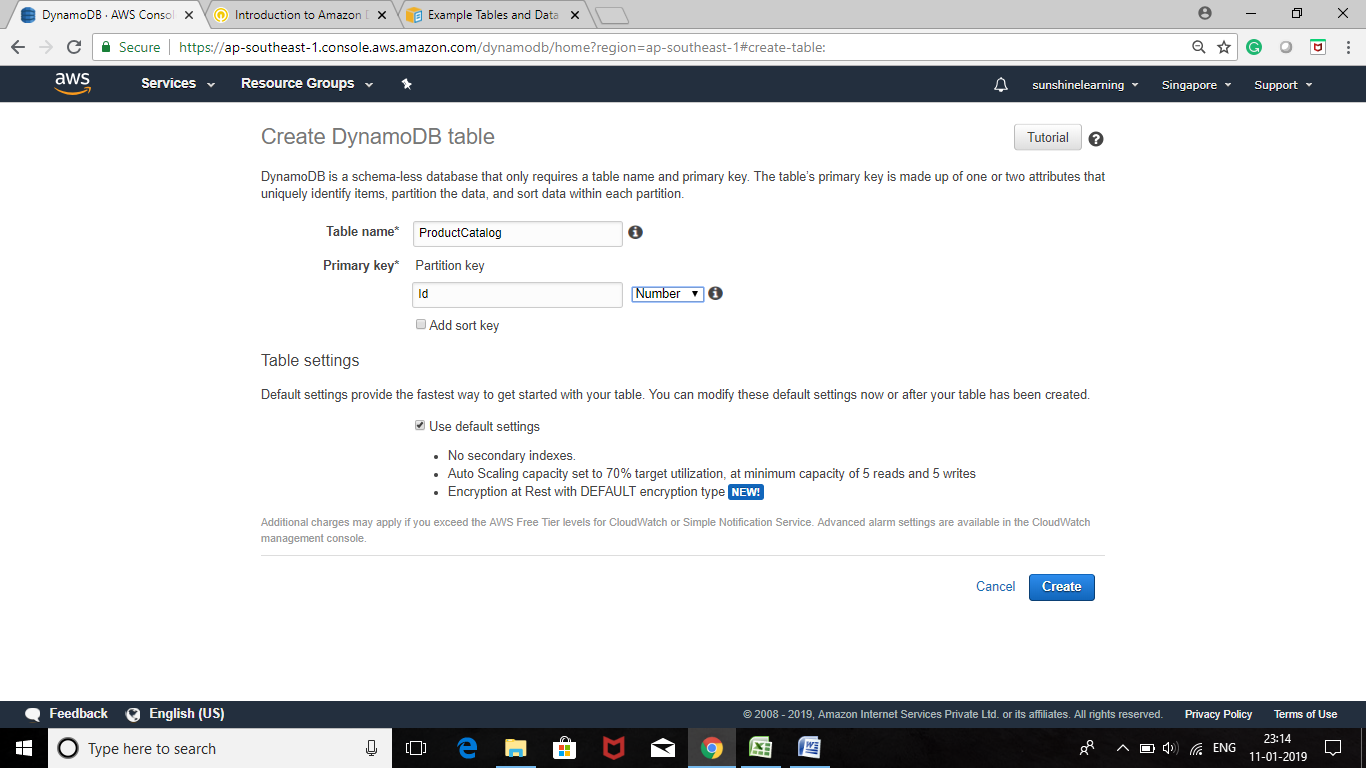
**Amazon DynamoDB** is a fast and flexible NoSQL database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a **fully managed database** and supports both document and key-value data models. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad-tech, lot, and many other applications.

**Example Tables and Data**

The *Amazon DynamoDB Developer Guide* uses sample tables to illustrate various aspects of DynamoDB.

|  |  |
| --- | --- |
| **Table Name** | **Primary Key** |
| *ProductCatalog* | Simple primary key:   * Id (Number) |
| *Forum* | Simple primary key:   * Name (String) |
| *Thread* | Composite primary key:   * ForumName (String) * Subject (String) |
| *Reply* | Composite primary key:   * Id (String) * ReplyDateTime (String) |

**Note:**We will use Product Catalog as an Example, the rest of the items you can practice yourself.



# Step 2: Load Data into Tables

In this step, you will load sample data into the tables that you created. You could enter the data manually into the DynamoDB console; however, to save time, you will use the AWS Command Line Interface instead.

**Note**

If you have not yet set up the AWS CLI, see AWS docs [Using the CLI](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.CLI.html) for instructions.

You will download a .zip archive that contains JSON files with sample data for each table. For each file, you will use the AWS CLI to load the data into DynamoDB. Each successful data load will produce the following output:

{

"UnprocessedItems": {}

}

## Download the Sample Data File Archive

* Download the sample data archive (sampledata.zip) using the link:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.LoadData.html

* + [sampledata.zip](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/samples/sampledata.zip)
* Extract the .json data files from the archive.
* Copy the .json data files to your current directory.

## Load the Sample Data Into DynamoDB Tables

1. To load the ProductCatalog table with data, enter the following command:

awsdynamodb batch-write-item --request-items file://ProductCatalog.json

1. To load the Forum table with data, enter the following command:

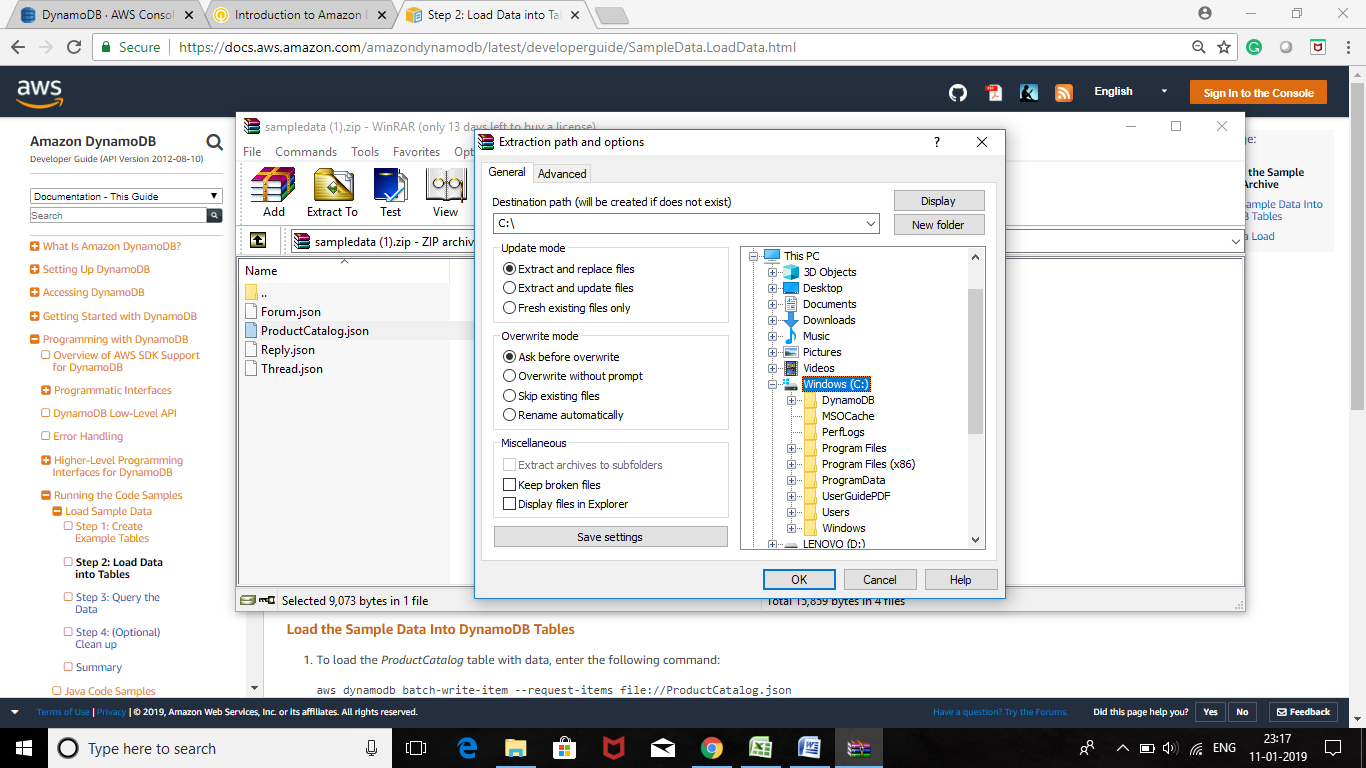
awsdynamodb batch-write-item --request-items file://Forum.json

