

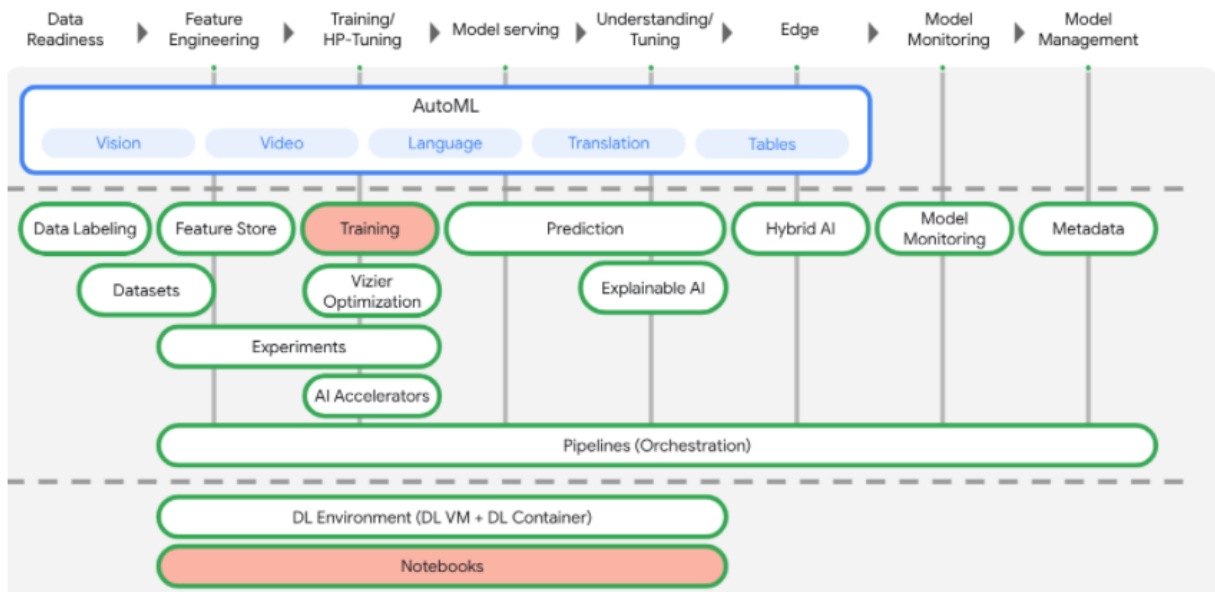
Multi-Worker Training and Transfer Learning with TensorFlow

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

The screenshot shows the Google Cloud Platform console with the Compute Engine API enabled. The left sidebar lists various services, with 'Compute Engine' selected. The main content area displays 'VM instances' with a table of active instances. Below the table, there are 'Related actions' such as 'View billing report', 'Monitor VMs', 'Explore VM logs', 'Set up firewall rules', and 'Patch management'.

Status	Name	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	tensorflow-2-3-20211209-130458	us-central1-a			10.128.0.4 (nic0)	None	SSH
<input checked="" type="checkbox"/>	tensorflow-2-7-20211208-091022	us-central1-a			10.128.0.3 (nic0)	34.133.252.113	SSH

Enable the Vertex AI API

The screenshot shows the Google Cloud Platform console with the Vertex AI API enabled. The left sidebar lists various services, with 'Vertex AI' selected. The main content area displays the 'Dashboard' with a 'Get started with Vertex AI' section. Below this, there are three panels: 'Recent datasets' (showing 'iowa_daily'), 'Recent models' (showing 'beans-model-pipeline'), and 'Get predictions'.

