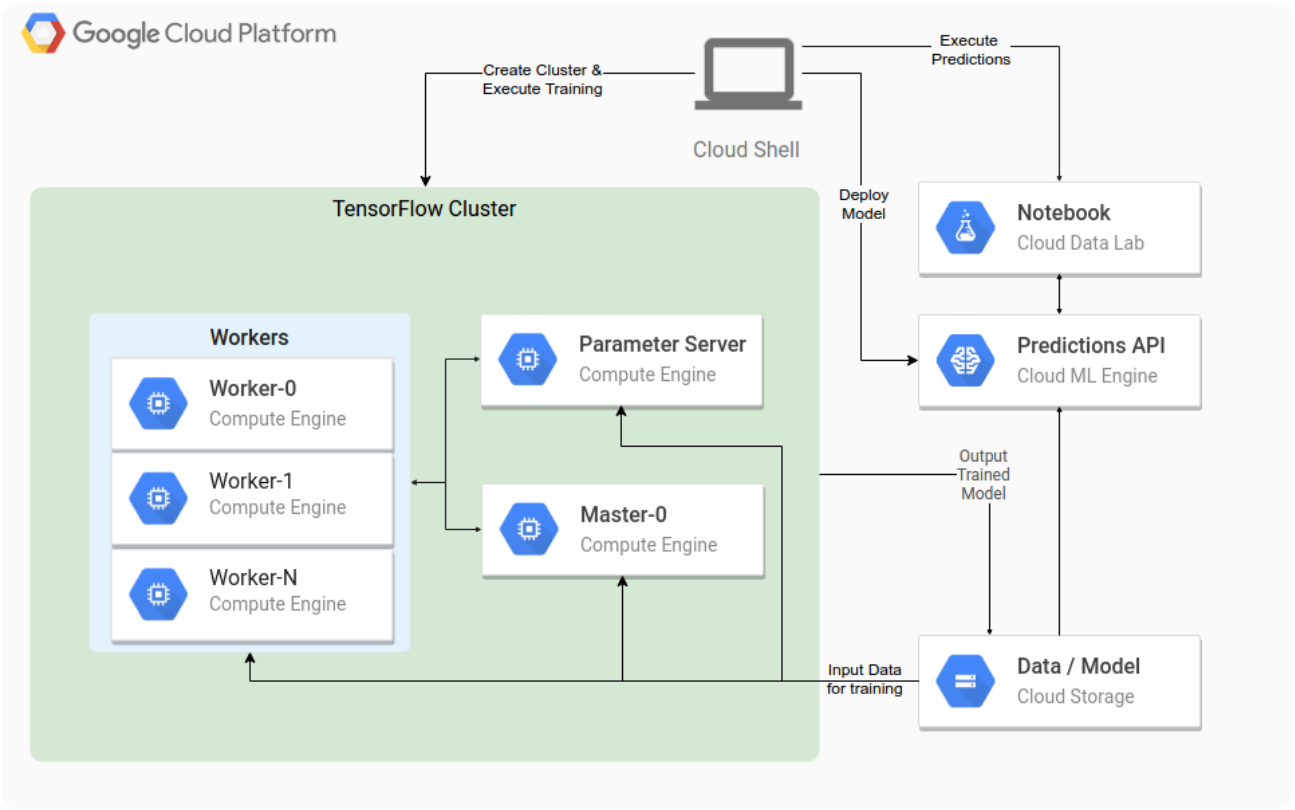
This tutorial shows how to use a distributed configuration of TensorFlow (https://www.tensorflow.org/) on multiple Compute Engine instances to train a convolutional neural network (https://wikipedia.org/wiki/Convolutional_neural_network) model using the MNIST dataset (http://yann.lecun.com/exdb/mnist/). The MNIST dataset enables handwritten digit recognition, and is widely used in machine learning as a training set for image recognition.

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state of the art in ML and developers build and deploy ML-powered applications. TensorFlow is designed to run on multiple computers to distribute training workloads. In this tutorial, you run TensorFlow on multiple Compute Engine virtual machine (VM) instances to train the model. You can use AI Platform instead, which manages resource allocation tasks for you and can host your trained models. We recommend that you use AI Platform unless you have a specific reason not to. You can learn more in the version of this tutorial that uses AI Platform and Datalab (/ml-engine/docs/distributed-tensorflow-mnist-cloud-datalab).

