Architecting Serverless Big Data Solutions Using Google Dataflow

INTRODUCING DATAFLOW

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Serverless and fully managed data transformation service

Pipelines for batch and streaming data

Conforms to Apache Beam API

Some similarities to Apache Spark

Important architectural differences as well

Prerequisites and Course Outline

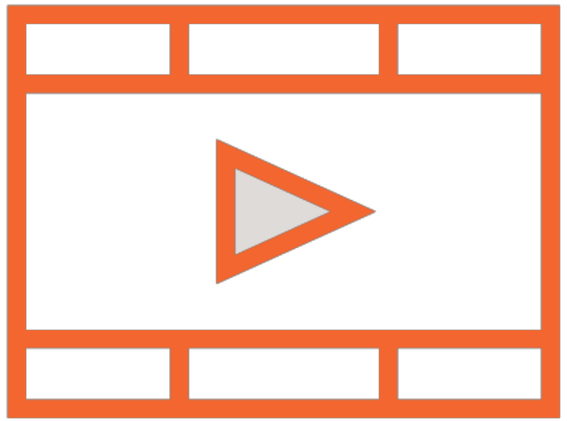
Software and Skills

Basic understanding of cloud computing

Basic understanding of distributed processing

Basic Python programming

Experience with Spark useful



Prerequisites: Basic Cloud Computing

Architecting Big Data Solutions Using Google Dataproc

- Hadoop and Spark on Google Cloud

Architecting Google Cloud Storage Configurations

- Basics of storage on Google Cloud

Course Outline

Introducing Dataflow

- Basic pipeline components
- Execute a pipeline locally and on the cloud

Apache Beam APIs

- Writing Dataflow jobs in Java and Python
- Integrating with other GCP services e.g. BigQuery

Working with collections

- Multiple outputs, branching and side inputs

Using Dataflow templates to create pipelines

- Use pre-defined templates for common Dataflow operations

Scenarios: SpikeySales.com

Hypothetical online retailer

- Flash sales of trending products
- Spikes in user traffic

SpikeySales on the GCP

- Cloud computing fits perfectly
- Pay-as-you-go
- No idle capacity during off-sale periods

SpikeySales and Cloud Storage Buckets

- Move data to the cloud
- Elastic, pay-as-you-go, global access

Use Cases: SpikeySales.com

Large on-premise Hadoop cluster

- Utilization concerns
- Administration of cluster is onerous

Scaling big data processing

- On-premise Hadoop cluster expensive to scale

Spikes in workloads

- Ordinary use: Cluster mostly idle
- Sale Days: Cluster struggles to keep pace

Introducing Google Cloud Dataflow

Use Cases: SpikeySales.com

Large on-premise Hadoop cluster

- Utilization concerns
- Administration of cluster is onerous

Scaling big data processing

- On-premise Hadoop cluster expensive to scale

Spikes in workloads

- Ordinary use: Cluster mostly idle
- Sale Days: Cluster struggles to keep pace

Complex Data Processing

Day-1 Solution

Dataproc

Hadoop, Spark, Pig code as-is

Day-365 Solution

Dataflow

Re-code to Apache Beam

Cloud Dataproc is a fully managed service for running MapReduce, Spark, Pig and Hive on the GCP

Hadoop vs. Dataproc

Traditional On-premise Hadoop

Clusters stay in existence forever

HDFS runs on cluster nodes

Data for jobs stored in HDFS

Cluster encapsulates state

Managed Hadoop with Dataproc

Clusters created and used dynamically

HDFS runs on persistent disks of VMs

Data ideally in GCS buckets

Cluster ideally stateless

Cloud **Dataflow** is a **serverless** approach for data transformation of batch as well as streaming data

Dataproc vs. Dataflow

Dataproc	Dataflow
Still requires cluster provisioning	Serverless, no clusters provisioned
Manual (but managed) scaling	Autoscaling
Hadoop ecosystem	Apache Beam API
Inherently batch-oriented (can work with Spark Streaming)	Integrates batch and streaming (with Pub/Sub and windowing)
Spark ML	TensorFlow with Cloud ML Engine
Separation of compute and storage	Separation of compute and storage

