Dataflow code -KirkYagami👨‍💻🕵️

```
gcloud auth login
gcloud services enable dataflow compute_component logging storage_component
storage_api bigquery pubsub datastore.googleapis.com
cloudresourcemanager.googleapis.com

# Create local authentication credentials for your user account:
# Instead of generating and managing service account keys manually, you can
use this method during development.
gcloud auth application-default login



# Add Dataflow Admin role
gcloud projects add-iam-policy-binding bigdata3844 \
  --member="serviceAccount:22777180107-compute@developer.gserviceaccount.com"
\
  --role="roles/dataflow.admin"


# Add Dataflow Worker role
gcloud projects add-iam-policy-binding bigdata3844 \
  --member="serviceAccount:22777180107-compute@developer.gserviceaccount.com"
\
  --role="roles/dataflow.worker"


# Add Storage Object Admin role
gcloud projects add-iam-policy-binding bigdata3844 \
  --member="serviceAccount:22777180107-compute@developer.gserviceaccount.com"
\
  --role="roles/storage.objectAdmin"



# storage bucket
gcloud storage buckets create gs://rev_dataflow --default-storage-class
STANDARD --location US

# Check version
# python --version # 3.10
# python -m pip --version
```

```
pip install 'apache-beam[gcp]'

# locally
python -m apache_beam.examples.wordcount \
  --output outputs

cat outputs*


# Use Dataflow
python -m apache_beam.examples.wordcount \
    --region DATAFLOW_REGION \
    --input gs://dataflow-samples/shakespeare/kinglear.txt \
    --output gs://BUCKET_NAME/results/outputs \
    --runner DataflowRunner \
    --project PROJECT_ID \
    --temp_location gs://BUCKET_NAME/tmp/


python -m apache_beam.examples.wordcount \
    --region us-central1 \
    --input gs://dataflow-samples/shakespeare/kinglear.txt \
    --output gs://rev_dataflow/results/outputs \
    --runner DataflowRunner \
    --project bigdata3844 \
    --temp_location gs://rev_dataflow/tmp/


gcloud projects add-iam-policy-binding bigdata3844 \
  --member="serviceAccount:sa-local2gcs@bigdata3844.iam.gserviceaccount.com"
\
  --role="roles/storage.objectAdmin"

gcloud projects add-iam-policy-binding bigdata3844 \
    --member="serviceAccount:sa-
local2gcs@bigdata3844.iam.gserviceaccount.com" \
    --role="roles/dataflow.admin"


# The above service account is a user variable.
```

https://github.com/apache/beam/blob/master/sdks/python/apache_beam/examples/wordcount.py

wordcount.py

```
"""A word-counting workflow."""
import argparse
import logging
import re

import apache_beam as beam
from apache_beam.io import ReadFromText
from apache_beam.io import WriteToText
from apache_beam.options.pipeline_options import PipelineOptions
from apache_beam.options.pipeline_options import SetupOptions


class WordExtractingDoFn(beam.DoFn):
    def process(self, element):
        """Receives a single element (a line of text) and splits it into
words."""
        return element.split()


def run(argv=None, save_main_session=True):
  """Main entry point; defines and runs the wordcount pipeline."""
  parser = argparse.ArgumentParser()
  parser.add_argument(
      '--input',
      dest='input',
      default='gs://dataflow-samples/shakespeare/kinglear.txt',
      help='Input file to process.')
  parser.add_argument(
      '--output',
      dest='output',
      required=True,
      help='Output file to write results to.')
  known_args, pipeline_args = parser.parse_known_args(argv)

  # We use the save_main_session option because one or more DoFn's in this
  # workflow rely on global context (e.g., a module imported at module
level).

  pipeline_options = PipelineOptions(pipeline_args)
  pipeline_options.view_as(SetupOptions).save_main_session =
save_main_session

  # The pipeline will be run on exiting the with block.
```

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

    # Read the text file[pattern] into a PCollection.
    lines = p | 'Read' >> ReadFromText(known_args.input)

    counts = (
        lines
        | 'Split' >>
(beam.ParDo(WordExtractingDoFn()).with_output_types(str))
        | 'PairWithOne' >> beam.Map(lambda x: (x, 1))
        | 'GroupAndSum' >> beam.CombinePerKey(sum))

    # Format the counts into a PCollection of strings.
    def format_result(word, count):
      return '%s: %d' % (word, count)

    output = counts | 'Format' >> beam.MapTuple(format_result)

    # Write the output using a "Write" transform that has side effects.
    # pylint: disable=expression-not-assigned
    output | 'Write' >> WriteToText(known_args.output)

if __name__ == '__main__':
  logging.getLogger().setLevel(logging.INFO)
  run()
```

```
python wordcount.py --input sample.txt --output ./output.txt
```