GOTOpia
FEBRUARY 2021

gotopia;

From Experimentation to Products

The Production ML Journey

Robert Crowe
TensorFlow Developer Engineer
@robert_crowe


Carl Zeiss
44132-9901
2

1 cell hidden

TensorFlow 2 quickstart for beginners

View on TensorFlow.org Run in Google Colab View source on GitHub Download notebook

This short introduction uses Keras to:

1. Build a neural network that classifies images.
2. Train this neural network.
3. And, finally, evaluate the accuracy of the model.

This is a Google Colaboratory notebook file. Python programs are run directly in the browser—a great way to learn and use TensorFlow. To follow this tutorial, run the notebook in Google Colab by clicking the button at the top of this page.

1. In Colab, connect to a Python runtime: At the top-right of the menu bar, select *CONNECT*.
2. Run all the notebook code cells: Select *Runtime* > *Run all*.

Download and install TensorFlow 2. Import TensorFlow into your program:

Note: Upgrade `pip` to install the TensorFlow 2 package. See the install guide for details.

```
[ ]  import tensorflow as tf
```

Load and prepare the MNIST dataset. Convert the samples from integers to floating-point numbers:

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin **Ming-Wei Chang** **Kenton Lee** **Kristina Toutanova**
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-

Production Machine Learning

Machine Learning Development

- Labeled data
- Feature space coverage
- Minimal dimensionality
- Maximum predictive data
- Fairness
- Rare conditions
- Data lifecycle management

Modern Software Development

- Scalability
- Extensibility
- Configuration
- Consistency & Reproducibility
- Modularity
- Best Practices
- Testability
- Monitoring
- Safety & Security

We need MLOps

“MLOps is an ML engineering culture and practice that aims at unifying ML system development (Dev) and ML system operation (Ops). Practicing MLOps means that you advocate for automation and monitoring at all steps of ML system construction, including integration, testing, releasing, deployment, and infrastructure management.” - http://goo.gle/mlops-levels

Continuous Integration, Deployment, and Training

Continuous Integration

- Testing and validating code and components, but also testing and validating data, data schemas, and models

Continuous Integration, Deployment, and Training

Continuous Deployment

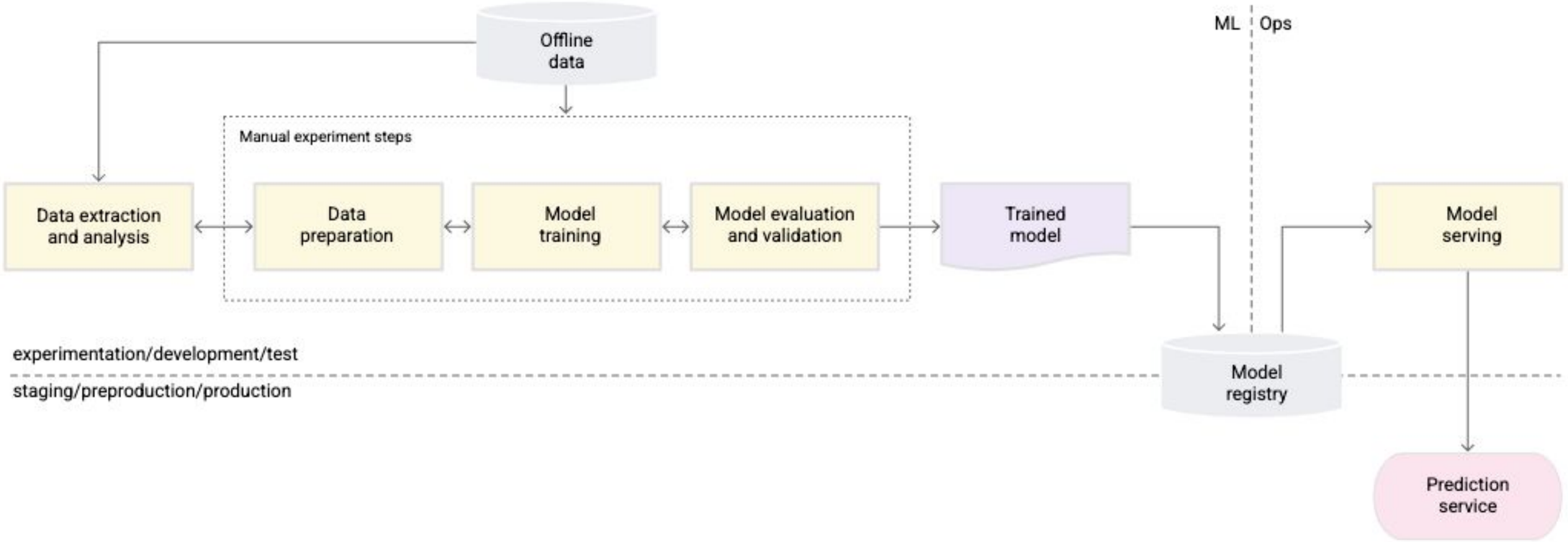
- No longer about a single software package or service, but a system (an ML training pipeline) that should automatically deploy another service (model prediction service)

Continuous Integration, Deployment, and Training

Continuous Training

- CT is a new process, unique to ML systems, that's concerned with automatically gathering and labeling new data, retraining, and serving new models

MLOps Level 0: Manual Process



Why isn’t one model good enough?

You’re an Online Retailer Selling Shoes ...

Your model predicts **click-through rates (CTR)**, helping you decide how much inventory to order

When suddenly

Your AUC and prediction accuracy
have **dropped** on men’s dress shoes!


Why?

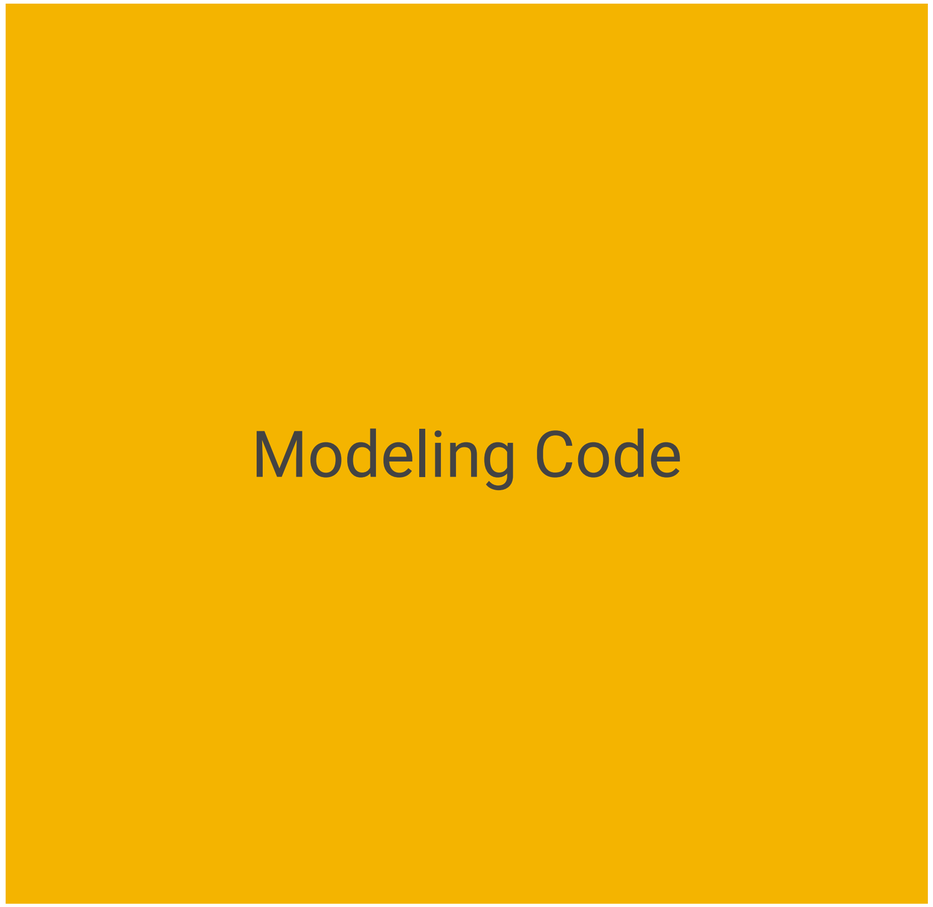
How do we know that we have a problem?


Why?

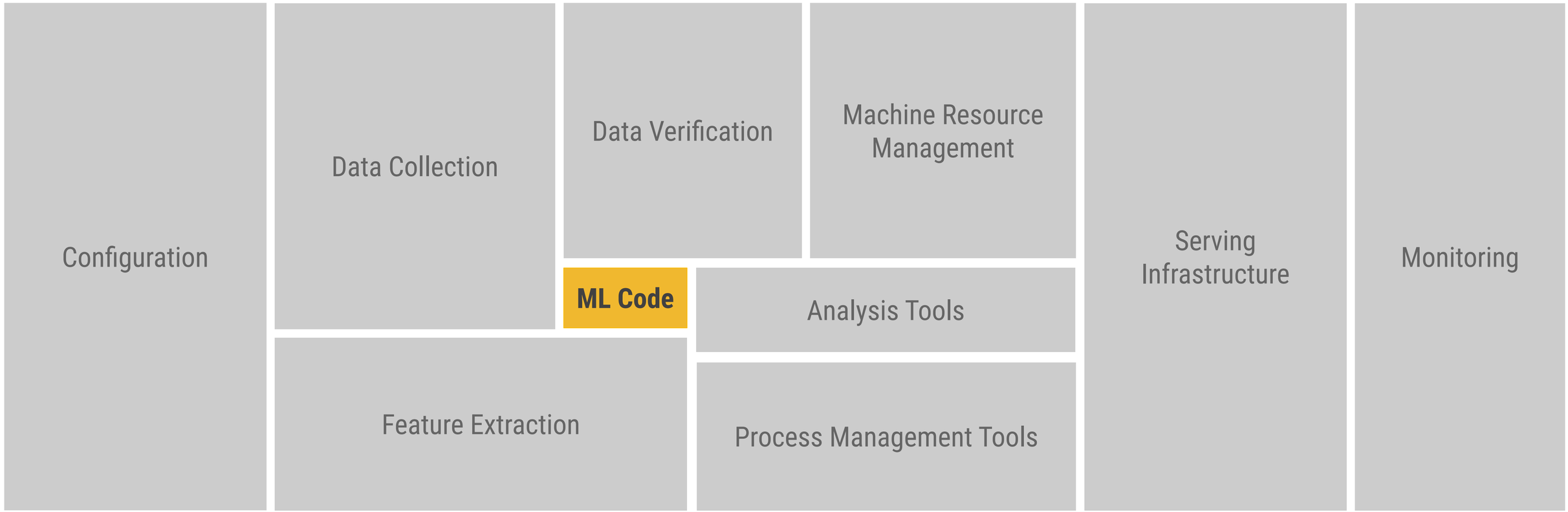
Change

Production Machine Learning

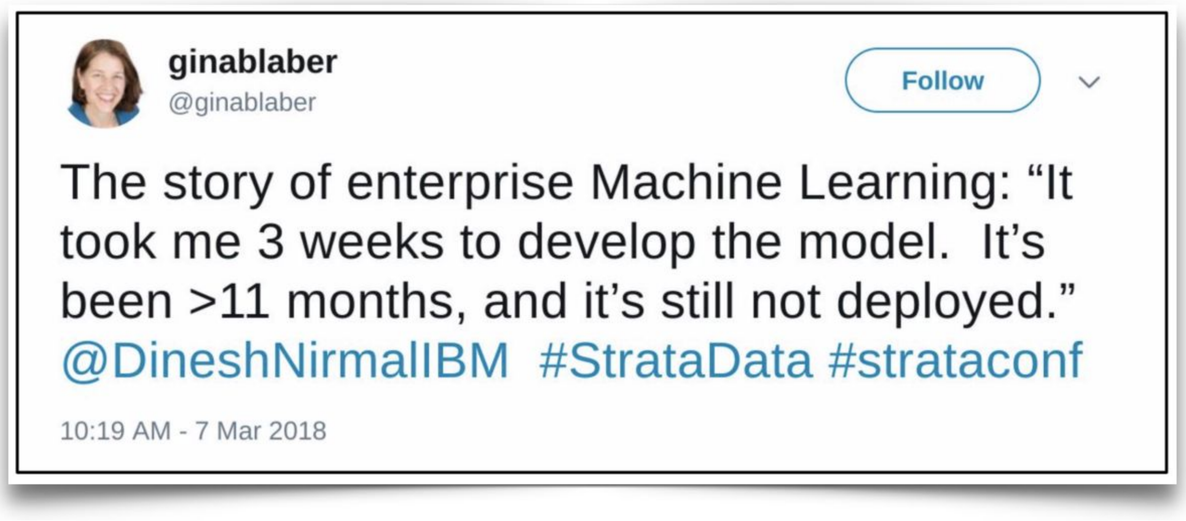
In addition to training an amazing model ...



… a production solution requires so much more



Tales From The Trenches



https://twitter.com/ginablaber/status/971450218095943681

Production Machine Learning

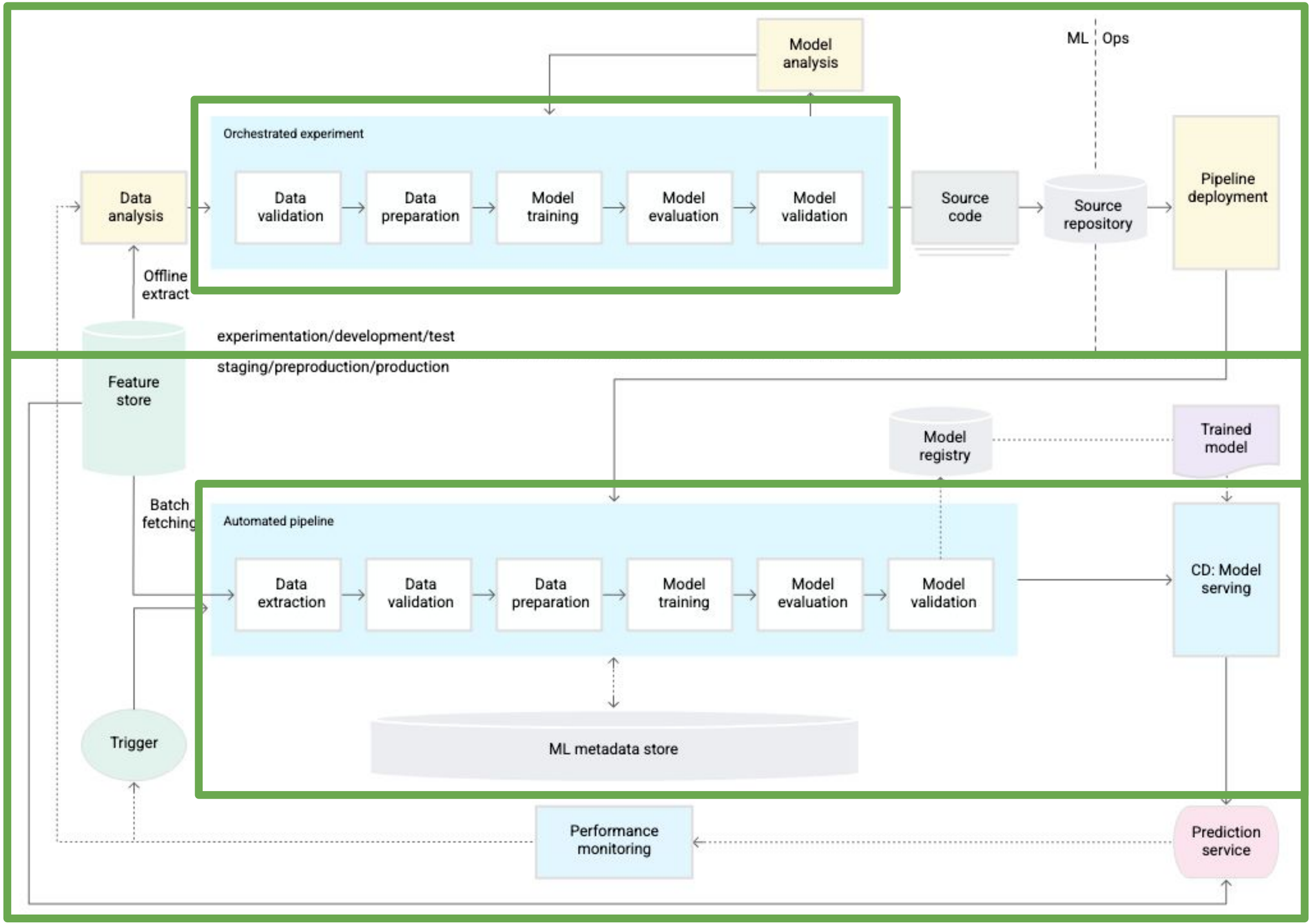
“Hidden Technical Debt in Machine Learning Systems”

NIPS 2015

http://bit.ly/ml-techdebt

What now?

