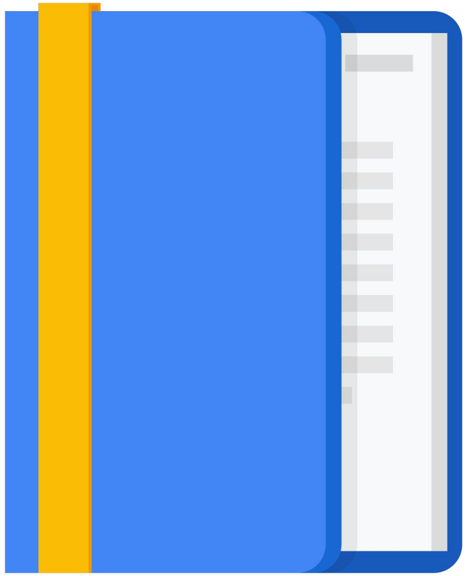
Beam/Dataflow

Agenda

Apache Beam

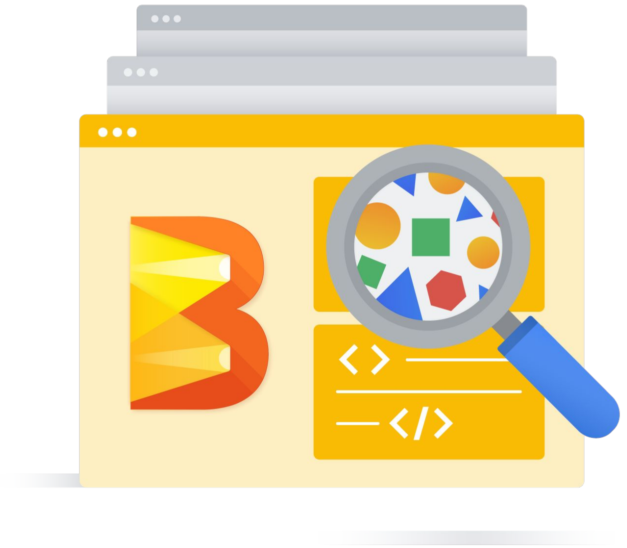
Pipeline for Creating TFRecords

Executing Pipelines with Dataflow

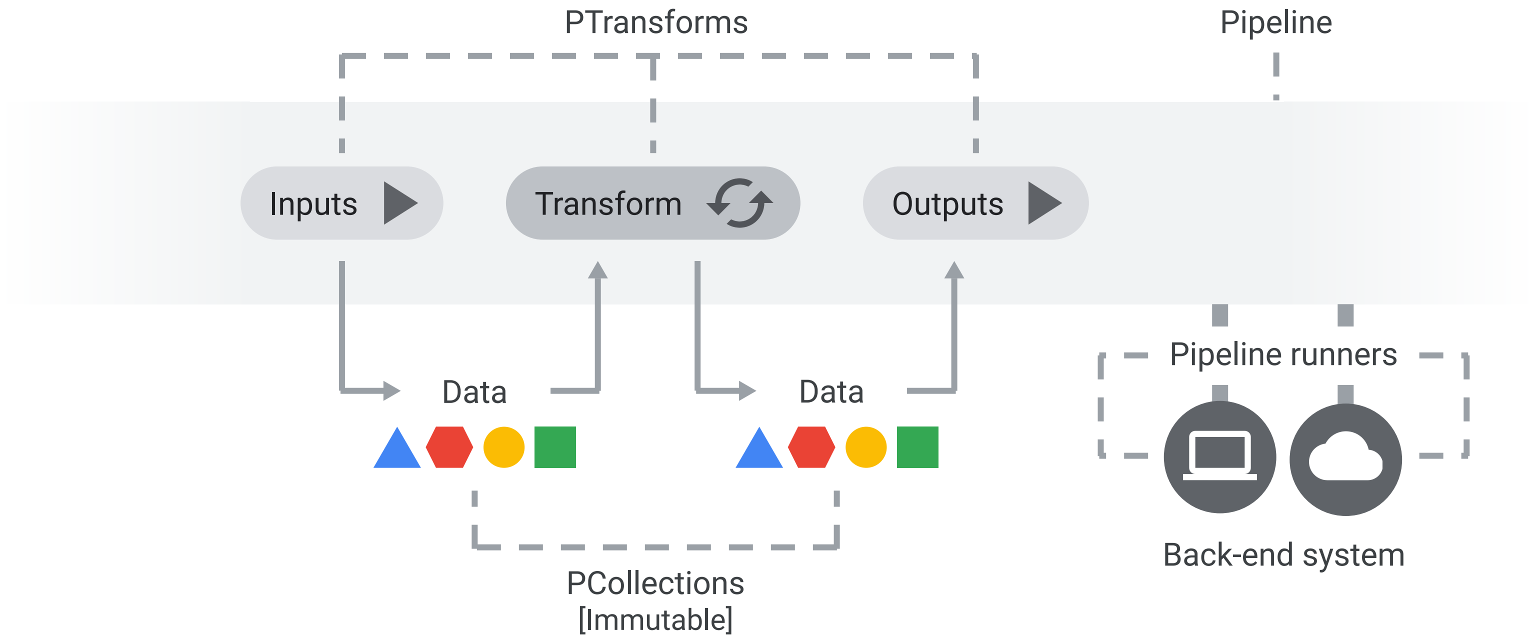


What is Apache Beam?

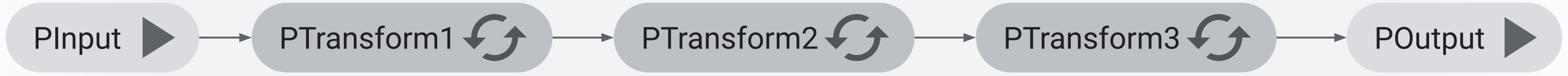
- Apache Beam is an open source unified programming model to define both batch and streaming data-processing pipelines.
- Beam SDKs are used to create the pipeline in the programming language of your choice.
- Pipelines can be run locally on your machine or on other backend services with their own features.
- A Runner is used to execute your pipeline on a backend of your choice.
- Dataflow is one of the runners available in Beam.



