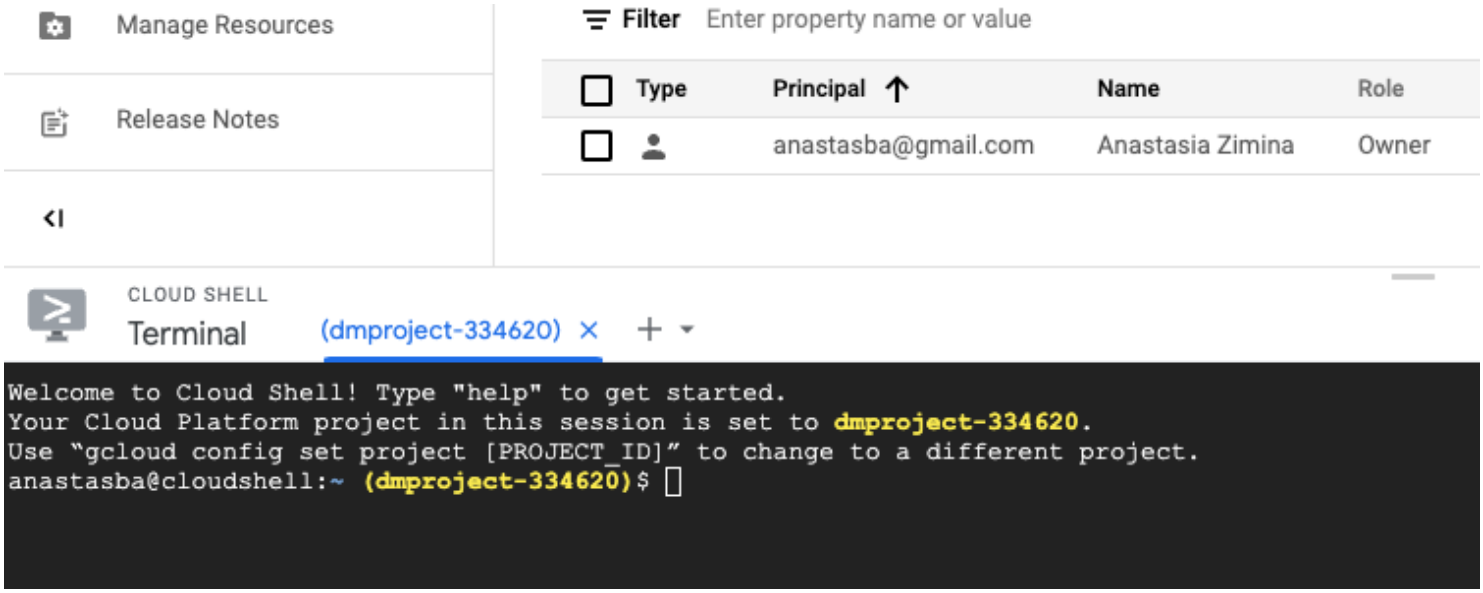
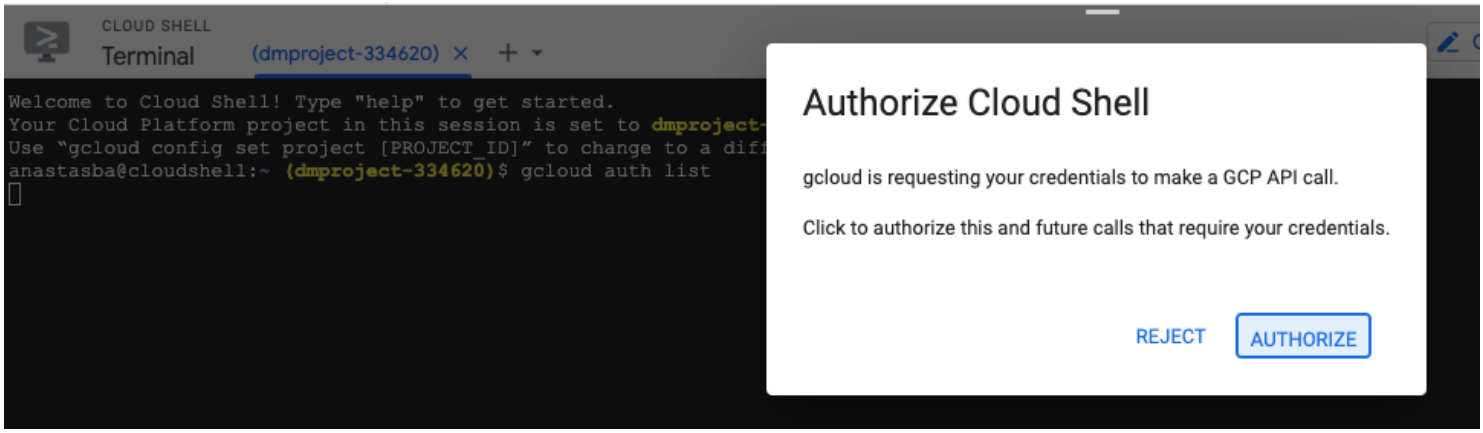
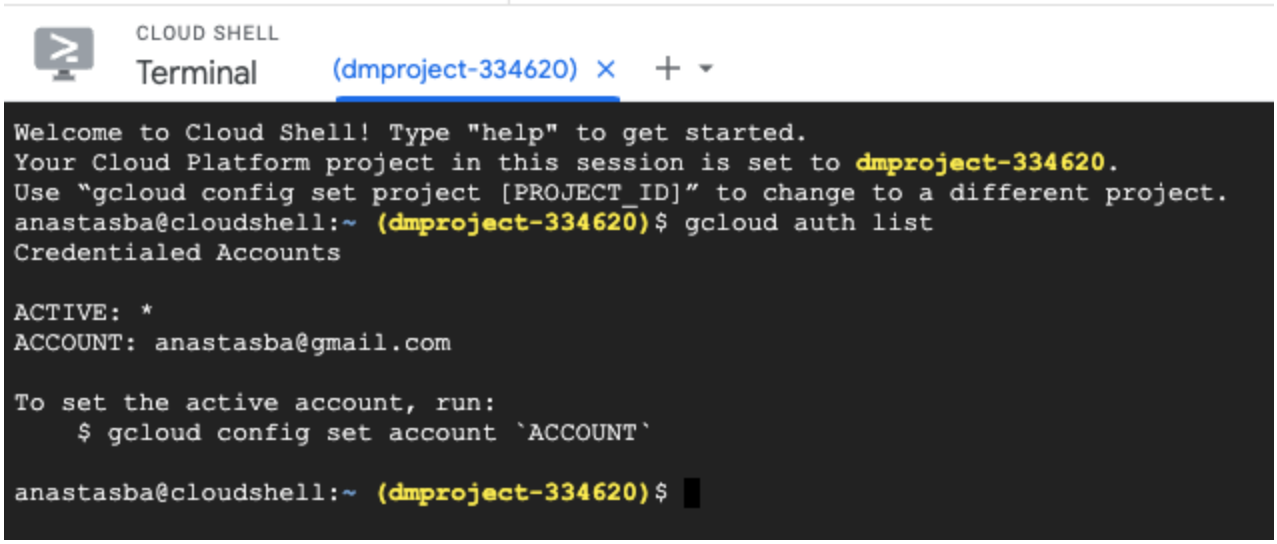
Intro to Vertex Pipelines

1. Create GCP project (with billing enabled)
2. Start Cloud Shell

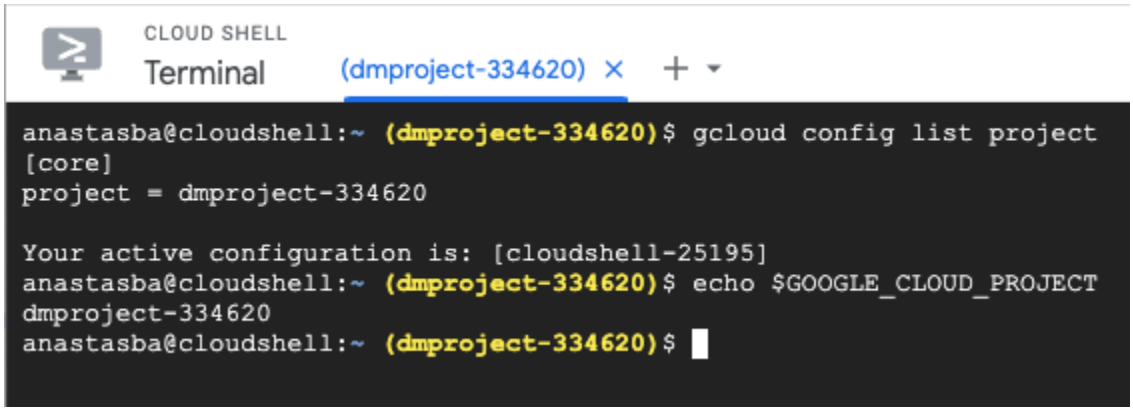


3. Authorize the Shell

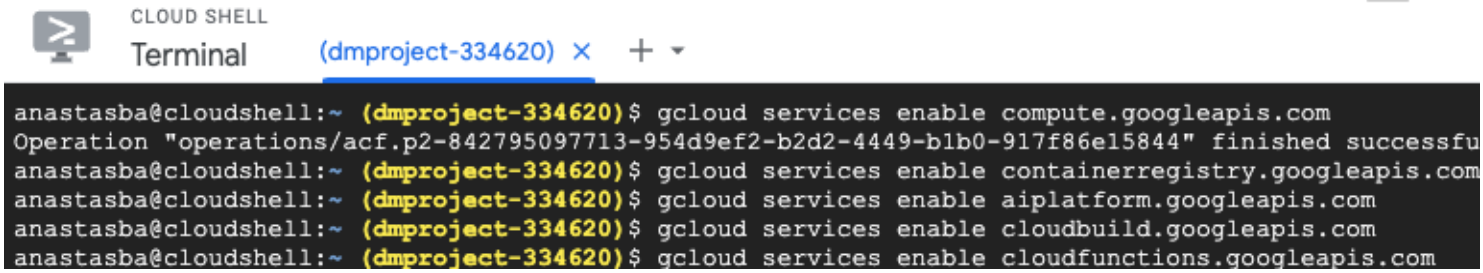




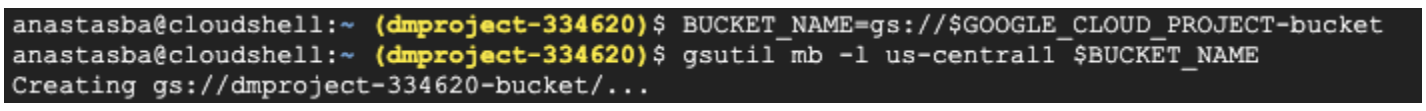
4. Confirm that the gcloud command knows about your project



5. Enable APIs (Compute Engine, Container Registry, and Vertex AI)



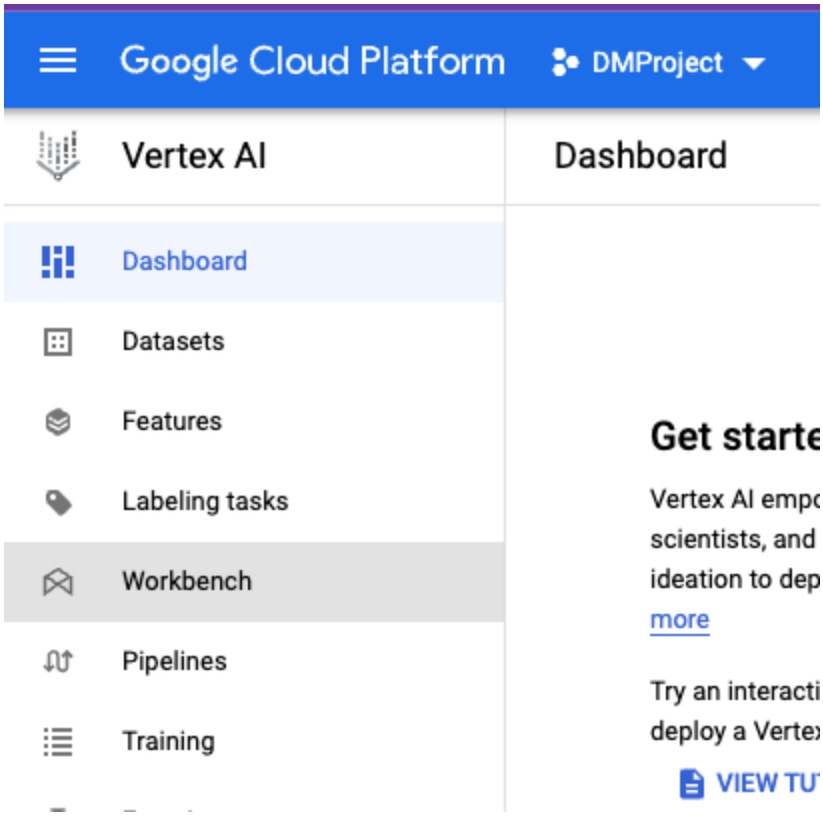
6. Create Cloud Storage Bucket



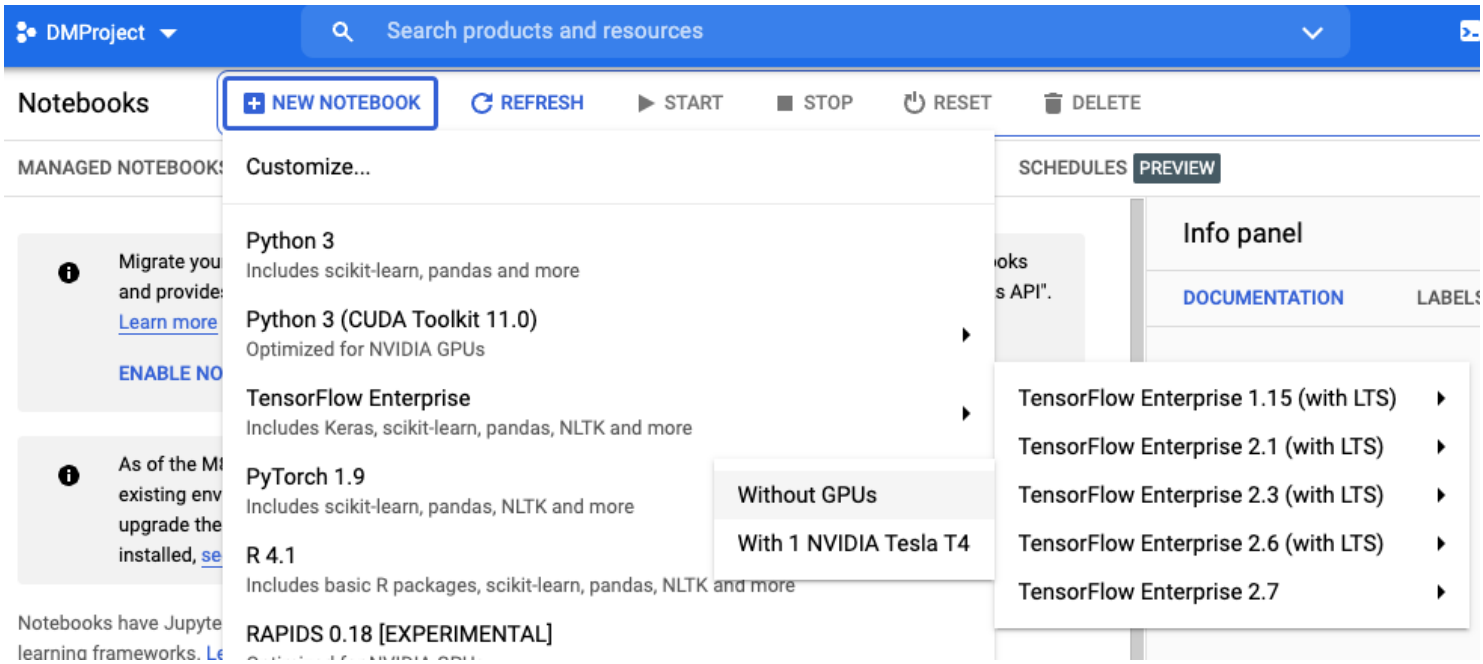
7. Give compute service account access to this bucket

```
anastasba@cloudshell:~ (dmproject-334620)$ gcloud projects describe $GOOGLE_CLOUD_PROJECT > project-info.txt
anastasba@cloudshell:~ (dmproject-334620)$ PROJECT_NUM=$(cat project-info.txt | sed -nre 's:.*projectNumber\: (.*):\1:p')
anastasba@cloudshell:~ (dmproject-334620)$ SVC_ACCOUNT="${PROJECT_NUM//\'/}-compute@developer.gserviceaccount.com"
anastasba@cloudshell:~ (dmproject-334620)$ gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT --member serviceAc
Updated IAM policy for project [dmproject-334620].
bindings:
- members:
  - serviceAccount:842795097713@cloudbuild.gserviceaccount.com
  role: roles/cloudbuild.builds.builder
- members:
  - serviceAccount:service-842795097713@gcp-sa-cloudbuild.iam.gserviceaccount.com
  role: roles/cloudbuild.serviceAgent
- members:
  - serviceAccount:service-842795097713@gcf-admin-robot.iam.gserviceaccount.com
  role: roles/cloudfunctions.serviceAgent
- members:
  - serviceAccount:service-842795097713@compute-system.iam.gserviceaccount.com
  role: roles/compute.serviceAgent
- members:
  - serviceAccount:service-842795097713@containerregistry.iam.gserviceaccount.com
  role: roles/containerregistry.ServiceAgent
- members:
  - serviceAccount:842795097713-compute@developer.gserviceaccount.com
  - serviceAccount:842795097713@cloudservices.gserviceaccount.com
  - serviceAccount:dmproject-334620@appspot.gserviceaccount.com
  role: roles/editor
- members:
  - user:anastasba@gmail.com
  role: roles/owner
- members:
  - serviceAccount:service-842795097713@gcp-sa-pubsub.iam.gserviceaccount.com
  role: roles/pubsub.serviceAgent
- members:
  - serviceAccount:842795097713-compute@developer.gserviceaccount.com
  role: roles/storage.objectAdmin
etag: BwXSvIuuSXg=
version: 1
```

