

Assignment-1

Big Data Programming

Report on Set-up of Hadoop and Running Word count example

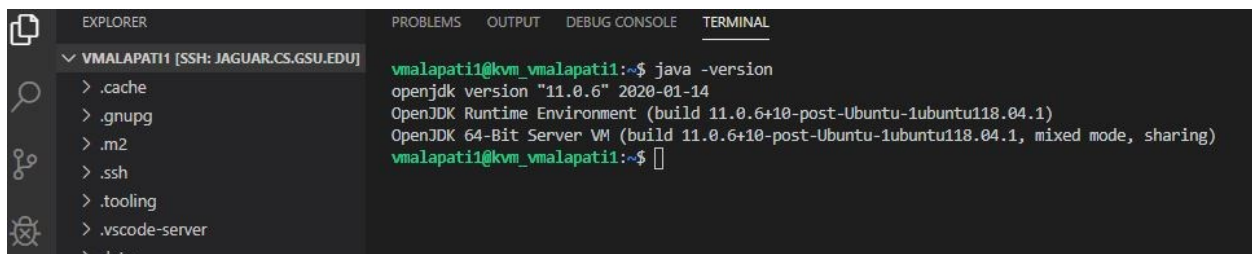
Setting up Hadoop and running word-count example consists of different steps, All these steps as follow:

1.a: Most of the Hadoop core is developed using Java, so before setting up Hadoop Java need to be installed in the local machine or server.

Java Installation:

```
$ sudo apt update  
$ sudo apt install default-jdk
```

Since Java is already Installed in my Jaguar server, below is the image with jaguar server user and Java version



The screenshot shows a terminal window with the following output:

```
vmalapati1@kvm_vmalapati1:~$ java -version  
openjdk version "11.0.6" 2020-01-14  
OpenJDK Runtime Environment (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1)  
OpenJDK 64-Bit Server VM (build 11.0.6+10-post-Ubuntu-1ubuntu118.04.1, mixed mode, sharing)  
vmalapati1@kvm_vmalapati1:~$
```

1.b: After Java is installed and updated install hadoop using command line since here Ubuntu and server is used. There are different versions of hadoop can be download from apache hadoop website based on the requirement. In this class we downloaded 3.2.1 version.

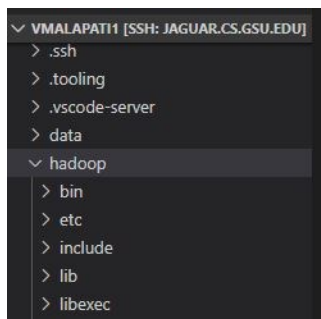
Hadoop Installation and extracting it into folder named hadoop:

```
$ wget http://www-us.apache.org/dist/hadoop/common/hadoop-3.0.3/hadoop-3.0.3.tar.gz
```

```
$ tar -xvf hadoop-3.2.1.tar.gz -C ~/
```

```
$ mv hadoop-3.2.1/ hadoop
```

A folder with hadoop is created shown below.



Once Downloading is done, extract the files using tar and place them in a new folder named hadoop.

1.c:The Hadoop don't where the java is located, it requires to set the environmental path variable or in config files. To complete this the path of java is added in the hadoop-env.sh file which is located at hadoop/etc/hadoop.

Adding java path in config files:

```
export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64/
```

