## HU Extension School E-63 Big Data Analytics

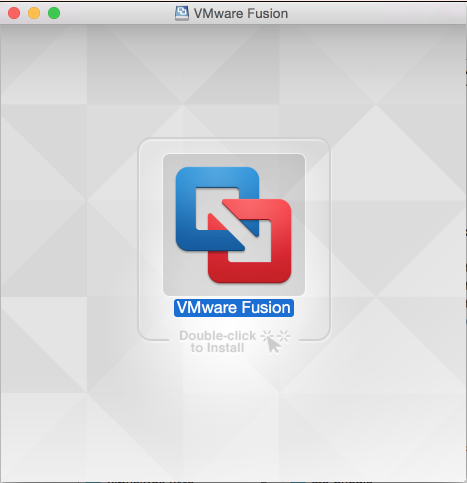
## Assignment 02

### Handed out: 02/06/2016 Due by 11:30 PM on Friday, 02/12/2016

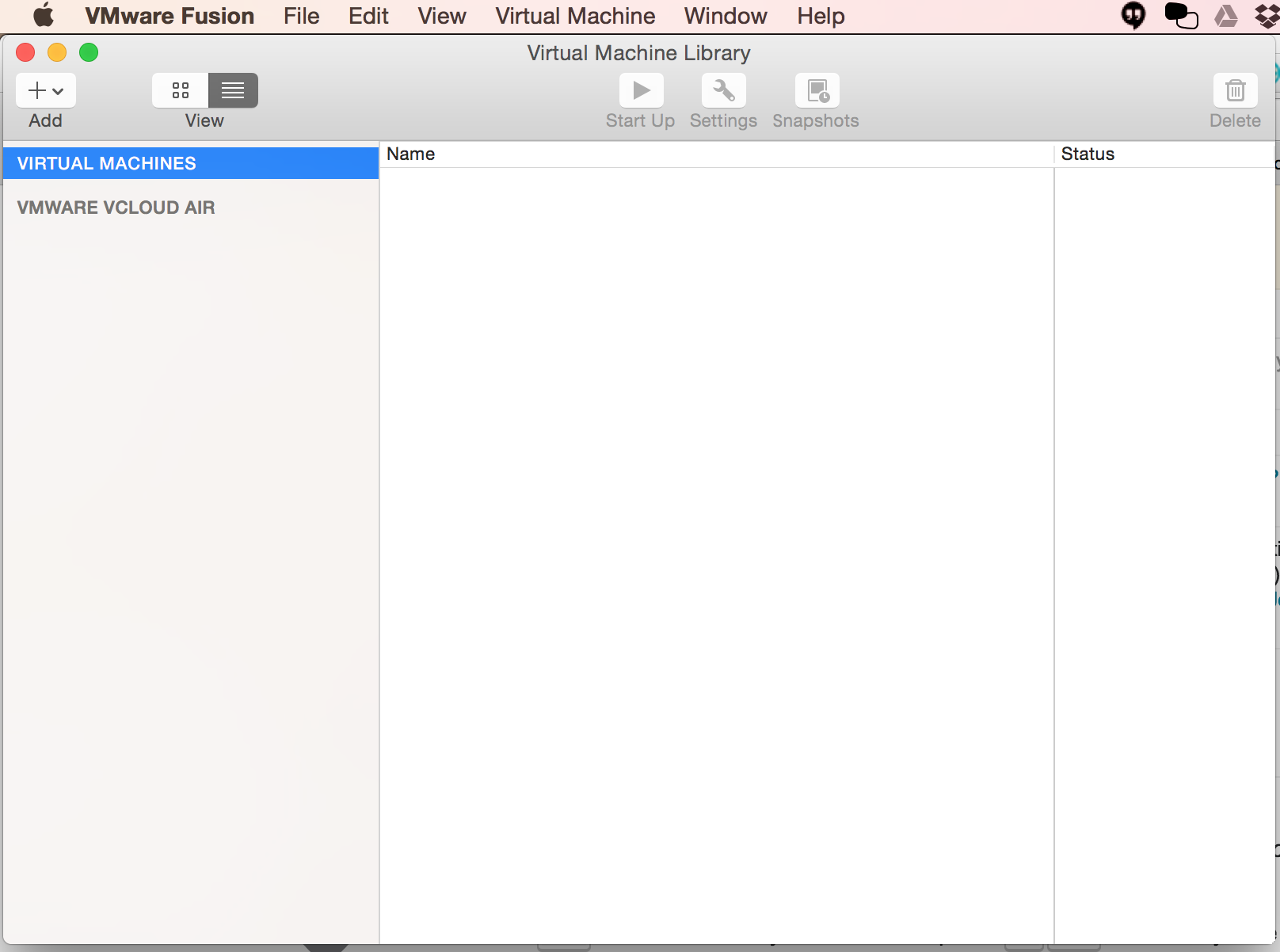
Capture all steps of your implementation with comments indicating what are accomplishing with every step. Place those in this MS Word document bellow the problem statement. Please send comments and questions to the Discussion Forum on the class site.

### 

**Problem 1)** Please, download and install VMware Workstation 11 on your 64 bit Windows PC or VMWare Fusion 7, if you are on a MAC.

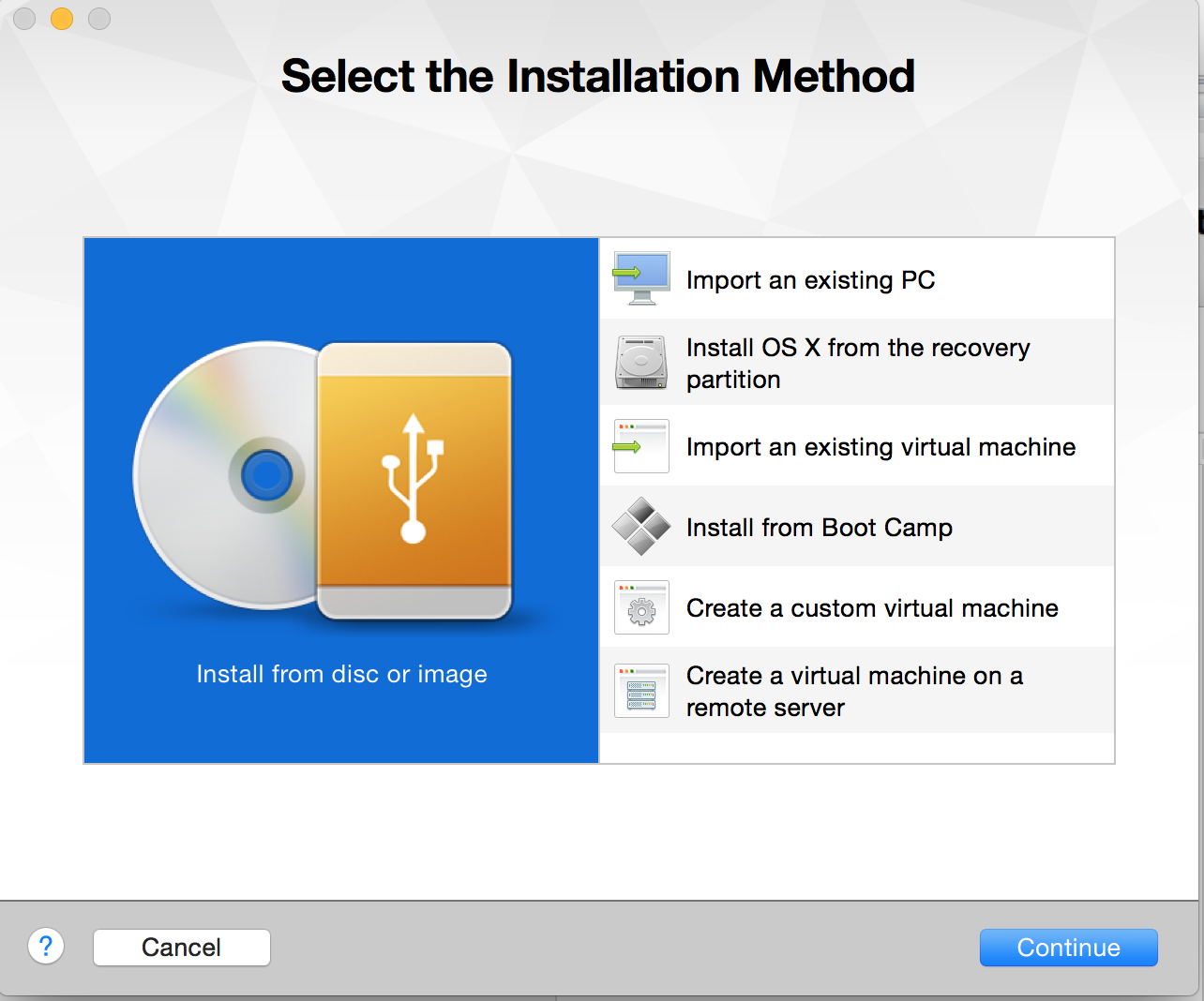


Description: Began installation of VMWare Fusion 8 on my Mac.

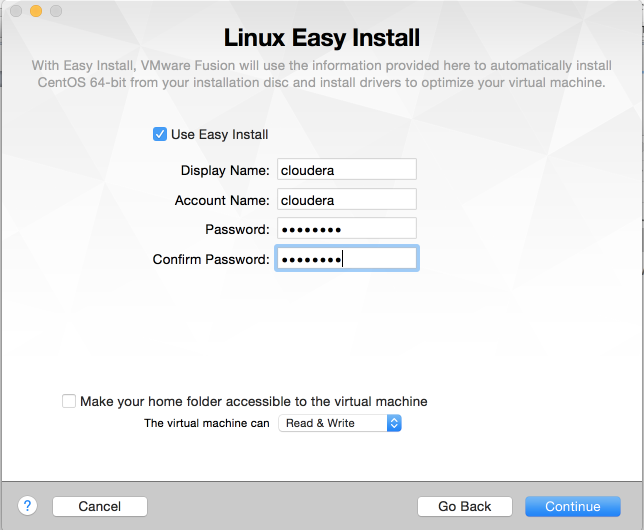


Description: I have downloaded and installed VMWare Fusion on my Mac.

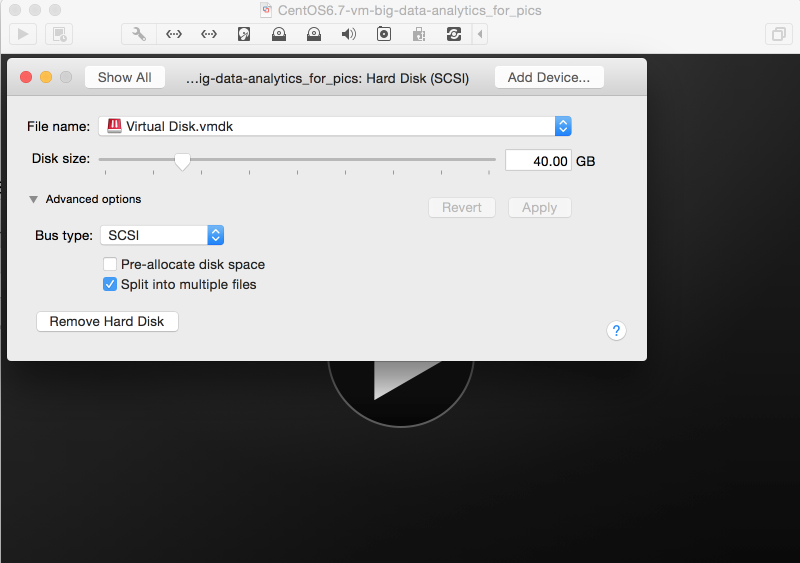
Please download 64 bit CentOS6.7 and create a 64 bit VM. If you know what you are doing and you work with another flavor of Linux supported by CDH5.5.1, please be free to create a virtual machine based on your favorite Linux flavor.



