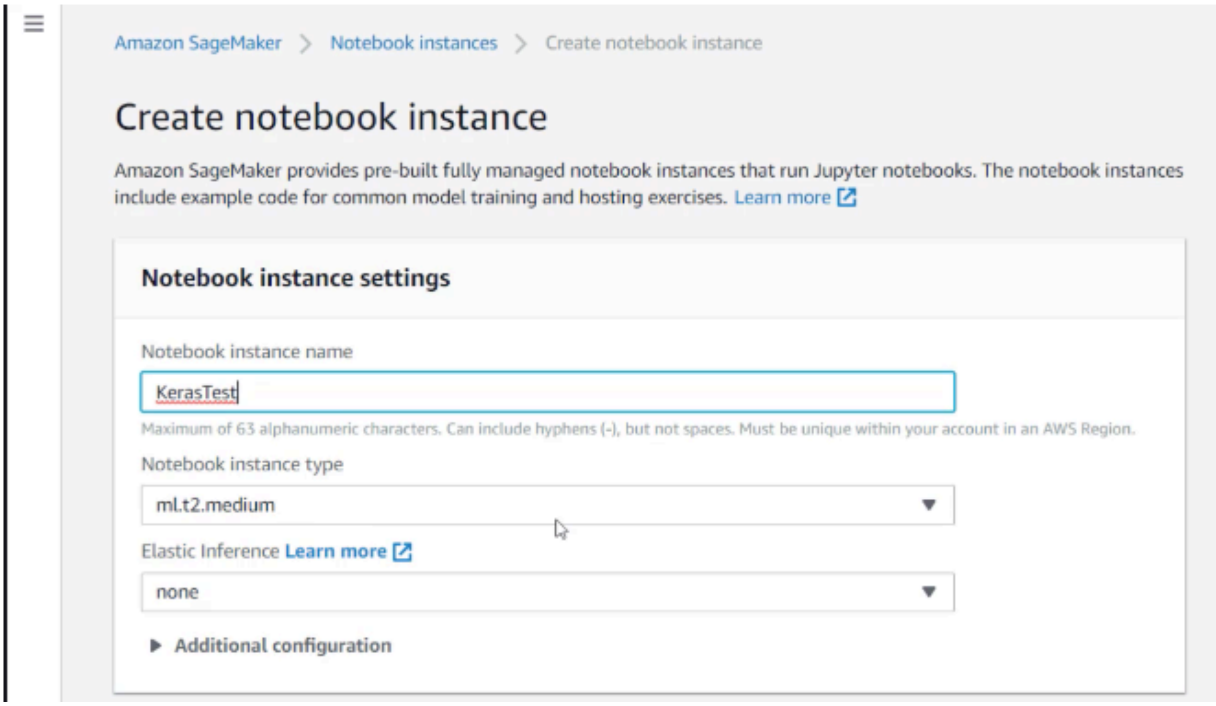
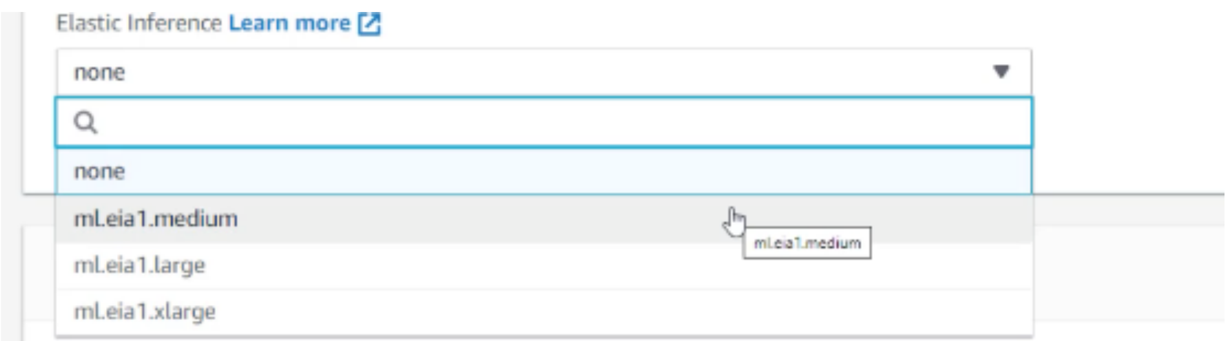
# Modeling - 4 - Implementation & Operation Lab

Use SM to run our own custom model
We will use the CNN Tensorflow model on MNIST dataset from previous exercise and adapt it to run in SageMaker

