

Metropolitan State University

ICS 432 - 01: Distributed and Cloud Computing

Fall 2021

Lab 09: Running a map-reduce job on a Hadoop cluster

Total points: **35**

Out: Saturday, October 30, 2021

Due: 11:59 PM on Friday, November 5, 2021

What to submit?

The objective of this lab is to practice using virtual machines on the cloud. To complete this lab:

- Read this lab assignment carefully.
- At various parts of the lab, you are asked to **take screen shots** of your work. Open a word document and paste the screen shots in this document in the same order as mentioned in the lab. Make sure to highlight the screen shot number.
- After you complete all the lab exercises, upload the word document to the designated D2L folder by 11:59 PM on Friday, November 5, 2021.

NOTE: On Windows machines, you may consider using [Snip & Sketch](#) for screenshot handling.

Exercise 1: Running map-reduce jobs on the cloud

In this lab, you will run the word count map reduce exercise on a Hadoop cluster on AWS. You will complete this exercise in the following steps:

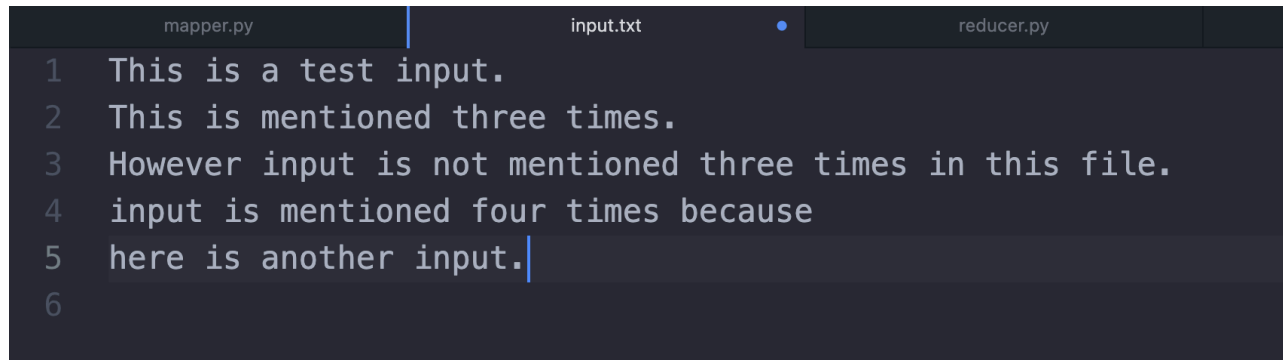
1. Testing the word count map reduce job on one machine.
2. Setting up an AWS Elastic **Map-Reduce** (EMR) cluster.
3. Connecting to the cluster's master node using SSH.
4. Working with HDFS.
5. Running a Python map-reduce job on the cluster.
6. Modifying the map code to remove special characters.
7. Terminate the cluster.

Step 1: Running word count on one machine

In this step, you will write the word count map reduce python code on Cloud9 and test the code before submitting it to the cluster.

- 1- To test the python code, you can either use AWS Cloud9 or GCP Editor.
- 2- Create a folder called `your-last-name-mr-wordcount`.

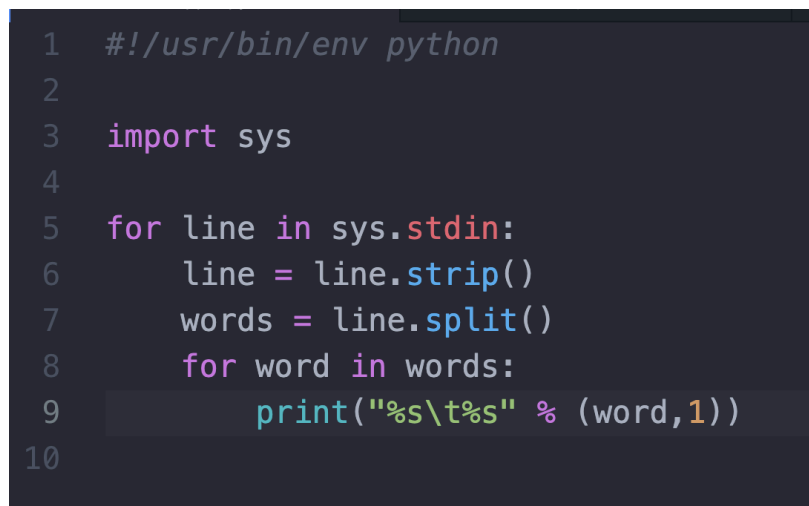
- 3- Create a file called `input.txt`. Write some text in the file to test. Make sure to repeat some words and use a mixed case for letters of the same word (e.g., This, this, and THIS). The following figure is an example text file.



```
mapper.py  input.txt  reducer.py
1  This is a test input.
2  This is mentioned three times.
3  However input is not mentioned three times in this file.
4  input is mentioned four times because
5  here is another input.
6
```

Lab screenshot #1: take a screenshot of your text file.

