has new data or to read error messages.

Here’s the general format:

tail [option] [file]

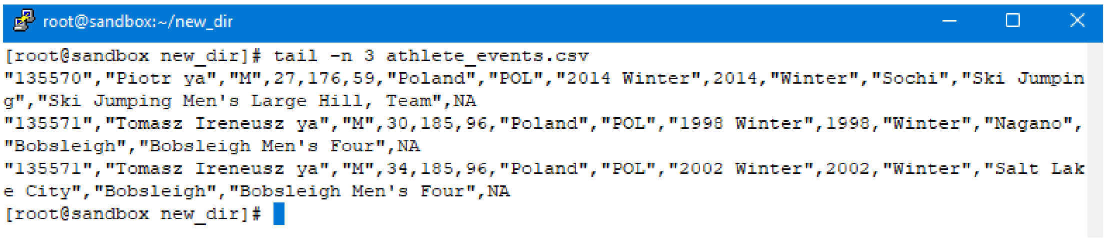
Displaying the last ten lines of athlete_events.csv



root@sandbox:~/new_dir

```
[root@sandbox new_dir]# tail athlete_events.csv
"135565","Fernando scar Zylberberg","M",27,168,76,"Argentina","ARG","2004 Summer",200
4,"Summer","Athina","Hockey","Hockey Men's Hockey",NA
"135566","James Francis ""Jim"" Zylker","M",21,175,75,"United States","USA","1972 Sum
mer",1972,"Summer","Munich","Football","Football Men's Football",NA
"135567","Aleksandr Viktorovich Zyuzin","M",24,183,72,"Russia","RUS","2000 Summer",20
00,"Summer","Sydney","Rowing","Rowing Men's Lightweight Coxless Fours",NA
"135567","Aleksandr Viktorovich Zyuzin","M",28,183,72,"Russia","RUS","2004 Summer",20
04,"Summer","Athina","Rowing","Rowing Men's Lightweight Coxless Fours",NA
"135568","Olga Igorevna Zyuzkova","F",33,171,69,"Belarus","BLR","2016 Summer",2016,"S
ummer","Rio de Janeiro","Basketball","Basketball Women's Basketball",NA
"135569","Andrzej ya","M",29,179,89,"Poland-1","POL","1976 Winter",1976,"Winter","Inn
sbruck","Luge","Luge Mixed (Men)'s Doubles",NA
"135570","Piotr ya","M",27,176,59,"Poland","POL","2014 Winter",2014,"Winter","Sochi",
"Ski Jumping","Ski Jumping Men's Large Hill, Individual",NA
"135570","Piotr ya","M",27,176,59,"Poland","POL","2014 Winter",2014,"Winter","Sochi",
"Ski Jumping","Ski Jumping Men's Large Hill, Team",NA
"135571","Tomasz Ireneusz ya","M",30,185,96,"Poland","POL","1998 Winter",1998,"Winter
","Nagano","Bobsleigh","Bobsleigh Men's Four",NA
"135571","Tomasz Ireneusz ya","M",34,185,96,"Poland","POL","2002 Winter",2002,"Winter
","Salt Lake City","Bobsleigh","Bobsleigh Men's Four",NA
[root@sandbox new_dir]#
```

Display last 3 lines of file:



root@sandbox:~/new_dir

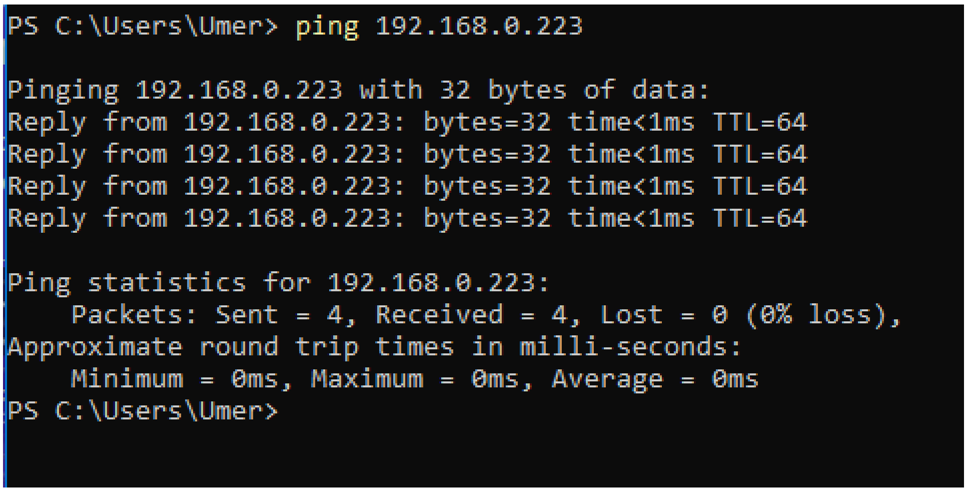
```
[root@sandbox new_dir]# tail -n 3 athlete_events.csv
"135570","Piotr ya","M",27,176,59,"Poland","POL","2014 Winter",2014,"Winter","Sochi","Ski Jumpin
g","Ski Jumping Men's Large Hill, Team",NA
"135571","Tomasz Ireneusz ya","M",30,185,96,"Poland","POL","1998 Winter",1998,"Winter","Nagano",
"Bobsleigh","Bobsleigh Men's Four",NA
"135571","Tomasz Ireneusz ya","M",34,185,96,"Poland","POL","2002 Winter",2002,"Winter","Salt Lak
e City","Bobsleigh","Bobsleigh Men's Four",NA
[root@sandbox new_dir]#
```

