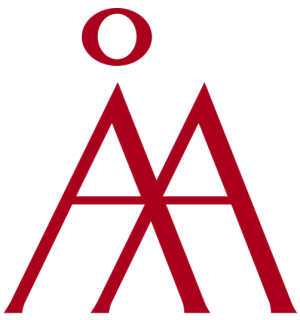

Åbo Akademi University

Cloud Computing

Assignment 5

Filipe 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.

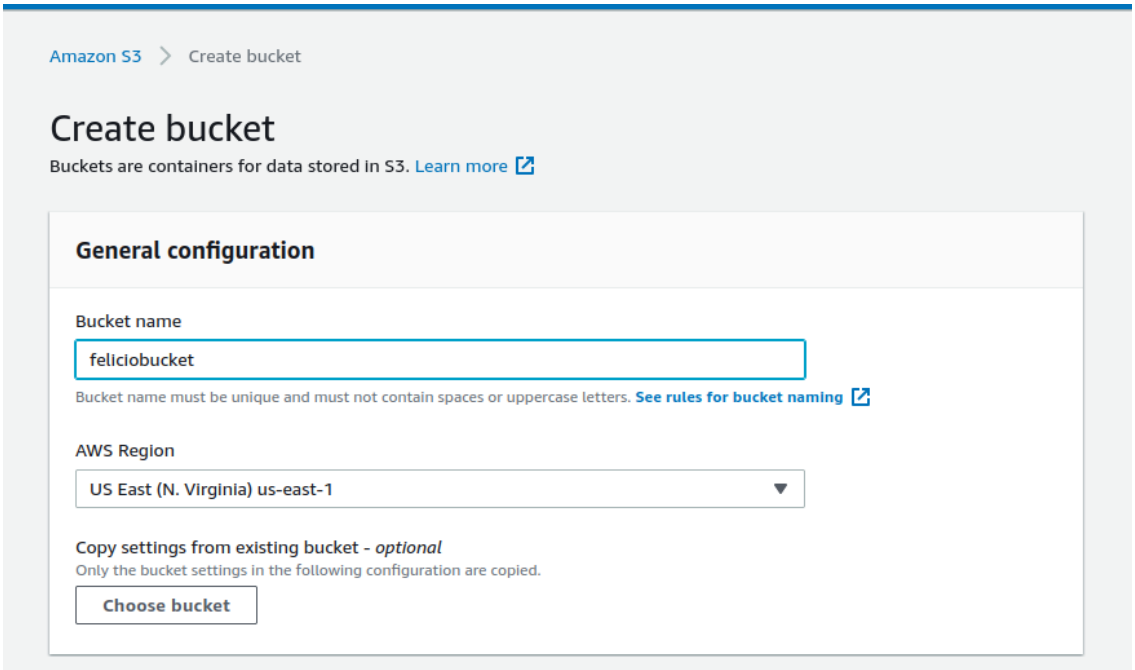

Figure 1

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.

```
#!/usr/bin/env python
"""mapper.py"""

import sys

# input comes from STDIN (standard input)
for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()
    # split the line into words
    words = line.split()
    # increase counters
    for word in words:
        # write the results to STDOUT (standard output);
        # what we output here will be the input for the
        # Reduce step, i.e. the input for reducer.py
        #
        # tab-delimited; the trivial word count is 1
        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.

```
#!/usr/bin/env python
"""reducer.py"""

import sys
import collections

counter = collections.Counter()

for line in sys.stdin:
    k, v = line.strip().split("\t", 2)

    counter[k] += int(v)

