COSC-6376 Assignment-4

Shaili Dave

PSID: 2052659

For your 4th Assignment You will do transfer learning on a well-known dataset in Sagemaker (or Vertex AI), and expose that inference engine via an API. The base data would be stored in a database of your choice

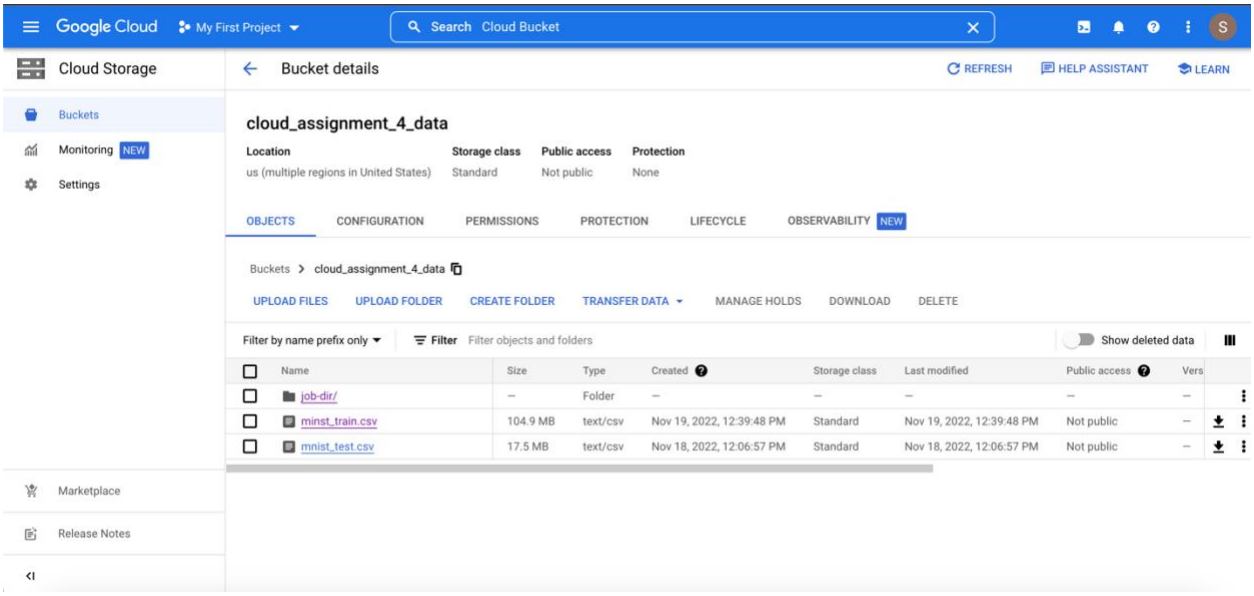
Task in Detail:

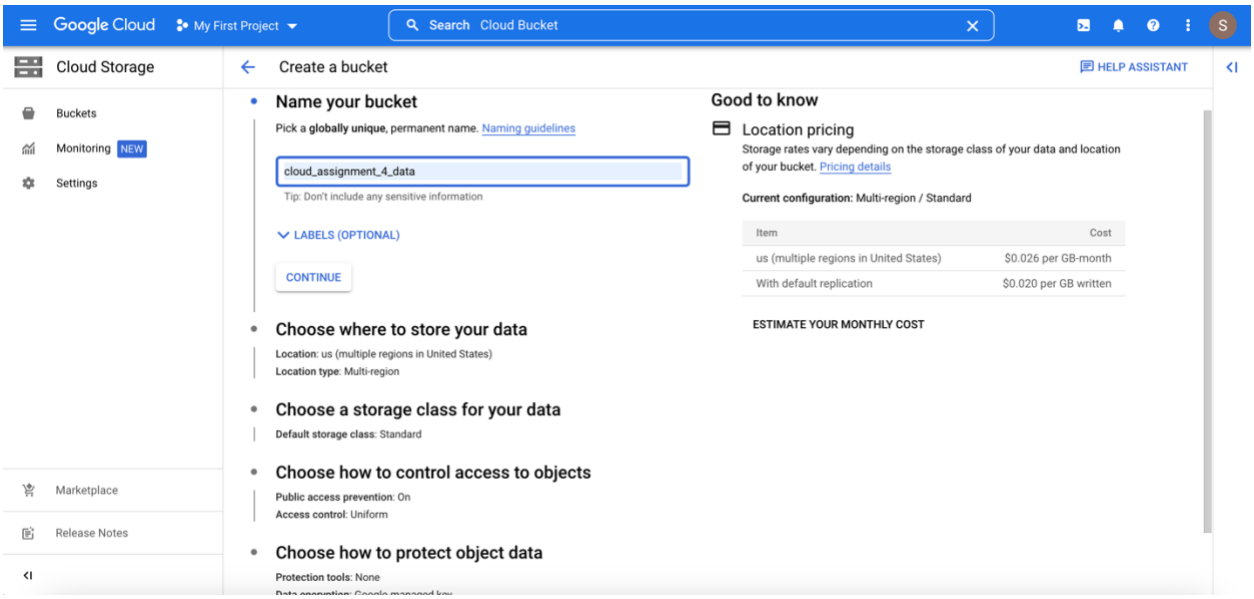
- **You will use the MNIST dataset (https://huggingface.co/datasets/mnist, https://deepai.org/dataset/mnist, https://www.kaggle.com/datasets/oddrationale/mnist-in-csv)**
- **The dataset must be in a database**
- **You will have a sagemaker backend interacting with this data**
- **You will do transfer learning on your own handwritten digits on top of the existing dataset**
- **Generate an API which takes an image data (hosted image so it should take a URl of image as input) and give you the result**

Steps followed in this assignment using Vertex AI on Google Cloud Platform

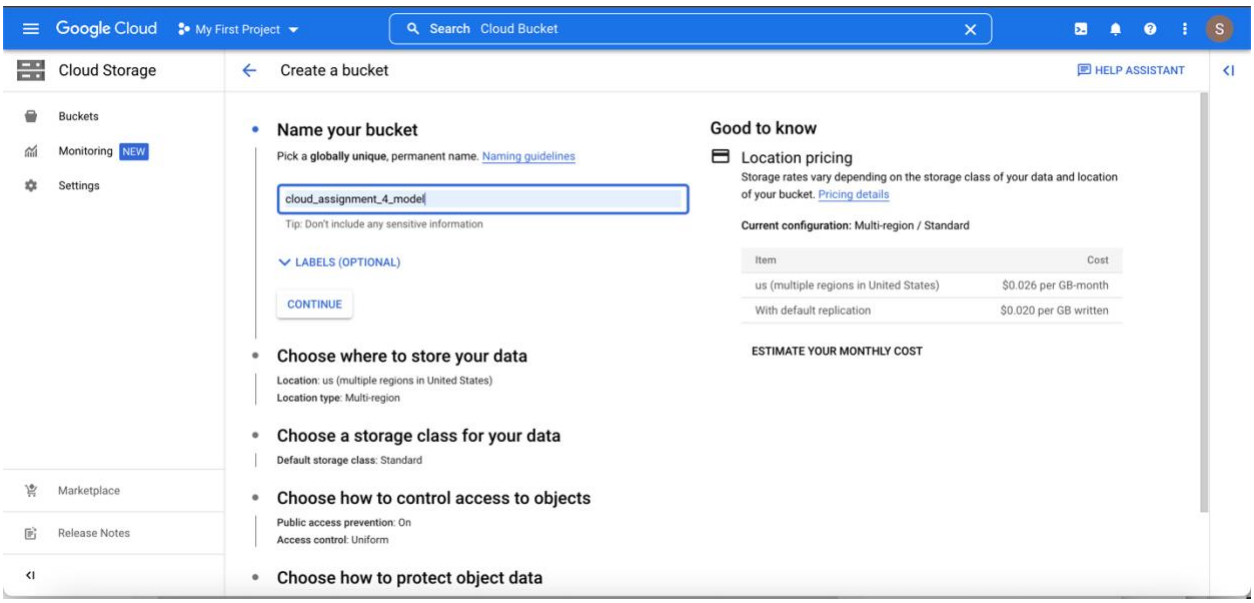
Upload Data to Database (Big Query)

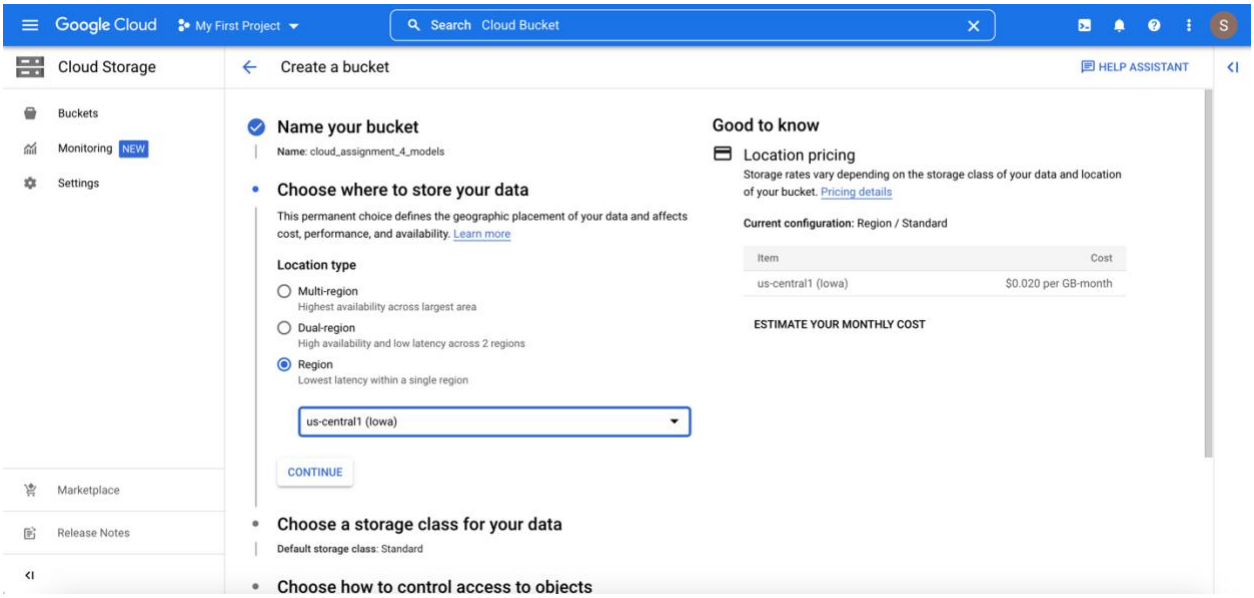
1. Create cloud_assignment_4_data Bucket on Google Cloud Platform and Upload MNIST Data csv files to the bucket



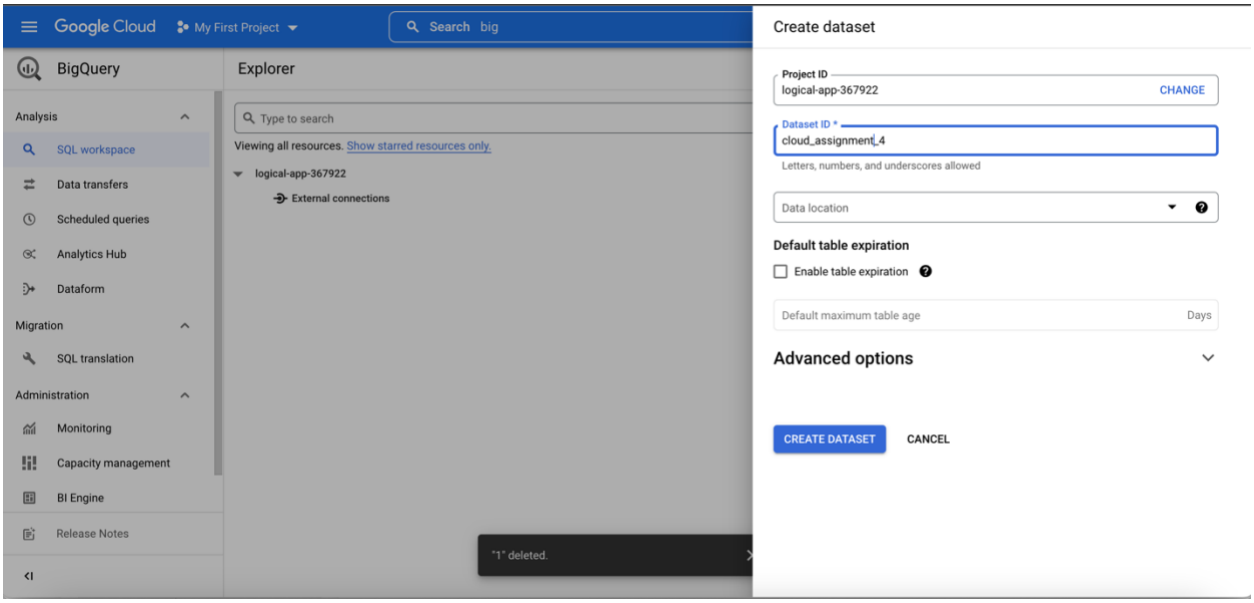


Also, create bucket loud_assignment_4_model in which specify region us-central 1 (lowa). This bucket is created to store model and we will specify region same in our vertex ai workbench notebook.
It is necessary in google cloud that all resources should be same region otherwise it will give error