Apache Beam = Batch + stream

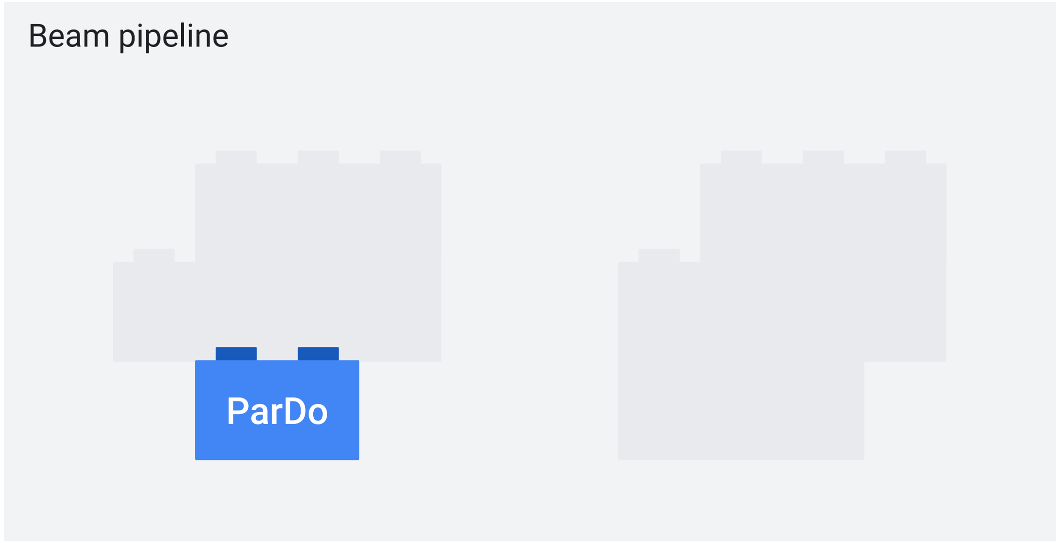


How to create a pipeline



```
with beam.Pipeline(options=PipelineOptions()) as p:

  p_input = p | …

  p_output = ( p_input
                | p_transform_1
                | p_transform_2
                | p_transform_3 )
```

Transforms



Transforms

ParDo

DoFn

```
class ComputeWordLengthFn(beam.DoFn):
  def process(self, element):
    return [len(element)]



word_lengths = words | beam.ParDo(ComputeWordLengthFn())
```

Friends of ParDo

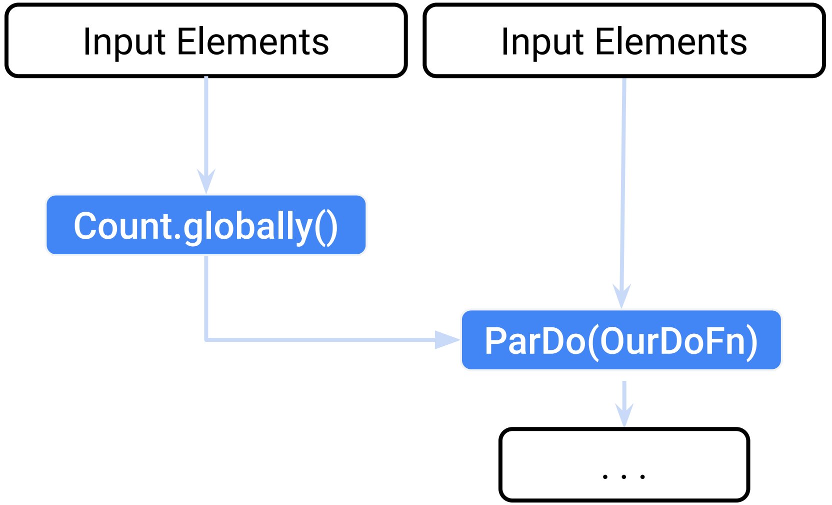
	Input	Output	Side inputs and side outputs
ParDo	**1**	**0, 1 or many**	✓
Filter	1	0 or 1	✗
MapElements	1	1	✗
FlatMapElements	1	0, 1 or Many	✗
WithKeys	value	(f(value), value)	✗
Keys	(key, value)	key	✗
Values	(key, value)	value	✗

Side Inputs

ParDos can receive extra inputs “on the side". Equivalent to a broadcast join in Data warehouses.

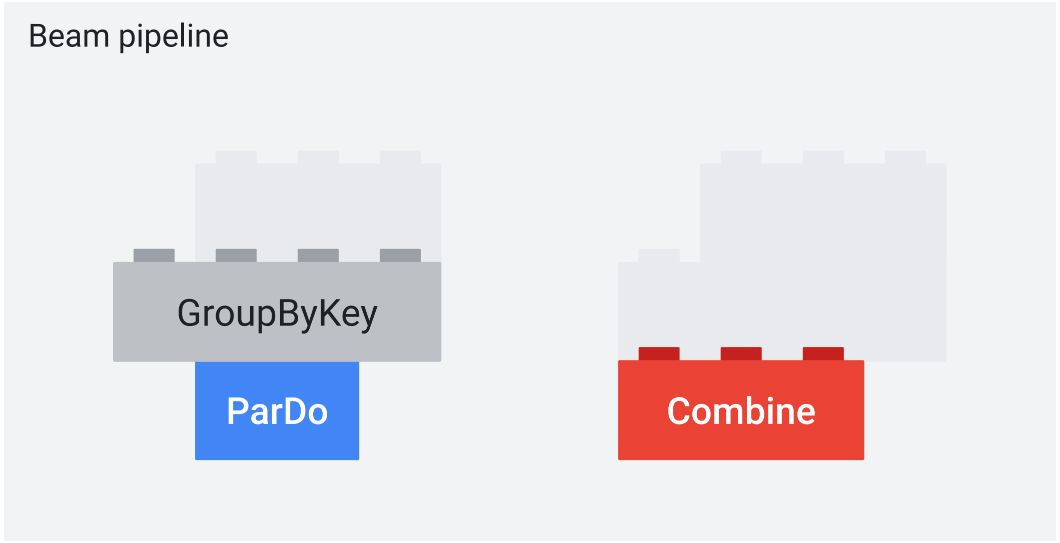
For example broadcast the count of elements to the processing of each element