- 4- Create a file called `mapper.py` and write the following code. Note that the first line of the file (`#!/usr/bin/env python`) is needed later when we run the code on the cluster. For now, just include this line in any mapper or reducer python files.



```
1  #!/usr/bin/env python
2
3  import sys
4
5  for line in sys.stdin:
6      line = line.strip()
7      words = line.split()
8      for word in words:
9          print("%s\t%s" % (word,1))
10
```

- 5- Run the mapper code by typing the following command in the terminal window. Note that, in Linux OS, the pipe symbol (`|`) is used to forward the output of command to the next command. For example, in the following command the output of `cat input.txt` is used as the input to `python mapper.py`.

```
cat input.txt | python mapper.py
```

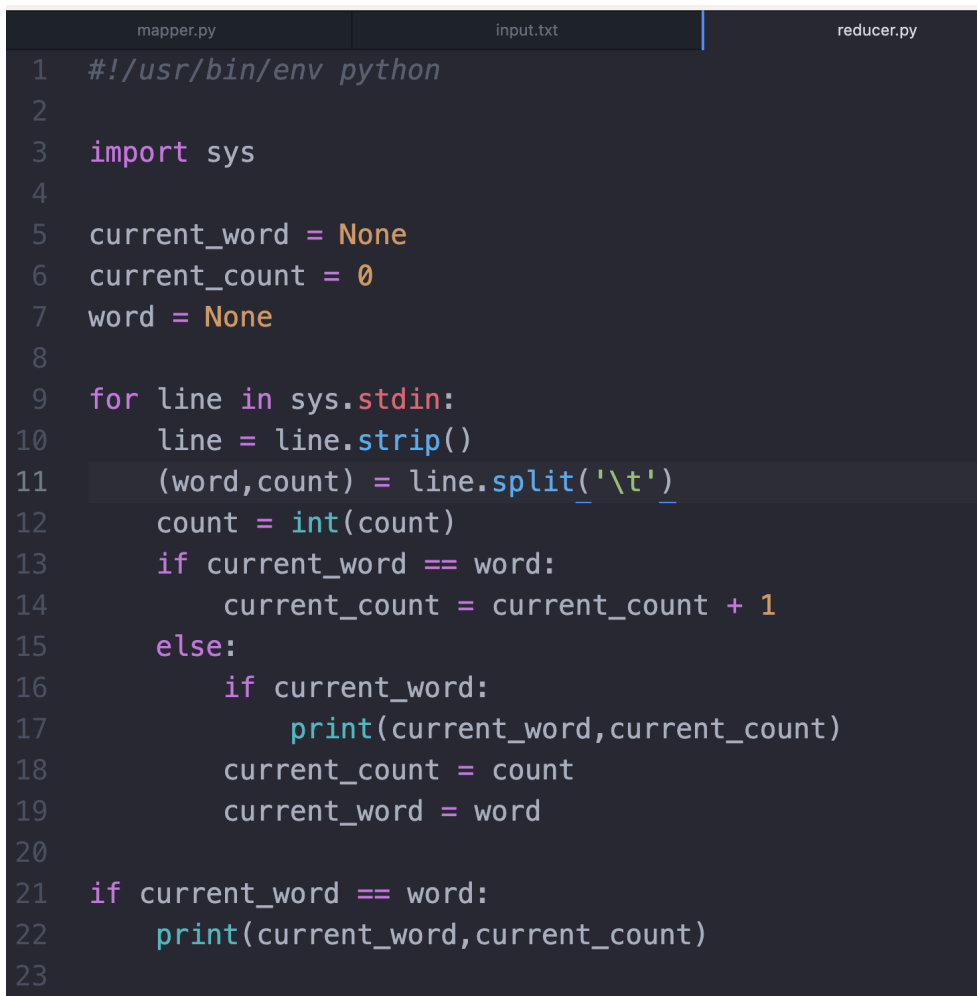
Lab screenshot #2: take a screenshot to show the output of your mapper.

- 6- Note that the correctness of the map-reduce job is dependent on the fact that the output of the mapper is sorted before being fed to the reducer. When the job is running on the cluster, the shuffle & sort component is responsible for the sorting. To test the job locally, you can use the Linux's sort tool to sort the output of the mapper before feeding it to the reducer. For example, the following command is used to sort the mapper output.

```
cat input.txt | python mapper.py | sort
```

Lab screenshot #3: take a screenshot to show the sorted output.

