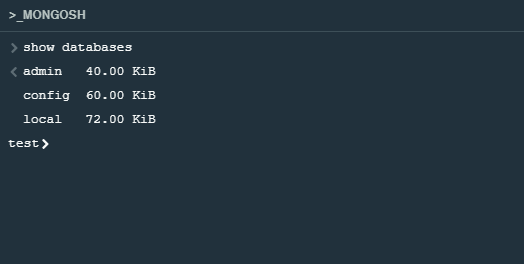
**Practical 1**

**Q1) Show existing Databases and create a new DB MSC.**

🡪

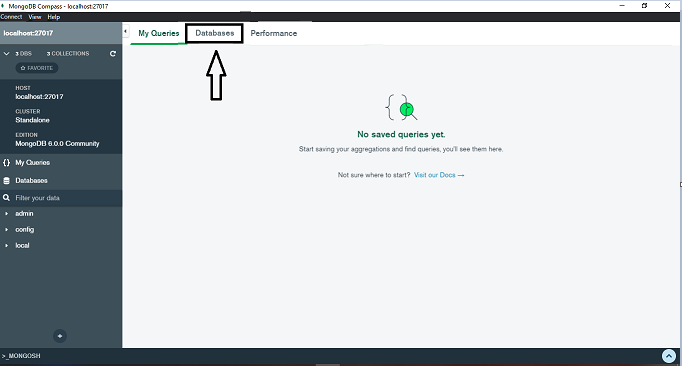
## Showing existing Databases

show databases

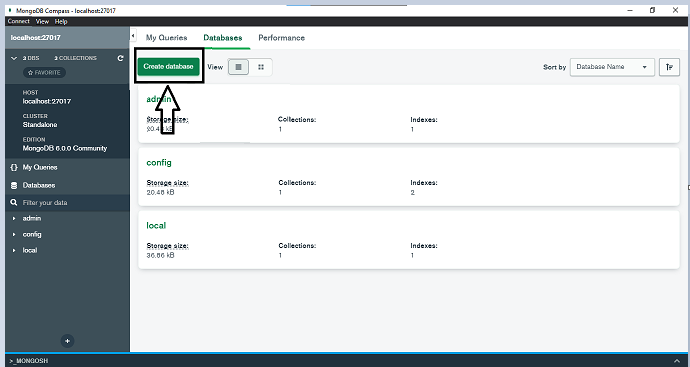


## Creating New Database

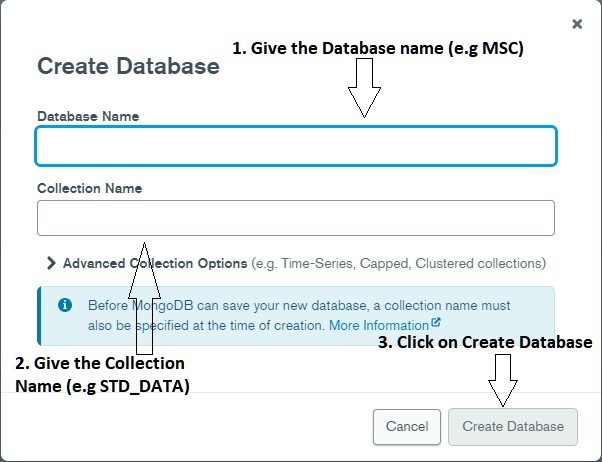
### **Step 1**: Click on the Databases Tab



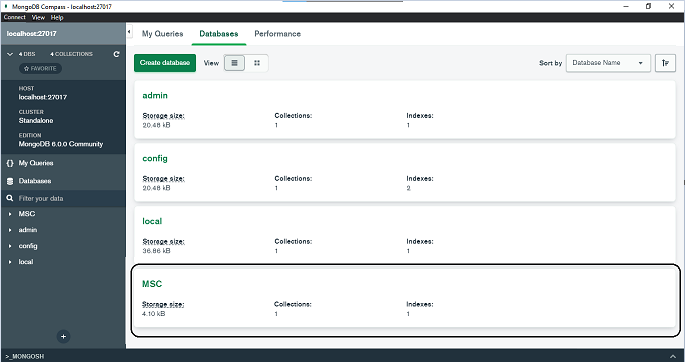
### **Step 2**: Click on Create Database



### **Step 3:** Assign the database name and the collection name and click on the **Create Database**

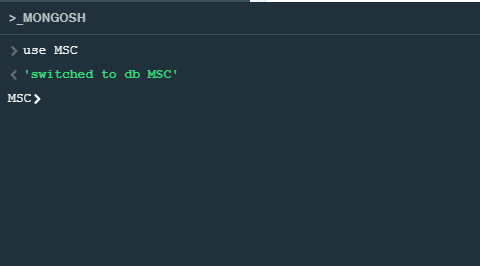


### **Step 4:** After the Create Database you will be able to see the database



## Switching database to MSC

use MSC

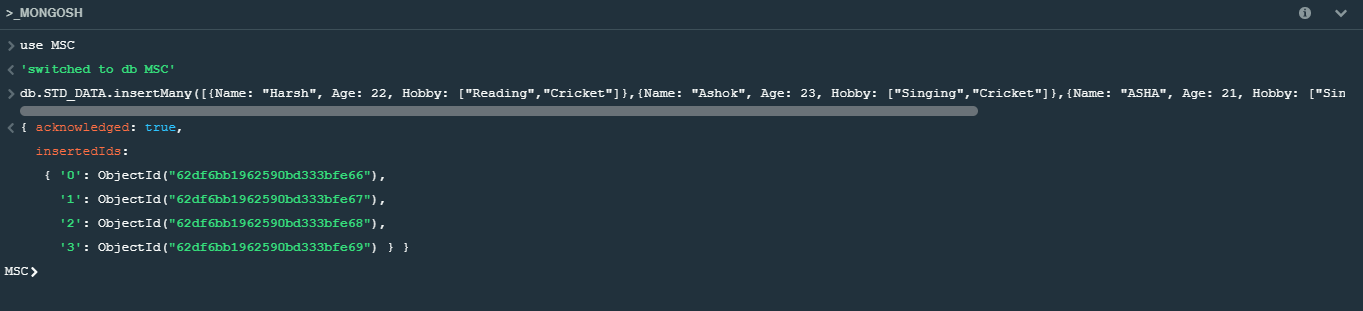


**Q2. Create a new collection Students and add values and display all records.**

**🡪**

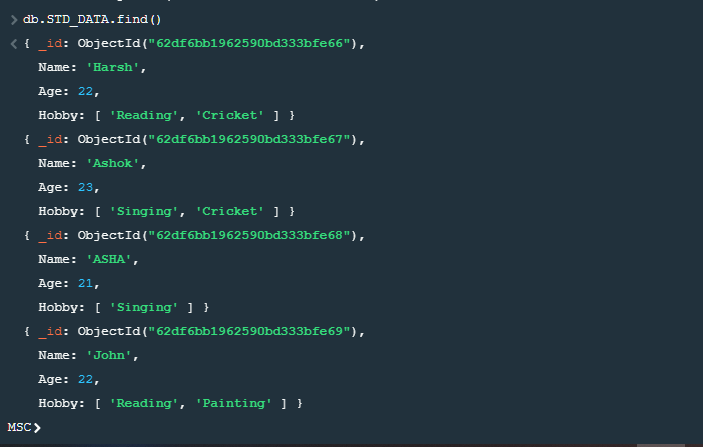
## Inserting Records into the collection STD\_DATA

db.STD\_DATA.insertMany([{Name: "Harsh", Age: 22, Hobby: ["Reading","Cricket"]},{Name: "Ashok", Age: 23, Hobby: ["Singing","Cricket"]},{Name: "ASHA", Age: 21, Hobby: ["Singing"]},{Name: "John", Age: 22, Hobby: ["Reading","Painting"]}])



## Displaying all the records

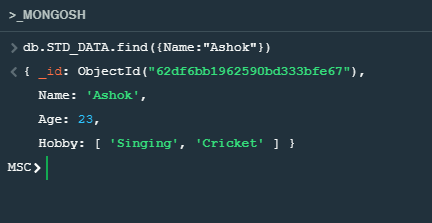
db.STD\_DATA.find()



**Q3. Display details of Ashok**

🡪

db.STD\_DATA.find({Name:"Ashok"})



**Q4. Update age of John to 20 keep rest of the data same**

🡪

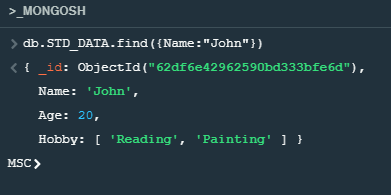
## Updating the record of John

db.STD\_DATA.findOneAndUpdate({Name:"John"},{$set :{Age:20}})



## Displaying the updated record for John

db.STD\_DATA.find({Name:"John"})

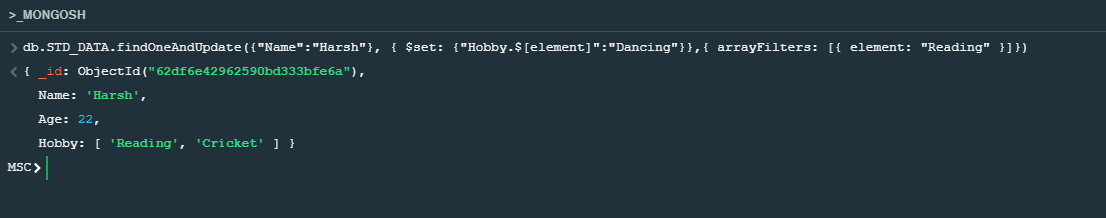


**Q5. Update hobby of Harsh as Dancing instead of Reading.**

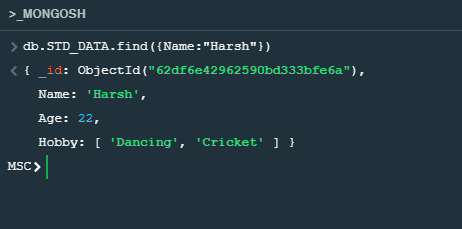
🡪

## Updating the record of Harsh

db.STD\_DATA.findOneAndUpdate({"Name":"Harsh"}, { $set: {"Hobby.$[element]":"Dancing"}},{ arrayFilters: [{ element: "Reading" }]})



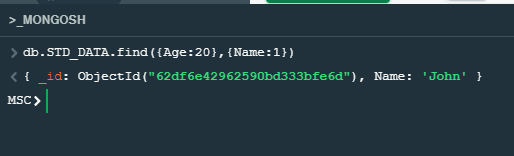
## Displaying Records of Harsh



**Q6. Display name whose age is 22.**

🡪

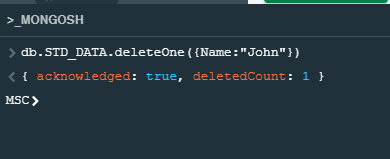
db.STD\_DATA.find({Age:20},{Name:1})



