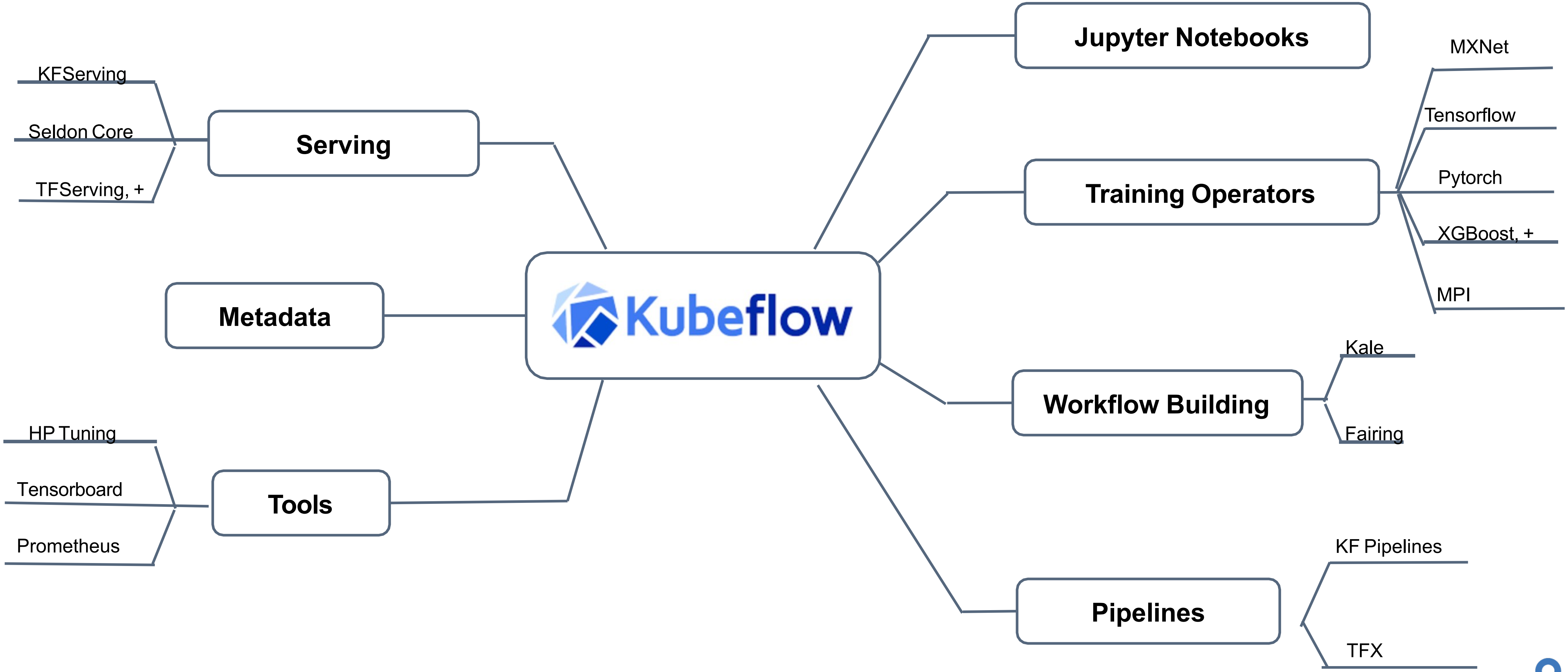
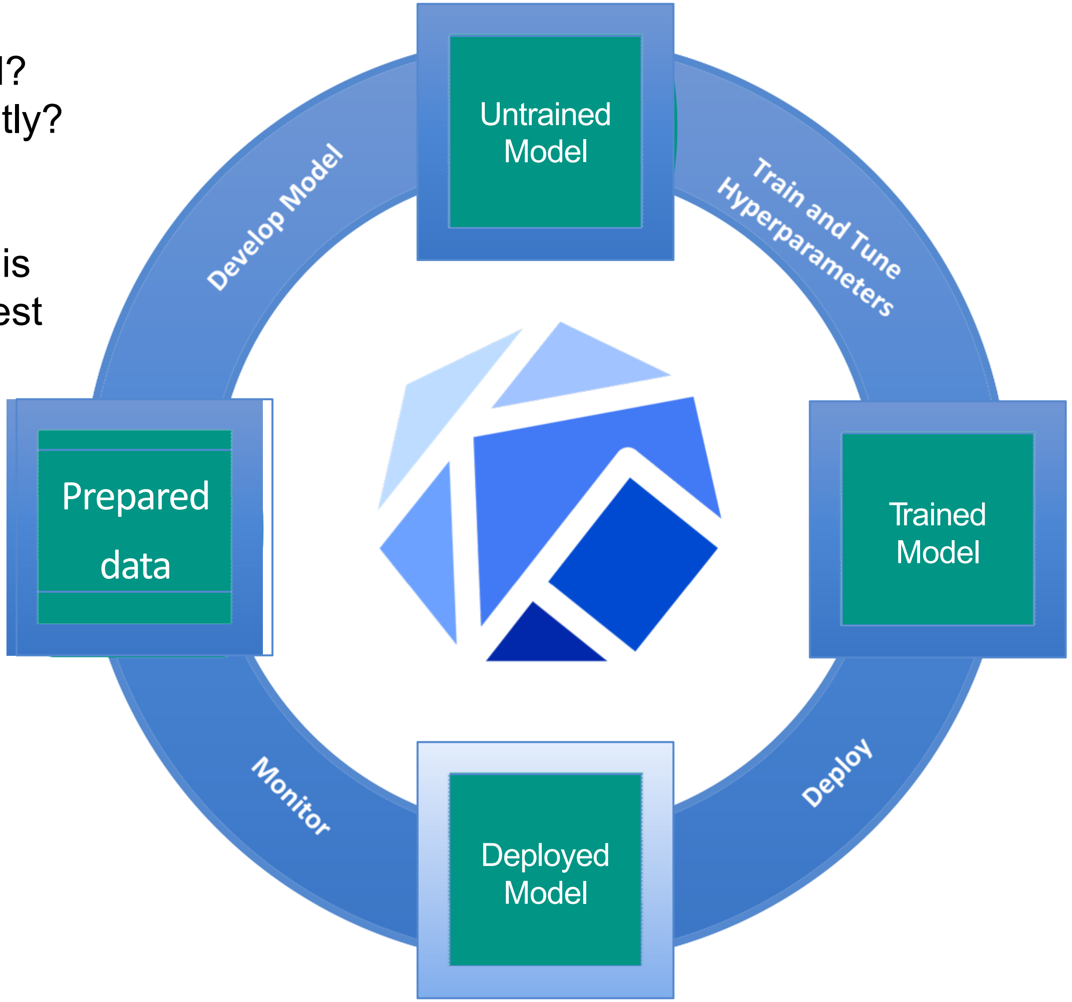
KFServing

Tommy Li, Pugang Xa



Production Model Serving? How hard could it be?

- ❑ Cost:
  Is the model over or under scaled?
  Are resources being used efficiently?
- ❑ Monitoring:
  Are the endpoints healthy? What is the performance profile and request trace?
- ❑ Rollouts:
  Is this rollout safe? How do I roll back? Can I test a change without swapping traffic?
- ❑ Protocol Standards:
  How do I make a prediction? GRPC? HTTP? Kafka?



- ❑ How do I handle batch predictions?
- ❑ How do I leverage standardized Data Plane protocol so that I can move my model across MLServing platforms?
- ❑ Frameworks:
  How do I serve on Tensorflow? XGBoost? Scikit Learn? Pytorch? Custom Code?
- ❑ Features:
  How do I explain the predictions? What about detecting outliers and skew? Bias detection? Adversarial Detection?
- ❑ How do I wire up custom pre and post processing

Experts fragmented across industry

- Seldon Core was pioneering Graph Inferencing.
- IBM and Bloomberg were exploring serverless ML lambdas. IBM gave a talk on the ML Serving with Knative at last KubeCon in Seattle
- Google had built a common Tensorflow HTTP API for models.
- Microsoft Kubernetizing their Azure ML Stack

