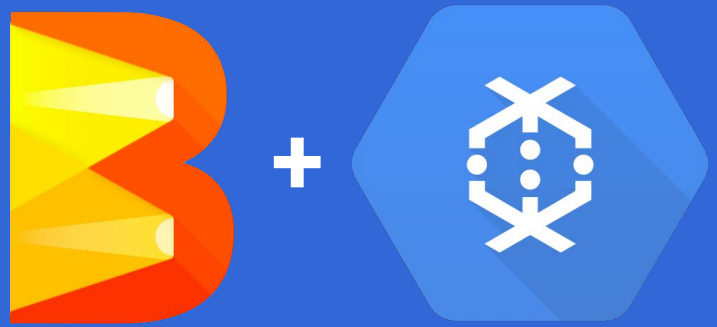
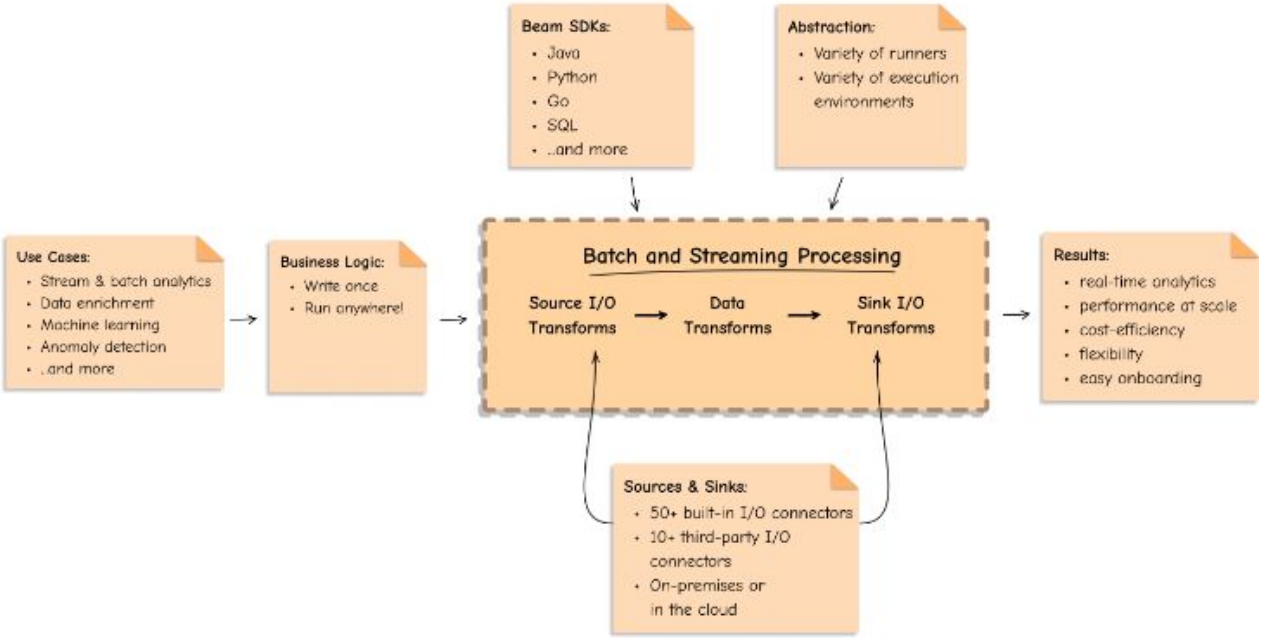
Apache Beam &
Google Cloud Dataflow



Slides by Frances Perry, April 2016

Why Apache Beam?

Apache Beam is an open-source, unified programming model for batch and streaming data processing pipelines that simplifies large-scale data processing dynamics



Key Concepts

- A Beam pipeline is a graph (specifically, a directed acyclic graph) of all the data and computations in your data processing task.
- A PCollection is an unordered bag of elements.
    - Each PCollection is a potentially distributed, homogeneous data set or data stream, and is owned by the specific Pipeline object for which it is created. Multiple pipelines cannot share a PCollection. Beam pipelines process PCollections, and the runner is responsible for storing these elements.
- A PTransform (or transform) represents a data processing operation, or a step, in your pipeline. A transform is usually applied to one or more input PCollection objects

Transforms

- Source transforms such as `TextIO.Read` and `Create`.
- A source transform conceptually has no input.
- Processing and conversion operations such as
  - `ParDo`, `GroupByKey`, `CoGroupByKey`, `Combine`, and `Count`.
- Outputting transforms such as `TextIO.Write`.
- User-defined, application-specific composite transforms.

