Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

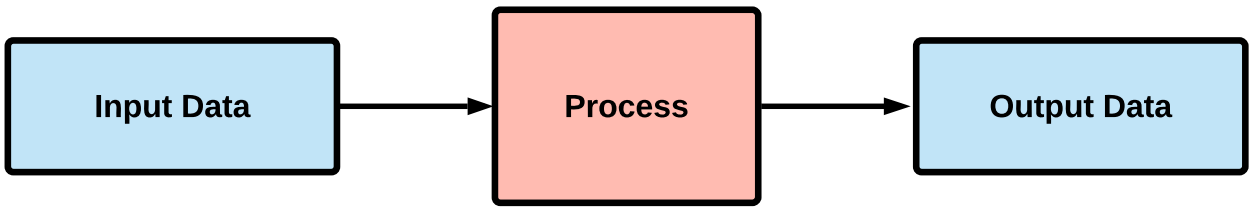
Data Processing Challenges

Next

What is Data Processing?

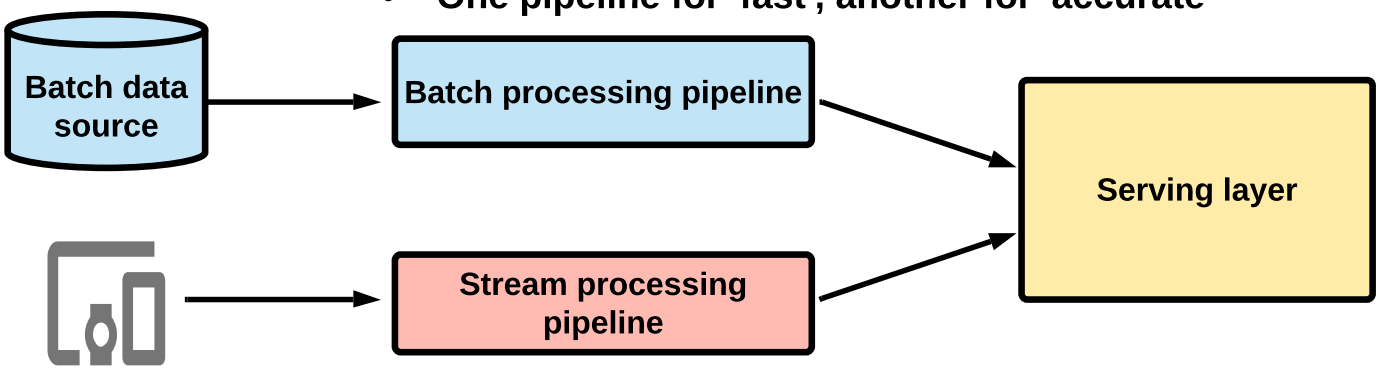
- Read Data (Input)
- Transform it to be relevant - Extract, Transform, and Load (ETL)
- Create output

to our particular interest



Challenge: Streaming and Batch data pipelines:

- Until recently, separate pipelines are required for each
- Difficult to compare recent and historical data
- One pipeline for 'fast', another for 'accurate'



Why both?

- Credit card monitoring
- Compare streaming transactions to historical batch data to detect fraud

Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Data Processing Challenges

Previous

Challenge: Complex element processing:

- Element = single data input
- One at a time element ingest from single source = easy
- Combining elements (aggregation) = hard
- Processing data from different sources, streaming, and out of order (composite) = REALLY hard

