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

What is an RDD?

- **Resilient Distributed Dataset:** Like a big table, broken into parts, each part available on a separate node
- It’s an immutable distributed collection of objects
- Spark provides **two types** for such datasets

Resilient	Can recover from abnormal events (i.e.: hardware, network issues)
Distributed	Data might be held at different times in different storage elements (memory, disks, nodes), but it’s still managed by the user as a single continuous sequence of items.

What is an RDD?

RDDs can be of two families
They can contain any Java class (`RDD<Int>`, `RDD<User>`, etc...)

`JavaRDD<V>`	`JavaPairRdd<K,V>`
A dataset with a single type of value	A dataset with two elements, a key, and a value

Row 1

Row 2

Row n

```
V1
--------------
V2
--------------
Vn
```

```
K1    |    V1
--------------
K2    |    V2
--------------
Kn    |    Vn
```

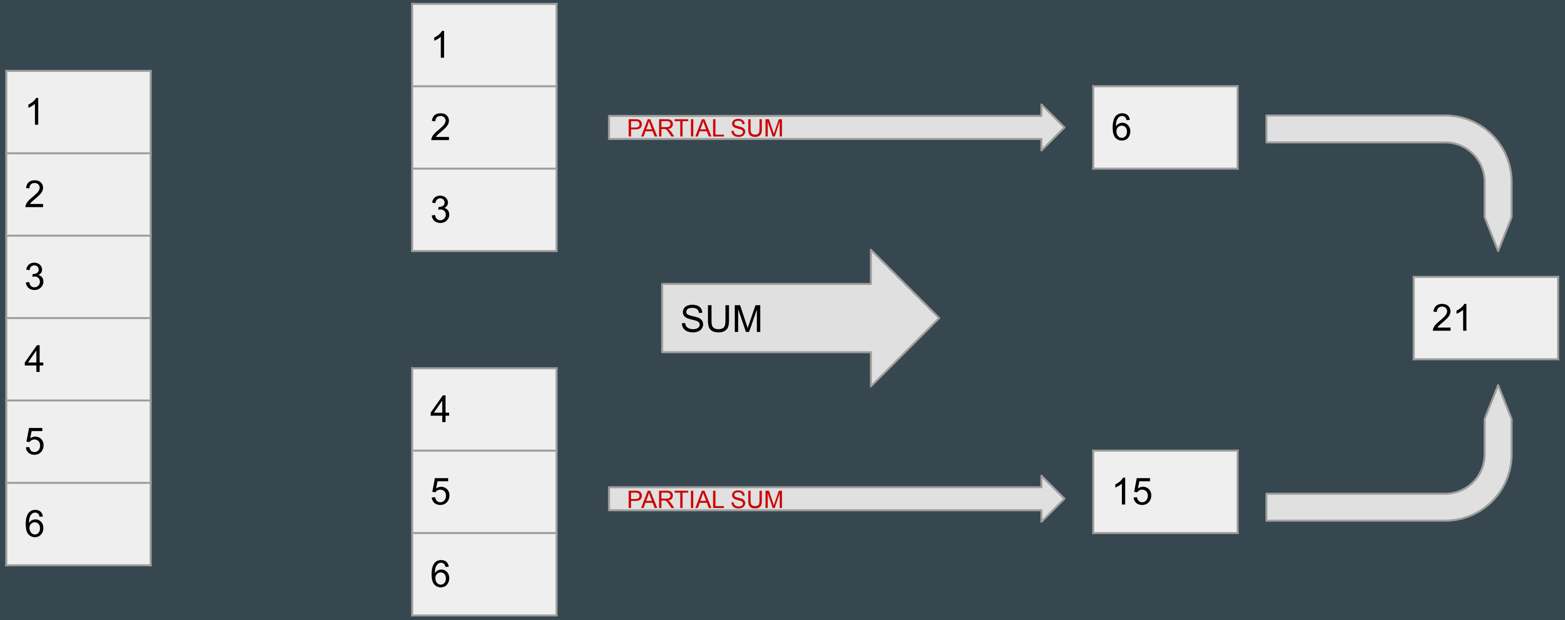
What is an RDD?

- Transformations on RDDs are defined differently depending on the type

	JavaRDD<V>	JavaPairRdd<K,V>
Mapping:	map()	mapValues()
Reducing:	reduce()	reduceByKey()
Flattening:	flatMap()	flatMapValues()

What is an RDD?

Spark takes care of **orchestrating** the sequence of distributed operations



USER VIEW

EFFECTIVE STORAGE

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)

