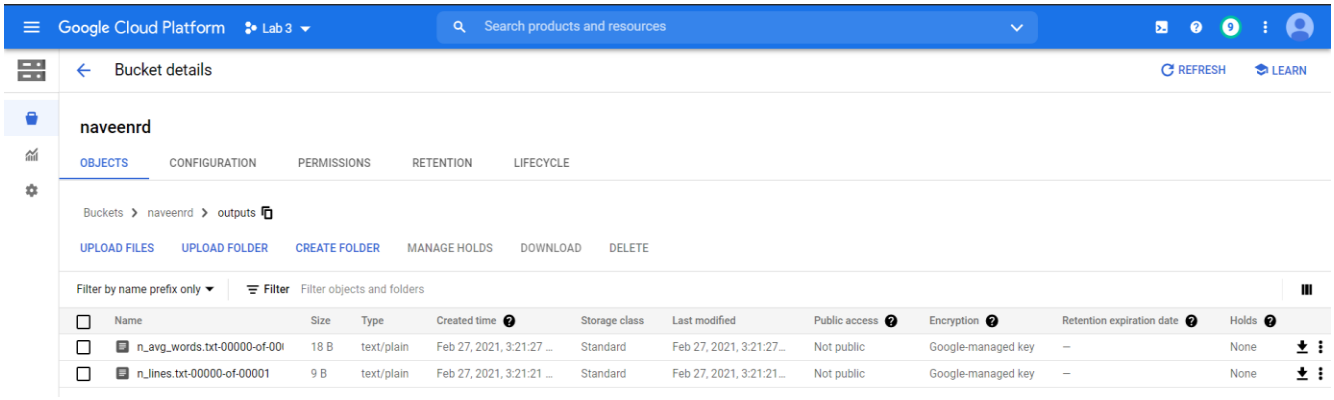
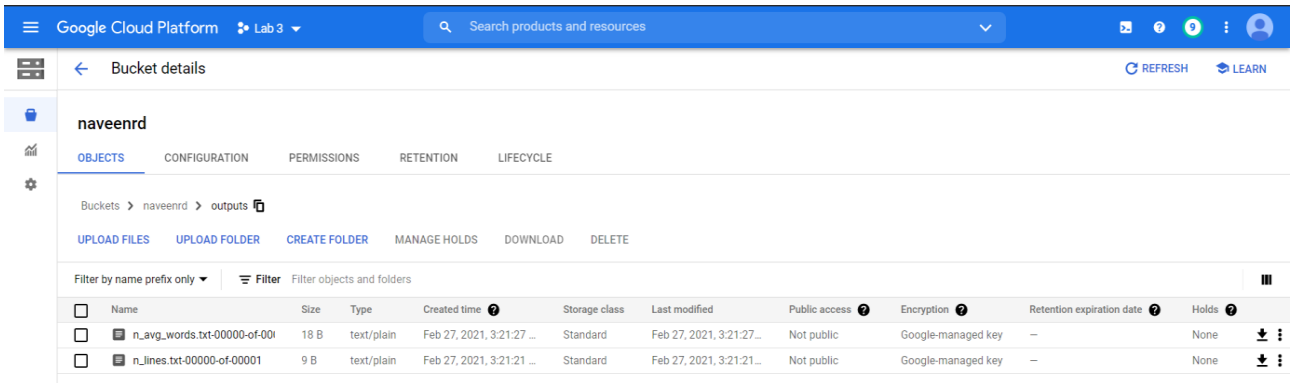
Name: Dyava Naveen Reddy **Lab - 3** **Roll:** ME17B140

1) *(The code for question 1 & 2 is written in a single file and attached with the ZIP file submission)*



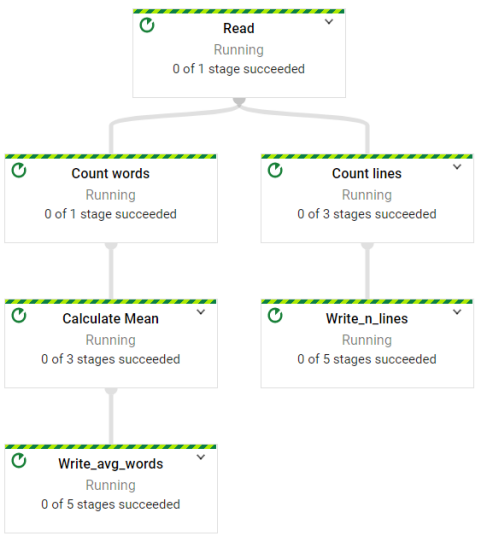
In the above screenshot ***n_lines.txt*** contains the number of lines in the given data file.
The number of words as obtained after dataflow operation = 38990771.

