

INSTALLATION OF HADOOP SINGLE NODE ON UBUNTU

Prerequisites

- Access to a terminal window/command line
- Sudo or root privileges on local /remote machines

Install OpenJDK on Ubuntu

The Hadoop framework is written in Java, and its services require a compatible Java Runtime Environment (JRE) and Java Development Kit (JDK).

Use the following command to update your system before initiating a new installation:

```
sudo apt update
```

Note: Apache Hadoop 3.x fully supports Java 8

Type the following command in your terminal to install OpenJDK 8:

```
sudo apt install openjdk-8-jdk -y
```

Once the installation process is complete, verify the current Java version:

```
java -version; javac -version
```

```
onap@onap-VirtualBox:~$ java -version;javac -version
openjdk version "1.8.0_252"
OpenJDK Runtime Environment (build 1.8.0_252-8u252-b09-1~18.04-b09)
OpenJDK 64-Bit Server VM (build 25.252-b09, mixed mode)
javac 1.8.0_252
```

Set Up a Non-Root User for Hadoop Environment

Advisable to create a non-root user, specifically for the Hadoop environment.

To ensure the smooth functioning of Hadoop services, the user should have the ability to establish a [passwordless SSH connection](#) with the localhost

Install OpenSSH on Ubuntu

```
sudo apt install openssh-server openssh-client -y
```

Create Hadoop User

```
sudo adduser hdoop
```

username, in this example, is **hdoop**. You are free to use any username and password

```
su - hdoop
```

Enable Passwordless SSH for Hadoop User

[Generate an SSH key pair](#) and define the location it is to be stored in:

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

```
hdoop@pnap-VirtualBox:~$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /home/hdoop/.ssh/id_rsa.
Your public key has been saved in /home/hdoop/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:9GEXT7rDLjtcp5dT4KE0LevsYzloAKxir/E0zPjP58Y hdoop@pnap-VirtualBox
The key's randomart image is:
+----[RSA 2048]-----+
|          .  .  |
|          =     |
|       . . 0 0 . |
|      0. 0 ++. + |
|       . S ..+* 0 |
|    0+.   . +.0 . |
| .00=   . 0.=.+ 0 |
|   =.0  E =00 +  |
|   ..0.0+...+. + |
+-----[SHA256]-----+
hdoop@pnap-VirtualBox:~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
hdoop@pnap-VirtualBox:~$ chmod 0600 ~/.ssh/authorized_keys
```

Use the **cat** command to store the public key as **authorized_keys** in the *ssh* directory:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Set the permissions for your user with the **chmod** command:

```
chmod 0600 ~/.ssh/authorized_keys
```

The new user is now able to SSH without needing to enter a password every time. Verify everything is set up correctly by using the **hdoop** user to SSH to localhost:

```
ssh localhost
```

After an initial prompt, the Hadoop user is now able to establish an SSH connection to the localhost seamlessly.

Download and Install Hadoop on Ubuntu

The steps outlined in this tutorial use the Binary download for **Hadoop Version 3.2.1**.

```
wget https://downloads.apache.org/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz
```

```
hadoop@pnap-VirtualBox:~$ wget https://downloads.apache.org/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz
--2020-04-22 09:28:51-- https://downloads.apache.org/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz
Resolving downloads.apache.org (downloads.apache.org)... 88.99.95.219, 2a01:4f8:10a:201a::2
Connecting to downloads.apache.org (downloads.apache.org)|88.99.95.219|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 359196911 (343M) [application/x-gzip]
Saving to: 'hadoop-3.2.1.tar.gz'

hadoop-3.2.1.tar.gz  100%[=====] 342.56M  10.2MB/s   in 31s

2020-04-22 09:29:23 (10.9 MB/s) - 'hadoop-3.2.1.tar.gz' saved [359196911/359196911]
```

Extract the files to initiate the Hadoop installation:

```
tar xzf hadoop-3.2.1.tar.gz
```

The Hadoop binary files are now located within the *hadoop-3.2.1* directory.

