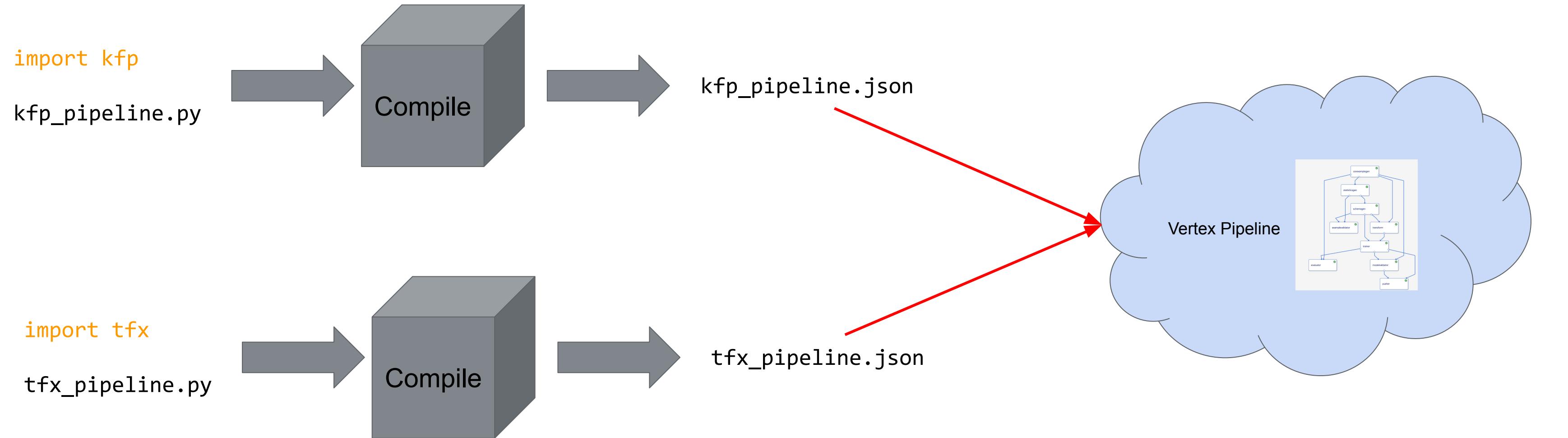


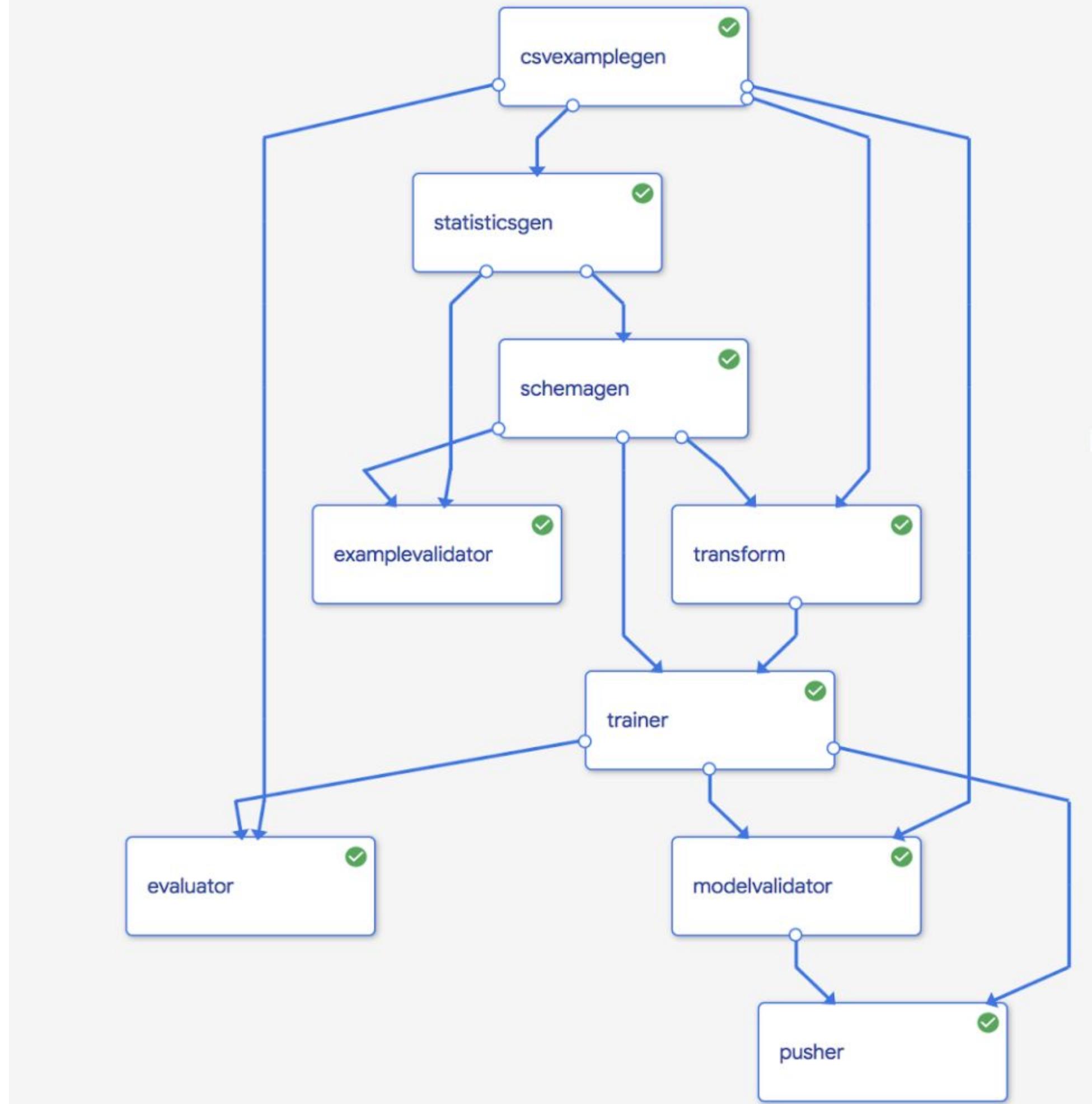


TFX Walkthrough





- Pre-built ML pipeline from Google Best Practices
- Each component can run individually in the notebook for debugging purposes!

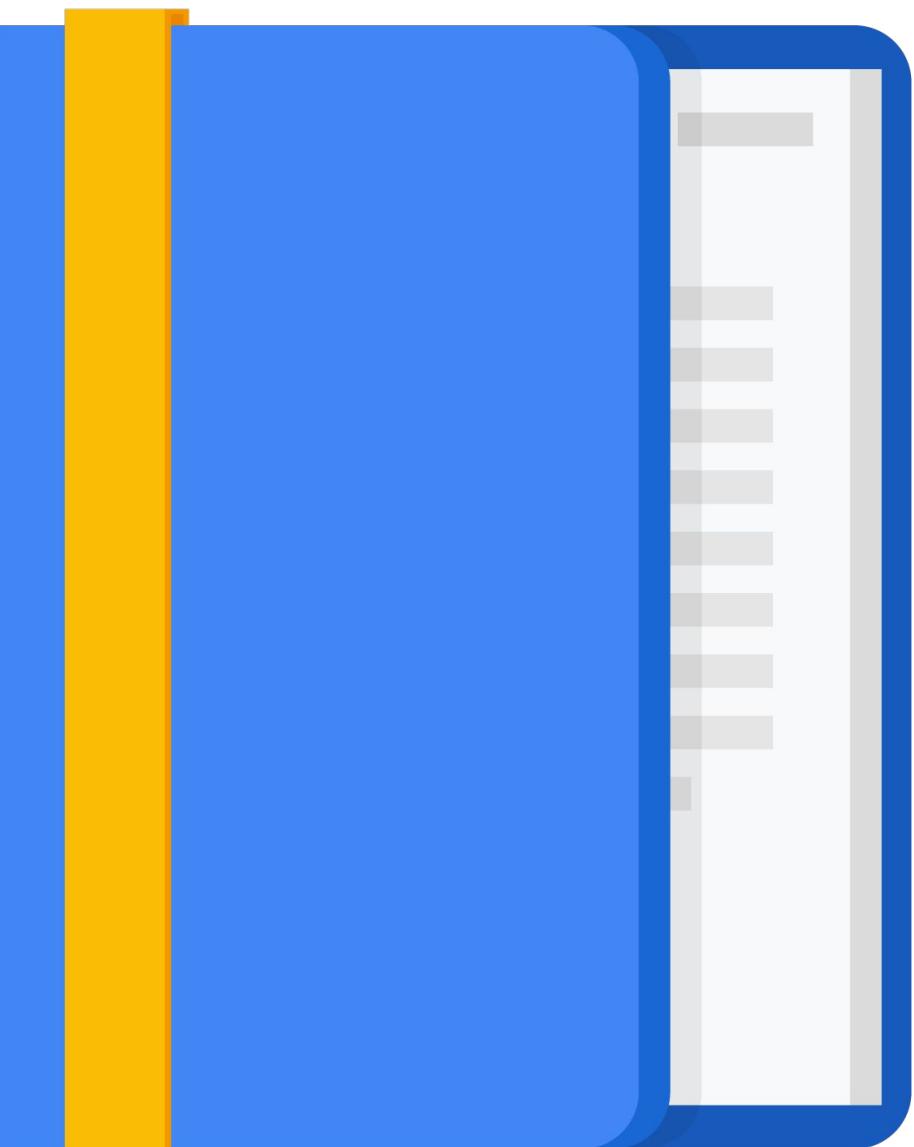


Agenda

TensorFlow Extended (TFX)

