November 5, 2018

Managed ML Model Training and Serving

Lisa Quera, ML Specialist

Google Cloud

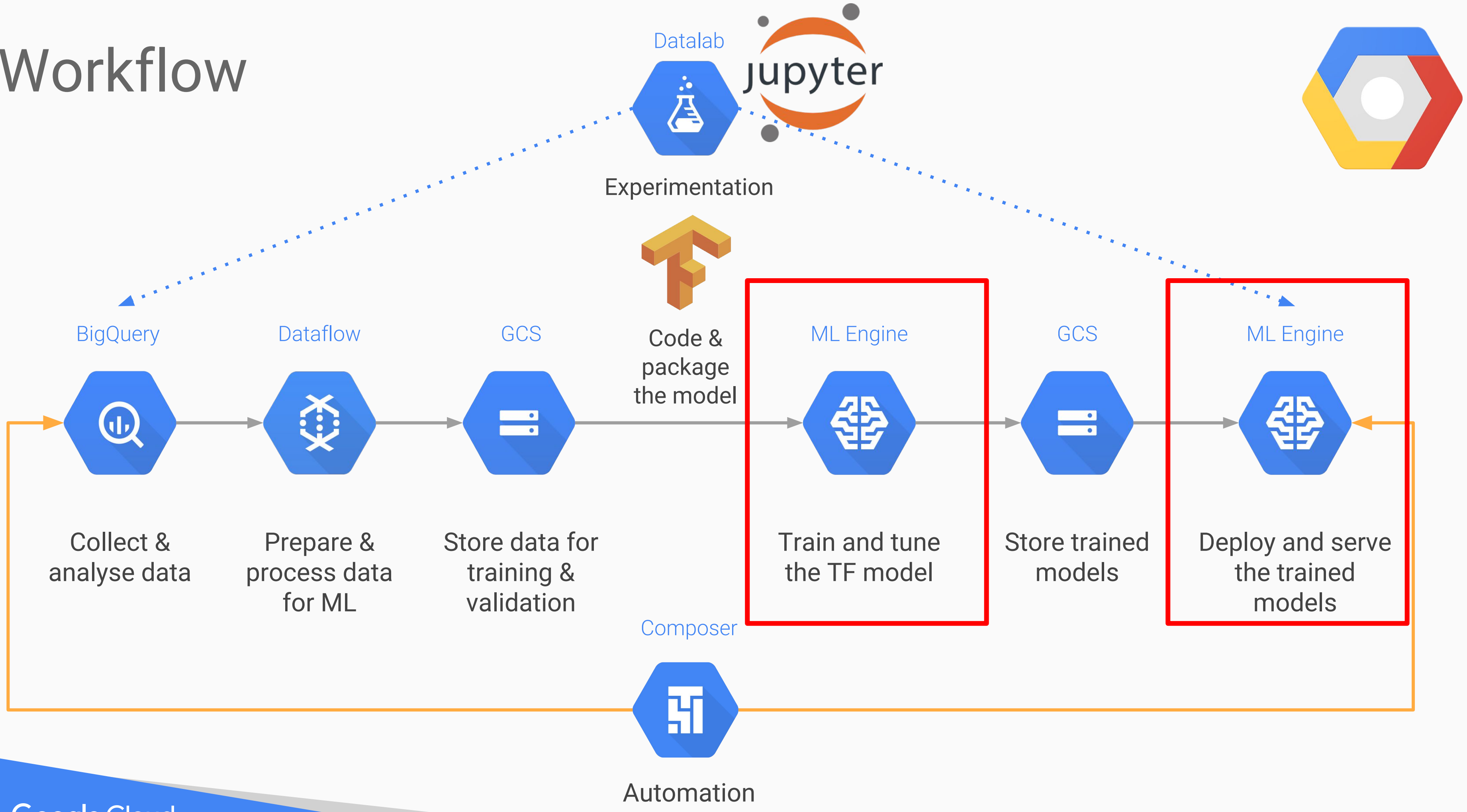
Machine Learning at Scale

Cloud ML Engine

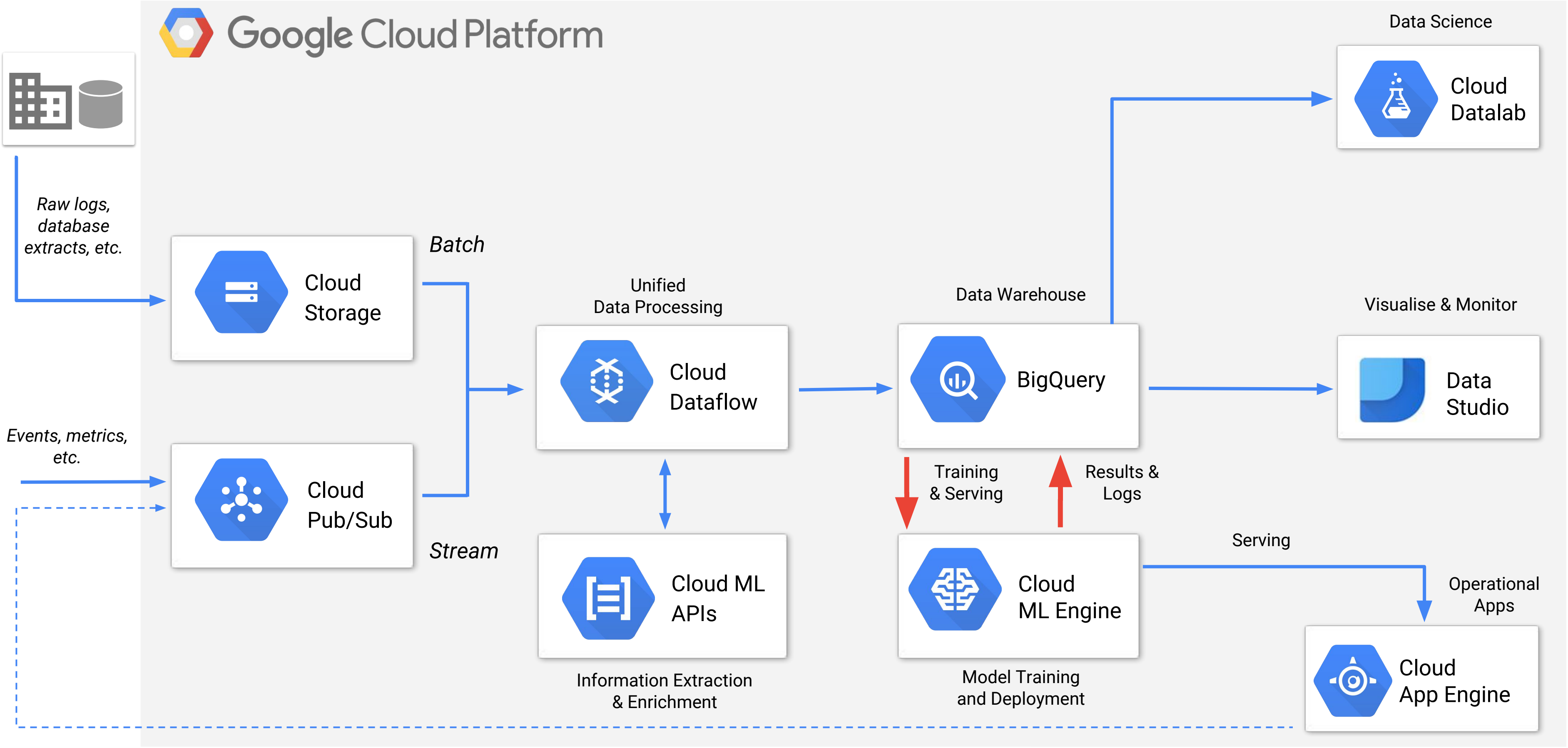
- Serverless, no-ops, ML training and serving platform
- Distributed training infrastructure that supports CPUs, GPUs, and TPUs
- Automatic hyperparameter tuning
- Train, tune, and serve TensorFlow models (batch and online prediction)
- Train and serve Scikit-learn and XGBoost models for online predictions
- TPUs for training TensorFlow models **(Beta)**

Google Cloud

Workflow


Datalab
jupyter
Experimentation
BigQuery
Dataflow
GCS
Code & package the model
ML Engine
GCS
ML Engine
Collect & analyse data
Prepare & process data for ML
Store data for training & validation
Train and tune the TF model
Store trained models
Deploy and serve the trained models
Composer
Automation

Example Architecture: BigQuery to deployed model API



Training locally

train locally

Local path

output directory

```
gcloud ml-engine local train \
    --module-name trainer.task --package-path trainer/ \
    -- \
    --train-files $TRAIN_DATA --eval-files $EVAL_DATA --job-dir $MODEL_DIR
```

training data

evaluation data

Google Cloud

Training in the cloud

with single node

train in the cloud

GCS location

Can be a package in GCS

region

```
gcloud ml-engine jobs submit training $JOB_NAME --job-dir $OUTPUT_PATH \
   --runtime-version 1.10 --module-name trainer.task --package-path trainer --region $REGION \
   --scale-tier BASIC
   --train-files $TRAIN_DATA --eval-files $EVAL_DATA --num-epoch 1000 --learning-rate 0.01
```

single worker

GCS locations

model-specific params

https://cloud.google.com/ml-engine/docs/tensorflow/machine-types

Training in the cloud at scale

with multiple workers

https://cloud.google.com/ml-engine/docs/tensorflow/machine-types

```
gcloud ml-engine jobs submit training $JOB_NAME --job-dir $OUTPUT_PATH \
    --runtime-version 1.10 --module-name trainer.task --package-path trainer --region $REGION \
    --scale-tier STANDARD_1
    -- \
    --train-files $TRAIN_DATA --eval-files $EVAL_DATA
```