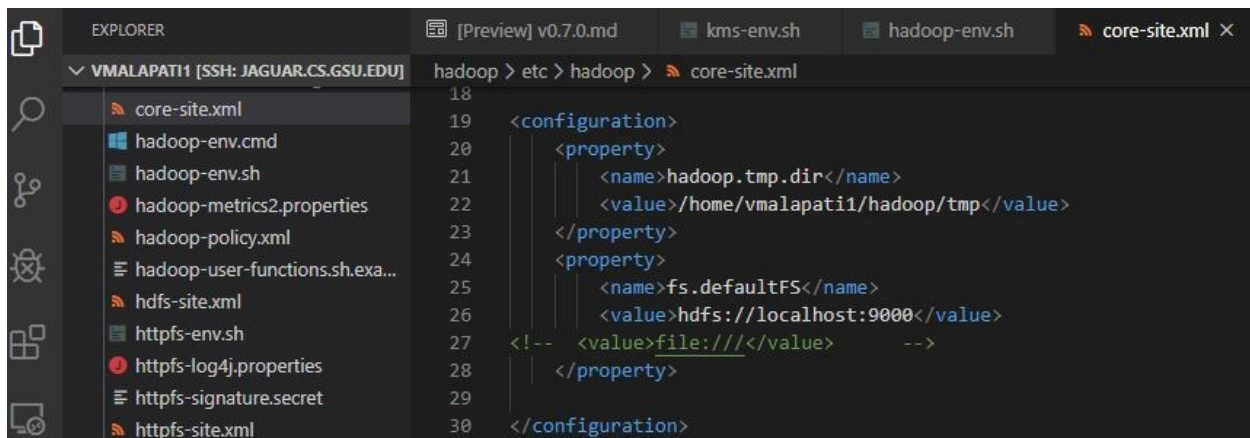


Fig.Added java path in hadoop config

1.d: Hadoop configuration is set up by Editing the core-site file in hadoop/etc/hadoop, it is a Hadoop core configuration setting file in xml for input output and other operations. In this core-site file the path for above folder is given which is used for hdfs.

The following code is added :

```
<configuration>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>/home/rob/hadoop/tmp</value>
  </property>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://localhost:9000</value>
  <!-- <value>file:///</value> -->
  </property>
</configuration>
```

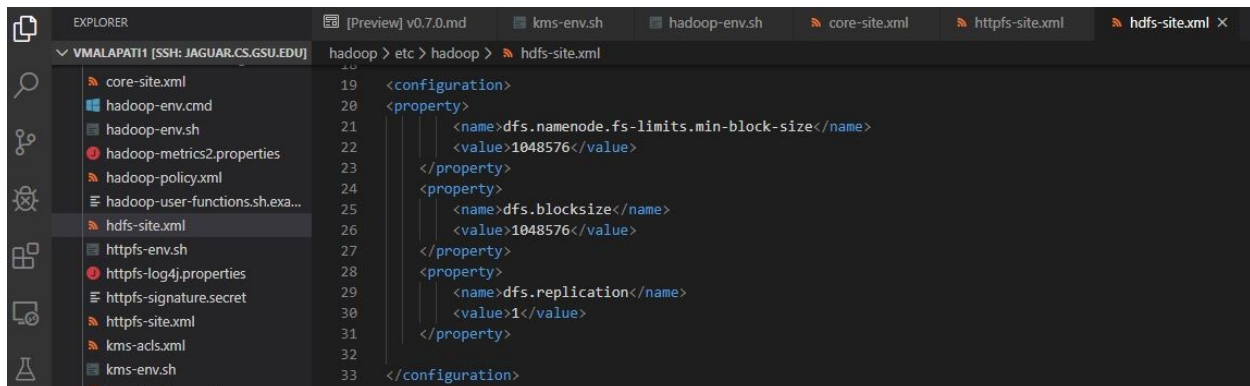


The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows a list of files under the path 'VMALAPATH1 [SSH: JAGUAR.CS.GSU.EDU]'. The code editor displays the contents of 'core-site.xml' with the following XML structure:

```
18
19 <configuration>
20   <property>
21     <name>hadoop.tmp.dir</name>
22     <value>/home/vmalapati1/hadoop/tmp</value>
23   </property>
24   <property>
25     <name>fs.defaultFS</name>
26     <value>hdfs://localhost:9000</value>
27   <!-- <value>file:/// </value> -->
28   </property>
29
30 </configuration>
```

Fig: showing core-site.xml in server

1.e: Further the two properties blocksize and number of replication is updated as 1mb As the server don't support for default values. It is done in hdfs-site.xml file.