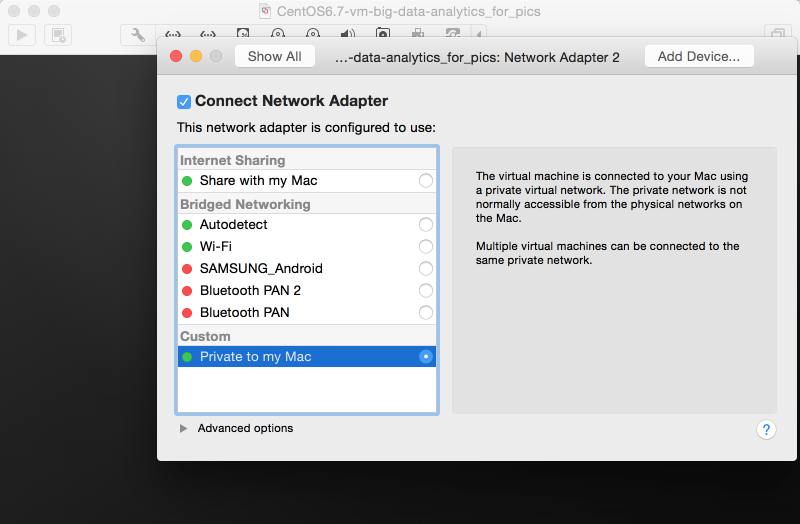
Description: Started creation of virtual machine from CentOS6.7 iso image



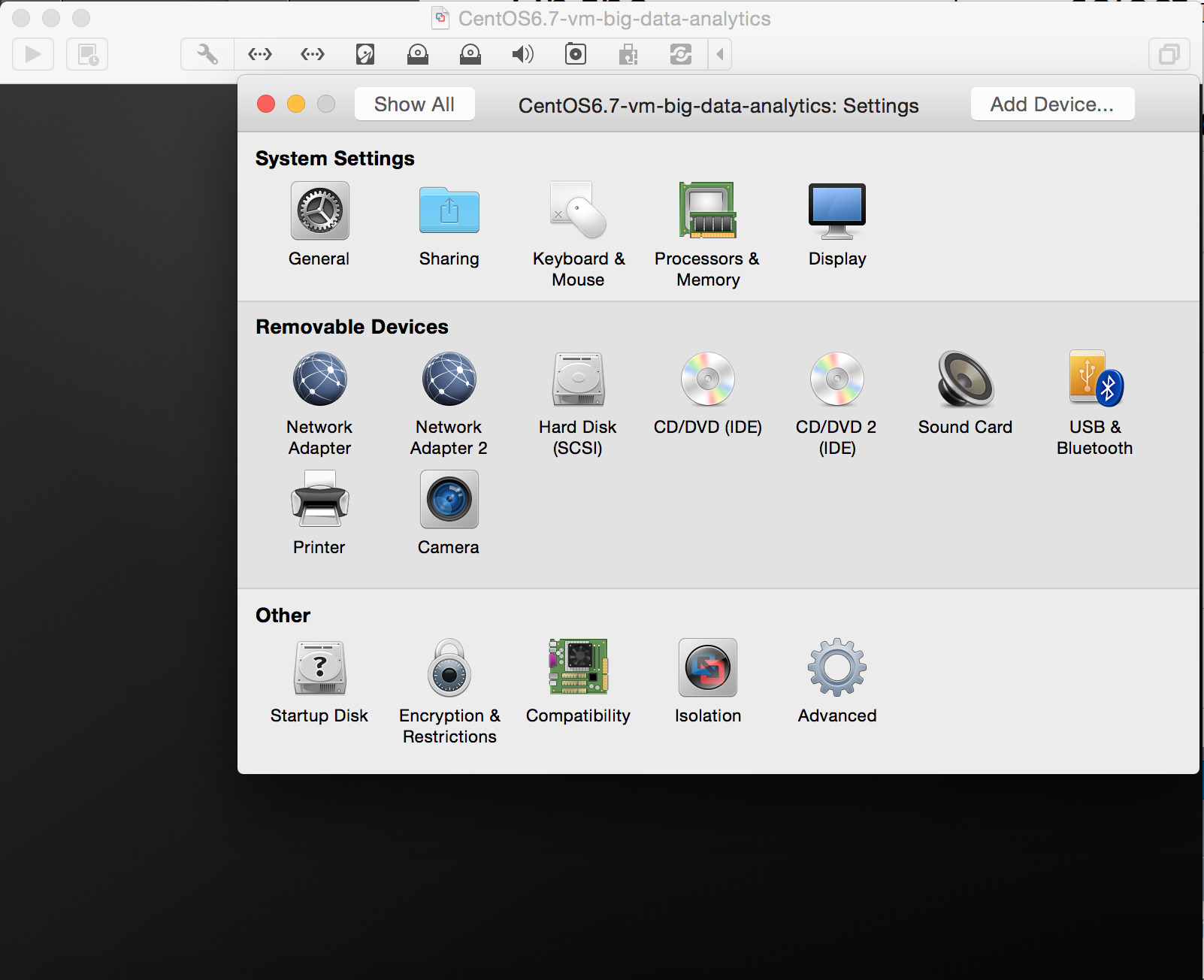
Description: After finishing the installation of CentOS VM, I created a user called cloudera



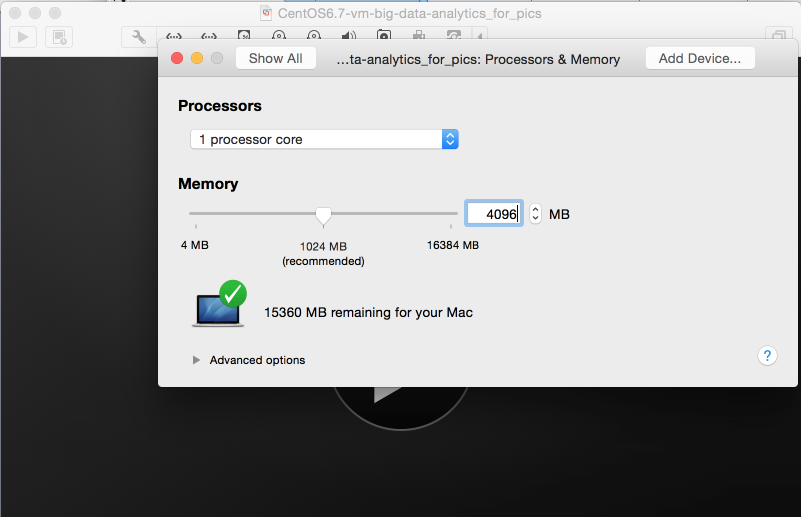
Description: Allocated 40GB disk size for the virtual machine.



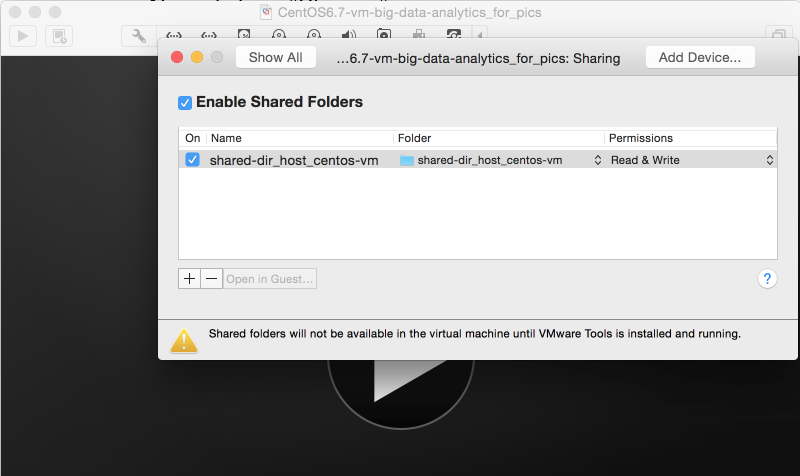
Description: Created another network adapter and configured it to use a private virtual network which will exist between my Mac and the VM



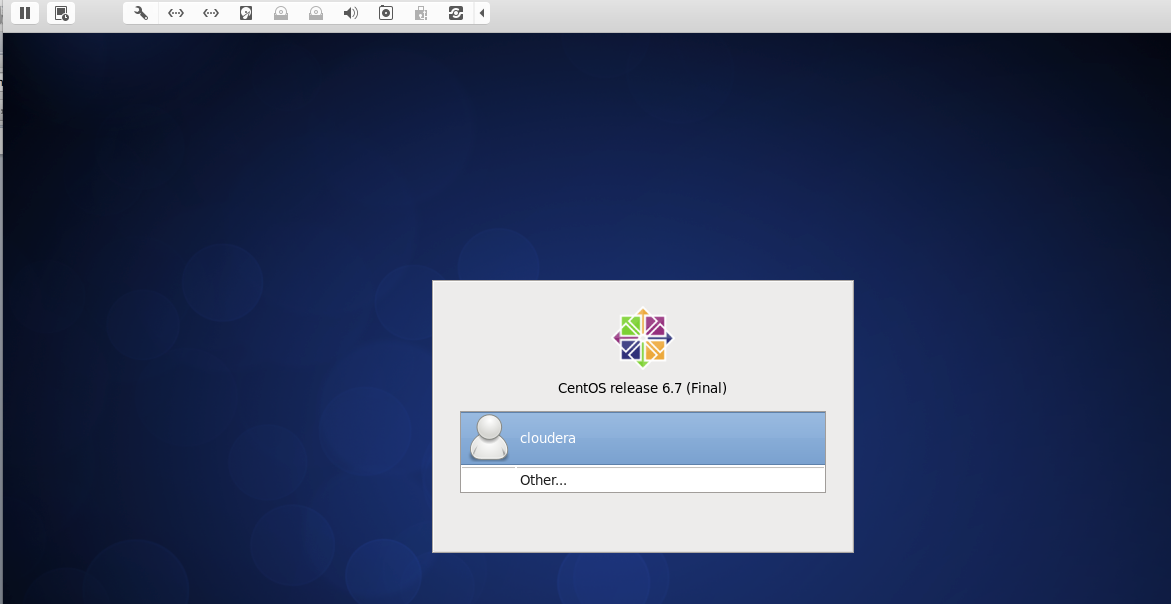
Description: So, now there are 2 network adapters for this VM



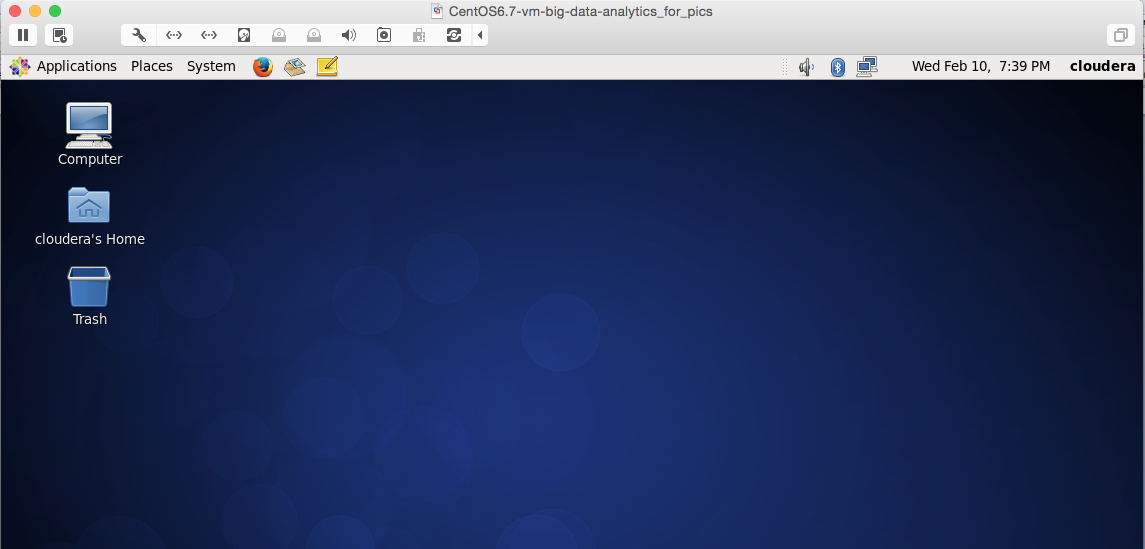
Description: Allocated 4GB RAM to the VM



