Google Cloud

Introduction to TensorFlow

Machine Learning on Google Cloud Platform

Josh Cogan

Learn how to...

Train a model on Google Cloud

a

Learn how to...

Train a model on Google Cloud

a

Monitor model training

Learn how to...

Train a model on Google Cloud

a

Monitor model training

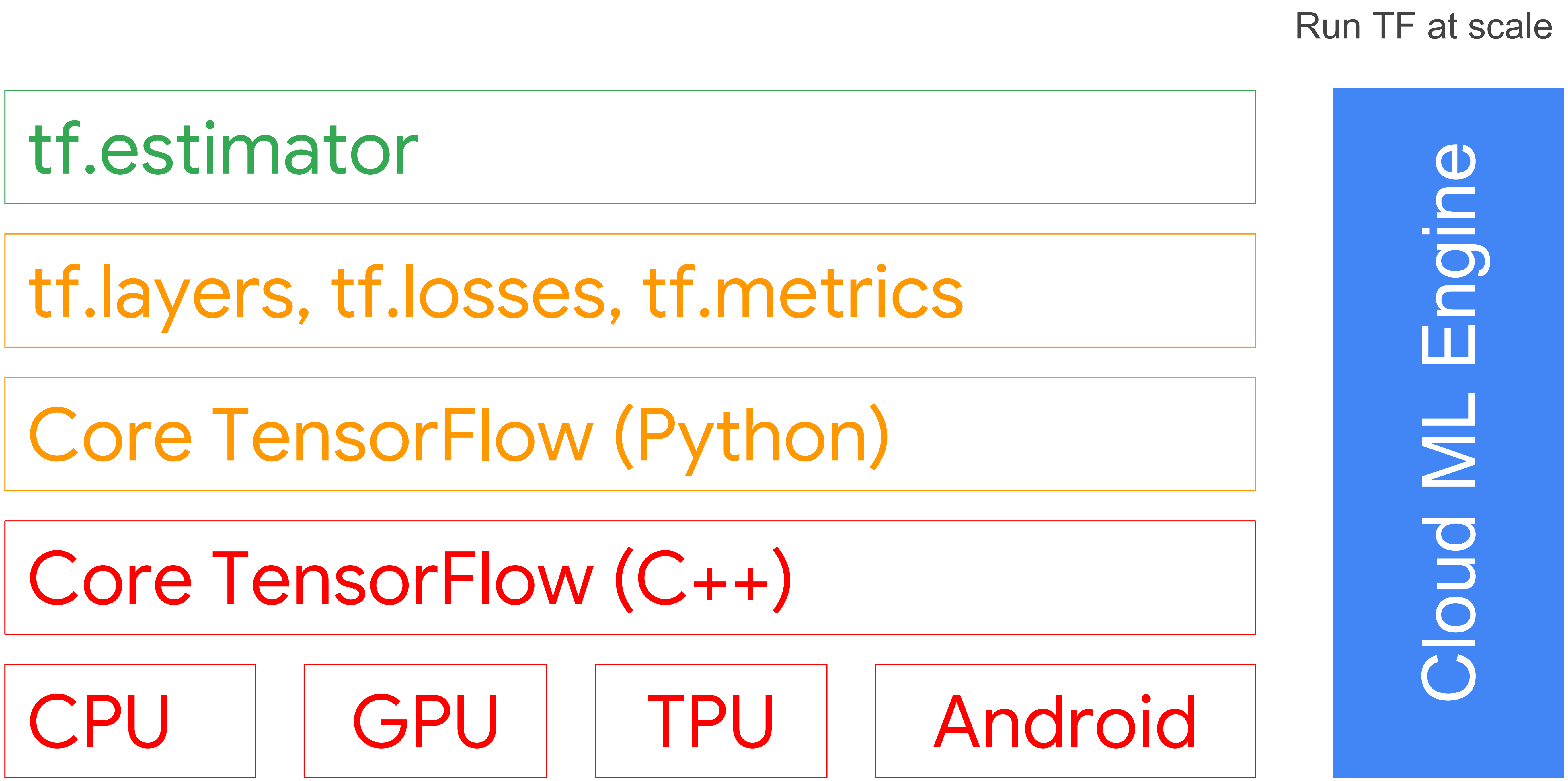
Deploy a trained model as a microservice

Google Cloud

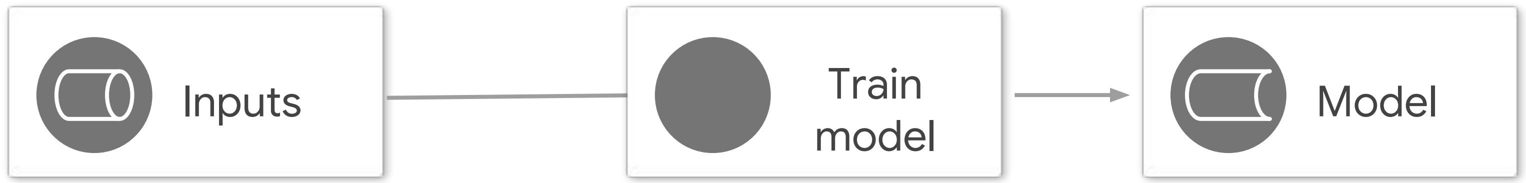
Why Cloud ML Engine?

Josh Cogan

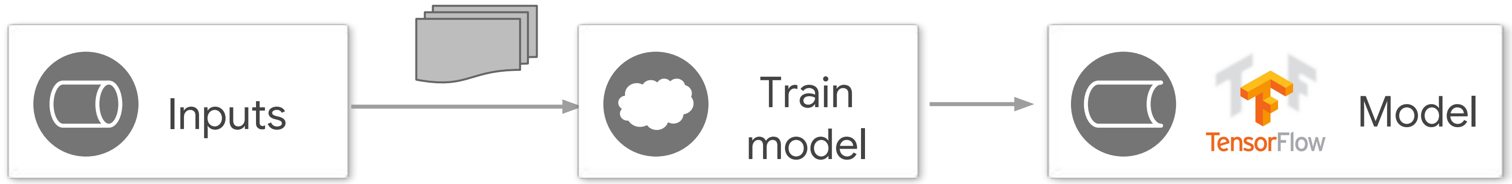
We will use distributed TensorFlow on Cloud ML Engine



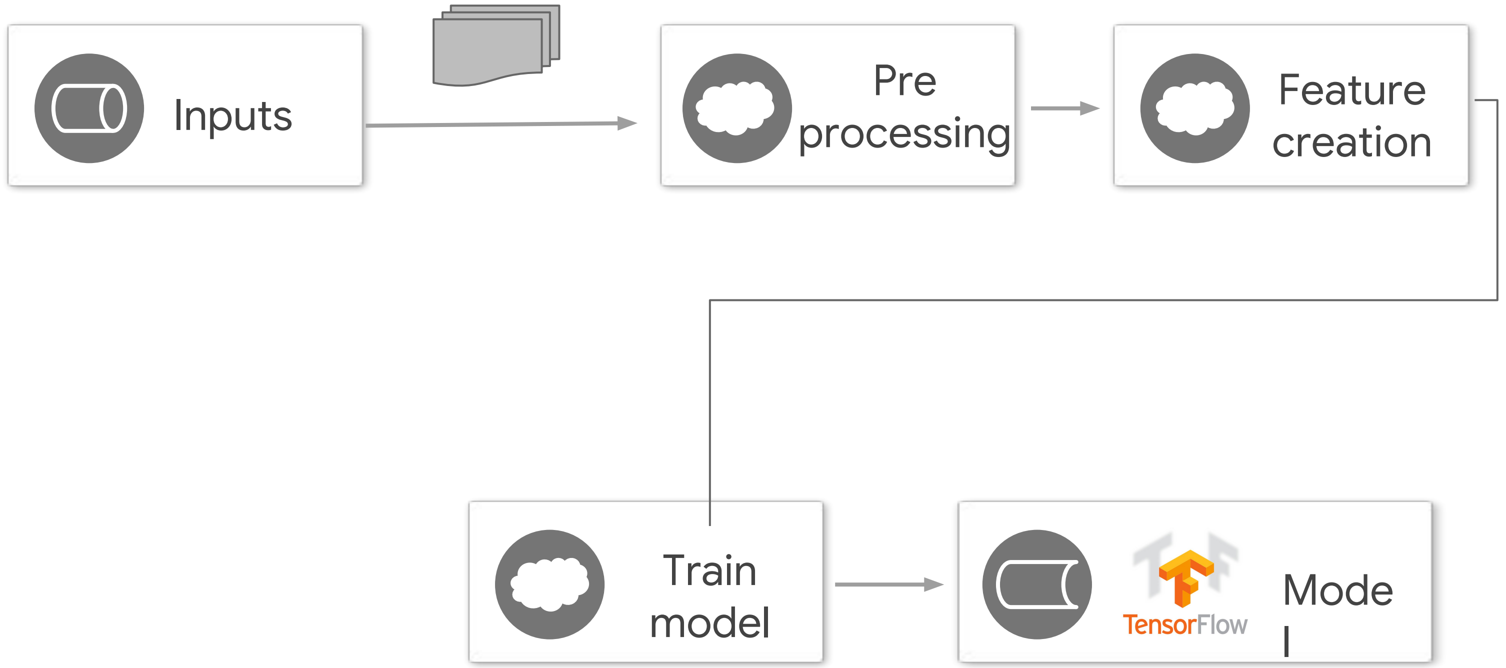
Many machine learning frameworks can handle toy problems



