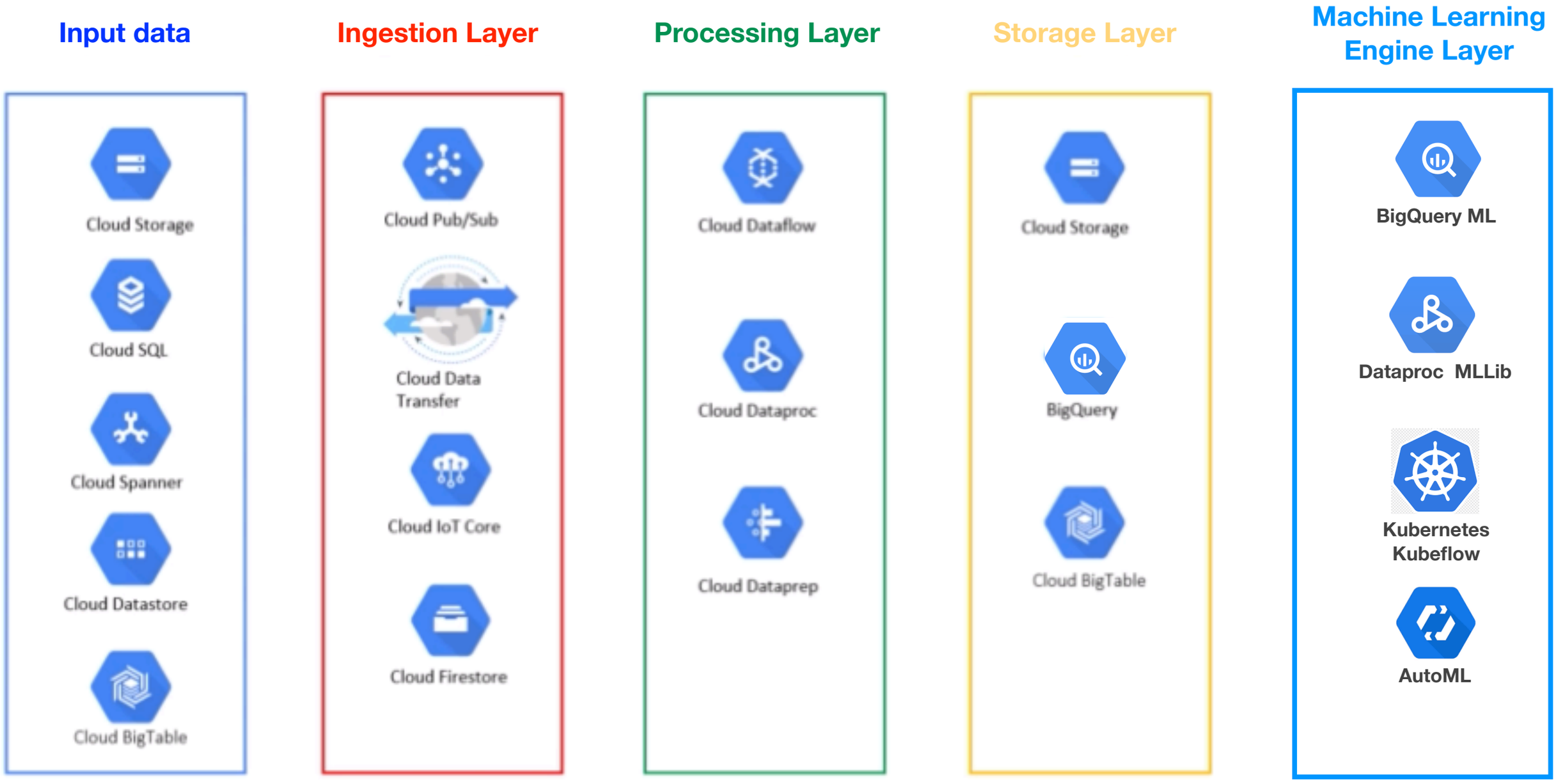
Machine Learning/ Data Ingestion

Dataflow (Apache Beam)
Dataproc (Spark + Hadoop)

