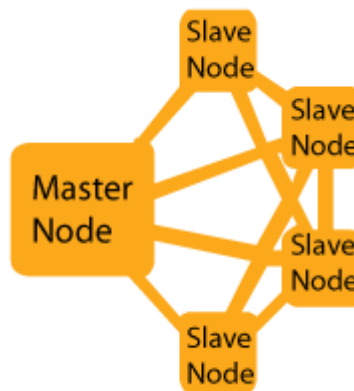


## Lab 7: Hadoop MapReduce

### 1- What is Hadoop?

Hadoop uses a distributed processing architecture called MapReduce in which a task is mapped to a set of servers for processing. The results of the computation performed by those servers are then reduced down to a single output set. One node, designated as the master node, controls the distribution of tasks. The following diagram shows a Hadoop cluster with the master node directing a group of slave nodes which process the data.



### 2-What is Amazon Elastic MapReduce?

With Amazon Elastic MapReduce (Amazon EMR) you can analyze and process vast amounts of data. It does this by distributing the computational work across a cluster of virtual servers running in the Amazon cloud. The cluster is managed using an open-source framework called Hadoop.

## Objectives








- Learn how to write map reduce code using java.
- Learn how to setup Hadoop locally and test your Mapreduce code using Single Node cluster
- Learn how to run MapReduce code in cloud using AWS EMR feature

## 1 Setup a local Hadoop Environment:

In this section, we will go over the steps to create Hadoop applications in Java using Eclipse

Go to <https://archive.apache.org/dist/hadoop/core/hadoop-2.7.3/> and download the version-2.7.3 of Hadoop release on your computer.

# Index of /dist/hadoop/core/hadoop-2.7.3

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">Parent Directory</a>		-	
 <a href="#">hadoop-2.7.3-src.tar.gz</a>	2016-08-25 19:25	17M	
 <a href="#">hadoop-2.7.3-src.tar.gz.asc</a>	2016-08-25 19:25	521	
 <a href="#">hadoop-2.7.3-src.tar.gz.mds</a>	2016-08-25 19:25	1.1K	
 <a href="#">hadoop-2.7.3.tar.gz</a>	2016-08-25 19:25	204M	
 <a href="#">hadoop-2.7.3.tar.gz.asc</a>	2016-08-25 19:25	521	
 <a href="#">hadoop-2.7.3.tar.gz.mds</a>	2016-08-25 19:25	958	

Extract the content of the .tar.gz file to a location of your choice.

On my Mac, I extracted to a location: [/Users/vumai/Documents/Hadoop/hadoop-2.7.3](#)

In windows you need to extract twice (first to tar and then next to original folder) using 7-zip or similar extractor.

After you extract the content of the compressed hadoop-2.7.3.tar.gz file to a folder named hadoop-2.7.3, go to hadoop-2.7.3/bin, you will see that the folder contains the commands that we will use to run Hadoop Mapreduce, especially important is the hadoop command in that folder.

**Windows users** need to do an extra step. First download **bin.zip** from Canvas and extract it. Remove **hadoop-2.7.3/bin** folder and copy your extracted **bin** folder in **hadoop-2.7.3/** folder.

For Hadoop to work, Java needs to be installed (You should install it during lab 1). Check if you have install Java using the following command using command line: java -version. You should be able to see something like this (java version can be different depending on your installation):

```
[~$ java -version
java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)
```

In the next step, we will set some environment variables so that we can invoke the hadoop command on the command line.

**If you are using Mac**, there is a hidden file in your home folder called `.bash_profile` which allows you to set permanent environment variables.

If the `.bash_profile` file does not exist in your home folder, just create that file in your home folder using the touch command:

```
~$ touch .bash_profile
```

You can then open `.bash-profile` in your home folder using a text editor, for example

```
| ~$ vim .bash_profile
```

We will do the following tasks:

- Set the `JAVA_HOME` variable to the place where you installed the Java sdk
- Set the `HADOOP_HOME` variable to the place where you extract the Hadoop distribution

The content of my `.bash_profile` file is shown below:

```
export GOPATH=/Users/vumai/GDrive/Go
#export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home
export JAVA_HOME=$(/usr/libexec/java_home)
export HADOOP_HOME=/Users/vumai/Documents/Hadoop/hadoop-2.7.3
export PATH=$PATH:$GOPATH/bin:$HADOOP_HOME/bin
```

Save the `.bash_profile` file, quit your Terminal and open it again.