The following diagram describes the architecture for running a distributed configuration of TensorFlow on Compute Engine, and using AI Platform with Datalab to execute predictions with your trained model.



This tutorial shows you how to set up and use this architecture, and explains some of the concepts along the way.

- Set up Compute Engine to create a cluster of VMs to run TensorFlow.
- Learn how to run the distributed TensorFlow sample code on your Compute Engine cluster to train a model. The example code uses the latest TensorFlow libraries and patterns, so you can use it as a reference when designing your own training code.

- Deploy the trained model to AI Platform to create a custom API for predictions and then execute predictions using a Datalab notebook.

The estimated price to run this tutorial, assuming you use every resource for an entire day, is approximately $1.20 based on this pricing calculator (/products/calculator/#id=50e2750e-4c10-4d07-962a-6ce19c2bb0d6).

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).

4. Enable the Compute Engine API, AI Platform Training and Prediction API, and Cloud Source Repositories APIs.

   Enable the APIs (https://console.cloud.google.com/flows/enableapi?apiid=ml.googleapis.com,sourcerepo.googleapis.com,compute_component)

This tutorial uses Cloud Shell, a fully functioning Linux shell in the Google Cloud Console.

1. Go to Cloud Shell.

   Open Cloud Shell (https://console.cloud.google.com/cloudshell/?_ga=1.258828298.1884796882.1495563798)

2. Set your default Compute Engine zone and your default project. Replace `[YOUR_PROJECT_ID]` with your Google Cloud project.

3. Clone the GitHub repository:

4. Create the initial VM instance from an Ubuntu Wily image:

5. Use `ssh` to connect to the VM:

6. Install `pip`:

★ **Note:** Installing `pip` might interfere with other Python-based installations on your local system. Because this VM is purpose-built for TensorFlow, the simple installation option shouldn't cause problems. If you're using TensorFlow in a critical production scenario, we recommend using virtualenv to create a virtual environment for isolation. Follow these instructions to install TensorFlow with virtualenv (https://www.tensorflow.org/install/install_linux#installing_with_virtualenv).

7. Install TensorFlow:

★ **Note:** The TensorFlow installation uses the CPU-only version 1.2.1 for Python 2.7. To install a different distribution, follow the instructions in Installing TensorFlow on Ubuntu (https://www.tensorflow.org/install/install_linux).

8. Type `exit` to return to Cloud Shell.

Next, create a Cloud Storage bucket to store your MNIST files. Follow these steps:

1. Create a regional Cloud Storage bucket to hold the MNIST data files that are to be shared among the worker instances:

2. Install TensorFlow:

3. Use the following script to download the MNIST data files and copy them to the bucket:

To create the worker, master, and parameter server instances, convert the template instance into an image, and then use the image to create each new instance.

1. Turn off auto-delete for the `template-instance` VM, which preserves the disk when the VM is deleted:

2. Delete `template-instance`:

3. Create the image `template-image` from the `template-instance` disk:

4. Create the additional instances. For this tutorial, create four instances named `master-0`, `worker-0`,`worker-1`,and`ps-0`. The `storage-rw` scope allows the instances to access your Cloud Storage bucket. Be sure to delimit the instance names with spaces, as follows:

Note: In this example, the machine type is set to `n1-standard-4`. However, because parameter and master servers typically require fewer cores than workers, you could create master servers with a different machine type. To do this, you would run the command twice: once for the workers and once for the parameter and master servers, but specifying fewer cores than for the workers.

The cluster is ready to run distributed TensorFlow.

In this section, you run a script to instruct all of your VM instances to run TensorFlow code to train the model.

1. In Cloud Shell, run the following command from the `cloudml-dist-mnist-example` directory:

   The script named `start-training.sh` pushes the code to each VM and sends the necessary parameters to start the TensorFlow process on each machine to create the distributed cluster. The output stream in Cloud Shell shows the loss and the accuracy values for the test dataset.

   ```
   INFO:tensorflow:Evaluation [99/100]
   INFO:tensorflow:Evaluation [100/100]
   INFO:tensorflow:Finished evaluation at 2017-04-19-11:58:27
   INFO:tensorflow:Saving dict for global step 2627: accuracy = 0.9893, global_step = 2627, loss = 0.0335496
   ```

   When the training is done, the script prints out the location of the newly generated model files:

2. Copy the location of your bucket path for use in later steps.

You've successfully generated a new model that you can use to make predictions. Training more-sophisticated models requires more-complex TensorFlow code, but the configuration of the compute and storage resources are similar.

