Google Cloud

Machine Learning at Scale

TensorFlow in the Cloud

Yufeng Guo
Developer Advocate
@YufengG

Machine Learning
is
using many ***examples***
to ***answer questions***

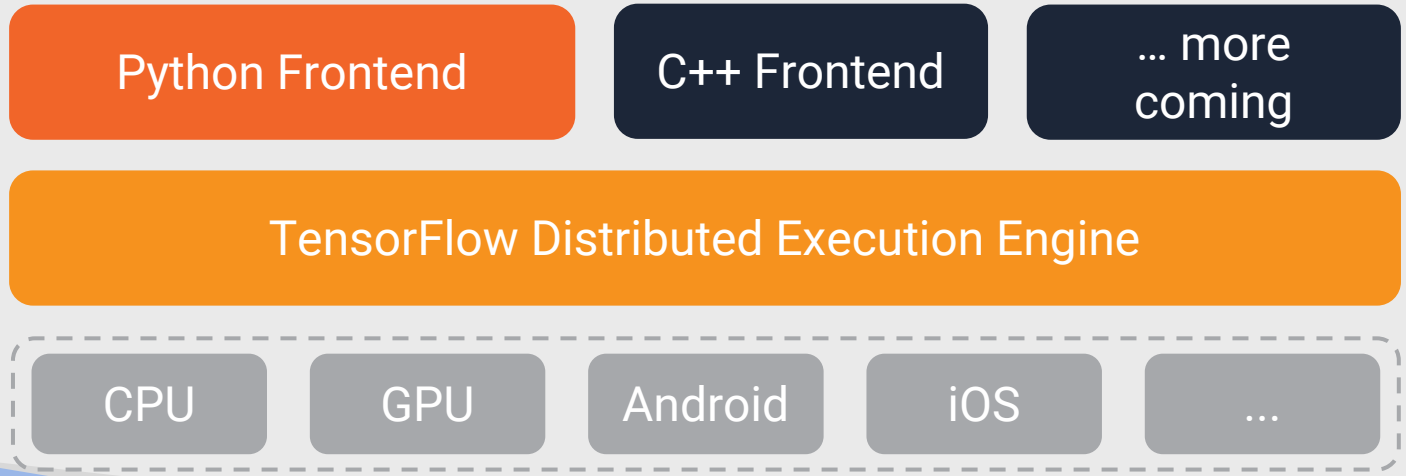
Training

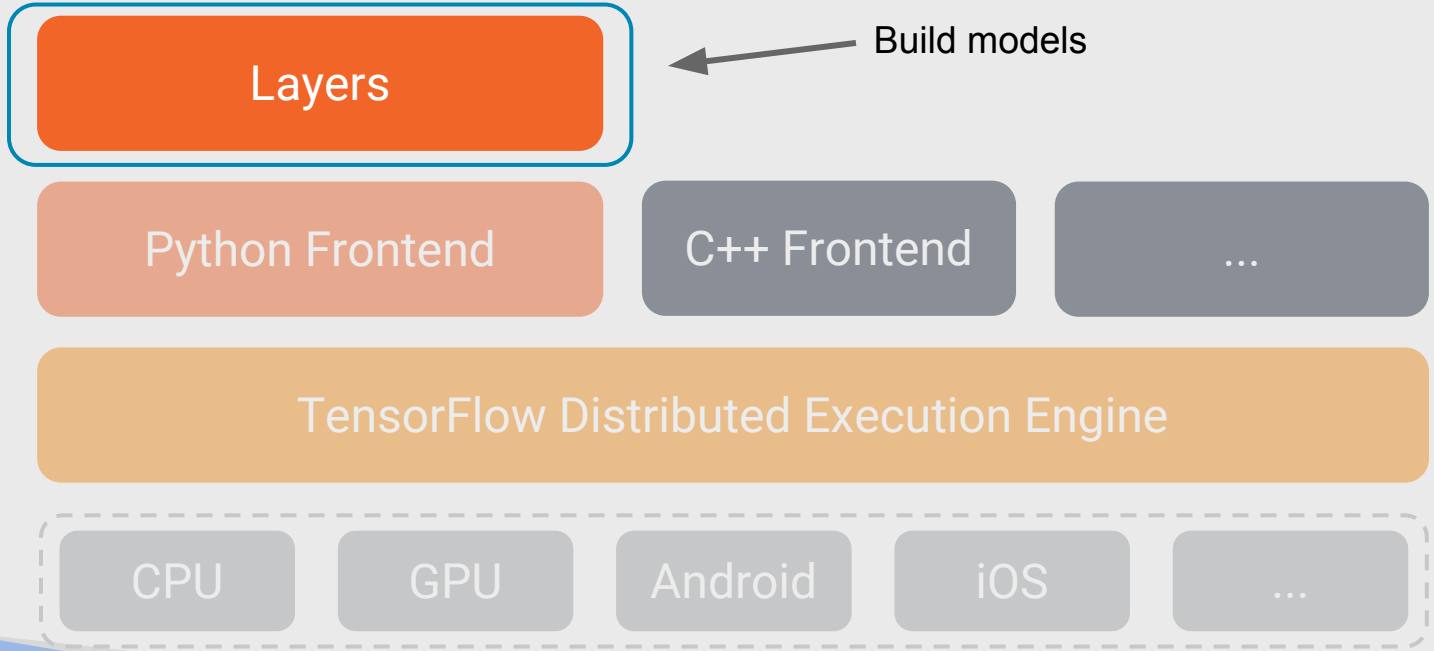
many

examples

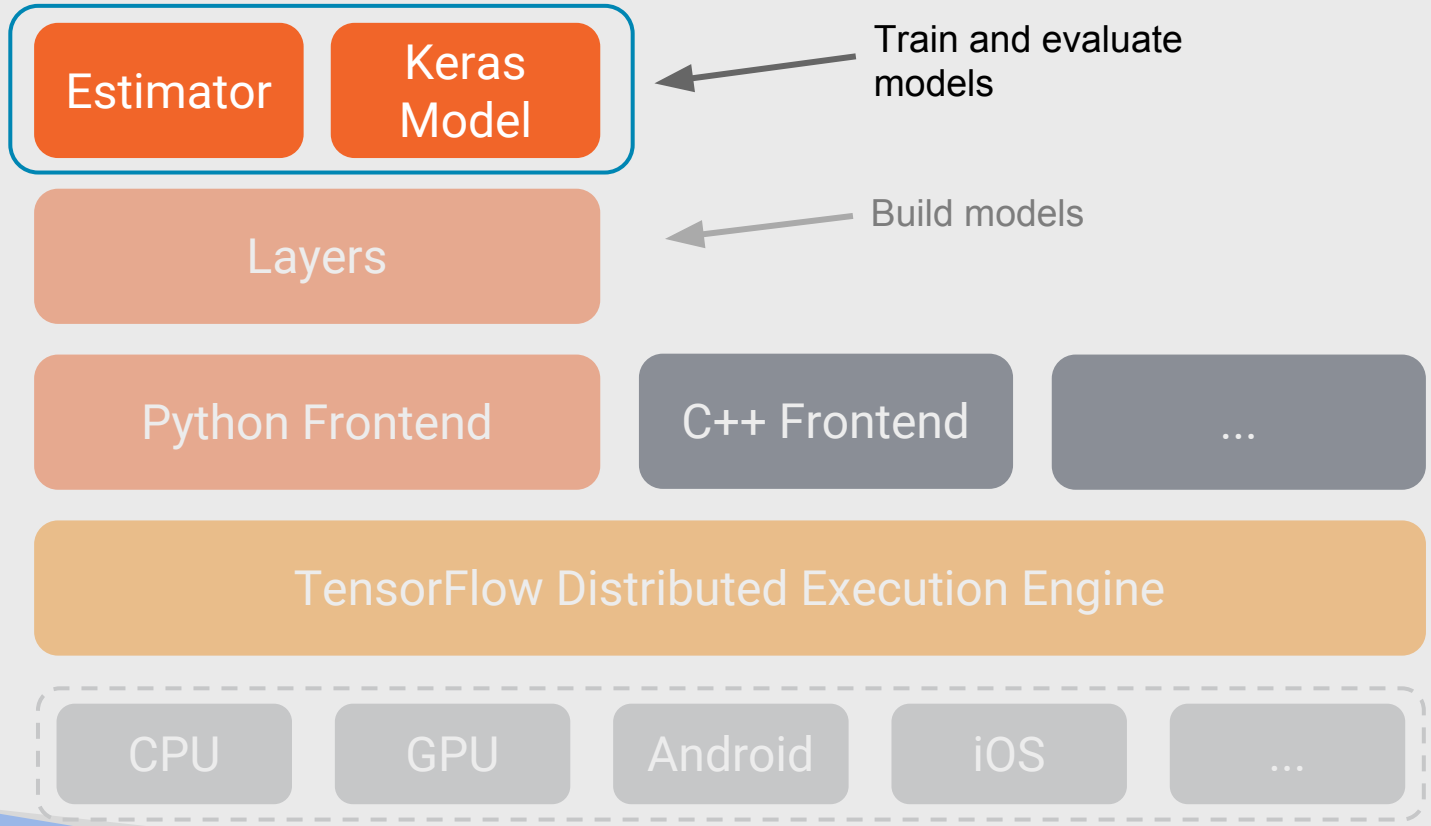
Prediction

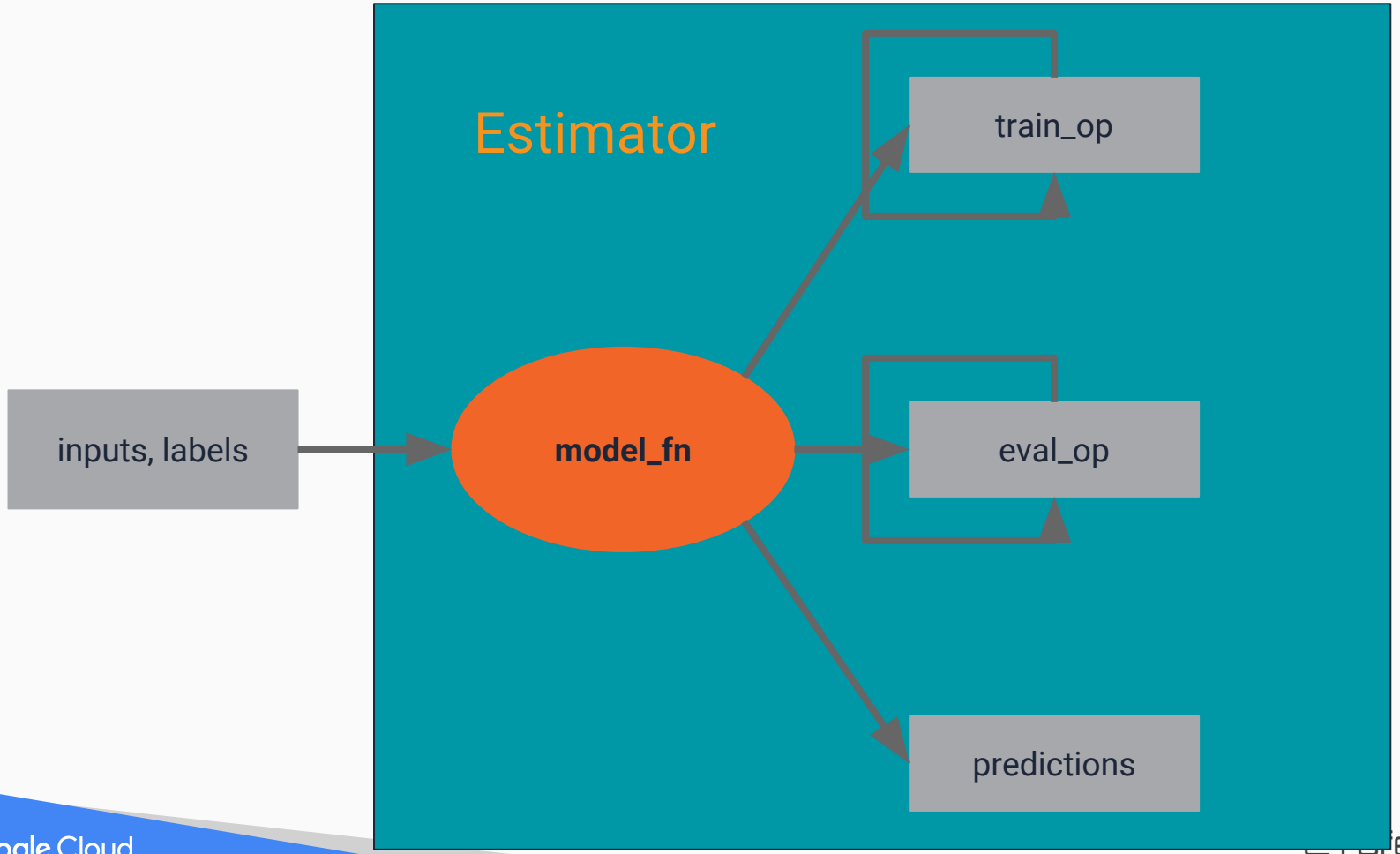
answer questions

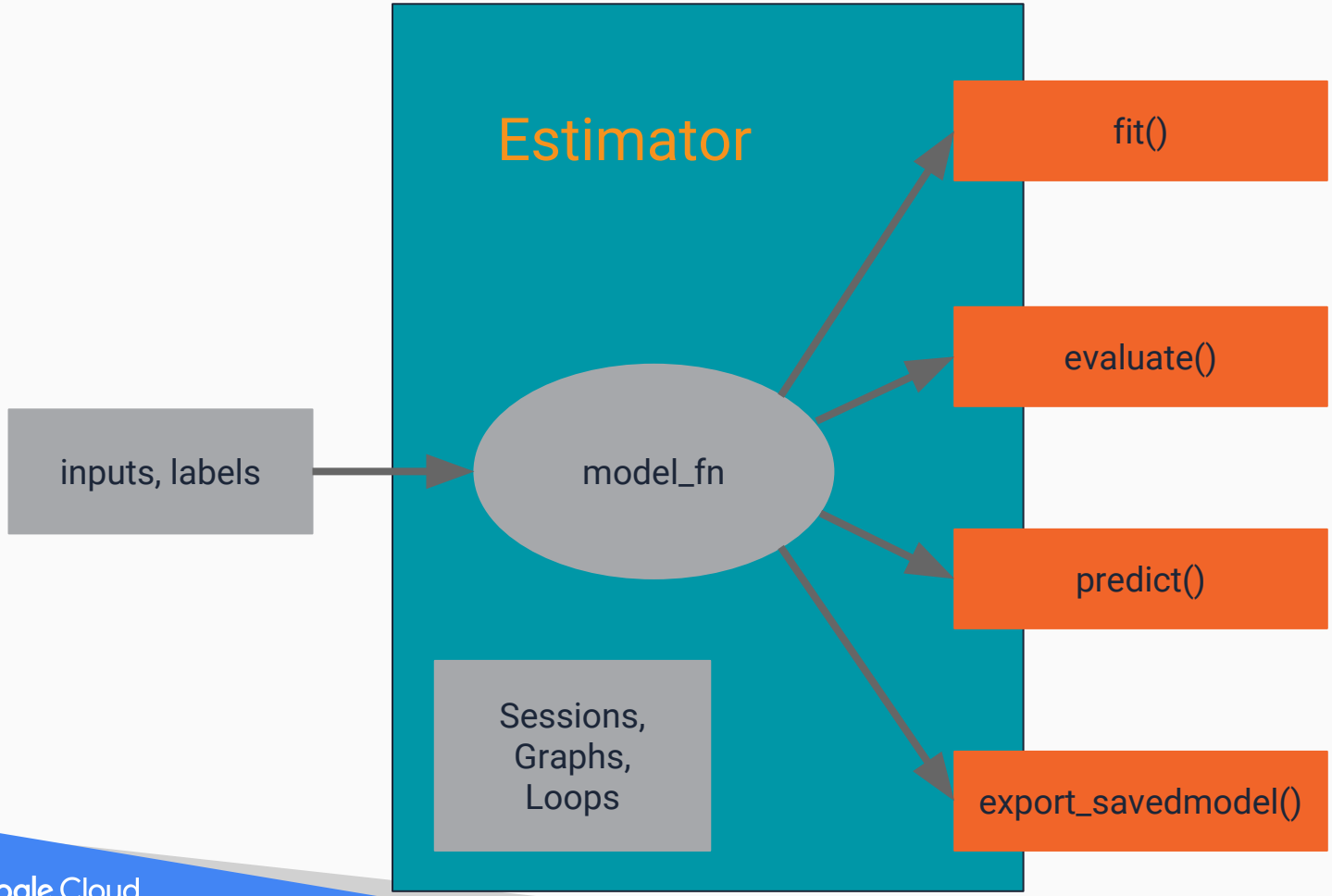

Python Frontend
C++ Frontend
… more coming
TensorFlow Distributed Execution Engine
CPU
GPU