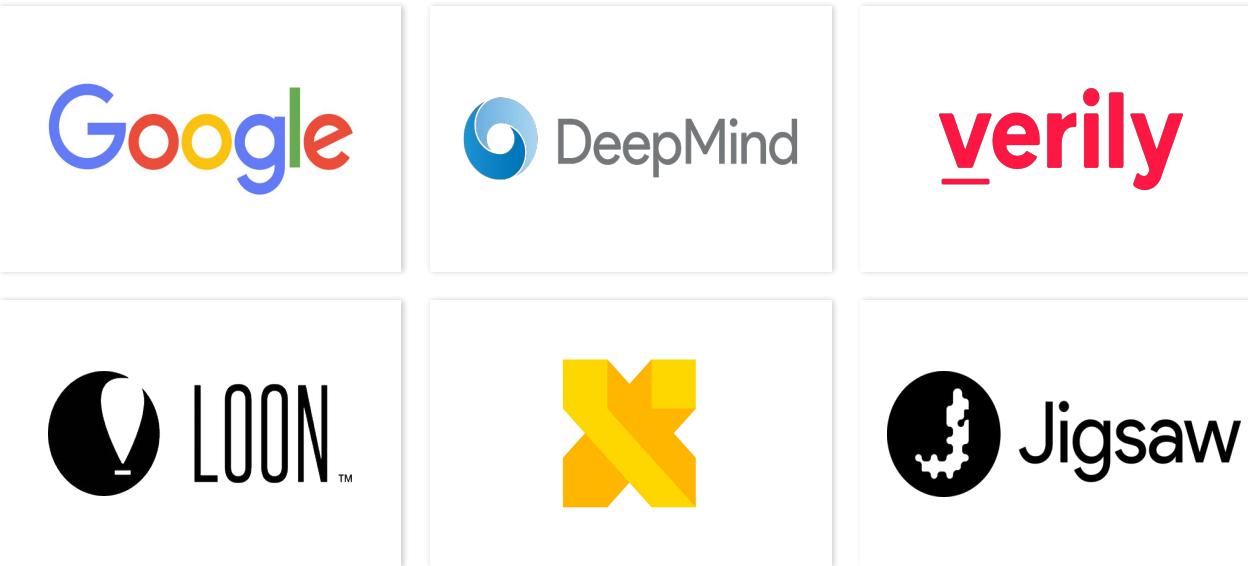
TFX standard components

TFX libraries

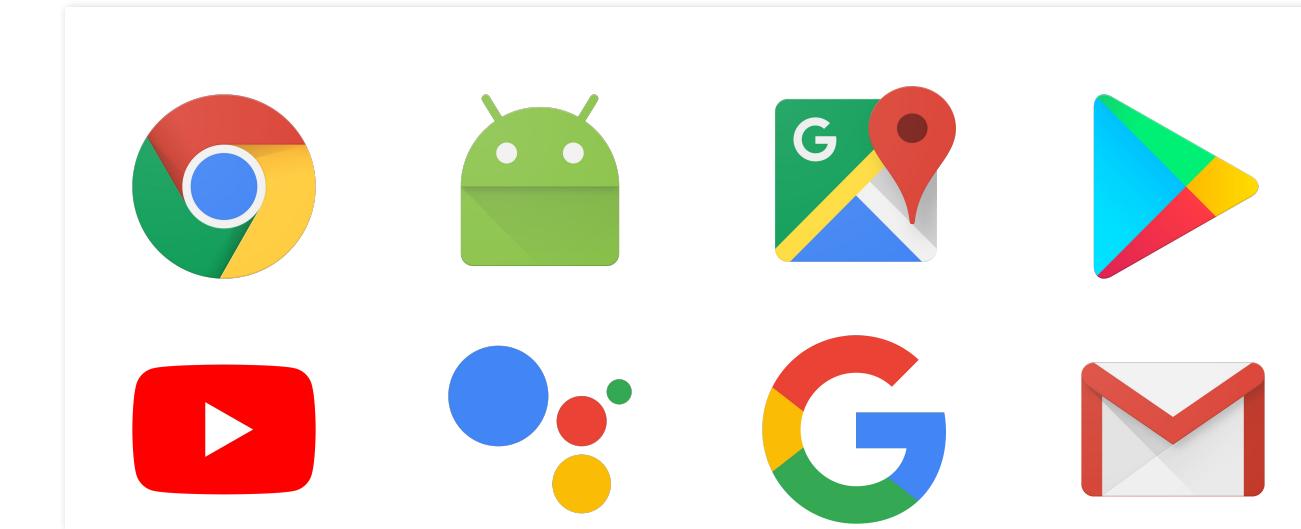


TFX powers Alphabets most important bets and products

AlphaBets



Google Products



Google Cloud AI Products

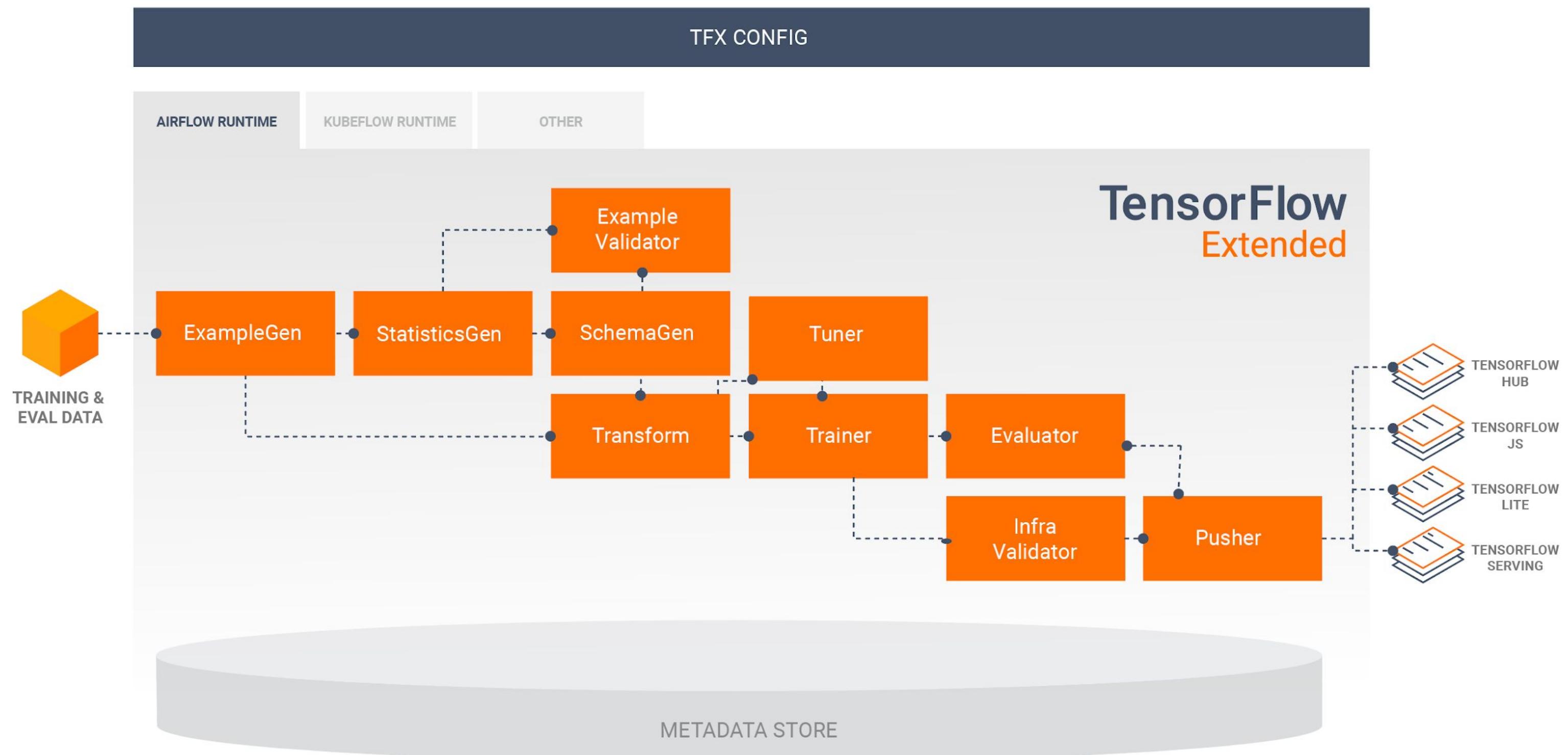
TFX powered



TFX integrations



TFX: Google's production machine learning platform



OSS TFX enables global ML use cases

“... 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

SAP Concur

Etsy

Airbus

Twitter

PayPal

Tencent

NetEase

Yahoo Japan

JD.Com

TFX's Lineage

Sibyl

(2007 - 2019) goo.gle/sibyl-video

User Focused, Production ML, Massive Scale,
(Somewhat) Flexible.

Once upon a time, one of the most widely used E2E
ML platforms at Google.

TFX

(2016 - Present) tensorflow.org/tfx

User Focused, Production ML, Massive Scale,
Flexible, Modular, Portable, Substrate-y.

The most widely used E2E ML platform at **Alphabet**
(including Google).

E2E OSS ML platform **OnPrem** and on **GCP**

TFX has Google's ML best practices built in

- Machine Learning: The High Interest Credit Card of Technical Debt. NeurIPS (2015).
- TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. KDD (2017).
- Data Management Challenges in Production Machine Learning. SIGMOD (2017).
- Rules of Machine Learning: Best Practices for ML Engineering. Google AI Web (2017).
- Continuous Training for Production ML in the TFX Platform. OpML (2019).
- Slice Finder: Automated Data Slicing for Model Validation. ICDE (2019).
- Data Validation for Machine Learning. SysML (2019).

The High-Interest Credit Card of Technical Debt. NeurIPS (2015).
Denis Baylor, Eric Breck, Heng Li, Salem Haykal, Mustafa Ispir, Clemens Mewald, Akshay Narayan, Steven Euijong Whang, Martin Zinkevich
D. Sculley, Todd Phillips, {dsculley, tdp} @toddphilli...
Machine learning often comes for free. This is as coming for free, markedly easy to implement when applying machine learning specifications where possible. The loops, undeclared or...

TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. KDD (2017).
Denis Baylor, Eric Breck, Heng Li, Salem Haykal, Mustafa Ispir, Clemens Mewald, Akshay Narayan, Steven Euijong Whang, Martin Zinkevich
D. Sculley, Todd Phillips, {dsculley, tdp} @toddphilli...
Machine learning often comes for free. This is as coming for free, markedly easy to implement when applying machine learning specifications where possible. The loops, undeclared or...

Data Management Challenges in Production Machine Learning. SIGMOD (2017).
Neoklis Polyzotis, Sudip Roy, Martin Zinkevich
nnpolyzotis, sudip.roy, mzinkevich@...
ABSTRACT
Creating and maintaining a platform for reliable and deploying machine learning models require orchestration of many components—a learner for models based on training data, modules for analyzing both data as well as models, and finally infrastructure for serving models in production. This becomes challenging when data changes over time and needs to be produced continuously. Unfortunately, orchestration is often done ad hoc using glue code scripts developed by individual teams for specific needs, leading to duplicated effort and fragile system technical debt.
We present TensorFlow Extended (TFX), a general-purpose machine learning platform at Google. By integrating the aforementioned components into one platform, we were able to standardize, simplify the platform configuration, and time to production from the order of months to days, undecleared or...

Best Practices for Machine Learning. Google AI Web (2017).
Denis Baylor, Eric Breck, Heng Li, Salem Haykal, Mustafa Ispir, Clemens Mewald, Akshay Narayan, Steven Euijong Whang, Martin Zinkevich
D. Sculley, Todd Phillips, {dsculley, tdp} @toddphilli...
Machine learning often comes for free. This is as coming for free, markedly easy to implement when applying machine learning specifications where possible. The loops, undeclared or...

Continuous Training for Production Machine Learning. OpML (2019).
Denis Baylor, Kevin Haas, Rose Liu, Clemens Mewald, Mitchell Welling, Yeounoh Chung, Tim Kraska, Neoklis Polyzotis
Denis Baylor, Kevin Haas, Rose Liu, Clemens Mewald, Mitchell Welling, Yeounoh Chung, Tim Kraska, Neoklis Polyzotis
ABSTRACT
As machine learning systems become deployed in production, however, current data tools are still primitive. We focus on the particular problem of slicing data. This is an important problem in model validation because it can find arbitrary slices, our goal is to find interpretable, problematic and large. We propose Slice Finder, which finds slices that are problematic. Applications include diagnosing model failure. Human intervention is crucial. This research is part of a larger trend of opportunities for new research.

Automated Data Slicing for Model Validation. ICDE (2019).
Denis Baylor, Kevin Haas, Rose Liu, Clemens Mewald, Mitchell Welling, Yeounoh Chung, Tim Kraska, Neoklis Polyzotis
Denis Baylor, Kevin Haas, Rose Liu, Clemens Mewald, Mitchell Welling, Yeounoh Chung, Tim Kraska, Neoklis Polyzotis
ABSTRACT
Large organizations rely increasingly on continuous pipelines in order to keep machine-learned models up-to-date with respect to data. In these pipelines, disruptions in the pipeline can increase model staleness, thus degrading the quality of downstream services such as these models. In this paper we describe the operational challenges of running continuous pipelines in the Tensorflow Extended (TFX) system that we developed and deployed at Google. We present the main mechanisms in TFX to support this type of production and the lessons learned from the deployment of the platform internally at Google.

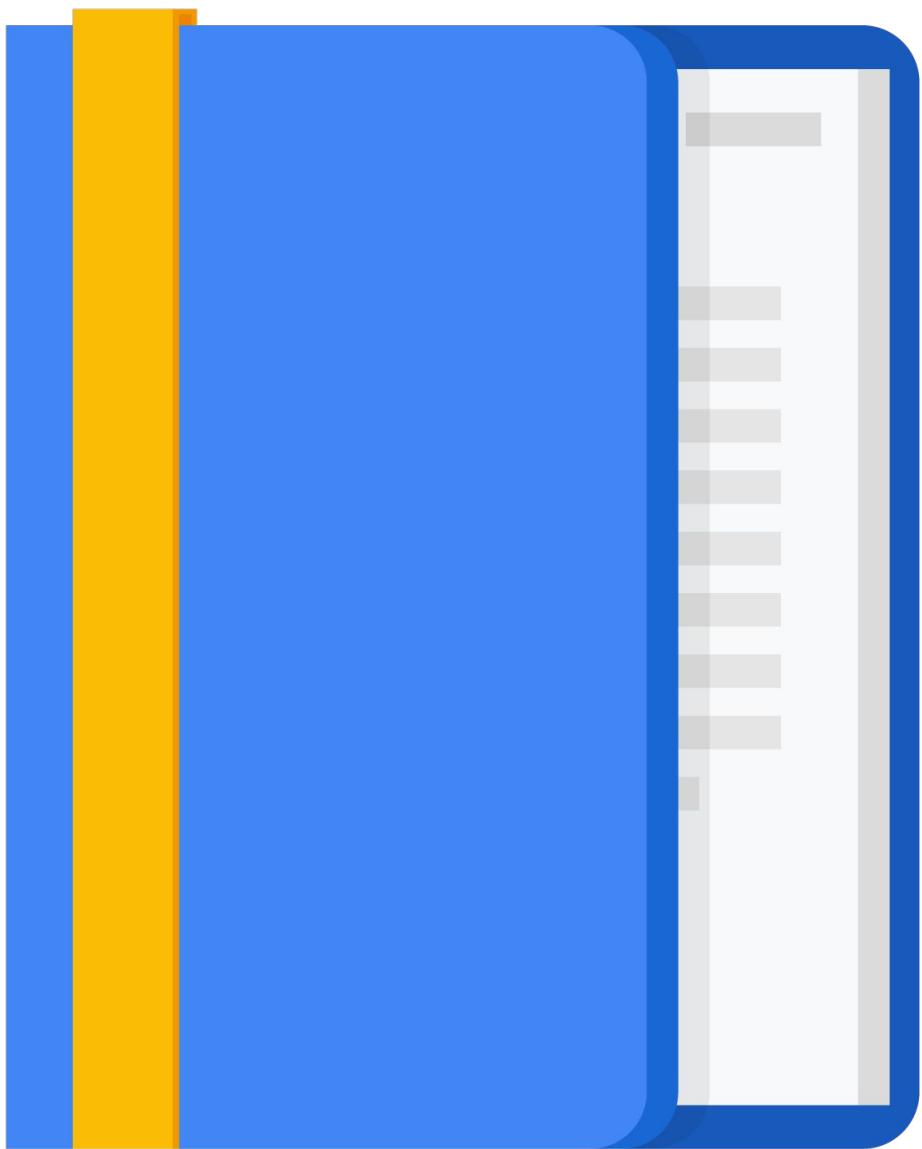
DATA VALIDATION FOR MACHINE LEARNING. SysML (2019).
Eric Breck¹, Neoklis Polyzotis¹, Sudip Roy¹, Steven Euijong Whang², Martin Zinkevich¹
ABSTRACT
Machine learning is a powerful tool for gleaning knowledge from massive amounts of data. While a great deal of machine learning research has focused on improving the accuracy and efficiency of training and inference algorithms, there is less attention in the equally important problem of monitoring the quality of data fed to machine learning. The importance of this problem is hard to dispute: errors in the input data can nullify any benefits on speed and accuracy for training and inference. This argument points to a data-centric approach to machine learning that treats training and serving data as an important production asset, on par with the algorithm and infrastructure used for learning.
In this paper, we tackle this problem and present a data validation system that is designed to detect anomalies specifically in data fed into machine learning pipelines. This system is deployed in production as an integral part of TFX(Baylor et al., 2017) – an end-to-end machine learning platform at Google. It is used by hundreds of product teams use it to continuously monitor and validate several petabytes of production data per day. We faced several challenges in developing our system, most notably around the ability of ML pipelines to soldier on in the face of unexpected patterns, schema-free data, or training/serving skew. We discuss these challenges, the techniques we used to address them, and the various design choices that we made in implementing the system. Finally, we present evidence from the system’s deployment in production that illustrate the tangible benefits of data validation in the context of ML: early detection of errors, model-quality wins from using better data, savings in engineering hours to debug problems, and a shift towards data-centric workflows in model development.