Side inputs are computed (and accessed) per-window



```
beam.ParDo(OurDoFn(), side_input=pvalue.AsSingleton(count))
```

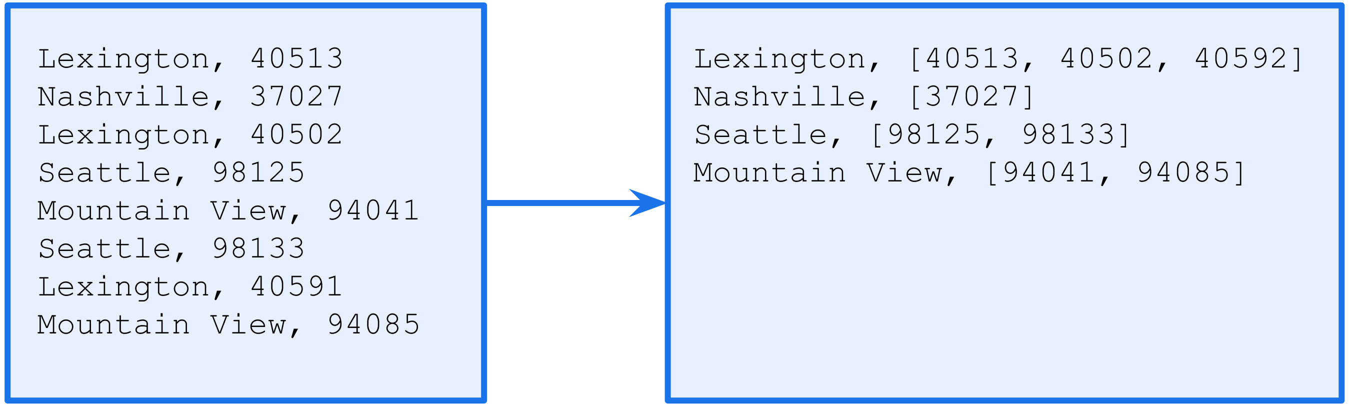
Transforms



Transforms

GroupByKey

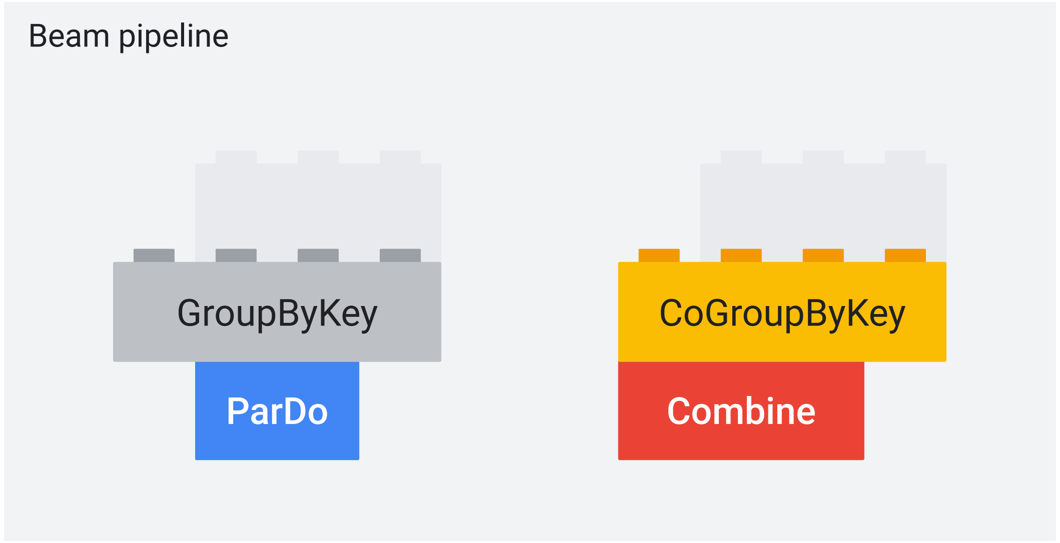
```
cityAndZipcodes = p | beam.Map(lambda fields : (fields[0], fields[1]))

grouped = cityAndZipCodes | beam.GroupByKey()
```



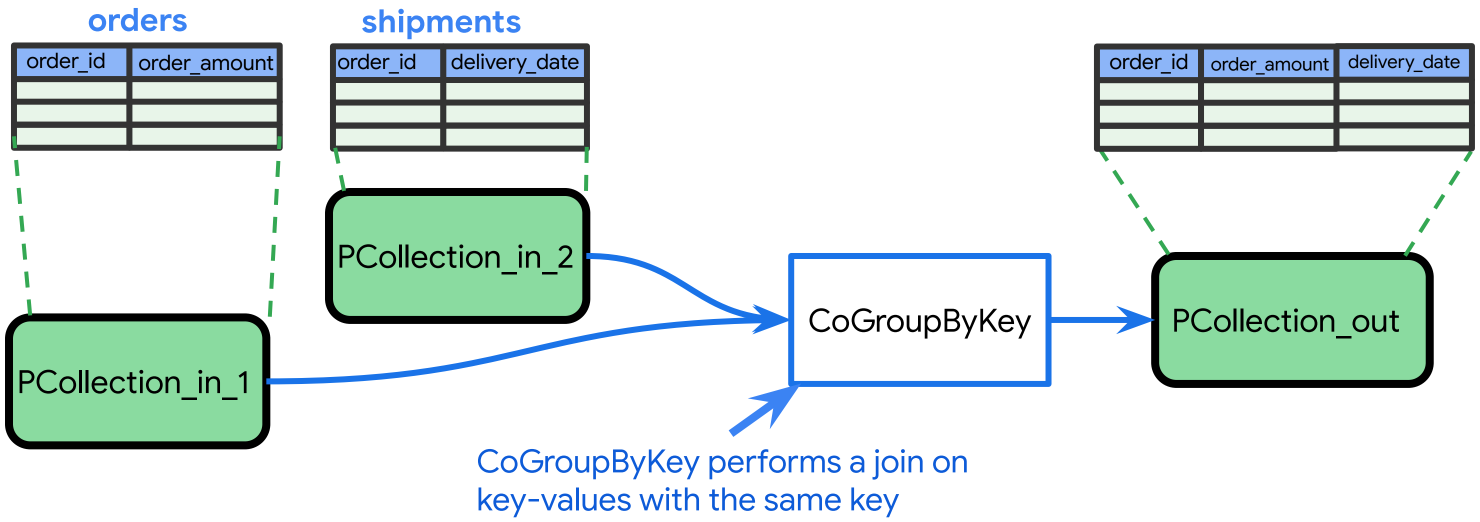
```
Lexington, 40513
Nashville, 37027
Lexington, 40502
Seattle, 98125
Mountain View, 94041
Seattle, 98133
Lexington, 40591
Mountain View, 94085
```

```
Lexington, [40513, 40502, 40592]
Nashville, [37027]
Seattle, [98125, 98133]
Mountain View, [94041, 94085]
```

