CSCI 5410 - Serverless Data Processing

Assignment - 4

Name	Banner Id	Mail id
Bommera Geetanjali	B00881511	gt584369@dal.ca

Table of Contents

1. Literature Study 3
   1. Sage maker 3
   2. Comprehend 3
2. GCP Machine Learning 4
3. AWS Comprehend 13
4. References 16
5. Git Link 17

1. Literature Study

1. Sage maker

Sage Maker [1] is the integrated tool used for machine learning. It includes visual editors, debuggers, profilers, CI/CD for building machine learning model. Huge data can be imported for S3 or reddit and process, combine and transform the data. It also helps in created customized features in fractions of seconds with saga data wrangler. Sage maker clarify helps in checking the data is balanced which helps in evaluating the model.

Considering car rental application, S3 helps in storing the huge data collected every day and Sage maker helps in data transformation, cleaning, processing, and extracting relevant features from the data. It also helps in building, training, and deploying the model. When model is trained periodically for instance, observing the results might give knowledge on facts and help in better prediction for future. It also helps to observe the continuous changes happening that helps in taking business decisions.

2. Comprehend

Amazon comprehend [2] on the other hand helps in finding the insights from the data in different formats. Everyday lot of data is created in the form of files, mails, documents etc. It takes lot of time for a human to go through each of these data forms and find the insights in them. Amazon comprehend helps in this process. It can go through huge number of documents and label them based on the data in few seconds. It develops intents based on language, key words, entities, syntax, sentiment, and personal information.

Considering car rental application, we can give number of topics we want to know from the given data for example, busy hours or customer ratings etc. Based on this we can find how the customer is responding to our application. Analyzing these documents and results helps us find the areas where we can improve or change to make the application better.

In this process we deal with private data of customers such as customer address, date of birth, bank details are any other information. Comprehend helps in securing this information. It provides an option to encrypt with the help of key before storing data in S3.

Comprehend also helps in running analysis jobs on the data. These jobs can also be encrypted. Output of the jobs can be secured. Also, we can give access to only few roles to access the results. Output can be downloaded in different formats. Comprehend helps in visualizing the results with the help of Amazon Quick Sight Console.

2. GCP Machine Learning

1. Created four buckets as below.
2. Code to create buckets and upload files is also attached in GcpUploadFiles.java file.
3.

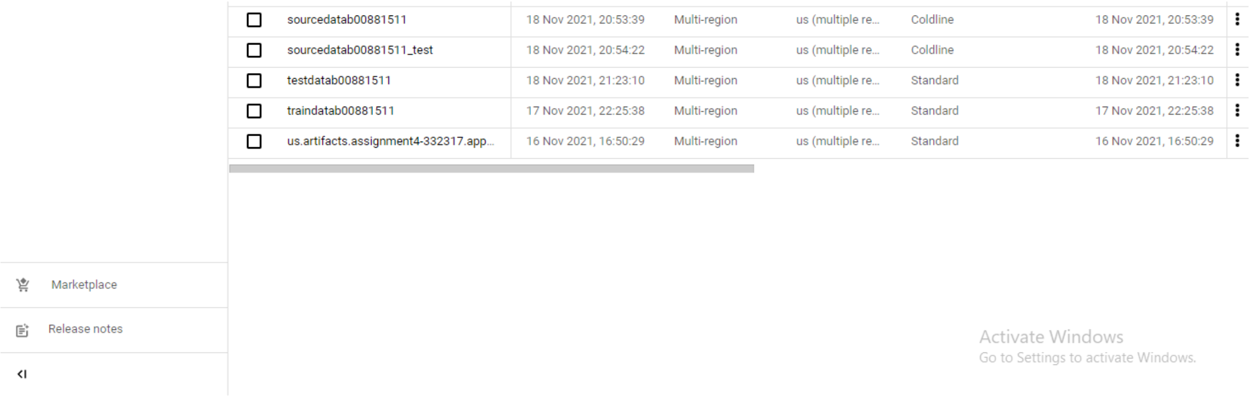
```
import com.google.auth.oauth2.GoogleCredentials;
import com.google.cloud.storage.*;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class GcpUploadFiles {
    public static void main(String[] args) throws IOException {
//         method to create bucket
        createbucket();
//       Upload files in the bucket
        uploadfiles();
    }

    private static void uploadfiles() throws IOException {
//         References for upload files.
        // https://googleapis.dev/java/google-cloud-
clients/latest/com/google/cloud/storage/StorageClass.html

        // http://g.co/cloud/storage/docs/bucket-locations#location-mr
        String projectId = "assignment4-332317";
        String bucketName = "sourcedatab00881511_test";
        String objectName = "train_data";
        StorageOptions storageOptions = StorageOptions.newBuilder()
                .setProjectId(projectId)
                .setCredentials(GoogleCredentials.fromStream(new

FileInputStream("C:\\Users\\geeta\\OneDrive\\Documents\\Serverless\\Ser
verlessAssignment4\\PartB\\src\\main\\java\\assignment4-332317-
c348f6b12aac.json"))).build();
        Storage storage = storageOptions.getService();


        String location = "US";

        String path =
"C:\\Users\\geeta\\OneDrive\\Documents\\Serverless\\ServerlessAssignmen
t4\\PartB\\src\\main\\resources\\Dataset\\Dataset\\Test";
        File dir = new File(path);
        File[] directoryListing = dir.listFiles();
        for (File child : directoryListing) {
            objectName = child.getName();
            BlobId blobId = BlobId.of(bucketName, objectName);
            BlobInfo blobInfo = BlobInfo.newBuilder(blobId).build();
            storage.create(blobInfo,
Files.readAllBytes(Paths.get(String.valueOf(child))));
        }
        System.out.println(
                "Files " + " uploaded to bucket " + bucketName);
    }

    private static void createbucket() throws IOException {
//        References to create bucket

//https://cloud.google.com/appengine/docs/standard/java11/specifying-
dependencies
        //https://cloud.google.com/storage/docs/creating-
buckets#storage-create-bucket-code_samples
        String projectId = "assignment4-332317";
        String bucketName = "sourcedatab00881511";
        StorageOptions storageOptions = StorageOptions.newBuilder()
                .setProjectId(projectId)
                .setCredentials(GoogleCredentials.fromStream(new

FileInputStream("C:\\Users\\geeta\\OneDrive\\Documents\\Serverless\\Ser
verlessAssignment4\\PartB\\src\\main\\java\\assignment4-332317-
c348f6b12aac.json"))).build();
        Storage storage = storageOptions.getService();

        // https://googleapis.dev/java/google-cloud-
clients/latest/com/google/cloud/storage/StorageClass.html
        StorageClass storageClass = StorageClass.COLDLINE;

        // http://g.co/cloud/storage/docs/bucket-locations#location-mr
        String location = "US";

        Bucket bucket =
                storage.create(
                        BucketInfo.newBuilder(bucketName)
                                .setStorageClass(storageClass)
                                .build());

        System.out.println(
                "Created bucket "
                        + bucket.getName()
                        + " in "
                        + bucket.getLocation()
                        + " with storage class "
                        + bucket.getStorageClass());

    }
}
```

