Assignment 1

Big Data Analytics – CS1701 (VII Sem)

Start Date: 7 Sep 2020 **Submission Date: 15 Sep 2020**

MM: 5 Marks

Question: What is Hadoop? Explain features of Hadoop with the Hadoop Ecosystem. Describe with snapshot, all the installation and working process step by step of Hadoop 3 on Ubuntu / Windows.

Solution:

Apache Hadoop is a collection of open-source software utilities that facilitate using a network of many computers to solve problems involving massive amounts of data and computation. It provides a software framework for distributed storage and processing of big data using the MapReduce programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage.

Features of Hadoop:

- **Ability to store and process huge amounts of any kind of data, quickly.** With data volumes and varieties constantly increasing, especially from social media and the Internet of Things (IoT), that's a key consideration.
- **Computing power.** Hadoop's distributed computing model processes big data fast. The more computing nodes you use, the more processing power you have.
- **Fault tolerance.** Data and application processing are protected against hardware failure. If a node goes down, jobs are automatically redirected to other nodes to make sure the distributed computing does not fail. Multiple copies of all data are stored automatically.
- **Flexibility.** Unlike traditional relational databases, you don't have to preprocess data before storing it. You can store as much data as you want and decide how to use it later. That includes unstructured data like text, images and videos.

- **Low cost.** The open-source framework is free and uses commodity hardware to store large quantities of data.
- **Scalability.** You can easily grow your system to handle more data simply by adding nodes. Little administration is required.
- **Feasibility.** Unlike the traditional system, Hadoop can process unstructured data. Thus provide feasibility to the users to analyse data of any formats and size.
- **Data Reliability.** In Hadoop due to the replication of data in the cluster, data is stored reliably on the cluster machines despite machine failures.
- **Hadoop is based on Data Locality Concept.** Hadoop is popularly known for its data locality feature means moving computation logic to the data, rather than moving data to the computation logic. This features of Hadoop reduces the bandwidth utilization in a system.

Hadoop Ecosystem:

Hadoop Ecosystem is a platform or a suite which provides various services to solve the big data problems. It includes Apache projects and various commercial tools and solutions. There are four major elements of Hadoop i.e. HDFS, MapReduce, YARN, and Hadoop Common. Most of the tools or solutions are used to supplement or support these major elements. All these tools work collectively to provide services such as absorption, analysis, storage and maintenance of data etc.

Following are the components that collectively form a Hadoop ecosystem:

- **HDFS:** Hadoop Distributed File System
- **YARN:** Yet Another Resource Negotiator
- **MapReduce:** Programming based Data Processing
- **Spark:** In-Memory data processing
- **PIG, HIVE:** Query based processing of data services
- **HBase:** NoSQL Database
- **Mahout, Spark MLLib:** Machine Learning algorithm libraries
- **Solar, Lucene:** Searching and Indexing
- **Zookeeper:** Managing cluster
- **Oozie:** Job Scheduling

HDFS:

- HDFS is the primary or major component of Hadoop ecosystem and 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.
- HDFS consists of two core components i.e.
  1. Name node
  2. Data Node
- Name Node is the prime node which contains metadata (data about data) requiring comparatively fewer resources than the data nodes that stores the actual data. These data nodes are commodity hardware in the distributed environment. Undoubtedly, making Hadoop cost effective.
- HDFS maintains all the coordination between the clusters and hardware, thus working at the heart of the system.

YARN:

- Yet Another Resource Negotiator, as the name implies, YARN is the one who helps to manage the resources across the clusters. In short, it performs scheduling and resource allocation for the Hadoop System.
- Consists of three major components i.e.
  1. Resource Manager
  2. Nodes Manager
  3. Application Manager
- Resource manager has the privilege of allocating resources for the applications in a system whereas Node managers work on the allocation of resources such as CPU, memory, bandwidth per machine and later on acknowledges the resource manager. Application manager works as an interface between the resource manager and node manager and performs negotiations as per the requirement of the two.