Description: Added shared directory (this shared directory will be useful for file transfer between my Mac (host OS) and VM (guest OS)



Description: Finally I get to the login screen of VM

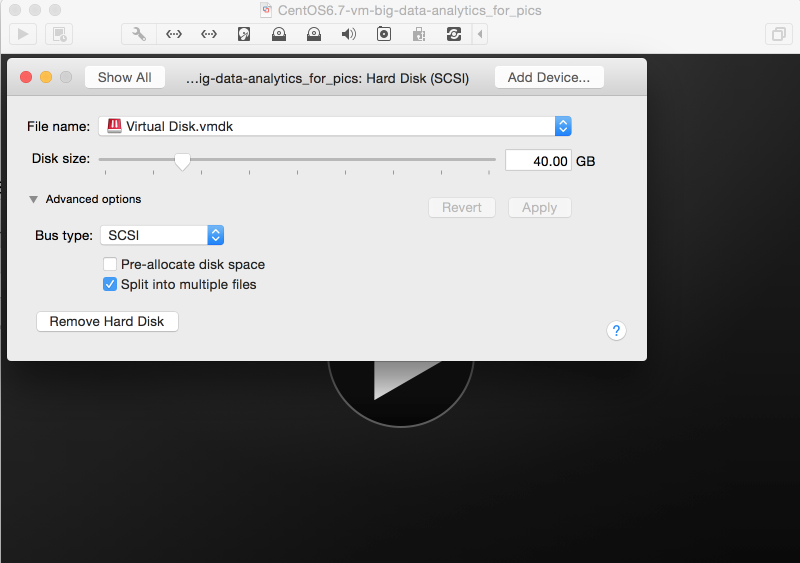


Description: This is the desktop of just created CentOS6.7 VM



Description: In the above pic, you can see shared directory accessible on /mnt/hgfs/

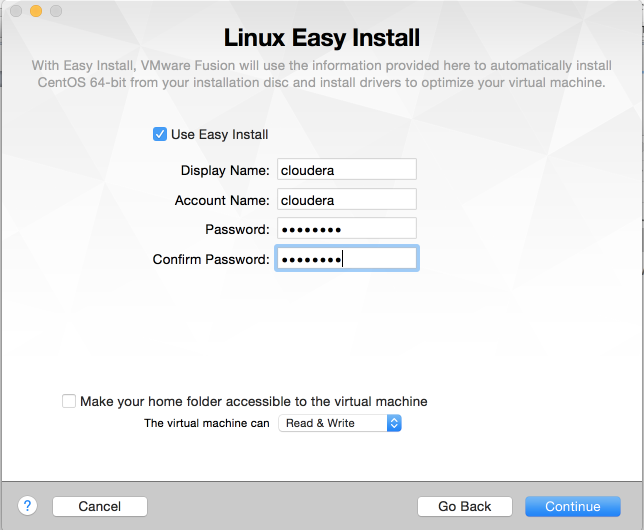
Provide your virtual machine with some 40GB of disk space, if you can spare it. For whatever reasons, Hadoop installation appears to prefer to have more than 20 GB of available space.



Description: 40GB allocated to the VM

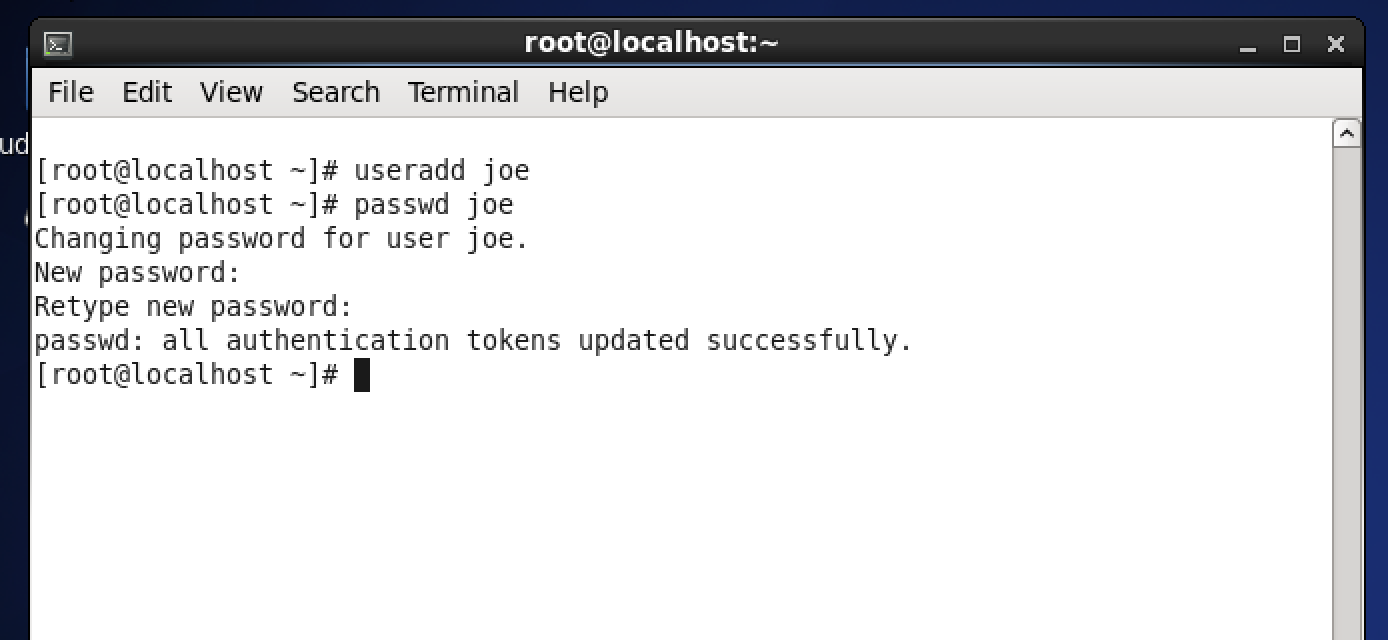
Name the main user of your VM cloudera.

Do not use name hadoop. “hadoop” is a bad name for a user, since Hadoop framework has an executable called hadoop and it creates many directories with that same name and those would not necessarily be owned by the VM user called hadoop.



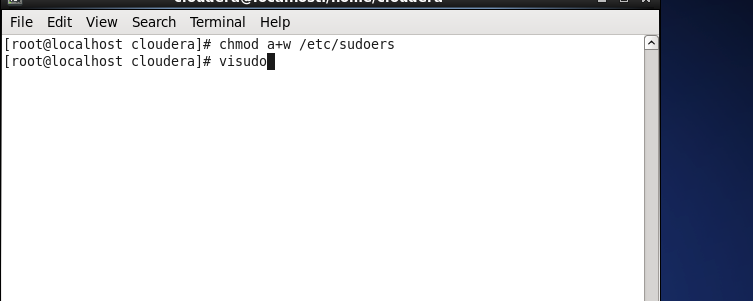
Description: During installation process, primary user *cloudera* was created

On that VM create yet another user called joe.

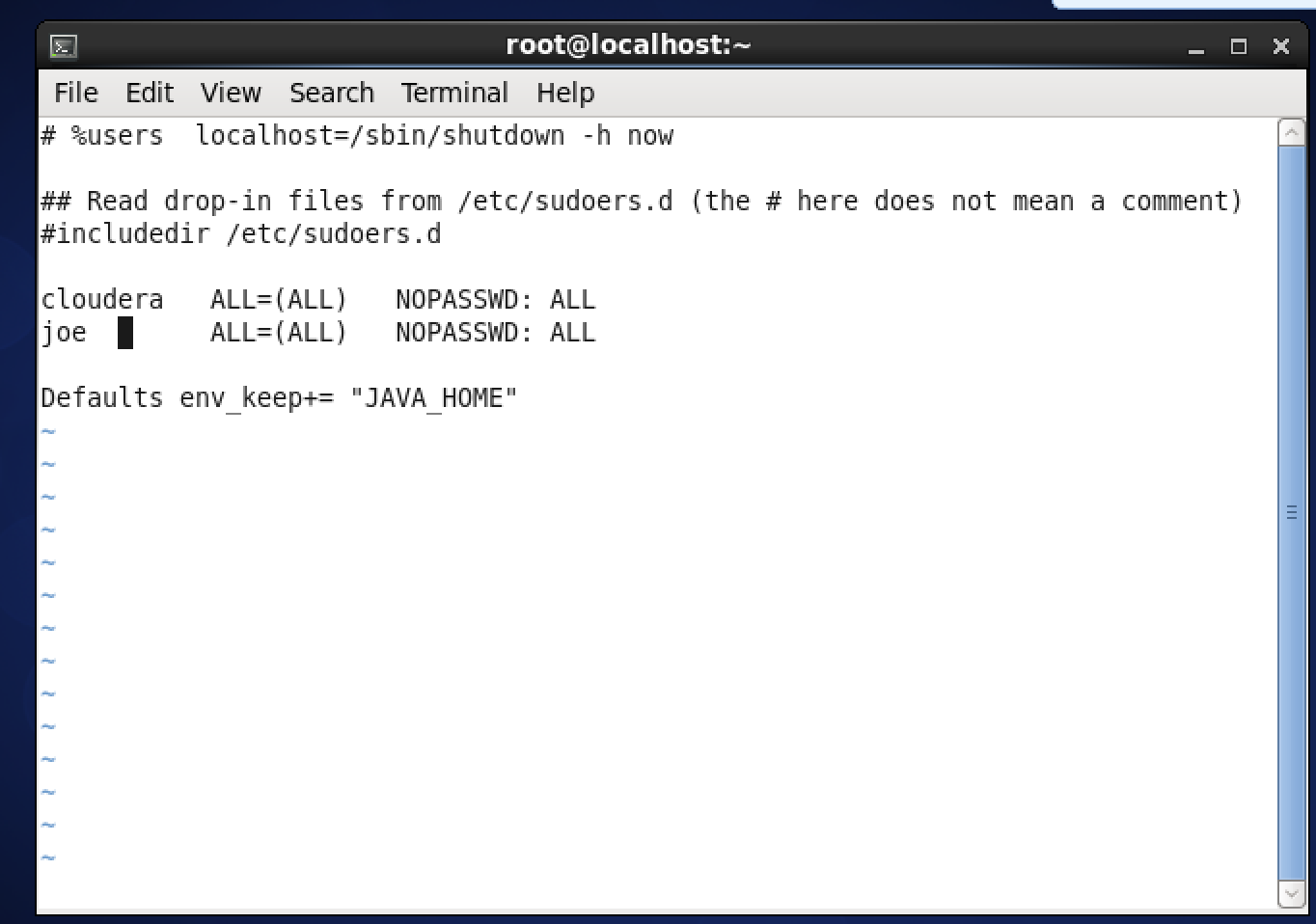


Description: Created a user called joe and assigned it a password.

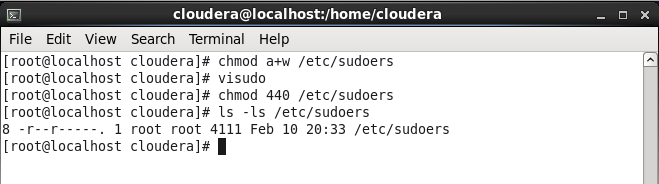
Make both users sudo users.