Get started with Vertex AI

Vertex AI empowers machine learning developers, data scientists, and data engineers to take their projects from idea to deployment, quickly and cost-effectively. [Learn more](#)

Try an interactive tutorial to learn how to train, evaluate, and deploy a Vertex AI AutoML or custom-trained model. [VIEW TUTORIALS](#)

Region: **us-central1 (Iowa)**

Recent datasets

- ☒ iowa_daily 37 minutes ago

[+ CREATE DATASET](#)

Recent models

- ☒ beans-model-pipeline 3 hours ago
- ☒ beans-model-pipeline 3 hours ago

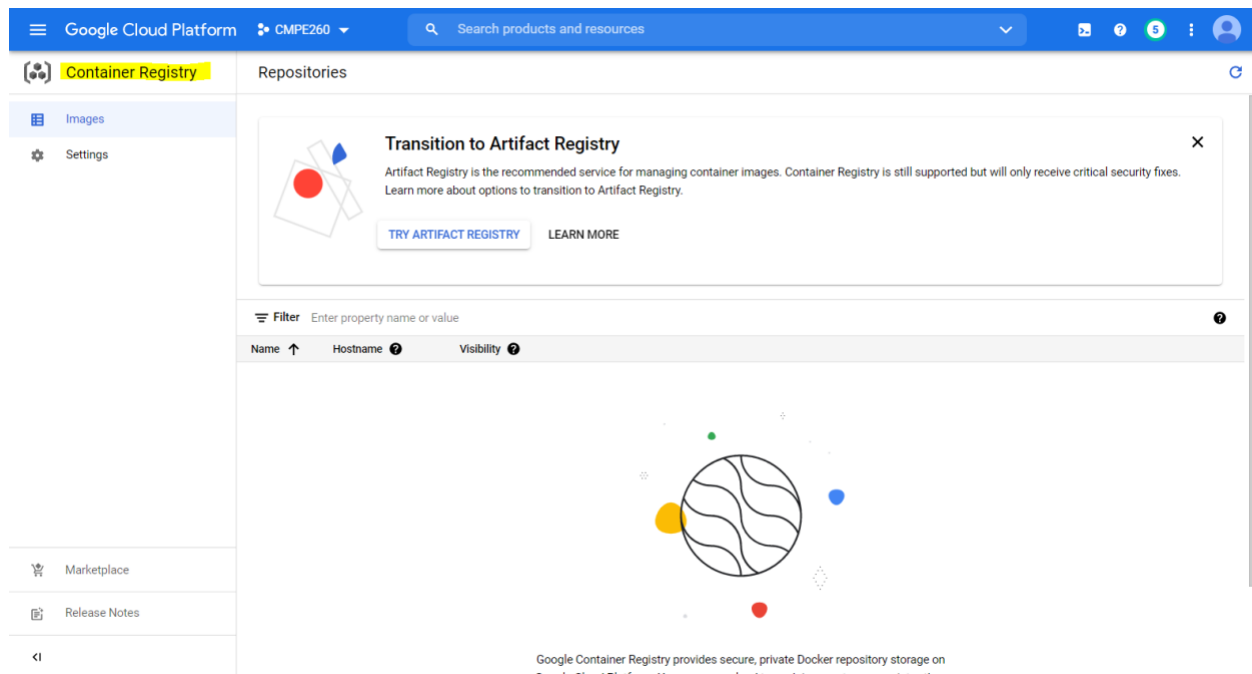
[+ TRAIN NEW MODEL](#)