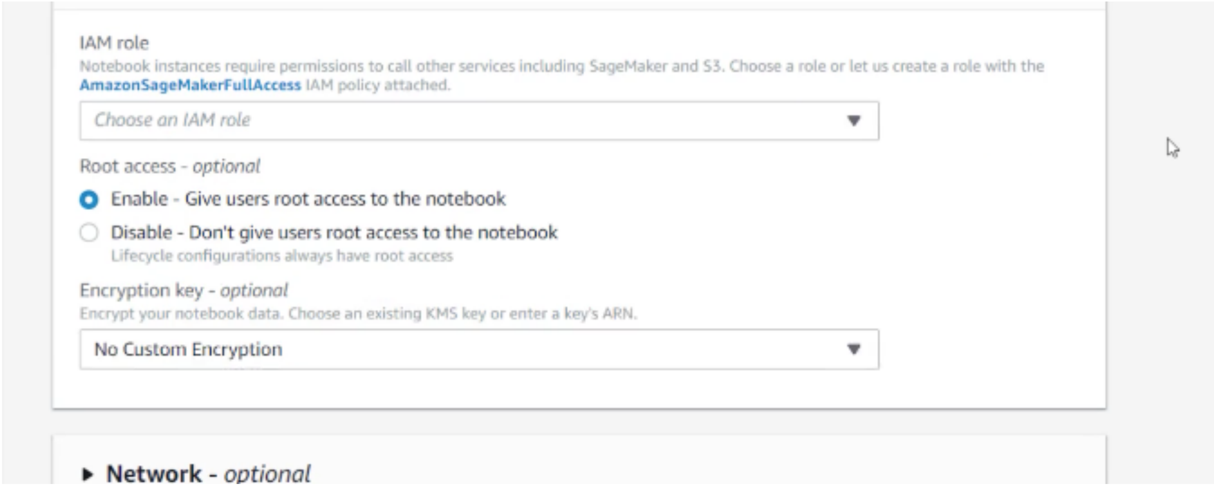
1/ Create a SM Notebook instance



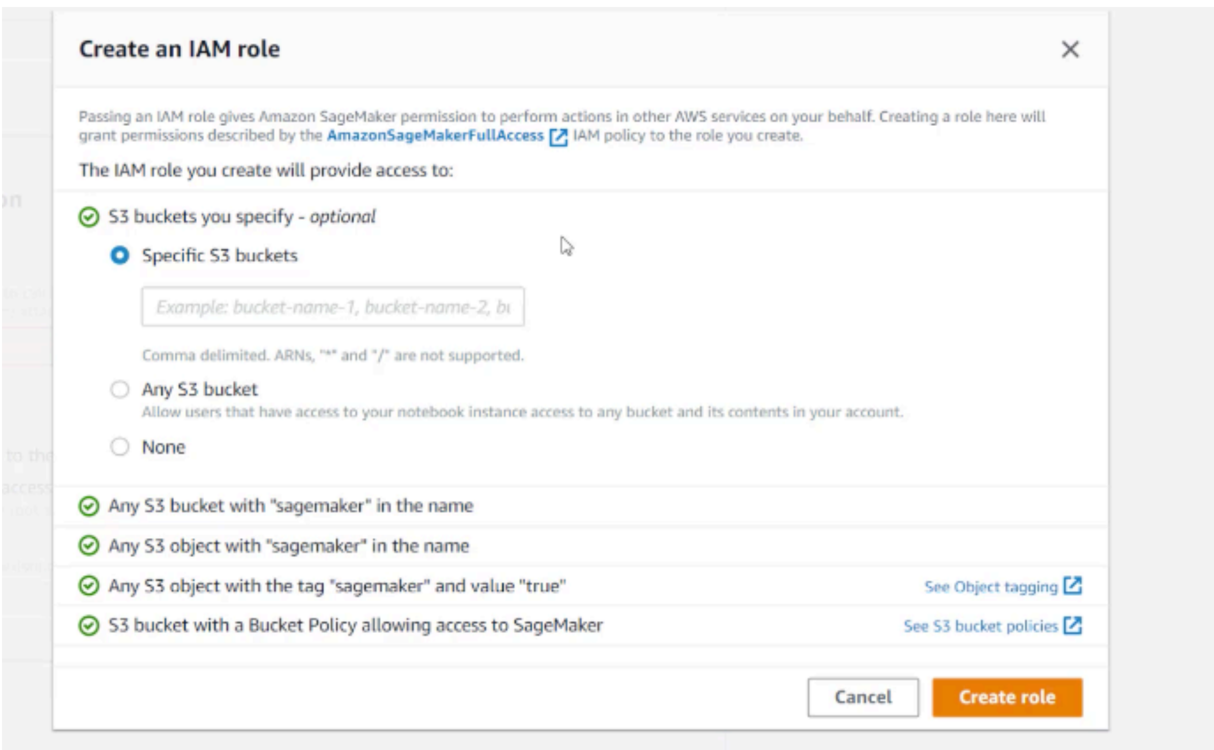

To note we can add Elastic Inferences to accelerate the Notebook



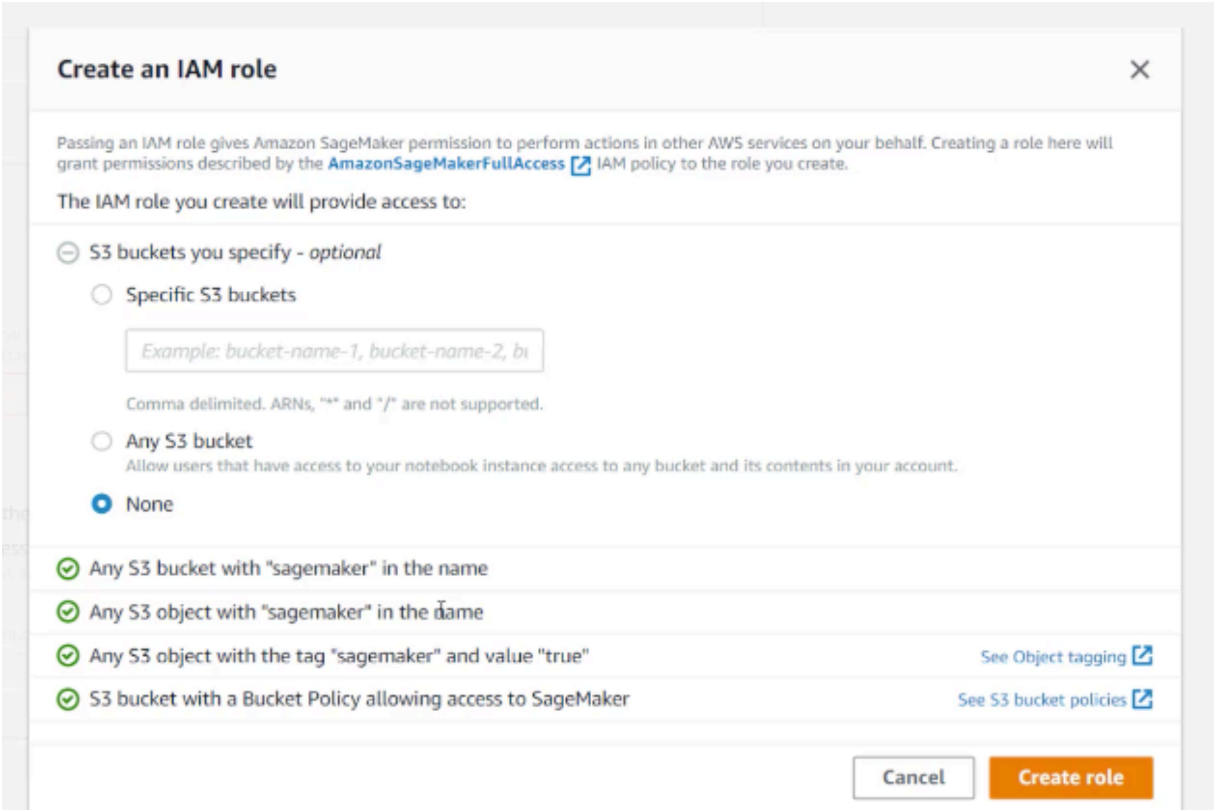

