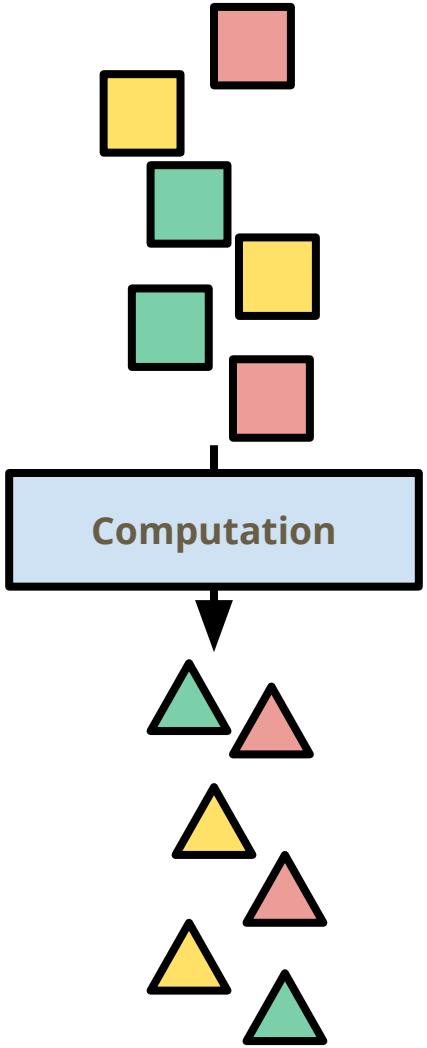
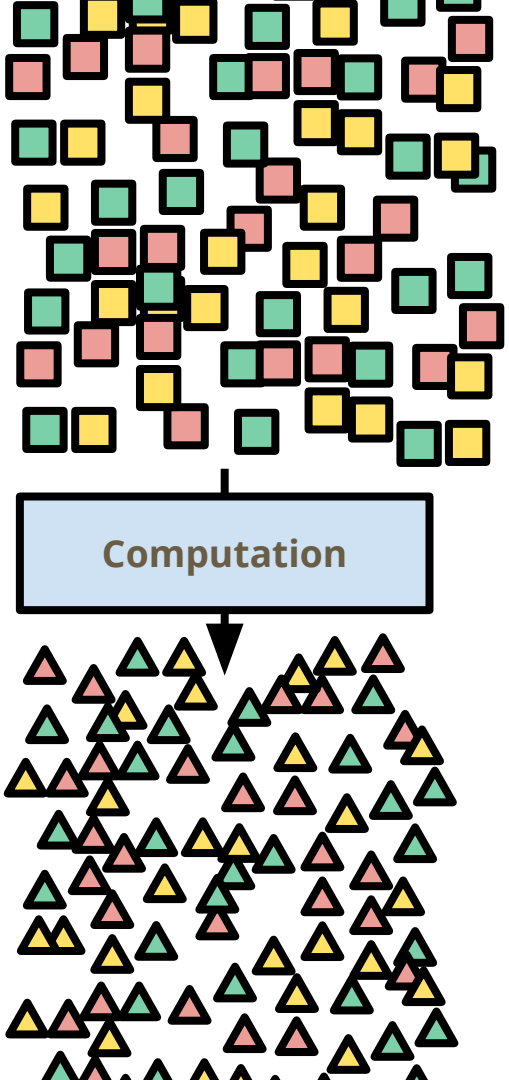
Apache Beam: Portability in the times of Real-time Streaming

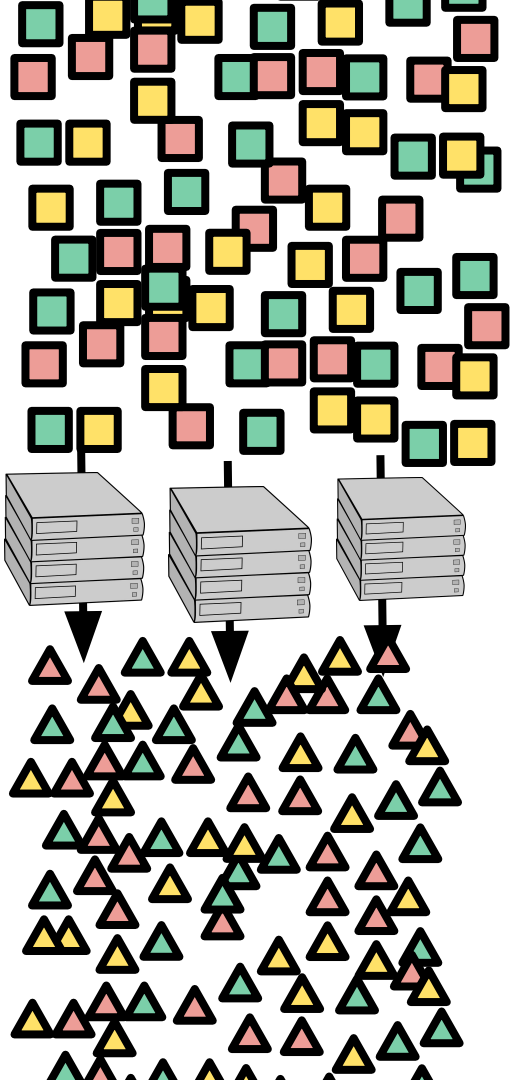
Pablo Estrada
Apache Beam Committer
Software Engineer @ Google
pabloem@apache.org - polecito@

