

Exp.1 Downloading and installing Hadoop, Understanding different Hadoop modes, Startup scripts, Configuration files.

AIM:

To Download and install Hadoop, Understanding different Hadoop modes, Startup scripts, Configuration files.

Procedure:

Step 1 : Install Java Development Kit

The default Ubuntu repositories contain Java 8 and Java 11 both. But, Install Java 8 because it only works on this version. Use the following command to install it.

```
$sudo apt update&&sudo apt install openjdk-8-jdk
```

Step 2 : Verify the Java version

Once installed, verify the installed version of Java with the following command:

```
$ java -version
```

Output:

```
sanjay@sanjay-VirtualBox:~$ java -version
openjdk version "1.8.0_382"
OpenJDK Runtime Environment (build 1.8.0_382-8u382-ga-1~23.04.1-b05)
OpenJDK 64-Bit Server VM (build 25.382-b05, mixed mode)
```

Step 3: Install SSH

SSH (Secure Shell) installation is vital for Hadoop as it enables secure communication between nodes in the Hadoop cluster. This ensures data integrity, confidentiality, and allows for efficient distributed processing of data across the cluster.

```
$sudo apt install ssh
```

Step 4 : Create the hadoop user :

All the Hadoop components will run as the user that you create for Apache Hadoop, and the user will also be used for logging in to Hadoop's web interface.

Run the command to create user and set password:

```
$ sudo adduser hadoop
```

Output:

```

sanjay@sanjay-VirtualBox:~$ sudo adduser hadoop
Adding user 'hadoop' ...
Adding new group 'hadoop' (1001) ...
Adding new user 'hadoop' (1001) with group 'hadoop (1001)' ...
adduser: The home directory '/home/hadoop' already exists. Not copying from '/etc/skel'.
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
Changing the user information for hadoop
Enter the new value, or press ENTER for the default
  Full Name []:
  Room Number []:
  Work Phone []:
  Home Phone []:
  Other []:
Is the information correct? [Y/n] y
Adding new user 'hadoop' to supplemental / extra groups 'users' ...
Adding user 'hadoop' to group 'users' ...
sanjay@sanjay-VirtualBox:~$ su - hadoop
Password:
hadoop@sanjay-VirtualBox:~$

```

Step 5 : Switch user

Switch to the newly created hadoop user:

\$ su - hadoop

Step 6 : Configure SSH

Now configure password-less SSH access for the newly created hadoop user, so didn't enter the key to save file and passphrase. Generate an SSH keypair (generate Public and Private Key Pairs)first

\$ssh-keygen -t rsa

```

hadoop@sanjay-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):
Created directory '/home/hadoop/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/hadoop/.ssh/id_rsa
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:yu8Hsie3mbQ7UifnFH6iam4kFLRRbEb9zVYGutbaYyg hadoop@sanjay-VirtualBox
The key's randomart image is:
+---[RSA 3072]-----+
|      .o+o.  ..+o |
|      .o+ .  o  + |
|      .+   . o.= |
|      .   E..=o. |
|      . S   ... . |
|      .o.= o     |
|      o*.B       |
|      +o*++ o .  |
|      X0*+.o     |
+---[SHA256]-----+
hadoop@sanjay-VirtualBox:~$

```

Step 7 : Set permissions :

Next, append the generated public keys from id_rsa.pub to authorized_keys and set proper permission:

\$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

\$ chmod 640 ~/.ssh/authorized_keys

Step 8 : SSH to the localhost

Next, verify the password less SSH authentication with the following command:

\$ ssh localhost

You will be asked to authenticate hosts by adding RSA keys to known hosts. Type yes and hit Enter to authenticate the localhost:

```
hadoop@sanjay-VirtualBox: ~/hadoop$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:2ZsL3K5BKG6h8lsZpTufDvB69zWFKS7lFjvsnhW53I.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
Welcome to Ubuntu 23.04 (GNU/Linux 6.2.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

122 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

hadoop@sanjay-VirtualBox: ~$
```

Step 9 : Switch user

Again switch to hadoop. So, First, change the user to hadoop with the following command:

\$ su-hadoop

Step 10 : Install hadoop

Next, download the latest version of Hadoop using the wget command:

\$ wget <https://downloads.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz>

Once downloaded, extract the downloaded file:

\$ tar -xvzf hadoop-3.3.6.tar.gz

Next, rename the extracted directory to hadoop:

\$ mv hadoop-3.3.6 hadoop

```
hadoop@sanjay-VirtualBox: ~$ mv hadoop-3.3.6 hadoop
hadoop@sanjay-VirtualBox: ~$ ls
hadoop  hadoop-3.3.6.tar.gz
hadoop@sanjay-VirtualBox: ~$
```

Next, you will need to configure Hadoop and Java Environment Variables on your system. Open the ~/.bashrc file in your favorite text editor. Use nano editor , to pasting the code we use ctrl+shift+v for saving the file ctrl+x and ctrl+y ,then hit enter:

Next, you will need to configure Hadoop and Java Environment Variables on your system.

Open the ~/.bashrc file in your favorite text editor:

\$ nano ~/.bashrc

Append the below lines to file.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_HOME=/home/hadoop/hadoop
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Save and close the file. Then, activate the environment variables with the following command:

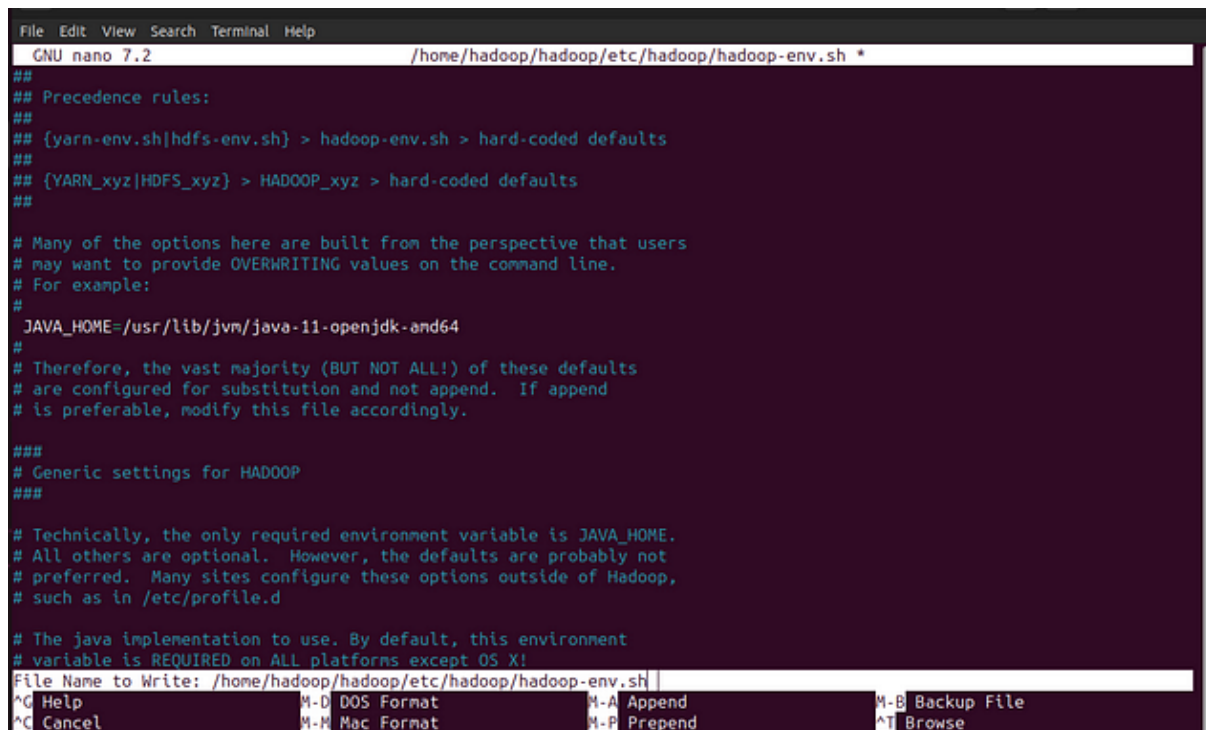
```
s$ source ~/.bashrc
```

Next, open the Hadoop environment variable file:

```
$ nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

Search for the “export JAVA_HOME” and configure it.

