

Exp No: 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 hive 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

```
haresh@fedora:~$ java -version
openjdk version "21.0.4" 2024-07-16
OpenJDK Runtime Environment (Red_Hat-21.0.4.0.7-2) (build 21.0.4+7)
OpenJDK 64-Bit Server VM (Red_Hat-21.0.4.0.7-2) (build 21.0.4+7, mixed mode, sha
ring)
```

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

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

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:

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



```
haresh@fedora:~/Downloads$ ls
apache-hive-3.1.2-bin.tar.gz      hadoop-3.4.0.tar.gz
apache-hive-3.PN_Sf4-n.1.2-bin.tar.gz.part  pig-0.16.0.tar.gz
hadoop-3.3.6.tar.gz
```

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-1.8.0-openjdk
export HADOOP_HOME=/home/haresh/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 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"
export HADOOP_STREAMING=$HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming-3.
export HADOOP_LOG_DIR=$HADOOP_HOME/logs
export PDSH_RCMD_TYPE=ssh
```

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
```

```
GNU nano 7.2                                hadoop-env.sh

##
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!
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk

Location of Hadoop.  By default, Hadoop will attempt to determine
this location based upon its execution path.
export HADOOP_HOME=

Location of Hadoop's configuration information.  i.e., where this
file is living.  If this is not defined, Hadoop will attempt to
```

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

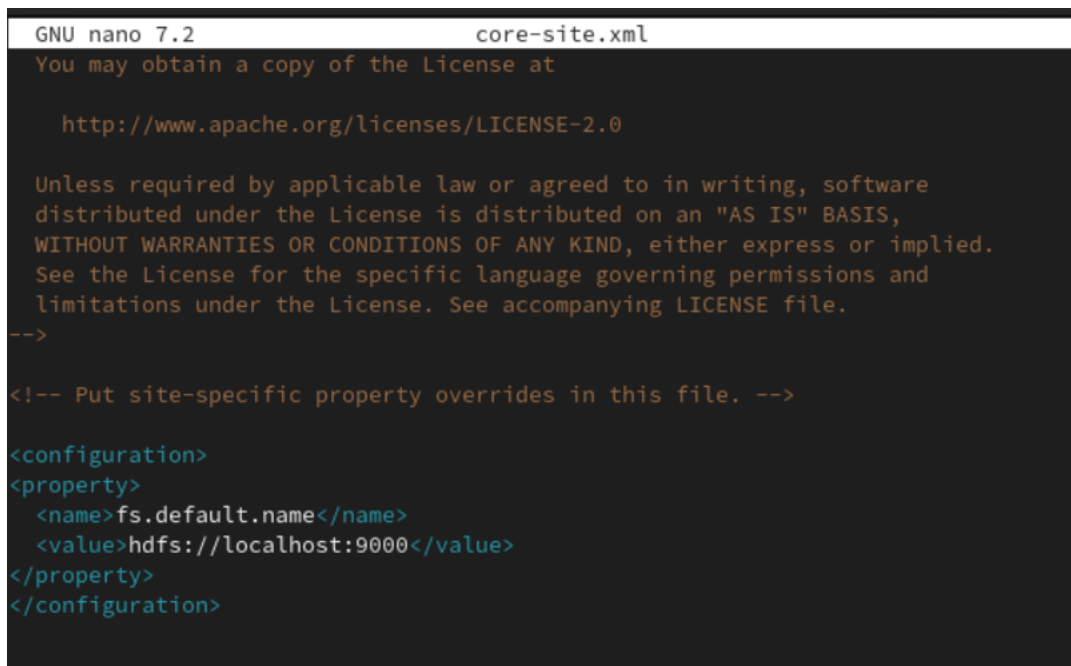
home directory. Run the following command to create both directories:

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

- Next, edit the core-site.xml file and update with your system hostname:

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

Change the following name as per your system hostname:



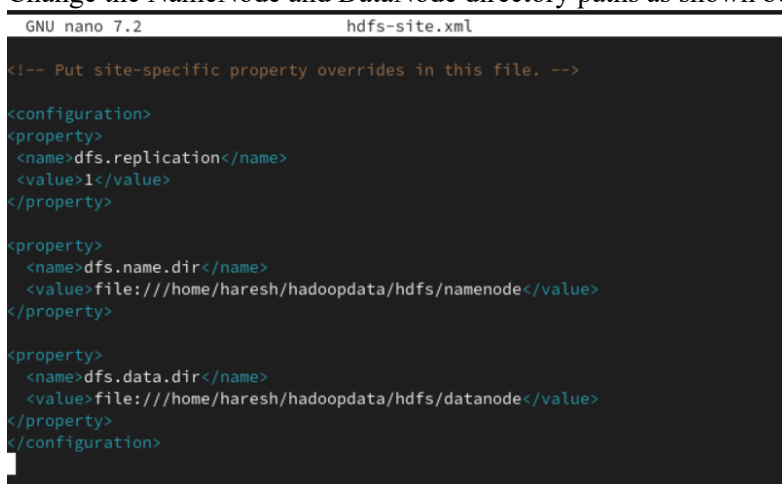
```
GNU nano 7.2 core-site.xml  
You may obtain a copy of the License at  
  
http://www.apache.org/licenses/LICENSE-2.0  
  
Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an "AS IS" BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY 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://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:



