

# Beam ML past, present and future

Kerry Donny-Clark & Reza Rokni



# I'm Kerry





I'm Reza





# Agenda



- Intro
- The Past: RunInference is born
- The Present: Model handlers, model updates, model zoos, and more!
- The Future: Make ML tasks easy
- Q&A



Turn key solutions....



# Apache Beam

## Java

```
Input.apply  
(Sum.integersPerKey())
```

## Python

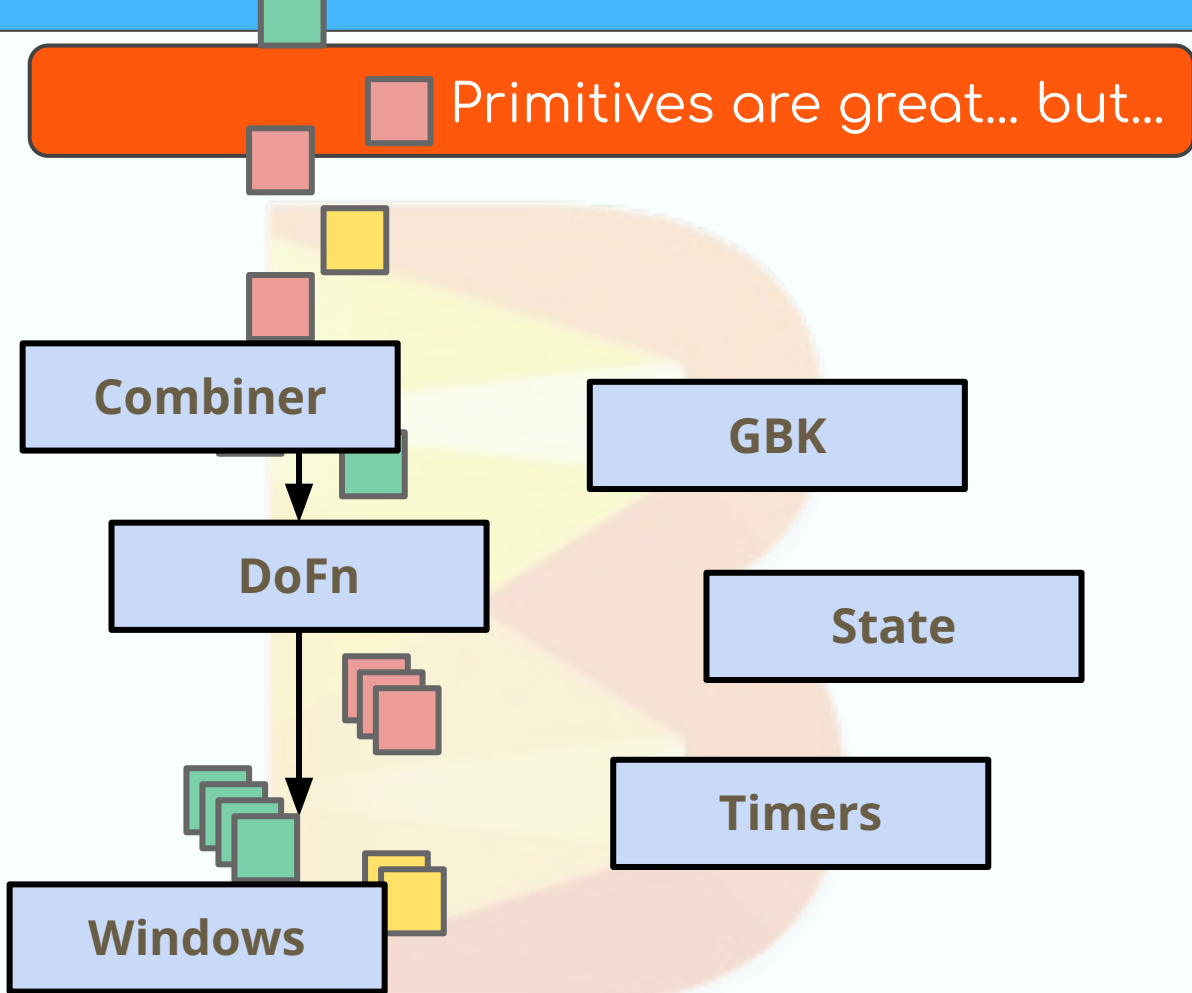
```
input | Sum.PerKey()
```

## SQL

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

## Go

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

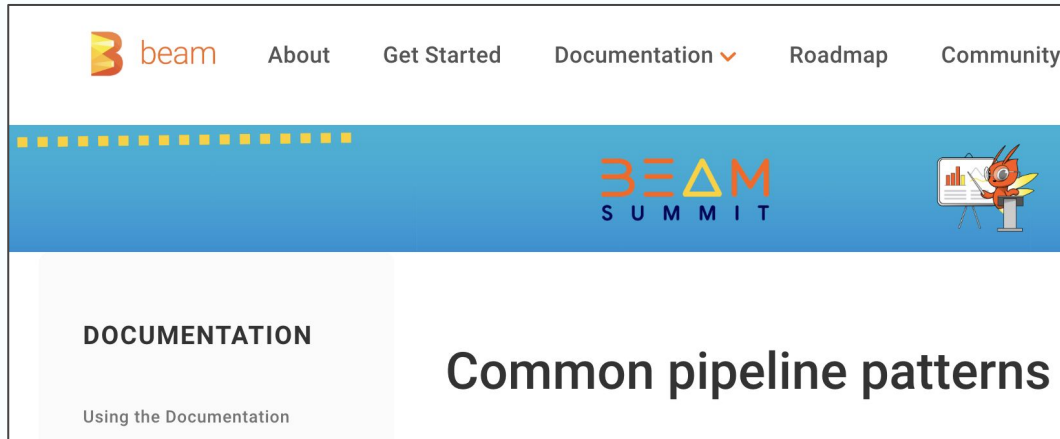


# Apache Beam

Primitives are great.. but..



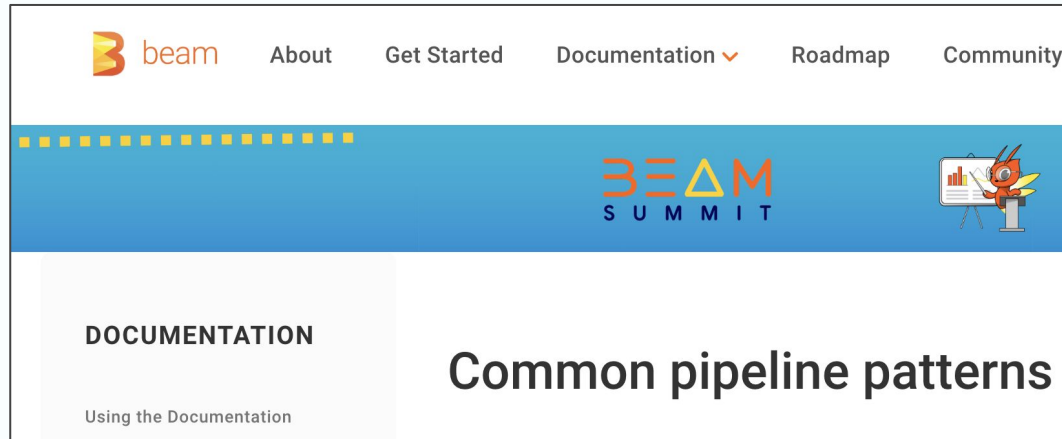
Patterns are great ... but



The screenshot shows the Apache Beam documentation website. At the top, there is a navigation bar with the following items: the Apache Beam logo (a stylized 'B' with 'beam' text), 'About', 'Get Started', 'Documentation' (with a dropdown arrow), 'Roadmap', and 'Community'. Below the navigation bar is a blue banner with a dashed yellow line on the left. The banner contains the 'BEAM SUMMIT' logo (with 'BEAM' in orange and 'SUMMIT' in blue) and a cartoon character of a red robot with yellow wings standing next to a presentation board. Below the banner, the page content is divided into two columns. The left column has a grey background and contains the text 'DOCUMENTATION' and 'Using the Documentation'. The right column has a white background and contains the main heading 'Common pipeline patterns'.