11. ping command:

The ping command is one of the most used basic Linux commands for checking whether a network or a server is reachable. In addition, it is used to troubleshoot various connectivity issues.

Here’s the general format:

ping [option] [hostname_or_IP_address]

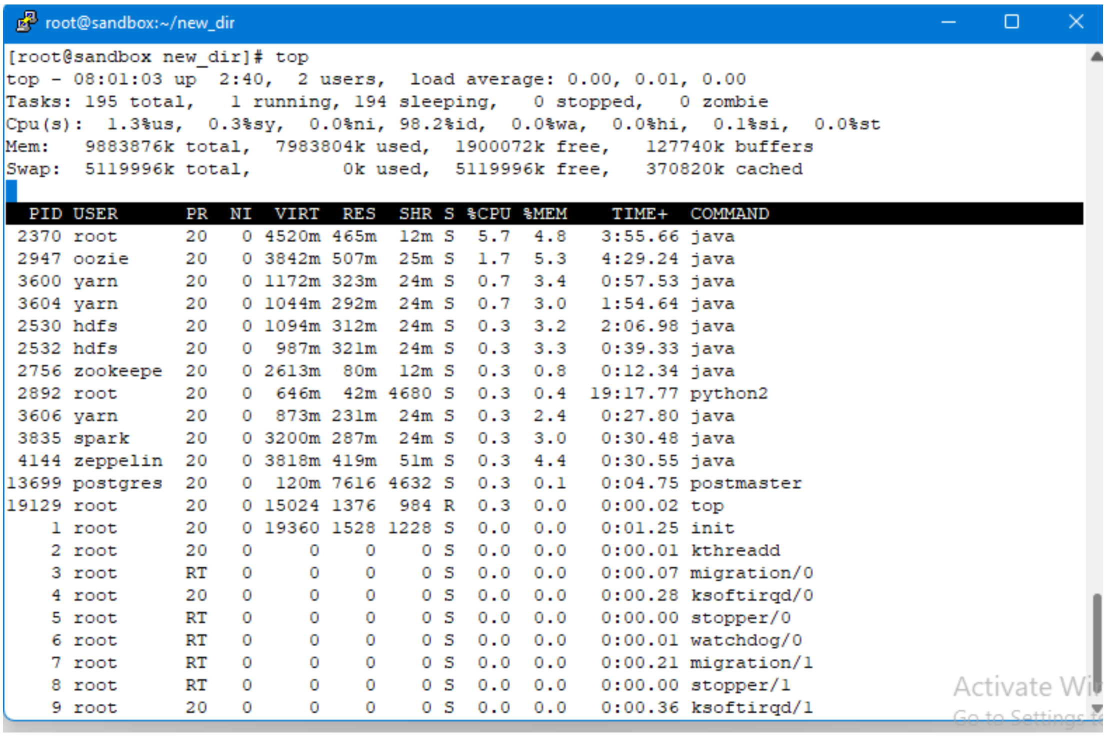


```
PS C:\Users\Umer> ping 192.168.0.223

Pinging 192.168.0.223 with 32 bytes of data:
Reply from 192.168.0.223: bytes=32 time<1ms TTL=64
Reply from 192.168.0.223: bytes=32 time<1ms TTL=64
Reply from 192.168.0.223: bytes=32 time<1ms TTL=64
Reply from 192.168.0.223: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.0.223:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms
PS C:\Users\Umer>
```

12. top command:

The top command in Linux Terminal will **display all the running processes and a dynamic real-time view** of the current system. It sums up the resource utilisation, from CPU to memory usage.