Get predictions

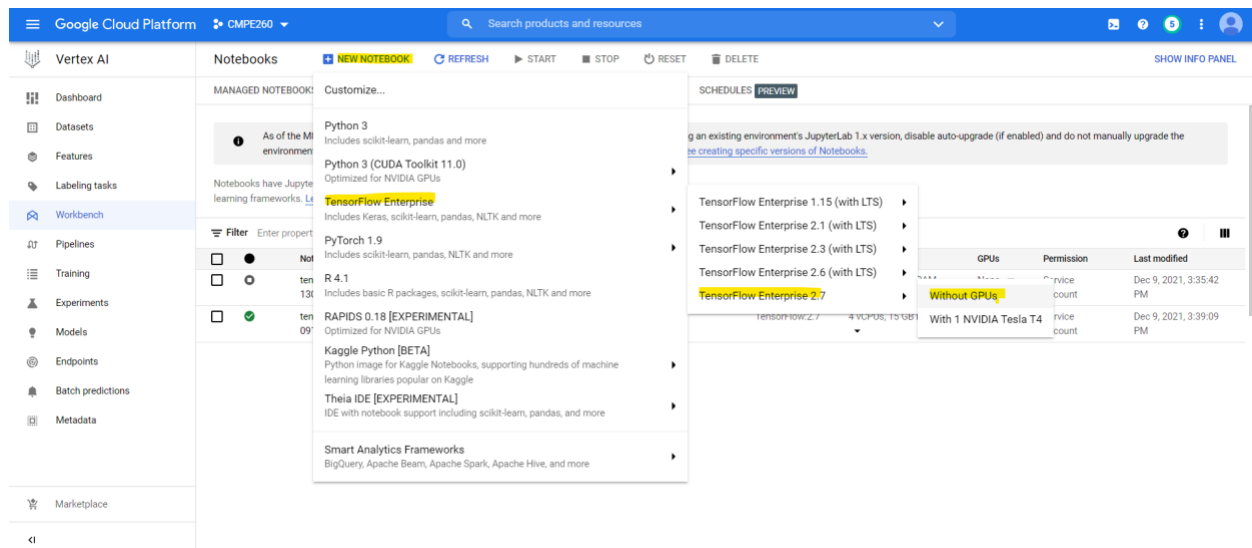
After you train a model, you can use it to get predictions, either online as an endpoint or through batch requests.

[+ CREATE BATCH PREDICTION](#)