Security

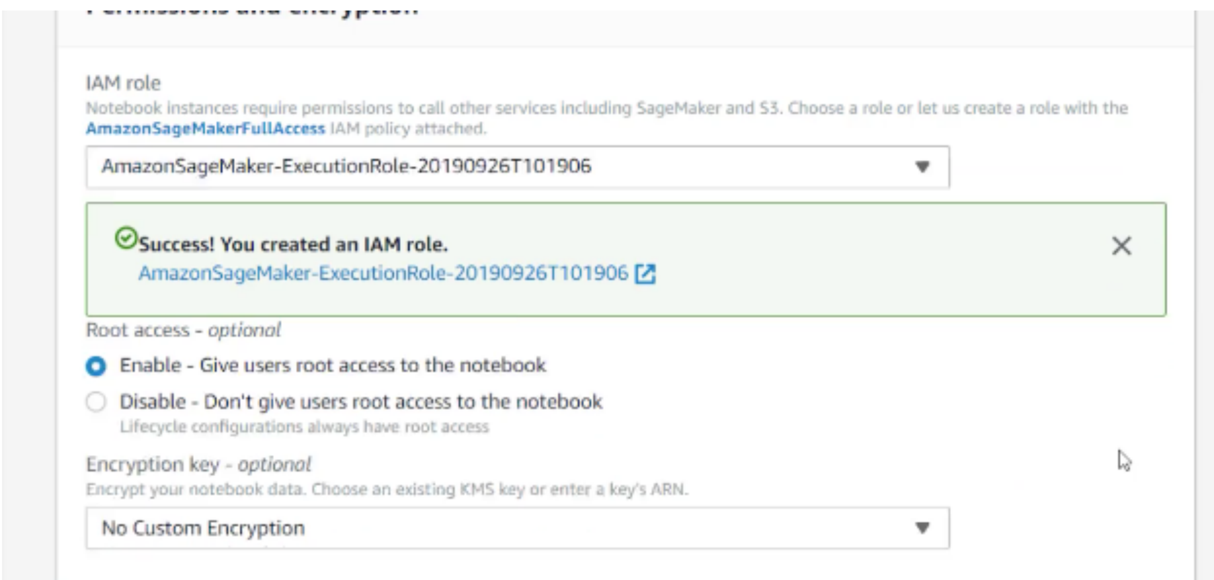



Allow IAM to create a new role
We can specify specific S3 buckets



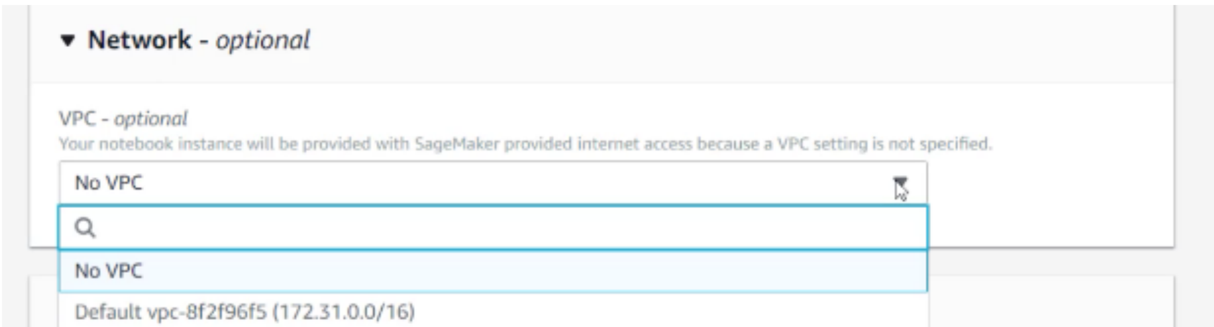

But for our example, we will select None and can still access it if