distributed

Google Cloud

Manually Distributing the Training

```
with tf.device("/job:ps/task:0"):
  weights_1 = tf.Variable(...)
  biases_1 = tf.Variable(...)

with tf.device("/job:ps/task:1"):
  weights_2 = tf.Variable(...)
  biases_2 = tf.Variable(...)

with tf.device("/job:worker/task:7"):
  input, labels = ...
  layer_1 = tf.nn.relu(tf.matmul(input, weights_1) + biases_1)
  logits = tf.nn.relu(tf.matmul(layer_1, weights_2) + biases_2)
  # ...
  train_op = ...

with tf.Session("grpc://worker7.example.com:2222") as sess:
  for _ in range(10000):
    sess.run(train_op)
```

tf.train.ClusterSpec construction

```
tf.train.ClusterSpec({"local": ["localhost:2222", "localhost:2223"]
```

```
tf.train.ClusterSpec({
    "worker": [
        "worker0.example.com:2222",
        "worker1.example.com:2222",
        "worker2.example.com:2222"
    ],
    "ps": [
        "ps0.example.com:2222",
        "ps1.example.com:2222"
    ]})
```

Training in the cloud at scale

with GPUs (K80/P100/V100 - *availability by region*)

https://cloud.google.com/ml-engine/docs/tensorflow/machine-types

```
gcloud ml-engine jobs submit training $JOB_NAME --job-dir $OUTPUT_PATH \
    --runtime-version 1.10 --module-name trainer.task --package-path trainer --region $REGION \
    --scale-tier BASIC_GPU
    -- \
    --train-files $TRAIN_DATA --eval-files $EVAL_DATA
```

single GPU

Google Cloud

CMLE Machine Types

Cloud ML Engine scale tier	
`BASIC`	A single worker instance. This tier is suitable for learning how to use Cloud ML Engine and for experimenting with new models using small datasets. **Compute Engine machine name:** n1-standard-4
`STANDARD_1`	One master instance, plus four workers and three parameter servers. **Compute Engine machine name, master:** n1-highcpu-8, **workers:** n1-highcpu-8, **parameter servers:** n1-standard-4
`PREMIUM_1`	One master instance, plus 19 workers and 11 parameter servers. **Compute Engine machine name, master:** n1-highcpu-16, **workers:** n1-highcpu-16, **parameter servers:** n1-highmem-8
`BASIC_GPU`	A single worker instance with a single NVIDIA Tesla K80 GPU. To learn more about graphics processing units (GPUs), see the section on training with GPUs. **Compute Engine machine name:** n1-standard-8 with one k80 GPU
`BASIC_TPU` *(Beta)*	A master VM and a Cloud TPU. See how to use TPUs for your training job. **Compute Engine machine name, master:** n1-standard-4, **workers:** Cloud TPU
`CUSTOM`	The CUSTOM tier is not a set tier, but rather enables you to use your own cluster specification. When you use this tier, set values to configure your processing cluster according to these guidelines:

Google

CMLE Custom Options

Cloud ML Engine machine name	
`standard`	A basic machine configuration suitable for training simple models with small to moderate datasets. **Compute Engine machine name:** n1-standard-4
`large_model`	A machine with a lot of memory, specially suited for parameter servers when your model is large (having many hidden layers or layers with very large numbers of nodes). **Compute Engine machine name:** n1-highmem-8
`complex_model_s`	A machine suitable for the master and workers of the cluster when your model requires more computation than the standard machine can handle satisfactorily. **Compute Engine machine name:** n1-highcpu-8
`complex_model_m`	A machine with roughly twice the number of cores and roughly double the memory of complex_model_s. **Compute Engine machine name:** n1-highcpu-16
`complex_model_l`	A machine with roughly twice the number of cores and roughly double the memory of complex_model_m. **Compute Engine machine name:** n1-highcpu-32
`standard_gpu`	A machine equivalent to standard that also includes a single NVIDIA Tesla K80 GPU. **Compute Engine machine name:** n1-standard-8 with one k80 GPU
`complex_model_m_gpu`	A machine equivalent to complex_model_m that also includes four NVIDIA Tesla K80 GPUs. **Compute Engine machine name:** n1-standard-16-k80x4

Google Cloud

Cloud TPUs

Cloud TPUs

- Excellent performance / $
- Train in days instead of weeks
- No more fighting with drivers
- Flexibility and scale
- Fully-managed in the cloud

Google Cloud

https://github.com/tensorflow/tpu-demos

Supported models for TPUs

Image recognition & object detection

Image recognition:
AmoebaNet-D
ResNet-50/101/152/200
Inception v2/v3/v4
DenseNet

Object detection:
RetinaNet

Low-resource models:
MobileNet
SqueezeNet

Machine translation and language modeling

Models:
Machine translation
Language modeling
Sentiment analysis
Question-answering
(all transformer-based)

Speech recognition

Model:
ASR Transformer
(LibriSpeech)

Image generation

Models:
Image Transformer
DCGAN

Google Cloud

Training Scikit-learn & XGBoost models

Key parameters to scikit-learn and XGBoost training on CMLE

```
gcloud ml-engine jobs submit training $JOB_NAME --job-dir $OUTPUT_PATH \
   --runtime-version 1.9 --python-version 2.7 --scale-tier BASIC \
   --module-name sklearn_trainer.task --package-path sklearn_trainer --region $REGION \
   -- \
   --train-files $TRAIN_DATA --eval-files $EVAL_DATA --num-epoch 1000 --learning-rate 0.01
```

Google Cloud

https://cloud.google.com/ml-engine/docs/scikit/getting-started-training

XGBoost

scikit learn

Keras: what is happening?