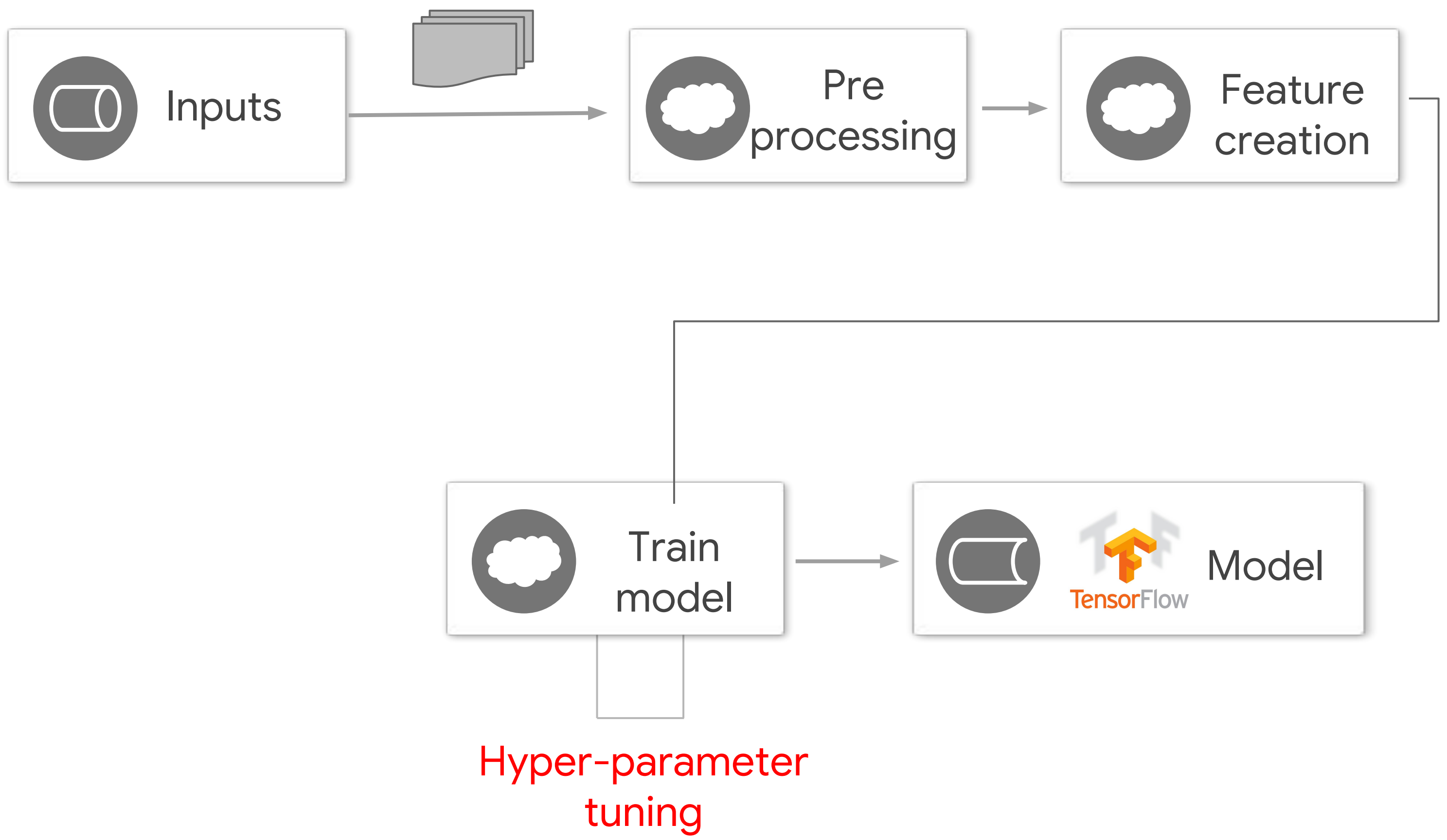
As your data size increases, batching and distribution become important



