CS205: Computing Foundations for Computational Science, Spring 2021

IACS INSTITUTE FOR APPLIED COMPUTATIONAL SCIENCE AT HARVARD UNIVERSITY

HARVARD School of Engineering and Applied Sciences

Guide: Spark Cluster on AWS

Ignacio M. Llorente, Simon Warchol

v3.0 - 4 April 2021

Abstract

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

Requirements

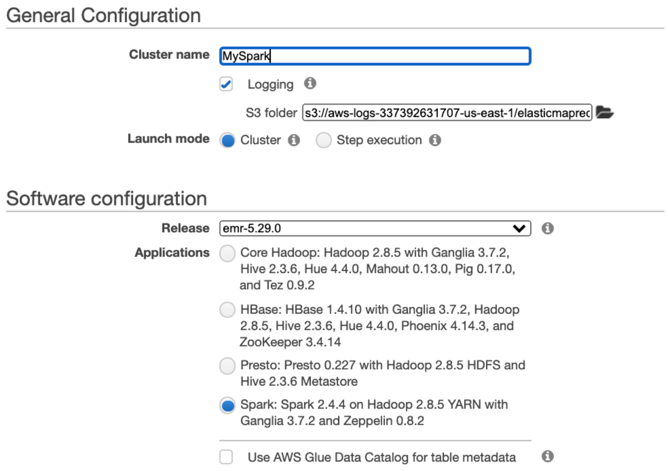
- **First you should have followed the Guide “First Access to AWS”**. It is assumed you already have an AWS account and a key pair, and you are familiar with the AWS EC2 environment.
- **Its is strongly recommended to firstly follow the Guide “Install Spark in Local Mode”** in order to get familiar with 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.

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”)
- **Make sure to select EMR release 5.29.0**



General Configuration

Cluster name: MySpark

☑ Logging

S3 folder: s3://aws-logs-337392631707-us-east-1/elasticmaprec

Launch mode: ● Cluster ○ Step execution

Software configuration

Release: emr-5.29.0

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

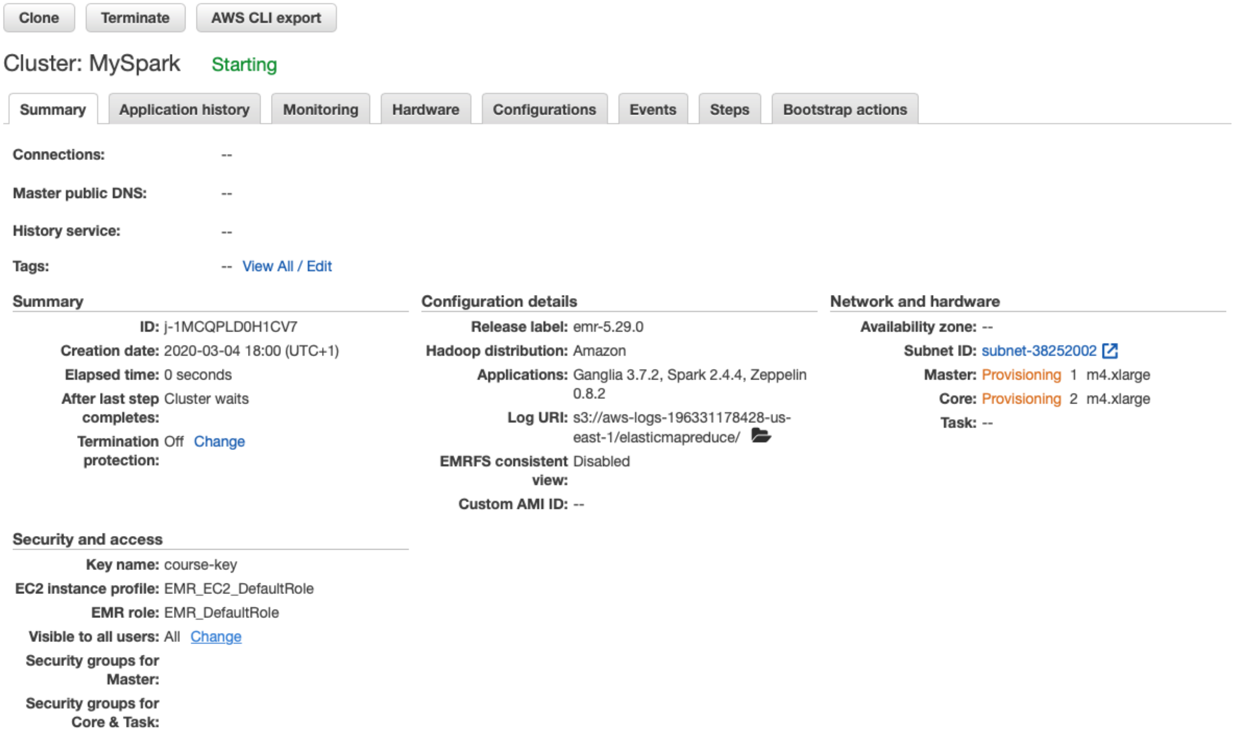
○ HBase: HBase 1.4.10 with Ganglia 3.7.2, Hadoop 2.8.5, Hive 2.3.6, Hue 4.4.0, Phoenix 4.14.3, and ZooKeeper 3.4.14

