AmazonEMRIntro

Introduction

Amazon EMR provides a managed Hadoop framework that makes it easy, fast, and cost-effective to process vast amounts of data across dynamically scalable Amazon EC2 instances.

You can run popular distributed frameworks such as Apache Spark, HBase, Presto, and Flink in Amazon EMR, and interact with data in other AWS (Amazon Web Services) data stores such as Amazon S3 and Amazon DynamoDB.

Amazon EMR securely and reliably handles a broad set of big data use cases, including log analysis, web indexing, data transformations (ETL), machine learning, financial analysis, scientific simulation, and bioinformatics.

References:

Amazon Elastic MapReduce

https://aws.amazon.com/emr/

Amazon EMR Management Guide
http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-what-is-emr.html

Amazon Elastic MapReduce API
http://docs.aws.amazon.com/ElasticMapReduce/latest/API/Welcome.html

Amazon EMR Release Guide
http://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-hadoop-application.html

Apache Hadoop

http://hadoop.apache.org/

Programming Elastic MapReduce Using AWS Services to Build an End-to-End Application, Kevin Schmidt, Christopher Phillips, O'Reilly Media, December 2013.
http://shop.oreilly.com/product/0636920029304.do

Key components

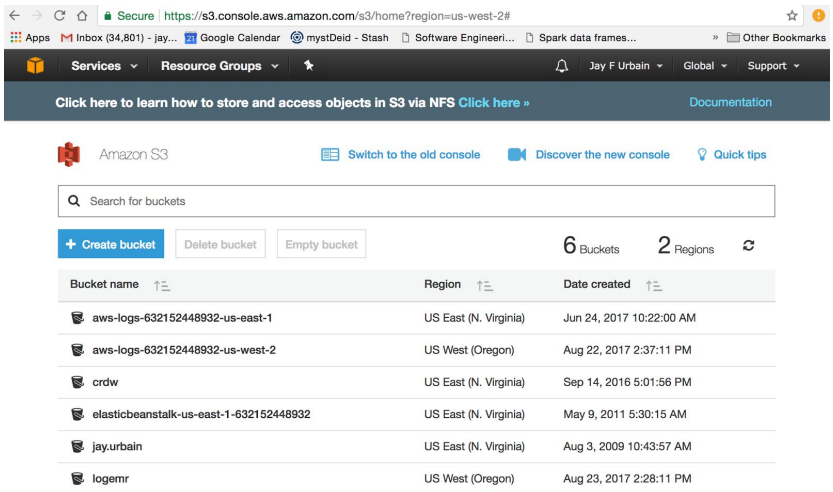
Amazon Elastic MapReduce (EMR)

Amazon EMR is an in-the-cloud platform of the Hadoop framework. Amazon EMR makes heavy use of the Amazon Simple Storage Service (S3) to store analysis results and host data sets for processing, and leverages Amazon Elastic Cloud Compute (EC2) resources to run applications.

There is an additional charge of about 30 percent for the EMR EC2 instances. To read Amazon's overview of EMR, visit the Amazon EMR web page.

Amazon Simple Storage Service (S3)

Amazon S3 is the persistent storage for AWS. It provides a web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the Web. There are some restrictions. Data in S3 must be stored in named buckets, and any single object can be no more than 5 terabytes in size. The data stored in S3 is highly durable and is stored in multiple facilities and multiple devices within a facility.



There are standard REST- and SOAP-based web service APIs to interact with files stored on S3.

Amazon S3's permanent storage will be used to store data sets and computed result sets generated by Amazon EMR Job Flows. Applications built with Amazon EMR need to use some S3 services for data storage.

Amazon Elastic Compute Cloud (EC2)

Amazon EC2 makes it possible to run multiple instances of virtual machines on demand inside any one of the AWS regions. You can start as many or as few instances as you need without having to buy or rent physical hardware like in traditional hosting services. In the case of Amazon EMR, this means we can scale the size of your Hadoop cluster to any size needed. Individual EC2 instances come in a variety of sizes and specifications to meet the needs of different types of applications. There are instances tailored for high CPU load, high memory, high I/O, etc.

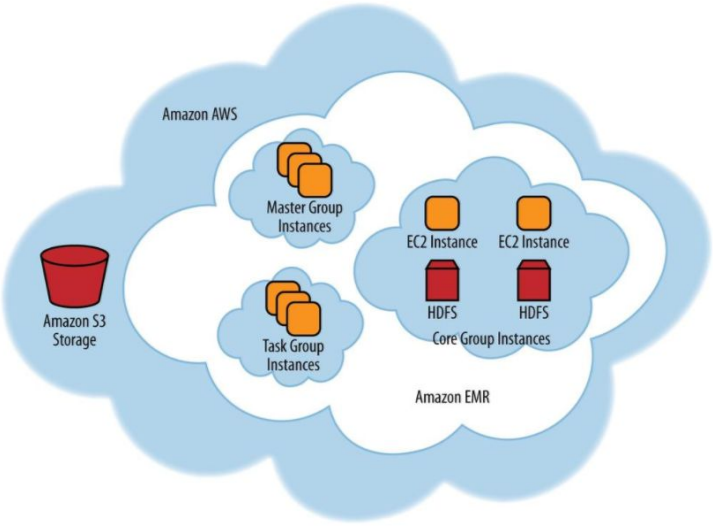
To read Amazon's overview of EC2, visit the Amazon EC2 web page. Amazon EC2 instances are used as part of an Amazon EMR cluster.

Amazon Elastic MapReduce Architecture

Amazon EMR allows users to launch and use resizable Hadoop clusters inside of Amazon's infrastructure. Amazon EMR, like Hadoop, can be used to analyze large data sets. EMR greatly simplifies the setup and management of the cluster of Hadoop and MapReduce components.

EMR instances use Amazon's pre-built and customized EC2 instances, which can take full advantage of Amazon's infrastructure and other AWS services. These EC2 instances are invoked when a new Job Flow is started to form an EMR cluster. A *Job Flow* is Amazon's term for the complete data processing that occurs through a number of compute steps in Amazon EMR. A Job Flow is specified by the MapReduce application and its input and output parameters.

https://aws.amazon.com/emr/



Amazon EMR performs the computational analysis using the MapReduce framework. The MapReduce framework splits the input data into smaller fragments, or shards, that are distributed to the nodes that compose the cluster

Master group instance

The master group instance manages the Job Flow and allocates all the needed executables, JARs, scripts, and data shards to the core and task instances. The master node monitors the health and status of the core and task instances and also collects the data from these instances and writes it back to Amazon S3.

The master group instances serve a critical function in our Amazon EMR cluster. If a master node is lost, you lose the work in progress by the master *and* the core and task nodes to which it had delegated work.

Core group instance

Core group instance members run the map and reduce portions of our Job Flow, and store intermediate data to the Hadoop Distributed File System (HDFS) storage in our Amazon EMR cluster.

The master node manages the tasks and data delegated to the core and task nodes. Due to the HDFS storage aspects of core nodes, a loss of a core node will result in data loss and possible failure of the complete Job Flow.

Task group instance

The task group is optional. It can do some of the dirty computational work of the map and reduce jobs, but does not have HDFS storage of the data and intermediate results.

The lack of HDFS storage on these instances means the data needs to be transferred to these nodes by the master for the task group to do the work in the Job Flow.

Amazon EMR and the Hadoop Ecosystem

Amazon EMR uses Hadoop and its MapReduce framework at its core. Accordingly, many of the other core Apache Software Foundation projects that work with Hadoop also work with Amazon EMR. There are also many other AWS services that may be useful when you're running and monitoring Amazon EMR applications. For example:

Hive

Hive is a distributed data warehouse that allows you to create a Job Flow using a SQL-like language. Hive can be run from a script loaded in S3 or interactively inside of a running EMR instance.

Pig

Pig is a data flow language. The language is called *Pig Latin*. Pig scripts can be loaded into S3 and used to perform the data analysis in a Job Flow. Pig, like Hive, is one of the Job Flow types that can be run interactively inside of a running EMR instance.

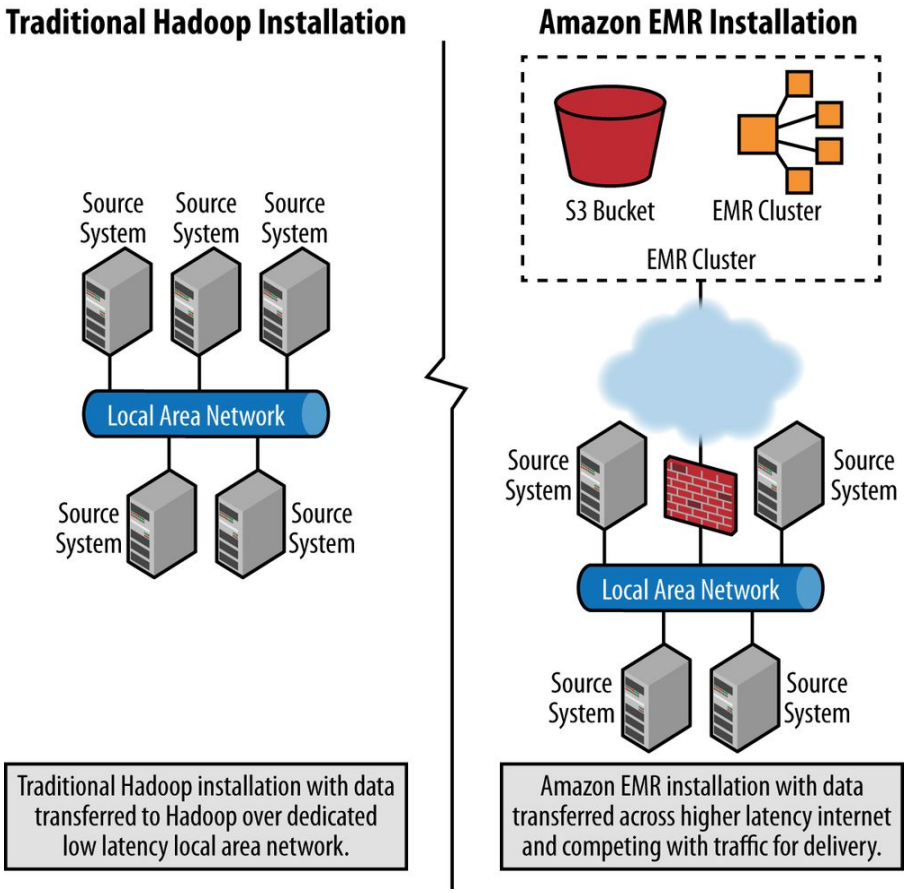
Amazon Cloudwatch

Cloudwatch allows you to monitor the health and progress of Job Flows. It also allows you to set alarms when metrics are outside of normal execution parameters.

Amazon Elastic MapReduce Versus Traditional Hadoop Installs

Amazon EMR uses S3 storage for the input and output of data sets to be processed and analyzed. In order to process data, you need to transport it to Amazon's cloud into S3 buckets.

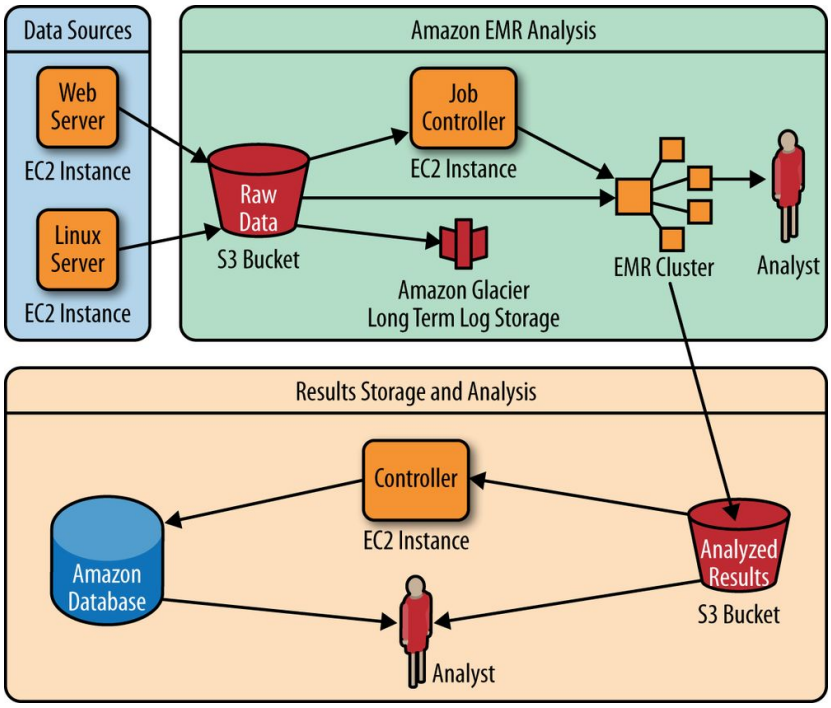
In the traditional Hadoop install, data transport between the current source locations and the Hadoop cluster may be collocated in the same data center on high-speed internal networks. This lowers the data transport barriers and the amount of time to get data into Hadoop for analysis.



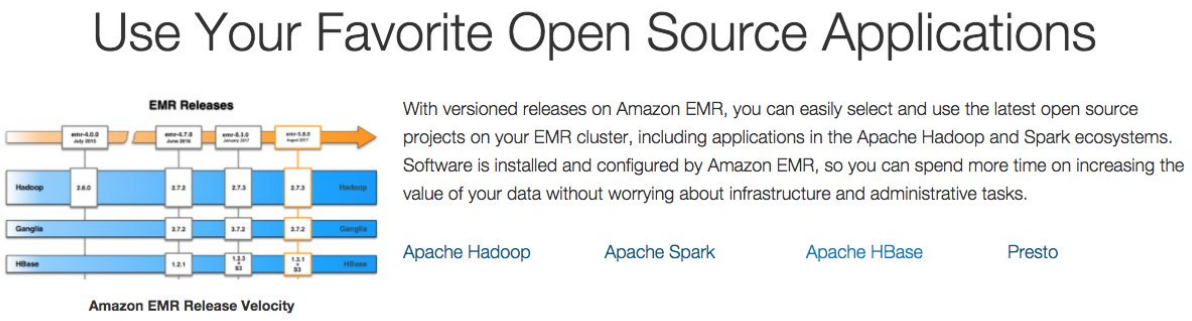
Note: Amazon has S3 Import and Export services.

Application Building Blocks

- ingest large volumes of log data,
- perform real-time and
- batch analysis, and ultimately produce results that can be shared with end users.



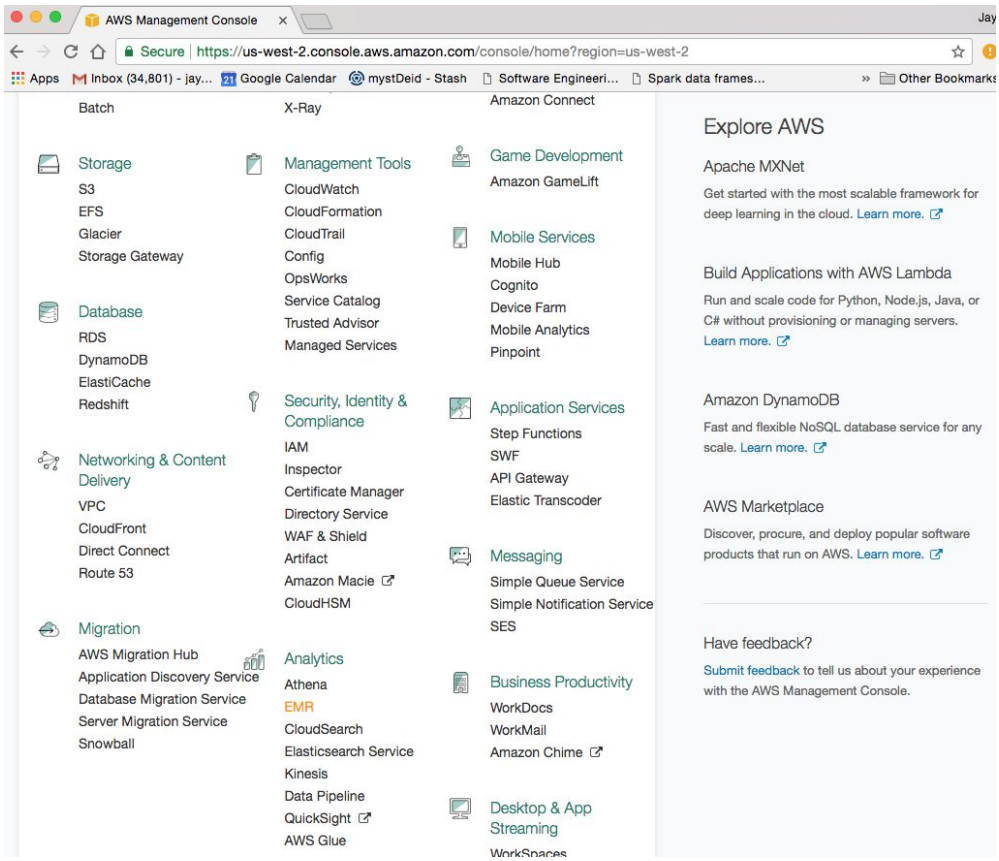
Application Integration



Creating and deploying a MapReduce App on Amazon EMR

Go to aws.amzon.com create an AWS account.

Log on to AWS. You should see the following services.



Amazon EMR Management Guide
http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-what-is-emr.html

Amazon Elastic MapReduce API
http://docs.aws.amazon.com/ElasticMapReduce/latest/API/Welcome.html

Amazon EMR Release Guide
http://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-hadoop-application.html

Basic steps for creating a job:

- Upload
- Create
- Monitor

Standard approach:

1. Upload data from S3 (cloud storage) into HDFS (Hadoop File System)
2. Can do MapReduce directly against S3, or you can upload your data from S3 into the HDFS file systems that's part of the EC2 (Elastic Cloud Compute) instances with the JBMs with the Hadoop daemons installed on them.