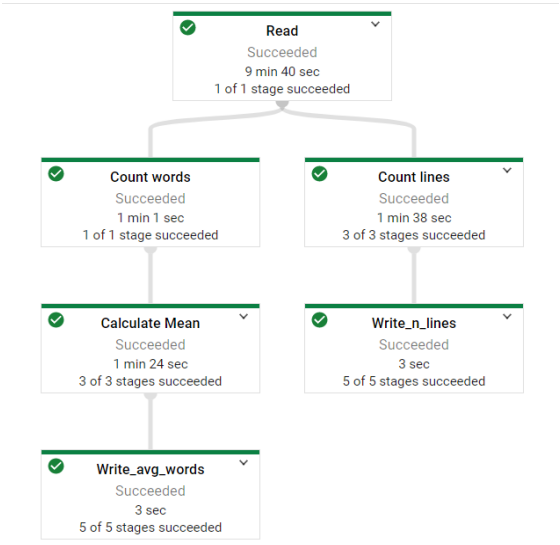
2) *(The code for question 1 & 2 is written in a single file and attached with the ZIP file submission)*



In the above screenshot ***n_avg_words.txt*** contains the average number of words per line in the file.
The average number of words per line obtained by dataflow operation = 2.000363470627447.

3)





4) The pipeline is built in such a way that calculating the number of lines & average number of words per line is done in a single run of the dataflow.

The first step is to read the text file placed at ***"gs://iitmbd/out.txt "*** *and* convert the lines into PCollection.

```
lines = p | 'Read' >> beam.io.ReadFromText('gs://iitmbd/out.txt')
```

Now we have PCollection & PCollections are immutable. That means the same PCollections can be used for various transformation operations. Here we have two tasks at hand: counting number of lines in the text file & counting average number of words per line in the text file. So, the lines PCollection is used for two branches. It can be seen in the execution graph presented in the previous question. The left branch is for calculating the average number of lines in a line & the right branch is for calculating the number is lines in the text file.

```
# The below line of code is to count the number of lines in the text file.
    n_lines = lines | 'Count lines' >>  beam.combiners.Count.Globally()
                    | 'Write_n_lines' >> beam.io.WriteToText('gs://naveenrd/outputs/n_lines.txt')

    # The below line of code is to evaluate the average number of words per line.
    n_avg_words = lines | 'Count words' >>  beam.FlatMap(lambda line: [len(line.split(' '))])
                        | 'Calculate Mean' >>  beam.combiners.Mean.Globally()
                        | 'Write_avg_words' >> beam.io.WriteToText('gs://naveenrd/outputs/n_avg_words.txt')
```

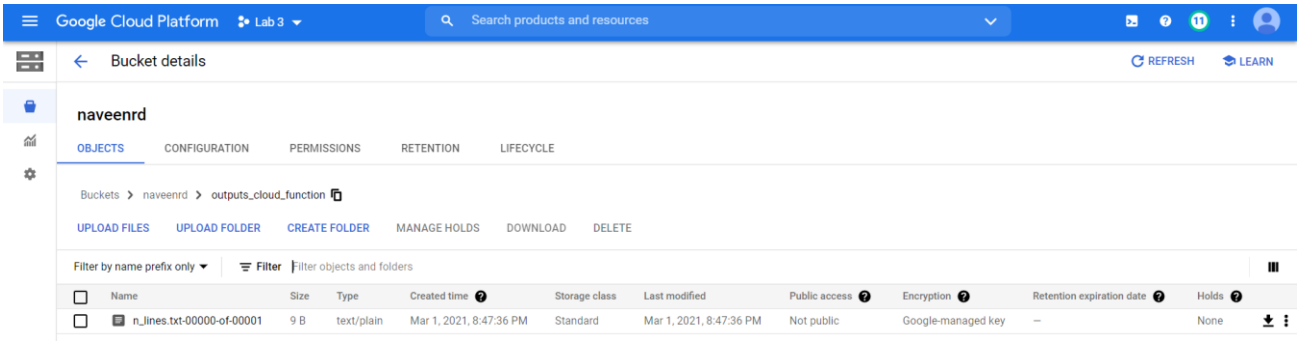
Counting Lines: We used **"beam.*combiners.Count.Globally()*"** to count the Number of elements in all the PCollections, which is basically the number of lines. Then wrote it to the cloud bucket as ***"n_lines.txt".***