```
JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```



```
File Edit View Search Terminal Help
GNU nano 7.2 /home/hadoop/hadoop/etc/hadoop/hadoop-env.sh *
##
## Precedence rules:
##
## {yarn-env.sh|hdfs-env.sh} > hadoop-env.sh > hard-coded defaults
## {YARN_xyz|HDFS_xyz} > HADOOP_xyz > hard-coded defaults
##
# Many of the options here are built from the perspective that users
# may want to provide OVERWRITING values on the command line.
# For example:
#
# JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
#
# Therefore, the vast majority (BUT NOT ALL!) of these defaults
# are configured for substitution and not append. If append
# is preferable, modify this file accordingly.
###
# Generic settings for HADOOP
###
# Technically, the only required environment variable is JAVA_HOME.
# All others are optional. However, the defaults are probably not
# preferred. Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d
#
# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
File Name to Write: /home/hadoop/hadoop/etc/hadoop/hadoop-env.sh
^G Help      M-D DOS Format  M-A Append     M-B Backup File
^C Cancel    M-M Mac Format  M-P Prepend    ^T Browse
```

Save and close the file when you are finished.

Step 11 : Configuring Hadoop :

First, you will need to create the namenode and datanode directories inside the Hadoop user home directory. Run the following command to create both directories:

```
$ cd hadoop/
```

```
$ mkdir -p ~/hadoopdata/hdfs/{namenode,datanode}
```

- Next, edit the core-site.xml file and

```
hadoop@sanjay-VirtualBox:~$ nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
hadoop@sanjay-VirtualBox:~$ cd hadoop/
hadoop@sanjay-VirtualBox:~/hadoop$ mkdir -p ~/hadoopdata/hdfs/{namenode,datanode}
hadoop@sanjay-VirtualBox:~/hadoop$
```

update with your system hostname:

```
$ nano $HADOOP_HOME/etc/hadoop/core-site.xml
```

Change the following name as per your system hostname:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Save and close the file.

Then, edit the hdfs-site.xml file:

```
$ nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

- Change the NameNode and DataNode directory paths as shown below:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>

  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
  </property>

  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
  </property>
</configuration>
```

- Then, edit the mapred-site.xml file:

```
$ nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

- Make the following changes:

```
<configuration>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME/home/hadoop/hadoop/bin/hadoop</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME/home/hadoop/hadoop/bin/hadoop</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=$HADOOP_HOME/home/hadoop/hadoop/bin/hadoop</value>
  </property>
</configuration>
```

- Then, edit the yarn-site.xml file:
\$nano \$HADOOP_HOME/etc/hadoop/yarn-site.xml
- Make the following changes:

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

Save the file and close it .

Step 12 – Start Hadoop Cluster

Before starting the Hadoop cluster. You will need to format the Namenode as a hadoop user.

Run the following command to format the Hadoop Namenode:

```
$hdfs namenode -format
```

Once the namenode directory is successfully formatted with hdfs file system, you will see the message “Storage directory /home/hadoop/hadoopdata/hdfs/namenode has been successfully formatted “

Then start the Hadoop cluster with the following command.

```
$ start-all.sh
```



```

hadoop@sanjay-VirtualBox:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: This is not a recommended production deployment configuration.
WARNING: Use CTRL-C to abort.
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [sanjay-VirtualBox]
Starting resourcemanager
Starting nodemanagers
hadoop@sanjay-VirtualBox:~$

```

You can now check the status of all Hadoop services using the jps command:

\$ jps

```

hadoop@sanjay-VirtualBox:~/hadoop$ jps
7235 NodeManager
6677 DataNode
7593 Jps
6554 NameNode
7116 ResourceManager
6893 SecondaryNameNode
hadoop@sanjay-VirtualBox:~/hadoop$

```

Step 13 – Access Hadoop Namenode and Resource Manager

- First we need to know our ipaddress, In Ubuntu we need to install net-tools to run ifconfig command,
If you installing net-tools for the first time switch to default user:

\$sudo apt install net-tools

- Then run ifconfig command to know our ip address:

ifconfig

```

hadoop@sanjay-VirtualBox:~/hadoop$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.6 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 2401:4900:1c28:46c4:f76c:b206:abe3:2d45 prefixlen 64 scopeid 0x0<global>
    inet6 2401:4900:1c28:46c4:ed13:53f4:5c05:50c6 prefixlen 64 scopeid 0x0<global>
    inet6 fe80::112b:300a:9242:51f3 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:83:31:35 txqueuelen 1000 (Ethernet)
    RX packets 645228 bytes 934388358 (934.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 93618 bytes 8998032 (8.9 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 3331 bytes 491873 (491.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3331 bytes 491873 (491.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

hadoop@sanjay-VirtualBox:~/hadoop$

```

Here my ip address is 192.168.1.6.

- To access the Namenode, open your web browser and visit the URL <http://your-server-ip:9870>.
- You should see the following screen:
<http://192.168.1.6:9870>

← → ↻

192.168.1.6:9870/dfshealth.html#tab-overview

☆

🔍

☰

Hadoop

Overview

Datanodes

Datanode Volume Failures

Snapshot

Startup Progress

Utilities

Overview 'localhost:9000' (✓active)

Started:	Sun Sep 10 13:08:22 +0530 2023
Version:	3.3.6, r1be78238728da9266a4f88195058f08fd012bf9c
Compiled:	Sun Jun 18 13:52:00 +0530 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
Cluster ID:	CID-dc5a1253-b0cd-4686-a807-fd0dd4c5a9a
Block Pool ID:	BP-1272319295-127.0.1.1-1694331447796

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).

Heap Memory used 69.85 MB of 107 MB Heap Memory. Max Heap Memory is 748 MB.

Non Heap Memory used 51.89 MB of 55.44 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	24.44 GB
----------------------	----------

To access Resource Manage, open your web browser and visit the URL <http://your-server-ip:8088>. You should see the following screen:

<http://192.168.16:8088>


← → ↻

192.168.1.6:8088/cluster

☆

🔍

☰



Cluster

About

Nodes

Node Labels

Applications

NEW

NEW SAVING

SUBMITTED

ACCEPTED

RUNNING

FINISHED

FAILED

KILLED

Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running
0	0	0	0	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decom
1	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Min
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCore:

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	Finis
Showing 0 to 0 of 0 entries									

Step 14 – Verify the Hadoop Cluster

At this point, the Hadoop cluster is installed and configured. Next, we will create some directories in the HDFS filesystem to test the Hadoop.

Let's create some directories in the HDFS filesystem using the following command:

```
$ hdfsdfs -mkdir /test1
$ hdfsdfs -mkdir /logs
```


Next, run the following command to list the above directory:

```
$ hdfs dfs -ls /
```

You should get the following output:

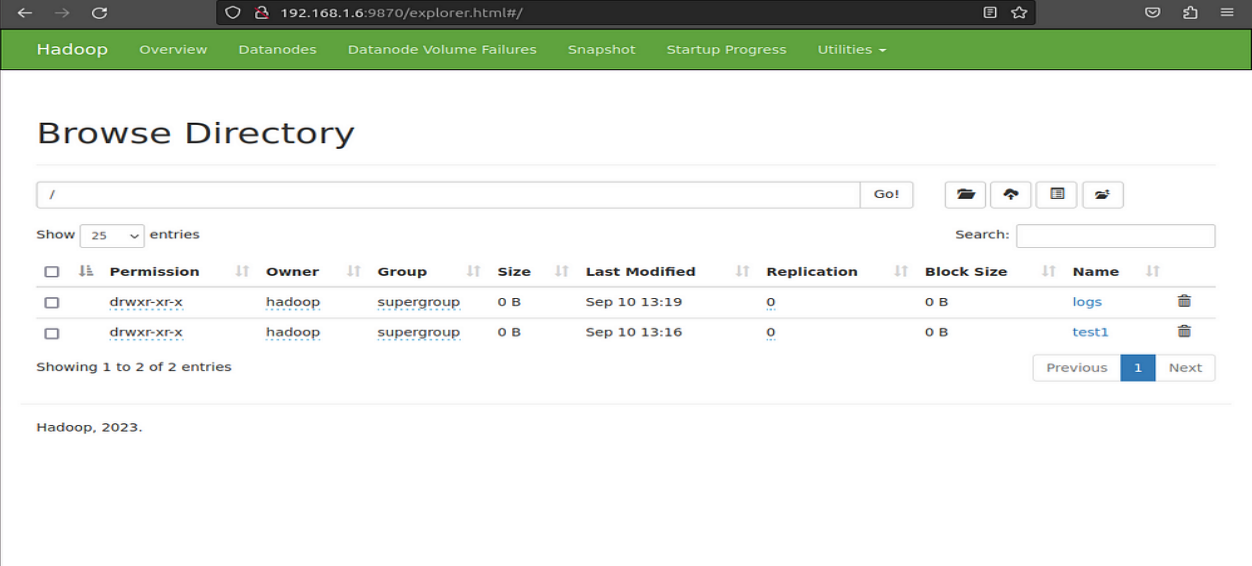
```
hadoop@sanjay-VirtualBox:~/hadoop$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x - hadoop supergroup 0 2023-09-10 13:16 /logs
drwxr-xr-x - hadoop supergroup 0 2023-09-10 13:16 /test1
hadoop@sanjay-VirtualBox:~/hadoop$
```

Also, put some files to hadoop file system. For the example, putting log files from host machine to hadoop file system.

```
$ hdfs dfs -put /var/log/* /logs/
```

You can also verify the above files and directory in the Hadoop Namenode web interface.

Go to the web interface, click on the Utilities => Browse the file system. You should see your directories which you have created earlier in the following screen:



The screenshot shows the Hadoop Namenode web interface. The top navigation bar includes links for Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The main content area is titled 'Browse Directory' and shows a file browser view of the root directory. A table lists the contents of the root directory, including the 'logs' and 'test1' directories. The 'logs' directory is highlighted in blue, indicating it is the current selection. The table columns are: Permission, Owner, Group, Size, Last Modified, Replication, Block Size, and Name. The 'logs' directory has a permission of 'drwxr-xr-x', owner 'hadoop', group 'supergroup', size '0 B', last modified 'Sep 10 13:19', replication '0', and block size '0 B'. The 'test1' directory has a permission of 'drwxr-xr-x', owner 'hadoop', group 'supergroup', size '0 B', last modified 'Sep 10 13:16', replication '0', and block size '0 B'. The page also shows a search bar and a pagination control at the bottom.

Step 15 – Stop Hadoop Cluster

To stop the Hadoop all services, run the following command:

```
$ stop-all.sh
```

```
hadoop@sanjay-VirtualBox:~/hadoop$ stop-all.sh
WARNING: Stopping all Apache Hadoop daemons as hadoop in 10 seconds.
WARNING: Use CTRL-C to abort.
Stopping namenodes on [localhost]
Stopping datanodes
Stopping secondary namenodes [sanjay-VirtualBox]
Stopping nodemanagers
Stopping resourcemanager
hadoop@sanjay-VirtualBox:~/hadoop$
```

Result:

The step-by-step installation and configuration of Hadoop on Ubuntu linux system have been successfully completed.

EXP 2: Run a basic Word Count Map Reduce program to understand Map Reduce Paradigm.

AIM:

To run a basic Word Count MapReduce program.

Procedure:

Step 1: Create Data File:

Create a file named "word_count_data.txt" and populate it with text data that you wish to analyse.

Login with your hadoop user.

```
nano word_count.txt
```

Output: Type the below content in word_count.txt

```
GNU nano 7.2 word_count.txt
Made it to LA yeah
Finally in LA yeah
Lookin for the weed though
Tryna make my own dough
Callin for Maria
Lost without Maria
Might dive in the marina

[ Read 7 lines ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line
```

Step 2: Mapper Logic - mapper.py:

Create a file named "mapper.py" to implement the logic for the mapper. The mapper will read input data from STDIN, split lines into words, and output each word with its count.

```
nano mapper.py
```

```
# Copy and paste the mapper.py code
```

```
#!/usr/bin/env python3
# import sys because we need to read and write data to STDIN and STDOUT
#!/usr/bin/python3
import sys
for line in sys.stdin:
    line = line.strip() # remove leading and trailing whitespace
    words = line.split() # split the line into words
    for word in words:
```

```
print( '%s\t%s' % (word, 1))
```

Step 3: Reducer Logic - reducer.py:

Create a file named "reducer.py" to implement the logic for the reducer. The reducer will aggregate the occurrences of each word and generate the final output.

```
nano reducer.py
# Copy and paste the reducer.py code
```

reducer.py

```
#!/usr/bin/python3
from operator import itemgetter
import sys
current_word = None
current_count = 0
word = None
for line in sys.stdin:
    line = line.strip()
    word, count = line.split('\t', 1)
    try:
        count = int(count)
    except ValueError:
        continue
    if current_word == word:
        current_count += count
    else:
        if current_word:
            print( '%s\t%s' % (current_word, current_count))
            current_count = count
            current_word = word
        if current_word == word:
            print( '%s\t%s' % (current_word, current_count))
```

Step 4: Prepare Hadoop Environment:

Start the Hadoop daemons and create a directory in HDFS to store your data.

```
start-all.sh
hdfsdfs -mkdir /word_count_in_python
hdfsdfs -copyFromLocal /path/to/word_count.txt/word_count_in_python
```

Step 6: Make Python Files Executable:

Give executable permissions to your mapper.py and reducer.py files.

```
chmod 777 mapper.py reducer.py
```

Step 7: Run Word Count using Hadoop Streaming:

Download the latest hadoop-streaming jar file and place it in a location you can easily access.

Then run the Word Count program using Hadoop Streaming.

```
hadoop jar /path/to/hadoop-streaming-3.3.6.jar \  
-input /word_count_in_python/word_count_data.txt \  
-output /word_count_in_python/new_output \  
-mapper /path/to/mapper.py \  
-reducer /path/to/reducer.py
```

```
hadoop@sanjay-VirtualBox:~/wordcount$ hadoop jar /home/hadoop/Documents/hadoop-streaming-3.3.6.jar -input  
t /word_count_in_python/word_count.txt -output /word_count_in_python/output -mapper mapper.py -r  
educer reducer.py  
2023-10-25 22:54:51,391 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties  
2023-10-25 22:54:51,526 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).  
2023-10-25 22:54:51,526 INFO impl.MetricsSystemImpl: JobTracker metrics system started  
2023-10-25 22:54:51,537 WARN impl.MetricsSystemImpl: JobTracker metrics system already initialized!  
2023-10-25 22:54:51,785 INFO mapred.FileInputFormat: Total input files to process : 1  
2023-10-25 22:54:51,912 INFO mapreduce.JobSubmitter: number of splits:1  
2023-10-25 22:54:52,141 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local1007878220_0001  
2023-10-25 22:54:52,141 INFO mapreduce.JobSubmitter: Executing with tokens: []  
2023-10-25 22:54:52,320 INFO mapreduce.Job: The url to track the job: http://localhost:8080/  
2023-10-25 22:54:52,322 INFO mapreduce.Job: Running job: job_local1007878220_0001  
2023-10-25 22:54:52,327 INFO mapred.LocalJobRunner: OutputCommitter set in config null  
2023-10-25 22:54:52,328 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapred.FileOutputCo  
mmitter  
2023-10-25 22:54:52,332 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2  
2023-10-25 22:54:52,333 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders  
under output directory:false, ignore cleanup failures: false  
2023-10-25 22:54:52,409 INFO mapred.LocalJobRunner: Waiting for map tasks  
2023-10-25 22:54:52,412 INFO mapred.LocalJobRunner: Starting task: attempt_local1007878220_0001_m_000000_0  
2023-10-25 22:54:52,455 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2  
2023-10-25 22:54:52,455 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders  
under output directory:false, ignore cleanup failures: false  
2023-10-25 22:54:52,493 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]  
2023-10-25 22:54:52,517 INFO mapred.MapTask: Processing split: hdfs://localhost:9000/word_count_in_python/wo  
rd_count.txt:0+150  
2023-10-25 22:54:52,570 INFO mapred.MapTask: numReduceTasks: 1  
2023-10-25 22:54:52,647 INFO mapred.MapTask: (SPLITTER) 0: hdfs://localhost:9000/word_count_in_python/wo
```

Step 8: Check Output:

Check the output of the Word Count program in the specified HDFS output directory.

