

PART 1:

Hadoop Installation

Fathima Syeda

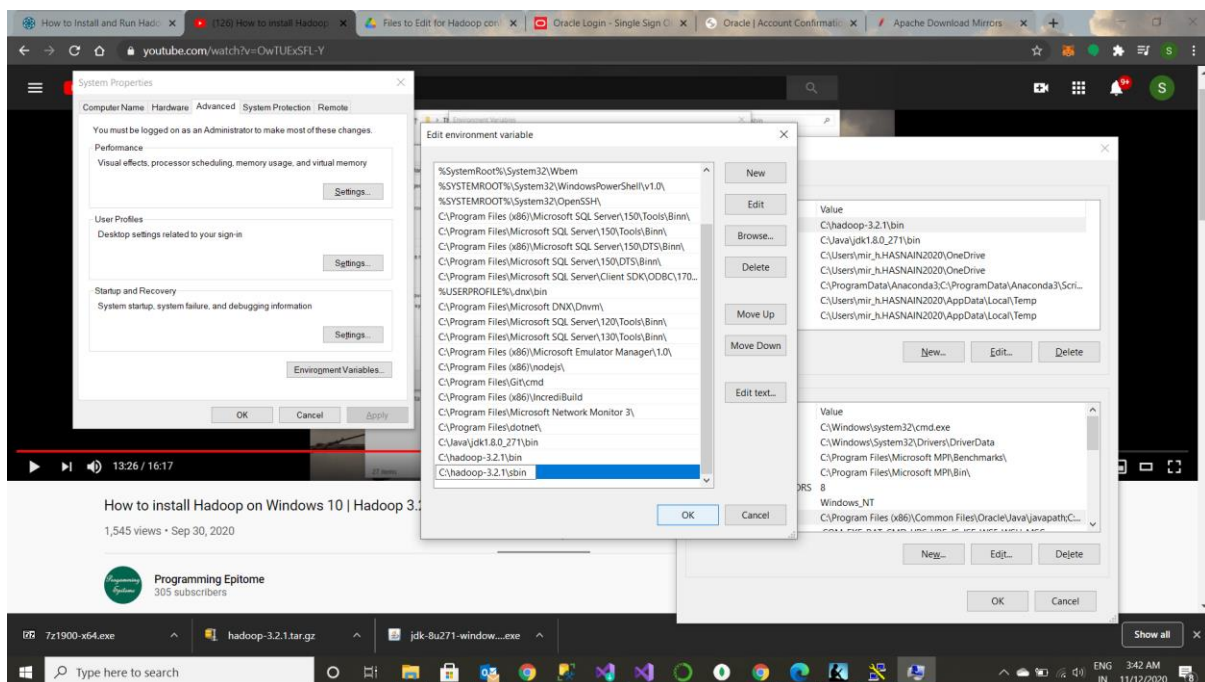
Platform Setup:

Hadoop 3.2.1 is installed in this lab, but in order to do that we must have JAVA SDK-8 installed on our system , so we install that first.

Hadoop would be setup as a single node system in pseudo-distributed mode

We download Hadoop from- <https://hadoop.apache.org/releases.html>

Once Hadoop is installed , we must set the path and environment variables for it as:



After that 5 configuration files in the C:\hadoop-3.2.1\etc\hadoop folder, viz core-site.xml ,mapred-site.xml, yarn -site.xml , hdfs-site.xml and hadoop-env.cmd are edited.

a) File C:/Hadoop-3.2.1/etc/hadoop/core-site.xml, paste below xml paragraph and save this file.

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

b) C:/Hadoop-3.2.1/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>
```

c) Create folder "data" under "C:\Hadoop-3.2.1"

1) Create folder "datanode" under "C:\Hadoop-3.2.1\data"

2) Create folder "namenode" under "C:\Hadoop-3.2.1\data" data

d) Edit file C:\Hadoop-3.2.1/etc/hadoop/hdfs-site.xml, paste below xml paragraph and save this file.