Android
iOS
…

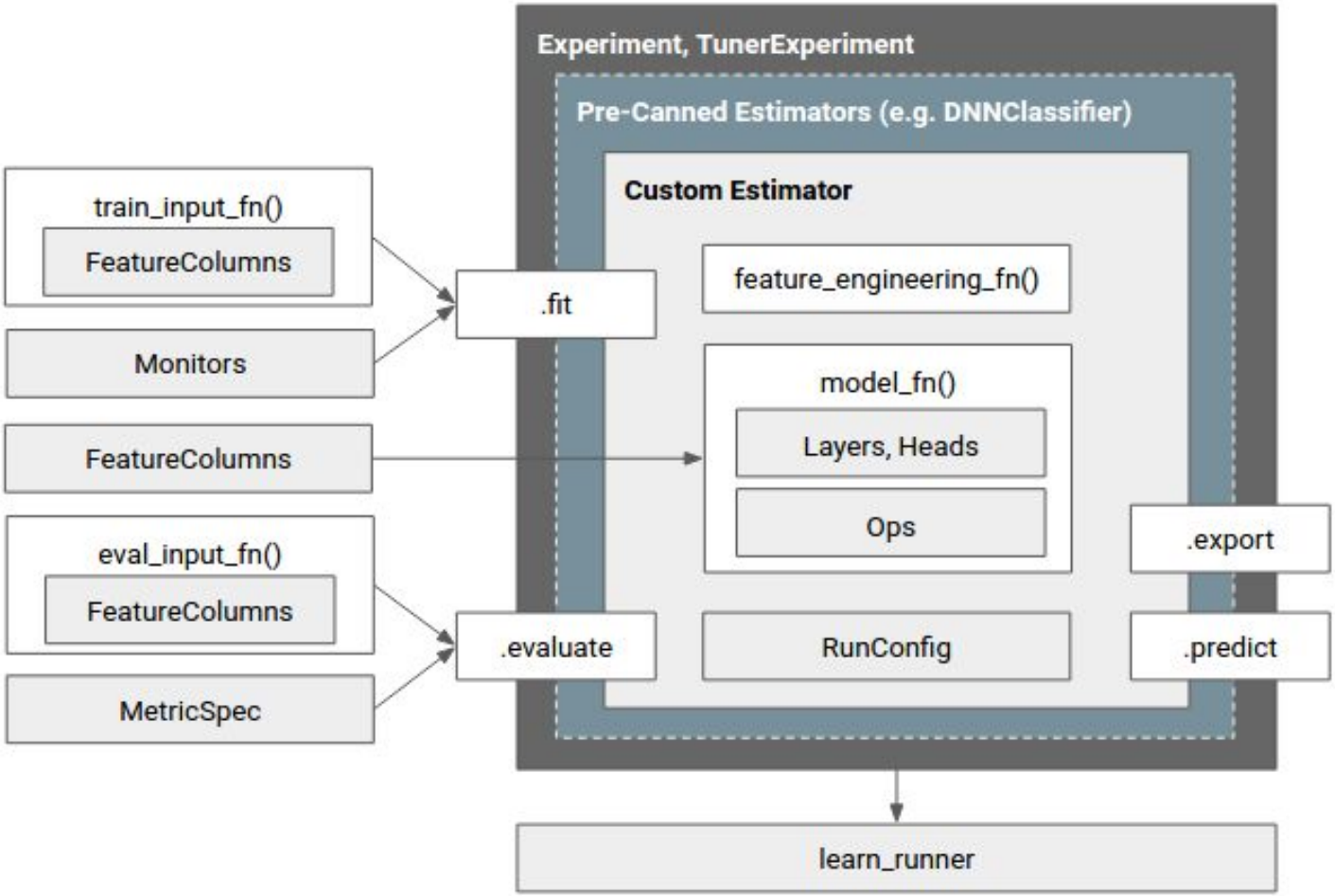

Layers
Build models
Python Frontend
C++ Frontend
...
TensorFlow Distributed Execution Engine
CPU
GPU
Android
iOS
...

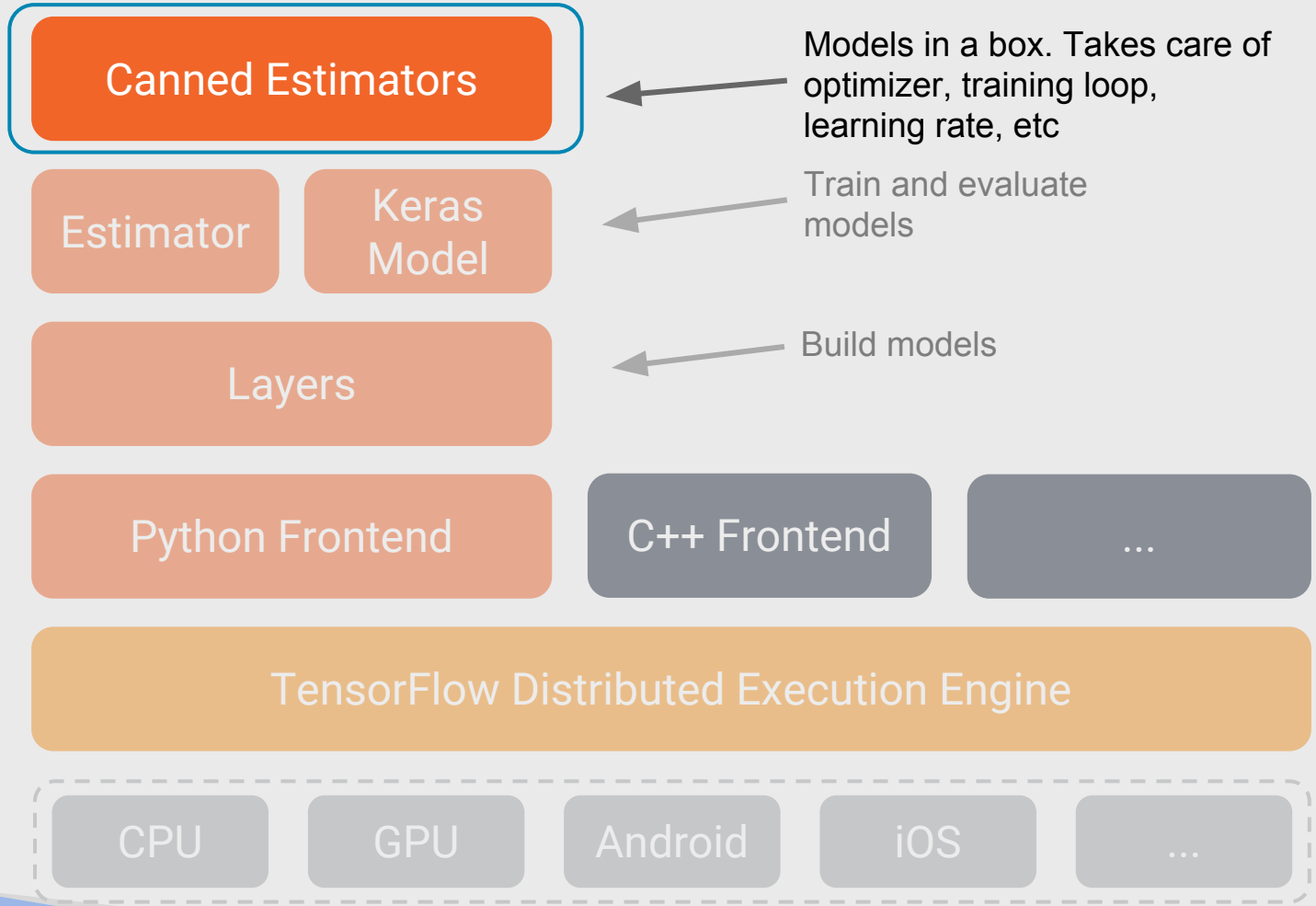

```
x = tf.layers.conv2d(x, kernel_size=[5,5], ...)
x = tf.layers.max_pooling2d(x, kernel_size=[2,2], ...)
x = tf.layers.conv2d(x, kernel_size=[5,5], ...)
x = tf.layers.max_pooling2d(x, kernel_size=[2,2], ...)
x = tf.layers.dense(x, activation_fn=tf.nn.relu)
x = tf.layers.dropout(x, 0.5)
x = tf.layers.dense(x)
```

