Guide: Spark Cluster on AWS

Ignacio M. Llorente

v2.0 - 2 March 2020

Abstract

This is a screenshot document of how to run an EMR Spark cluster andSpark scripts in the AWS environment.

Requirements

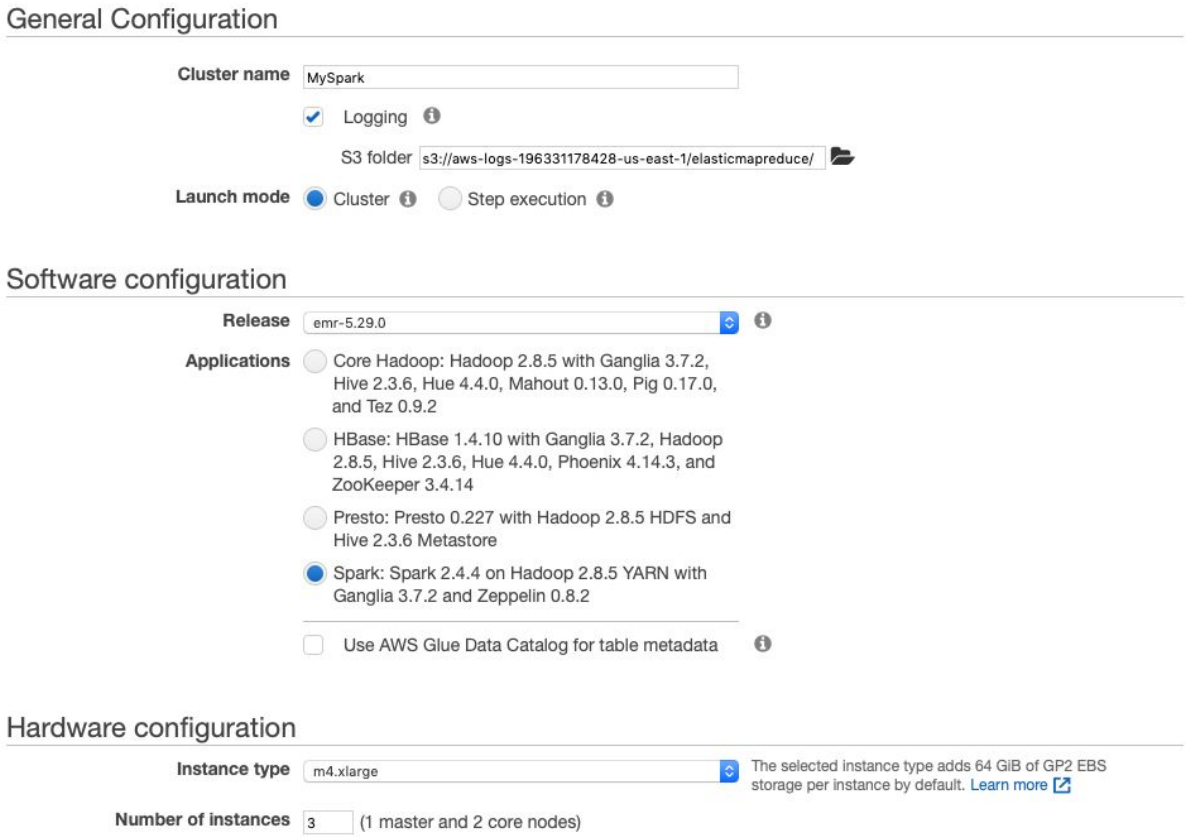
- **First you should have followed the Guide "First Access to AWS"**. It is assumed you already have an AWS account and a key pair, and you are familiar with the AWS EC2 environment.
- **Its is strongly recommended to firstly follow the Guide "Install Spark in Local Mode"** in order to get familiar with the Spark environment.
- We strongly recommend cluster instances with at least 4 vCPUs (**m4.xlarge)** to be able to evaluate parallel implementation within each node.
- The files needed to do the exercises are available for download from **Canvas**.

