

# Assignment 7: optional catchup assignment 2 - VERTEX AI - for midterm and quiz - this will catch up midterm.

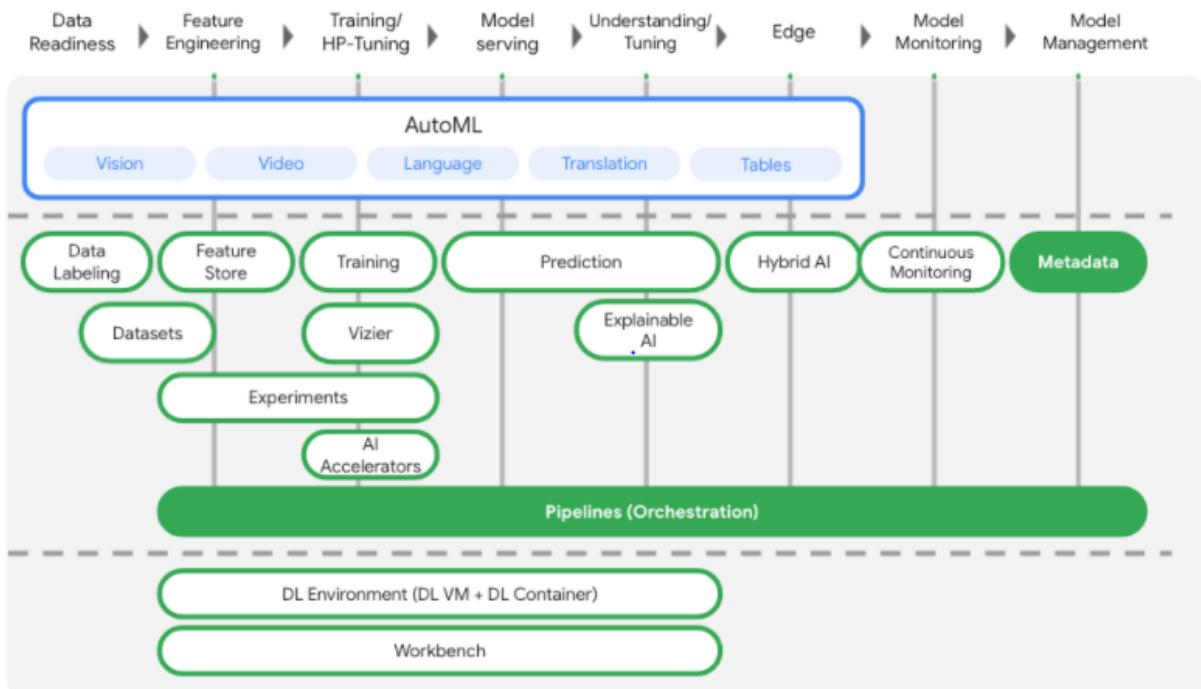
## 1) Using Vertex ML Metadata with Pipelines

Reference: <https://codelabs.developers.google.com/vertex-mlmd-pipelines#0>

### Objectives:

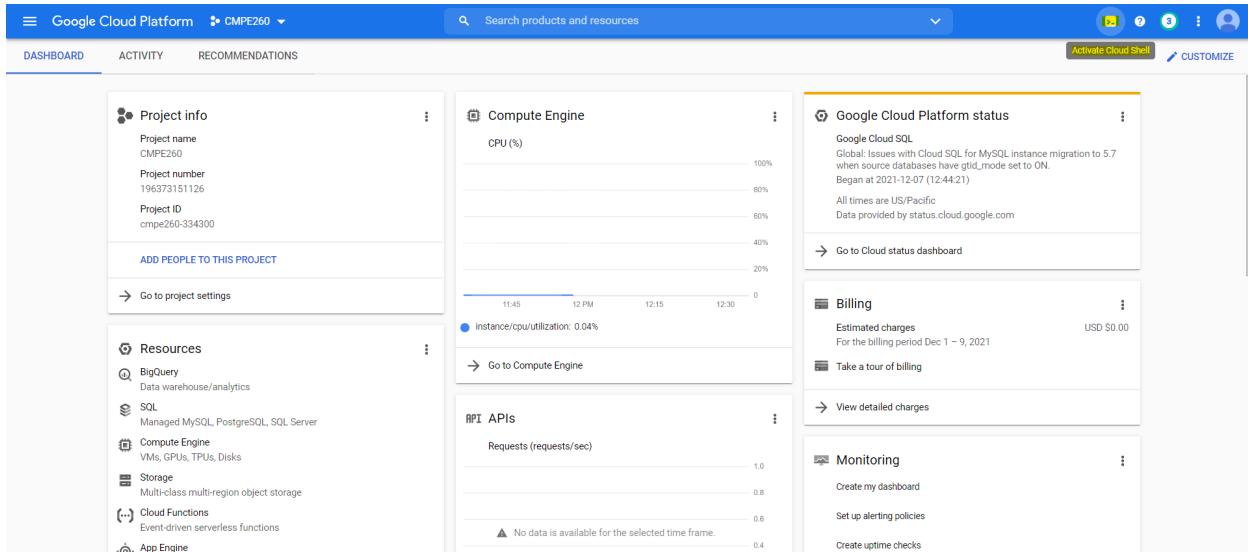
- Use the Kubeflow Pipelines SDK to build an ML pipeline that creates a dataset in Vertex AI, and trains and deploys a custom Scikit-learn model on that dataset
- Write custom pipeline components that generate artifacts and metadata
- Compare Vertex Pipelines runs, both in the Cloud console and programmatically
- Trace the lineage for pipeline-generated artifacts
- Query your pipeline run metadata

Vertex AI also includes a variety of MLOps products, including Vertex Pipelines, ML Metadata, Model Monitoring, Feature Store, and more. This lab focuses on **Vertex Pipelines** and **Vertex ML Metadata**.



# Cloud environment setup

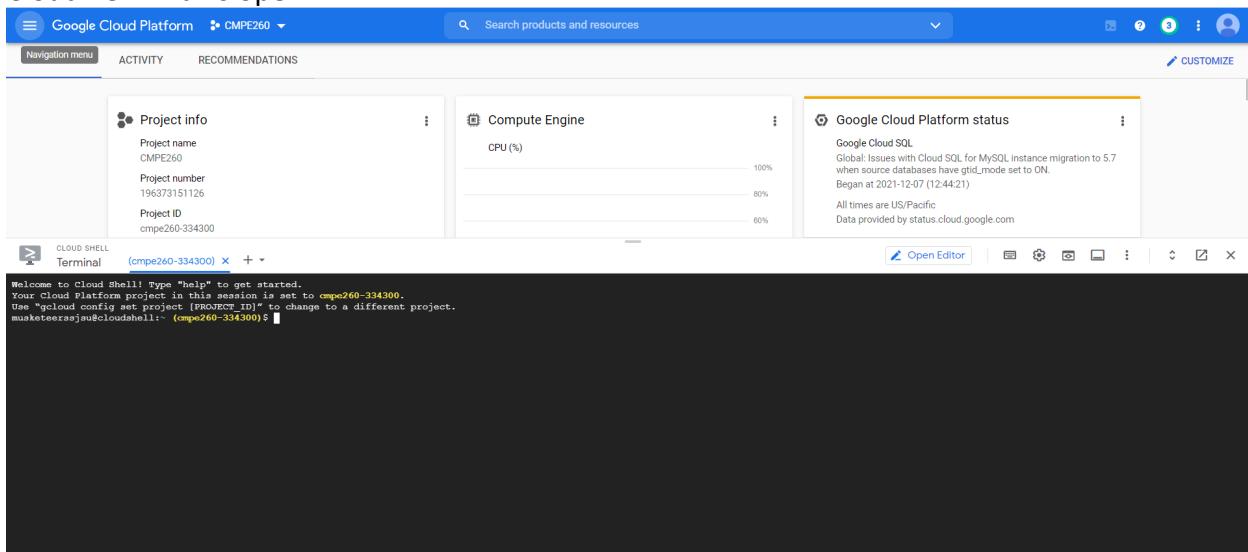
## Start Cloud Shell



The screenshot shows the Google Cloud Platform dashboard for project CMPE260. The dashboard is divided into several sections:

- Project info:** Project name CMPE260, Project number 196373151126, Project ID cmpe260-334300.
- Compute Engine:** CPU utilization chart from 11:45 AM to 12:30 PM. The utilization is at 0.04%.
- Google Cloud Platform status:** Global issues with Cloud SQL for MySQL instance migration to 5.7 when source databases have gtid\_mode set to ON. Began at 2021-12-07 (12:44:21). All times are US/Pacific. Data provided by status.cloud.google.com.
- Billing:** Estimated charges USD \$0.00 for the billing period Dec 1 – 9, 2021. Take a tour of billing, View detailed charges.
- Monitoring:** Create my dashboard, Set up alerting policies, Create uptime checks.
- API APIs:** Requests (requests/sec) chart from 11:45 AM to 12:30 PM. The requests per second are at 1.0.
- Resources:** BigQuery, SQL, Compute Engine, Storage, Cloud Functions, App Engine.

## Cloud Terminal is open

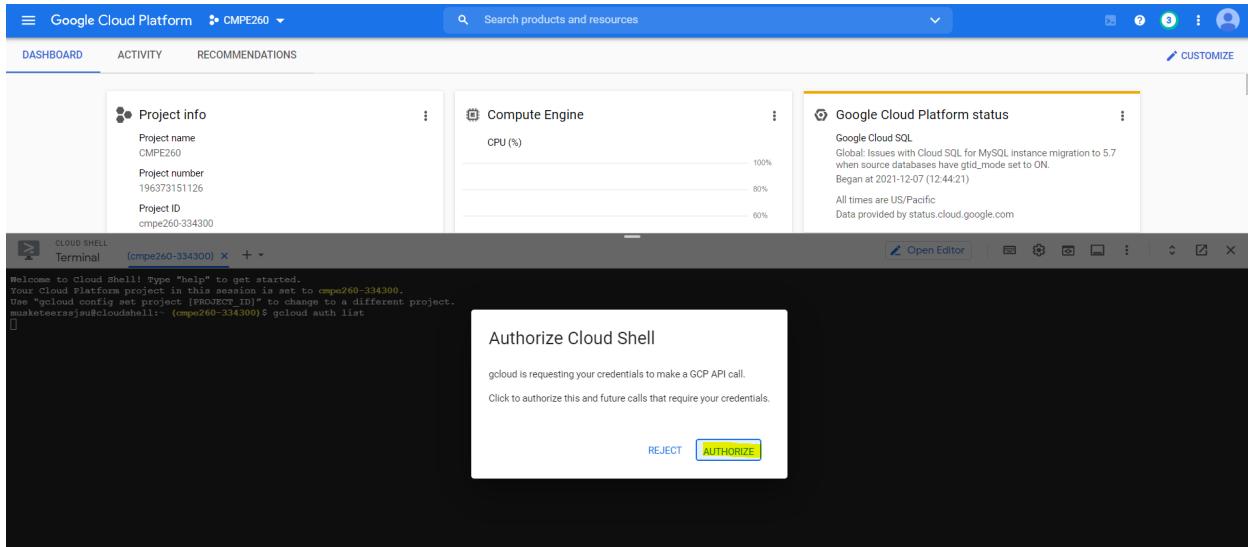


The screenshot shows the Google Cloud Platform dashboard with an open Cloud Shell terminal window. The terminal window title is "CLOUD SHELL Terminal (cmpe260-334300)". The terminal output is as follows:

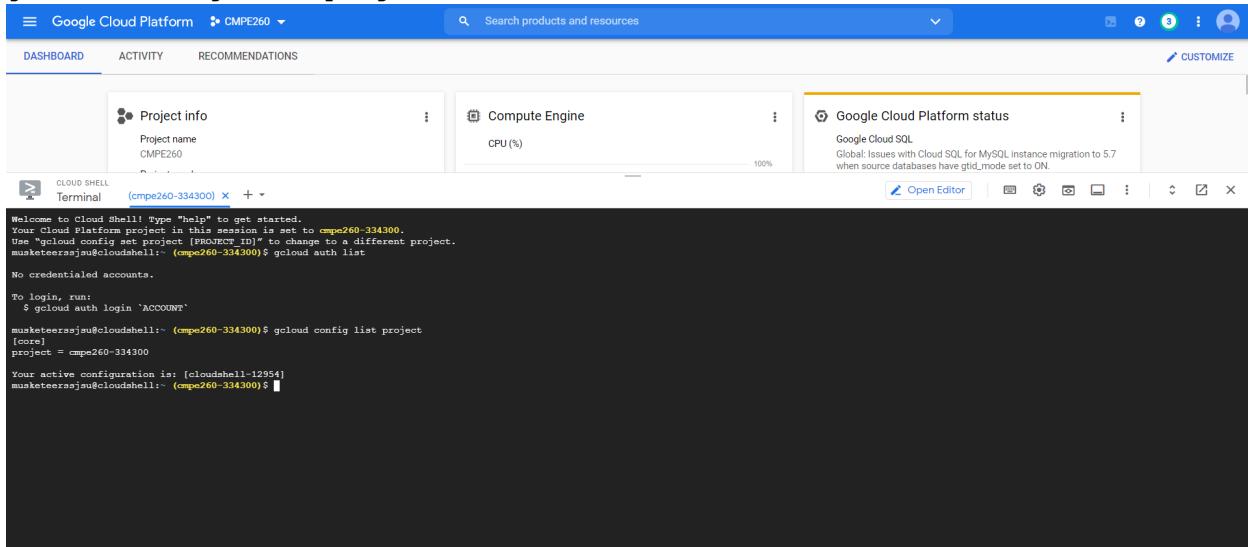
```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cmpe260-334300.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
musketeers@su:~$
```

Run the following commands:

```
gcloud auth list
```



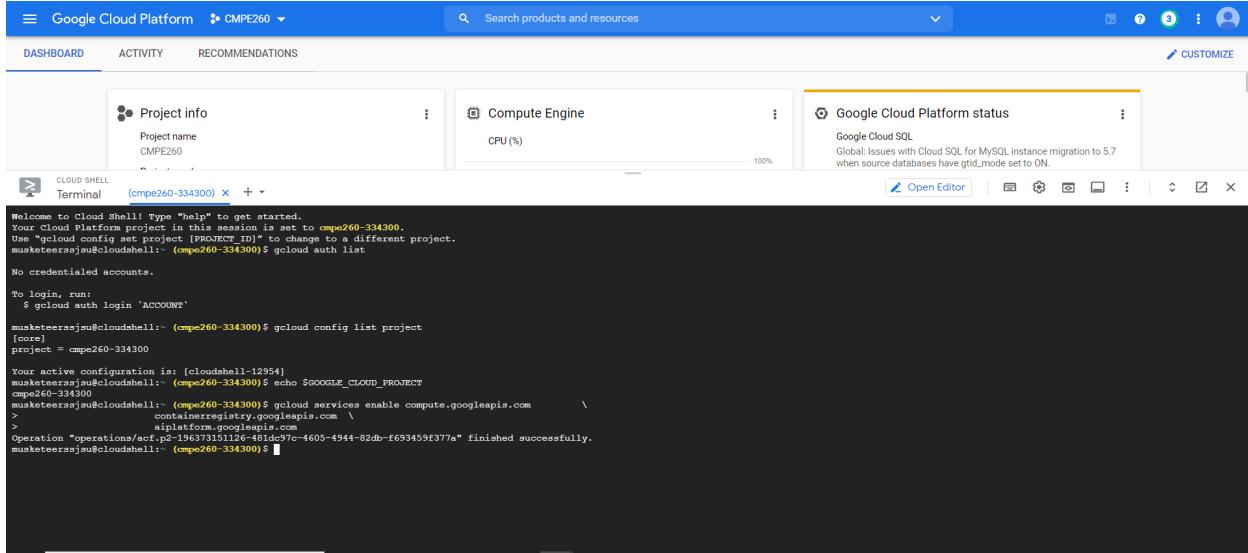
```
gcloud config list project
```



Enable APIs

```
echo $GOOGLE_CLOUD_PROJECT
```

```
gcloud services enable compute.googleapis.com \
    containerregistry.googleapis.com \
    aiplatform.googleapis.com
```



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cmpe260-334300.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
muskteers@subcloudshell:~ (cmpe260-334300)$ gcloud auth list

No credentialed accounts.

To login, run:
$ gcloud auth login 'ACCOUNT'

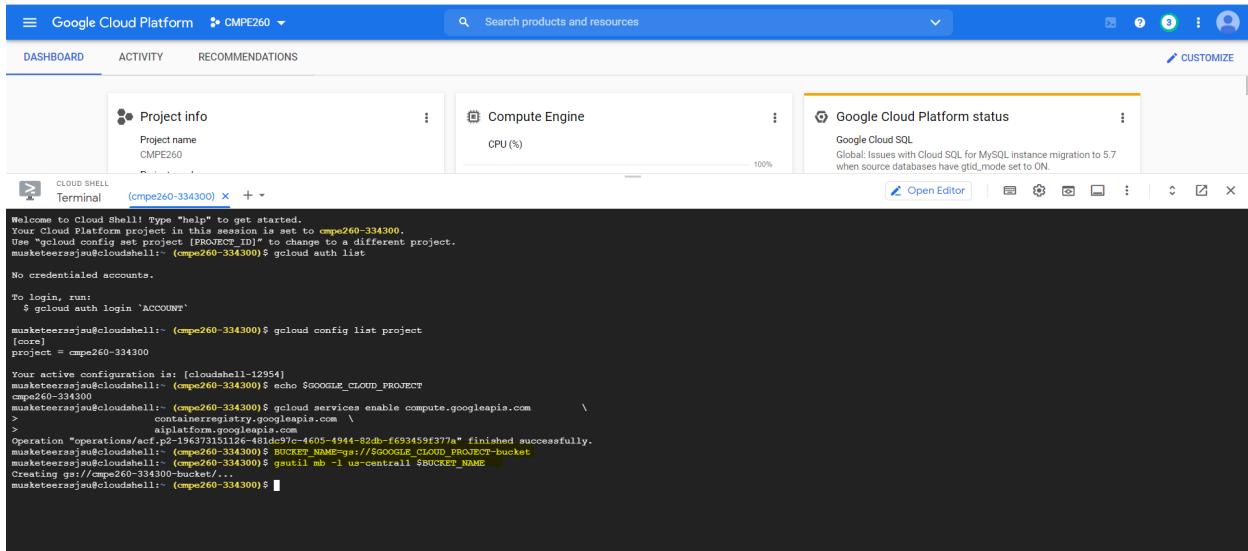
muskteers@subcloudshell:~ (cmpe260-334300)$ gcloud config list project
[core]
project = cmpe260-334300

Your active configuration is: [cloudshell-12954]
muskteers@subcloudshell:~ (cmpe260-334300)$ echo $GOOGLE_CLOUD_PROJECT
cmpe260-334300
muskteers@subcloudshell:~ (cmpe260-334300)$ gcloud services enable compute.googleapis.com \
>     containerregistry.googleapis.com \
>     aiplatform.googleapis.com
Operation "operations/act,p2-196373151126-401dc97c-4605-4944-82db-f693459f377a" finished successfully.
muskteers@subcloudshell:~ (cmpe260-334300)$
```

## Create a Cloud Storage Bucket

Run the following commands:

```
BUCKET_NAME=gs://$GOOGLE_CLOUD_PROJECT-bucket
gsutil mb -l us-central1 $BUCKET_NAME
```



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to cmpe260-334300.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
muskteers@subcloudshell:~ (cmpe260-334300)$ gcloud auth list

No credentialed accounts.

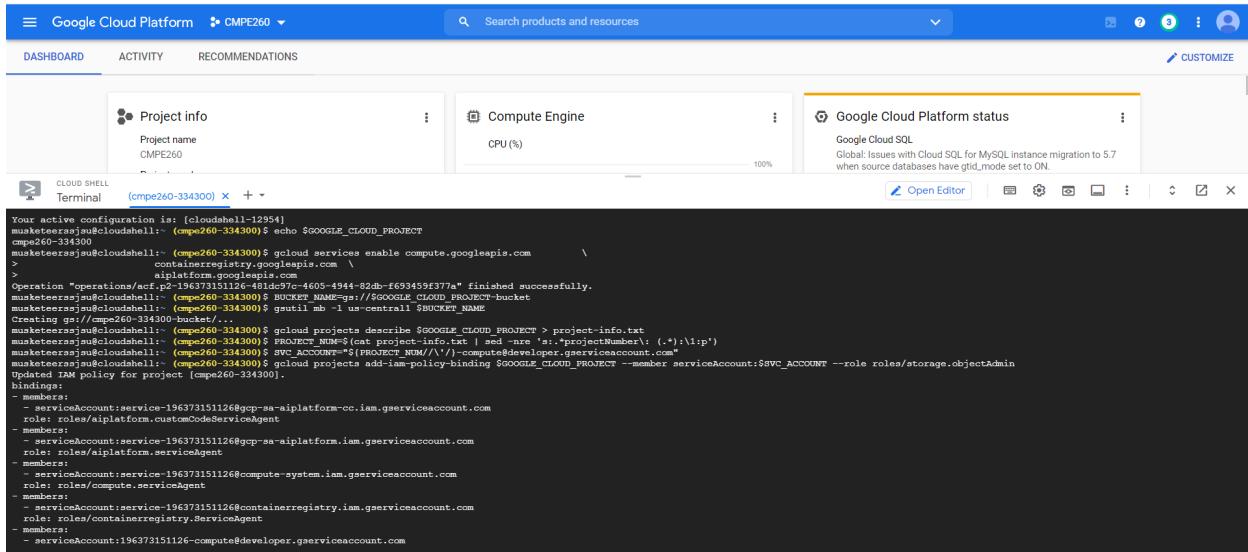
To login, run:
$ gcloud auth login 'ACCOUNT'

muskteers@subcloudshell:~ (cmpe260-334300)$ gcloud config list project
[core]
project = cmpe260-334300

Your active configuration is: [cloudshell-12954]
muskteers@subcloudshell:~ (cmpe260-334300)$ echo $GOOGLE_CLOUD_PROJECT
cmpe260-334300
muskteers@subcloudshell:~ (cmpe260-334300)$ gcloud services enable compute.googleapis.com \
>     containerregistry.googleapis.com \
>     aiplatform.googleapis.com
Operation "operations/act,p2-196373151126-401dc97c-4605-4944-82db-f693459f377a" finished successfully.
muskteers@subcloudshell:~ (cmpe260-334300)$ gsutil mb -l us-central1 $BUCKET_NAME
Creating gs://cmpe260-334300-bucket/...
muskteers@subcloudshell:~ (cmpe260-334300)$
```

Run the following command to add the permission for compute service account access to this Bucket

```
gcloud projects describe $GOOGLE_CLOUD_PROJECT > project-info.txt  
PROJECT_NUM=$(cat project-info.txt | sed -nre 's/:.*projectNumber\:(.):/1:p')  
SVC_ACCOUNT="${PROJECT_NUM//\//}-compute@developer.gserviceaccount.com"  
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT --member  
serviceAccount:$SVC_ACCOUNT --role roles/storage.objectAdmin
```