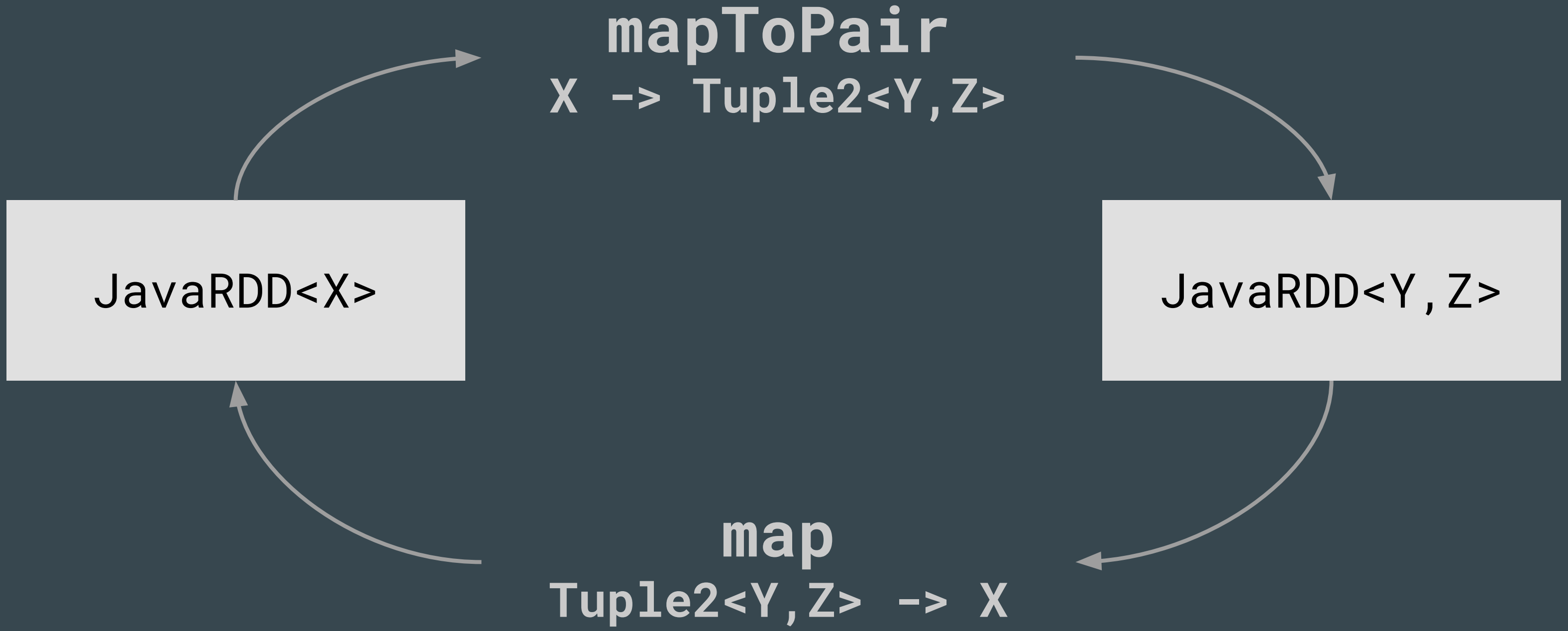
```
.filter(elem->condition(elem))
```

Selects all the elements satisfying a condition.

```
.reduceByKey(key1, key2->...)
```

Groups and transforms all the values from rows having identical keys (requires a PairRDD)

Moving between RDD<K> and RDD<K,V>



mapToPair

X -> Tuple2<Y,Z>

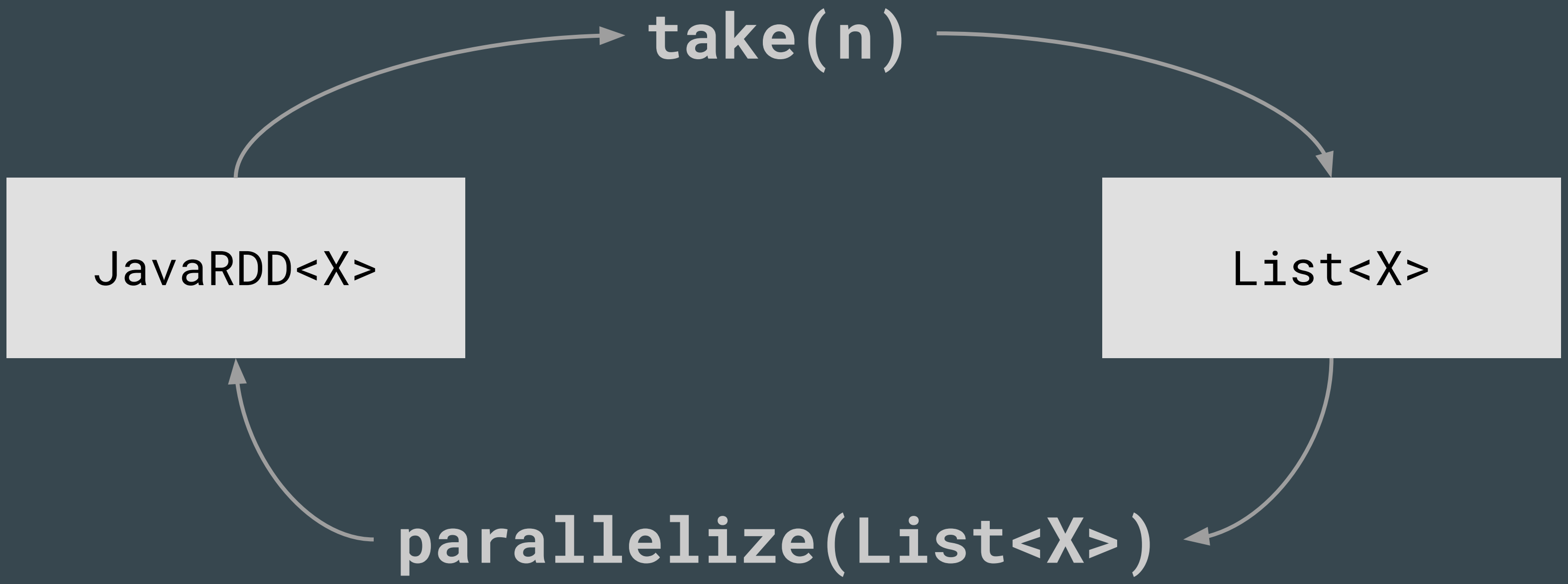
JavaRDD<X>

JavaRDD<Y,Z>

map

Tuple2<Y,Z> -> X

Moving between RDDs and Lists



`take(n)`

`JavaRDD<X>`

`List<X>`

`parallelize(List<X>)`

This method is available in the Spark context

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.

