What is Dataflow

Google Cloud Dataflow is a fully managed, serverless service for building and executing data pipelines that process both batch and streaming data, enabling scalable ETL workflows, real-time stream analytics, and more, using the Apache Beam programming mode

SDKs

Find status and reference information on all of the available Beam SDKs.

- **Java SDK**
- **Python SDK**
- **Go SDK**

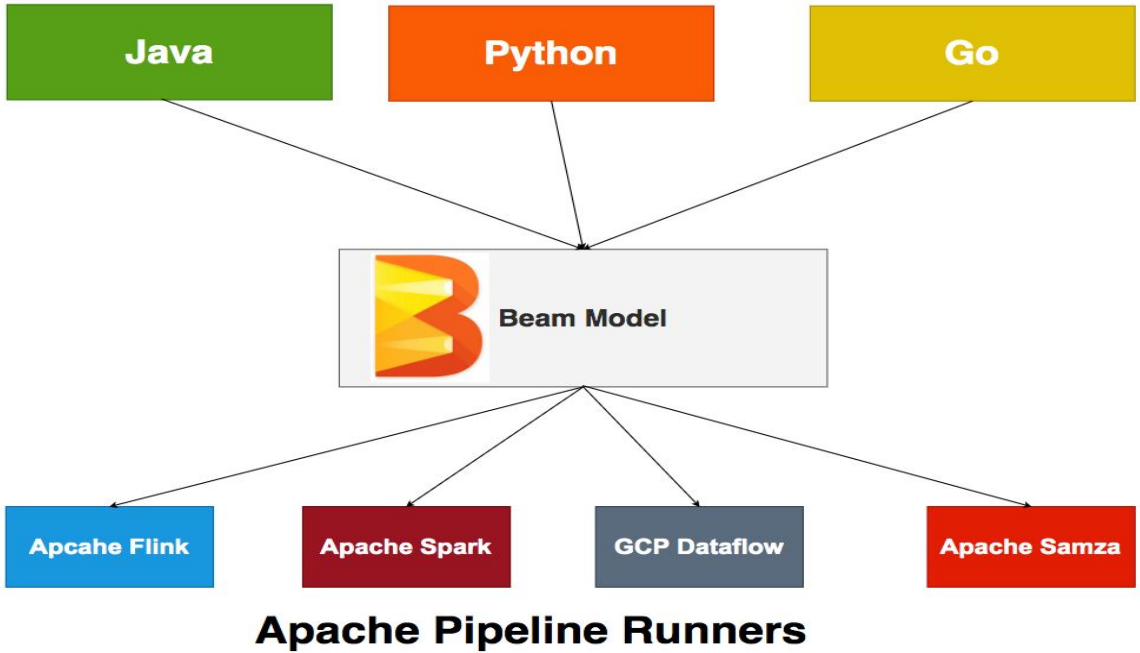
Transform catalogs

Beam’s transform catalogs contain explanations and code snippets for Beam’s built-in transforms.

- Java transform catalog
- Python transform catalog

Runners

A Beam Runner runs a Beam pipeline on a specific (often distributed) data processing system.



Beam Runners

- Supported Beam Runners are
  - Direct Runner (test and development)
  - Apache Apex
  - Apache Flink
  - Apache Gearpump
  - Apache Hadoop MapReduce
  - Apache Nemo
  - Apache Samza
  - Apache Spark
  - Google Cloud Dataflow
  - Hazelcast Jet
  - IBM Streams
  - JStorm

Dataflow runner usecase

Use Case 1: ETL (Extract, Transform, Load) for Data Warehousing

- **Description:** In a typical ETL process, data is extracted from different sources, transformed (cleaned, normalized), and then loaded into a data warehouse for analytics.