Catalog of transforms

Beam Model: Asking the Right Questions

What results are calculated?

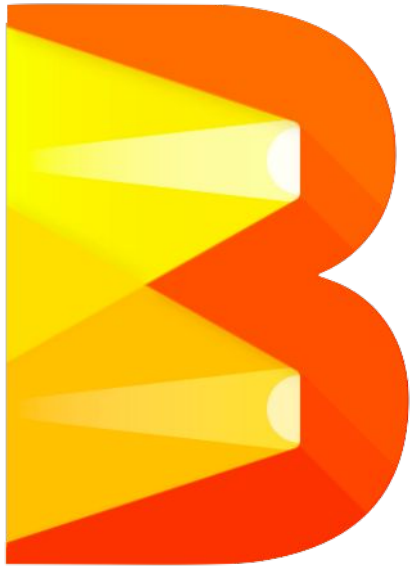
Where in event time are results calculated?

When in processing time are results materialized?

How do refinements of results relate?

What is Apache Beam?

1. The Beam Model: What / Where / When / How
2. SDKs for writing Beam pipelines -- starting with Java
3. Runners for Existing Distributed Processing Backends
   a. Apache Flink (thanks to dataArtisans)
   b. Apache Spark (thanks to Cloudera)
   c. Google Cloud Dataflow (fully managed service)
   d. Local (in-process) runner for testing