- Compatibility module introduced in TensorFlow: tf.keras
- Write your custom estimator model_fn using tf.keras.layers and/or tf.layers (mix-and-match)
- Convert your compiled keras model to tf.estimator using tf.keras.estimator.model_to_estimator
- With TensorFlow, Keras users gain access to new features:
    - Distributed training
    - Multiple GPUs
    - Cloud ML
    - Hyperparameter tuning
    - TF-Serving

Keras example

```
y_train = ...
y_test = ...

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
             optimizer=keras.optimizers.Adadelta(), metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
         validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
```

Google Cloud

Keras

Premade Estimators

TensorFlow

- LinearClassifier
- DNNClassifier
- DNNLinearCombinedClassifier
- DNNLinearCombinedRegressor
- (tf.contrib.kernel_methods.) KernelLinearClassifier
- (tf.contrib.factorization.) KMeansClustering
- (tf.contrib.timeseries.) ARRegressor
- BoostedTreesClassifier
- BoostedTreesRegressor

- LinearRegressor
- DNNRegressor
- (tf.contrib.estimator)RNNClassifier
- (tf.contrib.estimator)RNNEstimator

Google Cloud

Hyperparameter tuning

```
gcloud ml-engine jobs submit training $JOB_NAME --job-dir $OUTPUT_PATH \
   --runtime-version 1.7 --module-name trainer.task --package-path trainer/ --region $REGION \
   --scale-tier PREMIUM_1 --config hyperparams.yaml
   -- \
   --train-files $TRAIN_DATA --eval-files $EVAL_DATA
```

hypertuning

Google Cloud

Hyperparameter tuning

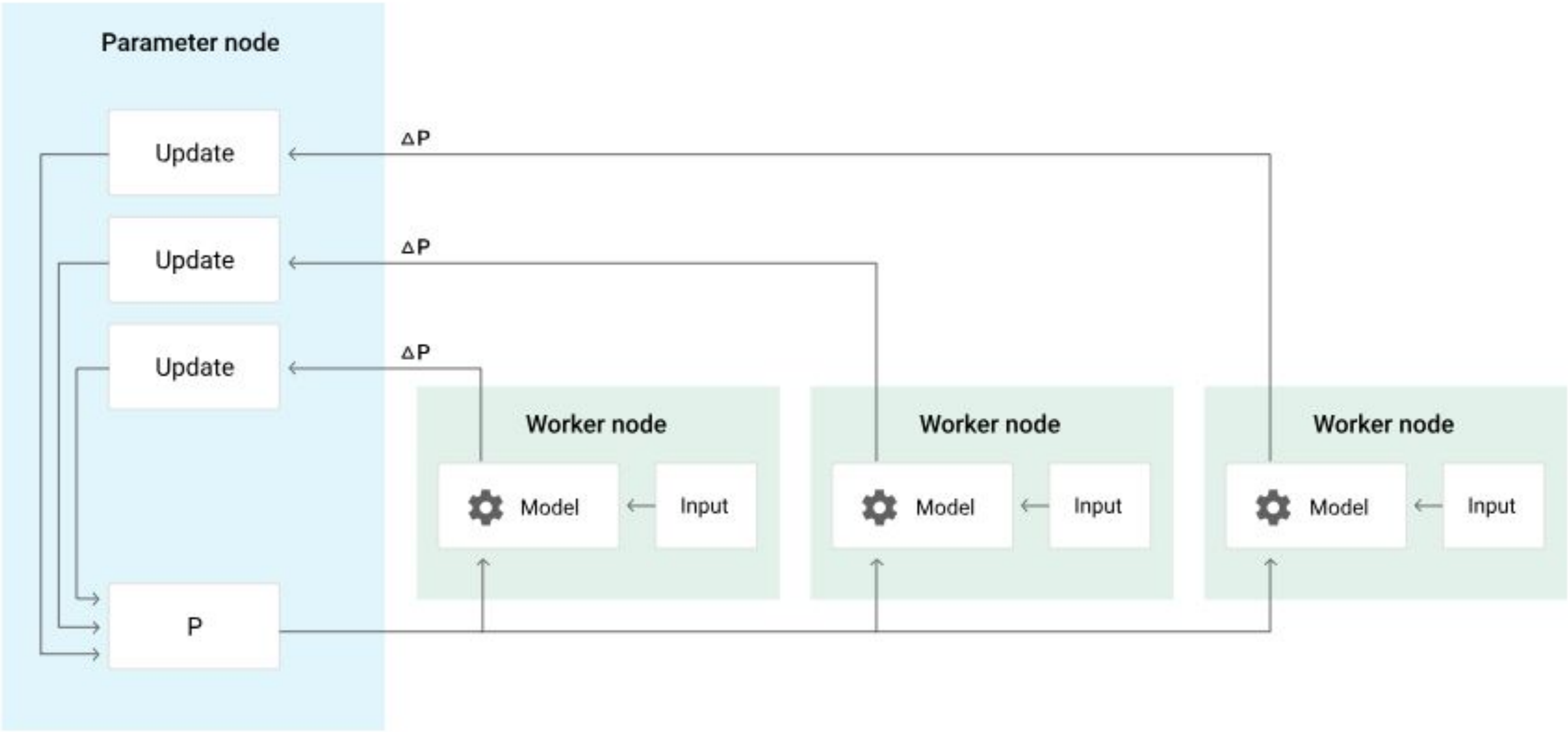
hyperparams.yaml

```yaml
trainingInput:
 hyperparameters:
   goal: MAXIMIZE
   hyperparameterMetricTag: accuracy
   maxTrials: 40
   enableTrialEarlyStopping: True
   maxParallelTrials: 2
   algorithm: UNSPECIFIED
   params:
     - parameterName: learning-rate
       type: FLOAT
       minValue: 0.001
       maxValue: 0.1
       scaleType: UNIT_LOG_SCALE

...
```