```
<configuration>
  <property>
<name>dfs.replication</name>
  <value>1</value>
  </property>
  <property>
<name>dfs.namenode.name.dir</name>
<value>C:\hadoop-3.2.1\data\namenode</value>
  </property>
  <property>
<name>dfs.datanode.data.dir</name>
<value>C:\hadoop-3.2.1\data\datanode</value>
  </property>
</configuration>
```

e) Edit file C:\Hadoop-3.2.1/etc/hadoop/yarn-site.xml, paste below xml paragraph and save this file.

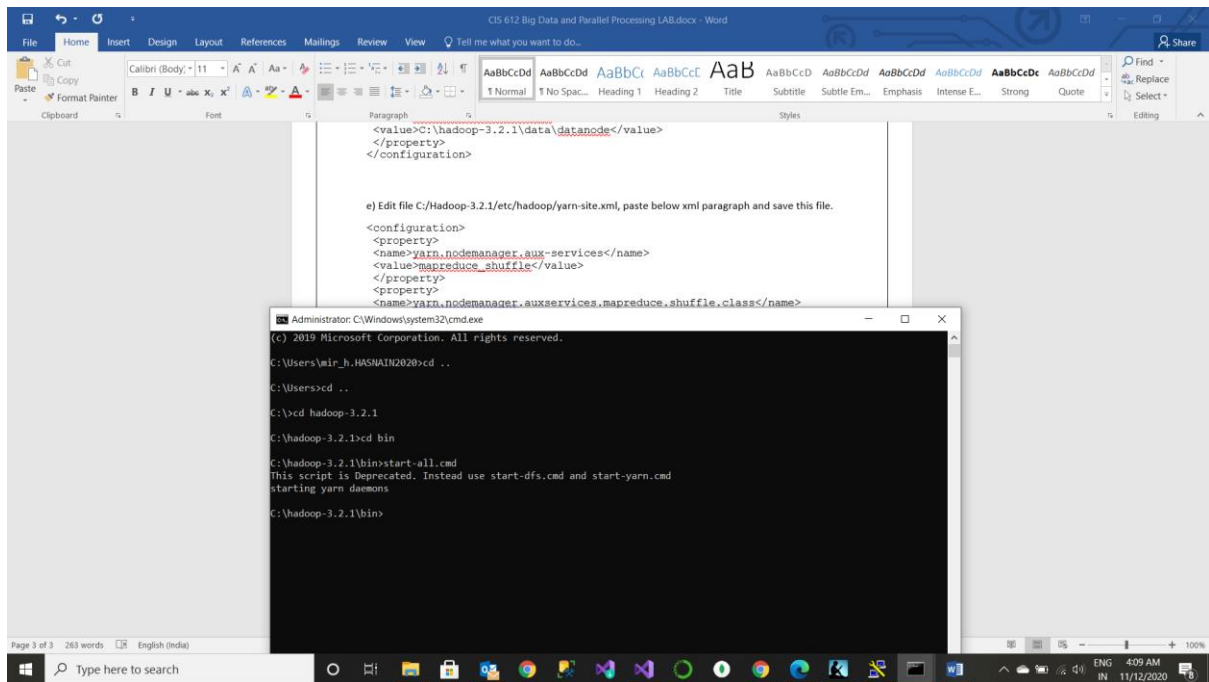
```
<configuration>
  <property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
  </property>
  <property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
</configuration>
```

f) save the java path in the hadoop-env.cmd file as the path of the java sdk's bin folder.

Once the configurations files have been edited and saved, the Hadoop configuration is completed successfully.

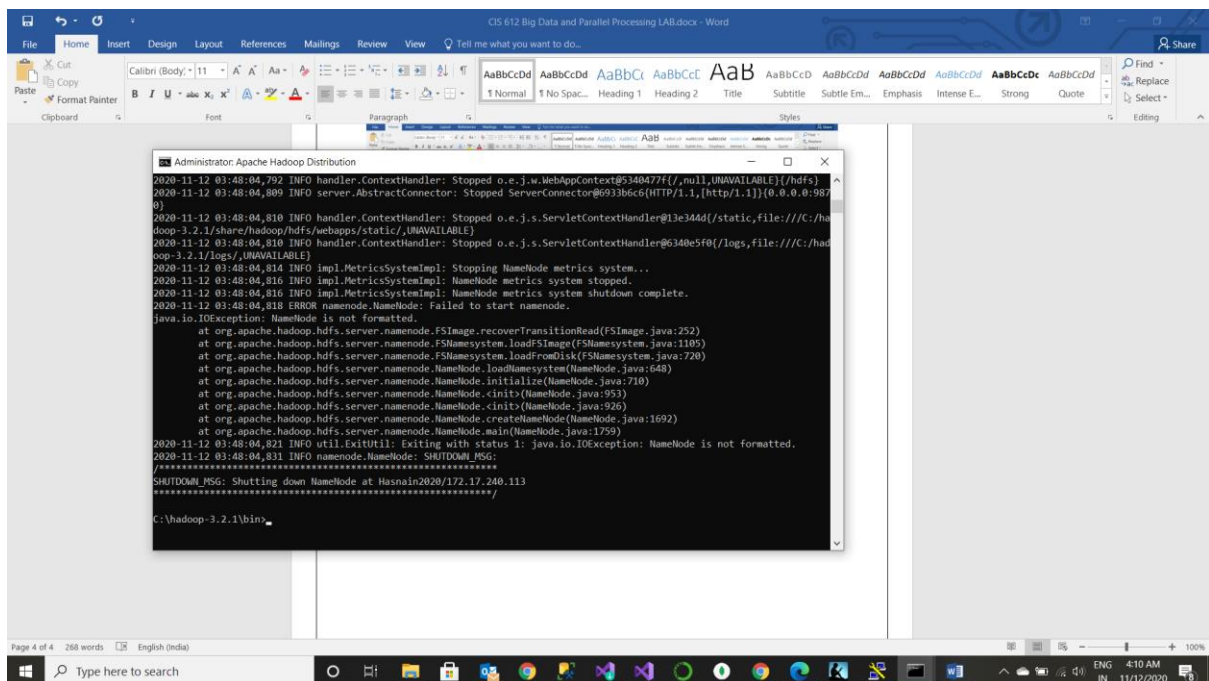
Running the Hadoop single node system

Go to the cmd prompt and go to the Hadoop-3.2.1/bin folder and type start-all.cmd to start all the nodes in the Hadoop system.



This would start the following

Name Node



Data Node

The screenshot shows the 'Data Node' log window for the 'datanode' process. The log contains multiple entries indicating connection attempts to the server at localhost/127.0.0.1:9000. The entries show a sequence of 'INFO' messages from the 'ipc.Client' and 'ipc.Server' components, followed by a 'WARN' message from 'datanode.DataNode' stating 