Review

- 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
```

Example

Running WordCount locally

Download it from moodle

Elastic Map Reduce (EMR)

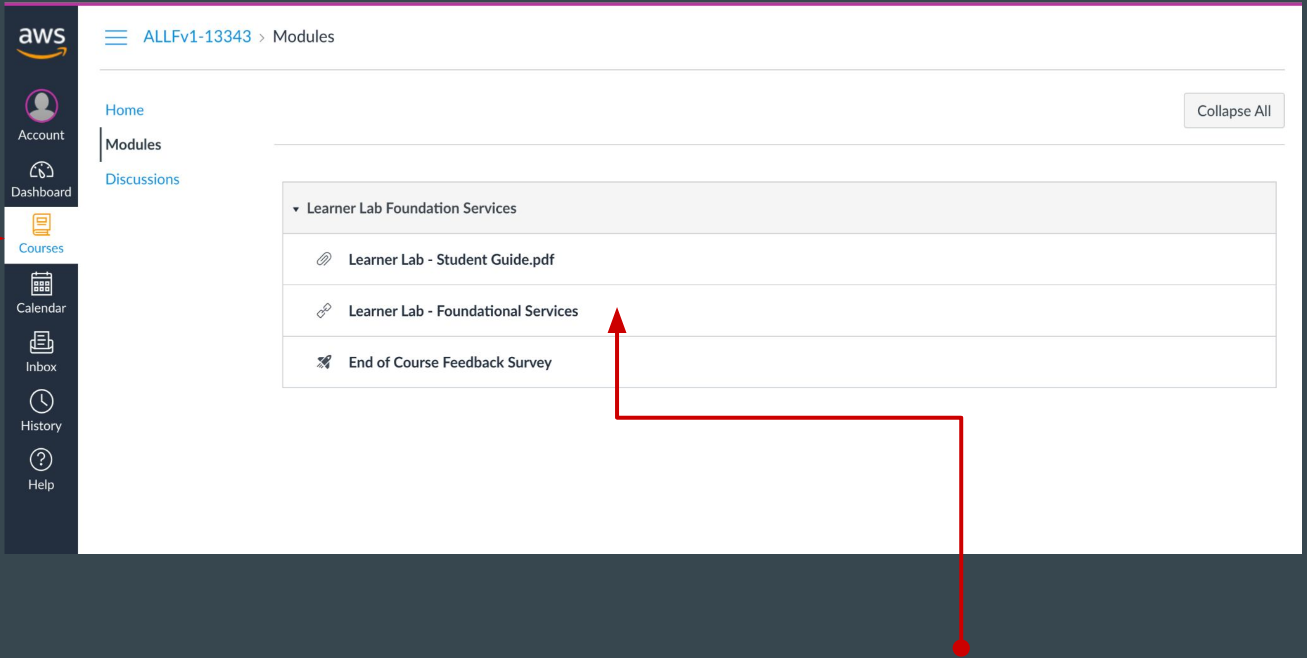
What is EMR ?

- AWS EMR is a tool for big data processing and analysis
- Supports processing of large datasets in a distributed computing environment
- AWS EMR offers expandable, low-configuration, managed 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



▾ Learner Lab Foundation Services

- Learner Lab - Student Guide.pdf
- Learner Lab - Foundational Services
- End of Course Feedback Survey

1. Click on Courses to access the AWS resources

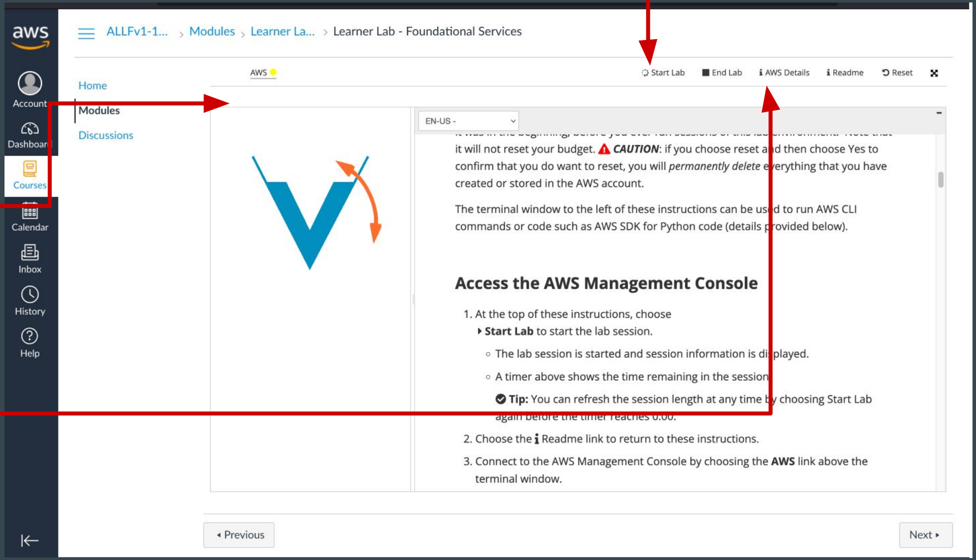
2. Click on Foundational Services and access terms and conditions

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



it will not reset your budget. ⚠ ***CAUTION***: if you choose reset and then choose Yes to confirm that you do want to reset, you will *permanently delete* everything that you have created or stored in the AWS account.

The terminal window to the left of these instructions can be used to run AWS CLI commands or code such as AWS SDK for Python code (details provided below).