we use "sagemaker" in the name



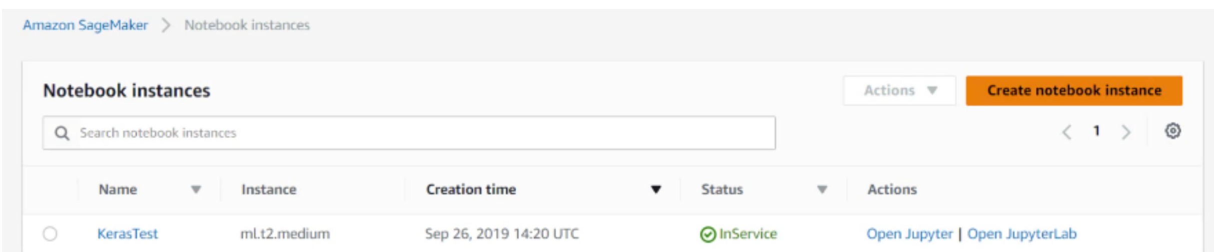




We can optionally encrypt the notebook data

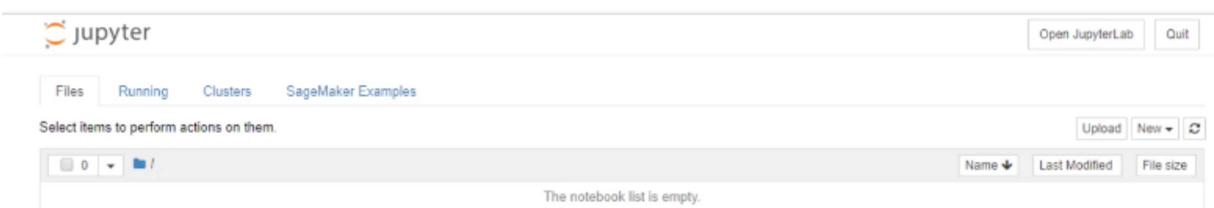

We can optionally set up a private VPC (we won't do it here)



