

CSCI 5410: Assignment 4

Part B: Build an event-driven serverless application using GCP ML.

GitLab Link: https://git.cs.dal.ca/drshah/dhrumilrakeshshah_csci5410.git

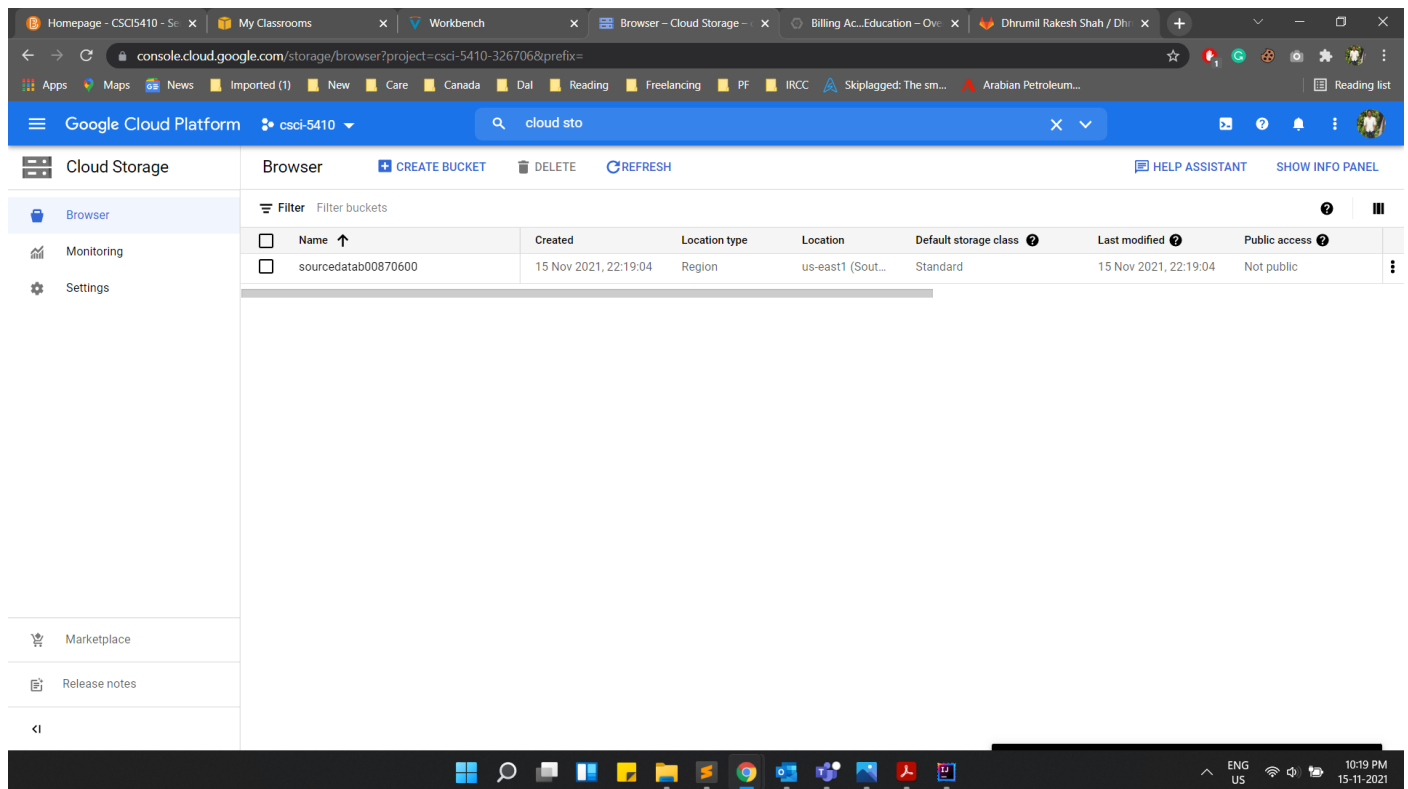


Figure 1: sourcedatab00870600 Bucket

The screenshot shows the Google Cloud Platform console interface. The left sidebar contains navigation options: Cloud Storage, Marketplace, and Release notes. The main content area displays the 'Bucket details' for 'sourcedatab00870600'. The bucket is located in 'us-east1 (South Carolina)', has a 'Standard' storage class, 'Not public' access, and 'None' protection. Below the details, there are tabs for OBJECTS, CONFIGURATION, PERMISSIONS, PROTECTION, and LIFECYCLE. The 'OBJECTS' tab is selected, showing a list of objects. The list is empty, with a message 'No rows to display' at the bottom. The top of the console shows the project name 'csci-5410' and a search bar. The bottom of the console shows the Windows taskbar with various application icons and the system clock indicating 10:19 PM on 15-11-2021.

Figure 2: sourcedatab00870600 Empty Bucket

The screenshot shows the Google Cloud Platform console interface, specifically the 'Bucket details' for 'sourcedatab00870600'. The 'CONFIGURATION' tab is selected. The configuration details are as follows:

Property	Value
Created	15 November 2021 at 22:19:04 GMT-4
Updated	15 November 2021 at 22:19:04 GMT-4
Location type	Region
Location	us-east1 (South Carolina)
Replication	—
Default storage class	Standard
Requester pays	OFF
Labels	None
Cloud Console URL	https://console.cloud.google.com/storage/browser/sourcedatab00870600
gsutil URI	gs://sourcedatab00870600
Access control	Uniform
Public access prevention	Not enabled by org policy or bucket setting
Public access status	Not public

The bottom of the console shows the Windows taskbar with various application icons and the system clock indicating 10:21 PM on 15-11-2021.

Figure 3: sourcedatab00870600 Bucket Config

Cloud Storage | **Bucket details** | **sourcedatab00870600**

Location: us-east1 (South Carolina) | Storage class: Standard | Public access: Not public | Protection: None

OBJECTS | CONFIGURATION | PERMISSIONS | PROTECTION | LIFECYCLE

