

---

## Group B

### Assignment No: 1

---

**Title of the Assignment:** Write a code in JAVA for a simple WordCount application that counts the number of occurrences of each word in a given input set using the Hadoop MapReduce framework on local-standalone set-up.

---

**Objective of the Assignment:** Students should be able to install and Hadoop MapReduce framework on local-standalone set-up and they should able to Write a code in JAVA for a simple WordCount application that counts the number of occurrences of each word in a given input.

---

**Prerequisite:**

1. **JAVA-Java JDK (installed)**
  2. **HADOOP-Hadoop package (Downloaded)**
- 

**Contents for Theory:**

1. **Hadoop MapReduce framework**
  2. **Procedures to be Follow**
  3. **Installation of Hadoop on Windows**
  4. **Installation of Hadoop on Linux/Unix/Fedora.etc**
  5. **Running a Wordcount Mapreduce Example**
  6. **Java Code for word count**
- 

## 1. Hadoop MapReduce framework

### What is Hadoop?

Hadoop is an open-source software framework used for storing and processing Big Data in a distributed manner on large clusters of commodity hardware. Hadoop is licensed under the Apache v2 license.

Hadoop was developed, based on the paper written by Google on the MapReduce system and it applies concepts of functional programming. Hadoop is written in the Java programming language and ranks among the highest-level Apache projects. Hadoop was developed by **Doug Cutting and Michael J. Cafarella.**

## 2. Procedures to be Follow: -

1. Install Java
2. install Hadoop
3. Configure hadoop
4. Test hadoop installation
5. Create wordcount program
6. Input file to mapreduce
7. Display the output

## 3. How to Install Single Node Cluster Hadoop on Windows?

### Step 1: Verify the Java installed

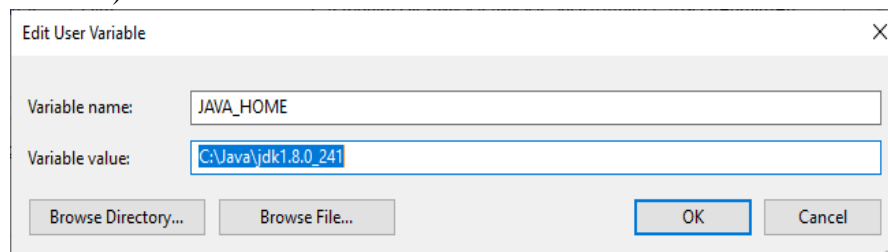
Java is the primary requirement to setup Hadoop on any system, So make sure you have Java installed on your system using the following command.

```
javac -version
```

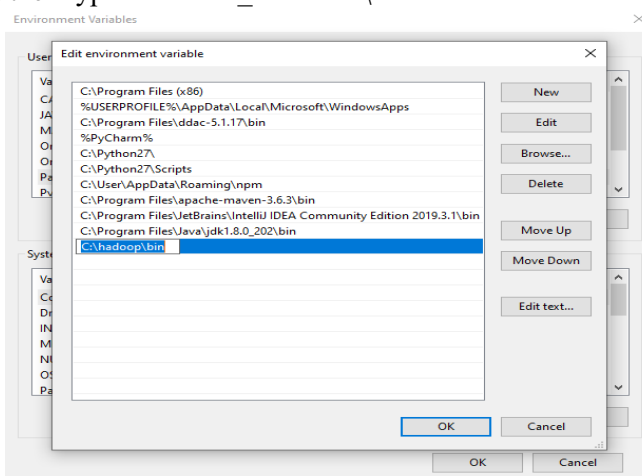
if java is not installed the follow following steps to install Java

