Apache Beam Pipeline:

1. A Pipeline **encapsulates your entire data processing task, from start to finish**. This includes reading input data, transforming that data, and writing output data.
2. All Beam driver programs must create a Pipeline. When you create the Pipeline, you must also specify the execution options that tell the Pipeline where and how to run.

    - A Beam pipeline is a graph (specifically, a **directed acyclic graph**) of all the data and computations in your data processing task.
    - Directed Acyclic Graph (DAG) is a graph that is directed and without cycles connecting the other edges. This means that it is impossible to traverse the entire graph starting at one edge. The edges of the directed graph **only go one way**.
    - This includes reading input data, transforming that data, and writing output data.
    - A pipeline is constructed by a user in their SDK of choice.
        - Python SDK
        - Java SDK
        - Go SDK
    - Then, the pipeline makes its way to the runner either through the SDK directly or through the Runner API's RPC interface.

The simplest pipelines represent a linear flow of operations, as shown in figure 1.

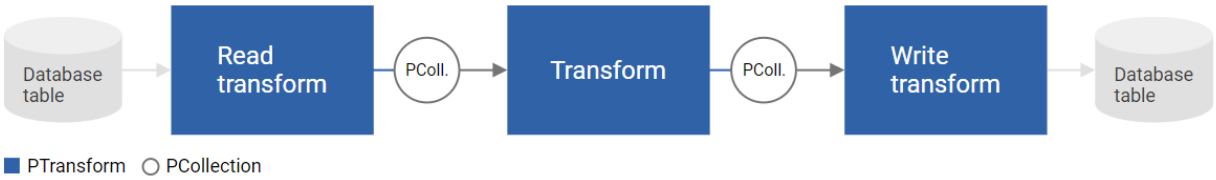

Figure 1: A linear pipeline.

- In this diagram, the boxes represent the parallel computations called *PTransforms* and the arrows with the circles represent the data (in the form of *PCollections*) that flows between the transforms.
- The data might be bounded, stored, data sets, or the data might also be unbounded streams of data.
- In Beam, most transforms apply equally to bounded and unbounded data.
- A Beam driver program typically starts by creating a `Pipeline` object, and then uses that object as the basis for creating the pipeline's data sets and its transforms.

```
#importing library
import apache_beam as beam

#from external resources
p1 = beam.Pipeline()

grocery = (p1
           | "Read from Text" >> beam.io.ReadFromText("grocery.txt", skip_header_lines=1)
           | "split the record" >> beam.Map(lambda record: record.split(','))
           | 'Filter regular' >> beam.Filter(lambda record: record[2] == 'Regular')
           | 'Write to text'>> beam.io.WriteToText('regular_filter.txt'))  #| beam.Map(print))

p1.run()
```

Yellow – pipeline

Red – Ptransform

Blue – creating beam pipeline

Cloud Dataflow:

- Unified stream and batch data processing that's serverless, fast, and cost-effective.
- Fully managed data processing service
- Automated provisioning and management of processing resources
- Horizontal autoscaling of worker resources to maximize resource utilization
- OSS community-driven innovation with Apache Beam SDK
- Reliable and consistent exactly-once processing

Resources:

https://beam.apache.org/documentation/programming-guide/#creating-a-pipeline

https://beam.apache.org/documentation/basics/#pipeline

https://cloud.google.com/dataflow