



---

Effective ML



# Advanced ML with TensorFlow on GCP

---

## **End-to-End Lab on Structured Data ML**

Production ML Systems

Image Classification Models

Sequence Models

Recommendation Systems

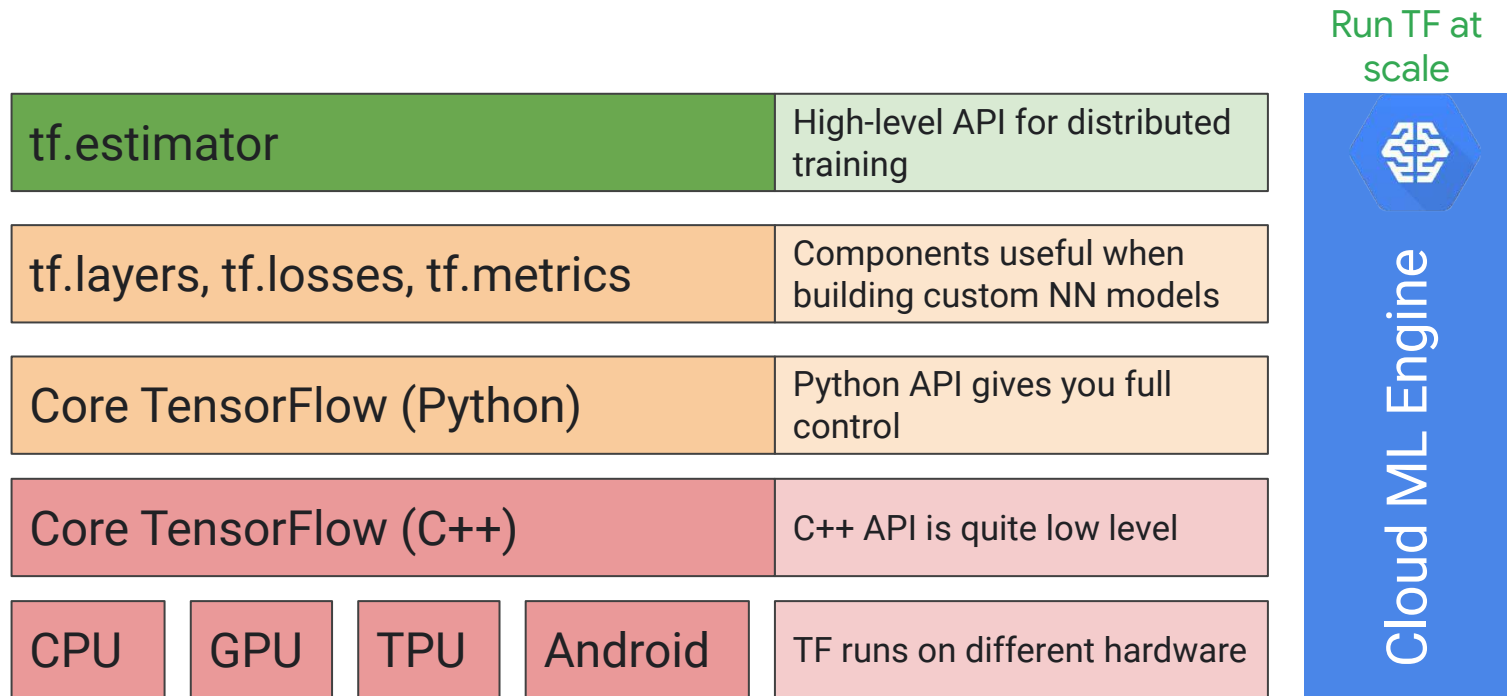


# Steps involved in doing ML on GCP

- 1 Explore the dataset
- 2 Create the dataset
- 3 Build the model
- 4 Operationalize the model