task.py

```python
...

      # Initialise the optimizer for the DNN
      optimizer = tf.train.AdagradOptimizer(
            learning_rate=hparams.learning_rate)

...

parser.add_argument(
     '--learning-rate',
     help='Learning rate used by the DNN optimizer',
     default=0.01,
     type=float
 )

...
```

Distributed Training



Consuming the deployed ML model API for predictions

Deploy the trained TF model

gcloud command line tool:

```
# Creating model
NAME=demo_classifier
gcloud ml-engine models create $NAME --regions $REGION

# Creating versions\
VERSION=v2.3
MODEL_DIR=gs://ksalama-gcs/trained_models/demo_classifier_output
gcloud ml-engine versions create $VERSION --model $NAME --origin $MODEL_DIR \
  --runtime-version 1.7 --config config.yaml
```

```
description: A free-form description of the version.
 deploymentUri: gs://path/to/source
 runtimeVersion: '1.7'
 manualScaling:
   nodes: 10
 autoScaling:
   minNodes: 0
```

```
# List deployed models
gcloud ml-engine models list
```

Google Cloud

Predicting with TF Model

Online versus Batch Prediction

SEND FEEDBACK

Internal: Count: 1, Average: 5.0

Cloud ML Engine provides two ways to get predictions from trained models: *online prediction* (sometimes called HTTP prediction), and *batch prediction*. In both cases, you pass input data to a cloud-hosted machine-learning model and get inferences for each data instance. The differences are shown in the following table:

Online prediction	Batch prediction
Optimized to minimize the latency of serving predictions.	Optimized to handle a high volume of instances in a job and to run more complex models.
Can process one or more instances per request.	Can process one or more instances per request.
Predictions returned in the response message.	Predictions written to output files in a Cloud Storage location that you specify.
Input data passed directly as a JSON string.	Input data passed indirectly as one or more URIs of files in Cloud Storage locations.
Returns as soon as possible.	Asynchronous request.
Accounts with the following IAM roles can request online predictions: • Legacy Editor or Viewer • Cloud ML Engine Admin or Developer	Accounts with the following IAM roles can request batch predictions: • Legacy Editor • Cloud ML Engine Admin or Developer
Runs on the runtime version and in the region selected when you deploy the model.	Can run in any available region, using any available runtime version. Though you should run with the defaults for deployed model versions.

Google Cloud

Predicting with TF Model

gcloud command line tool:

```
gcloud ml-engine predict --model $NAME --version $VERSION --json-instances test.json
```

gcloud **batch prediction**:

```
gcloud ml-engine job submit prediction \

$JOB_NAME --model $NAME --version $VERSION \

data-format TEXT \

input-paths $GCS_DATA_DIR \

output-path $GCS_OUT_DIR \
```

python code - REST API call - **online prediction**:

```
def estimate(project, model_name, version, instances):

    credentials = GoogleCredentials.get_application_default()
    api = discovery.build('ml', 'v1',
          credentials=credentials,
          discoveryServiceUrl =
          'https://storage.googleapis.com/cloud-ml/discovery/ml_v1_discovery.json')

    request_data = {'instances': instances}

    model_url = 'projects/{}/models/{}/versions/{}'.format(
                 project,model_name, version)

    response = api.projects().predict(body=request_data, name=model_url).execute()

    estimates = list(map(lambda item: item["scores"]
        ,response["predictions"]
    ))

    return estimates
```