Buckets > sourcedatab00870600

UPLOAD FILES | UPLOAD FOLDER | CREATE FOLDER | MANAGE HOLDS | DOWNLOAD | DELETE

Filter by name prefix only | Filter | Filter objects and folders | Show deleted data

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Actions
001.txt	3.9 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share
002.txt	2.2 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share
003.txt	1.3 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share
004.txt	2.5 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share
005.txt	4.8 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share
006.txt	3.8 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share
007.txt	1.7 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share
008.txt	1.7 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share
009.txt	7 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share
010.txt	2.0 KB	text/plain	16 Nov 20...	Standard	16 Nov 20...	Not public	—	Google-managed key	Download, Delete, Share

Figure 4: sourcedatab00870600 Bucket containing Train folder files

```

public class Main {
    public static void main(String[] args) throws IOException {
        // The ID of your GCP project
        String projectId = "csci-5410-326706";

        // The ID of your GCS bucket
        String bucketName = "sourcedatab00870600";

        for (int i = 1; i <= 299; i++) {
            // The ID of your GCS object
            String objectName = "";

            // The path to your file to upload
            String filePath = "";

            if (i < 10) {
                objectName = "00" + i;
                filePath = "Train/00" + i + ".txt";
            } else if (i > 9 && i < 100) {
                objectName = "0" + i;
                filePath = "Train/0" + i + ".txt";
            } else {
                objectName = "" + i;
            }

            Storage storage = StorageOptions.newBuilder().setProjectId(projectId).build().getService();
            BlobId blobId = BlobId.of(bucketName, objectName);
            BlobInfo blobInfo = BlobInfo.newBuilder(blobId).build();
            storage.create(blobInfo, Files.readAllBytes(Paths.get(filePath)));
        }
    }
}