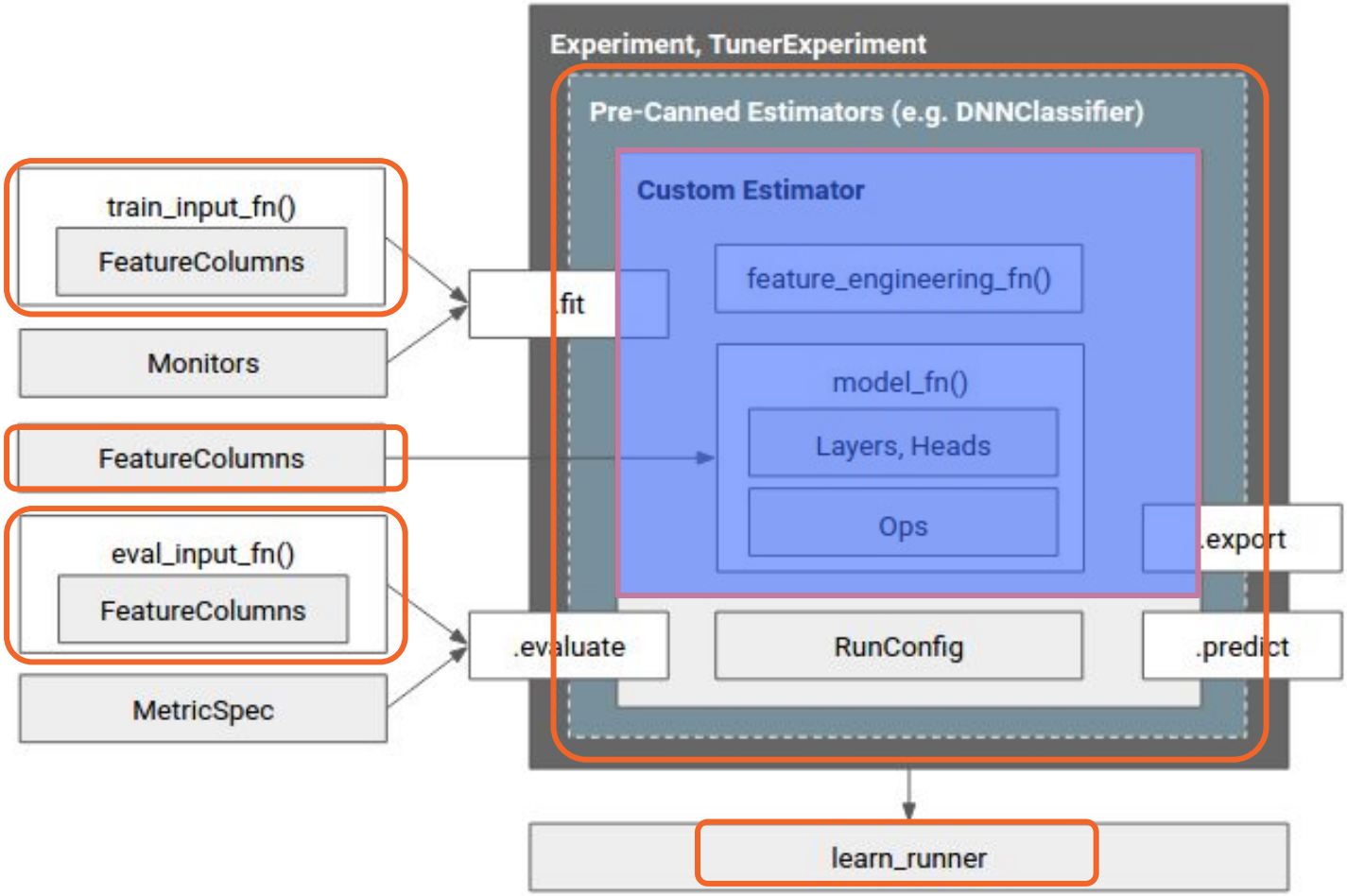

Estimator
Keras Model
Train and evaluate models
Layers
Build models
Python Frontend
C++ Frontend
...
TensorFlow Distributed Execution Engine
CPU
GPU
Android
iOS
...

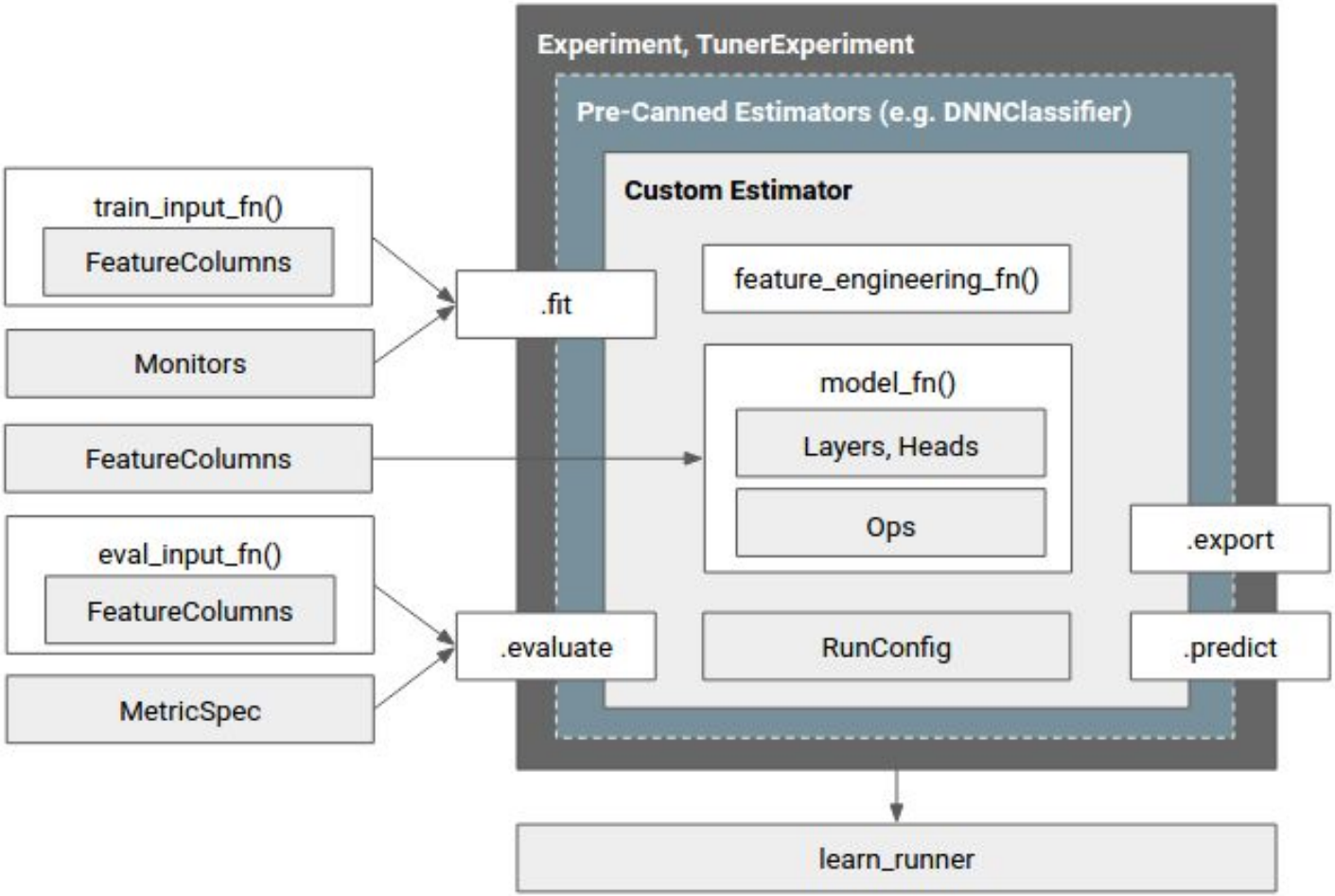

Estimator
train_op
inputs, labels
model_fn
eval_op
predictions


Estimator
fit()
evaluate()
predict()
export_savedmodel()
inputs, labels
model_fn
Sessions,
Graphs,
Loops


Experiment, TunerExperiment
Pre-Canned Estimators (e.g. DNNClassifier)
Custom Estimator
train_input_fn()
FeatureColumns
Monitors
FeatureColumns
eval_input_fn()
FeatureColumns
MetricSpec
.fit
.evaluate
feature_engineering_fn()
model_fn()
Layers, Heads
Ops
RunConfig
.export
.predict
learn_runner


Canned Estimators
Models in a box. Takes care of optimizer, training loop, learning rate, etc
Estimator
Keras Model
Train and evaluate models
Layers
Build models
Python Frontend
C++ Frontend
...
TensorFlow Distributed Execution Engine
CPU
GPU
Android
iOS
...


Experiment, TunerExperiment
Pre-Canned Estimators (e.g. DNNClassifier)
Custom Estimator
train_input_fn()
FeatureColumns
Monitors
FeatureColumns
eval_input_fn()
FeatureColumns
MetricSpec
.fit
feature_engineering_fn()
model_fn()
Layers, Heads
Ops
.export
.evaluate
RunConfig
.predict
learn_runner


Experiment, TunerExperiment
Pre-Canned Estimators (e.g. DNNClassifier)
Custom Estimator
train_input_fn()
FeatureColumns
Monitors
FeatureColumns
eval_input_fn()
FeatureColumns
MetricSpec
.fit
.evaluate
feature_engineering_fn()
model_fn()
Layers, Heads
Ops
RunConfig
.export
.predict
learn_runner

```
area = real_valued_column(“square_foot”),
rooms = real_valued_column(“num_rooms”),
zip_code = sparse_column_with_integerized_feature(“zip_code”, 10000)


classifier = DNNRegressor(
    feature_columns=[area, rooms, embedding_column(zip_code, 8)],
    hidden_units=[1024, 512, 256])

classifier.fit(train_input_fn)

classifier.evaluate(eval_input_fn)
```

<Storytime>

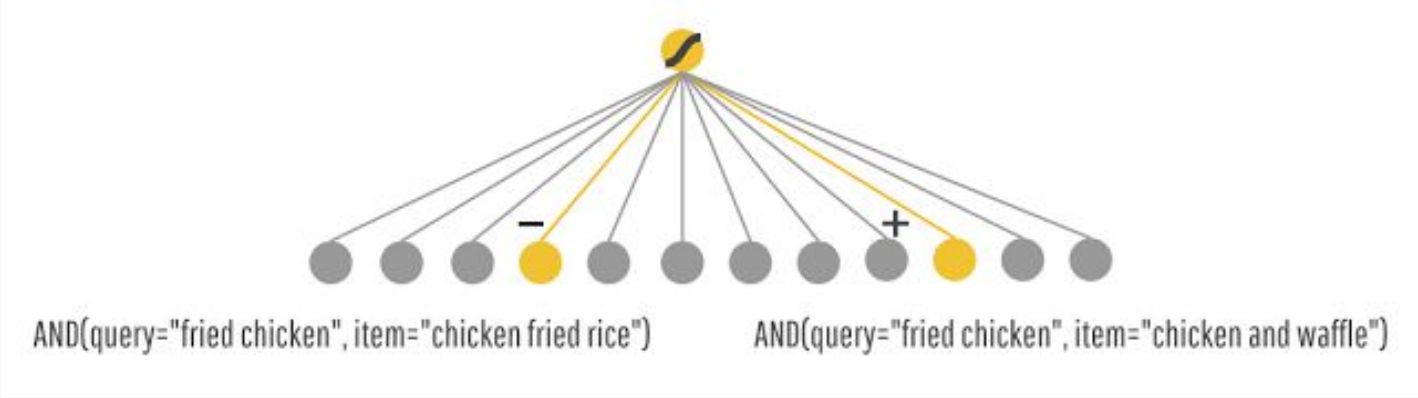
Motivation - a "magical" food app

Just **Launch** and Iterate

- Naive character matching
- Say "Fried chicken"
- Get "Chicken Fried Rice"
- Oops. Now what?
- Machine learning to the rescue!

v2.0: **memorize** all the things!

