Experiment - 2
Lab Manual

Aim : "Configuring and experiencing Hadoop Single node cluster. Start/stop script and monitoring."

Hadoop version reference – 2.8.2

Theory

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault-tolerant and designed using low-cost hardware.

HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

Apache HDFS or **Hadoop Distributed File System** is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines. Apache Hadoop HDFS Architecture follows a *Master/Slave Architecture*, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes). HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.

- **NameNode**

  The namenode is the commodity hardware that contains the GNU/Linux operating system and the namenode software. It is a software that can be run on commodity hardware. NameNode is the master node in the Apache Hadoop HDFS Architecture that maintains and manages the blocks present on the DataNodes (slave nodes). NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients. The HDFS architecture is built in such a way that the user data never resides on the NameNode. The data resides on DataNodes only.

Functions of Namenode:

- It is the master daemon that maintains and manages the DataNodes (slave nodes)
- It records the metadata of all the files stored in the cluster, e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc. There are two files associated with the metadata:
  - **FsImage:** It contains the complete state of the file system namespace since the start of the NameNode.
  - **EditLogs:** It contains all the recent modifications made to the file system with respect to the most recent FsImage.
- It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
- The NameNode is also responsible to take care of the **replication factor** of all the blocks.
- In **case of the DataNode failure**, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

DataNode:

DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. The DataNode is a block server that stores the data in the local file ext3 or ext4.

Functions of DataNode:

- These are slave daemons or process which runs on each slave machine.
- The actual data is stored on DataNodes.
- The DataNodes perform the low-level read and write requests from the file system's clients.
- They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.

Secondary NameNode:

Apart from these two daemons, there is a third daemon or a process called Secondary NameNode. The Secondary NameNode works concurrently with the primary NameNode as a **helper daemon.** And don't be confused about the Secondary NameNode being a **backup NameNode because it is not.**

Functions of Secondary NameNode:

- The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.
- It is responsible for combining the EditLogs with FsImage from the NameNode.
- It downloads the EditLogs from the NameNode at regular intervals and applies to FsImage. The new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time.

Hence, Secondary NameNode performs regular checkpoints in HDFS. Therefore, it is also called CheckpointNode.

Practical

Check out the contents of .bashrc file using following command.

```
[hadoop@hadoop-clone Desktop]$ cat ~/.bashrc
```

Following output can be seen (on an instance where hadoop setup is already done).

```
# .bashrc
# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
export JAVA_HOME=/usr/java/jdk1.8.0_131
export HADOOP_HOME=/opt/hadoop
export HADOOP_PREFIX=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
export HBASE_HOME=/opt/hbase
export HBASE_PREFIX=$HBASE_HOME
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:
$HBASE_HOME/bin:
```

Also check the contents of PATH environment variable using following command:

```
[hadoop@hadoop-clone Desktop]$ echo $PATH
```

Following output can be seen (as hadoop setup is already done).

```
/opt/ns-allinone-2.35/bin:/opt/ns-allinone-2.35/tcl8.5.10/unix:/opt/ns-allinone-2.35/tk8.5.10/unix:/usr/
lib64/qt-
3.3/bin:/usr/local/bin:/usr/bin:/bin:/usr/local/sbin:/usr/sbin:/sbin:/usr/java/jdk1.8.0_131/bin:/opt/hadoop
/bin:/opt/hadoop/sbin:/opt/hbase/bin::/home/hadoop/bin:/usr/java/jdk1.8.0_131/bin:/opt/hadoop/bin:/op
t/hadoop/sbin:/opt/hbase/bin:
```

Once setup is ready, start hdfs using following command:

```
[hadoop@hadoop-clone Desktop]$ start-dfs.sh
```

After excuting previous command, it's effect can be seen using following command.