```

Figure 5: sourcedatab00870600 Bucket File Upload Code

The screenshot shows the Google Cloud Platform console interface. The left sidebar contains navigation links: Cloud Storage, Browser, Monitoring, and Settings. The main content area is titled 'Bucket details' for the bucket 'traindatab00870600'. The bucket's location is 'us-east1 (South Carolina)', storage class is 'Standard', public access is 'Not public', and protection is 'None'. Below this, there are tabs for OBJECTS, CONFIGURATION, PERMISSIONS, PROTECTION, and LIFECYCLE. The 'OBJECTS' tab is active, showing a list of objects. The list is empty, with the message 'No rows to display' at the bottom. The top of the console shows the Google Cloud Platform logo, a search bar, and various status icons. The bottom of the console shows a Windows taskbar with various application icons and the system clock showing 11:13 PM on 17-11-2021.

Cloud Storage

Bucket details

traindatab00870600

Location: us-east1 (South Carolina) Storage class: Standard Public access: Not public Protection: None

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE

Buckets > traindatab00870600

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE

Filter by name prefix only Filter objects and folders Show deleted data

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiry date	Holds
No rows to display										

Figure 6: traindatab00870600 Empty Bucket

The screenshot shows the Google Cloud Platform console interface. The left sidebar contains navigation links: Cloud Storage, Browser, Monitoring, and Settings. The main content area is titled 'Bucket details' for the bucket 'traindatab00870600'. The bucket's location is 'us-east1 (South Carolina)', storage class is 'Standard', public access is 'Not public', and protection is 'None'. Below this, there are tabs for OBJECTS, CONFIGURATION, PERMISSIONS, PROTECTION, and LIFECYCLE. The 'OBJECTS' tab is active, showing a list of objects. The list contains one object, 'trainvector.csv', with a size of 45 B, type 'application/vnd.ms-excel', created on 17 Nov 2021, storage class 'Standard', last modified on 17 Nov 2021, public access 'Not public', version history '--', encryption 'Google-managed', and retention expiry date. The top of the console shows the Google Cloud Platform logo, a search bar, and various status icons. The bottom of the console shows a Windows taskbar with various application icons and the system clock showing 11:15 PM on 17-11-2021.

Cloud Storage

Bucket details

traindatab00870600

Location: us-east1 (South Carolina) Storage class: Standard Public access: Not public Protection: None

OBJECTS CONFIGURATION PERMISSIONS PROTECTION LIFECYCLE

Buckets > traindatab00870600

UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE

Filter by name prefix only Filter objects and folders Show deleted data

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption	Retention expiry date	Holds
trainvector.csv	45 B	application/vnd.ms-excel	17 Nov 2021	Standard	17 Nov 2021	Not public	--	Google-managed		

Figure 7: traindatab00870600 Bucket containing trainVector File

```

trainvector.csv
1 Current_Word Next_Word Levenshtein Distance
2
[nodemon] app crashed - waiting for file changes before starting...
[nodemon] restarting due to changes...
[nodemon] starting `node generateVector.js generateVector.js`

word 1= Ink word2 = Distance =
word 1= Ink word2 = helps Distance =5
word 1= helps word2 = drive Distance =5
word 1= drive word2 = democracy Distance =7
word 1= democracy word2 = in Distance =9
word 1= in word2 = Asia Distance =3
word 1= Asia word2 = The Distance =4
word 1= The word2 = Kyrgyz Distance =6
word 1= Kyrgyz word2 = Republic Distance =8
word 1= Republic word2 = a Distance =8
word 1= a word2 = small Distance =4
word 1= small word2 = mountainous Distance =10
word 1= mountainous word2 = state Distance =9
word 1= state word2 = of Distance =5
word 1= of word2 = the Distance =3
word 1= the word2 = former Distance =5
word 1= former word2 = Soviet Distance =4
word 1= Soviet word2 = republic Distance =8
word 1= republic word2 = is Distance =7
word 1= is word2 = using Distance =4
word 1= using word2 = invisible Distance =7
word 1= invisible word2 = ink Distance =7
word 1= ink word2 = and Distance =2
word 1= and word2 = ultraviolet Distance =10
word 1= ultraviolet word2 = readers Distance =10
word 1= readers word2 = in Distance =7
word 1= in word2 = the Distance =3
word 1= the word2 = countrys Distance =7
word 1= countrys word2 = elections Distance =7
word 1= elections word2 = as Distance =8
word 1= as word2 = part Distance =3
word 1= part word2 = of Distance =4
word 1= of word2 = a Distance =2
word 1= a word2 = drive Distance =5