- Train a linear TF model



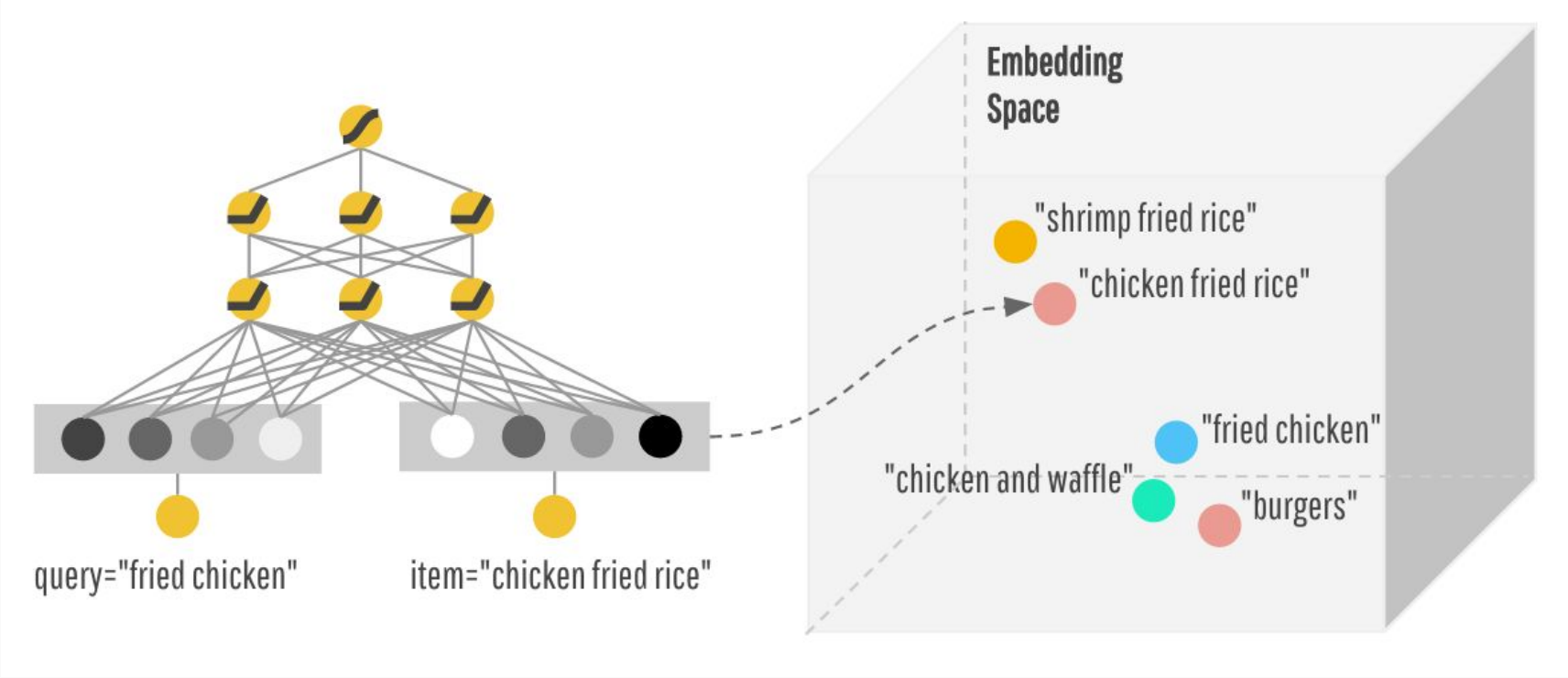
- Your app is gaining traction!

Problem: Your users are bored!

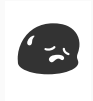
- Too many  & waffles
- Show me similar, but different food

- Your users are **picky** 

v3.0: More generalized recommendations for all



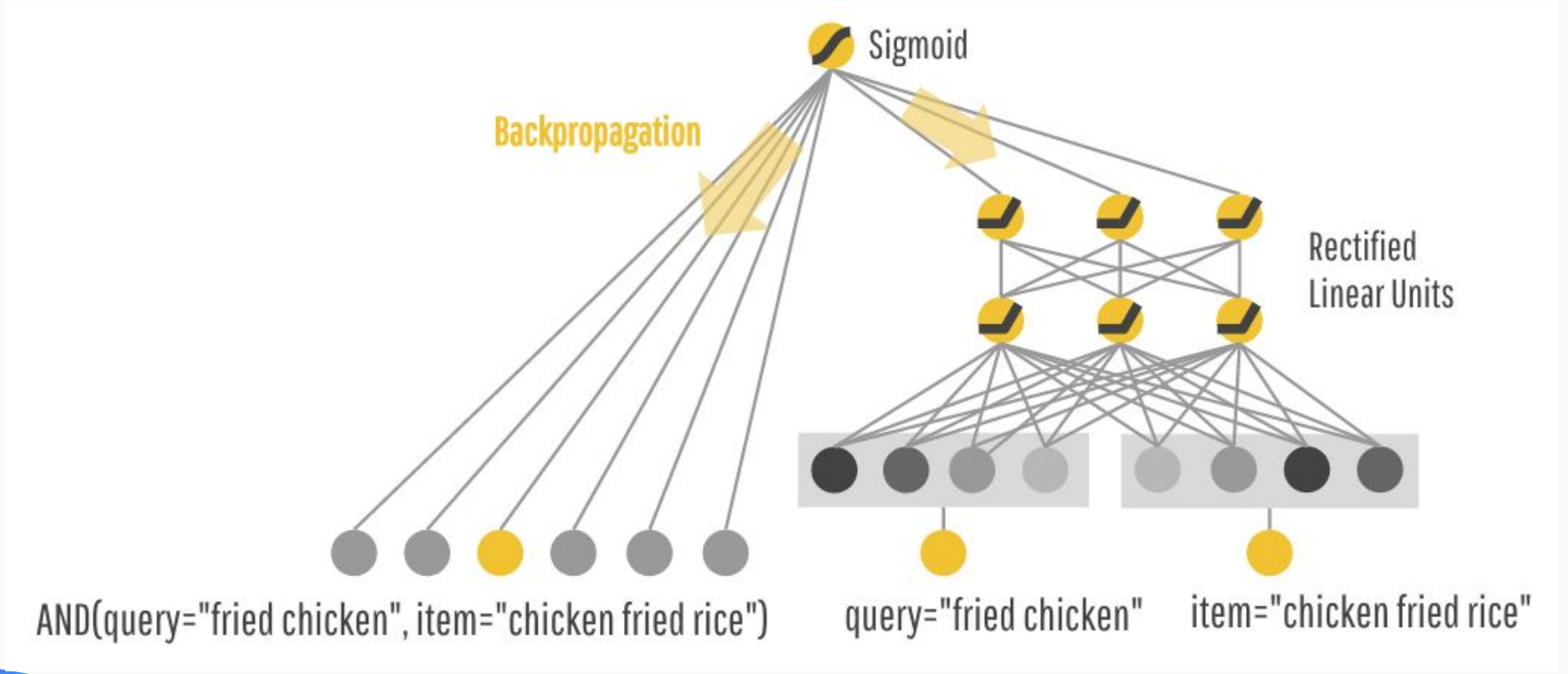
No good deed goes unpunished

- Some recommendations are "too general"
  - Irrelevant dishes are being sent
- Your users are **still picky** 

No good deed goes unpunished

- 2 types of requests: specific and general
- "iced decaf latte with nonfat milk" != "hot latte with whole milk"
- “seafood” or “italian food” or “fast food”
- **How to balance this?**

v4.0: Why not both?

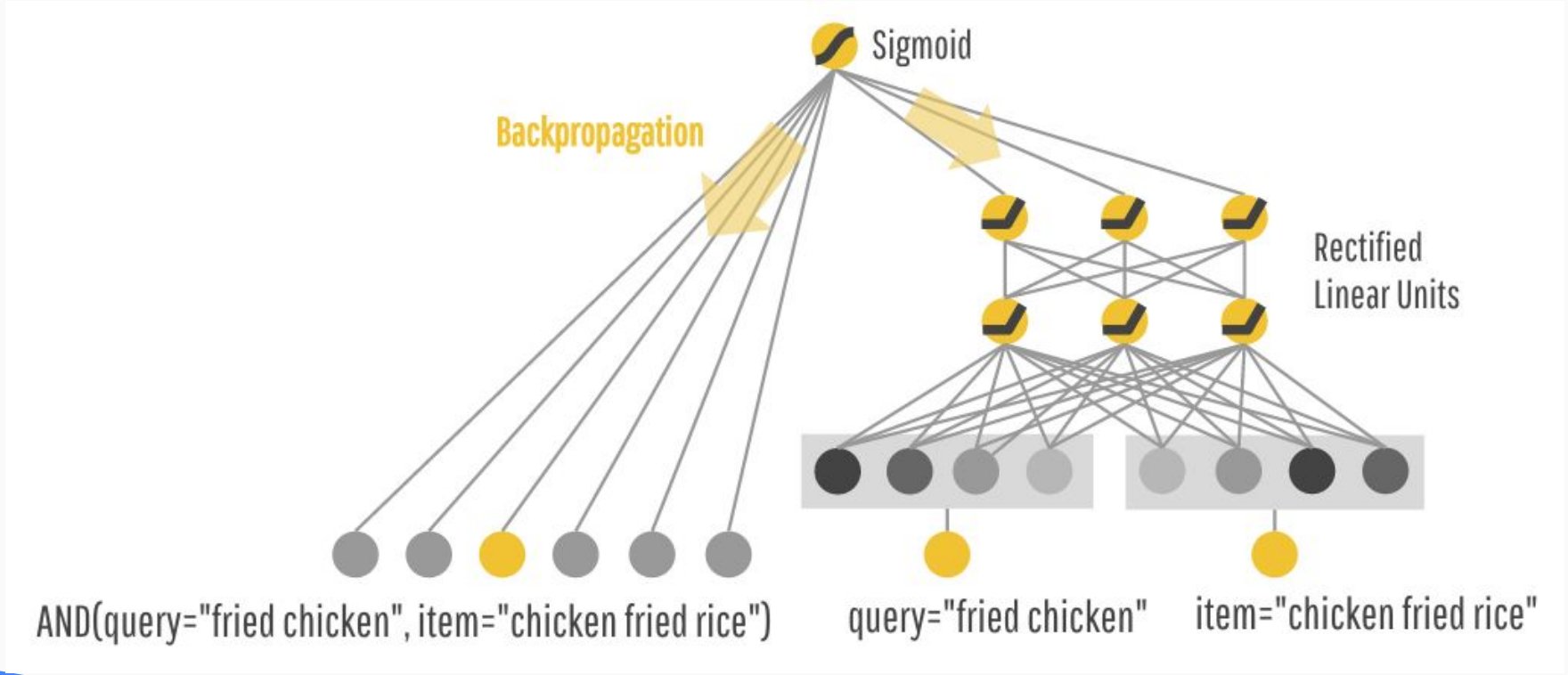


Wide & Deep

memorization
relevance

generalization
diversity

Wide and Deep



</Storytime>

Meet our dataset: Criteo click-data

- **Task**: predict the whether an ad was clicked on, based on **39 *anonymized* factors**
- Over **45 million** training examples, ~5GB
- Released to the public, and in a full, much larger form (~1TB) as well.