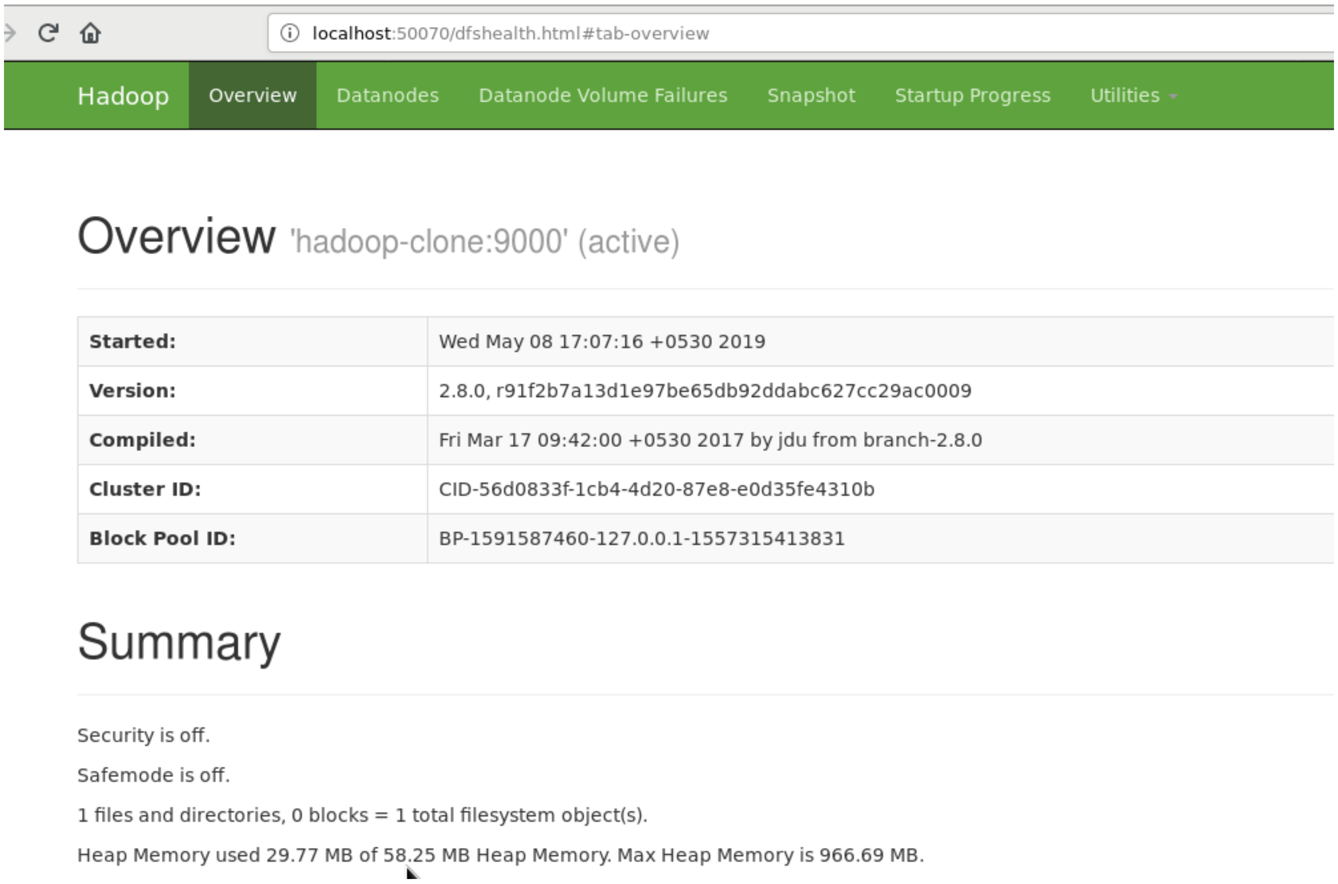
```
[hadoop@hadoop-clone Desktop]$ jps
```

This command will display output similar to below:

```
3426 DataNode
3766 Jps
3302 NameNode
3642 SecondaryNameNode
```

This actually shows that these many java processes are currently running on the machine.

Hadoop monitoring and HDFS Web GUI can be accessed by the URL http://localhost:50070/. Explore this Web GUI.


localhost:50070/dfshealth.html#tab-overview

Overview 'hadoop-clone:9000' (active)

Started:	Wed May 08 17:07:16 +0530 2019
Version:	2.8.0, r91f2b7a13d1e97be65db92ddabc627cc29ac0009
Compiled:	Fri Mar 17 09:42:00 +0530 2017 by jdu from branch-2.8.0
Cluster ID:	CID-56d0833f-1cb4-4d20-87e8-e0d35fe4310b
Block Pool ID:	BP-1591587460-127.0.0.1-1557315413831

Summary

Security is off.

Safemode is off.

1 files and directories, 0 blocks = 1 total filesystem object(s).

Heap Memory used 29.77 MB of 58.25 MB Heap Memory. Max Heap Memory is 966.69 MB.

Hadoop is installed in /opt/hadoop directory which can be seen in .bashrc file too. $HADOOP_HOME variable stores this path. So we can also use that variable whenever we want to move to this location.

Execute following commands to move to that location.

