Assignment 4

Part B

Meet Patel (B00899516)

Dalhousie University

Subject

CSCI 5410 (Serverless Data Processing)

Professor

Dr. Saurabh Dey

Project Git Repository

Gitlab Repository Link: https://git.cs.dal.ca/patel13/csci5410_b00899516_meet_patel.git

Screenshots of all the steps performed

Initial Google Cloud Platform Setup

Figure 1 to 11 shows the prerequisite steps that are required to do build this application. These steps are performed on google cloud platform.

Figure 1 shows the creation of new project on GCP with name "**CSCI5410-Assignment4-PartB**".

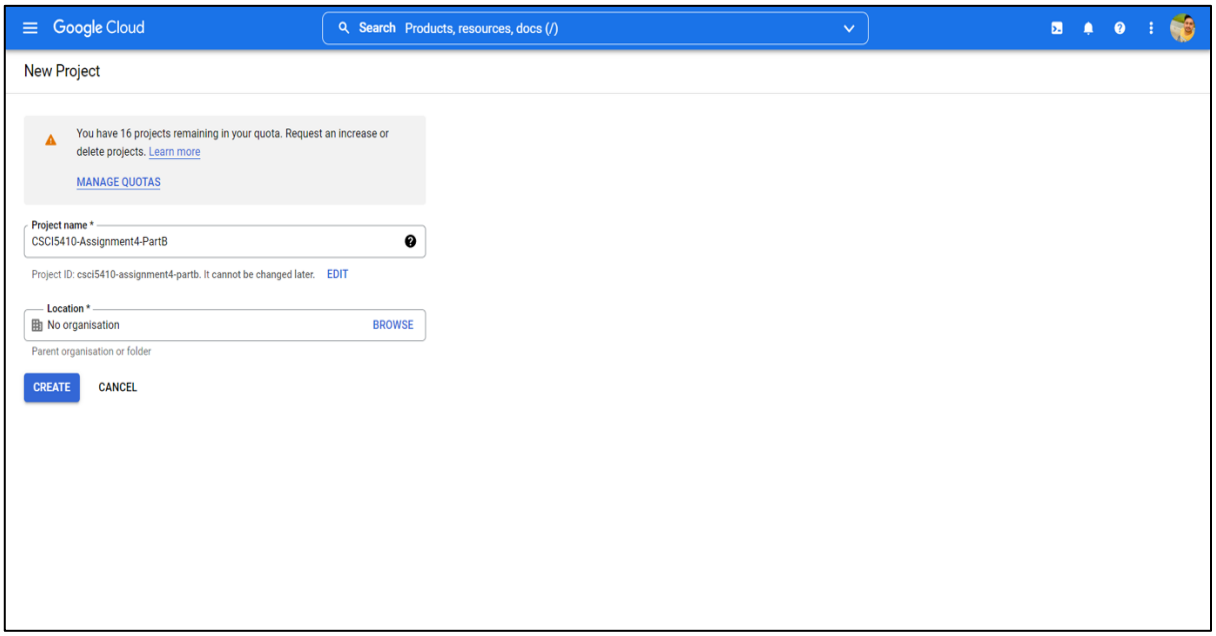


Figure 1: Creation of CSCI5410-Assignment4-PartB in Google Cloud Platform (GCP)

Figure 2 shows the dashboard of the newly created project "**CSCI5410-Assignment4-PartB**"

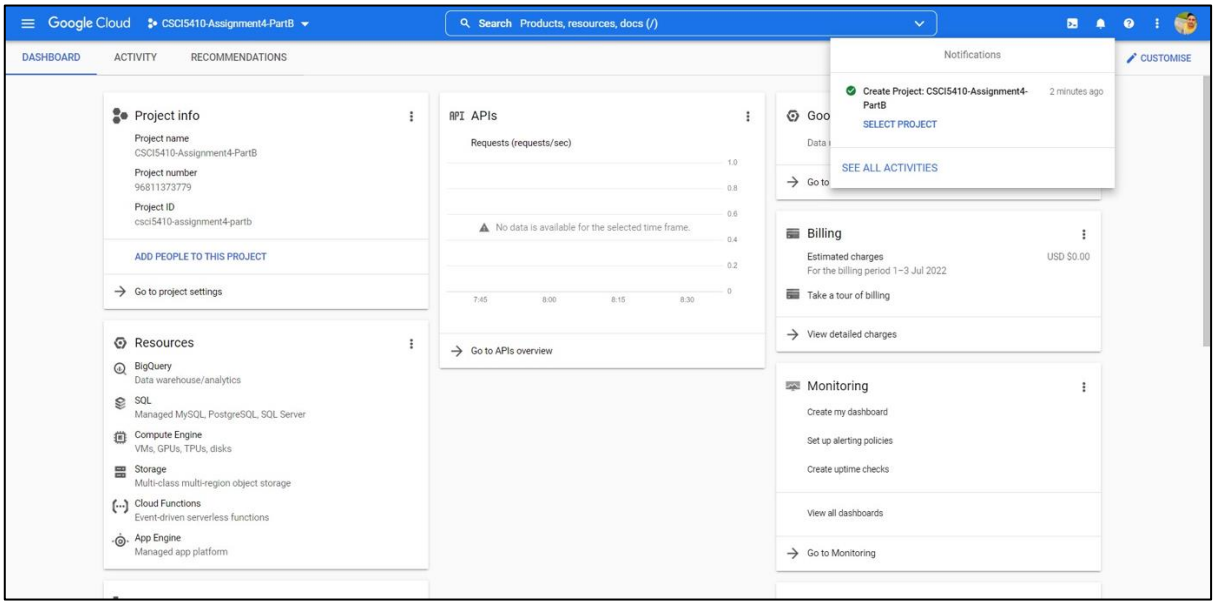


Figure 2: Dashboard of the newly created project "CSCI5410-Assignment4-PartB"

Once the creation of new project on google cloud platform is completed, we need to create a service account on GCP which helps us to connect our application with various other google cloud services.

Figure 3 to 5 represents the steps that are required to create a new service account with name "**CSCI5410-Assignment4-PartB-SA**" Inside the google cloud project with name "**CSCI5410-Assignment4-PartB**"

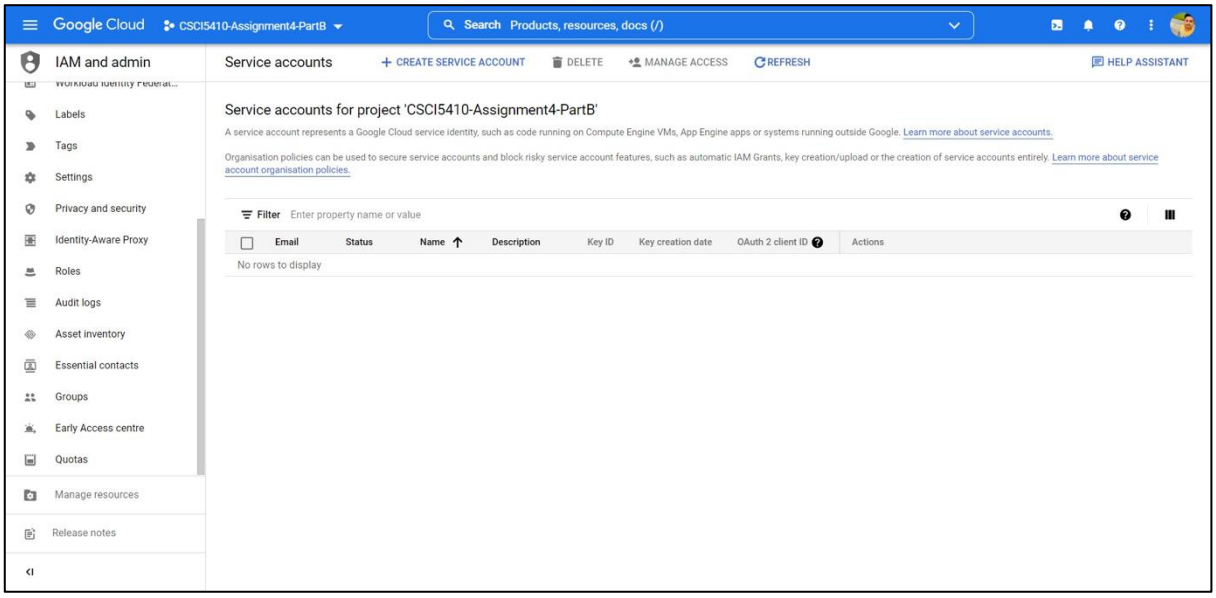


*Figure 3: Creation of service account for GCP project " **CSCI5410-Assignment4-PartB** "*

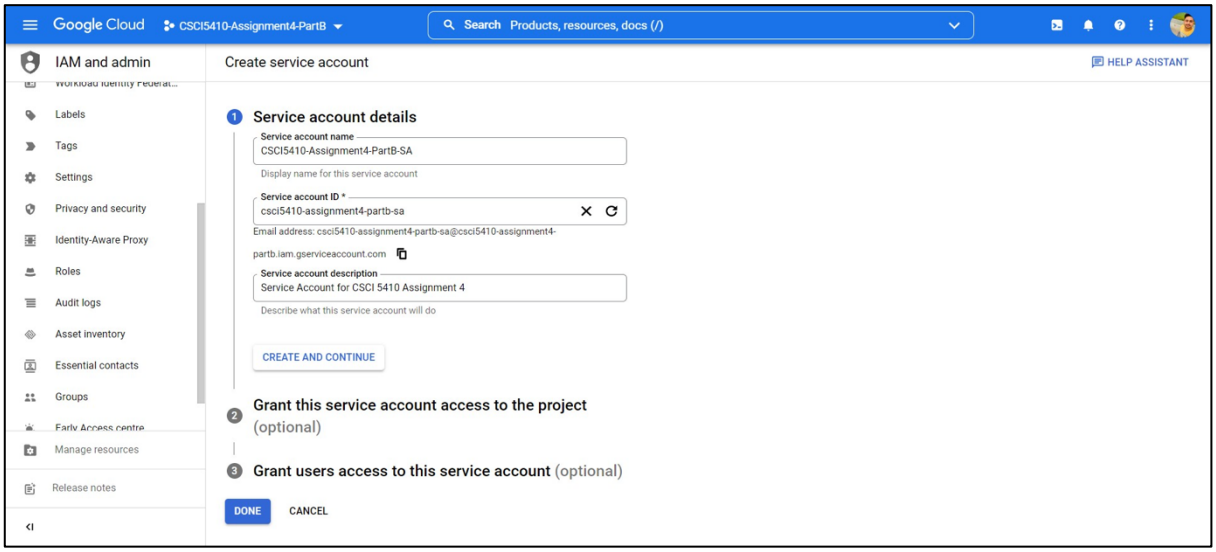


*Figure 4: Creation of service account for GCP project " **CSCI5410-Assignment4-PartB** "*

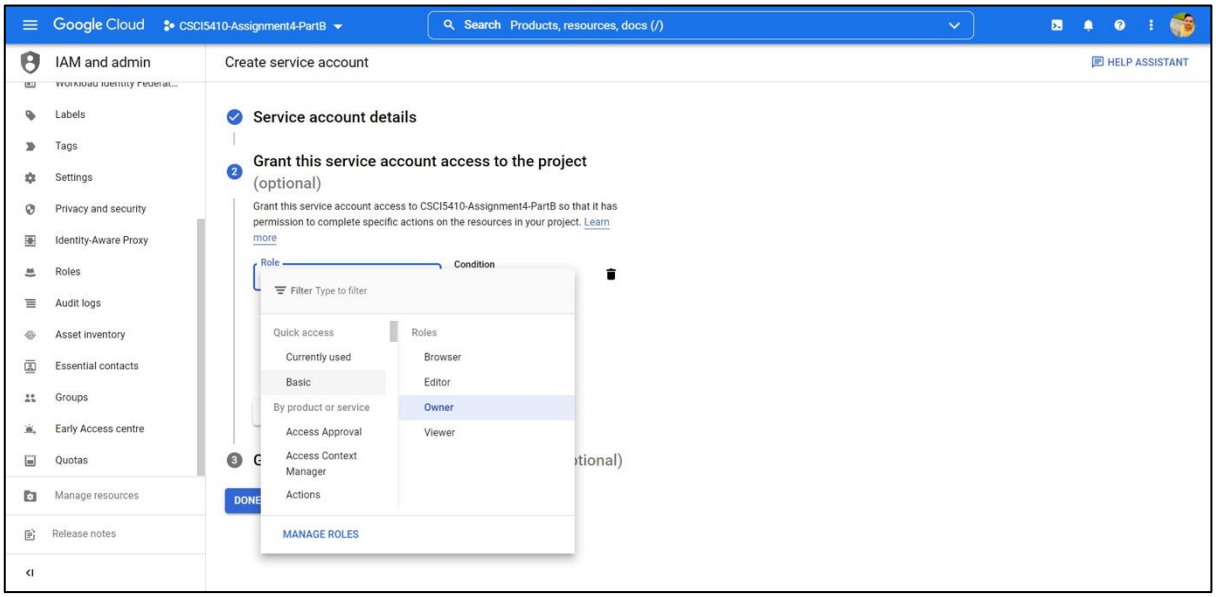


*Figure 5: Creation of service account for GCP project " **CSCI5410-Assignment4-PartB** "*

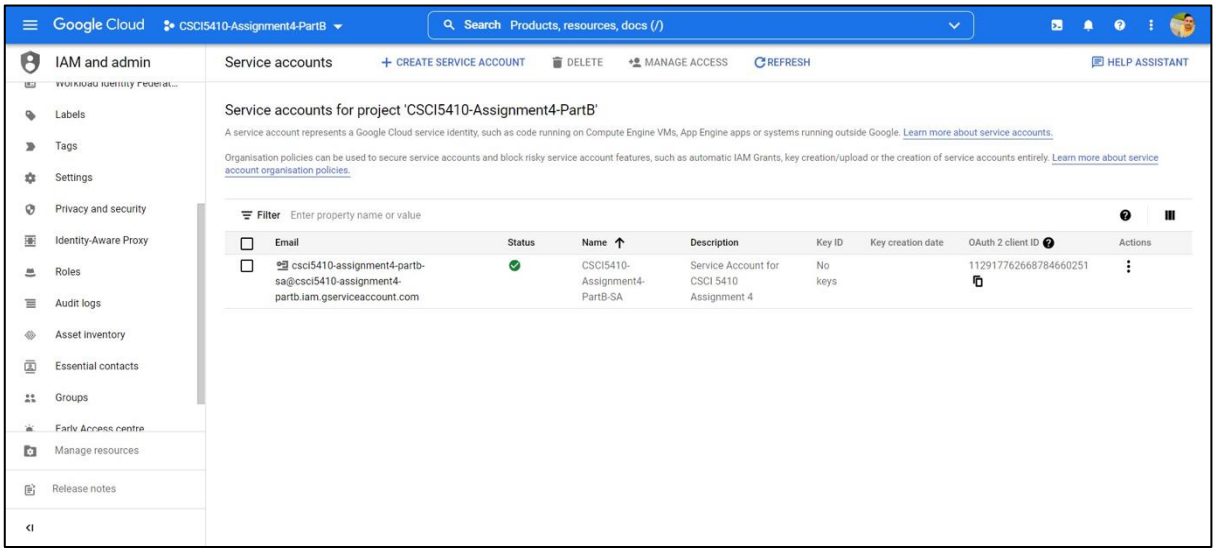


*Figure 6: Creation of service account for GCP project " **CSCI5410-Assignment4-PartB** "*

Once the creation of service account "**CSCI5410-Assignment4-PartB-SA**" is completed, we need to create a key pair using the available option "**Add Key**" which generates a JSON file. The content of this JSON file is showed below.

```
{
  "type": "type",
  "project_id": "project_id",
  "private_key_id": "private_key_id",
  "private_key": "private_key",
  "client_email": "client_email",
  "client_id": "client_id",
  "auth_uri": "hauth_uri",
  "token_uri": "token_uri",
  "auth_provider_x509_cert_url":
"auth_provider_x509_cert_url",
  "client_x509_cert_url": "client_x509_cert_url"
}
```

Figure 7 is responsible for displaying the creation of JSON file or key pair inside the "**CSCI5410-Assignment4-PartB-SA**" service account using the available option "**Add Key**".

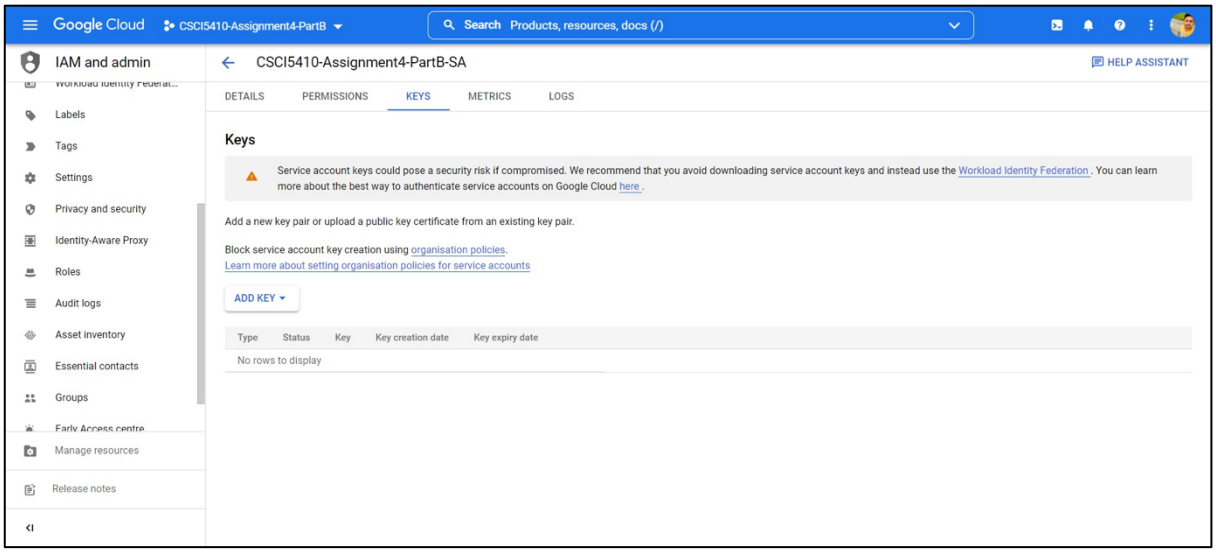


*Figure 7: Creation of key Pair for service account "**CSCI5410-Assignment4-PartB-SA**"*

Figure 8 shows the successful creation of key pair in the "**CSCI5410-Assignment4-PartB-SA"** service account.