### JAVA Installation

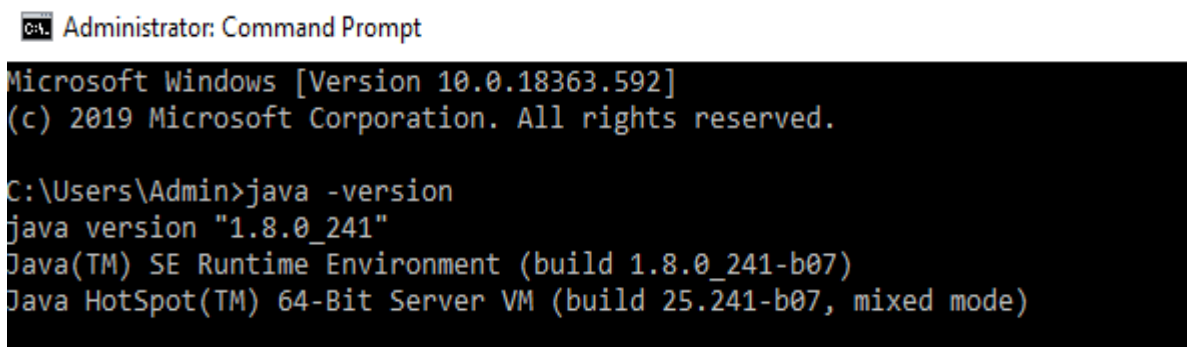
1. Go to official Java Downloading page  
<https://www.oracle.com/java/technologies/javase-jre8-downloads.html>
2. After downloading java, run the jdk-8u241-windows-x64.exe file
3. Follow the instructions and click next.
4. After finishing the installation, it is need to set Java environment variable
5. Go to Start->Edit the System environment variable->Environment variable
6. Then Click new and enter variable name as “JAVA\_HOME”
7. In the value field Enter the java path such as “C:\Java\jdk1.8.0\_241” (Consider your installation folder)



8. Go to path and click edit then type “%JAVA\_HOME%\bin”



9. Then click Ok and Go to Command Prompt
10. Type “Java -version”. If it prints the installed version of java, now java successfully installed in your System.



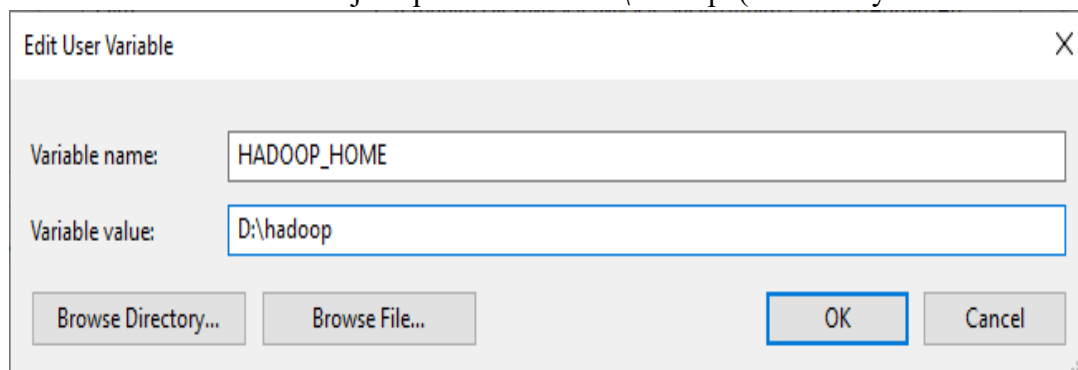
```
Administrator: Command Prompt

Microsoft Windows [Version 10.0.18363.592]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Admin>java -version
java version "1.8.0_241"
Java(TM) SE Runtime Environment (build 1.8.0_241-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)
```

## Step 2: Installing Hadoop

1. Download Hadoop 2.8.0(any latest version of hadoop) from <http://archive.apache.org/dist/hadoop/core/hadoop-2.8.0/hadoop-2.8.0.tar.gz>
2. Extract the tar file ( in my case I used 7-zip to extract the file and I stored the extracted file in the D:\hadoop)
3. After finishing the extraction, it is need to set Hadoop environment variable
4. Go to Start->Edit the System environment variable->Environment variable
5. Then Click new and enter variable name as “HADOOP\_HOME”
6. In the value field Enter the java path such as “D:\hadoop”(Consider your installation folder)



7. Go to path and click edit then type “%HADOOP\_HOME%\bin”



8. Now we have to configure the hadoop.

## Step 3: Hadoop Configuration:

For Hadoop Configuration we need to modify Six files that are listed below-

1. Core-site.xml
2. Mapred-site.xml
3. Hdfs-site.xml
4. Yarn-site.xml
5. Hadoop-env.cmd
6. Create two folders datanode and namenode

Go to D:/hadoop/etc/hadoop/.. folder, find the below mentioned files and paste the following.

