Lab 2 - Spark & Big Data

Luca Telloli
luca.telloli@upf.edu

Pablo Martín Castagnaro
pablo.castagnaro@upf.edu

Summary

- Considerations on Lab 1
- Spark
- Elastic Map Reduce (EMR)
- Running Spark on EMR
- Example: Running Word Count on EMR

Learnings of Lab 1

- Java: review I/O (read & write files), basic Exceptions
- Parsing JSON to an instance of a Java Class and vice-versa
- S3:
  - with the CLI or Web UI: create buckets, change permissions
  - programmatically with the SDK: S3Uploader
-

Considerations on Lab 1

- Read all data in a single node <- does not scale well
- Read all data sequentially <- multi-thread programming? difficult & still limited
- Processing is a small part of the program, still it's the relevant part
- I/O is a considerable part of the application and yet is not the focus, while being mostly tedious & repetitive

Spark review

Review

- RDDs - Like a big table, broken into parts, each part available on a separate node
- Functional operators in Spark: continuous transformation of a RDD<T> to a RDD<U> using functional operators
  - Transformations & Actions
- Spark execution model: only Actions operation trigger a materialisation (data physically present on a storage level).
- Programming guide: https://spark.apache.org/docs/2.4.7/rdd-programming-guide.html

Installing Spark locally

If you install Spark locally (*highly recommended*), please use the latest stable version.

Check your installed version with `spark-submit --version`

Spark Applications in Java

- A Spark application is a Java application that uses the Apache Spark libraries
- In a new Spark application, create first a `SparkConfiguration` and pass it to a `JavaSparkContext`, as displayed in this example:

```
SparkConf conf = new SparkConf().setAppName(<your app name>);

JavaSparkContext sc = new JavaSparkContext(conf);
```

An example application for WordCount is available on Moodle, and can be used a starting point for your applications.

Running spark applications

It's frequent during the development phase to run a Spark application locally, usually on a smaller subset of the input.

To run the application locally, you won't use Java; instead, you'll use the Spark framework, accessible with the spark-submit command. The command will be similar to:

```
spark-submit --master <MASTER> --class <MAIN CLASS> your.jar arg1 ...  argN
```

Common spark functions

`.map(elem-> ...)`	
	Applies a transformation 1 to 1 to each element
`.mapToPair(elem-> transformation(elem))`	
	Applies a transformation 1 to 1 to each element. Generates an entry of a PairRDD for each of them
`.flatMap(elem-> transformation(elem))`	
	Applies a transformation 1 to N to each element. I**f the resulting type is a list, it requires an iterator** (see WordCount example for that)
`.filter(elem->condition(elem))`	
	Selects all the elements satisfying a condition.
`.reduceByKey(key1, key2->...)`	
	Groups and transforms all the values from rows having identical keys (requires a PairRDD)

Common spark functions

JavaRDD<V>	JavaPairRDD<K,V>
`sort([`*ascending=True*`])`	`sortByKey([`*ascending=True*`])`
`take()` `Materializes N elements from a given RDD`	`take(N)` `Materializes N elements from a given PairRDD`
`join()`	`join()` Returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key.

Items in a JavaPairRDD are of type scala.Tuple2

Spark does not have a sortByValue() function. If you want to do so, you need to swap ((K,V) to (V,K)) tuples in the RDD, sortByKey and swap again. See the function swap() of the Tuple2 class.

Example

Running WordCount locally

Download it from moodle

Elastic Map Reduce (EMR)

