Exploring the Apache Beam SDK for Modeling Streaming Data for Processing

UNDERSTANDING PIPELINES, PCOLLECTIONS, AND PTRANSFORMS

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Apache Beam for embarassingly parallel operations

Pipelines, PCollections, and PTransforms

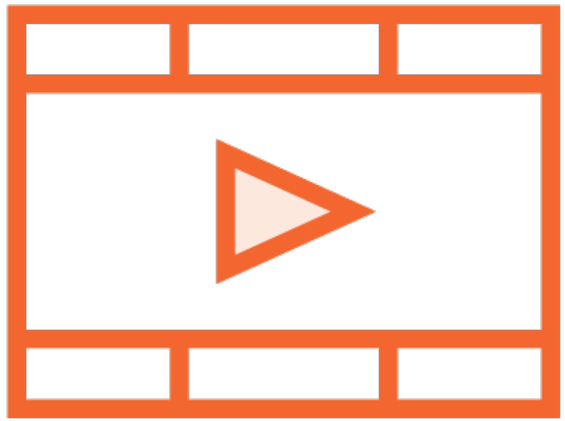
Creating PCollections

Characteristics of PCollections

Drivers and runners

Prerequisites and Course Outline

Prerequisites

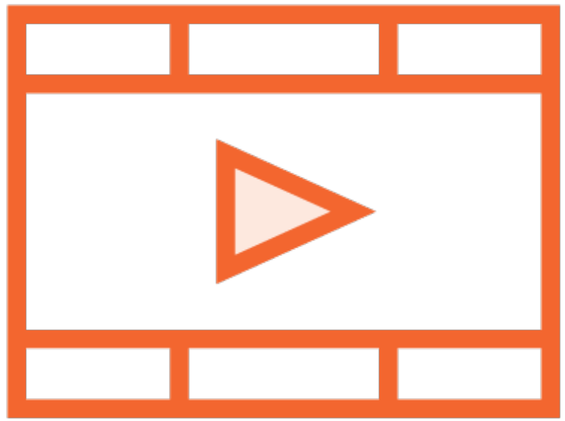


Basic understanding of working with unbounded streams

Experience programming in Java

Apache Maven for dependency management

Prerequisites



Modeling Streaming Data for Processing with Apache Beam

Prerequisites

- Understanding Pipelines, PCollections, and PTransforms
- Executing Pipelines to Process Streaming Data
- Applying Transformations to Streaming Data
- Working with Windowing and Join Operations
- Performing SQL Queries on Streaming Data

Introducing Apache Beam

Apache Beam

Open-source, unified model for defining both batch and streaming, data-parallel piplines.

Apache Beam

Open-source, **unified model** for defining both **batch and streaming**, data-parallel piplines.

Apache Beam

Pipelines specify transformations on the data. Pipelines are executed by a distributed processing back-end.

Apache Beam

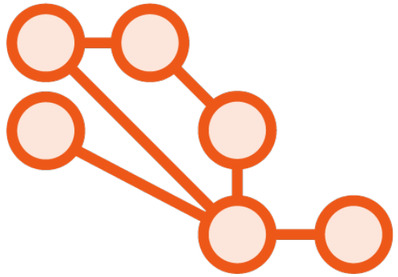
Pipelines specify transformations on the data. Pipelines are **executed by a distributed processing back-end**.

Using Apache Beam

Write code for pipeline

Submit job for execution

Back-end assigns workers to execute



Pipeline parallelized and executed

Writing Code

Java

Python

Go

Scio - a Scala interface

Driver Program

Driver program utilizes Beam SDKs

Defines pipeline

Input, transforms, outputs

Execution options for pipeline

Driver program is executed on one of the Apache Beam back-ends

Available Back-ends

Apache Flink

Apache Spark

Google Cloud Dataflow

Apache Samza

Hazelcast Jet

Beam and Runners

Apache Beam	Runners
API specification	API implementation
Platform-agnostic	Platform-dependent
Superset of all actually provided capabilities	Only subset of Apache Beam APIs implemented by each back-end