print counter.most_common(100)
```

3) Uploading your files to Amazon S3

First, I created a folder streamingcode in my bucket:

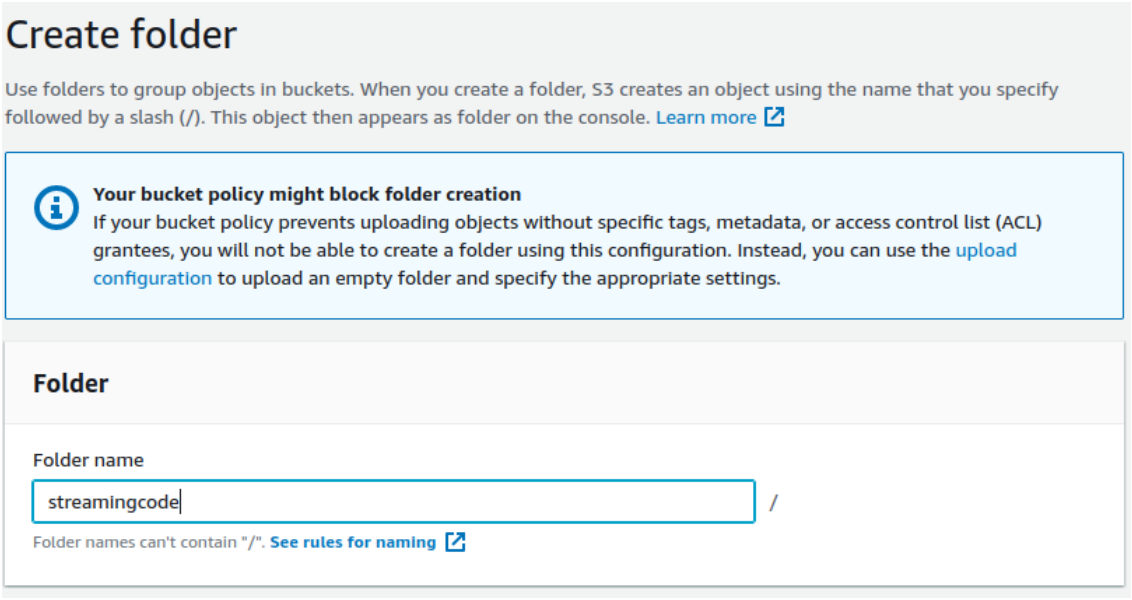

Figure 2

