**HADOOP Installation:**

1. set VM: Ubuntu v18.04, 4 gigs RAM, one full core of CPU, Bridge adaptors

* Minimum Required specifications for installing HADOOP.

2. sudo apt-get update

* Update OS on newly created VM

3. java -version

* check if java exists
* must be same version on all VMs, down to the micro release this is default JDK, check others for other versions. Additionally, cloudra RPM needs Oracle Java RPM due to dependency

4. sudo apt-get install default-jdk

sudo apt-get install default-jre

* install jdk and jre
* installing both the JDK and JRE ensures that you have the complete set of tools and runtime environment necessary to run and develop with Hadoop.

5. sudo groupadd hdp1313

sudo adduser hdp1313 -ingroup hdp1313

sudo adduser hdfs1313 -ingroup hdp1313

sudo usermod -aG hdp1313 root

sudo adduser mprd1313 -ingroup hdp1313

sudo usermod -aG sudo hdfs1313

sudo usermod -aG sudo hdp1313

sudo usermod -aG sudo mprd1313

* Creates a user group hdp1313 and hdp1313, hdfs1313, mprd1313 are added in the hdp1313 group
* Creates hdfs1313 and added to sudo user similarly mprd1313, hdp1313 are added to sudo user
* Creating multiple users and groups primarily related to security, organization, and access control.
* Few users responsible for managing HDFS (hdfs1313), Hadoop administration (hdp1313), MapReduce jobs (mprd1313)
* Adding to Sudo user administrative privileges, Controlled Elevation like can perform high-privilege operations and perform tasks that require elevated permissions.

6. sudo vi /etc/security/limits.conf

* update /etc/security/limits.conf -- as root; add "@hdp1313 hard nofiles 32768"
* Setting the limits to the hdp1313 group on using system resources, by above commands configuring the system to allow members of the hdp1313 group to open up to 32,768 files simultaneously.

7. sysctl -a

sudo vi /etc/sysctl.conf

vm.swappiness=0

vm.overcommit\_memory=1

vm.overcommit\_ratio=50

sudo sysctl -p

* check the Kernal parameters and modifying Kernal parameters using sysctl for adjust Linux Kernal behaviour.
* vm.swappiness=0: Reduces the kernel’s tendency to swap out pages rom physical memory to swap space
* vm.overcommit\_memory=1: Allows the kernel to allocate memory beyond physical RAM limits
* vm.overcommit\_ratio=50: When overcommit memory is set to 2, this parameter allows overcommitting up to 50% of total RAM

8. df -Th

* Check disk format, it must be ext3, ext4, or compatible.

9. which ssh

sudo apt-get install openssh-server

sudo ss -lnp | grep sshd

ssh-keygen -t rsa -b 2048

cat id\_rsa.pub >> authorized\_keys

ssh root@localhost

* check id SSH installed, if not install SSH server.
* Check the ssh ports, generate keys under /root/.ssh. If running as hadoop user, do it with hadoop user and generate ssh keys and set up password less SSH
* By ssh root@localhost to ensure connects passwordlessly

10. wget <https://downloads.apache.org/hadoop/common/hadoop-2.10.2/hadoop-2.10.2.tar.gz>

* Download Hadoop - From under /usr/local

11. tar -zxf hadoop-2.10.2.tar.gz

sudo chown -R root:root hadoop-2.10.2

sudo ln -s hadoop-2.10.2 hadoop

* unpack the downloaded Hadoop in /usr/local

12. update-alternatives --config java

export HADOOP\_HOME=/usr/local/hadoop

export HADOOP\_CONF\_DIR=/usr/local/hadoop/etc/hadoop

export HADOOP\_COMMON\_HOME=/usr/local/hadoop

export HADOOP\_HDFS\_HOME=/usr/local/hadoop

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64

export CLASSPATH=$CLASSPATH:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib

export PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin:.

export HADOOP\_VERSION=2.10.2

* find where is java
* set variables in .bashrc, for this use vi.bashrc command to set environment variables other conf that apply to user.

13.. ~/.bashrc

* source bashrc

14. java -version

hadoop version

* test - check versions

15. mkdir -m 775 /home/hdfs1313/data

chown hdfs1313:hdp1313 /home/hdfs1313/data

mkdir -m 775 -p /hdata/data

chown -R hdfs1313:hdp1313 /hdata

chmod -R 775 /hdata

* create data directories and giving permissions and ownerships recursively.

**Conf HADOOP XML file - /usr/local/hadoop/etc/Hadoop**

16. vi.core-site.xml

<property>

<name>hadoop.tmp.dir</name>

<value>/home/hdfs1313/data</value>

</property>

<property>

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

<!-- NameNode host -->

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

</property>

17. Copy from mapred-site.xml.template to mapred-site.xml first

vi.mapred-site.xml

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

18. vi.hdfs-site.xml

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

<property>

<name>dfs.permission</name>

<value>false</value>

</property>

19. vi.yarn-site.xml

<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>

<property>

<name>yarn.resourcemanager.hostname</name>

<value>localhost</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>localhost:8088</value>

</property>

<property>

<name>yarn.nodemanager.resource.memory-mb</name>

<value>4096</value>

</property>

20. hadoop-env.sh

* export JAVA\_HOME=<use the value used for .bashrc> (e.g., /usr/lib/jvm/java-11-openjdk-amd64)
* JAVA\_HOME in hadoop-env.sh if leave it at its default value- Incorrect Java Version, Inconsistent Environment, Application Failures.

21. hdfs namenode -format

* format namenode, check return code, understand entries
* By formation namenode initialize metadata storage, create metadata files, delete existing metadata, logs and return code.
* By formatting namenode the datanode no direct impact, datanode communication, datanode restart.
* To check the return code using echo $? to confirm successful execution if return 0 success or non-zero value indicates error

22. start-dfs.sh

jps

* Start the hadoop daemons, starts the namenode, datanodes, and secondary namenode processes

23. start-yarn.sh

jps

* Start other hadoop services, start the resource manager, node manager
* If the jps is not show any demons check the Hadoop logs - /usr/local/hadoop/logs

24. hdfs dfs -mkdir /tmp

hdfs dfs -chmod -R 777 /tmp

hdfs dfs -mkdir -p /user/hdp1313

hdfs dfs -chown -R hdp1313:hdp1313 /user/hdp1313

hdfs dfsadmin -refreshUserToGroupsMappings

* /tmp set up with open permissions for temporary file storage and /user/hdp1313 created for user-specific storage with proper ownership
* -refreshUserToGroupsMappings updates HDFS to recognize any changes in user group memberships.

**Prepare for a test job**

24. mkdir -m 755 ~/scripts

curl https://www.gutenberg.org/files/98/98-0.txt > a\_tale\_of\_two\_cities.csv

cat a\_tale\_of\_two\_cities.csv > tmp; cat tmp >> a\_tale\_of\_two\_cities.csv

* in home directory, download the initial data, Inflate the file by running cat command 5-6 times

25. hdfs dfs -mkdir -p /userdata/input

* create hdfs source directory as a root

26. hdfs dfs -put -f ~/scripts/a\_tale\_of\_two\_cities.csv /userdata/input

* copy source file onto hdfs

27. hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop\*examples\*.jar wordcount /userdata/input /userdata/output

* execute the test job and observe the output, save it to file for later reference