MapReduce:

- By making the use of distributed and parallel algorithms, MapReduce makes it possible to carry over the processing's logic and helps to write applications which transform big data sets into a manageable one.
- MapReduce makes the use of two functions i.e. `Map()` and `Reduce()` whose task is:
  1. *Map()* performs sorting and filtering of data and thereby organizing them in the form of group. Map generates a key-value pair based result which is later on processed by the Reduce() method.
  2. *Reduce()*, as the name suggests does the summarization by aggregating the mapped data. In simple, Reduce() takes the output generated by Map() as input and combines those tuples into smaller set of tuples.

PIG:

- Pig was basically developed by Yahoo which works on a pig Latin language, which is Query based language similar to SQL.
- It is a platform for structuring the data flow, processing and analyzing huge data sets.
- Pig does the work of executing commands and in the background, all the activities of MapReduce are taken care of. After the processing, pig stores the result in HDFS.

- Pig Latin language is specially designed for this framework which runs on Pig Runtime. Just the way Java runs on the JVM.
- Pig helps to achieve ease of programming and optimization and hence is a major segment of the Hadoop Ecosystem.

HIVE:

- With the help of SQL methodology and interface, HIVE performs reading and writing of large data sets. However, its query language is called as HQL (Hive Query Language).
- It is highly scalable as it allows real-time processing and batch processing both. Also, all the SQL datatypes are supported by Hive thus, making the query processing easier.
- Similar to the Query Processing frameworks, HIVE too comes with two components: *JDBC Drivers* and *HIVE Command Line*.
- JDBC, along with ODBC drivers work on establishing the data storage permissions and connection whereas HIVE Command line helps in the processing of queries.

Mahout:

- Mahout, allows Machine Learnability to a system or application. Machine Learning, as the name suggests helps the system to develop itself based on some patterns, user/environmental interaction or om the basis of algorithms.
- It provides various libraries or functionalities such as collaborative filtering, clustering, and classification which are nothing but concepts of Machine learning. It allows invoking algorithms as per our need with the help of its own libraries.

Apache Spark:

- It's a platform that handles all the process consumptive tasks like batch processing, interactive or iterative real-time processing, graph conversions, and visualization, etc.
- It consumes in memory resources hence, thus being faster than the prior in terms of optimization.
- Spark is best suited for real-time data whereas Hadoop is best suited for structured data or batch processing, hence both are used in most of the companies interchangeably.

Apache HBase:

- It's a NoSQL database which supports all kinds of data and thus capable of handling anything of Hadoop Database. It provides capabilities of Google's BigTable, thus able to work on Big Data sets effectively.
- At times where we need to search or retrieve the occurrences of something small in a huge database, the request must be processed within a short quick span of time. At such times, HBase comes handy as it gives us a tolerant way of storing limited data.

Other Components: Apart from all of these, there are some other components too that carry out a huge task in order to make Hadoop capable of processing large datasets. They are as follows:

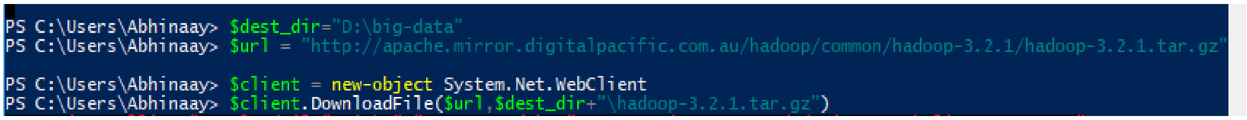
- **Solr, Lucene:** These are the two services that perform the task of searching and indexing with the help of some java libraries, especially Lucene is based on Java which allows spell check mechanism, as well. However, Lucene is driven by Solr.

- **Zookeeper:** There was a huge issue of management of coordination and synchronization among the resources or the components of Hadoop which resulted in inconsistency, often. Zookeeper overcame all the problems by performing synchronization, inter-component based communication, grouping, and maintenance.
- **Oozie:** Oozie simply performs the task of a scheduler, thus scheduling jobs and binding them together as a single unit. There is two kinds of jobs .i.e Oozie workflow and Oozie coordinator jobs. Oozie workflow is the jobs that need to be executed in a sequentially ordered manner whereas Oozie Coordinator jobs are those that are triggered when some data or external stimulus is given to it.

