# **CSC 555** Assignment 2

Ronaldlee Ejalu

Reading: *Mining of Massive Datasets*: Chapter 2.

*Hadoop: The Definitive Guide*: Chapter 17 (Hive), Appendix A (file also available on D2L).

Supplemental document **UsingAmazonAWS.doc** and **Instructions_ReformatHDFS_Hive.doc**

The reformatting instructions are included in case you have to re-initialize your instance. You would **only** need it if you need to reformat your Hadoop set up.

## Part 1

1) Describe how you would implement a MapReduce job consisting of Map and Reduce description. You can describe it in your own words or as pseudo-code. Keep in mind that map task reads the input file and produces (key, value) pairs. Reduce task takes a list of (key, value) pairs for each key and combines all values for each key.
Remember that Map operates on individual blocks and Reduce on individual keys with a set of values. Thus, for Mapper you need to state what your code does given a block of data and for Reduce you need to state what your reducer does <u>for each key</u>. You can assume that all of the columns accessed by the query exist in the original table.

a) SELECT First, SUM(Grade)
FROM Student
GROUP BY First;

Pseudo-code:

| | Key | Value |
|---|---|---|
| Mapper | First | Grade |
| Reducer | First | SUM(Grade) |

MapReduce works by breaking the processing into two phases: the map and the reduce phase as explained below:

The input to the mapper will be a row and it will produce key value pairs of First as the key and Grade as the value. Furthermore, First and Grade will be the key and value pairs respectively for each row that is processed by the mapper. The mapper emits the key value pairs (First: Grade), as shown above in the table, as its output.

The MapReduce framework receives output from the Mapper in form of keys and values and uses a deterministic hash function on the keys where it uses the value MOD N to distribute values to the different reducers. This process sorts and groups the key – value pairs by key, which is First. The list of grades will be the values for a particular key, First. So, the key (First) and the values (list of grades) will act as an input to the reduce function or phase.

Using the list of grades per key, the reducer will generate the sum of grades with an output of key-value pairs corresponding to First and sum of Grades respectively.

b) SELECT City, State, COUNT(DISTINCT Name)
FROM Student
GROUP BY City, State;
Pseudo-code:

| | Key | Value |
|---|---|---|
| Mapper | City-State | DISTINCT NAME |
| Reducer | City-State | COUNT(DISTINCT NAME) |

For each row the mapper processes, it emits an output of key value pairs.
The key will be a combination of City and State and the value will be the "Distinct" Name.

The output of the mapper i.e., City-State : {Distinct Name} will be received by the MapReduce framework, which uses a deterministic hash function on the keys by using the values of the hash function on keys MOD N to distribute values to the different reducers. This process sorts and groups the key – value pairs by the key, which is the combination of City and State.
The reducers will receive the key, which is a combination of City and State and the values, which is a list of the "DISTINCT" Names as its inputs.

For each key, the reducer will count the distinct names in the list and the output City-State as the key and Count(DISTINCT NAME) as the value.

2) Suppose you are tasked with analysis of the company's web server logs. The log dump contains a large amount of information with up to 7 different attributes (columns). You regularly run a Hadoop job to perform analysis pertaining to 3 specific attributes – TimeOfAccess, OriginOfAccess and FileName out of 7 total in the file.

a) How would you attempt to speed up the regular execution of the query? (2-a is intentionally an open-ended question, there are several acceptable answers)
This will be achieved in a variety of ways:
   - Rather than operating on individual rows of data at a time, the Hadoop job performs a batch of rows at a time. Operating on individual rows is like running a for loop on the individual elements of a big list which is slower.
   - Out of the 7 attributes, we extract out the four attributes which are not needed for the job and only live TimeOfAccess, OriginOfAccess and FileName in the file on which we regularly run a Hadoop job to perform analysis. This will be done in the Map function.

- In the load function, I would implement the LoadPushDown Interface as a means of finding out which columns the query is asking for, this is where I would strictly specify the TimeOfAccess, OriginOfAccess and FileName attributes and leave out all the unnecessary attributes. This acts as an optimization technique for column-oriented storage, so that the loader only loads only those attributes needed by the query.

b) If a Mapper task fails while processing a block of data – which node(s) where MapReduce framework will prefer to restart it?
MapReduce framework will restart the NameNode.

c) If the job is executed with 3 Reducers
   i) How many files does the output generate?
   The output generates 1 file.
   ii) Suggest one possible hash function that may be used to assign keys to reducers.
   The deterministic Hash function works on the key field of a record; use its value MOD N to distribute values over range 0 …… N-1. N is equal to the number of buckets, which correspond to reducers.

3) Consider a Hadoop job that processes an input data file of size equal to 65 disk blocks (65 different blocks, you can assume that HDFS replication factor is set to 1). The mapper in this job requires 1 minute to read and process a single block of data. For the purposes of this assignment, you can assume that the reduce part of this job takes zero time. You can also refer to the supplemental example on how to make this estimate.

a) Approximately how long will it take to process the file if you only had one Hadoop worker node? You can assume that that only one mapper is created on every node.

Remember, the mapper in the job requires 1 minute to read and process a single block of data therefore for 65 blocks:

1 * 65 = 65 minutes.

b) 10 Hadoop worker nodes?

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | B1 | B7 | B13 | B19 | B25 | B31 | B37 | B43 | B49 | B55 |
| 2 | B2 | B8 | B14 | B20 | B26 | B32 | B38 | B44 | B50 | B56 |
| 3 | B3 | B9 | B15 | B21 | B27 | B33 | B39 | B45 | B51 | B57 |
| 4 | B4 | B10 | B16 | B22 | B28 | B34 | B40 | B46 | B52 | B58 |
| 5 | B5 | B11 | B17 | B23 | B29 | B35 | B41 | B47 | B53 | B59 |
| 6 | B6 | B12 | B18 | B24 | B30 | B36 | B42 | B48 | B54 | B60 |
| 7 | B61 | B62 | B63 | B64 | B65 | | | | | |

- So, 10 Hadoop worker nodes take 7 mins.

c) 30 Hadoop worker nodes?

| | Node 1 | Node 2 | Node 3 | Node 4 | Node 5 | Node 6 | Node 7 | Node 8 | Node 9 | Node10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | B1 | B3 | B5 | B7 | B9 | B11 | B13 | B15 | B17 | B19 |
| 2 | B2 | B4 | B6 | B8 | B10 | B12 | B14 | B16 | B18 | B20 |
| 3 | B61 | B62 | B63 | B64 | B65 | | | | | |

| Node 11 | Node 12 | Node 13 | Node 14 | Node 15 | Node 16 | Node 17 | Node 18 | Node 19 | Node 20 |
|---|---|---|---|---|---|---|---|---|---|
| B21 | B23 | B25 | B27 | B29 | B31 | B33 | B35 | B37 | B39 |
| B22 | B24 | B26 | B28 | B30 | B32 | B34 | B36 | B38 | B40 |

| Node 21 | Node 22 | Node 23 | Node 24 | Node 25 | Node 26 | Node 27 | Node 28 | Node 29 | Node 30 |
|---|---|---|---|---|---|---|---|---|---|
| B41 | B43 | B45 | B47 | B49 | B51 | B53 | B55 | B57 | B59 |
| B42 | B44 | B46 | B48 | B50 | B52 | B54 | B56 | B58 | B60 |

- So, 30 Hadoop worker nodes are going to take 3 mins.

d) 100 Hadoop worker nodes?
- The 65 blocks will be distributed across the 65 worker nodes and 35 nodes will be left idle so it will take 1 min to process the 65 blocks on 100 Hadoop worker nodes.

e) Now suppose you were told that the replication factor has been changed to 3. That is, each block is stored in triplicate, but file size is still 65 blocks. Which of the answers (if any) in a)-d) above will have to change?
- Changing the replication factor to 3 has nothing to do with the time of the processing. A replication factor of 3 ensures a good balance among reliability, write bandwidth, read performance and block distribution across the cluster.

You can ignore the network transfer costs and other potential overheads as well as the possibility of node failure. State any assumptions you make.

- The communication cost, which often influences our choice of algorithm to use within a cluster computing environment.
- The importance of wall-clock time, the time it takes a parallel algorithm to finish; the operations should be divided fairly among tasks implying that the wall clock time would be approximately as small as it would be, given the number of compute nodes available.

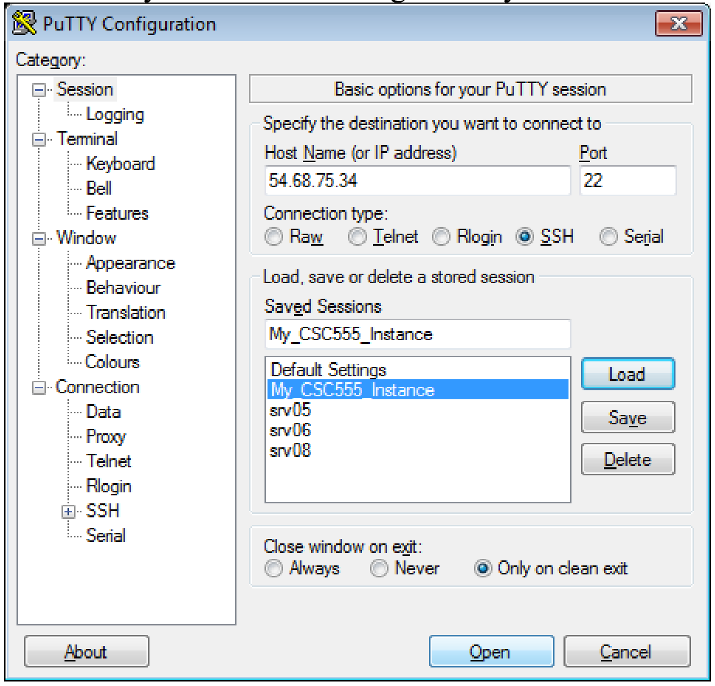
## Part 2: Linux Intro

This part of the assignment will serve as an introduction to Linux. Make sure you go through the steps below and submit screenshots where requested – submit the entire screenshot of a command terminal.

Use at least a t2.small instance or Hadoop may not run properly with insufficient memory.
All Linux commands are in **Berlin Sans FB**. Do not type the "$" symbol. The "$" represents the prompt "[ec2-user@ip-xxx-xx-xx-xxx ~] $ " in your particular Linux instance.

**0. Login to your Amazon EC2 Instance** (NOTE: instructions on how to create a new instance and log in to it are provided in a separate file, UsingAmazonAWS.doc)

Connect to your instance through PuTTy or a Mac terminal.


PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 54.68.75.34
Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

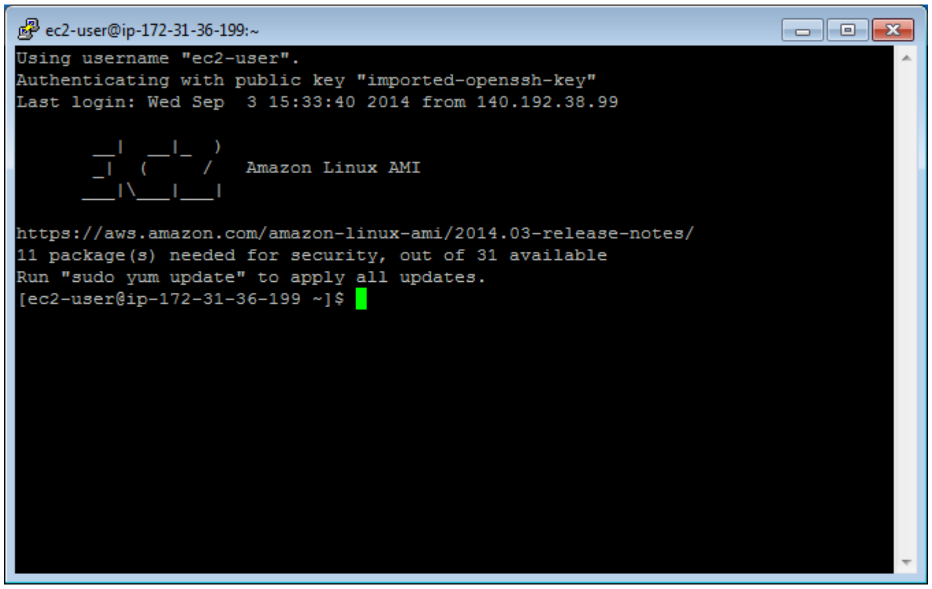
Load, save or delete a stored session

Saved Sessions: My_CSC555_Instance

Default Settings
My_CSC555_Instance
srv05
srv06
srv08

Close window on exit: Always Never [✓ Only on clean exit]