a. To store train data – files from 0 to 299

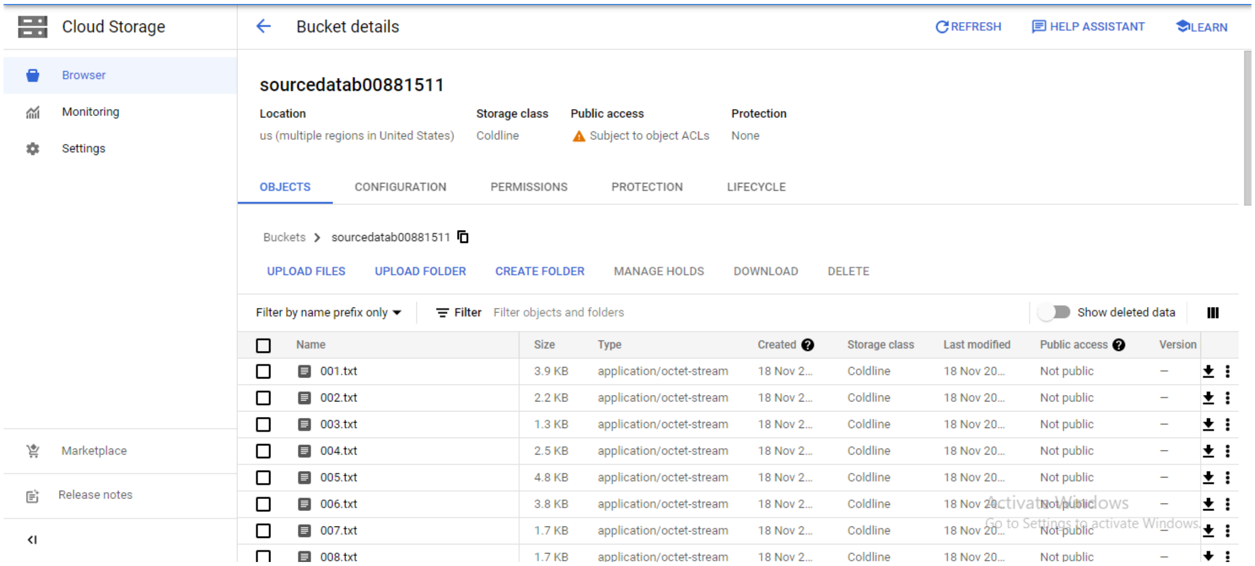
b. To store test data – files from 300 to 401

c. To store csv file of train data: In order to reduce the execution time. Few random instances are considered to save in csv file

d. To store csv file of test data: In order to reduce the execution time. Few random instances are considered to save in csv file



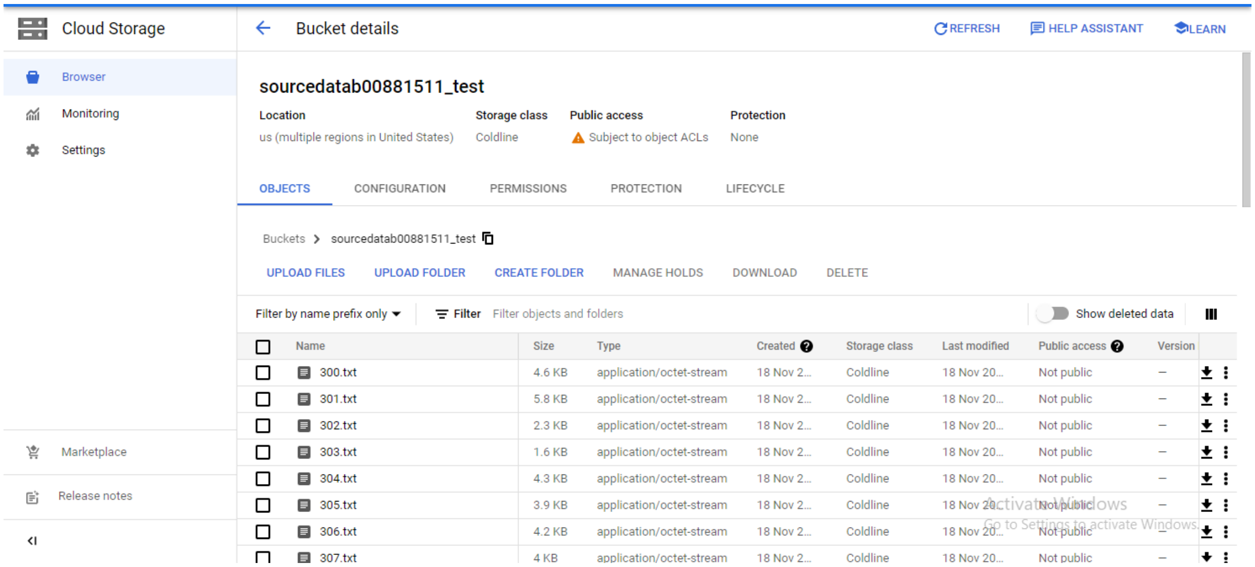
sourcedatab00881511	18 Nov 2021, 20:53:39	Multi-region	us (multiple re...	Coldline	18 Nov 2021, 20:53:39
sourcedatab00881511_test	18 Nov 2021, 20:54:22	Multi-region	us (multiple re...	Coldline	18 Nov 2021, 20:54:22
testdatab00881511	18 Nov 2021, 21:23:10	Multi-region	us (multiple re...	Standard	18 Nov 2021, 21:23:10
traindatab00881511	17 Nov 2021, 22:25:38	Multi-region	us (multiple re...	Standard	17 Nov 2021, 22:25:38
us.artifacts.assignment4-332317.app...	16 Nov 2021, 16:50:29	Multi-region	us (multiple re...	Standard	16 Nov 2021, 16:50:29



sourcedatab00881511

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Coldline	Subject to object ACLs	None

Name	Size	Type	Created	Storage class	Last modified	Public access	Version
001.txt	3.9 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
002.txt	2.2 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
003.txt	1.3 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
004.txt	2.5 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
005.txt	4.8 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
006.txt	3.8 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
007.txt	1.7 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
008.txt	1.7 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–

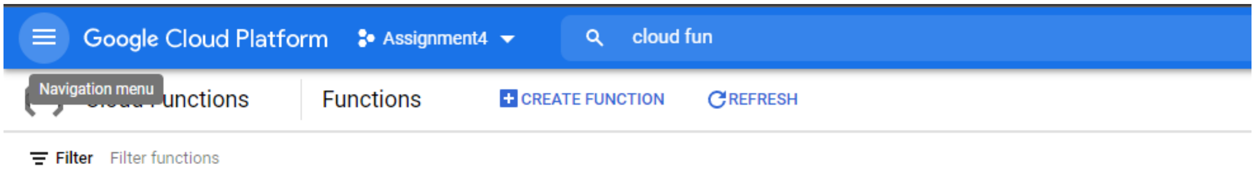


sourcedatab00881511_test

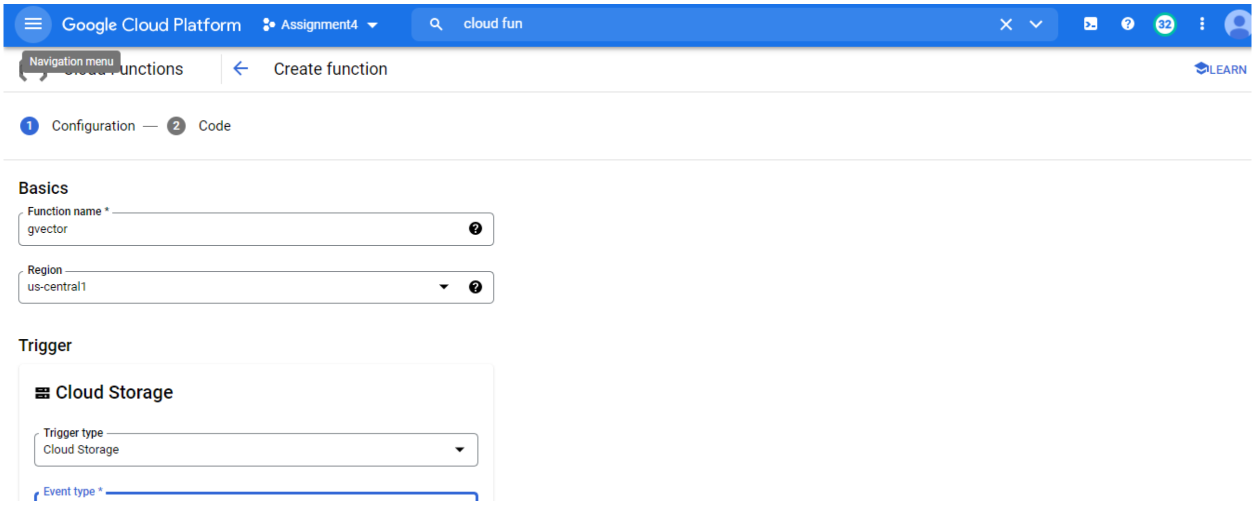
Location	Storage class	Public access	Protection
us (multiple regions in United States)	Coldline	Subject to object ACLs	None