Single Node Hadoop Deployment (Pseudo-Distributed Mode)

A Hadoop environment is configured by editing a set of configuration files:

- `bashrc`
- `hadoop-env.sh`
- `core-site.xml`
- `hdfs-site.xml`
- `mapred-site.xml`
- `yarn-site.xml`

Configure Hadoop Environment Variables (`bashrc`)

Edit the `.bashrc` shell configuration file using a text editor of your choice (we will be using `nano`):
(In the terminal, outside the Hadoop directory)

```
nano .bashrc
```

Define the Hadoop environment variables by adding the following content to the end of the file:

```
#Hadoop Related Options

export HADOOP_HOME=/home/hadoop/hadoop-3.2.1

export HADOOP_INSTALL=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_HDFS_HOME=$HADOOP_HOME
```

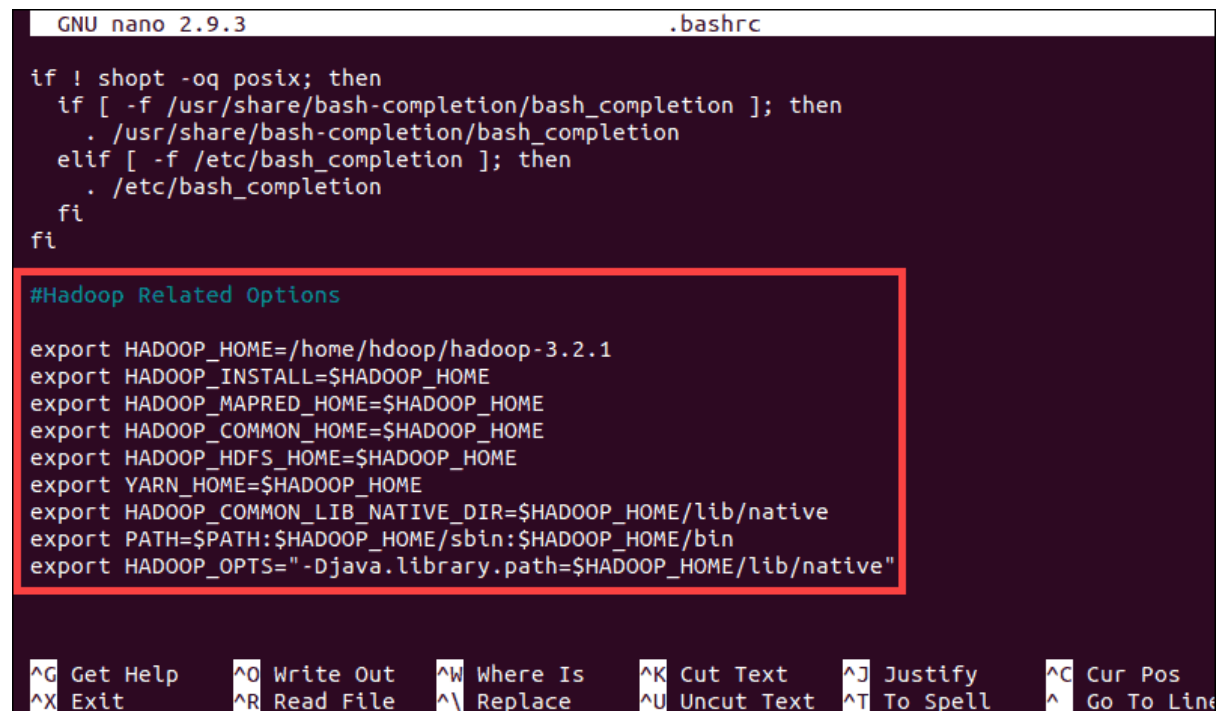
```
export YARN_HOME=$HADOOP_HOME

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

Once you add the variables, save and exit the `.bashrc` file. Press Ctrl+X to exit and save the file.



```
GNU nano 2.9.3 .bashrc

if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

#Hadoop Related Options

export HADOOP_HOME=/home/hdooop/hadoop-3.2.1
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

It is vital to apply the changes to the current running environment by using the following command:

```
source ~/.bashrc
```

The `hadoop-env.sh` file serves as a master file to configure YARN, [HDFS](#), [MapReduce](#), and Hadoop-related project settings.