Transforms

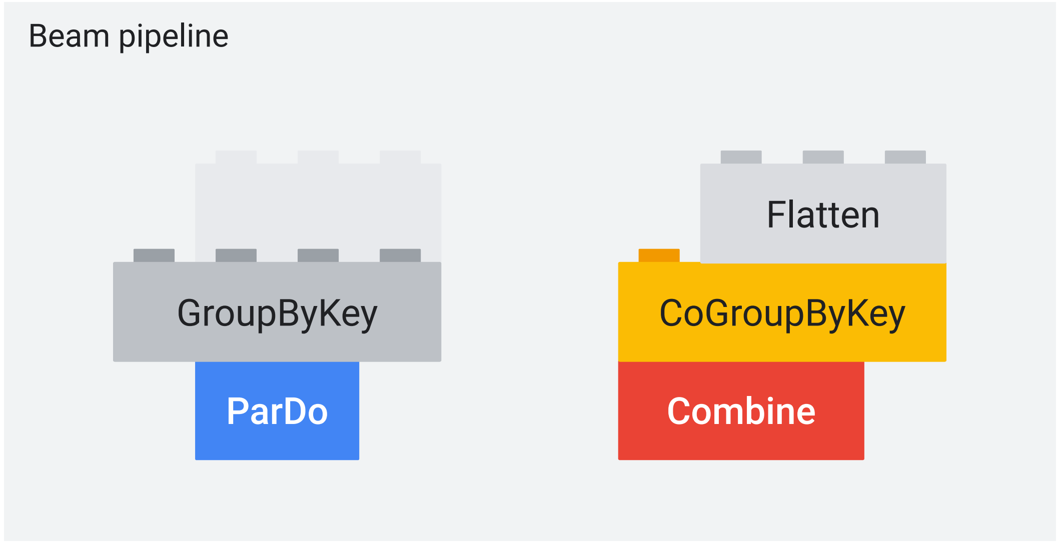


Transforms



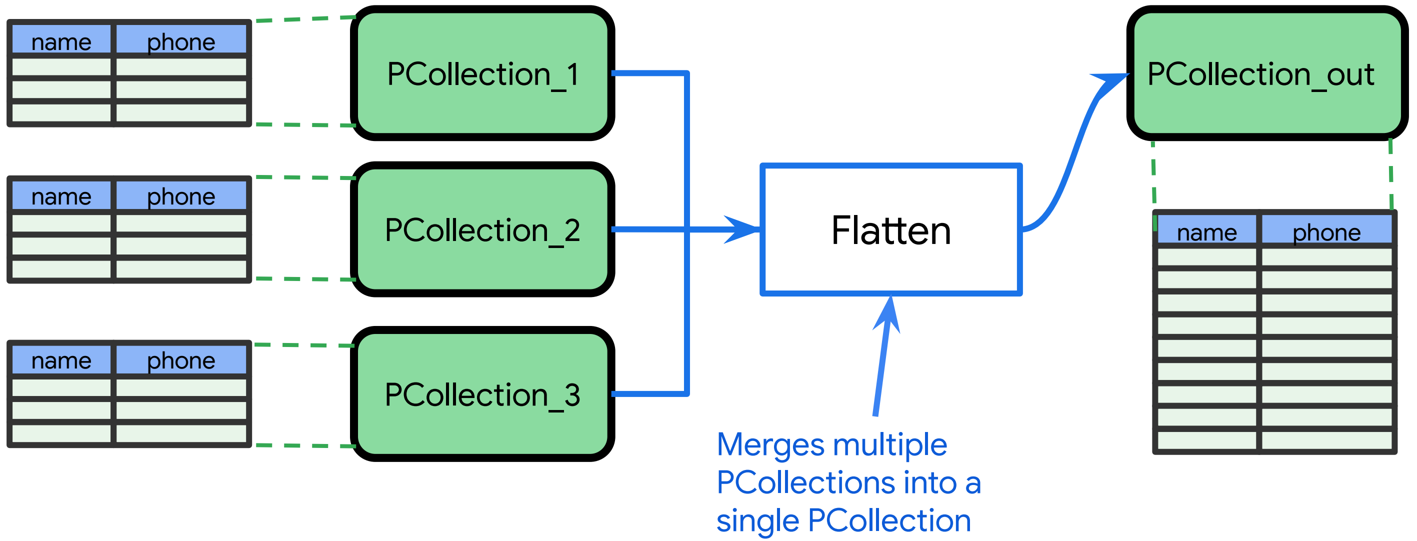
```
results = (
    {'orders': orders, 'shipments': shipments} | beam.CoGroupByKey()
)
```

