



Cloud Services in Practice


Documents, Services and Data on the Web

Norman Paton, Sandra Sampaio
School of Computer Science
University of Manchester

Case Study: Amazon Web Services

- Amazon Web Services (AWS – aws.amazon.com) provide a wide range of services, including:
 - Infrastructure as a service:
 - EC2: purchase of virtual machines of different capabilities, with different operating systems, and for different periods. 
 - S3: purchase of storage that is accessed through a simple file system style interface.
 - Platform as a service:
 - EMR (Elastic Map Reduce): the ability to run scalable applications written using the map reduce programming model over EC2 and S3 infrastructure. 
- Using EMR involves storing some data in S3, and then creating a map reduce job that runs on some EC2 machines.

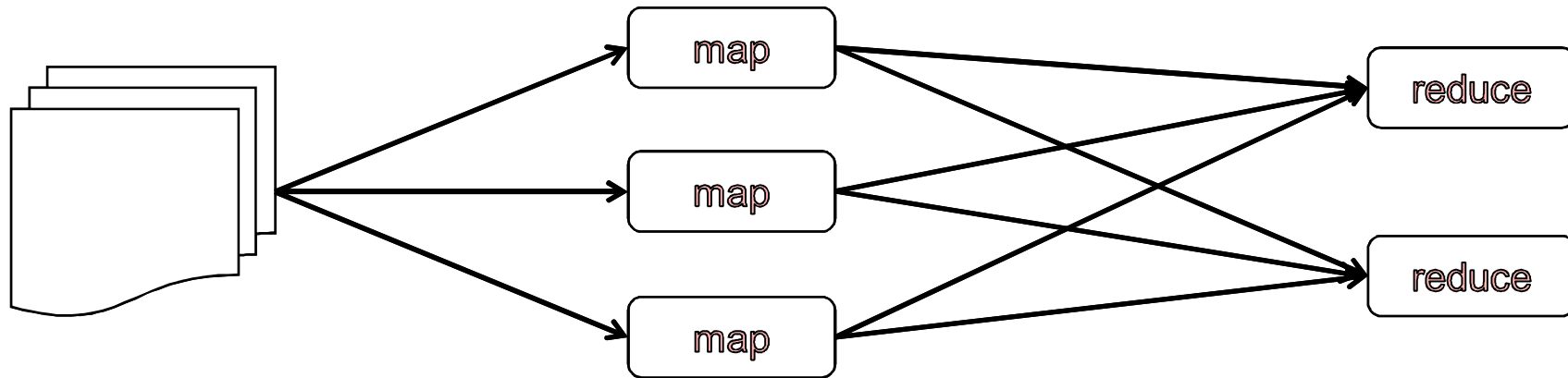
Map Reduce

- *Map Reduce* is a scalable programming model, originally developed by Google for tasks such as index building.
- In Map Reduce:
 - applications are developed using two simple, functional operations (*map* and *reduce*);
 - the infrastructure supports the running of Map Reduce applications in parallel on potentially huge data sets on potentially numerous commodity machines. 
- *Hadoop* is a widely used open source implementation of map reduce (hadoop.apache.org).

Word Count Example


- The standard map reduce example program counts the number of occurrences of each word in a document (although this is in some ways a toy task, it is relevant to web indexing).

Word Count at Runtime



the reduce operation, given the id of a document and the document, emits a collection of key-value pairs ...	the reduce operation, given the id of	(the, 1) (reduce, 1) (the, 1) ...	(the, [1,1,1]) (reduce,[1])) ...	(the, 3) (reduce, 1) ...
	a document and the document,	(the, 1) (document, 1) ...	(of, [1,1]) (pairs,[1])) ...	(of, 2) (pairs, 1) ...
		

Word Count Map

- Recall the description of map: 
 - $\text{map}(\text{key}_1, \text{value}_1) \rightarrow [(\text{key}_2, \text{value}_2)]$
 - map, given a key key_1 and a value value_1 , generates a collection of key-value pairs $(\text{key}_2, \text{value}_2)$.
- In WordCount:
 - key_1 is the identifier of the document (not used).
 - value_1 is the document (or part of the document).
 - key_2 is a word from the document.
 - value_2 is an occurrence count for key_2 .

