

# Intro to Vertex Pipelines

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

The screenshot shows the GCP Cloud Shell interface. On the left, there's a sidebar with 'Manage Resources' and a 'Release Notes' section. The main area is a terminal window titled 'Terminal (dmproject-334620)'. The terminal output is as follows:

```
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to dmproject-334620.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
anastasba@cloudshell:~ (dmproject-334620)$
```

3. Authorize the Shell

The screenshot shows the GCP Cloud Shell interface with a modal dialog box titled 'Authorize Cloud Shell'. The dialog contains the following text:

gcloud is requesting your credentials to make a GCP API call.  
Click to authorize this and future calls that require your credentials.

At the bottom right of the dialog are two buttons: 'REJECT' and 'AUTHORIZE'.

CLOUD SHELL

Terminal (dmproject-334620) X + ▾

```
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to dmproject-334620.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
anastasba@cloudshell:~ (dmproject-334620)$ gcloud auth list  
Credentialed Accounts  
  
ACTIVE: *  
ACCOUNT: anastasba@gmail.com  
  
To set the active account, run:  
  $ gcloud config set account `ACCOUNT`  
anastasba@cloudshell:~ (dmproject-334620)$
```

#### 4. Confirm that the gcloud command knows about your project

CLOUD SHELL

Terminal (dmproject-334620) X + ▾

```
anastasba@cloudshell:~ (dmproject-334620)$ gcloud config list project  
[core]  
project = dmproject-334620  
  
Your active configuration is: [cloudshell-25195]  
anastasba@cloudshell:~ (dmproject-334620)$ echo ${GOOGLE_CLOUD_PROJECT}  
dmproject-334620  
anastasba@cloudshell:~ (dmproject-334620)$
```

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

CLOUD SHELL

Terminal (dmproject-334620) X + ▾

```
anastasba@cloudshell:~ (dmproject-334620)$ gcloud services enable compute.googleapis.com  
Operation "operations/acf.p2-842795097713-954d9ef2-b2d2-4449-b1b0-917f86e15844" finished successfully  
anastasba@cloudshell:~ (dmproject-334620)$ gcloud services enable containerregistry.googleapis.com  
anastasba@cloudshell:~ (dmproject-334620)$ gcloud services enable aiplatform.googleapis.com  
anastasba@cloudshell:~ (dmproject-334620)$ gcloud services enable cloudbuild.googleapis.com  
anastasba@cloudshell:~ (dmproject-334620)$ gcloud services enable cloudfunctions.googleapis.com
```

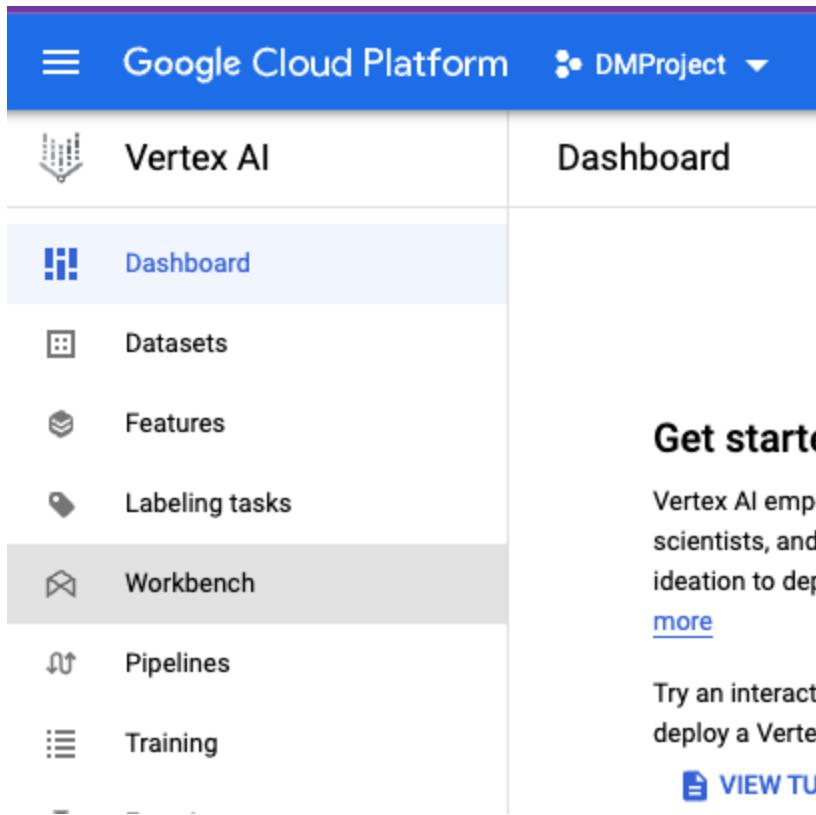
#### 6. Create Cloud Storage Bucket

```
anastasba@cloudshell:~ (dmproject-334620)$ BUCKET_NAME=gs://${GOOGLE_CLOUD_PROJECT}-bucket  
anastasba@cloudshell:~ (dmproject-334620)$ gsutil mb -l us-central1 $BUCKET_NAME  
Creating gs://dmproject-334620-bucket/...
```

