Talk to your pipeline: use AI to create dynamic transforms in streaming

Kfir Naftali & Israel Herraiz

BEAM SUMMIT NYC 2025

Streaming inference

Beyond just calling a model,

what does mean to be able to make

real time inference?

Authors

Kfir Naftali

Israel Herraiz

Agenda

- Beam ML and **turnkey transforms**
- How to **prepare your data** for a dynamic transformation?
- Side inputs: questions and code
- Conclusions & sample **repositories**

Beam ML

Beam ML: Run inference

https://beam.apache.org/documentation/ml/overview/

```
model_handler = PytorchModelHandlerTensor(
                 state_dict_path='gs://path/to/my_model.pt',
                 model_class=my_model_class,
                 model_params={'input_dim': 1, 'output_dim': 1},
)
with beam.Pipeline(options=pipeline_options)  as p:
    (p
    | beam.io.ReadFromPubSub( my_topic)
    | beam.Map(preprocess)
    | beam.ml.inference.RunInference(model_handler=<config>)
    | beam.Map(post_process)
    ...
```

What models can I use with RunInference?

- Local models
  - Tensorflow
  - PyTorch
  - VLLM
  - sklearn

```
example = ["translate English to Spanish: We are in New York City."]

pipeline = beam.Pipeline(options=PipelineOptions(save_main_session=True,pickle_library="cloudpickle"))

with pipeline as p:
  _ = (
          p
          | "Create Examples" >> beam.Create(example)
          | "To tensors" >> beam.Map(to_tensors, tokenizer)
          | "RunInference"
            >> RunInference(
                model_handler,
                inference_args={"max_new_tokens": MAX_RESPONSE_TOKENS},
            )
          | "From tensors" >> beam.Map(from_tensors, tokenizer)
          | "Print" >> beam.Map(print)
      )
```

```
Estamos en Nueva York City.
```

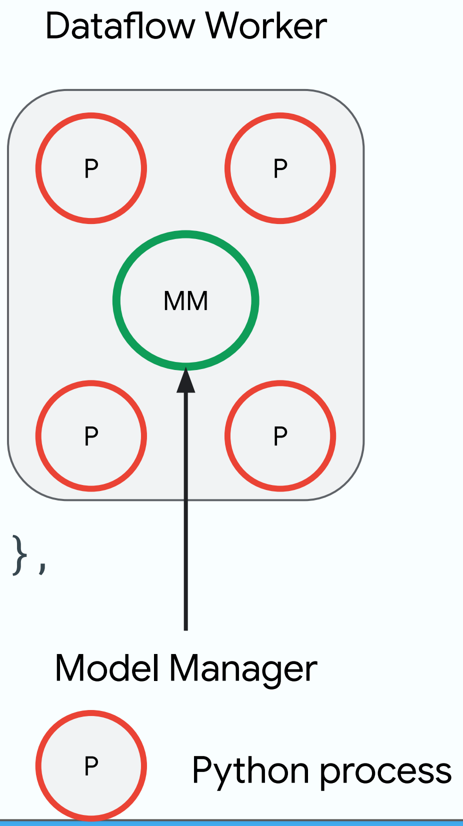
https://cloud.google.com/dataflow/docs/notebooks/run_inference_generative_ai

Local models? Aren’t those too heavy?

Memory management

- Model sharing within process
    - RunInference takes care of that by default
- Model sharing across processes?
    - Available with large_model=true

```
>>> model_handler = PytorchModelHandlerTensor(
...     model_class=LinearRegression,
...     large_model=True,
...     model_params={'input_dim': 1, 'output_dim': 1},
...     state_dict_path='gs://path/to/model.pt')
```



What models can I use with RunInference?

- Remote models
  - HuggingFace, TensorflowHub, Vertex AI endpoints
  - Really, anything that you can call from your pipeline (custom handler)

Implementing a ML Pipeline with Google AI Studio

Presented at Beam College 2025

This tutorial demonstrates how to perform streaming inference with Apache Beam and Google AI Studio's Gemini model, based example to get country capitals.