Pipeline and PipelineOptions

Apache Beam Pipeline Components

Data source

Batch or streaming data to be processed

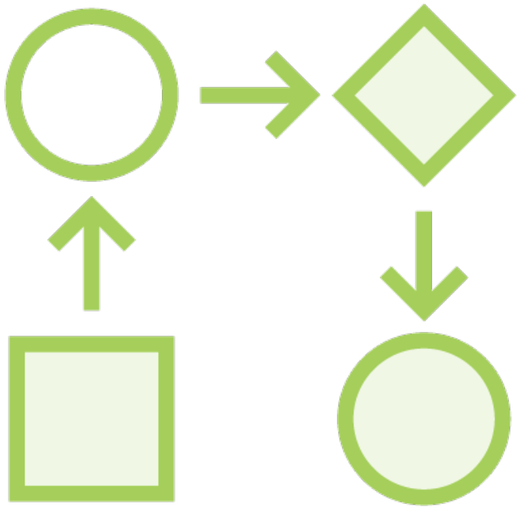
Transformations

Modify the data to get it in the right final form

Data sink

Store the data in some kind of persistent storage

Pipeline



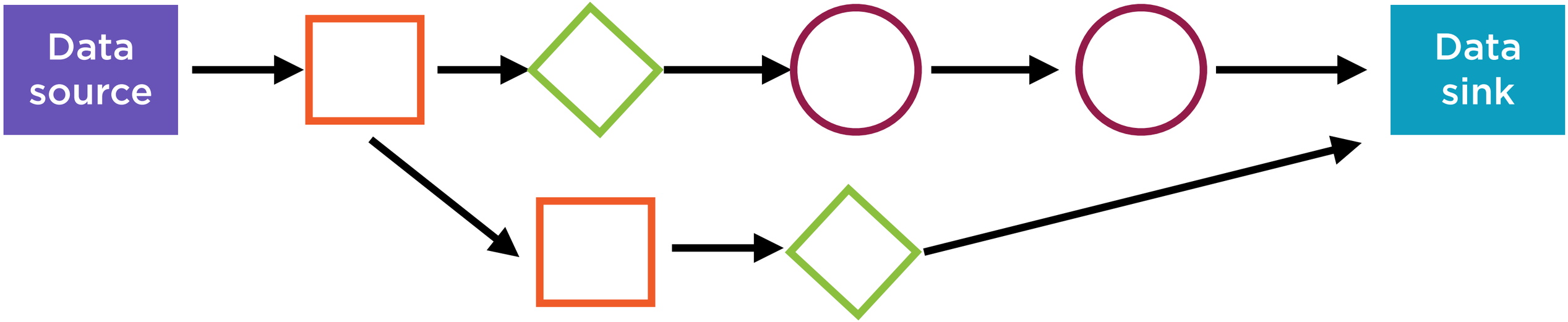
Single, potentially repeatable job

Executes from start to finish

Applies transformations to the data

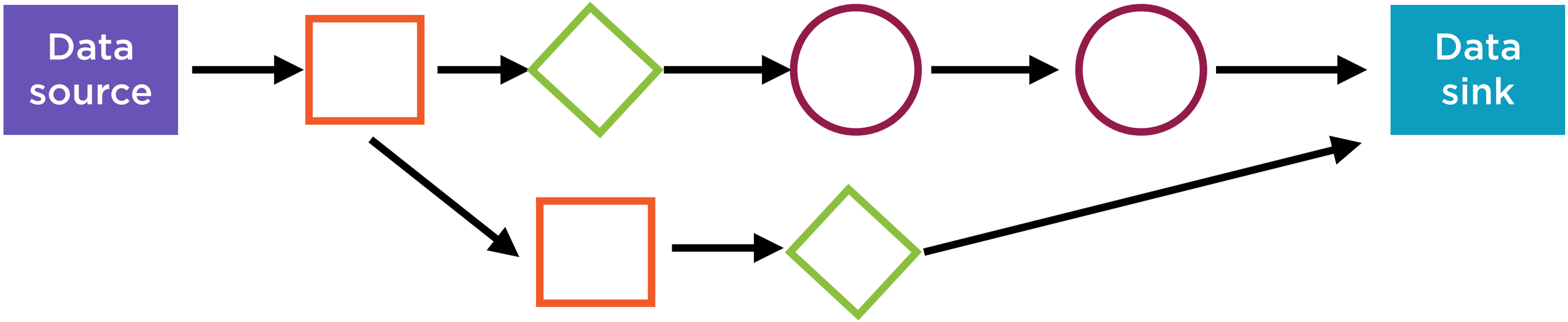
May have operations which can be performed in parallel

