Hadoop 3 Installation Notes

Generally Hadoop can be run in three modes.

1. **Standalone (or local) mode**: There are no daemons used in this mode. Hadoop uses the local file system as a substitute for HDFS file system. The jobs will run as if there is 1 mapper and 1 reducer.
2. **Pseudo-distributed mode**: All the daemons run on a single machine and this setting mimics the behavior of a cluster. All the daemons run on your machine locally using the HDFS protocol. There can be multiple mappers and reducers.
3. **Fully-distributed mode**: This is how Hadoop runs on a real cluster.

In these notes we will describe how to set up an Hadoop 3 installation to work with. We will set up a *fully-distributed cluster* on your assigned virtual machines.

In the following Section we will see how:

- Setup you cluster's machines to install Hadoop.
- Install Hadoop on your cluster's machines.
- Start Hadoop on your cluster.

Setting up your cluster

1. Prerequisites

On each machine we must ensure that:

- GNU/Linux is supported as a development and production platform.
- Java SDK must be installed. Hadoop 3 requires Java 8. To install the Java 8 distribution and to update the current OS use the following commands:

```
$ sudo apt update
$ sudo apt install openjdk-8-jdk
```

- `ssh` must be installed and `sshd` must be running.

2. Preparation

We will install Hadoop on three (virtual) machines. **Please retrieve and write down the IP addresses of your virtual machines before moving on.** In the following, we will use the following 3 example IP addresses (**change them according to your cluster**):

Sample IP address	Namenode	Datanode	Hostname
172.16.0.1	yes	yes	hadoop-namenode
172.16.0.2	no	yes	hadoop-datanode-2
172.16.0.3	no	yes	hadoop-datanode-3

The first machine will act as the **name node** and as a **data node**, the other two machines will be **data nodes** only.

We will use a dedicated Hadoop user account for running Hadoop applications. While that's not required, it is recommended because it helps to separate the Hadoop installation from other software applications and user accounts running on the same machine (security, permissions, backups, etc). We will perform all required operations as the dedicated Hadoop user.

2.1 Network setup

On every machine in your cluster, execute the following steps.

1. Edit the `/etc/hosts` file with the following command:

   ```
   $ sudo nano /etc/hosts
   ```

   and replace its content with the following:

   ```
   127.0.0.1 localhost
   172.16.0.1 hadoop-namenode
   172.16.0.2 hadoop-datanode-2
   172.16.0.3 hadoop-datanode-3
   ```

2. Edit the `/etc/hostname` file with the corresponding hostname. Open the file with the following command:

   ```
   $ sudo nano /etc/hostname
   ```

   and replace its content with the corresponding hostname. For example, on the machine with IP address `172.16.0.1`, the `/etc/hostname` file should contain the following single line:

   ```
   hadoop-namenode
   ```

3. To change the hostname permanently, run the following commands (one at a time):

```
# cd /etc/one-context.d/
# ls -la
# mv net-15-hostname /root/
```

4. Reboot the machine with the following command:

```
$ sudo reboot
```

2.2 Hadoop user creation

On every machine in your cluster, execute the following steps.

1. Create the `hadoopuser` group, and the `hadoop` user account.

```
$ sudo addgroup hadoopgroup
$ sudo adduser --ingroup hadoopgroup hadoop
$ sudo adduser hadoop sudo
```

   The `hadoop` user is a non-root user, but with `sudo` privileges.

2. Login as the `hadoop` user, and move in your home folder

```
$ sudo su -- hadoop
$ cd
```

   From now on, all operations will be performed as the `hadoop` user.

3. Hadoop requires SSH access to manage its different nodes, i.e., remote machines plus your local machine. The following commands are used to create the `.ssh` folder in your home folder and to setup its access properties:

```
$ mkdir .ssh
$ chmod 700 .ssh
```

   The following commands are used for generating a key value pair using SSH:

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

The first command will create a private/public pair of keys in the `.ssh` folder in your home folder. If asked for passphrase, leave blank and hit return. The second command will append your public key to the list of authorized hosts. The last command will setup the correct file access properties. Check that you can ssh to your **local machine** without a passphrase:

```
$ ssh localhost
<answer YES>
<after login, close the SSH connection>
$ exit
```

Now you can ssh to your local machine without a password/passphrase.