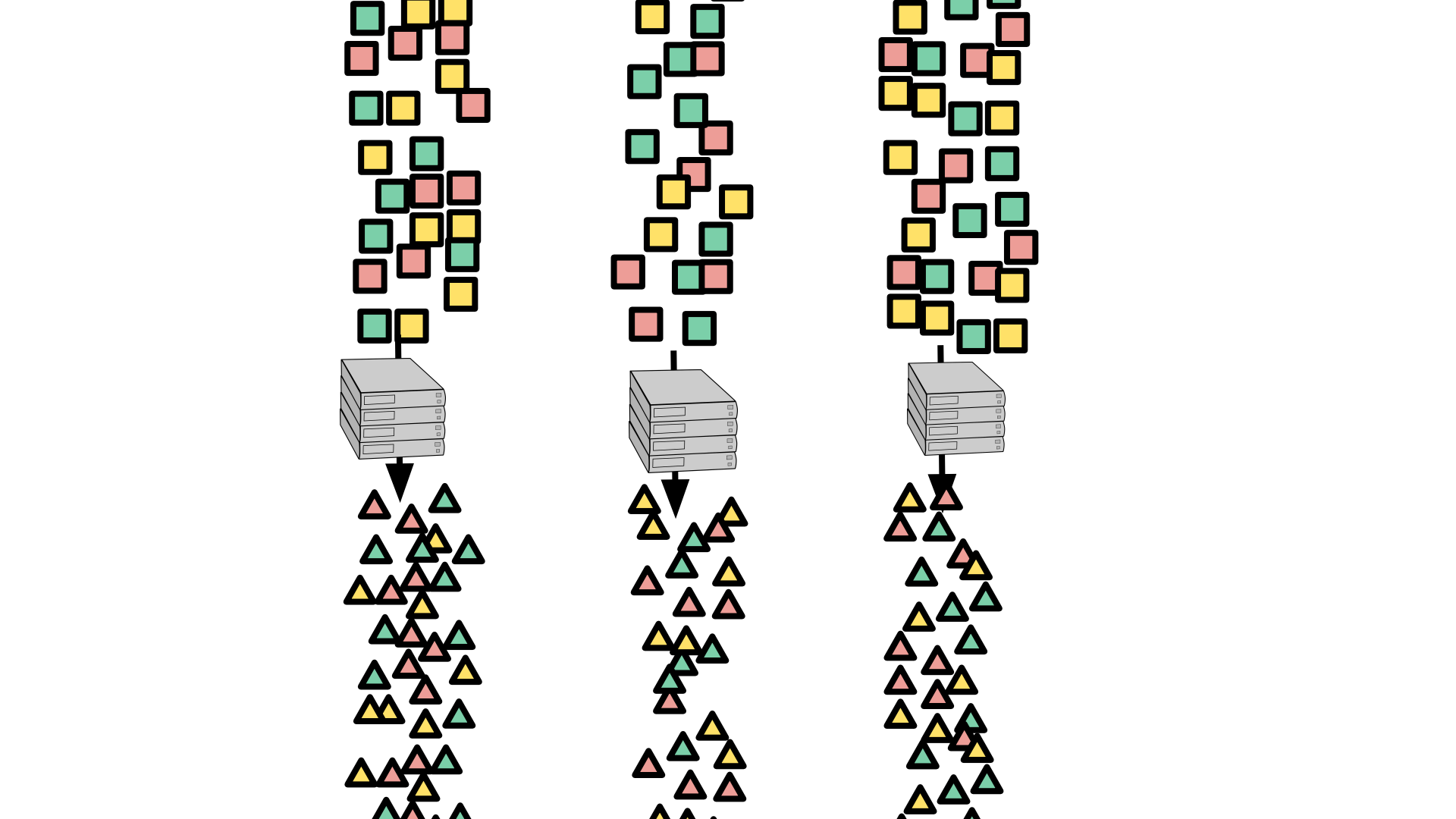
Agenda

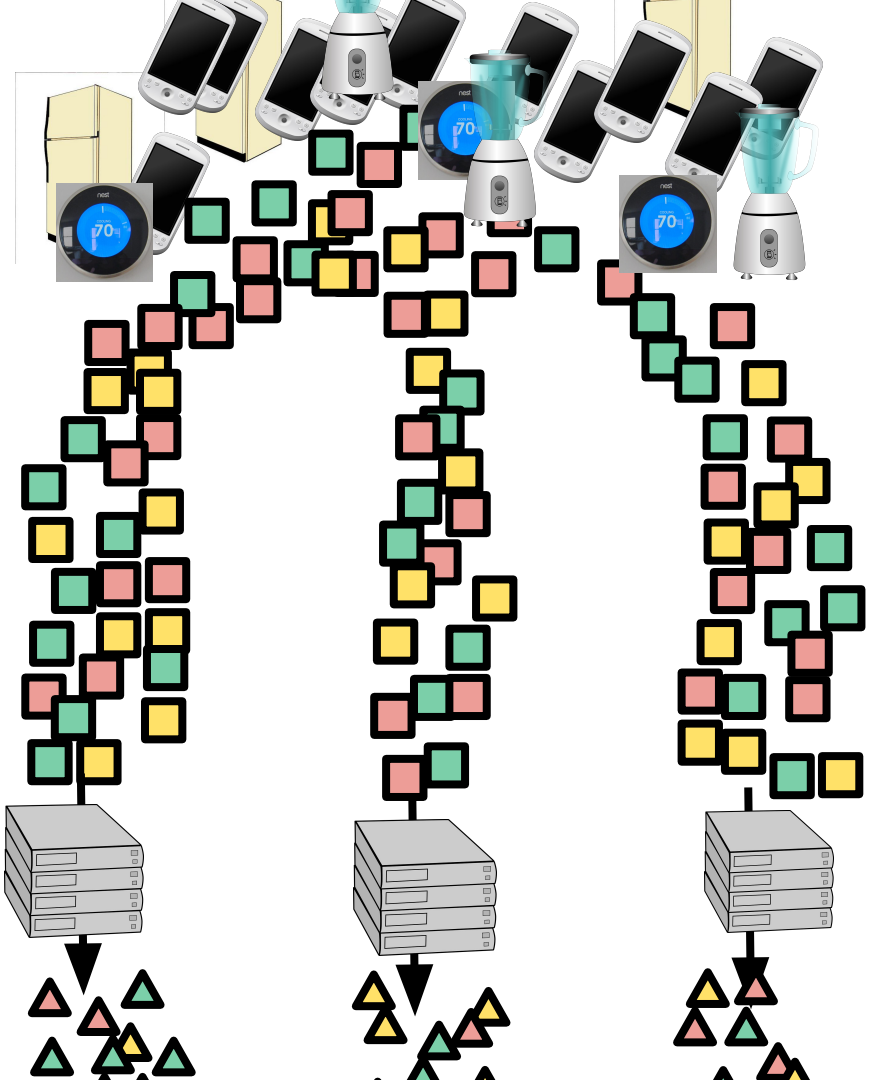
- Quick intro to Beam
- What is Portability
- Why we invest in Beam
- Flink + Beam integration architecture
  - Talk about the rough edges
- Bleeding edge of Beam / Upcoming features
- How to get involved