**Q7. Delete record of John.**

🡪

db.STD\_DATA.deleteOne({Name:"John"})



**Q8. Update Age of Ashok first occurrence as 19.**

🡪

## Updating Record

db.STD\_DATA.findOneAndUpdate({Name:"Ashok"},{$set :{Age:19}})

db.STD\_DATA.find({Name:"Ashok"})

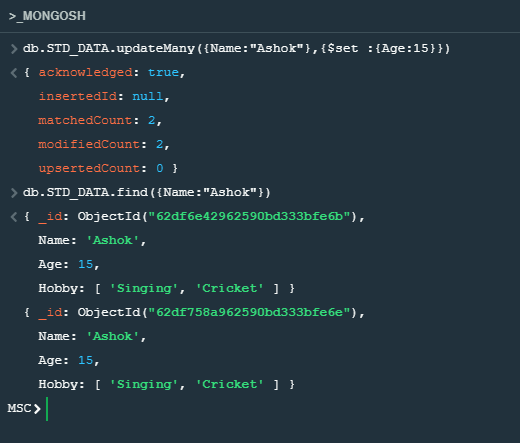


**Q9. Update Age of Ashok, all Occurrences as 15.**

🡪

db.STD\_DATA.updateMany({Name:"Ashok"},{$set :{Age:15}})

db.STD\_DATA.find({Name:"Ashok"})



**Q10. Add mobile no. of Harsh.**

🡪

db.STD\_DATA.updateMany({Name:"Harsh"},{$set :{Mobile:9372685907}})

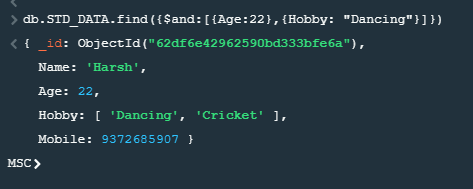
db.STD\_DATA.find({Name:"Harsh"})



**Q11. Display Record whose age is 22 and hobby as Dancing.**

**🡪**

db.STD\_DATA.find({$and:[{Age:22},{Hobby: "Dancing"}]})



**Q12. Display Record whose age is 15 or hobby as Singing**

🡪

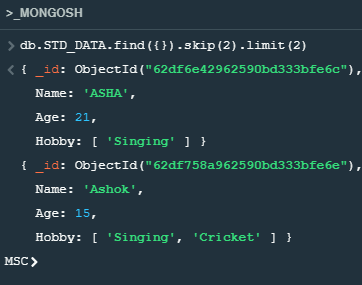
db.STD\_DATA.find({$or: [{Age:15},{ Hobby:["Singing"]}]})



**Q13. Display records from the collection by skipping first 2 records.**

🡪

db.STD\_DATA.find({}).skip(2).limit(2)



**Q14. Display records by sorting names.**

🡪

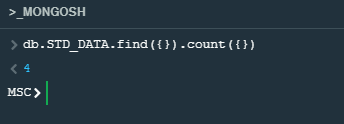
db.STD\_DATA.find({}).sort({name:-1})



**Q15. Count the number of records into the collection**

🡪

db.STD\_DATA.find({}).count({})



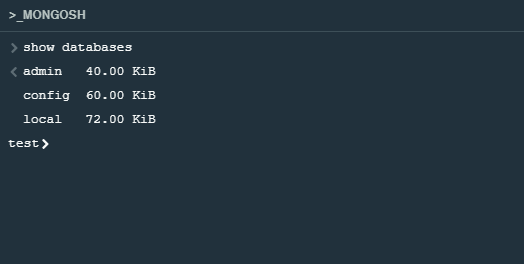
**Practical 2**

**Q1) Show existing Databases and create a new DB MSC.**

🡪

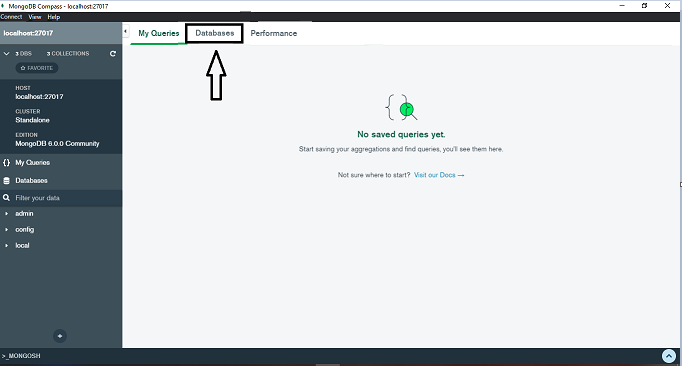
## Showing existing Databases

show databases

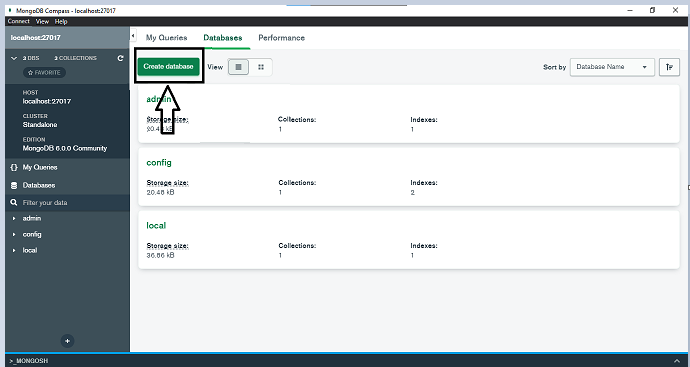


## Creating New Database

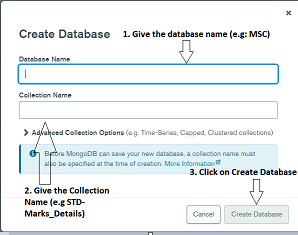
### **Step 1**: Click on the Databases Tab



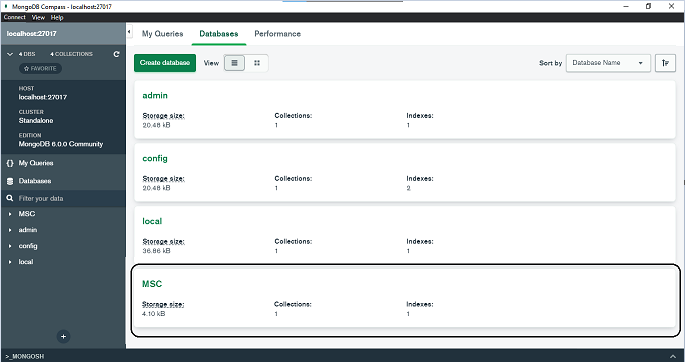
### **Step 2**: Click on Create Database



### **Step 3:** Assign the database name and the collection name and click on the **Create Database**

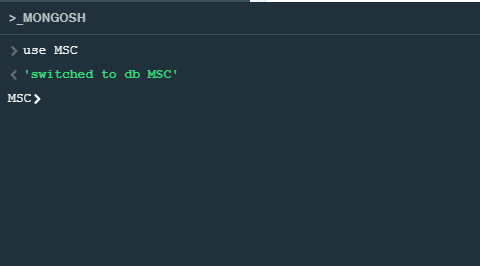


### **Step 4:** After the Create Database you will be able to see the database



## Switching database to MSC

use MSC



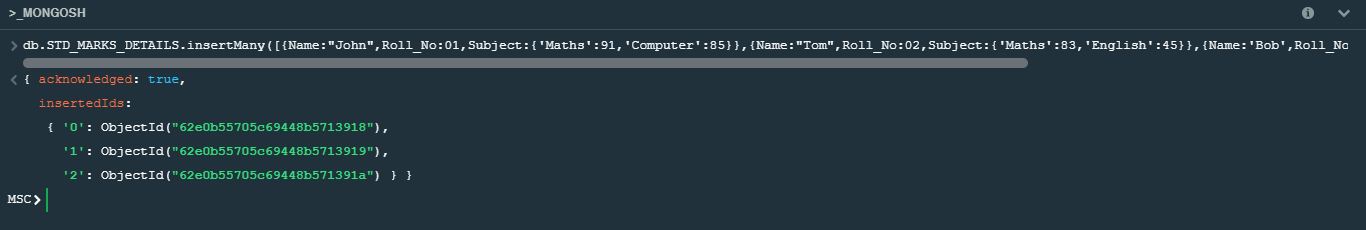
**Q2) Create a MongoDB containing marks of students for subjects like English, Maths and Computer.**

**Insert 3 documents where**

1. **First document is having marks of Maths and Computer**
2. **Second document is having marks of Maths and English**
3. **Third document is having marks of all subjects**

🡪

db.STD\_MARKS\_DETAILS.insertMany([{Name:"John",Roll\_No:01,Subject:{'Maths':91,'Computer':85}},{Name:"Tom",Roll\_No:02,Subject:{'Maths':83,'English':45}},{Name:'Bob',Roll\_No:03,Subject:{'Maths':76,'English':70,'Computer':85}}])

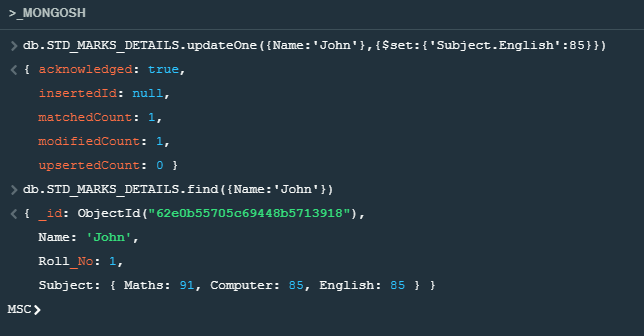


**Q3. Update marks of 1st document English as 85.**

🡪

db.STD\_MARKS\_DETAILS.updateOne({Name:'John'},{$set:{'Subject.English':85}})

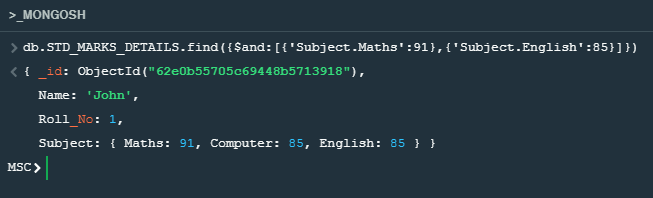
db.STD\_MARKS\_DETAILS.find({Name:'John'})



**Q4. Retrieve does contain marks of English as 85 and Maths as 91.**

**🡪**

db.STD\_MARKS\_DETAILS.find({$and:[{'Subject.Maths':91},{'Subject.English':85}]})

