This tutorial shows you how to train the TensorFlow ResNet-50 (https://github.com/tensorflow/tpu/tree/master/models/official/resnet) model on Cloud TPU and GKE.

In summary, the tutorial leads you through the following steps to run the model, using a fake data set provided for testing purposes:

- Create a Cloud Storage bucket to hold your model output.
- Create a GKE cluster to manage your Cloud TPU resources.
- Download a Kubernetes Job spec describing the resources needed to train ResNet-50 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.

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 following APIs on 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 can use either Cloud Shell (/shell/docs/) or your local shell to complete this tutorial. Cloud Shell comes preinstalled with the gcloud (/sdk/gcloud/) and kubectl (https://kubernetes.io/docs/user-guide/kubectl-overview/) command-line tools. `gcloud` is the command-line interface for Google Cloud, and `kubectl` provides the command-line interface for running commands against Kubernetes clusters.

If you prefer using your local shell, you must install the `gcloud` and `kubectl` command-line tools in your environment.

Note the following when defining your configuration:

- You must use GKE version `1.13.4-gke.5` or later. You can specify the version by adding the `--cluster-version` parameter to the `gcloud container clusters create` (/sdk/gcloud/reference/container/clusters/create) command as described below (#cluster). For more information, see the SDK documentation (/sdk/gcloud/reference/container/clusters/).
- You must use TensorFlow 1.13 or later. You can specify the TensorFlow version in your Kubernetes pod specification like the resnet config (https://github.com/tensorflow/tpu/tree/master/models/official/resnet/resnet_k8s.yaml).
- You must create your GKE cluster and node pools in a zone where Cloud TPU is available. You must also create the Cloud Storage buckets to hold your training data and models in the same region as where you created the GKE cluster. The following zones are available:

- Each container can request at most one Cloud TPU, but multiple containers in a Pod can request a Cloud TPU each.
- Cluster Autoscaler supports Cloud TPU on GKE 1.11.4-gke.12 and later.

You can use a fake data set provided with this tutorial or the full ImageNet data to train your model. Either way, you need to set up a Cloud Storage bucket as described below.

The instructions below assume that you want to use a randomly generated fake dataset to test the model. Alternatively, you can follow the instructions for using the full ImageNet dataset (https://github.com/tensorflow/tpu/blob/master/tools/datasets/imagenet_to_gcs_k8s.yaml).

The fake dataset is at this location on Cloud Storage:

Note that the fake dataset is only useful for understanding how to use a Cloud TPU, and validating end-to-end performance. The accuracy numbers and saved model will not be meaningful.

You can read from **`gs://cloud-tpu-test-datasets`** but you can't write to it. As a result, you can't use it to write o g logs. Make sure you use your own bucket to store the output model and training logs.

You need a Cloud Storage bucket to store the results of training your machine learning model. If you decide to use real training data rather than the fake dataset provided with this tutorial, you can store the data in the same bucket.

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: `Standard`
    - Location: `us-central1`

    The bucket location must be in the same region as the TPU resource provisioned in the GKE cluster.

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).

You can create a GKE cluster on the Cloud Console or using the `gcloud` command-line tool. Select an option below to see the relevant instructions:

Everything is now in place for you to run the ResNet-50 model using Cloud TPU and GKE.

1. Create a Job (/kubernetes-engine/docs/how-to/jobs) spec in a file named `resnet_k8s.yaml`:
    - Download or copy the prepared Job spec from GitHub (https://github.com/tensorflow/tpu/tree/master/models/official/resnet/resnet_k8s.yaml).
    - In the Job spec, change `<my-model-bucket>` to the name of the Cloud Storage bucket you created earlier.

    Note that the Job spec references the TensorFlow TPU models that are available in a Docker (https://docker.com/) container at gcr.io/tensorflow/tpu-models

(https://console.cloud.google.com/gcr/images/tensorflow/GLOBAL/tpu-models). (That's a location on Container Registry (/container-registry/).)

2. Create the Job in the GKE cluster:

3. Wait for the Job to be scheduled.

   The lifetime of Cloud TPU nodes is bound to the Pods that request them. The Cloud TPU is created on demand when the Pod is scheduled, and recycled when the Pod is deleted.

   It takes about 5 minutes for the Pod scheduling to finish.

   After 5 minutes, you should see something like this:

4. Check the Pod logs to see how the job is doing:

   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:

5. View the trained model at `gs://<my-model-bucket>/resnet/model.ckpt`. You can see your buckets on the Cloud Storage browser page (https://console.cloud.google.com/storage/browser) on the Cloud Console.

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

- 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/).