This page is a quick guide to setting up Cloud TPU with Google Kubernetes Engine (/kubernetes-engine/docs/). If you're looking for a detailed walkthrough, follow the tutorial (/tpu/docs/tutorials/kubernetes-engine-resnet) which shows you how to train the TensorFlow ResNet-50 model using Cloud TPU and GKE.

GKE offers Kubernetes clusters as a managed service.

- **Easier setup and management:** When you use Cloud TPU, you need a Compute Engine VM to run your workload, and a Classless Inter-Domain Routing (CIDR) block for Cloud TPU. GKE sets up and manages the VM and the CIDR block for you.
- **Optimized cost:** GKE scales your VMs and Cloud TPU nodes automatically based on workloads and traffic. You only pay for Cloud TPU and the VM when you run workloads on them.
- **Flexible usage:** It's a one-line change in your Pod spec to request a different hardware accelerator (CPU, GPU, or TPU):

- **Scalability:** GKE provides APIs (Job (/kubernetes-engine/docs/how-to/jobs#scaling_a_job) and Deployment (/kubernetes-engine/docs/how-to/stateless-apps#scale)) that can easily scale to hundreds of Pods and Cloud TPU nodes.
- **Fault tolerance:** GKE's Job API (/kubernetes-engine/docs/how-to/jobs), along with the TensorFlow checkpoint mechanism, provide the run-to-completion semantic. Your training jobs will automatically rerun with the latest state read from the checkpoint if failures occur on the VM instances or Cloud TPU nodes.

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). See more information about the version in the SDK documentation (/sdk/gcloud/reference/container/clusters/create).
- You must use TensorFlow 1.13 or later. You should specify the TensorFlow version used by the Cloud TPU in your Kubernetes Pod spec, as described below (#pod-spec).
- 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 your 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 above.

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 need a Cloud Storage bucket to store the results of training your machine learning model.

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:

You can use either an official TPU model that has been containerized in Docker images, or build and containerize your own model.

- Use the official TPU models

  The latest official TPU models (https://github.com/tensorflow/tpu) are containerized in Docker images (http://gcr.io/tensorflow/tpu-models). These Docker images were built with a Dockerfile (https://github.com/tensorflow/tpu/blob/master/tools/docker/Dockerfile). The following sections use the official TPU model Docker images.

- Build your own models

  If you want to build your own model to run Cloud TPU on GKE, you can utilize Cloud TPU in TensorFlow with tf.contrib.cluster_resolver.TPUClusterResolver (https://www.tensorflow.org/api_docs/python/tf/contrib/cluster_resolver/TPUClusterResolver) and tf.contrib.tpu.TPUEstimator (https://www.tensorflow.org/api_docs/python/tf/contrib/tpu/TPUEstimator), as follows:

For more information, see the TensorFlow documentation (https://www.tensorflow.org/guide/using_tpu) on how to use Cloud TPUs in TensorFlow.

You can follow steps 1 and 2 in Deploying a containerized web application (https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app) to containerize your model in a Docker image and push it to the Google Container Registry (https://cloud.google.com/container-registry).

In your Pod spec:

- Use the following Pod annotation to specify the TensorFlow version that the Cloud TPU nodes use:

Where `x.y` is the TensorFlow version supported by the Cloud TPU. You must use TensorFlow 1.13 or above. All Cloud TPU instances created for a Pod must use the same TensorFlow version. You must build your models in your containers using the same TensorFlow version. See the supported versions (/tpu/docs/supported-versions).

- Specify the Cloud TPU resource in the `limits` section under the `resource` field in the container spec.

  Note that the unit of the Cloud TPU resource is the number of Cloud TPU cores. The following table lists all the valid resource requests.

  If the resource intended to be used is a Cloud TPU Pod, please request quota (https://cloud.google.com/tpu/docs/quota#request_quota_for_tpu_pods) since the default quota for Cloud TPU Pod is zero.