MLOps Level 1: ML pipeline automation



TensorFlow Extended (TFX)

Tensorflow Extended (TFX)

Powers Alphabet's most important bets and products









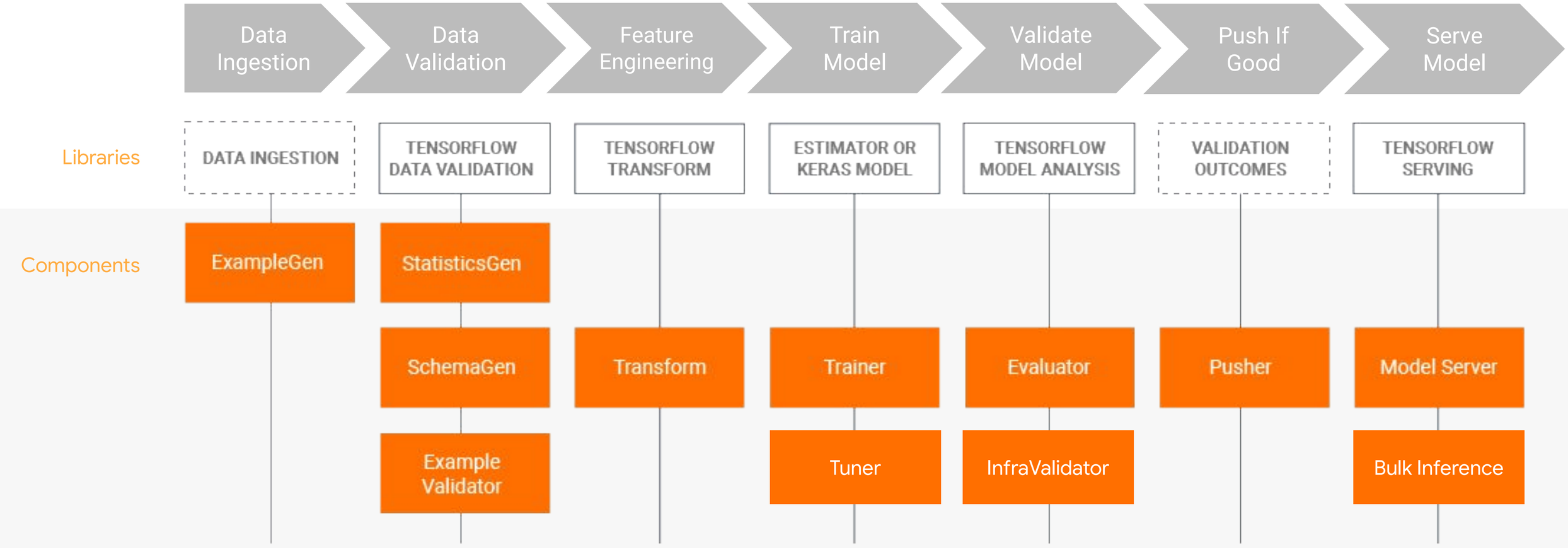

“... we have re-tooled our machine learning platform to use TensorFlow. This yielded significant productivity gains while positioning ourselves to take advantage of the latest industry research.”

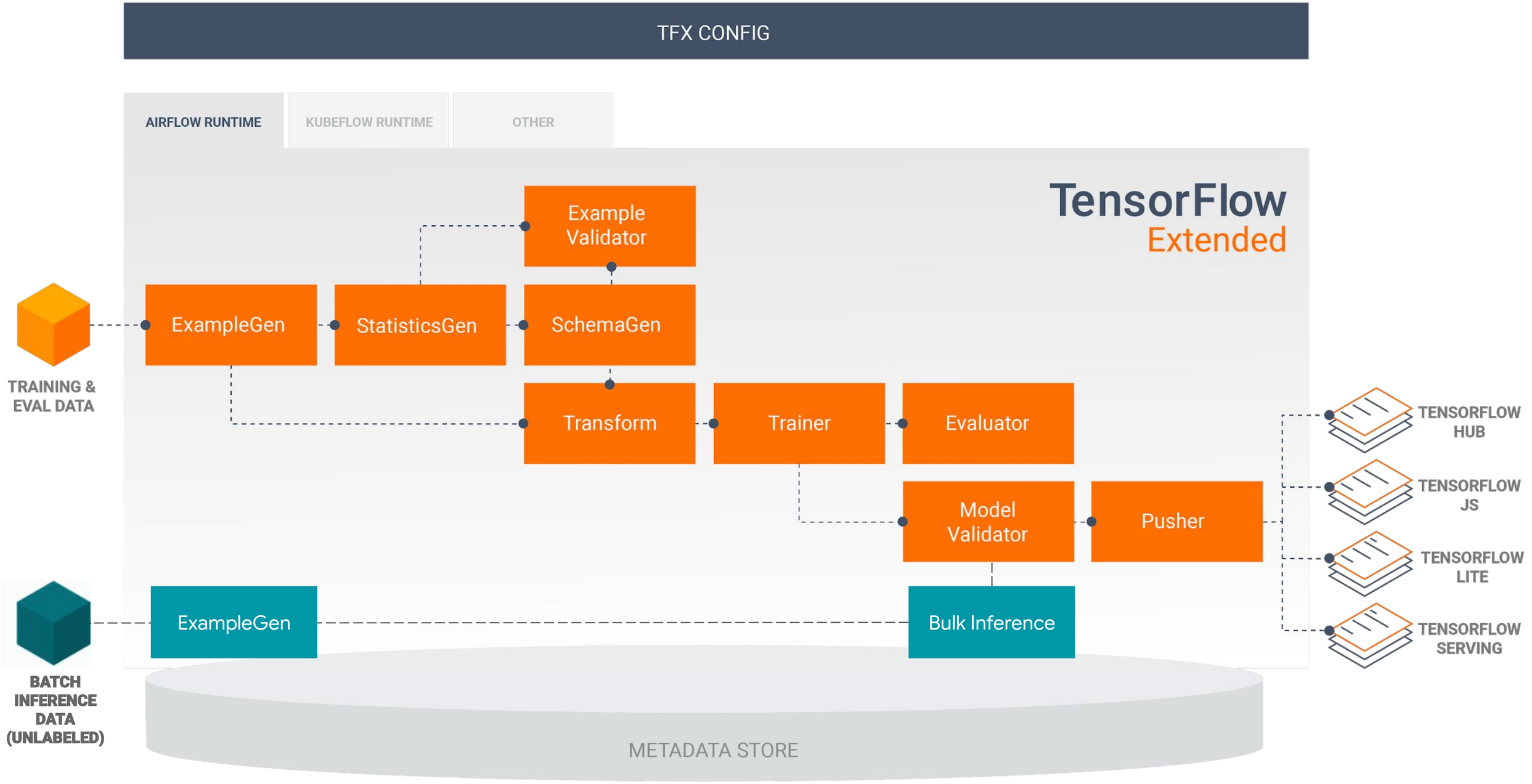
Ranking Tweets with TensorFlow - Twitter

https://goo.gle/tf-twitter-rank

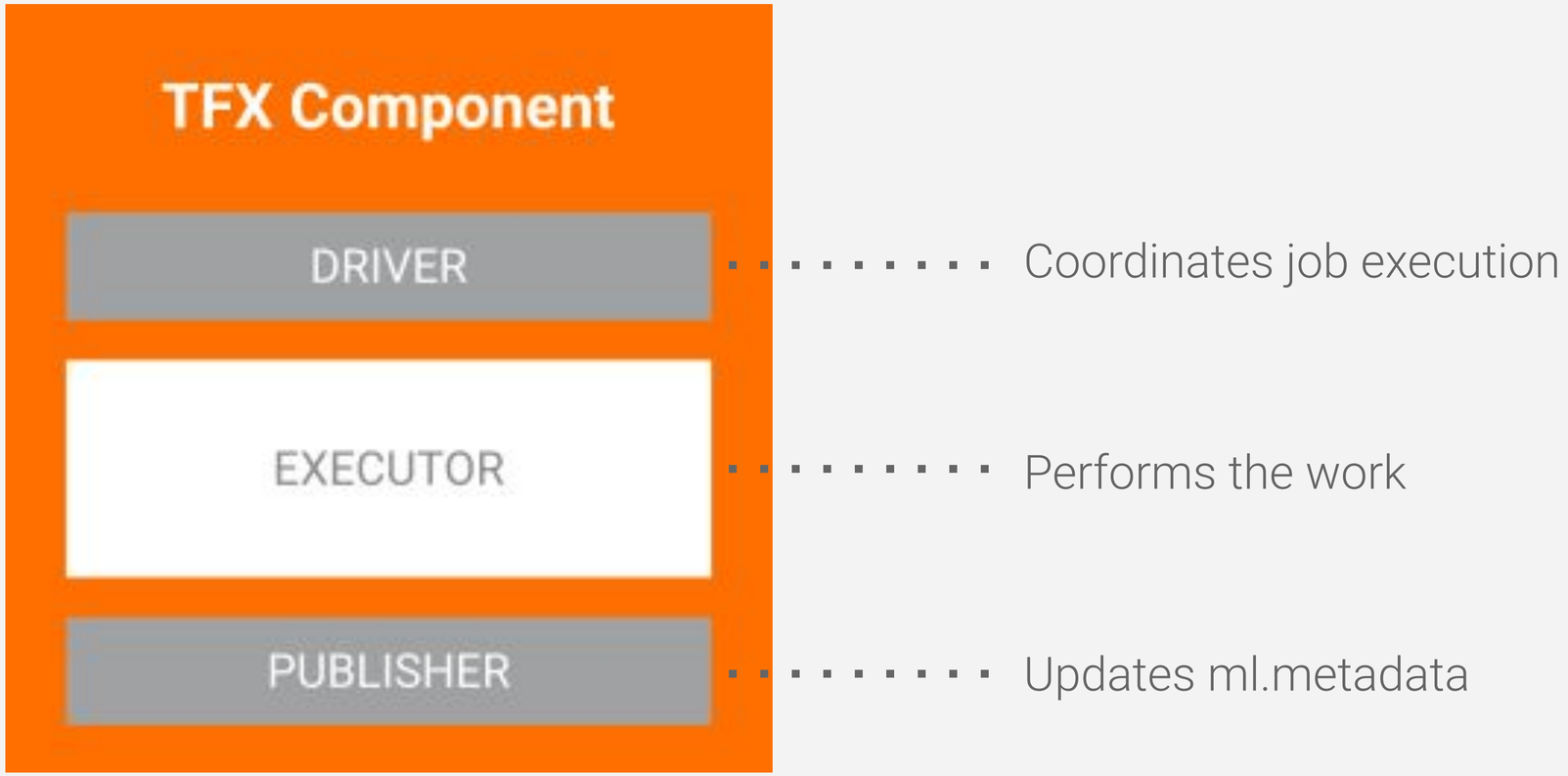
Spotify	Snap	Etsy
Airbus	Twitter	OpenX
Tencent	NetEase	Yahoo Japan
	NBC	

TFX Production Components

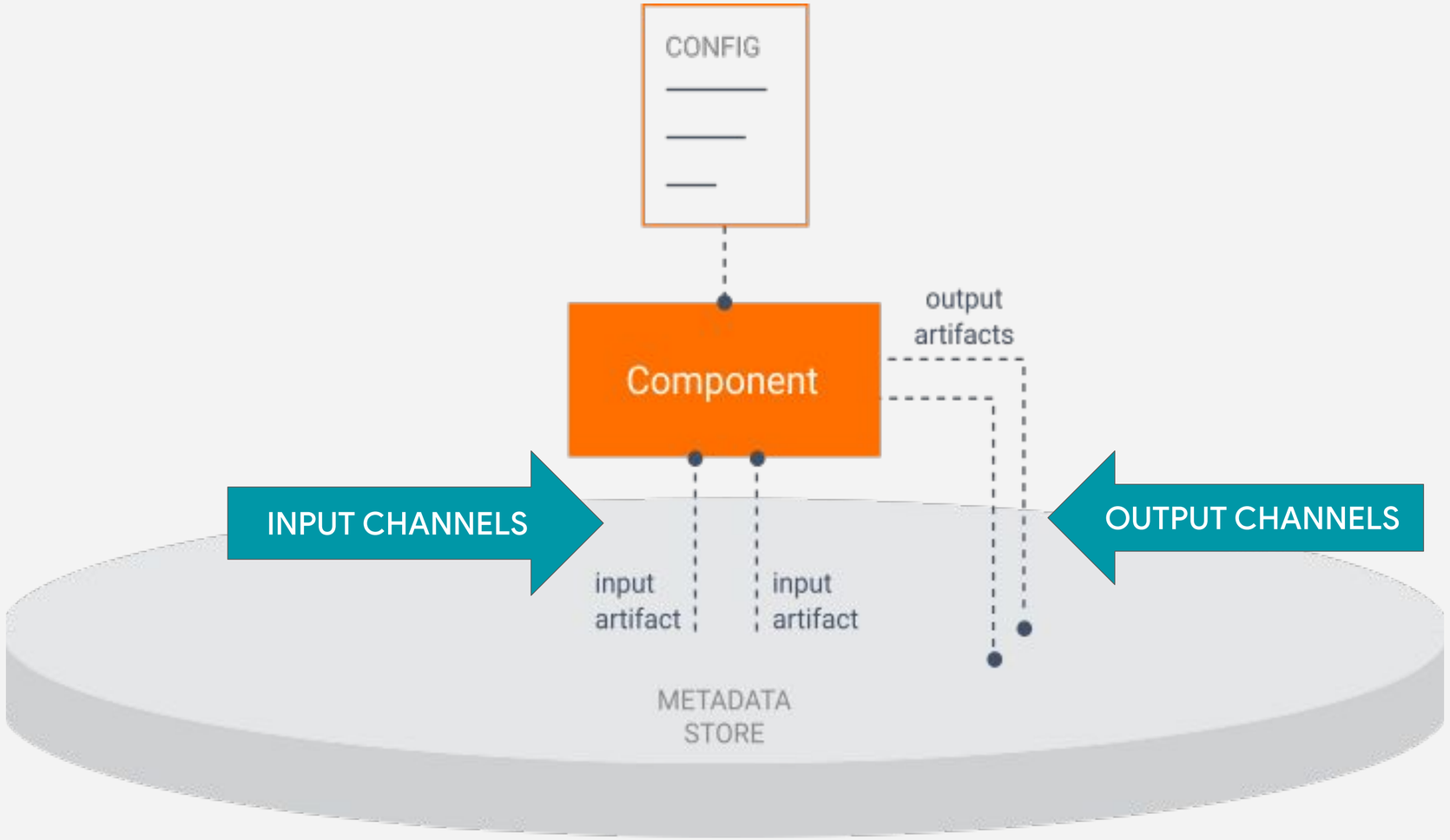



TFX CONFIG
AIRFLOW RUNTIME
KUBEFLOW RUNTIME
OTHER
TensorFlow
Extended
Example Validator
ExampleGen
StatisticsGen
SchemaGen
TRAINING & EVAL DATA
Transform
Trainer
Evaluator
Model Validator
Pusher
TENSORFLOW HUB
TENSORFLOW JS
TENSORFLOW LITE
TENSORFLOW SERVING
ExampleGen
Bulk Inference
BATCH INFERENCE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