# You use distributed TensorFlow on Cloud ML Engine



# Many machine learning frameworks can handle toy problems

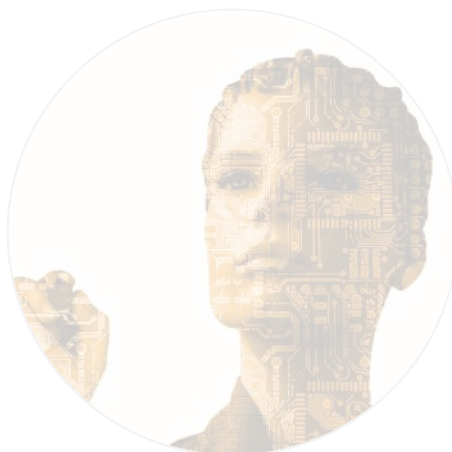


# To build effective ML, you need:



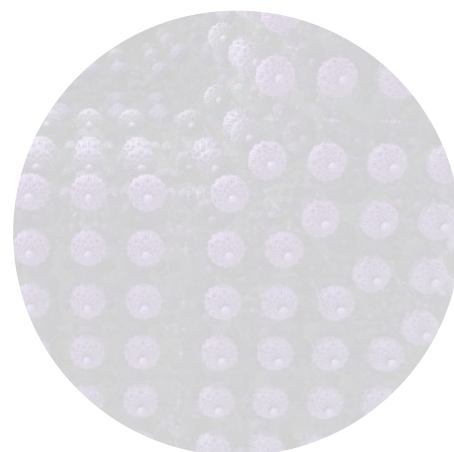
---

Big Data



---

Feature  
Engineering

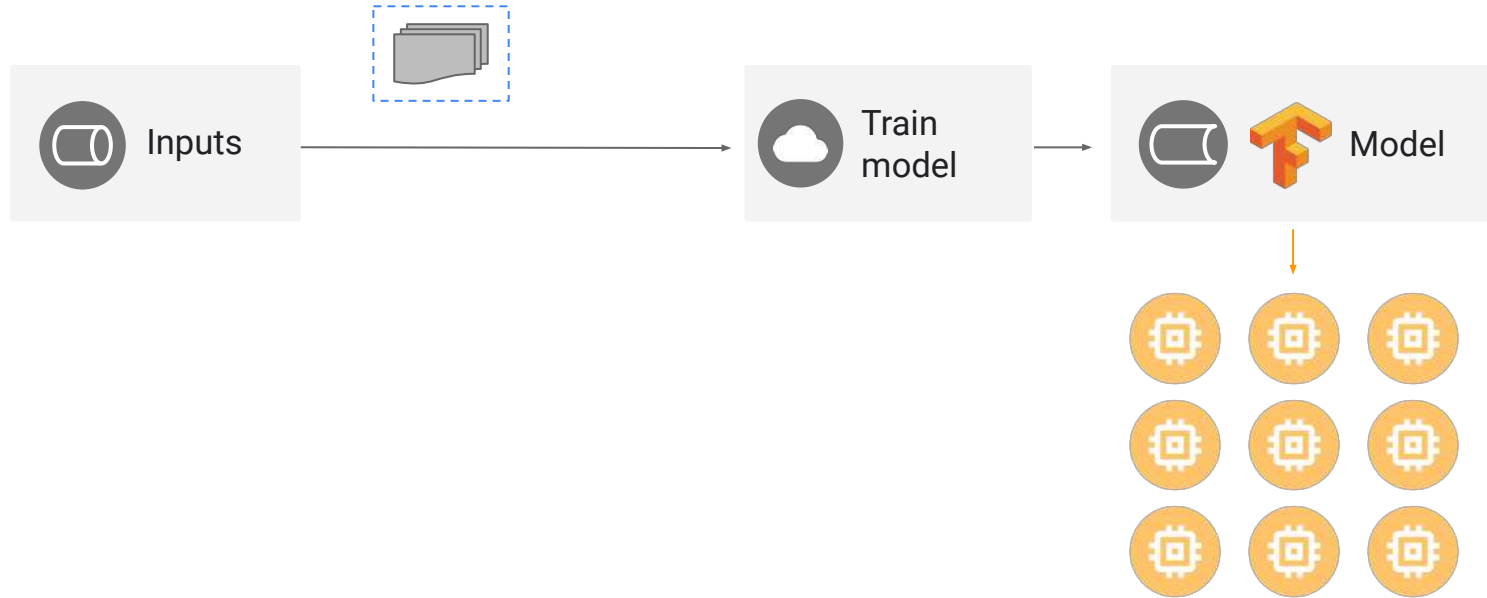


---

Model  
Architectures



As your data size increases, batching and distribution become important

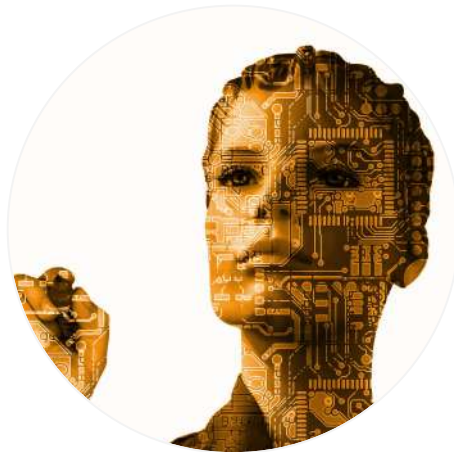


# To build effective ML, you need:



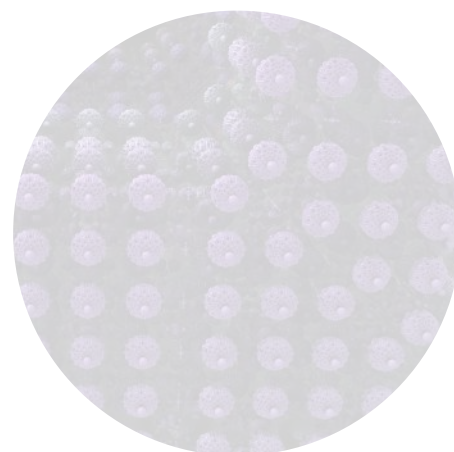
---

Big Data



---

Feature  
Engineering



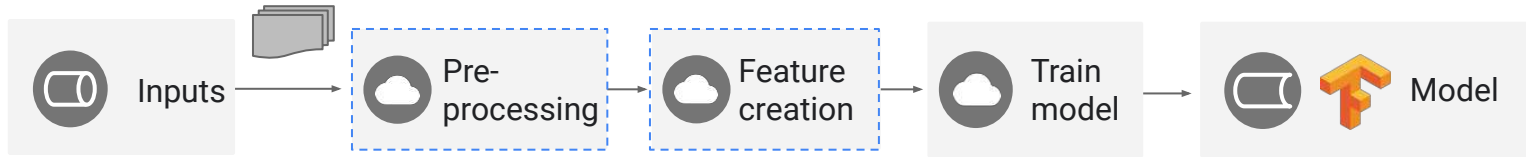
---

Model  
Architectures





# Input necessary transformations

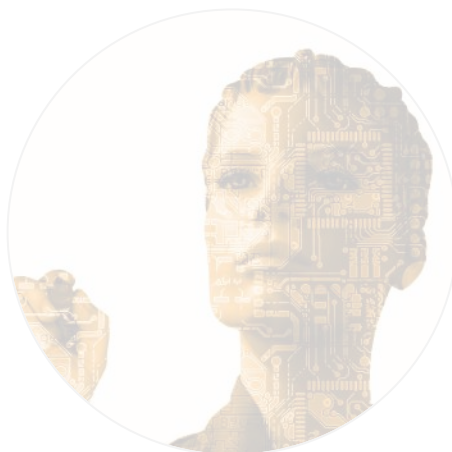


# To build effective ML, you need:



---

Big Data



---

Feature  
Engineering



---

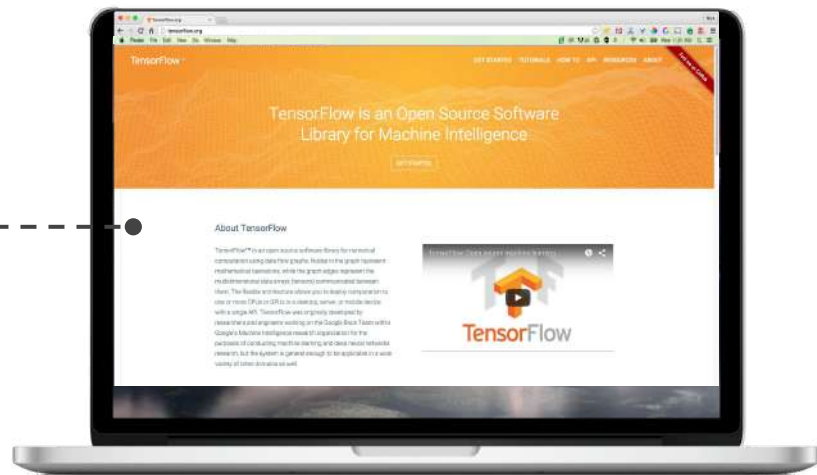
Model  
Architectures