Pipeline



Directed Acyclic Graph (DAG)

Pipeline

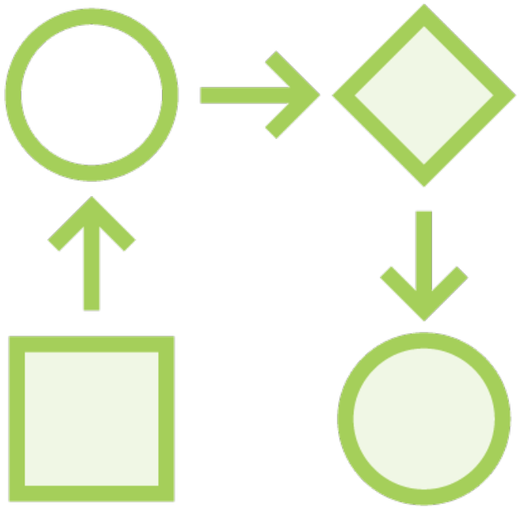


Pipeline: Entire set of computations

Pipeline

Encapsulates all data and steps in a data processing task in an object of the Pipeline class of the Beam SDK.

Pipeline



Configure using PipelineOptions objects

- Encapsulate key-value pairs

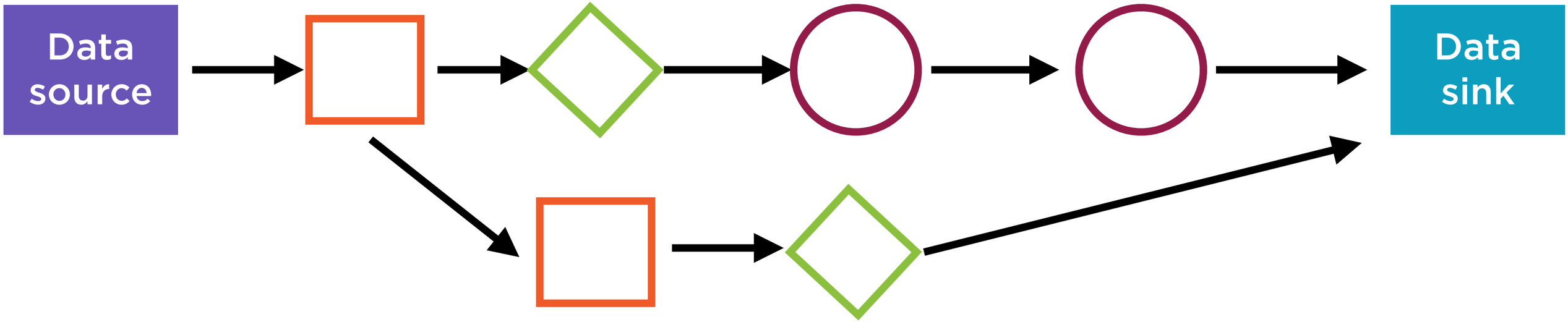
Includes choice of runner

Can do via command line arguments

Alternatively, create custom options

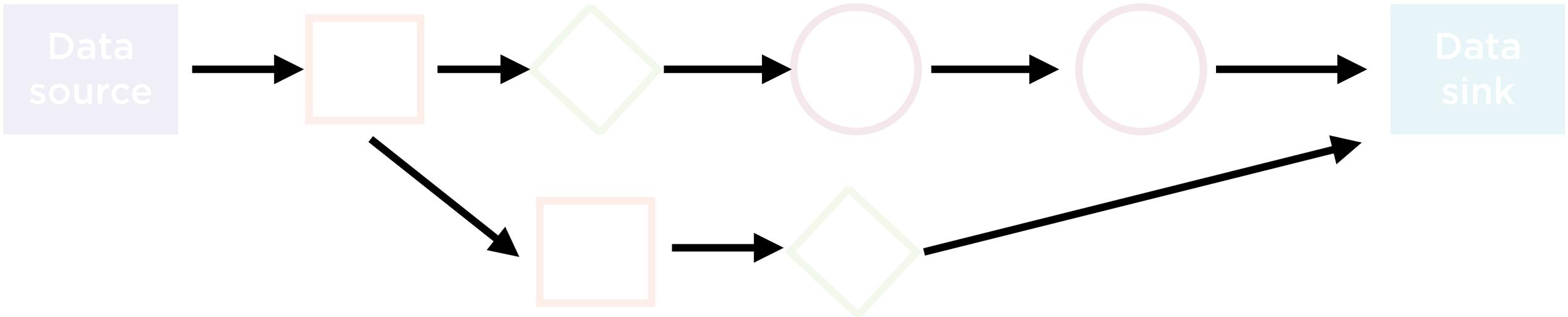
PCollections and PTransforms

Pipeline



Pipeline: Entire set of computations

PCollections

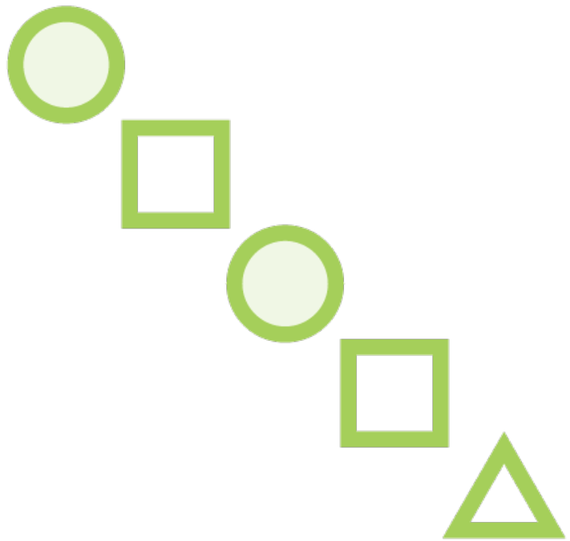


PCollections: Edges of DAG

PCollection

Interface in the Beam SDK; represents a multi-element data set which may or may not be distributed. Can be created by reading from an external data source, or by transforming another PCollection.