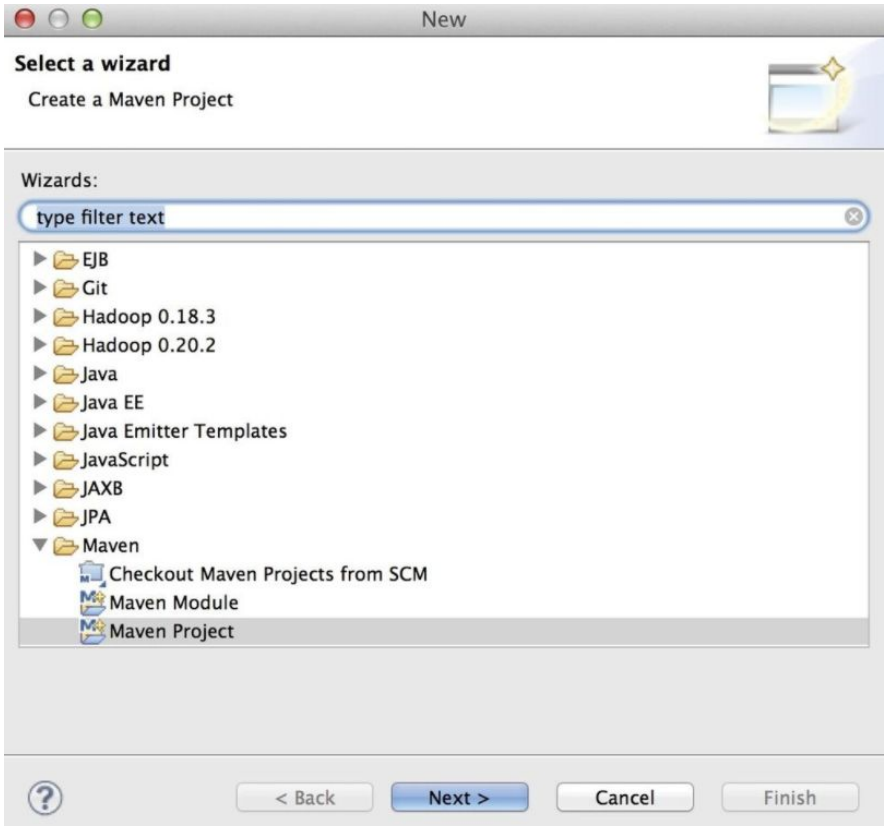
Building MapReduce Applications

This example uses Eclipse Java IDE and the Eclipse Maven plug-in, m2eclipse, to manage application dependencies.

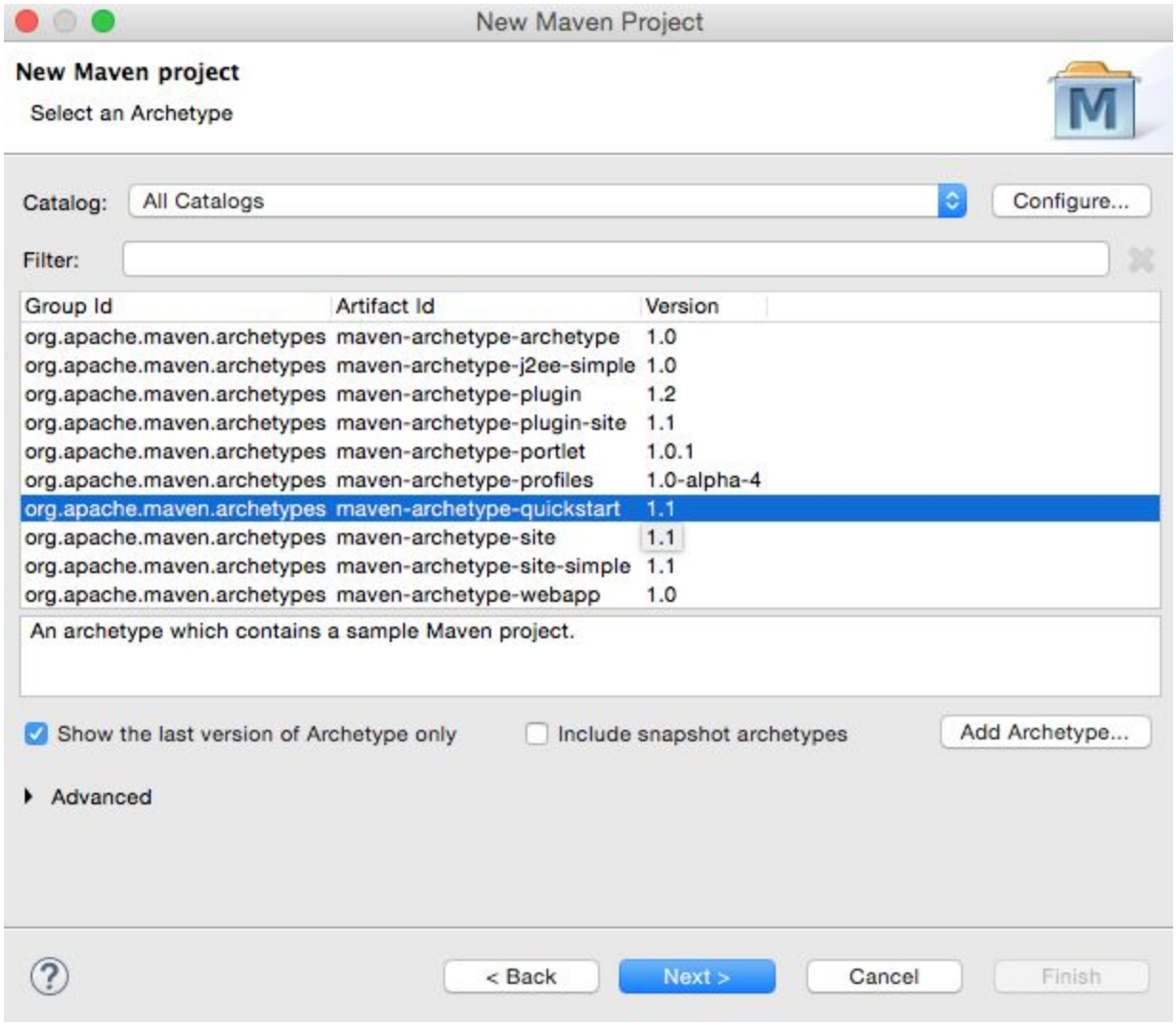
You can install the *m2eclipse plug-in* through the *Install New Software* option inside of Eclipse.

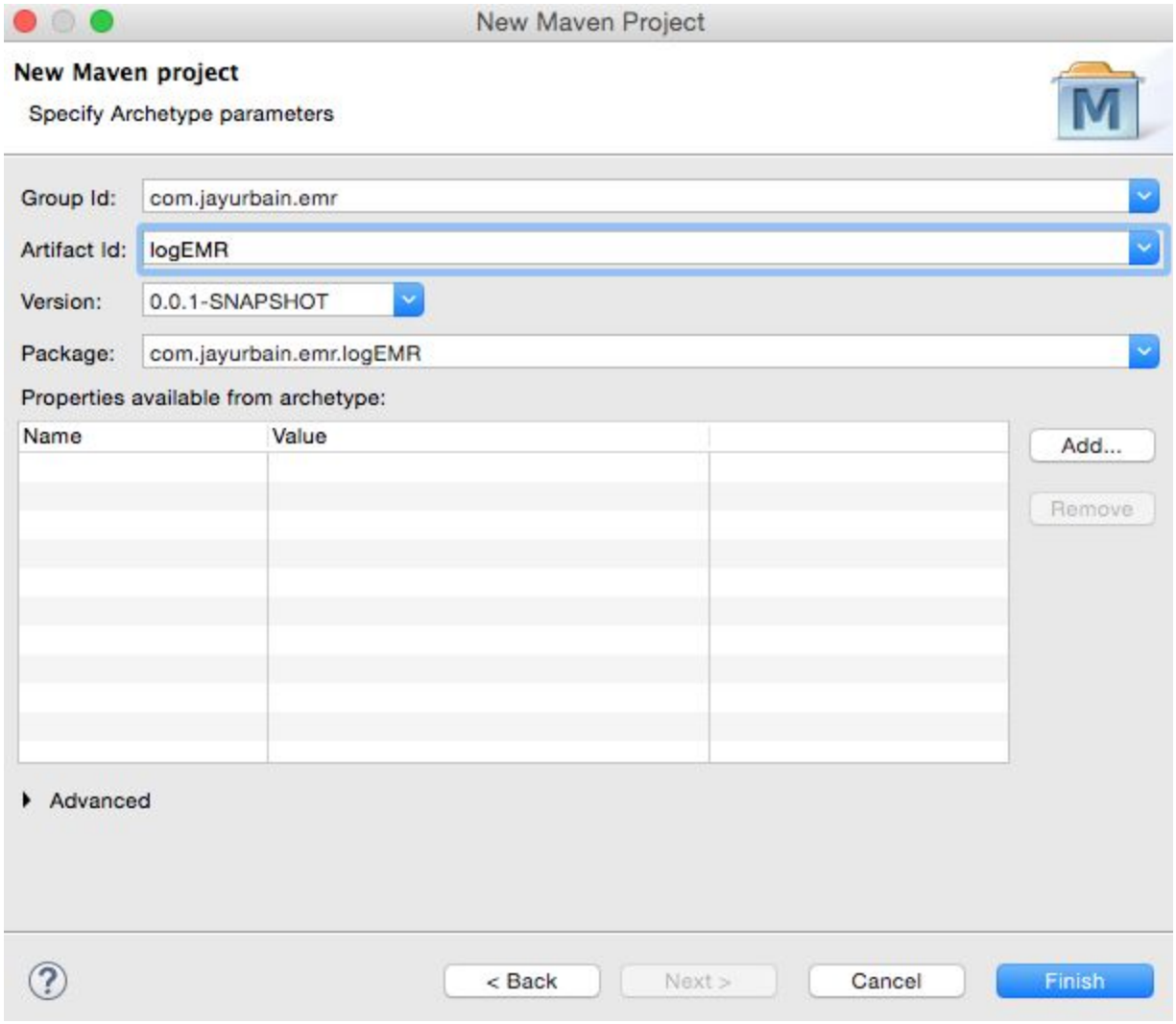
To include the dependencies needed to build the MapReduce applications:

Create a Maven project inside of Eclipse by selecting *File→New→Other*. The Maven project option should be available after you install the m2eclipse plug-in.



Select the program and project name of your application when going through the Eclipse New Project Wizard.


New Maven Project
New Maven project
Select an Archetype
Catalog:
All Catalogs
Configure...
Filter:
Group Id
Artifact Id
Version
org.apache.maven.archetypes maven-archetype-archetype 1.0
org.apache.maven.archetypes maven-archetype-j2ee-simple 1.0
org.apache.maven.archetypes maven-archetype-plugin 1.2
org.apache.maven.archetypes maven-archetype-plugin-site 1.1
org.apache.maven.archetypes maven-archetype-portlet 1.0.1
org.apache.maven.archetypes maven-archetype-profiles 1.0-alpha-4
org.apache.maven.archetypes maven-archetype-quickstart 1.1
org.apache.maven.archetypes maven-archetype-site 1.1
org.apache.maven.archetypes maven-archetype-site-simple 1.1
org.apache.maven.archetypes maven-archetype-webapp 1.0
An archetype which contains a sample Maven project.
Show the last version of Archetype only
Include snapshot archetypes
Add Archetype...
Advanced
< Back
Next >
Cancel
Finish



FYI: Maven naming conventions:
https://maven.apache.org/guides/mini/guide-naming-conventions.html

After the project is created, the Hadoop dependencies will need to be added to the project so the application can make use of the Hadoop base classes, types, and methods. You can add the Hadoop core dependencies by opening the *pom.xml* file that is in the root of the project.

Add the following to your project pom.xml text

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.jayurbain.emr</groupId>
  <artifactId>logEMR</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>com.jayurbain.emr</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <build>
                        <plugins>
                                <plugin>
                                        <groupId>org.apache.maven.plugins</groupId>
                                        <artifactId>maven-shade-plugin</artifactId>
                                        <executions>
                                                <execution>
                                                        <phase>package</phase>
                                                        <goals>
                                                                <goal>shade</goal>
                                                        </goals>
                                                </execution>
                                        </executions>
                                        <configuration>
                                                <finalName>uber-${artifactId}-${version}</finalName>
                                        </configuration>
                                </plugin>
                        </plugins>
          </build>

  <dependencies>
          <!--  deid -->
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
    <dependency>
                        <groupId>org.apache.hadoop</groupId>
                        <artifactId>hadoop-client</artifactId>
                        <version>2.2.0</version>
          </dependency>
  </dependencies>
</project>
```

In your application:

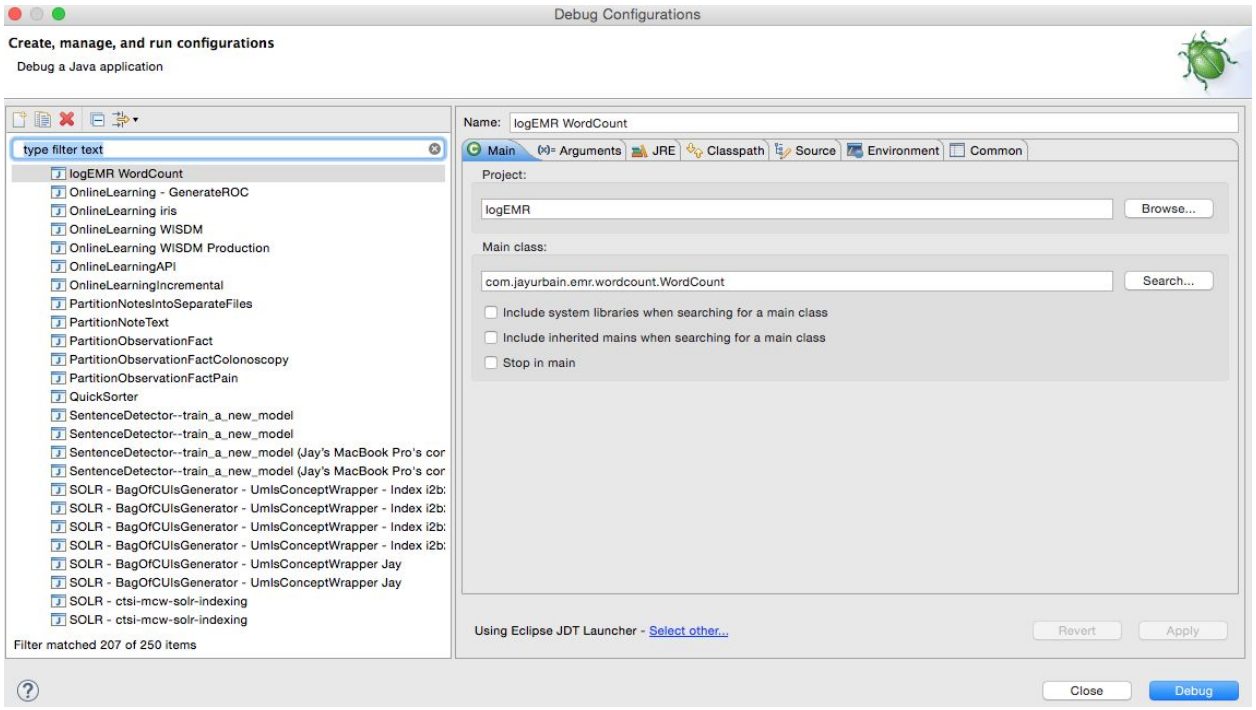
Create a package *com.jayurbain.emr*

Create a new Java class file *WordCount.java* as follows:

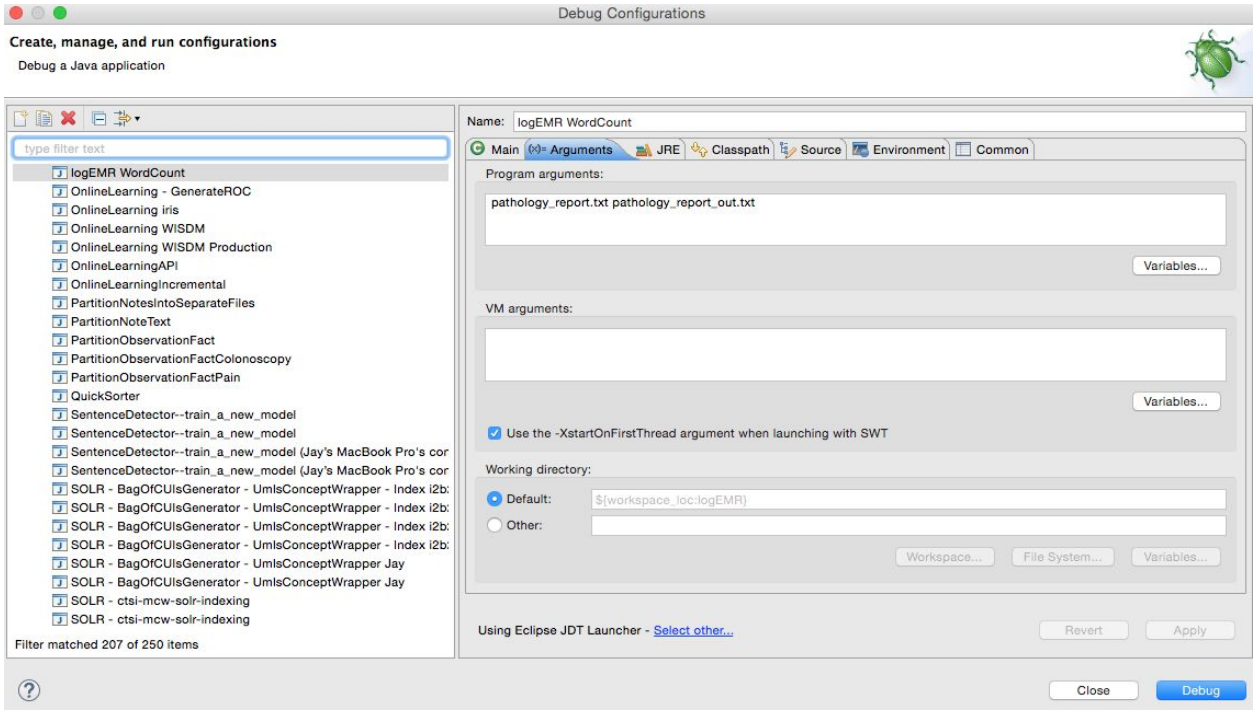
```
//////////////////
package com.jayurbain.emr.wordcount;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

