



ÅBO AKADEMI UNIVERSITY

CLOUD COMPUTING

Assignment 5



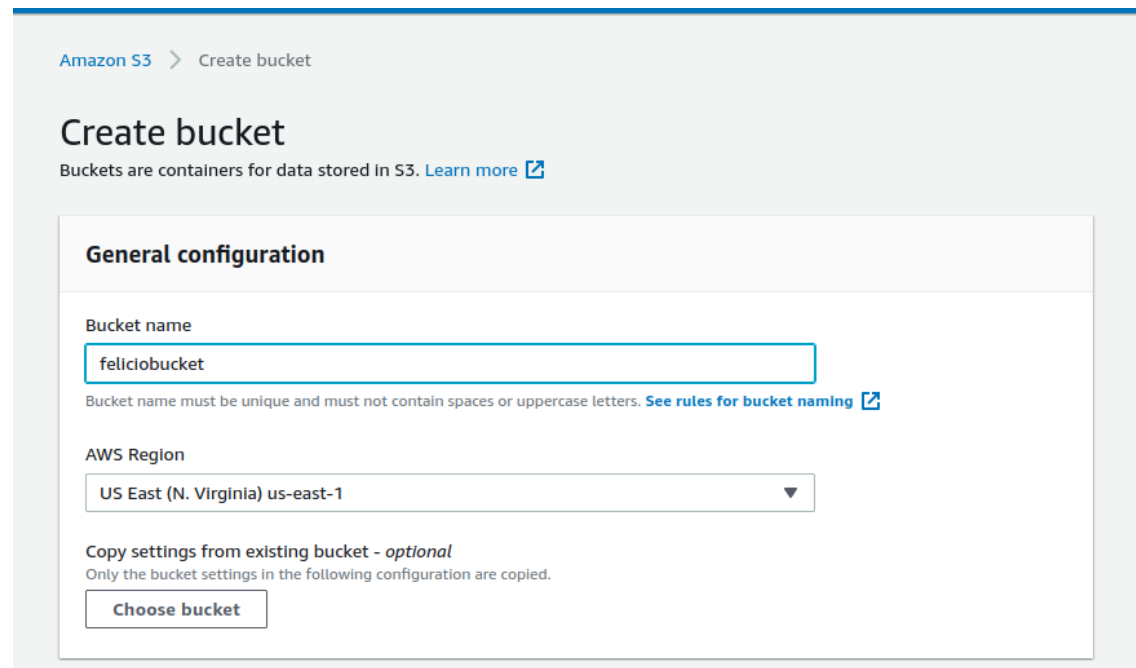
FILIBE FELÍCIO (2004623)

MAY 5, 2021

Problem 1: Word counting

1) Buckets, folders and files

In order to run an Elastic MapReduce job, we need to create a bucket on Amazon Simple Storage Service (S3), the AWS storage service, which will allow us to store our input files, our algorithms, and to save the outputs.



Amazon S3 > Create bucket

Create bucket

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

US East (N. Virginia) us-east-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Figure 1

2) Python MapReduce Code

The Mapper will read data from STDIN, split it into words and output a list of lines mapping words to their (intermediate) counts to STDOUT. The Map script will not compute an (intermediate) sum of a word's occurrences though. Instead, it will output `{word} 1` tuples immediately – even though a specific word might occur multiple times in the input.

```

1  #!/usr/bin/env python
2  """mapper.py"""
3
4  import sys
5
6  # input comes from STDIN (standard input)
7  for line in sys.stdin:
8      # remove leading and trailing whitespace
9      line = line.strip()
10     # split the line into words
11     words = line.split()
12     # increase counters
13     for word in words:
14         # write the results to STDOUT (standard output);
15         # what we output here will be the input for the
16         # Reduce step, i.e. the input for reducer.py
17         #
18         # tab-delimited; the trivial word count is 1
19         print '%s\t%s' % (word, 1)