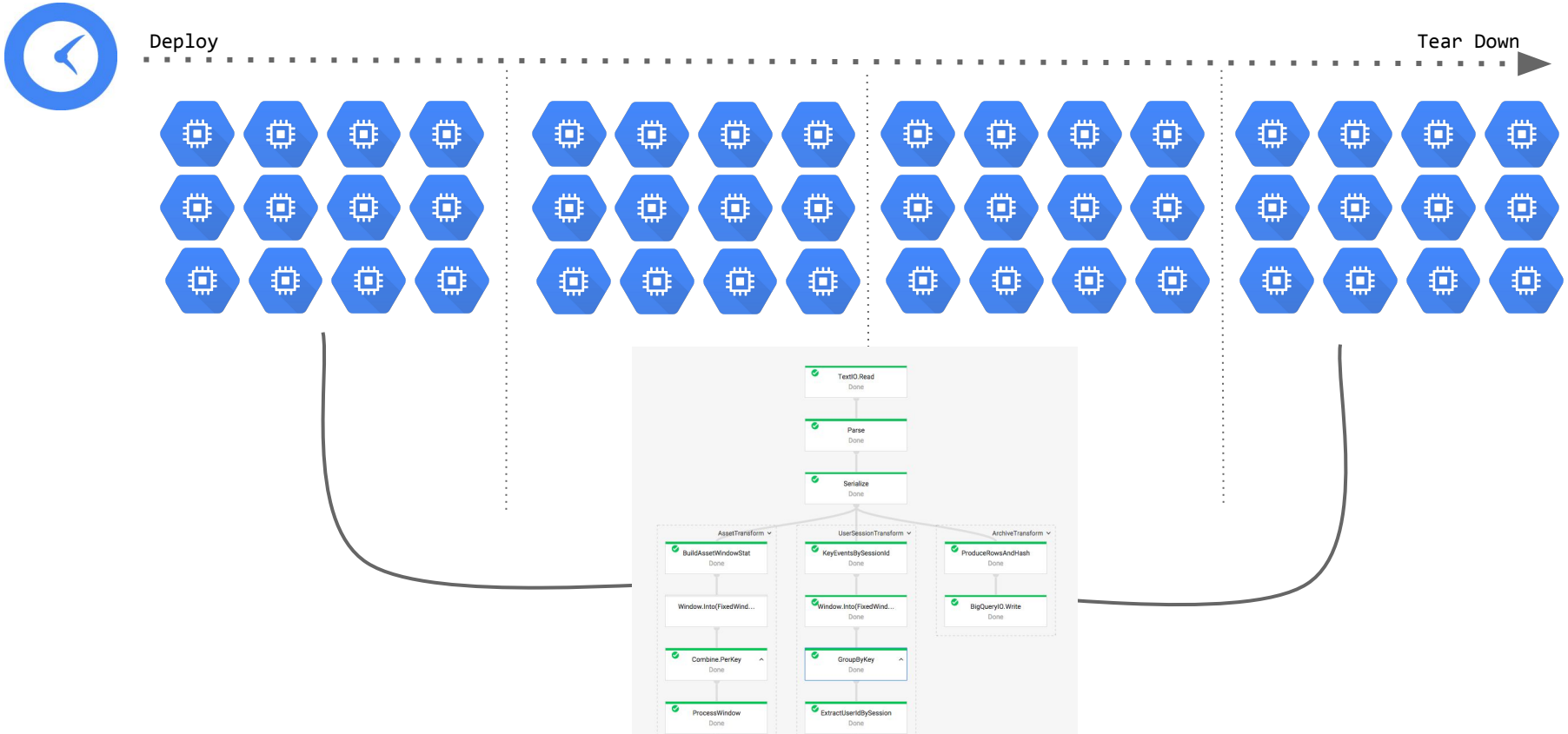


The Cloud Dataflow Service