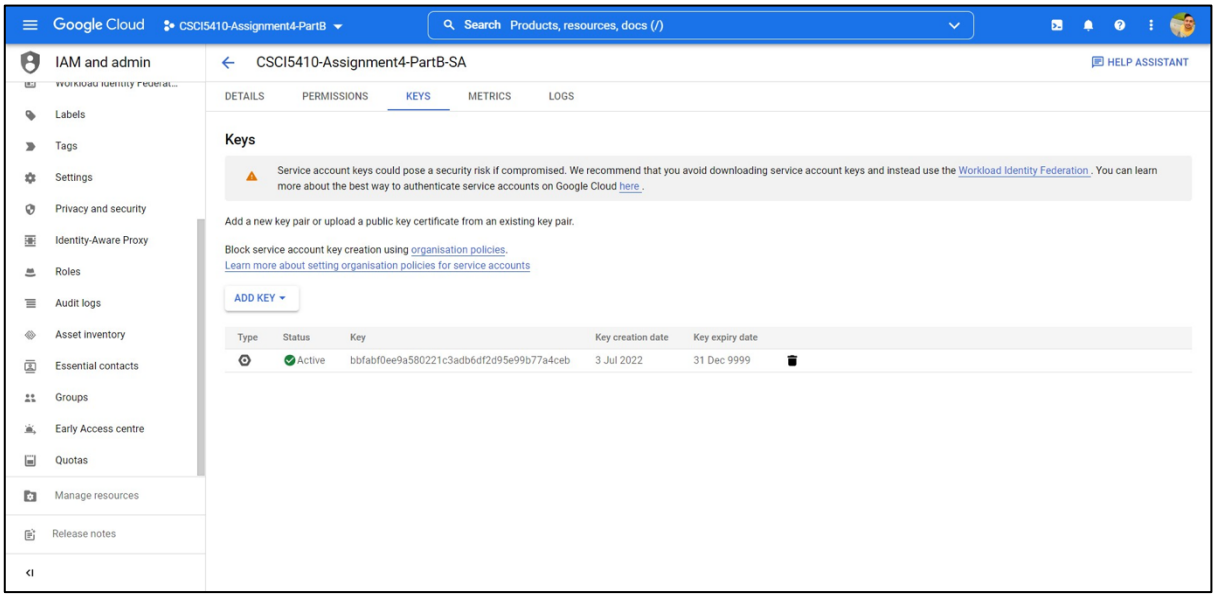


*Figure 8: Successful creation of Key Pair for service account "**CSCI5410-Assignment4-PartB-SA**"*

Now we need to enable two APIs in **CSCI5410-Assignment4-PartB** project.

1. Cloud Functions API - cloudfunctions.googleapis.com
2. Google Cloud Build API - cloudbuild.googleapis.com

Figure 9 shows Cloud Function API and **Figure 10** shows enabled Cloud Function API

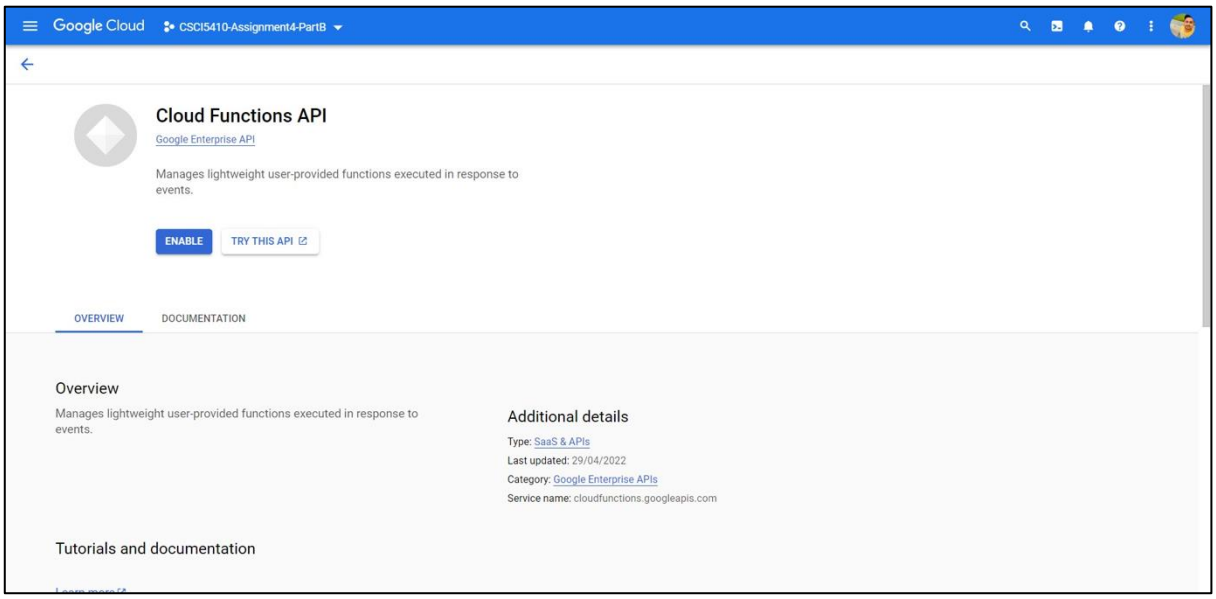


Figure 9: Could Function API

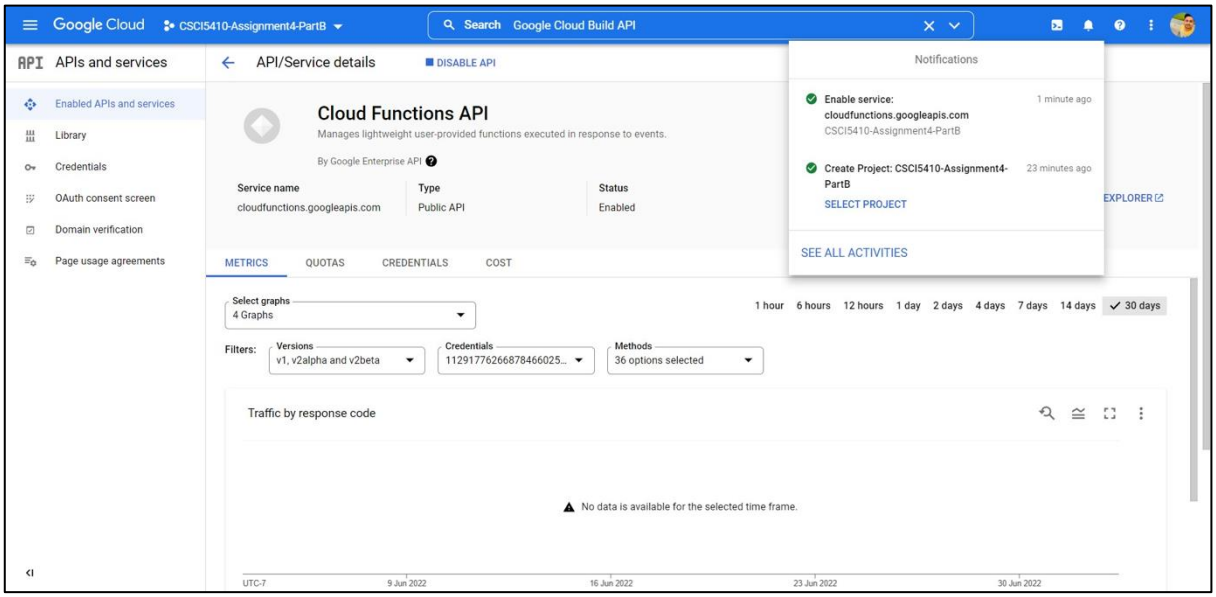


Figure 10: Enabled Cloud Function API

Figure 11 shows Google Cloud build API and **Figure 12** Shows the enabled Cloud Build API.

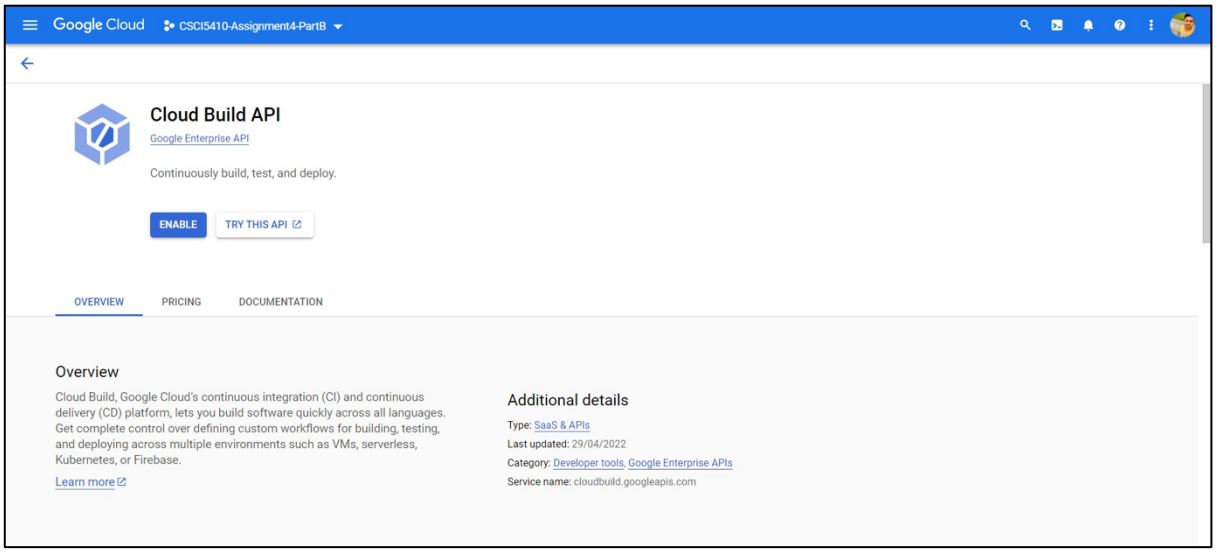


Figure 11: Cloud Build API

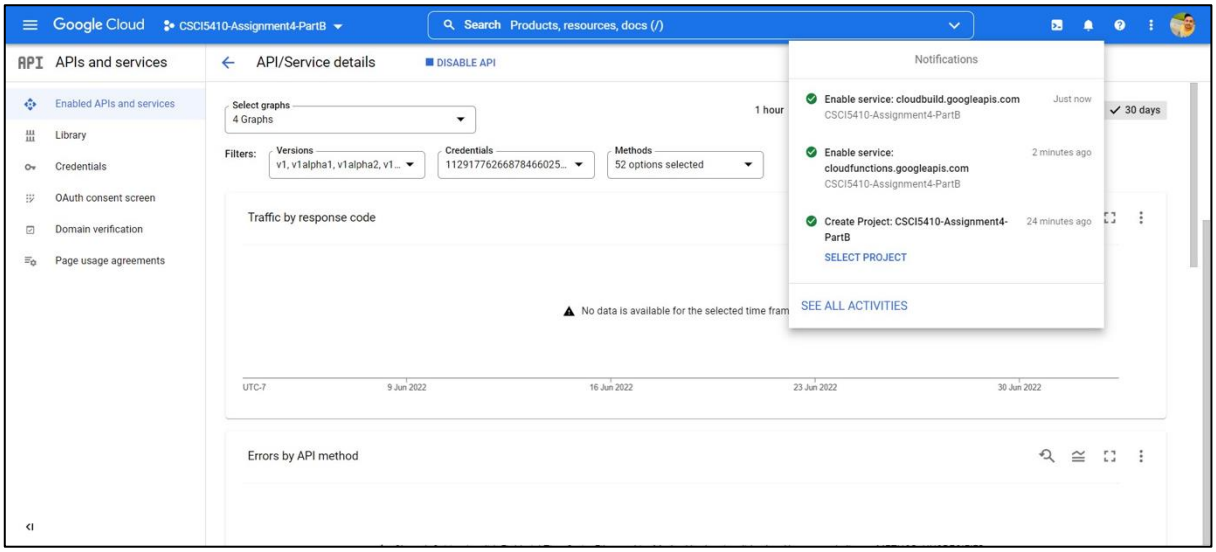


Figure 12: Enabled Cloud Build API

A. Create your 1st storage bucket SourceDataB00xxxxxx and upload the files (from 001 to 299) given in the Train folder. You need to write a script or use the SDK to upload the files on the bucket

Figure 13 shows the empty Cloud Storage with no bucket initially

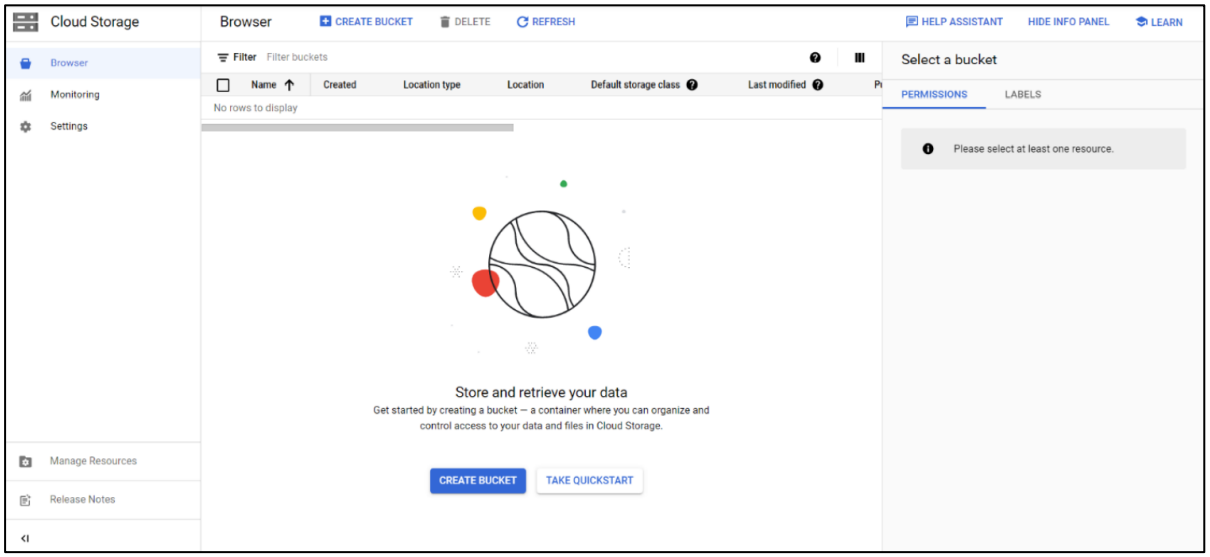


Figure 13: Cloud Function with no bucket

Figure from 13 to 17 shows the execution of program that creates the bucket with name "**sourcedatab00899516**" in google cloud storage. This Program will create this bucket "**sourcedatab00899516**" in when there is no bucket exists with the name "**sourcedatab00899516**". This program will check the cloud storage before creating the bucket. After creating the bucket, the program will upload all the files inside the **Dataset\Train** folder to the "**sourcedatab00899516"** bucket one by one.