Name	Size	Type	Created	Storage class	Last modified	Public access	Version
300.txt	4.6 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
301.txt	5.8 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
302.txt	2.3 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
303.txt	1.6 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
304.txt	4.3 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
305.txt	3.9 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
306.txt	4.2 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–
307.txt	4 KB	application/octet-stream	18 Nov 2...	Coldline	18 Nov 20...	Not public	–

2. Create cloud function to take data from first two files and process to store in last two buckets correspondingly.
   a. Search for cloud function in search and click on it.
   b. Click on create function.



   c. Give the function name, location and select trigger as cloud storage by giving the bucket name. Click on next to create the function



Basics

Function name *
gvector

Region
us-central1

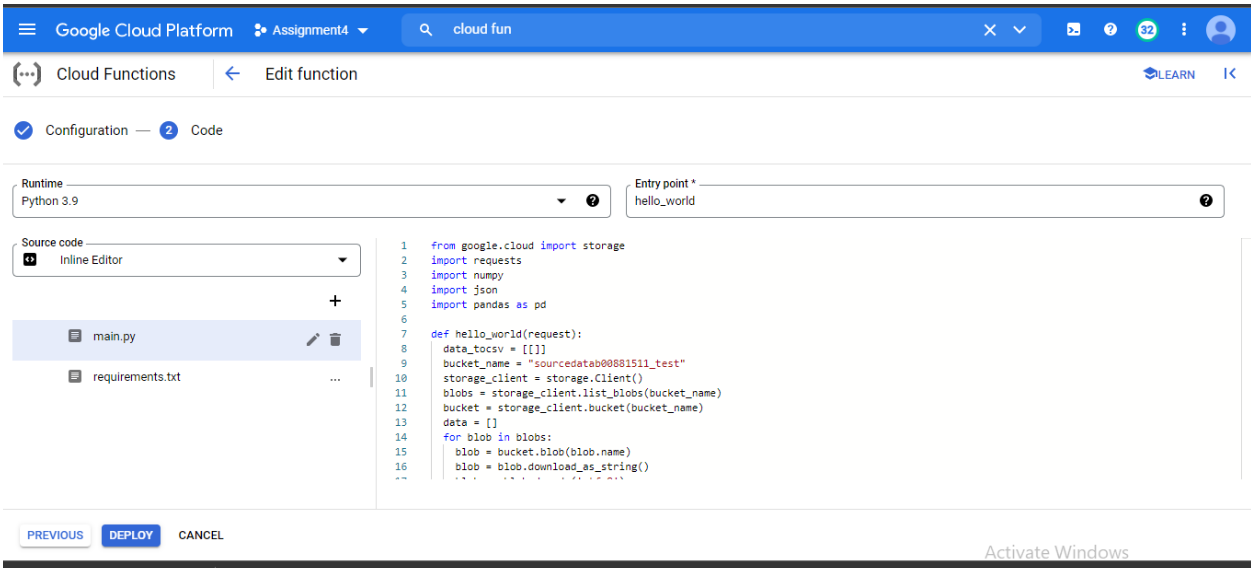
Trigger

Cloud Storage

Trigger type
Cloud Storage

Event type *

   d. New window to write the code is displayed as below.



Runtime: Python 3.9

Entry point *: hello_world

Source code: Inline Editor

main.py

requirements.txt

```python
from google.cloud import storage
import requests
import numpy
import json
import pandas as pd

def hello_world(request):
  data_tocsv = [[]]
  bucket_name = "sourcedatab00881511_test"
  storage_client = storage.Client()
  blobs = storage_client.list_blobs(bucket_name)
  bucket = storage_client.bucket(bucket_name)
  data = []
  for blob in blobs:
    blob = bucket.blob(blob.name)
    blob = blob.download_as_string()
```

e. Include all the dependencies in the requirements.txt [3] and write the code in main.py.

f. Once done click on deploy to deploy the function.

g. Deployment of function takes time based on data present in the files.

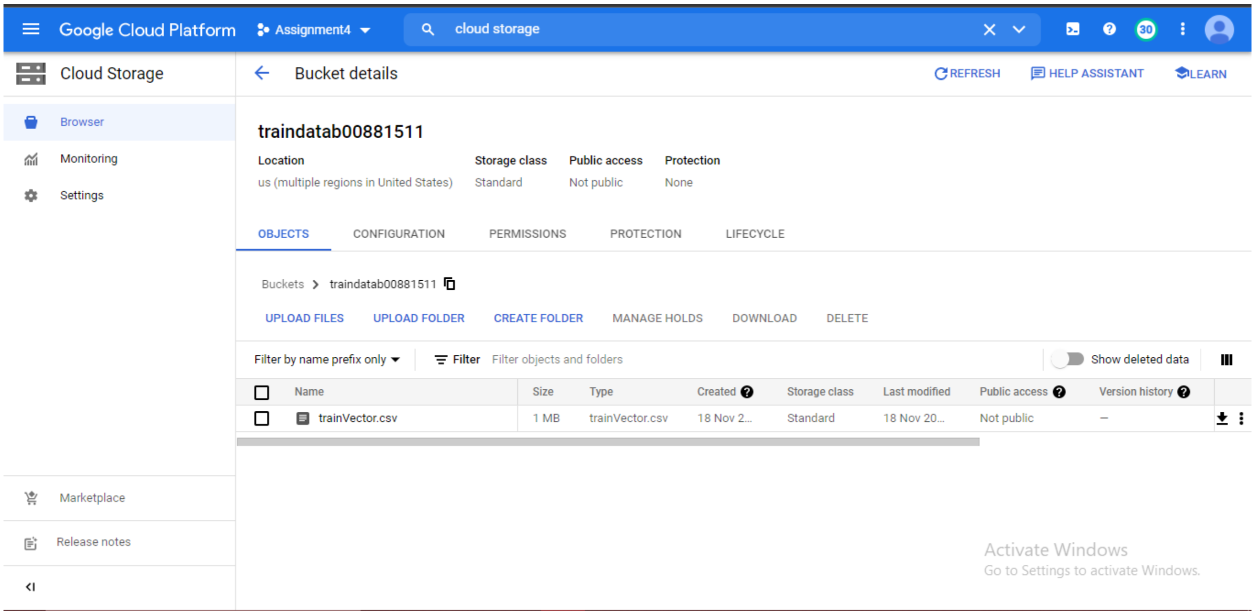
h. Errors can be seen in the log tab after testing.

i. References: Levenshtein distance [4]. Levenshtein distance can be calculated using python library also.

