Easy cross-language with SchemaTransforms: use your favorite Java transform in Python SDK

Ahmed Abualsaud

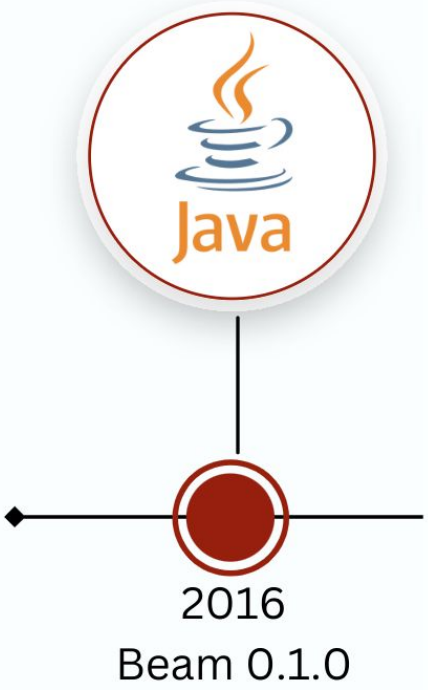
BEAM SUMMIT
NYC 2023

About me

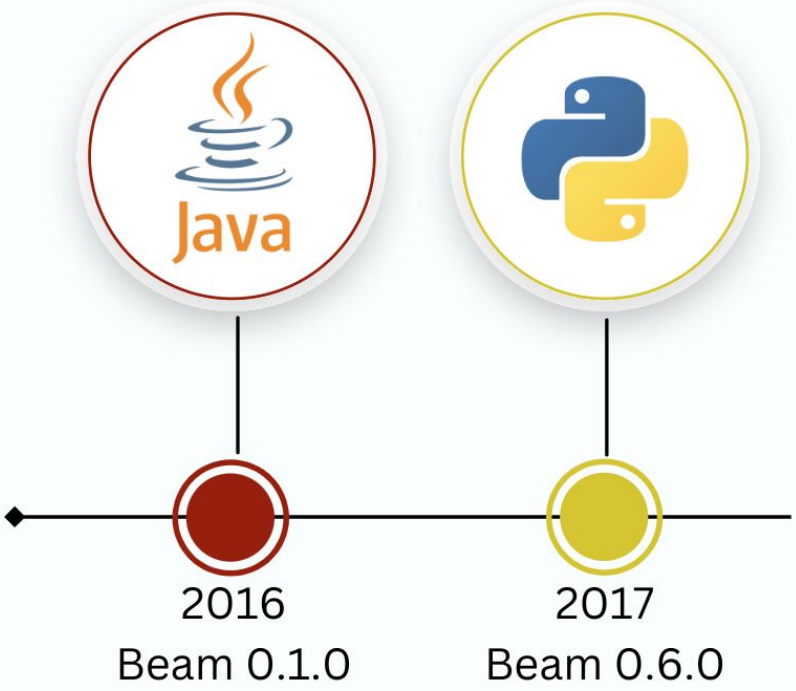
Agenda

- Background and problem statement
  - Why cross-language is important for the future of Beam
- What is a SchemaTransform?
  - Why is it called a SchemaTransform?
- Creating a SchemaTransform
- Running a Java expansion service
- Demo: Using Java's wordcount transform in a Python pipeline
- Current limitations/unknowns