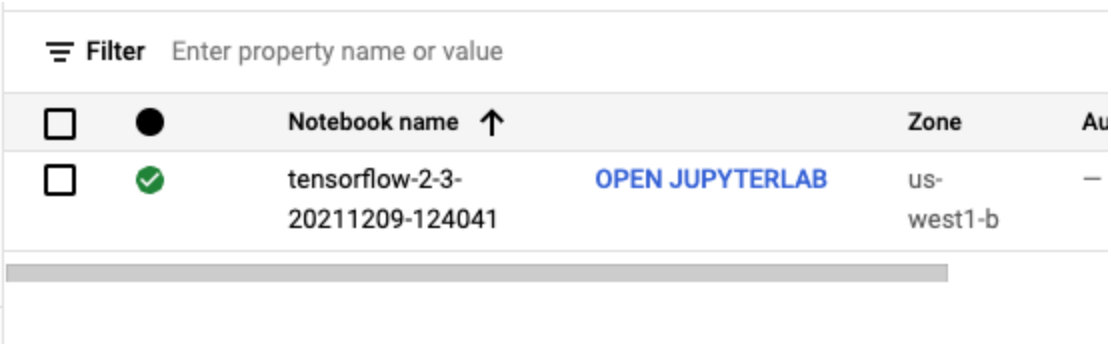
8. In the GUI create a Vertex AI Workbench instance



9. Create new Notebook (TensorFlow 2.3 Without GPUs)



10. Open Jupyterlab



11. Create Python3 notebook and install libraries

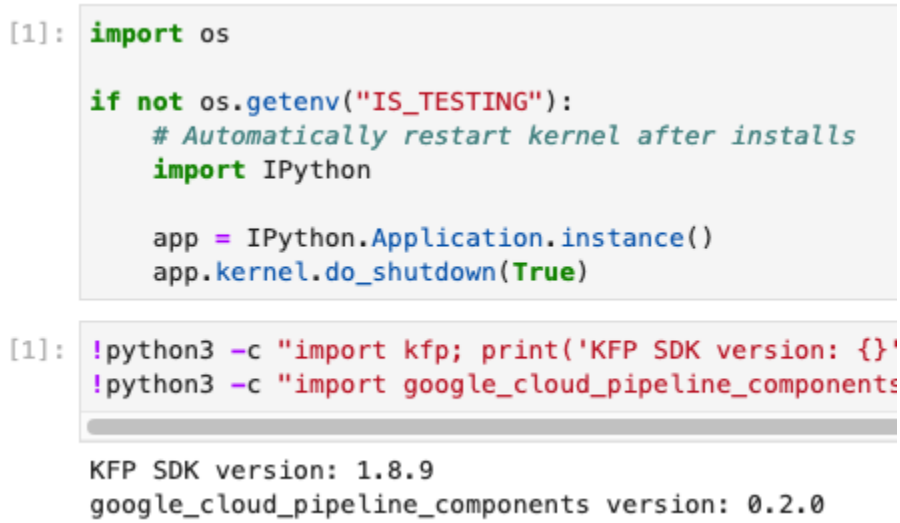
```
USER_FLAG = "--user"
!pip3 install {USER_FLAG} google-cloud-aiplatform==1.7.0 --upgrade
!pip3 install {USER_FLAG} kfp==1.8.9 google-cloud-pipeline-components==0.2.0
```

12. Restart Kernel

```
import os

if not os.getenv("IS_TESTING"):
    # Automatically restart kernel after installs
    import IPython

    app = IPython.Application.instance()
    app.kernel.do_shutdown(True)
```

13. Check that packages are installed

```
!python3 -c "import kfp; print('KFP SDK version: {}'.format(kfp.__version__))"
!python3 -c "import google_cloud_pipeline_components; print('google_cloud_pipeline_components
version: {}'.format(google_cloud_pipeline_components.__version__))"
```



14. Setup Project Id nad bucket

[2]:
```
import os
PROJECT_ID = ""

# Get your Google Cloud project ID from gcloud
if not os.getenv("IS_TESTING"):
    shell_output=!gcloud config list --format 'value(core.project)' 2>/dev/null
    PROJECT_ID = shell_output[0]
    print("Project ID: ", PROJECT_ID)
```

```
Project ID:  dmproject-334620
```

15. Create a variable to store the bucket name

```
BUCKET_NAME="gs://" + PROJECT_ID + "-bucket"
```

16. Import additional libraries

```
import kfp

from kfp.v2 import compiler, dsl
from kfp.v2.dsl import component, pipeline, Artifact, ClassificationMetrics, Input, Output,
Model, Metrics

from google.cloud import aiplatform
from google_cloud_pipeline_components import aiplatform as gcc_aip
from typing import NamedTuple
```

17. Define constants

I'm using us-west1-b

```
PATH=%env PATH
%env PATH={PATH}:/home/jupyter/.local/bin
REGION="us-west1-b"

PIPELINE_ROOT = f"{BUCKET_NAME}/pipeline_root/"
PIPELINE_ROOT
```

[5]:
```
PATH=%env PATH
%env PATH={PATH}:/home/jupyter/.local/bin
REGION="us-west1-b"

PIPELINE_ROOT = f"{BUCKET_NAME}/pipeline_root/"
PIPELINE_ROOT
```

```
env: PATH=/usr/local/cuda/bin:/opt/conda/bin:/opt/c
es:/home/jupyter/.local/bin
```

[5]:
```
'gs://dmproject-334620-bucket/pipeline_root/'
```

18. Create Test Pipeline

Pipeline prints out a sentence using two outputs: a product name and an emoji description.

This pipeline will consist of three components: product_name, emoji, build_sentence

a) Create a Python function based component

Base_image - container image

```
@component(base_image="python:3.9", output_component_file="first-component.yaml")
def product_name(text: str) -> str:
    return text

product_name_component = kfp.components.load_component_from_file('./first-component.yaml')
```

b) Create two additional components

```
@component(packages_to_install=["emoji"])
def emoji(
    text: str,
) -> NamedTuple(
    "Outputs",
    [
        ("emoji_text", str),  # Return parameters
        ("emoji", str),
    ],
):
    import emoji

    emoji_text = text
    emoji_str = emoji.emojize(':' + emoji_text + ':', use_aliases=True)
    print("output one: {}; output_two: {}".format(emoji_text, emoji_str))
    return (emoji_text, emoji_str)
```

```
@component
def build_sentence(
    product: str,
    emoji: str,
    emojitext: str
) -> str:
    print("We completed the pipeline, hooray!")
    end_str = product + " is "
    if len(emoji) > 0:
        end_str += emoji
    else:
        end_str += emojitext
    return(end_str)
```

c) Put components together into pipeline

Define an intro_pipeline function - specify the inputs to the initial pipeline steps, and how steps connect to each other

```
@pipeline(
    name="hello-world",
    description="An intro pipeline",
    pipeline_root=PIPELINE_ROOT,
)

# You can change the `text` and `emoji_str` parameters here to update the pipeline output
def intro_pipeline(text: str = "Vertex Pipelines", emoji_str: str = "sparkles"):
    product_task = product_name(text)
    emoji_task = emoji(emoji_str)
    consumer_task = build_sentence(
        product_task.output,
        emoji_task.outputs["emoji"],
        emoji_task.outputs["emoji_text"],
    )
```

d) Compile and run the pipeline

```
compiler.Compiler().compile(
    pipeline_func=intro_pipeline, package_path="intro_pipeline_job.json"
)
from datetime import datetime

TIMESTAMP = datetime.now().strftime("%Y%m%d%H%M%S")
```

e) Define pipeline job

```
job = aiplatform.PipelineJob(
    display_name="hello-world-pipeline",
    template_path="intro_pipeline_job.json",
    job_id="hello-world-pipeline-{0}".format(TIMESTAMP),
    enable_caching=True
)
```

f) Run the job

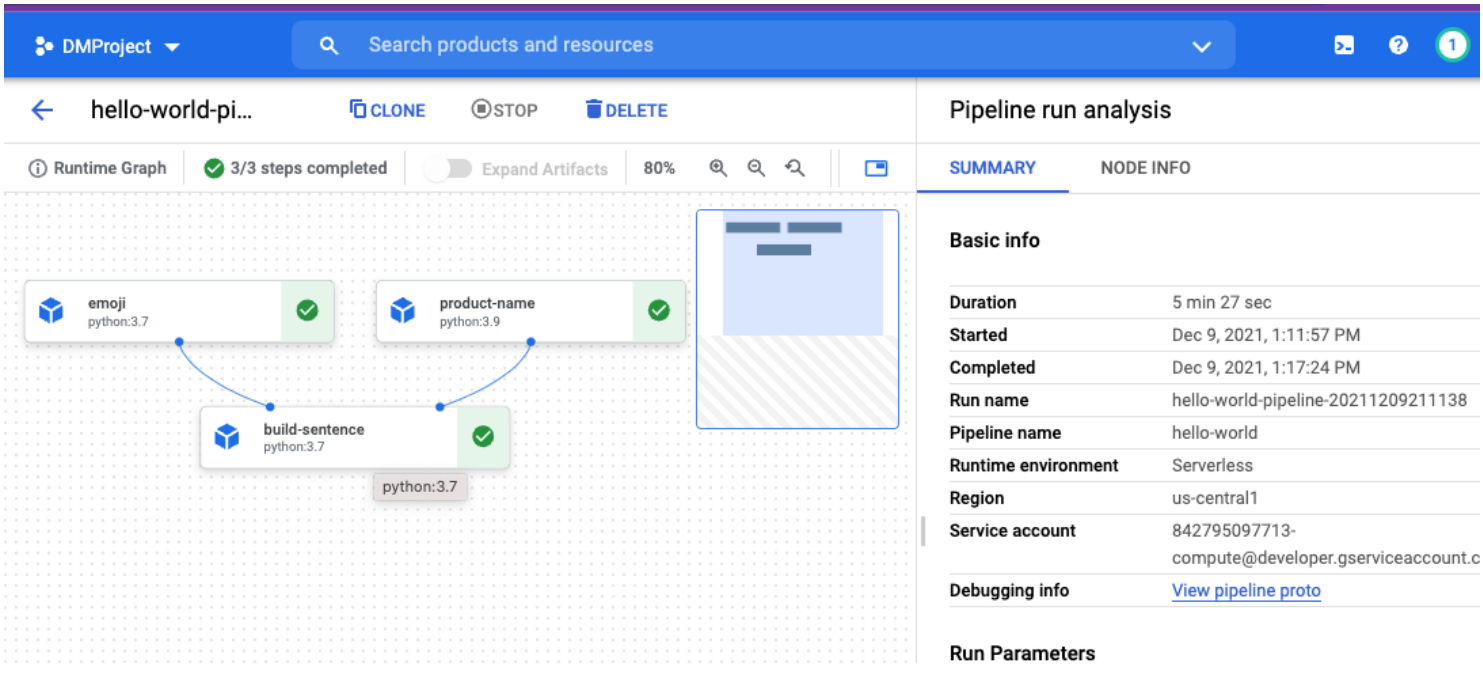
```
job.submit()
```

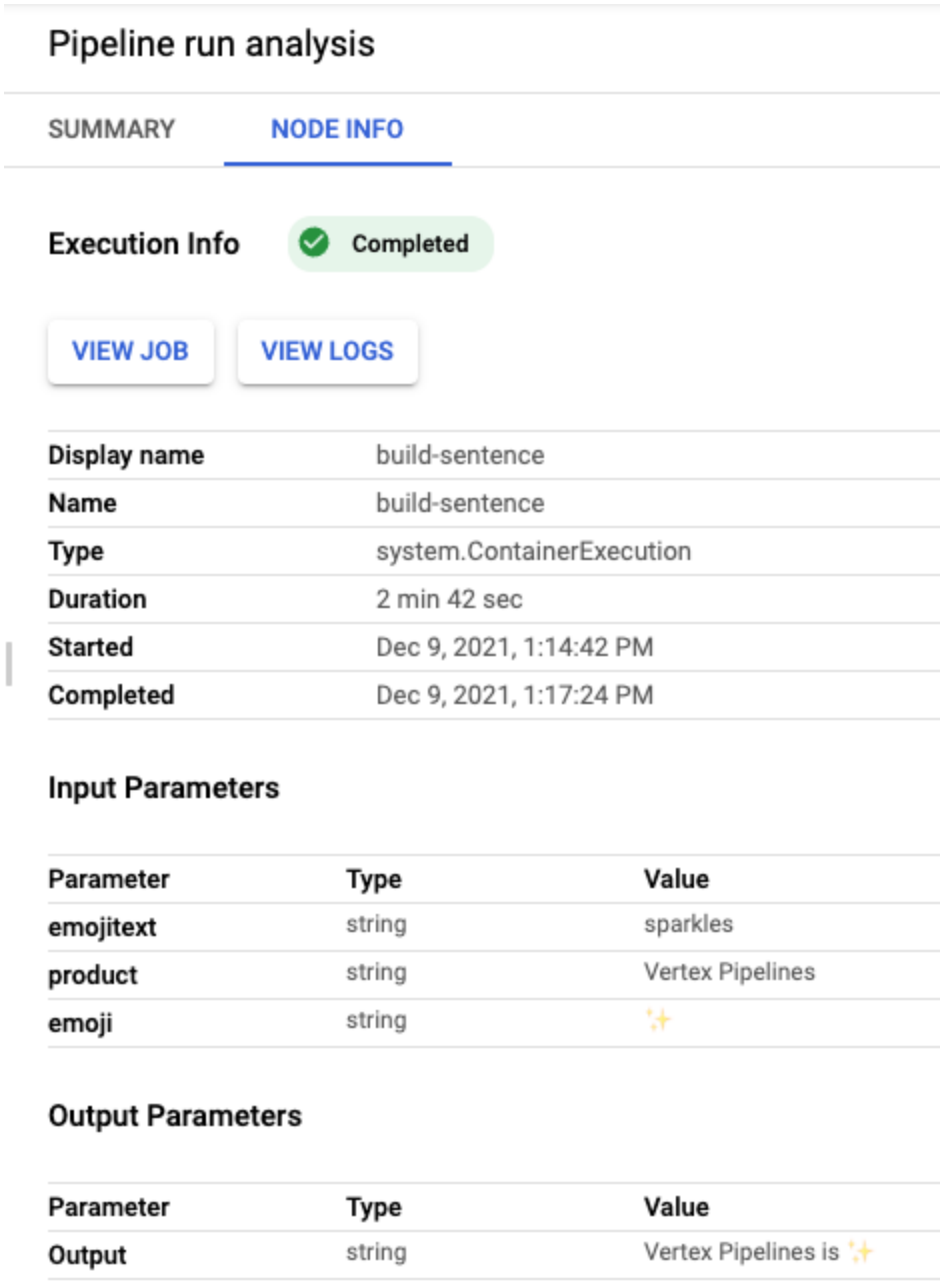
[12]:
```
from datetime import datetime

TIMESTAMP = datetime.now().strftime("%Y%m%d%H%M%S")
```

[13]:
```
job = aiplatform.PipelineJob(
    display_name="hello-world-pipeline",
    template_path="intro_pipeline_job.json",
    job_id="hello-world-pipeline-{0}".format(TIMESTAMP),
    enable_caching=True
)
```

[14]:
```
job.submit()
```

```
INFO:google.cloud.aiplatform.pipeline_jobs:Creating Pipeli
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob cre
al1/pipelineJobs/hello-world-pipeline-20211209211138
INFO:google.cloud.aiplatform.pipeline_jobs:To use this Pip
INFO:google.cloud.aiplatform.pipeline_jobs:pipeline_job =
s-central1/pipelineJobs/hello-world-pipeline-2021120921113
INFO:google.cloud.aiplatform.pipeline_jobs:View Pipeline J
https://console.cloud.google.com/vertex-ai/locations/us-ce
ject=842795097713
```