**Step 3.1: Core-site.xml configuration**

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

**Step 3.2: Mapred-site.xml configuration**

Rename "**mapred-site.xml.template**" to "**mapred-site.xml**" and edit this file D:/Hadoop/etc/hadoop/mapred-site.xml, paste below xml paragraph and save this file.

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

**Step 3.3: Hdfs-site.xml configuration**

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>C:\hadoop-2.8.0\data\namenode</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>C:\hadoop-2.8.0\data\datanode</value>
  </property>
</configuration>
```

**Step 3.4: Yarn-site.xml configuration**

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```

**Step 3.5: Hadoop-env.cmd configuration**

Set "JAVA\_HOME=C:\Java" (On C:\java this is path to file jdk.18.0)

Edit file D:\Hadoop\etc\hadoop\**hadoop-env.cmd** by closing the command line "JAVA\_HOME=%JAVA\_HOME%" instead of set "JAVA\_HOME= C:\Java\jdk1.8.0\_241" (if your java file in Program Files the instead of give **Progra~1** otherwise you will get JAVA\_HOME incorrectly set error)

**Step 3.6: Create datanode and namenode folders**

1. Create folder "data" under "D:\Hadoop"
2. Create folder "datanode" under "D:\Hadoop\data"
3. Create folder "namenode" under "D:\Hadoop\data" data

Download file Hadoop Configuration.zip <https://github.com/Prithiviraj2503/hadoop-installation-windows>

Delete file bin on D:\Hadoop\bin and replace it by the bin file of Downloaded configuration file (from Hadoop Configuration.zip).

### Format the namenode folder

Open cmd and typing command "hdfs namenode -format". You will see through command prompt which tasks are processing, after completion you will get a message like namenode format successfully and shutdown message.

## Step 4: Testing Hadoop Installation

1. Open Cmd and type the following "Hadoop -version"

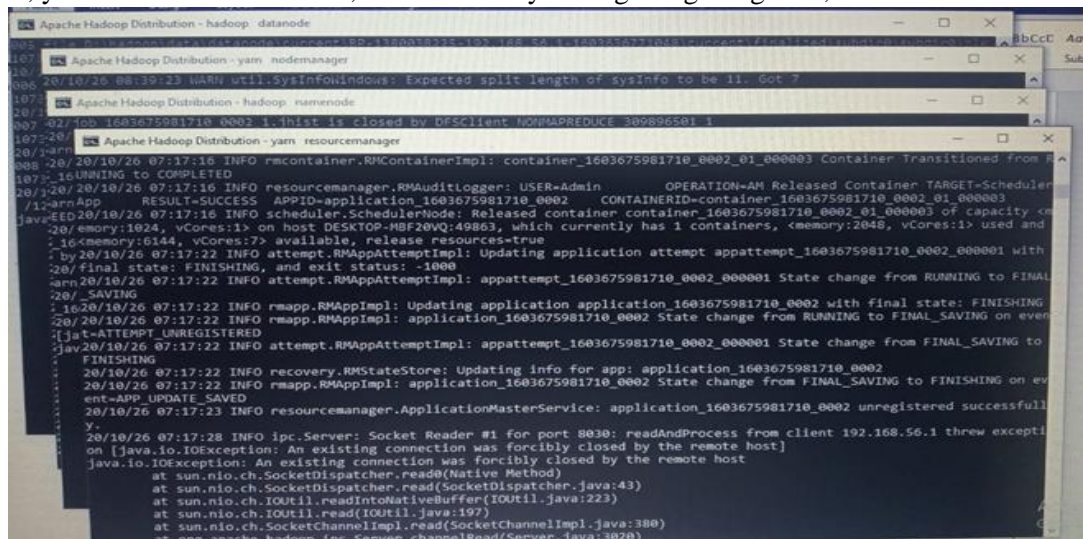
```
C:\Users\Admin>hadoop -version
java version "1.8.0_241"
Java(TM) SE Runtime Environment (build 1.8.0_241-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)
```

2. To start the hadoop locate to "D:\hadoop\sbin" via command prompt and press **start-all.cmd**

Administrator: Command Prompt

```
C:\Users\Admin>D:
D:\>cd hadoop/sbin
D:\hadoop\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons
```

Now, you can see the namenode, datanode and yarn engines getting start,



3. Now type "jps". JPS (Java Virtual Machine Process Status Tool) is a command is used to check all the Hadoop daemons like NameNode, DataNode, ResourceManager, NodeManager etc.