The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows a list of files under the path 'VMALAPATH1 [SSH: JAGUAR.CS.GSU.EDU]'. The code editor displays the contents of 'hdfs-site.xml' with the following XML structure:

```
19 <configuration>
20 <property>
21   <name>dfs.namenode.fs-limits.min-block-size</name>
22   <value>1048576</value>
23 </property>
24 <property>
25   <name>dfs.blocksize</name>
26   <value>1048576</value>
27 </property>
28 <property>
29   <name>dfs.replication</name>
30   <value>1</value>
31 </property>
32
33 </configuration>
```

Fig: showing hdfs-site.xml in server

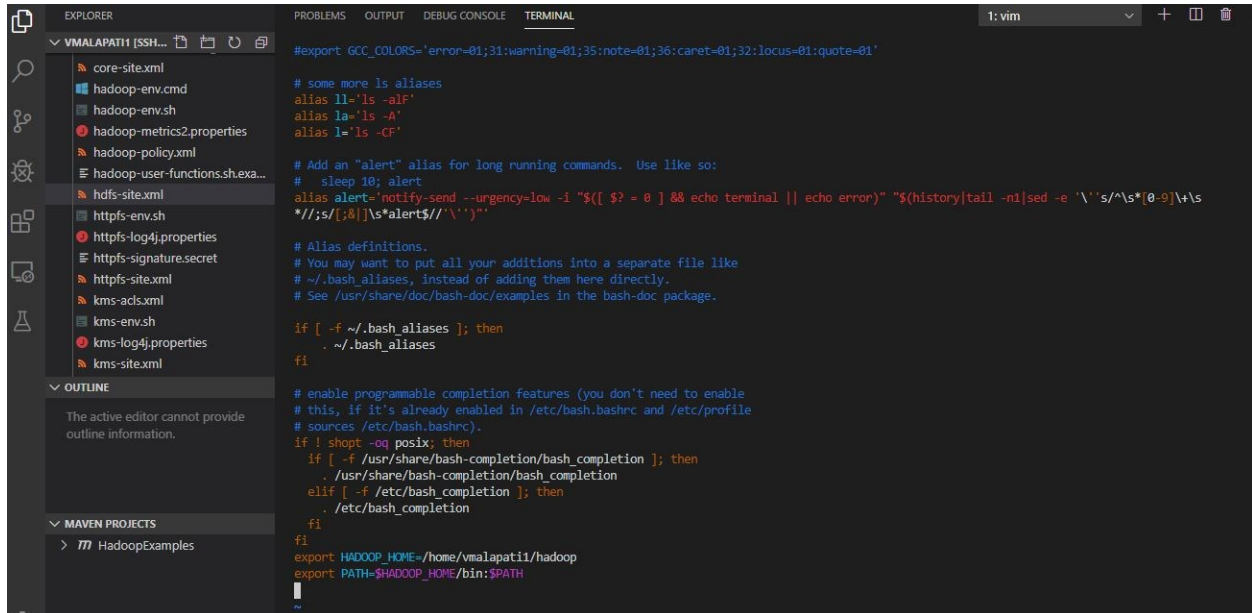
The code is:



```
<configuration>
<property>
  <name>dfs.namenode.fs-limits.min-block-size</name>
  <value>1048576</value>
</property>
<property>
  <name>dfs.blocksize</name>
  <value>1048576</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
```

</configuration>

1.f: As the Next step .bashrc is updated such that OS will recognize hadoop Updating the path of hadoop in bash script.



```
#export GCC_COLORS='error=01;31:warning=01;35:note=01;36:caret=01;32:locus=01:quote=01'

# some more ls aliases
alias ll='ls -alf'
alias la='ls -A'
alias l='ls -CF'

# Add an "alert" alias for long running commands. Use like so:
# sleep 10; alert
alias alert='notify-send --urgency=low -i "${?} = 0" && echo terminal || echo error)' "${history|tail -n1|sed -e '\s/\s*[0-9]\v+\s
*//;s/[:&]\s*alert$//'\''}"

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.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

export HADOOP_HOME=/home/vmalapati1/hadoop
export PATH=$HADOOP_HOME/bin:$PATH
```