Patterns are great ... but

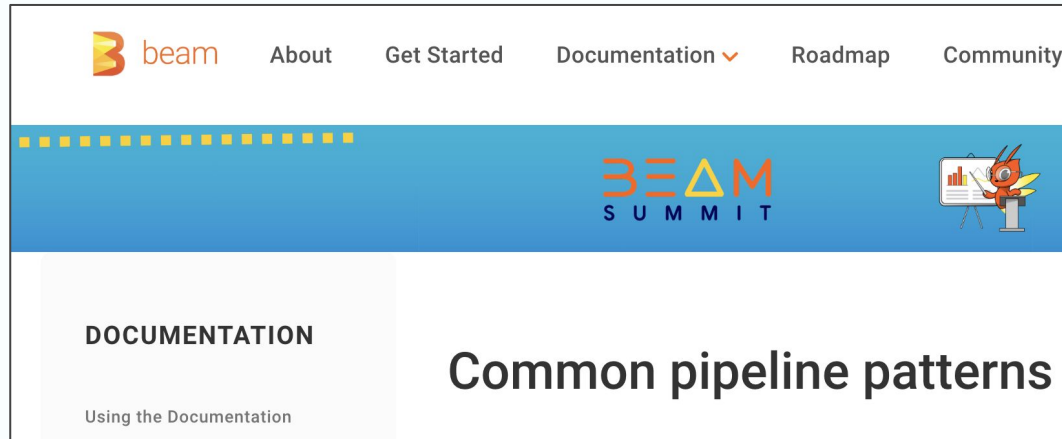


The screenshot shows the Apache Beam documentation website. At the top, there is a navigation bar with the following items: the Apache Beam logo (a stylized 'B' with 'beam' text), 'About', 'Get Started', 'Documentation' (with a dropdown arrow), 'Roadmap', and 'Community'. Below the navigation bar is a blue banner with the 'BEAM SUMMIT' logo and a cartoon character holding a presentation board. The main content area features a sidebar on the left with the heading 'DOCUMENTATION' and a sub-link 'Using the Documentation'. The main heading of the page is 'Common pipeline patterns'.



Which set of patterns do I need...?

Patterns are great ... but

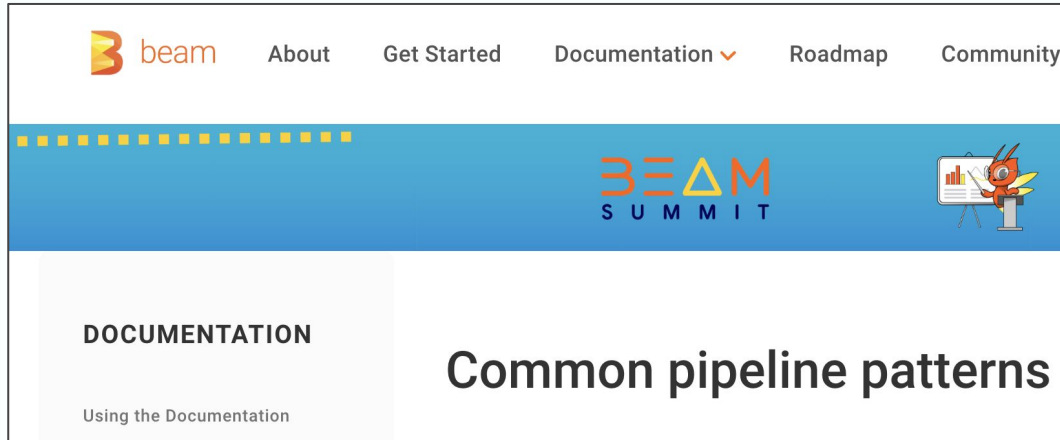


The screenshot shows the top navigation bar of the Beam website with links for 'About', 'Get Started', 'Documentation', 'Roadmap', and 'Community'. Below the navigation is a blue banner for 'BEAM SUMMIT' featuring a cartoon character. The main content area has a sidebar with 'DOCUMENTATION' and 'Using the Documentation' links, and a main heading for 'Common pipeline patterns'.



Set reminder..

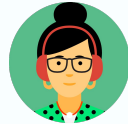
Patterns are great ... but



The screenshot shows the Apache Beam documentation website. At the top, there is a navigation bar with the Beam logo and links for 'About', 'Get Started', 'Documentation' (with a dropdown arrow), 'Roadmap', and 'Community'. Below the navigation bar is a blue banner with the 'BEAM SUMMIT' logo and an illustration of a character presenting. The main content area features a 'DOCUMENTATION' sidebar on the left with the sub-link 'Using the Documentation'. The main heading is 'Common pipeline patterns'.



Writing that code!



Writing same code!



Yup same again!



Déjà vu?

## Turn Key transforms

```
with beam.Pipeline(options=pipeline_options) as p:  
    (p  
     | beam.io.fileio.MatchFiles(gs://my_bucket/*)  
     | beam.io.fileio.ReadMatches()  
     | beam.Map(preprocess_image)  
     | beam.xxx_pattern_xxx(Configuration)  
     ...
```

## credit karma<sup>®</sup>

### Self-service Machine Learning Workflows and Scaling MLOps with Apache Beam

Apache Beam has future-proofed Credit Karma's data and ML platform for scalability and efficiency, enabling MLOps with unified pipelines, processing 5-10 TB daily at 5K events per second, and managing 20K+ ML features.