```
hdfs dfs -cat /word_count_in_python/new_output/part-00000
```

```
hadoop@sanjay-VirtualBox:~/wordcount$ hdfs dfs -cat /word_count_in_python/output/part-000000
Callin 1
Finally 1
LA 2
Lookin 1
Lost 1
Made 1
Maria 2
Might 1
Tryna 1
dive 1
dough 1
for 2
in 2
it 1
make 1
marina 1
my 1
own 1
the 2
though 1
to 1
weed 1
without 1
yeah 2
hadoop@sanjay-VirtualBox:~/wordcount$
```

Result:

Thus, the program for basic Word Count Map Reduce has been executed successfully.

EXP 3: Map Reduce program to process a weather dataset.

AIM:

To implement MapReduce program to process a weather dataset.

Procedure:

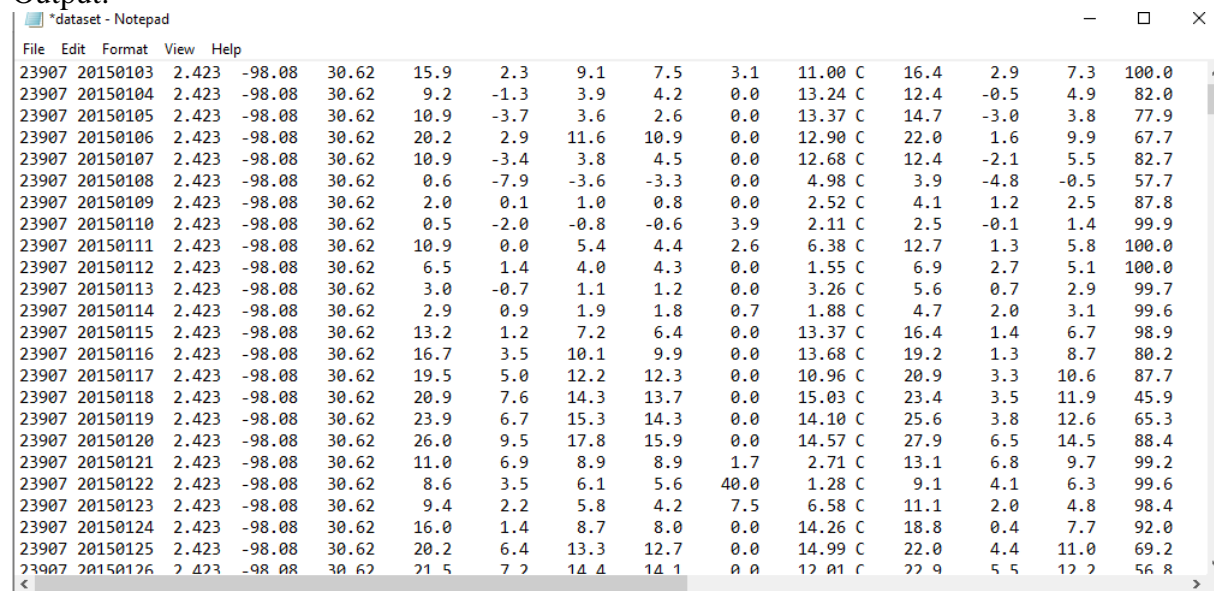
Step 1: Create Data File:

Create a file named "word_count_data.txt" and populate it with text data that you wish to analyse.

Login with your hadoop user.

Download the dataset (weather data)

Output:



23907	20150103	2.423	-98.08	30.62	15.9	2.3	9.1	7.5	3.1	11.00	C	16.4	2.9	7.3	100.0
23907	20150104	2.423	-98.08	30.62	9.2	-1.3	3.9	4.2	0.0	13.24	C	12.4	-0.5	4.9	82.0
23907	20150105	2.423	-98.08	30.62	10.9	-3.7	3.6	2.6	0.0	13.37	C	14.7	-3.0	3.8	77.9
23907	20150106	2.423	-98.08	30.62	20.2	2.9	11.6	10.9	0.0	12.90	C	22.0	1.6	9.9	67.7
23907	20150107	2.423	-98.08	30.62	10.9	-3.4	3.8	4.5	0.0	12.68	C	12.4	-2.1	5.5	82.7
23907	20150108	2.423	-98.08	30.62	0.6	-7.9	-3.6	-3.3	0.0	4.98	C	3.9	-4.8	-0.5	57.7
23907	20150109	2.423	-98.08	30.62	2.0	0.1	1.0	0.8	0.0	2.52	C	4.1	1.2	2.5	87.8
23907	20150110	2.423	-98.08	30.62	0.5	-2.0	-0.8	-0.6	3.9	2.11	C	2.5	-0.1	1.4	99.9
23907	20150111	2.423	-98.08	30.62	10.9	0.0	5.4	4.4	2.6	6.38	C	12.7	1.3	5.8	100.0
23907	20150112	2.423	-98.08	30.62	6.5	1.4	4.0	4.3	0.0	1.55	C	6.9	2.7	5.1	100.0
23907	20150113	2.423	-98.08	30.62	3.0	-0.7	1.1	1.2	0.0	3.26	C	5.6	0.7	2.9	99.7
23907	20150114	2.423	-98.08	30.62	2.9	0.9	1.9	1.8	0.7	1.88	C	4.7	2.0	3.1	99.6
23907	20150115	2.423	-98.08	30.62	13.2	1.2	7.2	6.4	0.0	13.37	C	16.4	1.4	6.7	98.9
23907	20150116	2.423	-98.08	30.62	16.7	3.5	10.1	9.9	0.0	13.68	C	19.2	1.3	8.7	80.2
23907	20150117	2.423	-98.08	30.62	19.5	5.0	12.2	12.3	0.0	10.96	C	20.9	3.3	10.6	87.7
23907	20150118	2.423	-98.08	30.62	20.9	7.6	14.3	13.7	0.0	15.03	C	23.4	3.5	11.9	45.9
23907	20150119	2.423	-98.08	30.62	23.9	6.7	15.3	14.3	0.0	14.10	C	25.6	3.8	12.6	65.3
23907	20150120	2.423	-98.08	30.62	26.0	9.5	17.8	15.9	0.0	14.57	C	27.9	6.5	14.5	88.4
23907	20150121	2.423	-98.08	30.62	11.0	6.9	8.9	8.9	1.7	2.71	C	13.1	6.8	9.7	99.2
23907	20150122	2.423	-98.08	30.62	8.6	3.5	6.1	5.6	40.0	1.28	C	9.1	4.1	6.3	99.6
23907	20150123	2.423	-98.08	30.62	9.4	2.2	5.8	4.2	7.5	6.58	C	11.1	2.0	4.8	98.4
23907	20150124	2.423	-98.08	30.62	16.0	1.4	8.7	8.0	0.0	14.26	C	18.8	0.4	7.7	92.0
23907	20150125	2.423	-98.08	30.62	20.2	6.4	13.3	12.7	0.0	14.99	C	22.0	4.4	11.0	69.2
23907	20150126	2.423	-98.08	30.62	21.5	7.2	14.4	14.1	0.0	12.01	C	22.9	5.5	12.2	56.8

Step 2: Mapper Logic - mapper.py:

Create a file named "mapper.py" to implement the logic for the mapper. The mapper will read input data from STDIN, split lines into words, and output each word with its count.

```
nano mapper.py
```

```
# Copy and paste the mapper.py code
```

```
#!/usr/bin/env python
```

```
import sys
```

```
# input comes from STDIN (standard input)
```

```
# the mapper will get daily max temperature and group it by month. so output will be  
(month,dailymax_temperature)
```

```

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    #See the README hosted on the weather website which help us understand how each
position represents a column
    month = line[10:12]
    daily_max = line[38:45]
    daily_max = daily_max.strip()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be go through the shuffle process and then
        # be the input for the Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; month and daily max temperature as output
        print ('%s\t%s' % (month ,daily_max))
.

```

Step 3: Reducer Logic - reducer.py:

Create a file named "reducer.py" to implement the logic for the reducer. The reducer will aggregate the occurrences of each word and generate the final output.