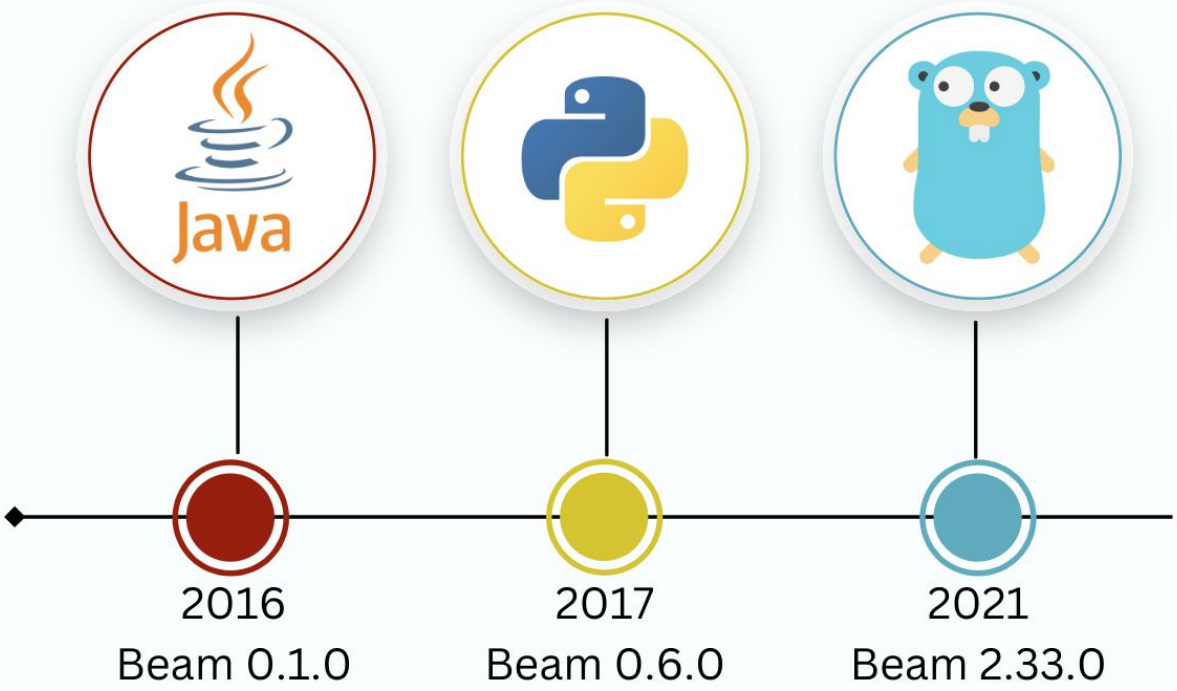
Problem statement



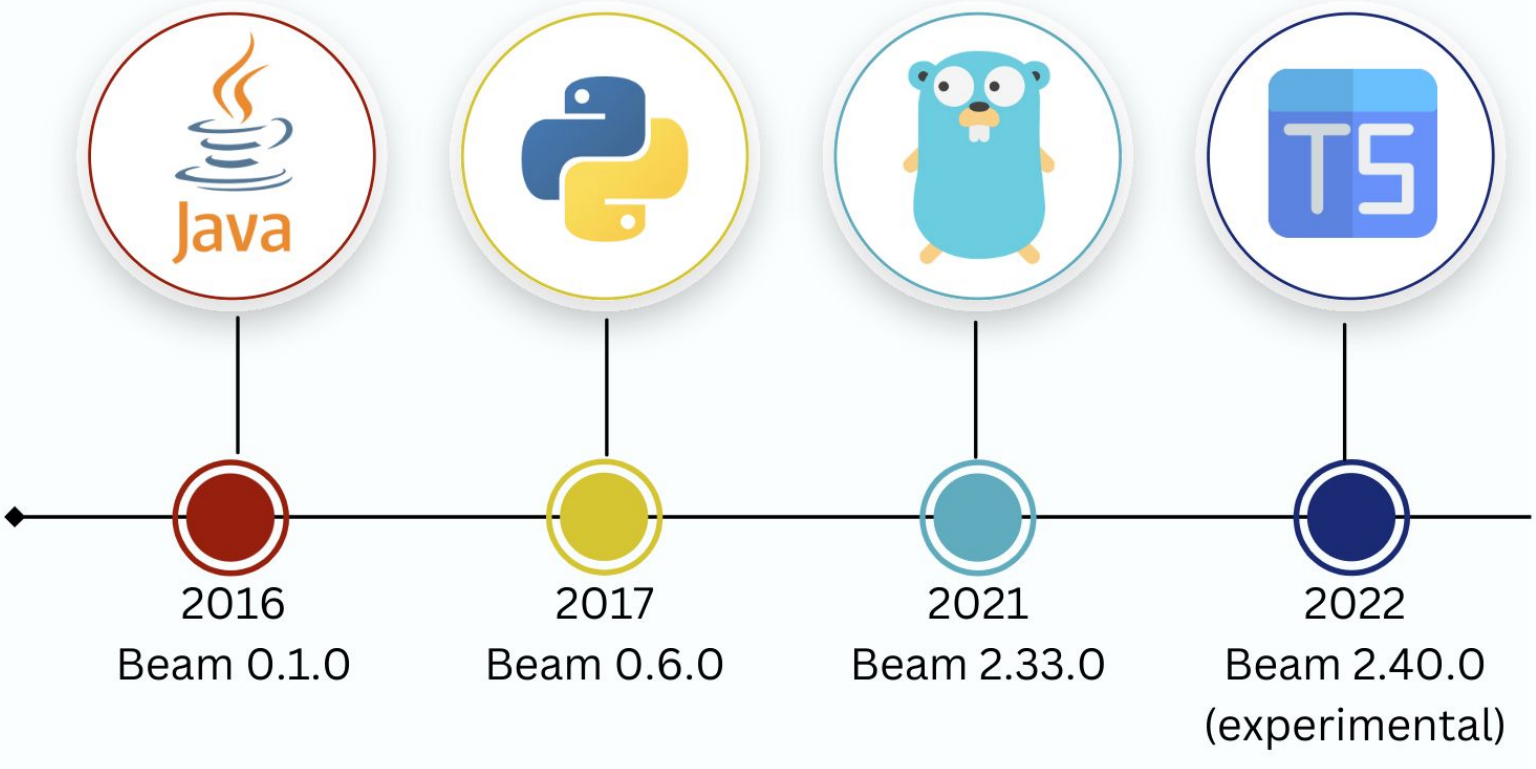
Problem statement



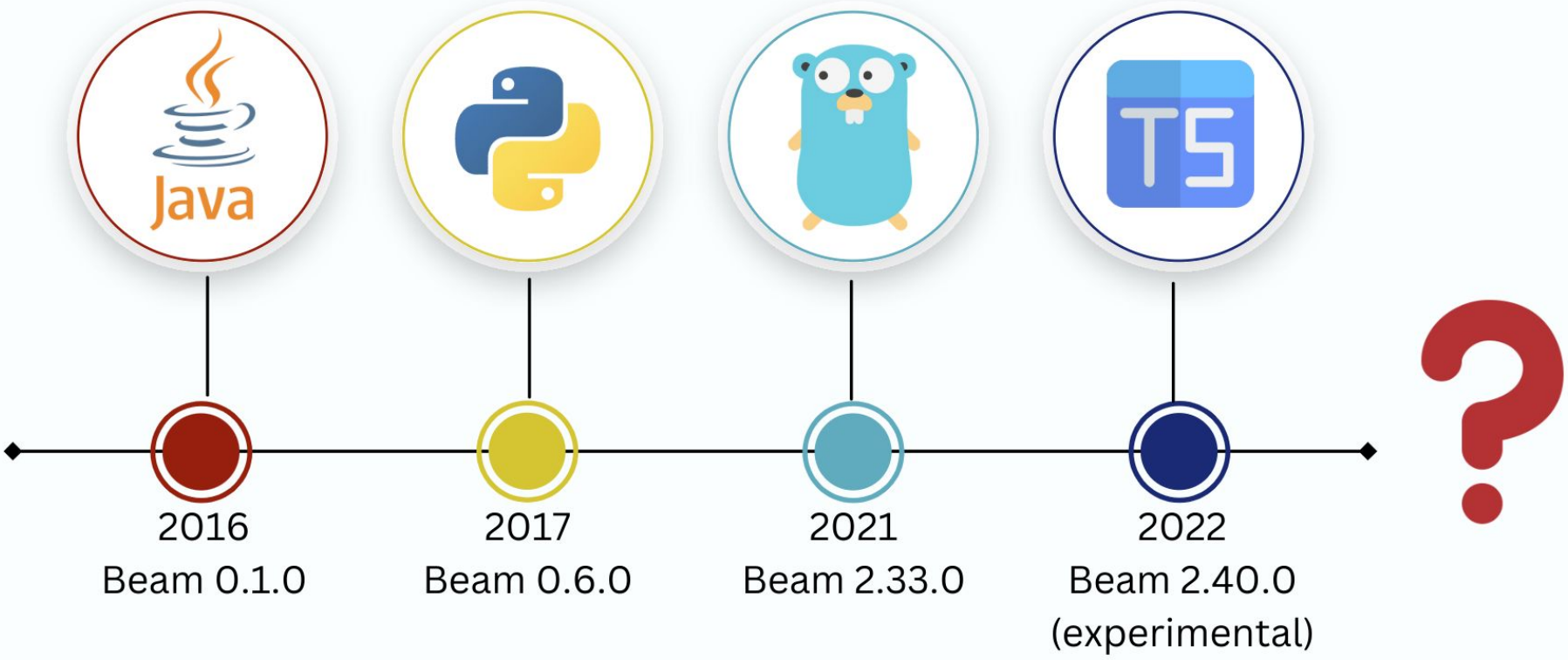