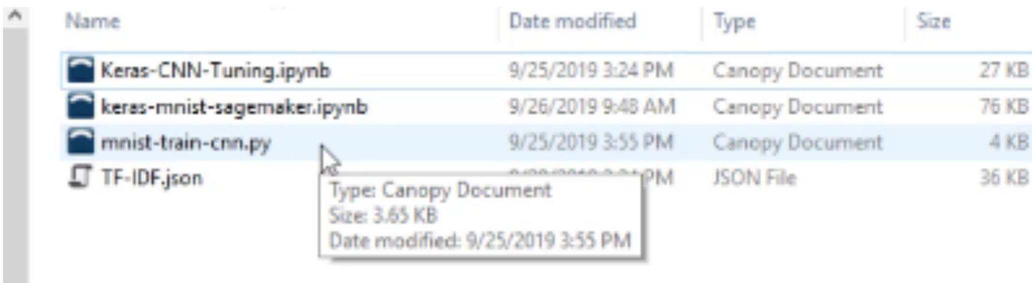

We now have our notebook instance ready

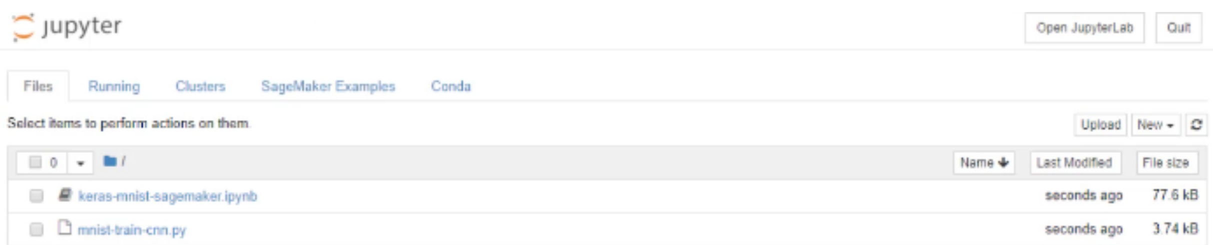


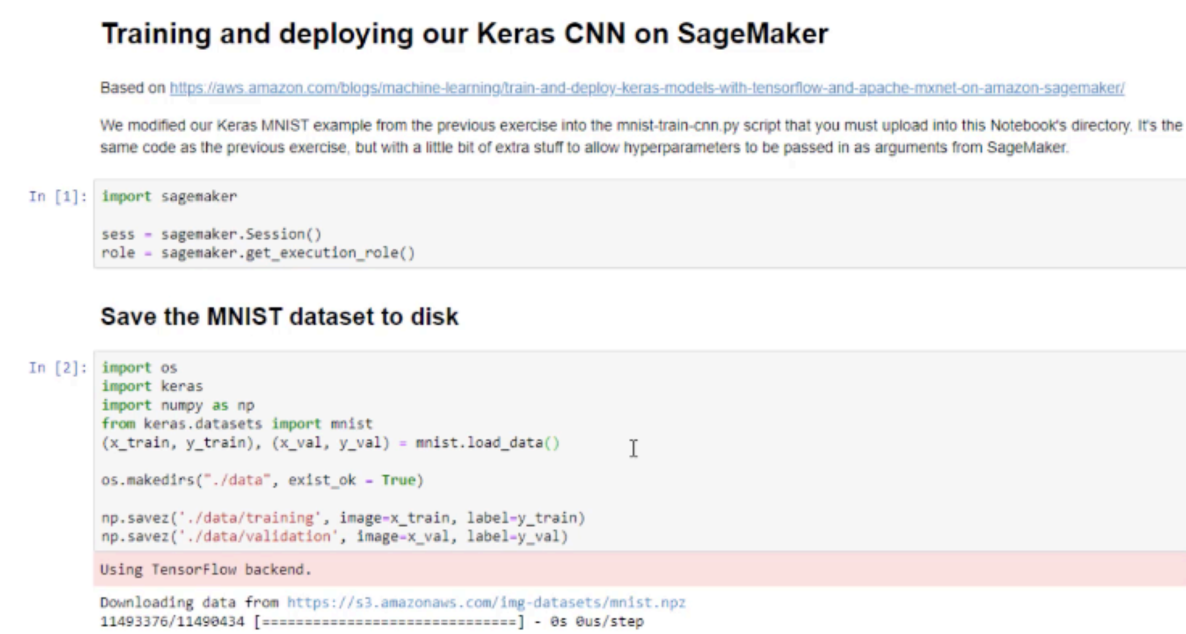

Open Jupiter notebook




Upload the notebook and the python file





To note, we allowed our notebook to access s3 if the bucket includes “sagemaker” name which is the case here

"Local" Run as a Test (on the sagemaker instance)

### Test out our CNN training script locally on the notebook instance

We'll test out running a single epoch, just to make sure the script works before we start spending money on P3 instances to train it further.

```
In [4]: !pygmentize mnist-train-cnn.py
```

```
In [5]: from sagemaker.tensorflow import TensorFlow

        tf_estimator = TensorFlow(entry_point='mnist-train-cnn.py',
                                  role=role,
                                  train_instance_count=1,
                                  train_instance_type='local',
                                  framework_version='1.12',
                                  py_version='py3',
                                  script_mode=True,
                                  hyperparameters={'epochs': 1}
                                 )
```

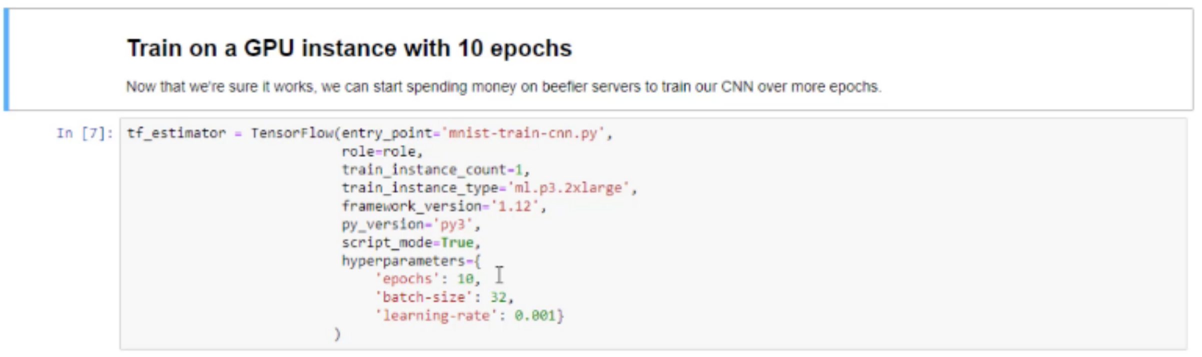
To start the training, call fit()

```
In [6]: tf_estimator.fit({'training': training_input_path, 'validation': validation_input_path})

Creating tmpqust9ati_algo-1-qr56z_1 ...
Attaching to tmpqust9ati_algo-1-qr56z_12mdone
algo-1-qr56z_1  | 2019-09-26 14:34:08,225 sagemaker-containers INFO     Imported framework sagemaker_tensorflow_container.tra
ining
algo-1-qr56z_1  | 2019-09-26 14:34:08,231 sagemaker-containers INFO     No GPUs detected (normal if no gpus installed)
algo-1-qr56z_1  | 2019-09-26 14:34:08,453 sagemaker-containers INFO     No GPUs detected (normal if no gpus installed)
algo-1-qr56z_1  | 2019-09-26 14:34:08,472 sagemaker-containers INFO     No GPUs detected (normal if no gpus installed)
algo-1-qr56z_1  | 2019-09-26 14:34:08,486 sagemaker-containers INFO     Invoking user script
algo-1-qr56z_1  |
algo-1-qr56z_1  | Training Env:
algo-1-qr56z_1  |
algo-1-qr56z_1  | {
algo-1-qr56z_1  |     "additional_framework_parameters": {},
algo-1-qr56z_1  |     "channel_input_dirs": {
algo-1-qr56z_1  |         "training": "/opt/ml/input/data/training",
```