Map Pseudo-Code

- The map operation, given the id of a document and the document, emits a collection of key-value pairs, where the key is a word in the document and the value is a (partial) count of the number of occurrences of the word in the document.

```
map(documentId key1, document value1) {  
    for each term t in value1 do  
        emit(term t, count 1)  
}
```

Word Count Reduce

- Recall the description of reduce:
 - $\text{reduce}(\text{key}_2, [\text{value}_2]) \rightarrow [(\text{key}_3, \text{value}_3)]$
 - reduce, given a key key_2 output by map, and a collection of all the values value_2 associated with that key, returns a new collection of key-value pairs.
- In WordCount:
 - key_2 is a term from the document processed by map.
 - value_2 is a (partial) count of occurrences of key_2 from map.
 - key_3 is the same as key_2 .
 - value_3 is the total occurrence count for key_3 .

Reduce Pseudo-Code

- The reduce operation, given a term and a list of partial counts of the term from map, emits a collection of key-value pairs, where the key is the term and the value is the sum of the partial counts.

```
reduce(term key2, list of count value2) {  
    sum = 0  
    for each count in value2 do  
        sum = sum + count;  
    emit(term key2, count sum)  
}
```

Why like this...

- The basic idea is that, without modifying your code (partial truth), a map reduce program can run with very high levels of parallelism over huge files.
- Normally to write a parallel application you need to worry about data partitioning, scheduling, load balancing, ...
- The classic web scale example: given data from a web crawl, build an index for use by a search engine.



How to Run it on the Cloud?

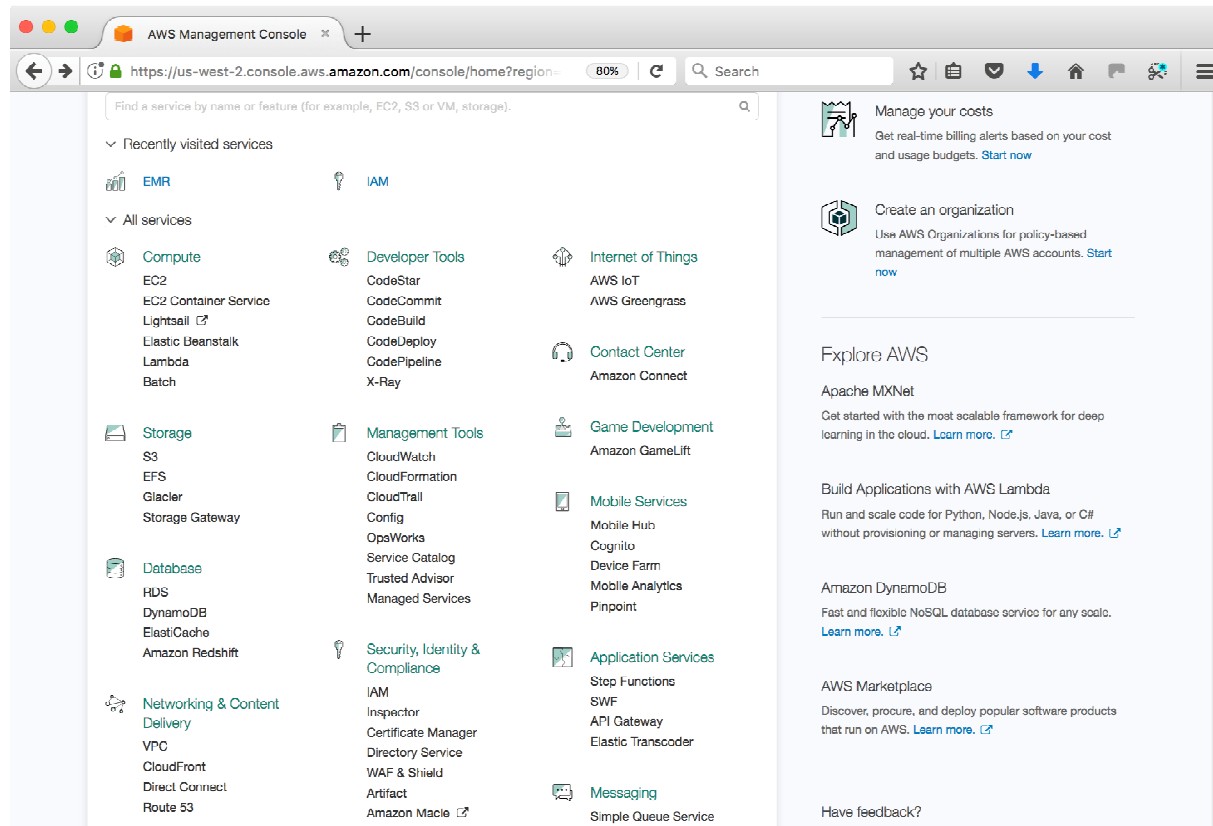
- You can run a Word Count example on the cloud using Amazon Web Services (AWS).
- Note that the AWS web interface changes on an ongoing basis, so some of the screen shots may not exactly match what you see.
- The basic idea:
 - You can upload a text file of your choice to a cloud data store (S3).
 - You can run an existing WordCount map/reduce program using a (small) dynamically provisioned cluster (EMR).
 - You can look at the results of the WordCount run that will also be stored in S3.

