Dataflow Templates

Akvelon

https://akvelon.com/

Agenda

Dataflow Templates

Use Case - Data Protection Using Tokenization

Solution Architecture & Technical Highlights

Demo

Contributing to open source DataflowTemplates

AKVELON
We make BIG software happen

Google Cloud
Partner

Beam Pipeline

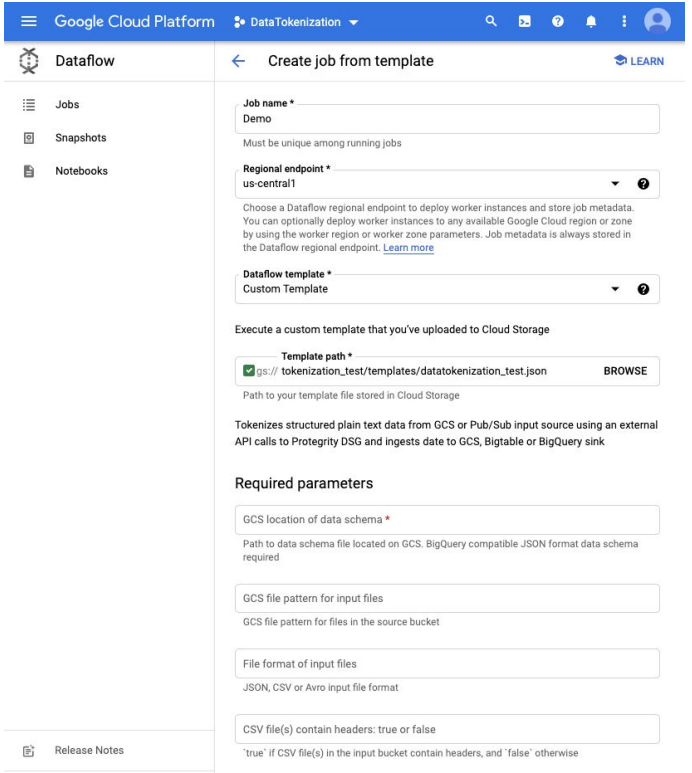
Dataflow Templates

Template - a way to package and stage pipelines

Template types: Classic and Flex

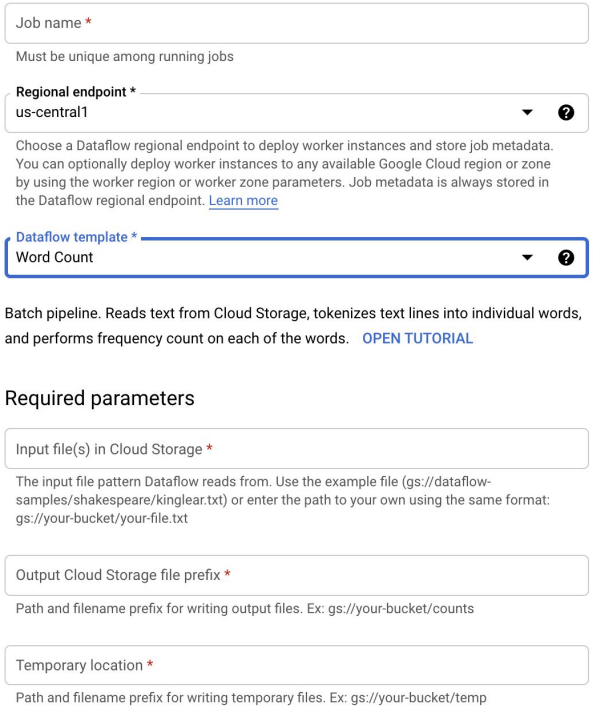
Flex template benefits

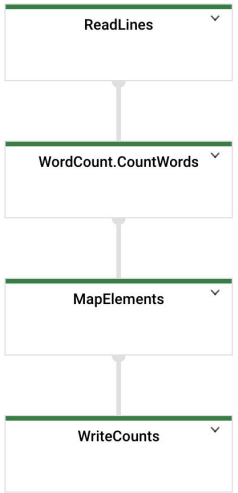
- Dynamic DAG
- Eliminated need for ValueProvider
- Expanded templates flexibility