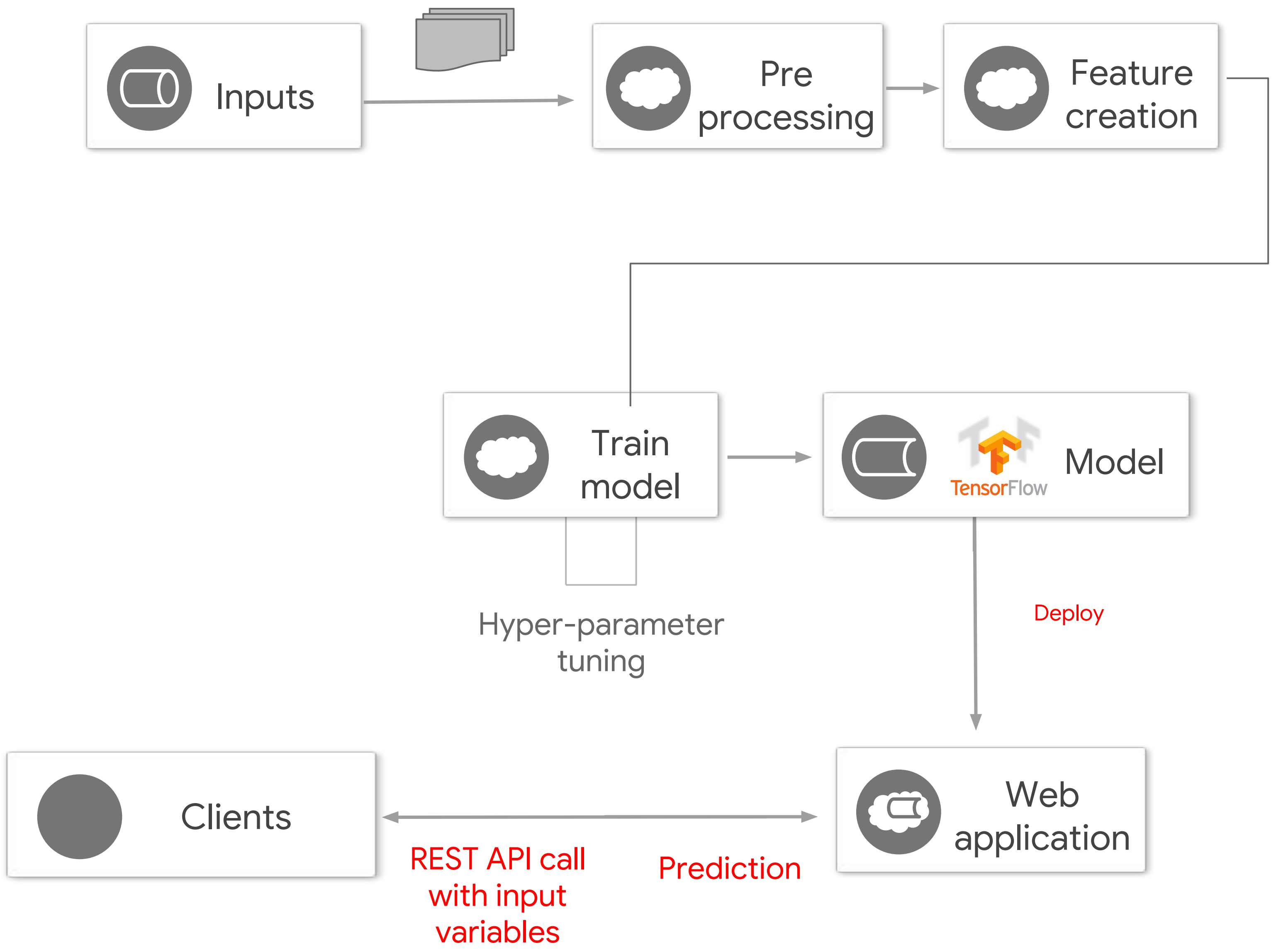
Input necessary transformations



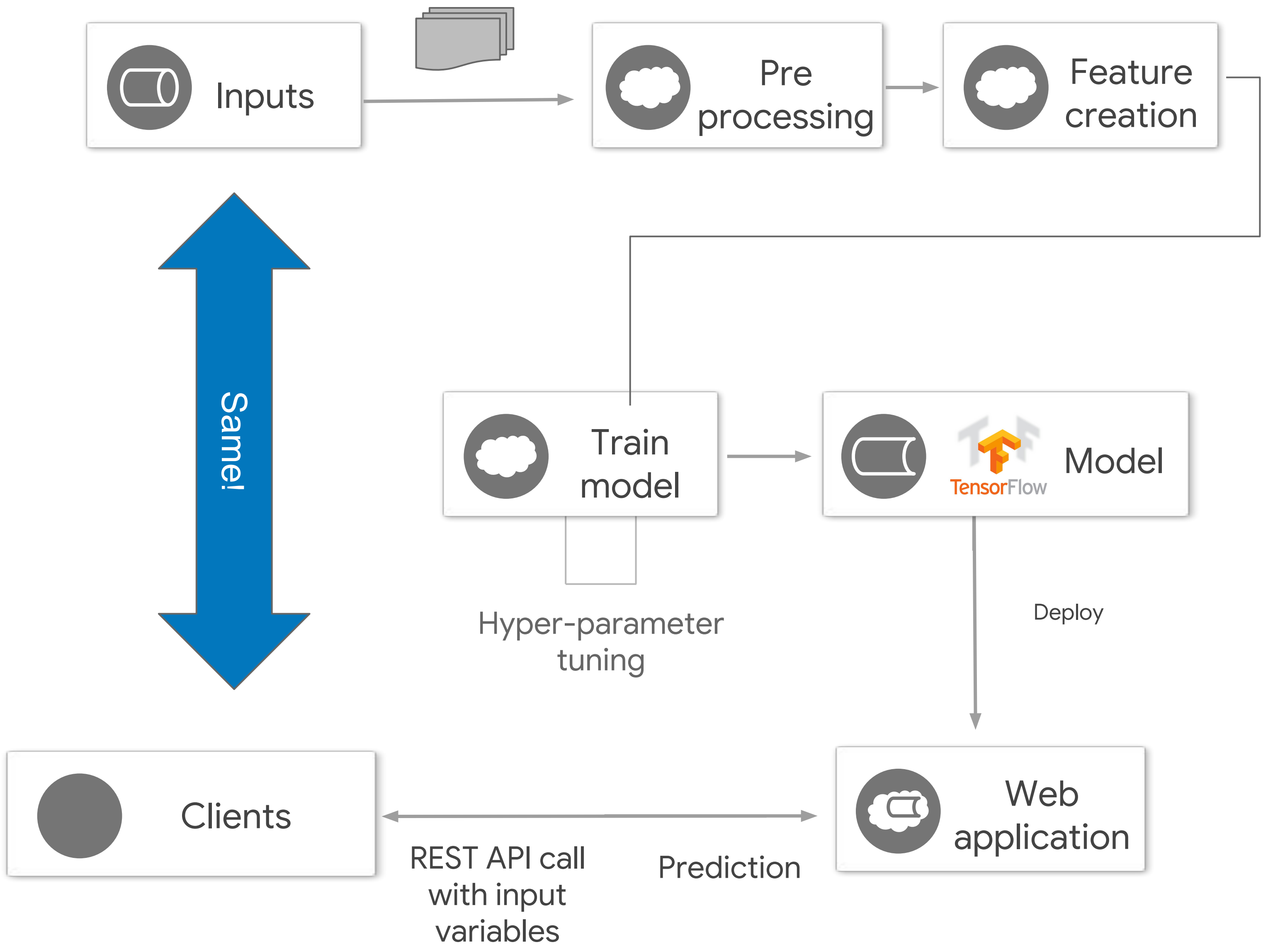
Hyperparameter tuning might be nice



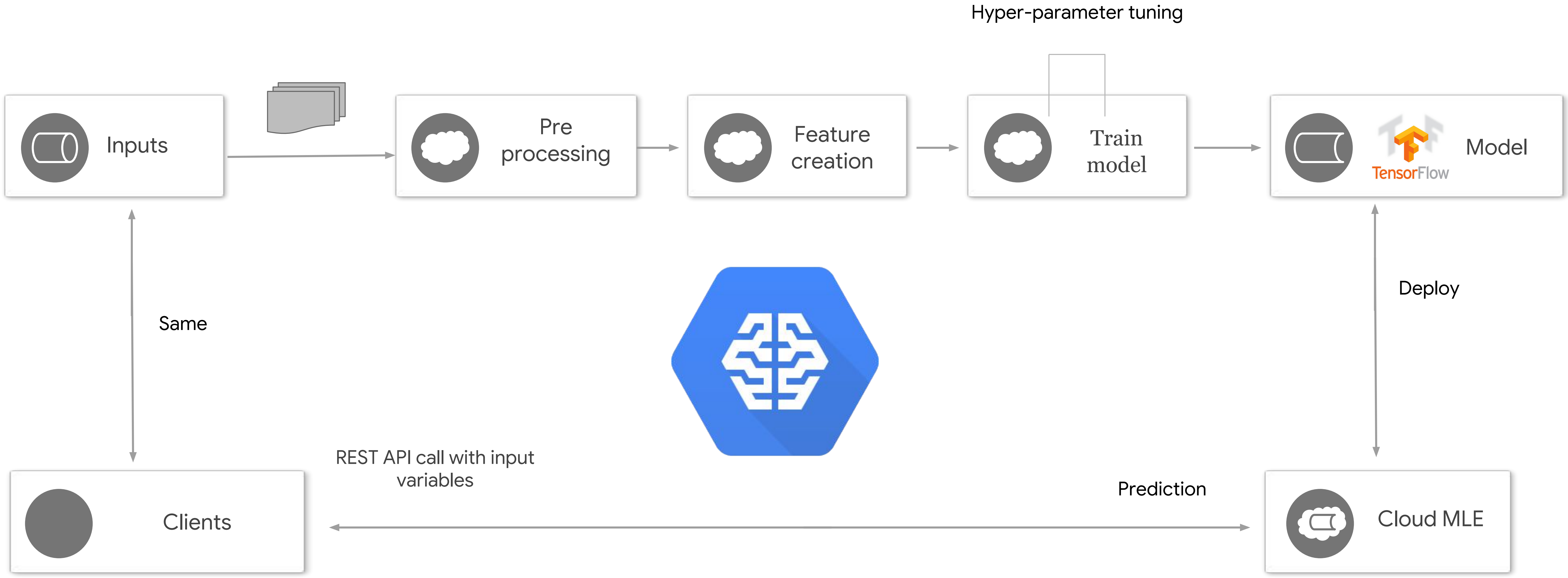
Need to autoscale prediction code



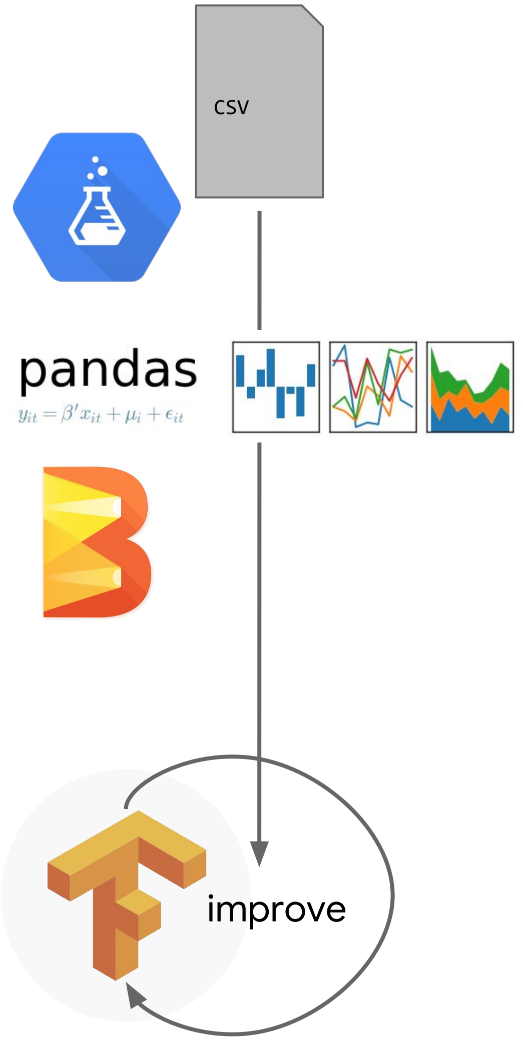
Who does the preprocessing?