Putting the pieces together

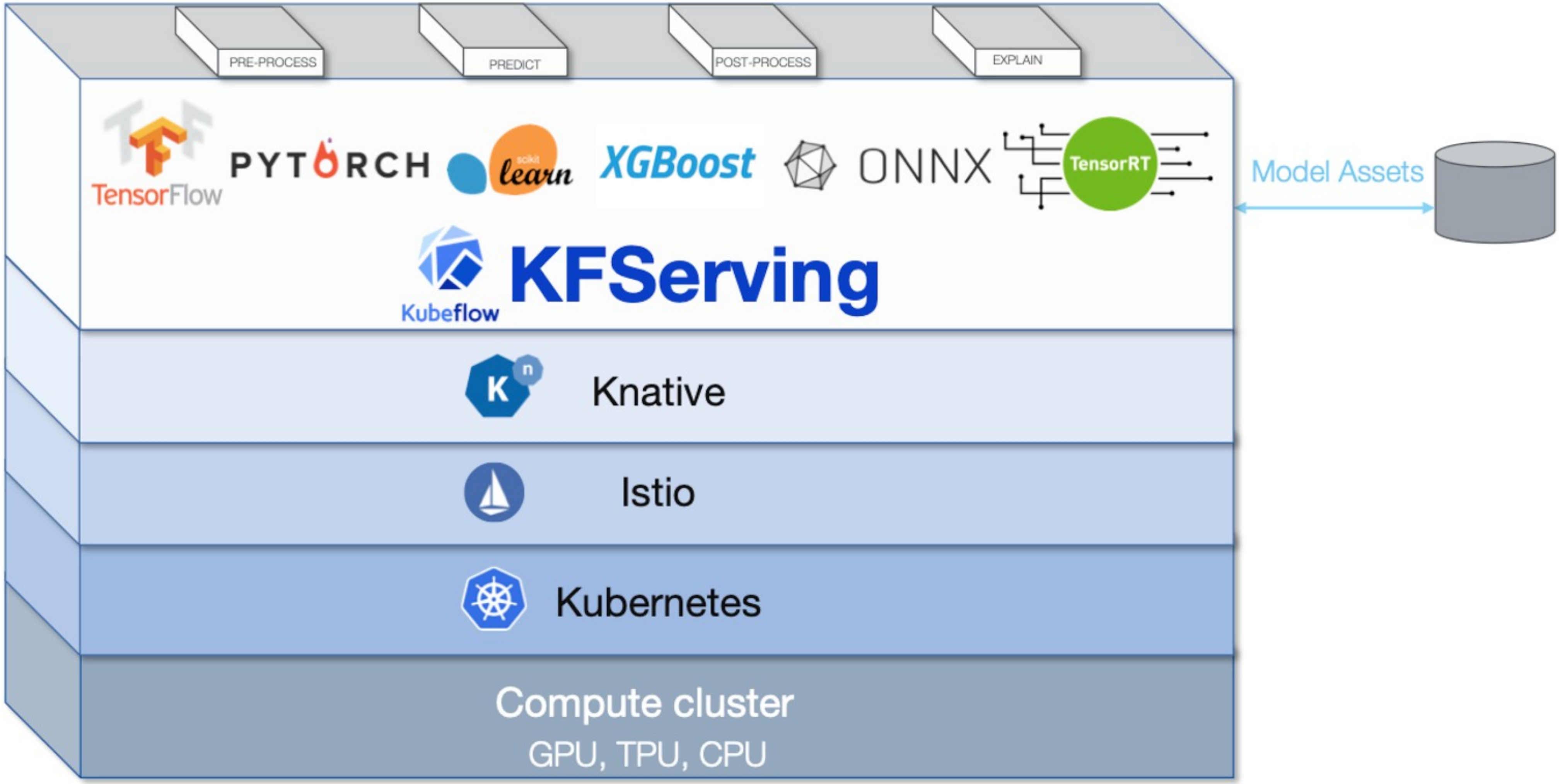
- Kubeflow created the conditions for collaboration.
- A promise of open code and open community.
- Shared responsibilities and expertise across multiple companies.
- Diverse requirements from different customer segments

Introducing KFServing

KFServing

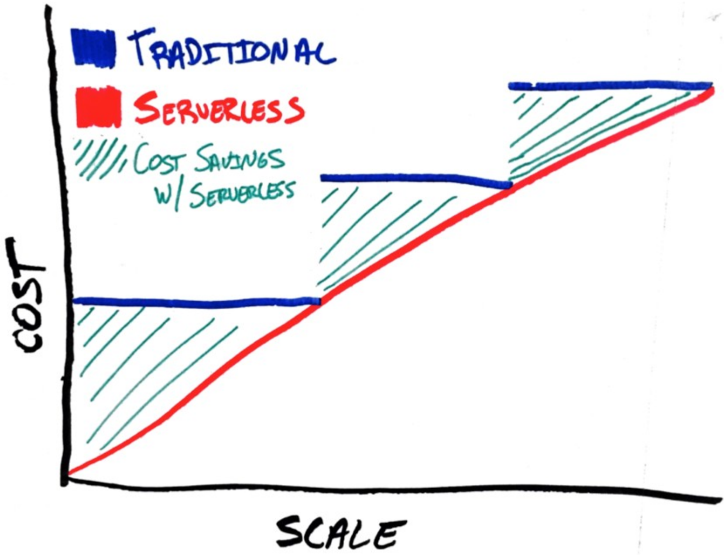
- Founded by Google, Seldon, IBM, Bloomberg and Microsoft
- Part of the Kubeflow project
- Focus on 80% use cases - single model rollout and update
- Kfserving 1.0 goals:
    - Serverless ML Inference
    - Canary rollouts
    - Model Explanations
    - Optional Pre/Post processing

KFServing Stack



KNative

IBM is
2nd largest contributor

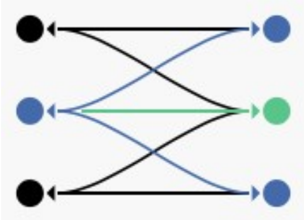


Knative provides a set of building blocks that enable declarative, container-based, serverless workloads on Kubernetes. Knative Serving provides primitives for serving platforms such as:

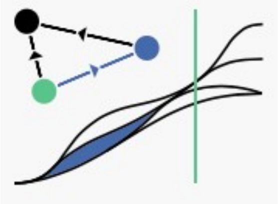
- Event triggered functions on Kubernetes
- Scale to and from zero
- Queue based autoscaling for GPUs and TPUs. KNative autoscaling by default provides inflight requests per pod
- Traditional CPU autoscaling if desired. Traditional scaling hard for disparate devices (GPU, CPU, TPU)

Istio

An open service mesh platform to **connect**, **observe**, **secure**, and **control** microservices.
Founded by Google, IBM and Lyft. IBM is the 2nd largest contributor

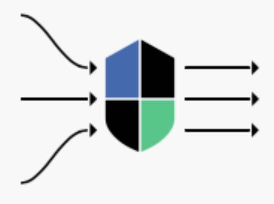


Connect: Traffic Control, Discovery, Load Balancing, Resiliency



Observe: Metrics, Logging, Tracing

Secure: Encryption (TLS), Authentication, and Authorization of service-to-service communication



Control: Policy Enforcement

KFServing: Default and Canary Configurations

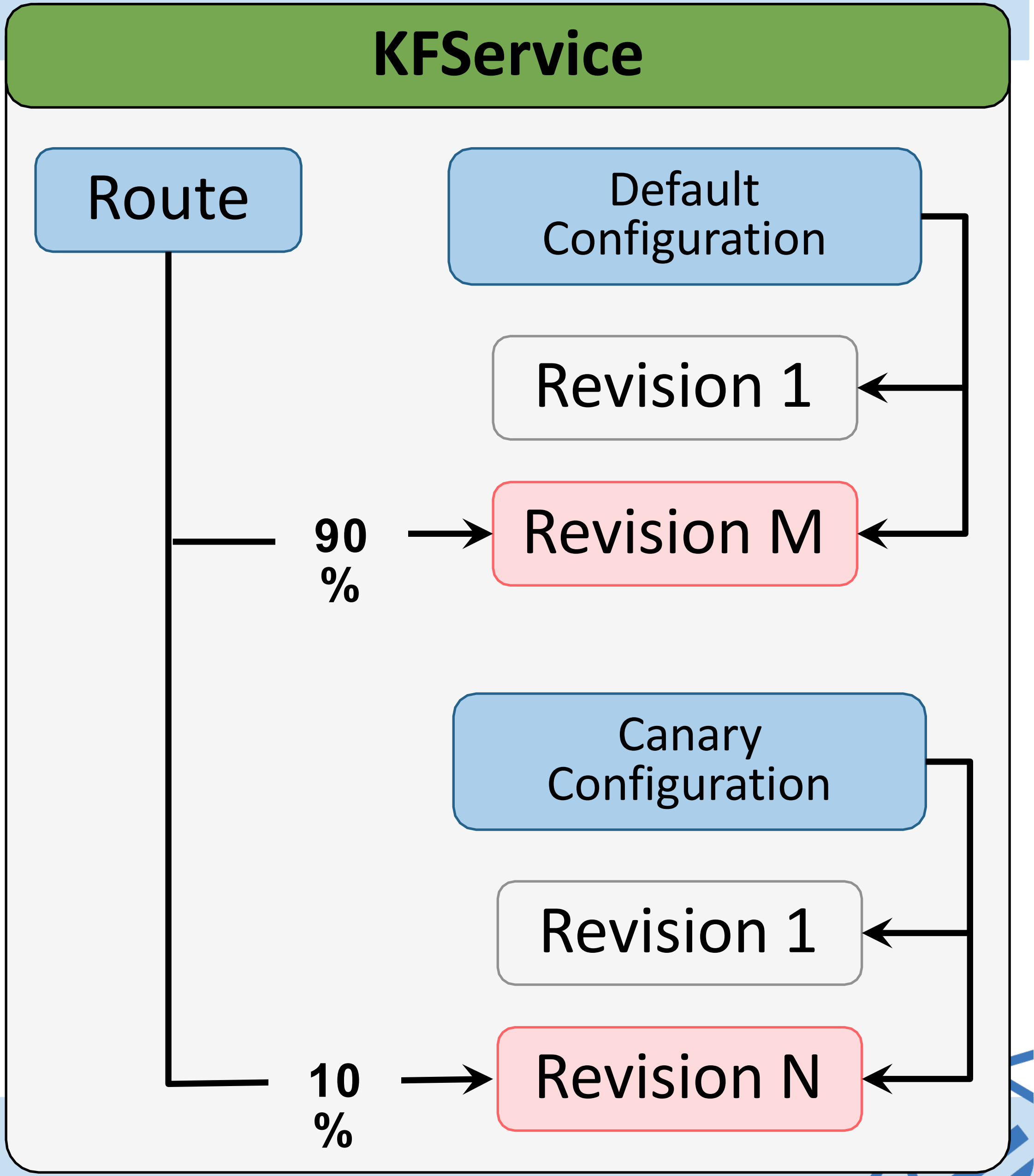
Manages the hosting aspects of your models

- **InferenceService** - manages the lifecycle of models

- **Configuration** - manages history of model deployments. Two configurations for default and canary.