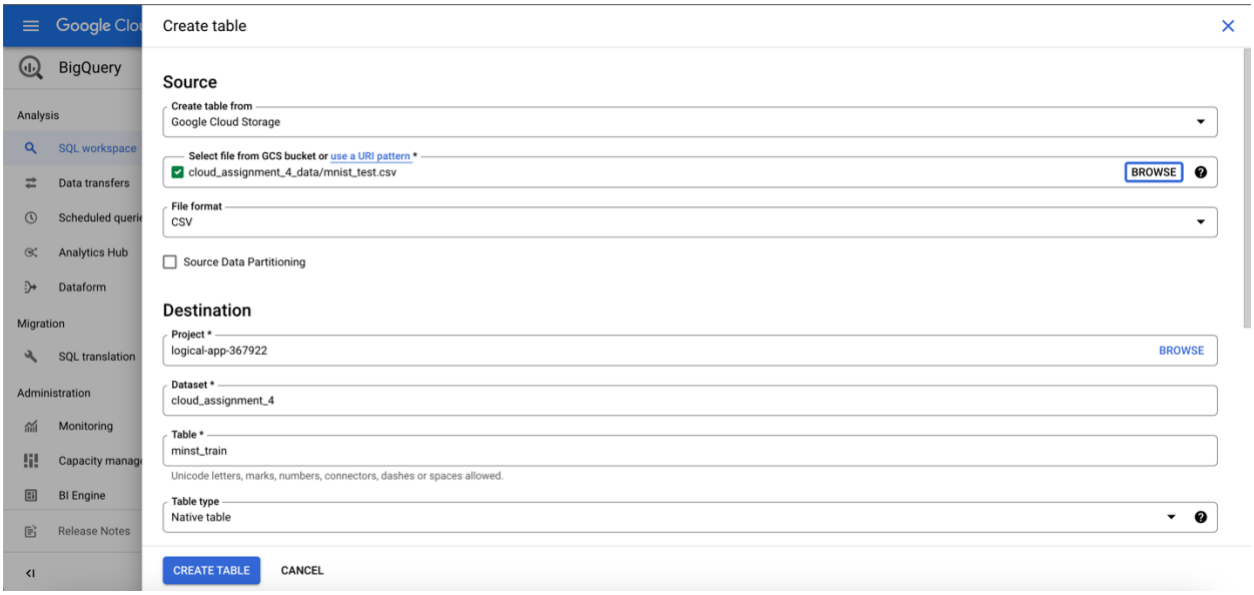


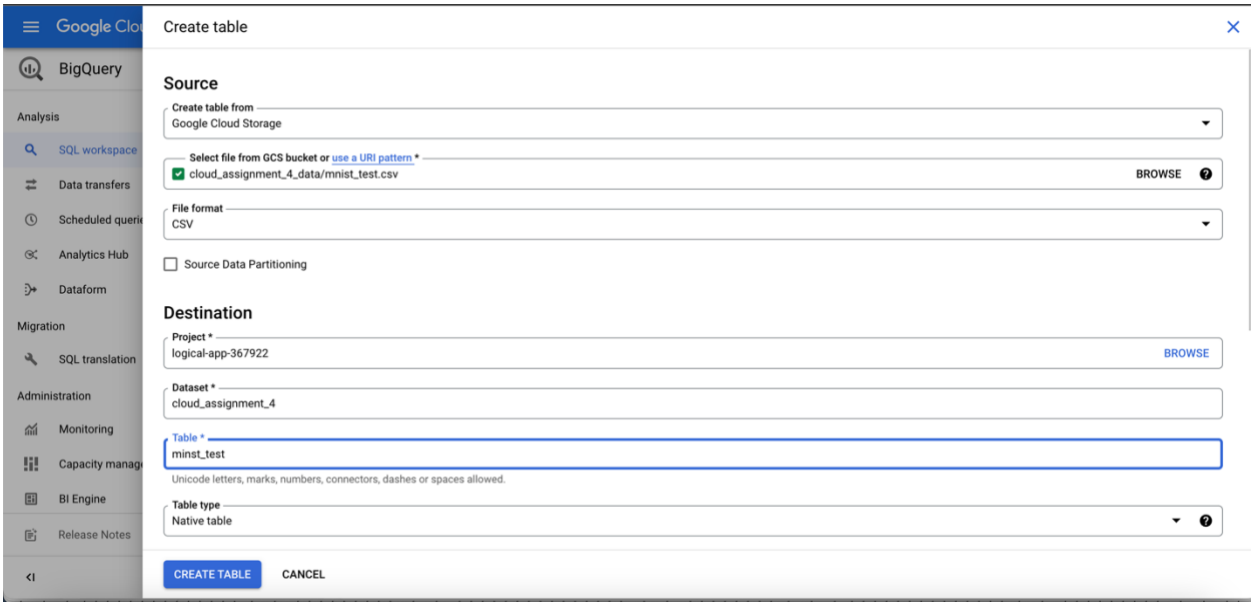


2. For database we are using Big Query so go to BigQuery and create database cloud_assignment_4

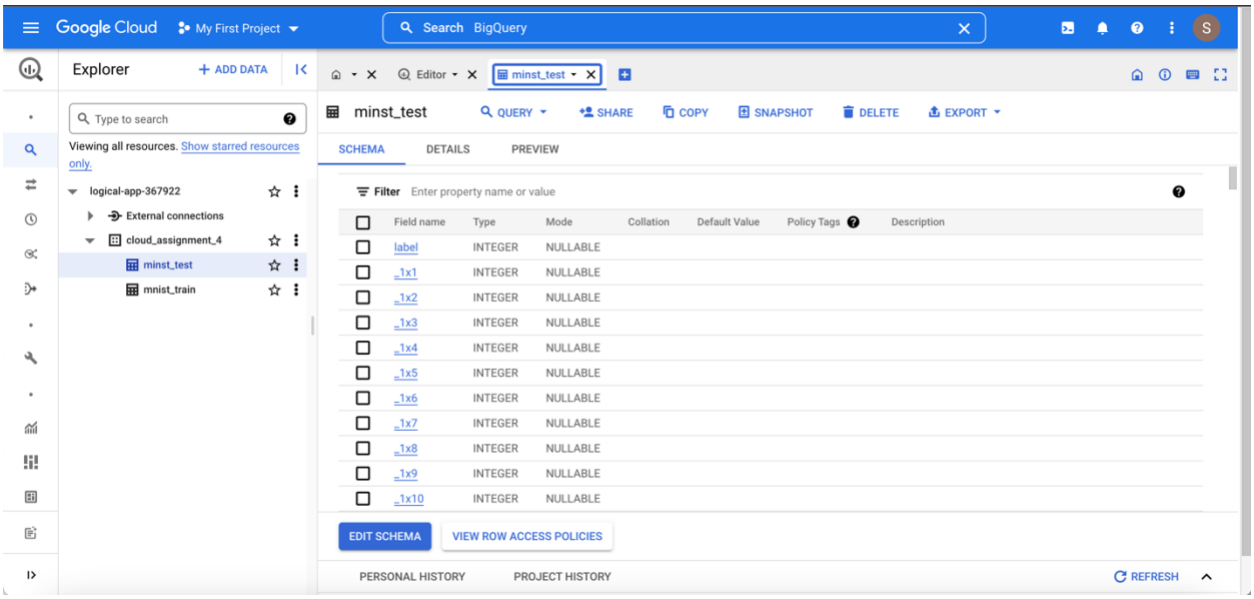


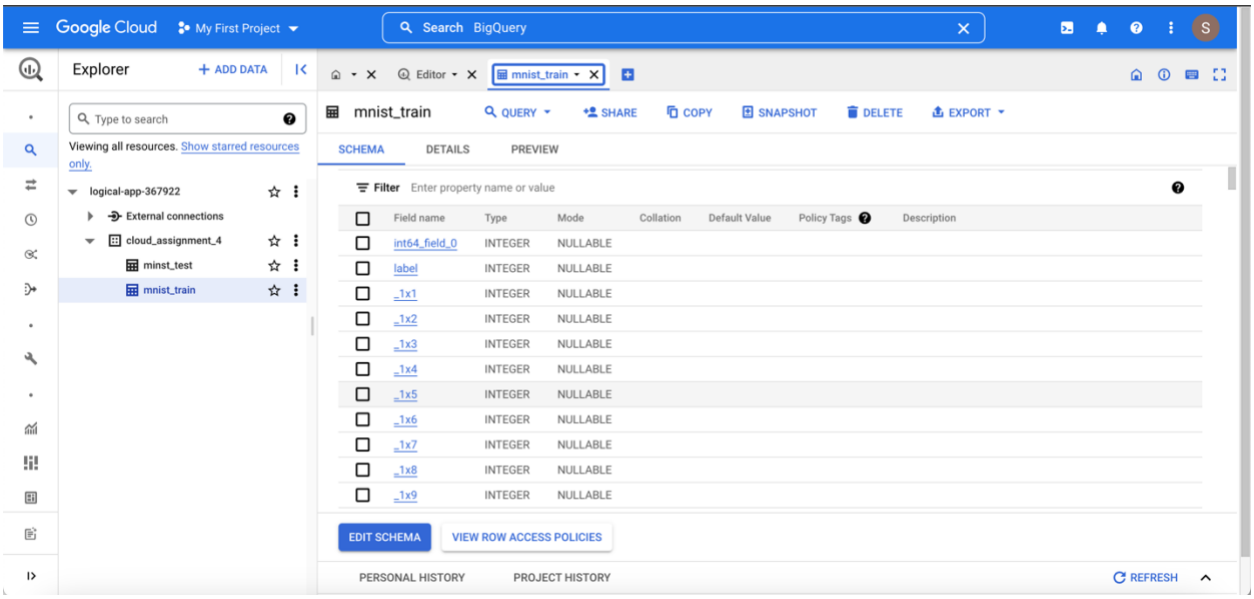
3. Create Table minst_train and minst_test and transfer dataset from cloud storage to BigQuery Database