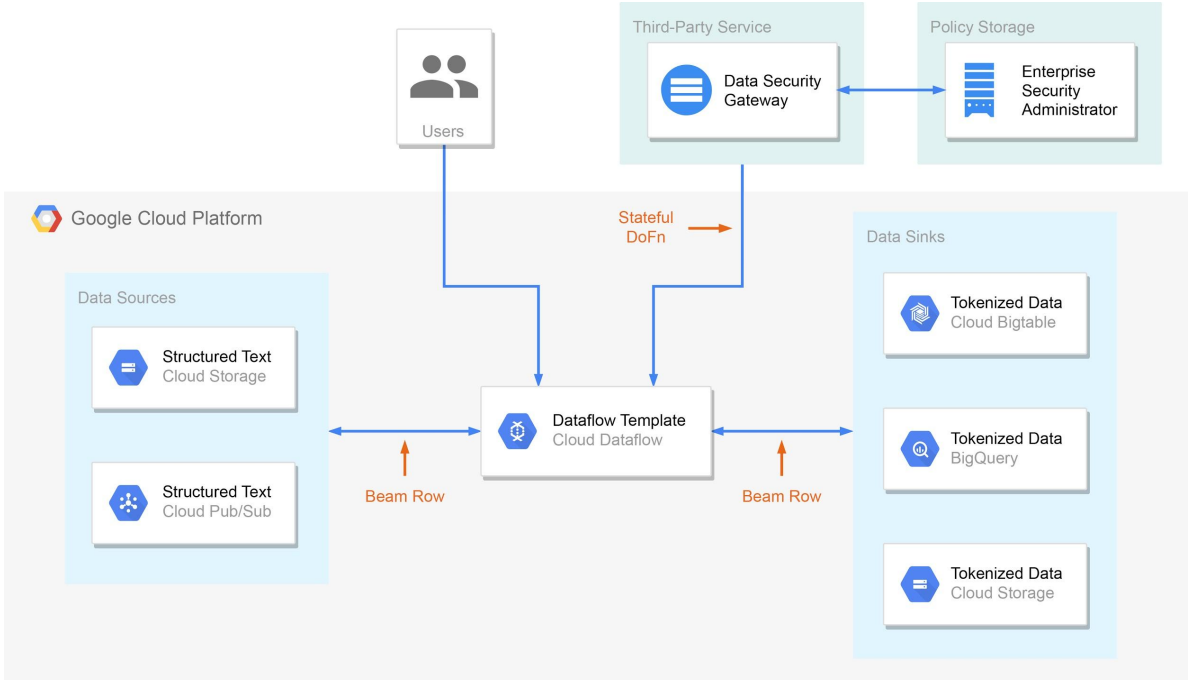
Create Template

1. Implement pipeline
2. Create metadata
3. Build template

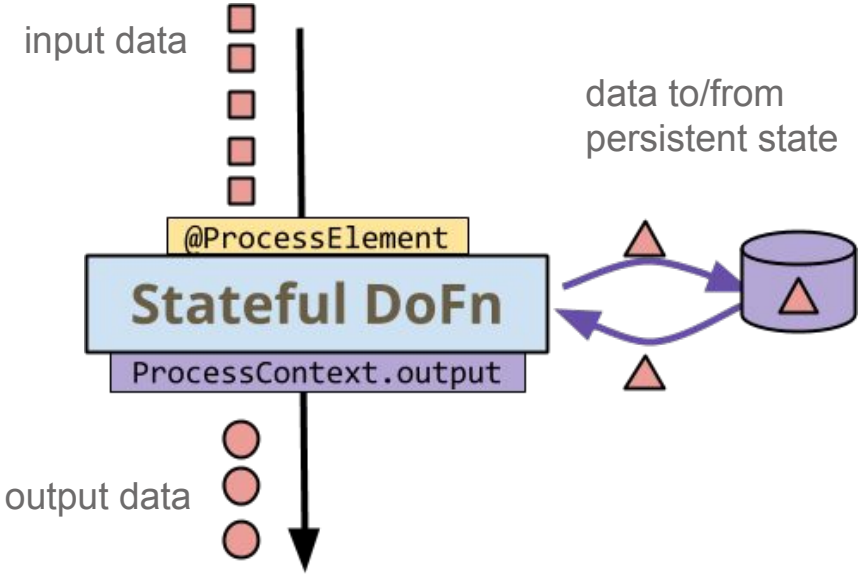




Data Protection Using Tokenization

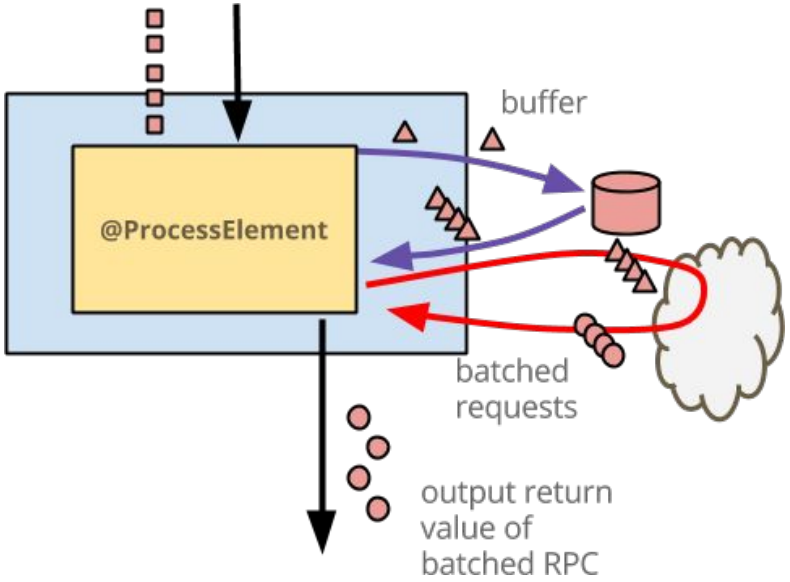


Stateful Processing



More info https://beam.apache.org/blog/stateful-processing/

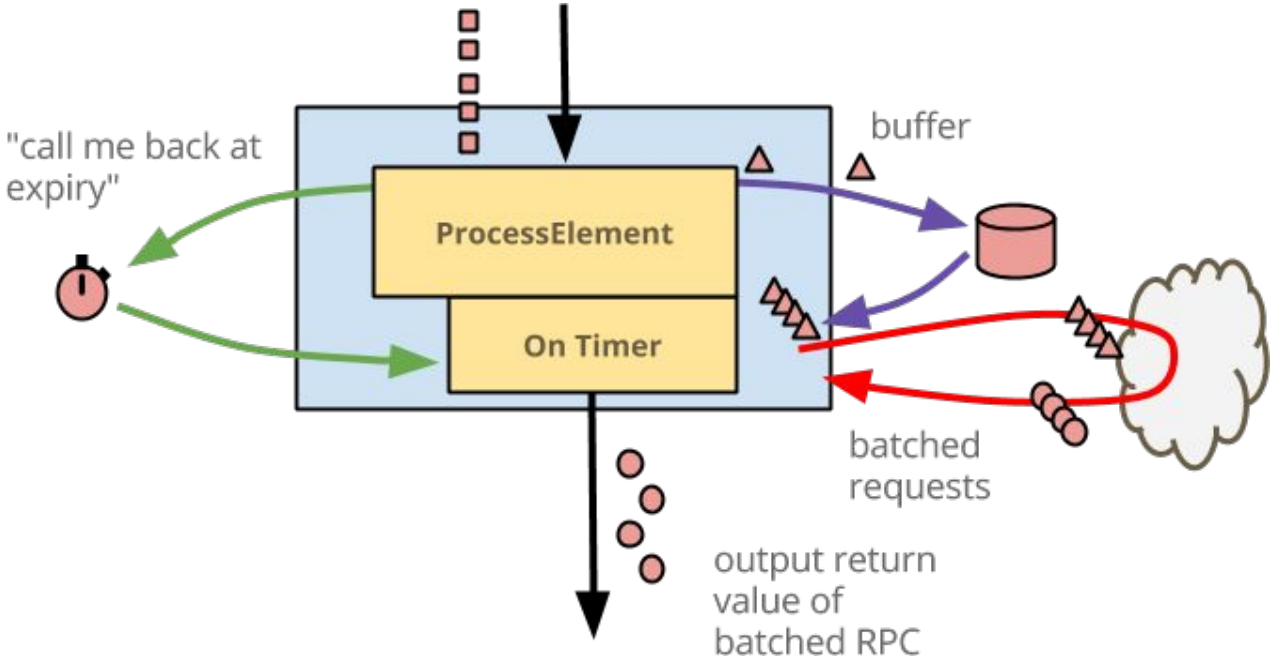
Stateful DoFn




```
new DoFn<Event, EnrichedEvent>() {

  private static final int MAX_BUFFER_SIZE = 500;

  @StateId("buffer")
  private final StateSpec<BagState<Event>> bufferedEvents = StateSpecs.bag();

  @StateId("count")
  private final StateSpec<ValueState<Integer>> countState = StateSpecs.value();

  @ProcessElement
  public void process(
      ProcessContext context,
      @StateId("buffer") BagState<Event> bufferState,
      @StateId("count") ValueState<Integer> countState) {

    int count = firstNonNull(countState.read(), 0);
    count = count + 1;
    countState.write(count);
    bufferState.add(context.element());

    if (count >= MAX_BUFFER_SIZE) {
      for (EnrichedEvent enrichedEvent : enrichEvents(bufferState.read())) {
        context.output(enrichedEvent);
      }
      bufferState.clear();
      countState.clear();
    }
  }
  //... TBD ...
```