### Real-time ML with Beam at Lyft

Lyft Marketplace team aims to improve our business efficiency by being nimble to real-world dynamics. Apache Beam has enabled us to meet the goal of having a robust and scalable ML infrastructure for improving model accuracy with features in real-time. These real-time features support critical functions like

### Large-Scale Generation of ML Podcast Previews at Spotify with Google Dataflow













April 15, 2023  
Published by Diego Casabuena (ML Engineer, Spotify), Edgar Tanaka (ML Engineer, Spotify), Winstead Zhu (ML Engineer, Spotify), Reza Rokni (PM, Google Dataflow ML), and Danny McCormick (Senior Software Engineer, Google)



### Integrating the Podz ML pipeline into Spotify

# Lots of community engagement!

## Beam Summit 2022:

Machine learning design patterns: between Beam and a hard place	Lak Lakshmanan		
Vega: Scaling MLOps Pipelines at Credit Karma using Apache Beam and Dataflow	Debasish Das, Vishnu Venkataraman & Raj Katakam		
Implementing Cloud Agnostic Machine Learning Workflows with Apache Beam on Kubernetes	Charles Adetiloye & Alexander Lerma		
Streaming NLP infrastructure on Dataflow	Alex Chan & Angus Neilson		
Improving Beam-Dataflow Pipelines for Text Data Processing	Sayak Paul & Nilabhra Roy Chowdhury		
The Ray Beam Runner Project: A Vision for Unified Batch, Streaming, and ML	Patrick Ames, Jiajun Yao & Chandan Prasad		

# Beam RunInference

## Removing boiler plate chore!

```
with beam.Pipeline(options=pipeline_options) as p:  
    (p  
     | beam.io.fileio.MatchFiles(gs://my_bucket/images*)  
     | beam.io.fileio.ReadMatches())
```



## Removing boiler plate chore!

```
with beam.Pipeline(options=pipeline_options) as p:  
    (p  
     | beam.io.fileio.MatchFiles(gs://my_bucket/images*)  
     | beam.io.fileio.ReadMatches()  
     | beam.Map(preprocess_image))
```

## Removing boiler plate chore!

```
class MyComplicatedPridctionStuff (beam.DoFn) :
```

## Removing boiler plate chore!

```
class MyComplicatedPredictionStuff(beam.DoFn):  
    def setup():  
        #Code for loading once  
        ...  
  
    def process(self, element):  
        #Use model handle to call
```

## Removing boiler plate chore!

```
class MyComplicatedPredictionStuff(beam.DoFn):  
    def setup():  
        #Code for loading once  
        ...  
  
    def process(self, element):  
        #Use model handle to call  
        ...  
        #Handle errors, do nice error logging
```

## Removing boiler plate chore!

```
class MyComplicatedPredictionStuff(beam.DoFn):  
    def setup():  
        #Code for loading once  
        ...  
  
    def process(self, element):  
        #Use model handle to call  
        ...  
        #Handle errors, do nice error logging  
        ...  
        #Output useful metrics from the process  
        ...
```

## Removing boiler plate chore!

```
class MyComplicatedPredictionStuff(beam.DoFn):  
    def setup():  
        #Code for loading once  
        ...  
  
    def process(self, element):  
        #Use model handle to call  
        ...  
        #Handle errors, do nice error logging  
        ...  
        #Output useful metrics from the process  
        ...  
        TODO Oh wait! I need to batch stuff first ...
```

## Removing boiler plate chore!

```
class MyComplicatedPredictionStuff(beam.DoFn):
    def setup():
        TODO Code for loading once .....

    def process(self, element):
        TODO Use model handle to call
        TODO Handle errors, do nice error logging
        TODO Output useful metrics from the process
        TODO Oh wait! I need to batch stuff first ...
        TODO Wait.. I need model configuration ....
```

## Removing boiler plate chore!

```
with beam.Pipeline(options=pipeline_options) as p:  
    (p  
     | beam.io.fileio.MatchFiles(gs://my_bucket/images*)  
     | beam.io.fileio.ReadMatches()  
     | beam.Map(preprocess_image)  
     | beam.ml.inference.RunInference(model_handler)  
     ...
```



# PytorchModelHandlerTensor

Framework

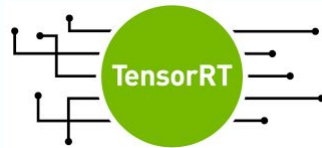
Type

# The Present: Model Handlers



 PyTorch

*dmlc*  
**XGBoost**



# The Present: Notebooks

<https://github.com/apache/beam/tree/master/examples/notebooks/beam-ml>

- **Prediction and inference with pretrained models**
- **Custom inference**
- **Automatic Model Refresh**
- **Multi-model pipelines**
- **Model Evaluation**
- **Data processing**

DEMO

[LLM Demo Link](#)