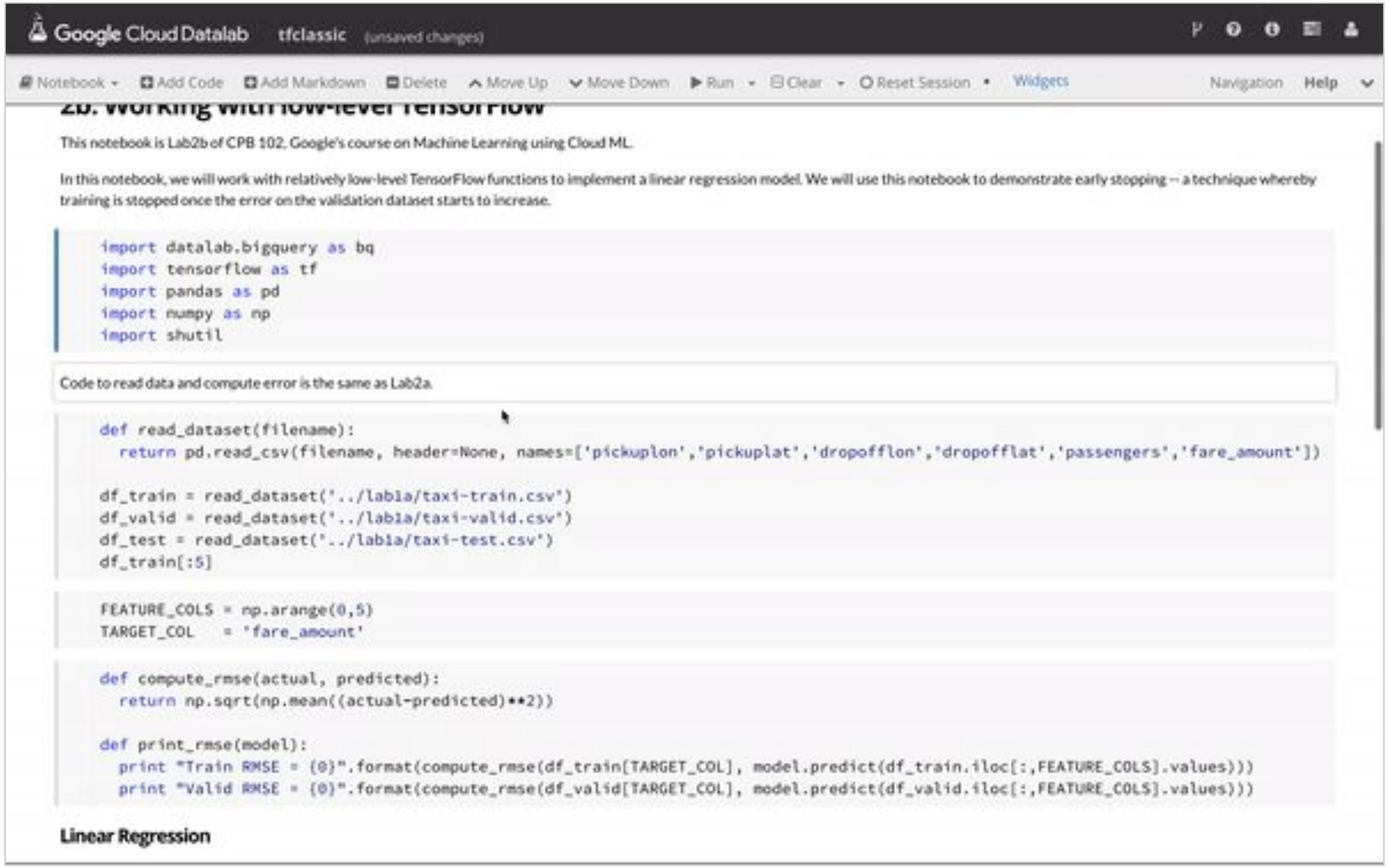


Cloud Machine Learning Engine - repeatable, scalable, tuned

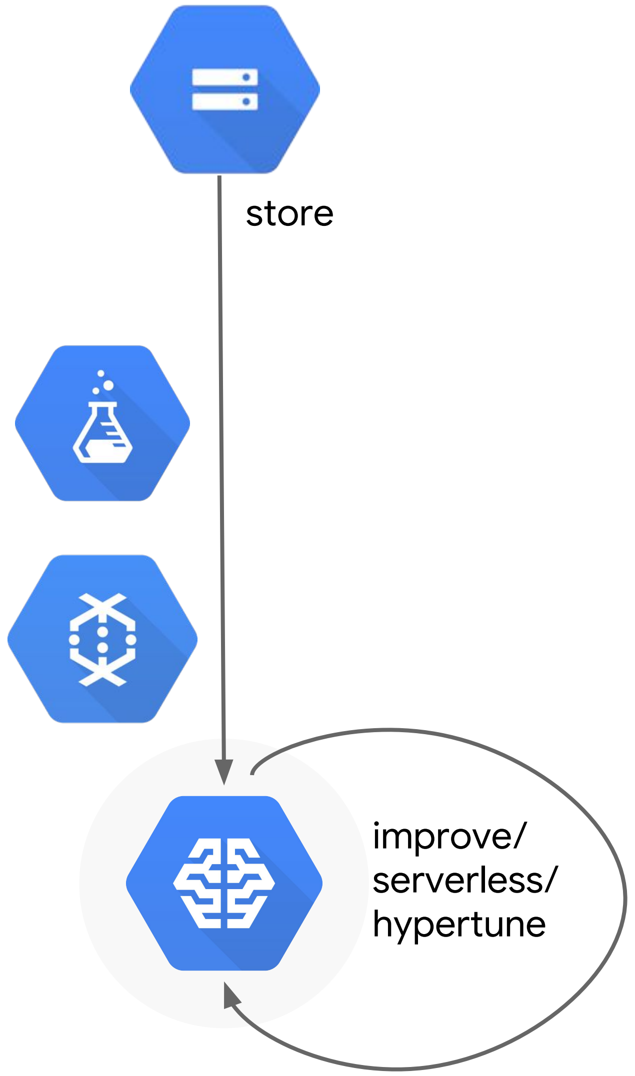


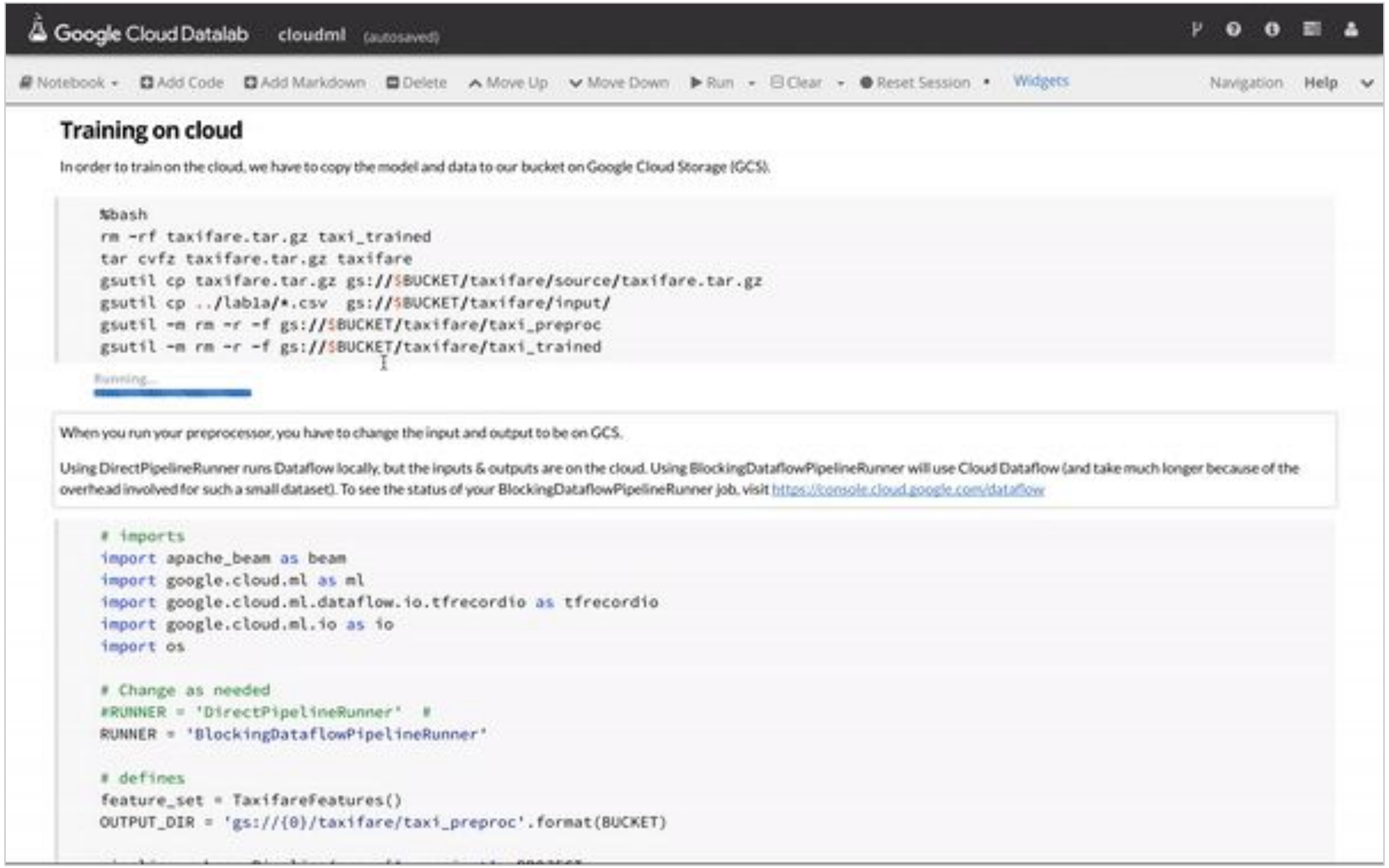
In Datalab, start locally on sampled dataset





Then, scale it out to GCP using serverless technology





Google Cloud

Train a model

Josh Cogan

Training your model with Cloud Machine Learning Engine

1. Use TensorFlow to create computation graph and training application

2. Package your trainer application

3. Configure and start a Cloud ML Engine job

Create task.py to parse command-line parameters and send along to train_and_evaluate

task.py

```
parser.add_argument(
     '--train_data_paths', required=True)
parser.add_argument(
     '--train_steps', ...
```

Create task.py to parse command-line parameters and send along to train_and_evaluate

task.py

```
parser.add_argument(
     '--train_data_paths', required=True)
parser.add_argument(
     '--train_steps', ...
```

model.py

```
def train_and_evaluate(args):
    estimator = tf.estimator.DNNRegressor(
                     model_dir=args['output_dir'],
                     feature_columns=feature_cols,
                     hidden_units=args['hidden_units'])
    train_spec=tf.estimator.TrainSpec(
                     input_fn=read_dataset(args['train_data_paths'],
                                batch_size=args['train_batch_size'],
                                mode=tf.contrib.learn.ModeKeys.TRAIN),
                     max_steps=args['train_steps'])
    exporter = tf.estimator.LatestExporter('exporter', serving_input_fn)
    eval_spec=tf.estimator.EvalSpec(...)
    tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)
```

Create task.py to parse command-line parameters and send along to train_and_evaluate

task.py

```
parser.add_argument(
     '--train_data_paths', required=True)
parser.add_argument(
     '--train_steps', ...
```

model.py

```
def train_and_evaluate(args):
    estimator = tf.estimator.DNNRegressor(
                         model_dir=args['output_dir'],
                         feature_columns=feature_cols,
                         hidden_units=args['hidden_units'])
    train_spec=tf.estimator.TrainSpec(
                         input_fn=read_dataset(args['train_data_paths'],
                                   batch_size=args['train_batch_size'],
                                   mode=tf.contrib.learn.ModeKeys.TRAIN),
                         max_steps=args['train_steps'])
    exporter = tf.estimator.LatestExporter('exporter', serving_input_fn)
    eval_spec=tf.estimator.EvalSpec(...)
    tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)
```

Create task.py to parse command-line parameters and send along to train_and_evaluate

task.py

```
parser.add_argument(
     '--train_data_paths', required=True)
parser.add_argument(
     '--train_steps', ...
```

model.py

```
def train_and_evaluate(args):
    estimator = tf.estimator.DNNRegressor(
                     model_dir=args['output_dir'],
                     feature_columns=feature_cols,
                     hidden_units=args['hidden_units'])
    train_spec=tf.estimator.TrainSpec(
                     input_fn=read_dataset(args['train_data_paths'],
                                    batch_size=args['train_batch_size'],
                                    mode=tf.contrib.learn.ModeKeys.TRAIN),
                     max_steps=args['train_steps'])
    exporter = tf.estimator.LatestExporter('exporter', serving_input_fn)
    eval_spec=tf.estimator.EvalSpec(...)
    tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)
```

The model.py contains the ML model in TensorFlow (Estimator API)

	Example of the code in model.py (see previous chapter)

Training and evaluation input functions	`CSV_COLUMNS = ...`
	`def read_dataset(filename, mode, batch_size=512):`
	`    ...`

Feature columns	`INPUT_COLUMNS = [` `    tf.feature_column.numeric_column('pickuplon'),`

Feature engineering	`def add_more_features(feats):`
	`  # will be covered in next course; for now, just a no-op`
	`  return feats`

Serving input function	`def serving_input_fn():`
	`    ...`
	`    return tf.estimator.export.ServingInputReceiver(features, feature_pholders)`

Train and evaluate loop	`def train_and_evaluate(args):`
	`    ...`
	`    tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)`

The model.py contains the ML model in TensorFlow (Estimator API)

	Example of the code in model.py (see previous chapter)

Training and evaluation input functions	`CSV_COLUMNS = ...`
	`def read_dataset(filename, mode, batch_size=512):`
	`    ...`

Feature columns	`INPUT_COLUMNS = [` `    tf.feature_column.numeric_column('pickuplon'),`

Feature engineering	`def add_more_features(feats):`
	`  # will be covered in next course; for now, just a no-op`
	`  return feats`

Serving input function	`def serving_input_fn():`
	`    ...`
	`   return tf.estimator.export.ServingInputReceiver(features, feature_pholders)`

Train and evaluate loop	`def train_and_evaluate(args):`
	`    ...`
	`    tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)`

The model.py contains the ML model in TensorFlow (Estimator API)

	Example of the code in model.py (see previous chapter)

Training and evaluation input functions	`CSV_COLUMNS = ...`
	`def read_dataset(filename, mode, batch_size=512):`
	`    ...`

Feature columns	`INPUT_COLUMNS = [` `    tf.feature_column.numeric_column('pickuplon'),`

Feature engineering	`def add_more_features(feats):`
	`  # will be covered in next course; for now, just a no-op`
	`  return feats`

Serving input function	`def serving_input_fn():`
	`    ...`
	`    return tf.estimator.export.ServingInputReceiver(features, feature_pholders)`

Train and evaluate loop	`def train_and_evaluate(args):`
	`    ...`
	`    tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)`

The model.py contains the ML model in TensorFlow (Estimator API)

	Example of the code in model.py (see previous chapter)

Training and evaluation input functions	`CSV_COLUMNS = ...`
	`def read_dataset(filename, mode, batch_size=512):`
	`    ...`

Feature columns	`INPUT_COLUMNS = [` `    tf.feature_column.numeric_column('pickuplon'),`

Feature engineering	`def add_more_features(feats):`
	`  # will be covered in next course; for now, just a no-op`
	`  return feats`

Serving input function	`def serving_input_fn():`
	`    ...`
	`    return tf.estimator.export.ServingInputReceiver(features, feature_pholders)`

Train and evaluate loop	`def train_and_evaluate(args):`
	`    ...`
	`    tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)`

The model.py contains the ML model in TensorFlow (Estimator API)

	Example of the code in model.py (see previous chapter)

Training and evaluation input functions	`CSV_COLUMNS = ...`
	`def read_dataset(filename, mode, batch_size=512):`
	`    ...`

Feature columns	`INPUT_COLUMNS = [` `    tf.feature_column.numeric_column('pickuplon'),`

Feature engineering	`def add_more_features(feats):`
	`  # will be covered in next course; for now, just a no-op`
	`  return feats`

Serving input function	`def serving_input_fn():`
	`    ...`
	`    return tf.estimator.export.ServingInputReceiver(features, feature_pholders)`

Train and evaluate loop	`def train_and_evaluate(args):`
	`    ...`
	`    tf.estimator.train_and_evaluate(estimator, train_spec, eval_spec)`