On Windows, your instance would look similar to the following image. On a Mac, it would be an XTerm terminal with same text:

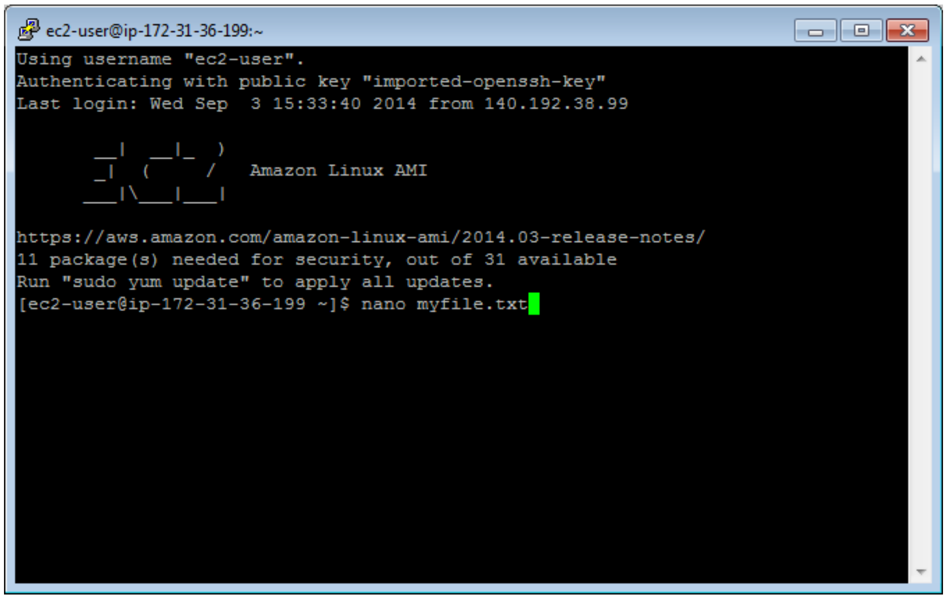


```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Wed Sep  3 15:33:40 2014 from 140.192.38.99

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.03-release-notes/
11 package(s) needed for security, out of 31 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-36-199 ~]$
```

1. **Create a text file.**
   Instructions for 3 different text editors are provided below. You only need to choose one editor that you prefer. **nano** is a more basic text editor, and is much easier to start. **vim** and **emacs** are more advanced and rely on keyboard shortcuts quite a bit and thus have a steeper learning curve.
   • **Tip**: To paste into Linux terminal you can use right-click. To copy from the Linux terminal, you only need to highlight the text that you want to copy with your mouse. Also please remember that Linux is case-sensitive, which means **Nano** and **nano** are not equivalent.

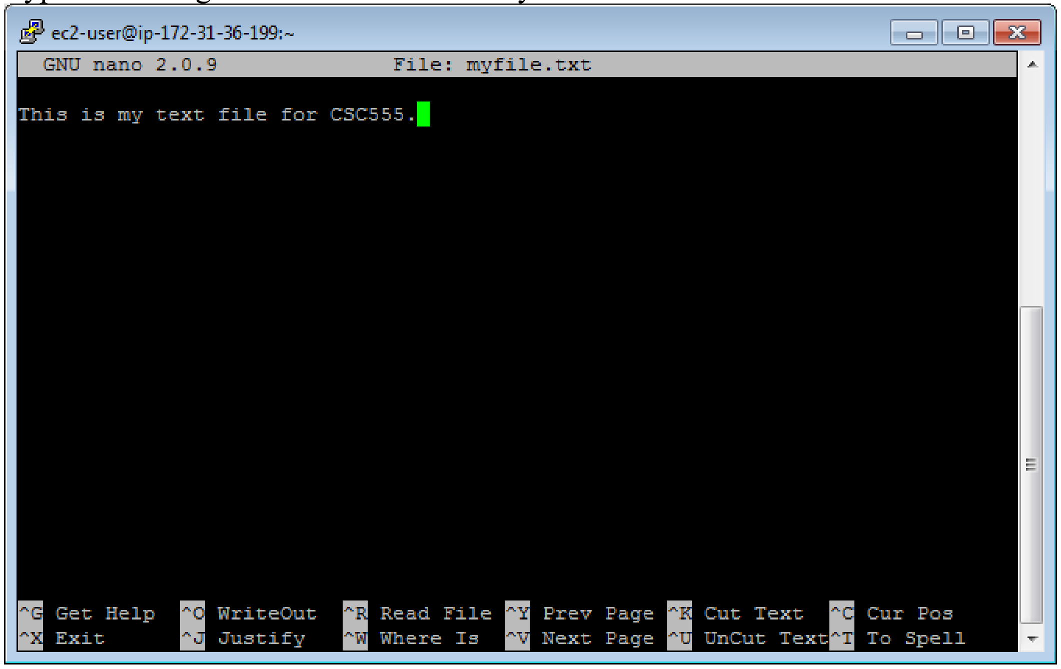
   Nano Instructions(Option 1):

   ```
   $ nano myfile.txt
   ```



```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Wed Sep  3 15:33:40 2014 from 140.192.38.99

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.03-release-notes/
11 package(s) needed for security, out of 31 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-36-199 ~]$ nano myfile.txt
```

Type something into the file: “This is my text file for CSC555.”



```
  GNU nano 2.0.9                File: myfile.txt

This is my text file for CSC555.

^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Save changes: Ctrl-o and hit Enter.
Exit: Ctrl-x

Emacs Instructions (Option 2):

You will need to install emacs.
**$ sudo yum install emacs**
Type “y” when asked if this is OK to install.

**$ emacs myfile.txt**
Type something into the file: “This is my text file for CSC555.”
Save changes: Ctrl-x, Ctrl-s

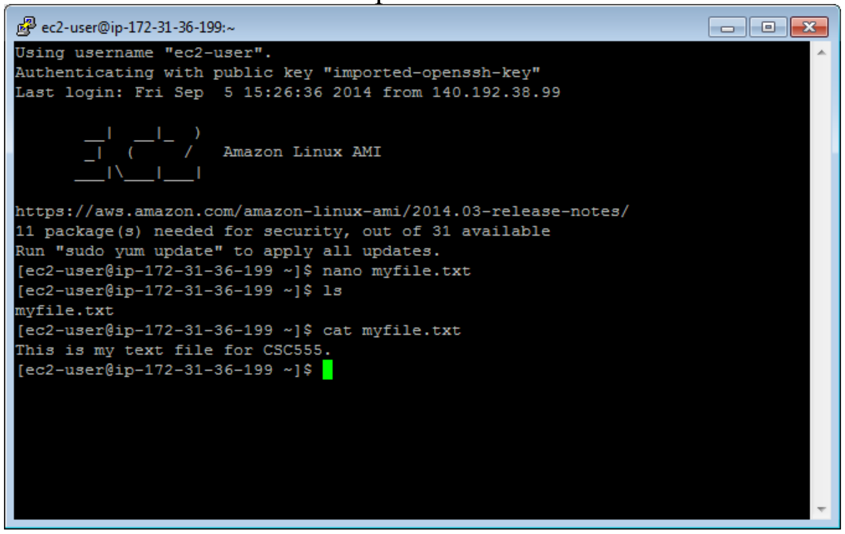
Exit: Ctrl-x Ctrl-z

Vim Instructions(Option 3):
• NOTE: When **vim** opens, you are in *command* mode. Any key you enter will be bound to a command instead of inserted into the file. To enter *insert* mode press the key “i”. To save a file or exit you will need to hit Esc to get back into *command* mode.

**$ vim myfile.txt**
Type “i” to enter *insert* mode
Type something into the file: “This is my text file for CSC555.”
Save changes: hit Esc to enter *command* mode then type “:w”
Exit: (still in *command* mode) type “:x”

Confirm your file has been saved by listing the files in the working directory.
**$ ls**
You should see your file.
Display the contents of the file on the screen.
**$ cat myfile.txt**

Your file contents should be printed to the terminal.



```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Fri Sep  5 15:26:36 2014 from 140.192.38.99

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2014.03-release-notes/
11 package(s) needed for security, out of 31 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-36-199 ~]$ nano myfile.txt
[ec2-user@ip-172-31-36-199 ~]$ ls
myfile.txt
[ec2-user@ip-172-31-36-199 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-36-199 ~]$
```

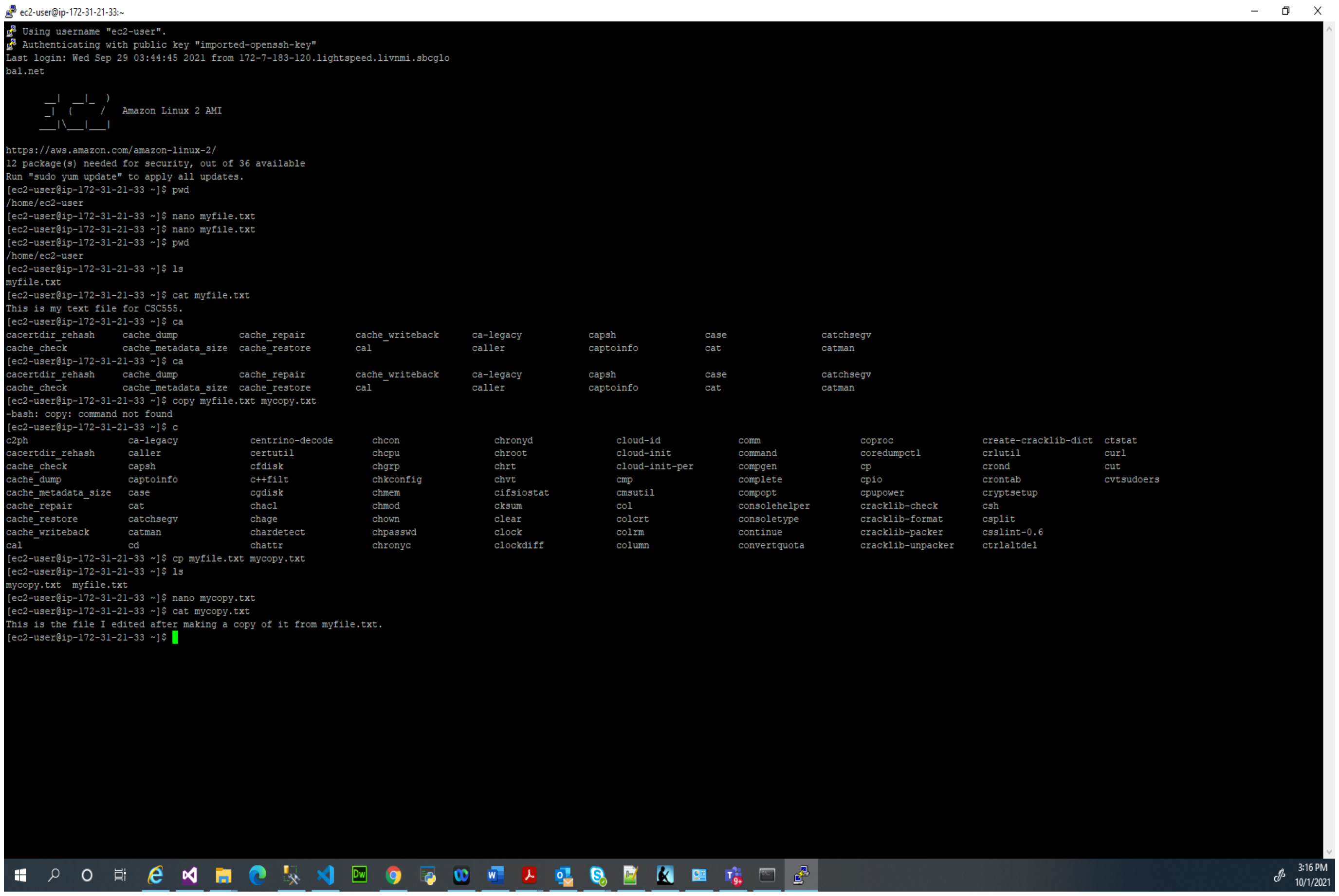
• **Tip**: Linux will fill in partially typed commands if you hit Tab.
  **$cat myfi**
  Hit Tab and “myfi” should be completed to “myfile.txt”. If there are multiple completion options, hit Tab twice and a list of all possible completions will be printed. This also applies to commands themselves, i.e. you can type in **ca** and see all possible commands that begin with ca.

## 2. Copy your file.

Make a copy.
**$ cp myfile.txt mycopy.txt**

Confirm this file has been created by listing the files in the working directory.
Edit this file so it contains different text than the original file using the text editor instructions, and confirm your changes by displaying the contents of the file on the screen.

**SUBMIT:** Take a screen shot of the <u>contents</u> of your copied file displayed on the terminal screen.