Prerequisite


- First, log into the AWS management console.
- This should have been done by the time you need to move on to the next step...

AWS Management Console

- The Management Console provides access to a comprehensive list of services.



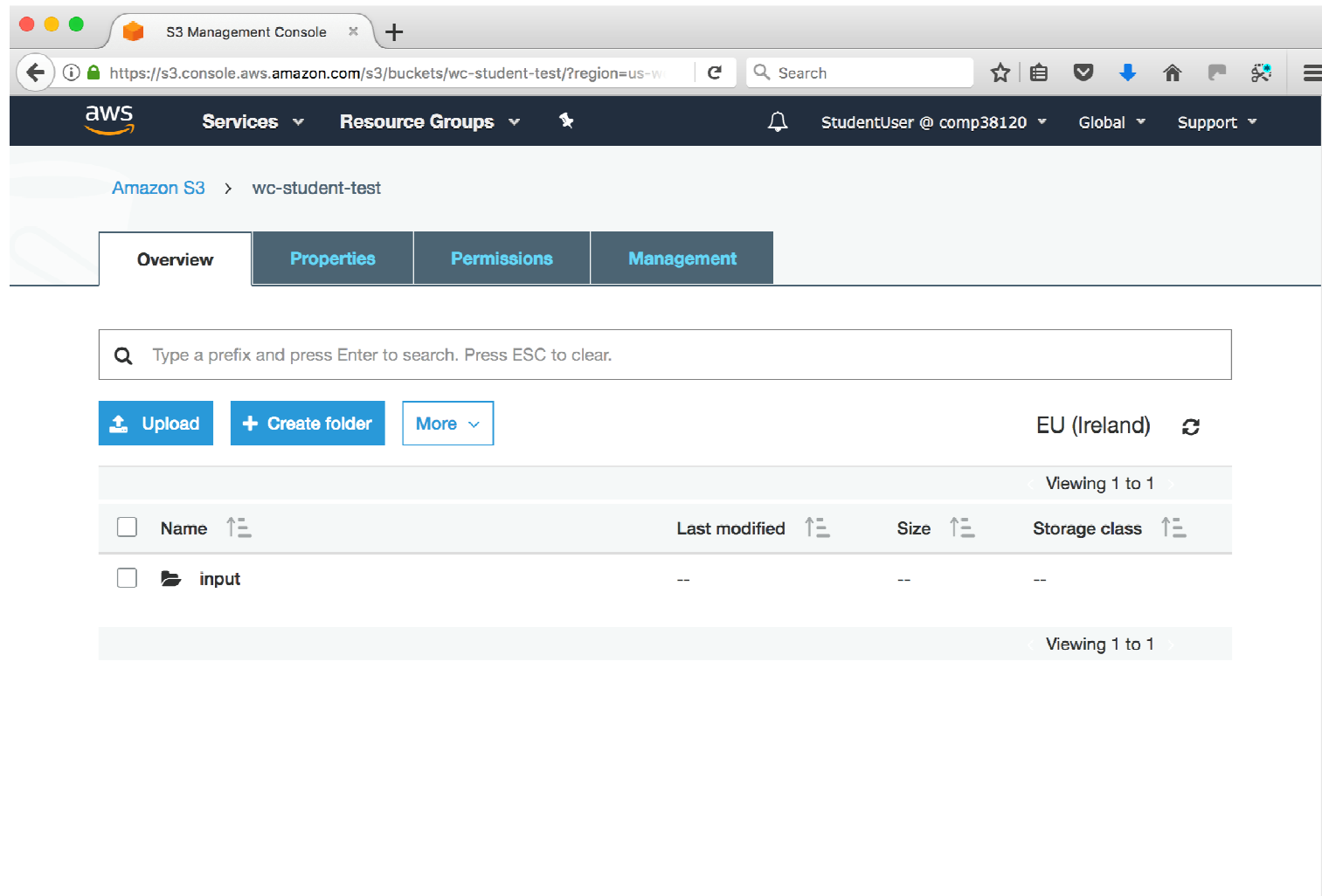
Amazon S3 Storage

- S3 provides a hierarchical file structure, which will be used for:
 - The input to a map/reduce program.
 - The JAR that contains the WordCount.
 - The output from the execution of the program.
 - Logging information.
- For this activity:
 - Buckets are essentially folders. 
 - You can create a bucket in S3, where you can store the JAR you will use, for example:
 - `s3://uom38120wordcount/WordCount.jar`

Prepare S3 For Your Run

- Move to the S3 Service.
 - Services -> S3.
- Create another bucket:
 - Action: *+CreateBucket*.
 - Give it a distinctive name.
 - The one in the slides is called *wc-student-test*.
 - Take the default properties for the bucket.
- Create a folder called *input* in your group bucket.
 - Change into your bucket by clicking on it.
 - Action: *+CreateFolder*.