Now you can check all your environment variables:

```
[~$ echo $JAVA_HOME ]
/Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home
[~$ echo $HADOOP_HOME ]
/Users/vumai/Documents/Hadoop/hadoop-2.7.3
[~$ echo $PATH ]
/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin:/opt/X11/bin:/usr/l
ocal/go/bin:/usr/texbin:/Users/vumai/GDrive/Go/bin:/Users/vumai/
Documents/Hadoop/hadoop-2.7.3/bin
```

**If you are using WINDOWS**, you can set these environmental variables using command line.

In my machine java is installed in: `C:\Program Files\Java\jdk1.8.0_25`

Note that, java version can be different depending on your installation

So I can add JAVA\_HOME by using the command:

```
C:\Users\ee04440>set JAVA_HOME=C:\PROGRA~1\Java\jdk1.8.0_25
```

I extracted my hadoop 2.7.3 folder in c:\scratch. So, I can add HADOOP\_HOME by using the command:

```
C:\Users\ee04440>set HADOOP_HOME=C:\scratch\hadoop-2.7.3
```

Then I updated the PATH variable using the command:

```
C:\Users\ee04440>set PATH=%PATH%;%HADOOP_HOME%\bin
```

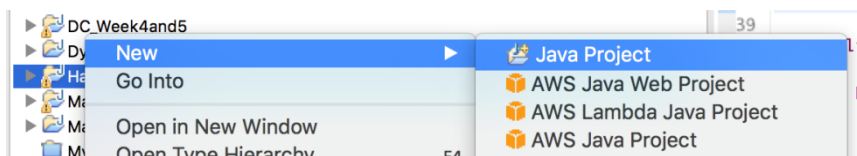
Test your hadoop command; you should be able to see something like:

```
[~$] hadoop version
Hadoop 2.7.3
Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r baa91f7c6bc9cb9
2be5982de4719c1c8af91ccff
Compiled by root on 2016-08-18T01:41Z
Compiled with protoc 2.5.0
From source with checksum 2e4ce5f957ea4db193bce3734ff29ff4
This command was run using /Users/vumai/Documents/Hadoop/hadoop-2.7.3/share/hado
op/common/hadoop-common-2.7.3.jar
```

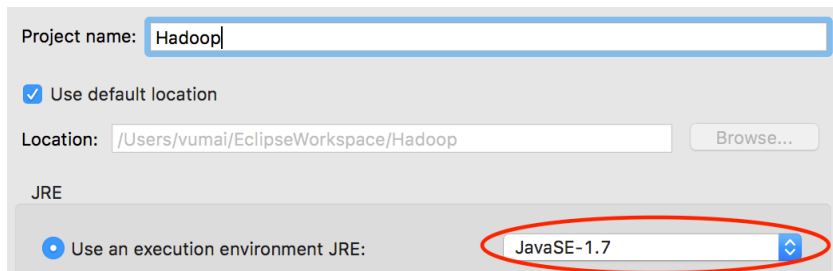
Now we are ready to run our Hadoop projects. In summary, we will write code and compile our Hadoop code on Eclipse, then export our code to a jar file and run Hadoop on the command line using that jar file.

## 1.1 Setup and write MapReduce code in Eclipse

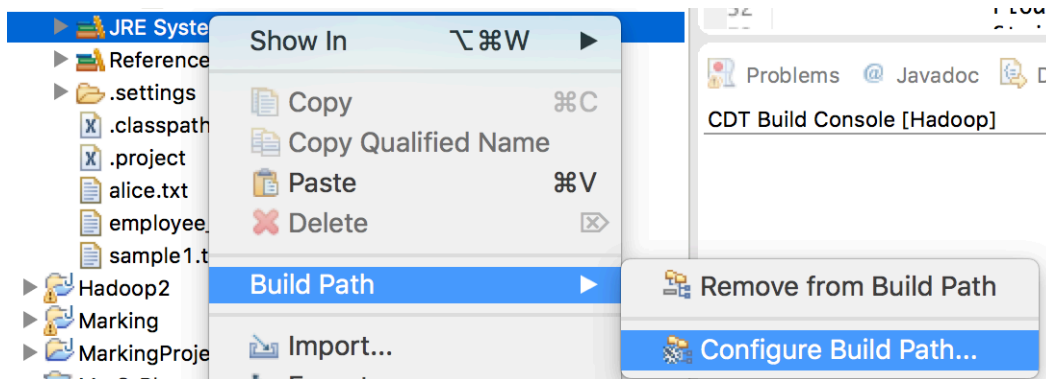
Create a Java project on Eclipse called Hadoop



