Inteligencia Artificial no Google Cloud Platform

AI Platform, KubeFlow e TFX

Pythian

Recifense
MSc em Inteligencia Artificial

Data Scientist | ML Engineer @ Pythian
GDE in ML
GDG Cloud Ottawa-Montreal

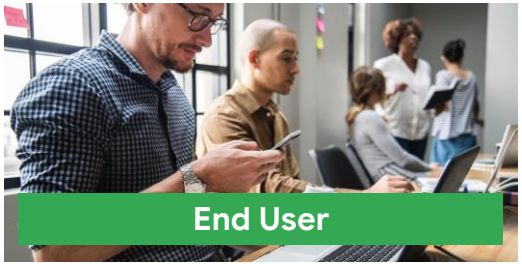
carlos-timoteo-data-scientist

Pythian

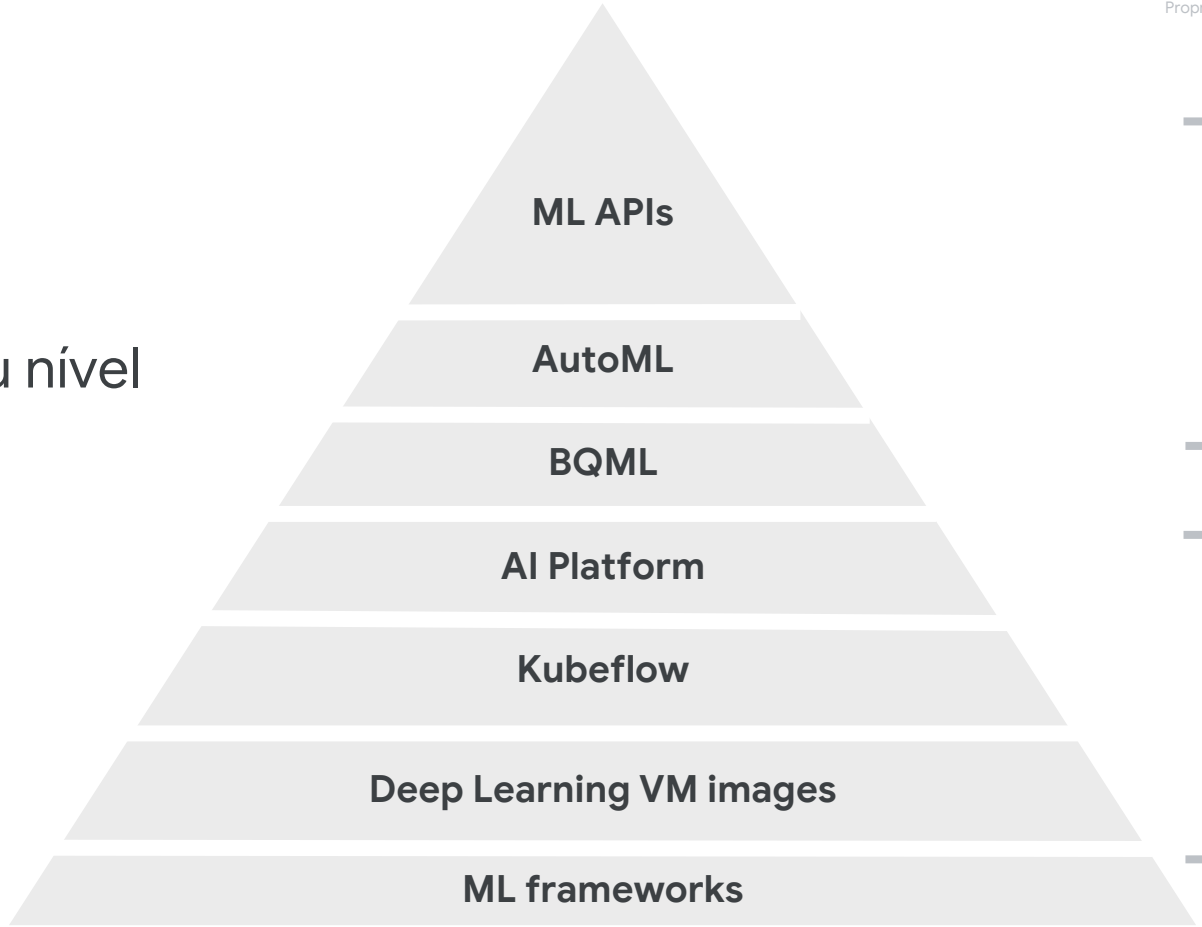
AI Platform Overview

Google Cloud

Como conectar usuários para produzir resultados 10x mais impactantes?

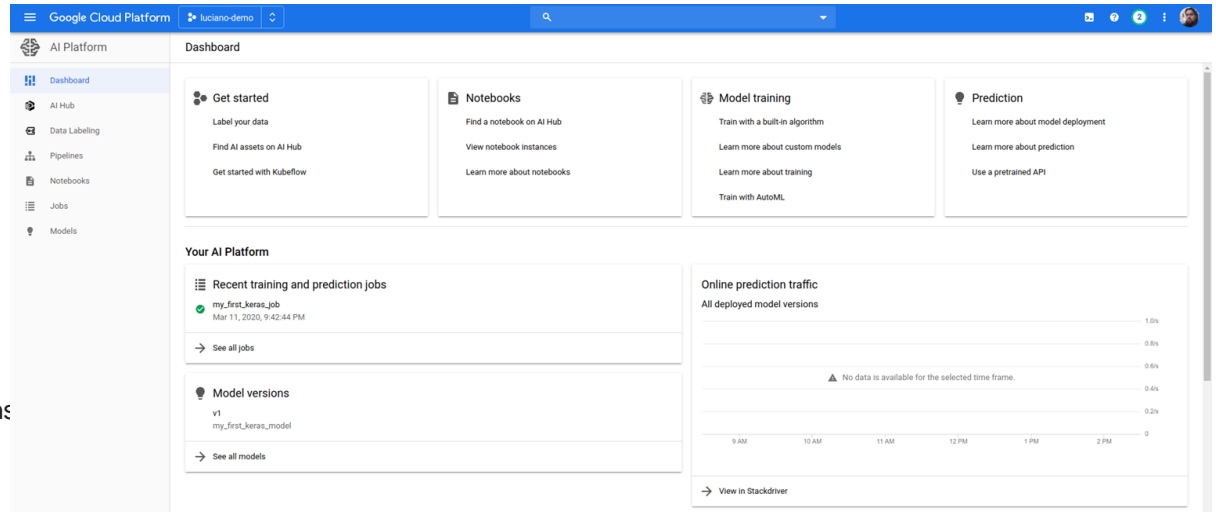


Escolha o seu nível de abstração



Introdução ao Cloud AI Platform

- Ambiente de desenvolvimento fim-a-fim para IA dentro da console GCP
- Construído com Kubeflow, oferece uma cadeia de ferramentas integrada - de engenharia de dados à implementação de modelos. Sem “lock-in”
- Permite implementar suas soluções on-premises ou na Google Cloud sem mudanças significativas de código
- Acesso à tecnologias Google AI de ponta como Tensorflow, Tensorflow Extended (TFX), TPUs quando implementando sua solução em produção



O que está incluso no Cloud AI Platform

AI Hub

Kubeflow
(On premises)

Deep learning VM image (DLVM)

Machine Learning mais rápido e fácil no GCE

- **Prototipação Acelerada**
  Prototipe seu projeto rapidamente utilizando VMs pré-configuradas para Deep Learning

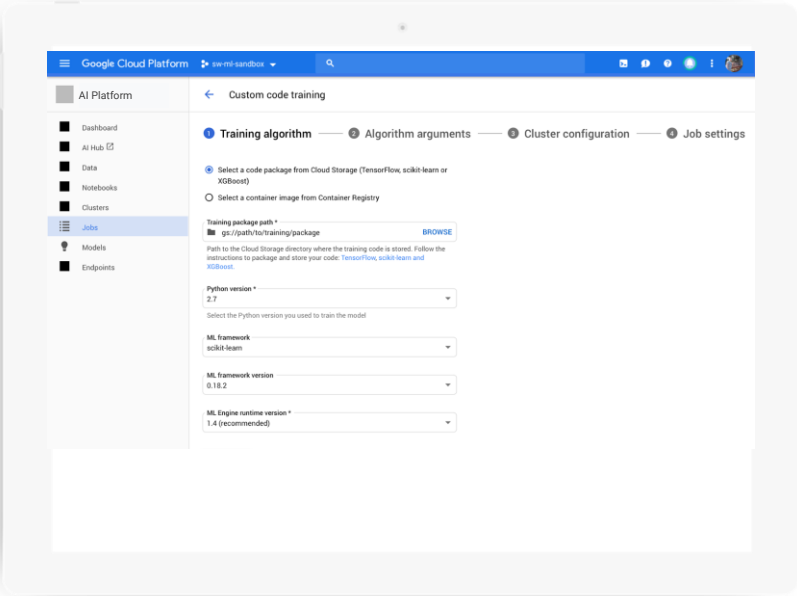
- **Suporte a CPU, GPU e TPU**
  Utilize aceleradores para seus modelos, como GPU ou TPU, com poucos cliques

- **Performance otimizada para Google Cloud**
  Bibliotecas e configurações otimizadas para a melhor performance para a sua infraestrutura - assim você não se preocupa com isso

- **Flexibilidade**
  Escolha entre diferentes ML frameworks como TensorFlow, PyTorch e scikit-learn ou instale o framework que você preferir sob a imagem base

Gerenciamento serverless de modelos

- Treine modelos abstraindo a infraestrutura
- Suporta os frameworks de machine learning mais populares. Suporta, também, containers Docker customizados
- Realiza treinamentos distribuídos e utiliza GPUs e TPUs para finalizar jobs mais rápido
- Melhora modelos através da otimização de hiperparâmetros automatizada



Notebooks gerenciados e ambientes pré-configurados

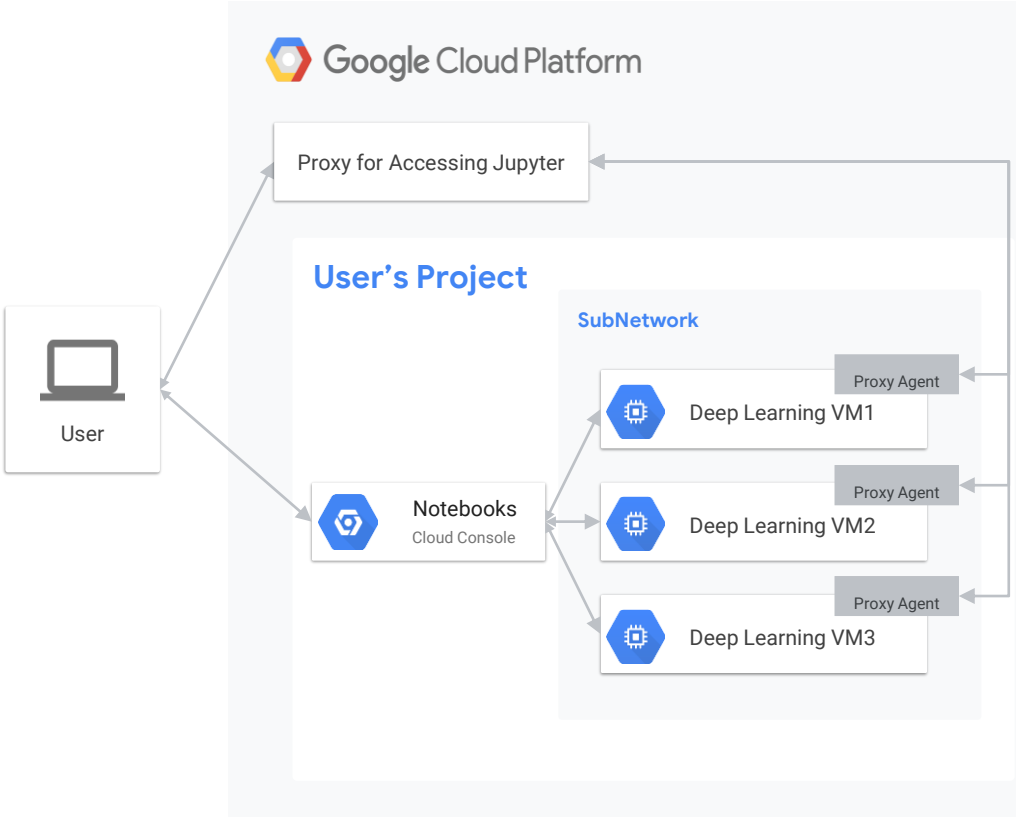
Bibliotecas GCP pré-instaladas

Ambientes pré-configurados para:

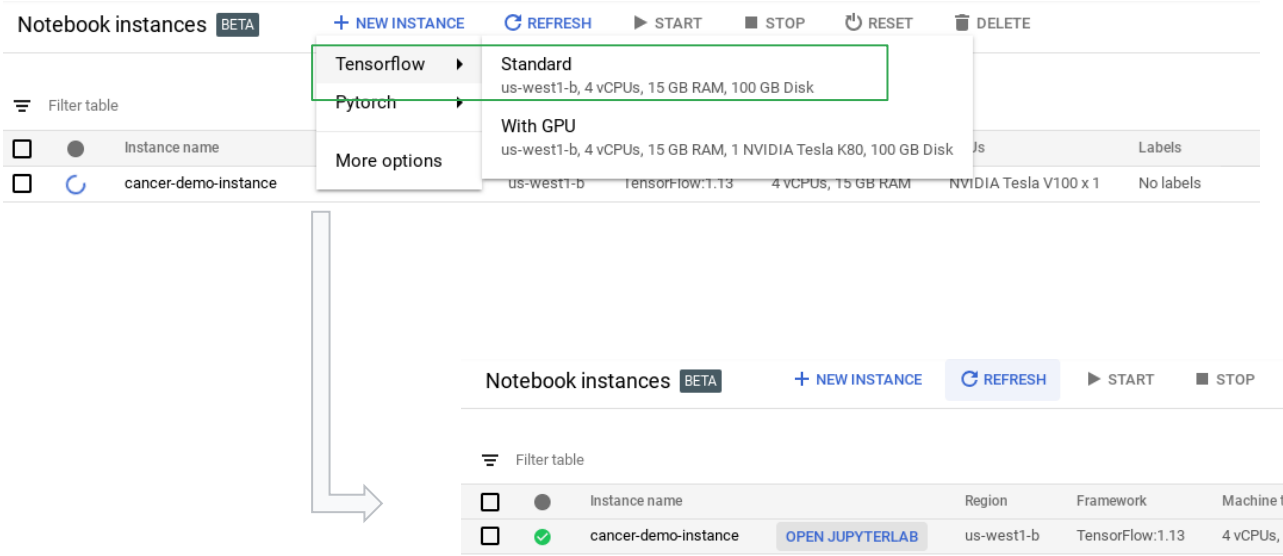
NumPy SciPy scikit learn

matplotlib TensorFlow Keras

dmlc XGBoost PyTorch



Crie uma instância de Notebook



Importe seus dados do Cloud Storage

[ ]:
```
!pip3 install tensorflow_hub
```

[1]:
```
import sys
# Workaround: pip installs to a wrong location on these notebooks
sys.path.insert(0,'/home/jupyter/.local/lib/python3.5/site-packages')
import os
import json
import random
import base64
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow.keras as ks
import IPython.display
from tensorflow.python.lib.io import file_io

storage_bucket = 'gs://cancer-demo-data/ht-kg-histopathologic-cancer-detection/'
train_files = [storage_bucket+f for f in ['train_0.tfrecords', 'train_1.tfrecords', 'train_2.tfrecords']]
eval_files = [storage_bucket+f for f in ['train_3.tfrecords']]
xval_files = [storage_bucket+f for f in ['train_4.tfrecords']]

row_count_file = storage_bucket + 'record_counts.json'
examples_file = storage_bucket + 'examples.zip'
```

Treine seu modelo onde quiser

On-Premises

```
from fairing.config.config_kubeflow import KubeflowConfig
from fairing.jobs.train_job import TrainJob

job = TrainJob(train,
               base_docker_image="gcr.io/caip-dexter-dev/jaas:py3.5.3",
               destination_dir='gs://cancer_detection_demo/',
               runtime_config=KubeflowConfig())

job.submit()
job.get_logs()
```