Machine Learning/ Big Data Ecosystem



Data ingestion - Best practicies 1/3

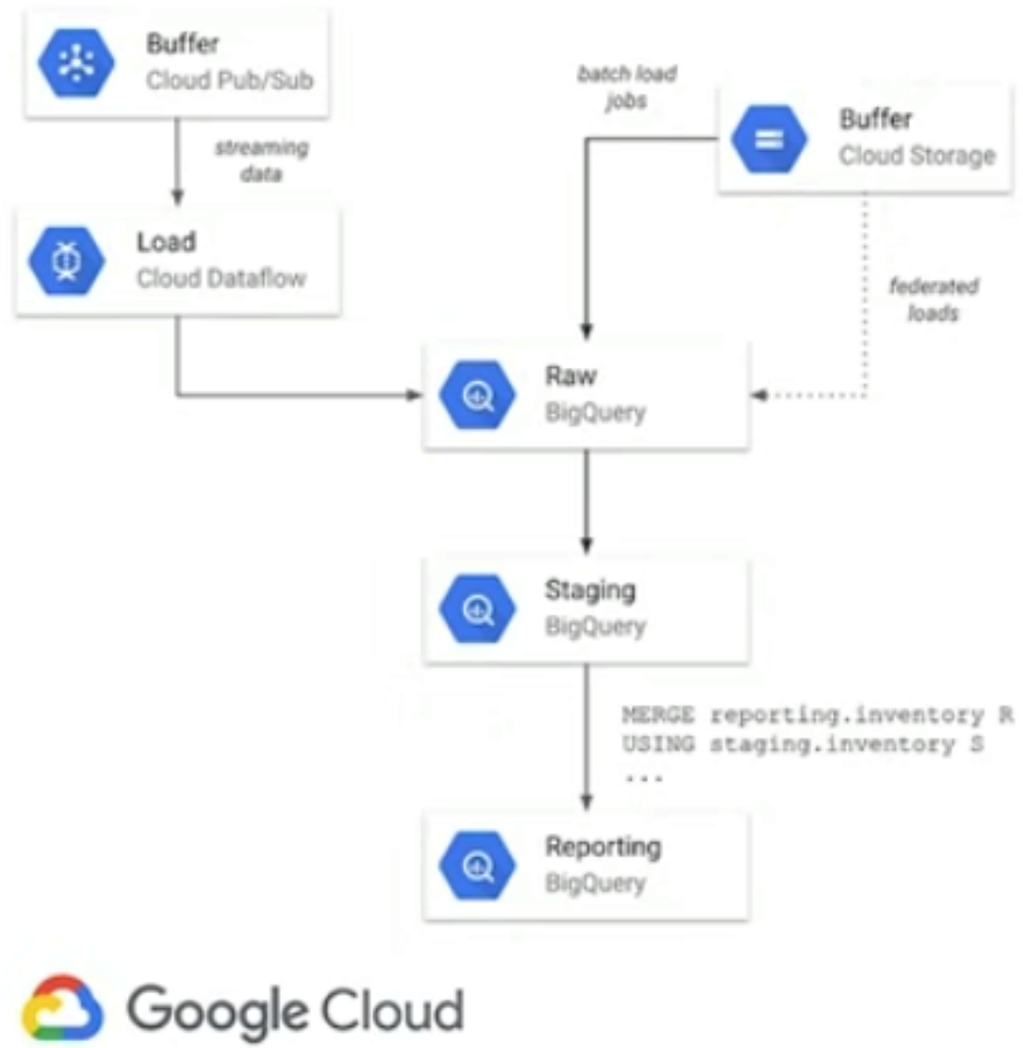
Data source format - from faster to slower

- **AVRO (compressed) - the fastest**
  - Avro (uncompressed)
  - Parquet /ORC
  - CSV
  - JSON
  - CSV (compressed)
- **JSON (compressed) - the slowest**

Source: Data Warehousing With BigQuery: Best Practices (Cloud Next '19)
Link: https://www.youtube.com/watch?v=ZVgt1-LfWW4

Data ingestion - Best practicies 2/3

Best Practice: ELT / ETL



Prefer ELT into BigQuery over ETL where possible

Leverage federated queries to GCS to load and transform data in a single-step

Load data into raw and staging tables before publishing to reporting tables

Utilize Dataflow or Data Fusion for streaming pipelines and to speed up large complex batch jobs