19. Create an end-to-end ML pipeline
Dataset: [Dry beans dataset]
Process: in the pipeline use the dataset to train, evaluate, and deploy an AutoML model
Task: classifies beans into one of 7 types based on their characteristics
    a) Create a custom component for model evaluation

[15]:
```
@component(
    base_image="gcr.io/deeplearning-platform-release/tf2-cpu.2-3:latest",
    output_component_file="tabular_eval_component.yaml",
    packages_to_install=["google-cloud-aiplatform"],
)
def classification_model_eval_metrics(
    project: str,
    location: str,  # "us-central1",
    api_endpoint: str,  # "us-central1-aiplatform.googleapis.com",
    thresholds_dict_str: str,
    model: Input[Artifact],
    metrics: Output[Metrics],
    metricsc: Output[ClassificationMetrics],
) -> NamedTuple("Outputs", [("dep_decision", str)]):  # Return parameter.

    import json
    import logging
```

b) Add Google prebild components

```
import time
DISPLAY_NAME = 'automl-beans{}'.format(str(int(time.time())))
print(DISPLAY_NAME)
```

[16]:
```
import time
DISPLAY_NAME = 'automl-beans{}'.format(str(int(time.time())))
print(DISPLAY_NAME)
```

```
automl-beans1639085123
```

c) Define the input parameters the pipeline takes. The rest of the pipeline uses a few pre-built components for interacting with Vertex AI services

[17]:
```
@pipeline(name="automl-tab-beans-training-v2",
                  pipeline_root=PIPELINE_ROOT)
def pipeline(
    bq_source: str = "bq://aju-dev-demos.beans.beans1",
    display_name: str = DISPLAY_NAME,
    project: str = PROJECT_ID,
    gcp_region: str = "us-central1",
    api_endpoint: str = "us-central1-aiplatform.googleapis.
    thresholds_dict_str: str = '{"auRoc": 0.95}',
):
    dataset_create_op = gcc_aip.TabularDatasetCreateOp(
        project=project, display_name=display_name, bq_sour
    )

    training_op = gcc_aip.AutoMLTabularTrainingJobRunOp(
        project=project,
        display_name=display_name,
```

d) Compile

```
compiler.Compiler().compile(
    pipeline_func=pipeline, package_path="tab_classif_pipeline.json"
)
```

e) Define the job

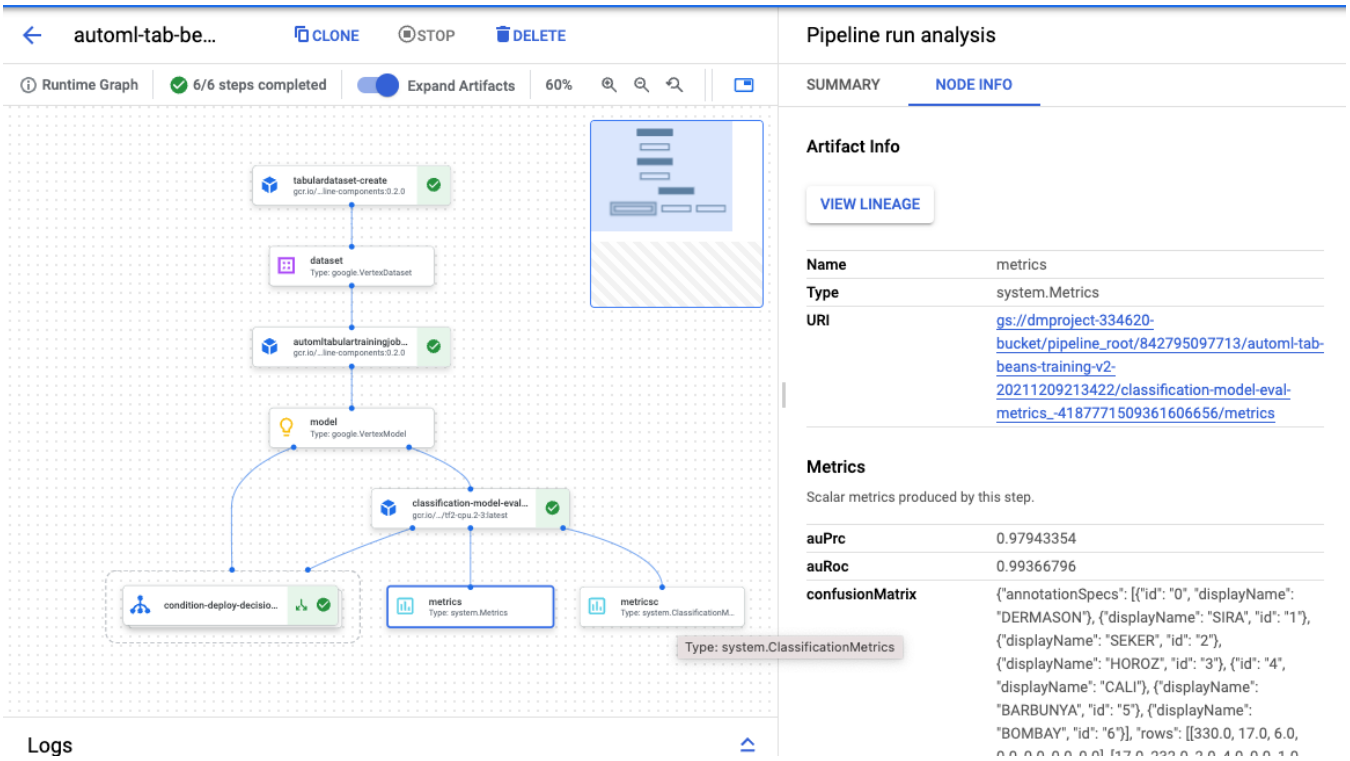
```
ml_pipeline_job = aiplatform.PipelineJob(
    display_name="automl-tab-beans-training",
    template_path="tab_classif_pipeline.json",
    pipeline_root=PIPELINE_ROOT,
    parameter_values={"project": PROJECT_ID, "display_name": DISPLAY_NAME},
    enable_caching=True
)
```

f) Run the job

```
ml_pipeline_job.submit()
```

[21]:
```
ml_pipeline_job.submit()
```

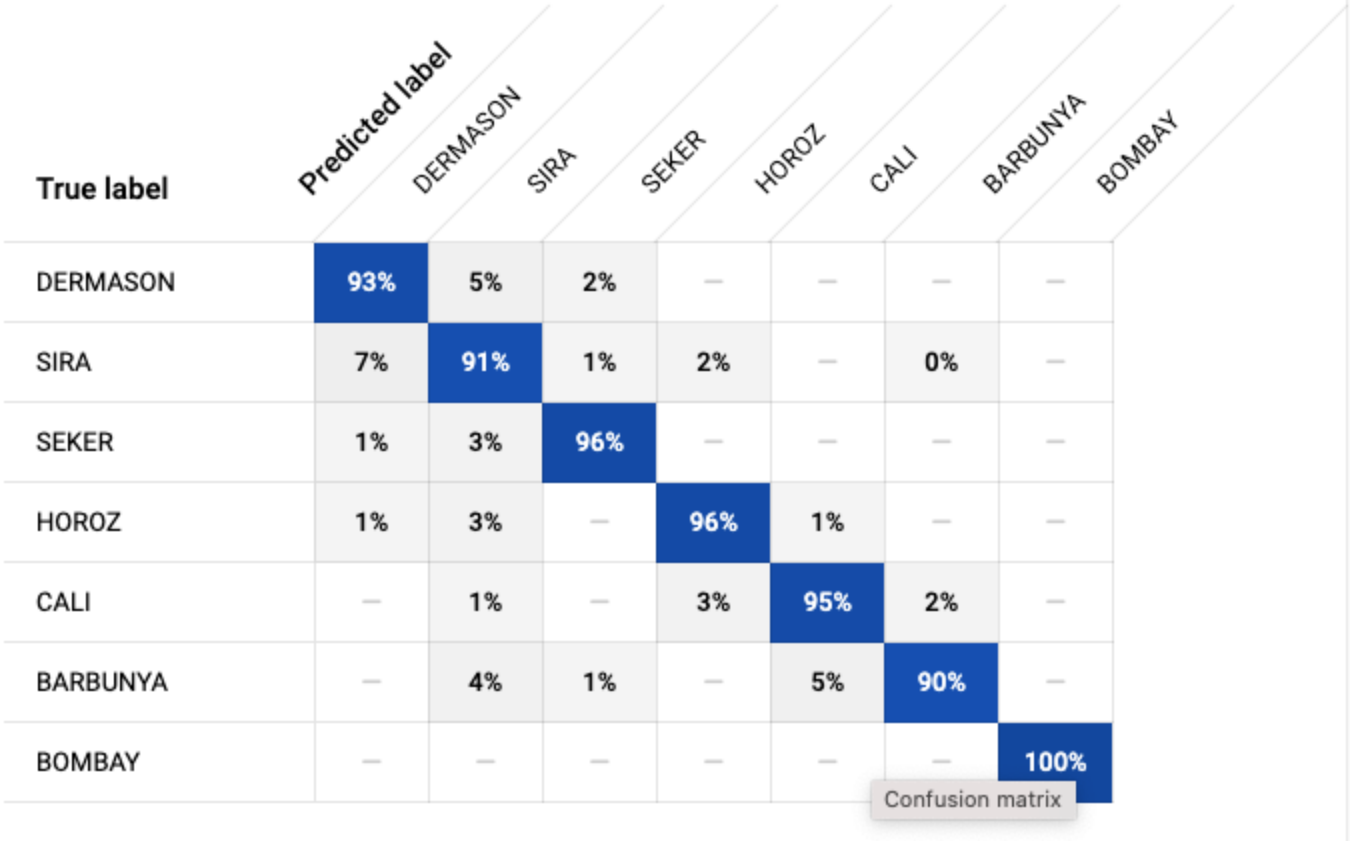
```
INFO:google.cloud.aiplatform.pipeline_jobs:Creating PipelineJob
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob created. Resource name: projects/842795097
al1/pipelineJobs/automl-tab-beans-training-v2-20211209213422
INFO:google.cloud.aiplatform.pipeline_jobs:To use this PipelineJob in another session:
INFO:google.cloud.aiplatform.pipeline_jobs:pipeline_job = aiplatform.PipelineJob.get('projects/84
s-central1/pipelineJobs/automl-tab-beans-training-v2-20211209213422')
INFO:google.cloud.aiplatform.pipeline_jobs:View Pipeline Job:
https://console.cloud.google.com/vertex-ai/locations/us-central1/pipelines/runs/automl-tab-beans-
3422?project=842795097713
```



Confusion matrix

Item counts

This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in gray).



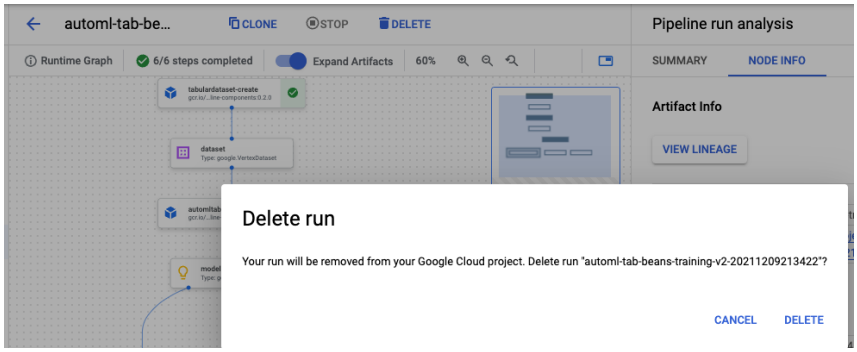
g) Comparing metrics across pipeline runs

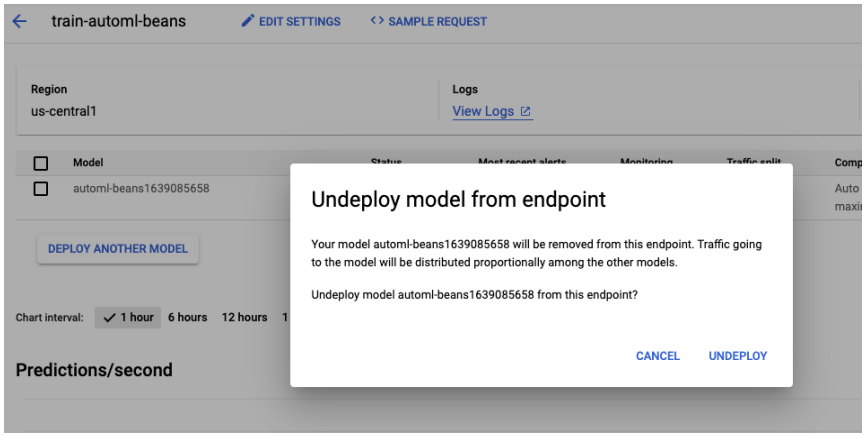
```
[22]: pipeline_df = aiplatform.get_pipeline_df(pipeline="automl-tab-beans-training-v2")
      small_pipeline_df = pipeline_df.head(2)
      small_pipeline_df
```

[22]:

	pipeline_name	run_name	param.input:thresholds_dict_str	param.input:gcp_region	param.input:api_endpoint	param.input:display_name	param.input:bq_source	param.input:proje
0	automl-tab-beans-training-v2	automl-tab-beans-training-v2-20211209213422	{"auRoc": 0.95}	us-central1	us-central1-aiplatform.googleapis.com	automl-beans1639085658	bq://aju-dev-demos.beans.beans1	dmproject-3346
1	automl-tab-beans-training-v2	automl-tab-beans-training-v2-20211209212820	{"auRoc": 0.95}	us-central1	us-central1-aiplatform.googleapis.com	automl-beans1639085123	bq://aju-dev-demos.beans.beans1	dmproject-3346

```
[ ]:
```

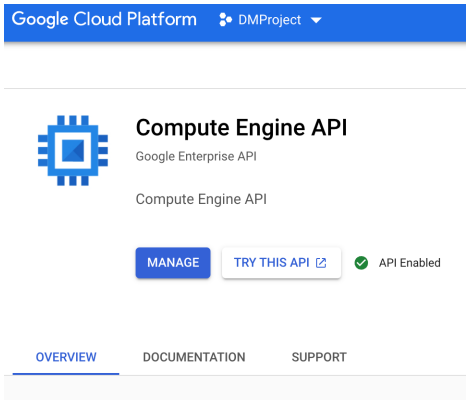




Building a fraud detection model with AutoML

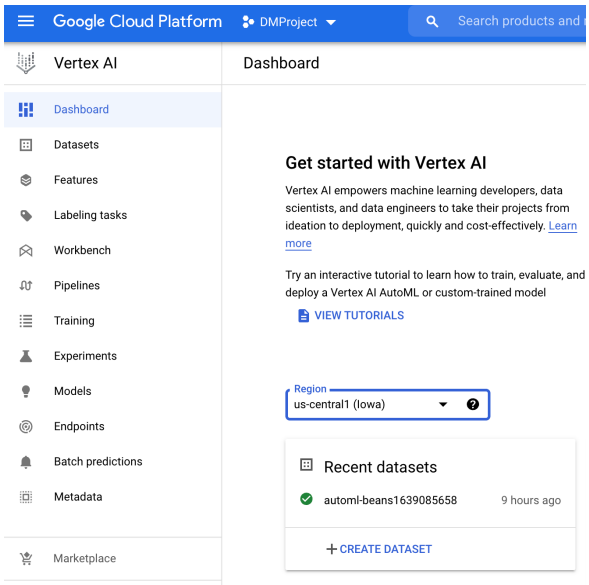
1. Enable Compute Engine

Done during the previous Lab.