Google Cloud

```
from fairing.config.config_cmle import CMLEConfig
from fairing.jobs.train_job import TrainJob

job = TrainJob(train,
               base_docker_image="gcr.io/caip-dexter-dev/jaas:py3.5.3",
               destination_dir='gs://cancer_detection_demo/',
               runtime_config=CMLEConfig())

job.submit()
```

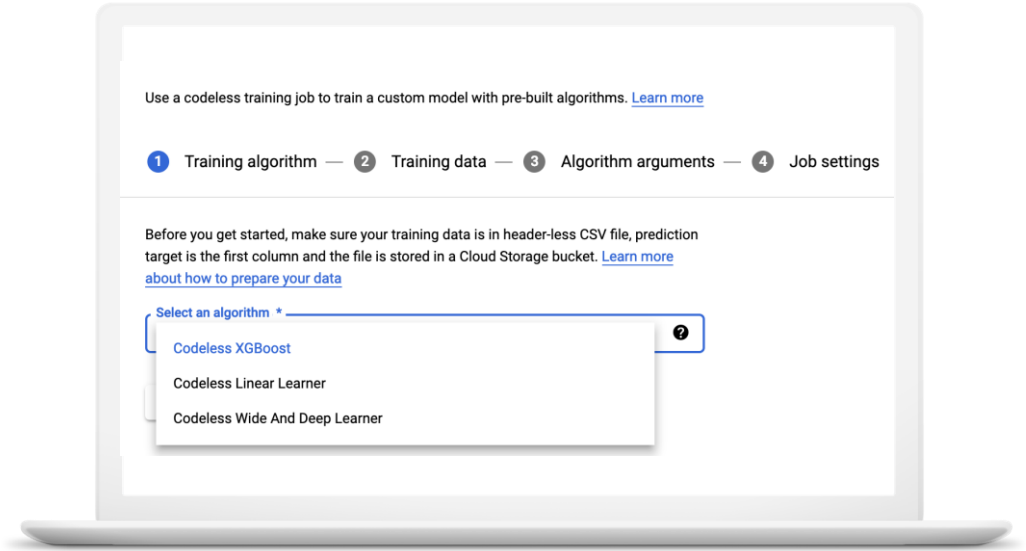
Simples modificações para treinar seu modelo on-premises com Kubeflow ou na Google Cloud using AI Platform

Implemente seu modelo com facilidade

```
%%bash -s "$SAVED_MODEL_DIR"
sm_dir=$1
model_name=kg_cancer_detection
version_name=versions


# Create the model and base version if it doesn't exist
gcloud ml-engine models describe $model_name > /dev/null
if [ $? -ne 0 ]; then gcloud ml-engine models create $model_name; fi
gcloud ml-engine versions describe $version_name --model $model_name 2> /dev/null
if [ $? -ne 0 ]; then
    gcloud ml-engine versions create $version_name \
      --model $model_name \
      --origin=$sm_dir \
fi
```

Modelos Built-in no Cloud AI Platform

- Otimização de hiperparâmetros embutida
- Suporte à algorítimos populares
- Busque soluções no AI Hub
- Facilidade para incluir novos algorítimos como containers



Três caminhos de AI na Google Cloud

seus dados + seu modelo


Cloud TPUs


Compute Engine


Cloud Dataproc

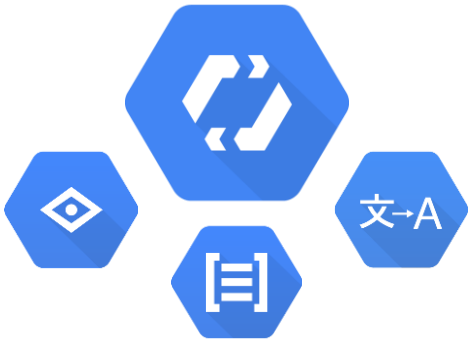

Kubernetes Engine


Cloud ML Engine


BigQuery ML

seus dados + nossos modelos

AutoML



nossos dados + nossos modelos


Cloud Translation API


Cloud Vision API


Cloud Speech API


Cloud Video Intelligence API


Data Loss Prevention API


Cloud Speech Synthesis API


Cloud Natural Language API


Dialogflow

Customização

Facilidade de Uso

Construa seus modelos

Treine os nossos modelos

Use nossas APIs inteligentes

BigQuery ML Overview

Google Cloud

BigQuery ML

BigQuery ML empowers both **data analysts** and **data scientists**

Execute ML initiatives without moving data from BigQuery

Iterate on models in SQL in BigQuery to increase development speed

Automate model selection, and hypertuning

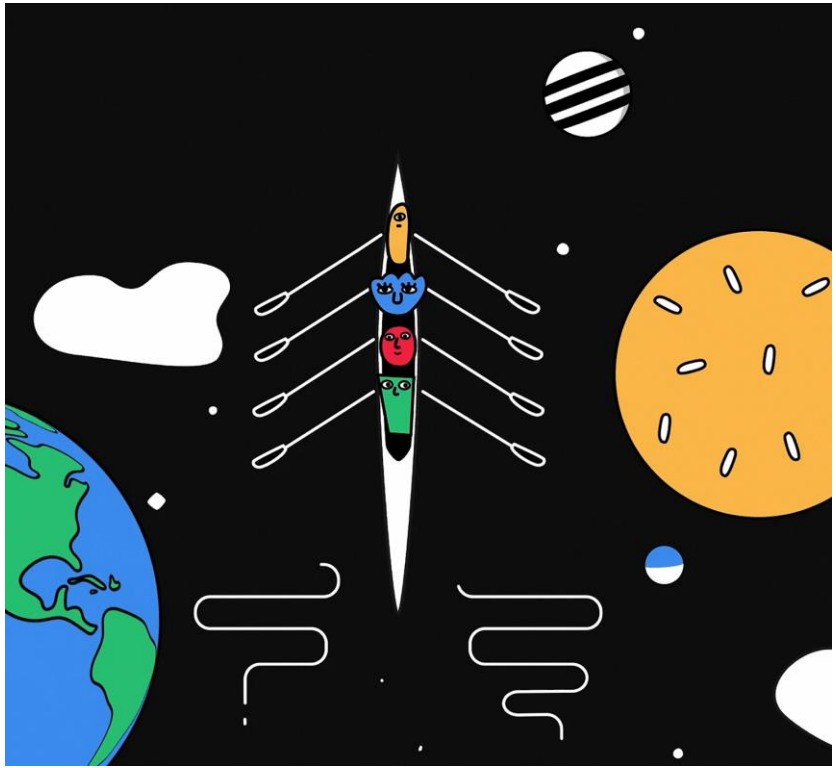
BigQuery ML

1. **Execute** ML initiatives without moving data from BigQuery
2. **Iterate** on models in SQL in BigQuery to increase development speed
3. **Automate** common ML tasks and hyperparameter tuning

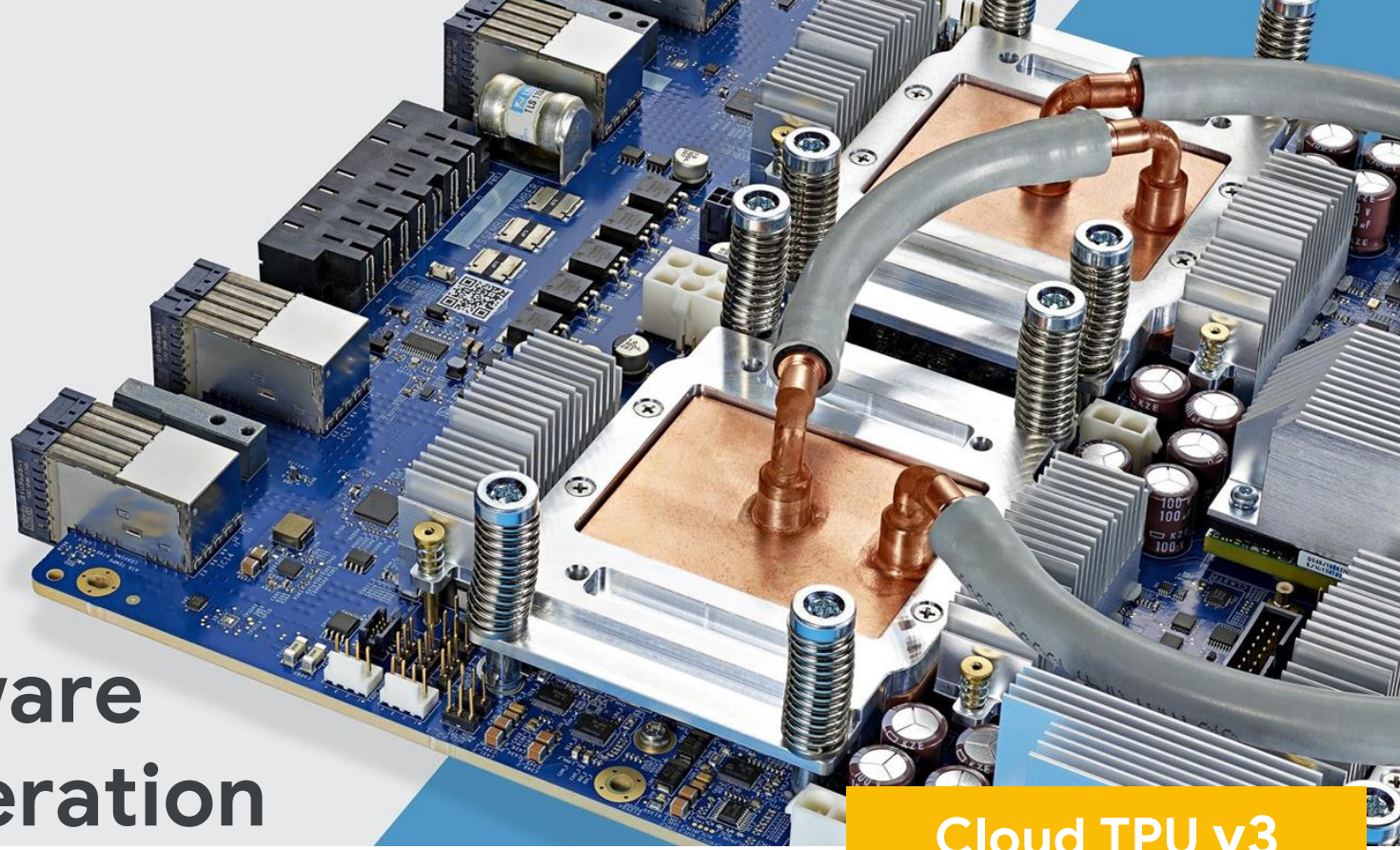
Behind the scenes

Through two lines of SQL

- Leverage BigQuery's processing power to build a model
- Auto-tuned learning rate
- Auto-split of data into training and test
- Null imputation
- Standardization of numeric features
- One-hot encoding of strings
- Class imbalance handling



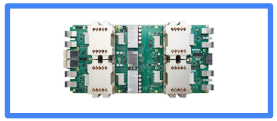
Hardware Acceleration



Cloud TPU v3

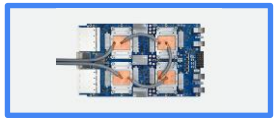
Accelerator families comparison

TPU v1
int8 only - inference only required model quantization



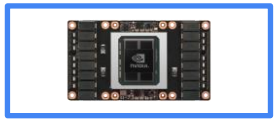
TPU v2
bfloat16 - inference and training no model changes necessary

180 TFLOPS (bfloat16)
/ board



TPU v3
bfloat16 - inference and training no model changes necessary

420 TFLOPS
(bfloat16)
/ board



NVIDIA P100
3584 CUDA cores - float16
but float16 x float16 => float32 not available

18 TFLOPS (float16)
/ chip

NVIDIA V100
5120 CUDA cores + 640 TensorCores - float16
model tweaking necessary for float16

112 TFLOPS (float16)
/ chip

Reference models for Cloud 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
& Language Modeling

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

Speech
Recognition

Models
ASR Transformer
(LibriSpeech)

Image
Generation

Models
Image Transformer
DCGAN

github.com/tensorflow/tpu

Training hardware options on AI Platform

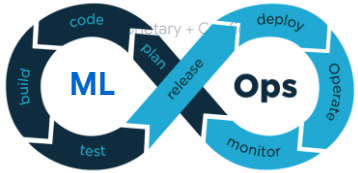
	Training Time	Cost
GPU - P100	5h50	$15
GPU - V100	4h30	$18
Cluster 5 GPUs - P100	1h15	$15
Cluster 5 GPUs - V100	1h00	$18
VM with 4 GPUs - V100	1h50	$27
Cloud TPU v2	**1h00**	**$6.6**

```
# config.yaml

trainingInput:
 scaleTier: BASIC_TPU
```

CLOUD TPU
8 CORES

Google Cloud

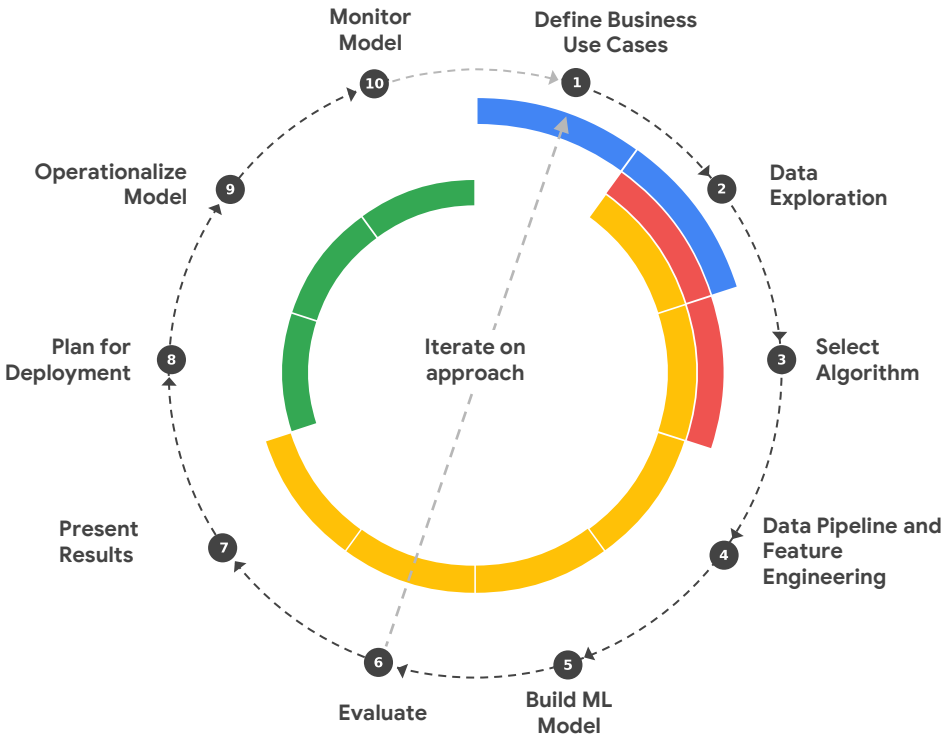
updated Jan 25th 2019, Tensorflow 1.12, time and cost to train model on same number of epochs with 4 evaluations per training, on ML Engine (TPU/GPU cost on raw VM is less)