Installation Process of Hadoop 3 on Windows:

Step1: Download Hadoop binary package

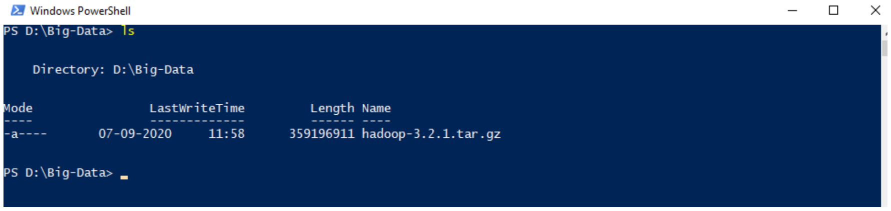
```
$dest_dir="F:\big-data"
$url = "http://apache.mirror.digitalpacific.com.au/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz"
$client = new-object System.Net.WebClient
$client.DownloadFile($url,$dest_dir+"\hadoop-3.2.1.tar.gz")
```



```
PS C:\Users\Abhinaay> $dest_dir="D:\big-data"
PS C:\Users\Abhinaay> $url = "http://apache.mirror.digitalpacific.com.au/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz"

PS C:\Users\Abhinaay> $client = new-object System.Net.WebClient
PS C:\Users\Abhinaay> $client.DownloadFile($url,$dest_dir+"\hadoop-3.2.1.tar.gz")
```

Once it is downloaded, we can verify.



```
PS D:\Big-Data> ls


    Directory: D:\Big-Data

Mode                LastWriteTime         Length Name
----                -------------         ------ ----
-a----       07-09-2020     11:58      359196911 hadoop-3.2.1.tar.gz

PS D:\Big-Data> _
```

Step2: Unpack the package

Now we need to unpack the downloaded package using GUI tool (like 7 Zip) or command line. For me, I will use git bash to unpack it.

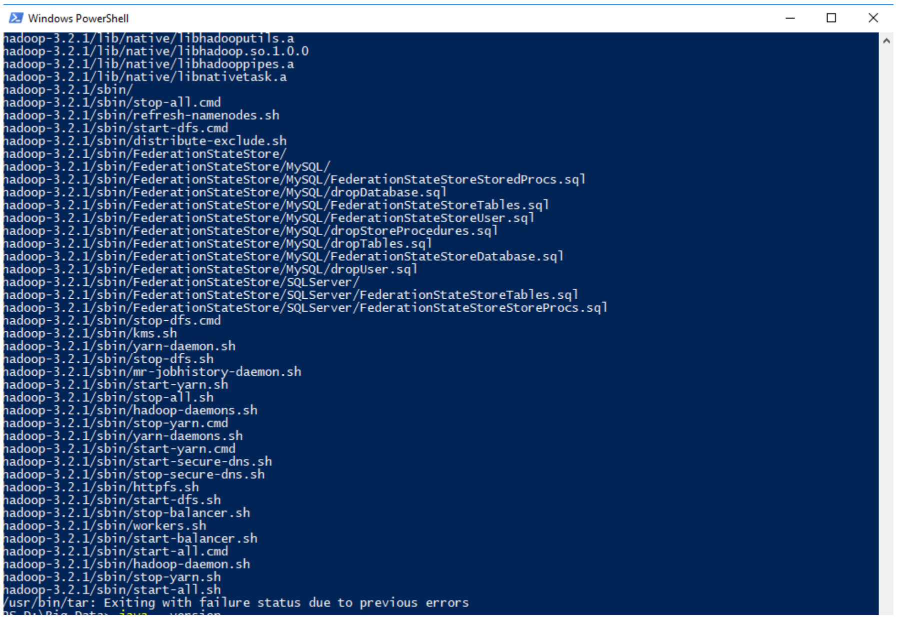
Open git bash and change the directory to the destination folder:

```
cd F:/big-data
```