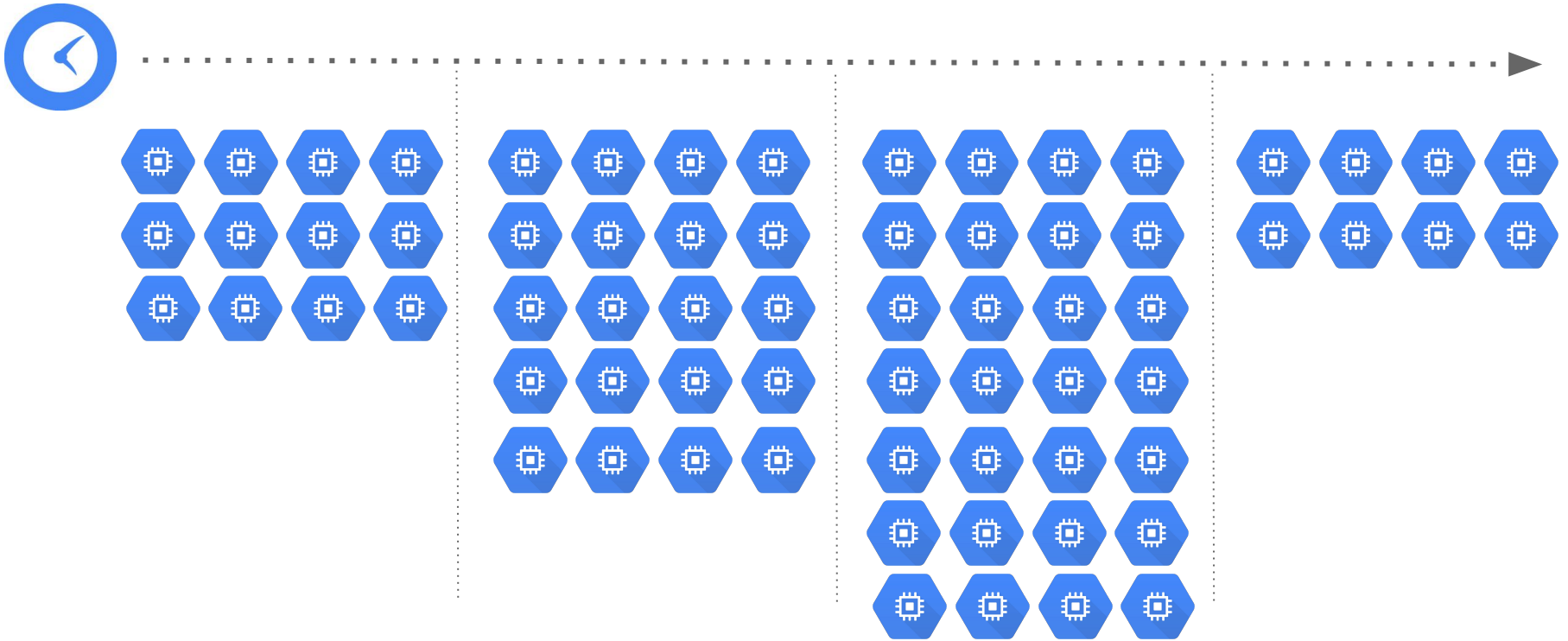
A great place for executing Beam pipelines which provides:

- Fully managed, no-ops execution environment
- Integration with Google Cloud Platform
- Java support in GA. Python in Alpha.

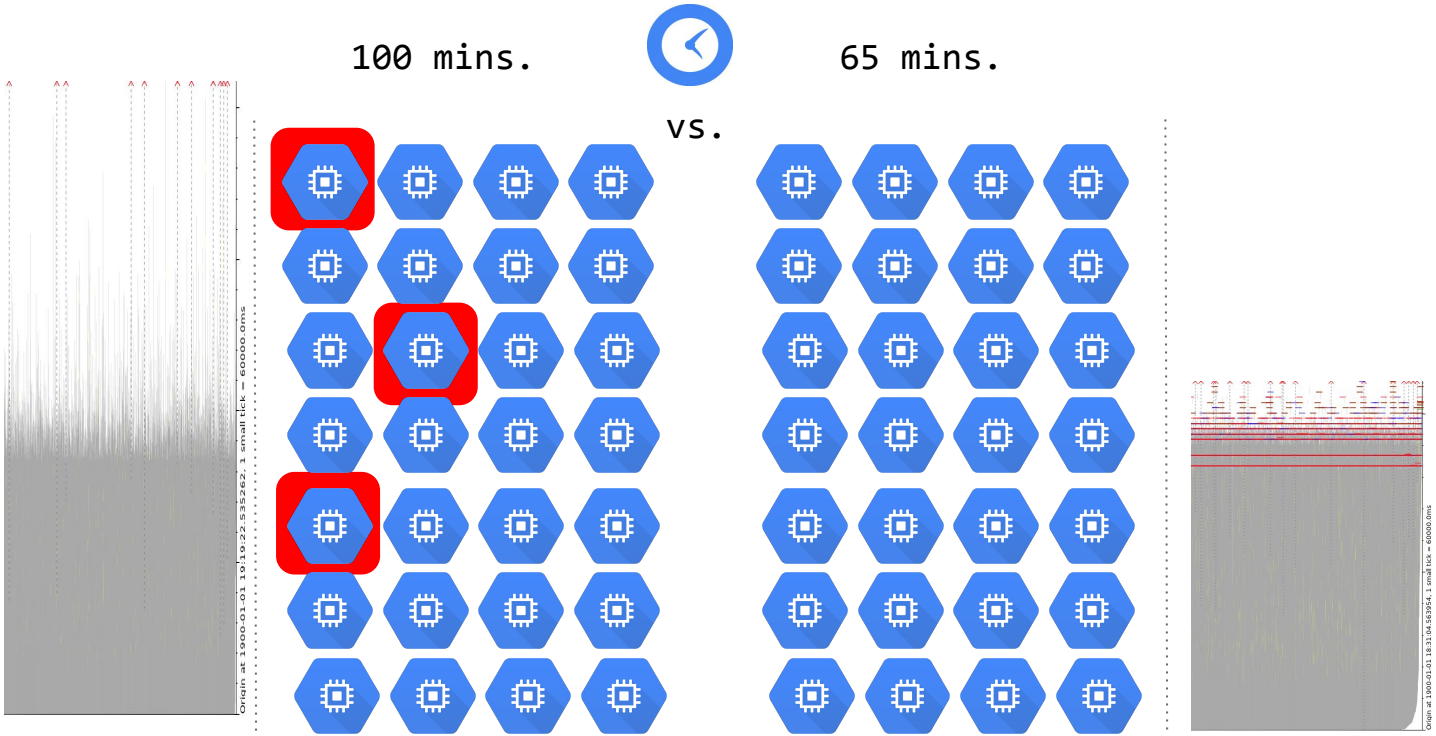
Fully Managed: Worker Lifecycle Management