- 7- Create a file called `reducer.py` and write the following code.



```
1  #!/usr/bin/env python
2
3  import sys
4
5  current_word = None
6  current_count = 0
7  word = None
8
9  for line in sys.stdin:
10     line = line.strip()
11     (word,count) = line.split('\t')
12     count = int(count)
13     if current_word == word:
14         current_count = current_count + 1
15     else:
16         if current_word:
17             print(current_word,current_count)
18             current_count = count
19             current_word = word
20
21 if current_word == word:
22     print(current_word,current_count)
23
```

- 8- Run the map-reduce job using the following command.

```
cat input.txt | python mapper.py | sort | python reducer.py
```

Lab screenshot #4: take a screenshot to show the output.

- 9- Notice that, the same word but with different case letters are counted separately. Change the code of the mapper to convert the word to all lowercase letters before writing the word to the output stream. Run the code again.

Lab screenshot #5: take a screenshot to show the output and make sure to show at least one word for which the count is merged for the two versions of the word.

- 10- You are going to test the word count job on the cluster using the Shakespeare data set. To test this data set first on one machine, upload the `shakespeare` directory to your folder and use the following command to run the job. Note that `cat shakespeare/*` is used to stream all files in the directory.

```
cat shakespeare/* | python mapper.py | sort | python reducer.py
```

Lab screenshot #6: take a screenshot to show some lines of the output.

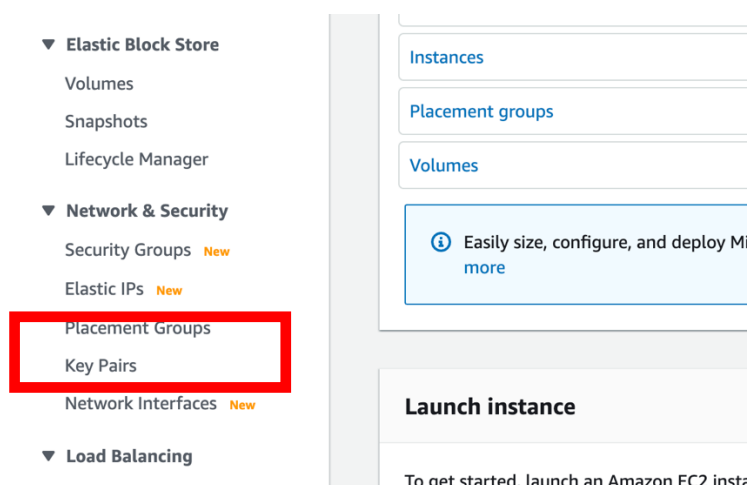
- 11- Download the `mapper.py`, `reducer.py`, and `input.txt` to your local computer because you will later transfer these files to the Hadoop cluster.

Step 2: Setting up an AWS Elastic MapReduce (EMR) cluster.

- 1- Connect to your AWS Educate account.

Task 1: Creating a key pair to access the cluster

- 2- Go to Services → EC2.
- 3- On the EC2 dashboard, From the left menu, scroll down to the 'Network and Security' section then click on 'Key Pairs'.



- 4- Click on Create Key Pair.
- 5- Type the name of the key pair's file (for example, awsemrkeypair). Choose:
 - a. Key pair type: RSA
 - b. Private key file format: extension .ppk for windows and choose extension .pem for Mac computers. Then click on 'Create key pair'.
- 6- The key pair file is created and downloaded on your computer under the 'Downloads' folder.

Task 2: Setting up the EMR cluster

- 7- Go back to the Service → Analytics → EMR (Elastic Map Reduce).
- 8- From the EMR console, click on 'Create Cluster'. Give the cluster a name (include your last name).
- 9- Click on 'Go to Advanced Options'

Create Cluster - Quick Options [Go to advanced options](#)

General Configuration

Cluster name

☒ Logging ⓘ

S3 folder

Launch mode ☒ Cluster ⓘ ☐ Step execution ⓘ

Software configuration

- 10- Select the components of the Hadoop ecosystem that you need in the cluster. Choose the services that are shown in the following figure. Note that we are NOT using all these services in this homework. Click on Next.

Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

Software Configuration

Release


<input checked="" type="checkbox"/> Hadoop 2.10.1	<input type="checkbox"/> Zeppelin 0.8.2	<input type="checkbox"/> Livy 0.7.0
<input checked="" type="checkbox"/> JupyterHub 1.1.0	<input type="checkbox"/> Tez 0.9.2	<input type="checkbox"/> Flink 1.11.2
<input type="checkbox"/> Ganglia 3.7.2	<input checked="" type="checkbox"/> HBase 1.4.13	<input checked="" type="checkbox"/> Pig 0.17.0
<input checked="" type="checkbox"/> Hive 2.3.7	<input type="checkbox"/> Presto 0.240.1	<input type="checkbox"/> ZooKeeper 3.4.14
<input type="checkbox"/> JupyterEnterpriseGateway 2.1.0	<input type="checkbox"/> MXNet 1.7.0	<input checked="" type="checkbox"/> Sqoop 1.4.7
<input type="checkbox"/> Mahout 0.13.0	<input checked="" type="checkbox"/> Hue 4.8.0	<input type="checkbox"/> Phoenix 4.14.3
<input checked="" type="checkbox"/> Oozie 5.2.0	<input checked="" type="checkbox"/> Spark 2.4.7	<input type="checkbox"/> HCatalog 2.3.7
<input type="checkbox"/> TensorFlow 2.3.1		

- 11- On the Hardware configuration options page, keep all the defaults as shown in the following figure. Click on Next.

Hardware Configuration ⓘ


Specify the networking and hardware configuration for your cluster. Request Spot instances (unused EC2 capacity) to save money.

Cluster Composition

Specify the configuration of the master, core and task nodes as an instances group or instance fleet. This choice applies to all nodes for the lifetime of the cluster. Instance fleets and instance groups cannot coexist in a cluster. [see this topic](#) .

Instance group configuration

☒ **Uniform instance groups**
Specify a single instance type and purchasing option for each node type.


☐ **Instance fleets**
Specify target capacity and how Amazon EMR fulfills it for each node type. Mix instance types and purchasing options. [Learn more](#) 

12- In the Networking section, make sure the subnet is **us-east-1d**

Networking

Use a Virtual Private Cloud (VPC) to process sensitive data or connect to a private network. Launch the cluster into a VPC with a public, private or shared subnet. Subnets may be associated with an AWS Outpost or AWS Local Zone.

Launch the cluster into a VPC with a public, private, or shared subnet. Subnets may be associated with an AWS Outpost or AWS Local Zone.

Network [Create a VPC](#)  ⓘ

EC2 Subnet

Cluster Nodes and Instances

13- On the 'General Options' console, write the Cluster name (use your last name) and check boxes as shown in the following figure. Click on Next.

Create Cluster - Advanced Options [Go to quick options](#)

Step 1: Software and Steps

Step 2: Hardware

Step 3: General Cluster Settings

Step 4: Security

General Options

Cluster name

S3 folder

- ☐ Log encryption ⓘ
- ☒ Debugging ⓘ
- ☐ Termination protection ⓘ

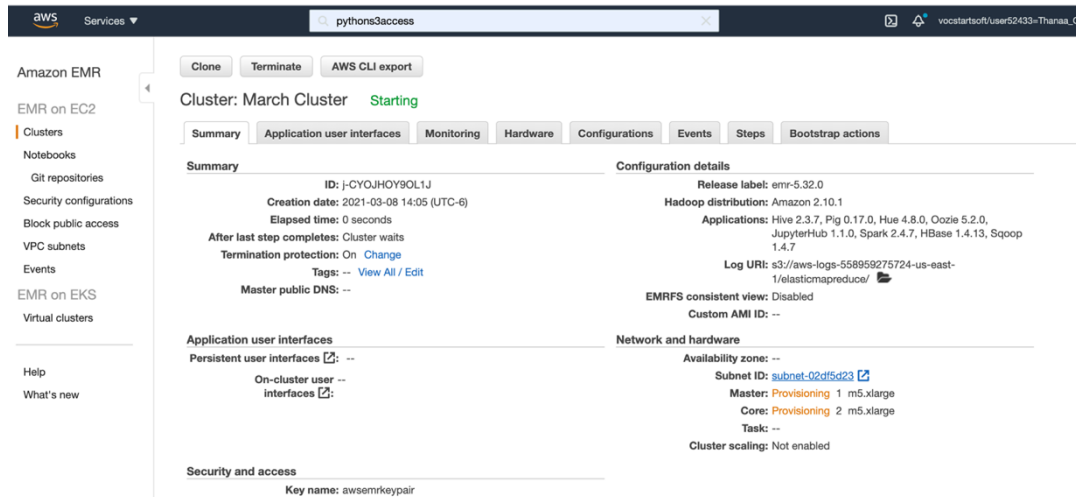
Tags ⓘ

Key	Value (optional)
<small>Add a key to create a tag</small>	

14- In the Security Options console, select the key-pair file that you created in Task 1.