- **Revision** - A snapshot of your model version

- **Route** - Endpoint and network traffic management



Supported Frameworks, Components and Storage Subsystems

Model Servers

- TensorFlow
- Nvidia TRTIS
- PyTorch
- XGBoost
- SKLearn
- ONNX

Components:

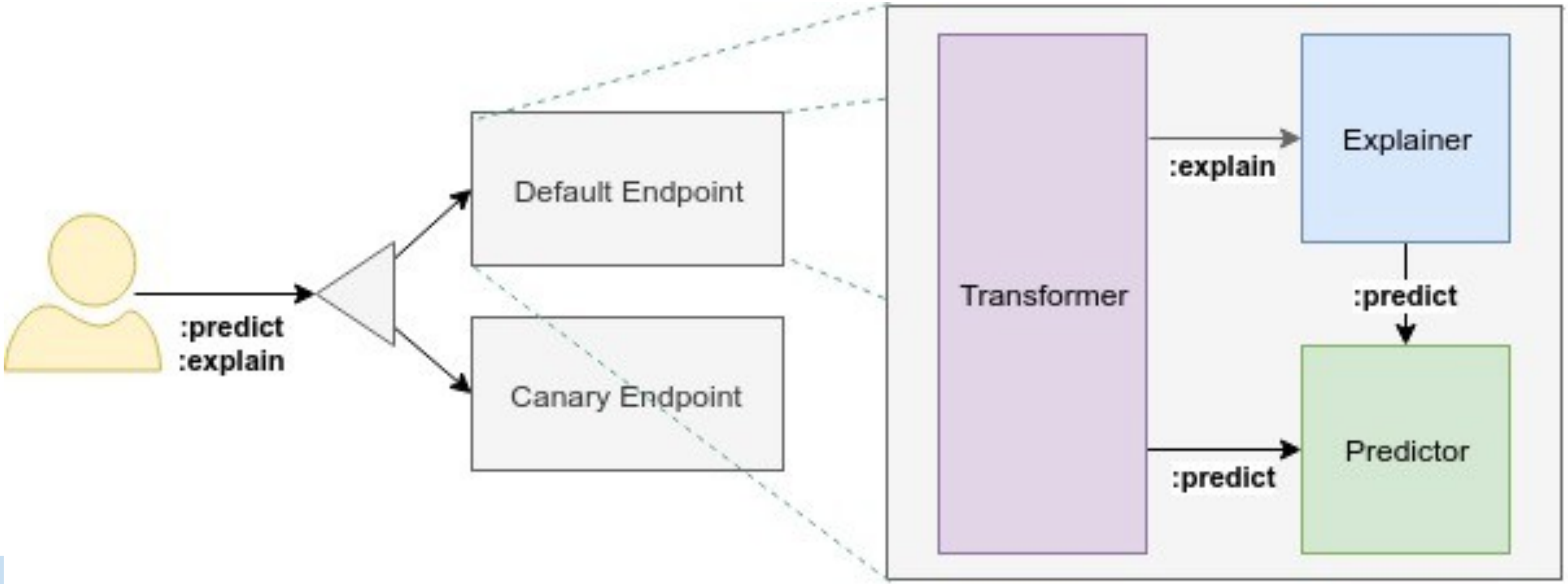
- Predictor, Explainer, Transformer (pre-processor, post-processor)

Storage

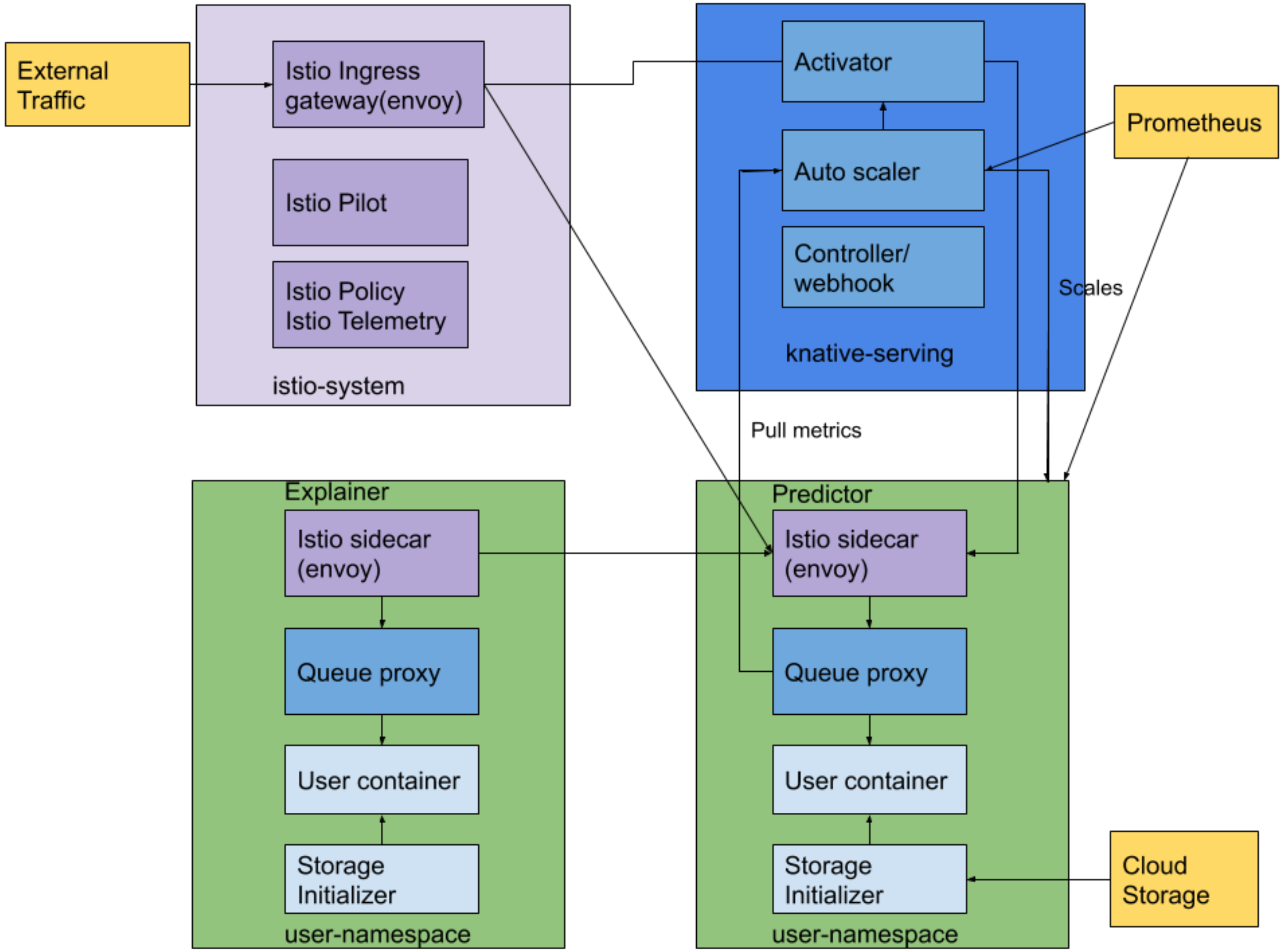
- AWS/S3
- GCS
- Azure Blob
- PVC

Inference Service Control Plane

The InferenceService architecture consists of a static graph of components which coordinate requests for a single model. Advanced features such as Ensembling, A/B testing, and Multi-Arm-Bandits should compose InferenceServices together.



KFServing Deployment View



KFServing Data Plane Unification

- Today's popular model servers, such as TFServing, ONNX, Seldon, TRTIS, all communicate using similar but non-interoperable HTTP/gRPC protocol
- KFServing v1 data plane protocol uses TFServing compatible HTTP API and introduces explain verb to standardize between model servers, punt on v2 for gRPC and performance optimization.