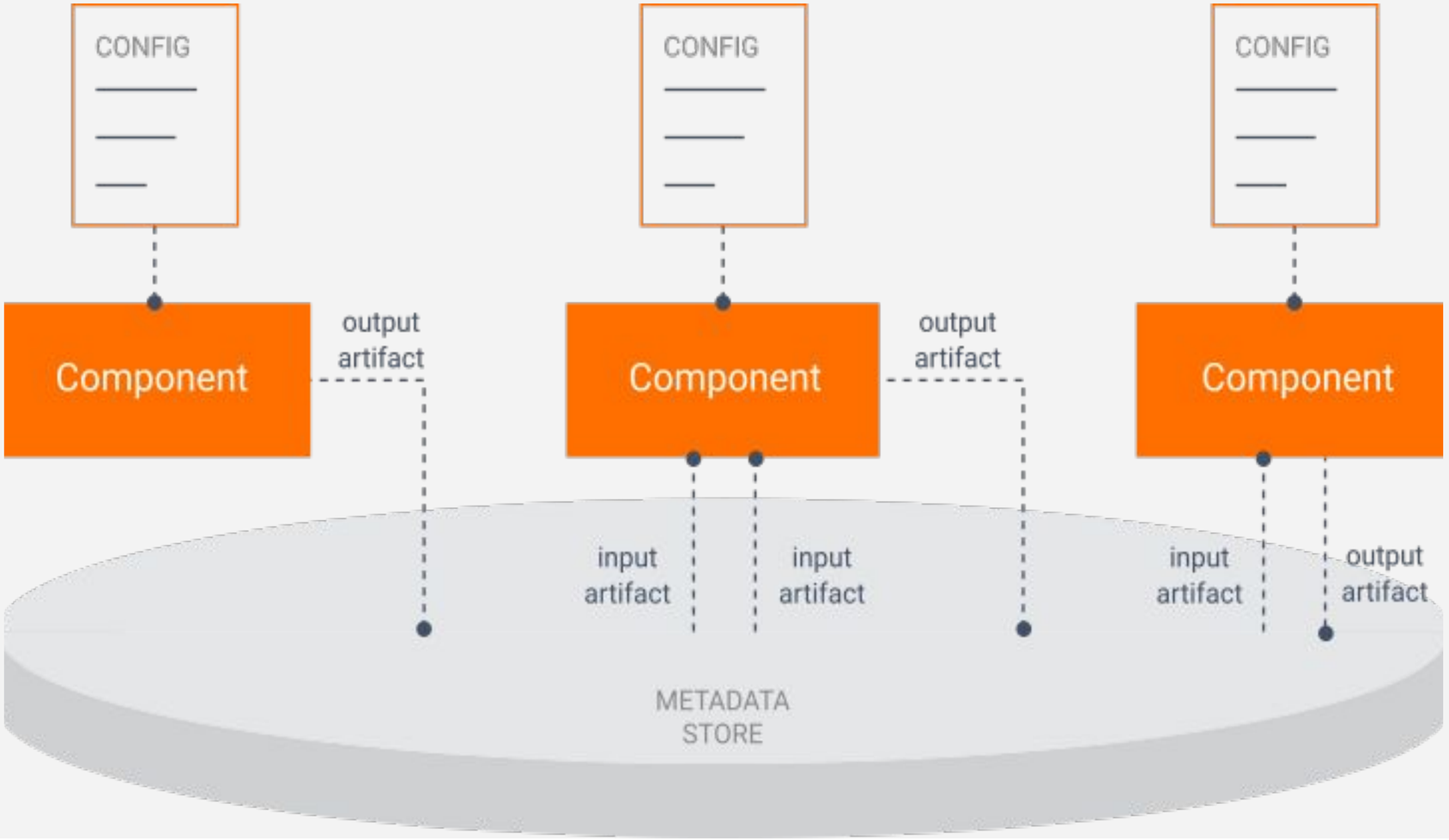
What is a TFX component?



What makes a Component

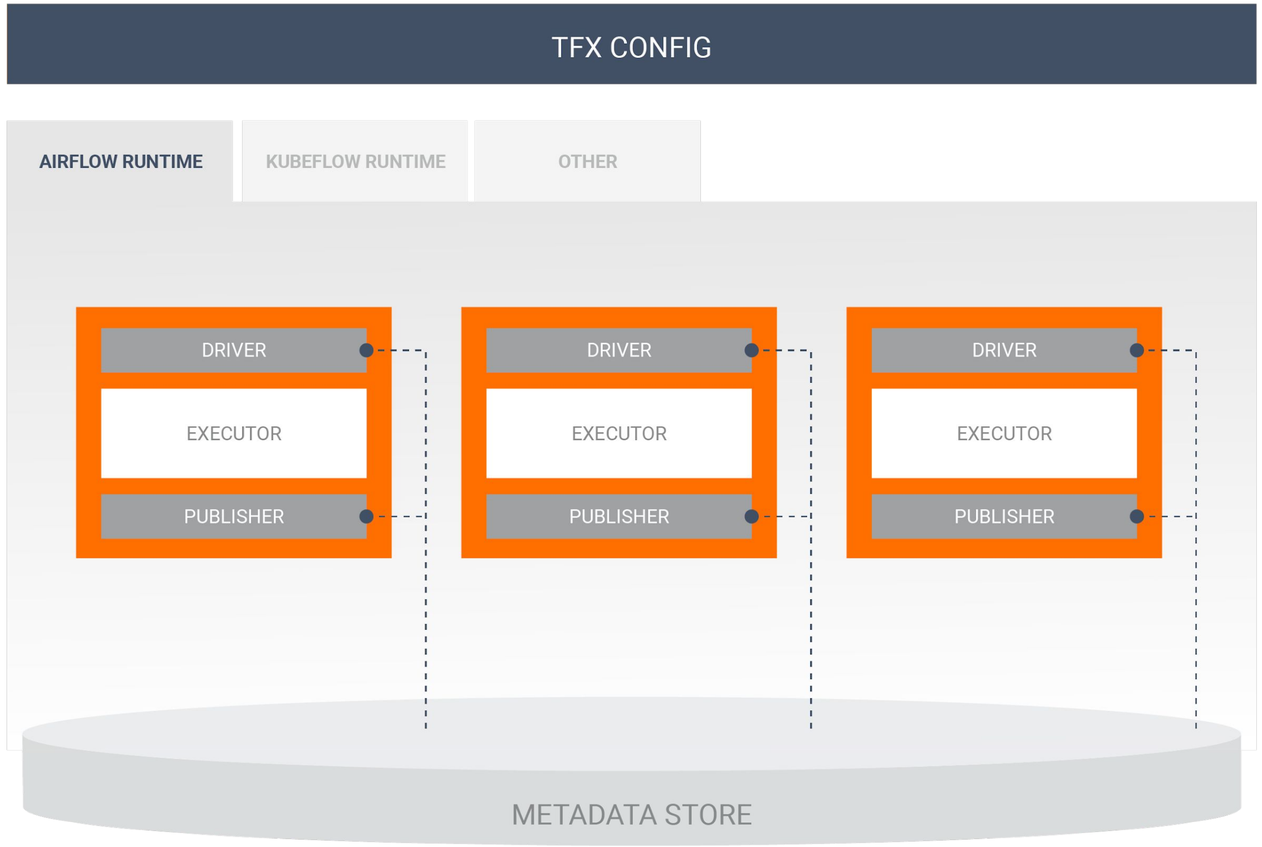


What makes a Component?



What makes a Component?

TFX Orchestration

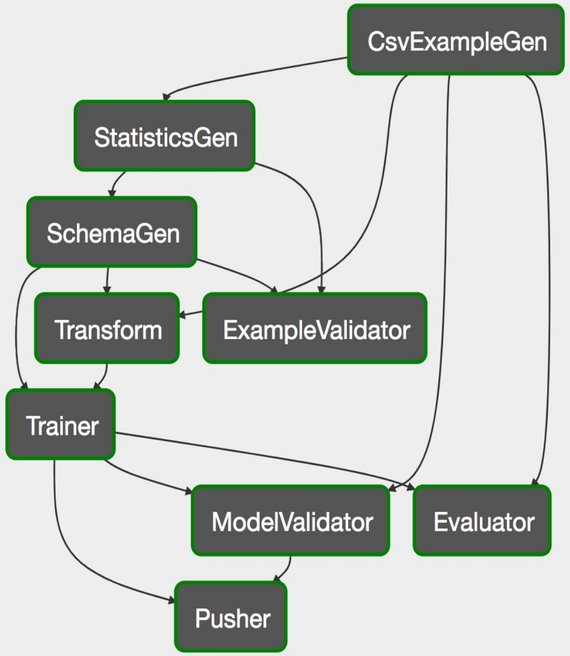


Bring your own Orchestrator

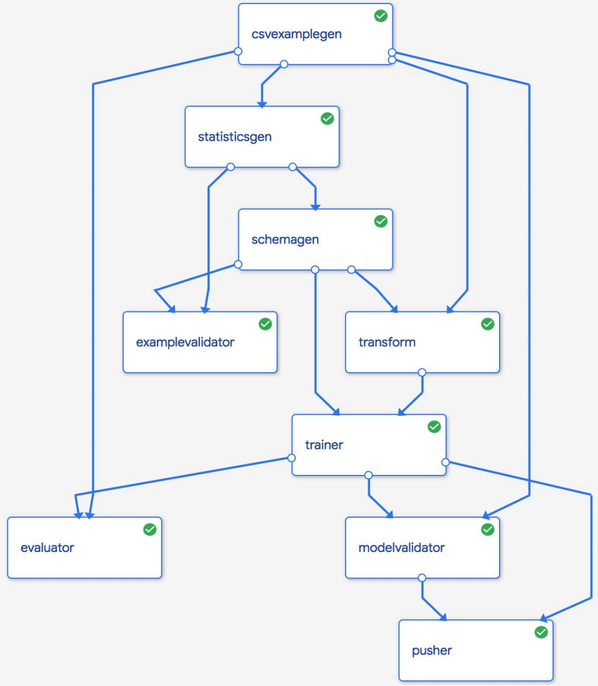
Flexible runtimes run components in the proper order using orchestration systems such as Airflow, Kubeflow, Interactive, or Local

Orchestrators and DAGs

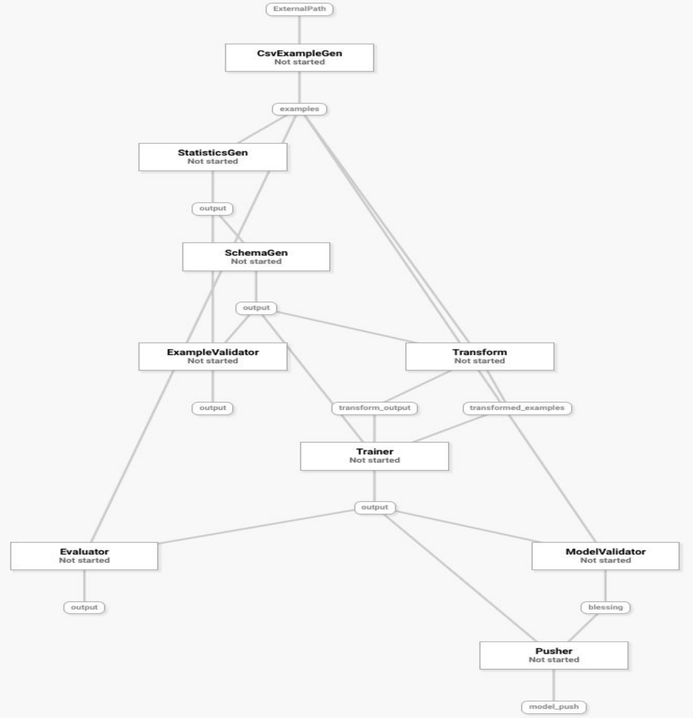
Apache Airflow



Kubeflow Pipelines



Local



TFX and Kubeflow Pipelines

Kubeflow

TensorFlow Extended (TFX)

- Open-source version of what Google uses internally for Production ML
- Currently supported orchestrators:
  - **Kubeflow Pipelines**
  - Apache Airflow
  - Local
  - Interactive
  - We're adding more
  - You can add more

AI Platform Pipelines

- Metadata tracking + caching enabled, ability to resume pipelines from crashes.
- Containers as custom components
- Orchestrate existing R/C++/Scala components and get metadata tracking + caching
- Artifact provenance and lineage visualized
- Deploy using Marketplace
- CloudSQL can be used to persist pipeline metadata across clusters

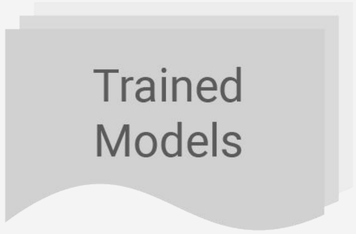
Best of both worlds when used together

TFX Orchestration in a Notebook

- Environment for experimental and iterative development
- Build up your pipeline iteratively in a Jupyter / Colab notebook and export to production with minimal changes
- InteractiveContext object handles component execution, metadata management, and artifact visualization
- In production, you would use Airflow, Kubeflow, or something similar

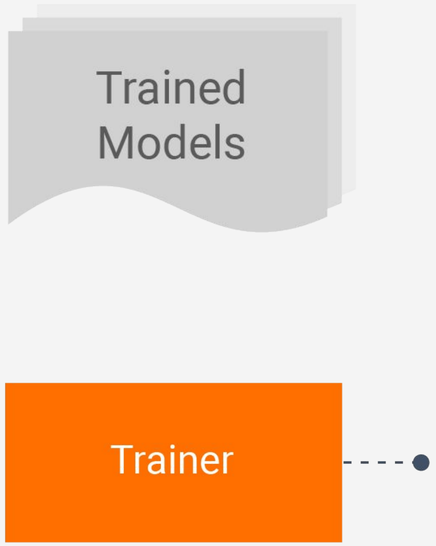
```
context = InteractiveContext()

component = MyComponent(...)
context.run(component)
context.show(component.outputs['my_output'])
```

Metadata Store



What is in Metadata Store?