```

Figure 8: trainVector File Contents

```

31 console.log("ObjectNot found")
32 } else {
33   let filedata = metadata.Body.toString('utf-8');
34   filedata = filedata.split("\n")
35   console.log()
36   word1 = "";
37   word2 = "";
38   for (let i = 0; i < filedata.length; i++) {
39     eachwords = filedata[i].split(' ');
40     for (let k = 0; k < eachwords.length; k++) {
41       if (eachwords[k] === '') {
42         continue;
43       }
44       distance = "";
45       eachwords[k] = eachwords[k].replace(/[^a-zA-Z ]/g, "");
46       if (i == 0 && k == 0) {
47         word1 = eachwords[k];
48       } else if (i == 0 && k == 1) {
49         word2 = eachwords[k];
50         distance = levenshteinDistance(word1, word2);
51       } else {
52         word1 = word2;
53       }
54     }
55   }
56 }

```

Figure 9: generateVector Cloud Function Code

The screenshot shows the Google Cloud Platform console with the Cloud Storage section selected. The bucket 'sourcedatab00870600' is displayed, located in us-east1 (South Carolina) with Standard storage class, Not public access, and No protection. The bucket contains 11 test files (300.txt to 308.txt) all created on Nov 20, 2021, with sizes ranging from 1.6 KB to 5.8 KB. The files are all text/plain type and use Google-managed encryption. The console interface includes a sidebar with navigation options like Browser, Monitoring, and Settings, and a top navigation bar with search and user information.

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
300.txt	4.6 KB	text/plain	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed key
301.txt	5.8 KB	text/plain	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed key
302.txt	2.3 KB	text/plain	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed key
303.txt	1.6 KB	text/plain	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed key
304.txt	4.3 KB	text/plain	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed key
305.txt	3.9 KB	text/plain	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed key
306.txt	4.2 KB	text/plain	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed key
307.txt	4 KB	text/plain	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed key
308.txt	2.8 KB	text/plain	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed key

Figure 10: sourcedatab00870600 Bucket containing Test Files

The screenshot shows the Google Cloud Platform console with the Cloud Storage section selected. The bucket 'testdatab00870600' is displayed, located in us-east1 (South Carolina) with Standard storage class, Not public access, and No protection. The bucket contains a single file 'testVector.csv' created on Nov 20, 2021, with a size of 45 B, application/vnd.ms-excel type, and Google-managed encryption. The console interface is similar to Figure 10, showing the bucket details and the file list.

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history	Encryption
testVector.csv	45 B	application/vnd.ms-excel	17 Nov 20...	Standard	17 Nov 20...	Not public	—	Google-managed

