**Spark Application on Amazon EMR cluster**

-Prepared by Bharath

-------------------------------------------------------------------------------------------------------------------------------

Steps:

1. Create an AWS account
2. Choose a region on which deploy the cluster
3. Create an Amazon S3 Bucket

* Open the Amazon S3 console
* Choose Create bucket
* Type a name for the bucket (ex : my-first-emr-bucket) and choose its AWS Region then click next.
* On the Set properties page, we can configure some properties for the bucket. We don’t need any specific properties as its for training, click next.
* On the Set permissions page, we manage permissions that are set on the bucket that we are creating. We will use the default permissions, click next.
* On the Review page, verify the settings and choose Create bucket.

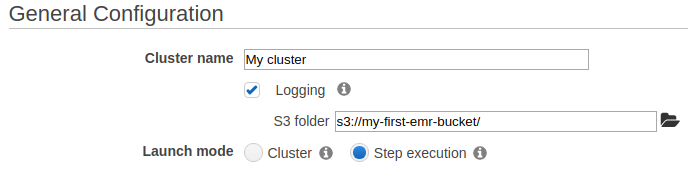
1. Upload files on Amazon S3

* Data Source: https://insights.stackoverflow.com/survey
* Download and untar it
* On the Amazon S3 console click on the bucket we just created
* Choose Create folder, enter the name of the folder (ex : myEMR), the encryption setting (ex : None) and save
* Click on the folder we just created, then choose upload to upload

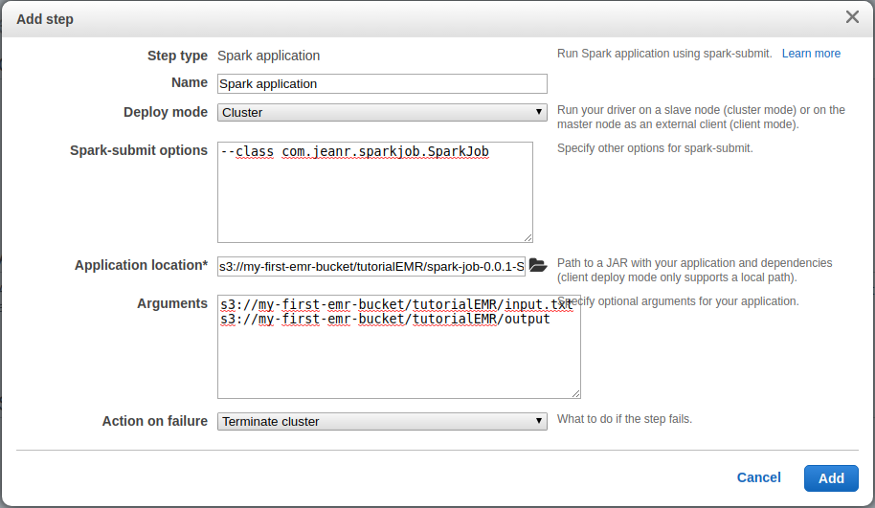
## **Create an Amazon EMR cluster & Submit the Spark Job**

We will launch a sample cluster running the Spark job and terminating automatically after the execution.

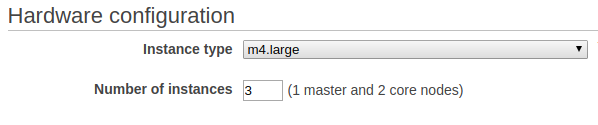
* Open the [Amazon EMR console](https://console.aws.amazon.com/elasticmapreduce/)
* On the right left corner, **change the region** on which we want to deploy the cluster (choose any one region)
* Choose **Create cluster**
* On the **General Configuration**section, enter the cluster name, **choose the S3 bucket** we created (the logs will be stored in this bucket) and **check Step execution.**



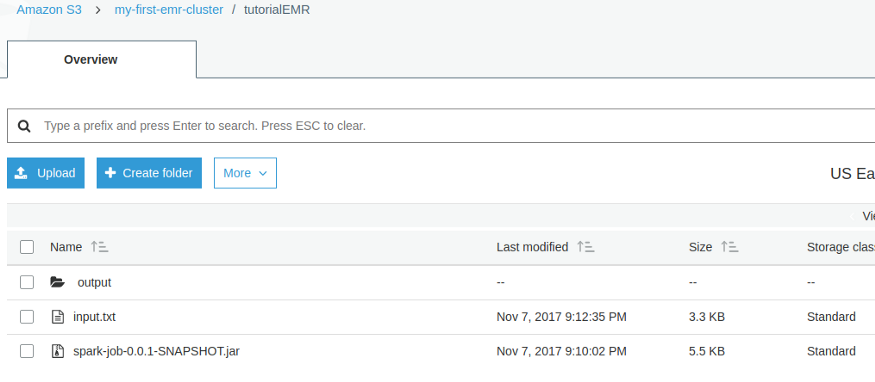
* On the **Add steps**section, **select Spark application**, click **Configure**and fill the popup like this:



* On the **Software Configuration**section, use the default release (the last one)
* On the **Hardware configuration** section, **choose the instance** type and **the number of instances**



* On the **Security and access** section, use the Default values**.**
* Click on **Create cluster**
* . Click on **the refresh icon** to see the status passing from Starting to Running to Terminating — All steps completed
* Now go back to the S3 console and we will see the output directory in which the result has been stored, we can click on it and download its contents