'Problem connecting to server: localhost/127.0.0.1:9000'. The log also includes 'INFO' messages from the 'ipc.Server' component, such as 'IPC Server Responder: starting' and 'IPC Server listener on 8031: starting'. The log is displayed in a window titled 'Apache Hadoop Distribution - hadoop - datanode'.

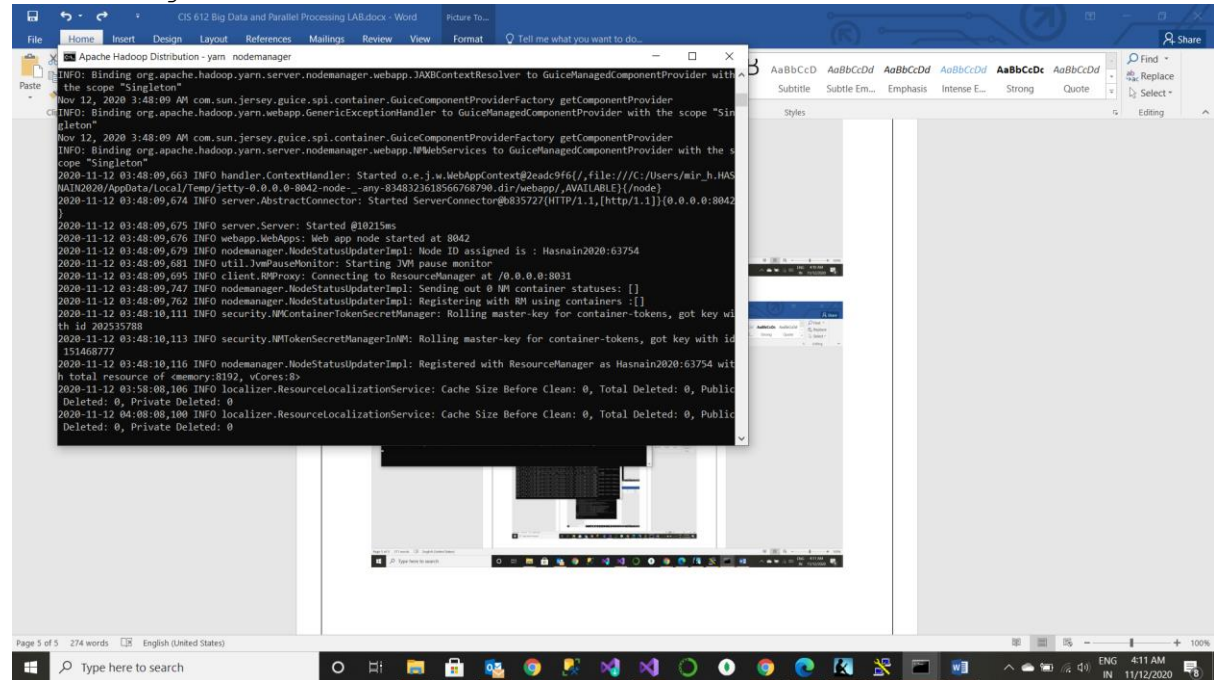
```
2020-11-12 04:09:52,430 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 2 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:09:55,431 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 3 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:09:58,434 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 4 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:01,438 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 5 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:04,440 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 6 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:07,445 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 7 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:10,448 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 8 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:13,452 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 9 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:15,457 WARN datanode.DataNode: Problem connecting to server: localhost/127.0.0.1:9000