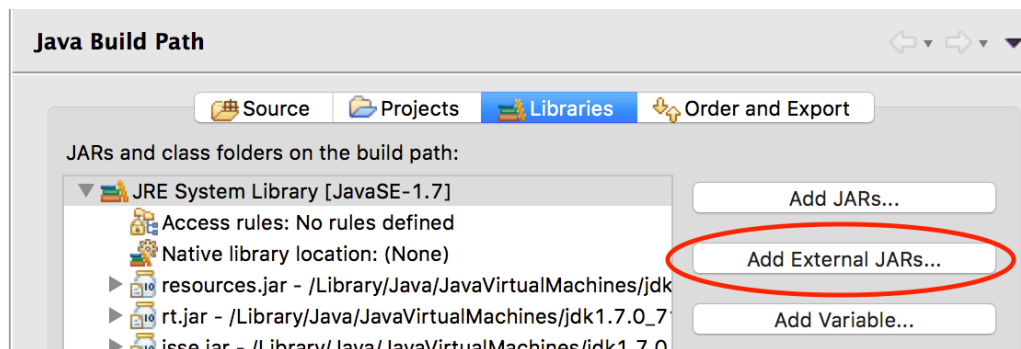
Make sure you use Java 1.7



Now we need to add 2 more external jar files to our newly created Hadoop project. Right click on the JRE system library of your project, select Build Path, Configure Build Path:



Click on Add External JARs

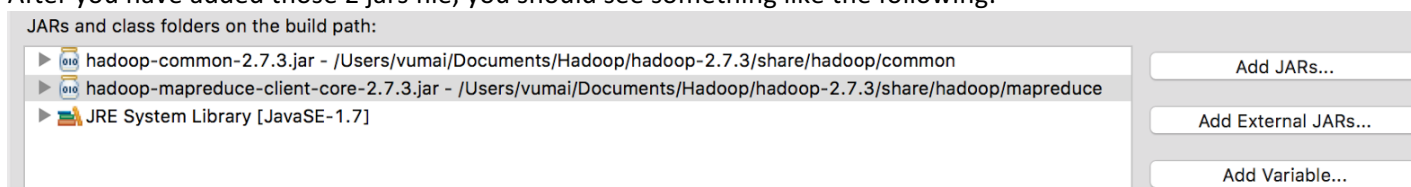


Search in the folder where you have extracted Hadoop and add the following 2 jar files (i've shown below the full path to those jar files):

\$HADOOP\_HOME/share/hadoop-2.7.3/common/hadoop-common-2.7.3.jar

\$HADOOP\_HOME/share/hadoop-2.7.3/mapreduce/hadoop-mapreduce-client-core-2.7.3.jar