It covers:

1. Setup
2. Prompt Engineering in Gemini
3. Building an ML pipeline with Beam
4. Running the pipeline

Resources:

- Starting (blank) notebook
- Notebook with solution

https://beamcollege.dev/sessions/2025/implementing-ml-pipeline-ai-studio/

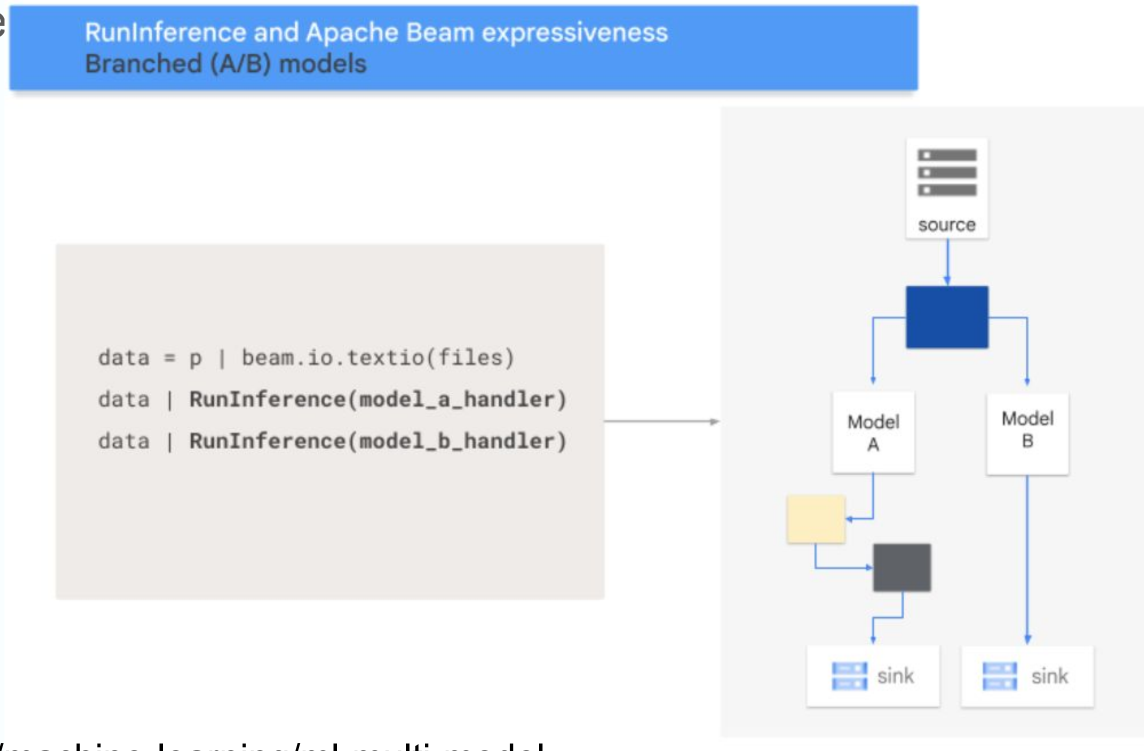
How difficult is to write a custom model handler?

```python
class CloudVisionModelHandler(RemoteModelHandler):
  def __init__(self):
    """DoFn that accepts a batch of images as bytearray
    and sends that batch to the Cloud Vision API for remote inference
    """
    super().__init__(namespace="CloudVisionModelHandler", retry_filter=_always_retry)
  def create_client(self):
    """Initiate the Google Vision API client."""
    client = vision.ImageAnnotatorClient()
    return client

  def request(self, batch, model, inference_args):
    feature = Feature()
    feature.type_ = Feature.Type.LABEL_DETECTION

    # The list of image_urls
    image_urls = [image_url for (image_url, image_bytes) in batch]

    # Create a batch request for all images in the batch.
    images = [vision.Image(content=image_bytes) for (image_url, image_bytes) in batch]
    image_requests = [vision.AnnotateImageRequest(image=image, features=[feature]) for image in images]
    batch_image_request = vision.BatchAnnotateImagesRequest(requests=image_requests)

    # Send the batch request to the remote endpoint.
    responses = model.batch_annotate_images(request=batch_image_request).responses

    return list(zip(image_urls, responses))
```

https://cloud.google.com/dataflow/docs/notebooks/custom_remote_inference

What models can I use with RunInference?