# The Present: Model Zoos

CLASSIFIER\_URL=

["https://tfhub.dev/google/wiki40b-lm-en/1"](https://tfhub.dev/google/wiki40b-lm-en/1)

with pipeline as p:

predictions = (

p

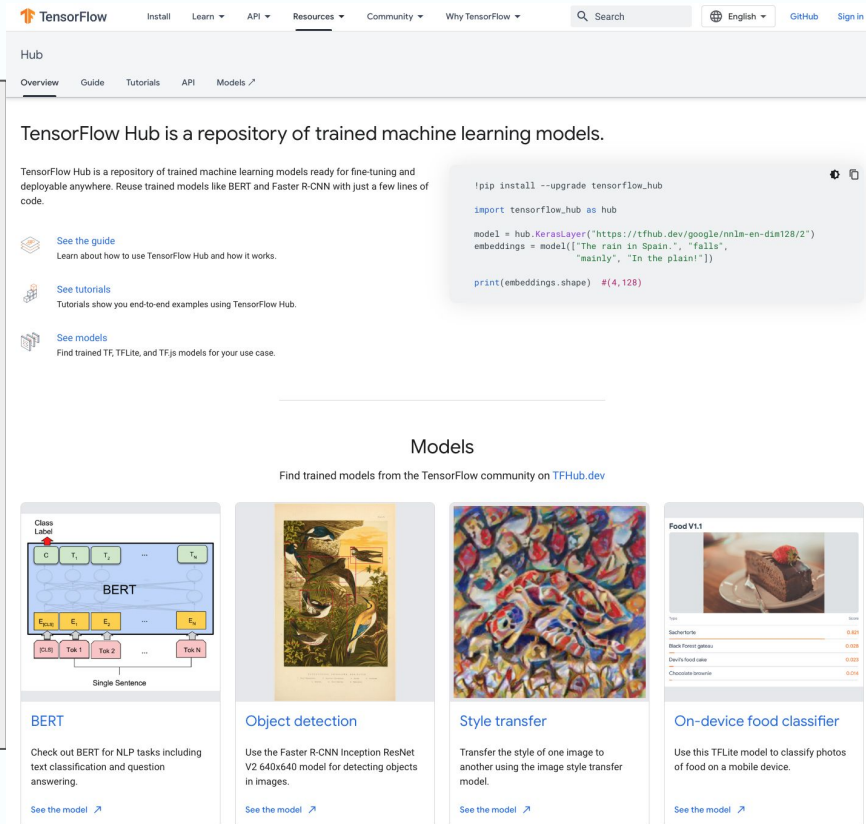
| ReadFromText(known\_args.input)

| RunInference(

TensorHubhandler(

{uri=CLASSIFIER\_URL}

))



The screenshot shows the TensorFlow Hub website. At the top, there's a navigation bar with 'TensorFlow' logo and links for 'Install', 'Learn', 'API', 'Resources', 'Community', and 'Why TensorFlow'. Below the navigation bar, the main heading is 'Hub' with sub-links for 'Overview', 'Guide', 'Tutorials', 'API', and 'Models'. The main content area starts with the text 'TensorFlow Hub is a repository of trained machine learning models.' followed by a brief description: 'TensorFlow Hub is a repository of trained machine learning models ready for fine-tuning and deployable anywhere. Reuse trained models like BERT and Faster R-CNN with just a few lines of code.' There are three links: 'See the guide', 'See tutorials', and 'See models'. A code block shows a Python snippet for installing and using a model. Below this, there's a 'Models' section with the text 'Find trained models from the TensorFlow community on TFHub.dev'. Four model cards are displayed: 'BERT' (text classification), 'Object detection' (image detection), 'Style transfer' (image style transfer), and 'On-device food classifier' (mobile food classification).

# DEMO

[notebooks/beam-ml/run\\_inference\\_with\\_tensorflow\\_hub.ipynb](#)

## The Present: Model map() encapsulation

```
inference = pcoll | RunInference(model_handler.with_postprocess_fn(lambda x : do_someth  
ing_to_result(x)))
```

```
inference = pcoll | RunInference(model_handler.with_preprocess_fn(lambda x : do_somethi  
ng(x)))
```

```
inference = pcoll | RunInference(  
  model_handler.with_preprocess_fn(  
    lambda x : do_something(x)  
  ).with_preprocess_fn(  
    lambda x : do_something_else(x)  
  ).with_postprocess_fn(  
    lambda x : do_something_after_inference(x)  
  ).with_postprocess_fn(  
    lambda x : do_something_else_after_inference(x)  
  ))
```

## The Present: Error handling

```
main, other = pcoll | RunInference(model_handler).with_exception_handling()  
other.failed_inferences | beam.Map(print) # insert logic to handle failed records here
```

```
main, other = pcoll | RunInference(model_handler.with_preprocess_fn(f1).with_postprocess  
s_fn(f2)).with_exception_handling()  
other.failed_preprocessing[0] | beam.Map(print) # handles failed preprocess operations,  
indexed in the order in which they were applied  
other.failed_inferences | beam.Map(print) # handles failed inferences  
other.failed_postprocessing[0] | beam.Map(print) # handles failed postprocess operation  
s, indexed in the order in which they were applied
```