#### 7. Give compute service account access to this bucket

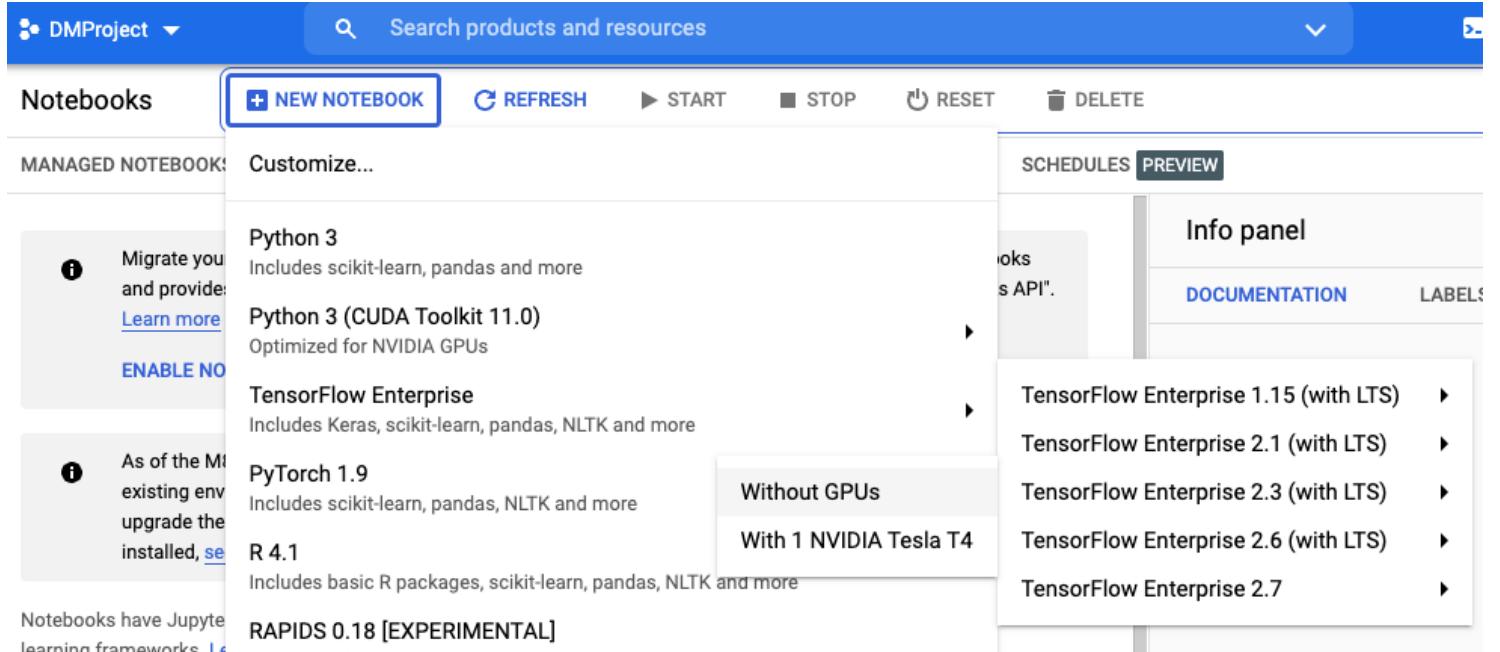
```
anastasba@cloudshell:~ (dmproject-334620)$ gcloud projects describe $GOOGLE_CLOUD_PROJECT > project-info.txt
anastasba@cloudshell:~ (dmproject-334620)$ PROJECT_NUM=$(cat project-info.txt | sed -nre 's::.*projectNumber\|: (.*):\l:p')
anastasba@cloudshell:~ (dmproject-334620)$ SVC_ACCOUNT="${PROJECT_NUM//\//}-compute@developer.gserviceaccount.com"
anastasba@cloudshell:~ (dmproject-334620)$ gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT --member serviceAccount:$SVC_ACCOUNT --role roles/editor
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



The screenshot shows the Google Cloud Platform Vertex AI dashboard. The top navigation bar includes the 'Google Cloud Platform' logo and a 'DMProject' dropdown. The left sidebar lists various options: Vertex AI (selected), Dashboard, Datasets, Features, Labeling tasks, Workbench (selected), Pipelines, Training, and a few other items partially visible. The main content area features a 'Get started' section with a brief introduction and a 'VIEW TUTORIAL' button. Below this is another section with a brief introduction and a 'VIEW TUTORIAL' button.

## 9. Create new Notebook (TensorFlow 2.3 Without GPUs)



The screenshot shows the Google Cloud Platform Notebooks page. The top navigation bar includes the 'DMProject' dropdown and a search bar. The main interface has tabs for 'Notebooks' (selected) and 'Customize...'. Below these are sections for 'MANAGED NOTEBOOKS' and 'NOTES'. The 'MANAGED NOTEBOOKS' section lists several options: Python 3 (with a note about scikit-learn, pandas, and more), Python 3 (CUDA Toolkit 11.0) (optimized for NVIDIA GPUs), TensorFlow Enterprise (includes Keras, scikit-learn, pandas, NLTK and more), PyTorch 1.9 (includes scikit-learn, pandas, NLTK and more), and R 4.1 (includes basic R packages, scikit-learn, pandas, NLTK and more). A note at the bottom states: 'Notebooks have Jupyter notebooks and support many learning frameworks. Learn more'. The 'NOTES' section shows a single item: 'RAPIDS 0.18 [EXPERIMENTAL]'. On the right side, there's a sidebar with tabs for 'SCHEDULES' (selected) and 'PREVIEW'. The 'PREVIEW' tab shows an 'Info panel' with 'DOCUMENTATION' and 'LABELS' tabs. A dropdown menu is open over the 'TensorFlow Enterprise' row, showing options: 'Without GPUs' (selected) and 'With 1 NVIDIA Tesla T4'.

## 10. Open Jupyterlab

Filter Enter property name or value					
<input type="checkbox"/>	<input checked="" type="radio"/>	Notebook name ↑	Zone	Au	
<input type="checkbox"/>	<input checked="" type="radio"/>	tensorflow-2-3-20211209-124041	OPEN JUPYTERLAB	us-west1-b	—

## 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__))"
```

```
[1]: import os

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

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

```
[1]: !python3 -c "import kfp; print('KFP SDK version: {}'
!python3 -c "import google_cloud_pipeline_components"
```

```
KFP SDK version: 1.8.9
google_cloud_pipeline_components version: 0.2.0
```

#### 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/conda/lib:/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

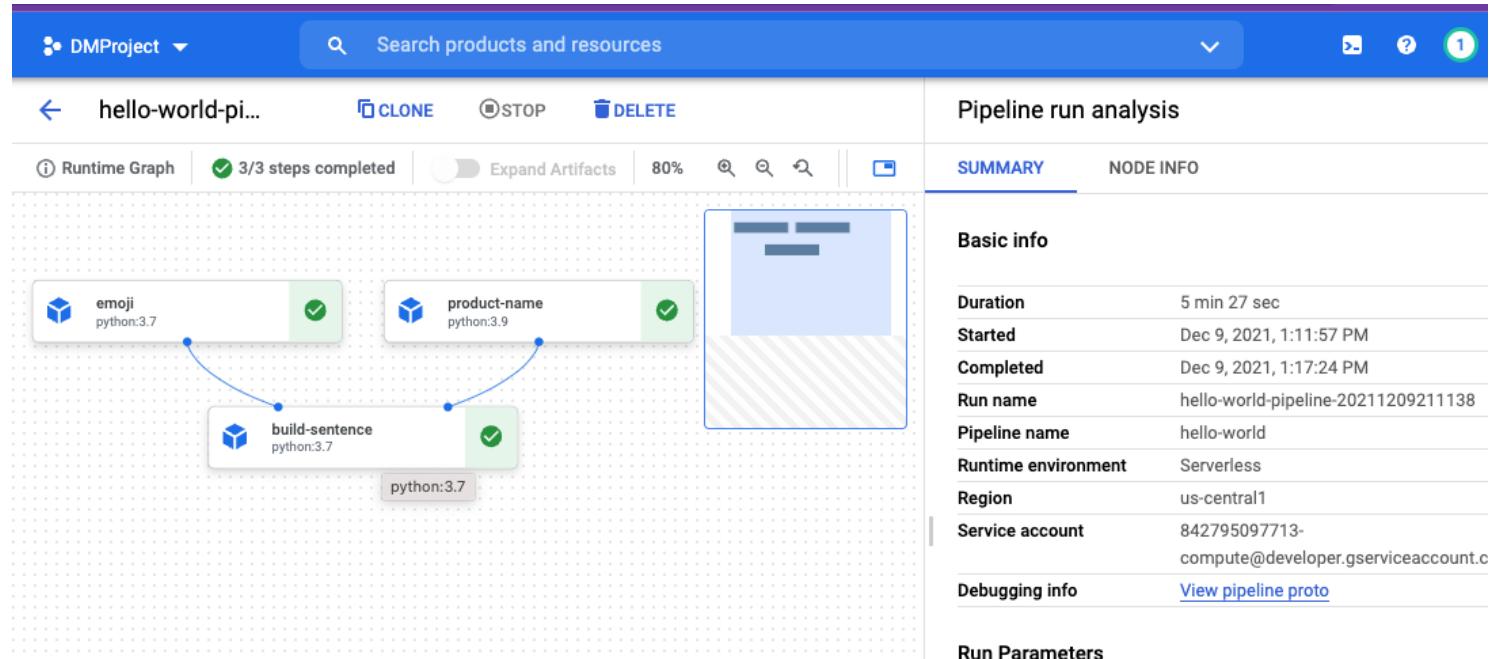
```
job.submit()
```