Agenda

TensorFlow Extended (TFX)

[TFX standard components](#)

TFX libraries

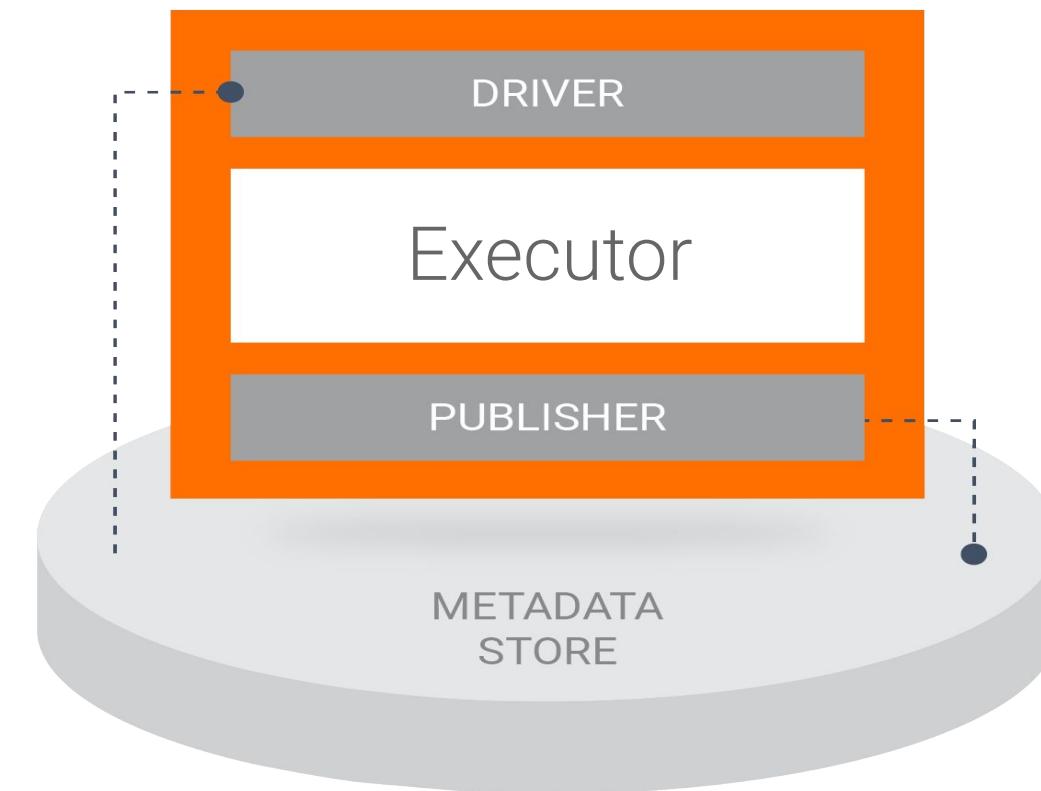


A TFX **component** implements a machine learning task



TFX Component

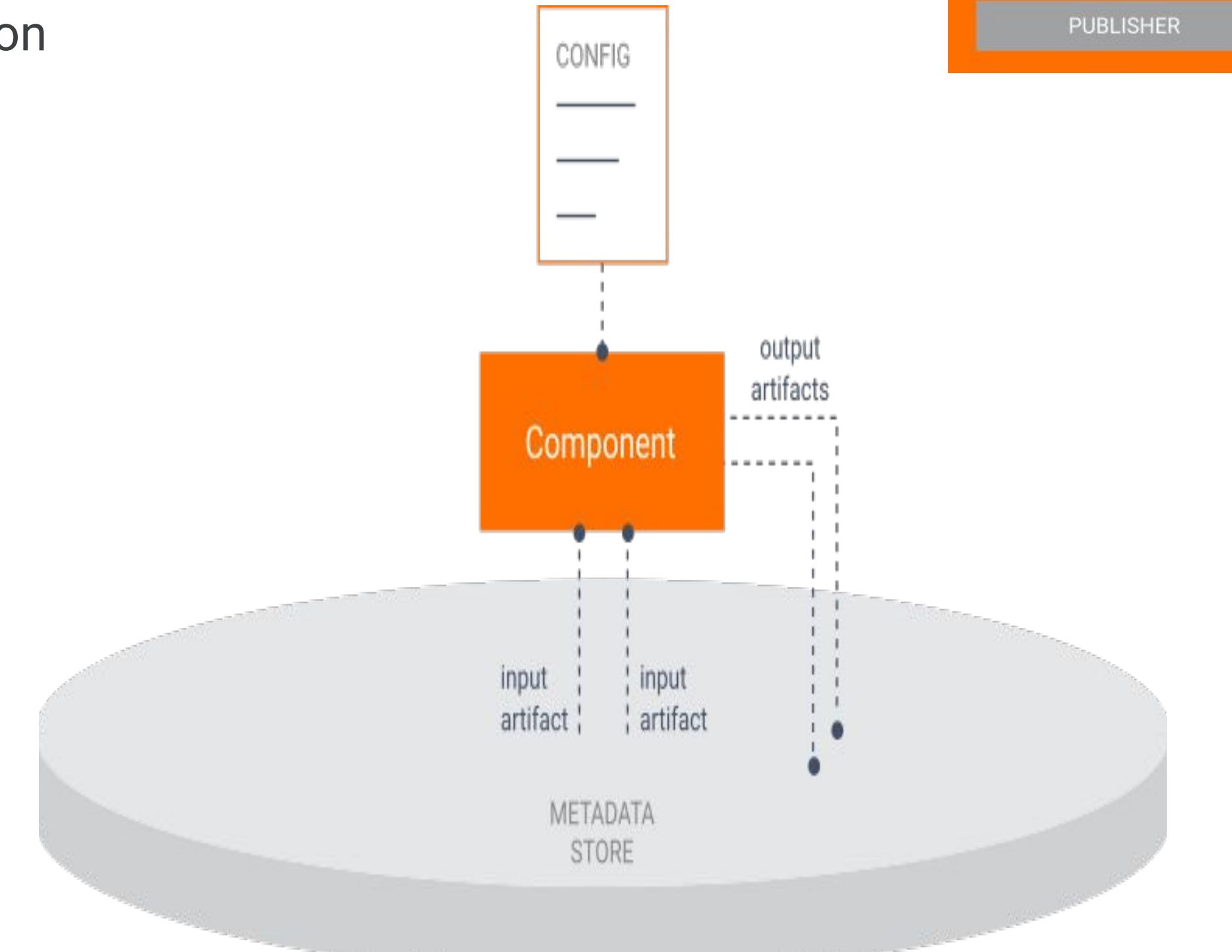
TFX standard components



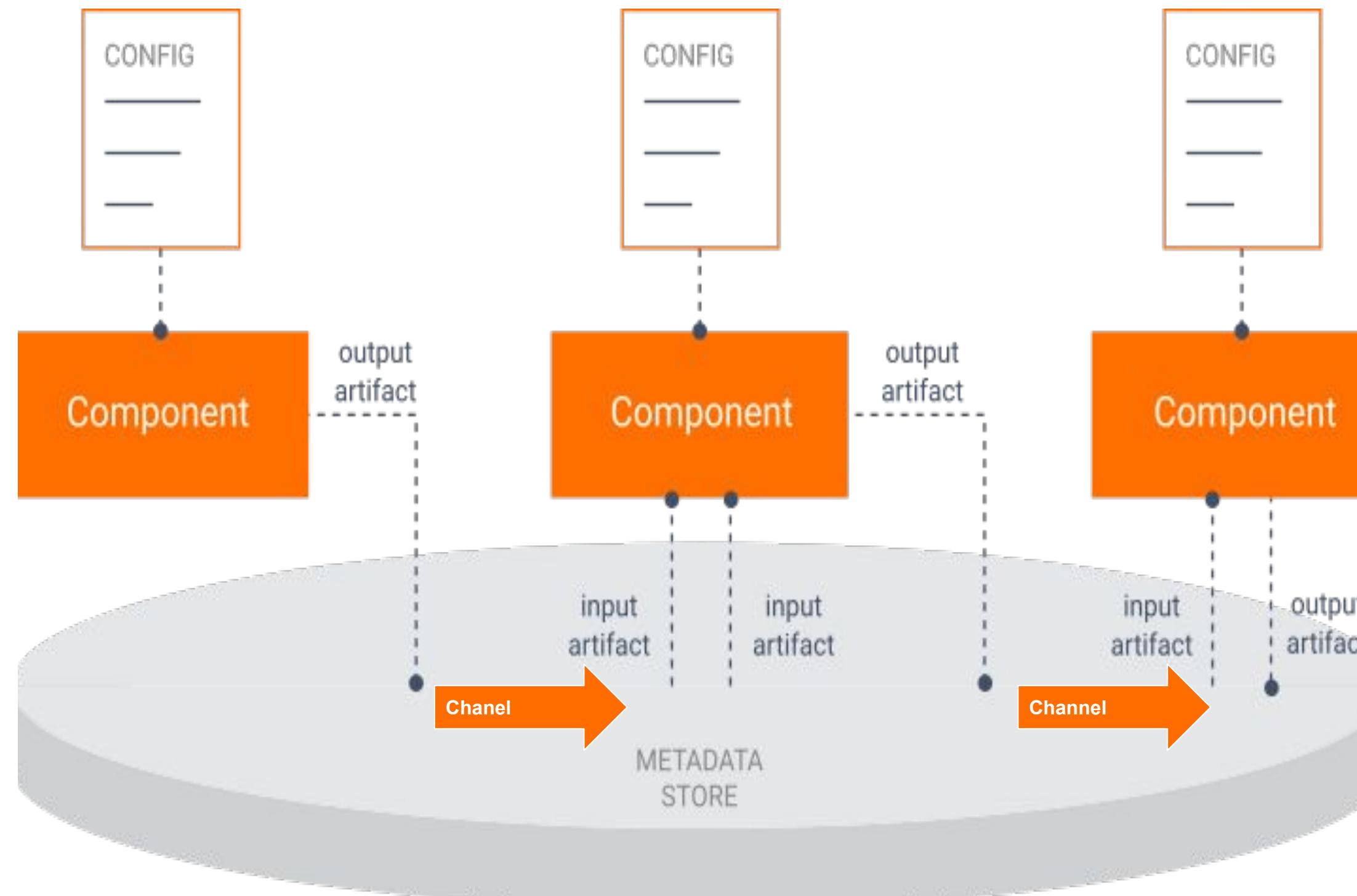
Compose ML pipelines faster with reusable components
pre-configured with production best practices

TFX components at runtime

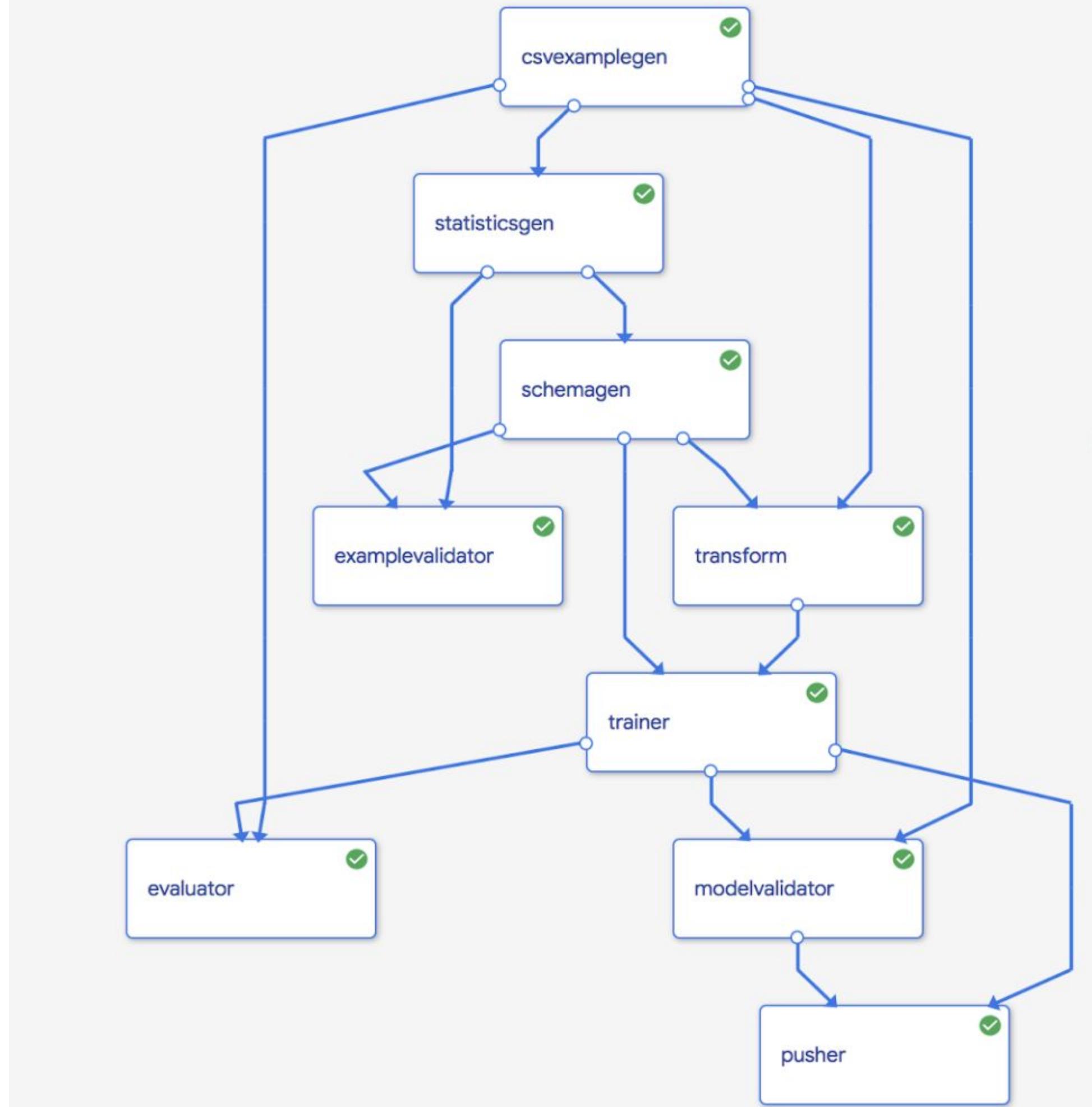
1. **Driver** reads the component specification for parameters and artifacts and retrieves input artifacts from metadata store for the component.
2. **Executor** performs computation on artifacts
3. **Publisher** uses the component specification and executor results to store component's output artifacts in metadata store.