○ 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

- Click on “Create Cluster”



Cluster: MySpark Starting

Connections: --

Master public DNS: --

History service: --

Tags: -- View All / Edit

Summary

ID: j-1MCQPLD0H1CV7
Creation date: 2020-03-04 18:00 (UTC+1)
Elapsed time: 0 seconds
After last step completes: Cluster waits
Termination protection: Off Change

Configuration details

Release label: emr-5.29.0
Hadoop distribution: Amazon
Applications: Ganglia 3.7.2, Spark 2.4.4, Zeppelin 0.8.2
Log URI: s3://aws-logs-196331178428-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

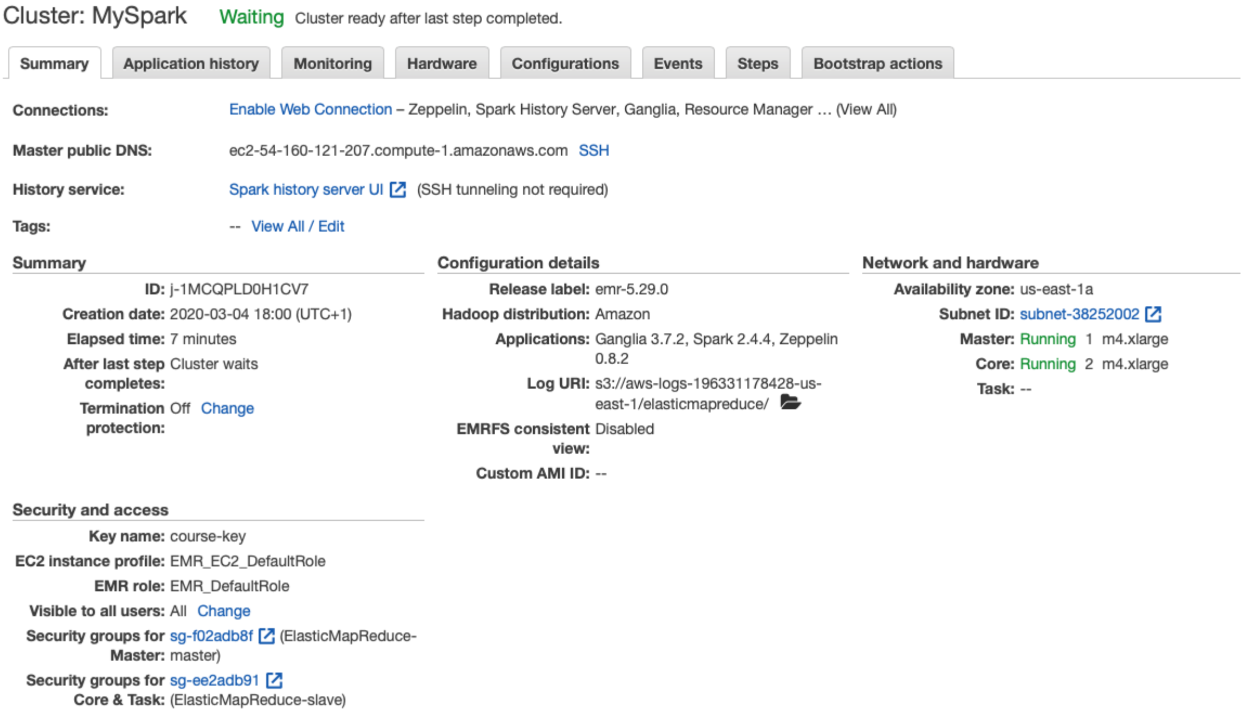
Network and hardware

Availability zone: --
Subnet ID: subnet-38252002
Master: Provisioning 1 m4.xlarge
Core: Provisioning 2 m4.xlarge
Task: --

Security and access

Key name: course-key
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:

- 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



Cluster: MySpark Waiting Cluster ready after last step completed.

Connections: Enable Web Connection – Zeppelin, Spark History Server, Ganglia, Resource Manager ... (View All)

Master public DNS: ec2-54-160-121-207.compute-1.amazonaws.com SSH

History service: Spark history server UI (SSH tunneling not required)

Tags: -- View All / Edit

Summary

ID: j-1MCQPLD0H1CV7
Creation date: 2020-03-04 18:00 (UTC+1)
Elapsed time: 7 minutes
After last step completes: Cluster waits
Termination protection: Off Change

Configuration details

Release label: emr-5.29.0
Hadoop distribution: Amazon
Applications: Ganglia 3.7.2, Spark 2.4.4, Zeppelin 0.8.2
Log URI: s3://aws-logs-196331178428-us-east-1/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

