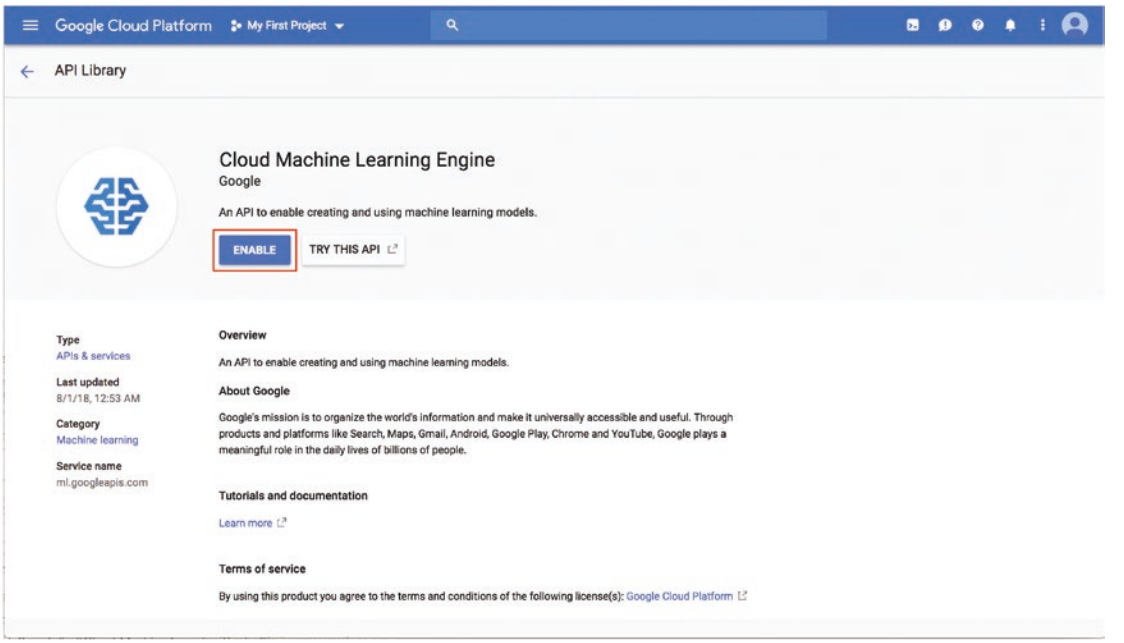

Figure 41-2. *Enable Cloud Machine Learning APIs*

Packaging the Code for Training on Cloud MLE

The code for training on Cloud MLE must be prepared as a python package. The recommended project structure is explained as follows:

IrisCloudML: [project name as parent folder]

- Trainer: [folder containing the model and execution code]
  - __init__.py: [an empty special python file indicating that the containing folder is a Python package]
  - model.py: [script contains the logic of the model written in TensorFlow, Keras, etc.]
  - task.py: [script contains the application that orchestrates or manages the training job]
- scripts: [folder containing scripts to execute jobs on Cloud MLE]
  - distributed-training.sh: [script to run a distributed training job on Cloud MLE]

    - hyper-tune.sh: [script to run a training job with hyper-parameter tuning on Cloud MLE]
    - single-instance-training.sh: [script to run a single instance training job on Cloud MLE]
    - online-prediction.sh: [script to execute an online prediction job on Cloud MLE]
    - create-prediction-service.sh: [script to create a prediction service on Cloud MLE]
- hptuning_config: [configuration file for hyper-parameter tuning on Cloud MLE]
- gpu_hptuning_config.yaml: [configuration file for hyper-parameter tuning with GPU training on Cloud MLE]

NOTE: FOLLOW THESE INSTRUCTIONS TO RUN THE EXAMPLES FOR TRAINING ON CLOUD MACHINE LEARNING ENGINE

1. Launch a Notebook Instance on GCP AI Platform.
2. Pull the code repository.
3. Navigate to the book folder. Run the scripts in the sub-folder `tensorflow'.
4. Should you choose to work with Google Colab, authenticate the user by running the code

```
from google.colab import auth
    auth.authenticate_user()
```

The TensorFlow Model

Now let's briefly examine the TF model code in the file '**model.py**'.

```
import six
```