j. Code given in cloud function is as below.

```python
from google.cloud import storage
import requests
import numpy
import json
import pandas as pd

def hello_world(request):
  data_tocsv = [[]]
  bucket_name = "sourcedatab00881511_test"
  storage_client = storage.Client()
  blobs = storage_client.list_blobs(bucket_name)
  bucket = storage_client.bucket(bucket_name)
  data = []
  for blob in blobs:
    blob = bucket.blob(blob.name)
    blob = blob.download_as_string()
    blobx = blob.decode('utf-8')
    temp = blobx.split()
    for i in temp:
      data.append(i)
  stopwords = ['i', 'me', "the", "The",'my', 'myself', 'we', 'our', 'ours', 'ourselves
', 'you', "you're", "you've", "you'll", "you'd",'your', 'yours', 'yourself', 'yourselv
es', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers','herself', 'it', "it
's", 'its', 'itself', 'they', 'them','their', 'theirs', 'themselves', 'what', 'which',
'who', 'whom', 'this', 'that', "that'll", 'these', 'those','am', 'is', 'are', 'was', '
were', 'be', 'been','being', 'have', 'has', 'had', 'having', 'do', 'does', 'did','doin
g', 'a', 'an', 'the', 'and', 'but','if', 'or', 'because', 'as', 'until', 'while', 'of'
, 'at','by', 'for', 'with', 'about', 'against','between', 'into', 'through', 'during',
 'before', 'after','above', 'below', 'to', 'from', 'up', 'down','in', 'out', 'on', 'of
f', 'over', 'under', 'again','further', 'then', 'once', 'here', 'there', 'when','where
', 'why', 'how', 'all', 'any', 'both', 'each', 'few','more', 'most', 'other', 'some',
'such', 'no','nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too','very', 's', 't'
, 'can', 'will', 'just', 'don',"don't", 'should', "should've", 'now', 'd', 'll', 'm',
'o','re', 've', 'y', 'ain', 'aren', "aren't",'couldn', "couldn't", 'didn', "didn't", '
doesn', "doesn't",'hadn', "hadn't", 'hasn', "hasn't", 'haven',"haven't", 'isn', "isn't
", 'ma', 'mightn', "mightn't",'mustn', "mustn't", 'needn', "needn't", 'shan',"shan't",
 'shouldn', "a", "shouldn't", 'wasn', "wasn't",'weren', "weren't", 'won', "won't", 'wo
uldn',"wouldn't"]
  for each in stopwords:  # iterating on a copy since removing will mess things up
    for word in data:
      if (each == word):
        data.remove(each)
  ndata = numpy.array([[]])
  for i in range(len(data)-1):
    distance = levenshteinDistanceDP(data[i], data[i+1])
    data_tocsv.append([data[i],data[i+1],distance])
  df = pd.DataFrame(data=data_tocsv, columns=["firstword", "nextword","distance"])
  destination_bucket = storage_client.get_bucket('testdatab00881511')
  destination_bucket.blob('testVector.csv').upload_from_string(df.to_csv(), 'testVecto
r.csv')
  return f'Success!'

def levenshteinDistanceDP(token1, token2):
    distances = numpy.zeros((len(token1) + 1, len(token2) + 1))

    for t1 in range(len(token1) + 1):
        distances[t1][0] = t1

    for t2 in range(len(token2) + 1):
        distances[0][t2] = t2

    a = 0
    b = 0
    c = 0

    for t1 in range(1, len(token1) + 1):
        for t2 in range(1, len(token2) + 1):
            if (token1[t1-1] == token2[t2-1]):
                distances[t1][t2] = distances[t1 - 1][t2 - 1]
            else:
                a = distances[t1][t2 - 1]
                b = distances[t1 - 1][t2]
                c = distances[t1 - 1][t2 - 1]
                if (a <= b and a <= c):
                    distances[t1][t2] = a + 1
                elif (b <= a and b <= c):
                    distances[t1][t2] = b + 1
                else:
                    distances[t1][t2] = c + 1

    return distances[len(token1)][len(token2)]
```

3. Once files are uploaded to source buckets, csv file is generated in train and test buckets.
4. TrainVector.csv has data as Index, First word, Next word and distance.



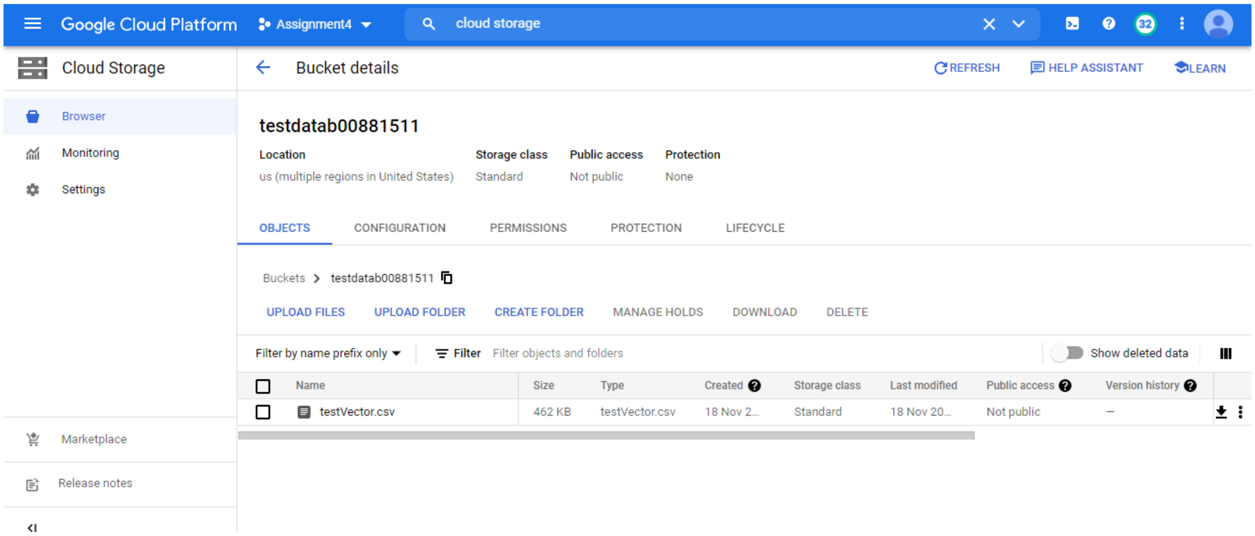
traindatab00881511

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	None

Buckets > traindatab00881511

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history
trainVector.csv	1 MB	trainVector.csv	18 Nov 2...	Standard	18 Nov 20...	Not public	–

5. TestVector.csv has data as Index, First word, Next word and distance similar to trainvector.csv



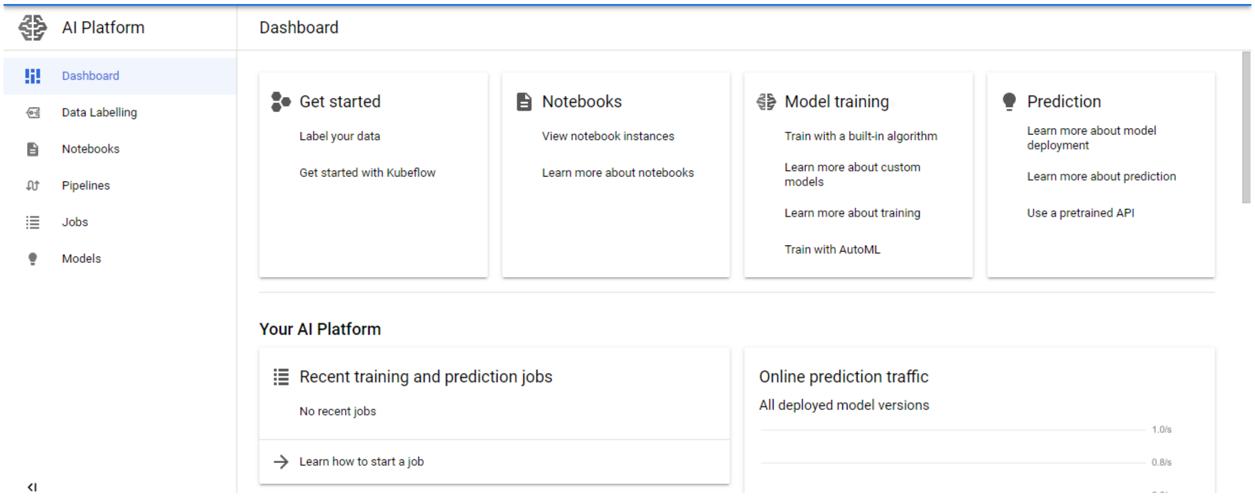
testdatab00881511

Location	Storage class	Public access	Protection
us (multiple regions in United States)	Standard	Not public	None

