

# Big Data Lab - Lab 3

RA Keerthan

CH17B078

## Specifics:

*codes/LineCount.py*: Python file for question 1

*codes/AvgWordCount.py*: Python file for question 2








*codes/trigger\_dataflow*: Contains files for bonus question

1)

By running *LineCount.py* in Cloud Shell, we get *count\_lines-00000-of-00001.txt* saved in the bucket.

**Output: 38990771**

Screenshot of the file in bucket:

|  |   |                            |                                 |                                    |                          |
|--|---|----------------------------|---------------------------------|------------------------------------|--------------------------|
| <a href="#">←</a>  | Object details  | <a href="#">↓ DOWNLOAD</a> | <a href="#">✎ EDIT METADATA</a> | <a href="#">👥 EDIT PERMISSIONS</a> | <a href="#">🗑 DELETE</a> |
| Buckets > bd_lab3 > outputs > count_lines-00000-of-00001  |   |                            |                                 |                                    |                          |
| Public access  | Not public  |                            |                                 |                                    |                          |
| Type   | text/plain  |                            |                                 |                                    |                          |
| Size   | 9 B   |                            |                                 |                                    |                          |
| Created  | Feb 26, 2021, 9:44:53 AM  |                            |                                 |                                    |                          |
| Last modified  | Feb 26, 2021, 9:44:53 AM  |                            |                                 |                                    |                          |
| Hold status  | None   |                            |                                 |                                    |                          |
| Retention policy   | None  |                            |                                 |                                    |                          |
| Encryption type  | Google-managed key  |                            |                                 |                                    |                          |
| Custom time  | —   |                            |                                 |                                    |                          |
| Public URL    | Not applicable  |                            |                                 |                                    |                          |
| Authenticated URL   | <a href="https://storage.cloud.google.com/bd_lab3/outputs/count_lines-00000-of-00001">https://storage.cloud.google.com/bd_lab3/outputs/count_lines-00000-of-00001</a>  |                            |                                 |                                    |                          |
| URI   | gs://bd_lab3/outputs/count_lines-00000-of-00001    |                            |                                 |                                    |                          |


2)







By running *AvgWordCount.py* in Cloud Shell, we get *count\_words-00000-of-00001.txt* saved in the bucket.

**Output: 1.9378672455592119**

Screenshot of the file in bucket:

[←](#) [Object details](#) [↓ DOWNLOAD](#) [✎ EDIT METADATA](#) [👥 EDIT PERMISSIONS](#) [🗑 DELETE](#)

[Buckets](#) > [bd\\_lab3](#) > [outputs](#) > [count\\_words-00000-of-00001](#) 

|   |   |
|---|---|
| Public access   | Not public  |
| Type  | text/plain  |
| Size  | 19 B  |
| Created   | Feb 26, 2021, 9:59:04 AM  |
| Last modified   | Feb 26, 2021, 9:59:04 AM  |
| Hold status   | None   |
| Retention policy  | None  |
| Encryption type   | Google-managed key  |
| Custom time   | —   |
| Public URL         | Not applicable  |
| Authenticated URL  | <a href="https://storage.cloud.google.com/bd_lab3/outputs/count_words-00000-of-00001">https://storage.cloud.google.com/bd_lab3/outputs/count_words-00000-of-00001</a>  |
| URI                | <a href="gs://bd_lab3/outputs/count_words-00000-of-00001">gs://bd_lab3/outputs/count_words-00000-of-00001</a>    |

3)

Below are the screenshots of execution graphs of first two questions.

### Question 1



### Question 2



4)

### Pipeline for Question 1

We first read the input file line by line in *Read lines* step. We create an indicator function which returns 1 when a line is read in *Making indicator variable* step. Next, we sum all the ones returned by the indicator function in *Counting lines* step to count the lines read. This count is written to the output txt file in *Write results* step.

### Issues with Question 1

During the inception of the implementation, I was stuck on how to record end of line using apache beam. With the help of many print statements (by using *beam.Map(print)* ), I realized that since the input is read line by line, we could use a similar approach as given in the wordcount example by apache beam i.e, by using an indicator for end of line and summing the indicators, we would directly get the line count. This is how I came up with an indicator function which simply returns 1 if a line is read.

### Pipeline for Question 2

The input is read line by line in *Read lines* step. Each line is parsed and split into a list of words in *Split to words* step. We then create a function to return length of each of these lists, which is done in *Map words* step. Now that we have number of words for each line, we can simply take the mean of it to find the average count of words per line. This is done in *Find average* step. This average is written to the output txt file in *Write results* step.