```
ec2-user@ip-172-31-21-33:~
```

```
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"
Last login: Wed Sep 29 03:44:45 2021 from 172-7-183-120.lightspeed.livnmi.sbcglo
bal.net

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
12 package(s) needed for security, out of 36 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-21-33 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-21-33 ~]$ nano myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ nano myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-21-33 ~]$ ls
myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-21-33 ~]$ ca
cacertdir_rehash     cache_dump           cache_repair         cache_writeback      ca-legacy            capsh                case                 catchsegv
cache_check          cache_metadata_size  cache_restore        cal                  caller               captoinfo            cat                  catman
[ec2-user@ip-172-31-21-33 ~]$ ca
cacertdir_rehash     cache_dump           cache_repair         cache_writeback      ca-legacy            capsh                case                 catchsegv
cache_check          cache_metadata_size  cache_restore        cal                  caller               captoinfo            cat                  catman
[ec2-user@ip-172-31-21-33 ~]$ copy myfile.txt mycopy.txt
-bash: copy: command not found
[ec2-user@ip-172-31-21-33 ~]$ c
c2ph                 ca-legacy            centrino-decode      chcon                chronyd              cloud-id             comm                 coproc               create-cracklib-dict ctstat
cacertdir_rehash     caller               certutil             chcpu                chroot               cloud-init           command              coredumpctl          crlutil              curl
cache_check          capsh                cfdisk               chgrp                chrt                 cloud-init-per       compgen              cp                   crond                cut
cache_dump           captoinfo            c++filt              chkconfig            chvt                 cmp                  complete             cpio                 crontab              cvtsudoers
cache_metadata_size  case                 cgdisk               chmem                cifsiostat           cmsutil              compopt              cpupower             cryptsetup
cache_repair         cat                  chacl                chmod                cksum                col                  consolehelper        cracklib-check       csh
cache_restore        catchsegv            chage                chown                clear                colcrt               consoletype          cracklib-format      csplit
cache_writeback      catman               chardetect           chpasswd             clock                colrm                continue             cracklib-packer      csslint-0.6
cal                  cd                   chattr               chronyc              clockdiff            column               convertquota         cracklib-unpacker    ctrlaltdel
[ec2-user@ip-172-31-21-33 ~]$ cp myfile.txt mycopy.txt
[ec2-user@ip-172-31-21-33 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ nano mycopy.txt
[ec2-user@ip-172-31-21-33 ~]$ cat mycopy.txt
This is the file I edited after making a copy of it from myfile.txt.
[ec2-user@ip-172-31-21-33 ~]$
```

## 3. Delete a file

Make a copy to delete.
**$ cp myfile.txt filetodelete.txt**
**$ ls**

Remove the file.
**$ rm filetodelete.txt**
**$ ls**

## 4. Create a directory to put your files.

Make a directory.
**$mkdir CSC555**

Change the current directory to your new directory.
**$cd CSC555**

Print your current working directory
**$pwd**

## 5. Move your files to your new directory.

Return to your home directory.
**$cd**
OR
**$cd ..**
OR
**$ cd /home/ec2-user/**

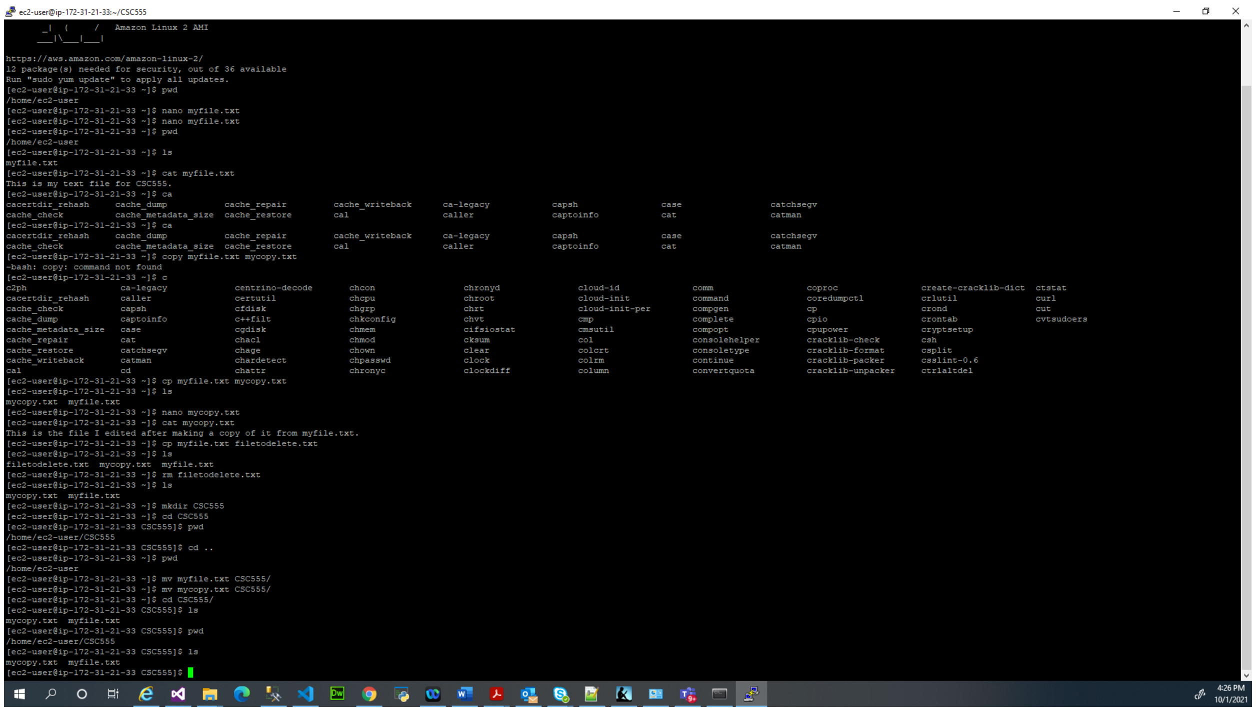
• NOTE: **cd** will always take you to your home directory. **cd ..** will move you up one directory level (to the parent). Your home directory is "/home/[user name]", /home/ec2-user in our case

Move your files to your new directory.

**$ mv myfile.txt CSC555/**
**$ mv mycopy.txt CSC555/**

Change the current directory to CSC555 and list the files in this directory.

**SUBMIT:** Take a screen shot of the files listed in the CSC555 directory.



```
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
12 package(s) needed for security, out of 36 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-21-33 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-21-33 ~]$ nano myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ nano myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-21-33 ~]$ ls
myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ cat myfile.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-21-33 ~]$ ca
cacertdir_rehash     cache_dump           cache_repair         cache_writeback      ca-legacy            capsh                case                 catchsegv
cache_check          cache_metadata_size  cache_restore        cal                  caller               captoinfo            cat                  catman
[ec2-user@ip-172-31-21-33 ~]$ ca
cacertdir_rehash     cache_dump           cache_repair         cache_writeback      ca-legacy            capsh                case                 catchsegv
cache_check          cache_metadata_size  cache_restore        cal                  caller               captoinfo            cat                  catman
[ec2-user@ip-172-31-21-33 ~]$ copy myfile.txt mycopy.txt
-bash: copy: command not found
[ec2-user@ip-172-31-21-33 ~]$ c
c2ph                 ca-legacy            centrino-decode      chcon                chronyd              cloud-id             comm                 coproc               create-cracklib-dict ctstat
cacertdir_rehash     caller               certutil             chcpu                chroot               cloud-init           command              coredumpctl          crlutil              curl
cache_check          capsh                cfdisk               chgrp                chrt                 cloud-init-per       compgen              cp                   crond                cut
cache_dump           captoinfo            c++filt              chkconfig            chvt                 cmp                  complete             cpio                 crontab              cvtsudoers
cache_metadata_size  case                 cgdisk               chmem                cifsiostat           cmsutil              compopt              cpupower             cryptsetup
cache_repair         cat                  chacl                chmod                cksum                col                  consolehelper        cracklib-check       csh
cache_restore        catchsegv            chage                chown                clear                colcrt               consoletype          cracklib-format      csplit
cache_writeback      catman               chardetect           chpasswd             clock                colrm                continue             cracklib-packer      csslint-0.6
cal                  cd                   chattr               chronyc              clockdiff            column               convertquota         cracklib-unpacker    ctrlaltdel
[ec2-user@ip-172-31-21-33 ~]$ cp myfile.txt mycopy.txt
[ec2-user@ip-172-31-21-33 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ nano mycopy.txt
[ec2-user@ip-172-31-21-33 ~]$ cat mycopy.txt
This is the file I edited after making a copy of it from myfile.txt.
[ec2-user@ip-172-31-21-33 ~]$ cp myfile.txt filetodelete.txt
[ec2-user@ip-172-31-21-33 ~]$ ls
filetodelete.txt  mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ rm filetodelete.txt
[ec2-user@ip-172-31-21-33 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ mkdir CSC555
[ec2-user@ip-172-31-21-33 ~]$ cd CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ cd ..
[ec2-user@ip-172-31-21-33 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-21-33 ~]$ mv myfile.txt CSC555/
[ec2-user@ip-172-31-21-33 ~]$ mv mycopy.txt CSC555/
[ec2-user@ip-172-31-21-33 ~]$ cd CSC555/
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$
```

## 6. Zip and Unzip your files.

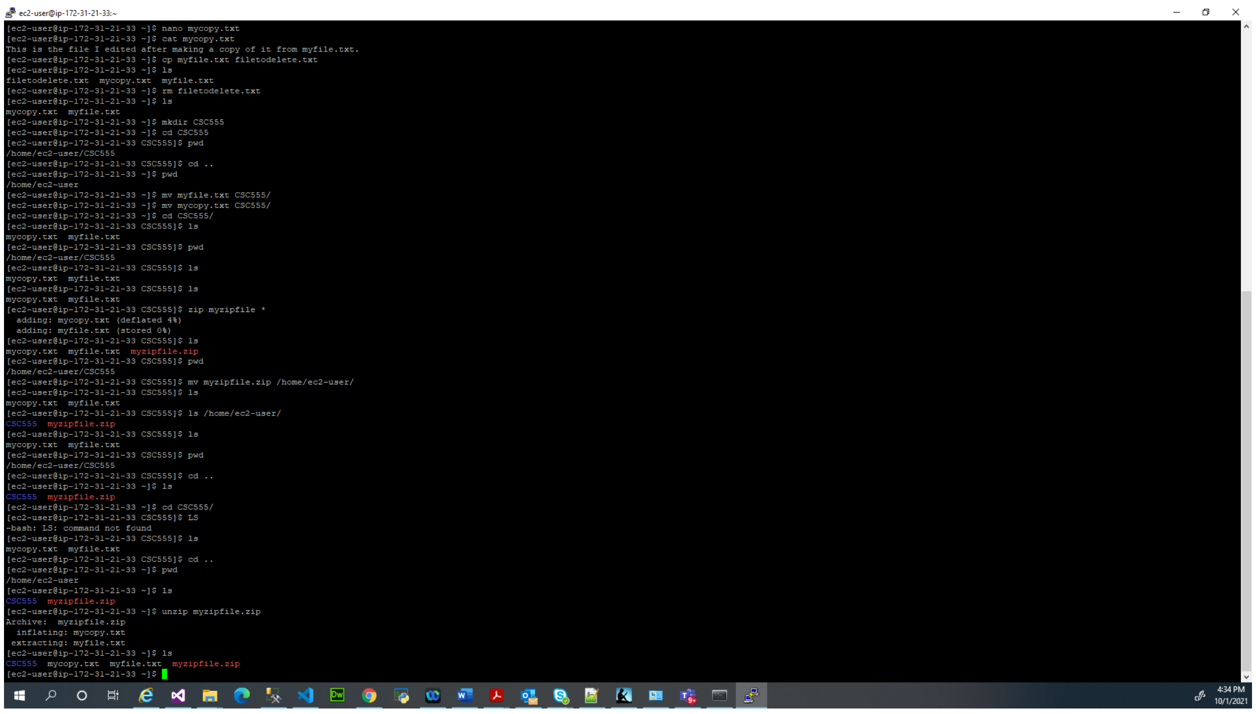
Zip the files.
**$ zip myzipfile mycopy.txt myfile.txt**
OR
**$ zip myzipfile ***

• NOTE: * is the wildcard symbol that matches <u>everything</u> in current directory. If there should be any additional files in the current directory, they will also be placed into the zip archive. Wildcard can also be used to match files selectively. For example **zip myzipfile my*** will zip-up all files beginning with "my" in the current directory.

Move your zip file to your home directory.
**$ mv myzipfile.zip /home/ec2-user/**

Return to your home directory.
Extract the files.
**$ unzip myzipfile.zip**

**SUBMIT:** Take a screen shot of the screen after this command.