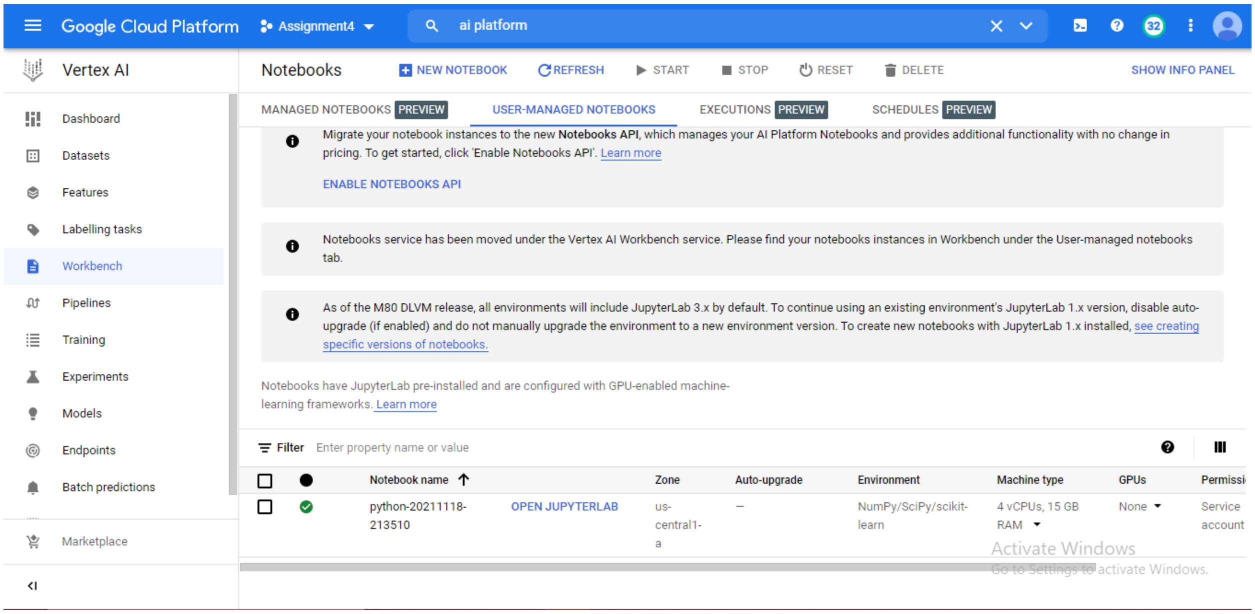
Buckets > testdatab00881511

Name	Size	Type	Created	Storage class	Last modified	Public access	Version history
testVector.csv	462 KB	testVector.csv	18 Nov 2...	Standard	18 Nov 20...	Not public	–

6. Go to AI platform [5] to create Machine learning instance. Click on notebooks and new notebook.


AI Platform
Dashboard
Dashboard
Data Labelling
Notebooks
Pipelines
Jobs
Models
Get started
Label your data
Get started with Kubeflow
Notebooks
View notebook instances
Learn more about notebooks
Model training
Train with a built-in algorithm
Learn more about custom models
Learn more about training
Train with AutoML
Prediction
Learn more about model deployment
Learn more about prediction
Use a pretrained API
Your AI Platform
Recent training and prediction jobs
No recent jobs
Learn how to start a job
Online prediction traffic
All deployed model versions
1.0/s
0.8/s

7. Once notebook is created click on open Jupiter tab.



Migrate your notebook instances to the new **Notebooks API**, which manages your AI Platform Notebooks and provides additional functionality with no change in pricing. To get started, click 'Enable Notebooks API'. Learn more

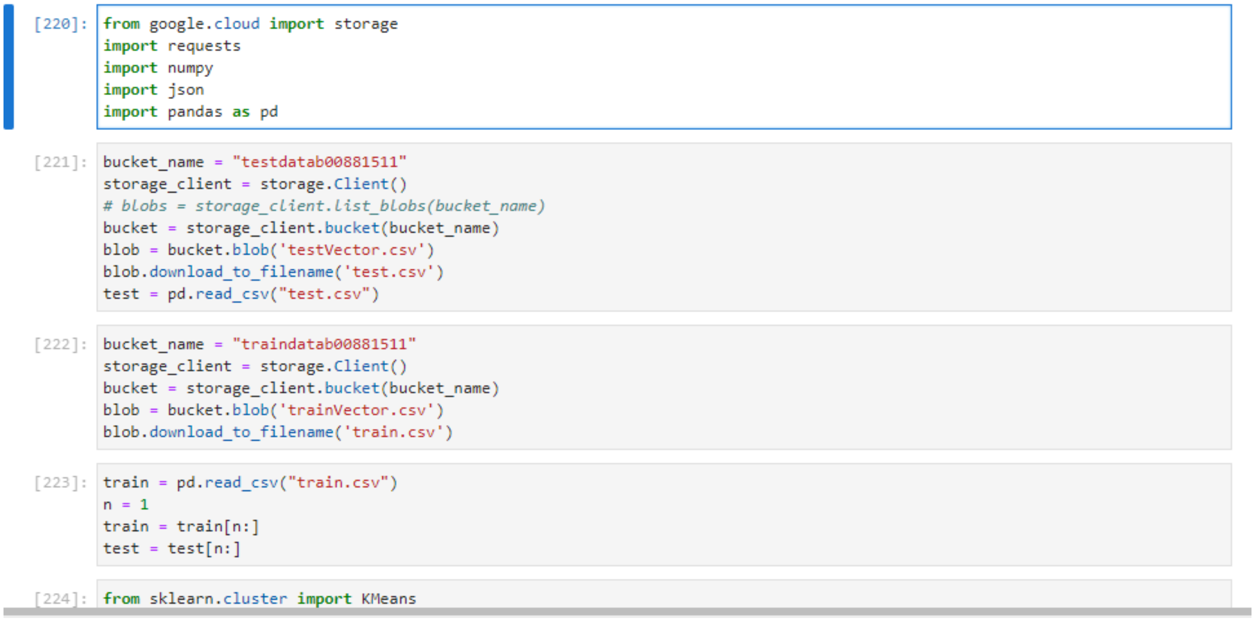
Notebooks service has been moved under the Vertex AI Workbench service. Please find your notebooks instances in Workbench under the User-managed notebooks tab.

As of the M80 DLVM release, all environments will include JupyterLab 3.x by default. To continue using an existing environment's JupyterLab 1.x version, disable auto-upgrade (if enabled) and do not manually upgrade the environment to a new environment version. To create new notebooks with JupyterLab 1.x installed, see creating specific versions of notebooks.

Notebooks have JupyterLab pre-installed and are configured with GPU-enabled machine-learning frameworks. Learn more

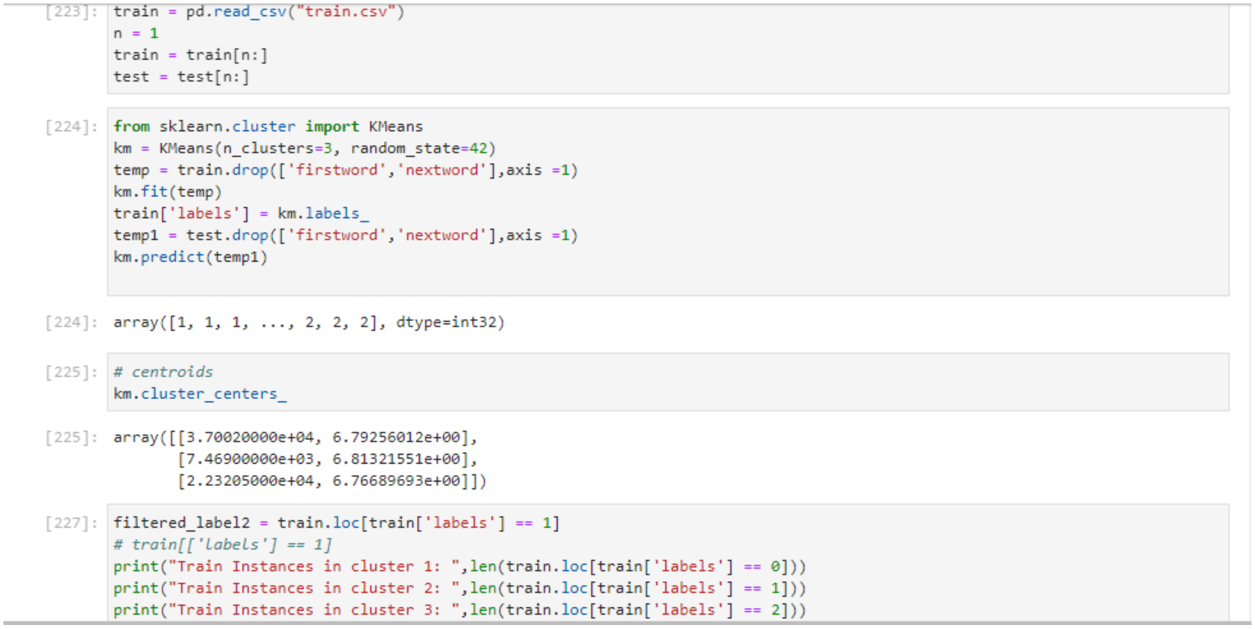
□	●	Notebook name ↑		Zone	Auto-upgrade	Environment	Machine type	GPUs	Permissi
□	✓	python-20211118-213510	OPEN JUPYTERLAB	us-central1-a	–	NumPy/SciPy/scikit-learn	4 vCPUs, 15 GB RAM	None	Service account

2. Write the code in Jupiter notebook and perform clustering. Results of clustering are as below. Code for the clustering is attached in ML_kmeans file.