4. **IMPORTANT**: you must make sure that **the name node has a password-less access to the data nodes**. Hence, **on the hadoop-namenode** machine, run the following commands:

```
$ ssh-copy-id -i /home/hadoop/.ssh/id_rsa.pub hadoop@hadoop-datanode-2
$ ssh-copy-id -i /home/hadoop/.ssh/id_rsa.pub hadoop@hadoop-datanode-3
```

These commands **will ask you the passphrase**, to copy the required files to the destination machine. Once this is done, log out with the command `exit` and login again, and you should be able to login without password.

Download and configure Hadoop

1. Download

On every machine in your cluster, execute the following steps.

1. We will install all the software under the `/opt` directory and store HDFS underlying data there as well.
   Use the following command to create the folders with a single command.

```
$ sudo mkdir -p /opt/{hadoop/logs,hdfs/{datanode,namenode},yarn/logs}
$ sudo chown -R hadoop:hadoopgroup /opt
```

The layout of the folder will looks like:

```
/opt
├── hadoop
│   └── logs
├── hdfs
│   ├── datanode
│   └── namenode
└── yarn
    └── logs
```

2. Download hadoop-3.1.3.tar.gz in your folder using the following command:

```
$ wget -c -O ~/hadoop.tar.gz https://archive.apache.org/dist/hadoop/common/hadoop-3.1.3/hadoop-3.1.3.tar.gz
```

3. Decompress the Hadoop package you can use the following command:

```
$ tar -xvf hadoop.tar.gz --directory=/opt/hadoop --exclude=hadoop-3.1.0/share/doc --strip 1
```

To save space, remove the `hadoop.tar.gz` file from your folder:

```
$ rm ~/hadoop.tar.gz
```

4. There are environment settings that will be used by Hadoop. In the `/home/hadoop/.bashrc` file must, please append **at the end** the following lines:

```
export HADOOP_HOME=/opt/hadoop
export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
export LD_LIBRARY_PATH=$HADOOP_HOME/lib/native:$LD_LIBRARY_PATH
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HDFS_NAMENODE_USER=hadoop
export HDFS_DATANODE_USER=hadoop
export HDFS_SECONDARYNAMENODE_USER=hadoop
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_LOG_DIR=$HADOOP_HOME/logs
```

5. Update the `JAVA_HOME` environment variable by also decommenting and updating its entry (`export JAVA_HOME...`) in the `/opt/hadoop/etc/hadoop/hadoop-env.sh` file.

6. Log out and re-login to your `hadoop` account and check Hadoop installation using command:

```
$ hadoop version
```

The output should look similar to the following:

```
Hadoop 3.1.3
Source code repository https://gitbox.apache.org/repos/asf/hadoop.git -r ba631c436b806728f8ec2f54ab1e289526c90579
Compiled by ztang on 2019-09-12T02:47Z
Compiled with protoc 2.5.0
From source with checksum ec785077c385118ac91aadde5ec9799
This command was run using /opt/hadoop/share/hadoop/common/hadoop-common-3.1.3.jar
```

2. Configure the name node

The following steps must be performed **on the machine defined as name node only**, in our case `hadoop-namenode` .

1. Update the `core-site.xml` file located at `/opt/hadoop/etc/hadoop/` to define the name node URI on this machine. The file must look like:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://hadoop-namenode:9820/</value>
  </property>
</configuration>
```

2. Update the `hdfs-site.xml` file located at `/opt/hadoop/etc/hadoop/` to define the path on the local filesystem where the name node stores the namespace and transactions logs persistently. The file must look like:

```
<configuration>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///opt/hdfs/namenode</value>
  </property>
</configuration>
```

3. Update the `yarn-site.xml` file located at `/opt/hadoop/etc/hadoop` . The file must look like:

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>hadoop-namenode</value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
  </property>
  <property>
    <name>yarn.nodemanager.local-dirs</name>
    <value>file:///opt/yarn/local</value>
  </property>
  <property>
    <name>yarn.nodemanager.log-dirs</name>
    <value>file:///opt/yarn/logs</value>
  </property>
</configuration>
```