When setting up a **single node Hadoop cluster**, you need to define which Java implementation is to be utilized. Use the previously created `$HADOOP_HOME` variable to access the `hadoop-env.sh` file:

```
nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

Uncomment the `$JAVA_HOME` variable (i.e., remove the `#` sign) and add the full path to the OpenJDK installation on your system. If you have installed the same version as presented in the first part of this tutorial, add the following line:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

The path needs to match the location of the Java installation on your system.

```
GNU nano 2.9.3 /home/hdoop/hadoop-3.2.1/etc/hadoop/hadoop-env.sh

###

# Technically, the only required environment variable is JAVA_HOME.
# All others are optional.  However, the defaults are probably not
# preferred.  Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use.  By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

# Location of Hadoop.  By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=

# Location of Hadoop's configuration information.  i.e., where this
# file is living.  If this is not defined, Hadoop will attempt to
# locate it based upon its execution path.
#

# NOTE: It is recommend that this variable not be set here but in
# /etc/profile.d or equivalent.  Some options (such as

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^J Justify      ^C Cur Pos
^X Exit          ^R Read File    ^\ Replace      ^U Uncut Text  ^T To Linter    ^_ Go To Line
```

To Locate the Java path:

If you need help to locate the correct Java path, run the following command in your terminal window:

```
which javac
```

The resulting output provides the path to the Java binary directory.

```
hdoop@pnap-VirtualBox:~$ which javac
/usr/bin/javac
```

Use the provided path to find the OpenJDK directory with the following command:

```
readlink -f /usr/bin/javac
```

The section of the path just before the `/bin/javac` directory needs to be assigned to the `$JAVA_HOME` variable.

```
hdoop@pnap-VirtualBox:~$ readlink -f /usr/bin/javac
/usr/lib/jvm/java-8-openjdk-amd64/bin/javac
```

Edit core-site.xml File

The `core-site.xml` file defines HDFS and Hadoop core properties.

To set up Hadoop in a pseudo-distributed mode, you need to **specify the URL** for your NameNode, and the temporary directory Hadoop uses for the map and reduce process.

Open the *core-site.xml* file in a text editor:

```
nano $HADOOP_HOME/etc/hadoop/core-site.xml
```

Add the following configuration to override the default values for the temporary directory and add your HDFS URL to replace the default local file system setting:

```
<configuration>

<property>

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

  <value>/home/hadoop/tmpdata</value>

</property>

<property>

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

  <value>hdfs://127.0.0.1:9000</value>

</property>
</configuration>
```

This example uses values specific to the local system. You should use values that match your systems requirements. The data needs to be consistent throughout the configuration process.

```
GNU nano 2.9.3 /home/hdoop/hadoop-3.2.1/etc/hadoop/core-site.xml

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>hadoop.tmp.dir</name>
  <value>/home/hdoop/tmpdata</value>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
</configuration>

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell   ^_ Go To Line
```

Do not forget to [create a Linux directory](#) in the location you specified for your temporary data.\$
\$**mkdir tmpdata**

Edit hdfs-site.xml File

The properties in the *hdfs-site.xml* file govern the location for storing node metadata, fsimage file, and edit log file. Configure the file by defining the **NameNode** and **DataNode storage directories**.

Create two new directories for Hadoop to store the Namenode and Datanode information.