### Issues with Question 2

The issue I faced with this question was how to get per line count of words. I tried approaches such creating a dictionary with key as line number and words as values but could not make it work. I was initially using *beam.FlatMap* to split a line to words. However, by using it, I could not stack words per line. To solve this, I used *beam.Map* which allowed me to make a list of words present in a line. On top of this, I wrote a function to return the length of this list which is nothing but number of words in that line. Finally, I took mean of number of words in each line to arrive at average count of words per line.

5)

In this question, we write a google cloud function to run a dataflow pipeline which gets triggered everytime a file is uploaded to a bucket. The job of the dataflow pipeline is to count the number of lines present in the file that triggered the cloud function.

To trigger a DataFlow pipeline using Google Cloud Functions, we need to first create a DataFlow template. This template is then staged on the bucket using the following command

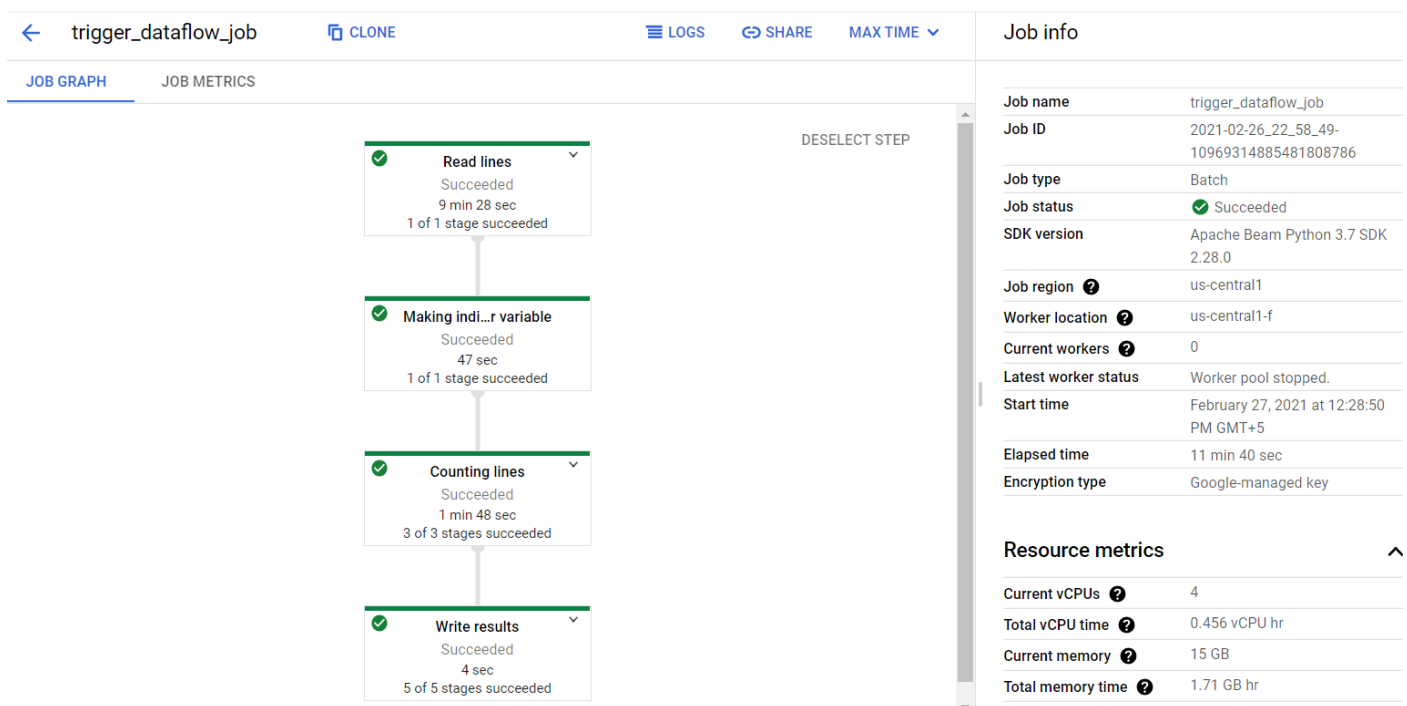
```
python -m linecount_template
```

Once the template is staged, we write the code for the function that triggers the dataflow pipeline that we have staged as dataflow template. This function is deployed using the command

```
gcloud functions deploy trigger_count_lines --runtime python37 --trigger-resource bd_lab3 --trigger-event google.storage.object.finalize
```

The codes for dataflow template and cloud function shall be found in [codes/trigger\\_dataflow/](#).

The screenshot of the job and execution graph are as follows



A separate bucket is created for saving outputs of the cloud function. This is to avoid the triggers going into an infinite loop. The output file is generated in the path `gs://trig_bucket/gcf_outputs/out-00000-00001.txt`