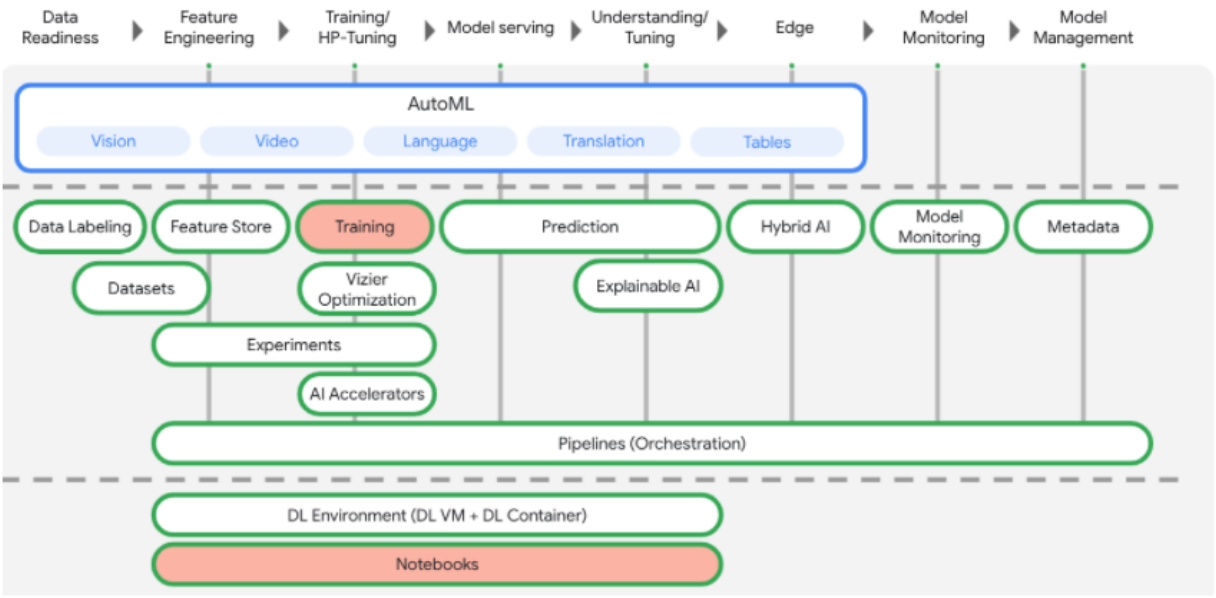
Assignment 7: optional catchup assignment 2 - VERTEX AI - for midterm and quiz - this will catch up midterm.

Multi-Worker Training and Transfer Learning with TensorFlow
Reference: https://codelabs.developers.google.com/vertex_multiworker_training#0

Objectives:

- Modify training application code for multi-worker training
- Configure and launch a multi-worker training job from the Vertex AI UI
- Configure and launch a multi-worker training job with the Vertex SDK

Vertex AI includes many different products to support end-to-end ML workflows. This lab will focus on the products highlighted below: Training and Notebooks



Use Case Overview

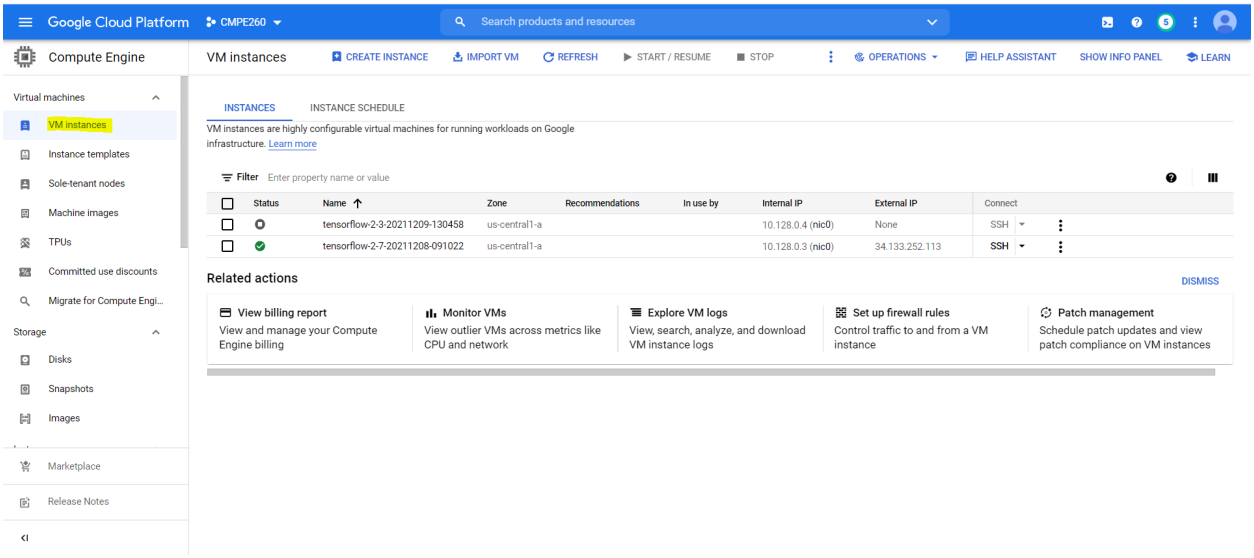
Use transfer learning to train an image classification model on the cassava dataset from TensorFlow Datasets. The architecture you'll use is a ResNet50 model from the tf.keras.applications library pretrained on the Imagenet dataset.

Why Distributed Training?

