We will use following three servers for this guide:

|  |  |  |
| --- | --- | --- |
| **Name** | **IP** | **Purpose** |
| master-node | 172.16.10.171 | Master Node |
| worker-node1 | 172.16.10.172 | Worker Node1 |
| worker-node2 | 172.16.10.173 | Worker Node2 |
| worker-node3 | 172.16.10.173 | Worker Node3 |
| worker-node4 | 172.16.10.174 | Worker Node4 |
| worker-node5 | 172.16.10.175 | Worker Node5 |
| worker-node6 | 172.16.10.176 | Worker Node6 |
| worker-node7 | 172.16.10.177 | Worker Node7 |
| worker-node8 | 172.16.10.178 | Worker Node8 |
| worker-node12 | 172.16.10.182 | Worker Node12 |

You should set hostname on each node using the below command but make sure you replace highlighted text with yours:

sudo hostnamectl set-hostname master-node

sudo hostnamectl set-hostname worker-node1

sudo hostnamectl set-hostname worker-node2

Also, set the correct timezone on each node using the below command:

sudo timedatectl set-timezone Asia/Karachi

**Update Hosts File**

For each node to communicate with each other by name, you need to map the ip addresses against their name.

Edit the /etc/hosts file on each node:

sudo vi /etc/hosts

Remove everything, add your nodes ip and name like below:

172.16.10.171 master-node

172.16.10.172 worker-node1

172.16.10.173 worker-node2

Save and close file when you are finished.

**Adding Hadoop User**

You need a user with sudo privileges for hadoop installation and configuration on each node.  
  
Type below command to create a user called hadoop:

sudo adduser -m hadoop -G wheel

Set hadoop user password with below command:

sudo passwd hadoop

This will prompt you for new password and confirm password.  
  
Make sure you create the same user on each node before moving to next step.

**SSH Key-Pair Authentication**

The master node in hadoop cluster will use an SSH connection to connect to other nodes with key-pair authentication to actively manage the cluster. For this, we need to set up key-pair ssh authentication on each node.

Login to your master-node as the hadoop user, and generate an SSH key like below:

ssh-keygen

This will prompt you for passphrase,  make sure you leave the fields blank.

Repeat the same step on each worker node as the hadoop user. When you are finished generating ssh key-pair on all nodes, move to next step.

Now you need to copy id\_rsa.pub contents to authorized\_keys file and then transfer authorized\_keys to remote node like below:

ssh-copy-id -i ~/.ssh/id\_rsa.pub localhost

scp ~/.ssh/authorized\_keys worker-node1:~/.ssh/

Next, login to worker-node1 as the hadoop user, copy id\_rsa.pub contents to authorized\_keys and then transfer to remove node like below:

ssh-copy-id -i ~/.ssh/id\_rsa.pub localhost

scp ~/.ssh/authorized\_keys worker-node2:~/.ssh/

Next, login to worker-node2, copy id\_rsa.pub contents to authorized\_keys and then transfer to remote node like below:

ssh-copy-id -i ~/.ssh/id\_rsa.pub localhost

scp ~/.ssh/id\_rsa.pub master-node:~/.ssh/

scp ~/.ssh/id\_rsa.pub worker-node1:~/.ssh/

If everything setup correctly as described,  you will be able to connect to each other node via ssh with key-pair authentication without providing password.

**Installing Java**

Hadoop comes with code and scripts that need java to run, you can install latest version of java on each node with below command:

sudo dnf -y install java-latest-openjdk java-latest-openjdk-devel

If you are on CentOS/RHEL 7, install latest java with yum package manager:

sudo yum -y install java-latest-openjdk java-latest-openjdk-devel

**Set Java Home Environment**

Hadoop comes with code and configuration that references the JAVA\_HOME environment variable. This variable points to the java binary file, allowing them to run java code.

You can set up JAVA\_HOME variable on each node like below:

echo "JAVA\_HOME=$(which java)" | sudo tee -a /etc/environment

Reload your system’s environment variables with below command:

source /etc/environment

Verify that variable was set correctly:

echo $JAVA\_HOME

This should return the path to the java binary. Make sure you repeat the same step on each worker node as well.

You need to manually set hadoop binaries location into system path so that default environment understand where to look for hadoop commands.

edit /home/hadoop/.bashrc like below:

vi /home/hadoop/.bashrc

add following lines at the end of the file:

export HADOOP\_HOME=/home/hadoop/hadoop

export PATH=${PATH}:${HADOOP\_HOME}/bin:${HADOOP\_HOME}/sbin

Save and close.  
  
Next, edit /home/hadoop/.bash\_profile:

vi ~/.bash\_profile

add following line at the end of the file:

PATH=/home/hadoop/hadoop/bin:/home/hadoop/hadoop/sbin:$PATH

Save and close file.  
  