15- Click on Create Cluster.

16- The Cluster information page will be displayed with the word 'Starting' as shown in the following figure. This process will take few minutes until the cluster is started. The cluster status will change from 'Starting' to 'Running' to finally 'Waiting'.



Lab screenshot #7: take a screenshot of your cluster's details screen and make sure the cluster's name appears in the screenshot.

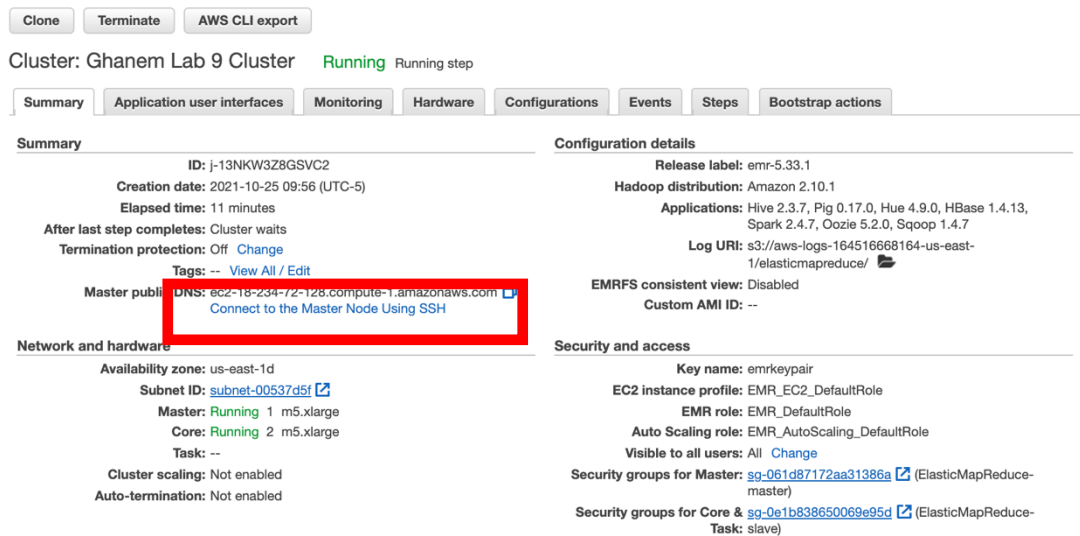
Lab screenshot #8: go to EC2 dashboard and take a screenshot to show the three running instances (one master and two worker).

Step 3: Connecting to the cluster's master node using SSH.

- 1- From the Cluster's console, under 'Security and Access', click the 'Security groups for Master'.



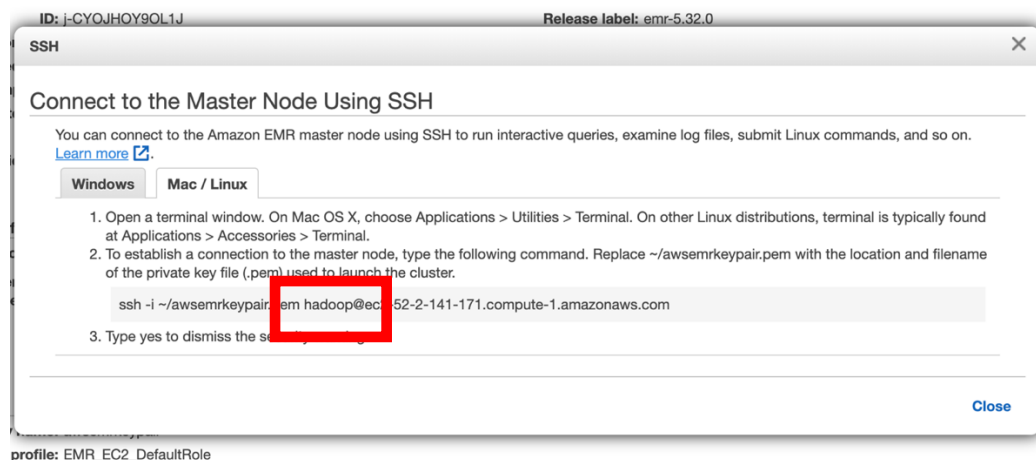
- 2- From the security group page, click on the 'Inbound Rules' tab. Click on Edit Inbound Rules and add a rule to allow SSH connection from Anywhere. If you get an error saying that this rule already exists, then just continue.
- 3- Go to Services → EMR and click on your cluster name.
- 4- Wait until the cluster's status changes to 'Running'.
- 5- Click on "Connect to the Master Node using SSH" as shown in the following figure.