public class WordCount {

  public static class Map extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
      String line = value.toString();
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens()) {
        word.set(tokenizer.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class Reduce extends Reducer<Text, IntWritable, Text, IntWritable> {
```

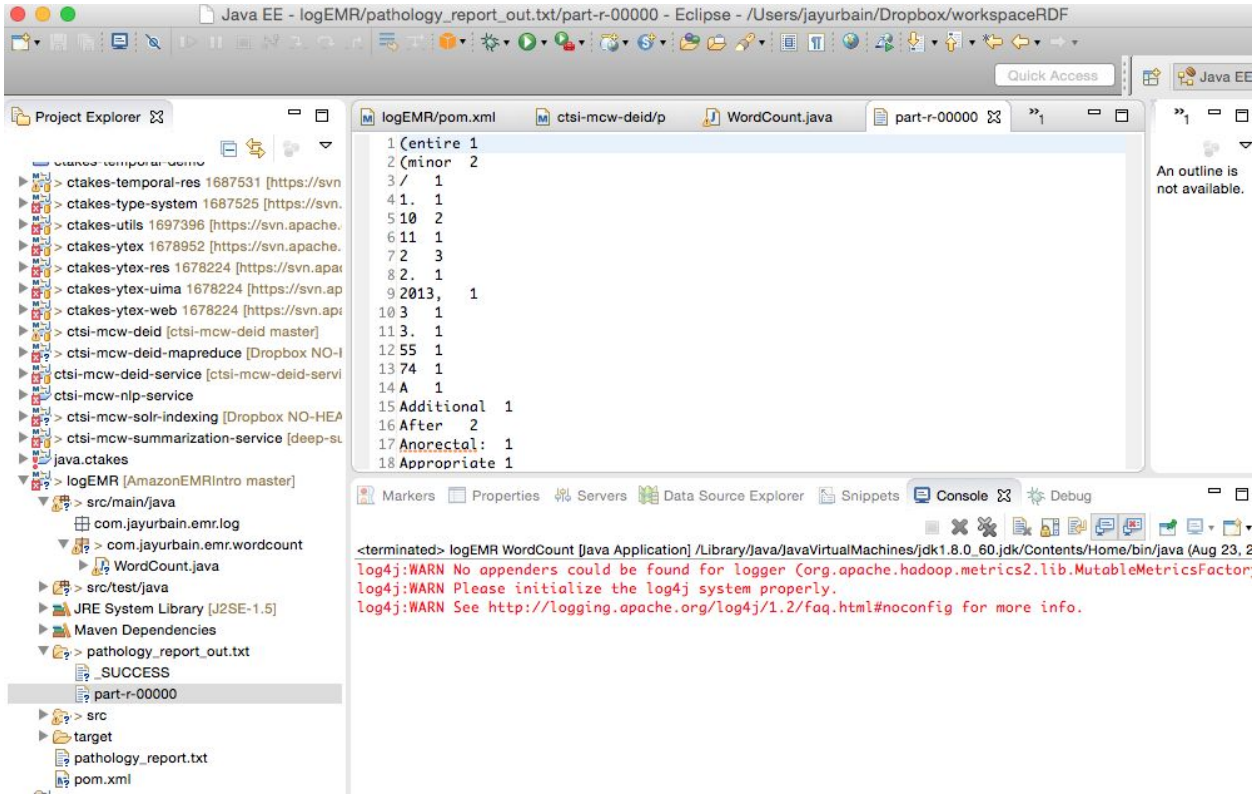
```
    public void reduce(Text key, Iterable<IntWritable> values, Context context)
      throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      context.write(key, new IntWritable(sum));
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();

    Job job = new Job(conf, "wordcount");

    job.setJarByClass(WordCount.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);

    job.setMapperClass(Map.class);
    job.setReducerClass(Reduce.class);