Enable the Container Registry API

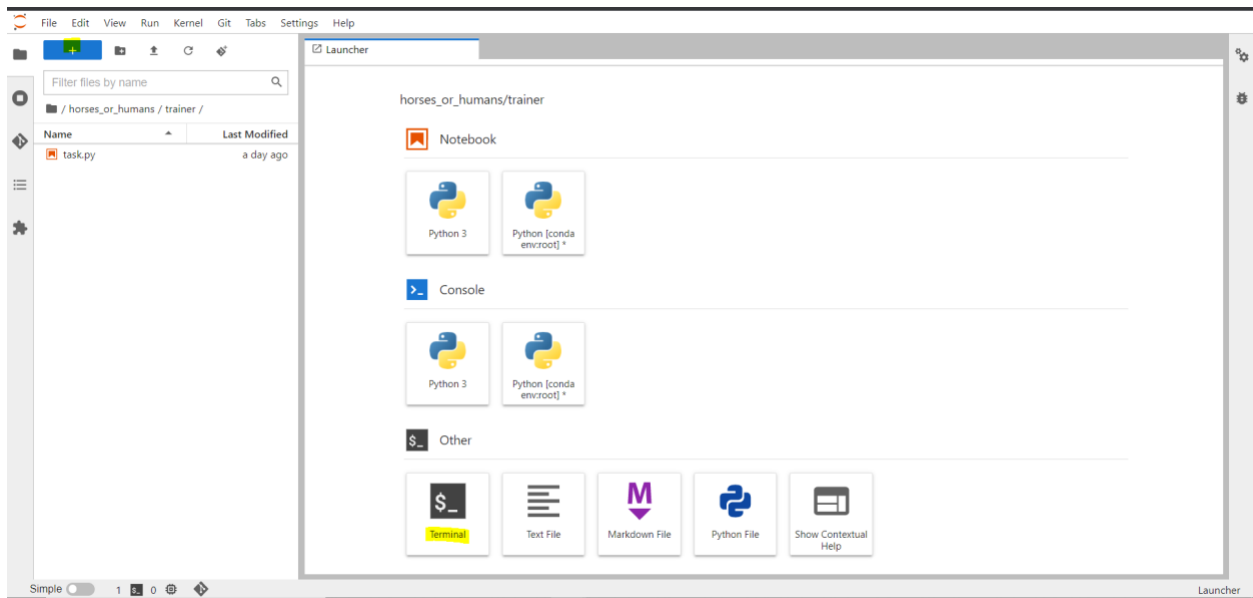


Create a Vertex AI Workbench instance

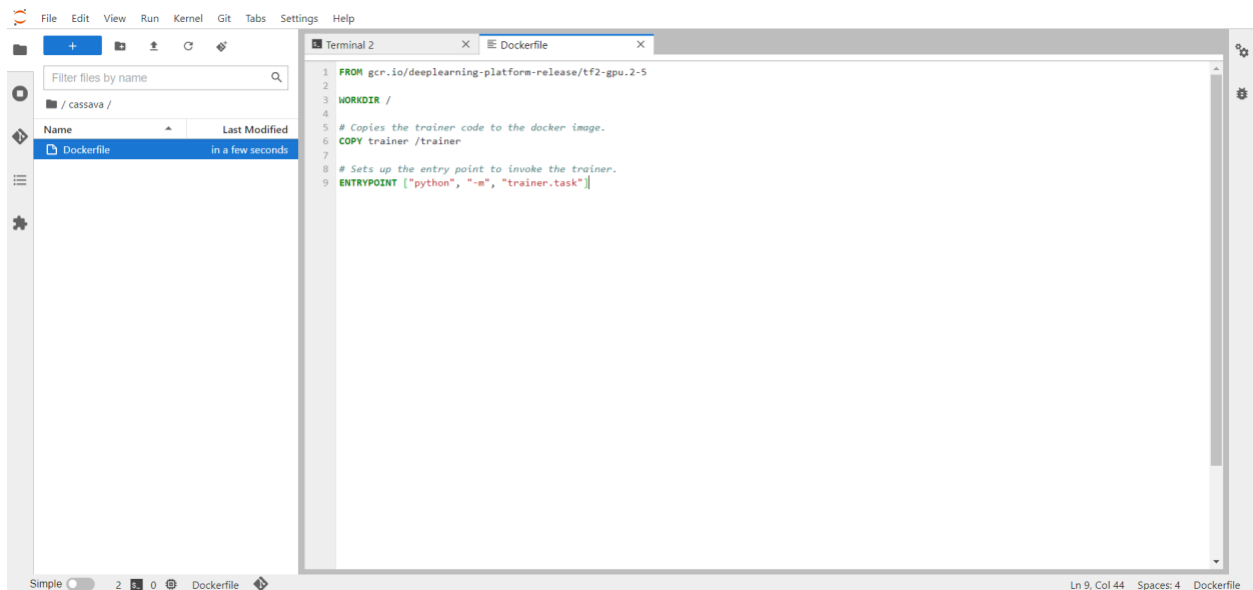


Containerize training application code