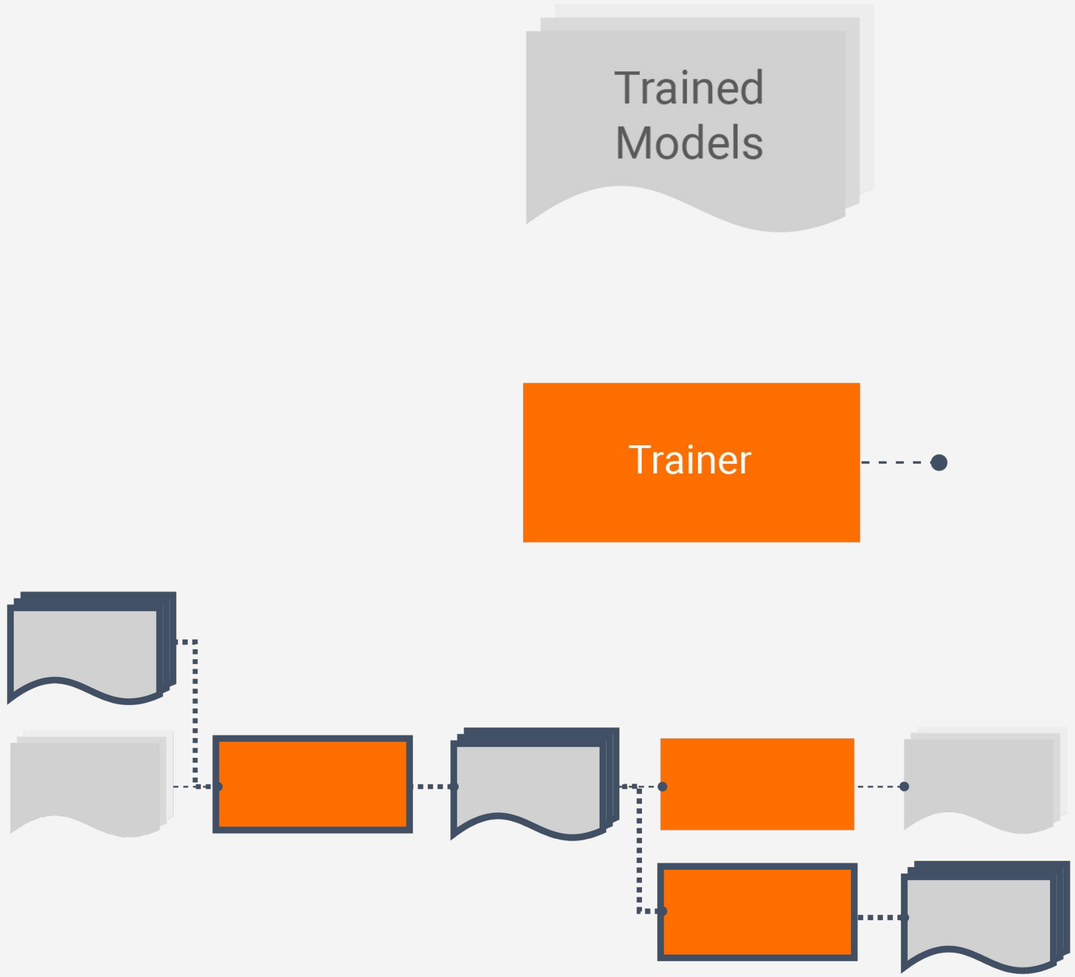
Type definitions of Artifacts and their Properties



What is in Metadata Store?

Type definitions of Artifacts and their Properties

Execution Records (Runs) of Components

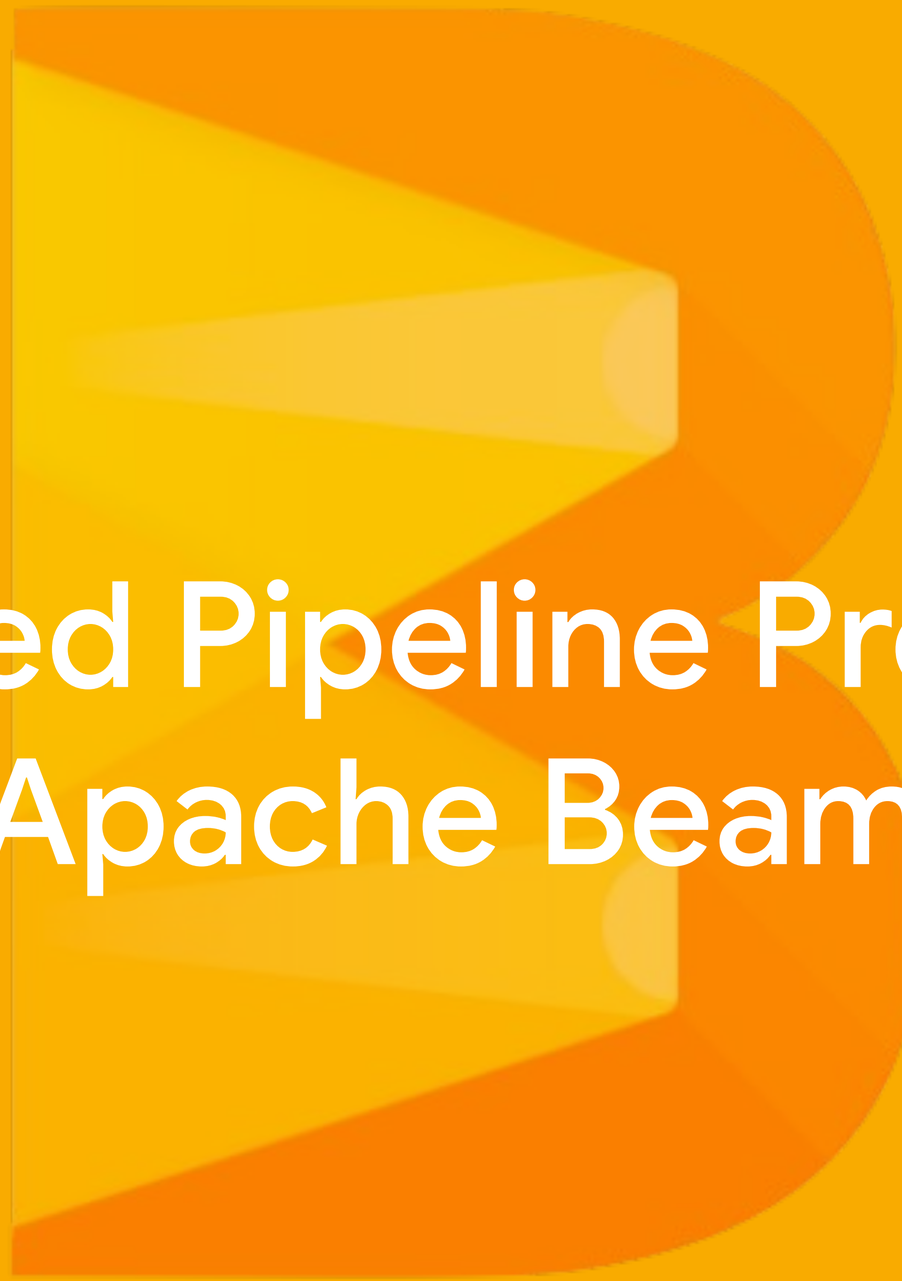


What is in Metadata Store?

Type definitions of Artifacts and their Properties

Execution Records (Runs) of Components

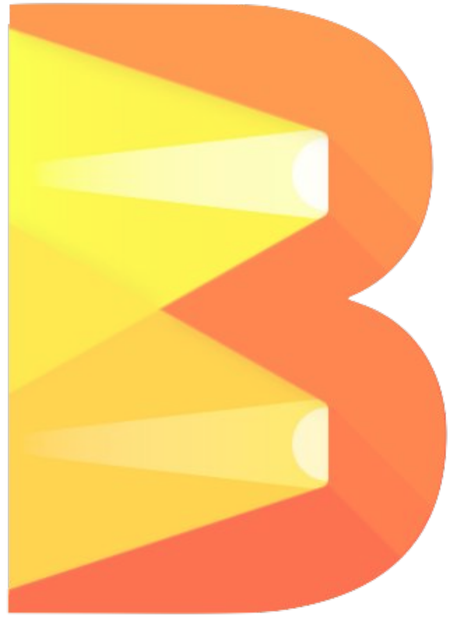
Data Provenance Across All Executions



Distributed Pipeline Processing: Apache Beam

What is Apache Beam?

- A unified **batch** and stream distributed processing API
- A set of **SDK frontends**: Java, **Python**, Go, Scala, SQL
- A set of **Runners** which can execute Beam jobs into various backends: **DirectRunner, Apache Flink**, **Apache Spark**, Apache Gearpump, **Apache Samza**, Apache Hadoop, **Google Cloud Dataflow**, ...



Apache Beam

Java

```
input.apply(
  Sum.integersPerKey())
```

Python

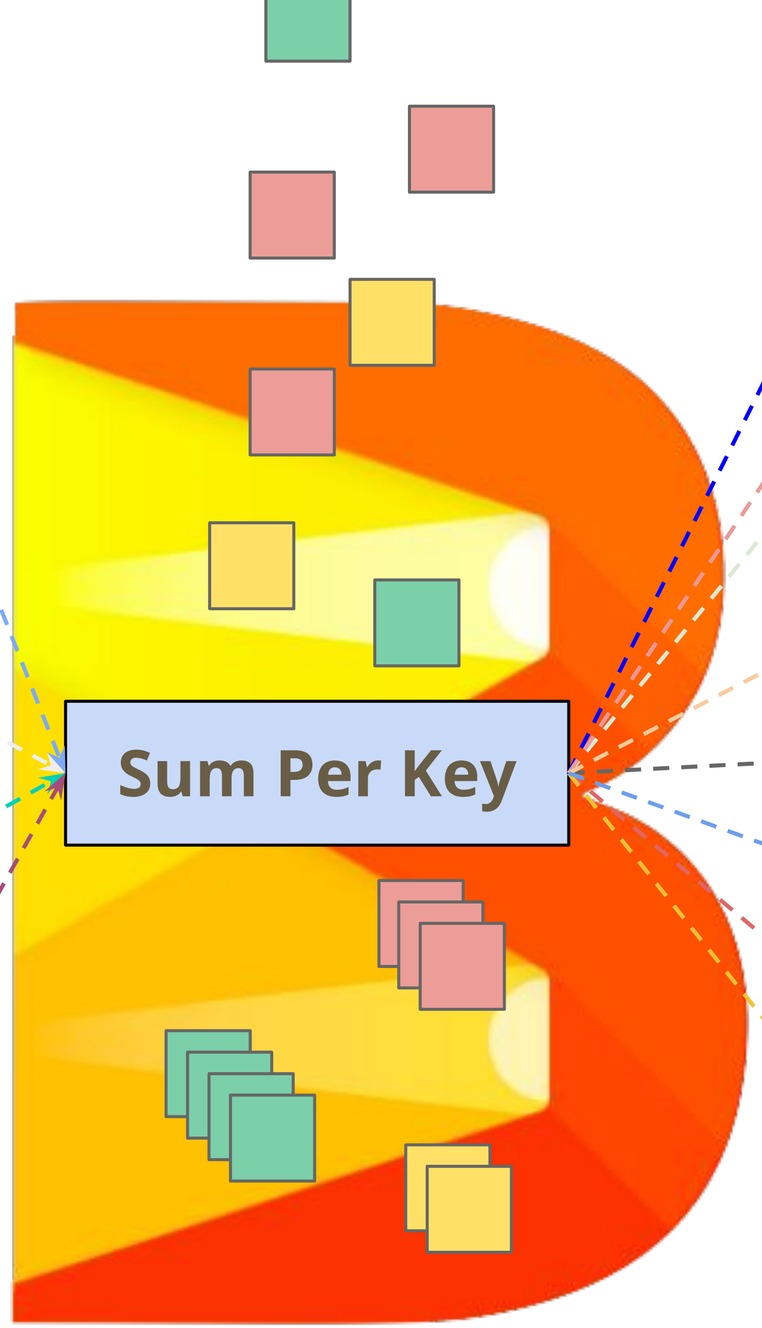
```
input | Sum.PerKey()
```

Go

```
stats.Sum(s, input)
```

SQL

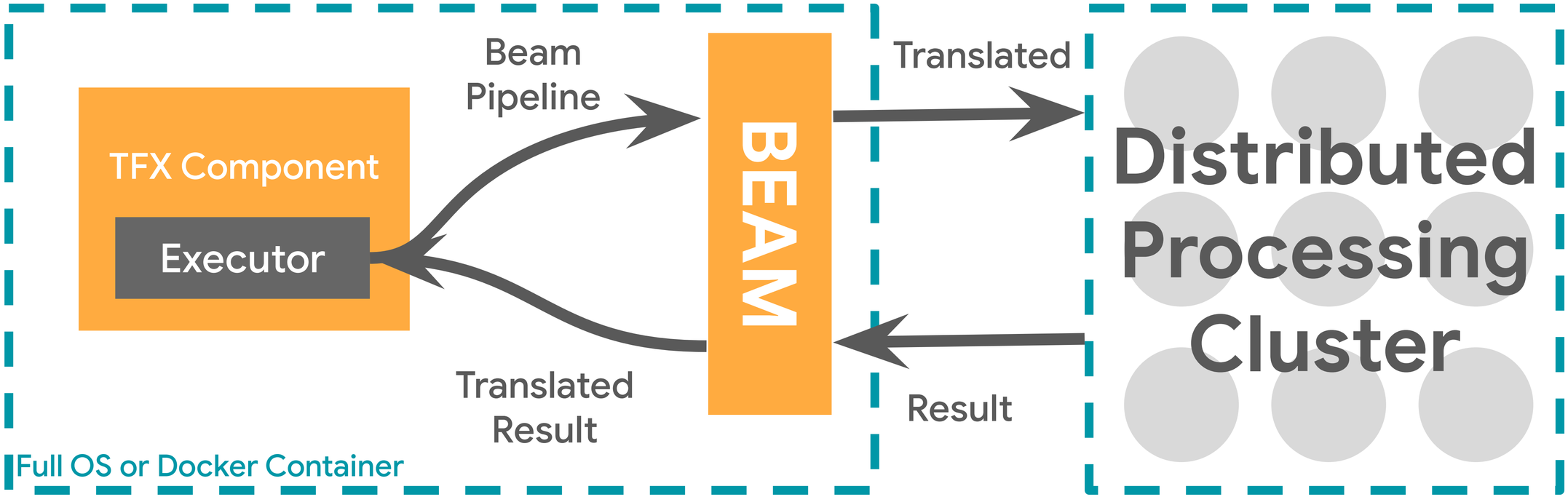
```
SELECT key, SUM(value)
FROM input GROUP BY key
```