Demo

Enable APIs to work with Dataflow

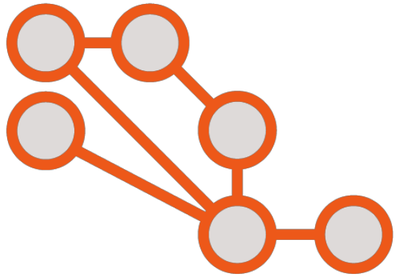
Installing client libraries for Python

Using Dataflow

Write code for pipeline

Submit job for execution

Dataflow assigns workers to execute



Pipeline parallelized and executed

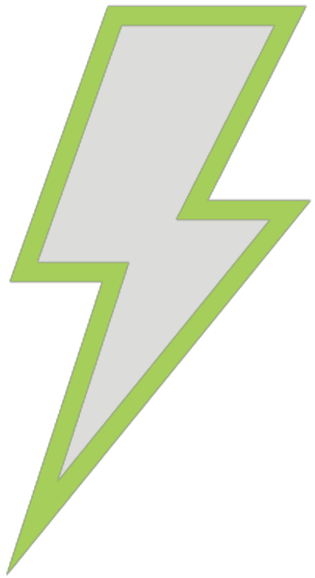
Apache Beam APIs

Code conforms to Apache Beam API

Python or Java

Code looks similar to Spark

https://beam.apache.org/documentation/programming-guide/



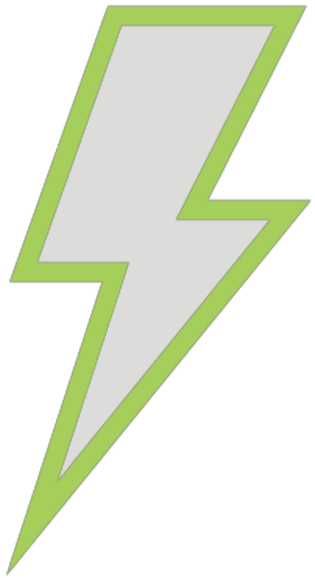
Dataflow Runner

Submit job for execution

Specify choice of runner (execution backend)

- Direct runners: executes pipelines on your machine
- Spark runners: Executes on a Spark cluster
- Cloud Dataflow runners: Run on Dataflow

