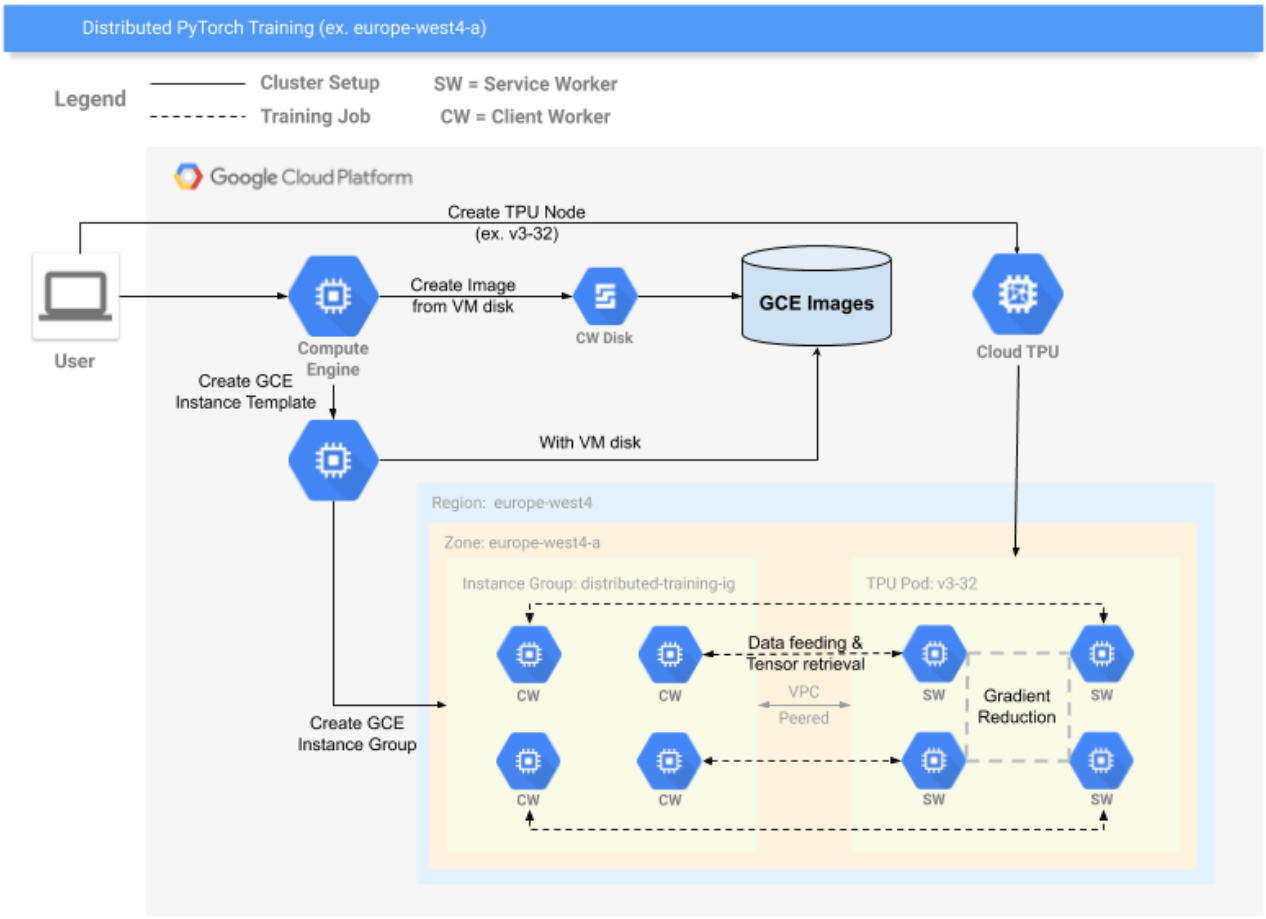
Alpha

This feature is in a pre-release state and might change or have limited support. For more information, see the product launch stages (/products/#product-launch-stages).

For information about access to this release, see the access request page (http://www.google.com).

This tutorial shows how to scale up training your model from a single Cloud TPU (v2-8 or v3-8) to a Cloud TPU Pod. Cloud TPUs accelerators in a TPU Pod are connected by a very high bandwidth interconnects making them great at scaling up training jobs. For more information about the Cloud TPU Pods offerings refer to the Cloud TPU product page (https://cloud.google.com/tpu/) or to this Cloud TPU presentation (https://storage.googleapis.com/nexttpu/index.html).

The following diagram provides an overview of the distributed cluster setup and shows how training happens. In the single device setup, a single VM (client worker = CW) feeds a single TPU worker (service worker = SW). Similarly, in distributed training a cluster of VMs/CWs and also a corresponding TPU Pod slice (cluster of SWs) and each of the CWs feeds a single SW. The input pipeline runs on the CW and all the model training happens on the SW.



- Set up a Compute Engine Instance Group and Cloud TPU Pod for training with PyTorch/XLA
- Run PyTorch/XLA training on a Cloud TPU Pod

Warning: This model uses a third-party dataset. Google provides no representation, warranty, or other guarantees about the validity, or any other aspects of this dataset.

Before starting distributed training on Cloud TPU Pods, verify that your model trains on a single v2-8 or v3-8 Cloud TPU device. If your model has significant performance problems on a single device, refer to the best practices (https://github.com/pytorch/xla/blob/master/API_GUIDE.md#performance-caveats) and troubleshooting (https://github.com/pytorch/xla/blob/master/TROUBLESHOOTING.md) guides.