More info https://beam.apache.org/blog/timely-processing/

Timely Stateful Processing



More info https://beam.apache.org/blog/timely-processing/

Stateful DoFn with Timer - Implementation


```
new DoFn<Event, EnrichedEvent>() {
  //…

  @TimerId("expiry")
  private final TimerSpec expirySpec = TimerSpecs.timer(TimeDomain.EVENT_TIME);

  @ProcessElement
  public void process(
      ProcessContext context,
      BoundedWindow window,
      @StateId("buffer") BagState<Event> bufferState,
      @StateId("count") ValueState<Integer> countState,
      @TimerId("expiry") Timer expiryTimer) {

    expiryTimer.set(window.maxTimestamp().plus(allowedLateness));

  //… same logic as above …
  }

  @OnTimer("expiry")
  public void onExpiry(
      OnTimerContext context,
      @StateId("buffer") BagState<Event> bufferState) {
    if (!bufferState.isEmpty().read()) {
      for (EnrichedEvent enrichedEvent : enrichEvents(bufferState.read())) {
        context.output(enrichedEvent);
      }
      bufferState.clear();
    }
  }
};
```

GroupIntoBatches

- Groups your data into batches
- Implemented using Stateful DoFn
- Has several optimizations
  - Prefetches data
  - Autosharding in Dataflow

```
import apache_beam as beam

with beam.Pipeline() as pipeline:
  batches_with_keys = (
      pipeline
      | 'Create produce' >> beam.Create([
          ('spring', '🍓'),
          ('spring', '🥕'),
          ('spring', '🍆'),
          ('spring', '🍅'),
          ('summer', '🥕'),
          ('summer', '🍅'),
          ('summer', '🌽'),
          ('fall', '🥕'),
          ('fall', '🍅'),
          ('winter', '🍆'),
      ])
      | 'Group into batches' >> beam.GroupIntoBatches(3)
      | beam.Map(print))
```

Output:

```
('spring', ['🍓', '🥕', '🍆'])
('summer', ['🥕', '🍅', '🌽'])
('spring', ['🍅'])
('fall', ['🥕', '🍅'])
('winter', ['🍆'])
```

More info
https://beam.apache.org/documentation/transforms/python/aggregation/groupintobatches

Processing with GroupIntoBatches


```
public PCollectionTuple expand(PCollection<KV<Integer, Row>> inputRows) {
  FailsafeElementCoder<Row, Row> coder =
      FailsafeElementCoder.of(RowCoder.of(schema()), RowCoder.of(schema()));

  Duration maxBuffering = Duration.millis(MAX_BUFFERING);
  PCollectionTuple pCollectionTuple =
      inputRows
          .apply(
              name: "GroupRowsIntoBatches",
              GroupIntoBatches.<Integer, Row>ofSize(batchSize())
                  .withMaxBufferingDuration(maxBuffering))
          .apply(
              name: "Tokenize",
              ParDo.of(new TokenizationFn(schema(), rpcURI(), failureTag()))
                  .withOutputTags(successTag(), TupleTagList.of(failureTag())));

  return PCollectionTuple.of(
          successTag(), pCollectionTuple.get(successTag()).setRowSchema(schema()))
      .and(failureTag(), pCollectionTuple.get(failureTag()).setCoder(coder));
}
```