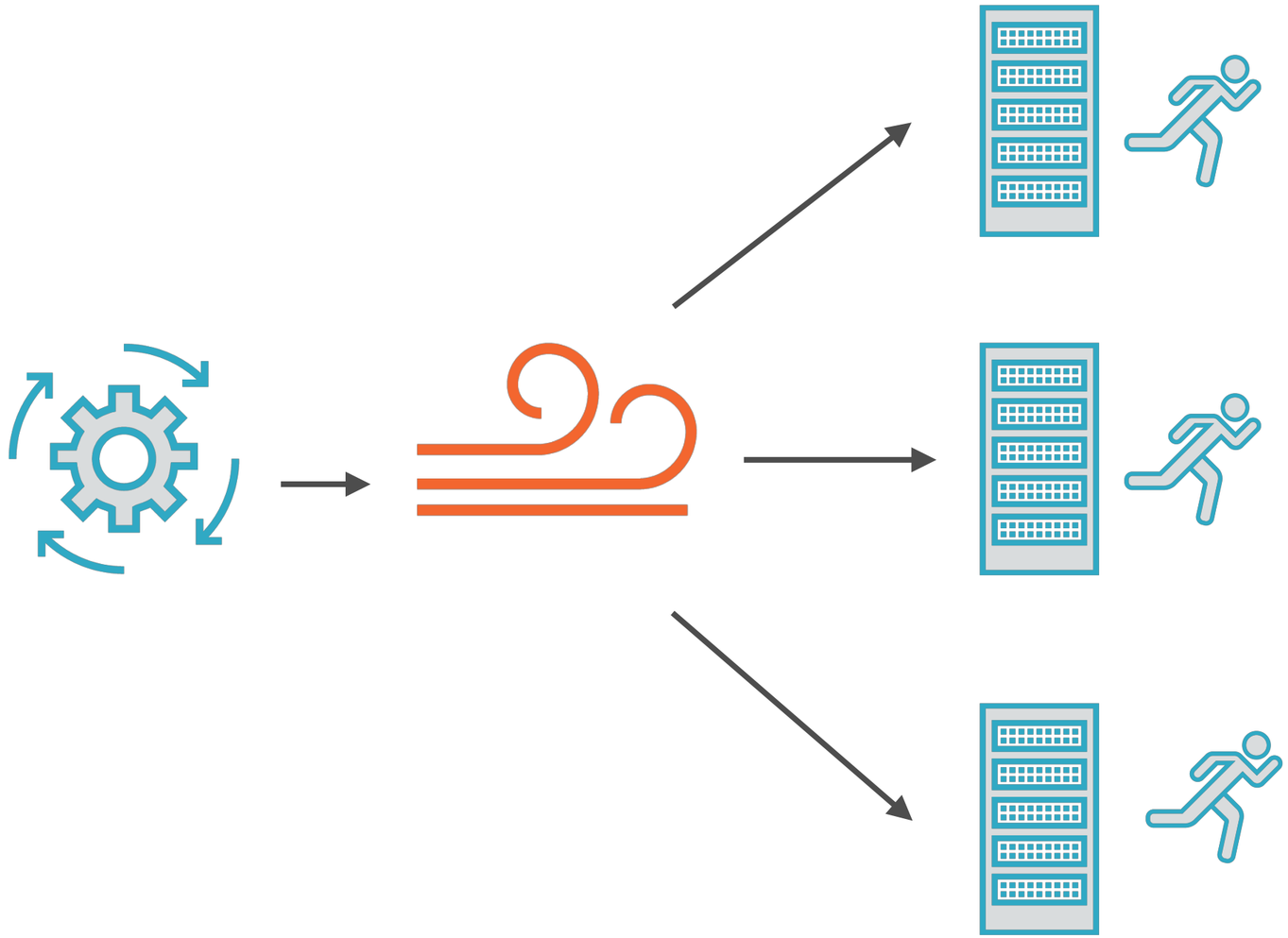
Dataflow Runner



Streaming jobs run indefinitely

Monitor/visualize job pipeline

Execution Using Workers



Execution Using Workers

Serverless and no-ops

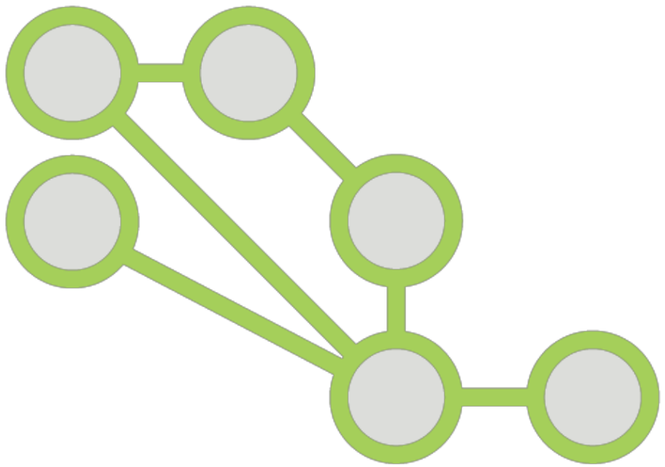
- GCP automatically assigns resources needed to run job

Autoscaling

- GCP automatically adds or removes capacity based on workload

Parallelized

- Operations in code are executed in parallel by platform



Pipelines

Pipeline as DAG

- Directed-acyclic-graph

Collections

- Edges are unbounded collections called PCollections

Transforms

- Apache Beam operations to modify data

Apache Beam

Open source, unified model
for batch and streaming data

Build pipelines which define
transformations on data, these can
be executed on different backends