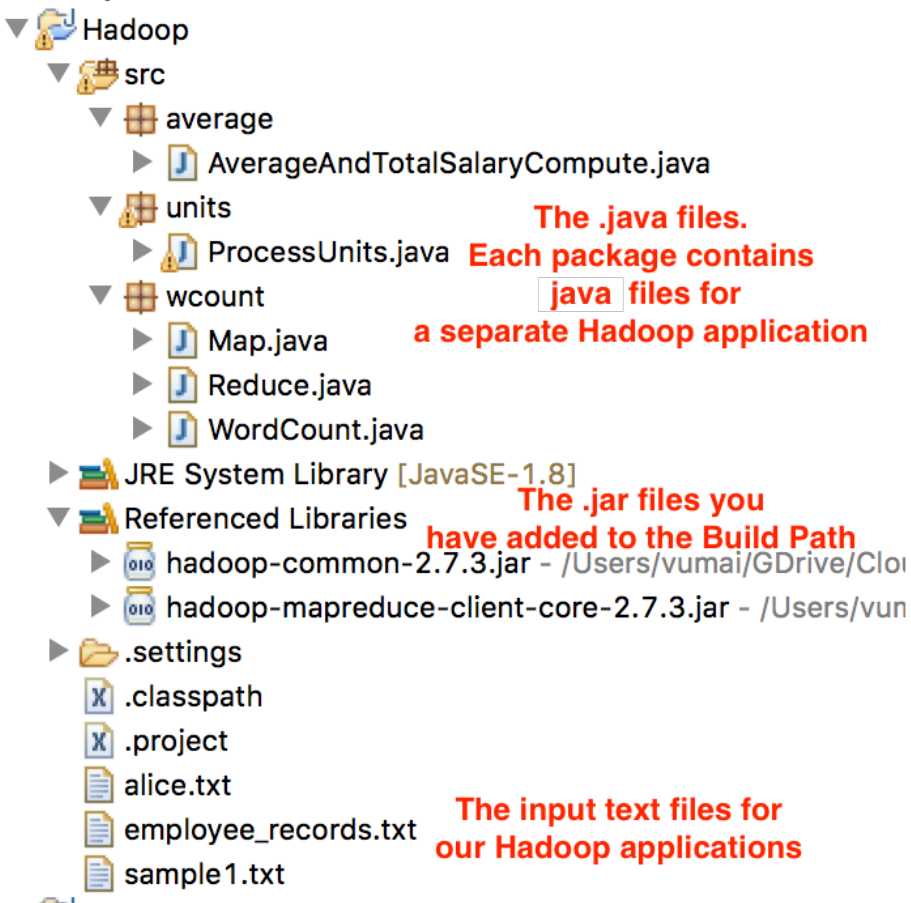
After you have added those 2 jars file, you should see something like the following:



In src folder create 3 new package

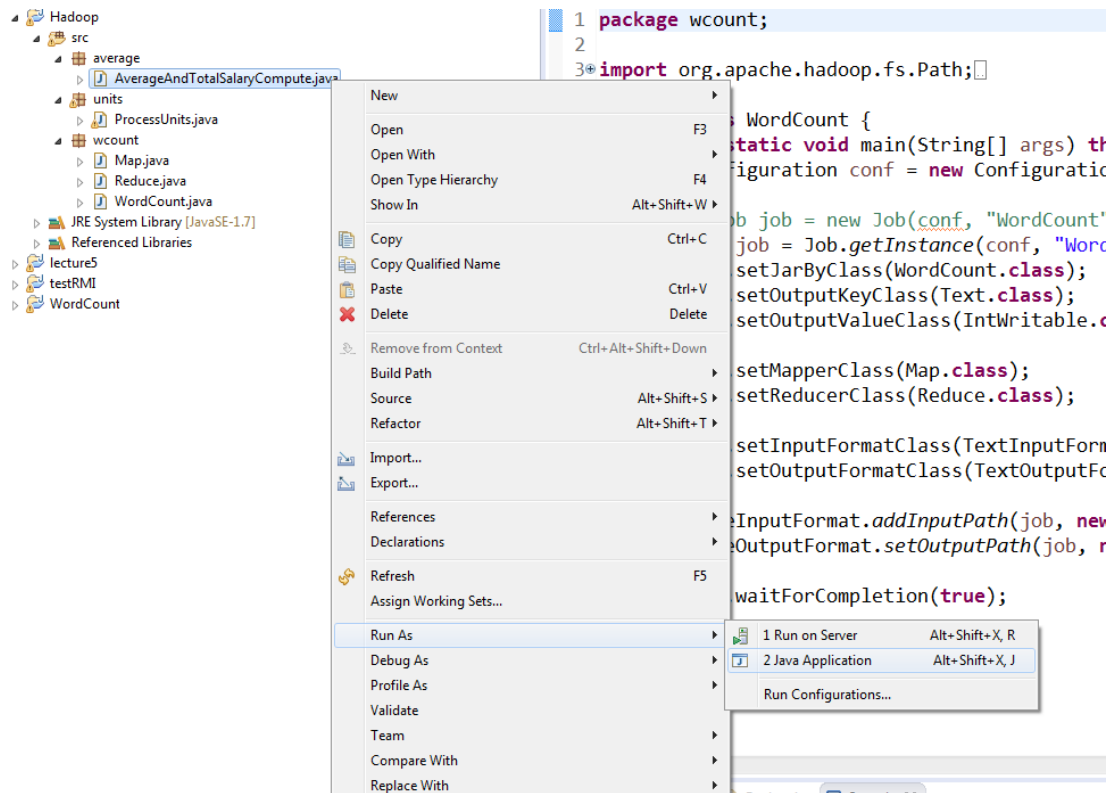
- 1) average
- 2) units
- 3) wcount

Now download the mrcode.zip folder from Blackboard. Copy the java files inside that folder in your Hadoop project according to the following structure.

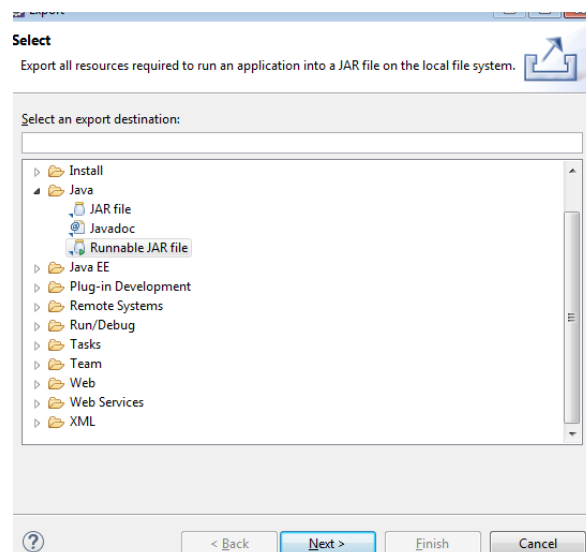


Go through the codes. Try to understand how the MapReduce process is implemented in the code.

Now run each of 3 packages as java application. Don't worry if you get exception.

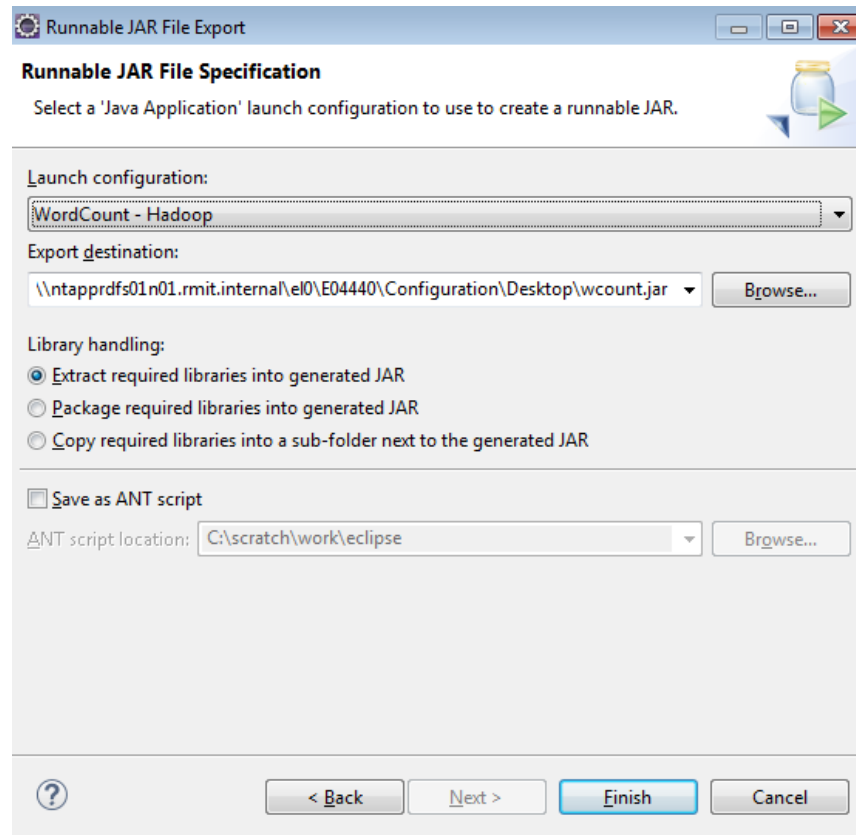


Now export each of 3 package as runnable jar. For example, to export wcount package as jar right click on the package name and select export. In the Export window select **Java > Runnable jar file**