The **top command** can also help you identify and terminate a process that may use too many system resources.



```
[root@sandbox new_dir]# top
top - 08:01:03 up  2:40,  2 users,  load average: 0.00, 0.01, 0.00
Tasks: 195 total,   1 running, 194 sleeping,   0 stopped,   0 zombie
Cpu(s):  1.3%us,  0.3%sy,  0.0%ni, 98.2%id,  0.0%wa,  0.0%hi,  0.1%si,  0.0%st
Mem:   9883876k total,  7983804k used,  1900072k free,   127740k buffers
Swap:  5119996k total,        0k used,  5119996k free,   370820k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2370 root      20   0 4520m 465m  12m S  5.7  4.8   3:55.66 java
 2947 oozie     20   0 3842m 507m  25m S  1.7  5.3   4:29.24 java
 3600 yarn      20   0 1172m 323m  24m S  0.7  3.4   0:57.53 java
 3604 yarn      20   0 1044m 292m  24m S  0.7  3.0   1:54.64 java
 2530 hdfs      20   0 1094m 312m  24m S  0.3  3.2   2:06.98 java
 2532 hdfs      20   0  987m 321m  24m S  0.3  3.3   0:39.33 java
 2756 zookeepe  20   0 2613m  80m  12m S  0.3  0.8   0:12.34 java
 2892 root      20   0  646m  42m 4680 S  0.3  0.4  19:17.77 python2
 3606 yarn      20   0  873m 231m  24m S  0.3  2.4   0:27.80 java
 3835 spark     20   0 3200m 287m  24m S  0.3  3.0   0:30.48 java
 4144 zeppelin  20   0 3818m 419m  51m S  0.3  4.4   0:30.55 java
13699 postgres  20   0  120m 7616 4632 S  0.3  0.1   0:04.75 postmaster
19129 root      20   0 15024 1376  984 R  0.3  0.0   0:00.02 top
    1 root      20   0 19360 1528 1228 S  0.0  0.0   0:01.25 init
    2 root      20   0     0    0    0 S  0.0  0.0   0:00.01 kthreadd
    3 root      RT   0     0    0    0 S  0.0  0.0   0:00.07 migration/0
    4 root      20   0     0    0    0 S  0.0  0.0   0:00.28 ksoftirqd/0
    5 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/0
    6 root      RT   0     0    0    0 S  0.0  0.0   0:00.01 watchdog/0
    7 root      RT   0     0    0    0 S  0.0  0.0   0:00.21 migration/1
    8 root      RT   0     0    0    0 S  0.0  0.0   0:00.00 stopper/1
    9 root      20   0     0    0    0 S  0.0  0.0   0:00.36 ksoftirqd/1
```

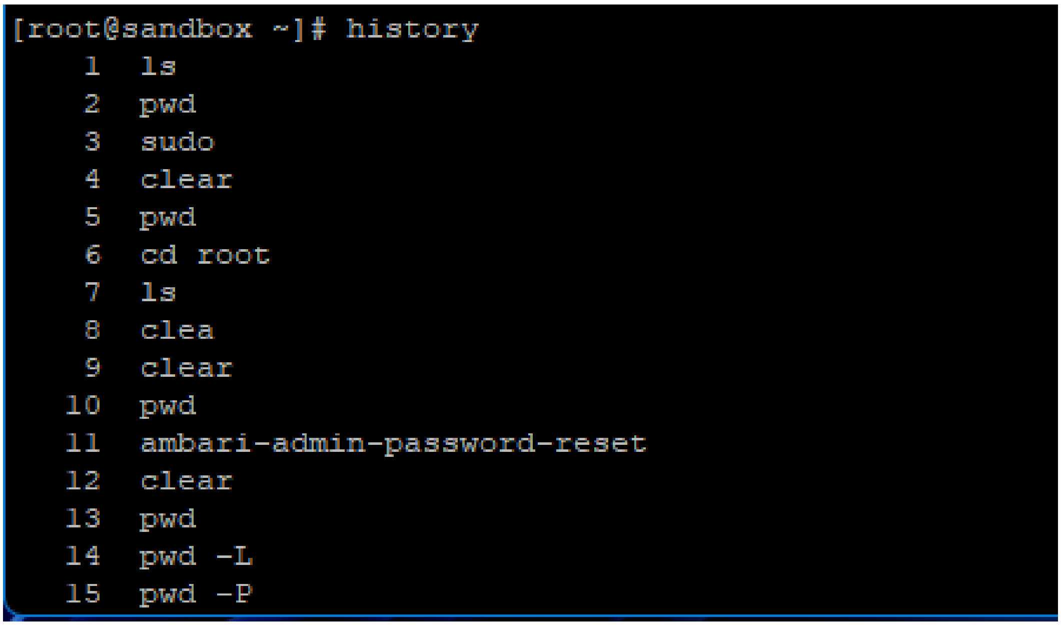
13. history command:

With a history command, **the system will list up to 500 previously executed commands**, allowing you to reuse them without re-entering.

Users with sudo privileges can execute this command.