Solution: Apache Beam + Cloud Dataflow

create batch and data processing pipeline

Cloud Dataflow

execute pipelines

Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

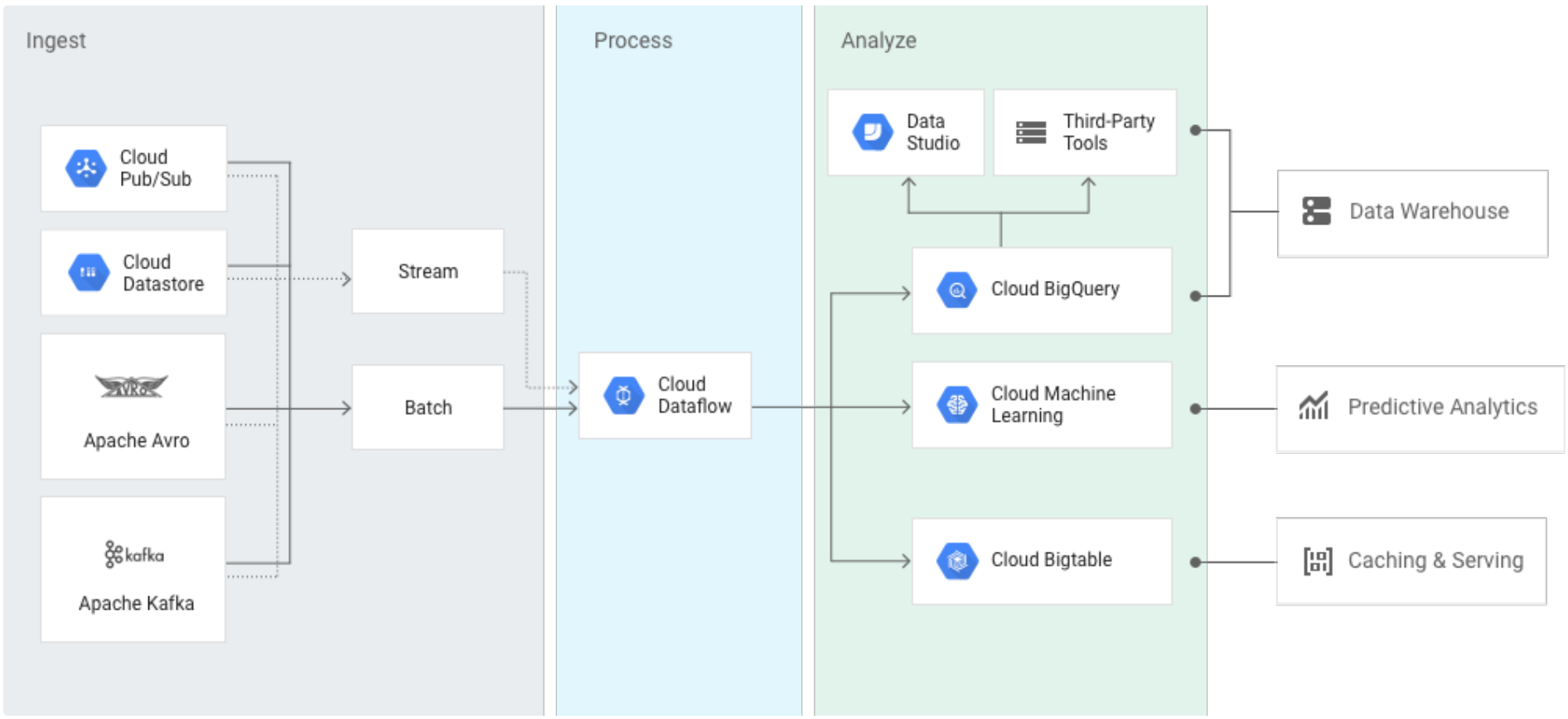
Cloud Dataflow Overview

Next

What is it?

- Auto-scaling, No-Ops, Stream, and Batch Processing
- Built on Apache Beam:
  - Documentation refers to Apache Beam site
  - Configuration is 100% code-based
- Integrates with other tools (GCP and external):
  - Natively - Pub/Sub, BigQuery, Cloud ML Engine
  - Connectors - Bigtable, Apache Kafka
- Pipelines are regional-based

Big Picture - Data Transformation



Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Cloud Dataflow Overview

Previous

Next

IAM:

- Project-level only - all pipelines in the project (or none)
- Pipeline data access separate from pipeline access
- Dataflow Admin - Full pipeline access plus machine type/storage bucket config access
- Dataflow Developer - Full pipeline access, no machine type/storage bucket access
- Dataflow Viewer - view permissions only
- Dataflow Worker - Specifically for service accounts

Dataflow vs Dataproc?

Beam vs. Hadoop/Spark?

Dataproc:

- Familiar tools/packages
- Employee skill sets
- Existing pipelines

Dataflow:

- Less Overhead
- Unified batch and stream processing
- Pipeline portability across Dataflow, Spark, and Flink as runtimes

WORKLOADS	CLOUD DATAPROC	CLOUD DATAFLOW
Stream processing (ETL)		X
Batch processing (ETL)	X	X
Iterative processing and notebooks	X	
Machine learning with Spark ML	X	
Preprocessing for machine learning		X (with Cloud ML Engine)

Dataproc can not do!