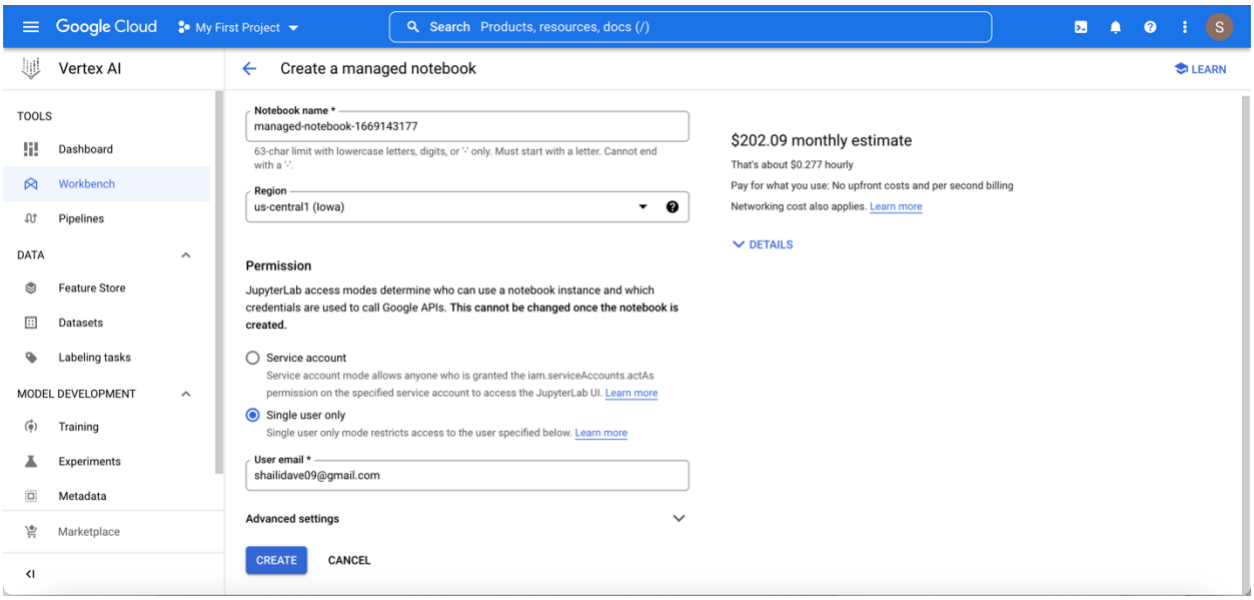
MNIST test and train Data in Database



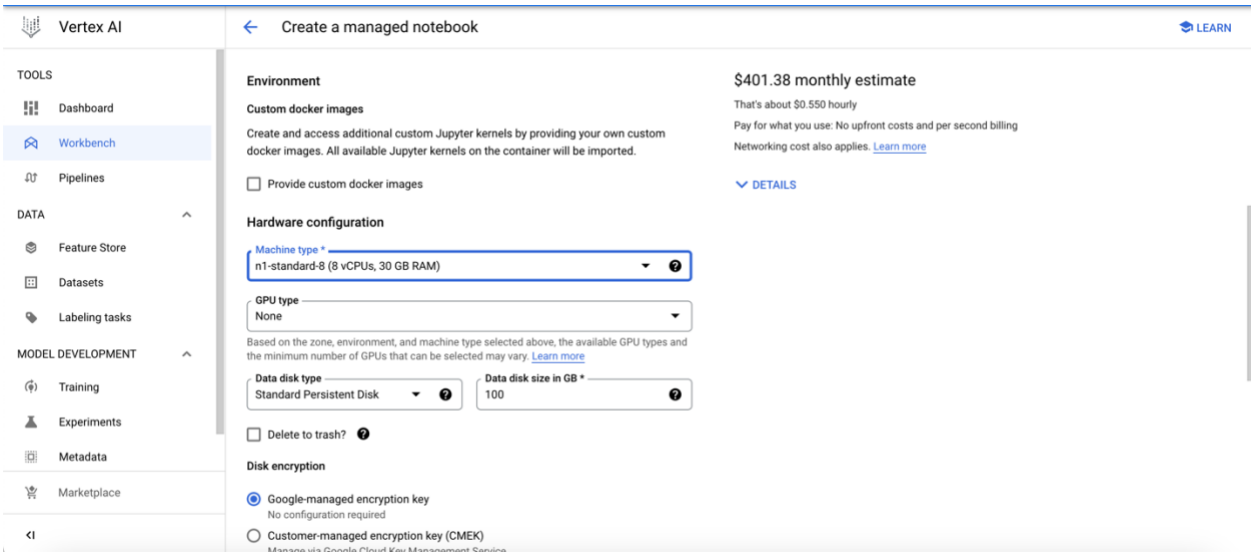


Vertex AI Backend Interacting with data

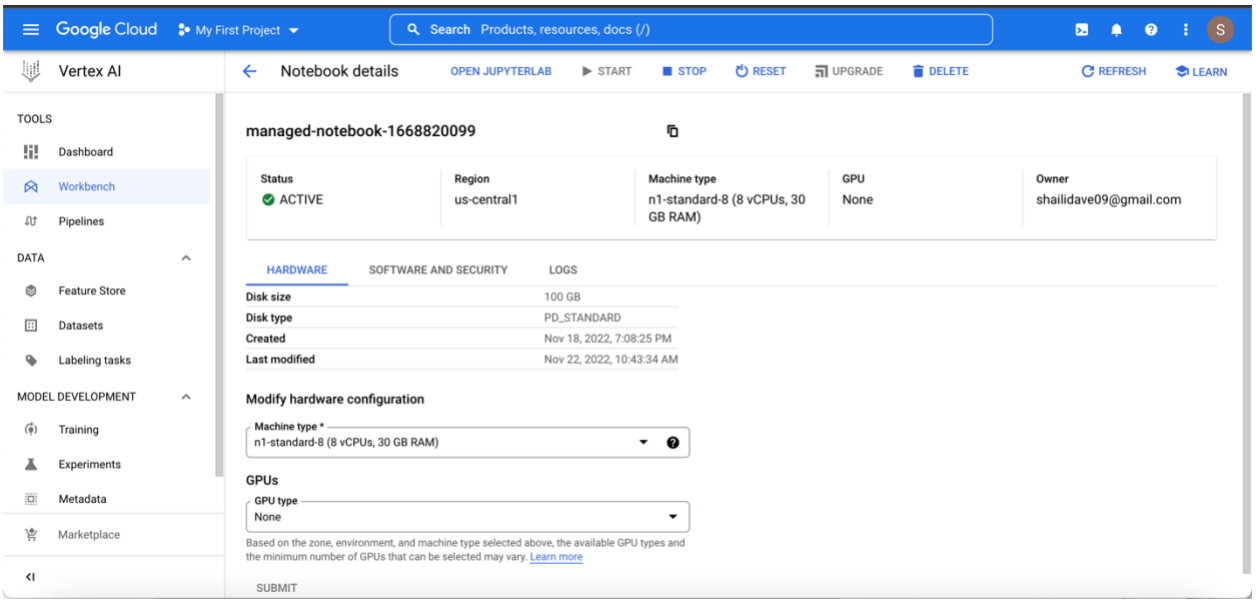
1. Go to vertex AI on Google Cloud Platform, go to Managed Notebook, and create new notebook

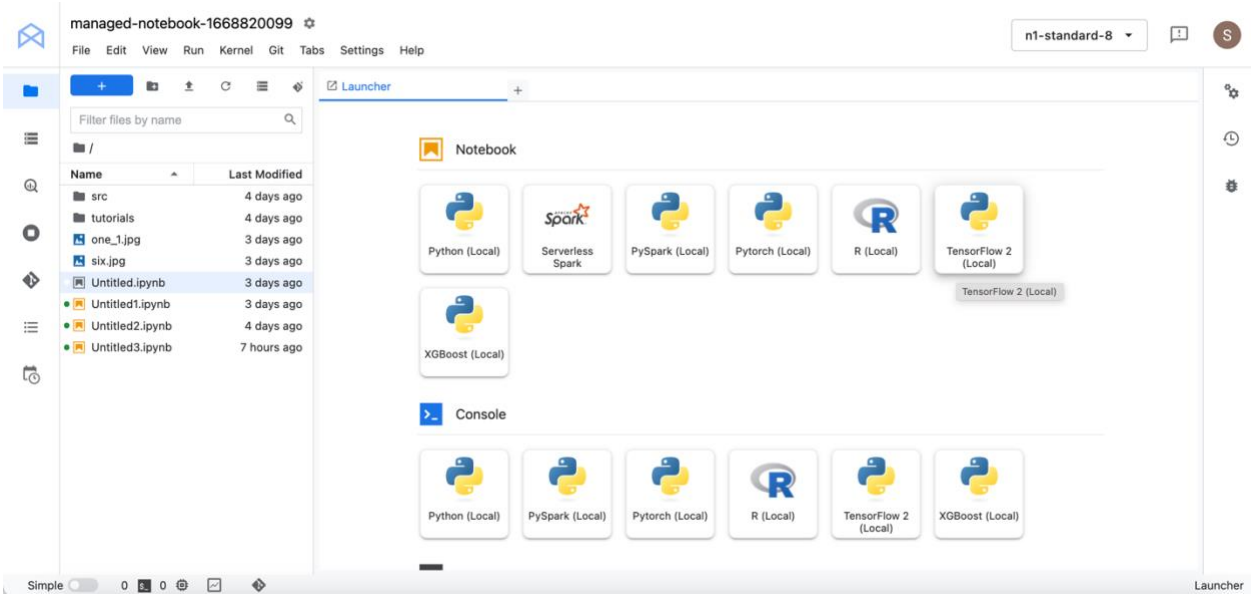


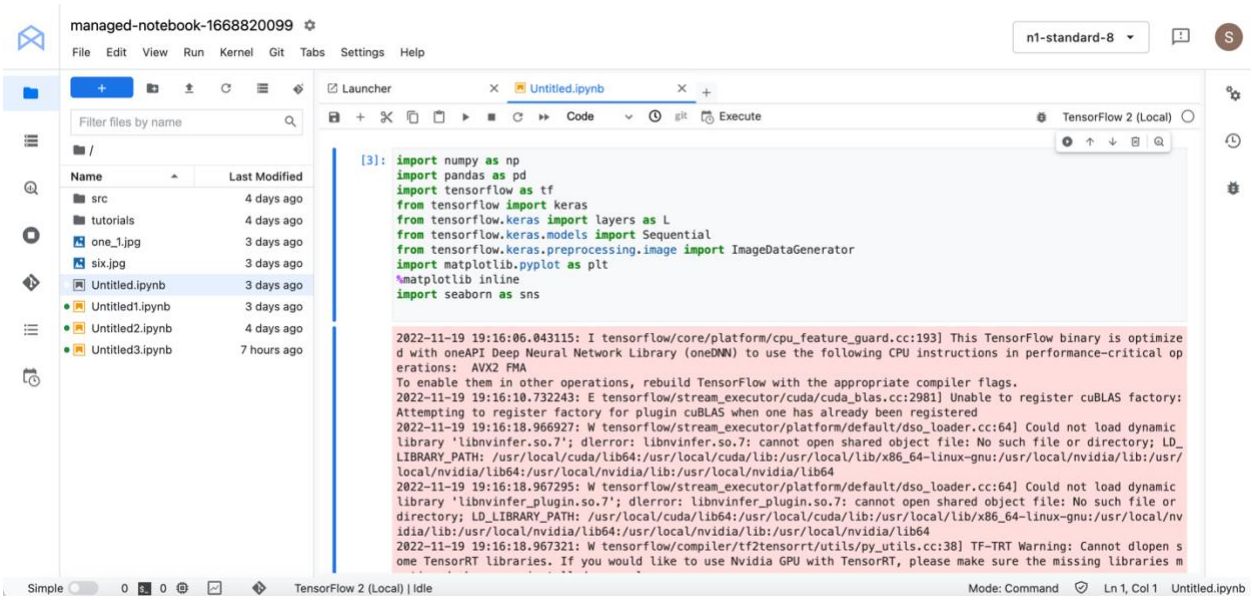
2. Assign 30GB RAM in Machine type, Hardware Configuration



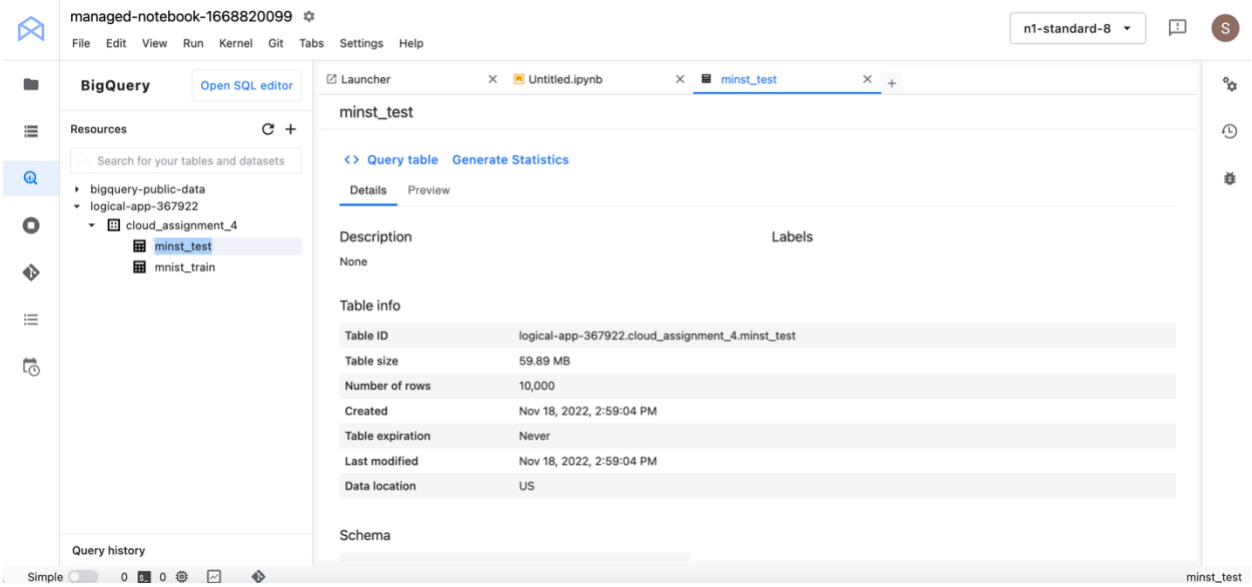
3. Activate managed notebook instance by clicking on start button as shown in screen shot and open TensorFlow Notebook