    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);
  }

}
///////////////
```

You can run the application locally first to test the application. Parameters: input text file, and output text file.

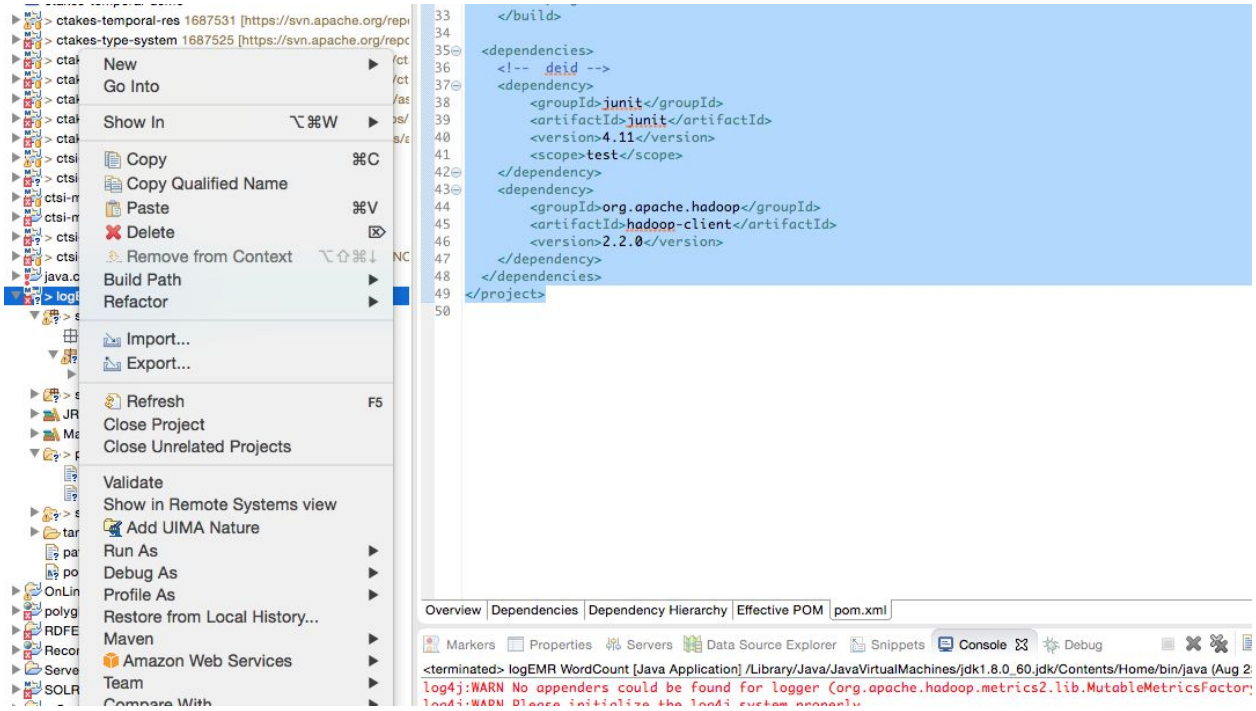


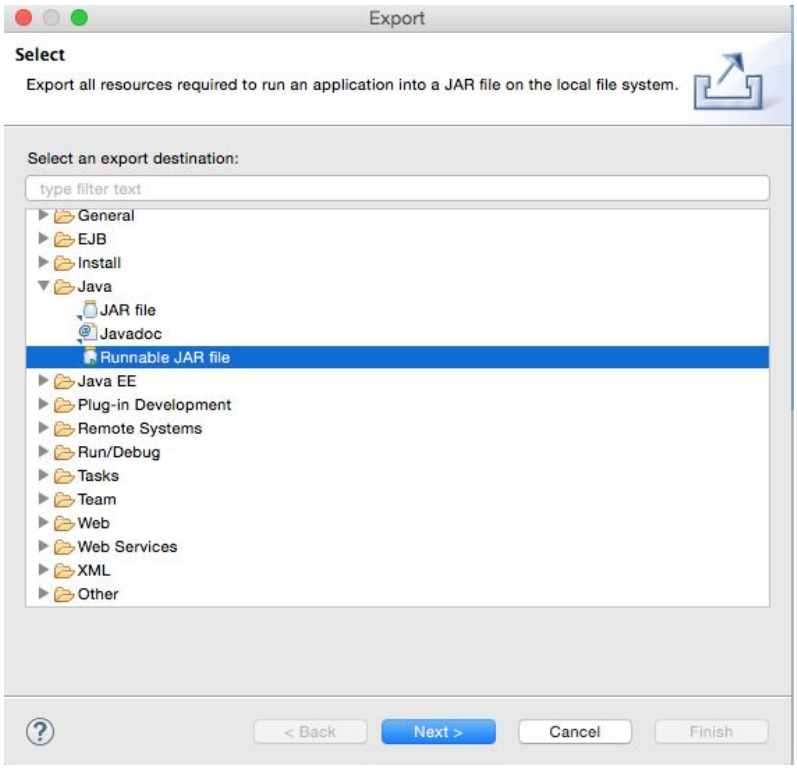
Select arguments.



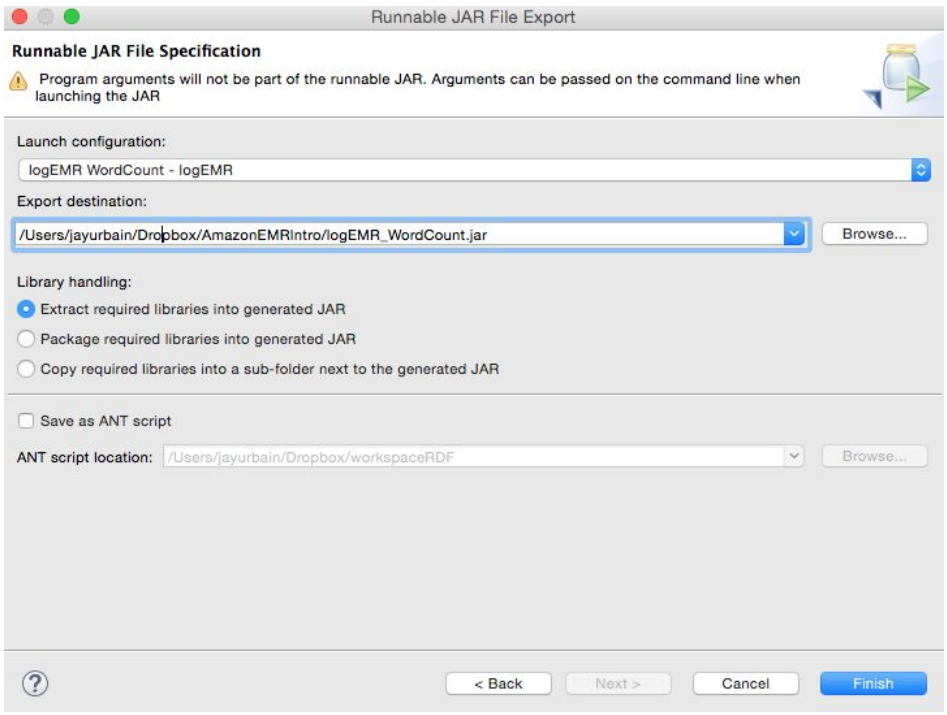

Java EE - logEMR/pathology_report_out.txt/part-r-00000 - Eclipse - /Users/jayurbain/Dropbox/workspaceRDF
Quick Access
Java EE
Project Explorer
logEMR/pom.xml
ctsi-mcw-deid/p
WordCount.java
part-r-00000
An outline is not available.
(entire 1
(minor 2
/ 1
1. 1
10 2
11 1
2 3
2. 1
2013, 1
3 1
3. 1
55 1
74 1
A 1
Additional 1
After 2
Anorectal: 1
Appropriate 1
ctakes-temporal-res 1687531 [https://svn
ctakes-type-system 1687525 [https://svn.
ctakes-utils 1697396 [https://svn.apache.
ctakes-ytex 1678952 [https://svn.apache.
ctakes-ytex-res 1678224 [https://svn.apa
ctakes-ytex-uima 1678224 [https://svn.ap
ctakes-ytex-web 1678224 [https://svn.apa
ctsi-mcw-deid [ctsi-mcw-deid master]
ctsi-mcw-deid-mapreduce [Dropbox NO-I
ctsi-mcw-deid-service [ctsi-mcw-deid-servi
ctsi-mcw-nlp-service
ctsi-mcw-solr-indexing [Dropbox NO-HEA
ctsi-mcw-summarization-service [deep-su
java.ctakes
logEMR [AmazonEMRIntro master]
src/main/java
com.jayurbain.emr.log
com.jayurbain.emr.wordcount
WordCount.java
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
pathology_report_out.txt
_SUCCESS
part-r-00000
src
target
pathology_report.txt
pom.xml
Markers
Properties
Servers
Data Source Explorer
Snippets
Console
Debug
<terminated> logEMR WordCount [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java (Aug 23, 2
log4j:WARN No appenders could be found for logger (org.apache.hadoop.metrics2.lib.MutableMetricsFactor
log4j:WARN Please initialize the log4j system properly.
log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noconfig for more info.

Export as executable jar file:

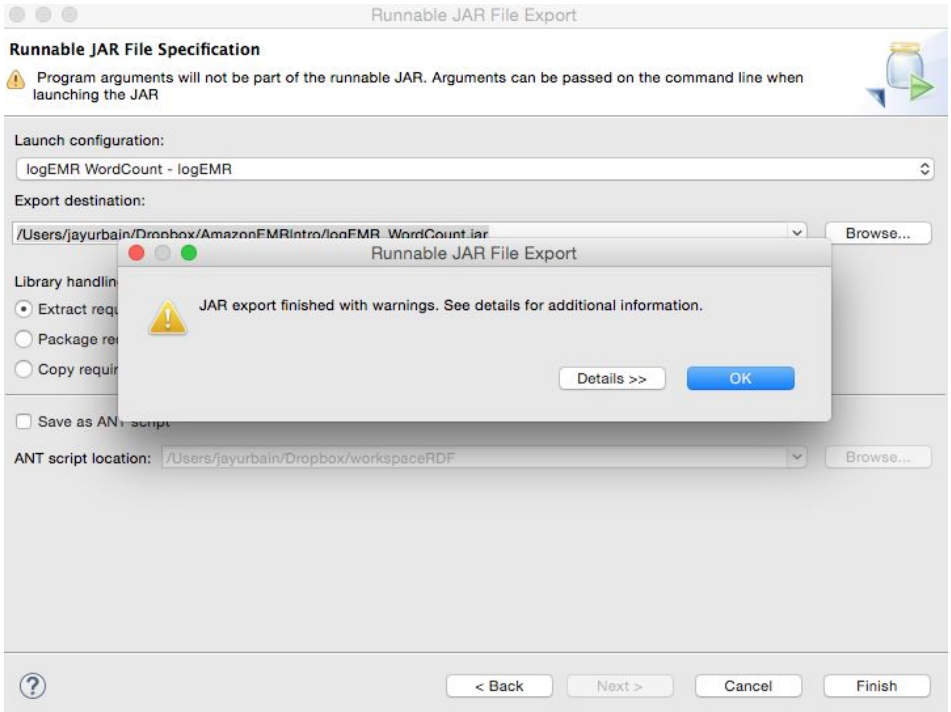




Select launch configuration, and export destination.



Select Ok.



Just hit Ok. Only duplicate packaging warnings.

To run the app on a Hadoop with local file system. (See end of this file for installing Hadoop locally).
$ hadoop jar logEMR_WordCount patholgoy_report.txt pathology_report_out.txt

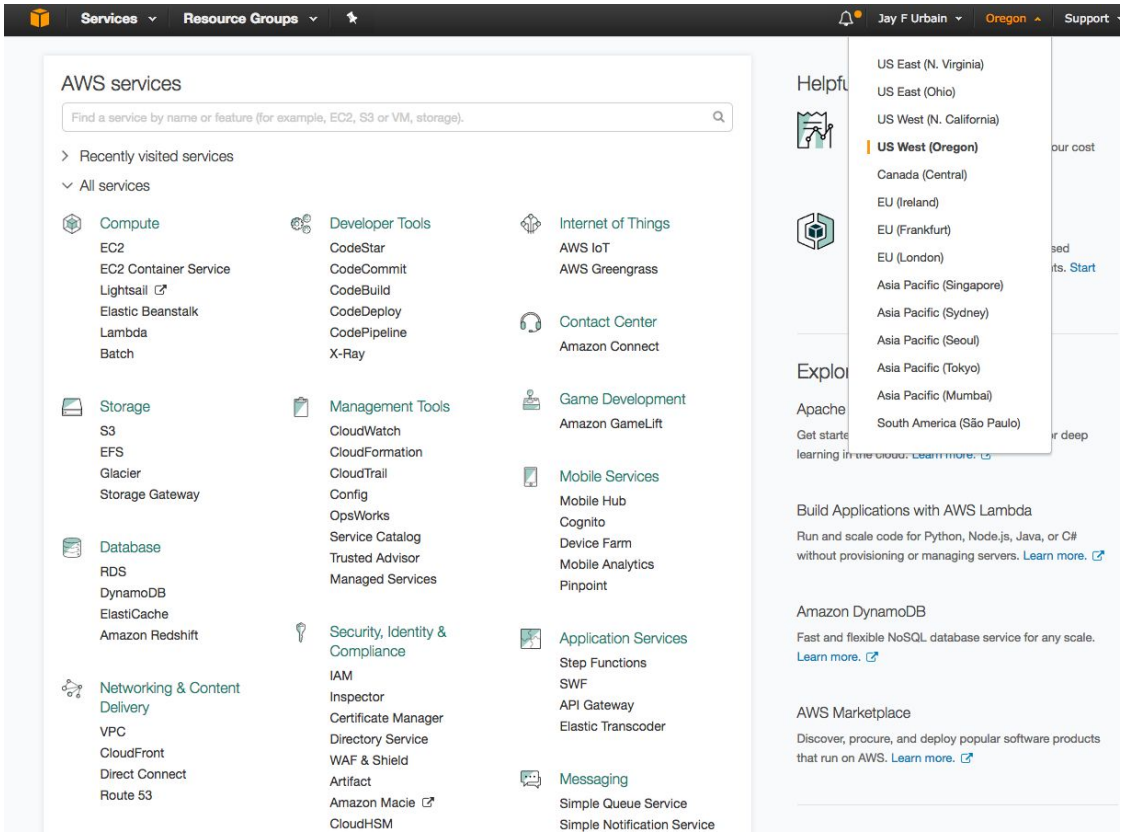
Run (for hdfs file system):
$hadoop jar logEMR_WordCount hdfs:///xxx//patholgoy_report.txt
hdfs:///xxx//pathology_report_out.txt