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

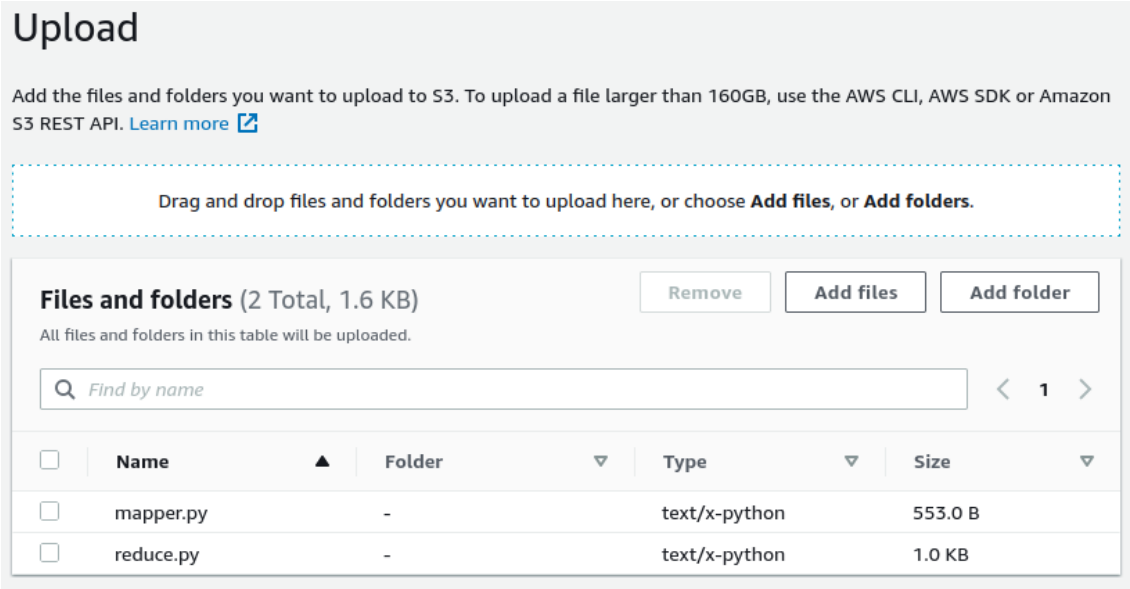

Figure 3

Then, I created a folder input in my bucket:

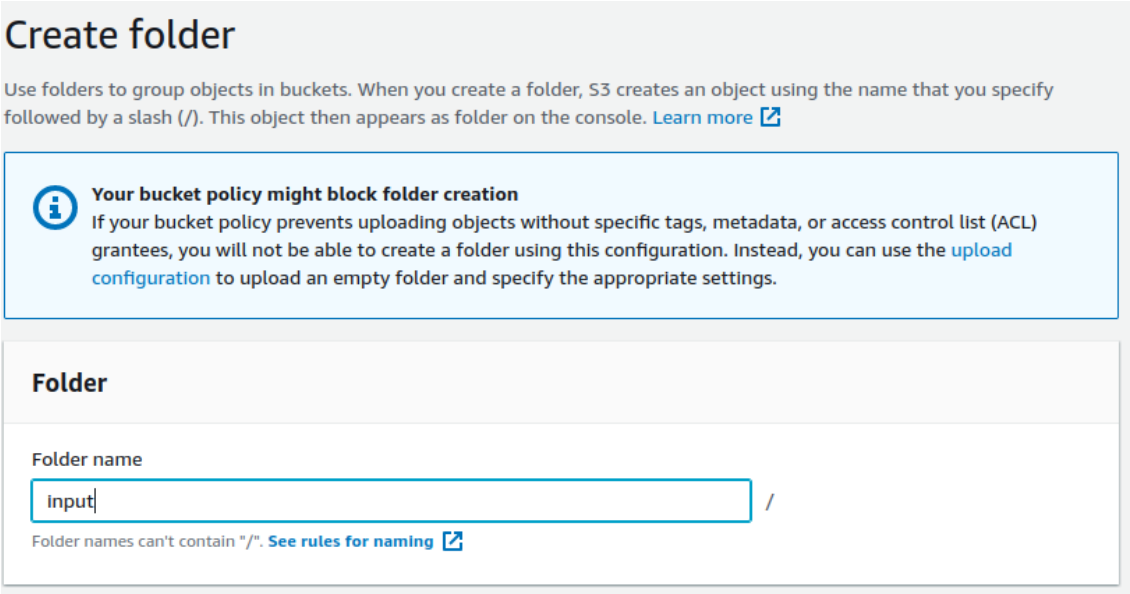

Figure 4