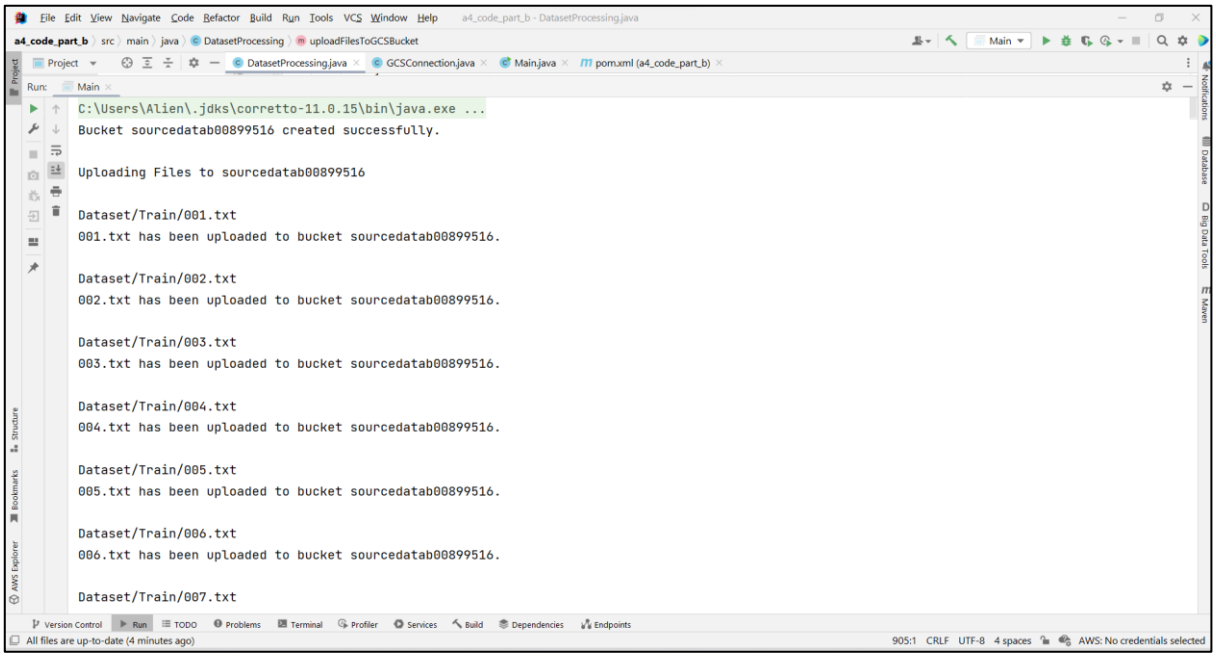


*Figure 14: Bucket "**sourcedatab00899516**" created successfully and then files from train folder uploaded to that bucket*

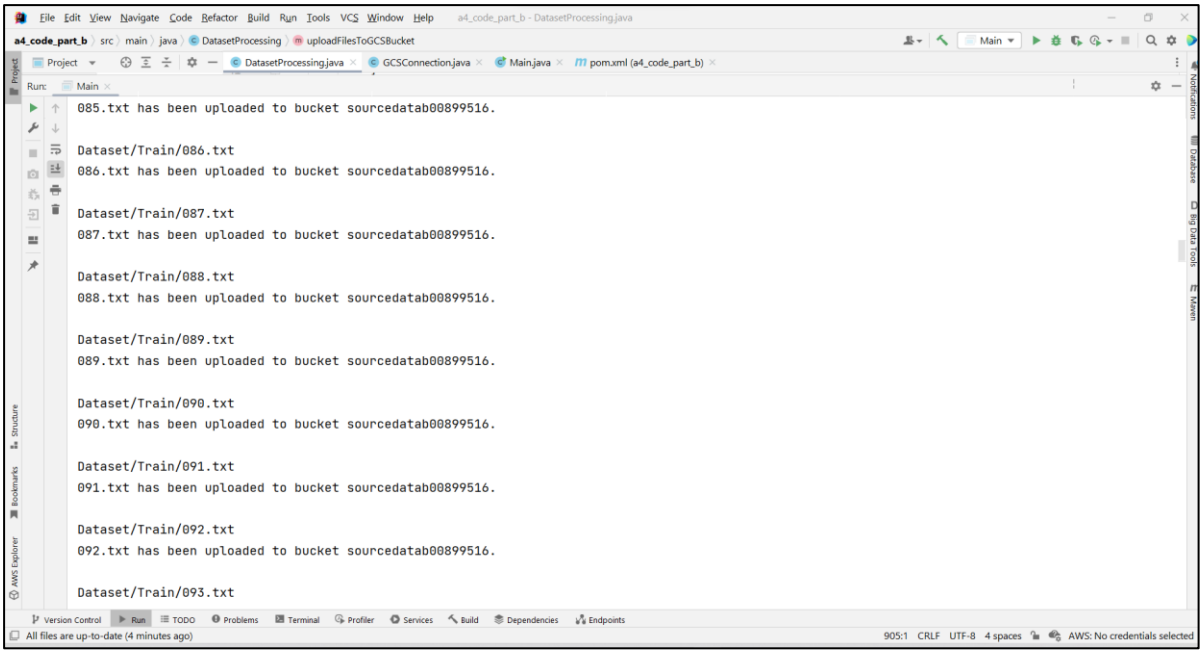


*Figure 15: Bucket "**sourcedatab00899516**" created successfully and then files from train folder uploaded to that bucket*

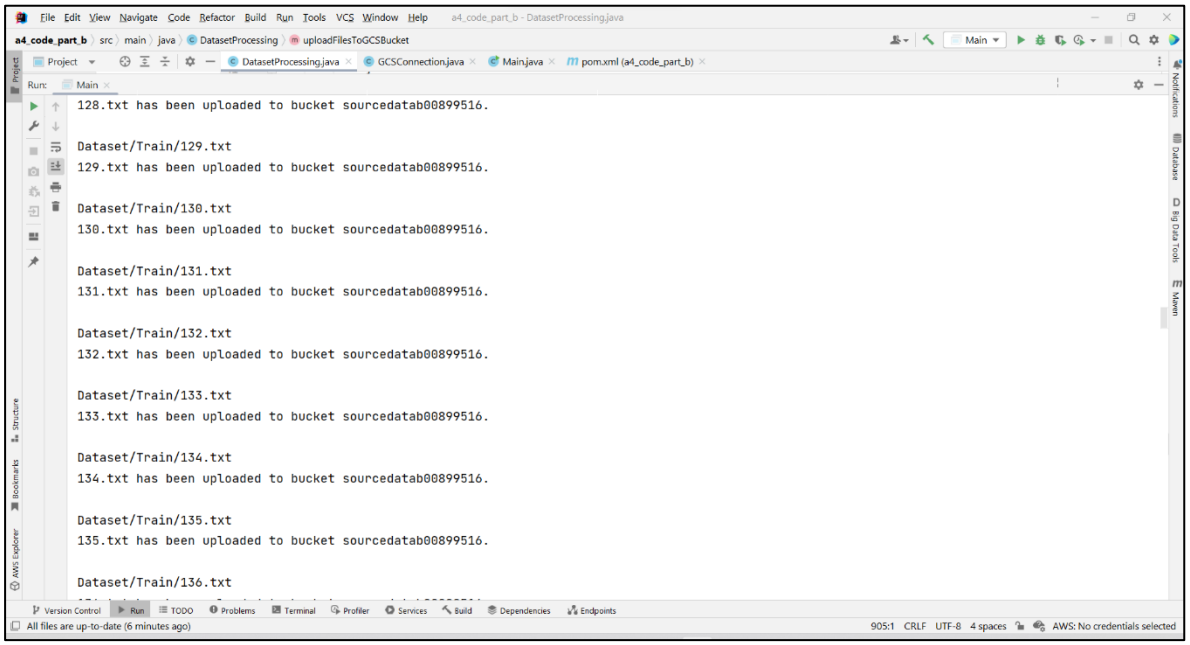


*Figure 16: Bucket "**sourcedatab00899516**" created successfully and then files from train folder uploaded to that bucket*

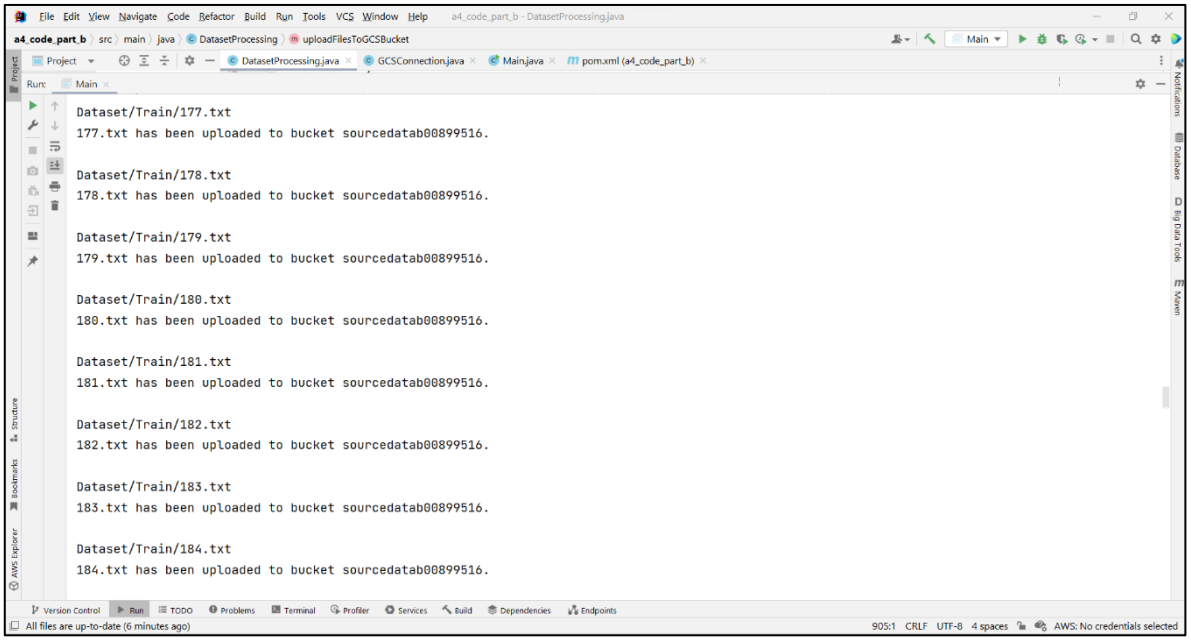


*Figure 17: Bucket "**sourcedatab00899516**" created successfully and then files from train folder uploaded to that bucket*

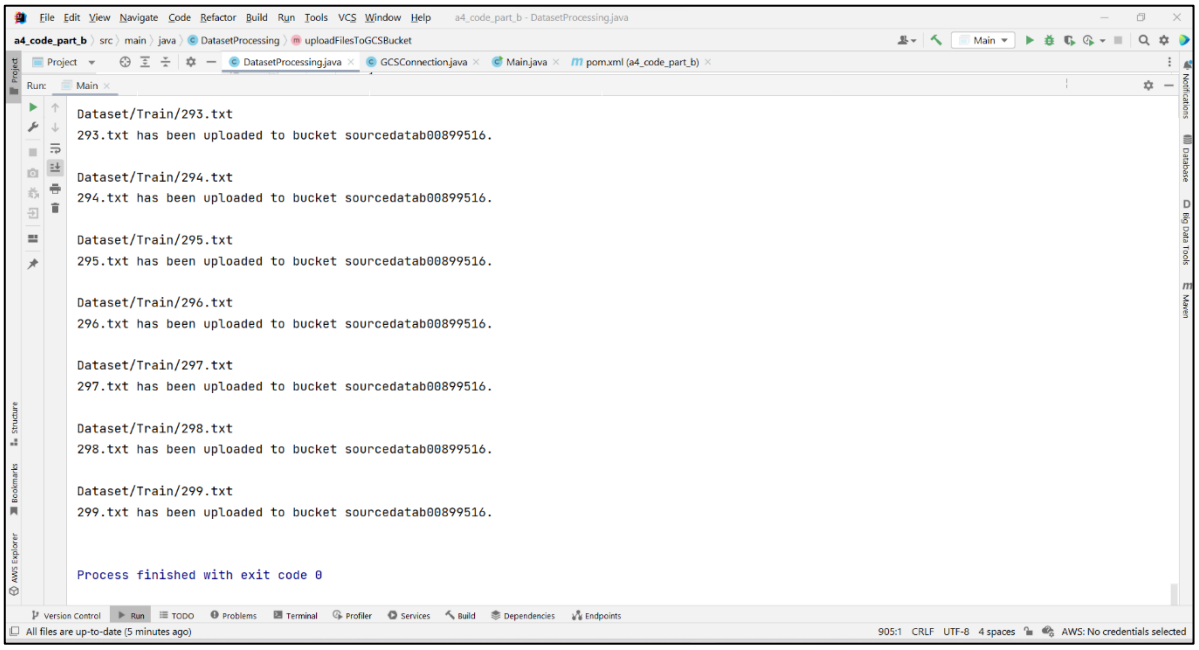


*Figure 18: Bucket "**sourcedatab00899516**" created successfully and then files from train folder uploaded to that bucket*

Figure 18 shows the bucket "**sourcedatab00899516**" created successfully on google storage under the project name **CSCI5410-Assignment4-PartB**

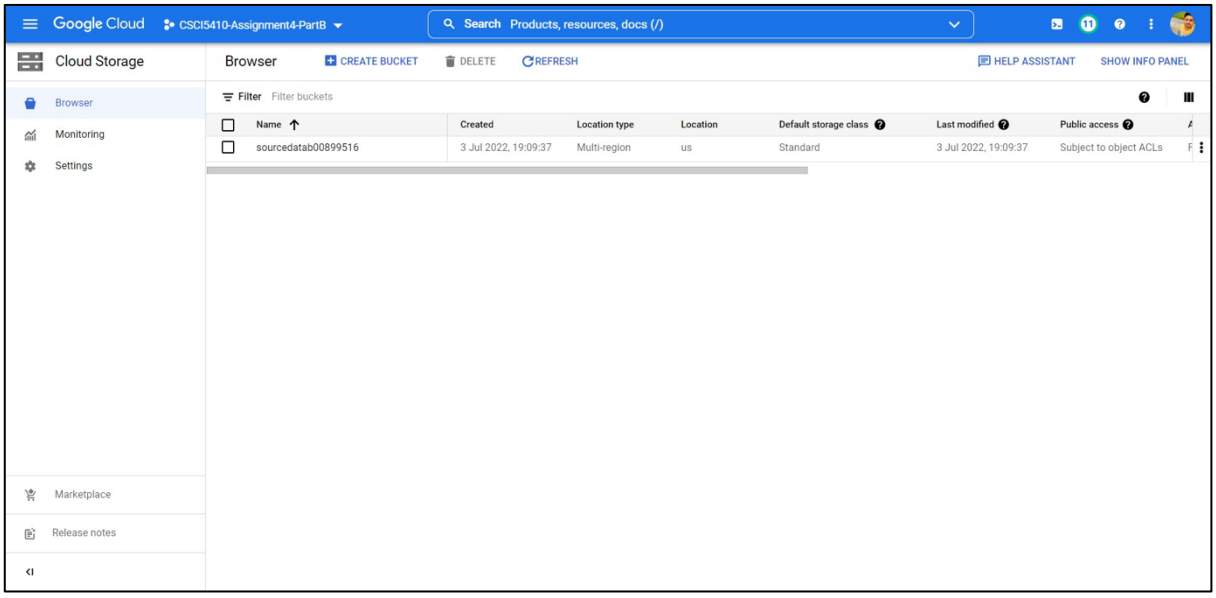


*Figure 19: Bucket "**sourcedatab00899516**" created successfully on Google Cloud Storage*

Figure 19 to 23 are responsible for showing the uploaded file on "**sourcedatab00899516**" bucket. These files are uploaded from Train folder. There are told 299 files (001.txt to 299.txt) which are uploaded to this bucket.

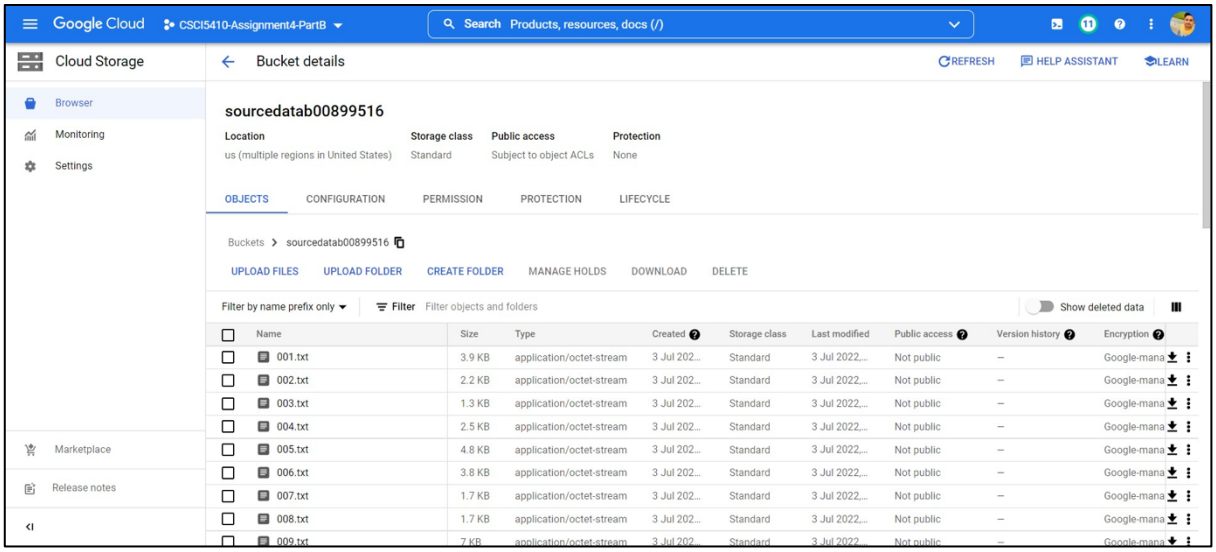


*Figure 20: Files from the train folder uploaded to "**sourcedatab00899516**" bucket*