Problem statement



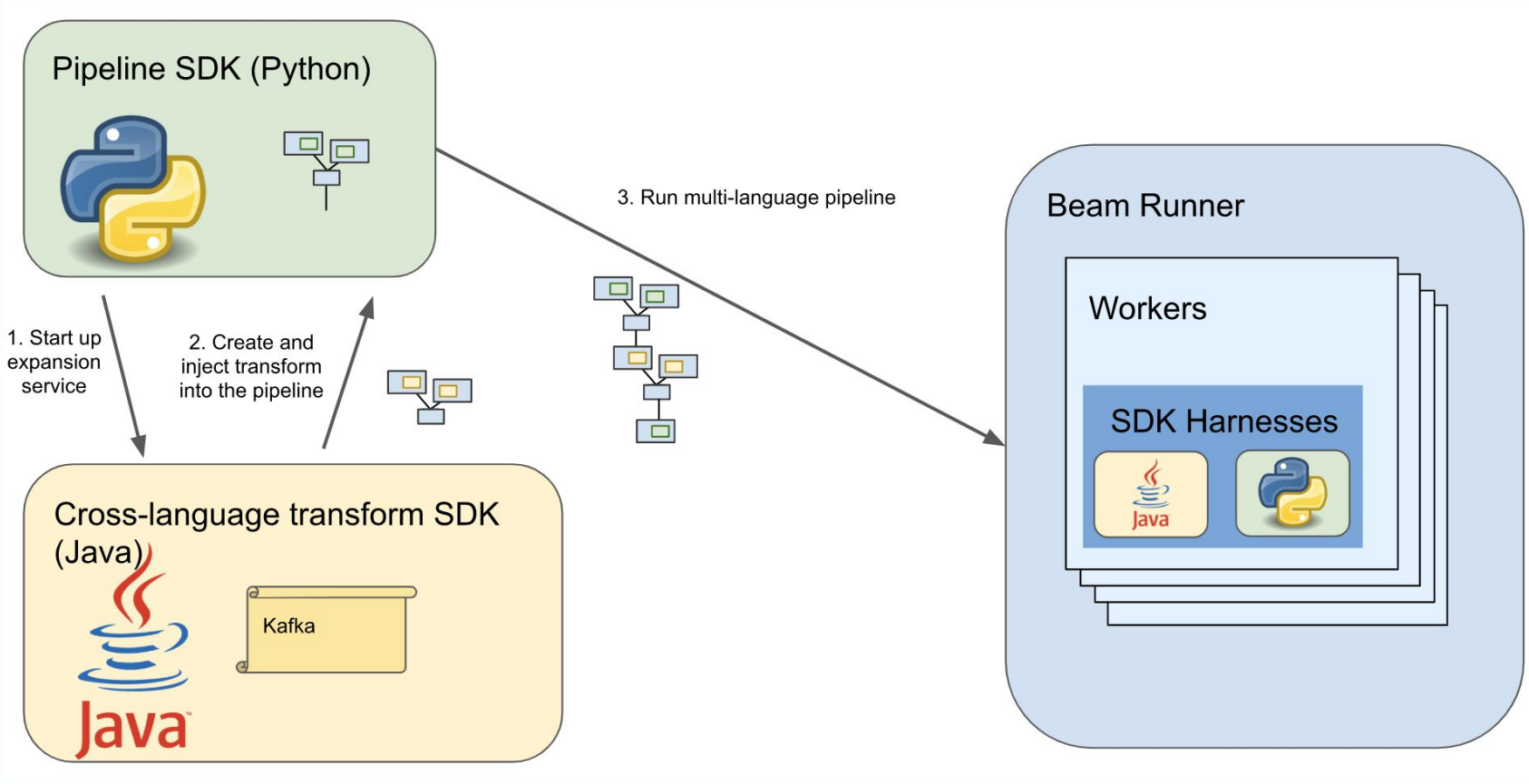
Problem statement



Problem statement



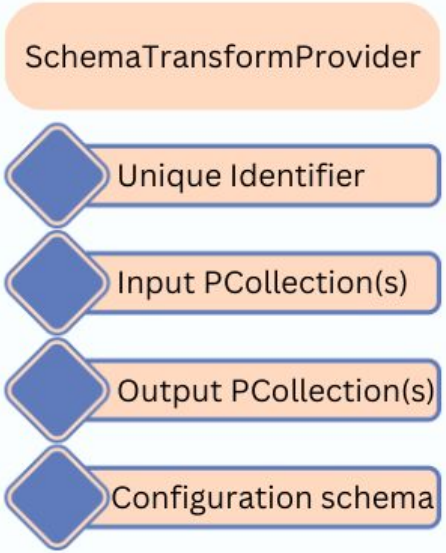
Enter cross-language