```
GNU nano 7.2 hdfs-site.xml  
  
<!-- Put site-specific property overrides in this file. -->  
  
<configuration>  
<property>  
  <name>dfs.replication</name>  
  <value>1</value>  
</property>  
  
<property>  
  <name>dfs.name.dir</name>  
  <value>file:///home/haresh/hadoopdata/hdfs/namenode</value>  
</property>  
  
<property>  
  <name>dfs.data.dir</name>  
  <value>file:///home/haresh/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:

```

GNU nano 7.2                                mapred-site.xml
<configuration>
<property>
<name>mapred.job.tracker</name>
<value>localhost:9001</value>
</property>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
<property>
  <name>yarn.app.mapreduce.am.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
  <name>mapreduce.map.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
  <name>mapreduce.reduce.env</name>
  <value>HADOOP_MAPRED_HOME=$HADOOP_HOME</value>
</property>
<property>
<name>mapreduce.application.classpath</name>
<value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/share/h
</property>
</configuration>

```

Then, edit the yarn-site.xml file:

\$nano \$HADOOP_HOME/etc/hadoop/yarn-site.xml

Make the following changes:

```

GNU nano 7.2                                yarn-site.xml
<!-- Site specific YARN configuration properties -->
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.env-whitelist</name>
<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_
</property>
<property>
<name>yarn.application.classpath</name>
<value>/home/haresh/hadoop/etc/hadoop:/home/haresh/hadoop/share/hadoop/common/ls
</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
```

```
haresh@fedora:~$ sshd service start
sshd re-exec requires execution with an absolute path
haresh@fedora:~$ sshd service start^C
haresh@fedora:~$ service sshd restart
Redirecting to /bin/systemctl restart sshd.service
haresh@fedora:~$ start-all.sh
WARNING: Attempting to start all Apache Hadoop daemons as haresh 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 [fedora]
Starting resourcemanager
Starting nodemanagers
haresh@fedora:~$
```

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

```
$ jps
```

```
Starting nodemanagers
haresh@fedora:~$ jps
3987 NameNode
4467 SecondaryNameNode
4699 ResourceManager
4843 NodeManager
4205 DataNode
5247 Jps
haresh@fedora:~$
```

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 ipconfig 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

```

naresh@fedora:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.103 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 2406:7400:c6:d4bb:bcaf:196c:6f5:3dfb prefixlen 64 scopeid 0x0<global>
    ether 08:00:27:cf:b4:51 txqueuelen 1000 (Ethernet)
    RX packets 558573 bytes 829252898 (790.8 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 167013 bytes 13061657 (12.4 MiB)
    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 979 bytes 160710 (156.9 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 979 bytes 160710 (156.9 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Here my ip address is 192.168.0.103

- 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>

Overview 'localhost:9000' (✓active)

Started:	Sat Oct 12 08:29:23 +0530 2024
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-faa63001-2b11-4758-b71a-eeded471964b
Block Pool ID:	BP-1225564520-10.0.2.15-1724987915688

Summary

Security is off.
Safemode is off.
237 files and directories, 161 blocks (161 replicated blocks, 0 erasure coded block groups) = 398 total filesystem object(s).
Heap Memory used 190.51 MB of 287.5 MB Heap Memory. Max Heap Memory is 1.31 GB.
Non Heap Memory used 52.24 MB of 54.44 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity:	14 GB
Configured Remote Capacity:	0 B
DFS Used:	54.24 MB (0.38%)
Non DFS Used:	8.95 GB
DFS Remaining:	13.95 GB (99.62%)

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

<http://192.168.16:8088>

The screenshot displays the Hadoop Resource Manager web interface. The browser address bar shows `localhost:8088/cluster`. The page title is 'All Applications'. On the left, there is a sidebar with a 'Cluster' menu and a 'Tools' menu. The main content area shows several metrics tables:

- Cluster Metrics:** A table with columns for Apps Submitted, Apps Pending, Apps Running, Apps Completed, Containers Running, Used Resources, Total Resources, and Reserved Resources. All values are 0.
- Cluster Nodes Metrics:** A table with columns for Active Nodes, Decommissioning Nodes, Decommissioned Nodes, Lost Nodes, and Unhealthy Nodes. All values are 0.
- Scheduler Metrics:** A table with columns for Scheduler Type, Scheduling Resource Type, Minimum Allocation, Maximum Allocation, and Maximum. Values include `[memory-mb (unit-M), vcores]`, `<memory:1024, vCores:1>`, `<memory:8192, vCores:4>`, and 0.
- Applications Table:** A table with columns for ID, User, Name, Application Type, Application Tags, Queue, Application Priority, StartTime, LaunchTime, FinishTime, State, FinalStatus, Running Containers, Allocated CPU Vcores, Allocated Memory MB, Allocated GPUs, Reserved CPU Vcores, and Reserved Memory MB. The table is empty, showing 'No data available in table'.

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 /
```