Package up TensorFlow model as Python package

```
taxifare/
taxifare/PKG-INFO
taxifare/setup.cfg
taxifare/setup.py
taxifare/trainer/
taxifare/trainer/__init__.py
taxifare/trainer/task.py
taxifare/trainer/model.py
```

Python modules need to contain an __init__.py in every folder

Verify that the model works as a Python package

```
export PYTHONPATH=${PYTHONPATH}:/somedir/taxifare
python -m trainer.task \
  --train_data_paths="/somedir/datasets/*train*" \
  --eval_data_paths=/somedir/datasets/*valid*  \
  --output_dir=/somedir/output \
  --train_steps=100 --job-dir=/tmp
```

Verify that the model works as a Python package

```
export PYTHONPATH=${PYTHONPATH}:/somedir/taxifare
python -m trainer.task \
  --train_data_paths="/somedir/datasets/*train*" \
  --eval_data_paths=/somedir/datasets/*valid*  \
  --output_dir=/somedir/output \
  --train_steps=100 --job-dir=/tmp
```

Then use the gcloud command to submit the training job, either locally or to cloud

```
gcloud ml-engine local train \
   --module-name=trainer.task \
   --package-path=/somedir/taxifare/trainer \
   -- \
   --train_data_paths etc.
   REST as before
```

```
gcloud ml-engine jobs submit training $JOBNAME \
   --region=$REGION \
   --module-name=trainer.task \
   --job-dir=$OUTDIR --staging-bucket=gs://$BUCKET \
   --scale-tier=BASIC \
   REST as before
```

Then use the gcloud command to submit the training job, either locally or to cloud

```
gcloud ml-engine local train \
   --module-name=trainer.task \
   --package-path=/somedir/taxifare/trainer \
   -- \
   --train_data_paths etc.
   REST as before
```

```
gcloud ml-engine jobs submit training $JOBNAME \
   --region=$REGION \
   --module-name=trainer.task \
   --job-dir=$OUTDIR --staging-bucket=gs://$BUCKET \
   --scale-tier=BASIC \
   REST as before
```

Then use the gcloud command to submit the training job, either locally or to cloud

```
gcloud ml-engine local train \
   --module-name=trainer.task \
   --package-path=/somedir/taxifare/trainer \
   -- \
   --train_data_paths etc.
   REST as before
```

```
gcloud ml-engine jobs submit training $JOBNAME \
   --region=$REGION \
   --module-name=trainer.task \
   --job-dir=$OUTDIR --staging-bucket=gs://$BUCKET \
   --scale-tier=BASIC \
   REST as before
```

Then use the gcloud command to submit the training job, either locally or to cloud

```
gcloud ml-engine local train \
   --module-name=trainer.task \
   --package-path=/somedir/taxifare/trainer \
   -- \
   --train_data_paths etc.
   REST as before
```

```
gcloud ml-engine jobs submit training $JOBNAME \
   --region=$REGION \
   --module-name=trainer.task \
   --job-dir=$OUTDIR --staging-bucket=gs://$BUCKET \
   --scale-tier=BASIC \
   REST as before
```

Scale Tier Options

BASIC

Scale Tier Options

BASIC

STANDARD

Scale Tier Options

BASIC

STANDARD

BASIC_GPU

Scale Tier Options

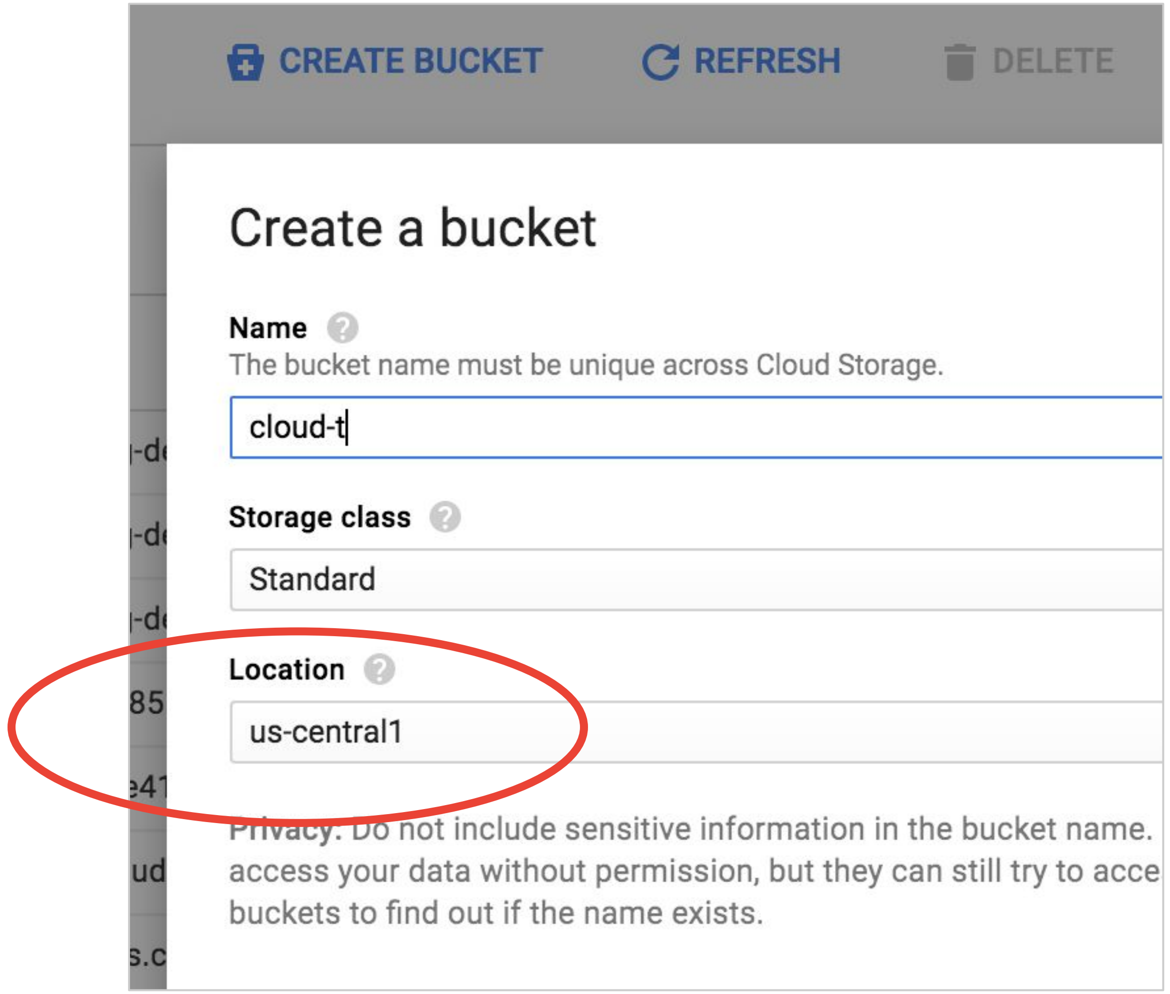
BASIC

STANDARD

BASIC_GPU

BASIC_TPU

Tip: Use single-region bucket for ML



Google Cloud

Monitoring and Deploying a Trained Model

Josh Cogan

Monitor training jobs with gcloud

Get details of current state of job

```
gcloud ml-engine jobs describe job_name
```

Monitor training jobs with gcloud

Get details of current state of job

```
gcloud ml-engine jobs describe job_name
```

Get latest logs from job

```
gcloud ml-engine jobs stream-jobs job_name
```

Monitor training jobs with gcloud

Get details of current state of job

```
gcloud ml-engine jobs describe job_name
```