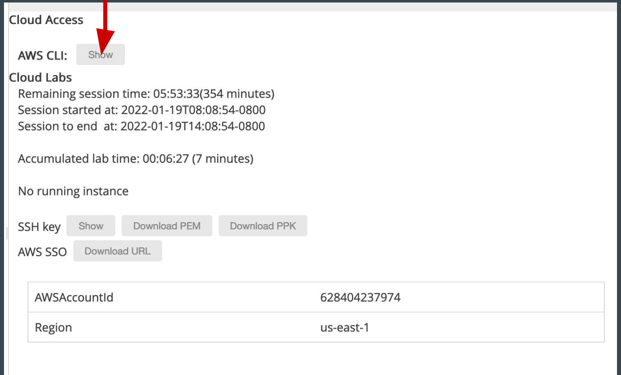
Access the AWS Management Console

1. At the top of these instructions, choose
   ▸ **Start Lab** to start the lab session.
   - The lab session is started and session information is displayed.
   - A timer above shows the time remaining in the session
   ✓ **Tip:** You can refresh the session length at any time by choosing Start Lab again before the timer reaches 0:00.
2. Choose the ℹ Readme link to return to these instructions.
3. Connect to the AWS Management Console by choosing the **AWS** link above the terminal window.

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



Cloud Access

AWS CLI: Show

Cloud Labs

Remaining session time: 05:53:33(354 minutes)
Session started at: 2022-01-19T08:08:54-0800
Session to end at: 2022-01-19T14:08:54-0800

Accumulated lab time: 00:06:27 (7 minutes)

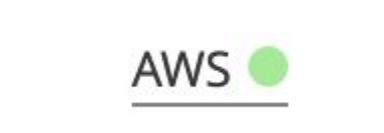
No running instance

SSH key Show Download PEM Download PPK

AWS SSO Download URL

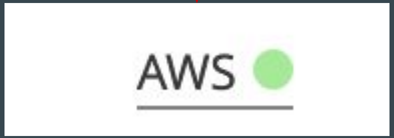
AWSAccountId	628404237974
Region	us-east-1

7. Click on AWS Console to open the Web Console



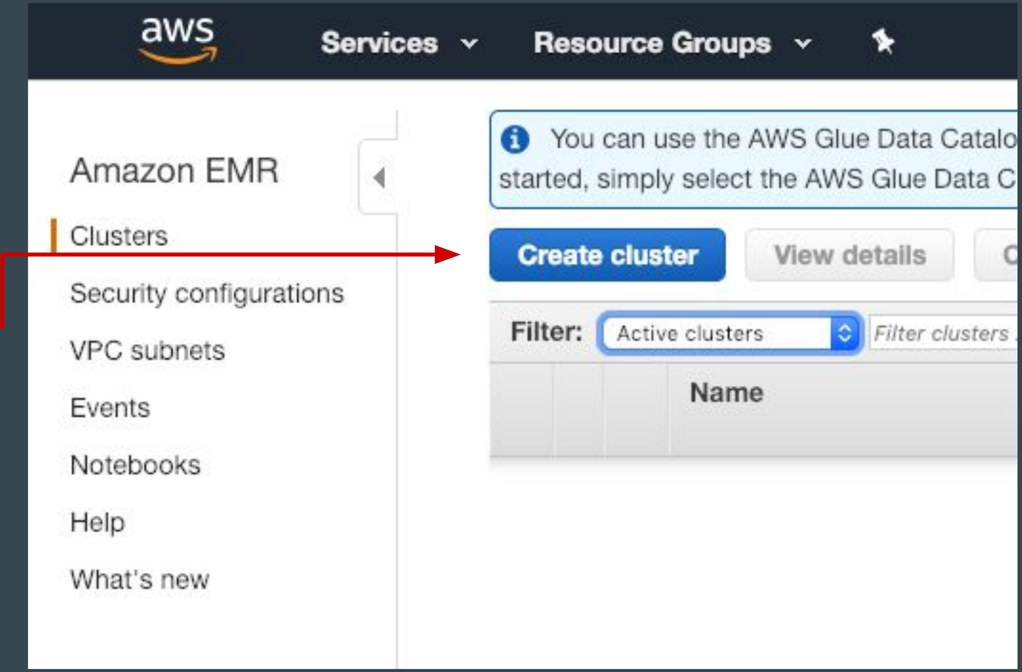
AWS console