Computation

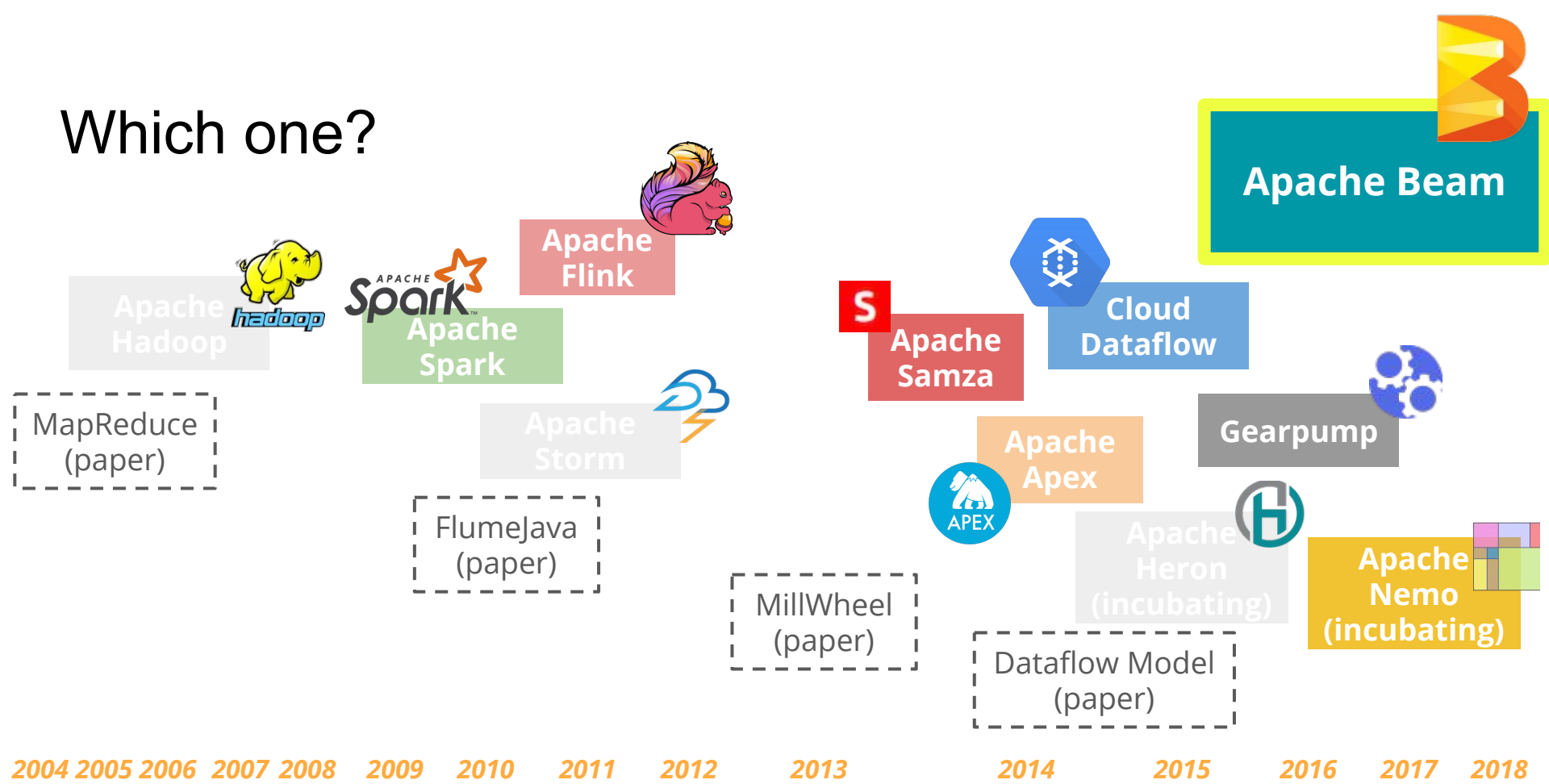

Computation







Which one?


Apache Beam
Apache Hadoop
Apache Spark
Apache Flink
Apache Storm
Apache Samza
Cloud Dataflow
Apache Apex
Gearpump
Apache Heron (incubating)
Apache Nemo (incubating)
MapReduce (paper)
FlumeJava (paper)
MillWheel (paper)
Dataflow Model (paper)
2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018

Apache Beam?

