This tutorial shows you how to train the Mask RCNN model on Cloud TPU and GKE.

- Create a Cloud Storage bucket to hold your dataset and model output.
- Create a GKE cluster to manage your Cloud TPU resources.
- Download a Kubernetes job spec describing the resources needed to train the Mask RCNN model with TensorFlow on a Cloud TPU.
- Run the job in your GKE cluster, to start training the model.
- Check the logs and the model output.

This tutorial uses billable components of Google Cloud, including:

- Compute Engine
- Cloud TPU
- Cloud Storage

Use the pricing calculator (/products/calculator/) to generate a cost estimate based on your projected usage. New Google Cloud users might be eligible for a free trial (/free/).

1. Sign in (https://accounts.google.com/Login) to your Google Account.

   If you don't already have one, sign up for a new account (https://accounts.google.com/SignUp).
2. In the Cloud Console, on the project selector page, select or create a Cloud project.

★ **Note**: If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

Go to the project selector page (https://console.cloud.google.com/projectselector2/home/dashboard)

3. Make sure that billing is enabled for your Google Cloud project. Learn how to confirm billing is enabled for your project (/billing/docs/how-to/modify-project).

   When you use Cloud TPU with GKE, your project uses billable components of Google Cloud. Check the Cloud TPU pricing (/tpu/docs/pricing) and the GKE pricing (/kubernetes-engine/pricing) to estimate your costs, and follow the instructions to clean up resources (#cleanup) when you've finished with them.

4. Enable the APIs needed for this tutorial by opening the following links in the Cloud Console:

   - Cloud TPU API (https://console.cloud.google.com/apis/library/tpu.googleapis.com)
   - Compute Engine API (https://console.cloud.google.com/apis/library/compute.googleapis.com)
   - GKE API (https://console.cloud.google.com/apis/library/container.googleapis.com)

You need a Cloud Storage bucket to store the data you use to train your model and the training results. The `ctpu up` tool used in this tutorial sets up default permissions for the Cloud TPU service account. If you want finer-grain permissions, review the access level permissions (/tpu/docs/cloud-tpu-tools#create_resources).

1. Go to the Cloud Storage page on the Cloud Console.

   Go to the Cloud Storage page (https://console.cloud.google.com/storage)

2. Create a new bucket, specifying the following options:

   - A unique name of your choosing.
   - Default storage class: `Regional`
   - Choose where to store your data: `region` Location: `us-central1`
   - Choose a default storage class for your data: Standard
   - Choose how to control access to objects: Set object-level and bucket-level permissions

You need to give your Cloud TPU read/write access to your Cloud Storage objects. To do that, you must grant the required access to the service account used by the Cloud TPU. Follow the guide to grant access to your storage bucket (/tpu/docs/storage-buckets#storage_access).

Follow the instructions below to set up your environment and create a GKE cluster with Cloud TPU support, using the `gcloud` command-line tool.

1. Open a Cloud Shell window.

   Open Cloud Shell (https://console.cloud.google.com/?cloudshell=true)

2. Specify your Google Cloud project:

   where `YOUR-CLOUD-PROJECT`is the name of your Google Cloud project.

3. Specify the zone where you plan to use a Cloud TPU resource. For this example, use the `us-central1-b` zone:

4. Use the `gcloud container clusters` (/sdk/gcloud/reference/container/clusters/) command to create a cluster on GKE with support for Cloud TPU. Note that the GKE cluster and its node-pools must be created in a zone where Cloud TPU is available, as described in the section on environment variables (#set-variables) above. The following command creates a cluster named `tpu-models-cluster`:

   In the above command:

- `--cluster-version=1.13` indicates that the cluster will use the latest Kubernetes 1.13 release. You must use version `1.13.4-gke.5` or later.
- `--scopes=cloud-platform` ensures that all nodes in the cluster have access to your Cloud Storage bucket in the Google Cloud defined as `YOUR-CLOUD-PROJECT` above. The cluster and the storage bucket must be in the same project. Note that the Pods by default inherit the scopes of the nodes to which they are deployed. This flag gives all Pods running in the cluster the `cloud-platform` scope. If you want to limit the access on a per Pod basis, see the GKE guide to authenticating with service accounts (/kubernetes-engine/docs/tutorials/authenticating-to-cloud-platform).
- `--enable-ip-alias` indicates that the cluster uses alias IP ranges (/kubernetes-engine/docs/how-to/alias-ips). This is required for using Cloud TPU on GKE.
- `--enable-tpu` indicates that the cluster must support Cloud TPU.
- `machine-type=n1-standard-4` is required to run this job.

When the command has finished running a confirmation message appears, similar to this:

Your next task is to create a **job**. In Google Kubernetes Engine, a job is a controller object that represents a finite task. The first job you need to create downloads and processes the COCO dataset used to train the Mask RCNN model.