- Multiple models in the same pipeline
  - Ensemble
  - Cohorts of models (A/B pattern)



https://cloud.google.com/dataflow/docs/machine-learning/ml-multi-model

Data preparation for dynamic transformations

BEAM SUMMIT
NYC 2025

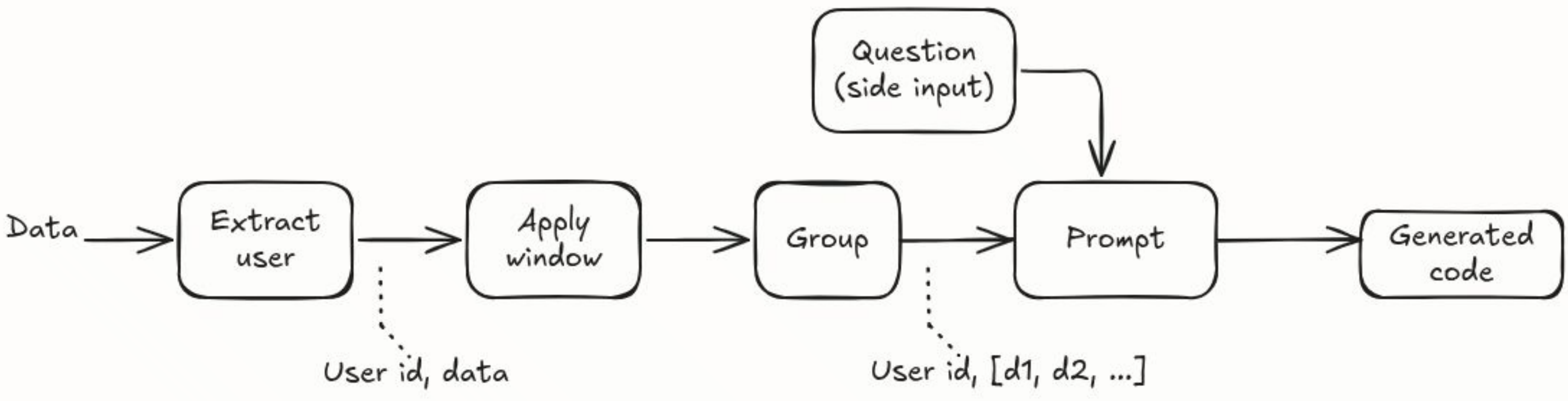
Data must be processed per key

- Partitioning required, vertical scalability issue
- Calculations can only be dynamic if they are inside a data processing step
- The partition define what kind of questions can be asked
- For instance, game activity, key by user id

Windowing

- The data (and metadata) for the transformation will be added to the prompt as context
- Again, vertical scalability issue
- This defines the granularity of the answers
- But also greatly improves the accuracy of the generated code to solve the question

Logical pipeline (static)