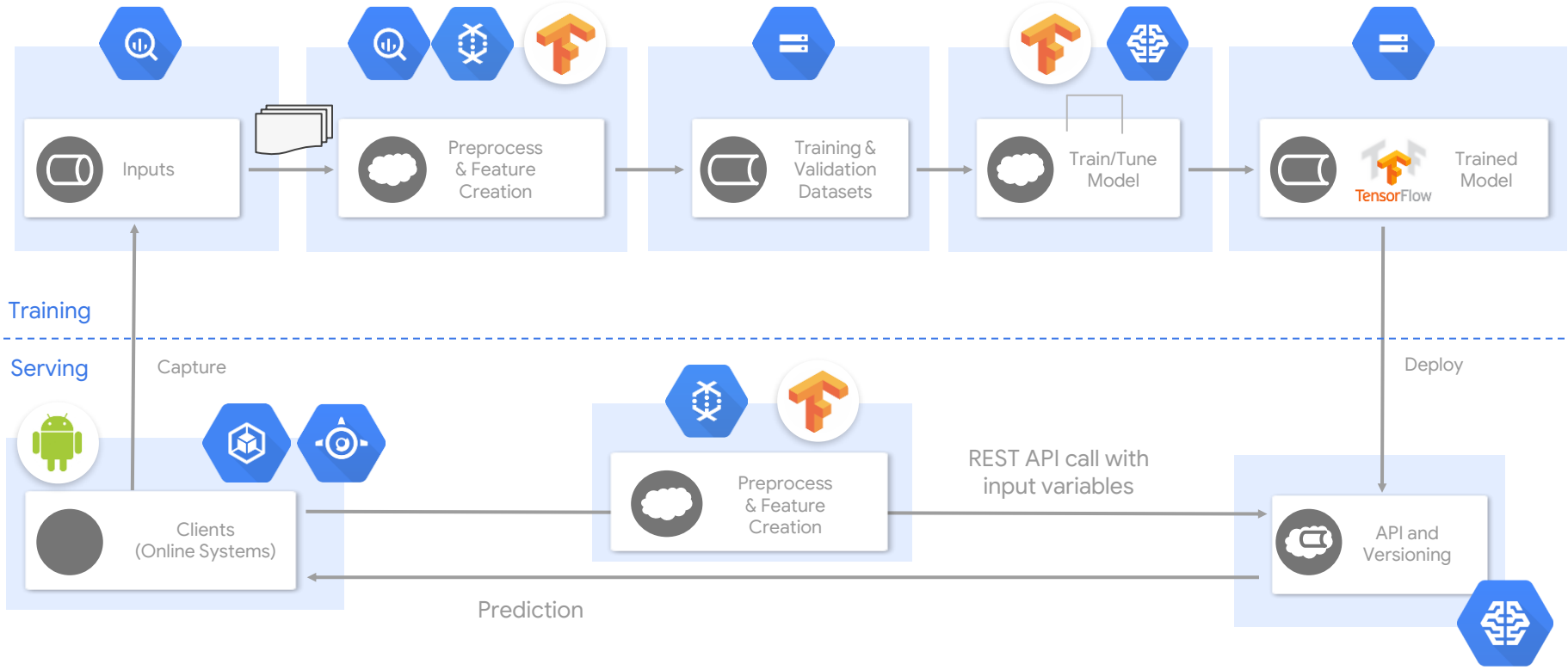
MLOps (ML Operationalization)

Typical machine learning lifecycle

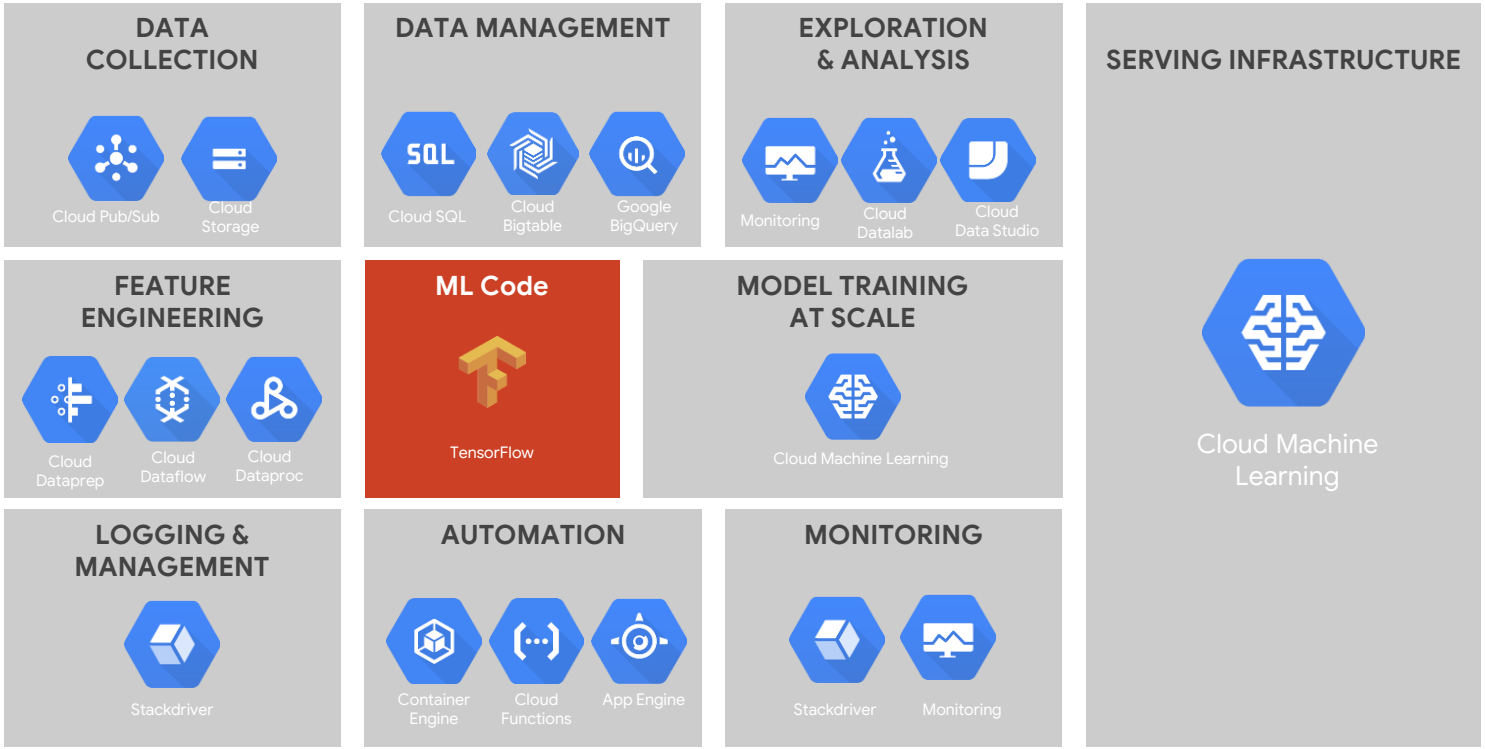
Step-by-step solution of ML problem



Machine learning pipeline @ GCP



Operational ML - end-to-end ML solution on GCP



We need cloud-native ML!

Slides taken from Google Cloud NEXT'18 talk (MLAI101)

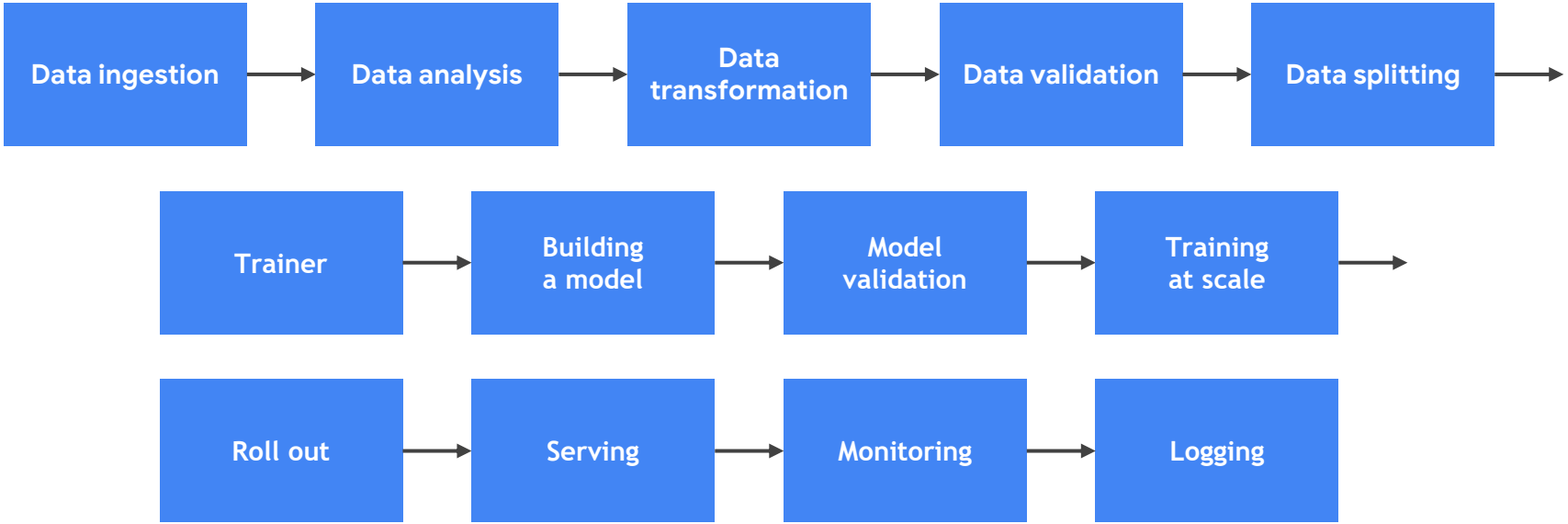
Google Cloud

Composability

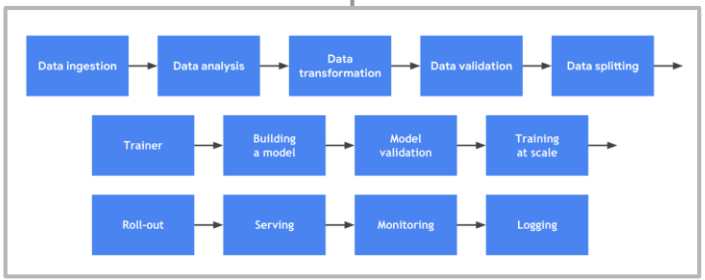
Portability

Scalability

Composability



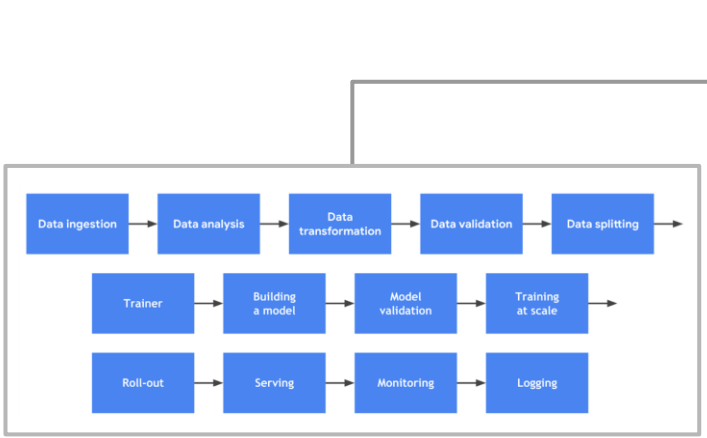
Portability



Experimentation

- Model
- UX
- Tooling
- Framework
- Storage
- Runtime
- Drivers
- OS
- Accelerator
- HW

Portability



Experimentation

Training

Cloud

You know what's really good at composability, portability and scalability?

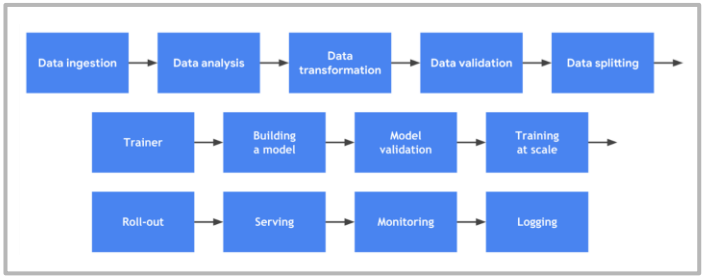
Containers & Kubernetes

Oh, you want to use ML on Kubernetes?

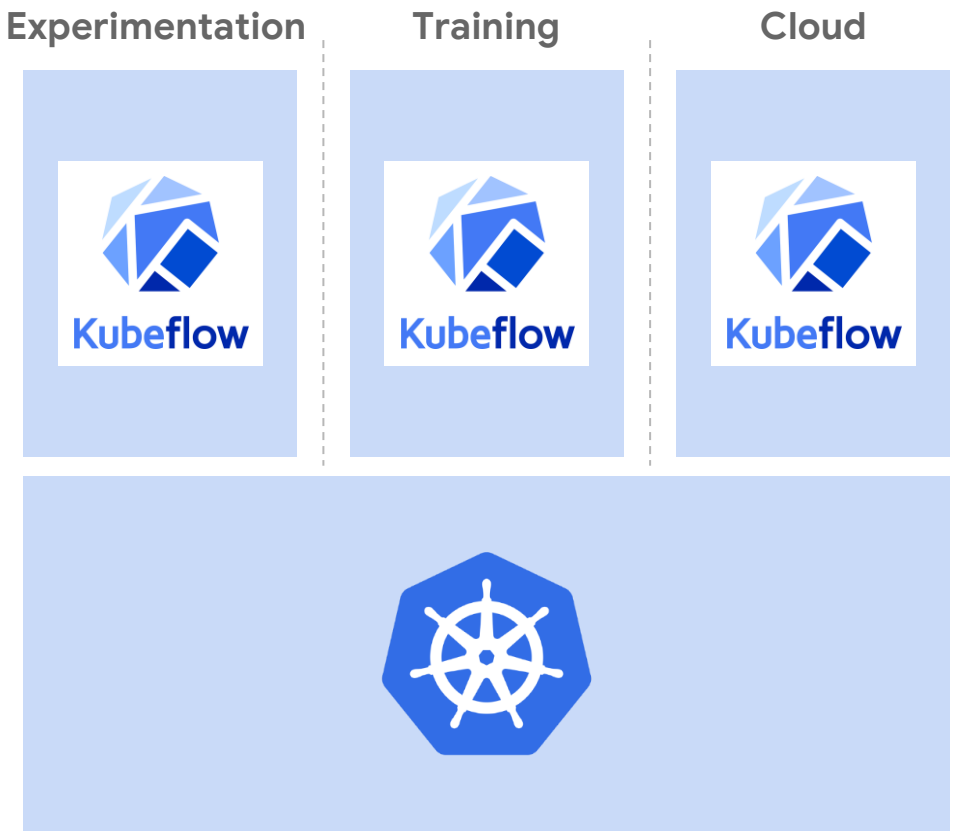
First, become an expert in:

- Containers
- Packaging
- Kubernetes service endpoints
- Persistent volumes
- Scaling
- Immutable deployments
- GPUs, drivers, and the GPL
- Cloud APIs
- DevOps
- ...

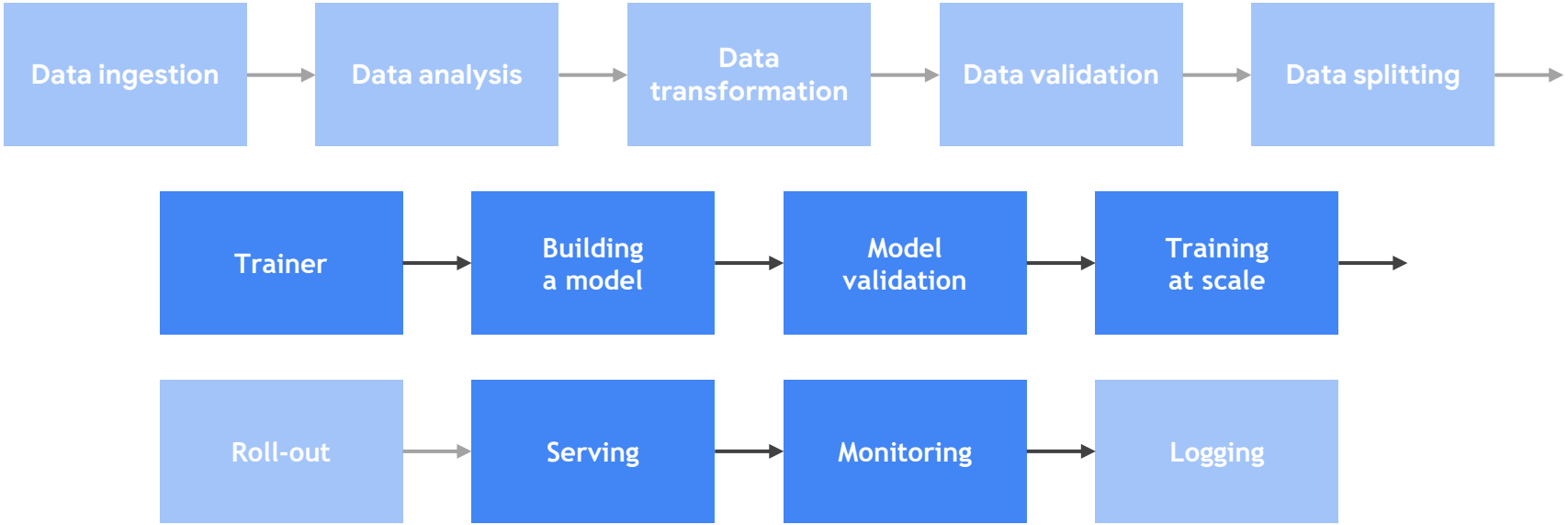
Portability



Portability



What's in the box?