Acknowledgments

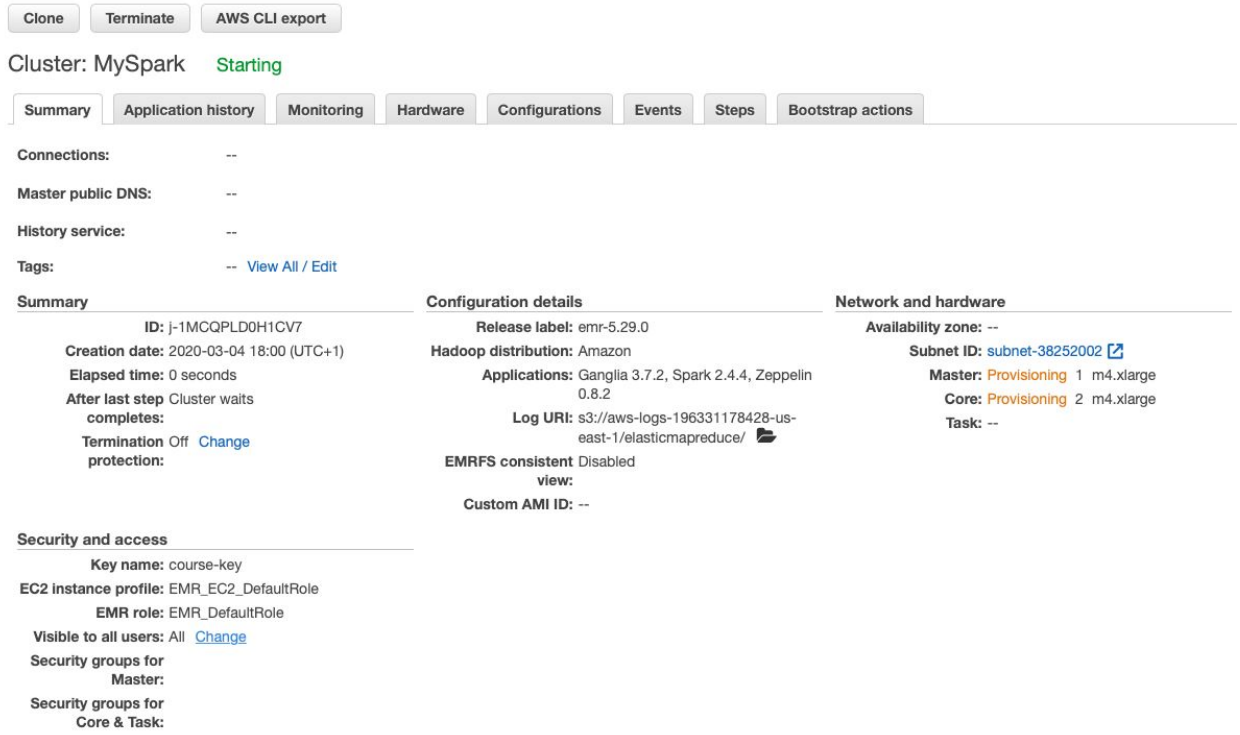
The author is grateful for constructive comments and suggestions from David Sondak, Charles Liu, Matthew Holman, Keshavamurthy Indireshkumar, Kar Tong Tan, Zudi Lin, Nick Stern, Dylan Randle, Hayoun Oh, Zhiying Xu and Zijie Zhao.