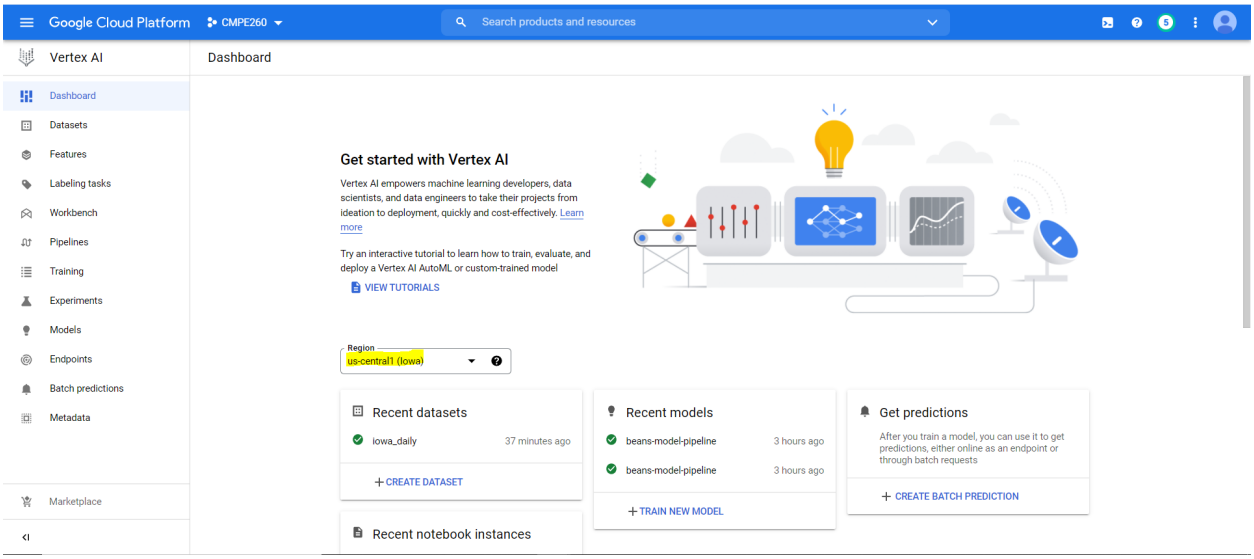
If you have a single GPU, TensorFlow will use this accelerator to speed up model training with no extra work on your part. However, if you want to get an additional boost from using multiple GPUs on a single machine or multiple machines (each with potentially multiple GPUs), then you'll need to use `tf.distribute`, which is TensorFlow's library for running a computation across multiple devices. A device refers to a CPU or accelerator, such as GPUs or TPUs, on some machine that TensorFlow can run operations on.