```
ec2-user@ip-172-31-21-33:~
[ec2-user@ip-172-31-21-33 ~]$ nano mycopy.txt
[ec2-user@ip-172-31-21-33 ~]$ cat mycopy.txt
This is the file I edited after making a copy of it from myfile.txt.
[ec2-user@ip-172-31-21-33 ~]$ cp myfile.txt filetodelete.txt
[ec2-user@ip-172-31-21-33 ~]$ ls
filetodelete.txt  mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ rm filetodelete.txt
[ec2-user@ip-172-31-21-33 ~]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ mkdir CSC555
[ec2-user@ip-172-31-21-33 ~]$ cd CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ cd ..
[ec2-user@ip-172-31-21-33 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-21-33 ~]$ mv myfile.txt CSC555/
[ec2-user@ip-172-31-21-33 ~]$ mv mycopy.txt CSC555/
[ec2-user@ip-172-31-21-33 ~]$ cd CSC555/
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ zip myzipfile *
  adding: mycopy.txt (deflated 4%)
  adding: myfile.txt (stored 0%)
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt  myzipfile.zip
[ec2-user@ip-172-31-21-33 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ mv myzipfile.zip /home/ec2-user/
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ ls /home/ec2-user/
CSC555  myzipfile.zip
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ cd ..
[ec2-user@ip-172-31-21-33 ~]$ ls
CSC555  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ cd CSC555/
[ec2-user@ip-172-31-21-33 CSC555]$ LS
-bash: LS: command not found
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ cd ..
[ec2-user@ip-172-31-21-33 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-21-33 ~]$ ls
CSC555  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ unzip myzipfile.zip
Archive:  myzipfile.zip
  inflating: mycopy.txt
 extracting: myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ ls
CSC555  mycopy.txt  myfile.txt  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ 
```

**7. Remove your CSC555 directory.**

• **Warning**: Executing "rm -rf" has the potential to delete ALL files in a given directory, including sub-directories ("r" stands for recursive). You should use this command very carefully.
Delete your CSC555 directory.
**$ rm -rf CSC555/**

**8. Download a file from the web.**

Download the script for Monty Python and the Holy Grail.
**$ wget http://www.textfiles.com/media/SCRIPTS/grail**

The file should be saved as "grail" by default.

**9. ls formats**

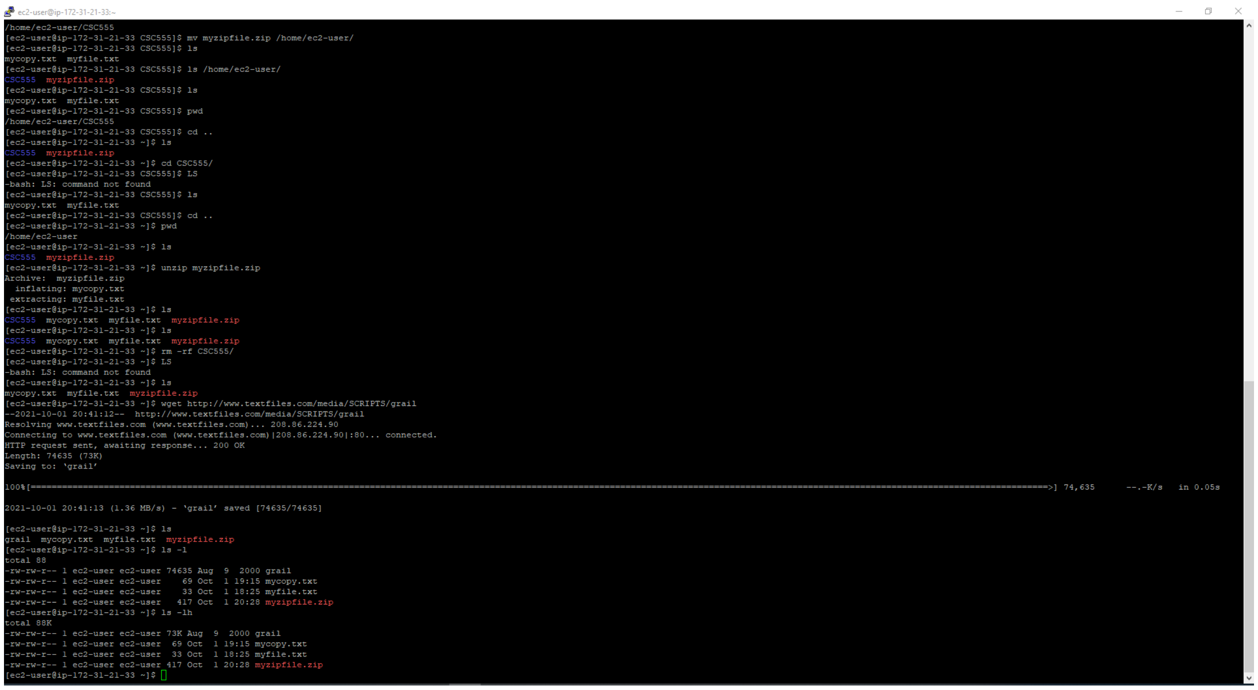
List all contents of the current directory in long list format.
• **Note**: the option following "ls" is the character "l"; not "one".
**$ ls -l**

The 1st column gives information regarding file permissions (which we will discuss in more detail later). For now, note that the first character of the 10 total will be "-" for normal files and "d" for directories. The 2nd Column is the number of links to the file. The 3rd and 4th columns are the owner and the group of the file. The 5th column displays the size of the file in bytes. The 6th column is the date and time the file was last modified. The 7th column is the file or directory name.

List all contents of the current directory in long list and human readable formats. "-h" will put large files in more readable units than bytes.
**$ ls -lh**

**SUBMIT:** The size of the grail file.

The size of the grail file is 73K as shown below:



```
/home/ec2-user/CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ mv myzipfile.zip /home/ec2-user/
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ ls /home/ec2-user/
CSC555  myzipfile.zip
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ pwd
/home/ec2-user/CSC555
[ec2-user@ip-172-31-21-33 CSC555]$ cd ..
[ec2-user@ip-172-31-21-33 ~]$ ls
CSC555  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ cd CSC555/
[ec2-user@ip-172-31-21-33 CSC555]$ LS
-bash: LS: command not found
[ec2-user@ip-172-31-21-33 CSC555]$ ls
mycopy.txt  myfile.txt
[ec2-user@ip-172-31-21-33 CSC555]$ cd ..
[ec2-user@ip-172-31-21-33 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-21-33 ~]$ ls
CSC555  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ unzip myzipfile.zip
Archive:  myzipfile.zip
  inflating: mycopy.txt
 extracting: myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ ls
CSC555  mycopy.txt  myfile.txt  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ ls
CSC555  mycopy.txt  myfile.txt  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ rm -rf CSC555/
[ec2-user@ip-172-31-21-33 ~]$ LS
-bash: LS: command not found
[ec2-user@ip-172-31-21-33 ~]$ ls
mycopy.txt  myfile.txt  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ wget http://www.textfiles.com/media/SCRIPTS/grail
--2021-10-01 20:41:12--  http://www.textfiles.com/media/SCRIPTS/grail
Resolving www.textfiles.com (www.textfiles.com)... 208.86.224.90
Connecting to www.textfiles.com (www.textfiles.com)|208.86.224.90|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 74635 (73K)
Saving to: 'grail'

100%[==============================================================================================================================================================================================>] 74,635      --.-K/s   in 0.05s

2021-10-01 20:41:13 (1.36 MB/s) - 'grail' saved [74635/74635]

[ec2-user@ip-172-31-21-33 ~]$ ls
grail  mycopy.txt  myfile.txt  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ ls -l
total 88
-rw-rw-r-- 1 ec2-user ec2-user 74635 Aug  9  2000 grail
-rw-rw-r-- 1 ec2-user ec2-user    69 Oct  1 19:15 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user    33 Oct  1 18:25 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user   417 Oct  1 20:28 myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ ls -lh
total 88K
-rw-rw-r-- 1 ec2-user ec2-user 73K Aug  9  2000 grail
-rw-rw-r-- 1 ec2-user ec2-user  69 Oct  1 19:15 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user  33 Oct  1 18:25 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user 417 Oct  1 20:28 myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ 
```

**10. More on viewing files.**

If you issue "cat grail", the contents of grail will be printed. However, this file is too large to fit on the screen.

Show the grail file one page at a time.

```
$ more grail
```

Hit the spacebar to go to the next page. Type "b" to go page up, hit "space" key to go page down. Type "q" to quit.

OR

```
$ less grail
```

*Less* has more options than *more* ("*less* is more and *more* is less"). You can now use the keyboard Arrows and Page Up/Down to scroll. You can type "h" for help, which will display additional options.

View the line numbers in the grail file. The *cat* command has the -n option, which prints line numbers, but you may also want to use *more* to view the file one page at a time. A solution is to *pipe* the output from *cat* to *more*. A *pipe* redirects the output from one program to another program for further processing, and it is represented with "|".

```
$ cat -n grail | more
```

Redirect the standard output (stdout) of a command.

```
$ cat myfile.txt > redirect1.txt
$ ls -lh > redirect2.txt
```

Append the stdout to a file.

```
$ cat mycopy.txt >> myfile.txt
```

mycopy.txt will be appended to myfile.txt.

• **Note**: "cat mycopy.txt > myfile.txt" will overwrite myfile.txt with the contents output by "cat mycopy.txt". Thus using >> is crucial if you want to preserve the existing file contents.

**11. Change access permissions to objects with the *change mode* command.**

The following represent roles:
u – user, g – group, o – others, a - all

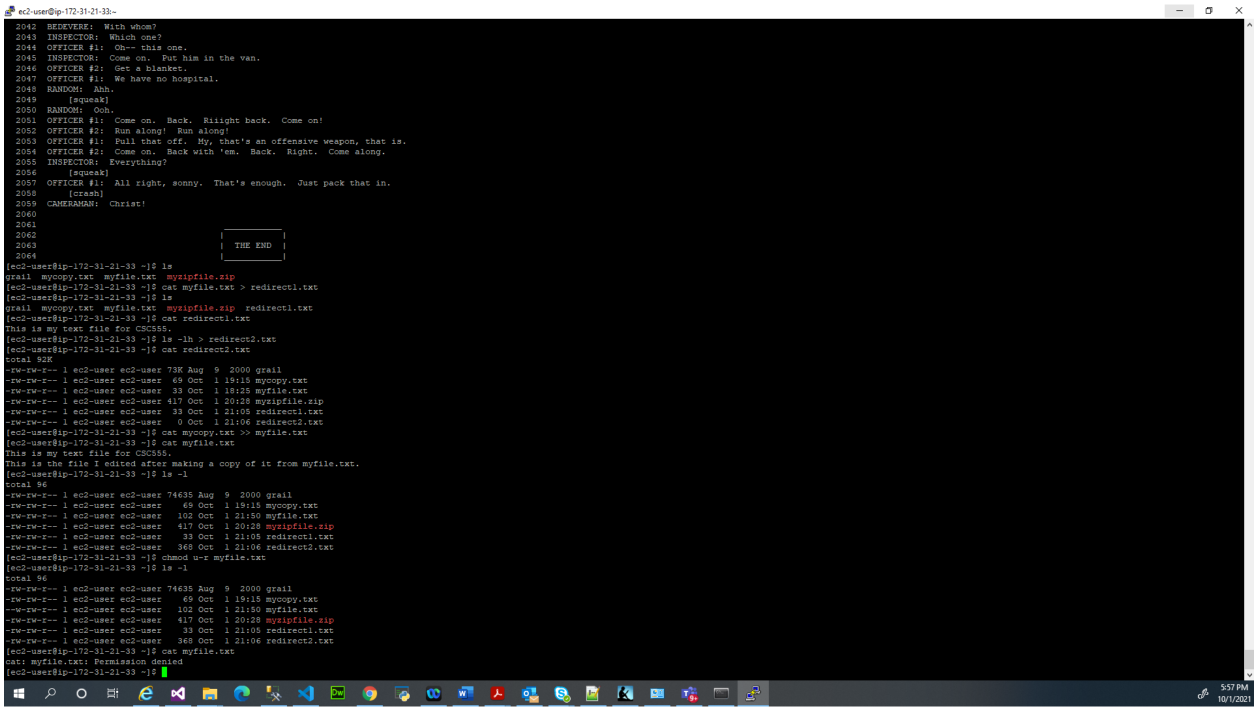
The following represent permissions:
r – read, w – write, x – execute

Remove the read permission for your user on a file.