To run it, enter the command below:

history [option]

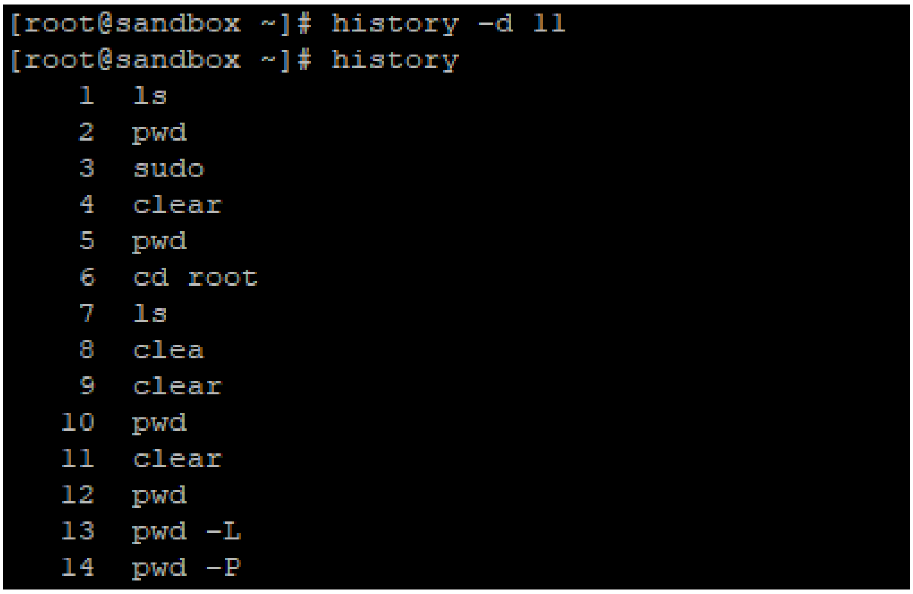


```
[root@sandbox ~]# history
    1  ls
    2  pwd
    3  sudo
    4  clear
    5  pwd
    6  cd root
    7  ls
    8  clea
    9  clear
   10  pwd
   11  ambari-admin-password-reset
   12  clear
   13  pwd
   14  pwd -L
   15  pwd -P
```

This command supports many options, such as:

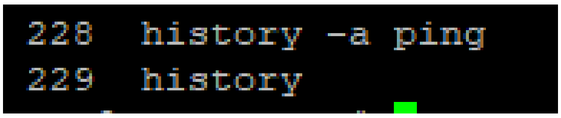
-c clears the complete history list.

-d offset deletes the history entry at the OFFSET position.



```
[root@sandbox ~]# history -d 11
[root@sandbox ~]# history
    1  ls
    2  pwd
    3  sudo
    4  clear
    5  pwd
    6  cd root
    7  ls
    8  clea
    9  clear
   10  pwd
   11  clear
   12  pwd
   13  pwd -L
   14  pwd -P
```

-a appends history lines.



```
228  history -a ping
229  history
```

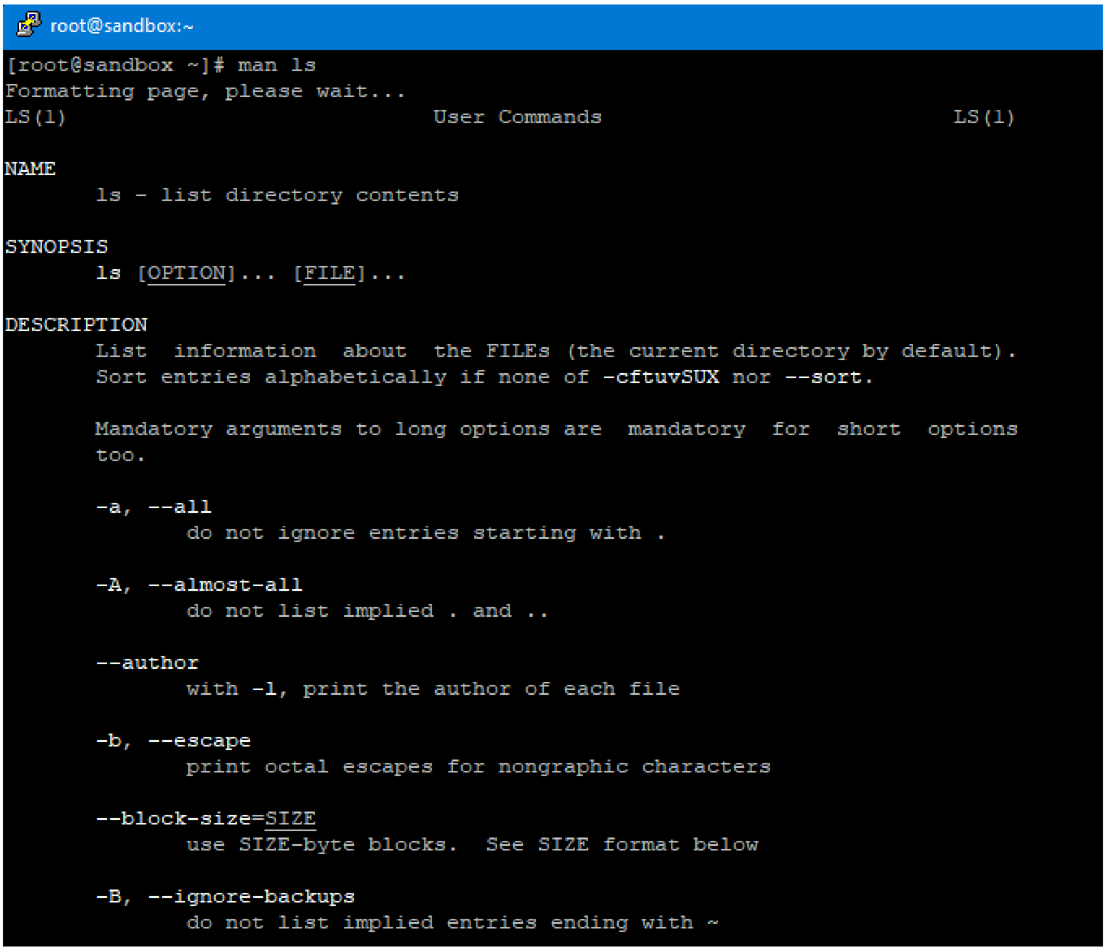
14. man command:

The man command provides a user manual of any commands or utilities you can run in Terminal, including the name, description, and options.

To display the complete manual, enter:

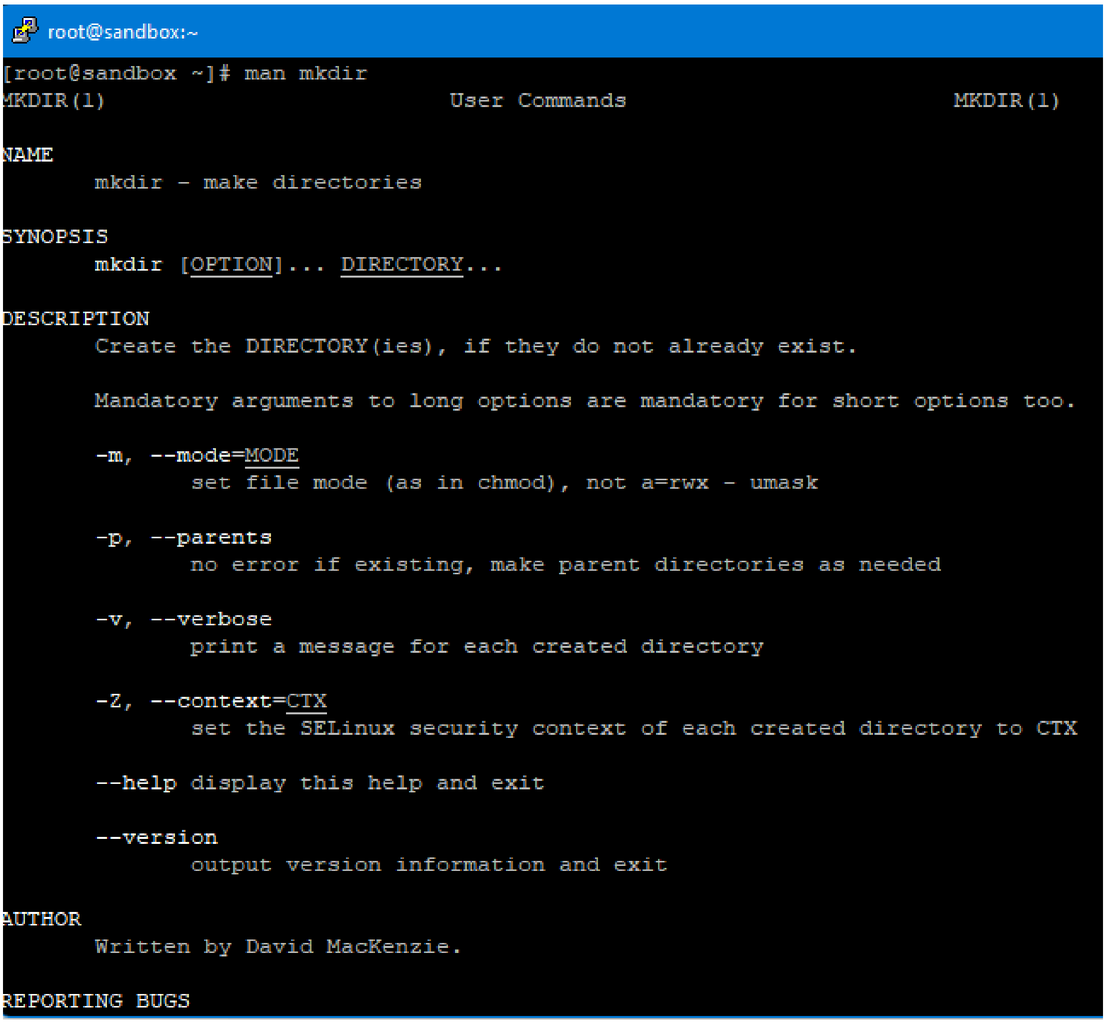
man [command_name]

Example: man ls, it will generate the complete manual of ls command.



```
[root@sandbox ~]# man ls
Formatting page, please wait...
LS(1)                            User Commands                           LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print octal escapes for nongraphic characters

       --block-size=SIZE
              use SIZE-byte blocks.  See SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~
```

