





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 and Spark 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")

General Configuration		
Cluster name	MySpark	
	Cogging	
	S3 folder s3://aws-logs-196331178428-us-east-1/elastic	mapreduce/ 📂
Launch mode	Cluster 1 Step execution 1	
Software configuration		
Release	emr-5.29.0	0
Applications	Core Hadoop: Hadoop 2.8.5 with Ganglia 3.7.2, Hive 2.3.6, Hue 4.4.0, Mahout 0.13.0, Pig 0.17.0, and Tez 0.9.2	
	HBase: HBase 1.4.10 with Ganglia 3.7.2, Hadoop 2.8.5, Hive 2.3.6, Hue 4.4.0, Phoenix 4.14.3, and ZooKeeper 3.4.14	
	Presto: Presto 0.227 with Hadoop 2.8.5 HDFS and Hive 2.3.6 Metastore	
	Spark: Spark 2.4.4 on Hadoop 2.8.5 YARN with Ganglia 3.7.2 and Zeppelin 0.8.2	
	Use AWS Glue Data Catalog for table metadata	0
Hardware configuration		
Instance type	m4.xlarge	The selected instance type adds 64 GiB of GP2 EBS storage per instance by default. Learn more 2
Number of instances	3 (1 master and 2 core nodes)	

• Click on "Create Cluster"



Clone T	erminate AWS CL	export						
Cluster: M	vSpark Startin							
oluster. Ivi	yopan otanin	9						
Summary	Application history	Monitoring	Hardware	Configurations	Events	Steps	Bootstrap actions	S
Connections:								
Master public	DNS:							
History service	e:							
Tags:	View	All / Edit						
Summary			Configu	ration details			Network a	and hardware
	ID: j-1MCQPLD0H1	CV7	I	Release label: emr-5	5.29.0		Availa	bility zone:
Creatio	n date: 2020-03-04 18:0	00 (UTC+1)	Hadoo	distribution: Amaz	on			Subnet ID: subnet-38252002 🔀
Elapse	d time: 0 seconds			Applications: Gang	lia 3.7.2, Spa	rk 2.4.4, Ze	ppelin	Master: Provisioning 1 m4.xlarge
After la	st step Cluster waits			0.8.2				Core: Provisioning 2 m4.xlarge
com	pletes:			Log URI: s3://a	ws-logs-196	331178428	-us-	Task:
Term	ination Off Change			east-	1/elasticmapr	•		
prot	lection:		EMR	-S consistent Disac	oled			
			Ci	stom AMI ID:				
Security and	access							
Key	name: course-key							
EC2 instance	profile: EMR_EC2_Defa	ultRole						
EN	IR role: EMR_DefaultRol	e						
Visible to all	lusers: All Change							
Security gro	ups for Master:							
Security gro Core	ups for & Task:							

• 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

Summary Application his	tory Monitoring	Hardware	Configurations	Events	Steps	Bootstrap actions	
Connections: Master public DNS: History service: Tags:	Enable Web Connect ec2-54-160-121-207. Spark history server I View All / Edit	ion – Zeppelin, Sp compute-1.amazo JI 🛃 (SSH tunne	ark History Server, G onaws.com SSH ling not required)	anglia, Reso	urce Manaç	ger (View All)	
Summary		Configu	uration details			Network an	id hardware
Creation date: 2020-03 Elapsed time: 7 minute After last step Cluster completes: Termination Off Cha protection:	-04 18:00 (UTC+1) s valts	Hadooj EMR	Applications: Amaze Applications: Gangi 0.8.2 Log URI: s3://a east-1 FS consistent Disabi view: istom AMI ID:	on ia 3.7.2, Spa ws-logs-196 /elasticmapi led	rk 2.4.4, Ze 331178428 educe/ 🗲	sppelin -us-	ubnet ID: subnet-38252002 Master: Running 1 m4.xlarge Core: Running 2 m4.xlarge Task:
Security and access							
Key name: course-/ EC2 instance profile: EMR_ECC EMR role: EMR_De Visible to all users: All Chai Security groups for sg-f02ac Master: master) Security groups for sg-ec2ac Core & Task: (Elastich	ey 2_DefaultRole faultRole 196 196 (CasticMapRe 1991 (CasticMapRe 1991 (CasticMapRe)	educe-					