```
$ chmod u-r myfile.txt
```

Try to read this file. You should receive a "permission denied" message because you are the user who owns the file.

**SUBMIT:** The screenshot of the permission denied error



```
 2042  BEDEVERE:  With whom?
 2043  INSPECTOR:  Which one?
 2044  OFFICER #1:  Oh-- this one.
 2045  INSPECTOR:  Come on.  Put him in the van.
 2046  OFFICER #2:  Get a blanket.
 2047  OFFICER #1:  We have no hospital.
 2048  RANDOM:  Ahh.
 2049      [squeak]
 2050  RANDOM:  Ooh.
 2051  OFFICER #1:  Come on.  Back.  Riiight back.  Come on!
 2052  OFFICER #2:  Run along!  Run along!
 2053  OFFICER #1:  Pull that off.  My, that's an offensive weapon, that is.
 2054  OFFICER #2:  Come on.  Back with 'em.  Back.  Right.  Come along.
 2055  INSPECTOR:  Everything?
 2056      [squeak]
 2057  OFFICER #1:  All right, sonny.  That's enough.  Just pack that in.
 2058      [crash]
 2059  CAMERAMAN:  Christ!
 2060
 2061                                      __________
 2062                                     |          |
 2063                                     |  THE END |
 2064                                     |__________|
[ec2-user@ip-172-31-21-33 ~]$ ls
grail  mycopy.txt  myfile.txt  myzipfile.zip
[ec2-user@ip-172-31-21-33 ~]$ cat myfile.txt > redirect1.txt
[ec2-user@ip-172-31-21-33 ~]$ ls
grail  mycopy.txt  myfile.txt  myzipfile.zip  redirect1.txt
[ec2-user@ip-172-31-21-33 ~]$ cat redirect1.txt
This is my text file for CSC555.
[ec2-user@ip-172-31-21-33 ~]$ ls -lh > redirect2.txt
[ec2-user@ip-172-31-21-33 ~]$ cat redirect2.txt
total 92K
-rw-rw-r-- 1 ec2-user ec2-user 73K Aug  9  2000 grail
-rw-rw-r-- 1 ec2-user ec2-user  69 Oct  1 19:15 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user  33 Oct  1 18:25 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user 417 Oct  1 20:28 myzipfile.zip
-rw-rw-r-- 1 ec2-user ec2-user  33 Oct  1 21:05 redirect1.txt
-rw-rw-r-- 1 ec2-user ec2-user   0 Oct  1 21:06 redirect2.txt
[ec2-user@ip-172-31-21-33 ~]$ cat mycopy.txt >> myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ cat myfile.txt
This is my text file for CSC555.
This is the file I edited after making a copy of it from myfile.txt.
[ec2-user@ip-172-31-21-33 ~]$ ls -l
total 96
-rw-rw-r-- 1 ec2-user ec2-user 74635 Aug  9  2000 grail
-rw-rw-r-- 1 ec2-user ec2-user    69 Oct  1 19:15 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user   102 Oct  1 21:50 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user   417 Oct  1 20:28 myzipfile.zip
-rw-rw-r-- 1 ec2-user ec2-user    33 Oct  1 21:05 redirect1.txt
-rw-rw-r-- 1 ec2-user ec2-user   368 Oct  1 21:06 redirect2.txt
[ec2-user@ip-172-31-21-33 ~]$ chmod u-r myfile.txt
[ec2-user@ip-172-31-21-33 ~]$ ls -l
total 96
-rw-rw-r-- 1 ec2-user ec2-user 74635 Aug  9  2000 grail
-rw-rw-r-- 1 ec2-user ec2-user    69 Oct  1 19:15 mycopy.txt
--w-rw-r-- 1 ec2-user ec2-user   102 Oct  1 21:50 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user   417 Oct  1 20:28 myzipfile.zip
-rw-rw-r-- 1 ec2-user ec2-user    33 Oct  1 21:05 redirect1.txt
-rw-rw-r-- 1 ec2-user ec2-user   368 Oct  1 21:06 redirect2.txt
[ec2-user@ip-172-31-21-33 ~]$ cat myfile.txt
cat: myfile.txt: Permission denied
[ec2-user@ip-172-31-21-33 ~]$
```

Give your user read permission on a file. Use the same file you removed the read permission from.

**$ chmod u+r myfile.txt**

You should now be able to read this file again.

## 12. Python examples

Install Python if it is not available on your machine.

**$ sudo yum install python**

Create a Python file. These instructions will use Emacs as a text editor, but you can still chose the text editor you want.

**$ emacs lucky.py**

(Write a simple Python program)

```
print "*"*25
print "My Lucky Numbers".rjust(20)
print "*"*25

for i in range(10):
   lucky_nbr = (i + 1)*2
   print "My lucky number is %s!" % lucky_nbr
```

Run your Python program.

**$ python lucky.py**

Redirect your output to a file
**$ python lucky.py > lucky.txt**

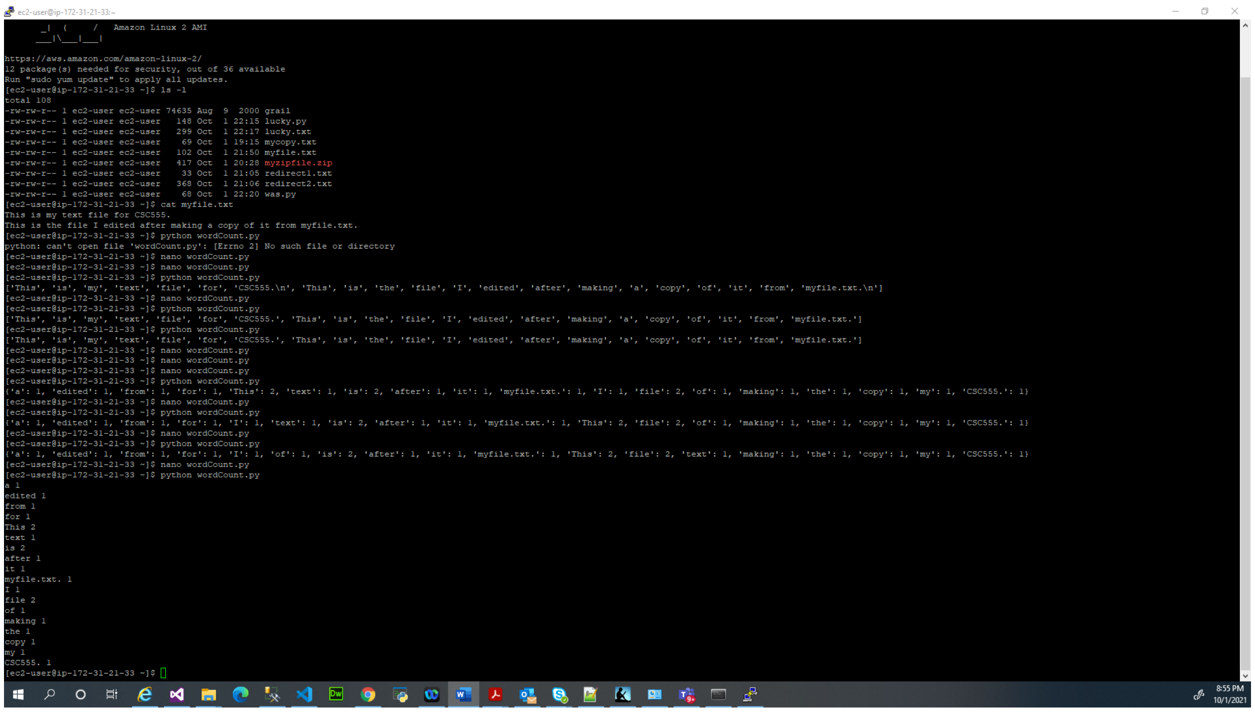
Pipe the stdout from lucky.py to another Python program that will replace “is” with “was”.
**$ emacs was.py**

```
import sys

for line in sys.stdin:
   print line.replace("is", "was")
```

**$ python lucky.py | python was.py**

Write python code to read a text file (you can use myfile.txt) and output a word count total for each word with the number of times that word occurs in the entire file. That is, if the file has the word “Hadoop” occurs in the file 5 times, your code should print “Hadoop 5”. It should output the count of all words occurring in a file.

**SUBMIT:** The screen output from running your python code and a copy of your python code. Homework submissions without code will receive no credit.



```
ec2-user@ip-172-31-21-33:~
     _|  (     /   Amazon Linux 2 AMI
    ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
12 package(s) needed for security, out of 36 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-21-33 ~]$ ls -l
total 108
-rw-rw-r-- 1 ec2-user ec2-user 74635 Aug  9  2000 grail
-rw-rw-r-- 1 ec2-user ec2-user   148 Oct  1 22:15 lucky.py
-rw-rw-r-- 1 ec2-user ec2-user   299 Oct  1 22:17 lucky.txt
-rw-rw-r-- 1 ec2-user ec2-user    69 Oct  1 19:15 mycopy.txt
-rw-rw-r-- 1 ec2-user ec2-user   102 Oct  1 21:50 myfile.txt
-rw-rw-r-- 1 ec2-user ec2-user   417 Oct  1 20:28 myzipfile.zip
-rw-rw-r-- 1 ec2-user ec2-user    33 Oct  1 21:05 redirect1.txt
-rw-rw-r-- 1 ec2-user ec2-user   368 Oct  1 21:06 redirect2.txt
-rw-rw-r-- 1 ec2-user ec2-user    68 Oct  1 22:20 was.py
[ec2-user@ip-172-31-21-33 ~]$ cat myfile.txt
This is my text file for CSC555.
This is the file I edited after making a copy of it from myfile.txt.
[ec2-user@ip-172-31-21-33 ~]$ python wordCount.py
python: can't open file 'wordCount.py': [Errno 2] No such file or directory
[ec2-user@ip-172-31-21-33 ~]$ nano wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ nano wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ python wordCount.py
['This', 'is', 'my', 'text', 'file', 'for', 'CSC555.\n', 'This', 'is', 'the', 'file', 'I', 'edited', 'after', 'making', 'a', 'copy', 'of', 'it', 'from', 'myfile.txt.\n']
[ec2-user@ip-172-31-21-33 ~]$ nano wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ python wordCount.py
['This', 'is', 'my', 'text', 'file', 'for', 'CSC555.', 'This', 'is', 'the', 'file', 'I', 'edited', 'after', 'making', 'a', 'copy', 'of', 'it', 'from', 'myfile.txt.']
[ec2-user@ip-172-31-21-33 ~]$ python wordCount.py
['This', 'is', 'my', 'text', 'file', 'for', 'CSC555.', 'This', 'is', 'the', 'file', 'I', 'edited', 'after', 'making', 'a', 'copy', 'of', 'it', 'from', 'myfile.txt.']
[ec2-user@ip-172-31-21-33 ~]$ nano wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ nano wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ nano wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ python wordCount.py
{'a': 1, 'edited': 1, 'from': 1, 'for': 1, 'This': 2, 'text': 1, 'is': 2, 'after': 1, 'it': 1, 'myfile.txt.': 1, 'I': 1, 'file': 2, 'of': 1, 'making': 1, 'the': 1, 'copy': 1, 'my': 1, 'CSC555.': 1}
[ec2-user@ip-172-31-21-33 ~]$ nano wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ python wordCount.py
{'a': 1, 'edited': 1, 'from': 1, 'for': 1, 'I': 1, 'text': 1, 'is': 2, 'after': 1, 'it': 1, 'myfile.txt.': 1, 'This': 2, 'file': 2, 'of': 1, 'making': 1, 'the': 1, 'copy': 1, 'my': 1, 'CSC555.': 1}
[ec2-user@ip-172-31-21-33 ~]$ nano wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ python wordCount.py
{'a': 1, 'edited': 1, 'from': 1, 'for': 1, 'I': 1, 'of': 1, 'is': 2, 'after': 1, 'it': 1, 'myfile.txt.': 1, 'This': 2, 'file': 2, 'text': 1, 'making': 1, 'the': 1, 'copy': 1, 'my': 1, 'CSC555.': 1}
[ec2-user@ip-172-31-21-33 ~]$ nano wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ python wordCount.py
a 1
edited 1
from 1
for 1
This 2
text 1
is 2
after 1
it 1
myfile.txt. 1
I 1
file 2
of 1
making 1
the 1
copy 1
my 1
CSC555. 1
[ec2-user@ip-172-31-21-33 ~]$
```

A copy of my python code:

```
def readFile(fileName):
        """Function that returns a list of words after reading a file"""
        bagOfWordsL = []
        with open(fileName) as f:
                # read the file and generate a list of strings
                lines = f.readlines()

                for i in range(len(lines)):
                        # a list of words is created by splitting the
string
                        splittedWordsL = lines[i].split(' ')

                        # loop through the list of words and add words to
the bag of words list
                        for splittedItem in range(len(splittedWordsL)):
                                # if there are any empty spaces just pass
                                if len(splittedWordsL[splittedItem]) == 0:
                                        pass
                                else:
                                        # used a replace function to
remove all line breaks.

bagOfWordsL.append(splittedWordsL[splittedItem].replace('\n',''))

        return bagOfWordsL      # return a list of bag of words.


def wordCnt(listOfWords):
    """Function that generates a dictionary of words with their frequency
of occurance"""
    wordCnt = {}
# define an empty dictionary
    for i in range(len(listOfWords)):
# iterate through the list of words and count the number of words
        if listOfWords[i] in wordCnt.keys():
            wordCnt[listOfWords[i]] = wordCnt[listOfWords[i]] + 1
        else:
            wordCnt[listOfWords[i]] = 1
    return wordCnt
# return a dictionary of the different  words with their frequencies.

listOfBagOfWords = readFile('myfile.txt')
# return a list of words
wordsCntDict = wordCnt(listOfBagOfWords)
# uses a helper function that returns  a dictionary of  the different
words with their frequencies

# sort the dictionary
sortedWordCntDict = {key: value for (key, value) in
sorted(wordsCntDict.items(), key=lambda wordCnt: wordCnt[1], reverse =
True)}
for key in sortedWordCntDict:
        print('%s %s' %(key, sortedWordCntDict[key]))
```

## Part 3: Wordcount

For this part of the assignment, you will run wordcount on a single-node Hadoop instance.  I am going to provide detailed instructions to help you get Hadoop running. The instructions are following Hadoop: The Definitive Guide instructions presented in Appendix A: Installing Apache Hadoop.

You can download 2.6.4 from here. You can copy-paste these commands (right-click in PuTTy to paste, but please watch out for error messages and run commands one by one)

Install ant to list java processes
**sudo yum install ant**

(wget command stands for "web get" and lets you download files to your instance from a URL link)
**wget http:// dbgroup.cdm.depaul.edu/Courses/CSC555/hadoop-2.6.4.tar.gz**

(unpack the archive)
**tar xzf hadoop-2.6.4.tar.gz**

Modify the conf/hadoop-env.sh to add to it the JAVA_HOME configuration
You can open it by running (using nano or your favorite editor instead of nano).
**nano hadoop-2.6.4/etc/hadoop/hadoop-env.sh**
Note that the # comments out the line, so you would comment out the original JAVA_HOME line replacing it by the new one as below.

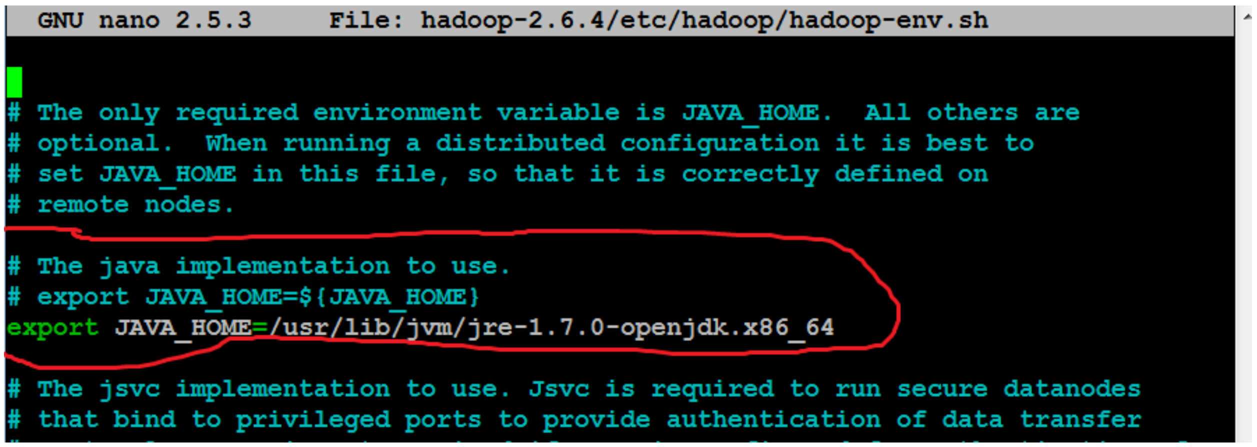
**NOTE**: you would need to determine the correct Java configuration line by executing the following (underlined) command

```
[ec2-user@ip-172-31-16-63 ~]$ readlink -f $(which java)
```

which will output something like:

```
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.191.b12-0.amzn2.x86_64/jre/bin/java
```

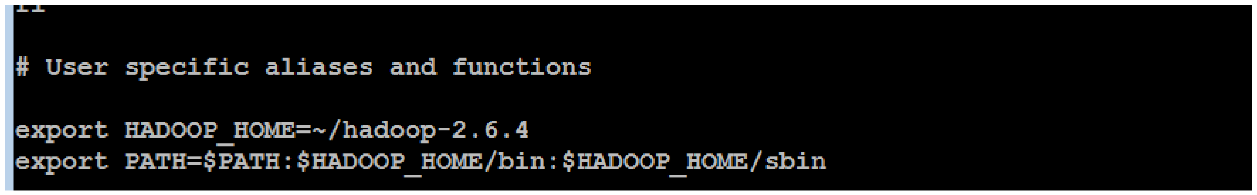
In my case, Java home is at (**remove the bin/java from the path**):
**/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.191.b12-0.amzn2.x86_64/jre/**



```
  GNU nano 2.5.3      File: hadoop-2.6.4/etc/hadoop/hadoop-env.sh


# The only required environment variable is JAVA_HOME.  All others are
# optional.  When running a distributed configuration it is best to
# set JAVA_HOME in this file, so that it is correctly defined on
# remote nodes.

# The java implementation to use.
# export JAVA_HOME=${JAVA_HOME}
export JAVA_HOME=/usr/lib/jvm/jre-1.7.0-openjdk.x86_64

# The jsvc implementation to use. Jsvc is required to run secure datanodes
# that bind to privileged ports to provide authentication of data transfer
```

modify the .bashrc file to add these two lines:
export HADOOP_HOME=~/hadoop-2.6.4
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

.bashrc file contains environment settings to be configured automatically on each login. You can open the .bashrc file by running
**nano ~/.bashrc**



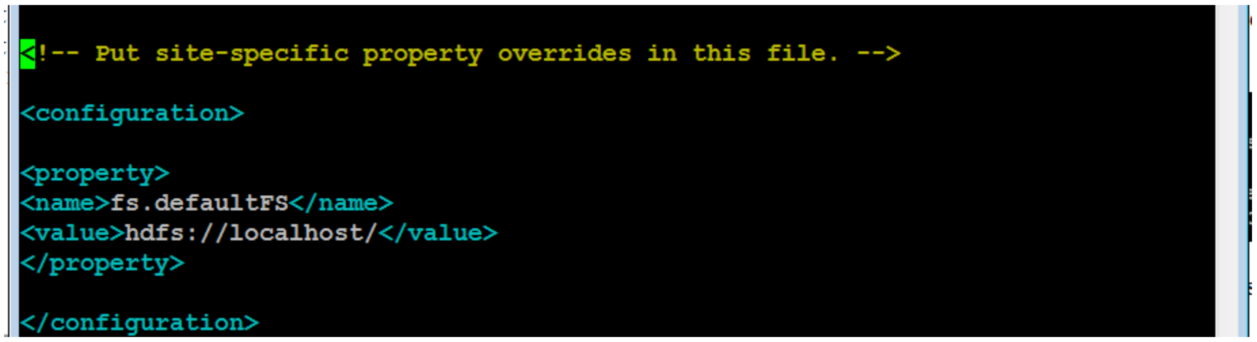
```
# User specific aliases and functions

export HADOOP_HOME=~/hadoop-2.6.4
export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin
```

To immediately refresh the settings (that will be <u>automatic on next login</u>), run
**source ~/.bashrc**

Next, follow the instructions for Pseudodistributed Mode for all 4 files.

(to edit the first config file)
**nano hadoop-2.6.4/etc/hadoop/core-site.xml**

Make sure you paste the settings between the <configuration> and </configuration> tags, like in the screenshot below. NOTE: The screenshot below is only one of the 4 files, all files are different. The contents of each file are described in the **Appendix A** in the Hadoop book, the relevant appendix is also included with the homework assignment. I am also including a .txt file (HadoopConfigurationText) so that it is easier to copy-paste.



```
<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
<name>fs.defaultFS</name>
<value>hdfs://localhost/</value>
</property>

</configuration>
```

**nano hadoop-2.6.4/etc/hadoop/hdfs-site.xml**
(mapred-site.xml file is not there, run the following single line command to create it by copying from template. Then you can edit it as other files.)
**<u>cp hadoop-2.6.4/etc/hadoop/mapred-site.xml.template  hadoop-2.6.4/etc/hadoop/mapred-site.xml</u>**
**nano hadoop-2.6.4/etc/hadoop/mapred-site.xml**
**nano hadoop-2.6.4/etc/hadoop/yarn-site.xml**

To enable passwordless ssh access (we will discuss SSH and public/private keys in class), run these commands:
**ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa**
**cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys**

test by running (and confirming yes to a one-time warning)
**ssh localhost**
**exit**

Format HDFS (i.e., first time initialize)

**hdfs namenode -format**

Start HDFS, Hadoop and history server (answer a 1-time yes if you asked about host authenticity)

**start-dfs.sh**
**start-yarn.sh**
**mr-jobhistory-daemon.sh start historyserver**

Verify if everything is running:
**jps**

(NameNode and DataNode are responsible for HDFS management; NodeManager and ResourceManager are serving the function similar to JobTracker and TaskTracker.)

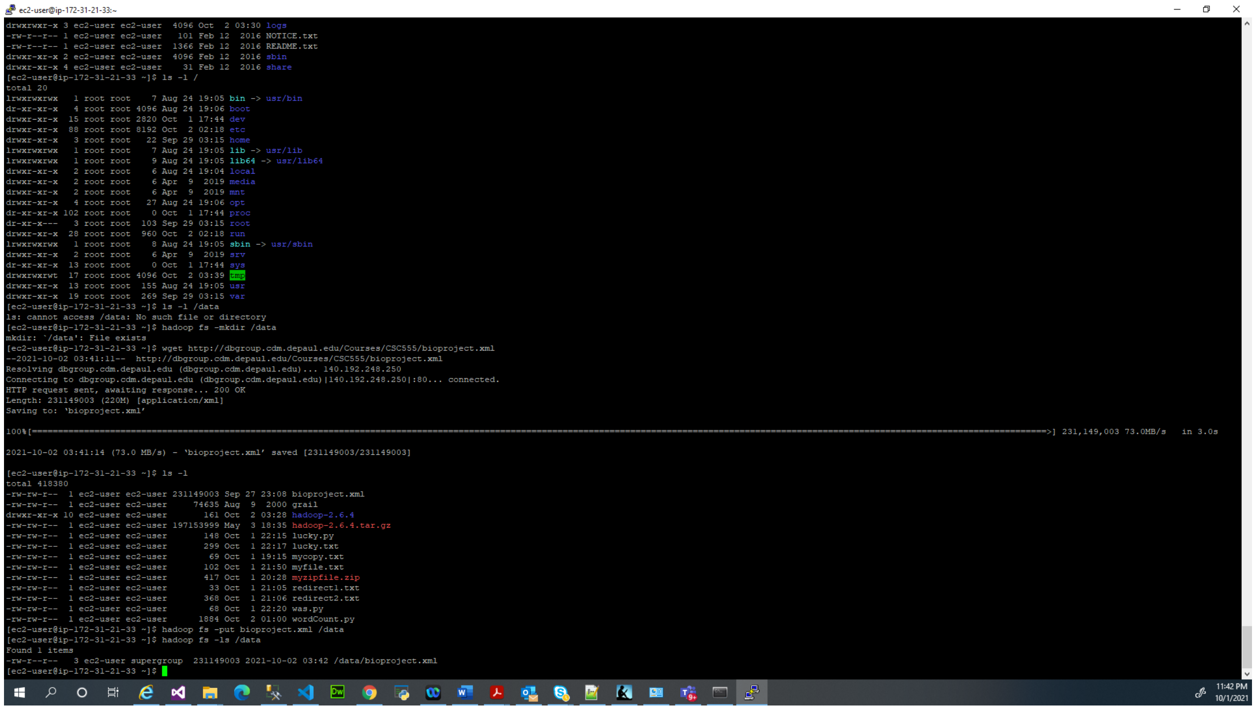
Create a destination directory
hadoop fs -mkdir /data

Download a large text file using
**wget** http://dbgroup.cdm.depaul.edu/Courses/CSC555/bioproject.xml

Copy the file to HDFS for processing

**hadoop fs -put bioproject.xml /data/**

(you can optimally verify that the file was uploaded to HDFS by **hadoop fs -ls /data**)
**Submit a screenshot of this command**

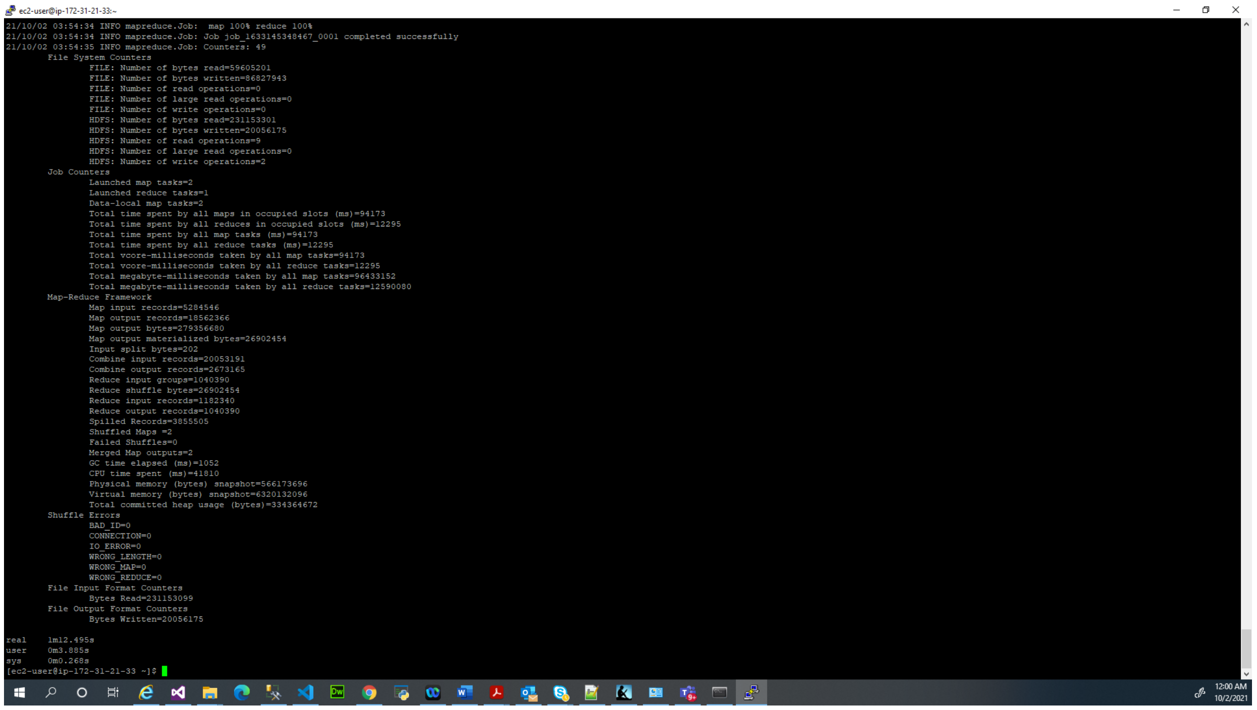