Return to Table of Contents

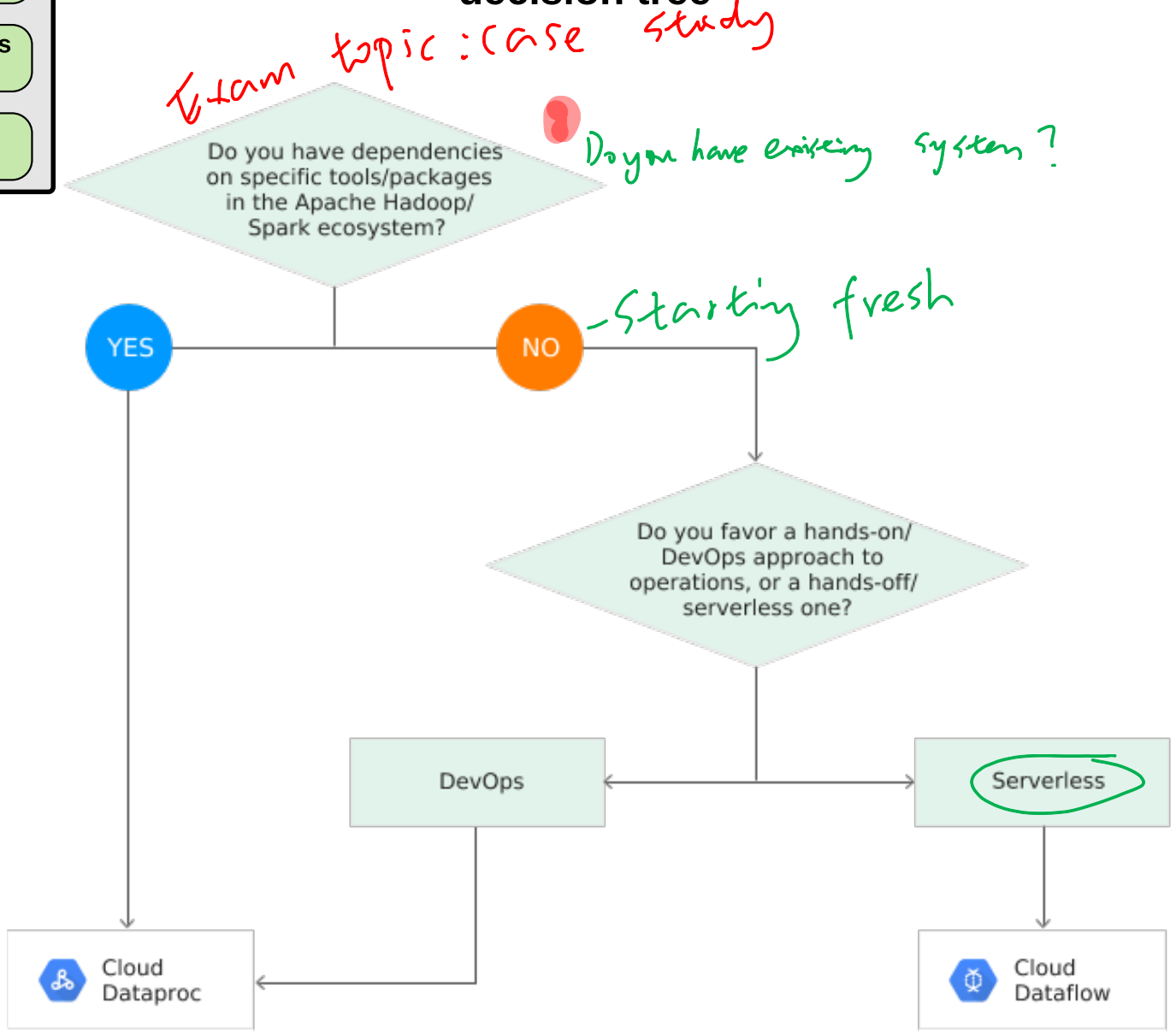
Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Cloud Dataflow Overview

Previous

Dataflow vs. Dataproc decision tree



Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Key Concepts

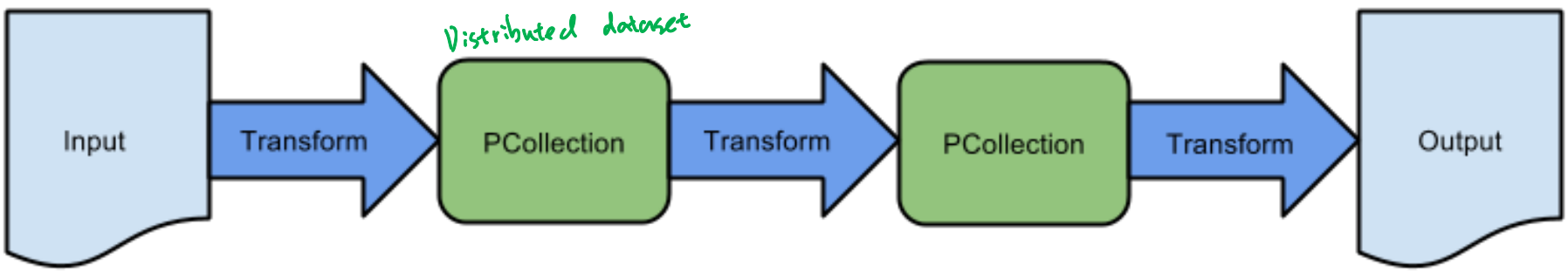
Next

Course/exam perspective:

- Dataflow is very code-heavy
- Exam does not go deep into coding questions
- Some key concepts/terminology will be tested

Key terms: (exam)

- Element - single entry of data (e.g., table row)
- PCollection - Distributed data set, data input and output
- Transform - Data processing operation (or step) in pipeline:
  - Uses programming conditionals (for/while loops, etc.)
- ParDo - Type of transform applied to individual elements:
  - Filter out/extract elements from a large group of data



PCollection and ParDo in example Java code.

One step in a multi-step transformation process.

```
PCollection<LaneInfo> currentConditions = p //
    .apply("GetMessages", PubsubIO.readStrings().fromTopic(topic)) //
    .apply("ExtractData", ParDo.of(new DoFn<String, LaneInfo>() {
      @ProcessElement
      public void processElement(ProcessContext c) throws Exception {
        String line = c.element();
        c.output(LaneInfo.newLaneInfo(line));
      }
    }));
```

Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Cloud Dataflow Overview

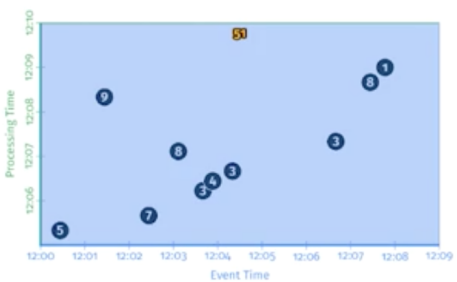
Previous

(streaming data)

Dealing with late/out of order data:

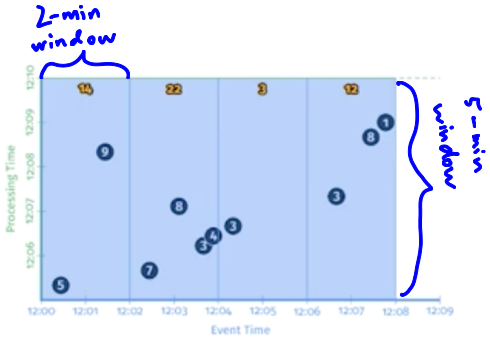
- Latency is to be expected (network latency, processing time, etc.)
- **Pub/Sub does not care about late data, that is resolved in Dataflow**
- Resolved with Windows (2), Watermarks (3), and Triggers (4)
- **Windows** = logically divides element groups by time span
- **Watermarks** = 'timestamp':
  - Event time = when data was generated
  - Processing time = when data processed anywhere in the processing pipeline
  - Can use Pub/Sub-provided watermark or source-generated
- **Trigger** = determine when results in window are emitted (submitted as complete):
  - Allow late-arriving data in allowed time window to re-aggregate previously submitted results
  - Timestamps, element count, combinations of both

watermark types



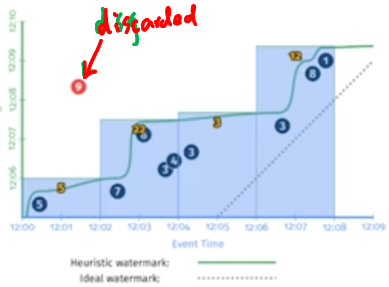
1

Classic Batch



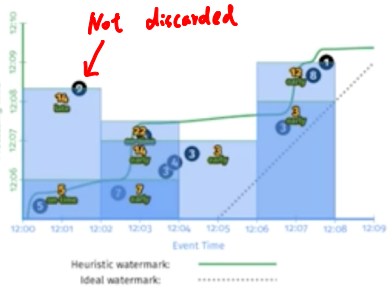
2

Windowed Batch



3

Streaming w/Discarding



4

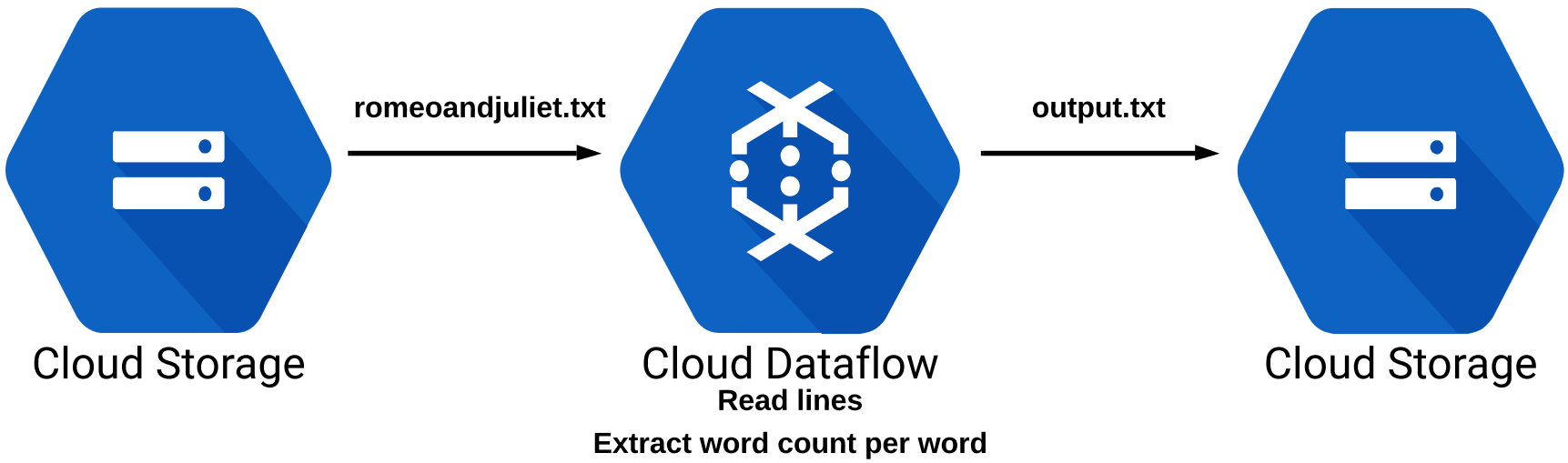
Streaming + Accumulation

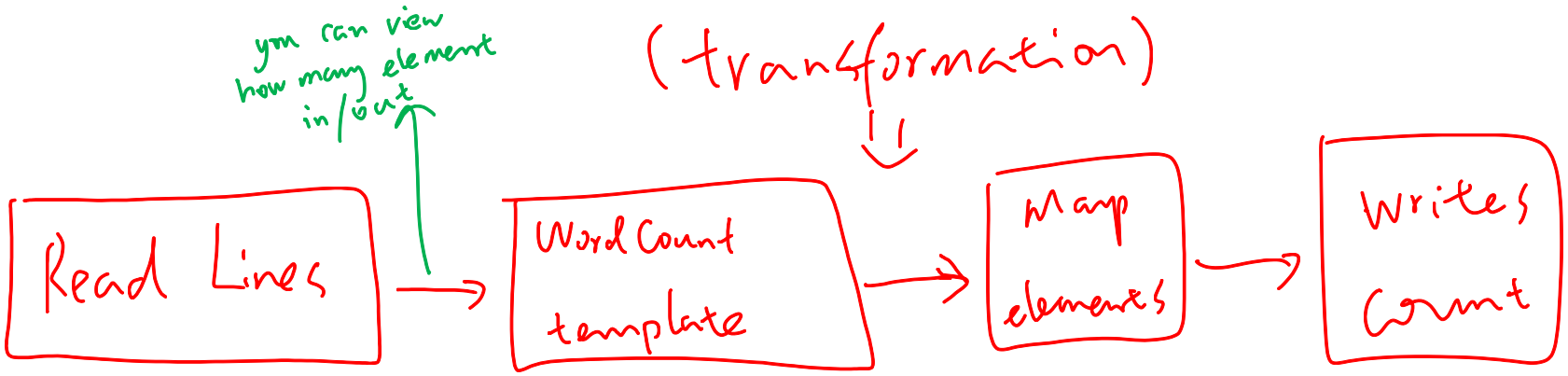
Return to Table of Contents

Choose a Lesson

Template Hands On

- Google-provided templates
- Simple word count extraction





Return to Table of Contents

Choose a Lesson

Streaming Ingest Pipeline Hands On

Next

- Take San Diego traffic data
- Ingest through Pub/Sub
- Process with Dataflow
- Analyze results with BigQuery
- First: Enable Dataflow API from API's and Services

Traffic data

Data ingest

Published streaming sensor data

Topic: sandiego

Subscription pulls messages

Cloud Dataflow

Transform data to calculate average speed.

Output to BigQuery.

BigQuery

Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Streaming Ingest Pipeline Hands On

Previous