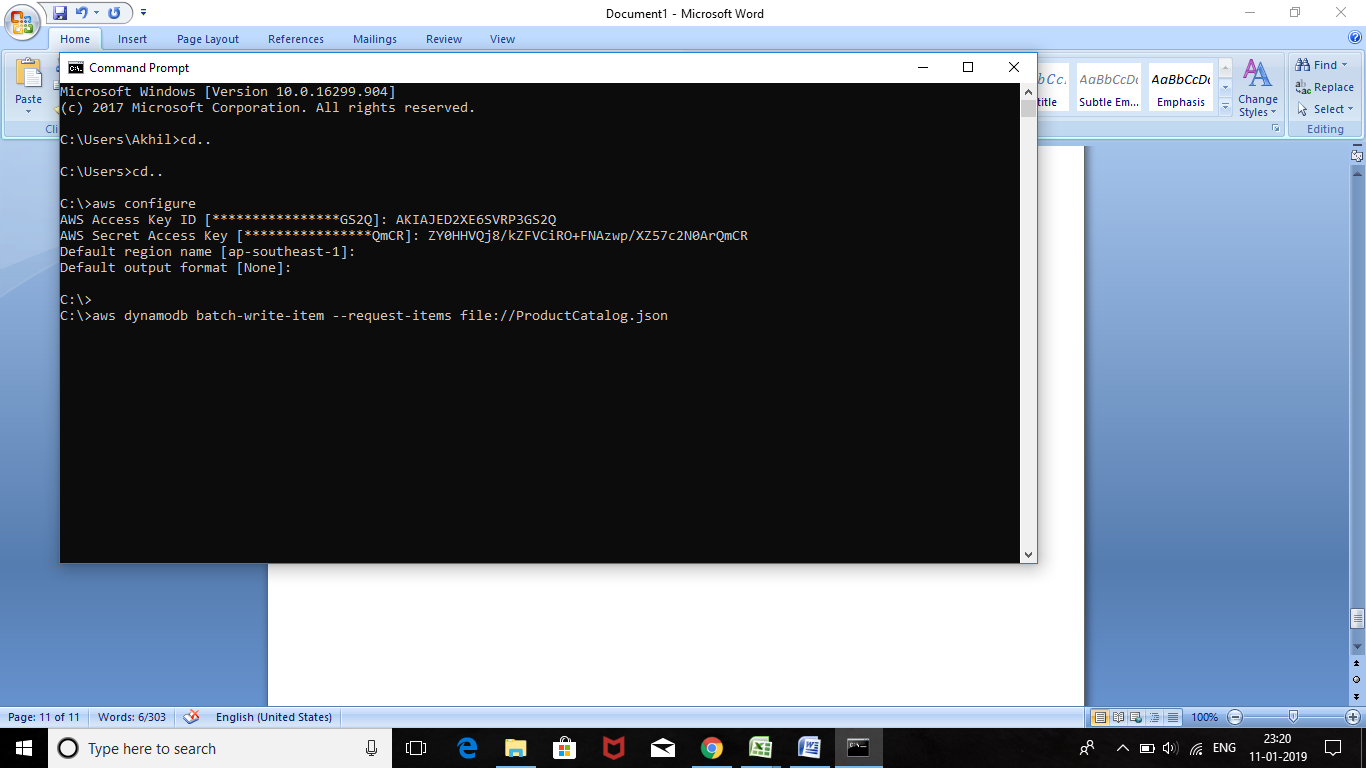
1. To load the Thread table with data, enter the following command:

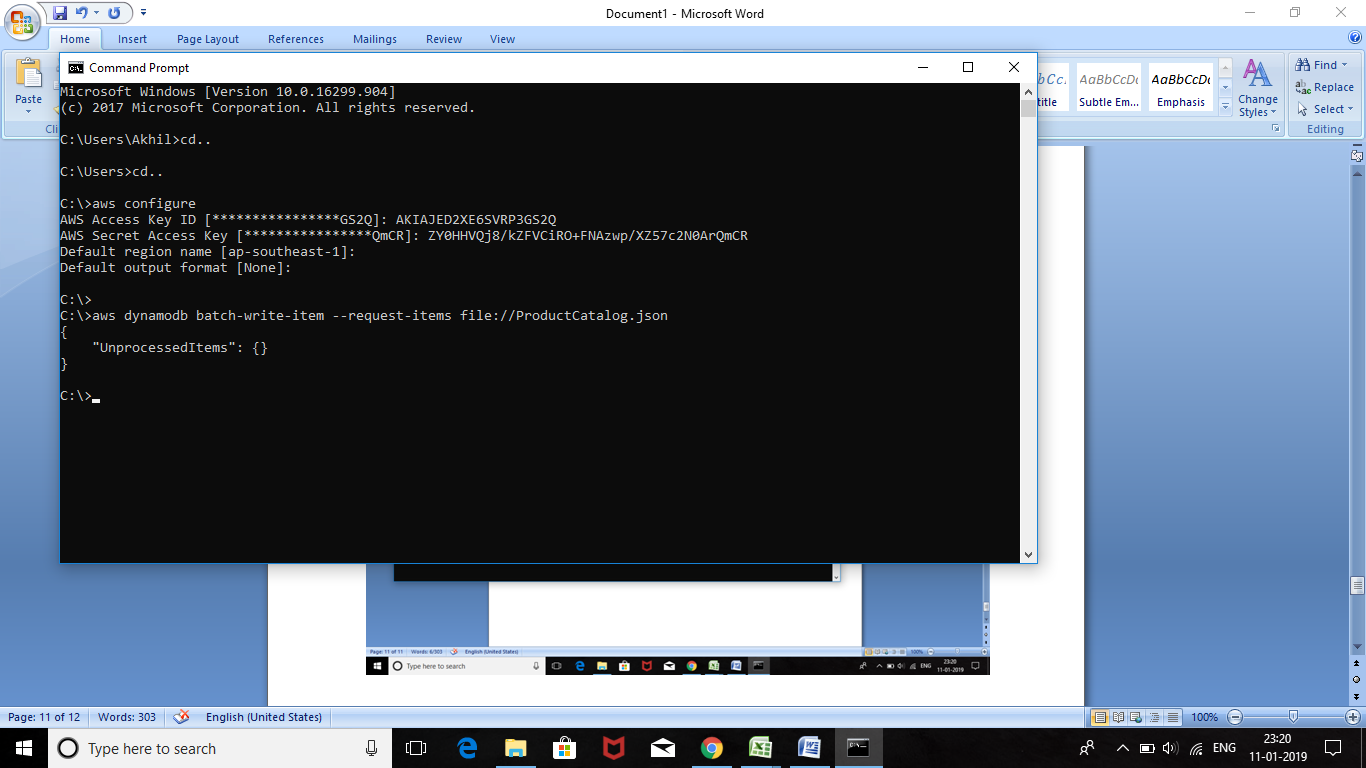
awsdynamodb batch-write-item --request-items file://Thread.json

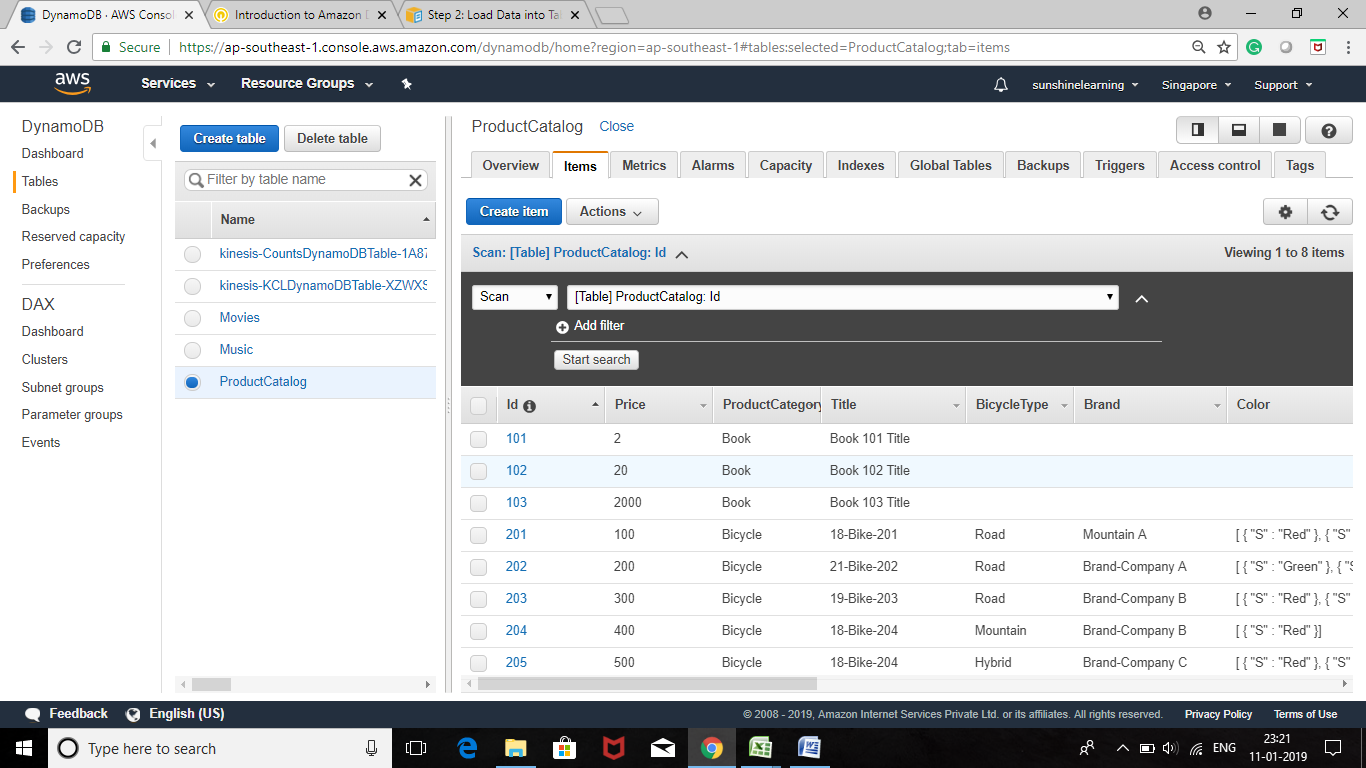
1. To load the Reply table with data, enter the following command:

awsdynamodb batch-write-item --request-items file://Reply.json









For step by step configuration of other tables follow the below link

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.CreateTables.html>

**Analyze Big Data with Hadoop**

**Overview**

**Amazon EMR** is a managed service that makes it fast, easy, and cost-effective to run Apache Hadoop and Spark to process vast amounts of data. Amazon EMR also supports powerful and proven Hadoop tools such as Presto, Hive, Pig, HBase, and more.