In either case, result file will be in pathology_report_out.txt/part-r-00000 (or something similar).

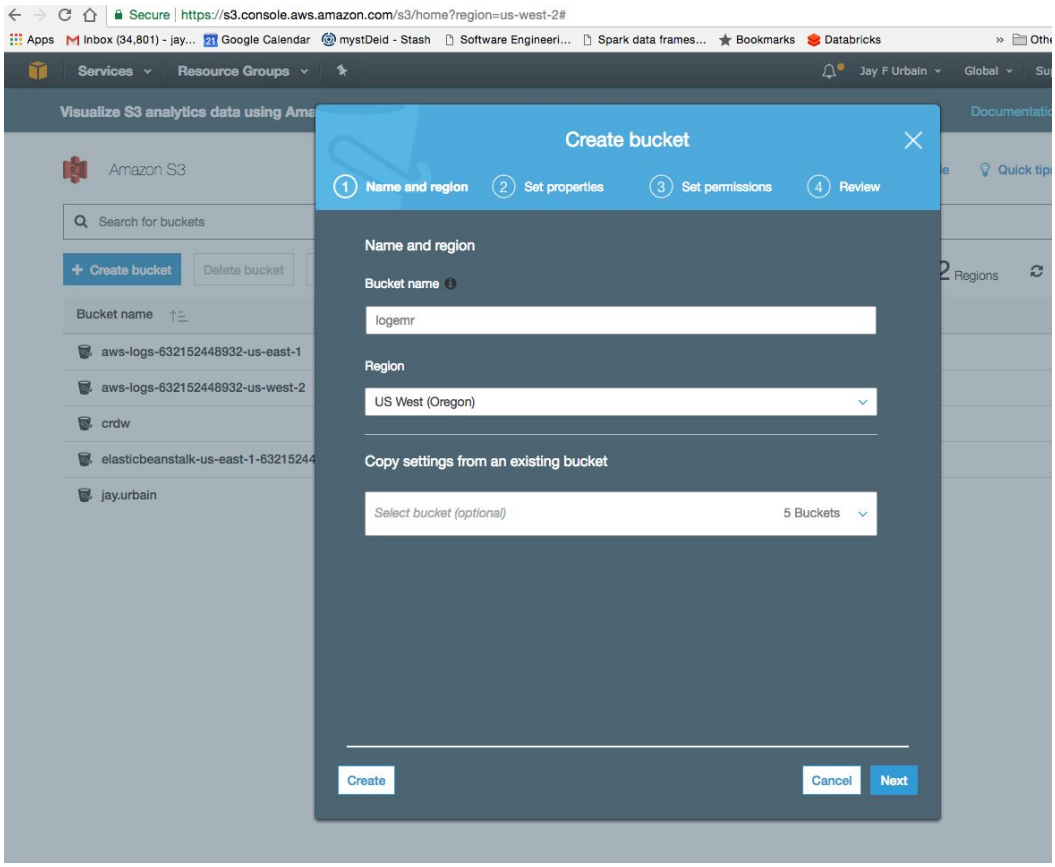
To run on AmazonEMR:

Sign into AWS Console.

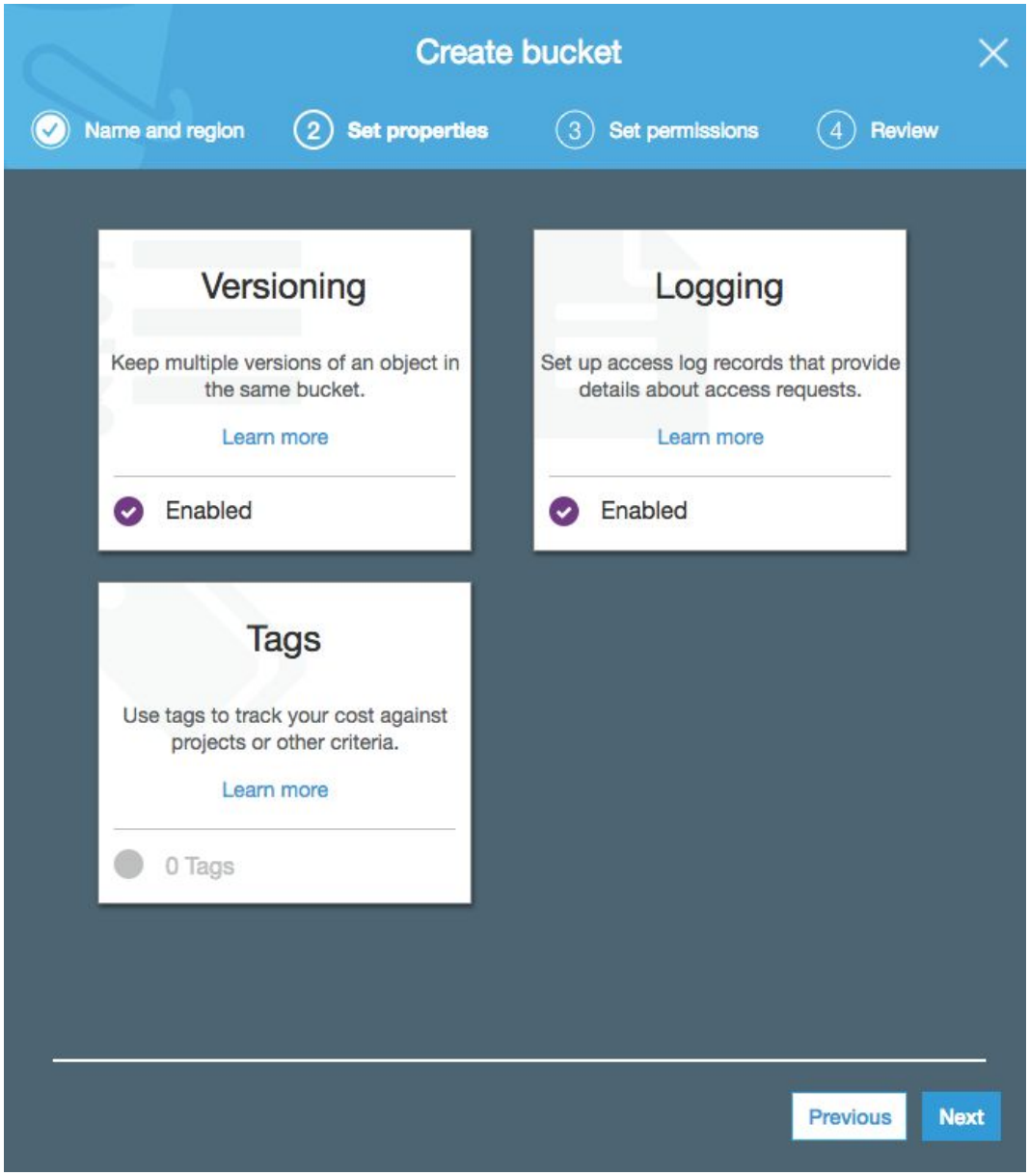
Select S3. Take note of the region. In this case **US West (Oregon)**



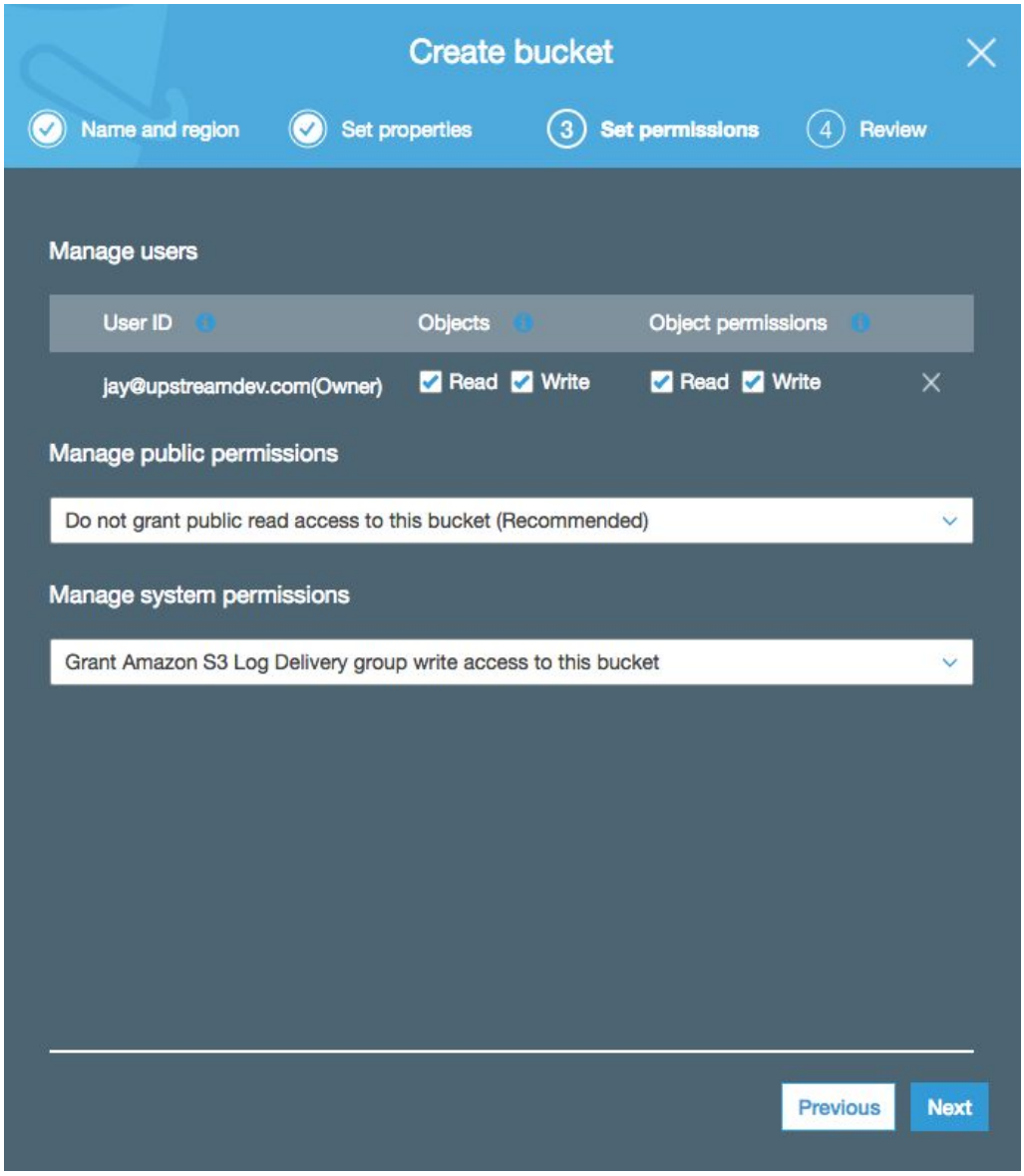
Create an S3 bucket.



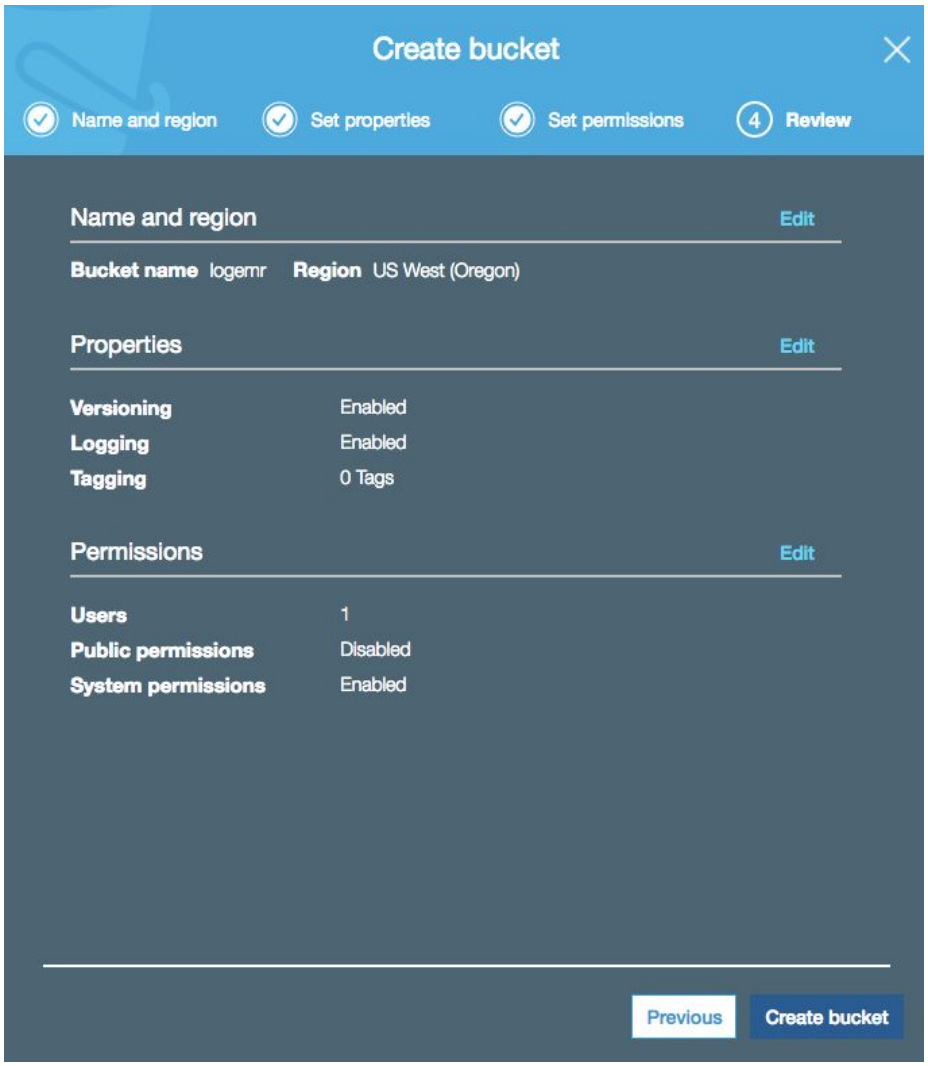
You can skip the following. Or set properties. They can be changed later.



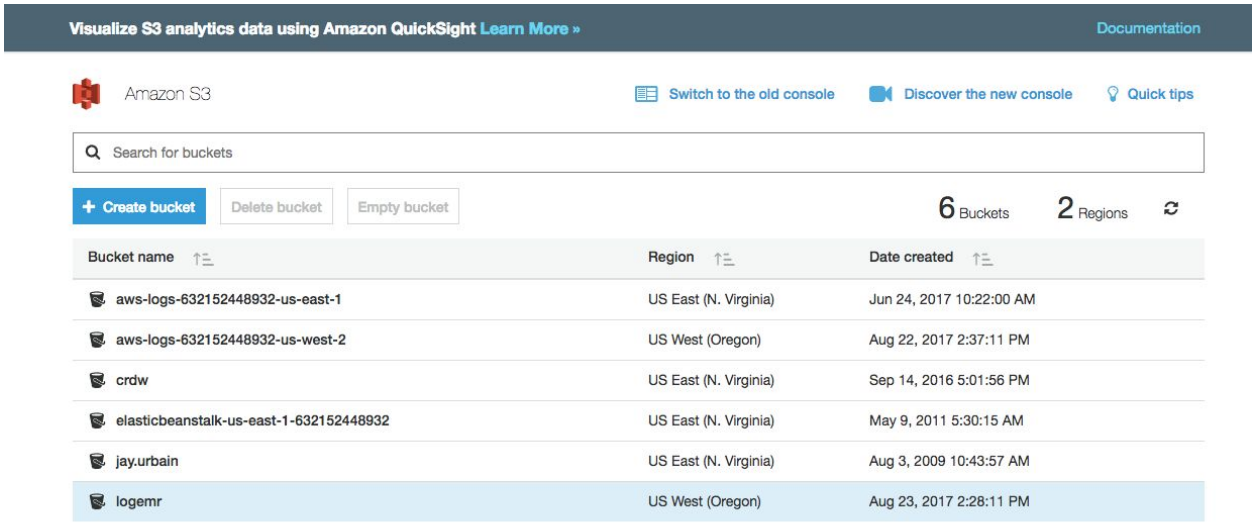
Select next.


Create bucket
Name and region
Set properties
3
Set permissions
4
Review
Manage users
User ID
Objects
Object permissions
jay@upstreamdev.com(Owner)
Read
Write
Read
Write
Manage public permissions
Do not grant public read access to this bucket (Recommended)
Manage system permissions
Grant Amazon S3 Log Delivery group write access to this bucket
Previous
Next

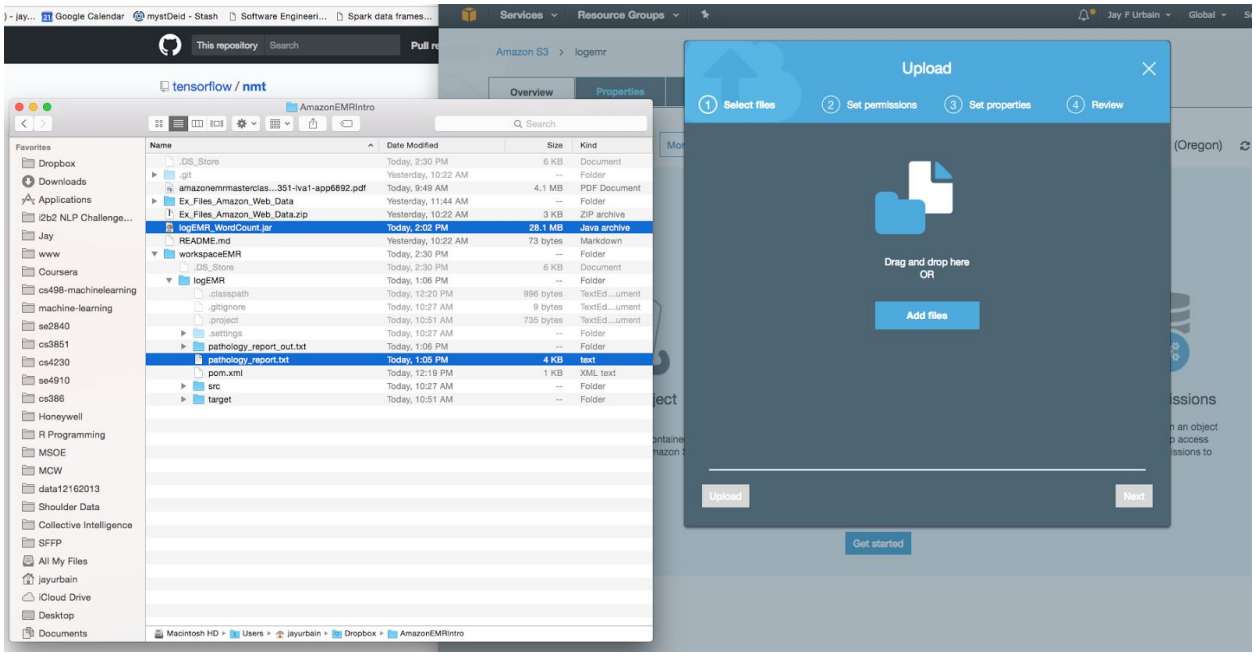
Select Create bucket.

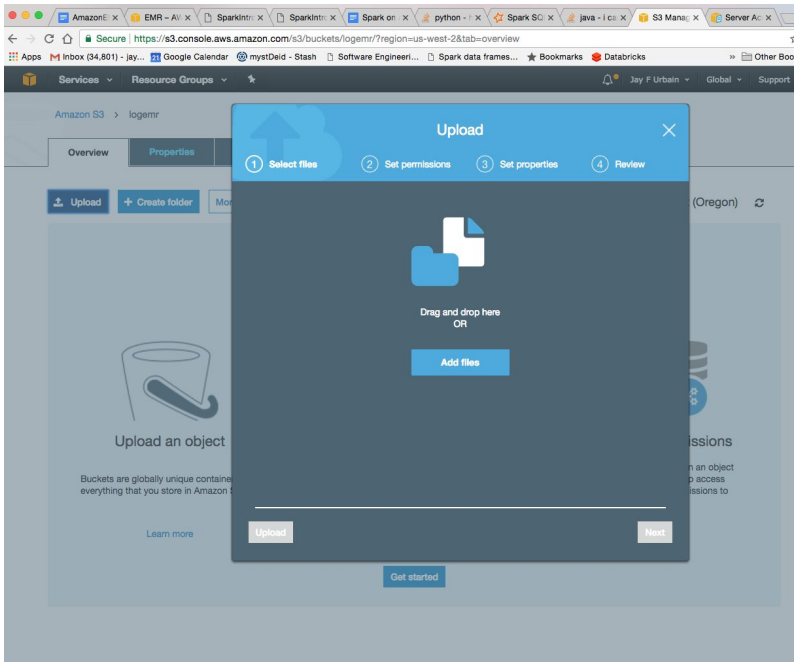


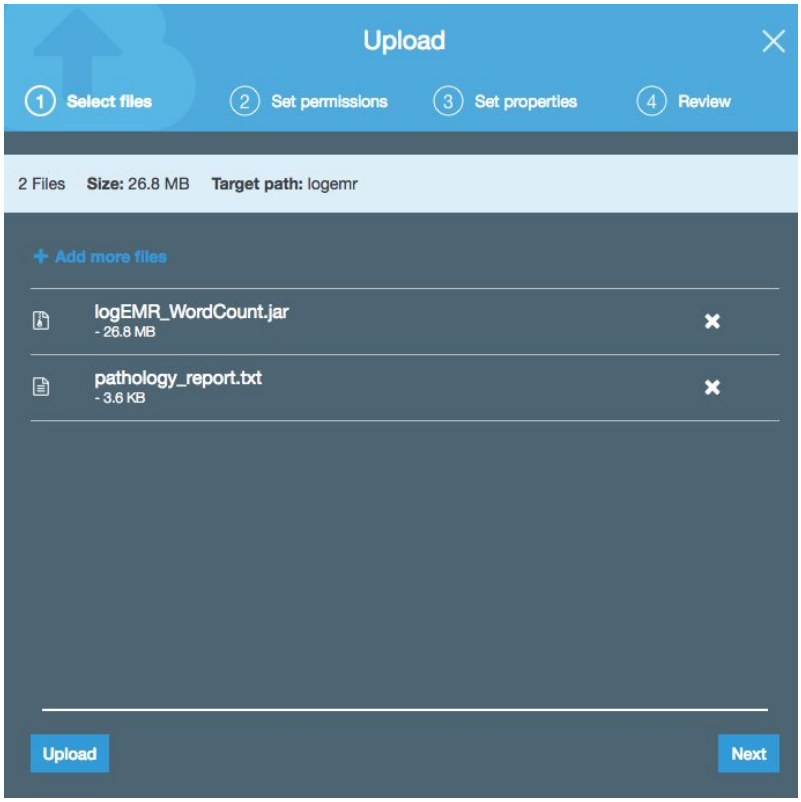
You should see the following.

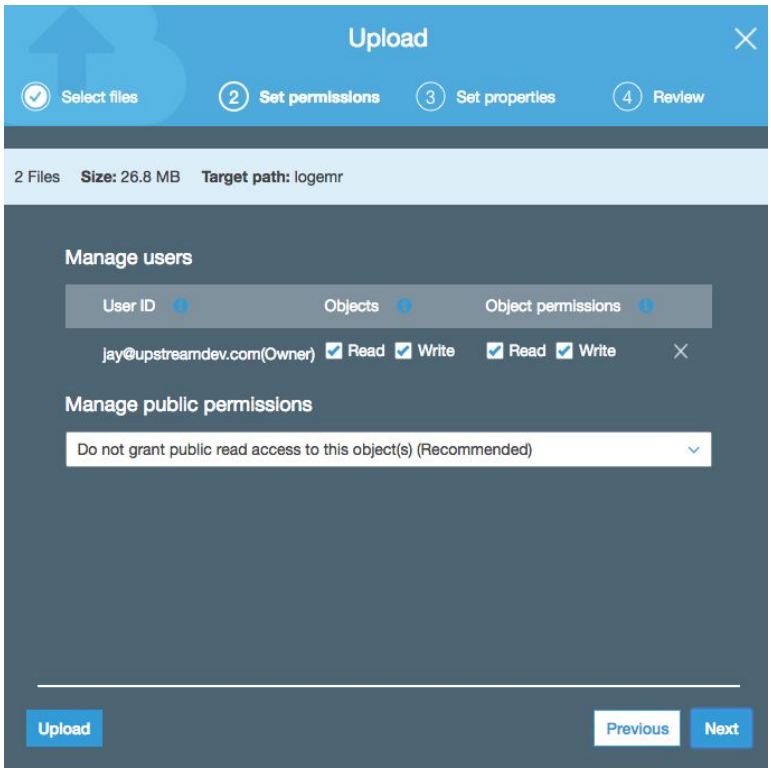


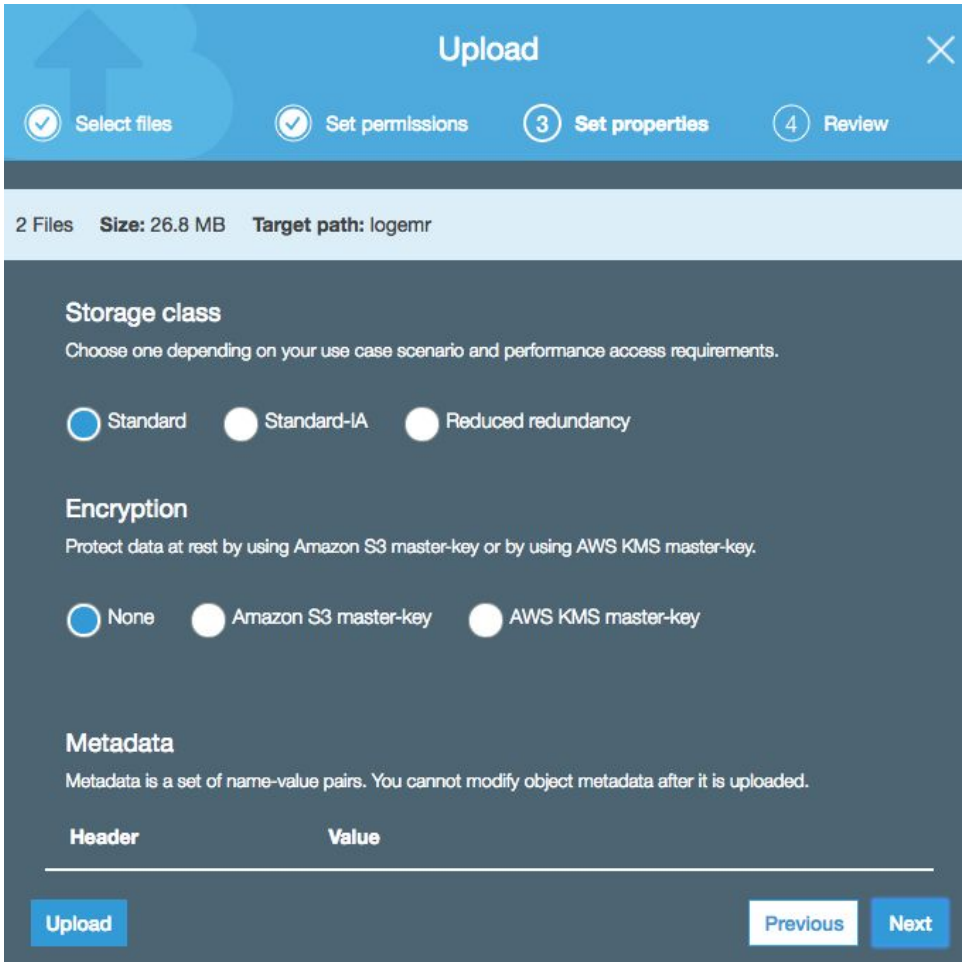
Select your logerr bucket and upload your jar and *text input* file.

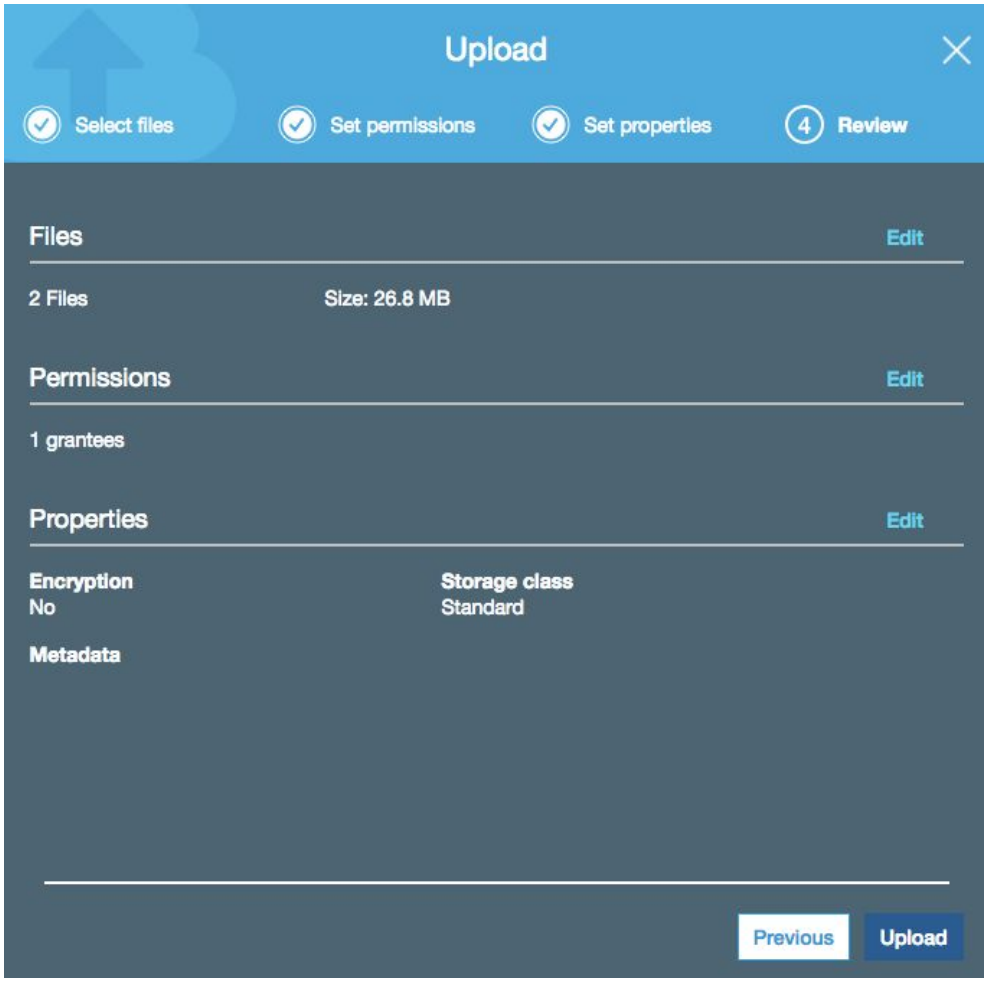



Upload
Select files
Set permissions
Set properties
Review
Drag and drop here
OR
Add files
Upload
Next
Upload an object
Get started


Upload
Select files
Set permissions
Set properties
Review
2 Files
Size: 26.8 MB
Target path: logemr
Add more files
logEMR_WordCount.jar
- 26.8 MB
pathology_report.txt
- 3.6 KB
Upload
Next

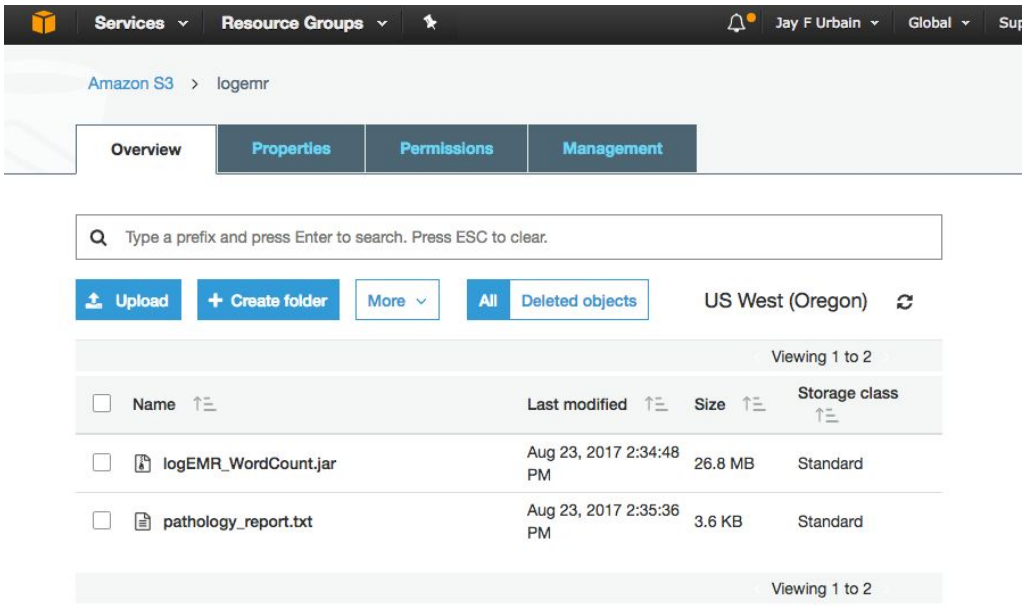

Upload
Select files
2
Set permissions
3
Set properties
4
Review
2 Files
Size: 26.8 MB
Target path: logemr
Manage users
User ID
Objects
Object permissions
jay@upstreamdev.com(Owner)
Read
Write
Read
Write
Manage public permissions
Do not grant public read access to this object(s) (Recommended)
Upload
Previous
Next


Upload
Select files
Set permissions
3
Set properties
4
Review
2 Files
Size: 26.8 MB
Target path: logemr
Storage class
Choose one depending on your use case scenario and performance access requirements.
Standard
Standard-IA
Reduced redundancy
Encryption
Protect data at rest by using Amazon S3 master-key or by using AWS KMS master-key.
None
Amazon S3 master-key
AWS KMS master-key
Metadata
Metadata is a set of name-value pairs. You cannot modify object metadata after it is uploaded.
Header
Value
Upload
Previous
Next



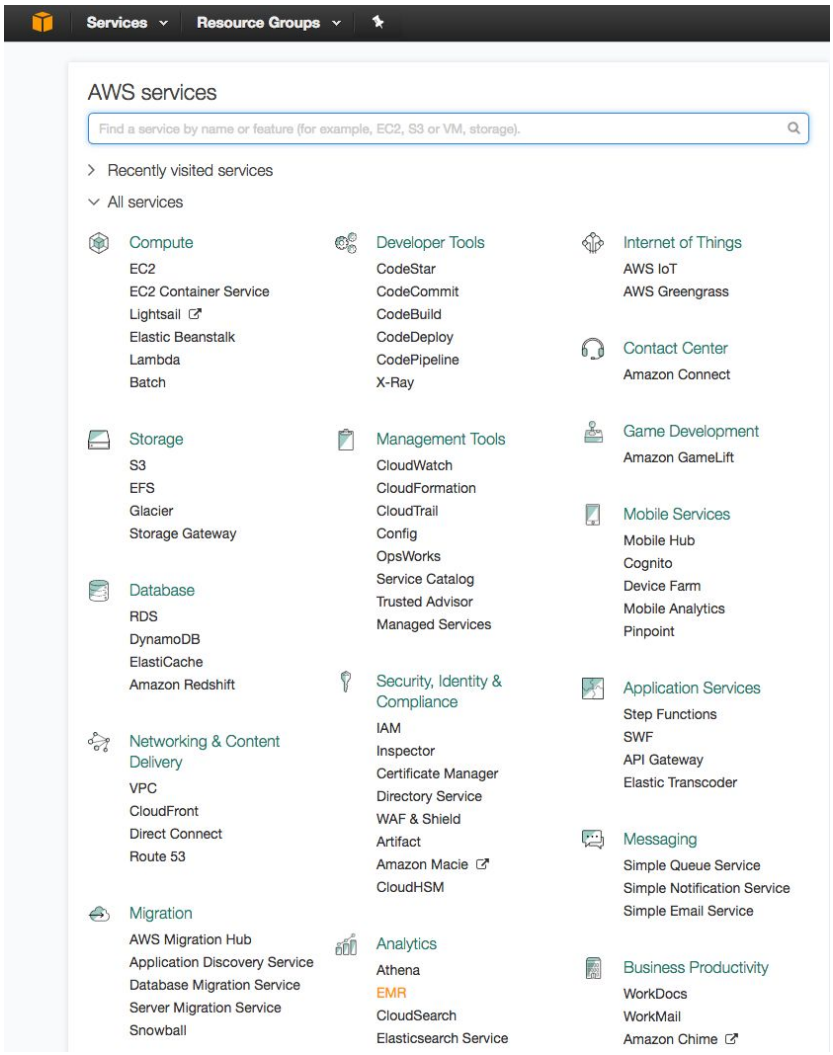