Beam is a **portable**, **unified** programming

model for Batch and Streaming.

Unified?

Same API for Batch and Streaming computations

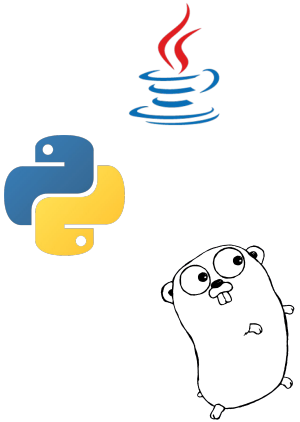
Portability?

Runner portability + Language portability

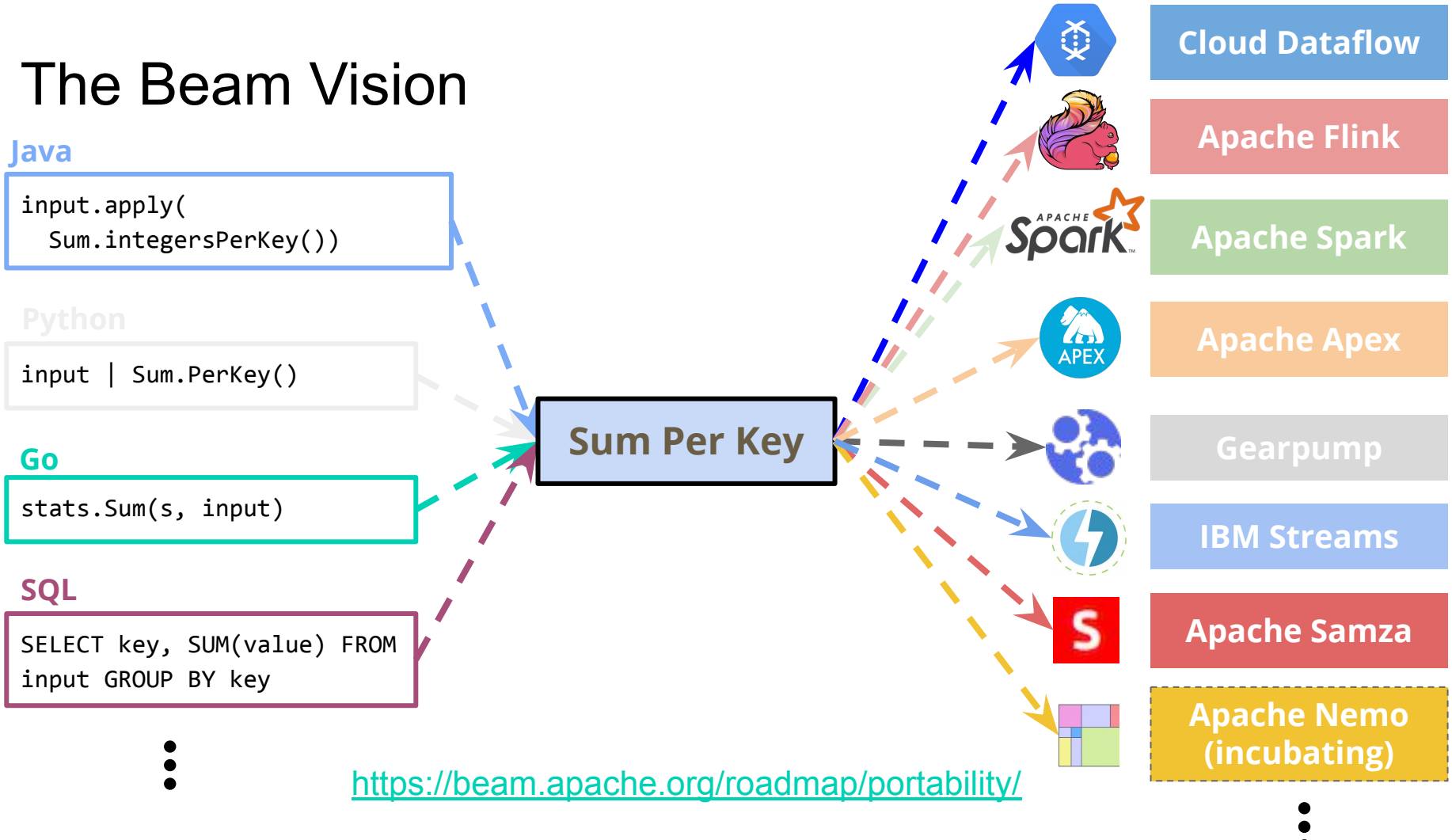
- Run your pipeline in the runner of your choice.
- Write your pipeline in the language that works / is most convenient for you.