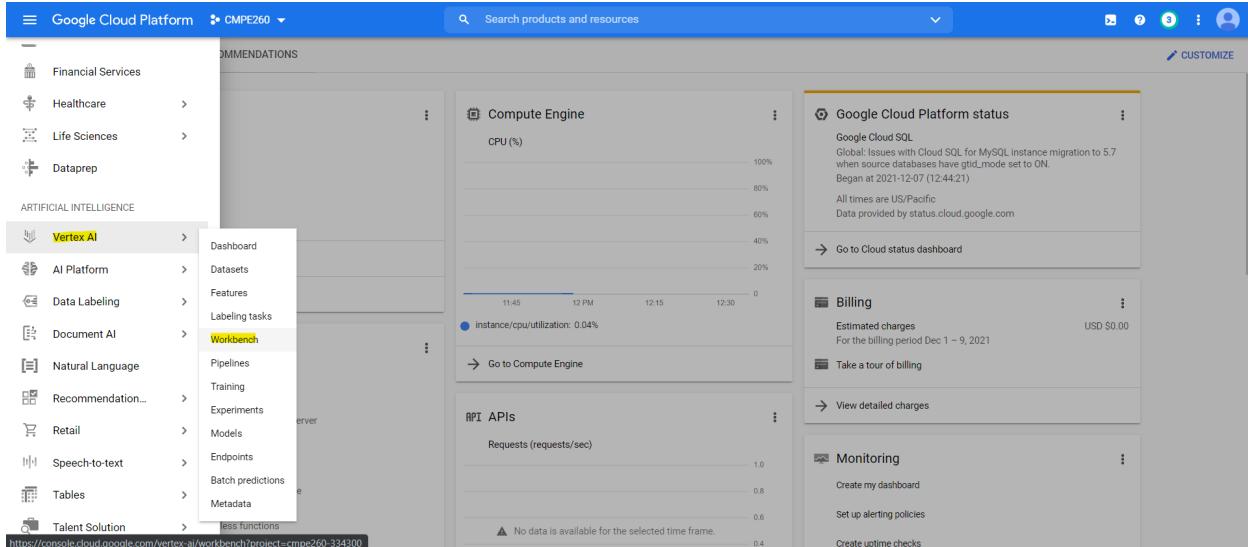


The screenshot shows the Google Cloud Platform dashboard for project CMPE260. A terminal window is open in the Cloud Shell, displaying the following command sequence:

```
Your active configuration is: [cloudshell-12954]  
muketeersaj@cloudshell:~ (cmpe260-334300)$ echo $GOOGLE_CLOUD_PROJECT  
cmpe260-334300  
muketeersaj@cloudshell:~ (cmpe260-334300)$ gcloud services enable compute.googleapis.com  
>     containerregistry.googleapis.com  
> Operation "operations/acf-p2-196373151126-401d97c4605-4944-82db-f693459f377a" finished successfully.  
muketeersaj@cloudshell:~ (cmpe260-334300)$ gcloud auth activate-service-account $SVC_ACCOUNT --key-file=cmpe260-334300.json  
muketeersaj@cloudshell:~ (cmpe260-334300)$ gsutil mb -l us-central1 $BUCKET_NAME  
Creating gs://cmpe260-334300-bucket/...  
muketeersaj@cloudshell:~ (cmpe260-334300)$ gcloud projects describe $GOOGLE_CLOUD_PROJECT > project-info.txt  
muketeersaj@cloudshell:~ (cmpe260-334300)$ PROJECT_NUM=$(cat project-info.txt | sed -nre 's/:.*projectNumber\:(.):/1:p')  
muketeersaj@cloudshell:~ (cmpe260-334300)$ SVC_ACCOUNT="${PROJECT_NUM//\//}-compute@developer.gserviceaccount.com"  
muketeersaj@cloudshell:~ (cmpe260-334300)$ gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT --member serviceAccount:$SVC_ACCOUNT --role roles/storage.objectAdmin  
Updated IAM policy for project [cmpe260-334300].  
bindings:  
- members:  
  - serviceAccount:service-196373151126@gcp-sa-aiplatform-cc.iam.gserviceaccount.com  
    role: roles/aiplatform.customCodeServiceAgent  
- members:  
  - serviceAccount:service-196373151126@gcp-sa-aiplatform.iam.gserviceaccount.com  
    role: roles/aiplatform.serviceAgent  
- members:  
  - serviceAccount:service-196373151126@compute-system.iam.gserviceaccount.com  
    role: roles/compute.serviceAgent  
- members:  
  - serviceAccount:service-196373151126@containerregistry.iam.gserviceaccount.com  
    role: roles/containerregistry.ServiceAgent  
- members:  
  - serviceAccount:196373151126-compute@developer.gserviceaccount.com
```

Create a Vertex AI Workbench instance

Navigate to Vertex AI → Workbench



## Select TensorFlow Enterprise 2.3 (with LTS) instance type without GPUs:

The screenshot shows the Google Cloud Platform Vertex AI Notebooks interface. On the left, there's a sidebar with various options like Dashboard, Datasets, Features, Labeling tasks, Workbench (which is selected), Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The main area is titled "Notebooks" and has a "NEW NOTEBOOK" button. Below it, there's a "MANAGED NOTEBOOKS" section with a "PREVIEW" tab selected. A dropdown menu is open under "TensorFlow Enterprise", showing "Without GPUs" as the selected option. Other options in the dropdown include "With 1 NVIDIA Tesla T4", "With 1 NVIDIA V100", and "With 1 NVIDIA P100". To the right, there's a "SCHEDULES" section with a table showing a single entry for "TensorFlow Enterprise 2.3 (with LTS)".

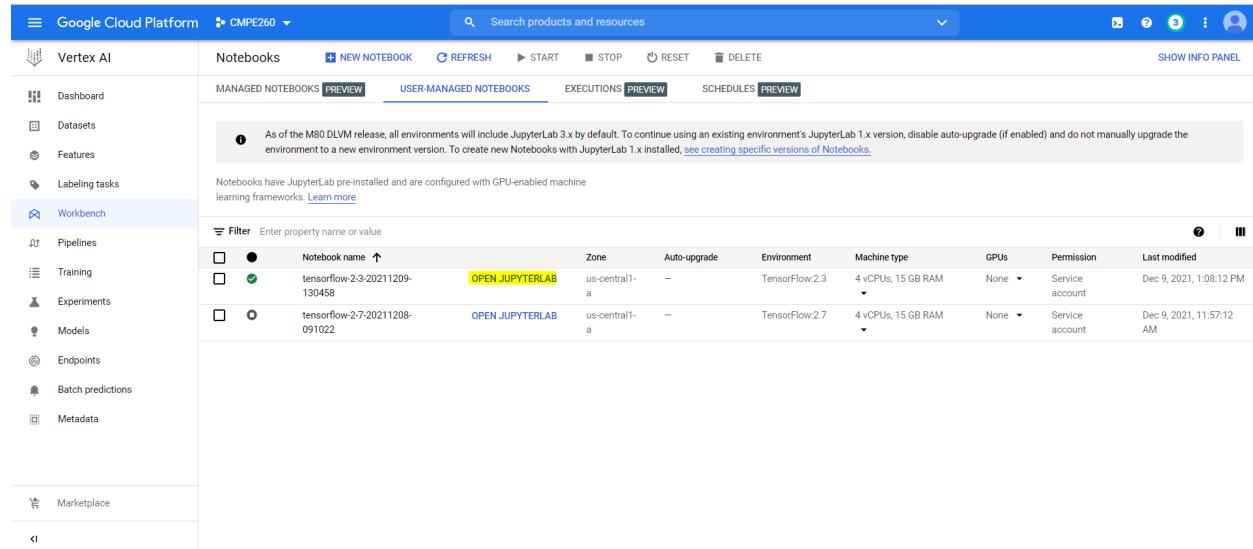
This screenshot shows the "New notebook" creation dialog. In the "Notebook properties" section, the "Environment" is set to "TensorFlow Enterprise 2.3 (with LTS and Intel® MKL-DNN/MKL)" and the "Machine type" is set to "4 vCPUs, 12 GB RAM". The "External IP" is listed as "Ephemeral(Automatic)". The "Permission" is set to "Compute Engine default service account". The "Estimated cost" is shown as "\$102.70 monthly, \$0.141 hourly". At the bottom, there are "ADVANCED OPTIONS", "CANCEL", and "CREATE" buttons. The background shows the same managed notebooks interface as the previous screenshot.

## Vertex Pipelines setup

There are a few additional libraries we'll need to install in order to use Vertex Pipelines:

- **Kubeflow Pipelines**: This is the SDK we'll be using to build our pipeline. Vertex Pipelines supports running pipelines built with both Kubeflow Pipelines or TFX.
- **Vertex AI SDK**: This SDK optimizes the experience for calling the Vertex AI API. We'll use it to run our pipeline on Vertex AI.

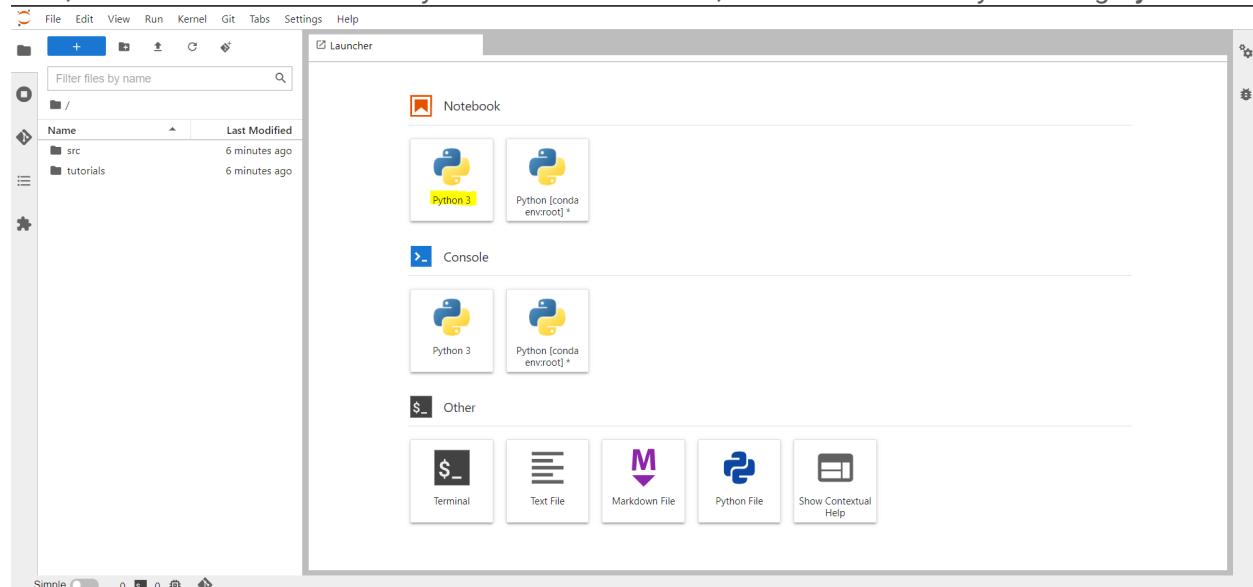
Create Python notebook and install libraries



The screenshot shows the Google Cloud Platform Vertex AI interface. On the left, there's a sidebar with various options like Dashboard, Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The main area is titled "Notebooks" and shows two entries in a table:

Notebook name	Zone	Auto-upgrade	Environment	Machine type	GPUs	Permission	Last modified
tensorflow-2-3-20211209-130458	us-central1-a	—	TensorFlow2.3	4 vCPUs, 15 GB RAM	None	Service account	Dec 9, 2021, 1:08:12 PM
tensorflow-2-7-20211208-091022	us-central1-a	—	TensorFlow2.7	4 vCPUs, 15 GB RAM	None	Service account	Dec 9, 2021, 11:57:12 AM

First, from the Launcher menu in your Notebook instance, create a notebook by selecting **Python 3**:



The screenshot shows the JupyterLab launcher menu. It has three main sections: "Notebook" (with icons for Python 3 and Python [conda envroot]), "Console" (with icons for Python 3 and Python [conda envroot]), and "Other" (with icons for Terminal, Text File, Markdown File, Python File, and Show Contextual Help). The "Python 3" icon under "Notebook" is highlighted, indicating it's the selected option.