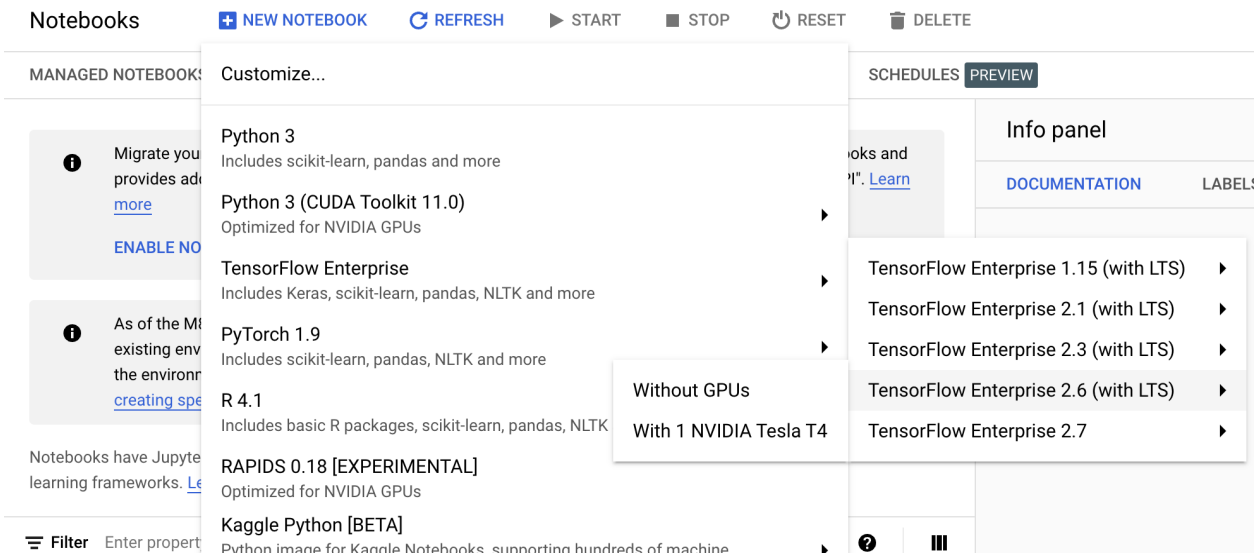


2. Enable the VertexAI API

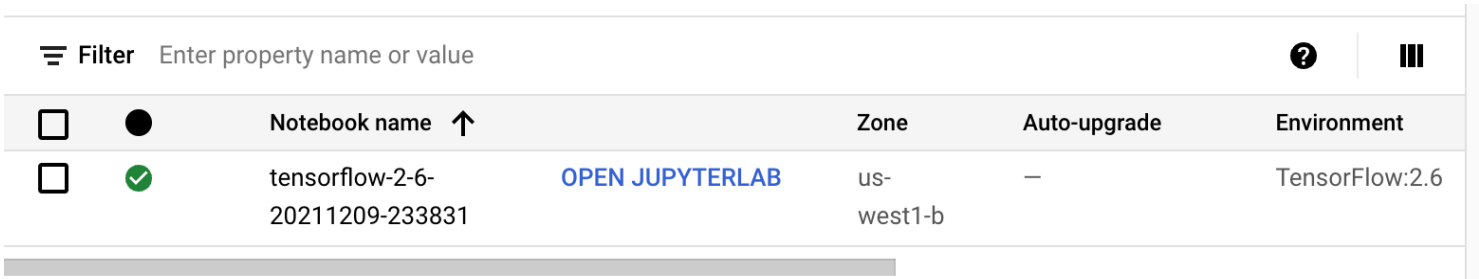
Done during previous Lab.



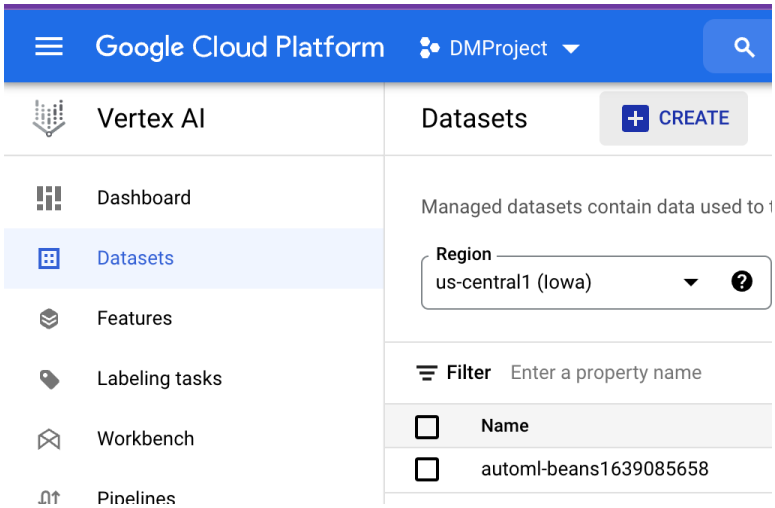
3. Create a Vertex AI Workbench instance and create new Notebook. TensorFlow 2.6 without GPU

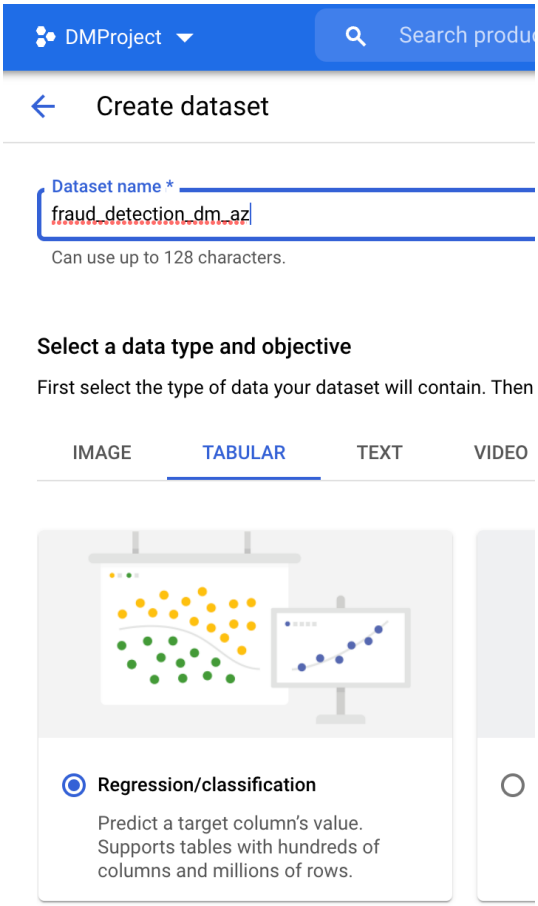


4. Open Notebook



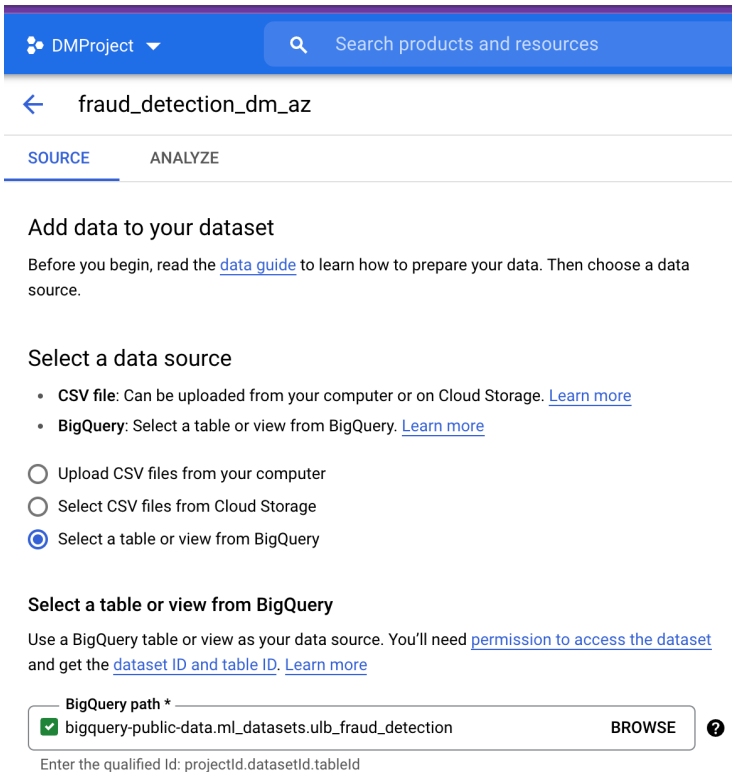
5. Create a dataset

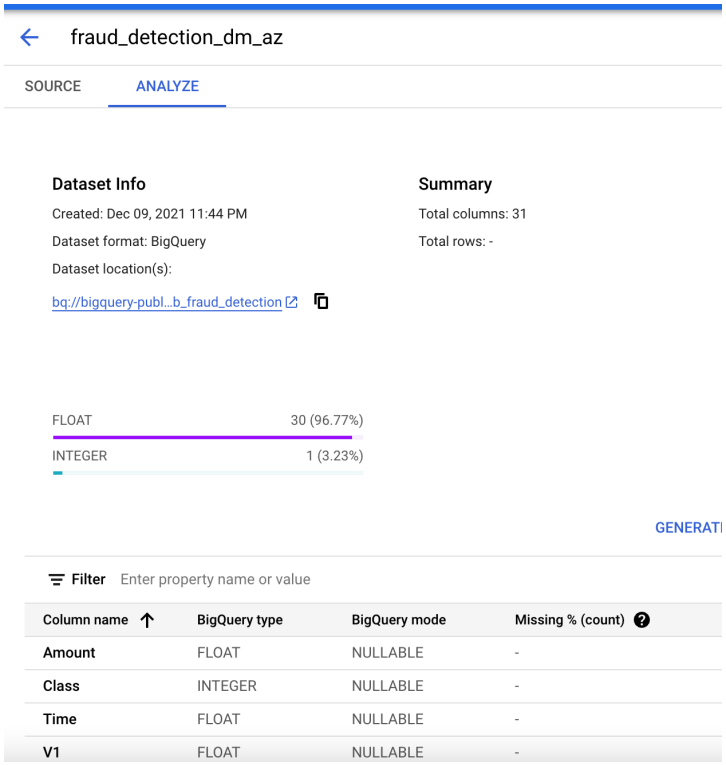




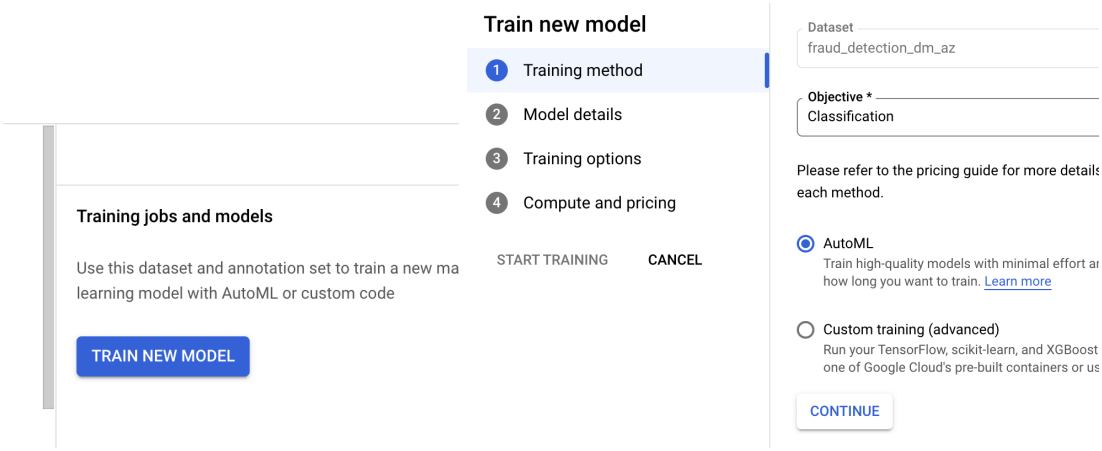
6. Select the datasource: BigQuery

`bigquery-public-data.ml_datasets.ulb_fraud_detection`

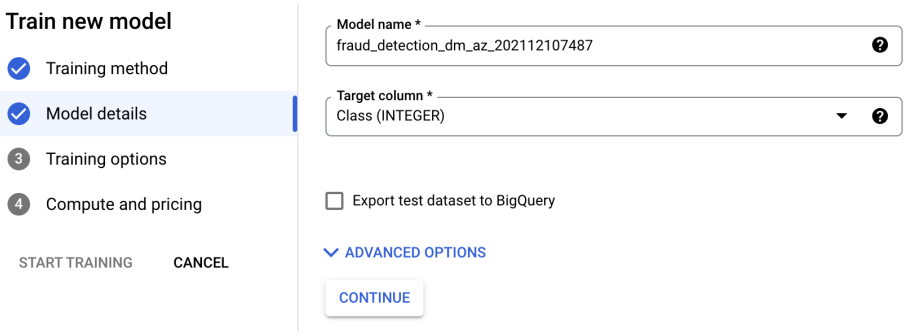




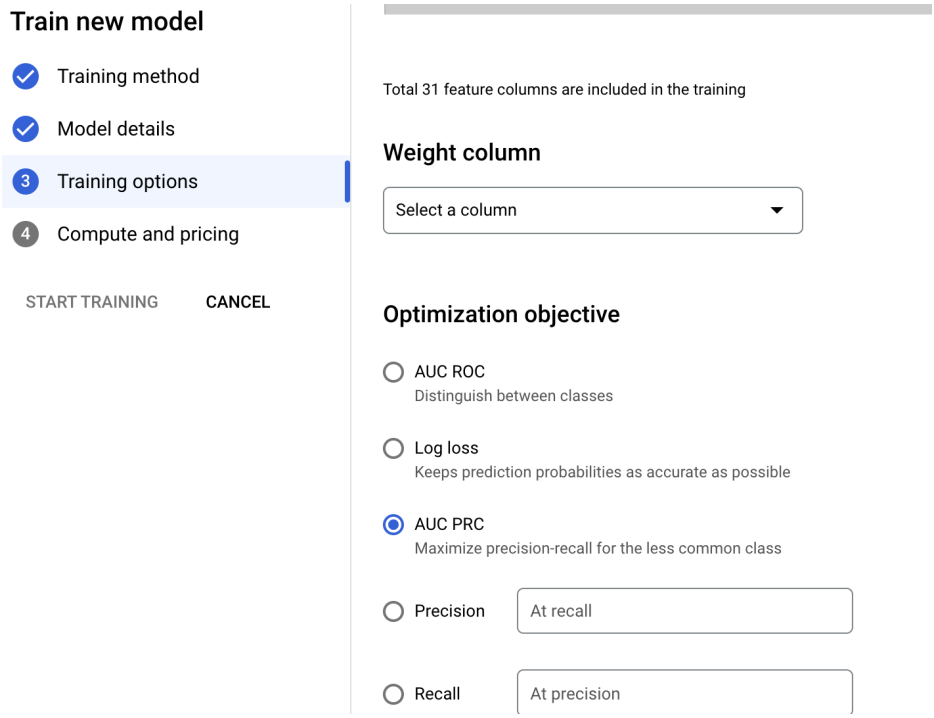
7. Train a model with AutoML



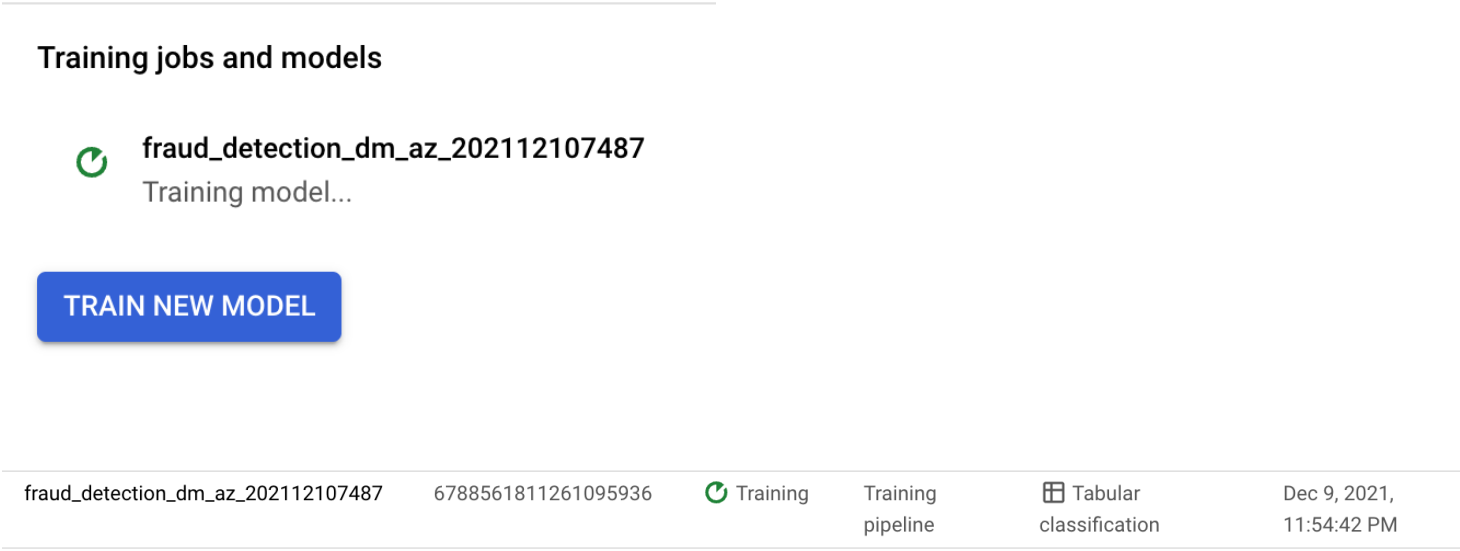
8. Select Target column (Class for this dataset)