PCollections



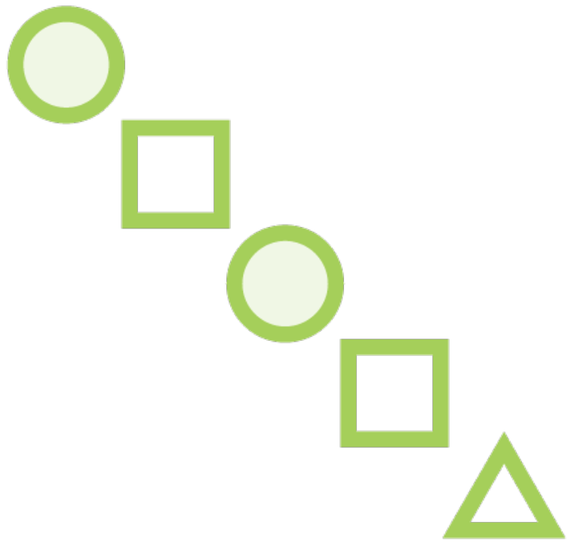
Specialized container classes

Can represent data sets of virtually unlimited size

Created using a Pipeline object

Owned by the Pipeline, cannot be shared across multiple pipelines

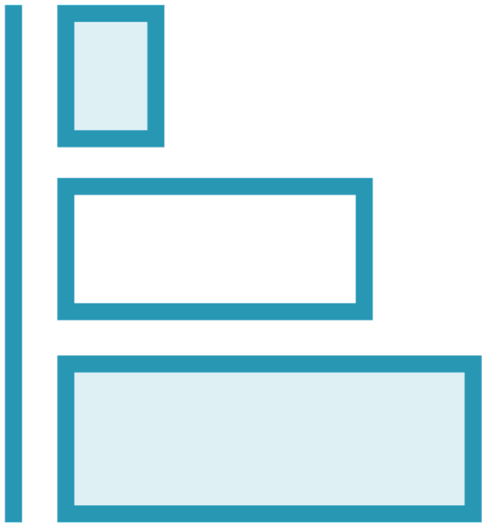
Creating PCollections



Reading data from an external source using Beam-provided I/O adapters

From in-memory data

Built-in I/O Connectors



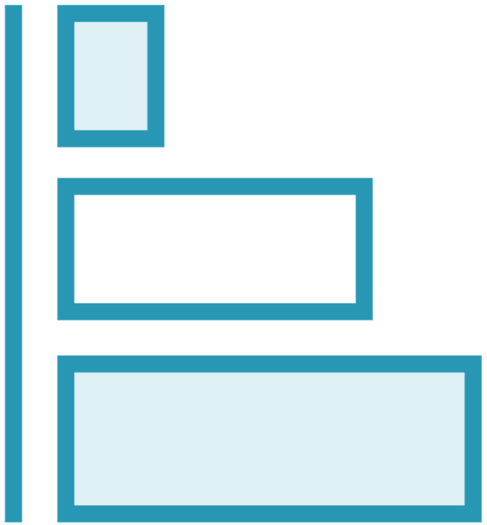
File-based I/O connectors

- Text, Avro, TensorFlow records, Parquet