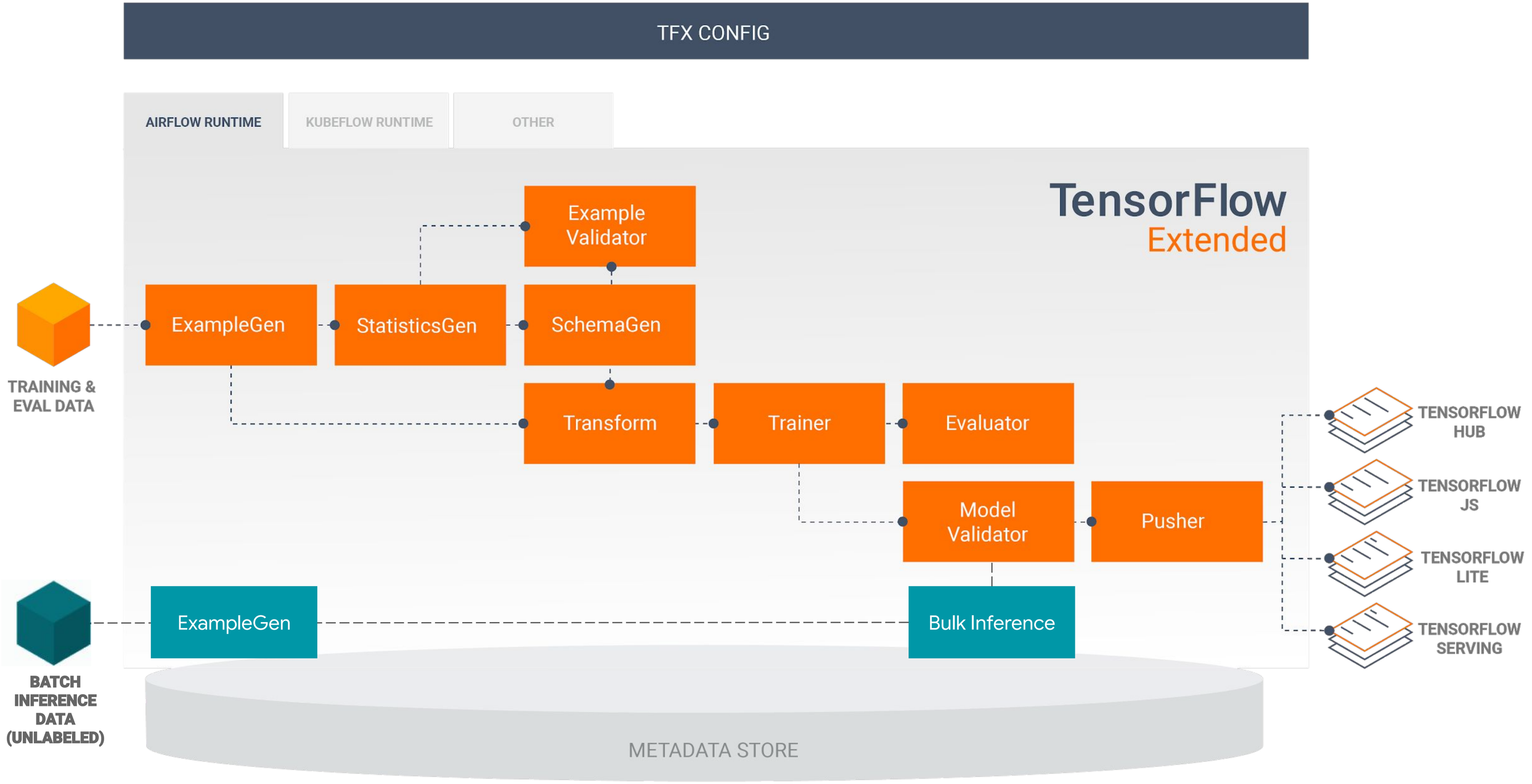
- Cloud Dataflow
- Apache Flink
- Apache Spark
- Apache Apex
- Gearpump
- IBM Streams
- Apache Samza
- Apache Nemo (incubating)

⋮

How TFX Components Use Beam

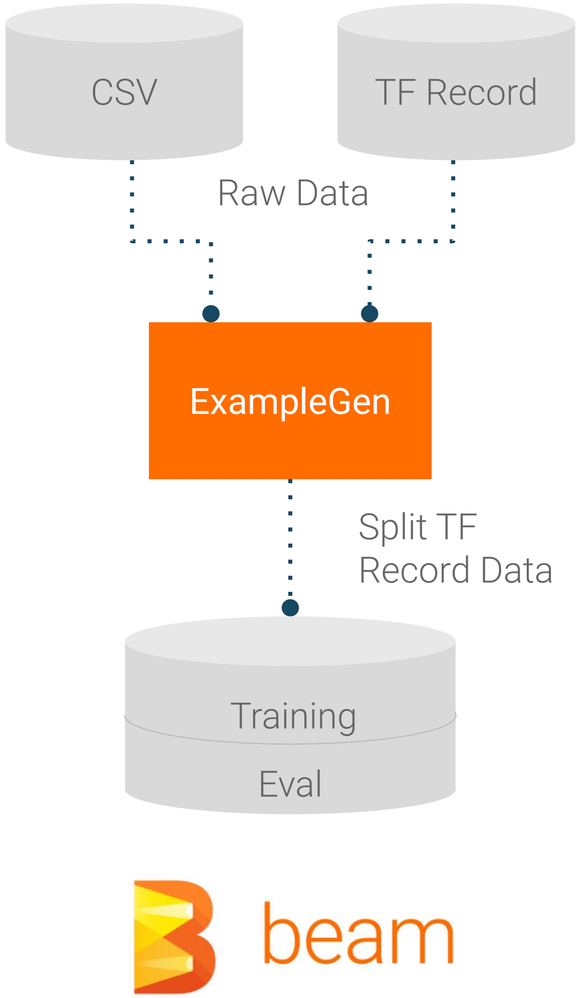


TFX Standard Components


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

Component: ExampleGen

Inputs and Outputs



Configuration

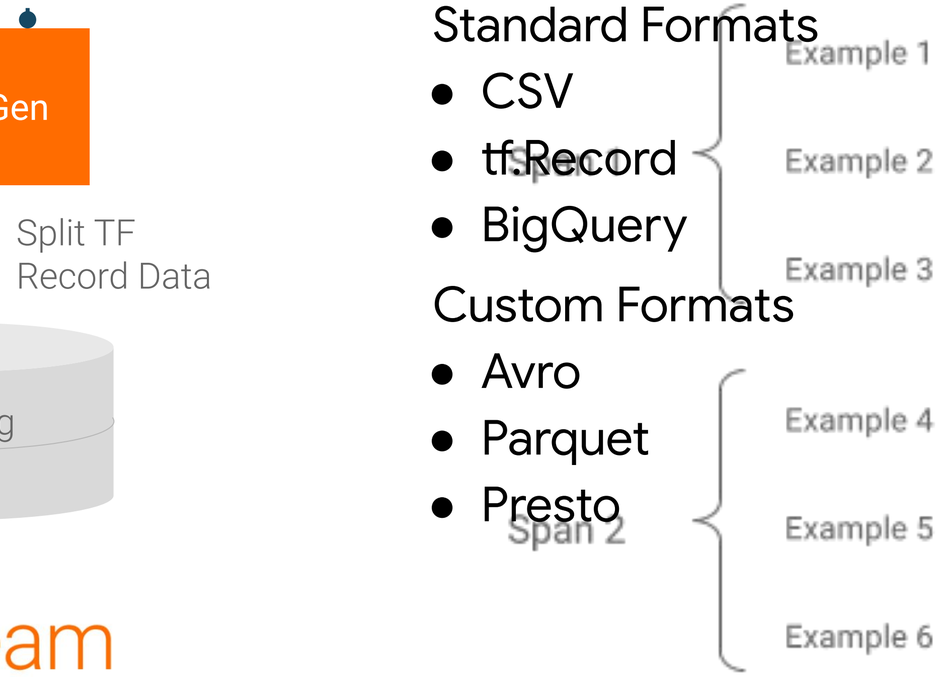
```
examples = csv_input(os.path.join(base_dir, 'data/simple'))
example_gen = CsvExampleGen(input=examples)
```

Standard Formats

- CSV
- tf.Record
- BigQuery

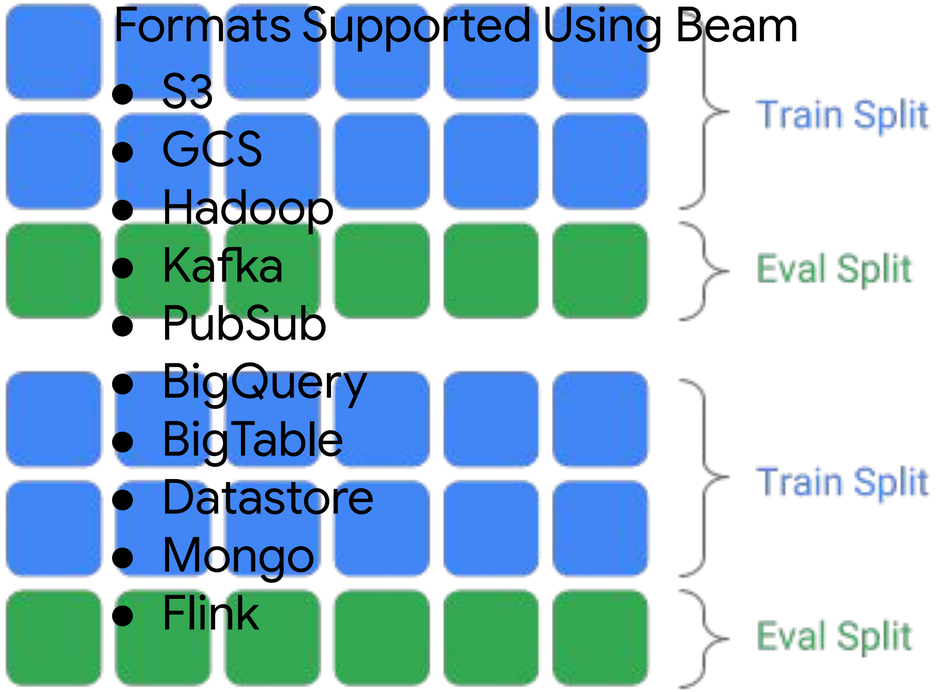
Custom Formats

- Avro
- Parquet
- Presto



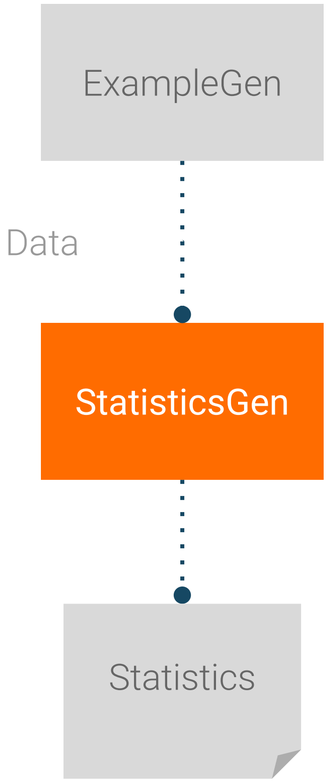
Formats Supported Using Beam

- S3
- GCS
- Hadoop
- Kafka
- PubSub
- BigQuery
- BigTable
- Datastore
- Mongo
- Flink



Component: StatisticsGen

Inputs and Outputs

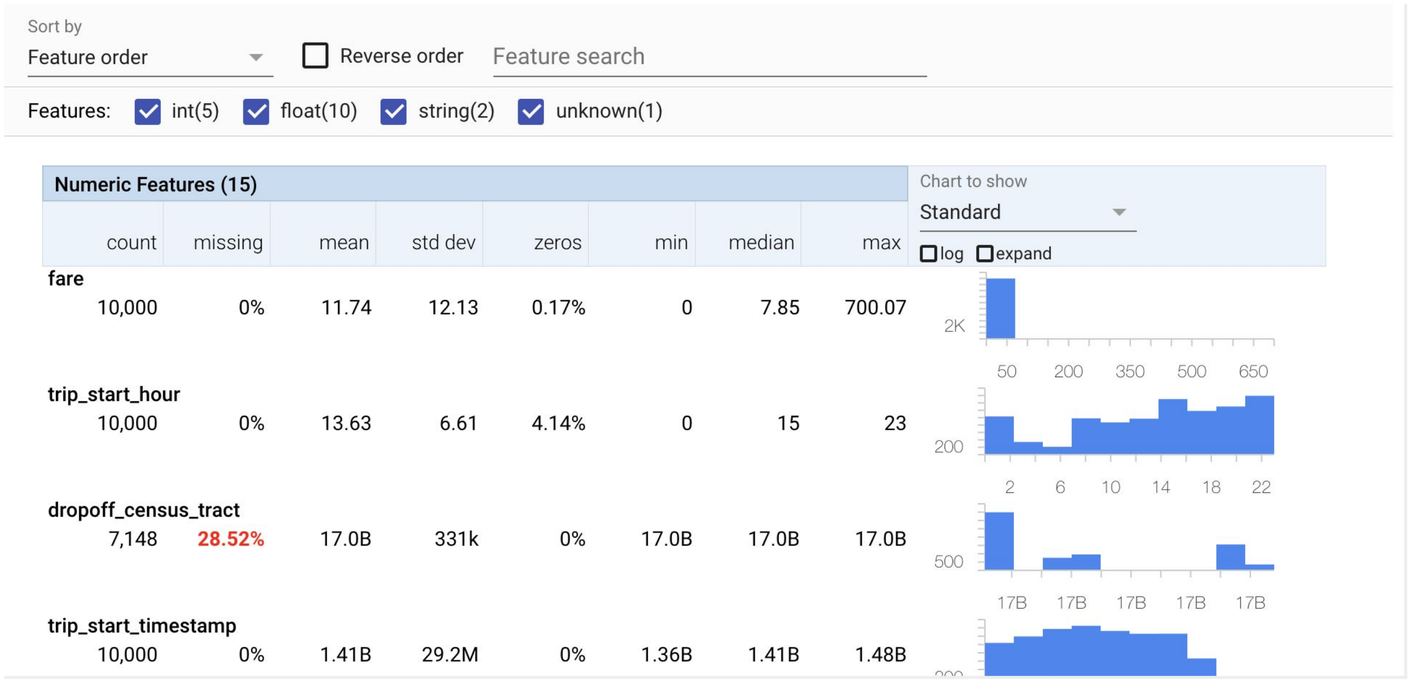


beam

Configuration

```
statistics_gen = StatisticsGen(
    input_data=example_gen.outputs['examples'])
```

Visualization



Analyzing Data with TensorFlow Data Validation

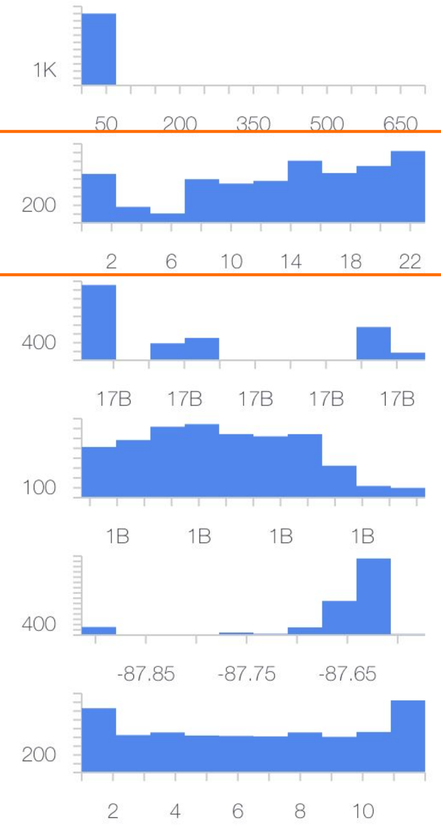
Sort by
Feature order ▾ ☐ Reverse order Feature search (regex enabled)

Features: ☑ int(5) ☑ float(5) ☑ variable-length floats(5) ☑ string(1) ☑ fixed-length strings(1) ☑ variable-length strings(1)

Numeric Features (15)

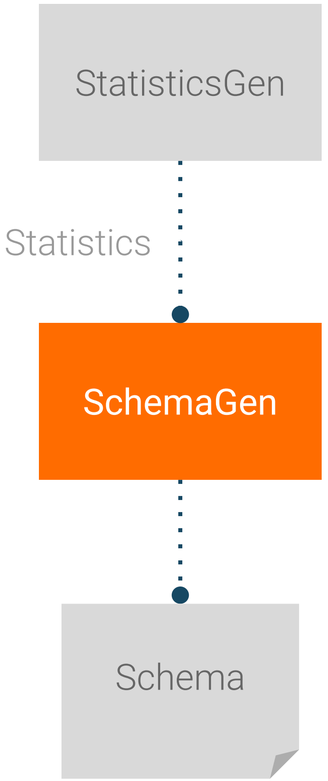
Chart to show: Standard ▾ ☐ log ☐ expand

	count	missing	mean	std dev	zeros	min	median	max
fare	5,009	0%	11.97	14.12	0.12%	0	7.85	700.07
trip_start_hour	5,009	0%	13.51	6.73	4.13%	0	15	23
dropoff_census_tract	5,009	0%	17.0B	328k	0%	17.0B	17.0B	17.0B
trip_start_timestamp	5,009	0%	1.41B	29.0M	0%	1.36B	1.41B	1.48B
pickup_longitude	5,009	0%	-87.66	0.07	0%	-87.91	-87.63	-87.57
trip_start_month	5,009	0%	6.6	3.4	0%	1	7	12