And then run the following command to unzip:

```
tar -xvzf  hadoop-3.2.1.tar.gz
```

The command will take quite a few minutes as there are numerous files included and the latest version introduced many new features.



```
hadoop-3.2.1/lib/native/libhadooputils.a
hadoop-3.2.1/lib/native/libhadoop.so.1.0.0
hadoop-3.2.1/lib/native/libhadooppipes.a
hadoop-3.2.1/lib/native/libnativetask.a
hadoop-3.2.1/sbin/
hadoop-3.2.1/sbin/stop-all.cmd
hadoop-3.2.1/sbin/refresh-namenodes.sh
hadoop-3.2.1/sbin/start-dfs.cmd
hadoop-3.2.1/sbin/distribute-exclude.sh
hadoop-3.2.1/sbin/FederationStateStore/
hadoop-3.2.1/sbin/FederationStateStore/MySQL/
hadoop-3.2.1/sbin/FederationStateStore/MySQL/FederationStateStoreStoredProcs.sql
hadoop-3.2.1/sbin/FederationStateStore/MySQL/dropDatabase.sql
hadoop-3.2.1/sbin/FederationStateStore/MySQL/FederationStateStoreTables.sql
hadoop-3.2.1/sbin/FederationStateStore/MySQL/FederationStateStoreUser.sql
hadoop-3.2.1/sbin/FederationStateStore/MySQL/dropStoreProcedures.sql
hadoop-3.2.1/sbin/FederationStateStore/MySQL/dropTables.sql
hadoop-3.2.1/sbin/FederationStateStore/MySQL/FederationStateStoreDatabase.sql
hadoop-3.2.1/sbin/FederationStateStore/MySQL/dropUser.sql
hadoop-3.2.1/sbin/FederationStateStore/SQLServer/
hadoop-3.2.1/sbin/FederationStateStore/SQLServer/FederationStateStoreTables.sql
hadoop-3.2.1/sbin/FederationStateStore/SQLServer/FederationStateStoreStoreProcs.sql
hadoop-3.2.1/sbin/stop-dfs.cmd
hadoop-3.2.1/sbin/kms.sh
hadoop-3.2.1/sbin/yarn-daemon.sh
hadoop-3.2.1/sbin/stop-dfs.sh
hadoop-3.2.1/sbin/mr-jobhistory-daemon.sh
hadoop-3.2.1/sbin/start-yarn.sh
hadoop-3.2.1/sbin/stop-all.sh
hadoop-3.2.1/sbin/hadoop-daemons.sh
hadoop-3.2.1/sbin/stop-yarn.cmd
hadoop-3.2.1/sbin/yarn-daemons.sh
hadoop-3.2.1/sbin/start-yarn.cmd
hadoop-3.2.1/sbin/start-secure-dns.sh
hadoop-3.2.1/sbin/stop-secure-dns.sh
hadoop-3.2.1/sbin/httpfs.sh
hadoop-3.2.1/sbin/start-dfs.sh
hadoop-3.2.1/sbin/stop-balancer.sh
hadoop-3.2.1/sbin/workers.sh
hadoop-3.2.1/sbin/start-balancer.sh
hadoop-3.2.1/sbin/start-all.cmd
hadoop-3.2.1/sbin/hadoop-daemon.sh
hadoop-3.2.1/sbin/stop-yarn.sh
hadoop-3.2.1/sbin/start-all.sh
/usr/bin/tar: Exiting with failure status due to previous errors
PS D:\Big-Data> java --version
```

Step 3 - Install Hadoop native IO binary

Hadoop on Linux includes optional Native IO support. However Native IO is mandatory on Windows and without it you will not be able to get your installation working. The Windows native IO libraries are not included as part of Apache Hadoop release. Thus we need to build and install it.