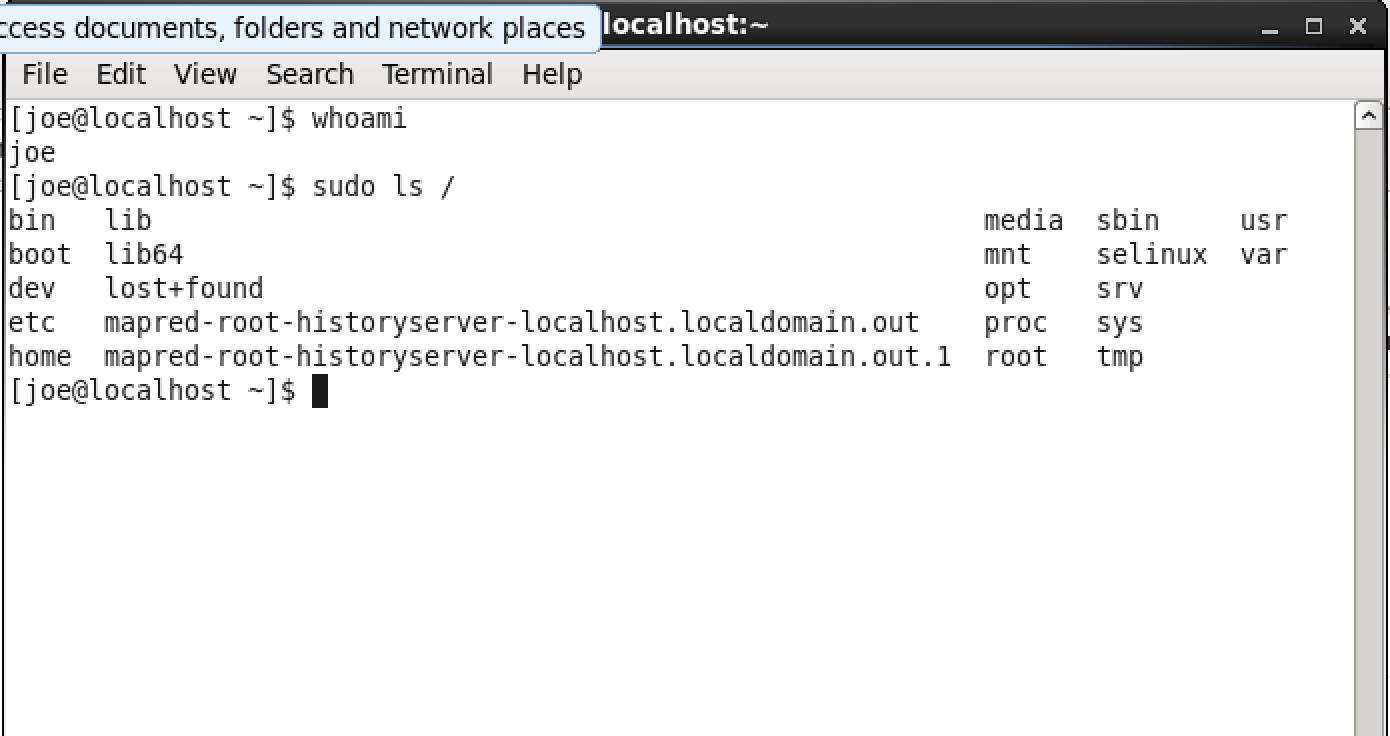
Description: I made the /etc/sudoers file writable. And then I started visudo program to edit the sudoers file.



Description: Added entries to /etc/sudoers file so that cloudera and joe users can use their sudo privileges without being asked for password.



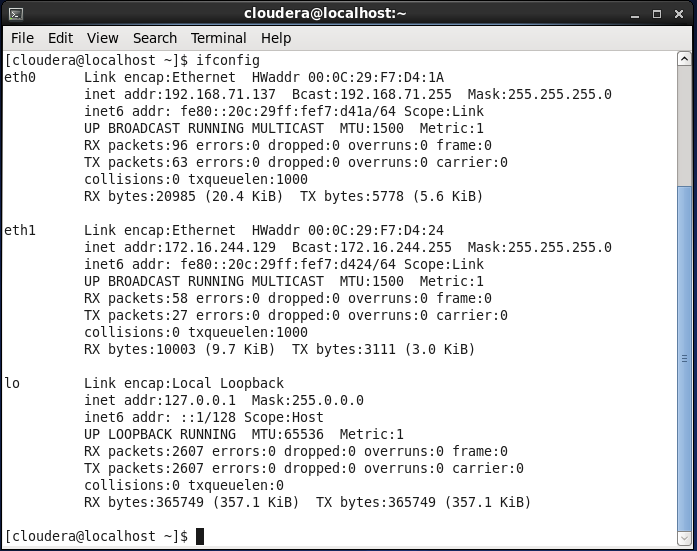
Description: Changed back the permissions of /etc/sudoers so that the file is read only.



Description: Above commands indicate that user joe has sudo privileges and it is not prompted for password while exercising it’s sudo privileges.



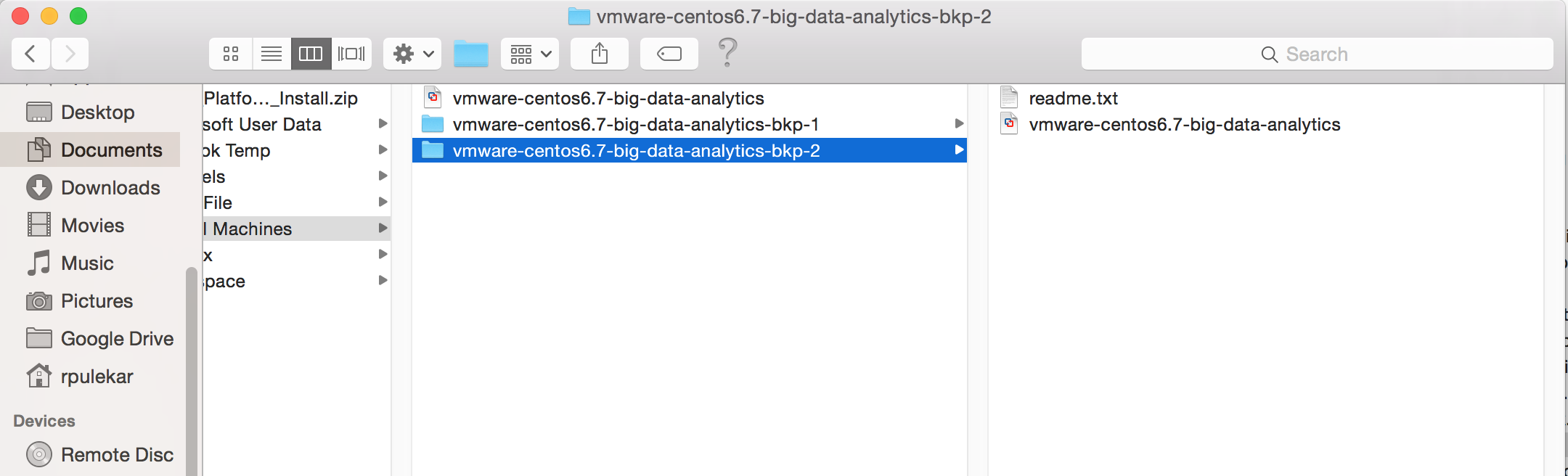
Description: Above commands indicate that user cloudera has sudo privileges and it is not prompted for password while exercising its sudo privileges.



Description: This is output of ifconfig command. It shows 2 network adapters with their own IP addresses.

Once your CentOS is fully installed, please shut the VM down and make a copy of the entire directory containing that VM. Name the folder containing that copy differently.

Two VMs are identical and you could even run them simultaneously if your machine has enough memory. In the folder of each VM add a text file describing OS on your VM, usernames and passwords of important users. This little file will make your VMs useful long into the future. The reason you are creating the backup VM is to save time, if you damage the one VM on which you are installing your software.

****

Description: I have created 2 backups of the VM files. Each backup directory contains a readme file which has OS name and username and passwords.

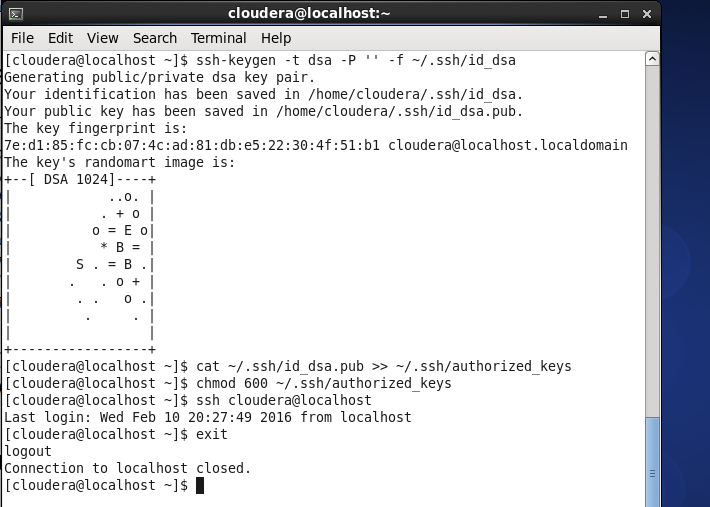
Please do not capture installation of Workstation 12 or Fusion. Please do not capture every step in creation of VM. Show addition of the second network adapter. Show steps in creation of user joe. Demonstrate that joe is a sudo user. Show results of your ifconfig command.

All this has been indicated in the screenshots I have taken above.

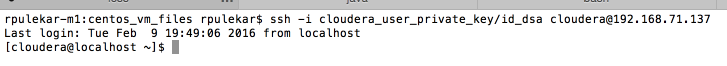
**Problem 2)** Use one of yourVMs and follow closely steps in the CDH5.5.1 Quick Start Guide, or my notes. PDF and PPT formats and characters on PC do not always map well into Unix (Linux) characters. If you want to copy commands from the guide you are better off doing it from the HTML version of the CDH Quick Start Guide, which you could find at:

[http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm\_qs\_quick\_start.html](http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cm_qs_quick_start.html%20) and open from with your VM.

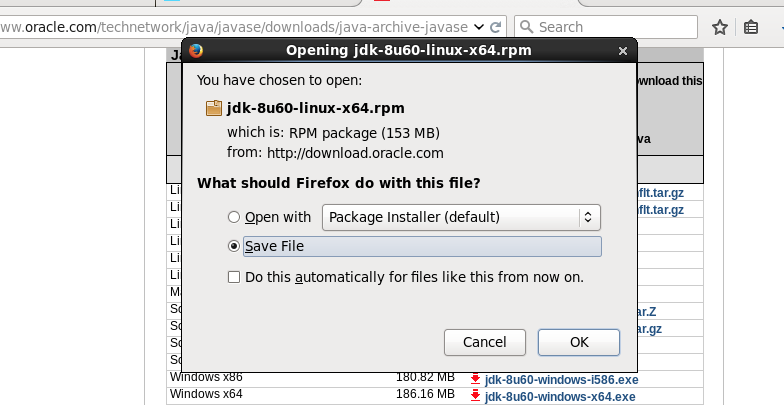
Both my notes and the Quick Start Guide will lead you through a “semi-automated” process of installing Hadoop. Please install YARN version of Hadoop. My notes add a few explanations beyond what you can see in the Cloudera’s guide. Read the notes and the guide very carefully. Do not execute commands for flavors of Linux other than RedHat (CentOS) unless you are working with another flavor purposefully. You will know that you have successfully installed Hadoop if all of tests described in the guide work properly.



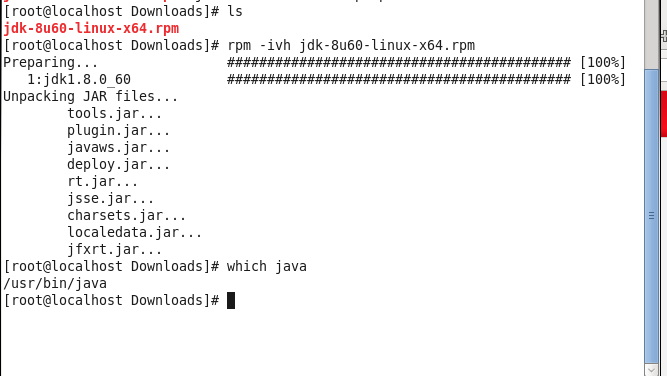
Description: Here I have created ssh key pair for cloudera user. I have added it to the public key to authorized\_keys file. What this achieves is that it is possible to ssh to cloudera user without entering password.