```
[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 Pipeline Job  
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob created  
all/pipelineJobs/hello-world-pipeline-20211209211138  
INFO:google.cloud.aiplatform.pipeline_jobs:To use this Pipeline Job  
INFO:google.cloud.aiplatform.pipeline_jobs:pipeline_job = https://us-central1/pipelineJobs/hello-world-pipeline-20211209211138  
INFO:google.cloud.aiplatform.pipeline_jobs:View Pipeline Job  
https://console.cloud.google.com/vertex-ai/locations/us-central1/jobs/842795097713
```



## Pipeline run analysis

SUMMARY

NODE INFO

### Execution Info

Completed

[VIEW JOB](#)

[VIEW LOGS](#)

Display name	build-sentence
Name	build-sentence
Type	system.ContainerExecution
Duration	2 min 42 sec
Started	Dec 9, 2021, 1:14:42 PM
Completed	Dec 9, 2021, 1:17:24 PM

### Input Parameters

Parameter	Type	Value
emojitext	string	sparkles
product	string	Vertex Pipelines
emoji	string	:)

### Output Parameters

Parameter	Type	Value
Output	string	Vertex Pipelines is :)

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 prebuilt 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.com",
    thresholds_dict_str: str = '{"auRoc": 0.95}',
):
    dataset_create_op = gcc_aip.TabularDatasetCreateOp(
        project=project, display_name=display_name, bq_sourc
    )

    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

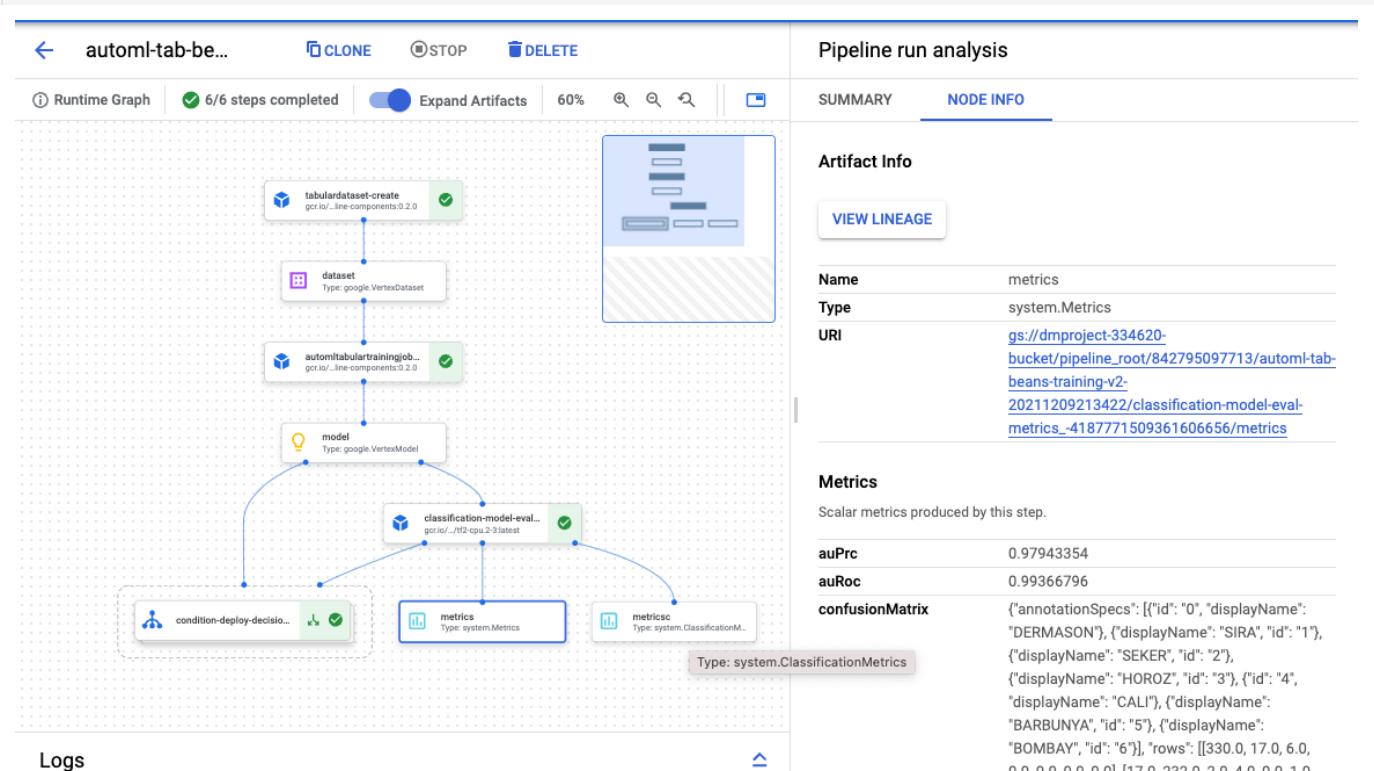
```
ml_pipeline_job = aiplatform.PipelineJob(
    display_name="automl-tab-beans-training",
    template_path="tab_classif_pipeline.json",
    pipeline_root=PIPELINE_ROOT,
    parameter_values={"project": PROJECT_ID, "display_name": DISPLAY_NAME},
    enable_caching=True
)
```

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 

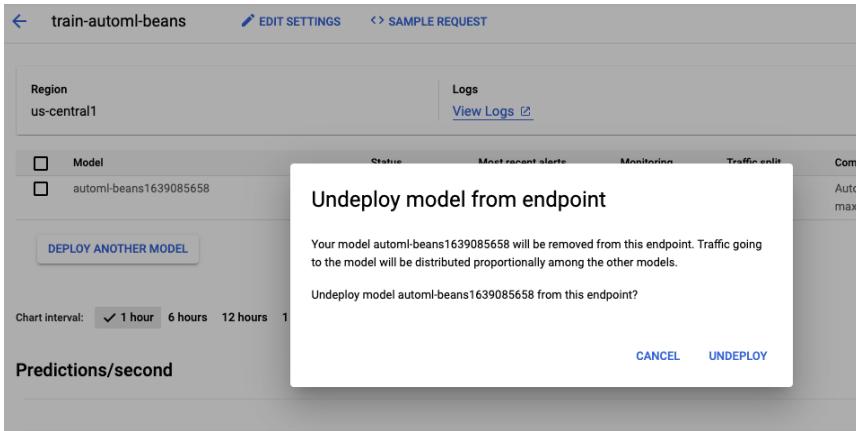
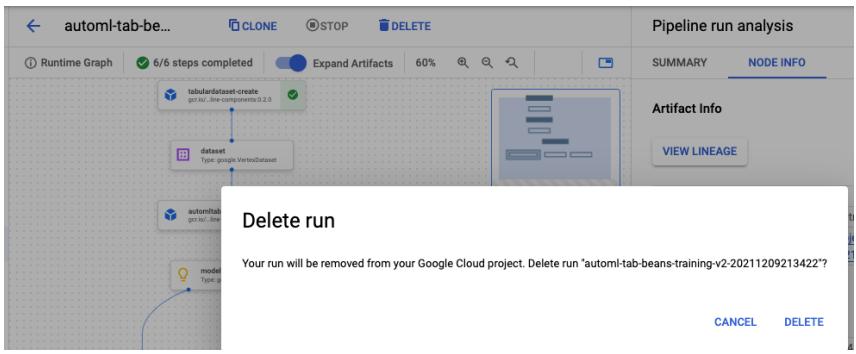
This table shows how often the model classified each label correctly (in blue), and which labels were most often confused for that label (in gray).

True label	Predicted label							
	DERMASON	SIRA	SEKER	HOROZ	CALI	BARBUNYA	BOMBAY	
DERMASON	93%	5%	2%	—	—	—	—	—
SIRA	7%	91%	1%	2%	—	0%	—	—
SEKER	1%	3%	96%	—	—	—	—	—
HOROZ	1%	3%	—	96%	1%	—	—	—
CALI	—	1%	—	3%	95%	2%	—	—
BARBUNYA	—	4%	1%	—	5%	90%	—	—
BOMBAY	—	—	—	—	—	—	100%	—

Confusion matrix

### g) Comparing metrics across pipeline runs

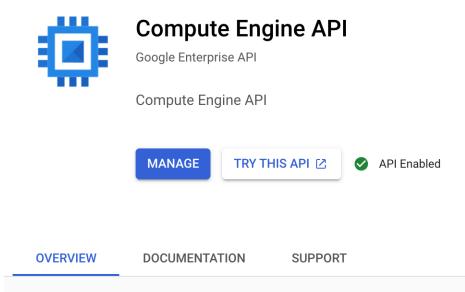
[22]:	pipeline_df = aiplatform.get_pipeline_df(pipeline="automl-tab-beans-training-v2") small_pipeline_df = pipeline_df.head(2) small_pipeline_df																								
[22]:	<table border="1"> <thead> <tr> <th>pipeline_name</th> <th>run_name</th> <th>param.input:thresholds_dict_str</th> <th>param.input:gcp_region</th> <th>param.input:api_endpoint</th> <th>param.input:display_name</th> <th>param.input:bq_source</th> <th>param.input:project_id</th> </tr> </thead> <tbody> <tr> <td>automl-tab-beans-training-v2</td> <td>automl-tab-beans-training-v2-20211209213422</td> <td>{"auRoc": 0.95}</td> <td>us-central1</td> <td>us-central1-aiplatform.googleapis.com</td> <td>automl-beans1639085658</td> <td>bq://aju-dev-demos.beans.beans1</td> <td>dmproject-3346</td> </tr> <tr> <td>automl-tab-beans-training-v2</td> <td>automl-tab-beans-training-v2-20211209212820</td> <td>{"auRoc": 0.95}</td> <td>us-central1</td> <td>us-central1-aiplatform.googleapis.com</td> <td>automl-beans1639085123</td> <td>bq://aju-dev-demos.beans.beans1</td> <td>dmproject-3346</td> </tr> </tbody> </table>	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:project_id	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	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