****

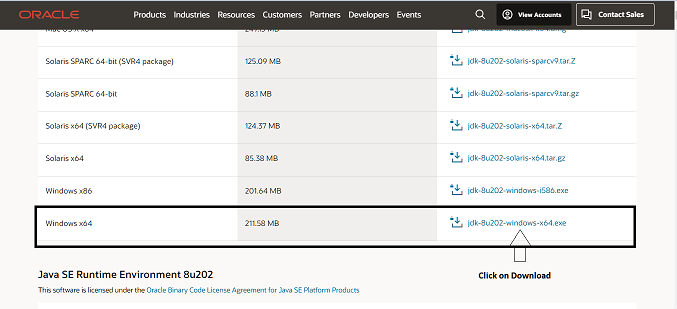
**Practical 3**

**Aim: Installation of Hadoop and java in windows.**

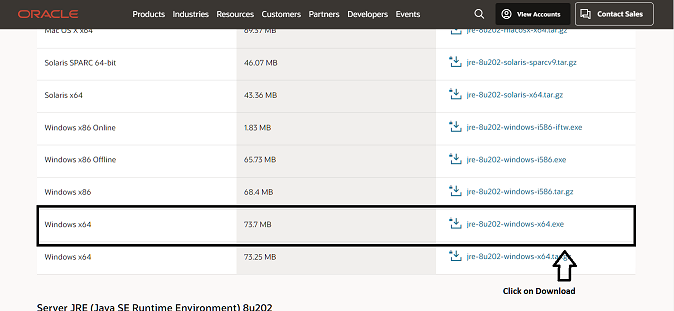
**🡪**

## Installation on java jdk and jre

### Step 1: Download SE Development Kit from the given link <https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html>



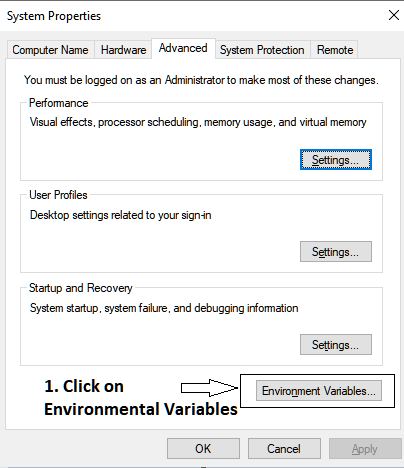
#### Step 2: Download Java SE Runtime Environment 8u202 from the given link <https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html>



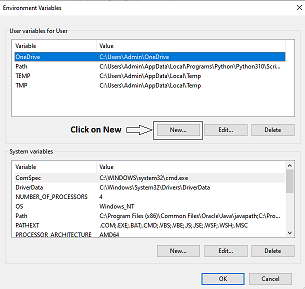
## Setting up the JDK in Environmental Variable

### Step 1: Start 🡪 Edit the system environment variables

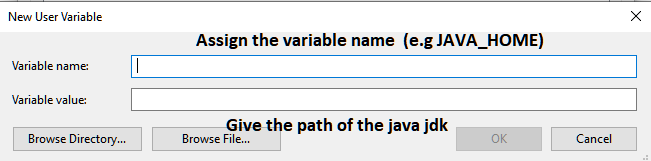
### Step 2: Click on the environment Variables

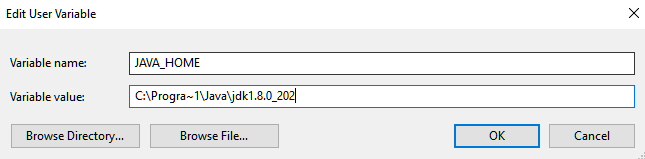


### Step 3: Create the new user variables

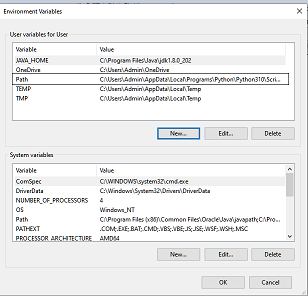


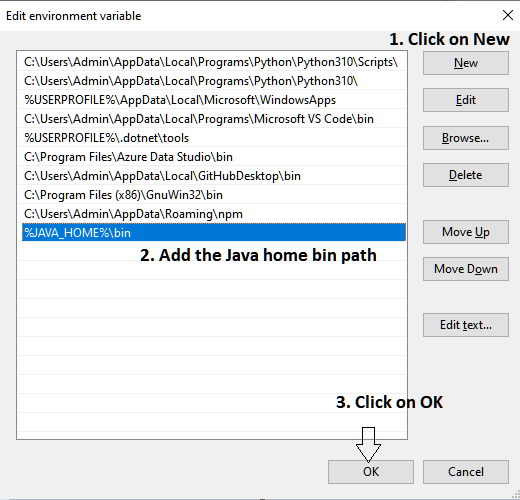
### Step 4: Give the Variable Name and java jdk path (NOTE: Rename the path C:\Program files\Java\jdk1.8.0\_202 to C:\Progra~1\Java\jdk1.8.0\_202)



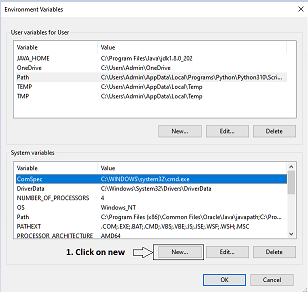


### Step 5: Add the Variable into the Path

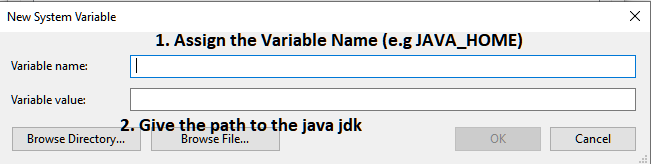


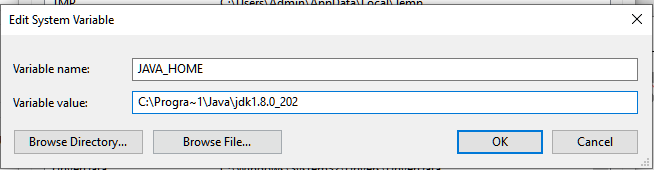


### Step 6: Create the new system variables

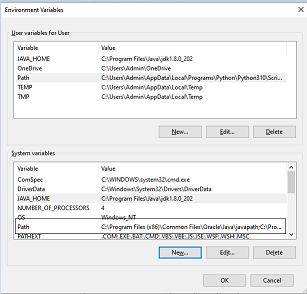


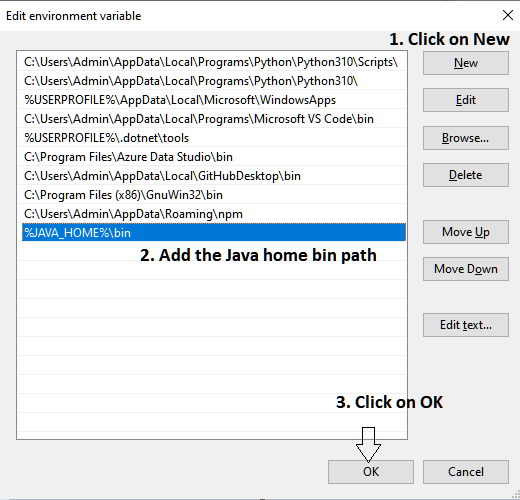
### Step 7: Give the Variable Name and java jdk path (NOTE: Rename the path C:\Program files\Java\jdk1.8.0\_202 to C:\Progra~1\Java\jdk1.8.0\_202)





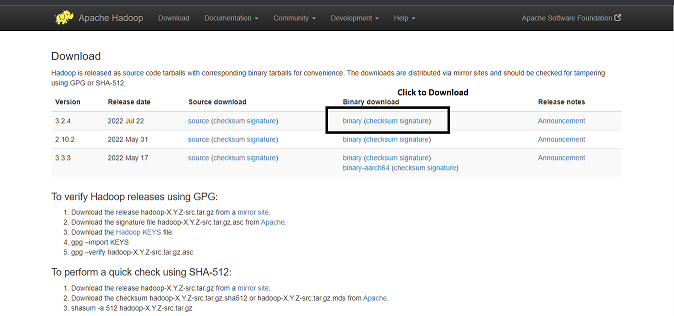
### Step 8: Add the Variable into the Path





## Installation of Hadoop

### Step 1: Download Apache Hadoop from the given link and click on the binary <https://hadoop.apache.org/releases.html>



### Step 2: Click the link to download the zip

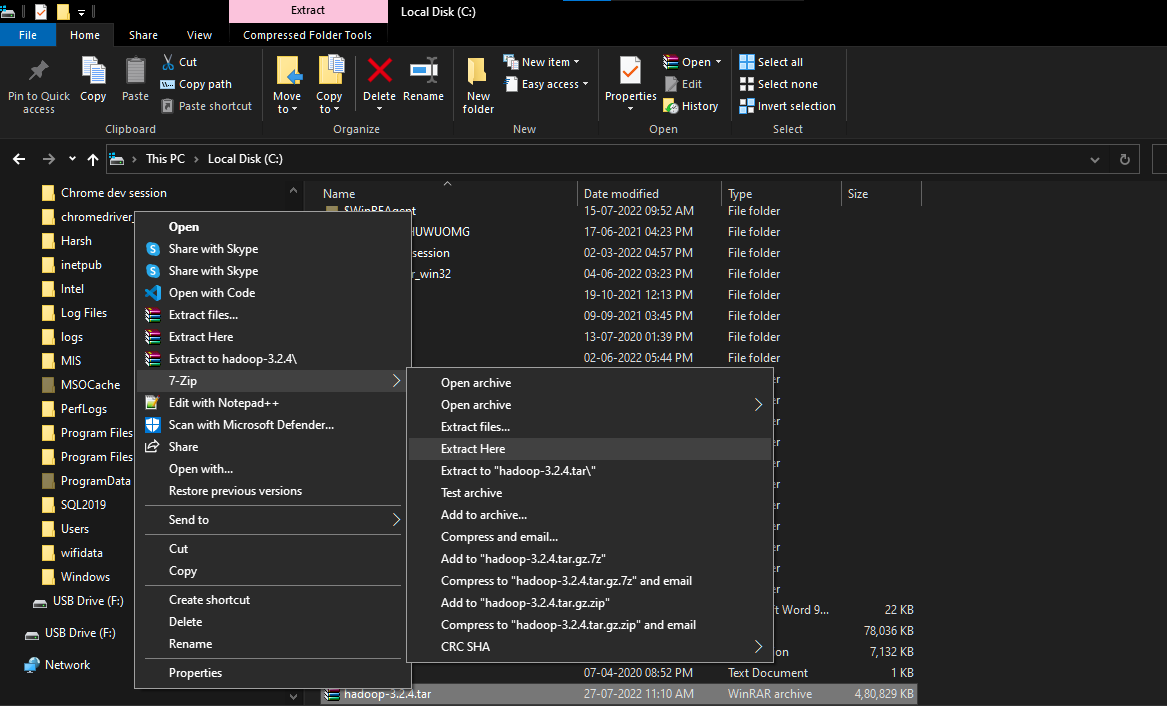


## Installation of the 7zip

### Step 1: Download the 7zip and install into the system <https://www.7-zip.org/download.html>

## Extracting Hadoop zip using 7Zip

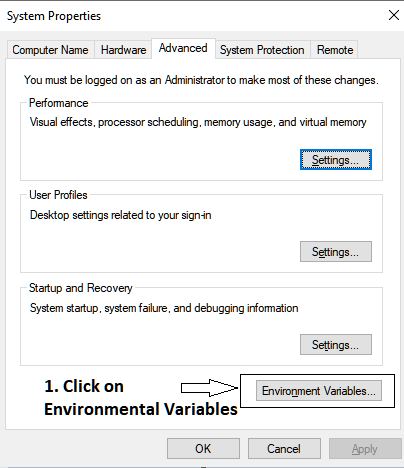
### Step 1 : Right Click on the zip 🡪 Click on 7-zip 🡪 Extract Here