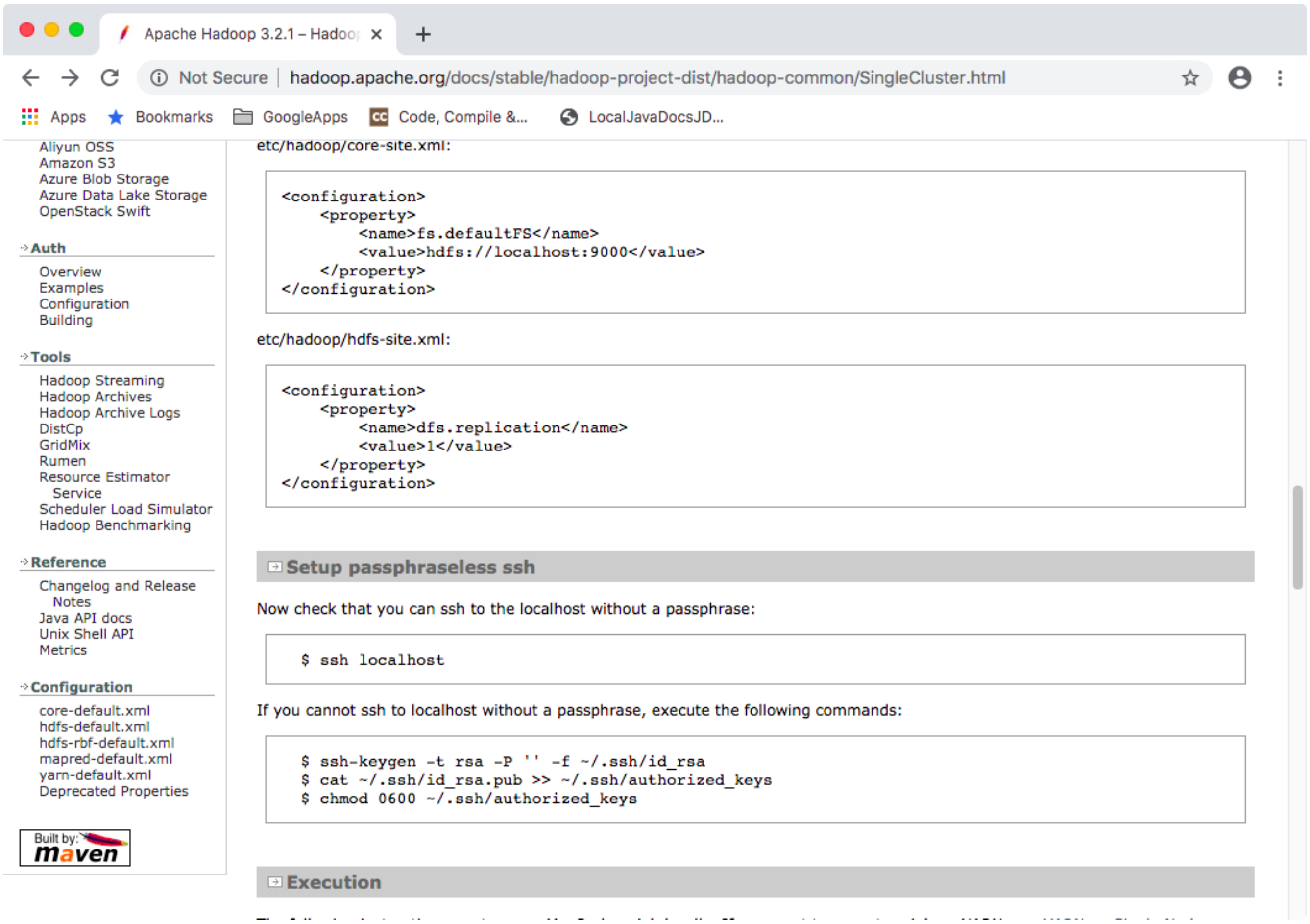
```
[hadoop@hadoop-clone Desktop]$ cd $HADOOP_HOME
[hadoop@hadoop-clone hadoop]$ pwd
/opt/hadoop
[hadoop@hadoop-clone hadoop]$ ls
bin  include  libexec      logs        README.txt  share
etc  lib      LICENSE.txt  NOTICE.txt  sbin        tmp
```

Configuration files of hadoop are located in /opt/hadoop/etc/hadoop directory. To explore these files, move to that location.

```
[hadoop@hadoop-clone hadoop]$ cd etc
```

```
[hadoop@hadoop-clone etc]$ ls
hadoop
[hadoop@hadoop-clone etc]$ cd hadoop
[hadoop@hadoop-clone hadoop]$ ls
capacity-scheduler.xml      httpfs-env.sh            mapred-env.sh
configuration.xsl           httpfs-log4j.properties  mapred-queues.xml.template
container-executor.cfg      httpfs-signature.secret  mapred-site.xml
core-site.xml               httpfs-site.xml          mapred-site.xml.template
hadoop-env.cmd              kms-acls.xml             slaves
hadoop-env.sh               kms-env.sh               ssl-client.xml.example
hadoop-metrics2.properties  kms-log4j.properties     ssl-server.xml.example
hadoop-metrics.properties   kms-site.xml             yarn-env.cmd
hadoop-policy.xml           log4j.properties         yarn-env.sh
hdfs-site.xml               mapred-env.cmd           yarn-site.xml
```