2020-11-12 04:10:23,460 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 0 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:26,464 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 1 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:29,469 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 2 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:32,474 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 3 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:35,476 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 4 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
2020-11-12 04:10:38,479 INFO ipc.Client: Retrying connect to server: localhost/127.0.0.1:9000. Already tried 5 time(s);
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10, sleepTime=1000 MILLISECONDS)
```

Resource Manager

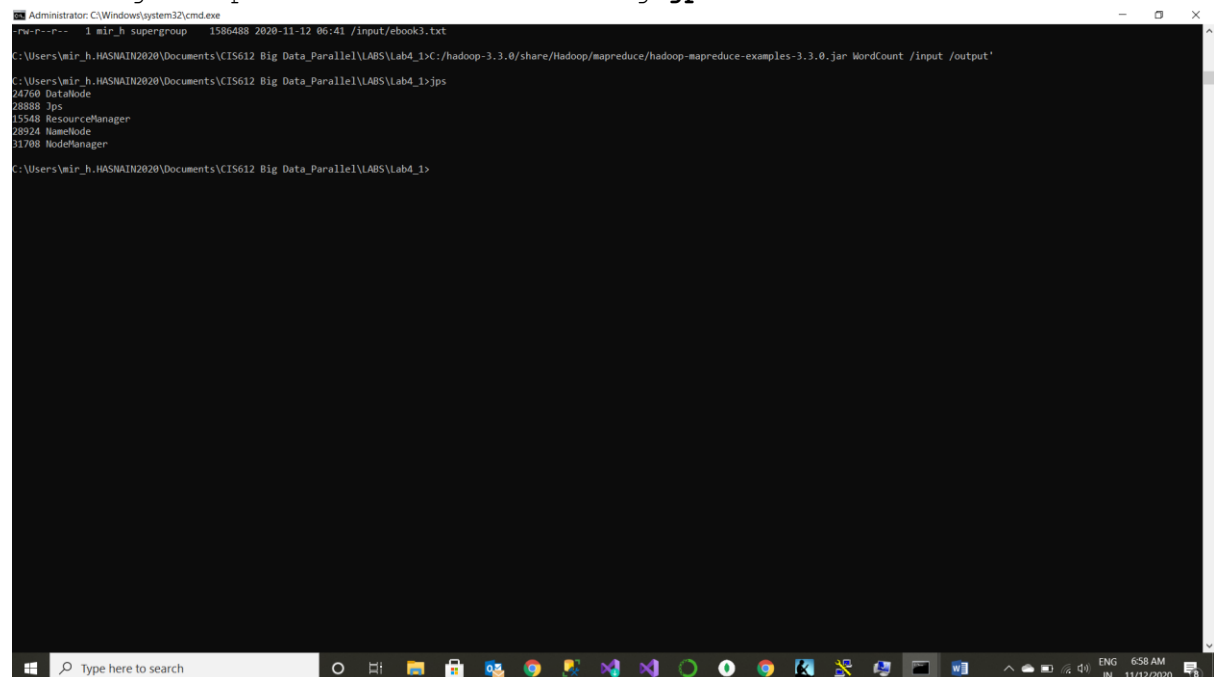
The screenshot shows the 'Resource Manager' log window for the 'resourcemanager' process. The log contains multiple entries indicating the startup of the Resource Manager. The entries show a sequence of 'INFO' messages from the 'ipc.Server' and 'ipc.Client' components, followed by a 'WARN' message from 'datanode.DataNode' stating 'Problem connecting to server: localhost/127.0.0.1:9000'. The log also includes 'INFO' messages from the 'ipc.Server' component, such as 'IPC Server Responder: starting' and 'IPC Server listener on 8031: starting'. The log is displayed in a window titled 'Apache Hadoop Distribution - yarn - resourcemanager'.