# The Present: Streaming Model Updates

RunInference auto-model update

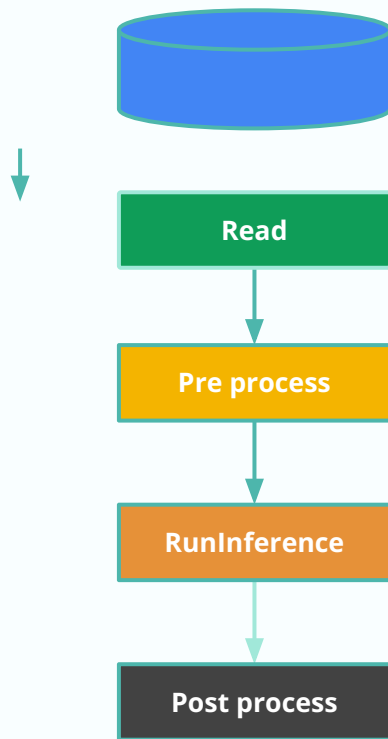
Two modes:

1. Watch Mode

Upload updated model to files stores like GCS and RunInference will auto pull the new model for you

2. Event Mode

Push an update message to RunInference via a streaming source such as Kafka



 PyTorch

 scikit  
learn

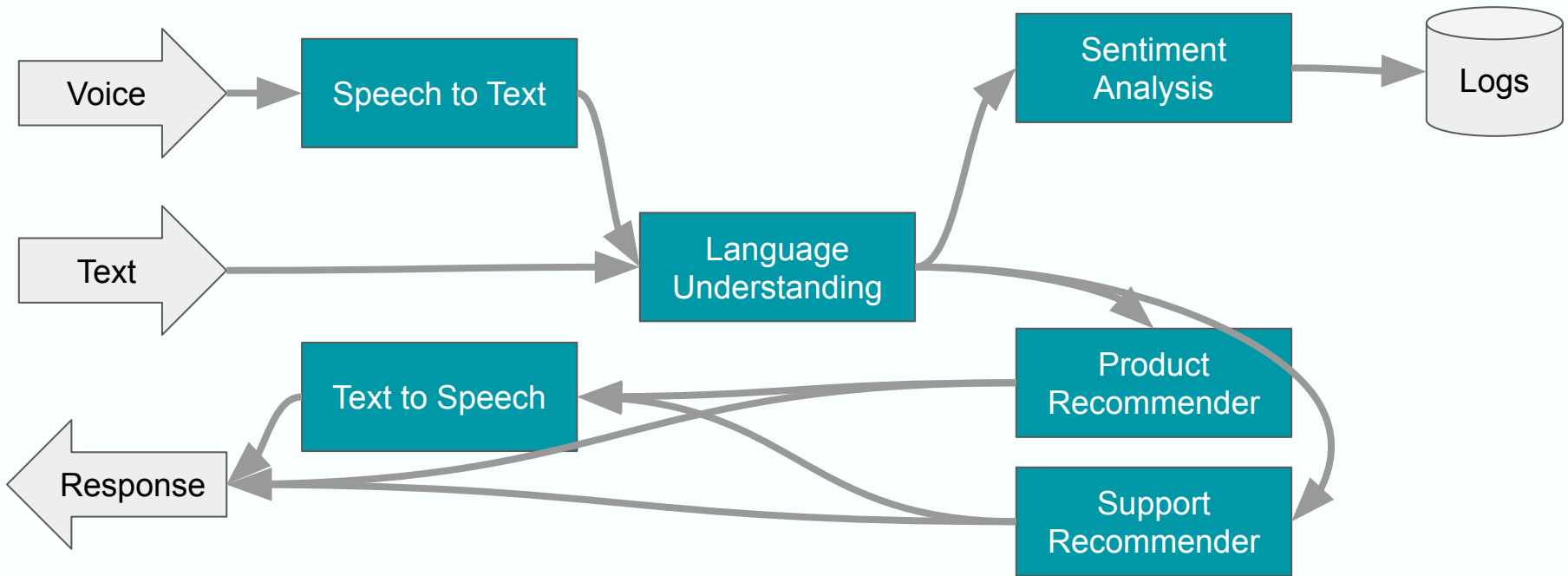


## The Present: Efficient large models

```
share_model_across_processes() → bool \[source\]
```

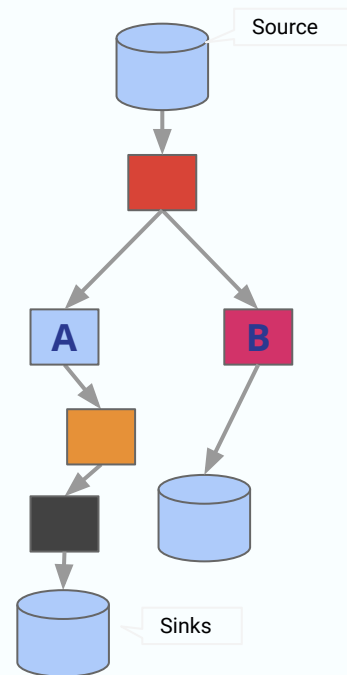
**apache\_beam.utils.multi\_process\_shared** module