Counting Average number of words/line: First we used ***Flatmap inline lambda*** function to split every line into words and output number of words as a PCollection. Then this PCollection is used with ***"beams.combiners.Mean.Globally()"*** to evaluate the average number of lines per line. This number is then written to ***'n_avg_words.txt'*** in the cloud bucket.
Bucket Name = *naveenrd* **Runner Name =** *lab-3-lines*

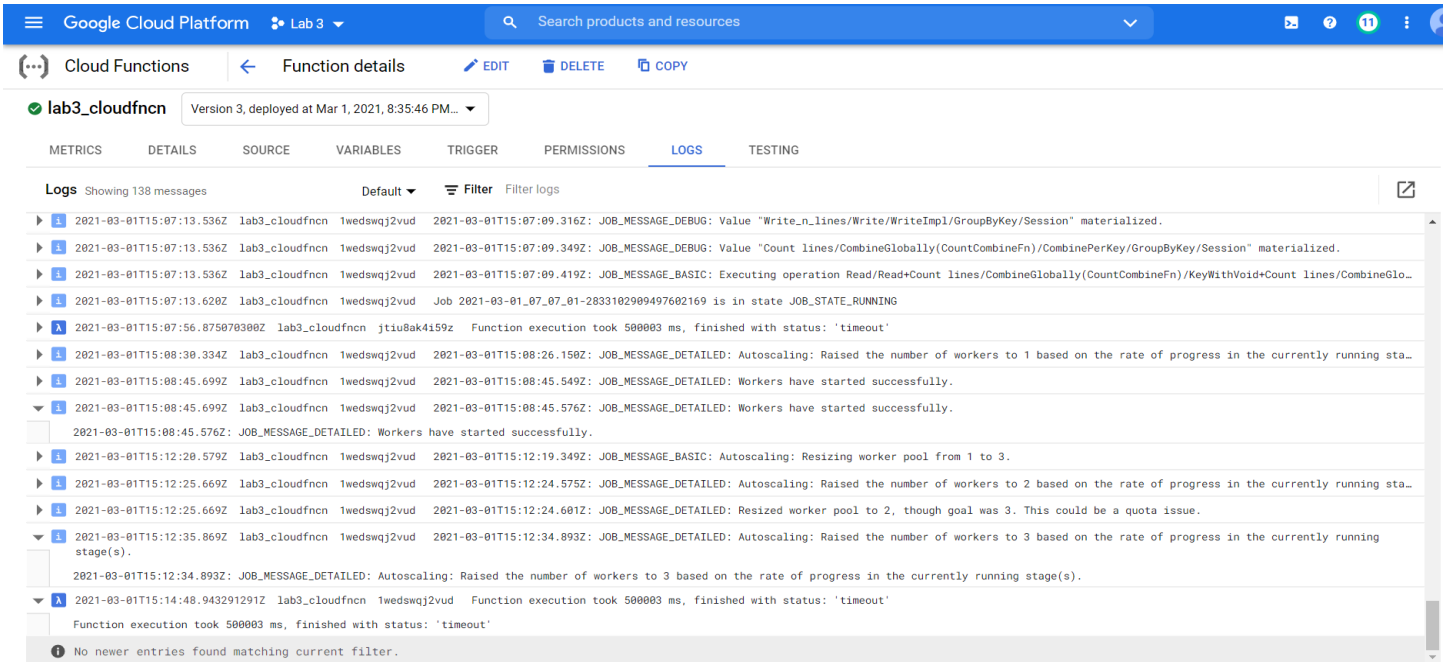
5) To run google cloud function to calculate the number of lines and average number of words per line, a separate bucket should be used for the text file because if same bucket is used for temporary folder and outputs then it will go on an infinite loop because the outputs are text files.
The **main.py** file & **requirements.txt** are placed in the cloud function folder in the zip file.