Meet our dataset: Criteo click-data

- **Features**: 13 integer columns, followed by 26 columns of 32bit hashed values
- **Labels:** Just a 0 or 1 :)

Training

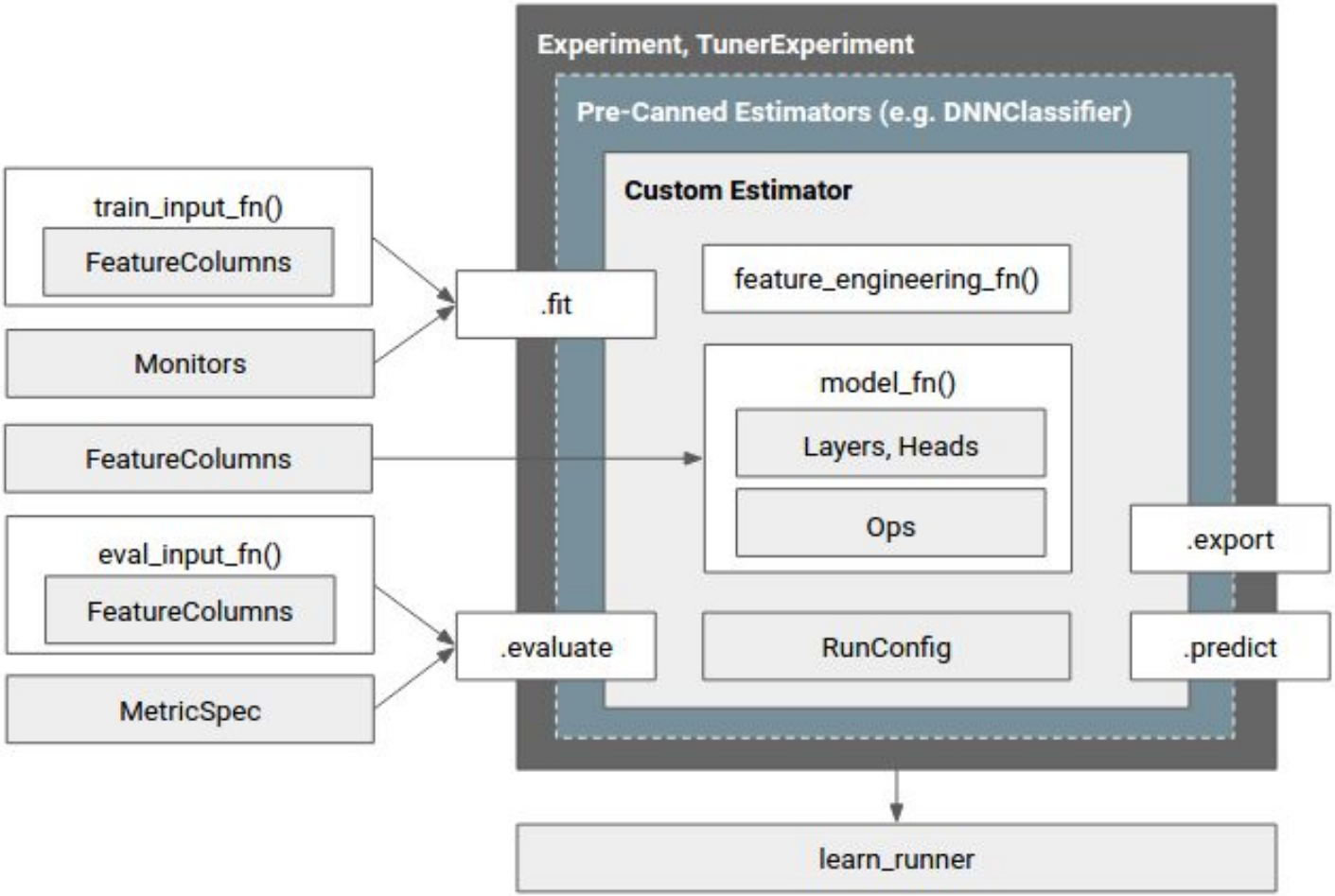
many

examples

Prediction

answer questions

Time to code! Almost...



```
git clone
https://github.com/yufengg/widendeep.git
```

or go to

bit.ly/widendeep-code

These slides

bit.ly/widendeep-slides

Dude where's my data?

bit.ly/widendeep-slides

Use **`gsutil cp`** `LINK.{train.csv|eval.csv}`

Where LINK is one of these:

gs://dataset-uploader/criteo-kaggle/small_version -- **2.5MB**, 10K rows

gs://dataset-uploader/criteo-kaggle/medium_version -- **273MB**, 1M rows

~~gs://dataset-uploader/criteo-kaggle/large_version~~ -- **2.7GB**, 10M rows

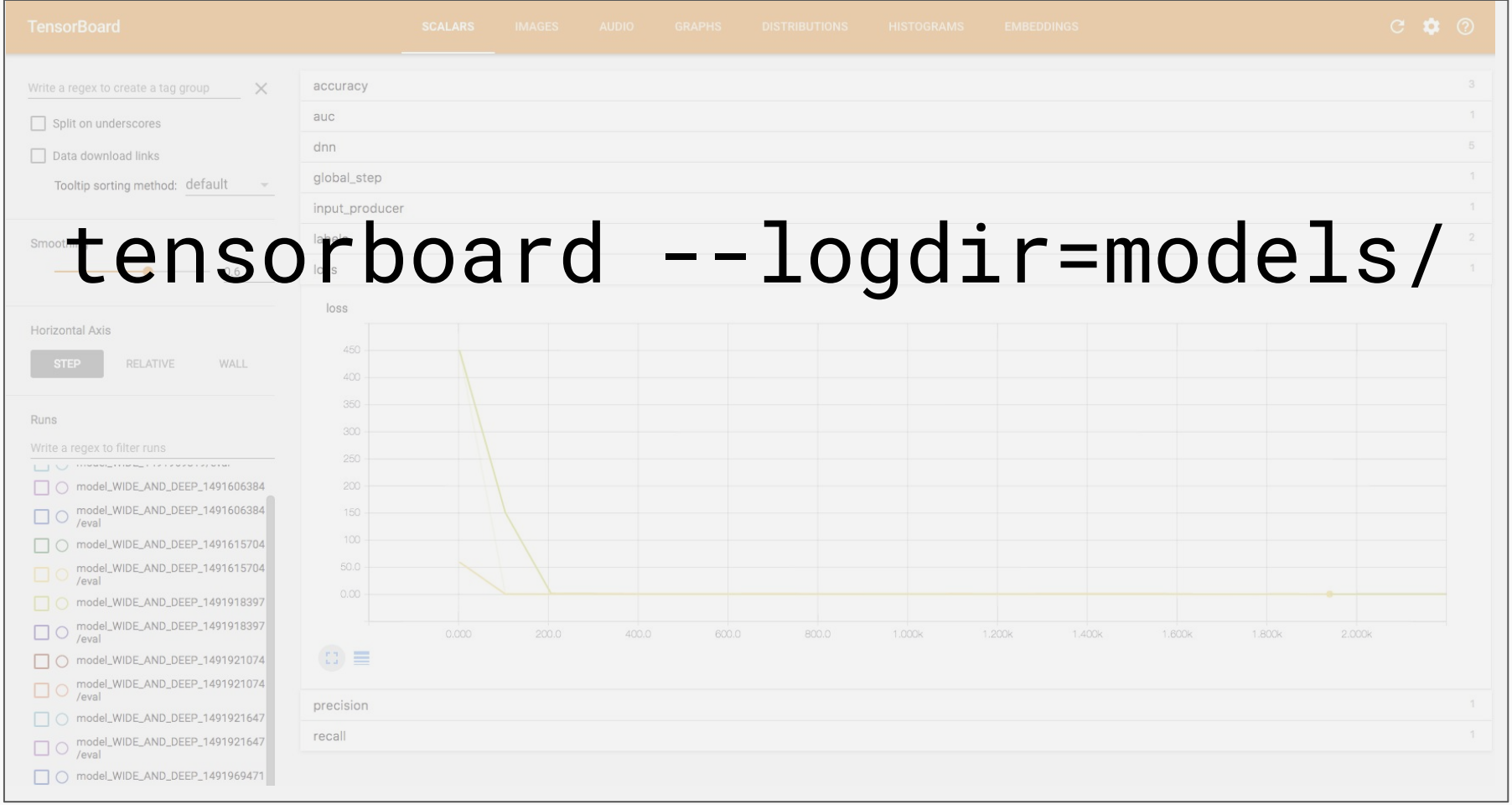
No gsutil? Replace "gs://" with 'https://storage.googleapis.com/' , for example:

https://storage.googleapis.com/dataset-uploader/criteo-kaggle/**medium_version**/train.csv

To the code!

https://github.com/yufengg/widendeep/blob/master/wnd_criteo.ipynb

`bit.ly/widendeep-code`  `bit.ly/widendeep-slides`



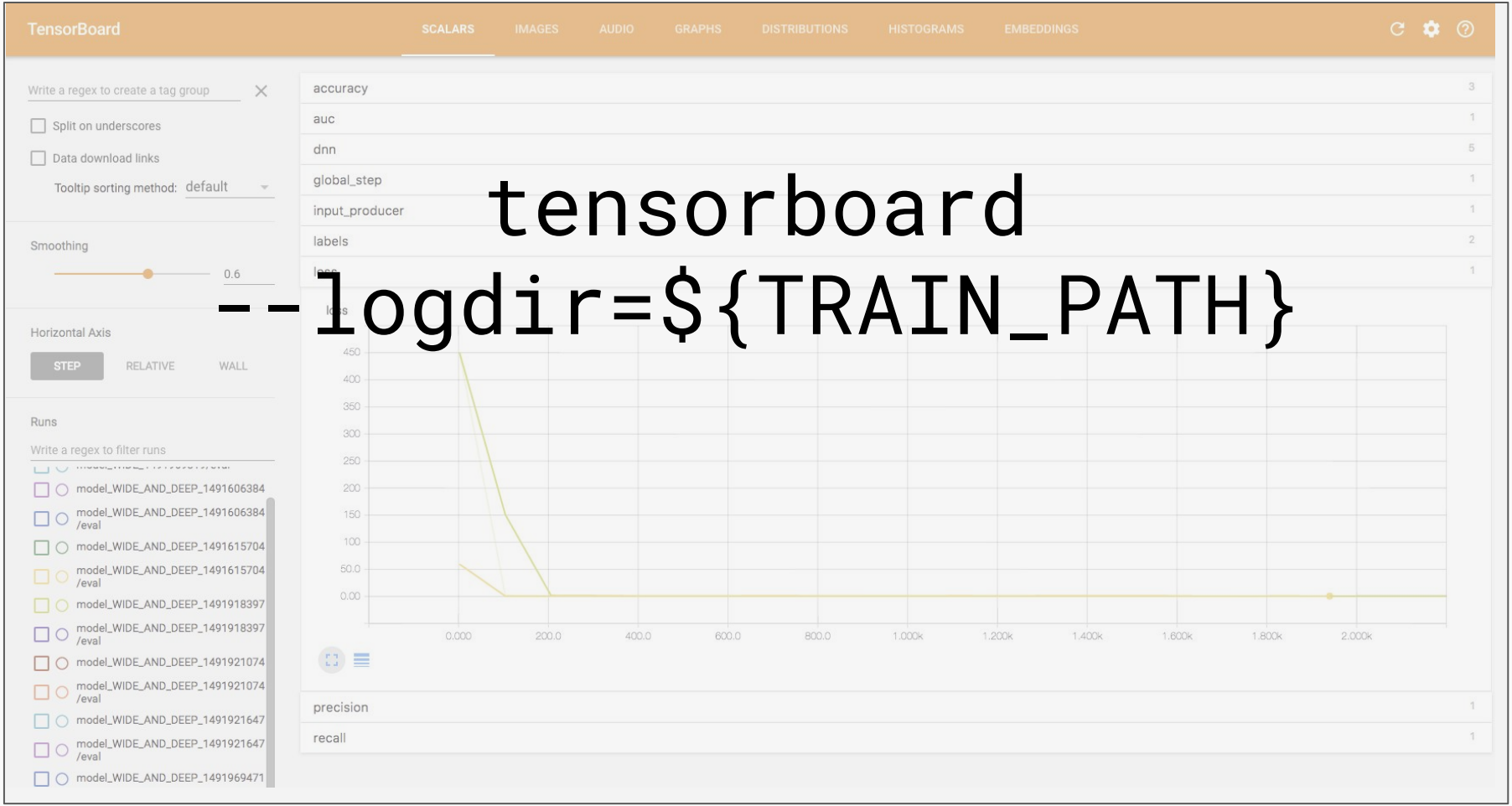
tensorboard --logdir=models/

Cloud Training

- Point the data to the cloud
- Set some configuration(s)
- Train!