Next

Quick command line setup (Cloud Shell)

- Create BigQuery dataset for processing pipeline output:
  - bq mk --dataset $DEVSHELL_PROJECT_ID:demos
- Create Cloud Storage bucket for Dataflow staging:
  - gsutil mb gs://$DEVSHELL_PROJECT_ID
- Create Pub/Sub topic and stream data:
  - cd ~/googledataengineer/courses/streaming/publish
  - gcloud pubsub topics create sandiego
  - ./download_data.sh
  - sudo pip install -U google-cloud-pubsub
  - ./send_sensor_data.py --speedFactor=60 --project=$DEVSHELL_PROJECT_ID

Open a new Cloud Shell tab:

- Execute Dataflow pipeline for calculating average speed:
  - cd ~/googledataengineer/courses/streaming/process/sandiego
  - ./run_oncloud.sh $DEVSHELL_PROJECT_ID $DEVSHELL_PROJECT_ID AverageSpeeds
- Error resolution:
  - Pub/Sub permission denied, re-authenticate
    - gcloud auth application-default login
  - Dataflow workflow failed - enable Dataflow API

Return to Table of Contents

Choose a Lesson

Data Processing Challenges

Cloud Dataflow Overview

Key Concepts

Template Hands On

Streaming Ingest Pipeline Hands On

Additional Best Practices

Streaming Ingest Pipeline Hands On

Previous

View results in BigQuery:

- List first 100 rows:
  - SELECT * FROM [<PROJECTID>:demos.average_speeds] ORDER BY timestamp DESC LIMIT 100
- Show last update to table:
  - SELECT MAX(timestamp) FROM [<PROJECTID>:demos.average_speeds]
- Look at results from the last minute:
  - SELECT * FROM [<PROJECTID>:demos.average_speeds@-60000] ORDER BY timestamp DESC

ms = 1 min

Shut down pipeline:

- Drain - finishing processing buffered jobs before shutting down (Stop accepting new data, and shut down gracefully)
- Cancel - full stop, cancels existing buffered jobs