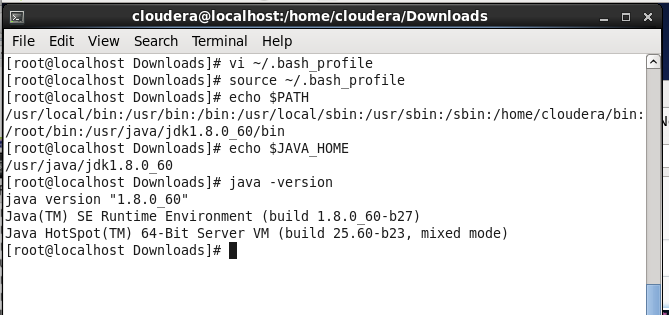
Description: The above shows that I can ssh into the VM from host machine using cloudera user’s private key. And as I am using private key of the cloudera user, I don’t need to enter password.



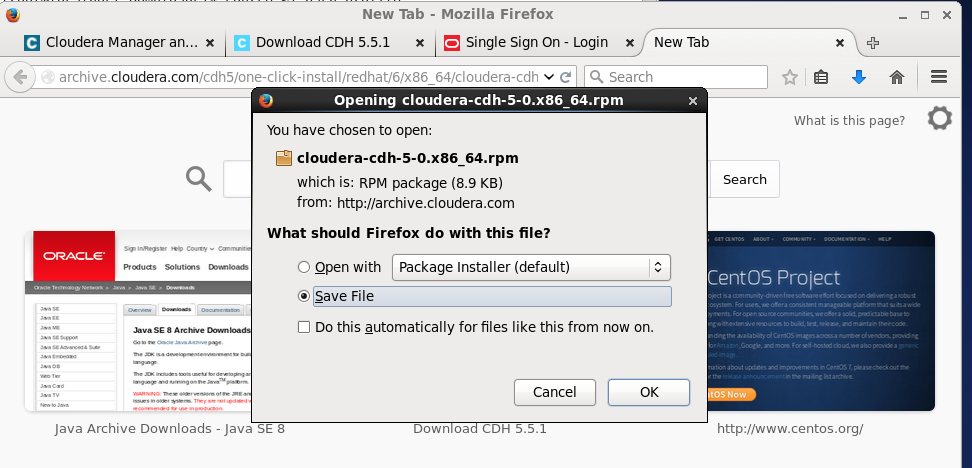
Description: Here I am downloading jdk 8 update 60 rpm.

****

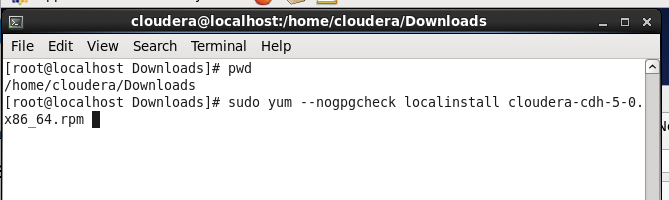
Description: I installed java 8 using the above downloaded rpm file.

****

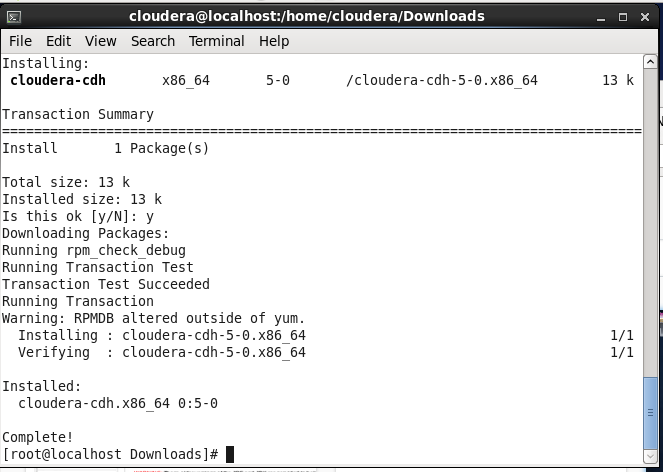
Description: The above indicates that I have java 8 installed and that JAVA\_HOME environment variable is set to point to the correct directory.

****

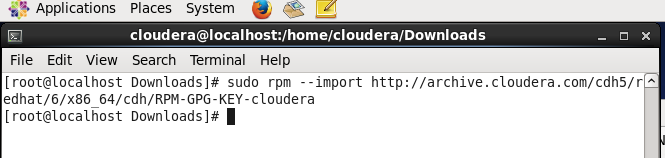
Description: Downloaded cloudera cdh rpm file.

****

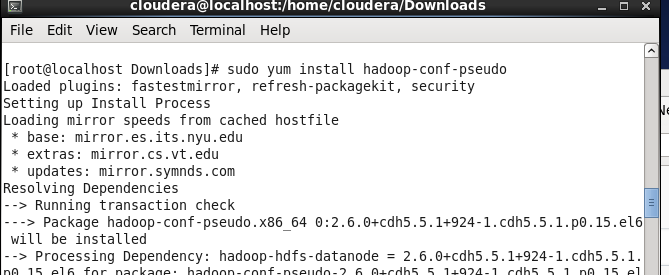
Description: Started installation of cloudera cdh rpm file. Using localinstall option to install a set of local rpm files. Using nogpgcheck to disable checking for signature of rpm creator.

****

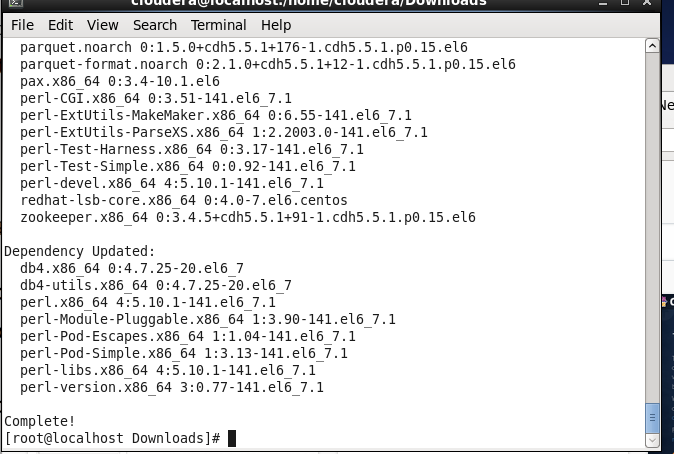
Description: Finished installation of cloudera cdh rpm.

****

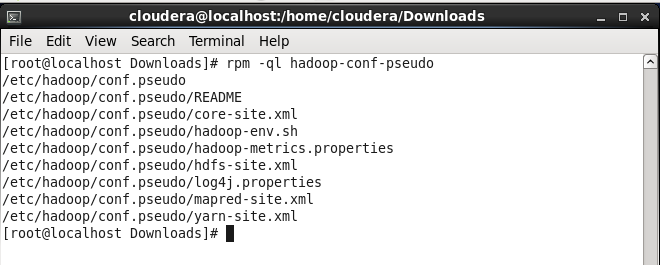
Description: Added cloudera public gpg key to be used for encrypting traffic between the nodes.

****

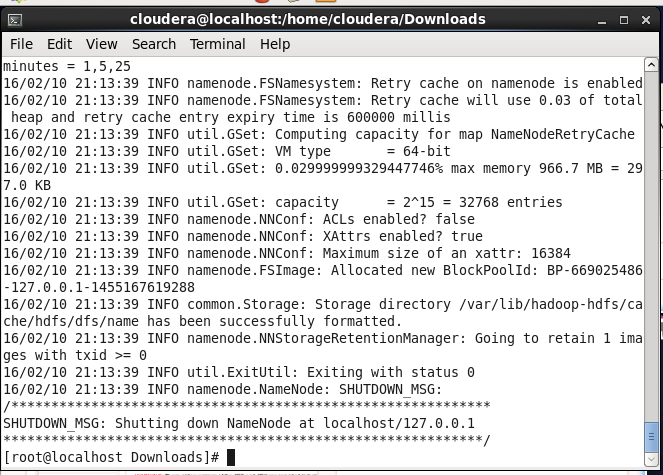
Description: Begin installation of yarn in pseudo-distributed mode

****

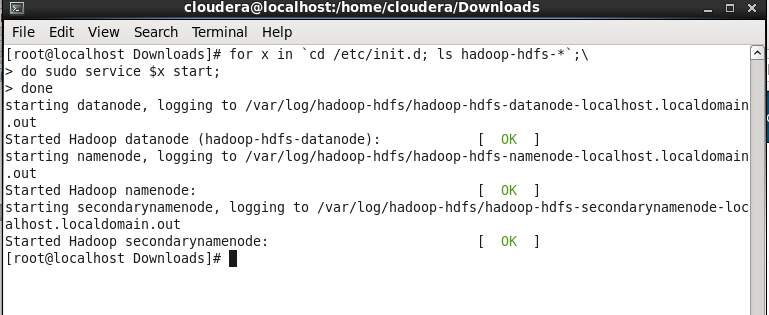
Description: Complete installation of yarn in pseudo-distributed mode

****

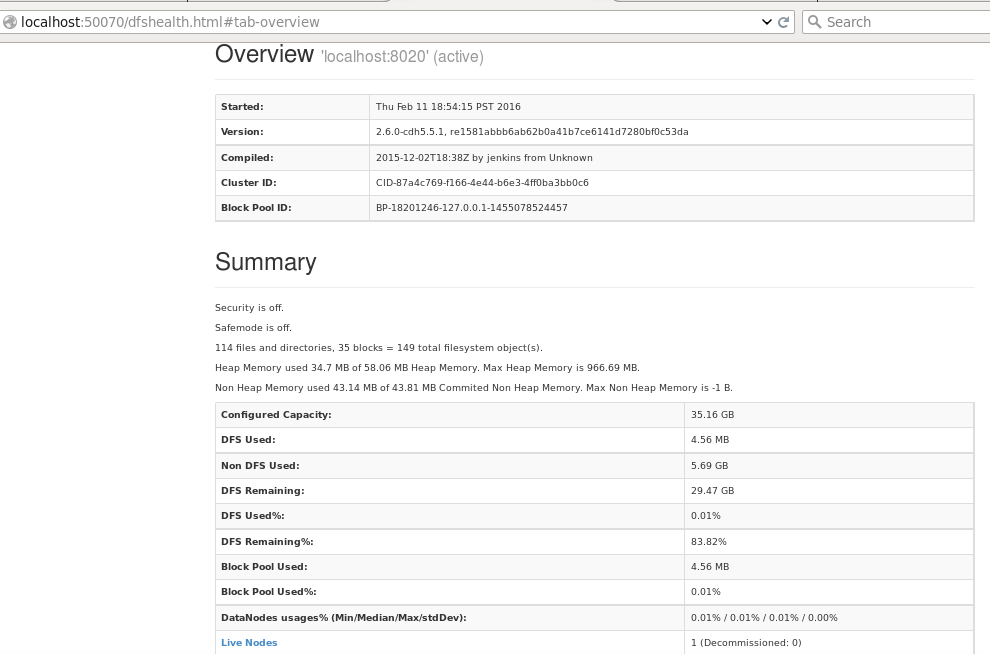
Description: Below shows configuration files of yarn

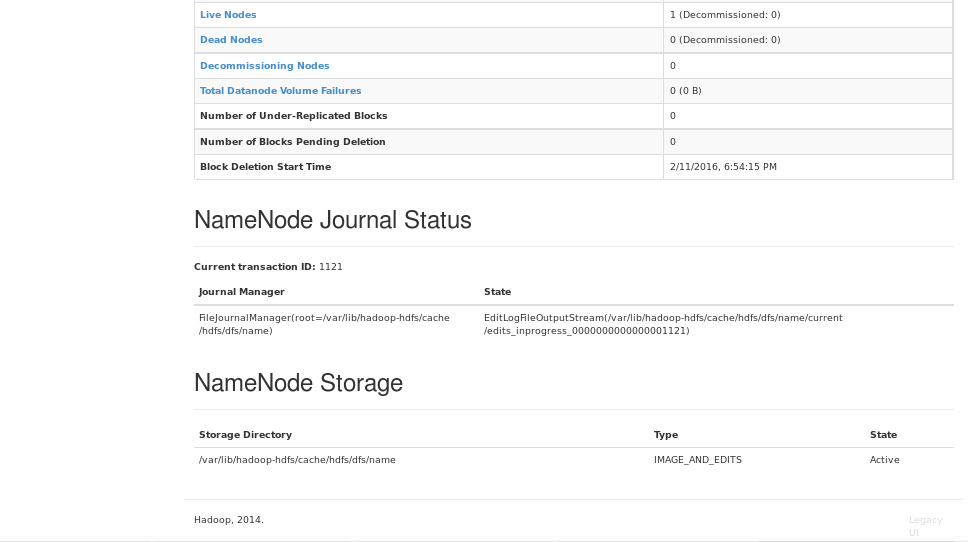
****

Description: Here, I have performed formatting of namenode. The above shows trailing end of the command output.