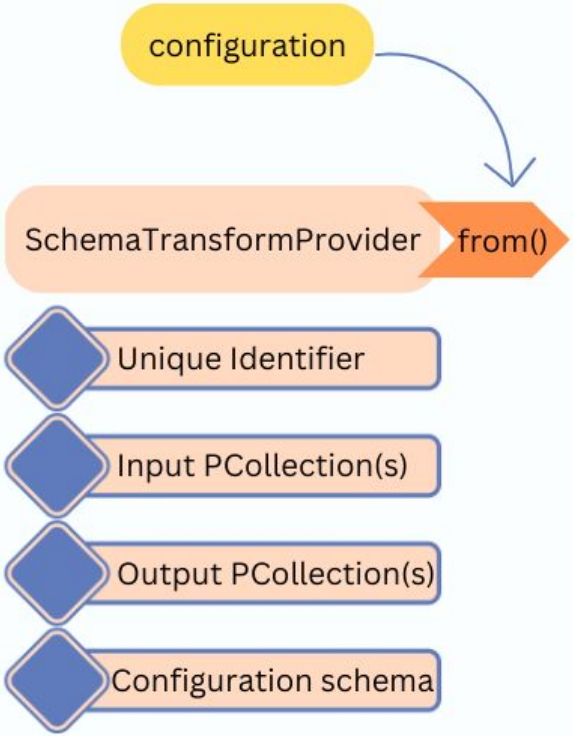
What is a SchemaTransform?

Ahmed Abualsaud

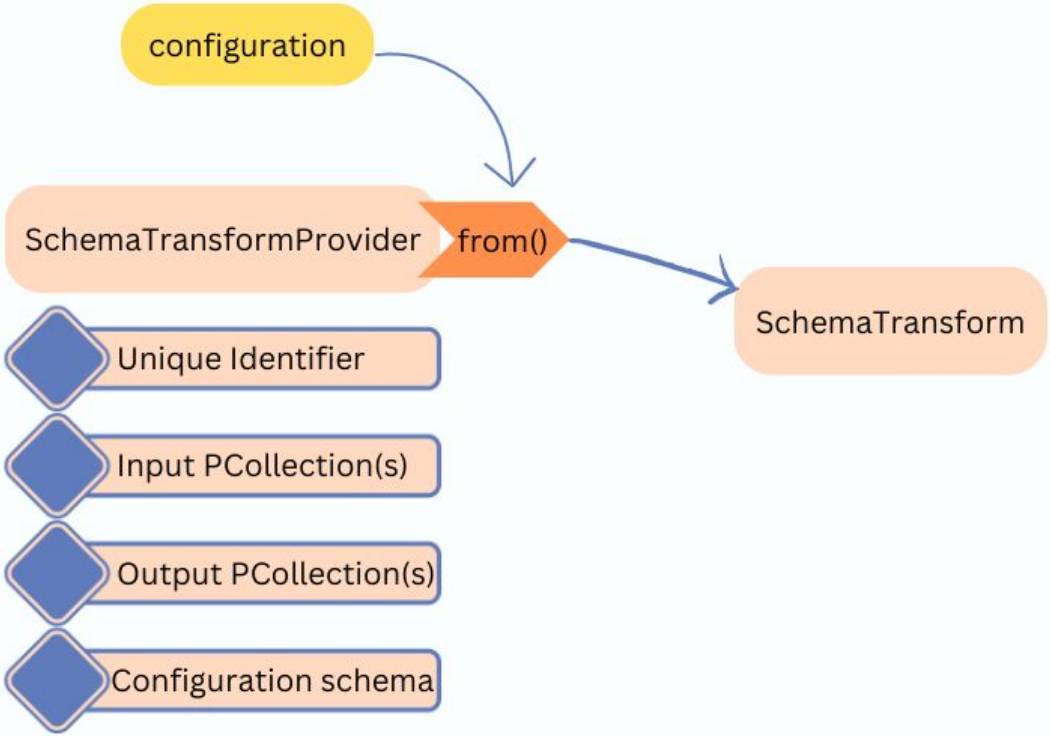
What is a SchemaTransform?