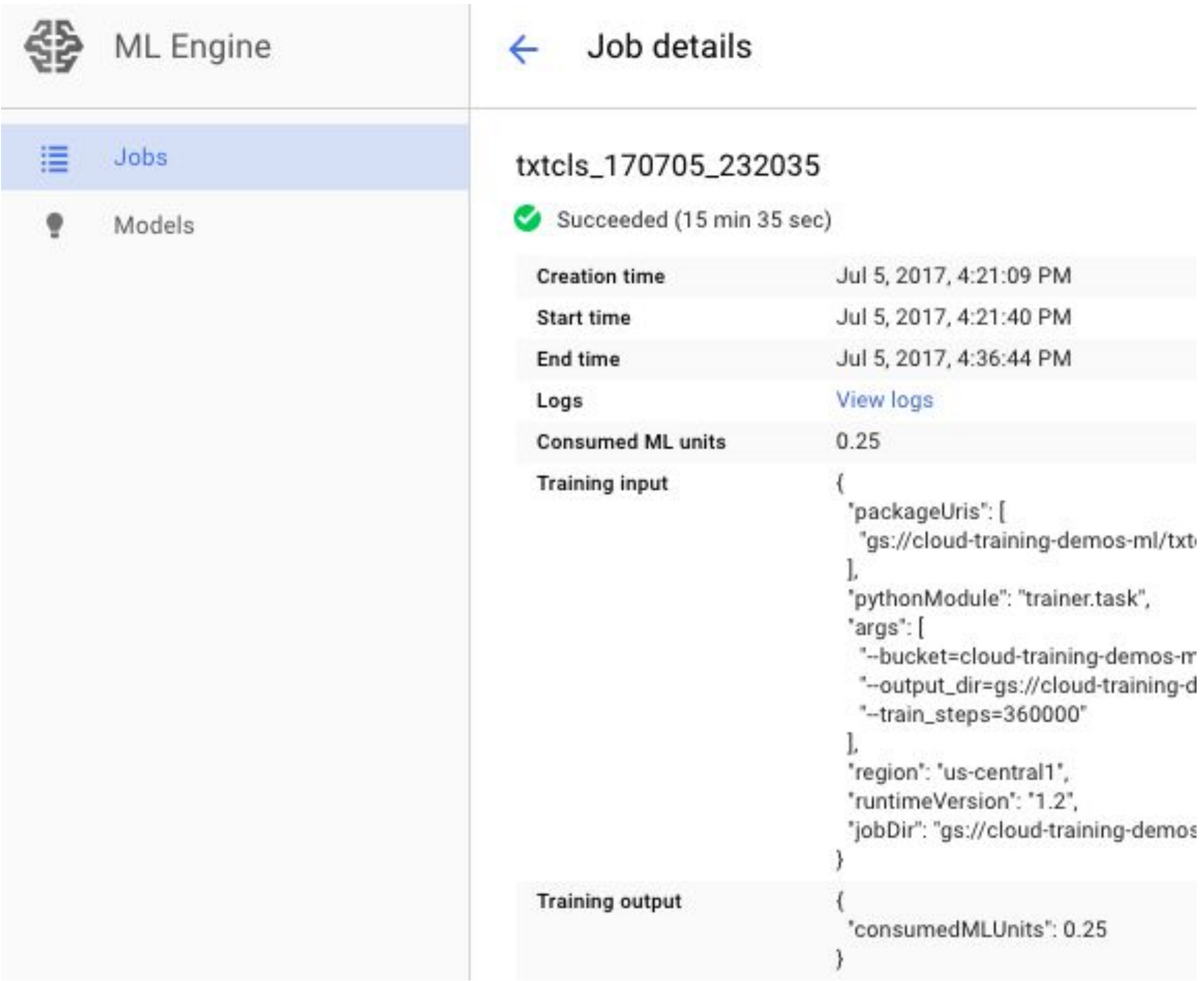
Get latest logs from job

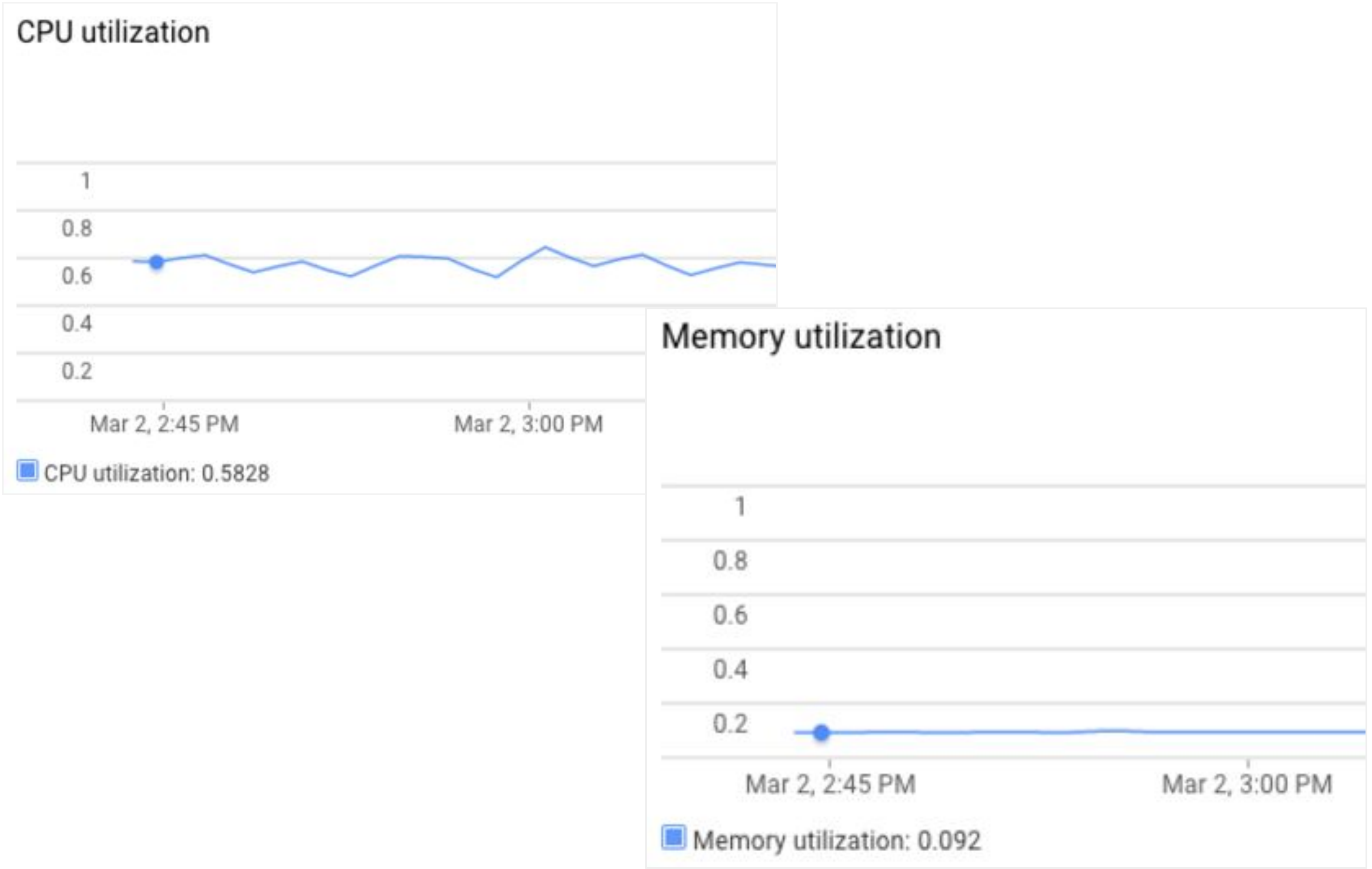
```
gcloud ml-engine jobs stream-jobs job_name
```

Filter jobs based on creation time or name

```
gcloud ml-engine jobs list --filter='createTime>2017-01-15T19:00'
gcloud ml-engine jobs list --filter='jobId:census*' --limit=3
```

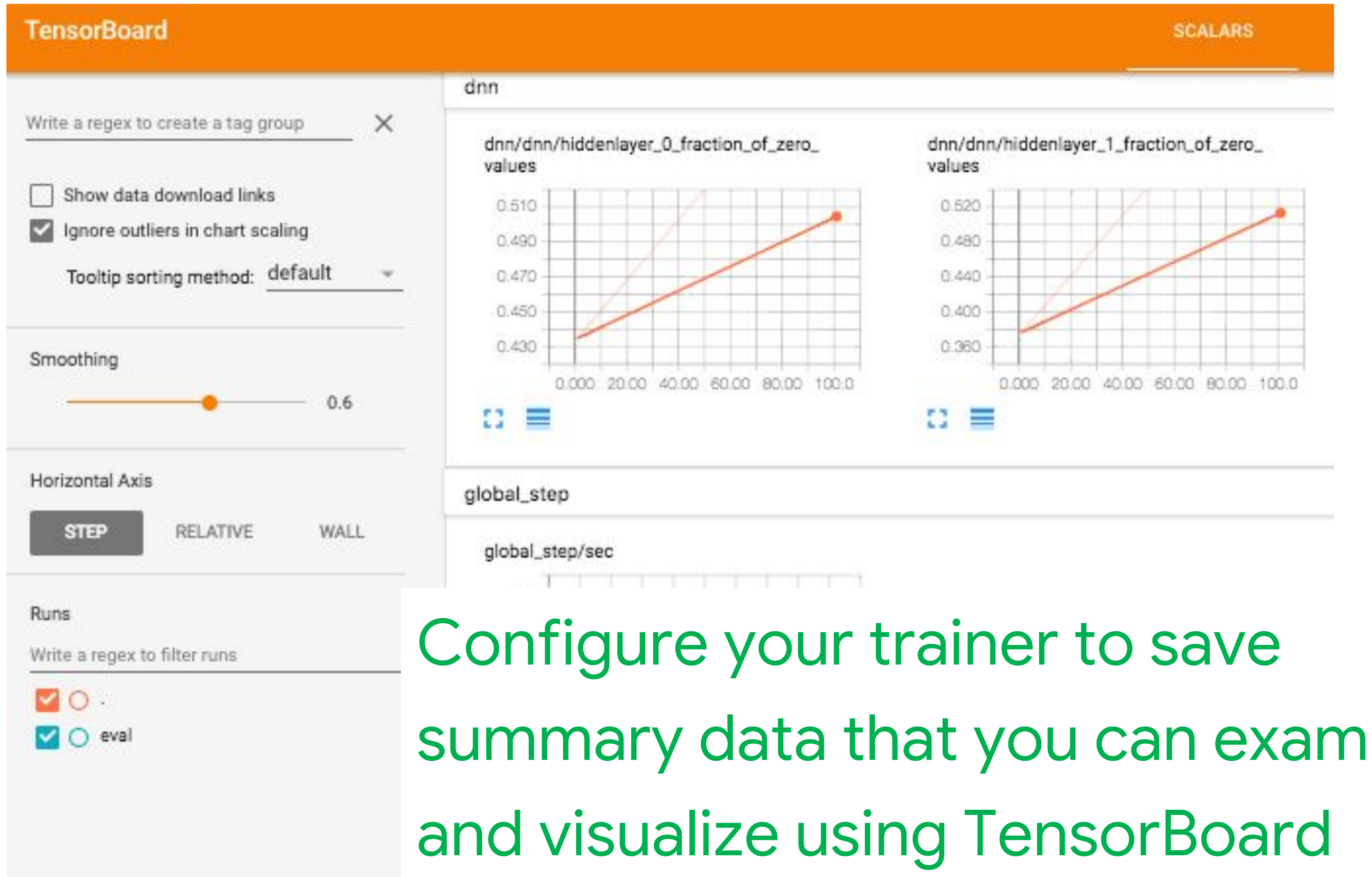
Monitor training jobs with GCP console





You can also view CPU and Memory utilization charts for this training job with Stack Driver Monitoring

Monitor training jobs with TensorBoard

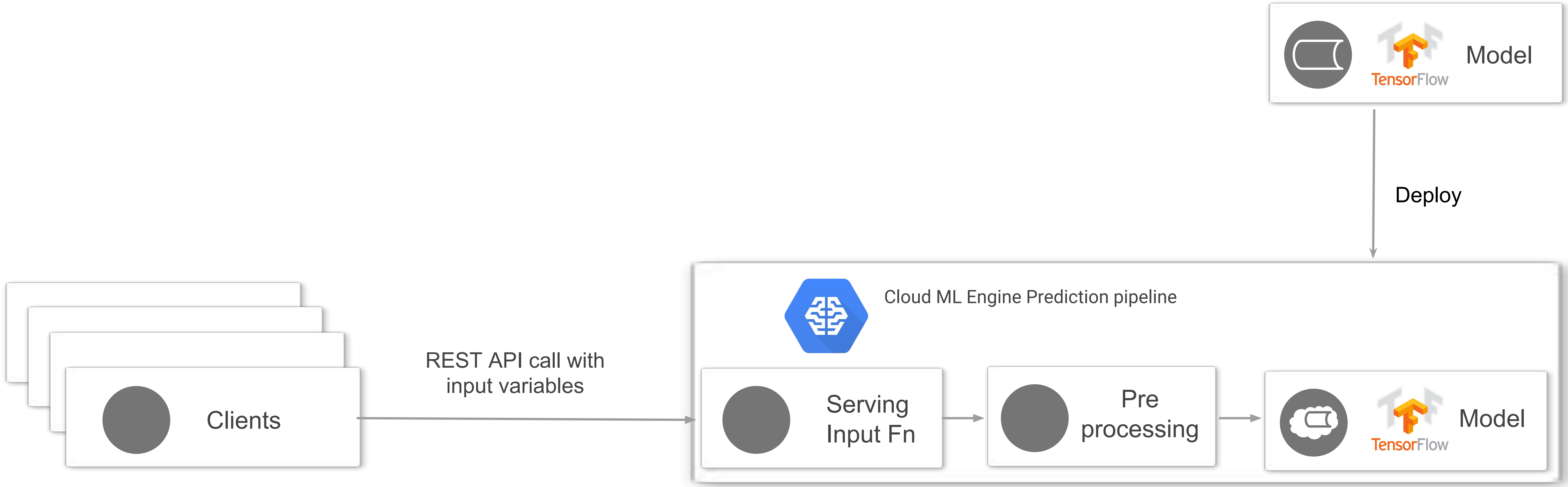


Configure your trainer to save summary data that you can examine and visualize using TensorBoard