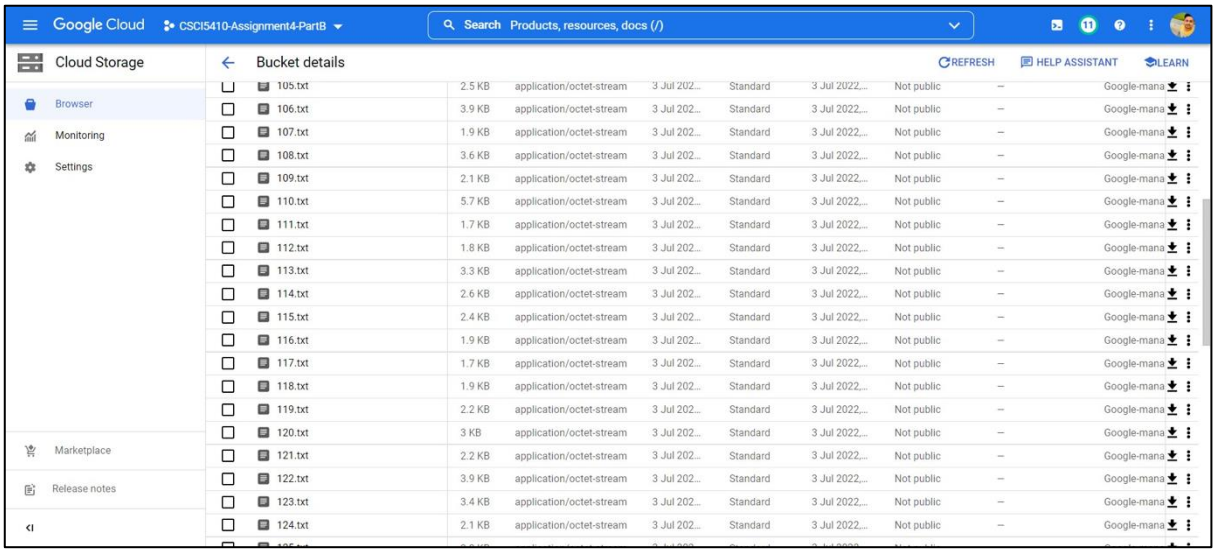


*Figure 21: Files from the train folder uploaded to "**sourcedatab00899516**" bucket*

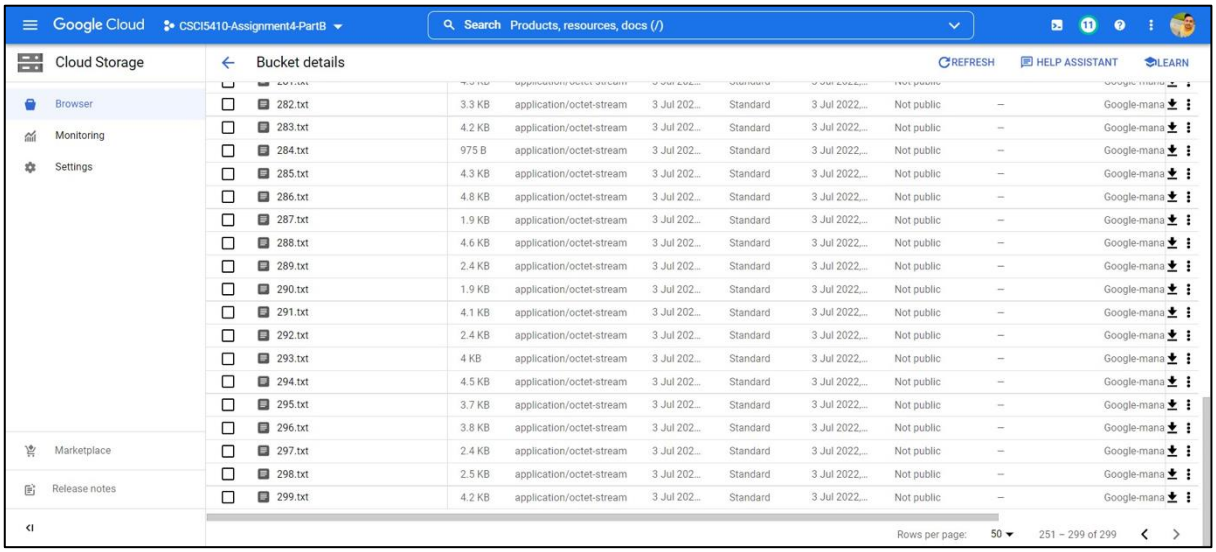


*Figure 22: Files from the train folder uploaded to "**sourcedatab00899516**" bucket*

B. Once a file is uploaded, a cloud function - "generateVector" should extract words from all the files (remove the stop words). Then compute Levenshtein distance between the Current, and Next word.

Figure 22 is responsible for representing the creation of cloud function with name **generateVector** and the memory allocated to this cloud function is 512 MB and the timeout is set to 540 seconds. With an event type Finalize/Create, a trigger is created and set it to the cloud storage bucket "**sourcedatab00899516**". So, whenever a new file is uploaded to the "**sourcedatab00899516**" bucket, the **generateVector** cloud function will be executed.

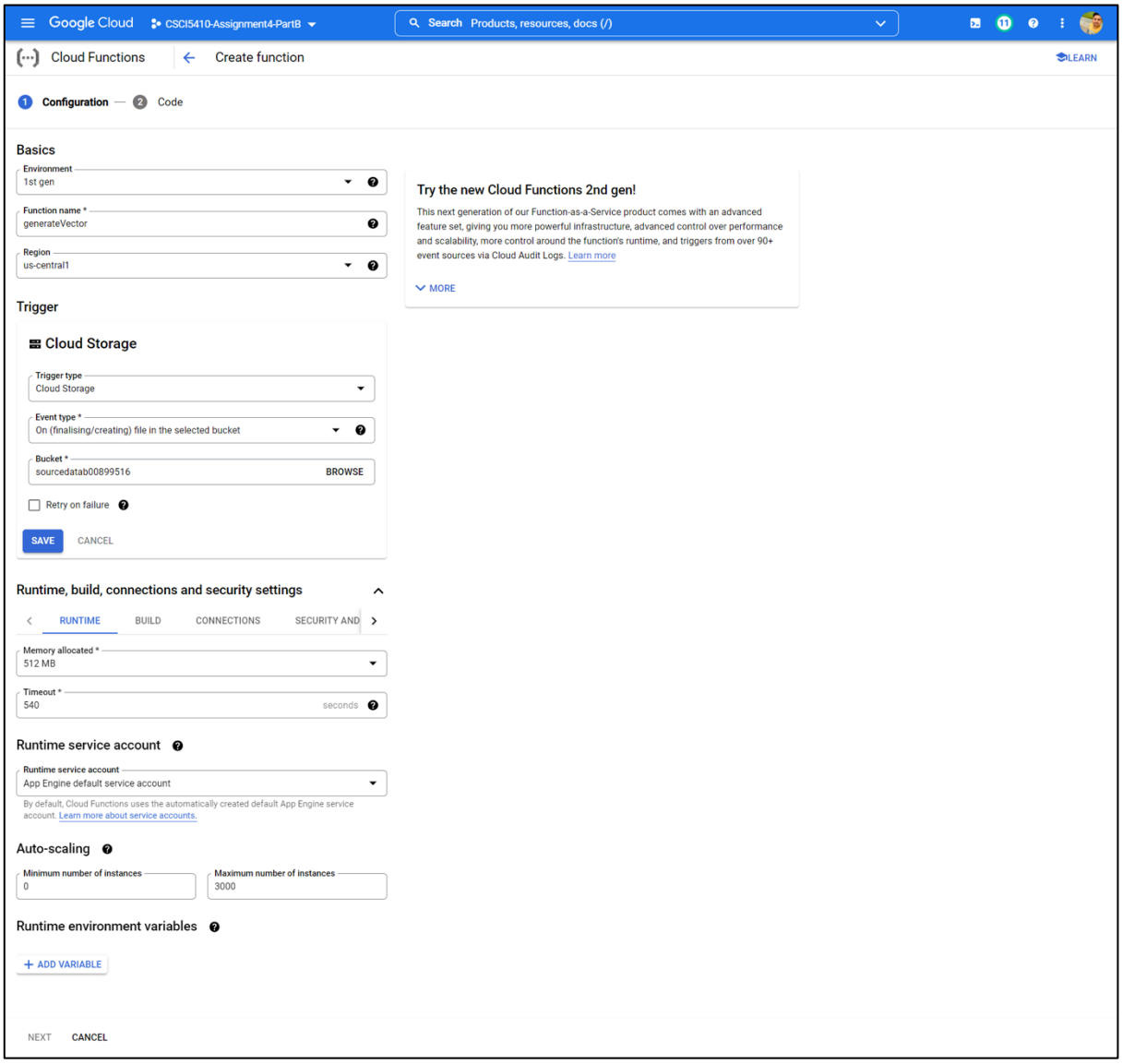
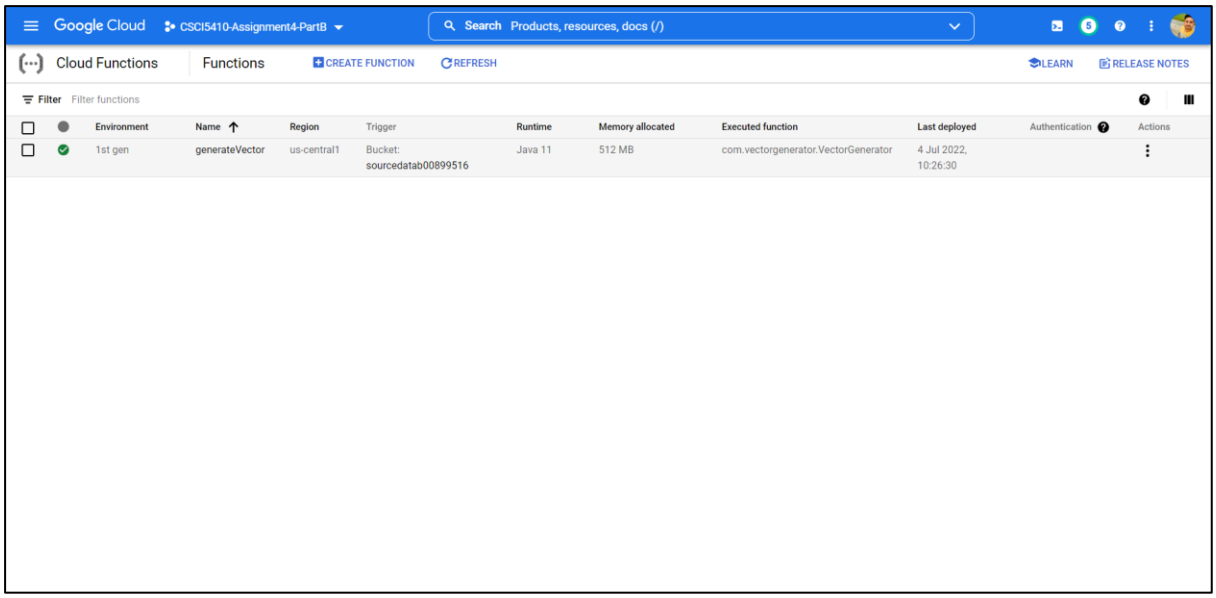


*Figure 23: Creation of **generateVector** cloud function*

Figure 24 is responsible for representing the successful deployment of cloud function with name "**generateVector**"



Figure

*24: Successfully deployed **generateVeactor** cloud function*

Figure 25 is responsible for showing the code of "generateVector". The code for the "generateVector" is directly written inside the "generateVector" cloud function.

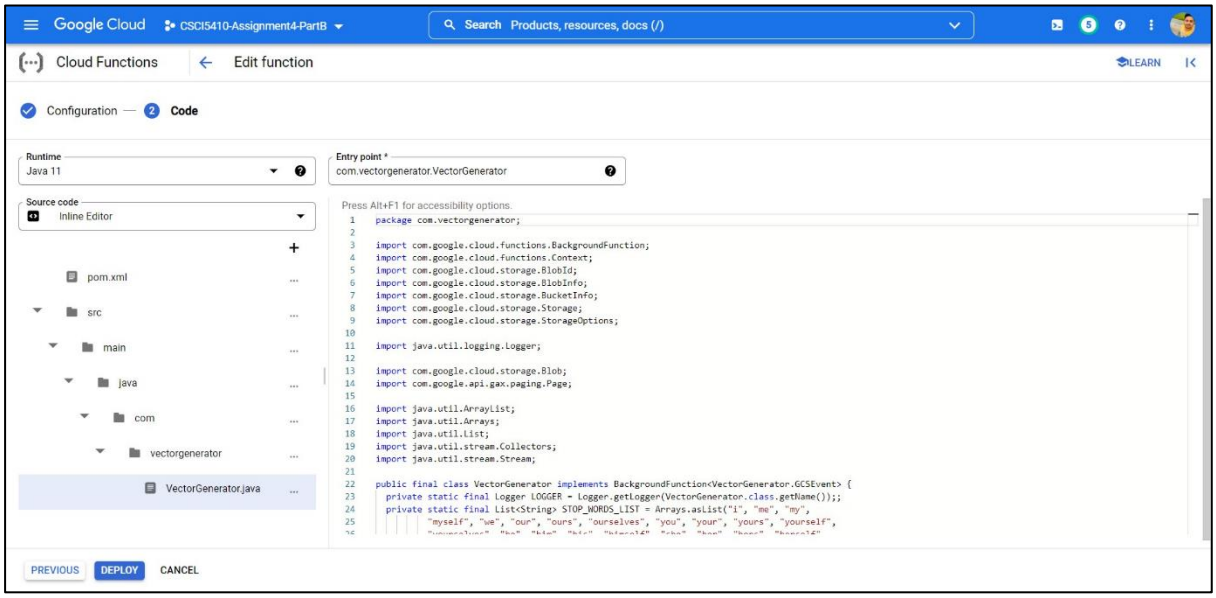


*Figure 25: Added Code for "**generateVector**" Cloud Function*

Figure 26 and 27 shows the logs which are generated when "**generateVector**" cloud function is triggered. This cloud function will be triggered when files from Train folder were uploaded to the "**sourcedatab00899516**". And when this function is triggered, it will also generate logs. This log provides lots of information like the files from the "**sourcedatab00899516**" bucket will be processed one by one. Then it will also show the creation of new bucket "**traindatab00899516**' where **trainVectors.csv** will be uploaded.

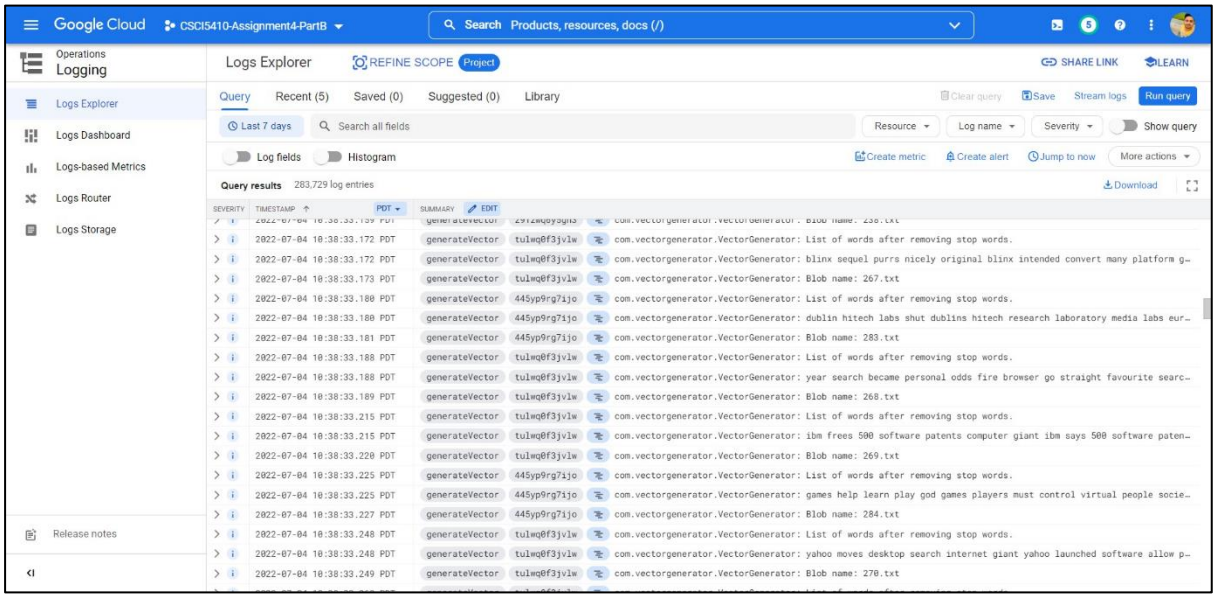


*Figure 26: Generated logs messages on successful execution of cloud function "**generatedVector**"*