Sure asks a lot of questions. Select Upload.

Be patient.

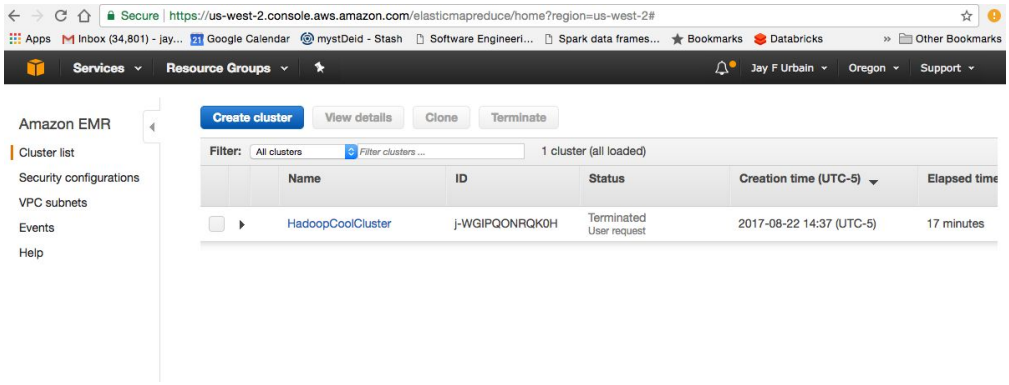

Services
Resource Groups
Jay F Urbain
Global
Amazon S3 > logemr
Overview
Properties
Permissions
Management
Type a prefix and press Enter to search. Press ESC to clear.
Upload
+ Create folder
More
All
Deleted objects
US West (Oregon)
Viewing 1 to 2
Name
Last modified
Size
Storage class
logEMR_WordCount.jar
Aug 23, 2017 2:34:48 PM
26.8 MB
Standard
pathology_report.txt
Aug 23, 2017 2:35:36 PM
3.6 KB
Standard
Viewing 1 to 2

tada!

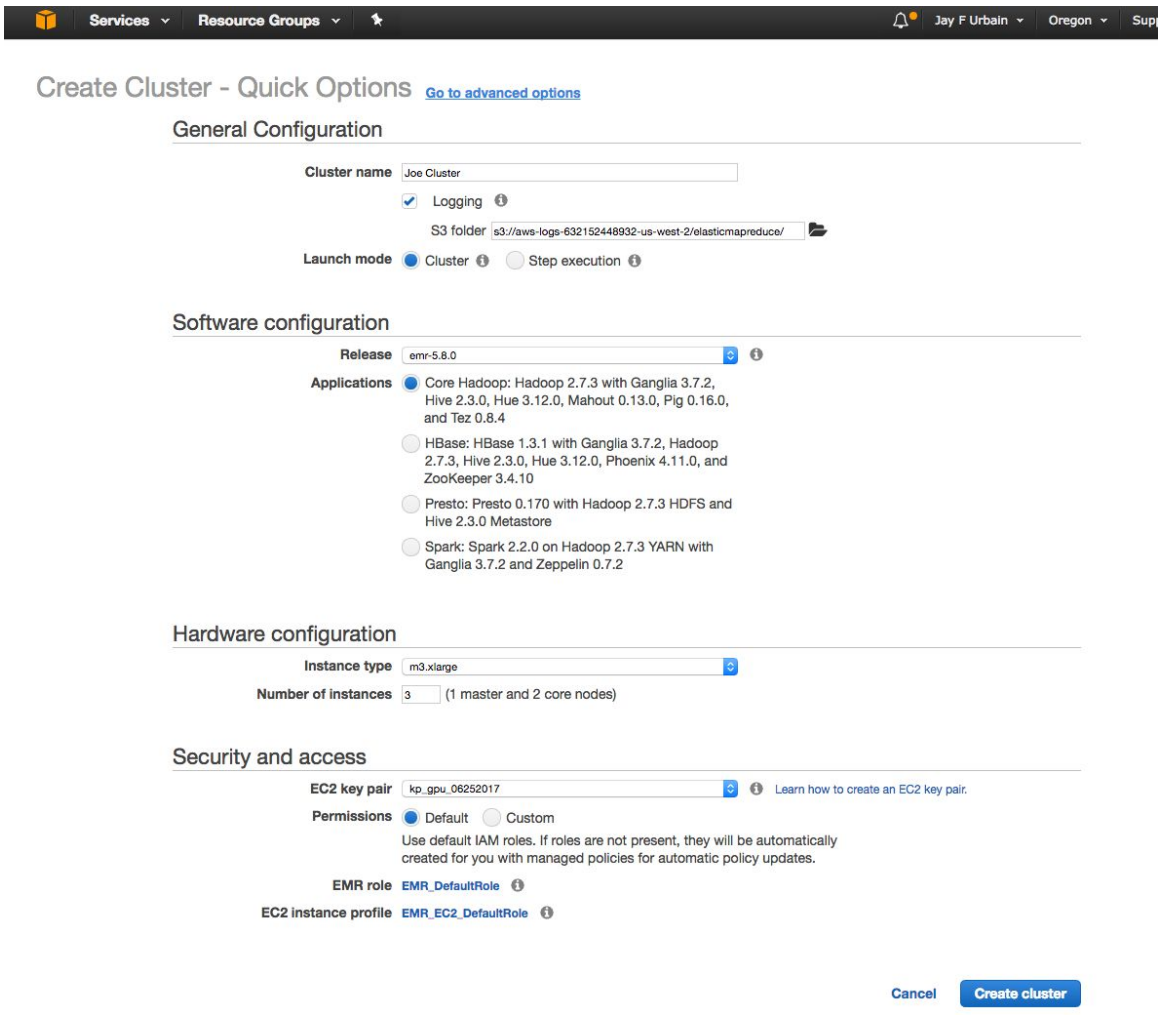
Select EMR from AWS Services



Select Create Cluster:

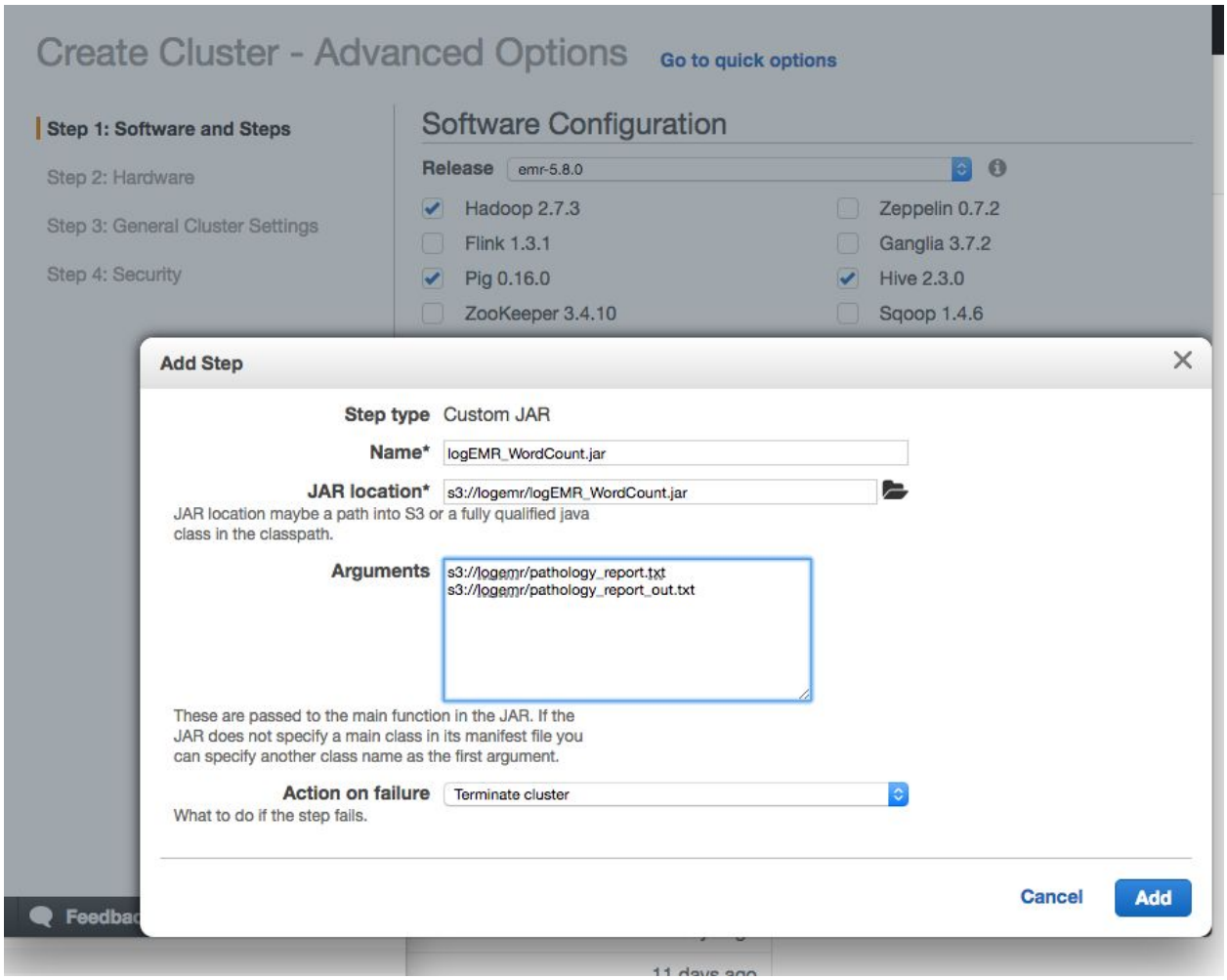


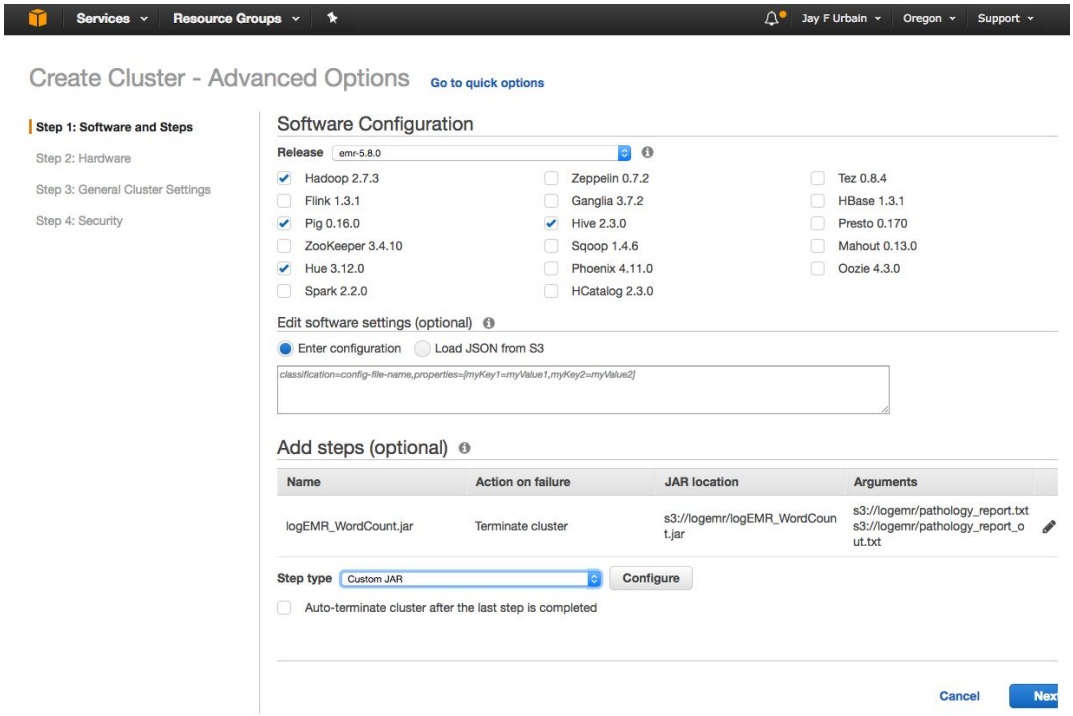
Make the following selections (do NOT select Create cluster until you scroll down):



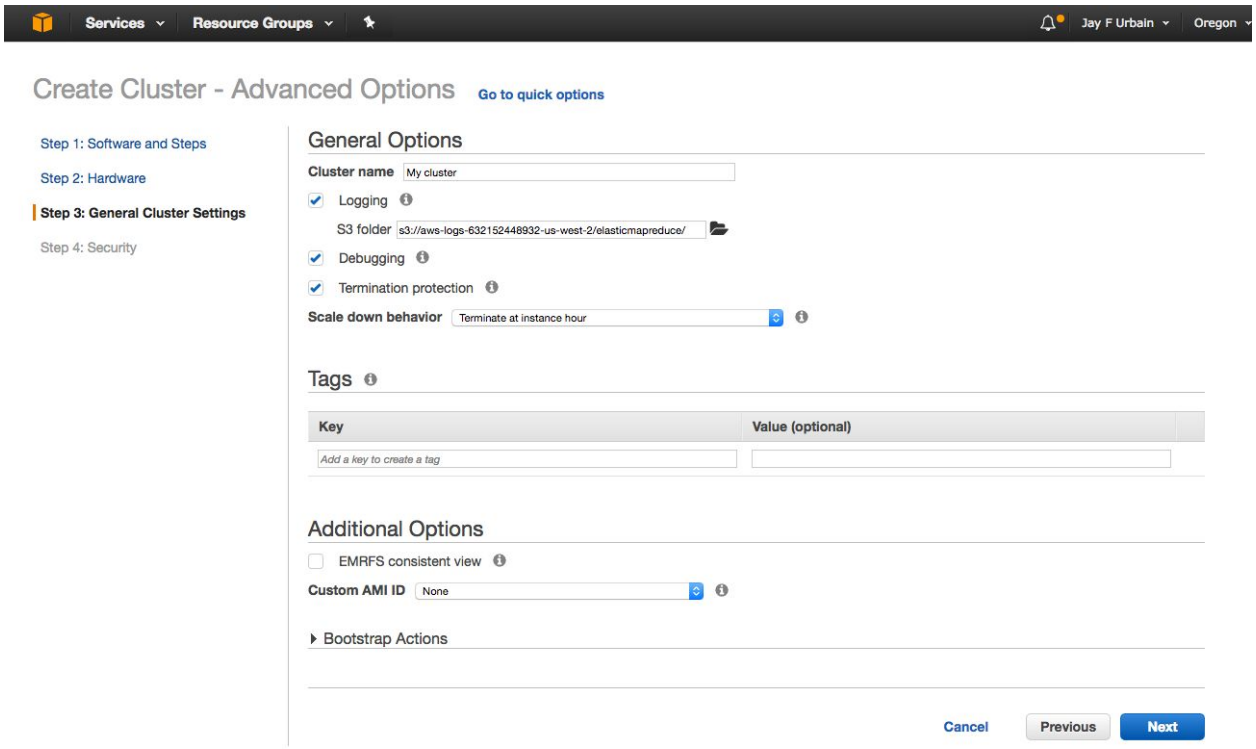
Select Go to advanced options

Create a custom JAR



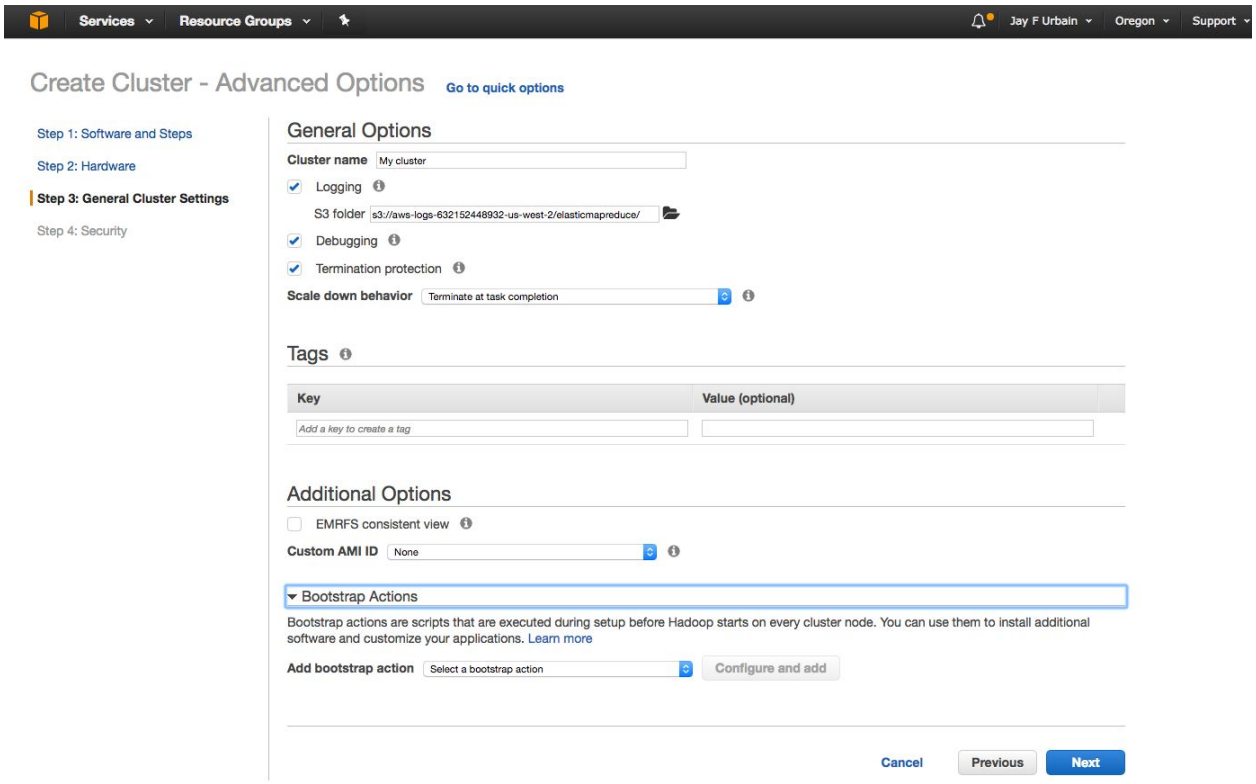

Services
Resource Groups
Jay F Urbain
Oregon
Support
Create Cluster - Advanced Options
Go to quick options
Step 1: Software and Steps
Step 2: Hardware
Step 3: General Cluster Settings
Step 4: Security
Software Configuration
Release
emr-5.8.0
Hadoop 2.7.3
Flink 1.3.1
Pig 0.16.0
ZooKeeper 3.4.10
Hue 3.12.0
Spark 2.2.0
Zeppelin 0.7.2
Ganglia 3.7.2
Hive 2.3.0
Sqoop 1.4.6
Phoenix 4.11.0
HCatalog 2.3.0
Tez 0.8.4
HBase 1.3.1
Presto 0.170
Mahout 0.13.0
Oozie 4.3.0
Edit software settings (optional)
Enter configuration
Load JSON from S3
classification=config-file-name,properties=[myKey1=myValue1,myKey2=myValue2]
Add steps (optional)
Name
Action on failure
JAR location
Arguments
logEMR_WordCount.jar
Terminate cluster
s3://logemr/logEMR_WordCount.jar
s3://logemr/pathology_report.txt s3://logemr/pathology_report_out.txt
Step type
Custom JAR
Configure
Auto-terminate cluster after the last step is completed
Cancel
Next

Select next.

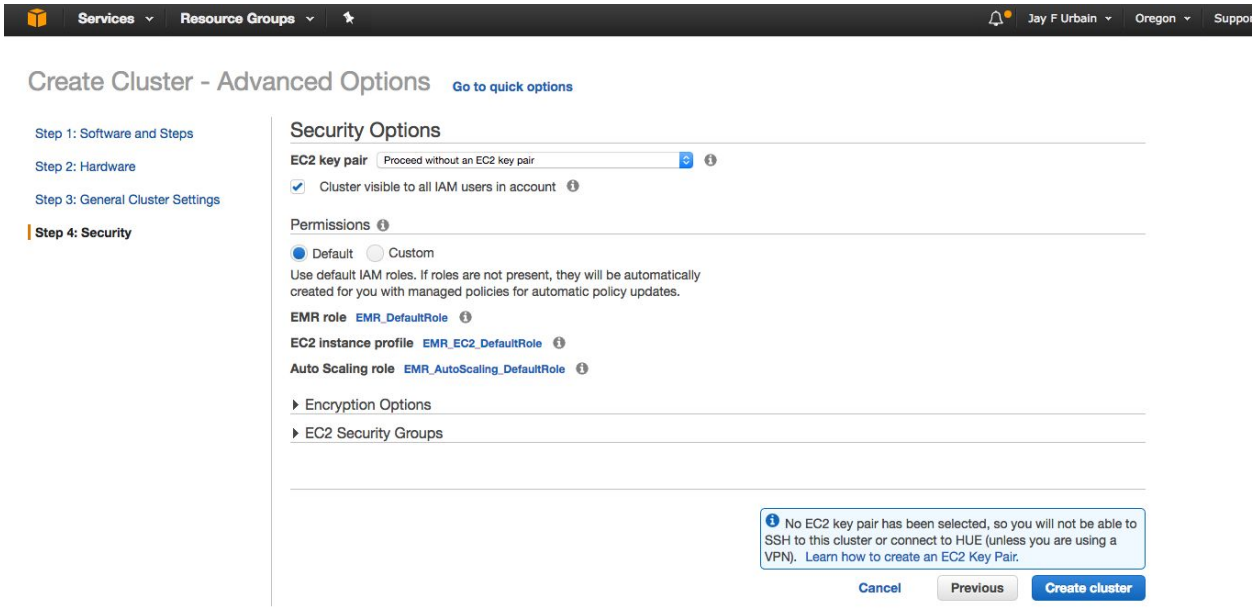

Services
Resource Groups
Jay F Urbain
Oregon
Create Cluster - Advanced Options
Go to quick options
Step 1: Software and Steps
Step 2: Hardware
Step 3: General Cluster Settings
Step 4: Security
General Options
Cluster name
My cluster
Logging
S3 folder
s3://aws-logs-632152448932-us-west-2/elasticmapreduce/
Debugging
Termination protection
Scale down behavior
Terminate at instance hour
Tags
Key
Value (optional)
Add a key to create a tag
Additional Options
EMRFS consistent view
Custom AMI ID
None
Bootstrap Actions
Cancel
Previous
Next