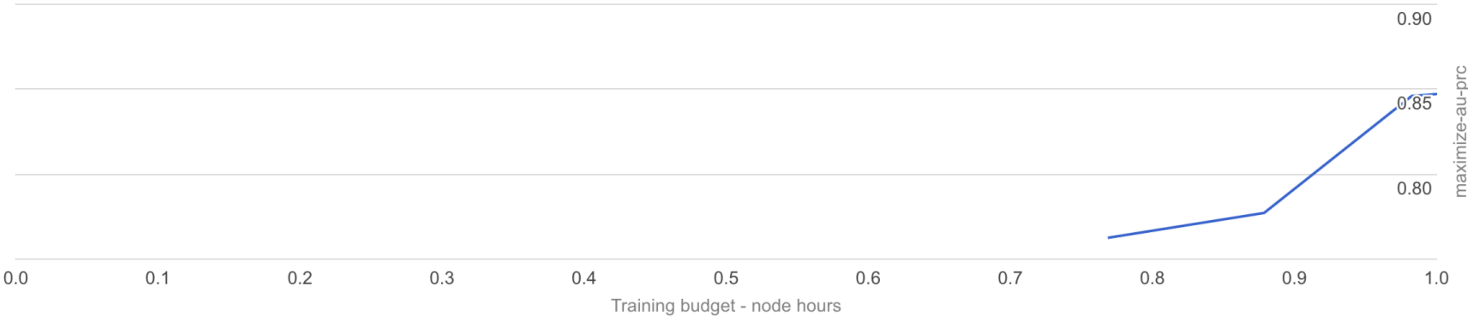
9. Add Optimization Objective (AUC PRC)



10. Start training and wait until the training is done



fraud_detection_dm_az_202112107487	6788561811261095936	Training	Training pipeline	Tabular classification	Dec 9, 2021, 11:54:42 PM



Filter Enter a property name

Name	ID	Status	Job type	Model type	Created	Elapsed time	L
fraud_detection_dm_az_202112107487	6788561811261095936	Finished	Training pipeline	Tabular classification	Dec 9, 2021, 11:54:42 PM	2 hr 16 min	-

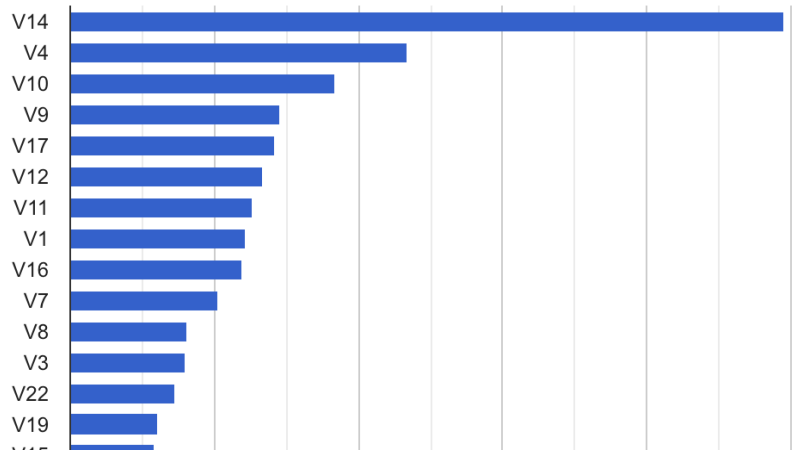
11. Explore model evaluation metrics

Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which

True label	Predicted label 0	Predicted label 1
0	100%	0%
1	22%	78%

Feature Importance



12. Deploy the model to model to the endpoint

fraud_detection_dm_az_202112107487

EVALUATE | DEPLOY & TEST | BATCH PREDICTIONS

Use your edge-optimized model

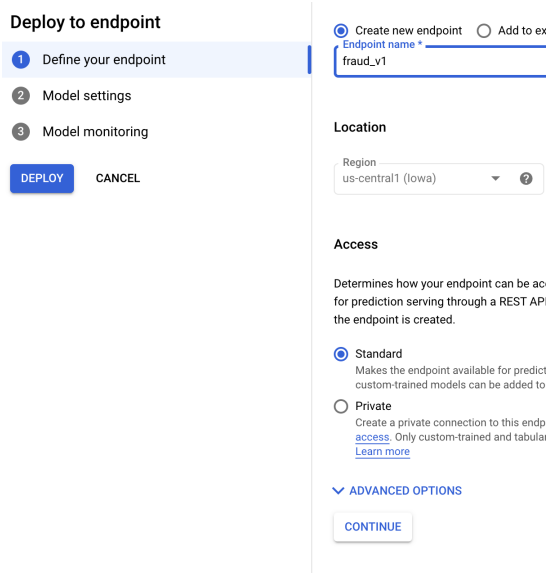
Container

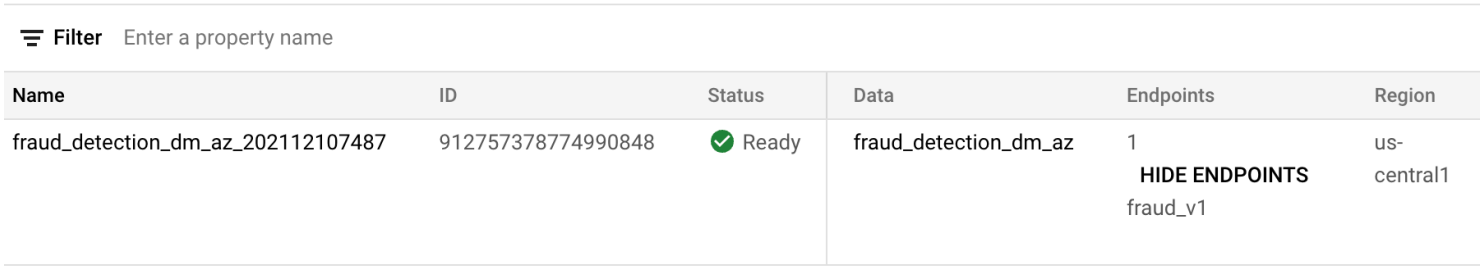
Export your model as a TF Saved Model to run on a Docker container.

Deploy your model

Endpoints are machine learning models made available for online predict
are useful for timely predictions from many users (for example, in respon
request). You can also request batch predictions if you don't need immec

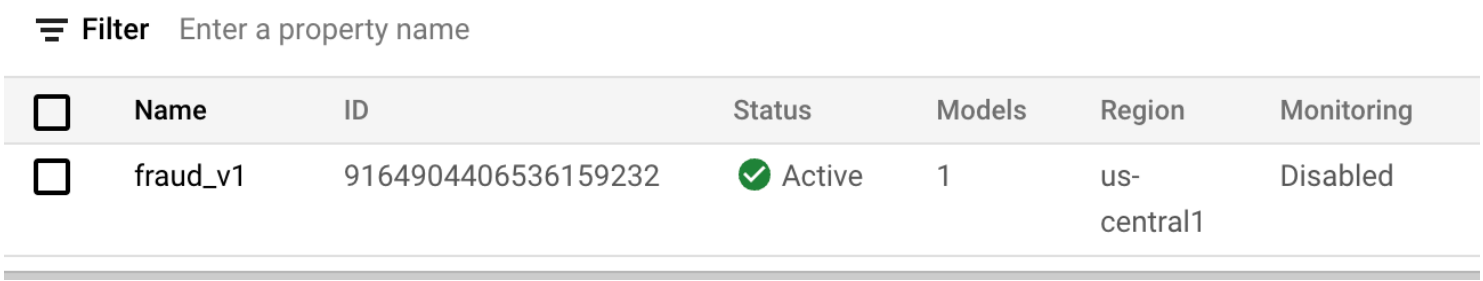
DEPLOY TO ENDPOINT





Filter Enter a property name

Name	ID	Status	Data	Endpoints	Region
fraud_detection_dm_az_202112107487	912757378774990848	Ready	fraud_detection_dm_az	1 HIDE ENDPOINTS fraud_v1	us-central1



Filter Enter a property name

	Name	ID	Status	Models	Region	Monitoring
☐	fraud_v1	9164904406536159232	Active	1	us-central1	Disabled

13. Create a new notebook (python3) and run the following command:

```
!pip3 install google-cloud-aiplatform --upgrade --user
```

14. Initiate object to call the endpoint

```
from google.cloud import aiplatform

endpoint = aiplatform.Endpoint(

endpoint_name="projects/YOUR-PROJECT-NUMBER/locations/us-central1/endpoints/YOUR-ENDPOINT-ID"
)
```

15. Make a prediction

[4]:
```
test_instance={
    'Time': 80422, 'Amount': 17.99, 'V1': -0.24,
    'V2': -0.027, 'V3': 0.064, 'V4': -0.16,
    'V5': -0.152, 'V6': -0.3, 'V7': -0.03,
    'V8': -0.01, 'V9': -0.13, 'V10': -0.18,
    'V11': -0.16, 'V12': 0.06, 'V13': -0.11,
    'V14': 2.1, 'V15': -0.07,'V16': -0.033,
    'V17': -0.14,
    'V18': -0.08,
    'V19': -0.062,
    'V20': -0.08,
    'V21': -0.06,
    'V22': -0.088,
    'V23': -0.03,
    'V24': 0.01,
    'V25': -0.04,
    'V26': -0.99,
    'V27': -0.13,
    'V28': 0.003
}

response = endpoint.predict([test_instance])

print('API response: ', response)
```

```
API response:  Prediction(predictions=[{'scores': [0.9999786615371704, 2.138644595106598e-05], 'classes': ['0', '1']}], d
eployed_model_id='6752136090544504832', explanations=None)
```

UI Prediction

fraud_detection_dm_az_202112107487 VIEW DATASET EXPORT

EVALUATE | DEPLOY & TEST | BATCH PREDICTIONS | MODEL PROPERTIES

Name	ID	Status	Models	Region	Monitoring	Most recent monitoring job	Most recent alerts	Last updated
fraud_v1	9164904406536159232	Active	1	us-central1	Disabled	—	—	Dec 10, 2021, 2:33:19 AM

Test your model PREVIEW

Feature column name	Type	Required or optional	Value	Local feature
Time	Numerical	Required	84871	0
V1	Numerical	Required	0.0202732068951347	0
V2	Numerical	Required	0.0657600286408065	0

Predict label

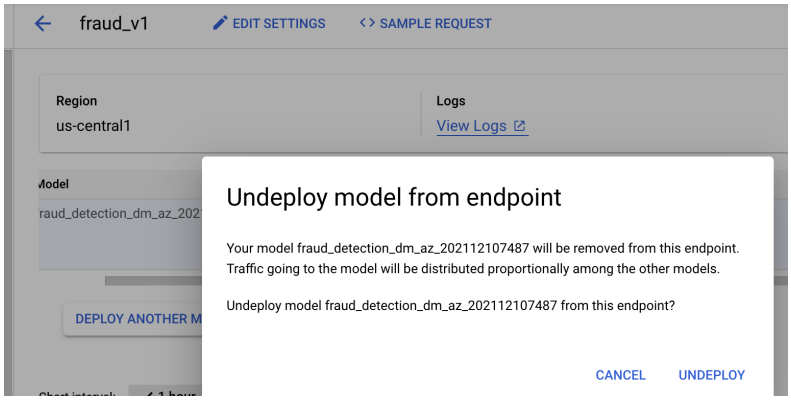
Prediction result

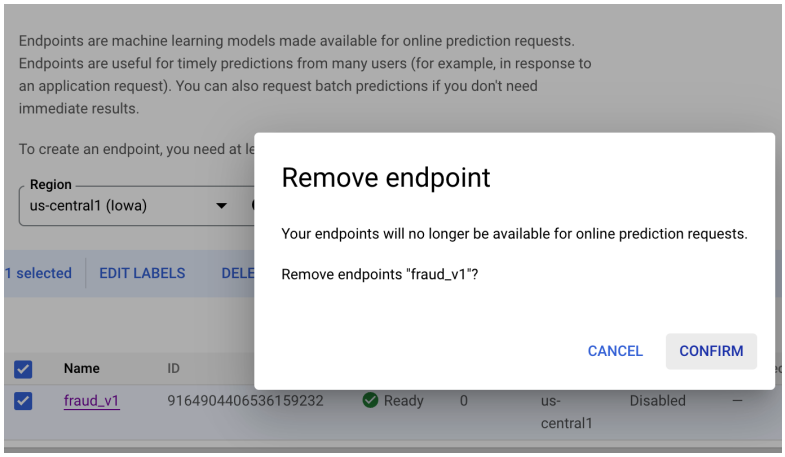
Selected label: 0

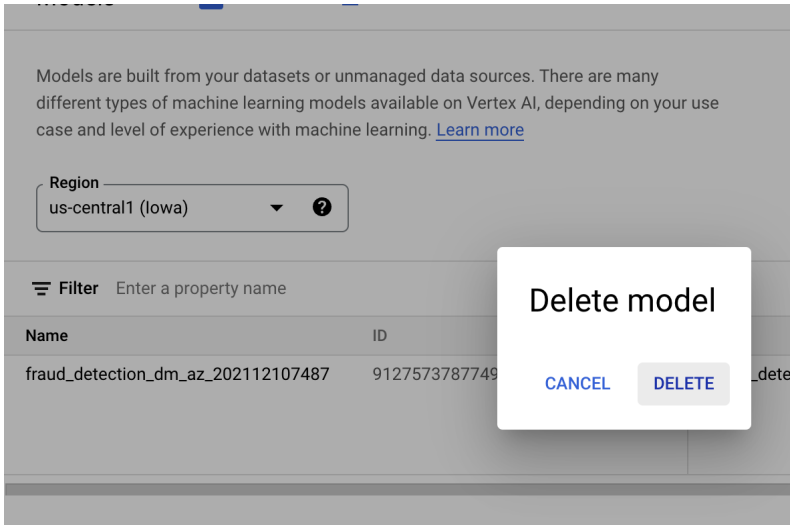
Baseline prediction value: 0.9934428334236145

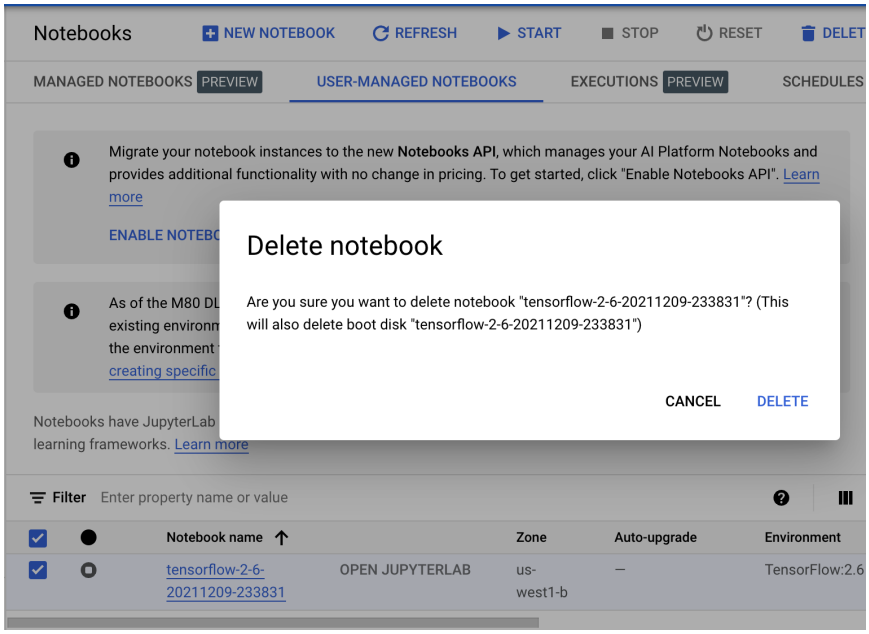
Confidence score: 0.9934428334236145

16. CleanUP


fraud_v1
EDIT SETTINGS
SAMPLE REQUEST
Region
us-central1
Logs
View Logs
Undeploy model from endpoint
Your model fraud_detection_dm_az_202112107487 will be removed from this endpoint.
Traffic going to the model will be distributed proportionally among the other models.
Undeploy model fraud_detection_dm_az_202112107487 from this endpoint?
CANCEL
UNDEPLOY


Endpoints are machine learning models made available for online prediction requests.
Endpoints are useful for timely predictions from many users (for example, in response to an application request). You can also request batch predictions if you don't need immediate results.
Region
us-central1 (Iowa)
Remove endpoint
Your endpoints will no longer be available for online prediction requests.
Remove endpoints "fraud_v1"?
CANCEL
CONFIRM
EDIT LABELS
Name
ID
fraud_v1
Ready
Disabled


Models are built from your datasets or unmanaged data sources. There are many different types of machine learning models available on Vertex AI, depending on your use case and level of experience with machine learning. Learn more
Region
us-central1 (Iowa)
Filter
Enter a property name
Name
ID
fraud_detection_dm_az_202112107487
Delete model
CANCEL
DELETE