Set up your environment

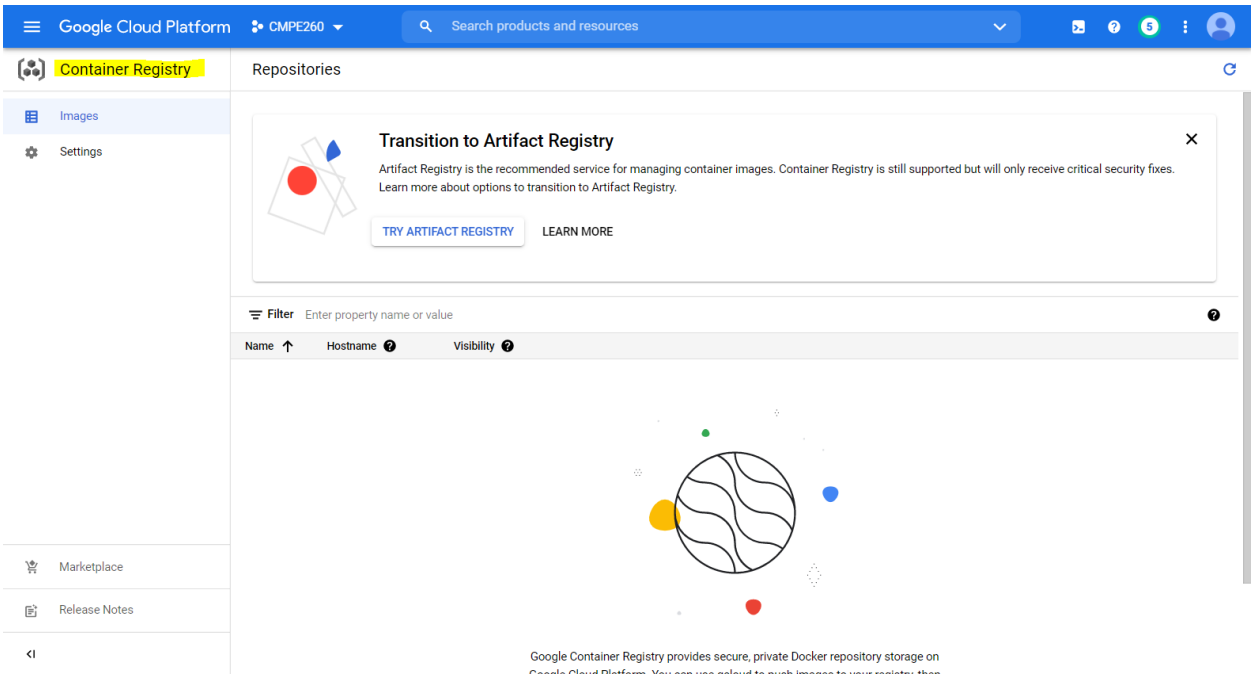
Enable the Compute Engine API



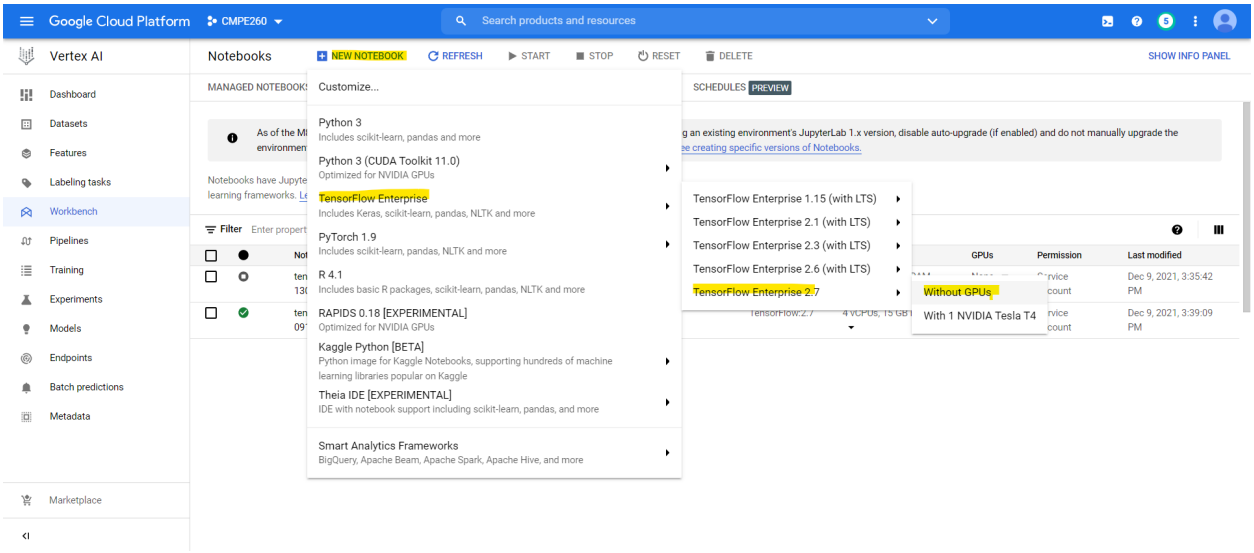
Enable the Vertex AI API



Enable the Container Registry API

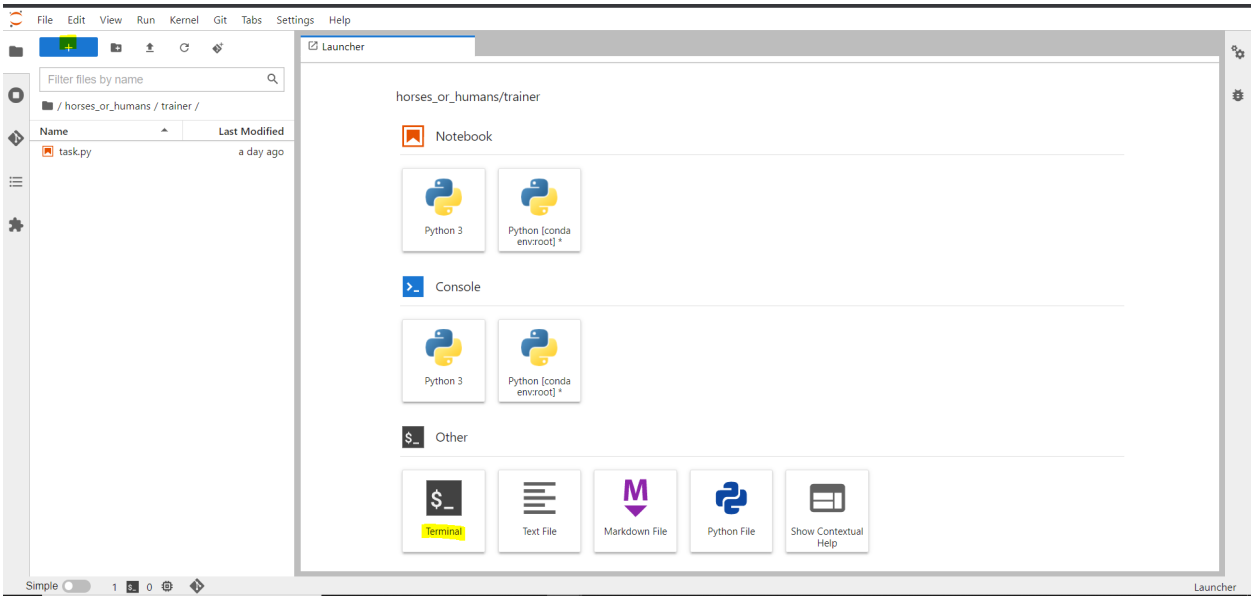