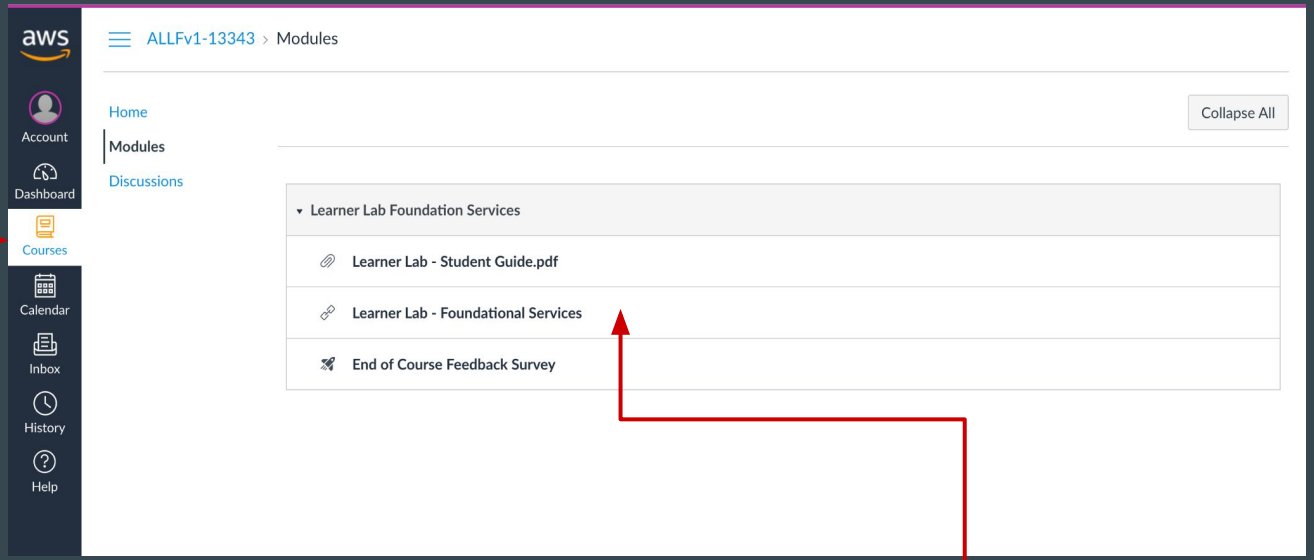
What is EMR ?

- AWS EMR is a tool tool for big data processing and analysis
- Supports processing of large datasets in a distributed computing environment
- AWS EMR offers expandable, low-configuration service
- Based on Apache Hadoop (MapReduce) cluster of virtual servers on AWS EC2 and AWS S3

Running Spark on EMR

Access to AWS Academy and its resources

0. Check your email and register in AWS Academy with the provided link



1. Click on Courses to access the AWS resources

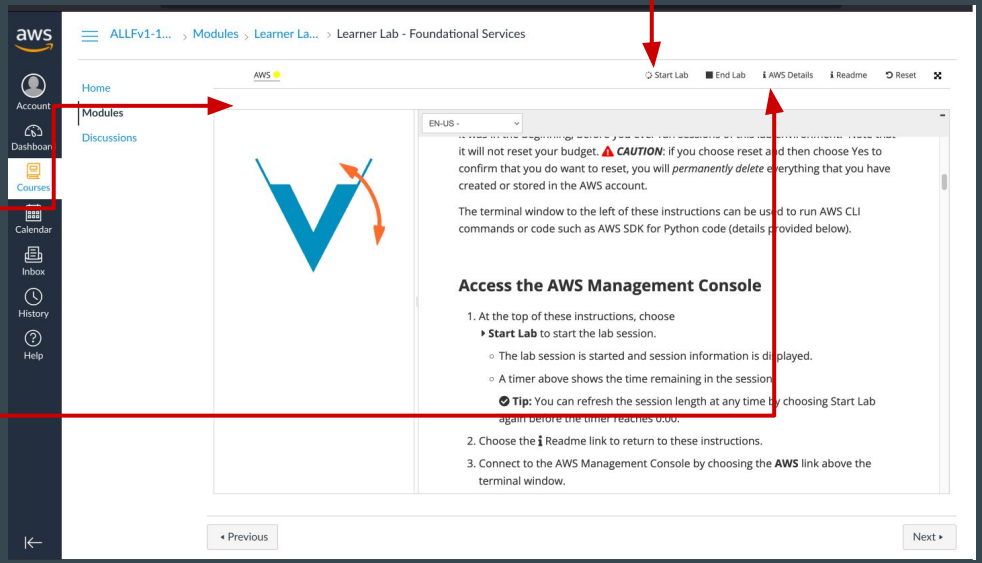
2. Click on Foundational Services and access terms and conditions