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:project_id																		
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																		
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																		
[1]:																									



# 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.

The screenshot shows the Vertex AI dashboard. On the left, there's a sidebar with links like Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, Metadata, and Marketplace. The main area has a section titled 'Get started with Vertex AI' with a link to 'VIEW TUTORIALS'. Below that is a 'Region' dropdown set to 'us-central1 (Iowa)'. A 'Recent datasets' card shows a single entry: 'automl-beans1639085658' created '9 hours ago'. At the bottom of this card is a '+ CREATE DATASET' button.

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

The screenshot shows the 'Notebooks' section of the Vertex AI Workbench interface. It includes tabs for 'MANAGED NOTEBOOKS' and 'CUSTOM'. Under 'MANAGED NOTEBOOKS', there are several options: Python 3 (includes scikit-learn, pandas), Python 3 (CUDA Toolkit 11.0) (optimized for NVIDIA GPUs), TensorFlow Enterprise (includes Keras, scikit-learn, pandas, NLTK), PyTorch 1.9 (includes scikit-learn, pandas, NLTK), R 4.1 (includes basic R packages), RAPIDS 0.18 [EXPERIMENTAL] (optimized for NVIDIA GPUs), and Kaggle Python [BETA]. On the right, there's a 'SCHEDULES' tab and a 'PREVIEW' panel. The 'PREVIEW' panel shows 'TensorFlow Enterprise 1.15 (with LTS)', 'TensorFlow Enterprise 2.1 (with LTS)', 'TensorFlow Enterprise 2.3 (with LTS)', 'TensorFlow Enterprise 2.6 (with LTS)', and 'TensorFlow Enterprise 2.7'. There are also tabs for 'Info panel', 'DOCUMENTATION', and 'LABELS'.

## 4. Open Notebook

The screenshot shows a list of notebooks. There are two entries: 'tensorflow-2-6-' and '20211209-233831'. The first notebook has a 'Notebook name' column with a radio button next to it, and an 'OPEN JUPYTERLAB' button. The second notebook has a checked checkbox in the same column. Other columns include 'Zone' (set to 'us-west1-b'), 'Auto-upgrade' (set to '-'), and 'Environment' (set to 'TensorFlow:2.6'). There are also filter and search tools at the top.

## 5. Create a dataset

DMProject Search products

[Create dataset](#)

Dataset name \*  Can use up to 128 characters.

### Select a data type and objective

First select the type of data your dataset will contain. Then

Google Cloud Platform DMProject

Vertex AI Datasets + CREATE

Dashboard Datasets

Region us-central1 (Iowa)

Filter Enter a property name

Name automl-beans1639085658

IMAGE TABULAR TEXT VIDEO

Regression/classification Predict a target column's value. Supports tables with hundreds of columns and millions of rows.

## 6. Select the datasource: BigQuery

bigrquery-public-data.ml\_datasets.ulb\_fraud\_detection

**DMProject** ▾ Search products and resources

**fraud\_detection\_dm\_az**

**SOURCE** **ANALYZE**

**Add data to your dataset**

Before you begin, read the [data guide](#) to learn how to prepare your data. Then choose a data source.

**Select a data source**

- CSV file: Can be uploaded from your computer or on Cloud Storage. [Learn more](#)
- BigQuery: Select a table or view from BigQuery. [Learn more](#)

Upload CSV files from your computer

Select CSV files from Cloud Storage