Google Cloud

Deploy XGBoost & Scikit-learn models

Save and dump model to GCS

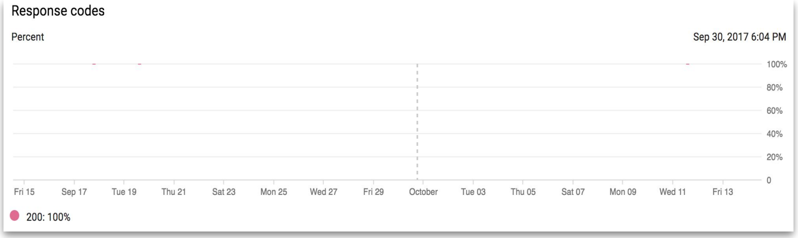
```
from sklearn.externals import joblib
joblib.dump(estimator, model.joblib)

gsutil cp ./model.joblib ${MODEL_PATH}/model.joblib
```

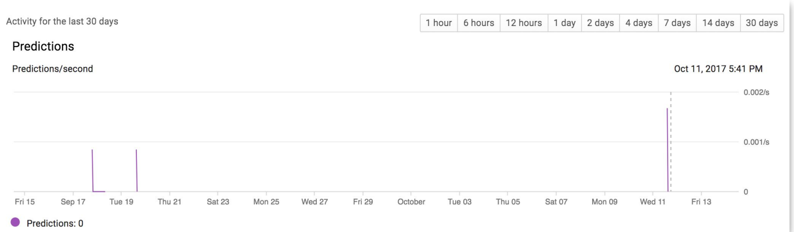
deploy the model to CMLE

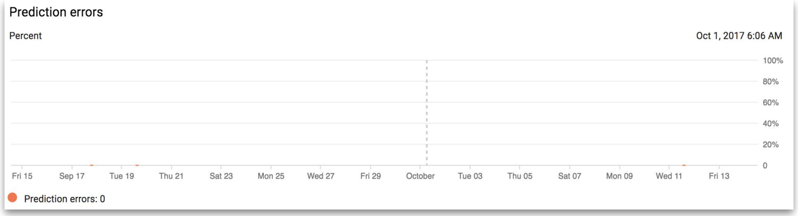
```
gcloud ml-engine models create ${MODEL_NAME} --regions=${REGION}

gcloud ml-engine versions create ${VERSION} --model=${MODEL_NAME} \
    --origin=${MODEL_PATH} \
    --runtime-version="1.4" \
    --framework="SCIKIT_LEARN"
    --pythonVersion="2.7"
```

https://cloud.google.com/blog/big-data/2018/04/serving-real-time-scikit-learn-and-xgboost-predictions