Resource Request	Cloud TPU type	Required GKE version
cloud-tpus.google.com/v2: 8	A Cloud TPU v2 device (8 cores)	1.10.4-gke.2 or later
cloud-tpus.google.com/v2: 32	A v2-32 Cloud TPU Pod (32 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v2: 128	A v2-128 Cloud TPU Pod (128 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v2: 256	A v2-256 Cloud TPU Pod (256 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v2: 512	A v2-512 Cloud TPU Pod (512 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v3: 32	A v3-32 Cloud TPU Pod (32 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v3: 64	A v3-64 Cloud TPU Pod (64 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v3: 128	A v3-128 Cloud TPU Pod (128 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v3: 256	A v3-256 Cloud TPU Pod (256 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v3: 512	A v3-512 Cloud TPU Pod (512 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v3: 1024	A v3-1024 Cloud TPU Pod (1024 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/v3: 2048	A v3-2048 Cloud TPU Pod (2048 cores) *(beta)*	1.10.7-gke.6 or later
cloud-tpus.google.com/preemptible-v2: 8	A Preemptible Cloud TPU v2 device (8 cores)	1.10.6-gke.1 or later
cloud-tpus.google.com/v3: 8	A Cloud TPU v3 device (8 cores)	1.10.7-gke.6 or later
cloud-tpus.google.com/preemptible-v3: 8	A Preemptible Cloud TPU v3 device (8 cores)	1.10.7-gke.6 or later

For more information on specifying resources and limits in the Pod spec, see the Kubernetes documentation (https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/).

For example, this Job spec requests one Preemptible Cloud TPU v2 device with TensorFlow 1.15:

Follow these steps to create the Job in the GKE cluster, and to install kubectl (https://kubernetes.io/docs/tasks/tools/install-kubectl/):

1. Create your Job spec, `example-job.yaml`, including the Job spec described above.
2. Run the Job:

   This command creates the job that automatically schedules the Pod.
3. Verify that the Pod has been scheduled and Cloud TPU nodes have been provisioned. A Pod requesting Cloud TPU nodes can be pending for 5 minutes before running. You will see output similar to the following until the Pod is scheduled.

   After 5 minutes, you should see something like this:

   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.

Follow these steps to get the logs of the Cloud TPU instances used by your Kubernetes Pods.

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

   Go to the GKE page (https://console.cloud.google.com/kubernetes)
2. Click **Workloads**.

3. Find your Deployment/Job and click it.
4. Find the Pod under **Managed pods** and click it.
5. Find the Container under **Containers** and click it.
6. Click the **Stackdriver logs** to see the logs of the Cloud TPU used by this Container.

TensorBoard (https://www.tensorflow.org/guide/summaries_and_tensorboard) is a suite of tools designed to present TensorFlow data visually. TensorBoard can help identify bottlenecks in processing and suggest ways to improve performance.

TPU Profiler (/tpu/docs/cloud-tpu-tools#profile_tab) is a TensorBoard plugin for capturing a profile on an individual Cloud TPU or a Cloud TPU Pod which can be visualized on TensorBoard. The Cloud TPU tool selector will become available under the **Profile** tab on the TensorBoard menu bar only after you have collected trace information from a running TensorFlow model using TPU Profiler.

Follow these steps to run TensorBoard in the GKE cluster:

1. Download the TensorBoard Deployment YAML configuration.
2. Change the environment variable `MODEL_BUCKET` in the file to the Cloud Storage location where your model and the TensorFlow events exist.
3. Run TensorBoard in a GKE cluster.
4. Get the `EXTERNAL_IP` of the TensorBoard. Note that a Load Balancer will be created to route the requests to TensorBoard, which will incur additional cost (/compute/pricing#lb).

5. Access `http://EXTERNAL_IP:6006` within your browser.

Follow these steps to run TPU Profiler in the GKE cluster:

1. Download the TPU Profiler Job YAML configuration.

2. Change the environment variable `TPU_NAME` in the file to the name of the Cloud TPU you want to profile. The TPU name appears in the Cloud Console (https://console.cloud.google.com/) on the **Compute Engine** > **TPUs** page, in the following format:

   For example:

★ **Note:** The Cloud TPU is only visible from the Cloud Console while it is running. after the execution completes, it is deleted and won't appear in the Console.

3. Change the environment variable `MODEL_BUCKET` in the file to the Cloud Storage location where your model and the TensorFlow events exist.
4. Run TPU Profiler in a GKE cluster.

5. Refresh your browser to see the tracing data under the **Profile** tab on TensorBoard.

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

- Work through the tutorial (/tpu/docs/tutorials/kubernetes-engine-resnet) to train the TensorFlow ResNet-50 model on Cloud TPU and GKE.
- 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.