KFServing Data Plane v1 protocol

API	Verb	Path	Payload
List Models	GET	/v1/models	[model_names]
Readiness	GET	/v1/models/<model_name>	
Predict	POST	/v1/models/<model_name>:predict	Request: {instances:[]} Response: {predictions:[]}
Explain	POST	/v1/models<model_name>:explain	Request: {instances:[]} Response: {predictions:[], explanations:[]}

KFServing Examples

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

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

```
apiVersion: "serving.kubeflow.org/v1alpha1"
kind: "InferenceService"
metadata:
  name: ”pytorch-iris"
spec:
  default:
            pytorch:
            modelUri: "gs://kfserving-samples/models/pytorch/iris"
```

Canary/Pinned Examples

```
apiVersion: "serving.kubeflow.org/v1alpha1"
kind: "KFService"
metadata:
  name: "my-model"
spec:
  default:
      # 90% of traffic is sent to this model
      tensorflow:
      modelUri: "gs://mybucket/mymodel-2"
  canaryTrafficPercent: 10
  canary:
      # 10% of traffic is sent to this model
      tensorflow:
      modelUri: "gs://mybucket/mymodel-3"
```

Canary

```
apiVersion: "serving.kubeflow.org/v1alpha1"
kind: "KFService"
metadata:
  name: "my-model"
spec:
  default:
      tensorflow:
      modelUri: "gs://mybucket/mymodel-2"
  # Defaults to zero, so can also be omitted or explicitly set to zero.
  canaryTrafficPercent: 0
  canary:
      # Canary is created but no traffic is directly forwarded.
      tensorflow:
      modelUri: "gs://mybucket/mymodel-3"
```

Pinned

TensorFlow Serving(TFServing)

- Flexible, high performance serving system for TensorFlow
- https://www.tensorflow.org/tfx/guide/serving
- Stable and Google has been using it since 2016
- Saved model format and graphdef
- Written in C++, support both REST and gRPC
- KFServing allows you to easily spin off an Inference Service with TFServing to serve your tensorflow model on CPU or GPU with serverless features like canary rollout, autoscaling.

Inference Service with Transformer and TFServing

Kubeflow

```yaml
apiVersion: serving.kubeflow.org/v1alpha2
kind: InferenceService
metadata:
  name: bert-serving
spec:
  default
    transformer:
      custom:
        container:
          image: bert-transformer:v1
    predictor:
      tensorflow:
        storageUri: s3://examples/bert
        runtimeVersion: 1.14.0-gpu
        resources:
          limits:
            nvidia.com/gpu: 1
```

Pre/Post Processing

Tensorflow Model Server

```python
class BertTransformer(kfserving.KFModel):
    def_init_(self, name):
        super()._init__(name)
        self.bert_tokenizer =
    BertTokenizer(vocab_file)  def preprocess(self,
    inputs: Dict) -> Dict:
        encoded_features = bert_tokenizer.encode_plus(
        text=text_a, text_pair=text_b)
        return {“input_ids”:
          encoded_features[“input_ids”], “input_mask”:
          encoded_features[“attention_mask”],
          “segment_ids”:
          encoded_features[“segment_ids”], “label_ids”:
          1}
    def postprocess(self, inputs: Dict) -> Dict:
        return inputs
```

NVIDIA Triton Inference Server

- NVIDIA's highly-optimized model runtime on GPUs
- https://docs.nvidia.com/deeplearning/sdk/tensorrt-inference-server-guide/docs
- Supports model repository, versioning
- Dynamic batching
- Concurrent model execution
- Supports TensorFlow, PyTorch, ONNX models
- Written in C++, support both REST and gRPC
- TensorRT Optimizer can further bring down the BERT inference latency

Inference Service with Triton Inference Service

```yaml
apiVersion: serving.kubeflow.org/v1alpha2
kind: InferenceService
metadata:
  name: bert-serving
spec:
  default
    transformer:
      custom:
        container:
          image: bert-transformer:v1
          env:
            name: STORAGE_URI
            value: s3://examples/bert_transformer
    predictor:
      tensorrt:
        storageUri: s3://examples/bert
        runtimeVersion: r20.02
        resources:
          limits:
            nvidia.com/gpu: 1
```

Pre/Post Processing

Triton Inference Server

```python
infer_ctx = InferContext(url, protocol, model_name,
model_version)
unique_ids = np.int32([1])
segment_ids = features["segment_ids"]
input_ids = features["input_ids"]
input_mask = features["input_mask"]
result = infer_ctx.run({ 'unique_ids' : (unique_ids,),
                         'segment_ids' : (segment_ids,),
                         'input_ids' : (input_ids,),
                         'input_mask' : (input_mask,) },
                       { 'end_logits' :
InferContext.ResultFormat.RAW,
                         'start_logits' :
InferContext.ResultFormat.RAW }, batch_size)
```

PyTorch Model Server

- PyTorch model server maintained by KFServing
- https://github.com/kubeflow/kfserving/tree/master/python/pytorchserver
- Implemented in Python with Tornado server
- Loads model state dict and model class python file
- GPU Inference is supported in KFServing 0.3 release

- Alternatively you can export PyTorch model in ONNX format and serve on TensorRT Inference Server or ONNX Runtime Server.

ONNX Runtime Server

- ONNX Runtime is a performance-focused inference engine for ONNX models
- https://github.com/microsoft/onnxruntime
- Supports Tensorflow, Pytorch models which can be converted to ONNX
- Written in C++, support both REST and gRPC
- ONNX Runtime optimized BERT transformer network to further bring down the latency
  https://github.com/onnx/tutorials/blob/master/tutorials/Inference-TensorFlow-Bert-Model-for-High-Performance-in-ONNX-Runtime.ipynb

Inference Service with PyTorch/ONNX Runtime

Kubeflow

```yaml
apiVersion: serving.kubeflow.org/v1alpha2
kind: InferenceService
metadata:
  name: bert-serving
spec:
  default
    transformer:
      custom:
        container:
          image: bert-transformer:v1
          env:
            name: STORAGE_URI
            value: s3://examples/bert_transformer
    predictor:
      pytorch:
        storageUri: s3://examples/bert
        runtimeVersion: v0.3.0-gpu
        resources:
          limits:
            nvidia.com/gpu: 1
```

Pre/Post Processing

Pytorch Model Server

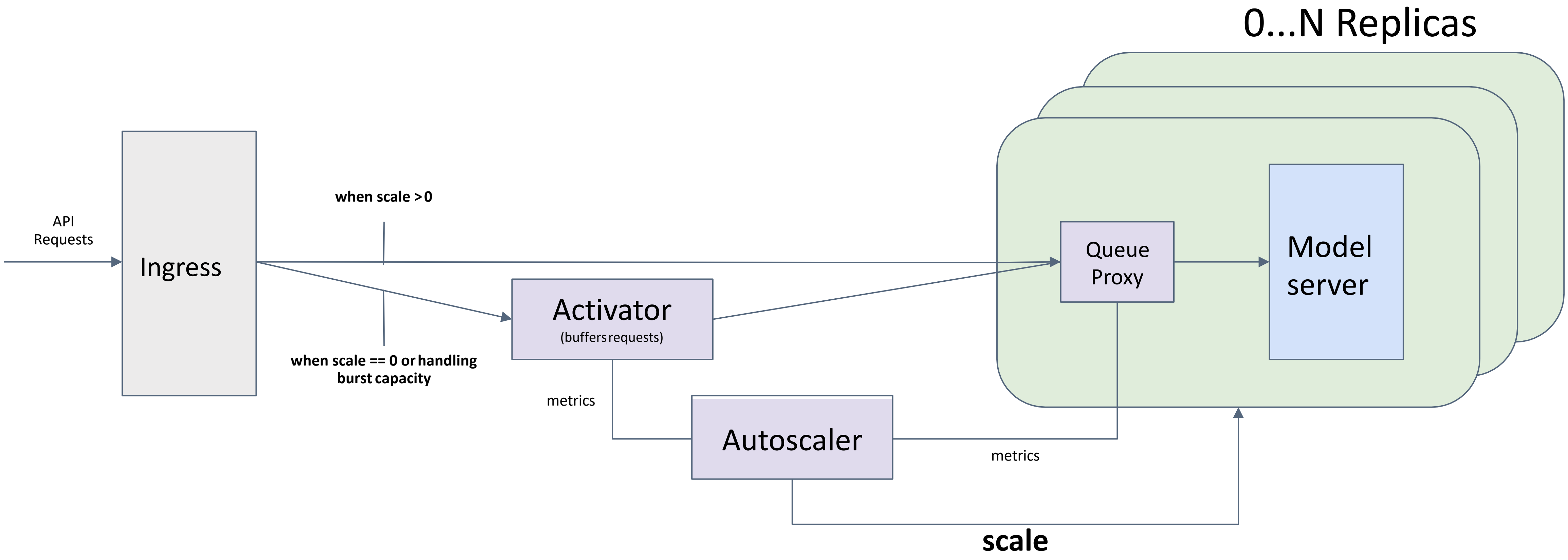
```yaml
apiVersion: serving.kubeflow.org/v1alpha2
kind: InferenceService
metadata:
  name: bert-serving-onnx
spec:
  default
    transformer:
      custom:
        container:
          image: bert-transformer:v1
          env:
            name: STORAGE_URI
            value: s3://examples/bert_transformer
    predictor:
      onnx:
        storageUri: s3://examples/bert
        runtimeVersion: 0.5.1
```