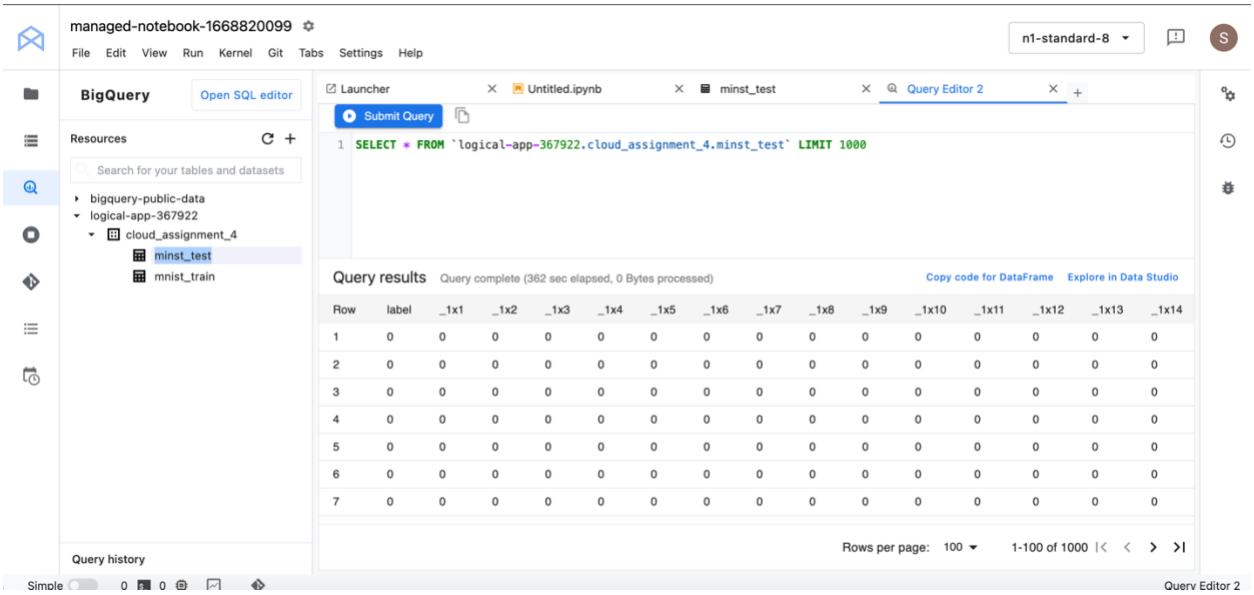




4. Vertex AI is interacted with Big Query Database, click on the icon shown in screen shoot it will load our created database cloud_assignment_4

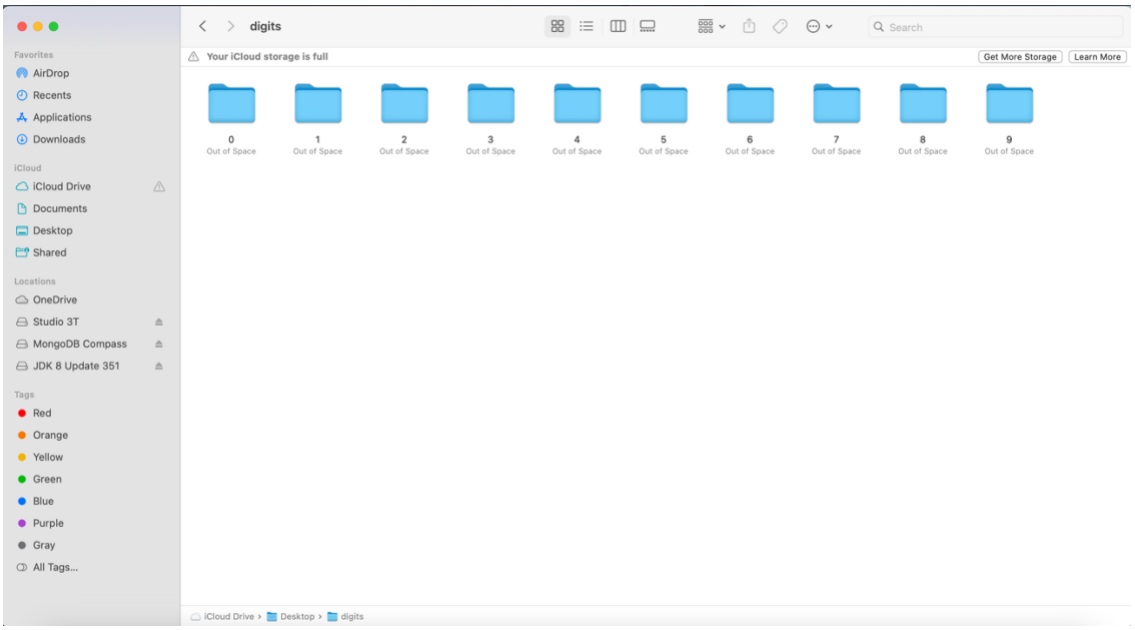


5. Click on the database, then click Submit Query and retrieve code of query to create tables in our database

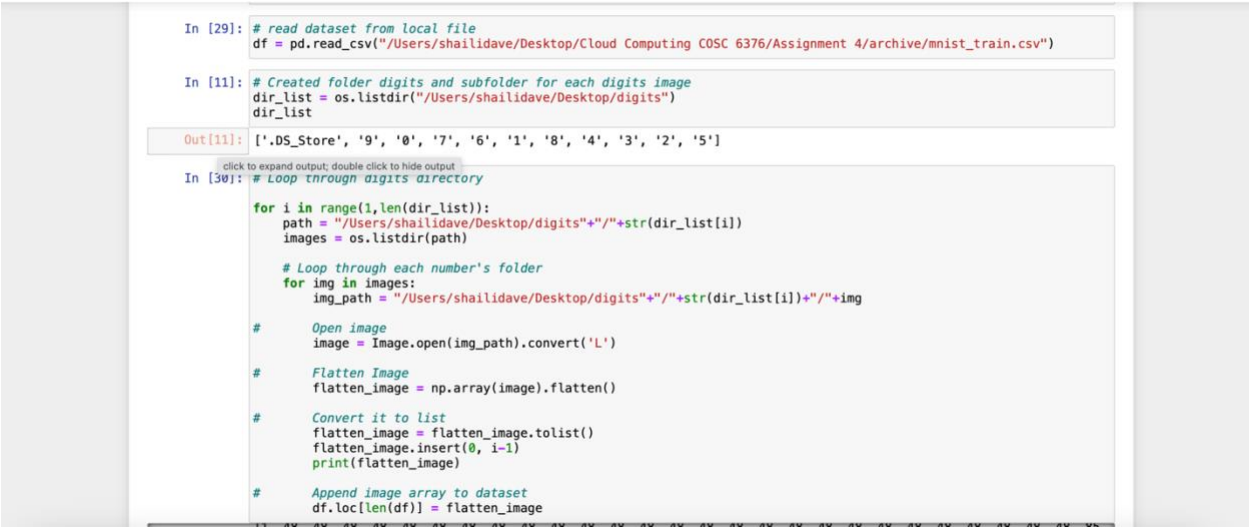


Transfer Learning on my own handwritten digits on top of existing dataset

1. Open Paint S in Mac and create 20 handwritten images of 28 * 28 pixels, each digit comprises two images
2. Create folder for each digit's image as shown in screen shot and store all this images in one folder Digits



3. Open jupyter notebook locally and create code to convert all images to grey scale and then flatten image and convert it to list



```
In [29]: # read dataset from local file
         df = pd.read_csv("/Users/shailidave/Desktop/Cloud Computing COSC 6376/Assignment 4/archive/mnist_train.csv")

In [11]: # Created folder digits and subfolder for each digits image
         dir_list = os.listdir("/Users/shailidave/Desktop/digits")
         dir_list

Out[11]: ['.DS_Store', '9', '0', '7', '6', '1', '8', '4', '3', '2', '5']

         click to expand output; double click to hide output
In [30]: # Loop through digits directory

         for i in range(1,len(dir_list)):
             path = "/Users/shailidave/Desktop/digits"+"/"+str(dir_list[i])
             images = os.listdir(path)

             # Loop through each number's folder
             for img in images:
                 img_path = "/Users/shailidave/Desktop/digits"+"/"+str(dir_list[i])+"/"+img

         #       Open image
                 image = Image.open(img_path).convert('L')

         #       Flatten Image
                 flatten_image = np.array(image).flatten()

         #       Convert it to list
                 flatten_image = flatten_image.tolist()
                 flatten_image.insert(0, i-1)
                 print(flatten_image)

         #       Append image array to dataset
                 df.loc[len(df)] = flatten_image
```

4. Append this list along with label to mnist_train.csv

```
In [31]: # Updated dataframe with hand written digits images
         df
```

Out[31]:

	label	1x1	1x2	1x3	1x4	1x5	1x6	1x7	1x8	1x9	...	28x19	28x20	28x21	28x22	28x23	28x24	28x25	28x26	28x27	28x28
0	5	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	9	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
60015	7	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
60016	8	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
60017	8	48	48	48	48	48	48	48	48	47	...	48	48	48	48	48	48	48	48	48	85
60018	9	48	48	48	48	48	48	48	48	48	...	48	48	48	48	48	48	48	48	48	85
60019	9	48	48	48	48	48	48	48	48	48	...	48	48	48	48	48	48	48	48	48	85

60020 rows × 785 columns

```
In [32]: # Save csv file
         df.to_csv("/Users/shailidave/Desktop/mnist_train.csv")
```

5. Upload updated mnist_train.csv to cloud_assignment_4 google cloud bucket
6. Also, create new table in Big Query Database named mnist_train and remove the old one then read dataset in vertex AI jupyter notebook

```
[12]: # Read dataset from bigquery
      from google.cloud.bigquery import Client, QueryJobConfig
      client = Client()
```

```
[46]: query1 = """SELECT * FROM `logical-app-367922.cloud_assignment_4.minst_test`"""
      job1 = client.query(query1)
      test = job1.to_dataframe()
      test
```

[46]:

	label	_1x1	_1x2	_1x3	_1x4	_1x5	_1x6	_1x7	_1x8	_1x9	...	_28x19	_28x20	_28x21	_28x22	_28x23	_28x24	_28x
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	

7. Now we data read from database, remove labels from dataset and reshape training array as shown in screenshot

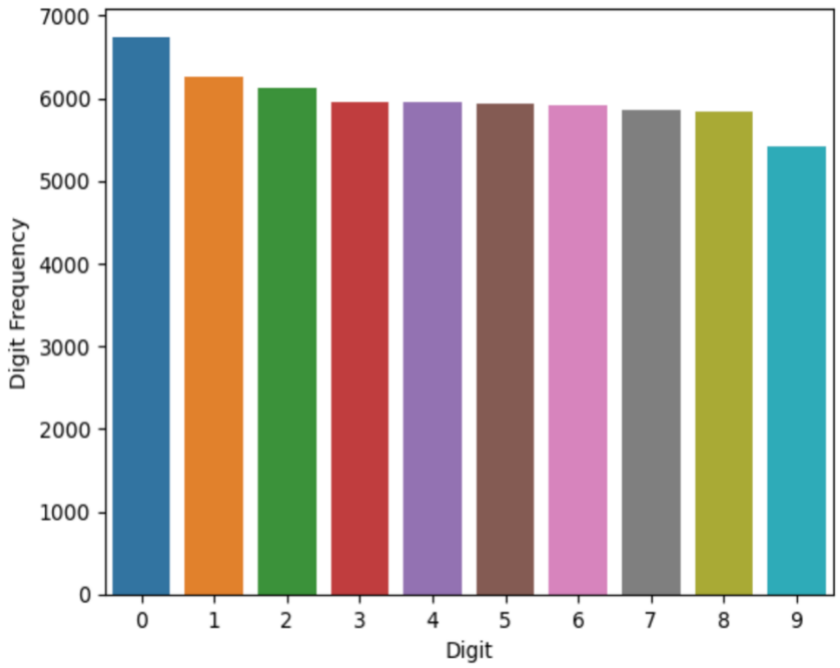
```
[16]: train_arr = np.array(train).reshape(-1, 28, 28, 1)
```

```
[17]: train_arr.shape
```

```
[17]: (60020, 28, 28, 1)
```

Data Visualization

8. Create bar plot to visualize frequency of each digit



Data Preprocessing

9. Converting array to image, resize image, convert it to RGB and converting it back to array

[19]:
```
from tensorflow.keras.preprocessing.image import load_img, img_to_array, array_to_img

def change_size(image):
    img = array_to_img(image, scale=False) #returns PIL Image
    img = img.resize((75, 75)) #resize image
    img = img.convert(mode='RGB') #makes 3 channels
    arr = img_to_array(img) #convert back to array
    return arr.astype(np.float64)
```

[20]:
```
train_arr_75 = [change_size(img) for img in train_arr]
del train_arr
train_arr_75 = np.array(train_arr_75)
train_arr_75.shape
```

[20]:
```
(60020, 75, 75, 3)
```

Data Transformation

10. Transform training and validation data using ImageDataGenerator

[21]:
```
image_gen = ImageDataGenerator(rescale=1./255, #easier for network to interpret numbers in range [0,1]
                               zoom_range=0.1,
                               width_shift_range=0.2,
                               height_shift_range=0.2,
                               validation_split=0.2) # 80/20 train/val split

train_generator = image_gen.flow(train_arr_75,
                                 y,
                                 batch_size=32,
                                 shuffle=True,
                                 subset='training',
                                 seed=42)
valid_generator = image_gen.flow(train_arr_75,
                                 y,
                                 batch_size=16,
                                 shuffle=True,
                                 subset='validation')
del train_arr_75 #saves RAM
```