For mkdir command:



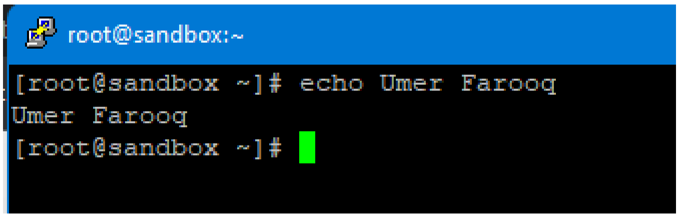
```
[root@sandbox ~]# man mkdir
MKDIR(1)                         User Commands                        MKDIR(1)

NAME
       mkdir - make directories

SYNOPSIS
       mkdir [OPTION]... DIRECTORY...

DESCRIPTION
       Create the DIRECTORY(ies), if they do not already exist.

       Mandatory arguments to long options are mandatory for short options too.

       -m, --mode=MODE
              set file mode (as in chmod), not a=rwx - umask

       -p, --parents
              no error if existing, make parent directories as needed

       -v, --verbose
              print a message for each created directory

       -Z, --context=CTX
              set the SELinux security context of each created directory to CTX

       --help display this help and exit

       --version
              output version information and exit

AUTHOR
       Written by David MacKenzie.

REPORTING BUGS
```

15. echo command:

The echo command is a **built-in utility that displays a line of text** or string using the standard output.

Here’s the basic syntax:

echo [option] [string]



```
[root@sandbox ~]# echo Umer Farooq
Umer Farooq
[root@sandbox ~]#
```

This command supports many options, such as:

-n displays the output without the trailing newline.

-e enables the interpretation of the following backslash escapes:

\a plays sound alert.

\b removes spaces in between a text.

\c produces no further output.

-E displays the default option and disables the interpretation of backslash escapes.

Topic Title: Read About Following Terms

1. Data Node
2. Journal Node
3. Edge Node
4. HA Name Node
5. Secondary Name Node

1. Data Node:

DataNode is also known as **Slave node**. In Hadoop HDFS Architecture, DataNode stores actual data in HDFS. DataNodes responsible for serving, reading and writing requests for the clients.

DataNodes can **deploy on commodity hardware**. DataNodes sends information to the NameNode about the files and blocks stored in that node and responds to the NameNode for all filesystem operations. When a DataNode starts up it announces itself to the NameNode along with the list of blocks it is responsible for. DataNode is usually configured with a lot of hard disk space. Because the actual data is stored in the DataNode.

Functions of DataNode in HDFS:

- These are slave daemons or process which runs on each slave machine.
- The actual data is stored on DataNodes.
- The DataNodes perform the low-level read and write requests from the file system's clients.
- Every **DataNode sends a heartbeat message** to the Name Node every 3 seconds and conveys that it is alive. In the scenario when a NameNode does not receive a heartbeat from a DataNode for

10 minutes, the Name Node considers that particular Data Node as dead and starts the process of Block replication on some other Data Node.

- All **Data Nodes are synchronised in the Hadoop cluster** in a way that they can communicate with one another and make sure of:
  - Balancing the data in the system,
  - Move data for keeping high replication,
  - Copy Data when required.

2. Journal Node:

JournalNode is a daemon that **enables high availability** of namenode.

In a typical HA cluster, two separate machines are configured as NameNodes. At any point in time, exactly one of the NameNodes is in an Active state, and the other is in a Standby state.

The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to **provide a fast failover** if necessary.

In order for the Standby node to keep its state synchronized with the Active node, both nodes communicate with a group of separate daemons called JournalNodes (JNs).

3. Edge Node:

An **edge node** is a computer that **acts as an end user portal for communication with other nodes** in cluster computing. Edge nodes are also sometimes called gateway nodes or edge communication nodes.

In a Hadoop cluster, three types of nodes exist: master, worker and edge nodes. The distinction of roles helps maintain efficiency.

Master nodes control which nodes perform which tasks and what processes run on what nodes. The majority of work is assigned to worker nodes. Worker nodes store most of the data and perform most of the calculations Edge nodes facilitate communications from end users to master and worker nodes.

Some nodes have important tasks, which may impact performance if interrupted. Edge nodes allow end users to contact worker nodes when necessary, **providing a network interface for the cluster** without leaving the entire cluster open to communication. That limitation improves reliability and security. As work is evenly distributed between work nodes, the edge node's role helps avoid data skewing and performance issues.

Purpose:

I. Edge nodes act as a network interface for the cluster and outside world **(you don't want to leave the entire cluster open to the outside world when you can make do with a few nodes instead)**. This also helps keep the network architecture costs low.

II. Uniform data/work distribution. If users directly connect to the same set of few worker nodes won't harness the entire cluster's resources resulting in data skew/performance issues.