This would be a good time to upgrade our dataset

To the code!



tensorboard --logdir=${TRAIN_PATH}

@YufengG

Training

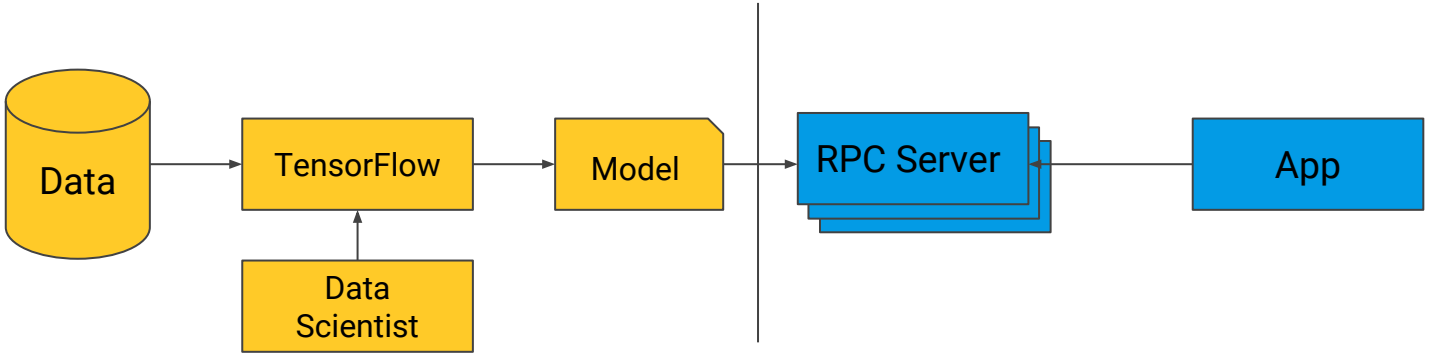
many

examples

Prediction

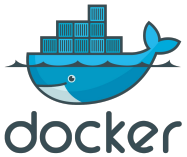
answer questions

What is Serving?

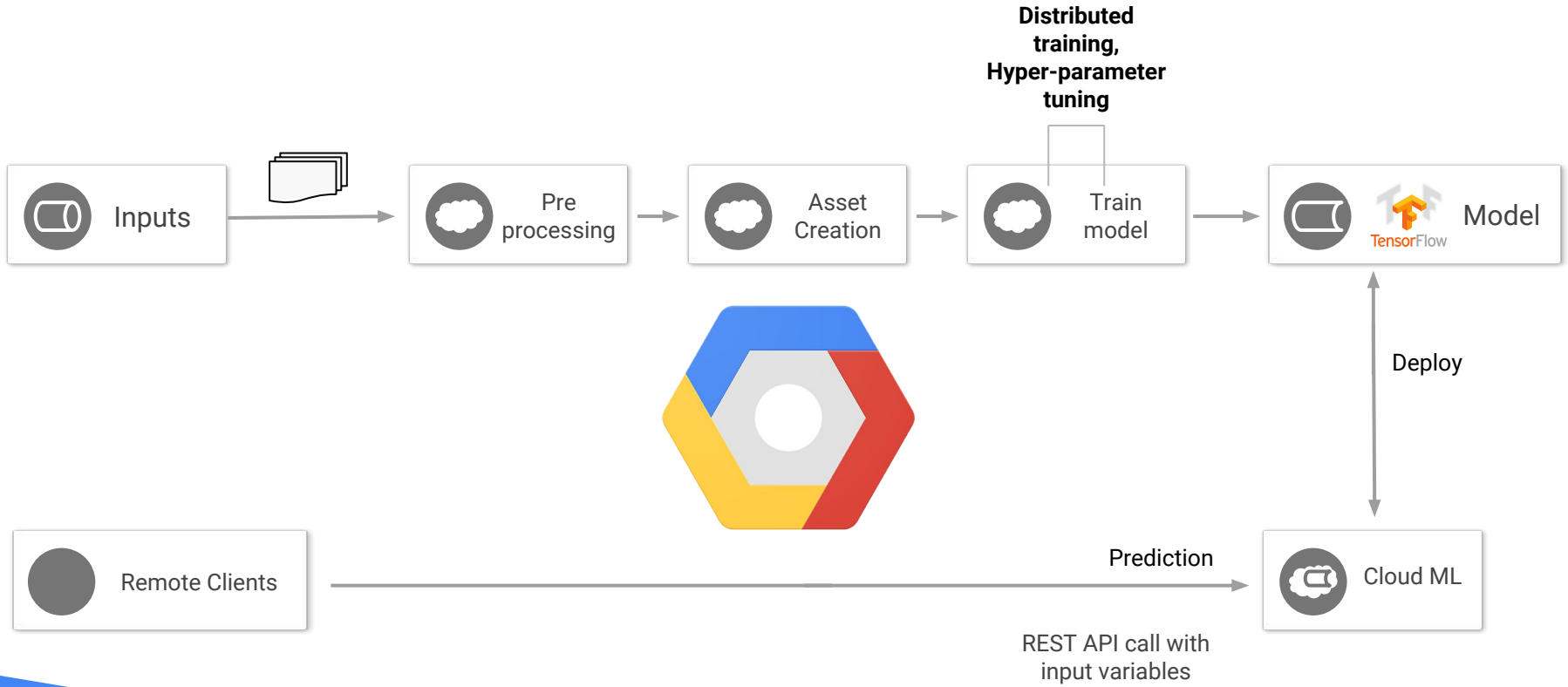


What is TensorFlow Serving?

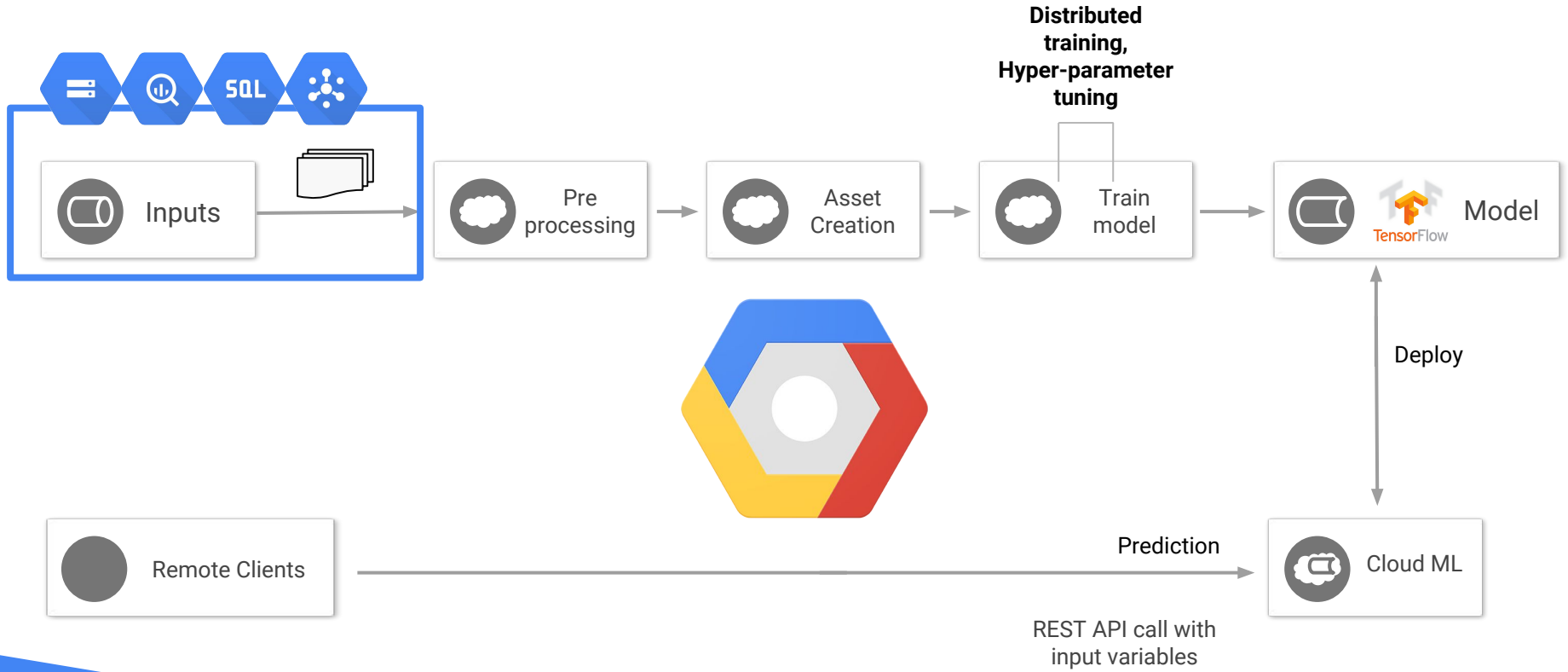
- C++ Libraries
  - TensorFlow model save / export formats
  - Generic core platform
- Binaries
  - Best practices out of the box
  - Docker containers, K8s tutorial
- Hosted Service across
  - Google Cloud ML
  - Internal service