A TFX pipeline is a sequence of **components** connected by **channels** in a directed acyclic graph (DAG) of artifact dependencies

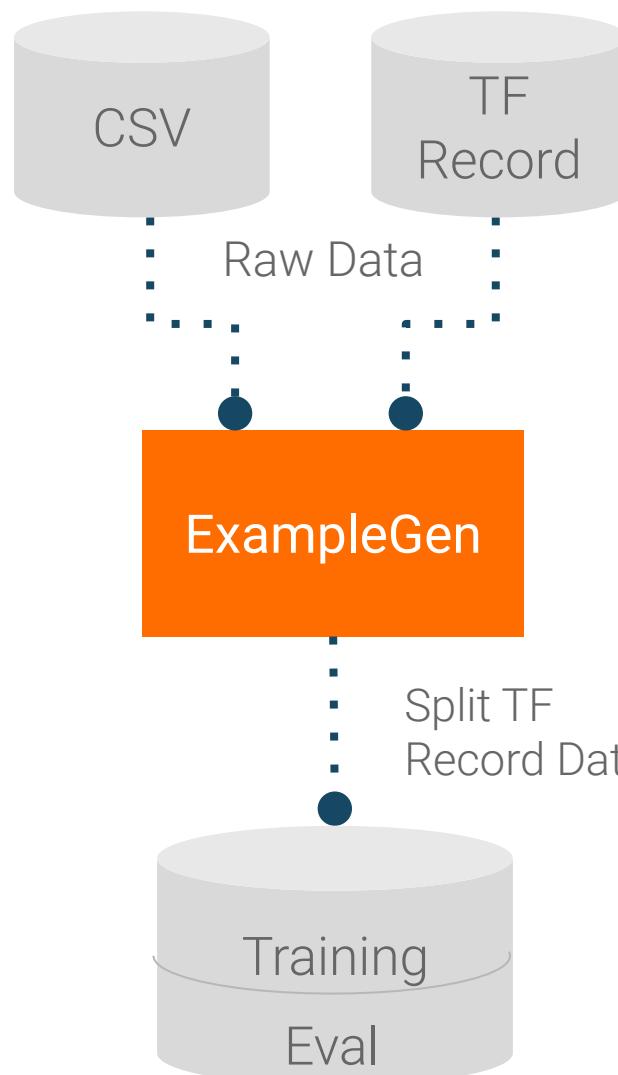


There is a fixed number of TFX components.



Component: ExampleGen

Inputs and Outputs



Configuration

```
output_config = example_gen_pb2.Output(
    split_config=example_gen_pb2.SplitConfig(splits=[
        example_gen_pb2.SplitConfig.Split(name='train', hash_buckets=4),
        example_gen_pb2.SplitConfig.Split(name='eval', hash_buckets=1)
    ])
)

example_gen = tfx.components.CsvExampleGen(
    input=external_input(DATA_ROOT),
    output_config=output_config
).with_id("CsvExampleGen")
```



ML project lifecycle benefits

- Reproducible and configurable data partitioning and shuffling into common data representation
- External CSV, Avro, Parquet, BigQuery, and TFRecord ingestion
- Apache Beam supported for scalable data ingestion
- Customizable to new input data formats and ingestion methods

Component: StatisticsGen

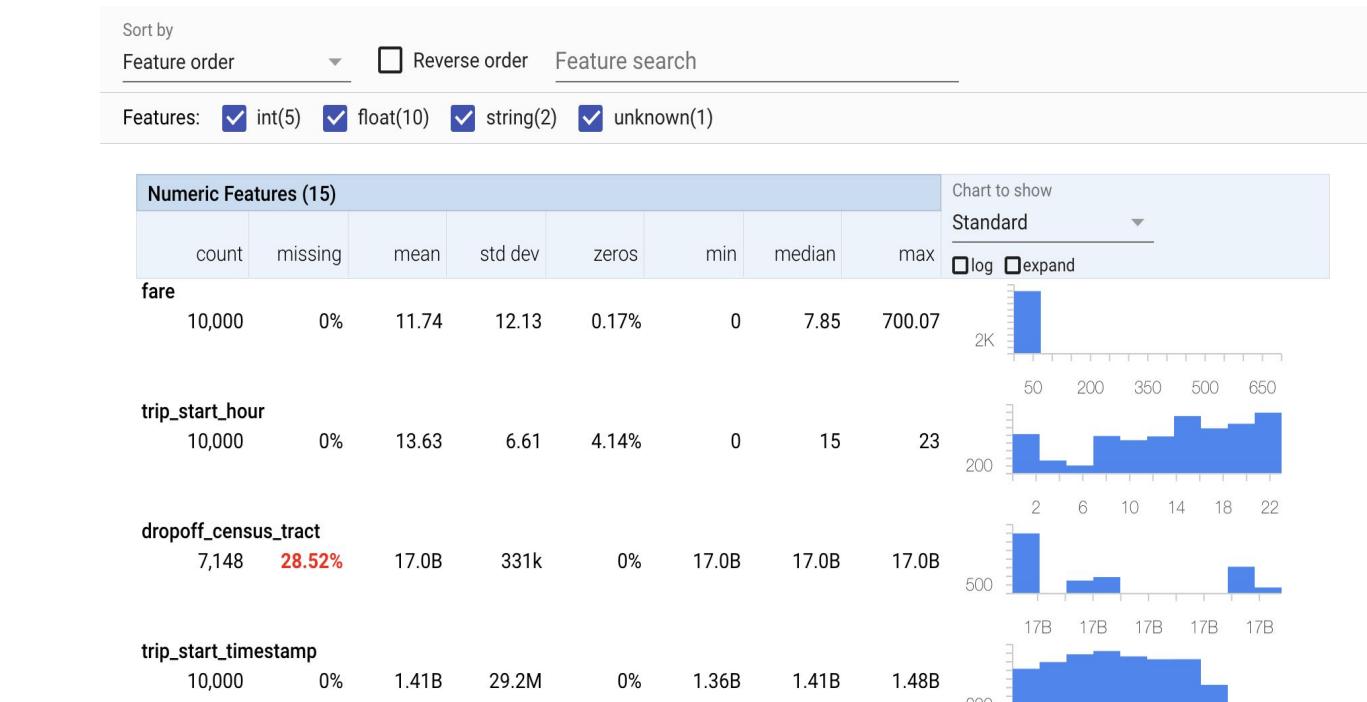
Inputs and Outputs



Configuration

```
statistics_gen = tfx.components.StatisticsGen(  
    examples=example_gen.outputs['examples']).with_id("StatisticsGen")
```

TFDV Feature Statistics Visualization



ML project lifecycle benefits

- TensorFlow Data Validation (TFDV) library for calculating feature statistics
- Scalable full-pass dataset feature statistics processing with Apache Beam

Component: SchemaGen

Inputs and Outputs



Configuration

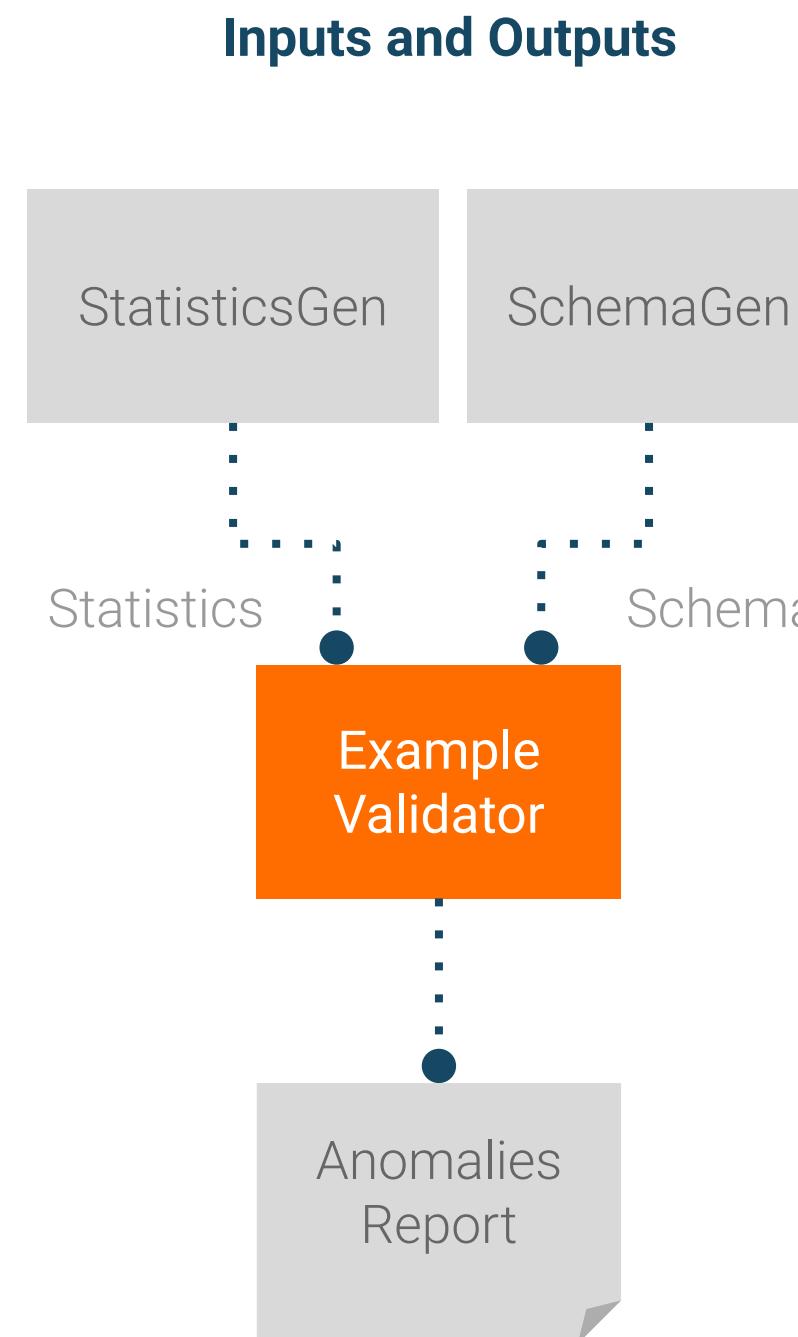
```
schema_gen = tfx.components.SchemaGen(  
    statistics=statistics_gen.outputs['statistics'],  
    infer_feature_shape=False).with_id("SchemaGen")
```

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'

ML project lifecycle benefits

- **TensorFlow Data Validation (TFDV)** creates common data description for pipeline components
- A Schema enables continuous data monitoring and validation during pipeline training

Component: ExampleValidator



Configuration

```
example_validator = tfx.components.ExampleValidator(  
    statistics=statistics_gen.outputs['statistics'],  
    schema=schema_importer.outputs['result'],  
).with_id("ExampleValidator")
```

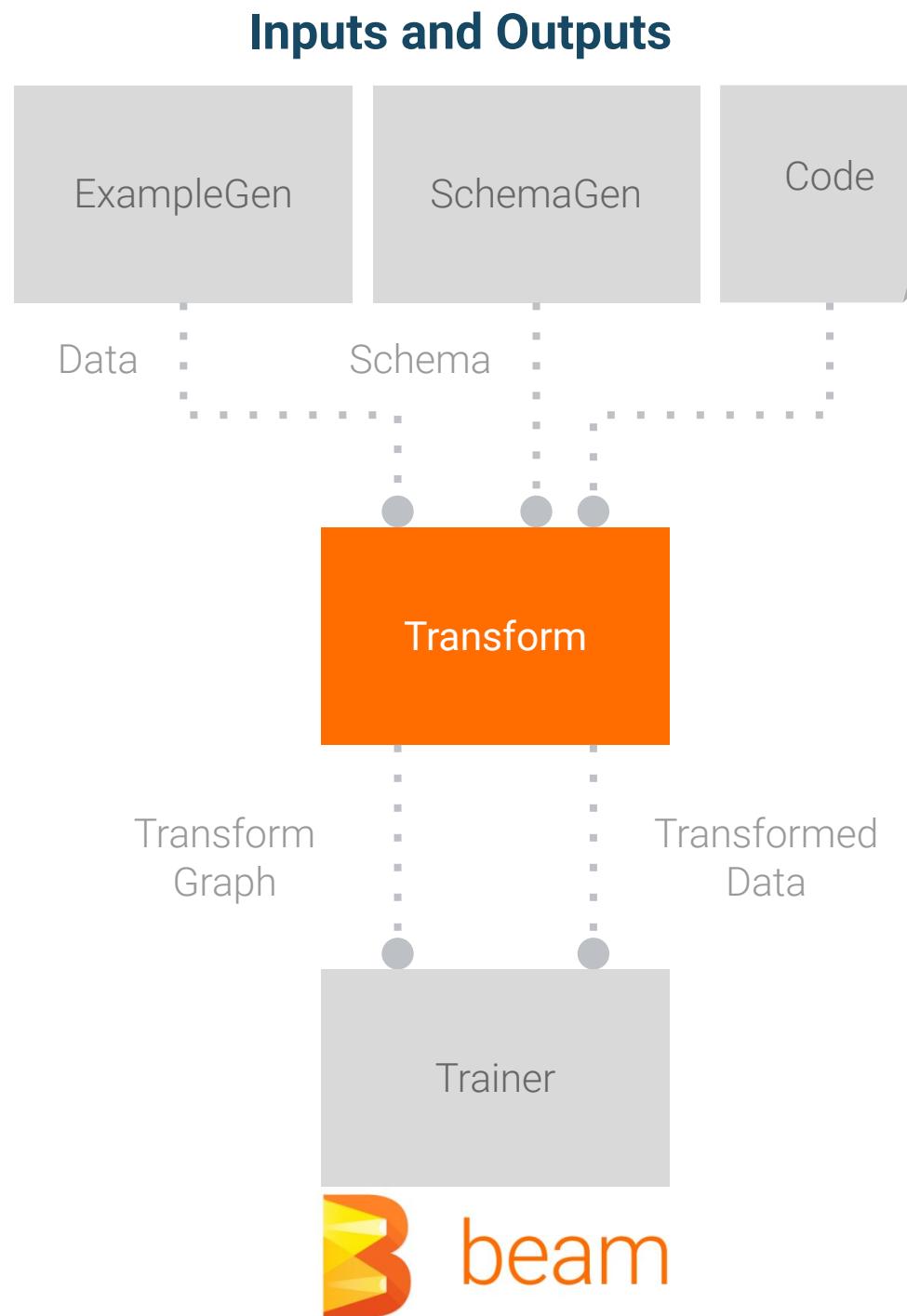
Visualization

Anomaly short description	Anomaly long description
Feature name	
'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%).