File system interface for file systems agnostic code

- HDFS, GCS, Local, S3

Built-in I/O Connectors



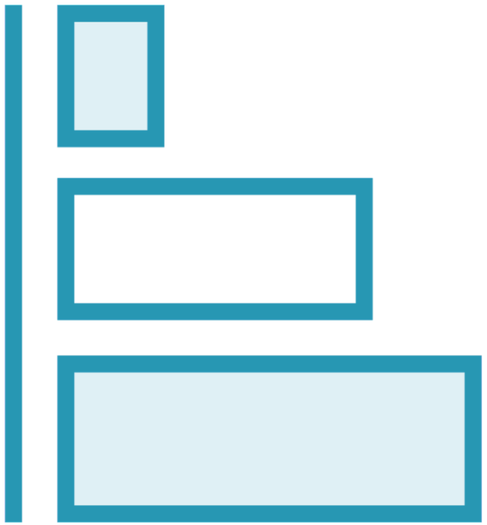
Messaging connectors

- Kinesis, Kafka, PubSub, RabbitMQ

Database connectors

- Cassandra, Elasticsearch, BigQuery, MongoDB, Redis

Custom I/O Connectors

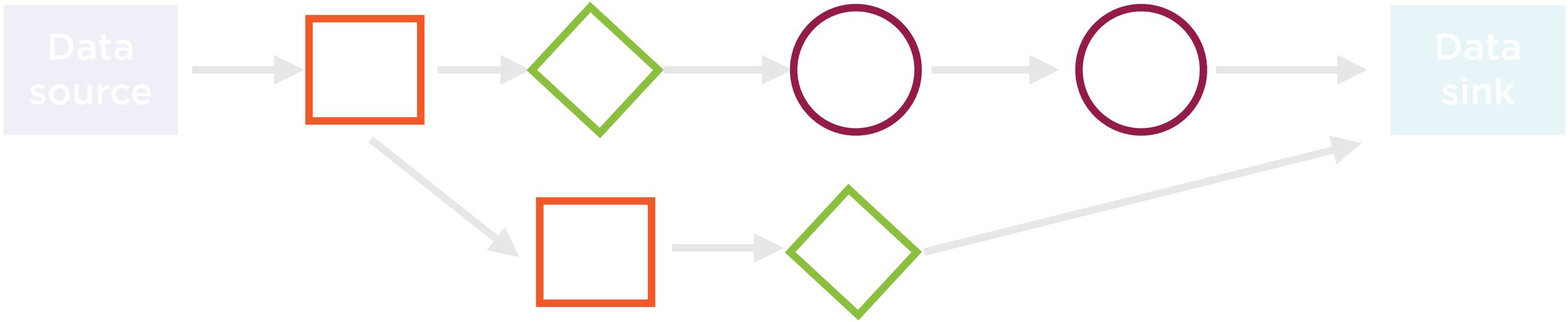


If streaming source or sink not supported by an existing connector

Can create custom I/O connectors

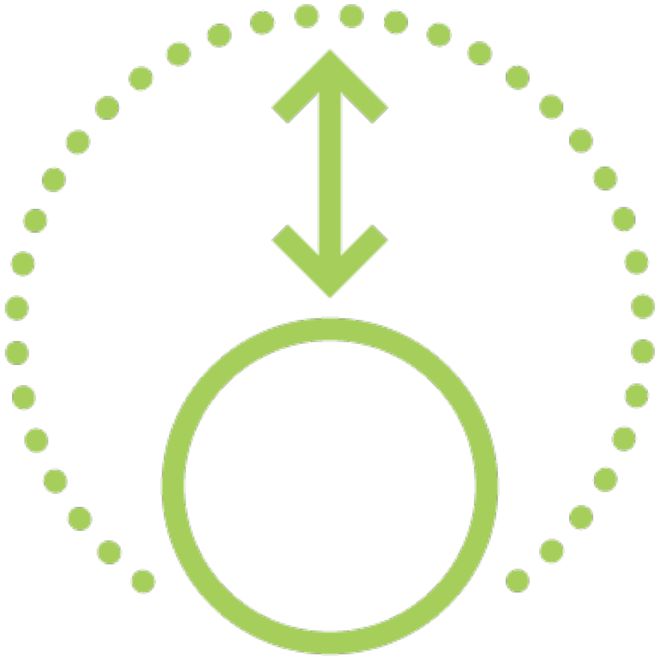
All Beam sources and sinks are composite transforms