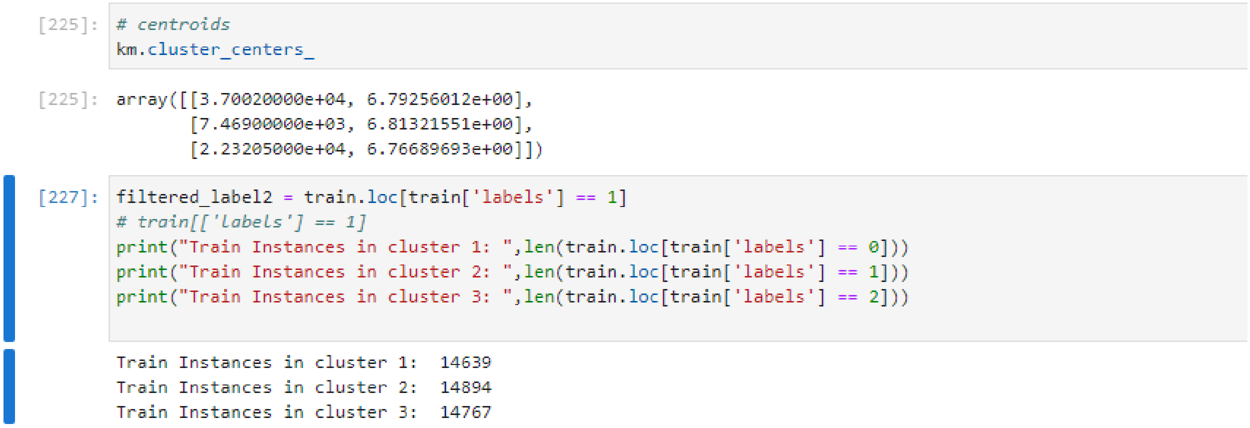


```
[220]: from google.cloud import storage
       import requests
       import numpy
       import json
       import pandas as pd

[221]: bucket_name = "testdatab00881511"
       storage_client = storage.Client()
       # blobs = storage_client.list_blobs(bucket_name)
       bucket = storage_client.bucket(bucket_name)
       blob = bucket.blob('testVector.csv')
       blob.download_to_filename('test.csv')
       test = pd.read_csv("test.csv")

[222]: bucket_name = "traindatab00881511"
       storage_client = storage.Client()
       bucket = storage_client.bucket(bucket_name)
       blob = bucket.blob('trainVector.csv')
       blob.download_to_filename('train.csv')

[223]: train = pd.read_csv("train.csv")
       n = 1
       train = train[n:]
       test = test[n:]

[224]: from sklearn.cluster import KMeans
```

3.



```
[223]: train = pd.read_csv("train.csv")
       n = 1
       train = train[n:]
       test = test[n:]

[224]: from sklearn.cluster import KMeans
       km = KMeans(n_clusters=3, random_state=42)
       temp = train.drop(['firstword','nextword'],axis =1)
       km.fit(temp)
       train['labels'] = km.labels_
       temp1 = test.drop(['firstword','nextword'],axis =1)
       km.predict(temp1)

[224]: array([1, 1, 1, ..., 2, 2, 2], dtype=int32)

[225]: # centroids
       km.cluster_centers_

[225]: array([[3.70020000e+04, 6.79256012e+00],
              [7.46900000e+03, 6.81321551e+00],
              [2.23205000e+04, 6.76689693e+00]])

[227]: filtered_label2 = train.loc[train['labels'] == 1]
       # train[['labels'] == 1]
       print("Train Instances in cluster 1: ",len(train.loc[train['labels'] == 0]))
       print("Train Instances in cluster 2: ",len(train.loc[train['labels'] == 1]))
       print("Train Instances in cluster 3: ",len(train.loc[train['labels'] == 2]))
```

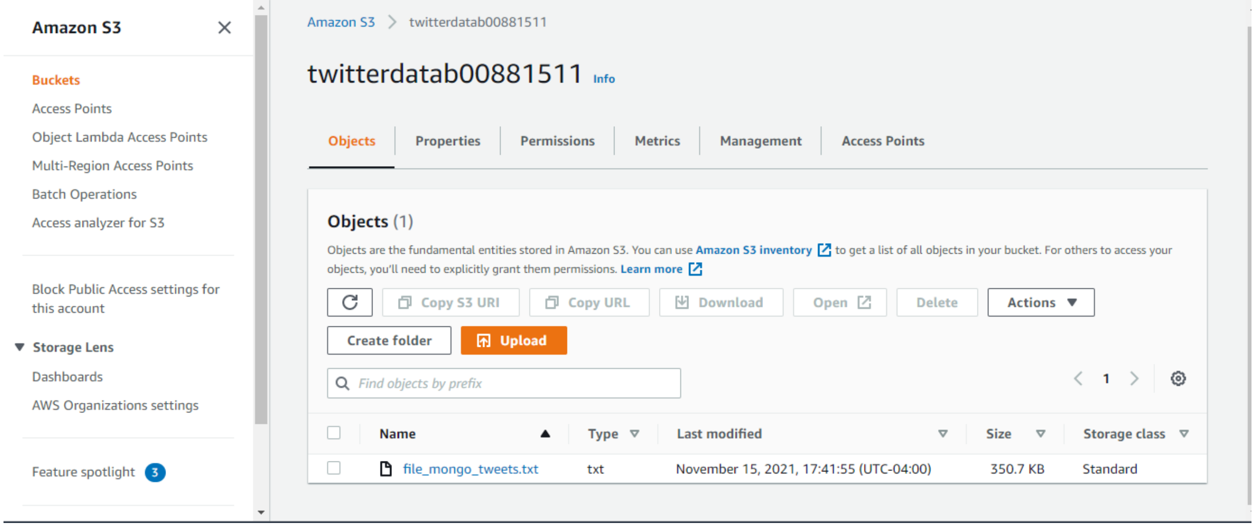


```
[225]: # centroids
       km.cluster_centers_

[225]: array([[3.70020000e+04, 6.79256012e+00],
              [7.46900000e+03, 6.81321551e+00],
              [2.23205000e+04, 6.76689693e+00]])

[227]: filtered_label2 = train.loc[train['labels'] == 1]
       # train[['labels'] == 1]
       print("Train Instances in cluster 1: ",len(train.loc[train['labels'] == 0]))
       print("Train Instances in cluster 2: ",len(train.loc[train['labels'] == 1]))
       print("Train Instances in cluster 3: ",len(train.loc[train['labels'] == 2]))

       Train Instances in cluster 1:  14639
       Train Instances in cluster 2:  14894
       Train Instances in cluster 3:  14767
```

3. AWS Comprehend

Steps to use Amazon Comprehend [6].