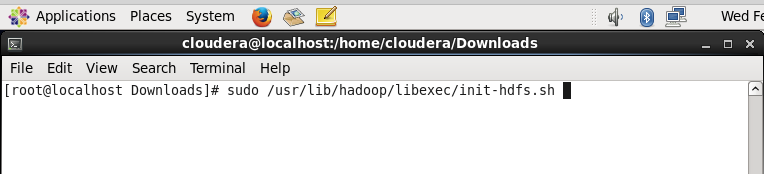
****

Description: The above commands starts 3 hadoop processes (hadoop datanode, hadoop namenode, hadoop secondarynamenode) as linux services.

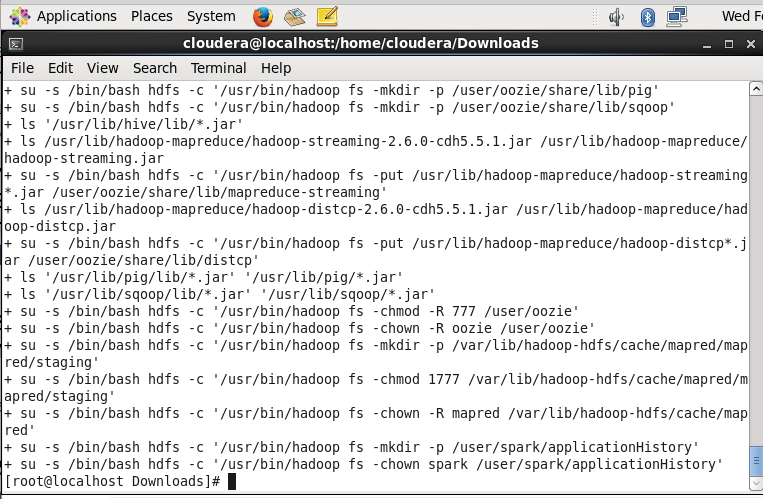
****

****

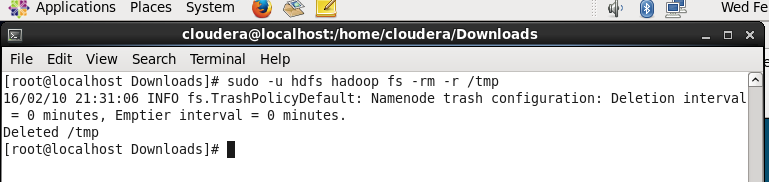
Description: The above status page shows status of namenode and status of services.

****

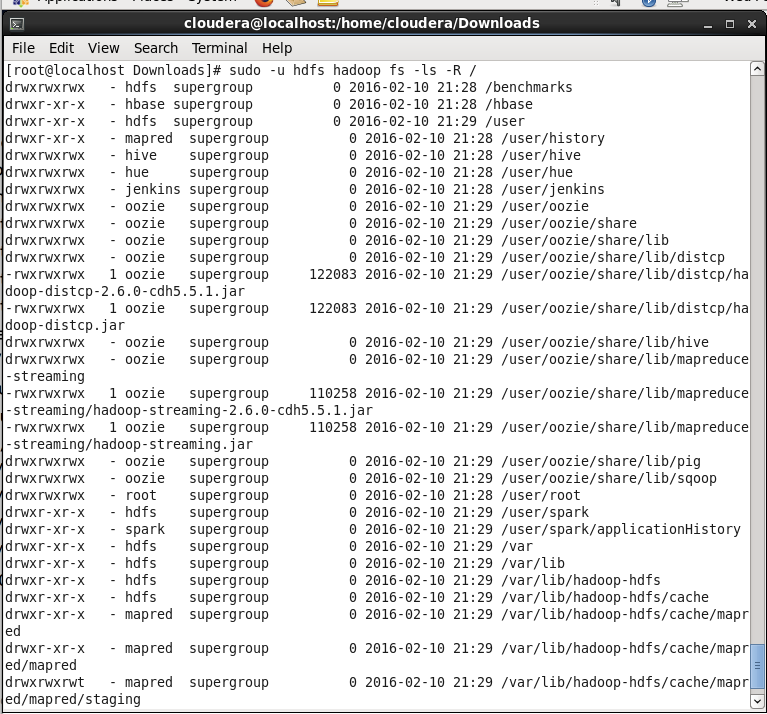
Description: Here I am about to run a script that changes the modes and ownerships of necessary files/folders.

****

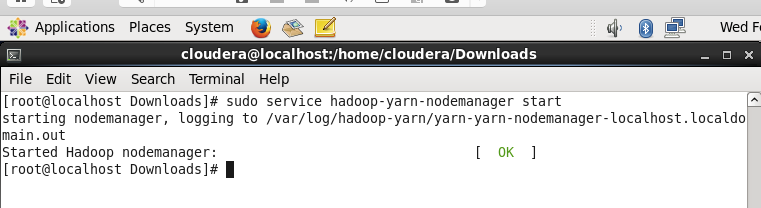
Description: Finished running the script that changes modes and ownerships of necessary files/folders.

****

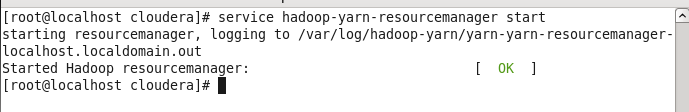
Description: Here I have recursively removed /tmp directory.

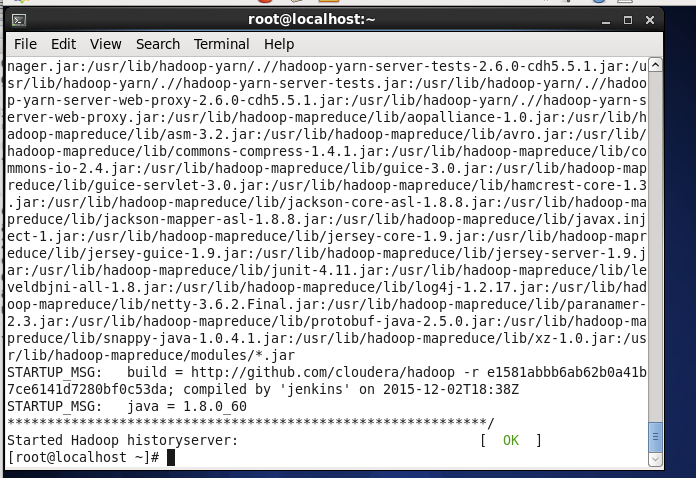
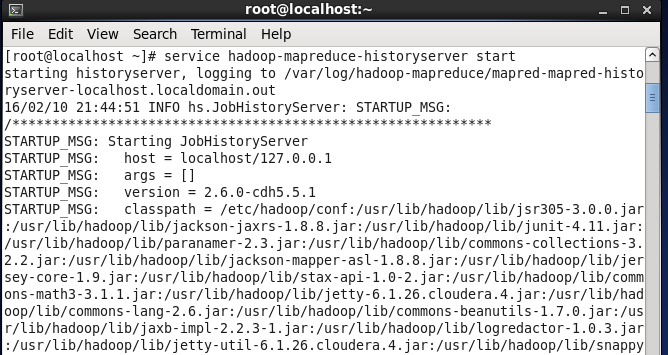
****

Description: This verifies the HDFS file structure

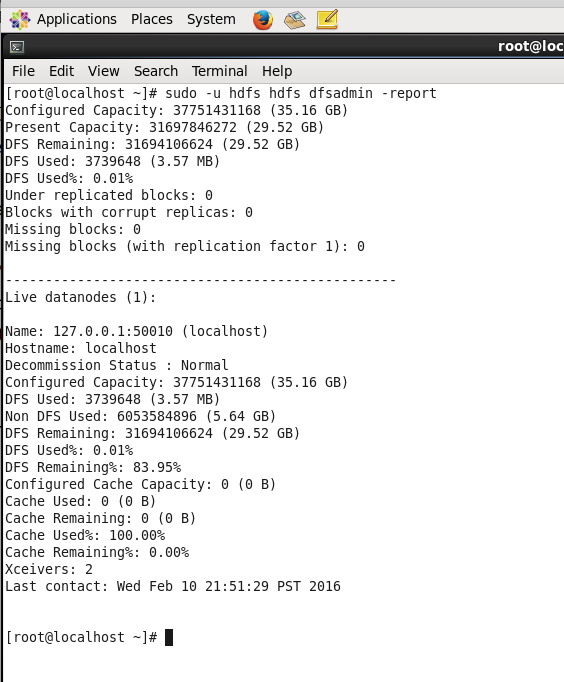
****

Description: This starts the hadoop yarn nodemanager service.

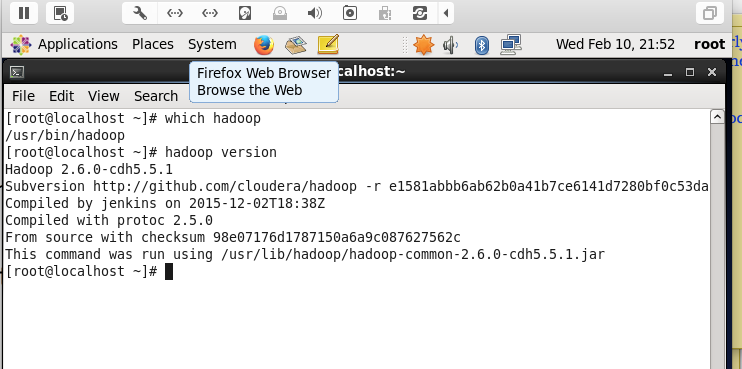
Description: This starts hadoop yarn resourcemanager service

****

Description: This command starts the hadoop mapreduce history server service

****

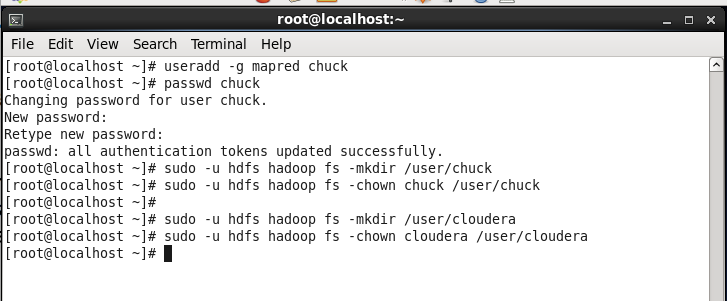
Description: This command shows status of services and data nodes.