Access to AWS Academy and its resources

3. Click on Start Lab to access the AWS resources

4. Wait until your lab is ready

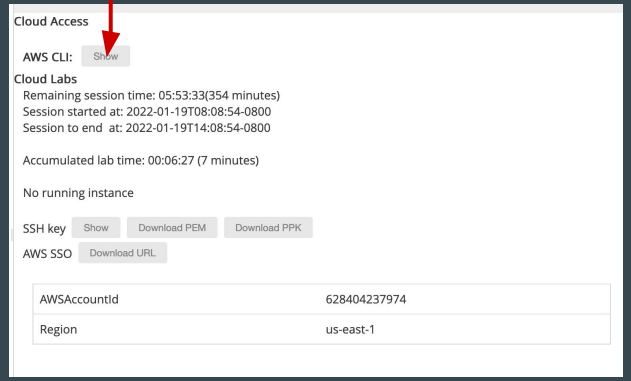
5. Click AWS Details to access your account credentials



What does AWS Academy offer?

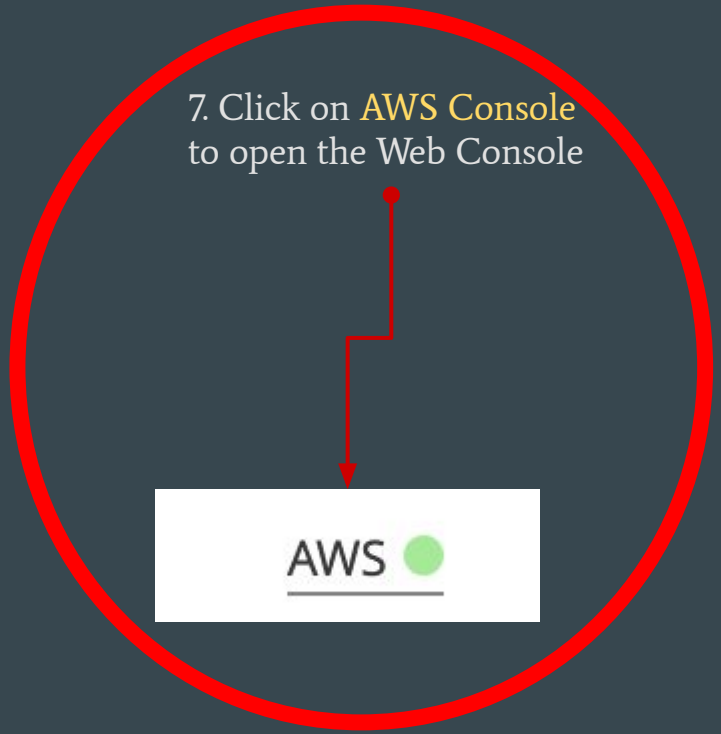
- "Student" version of the AWS console
- Pre-loaded with some expendable credit ($100)
- The CLI credentials expire every 3 hours

6. Click on Show to get CLI Credentials



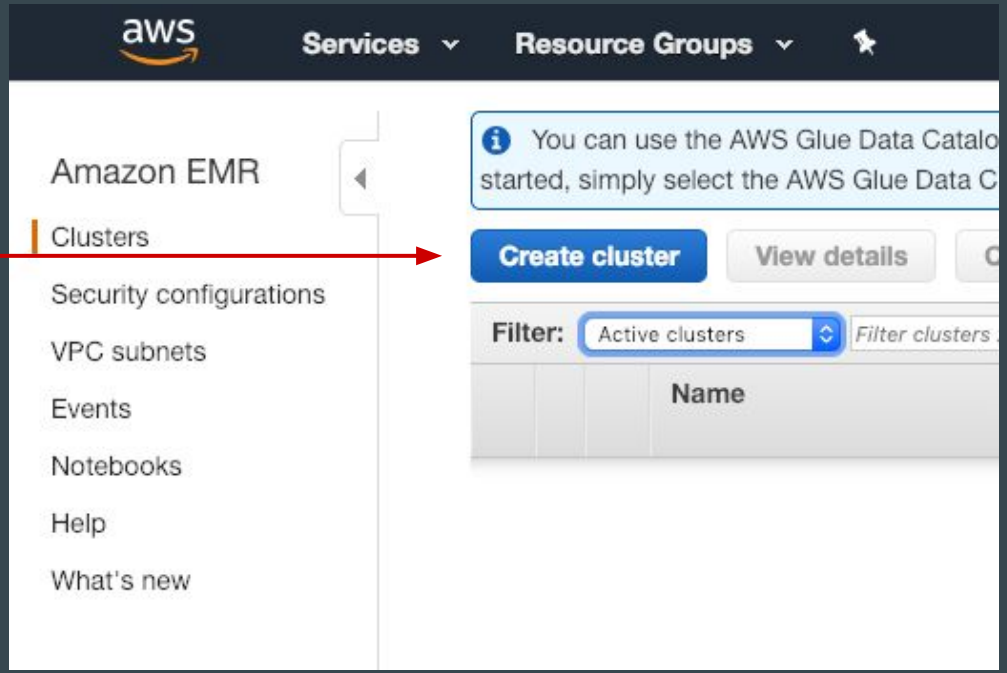
7. Click on AWS Console to open the Web Console