Fig: Showing the Hadoop path adding in bash.rc

The additional commands added are:

```
export HADOOP_HOME=/home/rob/hadoop

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

1.g: HDFS set up is done by installing open ssh client server, Then from hadoop directory format the name node and start the hadoop file system and load the data into it and close the file system the operation is done as follows.

Installing the ssh server:

```
$ sudo apt update

$ sudo apt-get install ssh openssh-client openssh-server

$ sudo ufw allow ssh

$ ssh localhost
```

After installing the sshserver it will ask for password to set, here don't enter anything and press "enter" to continue if one enter's password here it will become ssh-key issue and this will lead to error. If by mistake any errors occur here run the following lines.

```
$ ssh-keygen -t rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod og-wx ~/.ssh/authorized_keys
```

Hdfs name formatting and putting the data in:

```
$ cd hadoop
$ hdfs namenode -format
$ sbin/start-dfs.sh
$ hdfs dfs -mkdir /user
$ hdfs dfs -mkdir /user/rob
$ hdfs dfs -mkdir /user/rob/data
$ hdfs dfs -put /home/rob/data/peterpan.txt /user/rob/data
$ hdfs dfs -ls /user/rob/data
$ sbin/stop-dfs.sh
```

The Data uploaded in the above process can also be done by creating a folder locally named data and store the required data files in it. This way file transfer can be done flexibly.

2a: Java extension is added to the Visual studio with all its packages including maven project. And then create a maven project specifying the directory and version. The VS code is built with extraordinary features such that all the java extensions are added at a time.

A maven project is created after downloading extensions with specifying the details like group-id, version, package, folder etc.

Maven repository contains many dependencies for projects in java in that here in hadoop needs map-reduce and some maven plugins.

Edit the pom.xml file and add maven and map reduce dependencies code from the maven repository. Before Create a runnable jar file, maven assembly plugins are added to pom.xml file and then build the maven project.

```
<dependency>
  <groupId>org.apache.hadoop</groupId>
  <artifactId>hadoop-common</artifactId>
  <version>3.2.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.hadoop/hadoop-mapreduce-
client-core -->
<dependency>
```