```

D:\hadoop\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

D:\hadoop\sbin>jps
5296 NameNode
2372 Jps
9192 ResourceManager
10140 NodeManager
9420 DataNode

```

4. Open: <http://localhost:8088> in any browser

**Nodes of the cluster**

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved
0	0	0	0	0	0 B	8 GB	0 B	0	8	0

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes	Shutdown Nodes
1	0	0	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation	Maximum Cluster Application Priority
Capacity Scheduler	[MEMORY]	<memory:1024, vCores:1>	<memory:8192, vCores:4>	0

Showing 1 to 1 of 1 entries

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail	VCores Used	VCores Avail	Version
/default-rack		RUNNING	DESKTOP-MBF20VQ-49863	DESKTOP-MBF20VQ-8042	Mon Oct 26 07:13:09 +0530 2020		0	0 B	8 GB	0	8	2.8.0

5. Open: <http://localhost:50070> in any browser

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

Started:	Mon Oct 26 07:03:02 +0530 2020
Version:	2.8.0, r91f2b7a13d1e97be85db92dabc627cc29ac0009
Compiled:	Fri Mar 17 09:42:00 +0530 2017 by jdu from branch-2.8.0
Cluster ID:	
Block Pool ID:	

**Summary**

Security is off  
Safemode is off.

Now hadoop succesfully installed in your System.

## Step 5: Create wordcount program

1. After successful hadoop installation we need to create an directory in the hadoop file system
2. Start the hadoop via command prompt \$ **start-all.cmd**
3. By using \$**jps** command Ensure hadoop nodes are running
4. To create a directory, use: \$ **hadoop fs -mkdir /inputdir**
5. To input a file within a directory, use: \$ **hadoop fs -put D:/input\_file.txt/inputdir**
6. To ensure wether your file succesfully imported, use: \$ **hadoop fs -ls /inputdir/**
7. To view the content of the file, use: \$ **hadoop dfs -cat /inputdir/input\_file.txt**



```

Administrator: Command Prompt
D:\hadoop\sbin>hadoop fs -mkdir /inputdir

D:\hadoop\sbin>hadoop fs -put D:/input_file.txt /inputdir

D:\hadoop\sbin>hadoop fs -ls /inputdir/
Found 1 items
-rw-r--r-- 1 Admin supergroup 1888 2020-10-26 07:10 /inputdir/input_file.txt

D:\hadoop\sbin>hadoop dfs -cat /inputdir/input_file.txt
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.
23 23 27 43 24 25 26 26 26 25 26 25
26 27 28 28 28 30 31 31 31 30 30 29
31 32 32 32 33 34 35 36 36 34 34 34
39 38 39 39 39 41 42 43 40 39 38 40
38 39 39 39 39 41 41 41 28 40 39 45
23 23 27 43 24 25 26 26 26 25 26 25
26 27 28 28 28 30 31 31 31 30 30 29
31 32 32 32 33 34 35 36 36 34 34 34
39 38 39 39 39 41 42 43 40 39 38 40
38 39 39 39 39 41 41 41 28 40 39 45
23 23 27 43 24 25 26 26 26 25 26 25
26 27 28 28 28 30 31 31 31 30 30 29
31 32 32 32 33 34 35 36 36 34 34 34
39 38 39 39 39 41 42 43 40 39 38 40
38 39 39 39 39 41 41 41 28 40 39 45
23 23 27 43 24 25 26 26 26 25 26 25
26 27 28 28 28 30 31 31 31 30 30 29
31 32 32 32 33 34 35 36 36 34 34 34
39 38 39 39 39 41 42 43 40 39 38 40
38 39 39 39 39 41 41 41 28 40 39 45
23 23 27 43 24 25 26 26 26 25 26 25
26 27 28 28 28 30 31 31 31 30 30 29
31 32 32 32 33 34 35 36 36 34 34 34
39 38 39 39 39 41 42 43 40 39 38 40
38 39 39 39 39 41 41 41 28 40 39 45
23 23 27 43 24 25 26 26 26 25 26 25
26 27 28 28 28 30 31 31 31 30 30 29
31 32 32 32 33 34 35 36 36 34 34 34
39 38 39 39 39 41 42 43 40 39 38 40
38 39 39 39 39 41 41 41 28 40 39 45
D:\hadoop\sbin>hadoop jar D:/MapReduceClient.jar wordcount /input_dir /output_dir
20/10/26 07:15:19 INFO client.RMPProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/10/26 07:15:22 INFO mapreduce.JobSubmitter: Cleaning up the staging area /tmp/hadoop-yarn/staging/Adm

```

8. Now apply mapreduce program to the input file. We have a mapReduceClient.jar which contain java mapper and reducer programs. After applying the jar file you can see the task performed in the mapreduce phase. All the results of completed tasks will be printed in the command prompt.
9. After completed the mapreduce tasks the output will be stored in the **output\_dir** directory. To see the output, use: **\$ hadoop dfs -cat /output\_dir/**
10. To stop the hadoop type **\$stop-all.cmd**