- 6- A screen will be displayed with instructions to connect to master node. This is the same as connecting to any EC2 instance. For Mac computers, remember to run the `chmod` command on the key pair file. Note that, the SSH connection command now has user as **hadoop** instead of the usual **ec2-user**.

```
chmod 400 emrkeypair.pem
```

```
ssh -i awsemrkeypair.pem hadoop@ec2-52-2-141-171.compute-1.amazonaws.com
```



- 7- Once connected to the cluster's node terminal, create a directory on the master node to store files for the word count exercise. Include your last name in the directory name.

```
sudo mkdir ghanemwordcountonmaster
```

- 8- Transfer the following files from your computer to the master node inside the word-count-on-master directory: `mapper.py`, `reducer.py`, `input.txt`, and the folder `shakespeare`

Lab screenshot #9: take a screenshot of the EC2 terminal showing the contents of the word-count-on-master folder.

Step 4: Working with HDFS

In this step, you will practice reading and writing files to HDFS.

- 1- List all files (`ls`) in HDFS. Nothing will be displayed because you do not have any data in HDFS yet.

```
hadoop fs -ls
```

- 2- Create folder in HDFS to store input and output of the word count exercise. Include your last name in the folder name.

```
hadoop fs -mkdir ghanemwordcountonhdfs
```

- 3- Copy the `shakespeare` directory to HDFS for the map reduce job to use them. The format of the command is:

```
hadoop fs -copyFromLocal <source> <destination>
```

```
hadoop fs -copyFromLocal /home/hadoop/ghanemwordcountonmaster/shakespeare  
ghanemwordcountonhdfs/shakespeare
```

- 4- Make sure the directory is correctly copied to HDFS. The following command should show there are three files in the input folder. Note that `lsr` means recursive `ls` and it used to list all folders and files within the folders.

```
hadoop fs -lsr ghanemwordcountonhdfs
```

Lab screenshot #10: take a screenshot of the command window showing the output of the HDFS commands.

Task: Hands-on exercises with HDFS

- 5- In this task, you will do some more hands-on exercises practice with HDFS. Refer to slide #73 from week 9 slides for the format of the commands. For each of the following questions, write a command to execute the job, copy the command in your lab report, and **include lab screenshots** of the command window showing the output of the command.

Lab screenshot #11: take a screenshot of the command window after executing the HDFS commands. You may include more than one command in the same screenshot.

- a. Create a directory (with a name of your choice) in HDFS.
- b. Copy any file of your choice from the local master node to the newly created HDFS directory. You may create a simple text file using `cat` and `>`.
- c. List the new directory's contents.
- d. View the contents of the file (use `-cat`).
- e. Rename the file in HDFS. (use `-mv`)
- f. Create another directory in HDFS and move the file you created in #b to the new directory.
- g. Display the size of the file.
- h. Move the copied file from HDFS to the local machine.
- i. Delete the file you created in #b.
- j. Delete the directory you created in #a.

Step 5: Running the map-reduce job on the Hadoop cluster

- 1- In step 2 above, you transferred the `mapper.py` and `reducer.py` files to the cluster's master node.
- 2- Run the following commands to change the mode of mapper and reducer files to be executable.

```
sudo chmod +x mapper.py
sudo chmod +x reducer.py
```

- 3- First, you will test the job on the master's node using the `input.txt` file. Run the following command to test `mapper.py`. Note that, unlike Step 1 above, in this command we are not typing `python` before `mapper.py` because we changed the mode of the mapper file to be executable which means the file is executed by `./mapper.py`. This is possible because of the first line (`#!/usr/bin/env python`) in `mapper.py` and `reducer.py` that tells the operating system to run these files using python.

```
cat input.txt | ./mapper.py
```

Lab screenshot #12: take a screenshot of the command window showing the mapper output.

- 4- Test the map reduce job on the master node:

```
cat input/file1.txt | ./mapper.py | sort | ./reducer.py
```

Lab screenshot #13: take a screenshot of the command window showing the output.

- 5- Next, you will run the map reduce job on the Hadoop cluster. First, in the master node's terminal, run the following command to find Hadoop's streaming library:

```
find /usr/lib/ -name *hadoop*streaming*.jar
```

You should get an output similar to the following:

```
[hadoop@ip-172-31-37-208 wordcountlocal]$ find /usr/lib/ -name *hadoop*streaming*.jar
/usr/lib/hadoop/hadoop-streaming-2.10.1-amzn-1.1.jar
/usr/lib/hadoop/hadoop-streaming.jar
/usr/lib/hadoop-mapreduce/hadoop-streaming.jar
/usr/lib/hadoop-mapreduce/hadoop-streaming-2.10.1-amzn-1.1.jar
find: '/usr/lib/hadoop-kms/share/hadoop/kms/tomcat/logs': Permission denied
```

Lab screenshot #14: take a screenshot of the output of the find command.