Select a table or view from BigQuery

**Select a table or view from BigQuery**

Use a BigQuery table or view as your data source. You'll need [permission to access the dataset](#) and get the [dataset ID and table ID](#). [Learn more](#)

BigQuery path \*  bq://bigquery-public-data.ml\_datasets.ulb\_fraud\_detection BROWSE ?

Enter the qualified Id: projectID.datasetID.tableID

**7. Train a model with AutoML**

**Train new model**

**Training jobs and models**

Use this dataset and annotation set to train a new machine learning model with AutoML or custom code

**TRAIN NEW MODEL**

**Train new model**

**1 Training method**

**2 Model details**

**3 Training options**

**4 Compute and pricing**

**START TRAINING** **CANCEL**

**Dataset**  
fraud\_detection\_dm\_az

**Objective \***  
Classification

Please refer to the pricing guide for more details about each method.

**AutoML**  
Train high-quality models with minimal effort and let AutoML handle the heavy lifting. [Learn more](#)

**Custom training (advanced)**  
Run your TensorFlow, scikit-learn, and XGBoost code in one of Google Cloud's pre-built containers or use your own.

**CONTINUE**

**8. Select Target column (Class for this dataset)**

**Train new model**

**Training method**

**Model details**

**Training options**

**Compute and pricing**

**START TRAINING** **CANCEL**

**Model name \***  
fraud\_detection\_dm\_az\_202112107487 ?

**Target column \***  
Class (INTEGER) ▼ ?

Export test dataset to BigQuery

**ADVANCED OPTIONS**

**CONTINUE**

**9. Add Optimization Objective (AUC PRC)**

## Train new model

- Training method
- Model details
- 3 Training options
- 4 Compute and pricing

START TRAINING

CANCEL

Total 31 feature columns are included in the training

### Weight column

Select a column ▾

### Optimization objective

- AUC ROC  
Distinguish between classes
- Log loss  
Keeps prediction probabilities as accurate as possible
- AUC PRC  
Maximize precision-recall for the less common class
- Precision      At recall
- Recall      At precision

10. Start training and wait until the training is done

## Training jobs and models



[fraud\\_detection\\_dm\\_az\\_202112107487](#)

Training model...

[TRAIN NEW MODEL](#)

fraud\_detection\_dm\_az\_202112107487

6788561811261095936

Training

Training

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	-

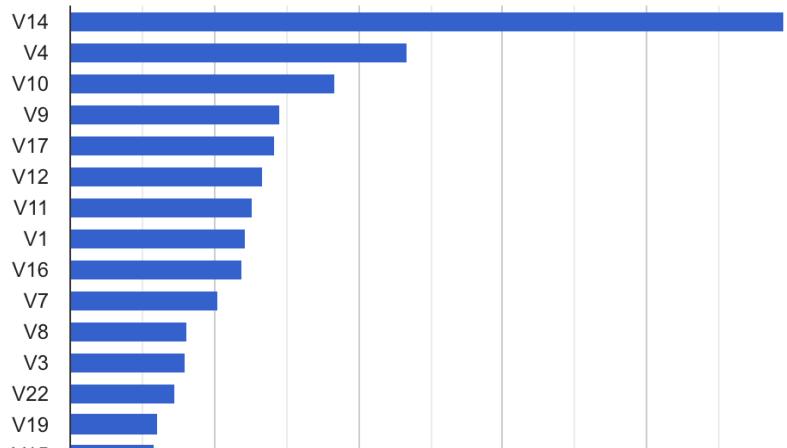
## 11. Explore model evaluation metrics

### Confusion matrix

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

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

## Feature Importance



## 12. Deploy the model to 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 to endpoint

- ① Define your endpoint
- ② Model settings
- ③ Model monitoring

DEPLOY

CANCEL

Create new endpoint    Add to existing endpoint  
Endpoint name \*

#### Location

Region us-central1 (Iowa) 

#### Access

Determines how your endpoint can be accessed for prediction serving through a REST API after the endpoint is created.

##### Standard

Makes the endpoint available for predictions. Custom-trained models can be added to it.

##### Private

Create a private connection to this endpoint. Only custom-trained and tabular models can be added to it.  
[Learn more](#)

#### ▼ ADVANCED OPTIONS

CONTINUE

### Deploy your model

Endpoints are machine learning models made available for online predictions. They are useful for timely predictions from many users (for example, in response to a request). You can also request batch predictions if you don't need immediate results.