```

The Reduce step do the final sum count. It will read the results of mapper.py from STDIN (so the output format of mapper.py and the expected input format of reducer.py must match) and sum the occurrences of each word to a final count, select the 100 most common, and then output its results to STDOUT.

```
1 #!/usr/bin/env python
2 """reducer.py"""
3
4 import sys
5 import collections
6
7 counter = collections.Counter()
8
9 for line in sys.stdin:
10     k, v = line.strip().split("\t", 2)
11
12     counter[k] += int(v)
13
14 print counter.most_common(100)
```

3) Uploading your files to Amazon S3

First, I created a folder streamingcode in my bucket:

Create folder

Use folders to group objects in buckets. When you create a folder, S3 creates an object using the name that you specify followed by a slash (/). This object then appears as folder on the console. [Learn more](#)

Your bucket policy might block folder creation
If your bucket policy prevents uploading objects without specific tags, metadata, or access control list (ACL) grantees, you will not be able to create a folder using this configuration. Instead, you can use the [upload configuration](#) to upload an empty folder and specify the appropriate settings.

Folder

Folder name

streamingcode /

Folder names can't contain "/>

Figure 2

After, I uploaded the files mapper.py and reducer.py to it.

Upload

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. [Learn more](#)

Drag and drop files and folders you want to upload here, or choose **Add files**, or **Add folders**.

Files and folders (2 Total, 1.6 KB) Remove Add files Add folder

All files and folders in this table will be uploaded.

<input type="checkbox"/>	Name	Folder	Type	Size
<input type="checkbox"/>	mapper.py	-	text/x-python	553.0 B
<input type="checkbox"/>	reduce.py	-	text/x-python	1.0 KB

Figure 3

Then, I created a folder input in my bucket:

Create folder

Use folders to group objects in buckets. When you create a folder, S3 creates an object using the name that you specify followed by a slash (/). This object then appears as folder on the console. [Learn more](#)

Your bucket policy might block folder creation

If your bucket policy prevents uploading objects without specific tags, metadata, or access control list (ACL) grantees, you will not be able to create a folder using this configuration. Instead, you can use the [upload configuration](#) to upload an empty folder and specify the appropriate settings.

Folder

Folder name

/

Folder names can't contain "/". [See rules for naming](#)

Figure 4

Finally, I upload the input.txt file to the folder input:

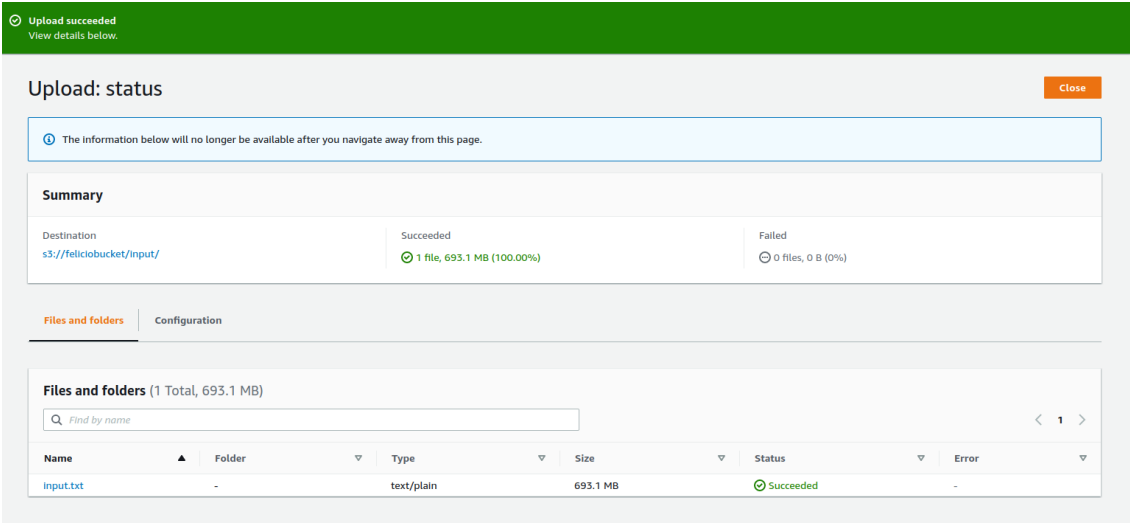


Figure 5

4) Create a Cluster

Then, I created a cluster with 3 m5.xlarge instances and the key-par I used in previous assignments.