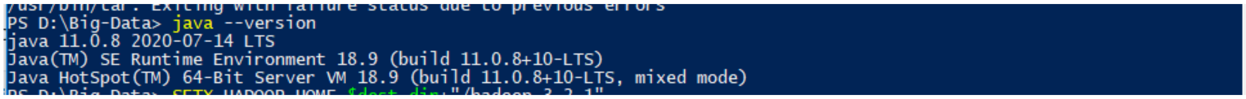
Download all the files in the following location and save them to the **bin** folder under Hadoop folder. For my environment, the full path is: **F:\big-data\hadoop-3.2.1\bin**. Remember to change it to your own path accordingly.

Step 4 - (Optional) Java JDK installation

Java JDK is required to run Hadoop. If you have not installed Java JDK please install it.

Once you complete the installation, please run the following command in PowerShell or Git Bash to verify:

```
$ java -version
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)
```



```
/usr/bin/tar: Exiting with failure status due to previous errors
PS D:\Big-Data> java --version
java 11.0.8 2020-07-14 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.8+10-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.8+10-LTS, mixed mode)
PS D:\Big-Data> SETX HADOOP_HOME $dest_dir+"/hadoop-3.2.1"
```

Step 5 - Configure environment variables

Now we've downloaded and unpacked all the artefacts we need to configure two important environment variables.

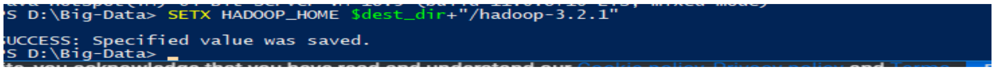
Configure JAVA_HOME environment variable

As mentioned earlier, Hadoop requires Java and we need to configure **JAVA_HOME** environment variable (though it is not mandatory but I recommend it).

First, we need to find out the location of Java SDK. In my system, the path is: **D:\Java\jdk1.8.0_161**.

And then run the following command in the previous PowerShell window:

```
SETX JAVA_HOME "D:\Java\jdk1.8.0_161"
```



```
S D:\Big-Data> SETX HADOOP_HOME $dest_dir+"/hadoop-3.2.1"

UCCESS: Specified value was saved.
S D:\Big-Data> _
```

Configure PATH environment variable

Once we finish setting up the above two environment variables, we need to add the **bin** folders to the **PATH** environment variable.

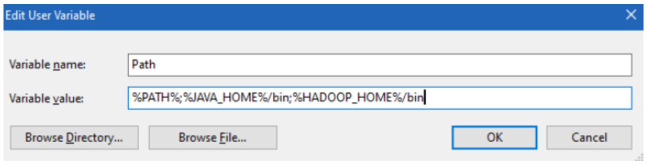
If **PATH** environment exists in your system, you can also manually add the following two paths to it:

- %JAVA_HOME%/bin
- %HADOOP_HOME%/bin

Alternatively, you can run the following command to add them:

```
setx PATH "$env:PATH;$env:JAVA_HOME/bin;$env:HADOOP_HOME/bin"
```

If you don't have other user variables setup in the system, you can also directly add a **Path** environment variable that references others to make it short:



Variable name: Path

Variable value: %PATH%;%JAVA_HOME%/bin;%HADOOP_HOME%/bin

You should also be able to run the following command:

```
hadoop -version
java version "1.8.0_161"
Java(TM) SE Runtime Environment (build 1.8.0_161-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.161-b12, mixed mode)
```

Step 6 - Configure Hadoop

Now we are ready to configure the most important part - Hadoop configurations which involves Core, YARN, MapReduce, HDFS configurations.

Configure core site

Edit file **core-site.xml** in **%HADOOP_HOME%\etc\hadoop** folder. For my environment, the actual path is **F:\big-data\hadoop-3.2.1\etc\hadoop**.

Replace **configuration** element with the following:

```
<configuration>
   <property>
     <name>fs.default.name</name>
     <value>hdfs://0.0.0.0:19000</value>
   </property>
</configuration>
```

Configure HDFS

Edit file **hdfs-site.xml** in **%HADOOP_HOME%\etc\hadoop** folder.