```
2020-11-12 03:48:07,960 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.server.api.ResourceTracker
to the server
2020-11-12 03:48:07,970 INFO ipc.Server: IPC Server Responder: starting
2020-11-12 03:48:07,970 INFO ipc.Server: IPC Server listener on 8031: starting
2020-11-12 03:48:07,985 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2020-11-12 03:48:08,004 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlockingQueue, queue
Capacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2020-11-12 03:48:08,020 INFO ipc.Server: Starting Socket Reader #1 for port 8030
2020-11-12 03:48:08,039 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.server.api.ApplicationMasterProt
ocolPB to the server
2020-11-12 03:48:08,045 INFO ipc.Server: IPC Server Responder: starting
2020-11-12 03:48:08,045 INFO ipc.Server: IPC Server listener on 8030: starting
2020-11-12 03:48:08,105 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlockingQueue, queue
Capacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2020-11-12 03:48:08,168 INFO ipc.Server: Starting Socket Reader #1 for port 8032
2020-11-12 03:48:08,172 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.server.api.ApplicationClientProt
ocolPB to the server
2020-11-12 03:48:08,173 INFO ipc.Server: IPC Server Responder: starting
2020-11-12 03:48:08,173 INFO ipc.Server: IPC Server listener on 8032: starting
2020-11-12 03:48:08,181 INFO resourcemanager.ResourceManager: Transitioned to active state
2020-11-12 03:48:10,084 INFO resourcemanager.ResourceTrackerService: NodeManager from node Hasnain2020:63754 htt
pPort: 8042 registered with capability: <memory:8192, vCores:8>, assigned nodeId Hasnain2020:63754
2020-11-12 03:48:10,090 INFO rmmode.RMNodeImpl: Hasnain2020:63754 Node Transitioned from NEW to RUNNING
2020-11-12 03:48:10,118 INFO capacity.CapacityScheduler: Added node Hasnain2020:63754 clusterResource: <memory:8192, vC
ores:8>
2020-11-12 03:58:07,868 INFO scheduler.AbstractYarnScheduler: Release request cache is cleaned up
```

Node Manager



Checking the port of the daemons using `jps` command



Open the browser and type localhost:9870 and localhost:8086/clusters to see the Hadoop connection and the nodes running on it .

The screenshot shows two screenshots of the Hadoop web interface. The top screenshot is the 'Overview' page for 'localhost:9000' (active). It displays metadata for the Hadoop cluster, including the start time, version, compiled date, cluster ID, and block pool ID. The bottom screenshot is the 'All Applications' page, which shows cluster metrics, node metrics, scheduler metrics, and a table of running applications. The table is currently empty, showing 'No data available in table'.

Overview 'localhost:9000' (✓active)

Started:	Thu Nov 12 06:24:46 +0530 2020
Version:	3.3.0, raa96f1871bfd858f9bac59cf2a81ec470da649af
Compiled:	Tue Jul 07 00:14:00 +0530 2020 by brahma from branch-3.3.0
Cluster ID:	CID-0ac3c6ac-bc64-4296-8f3b-362fab35b811
Block Pool ID:	BP-1846466883-172.17.240.113-1605142393221

Summary

Security is off.
 Safemode is off.
 1 files and directories, 0 blocks (0 replicated blocks, 0 erasure coded block groups) = 1 total filesystem object(s).
 Heap Memory used 78.58 MB of 306.5 MB Heap Memory. Max Heap Memory is 889 MB.
 Non Heap Memory used 48.26 MB of 49.86 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

Configured Capacity: 458.62 GB

All Applications

Cluster Metrics

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

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes
1	0	0	0

Scheduler Metrics

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

Show 20 entries

ID	User	Name	Application Type	Application Tags	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Alloc CP VCo
No data available in table													

Showing 0 to 0 of 0 entries

Putting the input files into the HDFS input folder

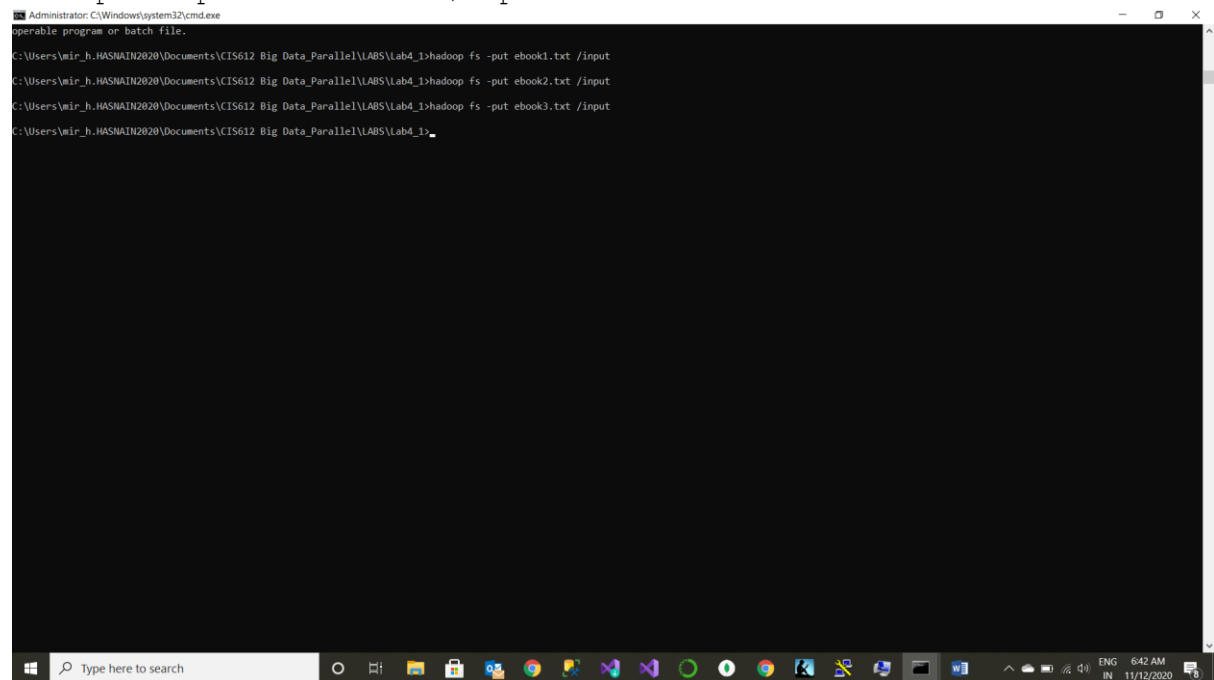
Download ebooks from the internet and store them into your local file folder. Now navigate to that folder in your command prompt and type the following commands to put the input files into from the local folder to the HDFS input folder. But before that we must make an input folder