Transforms


Beam pipeline
Flatten
GroupByKey
CoGroupByKey
ParDo
Combine

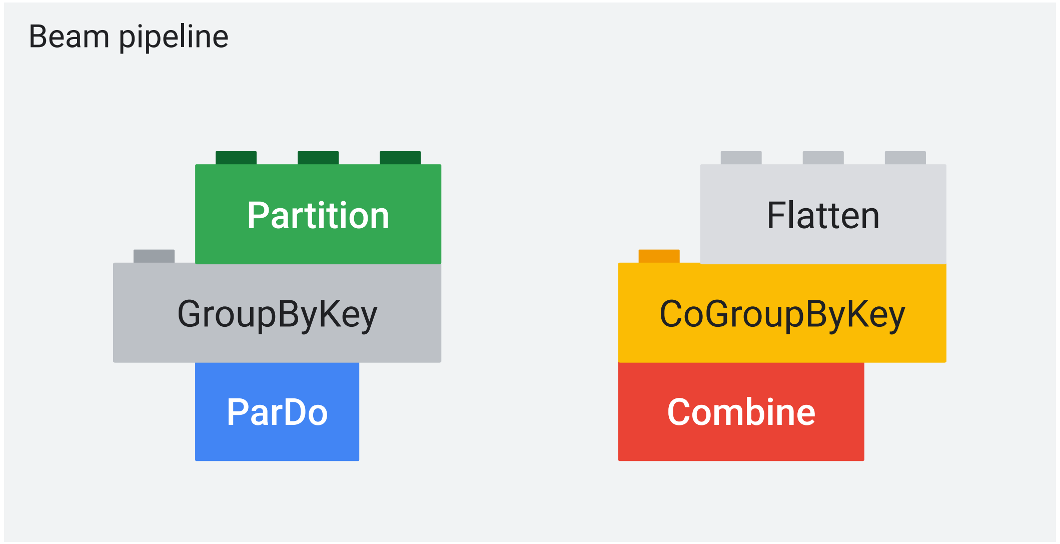
Transforms

Flatten



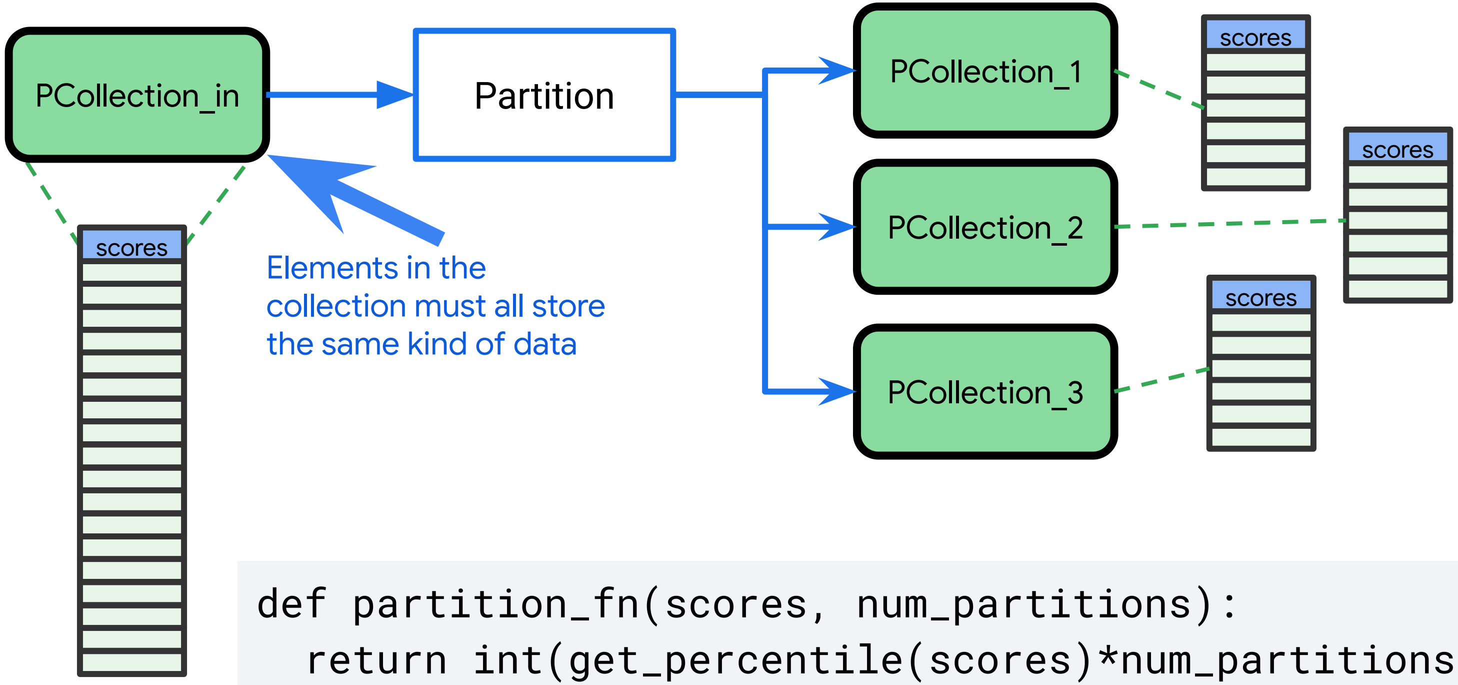
```
merged = ((pcoll1, pcoll2, pcoll3) | beam.Flatten())
```

Transforms


Beam pipeline
Partition
GroupByKey
ParDo
Flatten
CoGroupByKey
Combine

Transforms

Partition



```
def partition_fn(scores, num_partitions):
  return int(get_percentile(scores)*num_partitions/100)


by_decile = scores | beam.Partition(partition_fn, 10)
```