Pre/Post Processing

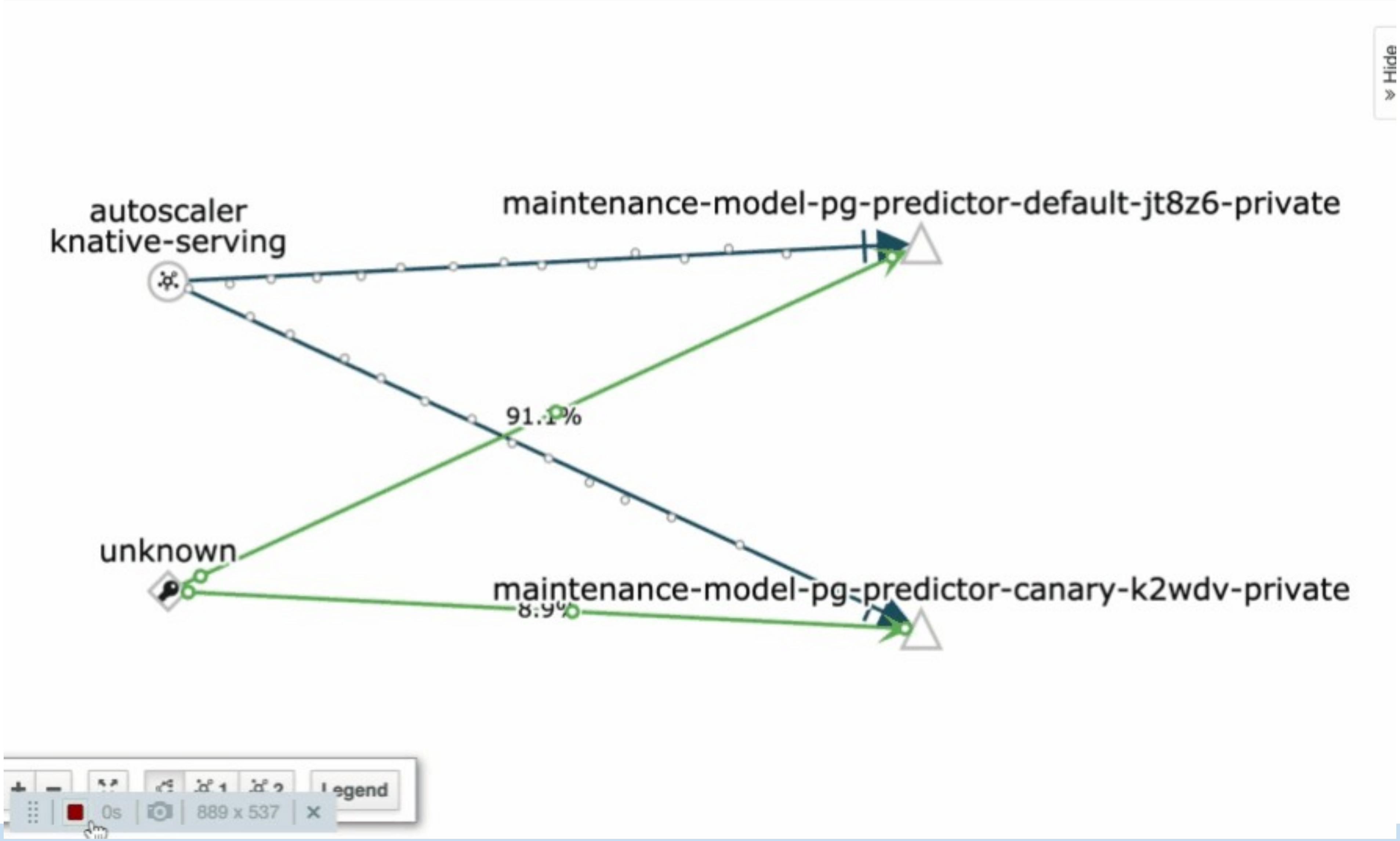
ONNX Runtime Server

GPU Autoscaling - KNative solution

- Scale based on # in-flight requests against expected concurrency
- Simple solution for heterogeneous ML inference autoscaling



KFServing: Default, Canary and`Autoscaler

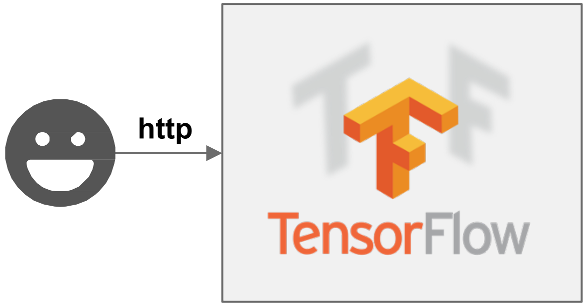


But the Data Scientist Sees...

Kubeflow

```yaml
apiVersion: "serving.kubeflow.org/v1alpha2"
kind: "InferenceService"
metadata:
  name: "flowers-sample"
spec:
  default:
    predictor:
      tensorflow:
        storageUri:  "gs://kfserving-samples/models/tensorflow/flowers"
```

=



- A pointer to a Serialized Model File
- 9 lines of YAML
- A live model at an HTTP endpoint

- Scale to Zero
- GPU Autoscaling
- Safe Rollouts
- Optimized Serving Containers
- Network Policy and Auth
- HTTP APIs (gRPC soon)
- Tracing
- Metrics

Production users include:

Bloomberg

CoreWeave

gojek

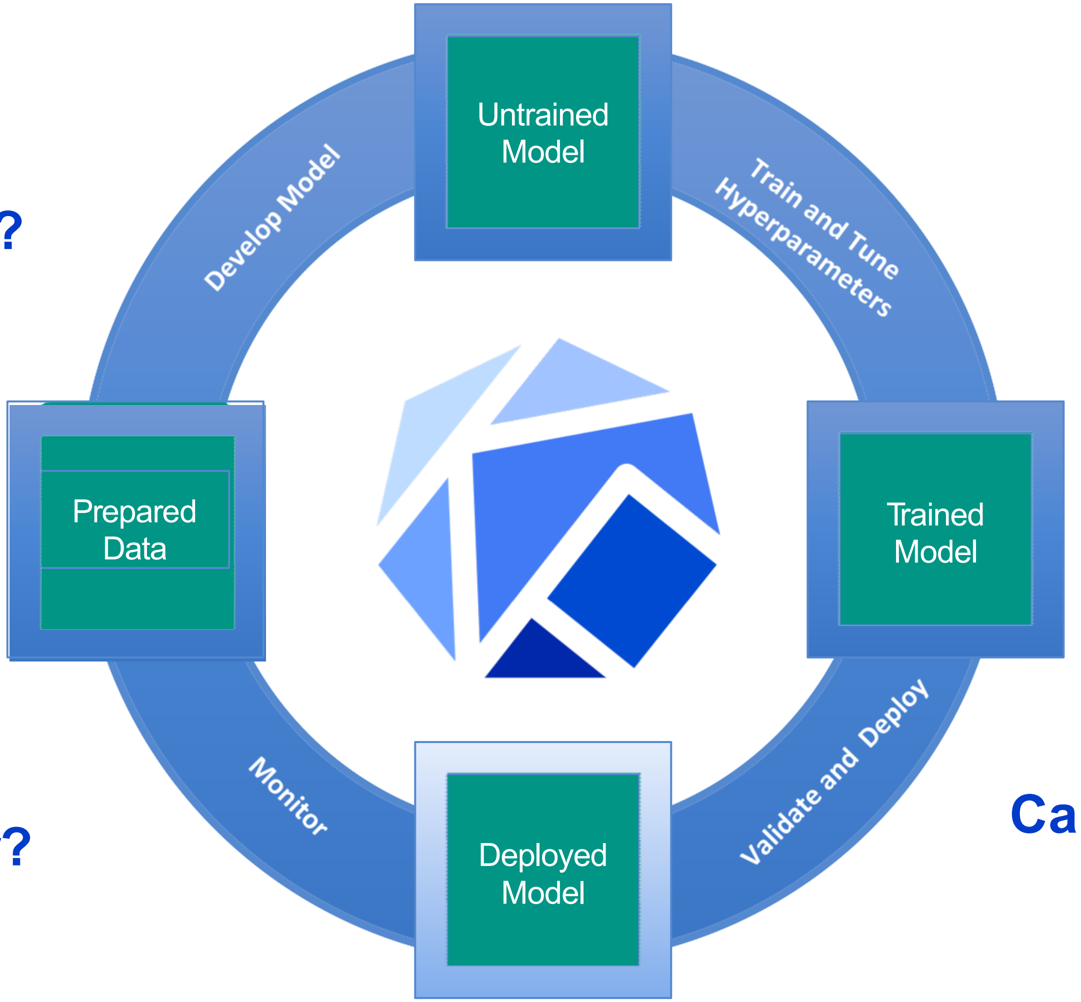
Summary

- With KFServing user can easily deploy the service for inference on GPU with performant industry leading model servers as well as benefiting from all the serverless features.
- Autoscale the inference workload based on your QPS, much better resource utilization.
- gRPC can provide better performance over REST which allows multiplexing and protobuf is a efficient and packed format than JSON.
- Transformer can work seamlessly with different model servers thanks to KFServing’s data plane standardization.
- GPUs benefit a lot from batching the requests.