```

<groupId>org.apache.hadoop</groupId>
<artifactId>hadoop-mapreduce-client-core</artifactId>
<version>3.2.1</version>
</dependency>

```

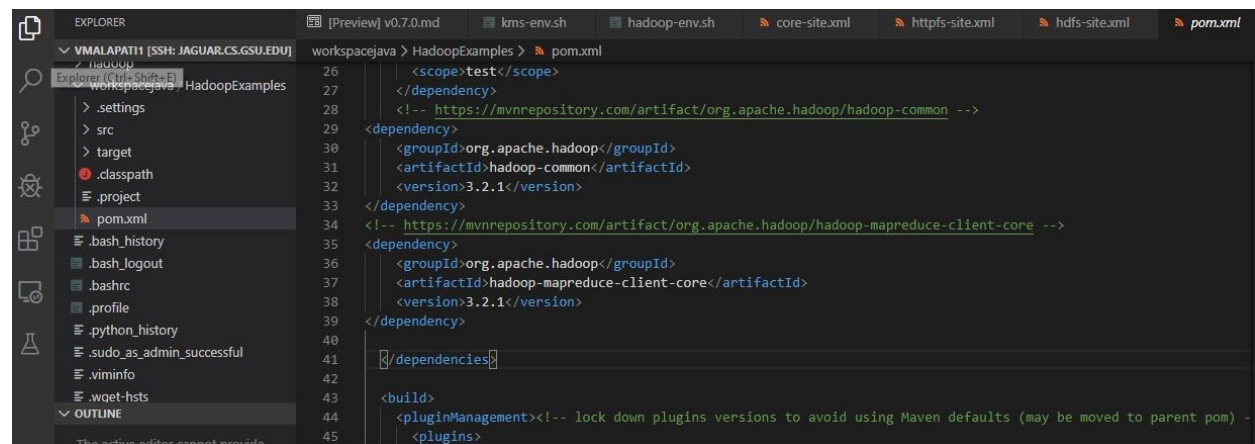


Fig: Showing the Dependencies add in pom xml file

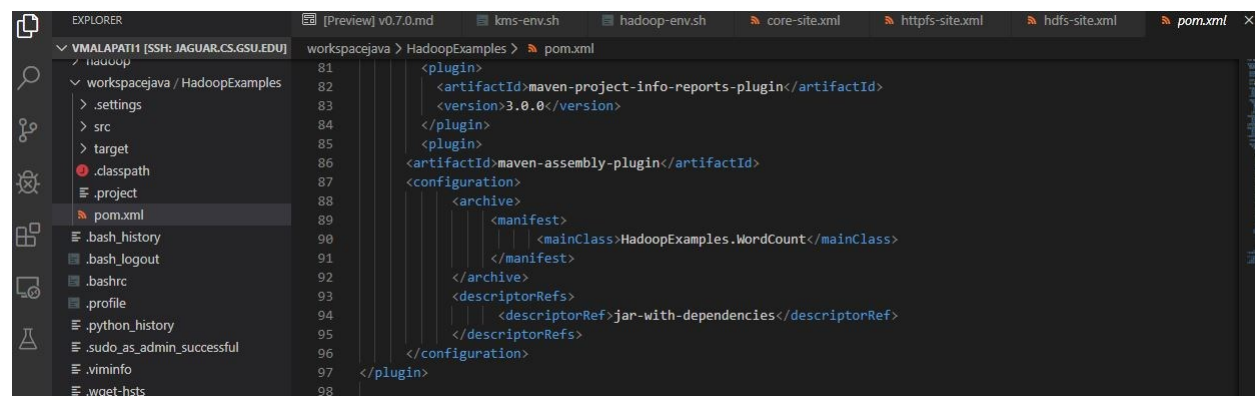


Fig: Showing the Plugins add in pom xml file

2.b: Finally after updating pom file with dependencies and plugins import the word-count example java file and generate a jar file by the command.

clean package assembly:single

A jar file is created in the target folder and we have everything and the word-count example is executed by specifying the input and output paths.

For test case:

hadoop jar /home/vmalapati1/workspacejava/HadoopExamples/target/HadoopExamples-1.0-jar-with-dependencies.jar file:///home/vmalapati1/data/test.txt file:///home/vmalapati1/data/output_test