```

nano reducer.py
# Copy and paste the reducer.py code

```

reducer.py

```

#!/usr/bin/env python

from operator import itemgetter
import sys

#reducer will get the input from stdin which will be a collection of key, value(Key=month ,
value= daily max temperature)
#reducer logic: will get all the daily max temperature for a month and find max temperature
for the month
#shuffle will ensure that key are sorted(month)
current_month = None
current_max = 0
month = None

# input comes from STDIN
for line in sys.stdin:

```

```

# remove leading and trailing whitespace
line = line.strip()
# parse the input we got from mapper.py
month, daily_max = line.split('\t', 1)

# convert daily_max (currently a string) to float
try:
    daily_max = float(daily_max)
except ValueError:
    # daily_max was not a number, so silently
    # ignore/discard this line
    continue

# this IF-switch only works because Hadoop shuffle process sorts map output
# by key (here: month) before it is passed to the reducer
if current_month == month:
    if daily_max > current_max:
        current_max = daily_max
else:
    if current_month:
        # write result to STDOUT
        print ('%s\t%s' % (current_month, current_max))
    current_max = daily_max
    current_month = month

# output of the last month
if current_month == month:
    print ('%s\t%s' % (current_month, current_max))

```

Step 4: Prepare Hadoop Environment:

Start the Hadoop daemons and create a directory in HDFS to store your data.

```
start-all.sh
```

Step 6: Make Python Files Executable:

Give executable permissions to your mapper.py and reducer.py files.

```
chmod 777 mapper.py reducer.py
```

Step 7: Run the program using Hadoop Streaming:

Download the latest hadoop-streaming jar file and place it in a location you can easily access.

Then run the program using Hadoop Streaming.

```
hadoop fs -mkdir -p /weatherdata
```

```
hadoop fs -copyFromLocal /home/sx/Downloads/dataset.txt /weatherdata
```

```
hdfs dfs -ls /weatherdata
```

```
hadoop jar /home/sx/hadoop-3.2.3/share/hadoop/tools/lib/hadoop-streaming-3.2.3.jar \  
-input /weatherdata/dataset.txt \  
-output /weatherdata/output \  
-file "/home/sx/Downloads/mapper.py" \  
-mapper "python3 mapper.py" \  
-file "/home/sx/Downloads/reducer.py" \  
-reducer "python3 reducer.py"
```

```
hdfs dfs -text /weatherdata/output/* > /home/sx/Downloads/outputfile.txt
```

Step 8: Check Output:

Check the output of the program in the specified HDFS output directory.

```
hdfs dfs -text /weatherdata/output/* > /home/sx/Downloads/output/  
/part-00000
```

```
01    26.5  
02    26.6  
03    29.1  
04    30.8  
05    31.1  
06    33.6  
07    38.5  
08    40.2  
09    36.5  
10    36.9  
11    27.6  
12    25.9
```

After copy and paste the above output in your local file give the below command to remove the directory from hdfs :

```
hadoop fs -rm -r /weatherdata/output
```

Result:

Thus, the program for weather dataset using Map Reduce has been executed successfully.

EXP 4: Create UDF in PIG

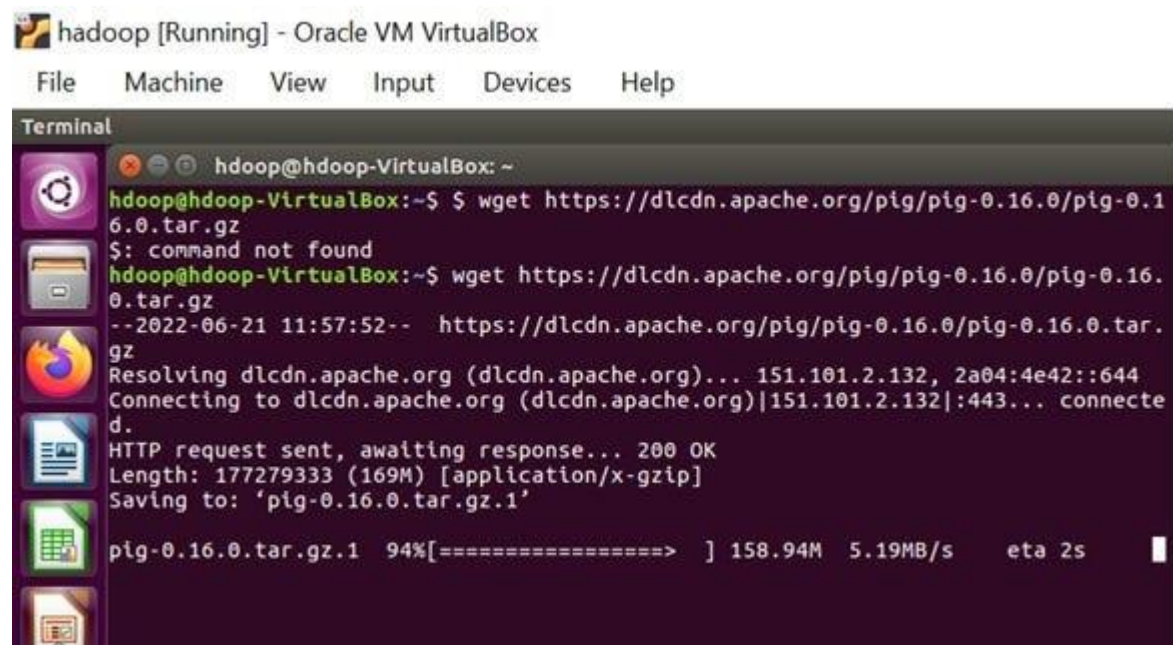
Step-by-step installation of Apache Pig on Hadoop cluster on Ubuntu

Pre-requisite:

- Ubuntu 16.04 or higher version running (I have installed Ubuntu on Oracle VM (Virtual Machine) VirtualBox),
- Run Hadoop on ubuntu (I have installed Hadoop 3.2.1 on Ubuntu 16.04). You may refer to my blog “How to install Hadoop installation” click [here](#) for Hadoop installation).

Pig installation steps

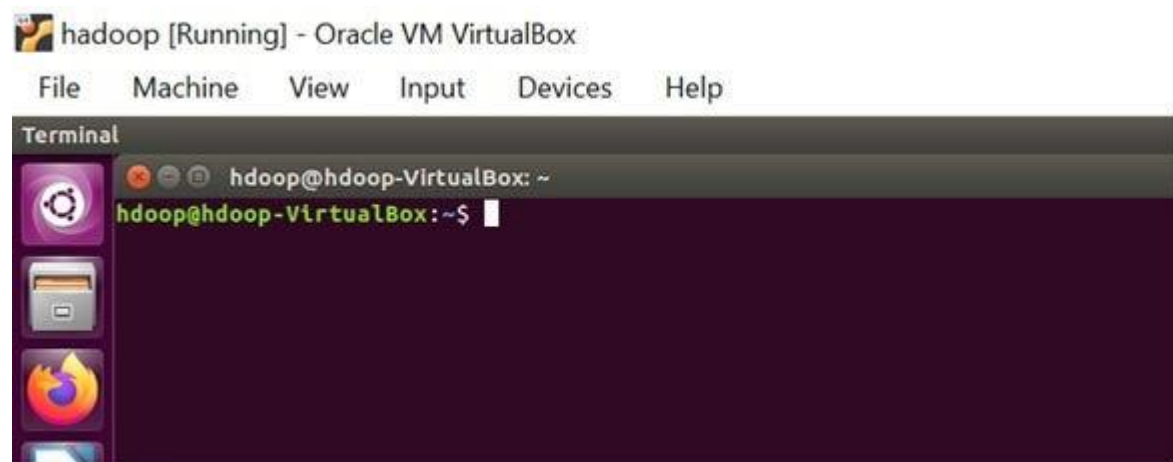
Step 1: Login into Ubuntu