Model Serving is accomplished. Can the predictions be trusted?

Are there concept drifts?

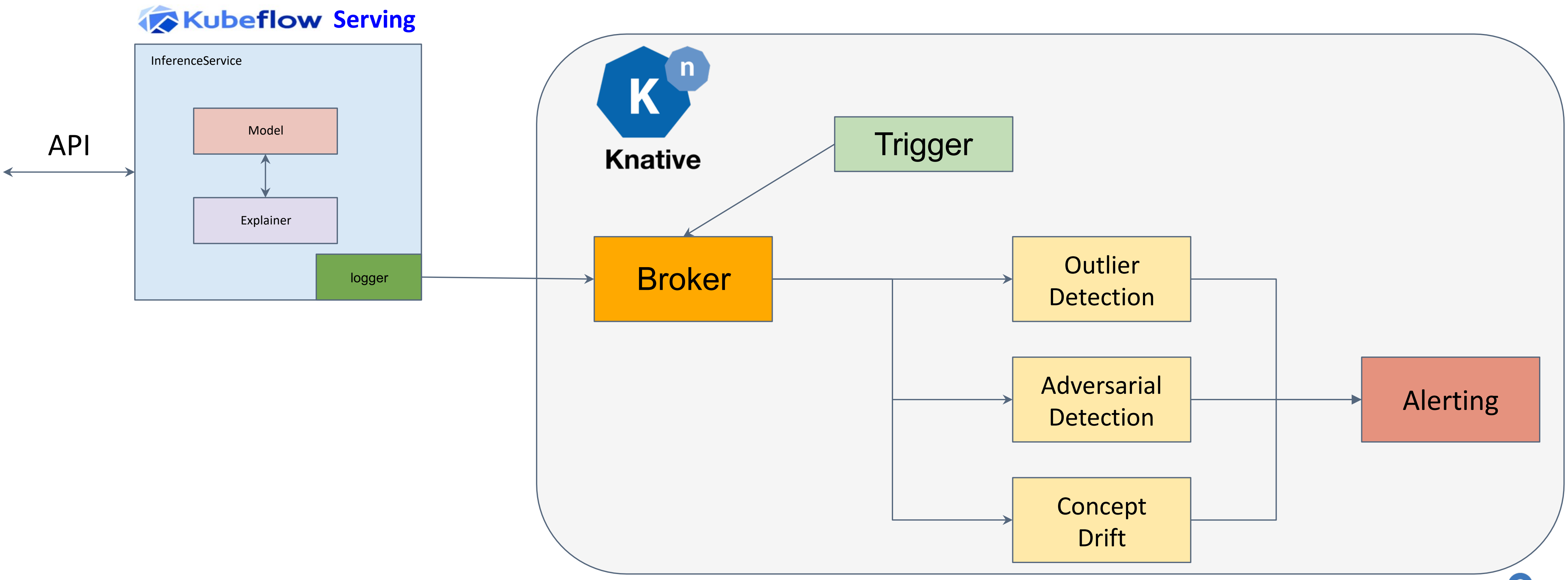
Is the model vulnerable to adversarial attacks?



Is there an outlier?

Can the model explain its predictions?

Production ML Architecture



Payload Logging

Payload Logging

Why:

- Capture payloads for analysis and future retraining of the model
- Perform offline processing of the requests and responses

KfServing Implementation (alpha):

- Add to any InferenceService Endpoint: Predictor, Explainer, Transformer
- Log Requests, Responses or Both from the Endpoint
- Simple specify a URL to send the payloads
- URL will receive CloudEvents

cloudevents

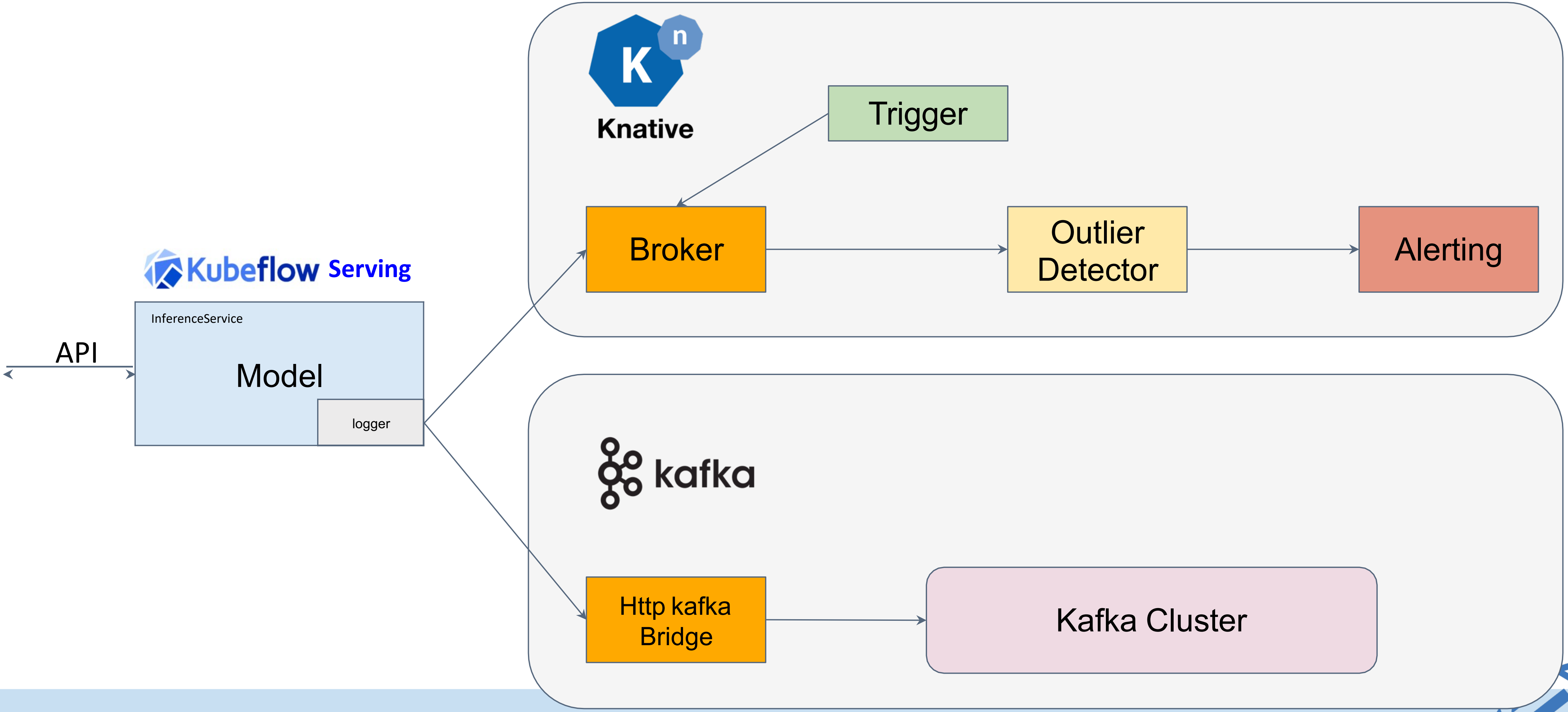
```
POST /event HTTP/1.0
Host: example.com
Content-Type: application/json
ce-specversion: 1.0
ce-type: repo.newItem
ce-source: http://bigco.com/repo
ce-id: 610b6dd4-c85d-417b-b58f-3771e532

<payload>
```

Payload Logging

```yaml
apiVersion: "serving.kubeflow.org/v1alpha2"
kind: "InferenceService"
metadata:
 name: "sklearn-iris"
spec:
 default:
   predictor:
   minReplicas: 1
     logger:
       url: http://message-dumper.default/
       mode: all
     sklearn:
       storageUri: "gs://kfserving-samples/models/sklearn/iris"
       resources:
         requests:
           cpu: 0.1
```

Payload Logging Architecture Examples



Machine Learning Explanations

KfServing Explanations

```yaml
apiVersion: "serving.kubeflow.org/v1alpha2"
kind: "InferenceService"
metadata:
 name: "income"
spec:
 default:
   predictor:
     sklearn:
       storageUri: "gs://seldon-models/sklearn/income/model"
   explainer:
     alibi:
       type: AnchorTabular
       storageUri: "gs://seldon-models/sklearn/income/explainer"
```

```yaml
apiVersion: "serving.kubeflow.org/v1alpha2"
kind: "InferenceService"
metadata:
 name: "moviesentiment"
spec:
 default:
   predictor:
     sklearn:
       storageUri: "gs://seldon-models/sklearn/moviesentiment"
   explainer:
     alibi:
       type: AnchorText
```

Seldon Alibi:Explain

Kubeflow

https://github.com/SeldonIO/alibi

Giovanni Vacanti

Janis Klaise

Arnaud Van Looveren

Alexandru Coca

State of the art implementations:

- Anchors
- Counterfactuals
- Contrastive explanations
- Trust scores

ALIBI EXPLAIN

Explanations: Resources

Kubeflow

AI Explainability 360
↳ (AIX360)

https://github.com/IBM/AIX360

AIX360 toolkit is an open-source library to help explain AI and machine learning models and their predictions. This includes three classes of algorithms: local post-hoc, global post-hoc, and directly interpretable explainers for models that use image, text, and structured/tabular data.