7. Click on AWS Console to open the Web Console

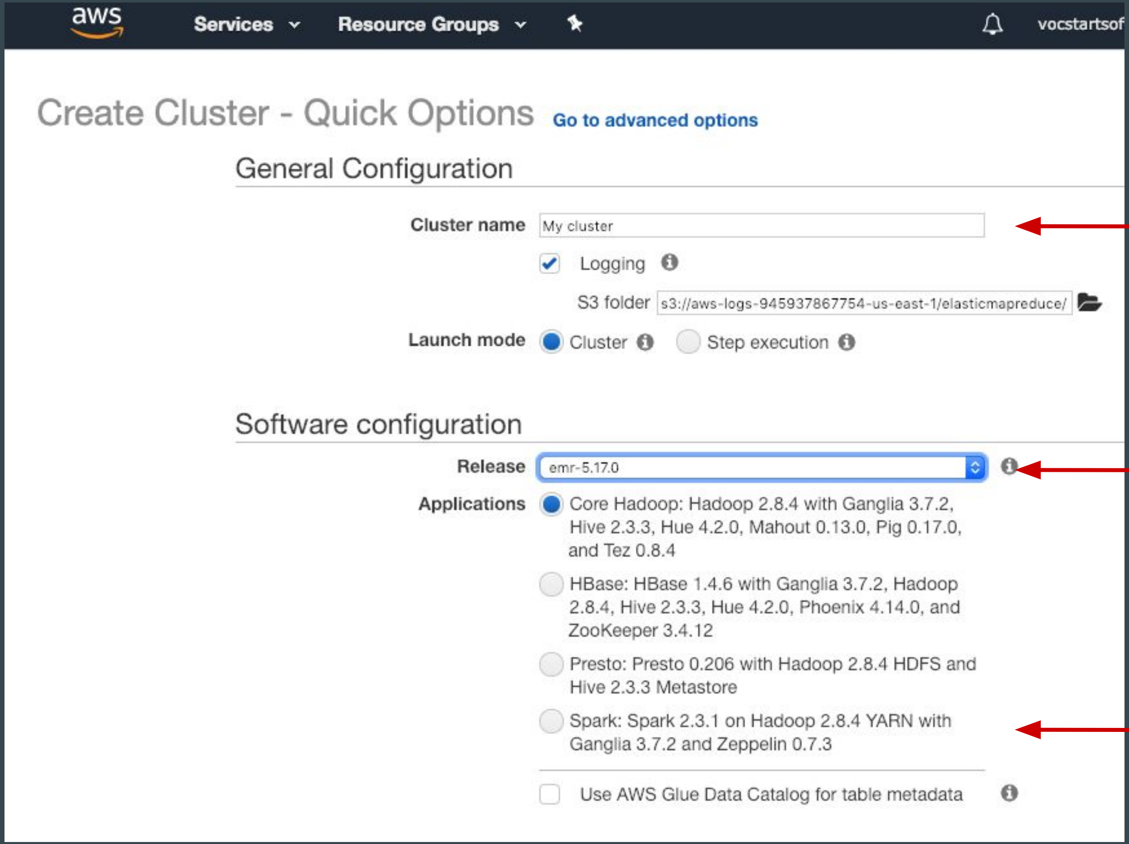


Setting up a Cluster on AWS

- 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



You can use the AWS Glue Data Catalo
started, simply select the AWS Glue Data C



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

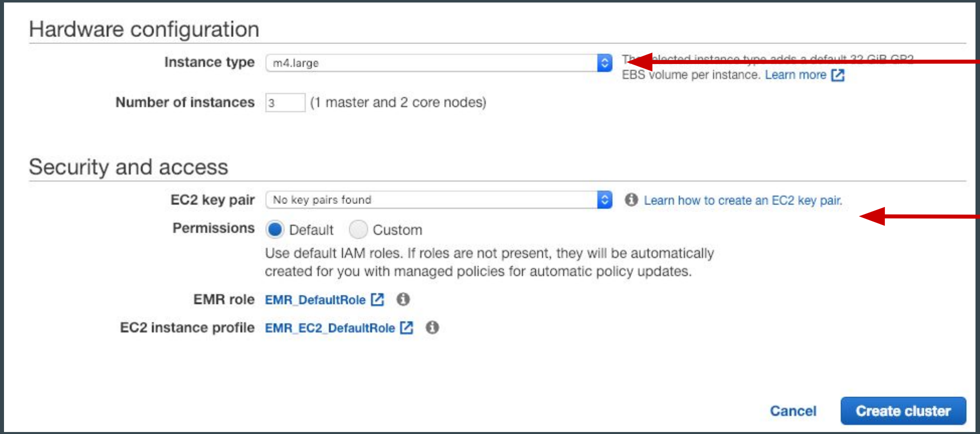
○ Spark: Spark 2.3.1 on Hadoop 2.8.4 YARN with Ganglia 3.7.2 and Zeppelin 0.7.3

□ Use AWS Glue Data Catalog for table metadata

1. Give it a name

2.Use `emr-5.17.2`

3. Select Spark support



Hardware configuration

Instance type: m4.large — The selected instance type adds a default 32 GiB GP2 EBS volume per instance. Learn more

Number of instances: 3 (1 master and 2 core nodes)

Security and access

EC2 key pair: No key pairs found — Learn how to create an EC2 key pair.

Permissions: ● Default ○ Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

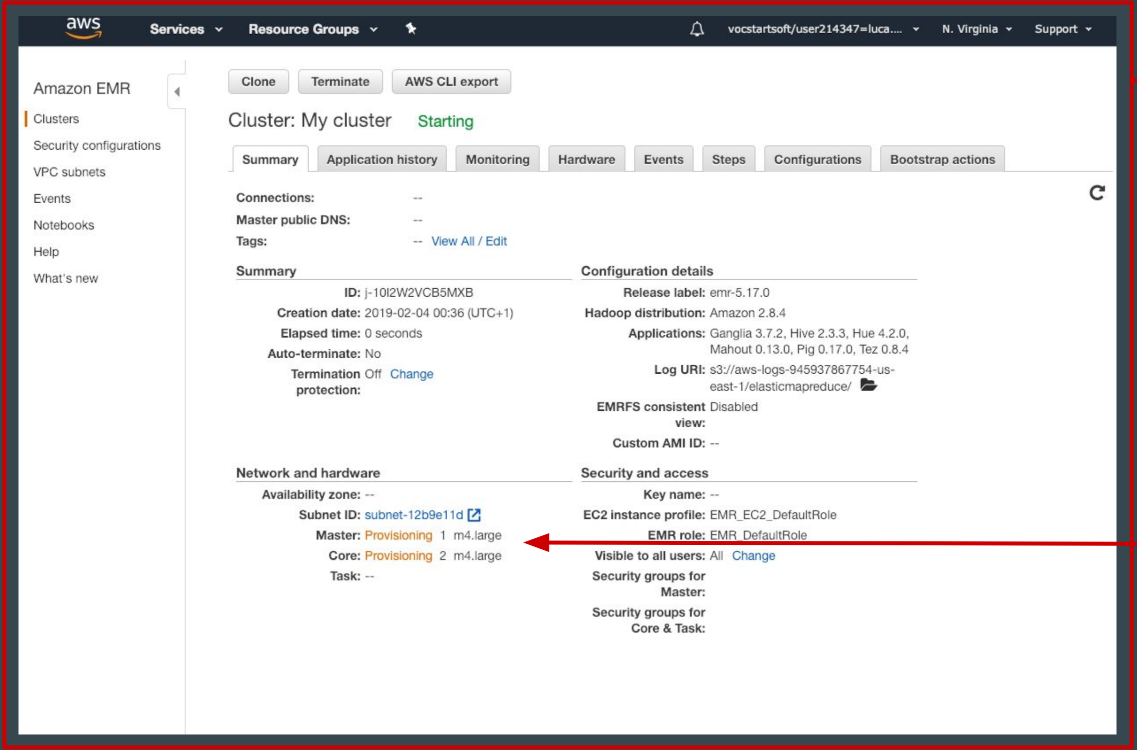
EMR role: EMR_DefaultRole

EC2 instance profile: EMR_EC2_DefaultRole

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*



Cluster: My cluster Starting

Connections: --

Master public DNS: --