# Sharing our tools with researchers and developers around the world

Released in Nov. 2015

**#1**  
**repository**  
for “machine learning”  
category on GitHub



# To build effective ML, you need:



---

Big Data



---

Feature  
Engineering



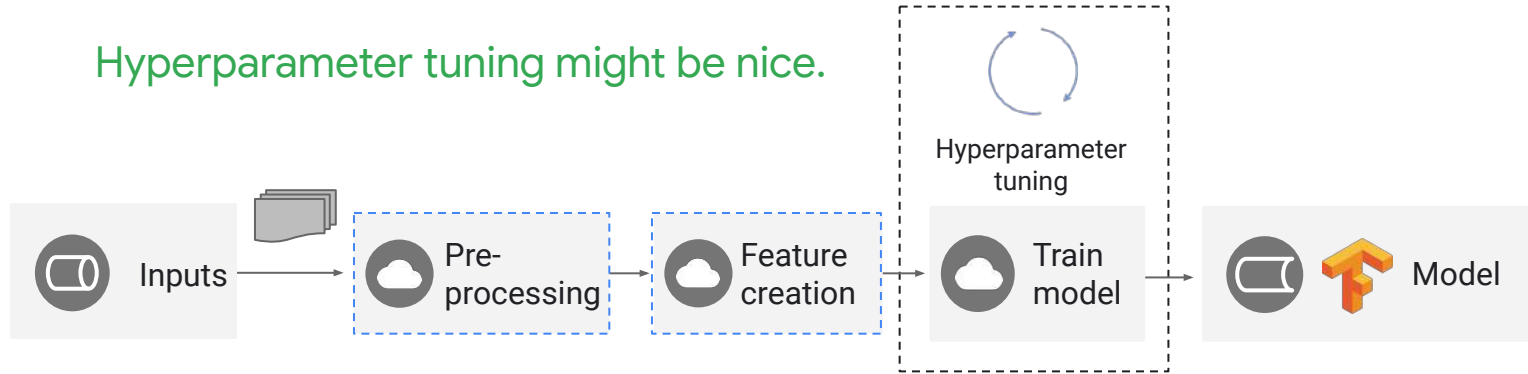
---

Model  
Architectures

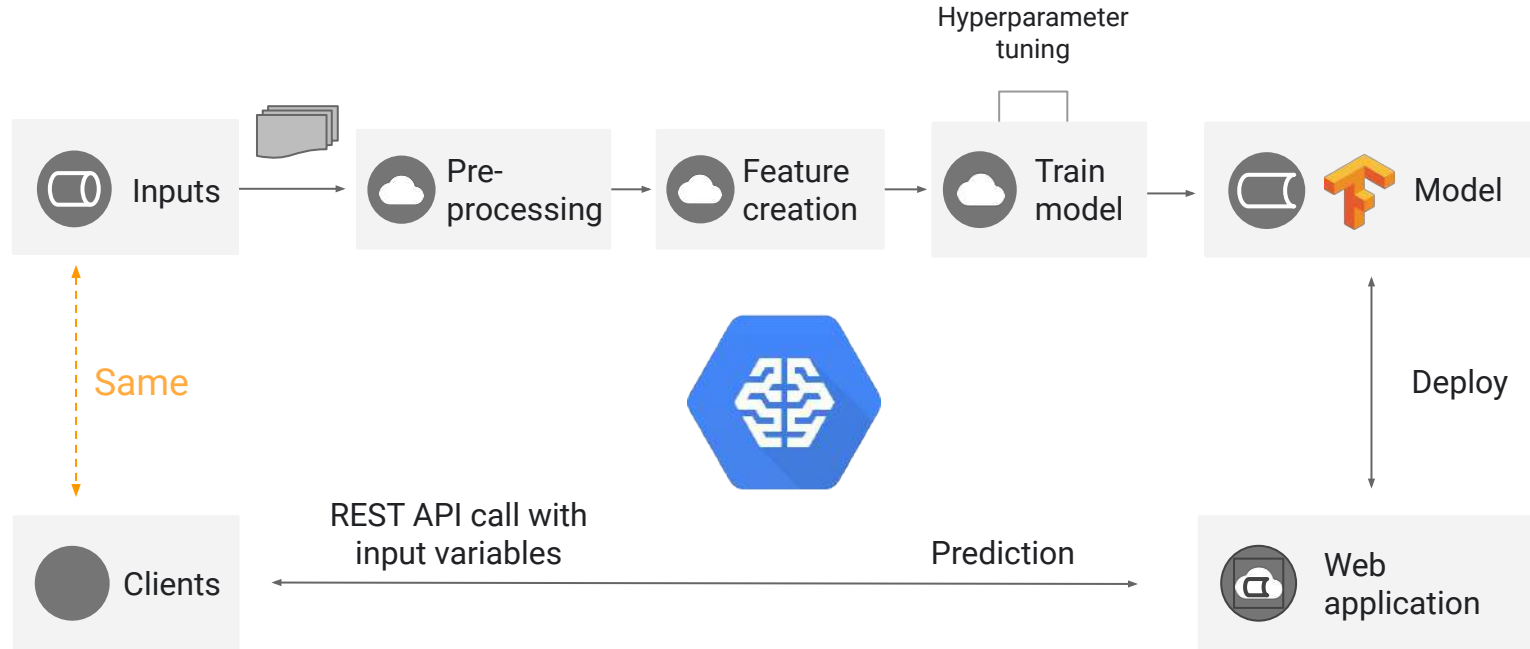


# What else does an ML framework need to provide?

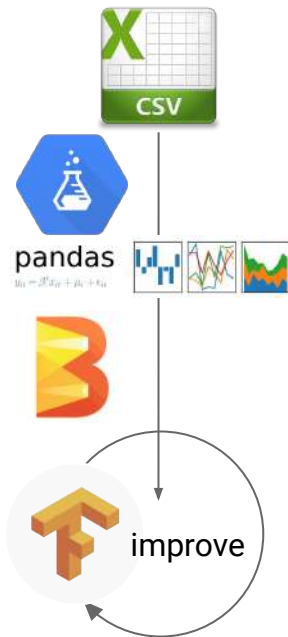
Hyperparameter tuning might be nice.



# Cloud machine learning: Repeatable, scalable, tuned



# In Cloud Datalab, start locally on a sampled dataset



Google Cloud Datalab tfclassic (unsaved changes)

Notebook Add Code Add Markdown Delete Move Up Move Down Run Clear Reset Session Wedgets Navigation Help

### 2D: WORKING WITH NOW-NEVER TENSORFLOW

This notebook is Lab2b of CPB 102: Google's course on Machine Learning using Cloud ML.