ML project lifecycle benefits

- Data monitoring for detection of anomalies, train-serving skew, and data drift

Component: Transform



Configuration

```
transform = tfx.components.Transform(  
    examples=example_gen.outputs['examples'],  
    schema=schema_importer.outputs['result'],  
    module_file=TRANSFORM_MODULE).with_id("Transform")
```

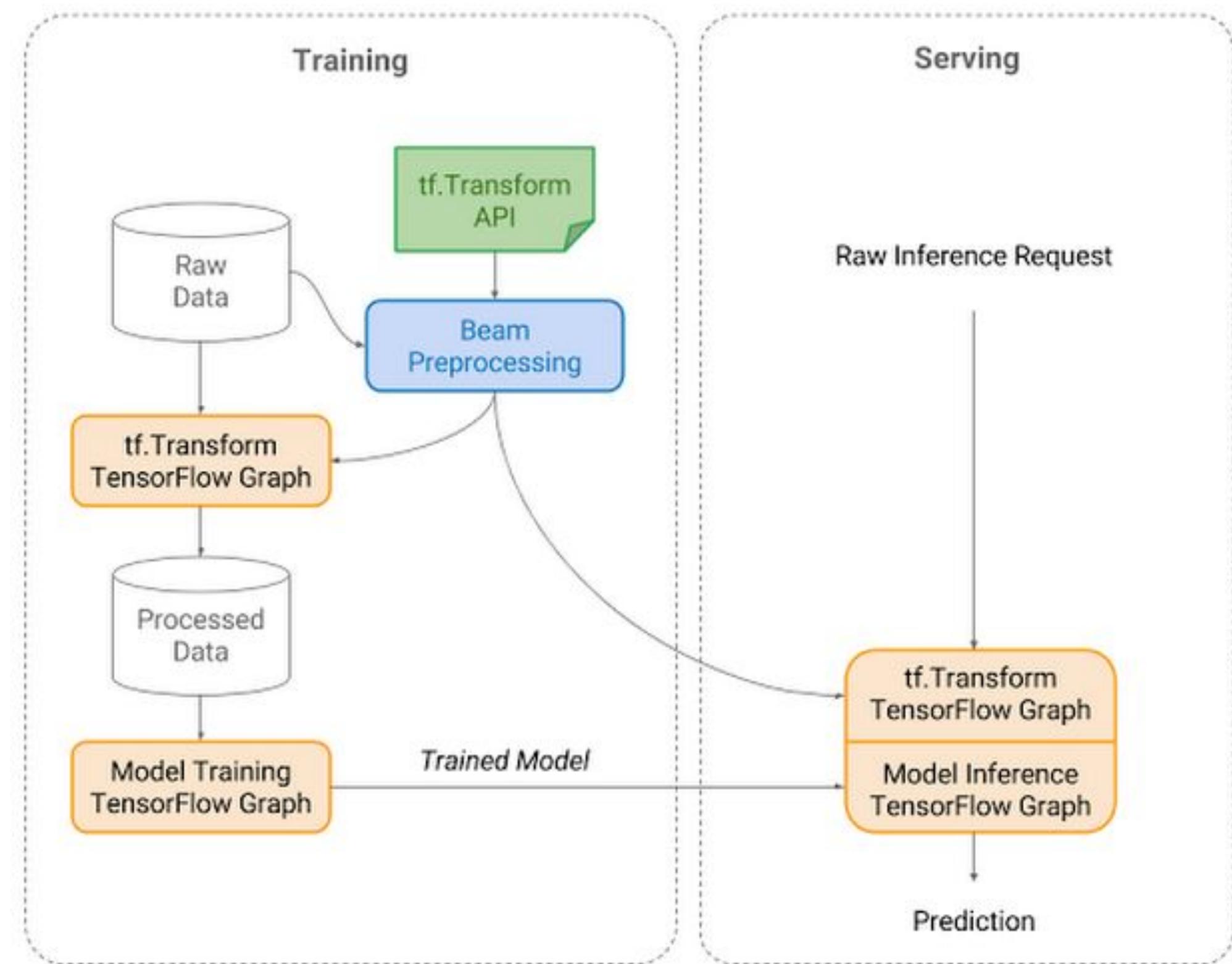
TRANSFORM_MODULE='preprocessing.py'

```
def preprocessing_fn(inputs):  
  
    outputs = {}  
  
    # Scale numerical features  
    for key in features.NUMERIC_FEATURE_KEYS:  
        outputs[features.transformed_name(key)] = tft.scale_to_z_score(  
            _fill_in_missing(inputs[key]))  
  
    # Generate vocabularies and maps categorical features  
    for key in features.CATEGORICAL_FEATURE_KEYS:  
        outputs[features.transformed_name(key)] = tft.compute_and_apply_vocabulary(  
            x=_fill_in_missing(inputs[key]), num_oov_buckets=1, vocab_filename=key)  
  
    outputs[features.transformed_name(features.LABEL_KEY)] = _fill_in_missing(  
        inputs[features.LABEL_KEY])  
  
    return outputs
```

Component: Transform

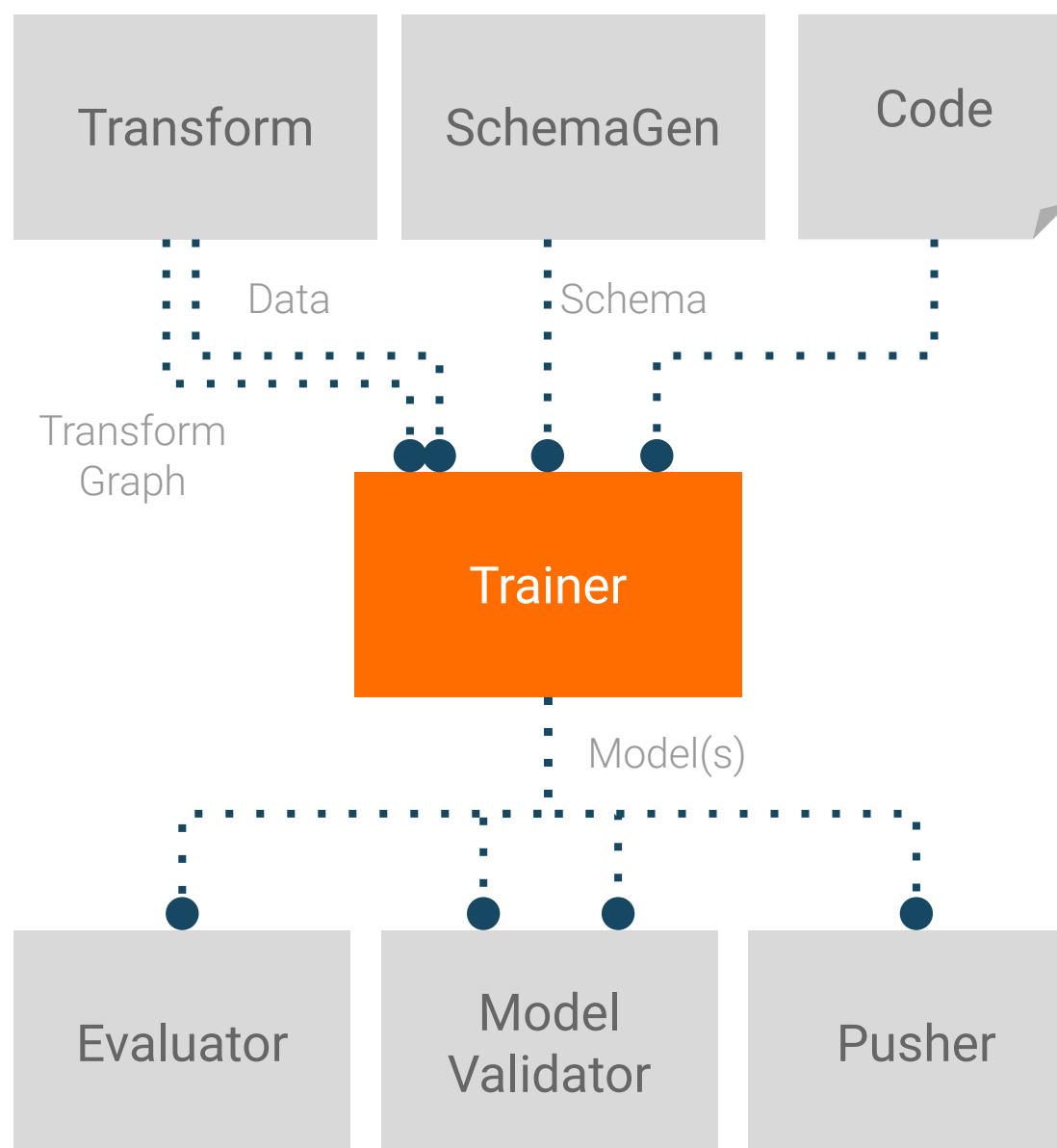
ML project lifecycle benefits

- Generates SavedModel for consistent training and serving feature engineering
- Backed by Apache Beam for scalable distributed data processing to large datasets