Tags: -- View All / Edit

Summary

ID: j-10I2W2VCB5MXB

Creation date: 2019-02-04 00:36 (UTC+1)

Elapsed time: 0 seconds

Auto-terminate: No

Termination protection: Off Change

Configuration details

Release label: emr-5.17.0

Hadoop distribution: Amazon 2.8.4

Applications: Ganglia 3.7.2, Hive 2.3.3, Hue 4.2.0, Mahout 0.13.0, Pig 0.17.0, Tez 0.8.4

Log URI: s3://aws-logs-945937867754-us-east-1/elasticmapreduce/

EMRFS consistent view: Disabled

Custom AMI ID: --

Network and hardware

Availability zone: --

Subnet ID: subnet-12b9e11d

Master: Provisioning 1 m4.large

Core: Provisioning 2 m4.large

Task: --

Security and access

Key name: --

EC2 instance profile: EMR_EC2_DefaultRole

EMR role: EMR_DefaultRole

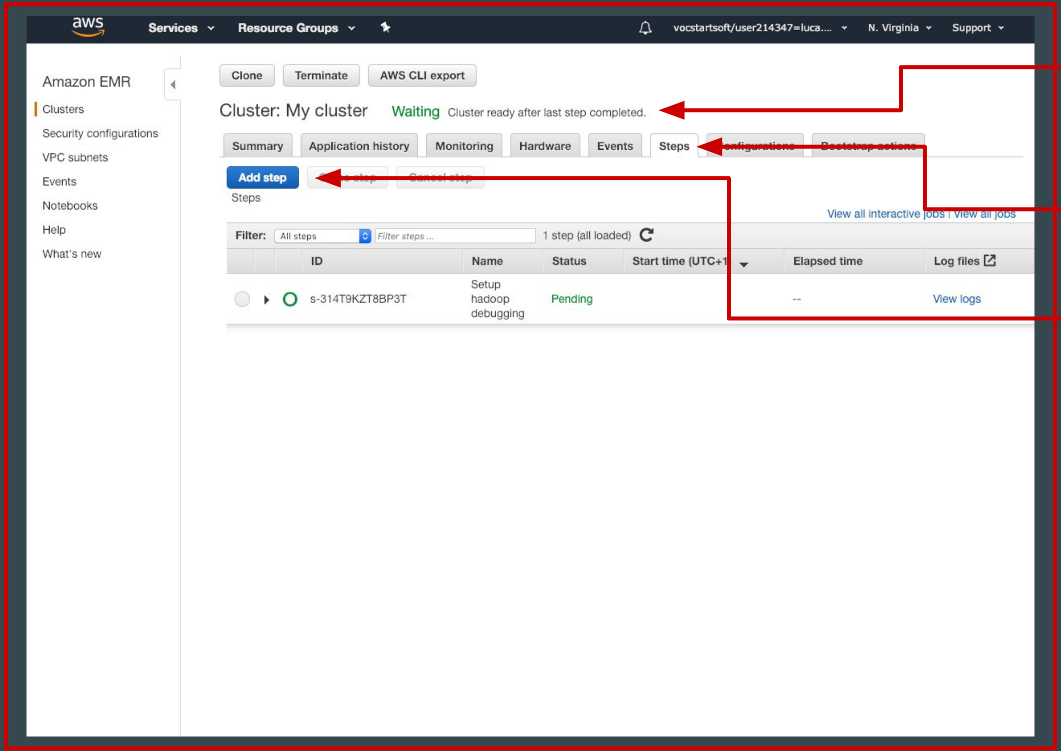
Visible to all users: All Change

Security groups for Master:

Security groups for Core & Task:

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



Cluster: My cluster Waiting Cluster ready after last step completed.

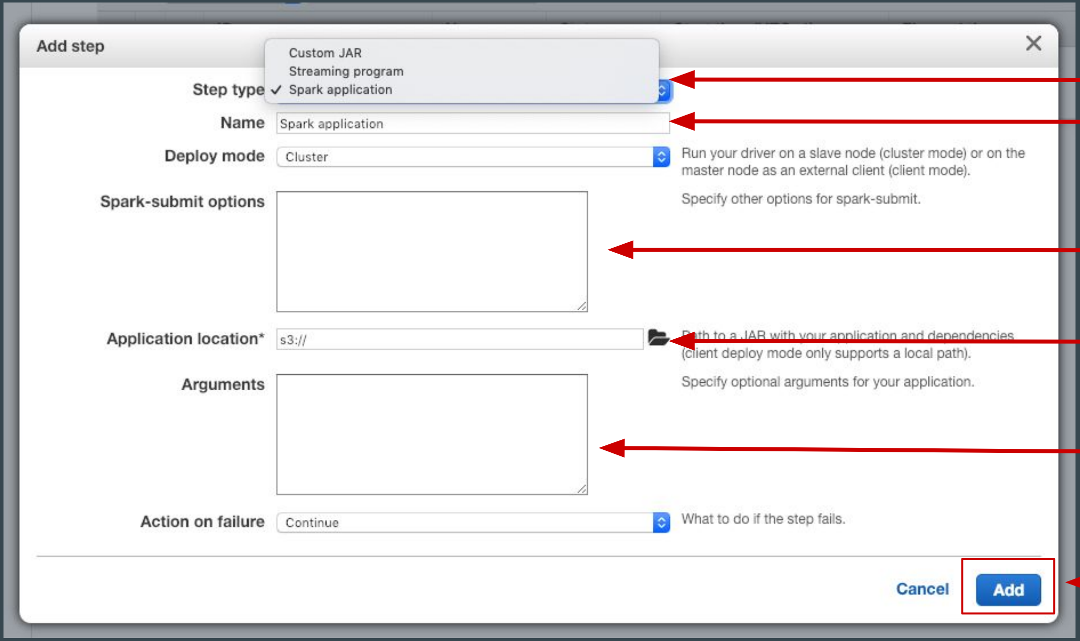
1 step (all loaded)

ID	Name	Status	Start time (UTC+1)	Elapsed time	Log files
s-314T9KZT8BP3T	Setup hadoop debugging	Pending		--	View logs