Create a Vertex AI Workbench instance

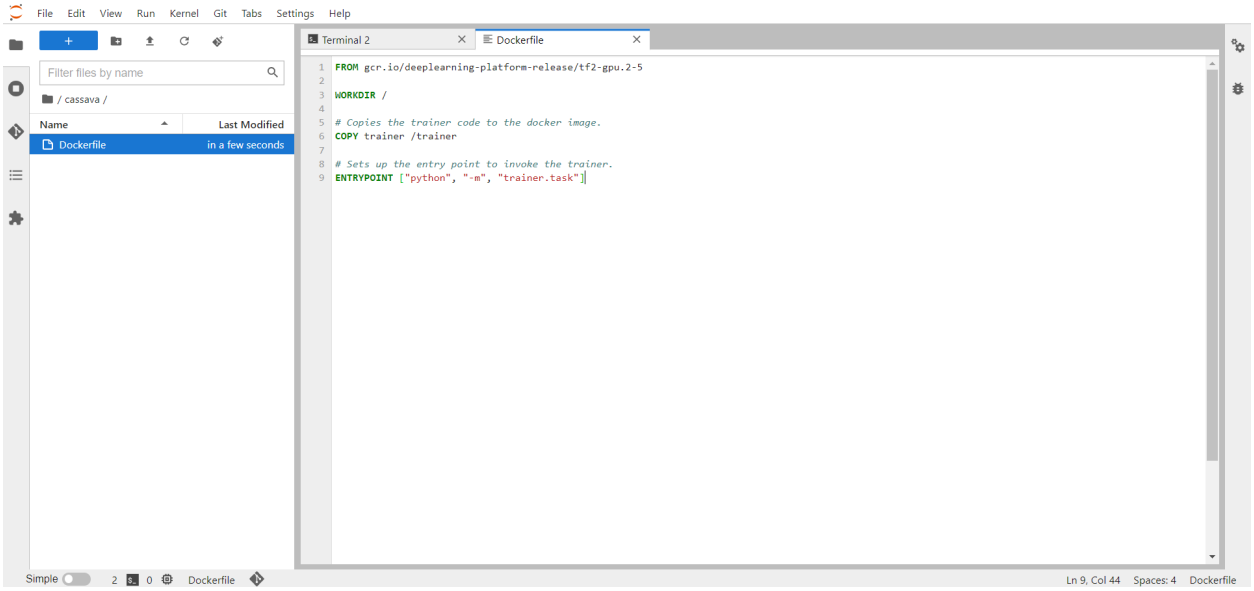


Containerize training application code

To start, from the Launcher menu, open a Terminal window in your notebook instance:



Step 1: Create a Dockerfile

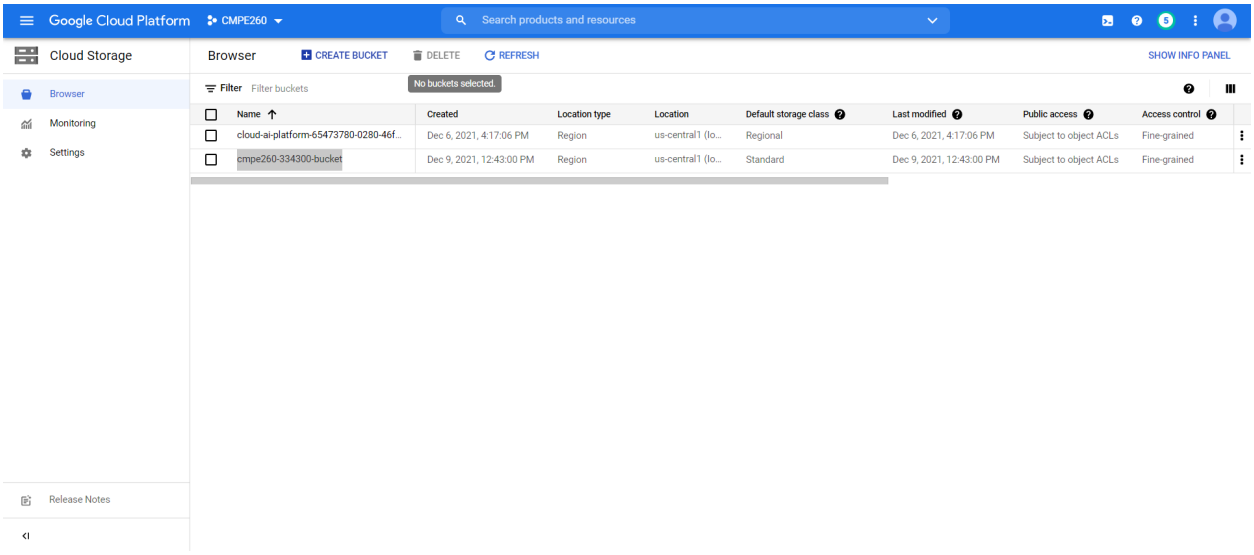


Step 2: Create a Cloud Storage bucket

Run the following commands, to created the Storage Bucket

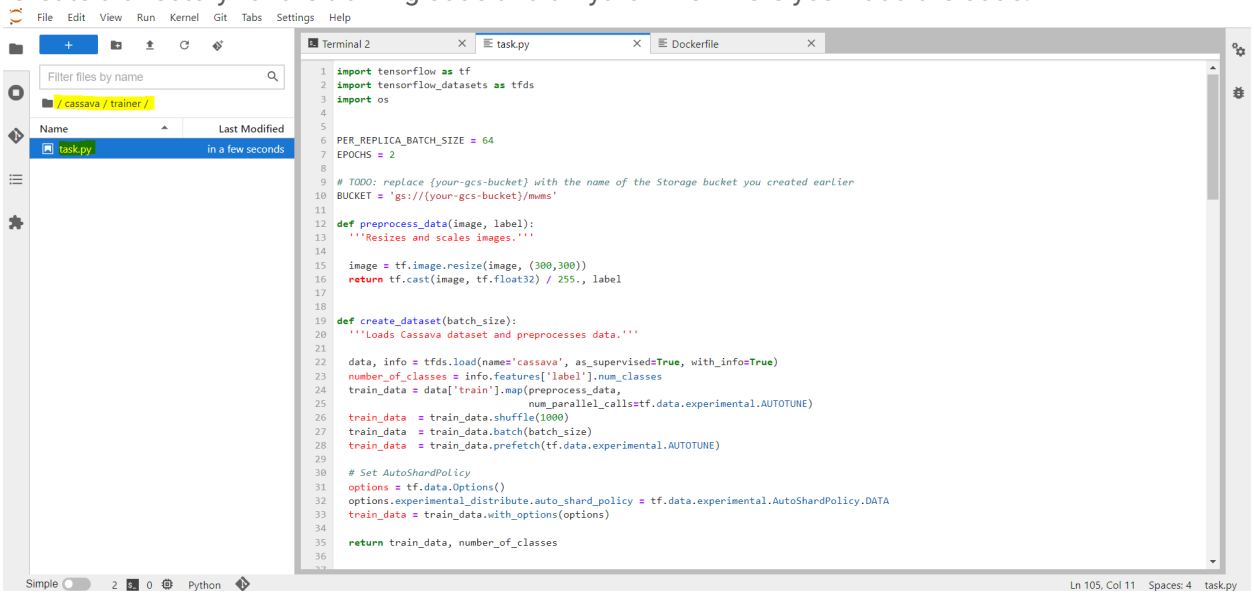
```
gcloud config list --format 'value(core.project)'
```

PROJECT_ID='your-cloud-project'
BUCKET="gs://${PROJECT_ID}-bucket"
gsutil mb -l us-central1 $BUCKET



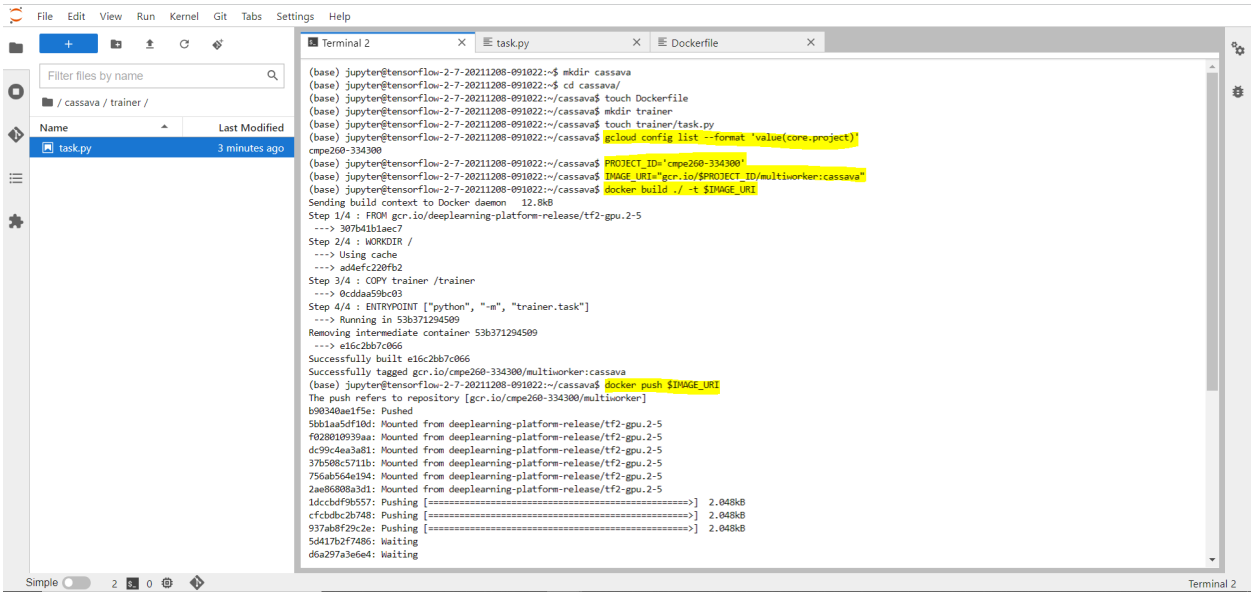
Step 3: Add model training code

Create a directory for the training code and a Python file where you'll add the code:



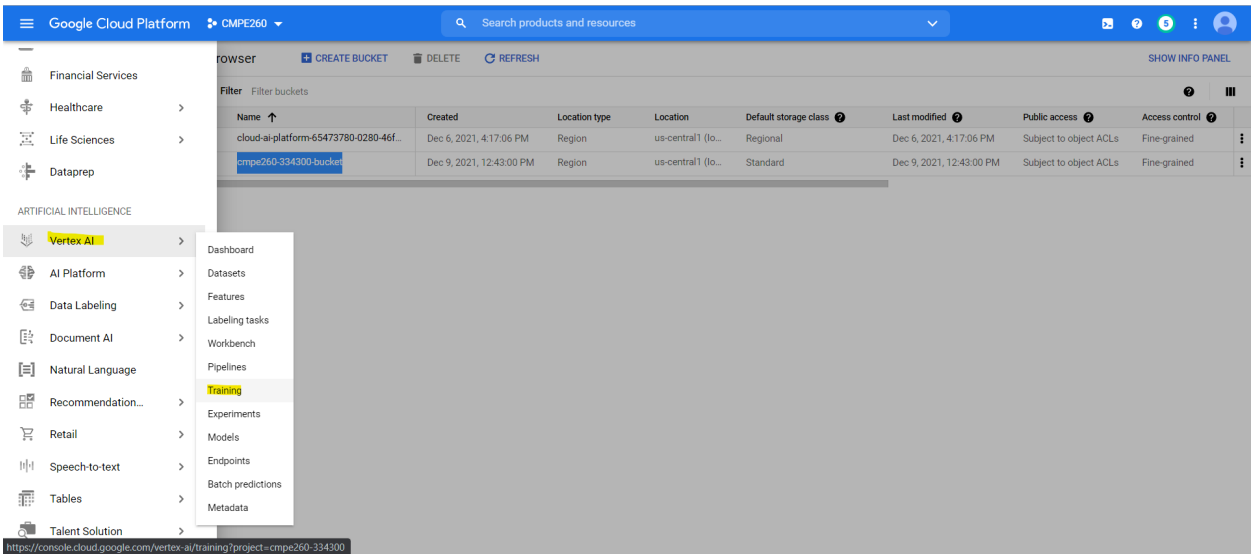
Step 4: Build the container

Run the following commands:
PROJECT_ID='your-cloud-project'
IMAGE_URI="gcr.io/$PROJECT_ID/multiworker:cassava"
docker build ./ -t $IMAGE_URI
docker push $IMAGE_URI



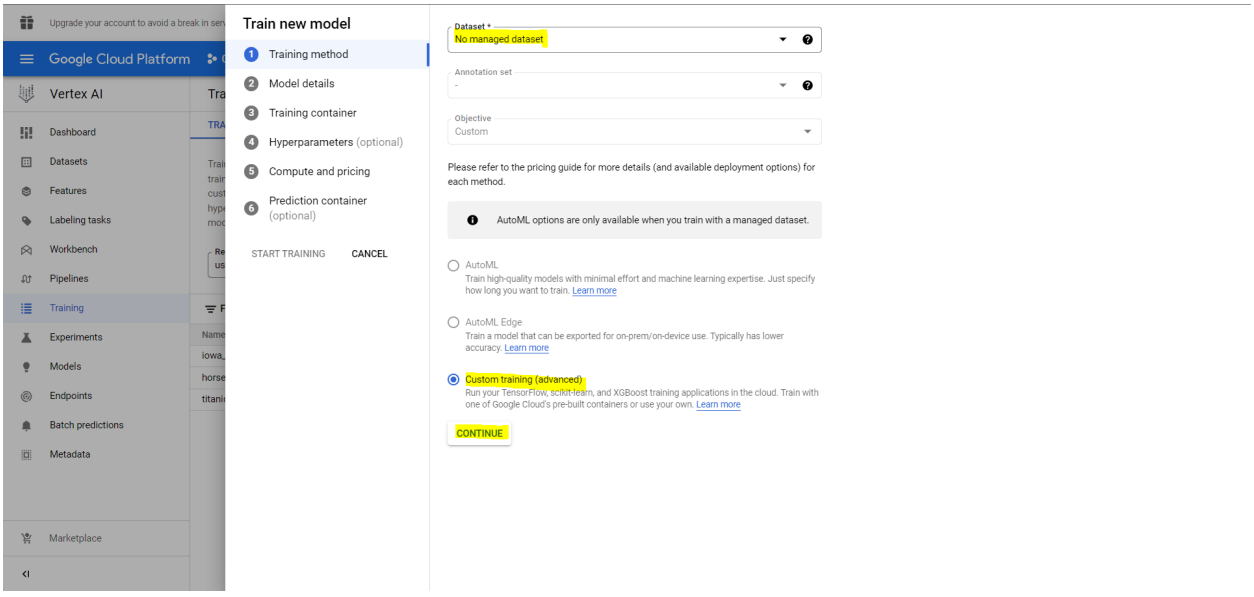
Run a multi-worker training job on Vertex AI