Using any editor, view the contents of core-site.xml and hdfs-site.xml. To understand the contents of these files, open https://hadoop.apache.org website. Refer Configuration section located in the bottom-left corner of the site to check various properties of various configuration files . Explore various properties.


Apache Hadoop 3.2.1 – Hadoo

Not Secure | hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

Apps Bookmarks GoogleApps Code, Compile &... LocalJavaDocsJD...

Aliyun OSS
Amazon S3
Azure Blob Storage
Azure Data Lake Storage
OpenStack Swift

Auth

Overview
Examples
Configuration
Building

Tools

Hadoop Streaming
Hadoop Archives
Hadoop Archive Logs
DistCp
GridMix
Rumen
Resource Estimator Service
Scheduler Load Simulator
Hadoop Benchmarking

Reference

Changelog and Release Notes
Java API docs
Unix Shell API
Metrics

Configuration

core-default.xml
hdfs-default.xml
hdfs-rbf-default.xml
mapred-default.xml
yarn-default.xml
Deprecated Properties

Built by: maven

etc/hadoop/core-site.xml:

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

etc/hadoop/hdfs-site.xml:

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

Setup passphraseless ssh

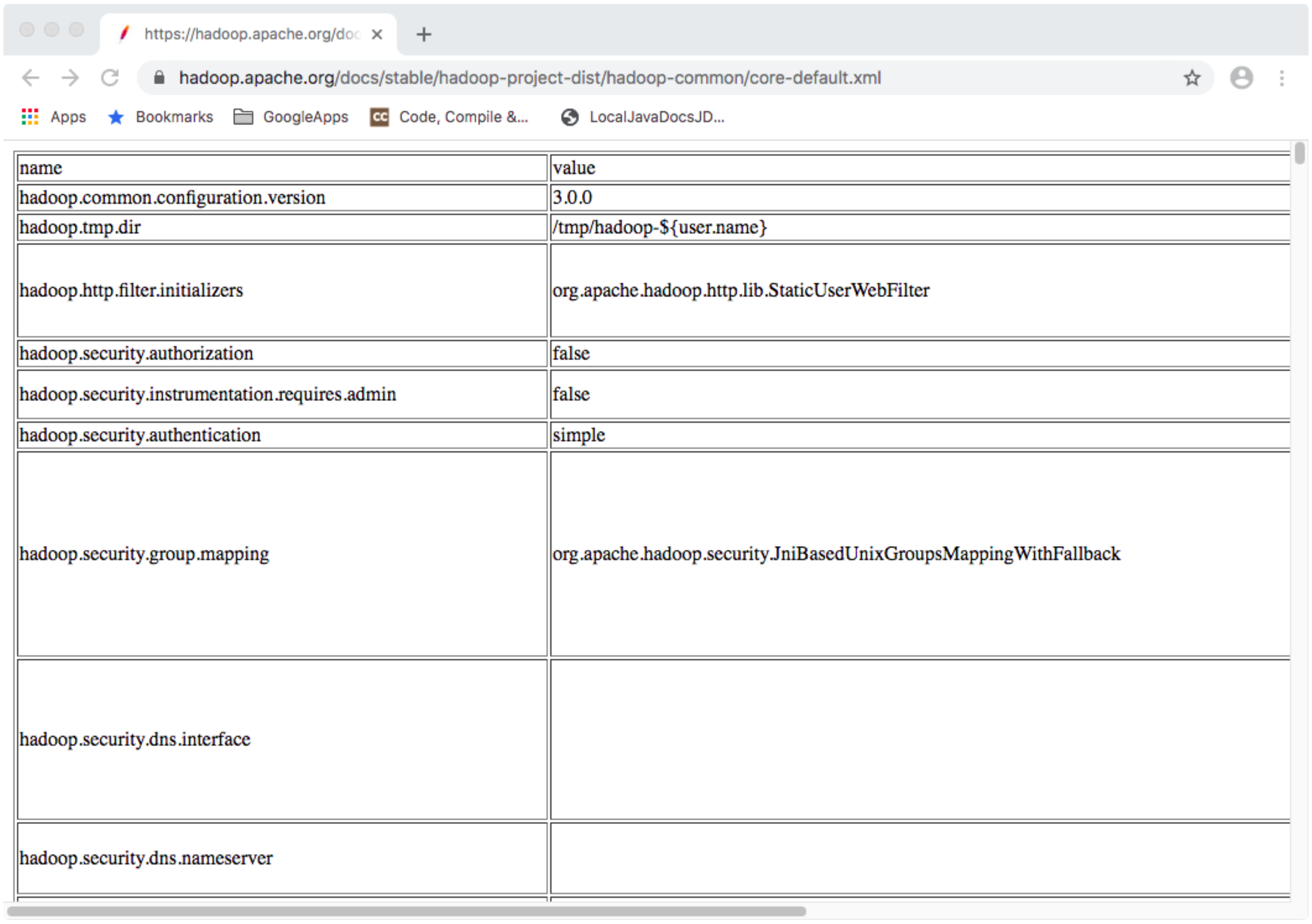
Now check that you can ssh to the localhost without a passphrase:

```
  $ ssh localhost
```