Portability?

Runner portability + Language portability



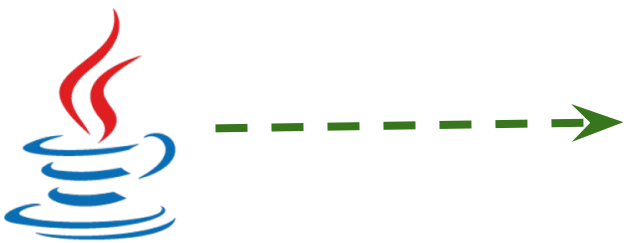
The Beam Vision

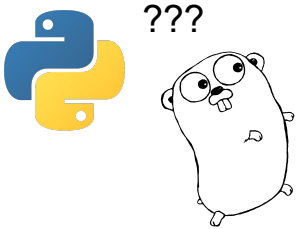


Interesting Abstractions

- User defined functions / Business logic
- Event time vs Processing time
- DAG execution

Runner portability (today)





Cloud Dataflow

Apache Flink

Apache Spark

Apache Apex

Gearpump

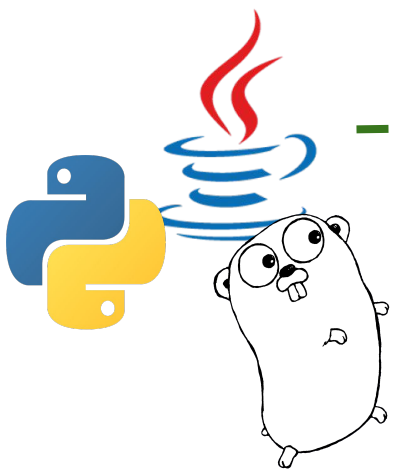
IBM Streams

Apache Samza

Apache Nemo (incubating)

Language portability (today)

In-progress + MVP



Cloud Dataflow

Apache Flink

Apache Spark

Apache Apex

Gearpump

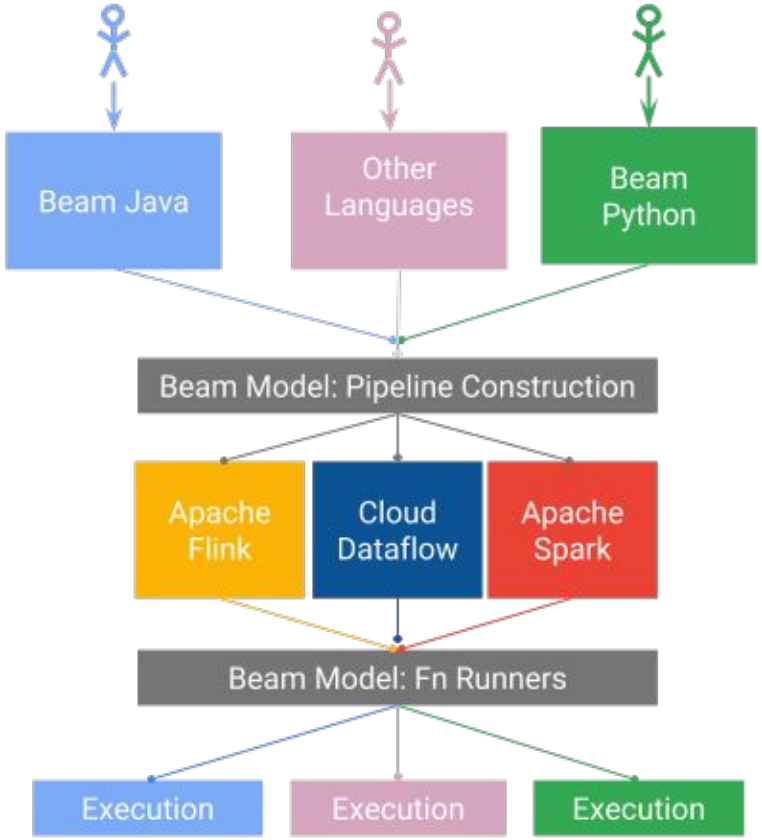
IBM Streams

Apache Samza

Apache Nemo (incubating)