In this lab, you will deploy a fully functional Hadoop cluster, ready to analyze log data in just a few minutes. You will start by launching an Amazon EMR cluster and then use a **HiveQL** script to process sample log data stored in an Amazon S3 bucket. HiveQL is a SQL-like scripting language for data warehousing and analysis. You can then use a similar setup to analyze your own log files.

By the end of this lab, you will be able to:

* Launch a fully functional Hadoop cluster using Amazon EMR
* Define the schema and create a table for sample log data stored in Amazon S3
* Analyze the data using a HiveQL script and write the results back to Amazon S3
* Download and view the results on your computer

**Lab Pre-requisites**

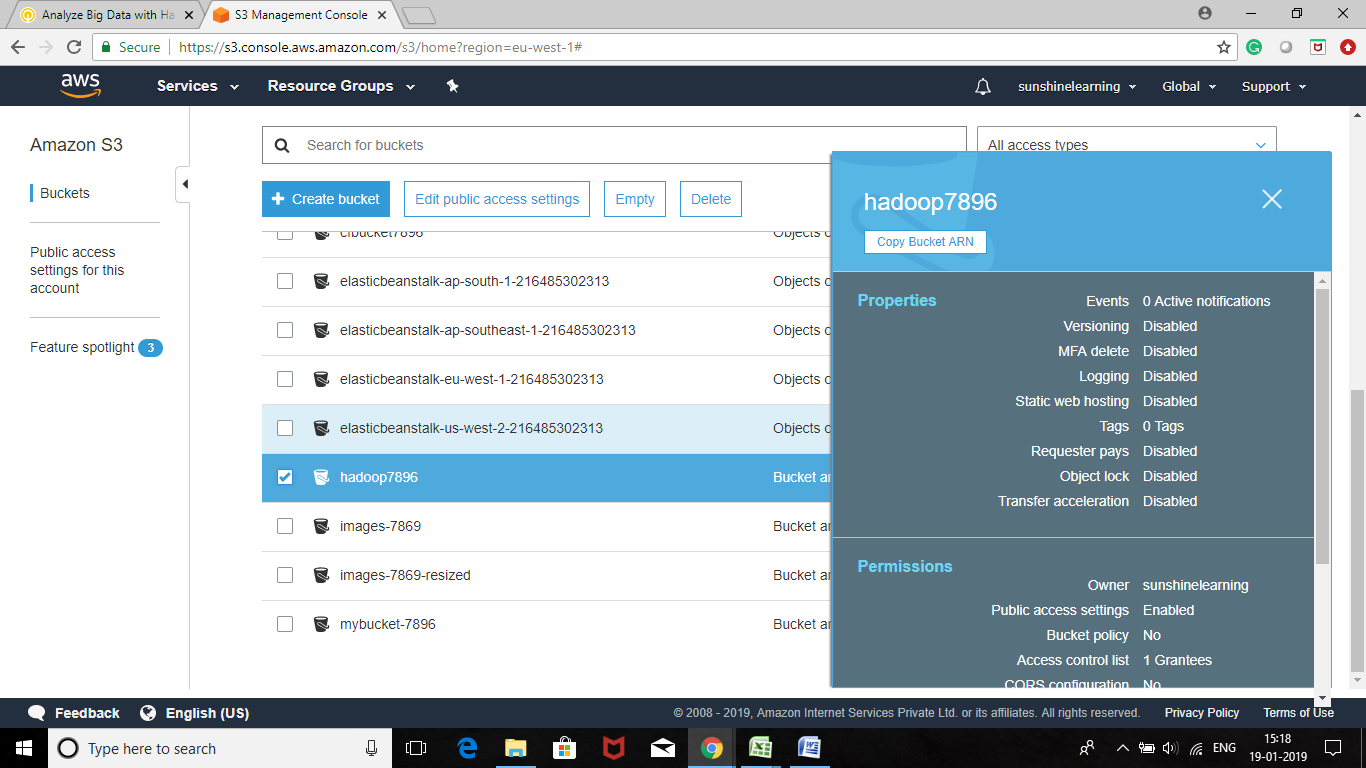
* **IT Experience:** Prior experience with Hadoop is recommended, but not required, to complete this lab
* **AWS Experience**: Basic familiarity with Amazon S3 and Amazon EC2 key pairs is suggested, but not required, to complete this project

**Task 1: Create an Amazon S3 bucket**

In this task, you will create a bucket to store your log files and output data.

* In the **AWS Management Console**, on the Services menu, click S3.
* Click **+Create bucket** then configure:
  + **Bucket name:**hadoopNUMBER
  + Replace **NUMBER** with a random number
  + Click Create

You now have a bucket you can use to store your log files and output data.



**Task 2: Launch an Amazon EMR cluster**

In this task, you will launch a Hadoop cluster that will be used to process data.

* On the **Services** menu, click **EMR**.
* Click Create cluster then configure:
* **Cluster name**: My cluster
* **S3 folder:** 
  + Click the folder Pi icon
  + Click the Hadoop bucket that you created earlier
  + Click Select

This will save the output of the Hadoop job into your Amazon S3 bucket.

* In the **Hardware configuration** section, configure:

This will configure the cluster to launch one master node and one slave ("core") node.

The key pair is used to login to the EMR cluster. In this lab, however, you will submit jobs via the management console so you do not require a key pair.

* **EC2 key pair**: Proceed without an EC2 key pair

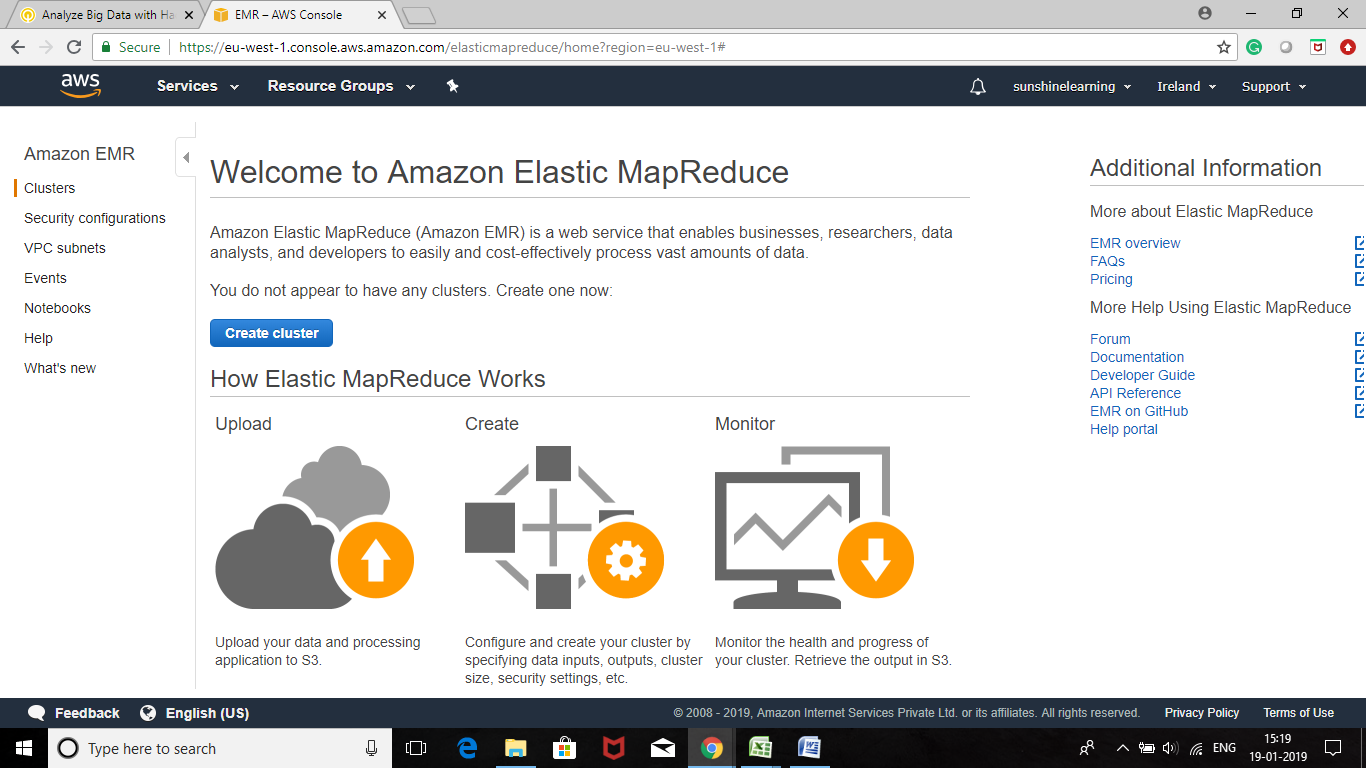
The key pair is used to login to the EMR cluster. In this lab, however, you will submit jobs via the management console so you do not require a key pair.

