

## Install and Configure Apache Hadoop on Ubuntu 20.04

Apache Hadoop is an open-source software framework used to store, manage and process large datasets for various big data computing applications running under clustered systems. It is Java-based and uses Hadoop Distributed File System (HDFS) to store its data and process data using MapReduce. Apache Hadoop is a collection of utilities that allows you to manage the processing of large datasets across clusters of computers.

Also, it is tolerant to cluster failures. If one of your cluster's crashes, Hadoop can be used to recover data from other nodes.

Now we will learn how to install and configure Apache Hadoop on Ubuntu 20.04.

To install Hadoop, we will have to go through various steps, which include:

- Installing Java and configuring environment variables
- Creating user and configuring SSH
- Installation and configuration of Hadoop

### Prerequisites

- Deploy a [fully updated](#) Ubuntu 20.04 Server.
- Create a [non-root user](#) with sudo access.

**Prepare Your Environment:** Ensure your Ubuntu system is up-to-date by running

```
$ sudo apt update
```

```
$ sudo apt upgrade
```

### Step 1 : Install Java Development Kit and verify

The default Ubuntu repositories contain Java 8 and Java 11 both. I am using Java 8 because hive only works on this version. Use the following command to install it.

```
$ sudo apt install openjdk-8-jdk -y
```

Once you have successfully installed it, check the current Java version:

```
$ java -version
```

## Step 2 : Create Hadoop User and Configure Password-less SSH

First, create a new user named **hadoop**:

```
$ sudo adduser hadoop
```

Add the **hadoop** user to the **sudo** group. It's To enable superuser privileges to the new user. *You are free to use any username and password you see fit.*

```
$ sudo usermod -aG sudo hadoop
```

Once done, switch to the user **hadoop**:

```
$ sudo su - hadoop
```

Next, Install the OpenSSH server and client.

SSH (Secure Shell) installation is important 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 openssh-server openssh-client -y
```

When you get a prompt, respond with:

keep the local version currently installed

Switch to the created user.

```
$ sudo su - hadoop
```

Now configure password-less SSH access for the newly created **hadoop** user, so I didn't enter key to save file and passphrase. Generate an SSH keypair first:

```
$ ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
```

**Add the generated public key from `id_rsa.pub` to `authorized_keys`.**

Copy the generated public key to the authorized key file and set the proper permissions:

```
$ sudo cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Change the permissions of the `authorized_keys` file.

```
$ sudo chmod 640 ~/.ssh/authorized_keys
```

**Verify if the password-less SSH is functional.**

```
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 3: Install Apache Hadoop

Log in with hadoop user.

```
$ sudo su - hadoop
```

Download the latest stable version of Hadoop. To get the latest version, go to Apache Hadoop [official download](https://downloads.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz) page.

```
$ wget https://downloads.apache.org/hadoop/common/hadoop-3.3.1/hadoop-3.3.1.tar.gz
```

Once you are done with the download, Extract the downloaded file using following command

```
$ tar -xvzf hadoop-3.3.1.tar.gz
```

Move the extracted directory to the /usr/local/ directory.

```
$ sudo mv hadoop-3.3.1 /usr/local/hadoop
```

Create directory to store system logs.

```
$ sudo mkdir /usr/local/hadoop/logs
```

Change the ownership of the hadoop directory.

```
$ sudo chown -R hadoop:hadoop /usr/local/hadoop
```

### Step 4: Single Node Hadoop Deployment

#### Step 4.1: Configure Hadoop

Edit file ~/.bashrc to configure the Hadoop environment variables.

```
$ sudo nano ~/.bashrc
```

Add the following lines to the file. Save and close the file.

```
export HADOOP_HOME=/usr/local/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"
```

Activate the environment variables.

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

## Step 4.2: Configure Java Environment Variables

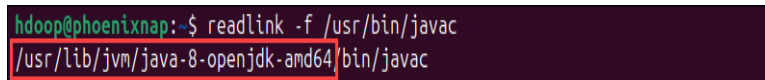
Hadoop has a lot of components that enable it to perform its core functions. To configure these components such as YARN, HDFS, MapReduce, and Hadoop-related project settings, you need to define Java environment variables in `hadoop-env.sh` configuration file.

Find the Java path.

```
$ which javac
```

Find the OpenJDK directory.

```
$ readlink -f /usr/bin/javac
```



```
hadoop@phoenixnap:~$ readlink -f /usr/bin/javac
/usr/lib/jvm/java-8-openjdk-amd64/bin/javac
```

Edit the `hadoop-env.sh` file.

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

Add the following lines to the file. Then, close and save the file.

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64
```

```
export HADOOP_CLASSPATH+=" $HADOOP_HOME/lib/*.jar"
```

Save changes and exit from the text editor.