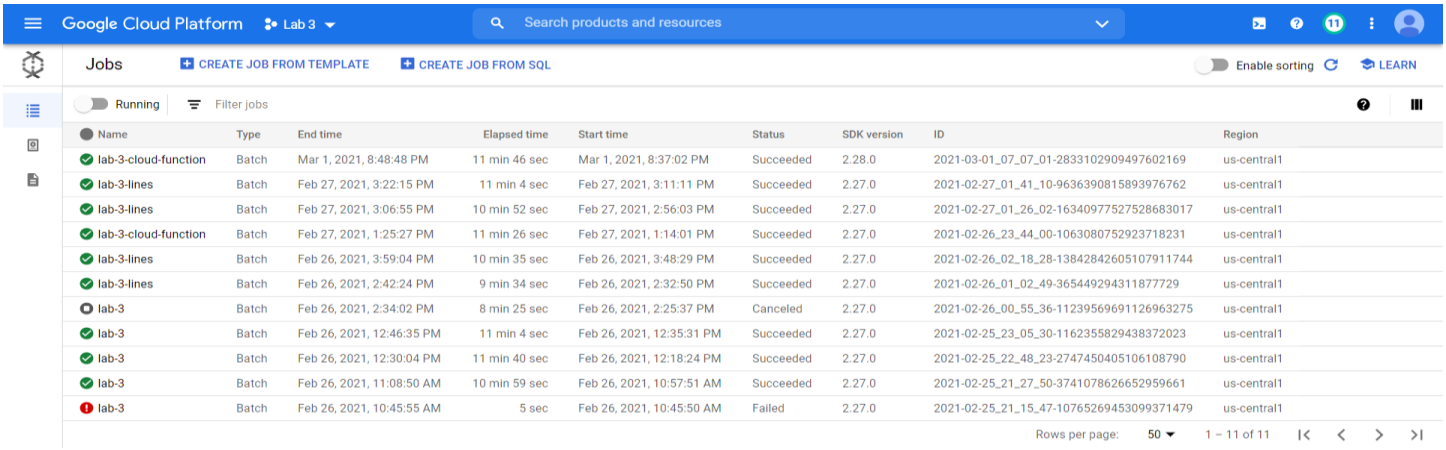
Code to deploy the cloud function.
gcloud functions deploy lab3_cloudfncn --runtime python37 --trigger-resource lab3_cloudfncn --trigger-event google.storage.object.finalize

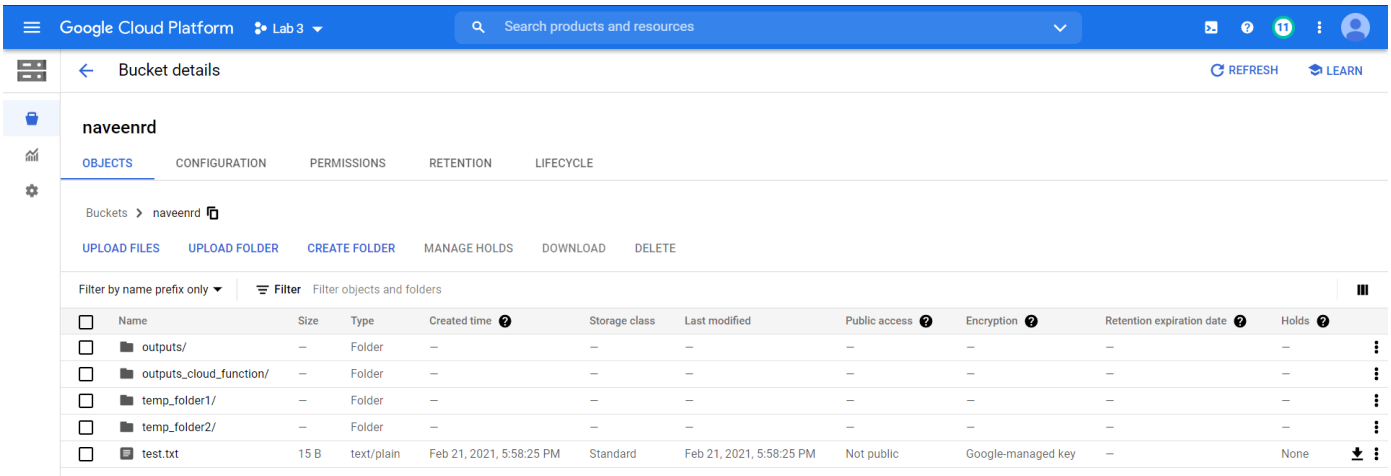
Trigger Resource Bucket: lab3_cloudfncn **Cloud Function Name:** lab3_cloudfncn



Name: Dyava Naveen Reddy **Lab - 3** **Roll:** ME17B140







Name: Dyava Naveen Reddy **Lab - 3** **Roll:** ME17B140