**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 <b>HIDE ENDPOINTS</b> fraud_v1	1	us-central1

**Filter** Enter a property name

<input type="checkbox"/>	Name	ID	Status	Models	Region	Monitoring
<input type="checkbox"/>	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']}], deployed\_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
<a href="#">fraud_v1</a>	9164904406536159232	<span>Active</span>	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	<input type="text" value="84871"/>	0
V1	Numerical	Required	<input type="text" value="0.0202732068951347"/>	0
V2	Numerical	Required	<input type="text" value="0.0657600286408065"/>	0
V3	Numerical	Required	<input type="text"/>	0

Predict label

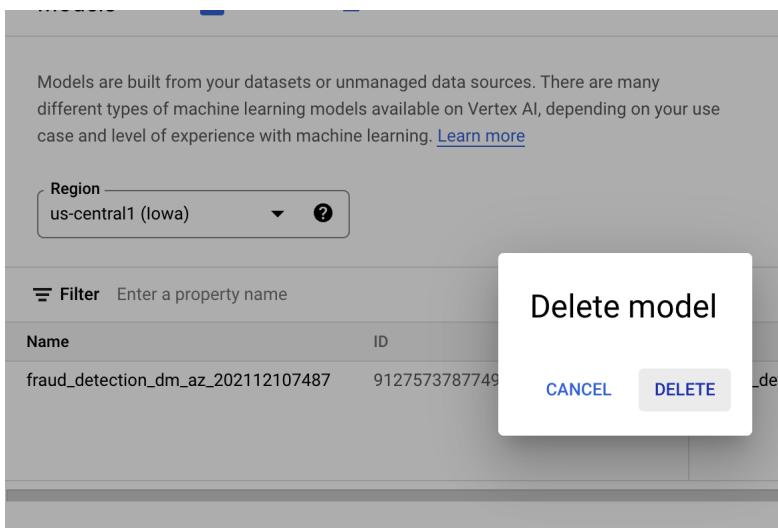
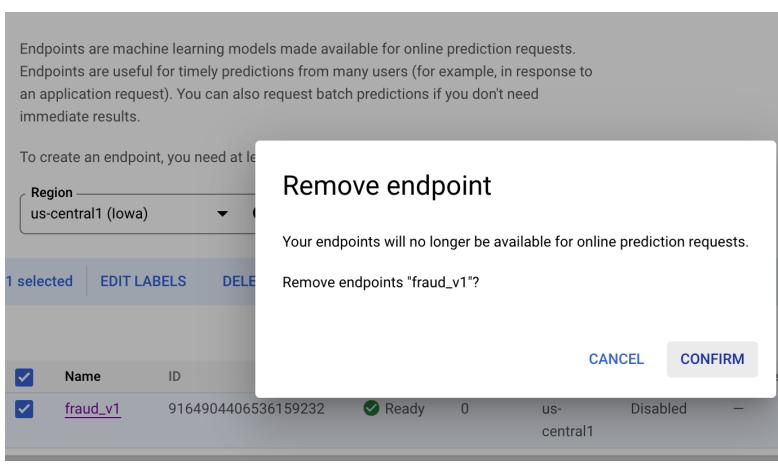
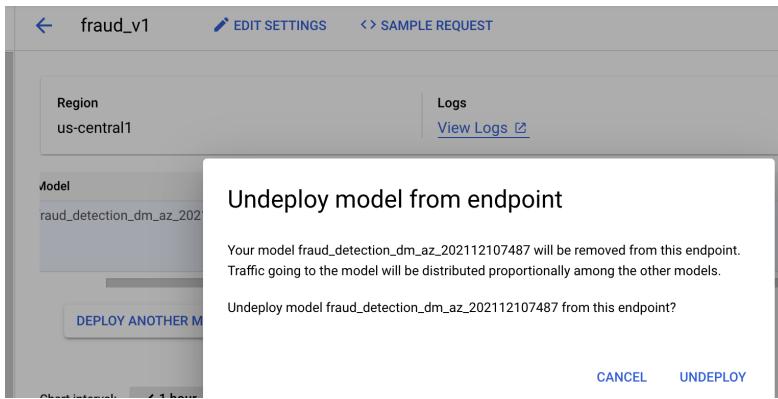
Prediction result

Selected label —

Baseline prediction value: 0.9934428334236145

Confidence score: 0.9934428334236145

## 16. CleanUP



The screenshot shows the Google Cloud Platform Notebooks interface. At the top, there are buttons for NEW NOTEBOOK, REFRESH, START, STOP, RESET, and DELETE. Below these are tabs for MANAGED NOTEBOOKS (PREVIEW), USER-MANAGED NOTEBOOKS (selected), EXECUTIONS (PREVIEW), and SCHEDULES. A prominent message box titled "Delete notebook" asks if the user is sure they want to delete the notebook "tensorflow-2-6-20211209-233831". The message also states that this will delete the boot disk "tensorflow-2-6-20211209-233831". There are "CANCEL" and "DELETE" buttons at the bottom of the dialog. In the background, a table lists notebooks. One row is selected, showing "tensorflow-2-6-20211209-233831" in the Notebook name column, "OPEN JUPYTERLAB" in the Zone column, and "TensorFlow:2.6" in the Environment column.

# Training and serving a custom model

1. Enable Compute Engine

See above

2. Enable Vertex AI API

See above

3. Enable Container Registry API

The screenshot shows the Google Cloud Platform Container Registry interface. At the top, there is a navigation bar with "Google Cloud Platform" and "DMProject". Below the navigation bar, there is a back arrow icon. The main area features a logo of three blue hexagons and the text "Google Container Registry". It says "Google Enterprise API" and provides a brief description: "Google Container Registry provides secure storage for container images across Google Cloud Platform. ...". There are "MANAGE" and "API Enabled" buttons at the bottom. The status "API Enabled" has a green checkmark.

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

The screenshot shows the Google Cloud Platform Notebooks interface. At the top, there is a filter bar with "Enter property name or value" and a search icon. Below the filter bar, a table lists notebooks. One row is selected, showing "dm-tensorflow-2-6-20211210-000336" in the Notebook name column, "OPEN JUPYTERLAB" in the Zone column, and "TensorFlow:2.6" in the Environment column.

5. Containerize training code  
a) Start a new terminal session



Notebook



Python 3



Python [conda  
env:root] \*



Console



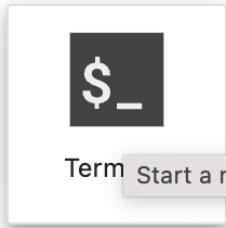
Python 3



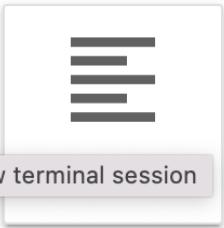
Python [conda  
env:root] \*



Other



Term



Start a new terminal session



Markdo

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

\$ jupyter@dm-tensorflow-2- X