Agenda

Apache Beam

Pipeline for Creating TFRecords

Executing Pipelines with Dataflow

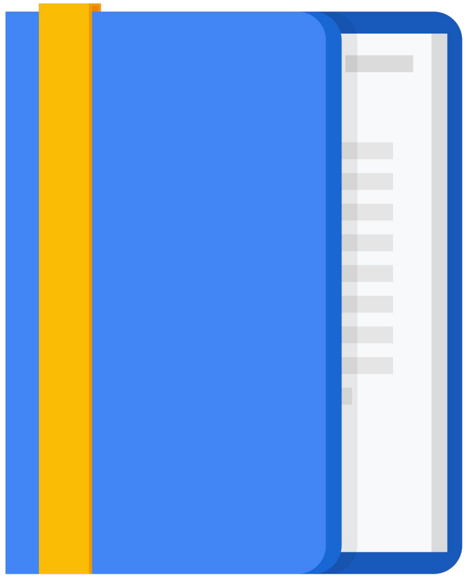
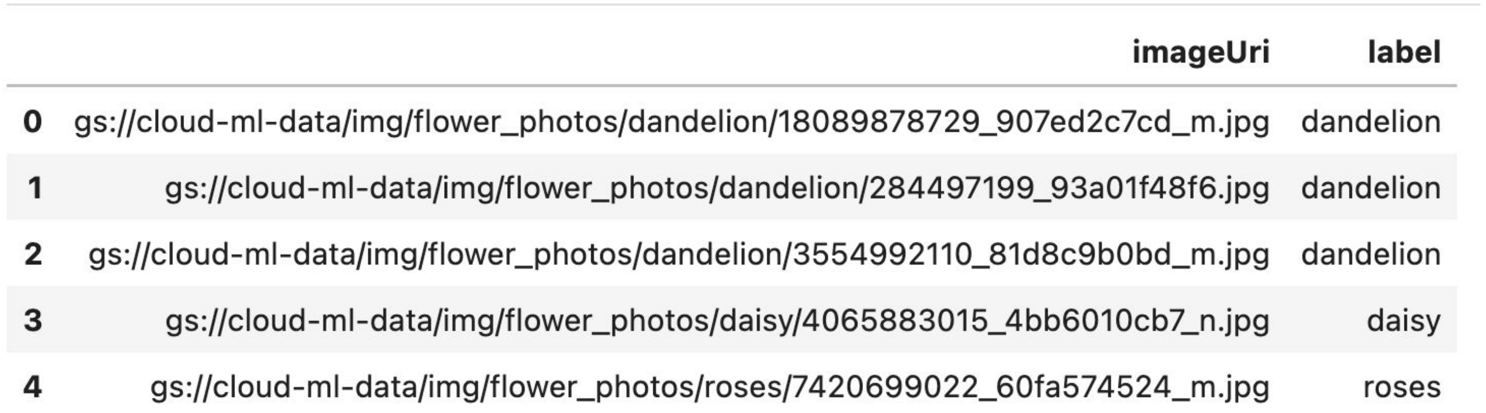


Image Classification Data

CSV file <image URI, label>



	imageUri	label
0	gs://cloud-ml-data/img/flower_photos/dandelion/18089878729_907ed2c7cd_m.jpg	dandelion
1	gs://cloud-ml-data/img/flower_photos/dandelion/284497199_93a01f48f6.jpg	dandelion
2	gs://cloud-ml-data/img/flower_photos/dandelion/3554992110_81d8c9b0bd_m.jpg	dandelion
3	gs://cloud-ml-data/img/flower_photos/daisy/4065883015_4bb6010cb7_n.jpg	daisy
4	gs://cloud-ml-data/img/flower_photos/roses/7420699022_60fa574524_m.jpg	roses

Pipeline

```
rows = ( p | "Read CSV" >> beam.io.ReadFromText(DATASET_FILE)
           | "Parse CSV" >> beam.ParDo(ParseCsv()))

train, val = ( rows
               | "Create TF Examples" >> beam.ParDo(CreateTFExample())
               | "Split Data" >> beam.Partition(partition_fn, 2, train_percent=TRAIN_PERCENT))

write_train = ( train
                   | "Serialize Training Examples" >> beam.Map(lambda x: x.SerializeToString())
                   | "Write Train" >> beam.io.tfrecordio.WriteToTFRecord(
                      f"{OUTPUT_DIR}/train.tfrecord", num_shards=10
                   )
              )
write_val = ( val
                   | "Serialize Validation Examples" >> beam.Map(lambda x:
                                                            x.SerializeToString())
                   | "Write Validation" >> beam.io.tfrecordio.WriteToTFRecord(
                      f"{OUTPUT_DIR}/eval.tfrecord", num_shards=3
                   )
            )
```

Pipeline

Read file from GCS

```
rows = ( p | "Read CSV" >> beam.io.ReadFromText(DATASET_FILE)
           | "Parse CSV" >> beam.ParDo(ParseCsv()))

train, val = ( rows
               | "Create TF Examples" >> beam.ParDo(CreateTFExample())
               | "Split Data" >> beam.Partition(partition_fn, 2, train_percent=TRAIN_PERCENT))

write_train = ( train
                    | "Serialize Training Examples" >> beam.Map(lambda x: x.SerializeToString())
                    | "Write Train" >> beam.io.tfrecordio.WriteToTFRecord(
                        f"{OUTPUT_DIR}/train.tfrecord", num_shards=10
                    )
              )
write_val = ( val
                    | "Serialize Validation Examples" >> beam.Map(lambda x:
                                                             x.SerializeToString())
                    | "Write Validation" >> beam.io.tfrecordio.WriteToTFRecord(
                        f"{OUTPUT_DIR}/eval.tfrecord", num_shards=3
                    )
            )
```

Pipeline

Parse each row into URI and label

```
rows = ( p | "Read CSV" >> beam.io.ReadFromText(DATASET_FILE)
           | "Parse CSV" >> beam.ParDo(ParseCsv()))

train, val = ( rows
               | "Create TF Examples" >> beam.ParDo(CreateTFExample())
               | "Split Data" >> beam.Partition(partition_fn, 2, train_percent=TRAIN_PERCENT))

write_train = ( train
                   | "Serialize Training Examples" >> beam.Map(lambda x: x.SerializeToString())
                   | "Write Train" >> beam.io.tfrecordio.WriteToTFRecord(
                       f"{OUTPUT_DIR}/train.tfrecord", num_shards=10
                   )
              )
write_val = ( val
                   | "Serialize Validation Examples" >> beam.Map(lambda x:
                                                              x.SerializeToString())
                   | "Write Validation" >> beam.io.tfrecordio.WriteToTFRecord(
                       f"{OUTPUT_DIR}/eval.tfrecord", num_shards=3
                   )
            )
```

Pipeline

```
class CSVRow(typing.NamedTuple):
    image_uri: str
    label: str



class ParseCsv(beam.DoFn):
    def process(self, element):
        image_uri, label = element.split(',')
        yield CSVRow(
            image_uri = image_uri,
            label = label
        )
```