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

Amazon EMR	Add step Resize	Clone Terminate AWS CLI export	
Cluster list	Cluster: MySpark	Waiting Cluster ready after last step completed.	
Security configurations /PC subnets Events	Connections: Master public DNS: Tags:	Enable Web Connection – Zeppelin, Spark History Server, Ganglia, Resource M ec2-34-229-72-173.compute-1.amazonaws.com SSH View All / Edit	anager (View All)
Help	Summary	Configuration Details	
	ID: i-	-3A2OQQ4IGPZLG Release label: emr-5.8.0	
	SSH Connect to the Master	-3420004(GPZLG Release label: emr.5.8.0	
	ID: I- SSH Connect to the Master You can connect to the Amazor Learn more	r Node Using SSH	ix commands, and so on.
	ID: I- SSH Connect to the Master You can connect to the Amazor Learn more	r Node Using SSH on EMR master node using SSH to run interactive queries, examine log files, submit Linu Windows Mac / Linux	ix commands, and so on.
	ID: I- SSH Connect to the Master You can connect to the Amazor Learn more. 1. Open a terminal window at Applications > Acces 2. To establish a connecti the private key file (per	Release label: emr-5.8.0 r Node Using SSH windows Mac / Linux windows Mac / Linux ww. On Mac OS X, choose Applications > Utilities > Terminal. On other Linux distributions sesories > Terminal. ion to the master node, type the following command. Replace ~/course-key.pem with th m) used to launch the cluster.	x commands, and so on. s, terminal is typically found he location and filename of
	ID: I- SSH Connect to the Master You can connect to the Amazor Learn more. 1. Open a terminal window at Applications > Acces 2. To establish a connecti the private key file (.per ssh -i ~/course-key.p.	Release label: emr-5.8.0 Node Using SSH windows Mac / Linux windows windows	ix commands, and so on. s, terminal is typically found he location and filename of

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")

```
👚 nacho — hadoop@ip-10-2-1-183:~ — ssh -i ~/.ssh/course-key.pem hadoop@ec2-107-23-71-26.compute-1.amazonaws.com — 90×24
. . .
      _1 ( /
                  Amazon Linux AMI
               _
https://aws.amazon.com/amazon-linux-ami/2017.03-release-notes/
11 package(s) needed for security, out of 15 available
Run "sudo yum update" to apply all updates.
EEEEEEEEEEEEEEEEE MMMMMMM
                                    E:::::::::::::::E M:::::::M
                                   M::::::R
EE::::EEEEEEEEE:::E M:::::::M
                                  M:::::::M R:::::RRRRRR:::::R
 E::::E
            EEEEE M:::::::M
                                 M::::::R
                                                        R::::R
 E::::E M::::M:::M M:::M:::M R:::R R:::F
E:::::EEEEEEEEE M::::M M:::M M:::M M:::M R::::R RRRRRR::::R
                                                        R::::R
 E:::::EEEEEEEEE M:::::M M:::::M R::::RRRRRR::::R
 E::::E
                  M:::::M
                            M:::M
                                     M::::M
                                             R:::R
                                                        R::::R
             EEEEE M:::::M
                             MMM
 E::::E
                                     M:::::M
                                              R:::R
                                                        R::::R
EE::::EEEEEEEE:::E M:::::M
                                     M:::::M
                                              R:::R
                                                        R::::R
E:::::::::::::::E M:::::M
                                    M:::::R RR::::R
                                                        R::::R
EEEEEEEEEEEEEEEE MMMMMMM
                                    MMMMMMM RRRRRR
                                                        RRRRRR
[hadoop@ip-10-2-1-183 ~]$
```



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 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)
```

© © © © ©

CS205: Computing Foundations for Computational Science, Spring 2020

```
('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