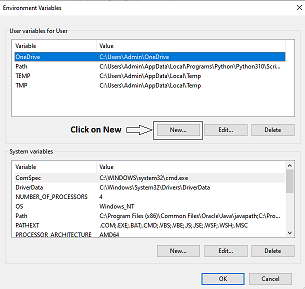
## Setting up the Hadoop in Environmental Variable

### Step 1: Start 🡪 Edit the system environment variables

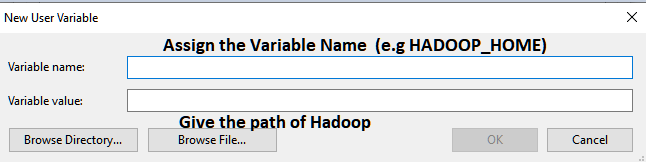
### Step 2: Click on the environment Variables

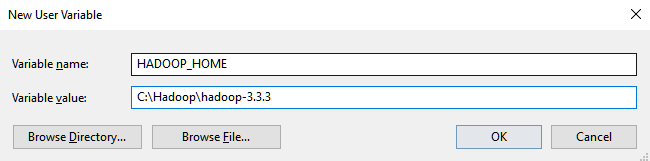


### Step 3: Create the new user variables

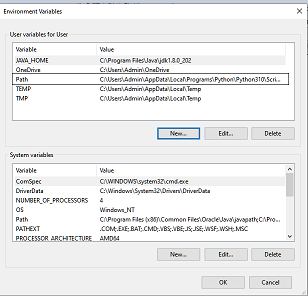


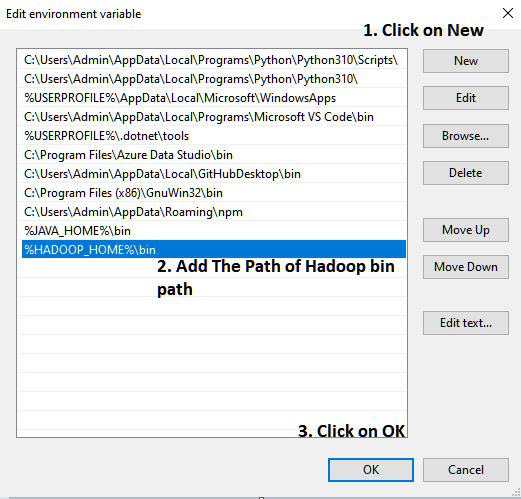
### Step 4: Give the Variable Name and java Hadoop path



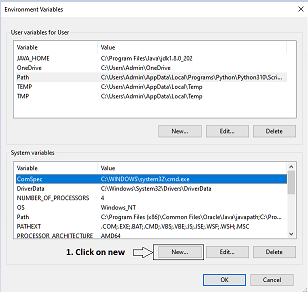


### Step 5: Add the Variable into the Path

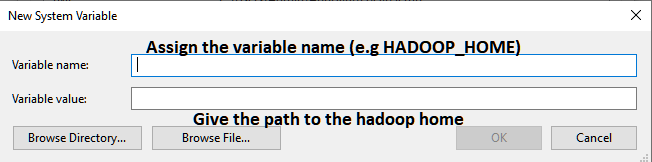


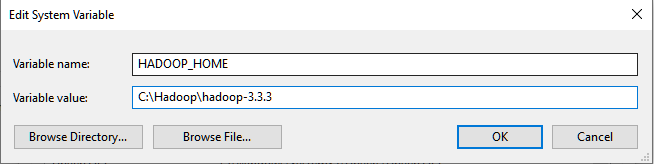


### Step 6: Create the new system variables

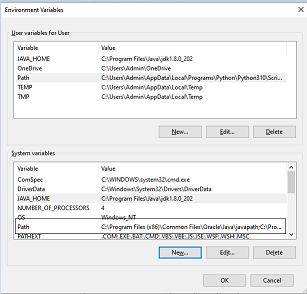


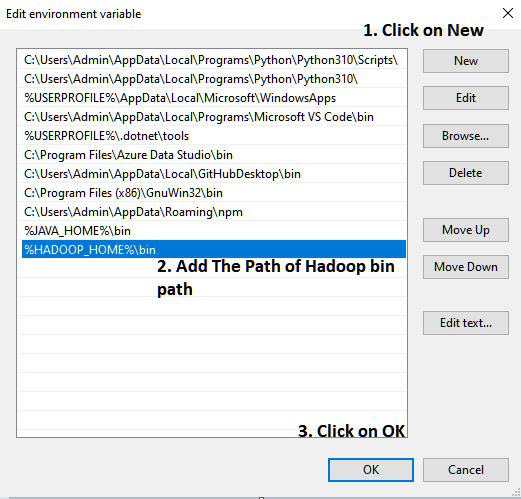
### Step 7: Give the Variable Name and java Hadoop path





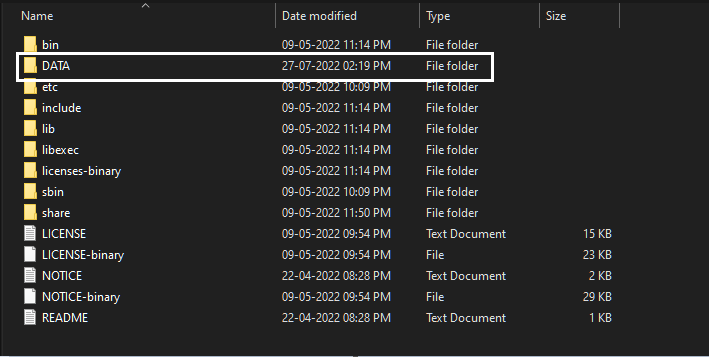
### Step 8: Add the Variable into the Path



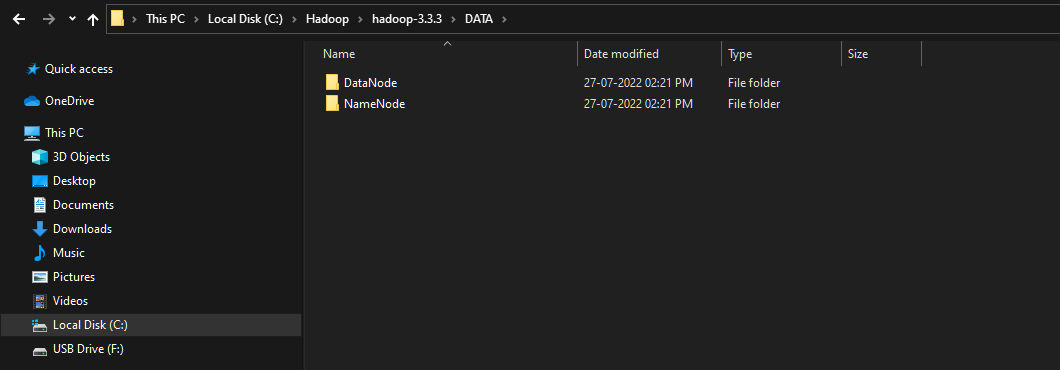


## Setting up Hadoop

### Step 1: Create the new Folder with the name DATA



### Step 2: Go inside the data folder and create 2 folder i.e. NameNode and DataNode



### Step 3: open the file **hdfs-site.xml** C:\Hadoop\hadoop-3.3.3\etc\hadoop

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>

  <property>

    <name>dfs.replication</name>

    <value>3</value>

  </property>

  <property>

<name>dfs.namenode.name.dir</name>

<value>C:\Hadoop\hadoop-3.3.3\DATA\NameNode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>C:\Hadoop\hadoop-3.3.3\DATA\DataNode</value>

</property>

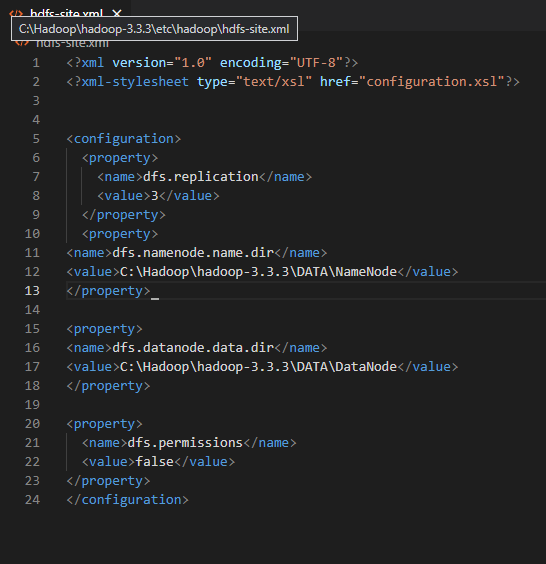
<property>

  <name>dfs.permissions</name>

  <value>false</value>

</property>

</configuration>



### Step 4: open the file **core-site.xml** C:\Hadoop\hadoop-3.3.3\etc\hadoop

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!--

  Licensed under the Apache License, Version 2.0 (the "License");

  you may not use this file except in compliance with the License.

  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>

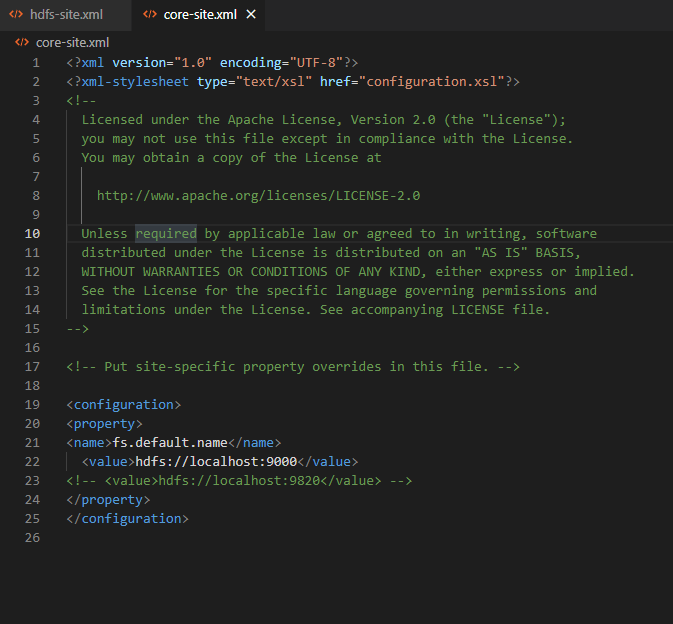
<name>fs.default.name</name>

  <value>hdfs://localhost:9000</value>

<!-- <value>hdfs://localhost:9820</value> -->

</property>

</configuration>



### Step 5: open the file **mapred-site.xml** C:\Hadoop\hadoop-3.3.3\etc\hadoop

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!--

  Licensed under the Apache License, Version 2.0 (the "License");

  you may not use this file except in compliance with the License.