# The Present: Multi-Model Ensembles



## The Present: Branched (A/B) models

```
data = p | beam.io.textio(files)
data | RunInference(model_a_handler)
data | RunInference(model_b_handler)
```

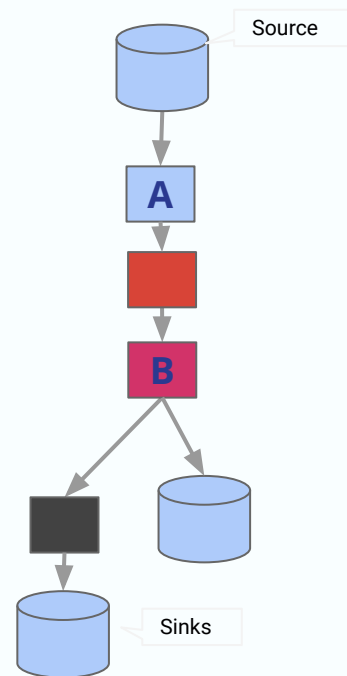


# The Present: Sequential Models

```
data = p | beam.io.textio(files)

model_a_output =
data | RunInference(model_a_handler)

model_a_output
| Map(postprocess)
| RunInference(model_b_handler)
```



## Hugging Face Model Handler for RunInference

Ritesh Ghose ([riteshghorse@apache.org](mailto:riteshghorse@apache.org))

```
with pipeline as p:  
    predictions = (  
        p  
        | beam.ReadFromSource('a_source')  
        | RunInference(  
            HuggingFaceModelHandler(...))
```

Future : Models from endpoints

## [WIP] Vertex AI Remote Model Handler #27091

 Draft jrmcccluskey wants to merge 3 commits into `apache:master` from `jrmcccluskey:vertexAI` 



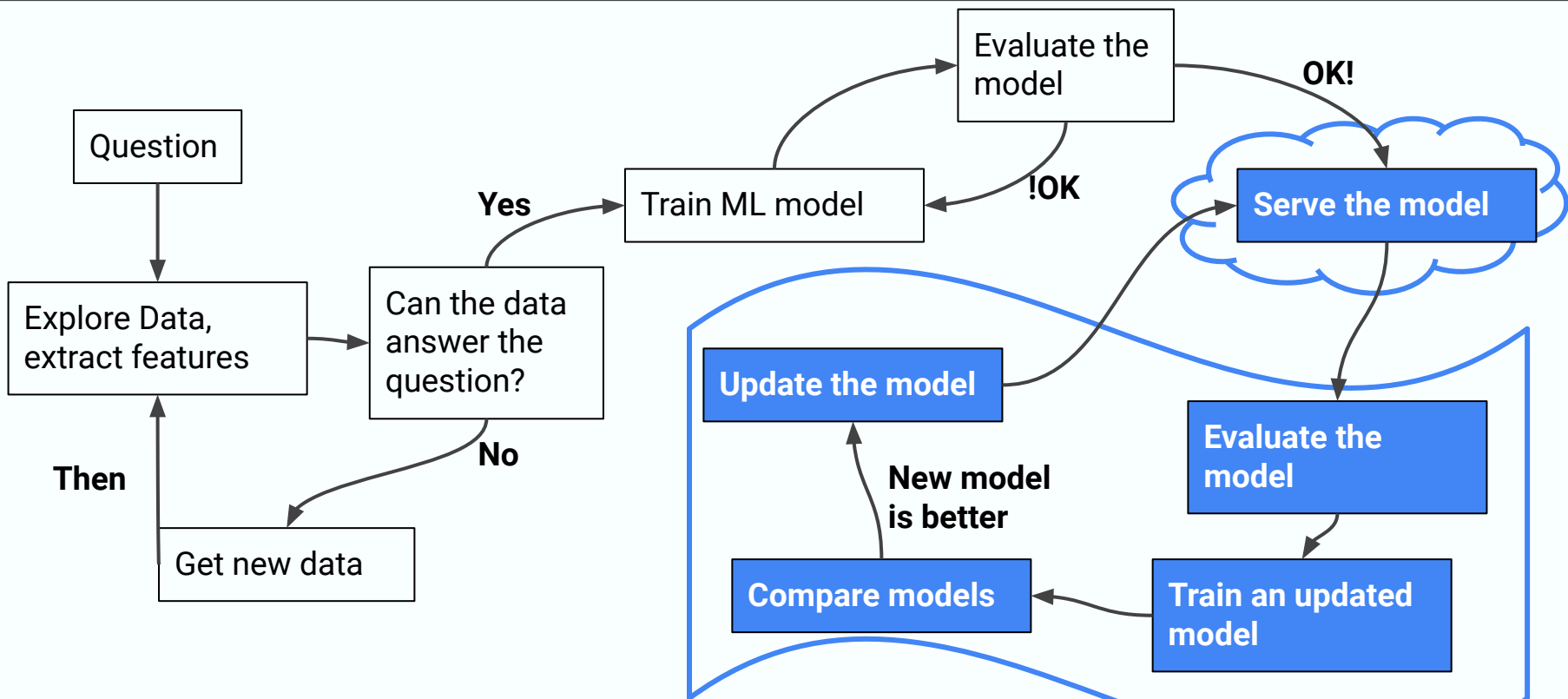
# Beyond Inference !