```
drwxrwxr-x 3 ec2-user ec2-user  4096 Oct  2 03:30 logs
-rw-r--r-- 1 ec2-user ec2-user   101 Feb 12  2016 NOTICE.txt
-rw-r--r-- 1 ec2-user ec2-user  1366 Feb 12  2016 README.txt
drwxr-xr-x 2 ec2-user ec2-user  4096 Feb 12  2016 sbin
drwxr-xr-x 4 ec2-user ec2-user    31 Feb 12  2016 share
[ec2-user@ip-172-31-21-33 ~]$ ls -l /
total 20
lrwxrwxrwx   1 root root    7 Aug 24 19:05 bin -> usr/bin
dr-xr-xr-x   4 root root 4096 Aug 24 19:06 boot
drwxr-xr-x  15 root root 2820 Oct  1 17:44 dev
drwxr-xr-x  88 root root 8192 Oct  2 02:18 etc
drwxr-xr-x   3 root root   22 Sep 29 03:15 home
lrwxrwxrwx   1 root root    7 Aug 24 19:05 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Aug 24 19:05 lib64 -> usr/lib64
drwxr-xr-x   2 root root    6 Aug 24 19:04 local
drwxr-xr-x   2 root root    6 Apr  9  2019 media
drwxr-xr-x   2 root root    6 Apr  9  2019 mnt
drwxr-xr-x   4 root root   27 Aug 24 19:06 opt
dr-xr-xr-x 102 root root    0 Oct  1 17:44 proc
dr-xr-x---   3 root root  103 Sep 29 03:15 root
drwxr-xr-x  28 root root  960 Oct  2 02:18 run
lrwxrwxrwx   1 root root    8 Aug 24 19:05 sbin -> usr/sbin
drwxr-xr-x   2 root root    6 Apr  9  2019 srv
dr-xr-xr-x  13 root root    0 Oct  1 17:44 sys
drwxrwxrwt  17 root root 4096 Oct  2 03:39 tmp
drwxr-xr-x  13 root root  155 Aug 24 19:05 usr
drwxr-xr-x  19 root root  269 Sep 29 03:15 var
[ec2-user@ip-172-31-21-33 ~]$ ls -l /data
ls: cannot access /data: No such file or directory
[ec2-user@ip-172-31-21-33 ~]$ hadoop fs -mkdir /data
mkdir: `/data': File exists
[ec2-user@ip-172-31-21-33 ~]$ wget http://dbgroup.cdm.depaul.edu/Courses/CSC555/bioproject.xml
--2021-10-02 03:41:11--  http://dbgroup.cdm.depaul.edu/Courses/CSC555/bioproject.xml
Resolving dbgroup.cdm.depaul.edu (dbgroup.cdm.depaul.edu)... 140.192.248.250
Connecting to dbgroup.cdm.depaul.edu (dbgroup.cdm.depaul.edu)|140.192.248.250|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 231149003 (220M) [application/xml]
Saving to: 'bioproject.xml'

100%[=========================================================================================================================================================>] 231,149,003 73.0MB/s   in 3.0s

2021-10-02 03:41:14 (73.0 MB/s) - 'bioproject.xml' saved [231149003/231149003]

[ec2-user@ip-172-31-21-33 ~]$ ls -l
total 418380
-rw-rw-r--  1 ec2-user ec2-user 231149003 Sep 27 23:08 bioproject.xml
-rw-rw-r--  1 ec2-user ec2-user     74635 Aug  9  2000 grail
drwxr-xr-x 10 ec2-user ec2-user       161 Oct  2 03:28 hadoop-2.6.4
-rw-rw-r--  1 ec2-user ec2-user 197153999 May  3 18:35 hadoop-2.6.4.tar.gz
-rw-rw-r--  1 ec2-user ec2-user       148 Oct  1 22:15 lucky.py
-rw-rw-r--  1 ec2-user ec2-user       299 Oct  1 22:17 lucky.txt
-rw-rw-r--  1 ec2-user ec2-user        69 Oct  1 19:15 mycopy.txt
-rw-rw-r--  1 ec2-user ec2-user       102 Oct  1 21:50 myfile.txt
-rw-rw-r--  1 ec2-user ec2-user       417 Oct  1 20:28 myzipfile.zip
-rw-rw-r--  1 ec2-user ec2-user        33 Oct  1 21:05 redirect1.txt
-rw-rw-r--  1 ec2-user ec2-user       368 Oct  1 21:06 redirect2.txt
-rw-rw-r--  1 ec2-user ec2-user        68 Oct  1 22:20 was.py
-rw-rw-r--  1 ec2-user ec2-user      1884 Oct  2 01:00 wordCount.py
[ec2-user@ip-172-31-21-33 ~]$ hadoop fs -put bioproject.xml /data
[ec2-user@ip-172-31-21-33 ~]$ hadoop fs -ls /data
Found 1 items
-rw-r--r--   3 ec2-user supergroup  231149003 2021-10-02 03:42 /data/bioproject.xml
[ec2-user@ip-172-31-21-33 ~]$
```

Run word count on the downloaded text file, using the time command to determine the total runtime of the MapReduce job.   You can use the following (single-line!) command. This invokes the wordcount example built into the example jar file, supplying /data/bioproject.xml as the input and /data/wordcount1 as the output directory. Please remember this is <u>one command</u>, if you do not paste it as a single line, it will not work.

**time hadoop jar hadoop-2.6.4/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.6.4.jar  wordcount /data/bioproject.xml /data/wordcount1**

Report the time that the job took to execute as screenshot



```
21/10/02 03:54:34 INFO mapreduce.Job:  map 100% reduce 100%
21/10/02 03:54:34 INFO mapreduce.Job: Job job_1633145348467_0001 completed successfully
21/10/02 03:54:35 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=59605201
                FILE: Number of bytes written=86827943
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=231153301
                HDFS: Number of bytes written=20056175
                HDFS: Number of read operations=9
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=94173
                Total time spent by all reduces in occupied slots (ms)=12295
                Total time spent by all map tasks (ms)=94173
                Total time spent by all reduce tasks (ms)=12295
                Total vcore-milliseconds taken by all map tasks=94173
                Total vcore-milliseconds taken by all reduce tasks=12295
                Total megabyte-milliseconds taken by all map tasks=96433152
                Total megabyte-milliseconds taken by all reduce tasks=12590080
        Map-Reduce Framework
                Map input records=5284546
                Map output records=18562366
                Map output bytes=279356680
                Map output materialized bytes=26902454
                Input split bytes=202
                Combine input records=20053191
                Combine output records=2673165
                Reduce input groups=1040390
                Reduce shuffle bytes=26902454
                Reduce input records=1182340
                Reduce output records=1040390
                Spilled Records=3855505
                Shuffled Maps =2
                Failed Shuffles=0
                Merged Map outputs=2
                GC time elapsed (ms)=1052
                CPU time spent (ms)=41810
                Physical memory (bytes) snapshot=566173696
                Virtual memory (bytes) snapshot=6320132096
                Total committed heap usage (bytes)=334364672
        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=231153099
        File Output Format Counters
                Bytes Written=20056175

