Python, Java, or Go

It's Your Choice with Apache Beam

BERLIN BUZZWORDS 2019 JUNE 16-18

10th edition

Who are we?

@stadtlegende

Maximilian Michels

Software Engineer / Consultant

Apache Beam / Apache Flink

PMC / Committer

@iemejia

Ismaël Mejía

Software Engineer @ Talend Inc

Apache Beam / Apache Avro

PMC / Committer

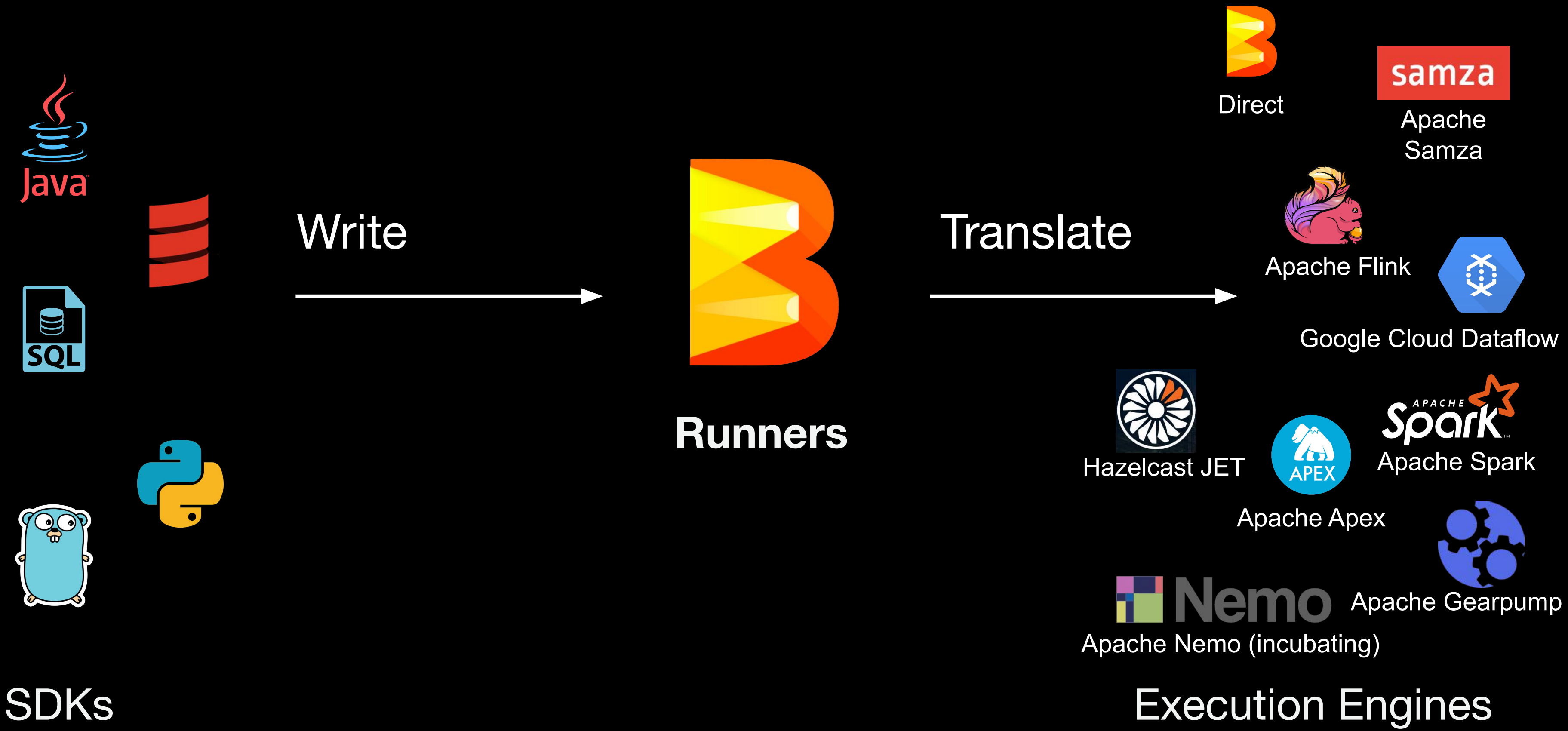
What is Apache Beam?