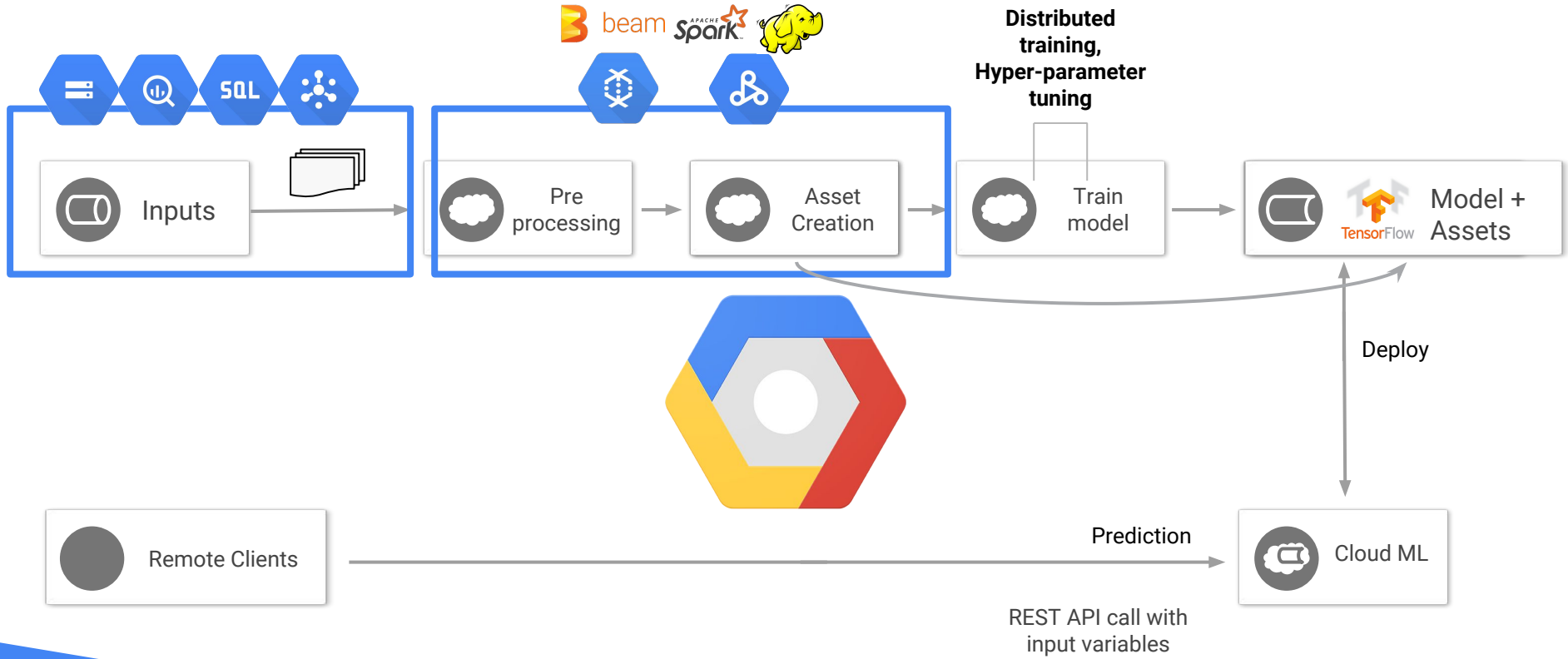
End to End ML Pipeline



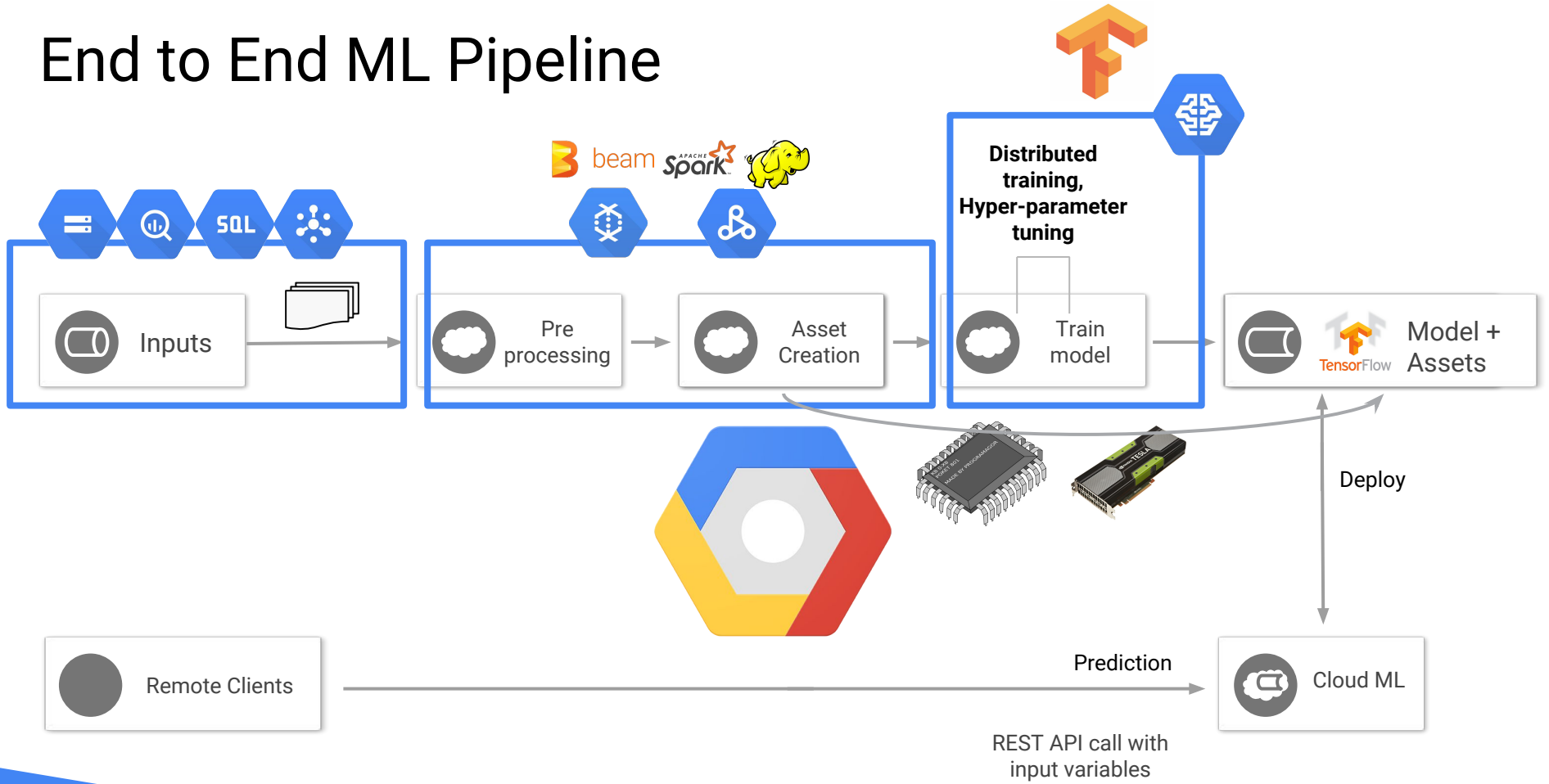
End to End ML Pipeline



End to End ML Pipeline



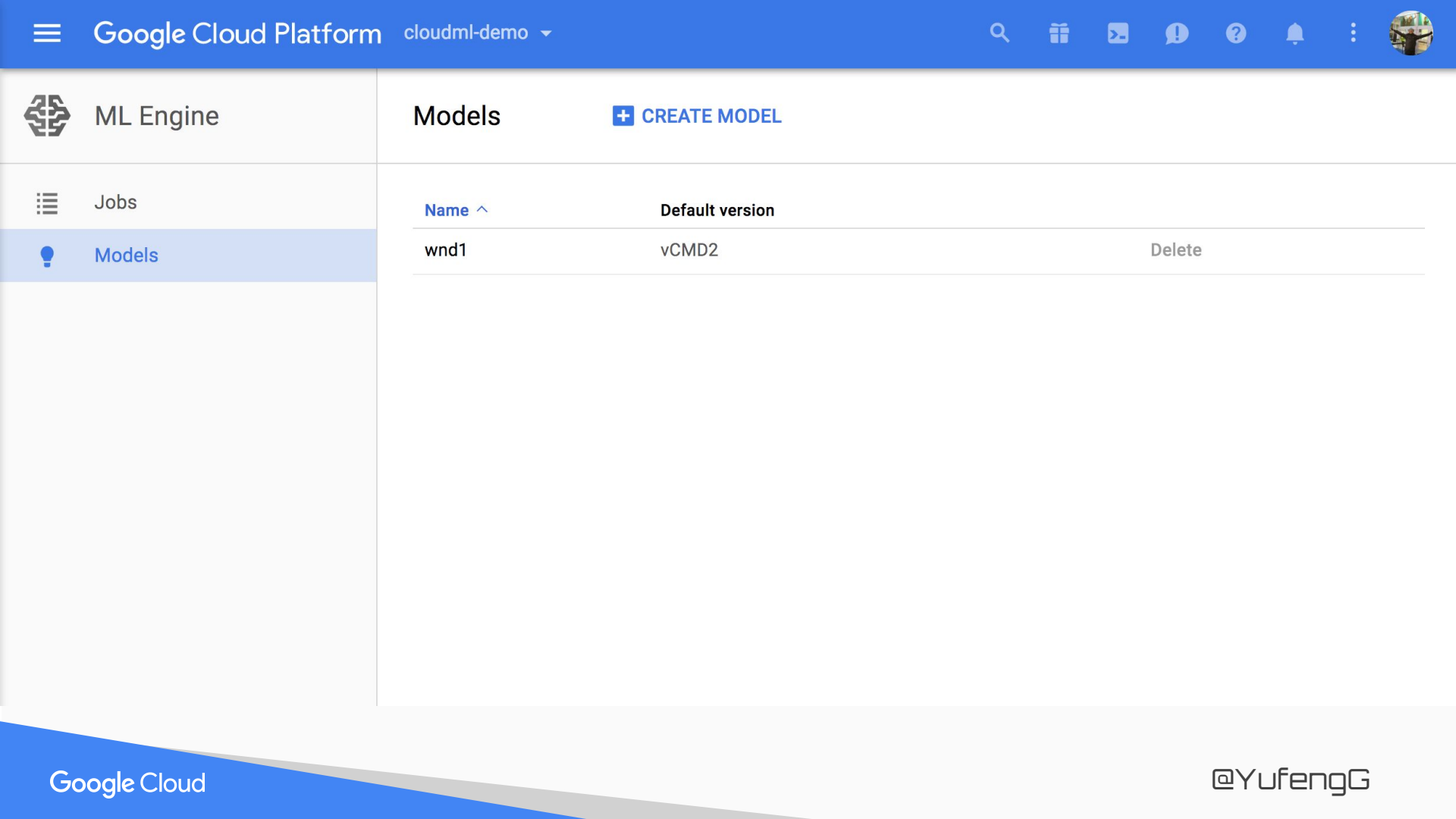
End to End ML Pipeline

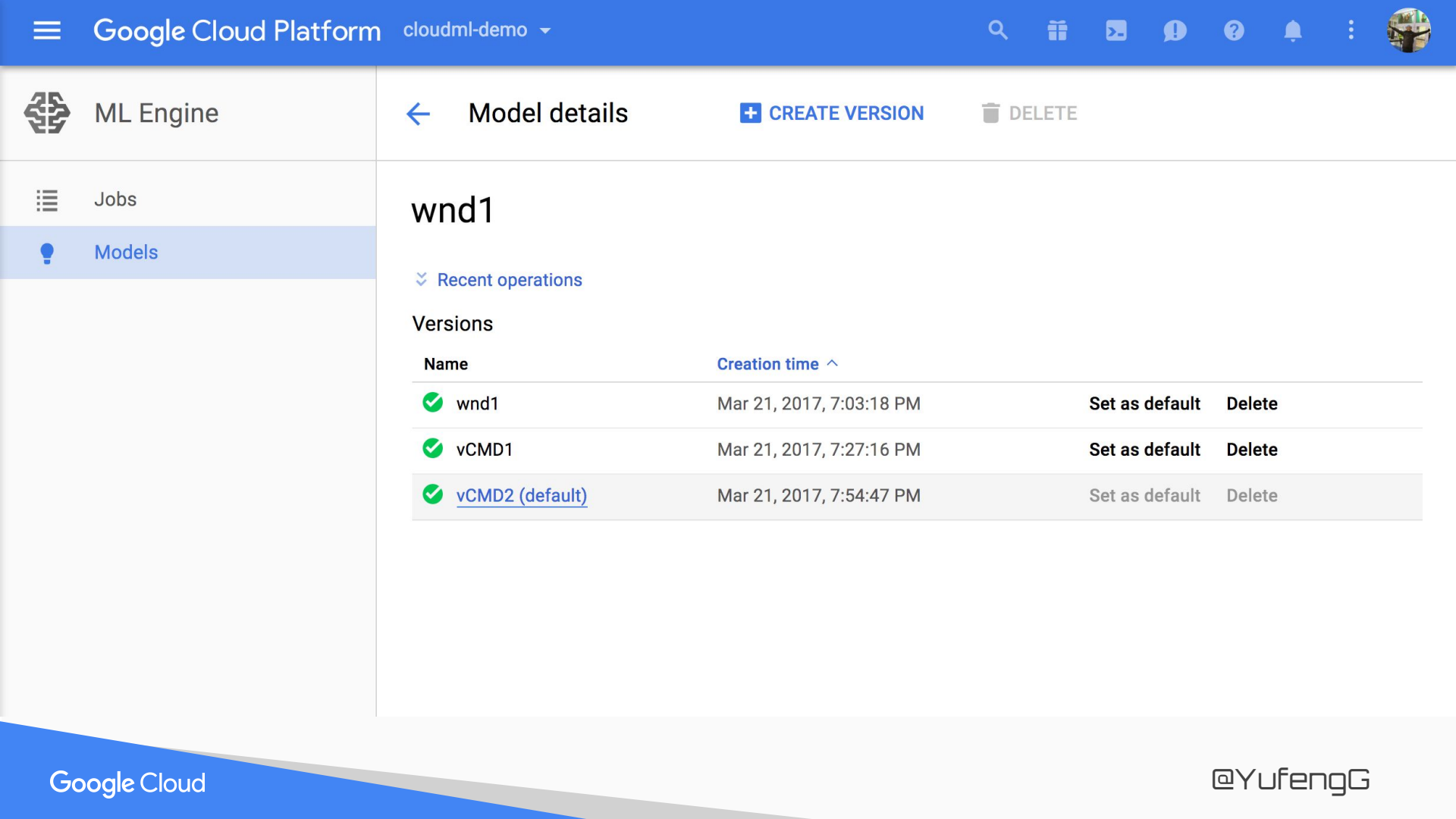


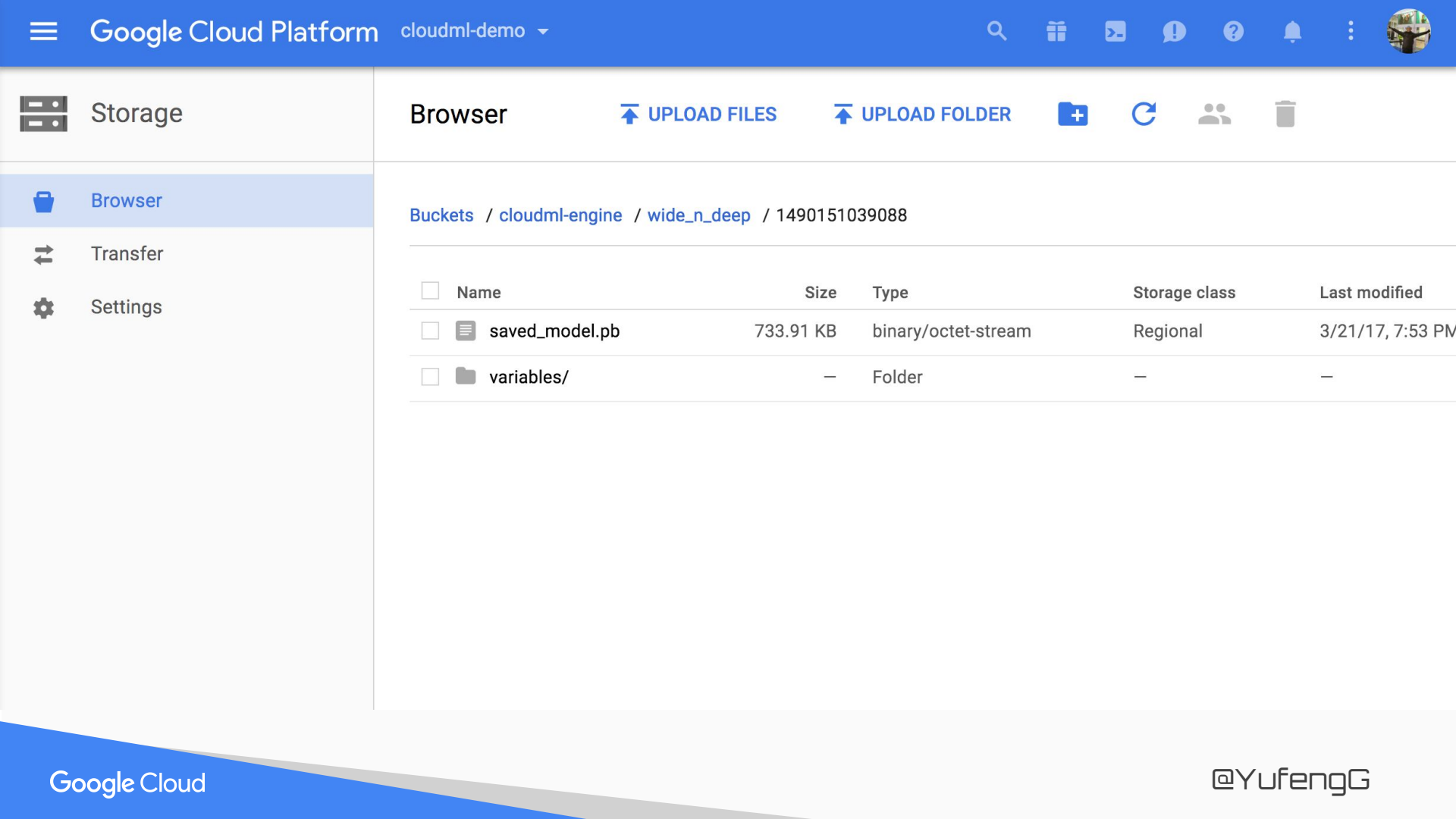
Model Creation

```
export MODEL_NAME='cloudwnd'
export VERSION_NAME='learn_runner_standard'
export DEPLOYMENT_SOURCE='gs://cloudml-engine/widendeep_yufeng
g_20170410_164903/model_WIDE_AND_DEEP_1491857627/export/Servo/
1491857907860'

$ gcloud ml-engine versions create $VERSION_NAME --model
$MODEL_NAME --origin $DEPLOYMENT_SOURCE
Creating version (this might take a few minutes)......
```


Google Cloud Platform
cloudml-demo
ML Engine
Jobs
Models
Models
CREATE MODEL
Name
Default version
wnd1
vCMD2
Delete


Google Cloud Platform
cloudml-demo
ML Engine
Jobs
Models
Model details
CREATE VERSION
DELETE
wnd1
Recent operations
Versions
Name
Creation time
wnd1
Mar 21, 2017, 7:03:18 PM
Set as default
Delete
vCMD1
Mar 21, 2017, 7:27:16 PM
Set as default
Delete
vCMD2 (default)
Mar 21, 2017, 7:54:47 PM
Set as default
Delete
Google Cloud
@YufengG


Google Cloud Platform
cloudml-demo
Storage
Browser
Transfer
Settings
Browser
UPLOAD FILES
UPLOAD FOLDER
Buckets / cloudml-engine / wide_n_deep / 1490151039088
Name
Size
Type
Storage class
Last modified
saved_model.pb
733.91 KB
binary/octet-stream
Regional
3/21/17, 7:53 PM
variables/
–
Folder
–
–
Google Cloud
@YufengG

Instance Prediction

Predictions from the command line

Also available as a REST API call

```
gcloud ml-engine predict --model
$MODEL_NAME --version $VERSION_NAME
--json-instances test.json
```

```
{
  "age": 25,
  "workclass": " Private",
  "education": " 11th",
  "education_num": 7,
  "marital_status": "
Never-married",
  "occupation": "
Machine-op-inspct",
  "relationship": "
Own-child",
  "race": " Black",
  "gender": " Male",
  "capital_gain": 0,
  "capital_loss": 0,
  "hours_per_week": 40,
  "native_country": "
United-States"
}
```

```
{
  "age": 42,
  "workclass": "
Self-emp-inc",
  "education": " HS-grad",
  "education_num": 9,
  "marital_status": "
Married-civ-spouse",
  "occupation": "
Exec-managerial",
  "relationship": " Husband",
  "race": " White",
  "gender": " Male",
  "capital_gain": 5178,
  "capital_loss": 0,
  "hours_per_week": 50,
  "native_country": "
United-States"
}
```

```
$ gcloud ml-engine predict --model wnd1 --version vCMD2 --json-instances census.json
CLASSES  LOGISTIC                 LOGITS                   PROBABILITIES
0        [0.0051437602378427 98]  [-5.2648138999938965]    [0.9948562383651733, 0.0051437602378427 98]
1        [0.8839852213859558]     [2.0307230949401855]     [0.1160147413611412, 0.8839852213859558]
```

```
PROBABILITIES
[0.9948562383651733, 0.0051437602378427 98]
[0.1160147413611412, 0.8839852213859558]
```

Break time!

When we come back:

Distributed learning, GPUs,

Parameter tuning

Experiments, learn_runner

```
m = build_estimator(model_type, model_dir)

m.fit(input_fn=generate_input_fn(train_file), steps=train_steps)
print('fit done')

results = m.evaluate(input_fn=generate_input_fn(test_file), steps=test_steps)
print('evaluate done')

print('Accuracy: %s' % results['accuracy'])

export_folder = m.export_savedmodel(
  export_dir_base = export_dir,
  input_fn=serving_input_fn
)

print('Model exported to ' + export_dir)