Component: SchemaGen

Inputs and Outputs



Configuration

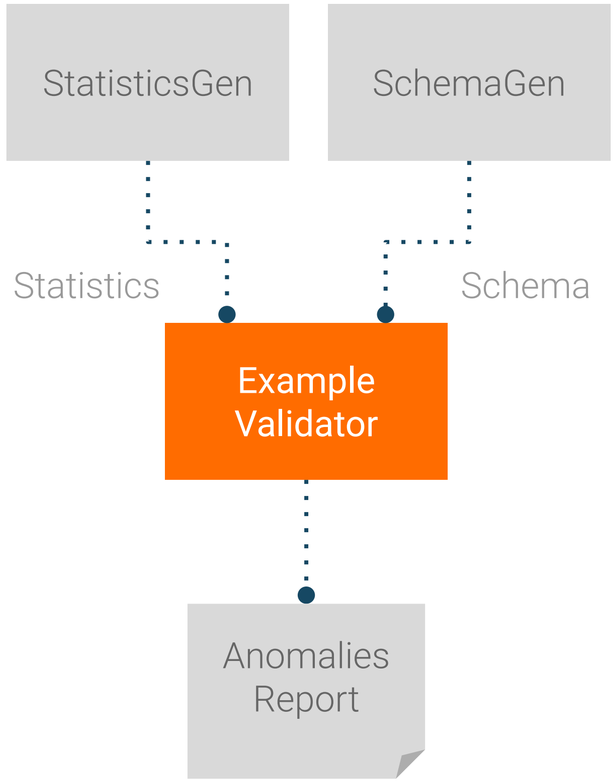
```
infer_schema =
    SchemaGen(statistics=statistics_gen.outputs['statistics'],
              infer_feature_shape=False)
```

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

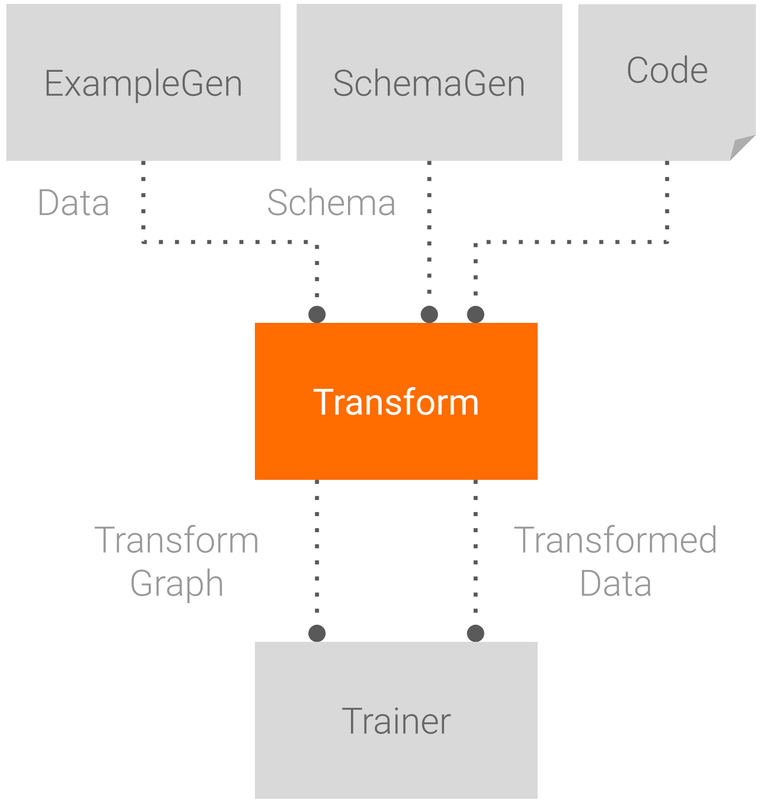
```
validate_stats = ExampleValidator(
      statistics=statistics_gen.outputs['statistics'],
      schema=infer_schema.outputs['schema'])
```

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

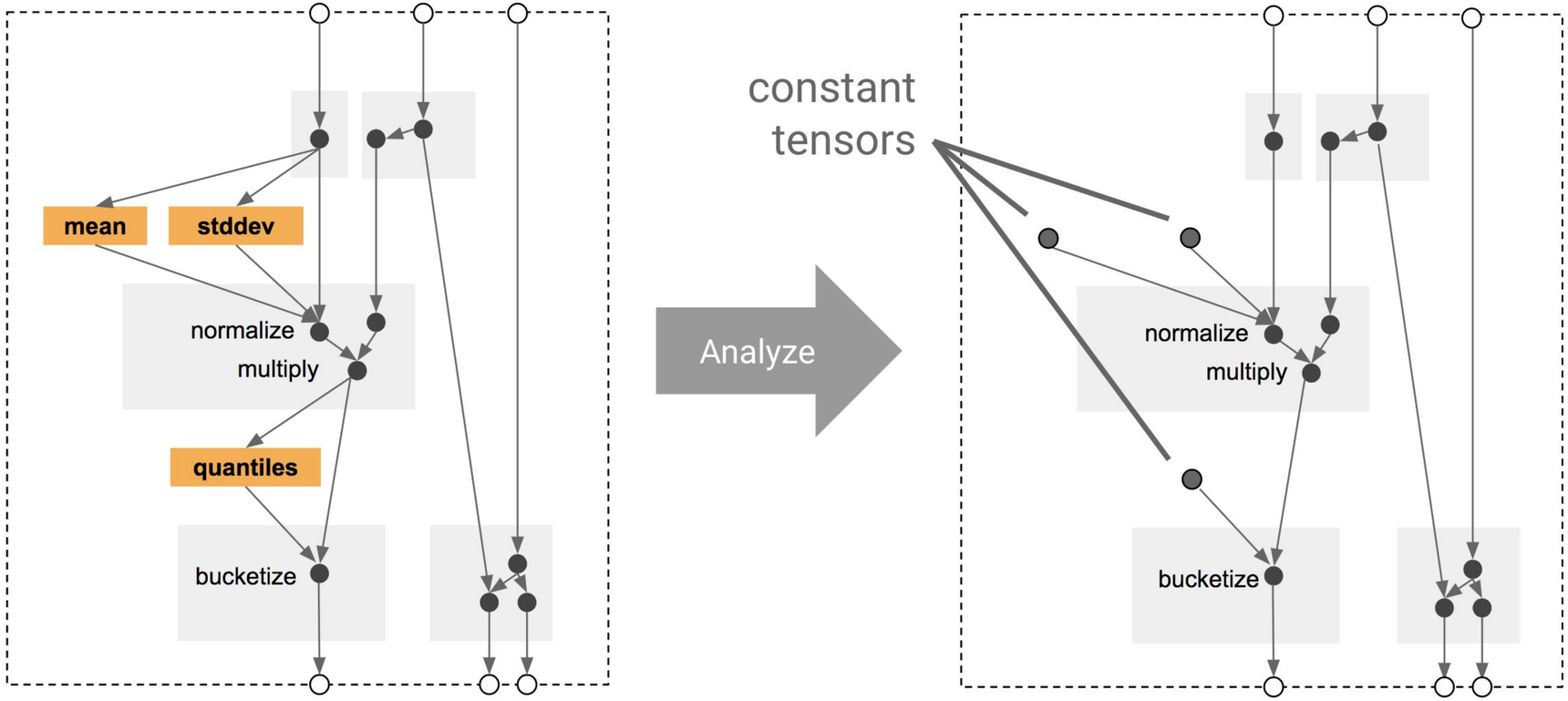
Configuration

```
transform = Transform(
      examples=example_gen.outputs['output_data'],
      schema=infer_schema.outputs['schema'],
      module_file=module_file)
```

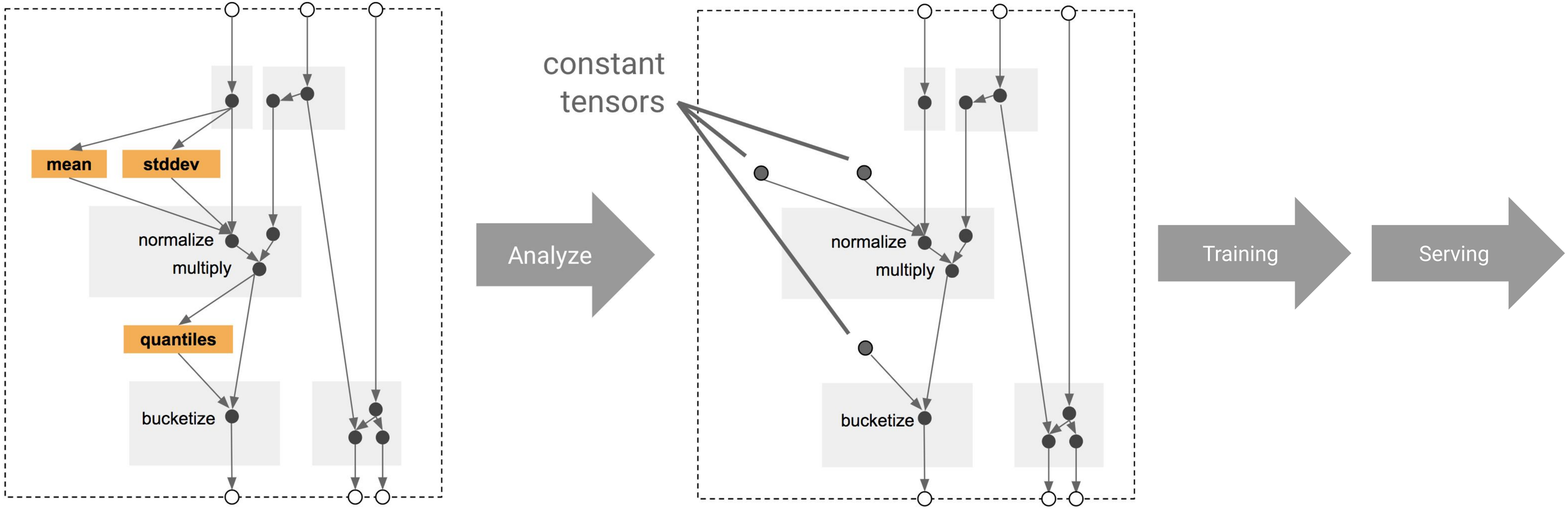
Code

```
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

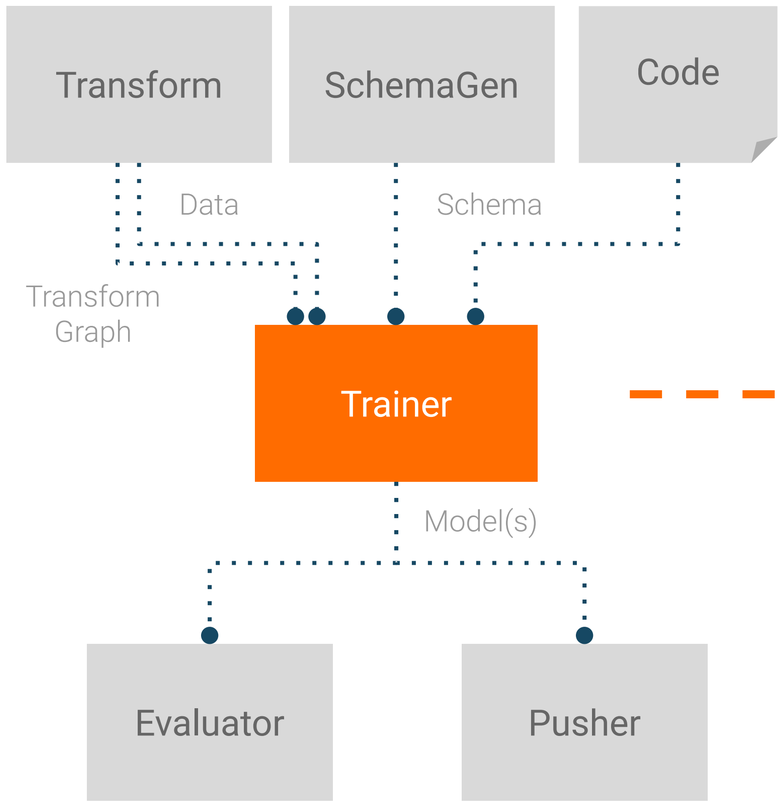


Using TensorFlow Transform for Feature Engineering



Component: Trainer

Inputs and Outputs



Highlight: SavedModel Format

Train, Eval, and Inference Graphs

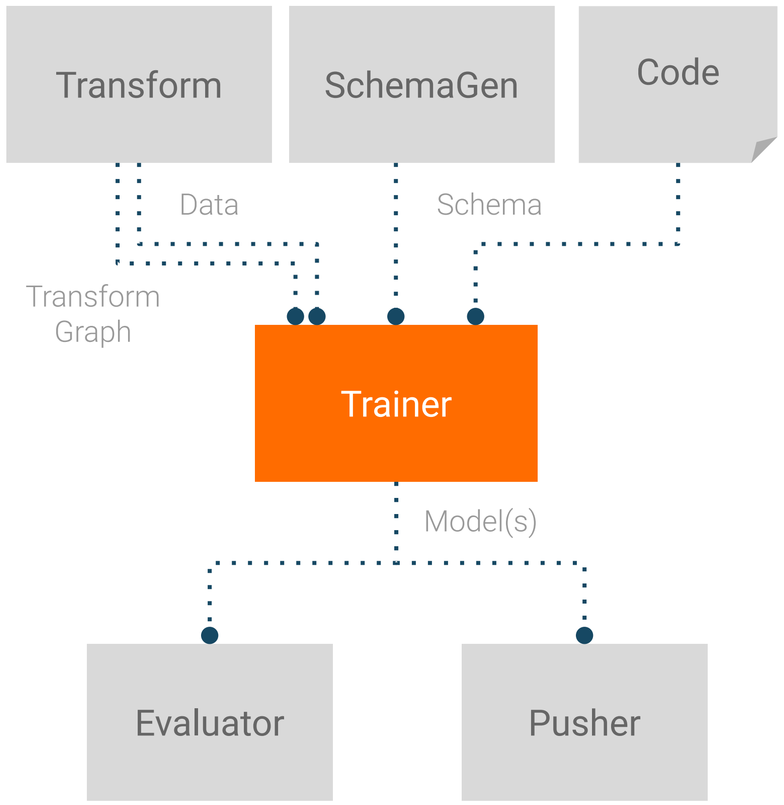
SavedModel

TensorFlow Model Analysis

TensorFlow Serving

Component: Trainer

Inputs and Outputs



Configuration