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

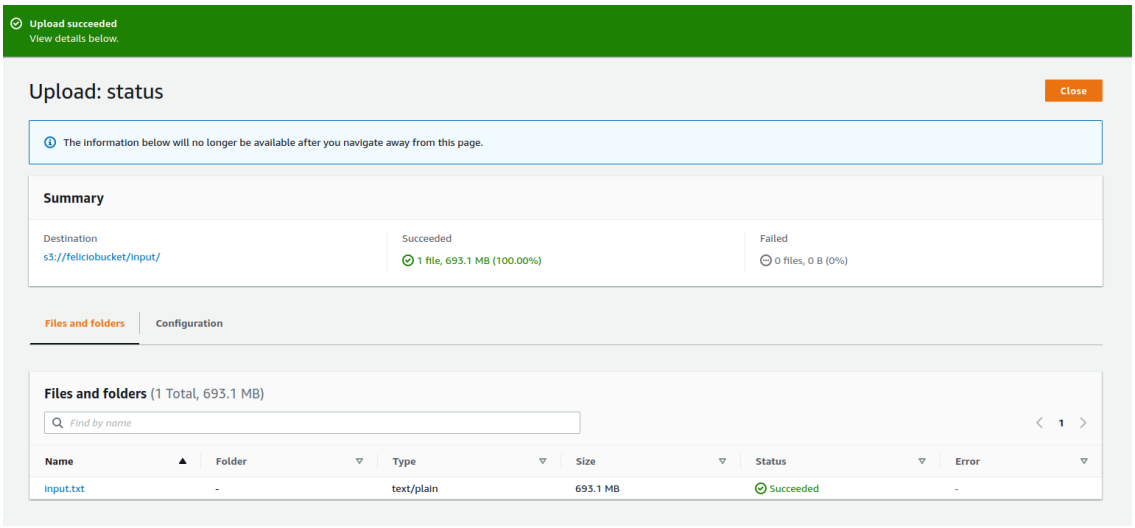

Figure 5

4) Create a Cluster

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

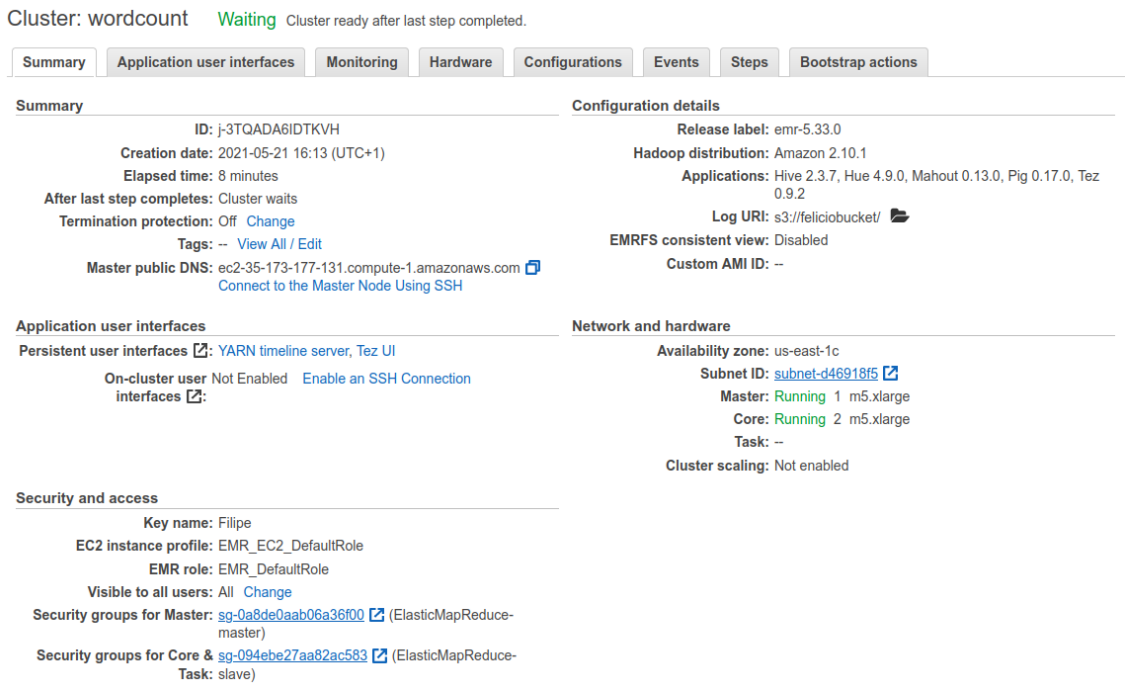