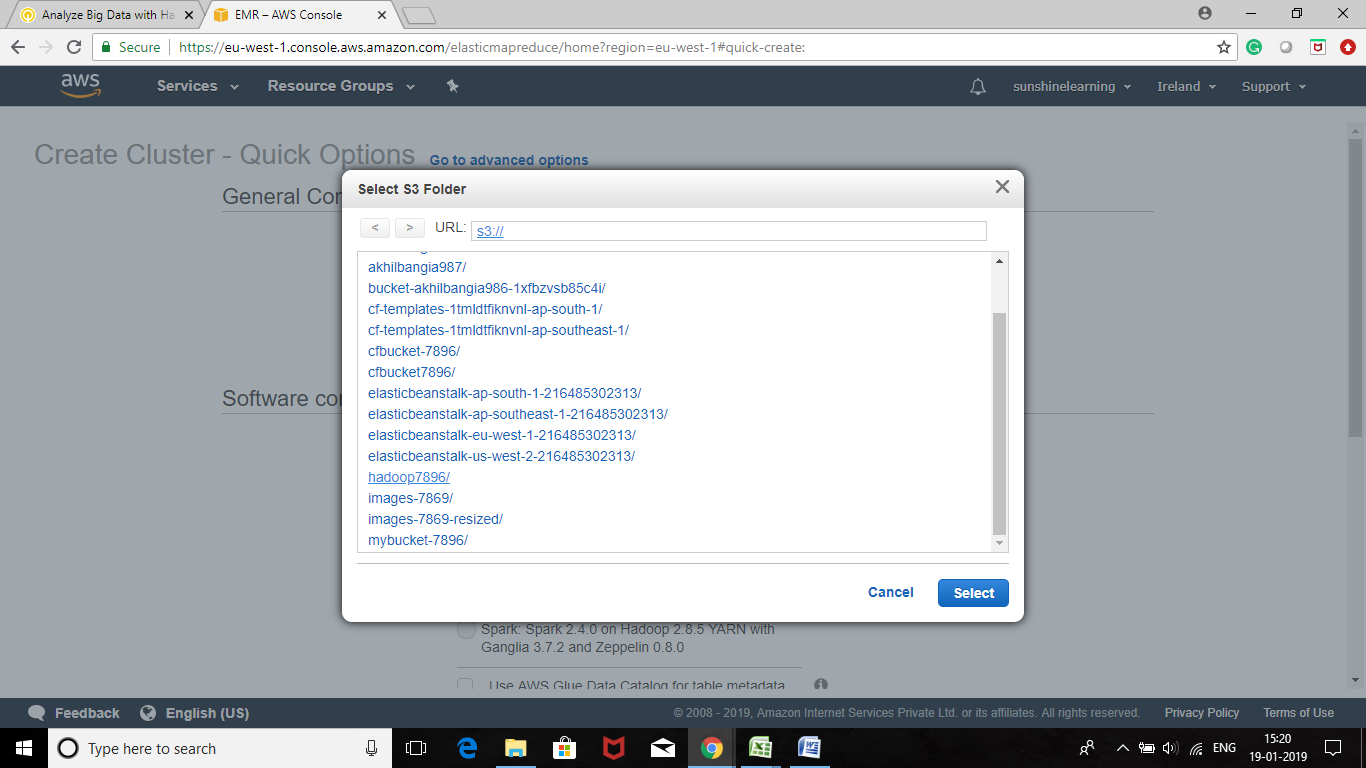
* **Permissions**: ()Custom
* **EMR role**: EMR DefaultRole
* **EC2 instance profile**: EMR EC2\_DefaultRole

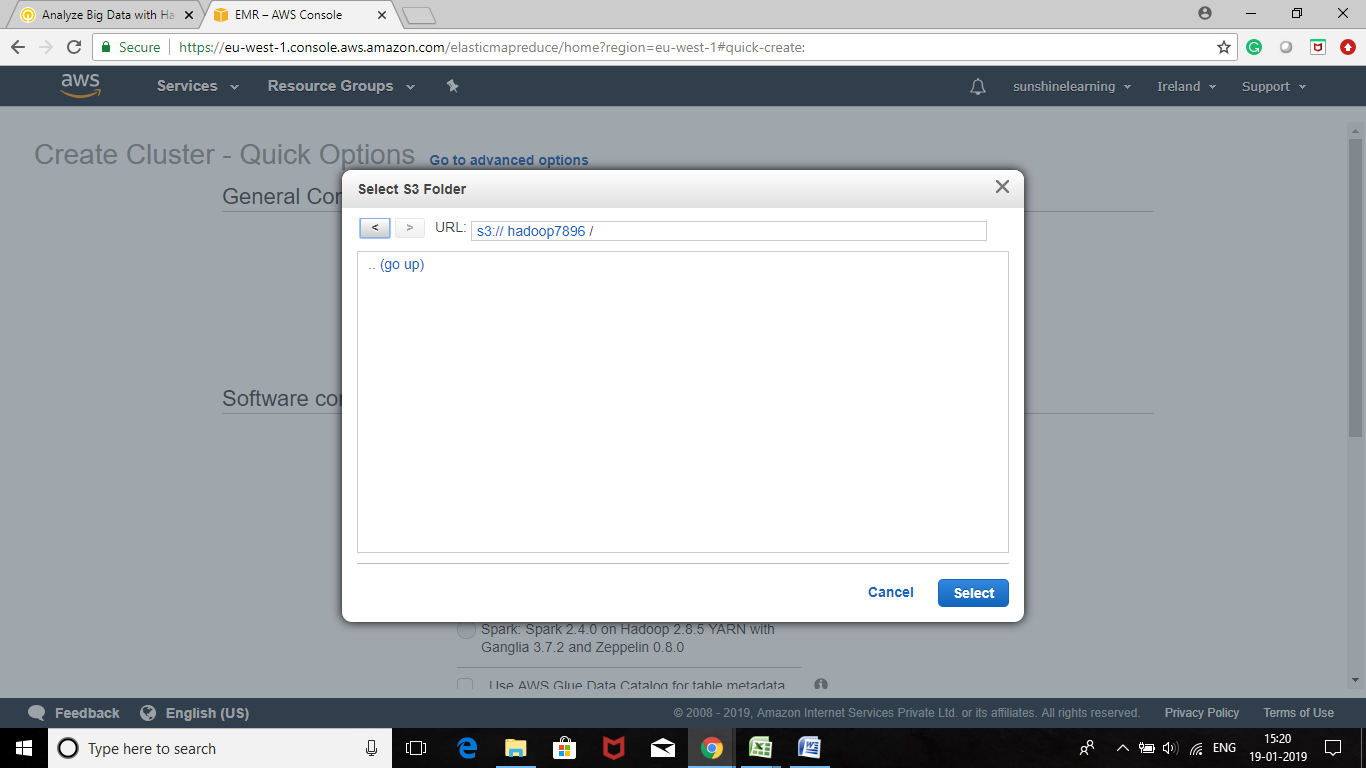
These roles will assign permissions to your cluster to access AWS services.