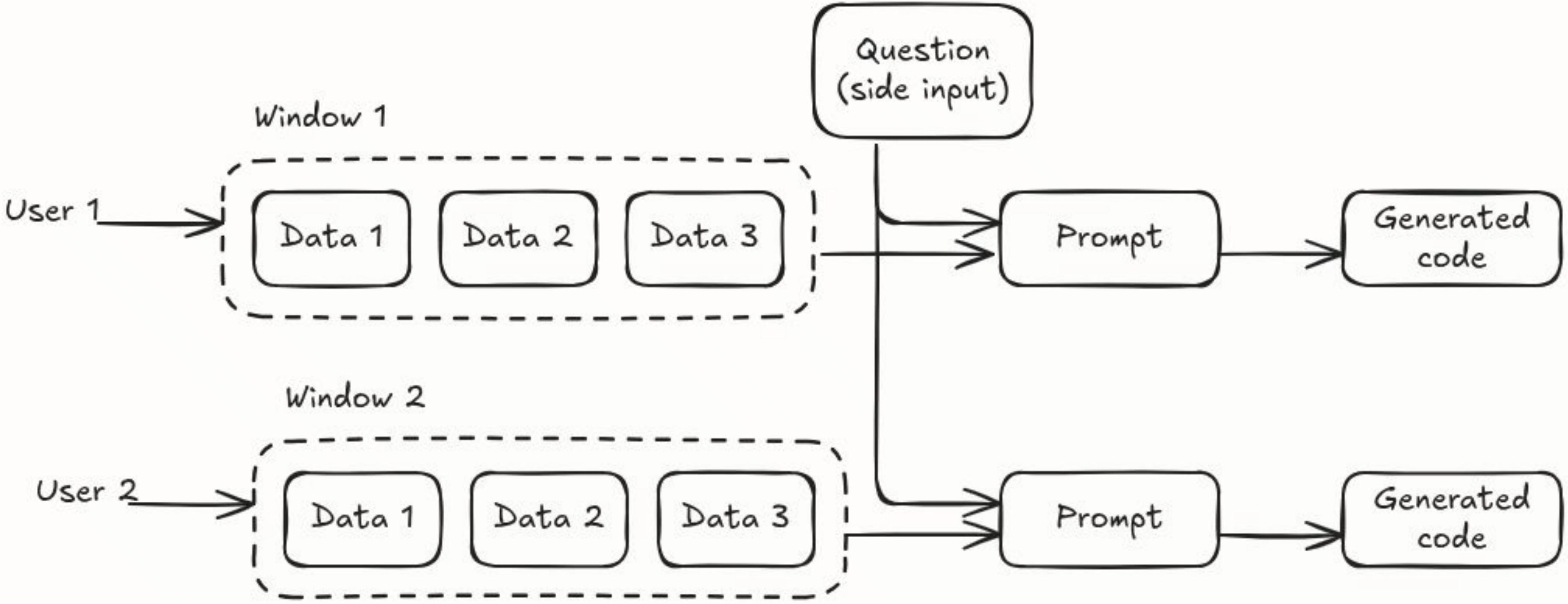


Side inputs: prompt creation and code execution

Prompt: importance of context

- Questions are important for the generation of the right code
- But context is almost as important
    - Data provide a lot of hints to the model about what code needs to be generated
    - Metadata (e.g. schema) also helps the model to create accurate code
- Side inputs
    - Questions will be small, so we can “join” with the data through a side input
- Prompt side needs to be:
    - Small enough as to fit in the worker memory
    - Small enough for the model used
        - For instance, Gemma 3 has a limit of 128k tokens
    - Large enough as to provide enough context to facilitate the task to the model

Physical execution (generate code)



Prompt structure

Context: Based *only* on the schema and sample data from a 1-minute window of 'gaming_events', generate a SQL query for the user's question.

Table name: 'gaming_events'

Schema: [Schema extracted from the data]

Sample data (first 3 rows): [Data from the group formatted as CSV]

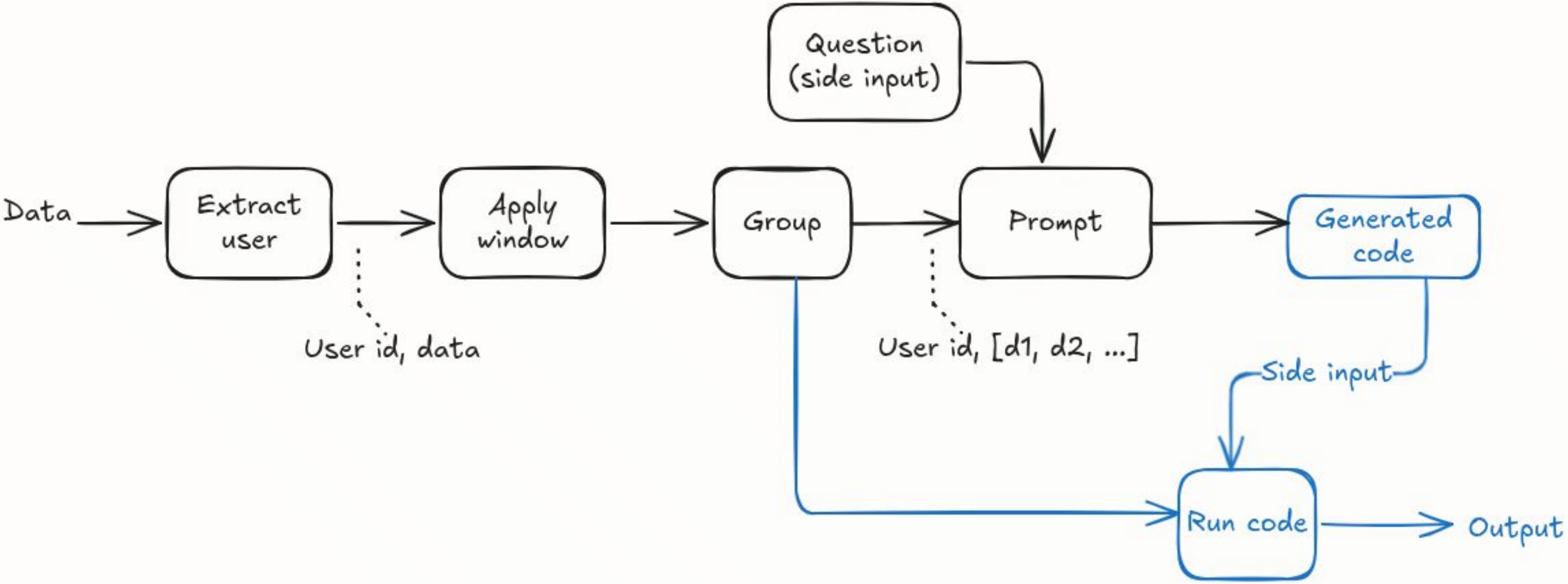
User question: [Side input]