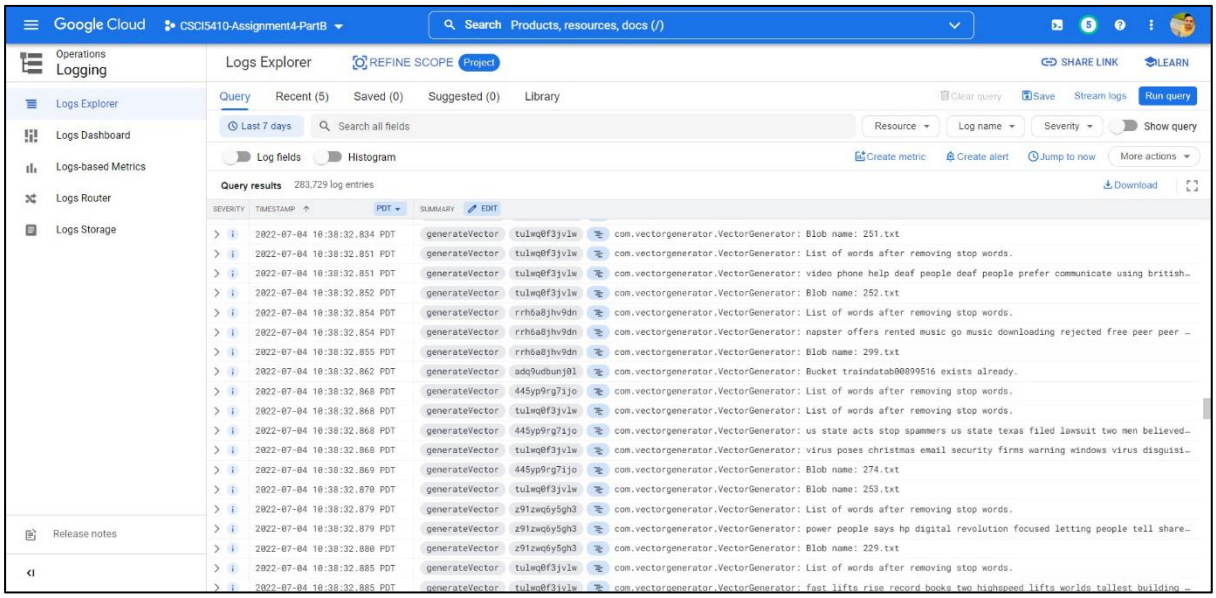


*Figure 27: Generated logs messages on successful execution of cloud function "**generatedVector**"*

Figure 28 represents the newly created bucket named as "**traindatab00899516**"

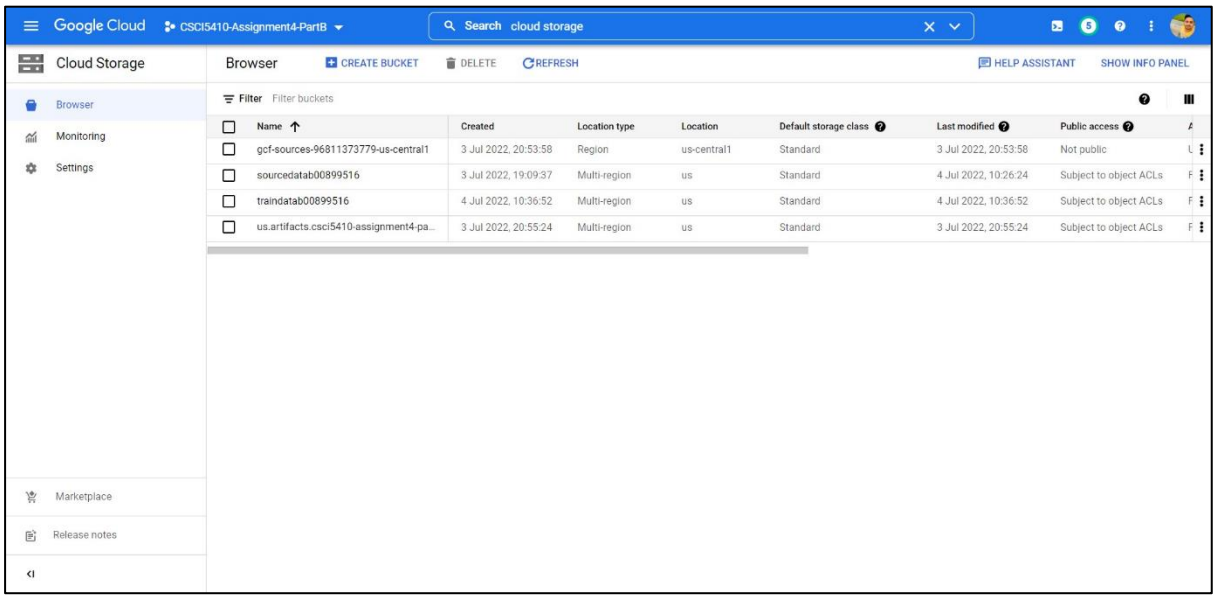


*Figure 28: Successfully created new bucket "**traindatab00899516**"*

C. This file "trainVector.csv" is saved in a new bucket TrainDataB00xxxxxx.

Figure 29 shows the **trainVector.csv** file created and uploaded to "**traindatab0899516**" bucket.

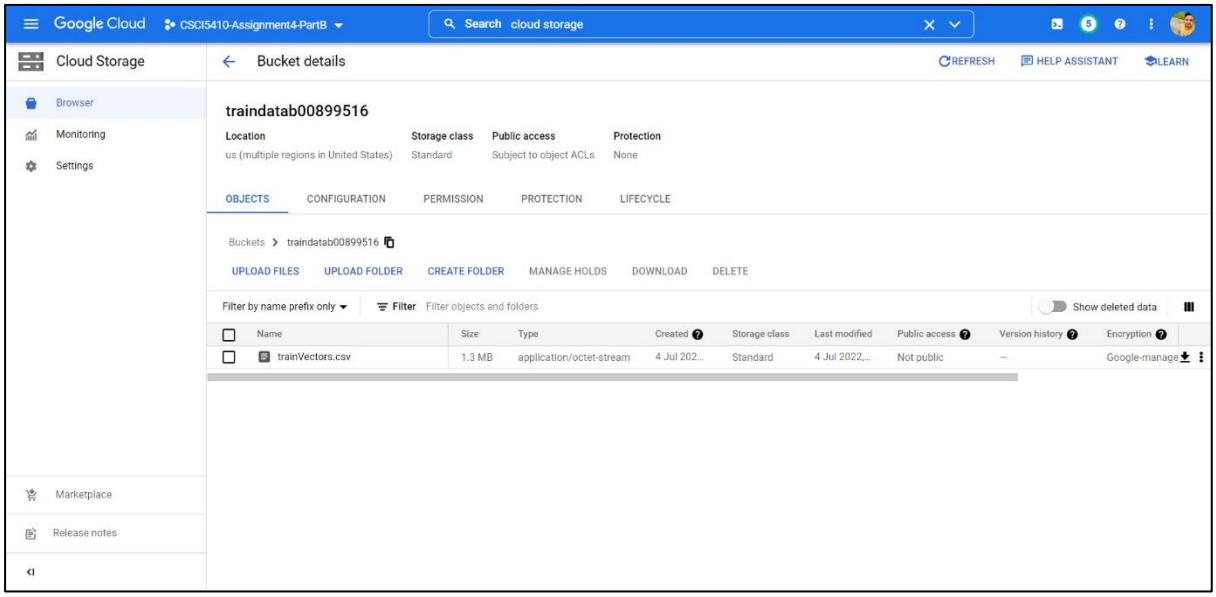
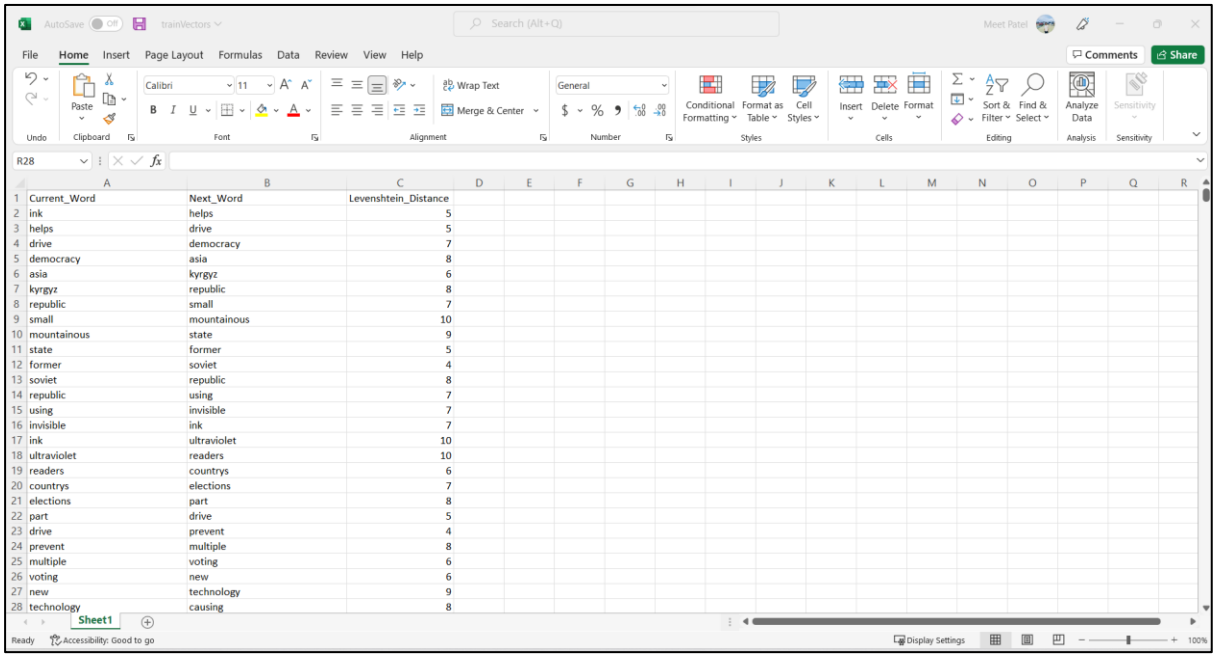


*Figure 29: Successfully created "trainVectors.csv" file in "**traindatab00899516**" bucket*

Figure 30 shows the content of **trainVector.csv** file, which is downloaded from "**traindatab00899516**" Bucket. This **trainVector.csv** file has three columns. The name of this columns is current word, the next word and Levenshtein distance.



	A	B	C
1	Current_Word	Next_Word	Levenshtein_Distance
2	ink	helps	5
3	helps	drive	5
4	drive	democracy	7
5	democracy	asia	8
6	asia	kyrgyz	6
7	kyrgyz	republic	8
8	republic	small	7
9	small	mountainous	10
10	mountainous	state	9
11	state	former	5
12	former	soviet	4
13	soviet	republic	8
14	republic	using	7
15	using	invisible	7
16	invisible	ink	7
17	ink	ultraviolet	10
18	ultraviolet	readers	10
19	readers	countrys	6
20	countrys	elections	7
21	elections	part	8
22	part	drive	5
23	drive	prevent	4
24	prevent	multiple	8
25	multiple	voting	6
26	voting	new	6
27	new	technology	9
28	technology	causing	8

*Figure 30: Content of **trainVector.csv***

D. GCP ML should get the training data for a clustering algorithm (KMenas) from the TrainDataB00xxxxxx bucket.

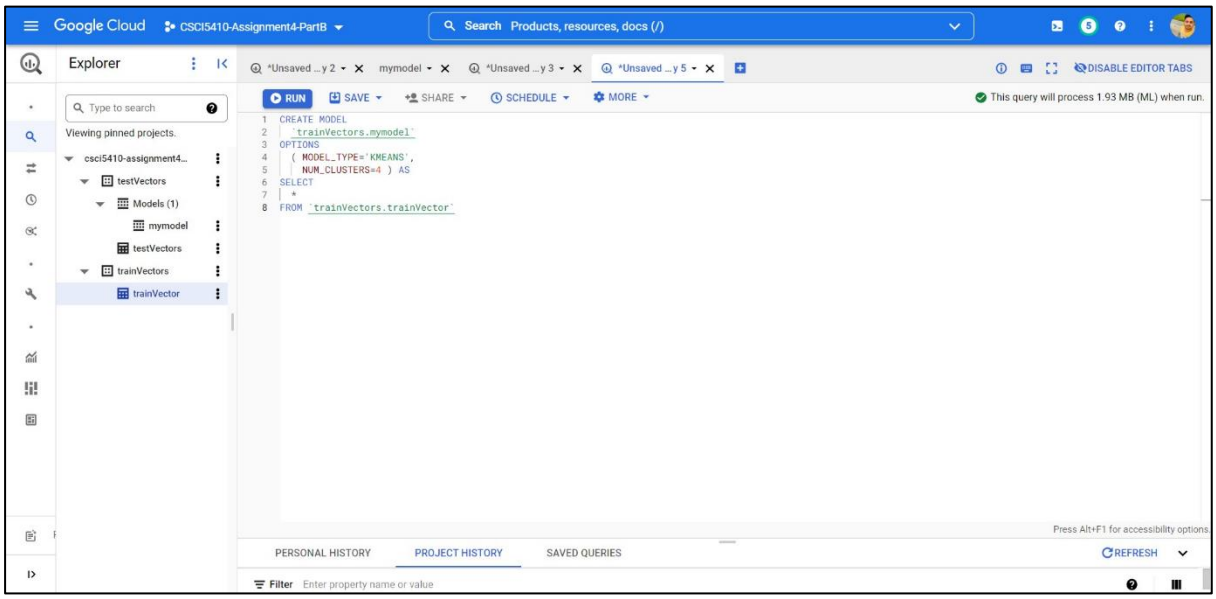


Figure 31: Training data generation for the KMeans algorithm from the TrainDataB00xxxxxx bucket using GCP ML

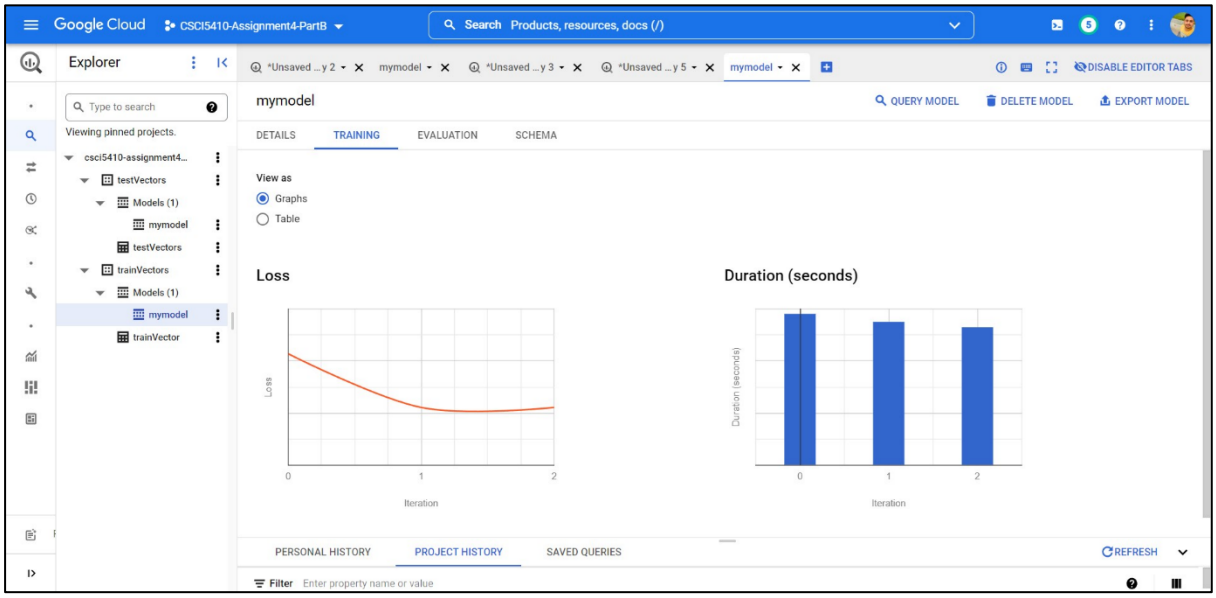


Figure 32: Training data generation for the KMeans algorithm from the TrainDataB00xxxxxx bucket using GCP ML

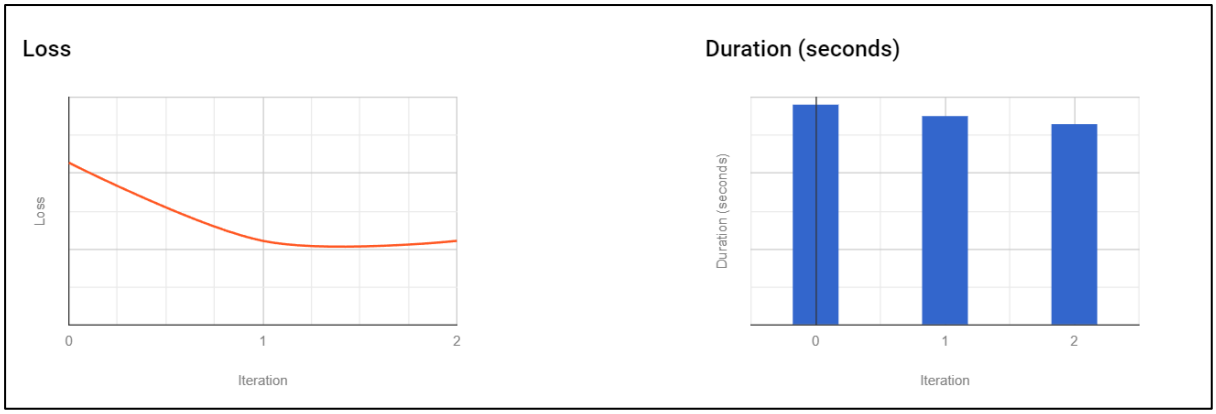


Figure 33: Training data generation for the KMeans algorithm from the TrainDataB00xxxxxx bucket using GCP ML

E. Once the training is done, like point (a), upload the test files given in the Test folder (300 to 401) to SourceDataB00xxxxxx.