Language portability

- Doesn’t mean “all languages for free!”
- It does mean: Reducing the cost-of-entry for a new SDK
- Result of abstracting:
  - DAG execution from UDF / Business Logic

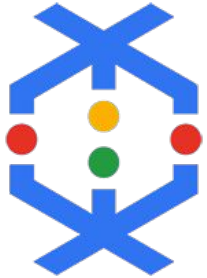


Why continue developing Beam?

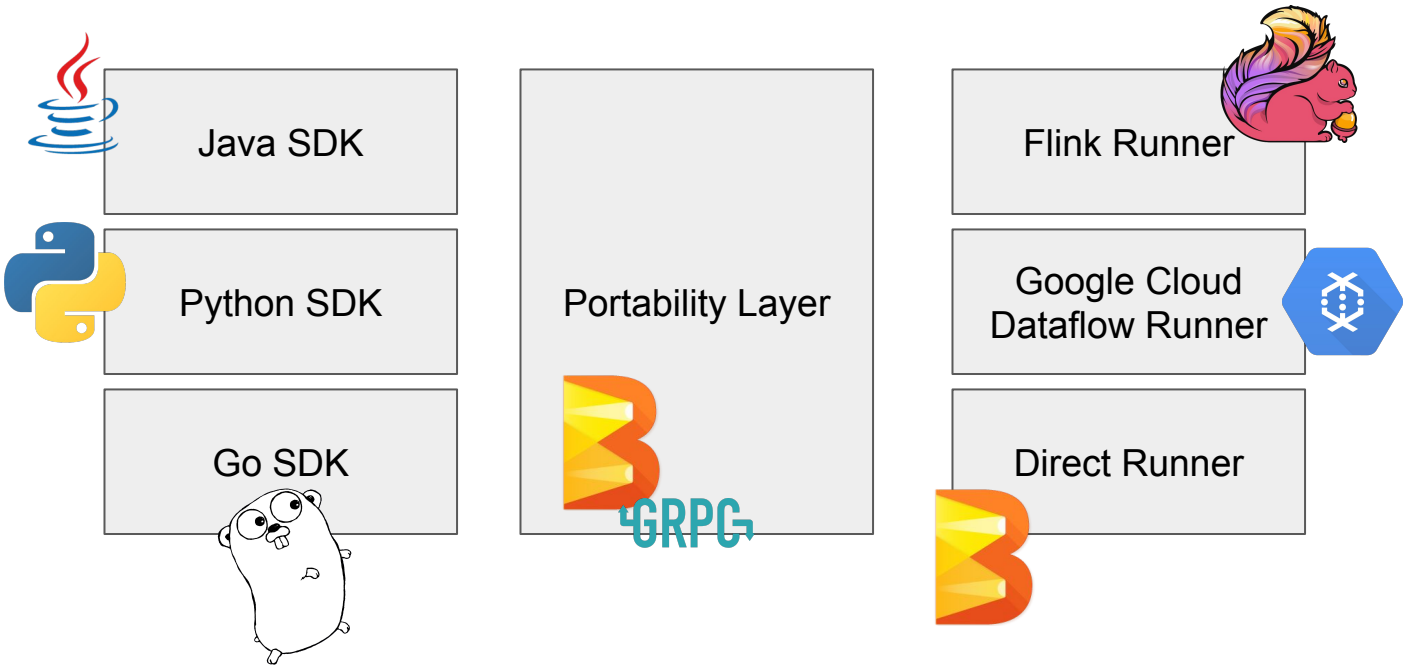
- Open and free technologies are important for Cloud
  - Working on Apache Beam helps us make Google Cloud Dataflow a better product
  - Try and meet customers where they are: On prem? On other clouds? On our own cloud?
- Contribute to the big data ecosystem
  - Some awesome collaborations between Beam / Flink / other projects
    - Single APIs for Batch and Streaming
    - Event Time Processing / Windowing
    - Streaming SQL
    - Splittable DoFn (Splittable / Redistributable Maps)

Investments in Beam

Model Analysis	TensorFlow Transform	Data Validation



Beam today


Java SDK
Python SDK
Go SDK
Portability Layer
gRPC
Flink Runner
Google Cloud Dataflow Runner
Direct Runner