Next, change your current working directory to **`/usr/local/hadoop/lib:`**

```
$ cd /usr/local/hadoop/lib
```

Here, download the javax activation file:

```
$ sudo wget https://jcenter.bintray.com/javax/activation/javax.activation-api/1.2.0/javax.activation-api-1.2.0.jar
```

Once done, check the Hadoop version in Ubuntu:

```
$ hadoop version
```

**Step 4.3 : Edit the `core-site.xml` configuration file to specify the URL for your NameNode.**

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

Add the following lines.

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://0.0.0.0:9000</value>
    <description>The default file system URI</description>
  </property>
</configuration>
```

Save the changes and exit from the text editor.

Create a directory for storing node metadata and change the ownership to hadoop.

```
$ sudo mkdir -p /home/hadoop/hdfs/{namenode,datanode}
```

```
$ sudo chown -R hadoop:hadoop /home/hadoop/hdfs
```

#### Step 4.4: Edit the hdfs-site.xml configuration file

By configuring the **hdfs-site.xml** file, you will define the location for storing node metadata, fs-image file.

So first open the configuration file:

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

Add the following lines. Close and save the file.

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.name.dir</name>
    <value>file:///home/hadoop/hdfs/namenode</value>
  </property>
```

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

Save changes and exit from the **hdfs-site.xml** file.

#### Step 4.5: Edit the mapred-site.xml file

By editing the **mapred-site.xml** file, you can define the MapReduce values.

To do that, first, open the configuration file using the following command:

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

And paste the following line in between **<configuration> ... </configuration>**:

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

Save and exit from the nano text editor.

#### Step 4.6: Edit the yarn-site.xml file

This is the last configuration file that needs to be edited to use the Hadoop service. The purpose of editing this file is to define the YARN settings.

First, open the configuration file:

```
sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

Add the following lines. Save and close the file.

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

Save changes and exit from the config file.

Log in with hadoop user.

```
$ sudo su - hadoop
```

Finally, use the following command to validate the Hadoop configuration and to format the HDFS NameNode:

```
$ hdfs namenode -format
```

```
cd ~/hadoop-3.3.1/sbin
```

```
ls -lrt
```

## Steps 5: Start the Apache Hadoop Cluster

Start the NameNode and DataNode.

```
$ start-dfs.sh
```

Start the YARN resource and node managers.

```
$ start-yarn.sh
```

Verify all the running components.

```
$ jps
```

- 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:~$
```

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

## Step 6 : Access Hadoop Namenode and Resource Manager :

- First we need to know our ip address, 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
```

```

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:d6c4:f76c:b206:abe3:2d45 prefixlen 64 scopeid 0x0<global>
    inet6 2401:4900:1c28:d6c4:ed13:53f4:5c95:50c6 prefixlen 64 scopeid 0x0<global>
    inet6 fe80::112b:300a:9242:51f3 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:83:13:35 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>**

**Overview** 'localhost:9000' (inactive)

<b>Started:</b>	Sun Sep 10 13:08:22 +0530 2023
<b>Version:</b>	3.3.6, r1be78238728da9266a4f8b195058f08f6012b9c
<b>Compiled:</b>	Sun Jun 18 13:52:00 +0530 2023 by ubuntu from (HEAD detached at release-3.3.6-RC1)
<b>Cluster ID:</b>	CID-dc5a1259-b0c0-4686-a807-fd0dd4c5a9a
<b>Block Pool ID:</b>	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 objects).

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://192.168.1.6:8088>**

**Cluster Metrics**

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

**Cluster Nodes Metrics**

Active Nodes	Decommissioning Nodes	Decommissioned Nodes
1	0	0

**Scheduler Metrics**

Scheduler Type	Scheduling Resource Type	Min
Capacity Scheduler	[memory-mb (unit=MB), vcores]	<memory:1024, vCores:1>

Show: 20 entries

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

## Step 7 :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:



```
hdfs dfs -mkdir /test1
```

```
hdfs dfs -mkdir /logs
```

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

```
hdfs dfs -ls /
```

You should get the following output:

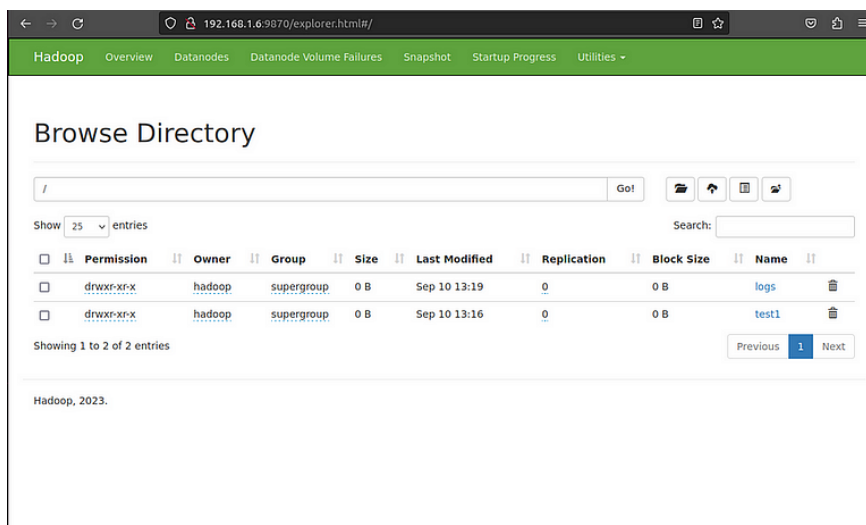
```
hadoop@sanjay-VirtualBox:~/hadoop$ hdfs dfs -ls /
Found 2 items
drwxr-xr-x  - hadoop supergroup      0 2023-09-10 13:16 /logs
drwxr-xr-x  - hadoop supergroup      0 2023-09-10 13:16 /test1
hadoop@sanjay-VirtualBox:~/hadoop$
```

- 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 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 8: To stop hadoop services :

To stop the Hadoop service, run the following command as a hadoop user:

```
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$
```

## HDFS COMMANDS

HDFS is the primary or major component of the Hadoop ecosystem which is responsible for storing large data sets of structured or unstructured data across various nodes and thereby maintaining the metadata in the form of log files. To use the HDFS commands, first you need to start the Hadoop services using the following command:

`start-all.sh`

To check the Hadoop services are up and running use the following command:

`jps`

```
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$ jps
2546 SecondaryNameNode
2404 DataNode
2295 NameNode
2760 ResourceManager
2874 NodeManager
4251 Jps
suraj@suraj:~/hadoop-2.5.0-cdh5.3.2$
```

1. **ls:** This command is used to list all the files. `hdfs dfs -ls <path>`  
EX: `hdfs dfs -ls /` It will print all the directories present in HDFS.

2. **mkdir:** To create a directory

```
hdfs dfs -mkdir <folder name>
```

creating home directory:

```
hdfs/bin -mkdir /user
```

```
hdfs/bin -mkdir /user/username -> write the username of your computer
```

3. **touchz:** It creates an empty file.

```
hdfs dfs -touchz <file_path>
```

EX: `hdfs dfs -touchz /user/myfile.txt`

4. **copyFromLocal (or) put:** To copy files/folders from local file system to hdfs store. Local filesystem means the files present on the OS.

```
hdfs dfs -copyFromLocal <local file path> <dest(present on hdfs)>
```

Let's suppose we have a file *AI.txt* on Desktop which we want to copy to folder **rohit** present on hdfs.

```
hdfs dfs -copyFromLocal ../Desktop/AI.txt /rohit
```

OR

```
hdfs dfs -put ../Desktop/AI.txt /rohit
```

5. **cat:** To print file contents.

```
hdfs dfs -cat <path>
```

print the content of *AI.txt* present inside **rohit** folder.

```
hdfs dfs -cat /rohit/AI.txt
```

6. **copyToLocal (or) get:** To copy files/folders from hdfs store to local file system.

```
hdfs dfs -copyToLocal <<srcfile(on hdfs)> <local file dest>
```

```
hdfs dfs -copyToLocal /rohit ../Desktop/hero
```

(OR)

```
hdfs dfs -get /rohit/myfile.txt ../Desktop/hero
```

7. **moveFromLocal:** This command will move file from local to hdfs.

```
hdfs dfs -moveFromLocal <local src> <dest(on hdfs)>
```

EX: `hdfs dfs -moveFromLocal ../Desktop/cutAndPaste.txt /rohit`

8. **cp:** This command is used to copy files within hdfs.

```
hdfs dfs -cp <src(on hdfs)> <dest(on hdfs)>
```

EX: Lets copy folder *rohit* to *rohit\_copied*.

```
hdfs dfs -cp /rohit /rohit_copied
```

9. **mv:** This command is used to move files within hdfs.

```
hdfs dfs -mv <src(on hdfs)> <dest(on hdfs)>
```

EX: Lets cut-paste a file *myfile.txt* from *rohit* folder to *rohit\_copied*.

```
hdfs dfs -mv /rohit/myfile.txt /rohit_copied
```

10. **rmdir:** This command deletes a file from HDFS.

```
hdfs dfs -rmdir <filename/directoryName>
```

EX: `hdfs dfs -rmdir /rohit_copied` -> It will delete all the content inside the directory then the directory itself.

11. **du:** It will give the size of each file in directory.

```
hdfs dfs -du <dirName>
```

EX: `hdfs dfs -du /rohit`

12. **du:::** This command will give the total size of directory/file.

```
hdfs dfs -du <dirName>
```

EX: `hdfs dfs -du /rohit`

REFERLINK: <https://www.youtube.com/watch?v=jjZIVMPpKTY&list=PLBJYCP7wO2gT-cguBjQnw8j9TPkhhowdK&index=2>