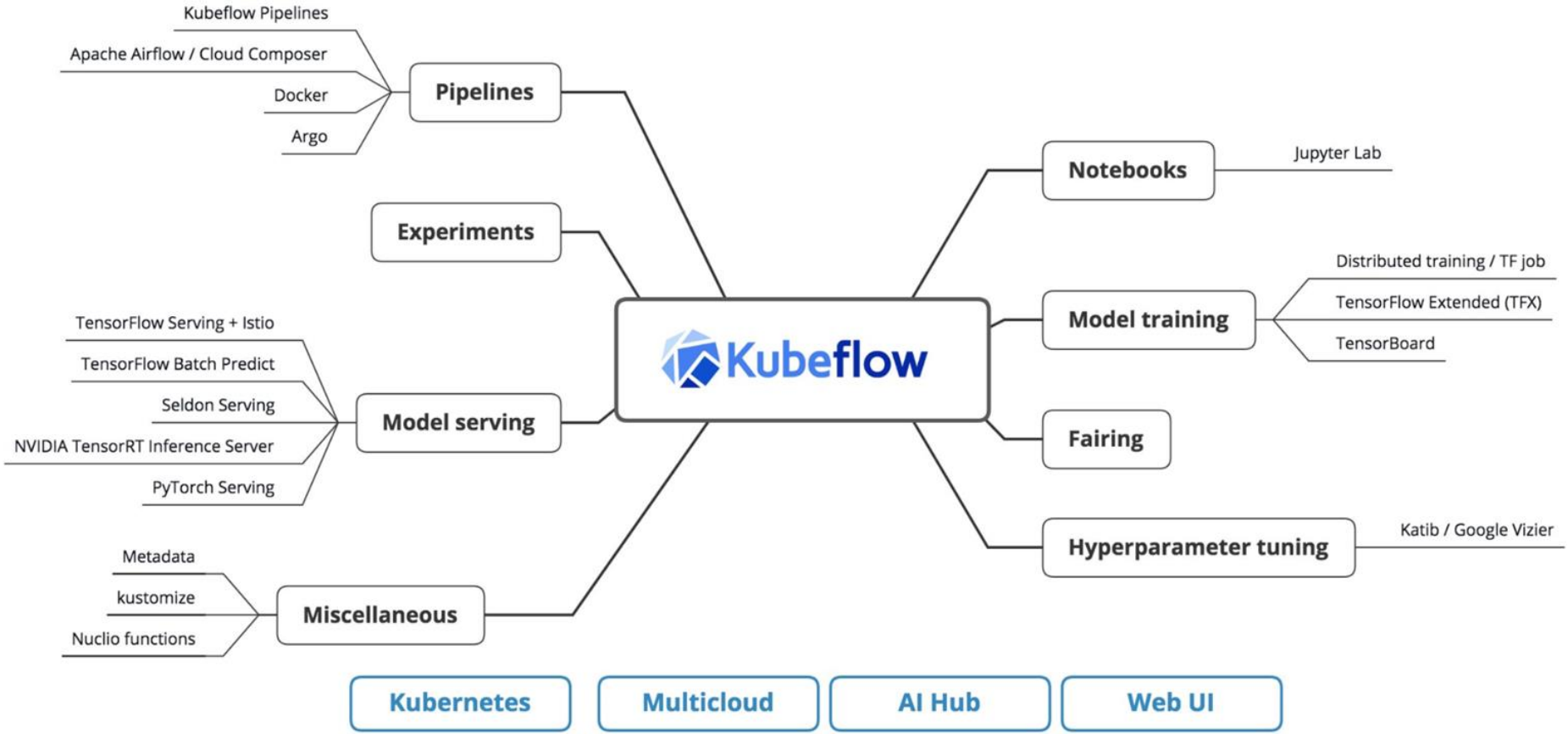
Kubeflow is open!

Open community

Open design

Open source

Open to ideas



Kubeflow Mission Statement

Makes it Easy for Everyone to Develop, Deploy and Manage a Portable, Distributed and Scalable ML system on Kubernetes

KubeFlow 1.0 GA Release	
Stable Components	**Beta Components**
Central Dashboard, kfctl, Profile Controller, Notebooks, TFServing, Training Operators (Tensorflow, Pytorch), Docs	Pipelines, Metadata Store, HP Tuning, KFServing, Fairing, Kale, Training Operators(XGBoost, MxNet)

Code: Today

Local

```
import xgboost

class MyModel(object):
    def train(self):
        # load data
        # do feature engineering
        # train a model

    def predict():
        # prediction logic

if __name__ == '__main__':
    model = MyModel()
    model.train()
```

Build & Deploy to AI Platform

Training

```
gcloud ml-engine jobs submit training my_job \
          --module-name trainer.task \
          --staging-bucket gs://my-bucket \
          --package-path /my/code/path/trainer \
          --packages additional-dep1.tar.gz,dep2.whl
```

Prediction

```
gcloud alpha ml-engine versions create
{VERSION_NAME} --model {MODEL_NAME} \ --origin
gs://{BUCKET}/{MODEL_DIR}/ \ --runtime-version
{RUNTIME_VERSION} \ --package-uris
gs://{BUCKET}/{PACKAGES_DIR}/my_package-0.2.tar.gz \
--model-class=my_model.ModelExample
```

Build & Deploy to Kubeflow

```
apiVersion: kubeflow.org/v1alpha2
kind: TFJob
metadata:
  labels:
    experiment: experiment10
  name: tfjob
  namespace: kubeflow
spec:
  tfReplicaSpecs:
    Ps:
      replicas: 1
      template:
        metadata:
          creationTimestamp: null
        spec:
          containers:
          - args:
            - python
            - tf_cnn_benchmarks.py
            image:
.
.
.
.
```

Code: With Kubeflow Fairing

Local

```
import xgboost

class MyModel(object):
    def train(self):
       # load data
       # train a model

    def predict():
       # prediction logic

from fairing import TrainJob
from fairing.backends import Backend

job = TrainJob(MyModel,
    backend=Backend("Local",
      "fairing.config"))
job.submit()

endpoint = PredictionEndpoint(MyModel,
    backend=Backend("Local",
      "fairing.config"))
endpoint.create()
```

Build & Deploy to AI Platform

```
import xgboost

class MyModel(object):
    def train(self):
       # load data
       # train a model

    def predict():
       # prediction logic

from fairing import TrainJob
from fairing.backends import Backend

job = TrainJob(MyModel,
    backend=Backend("ai_platform",
      "fairing.config"))
job.submit()

endpoint = PredictionEndpoint(MyModel,
    backend=Backend("ai_platform",
      "fairing.config"))
endpoint.create()
```

Build & Deploy to Kubeflow

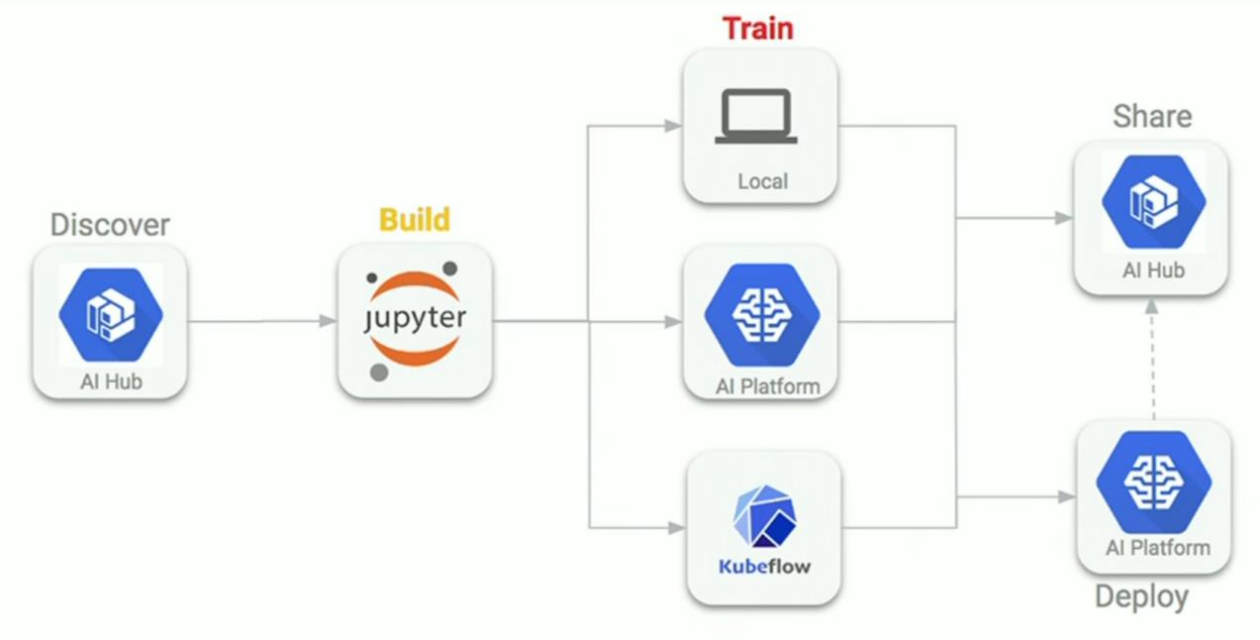
```
import xgboost

class MyModel(object):
    def train(self):
       # load data
       # train a model

    def predict():
       # prediction logic

from fairing import TrainJob
from fairing.backends import Backend

job = TrainJob(MyModel,
    backend=Backend("Kubeflow",
      "fairing.config"))
job.submit()

endpoint = PredictionEndpoint(MyModel,
    backend=Backend("Kubeflow",
      "fairing.config"))
endpoint.create()
```

Kubeflow Fairing

An open source Hybrid ML SDK for data scientists to ´write ML code once and run anywhere´

- **Data scientist focused:** Simple and uses language familiar Data Scientists
- **Multi-platform:** Supports AI Platform and Kubleflow, making it easy for users to switch between on-prem and GCP
- **Scalable and cost-effective:** Data Scientists can easily burst onto GCO when they need more resources (i.e. more machines, GPUs, or TPUs)
- **Easily train, tune, and deploy models:** Supports the full ML lifecycle
- **Multi-framework:** Supports XGBoost, TensorFlow (single node) and Pytorch (single node)

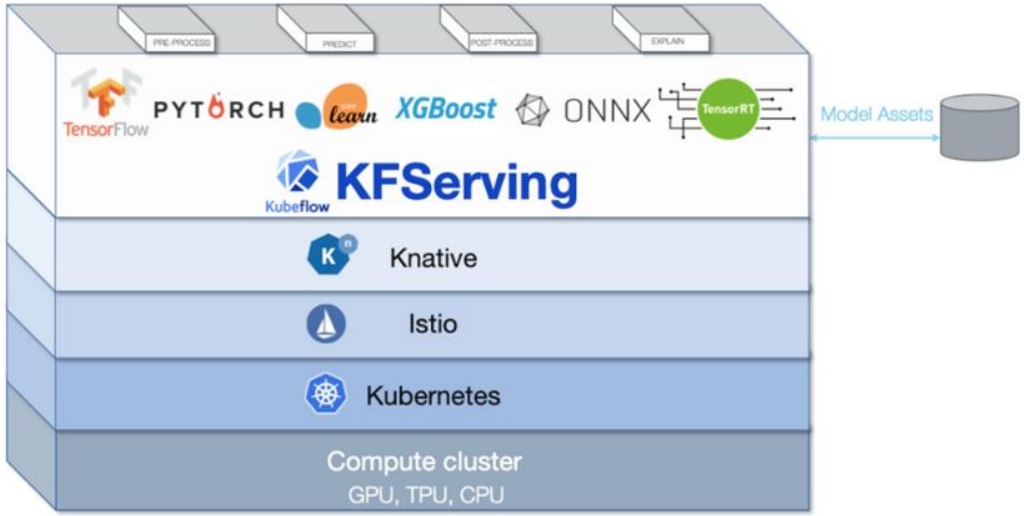
Sample: Hybrid E2E ML Kubeflow Fairing



KFServing

- Scalable, Kubernetes-native intererencing

- Usage pattern:
  - High Availability
  - Quick addition of capacity
  - Potentially need GPUs



A Clean Interface

```yaml
apiVersion: "serving.kubeflow.org/v1alpha1"
kind: "InferenceService"
metadata:
  name: "sklearn-iris"
spec:
  default:
    sklearn:
      storageUri: "gs://kfserving-samples/models/sklearn/iris"
```


```yaml
apiVersion: "serving.kubeflow.org/v1alpha1"
kind: "InferenceService"
metadata:
  name: "flowers-sample"
spec:
  default:
    tensorflow:
      storageUri: "gs://kfserving-samples/models/tensorflow/flowers"
```


```yaml
apiVersion: "serving.kubeflow.org/v1alpha1"
kind: "KFService"
metadata:
  name: "pytorch-cifar10"
spec:
  default:
    pytorch:
      storageUri: "gs://kfserving-samples/models/pytorch/cifar10"
      modelClassName: "Net"
```