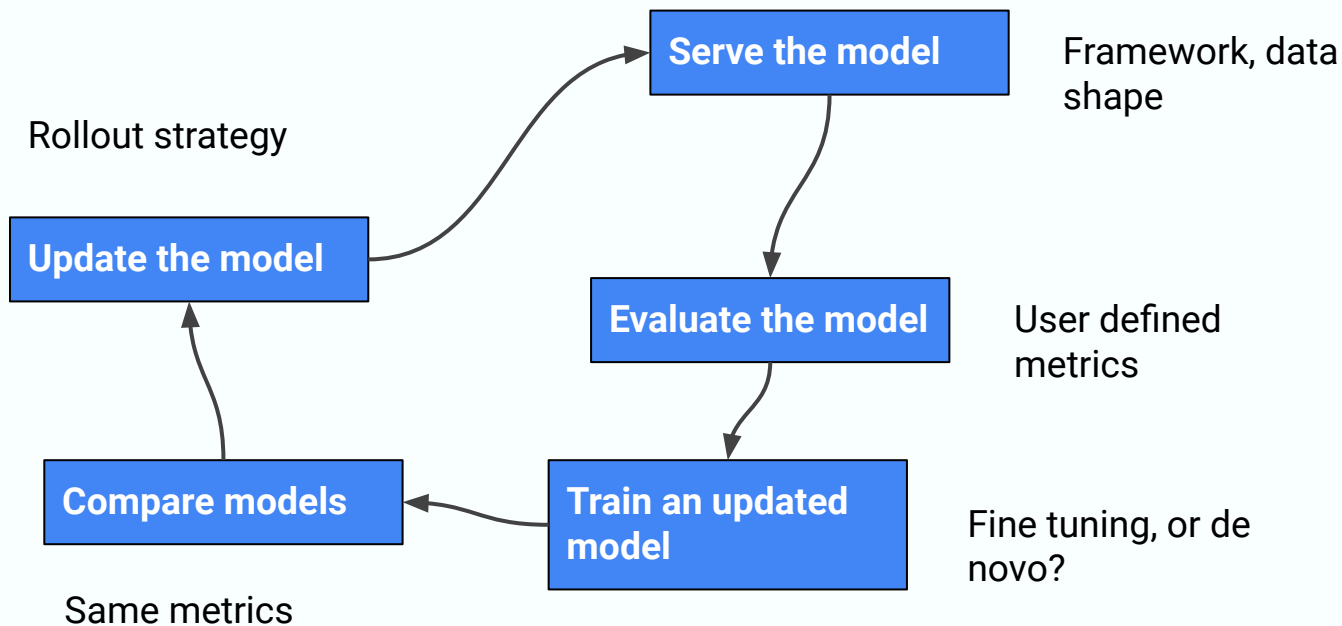


# Beyond Inference !





# Beyond Inference !



## Coming soon

### Beam MLTransform

<https://s.apache.org/beam-mltransform>

Anand Inguva ([anandinguva@google.com](mailto:anandinguva@google.com))

Last updated: May 30th, 2023

# WIP!

```
beam.MLTransform(  
    process_handler=ProcessHandler([  
        scale_to_0_1(  
            columns={'x': int, 'y': List[int]},  
            compute_and_apply_vocab(  
                columns={'x': int, 'y': List[int]})))])
```

# Turn Key Transformations

Coming soon



Reza Rokni & Kerry Donny-Clark

# QUESTIONS?

The logo for BEAM SUMMIT is centered on a blue background. The word "BEAM" is written in a stylized, bold font. The 'B' is orange, the 'E' is orange, the 'A' is a yellow triangle, and the 'M' is orange. Below "BEAM", the word "SUMMIT" is written in a dark blue, bold, sans-serif font.

**BEAM**  
**SUMMIT**