Figure 6

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

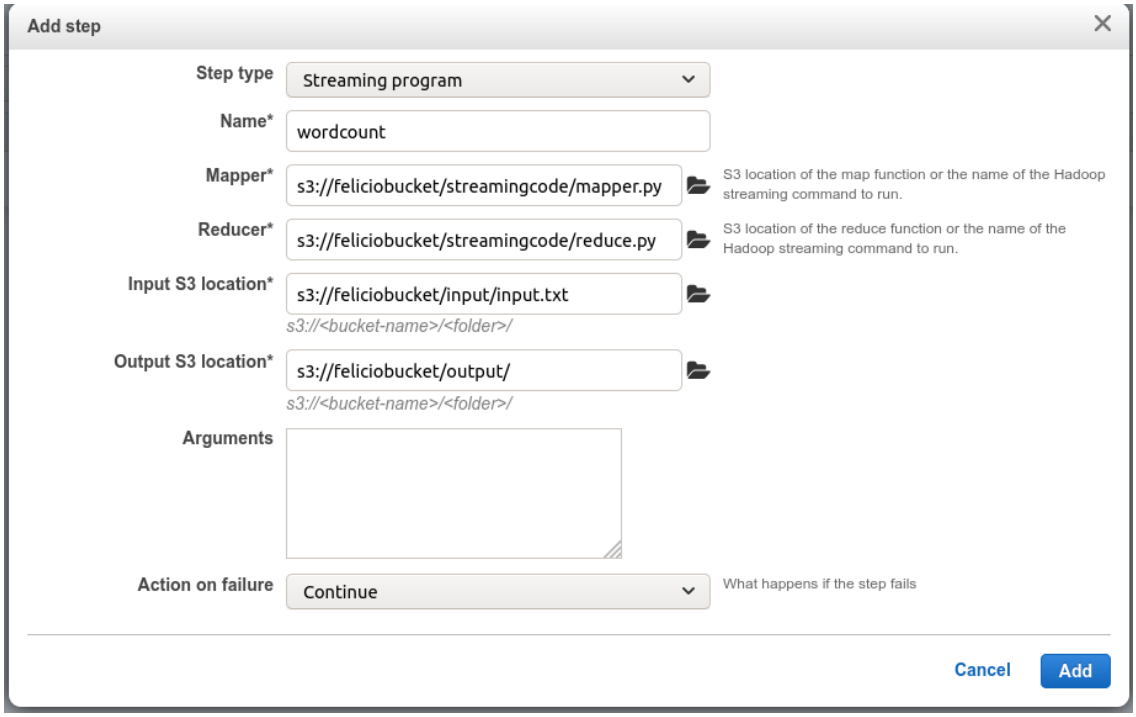

Figure 7

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

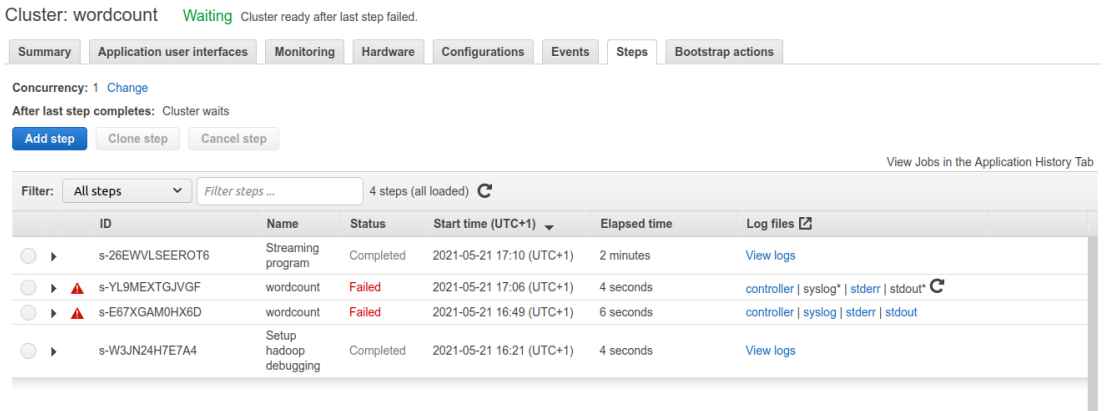