Component: Trainer

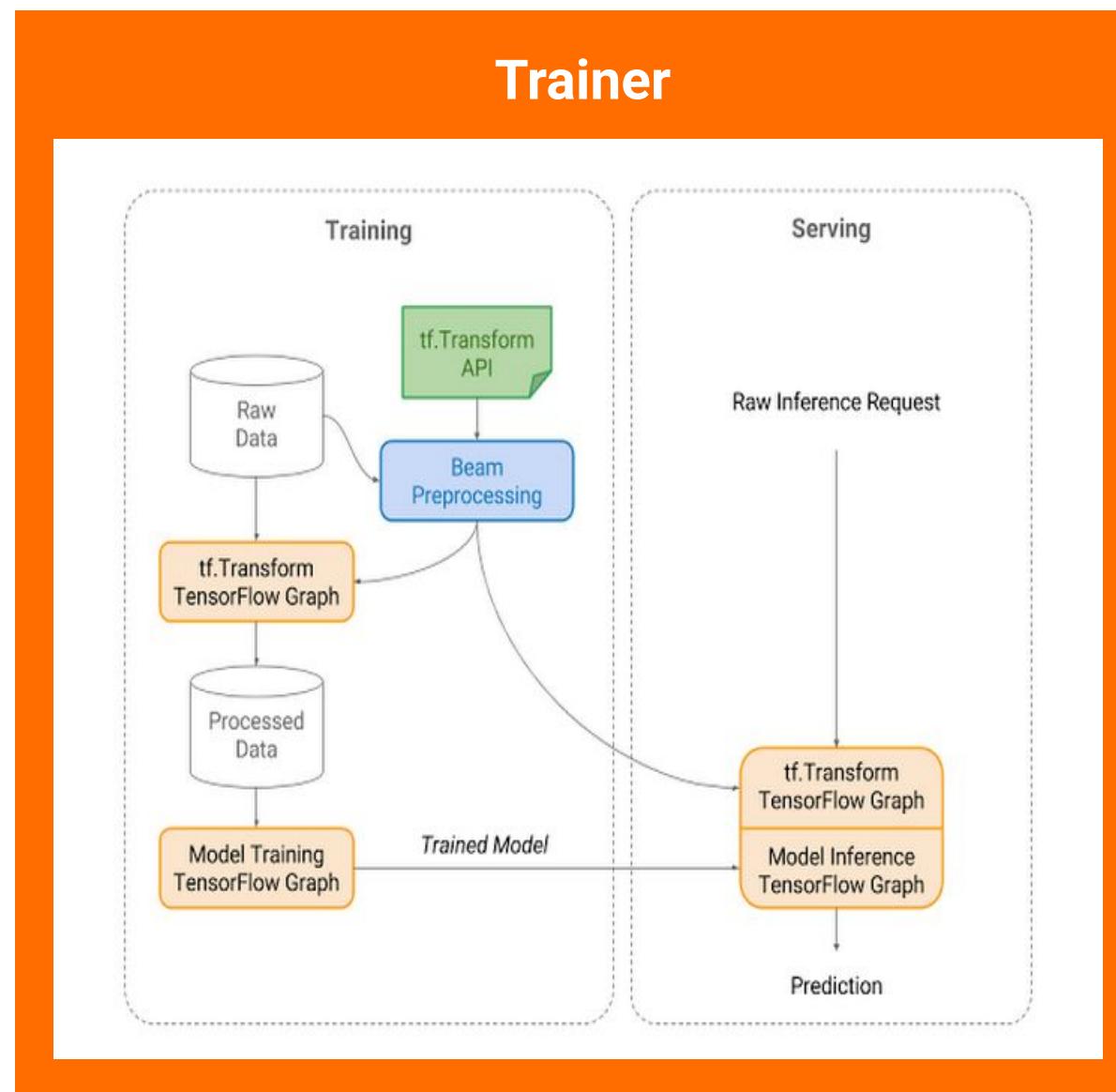
Inputs and Outputs



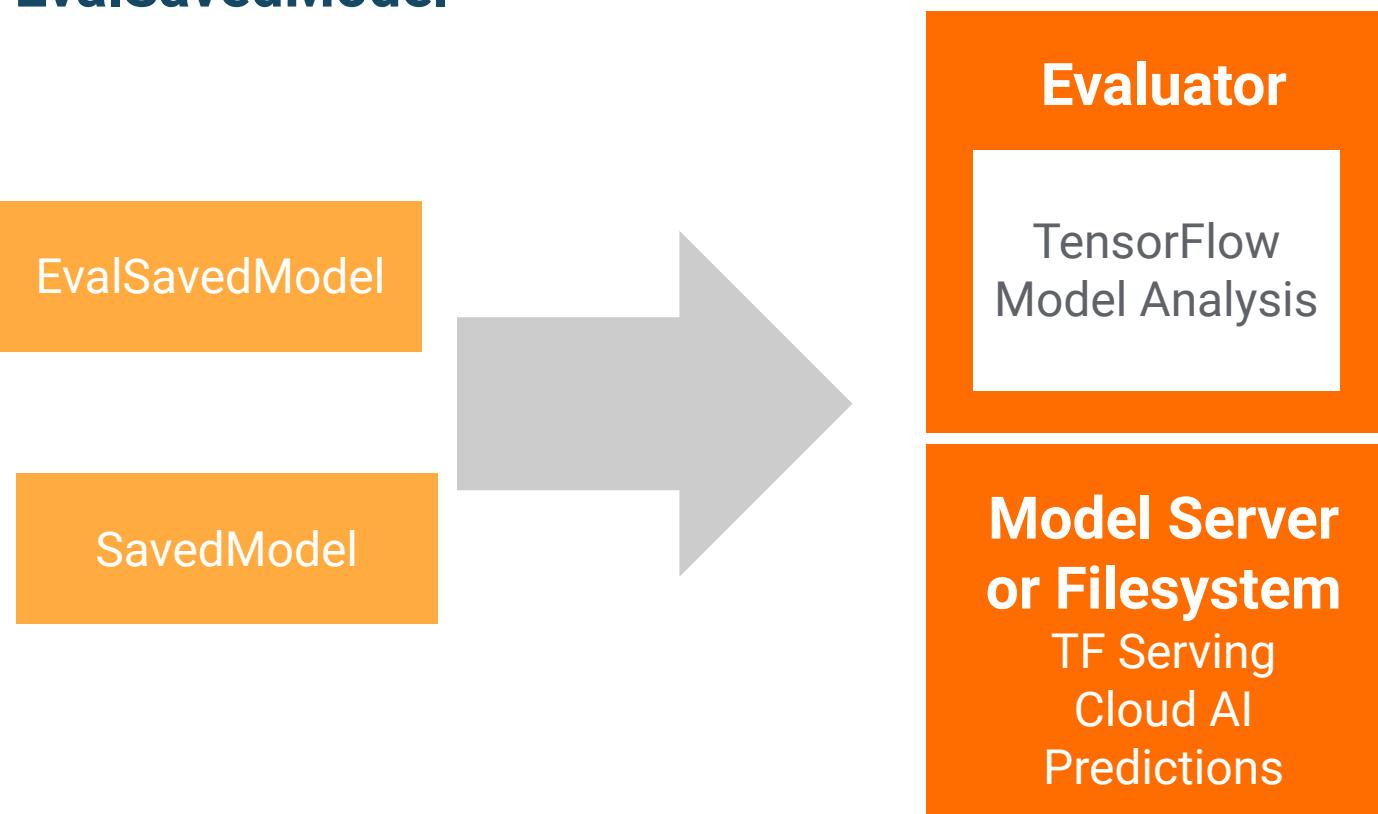
```
executor =  
executor_spec.ExecutorClassSpec(trainer_executor.GenericExecutor)  
  
trainer = tfx.components.Trainer(  
    custom_executor_spec=executor,  
    module_file=model.py,  
    transformed_examples=transform.outputs["transformed_examples"],  
    schema=schema_importer.outputs["result"],  
    transform_graph=transform.outputs["transform_graph"],  
    train_args=trainer_pb2.TrainArgs(num_steps=5000),  
    eval_args=trainer_pb2.EvalArgs(num_steps=1000)  
).with_id("Trainer")
```

Code -> your TensorFlow code in a pipeline/models/model.py

Component: Trainer



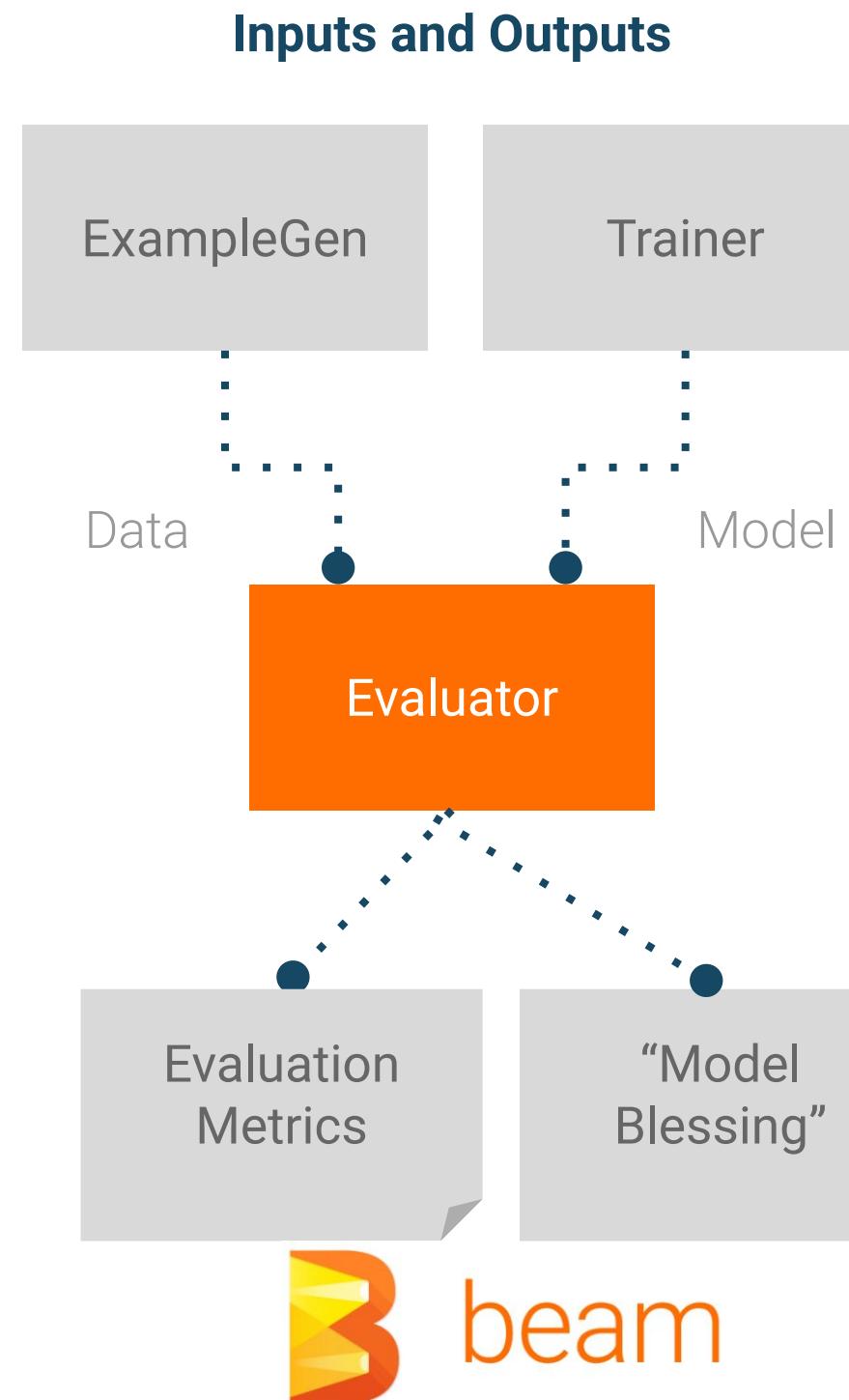
Trainer outputs a trained **SavedModel** and **EvalSavedModel**



ML project lifecycle benefits

- Produces standardized TensorFlow SavedModel model artifact for sharing and easier deployment
- Configurable with Generic Trainer for any TensorFlow model API such Estimator, tf.Keras, and TFLite

Component: Evaluator



```
model_analyzer = tfx.components.Evaluator(  
    examples=example_gen.outputs["examples"],  
    model=trainer.outputs["model"],  
    baseline_model=model_resolver.outputs["model"],  
    eval_config=eval_config).with_id("ModelEvaluator")
```

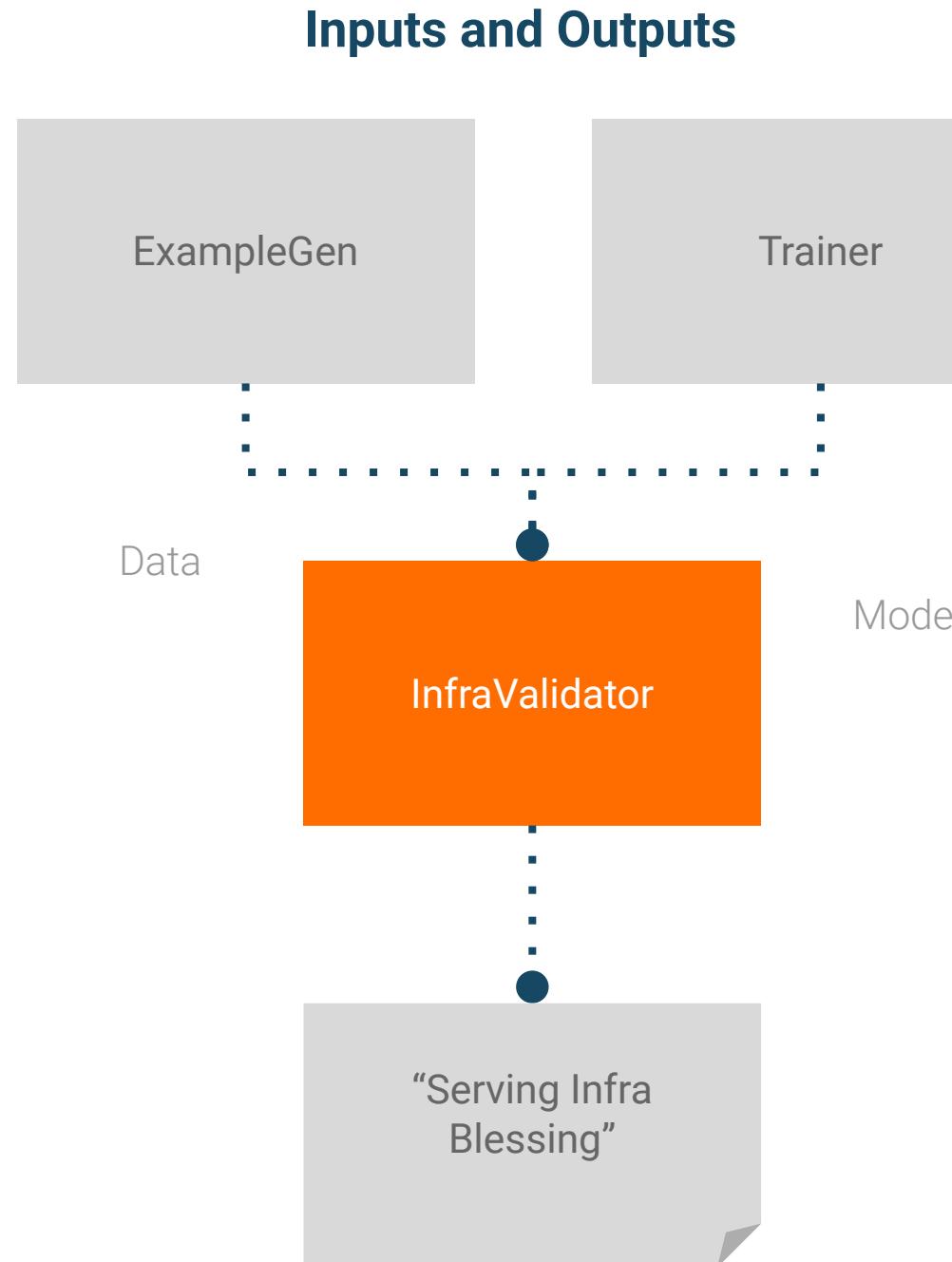
TensorFlow Model Analysis (TFMA) library for visualizing model evaluation



ML project lifecycle benefits

- Uses TensorFlow Model Analysis (TFMA) and Apache Beam for scalable model evaluation across data splits and slices.
- Validate model performance “good enough” to be pushed to production