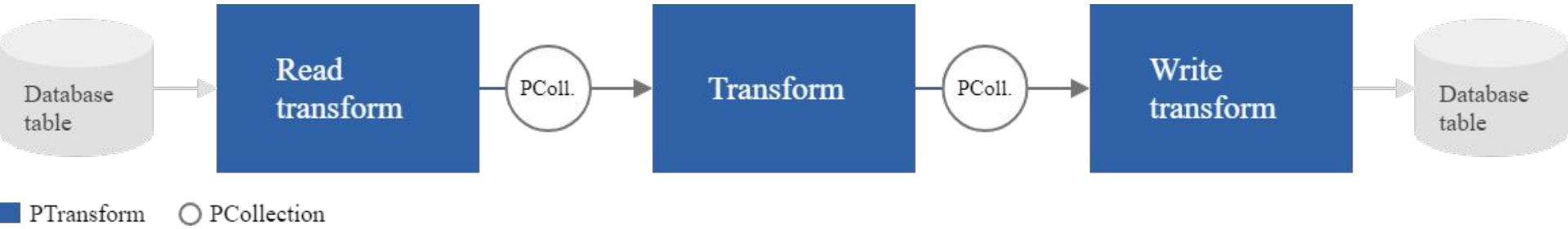
- **How Dataflow Helps:** Dataflow is used to build scalable ETL pipelines, transforming large datasets in parallel. It can handle ingestion of structured or unstructured data from various sources like Cloud Storage, BigQuery, or Pub/Sub.

`Pipeline`

`Pipeline` encapsulates your entire data processing task, from start to finish.

This includes:-

- reading input data
- transforming that data
- writing output data



`PCollection`

`PCollection` **represents a distributed data set that your Beam pipeline operates on**.

It's a collection of bounded data set or unbounded data data se:-

- ***Bounded*** - meaning it comes from a fixed source like a file.
- ***Unbounded*** - meaning it comes from a continuously updating source via a subscription (pub-sub/kafka).

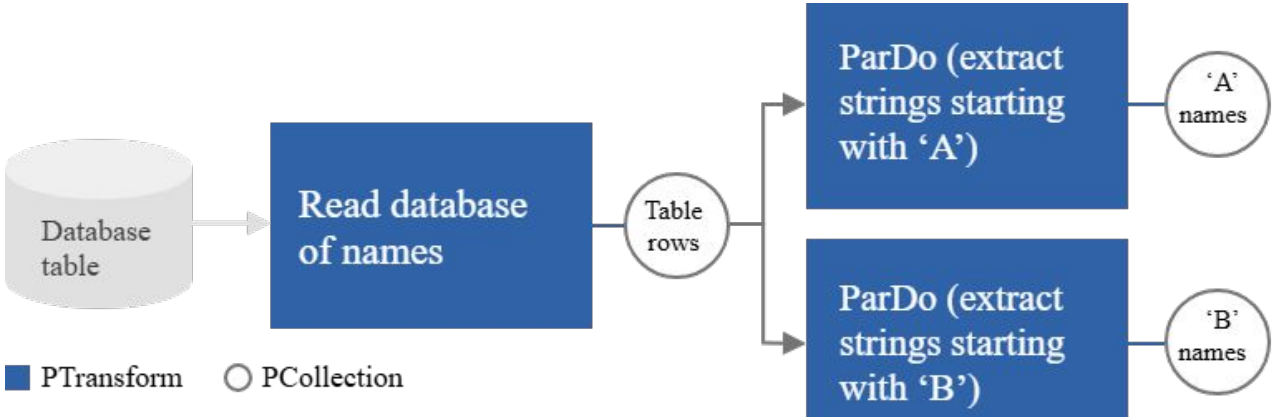
Your pipeline typically creates an initial `PCollection` by reading data from an external data source.

We can also create a `PCollection` from in-memory data within your driver program. From there, `PCollection`s are the inputs and outputs for each step in your pipeline

`PTransform`

`PTransform` represents a **data processing operation**, or a step, in your pipeline.

- **Every `PTransform` takes one or more `PCollection` objects as input**.
- performs a processing function that you provide on the elements of that `PCollection`.
- **`PTransform`** produces zero or more output `PCollection` objects.
- Transforms can **change, filter, group, analyze**, or otherwise process the elements in a **`PCollection`**



I/O transforms

Beam comes with a number of “IOs” - library.

`PTransform`s that read or write data to various external storage systems

For Ex :-

Beam.io.ReadFromText(“file_path”)

Creating a pipeline

The `Pipeline` abstraction encapsulates all the data and steps in your data processing task.

The **Beam driver program typically starts by constructing** a **Pipeline object**, and then using that object as the basis for **creating the pipeline's data sets as `PCollection`s.**

```
import apache_beam as beam


with beam.Pipeline(runner='DirectRunner') as p:

    (p

     | 'Read' >> beam.io.ReadFromText('input.txt')

     | 'Parse' >> beam.Map(lambda line: line.split(','))

     | 'Filter' >> beam.Filter(lambda rec: rec[2] == 'NY')

     | 'Write' >> beam.io.WriteToText('output.txt'))
```