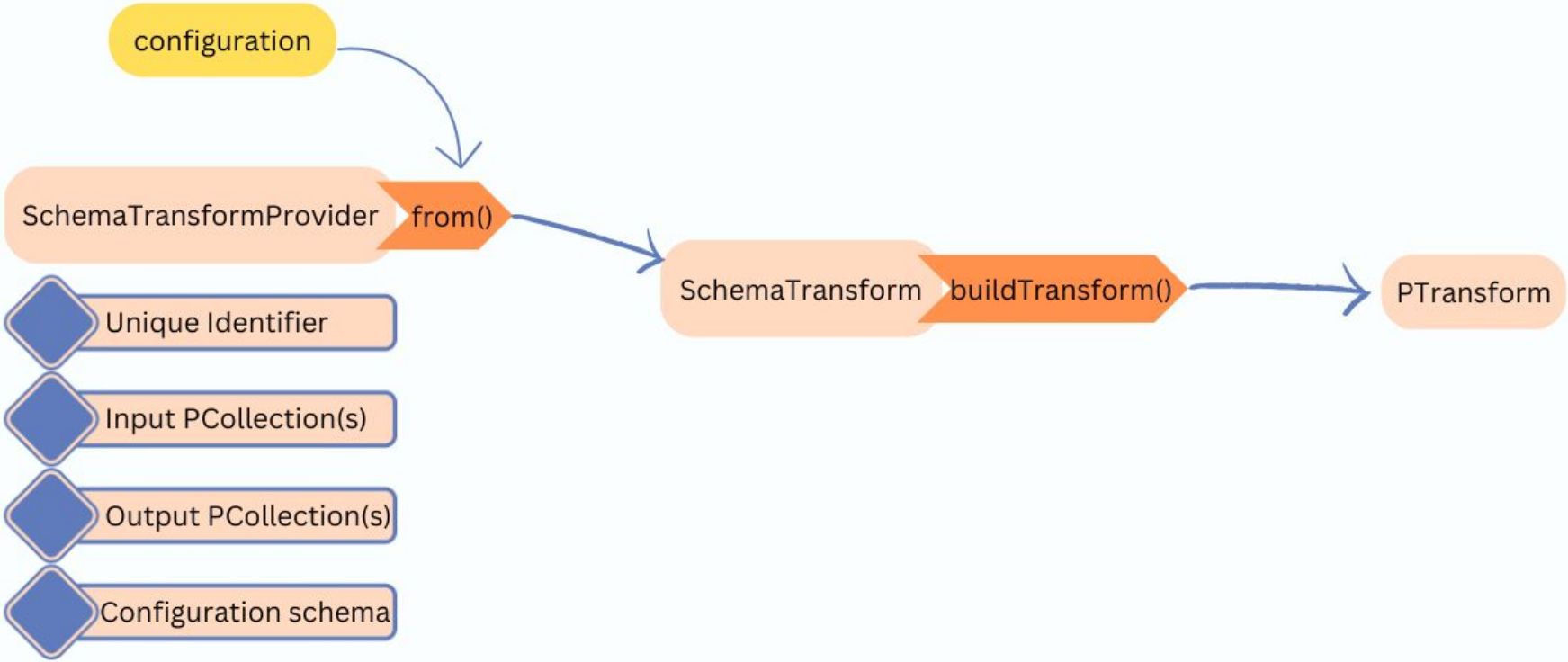
What is a SchemaTransform?



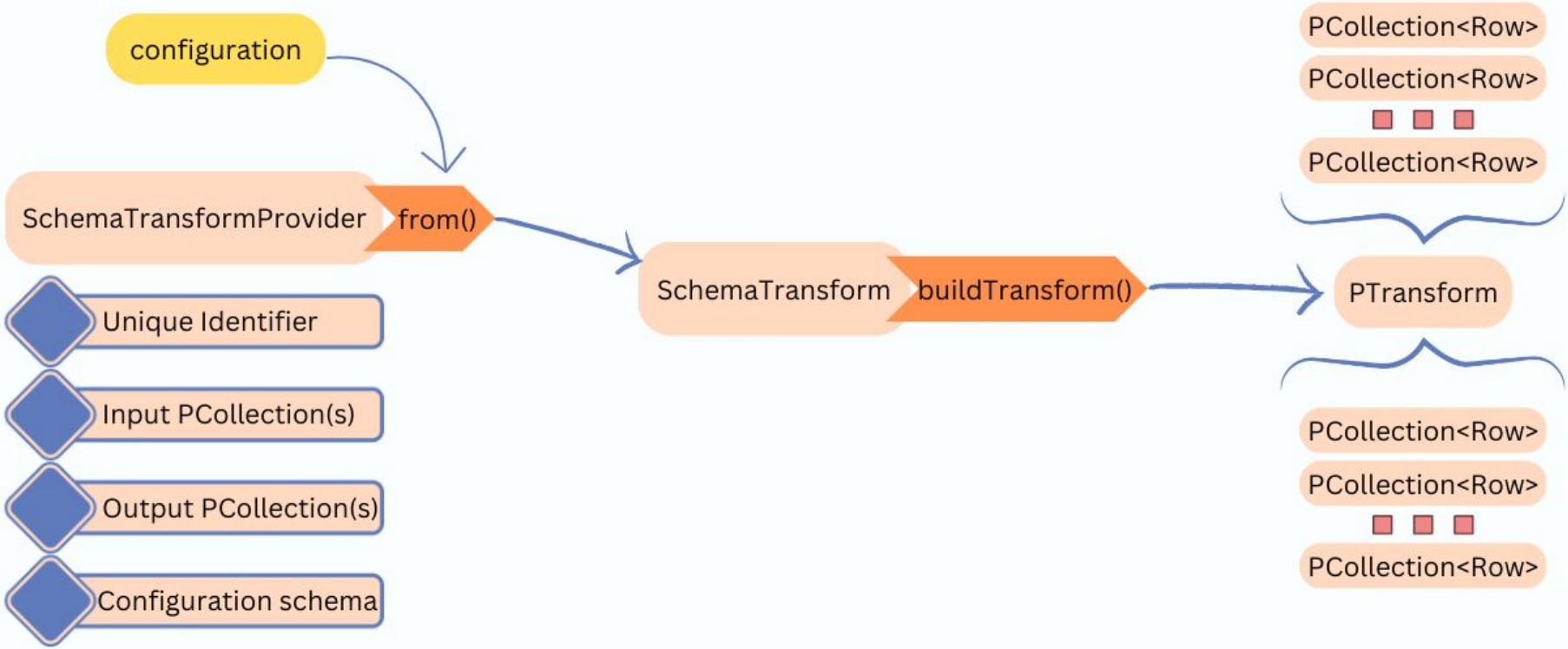
What is a SchemaTransform?