Deploy a trained model and make predictions

Josh Cogan

Cloud ML Engine makes deploying models and scaling the infrastructure easy



Deploy the saved model to GCP

Could also be a locally-trained model

```
MODEL_NAME="taxifare"
MODEL_VERSION="v1"
MODEL_LOCATION="gs://${BUCKET}/taxifare/smallinput/taxi_trained/export/Servo/.../"

gcloud ml-engine models create ${MODEL_NAME} --regions $REGION
gcloud ml-engine versions create ${MODEL_VERSION} --model ${MODEL_NAME} --origin
${MODEL_LOCATION} --runtime-version 1.4
```

Deploy the saved model to GCP

Could also be a locally-trained model

```
MODEL_NAME="taxifare"
MODEL_VERSION="v1"
MODEL_LOCATION="gs://${BUCKET}/taxifare/smallinput/taxi_trained/export/Servo/.../"

gcloud ml-engine models create ${MODEL_NAME} --regions $REGION
gcloud ml-engine versions create ${MODEL_VERSION} --model ${MODEL_NAME} --origin
${MODEL_LOCATION} --runtime-version 1.4
```

Client code can make REST calls

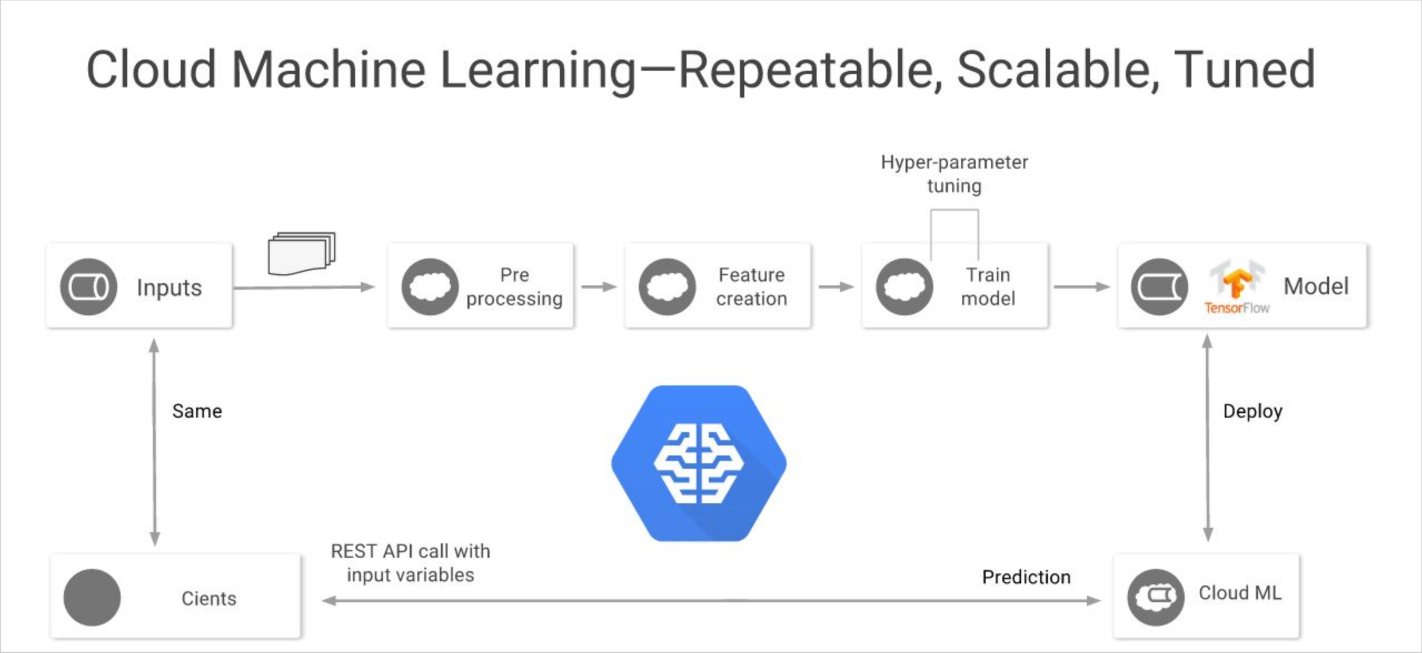
```
credentials = GoogleCredentials.get_application_default()
api = discovery.build('ml', 'v1', credentials=credentials)
request_data = [
    {'pickup_longitude': -73.885262,
     'pickup_latitude': 40.773008,
     'dropoff_longitude': -73.987232,
     'dropoff_latitude': 40.732403,
     'passenger_count': 2}]
parent = 'projects/%s/models/%s/versions/%s' % ('cloud-training-demos', 'taxifare', 'v1')
response = api.projects().predict(body={'instances': request_data},
name=parent).execute()
```