PTransform



PTransforms: Nodes in DAG

Transforms



Code which modifies elements in a PCollection

Two types of transforms

- PTransforms: logical operations
- I/O Transforms: read or write external storage systems

PTransform

Interface in the Beam SDK; represents single step of the pipeline that takes in an input PCollection and transforms it to zero or more output PCollections.

Characteristics of PCollections

Characteristics of PCollections

Element type	**Element schema**	**Immutability**
Random access	**Size and boundedness**	**Element timestamps**

Characteristics of PCollections

Element type

Element schema

Immutability

Random access

Size and boundedness

Element timestamps

Element Type

PCollections can hold any kind of element

All elements should be of the same type

Beam should be able to encode each element as a byte string

Encoded elements are passed to distributed workers

Characteristics of PCollections

Element type	**Element schema**	Immutability
Random access	Size and boundedness	Element timestamps

Element Schema

Elements can be complex structures with a schema

Schemas provide a way to express types as named fields

Beam supports JSON structures, Protocol Buffers, Avro, and database records

Element Schema

Fields in elements can be of primitive types int, long, string, boolean etc.

Certain composite types i.e. collections are also supported

Fields can be marked optional (nullable)

Element Schema

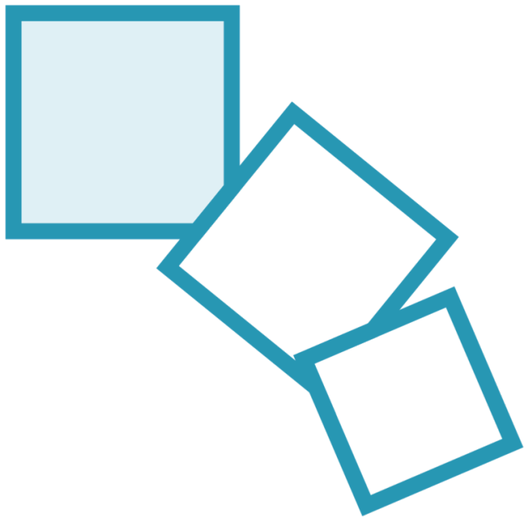
PCollections have the ability to infer schema from common Java types

POJOs with specific annotations will allow Beam to infer schema

Characteristics of PCollections

Element type	Element schema	**Immutability**
Random access	Size and boundedness	Element timestamps

Immutability



PCollections are immutable

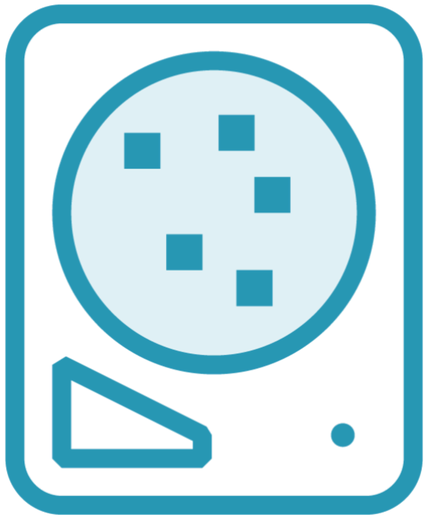
Cannot add, remove, or change elements

Elements may be operated on i.e. transformed to create new elements

Characteristics of PCollections

Element type	Element schema	Immutability
Random access	Size and boundedness	Element timestamps