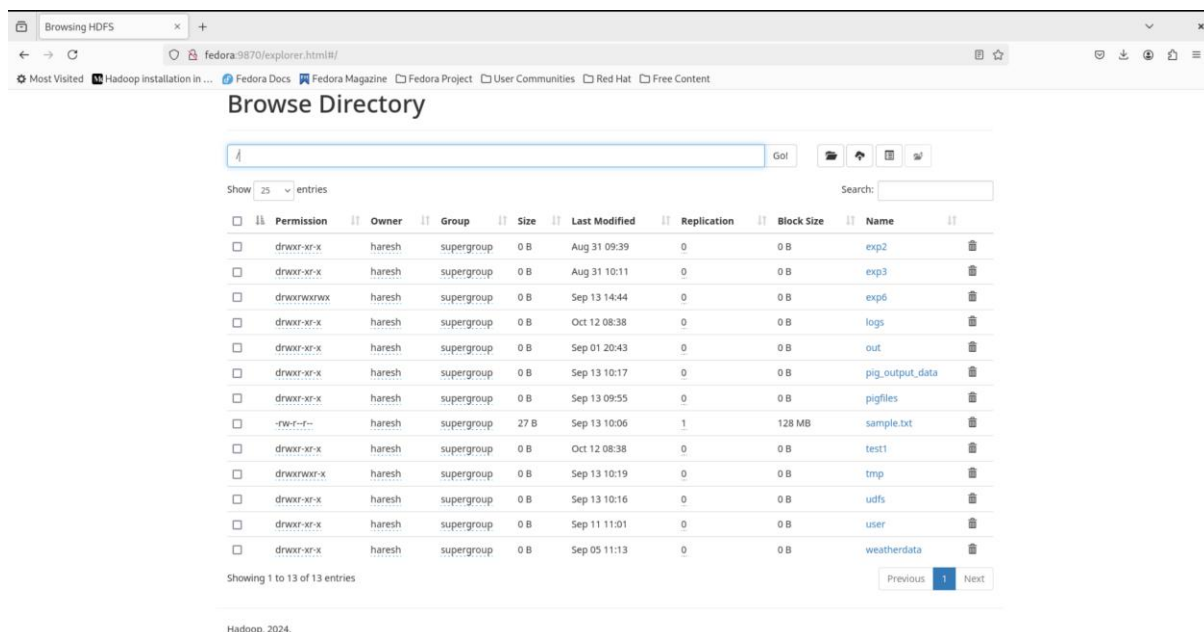
You should get the following output:

```
haresh@fedora:~$ hdfs dfs -ls /
Found 13 items
drwxr-xr-x - haresh supergroup 0 2024-08-31 09:39 /exp2
drwxr-xr-x - haresh supergroup 0 2024-08-31 10:11 /exp3
drwxrwxrwx - haresh supergroup 0 2024-09-13 14:44 /exp6
drwxr-xr-x - haresh supergroup 0 2024-10-12 08:38 /logs
drwxr-xr-x - haresh supergroup 0 2024-09-01 20:43 /out
drwxr-xr-x - haresh supergroup 0 2024-09-13 10:17 /pig_output_data
drwxr-xr-x - haresh supergroup 0 2024-09-13 09:55 /pigfiles
-rw-r--r-- 1 haresh supergroup 27 2024-09-13 10:06 /sample.txt
drwxr-xr-x - haresh supergroup 0 2024-10-12 08:38 /test1
drwxrwxr-x - haresh supergroup 0 2024-09-13 10:19 /tmp
drwxr-xr-x - haresh supergroup 0 2024-09-13 10:16 /udfs
drwxr-xr-x - haresh supergroup 0 2024-09-11 11:01 /user
drwxr-xr-x - haresh supergroup 0 2024-09-05 11:13 /weatherdata
```


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:



Step 15 – Stop Hadoop Cluster

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

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

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

Result:

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