Your New Bucket and Folder

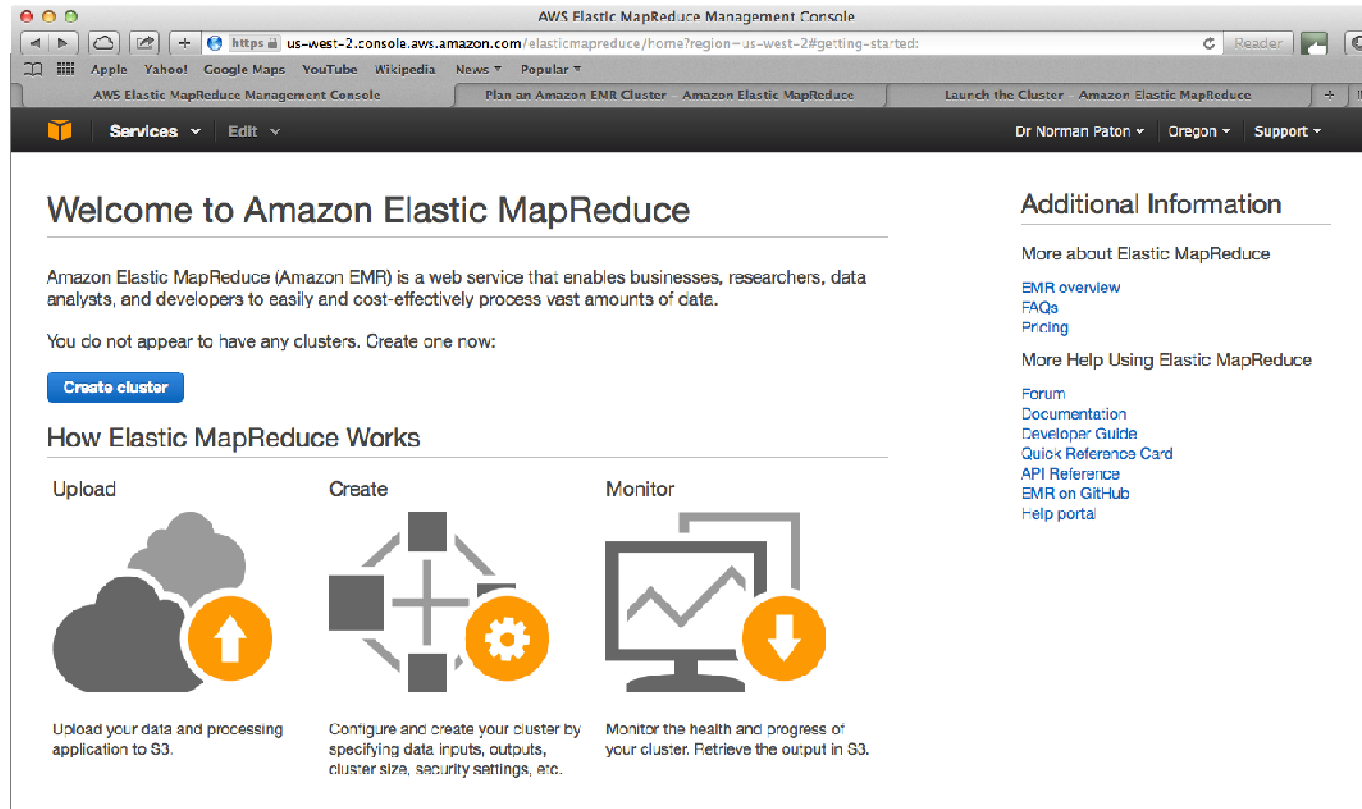


Upload your Data

- Change into your *input* folder by clicking on it.
- Upload a text file (neither tiny nor huge) of your choice:
 - Action: *Upload*.
- Just accept the default properties for the uploaded file.

Run Map Reduce Program

- Move to the Elastic Map Reduce (EMR) service:
 - Services -> Elastic Map Reduce.



Create a cluster for your job

- Click on Create Cluster.
- Use defaults unless otherwise specified.
- Define The Cluster:
 - Give it some form of distinctive name.
 - Define the log file in your bucket. Mine is: *s3://wc-student-test/log/*
 - This does not have to already exist.

The screenshot shows the AWS EMR console interface for creating a new cluster. The browser address bar indicates the URL is `https://us-west-2.console.aws.amazon.com/emr/v5/home?region=us-west-2#/create-cluster/quick-options`. The page title is "Create Cluster - Quick Options" with a link to "Go to advanced options".

The "General Configuration" section includes:

- Cluster name:** A text input field containing "My cluster".
- Logging:** A checkbox that is checked, with an information icon.
- S3 folder:** A text input field containing "s3://aws-logs-297771139352-us-west-2/elasticmapreduce/".
- Launch mode:** Two radio buttons: "Cluster" (selected) and "Step execution".

The "Software configuration" section includes:

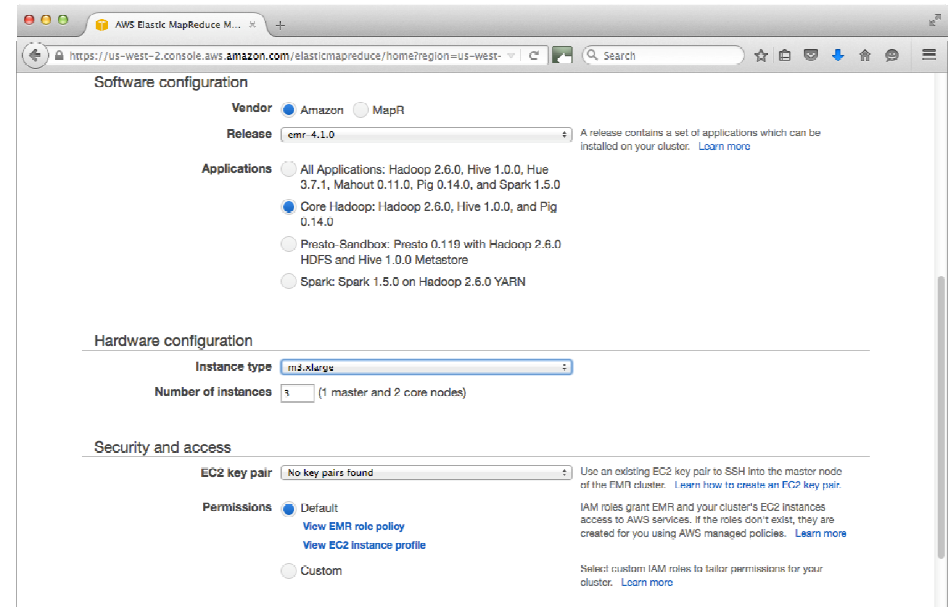
- Release:** A dropdown menu showing "emr-5.9.0".
- Applications:** A list of application stacks with radio buttons:
 - Core Hadoop:** Selected. Description: "Core Hadoop: Hadoop 2.7.3 with Ganglia 3.7.2, Hive 2.3.0, Hue 4.0.1, Mahout 0.13.0, Pig 0.17.0, and Tez 0.8.4".
 - HBase:** "HBase 1.3.1 with Ganglia 3.7.2, Hadoop 2.7.3, Hive 2.3.0, Hue 4.0.1, Phoenix 4.11.0, and ZooKeeper 3.4.10".
 - Presto:** "Presto 0.170 with Hadoop 2.7.3 HDFS and Hive 2.3.0 Metastore".
 - Spark:** "Spark 2.2.0 on Hadoop 2.7.3 YARN with Ganglia 3.7.2 and Zeppelin 0.7.2".
- Use AWS Glue Data Catalog for table metadata:** An unchecked checkbox with an information icon.