To install both services we'll be using in this lab, first set the user flag in a notebook cell:

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

The screenshot shows a Jupyter Notebook interface with two code cells. The first cell contains the command to install the Google Cloud AI Platform package with the user flag. The second cell contains the command to install the KFP package with the user flag. Both cells show the output of the pip3 installations, which include dependency resolution and download progress.

```
[2]: USER_FLAG = "--user"
[3]: !pip3 install {USER_FLAG} google-cloud-aiplatform==1.7.0
[4]: !pip3 install {USER_FLAG} kfp==1.8.9
```

```
Collecting google-cloud-aiplatform==1.7.0
  Downloading google_cloud_aiplatform-1.7.0-py2.py3-none-any.whl (1.6 MB)
    100% |██████████| 1.6 MB 5.2 MB/s
Requirement already satisfied: proto-plus>=1.10.1 in /opt/conda/lib/python3.7/site-packages (from google-cloud-aiplatform==1.7.0) (1.19.8)
Requirement already satisfied: google-cloud-storage<2.0.0dev,>=1.32.0 in /opt/conda/lib/python3.7/site-packages (from google-cloud-aiplatform==1.7.0) (1.43.0)
Requirement already satisfied: packaging>=14.3 in /opt/conda/lib/python3.7/site-packages (from google-cloud-aiplatform==1.7.0) (21.3)
Requirement already satisfied: google-api-core[grpc]<3.0.0dev,>=1.26.0 in /opt/conda/lib/python3.7/site-packages (from google-cloud-aiplatform==1.7.0) (2.2.2)
Requirement already satisfied: google-cloud-bigquery<3.0.0dev,>=1.15.0 in /opt/conda/lib/python3.7/site-packages (from google-cloud-aiplatform==1.7.0) (2.30.1)
Requirement already satisfied: google-auth<3.0.0dev,>=1.25.0 in /opt/conda/lib/python3.7/site-packages (from google-api-core[grpc]<3.0.0dev,>=1.26.0>google-cloud-aiplatform==1.7.0) (2.3.3)
Requirement already satisfied: requests<3.0.0dev,>=2.18.0 in /opt/conda/lib/python3.7/site-packages (from google-api-core[grpc]<3.0.0dev,>=1.26.0>google-cloud-aiplatform==1.7.0) (2.26.0)
Requirement already satisfied: setuptools>=40.3.0 in /opt/conda/lib/python3.7/site-packages (from google-api-core[grpc]<3.0.0dev,>=1.26.0>google-cloud-aiplatform==1.7.0) (59.4.0)
Requirement already satisfied: googleapis-common-protos<2.0.0dev,>=1.52.0 in /opt/conda/lib/python3.7/site-packages (from google-api-core[grpc]<3.0.0dev,>=1.26.0>google-cloud-aiplatform==1.7.0) (1.53.0)
Requirement already satisfied: protobuf>=3.12.0 in /opt/conda/lib/python3.7/site-packages (from google-api-core[grpc]<3.0.0dev,>=1.26.0>google-cloud-aiplatform==1.7.0) (3.19.1)
Requirement already satisfied: grpcio-status<2.0.0dev,>=1.33.2 in /opt/conda/lib/python3.7/site-packages (from google-api-core[grpc]<3.0.0dev,>=1.26.0>google-cloud-aiplatform==1.7.0) (1.42.0)
Requirement already satisfied: grpcio<2.0.0dev,>=1.33.2 in /opt/conda/lib/python3.7/site-packages (from google-api-core[grpc]<3.0.0dev,>=1.26.0>google-cloud-aiplatform==1.7.0) (1.42.0)
Requirement already satisfied: google-cloud-core<3.0.0dev,>=1.4.1 in /opt/conda/lib/python3.7/site-packages (from google-cloud-bigquery<3.0.0dev,>=1.15.0>google-cloud-aiplatform==1.7.0) (2.2.1)
Requirement already satisfied: python-dateutil<3.0.0dev,>=2.7.2 in /opt/conda/lib/python3.7/site-packages (from google-cloud-bigquery<3.0.0dev,>=1.15.0>google-cloud-aiplatform==1.7.0) (2.8.2)
```

The screenshot shows a Jupyter Notebook interface with three code cells. The first cell imports os and checks for the IS\_TESTING environment variable. The second cell runs a Python command to print the KFP SDK version. The third cell lists all installed packages using pip list, showing that both google-cloud-aiplatform and kfp are installed at version 1.7.0.

```
[4]: 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: {}'.format(kfp.__version__))"
KFP SDK version: 1.8.9

[2]: !pip list | grep aiplatform
google-cloud-aiplatform            1.7.0
```

## Set your project ID and bucket

The screenshot shows a Jupyter Notebook interface with a sidebar on the left displaying a file tree. The main area contains the following Python code:

```
[2]: !pip list | grep aiplatform
google-cloud-aiplatform      1.7.0

[3]:
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: cmpe260-334300

[4]: if PROJECT_ID == "" or PROJECT_ID is None:
    PROJECT_ID = "cmpe260-334300" # @param {type:"string"}

[5]: BUCKET_NAME="gs://" + PROJECT_ID + "-bucket"

[ ]:
```

## Import libraries

The screenshot shows a Jupyter Notebook interface with a sidebar on the left displaying a file tree. The main area contains the following Python code:

```
PROJECT_ID = "cmpe260-334300" # @param {type:"string"}

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

import matplotlib.pyplot as plt
import pandas as pd

from kfp.v2 import compiler, dsl
from kfp.v2.dsl import pipeline, component, Artifact, Dataset, Input, Metrics, Model, Output, InputPath, OutputPath

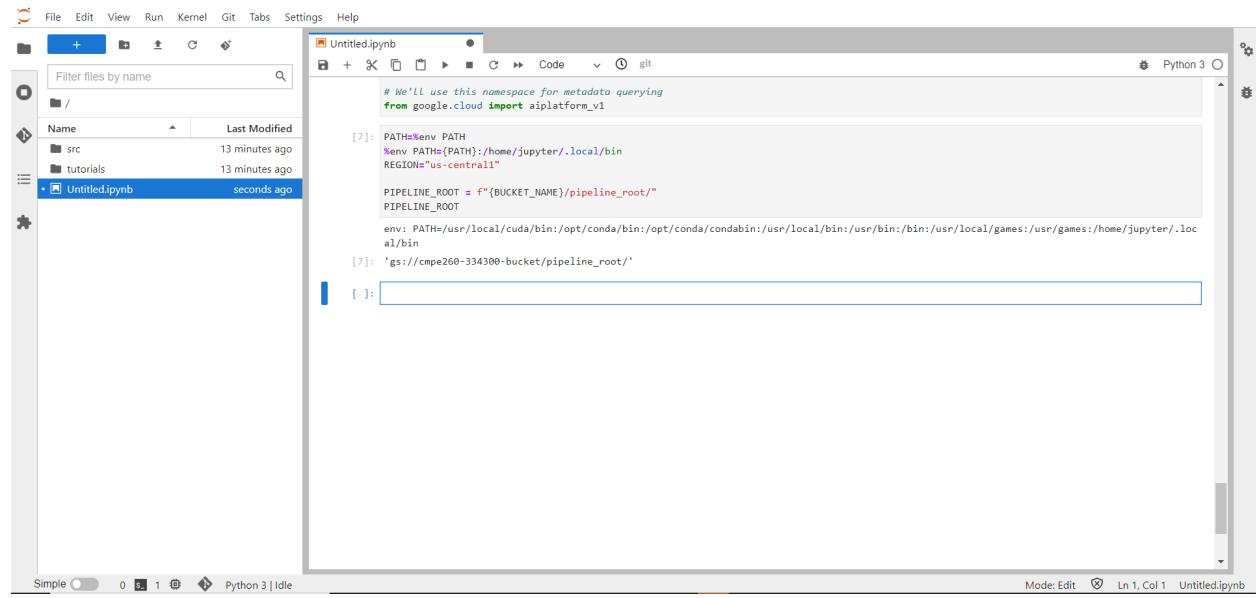
from google.cloud import aiplatform

# We'll use this namespace for metadata querying
from google.cloud import aiplatform_v1

[ ]:
```

## Define constants

`PIPELINE_ROOT` is the Cloud Storage path where the artifacts created by our pipeline will be written.



The screenshot shows a Jupyter Notebook interface with a sidebar on the left containing a file tree. The tree shows a root folder with 'src' and 'tutorials' subfolders, and a file named 'Untitled.ipynb' which was modified 'seconds ago'. The main area displays a code cell numbered [7]. The code in the cell is as follows:

```
# We'll use this namespace for metadata querying
from google.cloud import aiplatform_v1

[7]: PATH=os.getenv('PATH')
env = os.environ.copy()
env['PATH']=PATH+':/home/jupyter/.local/bin'
REGION='us-central1'

PIPELINE_ROOT = f'{os.getenv("BUCKET_NAME")}/pipeline_root/'

PIPELINE_ROOT

env['PATH']=env['PATH']+':/opt/conda/bin:/opt/conda/condabin:/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games:/home/jupyter/.local/bin'

[7]: 'gs://cmpe260-334300-bucket/pipeline_root/'
```

The cell output is shown in a blue box below the code, displaying the string: 'gs://cmpe260-334300-bucket/pipeline\_root/'.

## Creating a 3-step pipeline with custom components

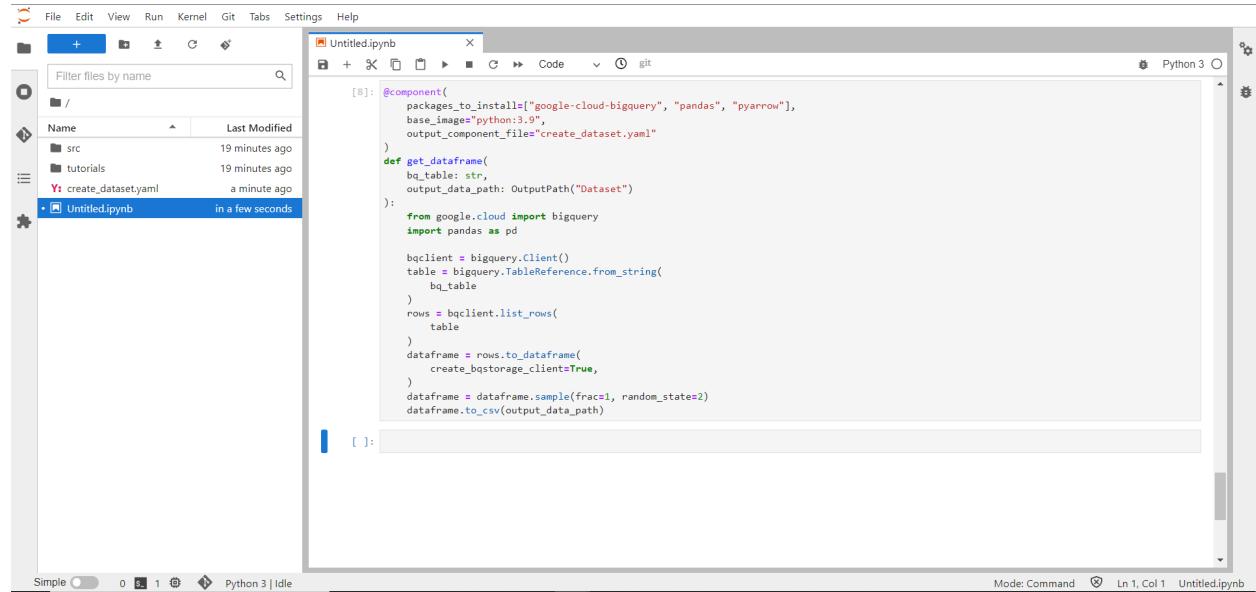
The focus of this lab is on understanding *metadata* from pipeline runs. In order to do that, we'll need a pipeline to run on Vertex Pipelines, which is where we'll start. Here we'll define a 3-step pipeline with the following custom components:

- `get_dataframe`: Retrieve data from a BigQuery table and convert it into a Pandas DataFrame
- `train_sklearn_model`: Use the Pandas DataFrame to train and export a Scikit Learn model, along with some metrics
- `deploy_model`: Deploy the exported Scikit Learn model to an endpoint in Vertex AI

### Create Python function based components

Using the KFP SDK, we can create components based on Python functions. We'll use that for the 3 components in this pipeline.