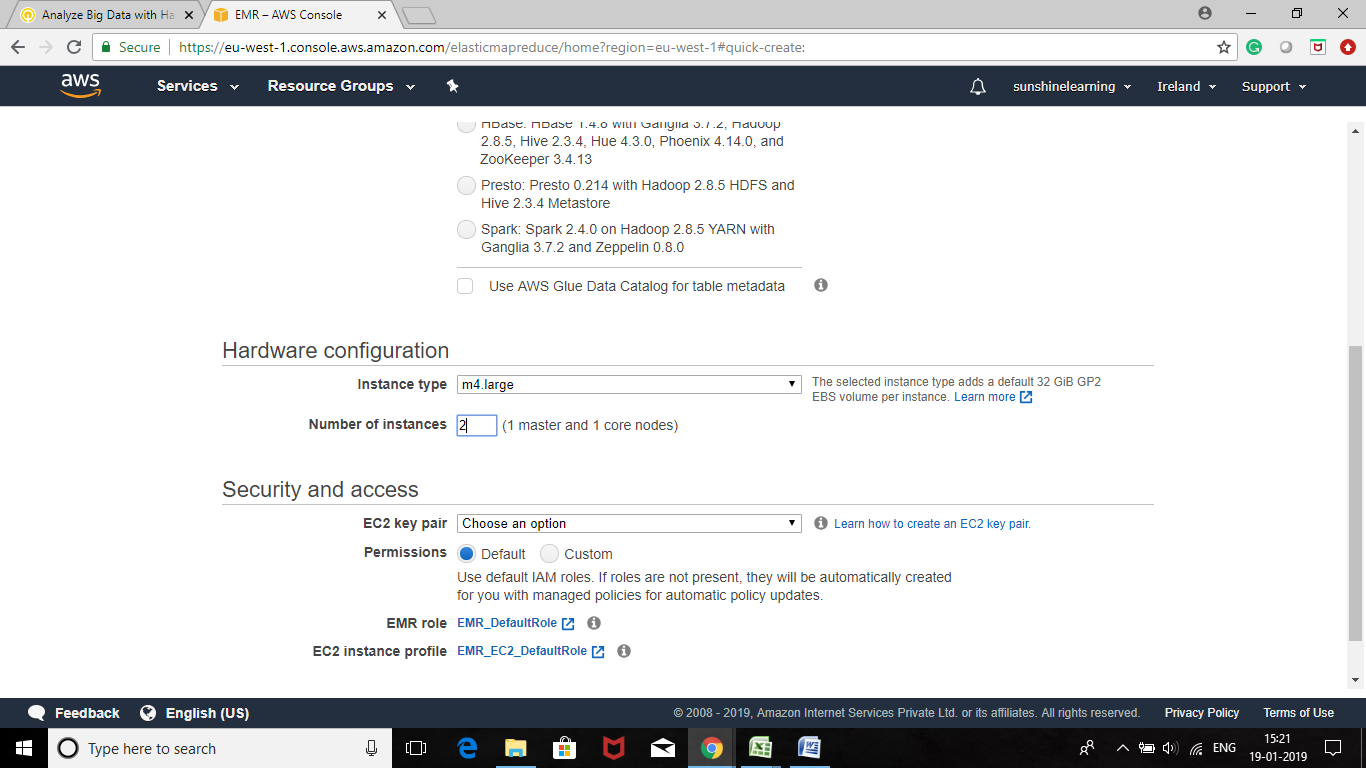
* Launch your EMR cluster by clicking Create cluster

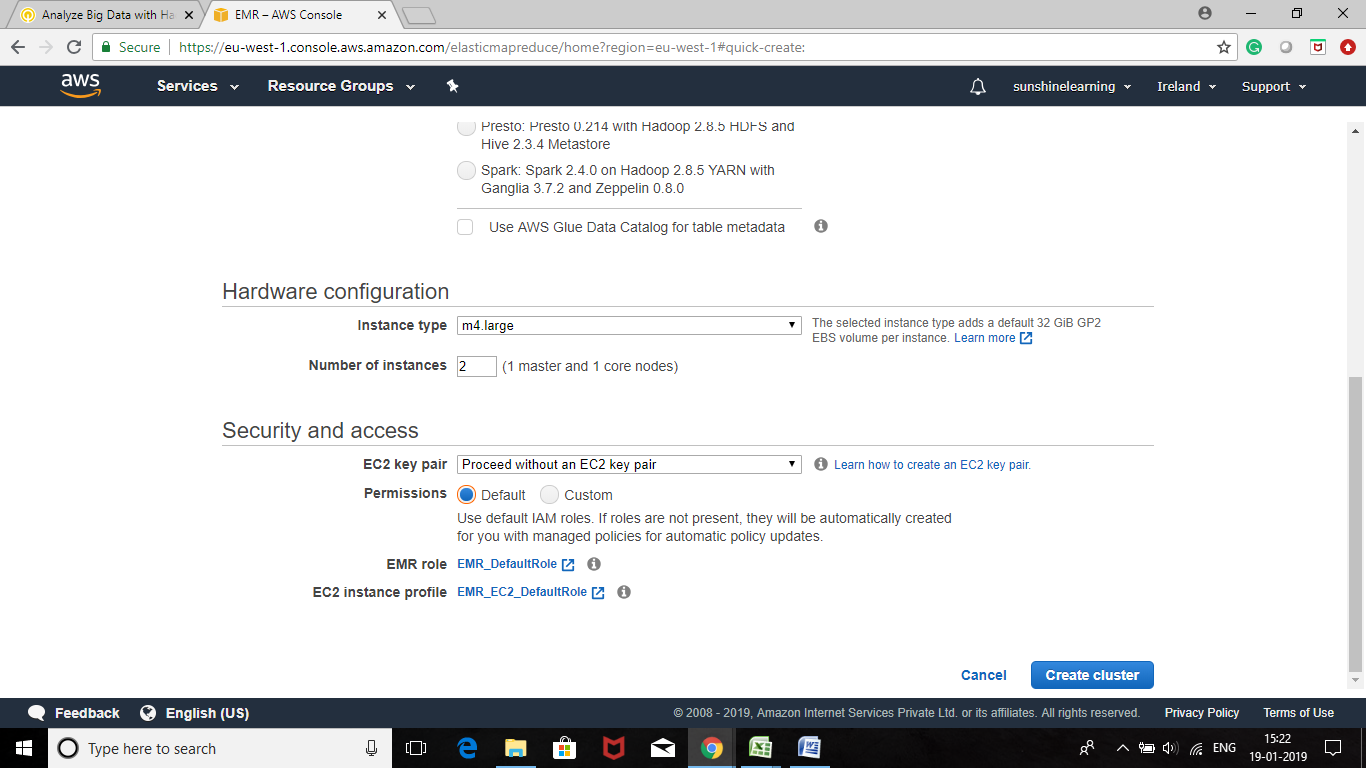
The cluster will take approximately five minutes to launch. Please continue reading while waiting for the cluster to launch.

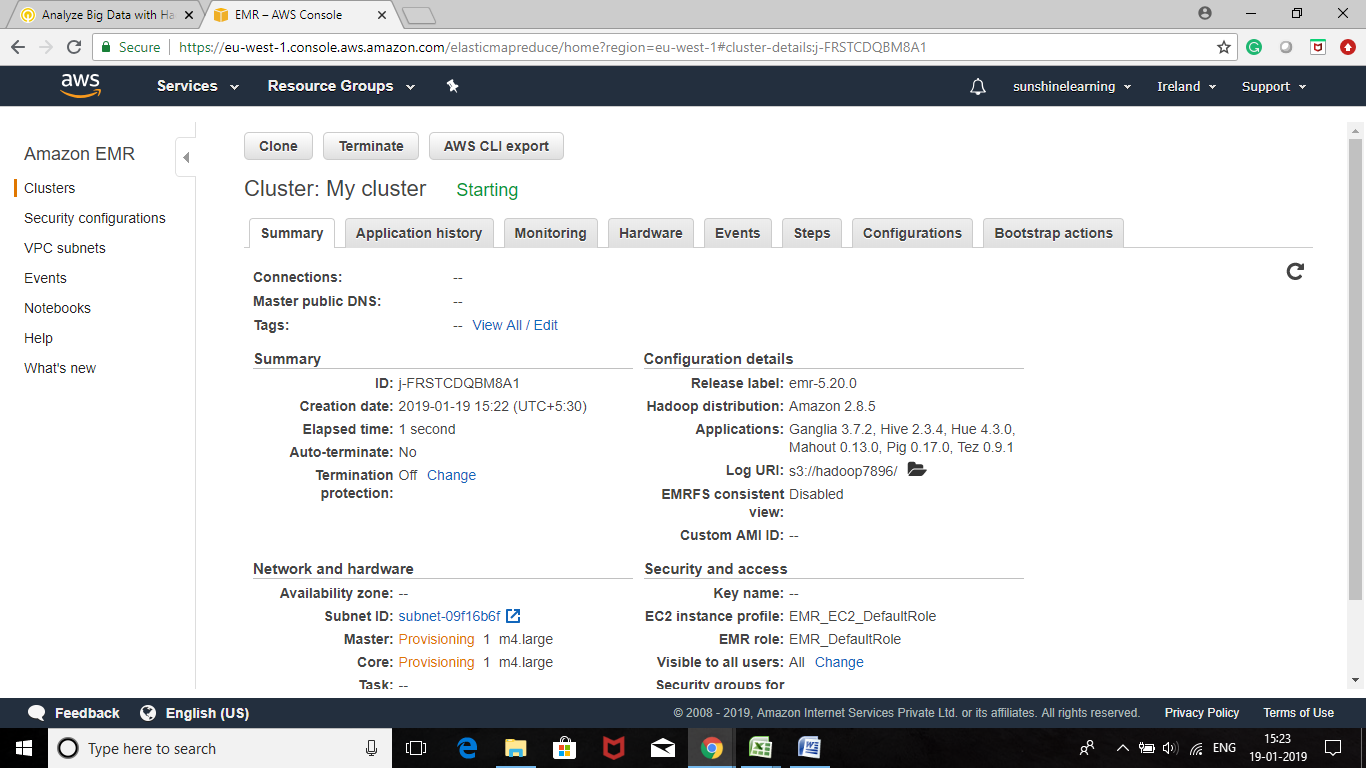












**Hadoop Applications**

The default configuration will automatically install several default applications on the cluster:

* **Apache Hadoop** is an open-source software project that can be used to efficiently process large datasets. Instead of using one large computer to process and store the data, Hadoop uses clusters of commodity hardware to analyze massive data sets in parallel.
* The **Ganglia** open-source project is a scalable, distributed system designed to monitor clusters and grids while minimizing the impact on their performance. Ganglia can generate reports and view the performance of the cluster as a whole, as well as inspect the performance of individual nodes.
* **Apache Tez** is a framework for creating a complex directed acyclic graph (DAG) of tasks for processing data. In some cases, it is used as an alternative to Hadoop MapReduce. For example, Pig and Hive workflows can run using Hadoop MapReduce or they can use Tez as an execution engine.
* **Hive** is an open-source data warehouse and analytic package that runs on top of a Hadoop cluster. Hive scripts use an SQL-like language called Hive QL (query language) that abstracts programming models and supports typical data warehouse interactions. Hive enables you to avoid the complexities of writing Tezjobs based on directed acyclic graphs (DAGs) or MapReduce programs in a lower level computer language, such as Java.
* **Hue** (Hadoop User Experience) is an open-source, web-based, graphical user interface for use with Amazon EMR and Apache Hadoop. Hue groups together several different Hadoop ecosystem projects into a configurable interface for your Amazon EMR cluster.
* **Pig** is an open-source Apache library that runs on top of Hadoop. The library takes SQL-like commands written in a language called Pig Latin and converts those commands into Tez jobs based on directed acyclic graphs (DAGs) or MapReduce programs. You do not have to write complex code using a lower level computer language, such as Java.