The "Hardware configuration" section is partially visible at the bottom.

The footer of the console includes "Feedback", "English (US)", and copyright information: "© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved." along with links to "Privacy Policy" and "Terms of Use".

Specify Parameters

- Software configuration:
 - Amazon.
 - Emr-5.19.0.
- Applications:
 - Core Hadoop 2.8.5.
- Hardware configuration:
 - You will get access to 3 (virtual) machines, 1 master and 2 workers.
 - Choose m4.large for them the configuration.
- Finally, Create cluster.



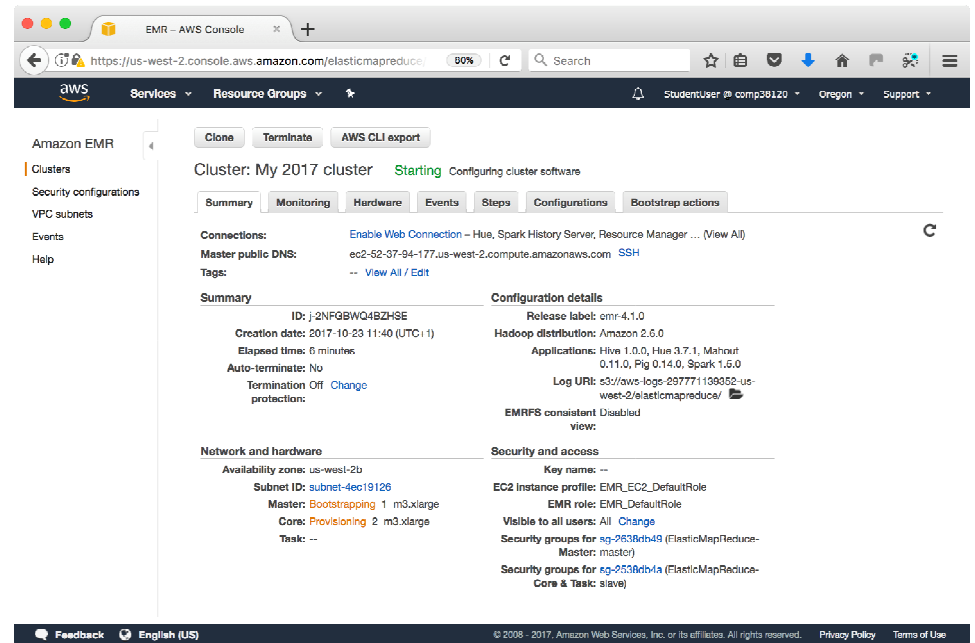
Add a Step

- Select Steps -> *Add Step*.
- Step type:
 - Custom JAR.
- Name:
 - Something distinctive
- JAR Location:
 - `s3://uom38120wordcount/WordCount_2.jar`
- Arguments: Your space-separated input and not-yet-existing output; mine are:
 - `s3://wc-student-test/input`
 - `s3://wc-student-test/output`
- Action on failure: continue