Modelling

11. Add following layers to Sequential model and compile model

[23]:
```
model = Sequential()

model.add(tf.keras.applications.resnet50.ResNet50(input_shape = (75, 75, 3),
                                 include_top = False,
                                 weights = 'imagenet'))

model.add(L.Flatten())
model.add(L.Dense(128, activation='relu'))
model.add(L.Dense(10, activation='softmax'))

model.compile(optimizer=keras.optimizers.Adam(lr=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
#Do not use default learning rate since it is too high!
```

12. Train model

```
[25]: history = model.fit(train_generator, validation_data=valid_generator, epochs=5,
                          steps_per_epoch=train_generator.n//train_generator.batch_size,
                         validation_steps=valid_generator.n//valid_generator.batch_size)
```

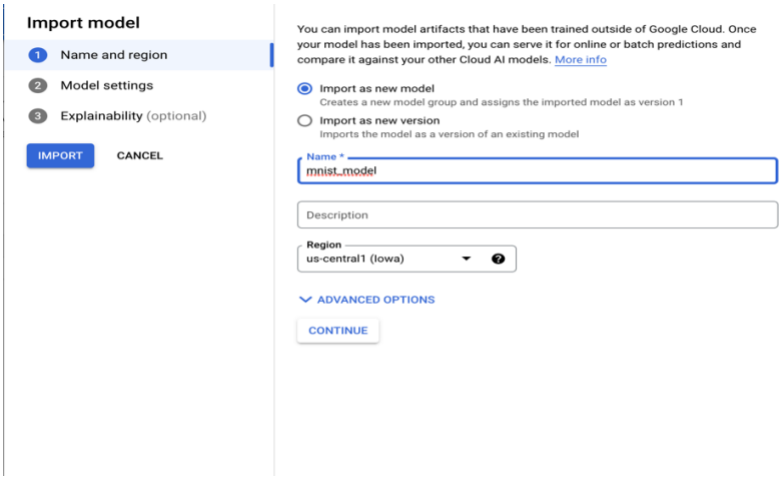
```
Epoch 1/5
1500/1500 [==============================] - 2051s 1s/step - loss: 0.1566 - accuracy: 0.9540 - val_loss: 0.0722 - v
al_accuracy: 0.9801
Epoch 2/5
1500/1500 [==============================] - 1889s 1s/step - loss: 0.0842 - accuracy: 0.9781 - val_loss: 0.2151 - v
al_accuracy: 0.9745
Epoch 3/5
1500/1500 [==============================] - 1924s 1s/step - loss: 0.0512 - accuracy: 0.9857 - val_loss: 0.0289 - v
al_accuracy: 0.9925
Epoch 4/5
1500/1500 [==============================] - 1980s 1s/step - loss: 0.0419 - accuracy: 0.9885 - val_loss: 0.0346 - v
al_accuracy: 0.9913
Epoch 5/5
1500/1500 [==============================] - 1945s 1s/step - loss: 0.0470 - accuracy: 0.9876 - val_loss: 0.0412 - v
al_accuracy: 0.9908
```

13. Save trained model to our cloud bucket (cloud_assignment_model) which should be in same region as our notebook

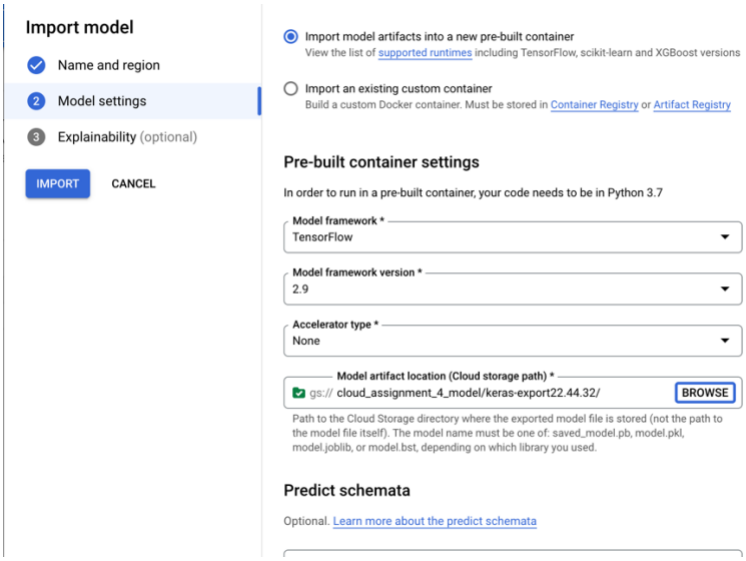
```
[29]: from datetime import datetime
now = datetime.now()
current_time = now.strftime("%H.%M.%S")
tf.compat.v1.keras.experimental.export_saved_model(model,'gs://cloud_assignment_4_model/keras-export'+current_time)
```

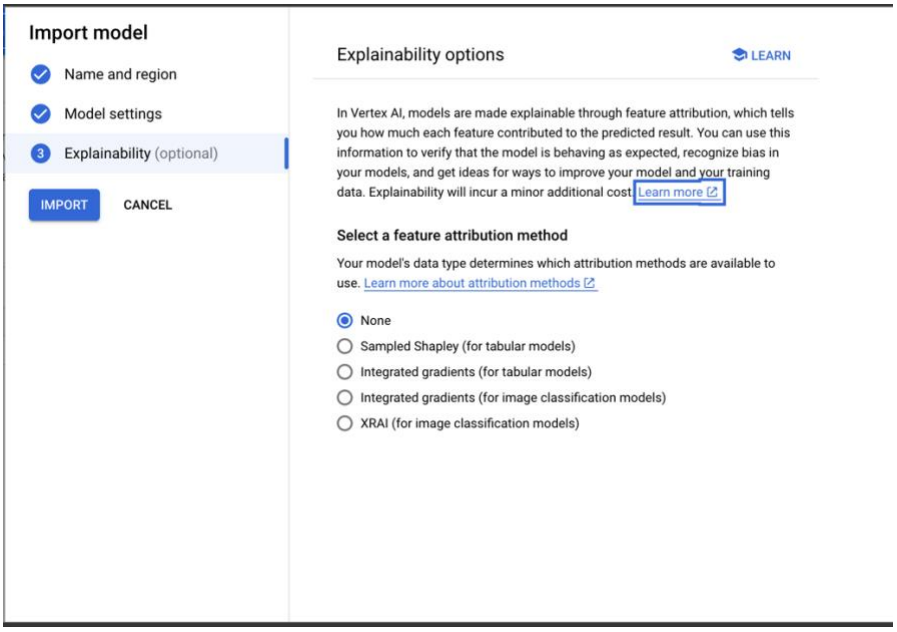
Create Endpoint and Deploy Model

14. On Vertex AI, go to model registry then click import and give name to model region should be same as cloud bucket for us central 1