Random Access



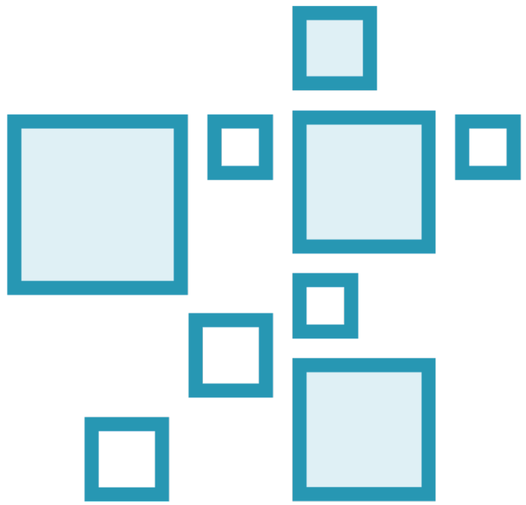
PCollections do not support random access to individual elements

Transforms need to consider every element of the collection in turn

Characteristics of PCollections

Element type	Element schema	Immutability
Random access	**Size and boundedness**	Element timestamps

Size and Boundedness



PCollections can be considered to be a large, immutable, “bag” of elements

Bounded or unbounded

Bounded PCollection from file, database

Unbounded PCollection from stream source

Characteristics of PCollections

Element type	Element schema	Immutability
Random access	Size and boundedness	**Element timestamps**

Element Timestamps

Every element has an intrinsic timestamp i.e. event time

Assigned by the data source

Used in performing stateful transformations using windows

Demo

Environment set up and pipeline properties

Driver and Runner

Driver and Runner

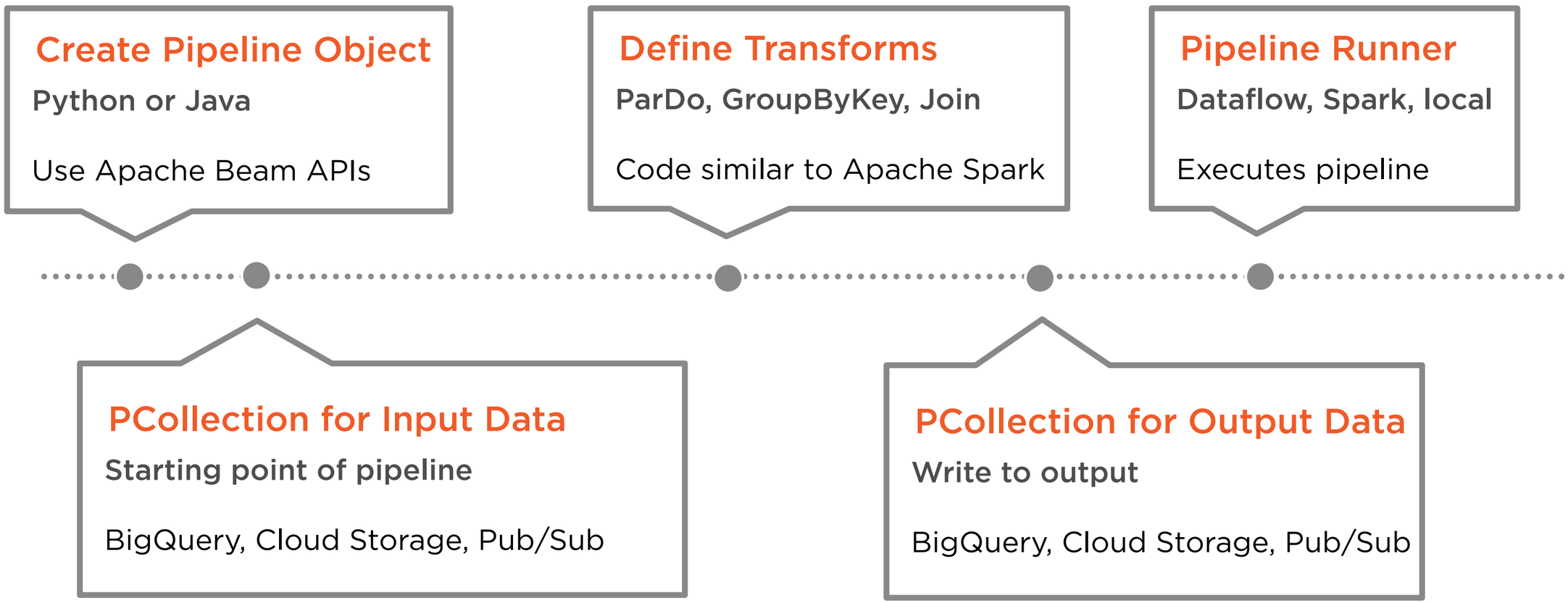
Driver defines computation DAG (pipeline)