Network and hardware

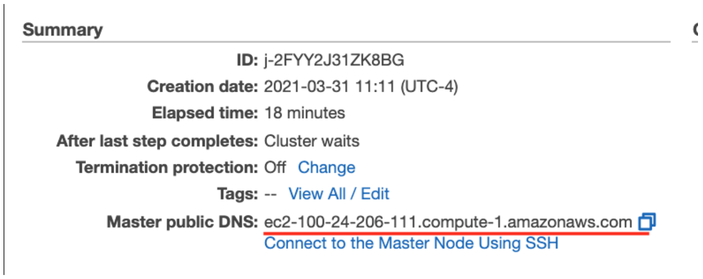
Availability zone: us-east-1a
Subnet ID: subnet-38252002
Master: Running 1 m4.xlarge
Core: Running 2 m4.xlarge
Task: --

Security and access

Key name: course-key
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All Change
Security groups for Master: sg-f02adb8f (ElasticMapReduce-master)
Security groups for Core & Task: sg-ee2adb91 (ElasticMapReduce-slave)

2. Login to the cluster

- Copy the “Master public DNS” SSH into the machine using your CS205-key. Note that the user you are logging into is `hadoop` not `ubuntu`

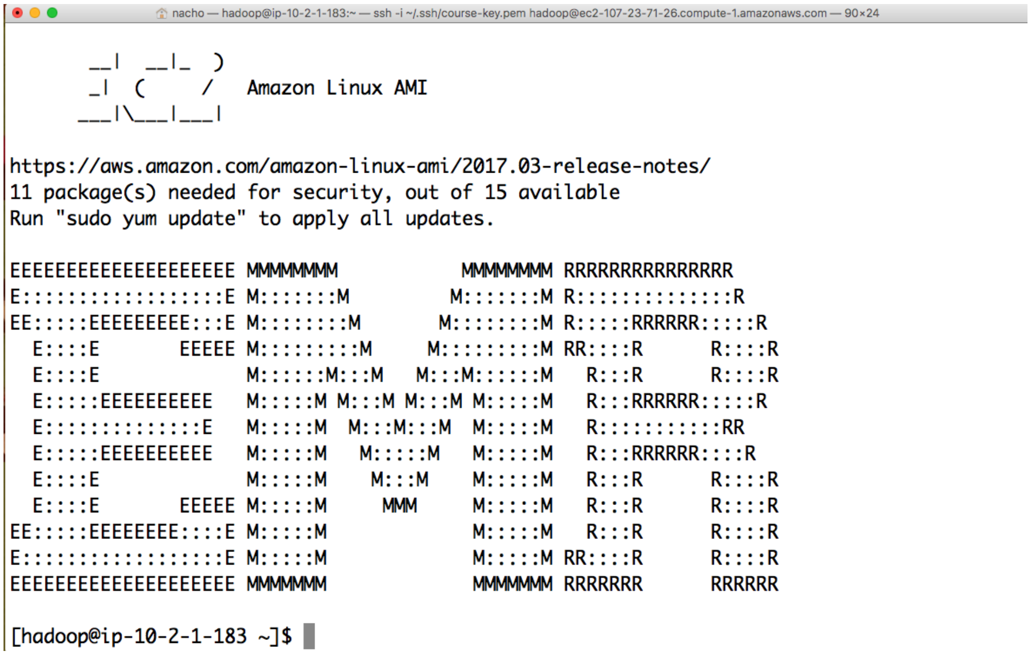


Summary

ID: j-2FYY2J31ZK8BG
Creation date: 2021-03-31 11:11 (UTC-4)
Elapsed time: 18 minutes
After last step completes: Cluster waits
Termination protection: Off Change
Tags: -- View All / Edit
Master public DNS: ec2-100-24-206-111.compute-1.amazonaws.com
Connect to the Master Node Using SSH

```
$ ssh -i ~/.ssh/CS205-key.pem hadoop@ec2-100-24-206-111.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

```
       __|  __|_  )
       _|  (     /   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.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      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 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.

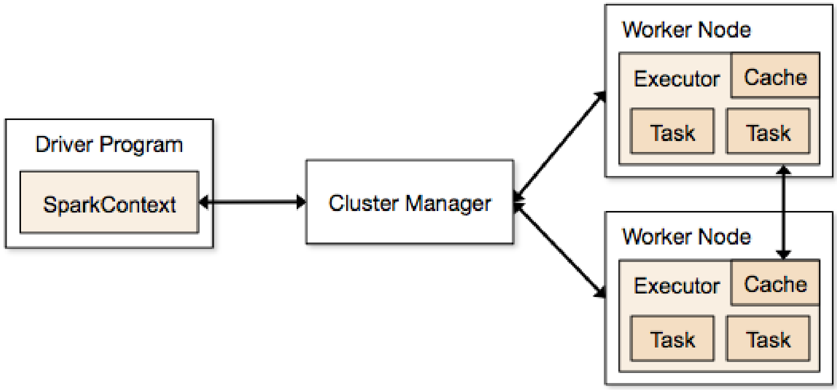
4. Parallel Execution on Multiple Nodes

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.

```
    spark-submit --num-executors 2 --executor-cores 1 pi.py
```

- 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