Return to Table of Contents

Choose a Lesson

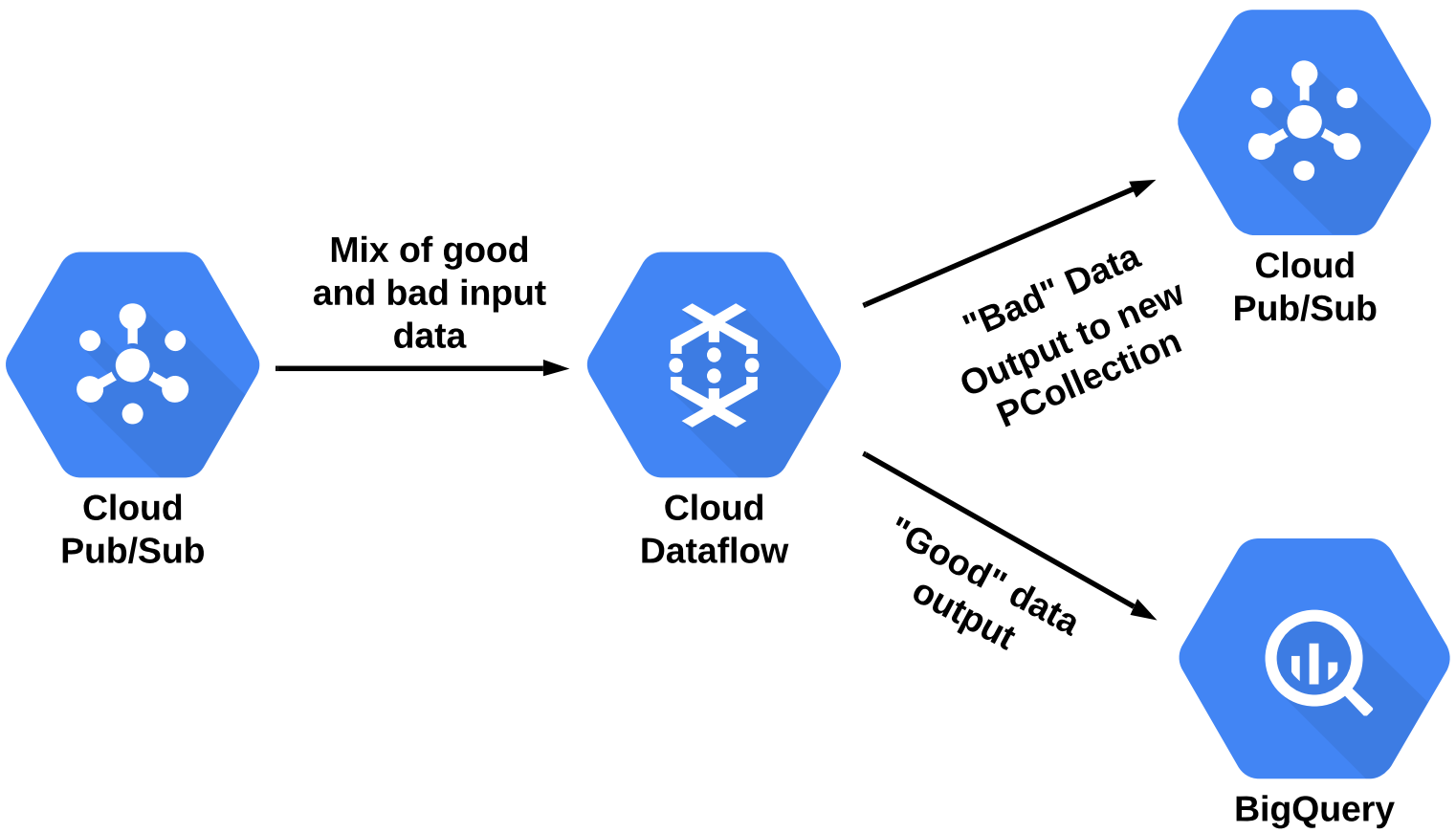
- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Additional Best Practices

Handling Pipeline Errors

Next

- If you do not have a mechanism in place to handle input data errors in your pipeline, the job can fail. How can we account for this?
- Gracefully catch errors:
  - Create separate output: (in Apache)
    - **Try-catch** block handles errors
    - Output errors to new **PCollection** - Send to **collector** for later analysis (Pub/Sub is a good target)
    - Think of it as *recycling* the *bad* data
- Technique is also valid for troubleshooting missing messages:
  - Scenario: Streaming pipeline missing some messages
  - Solution: Run a batch of the streaming data, and check output:
    - Create additional output to capturing and processing error data.



Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Additional Best Practices

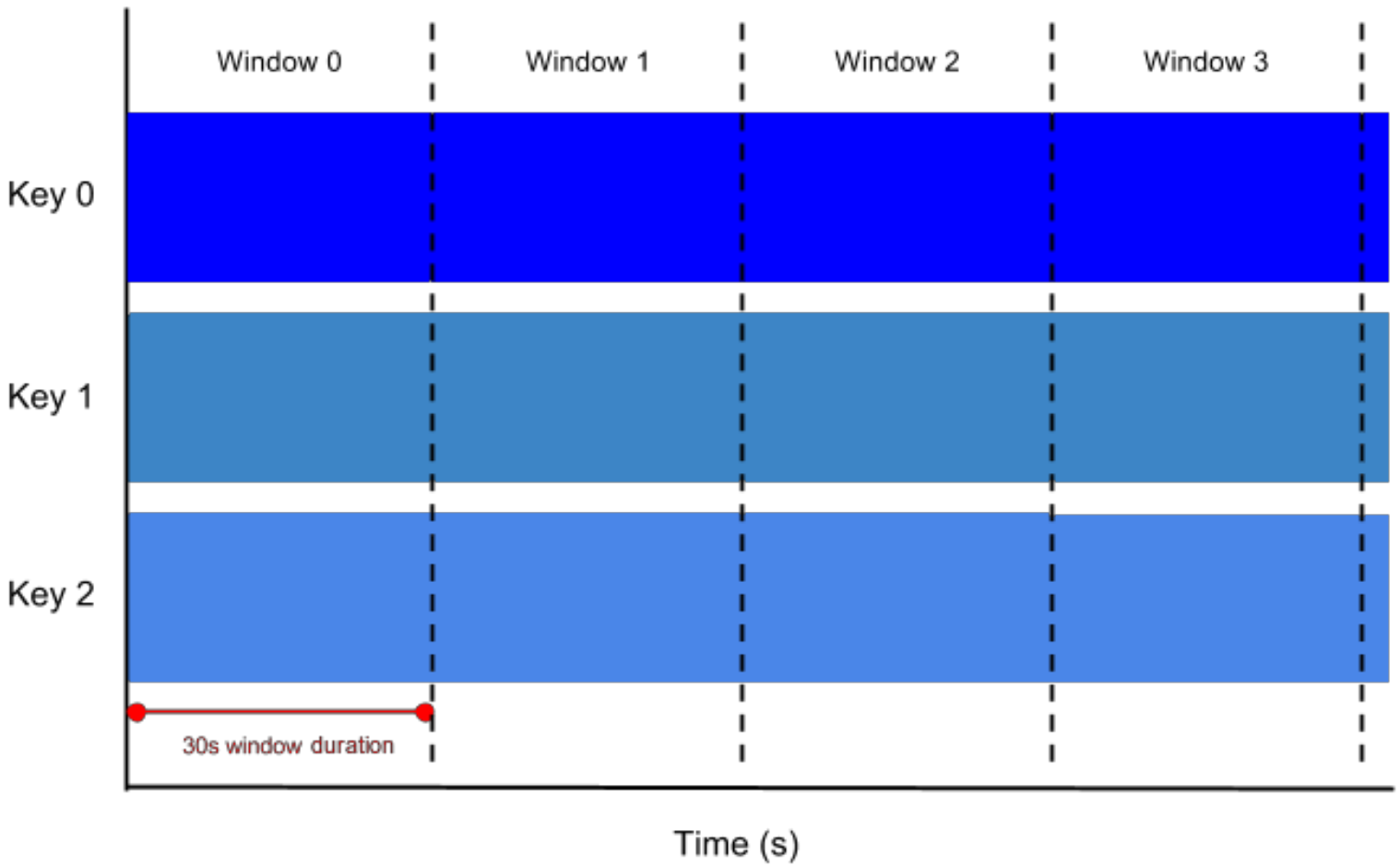
Previous

Next

② Know your window types

- **Global, Fixed, Sliding, Session** (4 types)
- **Global** - The default, uses a single window for entire pipeline
- **Fixed time** - Every (x) period of time
  - Every 5 seconds, 10 minutes, etc.
- **Sliding time** - Overlapping time windows
- **Session** - Within certain time of certain elements:
  - For example, *Time since last user/mouse activity*

Fixed time Window



Return to Table of Contents

Choose a Lesson

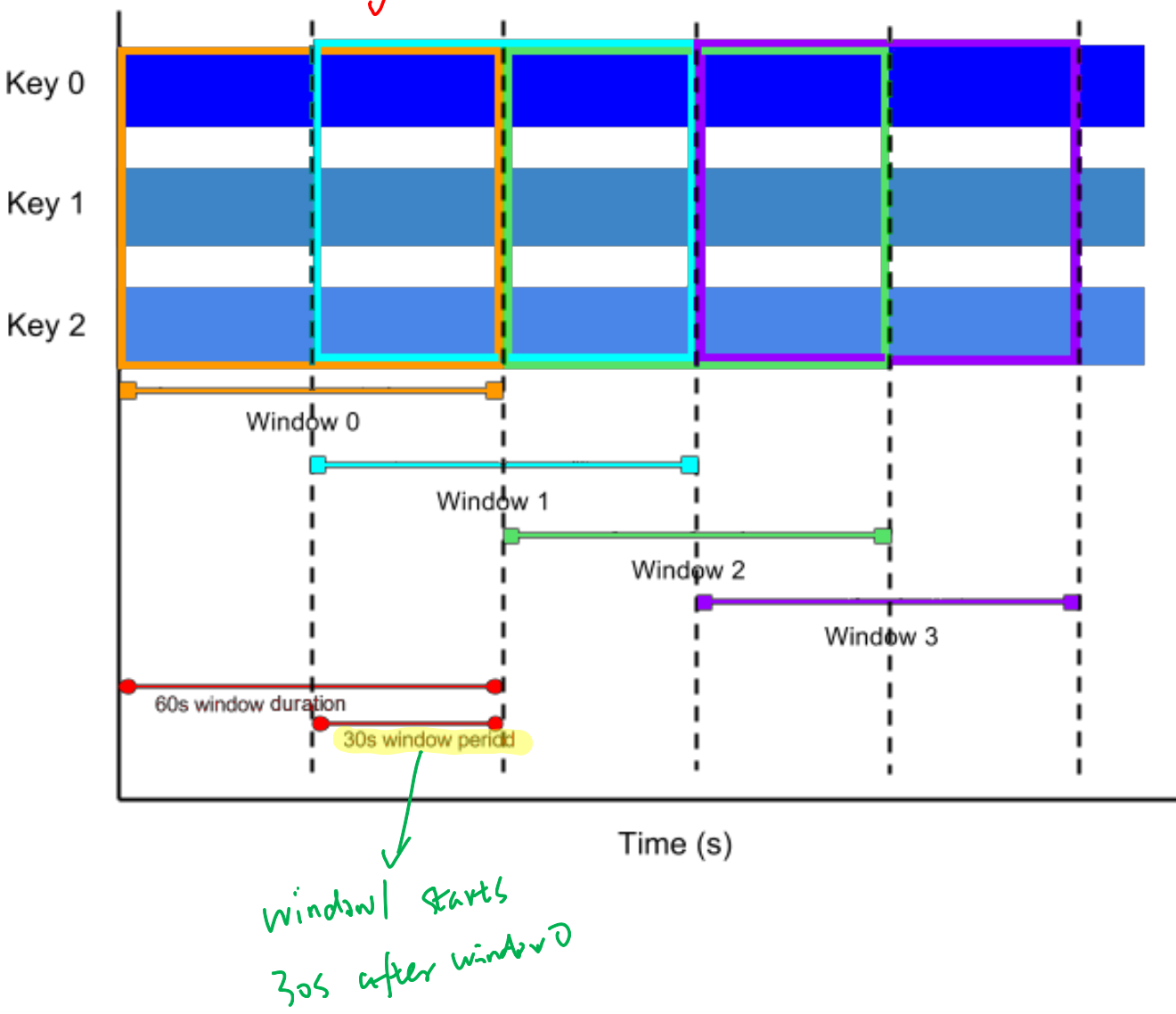
- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Additional Best Practices