real    1m12.495s
user    0m3.885s
sys     0m0.268s
[ec2-user@ip-172-31-21-33 ~]$
```

(this reports the size of a particular file or directory in HDFS. The output file will be named part-r-00000)
**hadoop fs -du /data/wordcount1/**

(Just like in Linux, the cat HDFS command will dump the output of the entire file and grep command will filter the output to all lines that matches this particular word). To determine the count of occurrences of "arctic", run the following command:

**hadoop fs -cat /data/wordcount1/part-r-00000 | grep arctic**

It outputs the entire content of part-r-00000 file and then uses pipe | operator to filter it through grep (filter) command. If you remove the pipe and grep, you will get the entire word count content dumped to screen, similar to cat command.

Congratulations, you just finished running wordcount using Hadoop.

## Part 4: Hive Intro

4) In this section we are going to use Hive to run a few queries over the Hadoop framework. These instructions assume that you are starting from a working Hadoop installation. If you are starting your instance, you need to start Hadoop as well.
   Hive commands are listed in **Calibri bold font**

   a) Download and install Hive:
      **cd**
      (this command is there to make sure you start from home directory, on the same level as where hadoop is located)

**wget http:// dbgroup.cdm.depaul.edu/Courses/CSC555/apache-hive-2.0.1-bin.tar.gz**
**gunzip apache-hive-2.0.1-bin.tar.gz**
**tar xvf apache-hive-2.0.1-bin.tar**

set the environment variables (can be automated by adding these lines in ~/.bashrc). If you don't, you will have to set these variables every time you use Hive.
**export HIVE_HOME=/home/ec2-user/apache-hive-2.0.1-bin**
**export PATH=$HIVE_HOME/bin:$PATH**

**$HADOOP_HOME/bin/hadoop fs -mkdir /tmp**
**$HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse**
(if you get an error here, it means that /user/hive does not exist yet. Fix that by running
**$HADOOP_HOME/bin/hadoop fs -mkdir -p /user/hive/warehouse instead)**

**$HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp**
**$HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse**

We are going to use Vehicle data (originally from http://www.fueleconomy.gov/feg/download.shtml)

You can get the already unzipped, comma-separated file from here:
**wget http://dbgroup.cdm.depaul.edu/Courses/CSC555/vehicles.csv**

You can take a look at the data file by either
**nano vehicles.csv** or
**more vehicles.csv** (you can press space to scroll and q or Ctrl-C to break out)

Note that the first row in the data is the list of column names. What follows after commands that start Hive, is the table that you will create in Hive loading the first 5 columns. Hive is not particularly sensitive about invalid or partial data, hence if we only define the first 5 columns, it will simply load the first 5 columns and ignore the rest. You can see the description of all the columns here (atvtype was added later) http://www.fueleconomy.gov/feg/ws/index.shtml#vehicle

Create the ec2-user directory on the HDFS side (absolute path commands should work anywhere and not just in Hadoop directory as bin/hadoop does). Here, we are creating the user "home" directory on the HDFS side.

**hadoop fs -mkdir /user/ec2-user/**

Run hive (from the hive directory because of the first command below):
**cd $HIVE_HOME**
**$HIVE_HOME/bin/schematool -initSchema -dbType derby**

(NOTE: This command initializes the database metastore. If you need to restart/reformat or see errors related to meta store, delete the metastore using **rm -rf metastore_db/** and then repeat the above initSchema command)

**bin/hive**

You can now create a table by pasting this into the Hive terminal:

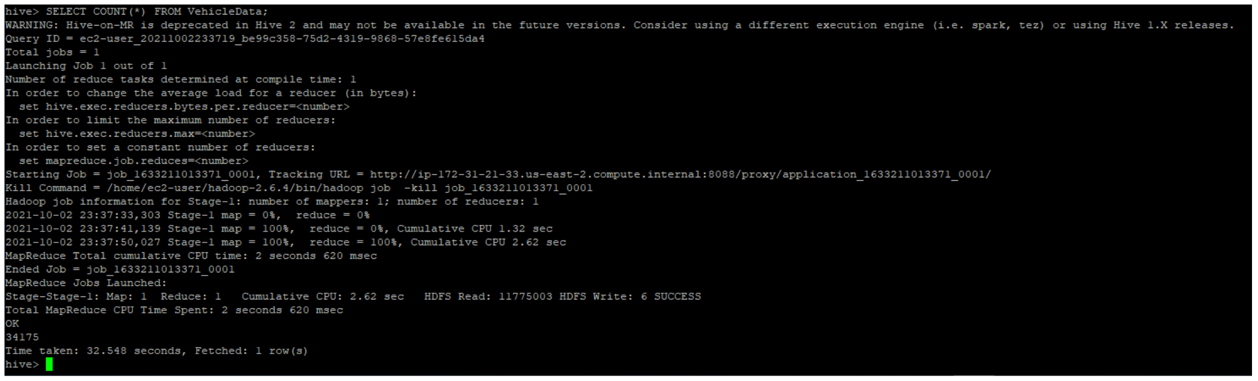
```
CREATE TABLE VehicleData (
barrels08 FLOAT, barrelsA08 FLOAT,
charge120 FLOAT, charge240 FLOAT,
city08 FLOAT)
ROW FORMAT DELIMITED FIELDS
TERMINATED BY ',' STORED AS TEXTFILE;
```

You can load the data (from the local file system, not HDFS) using:

```
LOAD DATA LOCAL INPATH '/home/ec2-user/vehicles.csv'
OVERWRITE INTO TABLE VehicleData;
```

(NOTE: If you downloaded vehicles.csv file into the hive directory, you have to change file name to /home/ec2-user/apache-hive-2.0.1-bin/vehicles.csv instead)

Verify that your table had successfully loaded by running
**SELECT COUNT(*) FROM VehicleData;**
(Copy the query output and report how many rows you got as an answer.)

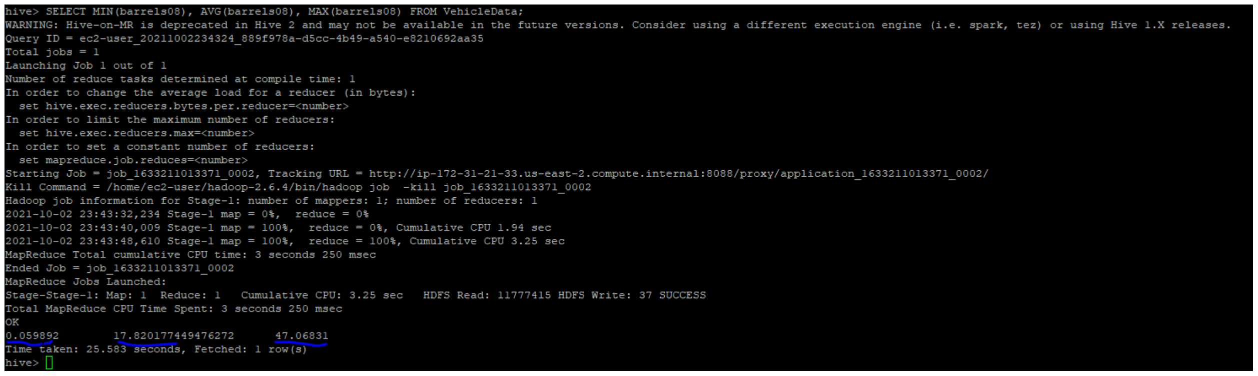


```
hive> SELECT COUNT(*) FROM VehicleData;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ec2-user_20211002233719_be99c358-75d2-4319-9868-57e8fe615da4
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1633211013371_0001, Tracking URL = http://ip-172-31-21-33.us-east-2.compute.internal:8088/proxy/application_1633211013371_0001/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1633211013371_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-10-02 23:37:33,303 Stage-1 map = 0%,  reduce = 0%
2021-10-02 23:37:41,139 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.32 sec
2021-10-02 23:37:50,027 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.62 sec
MapReduce Total cumulative CPU time: 2 seconds 620 msec
Ended Job = job_1633211013371_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.62 sec   HDFS Read: 11775003 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 620 msec
OK
34175
Time taken: 32.548 seconds, Fetched: 1 row(s)
hive>
```

There are 34,175 rows.

Run a couple of HiveQL queries to verify that everything is working properly:

**SELECT MIN(barrels08), AVG(barrels08), MAX(barrels08) FROM VehicleData;**
(copy the output from that query)



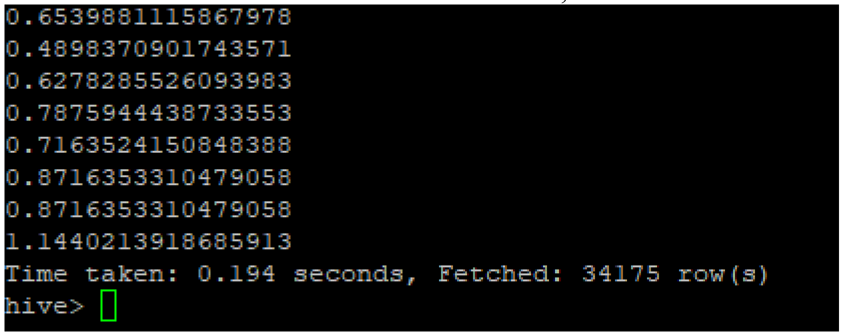
```
hive> SELECT MIN(barrels08), AVG(barrels08), MAX(barrels08) FROM VehicleData;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
Query ID = ec2-user_20211002234324_889f978a-d5cc-4b49-a540-e8210692aa35
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1633211013371_0002, Tracking URL = http://ip-172-31-21-33.us-east-2.compute.internal:8088/proxy/application_1633211013371_0002/
Kill Command = /home/ec2-user/hadoop-2.6.4/bin/hadoop job  -kill job_1633211013371_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-10-02 23:43:32,234 Stage-1 map = 0%,  reduce = 0%
2021-10-02 23:43:40,009 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.94 sec
2021-10-02 23:43:48,610 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.25 sec
MapReduce Total cumulative CPU time: 3 seconds 250 msec
Ended Job = job_1633211013371_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.25 sec   HDFS Read: 11777415 HDFS Write: 37 SUCCESS
Total MapReduce CPU Time Spent: 3 seconds 250 msec
OK
0.059892        17.820177449476272      47.06831
Time taken: 25.583 seconds, Fetched: 1 row(s)
hive> 
```

**As shown above:**

- **The Minimum of barrels08 is 0.06**
- **The Average of barrels08 is 17.82**
- **The Maximum of barrels08 is 47.07**

**SELECT (barrels08/city08) FROM VehicleData;**
(you do not need to report the output from that query, but report "Time taken")

The time taken is 0.194 seconds to fetch 34,175 rows as shown below:



```
0.6539881115867978
0.4898370901743571
0.6278285526093983
0.7875944438733553
0.7163524150848388
0.8716353310479058
0.8716353310479058
1.1440213918685913
Time taken: 0.194 seconds, Fetched: 34175 row(s)
hive> 
```

Next, we are going to output three of the columns into a separate file (as a way to transform data for further manipulation that you may be interested in)

**INSERT OVERWRITE DIRECTORY 'ThreeColExtract'**
**SELECT barrels08, city08, charge120**
**FROM VehicleData;**

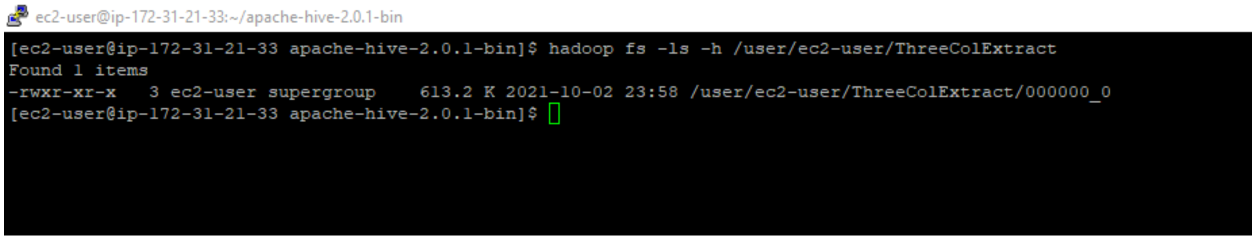
You can now exit Hive by running **exit;**

And verify that the new output file has been created (the file will be called 000000_0)

The file would be created in HDFS in user home directory (/user/ec2-user/ThreeColExtract)

Report the size of the newly created file and include the screenshot.

The size is 613.2 K as shown below:

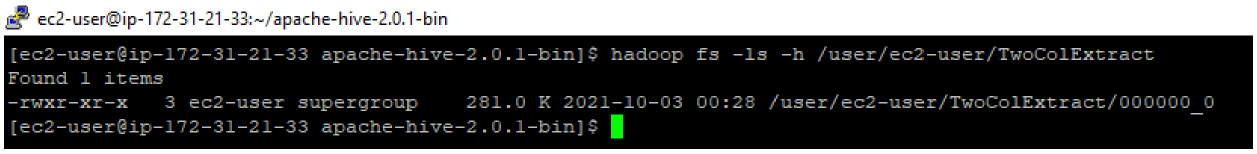


```
[ec2-user@ip-172-31-21-33 apache-hive-2.0.1-bin]$ hadoop fs -ls -h /user/ec2-user/ThreeColExtract
Found 1 items
-rwxr-xr-x   3 ec2-user supergroup    613.2 K 2021-10-02 23:58 /user/ec2-user/ThreeColExtract/000000_0
[ec2-user@ip-172-31-21-33 apache-hive-2.0.1-bin]$
```

Next, you should go back to the Hive terminal, create a new table that is going to load 8 columns instead of 5 in our example (i.e. create and load a new table that defines 8 columns by including columns city08U,cityA08,cityA08U) and use Hive to generate a new output file containing only the city08U and cityA08U columns from the vehicles.csv file. Report the size of that output file as well.

**If the size is zero, you are looking at a directory and the output file(s) are in that directory.**

The size of the file is 281.0K as shown below:



```
[ec2-user@ip-172-31-21-33 apache-hive-2.0.1-bin]$ hadoop fs -ls -h /user/ec2-user/TwoColExtract
Found 1 items
-rwxr-xr-x   3 ec2-user supergroup    281.0 K 2021-10-03 00:28 /user/ec2-user/TwoColExtract/000000_0
[ec2-user@ip-172-31-21-33 apache-hive-2.0.1-bin]$
```

Submit a single document containing your written answers. Be sure that this document contains your name and “CSC 555 Assignment 2” at the top.