```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ wget https://dclcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz  
$: command not found  
hadoop@hadoop-VirtualBox:~$ wget https://dclcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz  
--2022-06-21 11:57:52-- https://dclcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz  
Resolving dclcdn.apache.org (dclcdn.apache.org)... 151.101.2.132, 2a04:4e42::644  
Connecting to dclcdn.apache.org (dclcdn.apache.org)|151.101.2.132|:443... connecte  
d.  
HTTP request sent, awaiting response... 200 OK  
Length: 177279333 (169M) [application/x-gzip]  
Saving to: 'pig-0.16.0.tar.gz.1'  
pig-0.16.0.tar.gz.1 94%[=====] 158.94M 5.19MB/s eta 2s
```

Step 2: Go to <https://pig.apache.org/releases.html> and copy the path of the latest version of pig that you want to install. Run the following comment to download Apache Pig in Ubuntu:

\$ wget <https://dclcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz>



```
hadoop@hadoop-VirtualBox: ~  
hadoop@hadoop-VirtualBox:~$ wget https://dclcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz  
$: command not found  
hadoop@hadoop-VirtualBox:~$ wget https://dclcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz  
--2022-06-21 11:57:52-- https://dclcdn.apache.org/pig/pig-0.16.0/pig-0.16.0.tar.gz  
Resolving dclcdn.apache.org (dclcdn.apache.org)... 151.101.2.132, 2a04:4e42::644  
Connecting to dclcdn.apache.org (dclcdn.apache.org)|151.101.2.132|:443... connecte  
d.  
HTTP request sent, awaiting response... 200 OK  
Length: 177279333 (169M) [application/x-gzip]  
Saving to: 'pig-0.16.0.tar.gz.1'  
pig-0.16.0.tar.gz.1 94%[=====] 158.94M 5.19MB/s eta 2s
```

Step 3: To untar pig-0.16.0.tar.gz file run the following command:

```
$ tar xvzf pig-0.16.0.tar.gz
```

Step 4: To create a pig folder and move pig-0.16.0 to the pig folder, execute the following command:

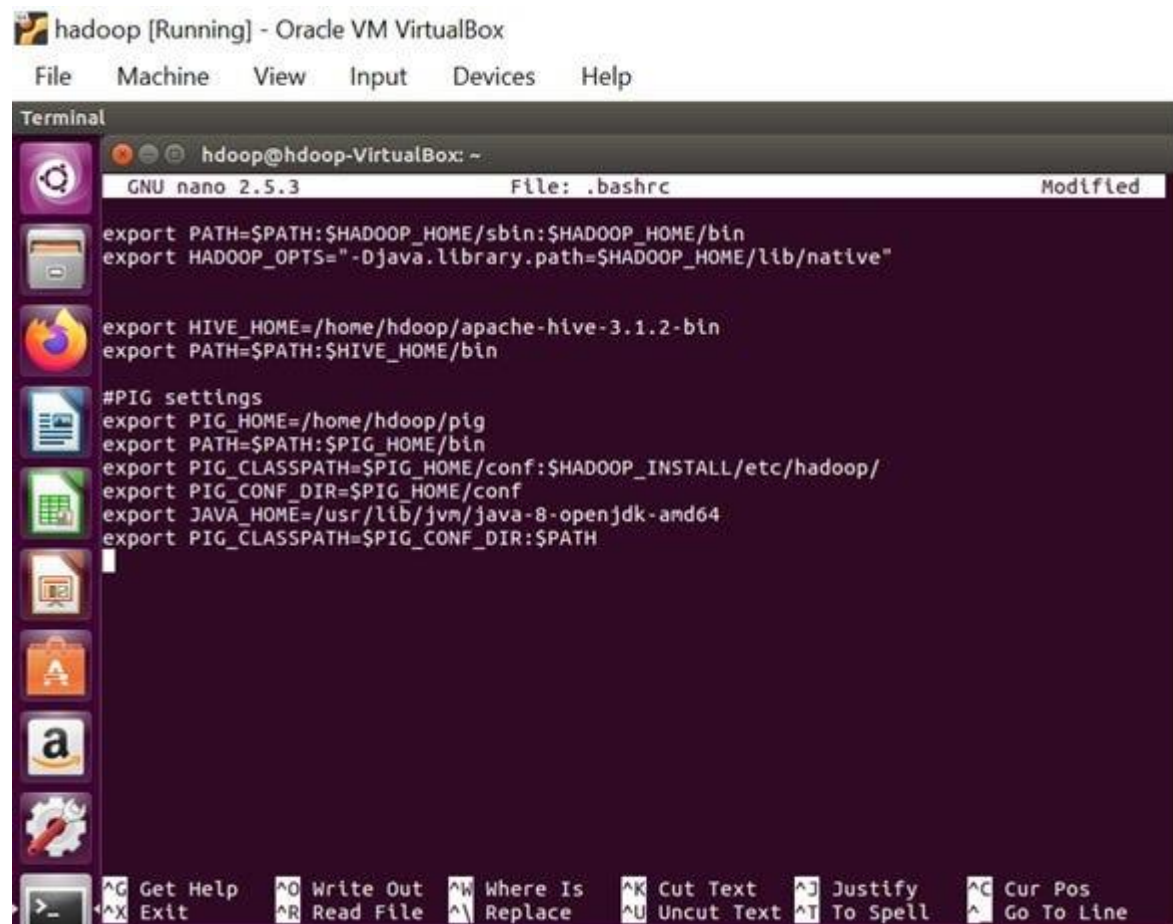
```
$ sudo mv /home/hdoop/pig-0.16.0 /home/hdoop/pig
```

Step 5: Now open the .bashrc file to edit the path and variables/settings for pig. Run the following command:

```
$ sudo nano .bashrc
```

Add the below given to .bashrc file at the end and save the file.

```
#PIG settings
export PIG_HOME=/home/hdoop/pig
export PATH=$PATH:$PIG_HOME/bin
export PIG_CLASSPATH=$PIG_HOME/conf:$HADOOP_INSTALL/etc/hadoop/
export PIG_CONF_DIR=$PIG_HOME/conf
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PIG_CLASSPATH=$PIG_CONF_DIR:$PATH
#PIG setting ends
```



```
hadoop [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

Terminal
hadoop@hdoop-VirtualBox: ~
GNU nano 2.5.3 File: .bashrc Modified

export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"

export HIVE_HOME=/home/hdoop/apache-hive-3.1.2-bin
export PATH=$PATH:$HIVE_HOME/bin

#PIG settings
export PIG_HOME=/home/hdoop/pig
export PATH=$PATH:$PIG_HOME/bin
export PIG_CLASSPATH=$PIG_HOME/conf:$HADOOP_INSTALL/etc/hadoop/
export PIG_CONF_DIR=$PIG_HOME/conf
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export PIG_CLASSPATH=$PIG_CONF_DIR:$PATH

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Step 6: Run the following command to make the changes effective in the .bashrc file:

```
$ source .bashrc
```