The screenshot shows the AWS Elastic MapReduce Management Console interface. At the top, the title bar reads 'AWS Elastic MapReduce Management Console'. Below it, the URL is '2.console.aws.amazon.com/elasticmapreduce/home?region=us-west-2#cluster-details:j-391HY4PS9I9UT'. The main navigation bar includes links for 'Wikipedia', 'News', and 'Popular'. The main content area has tabs for 'Plan an Amazon EMR Cluster - Amazon E...', 'Launch the Cluster - Amazon Elastic Ma...', and 'St...'. The 'Launch the Cluster' tab is active, showing 'EMR role: --' and 'Visible to all All Change users:'. Below this, the 'Add Step' form is displayed. The form includes a 'Step type' dropdown set to 'Custom JAR', a 'Name*' text field with 'Custom JAR', a 'JAR location*' text field, and an 'Arguments' text area. To the right of the 'JAR location*' field, there is a note: 'JAR location maybe a path into S3 o class in the classpath.' Below the 'Arguments' field, there is a note: 'These are passed to the main functi JAR does not specify a main class in can specify another class name as th'. At the bottom of the form, there is a 'What to do if the step fails.' dropdown set to 'Continue'. The footer of the console shows 's affiliates. All rights reserved. Privacy Policy Terms of Use'.

Running the Job

- This page should auto-update (but refresh at top right to hurry it along).
- Expand Steps to see the details. Your job flow should transition through:
 - Pending, Starting, Running, ... Completed.
- ... but on the other hand may cut to Failed at any point, in which case try the log files.
- At the end please

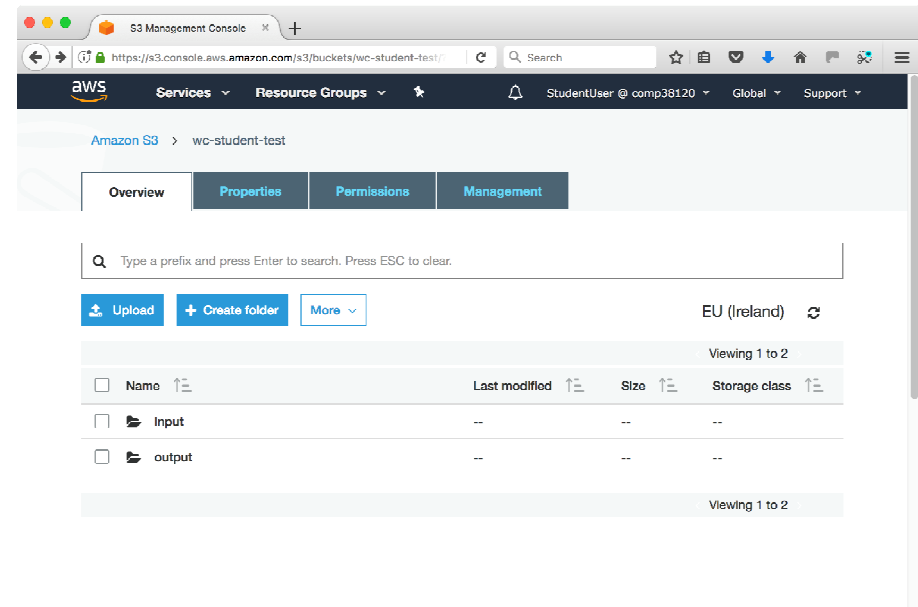


It can take some time to provision a cluster (several minutes).


If your run fails for some reason, you can always add another Step without needing to run the cluster again.

Viewing the Results

- To see the results, go to S3:
 - Services -> S3.
- In your output folder there will be a different result file for each reduce job.
- View results by:
 - Selecting the file.
 - Actions -> Download.



More Information

- The WordCount run here is from the following book, somewhat simplified to act on text files:
 - Donald Miner, Adam Shook, MapReduce Design Patterns, O'Reilly Media, 2012. 
- The user/developer guides for S3 and EMR (huge) are at:
 - <http://aws.amazon.com/documentation/s3/>
 - <http://aws.amazon.com/documentation/elasticmapreduce/>

Later you will...

- Learn more about application development using map reduce, including:
 - Hadoop – the infrastructure, its properties.
 - Map reduce – application design and patterns.
- Apply map reduce in labs to search applications in the web of documents and the web of data.
- Although you will not run hadoop on a cloud for your project, you have now seen what is involved in doing so.