**Data to be Processed**

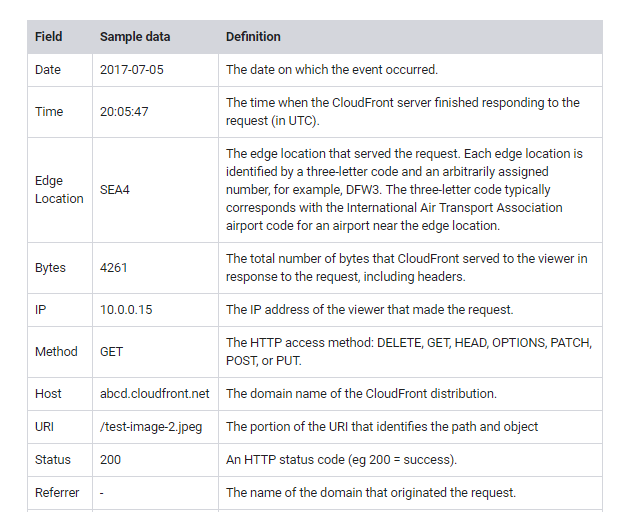
Once your cluster is ready, you will use the cluster to process log data from Amazon CloudFront.

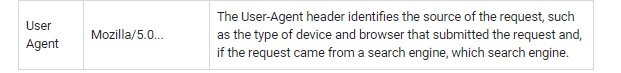
**Amazon CloudFront** is a web service that speeds up distribution of static and dynamic web content, such as .html, .css, .php, and image files. CloudFront delivers content through a worldwide network of data centers called edge locations. When a user requests content through CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance. If the content is already in the edge location with the lowest latency, CloudFront delivers it immediately. If the content is not in that edge location, CloudFront retrieves it from an Amazon S3 bucket or an HTTP server (for example, a web server) that you have identified as the source for the definitive version of your content.

Amazon CloudFront can produce access logs that show all data requested by users. The log files can grow very large, so Hadoop is an ideal way to process and analyze the log files.

Here is a sample of the log data:

2017-07-05 20:05:47 SEA4 4261 10.0.0.15 GET eabcd12345678.cloudfront.net /test-image-2.jpeg 200 - Mozilla/5.0%20(MacOS;%20U;%20Windows%20NT%205.1;%20en-US;%20rv:1.9.0.9)%20Gecko/2009040821%20Chrome/3.0.9





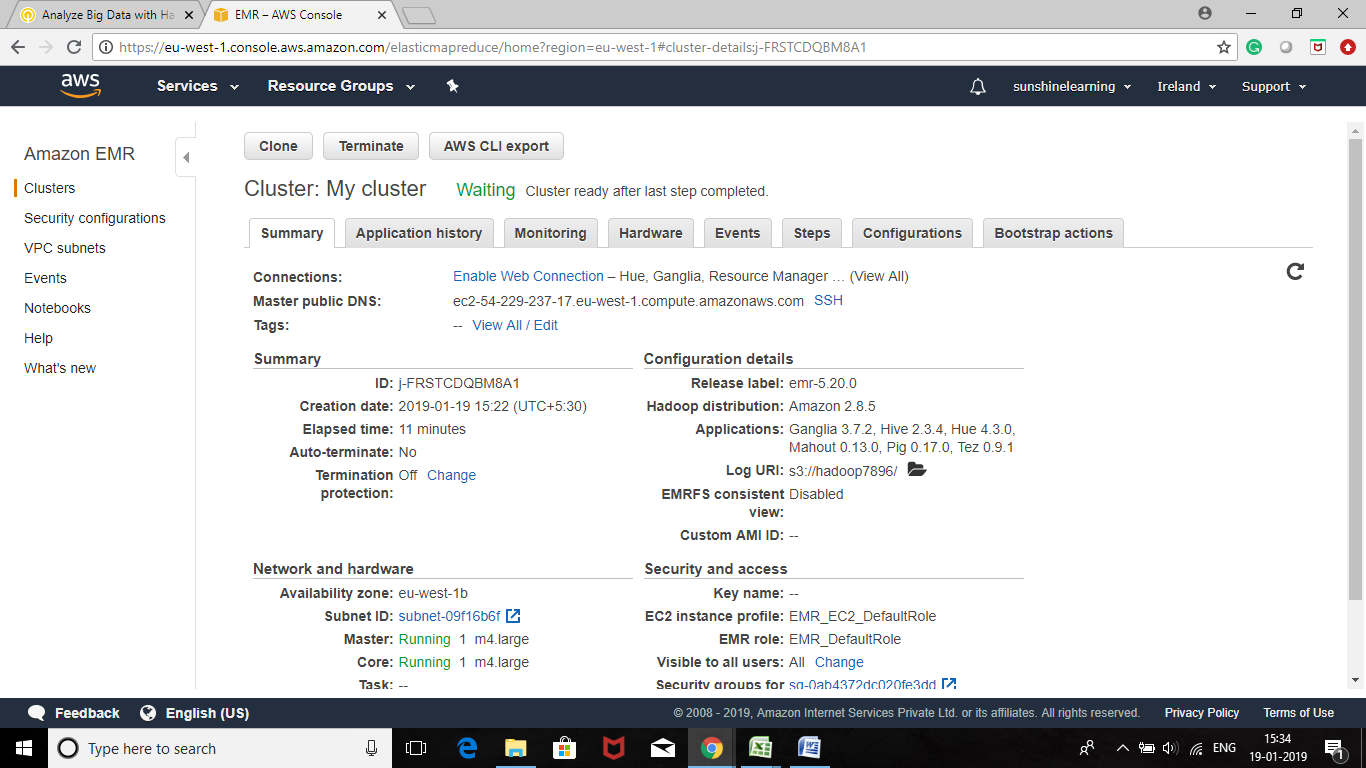
**Task 3: Process Your Sample Data by Running a Hive Script**

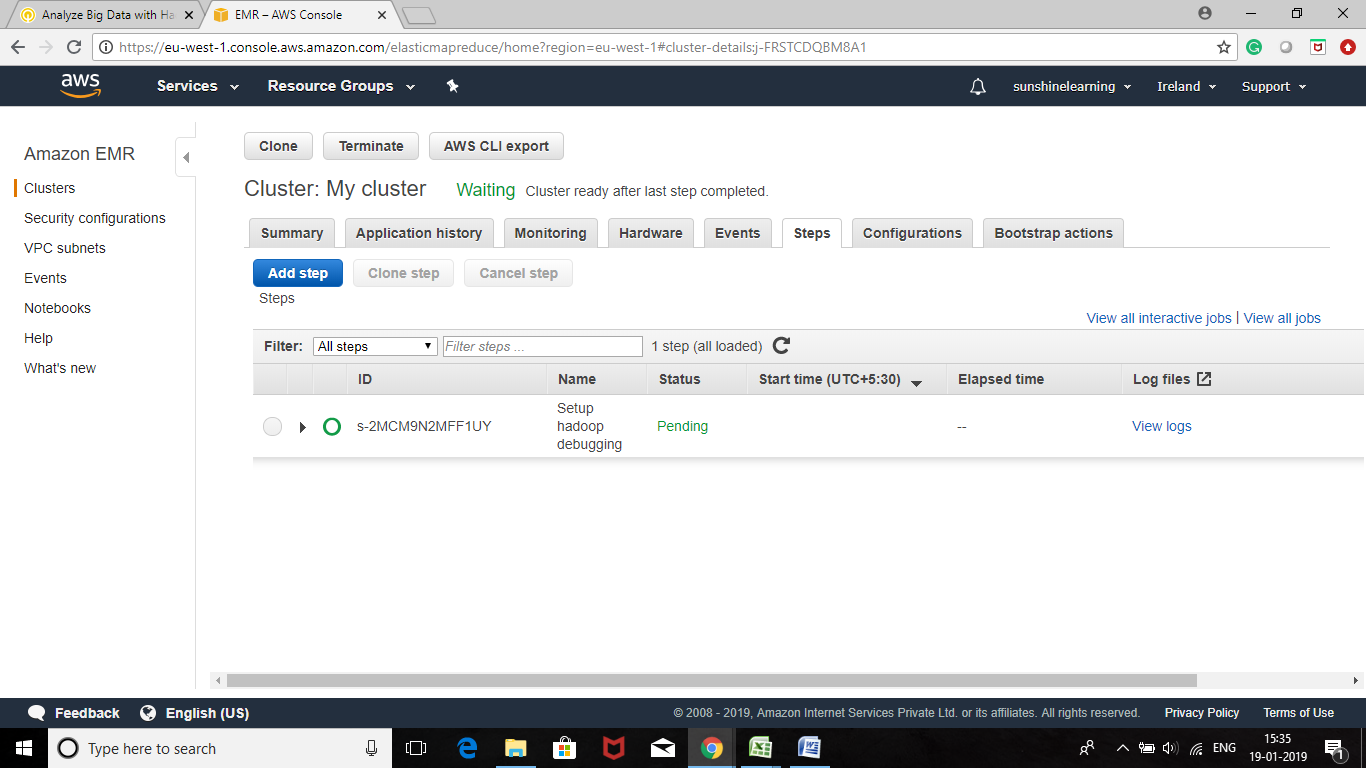
In this task, you will run a Hive script in your cluster as a step in the Amazon EMR console to process your sample data. In Amazon EMR, a **step** is a unit of work that contains one or more Hadoop jobs. You can submit steps when you create the cluster or when the cluster is running (if it is a long-running cluster).