```

D:\hadoop\sbin>stop-all.cmd
This script is Deprecated. Instead use stop-dfs.cmd and stop-yarn.cmd
SUCCESS: Sent termination signal to the process with PID 9340.
SUCCESS: Sent termination signal to the process with PID 10652.
stopping yarn daemons
SUCCESS: Sent termination signal to the process with PID 8576.
SUCCESS: Sent termination signal to the process with PID 11128.

INFO: No tasks running with the specified criteria.

D:\hadoop\sbin>

```

Now the hadoop single node cluster was installed successfully and the simple word count program were executed successfully in your windows system.

## 4. Installation of Hadoop on Linux/Unix/Fedora.etc

### Step 1: Verify the Java installed

Run the following command to update package index:

```
sudo apt update
```

However, the **Fedora** distribution **does not use the apt package manager**. Fedora uses yum instead

```
sudo yum update
```

**Check whether Java is installed already:**

```
java -version
```

**Command 'java' not found, but can be installed with:**

```
sudo apt install default-jre
sudo apt install openjdk-11-jre-headless
sudo apt install openjdk-8-jre-headless
```

**Install OpenJDK via the following command:**

```
sudo apt-get install openjdk-8-jdk
```

**Check the version installed:**

```
openjdk version "11.0.17" 2022-10-18
OpenJDK Runtime Environment (Red_Hat-11.0.17.0.8-2.fc35) (build 11.0.17+8)
OpenJDK 64-Bit Server VM (Red_Hat-11.0.17.0.8-2.fc35) (build 11.0.17+8, mixed mode, sharing)
```

### Step 2: Installing Hadoop

#### Download Hadoop binary

Go to release page of Hadoop website to find a download URL for Hadoop 3.3.0:

```
https://hadoop.apache.org/releases.html
```

OR

Run the following command in Ubuntu terminal to download a binary from the internet:

```
wget http://mirror.intergrid.com.au/apache/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
```

Wait until the download is completed:

```
--2023-05-06 16:05:56-- http://mirror.intergrid.com.au/apache/hadoop/common/hadoop-3.3.0/hadoop-3.3.0.tar.gz
Resolving mirror.intergrid.com.au (mirror.intergrid.com.au)... 43.245.161.202
Connecting to mirror.intergrid.com.au (mirror.intergrid.com.au)|43.245.161.202|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 500749234 (478M) [application/x-gzip]
Saving to: 'hadoop-3.3.0.tar.gz'

hadoop-3.3.0.tar.gz 100%[=====] 477.55M 842KB/s in 10m 19s

2023-05-06 16:16:15 (790 KB/s) - 'hadoop-3.3.0.tar.gz' saved [500749234/500749234]
```

**Extract the Hadoop tar File:**

```
tar -xvf hadoop-3.3.0.tar.gz
```

**Add the Hadoop and Java paths in the bash file (.bashrc):**



Open. bashrc file. Now, add Hadoop and Java Path as shown below:

```
# User specific aliases and functions

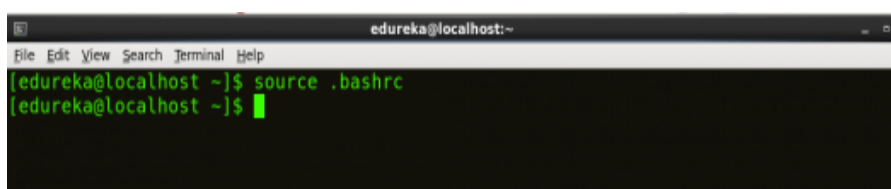
export HADOOP_HOME=$HOME/hadoop-2.7.3
export HADOOP_CONF_DIR=$HOME/hadoop-2.7.3/etc/hadoop
export HADOOP_MAPRED_HOME=$HOME/hadoop-2.7.3
export HADOOP_COMMON_HOME=$HOME/hadoop-2.7.3
export HADOOP_HDFS_HOME=$HOME/hadoop-2.7.3
export YARN_HOME=$HOME/hadoop-2.7.3
export PATH=$PATH:$HOME/hadoop-2.7.3/bin

Set JAVA_HOME

export JAVA_HOME=/home/edureka/jdk1.8.0_101
export PATH=/home/edureka/jdk1.8.0_101/bin:$PATH
```

Then, save the bash file and close it.

For applying all these changes to the current Terminal, execute the source command.



```
edureka@localhost:~
File Edit View Search Terminal Help
[edureka@localhost ~]$ source .bashrc
[edureka@localhost ~]$
```

### Step 3: Hadoop Configuration:

For Hadoop Configuration we need to modify Six files that are listed below-

1. Core-site.xml
2. Mapred-site.xml
3. Hdfs-site.xml
4. Yarn-site.xml
5. Hadoop-env.cmd
6. Create two folders datanode and namenode

Go to `hadoop-*.*/etc/Hadoop/` directory folder, find the below mentioned files and paste the following.

#### Step 3.1: Core-site.xml configuration

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

#### Step 3.2: Mapred-site.xml configuration

Edit the mapred-site.xml file and edit the property mentioned below inside configuration tag:

mapred-site.xml contains configuration settings of MapReduce application like number of JVM that can run in parallel, the size of the mapper and the reducer process, CPU cores available for a process, etc.

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

In some cases, mapred-site.xml file is not available. So, we have to create the mapred-site.xml file using mapred-site.xml template.

```
cp mapred-site.xml.template mapred-site.xml
[edureka@localhost hadoop]$ cp mapred-site.xml.template mapred-site.xml
[edureka@localhost hadoop]$
```

### Step 3.3: Hdfs-site.xml configuration

Edit hdfs-site.xml and edit the property mentioned below inside configuration tag:

hdfs-site.xml contains configuration settings of HDFS daemons (i.e. NameNode, DataNode, Secondary NameNode). It also includes the replication factor and block size of HDFS.

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

### Step 3.4: Yarn-site.xml configuration

Edit yarn-site.xml and edit the property mentioned below inside configuration tag:

yarn-site.xml contains configuration settings of ResourceManager and NodeManager like application memory management size, the operation needed on program & algorithm, etc.

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

### Step 3.5: Hadoop-env.sh configuration

Edit hadoop-env.sh and add the Java Path as mentioned below:

hadoop-env.sh contains the environment variables that are used in the script to run Hadoop like Java home path, etc.

```
# The java implementation to use.
export JAVA_HOME=/home/edureka/jdk1.8.0_101
```

### Step 3.6: Go to Hadoop home directory and format the NameNode.

Open cmd and typing command "hdfs namenode -format". You will see through command prompt which tasks are processing, after completion you will get a message like namenode format successfully and shutdown message.

```
[edureka@localhost hadoop]$ cd
[edureka@localhost ~]$ cd hadoop-2.7.3
[edureka@localhost hadoop-2.7.3]$ bin/hadoop namenode -format
```

This formats the HDFS via NameNode. This command is only executed for the first time. Formatting the file system means initializing the directory specified by the dfs.name.dir variable.

Never format, up and running Hadoop filesystem. You will lose all your data stored in the HDFS.

## Step 4: Testing Hadoop Installation

To make sure that Java and Hadoop have been properly installed on your system and can be accessed through the Terminal, execute the `java -version` and `hadoop version` commands.

**Command:** `java -version`

**Command:** `hadoop version`

### 1. start all the daemons:

Once the NameNode is formatted, go to `hadoop-2.7.3/sbin` directory and start all the daemons.