In next window,

Select the main file as Launch configuration. For example, for wcount package the main function is in WordCount.java file. Also give the jar file name same as package name (e.g. wcount.jar). As in the following screenshot:



Click at Finish. Click Ok if you see any warning. Now your jar file is ready to run in Hadoop.

Copy all generated runnable jar files inside your hadoop-2.7.3 folder. Also create a folder **input** inside hadoop-2.7.3 folder.

### Run the first project: Electrical Consumption

Copy the inputs/sample1.txt file to the input folder you just created. Then open command line and CD to your hadoop-2.7.3 folder. Then run the command:

```
hadoop jar units.jar input output
```

Here we have run the hadoop command to execute the units.jar file. The file process the data inside input folder and generate the output inside **output** folder. Note that here you don't need to create an output folder. Hadoop will automatically create it and will place the result in output folder.

After hadoop finish running you will see output like the following



```

File System Counters
  FILE: Number of bytes read=8184
  FILE: Number of bytes written=570617
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
Map-Reduce Framework
  Map input records=5
  Map output records=5
  Map output bytes=45
  Map output materialized bytes=39
  Input split bytes=111
  Combine input records=5
  Combine output records=3
  Reduce input groups=3
  Reduce shuffle bytes=39
  Reduce input records=3
  Reduce output records=3
  Spilled Records=6
  Shuffled Maps =1
  Failed Shuffles=0
  Merged Map outputs=1
  GC time elapsed (ms)=0
  Total committed heap usage (bytes)=514850816
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=343
File Output Format Counters
  Bytes Written=36

```

Let's look at the output

```

~/EclipseWorkspace/Hadoop$ ls output/
_SUCCESS      part-00000
~/EclipseWorkspace/Hadoop$ cat output/part-00000
1981      34
1984      40
1985      45

```

In the figure above, we see that when Hadoop has finished running successfully, you will see the content of the output folder as shown above.

To view the content of the output file, you use the cat command in mac command line or you can open the output file in some text editor.

Now remove the sample1.txt from input folder and remove the output folder. Then copy employee\_records.txt in input folder. Now test the second package **average** by using the command

```
hadoop jar average.jar input output
```

On success your output will be:

```

~/EclipseWorkspace/Hadoop$ ls output/
_SUCCESS      part-r-00000
~/EclipseWorkspace/Hadoop$ cat output/part-r-00000
F      Total: 291800.0 :: Average: 7117.073
M      Total: 424363.34 :: Average: 6333.7812

```

Once again remove the employee\_records.txt from input folder and remove the output folder. Then copy alice.txt in input folder. Now test the third package **wcount** by using the command.

```
hadoop jar wcount.jar input output
```

On success you will have output something like:

```
[~/EclipseWorkspace/Hadoop$ ls output/  
_SUCCESS      part-r-00000  
[~/EclipseWorkspace/Hadoop$ cat output/part-r-00000  
a          269  
about     32  
above      1  
accustomed      1  
across    5  
actually      1  
added      6
```

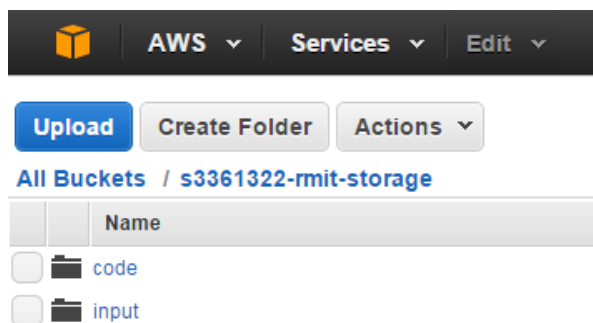
Here hadoop is generating only one output file because we are using a single node cluster. That is only one processing node can work. This process is helpful when your dataset is small and when you need to test your developed Mapreduce code locally. But if your dataset is large a single node cluster will not be enough and you need to take the advantage of cloud by using a Multi-Node cluster. AWS has such support in EMR. Now we will use our developed MapReduce code in EMR.