Use Case Recommendations

Stateful DoFn

Customization of stateful processing

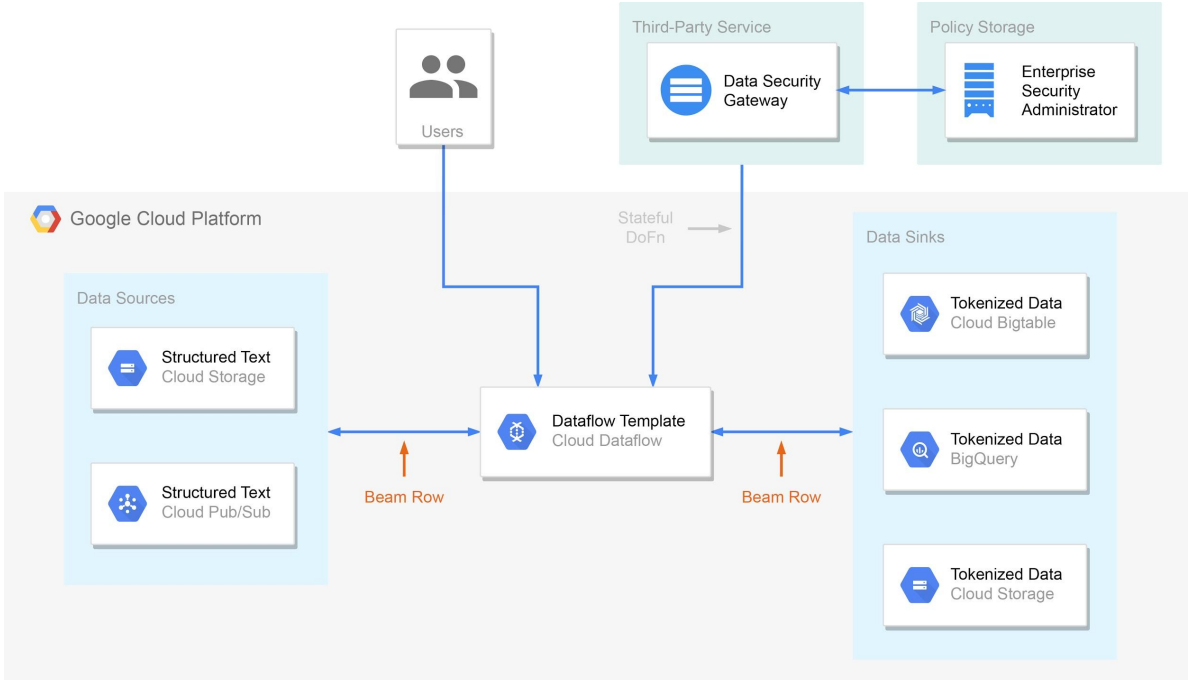
- Deduplication
- Arbitrary-but-consistent indexing

GroupIntoBatches

Optimized out-of-the-box transform

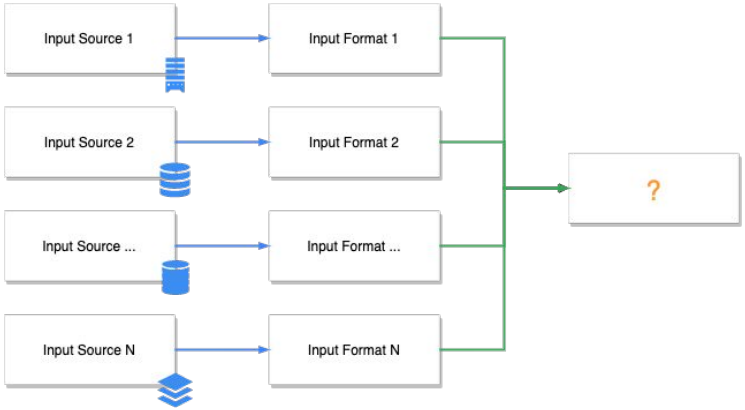
- External API call
- Data preparation for ML model

Data Protection Using Tokenization



How to Represent Data in Pipeline Code?