Apache Beam Pipeline Components

Data source

Batch or streaming data to be processed

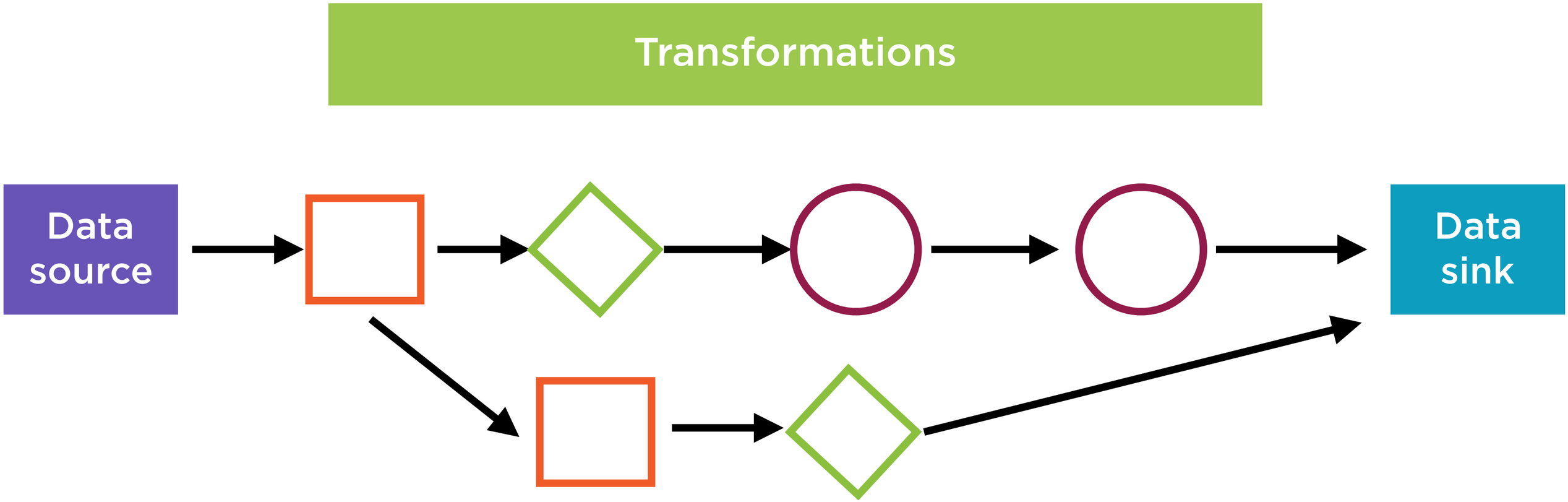
Transformations

Modify the data to get it in the right final form

Data sink

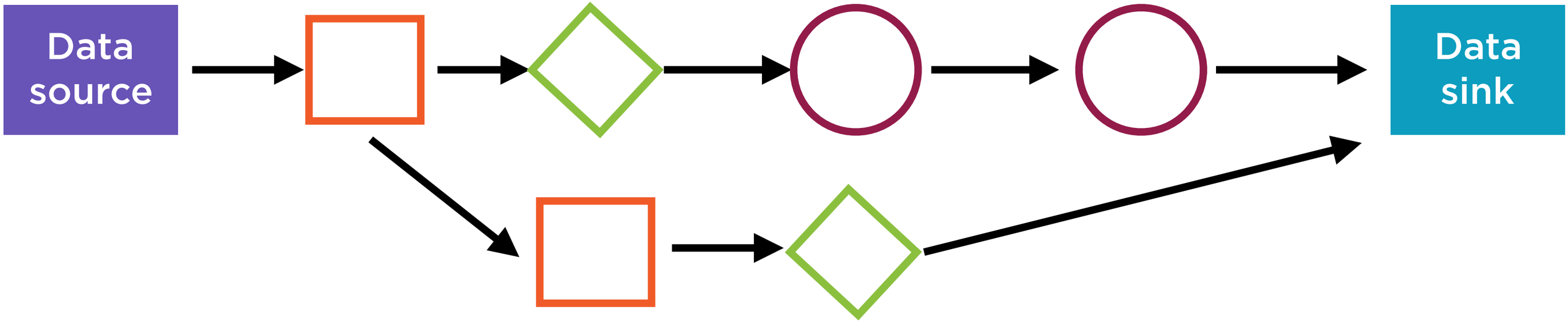
Store the data in some kind of persistent storage

Apache Beam Architecture



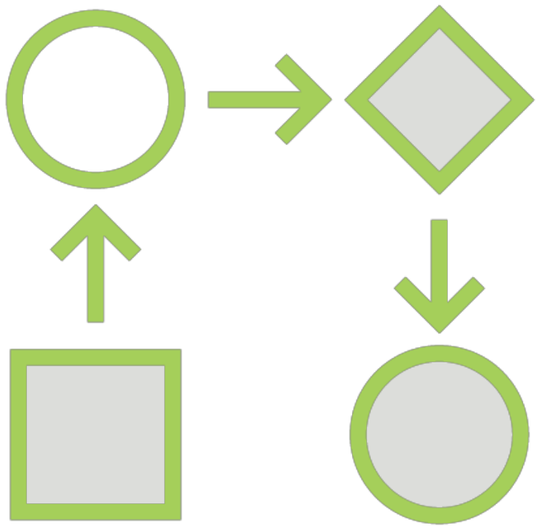
Directed Acyclic Graph (DAG)