Before editing, please correct two folders in your system: one for namenode directory and another for data directory. For my system, I created the following two sub folders:

- F:\big-data\data\dfs\namespace_logs
- F:\big-data\data\dfs\data

Replace **configuration** element with the following (remember to replace the highlighted paths accordingly):

```
<configuration>
   <property>
     <name>dfs.replication</name>
     <value>1</value>
   </property>
   <property>
     <name>dfs.namenode.name.dir</name>
     <value>file:///F:/big-data/data/dfs/namespace_logs</value>
   </property>
   <property>
     <name>dfs.datanode.data.dir</name>
     <value>file:///F:/big-data/data/dfs/data</value>
   </property>
</configuration>
```

Configure MapReduce and YARN site

Edit file **mapred-site.xml** in **%HADOOP_HOME%\etc\hadoop** folder.

Replace **configuration** element with the following:

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

<value>%HADOOP_HOME%/share/hadoop/mapreduce/*,%HADOOP_HOME%/share/hadoop/mapreduce/lib/*,%HADOOP_HOME%/share/hadoop/common/*,%HADOOP_HOME%/share/hadoop/common/lib/*,%HADOOP_HOME%/share/hadoop/yarn/*,%HADOOP_HOME%/share/hadoop/yarn/lib/*,%HADOOP_HOME%/share/hadoop/hdfs/*,%HADOOP_HOME%/share/hadoop/hdfs/lib/*</value>
    </property>
</configuration>
```

Edit file **yarn-site.xml** in **%HADOOP_HOME%\etc\hadoop** folder.

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

<value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>
    </property>
</configuration>
```

Step 7 - Initialise HDFS & bug fix

Run the following command in Command Prompt

```
hdfs namenode -format
```

Step 8 - Start HDFS daemons

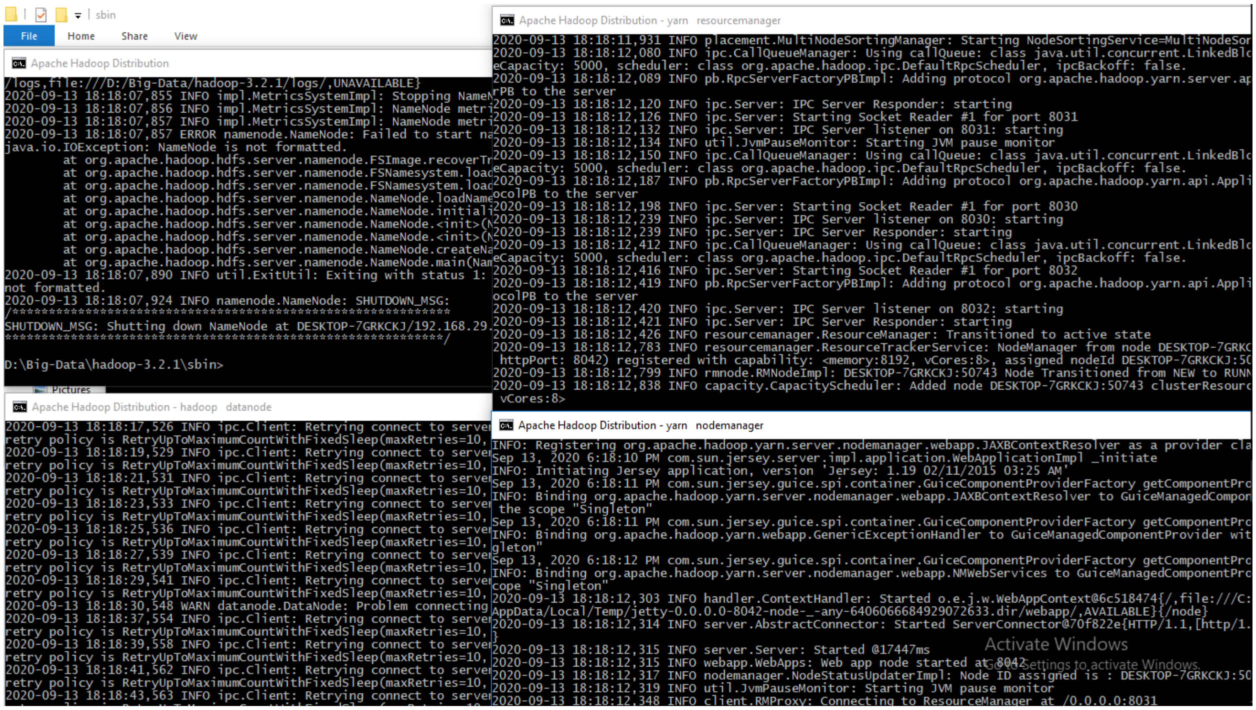
Run the following command to start HDFS daemons in Command Prompt:

```
%HADOOP_HOME%\sbin\start-dfs.cmd
```