1. Add document to the S3 bucket. File is uploaded in the bucket using Java SDK.

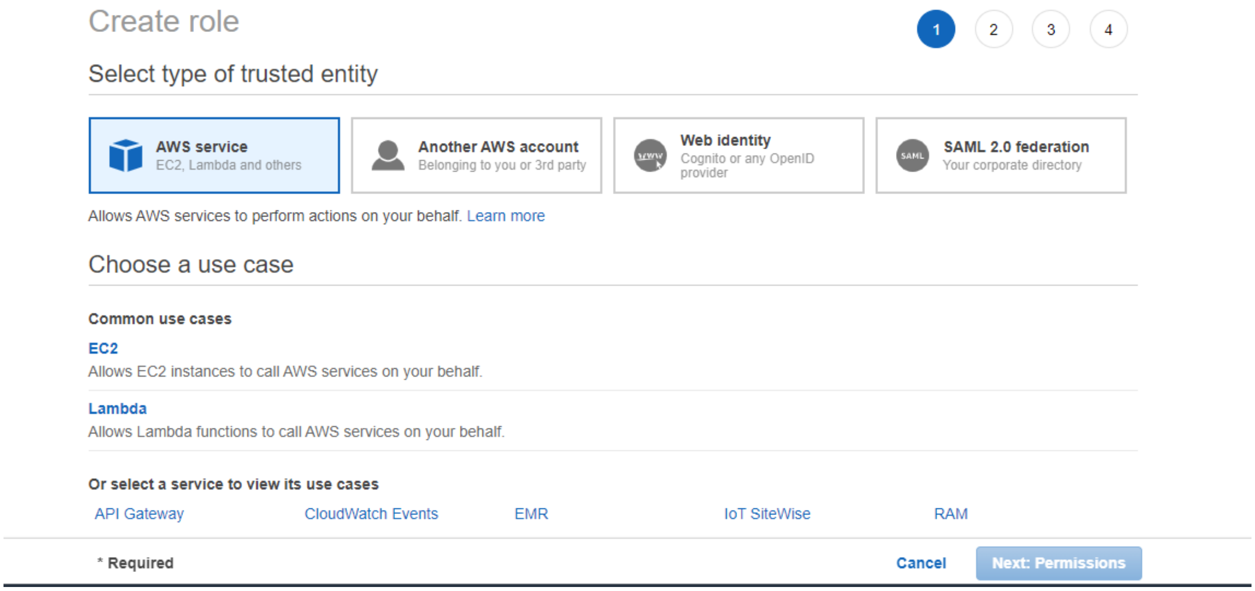


Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
file_mongo_tweets.txt	txt	November 15, 2021, 17:41:55 (UTC-04:00)	350.7 KB	Standard

2. Creating IAM role for amazon comprehend.



Create role

Select type of trusted entity

Allows AWS services to perform actions on your behalf. Learn more

Choose a use case

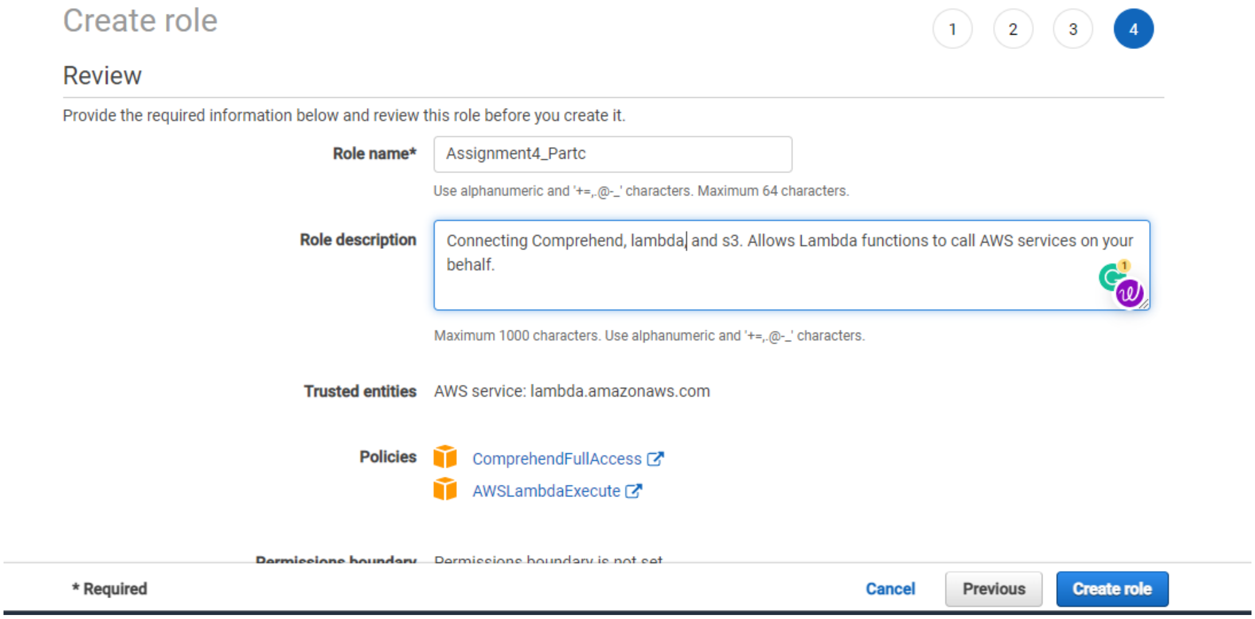
Common use cases

EC2
Allows EC2 instances to call AWS services on your behalf.

Lambda
Allows Lambda functions to call AWS services on your behalf.

Or select a service to view its use cases

3. Select lambda and click on next permissions.



Create role

Review

Provide the required information below and review this role before you create it.

Role name* Assignment4_Partc

Use alphanumeric and '+=,.@-_' characters. Maximum 64 characters.

Role description Connecting Comprehend, lambda and s3. Allows Lambda functions to call AWS services on your behalf.

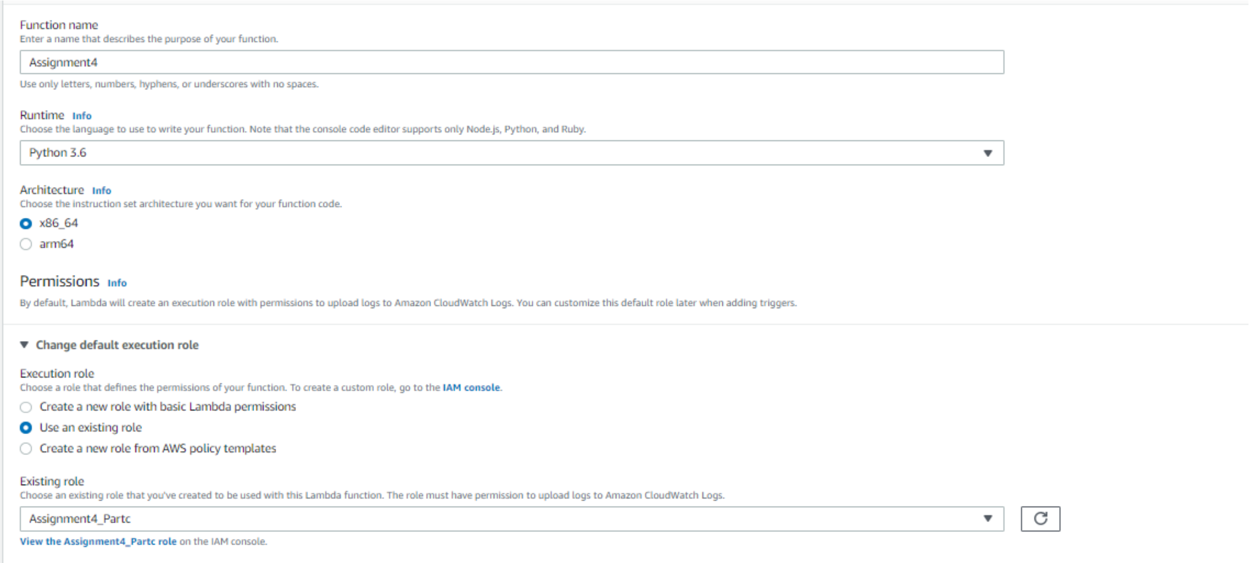
Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

Trusted entities AWS service: lambda.amazonaws.com

Policies ComprehendFullAccess
AWSLambdaExecute

Permissions boundary Permissions boundary is not set

4. Select the policies and create on the create role.
5. Go to amazon comprehend and click on launch amazon comprehend.
6. Creating lambda function to make this event driven.