****

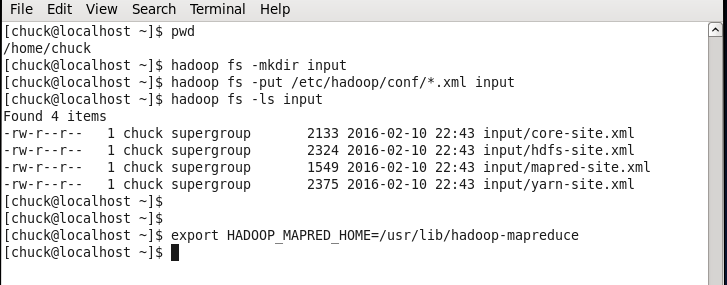
Description: This proves hadoop that is installed and then shows which version of hadoop is installed.

****

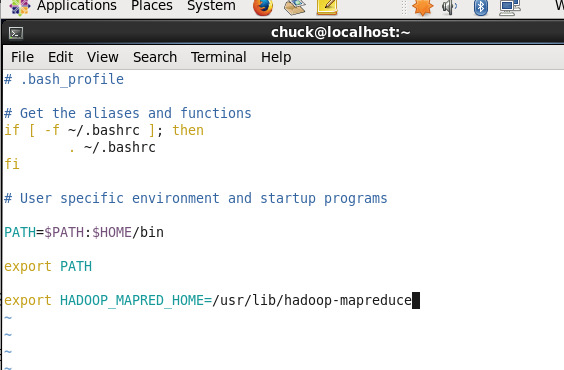
Description: This shows commands available for hadoop distributed file system

****

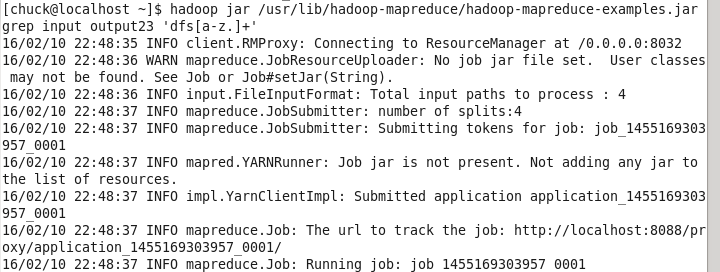
Description: These commands add user called chuck and assigns it to mapred group. Then sets password for user chuck.

****

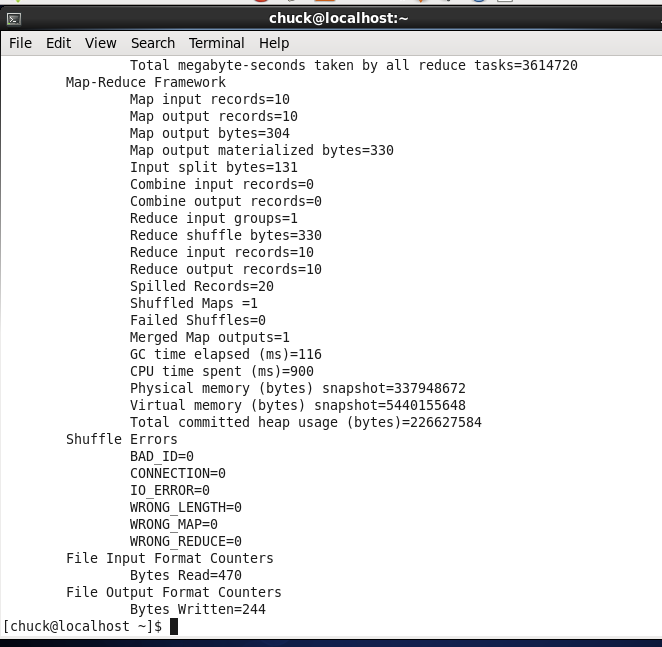
Description: Here we create a directory named input in hadoop file system. then we add some xml files in that directory

****

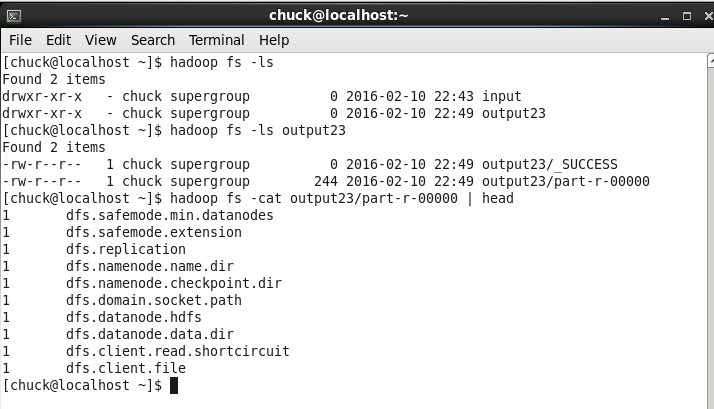
Description: Added HADOOP\_MAPRED\_HOME entry in .bash\_profile file.

****

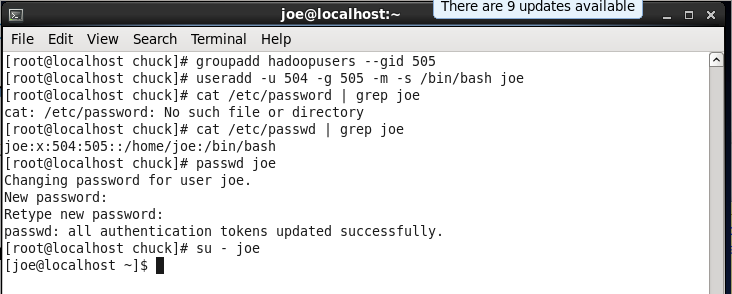
Description: Here we started running grep mapreduce job present in hadoop mapreduce examples jar file.

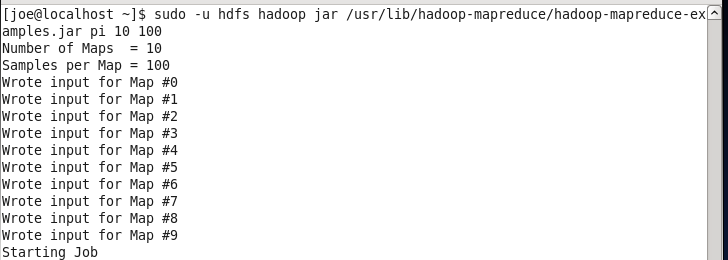
****

Description: Finished running grep mapreduce job.

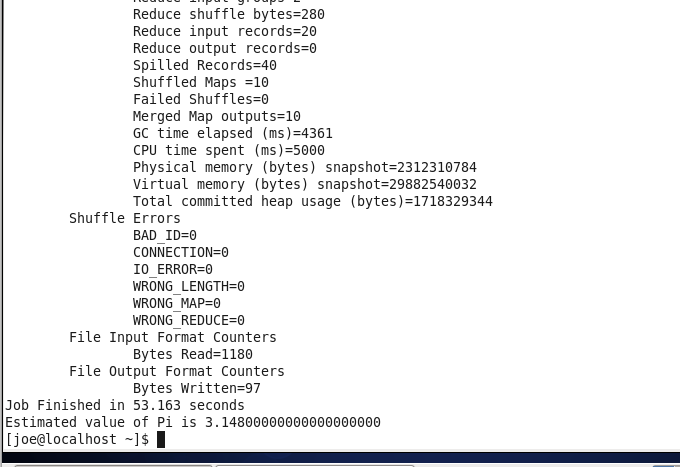
****

Description: This shows output of grep mapreduce job.

****

Description: This shows creation of new user-group called hadoopusers. Then user joe is added and assigned to that group. Password is set for user joe.****

Description: Here we start execution of mapreduce job pi present in mapreduce examples jar.

****

Description: This shows completion of mapreduce job pi.

**Problem 3)** As your new Linux user joe fetch the .txt version of James Joyce's Ulysses by issuing the following command on the command prompt:

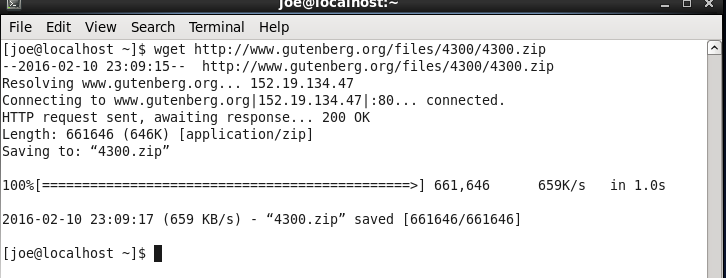
wget <http://www.gutenberg.org/files/4300/4300.zip>

Unzip the file. Open the resulting txt file with Vi and convince yourself that the life of Buck Mulligan is in front of you. Create a HDFS directory called ulysses and copy the .txt file into that HDFS directory. Do not create another HDFS directory called counted. The Map Reduce job you will run will create that directory for its output. Actually, if the directory preexists the job will raise an error. That same hadoop-mapreduce-examples.jar file mentioned in class notes and you used as the final proof that MapReduce works contains another program called wordcount. wordcount will tell you how many times a word appears in a provided text. Invoke wordcount by the following command:

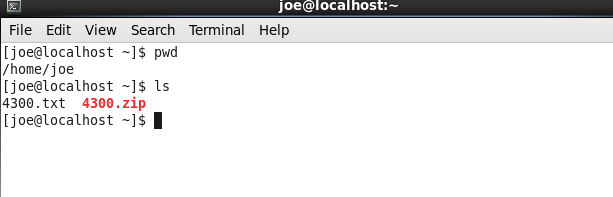
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount ulysses counted

Once the job is finished visit site <http://localhost:19888>. You will see some statistics on MapReduce jobs executed on your cluster. There will not be much for your short job. In general that is a very useful site.

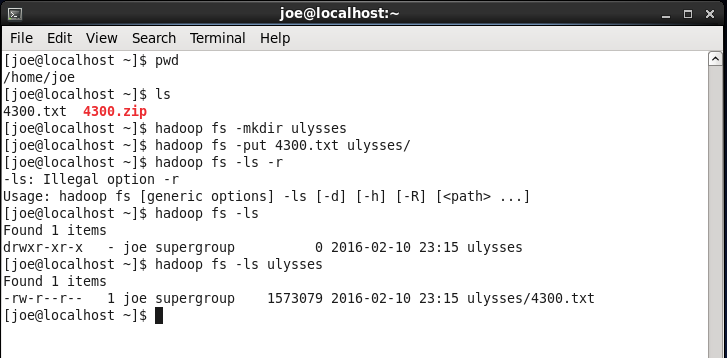
Copy results of word count analysis to the local file system. Write a small program in any language (or scripting tool) of your choice and order the counting results by the decreasing count. Present the portion of your final result which does not contain so called stop words (the, a, and, or, …) in your report. Submit top 200 words in separate .txt file with your report.



Description: Here we download 4300.zip file.

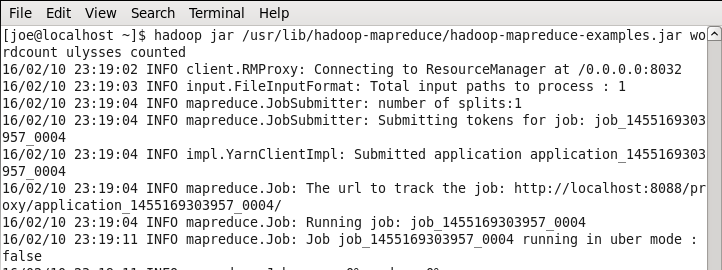


Description: 4300.txt file is present in our home directory.

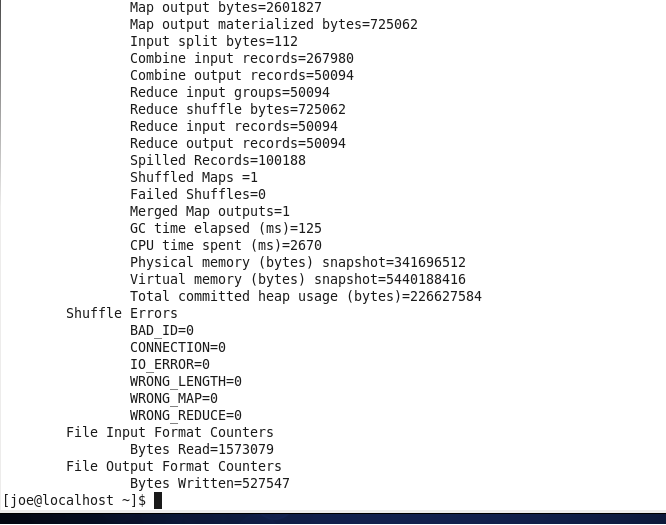


Description: we made a directory called Ulysses in the hadoop filesystem of user joe.