```
hadoop dfs -mkdir /input
```

Using the command **hadoop fs -put ebook3.txt /input**

```
hadoop fs -put ebook1.txt /input
```

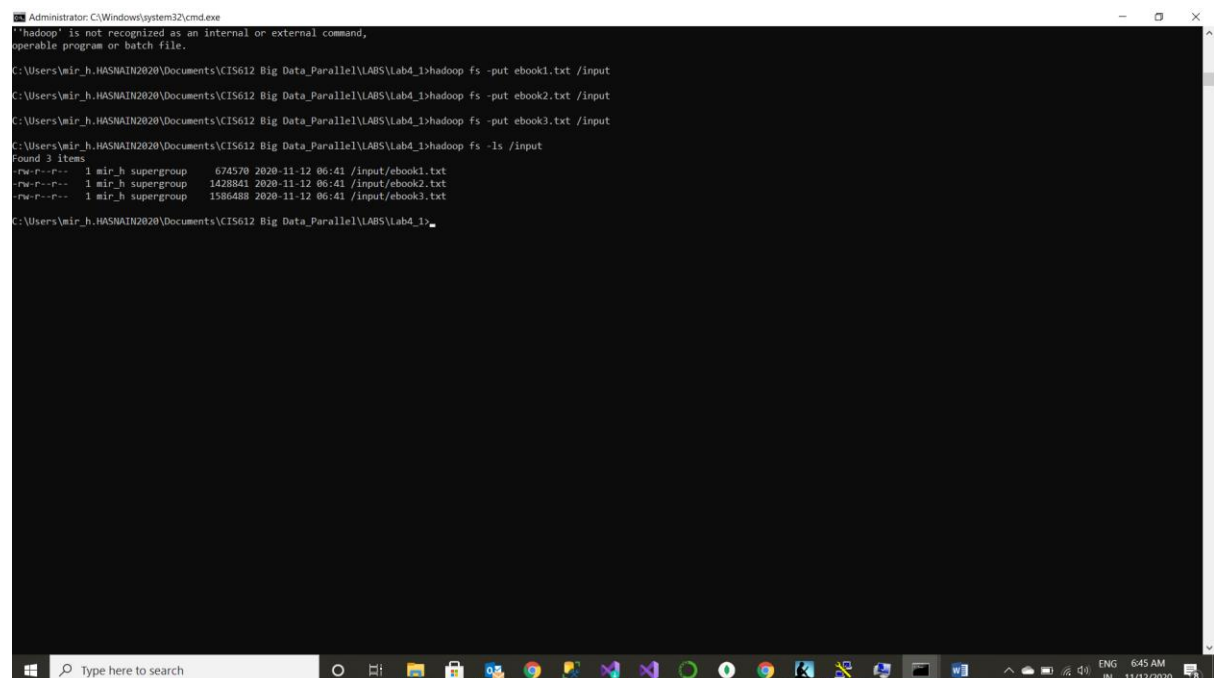
```
hadoop fs -put ebook2.txt /input
```



