SPARKIFY 5 – MapReduce streaming step

<<TASK 0>> create an EMR cluster

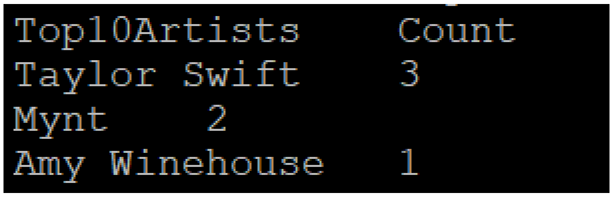
Make sure no custom inbound rules exist in ElasticMapReduce-master security group

1. Go to EMR
2. Click create cluster
3. s/w config: release – **emr-5.33.1**; applications – **Hadoop**
4. h/w config: **m4.xlarge**
5. add your EC2 key-pair
6. click create cluster
7. wait for it to start up – approx. takes up to 15 mins
   [when status is 'waiting'; you are ready to use the cluster]

<<TASK 1>> make mapper and reducer files

0. SSH into master node of EMR cluster [make sure port 22 is accessible; else add in inbound rules]
1. create a virtual python environment (if not already exists) and activate it
   [python3 -m venv venv
   source venv/bin/activate]
2. vim mapper.py – write the mapper code
3. vim reducer.py – write the reducer code
4. vim test.json – create a test file with some json objects [or use a single json file from the log_data]
5. allow execution permissions to mapper and reducer files:
   chmod a+x *.py
6. test the mapper and reducer code on your test data:
   cat test.json | ./mapper.py | sort | ./reducer.py

   output looks something like this: [the names and count depend on what values you took in the test]

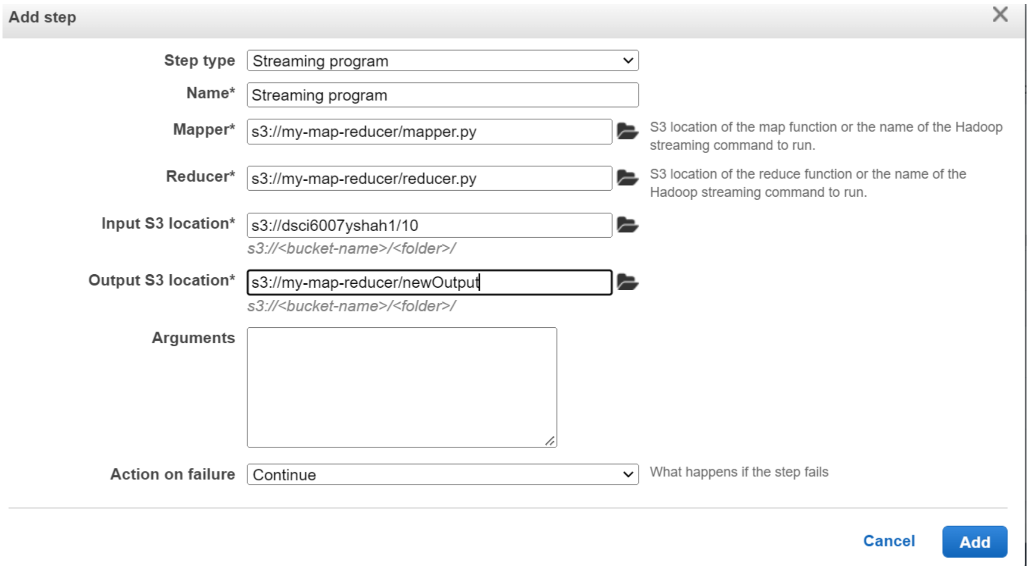


```
Top10Artists     Count
Taylor Swift     3
Mynt     2
Amy Winehouse    1
```

   *count of artists arranged in descending order

<<TASK 2>> Run a streaming step

1. Upload your mapper.py and reducer.py to an S3 bucket
   [not same bucket where the log_data resides]
2. Before creating a streaming job; let's test the functionality on a smaller scale
3. [Refer this for detailed instructions]
   a. Go to EMR and select your cluster
   b. Go to step and click on add step
   c. Create a streaming step as below:
      make sure to select the right locations for your mapper, reducer, and input files



Step type: Streaming program

Name*: Streaming program

Mapper*: s3://my-map-reducer/mapper.py

S3 location of the map function or the name of the Hadoop streaming command to run.

Reducer*: s3://my-map-reducer/reducer.py

S3 location of the reduce function or the name of the Hadoop streaming command to run.

Input S3 location*: s3://dsci6007yshah1/10

s3://<bucket-name>/<folder>/

Output S3 location*: s3://my-map-reducer/newOutput

s3://<bucket-name>/<folder>/

Arguments:

Action on failure: Continue

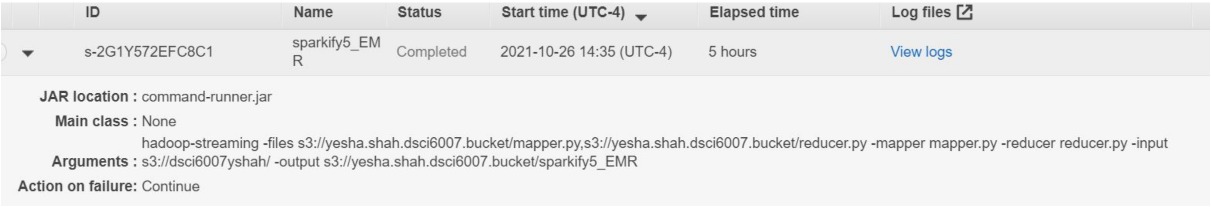
What happens if the step fails

* for output location, bucket can be an existing one. However, the folder MUST be new

d. click add and wait for it to finish running

e. once finished, you should be able to see the output folder in S3. You will notice that the output is in parts.

4. Now, lets run the map-reduce job on our entire log data
5. Like we did before [step 3], create a new streaming service.
   a. The only difference this time would be to use the input folder as your whole log data. In my case, s3://dsci6007yshah1/ - it takes all files in that bucket as input
   b. Again, make sure to add a new output folder [one that is not already existing]
6. Add the step and wait for it to finish – this can take 6-7 hours; depending on various factors, make sure to 'start' your AWS academy lab to avoid the 4 hour time limit.
7. Once done, you can expand the step to see the details. It should look something like below



ID	Name	Status	Start time (UTC-4)	Elapsed time	Log files
s-2G1Y572EFC8C1	sparkify5_EMR	Completed	2021-10-26 14:35 (UTC-4)	5 hours	View logs

JAR location : command-runner.jar

Main class : None

Arguments : hadoop-streaming -files s3://yesha.shah.dsci6007.bucket/mapper.py,s3://yesha.shah.dsci6007.bucket/reducer.py -mapper mapper.py -reducer reducer.py -input s3://dsci6007yshah/ -output s3://yesha.shah.dsci6007.bucket/sparkify5_EMR

Action on failure: Continue

8. You can also check the output folder in the S3 bucket; you will notice that the output is in parts

SPARKIFY 5 – MapReduce streaming step

mapper.py

```
#! /usr/bin/python3
import json
import sys

for line in sys.stdin:
    obj = json.loads(line)
    if obj['artist']:
        print('{}\t{}'.format(obj['artist'], 1))
```

reducer.py

```
#!/usr/bin/python
import sys
from collections import Counter
count = {}
for line in sys.stdin:
    line = line.split('\t')
    try:
        count[line[0]] += 1
    except:
        count[line[0]] = 1

print('Top10Artists\tCount')
count = Counter(count).most_common(10)
for (key, value) in count:
    print('{}\t{}'.format(key, value))
```