Go Word-count

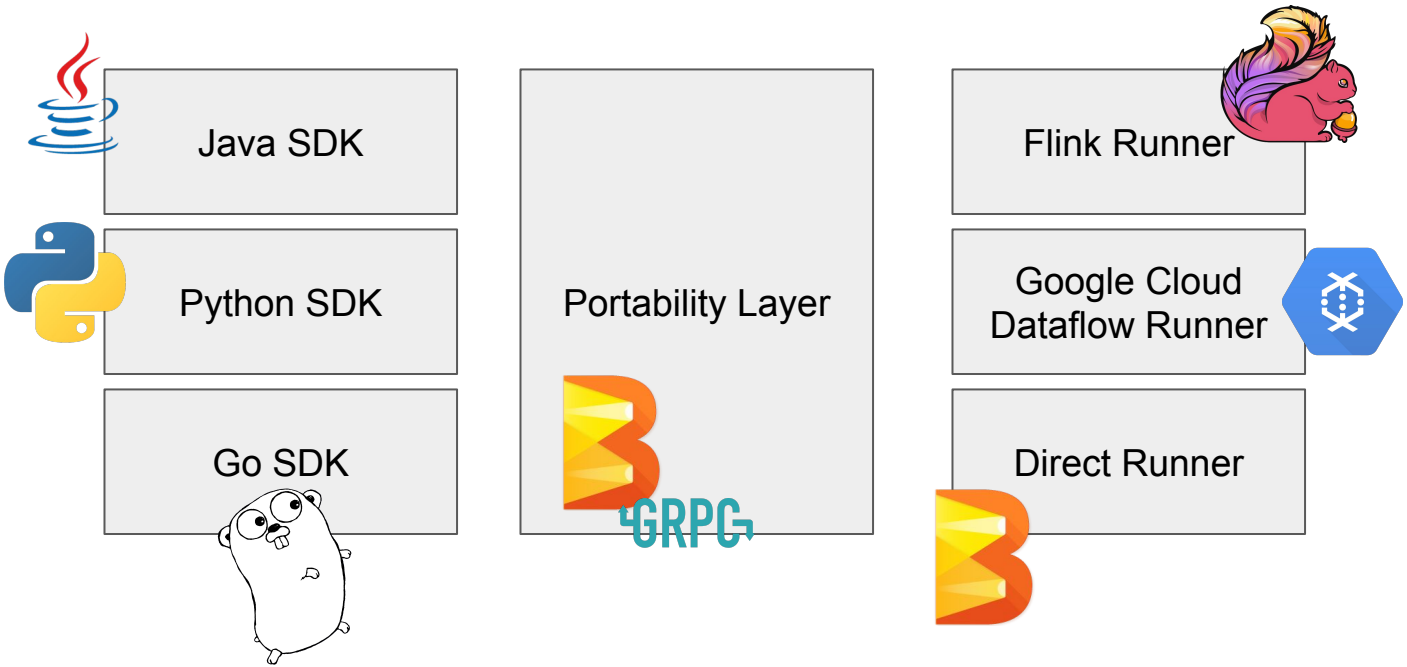
```
func main() {
        flag.Parse()
        beam.Init()

        p := beam.NewPipeline()
        s := p.Root()

        lines := textio.Read(s, *input)
        counted := CountWords(s, lines)
        formatted := beam.ParDo(s, formatFn, counted)
        textio.Write(s, *output, formatted)

        beamx.Run(context.Background(), p)
}
```

