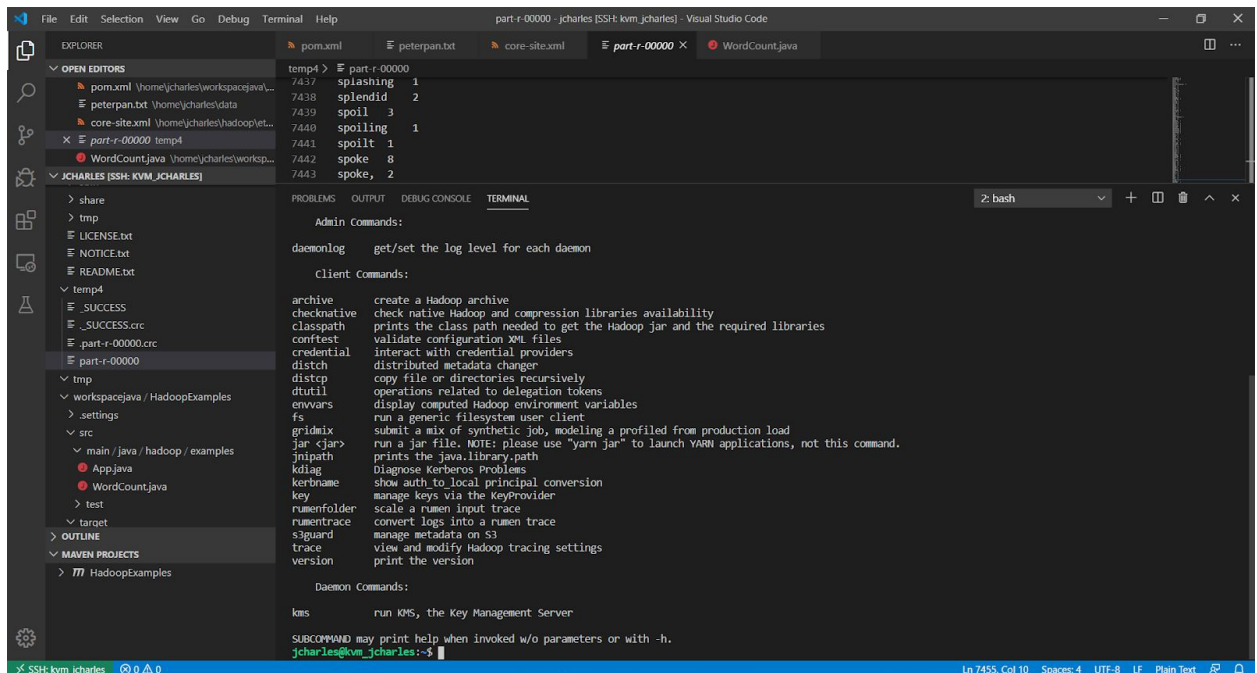


1. Setting up Hadoop

Setup Hadoop Setting up Hadoop allows us to later execute our java/jar files within hdfs. When you type “hadoop” in your command line, you should receive the following output to ensure that hadoop is set up properly.