Monitoring model API serving health

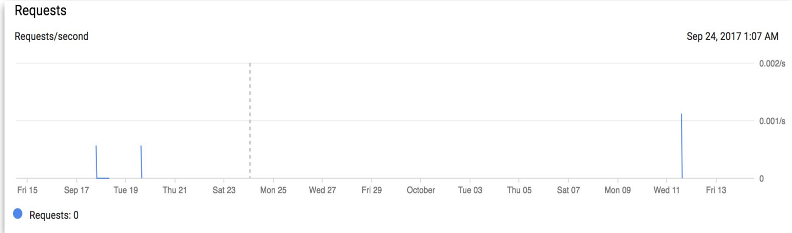


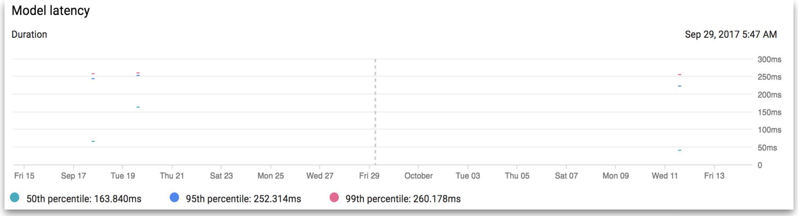




v1

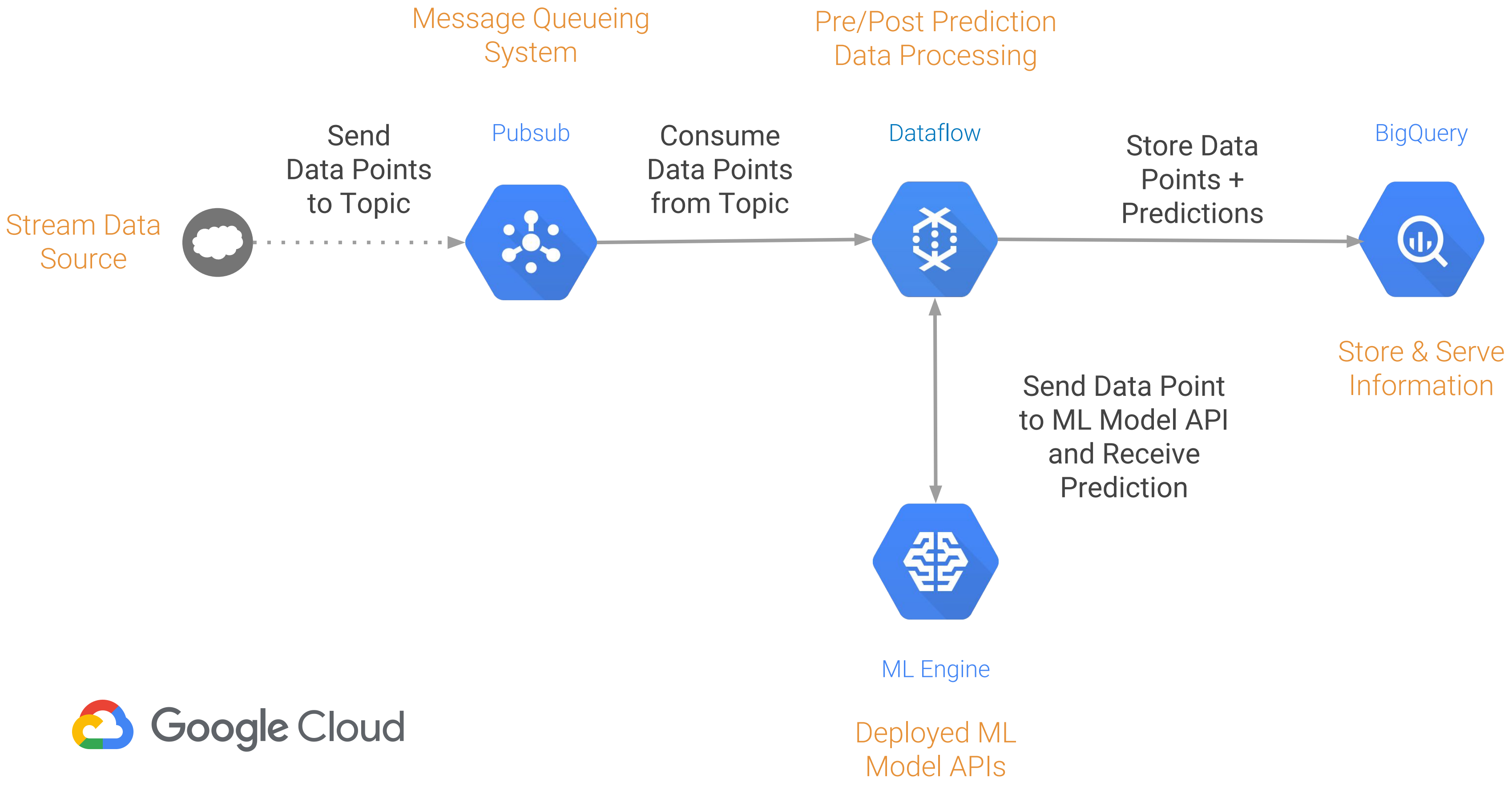
Description	
Model	taxifare_estimator
Model location	gs://ksalama-gcs-cloudml/ml-models/taxifare/dnn-combined-regression-small/
Creation time	Aug 22, 2017, 11:52:04 PM
Last use time	Oct 11, 2017, 1:20:01 PM