```

```
def serving_input_fn():
    feature_placeholders = {
        column: tf.placeholder(column_to_dtype(column), [None])
        for column in FEATURE_COLUMNS
    }
    # DNNCombinedLinearClassifier expects rank 2 Tensors,
    # but inputs should be rank 1, so that we can provide
    # scalars to the server
    features = {
        key: tf.expand_dims(tensor, -1)
        for key, tensor in feature_placeholders.items()
    }

    return input_fn_utils.InputFnOps(
        features, # input into graph
        None,
        feature_placeholders # tensor input converted from request
    )
```

learn_runner wraps it up with the experiment_fn

```
experiment_fn = generate_experiment(
  model_dir, train_file, test_file, model_type)

learn_runner.run(experiment_fn, model_dir)
```

```
def generate_experiment(output_dir, train_file, test_file, model_type):
  def _experiment_fn(output_dir):
    train_input_fn = generate_input_fn(train_file)
    eval_input_fn = generate_input_fn(test_file)
    my_model = build_estimator(model_type=model_type,
                               model_dir=output_dir)

    experiment = tf.contrib.learn.Experiment(
      my_model,
      train_input_fn=train_input_fn,
      eval_input_fn=eval_input_fn,
      train_steps=1000
      ,
      export_strategies=[saved_model_export_utils.make_export_strategy(
        serving_input_fn,
        default_output_alternative_key=None
      )]
    )
    return experiment

  return _experiment_fn
```

GPU? (Currently Beta)

- How many do you want? 1, 4, or 8. **Per machine!**
- Supported GPU regions
  - **us-east1, asia-east1, europe-west1**

GPU Configuration

```
export PROJECT_ID=<YOUR_PROJECT_ID>
export BUCKET=gs://<YOUR_BUCKET>
export JOB_NAME=widendeep_${USER}_$(date
+%Y%m%d_%H%M%S)
export TRAIN_PATH=${BUCKET}/criteo_wnd
```

```
$ gcloud ml-engine jobs
submit training ${JOB_NAME}
--package-path=trainer
--module-name=trainer.task
--job-dir=${TRAIN_PATH}
--config config_standard.yaml
```

config_standard.yaml

```
trainingInput:
  scaleTier: STANDARD_1
  region: us-central1
```

config_gpu.yaml

```
trainingInput:
  region: us-east1
  scaleTier: CUSTOM
  masterType: complex_model_m
  workerType: standard_gpu
  parameterServerType: large_model
  workerCount: 4
  parameterServerCount: 3
```

GPU will push your usage levels up, up, up

Machine type	ML training units per instance
standard	1
large_model	3
complex_model_s	2
complex_model_m	3
complex_model_l	6
standard_gpu	3
complex_model_m_gpu	12
complex_model_l_gpu	24

Default quota is **25** ML training units

master:
complex_model_m (3)

4x worker:
standard_gpu(3)

3x param server:
large_model(3)

= 3 + 4x3 + 3x3 = **24**

https://cloud.google.com/ml-engine/pricing#machine_types_for_custom_cluster_configurations

Experimentation

- Push the system with more machines, more data!
- Try different values for your parameters
  - we have *many, many* parameters!
- Batch size, optimizer, feature crosses (and other feature engineering options)
- Model structure: DNN layer size/count, embedding dimensions

Training

many *examples*

Prediction

answer questions

Training

many *examples*

Prediction

answer questions

Training

many *examples*

Prediction

answer questions

Thank you!

@YufengG

Resources:

Cloud Machine Learning Engine
`cloud.google.com/ml-engine`

TensorFlow
`tensorflow.org`

The End