AWS console



Setting up a Cluster on AWS

- Login into the AWS console through your Educate account
- Search or browse for the EMR service (Elastic Map Reduce)
- Select: *Create Cluster*
- It's better to create your first cluster immediately, because the first run can take longer



aws Services Resource Groups vocstartsof

Create Cluster - Quick Options

Go to advanced options

General Configuration

Cluster name: My cluster

Logging

S3 folder: s3://aws-logs-945937867754-us-east-1/elasticmapreduce/

Launch mode: Cluster / Step execution

Software configuration

Release: emr-5.17.0

Applications:

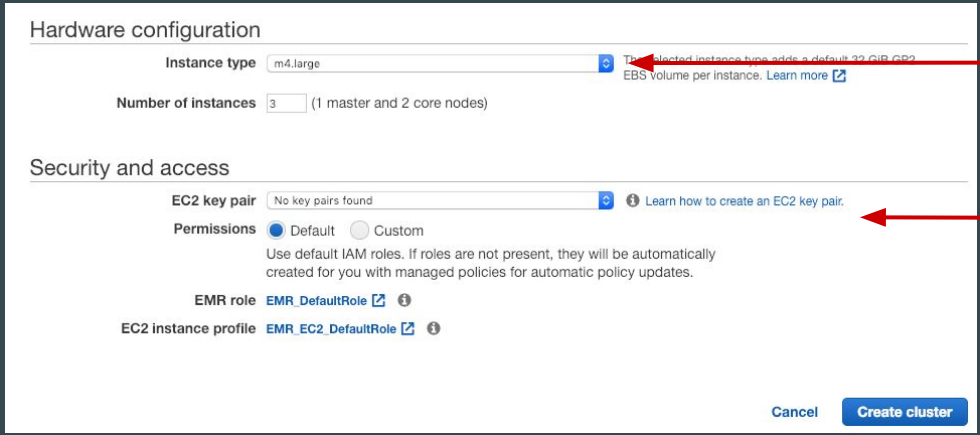
- Core Hadoop: Hadoop 2.8.4 with Ganglia 3.7.2, Hive 2.3.3, Hue 4.2.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.8.4
- HBase: HBase 1.4.6 with Ganglia 3.7.2, Hadoop 2.8.4, Hive 2.3.3, Hue 4.2.0, Phoenix 4.14.0, and ZooKeeper 3.4.12
- Presto: Presto 0.206 with Hadoop 2.8.4 HDFS and Hive 2.3.3 Metastore
- Spark: Spark 2.3.1 on Hadoop 2.8.4 YARN with Ganglia 3.7.2 and Zeppelin 0.7.3