Component: InfraValidator



```
infra_validator = tfx.components.InfraValidator(  
    model=trainer.outputs['model'],  
    examples=example_gen.outputs['examples'],  
    serving_spec=infra_validator_pb2.ServingSpec(...),  
    validation_spec=infra_validator_pb2.ValidationSpec(...),  
    request_spec=infra_validator_pb2.RequestSpec(...)  
).with_id("ModelInfraValidator")
```

Configuration options

ServingSpec: Type of model server and infrastructure to test with e.g. TF Serving

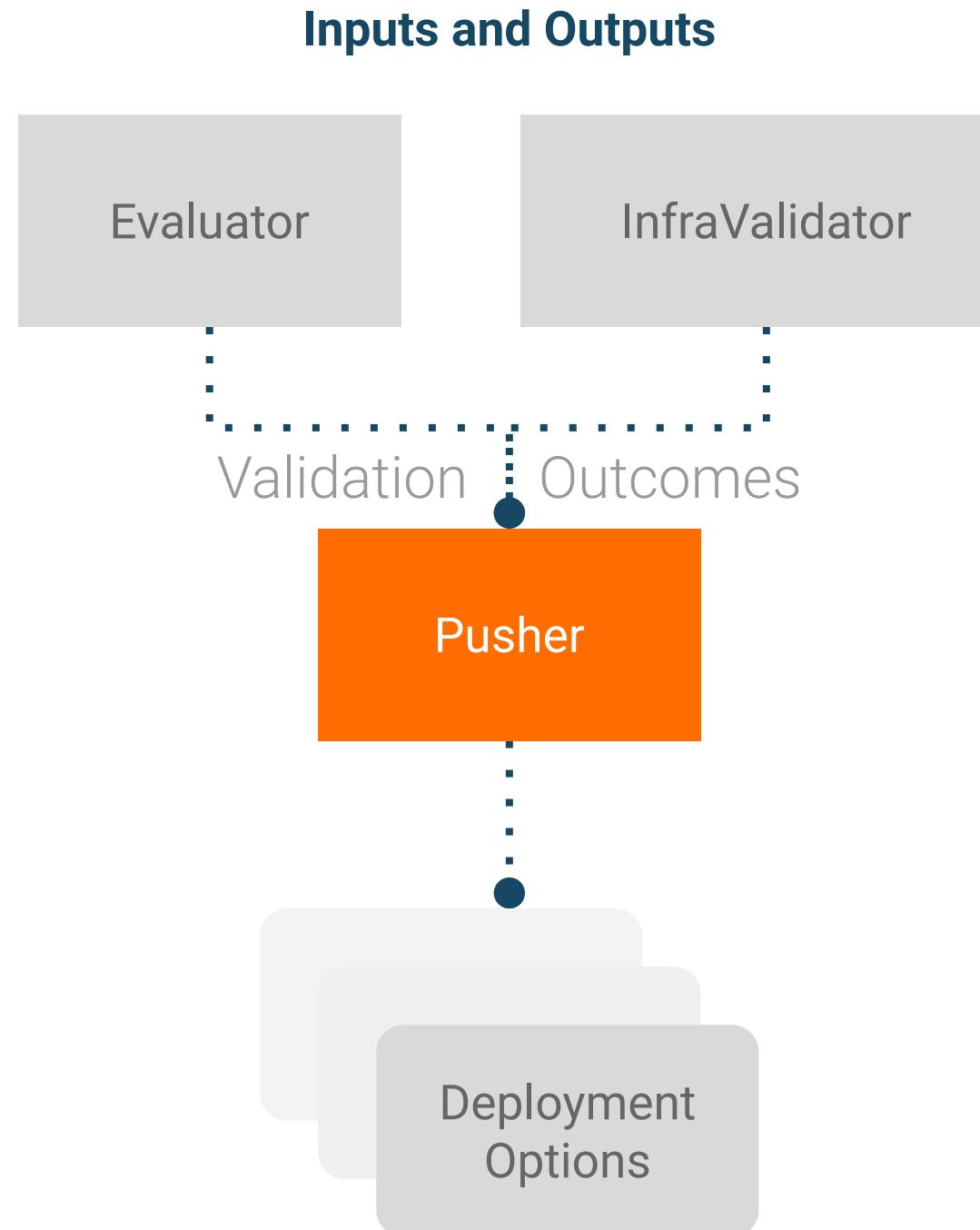
ValidationSpec: Adjusts the infra validation criteria or workflow.

RequestSpec: Which model signature, how many examples to test.

ML project lifecycle benefits

- Validate model in model serving environment before pushing to production
- Configurable to mirror different model serving environments such as Kubernetes and TF Serving

Component: Pusher

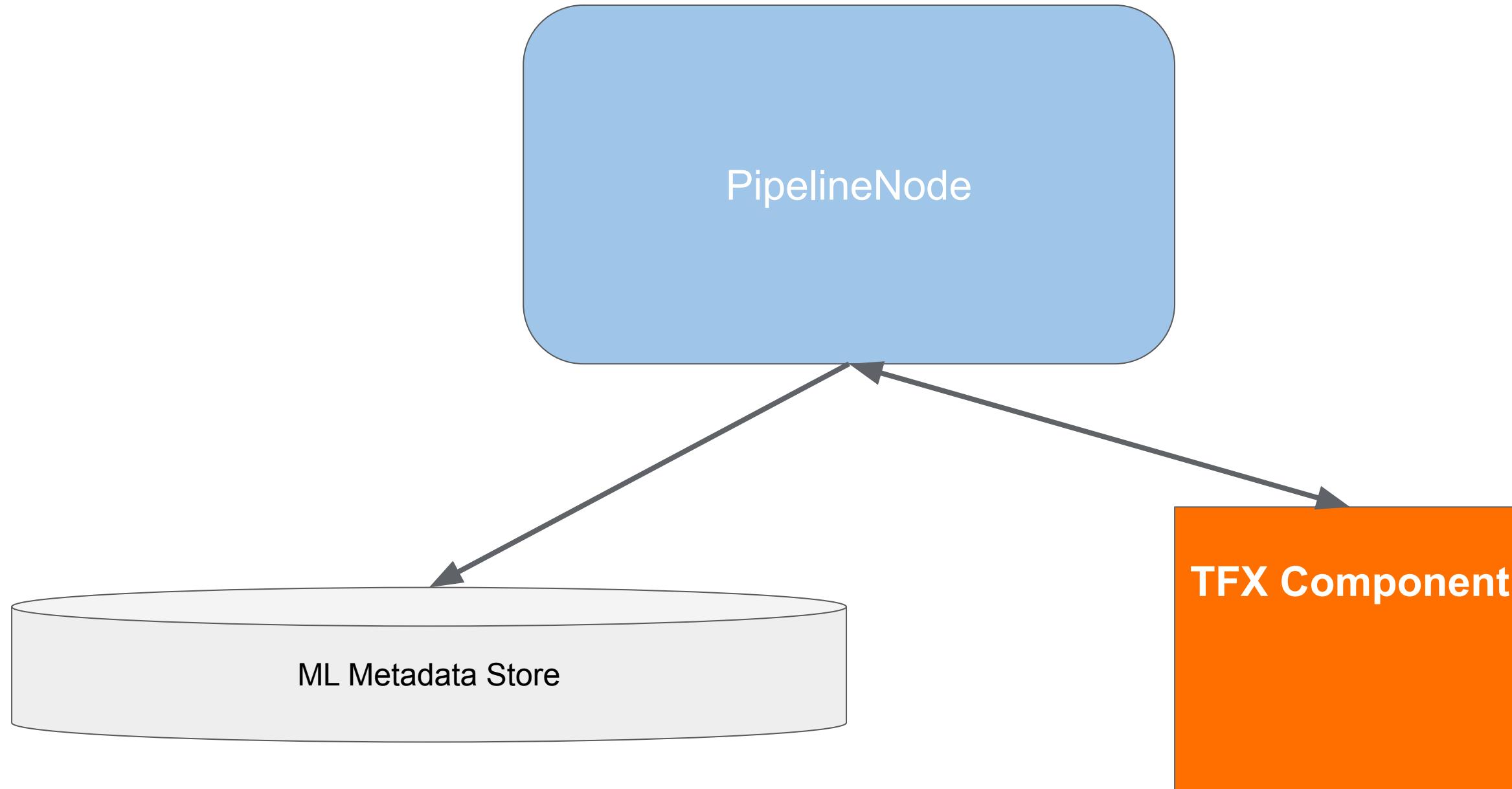


```
pusher = tfx.components.Pusher(  
    model=trainer.outputs['model'],  
    model_blessing=model_analyzer.outputs['blessing'],  
    infra_blessing=infra_validator.outputs['blessing'],  
    push_destination=pusher_pb2.PushDestination(  
        filesystem=pusher_pb2.PushDestination.Filesystem(  
            base_directory=SERVING_MODEL_DIR))  
).with_id("ModelPusher")
```

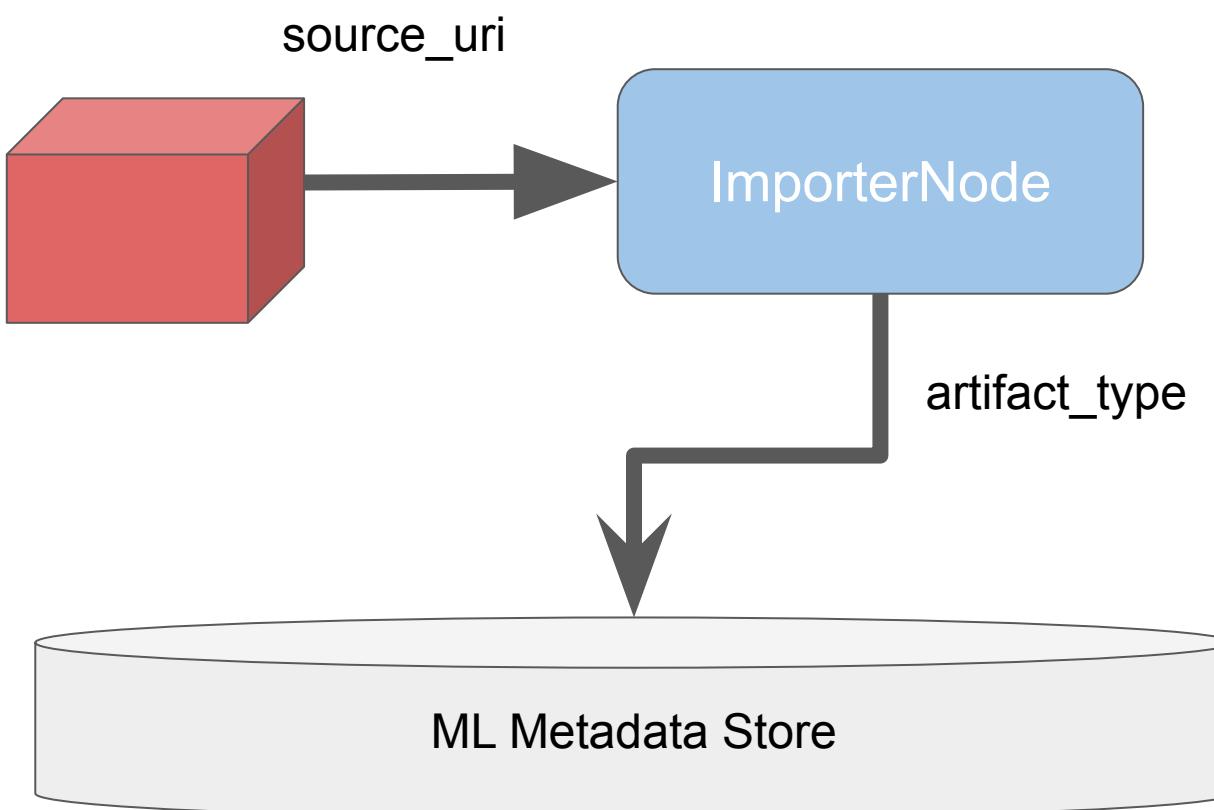
ML project lifecycle benefits

- Validation checks on model performance and serving ability before pushing to production.
- Reusable model export code that is configurable for different push destinations:
 - Filesystem (TensorFlow Lite, TensorFlow JS)
 - Model Server (TensorFlow Serving)

What are Pipeline Nodes?