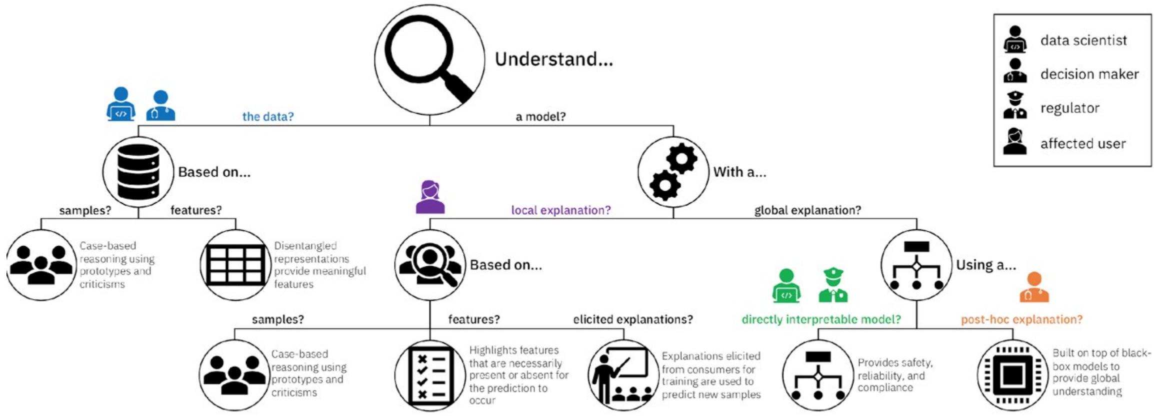
AIX360

The AI Explainability360 Python package includes a comprehensive set of explainers, both at global and local level.

Toolbox

Local post-hoc
Global post-hoc
Directly interpretable

http://aix360.mybluemix.net



AIX360 Explainability in KFServing

```yaml
apiVersion: serving.kubeflow.org/v1alpha2
kind: InferenceService
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: aixserver
spec:
  default:
    predictor:
      minReplicas: 1
      custom:
        container:
          name: predictor
          image: aipipeline/rf-predictor:0.2.2
          command: ["python", "-m", "rfserver", "--model_name", "aixserver"]
          imagePullPolicy: Always
          resources:
            requests:
              memory: "2Gi"
              cpu: "1"
            limits:
              memory: "2Gi"
              cpu: "1"
    explainer:
      minReplicas: 1
      custom:
        container:
          name: explainer
          image: aipipeline/aix-explainer:0.2.2
          command: ["python", "-m", "aixserver", "--predictor_host", "aixserver-predictor-default.default.svc.cluster.local", "explainer_type", "LimeImages"]
          imagePullPolicy: Always
          resources:
            requests:
              memory: "4Gi"
              cpu: "2"
            limits:
              memory: "4Gi"
              cpu: "2"
```

ML Inference Analysis

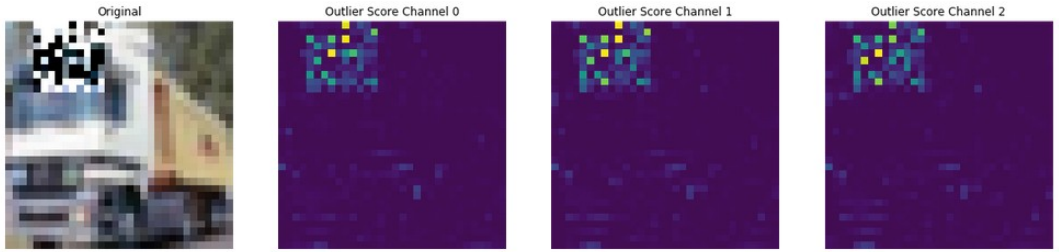
ML Inference Analysis

Don’t trust predictions on instances outside of training distribution!

- Outlier Detection
- Adversarial Detection
- Concept Drift

Outlier Detection

Don’t trust predictions on instances outside of training distribution!



→ ***Outlier Detection***

Detector types:

- stateful online vs. pretrained offline
- feature vs. instance level detectors

Data types:

- tabular, images & time series

Outlier types:

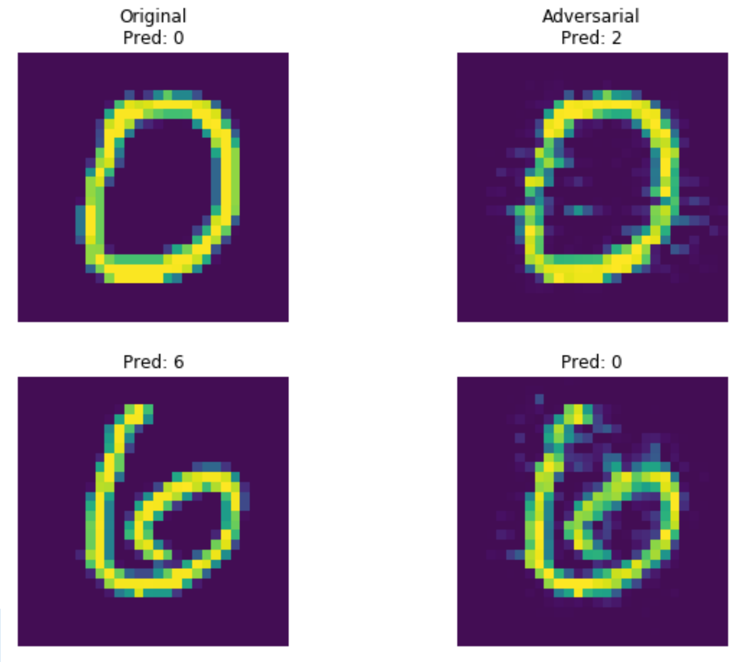
- global, contextual & collective outliers

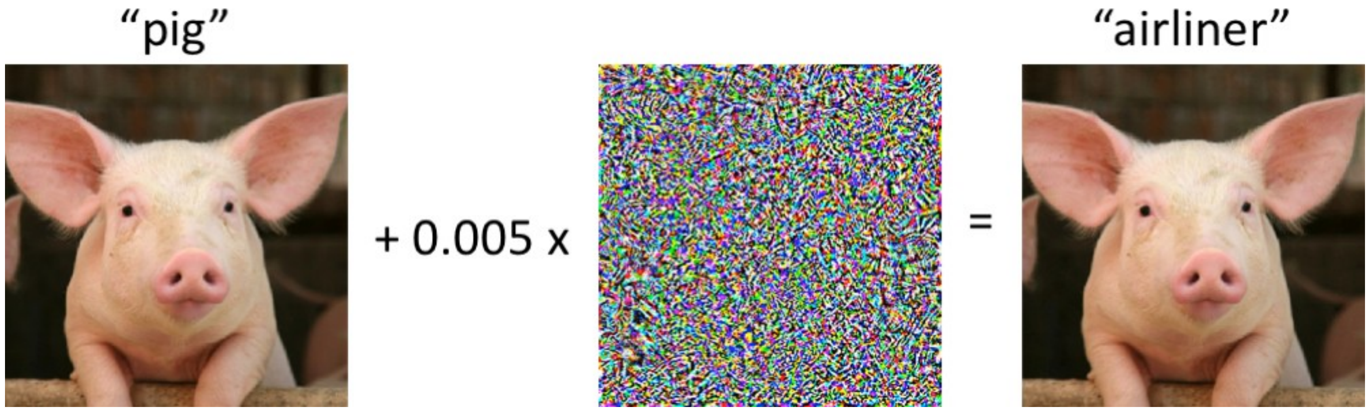
Adversarial Detection

<u>Don’t trust predictions on instances outside of training distribution!</u>

→ ***Adversarial Detection***

- Outliers w.r.t. the model prediction
- Detect small input changes with a big impact on predictions!





