Run the following code to create the prediction service.

```
source ./scripts/create-prediction-service.sh
```

```
Creating model...
Created ml engine model [projects/quantum-ally-219323/models/iris].
Creating model version...
Creating version (this might take a few minutes)......done.
```

The version details of the created model is as seen in Figure 41-5.

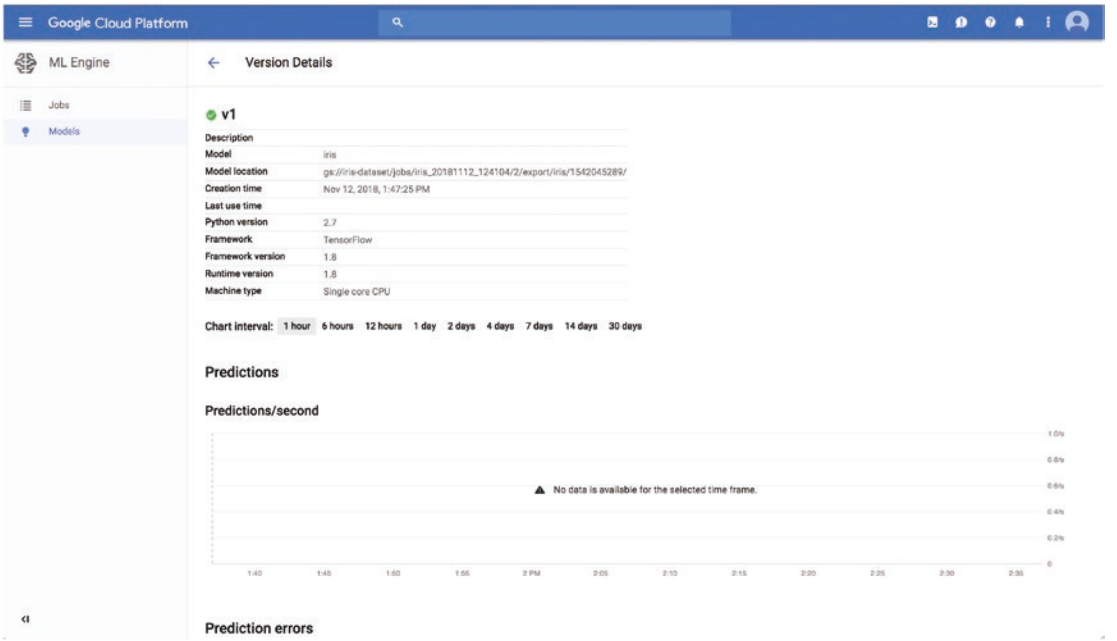

Figure 41-5. *Created model for serving on Cloud MLE*

Run Batch Prediction

Now let's run a batch prediction job on Cloud MLE. The code to execute a batch prediction call on Cloud MLE is provided in the following and stored in 'run-batch-predictions.sh.'

```
export JOB_NAME=iris_prediction
export MODEL_NAME=iris
```

```
export MODEL_VERSION=v1
export TEST_FILE=gs://iris-dataset/hold_out_test.csv

# submit a batched job
gcloud ai-platform jobs submit prediction $JOB_NAME \
        --model $MODEL_NAME \
        --version $MODEL_VERSION \
        --data-format TEXT \
        --region $REGION \
        --input-paths $TEST_FILE \
        --output-path $GCS_JOB_DIR/predictions

# stream job logs
echo "Job logs..."
gcloud ai-platform jobs stream-logs $JOB_NAME

# read output summary
echo "Job output summary:"
gsutil cat $GCS_JOB_DIR/predictions/prediction.results-00000-of-00001
```

Execute the code with the command

```
source ./scripts/run-batch-prediction.sh

Job [iris_prediction] submitted successfully.

jobId: iris_prediction
state: QUEUED
Job logs...
INFO    2018-11-12 14:48:18 -0500   service     Validating job
                                                requirements...
INFO    2018-11-12 14:48:18 -0500   service     Job creation request
                                                has been successfully
                                                validated.
INFO    2018-11-12 14:48:19 -0500   service     Job iris_prediction is
                                                queued.
Job output summary:
Job output summary:
```

```
{"classes": ["0", "1", "2"], "scores": [8.242315743700601e-06,
0.9921771883964539, 0.007814492098987103]}
{"classes": ["0", "1", "2"], "scores": [2.7296309657032225e-09,
0.015436310321092606, 0.9845637083053589]}
{"classes": ["0", "1", "2"], "scores": [5.207379217608832e-06,
0.9999237060546875, 7.100913353497162e-05]}
........          ........          ........          ........
{"classes": ["0", "1", "2"], "scores": [0.999919056892395,
8.089694165391847e-05, 9.295699552171275e-16]}
{"classes": ["0", "1", "2"], "scores": [0.9999765157699585,
2.3535780201200396e-05, 1.2826575252518792e-17]}
{"classes": ["0", "1", "2"], "scores": [1.8082465658153524e-06,
0.7016969919204712, 0.29830116033554077]}
```


The prediction job details on Cloud MLE is as shown in Figure 41-6.