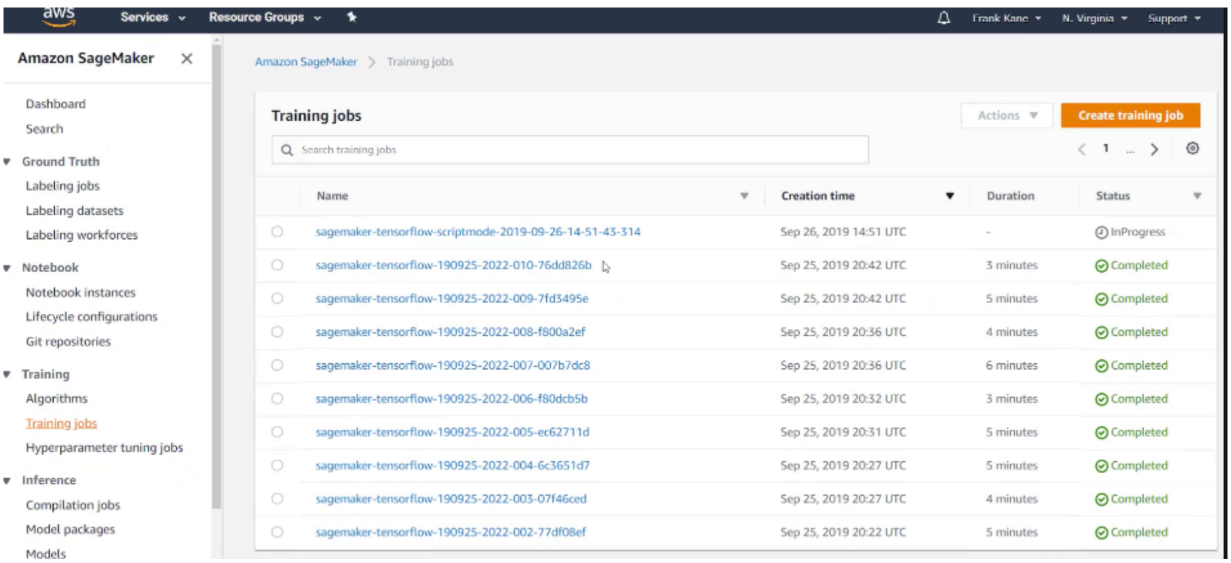
```
algo-1-qr56z_1  | Train on 60000 samples, validate on 10000 samples
algo-1-qr56z_1  | Epoch 1/1
algo-1-qr56z_1  |  - 92s - loss: 0.1974 - acc: 0.9417 - val_loss: 0.0519 - val_acc: 0.9840
algo-1-qr56z_1  | Validation loss    : 0.051937680732656734
algo-1-qr56z_1  | Validation accuracy: 0.984
algo-1-qr56z_1  | WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/saved_model/simple_save.p
y:85: calling SavedModelBuilder.add_meta_graph_and_variables (from tensorflow.python.saved_model.builder_impl) with legacy_in
it_op is deprecated and will be removed in a future version.
algo-1-qr56z_1  | Instructions for updating:
algo-1-qr56z_1  | Pass your op to the equivalent parameter main_op instead.
algo-1-qr56z_1  | 2019-09-26 14:35:46,794 sagemaker-containers INFO     Reporting training SUCCESS
tmpqust9ati_algo-1-qr56z_1 exited with code 0
Aborting on container exit...
```