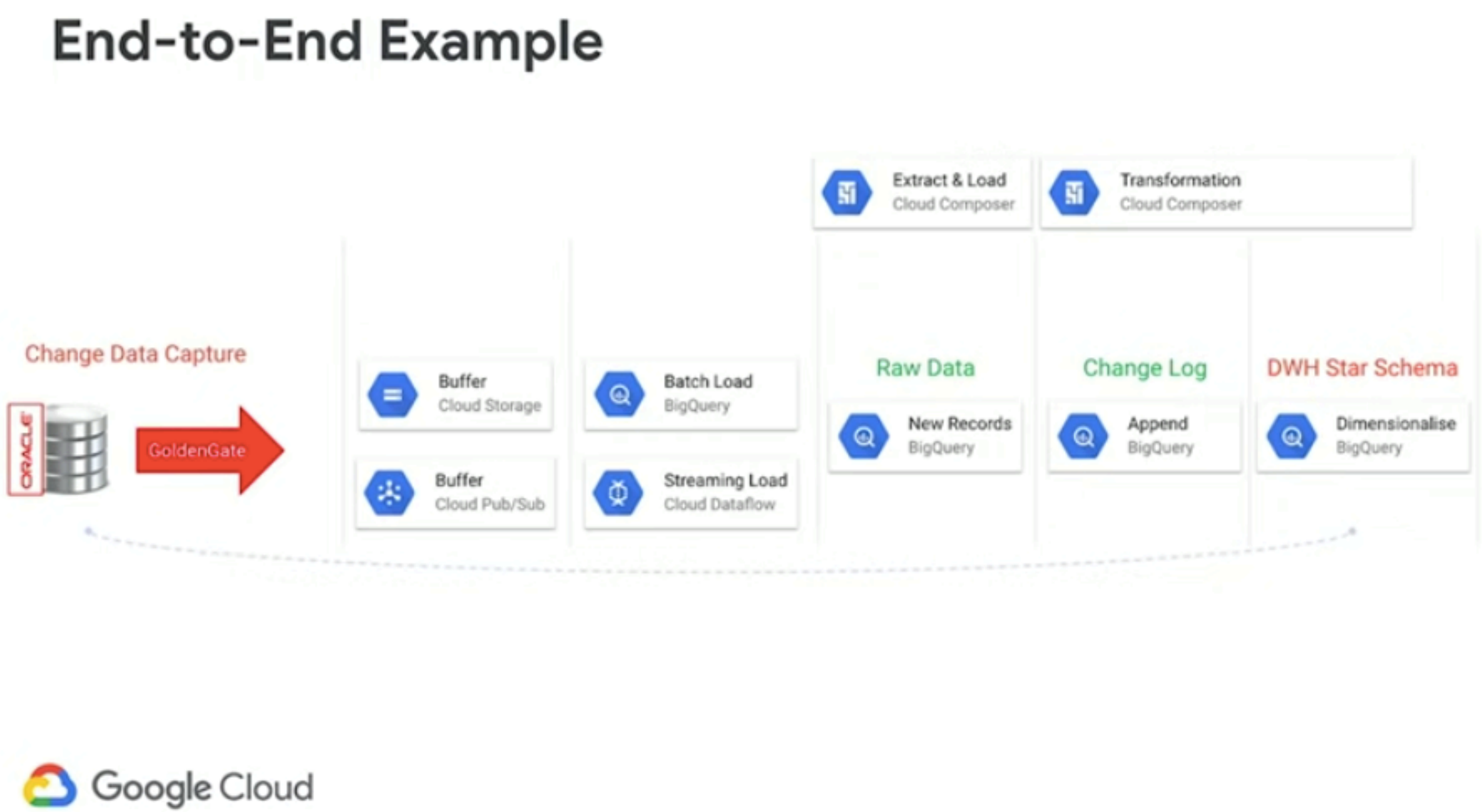
Get started streaming using Google-Provided Dataflow Templates and modify the open-source pipeline for more complex needs

Source: Data Warehousing With BigQuery: Best Practices (Cloud Next '19)
Link: https://www.youtube.com/watch?v=ZVgt1-LfWW4