1. Launch Hadoop EMR cluster

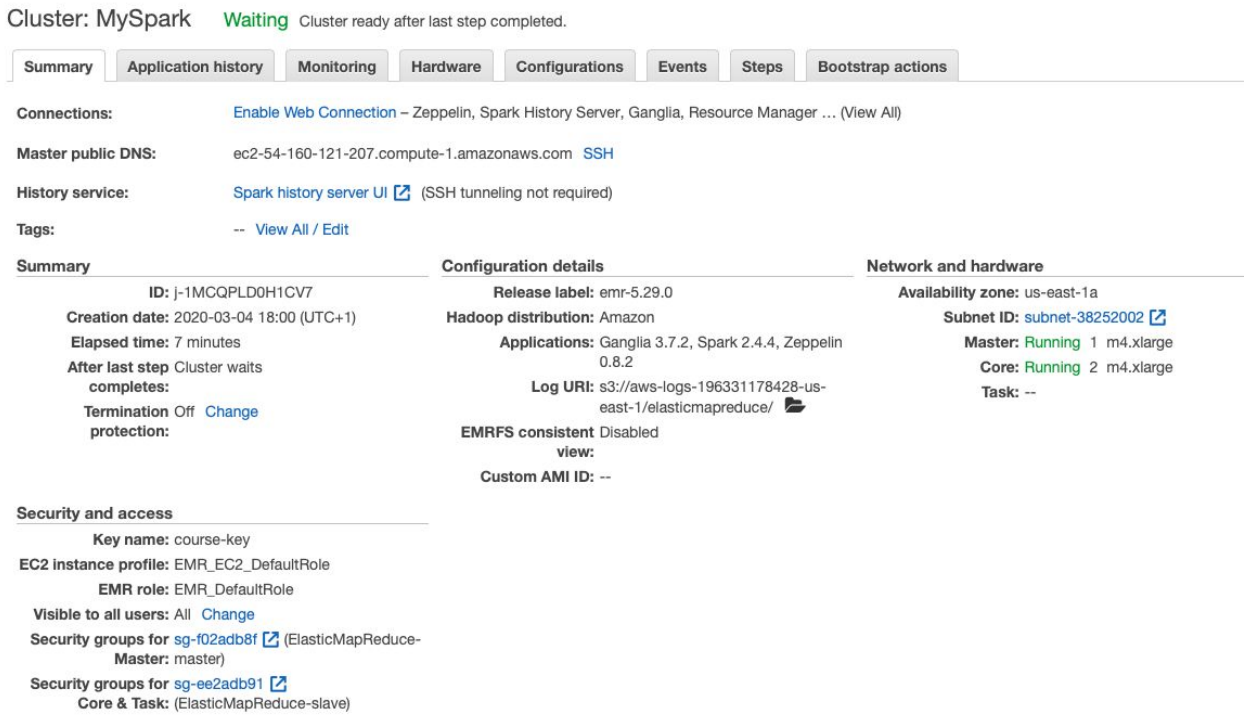
- Go to the EMR dashboard and click “Create cluster”. We recommend the following configuration
  - ClusterName: MySpark
  - Launch mode “Cluster”
  - Release: 5.29.0
  - Applications: Spark
  - Instance type: m4.xlarge
  - Number of Instances: 3
  - Key pair: course-key (or any other key you want to use, see Guide “First Access to AWS”)



- Click on “Create Cluster”

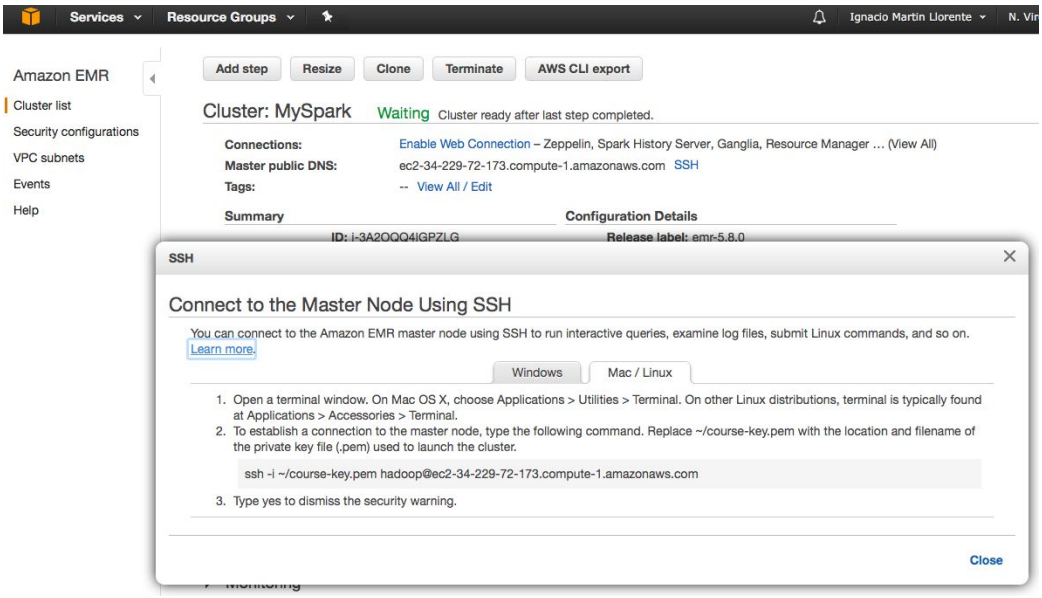


- Wait for the cluster to be ready. The cluster is ready when its state is “Waiting” and the Master and Core under the Networks and hardware section are both in “Running” state