Figure 8

6) Output

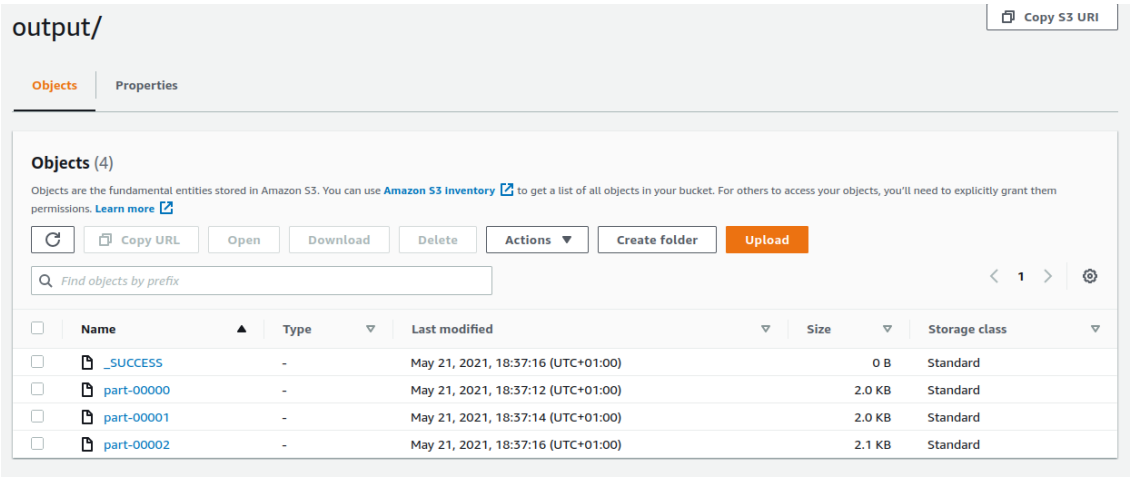

Figure 9

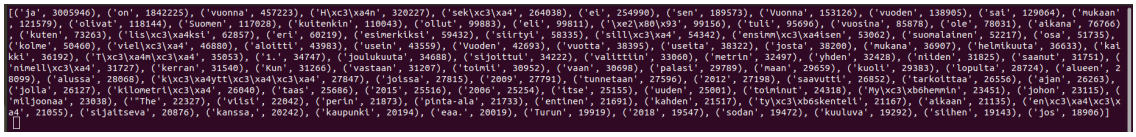

Figure 10

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))
```