  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>mapreduce.framework.name</name>

<value>yarn</value>

</property>

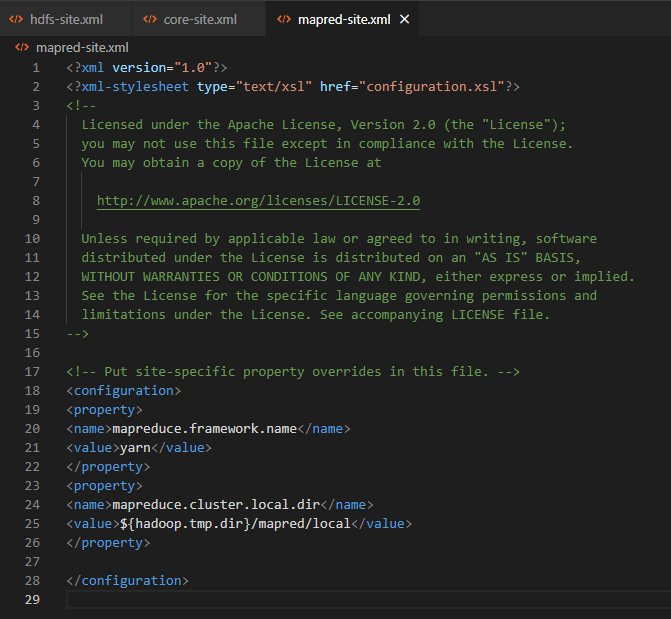
<property>

<name>mapreduce.cluster.local.dir</name>

<value>${hadoop.tmp.dir}/mapred/local</value>

</property>

</configuration>



### Step 6: open the file **yarn-site -site.xml** C:\Hadoop\hadoop-3.3.3\etc\hadoop

<?xml version="1.0"?>

<!--

  Licensed under the Apache License, Version 2.0 (the "License");

  you may not use this file except in compliance with the License.

  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.

-->

<configuration>

<!-- Site specific YARN configuration properties -->

<configuration>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

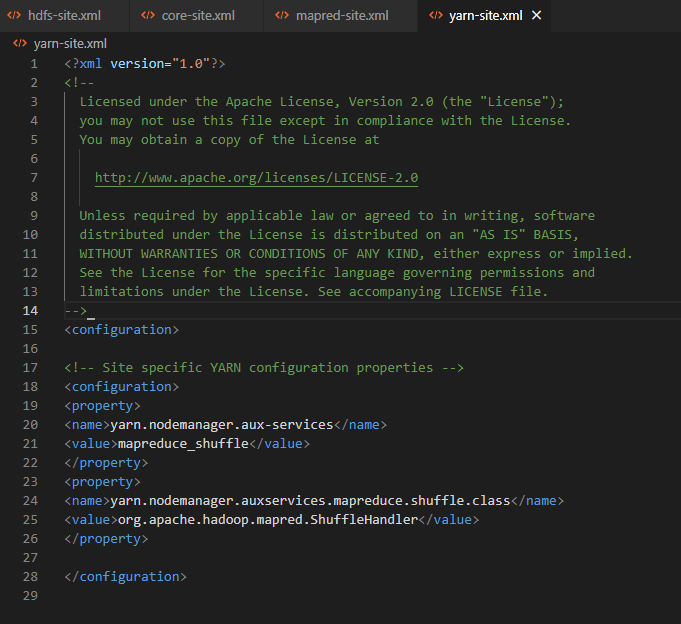
<property>

<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>

<value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

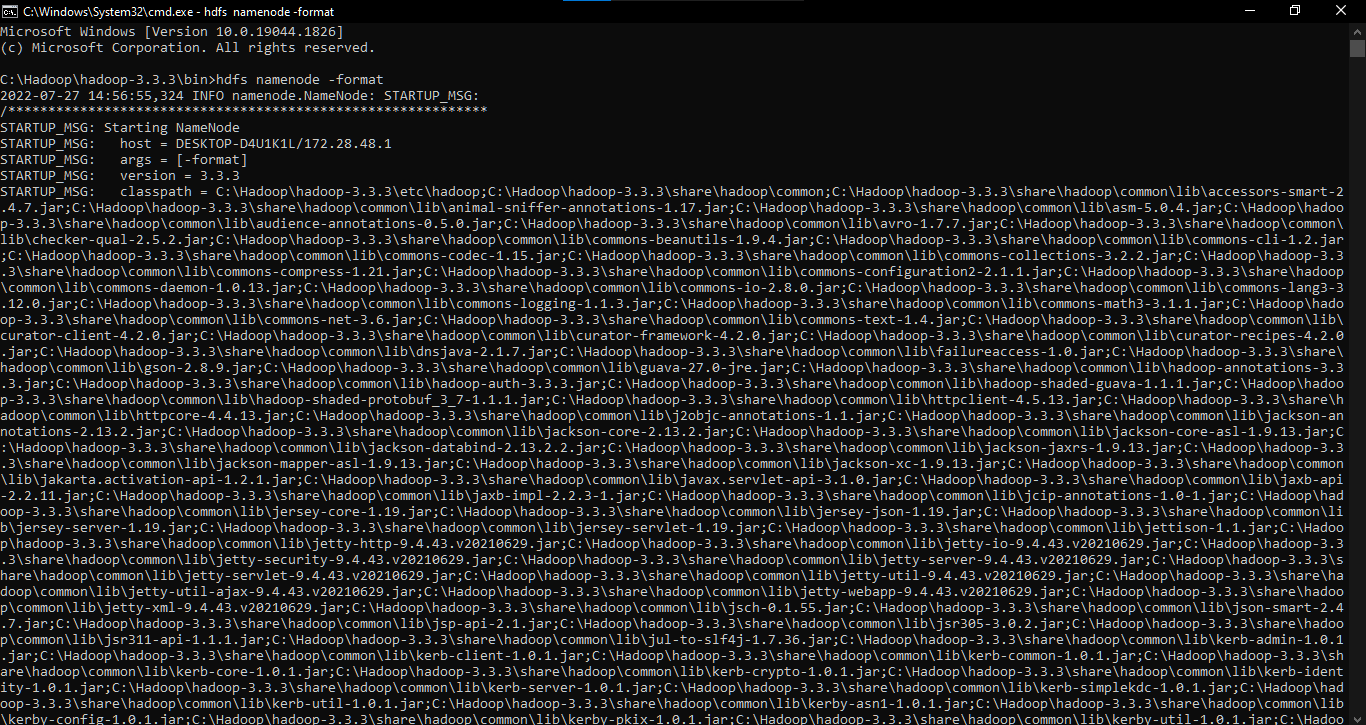
</configuration>

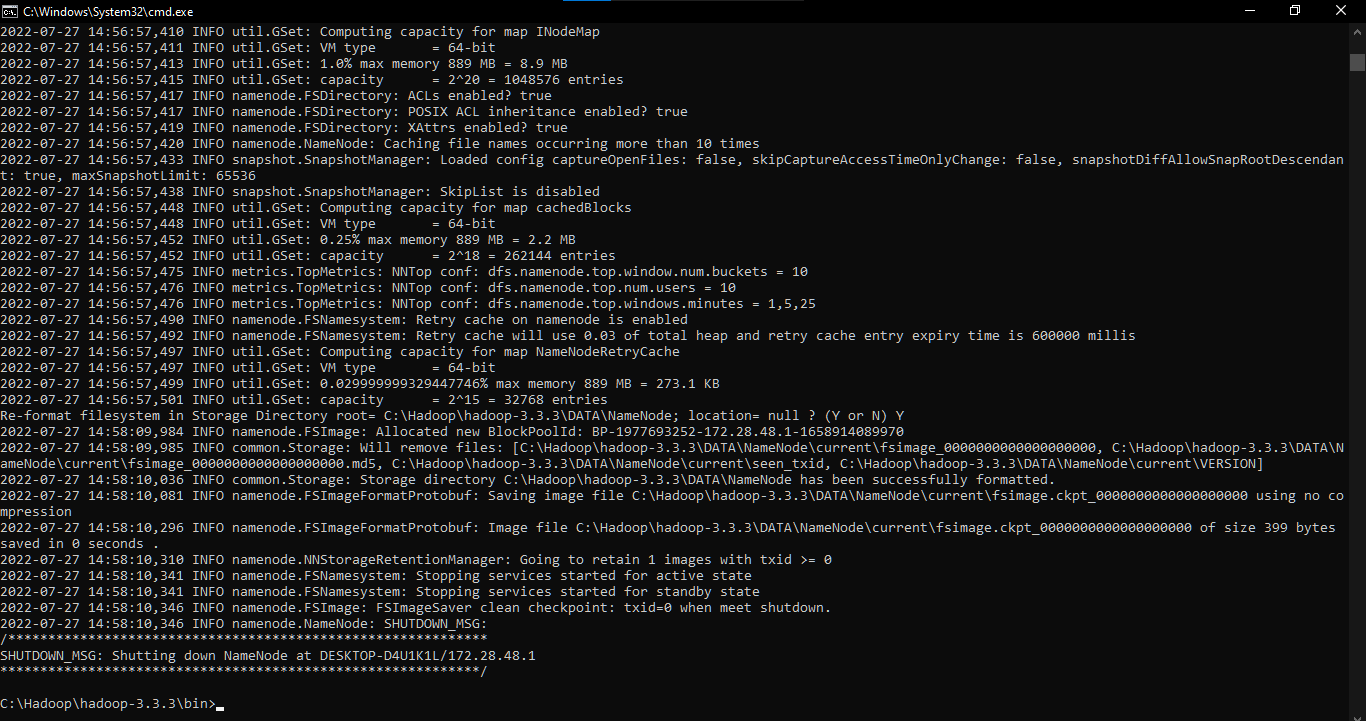


### Step 7: Go to GitHub <https://github.com/Selfgrowth/Apache-hadoop-3.1.1-winutils> and download the bin folder and replace all the files with the C:\Hadoop\hadoop-3.3.3\bin