2. Login to the cluster

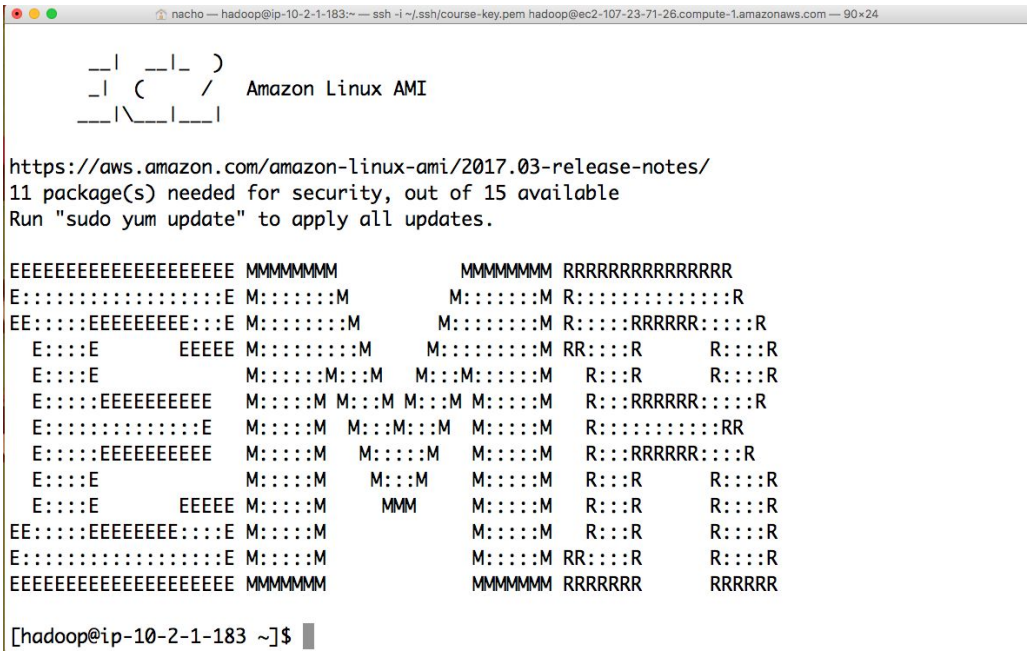
- Write down the “Master public DNS” and click on the SSH link next to it. The SSH link gives you the commands you might use to login to your cluster



- SSH to the machine using the private key. A sample command is as depicted (modify accordingly).

```
$ ssh -i $HOME/.ssh/course-key.pem hadoop@ec2-34-229-72-173.compute-1.amazonaws.com
```

- If you could not login then make sure that the security groups (firewalls) of the EMR cluster opens the port 22 to the outside world (see Guide “First Access to AWS”)



3. Submit a Spark Script

- This section shows how to submit spark jobs to a hadoop-powered spark framework using the command line interface from the master (front-end) node. See that in this case the Spark framework reads from and writes to a hadoop file system.
- Upload to the master VM the Spark `wordcount.py` script and the `input.txt` file with the ebook of Moby Dick used in the MapReduce labs
- Upload the `input.txt` file to the Hadoop file system

```
$ hadoop fs -put input.txt
```

```
$ hadoop fs -ls
Found 2 items
drwxr-xr-x   - hadoop hadoop          0 2017-09-07 15:38 .sparkStaging
-rw-r--r--   1 hadoop hadoop      16668 2017-09-07 16:26 input.txt
```

- Submit the job

```
$ spark-submit wordcount.py
17/09/07 16:52:42 INFO SparkContext: Running Spark version 2.2.0
17/09/07 16:52:42 INFO SparkContext: Submitted application: WordCount
17/09/07 16:52:42 INFO SecurityManager: Changing view acls to: hadoop
17/09/07 16:52:42 INFO SecurityManager: Changing modify acls to: hadoop
17/09/07 16:52:42 INFO SecurityManager: Changing view acls groups to:
17/09/07 16:52:42 INFO SecurityManager: Changing modify acls groups to:
…
```