Apache Beam Architecture



Pipeline: Entire set of computations

Pipeline



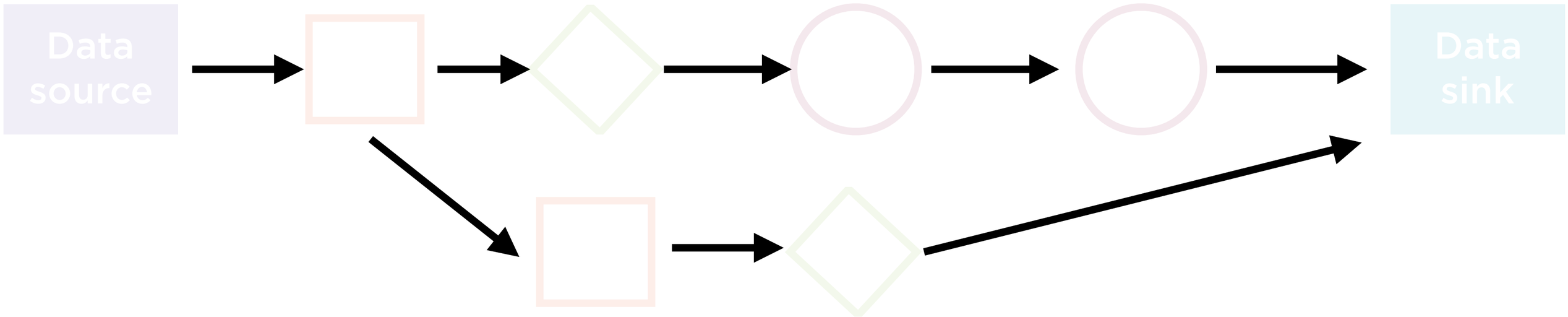
Single, potentially repeatable job

Executes from start to finish

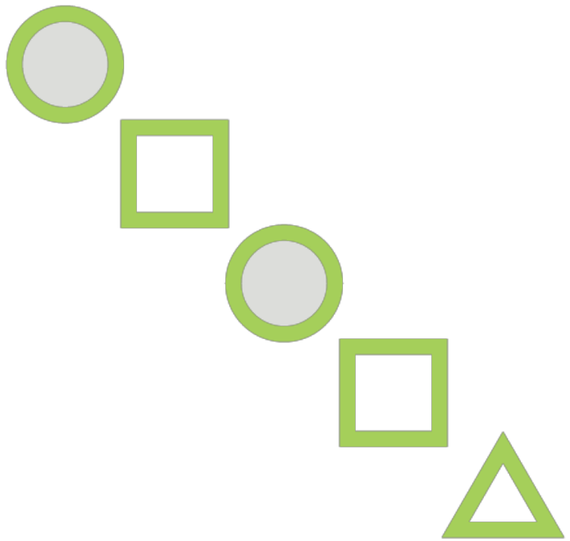
Applies transformations to the data

May have operations which can be performed in parallel

Apache Beam Architecture



PCollections: Edges of DAG



PCollections

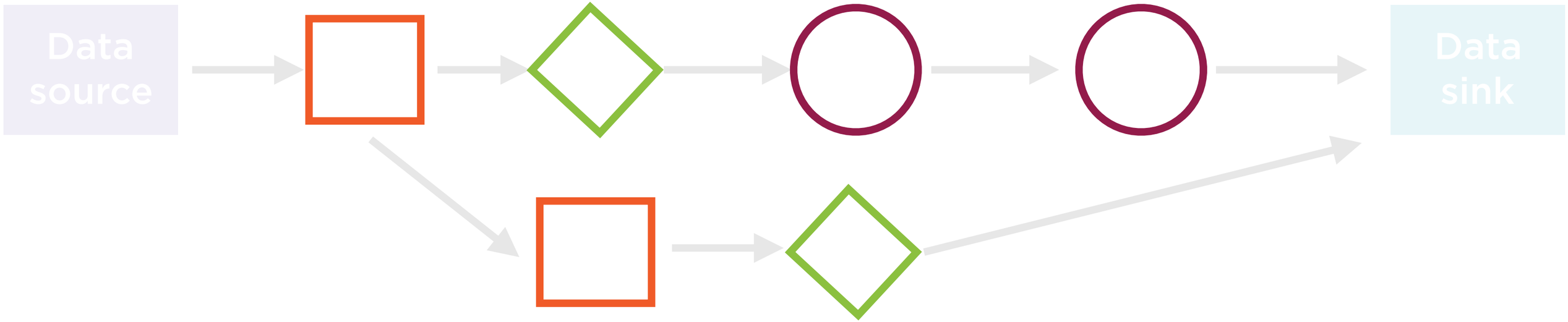
Specialized container classes