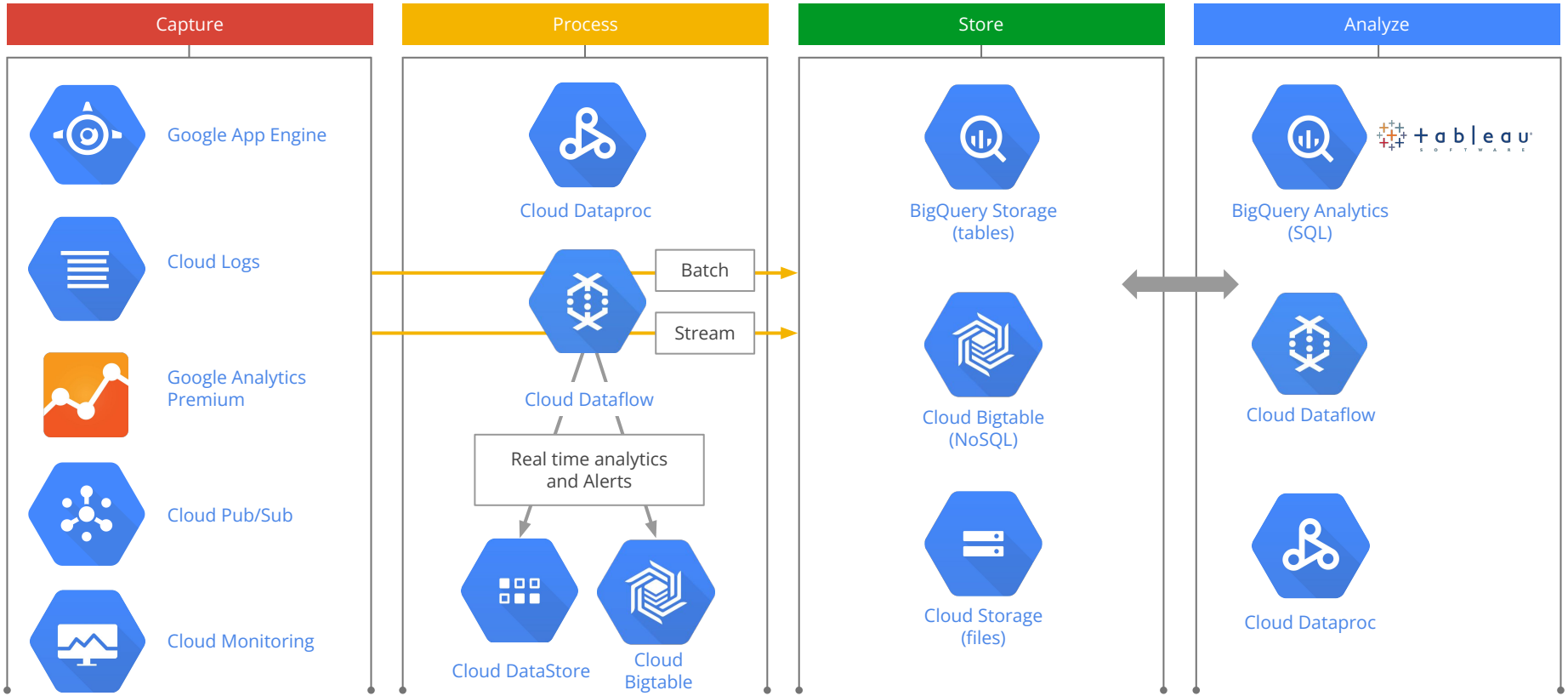
Fully Managed: Dynamic Worker Scaling



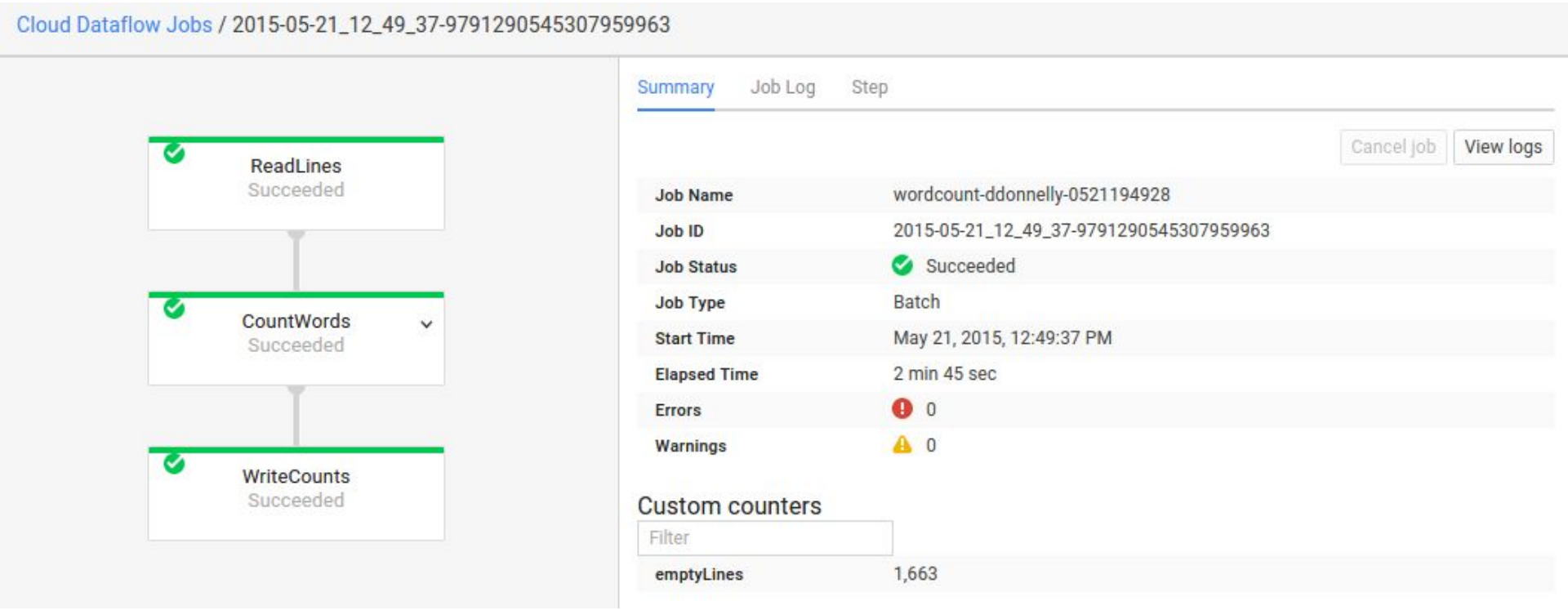