Previous

Next

overlap with each other



window1 starts 30s after window0

Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

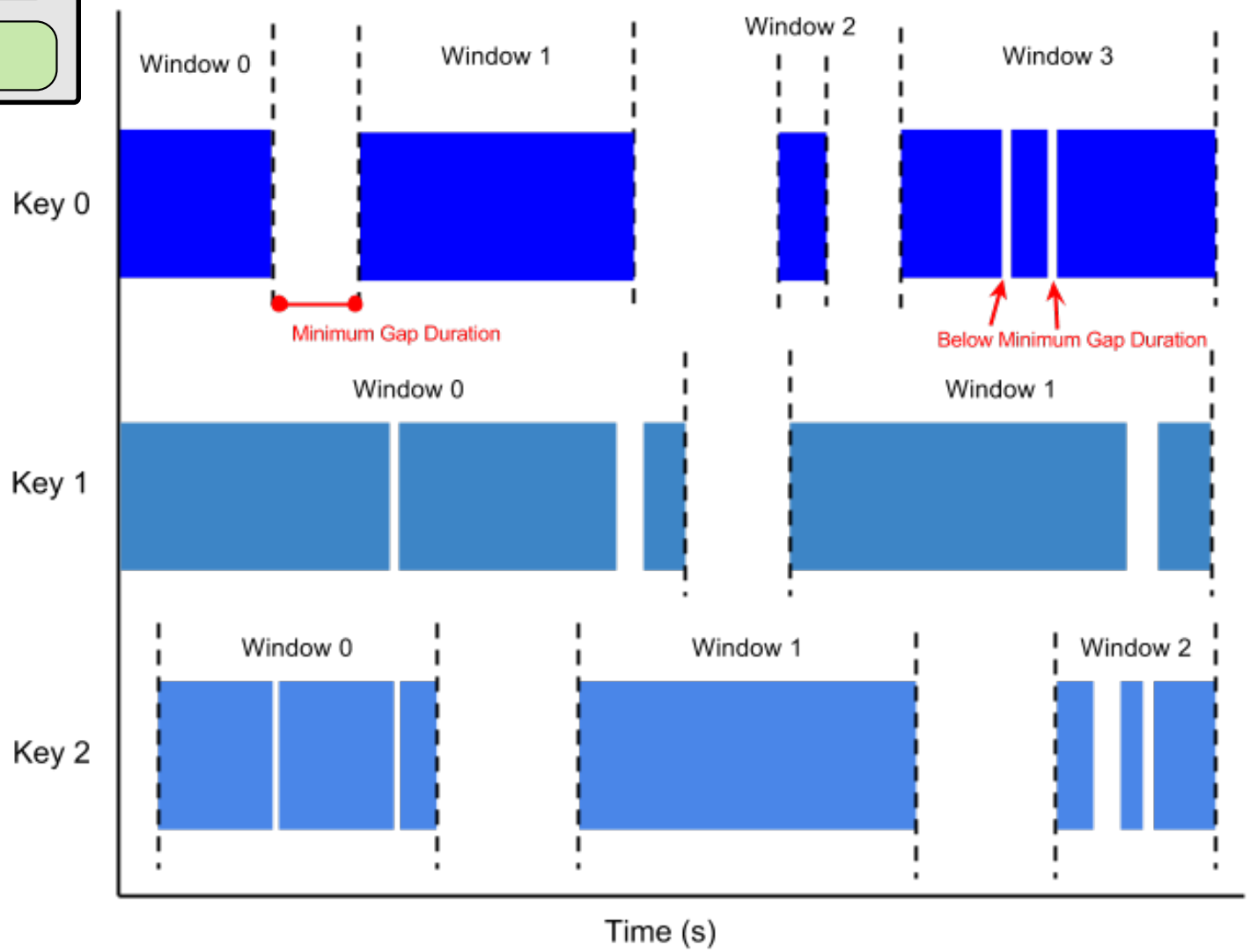
Additional Best Practices

Previous

Next

e-commerce website record "how long for each activity"

Session Window (based on incoming elements)



Return to Table of Contents

Choose a Lesson

- Data Processing Challenges
- Cloud Dataflow Overview
- Key Concepts
- Template Hands On
- Streaming Ingest Pipeline Hands On
- Additional Best Practices

Additional Best Practices

Previous

Updating Dataflow Pipelines

from a pub/sub topic

- **Scenario:** Update streaming Dataflow pipeline with new code:
  - New code = new pipeline not compatible with current version
  - Need data to *switch over* to new job/pipeline without losing anything in the process
- **Solution**: Update job:
  - Creates new job with same name/new *jobID*
- Compatibility between old/new jobs:
  - Map old to new job transforms with **transform mapping**
    - "Bridge" between old and new code base *
  - After compatibility check:
    - Buffered data transferred to new job, using transform mapping to translate changes