```
SUBCOMMAND may print help when invoked w/o parameters or with -h.
vmalapati1@kvm_vmalapati1:~$ vim ~/.bashrc
vmalapati1@kvm_vmalapati1:~$ hadoop jar /home/vmalapati1/workspacejava/HadoopExamples/target/HadoopExamples-1.0-jar-with-dependencies.jar file:///home/vmalapati1/data/test.txt file:///home/vmalapati1/data/output_test
```

Fig: Showing the command running in for test case

The Output file:

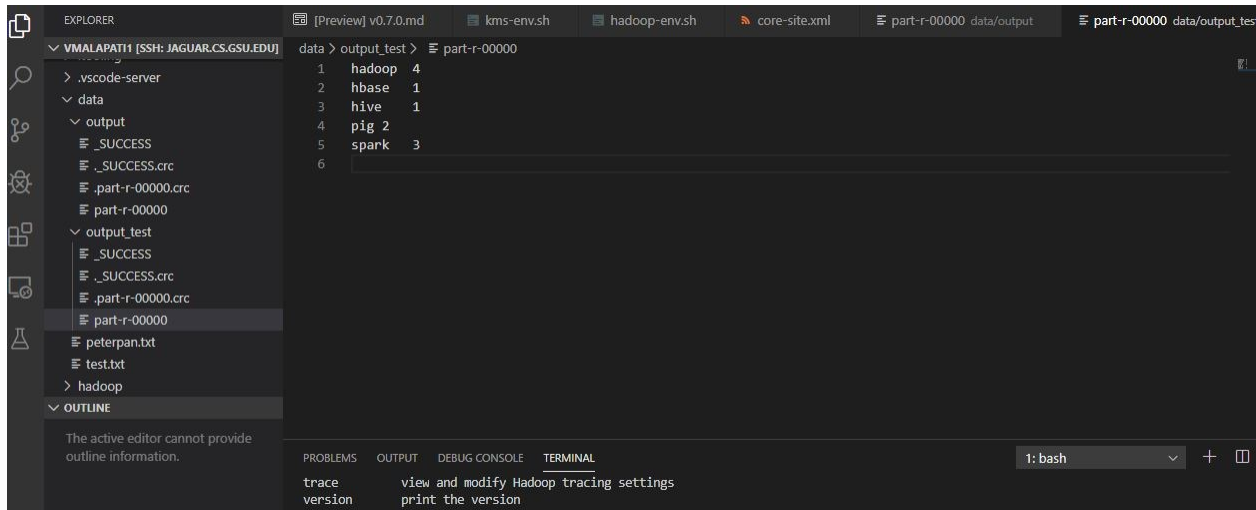


Fig: Showing the test output created and results

For peter-pan data:

hadoop jar /home/vmalapati1/workspacejava/HadoopExamples/target/HadoopExamples-1.0-jar-with-dependencies.jar file:///home/vmalapati1/data/test.txt <file:///home/vmalapati1/data/output>

```
vmalapati1@kvm_vmalapati1:~$ vim ~/.bashrc
vmalapati1@kvm_vmalapati1:~$ hadoop jar /home/vmalapati1/workspacejava/HadoopExamples/target/HadoopExamples-1.0-jar-with-dependencies.jar file:///home/vmalapati1/data/peterpan.txt file:///home/vmalapati1/data/output
```

Fig: Showing the command running in for peterpan case

The output File is shown below:

The output files can be extracted into local using “`hdfs get <hdfsdir> <localdir>`”.

The screenshot shows a VS Code editor interface with the following components:

- EXPLORER:** Displays a file tree for a project named 'VMALAPATH1 [SSH: JAGUAR.CS.GSU.EDU]'. The tree includes folders like '.vscode-server', 'data', 'output', and 'output_test'. The 'output' folder is expanded, showing files like '_SUCCESS', '_SUCCESS.crc', and 'part-r-00000.crc'. The 'output_test' folder is also expanded, showing files like '_SUCCESS', '_SUCCESS.crc', and 'peterpan.txt'.
- Terminal:** Shows the output of a command, likely 'cat', displaying a list of words and their frequencies. The output is as follows:

```
7925 things," 2
7926 things. 1
7927 think 47
7928 think, 5
7929 think," 1
7930 think. 1
7931 think: 1
7932 think?" 3
7933 thinking 14
7934 thinking," 1
7935 thinking." 1
7936 thinks 5
7937 thinnest 1
7938 thins 1
7939 thind 6
7940 third, 1
7941 thirty 3
7942 this 195
7943 this, 18
7944 this," 1
7945 this. 6
7946 this." 1
7947 this: 2
7948 this; 1
7949 this;" 1
7950 thither 1
7951 thither, 1
7952 thorough 1
7953 those 17
7954 thou 1
7955 thou?" 1
7956 though 46
```
- MAVEN PROJECTS:** Shows a project named 'HadoopExamples'.

The status bar at the bottom indicates the current file is 'part-r-00000 data/output' and the editor is in 'Plain Text' mode.

Fig: Showing output after running the peterpan case