Some serious computer power here.

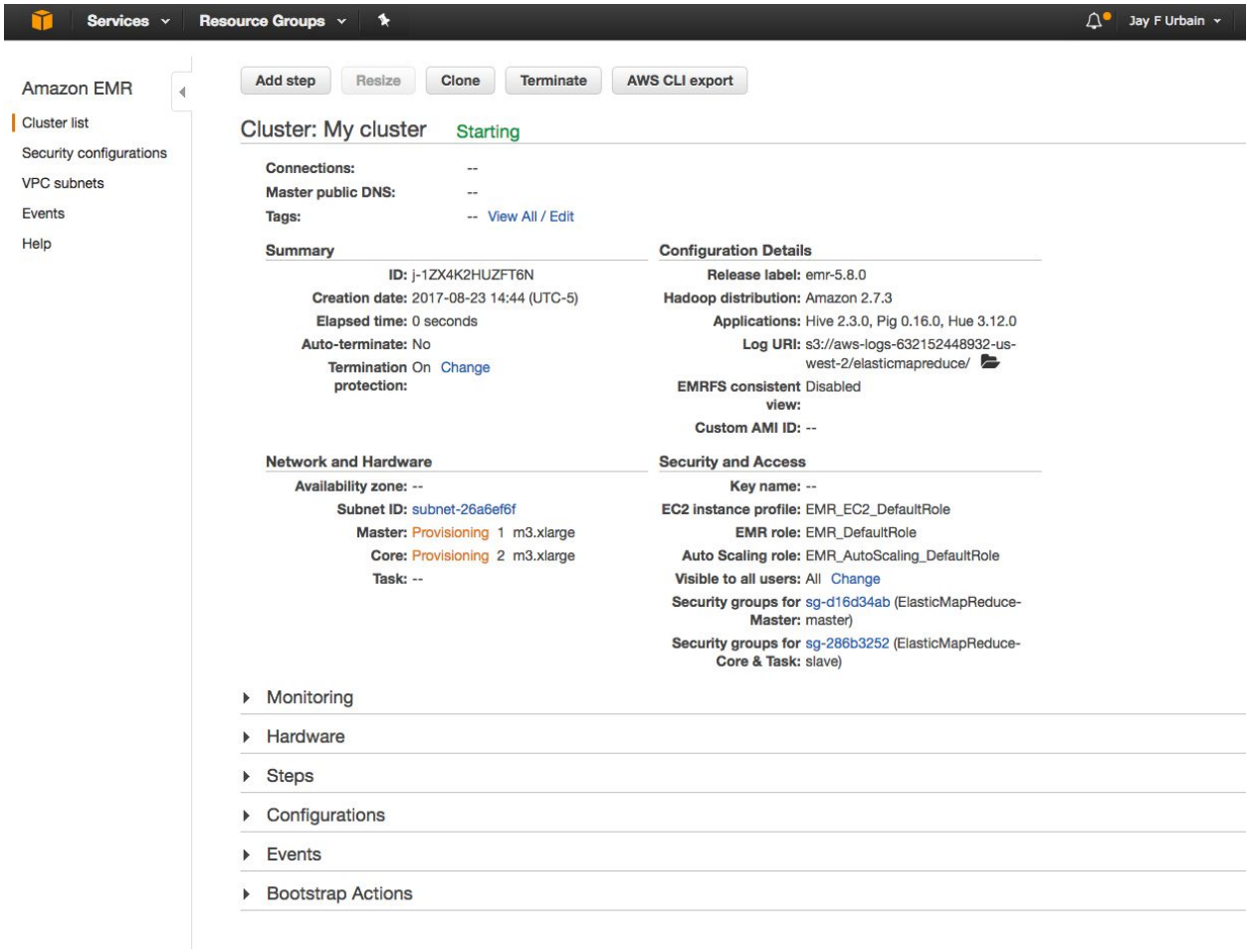
Select terminate at task completion



Next.

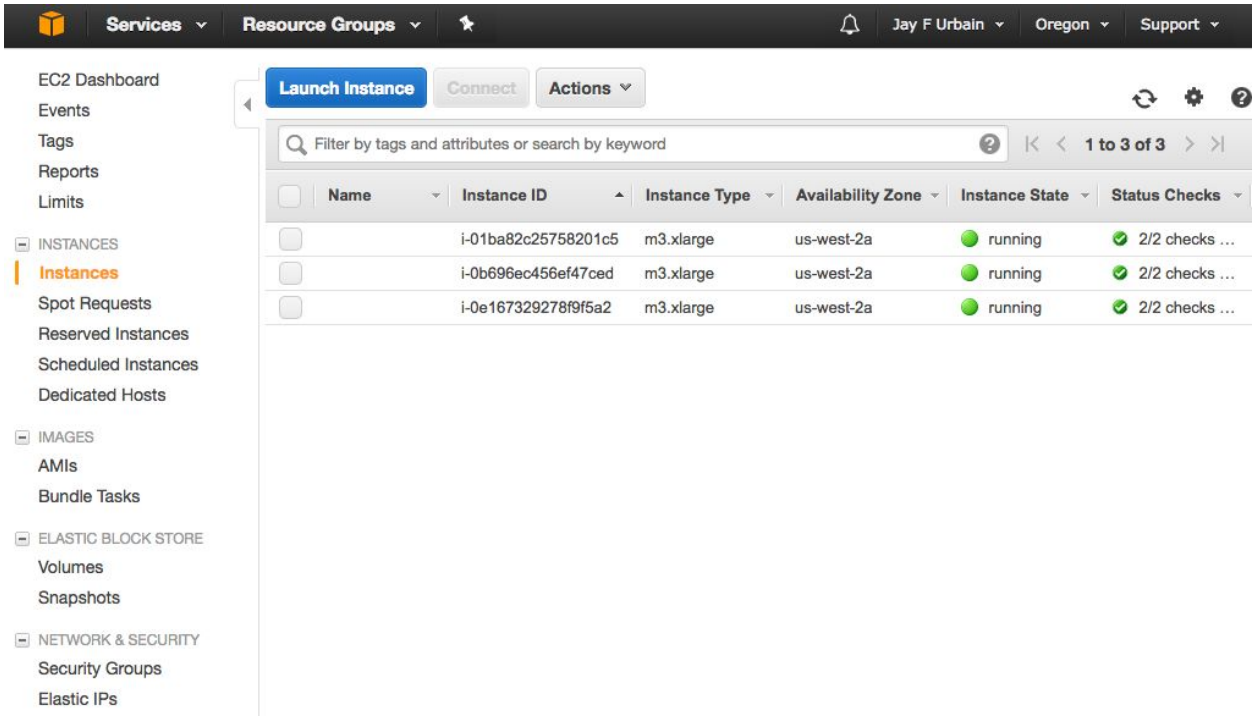


Create cluster

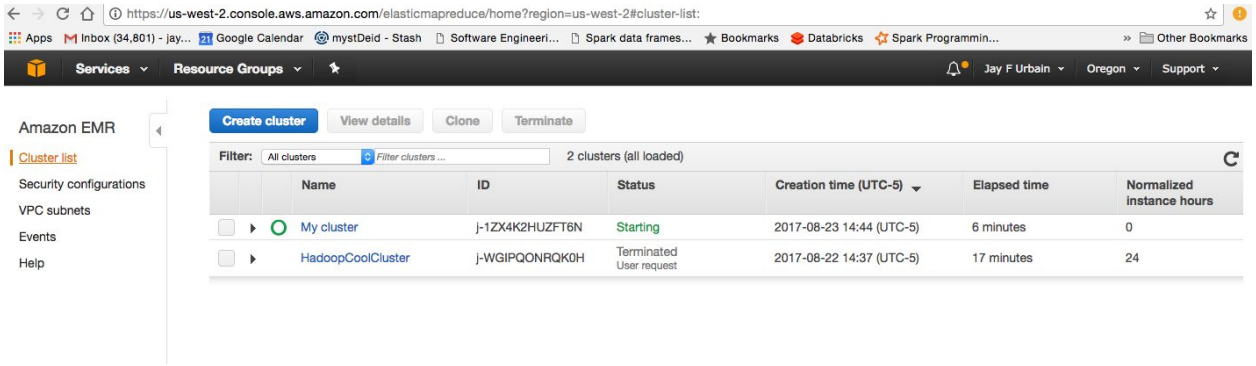


Be patient. If you're bored, you can look at the instances starting:

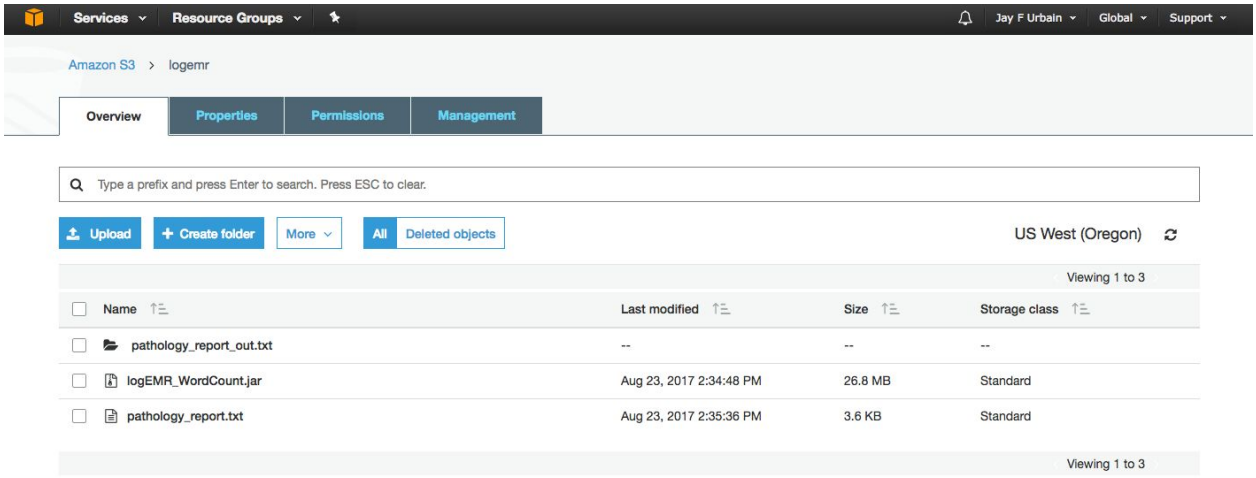
Go back to services, select EC2, and then select EC2 instances running:



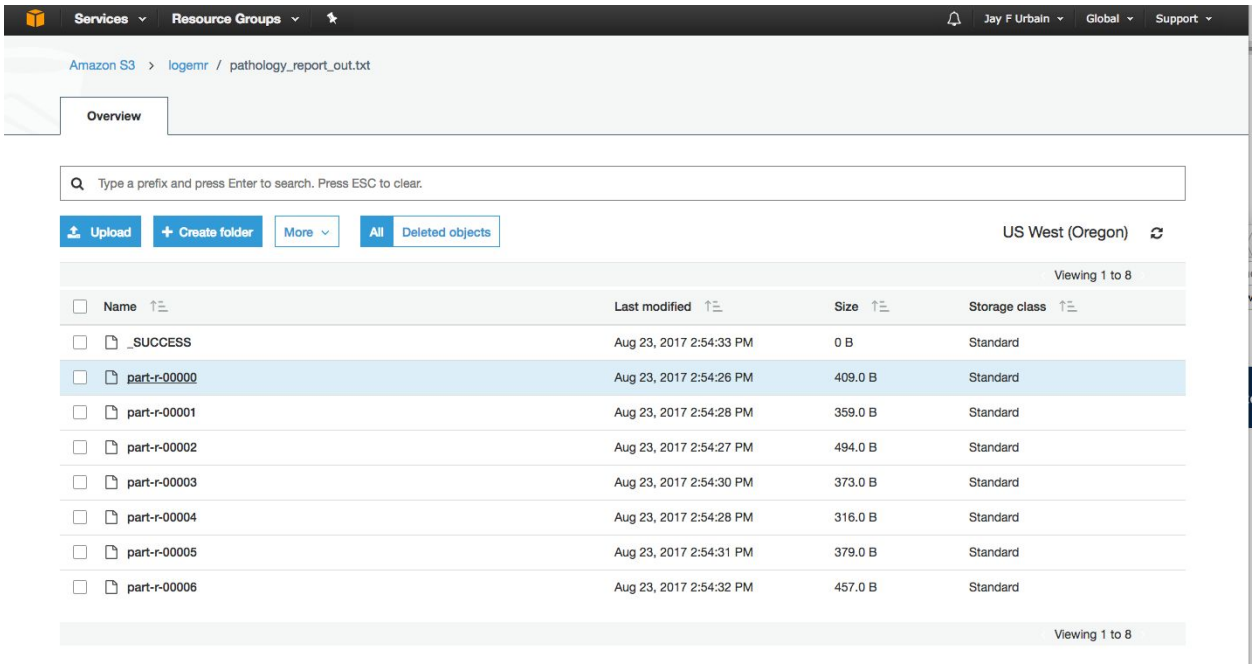
Go back to your clusters.



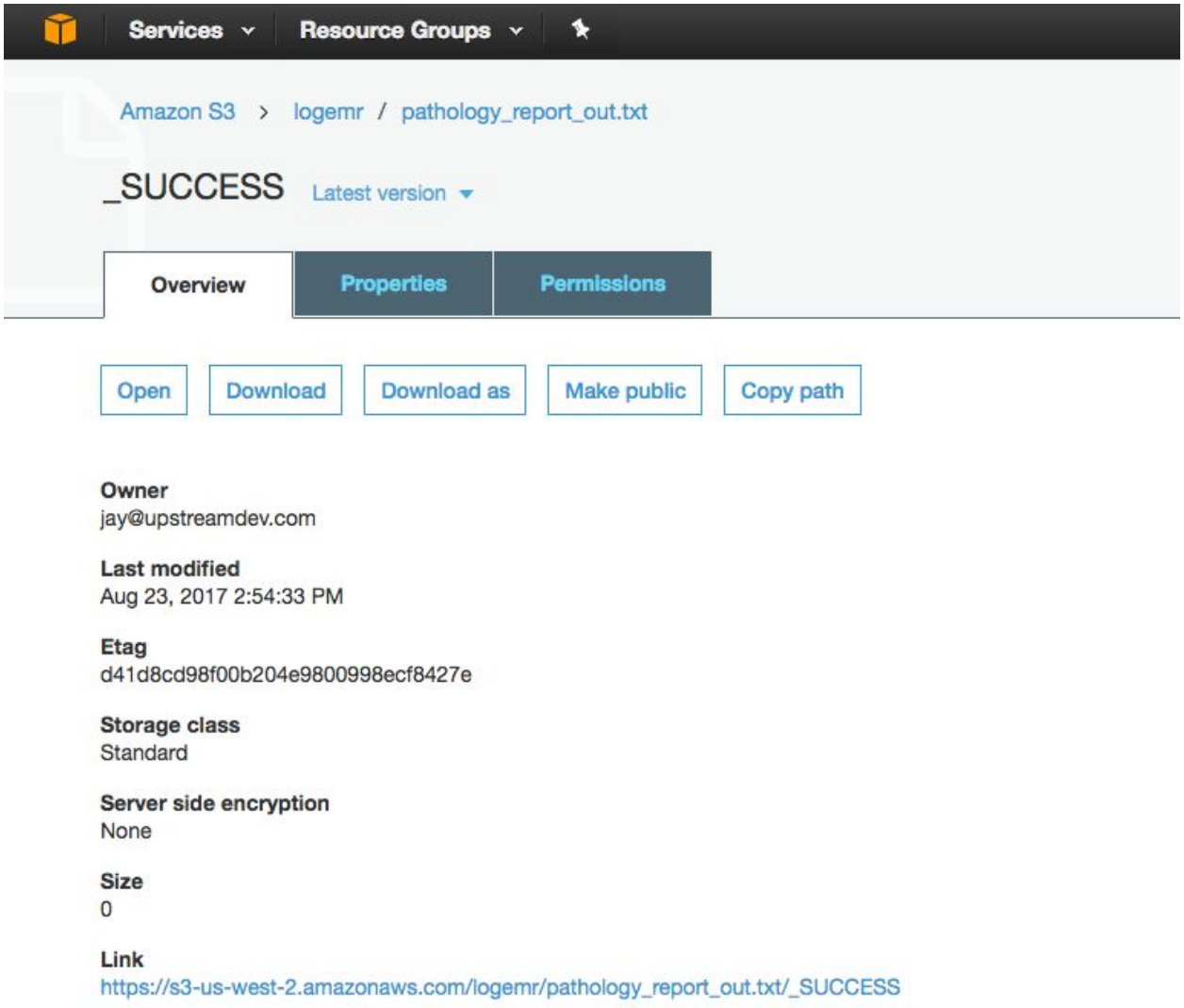
Check S3



This is exciting! Select your output directory.

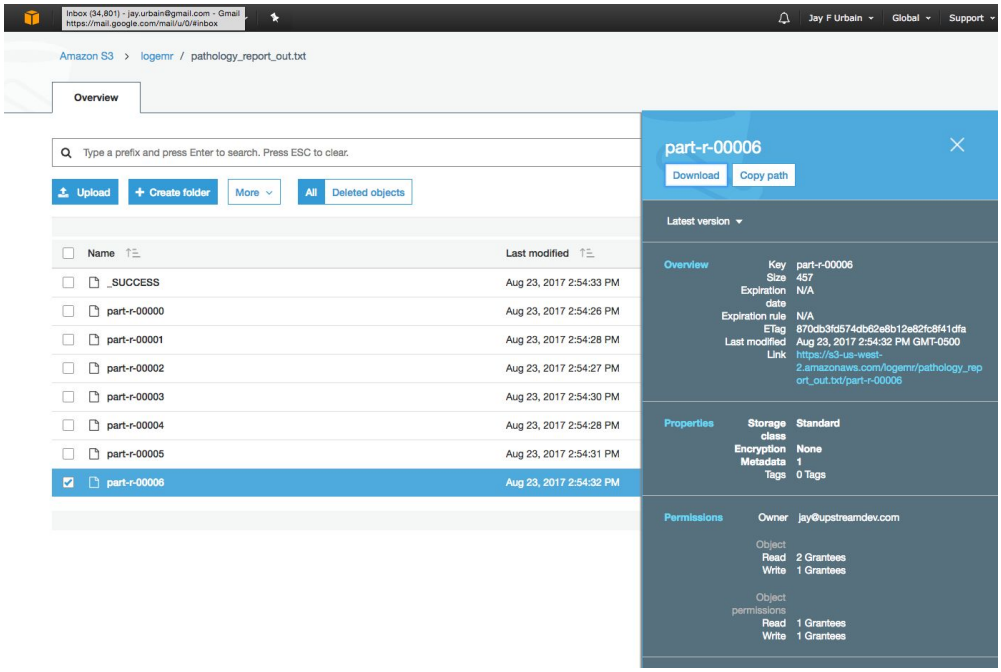


SUCCESS!!!



You can download output files individually. S3 is not designed for browsing file contents.

Note: if you install the command line tools, you can do the following:
aws s3 sync s3://mybucket .

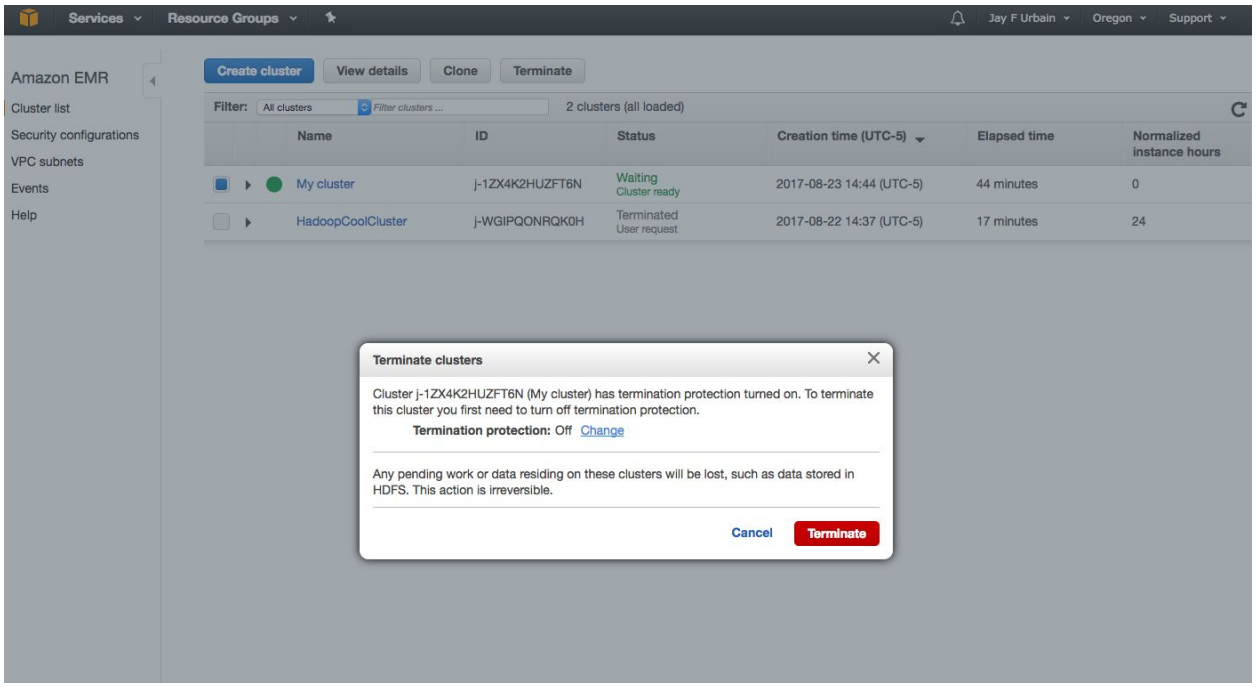

Inbox (34,801) - jay.urbain@gmail.com - Gmail
https://mail.google.com/mail/u/0/#inbox
Jay F Urbain
Global
Support
Amazon S3 › logemr / pathology_report_out.txt
Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
+ Create folder
More
All
Deleted objects
Name
Last modified
_SUCCESS Aug 23, 2017 2:54:33 PM
part-r-00000 Aug 23, 2017 2:54:26 PM
part-r-00001 Aug 23, 2017 2:54:28 PM
part-r-00002 Aug 23, 2017 2:54:27 PM
part-r-00003 Aug 23, 2017 2:54:30 PM
part-r-00004 Aug 23, 2017 2:54:28 PM
part-r-00005 Aug 23, 2017 2:54:31 PM
part-r-00006 Aug 23, 2017 2:54:32 PM
part-r-00006
Download
Copy path
Latest version
Overview
Key part-r-00006
Size 457
Expiration date N/A
Expiration rule N/A
ETag 870db3fd574db62e8b12e82fc8f41dfa
Last modified Aug 23, 2017 2:54:32 PM GMT-0500
Link https://s3-us-west-2.amazonaws.com/logemr/pathology_report_out.txt/part-r-00006
Properties
Storage class Standard
Encryption None
Metadata 1
Tags 0 Tags
Permissions
Owner jay@upstreamdev.com
Object
Read 2 Grantees
Write 1 Grantees
Object permissions
Read 1 Grantees
Write 1 Grantees


part-r-00006
part-r-00000
part-r-00001
part-r-00002
part-r-00003
part-r-00004
part-r-00005
part-r-00006
1. 1
Additional 1
Colon: 7
Descending 1
Diagnosis:none 1
During 1
Histopathologic 1
IV 2
Left 1
Medications 1
Medications: 1
Midazolam 1
Normal 9
Olympus 1
Recommendations: 1
Sigmoid 1
The 3
[LOCATION] 1
achieve 1
are 1
average 2
desaturation. 2
difficult. 1
direct 2
discomfort, 2
examination, 1
for 4
help 1
here 2
ileum. 1
inadequate 1
lavage 1
liquid) 1
monitored 1
oximetry 1
place. 1

You can create a single file using cat:

```
Jays-MacBook-Pro-2:emr_output jayurbain$ ls
part-r-00000    part-r-00001    part-r-00002    part-r-00003    part-r-00004    part-r-00005    part-r-00006    part-r-al
Jays-MacBook-Pro-2:emr_output jayurbain$ cat part-r-00000 part-r-00001 part-r-00002 part-r-00003 part-r-00004 part-r-00005 part-r-00006 > part-r-all
Jays-MacBook-Pro-2:emr_output jayurbain$ ls
part-r-00000    part-r-00001    part-r-00002    part-r-00003    part-r-00004    part-r-00005    part-r-00006    part-r-al       part-r-all
Jays-MacBook-Pro-2:emr_output jayurbain$ cat part-r-all
After   2
CF      1
Currently       1
Details:        1
Grade   3
Impression:     1
MD      3
Physician       1
Prep    1
Procedure       2
Rectum: 1
Small   2
[PERSON]        5
and     10
and/or  2
completion      1
described       1
diet.   1
digital 1
evidence        1
found   1
hemorrhoids     2
home,   1
in      6
incrementally   1
lateral 1
made    1
obtain  1
obtained.       1
over    1
procedure       5
```

Return to EMR screen, make sure your cluster is dead. The meter is running, Select Terminate.



AWS Command line tools

You can install AWS command line tools and interact with AWS through a terminal window.
https://aws.amazon.com/cli/?sc_channel=PS&sc_campaign=acquisition_US&sc_publisher=google&sc_medium=command_line_b&sc_content=aws_cli_p&sc_detail=aws%20cli&sc_category=command_line&sc_segment=159752350313&sc_matchtype=p&sc_country=US&s_kwcid=AL!4422!3!159752350313!p!!g!!aws%20cli&ef_id=WM1YAAAAAHwO1Q7Z:20170823201900:s

Download Hadoop and run locally

http://hadoop.apache.org/releases.html

Download source tar ball.

You should install Cygwin or better yet, buy a Mac. When installing Cygwin, make sure to select the Bash shell and OpenSSL features to be able to develop and run the MapReduce examples locally on Windows systems.

Hadoop, Hive, and Pig require the JAVA_HOME environment variable to be set. It is also typically good practice to have Java in the PATH so scripts and applications can easily find it. On a Linux machine, you can use the following command to specify these settings:

```
export JAVA_HOME=/usr/java/latest
export PATH=$PATH:$JAVA_HOME/bin
```

After you install Hadoop, it is convenient to add Hadoop to the path and define a variable that references the location of Hadoop for other scripts and routines that use it. The following example shows these variables being added to the .bash_profile on a Linux system to define the home location and add Hadoop to the path:

```
$ export HADOOP_INSTALL=/home/user/hadoop-0.20.205.0
$ export PATH=$PATH:$HADOOP_INSTALL/bin
```

You can confirm the installation and setup of Hadoop by running it at the command line. The following example shows running the hadoop command line and the version installed:

```
$ hadoop version
Hadoop 0.20.205.0
Subversion https://svn.apache.org/repos/asf/hadoop/
```

common/branches/branch-0.20-security-205 -r 1179940
Compiled by hortonfo on Fri Oct 7 06:26:14 UTC 2011