Generated code:

Generated code

Once the code is generated, how can I **apply it**?

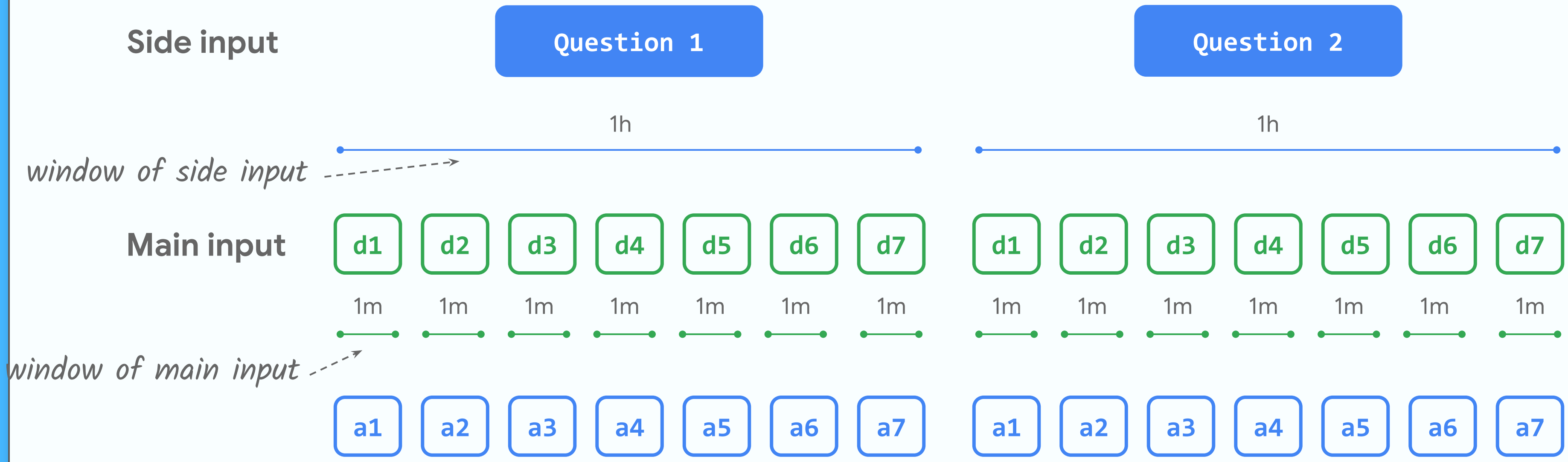
Full pipeline (static)



Side inputs in streaming

- When a `PCollectionView` of a windowed `PCollection` is created, the `PCollectionView` represents a single entity per window (one singleton per window in this case).
- Beam projects the main input element's window into the side input's window set, and then uses the side input from the resulting window.
  - **Identical windows** ➞ projection provides exact corresponding window.
  - **Different windows** ➞ projection used to choose most suitable side input window.
- If the main input element exists in more than one window, `processElement` gets called once for each window. Each call projects the "current" window for the main input element, and thus might provide a different view of the side input each time.
- If the side input has multiple trigger firings, the value from the latest trigger firing is used.

Side inputs in streaming: careful with the windows



Conclusions and sample projects

Beam, framework for complex inference patterns

- Inference is much more than calling a model for a punctual prediction
- Beam greatly simplifies the creation of complex patterns for streaming inference
- The future of AI is context
  - Leverage data and metadata to improve the accuracy of the model in providing the best code to solve the question

https://github.com/kfirnaftali/Talk-to-your-data

github.com/GoogleCloudPlatform/dataflow-solution-guides

Kfir Naftali & Israel Herraiz

QUESTIONS?

linkedin.com/in/kfir-naftali/
linkedin.com/in/herraiz/