- Common abstractions for data representation
- Avoid writing transformation again and again
- Common interface to all IO transforms
- Easy and effective to serialize



Common Approaches

Text based formats

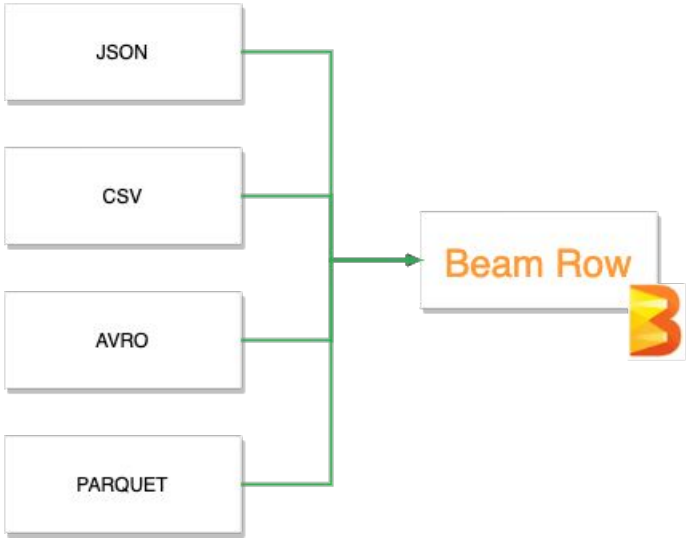
- JSON
- CSV
- XML
- YAML

Binary based formats

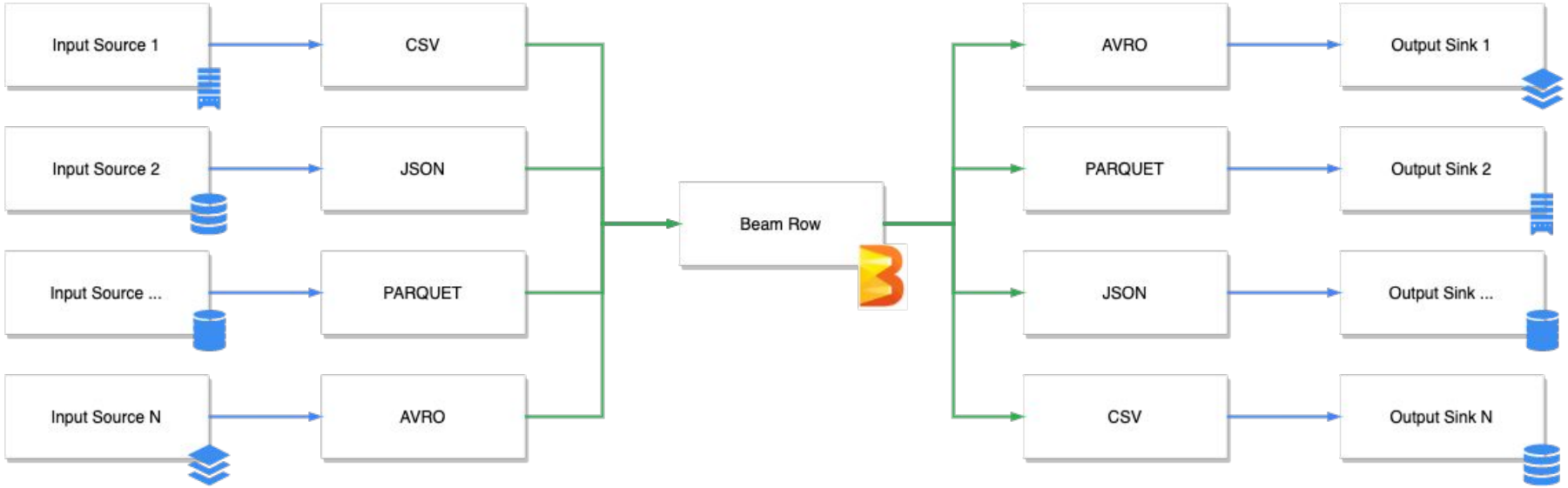
- Apache Avro
- Parquet
- Protobuf
- MessagePack