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the output of the 'hadoop' command, which lists various subcommands and their descriptions. The subcommands are categorized into Admin, Client, and Daemon commands. The Client commands include archive, checknative, classpath, conftest, credential, distch, distcp, dtutil, envvars, fs, gridmix, jar, jni, jniutil, kdiag, krbname, key, rumenFolder, rumenTrace, s3guard, trace, and version. The Daemon commands include kms. The terminal also shows the output of the 'hadoop version' command, which prints the version information.

```
part-r-00000 - jcharles [SSH: kvm_jcharles] - Visual Studio Code
temp4> hadoop
Admin Commands:
daemonlog get/set the log level for each daemon

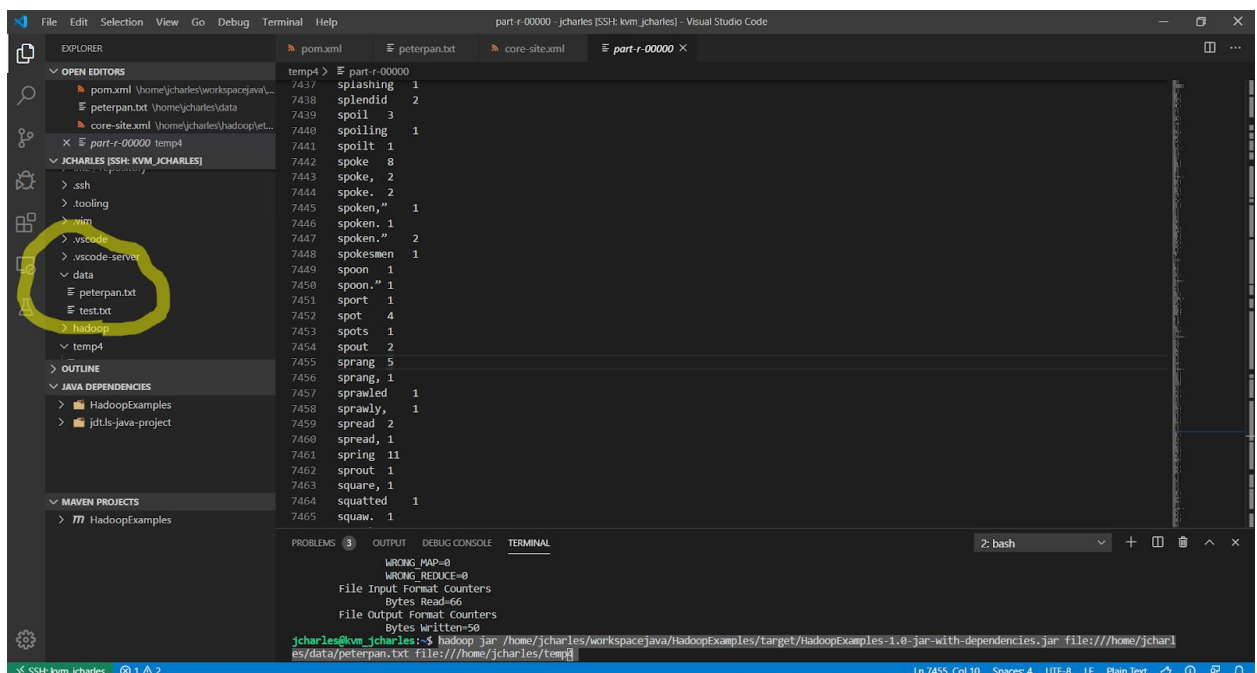
Client Commands:
archive create a Hadoop archive
checknative check native Hadoop and compression libraries availability
classpath prints the class path needed to get the Hadoop jar and the required libraries
conftest validate configuration XML files
credential interact with credential providers
distch distributed metadata changer
distcp copy file or directories recursively
dtutil operations related to delegation tokens
envvars display computed Hadoop environment variables
fs run a generic filesystem user client
gridmix submit a mix of synthetic job, modeling a profiled from production load
jar run a jar file. NOTE: please use "yarn jar" to launch YARN applications, not this command.
jni prints the java.library.path
jniutil Diagnose Kerberos Problems
krbname show auth to local principal conversion
key manage keys via the KeyProvider
rumenFolder scale a rumen input trace
rumenTrace convert logs into a rumen trace
s3guard manage metadata on S3
trace view and modify Hadoop tracing settings
version print the version

Daemon Commands:
kms run KMS, the Key Management Server

SUBCOMMAND may print help when invoked w/o parameters or with -h.
jcharles@kvm_jcharles:~$
```

2. Setting up HDFS

Setup HDFS and upload the datasets “test.txt” and “peterpan.txt” into HDFS. The setup of HDFS allows us to upload our files from our local system to our KVM. In the following output we see the files uploaded into our HDFS



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the output of the 'hadoop fs' command, which lists the contents of the HDFS file system. The files listed are 'test.txt' and 'peterpan.txt', both of which are highlighted with a yellow circle in the Explorer view. The terminal also shows the output of the 'hadoop fs -ls' command, which lists the files in the HDFS file system. The output shows that the files have been successfully uploaded to the HDFS file system.

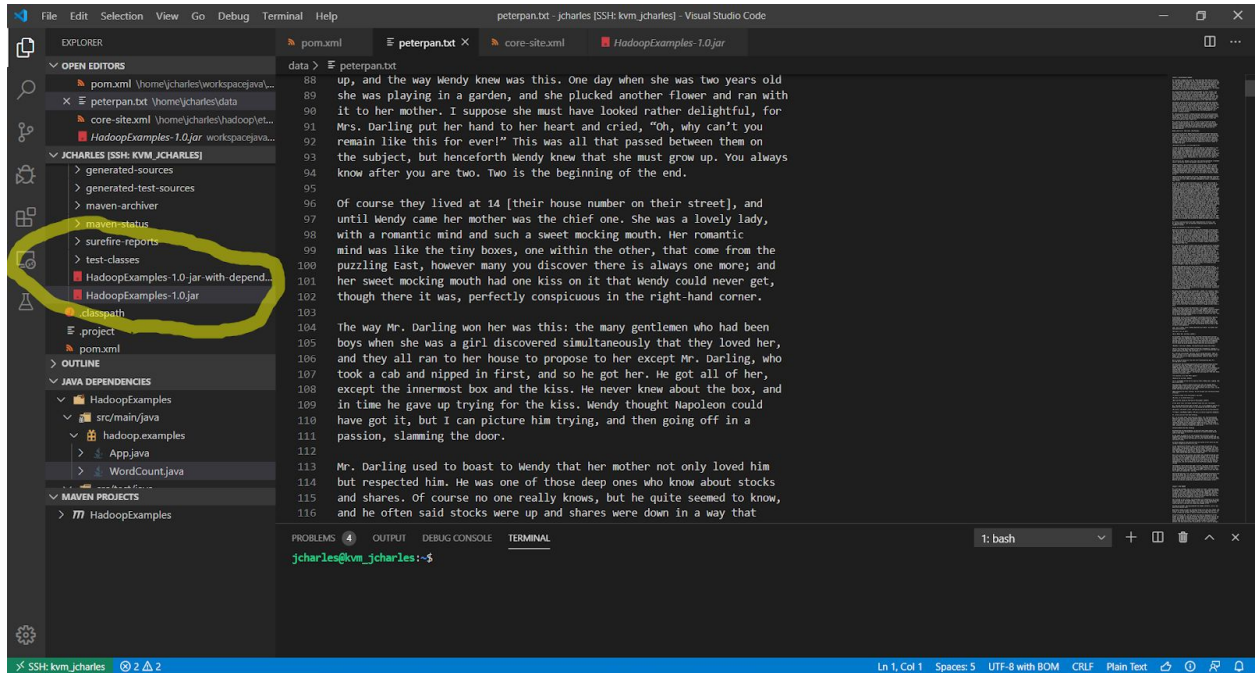
```
part-r-00000 - jcharles [SSH: kvm_jcharles] - Visual Studio Code
temp4> hadoop fs
ls
-rw-r--r-- 1 jcharles 1000 2020-08-10 10:10 test.txt
-rw-r--r-- 1 jcharles 1000 2020-08-10 10:10 peterpan.txt

jcharles@kvm_jcharles:~$ hadoop fs -ls
ls
-rw-r--r-- 1 jcharles 1000 2020-08-10 10:10 test.txt
-rw-r--r-- 1 jcharles 1000 2020-08-10 10:10 peterpan.txt

jcharles@kvm_jcharles:~$
```