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



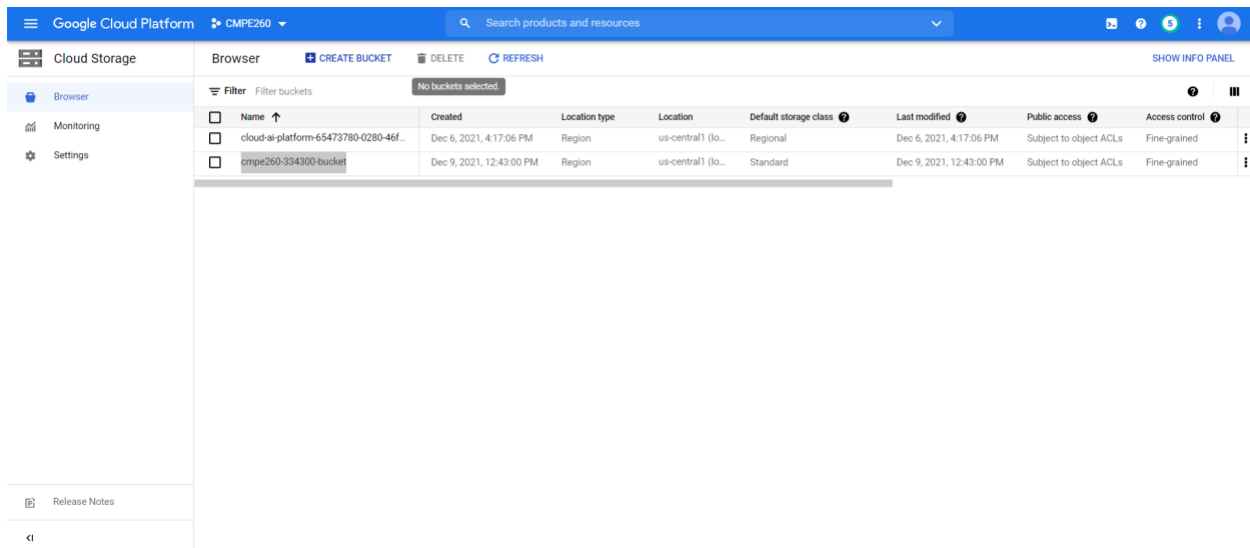
Step 1: Create a Dockerfile



Step 2: Create a Cloud Storage bucket

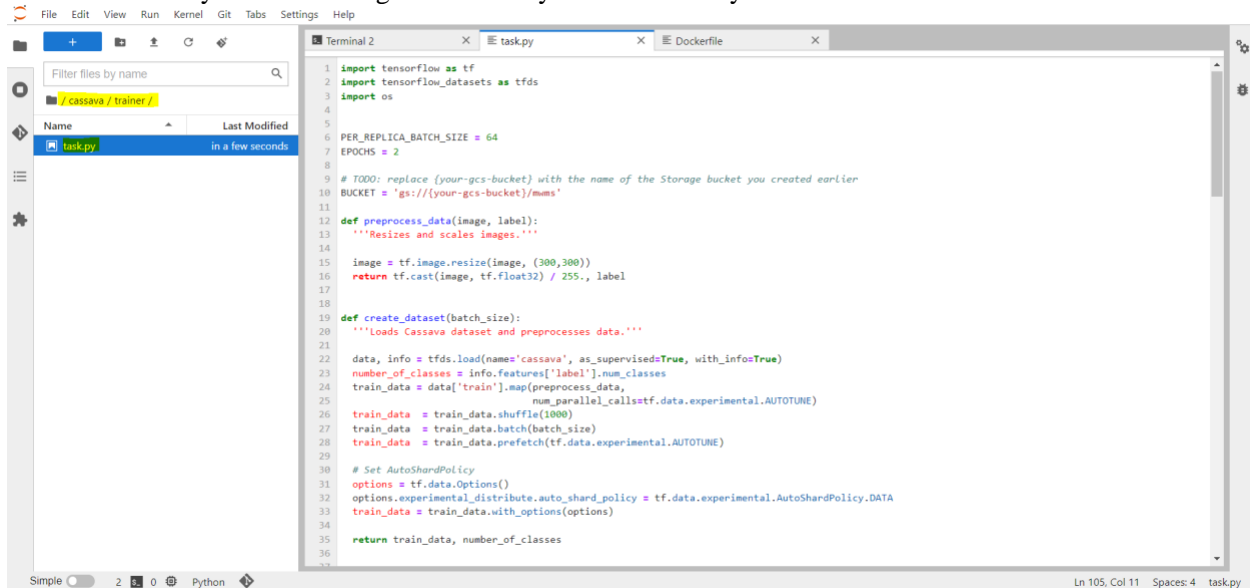