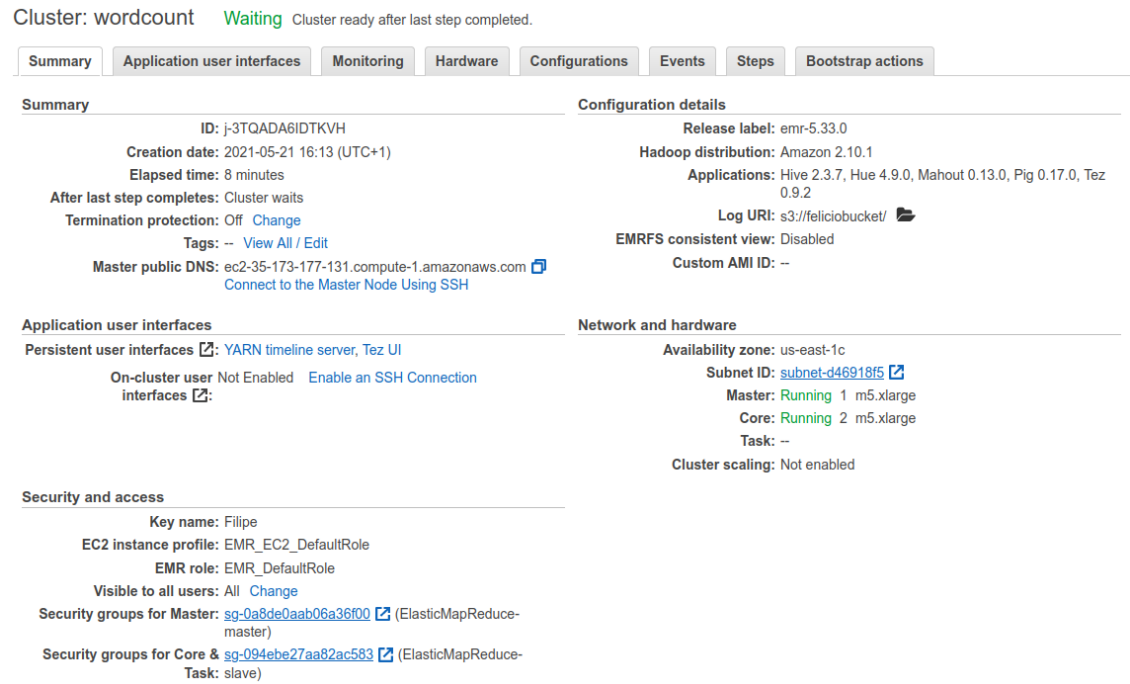


Figure 6

5) Launch a Step (Job Flow) in the Cluster

Add step

Step type: Streaming program

Name*: wordcount

Mapper*: s3://feliciobucket/streamingcode/mapper.py
S3 location of the map function or the name of the Hadoop streaming command to run.

Reducer*: s3://feliciobucket/streamingcode/reduce.py
S3 location of the reduce function or the name of the Hadoop streaming command to run.

Input S3 location*: s3://feliciobucket/input/input.txt
s3://<bucket-name>/<folder>/

Output S3 location*: s3://feliciobucket/output/
s3://<bucket-name>/<folder>/

Arguments:

Action on failure: Continue
What happens if the step fails

[Cancel](#) [Add](#)

Figure 7

After, having some outputs errors in two tries, I change the output location parameter and it worked:

Cluster: wordcount Waiting Cluster ready after last step failed.