3. Create a Project in VS Code

Create a project in VS Code, import the “WordCount.java”, configure the project, and export the “WordCount.jar” file. The WordCount.java file will help to create the WordCount.jar file in our target folder of the maven project. You need to create a custom build with “clean package assembly:single” as the keywords. This is what the jar file should look like after it is created.



4.

Open a terminal, and run the “WordCount.jar” file on “test.txt” and “peterpan.txt” respectively

I ended up coming to office hours to get peterpan.txt to run and later getting test.txt to run as well. I made that happen by using the command “*hadoop jar /home/jcharles/workspacejava/HadoopExamples/target/HadoopExamples-1.0-jar-with-dependencies.jar file:///home/jcharles/data/test.txt file:///home/jcharles/temp2*” for test.txt and “*hadoop jar /home/jcharles/workspacejava/HadoopExamples/target/HadoopExamples-1.0-jar-with-dependencies.jar file:///home/jcharles/data/peterpan.txt file:///home/jcharles/temp4*” for peterpan.txt and I had their outputs saved to different folders.

```
File Edit Selection View Go Debug Terminal Help
part-r-00000 - jcharles [SSH: kvm_jcharles] - Visual Studio Code

EXPLORER
pom.xml
peterpan.txt
core-site.xml
part-r-00000 temp4
JCHARLES [SSH: KVM_JCHARLES]
test.txt
hadoop
temp4
_SUCCESS
_SUCCESS.crc
part-r-00000.crc
part-r-00000
tmp
workspacejava / HadoopExamples
settings
OUTLINE
JAVA DEPENDENCIES
HadoopExamples
jdtls-Java-project
MAVEN PROJECTS
HadoopExamples

temp4> E part-r-00000
7437 splashing 1
7438 splendid 2
7439 spoil 3
7440 spoiling 1
7441 spoilt 1
7442 spoke 8
7443 spoke, 2
7444 spoke, 2
7445 spoken," 1
7446 spoken, 1
7447 spoken," 2
7448 spokesmen 1
7449 spoon 1
7450 spoon," 1
7451 sport 1
7452 spot 4
7453 spots 1
7454 spout 2
7455 sprang 5
7456 sprang, 1
7457 sprawled 1
7458 sprawly, 1
7459 spread 2
7460 spread, 1
7461 spring 11
7462 sprout 1
7463 square, 1
7464 squatted 1
7465 squaw, 1

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
2: bash
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=66
File Output Format Counters
Bytes Written=50
jcharles@kvm_jcharles:~$ hadoop jar /home/jcharles/workspacejava/HadoopExamples/target/HadoopExamples-1.0-jar-with-dependencies.jar file:///home/jcharles/data/peterpan.txt file:///home/jcharles/temp4
```

```
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL
2: bash

HDFS: Number of read operations=0
HDFS: Number of large read operations=0
HDFS: Number of write operations=0
HDFS: Number of bytes read erasure-coded=0
Map-Reduce Framework
Map input records=3
Map output records=11
Map output bytes=109
Map output materialized bytes=64
Input split bytes=98
Combine input records=11
Combine output records=5
Reduce input groups=5
Reduce shuffle bytes=64
Reduce input records=5
Reduce output records=5
Spilled Records=10
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=19
Total committed heap usage (bytes)=306003968
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=66
File Output Format Counters
Bytes Written=50
jcharles@kvm_jcharles:~$
```