Pipeline

Create TF Examples

```
rows = ( p | "Read CSV" >> beam.io.ReadFromText(DATASET_FILE)
           | "Parse CSV" >> beam.ParDo(ParseCsv()))

train, val = ( rows
               | "Create TF Examples" >> beam.ParDo(CreateTFExample())
               | "Split Data" >> beam.Partition(partition_fn, 2, train_percent=TRAIN_PERCENT))

write_train = ( train
                    | "Serialize Training Examples" >> beam.Map(lambda x: x.SerializeToString())
                    | "Write Train" >> beam.io.tfrecordio.WriteToTFRecord(
                        f"{OUTPUT_DIR}/train.tfrecord", num_shards=10
                    )
              )
write_val = ( val
                    | "Serialize Validation Examples" >> beam.Map(lambda x:
                                                              x.SerializeToString())
                    | "Write Validation" >> beam.io.tfrecordio.WriteToTFRecord(
                        f"{OUTPUT_DIR}/eval.tfrecord", num_shards=3
                    )
            )
```

Pipeline

```
class CreateTFExample(beam.DoFn):

    def process(self, element):
        CLASSES = ['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
        img = tf.io.decode_jpeg(tf.io.read_file(element.image_uri))

        feature = {
            "image": _image_feature(img),
            "label": _int64_feature(CLASSES.index(element.label)),
        }

        yield tf.train.Example(features=tf.train.Features(feature=feature))
```

Pipeline

Train/validation split

```
rows = ( p | "Read CSV" >> beam.io.ReadFromText(DATASET_FILE)
           | "Parse CSV" >> beam.ParDo(ParseCsv()))

train, val = ( rows
               | "Create TF Examples" >> beam.ParDo(CreateTFExample())
               | "Split Data" >> beam.Partition(partition_fn, 2, train_percent=TRAIN_PERCENT))

write_train = ( train
                    | "Serialize Training Examples" >> beam.Map(lambda x: x.SerializeToString())
                    | "Write Train" >> beam.io.tfrecordio.WriteToTFRecord(
                        f"{OUTPUT_DIR}/train.tfrecord", num_shards=10
                    )
              )
write_val = ( val
                    | "Serialize Validation Examples" >> beam.Map(lambda x:
                                                                  x.SerializeToString())
                    | "Write Validation" >> beam.io.tfrecordio.WriteToTFRecord(
                        f"{OUTPUT_DIR}/eval.tfrecord", num_shards=3
                    )
            )
```

Pipeline

```
def partition_fn(example, num_partitions, train_percent):
    if random.random() < train_percent:
        return 0
    return 1
```

Pipeline

Serialize TF Examples

```
rows = ( p | "Read CSV" >> beam.io.ReadFromText(DATASET_FILE)
           | "Parse CSV" >> beam.ParDo(ParseCsv()))

train, val = ( rows
               | "Create TF Examples" >> beam.ParDo(CreateTFExample())
               | "Split Data" >> beam.Partition(partition_fn, 2, train_percent=TRAIN_PERCENT))

write_train = ( train
                  | "Serialize Training Examples" >> beam.Map(lambda x: x.SerializeToString())
                  | "Write Train" >> beam.io.tfrecordio.WriteToTFRecord(
                    f"{OUTPUT_DIR}/train.tfrecord", num_shards=10
                  )
              )
write_val = ( val
                  | "Serialize Eval Examples" >> beam.Map(lambda x: x.SerializeToString())
                  | "Write Validation" >> beam.io.tfrecordio.WriteToTFRecord(
                    f"{OUTPUT_DIR}/eval.tfrecord", num_shards=3
                  )
            )
```

Pipeline

Write to .tfrecords

```
rows = ( p | "Read CSV" >> beam.io.ReadFromText(DATASET_FILE)
           | "Parse CSV" >> beam.ParDo(ParseCsv()))

train, val = ( rows
               | "Create TF Examples" >> beam.ParDo(CreateTFExample())
               | "Split Data" >> beam.Partition(partition_fn, 2, train_percent=TRAIN_PERCENT))

write_train = ( train
                   | "Serialize Training Examples" >> beam.Map(lambda x: x.SerializeToString())
                   | "Write Train" >> beam.io.tfrecordio.WriteToTFRecord(
                       f"{OUTPUT_DIR}/train.tfrecord", num_shards=10
                   )
              )
write_val = ( val
                   | "Serialize Validation Examples" >> beam.Map(lambda x:
                                                              x.SerializeToString())
                   | "Write Validation" >> beam.io.tfrecordio.WriteToTFRecord(
                       f"{OUTPUT_DIR}/eval.tfrecord", num_shards=3
                   )
            )
```

Pipeline

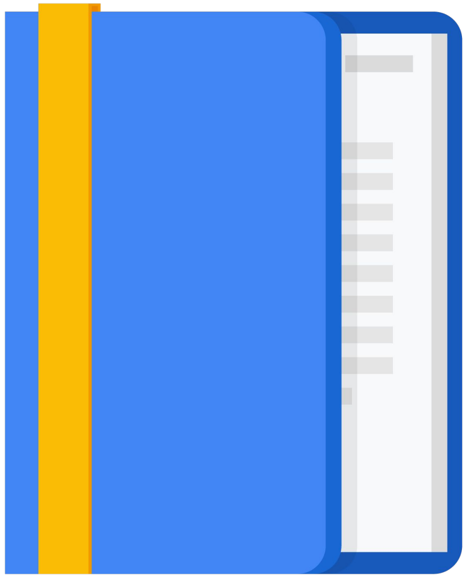
```
rows = ( p | "Read CSV" >> beam.io.ReadFromText(DATASET_FILE)
           | "Parse CSV" >> beam.ParDo(ParseCsv()))

train, val = ( rows
               | "Create TF Examples" >> beam.ParDo(CreateTFExample())
               | "Split Data" >> beam.Partition(partition_fn, 2, train_percent=TRAIN_PERCENT))

write_train = ( train
                   | "Serialize Training Examples" >> beam.Map(lambda x: x.SerializeToString())
                   | "Write Train" >> beam.io.tfrecordio.WriteToTFRecord(
                       f"{OUTPUT_DIR}/train.tfrecord", num_shards=10
                   )
              )
write_val = ( val
                   | "Serialize Validation Examples" >> beam.Map(lambda x:
                                                          x.SerializeToString())
                   | "Write Validation" >> beam.io.tfrecordio.WriteToTFRecord(
                       f"{OUTPUT_DIR}/eval.tfrecord", num_shards=3
                   )
            )

p.run()
```