`get_dataframe` component:

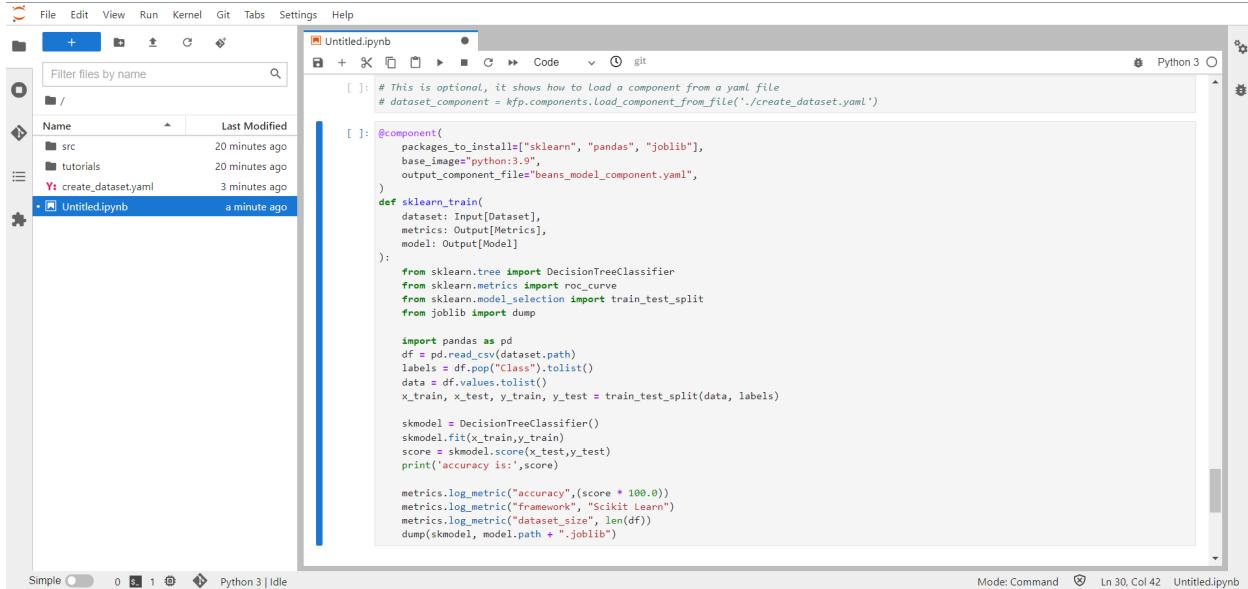


The screenshot shows a Jupyter Notebook environment with a sidebar containing file navigation and a code editor window. The code editor contains the following Python code:

```
[8]: @component(
    packages_to_install=["google-cloud-bigquery", "pandas", "pyarrow"],
    base_image="python:3.9",
    output_component_file="create_dataset.yaml"
)
def get_dataframe(
    bq_table: str,
    output_data_path: OutputPath("Dataset")
):
    from google.cloud import bigquery
    import pandas as pd

    bqClient = bigquery.Client()
    table = bigquery.TableReference.from_string(
        bq_table
    )
    rows = bqClient.list_rows(
        table
    )
    dataframe = rows.to_dataframe(
        create_bqstorage_client=True,
    )
    dataframe = dataframe.sample(frac=1, random_state=2)
    dataframe.to_csv(output_data_path)
```

This component exports the resulting Scikit model, along with a `Metrics` artifact that includes our model's accuracy, framework, and size of the dataset used to train it



```
# This is optional, it shows how to load a component from a yaml file
# dataset_component = kfp.components.load_component_from_file('./create_dataset.yaml')

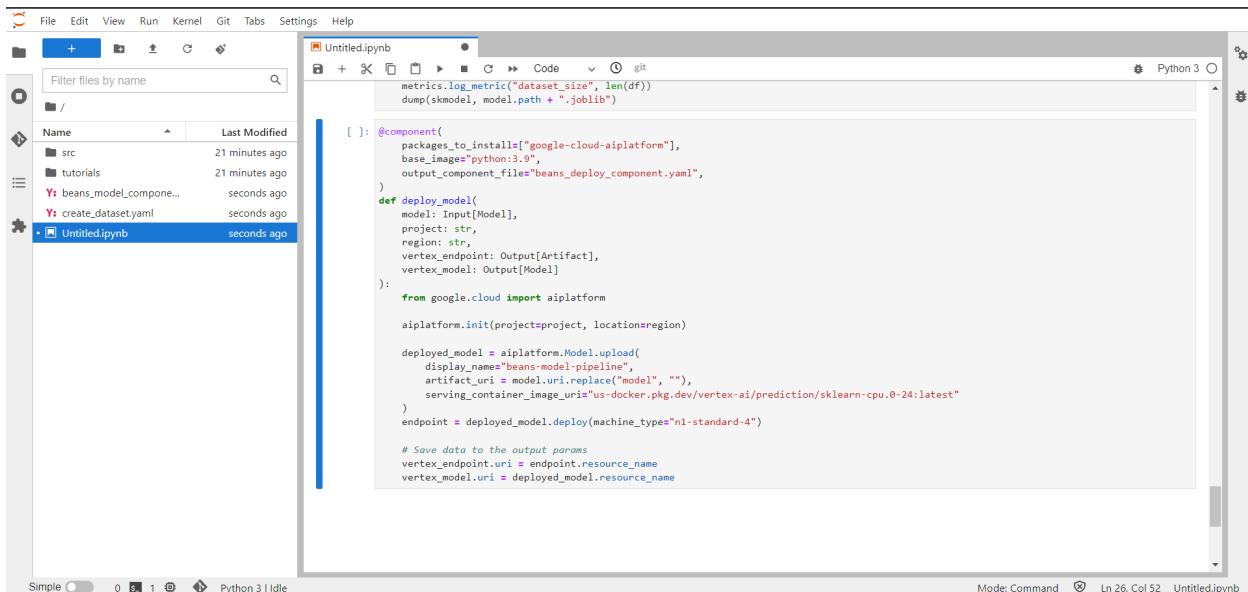
@component(
    packages_to_install=['sklearn', 'pandas', 'joblib'],
    base_image='python3:9',
    output_component_file='beans_model_component.yaml',
)
def sklearn_train(
    dataset: Input[Dataset],
    metrics: Output[Metrics],
    model: Output[Model]
):
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.metrics import roc_curve
    from sklearn.model_selection import train_test_split
    from joblib import dump

    import pandas as pd
    df = pd.read_csv(dataset.path)
    labels = df.pop("Class").tolist()
    data = df.values.tolist()
    x_train, x_test, y_train, y_test = train_test_split(data, labels)

    skmodel = DecisionTreeClassifier()
    skmodel.fit(x_train,y_train)
    score = skmodel.score(x_test,y_test)
    print('accuracy is:',score)

    metrics.log_metric("accuracy", (score * 100.0))
    metrics.log_metric("framework", "Scikit Learn")
    metrics.log_metric("dataset_size", len(df))
    dump(skmodel, model.path + ".joblib")
```

## Define a component to upload and deploy the model to Vertex AI



```
metrics.log_metric("dataset_size", len(df))
dump(skmodel, model.path + ".joblib")

@component(
    packages_to_install=['google-cloud-aiplatform'],
    base_image='python3:9',
    output_component_file='beans_deploy_component.yaml',
)
def deploy_model(
    model: Input[Model],
    project: str,
    region: str,
    vertex_endpoint: Output[Artifact],
    vertex_model: Output[Model]
):
    from google.cloud import aiplatform

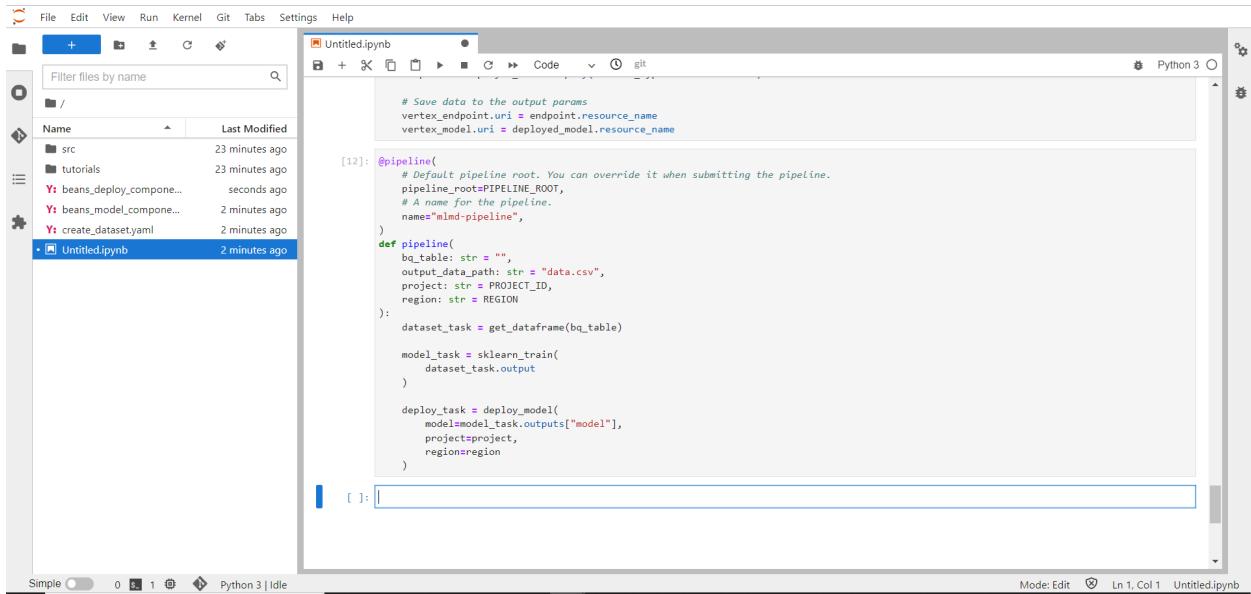
    aiplatform.init(project=project, location=region)

    deployed_model = aiplatform.Model.upload(
        display_name="beans-model-pipeline",
        artifact_uri=model.uri.replace("model", ""),
        serving_container_image_uri="us-docker.pkg.dev/vertex-ai/prediction/sklearn-cpu.0-24:latest"
    )
    endpoint = deployed_model.deploy(machine_type="n1-standard-4")

    # Save data to the output params
    vertex_endpoint.uri = endpoint.resource_name
    vertex_model.uri = deployed_model.resource_name
```

## Define and compile the pipeline

Now that we've defined our three components, next we'll create our *pipeline definition*.