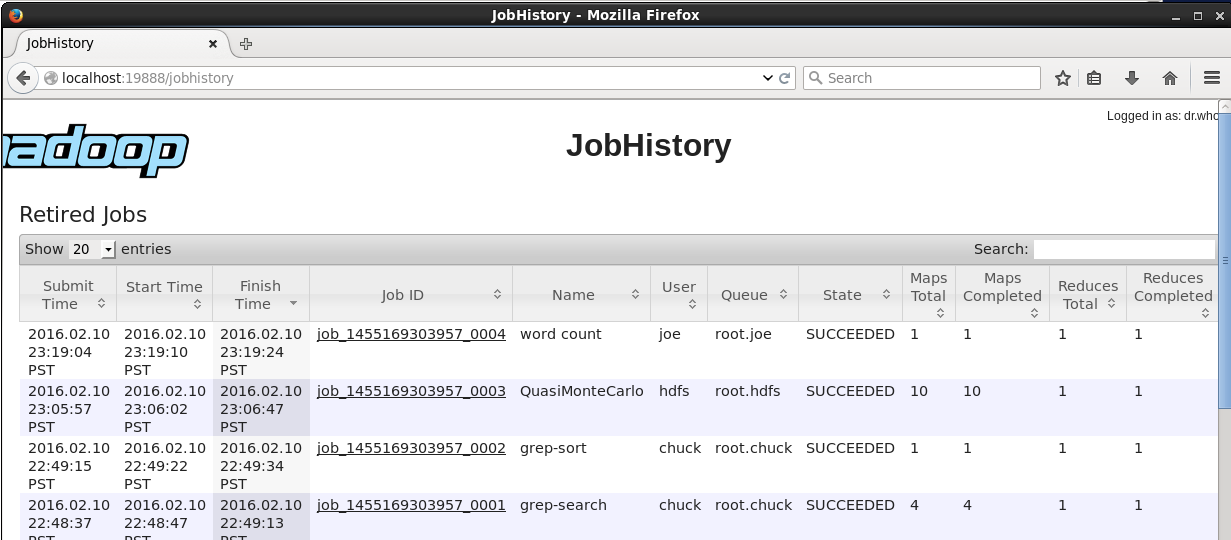
We added 4300.txt file in the Ulysses directory.

****

Description: We run the mapreduce job wordcount with Ulysses/4300.txt as its input. It outputs the findings in a directory named counted

****

Description: The mapreduce job wordcount is complete and it has given its findings in the directory named counted

****

Description: Here we can see the mapreduce job wordcount being shows as Run successfully.

Description: This is the JAVA program used for counting results in the order of deceasing count.

**public** **class** Problem\_3 {

**private** **static** String[] *STOP\_WORDS* = **new** String[] { "a", "an", "and", "are", "as", "at", "be", "by", "for", "from",

"has", "he", "in", "is", "it", "its", "of", "on", "that", "the", "to", "was", "were", "will", "with" };

**public** **static** **void** main(String[] args) {

List<String> stopWordsList = Arrays.<String>*asList*(*STOP\_WORDS*);

Charset charset = Charset.*forName*("ISO-8859-1");

**try** {

Path filePath = Paths.*get*("./files/original\_output\_of\_eulysses", "part-r-00000");

List<String> lines = Files.*readAllLines*(filePath, charset);

List<WordAndItsCount> listOfWordsAndItsCount = **new** ArrayList<WordAndItsCount>(500080);

System.***out***.println("Input file processing commenced...");

**for** (String line : lines) {

String[] stringsOnTheLine = line.split("\t");

WordAndItsCount wordAndItsCount = **new** WordAndItsCount();

wordAndItsCount.setWord(stringsOnTheLine[0]);

wordAndItsCount.setCount(Integer.*parseInt*(stringsOnTheLine[1]));

**if** (stringsOnTheLine[0] != **null**) {

**if** (stopWordsList.contains(wordAndItsCount.getWord())) {

**continue**;

}

}

listOfWordsAndItsCount.add(wordAndItsCount);

}

System.***out***.println("...Input file processing done");

Collections.*sort*(listOfWordsAndItsCount);

System.***out***.println("Output file processing commenced...");

FileWriter fileWriter = **new** FileWriter(**new** File("./files/parsed\_filtered\_sorted\_top200\_op\_of\_eulysses"));

List<WordAndItsCount> top200ListOfWordsAndItsCount = listOfWordsAndItsCount.subList(0, 200);

**for** (WordAndItsCount wordAndItsCount : top200ListOfWordsAndItsCount) {

fileWriter.write(String.*valueOf*(wordAndItsCount));

fileWriter.write(System.*lineSeparator*());

}

fileWriter.close();

System.***out***.println("...Output file processing done");

} **catch** (IOException e) {

e.printStackTrace();

}

}

**private** **static** **class** WordAndItsCount **implements** Comparable<WordAndItsCount> {

**private** String word;

**private** **int** count;

**public** String getWord() {

**return** word;

}

**public** **void** setWord(String word) {

**this**.word = word;

}

**public** **int** getCount() {

**return** count;

}

**public** **void** setCount(**int** count) {

**this**.count = count;

}

**public** String toString() {

**return** word + "\t" + count;

}

@Override

**public** **int** compareTo(WordAndItsCount wordAndItsCount) {

**if** (wordAndItsCount != **null**) {

**return** wordAndItsCount.getCount() - **this**.count;

}

**return** 0;

}

}

}

Description: These are the top 200 most frequently occurring words in the document (the stop words have not been considered).

his 3035

I 2432

her 1505

you 1362

him 1113

all 1042

The 1031

or 939

He 908

she 769

they 768

had 766

out 760

not 737

my 708

Mr 699

their 677

up 661

like 649

me 640

have 617

A 558

one 498

them 497

And 494

about 493

when 482

said. 480

what 458

which 456

your 451

says 450

so 450

if 438

Bloom 428

there 428

but 425

said 423

old 419

over 392

this 377

down 367

no 364

would 349

then 347

after 344

who 342

into 324

Stephen 315

did 305

What 302

two 302

do 299

off 299

those 285

some 283

see 282

could 280

we 280

BLOOM: 276

man 271

other 268

said, 264

little 263

She 252

back 252

too 246

His 238

it. 238

more 238

our 238

You 235

time 233

through 231

know 230

good 225

get 224

THE 223

But 217

eyes 217

round 214

They 213

only 211

now 209

under 208

long 206

any 204

never 203

where 202

put 198

way 197

very 196

hand 195

It 194

been 192

\_(He 191

us 190

him. 189

just 188

came 187

In 186

can 184

go 181

before 180

first 179

John 175

going 175

went 175

made 171

because 169

than 169

J. 167

That 164

being 164

All 163

thing 163

young 162

Mrs 158

make 157

come 153

always 152

must 151

face 150

himself 150

say 149

got 147

last 147

me. 146

night 146

day 145

ever 145

name 145

head 144

might 144

much 143

left 142

took 139

without 139

poor 138

saw 138

street 138

told 138

something 136

let 135

white 135

him, 133

how 133

new 133

right 133

though 133

am 132

well 132

--I 130

Then 130

upon 130

and, 129

own 129

same 129

I'm 128

O 128

don't 128

most 128

Father 127

tell 127

till 127

woman 125

again 123

against 123

between 123

every 123

look 123

want 123

To 121

bit 121

give 121

such 121

turned 120

course 119

towards 119

God 118

gave 118

take 118

Dedalus 116

Mulligan 115

away 115

it, 115

here 114

you. 114

hat 113

O, 112

We 111

black 111

that. 111

too. 111

S. 110

behind 110

three 110

voice 110

How 108

For 107

**Problem 4).** Consider a symmetric matrix

A =

Using R demonstrate that all three eigenvectors of that matrix are mutually orthogonal. Let be the matrix of eigenvectors of matrix A. Calculate product of tree matrices:

A Λ

Symbol T indicates the transpose matrix. Google around for properties of eigenvectors and eigenvalues of real symmetric matrices. What is the general statement you can make about the observation on the value of the above product. Include copies of your R commands in your MS Word report.

> A <- matrix(c(3,2,4,2,0,2,4,2,3), nrow=3)

> A

[,1] [,2] [,3]

[1,] 3 2 4

[2,] 2 0 2

[3,] 4 2 3

Description: Here we construct the matrix A.

> eigen(A)

$values

[1] 8 -1 -1

$vectors

[,1] [,2] [,3]

[1,] 0.6666667 0.7453560 0.0000000

[2,] 0.3333333 -0.2981424 -0.8944272

[3,] 0.6666667 -0.5962848 0.4472136

> eigen(A)$vectors

[,1] [,2] [,3]

[1,] 0.6666667 0.7453560 0.0000000

[2,] 0.3333333 -0.2981424 -0.8944272

[3,] 0.6666667 -0.5962848 0.4472136

Description: Here we show eigen vector matrix of A.

> eigen(A)$vectors[,1]

[1] 0.6666667 0.3333333 0.6666667

> eigen(A)$vectors[,2]

[1] 0.7453560 -0.2981424 -0.5962848

> eigen(A)$vectors[,3]

[1] 0.0000000 -0.8944272 0.4472136

Description: Here we show the 3 eigen vectors of A

Proof 1:

> round(sum(eigen(A)$vectors[,1] \* eigen(A)$vectors[,2]))

[1] 0

Description: here we see that dot product of 1st and 2nd eigen vectors of A is 0.

So, 1st and 2nd eigen vectors are mutually orthogonal.

> round(sum(eigen(A)$vectors[,2] \* eigen(A)$vectors[,3]))

[1] 0

Description: here we see that dot product of 2nd and 3rd eigen vectors of A is 0.

So, 2nd and 3rd eigen vectors are mutually orthogonal.

> round(sum(eigen(A)$vectors[,1] \* eigen(A)$vectors[,3]))

[1] 0

Description: here we see that dot product of 1st and 3rd eigen vectors of A is 0.

So, 1st and 3rd eigen vectors are mutually orthogonal.

Proof 2:

> round(eigen(A)$vectors %\*% t(eigen(A)$vectors))

[,1] [,2] [,3]

[1,] 1 0 0

[2,] 0 1 0

[3,] 0 0 1

Description: Here we find matrix product of A’s eigen-vector-matrix and its transpose.

We see that the matrix product = Identity matrix

A square matrix is orthogonal if matrix product of itself with its transpose is an identity matrix.

And since (eigen(A)$vectors %\*% t(eigen(A)$vectors)) = Identity matrix,

We can conclude that eigen-vectors-matrix of A is orthogonal.

i.e. all three eigen vectors of A are mutually orthogonal.

Proof 3:

A is a symmetric matrix. And eigen-vectors-matrix of a symmetric matrix is always orthogonal.

i.e. all three eigen vectors of A are mutually orthogonal.

> round(t(eigen(A)$vectors) %\*% A %\*% eigen(A)$vectors)

[,1] [,2] [,3]

[1,] 8 0 0

[2,] 0 -1 0

[3,] 0 0 -1

Description: Here we find matrix product of transpose of A’s eigen-vector-matrix, A and A’s eigen-vector-matrix.

We see that the matrix product = a diagonal matrix

So basically A Λ gives a diagonal matrix with diagonal entries=eigen values of A.

General Statement:

For a symmetrix matrix A,

A Λ = a diagonal matrix (where diagonal entries=eigen values of A)

(where Λ is eigen matrix of A and is transpose of Λ)