Can represent data sets of virtually unlimited size

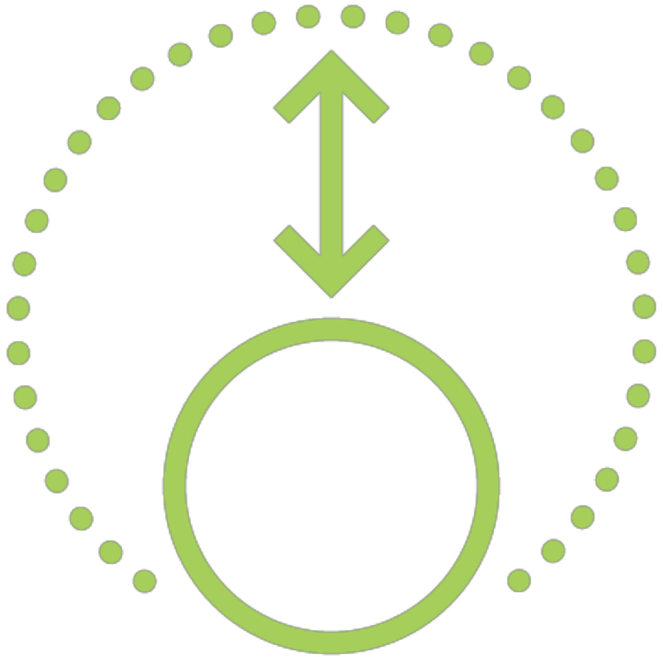
Fixed size: text file or BigQuery table

Unbounded: Pub/Sub subscription

Apache Beam Architecture



Transforms: Nodes in DAG

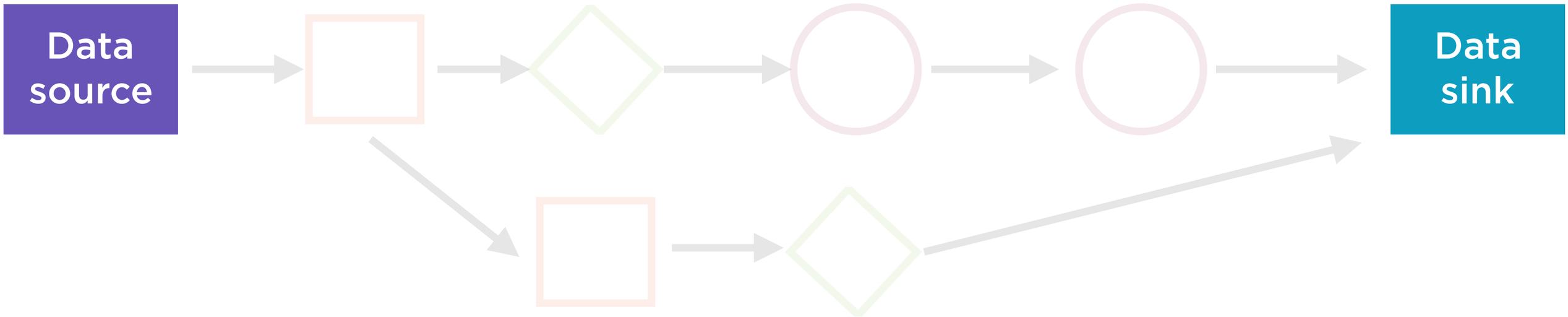


Transforms

Code which modifies elements in a PCollection

- ParDo: Generic parallel processing
- GroupByKey: Process key-value pairs
- Combine: Aggregate elements
- many others