Training the model is only half of the story. You need to plug your model into your application, or wrap an API service around it with authentication, and eventually make it all scale. There is a relatively significant amount of engineering work left to make your model useful.

AI Platform can help with some of this work. AI Platform provides a fully managed version of TensorFlow running on Google Cloud. AI Platform gives you all of powerful features of TensorFlow without needing to set up any additional infrastructure or install any software. You can automatically scale your distributed training to use as many CPUs or GPUs as you need, and you pay for only what you use.

Because AI Platform runs TensorFlow behind the scenes, all of your work is portable, and you aren't locked into a proprietary tool.

Try the tutorial Using distributed TensorFlow with Datalab (/ml-engine/docs/distributed-tensorflow-mnist-cloud-datalab) to use the same example code to train your model by using AI Platform.

You can also deploy your model to AI Platform for predictions. Use the following steps to deploy your model to AI Platform. Deploying your model helps give you the ability to quickly test and apply your model at scale, with all the security and reliability features you would expect from a Google-managed service.

The following steps use the model bucket path that was output earlier by the script named `start-training.sh`.

1. Recall the output path to your Cloud Storage bucket with generated models. It is in the following format, where `[JOB_ID]` is the job ID. You use this path in the next step:

2. Define a new `v1` version of your model by using the `gcloud` command-line tool, and point it to the model files in your bucket. The following command can take several minutes to complete. Replace `[YOUR_BUCKET_PATH]` with the output path from the previous step. The path starts with `gs://`.

3. Set the default version of your model to `v1`:

The model is now running with AI Platform and able to process predictions. In the next section, you use Datalab to make and visualize predictions.

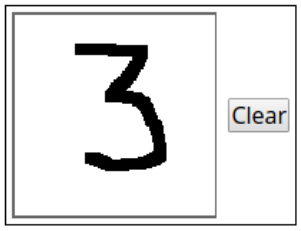
To test your predictions, create a Datalab instance that uses interactive Jupyter Notebooks to execute code.

1. In Cloud Shell, enter the following command to create a Datalab instance:

2. From Cloud Shell, launch Datalab notebook listing page by clicking **Cloud Shell Web preview** (the square icon in the top right).
3. Select **Change port** and select **Port 8081** to launch a new tab in your browser.
4. In the Datalab application, create a new notebook by clicking **+Notebook** in the upper right.
5. Paste the following text into the first cell of the new notebook:

6. Click **Run** at the top of the page to download the **Online prediction example.ipynb** notebook. The script copies the remote notebook's contents into the current notebook.
7. Reload the browser page to load the new notebook content. Then select the first cell containing the JavaScript code and click **Run** to execute it.
8. Scroll down the page until you see the number drawing panel, and draw a number with your cursor:

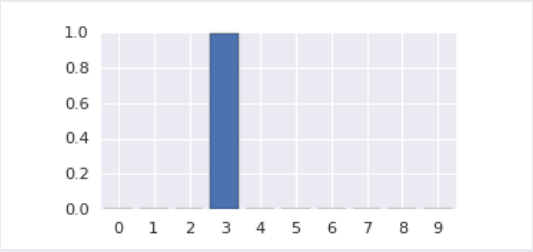
```
</script>
'''

from IPython.display import HTML
HTML(input_form + javascript)
```



9. Click in the next cell to activate it and then click on the down arrow next to the **Run** button at the top and select **Run from this Cell**.

   The output of the prediction is a length-10 array in which each index, 0-9, contains a number corresponding to that digit. The closer the number is to 1, the more likely that index matches the digit you entered. You can see that the number 3 slot highlighted in the list is very close to 1, and therefore has a high probability of matching the digit.

The last cell in the notebook displays a bar chart that clearly shows that it predicted your number, which was 3 in this case.



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

1. Delete the version of your model:

2. Delete the model:

3. Delete your Cloud Storage bucket:

4. Delete your VMs including Datalab:

5. Delete your VM template image:

6. Delete your Datalab persistent disk:

- To try the same code using AI Platform, read the tutorial Using distributed TensorFlow with Datalab (/ml-engine/docs/distributed-tensorflow-mnist-cloud-datalab).
- Try out other Google Cloud features for yourself. Have a look at our tutorials (/docs/tutorials).