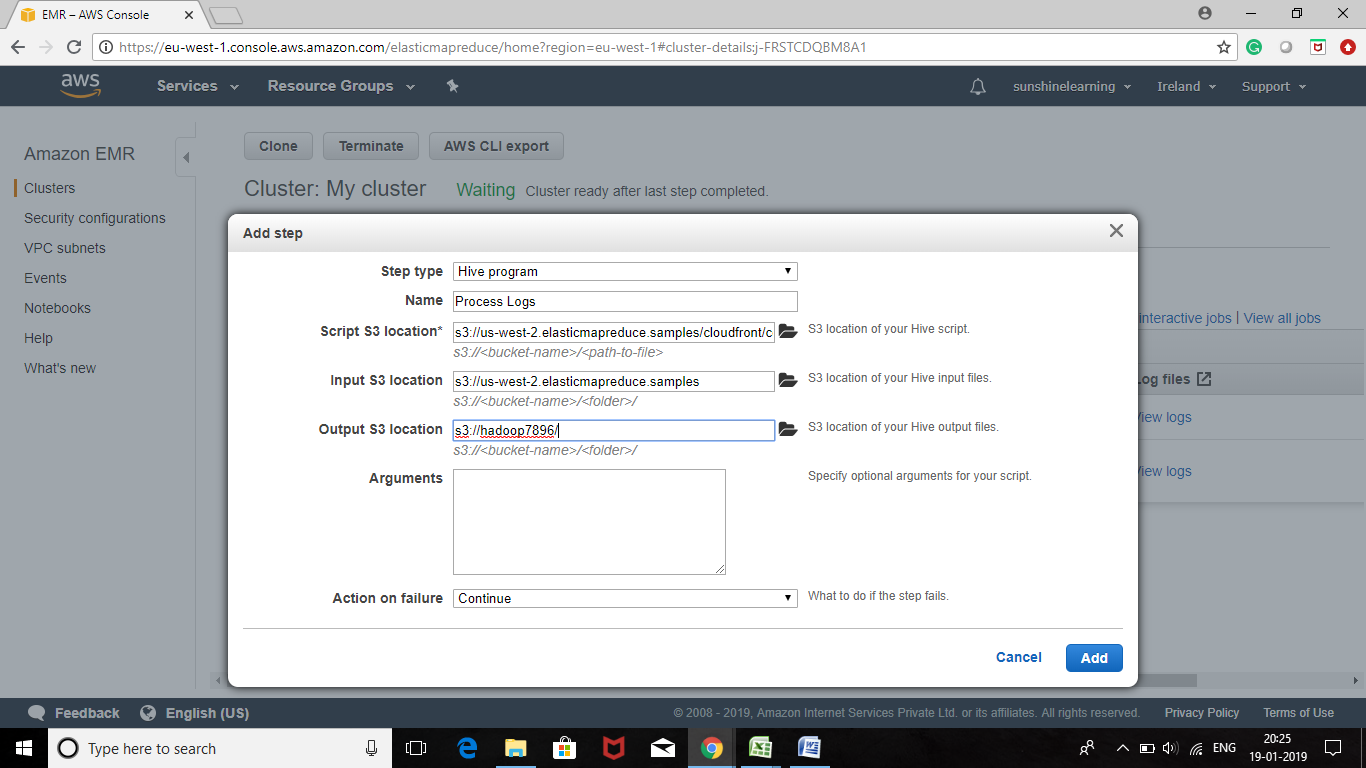
* Wait until your cluster is showing a status of **Waiting**.

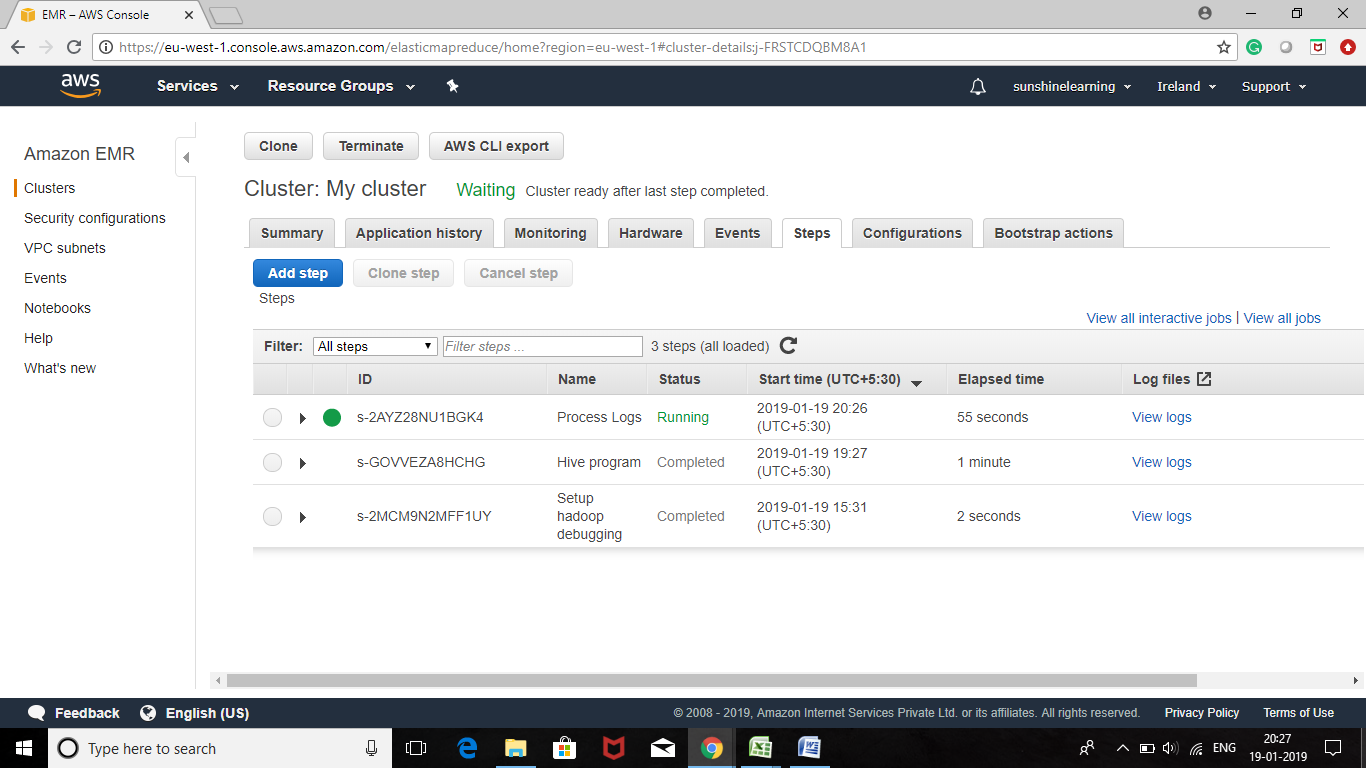
If your cluster is not yet in the Waiting state, wait a couple of minutes and click the refresh icon. Try again every few minutes until the cluster status is Waiting.

* Click the **Steps** tab.
* Click Add step then configure:
  + - **Step type**: Hive program
    - **Name**: Process logs
    - **Script S3 location**:









* Click Add

The status of the step changes from *Pending* to *Running* to Completed as the step runs.

The step takes approximately one minute to run.