Figure 11: testdatab00870600 Bucket containing testVector File

generateVector Code:

```

/**
 * Responds to any HTTP request.
 *
 * @param {!express:Request} req HTTP request context.
 * @param {!express:Response} res HTTP response context.
 */
exports.helloWorld = (req, res) => {
  let message = req.query.message || req.body.message || 'Hello World!';
  res.status(200).send(message);
  if (err && err.code === 'NotFound') {
    console.log("ObjectNot found")
  } else {
    let filedata = metadata.Body.toString('utf-8');

    filedata = filedata.split("\n")
    console.log()
    word1 = "";
    word2 = "";

    for (let i = 0; i < filedata.length; i++) {
      eachwords = filedata[i].split(' ');
      for (let k = 0; k < eachwords.length; k++) {
        if (eachwords[k] === '') {
          continue;
        }
        distance = "";
        eachwords[k] = eachwords[k].replace(/[^a-zA-Z ]/g, "");
        if (i == 0 && k == 0) {
          word1 = eachwords[k];
        } else if (i == 0 && k == 1) {
          word2 = eachwords[k];
          distance = levenshteinDistance(word1, word2);
        } else {
          word1 = word2;
          word2 = eachwords[k];
          distance = levenshteinDistance(word1, word2);
        }
        console.log("word 1= " + word1 + "\t word2 = " + word2 + "\t Distance =" +
distance);
      }
    }
  }
}

const levenshteinDistance = (str1 = '', str2 = '') => {
  const track = Array(str2.length + 1).fill(null).map(() =>
    Array(str1.length + 1).fill(null));
  for (let i = 0; i <= str1.length; i += 1) {
    track[0][i] = i;
  }
}

```

```

    }
    for (let j = 0; j <= str2.length; j += 1) {
        track[j][0] = j;
    }
    for (let j = 1; j <= str2.length; j += 1) {
        for (let i = 1; i <= str1.length; i += 1) {
            const indicator = str1[i - 1] === str2[j - 1] ? 0 : 1;
            track[j][i] = Math.min(
                track[j][i - 1] + 1, // deletion
                track[j - 1][i] + 1, // insertion
                track[j - 1][i - 1] + indicator, // substitution
            );
        }
    }
    return track[str2.length][str1.length];
};
};

```

Main.java Code:

```

/**
 * Responds to any HTTP request.
 *
 * @param {!express:Request} req HTTP request context.
 * @param {!express:Response} res HTTP response context.
 */
exports.helloWorld = (req, res) => {
    let message = req.query.message || req.body.message || 'Hello World!';
    res.status(200).send(message);
    if (err && err.code === 'NotFound') {
        console.log("ObjectNot found")
    } else {
        let filedata = metadata.Body.toString('utf-8');

        filedata = filedata.split("\n")
        console.log()
        word1 = "";
        word2 = "";

        for (let i = 0; i < filedata.length; i++) {
            eachwords = filedata[i].split(' ');
            for (let k = 0; k < eachwords.length; k++) {
                if (eachwords[k] === '') {
                    continue;
                }
                distance = "";
                eachwords[k] = eachwords[k].replace(/^[a-zA-Z ]/g, "");
                if (i == 0 && k == 0) {

```



```

        word1 = eachwords[k];
    } else if (i == 0 && k == 1) {
        word2 = eachwords[k];
        distance = levenshteinDistance(word1, word2);
    } else {
        word1 = word2;
        word2 = eachwords[k];
        distance = levenshteinDistance(word1, word2);
    }
    console.log("word 1= " + word1 + "\t word2 = " + word2 + "\t Distance =" +
distance);
    }
}

const levenshteinDistance = (str1 = '', str2 = '') => {
    const track = Array(str2.length + 1).fill(null).map(() =>
        Array(str1.length + 1).fill(null));
    for (let i = 0; i <= str1.length; i += 1) {
        track[0][i] = i;
    }
    for (let j = 0; j <= str2.length; j += 1) {
        track[j][0] = j;
    }
    for (let j = 1; j <= str2.length; j += 1) {
        for (let i = 1; i <= str1.length; i += 1) {
            const indicator = str1[i - 1] === str2[j - 1] ? 0 : 1;
            track[j][i] = Math.min(
                track[j][i - 1] + 1, // deletion
                track[j - 1][i] + 1, // insertion
                track[j - 1][i - 1] + indicator, // substitution
            );
        }
    }
    return track[str2.length][str1.length];
};
};

```