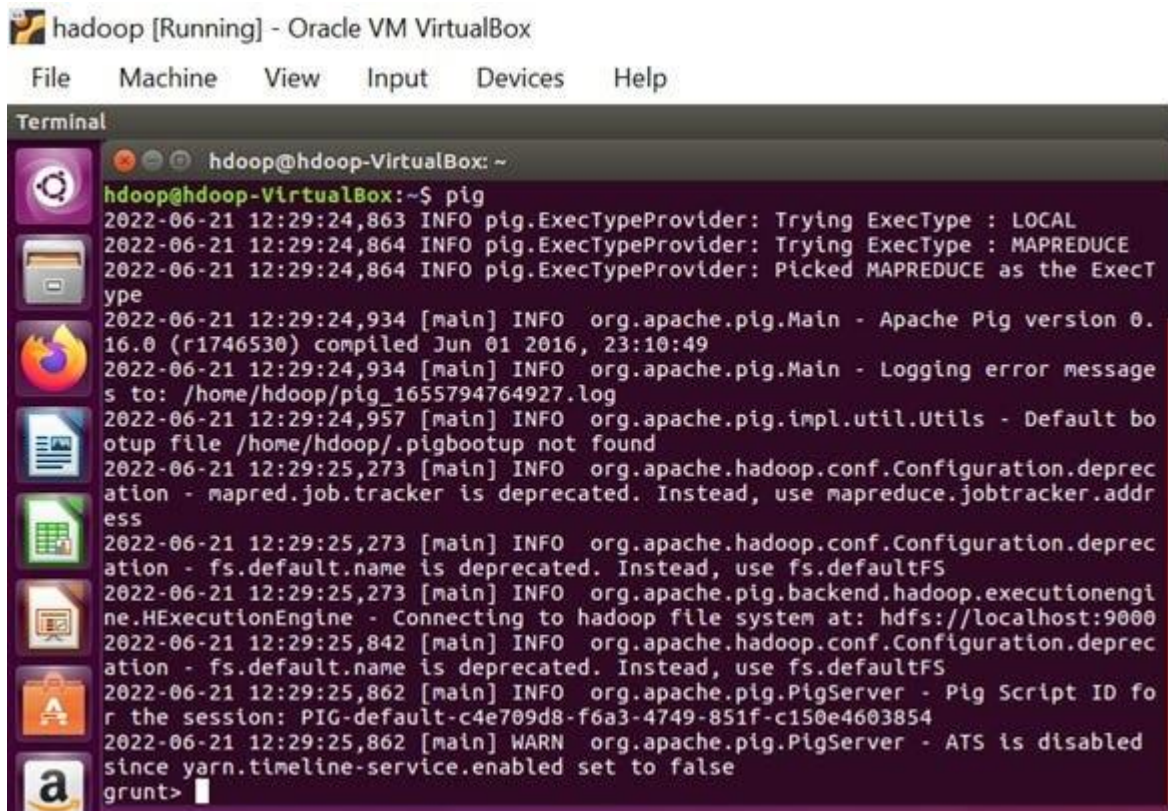

Step 7: To start all Hadoop daemons, navigate to the `hadoop-3.2.1/sbin` folder and run the following commands:

```
$ ./start-dfs.sh $ ./start-yarn$ jps
```

```
hadoop@hadoop-VirtualBox:~$ cd hadoop-3.2.1/sbin
hadoop@hadoop-VirtualBox:~/hadoop-3.2.1/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [hadoop-VirtualBox]
hadoop@hadoop-VirtualBox:~/hadoop-3.2.1/sbin$ ./start-yarn.sh
Starting resource manager
Starting node managers
hadoop@hadoop-VirtualBox:~/hadoop-3.2.1/sbin$ jps
4817 DataNode
5298 ResourceManager
5000 SecondaryNameNode
5450 NodeManager
4683 NameNode
5982 Jps
hadoop@hadoop-VirtualBox:~/hadoop-3.2.1/sbin$
```

Step 8: Now you can launch pig by executing the following command:

```
$ pig
```



Step 9: Now you are in pig and can perform your desired tasks on pig. You can come out of the pig by the quit command:

```
> quit;
```

CREATE USER DEFINED FUNCTION(UDF)

Aim : To create User Define Function in Apache Pig and execute it on map reduce.

Procedure:

Create a sample text file

```
hadoop@Ubuntu:~/Documents$ nano sample.txt
```

Paste the below content to sample.txt

```
1,John
2,Jane
3,Joe
4,Emma
```

```
hadoop@Ubuntu:~/Documents$ hadoop fs -put sample.txt /home/hadoop/piginput/
```

Create PIG File

```
hadoop@Ubuntu:~/Documents$ nano demo_pig.pig
```

paste the below the content to demo_pig.pig

```
-- Load the data from HDFS
```

```
data = LOAD '/home/hadoop/piginput/sample.txt' USING PigStorage(',') AS (id:int>
```

```
-- Dump the data to check if it was loaded correctly
```

```
DUMP data;
```

Run the above file

```
hadoop@Ubuntu:~/Documents$ pig demo_pig.pig
```

```
2024-08-07 12:13:08,791 [main] INFO
```

```
org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil
```

- Total input paths to process : 1

(1,John)

(2,Jane)

(3,Joe)

(4,Emma)

Create udf file and save as uppercase_udf.py

uppercase_udf.py

```
def uppercase(text):
```

```
    return text.upper()
```

```
if __name__ == "__main__":
```

```
    import sys
```

```
    for line in sys.stdin:
```

```
        line = line.strip()
```

```
        result = uppercase(line)
```

```
        print(result)
```

Create the udfs folder on hadoop

```
hadoop@Ubuntu:~/Documents$ hadoop fs -mkdir /home/hadoop/udfs
```

put the uppercase_udf.py in to the abv folder

```
hadoop@Ubuntu:~/Documents$ hdfs dfs -put uppercase_udf.py /home/hadoop/udfs/
```

```
hadoop@Ubuntu:~/Documents$ nano udf_example.pig
```

copy and paste the below content on udf_example.pig

```
-- Register the Python UDF script
```

```
REGISTER 'hdfs:///home/hadoop/udfs/uppercase_udf.py' USING jython AS udf;
```

-- Load some data

```
data = LOAD 'hdfs:///home/hadoop/sample.txt' AS (text:chararray);
```

-- Use the Python UDF

```
uppercased_data = FOREACH data GENERATE udf.uppercase(text) AS uppercase_text;
```

-- Store the result

```
STORE uppercased_data INTO 'hdfs:///home/hadoop/pig_output_data';
```

place sample.txt file on hadoop

```
hadoop@Ubuntu:~/Documents$ hadoop fs -put sample.txt /home/hadoop/
```

To Run the pig file

```
hadoop@Ubuntu:~/Documents$ pig -f udf_example.pig
```

finally u get

Success!

Job Stats (time in seconds):

JobId Maps Reduces MaxMapTimeMinMapTime AvgMapTime MedianMapTime

MaxReduceTime MinReduceTime AvgReduceTime MedianReducetime

Alias Feature Outputs

```
job_local1786848041_0001 1 0 n/a n/a n/a n/a 00 0 0
```

```
data,uppercased_data MAP_ONLY hdfs:///home/hadoop/pig_output_data,
```

Input(s):

Successfully read 4 records (42778068 bytes) from: "hdfs:///home/hadoop/sample.txt"

Output(s):

Successfully stored 4 records (42777870 bytes) in: "hdfs:///home/hadoop/pig_output_data"

Counters:

Total records written : 4

Total bytes written : 42777870

Spillable Memory Manager spill count : 0

Total bags proactively spilled: 0

Total records proactively spilled: 0

Job DAG:

job_local1786848041_0001

2024-08-07 13:33:04,631 [main] WARN
org.apache.hadoop.metrics2.impl.MetricsSystemImpl -
JobTracker metrics system already initialized!

2024-08-07 13:33:04,639 [main] WARN
org.apache.hadoop.metrics2.impl.MetricsSystemImpl -
JobTracker metrics system already initialized!

2024-08-07 13:33:04,644 [main] WARN
org.apache.hadoop.metrics2.impl.MetricsSystemImpl -
JobTracker metrics system already initialized!

2024-08-07 13:33:04,667 [main] INFO
org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher -
Success!

Note :

If any error check jython package is installed and check the path specified on the above steps are give correctly

To check the output file is created

```
hadoop@Ubuntu:~/Documents$ hdfs dfs -ls /home/hadoop/pig_output_data
```

```
Found 2 items
```

If you need to examine the files in the output folder, use:

To view the output

```
hadoop@Ubuntu:~/Documents$ hdfs dfs -cat /home/hadoop/pig_output_data/part-m-00000
```

```
1,JOHN
```

```
2,JANE
```

```
3,JOE
```

```
4,EMMA
```

Result:

Thus the program is executed successfully

Exp5: Installation of Hive on Ubuntu

Aim:

To Download and install Hive, Understanding Startup scripts, Configuration files.