```
cd hadoop-2.7.3/sbin
```

Either you can start all daemons with a single command or do it individually.

```
./start-all.sh
```

The above command is a combination of `start-dfs.sh`, `start-yarn.sh` & `mr-jobhistory-daemon.sh`

Or you can run all the services individually as below:

#### 1. Start NameNode:

The NameNode is the centerpiece of an HDFS file system. It keeps the directory tree of all files stored in the HDFS and tracks all the file stored across the cluster.

**Command:** `./hadoop-daemon.sh start namenode`

#### 2. Start DataNode:

On startup, a DataNode connects to the Namenode and it responds to the requests from the Namenode for different operations.

**Command:** `./hadoop-daemon.sh start datanode`

#### 3. Start ResourceManager:

ResourceManager is the master that arbitrates all the available cluster resources and thus helps in managing the distributed applications running on the YARN system. Its work is to manage each NodeManagers and the each application's ApplicationMaster.

**Command:** `./yarn-daemon.sh start resourcemanager`

#### 4. Start NodeManager:

The NodeManager in each machine framework is the agent which is responsible for managing containers, monitoring their resource usage and reporting the same to the ResourceManager.

**Command:** `./yarn-daemon.sh start nodemanager`

#### 5. Start JobHistoryServer:

JobHistoryServer is responsible for servicing all job history related requests from client.

**Command:** `./mr-jobhistory-daemon.sh start historyserver`

### 2. check all the daemons:

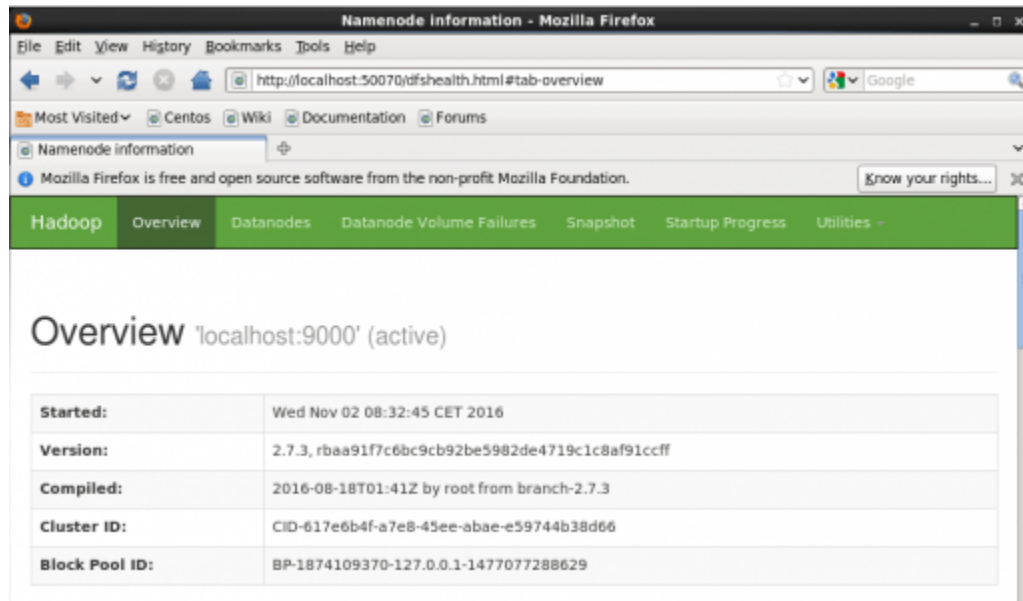
To check that all the Hadoop services are up and running, run the below command.

**Command:** `jps`