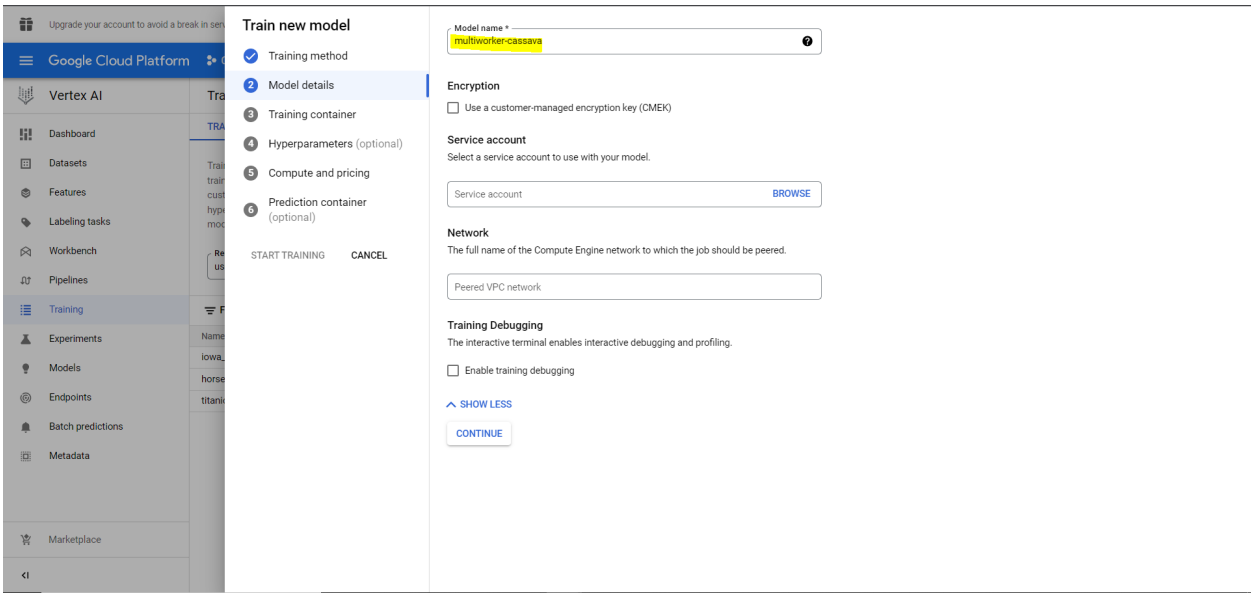
Navigate to Vertex AI → Training

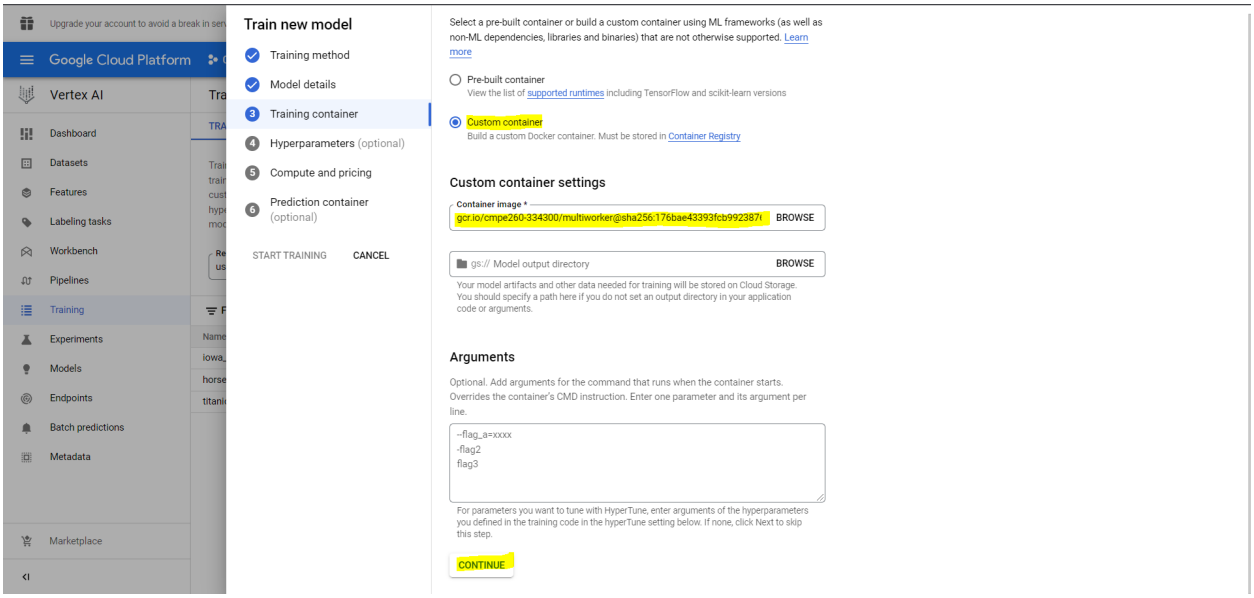


Step 1: Configure training job

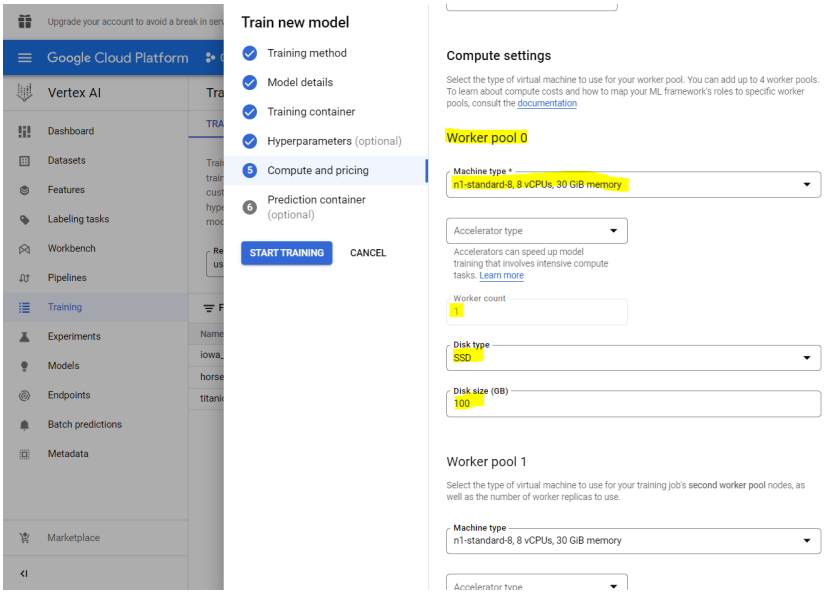
Click **Create** to enter the parameters for your training job.