```
jupyter@dm-tensorflow-2-6-20211210-000336:~$ mkdir mpg
jupyter@dm-tensorflow-2-6-20211210-000336:~$ cd mpg
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$
```

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
```

Filter files by name



/ mpg / trainer /

Name	Last Modified
train.py	seconds ago

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
7bla6ab2e44d: 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_CPIITNFO_FTI_E: 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 $IMAGE_URI
The push refers to repository [gcr.io/dmproject-334620/mpg]
70cedac4141e: Pushed
2699358e44b2: Mounted from deeplearning-platform-release/tf2-cpu.2-6
393bde4f6bc5: Mounted from deeplearning-platform-release/tf2-cpu.2-6
cf3738cf85bb: Mounted from deeplearning-platform-release/tf2-cpu.2-6
1b3c3815787f: Mounted from deeplearning-platform-release/tf2-cpu.2-6
244680cd2339: Mounted from deeplearning-platform-release/tf2-cpu.2-6
2d3b6e42c63d: Mounted from deeplearning-platform-release/tf2-cpu.2-6
b538f5dfbd26: Mounted from deeplearning-platform-release/tf2-cpu.2-6
539edf6b8fe8: Mounted from deeplearning-platform-release/tf2-cpu.2-6
f212849c96cc: Mounted from deeplearning-platform-release/tf2-cpu.2-6
7121d6cc04ae: Mounted from deeplearning-platform-release/tf2-cpu.2-6
c787e4bdd37e: Mounted from deeplearning-platform-release/tf2-cpu.2-6
9f6f8dd09608: Mounted from deeplearning-platform-release/tf2-cpu.2-6
9feb64009273: Mounted from deeplearning-platform-release/tf2-cpu.2-6
b69e9b09c4c0: Mounted from deeplearning-platform-release/tf2-cpu.2-6
54363f143f4d: Mounted from deeplearning-platform-release/tf2-cpu.2-6
5f70bf18a086: Layer already exists
b3b3550ec84e: Mounted from deeplearning-platform-release/tf2-cpu.2-6
48515e7cf8a8: Mounted from deeplearning-platform-release/tf2-cpu.2-6
82b8c1c41f58: Mounted from deeplearning-platform-release/tf2-cpu.2-6
9f54eef41275: Layer already exists
v1: digest: sha256:26b36cd07c0286d1fb2f4ecb41a3a543c70d6d6189061c585e55c
jupyter@dm-tensorflow-2-6-20211210-000336:~/mpg$ █
```

## 6. Run a training job on VertexAI

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

## Train new model

- 1 Training method
- 2 Model details
- 3 Training container
- 4 Hyperparameters (optional)
- 5 Compute and pricing
- 6 Prediction container (optional)

START TRAINING

CANCEL

Dataset \* \_\_\_\_\_  
No managed dataset

Annotation set \_\_\_\_\_  
-

Objective \_\_\_\_\_  
Custom

Please refer to the pricing guide for more information about each method.

 AutoML options are only available for certain training methods.

- AutoML  
Train high-quality models with minimal effort and specify how long you want to train. [Learn more](#)
- AutoML Edge  
Train a model that can be exported for or deployed on edge devices with high accuracy. [Learn more](#)
- Custom training (advanced)  
Run your TensorFlow, scikit-learn, and XGBoost code in one of Google Cloud's pre-built containers.

CONTINUE

## 7. Select model name

## Train new model

- Training method
- 2 Model details
- 3 Training container

Model name \* \_\_\_\_\_  
dm-mpg

 ADVANCED OPTIONS

CONTINUE

## 8. Select custom container image

**Train new model**

- Training method
- Model details
- Training container
- Hyperparameters (optional)
- Compute and pricing
- Prediction container (optional)

**START TRAINING      CANCEL**

Select a pre-built container or build a custom one. Add non-ML dependencies, libraries and binaries [more](#)

Pre-built container  
View the list of [supported runtimes](#) including

Custom container  
Build a custom Docker container. Must be a valid Dockerfile.

**Custom container settings**

Container image \*

! Container image URL is required.

gs:// Model output directory

Your model artifacts and other data needed for training will be stored here.

## Select container image

[CONTAINER REGISTRY](#) [ARTIFACTS](#)

Project: dmproject-334620 [CHANGE](#)

▼ gcr.io/dmproject-334620/mpg

26b36cd07c v1

**SELECT**

CANCEL

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

**Train new model**

- Training method
- Model details
- Training container
- Hyperparameters (optional)
- Compute and pricing
- Prediction container (optional)

**START TRAINING      CANCEL**

Model training pricing is based on the length of time and any accelerators used. [Learn more](#)

Region  ▼ ?

**Compute settings**

Select the type of virtual machine to use for your work. To learn about compute costs and how to map your ML worker pools, consult the [documentation](#)

**Worker pool 0**

Machine type \*

## 10. Select Pre-built container and start building

## Train new model

- Training method
- Model details
- Training container
- Hyperparameters (optional)
- Compute and pricing
- Prediction container (optional)

**START TRAINING**

CANCEL

You can always import your model artifact later to serve prediction requests

### Pre-built container

View the list of [supported runtimes](#) including TensorFlow, scikit-learn and PyTorch versions

### Custom container

Build a custom Docker container. Must be stored in [Container Registry or Artifact Registry](#)

### Pre-built container settings

Vertex AI provides Docker container images for serving predictions. To use a pre-built container, your trained model code must be in Python 3.7. [Learn more about pre-built containers](#)

In order to run in a pre-built container, your code needs to be in Python 3.7

Model framework *	TensorFlow
Model framework version *	2.6
Accelerator type *	None
Model directory *	<input type="text" value="gs://dmproject-334620-bucket/mpg"/> <b>BROWSE</b>

Cloud Storage location containing the model artifact and any supporting files

**Filter** Enter a property name

Name	ID	Status	Job type	Model type	Created
dm-mpg	4129186231298818048	<span>Training</span>	Training pipeline	<span>Custom</span>	Dec 10, 2021, 12:44:17 AM

## 11. Deploy the endpoint

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

**Filter** Enter a property name

Name	ID	Status	Data	Endpoints	Region	Type	Created	Notifications	Labels
dm-mpg	5943278162547834880	<span>Ready</span>	—	0	us-central1	<span>Custom trained</span> Custom training	Dec 10, 2021, 12:44:17 AM		<span>Add to endpoint</span>

### b) Provide endpoint the name

#### Deploy to endpoint

- 1** Define your endpoint
- 2** Model settings
- 3** Model monitoring

##### Create new endpoint Add to existing endpoint

**Endpoint name \***

v1

##### Location

Region

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

## Deploy to endpoint

Define your endpoint

**2** Model settings

**3** Model monitoring

**DEPLOY**

CANCEL

### dm-mpg

Traffic split \*

100

### Compute resources

Choose how compute resources will serve prediction traffic to

- **Autoscaling:** If you set a minimum and maximum, compute resources will scale to meet traffic demand within those boundaries
- **No scaling:** If you only set a minimum, then that number of nodes will always run regardless of traffic demand (the maximum will be set to minimum)

Once scaling settings are set, they can't be changed unless you delete the model. [Pricing guide](#)

Minimum number of compute nodes \*

1

Default is 1. If set to 1 or more, then compute resources will continue to run even without traffic demand. This can increase cost but avoid dropped requests due to node initialization.

Maximum number of compute nodes (optional)

Enter a number equal to or greater than the minimum nodes. Can increase costs but may cause reliability issues for high traffic.

### ▼ ADVANCED SCALING OPTIONS

Machine type \*

n1-standard-2, 2 vCPUs, 7.5 GiB memory

**Filter** Enter a property name

Name	ID	Status	Data	Endpoints
dm-mpg	5943278162547834880	<input checked="" type="checkbox"/> Ready	—	1 <b>HIDE ENDPOINTS</b> v1

**Filter** Enter a property name

<input type="checkbox"/>	Name	ID	Status	Models	Region	Monitoring	Logs
<input type="checkbox"/>	v1	4082381120816021504	<input checked="" type="checkbox"/> Active	0	us-central1	Disabled	—

## 12. Get predictions on the deployed model

- 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 (1.1  
    |██████████| 1.7 MB 8.0 MB/s
```

```
Requirement already satisfied: google-api-core[grpc]<3.0.0dev,>=1.26.  
google-cloud-aiplatform) (2.2.2)
```

```
Requirement already satisfied: packaging>=14.3 in /opt/conda/lib/python  
(21.3)
```

```
Requirement already satisfied: proto-plus>=1.10.1 in /opt/conda/lib/python  
(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"  
)
```