- 6- From the master node's terminal window ,run the following command to execute the map-reduce job on the cluster. Note that these are the parts of the command:
- Command to run a map-reduce job: `hadoop jar`
 - Python library to run map reduce jobs, you may change that according to the location of the library from the previous figure: `/usr/lib/hadoop-mapreduce/hadoop-streaming-2.10.1-amzn-1.1.jar`
 - Implementation of map and reduce methods: `-file mapper.py, reducer.py`
 - Map implementation: `-mapper mapper.py`
 - Reduce implementation: `-reducer reducer.py`
 - Path to the input on HDFS: `-input ghanemwordcountonhdfs/Shakespeare`
 - Path to the output directory on HDFS to be used to write the output of the job: `-output ghanemwordcountonhdfs/output`

```
hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming-2.10.1-amzn-1.1.jar -files mapper.py,reducer.py -mapper mapper.py -reducer reducer.py -input ghanemwordcountonhdfs/shakespeare -output ghanemwordcountonhdfs/output
```

- 7- The map-reduce job is executed and you will be receiving notifications about how much percent of the job is completed. Wait until the job is completed and then take a screenshot.

Lab screenshot #15: take a screenshot to show the command window after the map reduce job is completed.

- 8- From the output of the job on the terminal window, examine the **Job Counters/Map-reduce framework** statistics at the end of the job and find answers to the following questions and fill in the following table with the numbers.

How many data-local map tasks were executed?	
How many rack-local map tasks were executed?	
How many reduce tasks?	
Total time spent by all maps?	
Total time spent by all reduce tasks?	
How many map-input records?	
How many map output records?	
How many reduce input groups?	

- 9- Run the following command to see how many output files are written to HDFS.

```
hadoop fs -ls ghanemwordcountonhdfs/output
```

Lab screenshot #16: take a screenshot to show the list of output files.

- 10- Examine the contents of the three output files by running the following command three times once for each file.

```
hadoop fs -cat ghanemwordcounthdfs/output/part-00000
```

- 11- Note that the words are sorted alphabetically in each output file. Find in which file of the three output files the word 'winds' is located. Take a screen shot to show how many variations of the word (e.g., winds?, winds!, winds, winds:) exists and the count of each variation. You may choose any other word.

Lab screenshot #17: take a screenshot to show the variations of the word winds.

Step 6: Modifying the map code to remove special characters

In this step, you will modify the mapper code to remove special characters so that we can an accurate count of each word.

- 1- Go back to the job implementation on one machine from Step 1. Create a new file called `mapper2.py` and copy the code from `mapper.py` to `mapper2.py`. Change the `mapper2.py` code to remove all special characters from the code. To do that in Python, import the regular expression library (`import re`). Then remove all special characters from a word using the following code.

```
word = re.sub('[^A-Za-z0-9]+', '', word)
```

- 2- Include some special characters in some words in your `input.txt` file and test `mapper2.py`.

Lab screenshot #18: take a screenshot to show your mapper code.

Lab screenshot #19: take a screenshot to show the output of the mapper when executed on `input.txt`.

- 3- After you make sure the new mapper works as expected, execute the whole map-reduce job on `input.txt`.

Lab screenshot #20: take a screenshot to show the output of the map-reduce job with the new mapper on `input.txt`.

- 4- Test the map-reduce job on `shakespeare` data. You may receive an error because after removing the special characters, there will be words with length 0 in the output. To resolve this error, add an `if` statement to `mapper2.py` to produce in the output only words with `length > 0` (use `len(word)`).

Lab screenshot #21: take a screenshot to show the output of the map-reduce job with the new mapper on `shakespeare`.

- 5- Transfer `mapper2.py` to the cluster's master node in the same directory as the `mapper.py` and `reducer.py`.
- 6- Test the new job on the master's node before sending it to the cluster. This is the same as what you have done in Step 5.
- 7- Run the map-reduce job on the cluster using `mapper2.py` instead of `mapper.py`. Use `output2` as the name of the output file instead of `output`.

Lab screenshot #22: take a screenshot to show the output of the map-reduce job after running on the cluster.

- 8- Fill in the third column in the following table with the statistics of the job. Note that the second column was filled in Step 5. **Make sure to include the table in your submitted lab report.**

	mapper.py	mapper2.py
How many data-local map tasks were executed?		
How many rack-local map tasks were executed?		
How many reduce tasks?		
Total time spent by all maps?		
Total time spent by all reduce tasks?		
How many map-input records?		
How many map output records?		
How many reduce input groups?		