Beam Row

- A schema which supports primitives
- Ordering same with schema
- Quite effective serialization using RowCoder
- Generic for json specified schema
- Available for Beam SQL

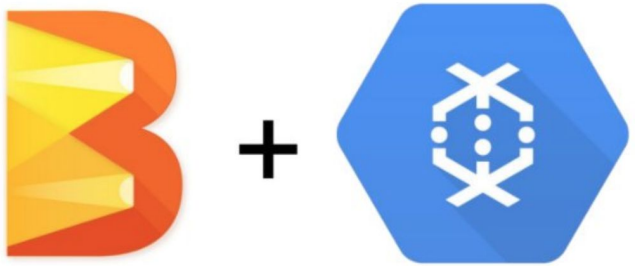


How We Use Beam Row?



Demo

- Template overview
- Add a new input source Parquet IO
- Implement Transform Parquet to Row
- Build and Run template



Demo

1 — 2 — 3 — 4

Template Overview

- ☑ Java Template structure
- ☑ Metadata file
- ☑ Schemas
- ☑ IO transforms

Add a New Input Source Parquet IO

- ☑ Read .parquet files

Implement Transform Parquet to Row

- ☑ Transform from Parquet to Beam Row
- ☑ Work with schemas

Build and Run Template

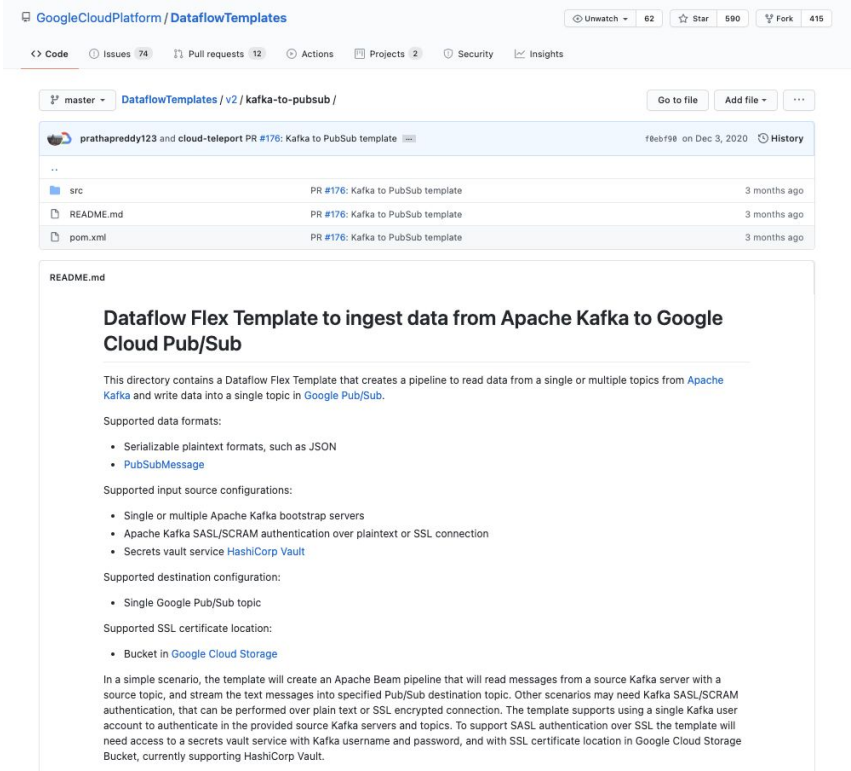
- ☑ Template building
- ☑ Ways to run template

Visit codelab at https://github.com/griscz/beam-college/tree/main/day3/dataflow-templates

Contributing to Dataflow Templates

https://github.com/GoogleCloudPlatform/DataflowTemplates

1. Fork the repository to develop your template
2. Follow style guides and best practices
3. Sign CLA
4. Create a PR
5. LGTM code review!



Summary

- Dataflow Flex templates package pipelines into containers
- Stateful processing in Apache Beam
- BeamRow provides flexible abstraction for data representation
- DataflowTemplates repository - Google and community contributed templates and utilities

Thank You!

https://akvelon.com/

@AkvelonInc