```
$ mkdir -p ~/dfsdata/namenode ~/dfsdata/datanode
```

Set the **dfs.replication** value to **1** to match the single node setup.

Use the following command to open the *hdfs-site.xml* file for editing:

```
nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml
```

Add the following configuration to the file and, if needed, adjust the NameNode and DataNode directories to your custom locations:

```
<configuration>

<property>

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

```

<value>/home/hadoop/dfsdata/namenode</value>

</property>

<property>

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

  <value>/home/hadoop/dfsdata/datanode</value>

</property>

<property>

  <name>dfs.replication</name>

  <value>1</value>

</property>
</configuration>

```

```

GNU nano 2.9.3 /home/hadoop/hadoop-3.2.1/etc/hadoop/hdfs-site.xml

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.data.dir</name>
  <value>/home/hadoop/dfsdata/namenode</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/hadoop/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
</configuration>

^G Get Help  ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify    ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell   ^_ Go To Line

```

Edit mapred-site.xml File

Use the following command to access the *mapred-site.xml* file and **define MapReduce values**:

```
nano $HADOOP_HOME/etc/hadoop/mapred-site.xml
```

Add the following configuration to change the default MapReduce framework name value to **yarn**:

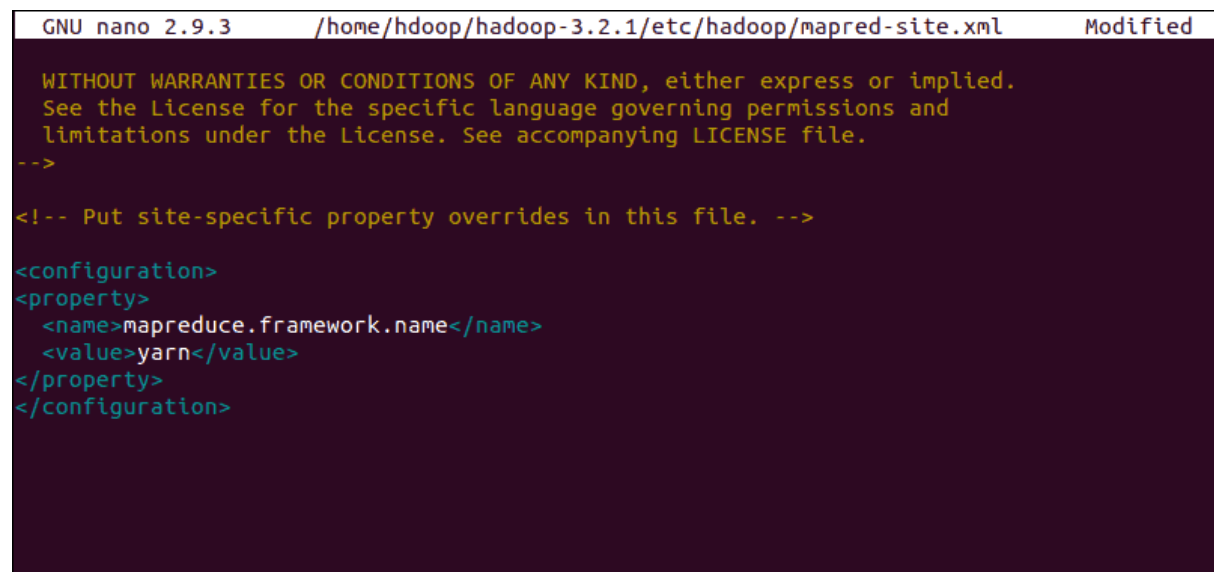

```
<configuration>

<property>

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

  <value>yarn</value>

</property>
</configuration>
```



```
GNU nano 2.9.3 /home/hdoop/hadoop-3.2.1/etc/hadoop/mapred-site.xml Modified

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>
</configuration>
```

Edit yarn-site.xml File

The *yarn-site.xml* file is used to define settings relevant to **YARN**. It contains configurations for the **Node Manager**, **Resource Manager**, **Containers**, and **Application Master**.