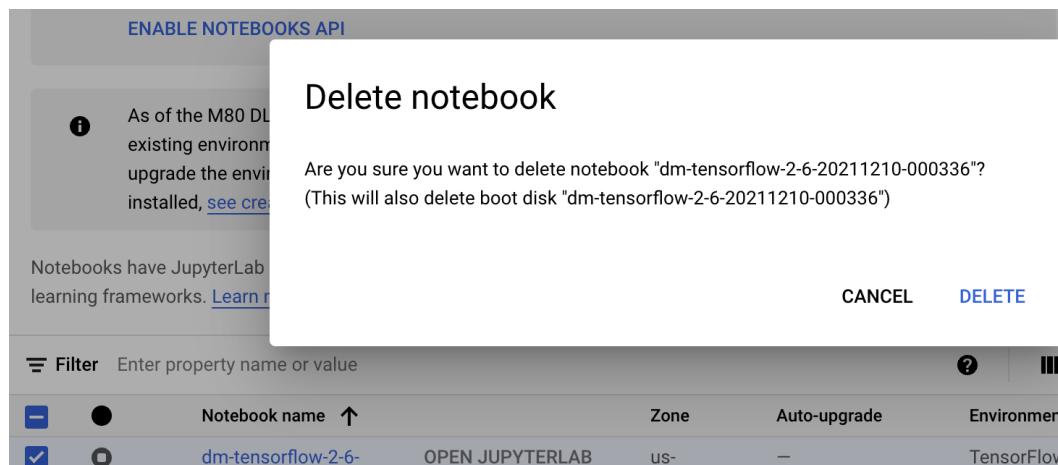
c) Make a prediction

```
[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

Model	Status	Most recent alerts	Monitoring	Traffic split	Compute nodes	Type	Created	⋮
dm-mpg	Ready	—	Disabled	100%	Auto (1 minimum, 1 maximum)	Custom	Dec 10,	<b>Undeploy model from endpoint</b>

1 selected    EDIT LABELS    **DELETE**

Name	ID
v1	4082381120816021504

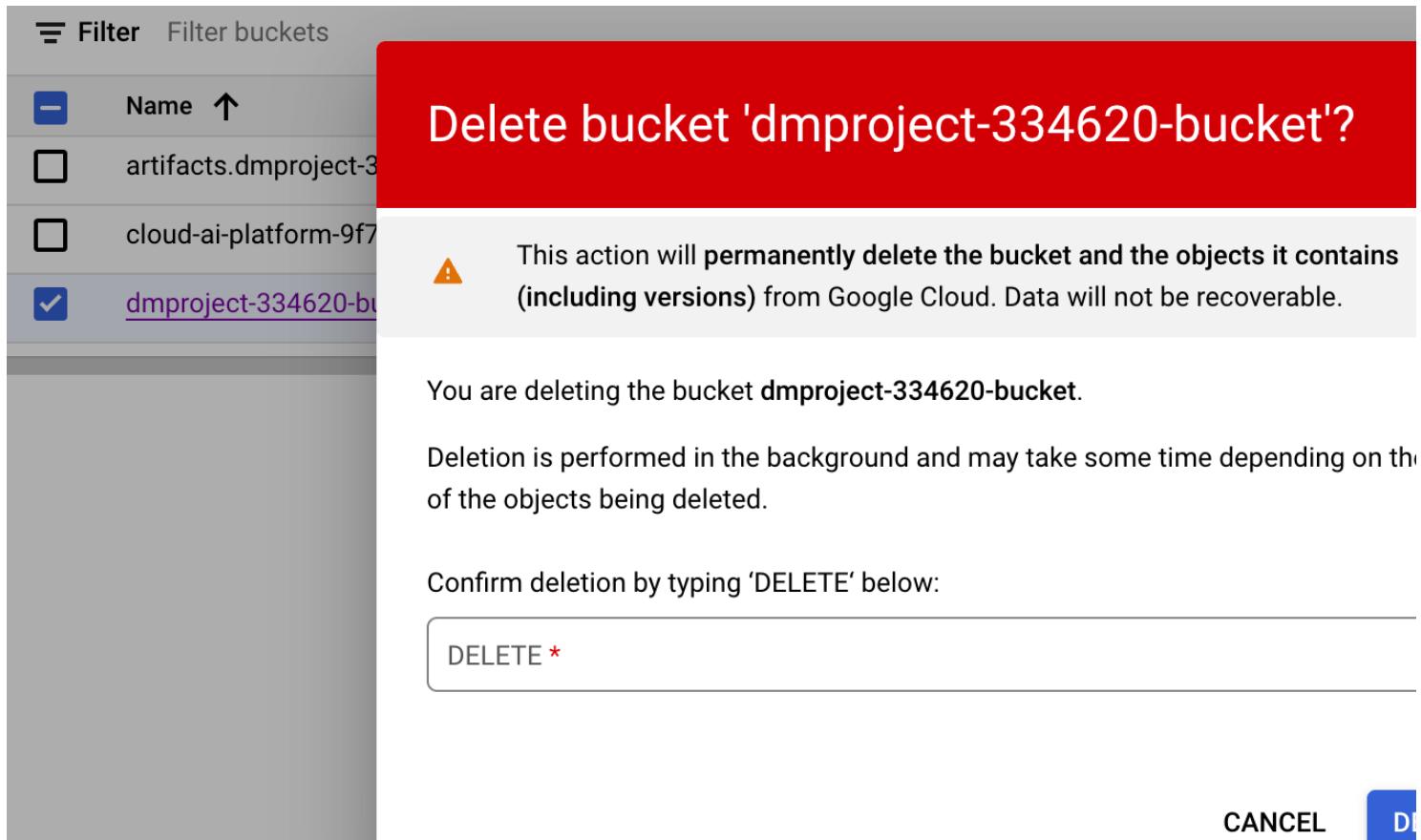
### c) Delete the model

Name	ID	Status
dm-mpg	5943278162547834880	Ready

**Delete model**

CANCEL    **DELETE**

### 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



**Notebooks API**  
Google Enterprise API

Notebooks API is used to manage notebooks.

**ENABLE** TRY THIS API ↗  
Click to enable this API

**OVERVIEW** PRICING DOCUMENTATION

Search products and resources

Create a managed notebook

Notebook name \* — managed-notebook-1639185381  
Name must be 63 characters or less, must start with a letter and include only letters, digits, or '-'.

Region \* — us-central1 (Iowa)

Advanced settings

**CREATE** CANCEL

## Overview

Notebooks API is used to manage notebook resources in Google Cloud.

Quota exceeded for quota metric 'Create Runtime API requests' and limit 'Create Runtime API requests per minute' of service 'notebooks.googleapis.com' for consumer 'project\_number:842795097713'.



4. Unable to update Runtime API requests per minute

## 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

<input type="checkbox"/> Service	Quota	Dimensions (e.g. location)	Limit	Current usage percentage
<input type="checkbox"/> Notebooks API	Create Runtime API requests per minute		0	<div style="width: 0%;"><div style="width: 0%;"></div></div>

## Quota Name

## Limit

Create Runtime API requests per minute

0