III. **Edge nodes serve as staging space for final data** (stuff like data ingestion using Sqoop, Oozie workflow setup etc).

4. HA Name Node:

The HDFS NameNode High Availability feature enables you to run redundant NameNodes in the same cluster in an Active/Passive configuration with a hot standby. This eliminates the NameNode as a potential single point of failure **(SPOF)** in an HDFS cluster.

Formerly, if a cluster had a single NameNode, and that machine or process became unavailable, the entire cluster would be unavailable until the NameNode was either restarted or started on a separate machine. This situation impacted the total availability of the HDFS cluster in two major ways:

In the case of an unplanned event such as a machine crash, the cluster would be unavailable until an operator restarted the NameNode.

Planned maintenance events such as software or hardware upgrades on the NameNode machine would result in periods of cluster downtime.

HDFS NameNode HA avoids this by facilitating either a fast failover to the new NameNode during machine crash, or a graceful administrator-initiated failover during planned maintenance.

5. Secondary Name Node:

Apart from NameNode & DataNode daemons, there is a third daemon or a process called **Secondary NameNode**.

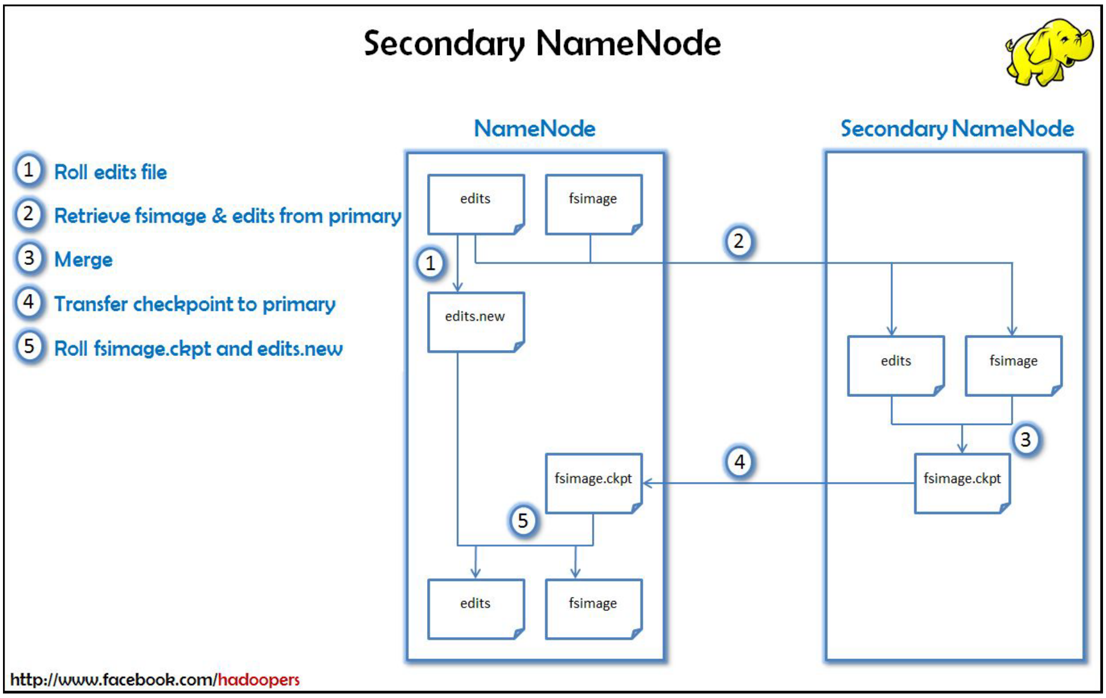
The Secondary NameNode **works concurrently** with the primary NameNode **as a helper daemon**.

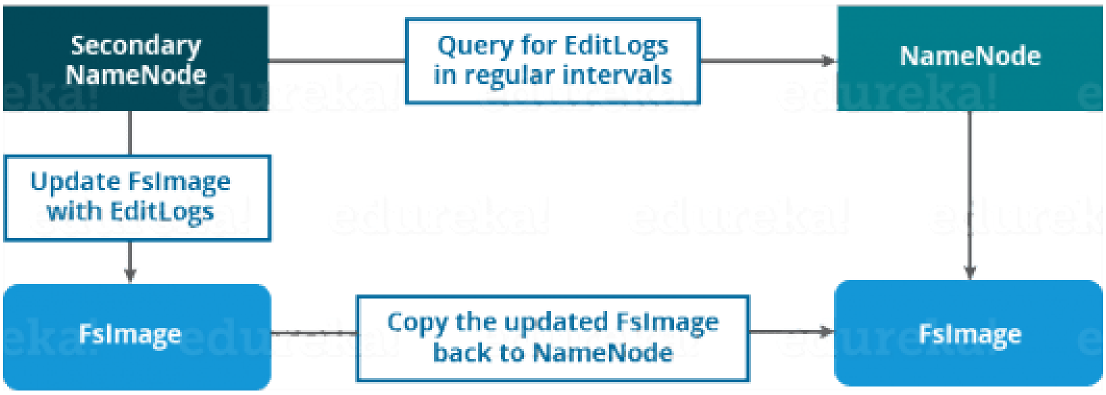
And don’t be confused about the Secondary NameNode being a backup NameNode because it is not. **It is not a hot-standby for NameNode.**