Make sure you repeat the same step on each worker node as well.

**Download Hadoop**

At the time of writing this article, hadoop 3.1.3 was the most latest available release.  
  
Login to master-node as the hadoop user, download the Hadoop tarball file, and unzip it:

cd ~

wget http://apache.cs.utah.edu/hadoop/common/current/hadoop-3.1.3.tar.gz

tar -xzf hadoop-3.1.3.tar.gz

mv hadoop-3.1.3 hadoop

**Configure Hadoop**

At this stage, we'll configure hadoop on master-node first, then replicate the configuration to worker nodes later.  
  
On master-node, type below command to find java installation path:

update-alternatives --display java

Take the value of the (link currently points to) and remove the trailing /bin/java. For example on CentOS or RHEL, the link is /usr/lib/jvm/java-11-openjdk-11.0.5.10-2.el8\_1.x86\_64/bin/java, so JAVA\_HOME should be /usr/lib/jvm/java-11-openjdk-11.0.5.10-2.el8\_1.x86\_64.

Edit hadoop-env.sh like below:

cd ~

vi hadoop/etc/hadoop/hadoop-env.sh

Uncomment by removing **#**and update **JAVA\_HOME** line like below:

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

Save and close when you are finished.  
  
Next, edit core-site.xml file to set the NameNode location to master-node on port 9000:

vi hadoop/etc/hadoop/core-site.xml

add the following code, make sure you replace master-node with yours:

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!--

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License. See accompanying LICENSE file.

-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>

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

<value>hdfs://master-node:9000</value>

</property>

</configuration>

Save and close.  
  
Next, edit hdfs-site.conf to resemble the following configuration:

vi hadoop/etc/hadoop/hdfs-site.xml

add following code:

<?xml version="1.0" encoding="UTF-8"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!--

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License. See accompanying LICENSE file.

-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>

<name>dfs.namenode.name.dir</name>

<value>/home/hadoop/data/nameNode</value>

</property>

<property>

<name>dfs.datanode.data.dir</name>

<value>/home/hadoop/data/dataNode</value>

</property>

<property>

<name>dfs.replication</name>

<value>2</value>

</property>

</configuration>

Note that the last property string dfs.replication, indicates how many times data is replicated in the cluster. We set **2**to have all the data duplicated on the two of our worker nodes. If you have only one worker node, enter 1, if you have three, enter 3 but don’t enter a value higher than the actual number of worker nodes you have.

Save and close file when you are finished.  
  
Next, edit the mapred-site.xml file, setting YARN as the default framework for MapReduce operations:

vi hadoop/etc/hadoop/mapred-site.xml

add following code:

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!--

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License. See accompanying LICENSE file.

-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>

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

<value>yarn</value>

</property>

<property>

<name>yarn.app.mapreduce.am.env</name>

<value>HADOOP\_MAPRED\_HOME=$HADOOP\_HOME</value>

</property>

<property>

<name>mapreduce.map.env</name>

<value>HADOOP\_MAPRED\_HOME=$HADOOP\_HOME</value>

</property>

<property>

<name>mapreduce.reduce.env</name>

<value>HADOOP\_MAPRED\_HOME=$HADOOP\_HOME</value>

</property>

<property>

<name>yarn.app.mapreduce.am.resource.mb</name>

<value>512</value>

</property>

<property>

<name>mapreduce.map.memory.mb</name>

<value>256</value>

</property>

<property>

<name>mapreduce.reduce.memory.mb</name>

<value>256</value>

</property>

</configuration>

Save and close.  
  
Next, edit yarn-site.xml, which contains the configuration options for YARN.

vi hadoop/etc/hadoop/yarn-site.xml

add below code, make sure you replace 172.16.10.171 with the your master-node's ip address:

<?xml version="1.0"?>

<!--

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License. See accompanying LICENSE file.

-->

<configuration>

<!-- Site specific YARN configuration properties -->

<property>

<name>yarn.acl.enable</name>

<value>0</value>

</property>

<property>

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

<value>172.16.10.171</value>

</property>

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce\_shuffle</value>

</property>

<property>

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

<value>2048</value>

</property>

<property>

<name>yarn.scheduler.maximum-allocation-mb</name>

<value>2048</value>

</property>

<property>

<name>yarn.scheduler.minimum-allocation-mb</name>

<value>1024</value>

</property>

<property>

<name>yarn.nodemanager.vmem-check-enabled</name>

<value>false</value>

</property>

</configuration>

The last property disables virtual-memory checking which can prevent containers from being allocated properly with openjdk if enabled.

Note: Memory allocation can be tricky on low RAM nodes because default values are not suitable for nodes with less than 8GB of RAM. We have manually set memory allocation for MapReduce jobs, and provide a sample configuration for 4GB RAM nodes.

Save and close.