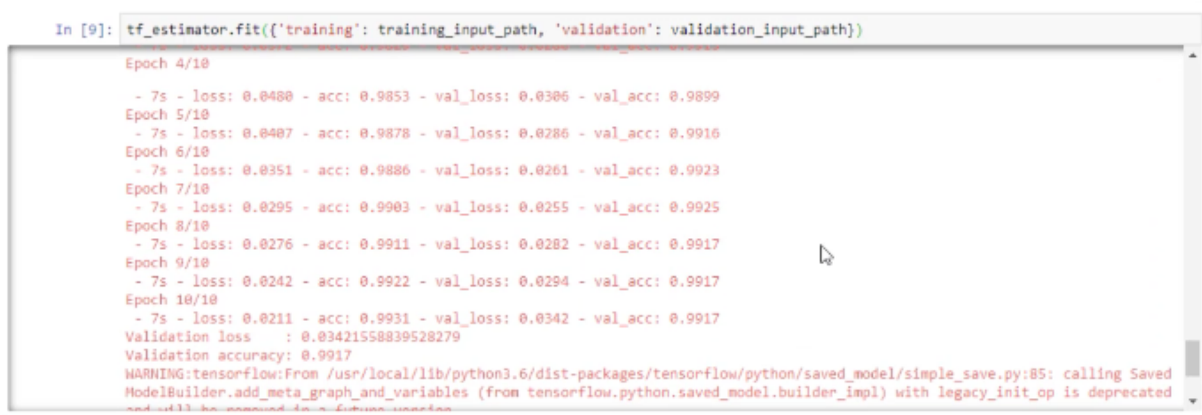
Validation accuracy of 98.4% with just 1 epoch - not bad

Now train this on a dedicated GPU instance and over 10 epochs
=> change instance_type and hyper parameter epoch




In the console, we can see the training jobs




Got to an accuracy of 99+=%

We can now deploy the model

We will use a CPU instance this time. And we will be adding an EI to help accelerate the inference (Elastic Inference instance)

### Deploy the model

To keep costs low, we'll deploy our model to a C5 instance to make inferences. You could also use GPU instances which would be faster, but at higher cost. Note we are also using an Elastic Inference accelerator here.

In [10]:
```
import time

tf_endpoint_name = 'keras-tf-mnist-'+time.strftime("%Y-%m-%d-%H-%M-%S", time.gmtime())

tf_predictor = tf_estimator.deploy(initial_instance_count=1,
                         instance_type='ml.c5.large',
                         accelerator_type='ml.eia1.medium',
                         endpoint_name=tf_endpoint_name)
```

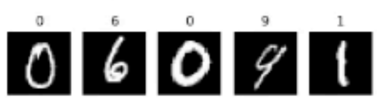
```
---------------------------------------------------------------------------------!
```

And now make predictions, selecting 5 random image and classifying them with tf_predictor.predict()

### Make predictions with the deployed model

In [10]:
```
%matplotlib inline
import random
import matplotlib.pyplot as plt

num_samples = 5
indices = random.sample(range(x_val.shape[0] - 1), num_samples)
images = x_val[indices]/255
labels = y_val[indices]

for i in range(num_samples):
    plt.subplot(1,num_samples,i+1)
    plt.imshow(images[i].reshape(28, 28), cmap='gray')
    plt.title(labels[i])
    plt.axis('off')

prediction = tf_predictor.predict(images.reshape(num_samples, 28, 28, 1))['predictions']
prediction = np.array(prediction)
predicted_label = prediction.argmax(axis=1)
print('Predicted labels are: {}'.format(predicted_label))
```

```
Predicted labels are: [0 6 0 4 1]
```




Now doing Hyper parameter Tuning with tuner.fit()

### Find the best hyperparameters with Automatic Model Tuning

This is by far the most expensive part of this exercise; we're going to use a fair amount of time on P3 instances here. If you're worried about your AWS costs, skip the rest of this notebook and just shut down your SageMaker notebook instance now.

```
In [12]: tf_estimator = TensorFlow(entry_point='mnist-train-cnn.py',
                                   role=role,
                                   train_instance_count=1,
                                   train_instance_type='ml.p3.2xlarge',
                                   framework_version='1.12',
                                   py_version='py3',
                                   script_mode=True
                                  )
```

Test range of epochs from 5 to 20,, learning rate from 0.001 and 0.1 using a logarithmic scale...

