Setup Oracle Java

Check the version of Java installed.).
Remove all the open source Java installations : sudo apt-get purge openjdk-
sudo mkdir -p /usr/local/java
sudo cp -r jdk-7u45-linux-i586.tar.gz /usr/local/java
sudo tar xvzf jdk-7u45-linux-x64.tar.gz
sudo gedit /etc/profile
JAVA_HOME=/usr/local/java/jdk1.7.0_45
JRE_HOME=$JAVA_HOME/jre
PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
export JAVA_HOME
export JRE_HOME
export PATH
sudo update-alternatives –install "/usr/bin/java" "java" "/usr/local/java/jdk1.7.0_45/jre/bin/java" 1
sudo update-alternatives –set java /usr/local/java/jdk1.7.0_45/jre/bin/java
. /etc/profile

Github Setup

Create a new repo in github.com).
In local machine, do git init
In local machine, do git remote add origin gitrepourl
In local machine, do git add .
In local machine, do git commit -m 'First commit'
In local machine, do git git pull
In local machine, do git push
In local machine, do git init

Samba Setup

Download samba using sudo apt-get install samba.
Edit /etc/hosts file and add the ip address of all the other machines and their respective hostnames.
Edit /etc/samba/smb.conf file and add a block to specify the shared drive on each server.
Run the commands smbd restart, and nmbd restart.

Hadoop Setup

Download hadoop distribution and copy it to /usr/local/hadoop and extract it using tar -xvf).
Install ssh using sudo apt-get install ssh
Run bin/hadoop from hadoop root directory. This prints the user manual for hadoop commands.
Create folder input in any desired directory.

0.1 Standalone Operation

cp /usr/local/hadoop/hadoop-3.0.0/etc/hadoop/*.xml input/
/usr/local/hadoop/hadoop-3.0.0/bin/hadoop jar /usr/local/hadoop/hadoop-3.0.0/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.0.0.jar grep input/ output/ 'dfs[a-z.]+'
Check the result using cat output/*
In local machine, do git init

0.2 Fully Distributed Operation

cp /usr/local/hadoop/hadoop-3.0.0/etc/hadoop/*.xml input/

Hadoop Distributed File System (HDFS)

Hardware Failure, Streaming Data Access, Large Data Sets, Simple Coherency Model, Moving Computation is Cheaper than Moving Data, Portability Across Heterogeneous Hardware and Software Platforms

- Is highly fault-tolerant.
- Is designed to be deployed on low-cost (commodity) hardware.
- Provides high throughput access to application data.
- Is suitable for applications that have large data sets.
- Applications that run on HDFS need streaming access to their data sets.
- HDFS is designed more for batch processing rather than interactive use by users.
- A typical file in HDFS is gigabytes to terabytes in size.
- HDFS applications need a write-once-read-many access model for files. A MapReduce application or a web crawler application fits perfectly with this model.
- HDFS provides interfaces for applications to move themselves closer to where the data is located.
- HDFS has been designed to be easily portable from one platform to another.
- The NameNode and Datanodes have built in web servers that makes it easy to check current status of the cluster.
- Metadata, data, namespace, block, replication-factor, master-slave, nodes, rack-awareness, heartbeat, blockreport, balancer, .

HDFS instance may consist of hundreds or thousands of server machines, each storing part of the file system?s data. Some component of HDFS is always non-functional (probably). Therefore, detection of faults and quick, automatic recovery from them is a core architectural goal of HDFS.
A HDFS cluster primarily consists of a NameNode that manages the file system metadata and DataNodes that store the actual data. Clients contact NameNode for file metadata or file modifications and perform actual file I/O directly with the DataNodes.