Procedure:

Step 1: Download and extract it

Download the Apache hive and extract it use tar, the commands given below:

```
$wgethttps://downloads.apache.org/hive/hive-3.1.2/apache-hive-3.1.2-bin.tar.gz
```

```
$ tar -xvf apache-hive-3.1.2-bin.tar.gz
```

Step 2: Place different configuration properties in Apache Hive

In this step, we are going to do two things

- Placing Hive Home path in bashrc file

```
$nano .bashrc
```

And append the below lines in it

```
export HIVE_HOME=/home/hadoop/apache-hive-3.1.2-bin
export PATH=$PATH:$HIVE_HOME/bin
export HADOOP_USER_CLASSPATH_FIRST=true
```

2. Exporting **Hadoop path in Hive-config.sh** (To communicate with the Hadoop ecosystem we are defining Hadoop Home path in hive config field) **Open the hive-config.sh as shown in below**

```
$cd apache-hive-3.1.2-bin/bin
```

```
$cp hive-env.sh.template hive-env.sh
```

```
$nano hive-env.sh
```

Append the below commands on it

```
export HADOOP_HOME=/home/Hadoop/Hadoop
```

```
export HIVE_CONF_DIR=/home/Hadoop/apache-hive-3.1.2/conf
```

```
# Set HADOOP_HOME to point to a specific hadoop install directory
# HADOOP_HOME=${bin}/../.. /hadoop
export HADOOP_HOME=/home/hadoop/hadoop
```

```
# Hive Configuration Directory can be controlled by:
# export HIVE_CONF_DIR=
export HIVE_CONF_DIR=/home/hadoop/apache-hive-3.1.2-bin/conf
# Folder containing extra libraries required for hive compilation/execution can be controlled by:
```

Step 3: Install mysql

1. Install mysql in Ubuntu by running this command:

```
$sudo apt update
```

```
$sudo apt install mysql-server
```

2. Alter username and password for MySQL by running below commands:

```
$sudomysql
```

Opens command line interface for MySQL and run the below SQL queries to change username and set password

```
mysql> SELECT user, host, plugin FROM mysql.user WHERE user = 'root';
```

```

hadoop@sanjay-VirtualBox:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 172
Server version: 8.0.34-0ubuntu0.23.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SELECT user, host, plugin FROM mysql.user WHERE user = 'root';
+-----+-----+-----+
| user | host      | plugin                |
+-----+-----+-----+
| root | %         | mysql_native_password |
| root | localhost | auth_socket           |
+-----+-----+-----+
2 rows in set (0.03 sec)

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH 'mysql_native_password' BY 'your_new_password';
Query OK, 0 rows affected (0.04 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.02 sec)

```

```

mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH 'mysql_native_password' BY
'your_new_password';
mysql> FLUSH PRIVILEGES;

```

Step 4: Config hive-site.xml

Config the hive-site.xml by appending this xml code and change the username and password according to your MySQL.

\$cd apache-hive-3.1.2-bin/bin

\$cp hive-default.xml.template hive-site.xml

\$nano hive-site.xml

Append these lines into it

Replace root as your username of MySQL

Replace your_new_password as with your password of MySQL

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true</value>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.cj.jdbc.Driver</value>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>root</value>

</property>

```
<property>
<name>javax.jdo.option.ConnectionPassword</name>
<value>your_new_password</value>
</property>
```

```
<property>
<name>datanucleus.autoCreateSchema</name>
<value>true</value>
</property>
```

```
<property>
<name>datanucleus.fixedDatastore</name>
<value>true</value>
</property>
```

```
<property>
<name>datanucleus.autoCreateTables</name>
<value>True</value>
</property>
```

```
</configuration>
```

Step 5: Setup MySQL java connector:

First, you'll need to download the MySQL Connector/J, which is the JDBC driver for MySQL. You can download it from the below link

https://drive.google.com/file/d/1QFhB7Kvc7a4LzDRe6GcmZva1yAxKz-/view?usp=drive_link

Copy the downloaded MySQL Connector/J JAR file to the Hive library directory. By default, the Hive library directory is usually located at */path/to/apache-hive-3.1.2/lib/* on Ubuntu. Use the following command to copy the JAR file:

```
$sudo cp /path/to/mysql-connector-java-8.0.15.jar /path/to/apache-hive-3.1.2/lib/
Replace /path/to/ with the actual path to the JAR file.
```

Step 6: Initialize the Hive Metastore Schema:

Run the following command to initialize the Hive metastore schema:

```
$$HIVE_HOME/bin/schematool -initSchema -dbTypemysql
```

```
hadoop@sanjay-VirtualBox:~$ schematool --dbType mysql --initSchema
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Metastore connection URL:      jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true
Metastore Connection Driver :  com.mysql.cj.jdbc.Driver
Metastore connection User:     root
```

Step 7: Start hive:

You can test Hive by running the Hive shell: Copy code hive You should be able to run Hive queries, and metadata will be stored in your MySQL database.

\$hive

```
hadoop@sanjay-VirtualBox:~$ hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/hadoop/apache-hive-3.1.2-bin/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/hadoop/hadoop/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 35fe72d8-3ed1-4428-8590-2c88743dedc7

Logging initialized using configuration in jar:file:/home/hadoop/apache-hive-3.1.2-bin/lib/hive-common-3.1.2.jar!/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Hive Session ID = adccd81f-8176-49ba-bda6-f65011e1f514
hive> show databases;
OK
default
Time taken: 0.484 seconds, Fetched: 1 row(s)
```

Result:

Thus, the Apache Hive installation is completed successfully on Ubuntu.

Exp5a: Design and test various schema models to optimize data storage and retrieval Using Hive.

Aim:

To Design and test various schema models to optimize data storage and retrieval Using Hbase.

Procedure:

Step 1: Start Hive

Open a terminal and start Hive by running:

```
$hive
```

Step 2: Create a Database

Create a new database in Hive:

```
hive>CREATE DATABASE financials;
```

```
hive> CREATE DATABASE financials;
```

```
OK
```

```
Time taken: 0.063 seconds
```

Step 3: Use the Database:

Switch to the newly created database:

```
hive>use financials;
```

```
hive> use financials;
```

```
OK
```

```
Time taken: 0.066 seconds
```

Step 4: Create a Table:

Create a simple table in your database:

```
hive>CREATE TABLE finance_table( id INT, name STRING );
```

```
hive> CREATE TABLE finance_table (  
    > id INT,  
    > name STRING  
    > );
```

```
OK
```

```
Time taken: 0.768 seconds
```

Step 5: Load Sample Data:

You can insert sample data into the table:

```
hive>INSERT INTO finance_tableVALUES (1, 'Alice'), (2, 'Bob'), (3, 'Charlie');
```



```
hive> INSERT INTO finance_table VALUES
  > (1, 'Alice'),
  > (2, 'Bob'),
  > (3, 'Charlie');
Query ID = hadoop_20231028192937_fdebeb4e-abf7-4bad-a248-ac908246e3c1
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
2023-10-28 19:29:41,158 Stage-1 map = 0%,  reduce = 0%
```

Step 6: Query Your Data

Use SQL-like queries to retrieve data from your table:

```
hive>CREATE VIEW myview AS SELECT name, id FROM finance_table;
```

Step 7: View the data:

To see the data in the view, you would need to query the view

```
hive>SELECT*FROM myview;
```

```
hive> SELECT * FROM myview;
OK
Alice    1
Bob      2
Charlie  3
Time taken: 0.238 seconds, Fetched: 3 row(s)
```

Step 8: Describe a Table:

You can describe the structure of a table using the DESCRIBE command:

```
hive>DESCRIBE finance_table;
```

```
hive> DESCRIBE finance_table;
OK
id                int
name              string
Time taken: 0.081 seconds, Fetched: 2 row(s)
```

Step 9: Alter a Table:

You can alter the table structure by adding a new column:

```
hive>ALTER TABLE finance_table ADD COLUMNS (age INT);
```

```
hive> ALTER TABLE finance_table ADD COLUMNS (age INT);
OK
Time taken: 0.165 seconds
```

Step 10: Quit Hive:

To exit the Hive CLI, simply type:

```
hive>quit;
```



```
hive> quit;  
hadoop@sanjay-VirtualBox:~/apache-hive-3.1.2-bin/bin$ █
```

Result:

Thus, the usage of various commands in Hive has been successfully completed.