4. Update the `mapred-site.xml` file located at `/opt/hadoop/etc/hadoop`. The file must look like:

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.address</name>
    <value>hadoop-namenode:10020</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.webapp.address</name>
    <value>hadoop-namenode:19888</value>
  </property>
  <property>
    <name>mapreduce.jobhistory.intermediate-done-dir</name>
    <value>/mr-history/tmp</value>
  </property>
  <property>
```

```
    <name>mapreduce.jobhistory.done-dir</name>
    <value>/mr-history/done</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
</configuration>
```

5. Format the name node:

```
$ hdfs namenode -format
```

6. Add the data nodes to the `workers` file located in `/opt/hadoop/etc/hadoop`. The file must look like:

```
172.16.0.1
172.16.0.2
172.16.0.3
```

3. Configure the data nodes

The following steps must be performed **on each machine defined as data node**, in our case `hadoop-namenode`, `hadoop-datanode-2`, and `hadoop-datanode-3`.

1. Update the `core-site.xml` file located a `/opt/hadoop/etc/hadoop/` to define the name node URI on this machine. The file must look like:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://hadoop-namenode:9820/</value>
  </property>
</configuration>
```

2. Update the `hdfs-site.xml` file located a `/opt/hadoop/etc/hadoop/` to define the data node parameters on this machine. The file must look like:

```
<configuration>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:///opt/hdfs/datanode</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>
  <property>
    <name>dfs.permissions</name>
    <value>false</value>
  </property>
  <property>
    <name>dfs.datanode.use.datanode.hostname</name>
    <value>false</value>
  </property>
</configuration>
```

3. Update the `yarn-site.xml` file located at `/opt/hadoop/etc/hadoop` . The file must look like:

```
<configuration>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>hadoop-namenode</value>
  </property>
</configuration>
```

4. Update the `mapred-site.xml` file located at `/opt/hadoop/etc/hadoop` . The file must look like:

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

```
  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=/opt/hadoop</value>
  </property>
</configuration>
```

4. Understanding the Hadoop memory allocation

Memory allocation can be complex on low RAM nodes because default values are not suitable for nodes with less than 8 GB of RAM. In this section we highlight how memory allocation works for MapReduce jobs, and provide a sample configuration for 2 GB RAM nodes.

The memory allocation properties

A YARN job is executed with two kinds of resources:

- An **application master** (AM), which is responsible for monitoring the application and coordinating distributed executors in the cluster.
- Some **executors**, that are created by the AM to actually run the job. For a MapReduce job, they will perform map or reduce operation, in parallel.

Both are run in **containers** on **worker nodes**. Each worker node runs a **NodeManager** daemon that is responsible for container creation on the node. The whole cluster is managed by a **ResourceManager** that schedules container allocation on all the worker nodes, depending on capacity requirements and current charge.

Four types of resource allocations need to be configured properly for the cluster to work. These are:

1. *How much memory can be allocated for YARN containers on a single node.* This limit should be higher than all the others; otherwise, container allocation will be rejected and applications will fail. However, it should not be the entire amount of RAM on the node. This value is configured in the `yarn-site.xml` file with the `yarn.nodemanager.resource.memory-mb` property.

2. *How much memory a single container can consume and the minimum memory allocation allowed.* A container will never be bigger than the maximum, or else allocation will fail and will always be allocated as a multiple of the minimum amount of RAM. Those values are configured in the `yarn-site.xml` file with the `yarn.scheduler.maximum-allocation-mb` and `yarn.scheduler.minimum-allocation-mb` properties.

3. *How much memory will be allocated to the ApplicationMaster.* This is a constant value that should fit in the container maximum size. This value is configured in the `mapred-site.xml` with the `yarn.app.mapreduce.am.resource.mb` property.

4. *How much memory will be allocated to each map or reduce operation.* This should be less than the maximum size. This value is configured in the `mapred-site.xml` file with the `mapreduce.map.memory.mb` and `mapreduce.reduce.memory.mb` properties.

The relationship between all those properties can be seen in the following figure:

For 2 GB nodes, a working configuration may be:

Property	Value
`yarn.nodemanager.resource.memory-mb`	1536
`yarn.scheduler.maximum-allocation-mb`	1536
`yarn.scheduler.minimum-allocation-mb`	128
`yarn.app.mapreduce.am.resource.mb`	512
`mapreduce.map.memory.mb`	256
`mapreduce.reduce.memory.mb`	256

According to the previous table, **on each machine** you should update the configuration files as follows.