If you cannot ssh to localhost without a passphrase, execute the following commands:

```
  $ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
  $ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
  $ chmod 0600 ~/.ssh/authorized_keys
```

Execution


https://hadoop.apache.org/doc
hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/core-default.xml
Apps
Bookmarks
GoogleApps
Code, Compile &...
LocalJavaDocsJD...

name	value
hadoop.common.configuration.version	3.0.0
hadoop.tmp.dir	/tmp/hadoop-${user.name}
hadoop.http.filter.initializers	org.apache.hadoop.http.lib.StaticUserWebFilter
hadoop.security.authorization	false
hadoop.security.instrumentation.requires.admin	false
hadoop.security.authentication	simple
hadoop.security.group.mapping	org.apache.hadoop.security.JniBasedUnixGroupsMappingWithFallback
hadoop.security.dns.interface	
hadoop.security.dns.nameserver	

When hadoop is started, it sees the contents of these files various configurations. So to make any modification in these files, first stop hadoop using following command:

```
[hadoop@hadoop-clone hadoop]$ stop-dfs.sh
```

Also check using 'jps' command that everything stopped properly, then make modification in the configuration file. After doing any modification, for the first time namenode has to be formatted using following command:

```
[hadoop@hadoop-clone hadoop]$ hdfs namenode -format
```

After this, start hadoop using 'strat-dfs.sh' and also run 'jps' command to check that everything started properly.

Navigation Links (Default):

NameNode:
http://localhost:50070/dfshealth.html#tab-overview

DataNode
http://localhost:50075/datanode.html

http://localhost:8088/cluster/nodes

Know that various approaches to seek for Big Data and Analytics with hadoop distributions.
- Download and install apache hadoop or other on linux based system.
- Download and install Virtual Machine Software i.e. VirtualBox and load any Linux Distribution image (.iso) file to move onto installing hadoop single node setup
- ssh to remote system having hadoop installed already for practice or install if not available or remote desktop login via clients like teamviewer, etc.
- Find out internet/cloud based free/affordable solutions
- For multi-cluster linux
- May connect multiple computers at home via router local network or cross-cable peer to peer (2 nodes)
- Install multiple images of Linux distribution within Virtual Machine Software and configure cluster

(Windows Subsystem for Ubuntu may help to learn basics of Linux on Windows platform)

References:

1. https://hadoop.apache.org/
2. https://www.tutorialspoint.com/hadoop/hadoop_hdfs_overview.htm

3. https://www.edureka.co/blog/apache-hadoop-hdfs-architecture/

Exercise:

1. Download and install hadoop on your workstation or seek alternatives. Monitor the same using version specific navigation links.
2. Locate the log directory and refer for troubleshooting.
3. Change the URL of HDFS Web UI to localhost:51234 instead of localhost:50070 and the access the web UI using new URL. [Hint: refer configuration files] [Know that traditionally configuration based servers need to be restarted after any server configuration change.]
4. Undo changes done in task1.
5. Find out what is current location of namenode and datanode storage directories. Normally whenever cleanup is required, tmp folder is one to be emptied up (not to confuse with formatting file system). Hence, it is advised data nad metadata better be outside tmp. Do the needful to have namenode and datenode directories in /opt/hadoop, if present within tmp.
6. Analyze the output of following command:

```
 [hadoop@hadoop-clone hadoop]$ ps -ef |grep NameNode
 [hadoop@hadoop-clone hadoop]$ ps -ef |grep DataNode
```

Document last updated on 13th July, 2020 by Prof. Jigar M. Pandya