Open the *yarn-site.xml* file in a text editor:

```
nano $HADOOP_HOME/etc/hadoop/yarn-site.xml
```

Append the following configuration to the file:

```
<configuration>

<property>

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

  <value>mapreduce_shuffle</value>
```

```
</property>

<property>

  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>

  <value>org.apache.hadoop.mapred.ShuffleHandler</value>

</property>

<property>

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

  <value>127.0.0.1</value>

</property>

<property>

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

  <value>0</value>

</property>

<property>

  <name>yarn.nodemanager.env-whitelist</name>

  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLA
SSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOME</value>

</property>
</configuration>
```

Format HDFS NameNode

It is important to **format the NameNode** before starting Hadoop services for the first time:

```
hdfs namenode -format
```

The shutdown notification signifies the end of the NameNode format process.

```

hadoop@pnap-VirtualBox:~$ hdfs namenode -format
WARNING: /home/hadoop/hadoop-3.2.1/logs does not exist. Creating.
2020-04-23 06:08:13,322 INFO namenode.NameNode: STARTUP_MSG:
/*****
STARTUP_MSG: Starting NameNode
STARTUP_MSG: host = pnep-VirtualBox/127.0.1.1
STARTUP_MSG: args = [-format]
STARTUP_MSG: version = 3.2.1
STARTUP_MSG: classpath = /home/hadoop/hadoop-3.2.1/etc/hadoop:/home/hadoop/hadoop-3.2.1/
hare/hadoop/common/lib/httpcore-4.4.10.jar:/home/hadoop/hadoop-3.2.1/share/hadoop/common/
ib/curator-recipes-2.13.0.jar:/home/hadoop/hadoop-3.2.1/share/hadoop/common/lib/kerby-asn
2020-04-23 06:08:17,966 INFO namenode.NNStorageRetentionManager: Going to retain 1 images
with txid >= 0
2020-04-23 06:08:18,036 INFO namenode.FSImage: FSImageSaver clean checkpoint: txid=0 wher
meet shutdown.
2020-04-23 06:08:18,036 INFO namenode.NameNode: SHUTDOWN_MSG:
/*****
SHUTDOWN_MSG: Shutting down NameNode at pnep-VirtualBox/127.0.1.1
*****/

```

Start Hadoop Cluster

Navigate to the *hadoop-3.2.1/sbin* directory and execute the following commands to start the NameNode and DataNode:

```
./start-dfs.sh
```

The system takes a few moments to initiate the necessary nodes.

```

hadoop@pnep-VirtualBox:~/hadoop-3.2.1/sbin$ ./start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [pnep-VirtualBox]

```

Once the namenode, datanodes, and secondary namenode are up and running, start the YARN resource and nodemanagers by typing:

```
./start-yarn.sh
```

As with the previous command, the output informs you that the processes are starting.

```

hadoop@pnep-VirtualBox:~/hadoop-3.2.1/sbin$ ./start-yarn.sh
Starting resourcemanager
Starting nodemanagers

```

Type this simple command to check if all the daemons are active and running as Java processes:

```
jps
```

If everything is working as intended, the resulting list of running Java processes contains all the HDFS and YARN daemons.

```
hadoop@pnep-VirtualBox:~/hadoop-3.2.1/sbin$ jps
469 DataNode
742 SecondaryNameNode
32759 NameNode
31180 NodeManager
31020 ResourceManager
988 Jps
```