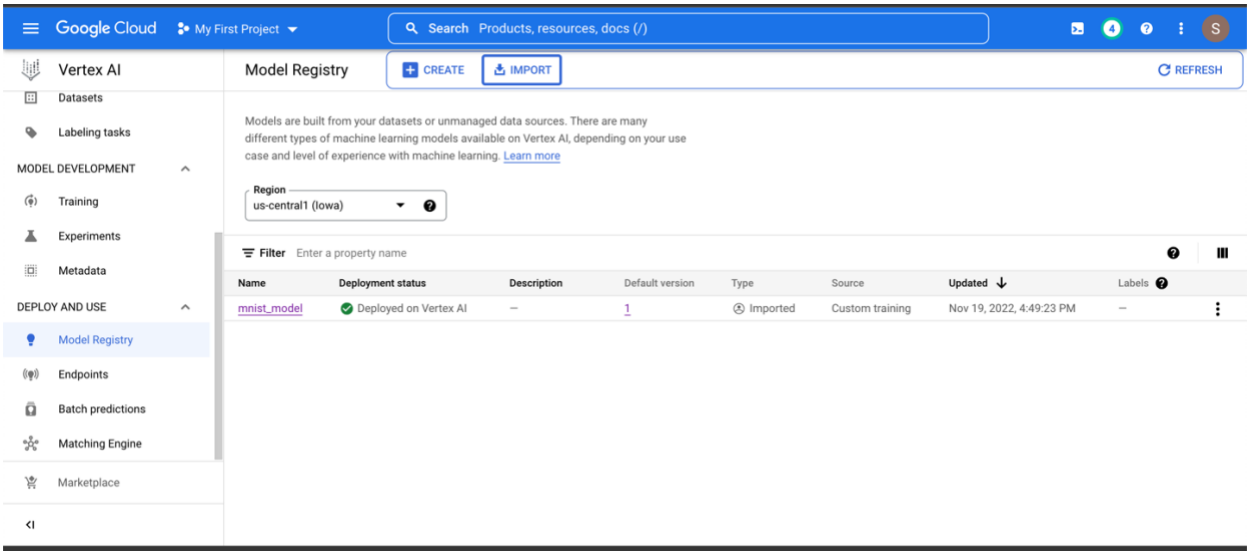


15. Select TensorFlow version and path to cloud storage directory where model is stored then click import

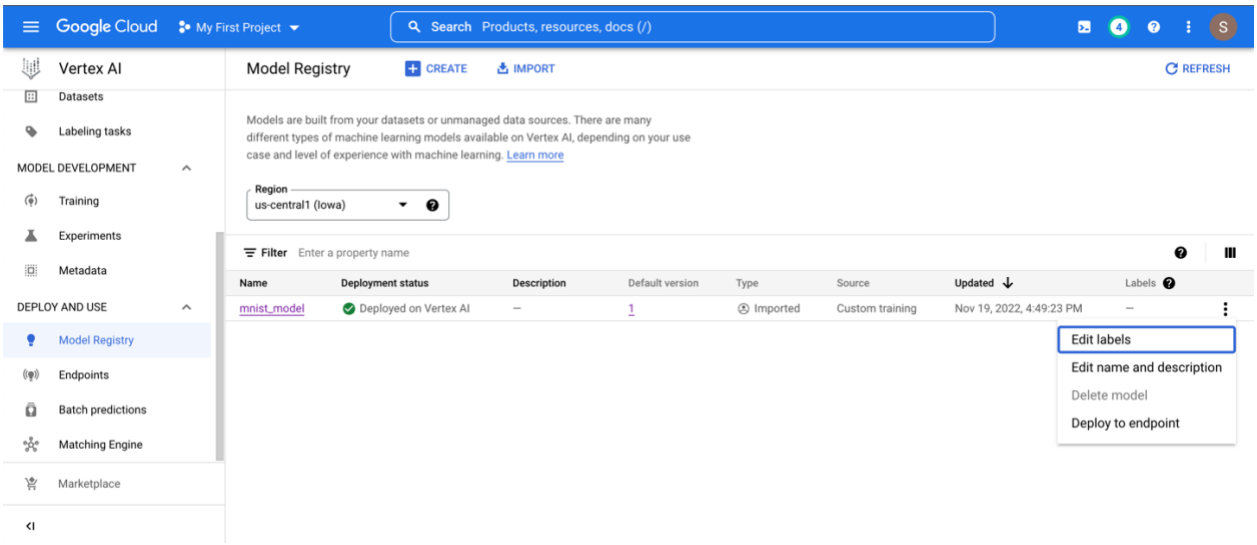




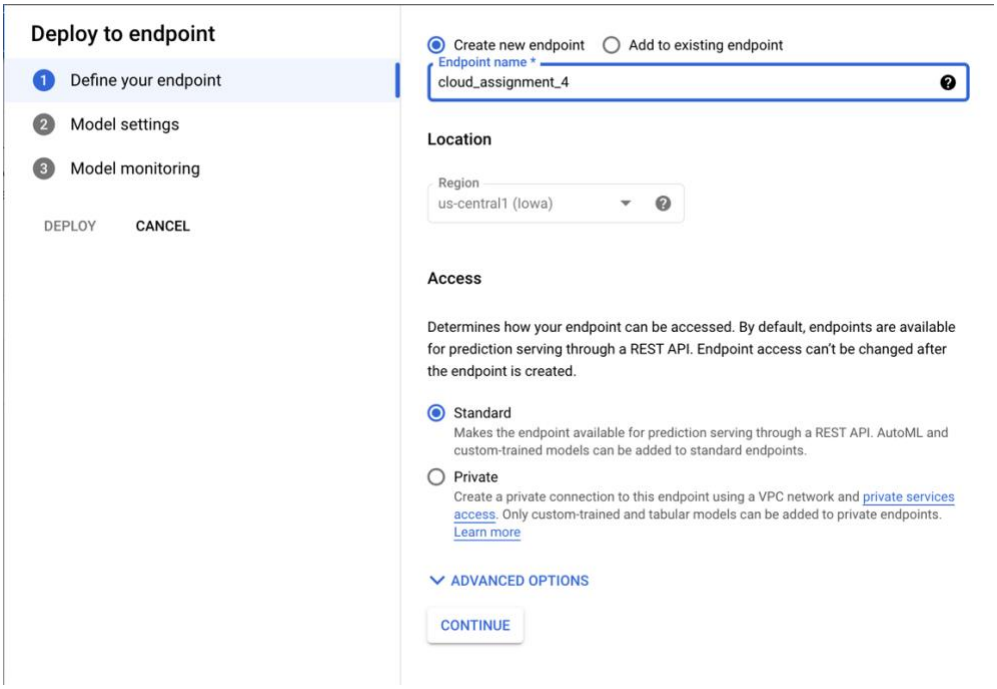
Model Deployed



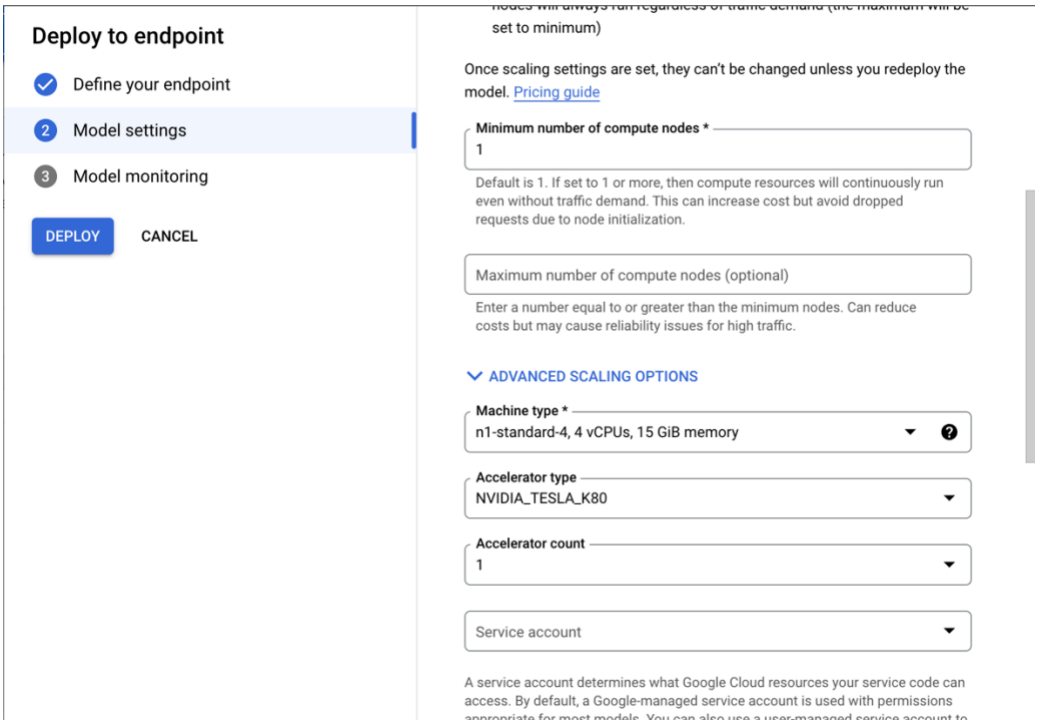
16. Once the model is deployed, click on model, and create new endpoint