```
trainer = Trainer(
      module_file=module_file,
      transformed_examples=
          transform.outputs['transformed_examples'],
      schema=infer_schema.outputs['schema'],
      transform_graph=transform.outputs['transform_graph'],
      train_args=trainer_pb2.TrainArgs(num_steps=10000),
      eval_args=trainer_pb2.EvalArgs(num_steps=5000))
```

Code

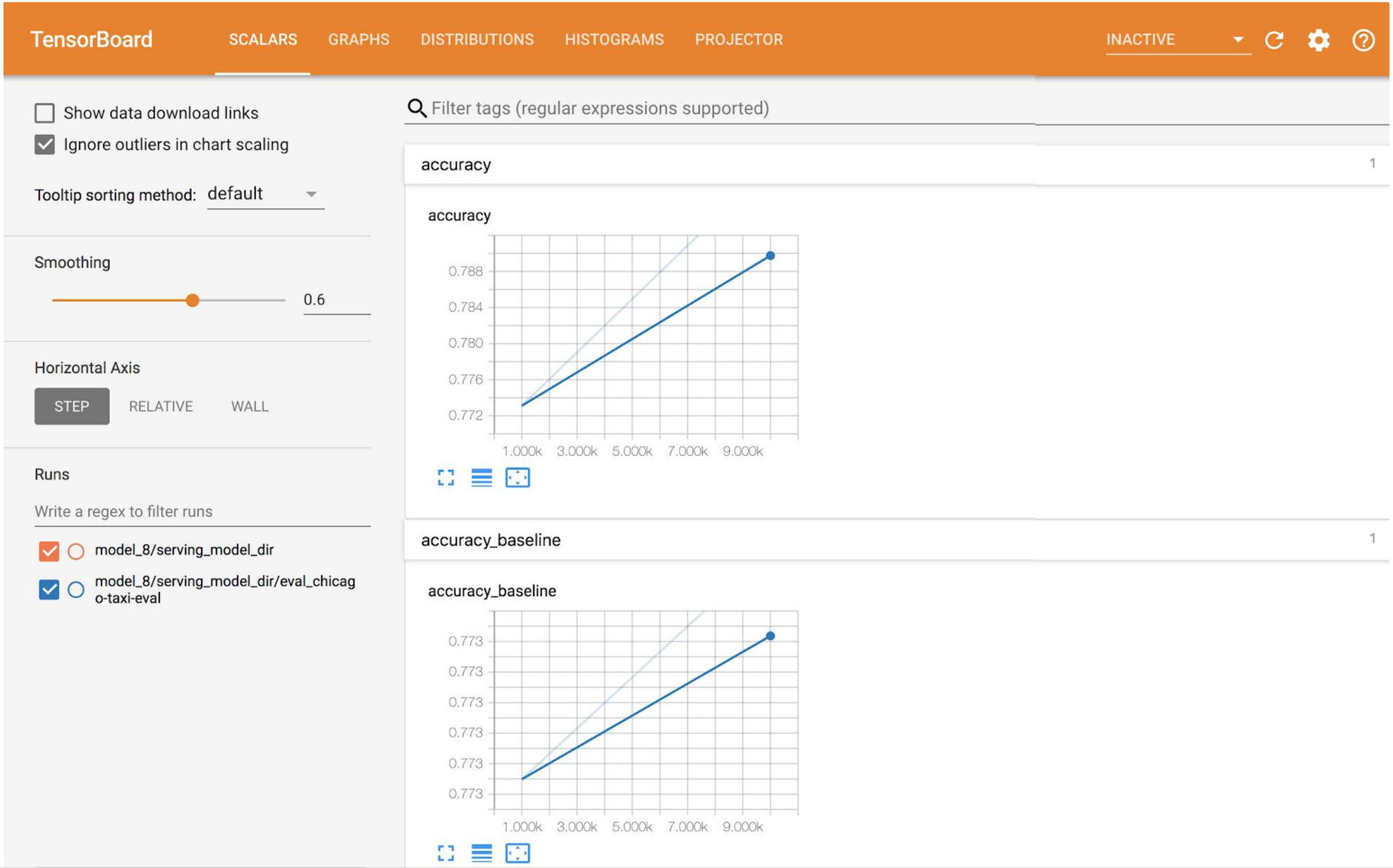
Your model

- TensorFlow (Keras or Estimator)
- Sci-Kit Learn
- With Cloud AI Platform
  - XGBoost
  - PyTorch

Advantages to using TensorFlow

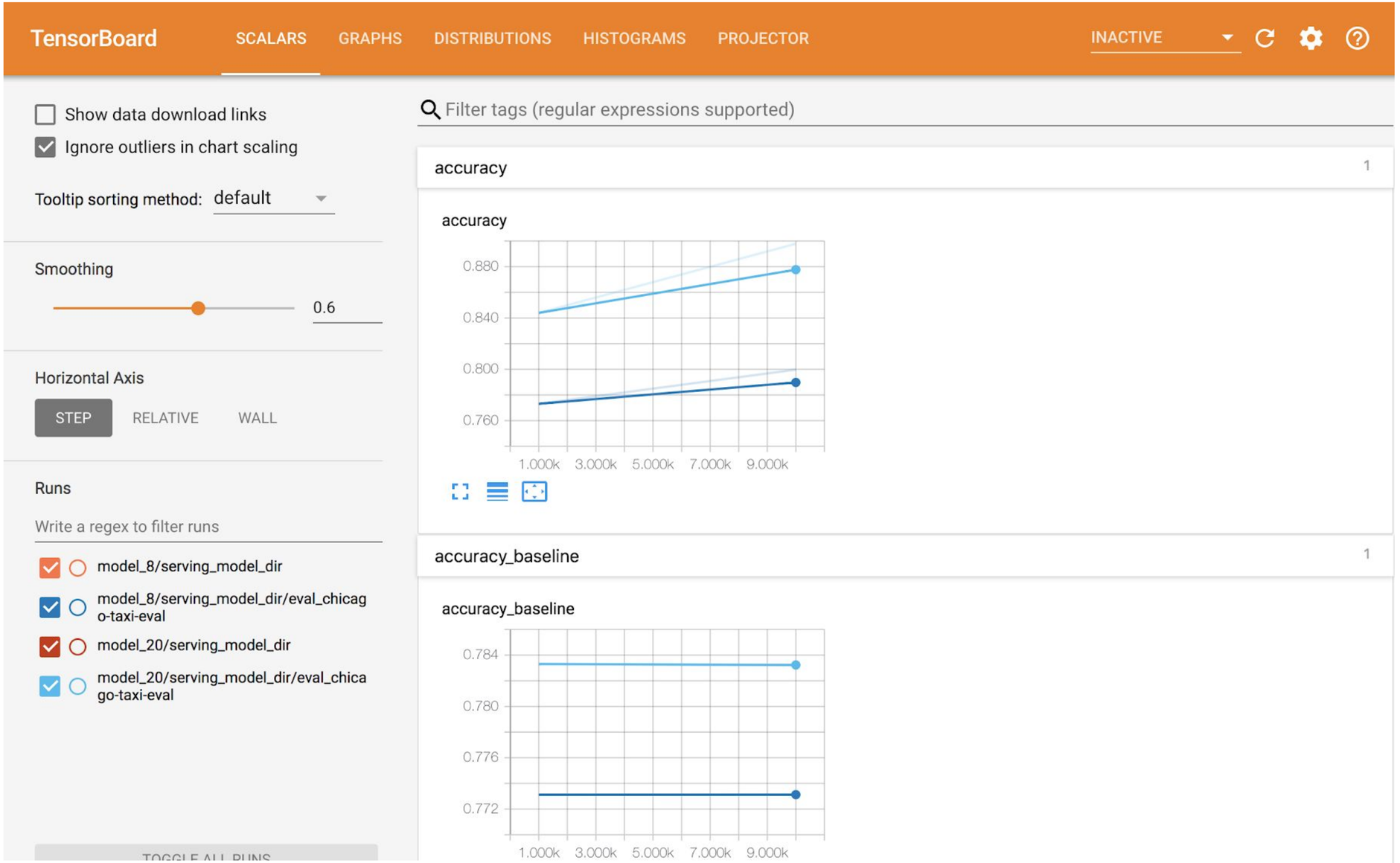
```
# Open up Tensorboard for model_id.
print(display_tensorboard(model_id))
```

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



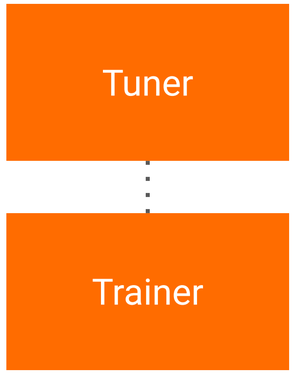
```
# Compare Tensorboard metrics for different models.
if num_models > 1:
    print(display_tensorboard(model_id, other_model_id=other_model_id))
```

http://your.host.name:53230



Component: Tuner

Tuner works with Trainer



Configuration

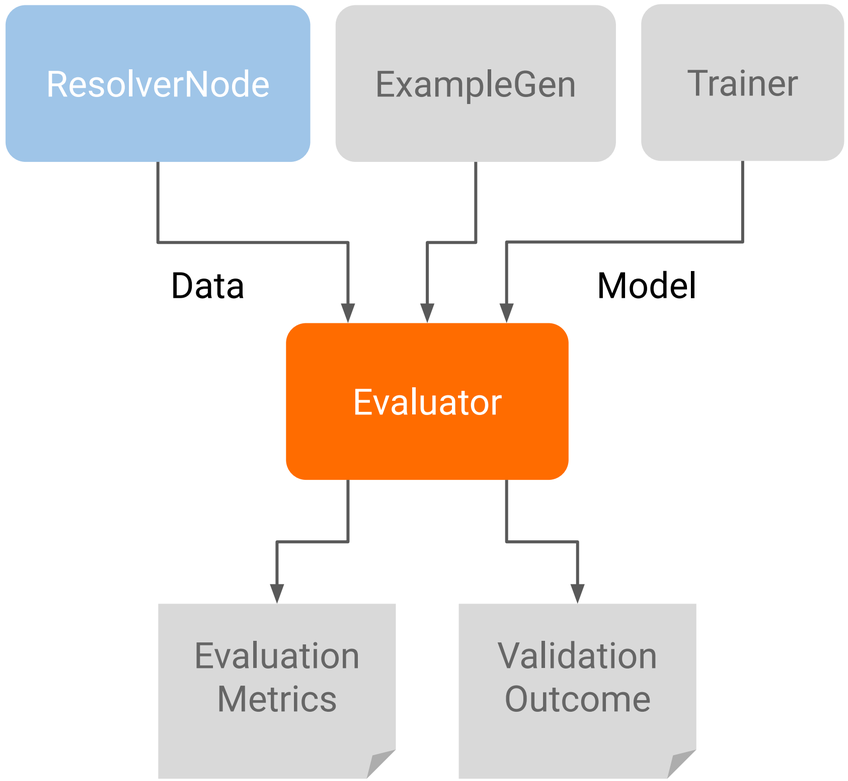
```
tuner = Tuner(
    module_file=module_file,  # Contains `tuner_fn`.
    examples=transform.outputs['transformed_examples'],
    transform_graph=transform.outputs['transform_graph'],
    train_args=trainer_pb2.TrainArgs(num_steps=20),
    eval_args=trainer_pb2.EvalArgs(num_steps=5))
```

Adding to Pipeline

```
trainer = Trainer(
    module_file=module_file,  # Contains `run_fn`.
    custom_executor_spec=executor_spec.
        ExecutorClassSpec(GenericExecutor),
    examples=transform.outputs['transformed_examples'],
    transform_graph=transform.outputs['transform_graph'],
    schema=schema_gen.outputs['schema'],
    # This will be passed to `run_fn`.
    hyperparameters=tuner.outputs['best_hyperparameters'],
    train_args=trainer_pb2.TrainArgs(num_steps=100),
    eval_args=trainer_pb2.EvalArgs(num_steps=5))
```

Component: Evaluator

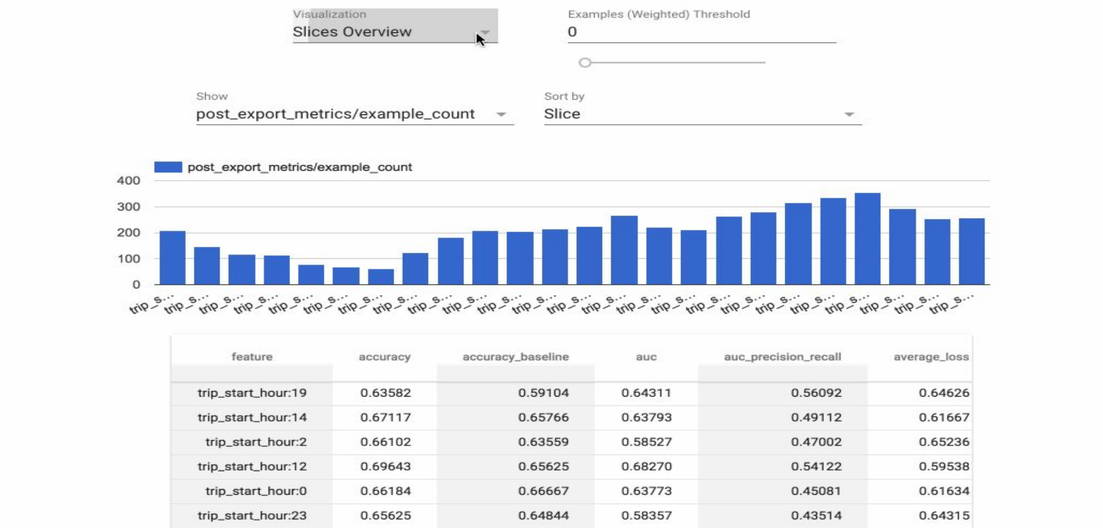
Inputs and Outputs



Configuration

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

Visualization



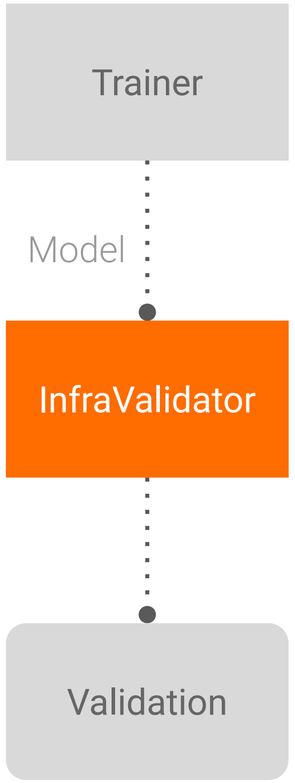
feature	accuracy	accuracy_baseline	auc	auc_precision_recall	average_loss
trip_start_hour:19	0.63582	0.59104	0.64311	0.56092	0.64626
trip_start_hour:14	0.67117	0.65766	0.63793	0.49112	0.61667
trip_start_hour:2	0.66102	0.63559	0.58527	0.47002	0.65236
trip_start_hour:12	0.69643	0.65625	0.68270	0.54122	0.59538
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

beam

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

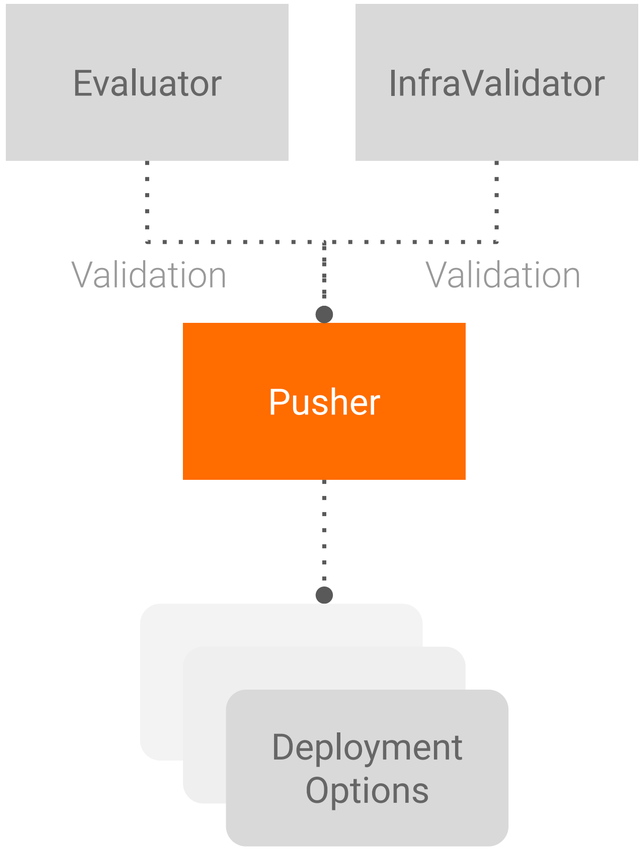
```
infra_validator = InfraValidator(
    model=trainer.outputs['model'],
    serving_spec=ServingSpec(...),
    validation_spec=ValidationSpec(...),
    request_spec=RequestSpec(...)
)
```

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)))
```