The screenshot shows a Jupyter Notebook interface with a file tree on the left and a code editor on the right. The code editor contains Python code defining a pipeline. The code includes imports for `mlmd` and `sklearn`, defines a pipeline function, and creates tasks for dataset, model training, and deployment.

```
# Save data to the output params
vertex.endpoint.uri = endpoint.resource_name
vertex.model.uri = deployed_model.resource_name

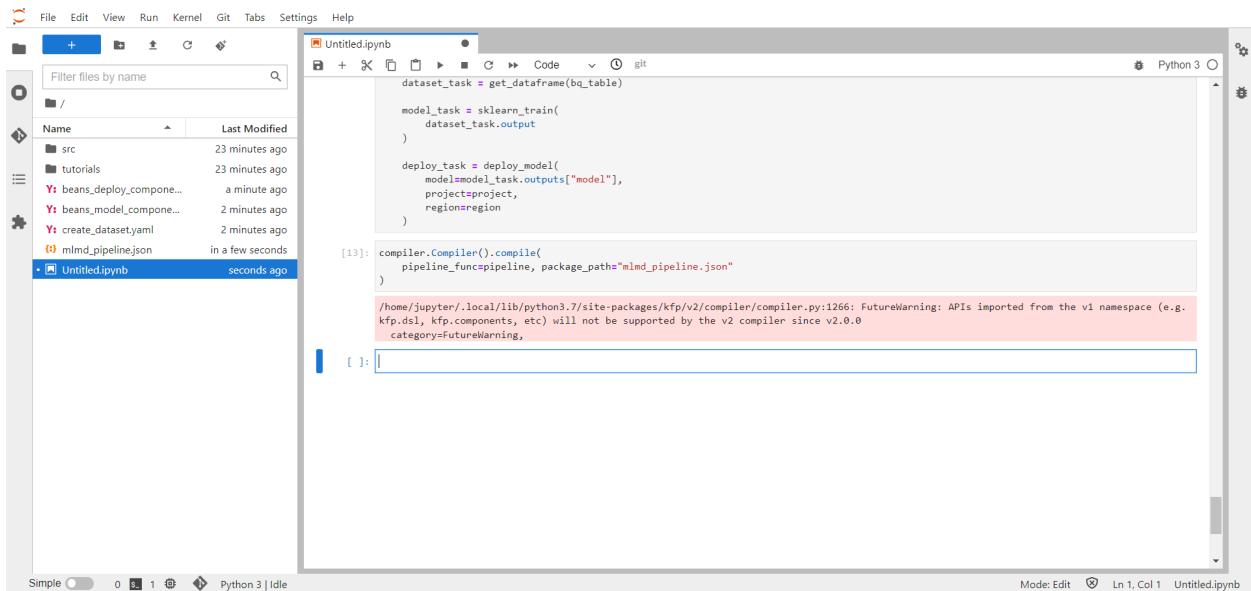
@pipeline(
    # Default pipeline root. You can override it when submitting the pipeline.
    pipeline_root=PIPELINE_ROOT,
    # A name for the pipeline.
    name="mlmd-pipeline",
)
def pipeline(
    bq_table: str = "",
    output_data_path: str = "data.csv",
    project: str = PROJECT_ID,
    region: str = REGION
):
    dataset_task = get_dataframe(bq_table)

    model_task = sklearn_train(
        dataset_task.output
    )

    deploy_task = deploy_model(
        model=model_task.outputs["model"],
        project=project,
        region=region
    )

[ ]:
```

The following will generate a JSON file that you'll use to run the pipeline:



The screenshot shows a Jupyter Notebook interface with a file tree on the left and a code editor on the right. The code editor contains Python code that compiles the pipeline into a JSON file named `mlmd\_pipeline.json`. A warning message is visible in the code editor area.

```
dataset_task = get_dataframe(bq_table)

model_task = sklearn_train(
    dataset_task.output
)

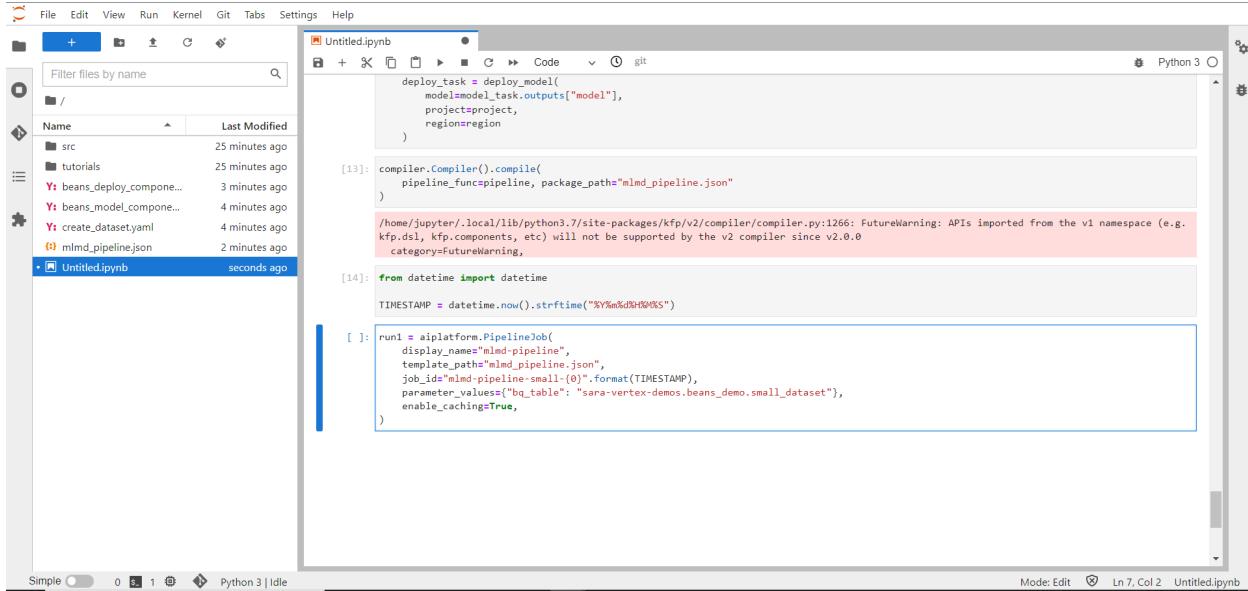
deploy_task = deploy_model(
    model=model_task.outputs["model"],
    project=project,
    region=region
)

[13]: compiler.Compiler().compile(
    pipeline_func=pipeline, package_path="mlmd_pipeline.json"
)

/home/jupyter/.local/lib/python3.7/site-packages/kfp/v2/compiler/compiler.py:1266: FutureWarning: APIs imported from the v1 namespace (e.g. kfp.dsl, kfp.components, etc) will not be supported by the v2 compiler since v2.0.0
    category=FutureWarning,
```

## Start two pipeline runs

Remember that our pipeline takes one parameter when we run it: the `bq_table` we want to use for training data. This pipeline run will use a smaller version of the beans dataset:



The screenshot shows a Jupyter Notebook interface with a sidebar containing file navigation and a code editor window titled "Untitled.ipynb". The code editor contains Python code for deploying a machine learning model and starting a pipeline job. The code includes imports for `deploy\_task`, `compiler.Compiler`, `aiplatform.PipelineJob`, and `datetime`. It defines a `deploy\_task` function and creates a `run1` pipeline job for a small dataset. A warning message from the compiler is visible in the code area.

```
deploy_task = deploy_model(
    model=model_task.outputs["model"],
    project=project,
    regions=region
)

[13]: compiler.Compiler().compile(
    pipeline_func=pipeline, package_path="mlmd_pipeline.json"
)

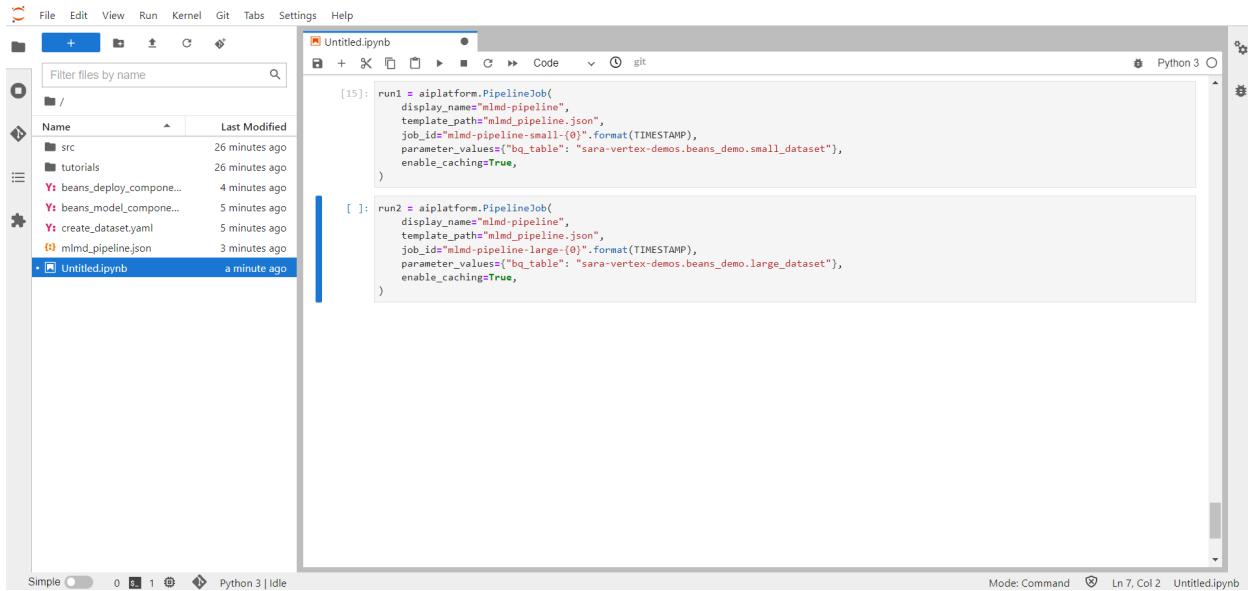
/home/jupyter/.local/lib/python3.7/site-packages/kfp/v2/compiler/compiler.py:1266: FutureWarning: APIs imported from the v1 namespace (e.g. kfp.dsl, kfp.components, etc) will not be supported by the v2 compiler since v2.0.0
      category=FutureWarning,
      )

[14]: from datetime import datetime

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

[ ]: run1 = aiplatform.PipelineJob(
    display_name="mlmd-pipeline",
    template_path="mlmd_pipeline.json",
    job_id="mlmd-pipeline-small-{0}".format(TIMESTAMP),
    parameter_values={"bq_table": "sara-vertex-demos.beans_demo.small_dataset"},
    enable_caching=True,
)
```

Next, create another pipeline run using a larger version of the same dataset.



The screenshot shows a Jupyter Notebook interface with a sidebar containing file navigation and a code editor window titled "Untitled.ipynb". The code editor contains Python code for starting two pipeline jobs, one for a small dataset and one for a large dataset. The code uses the same structure as the previous run but changes the dataset name in the parameter values. The code editor shows two code cells, [15] and [ ].

```
[15]: run1 = aiplatform.PipelineJob(
    display_name="mlmd-pipeline",
    template_path="mlmd_pipeline.json",
    job_id="mlmd-pipeline-small-{0}".format(TIMESTAMP),
    parameter_values={"bq_table": "sara-vertex-demos.beans_demo.small_dataset"},
    enable_caching=True,
)

[ ]: run2 = aiplatform.PipelineJob(
    display_name="mlmd-pipeline",
    template_path="mlmd_pipeline.json",
    job_id="mlmd-pipeline-large-{0}".format(TIMESTAMP),
    parameter_values={"bq_table": "sara-vertex-demos.beans_demo.large_dataset"},
    enable_caching=True,
```

## Kick off pipeline executions for both runs

The screenshot shows a Jupyter Notebook interface with a sidebar containing files like `src`, `tutorials`, and `mlmd\_pipeline.json`. The main area has three code cells:

```
[15]: run1 = aiplatform.PipelineJob(
    display_name="mlmd-pipeline",
    template_path="mlmd_pipeline.json",
    job_id="mlmd-pipeline-small-{0}.format(TIMESTAMP)",
    parameter_values={"bq_table": "sara-vertex-demos.beans_demo.small_dataset"},
    enable_caching=True,
)

[16]: run2 = aiplatform.PipelineJob(
    display_name="mlmd-pipeline",
    template_path="mlmd_pipeline.json",
    job_id="mlmd-pipeline-large-{0}.format(TIMESTAMP)",
    parameter_values={"bq_table": "sara-vertex-demos.beans_demo.large_dataset"},
    enable_caching=True,
)

[17]: run1.submit()

INFO:google.cloud.aiplatform.pipeline_jobs:Creating PipelineJob
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob created. Resource name: projects/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-small-20211209213113
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/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-small-20211209213113')
INFO:google.cloud.aiplatform.pipeline_jobs:View Pipeline Job:
https://console.cloud.google.com/vertex-ai/locations/us-central1/pipelines/runs/mlmd-pipeline-small-20211209213113?project=196373151126
```

[ ]:

kick off the second run:

The screenshot shows a Jupyter Notebook interface with a sidebar containing files like `src`, `tutorials`, and `mlmd\_pipeline.json`. The main area has three code cells:

```
[17]: run1.submit()

INFO:google.cloud.aiplatform.pipeline_jobs:Creating PipelineJob
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob created. Resource name: projects/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-small-20211209213113
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/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-small-20211209213113')
INFO:google.cloud.aiplatform.pipeline_jobs:View Pipeline Job:
https://console.cloud.google.com/vertex-ai/locations/us-central1/pipelines/runs/mlmd-pipeline-small-20211209213113?project=196373151126

[18]: run2.submit()

INFO:google.cloud.aiplatform.pipeline_jobs:Creating PipelineJob
INFO:google.cloud.aiplatform.pipeline_jobs:PipelineJob created. Resource name: projects/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-large-20211209213113
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/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-large-20211209213113')
INFO:google.cloud.aiplatform.pipeline_jobs:View Pipeline Job:
https://console.cloud.google.com/vertex-ai/locations/us-central1/pipelines/runs/mlmd-pipeline-large-20211209213113?project=196373151126
```

[ ]:

## Results:

Google Cloud Platform CMPE260 Search products and resources

Vertex AI mlmd-pipeline-small-20211209213113 CLONE STOP DELETE

Runtime Graph 3/3 steps completed Expand Artifacts 75% Pipeline run analysis

Dashboard Datasets Features Labeling tasks Workbench Pipelines Training Experiments Models Endpoints Batch predictions Metadata Marketplace Logs

get-dataframe python:3.9

sklearn-train python:3.9

deploy-model python:3.9

Basic info Duration 18 min 21 sec Started Dec 9, 2021, 1:33:09 PM Completed Dec 9, 2021, 1:51:31 PM Run name mlmd-pipeline-small-20211209213113 Pipeline name mlmd-pipeline Runtime environment Serverless Region us-central1 Service account 19637315116-compute@developer.gserviceaccount.com Debugging info View pipeline proto

Run Parameters Pipeline parameter values used for this run

Parameter	Type	Value
project	string	cmpe260-334300
output_data_path	string	data.csv
region	string	us-central1
bq_table	string	sara-vertex-demos beans_demo.small_dataset

Run metrics Metrics logged by this pipeline run

accuracy	0.85714285714296
----------	------------------

Google Cloud Platform > CMPE260 > mlmd-pipeline-large-20211209213113

Search products and resources

CLONE STOP DELETE

Runtime Graph 3/3 steps completed Expand Artifacts 75% Pipeline run analysis

SUMMARY NODE INFO

Basic info

Duration	16 min 40 sec
Started	Dec 9, 2021, 1:34:12 PM
Completed	Dec 9, 2021, 1:50:53 PM
Run name	mlmd-pipeline-large-20211209213113
Pipeline name	mlmd-pipeline
Runtime environment	Serverless
Region	us-central1
Service account	196373151126-compute@developer.gserviceaccount.com

Debugging info

[View pipeline proto](#)

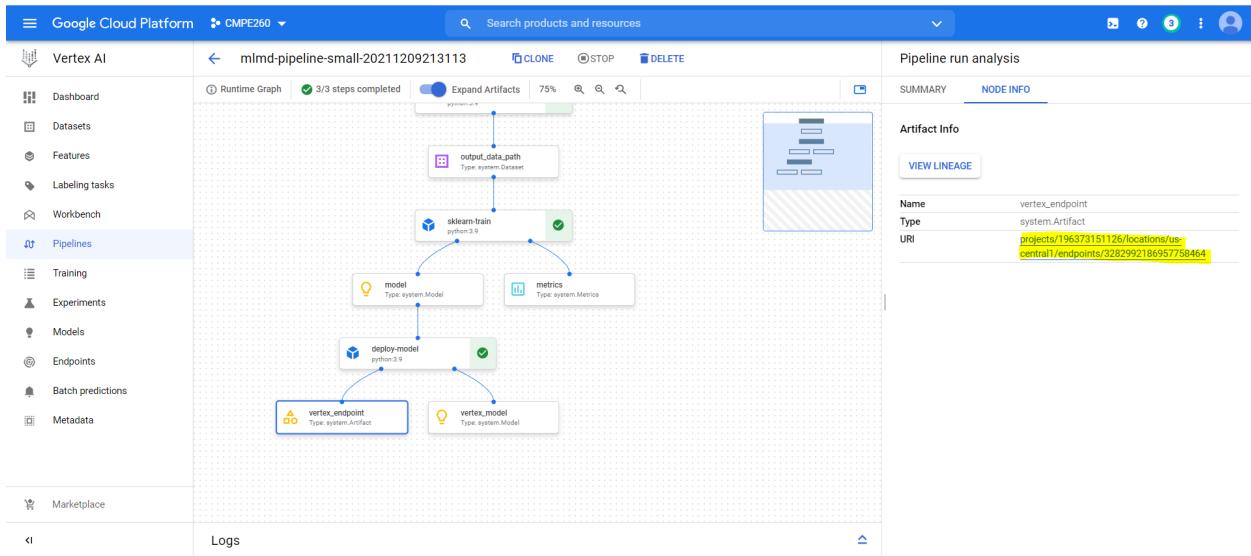
Run Parameters

Pipeline parameter values used for this run

Parameter	Type	Value
bq_table	string	sara-vertex-demos.beans_demo.large_dataset
region	string	us-central1
project	string	cmpe260-334300
output_data_path	string	data.csv

Run metrics

Metrics logged by this pipeline run



Google Cloud Platform CMPE260

Vertex AI Endpoints

**CREATE ENDPOINT**

REFRESH

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.

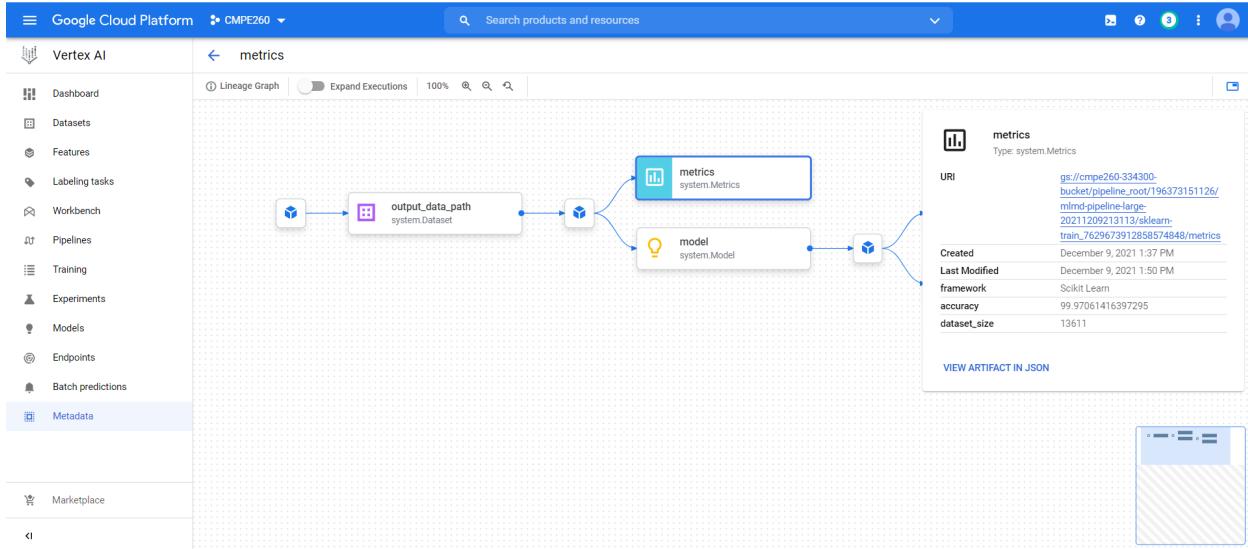
To create an endpoint, you need at least one machine learning model. [Learn more](#)

Region: us-central1 (Iowa)

Filter: Enter a property name

Name	ID	Status	Models	Region	Monitoring	Most recent alerts	Last updated	API	Notification	Labels
beans-model-pipeline_endpoint	1416461247645220864	Active	1	us-central1	Disabled	—	Dec 9, 2021, 1:50:03 PM	Sample request		
beans-model-pipeline_endpoint	3282992186957758464	Active	1	us-central1	Disabled	—	Dec 9, 2021, 1:49:59 PM	Sample request		
titanic-endpoint	432424729064767488	Ready	0	us-central1	Disabled	—	Dec 6, 2021, 10:02:08 PM	Sample request		

Marketplace



## Comparing pipeline runs

The screenshot shows the Google Cloud Platform Vertex AI Pipelines interface. The sidebar on the left has the 'Pipelines' option selected. The main area displays a table of pipeline runs:

Run	Status	Pipeline	Duration	Created	Ended	⋮
mlmd-pipeline-large-20211209213113	Succeeded	mlmd-pipeline	16 min 40 sec	Dec 9, 2021, 1:34:12 PM	Dec 9, 2021, 1:50:53 PM	⋮
mlmd-pipeline-small-20211209213113	Succeeded	mlmd-pipeline	18 min 21 sec	Dec 9, 2021, 1:33:05 PM	Dec 9, 2021, 1:51:31 PM	⋮

At the top of the main area, there are buttons for 'CREATE RUN', 'REFRESH', 'CLONE', 'COMPARE' (which is highlighted in yellow), 'STOP', and 'DELETE'.

The screenshot shows the Google Cloud Platform Vertex AI Pipelines interface. On the left, there's a sidebar with various options like Dashboard, Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The main area is titled "Pipelines" and shows a "Compare Runs" section. It lists two runs: "mlmd-pipeline-large-20211209213113" and "mlmd-pipeline-small-20211209213113". The "mlmd-pipeline-large" run has an accuracy of 99.97% and a dataset size of 13611, while the "mlmd-pipeline-small" run has an accuracy of 98.85% and a dataset size of 700. Both runs were created by "sara-vertex-demos.beans\_demo" and used the "large\_dataset" input. The interface also includes a "Parameters" table and a "Metrics" table.

## Comparing runs with the Vertex AI SDK

The screenshot shows a Jupyter Notebook interface with two code cells. The first cell displays logs from the Vertex AI API regarding pipeline creation and retrieval:

```

INFO:google.cloud.apitools.pipeline_jobs:Creating PipelineJob
INFO:google.cloud.apitools.pipeline_jobs:PipelineJob created. Resource name: projects/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-large-20211209213113
INFO:google.cloud.apitools.pipeline_jobs:To use this PipelineJob in another session:
INFO:google.cloud.apitools.pipeline_jobs:pipeline_job = apitools.PipelineJob.get('projects/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-large-20211209213113')
INFO:google.cloud.apitools.pipeline_jobs:View Pipeline Job:
https://console.cloud.google.com/vertex-ai/locations/us-central1/pipelines/runs/mlmd-pipeline-large-20211209213113?project=196373151126

```

The second cell uses the `apitools` library to retrieve the pipeline runs and prints their details:

```

[19]: df = apitools.get_pipeline_df(pipeline="mlmd-pipeline")

```

pipeline_name	run_name	param.input.project	param.input.region	param.input.output_data_path	param.input:bq_table	metric.dataset_size	metric.ac
0 mlmd-pipeline	mlmd-pipeline-large-20211209213113	cmpe260-334300	us-central1	data.csv	sara-vertex-demos.beans_demo.large_dataset	700.0	98.85
1 mlmd-pipeline	mlmd-pipeline-large-20211209213113	cmpe260-334300	us-central1	data.csv	sara-vertex-demos.beans_demo.large_dataset	13611.0	99.97