### Step 8: Open the command prompt and change the location to C:\Hadoop\hadoop-3.3.3\bin

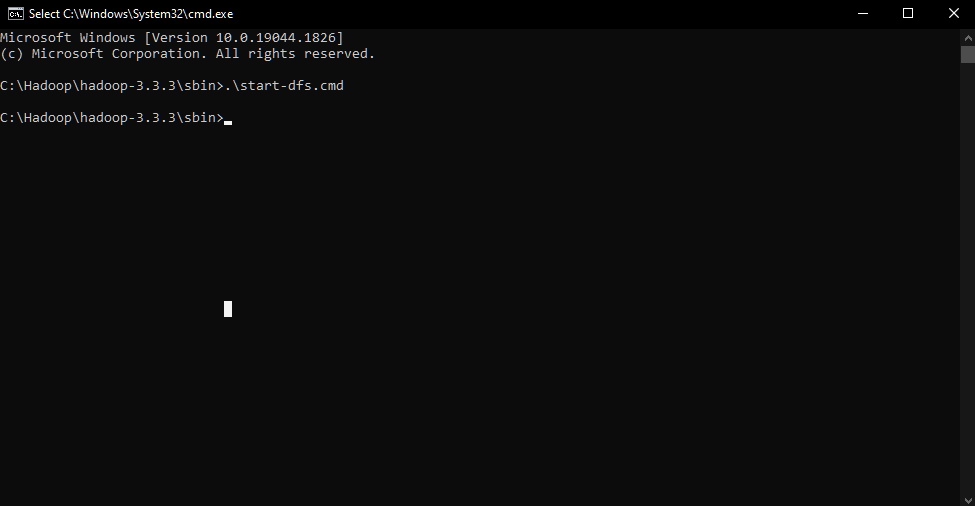
### Step 9: hdfs namenode -format



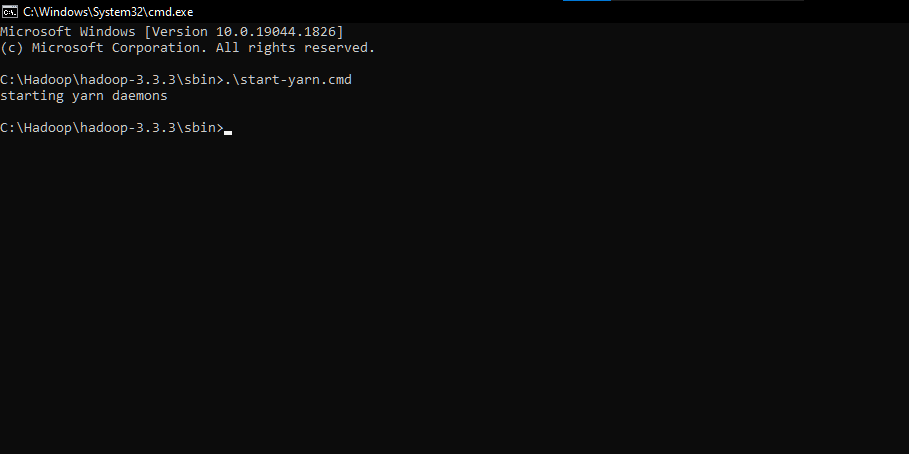


### Step 10: Go to sbin folder. Type cmd in the address bar and 2 different open command prompt

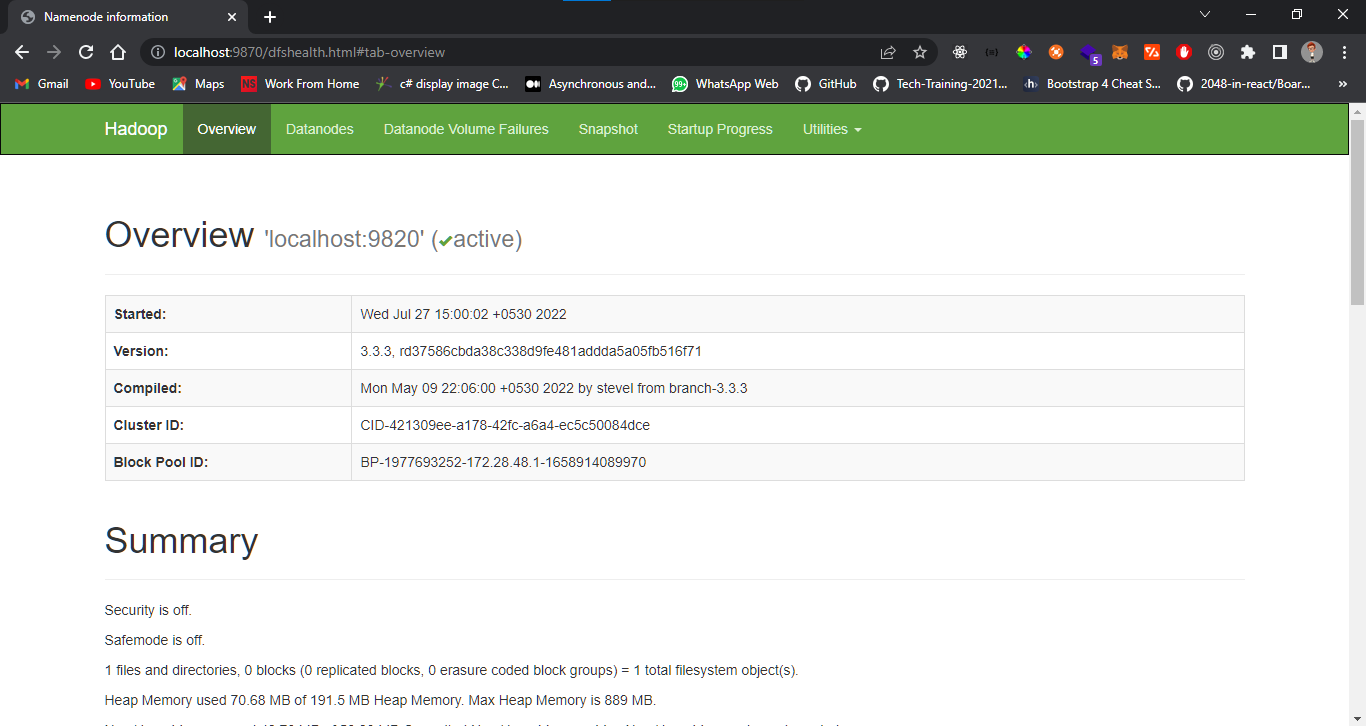
First cmd: **.\start-dfs.cmd**



Second cmd: **.\start-yarn.cmd**



### Step 11: now open chrome or any other browser and type <http://localhost:9870/>



**Practical 4**

**Aim: Write an Hadoop MapReduce Program in Python**

**🡪**

## Create the mapper.py

*#!/usr/bin/env python*

"""mapper.py"""

*import* sys

*# input comes from STDIN (standard input)*

*for* line in sys.stdin:

*# remove leading and trailing whitespace*

    line = line.strip()

*# split the line into words*

    words = line.split()

*# increase counters*

*for* word in words:

*# write the results to STDOUT (standard output);*

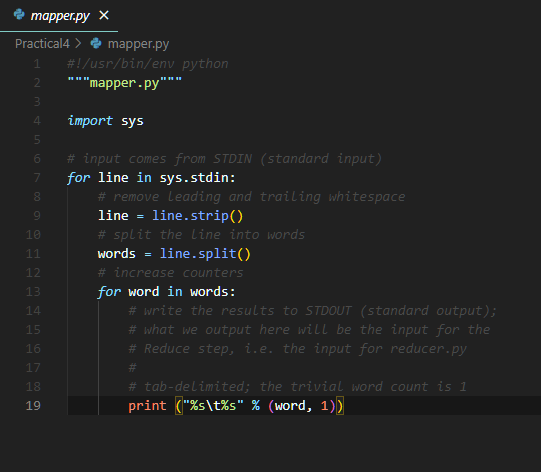
*# what we output here will be the input for the*

*# Reduce step, i.e. the input for reducer.py*

*#*

*# tab-delimited; the trivial word count is 1*

        print ("%s\t%s" % (word, 1))



## Create the reducer.py

*#!/usr/bin/env python*

"""reducer.py"""

*from* operator *import* itemgetter

*import* sys

current\_word = None

current\_count = 0

word = 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*

    word, count = line.split('\t', 1)

*# convert count (currently a string) to int*

*try*:

        count = int(count)

*except* ValueError:

*# count was not a number, so silently*

*# ignore/discard this line*

*continue*

*# this IF-switch only works because Hadoop sorts map output*

*# by key (here: word) before it is passed to the reducer*

*if* current\_word == word:

        current\_count += count

*else*:

*if* current\_word:

*# write result to STDOUT*

            print ("%s\t%s" % (current\_word, current\_count))

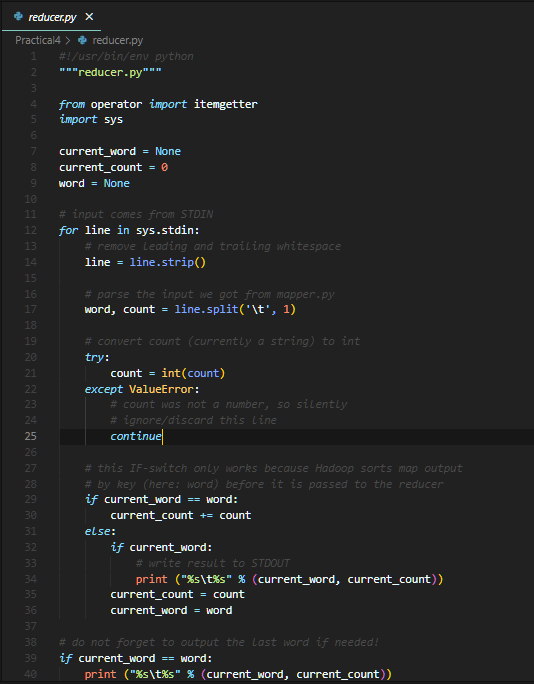
        current\_count = count

        current\_word = word

*# do not forget to output the last word if needed!*

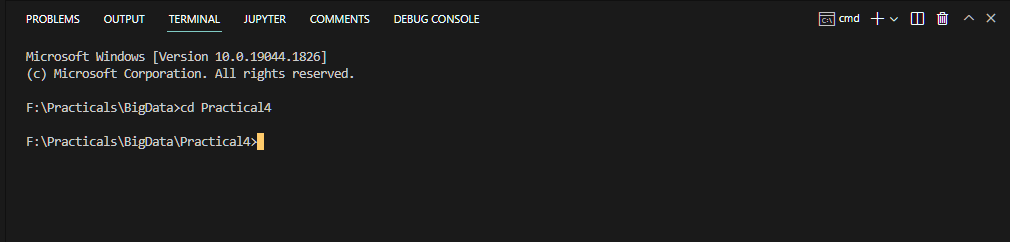
*if* current\_word == word:

    print ("%s\t%s" % (current\_word, current\_count))