Use AWS Glue Data Catalog for table metadata

1. Give it a name

2.Use `emr-5.17.2`

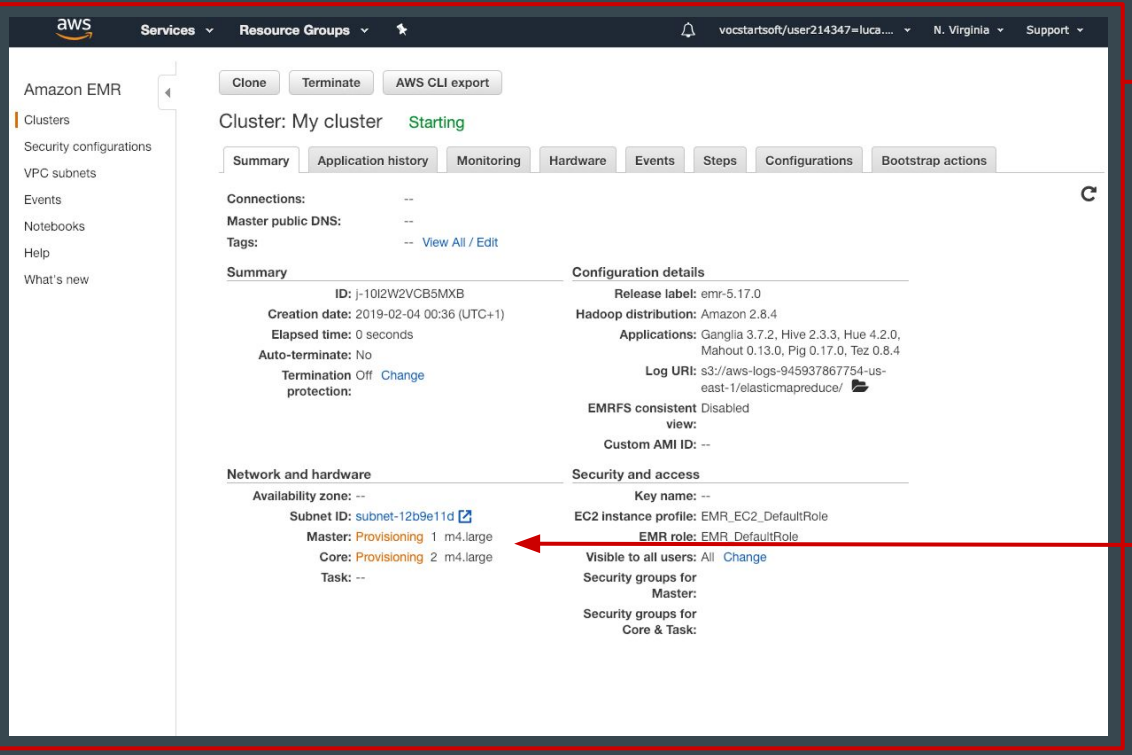
3. Select Spark support



4. Select `m4.large` instances

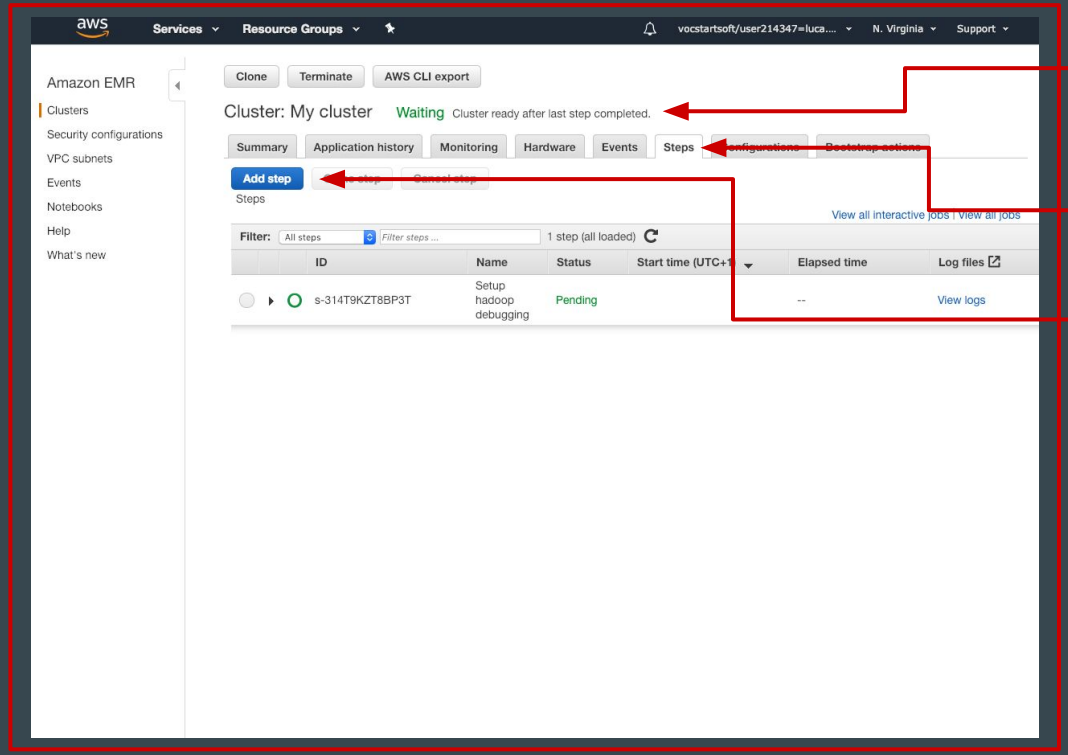
5. If you want, you can configure here an EC2 key pair to access other interfaces with more information about your cluster