Apache Beam Architecture



Cloud Storage bucket, BigQuery table, Pub/Sub subscription

Driver and Runner

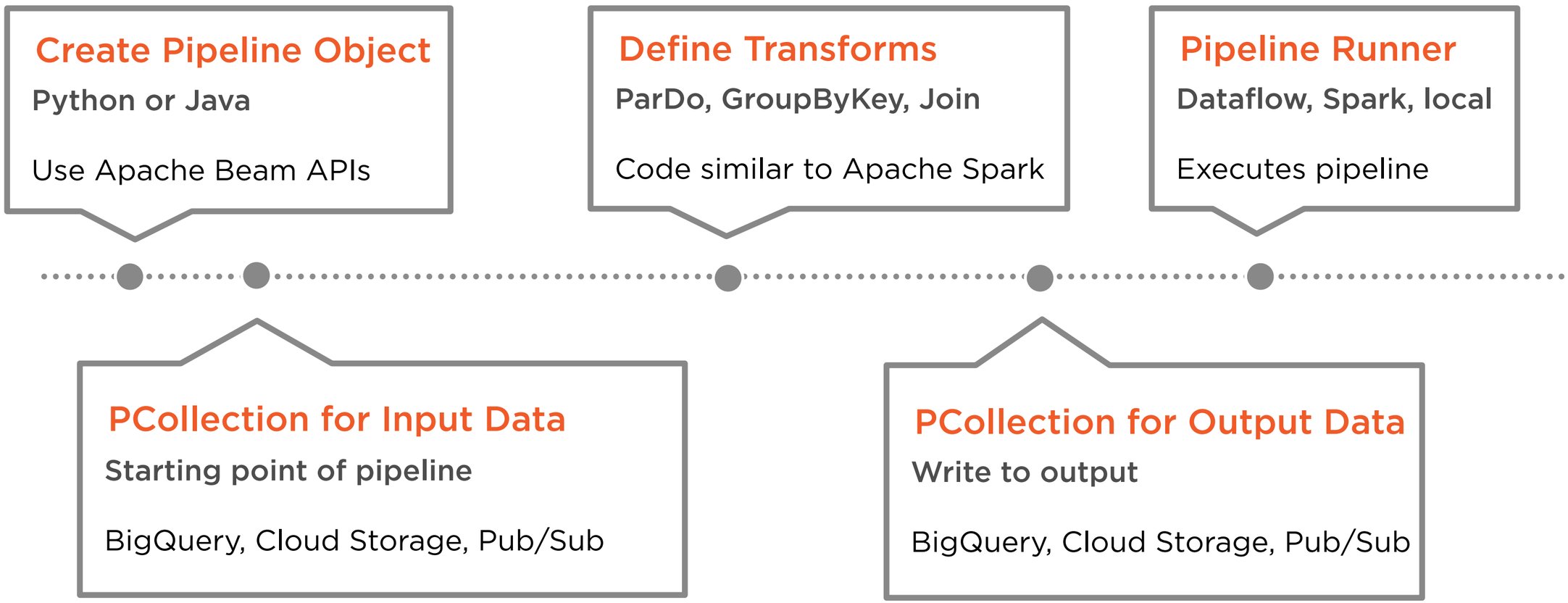
Driver defines computation DAG (pipeline)