Concept Drift

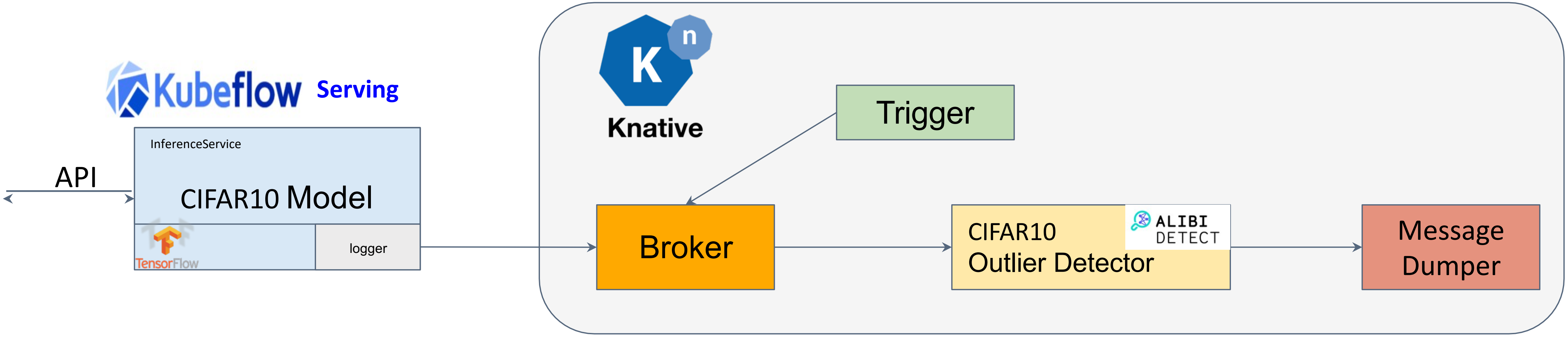
<u>Production data distribution != training distribution?</u>

→ ***Concept Drift! Retrain!***

Need to track the right distributions:

- feature vs. instance level
- continuous vs. discrete
- online vs. offline training data
- track streaming number of outliers

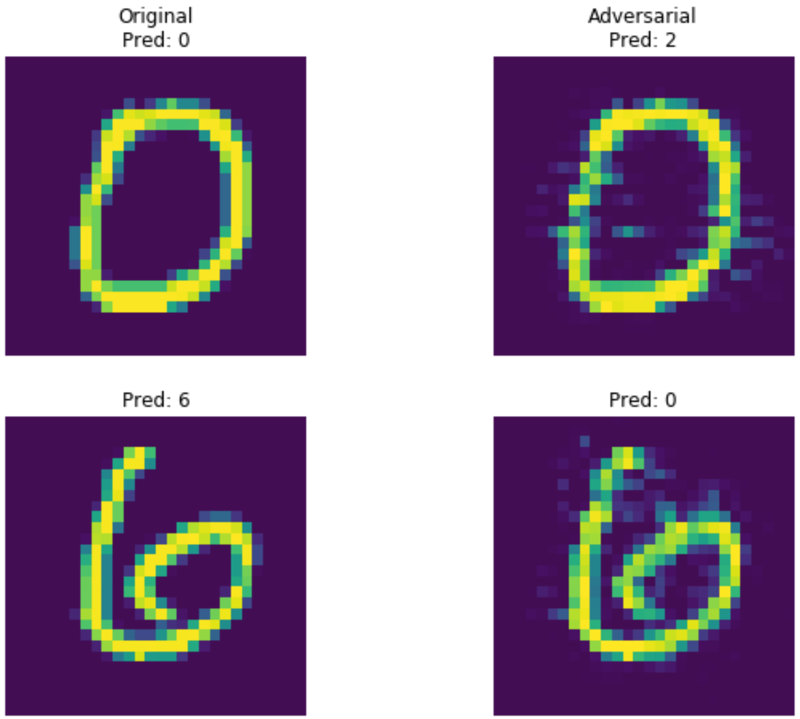
Outlier Detection on CIFAR10



Adversarial Detection Demos

KFServing MNIST Model with Alibi:Detect VAE Adversarial Detector

https://github.com/SeldonIO/alibi-detect/tree/master/integrations/samples/kfserving/ad-mnist



KFServing Traffic Signs Model with Alibi:Detect VAE Adversarial Detector

Adversarial Robustness 360 ↳ (ART)

https://github.com/IBM/adversarial-robustness-toolbox

ART is a library dedicated to adversarial machine learning. Its purpose is to allow rapid crafting and analysis of **attack, defense and detection methods** for machine learning models. Applicable domains include finance, self driving vehicles etc.

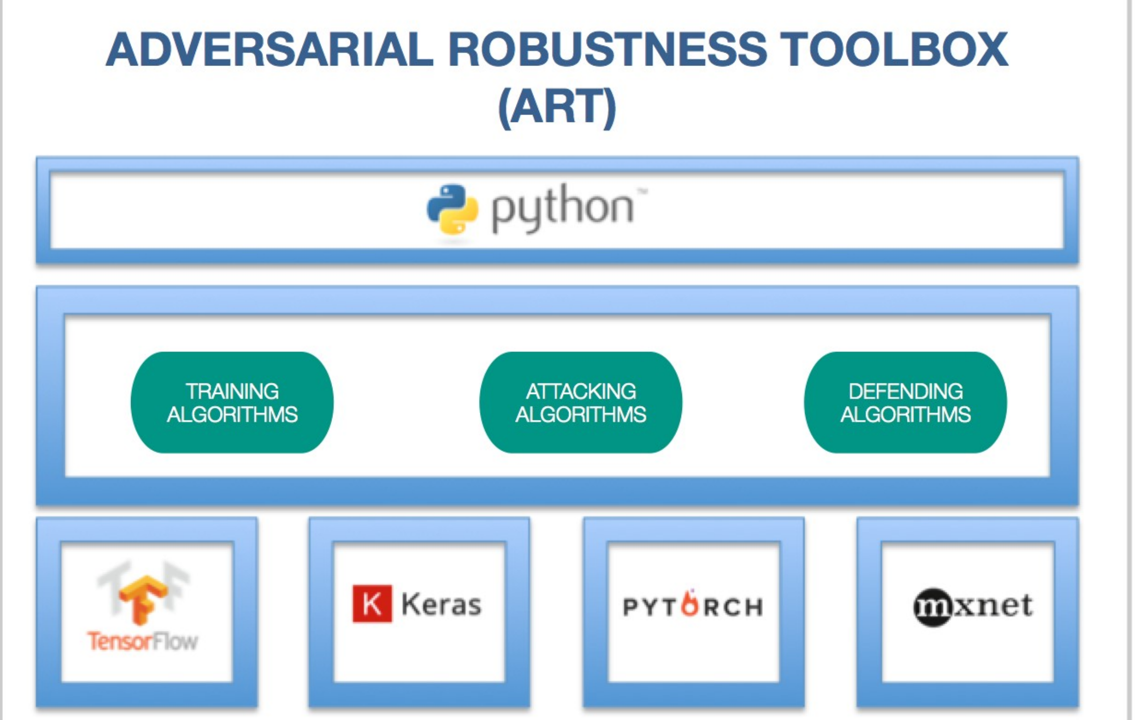
The Adversarial Robustness Toolbox provides an implementation for many state-of-the-art methods for attacking and defending classifiers.

Toolbox: Attacks, defenses, and metrics

Evasion attacks
Defense methods
Detection methods
Robustness metrics

https://art-demo.mybluemix.net/

ART



DEMO

KFServing – Existing Features

- Crowd sourced capabilities – Contributions by AWS, Bloomberg, Google, Seldon, IBM, NVidia and others.
- Support for multiple runtimes pre-integrated (TFServing, Nvdia Triton (GPU optimization), ONNX Runtime, SKLearn, PyTorch, XGBoost, Custom models.
- Serverless ML Inference and Autoscaling: Scale to zero (with no incoming traffic) and Request queue based autoscaling
- Canary and Pinned rollouts: Control traffic percentage and direction, pinned rollouts
- Pluggable pre-processor/post-processor via Transformer: Gives capabilities to plug in pre-processing/post-processing implementation, control routing and placement (e.g. pre-processor on CPU, predictor on GPU)
- Pluggable analysis algorithms: Explainability, Drift Detection, Anomaly Detection, Adversarial Detection (contributed by Seldon) enabled by Payload Logging (built using CloudEvents standardized eventing protocol)
- Batch Predictions: Batch prediction support for ML frameworks (TensorFlow, PyTorch, ...)
- Integration with existing monitoring stack around Knative/Istio ecosystem: Kiali (Service placements, traffic and graphs), Jaeger (request tracing), Grafana/Prometheus plug-ins for Knative)
- Multiple clients: kubectl, Python SDK, Kubeflow Pipelines SDK
- Standardized Data Plane V2 protocol for prediction/explainability et all: Already implemented by Nvidia Triton

KFServing – Upcoming Features

- MMS: Multi-Model-Serving for serving multiple models per custom KFService instance
- More Data Plane v2 API Compliant Servers: SKLearn, XGBoost, PyTorch…
- Multi-Model-Graphs and Pipelines: Support chaining multiple models together in a Pipelines
- PyTorch support via AWS TorchServe
- gRPC Support for all Model Servers
- Support for multi-armed-bandits
- Integration with IBM AIX360 for Explainability, AIF360 for Bias detection and ART for Adversarial detection

Open Source Projects

• ML Inference ◦ KFServing ◦ Seldon Core	https://github.com/kubeflow/kfserving https://github.com/SeldonIO/seldon-core
• Model Explanations ◦ Seldon Alibi ◦ IBM AI Explainability 360	https://github.com/seldonio/alibi https://github.com/IBM/AIX360
• Outlier and Adversarial Detection and Concept Drift ◦ Seldon Alibi-detect	https://github.com/seldonio/alibi-detect
• Adversarial Attack, Detection and Defense ◦ IBM Adversarial Robustness 360	https://github.com/IBM/adversarial-robustness-toolbox