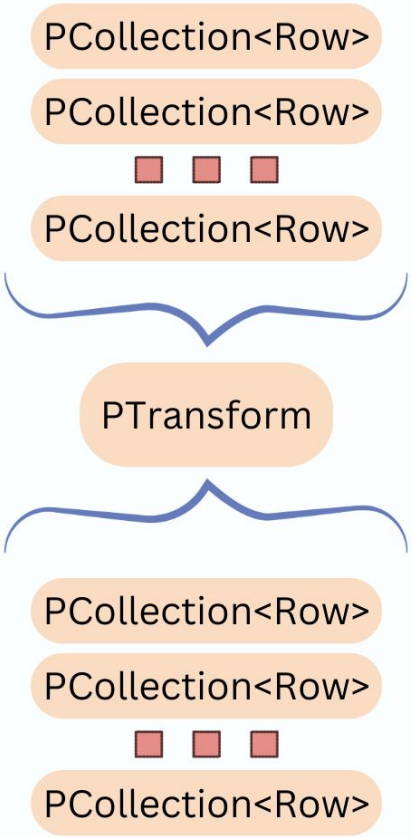
What is a SchemaTransform?



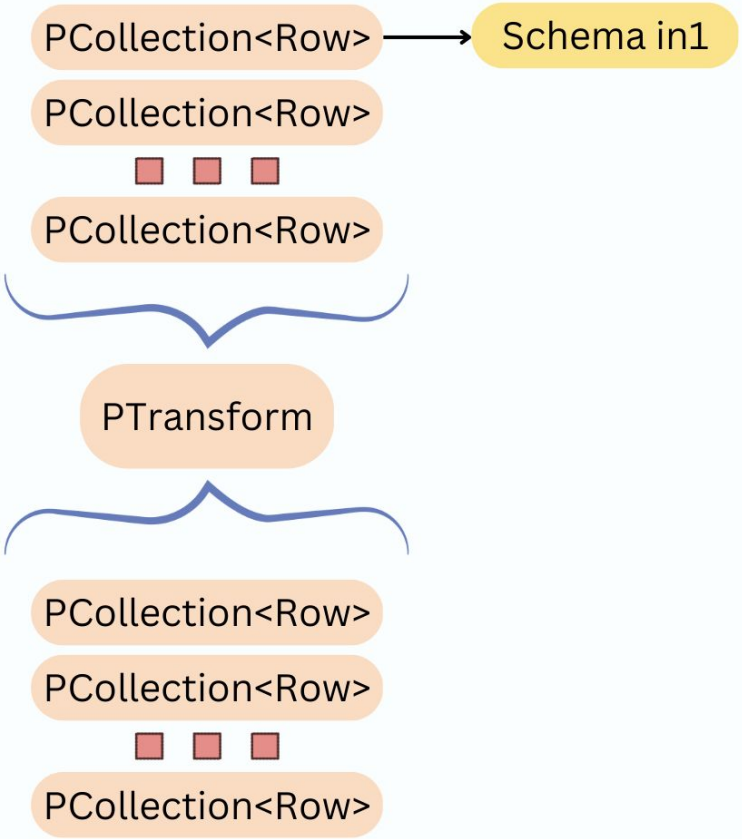
What is a SchemaTransform?



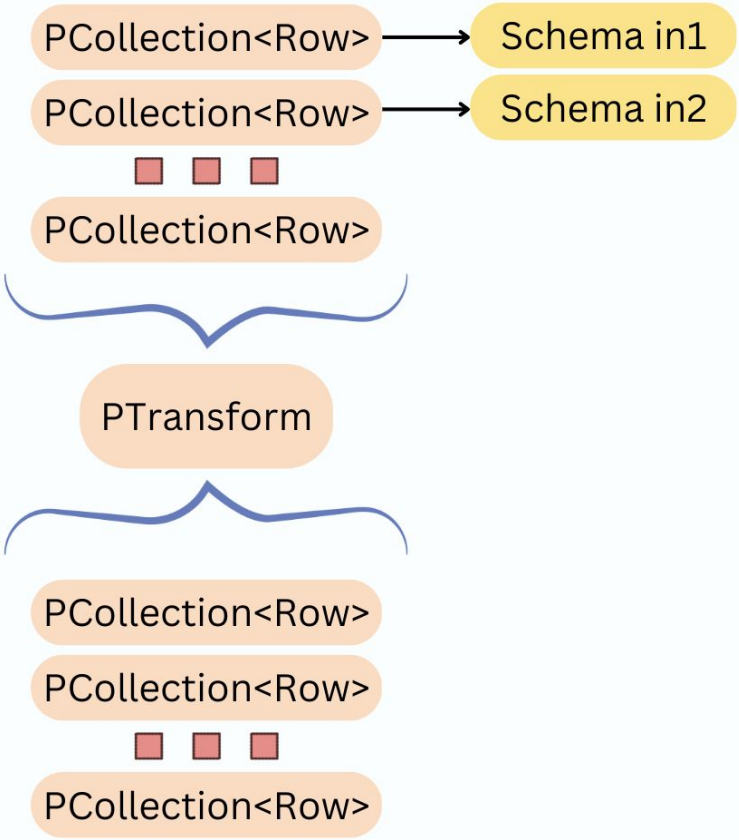
Why is it called a SchemaTransform?