- 9- Examine the output files in the `output2` folder and find the word winds.

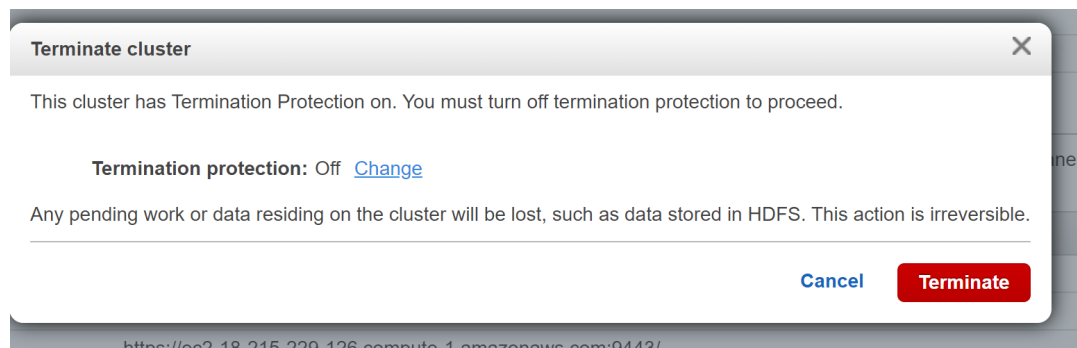
Lab screenshot #23: take a screenshot to show the count of the word winds.

Step 7: Terminate the cluster

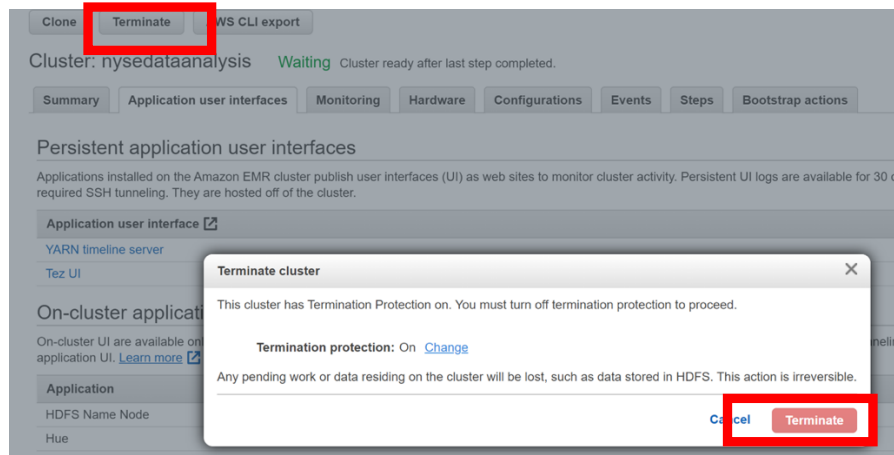
Unfortunately, you cannot stop an EMR cluster because the EMR cluster uses instance-store volumes and the EC2 start/stop feature relies on the use of EBS volumes which are not appropriate for high-performance, low-latency HDFS utilization. Terminate the EMR as follows.

- 1- From the cluster console, click on Terminate.
- 2- If your cluster does not have termination protection, then you can just terminate the cluster.

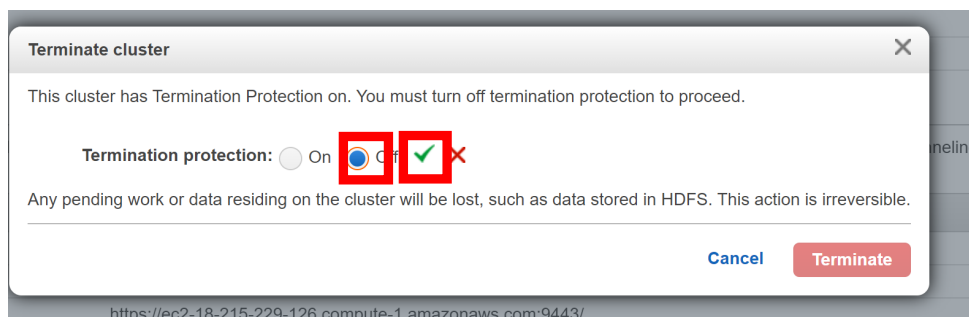
- 3- If you kept termination protection on when you created the cluster, then you should turn it off first then terminate the cluster.



- 4- To do that, click on Change on the pop up window to turn off the cluster termination protection



- 5- Choose Off. Then click on the green arrow.



6- After termination protection is turned off, click on Terminate.