Training and serving a custom model

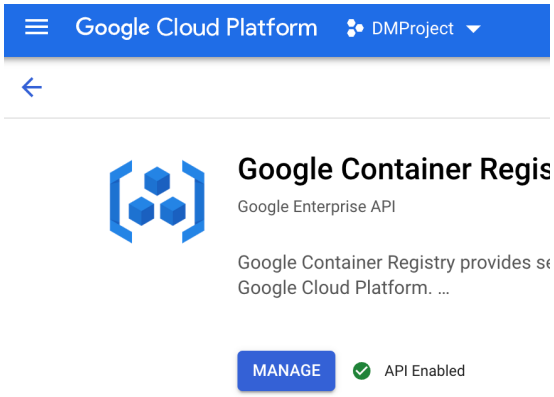
1. Enable Compute Engine

See above

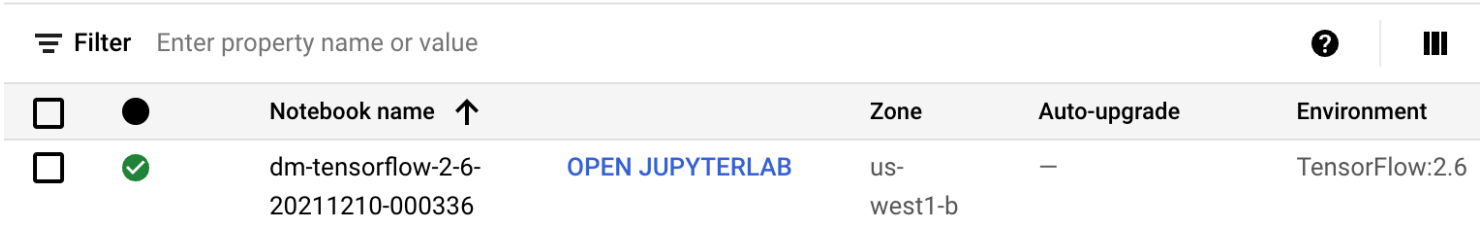
2. Enable Vertex AI API

See above

3. Enable Container Registry API

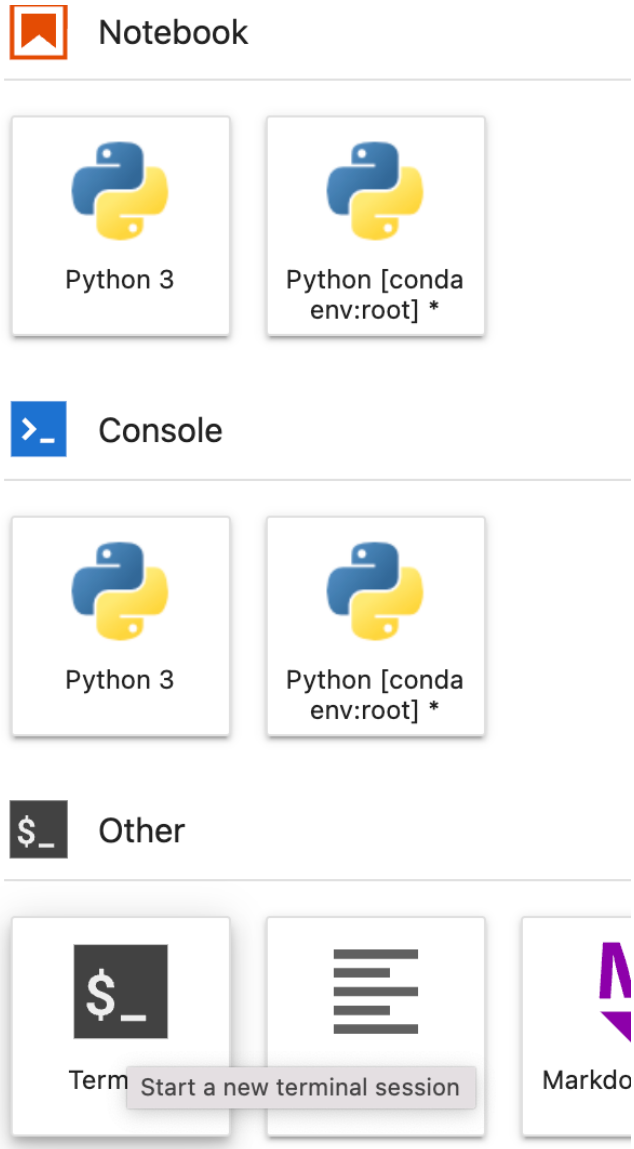


4. Create a new VertexAI Workbench Instance and a new Notebook (TensorFlow 2.6 without GPU)

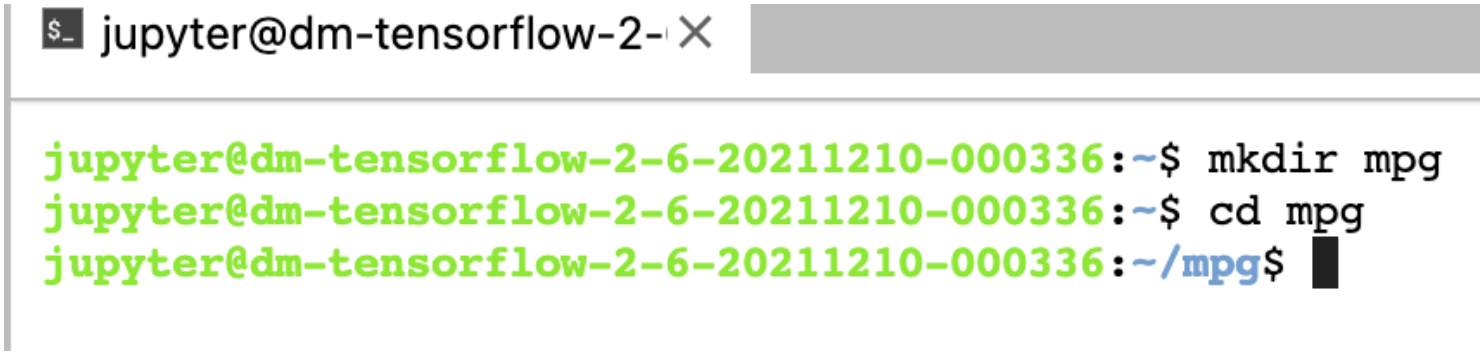


5. Containerize training code

a) Start a new terminal session



b) Create a new directory mpg and cd into it:



c) Create a docker file

```
FROM gcr.io/deeplearning-platform-release/tf2-cpu.2-6
WORKDIR /

# Copies the trainer code to the docker image.
COPY trainer /trainer

# Sets up the entry point to invoke the trainer.
ENTRYPOINT ["python", "-m", "trainer.train"]
```

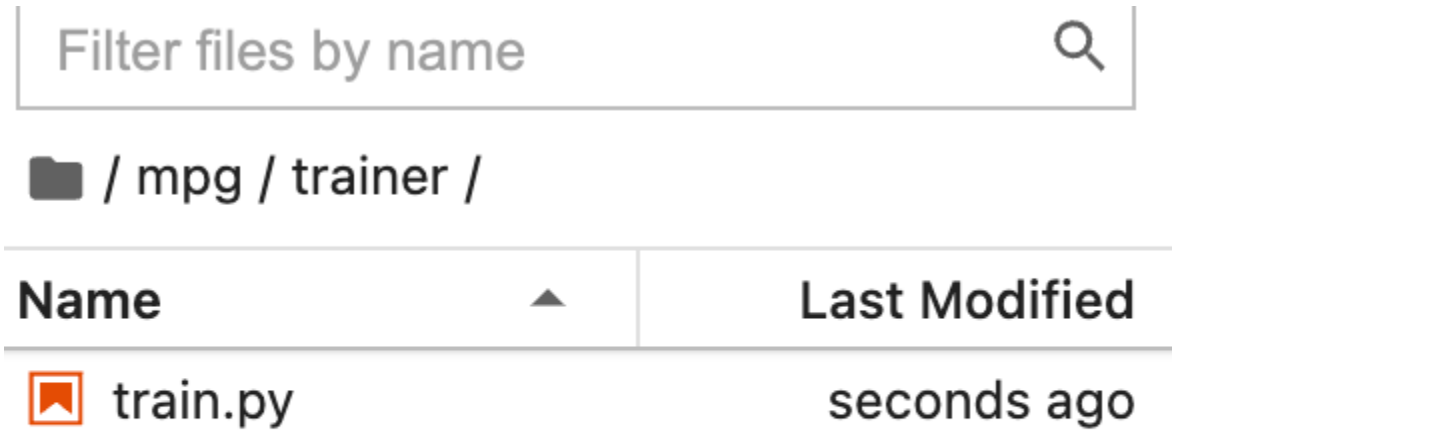
```
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ touch Dockerfile
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ cat Dockerfile
FROM gcr.io/deeplearning-platform-release/tf2-cpu.2-6
WORKDIR /

# Copies the trainer code to the docker image.
COPY trainer /trainer

# Sets up the entry point to invoke the trainer.
ENTRYPOINT ["python", "-m", "trainer.train"]jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$
```

d) Create a cloud Storage bucket

```
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ PROJECT_ID='dmproject-334620'
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ BUCKET_NAME="gs://${PROJECT_ID}-bucket"
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ gsutil mb -l us-central1 $BUCKET_NAME
Creating gs://dmproject-334620-bucket/...
```

e) Add model training code

```
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ mkdir trainer
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ touch trainer/train.py
```



f) Add content to the train.py file and update Bucket name

See notebook

g) Define a variable with the URI of your container image in Google Container Registry

```
IMAGE_URI="gcr.io/$PROJECT_ID/mpg:v1"
```

h) Build the container

```
docker build ./ -t $IMAGE_URI
```

```
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ IMAGE_URI="gcr.io/$PROJECT_ID/mpg:v1"
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ docker build ./ -t $IMAGE_URI
Sending build context to Docker daemon   16.9kB
Step 1/4 : FROM gcr.io/deeplearning-platform-release/tf2-cpu.2-6
latest: Pulling from deeplearning-platform-release/tf2-cpu.2-6
7b1a6ab2e44d: Pull complete
67a3d1735ac5: Pull complete
b99b0854a767: Pull complete
c48f8d8de60a: Pull complete
4f4fb700ef54: Pull complete
c816e0f4b043: Pull complete
71e0e2afbf19: Pull complete
```

i) Run the container

```
docker run $IMAGE_URI
```

```
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ docker run $IMAGE_URI
2.6.2
Downloading data from http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data
32768/30286 [================================] - 0s 1us/step
40960/30286 [========================================] - 0s 1us/step

User settings:

   KMP_AFFINITY=granularity=fine,verbose,compact,1,0
   KMP_BLOCKTIME=0
   KMP_SETTINGS=1

Effective settings:

   KMP_ABORT_DELAY=0
   KMP_ADAPTIVE_LOCK_PROPS='1,1024'
   KMP_ALIGN_ALLOC=64
   KMP_ALL_THREADPRIVATE=128
   KMP_ATOMIC_MODE=2
   KMP_BLOCKTIME=0
   KMP_CPUINFO_FILE: value is not defined
```

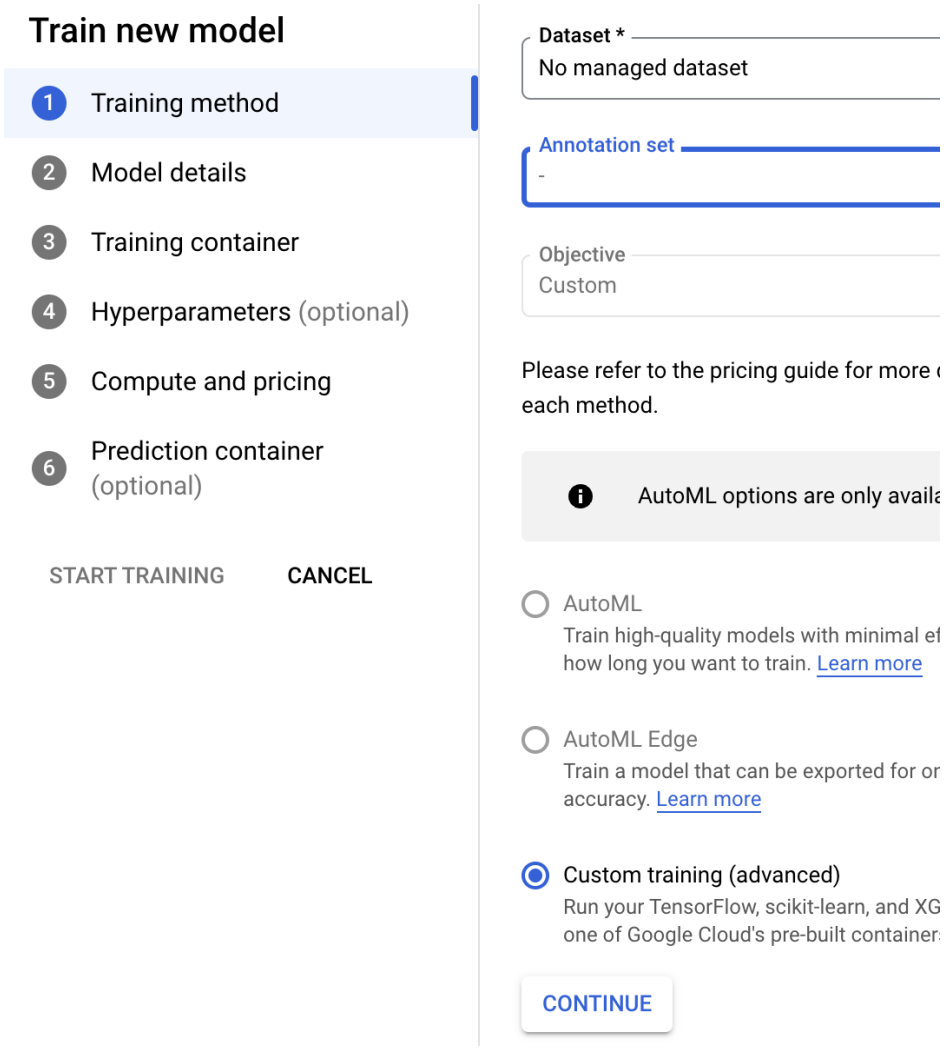
j) After training is done push the container to Google Container Registry

```
docker push $IMAGE_URI
```

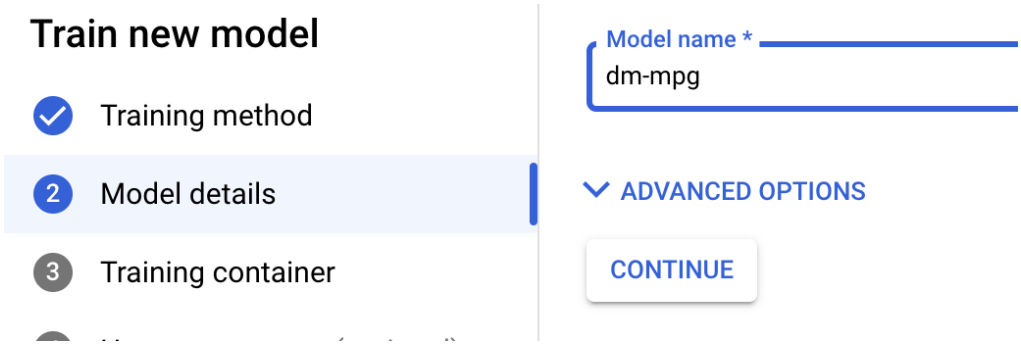
```
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ docker push $IMAGE_URI
The push refers to repository [gcr.io/dmproject-334620/mpg]
70cedac4141e: Pushed
2699358e44b2: Mounted from deeplearning-platform-release/tf2-cpu.2-6
393bde4f6bc5: Mounted from deeplearning-platform-release/tf2-cpu.2-6
cf3738cf85bb: Mounted from deeplearning-platform-release/tf2-cpu.2-6
1b3c3815787f: Mounted from deeplearning-platform-release/tf2-cpu.2-6
244680cd2339: Mounted from deeplearning-platform-release/tf2-cpu.2-6
2d3b6e42c63d: Mounted from deeplearning-platform-release/tf2-cpu.2-6
b538f5dfbd26: Mounted from deeplearning-platform-release/tf2-cpu.2-6
539edf6b8fe8: Mounted from deeplearning-platform-release/tf2-cpu.2-6
f212849c96cc: Mounted from deeplearning-platform-release/tf2-cpu.2-6
7121d6cc04ae: Mounted from deeplearning-platform-release/tf2-cpu.2-6
c787e4bdd37e: Mounted from deeplearning-platform-release/tf2-cpu.2-6
9f6f8dd09608: Mounted from deeplearning-platform-release/tf2-cpu.2-6
9feb64009273: Mounted from deeplearning-platform-release/tf2-cpu.2-6
b69e9b09c4c0: Mounted from deeplearning-platform-release/tf2-cpu.2-6
54363f143f4d: Mounted from deeplearning-platform-release/tf2-cpu.2-6
5f70bf18a086: Layer already exists
b3b3550ec84e: Mounted from deeplearning-platform-release/tf2-cpu.2-6
48515e7cf8a8: Mounted from deeplearning-platform-release/tf2-cpu.2-6
82b8c1c41f58: Mounted from deeplearning-platform-release/tf2-cpu.2-6
9f54eef41275: Layer already exists
v1: digest: sha256:26b36cd07c0286d1fb2f4ecb41a3a543c70d6d6189061c585e55
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$
```