Step 9 - Start YARN daemons

Run the following command in an elevated Command Prompt window (Run as administrator) to start YARN daemons:

```
%HADOOP_HOME%\sbin\start-yarn.cmd
```


sbin
File
Home
Share
View
Apache Hadoop Distribution
Pictures
Apache Hadoop Distribution - hadoop datanode
Apache Hadoop Distribution - yarn resourcemanager
Apache Hadoop Distribution - yarn nodemanager
Activate Windows
Go to Settings to activate Windows.

```
/logs,file:///D:/Big-Data/hadoop-3.2.1/logs/,UNAVAILABLE}
2020-09-13 18:18:07,855 INFO impl.MetricsSystemImpl: Stopping NameN
2020-09-13 18:18:07,856 INFO impl.MetricsSystemImpl: NameNode metri
2020-09-13 18:18:07,857 INFO impl.MetricsSystemImpl: NameNode metri
2020-09-13 18:18:07,857 ERROR namenode.NameNode: Failed to start na
java.io.IOException: NameNode is not formatted.
        at org.apache.hadoop.hdfs.server.namenode.FSImage.recoverTr
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.load
        at org.apache.hadoop.hdfs.server.namenode.FSNamesystem.load
        at org.apache.hadoop.hdfs.server.namenode.NameNode.loadName
        at org.apache.hadoop.hdfs.server.namenode.NameNode.initiali
        at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(N
        at org.apache.hadoop.hdfs.server.namenode.NameNode.<init>(N
        at org.apache.hadoop.hdfs.server.namenode.NameNode.createNa
        at org.apache.hadoop.hdfs.server.namenode.NameNode.main(Nam
2020-09-13 18:18:07,890 INFO util.ExitUtil: Exiting with status 1:
not formatted.
2020-09-13 18:18:07,924 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at DESKTOP-7GRKCKJ/192.168.29.
************************************************************/

D:\Big-Data\hadoop-3.2.1\sbin>
```

```
2020-09-13 18:18:17,526 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:19,529 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:21,531 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:23,533 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:25,536 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:27,539 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:29,541 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:30,548 WARN datanode.DataNode: Problem connecting
2020-09-13 18:18:37,554 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:39,558 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:41,562 INFO ipc.Client: Retrying connect to server
retry policy is RetryUpToMaximumCountWithFixedSleep(maxRetries=10,
2020-09-13 18:18:43,563 INFO ipc.Client: Retrying connect to server
```

```
2020-09-13 18:18:11,931 INFO placement.MultiNodeSortingManager: Starting NodeSortingService=MultiNodeSor
2020-09-13 18:18:12,080 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlo
eCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2020-09-13 18:18:12,089 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.server.ap
rPB to the server
2020-09-13 18:18:12,120 INFO ipc.Server: IPC Server Responder: starting
2020-09-13 18:18:12,126 INFO ipc.Server: Starting Socket Reader #1 for port 8031
2020-09-13 18:18:12,132 INFO ipc.Server: IPC Server listener on 8031: starting
2020-09-13 18:18:12,134 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2020-09-13 18:18:12,150 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlo
eCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2020-09-13 18:18:12,187 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.api.Appli
ocolPB to the server
2020-09-13 18:18:12,198 INFO ipc.Server: Starting Socket Reader #1 for port 8030
2020-09-13 18:18:12,239 INFO ipc.Server: IPC Server listener on 8030: starting
2020-09-13 18:18:12,239 INFO ipc.Server: IPC Server Responder: starting
2020-09-13 18:18:12,412 INFO ipc.CallQueueManager: Using callQueue: class java.util.concurrent.LinkedBlo
eCapacity: 5000, scheduler: class org.apache.hadoop.ipc.DefaultRpcScheduler, ipcBackoff: false.
2020-09-13 18:18:12,416 INFO ipc.Server: Starting Socket Reader #1 for port 8032
2020-09-13 18:18:12,419 INFO pb.RpcServerFactoryPBImpl: Adding protocol org.apache.hadoop.yarn.api.Appli
ocolPB to the server
2020-09-13 18:18:12,420 INFO ipc.Server: IPC Server listener on 8032: starting
2020-09-13 18:18:12,421 INFO ipc.Server: IPC Server Responder: starting
2020-09-13 18:18:12,426 INFO resourcemanager.ResourceManager: Transitioned to active state
2020-09-13 18:18:12,783 INFO resourcemanager.ResourceTrackerService: NodeManager from node DESKTOP-7GRKC
 httpPort: 8042) registered with capability: <memory:8192, vCores:8>, assigned nodeId DESKTOP-7GRKCKJ:50
2020-09-13 18:18:12,799 INFO rmnode.RMNodeImpl: DESKTOP-7GRKCKJ:50743 Node Transitioned from NEW to RUNN
2020-09-13 18:18:12,838 INFO capacity.CapacityScheduler: Added node DESKTOP-7GRKCKJ:50743 clusterResourc
 vCores:8>
```

```
INFO: Registering org.apache.hadoop.yarn.server.nodemanager.webapp.JAXBContextResolver as a provider cla
Sep 13, 2020 6:18:10 PM com.sun.jersey.server.impl.application.WebApplicationImpl _initiate
INFO: Initiating Jersey application, version 'Jersey: 1.19 02/11/2015 03:25 AM'
Sep 13, 2020 6:18:11 PM com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory getComponentPro
INFO: Binding org.apache.hadoop.yarn.server.nodemanager.webapp.JAXBContextResolver to GuiceManagedCompon
 the scope "Singleton"
Sep 13, 2020 6:18:11 PM com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory getComponentPro
INFO: Binding org.apache.hadoop.yarn.webapp.GenericExceptionHandler to GuiceManagedComponentProvider wit
gleton"
Sep 13, 2020 6:18:12 PM com.sun.jersey.guice.spi.container.GuiceComponentProviderFactory getComponentPro
INFO: Binding org.apache.hadoop.yarn.server.nodemanager.webapp.NMWebServices to GuiceManagedComponentPro
cope "Singleton"
2020-09-13 18:18:12,303 INFO handler.ContextHandler: Started o.e.j.w.WebAppContext@6c518474{/,file:///C:
AppData/Local/Temp/jetty-0.0.0.0-8042-node-_-any-6406066684929072633.dir/webapp/,AVAILABLE}{/node}
2020-09-13 18:18:12,314 INFO server.AbstractConnector: Started ServerConnector@70f822e{HTTP/1.1,[http/1.
}
2020-09-13 18:18:12,315 INFO server.Server: Started @17447ms
2020-09-13 18:18:12,315 INFO webapp.WebApps: Web app node started at 8042
2020-09-13 18:18:12,317 INFO nodemanager.NodeStatusUpdaterImpl: Node ID assigned is : DESKTOP-7GRKCKJ:50
2020-09-13 18:18:12,319 INFO util.JvmPauseMonitor: Starting JVM pause monitor
2020-09-13 18:18:12,348 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8031
```