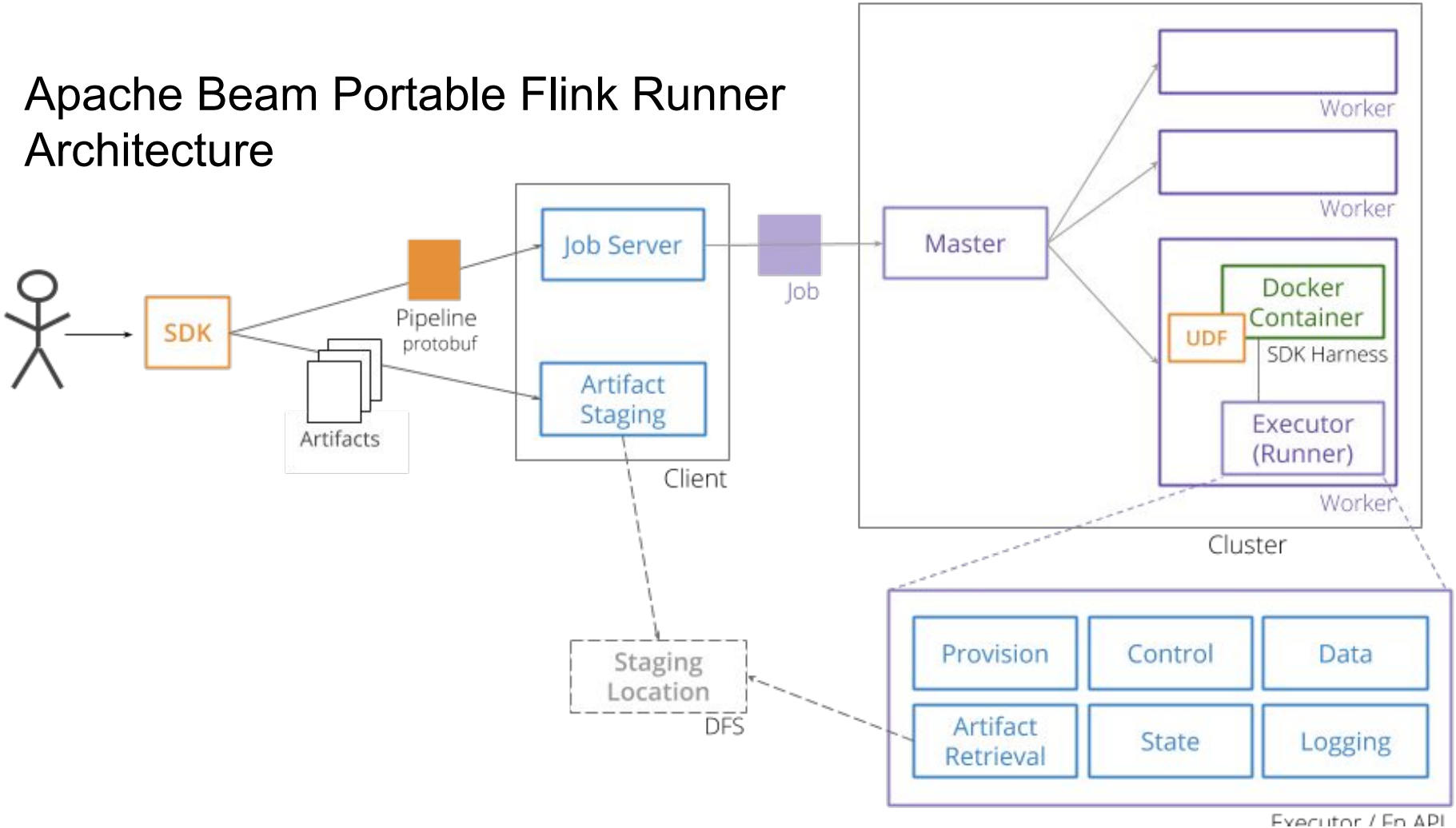
Beam today


Java SDK
Python SDK
Go SDK
Portability Layer
gRPC
Flink Runner
Google Cloud Dataflow Runner
Direct Runner

Apache Beam Portable Flink Runner Architecture

- Executor setup (Language environment)
- State, Logging, Artifact retrieval, etc..
- Job Submission API
- Artifact Staging API
- ...

Apache Beam Portable Flink Runner Architecture



Apache Beam Portable Flink Runner Rough Edges

- Data Sources
  - Parallel Reader API not yet there
  - “Splittable DoFn”
- Deployment
  - Still a little awkward, but working to improve
- Very large scale
  - TFX pipelines generate lots of Flink tasks (~250 tasks for basic pipelines)
  - Complex data exchange: many small messages and few >XXMB messages.
  - Able to run pipelines with 100s of MB as input, discovering issues with GBs.

Beam today (bleeding edge)

- Experimenting with multi-language pipelines

Beam today (bleeding edge)

- Python pipeline with Java-based Count

```
65  def run(p, input_file, output_file):
66    # Read the text file[pattern] into a PCollection.
67    lines = p | 'read' >> ReadFromText(input_file)
68
69    counts = (lines
70              | 'split' >> (beam.ParDo(WordExtractingDoFn())
71                            .with_output_types(bytes))
72              | 'count' >> beam.ExternalTransform(
73                  'pytest:beam:transforms:count', None, EXPANSION_SERVICE_ADDR))
74
75    # Format the counts into a PCollection of strings.
76    def format_result(word_count):
77      (word, count) = word_count
78      return '%s: %d' % (word, count)
79
80    output = counts | 'format' >> beam.Map(format_result)
81
82    # Write the output using a "Write" transform that has side effects.
83    # pylint: disable=expression-not-assigned
84    output | 'write' >> WriteToText(output_file)
85
86    result = p.run()
87    result.wait_until_finish()
88
```

Read file

Split Lines

Java-based COUNT

Write out

Features in the roadmap

- Great connectors for Beam on Python + Go
- Multi-language pipelines (e.g. SQL + Python ML)

How can I try it out?

- Clone Beam repository and try running our tests?
  - https://github.com/apache/beam
- Write a Go pipeline (and run it on Flink!)
  - Or Python pipeline :)
  - https://beam.apache.org/roadmap/portability/#python-on-flink
- Try TFX (https://www.tensorflow.org/tfx/)

Want to get involved?

Connect with us in the way you like best:

- Follow @ApacheBeam on Twitter
- Subscribe to the Apache Beam Youtube channel
- Chat on https://the-asf.slack.com/#beam
- User discussions on user@beam.apache.org
  - (to subscribe, send empty mail to user-subscribe@beam.apache.org)
- Development discussions on dev@beam.apache.org
  - (to subscribe, send empty mail to dev-subscribe@beam.apache.org)

END