## 2 Solve word count problem using EMR (Amazon Elastic MapReduce)

### 2.1 Prepare your scripts and input data

1- Log in to your AWS management console. Go to **S3** → **yourbucketname**. If you do not have buckets then create one. Let your bucket name is: sxxxxxx-rmit-storage where sxxxxxx is your student number.

2- Create two folders inside your bucket and name them **code** and **input**



3- Inside **code** folder upload the **wcount.jar** file you created before.

4- Download the text file from here: <http://www.umich.edu/~umfandsf/other/ebooks/alice30.txt> and upload alice30.txt inside **input** folder of your Bucket

## 2.2 Create Amazon MapReduce cluster

1- Login to AWS management console and from services select "EMR".

2- Click on **create cluster** and you will see a screen like following.

The screenshot shows the 'General Configuration' and 'Software configuration' sections of the AWS EMR console. In the 'General Configuration' section, the 'Cluster name' is 'My cluster', 'Logging' is checked, the 'S3 folder' is 's3://aws-logs-181740218722-us-west-2/elasticmapredu', and the 'Launch mode' is 'Cluster'. The 'Software configuration' section shows the 'Vendor' as 'Amazon', the 'Release' as 'emr-5.0.0', and the 'Applications' as 'Core Hadoop: Hadoop 2.7.2 with Ganglia 3.7.2, Hive 2.1.0, Hue 3.10.0, Mahout 0.12.2, Pig 0.16.0, and Tez 0.8.4'.

3- Give a name to your cluster. In the **Logging S3 folder location** field browse and select: s3://your\_bucket\_name, where your\_bucket\_name is one of your bucket.

4- In Launch mode select **Step execution**

The screenshot shows the 'General Configuration', 'Add steps', and 'Hardware configuration' sections of the AWS EMR console. In the 'General Configuration' section, the 'Cluster name' is 'My cluster', 'Logging' is checked, the 'S3 folder' is 's3://aws-logs-181740218722-us-west-2/elasticmapredu', and the 'Launch mode' is 'Step execution'. The 'Add steps' section shows a 'Step type' dropdown menu with options: 'Custom JAR', 'Select a step', 'Streaming program', 'Hive program', 'Pig program', and 'Spark application'. The 'Software configuration' section shows the 'Vendor' as 'Custom JAR', the 'Release' as 'emr-5.0.0', and the 'Applications' as 'Hadoop 2.7.2'. The 'Hardware configuration' section shows the 'Instance type' as 'm3.xlarge' and the 'Number of instances' as '3' (1 master and 2 core nodes).

5- Select step type Custom **JAR** as in the screenshot

6- Under **software configuration** select vendor **Amazon** and Release emr-5.5.0 (which will show hadoop 2.7.3 as Applications).

7- Click at **Configure** button beside **Custom Jar** selection. A popup window will open. Fill it as shown in the following figure. In the arguments you have to provide the input location as the first arguments and the output location as the second arguments as shown here.

s3://sxxxxxx-rmit-storage/input

s3://sxxxxxx-rmit-storage/output1

**Add Step** [X]

**Step type** Custom JAR

**Name\*** Custom JAR

**JAR location\*** s3://s3361322-rmit-storage/code/wordcount.jar JAR location maybe a path into S3 or a fully qualified java class in the classpath.

**Arguments** s3://s3361322-rmit-storage/input  
s3://s3361322-rmit-storage/output1  
These are passed to the main function in the JAR. If the JAR does not specify a main class in its manifest file you can specify another class name as the first argument.

**Action on failure** Terminate cluster ▼ What to do if the step fails.

Cancel Add

Also select Terminate cluster for Action on failure.

Note that, In case of Output **S3 location** you don't need to create **output** directory in our bucket. The EMR program will do this. But make sure you type '/output1/' after your bucket name.