Summary Application user interfaces Monitoring Hardware Configurations Events Steps Bootstrap actions

Concurrency: 1 [Change](#)

After last step completes: Cluster waits

[Add step](#) [Clone step](#) [Cancel step](#)

[View Jobs in the Application History Tab](#)

Filter:	All steps	Filter steps...	4 steps (all loaded)		
ID	Name	Status	Start time (UTC+1)	Elapsed time	Log files
s-26EWWLSEEROT6	Streaming program	Completed	2021-05-21 17:10 (UTC+1)	2 minutes	View logs
s-YL9MEXTGJVGF	wordcount	Failed	2021-05-21 17:06 (UTC+1)	4 seconds	controller syslog stderr stdout
s-E67XGAM0HX6D	wordcount	Failed	2021-05-21 16:49 (UTC+1)	6 seconds	controller syslog stderr stdout
s-W3JN24H7E7A4	Setup hadoop debugging	Completed	2021-05-21 16:21 (UTC+1)	4 seconds	View logs

Figure 8

6) Output

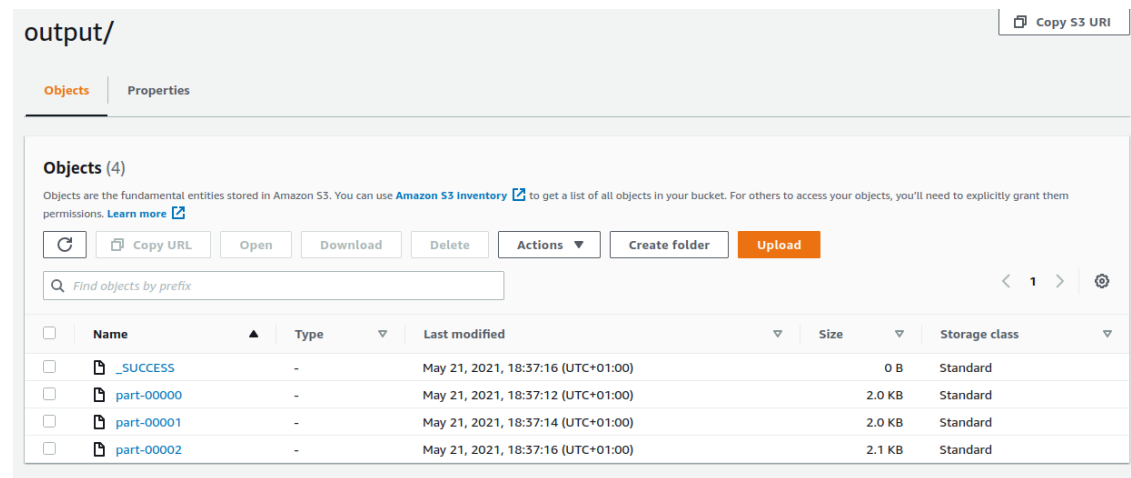


Figure 9

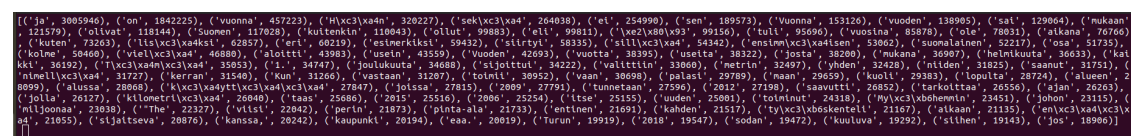


Figure 10

7) Implement a combiner in the map function

The combiner is used in between the map and the reduce to reduce the volume of data transfer between Map and Reduce.

```

1
2 #!/usr/bin/env python
3 """mapper_with_combiner.py"""
4
5 import sys
6
7 cur_station = ""
8 station_count = 0
9 sum_temps = 0
10
11 for line in sys.stdin:
12     try:
13         line = line.split('\t')
14         station, temp = line[0], float(line[1])
15
16         if station != cur_station:
17             if cur_station != "":
18                 print("%s\t%s\t%s"%(cur_station, sum_temps, station_count))
19                 cur_station = station
20                 station_count = 1
21                 sum_temps = temp
22             else:
23                 station_count += 1
24                 sum_temps += temp
25     except:
26         pass
27