Figure 34 to 36 is responsible for showing the files that are being uploaded from test folder to "**sourcedatab00899516**". This program does not create new bucket instead it will show the message like "**sourcedatab00899516**" bucket already exists and utilize that bucket to upload all files from **Dataset\Test** folder. Total 100 files will be uploaded (300.txt to 401.txt) to "**sourcedatab00899516**"

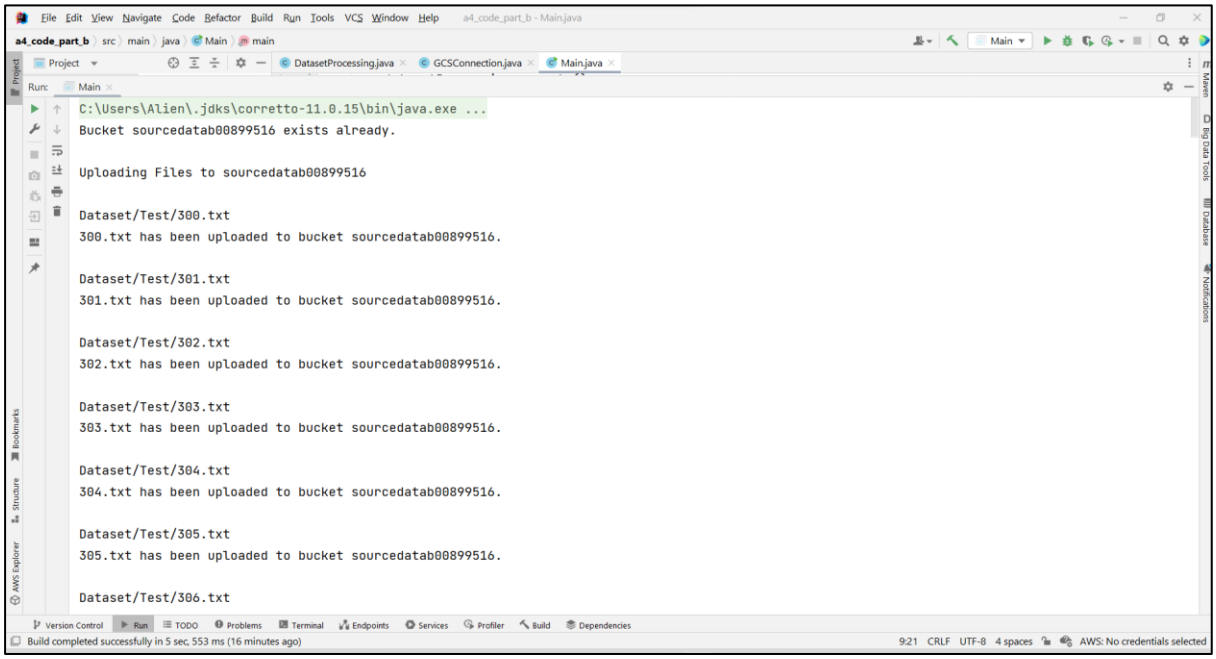


*Figure 34: Uploading **Dataset\Test** folder files to "**sourcedatab00899516**"*

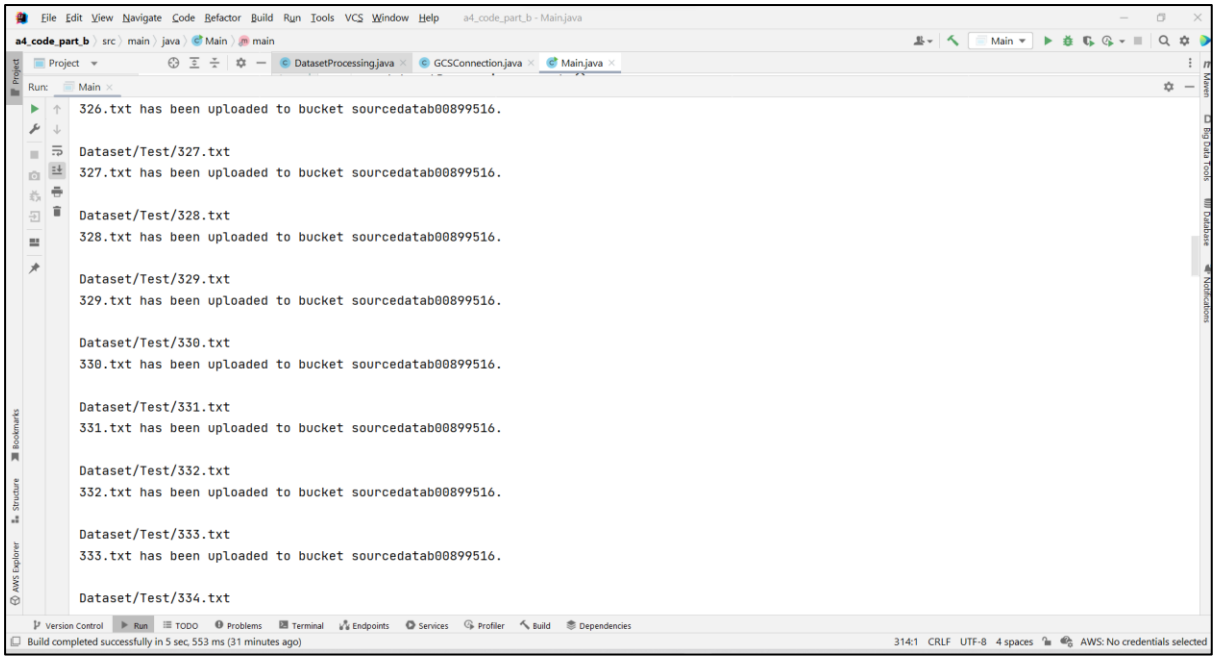


*Figure 35: Uploading **Dataset\Test** folder files to "**sourcedatab00899516**"*

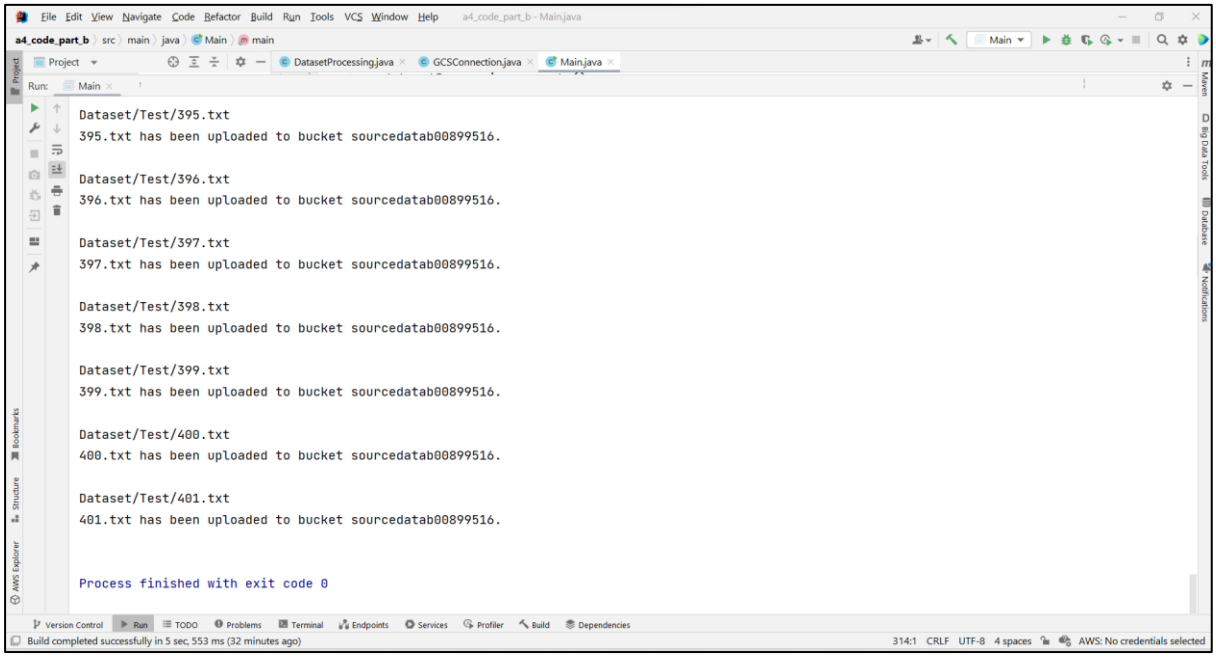


*Figure 36: Uploading **Dataset\Test** folder files to "**sourcedatab00899516**"*

Figure 34 to 36 shows the test files that are being successfully uploaded to **sourcedatab00899516** bucket on google storage. Total 102 files were uploaded (301.txt to 400.txt) on google storage.

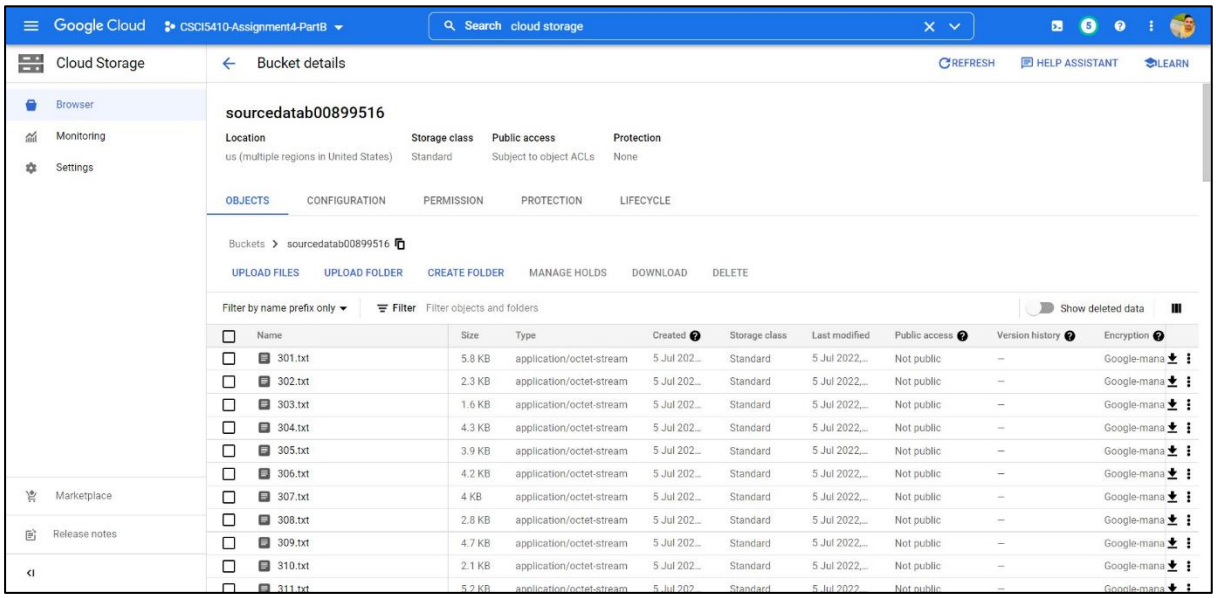


*Figure 37: Dataset/Text folder files uploaded to "**sourcedatab00899516**" bucket on google cloud storage*

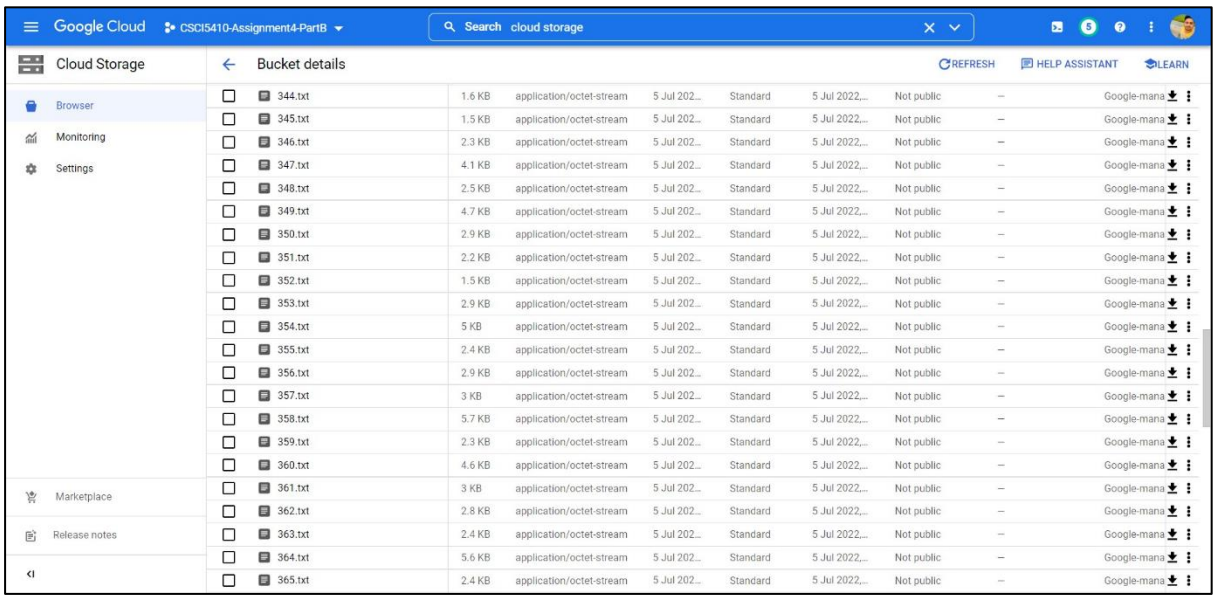


Figure 38: Dataset/Text folder files uploaded to "sourcedatab00899516" bucket on google cloud storage

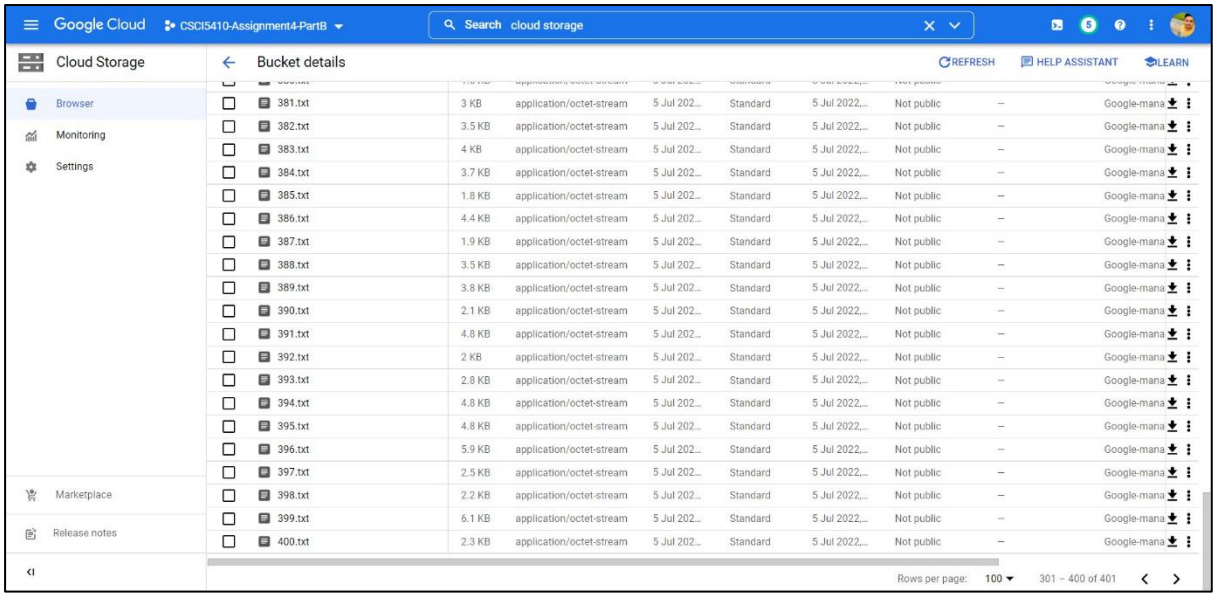


Figure 39: Dataset/Text folder files uploaded to "sourcedatab00899516" bucket on google cloud storage

F. The cloud function generateVector computes the distance vector same as point (b) and store it in "testVector.csv"

Figure 37 is responsible for displaying the newly created bucket with name **testdatab00899516**

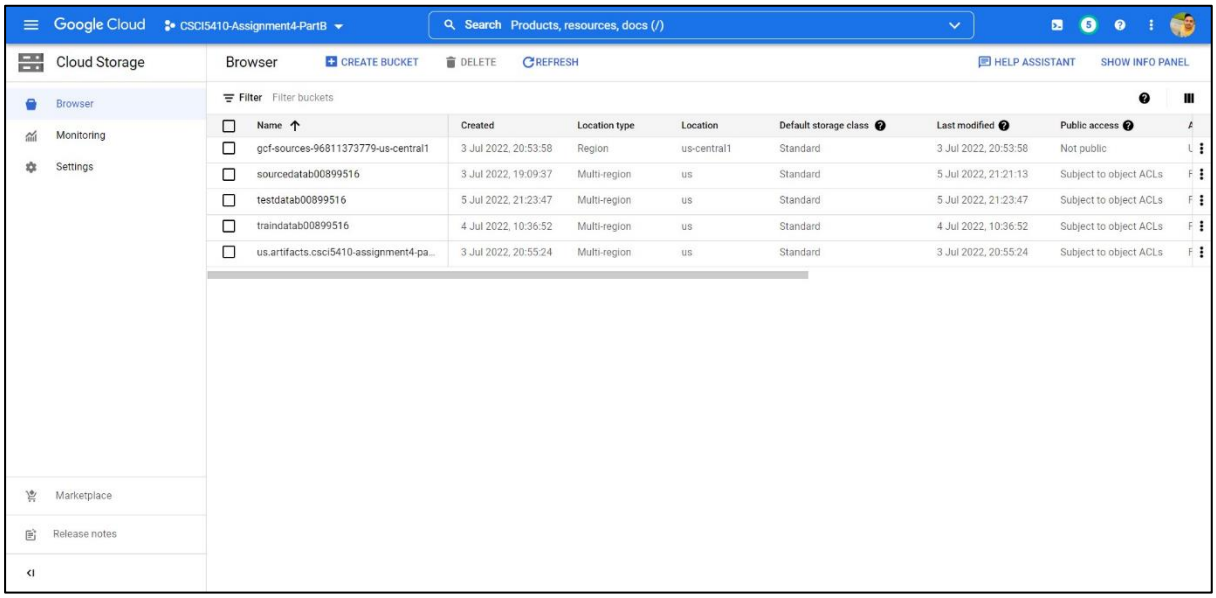


*Figure 40: Newly created bucket with bucket name **textdatab00899516***

Figure 38 is responsible for showing the testVector.csv which is uploaded to the newly created bucket **testdatab00899516**