Note that, in this cluster you are using 3 m3.xlarge VM instances. m3.xlarge is the least powerful instance you can use here. And minimum number of instance also should be 3. But you can use more powerful instance and higher number of instances here. The more you will use the more you will need to pay. We kept everything at minimum level to be in safe side. But from concept of MapReduce what you know that more powerful instance and more number of instances in cluster will improve the performance of MapReduce operation.

8- Keep **Security and Access** as default.

10- Now click on **Create cluster** and wait until it finishes. It can take 10 minutes or more to complete. It will show "**Starting**" for some time. It may take more than 5-10 minutes to start. **Starting** means the system is configuring the clusters you have created. When MapReduce job will start the status will show "**Running**" like following screen.

Cluster: My cluster **Running** Running step

**Connections:** [Enable Web Connection](#) – Resource Manager ... (View All)

**Master public DNS:** ec2-52-10-196-109.us-west-2.compute.amazonaws.com [SSH](#)

**Tags:** -- [View All](#) / [Edit](#)

Summary	Configuration Details	Network and Hardware	Security and Access
<b>ID:</b> j-3TPBQBA21Q9Q <b>Creation date:</b> 2015-03-18 14:27 (UTC+11) <b>Elapsed time:</b> 5 minutes <b>Auto-terminate:</b> Yes <b>Termination protection:</b> On <a href="#">Change</a>	<b>AMI version:</b> 3.5.0 <b>Hadoop distribution:</b> Amazon 2.4.0 <b>Applications:</b> -- <b>Log URI:</b> s3://s3361322-rmit-storage/ <b>EMRFS:</b> Disabled <b>consistent view:</b>	<b>Availability zone:</b> us-west-2b <b>Subnet ID:</b> subnet-fcc0198b <b>Master:</b> <b>Running</b> 1 m1.large <b>Core:</b> <b>Running</b> 2 m1.large <b>Task:</b> --	<b>Key name:</b> -- <b>EC2 instance profile:</b> EMR_EC2_DefaultRole <b>EMR role:</b> EMR_DefaultRole <b>Visible to all users:</b> All <a href="#">Change</a> <b>Security groups for Master:</b> sg-d2ffceb7 (ElasticMapReduce-master) <b>Security groups for Core &amp; Task:</b> sg-ddffceb8 (ElasticMapReduce-slave)

Wait for some more time. If the status shows "**Terminated steps completed**" like following screen, then it means MapReduce job has finished and the outputs are now ready for viewing.

## 2.3 View the output result

Once your job is marked as completed, go to AWS Management Console → S3, and click on the **output1** folder you specified (which should be **output**). You will see some files named part-0000, part-0001,... which contain the results of your MapReduce job. The number of output files generated based on number of reducers involvement to complete this task. This is managed by EMR.

AWS

Services

Edit

Abdur ForkanGlobalSupport

Upload

Create Folder

Actions

None

Properties

Transfers

All Buckets / s3361322-rmit-storage / output

	Name	Storage Class	Size	Last Modified
<input type="checkbox"/>	_SUCCESS	Standard	0 bytes	Wed Mar 18 14:34:36 GMT+1100 2015
<input type="checkbox"/>	part-00000	Standard	2.1 KB	Wed Mar 18 14:34:21 GMT+1100 2015
<input type="checkbox"/>	part-00001	Standard	2 KB	Wed Mar 18 14:34:32 GMT+1100 2015
<input type="checkbox"/>	part-00002	Standard	1.9 KB	Wed Mar 18 14:34:32 GMT+1100 2015
<input type="checkbox"/>	part-00003	Standard	2.1 KB	Wed Mar 18 14:34:36 GMT+1100 2015
<input type="checkbox"/>	part-00004	Standard	2 KB	Wed Mar 18 14:34:29 GMT+1100 2015

You can download all the files and open them in text editor to view the results. The results contain the word and number of occurrence of that word in "alice30.txt" file.

**Important note:** The commands we run to launch our job in this example are set to automatically terminate the EC2 instances. However, as you get into more custom commands, this will not necessarily be the case. In those situations, you have to terminate the instance. Otherwise, you will get credited for the time they are running (even if the job completed). To terminate an instance, you can Go on Amazon Management Console → Elastic MapReduce, select your Job Flow and click on Terminate.