Run the following commands, to created the Storage Bucket

```
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"

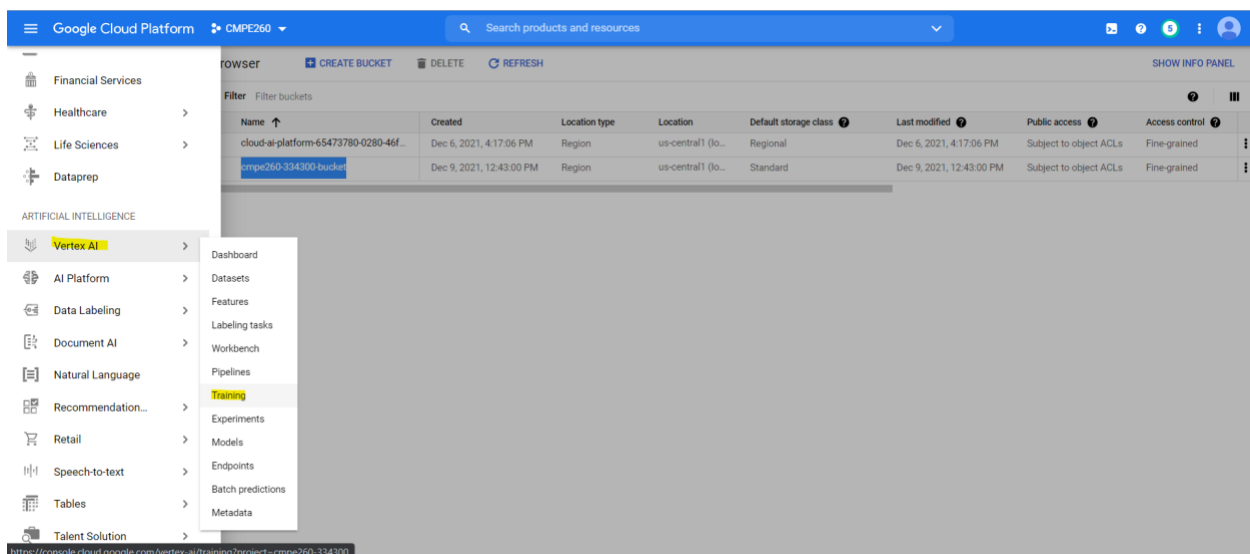
docker build ./ -t \$IMAGE_URI

docker push \$IMAGE_URI