Runner executes DAG on a back-end

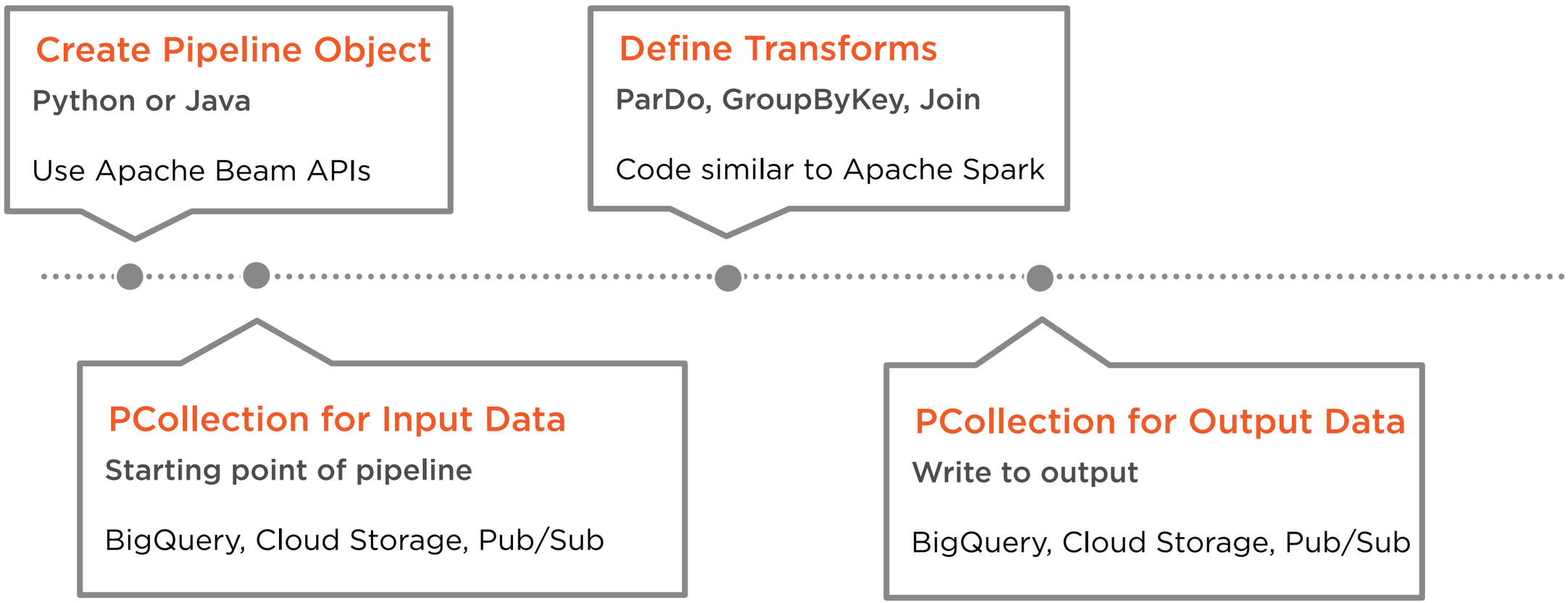
Various back-ends supported

- Apache Spark, Apache Flink, Google Cloud Dataflow

Driver and Runner



Driver


Create Pipeline Object
Python or Java
Use Apache Beam APIs
Define Transforms
ParDo, GroupByKey, Join
Code similar to Apache Spark
PCollection for Input Data
Starting point of pipeline
BigQuery, Cloud Storage, Pub/Sub
PCollection for Output Data
Write to output
BigQuery, Cloud Storage, Pub/Sub

Runner

Pipeline Runner

Dataflow, Spark, local

Executes pipeline

Direct Runner

Used in prototyping and testing code

Executes pipelines on your local machine

Checks to ensure that code uses the right semantics, guaranteed by the model

Direct Runner

Enforces immutability of elements

Enforces encodability of elements

Processes elements in arbitrary order

Checks serialization of user functions

Summary

Apache Beam for embarassingly parallel operations

Pipelines, PCollections, and PTransforms

Creating PCollections

Characteristics of PCollections

Drivers and runners

Up Next:

Executing Pipelines to Process Streaming Data