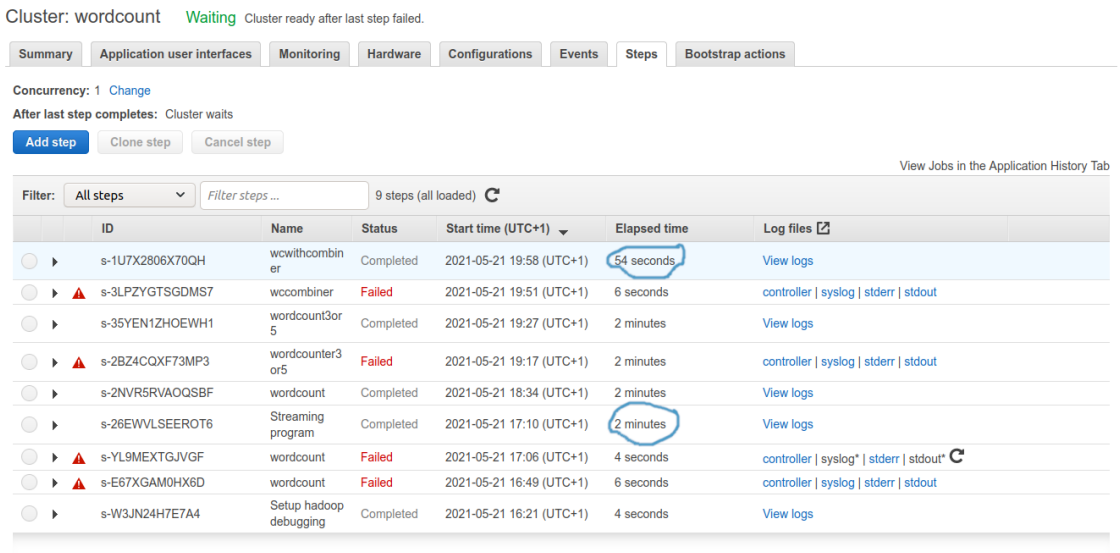

Figure 11

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)
```

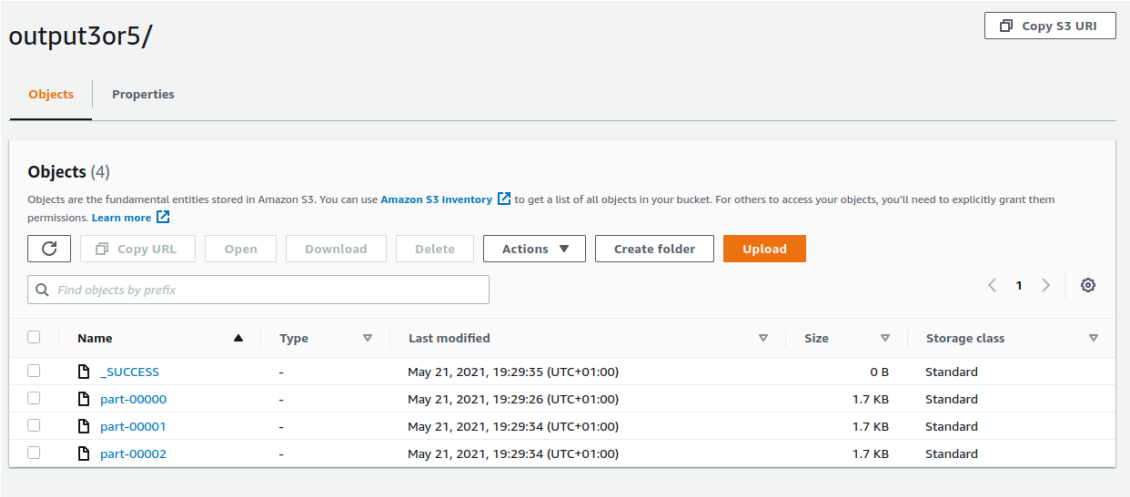

Figure 12

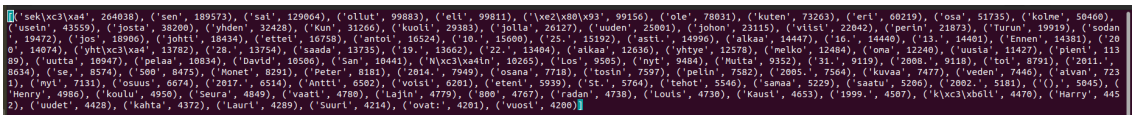

Figure 13

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.)

```python
#!/usr/bin/env python3
"""mapper.py"""

import sys

for line in sys.stdin:
    row = line.split(" ")
    ip = row[0]
    data = row[len(row) - 1]
    print(f'{ip}\t{data}', end="")
```

```python
#!/usr/bin/env phyton3

import sys


number_of_requests = 0
total_transferred_data = 0

for line in sys.stdin:
    ip, data = line.split("\t",2)
    number_of_requests += 1
    try:
        total_transferred_data += int(data)
    except:
        continue
```

```

def human_size(bytes, units=[' bytes','KB','MB','GB','TB', 'PB', 'EB']):
    """ Returns a human readable string representation of bytes """
    return str(bytes) + units[0] if bytes < 1024 else human_size(bytes>>10, units
    [1:])

#print(f"{ip}\t{number_of_requests}\t{human_size(total_transferred_data)}")
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

```
#!/usr/bin/env python3
"""mapper.py"""

import sys

for line in sys.stdin:
    row = line.split(" ")
    ip = row[0]
    data = row[len(row) - 1]
    print(f'{ip}\t{data}', end="")
```

```
#!/usr/bin/env phyton3

import sys
import collections

for line in sys.stdin:
    ip, data = line.split("\t",2)
    counter[k] += 1

counter = collections.Counter()

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.