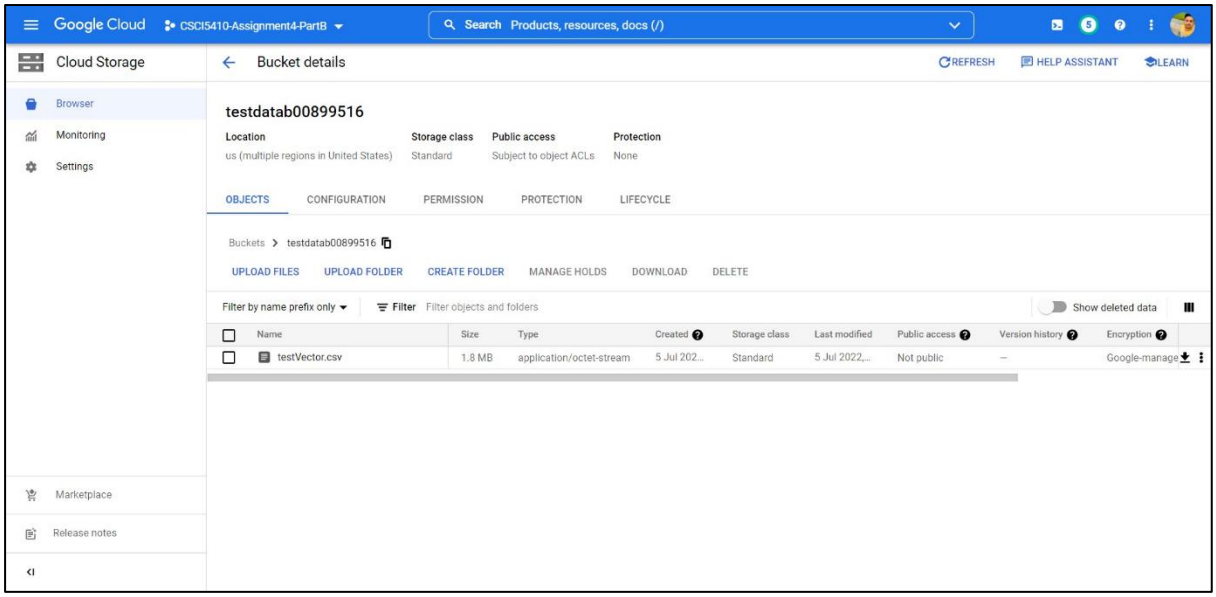
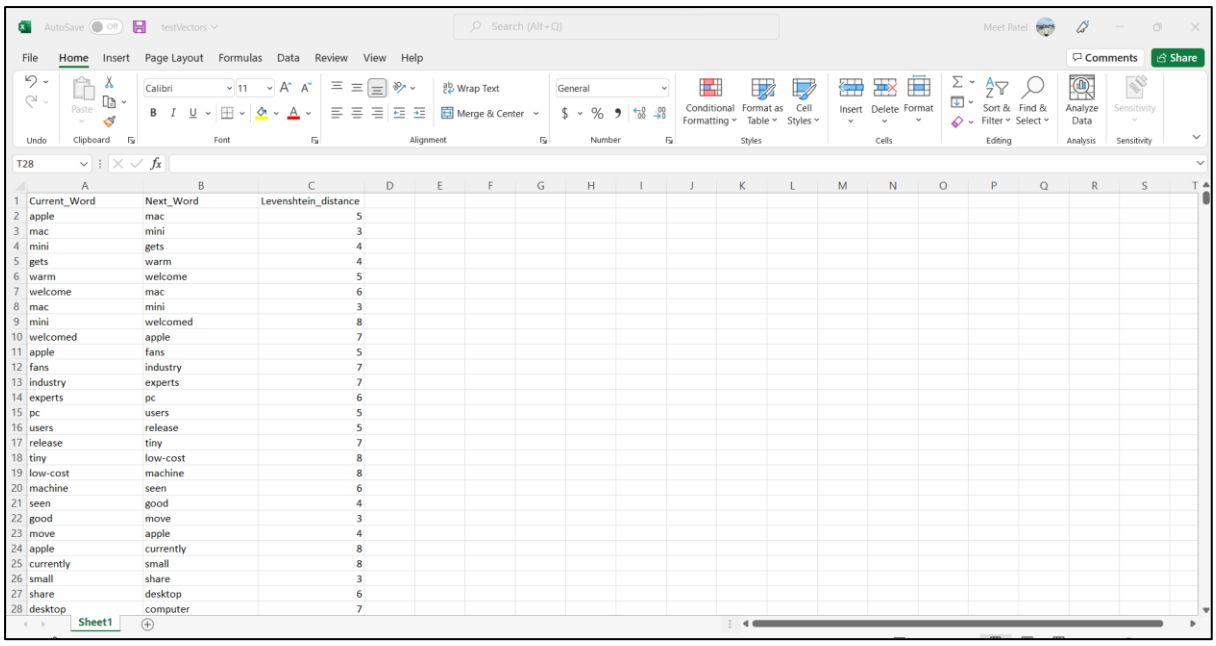


*Figure 41: textVector.csv file uploaded to newly created bucket **textdatab00899516** in cloud storage*

G. This file (testVector.csv) is saved in another new bucket, TestDataB00xxxxxx.

Figure 39 shows the content of **testVector.csv** file, which is downloaded from "**testdatab00899516**" Bucket. This **testVector.csv** file has three columns. The name of this columns is current word, the next word and Levenshtein distance.



	A	B	C
1	Current_Word	Next_Word	Levenshtein_distance
2	apple	mac	5
3	mac	mini	3
4	mini	gets	4
5	gets	warm	4
6	warm	welcome	5
7	welcome	mac	6
8	mac	mini	3
9	mini	welcomed	8
10	welcomed	apple	7
11	apple	fans	5
12	fans	industry	7
13	industry	experts	7
14	experts	pc	6
15	pc	users	5
16	users	release	5
17	release	tiny	7
18	tiny	low-cost	8
19	low-cost	machine	8
20	machine	seen	6
21	seen	good	4
22	good	move	3
23	move	apple	4
24	apple	currently	8
25	currently	small	8
26	small	share	3
27	share	desktop	6
28	desktop	computer	7

Figure 42: testVector.csv file content

H. GCP ML should get the test data for the KMeans algorithm from the TestDataB00xxxxxx bucket.

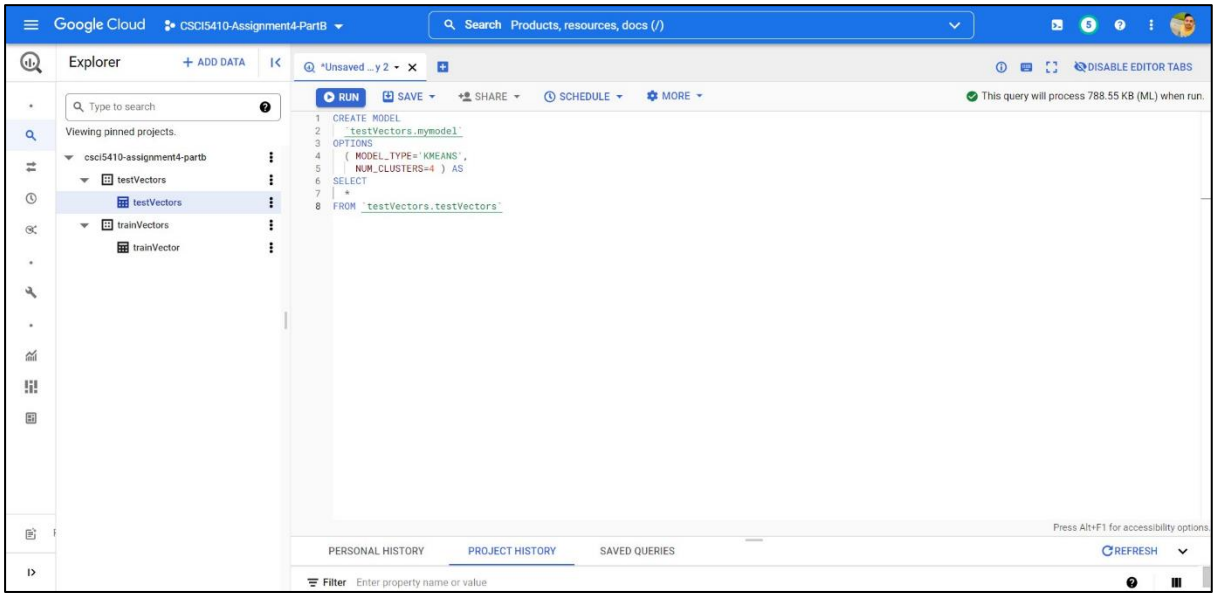


Figure 43: Test data generation for the KMeans algorithm from the TestDataB00xxxxxx bucket using GCP ML

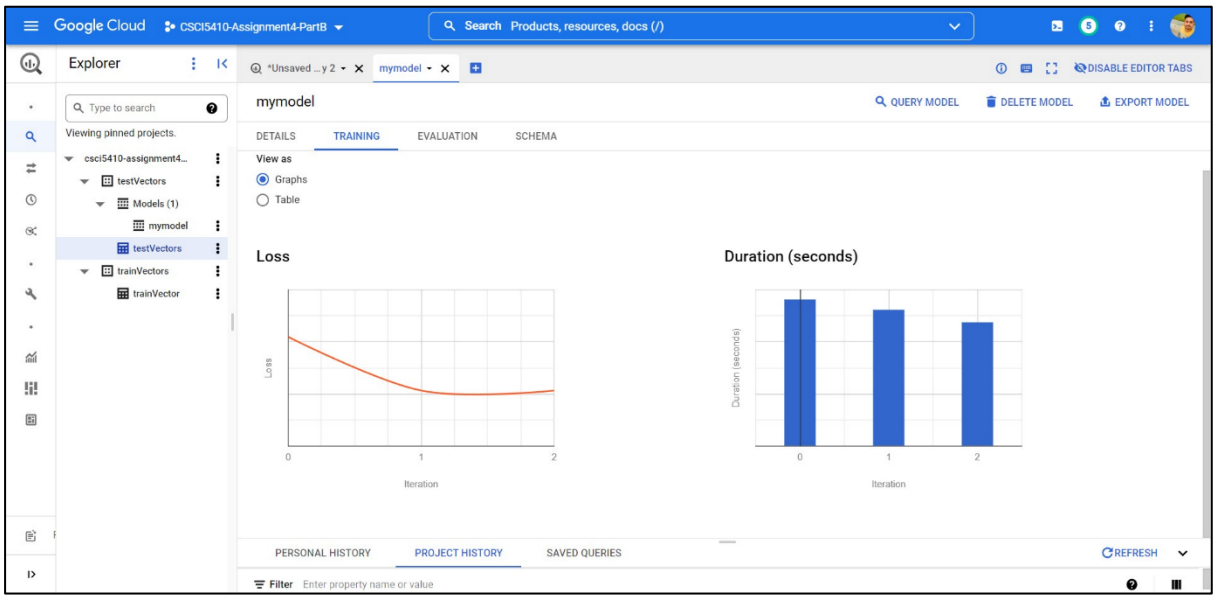


Figure 44: Test data generation for the KMeans algorithm from the TestDataB00xxxxxx bucket using GCP ML

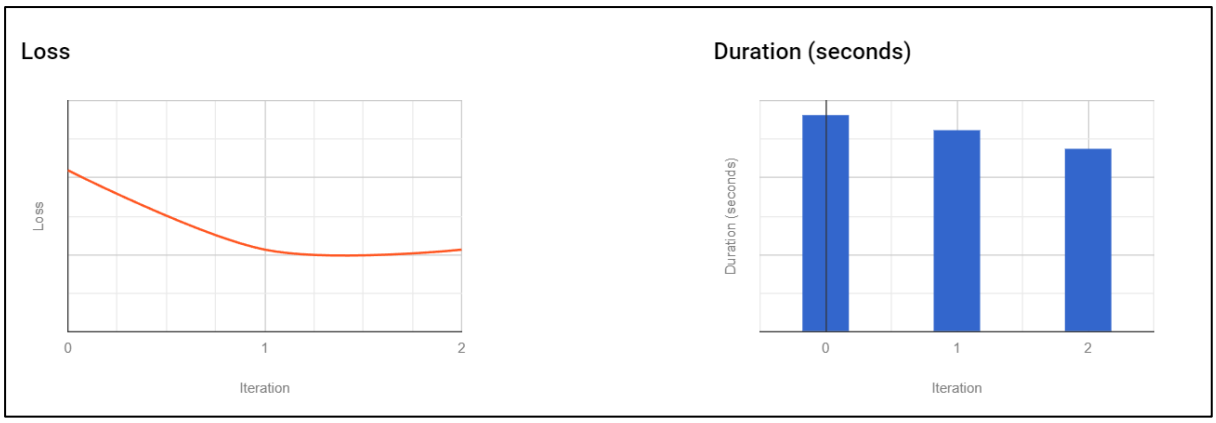


Figure 45: Test data for the KMeans algorithm from the TestDataB00xxxxxx bucket

I. Finally, write a code or configure a service to obtain information about clusters (e.g. centroids, cluster numbers, outliers etc. which are generated by GCP ML), and display the clusters.