Managing batch data ingestion process

1. Load the data source files (sometimes called feeds) into Google Cloud Storage. GCS can be seen as Data Lake component. As a principle use:
    - **one batch = one file in GCS = one raw-data table**.
2. Next load the files into BigQuery Raw Data area using Dataflow.
    - Use simple checks (Filters or Maps) to pass proper records only or augment the default parameters. This is the preliminary step of data cleansing.
    - You can agregate the data, join data from multiple sources and some other transformation (eg. upper to lower case) as well.
3. Transform data from Raw Data into Staging Area tables using SQL language to easily transform data, join data from multiple raw data sources, calculate some missing parameters (eg. agregation) etc.
4. Staging Area are - so called - “golden source of information” for reporting or machine-learning or any other purposes.
5. Finally, transform data from Staging Area into reporting format (eg. Star Schema “Kimball`s like”), ML models, Tensorflow matrix etc.
6. Steps 4 and 5 can be split into several subtasks if needed; it can be easily managed by using Cloud Composer service.

Data ingestion - Best practicies 3/3



Source: Data Warehousing With BigQuery: Best Practices (Cloud Next '19)
Link: https://www.youtube.com/watch?v=ZVgt1-LfWW4

Dataflow GCP service

Apache Beam + Python

Dataflow Cloud Service Overview

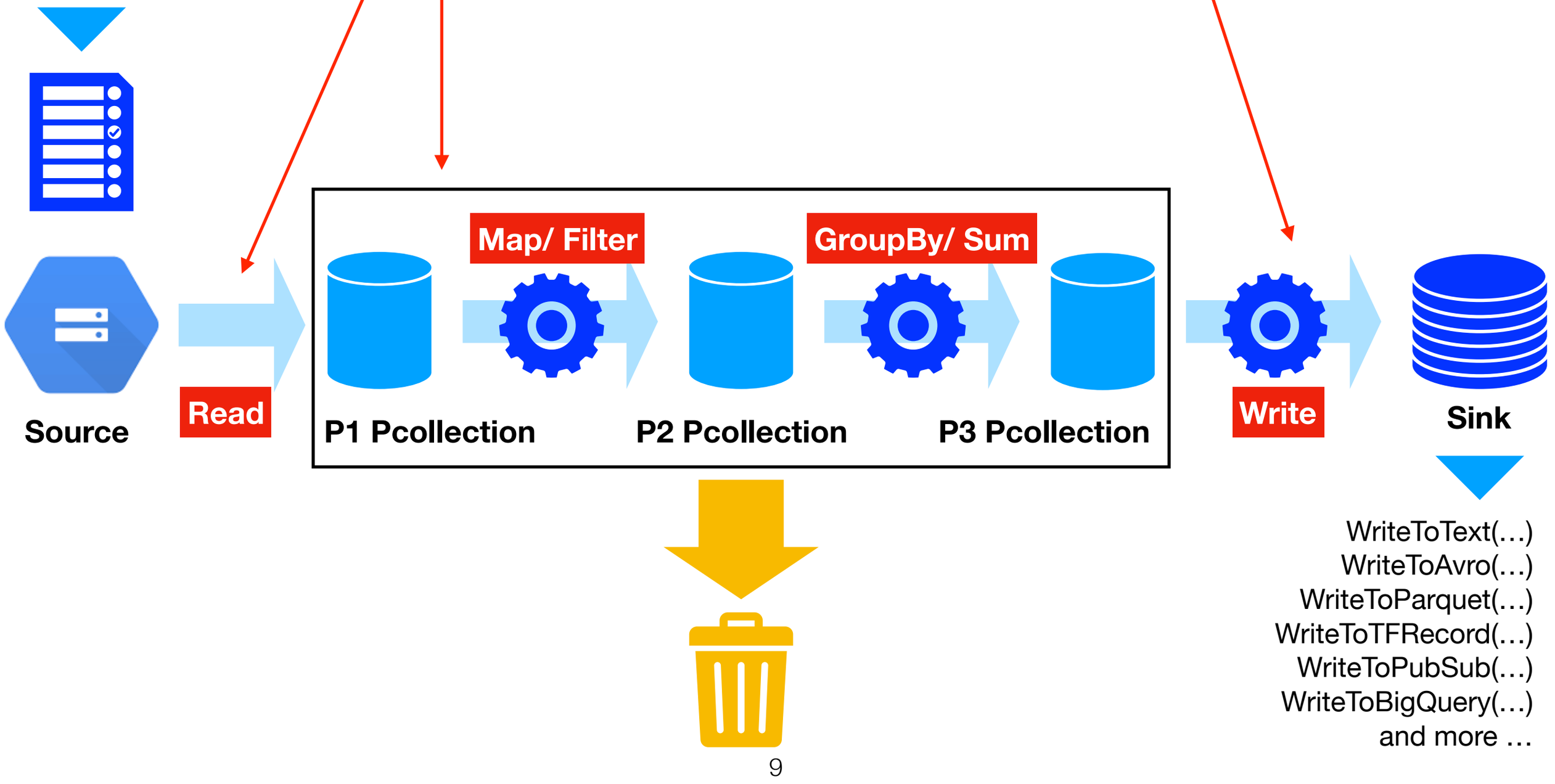
- Dataflow is a managed service for executing a wide variety of data processing patterns, built up on the Apache Beam project.
- The Apache Beam is an open source programming model that enables you to develop both batch and streaming pipelines. You create your pipelines with an Apache Beam program and then run them on the Dataflow service. The Apache Beam documentation provides in-depth conceptual information and reference material for the Apache Beam programming model, SDKs, and other runners.
- Dataflow cloud service key features:
  - enables batch and streaming processing (data pipeline),
  - simplifies operations and management - allows teams to focus on programming (Python, Java and Go) instead of managing server clusters,
  - provides serverless approach removes operational overhead from data engineering workloads,
  - automates resource management and dynamic work rebalancing,
  - flexible resource scheduling pricing for batch processing.