**What the script is doing**

The Hive script does the following:

* + Creates a **Hive table** named cloudfront logs
  + Reads the **CloudFront log files** from Amazon S3 and parses the files using the Regular Expression Serializer/Deserializer (RegExSerDe).
  + Writes the parsed results to the *cloudfron\_ logs Hive table*
  + Submits a HiveQL query against the data to retrieve the **total requests per operating system** for a given time frame
  + Writes the query results to your Amazon S3 output bucket

The Hive code that creates the table is:

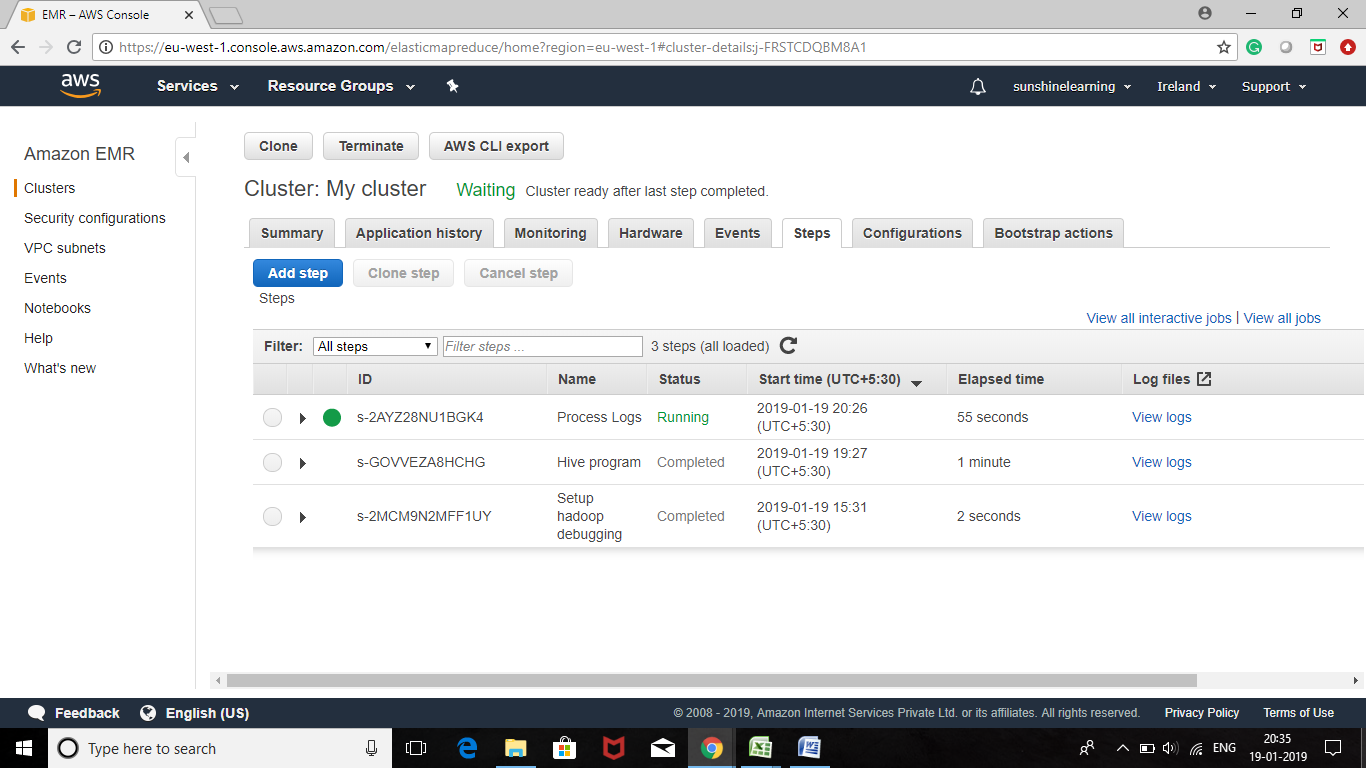
CREATEEXTERNALTABLEIFNOTEXISTScloudfront\_logs(DateObjectDate,TimeSTRING,LocationSTRING,BytesINT,RequestIPSTRING,MethodSTRING,HostSTRING,UriSTRING,StatusINT,ReferrerSTRING,OSString,BrowserString,BrowserVersionString)

The Hive code that parses the log files using the RegExSerDe looks line:

ROWFORMATSERDE'org.apache.hadoop.hive.serde2.RegexSerDe'WITHSERDEPROPERTIES("input.regex"="^(?!#)([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+([^ ]+)\\s+[^\(]+[\(](https://s3-us-west-2.amazonaws.com/us-west-2-aws-training/awsu-spl/spl-166/instructions/en\_us/[^\;]+).\*\%20([^\/]+)[\/](https://s3-us-west-2.amazonaws.com/us-west-2-aws-training/awsu-spl/spl-166/instructions/en\_us/.\*)$")LOCATION'${INPUT}/cloudfront/data/’;

The HiveQL query that calculates requests by operating system is:

INSERTOVERWRITEDIRECTORY'${OUTPUT}/os\_requests/'SELECTos,COUNT(\*)countFROMcloudfront\_logsWHEREdateobjectBETWEEN'2014-07-05'AND'2014-08-05'GROUPBYos;

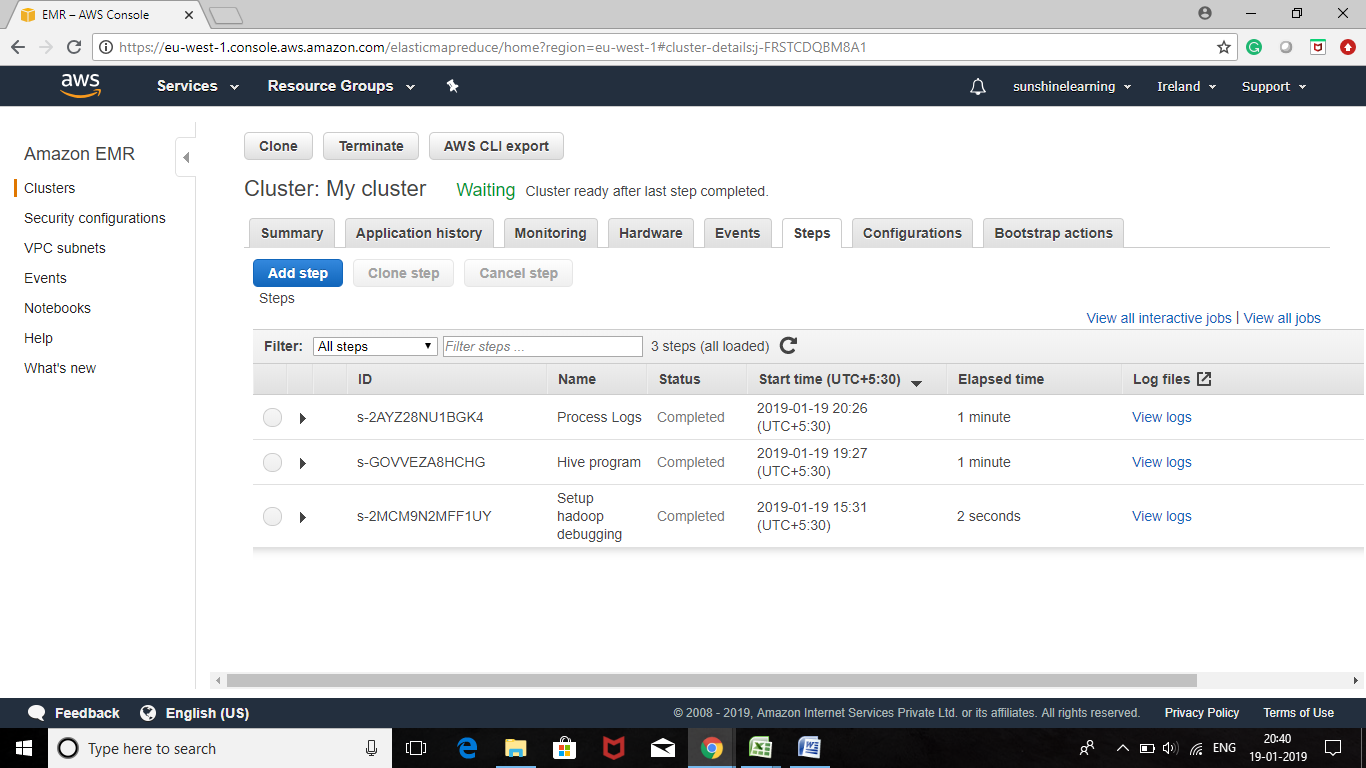


The sample data set contains approximately 5000 rows of data. This same process, however, can be used to process *millions of rows of data* in parallel across multiple nodes.

* Wait for the status of the step to change to Completed.

To update the status, click the refresh icon.

Your results are now available for viewing.



**Task 4: View the Results**

After the step completes successfully, the query output produced by the Hive script is stored in the Amazon S3 bucket that you specified when you submitted the step.

* On the **Services** menu, click **S3.**
* Click your *Hadoop* bucket.
* Click the **os\_requests**folder.

The Hive query results are stored in a text file.

* Select SI the *000000\_0* file.
* Click the download in the pop up box and save the file to your computer.

Open the file using a text editor such as WordPad (Windows), TextEdit (Mac OS), or gEdit (Linux).

In the output file, you should see the number of access requests by operating system:

|  |  |
| --- | --- |
| **Operating System** | **Count** |
| Android | 855 |
| Linux | 813 |
| MacOS | 852 |
| OSX | 799 |
| Windows | 883 |
| iOS | 794 |

This lab used a small sample of data. Normally, there would be significantly more rows of data in your CloudFront logs. Regardless of the size, the cluster automatically distributes the work across the cluster to complete the work required.