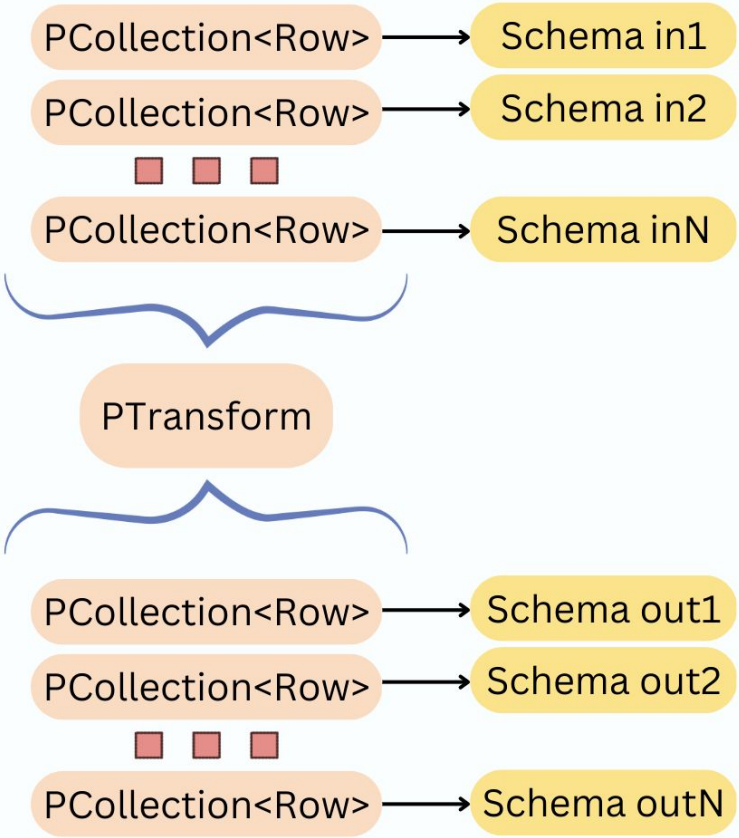
Why is it called a SchemaTransform?



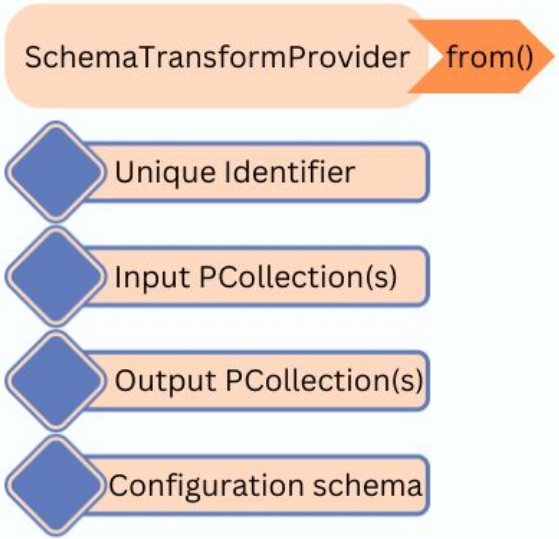
Why is it called a SchemaTransform?



Why is it called a SchemaTransform?



Creating a SchemaTransform




```java
public interface SchemaTransformProvider {
  /** Returns an id that uniquely represents this transform. */
  String identifier();

  /**
   * Returns the expected schema of the configuration object. Note this is distinct from the
   * schema of the transform itself.
   */
  Schema configurationSchema();

  /**
   * Produce a SchemaTransform some transform-specific configuration object. Can throw a
   * InvalidConfigurationException or a InvalidSchemaException.
   */
  SchemaTransform from(Row configuration);

  /** Returns the input collection names of this transform. */
  List<String> inputCollectionNames();

  /** Returns the output collection names of this transform. */
  List<String> outputCollectionNames();

  /**
   * List the dependencies needed for this transform. Jars from classpath are used by default when
   * Optional.empty() is returned.
   */
  default Optional<List<String>> dependencies(Row configuration, PipelineOptions options) {
    return Optional.empty();
  }
}
```