To shutdown the Hadoop Cluster

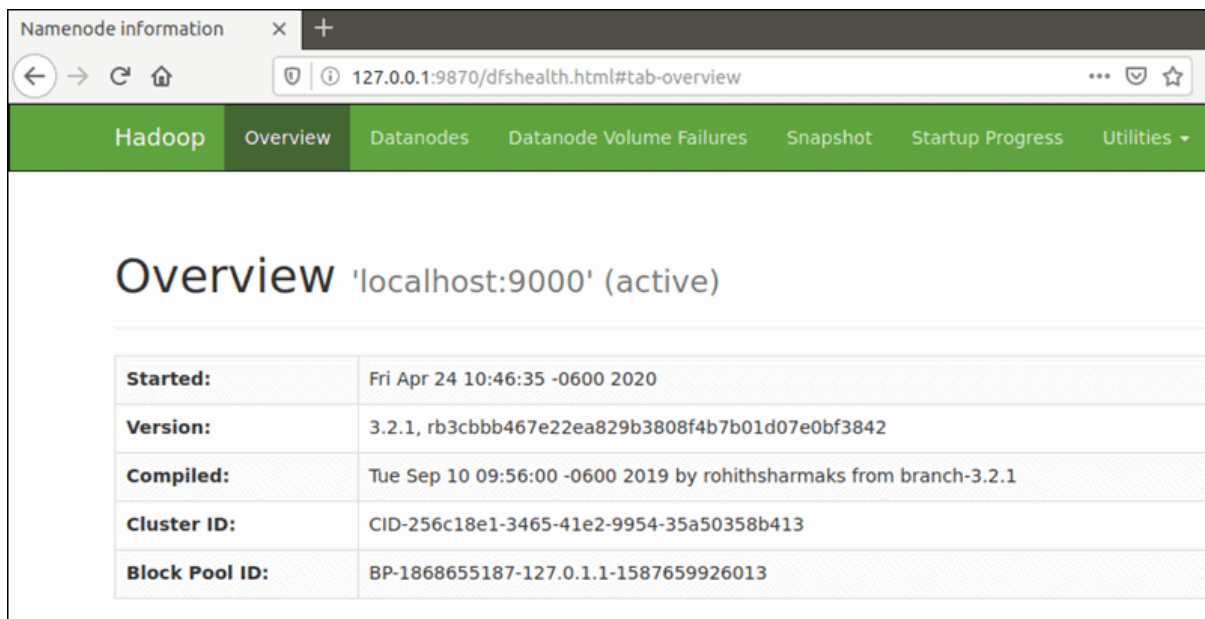
```
$. /stop-all.sh
```

Access Hadoop UI from Browser

Use your preferred browser and navigate to your localhost URL or IP. The default port number **9870** gives you access to the Hadoop NameNode UI:

```
http://localhost:9870
```

The NameNode user interface provides a comprehensive overview of the entire cluster.



The screenshot shows a web browser window with the title 'Namenode information'. The address bar displays '127.0.0.1:9870/dfshealth.html#tab-overview'. The page has a green navigation bar with tabs: 'Hadoop', 'Overview' (selected), 'Datanodes', 'Datanode Volume Failures', 'Snapshot', 'Startup Progress', and 'Utilities'. The main content area is titled 'Overview 'localhost:9000' (active)'. Below the title is a table with the following information:

Started:	Fri Apr 24 10:46:35 -0600 2020
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled:	Tue Sep 10 09:56:00 -0600 2019 by rohithsharmaks from branch-3.2.1
Cluster ID:	CID-256c18e1-3465-41e2-9954-35a50358b413
Block Pool ID:	BP-1868655187-127.0.1.1-1587659926013

The default port **9864** is used to access individual DataNodes directly from your browser:

```
http://localhost:9864
```

Cluster ID:	CID-256c18e1-3465-41e2-9954-35a50358b413
Version:	3.2.1, rb3cbbb467e22ea829b3808f4b7b01d07e0bf3842

Block Pools

Namenode Address	Block Pool ID	Actor State	Last Heartbeat	Last Block Report	Last Block Report Size (Max Size)
localhost:9000	BP-1868655187-127.0.1.1-1587659926013	RUNNING	0s	30 minutes	0 B (64 MB)

The YARN Resource Manager is accessible on port **8088**:

<http://localhost:8088>

The Resource Manager is an invaluable tool that allows you to monitor all running processes in your Hadoop cluster.

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved
0	0	0	0	0	0 B	8 GB	0 B

Cluster Nodes Metrics

Active Nodes	Decommissioning Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes
1	0	0	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
Capacity Scheduler	[memory-mb (unit=Mi), vcores]	<memory:1024, vCores:1>	<memory:8192, vCores:4>

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCoers	Allocated Memory MB	Reserved CPU VCoers
No data available in table														

Showing 0 to 0 of 0 entries

EXECUTION OF SPECIFIC WORDCOUNT JAR FILE

1. Create a directory “/wcinput” in HDFS

`/home/hdoop/hadoop-3.2.1/bin$ hdfs dfs -mkdir /wcinput`

2. Assume your input file is in “/home/hdoop/Downloads/fsample/sample.txt”

3. To copy the input files into the distributed file system use the below command

`/home/hdoop/hadoop-3.2.1/bin$ hdfs dfs -put /home/hdoop/Downloads/fsample/* /wcinput`