CORE

1. Containerise
2. Deploy
3. Monitor

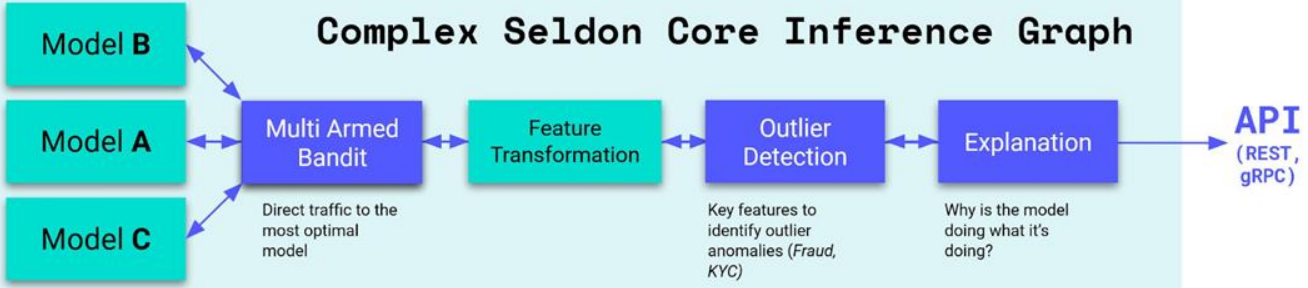
From model binary

Or language wrapper

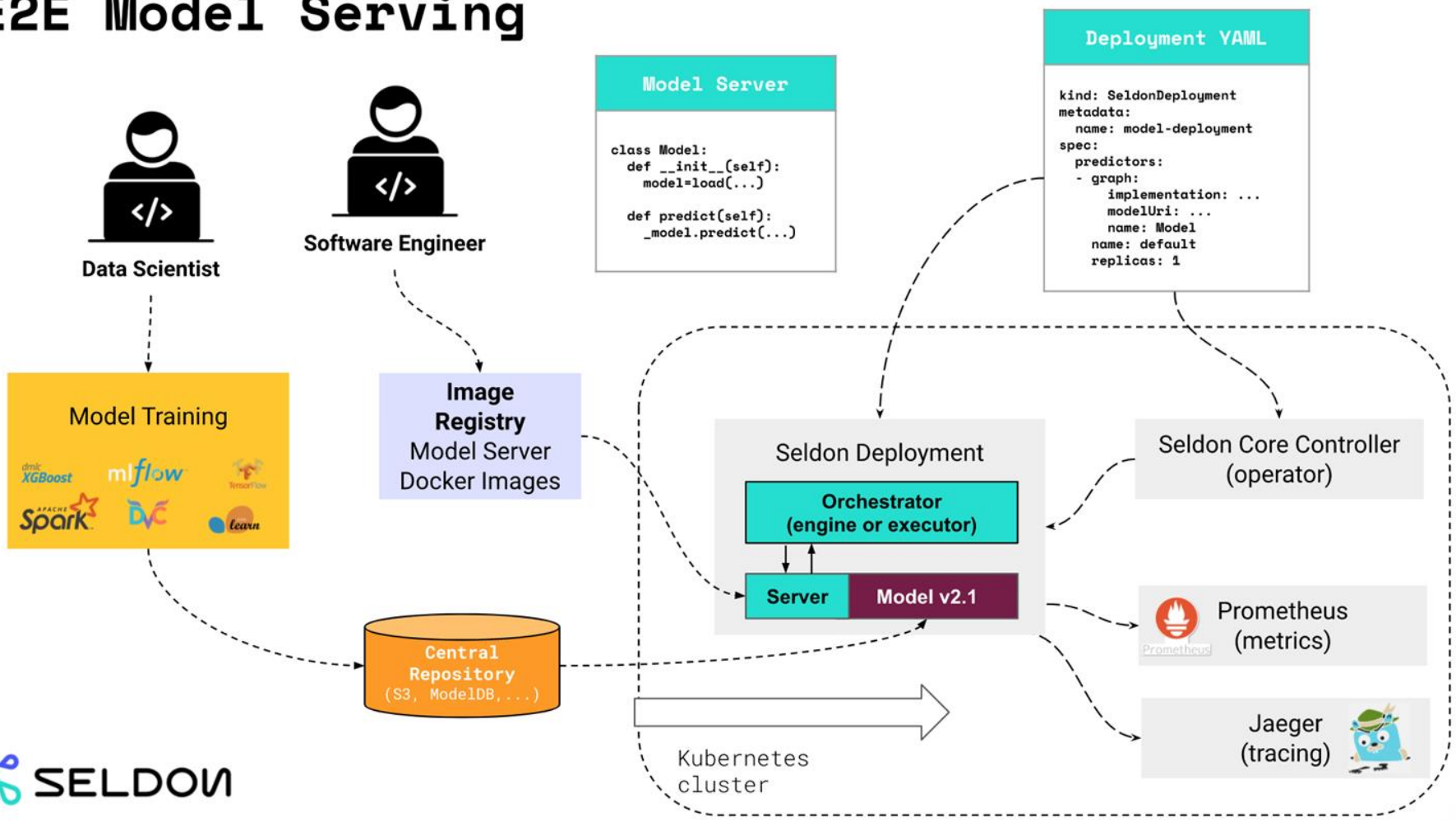
Into fully fledged microservice

Simple Seldon Core Inference Graph

Complex Seldon Core Inference Graph



E2E Model Serving



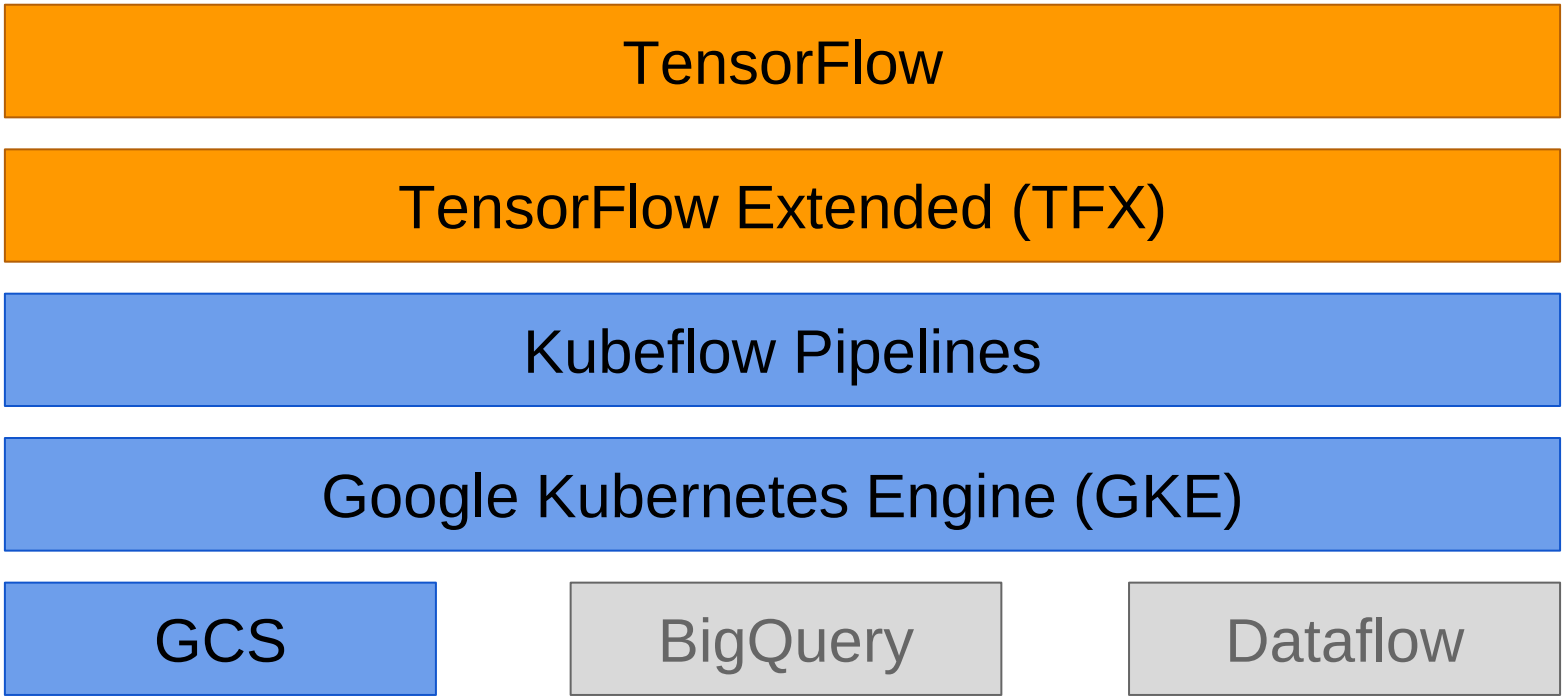
Production ML Pipelines in the Cloud

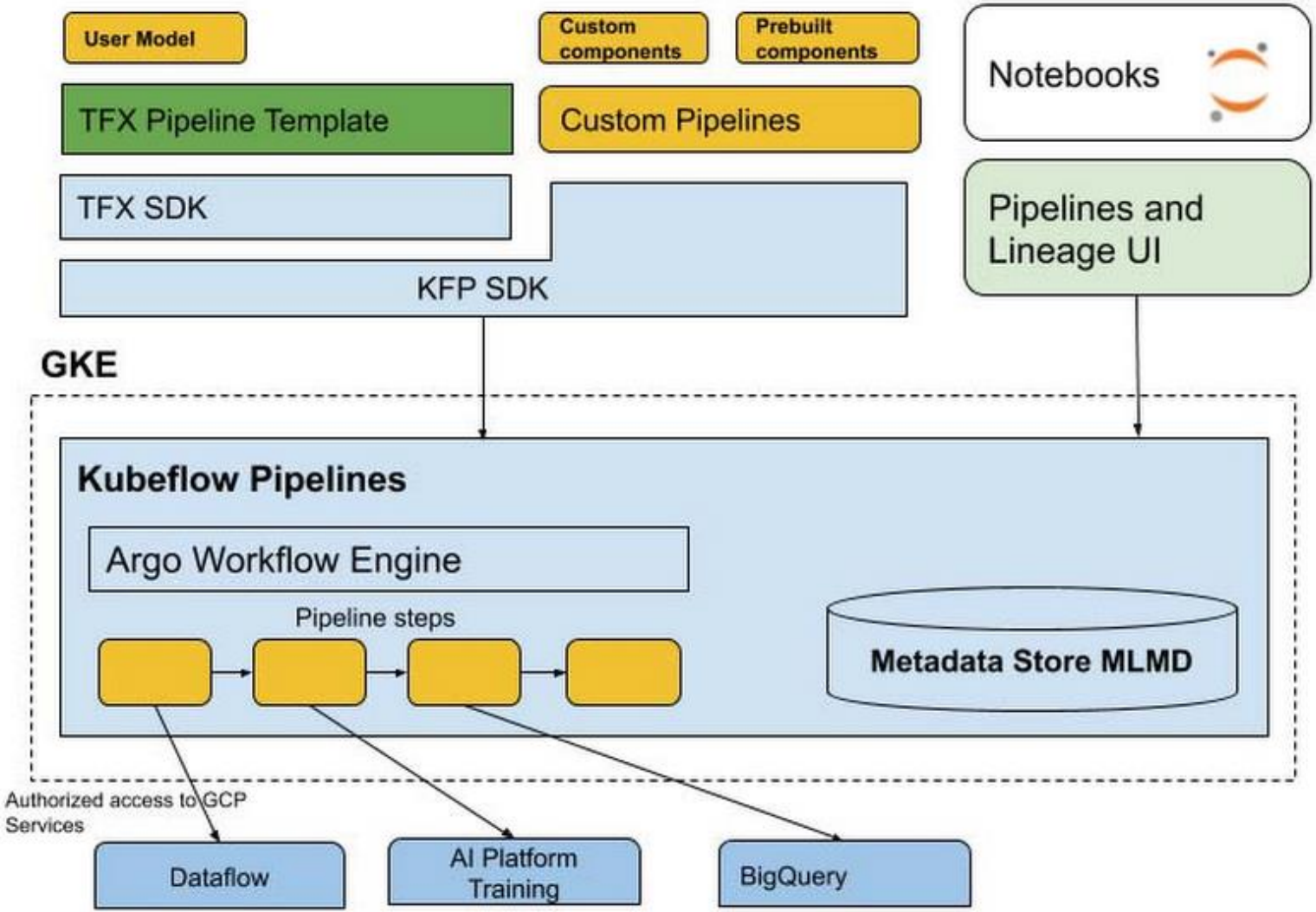
TensorFlow Extended

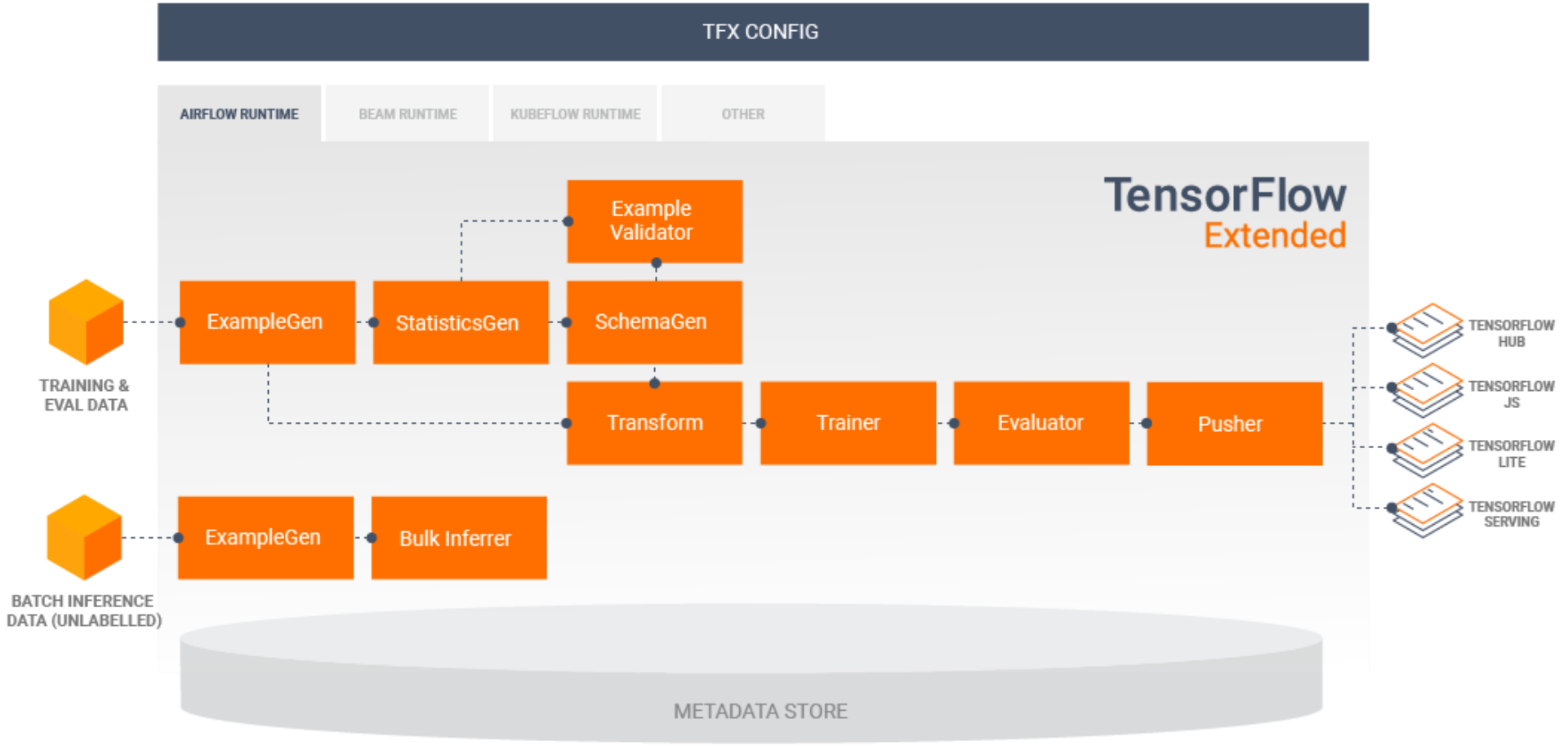
What we're doing

- We'll build a production-style end-to-end machine learning pipeline for training models
- We'll build it with TensorFlow Extended (TFX) on Google Cloud AI Platform

Very High Level Architecture




User Model
Custom components
Prebuilt components
Notebooks
TFX Pipeline Template
Custom Pipelines
TFX SDK
KFP SDK
Pipelines and Lineage UI
GKE
Kubeflow Pipelines
Argo Workflow Engine
Pipeline steps
Metadata Store MLMD
Authorized access to GCP Services
Dataflow
AI Platform Training
BigQuery


TFX CONFIG
AIRFLOW RUNTIME
BEAM RUNTIME
KUBEFLOW RUNTIME
OTHER
TensorFlow
Extended
Example Validator
ExampleGen
StatisticsGen
SchemaGen
TRAINING & EVAL DATA
Transform
Trainer
Evaluator
Pusher
TENSORFLOW HUB
TENSORFLOW JS
TENSORFLOW LITE
TENSORFLOW SERVING
ExampleGen
Bulk Inferrer
BATCH INFERENCE DATA (UNLABELLED)
METADATA STORE

Kubeflow Pipelines

A platform for building and deploying portable, scalable machine learning (ML) workflows based on Docker containers on Kubernetes.

Python SDK	Kubernetes runtime stack	Interactive UI
User-facing library for defining components and pipelines	Control plane server with Kubernetes-native workflow orchestration engine based on Argo, including custom extension for scheduled (cron-like) workflows.	Custom UI for launching & viewing pipeline runs, experiments, comparing output artifacts with out-of-the-box visualization capabilities.