1. Update the `yarn-site.xml` file located at `/opt/hadoop/etc/hadoop` by adding the following lines in the `configuration` element:

```
<property>
  <name>yarn.nodemanager.resource.memory-mb</name>
  <value>1536</value>
</property>
<property>
  <name>yarn.scheduler.maximum-allocation-mb</name>
  <value>1536</value>
</property>
<property>
  <name>yarn.scheduler.minimum-allocation-mb</name>
  <value>128</value>
</property>
<property>
  <name>yarn.nodemanager.vmem-check-enabled</name>
  <value>false</value>
</property>
```

The last property disables virtual memory checking which can prevent containers from being allocated properly with Java 8 if enabled.

2. Update the `mapred-site.xml` file located at `/opt/hadoop/etc/hadoop` by adding the following lines in the `configuration` element:

```
<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>
```

Start and test Hadoop

After finishing the steps above, we must execute the following commands:

1. To start the name node, data nodes and secondary name node, **from the name node** we must execute the following command:

```
start-dfs.sh
```

   You should get an output similar to the following:

```
Starting namenodes on [hadoop-namenode]
Starting datanodes
Starting secondary namenodes [hadoop-namenode]
```

2. To start the resource manager and node managers, **from the name node** we must execute the following command:

```
start-yarn.sh
```

   You should get an output similar to the following:

```
Starting resourcemanager
Starting nodemanagers
```

3. After these two steps, to ensure that Hadoop started successfully, we must run the `jps` command on name node and data nodes. The command must give the following output (process ids can be different): - on the name node: `bash 28403 DataNode 28675 SecondaryNameNode 29460 Jps 28919 ResourceManager 28183 NameNode 29308 NodeManager` - on the data nodes `bash 25721 Jps 25451 DataNode 25644 NodeManager` You may check logs at `/opt/hadoop/logs` on the 3 machines and check if everything is alright, or running the `hdfs dfsadmin -report` command (it must return `Live datanodes (3)`).

4. You can access Hadoop on a browser on your local machine (use IP addresses, not hostnames): - namenode: `http://172.16.0.1:9870/` - resource manager: `http://172.16.0.1:8088/`

5. Run an example provided with Hadoop. To do so, run the following instructions **from a node of your choice in your cluster**:

```
hadoop fs -mkdir /user
hadoop fs -mkdir /user/hadoop
hadoop fs -put /opt/hadoop/etc/hadoop/ input
hadoop jar /opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.1.3.jar grep /user/hadoop/input/*.xml /user/hadoop/output 'dfs[a-z.]+'
hadoop fs -cat output/part-r-00000
```

6. To stop the name node, data nodes and secondary name node, the resource manager and node managers, **from the name node** we must execute the following commands:

```
stop-yarn.sh
stop-dfs.sh
```

   You should get an output similar to the following:

```
Stopping resourcemanager
Stopping nodemanagers
Stopping namenodes on [hadoop-namenode]
Stopping datanodes
Stopping secondary namenodes [hadoop-namenode]
```

Troubleshooting

If you get an error like the following:

```
[2020-01-14 08:48:28.567]Container [pid=155967,containerID=container_1578991625193_0002_01_000023] is running 380426752B beyond the 'VIRTUAL' memory limit. Current usage: 15
```

you are using more virtual memory than your current limit of 2.1 Gb. This can be resolved in two ways:

1. **Disable Virtual Memory Limit Checking**
   YARN will simply ignore the limit; in order to do this, add this to your `yarn-site.xml` *on each machine*:

   ```
   <property>
     <name>yarn.nodemanager.vmem-check-enabled</name>
     <value>false</value>
   </property>
   ```

   The default for this setting is `true`.

2. **Increase Virtual Memory to Physical Memory Ratio**
   In your `yarn-site.xml` change this to a higher value than is currently set, *on each machine*:

   ```
   <property>
     <name>yarn.nodemanager.vmem-pmem-ratio</name>
     <value>5</value>
   </property>
   ```

   The default is 2.1.
   You could also increase the amount of physical memory you allocate to a container.
   Make sure you don't forget to restart yarn after you change the configuration.