Function name
Enter a name that describes the purpose of your function.
Assignment4
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.6

Architecture Info
Choose the instruction set architecture you want for your function code.
● x86_64
○ arm64

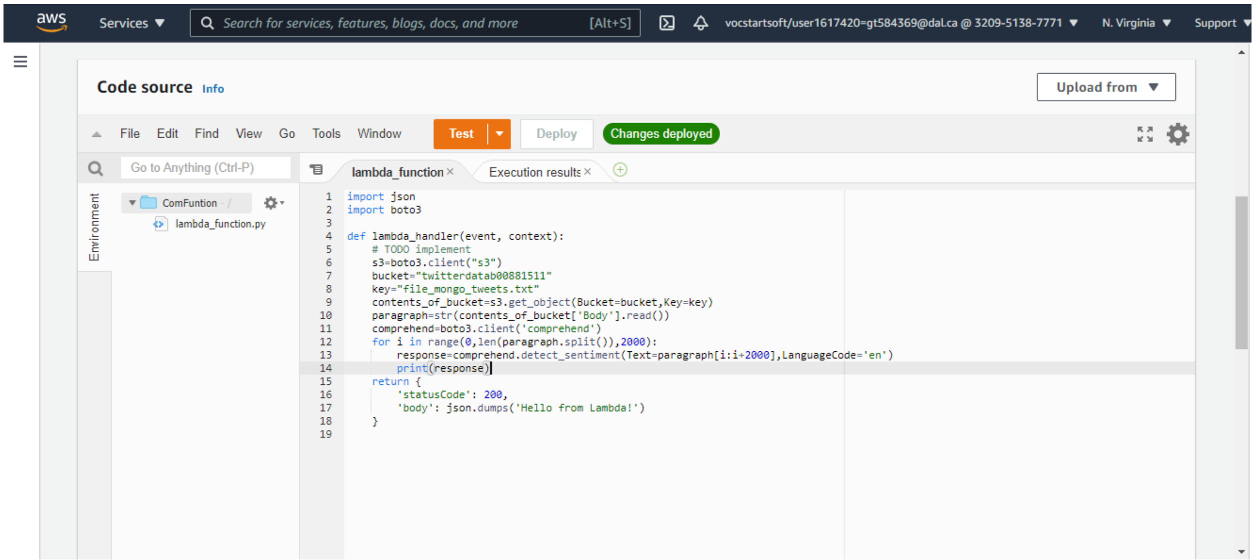
Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.
○ Create a new role with basic Lambda permissions
● Use an existing role
○ Create a new role from AWS policy templates

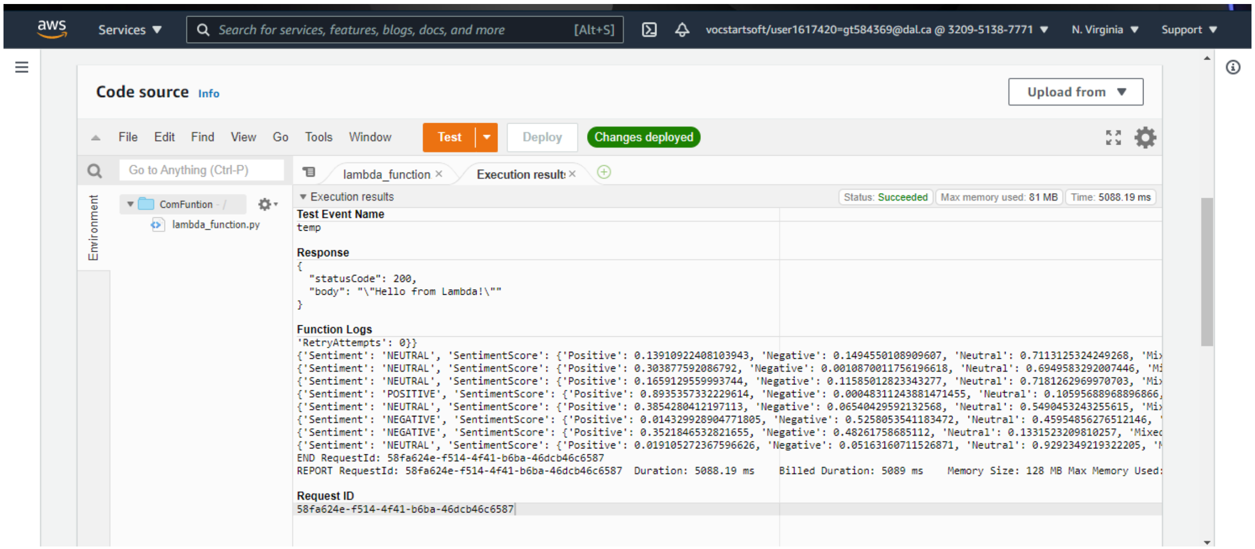
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.
Assignment4_Partc
View the Assignment4_Partc role on the IAM console.

7. Write the code as given below in the lambda function.



```python
import json
import boto3

def lambda_handler(event, context):
    # TODO implement
    s3=boto3.client("s3")
    bucket="twitterdatab00881511"
    key="file_mongo_tweets.txt"
    contents_of_bucket=s3.get_object(Bucket=bucket,Key=key)
    paragraph=str(contents_of_bucket['Body'].read())
    comprehend=boto3.client('comprehend')
    for i in range(0,len(paragraph.split()),2000):
        response=comprehend.detect_sentiment(Text=paragraph[i:i+2000],LanguageCode='en')
        print(response)
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```

8. Response of the comprehend sentimental analysis is in json format.



Status: Succeeded | Max memory used: 81 MB | Time: 5088.19 ms

Test Event Name

```
temp
```

Response

```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

Function Logs

```
'RetryAttempts': 0}}
{'Sentiment': 'NEUTRAL', 'SentimentScore': {'Positive': 0.13910922408103943, 'Negative': 0.1494550108909607, 'Neutral': 0.7113125324249268, 'Mix
{'Sentiment': 'NEUTRAL', 'SentimentScore': {'Positive': 0.303877592086792, 'Negative': 0.0010870011756196618, 'Neutral': 0.6949583292007446, 'Mi
{'Sentiment': 'NEUTRAL', 'SentimentScore': {'Positive': 0.1659129559993744, 'Negative': 0.11585012823343277, 'Neutral': 0.718126296970703, 'Mix
{'Sentiment': 'POSITIVE', 'SentimentScore': {'Positive': 0.8935357332229614, 'Negative': 0.00048311243881471455, 'Neutral': 0.10595688968896866,
{'Sentiment': 'NEUTRAL', 'SentimentScore': {'Positive': 0.385428041219713, 'Negative': 0.0654042959213256, 'Neutral': 0.549045324325561, 'Mix
{'Sentiment': 'NEGATIVE', 'SentimentScore': {'Positive': 0.014329928904771805, 'Negative': 0.5258053541183472, 'Neutral': 0.45954856276512146, '
{'Sentiment': 'NEGATIVE', 'SentimentScore': {'Positive': 0.3521846532821655, 'Negative': 0.48261758685112, 'Neutral': 0.133152320981025, 'Mixed
{'Sentiment': 'NEUTRAL', 'SentimentScore': {'Positive': 0.019105272367596626, 'Negative': 0.05163160711526871, 'Neutral': 0.9292349219322205, 'M
END RequestId: 58fa624e-f514-4f41-b6ba-46dcb46c6587
REPORT RequestId: 58fa624e-f514-4f41-b6ba-46dcb46c6587  Duration: 5088.19 ms    Billed Duration: 5089 ms    Memory Size: 128 MB Max Memory Used:
```

Request ID

```
58fa624e-f514-4f41-b6ba-46dcb46c6587
```

9. Note: Comprehend can also be used directly from comprehend console.

4. References

[1] Amazon, "Amazon SageMaker," [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/howitworks-nbexamples.html. [Accessed 12 November 2021].

[2] Amazon, "Amazon Comprehend," [Online]. Available: https://docs.aws.amazon.com/comprehend/latest/dg/process.html. [Accessed 12 November 2021].

[3] Stackoverflow, "Google Cloud Storage from Cloud Function (python)," [Online]. Available: https://stackoverflow.com/questions/52249978/write-to-google-cloud-storage-from-cloud-function-python. [Accessed 15 November 2021].

[4] PaperspaceBlog, "Implementing The Levenshtein Distance for Word Autocompletion and Autocorrection," [Online]. Available: https://blog.paperspace.com/implementing-levenshtein-distance-word-autocomplete-autocorrect/. [Accessed 15 November 2021].

[5] Amazon, "AI Platform," [Online]. Available: https://console.cloud.google.com/ai-platform/dashboard?project=assignment4-332317. [Accessed 18 November 2021].

[6] Amazon, "Amazon Comprehend," [Online]. Available: https://docs.aws.amazon.com/comprehend/latest/dg/tutorial-reviews.html. [Accessed 12 November 2021].

5. Git Link

https://git.cs.dal.ca/bommera/csci-5410-f2021-b00881511-geetanjali-bommera