```

```
28 print ("%s\t%s\t%s"%(cur_station, sum_temps, station_count))
```

Cluster: wordcount Waiting Cluster ready after last step failed.

Summary

Application user interfaces

Monitoring

Hardware

Configurations

Events

Steps

Bootstrap actions

Concurrency: 1

Change

After last step completes:

Cluster waits

Add step

Clone step

Cancel step

View Jobs in the Application History Tab

Filter:

All steps

Filter steps ...

9 steps (all loaded)

	ID	Name	Status	Start time (UTC+1)	Elapsed time	Log files
	s-1U7X2806X70QH	wcwithcombiner	Completed	2021-05-21 19:58 (UTC+1)	54 seconds	View logs
	s-3LPZYGTSGDMS7	wccombiner	Failed	2021-05-21 19:51 (UTC+1)	6 seconds	controller syslog stderr stdout
	s-35YEN1ZHOEWH1	wordcount3or5	Completed	2021-05-21 19:27 (UTC+1)	2 minutes	View logs
	s-2BZ4CQXF73MP3	wordcounter3or5	Failed	2021-05-21 19:17 (UTC+1)	2 minutes	controller syslog stderr stdout
	s-2NVR5RVAOQSBF	wordcount	Completed	2021-05-21 18:34 (UTC+1)	2 minutes	View logs
	s-26EVLSEEROT6	Streaming program	Completed	2021-05-21 17:10 (UTC+1)	2 minutes	View logs
	s-YL9MEXTGJVGF	wordcount	Failed	2021-05-21 17:06 (UTC+1)	4 seconds	controller syslog* stderr stdout
	s-E67XGAM0HX6D	wordcount	Failed	2021-05-21 16:49 (UTC+1)	6 seconds	controller syslog stderr stdout
	s-W3JN24H7E7A4	Setup hadoop debugging	Completed	2021-05-21 16:21 (UTC+1)	4 seconds	View logs

Figure 11

As we see can here, the execution time went from 2 min to 54 seconds due to the combiner. This means more or less 50% improvement.

8) Count the number of words of length 3 and 5

```
1 #!/usr/bin/env python
2 """reducer.py"""
3
4 import sys
5 import collections
6
7 counter = collections.Counter()
8
9 for line in sys.stdin:
10     k, v = line.strip().split("\t", 2)
11
12     if len(k) == 3 or len(k) == 5 :
13         counter[k] += int(v)
14
15 print counter.most_common(100)
```