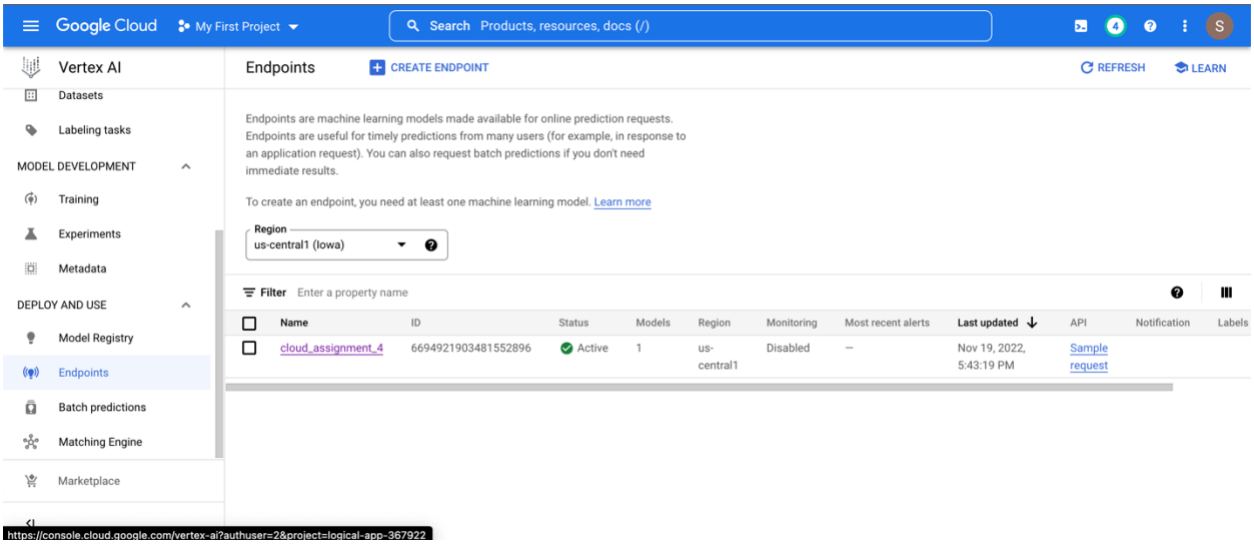
17. Give name to endpoint



18. Select Machine type from standard with 15 GB ram, keep everything default and click on deploy

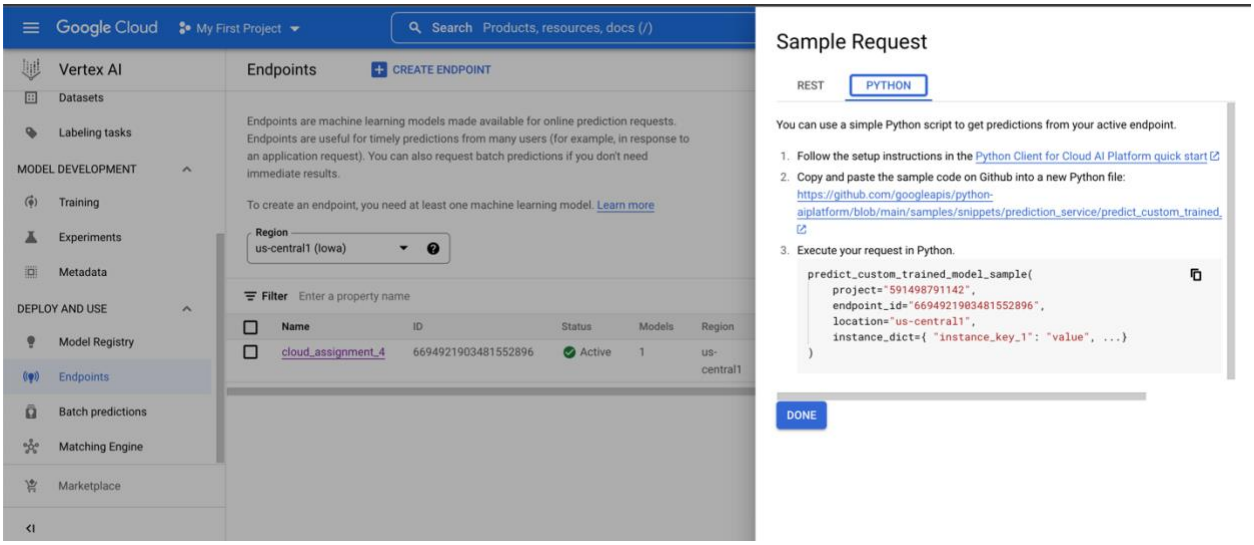


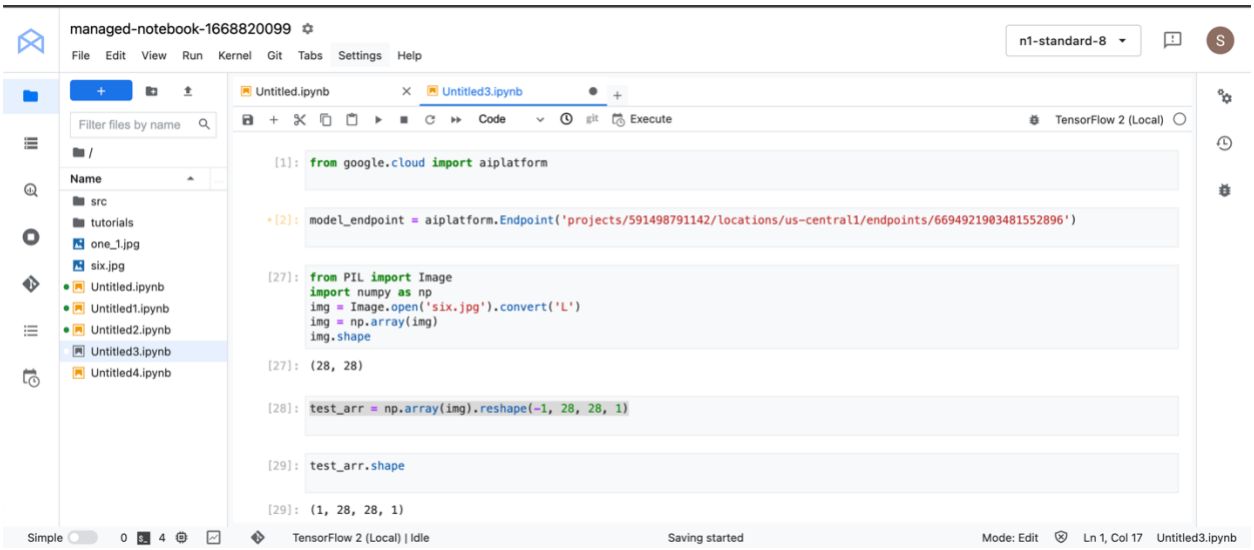
Deployed Endpoint cloud_assignment_4



Predict From Endpoint

19. Create new jupyter notebook, retrieve project id and endpoint id from sample request then specify endpoint link

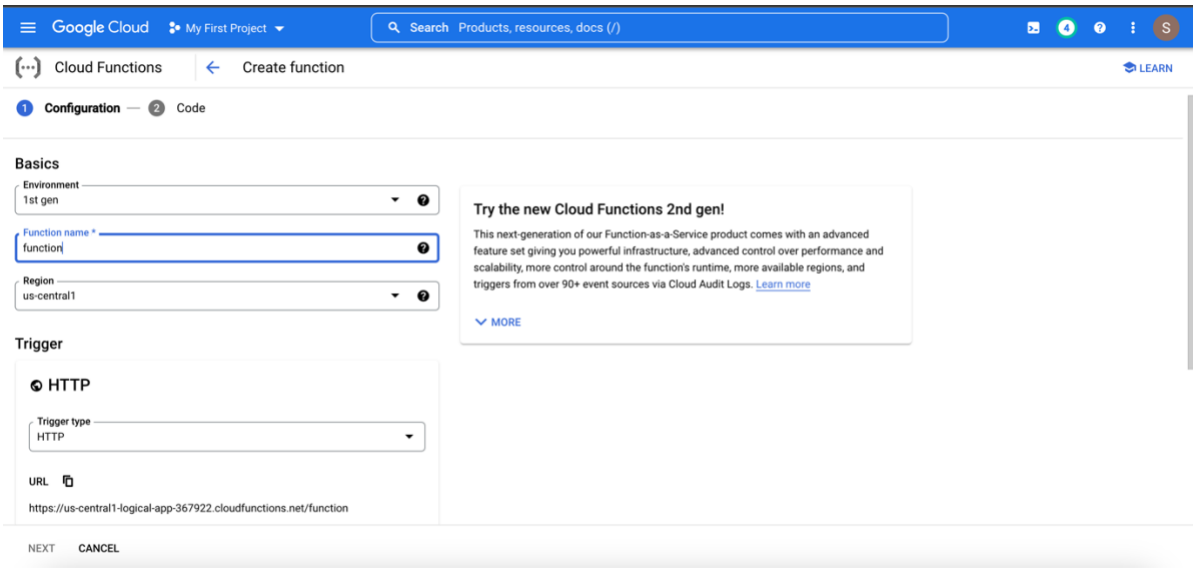


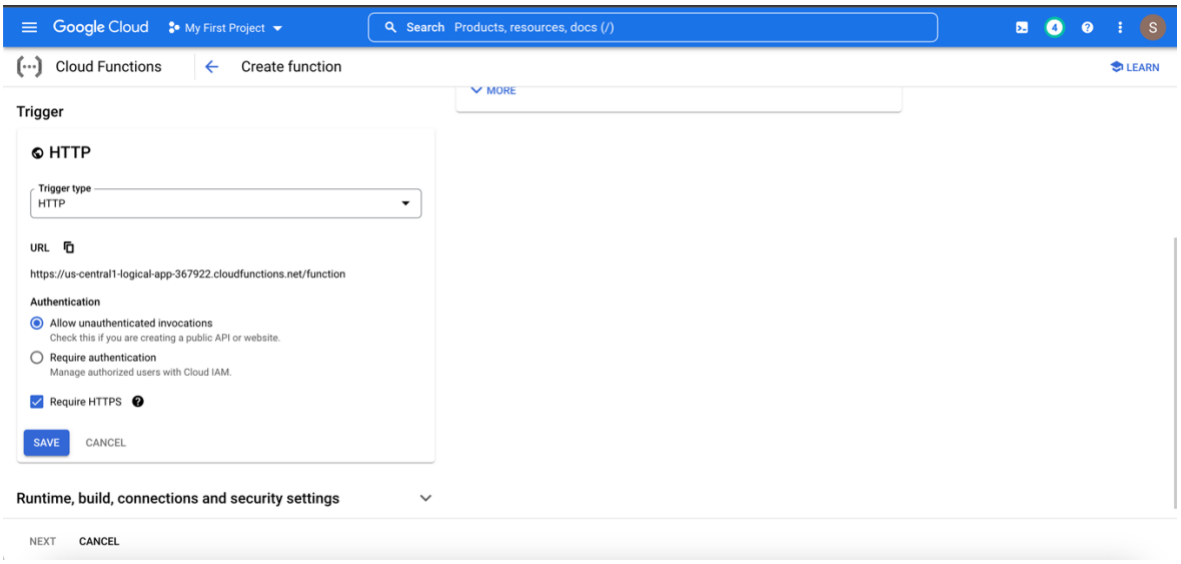


20. Upload handwritten image to workbench, do above mentioned preprocessing steps and give image to model to predict using endpoint

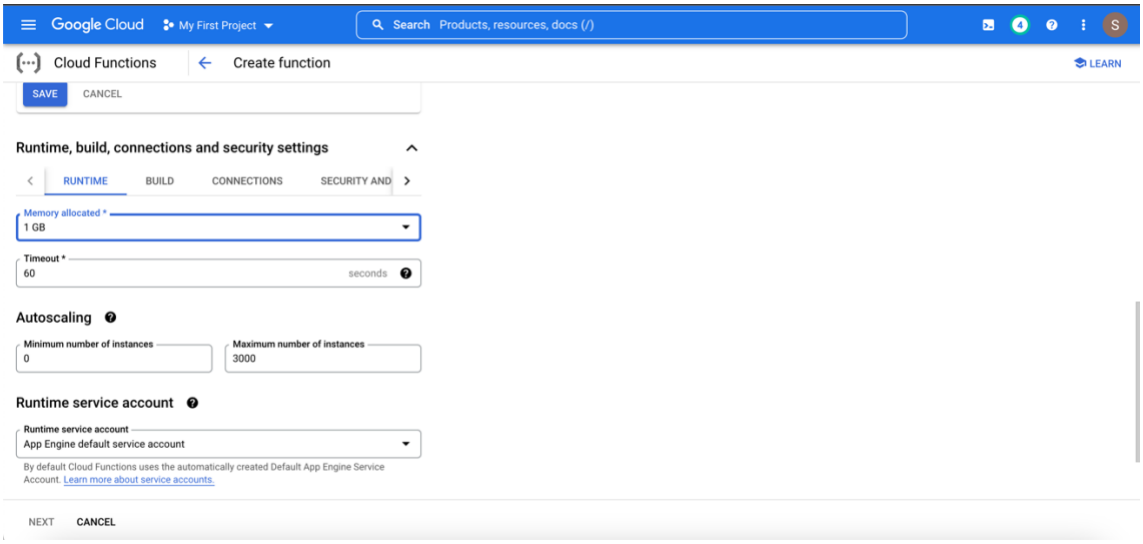
Predict using Cloud Function Serverless API

21. On cloud functions, create new function -> Give function name and make sure it is in same region as cloud bucket and for authentication select allow unauthenticated invocations