6. Press *Create Cluster*



You'll find yourself here

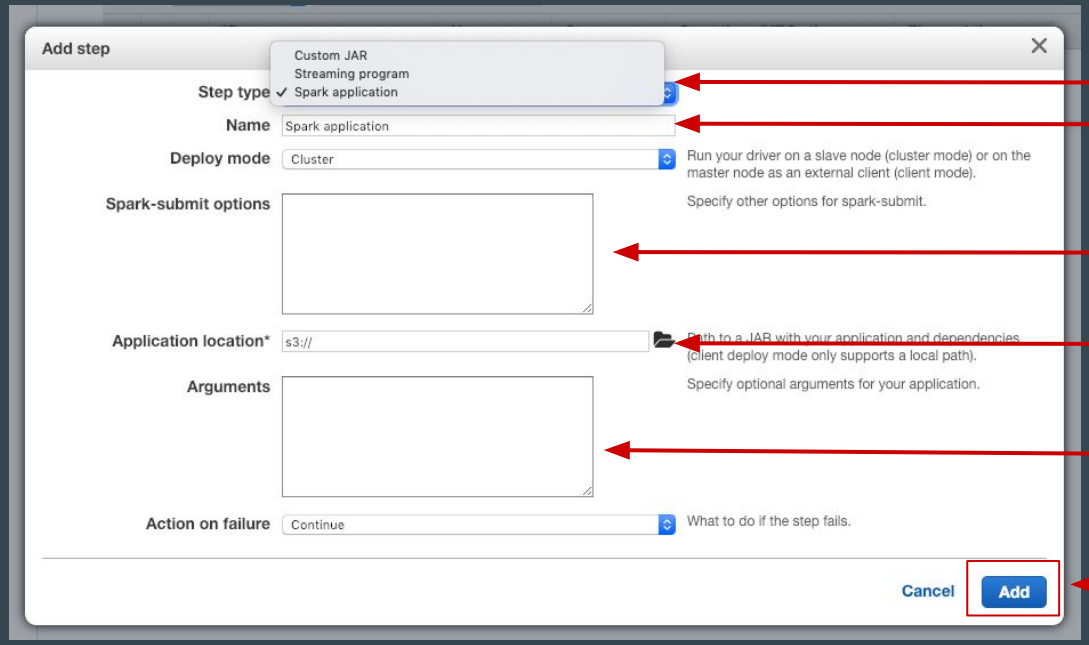
Your cluster is currently provisioning, it could take some minutes and maybe the first time it might need to wait for an authorisation: it should be a matter of minutes



After bootstrapping the cluster is green and waiting to run applications.
Remember to stop it after you finish!