TFX End-to-End Example

Chicago Taxi Cab Dataset

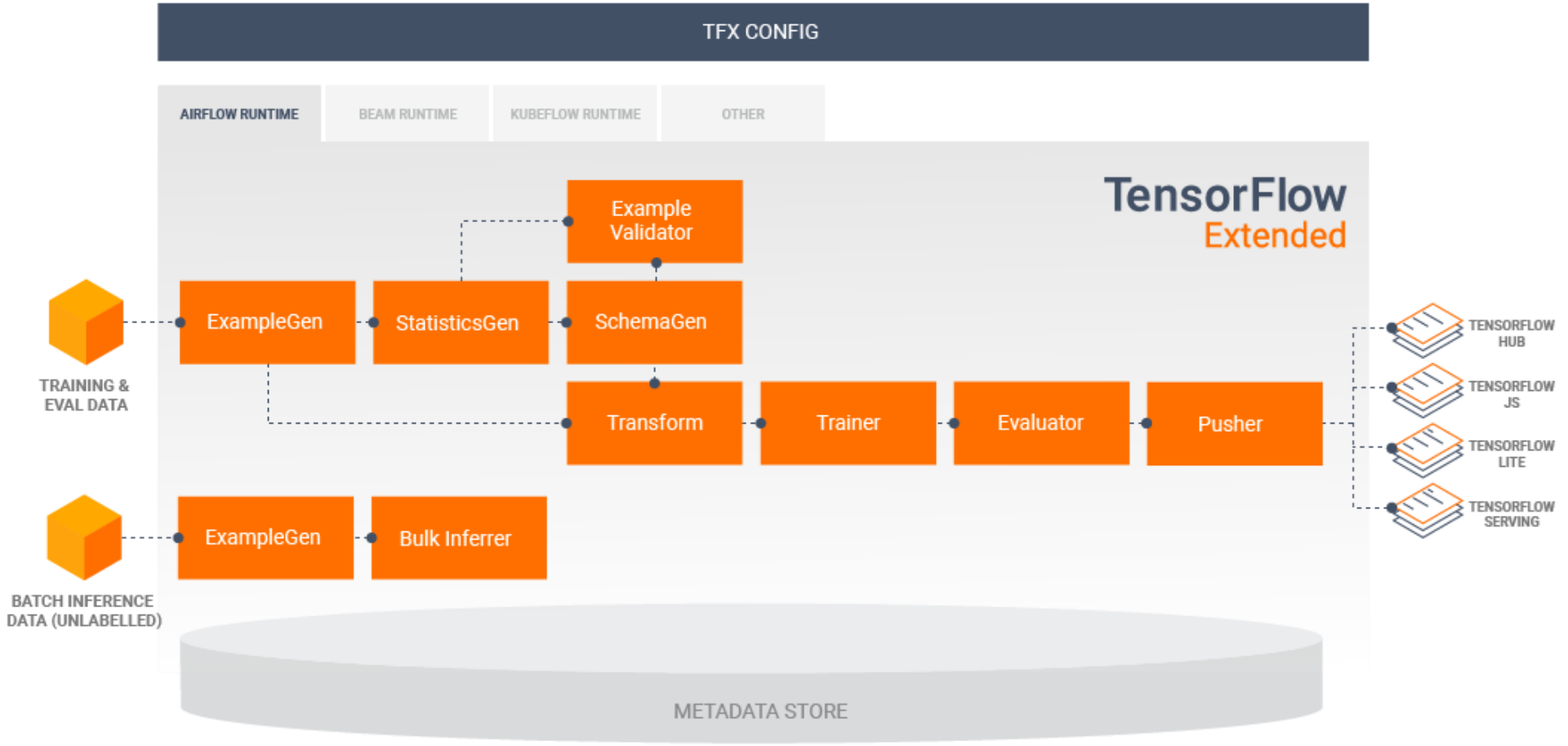
TFX End-to-End Example

Chicago Taxi Cab Dataset

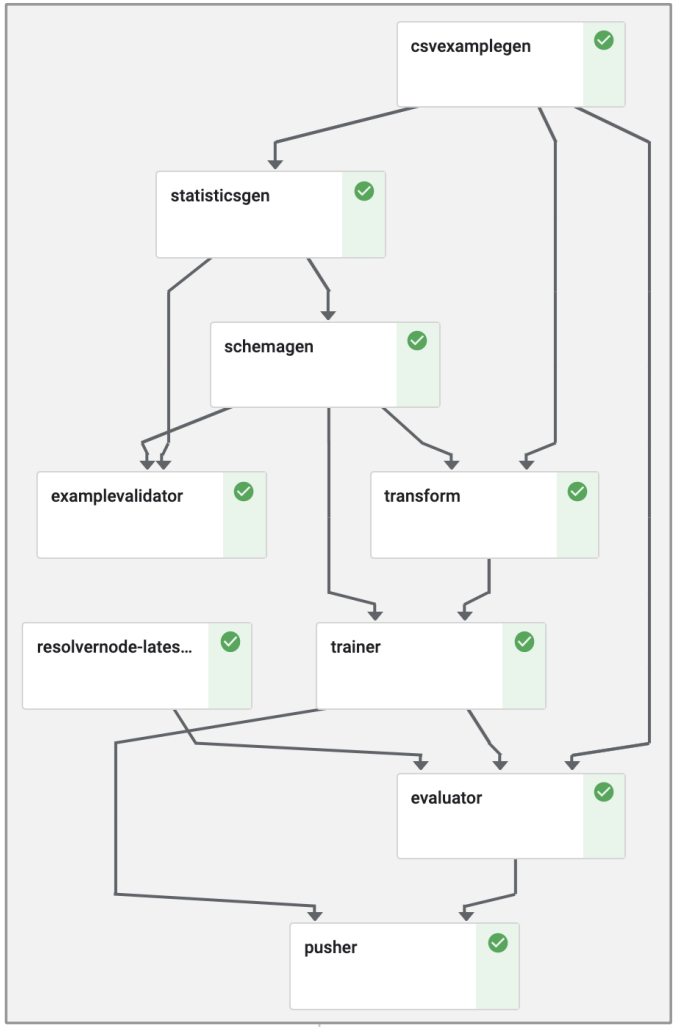
Features

Categorical Features	Bucket Features	Vocab Features	Dense Float Features
trip_start_hour	pickup_latitude	payment_type	trip_miles
trip_start_day	pickup_longitude	company	fare
trip_start_month	dropoff_latitude		trip_seconds
pickup/dropoff_census_tract	dropoff_longitude		
pickup/dropoff_community_area			

Label = tips > (fare * 20%)

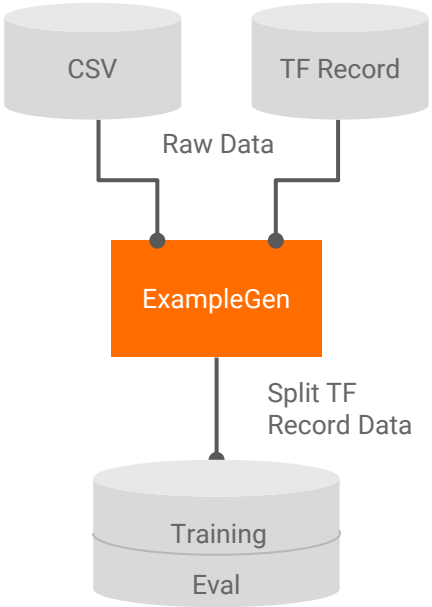

TFX CONFIG
AIRFLOW RUNTIME
BEAM RUNTIME
KUBEFLOW RUNTIME
OTHER
TensorFlow
Extended
Example Validator
ExampleGen
StatisticsGen
SchemaGen
TRAINING & EVAL DATA
Transform
Trainer
Evaluator
Pusher
TENSORFLOW HUB
TENSORFLOW JS
TENSORFLOW LITE
TENSORFLOW SERVING
ExampleGen
Bulk Inferrer
BATCH INFERENCE DATA (UNLABELLED)
METADATA STORE

TFX End-to-End Example



Component: ExampleGen

Inputs and Outputs



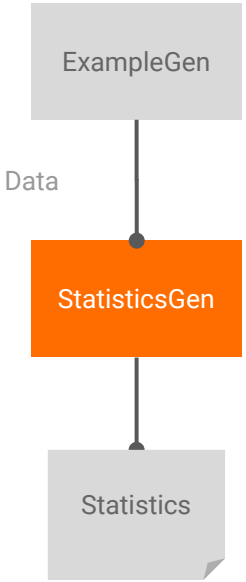
Configuration

```
examples = csv_input(os.path.join(data_root, 'simple'))

example_gen = CsvExampleGen(input_base=examples)
```

beam

Component: StatisticsGen

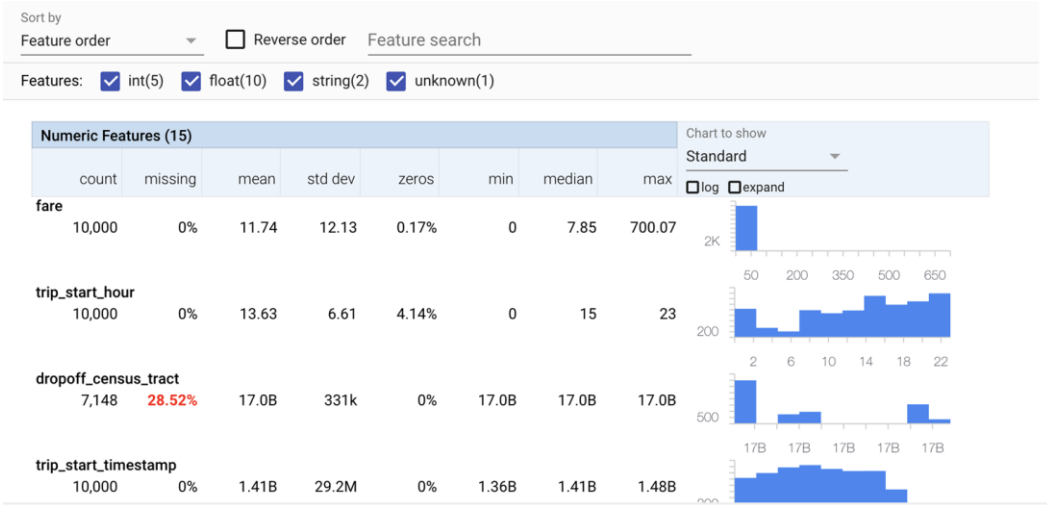
Inputs and Outputs



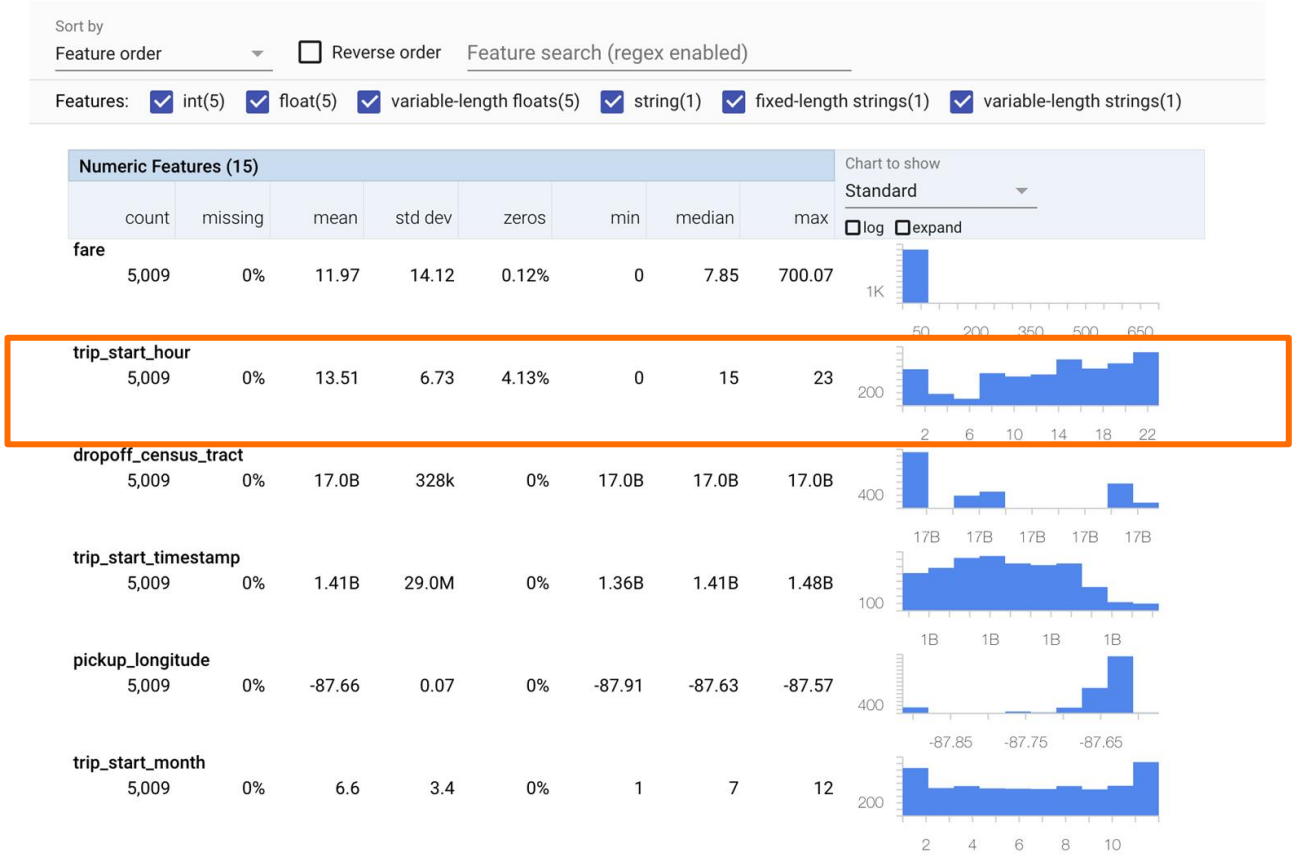
Configuration

```
statistics_gen =
  StatisticsGen(input_data=example_gen.outputs.examples)
```

Visualization

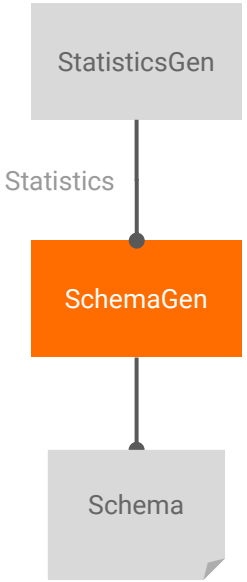


Analyzing Data with TensorFlow Data Validation



Component: SchemaGen

Inputs and Outputs



Configuration

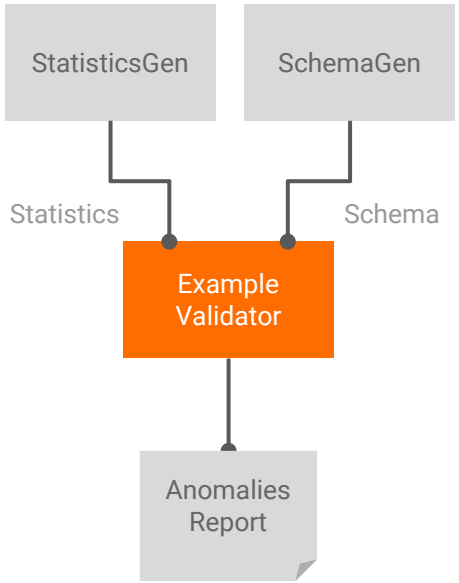
```
infer_schema = SchemaGen(stats=statistics_gen.outputs.output)
```

Visualization

Feature name	Type	Presence	Valency	Domain
'fare'	FLOAT	required	single	-
'trip_start_hour'	INT	required	single	-
'pickup_census_tract'	BYTES	optional		-
'dropoff_census_tract'	FLOAT	optional	single	-
'company'	STRING	optional	single	'company'

Component: ExampleValidator

Inputs and Outputs



Configuration

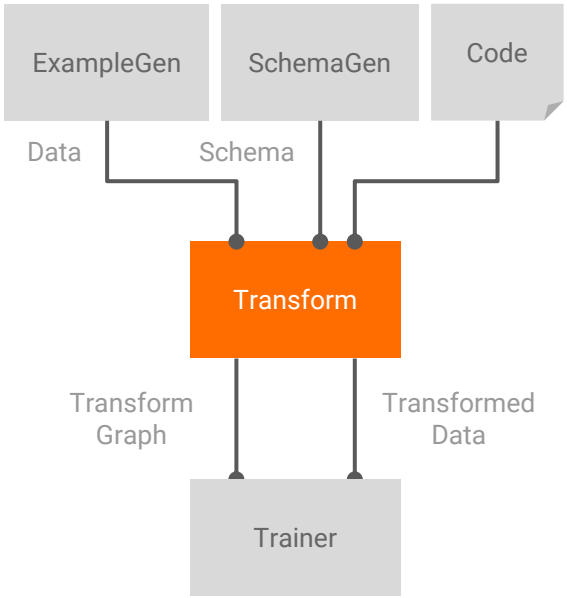
```
validate_stats = ExampleValidator(
    stats=statistics_gen.outputs.output,
    schema=infer_schema.outputs.output)
```

Visualization

Feature name	Anomaly short description	Anomaly long description
'payment_type'	Unexpected string values	Examples contain values missing from the schema: Prcard (<1%).
'company'	Unexpected string values	Examples contain values missing from the schema: 2092 - 61288 Sbeih company (<1%), 2192 - 73487 Zeymane Corp (<1%), 2192 - Zeymane Corp (<1%), 2823 - 73307 Seung Lee (<1%), 3094 - 24059 G.L.B. Cab Co (<1%), 3319 - CD Cab Co (<1%), 3385 - Eman Cab (<1%), 3897 - 57856 Ilie Malec (<1%), 4053 - 40193 Adwar H. Nikola (<1%), 4197 - Royal Star (<1%), 585 - 88805 Valley Cab Co (<1%), 5874 - Sergey Cab Corp. (<1%), 6057 - 24657 Richard Addo (<1%), 6574 - Babylon Express Inc. (<1%), 6742 - 83735 Tasha ride inc (<1%).