Fully Managed: Dynamic Work Rebalancing



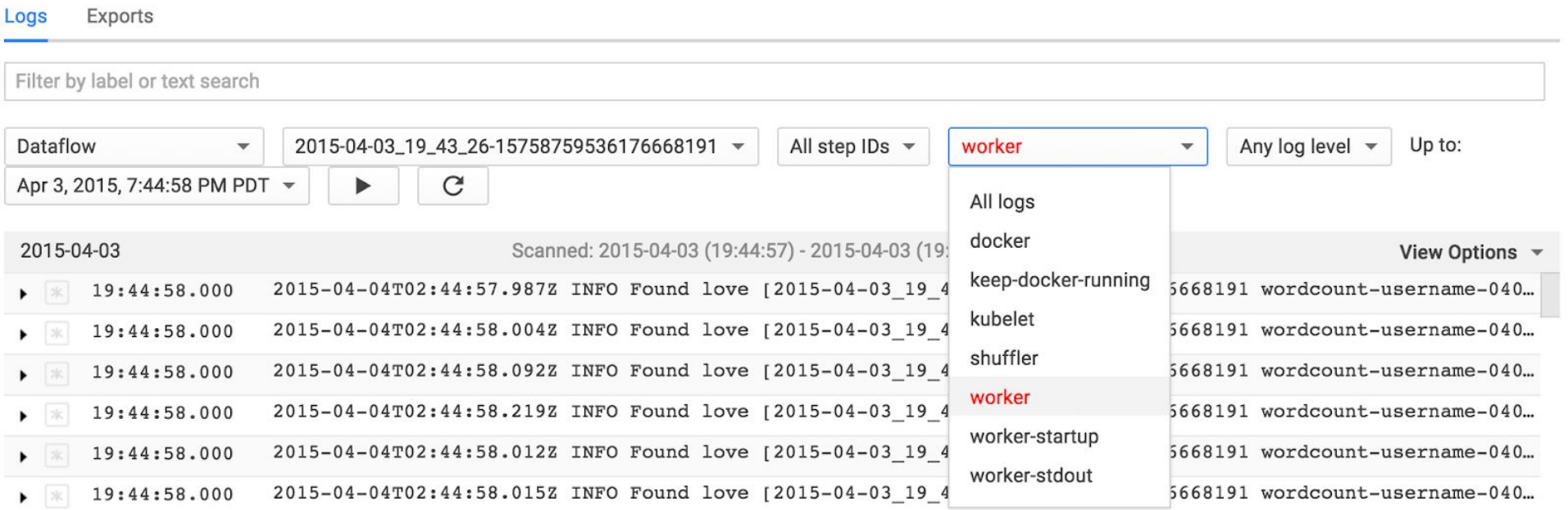
Integrated: Part of Google Cloud Platform