Creating a SchemaTransform

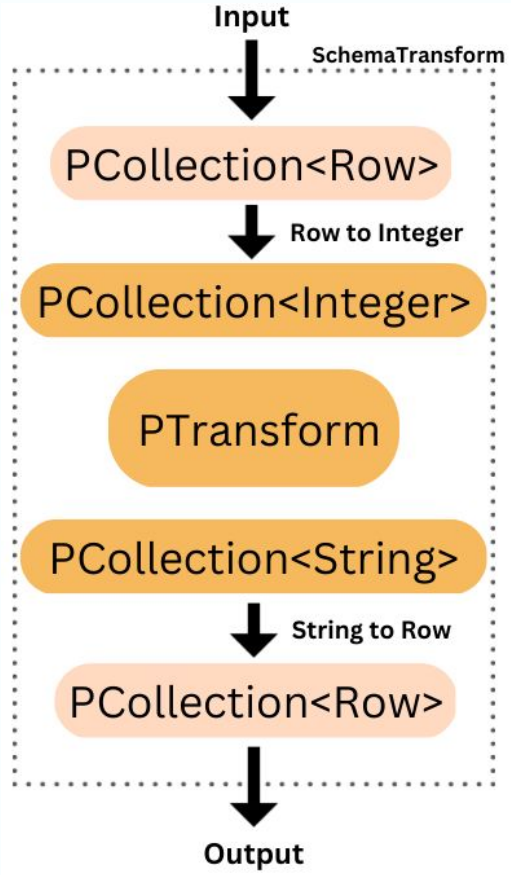
```
public interface SchemaTransform {
  PTransform<PCollectionRowTuple, PCollectionRowTuple> buildTransform();
}
```

Creating a SchemaTransform

```
public interface SchemaTransformProvider {

  SchemaTransform from(Row configuration);
```

```
public interface SchemaTransform {
  PTransform<PCollectionRowTuple, PCollectionRowTuple> buildTransform();
}
```



Creating a SchemaTransform

```
for (org.apache.beam.sdk.schemas.transforms.SchemaTransformProvider schemaTransformProvider :
    ServiceLoader.load(
        org.apache.beam.sdk.schemas.transforms.SchemaTransformProvider.class)) {
```

```
import com.google.auto.service.AutoService;

@AutoService(SchemaTransformProvider.class)
public class MySchemaTransformProvider implements SchemaTransformProvider {
```

Running a Java expansion service

Jar containing Beam's ExpansionService

Jar containing our SchemaTransform(s)

```
java -cp beam-sdks-java-io-expansion-service-2.47.0.jar:my-project.jar
org.apache.beam.sdk.expansion.service.ExpansionService 12345
```

Port the expansion service will run on

```
Registered SchemaTransformProviders:
        my_java_transform
        beam:schematransform:org.apache.beam:kafka_read:v1
        beam:schematransform:org.apache.beam:kafka_write:v1
```

Using the SchemaTransform in a Python Pipeline

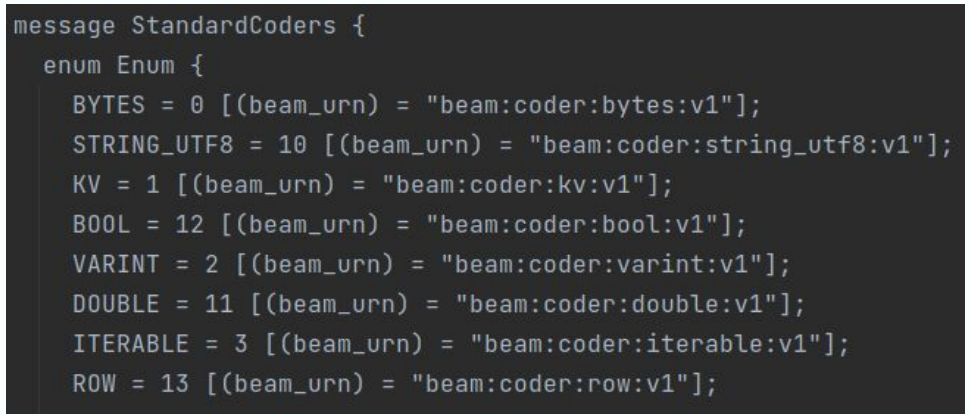
```
with beam.Pipeline() as p:
  _ = (
      p
      | beam.Create([
          beam.Row(text="Hello"),
          beam.Row(text="World!")])
      | SchemaAwareExternalTransform(
          identifier="my_java_transform",       # SchemaTransform URN
          expansion_service="localhost:12345",  # expansion service address
          arg1="string",       # configuration parameter
          arg2=1,              # configuration parameter
          arg3=False,          # configuration parameter
          rearrange_based_on_discovery=True)    # set to true if params may
                                                # be out of order
      | beam.ParDo(ProcessRows()))
```

Demo!

github.com/ahmedabu98/xlang-word-count

Current limitations/unknowns

- We are restricted to using Beam Rows
  - E.g. sending Strings is okay for cross-language interface
- Not all logical types are supported yet.
  - e.g. Java DateTime → Python Timestamp. No Python DateTime equivalent
- It still takes some time/effort to create a SchemaTransform
- We don't have performance metrics for most SchemaTransforms



```
message StandardCoders {
  enum Enum {
    BYTES = 0 [(beam_urn) = "beam:coder:bytes:v1"];
    STRING_UTF8 = 10 [(beam_urn) = "beam:coder:string_utf8:v1"];
    KV = 1 [(beam_urn) = "beam:coder:kv:v1"];
    BOOL = 12 [(beam_urn) = "beam:coder:bool:v1"];
    VARINT = 2 [(beam_urn) = "beam:coder:varint:v1"];
    DOUBLE = 11 [(beam_urn) = "beam:coder:double:v1"];
    ITERABLE = 3 [(beam_urn) = "beam:coder:iterable:v1"];
    ROW = 13 [(beam_urn) = "beam:coder:row:v1"];
```

Ahmed Abualsaud

QUESTIONS?

linkedin.com/in/ahmedabu98
github.com/ahmedbu98


BEAM
SUMMIT