- After bootstrapping the cluster is green and waiting to run applications. **Remember to stop it after your steps end!**

- Select the *Steps* tab

- Select *Add step* to configure a Spark application



Add step

Step type: Spark application

Name: Spark application

Deploy mode: Cluster — Run your driver on a slave node (cluster mode) or on the master node as an external client (client mode).

Spark-submit options: — Specify other options for spark-submit.

Application location*: s3:// — Path to a JAR with your application and dependencies (client deploy mode only supports a local path).

Arguments: — Specify optional arguments for your application.

Action on failure: Continue — What to do if the step fails.

Select *Spark Application*

Provide a name

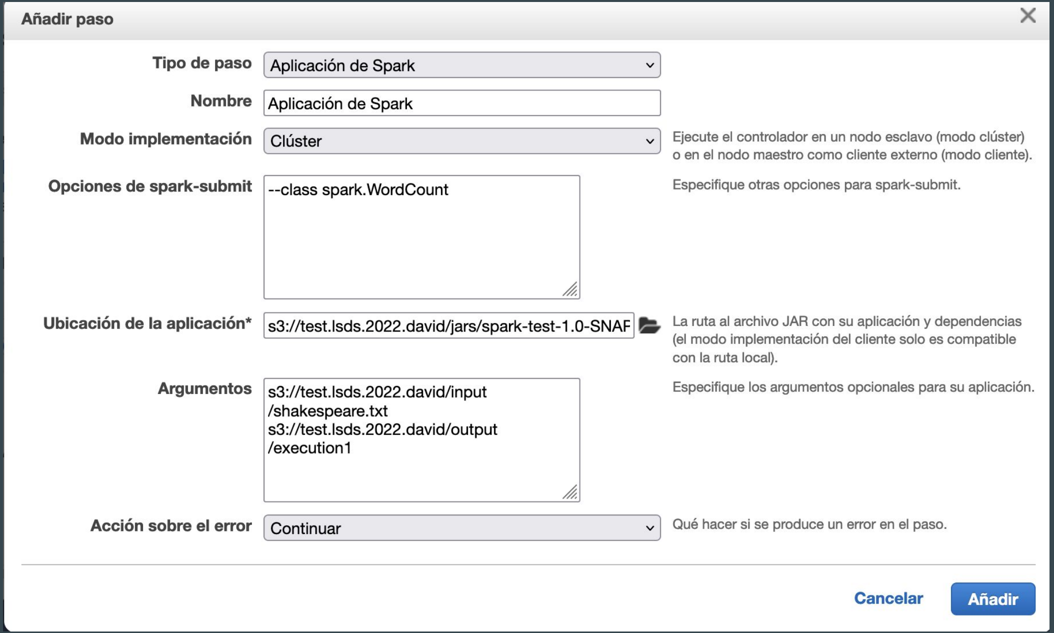
The options are the same that you'd give to `spark-submit` on the command line

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



Añadir paso

Tipo de paso: Aplicación de Spark

Nombre: Aplicación de Spark

Modo implementación: Clúster

Ejecute el controlador en un nodo esclavo (modo clúster) o en el nodo maestro como cliente externo (modo cliente).

Opciones de spark-submit:

```
--class spark.WordCount
```

Especifique otras opciones para spark-submit.

Ubicación de la aplicación*: s3://test.lsds.2022.david/jars/spark-test-1.0-SNAF

La ruta al archivo JAR con su aplicación y dependencias (el modo implementación del cliente solo es compatible con la ruta local).

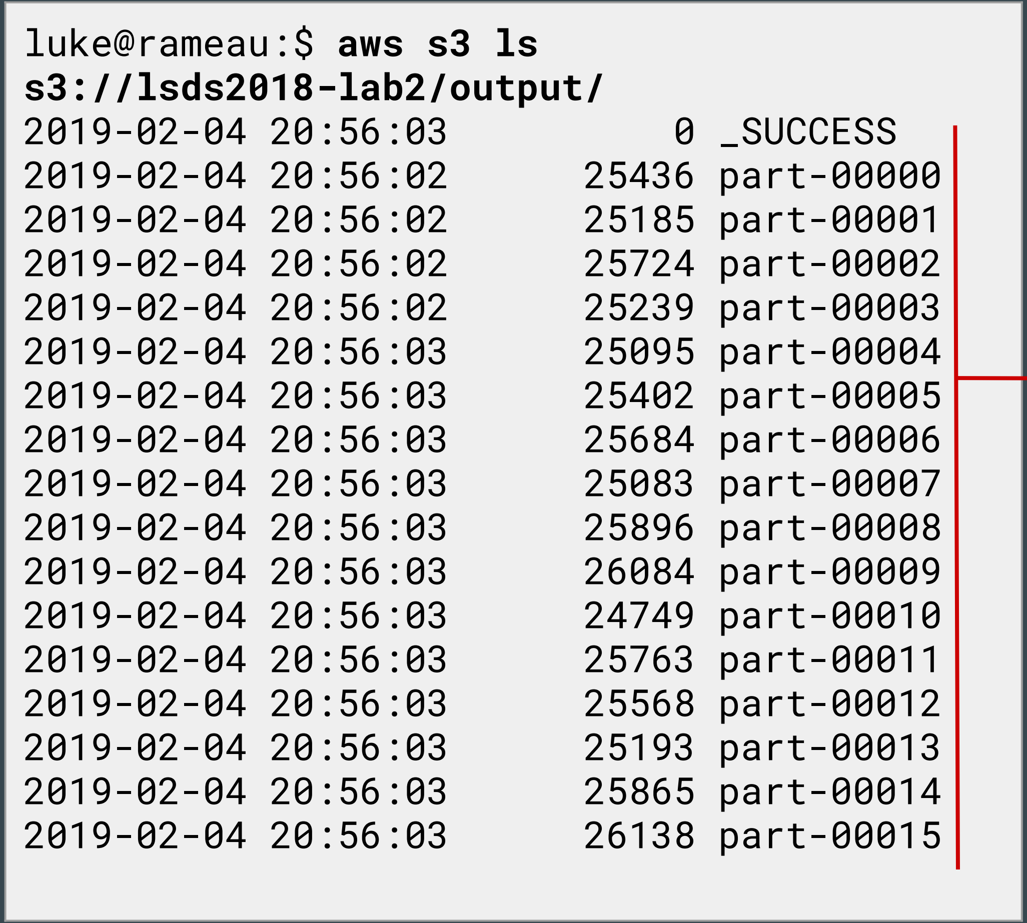
Argumentos:

```
s3://test.lsds.2022.david/input
/shakespeare.txt
s3://test.lsds.2022.david/output
/execution1
```

Especifique los argumentos opcionales para su aplicación.

Acción sobre el error: Continuar

Qué hacer si se produce un error en el paso.



```
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.

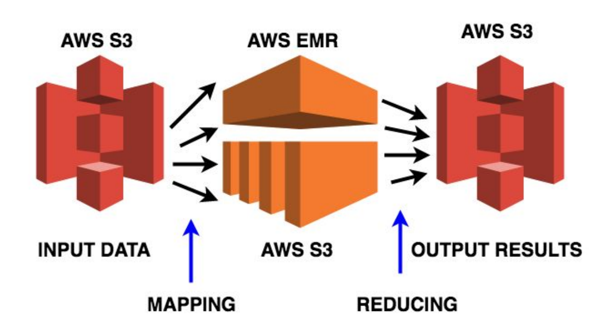
Use `mvn package` to generate an executable JAR from it

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?