1. In your shell environment, create a file named `download_and_preprocess_coco_k8s.yaml` as shown below. Alternatively, you can download this file from GitHub.

tools/datasets/download_and_preprocess_coco_k8s.yaml (https://github.com/tensorflow/tpu/blob/master/tools/datasets/download_and_preprocess_coco_k8s.yaml)

GitHub (https://github.com/tensorflow/tpu/blob/master/tools/datasets/download_and_preprocess_coco_k8s.yaml)

2. Locate the environment variable named `DATA_BUCKET`. Update the `value` for this variable to the path where you want to store the COCO dataset on the Cloud Storage bucket. For example: "gs://my-maskrcnn-bucket/coco/"
3. Run the following command to create the job in your cluster:

Jobs can take a few minutes to start. You can view the status of the jobs running on your cluster by using the following command:

This command returns information about all jobs running in the cluster. For example

Note: Google Kubernetes Engine provides a unique identifier (UID) to each job, which it appends to the job name specified in the `.yaml` file.

The `-w` flag instructs the command to watch for any changes in a pod's status. After a few minutes, the status for your job should update to `Running`.

If you encounter any issues with your job, you can delete it and try again. To do so, run the `kubectl delete -f download_and_preprocess_coco_k8s.yaml` command before running the `kubectl create` command again.

When the download and preprocessing completes, you are ready to run the model.

Everything is now in place for you to run the Mask RCNN model using Cloud TPU and GKE.

1. In your shell environment, create a file named `mask_rcnn_k8s.yaml` as shown below. Alternatively, you can download this file from GitHub.

   models/experimental/mask_rcnn/mask_rcnn_k8s.yaml (https://github.com/tensorflow/tpu/blob/r1.13/models/experimental/mask_rcnn/mask_rcnn_k8s.yaml)

   w on GitHub (https://github.com/tensorflow/tpu/blob/r1.13/models/experimental/mask_rcnn/mask_rcnn_k8s.yaml)

2. Locate the environment variable named `DATA_BUCKET`. Update the `value` for this variable to the path to the `DATA_BUCKET` file you specified in the `download_and_preprocess_coco_k8s.yaml` script.
3. Locate the environment variable named `MODEL_BUCKET`. Update the `value` for this variable to the path to your Cloud Storage bucket. This location is where the script saves the training output. If the specified `MODEL_BUCKET` does not exist, it is created when you run the job.
4. Run the following command to create the job in your cluster.

   When you run this command, the following confirmation message appears:

As mentioned in the section, Process the training data (#process_the_training_data), it can take a few minutes for a job to start. View the status of the jobs running on your cluster by using the following command:

You can track the training status by using the `kubectl` command-line utility.

1. Run the following command to get the status of the job:

   During the training process, the status of the pod should be `RUNNING`.

   ★ **Note:** While uncommon, jobs and pods can fail. If you see a status other than `RUNNING`, you might need to delete the job and re-create it.

2. To get additional information on the training process, run the following command:

   Where `job name` is the full name of the job, for example, `mask-rcnn-gke-tpu-abcd`.

   You can also check the output on the GKE Workloads dashboard (https://console.cloud.google.com/kubernetes/workload) on the Cloud Console.

   Note that it takes a while for the first entry to appear in the logs. You can expect to see something like this:

3. View the trained model at `gs://<my-model-bucket>/mask-rcnn/model.ckpt`.

When you've finished with Cloud TPU on GKE, clean up the resources to avoid incurring extra charges to your Google Cloud account.

If you haven't set the project and zone for this session, do so now. See the instructions (#set-variables) earlier in this guide. Then follow this cleanup procedure:

1. Run the following command to delete your GKE cluster, `tpu-models-cluster`, replacing `YOUR-PROJECT` with your Google Cloud project name:

2. When you've finished examining the data, use the `gsutil` command to delete the Cloud Storage bucket you created during this tutorial. Replace `YOUR-BUCKET` with the name of your Cloud Storage bucket:

See the Cloud Storage pricing guide (/storage/pricing) for free storage limits and other pricing information.

- Explore the TPU tools in TensorBoard (/tpu/docs/cloud-tpu-tools).
- Run more models and dataset retrieval jobs using one of the following job specs:
    - Download and preprocess the COCO dataset (https://github.com/tensorflow/tpu/tree/master/tools/datasets/download_and_preprocess_coco_k8s.yaml) on GKE.
    - Download and preprocess ImageNet (https://github.com/tensorflow/tpu/tree/master/tools/datasets/imagenet_to_gcs_k8s.yaml) on GKE.
    - Train AmoebaNet-D (https://github.com/tensorflow/tpu/tree/master/models/official/amoeba_net/amoeba_net_k8s.yaml) using Cloud TPU and GKE.
    - Train Inception v3 (https://github.com/tensorflow/tpu/tree/master/models/experimental/inception/inception_v3_k8s.yaml) using Cloud TPU and GKE.
    - Train RetinaNet (https://github.com/tensorflow/tpu/tree/master/models/official/retinanet/retinanet_k8s.yaml) using Cloud TPU and GKE.
- Experiment with more TPU samples (/tpu/docs/samples/).