Select the *Steps* tab

Select *Add step* to configure a Spark application



Select *Spark Application*

Provide a name

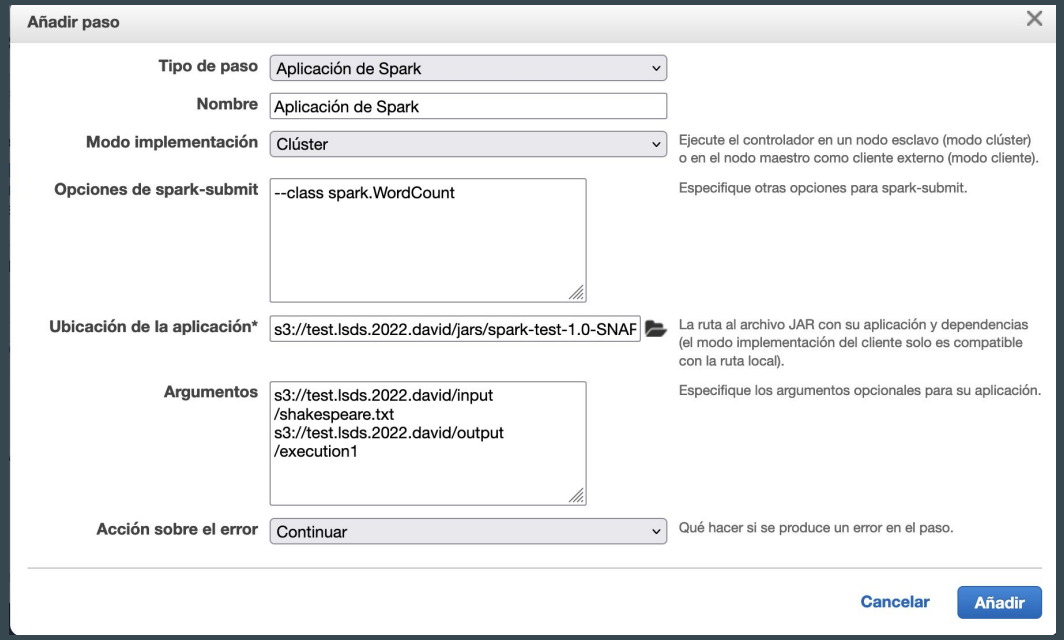
The options are the same that you'd give to `spark-submit` on the command line

Location on S3 where you put your JAR

The arguments are the same that you'd give on the command line

Go!

Example of parameters configuration to execute the WordCount



```
luke@rameau:$ aws s3 ls
s3://lsds2018-lab2/output/
2019-02-04 20:56:03          0 _SUCCESS
2019-02-04 20:56:02      25436 part-00000
2019-02-04 20:56:02      25185 part-00001
2019-02-04 20:56:02      25724 part-00002
2019-02-04 20:56:02      25239 part-00003
2019-02-04 20:56:03      25095 part-00004
2019-02-04 20:56:03      25402 part-00005
2019-02-04 20:56:03      25684 part-00006
2019-02-04 20:56:03      25083 part-00007
2019-02-04 20:56:03      25896 part-00008
2019-02-04 20:56:03      26084 part-00009
2019-02-04 20:56:03      24749 part-00010
2019-02-04 20:56:03      25763 part-00011
2019-02-04 20:56:03      25568 part-00012
2019-02-04 20:56:03      25193 part-00013
2019-02-04 20:56:03      25865 part-00014
2019-02-04 20:56:03      26138 part-00015
```

This is a typical output from an hadoop job, notice the _SUCCESS empty file that denotes the success of the execution

Example

Running WordCount on EMR

Part 1: Run the application locally

Download the project “WordCount Spark Example” from AulaGlobal.

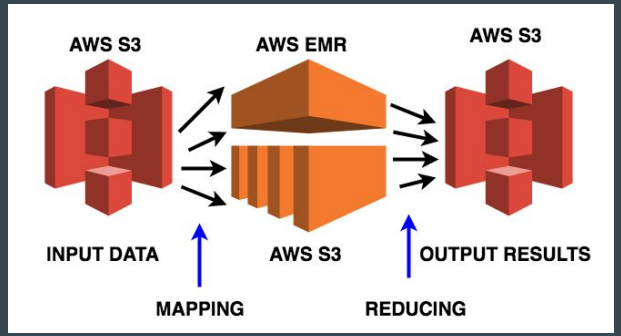
Use `mvn package` to generate an executable JAR from it

Download Shakespeare’s work from here

Use `spark-submit` to run locally the generated JAR for the downloaded data

How many times does the word ‘trojan’ appear?

Part 2: Now on EMR



- Create a bucket for the experiment
- Upload the generated jar and the text file
- Go to your EMR cluster:
  1) Add step
     a) Set JAR path to S3
     b) Set arguments
        i) Input file path
        ii) Output file path
  2) Wait for execution

Can you see your results on S3?