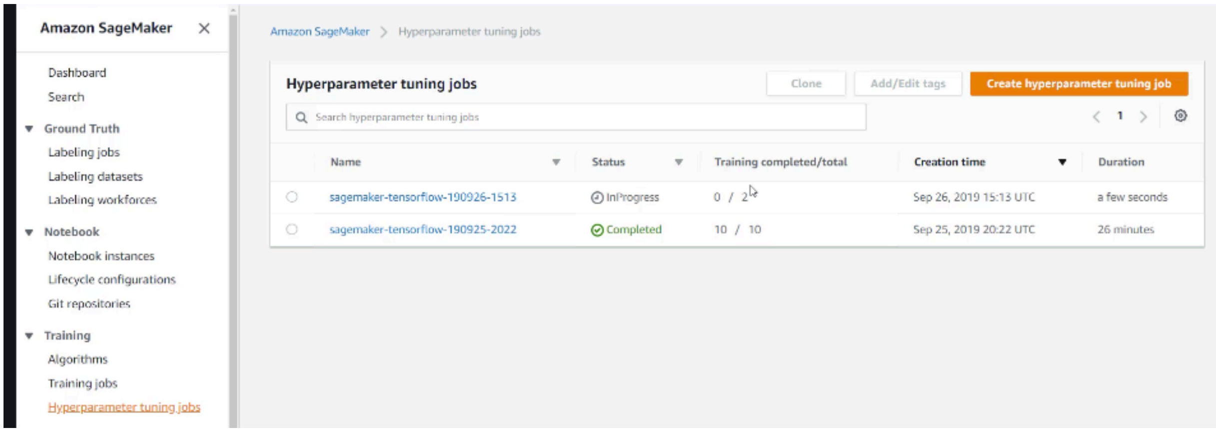
```
In [13]: from sagemaker.tuner import IntegerParameter, CategoricalParameter, ContinuousParameter, HyperparameterTuner

hyperparameter_ranges = {
    'epochs':        IntegerParameter(5, 20),
    'learning-rate': ContinuousParameter(0.0001, 0.1, scaling_type='Logarithmic'),
    'batch-size':    IntegerParameter(32, 1024),
}

objective_metric_name = 'val_acc'
objective_type = 'Maximize'
metric_definitions = [{'Name': 'val_acc', 'Regex': 'val_acc: ([0-9\\.]+)'}]

tuner = HyperparameterTuner(tf_estimator,
                            objective_metric_name,
                            hyperparameter_ranges,
                            metric_definitions,
                            max_jobs=10,
                            max_parallel_jobs=2,
                            objective_type=objective_type)
```

We try to optimize based on val_acc (accuracy on validation set), trying to Maximize it

max_parallel_job = 2
Tuning with 2 jobs at a time => because we learn from previous experiments and don't want too much parallelism

Run the tuner

```
In [16]: tuner.fit({'training': training_input_path, 'validation': validation_input_path})
```




Now deploy the best model with tuner.deploy()

## Deploy the best model

```
In [15]: import time

tf_endpoint_name = 'keras-tf-mnist-'+time.strftime("%Y-%m-%d-%H-%M-%S", time.gmtime())

tf_predictor = tuner.deploy(initial_instance_count=1,
                            instance_type='ml.c5.large',
                            accelerator_type='ml.eia1.medium',
                            endpoint_name=tf_endpoint_name)
```

Final vlidation accuracy: 99.2%

```
 - 1s - loss: 0.0230 - acc: 0.9926 - val_loss: 0.0298 - val_acc: 0.9916
Epoch 17/19
 - 1s - loss: 0.0219 - acc: 0.9926 - val_loss: 0.0307 - val_acc: 0.9913
Epoch 18/19
 - 1s - loss: 0.0201 - acc: 0.9933 - val_loss: 0.0321 - val_acc: 0.9905
Epoch 19/19
 - 1s - loss: 0.0194 - acc: 0.9938 - val_loss: 0.0285 - val_acc: 0.9922
Validation loss    : 0.028494897500296202
Validation accuracy: 0.9922
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/saved_model/simple_save.py:85: calling Saved
ModelBuilder.add_meta_graph_and_variables (from tensorflow.python.saved_model.builder_impl) with legacy_init_op is deprecated
and will be removed in a future version.
Instructions for updating:
Pass your op to the equivalent parameter main_op instead.
2019-09-25 20:47:25,324 sagemaker-containers INFO     Reporting training SUCCESS
Training seconds: 111
Billable seconds: 111

---------------------------------------------------------------------------------!
```

Hyper parameters that is used:

```
SM_HP_BATCH-SIZE=576
SM_HP_LEARNING-RATE=0.000870020323408704
SM_HP_MODEL_DIR=s3://sagemaker-us-east-1-159107795666/sagemaker-tensorflow-scriptmode-2019-09-25-20-22-24-343/model/sagemaker
-tensorflow-190925-2022-009-7fd3495e/model
SM_HP_EPOCHS=19
PYTHONPATH=/opt/ml/code:/usr/local/bin:/usr/lib/python36.zip:/usr/lib/python3.6:/usr/lib/python3.6/lib-dynload:/usr/local/li
b/python3.6/dist-packages:/usr/lib/python3/dist-packages

Invoking script with the following command:

/usr/bin/python mnist-train-cnn.py --batch-size 576 --epochs 19 --learning-rate 0.000870020323408704 --model_dir s3://sagemak
er-us-east-1-159107795666/sagemaker-tensorflow-scriptmode-2019-09-25-20-22-24-343/model/sagemaker-tensorflow-190925-2022-009-
7fd3495e/model


_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
```

Predict again

### Predict again

In [16]:

```
%matplotlib inline
import random
import matplotlib.pyplot as plt

num_samples = 5
indices = random.sample(range(x_val.shape[0] - 1), num_samples)
images = x_val[indices]/255
labels = y_val[indices]

for i in range(num_samples):
    plt.subplot(1,num_samples,i+1)
    plt.imshow(images[i].reshape(28, 28), cmap='gray')
    plt.title(labels[i])
    plt.axis('off')

prediction = tf_predictor.predict(images.reshape(num_samples, 28, 28, 1))['predictions']
prediction = np.array(prediction)
predicted_label = prediction.argmax(axis=1)
print('Predicted labels are: {}'.format(predicted_label))
```

```
Predicted labels are: [1 2 6 9 5]
```