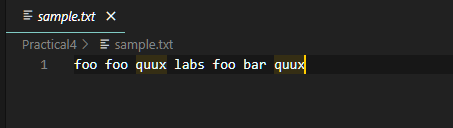


## Running mapper and reducer without Hadoop HDFS

### Step 1: Open **Command Prompt** where the mapper.py and reducer.py is located

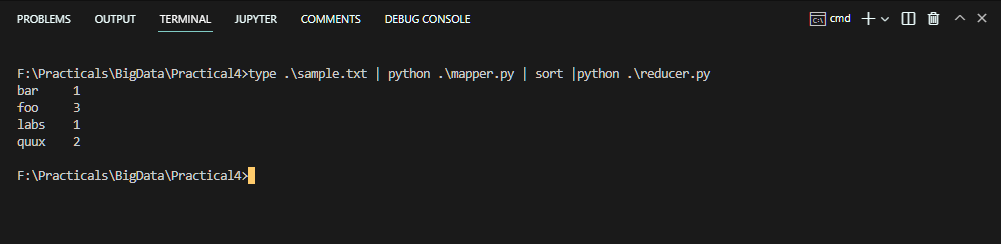


### Step 2: To Execute the program create one file in the same location with name sample.txt



### Step 3: Now run the following command to get the output

type .\sample.txt | python .\mapper.py | sort |python .\reducer.py



## Running the Python Code on Hadoop

### Step 1:Download example input data

We will use three eBooks from Project Gutenberg for this example:

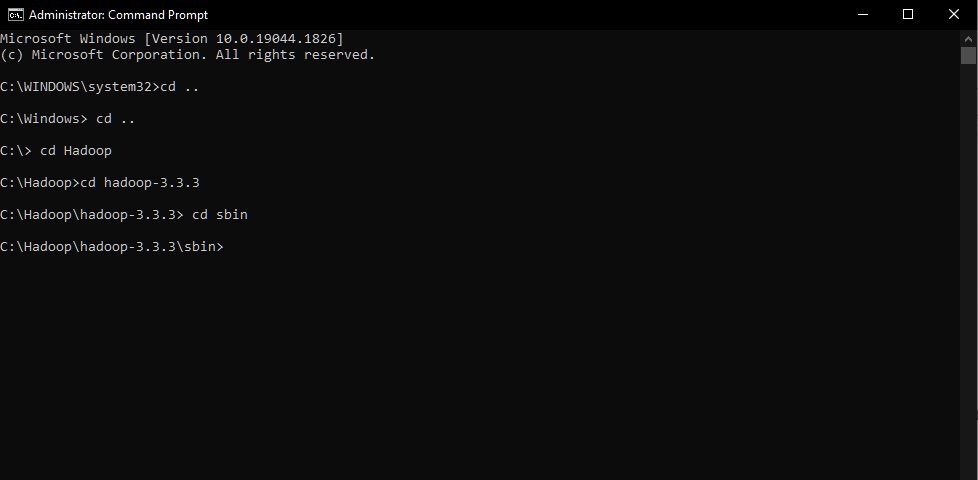
* [The Outline of Science, Vol. 1 (of 4) by J. Arthur Thomson](http://www.gutenberg.org/etext/20417)
* [The Notebooks of Leonardo Da Vinci](http://www.gutenberg.org/etext/5000)
* [Ulysses by James Joyce](http://www.gutenberg.org/etext/4300)

Download each eBook as text files in Plain Text UTF-8 encoding and store the files in a local temporary directory of choice.

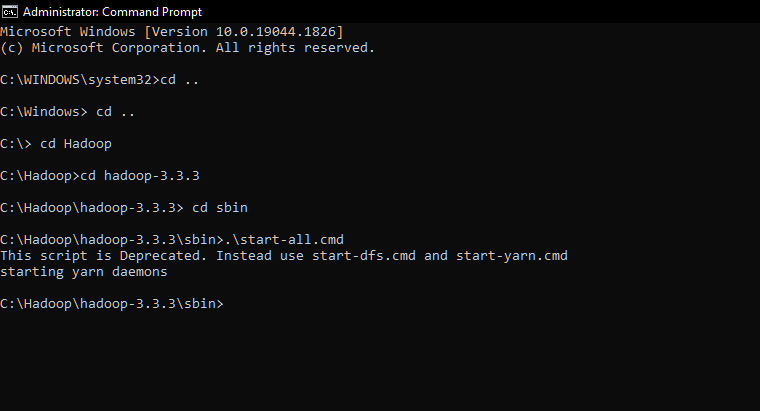
## Copy local example data to HDFS

Before we run the actual MapReduce job, we must first copy the files from our local file system to Hadoop’s HDFS.

### Step 1: Open Command Prompt in Administration Mode and change the present working directory to the **C:\Hadoop\hadoop-3.3.3\sbin**



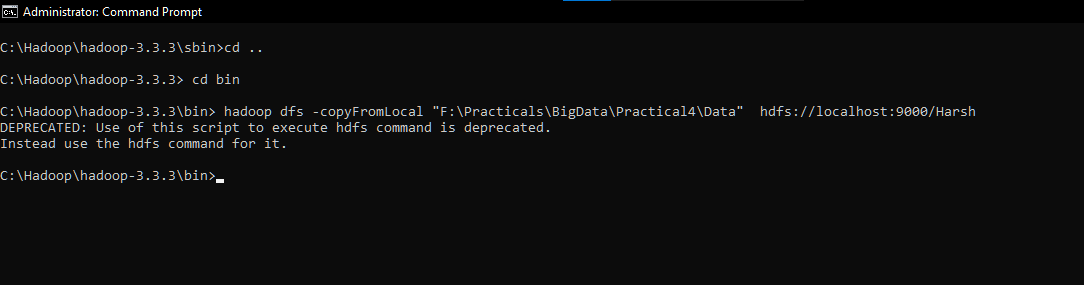
### Step 2: Now run the command **.\start-all.cmd**



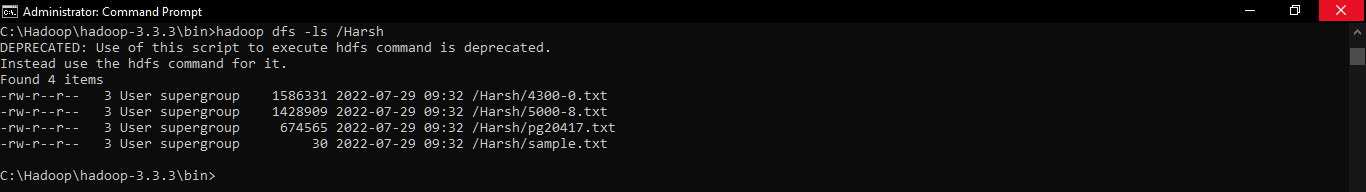
### Step3: Now change the present working directory to **C:\Hadoop\hadoop-3.3.3\bin** and run the command

#### hadoop dfs –copyFromLocal ‘path of the downloaded sample file’ ‘path to store on the hdfs’

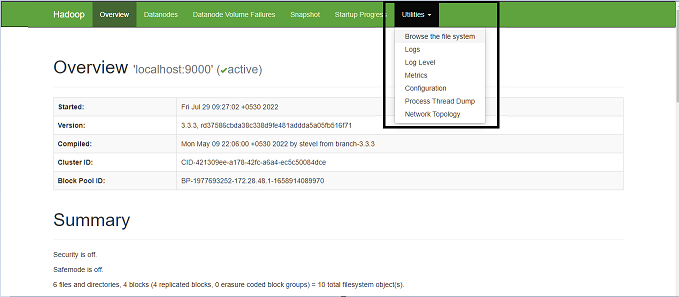
 hadoop dfs -copyFromLocal "F:\Practicals\BigData\Practical4\Data"  hdfs://localhost:9000/Harsh

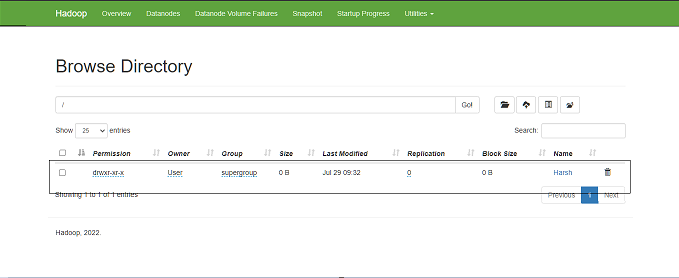


hadoop dfs -ls /Harsh



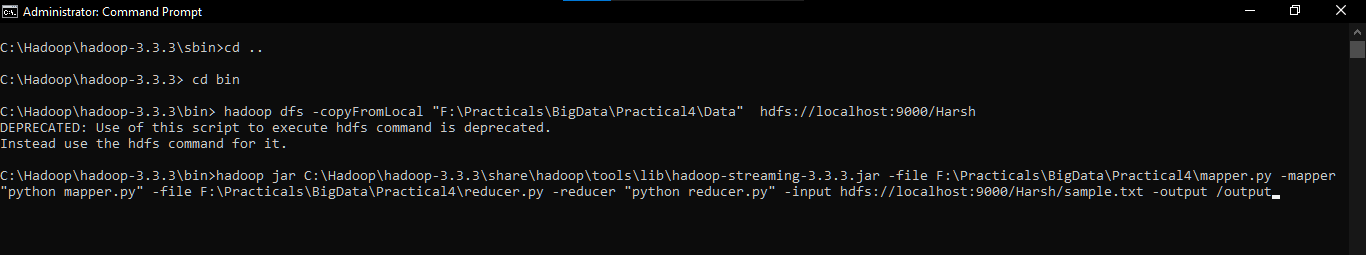
### Step 4: To check the files are uploaded to the Hadoop HDFS the visit <http://localhost:9870/dfshealth.html#tab-overview> 🡪 go to utilities in the navigation bar and click on the Browse the file system