```
Administrator: C:\Windows\system32\cmd.exe
operable program or batch file.

C:\Users\mir_h.HASNAIN2020\Documents\CIS612 Big Data_Parallel\LAB5\Lab4_1>hadoop fs -put ebook1.txt /input
C:\Users\mir_h.HASNAIN2020\Documents\CIS612 Big Data_Parallel\LAB5\Lab4_1>hadoop fs -put ebook2.txt /input
C:\Users\mir_h.HASNAIN2020\Documents\CIS612 Big Data_Parallel\LAB5\Lab4_1>hadoop fs -put ebook3.txt /input
C:\Users\mir_h.HASNAIN2020\Documents\CIS612 Big Data_Parallel\LAB5\Lab4_1>
```

We have successfully put the input files into the HDFS system and displaying it using **hadoop fs -ls /user/input**



```
Administrator: C:\Windows\system32\cmd.exe
'hadoop' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\mir_h.HASNAIN2020\Documents\CIS612 Big Data_Parallel\LAB5\Lab4_1>hadoop fs -put ebook1.txt /input
C:\Users\mir_h.HASNAIN2020\Documents\CIS612 Big Data_Parallel\LAB5\Lab4_1>hadoop fs -put ebook2.txt /input
C:\Users\mir_h.HASNAIN2020\Documents\CIS612 Big Data_Parallel\LAB5\Lab4_1>hadoop fs -put ebook3.txt /input
C:\Users\mir_h.HASNAIN2020\Documents\CIS612 Big Data_Parallel\LAB5\Lab4_1>hadoop fs -ls /input
Found 3 items
-rw-r--r-- 1 mir_h supergroup 674570 2020-11-12 06:41 /input/ebook1.txt
-rw-r--r-- 1 mir_h supergroup 1428841 2020-11-12 06:41 /input/ebook2.txt
-rw-r--r-- 1 mir_h supergroup 1586488 2020-11-12 06:41 /input/ebook3.txt
C:\Users\mir_h.HASNAIN2020\Documents\CIS612 Big Data_Parallel\LAB5\Lab4_1>
```



```
C:/hadoop-3.3.0/share/Hadoop/mapreduce/hadoop-mapreduce-examples-3.3.0.jar WordCount /input /output'
```

Running the WordCount.java file

```
/hadoop jar hadoop-examples-*.jar wordcount input output
```

Source Code:

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount {

    public static class Map extends 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, Context context) throws IOException,
        InterruptedException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                context.write(word, one);
            }
        }
    }

    public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
        throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();

        Job job = new Job(conf, "wordcount");

        job.setOutputKeyClass(Text.class);
```

```
job.setOutputValueClass(IntWritable.class);

job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);

job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

job.waitForCompletion(true);
}
```