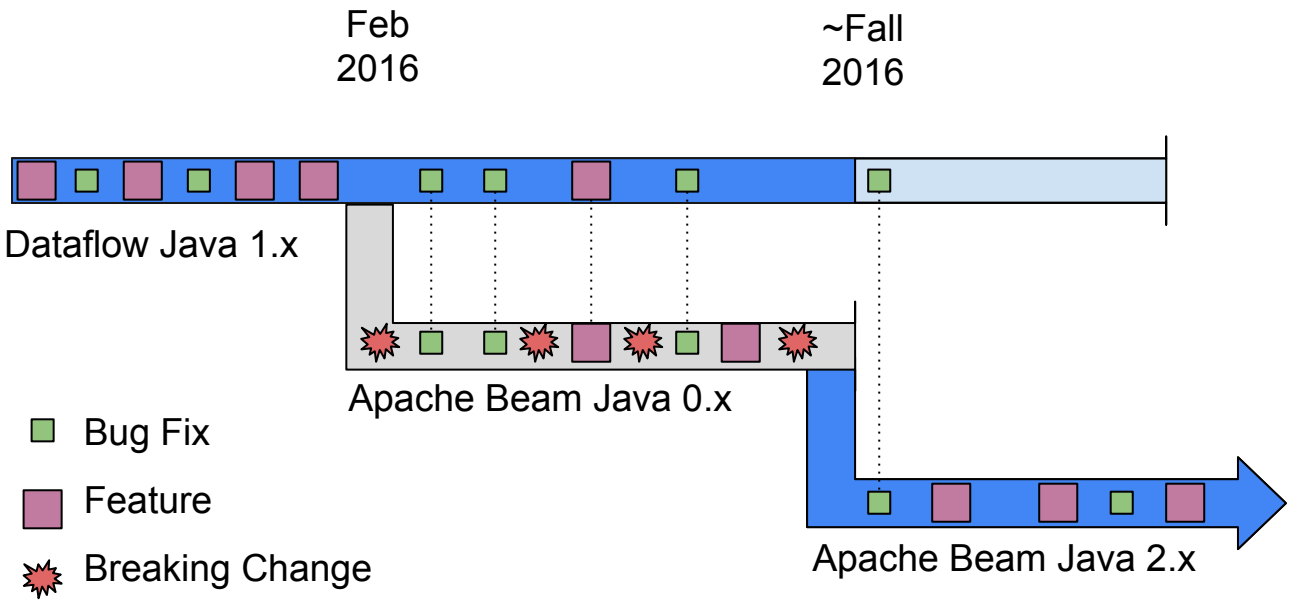
Integrated: Monitoring UI



Integrated: Distributed Logging



Transitioning from Dataflow 1.x to Beam

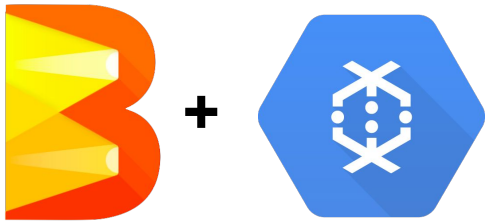


Learn More!

Cloud Dataflow
http://cloud.google.com/dataflow/

Cloud Dataflow on Stack Overflow
http://www.stackoverflow.com/questions/tagged/google-cloud-dataflow

Apache Beam
https://beam.apache.org

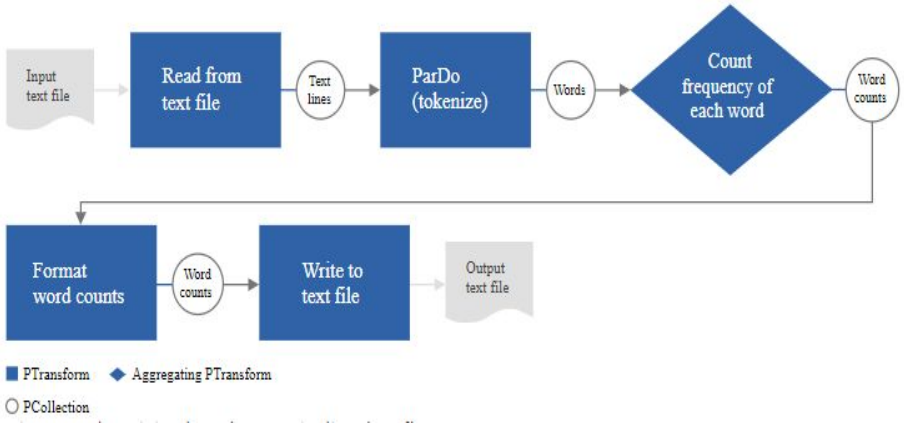


Thank you!

Apache Beam Wordcount Example

Key Concepts

- Creating the Pipeline
- Applying transforms to the Pipeline
- Reading input (in this example: reading text files)
- Applying ParDo transforms
- Applying SDK-provided transforms (in this example: Count)
- Writing output (in this example: writing to a text file)
- Running the Pipeline



Apache Beam Wordcount Example

```
# As part of the initial setup, install Google Cloud Platform specific extra components.
pip install apache-beam[gcp]
export PROJECT_ID=`gcloud config get project`
export BUCKET=$PROJECT_ID
export REGION="us-central1"
python -m apache_beam.examples.wordcount --input
gs://dataflow-samples/shakespeare/kinglear.txt \
                                --output gs://$BUCKET/counts \
                                --runner DataflowRunner \
                                --project $PROJECT_ID\
                                --region $REGION \
                                --temp_location gs://$BUCKET/tmp/
```

demo

Dataflow Templates

demo

Watermark and late data

- lag between the time a data event occurs (the “event time”, determined by the timestamp on the data element itself) and the time the actual data element gets processed at any stage in your pipeline (the “processing time”, determined by the clock on the system processing the element).
- In addition, there are no guarantees that data events will appear in your pipeline in the same order that they were generated.

Watermark and late data

For example, let’s say we have a PCollection that’s using fixed-time windowing, with windows that are five minutes long.

For each window, Beam must collect all the data with an event time timestamp in the given window range (between 0:00 and 4:59 in the first window, for instance).

Data with timestamps outside that range (data from 5:00 or later) belong to a different window.

Watermark and late data

Beam tracks a watermark, which is the system’s notion of when all data in a certain window can be expected to have arrived in the pipeline.

Once the watermark progresses past the end of a window, any further element that arrives with a timestamp in that window is considered late data.

Watermark and late data