- When the program finishes, check the hadoop file system again and look for the `output.txt` file (actually it is a folder containing the output files). Note that if we run the program again, it will fail unless `output.txt` is removed first. To remove `output.txt` use: `hadoop fs -rm -R -f output.txt`

```
$ hadoop fs -ls
Found 3 items
drwxr-xr-x   - hadoop hadoop          0 2017-09-07 15:38 .sparkStaging
-rw-r--r--   1 hadoop hadoop      16668 2017-09-07 16:26 input.txt
drwxr-xr-x   - hadoop hadoop          0 2017-09-07 16:55 output.txt
```

- Download the file from hadoop file system to the local file system and check the content

```
$ hadoop fs -get output.txt
```

```
$ cat output.txt/*
('swimming', 1)
('seemed', 1)
('pilot', 1)
('told', 3)
```

```
('balaene', 1)
('more', 4)
('history', 3)
('man', 2)
('wine', 1)
('speak', 1)
('quantity', 2)
('out', 7)
('davenant', 1)
```

- You have just executed the job on the master node but however you have NOT used the worker nodes yet.

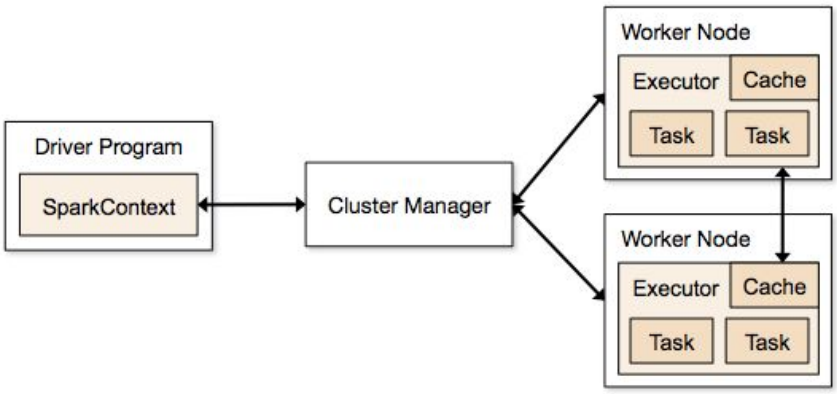
4. Parallel Execution on Multiple Nodes

Firstly see discussion about partitions, tasks and executors in the Guide "Start Spark in Local Mode". When using the Yarn Cluster Mode:

- The number of cores (threads within each executor) can be specified with the `--executor-cores` flag when invoking `spark-submit`, `spark-shell`, and `pyspark` from the command line, or by setting the `spark.executor.cores` property in the `spark-defaults.conf` file or on a `SparkConf` object. The cores property controls the number of concurrent tasks an executor can run.
- The number of executors (worker nodes) can be specified with the `--num-executors` command-line flag or `spark.executor.instances` configuration property.

For example, the following command will execute the script on 2 executors (worker nodes) with 4 threads per executor, achieving the execution of 8 simultaneous tasks **(when running a job on multiple nodes do NOT use the setMaster property with local in the SparkConf configuration**).

```
$ spark-submit --num-executors 2 --executor-cores 4  script
```



- Upload to the VM the Spark `pi.py` script, **remove the setMaster property in the SparkConf configuration to avoid local execution,** increase `N` to `100000000` to increase the CPU demand, and modify the code to use 16 partitions.

```
print sc.parallelize(xrange(N),16).map(...
```

- Execute the code in the cluster, and calculate the speedup for 2 executors and 1, 2 and 4 threads per executor.
- Resize the cluster (Hardware option) to have 4 worker nodes and calculate the speedup for 4 executors and 1, 2 and 4 threads per executor.

As sequential time to calculate the speed-up you can run the same code in local mode with only one thread (you should use `.setMaster("local[1]")` in the Spark configuration of the code.

Terminate the cluster when you are sure you are done for the day to avoid incurring charges