Information about cluster for testvectors.csv generated by GCP ML

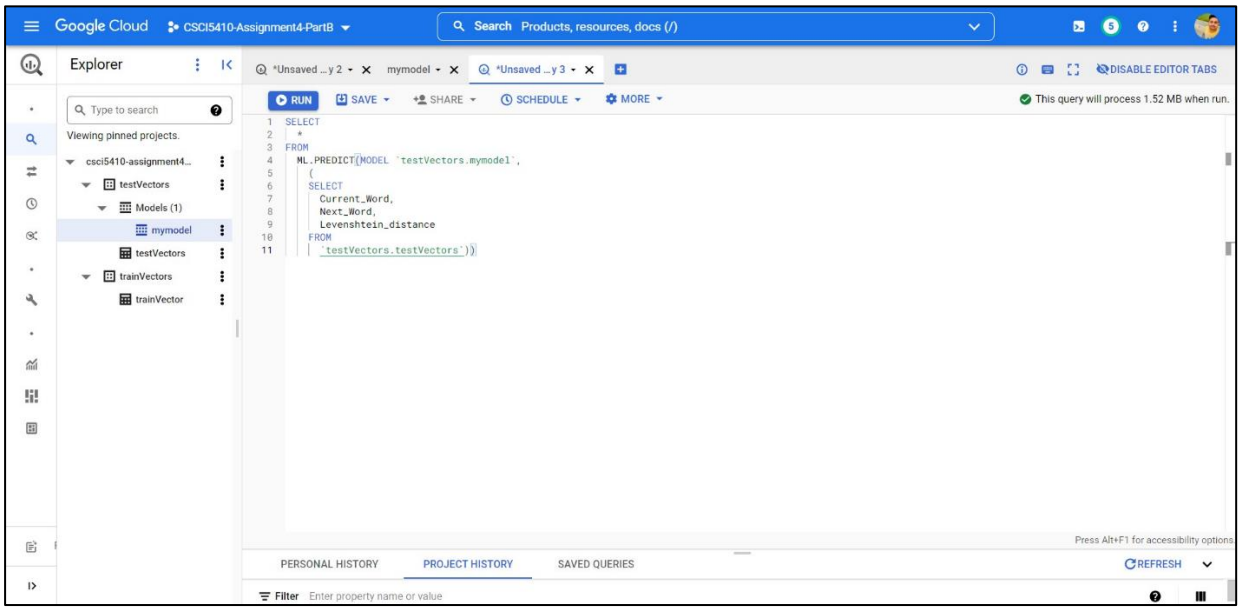


Figure 46: Information about clusters for testvector.csv generated by GCP ML

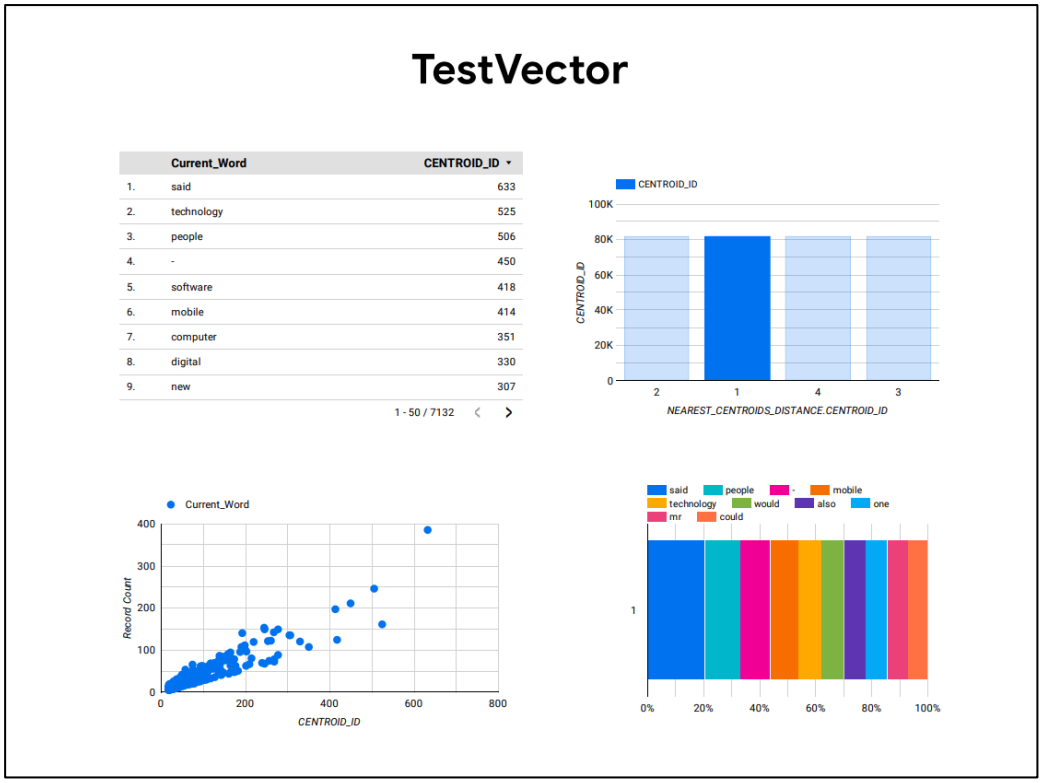


Figure 47: Information about clusters for testvector.csv generated by GCP ML

Information about cluster for trainvector.csv generated by GCP ML

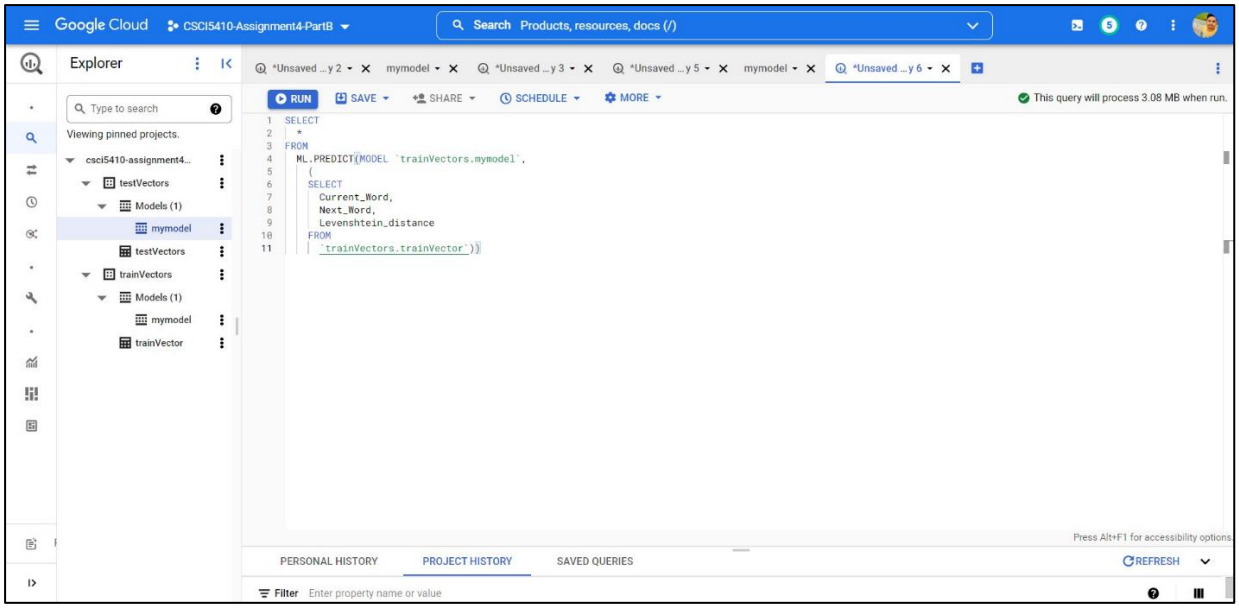


Figure 48: Information about clusters for trainvector.csv generated by GCP ML

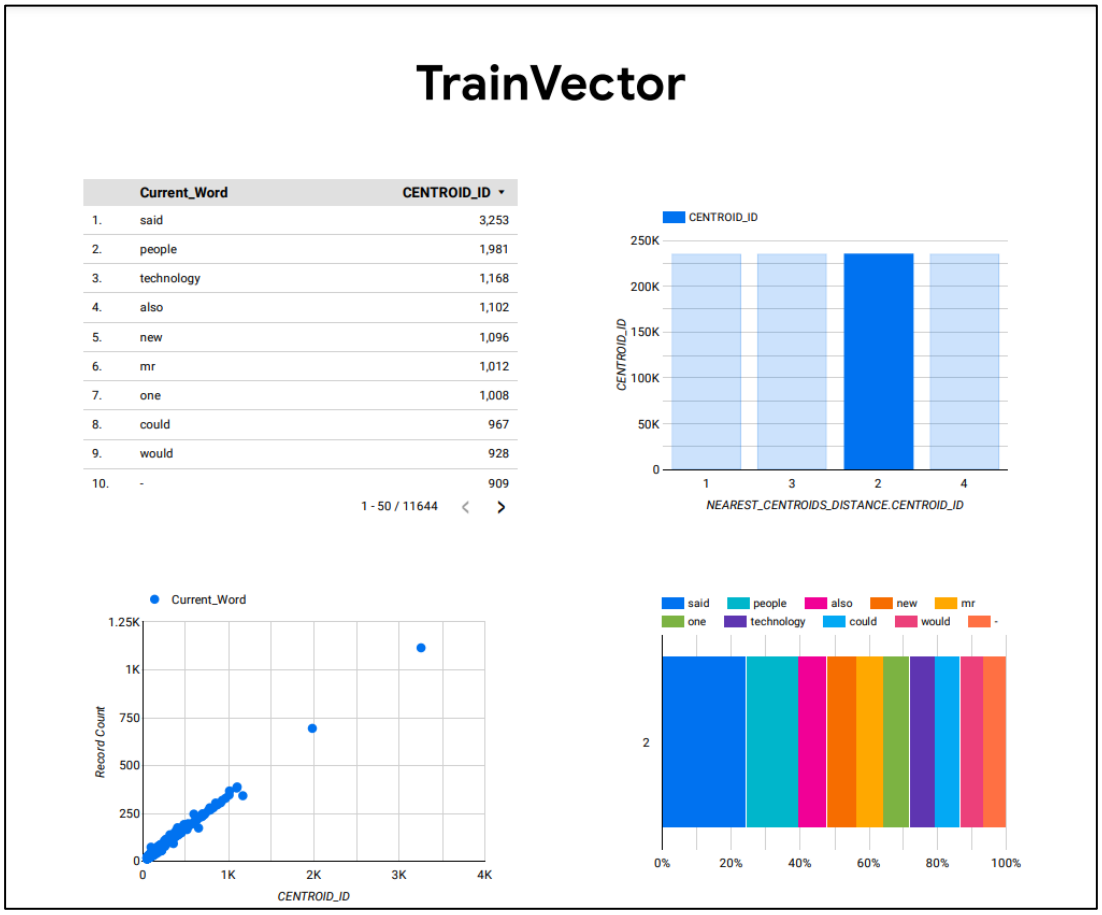


Figure 49: Information about clusters for trainvector.csv generated by GCP ML

Program/Scripts

GCSConnection.java

This java class is used to establish connection with the Google Cloud Storage

```
import com.google.auth.Credentials;
import com.google.auth.oauth2.GoogleCredentials;
import com.google.cloud.storage.Storage;
import com.google.cloud.storage.StorageOptions;
import java.io.FileInputStream;
import java.io.IOException;

public class GCSConnection {
    public Storage establishConnection() {
        try {
            Credentials credentials = GoogleCredentials.fromStream(new
FileInputStream("csci5410-assignment4-partb-bbfabf0ee9a5.json"));
            return
StorageOptions.newBuilder().setCredentials(credentials).setProjectId("csci5410-assignment4-
partb").build().getService();
        } catch (IOException e) {
            System.out.println("Error while creating GCP Connection: " + e.getMessage());
        }
        return null;
    }
}
```

DatasetProcessing.java

This class is used to create a bucket and then upload flies in that bucket on Google Cloud Storage

```
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import com.google.cloud.storage.BlobId;
import com.google.cloud.storage.BlobInfo;
import com.google.cloud.storage.BucketInfo;
import com.google.cloud.storage.Storage;
public class DatasetProcessing {
    GCSConnection gcsConnection = new GCSConnection();
    private static final String SOURCE_BUCKET = "sourcedatab00899516";

    public boolean checkIfBucketExists(){
        Storage storage = gcsConnection.establishConnection();
        if(storage.get(SOURCE_BUCKET) == null){
            return false;
        } else return storage.get(SOURCE_BUCKET).exists();
    }

    public Storage createBucket() {
        Storage storage = gcsConnection.establishConnection();
        if (storage != null) {
            if (!checkIfBucketExists()) {
                storage.create(BucketInfo.of(SOURCE_BUCKET));
                System.out.println("Bucket " + SOURCE_BUCKET + " created
successfully.");
```

```
            } else {
                System.out.println("Bucket " + SOURCE_BUCKET + " exists already.");
            }
        }
        return storage;
    }
    private void uploadFilesToGCSBucket(Storage storage) {
        try {
            File[] files = new File("Dataset/Test/").listFiles();
            if (files == null) {
                return;
            }
            System.out.println("\nUploading Files to "+SOURCE_BUCKET+"\n");
            for (File file : files) {
                System.out.println("Dataset/Test/" + file.getName());
                storage.create(BlobInfo.newBuilder(BlobId.of(SOURCE_BUCKET,
file.getName())).build(), Files.readAllBytes(Paths.get("Dataset/Test/" +
file.getName())));
                System.out.println(file.getName() + " has been uploaded to bucket " +
SOURCE_BUCKET + ".\n");
            }
        } catch (final IOException e) {
            System.out.println("Error while uploading file: " + e.getMessage());
        }
    }
    public void execute(){
        Storage storage = createBucket();
        uploadFilesToGCSBucket(storage);
    }
}
```

Main.java

```
public class Main {
    public static void main(String[] args) {
        DatasetProcessing datasetProcessing = new DatasetProcessing();
        datasetProcessing.execute();
    }
}
```

vectorGenerator.js

```
package vectorgenerator;
import com.google.api.gax.paging.Page;
import com.google.cloud.functions.BackgroundFunction;
import com.google.cloud.functions.Context;
import com.google.cloud.storage.*;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.stream.Collectors;
import java.util.stream.Stream;
public final class VectorGenerator implements
BackgroundFunction<VectorGenerator.GCSEvent> {
  private static final String TRAIN_VECTOR_CSV = "trainVectors.csv";
  private static final String TRAIN_DATA_BUCKET = "traindatab00899516";

  /**
   * Reference: https://www.baeldung.com/java-levenshtein-distance
   * Reference: https://www.baeldung.com/java-csv
   */
  @Override
  public void accept(GCSEvent event, Context context) {
    Storage storage = StorageOptions.getDefaultInstance().getService();
    Page<Blob> blobs = storage.get(event.bucket).list();
    ArrayList<String[]> dataLinesList = new ArrayList<>();
    StringBuilder listOfWords = new StringBuilder();

    for (Blob blob : blobs.iterateAll()) {
      String txtFileContent = regex(blob);

      ArrayList<String> wordsList = new
ArrayList<>(Arrays.asList(txtFileContent.toLowerCase().split(" ")));
      wordsList.removeAll(Arrays.asList("she", "her", "hers", "herself", "it", "its",
"itself", "they", "them", "their", "theirs", "themselves", "what", "which", "before",
"after", "above", "below", "to", "from", "up", "down", "in", "out", "on", "off", "over",
"under", "again", "further", "then", "once", "here", "there", "when", "where", "why",
"how", "all", "any", "both", "each", "few", "more", "who", "whom", "this", "that",
"these", "those", "am", "is", "are", "was", "were", "be", "been", "being", "have",
"has", "had", "having", "do", "does", "did", "doing", "a", "an", "the", "and", "but",
"if", "or", "because", "as", "until", "while", "of", "at", "by", "for", "with", "about",
"against", "between", "into", "through", "during", "most", "other", "some", "such",
"no", "nor", "not", "only", "own", "same", "so", "than", "too", "very", "s", "t", can",
"will", "just", "don", "should", "now"));
      for (String word : wordsList) {
        listOfWords.append(word).append(" ");
      }
      /**
       * Reference: https://www.baeldung.com/java-csv
       */
      for (int i = 0; i < wordsList.size() - 1; i++) {
        String currentWord = wordsList.get(i);
        String nextWord = wordsList.get(i + 1);
        int[][] dp = new int[currentWord.length() + 1][nextWord.length() + 1];
        for (int i1 = 0; i1 <= currentWord.length(); i1++) {
          for (int j = 0; j <= nextWord.length(); j++) {
            if (i1 == 0) {
              dp[i1][j] = j;
            } else if (j == 0) {
              dp[i1][j] = i1;
            } else {
              /**
               * Reference: https://www.baeldung.com/java-levenshtein-distance
               */
              dp[i1][j] = Arrays.stream(new int[]{dp[i1 - 1][j - 1] +
                      (currentWord.charAt(i1 - 1) == nextWord.charAt(j - 1) ? 0 : 1),
dp[i1 - 1][j] + 1, dp[i1][j - 1] + 1}).min().orElse(Integer.MAX_VALUE);
            }
```

```
          int distance = dp[currentWord.length()][nextWord.length()];
         dataLinesList.add(new String[]{currentWord, nextWord,
String.valueOf(distance)});
        }
      }
      StringBuilder sb = new StringBuilder();
      for (String[] strings : dataLinesList) {
        sb.append(Stream.of(strings)
                .map(data -> {
                  /**
                   * Reference: https://www.baeldung.com/java-csv
                   */
                  String escapedData = data.replaceAll("\\R", " ");
                  if (data.contains(",") || data.contains("\"") || data.contains("'")) {
                    data = data.replace("\"", "\"\"");
                    escapedData = "\"" + data + "\"";
                  }
                  return escapedData;
                })
                .collect(Collectors.joining(","))).append("\n");
      }
      if (isbucketNull(storage)) {
        storage.create(BucketInfo.of(TRAIN_DATA_BUCKET));
      } else if (isBucketExists(storage)) {
        storage.create(BucketInfo.of(TRAIN_DATA_BUCKET));
      } else {
      }
      storage.create(BlobInfo.newBuilder(BlobId.of(TRAIN_DATA_BUCKET,
TRAIN_VECTOR_CSV)).build(), sb.toString().getBytes());
    }

    /**
     * Reference: https://www.baeldung.com/java-csv
     */
    private String regex(Blob blob) {
      return new String(blob.getContent())
              .replaceAll("[,.\\-_'\"(){}?!@#$%^&*\\[\\]]+", "")
              .replaceAll("[\\s]+", " ");
    }

    private boolean isBucketExists(Storage storage) {
      return !storage.get(TRAIN_DATA_BUCKET).exists();
    }

    private boolean isbucketNull(Storage storage) {
      return storage.get(TRAIN_DATA_BUCKET) == null;
    }

    public static class GCSEvent {
      String bucket;
      String name;
      String metaGeneration;

      public GCSEvent(final String bucket,
                      final String name,
                      final String metaGeneration) {
        this.bucket = bucket;
        this.name = name;
        this.metaGeneration = metaGeneration;
      }
    }

}
```

References

[1] Google, "Cloud Functions," Google, [Online]. Available: https://cloud.google.com/functions. [Accessed 09 July 2022].

[2] Baeldung, "How to calculate Levenshtein Distance in Java?" Baeldung, [Online]. Available: https://www.baeldung.com/java-levenshtein-distance [Accessed 09 July 2022].

[3] Wikipedia, "Levenshtein Distance," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Levenshtein_distance [Accessed 09 July 2022].

[4] Google, "Cloud Storage," Google, [Online]. Available: https://cloud.google.com/storage. [Accessed 09 July 2022].

[5] Google, "Vertex AI," Google, [Online]. Available: https://cloud.google.com/vertex_ai/docs/reference/rest [Accessed 09 July 2022].

[6] Google, "Google Cloud," Google, [Online]. Available: https://cloud.google.com/gcp. [Accessed 09 July 2022].

[7] Google, "Operation Suits," Google, [Online]. Available: https://cloud.google.com/logging. [Accessed 09 July 2022].

[8] javaTpoint, "K-Means Clustering Algorithm," javaTpoint, [Online]. Available: https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning [Accessed 09 July 2022].

[9] Wikipedia, "Elbow method (clustering)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Elbow_method_(clustering) [Accessed 09 July 2022].

[10] Baeldung, "How to Write to a CSV File in Java," Baeldung, [Online]. Available: https://www.baeldung.com/java-csv [Accessed 09 July 2022].

[11] Wikipedia, "k-means clustering," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Levenshtein_distance [Accessed 09 July 2022].

[12] GitHub, "NLTK's list of english stopwords," GitHub, [Online]. Available: https://gist.github.com/sebleier/554280 [Accessed 09 July 2022]