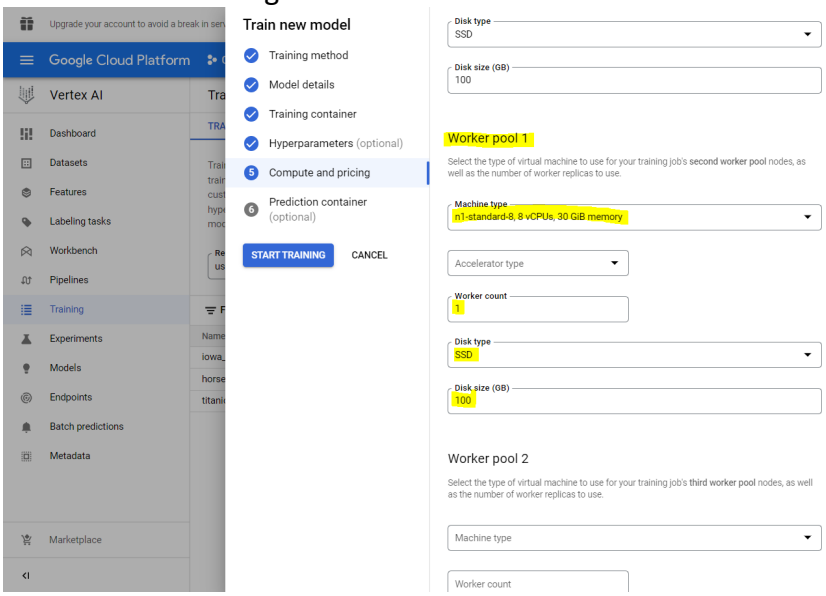

Upgrade your account to avoid a break in ser
Google Cloud Platform
Vertex AI
Dashboard
Datasets
Features
Labeling tasks
Workbench
Pipelines
Training
Experiments
Models
Endpoints
Batch predictions
Metadata
Marketplace
Train new model
Training method
Model details
Training container
Hyperparameters (optional)
Compute and pricing
Prediction container (optional)
START TRAINING
CANCEL
Model name *
multiworker-cassava
Encryption
Use a customer-managed encryption key (CMEK)
Service account
Select a service account to use with your model.
Service account
BROWSE
Network
The full name of the Compute Engine network to which the job should be peered.
Peered VPC network
Training Debugging
The interactive terminal enables interactive debugging and profiling.
Enable training debugging
SHOW LESS
CONTINUE


Upgrade your account to avoid a break in ser
Google Cloud Platform
Vertex AI
Dashboard
Datasets
Features
Labeling tasks
Workbench
Pipelines
Training
Experiments
Models
Endpoints
Batch predictions
Metadata
Marketplace
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 container using ML frameworks (as well as non-ML dependencies, libraries and binaries) that are not otherwise supported. Learn more
Pre-built container
View the list of supported runtimes including TensorFlow and scikit-learn versions
Custom container
Build a custom Docker container. Must be stored in Container Registry
Custom container settings
Container image *
gcr.io/cmpe260-334300/multiworker@sha256:176bae43393fcb992387
BROWSE
gs:// Model output directory
BROWSE
Your model artifacts and other data needed for training will be stored on Cloud Storage. You should specify a path here if you do not set an output directory in your application code or arguments.
Arguments
Optional. Add arguments for the command that runs when the container starts. Overrides the container's CMD instruction. Enter one parameter and its argument per line.
--flag_a=xxxx
-flag2
flag3
For parameters you want to tune with HyperTune, enter arguments of the hyperparameters you defined in the training code in the hyperTune setting below. If none, click Next to skip this step.
CONTINUE

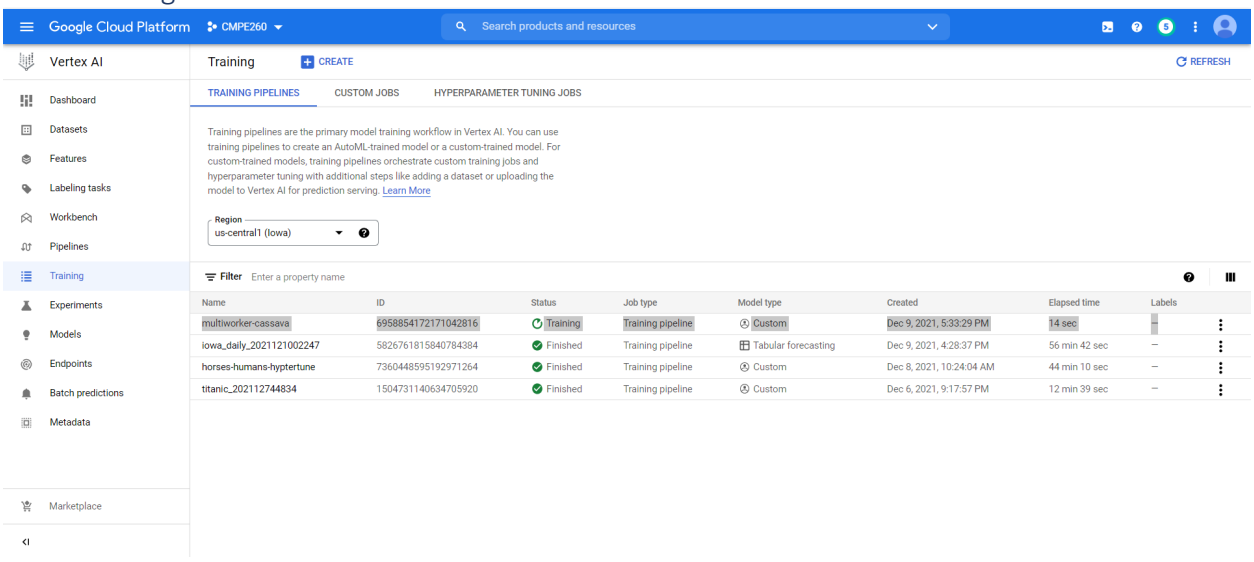
Worker Pool 0 Configuration:



Worker Pool 1 Configuration:



Start Training



Results