4. Set the environment variables as follows in the Hadoop Environment Configuration file:

```
nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh
```

```
export PATH=/usr/lib/jvm/java-8-openjdk-amd64/bin:${PATH}
export HADOOP_CLASSPATH=/usr/lib/jvm/java-8-openjdk-amd64/lib/tools.jar
```

Note:

Before you start compiling the program

- Make sure environment variables set right as given.
- Input files should be pushed inside HDFS.
- Both Input and Output File path should be HDFS path.
- Java file to be compiled should be in ../Hadoop-3.2.1/bin directory

5. Compile WordCount.java:

```
$ bin/hadoop com.sun.tools.javac.Main WordCount.java
```

```
hadoop@mercy-HP-Notebook:~/hadoop-3.2.1/bin$ hadoop com.sun.tools.javac.Main WordCount.java
/home/hadoop/hadoop-3.2.1/libexec/hadoop-functions.sh: line 2366: HADOOP_COM.SUN.TOOLS.JAVAC.MAIN_USE
R: invalid variable name
/home/hadoop/hadoop-3.2.1/libexec/hadoop-functions.sh: line 2461: HADOOP_COM.SUN.TOOLS.JAVAC.MAIN_OPT
S: invalid variable name
hadoop@mercy-HP-Notebook:~/hadoop-3.2.1/bin$ ls
container-executor  SalesCountryReducer.java
hadoop              SalesMapper.java
hadoop.cmd          test-container-executor
hdfs                'WordCount$IntSumReducer.class'
hdfs.cmd            'WordCount$TokenizerMapper.class'
mapred              WordCount.class
mapred.cmd          WordCount.java
oom-listener        yarn
SalesCountry        yarn.cmd
SalesCountryDriver.java
```

6. Create a jar

```
$ jar cf wc.jar WordCount*.class
```

```
hadoop@mercy-HP-Notebook:~/hadoop-3.2.1/bin$ jar cf wc.jar WordCount*.class
hadoop@mercy-HP-Notebook:~/hadoop-3.2.1/bin$ ls
container-executor  oom-listener  'WordCount$IntSumReducer.class'
hadoop              SalesCountry  'WordCount$TokenizerMapper.class'
hadoop.cmd          SalesCountryDriver.java  WordCount.class
hdfs                SalesCountryReducer.java  WordCount.java
hdfs.cmd            SalesMapper.java  yarn
mapred              test-container-executor  yarn.cmd
mapred.cmd          wc.jar
```

Run the application:

Input File Path: /wcinput

Output File Path: /wcoutput1

(Note: The out directory will be created and output will be saved in parts)

```
$ bin/hadoop jar wc.jar WordCount /wcinput /wcoutput1/out
```

Output:

List the output files on the distributed file system `/home/hadoop/hadoop-3.2.1/bin$ hdfs dfs -ls/wcoutput1/out`

```
hadoop@mercy-HP-Notebook:~/hadoop-3.2.1/bin$ hdfs dfs -ls /wcoutput1/out
Found 2 items
-rw-r--r--  1 hadoop supergroup          0 2021-03-17 00:51 /wcoutput1/out/_SUCCESS
-rw-r--r--  1 hadoop supergroup       205 2021-03-17 00:51 /wcoutput1/out/part-r-000000
```

View the output files on the distributed file system `/home/hadoop/hadoop-3.2.1/bin$ hdfs dfs -cat /wcoutput1/out/*`

```
hadoop@mercy-HP-Notebook:~/hadoop-3.2.1/bin$ hdfs dfs -cat /wcoutput1/out/part-r-000000
2021-03-17 01:10:33,087 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
Big      1
Data     1
Python   2
Sets     1
a        1
and      1
another  1
are      1
common   1
easily   1
exist    1
functionality  1
in       2
interpretable  1
is       2
language.  1
of       1
piece    1
processing.  1
```