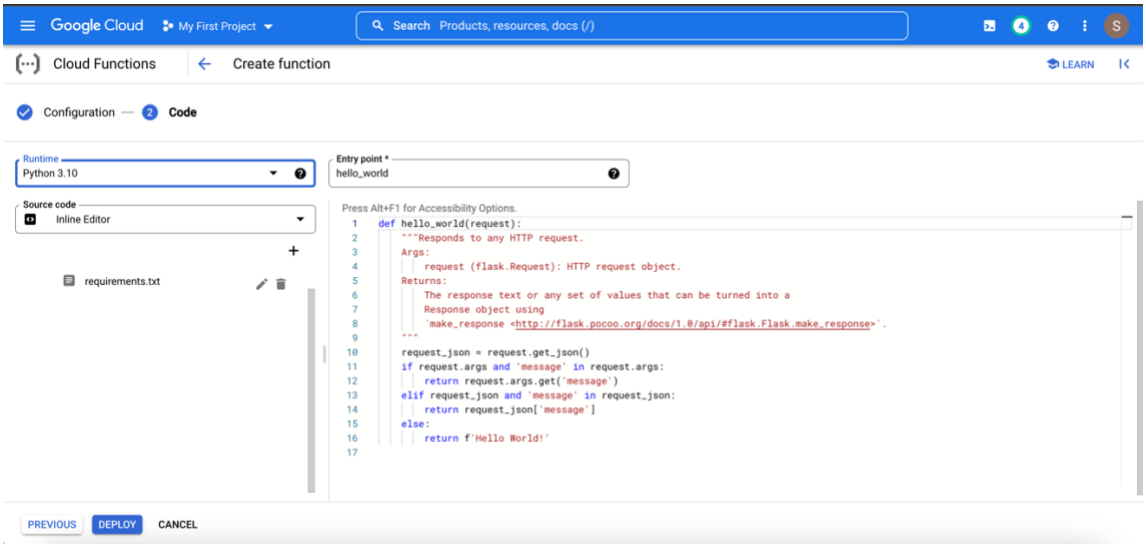




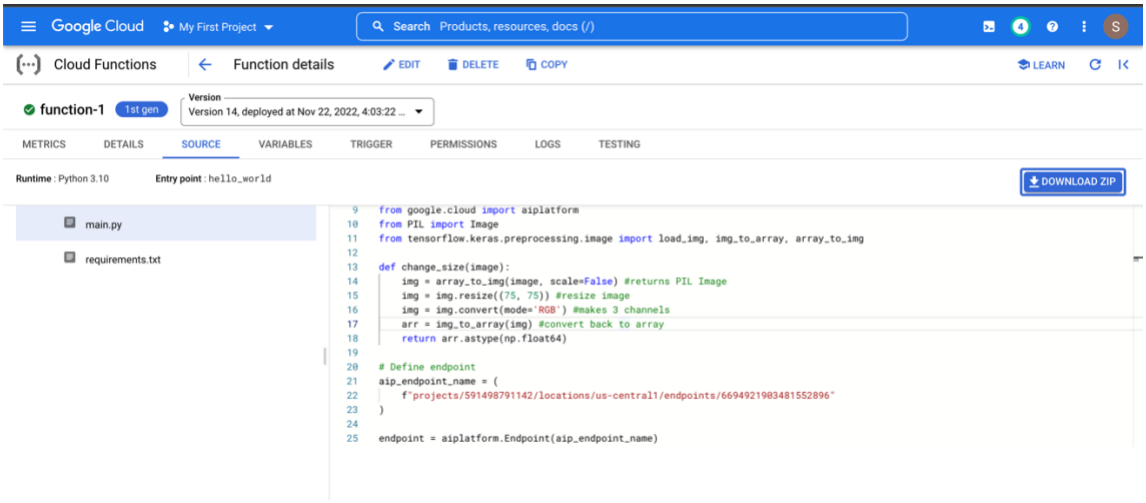
22. Allocate 1 GB memory from Runtime

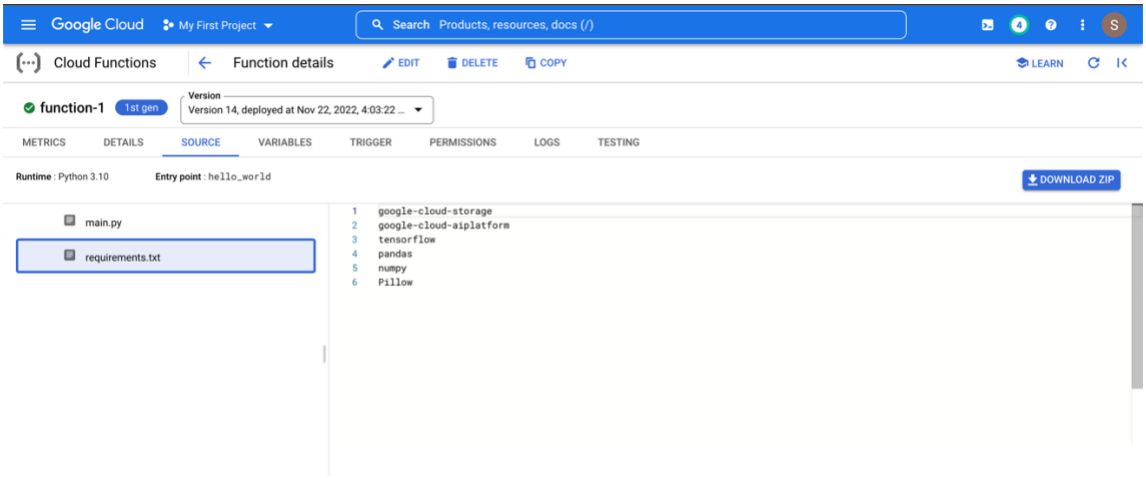


23. Select Python 3.10 for runtime and give entry point function name which will be triggered when we will hit our API



24. Mention required packages in requirements.txt

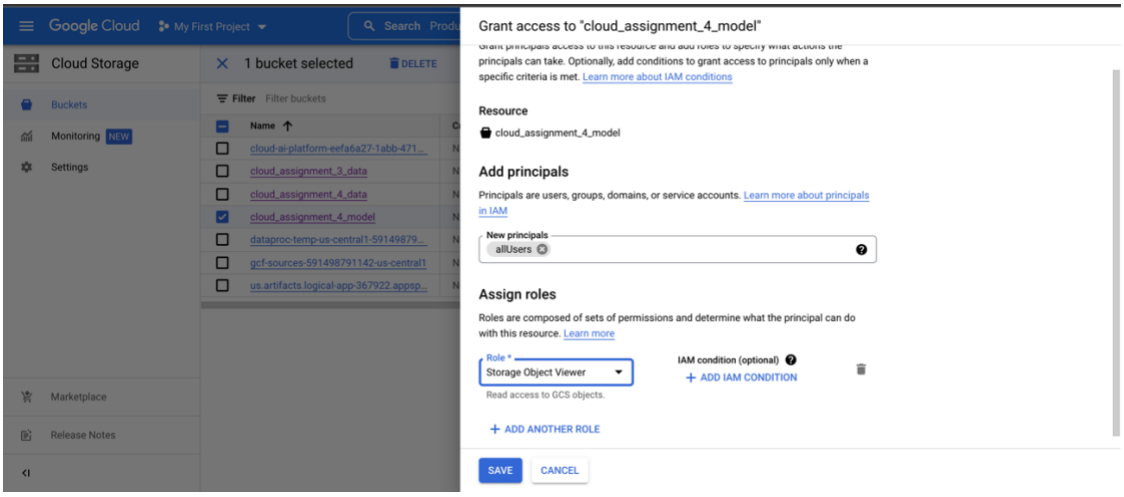




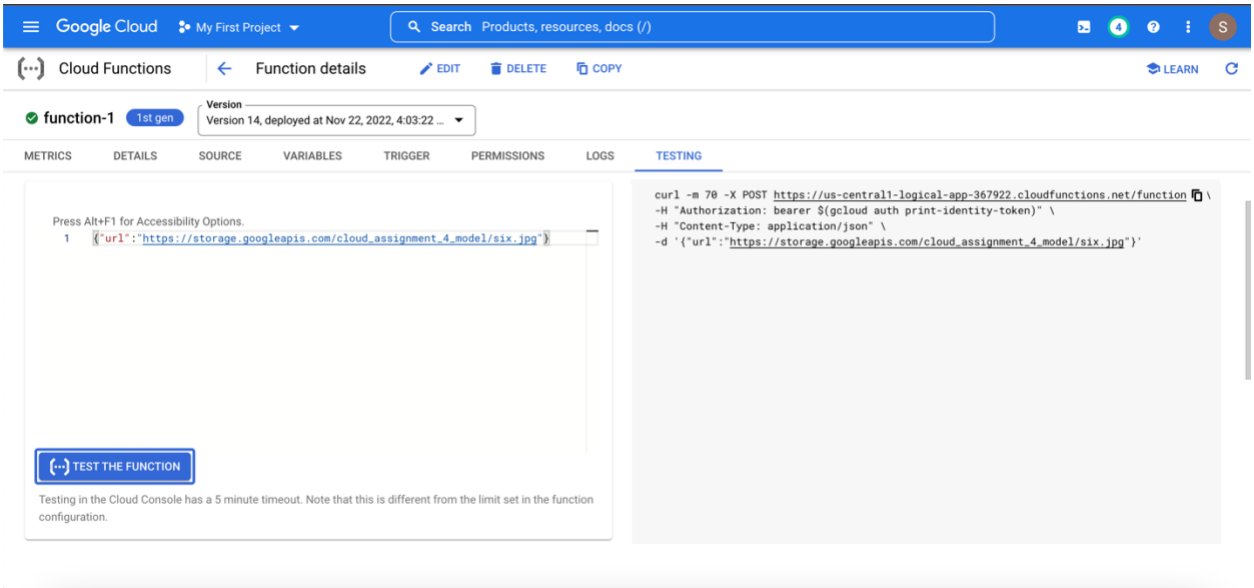
25. Finally Deploy Cloud Function

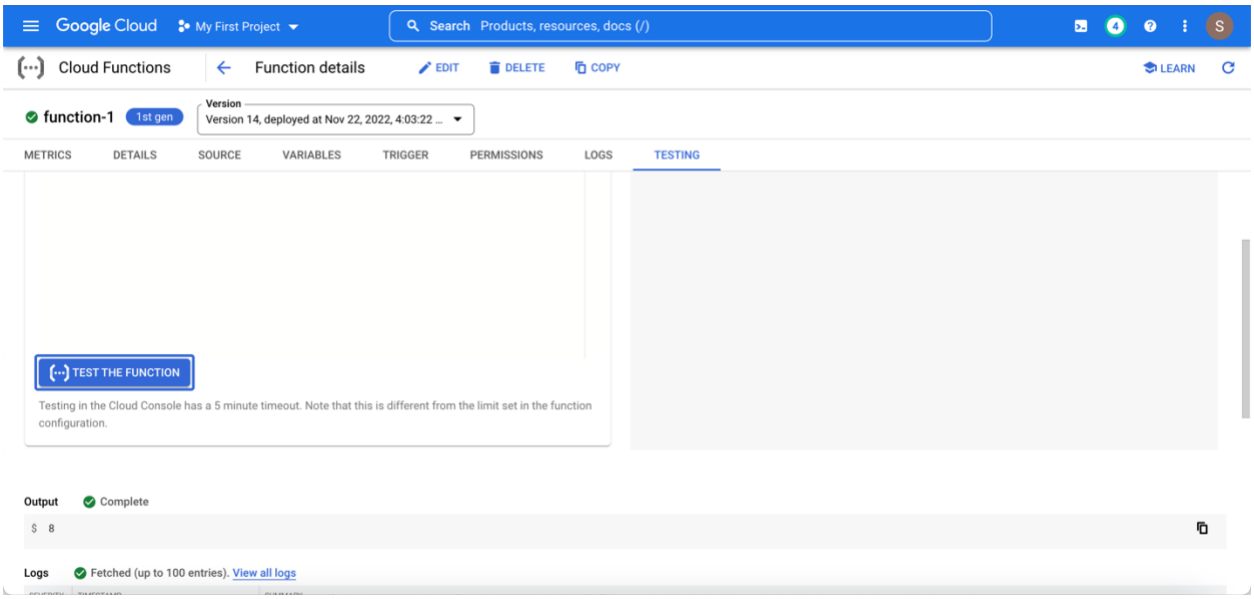
Testing

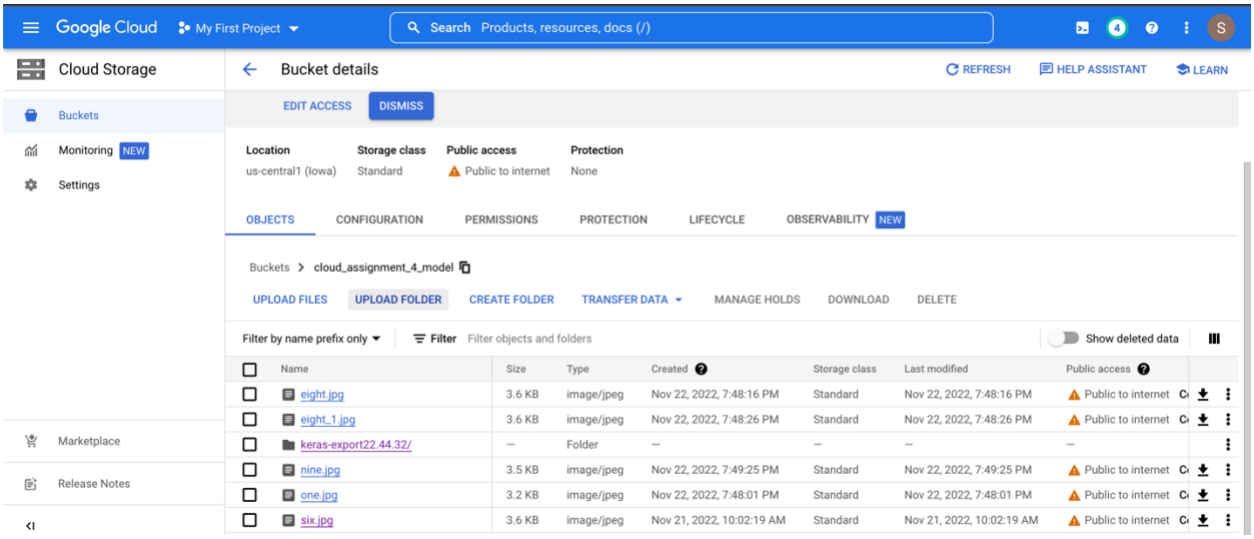
26. For testing provide link of image in cloud bucket, for that I have made cloud bucket public
27. To make cloud bucket public, select bucket -> click on Permissions -> Add Principals -> Add allUsers -> Role -> Add Storage Object Viewer
28. There are **5 images** in bucket, anyone can be used for testing



Go to test and run API, provide url of image in cloud bucket



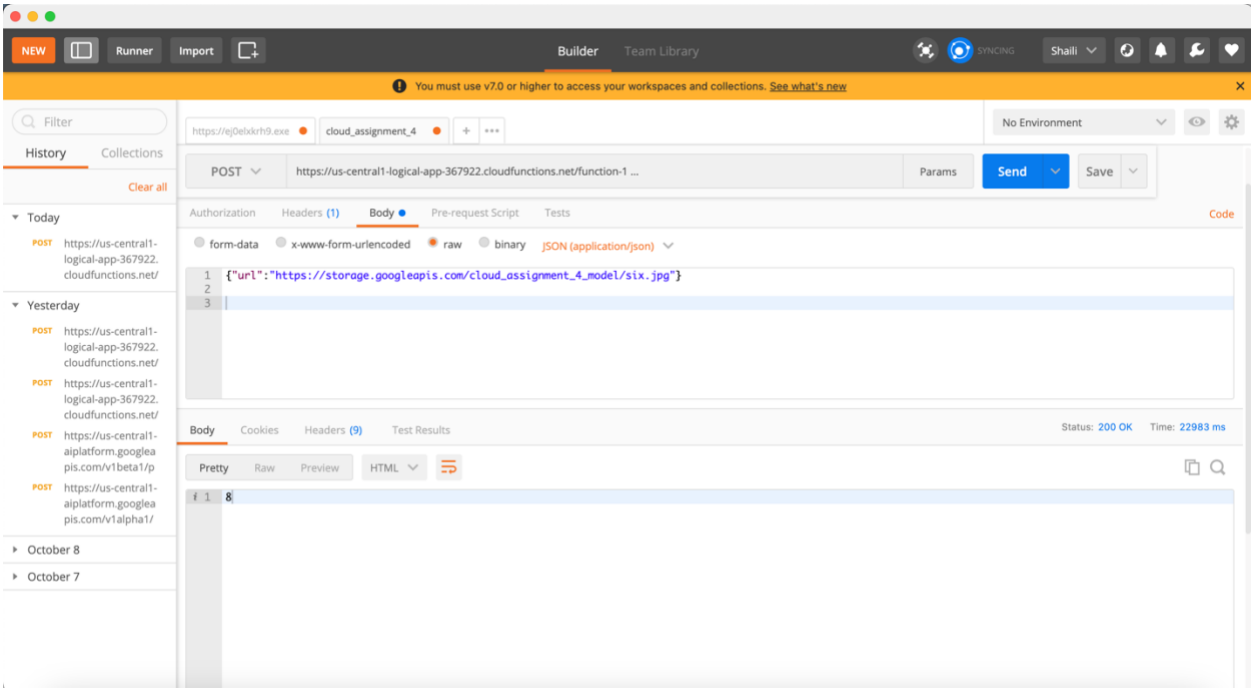




Cloud bucket link:
https://console.cloud.google.com/storage/browser/cloud_assignment_4_model;tab=objects?forceOnBucketsSortingFiltering=false&authuser=0&project=logical-app-367922&prefix=&forceOnObjectsSortingFiltering=false

Testing Using Postman

Link: https://us-central1-logical-app-367922.cloudfunctions.net/function-1
json input
{"url":"https://storage.googleapis.com/cloud_assignment_4_model/six.jpg"}

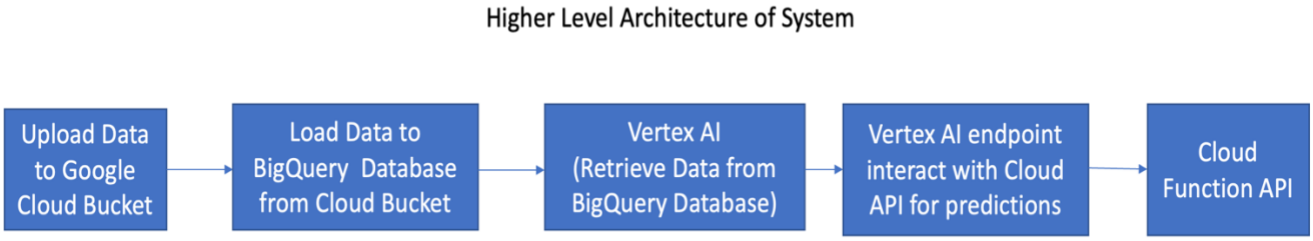


Testing using Curl request

curl -m 70 -X POST https://us-central1-logical-app-367922.cloudfunctions.net/function-1 \
-H "Authorization: bearer $(gcloud auth print-identity-token)" \
-H "Content-Type: application/json" \
-d '{"url":"https://storage.googleapis.com/cloud_assignment_4_model/six.jpg"}
'

```
(base) shailidave@Shailis-MacBook-Air ~ % curl -m 70 -X POST https://us-central1-logical-app-367922.cloudfunctions.net/function-1 \
-H "Authorization: bearer $(gcloud auth print-identity-token)" \
-H "Content-Type: application/json" \
-d '{"url":"https://storage.googleapis.com/cloud_assignment_4_model/six.jpg"}'
zsh: command not found: gcloud
8%
(base) shailidave@Shailis-MacBook-Air ~ % curl -m 70 -X POST https://us-central1-logical-app-367922.cloudfunctions.net/function-1 \
-H "Content-Type: application/json" \
-d '{"url":"https://storage.googleapis.com/cloud_assignment_4_model/six.jpg"}'
8%
```

Architecture Diagram



Limitations

⇒ Model requires improvement as current model does not give correct predictions every time

References

[1] https://www.youtube.com/watch?v=-9fU1xwBQYU&ab_channel=GoogleCloudTech
[2] https://stackoverflow.com/questions/73141754/vertex-ai-endpoint-call-with-json-invalid-json-payload-received
[3] https://cloud.google.com/vertex-ai/docs/tutorials/image-recognition-custom/serving
[4] https://towardsdatascience.com/machine-learning-model-as-a-serverless-endpoint-using-google-cloud-function-a5ad1080a59e
[5] https://cloud.google.com/bigquery/docs/introduction
[6] https://stackoverflow.com/questions/7391945/how-do-i-read-image-data-from-a-url-in-python
[7] https://stackoverflow.com/questions/70773518/fail-to-install-module-pil-error-could-not-find-a-version-that-satisfies-the
[8] https://cloud.google.com/functions/docs/writing
[9] https://www.kaggle.com/code/saumandas/intro-to-transfer-learning-with-mnist
[10] https://www.analyticsvidhya.com/blog/2021/05/transfer-learning-using-mnist/