Component: Transform

Inputs and Outputs



beam

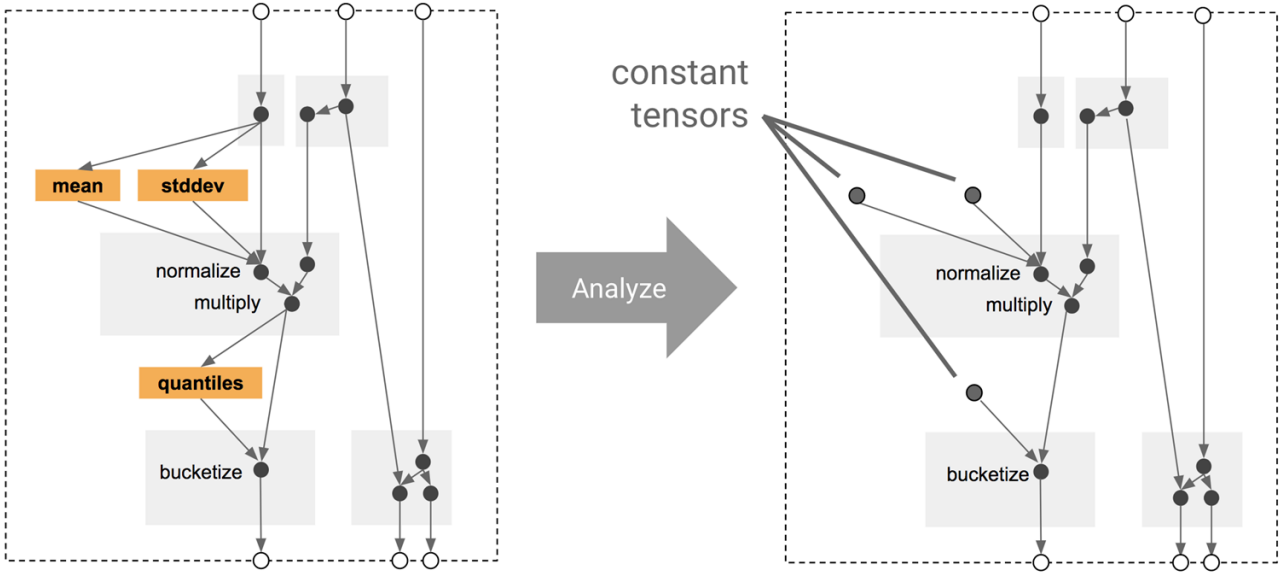
Configuration

```
transform = Transform(
    input_data=example_gen.outputs.examples,
    schema=infer_schema.outputs.output,
    module_file=taxi_module_file)
```

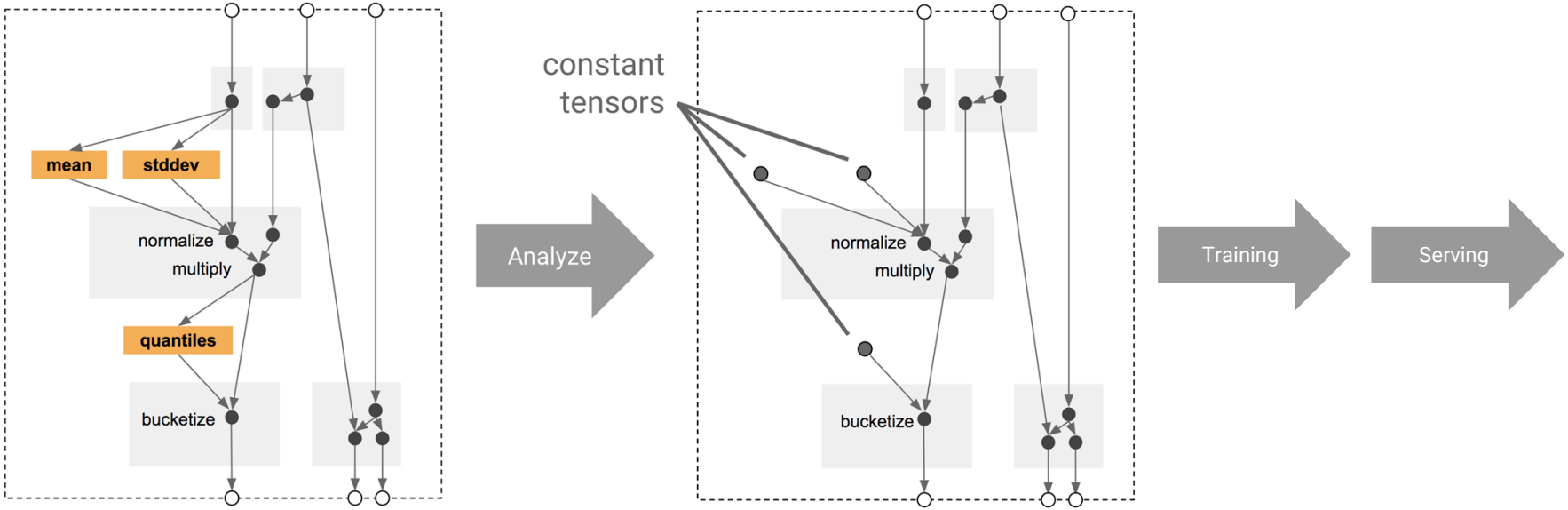
Code

```
for key in _DENSE_FLOAT_FEATURE_KEYS:
    outputs[_transformed_name(key)] = transform.scale_to_z_score(
        _fill_in_missing(inputs[key]))
# ...

outputs[_transformed_name(_LABEL_KEY)] = tf.where(
      tf.is_nan(taxi_fare),
      tf.cast(tf.zeros_like(taxi_fare), tf.int64),
      # Test if the tip was > 20% of the fare.
      tf.cast(
          tf.greater(tips, tf.multiply(taxi_fare, tf.constant(0.2))), tf.int64))

# ...
```

Using TensorFlow Transform for Feature Engineering

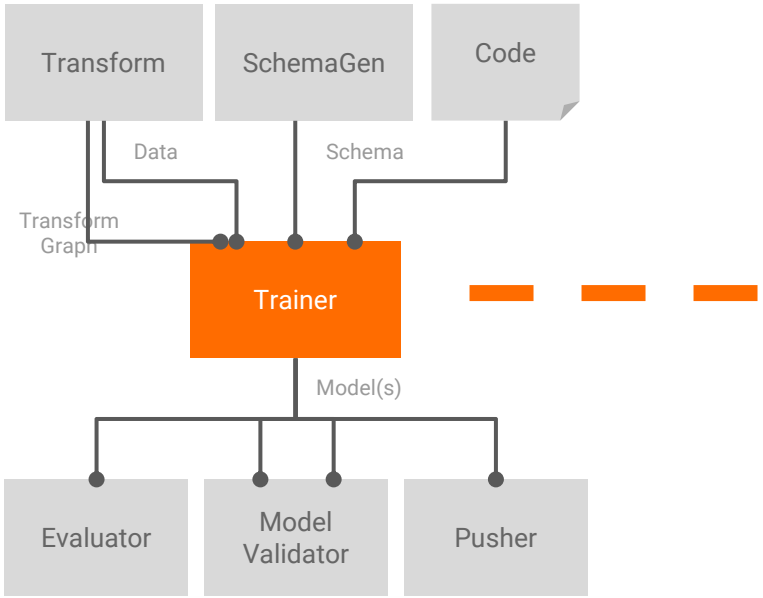


Using TensorFlow Transform for Feature Engineering



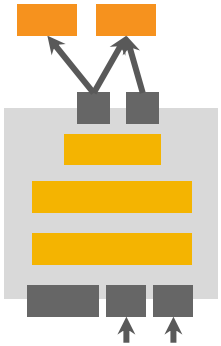
Component: Trainer

Inputs and Outputs



Highlight: SavedModel Format

Train, Eval, and Inference Graphs



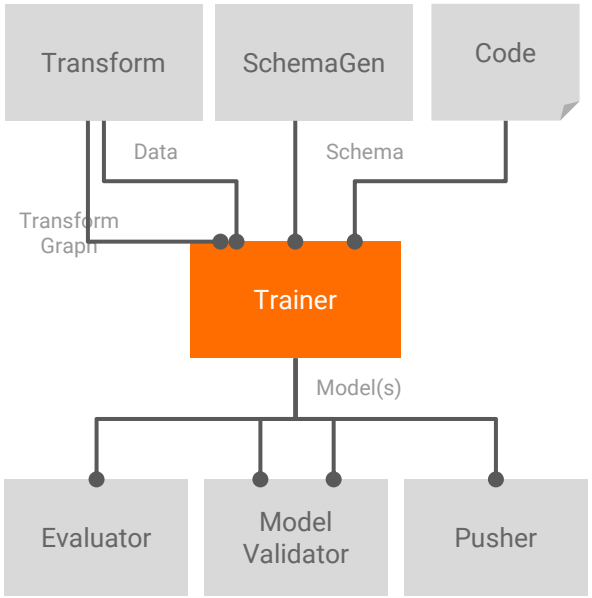
SavedModel

TensorFlow Model Analysis

TensorFlow Serving

Component: Trainer

Inputs and Outputs



Configuration

```
trainer = Trainer(
    module_file=taxi_module_file,
    transformed_examples=transform.outputs.transformed_examples,
    schema=infer_schema.outputs.output,
    transform_output=transform.outputs.transform_output,
    train_steps=10000,
    eval_steps=5000,
    warm_starting=True)
```

Code

Just TensorFlow :)

```
# Open up Tensorboard for model_id.
print(display_tensorboard(model_id))
```

```
http://your.host.name:53143
```

TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS PROJECTOR INACTIVE

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

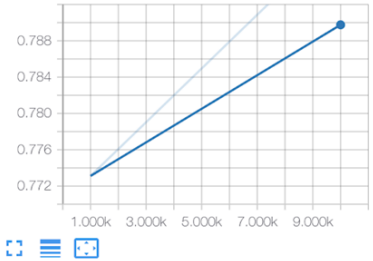
model_8/serving_model_dir

model_8/serving_model_dir/eval_chicago-taxi-eval

Filter tags (regular expressions supported)

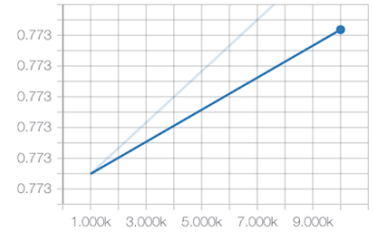
accuracy 1

accuracy



accuracy_baseline 1

accuracy_baseline



```
# Compare Tensorboard metrics for different models.
if num_models > 1:
    print(display_tensorboard(model_id, other_model_id=other_model_id))
```

```
http://your.host.name:53230
```

TensorBoard SCALARS GRAPHS DISTRIBUTIONS HISTOGRAMS PROJECTOR INACTIVE

Show data download links

Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing

0.6

Horizontal Axis

STEP RELATIVE WALL

Runs

Write a regex to filter runs

model_8/serving_model_dir

model_8/serving_model_dir/eval_chicago-taxi-eval

model_20/serving_model_dir

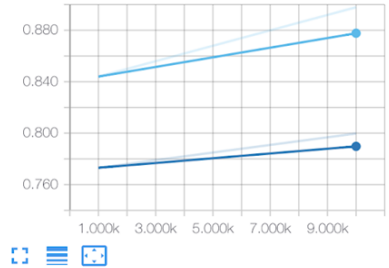
model_20/serving_model_dir/eval_chicago-taxi-eval

TOGGLE ALL RUNS

Filter tags (regular expressions supported)

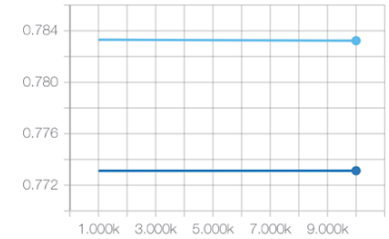
accuracy 1

accuracy



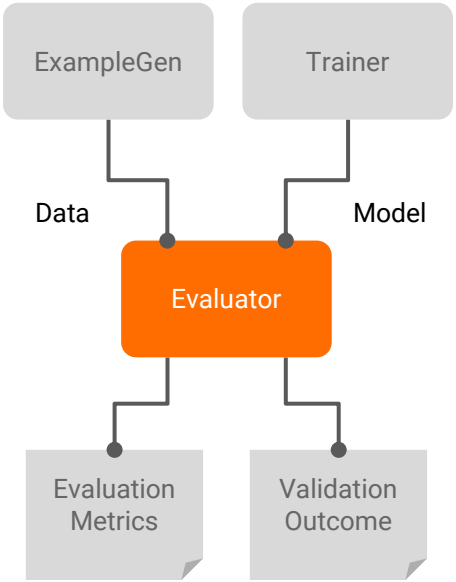
accuracy_baseline 1

accuracy_baseline



Component: Evaluator

Inputs and Outputs

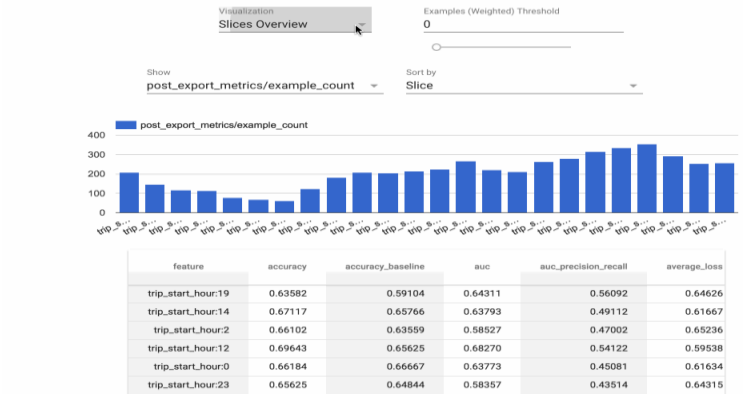


beam

Configuration

```
evaluator = Evaluator(
    examples=example_gen.outputs['examples'],
    model=trainer.outputs['model'],
    baseline_model=model_resolver.outputs['model'],
    eval_config=eval_config)
```

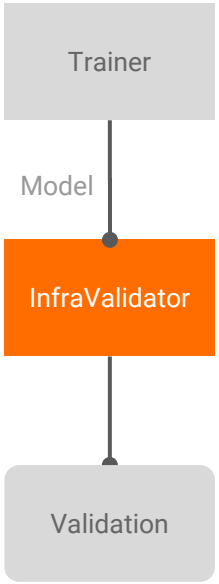
Visualization



feature	accuracy	accuracy_baseline	auc	auc_precision_recall	average_loss
trip_start_hour:19	0.63582	0.59104	0.64311	0.56092	0.64626
trip_start_hour:14	0.67117	0.65766	0.63793	0.49112	0.61667
trip_start_hour:2	0.66102	0.63559	0.58527	0.47002	0.65236
trip_start_hour:12	0.69643	0.65625	0.68270	0.54122	0.59538
trip_start_hour:0	0.66184	0.66667	0.63773	0.45081	0.61634
trip_start_hour:23	0.65625	0.64844	0.58357	0.43514	0.64315