In this notebook, we will work with relatively low-level TensorFlow functions to implement a linear regression model. We will use this notebook to demonstrate early stopping – a technique whereby training is stopped once the error on the validation dataset starts to increase.

```
import datalab.bigquery as bq
import tensorflow as tf
import pandas as pd
import numpy as np
import shutil
```

Code to read data and compute error is the same as Lab2a:

```
def read_dataset(filename):
    return pd.read_csv(filename, header=None, names=['pickuplon', 'pickuplat', 'dropofflon', 'dropofflat', 'passengers', 'fare_amount'])

df_train = read_dataset('../labia/taxi-train.csv')
df_valid = read_dataset('../labia/taxi-valid.csv')
df_test = read_dataset('../labia/taxi-test.csv')
df_train[:5]
```

```
FEATURE_COLS = np.arange(0,5)
TARGET_COL = 'fare_amount'
```

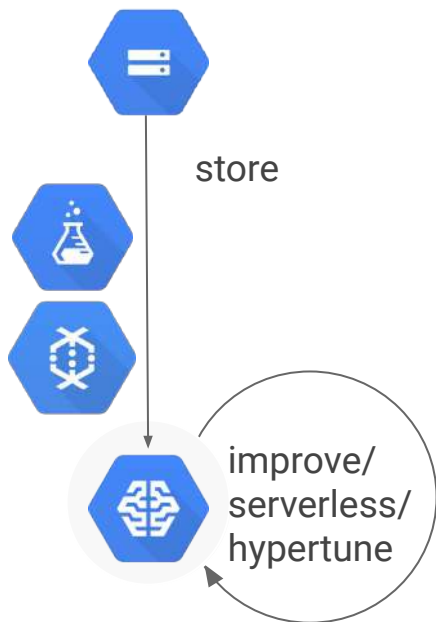
```
def compute_rmse(actual, predicted):
    return np.sqrt(np.mean((actual-predicted)**2))

def print_rmse(model):
    print "Train RMSE = {}".format(compute_rmse(df_train[TARGET_COL], model.predict(df_train.iloc[:,FEATURE_COLS].values)))
    print "Valid RMSE = {}".format(compute_rmse(df_valid[TARGET_COL], model.predict(df_valid.iloc[:,FEATURE_COLS].values)))
```

Linear Regression



# Then, scale it out to GCP using serverless technology



Google Cloud Datalab · cloudml (autosaved)

Notebook · Add Code · Add Markdown · Delete · Move Up · Move Down · Run · Clear · Reset Session · Widgets · Navigation · Help

### Training on cloud

In order to train on the cloud, we have to copy the model and data to our bucket on Google Cloud Storage (GCS).

```
$bash
rm -rf taxifare.tar.gz taxi_trained
tar cvfz taxifare.tar.gz taxifare
gsutil cp taxifare.tar.gz gs://$BUCKET/taxifare/source/taxifare.tar.gz
gsutil cp ../lab1a/*.csv gs://$BUCKET/taxifare/input/
gsutil -m rm -r -f gs://$BUCKET/taxifare/taxi_preproc
gsutil -m rm -r -f gs://$BUCKET/taxifare/taxi_trained
```

Pushing...