Apache Beam documentation ref: *https://beam.apache.org/documentation/*

Apache Beam Pipeline - 5 Key Steps

ReadFromText(…)
ReadFromAvro(…)
ReadFromParquet(…)
ReadFromTFRecord(…)
ReadFromPubSub(…)
and more ….

Step 1 Creating and giving pipeline a name
Step 2 Initial Pcollection by reading data from a source
Step 3 Ptransformation according to requirements
Step 4 Write Pcollection to a sink
Step 5 Run the pipeline



WriteToText(…)
WriteToAvro(…)
WriteToParquet(…)
WriteToTFRecord(…)
WriteToPubSub(…)
WriteToBigQuery(…)
and more …

Apache Beam Pipeline - Basic Template

File: ./dataflow-wordcount-pipeline/word-count.py

```
# step 1 Creating and giving pipeline a name
p1 = beam.Pipeline()

attendance_count = (

    p1
    # Step2 Initial Pcollection by reading data from a source
    | 'Read lines' >> beam.io.ReadFromText(INPUT_PATTERN)
    # Step 3 Ptransformation according to requirements
    | 'Find words' >> beam.FlatMap(SplitRow)
    | 'Pair words with counter' >> beam.Map(lambda element: (element, 1))
    | 'Group and sum' >> beam.CombinePerKey(sum)
    | 'Format results' >> beam.Map(lambda word_count: str(word_count))
    # Step 4 Write Pcollection to a sink
    | 'Write results' >> beam.io.WriteToText(OUTPUT_PREFIX)
)

# Step 5 Run the pipeline
p1.run()
```

Put your Python transformation code, here.

Dataflow boilerplate code see: ./dataflow-wordcount-pipeline/dataflow-boilerplate.py

Prepared Labs

- Lab 1: Direct batch loading into BigQuery table
  - good start to load training data and qiuckly start testing ML algorythms
- Lab 2: Using Dataflow Templates (wordcount)
  - how to use pre-built templates, provided by google
- Lab 3: Writing a simple Dataflow pipeline (Python)
  - diving in simple pipeline written in Python, which provides the same wordcounting as template used in Lab2

- Lab 4: Writing ouput to BigQuery table
- Lab 5: Using DLP (Data Lost Pretection) to secure PII data
- Lab 6: Stream processing using Pub/Sub service

Comming soon

Dataproc GCP service

Hadoop + Spark + Python

Dataproc Cloud Service Overview

- Managed Spark and Hadoop cloud service
- Used for large-sacle batch processing and Machine Learning
- Supports stream processing as well
- Used for both ELT and ETL (typicaly ELT is prefered)
- Ephemeral cluster (ie. Hadoop cluster in fact)

To be continued