Functions of Secondary NameNode:

- The Secondary NameNode is one which **constantly reads all the file systems and metadata from the RAM** of the NameNode and **writes it into the hard disk** or the file system.
- It is responsible for combining the EditLogs with FsImage from the NameNode.
- It downloads the EditLogs from the NameNode at regular intervals and applies them to FsImage. The new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time.

Hence, Secondary NameNode performs regular checkpoints in HDFS. Therefore, it is also called CheckpointNode.

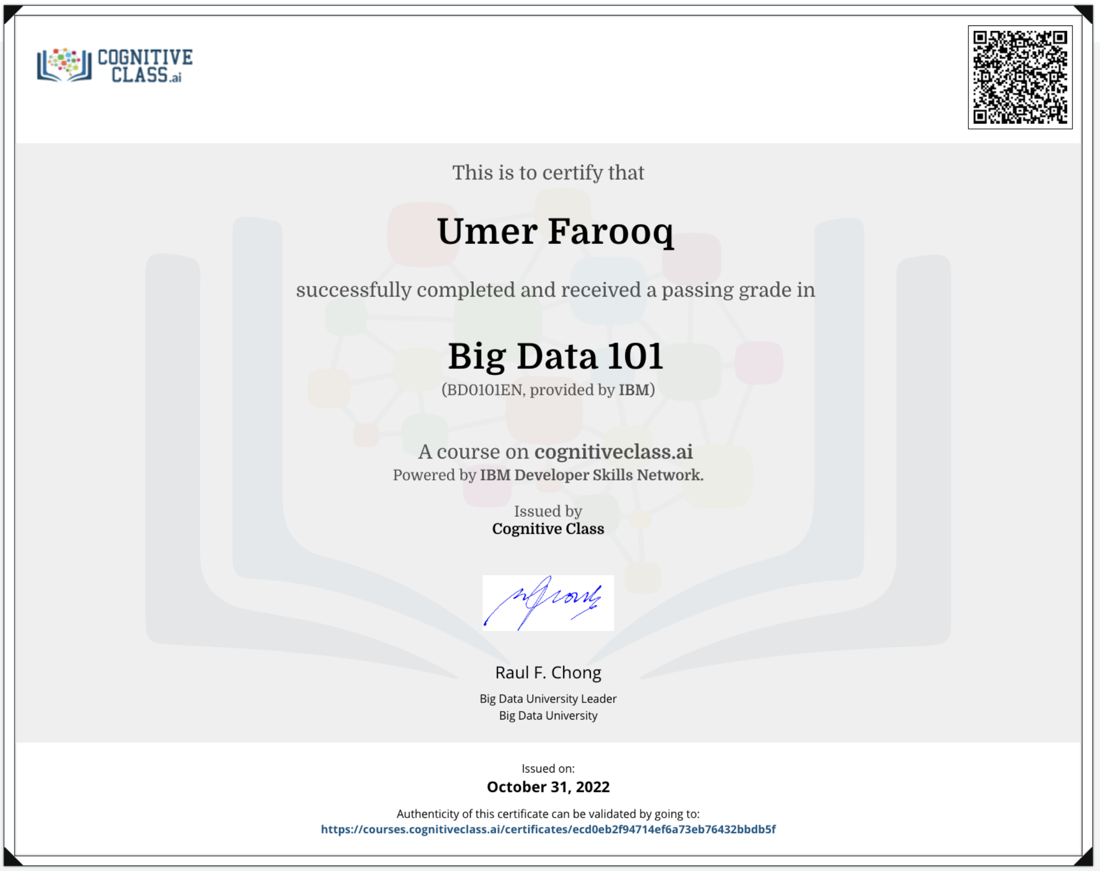



Secondary NameNode
Query for EditLogs in regular intervals
NameNode
Update FsImage with EditLogs
FsImage
Copy the updated FsImage back to NameNode
FsImage

Certificate

Certificate Title: Big Data 101

Link: Click Here



COGNITIVE CLASS.ai

This is to certify that

Umer Farooq

successfully completed and received a passing grade in

Big Data 101

(BD0101EN, provided by **IBM**)

A course on **cognitiveclass.ai**
Powered by **IBM Developer Skills Network.**

Issued by
Cognitive Class

Raul F. Chong

Big Data University Leader
Big Data University

Issued on:
October 31, 2022

Authenticity of this certificate can be validated by going to:
https://courses.cognitiveclass.ai/certificates/ecd0eb2f94714ef6a73eb76432bbdb5f

References:

Edge Node