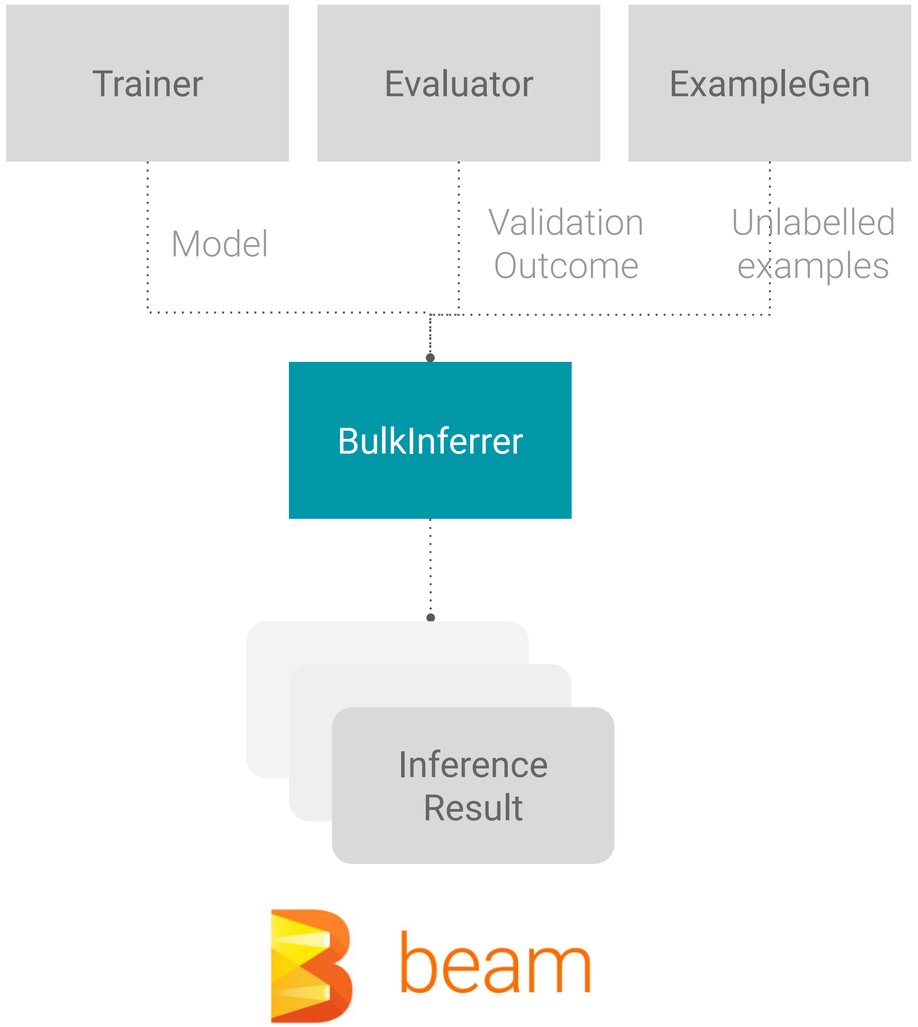
Block push on validation outcome

Push destinations supported today

- Filesystem (TensorFlow Lite, TensorFlow JS)
- TensorFlow Hub
- TensorFlow Serving
- Cloud AI Pipelines
- etc

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.

TFX Pipeline Nodes

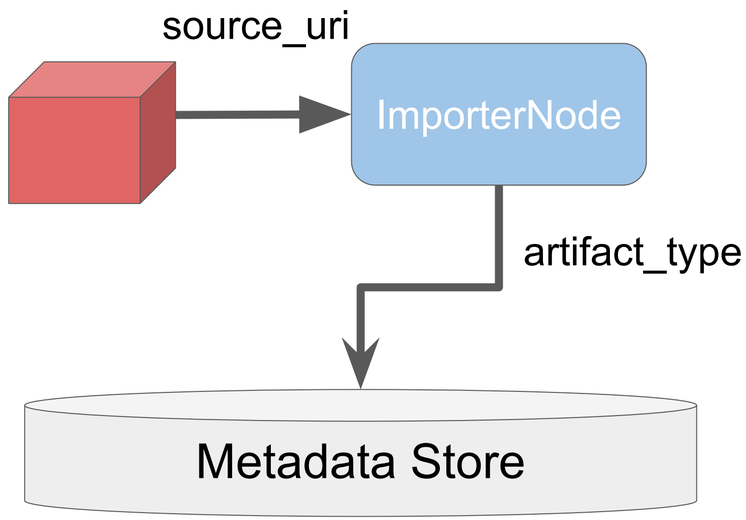
What are Pipeline Nodes?

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 imports an external data object into the metadata store

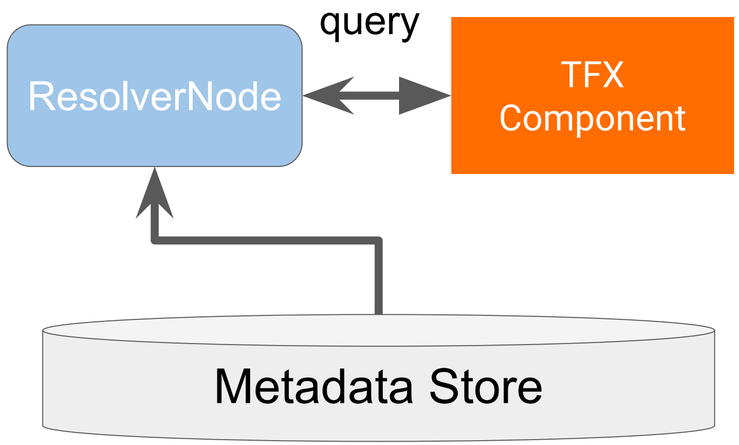


- Key requirements: source_uri, artifact_type

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

ResolverNode

ResolverNode is used to perform metadata 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

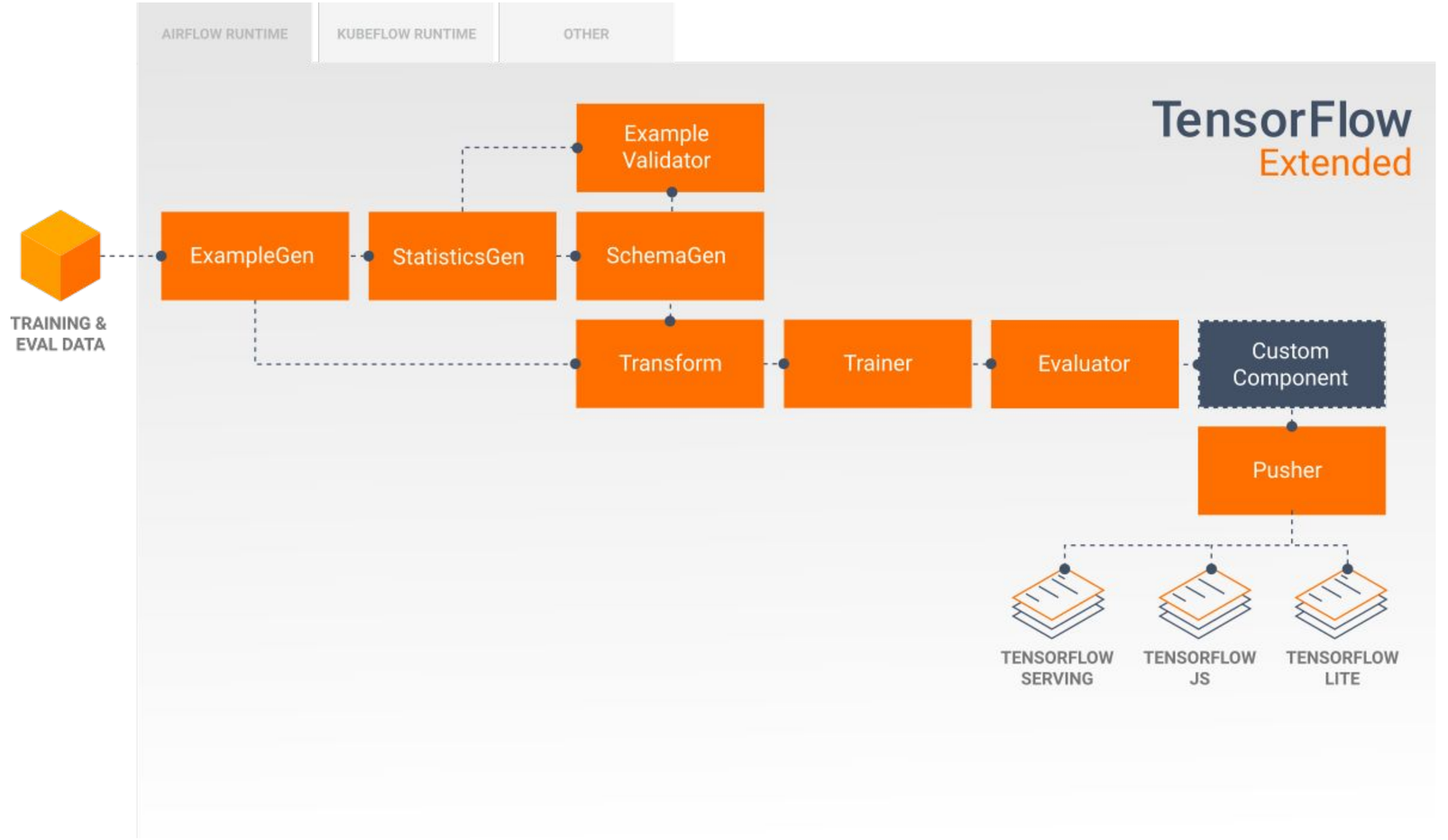
- LatestArtifactsResolver
  - Resolver that returns the latest N artifacts in a given channel
- LatestBlessedModelResolver
  - Returns the latest blessed model

TFX Custom Components

Types of Custom Components

3 ways to create a custom component

- Python function with decorator and annotations
- Components using containers
- Extending existing component classes



Build your own component

Create your own components to run within a TFX pipeline while still providing the benefits of metadata management, lineage, and pipeline monitoring.

Python function-based component

- Decorate a Python function
- Use Python annotations to declare inputs and outputs

```
@component
def MyComponent(
    model: InputArtifact[Model],
    blessing: OutputArtifact[Model],
    accuracy_threshold: Parameter[int] = 10,
    ) -> OutputDict(accuracy=float):

  # Component code here

  return {
    'something': something
  }
```

Container-based component

- Configure inputs, outputs, parameters, and the image
- Wrap with `container_component.create_container_component`

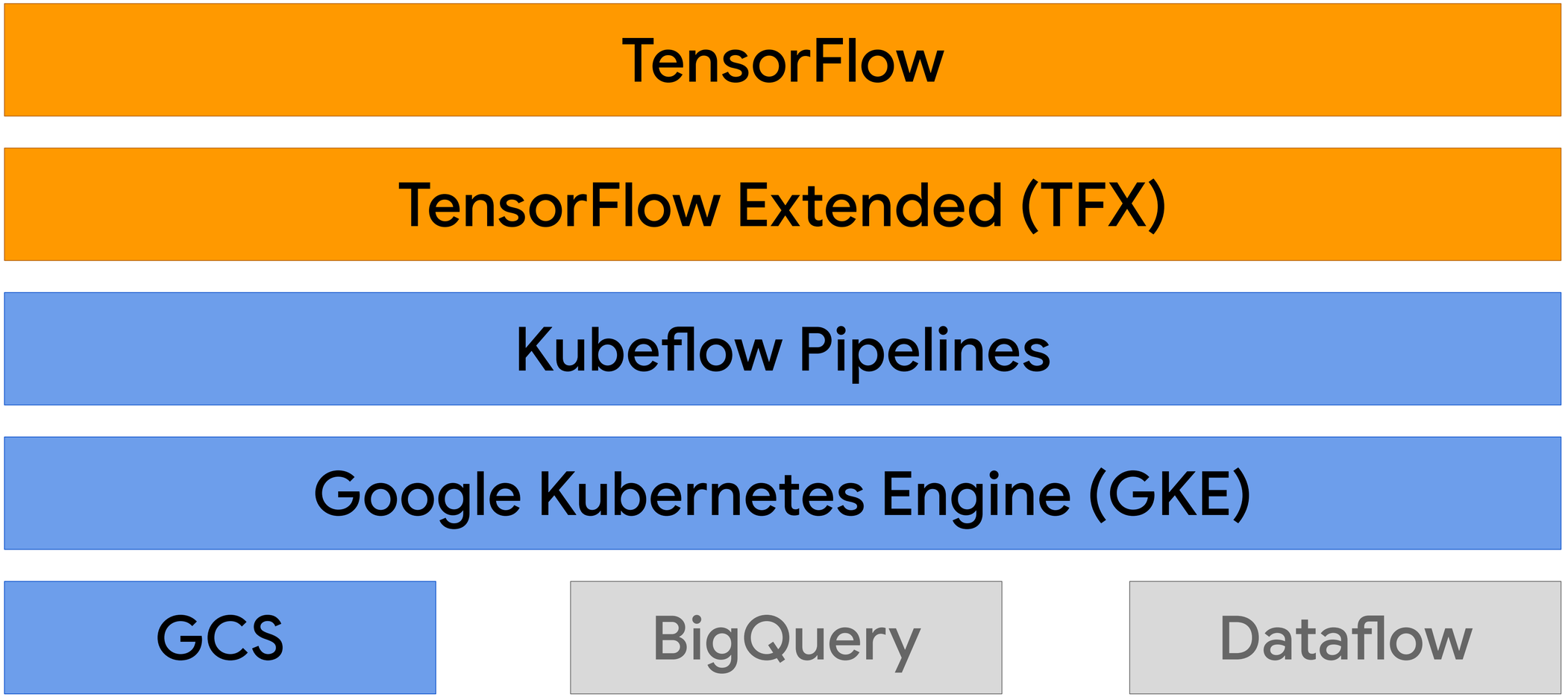
```
from tfx.dsl.component.experimental import container_component
from tfx.dsl.component.experimental import placeholders
from tfx.types import standard_artifacts

grep_component = container_component.create_container_component(
    name='...',
    inputs={ 'text': standard_artifacts.ExternalArtifact },
    outputs={ 'filtered_text': standard_artifacts.ExternalArtifact },
    parameters={ ... },

    image='google/cloud-sdk:278.0.0', # Image to run
    command=[ # Command to run inside container
        'sh', '-exc',
        '''
          # Body of command
        ''',
    ],
)
```

AI Platform

Cloud AI Platform Pipelines

TFX and Kubeflow Pipelines in GCP

Very High Level Architecture



TensorFlow Extended (TFX)

Standard components for your production model needs

Flexible orchestration and metadata

Extensible with custom components

TFX | TensorFlow

https://www.tensorflow.org/tfx

TensorFlow Install Learn API Resources Community Why TensorFlow Search GitHub

For Production

Overview Tutorials Guide API

Apply to speak at TensorFlow World. Deadline April 23rd. Propose talk

TensorFlow Extended (TFX) is an end-to-end platform for deploying production ML pipelines

When you're ready to move your models from research to production, use TFX to create and manage a production

https://www.tensorflow.org/tfx

Thank you!

Helpful resources

Web	https://tensorflow.org/tfx
Repo	https://github.com/tensorflow/tfx
Community	https://goo.gle/tfx-group
YouTube	https://goo.gle/tfx-youtube

Robert Crowe
TensorFlow Developer Engineer

@robert_crowe