Once your your single TPU device is successfully training, perform the following steps to set up and train on a Cloud TPU Pod:

1. [Optional] Capture a VM disk image into a VM image. (#capture-vm-disk)
2. Create an Instance template from a VM image. (#create-instance-template)
3. Create an Instance Group from your Instance template. (#create-instance-group)
4. SSH into your Compute Engine VM (#ssh)
5. Verify firewall rules to allow inter-VM communication. (#verify-firewall).
6. Create a Cloud TPU Pod. (#create-tpu-pod)
7. Run distributed training on the Pod. (#start-distributed-training)
8. Clean up. (#clean-up)

You can use the disk image from the VM you used to train the single TPU (that already has the dataset, packages installed, etc.) to create a VM Image (https://cloud.google.com/compute/docs/images/create-delete-deprecate-private-images#prepare_instance_for_image) that can be used for Pod training. You can skip this step if you do not require any additional packages or datasets for training and can just use a standard PyTorch/XLA image.

Create a default instance template (https://cloud.google.com/compute/docs/instance-templates/create-instance-templates#creating_a_new_instance_template)

. When you are creating an instance template, you can use the VM Image you created in the above step **OR** you can use the public PyTorch/XLA image Google provides by following these instructions:

1. On the create an instance template page, click the "Change" button under the "Boot Disk" Section.
2. If you captured a VM disk image, click on the "Custom images" tab and select the image you captured. If you did not capture a VM disk image, select the public PyTorch/XLA image from the "OS images" pull down menu.
3. Make sure that:
    a. Under Machine type, select `n1-standard-16` for this example that uses ResNet-50 training. For your own model choose whatever VM size you used to train on a v3-8/v2-8.
    b. Under "Identity and API access" → "Access Scopes", select "Allow full access to all Cloud APIs".
4. Click "Select" at the bottom to create your Instance template.

From the Google Cloud Console (https://console.cloud.google.com/), select **Compute Engine > instance groups** to access the **Create an instance group** form. When filling out the form, specify the following parameters for the configuration:

1. Under "Name", specify a name for your Instance Group.
2. Under "Location" section make sure to select "Single Zone".
3. Under "Region" and "Zone" sections make sure to select the same zone that the TPU Pod will be created in.
4. Under "Instance template" select the template you created in the previous step.
5. Under "Autoscaling mode" select "Off". If you see an "Autoscaling mode" dropdown, select "Don't configure autoscaling".
6. Under "Number of Instances": You will need N number instances where N equals the total number of cores you are using divided by 8, since each VM instance (CW) feeds 8 TPU

cores. In this example, N equals 4, so you need 4 Instances.

7. Under "Health check" select "No health check".
8. Click "Select" at the bottom to create your Instance Group.

After creating your Instance Group, SSH into your Compute Engine VM to continue these instructions.

1. From the Google Cloud Console (https://console.cloud.google.com/) select **Compute Engine > instance groups** and click on the Instance group you just created. This opens a window that displays all of the Instances in your Instance Group.
2. At the end of the description line for one of your Instances, click the SSH button. This brings up a VM session window. Run each command that begins with `(vm)$` in your VM session window.

Verify that your Compute VMs can communicate with each other on port `8477` by checking the firewall rules (https://cloud.google.com/filestore/docs/configuring-firewall) for your project **OR** by running the `nmap` command from your Compute Engine VM.

1. Run the `nmap` command. The variable ***instance-ID*** is one of the instances from your Instance Group. For example, instance-group-1-g0h2.

As long as the STATE field does not say **filtered** the firewall rules are set up correctly.

Go to the Google Cloud Console (https://console.cloud.google.com/) and select **Compute Engine > TPUs**. This brings up the page where you can create a TPU node and specify the following parameters:

1. Under "Name", specify a name for your TPU Pod.
2. Under "Zone" specify the zone (/tpu/docs/types-zones) to use for your Cloud TPU. Make sure it is in the same zone as your Instance Group.
3. Under "TPU type", select the Cloud TPU type. (/tpu/docs/types-zones).
4. Under "TPU software version" select the latest stable release (`pytorch-0.X`), for example `pytorch-0.5`.
5. Use the default network.
6. Set the IP address range (/tpu/docs/internal-ip-blocks). For example, 10.240.0.0.

Note: this example assumes you are using a conda environment for distributed training. Also, this example turns on XLA_USE_BF16=1 at training time, similarly you can use the **`--env`** variable to list any other environment variables you want to have distributed.

1. From your VM session window, export the Cloud TPU name and activate the conda environment.

2. Run the training script:

★ **Note:** The conda environment should correspond to the TPU software version. TPU software **`pytorch-0.5`** corresponds to conda environment **`torch-xla-0.5`** and TPU software **`pytorch-`**

nightly corresponds to conda environment **torch-xla-nightly**.

Once you run the above command you should see output similar to the following (note this is using `--fake_data`). The training takes about 1/2 hour on a v3-32 TPU Pod.

To avoid incurring charges to your Google Cloud Platform account for the resources used in this tutorial:

Exit from the Compute Engine VM and delete:

1. The Instance Group you created under **Compute Engine > Instance Groups**
2. The TPU Pod under **Compute Engine > TPUs**.

Try the PyTorch colabs:

- Training MNIST on TPUs (https://colab.sandbox.google.com/github/pytorch/xla/blob/master/contrib/colab/mnist-training-xrt-1-15.ipynb)
- Training ResNet18 on TPUs with Cifar10 dataset (https://colab.sandbox.google.com/github/pytorch/xla/blob/master/contrib/colab/resnet18-training-xrt-1-15.ipynb)
- Inference with Pretrained ResNet50 Model (https://colab.sandbox.google.com/github/pytorch/xla/blob/master/contrib/colab/resnet50-inference-xrt-1-15.ipynb)