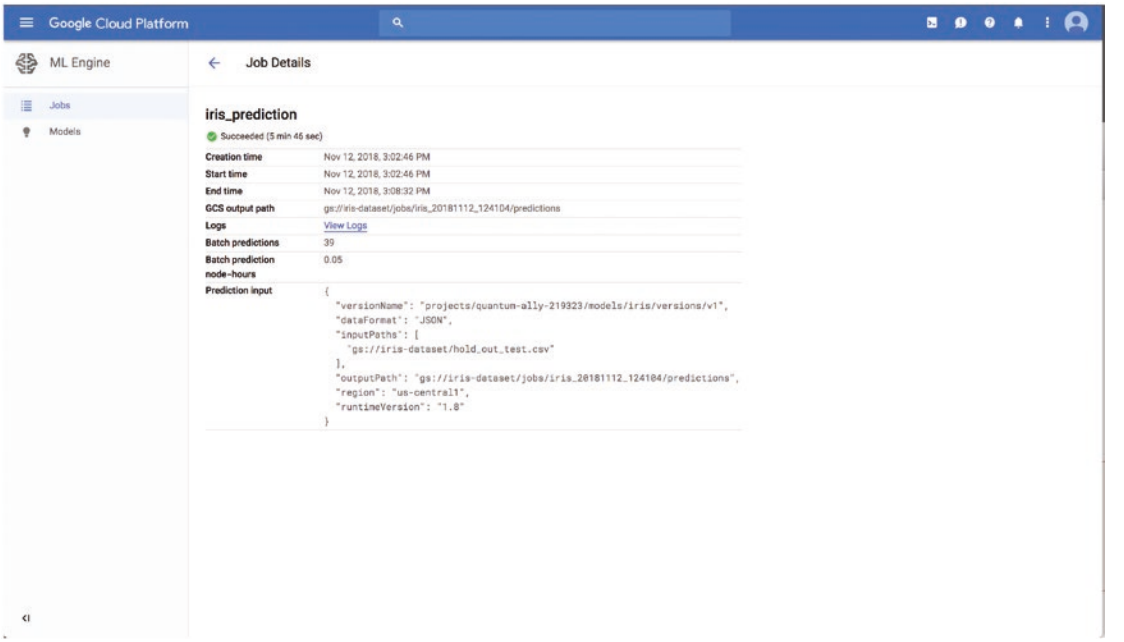

Figure 41-6. *Batch prediction job details*

Training with GPUs on Cloud MLE

Training models on GPUs can greatly reduce the processing time. In order to use GPUs on Cloud MLE, we make the following changes to our code example:

1. Change the scale tier to **'CUSTOM'.** The CUSTOM tier makes a number of GPU accelerators available, namely:
   a. standard_gpu: A single NVIDIA Tesla K80 GPU
   b. complex_model_m_gpu: Four NVIDIA Tesla K80 GPUs
   c. complex_model_l_gpu: Eight NVIDIA Tesla K80 GPUs
   d. standard_p100: A single NVIDIA Tesla P100 GPU
   e. complex_model_m_p100: Four NVIDIA Tesla P100 GPUs
   f. standard_v100: A single NVIDIA Tesla V100 GPU
   g. large_model_v100: A single NVIDIA Tesla V100 GPU
   h. complex_model_m_v100: Four NVIDIA Tesla V100 GPUs
   i. complex_model_l_v100: Eight NVIDIA Tesla V100 GPUs
2. Add the following parameters to the '.yaml' file to configure the GPU instance.

```
trainingInput:
  scaleTier: CUSTOM
  masterType: complex_model_m_gpu
  workerType: complex_model_m_gpu
  parameterServerType: large_model
  workerCount: 2
  parameterServerCount: 3
```

3. The full configuration file in 'gpu_hptuning_config.yaml' now looks like this:

```
trainingInput:
  scaleTier: CUSTOM
  masterType: complex_model_m_gpu
  workerType: complex_model_m_gpu
```