Below the notebook, a separate terminal window shows the command to run the notebook:

```

Simple 1 2 Python 3 | idle
File Edit View Run Kernel Git Tabs Settings Help
Untitled.ipynb
+ % C Code git Python 3
INFO:google.cloud.apitools.pipeline_jobs:Creating PipelineJob
INFO:google.cloud.apitools.pipeline_jobs:PipelineJob created. Resource name: projects/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-large-20211209213113
INFO:google.cloud.apitools.pipeline_jobs:To use this PipelineJob in another session:
INFO:google.cloud.apitools.pipeline_jobs:pipeline_job = apitools.PipelineJob.get('projects/196373151126/locations/us-central1/pipelineJobs/mlmd-pipeline-large-20211209213113')
INFO:google.cloud.apitools.pipeline_jobs:View Pipeline Job:
https://console.cloud.google.com/vertex-ai/locations/us-central1/pipelines/runs/mlmd-pipeline-large-20211209213113?project=196373151126

```

## Querying pipeline metrics

The screenshot shows a Jupyter Notebook interface with two tabs open: 'Untitled.ipynb' and 'XGboost.ipynb'. The 'Untitled.ipynb' tab contains Python code for querying pipeline metrics. The code uses the 'aiplatform\_v1' library to interact with Google Cloud AI Platform. It defines constants for the API endpoint and metadata client, sets up a request to list artifacts, applies filters for live artifacts, and then iterates through the results to build a DataFrame. The DataFrame has columns for uri, createTime, and type.

```
[21]: API_ENDPOINT = "{}-aiplatform.googleapis.com".format(REGION)
metadata_client = aiplatform_v1.MetadataServiceClient()
client_options = [
    "api_endpoint": API_ENDPOINT
]

[22]: MODEL_FILTER="schema_title = \"system.Model\""
artifact_request = aiplatform_v1.ListArtifactsRequest(
    parent="projects/{0}/locations/{1}/metadataStores/default".format(PROJECT_ID, REGION),
    filter=MODEL_FILTER
)
model_artifacts = metadata_client.list_artifacts(artifact_request)

[23]: LIVE_FILTER = "create_time > \"2021-08-10T00:00:00-00:00\" AND state = LIVE"
artifact_req = [
    "parent": "projects/{0}/locations/{1}/metadataStores/default".format(PROJECT_ID, REGION),
    "filter": LIVE_FILTER
]
live_artifacts = metadata_client.list_artifacts(artifact_req)

[24]: data = {'uri': [], 'createTime': [], 'type': []}

for i in live_artifacts:
    data['uri'].append(i.uri)
    data['createTime'].append(i.create_time)
    data['type'].append(i.schema_title)

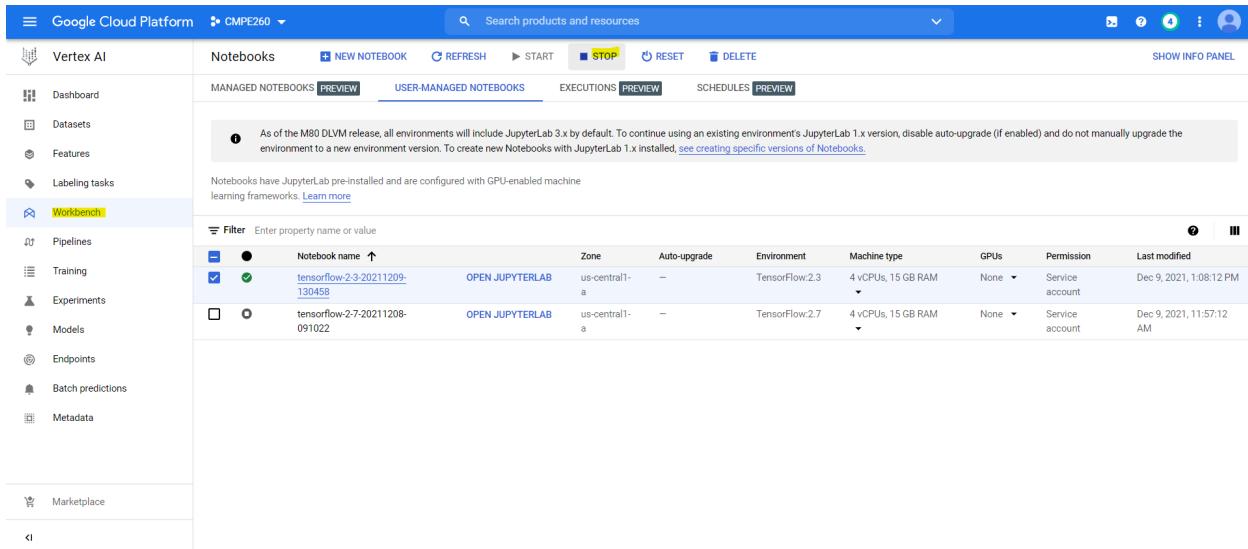
df = pd.DataFrame.from_dict(data)
df
```

The screenshot shows the same Jupyter Notebook interface after running the code from the previous screenshot. The 'Untitled.ipynb' tab now displays the resulting DataFrame. The DataFrame has three columns: 'uri', 'createTime', and 'type'. The data consists of 9 rows, each representing a different artifact or dataset. The 'uri' column contains URLs such as 'gs://cmpe260-334300-bucket/pipeline\_root/19637...' and 'projects/196373151126/locations/us-central1/mo...'. The 'createTime' column shows specific dates and times, like '2021-12-09 21:36:25.143000+0000' and '2021-12-09 21:39:09.359000+0000'. The 'type' column indicates the artifact type, with values like 'system.Metrics', 'system.Model', 'system.Artifact', and 'system.Dataset'.

	uri	createTime	type
0	gs://cmpe260-334300-bucket/pipeline_root/19637...	2021-12-09 21:36:25.143000+0000	system.Metrics
1	projects/196373151126/locations/us-central1/mo...	2021-12-09 21:39:09.359000+0000	system.Model
2	projects/196373151126/locations/us-central1/en...	2021-12-09 21:39:09.446000+0000	system.Artifact
3	gs://cmpe260-334300-bucket/pipeline_root/19637...	2021-12-09 21:37:28.343000+0000	system.Metrics
4	projects/196373151126/locations/us-central1/mo...	2021-12-09 21:41:43.955000+0000	system.Model
5	projects/196373151126/locations/us-central1/en...	2021-12-09 21:41:44.043000+0000	system.Artifact
6	gs://cmpe260-334300-bucket/pipeline_root/19637...	2021-12-09 21:37:28.260000+0000	system.Model
7	gs://cmpe260-334300-bucket/pipeline_root/19637...	2021-12-09 21:36:25.238000+0000	system.Model
8	gs://cmpe260-334300-bucket/pipeline_root/19637...	2021-12-09 21:34:13.835000+0000	system.Dataset
9	gs://cmpe260-334300-bucket/pipeline_root/19637...	2021-12-09 21:33:10.649000+0000	system.Dataset

# Cleanup

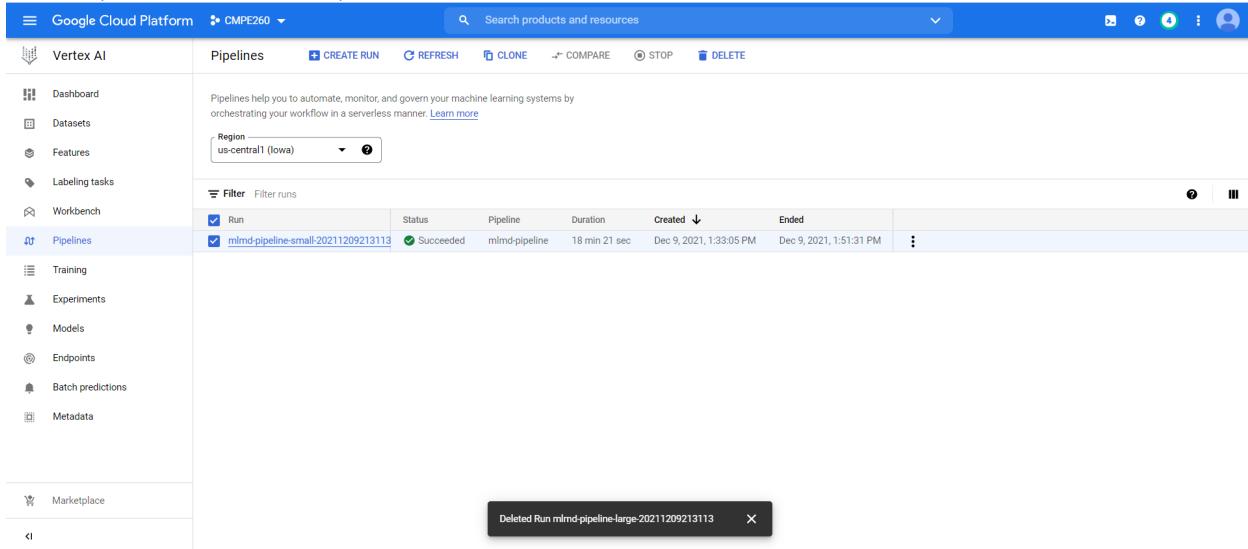
Stop or delete your Notebooks instance



The screenshot shows the Google Cloud Platform Vertex AI Workbench Notebooks interface. On the left, there's a sidebar with options like Dashboard, Datasets, Features, Labeling tasks, Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The Pipelines option is currently selected. The main area displays a table of managed notebooks:

Notebook name	Zone	Auto-upgrade	Environment	Machine type	GPUs	Permission	Last modified
tensorflow-2-3-20211209-130458	us-central1-a	—	TensorFlow:2.3	4 vCPUs, 15 GB RAM	None	Service account	Dec 9, 2021, 1:08:12 PM
tensorflow-2-7-20211208-091022	us-central1-a	—	TensorFlow:2.7	4 vCPUs, 15 GB RAM	None	Service account	Dec 9, 2021, 11:57:12 AM

Delete your Vertex AI endpoints



The screenshot shows the Google Cloud Platform Vertex AI Pipelines interface. The sidebar includes options like Dashboard, Datasets, Features, Labeling tasks, Workbench, Pipelines, Training, Experiments, Models, Endpoints, Batch predictions, and Metadata. The Pipelines option is selected. The main area shows a table of pipeline runs:

Run	Status	Pipeline	Duration	Created	Ended	⋮
mlmd-pipeline-small-20211209213113	Succeeded	mlmd-pipeline	18 min 21 sec	Dec 9, 2021, 1:33:05 PM	Dec 9, 2021, 1:51:31 PM	⋮

A confirmation message at the bottom states: "Deleted Run mlmd-pipeline-large-20211209213113".

Google Cloud Platform CMPE260 Search products and resources REFRESH

Vertex AI Endpoints + CREATE ENDPOINT

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.

To create an endpoint, you need at least one machine learning model.

Region: us-central1 (Iowa)

3 selected EDIT LABELS DELETE

Remove endpoint

Endpoints "beans-model-pipeline\_endpoint", "beans-model-pipeline\_endpoint" are not removable.

Your endpoints will no longer be available for online prediction requests.

Name ID Status Location Last updated API Notification Labels

Name	ID	Status	Location	Last updated	API	Notification	Labels
beans-model-pipeline_endpoint	141645	Active	us-central1	Dec 9, 2021, 1:50:03 PM	Sample request		
beans-model-pipeline_endpoint	3282992186957759464	Active	us-central1	Dec 9, 2021, 1:49:59 PM	Sample request		
titanic-endpoint	432424729064767488	Ready	us-central1	Dec 6, 2021, 10:02:08 PM	Sample request		

CANCEL CONFIRM

Marketplace

Google Cloud Platform CMPE260 Search products and resources SHOW INFO PANEL

Cloud Storage Browser + CREATE BUCKET DELETE REFRESH

Filter Filter buckets

Name	Created	Location type	Location	Default storage class	Last modified	Public access	Access control
cloud-ai-platform-65473780-0280-46f...	Dec 6, 2021, 4:17:06 PM	Region	us-central1 (Io...	Regional	Dec 6, 2021, 4:17:06 PM	Subject to object ACLs	Fine-grained
cmpe260-334300-bucket	Dec 9, 2021, 12:43:00 PM	Region	us-central1 (Io...	Standard	Dec 9, 2021, 12:43:00 PM	Subject to object ACLs	Fine-grained

Release Notes