Component: InfraValidator

Inputs and Outputs



Configuration

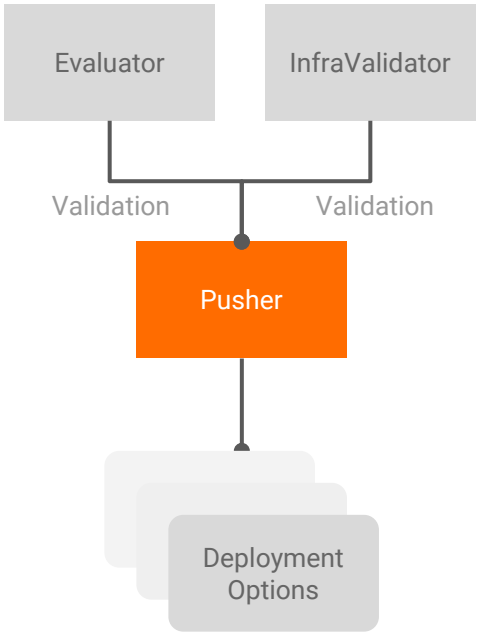
```
infra_validator = InfraValidator(
    model=trainer.outputs['model'],
    serving_spec=ServingSpec(...),
    validation_spec=ValidationSpec(...),
    request_spec=RequestSpec(...)
)
```

Configuration Options

ServingSpec: Type of model server and infrastructure to test with
ValidationSpec: Adjusts the infra validation criteria or workflow
RequestSpec: Which model signature, how many examples to test

Component: Pusher

Inputs and Outputs



Configuration

```
pusher = Pusher(
    model=trainer.outputs['model'],
    model_blessing=evaluator.outputs['blessing'],
    infra_blessing=infra_validator.outputs['blessing'],
    push_destination=pusher_pb2.PushDestination(
        filesystem=pusher_pb2.PushDestination.Filesystem(
            base_directory=_serving_model_dir)))
```

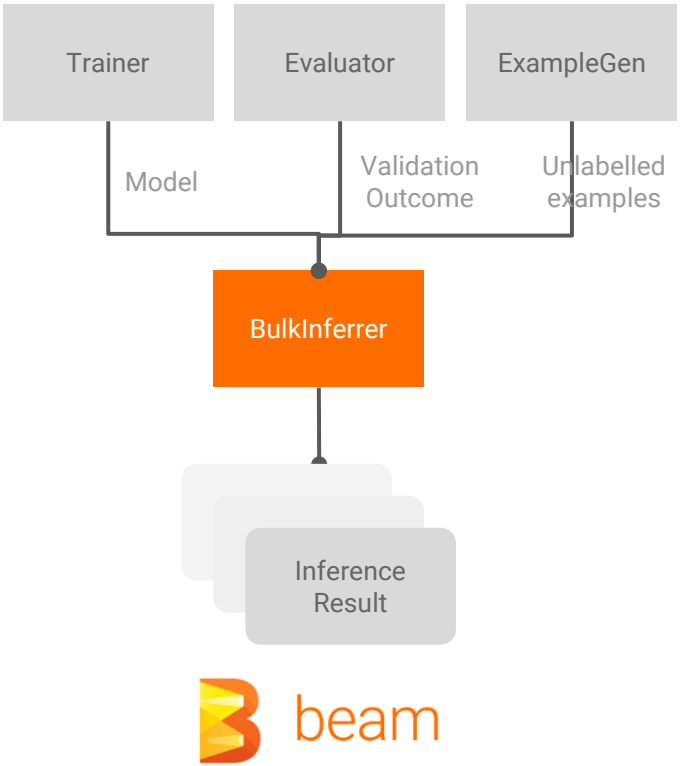
Block push on validation outcome

Push destinations supported today

- Filesystem (TensorFlow Lite, TensorFlow JS)
- TensorFlow Serving

Component: BulkInferrer

Inputs and Outputs



Configuration

```
bulk_inferrer = BulkInferrer(
    examples=inference_example_gen.outputs['examples'],
    model_export=trainer.outputs['output'],
    model_blessing=evaluator.outputs['blessing'],
    data_spec=bulk_inferrer_pb2.DataSpec(
        example_splits=['unlabelled']),
    model_spec=bulk_inferrer_pb2.ModelSpec())
```

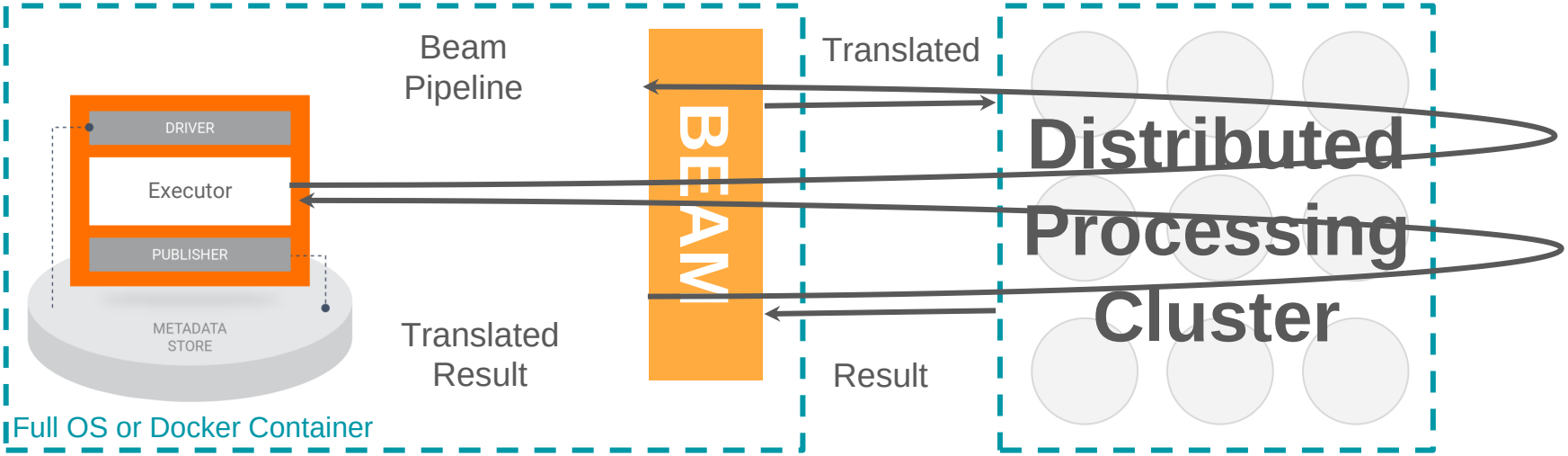
Configuration Options

Block batch inference on a successful model validation.
Choose the inference examples from example gen's output.
Choose the signatures and tags of inference model.

Inference Result

Contains features and predictions.

How TFX Components Use Beam



TFX Pipeline Nodes

What are Pipeline Nodes?

Pipeline nodes are special-purpose classes for performing advanced metadata operations

- Import external artifacts into ML-Metadata
- Perform queries of current ML Metadata based on artifact properties and history

ImporterNode

ImporterNode registers an external resource into MLMD so that downstream nodes can use the registered artifact as input

- Key requirements: source_uri, artifact_type

```
importer = ImporterNode(
      instance_name='import_schema',
      source_uri='uri/to/schema'
      artifact_type=standard_artifacts.Schema,
      reimport=False)
```

ResolverNode

ResolverNode is a special TFX node which components use to perform artifact resolution queries

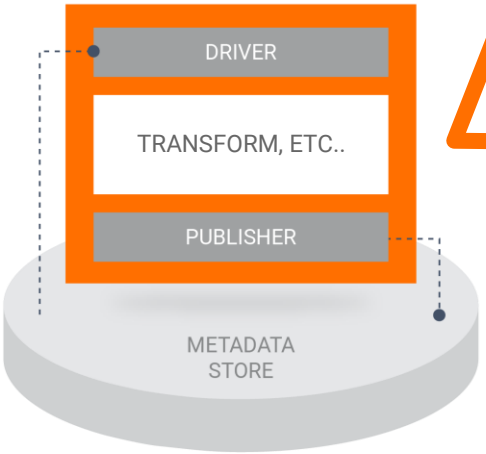
- Key requirements: Class, Query config, Channel

```
latest_five_examples_resolver = ResolverNode(
      instance_name='latest_five_examples_resolver',
      resolver_class=latest_artifacts_resolver.LatestArtifactsResolver, # Class
      resolver_config={'desired_num_of_artifacts' : 5},                 # Query config
      examples=example_gen.outputs['examples'])                         # Channel
```

Current ResolverNodes

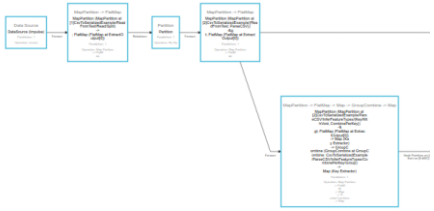
- LatestArtifactsResolver
  - Resolver that returns the latest n artifacts in a given channel
- LatestBlessedModelResolver
  - Returns the latest blessed model

Executors do the work

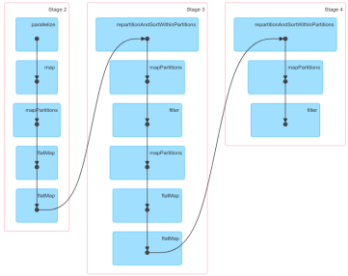


Beam

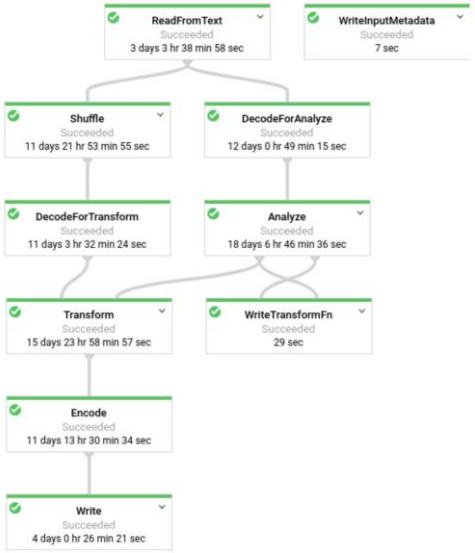
Flink



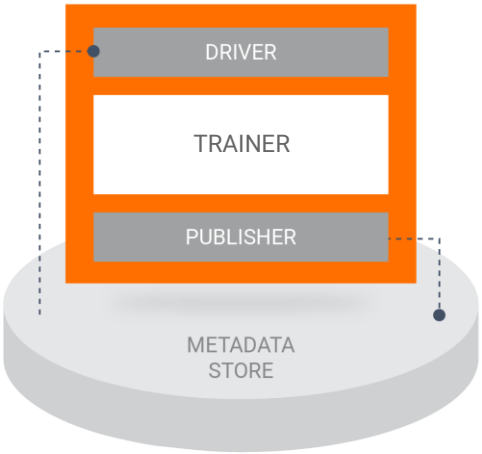
Spark

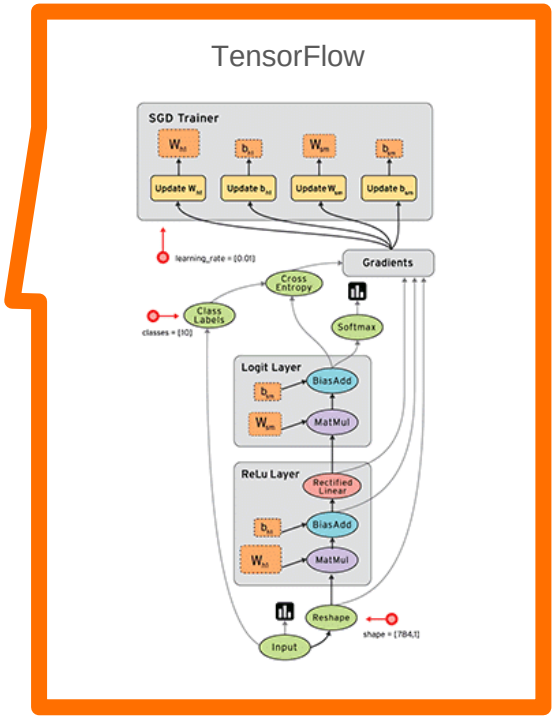


Dataflow



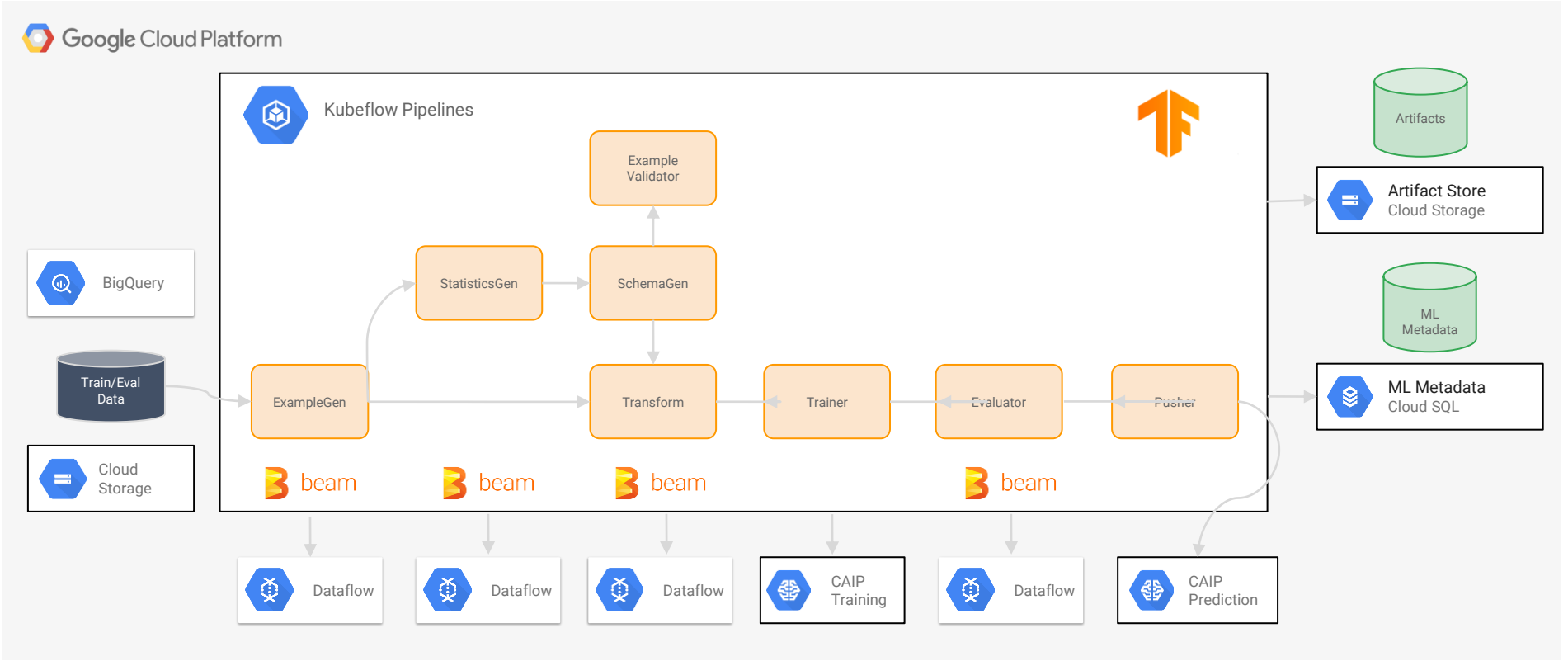
Executors do the work





Executors do the work

TFX in Cloud AI Platform



More Information

TFX Website: https://www.tensorflow.org/tfx

TFX Repo: https://github.com/tensorflow/tfx

TFX Community: https://goo.gle/tfx-group

TFX YouTube: https://goo.gle/tfx-youtube

Tutorials

TFX on AI Platform Pipelines: https://www.tensorflow.org/tfx/tutorials/tfx/cloud-ai-platform-pipelines

MiniKF on VM Instance: https://codelabs.developers.google.com/codelabs/cloud-kubeflow-minikf-kale/

KubeFlow on GKE: https://codelabs.developers.google.com/codelabs/kubeflow-introduction

Github Issue Summarization: https://codelabs.developers.google.com/codelabs/cloud-kubeflow-pipelines-gis/