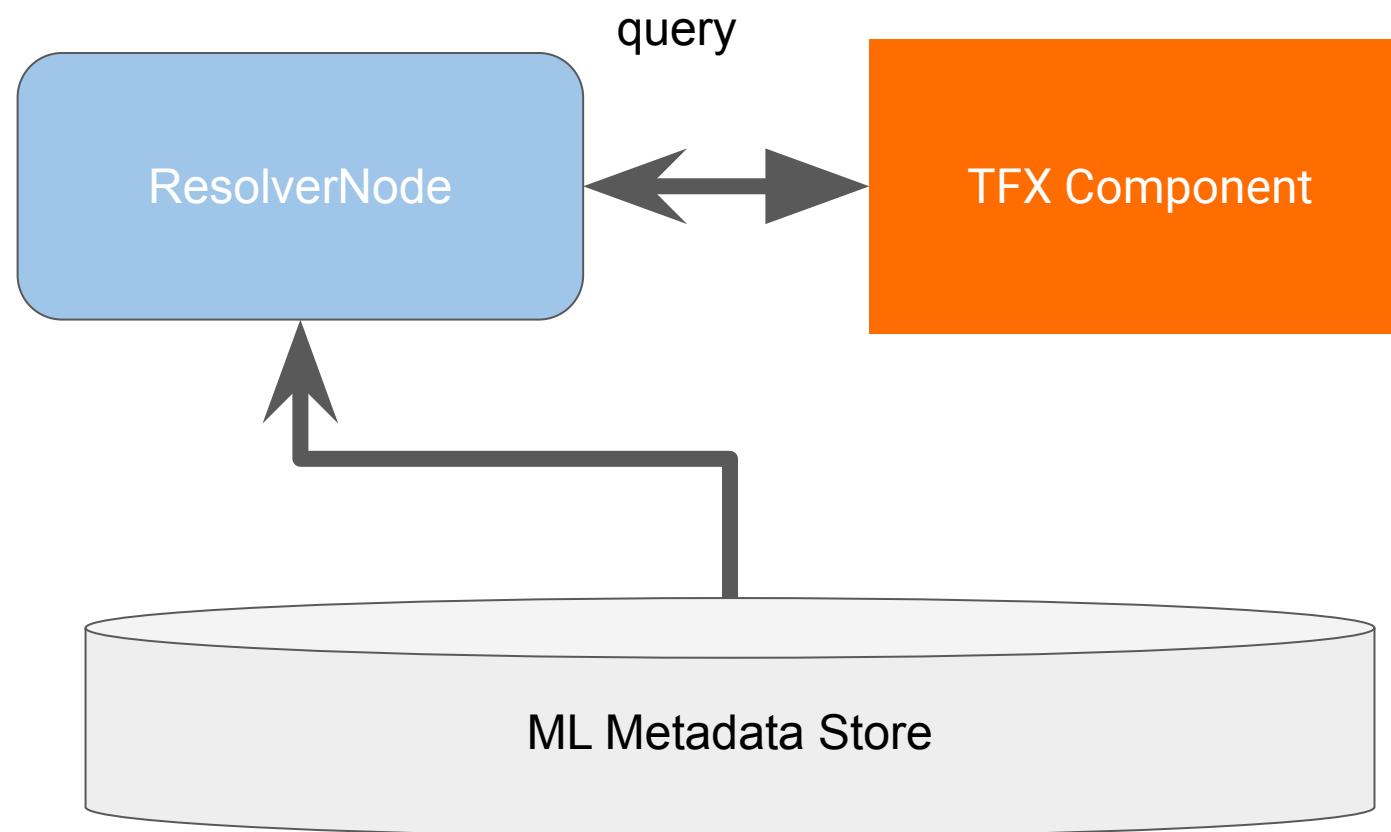


Importer imports an external data object into ML Metadata



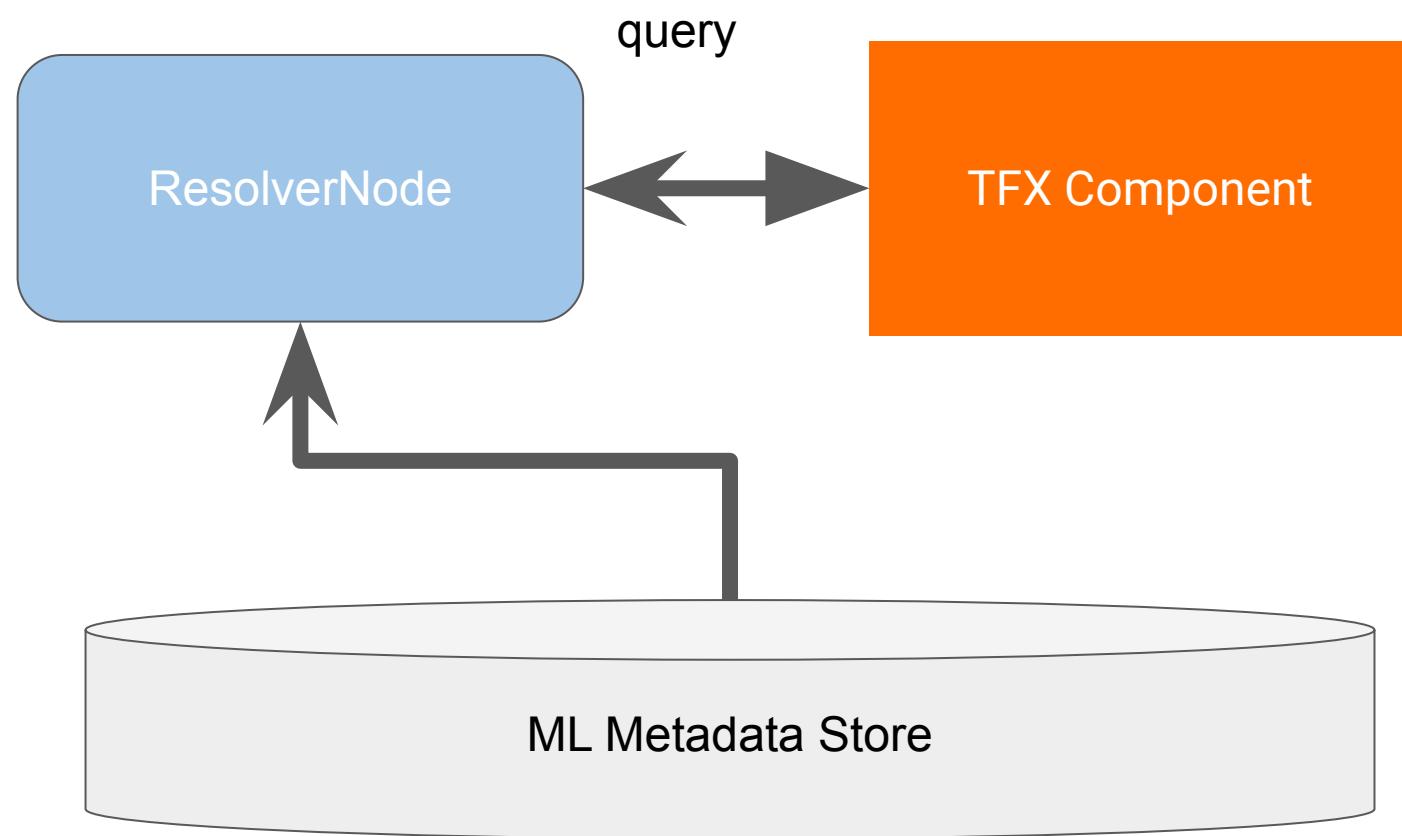
```
importer = tfx.dsl.components.common.importer.Importer(  
    source_uri='uri/to/schema'  
    artifact_type=standard_artifacts.Schema,  
).with_id("SchemaImporter")
```

Resolver performs metadata queries



```
latest_blessed_model = tfx.dsl.components.common.resolver.Resolver(  
    strategy_class=LatestBlessedModelStrategy,  
    model=Channel(type=tfx.types.standard_artifacts.Model),  
    model_blessing=Channel(type=tfx.types.standard_artifacts.ModelBlessing),  
).with_id("LatestBlessedModelResolver")
```

Current Resolver



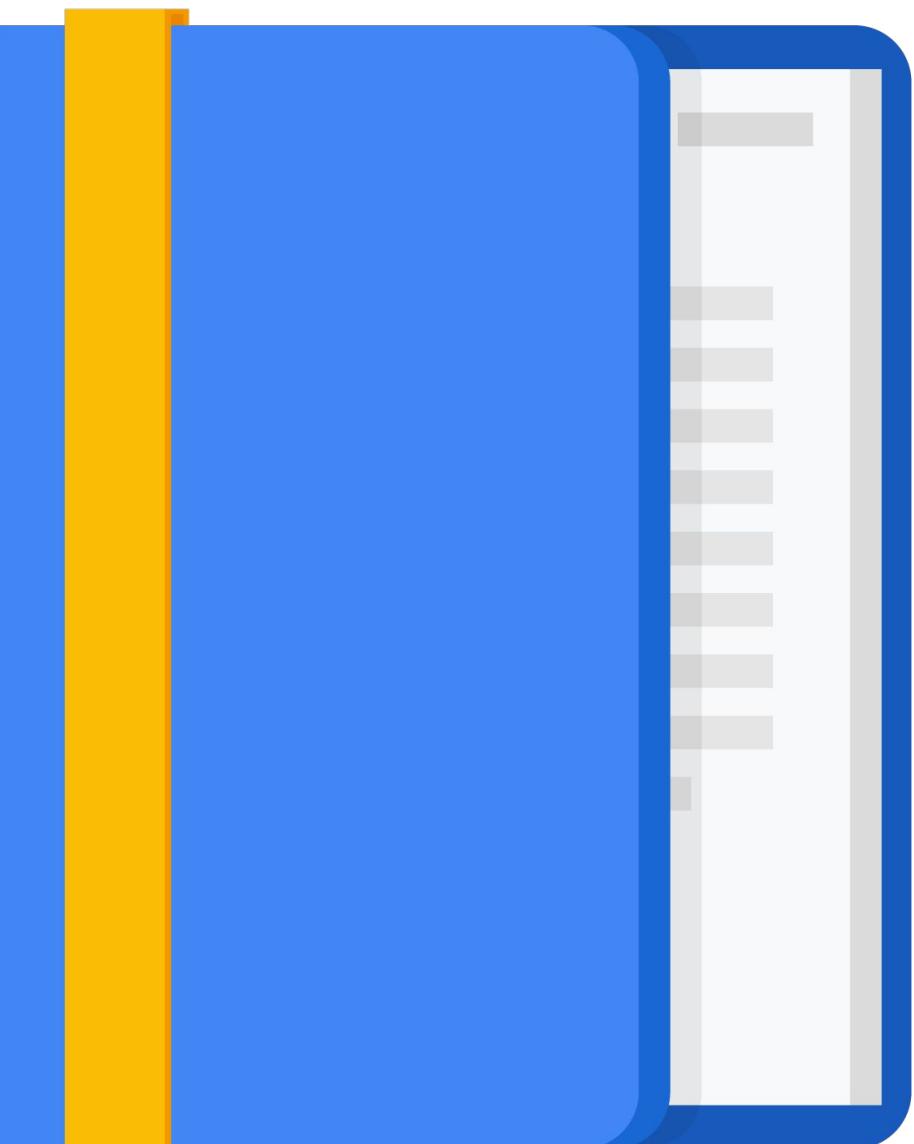
- **LatestBlessedModelStrategy**

Agenda

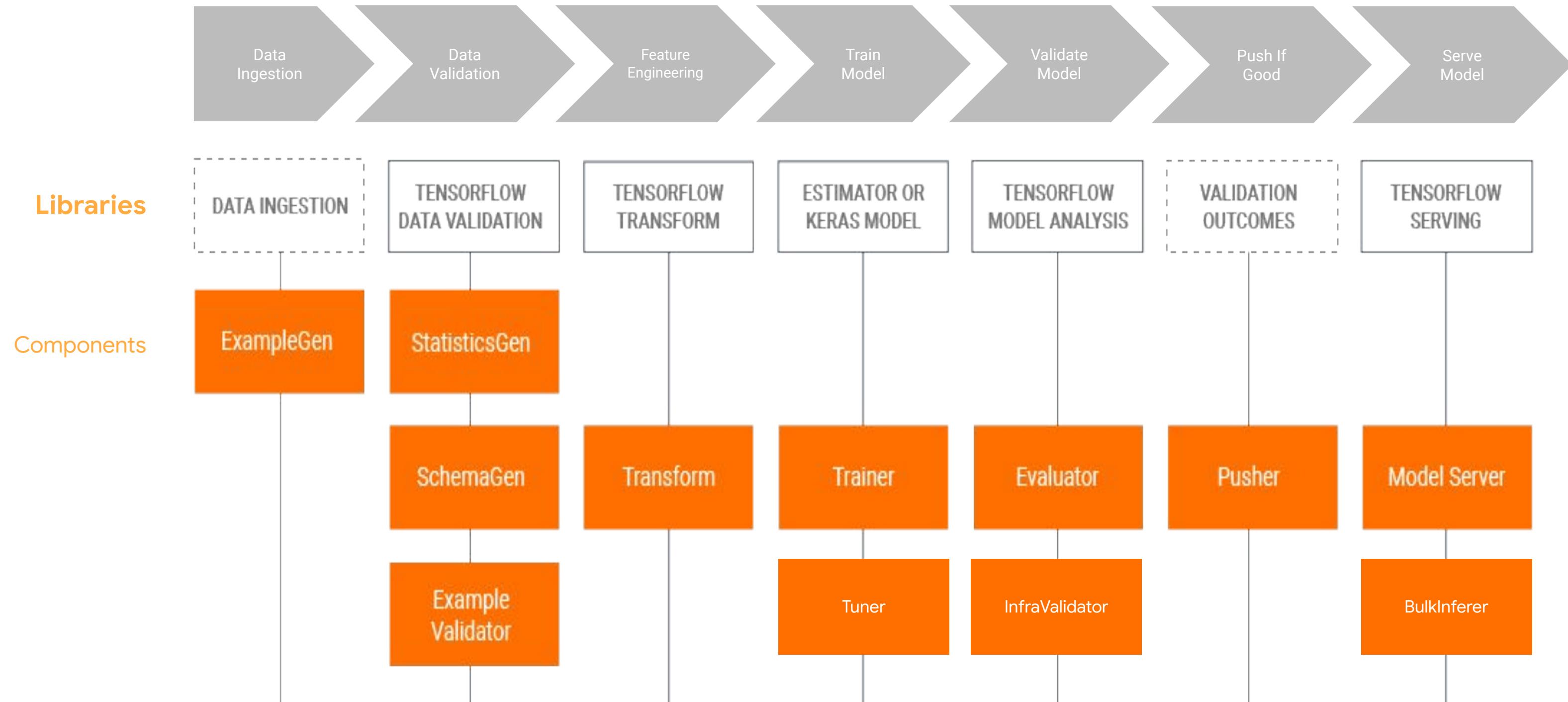
TensorFlow Extended (TFX)

TFX standard components

TFX libraries



TFX libraries integrate with ml pipeline each phase



Lab

TFX Walkthrough

In this lab, you will walk through the configuration and execution of the key TFX components.

[tfx_pipelines/walkthrough/labs/tfx_walkthrough_vertex.ipynb](#)