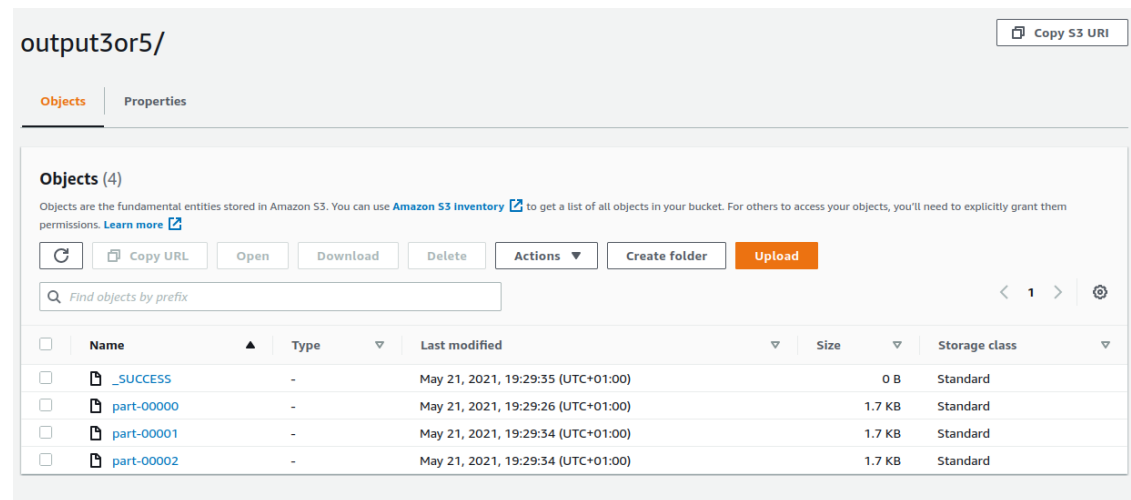



Figure 12

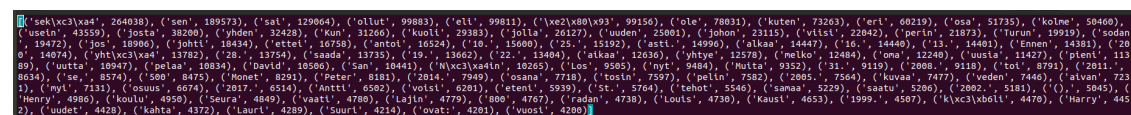


Figure 13

2) Problem 2:

1) Implement a MapReduce program to calculate the resulting CDN costs due to the number of served requests and the transferred data. Provide the total number of requests and the total volume of transferred data (in base 2, 1024B=1kB, 1024kB=1MB, etc.)

```
1 #!/usr/bin/env python3
2 """mapper.py"""
3
4 import sys
5
6 for line in sys.stdin:
7     row = line.split(" ")
8     ip = row[0]
9     data = row[len(row) - 1]
10    print(f'{ip}\t{data}', end="")

1 #!/usr/bin/env python3
2
3 import sys
4
5
6 number_of_requests = 0
7 total_transferred_data = 0
8
9 for line in sys.stdin:
10    ip, data = line.split("\t", 2)
11    number_of_requests += 1
12    try:
13        total_transferred_data += int(data)
14    except:
15        continue
```

```

16
17
18 def human_size(bytes, units=[' bytes','KB','MB','GB','TB', 'PB', 'EB']):
19     """ Returns a human readable string representation of bytes """
20     return str(bytes) + units[0] if bytes < 1024 else human_size(bytes>>10, units
    [1:])
21
22 #print(f"{ip}\t{number_of_requests}\t{human_size(total_transferred_data)}")
23 print(f"{number_of_requests}\t{human_size(total_transferred_data)}")

```

2) Implement a MapReduce program to provide the 5 most popular domain names (e.g. unicomp.net, letters.com or aa.net) from the client's machine names. Skip the requests having only IP address for the client's machine name

```

1 #!/usr/bin/env python3
2 """mapper.py"""
3
4 import sys
5
6 for line in sys.stdin:
7     row = line.split(" ")
8     ip = row[0]
9     data = row[len(row) - 1]
10    print(f'{ip}\t{data}', end="")

```

```

1 #!/usr/bin/env python3
2
3 import sys
4 import collections
5
6 for line in sys.stdin:
7     ip, data = line.split("\t",2)
8     counter[k] += 1
9
10 counter = collections.Counter()
11
12 print counter.most_common(5)

```

Report

GitHub: <https://github.com/it-teaching-abo-akademi/assignment-5-feliciofilipe/tree/master>

S3 Bucket: <https://s3.console.aws.amazon.com/s3/buckets/feliciobucket?region=us-east-1&tab=objects>

Conclusion

In this assignment all I learned was completely new. Here I learn how can I use cloud computing instances in a cluster to run a piece of code.