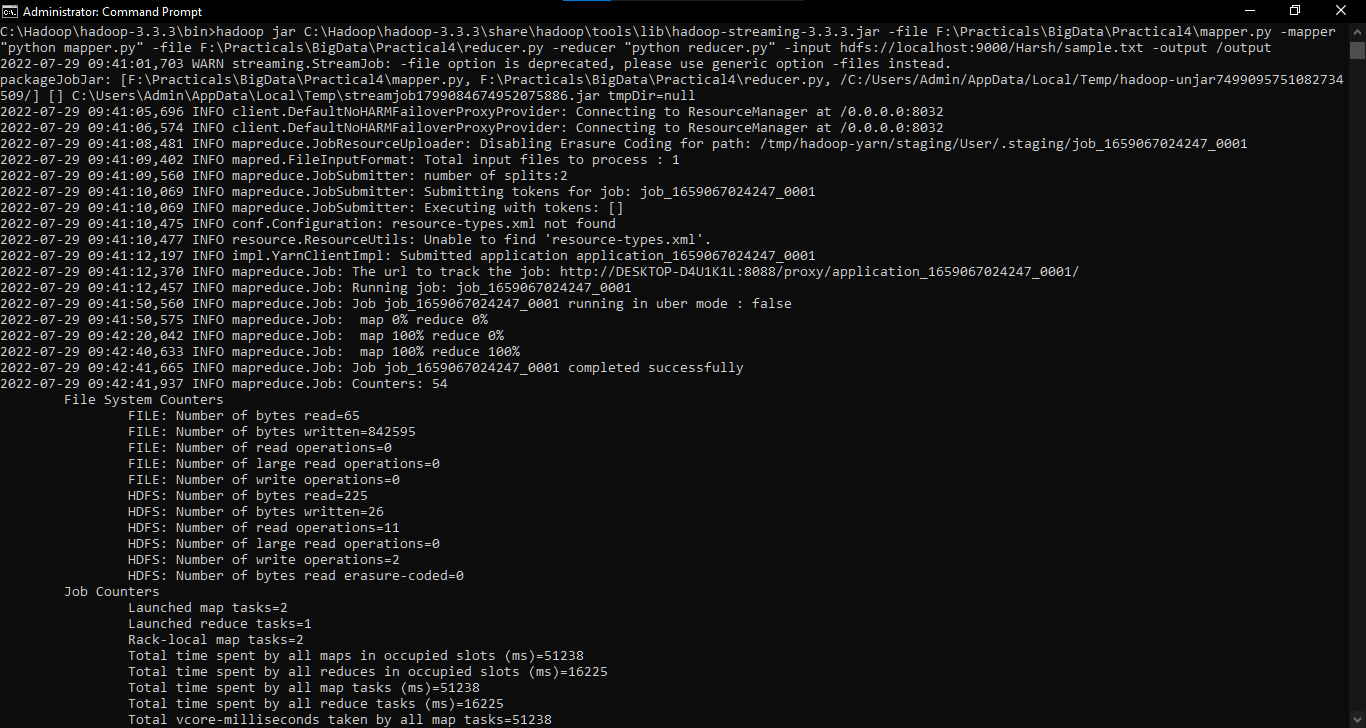


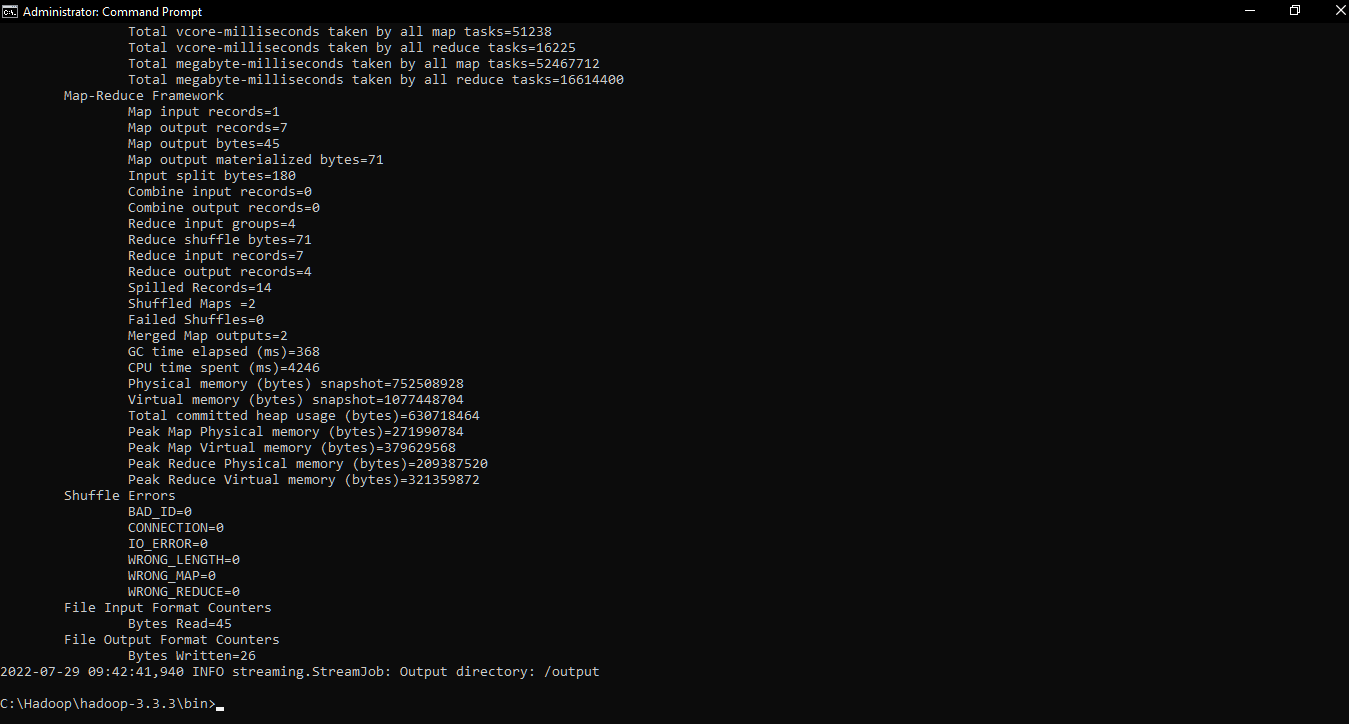


### Step 5: Run the MapReduce job

hadoop jar C:\Hadoop\hadoop-3.3.3\share\hadoop\tools\lib\hadoop-streaming-3.3.3.jar -file F:\Practicals\BigData\Practical4\mapper.py -mapper "python mapper.py" -file F:\Practicals\BigData\Practical4\reducer.py -reducer "python reducer.py" -input hdfs://localhost:9000/Harsh/sample.txt -output /output

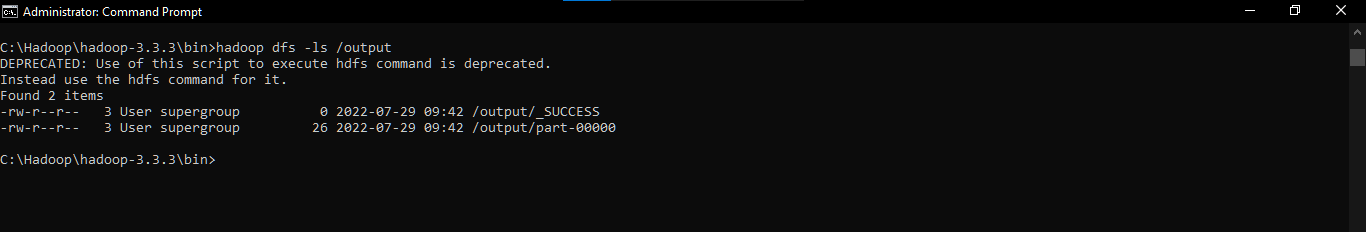




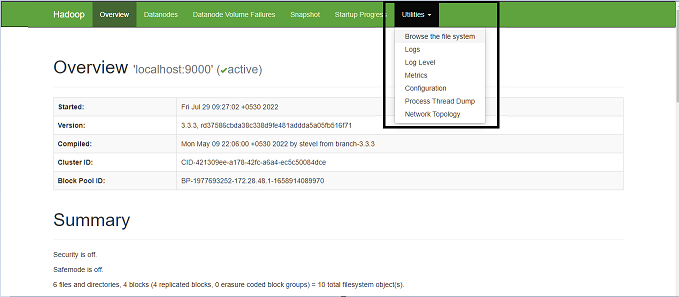


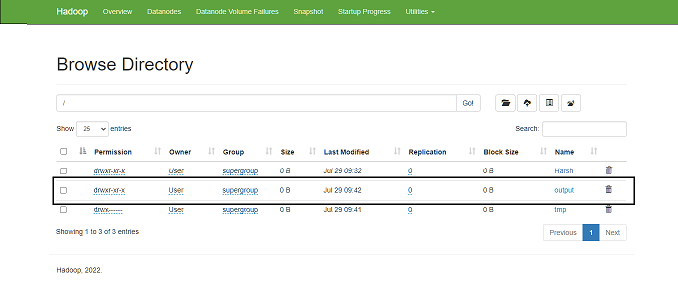
### Step 6: Check if the result is successfully stored in HDFS directory /output

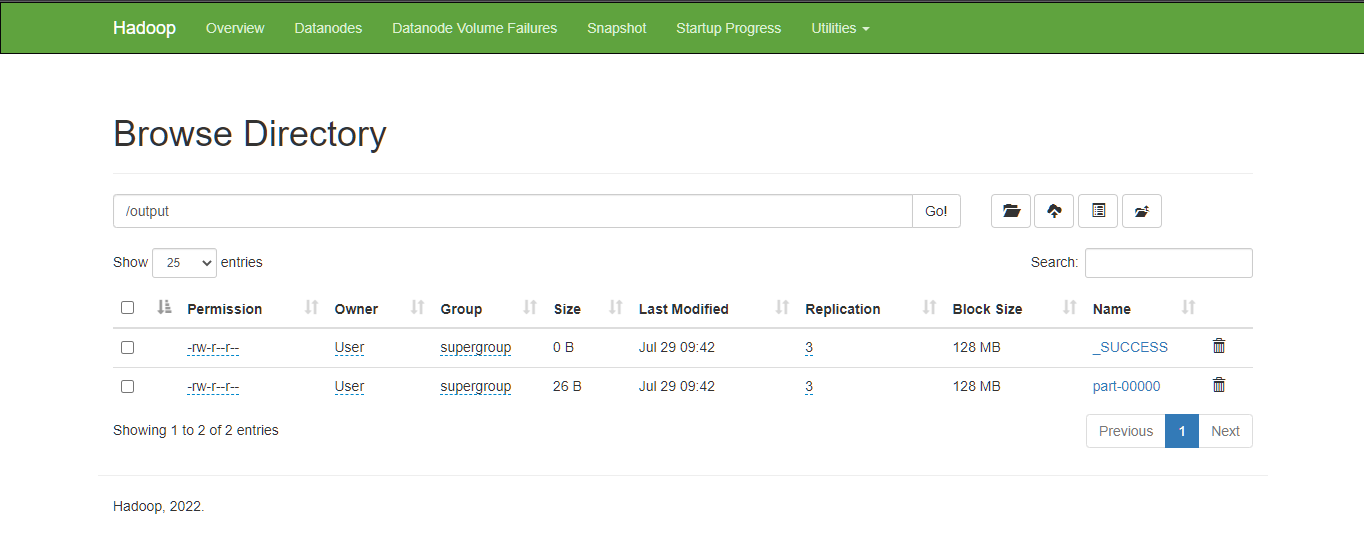
hadoop dfs -ls /output



### Step 7: To check the output is generated to the Hadoop HDFS the visit <http://localhost:9870/dfshealth.html#tab-overview> 🡪 go to utilities in the navigation bar and click on the Browse the file system







### Step 8: You can then inspect the contents of the file with the dfs -cat command:

hadoop fs -cat /output/part-00000