Lab

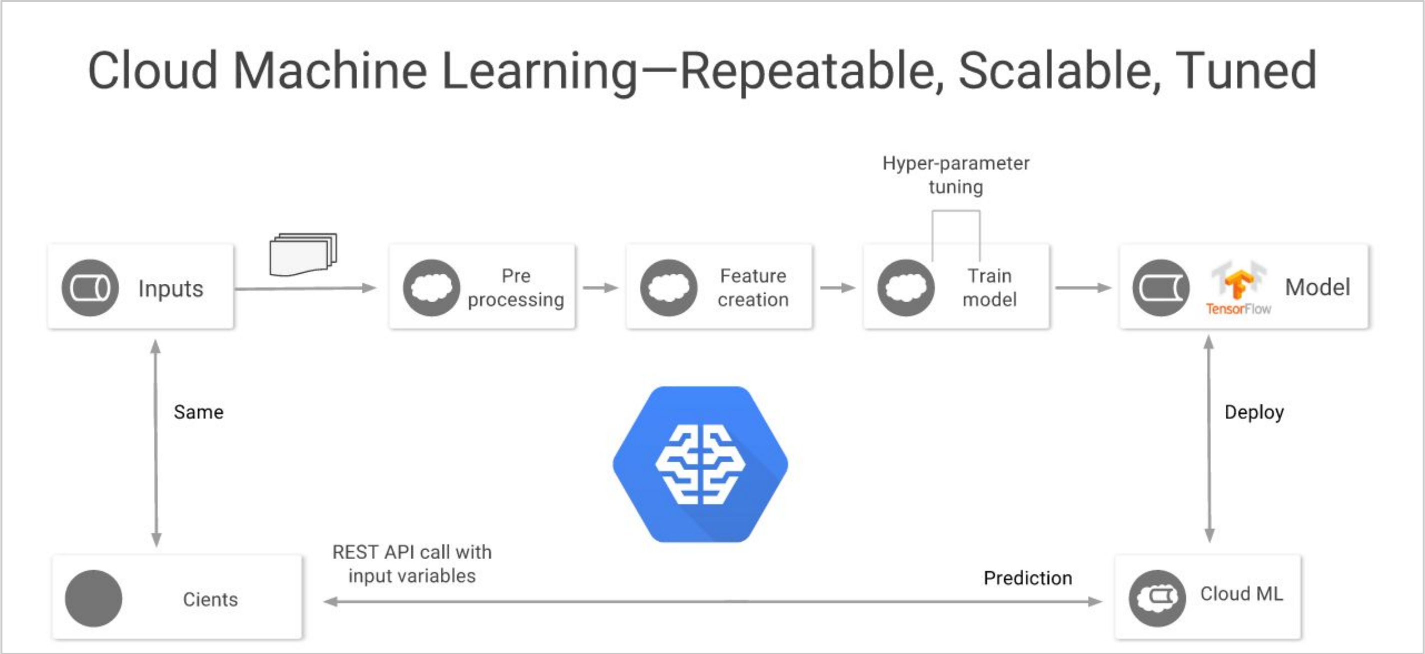
Scaling TensorFlow with Cloud MLE

Josh Cogan

Lab: Scaling TensorFlow with Cloud Machine Learning Engine

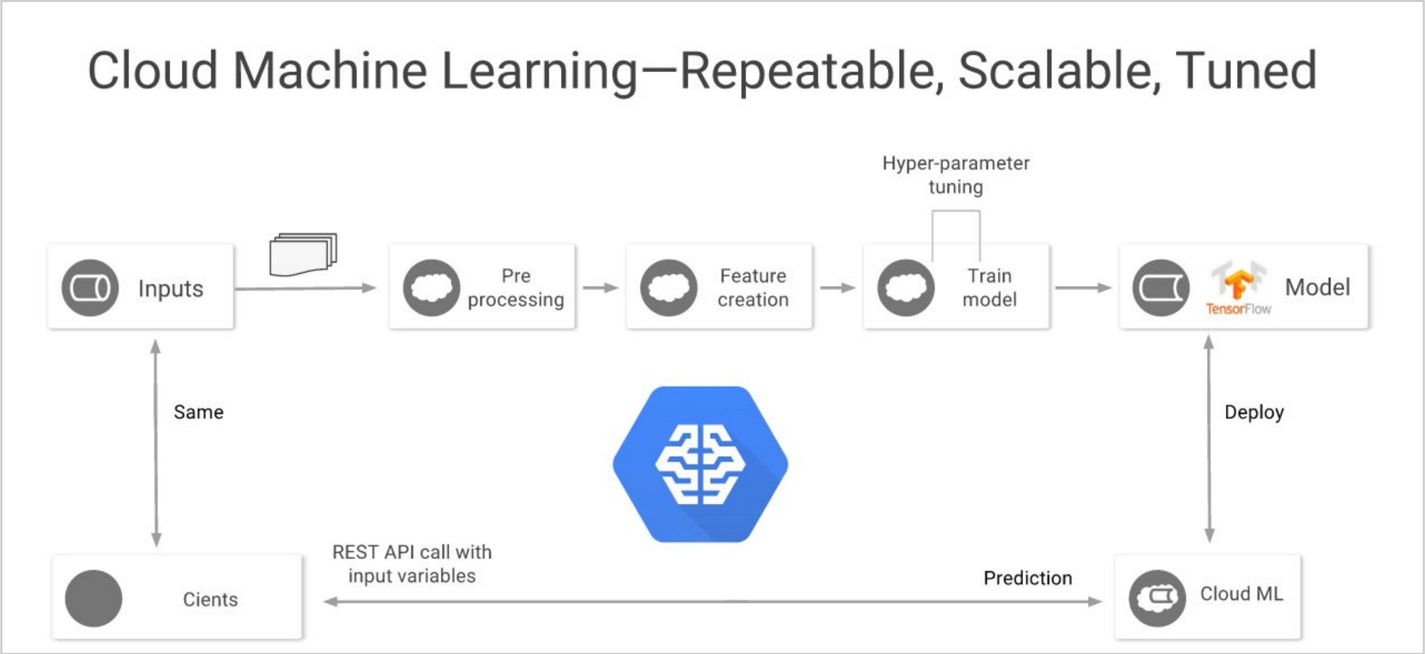


Lab: Scaling TensorFlow with Cloud Machine Learning Engine



Package up TensorFlow model

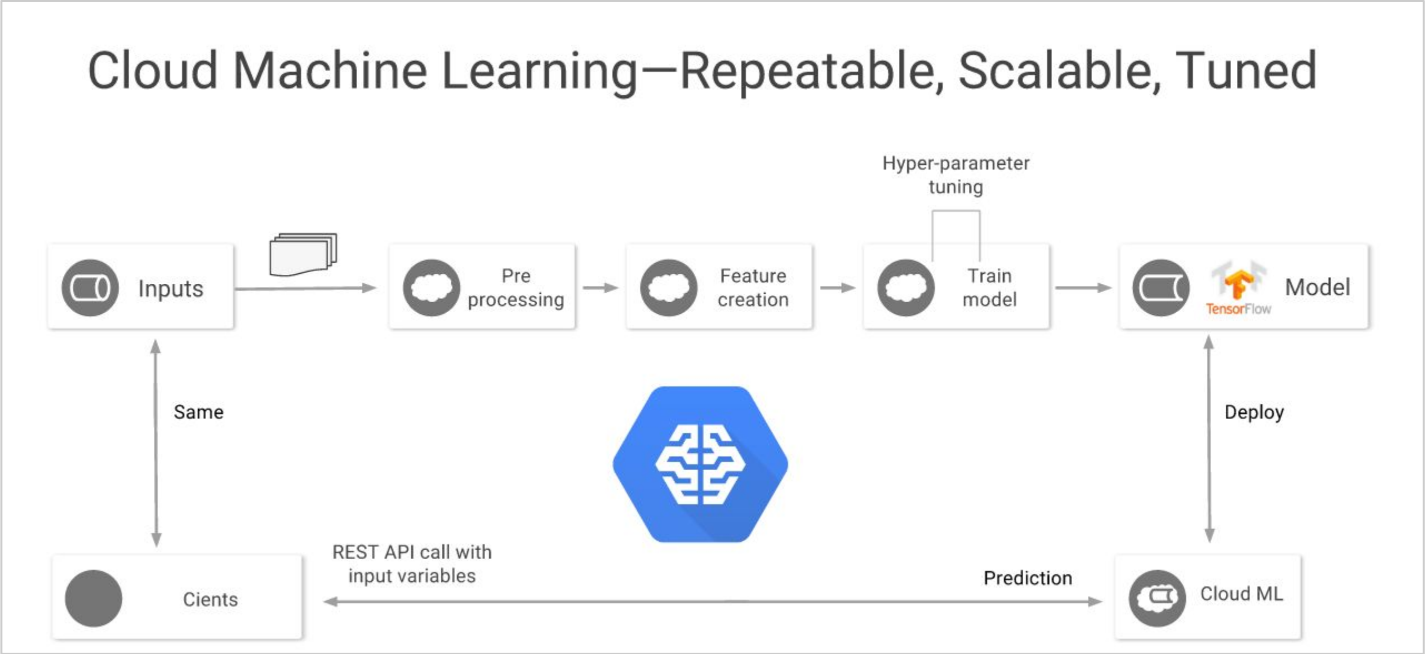
Lab: Scaling TensorFlow with Cloud Machine Learning Engine



Package up TensorFlow model

Run training locally

Lab: Scaling TensorFlow with Cloud Machine Learning Engine

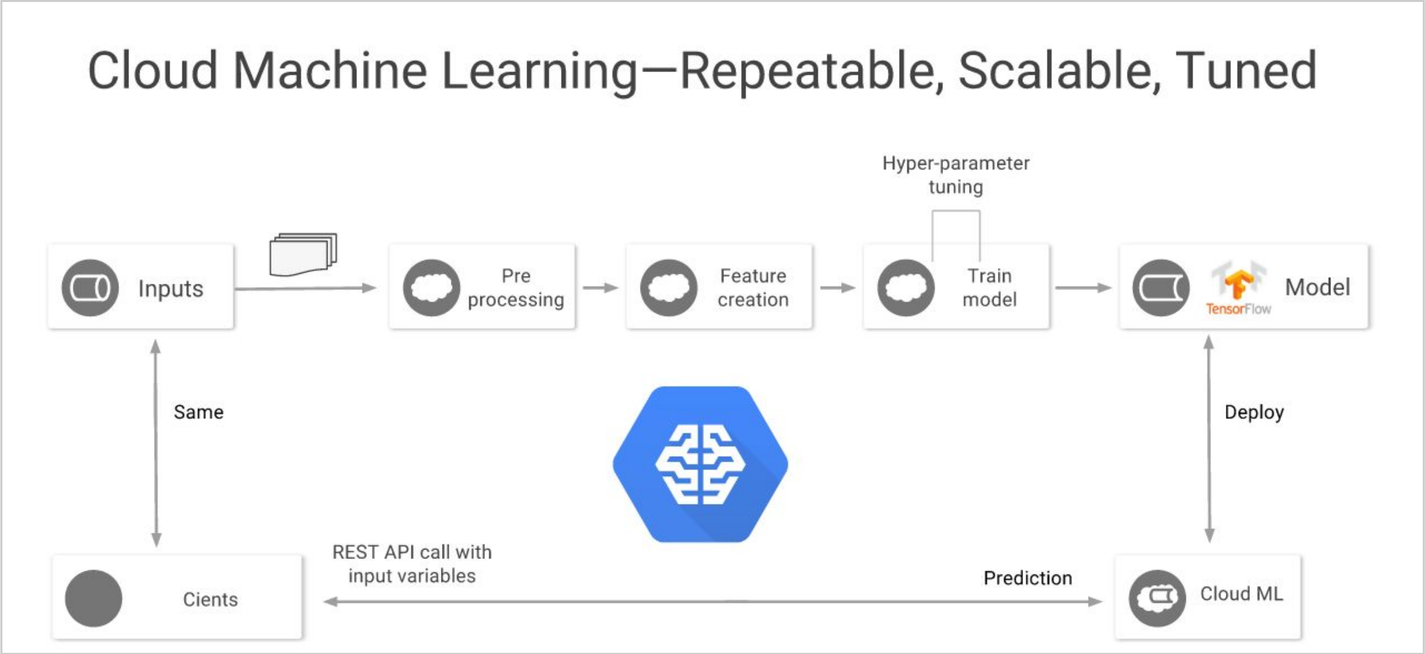


Package up TensorFlow model

Run training locally

Run training on cloud

Lab: Scaling TensorFlow with Cloud Machine Learning Engine



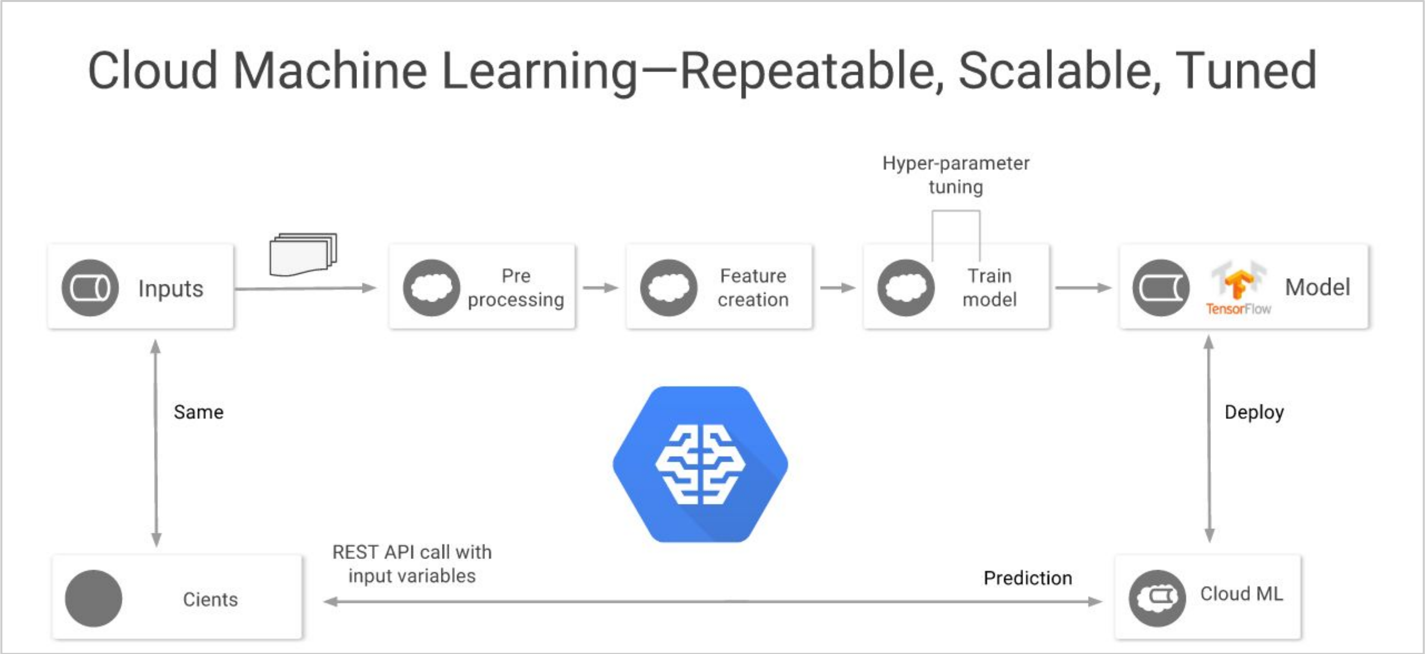
Package up TensorFlow model

Run training locally

Run training on cloud

Deploy model to cloud

Lab: Scaling TensorFlow with Cloud Machine Learning Engine



Package up TensorFlow model

Run training locally

Run training on cloud

Deploy model to cloud

Invoke model to carry out predictions

cloud.google.com