6. Run a training job on VertexAI

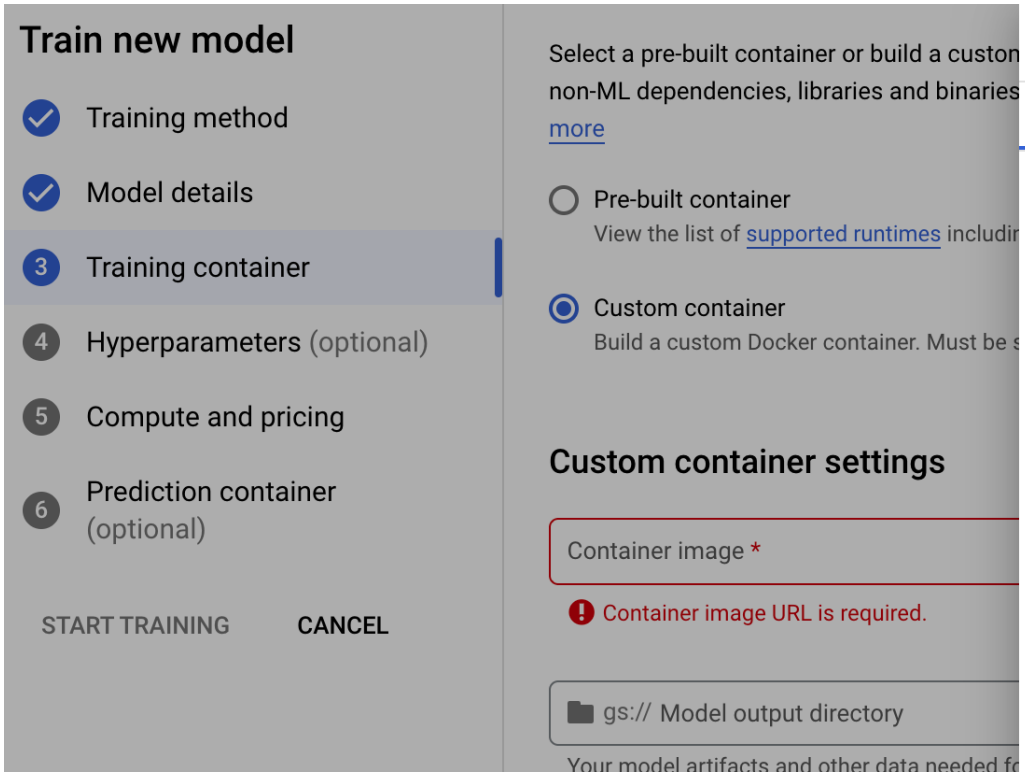
Select ‘No managed dataset’ and ‘Custom training (advanced)’ options



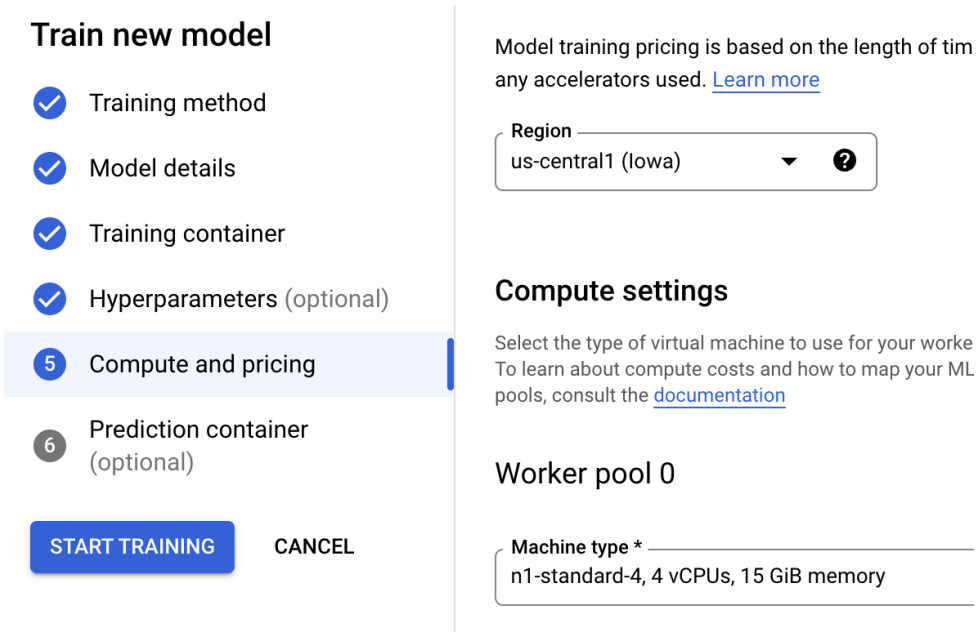
7. Select model name



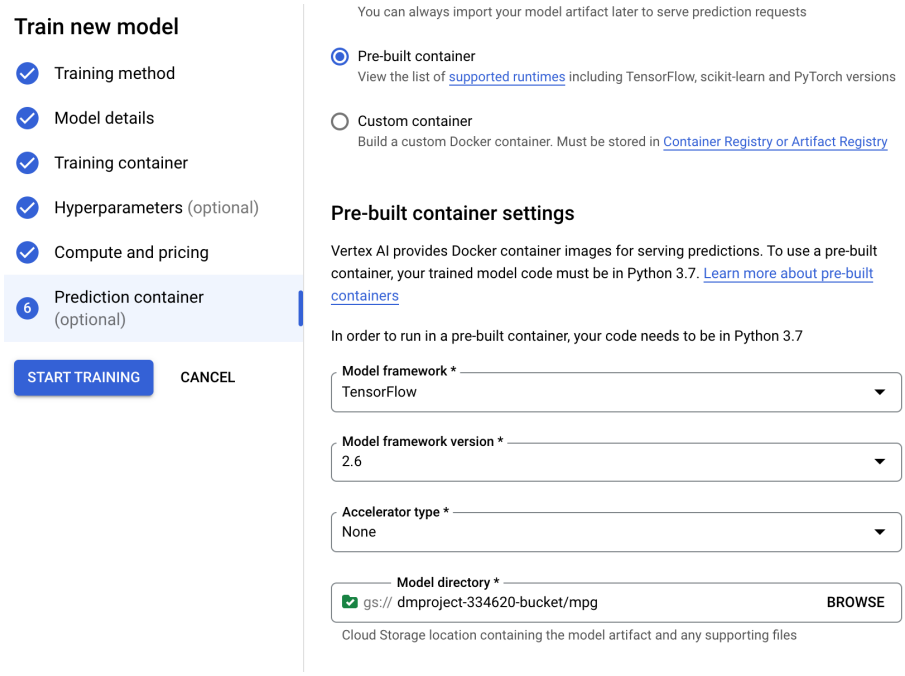
8. Select custom container image

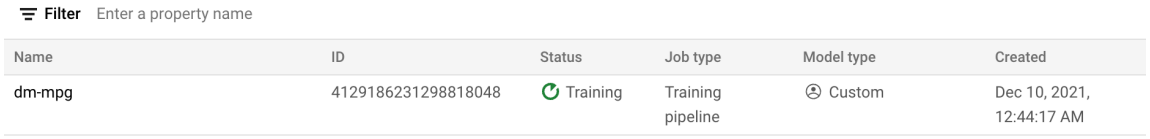


9. Choose compute settings: **n1-standard-4**



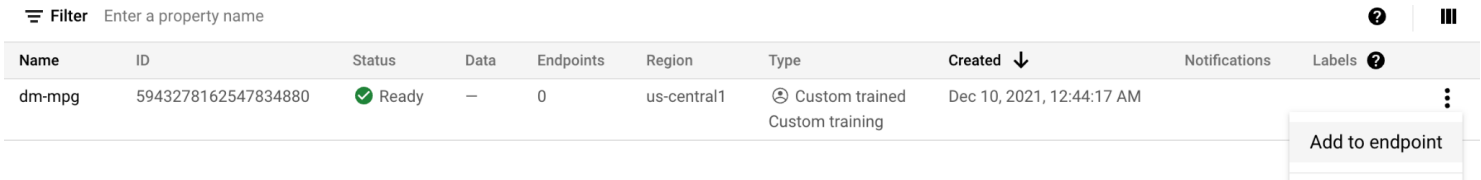
10. Select Pre-built container and start building



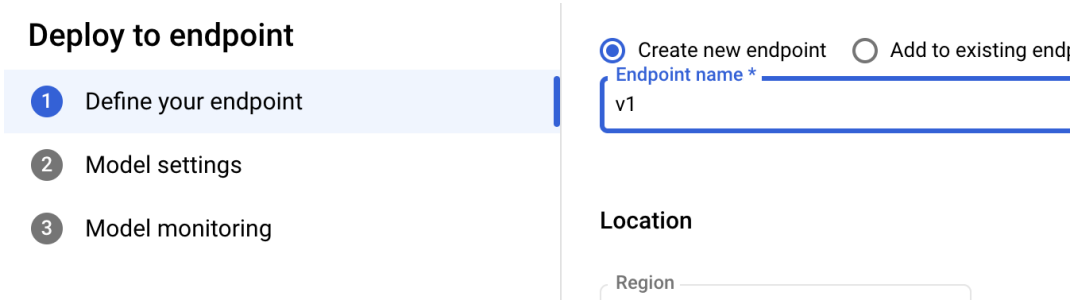


11. Deploy the endpoint

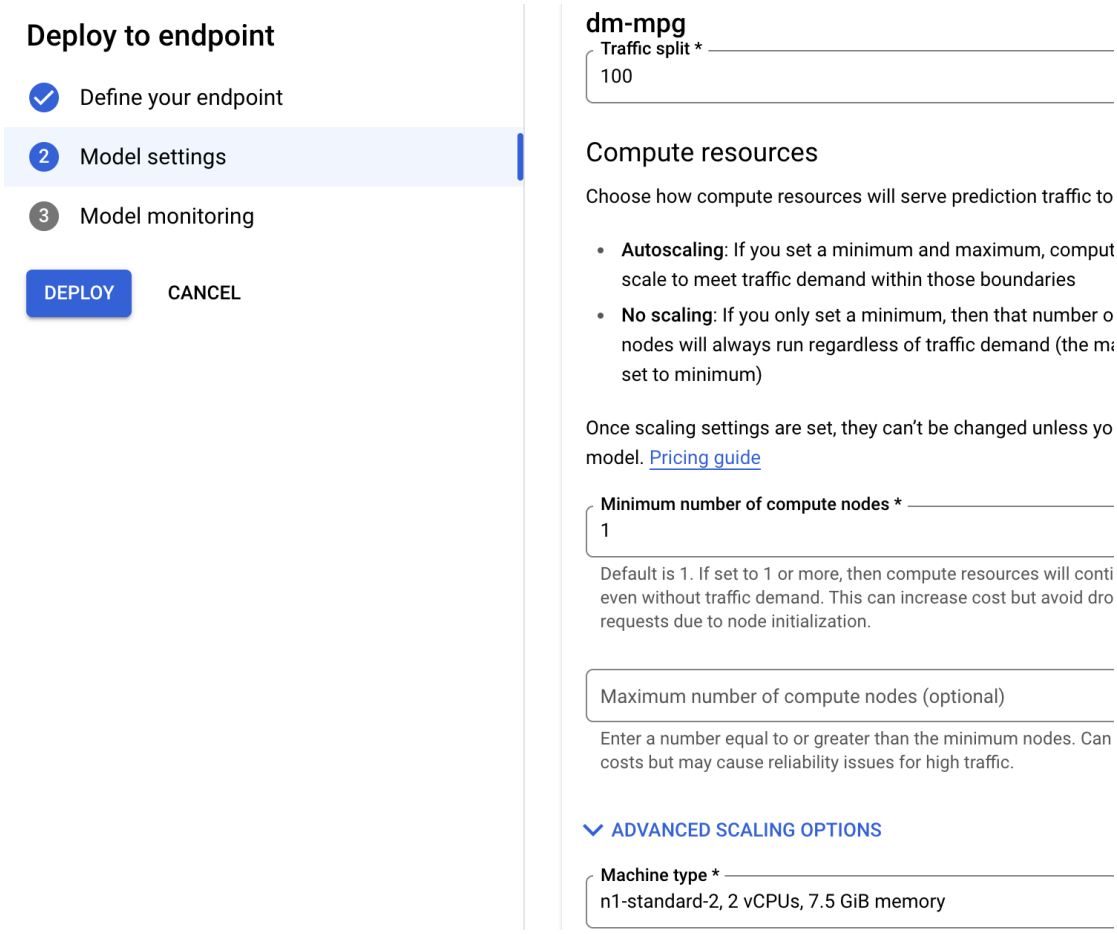
a) After the model was created add it to the endpoint

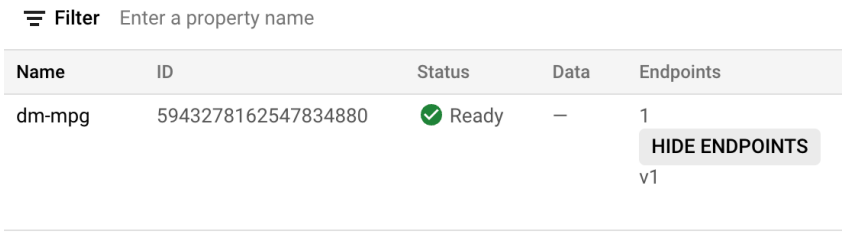


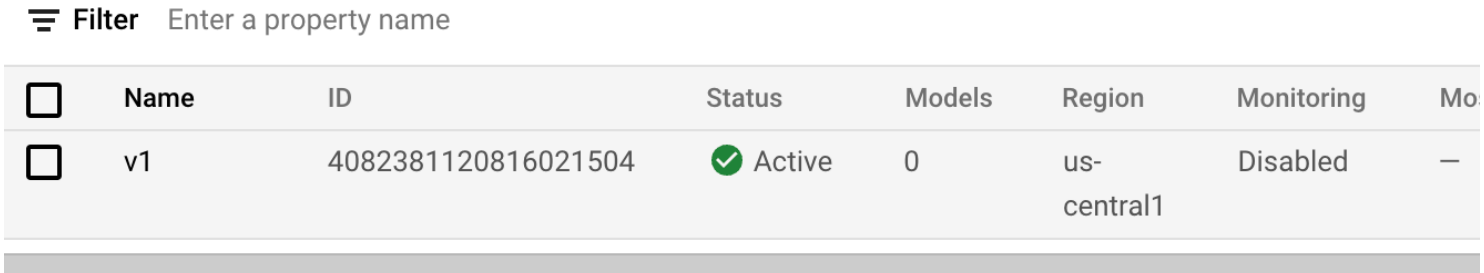
b) Provide endpoint the name



c) Set traffic split to 100, Minimum number of compute nodes to 1 and Machine type to n1-standart-2







12. Get predictions on the deployed model
    a) Create new Notebook (Python 3) and run the following command

```
!pip3 install google-cloud-aiplatform --upgrade --user
```

```
[1]: !pip3 install google-cloud-aiplatform --upgrade --user

Requirement already satisfied: google-cloud-aiplatform in /opt/conda/
Collecting google-cloud-aiplatform
  Downloading google_cloud_aiplatform-1.8.0-py2.py3-none-any.whl (1.7
     |████████████████████████████████| 1.7 MB 8.0 MB/s
Requirement already satisfied: google-api-core[grpc]<3.0.0dev,>=1.26.
ogle-cloud-aiplatform) (2.2.2)
Requirement already satisfied: packaging>=14.3 in /opt/conda/lib/pyth
(21.3)
Requirement already satisfied: proto-plus>=1.10.1 in /opt/conda/lib/p
m) (1.19.8)
Requirement already satisfied: google-cloud-storage<2.0.0dev,>=1.32.0
```

b) Add project and endpoint number

```
from google.cloud import aiplatform

endpoint = aiplatform.Endpoint(

endpoint_name="projects/YOUR-PROJECT-NUMBER/locations/us-central1/endpoints/YOUR-ENDPOINT-ID"
)
```

c) Make a prediction

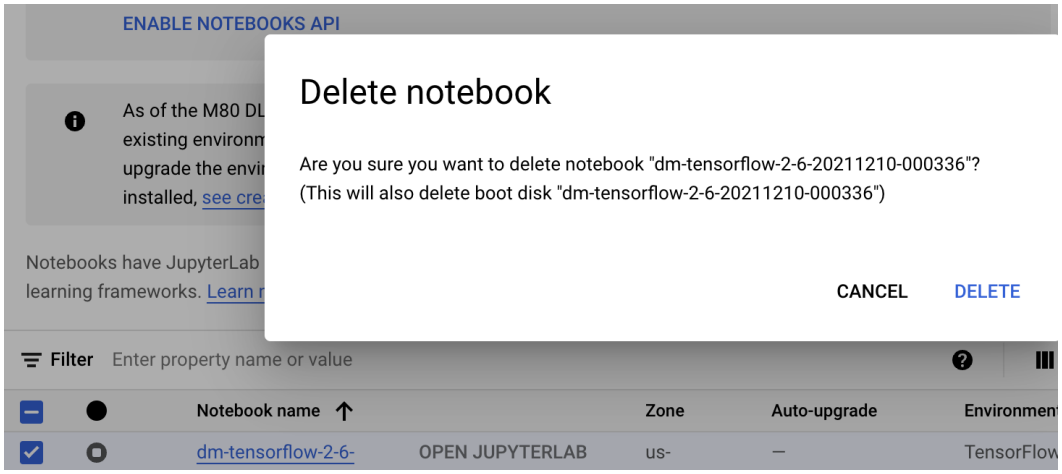
```
[3]: test_mpg = [1.4838871833555929,
 1.8659883497083019,
 2.234620276849616,
 1.0187816540094903,
 -2.530890710602246,
 -1.6046416850441676,
 -0.4651483719733302,
 -0.4952254087173721,
 0.7746763768735953]

response = endpoint.predict([test_mpg])

print('API response: ', response)

print('Predicted MPG: ', response.predictions[0][0])
```

```
API response:  Prediction(predictions=[[15.6667643]], deployed_model_id='661017594525908992', explanations=None)
Predicted MPG:  15.6667643
```