* Upload the python script in S3 (word count application or any)
* Click bucket name in aws and select upload
* Input dataset and Python Script should be inside the S3

**Method 1 : Connect/Login to the Spark Master Machine/ Node via SSH: (PuTTY)**

* Click SSH under cluster page in AWS
* For Windows: Have to use PuTTY
* Convert key pairs from AWS to putty private key format
* Open Putty key Gen click on Load
* Select key pair file and click on save private key
* From Cluster web page in AWS copy the host name of master machine
* Then open putty paste the host name under host name box
* Select SHH as connection type
* On left click connection and expand then expand SSH
* Click on Auth then click browse then select ppk file generated using puttygen
* Click open then we will be inside the EMR Master Machine
* Then fetch the .py file from S3 to master machine for execution
* Use aws s3 cp command to copy files from and to s3
* Supply with the destination file
* aws s3 cp s3://stackoverflow-analytics-pdero/stackoverflowsurvey.py .
* Then it fetches the files from s3 ,after .py file is downloaded submit the python script to spark EMR cluster
* spark-submit Stackoverflowsurvey .py
* And hit enter to submit the spark Job - log output and all job outputs will be displayed

**Method 2:**

1. AWS CLI (Optional)

https://docs.aws.amazon.com/cli/latest/userguide/install-cliv1.html

<https://docs.aws.amazon.com/general/latest/gr/rande.html#apigateway_region>

$ aws configure

AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE

AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLE

Default region name [None]: us-west-2

Default output format [None]: json

1. Using AWS CLI **[aws s3api create-bucket](http://docs.aws.amazon.com/cli/latest/reference/s3api/create-bucket.html)**

<https://docs.aws.amazon.com/cli/latest/reference/s3api/create-bucket.html> Using AWS CLI, use **[aws s3 sync](http://docs.aws.amazon.com/cli/latest/reference/s3/sync.html)** to upload the directory containing the Spark application jar and the input text file

$ aws s3 sync myEMR/ s3://my-second-emr-bucket/myEMR

[**Create default roles**](http://docs.aws.amazon.com/cli/latest/reference/emr/create-default-roles.html) which will be used to create the cluster

$ aws emr create-default-roles

[**Find the SubnetId**](http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-subnets.html) of our default region

$ aws ec2 describe-subnets \  
 --filters "Name=availabilityZone,Values=us-east-2b"

[**Create the cluster**](http://docs.aws.amazon.com/cli/latest/reference/emr/create-cluster.html) and submit the Spark step

$ aws emr create-cluster \  
 --name “Cluster” \  
 --release-label emr-5.9.0 \  
 --instance-type m4.large — instance-count 3 \  
 --applications Name=Spark \  
 --steps Type=Spark,Name=”Spark Program”,ActionOnFailure=CONTINUE,Args=[ — class,com.jeanr84.sparkjob.SparkJob,s3://my-second-emr-bucket/tutorialEMR/spark-job-0.0.1-SNAPSHOT.jar,s3://my-second-emr-bucket/tutorialEMR/input.txt,s3://my-second-emr-bucket/tutorialEMR/output] \  
 --use-default-roles \  
 --ec2-attributes SubnetId=subnet-e7ba9d9c \  
 --auto-terminate

Download locally the result repository

$ aws s3 sync s3://my-second-emr-bucket/tutorialEMR/output output

**Note:** To launch the classic word count Spark Job on EMR. The input and output files will be store using S3 storage.

When we Upload files on Amazon S3, choose upload to upload the Spark application jar/py files and the input text file on which we want to apply the wordcount

**PySpark Word Count :**

|  |
| --- |
| import sys |
|  | from operator import add |
|  |  |
|  | from pyspark.sql import SparkSession |
|  |  |
|  |  |
|  | if \_\_name\_\_ == "\_\_main\_\_": |
|  | if len(sys.argv) != 2: |
|  | print("Usage: wordcount <file>", file=sys.stderr) |
|  | sys.exit(-1) |
|  |  |
|  | spark = SparkSession\ |
|  | .builder\ |
|  | .appName("PythonWordCount")\ |
|  | .getOrCreate() |
|  |  |
|  | lines = spark.read.text(sys.argv[1]).rdd.map(lambda r: r[0]) |
|  | counts = lines.flatMap(lambda x: x.split(' ')) \ |
|  | .map(lambda x: (x, 1)) \ |
|  | .reduceByKey(add) |
|  | output = counts.collect() |
|  | for (word, count) in output: |
|  | print("%s: %i" % (word, count)) |
|  |  |
|  | spark.stop() |

import sys

from pyspark import SparkContext, SparkConf

if \_\_name\_\_ == "\_\_main\_\_":

# create Spark context with necessary configuration

sc = SparkContext("local","PySpark Word Count Bharath")

# read data from text file and split each line into words

words = sc.textFile("C:/Bharath/spark/input.txt").flatMap(lambda line: line.split(" "))

# count the occurrence of each word

wordCounts = words.map(lambda word: (word, 1)).reduceByKey(lambda a,b:a +b)

# save the counts to output

wordCounts.saveAsTextFile("C:/Bharath/spark/output/")

Run this Python Spark Application.

spark-submit pyspark\_Bharath.py