```
(base) jupyter@tensorflow-2-7-20211208-091022:~$ mkdir cassava
(base) jupyter@tensorflow-2-7-20211208-091022:~$ cd cassava/
(base) jupyter@tensorflow-2-7-20211208-091022:~/cassava$ touch Dockerfile
(base) jupyter@tensorflow-2-7-20211208-091022:~/cassava$ mkdir trainer
(base) jupyter@tensorflow-2-7-20211208-091022:~/cassava$ touch trainer/task.py
(base) jupyter@tensorflow-2-7-20211208-091022:~/cassava$ gcloud config list --format 'value(core.project)'
cpe260-334300
(base) jupyter@tensorflow-2-7-20211208-091022:~/cassava$ PROJECT_ID="cpe260-334300"
(base) jupyter@tensorflow-2-7-20211208-091022:~/cassava$ IMAGE_URI="gcr.io/$PROJECT_ID/multiworker:cassava"
(base) jupyter@tensorflow-2-7-20211208-091022:~/cassava$ docker build . -t $IMAGE_URI
Sending build context to Docker daemon 12.8kB
Step 1/4 : FROM gcr.io/deeplearning-platform-release/tf2-gpu.2-5
--> 307b41b1ae7
Step 2/4 : WORKDIR /
--> Using cache
--> ad4efc220fb2
Step 3/4 : COPY trainer /trainer
--> 0cdda59bc05
Step 4/4 : ENTRYPOINT ["python", "-m", "trainer.task"]
--> Running in 53b371294509
Removing intermediate container 53b371294509
--> e16c2bb7c066
Successfully built e16c2bb7c066
Successfully tagged gcr.io/cpe260-334300/multiworker:cassava
(base) jupyter@tensorflow-2-7-20211208-091022:~/cassava$ docker push $IMAGE_URI
The push refers to repository [gcr.io/cpe260-334300/multiworker]
b90340aef15e: Pushed
5bb1a5d5f10d: Mounted from deeplearning-platform-release/tf2-gpu.2-5
f028018939aa: Mounted from deeplearning-platform-release/tf2-gpu.2-5
dc59c4a3a81: Mounted from deeplearning-platform-release/tf2-gpu.2-5
376508c5711b: Mounted from deeplearning-platform-release/tf2-gpu.2-5
756ab564e194: Mounted from deeplearning-platform-release/tf2-gpu.2-5
2ae8680a3d1: Mounted from deeplearning-platform-release/tf2-gpu.2-5
1dcdbf9b557: Pushing [=====] 2.045kB
cfc0dc2b748f: Pushing [=====] 2.045kB
937ab8f29c2e: Pushing [=====] 2.045kB
5d417b2f7486: Waiting
d6a297a3e6e4: Waiting
```

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 service

Google Cloud Platform

Vertex AI

Dashboard

Datasets

Features

Labeling tasks

Workbench

Pipelines

Training

Experiments

Models

Endpoints

Batch predictions

Metadata

Marketplace

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 details (and available deployment options) for each method.

1 AutoML options are only available when you train with a managed dataset.

☐ AutoML

Train high-quality models with minimal effort and machine learning expertise. Just specify how long you want to train. [Learn more](#)

☐ AutoML Edge

Train a model that can be exported for on-prem/on-device use. Typically has lower accuracy. [Learn more](#)

☒ Custom training (advanced)

Run your TensorFlow, scikit-learn, and XGBoost training applications in the cloud. Train with one of Google Cloud's pre-built containers or use your own. [Learn more](#)

CONTINUE

Upgrade your account to avoid a break in service

Google Cloud Platform

Vertex AI

Dashboard

Datasets

Features

Labeling tasks

Workbench

Pipelines

Training

Experiments

Models

Endpoints

Batch predictions

Metadata

Marketplace

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

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 service

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/compute240-334300/multoworker@sha256:176bae43393fcb9923871

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:

Upgrade your account to avoid a break in service

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

Compute settings

Select the type of virtual machine to use for your worker pool. You can add up to 4 worker pools. To learn about compute costs and how to map your ML framework's roles to specific worker pools, consult the [documentation](#)

Worker pool 0

Machine type *

n1-standard-8, 8 vCPUs, 30 GiB memory

Accelerator type

Accelerators can speed up model training that involves intensive compute tasks. [Learn more](#)

Worker count

1

Disk type

SSD

Disk size (GB)

100

Worker pool 1

Select the type of virtual machine to use for your training job's second worker pool nodes, as well as the number of worker replicas to use.

Machine type

n1-standard-8, 8 vCPUs, 30 GiB memory

Accelerator type

Worker Pool 1 Configuration:

Upgrade your account to avoid a break in service

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)

5 Compute and pricing

6 Prediction container (optional)

START TRAINING CANCEL

Disk type

SSD

Disk size (GB)

100

Worker pool 1

Select the type of virtual machine to use for your training job's second worker pool nodes, as well as the number of worker replicas to use.

Machine type

n1-standard-8, 8 vCPUs, 30 GB memory

Accelerator type

Worker count

1

Disk type

SSD

Disk size (GB)

100

Worker pool 2

Select the type of virtual machine to use for your training job's third worker pool nodes, as well as the number of worker replicas to use.

Machine type

Worker count

Start Training

Google Cloud Platform

CMPE260

Search products and resources

REFRESH

Vertex AI

Dashboard

Datasets

Features

Labeling tasks

Workbench

Pipelines

Training

Experiments

Models

Endpoints

Batch predictions

Metadata

Marketplace

Training

CREATE

TRAINING PIPELINES CUSTOM JOBS HYPERPARAMETER TUNING JOBS

Training pipelines are the primary model training workflow in Vertex AI. You can use training pipelines to create an AutoML-trained model or a custom-trained model. For custom-trained models, training pipelines orchestrate custom training jobs and hyperparameter tuning with additional steps like adding a dataset or uploading the model to Vertex AI for prediction serving. [Learn More](#)

Region us-central1 (Iowa)

Filter Enter a property name

Name	ID	Status	Job type	Model type	Created	Elapsed time	Labels
multiworker-cassava	6958854172171042816	Training	Training pipeline	Custom	Dec 9, 2021, 5:33:29 PM	14 sec	
iowa_daily_2021121002247	5826761815840784384	Finished	Training pipeline	Tabular forecasting	Dec 9, 2021, 4:28:37 PM	56 min 42 sec	
horses-humans-hypertune	7360448595192971264	Finished	Training pipeline	Custom	Dec 8, 2021, 10:24 04 AM	44 min 10 sec	
titanic_202112744834	1504731140634705920	Finished	Training pipeline	Custom	Dec 6, 2021, 9:17:57 PM	12 min 39 sec	

Results