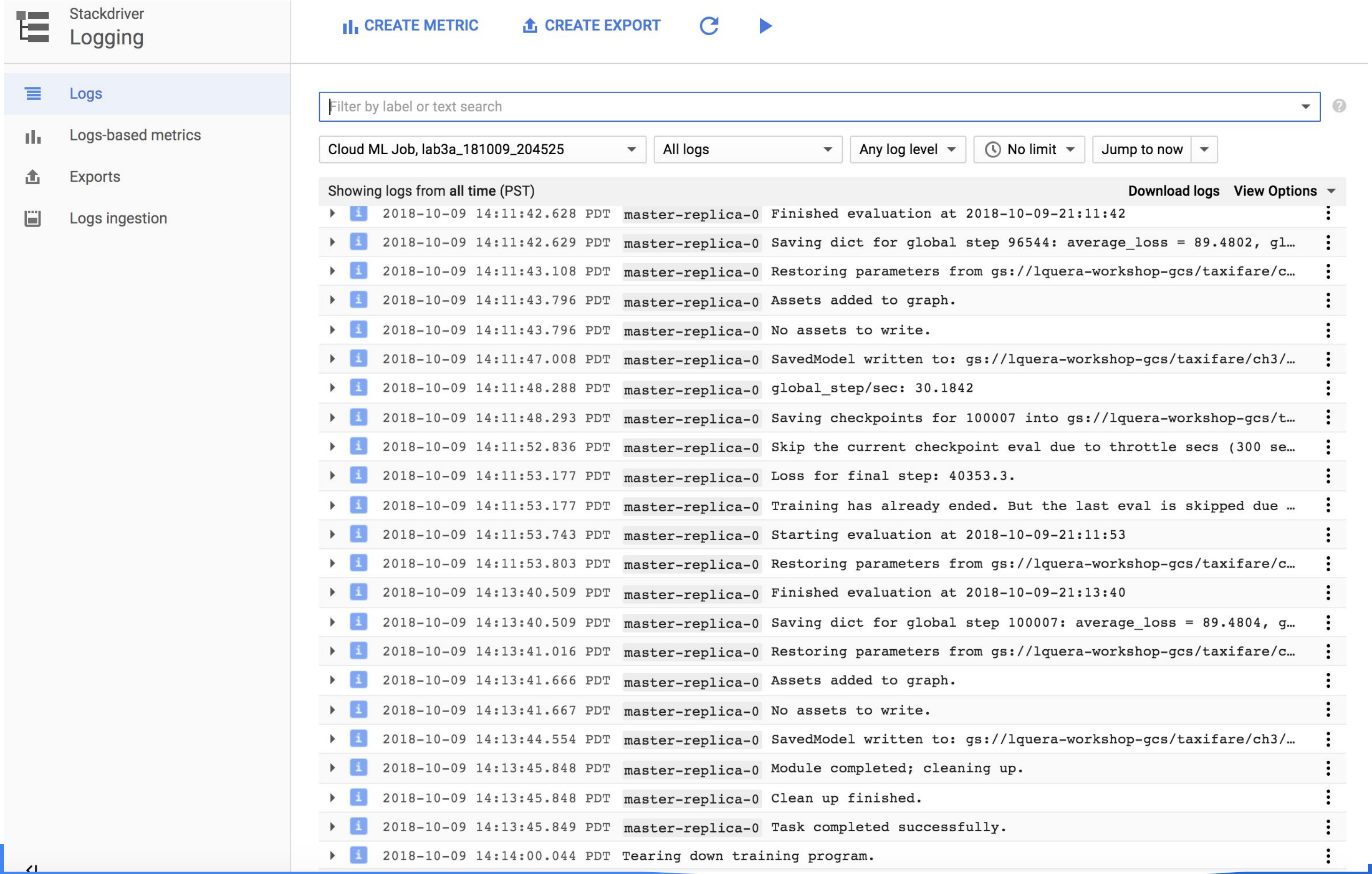
Google Cloud

Example Production Workflow

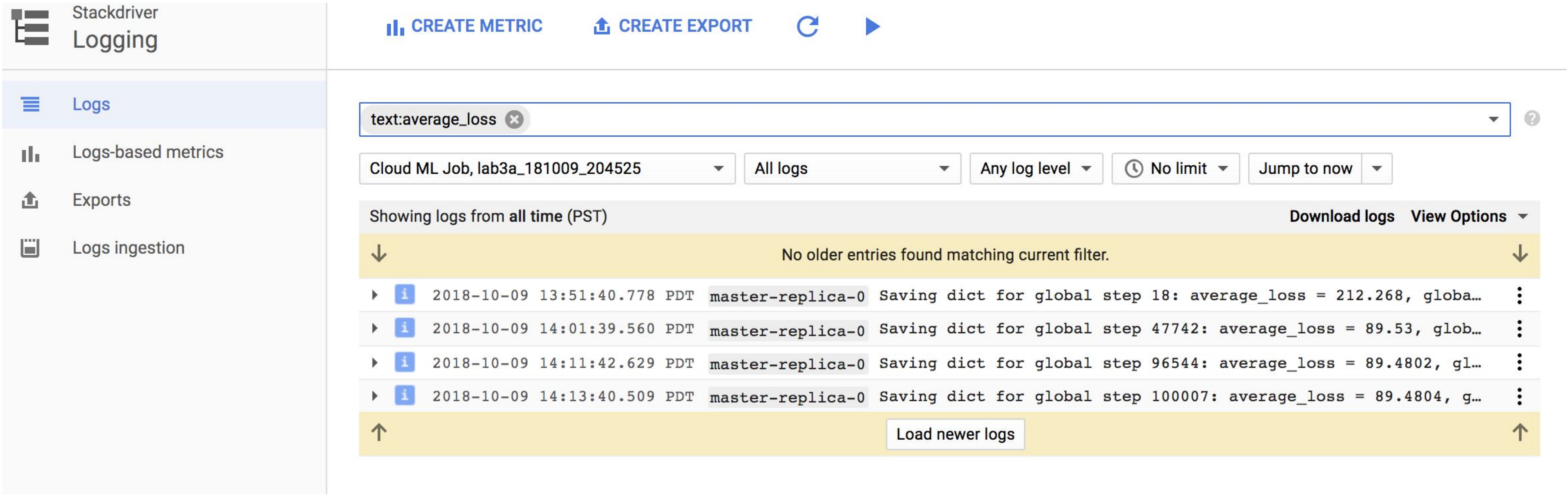


Google Cloud

Monitor ML Engine job on Cloud Console



Monitor ML Engine job on Cloud Console



Thank you!