Runner executes DAG on a backend

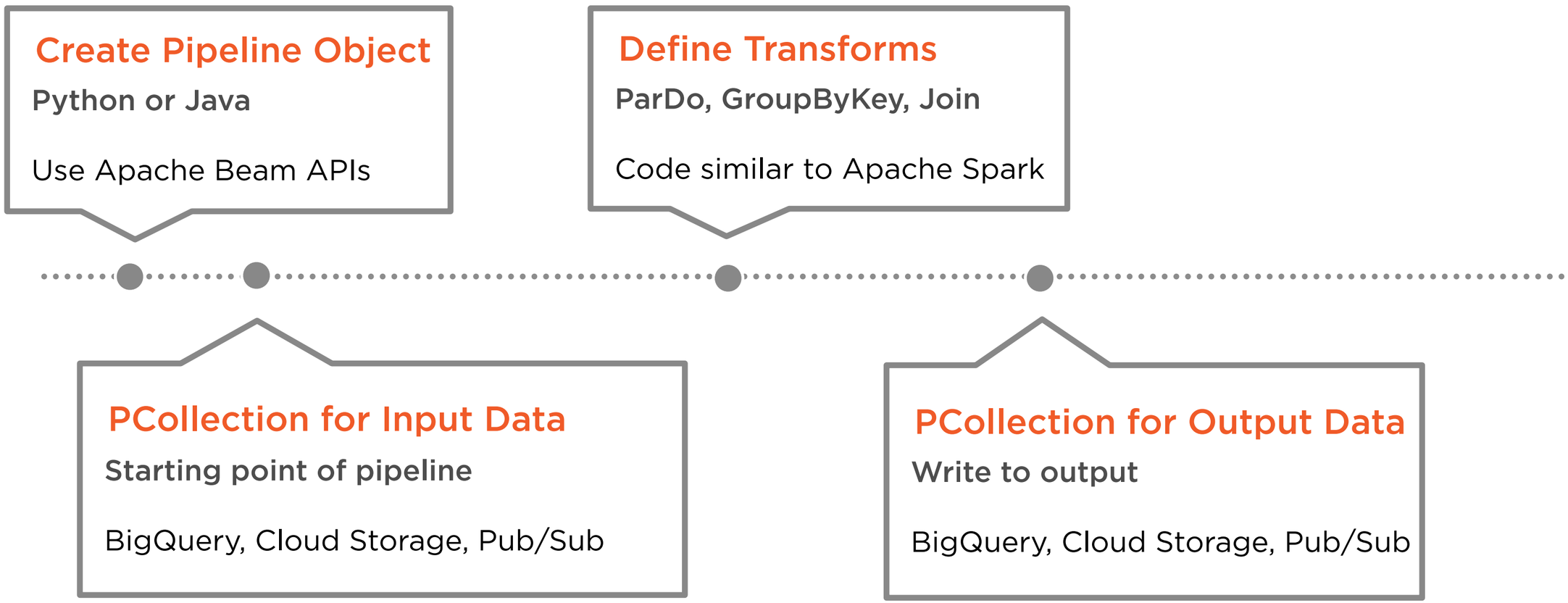
Apache Beam works with

- 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

Demo

Running a simple Dataflow job on the local machine

Demo

Running a simple Dataflow job on the Google Cloud

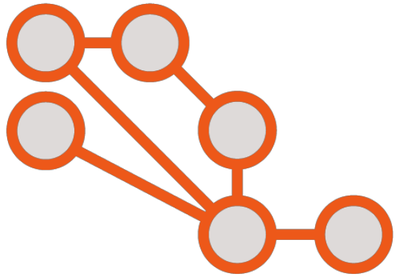
Dataflow Pricing

Using Dataflow

Write code for pipeline

Submit job for execution

Dataflow assigns workers to execute



Pipeline parallelized and executed

Using Dataflow

Write code for pipeline

Submit job for execution

Dataflow assigns workers to execute

Pipeline parallelized and executed

Total Price = Compute Cost + Storage Cost + Data Processing Cost

Cloud Dataflow Pricing

Three components of pricing - based on workers used in batch or stream processing

Total Price = **Compute Cost** + Storage Cost + Data Processing Cost

Compute Cost

Per-second billing of vCPUs and memory used in batch or streaming workers

Total Price = **Compute Cost** + Storage Cost + Data Processing Cost

Compute Cost

Standard worker configurations for batch: 1 vCPU, 3.75GB memory and 250 GB Persistent Disk

Total Price = **Compute Cost** + Storage Cost + Data Processing Cost

Compute Cost

Standard worker configurations for streaming: 4 vCPU, 15GB memory and 420 GB Persistent Disk

Total Price = **Compute Cost** + Storage Cost + Data Processing Cost

Compute Cost: vCPU

Batch: $0.056 /vCPU/hour; Streaming: $0.069 /vCPU/hour

Total Price = **Compute Cost** + Storage Cost + Data Processing Cost

Compute Cost: Memory

Batch: $0.003557 /GB/hour; Streaming: $0.003557 /GB/hour

Total Price = Compute Cost + **Storage Cost** + Data Processing Cost

Storage Cost: Standard Persistent Disk

Batch: $0.000054 /GB/hour; Streaming: $0.000054 /GB/hour

Total Price = Compute Cost + **Storage Cost** + Data Processing Cost

Storage Cost: SSD Persistent Disk

Batch: $0.000298 /GB/hour; Streaming: $0.000298 /GB/hour

Using Dataflow

Write code for pipeline

Submit job for execution

Dataflow assigns workers to execute

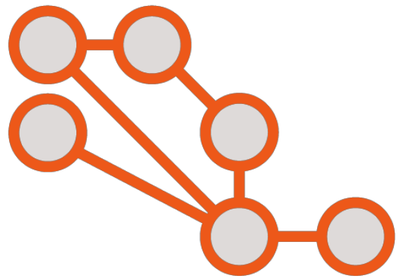
Pipeline parallelized and executed

Using Dataflow

Write code for pipeline

Submit job for execution

Dataflow assigns workers to execute



Pipeline parallelized and executed

Total Price = Compute Cost + Storage Cost + **Data Processing Cost**

Data Processing Cost

Batch: $0.011 /GB; Streaming: $0.018 /GB

Demo

Monitoring Dataflow jobs using Stackdriver

Summary

Serverless and fully managed data transformation service

Pipelines for batch and streaming data

Conforms to Apache Beam API

Some similarities to Apache Spark

Separates storage and compute