When you run your preprocessor, you have to change the input and output to be on GCS.

Using DirectPipelineRunner runs Dataflow locally, but the inputs & outputs are on the cloud. Using BlockingDataflowPipelineRunner will use Cloud Dataflow (and take much longer because of the overhead involved for such a small dataset). To see the status of your BlockingDataflowPipelineRunner job, visit <https://console.cloud.google.com/dataflow>.

```
# imports
import apache_beam as beam
import google.cloud.ml as ml
import google.cloud.ml.dataflow.io.tfrecordio as tfrecordio
import google.cloud.ml.io as io
import os

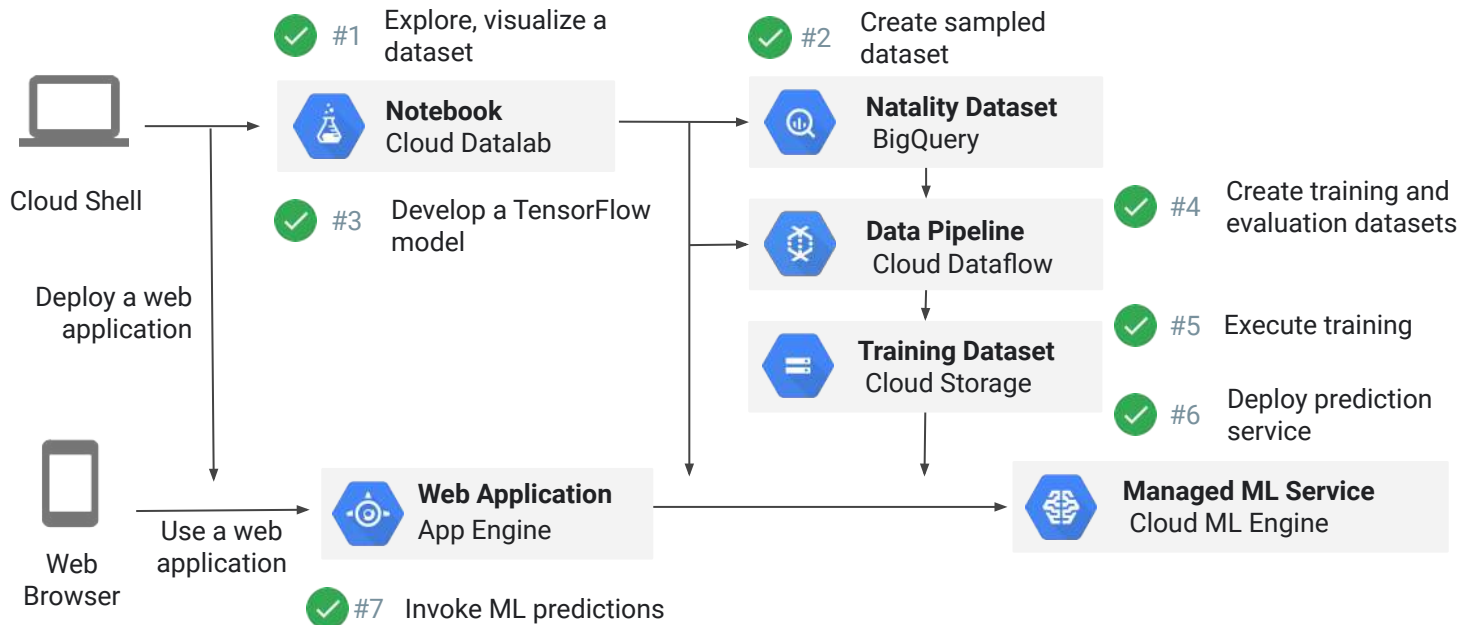
# Change as needed
#RUNNER = 'DirectPipelineRunner'
RUNNER = 'BlockingDataflowPipelineRunner'

# defines
feature_set = TaxifareFeatures()
OUTPUT_DIR = 'gs://{0}/taxifare/taxi_preproc'.format(BUCKET)
```





# The end-to-end machine learning set of labs



cloud.google.com