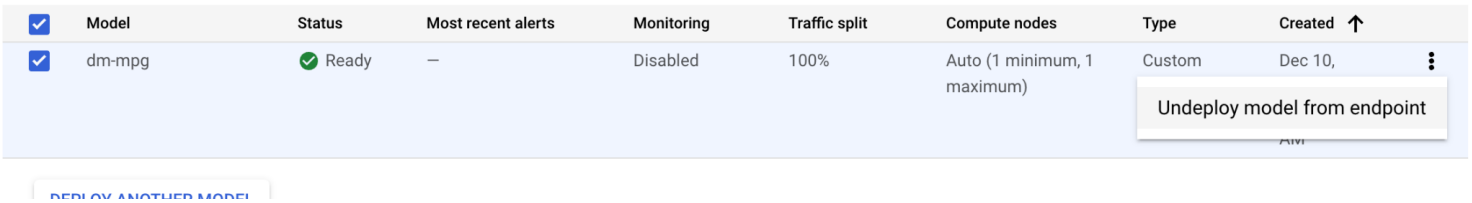
```
[ ]:
```

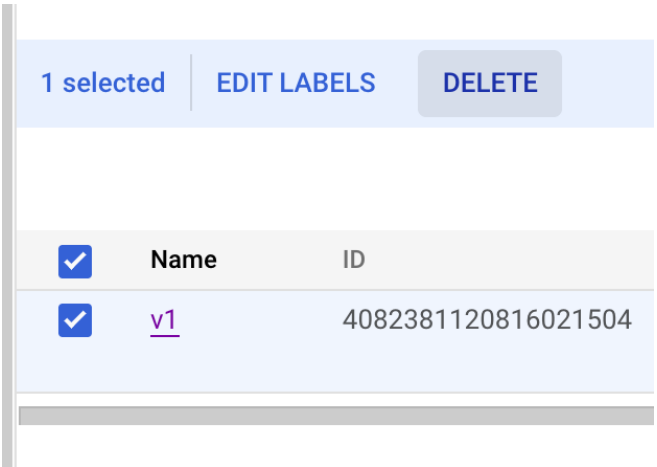
13. Clean UP

a) Stop the notebook from Vertex Workbench

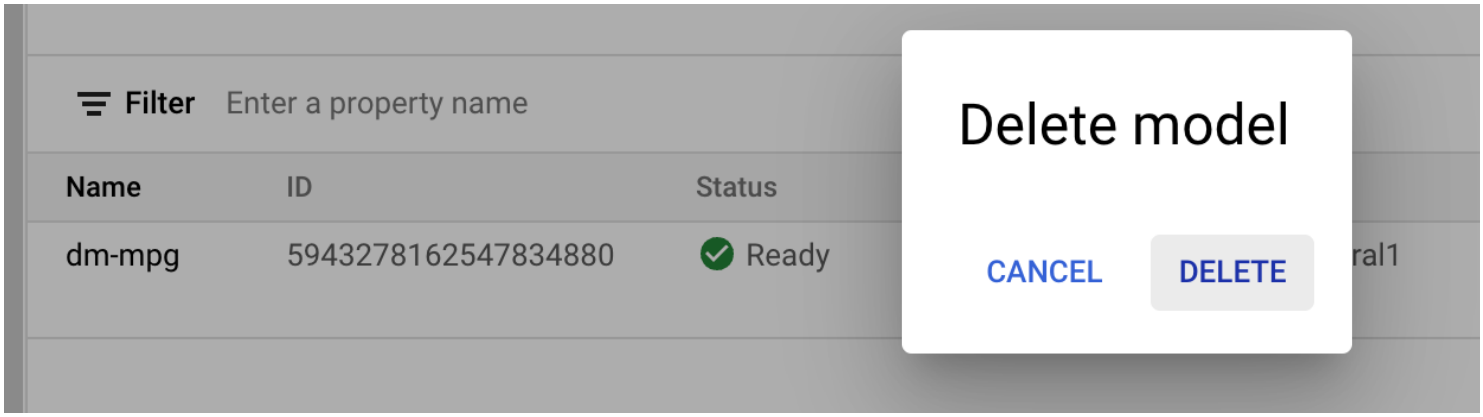


b) Undeploy model from endpoint

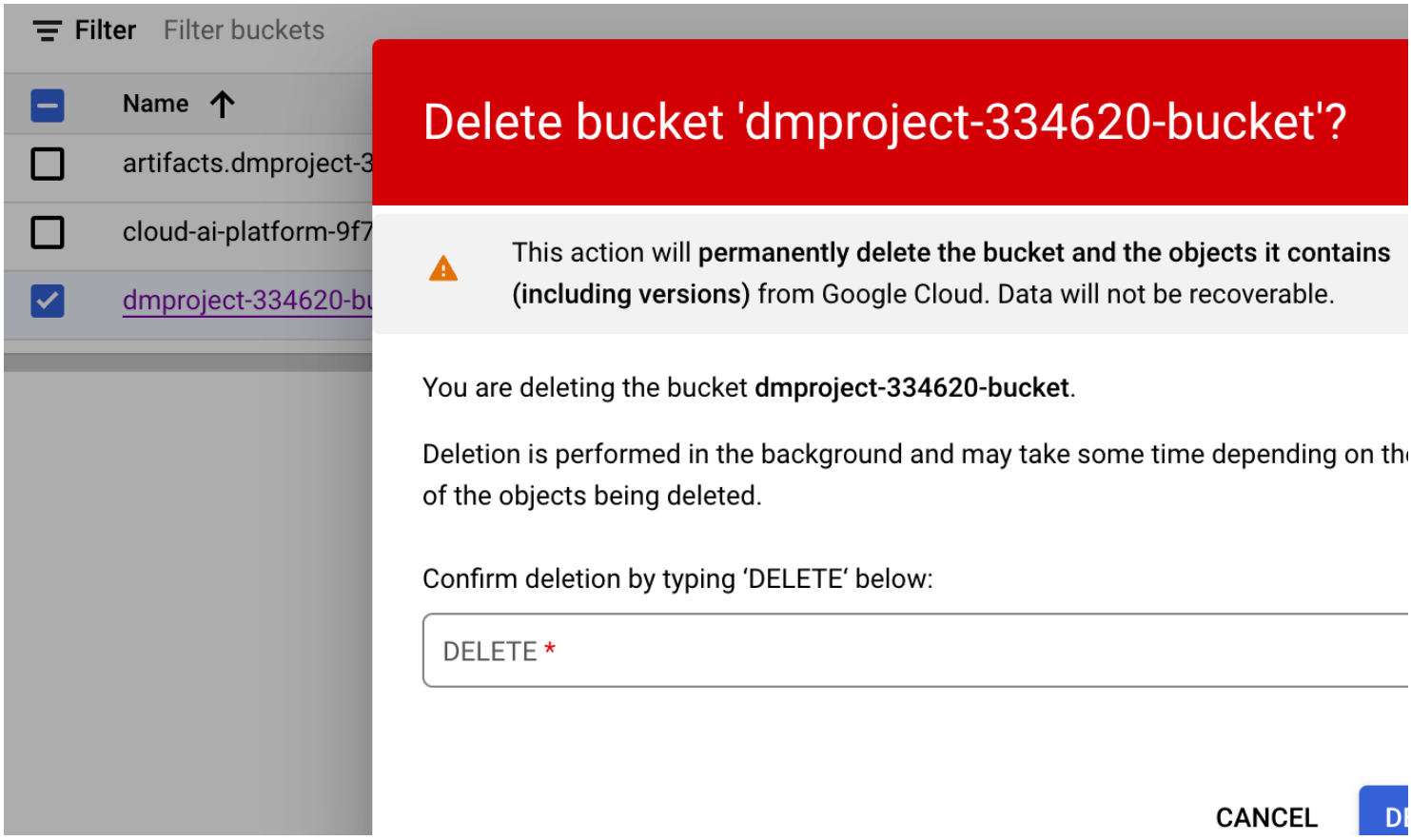




c) Delete the model



d) Delete storage



Vertex AI Workbench:Train a TensorFlow model with data from BigQuery - UNABLE TO COMPLETE

Dataset: London Bicycles Hire

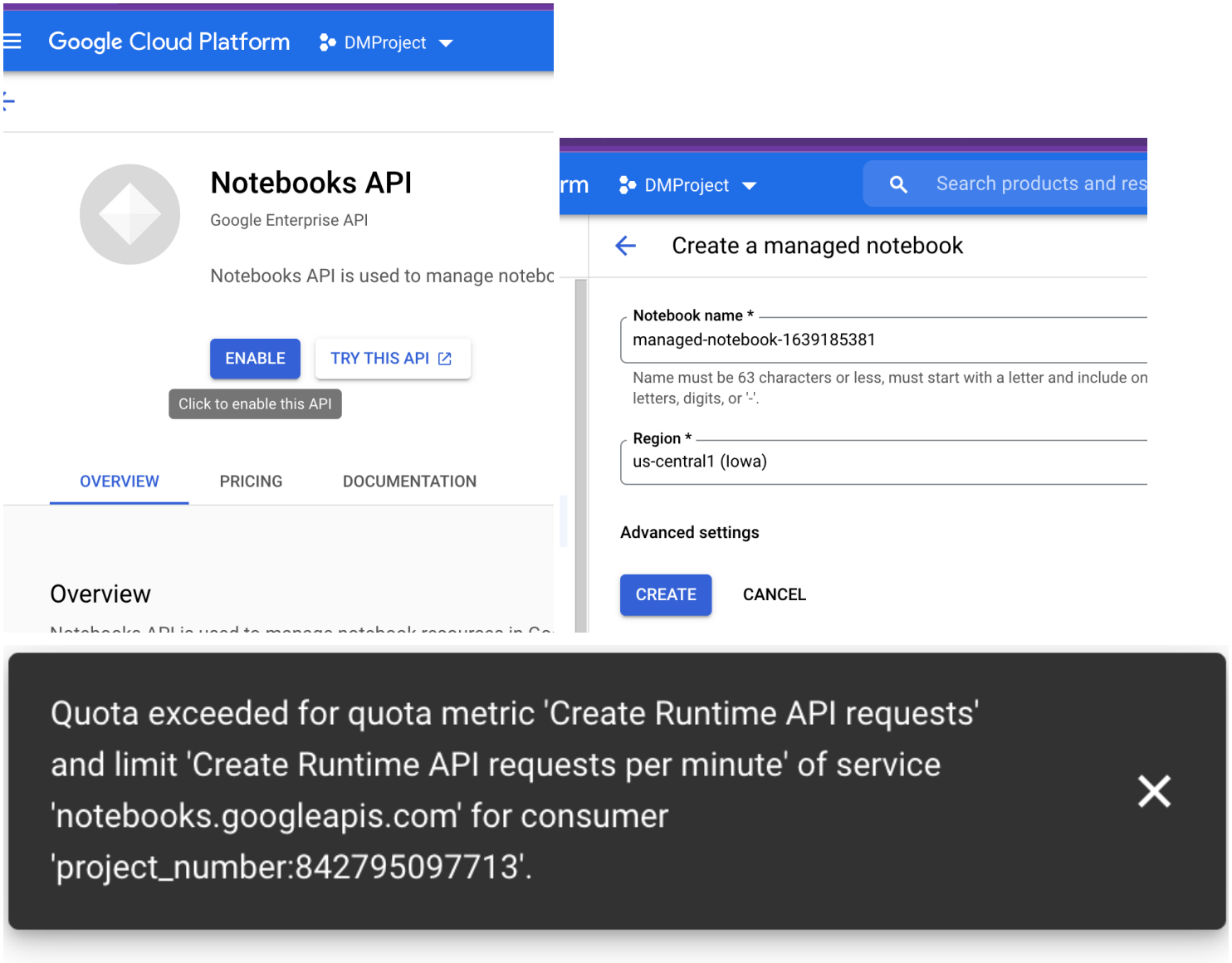
1. Enable Compute Engine

See above

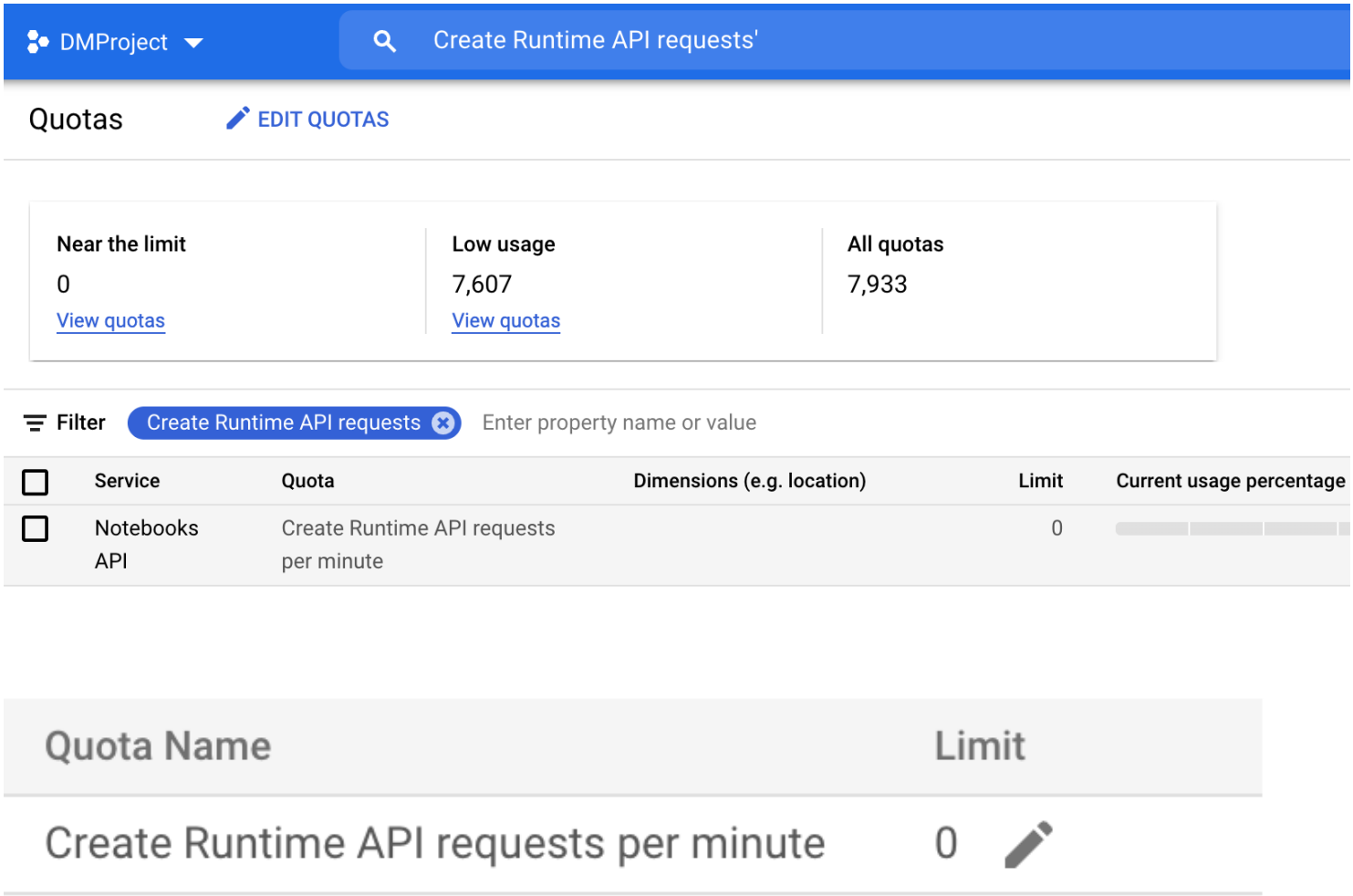
2. Enable the Vertex AI API

See above

3. Create a new managed notebook



4. Unable to update Runtime API requests per minute


DMProject
Create Runtime API requests'
Quotas
EDIT QUOTAS
Near the limit
0
View quotas
Low usage
7,607
View quotas
All quotas
7,933
Filter
Create Runtime API requests
Enter property name or value
Service
Quota
Dimensions (e.g. location)
Limit
Current usage percentage
Notebooks API
Create Runtime API requests per minute
0

Quota Name
Limit
Create Runtime API requests per minute
0