Agenda

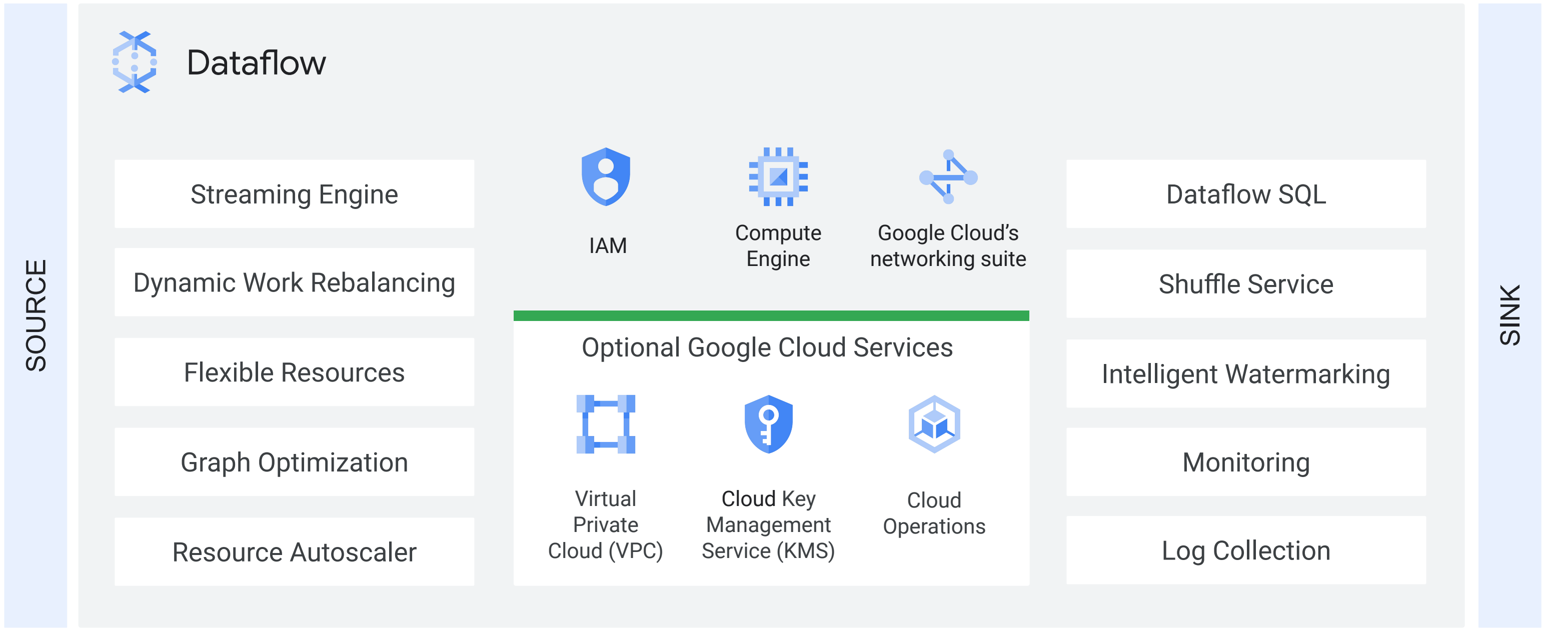
Apache Beam

Pipeline for Creating TFRecords

Executing Pipelines with Dataflow



The Google Cloud runner: Dataflow



How to run a pipeline in Python

```python
import apache_beam as beam

options = {'project': <project>,
           'runner': 'DataflowRunner',
           'region': <region>,
           'setup_file': <setup.py file>}


pipeline_options = beam.pipeline.PipelineOptions(
   flags=[], **options)


pipeline = beam.Pipeline(
   options = pipeline_options)
```

Run locally:

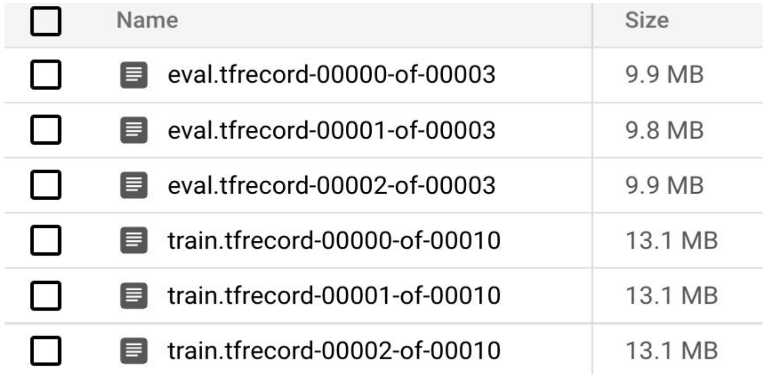
```
python3 my_pipeline.py
```

Run on Cloud Dataflow:

```
python3 my_pipeline.py  \
 --project=${PROJECT_ID} \
 --region=${REGION} \
 --stagingLocation=${BUCKET}/stage/ \
 --tempLocation=${BUCKET}/temp/ \
 --runner=DataflowRunner
```

Lab

Creating TFRecords with Beam/Dataflow



☐	Name	Size
☐	eval.tfrecord-00000-of-00003	9.9 MB
☐	eval.tfrecord-00001-of-00003	9.8 MB
☐	eval.tfrecord-00002-of-00003	9.9 MB
☐	train.tfrecord-00000-of-00010	13.1 MB
☐	train.tfrecord-00001-of-00010	13.1 MB
☐	train.tfrecord-00002-of-00010	13.1 MB

notebooks/image_models/labs/create_tfrecords_at_scale.ipynb