```
[edureka@localhost sbin]$ ./mr-jobhistory-daemon.sh start historyserver
starting historyserver, logging to /home/edureka/hadoop-2.7.3/logs/mapred-edureka-h
istoryserver-localhost.localdomain.out
[edureka@localhost sbin]$ jps
22113 NameNode
22310 ResourceManager
22694 JobHistoryServer
22727 Jps
22206 DataNode
22559 NodeManager
[edureka@localhost sbin]$
```

### 3. check the NameNode interface.

Now open the Mozilla browser and go to [localhost:50070/dfshealth.html](http://localhost:50070/dfshealth.html) to check the NameNode interface.



Congratulations, you have successfully installed a single-node Hadoop cluster in one go.

## 5. Hadoop – Running a Wordcount Mapreduce Example

run a wordcount mapreduce example in hadoop using command line. This can be also an initial test for your Hadoop setup testing.

### Step 1: Prerequisites:

You must have running hadoop setup on your system. If you don't have hadoop installed visit Hadoop installation on Linux tutorial(<https://tecadmin.net/setup-hadoop-2-4-single-node-cluster-on-linux/?amp>).

### Step 2: Copy Files to Namenode Filesystem

After successfully formatting namenode, you must have start all Hadoop services properly. Now create a directory in hadoop filesystem.

```
$ hdfs dfs -mkdir -p /user/hadoop/input
```

Copy copy some text file to hadoop filesystem inside input directory. Here I am copying LICENSE.txt to it. You can copy more that one files.

```
$ hdfs dfs -put LICENSE.txt /user/hadoop/input/
```

### Step 3: Running Wordcount Command

Now run the wordcount mapreduce example using following command. Below command will read all files from input folder and process with mapreduce jar file. After successful completion of task results will be placed on output directory.

```
$ cd $HADOOP_HOME
$hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar
wordcount input output
```

### Step 4: Show Results

First check the names of result file created under dfs@/user/hadoop/output filesystem using following command.

```
$ hdfs dfs -ls /user/hadoop/output
```

Now show the content of result file where you will see the result of wordcount. You will see the count of each word.

```
$ hdfs dfs -cat /user/hadoop/output/part-r-00000
```



```
hadoop@localhost:~/hadoop
File Edit View Search Terminal Help
[hadoop@localhost hadoop]$ hdfs dfs -cat /user/hadoop/output/part-r-00000
16/08/23 21:45:54 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usi
ng builtin-java classes where applicable
(http://www.apache.org/).      1
Apache 1
Foundation 1
Software 1
The 1
This 1
by 1
developed 1
includes 1
product 1
software 1
[hadoop@localhost hadoop]$
```

## 6. Java Code for word count

*/\*WC\_Mapper.java\*/*

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reporter;
public class WC_Mapper extends MapReduceBase implements
Mapper<LongWritable,Text,Text,IntWritable>{
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    public void map(LongWritable key, Text value,OutputCollector<Text,IntWritable> output,
        Reporter reporter) throws IOException{
        String line = value.toString();
        StringTokenizer tokenizer = new StringTokenizer(line);
        while (tokenizer.hasMoreTokens()){
            word.set(tokenizer.nextToken());
            output.collect(word, one);
        }
    }
}
```

*/\*WC\_Reducer.java \*/*

```
import java.io.IOException;
import java.util.Iterator;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
```



```
public class WC_Reducer extends MapReduceBase implements
Reducer<Text,IntWritable,Text,IntWritable> {
    public void reduce(Text key, Iterator<IntWritable>values,OutputCollector<Text,IntWritable>
output,
    Reporter reporter) throws IOException {
        int sum=0;
        while (values.hasNext()) {
            sum+=values.next().get();
        }
        output.collect(key,new IntWritable(sum));
    }
}
```

```
/*WC_Runner.java*/
```

```
import java.io.IOException;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.TextOutputFormat;
public class WC_Runner {
    public static void main(String[] args) throws IOException{
        JobConf conf = new JobConf(WC_Runner.class);
        conf.setJobName("WordCount");
        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(IntWritable.class);
        conf.setMapperClass(WC_Mapper.class);
        conf.setCombinerClass(WC_Reducer.class);
        conf.setReducerClass(WC_Reducer.class);
        conf.setInputFormat(TextInputFormat.class);
```

```
        conf.setOutputFormat(TextOutputFormat.class);  
        FileInputFormat.setInputPaths(conf,new Path(args[0]));  
        FileOutputFormat.setOutputPath(conf,new Path(args[1]));  
        JobClient.runJob(conf);  
    }  
}
```