What is Apache Beam?

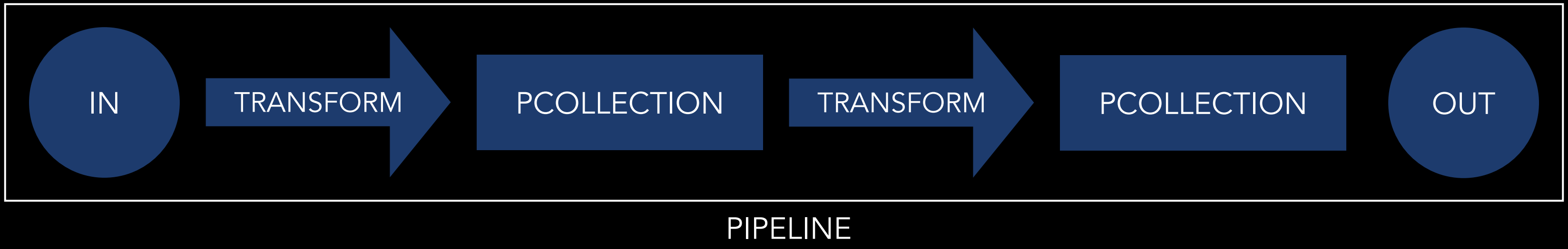
- Apache open-source project
- Parallel/distributed data processing
- Unified programming model for batch and streaming
- Portable execution engine of your choice ("Uber API")
- Programming language of your choice*

Apache Beam

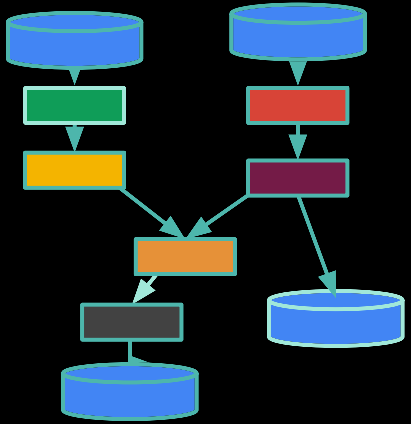
The Vision


Java
SQL
Write
Runners
Translate
Direct
samza
Apache Samza
Apache Flink
Google Cloud Dataflow
Hazelcast JET
APEX
Apache Apex
APACHE Spark
Apache Spark
Nemo
Apache Nemo (incubating)
Apache Gearpump
SDKs
Execution Engines

The API



```
1.  Pipeline p = Pipeline.create(options)
2.  PCollection pCol1 = p.apply(transform).apply(…).…
3.  PCollection pcol2 = pCol1.apply(transform)
4.  p.run()
```



Transforms

- Transforms can be **primitive** or **composite**
- Composite transforms expand to primitive
- Small set of primitive transforms
- Runners can support specialized translation of composite transforms,but don't have to

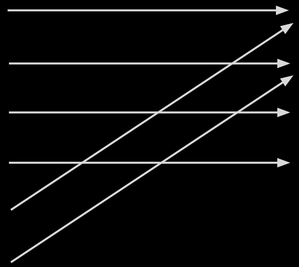
PRIMITIVE TRANSFORMS
ParDo
GroupByKey
AssignWindows
Flatten

Core "primitive" Transforms

ParDo

```
input -> output

"to"  -> KV<"to", 1>
"be"  -> KV<"be", 1>
"or"  -> KV<"or", 1>
"not"-> KV<"not",1>
"to"  -> KV<"to", 1>
"be"  -> KV<"be", 1>
```



GroupByKey

```
KV<k,v>… -> KV<k, [v…]>

KV<"to", [1,1]>
KV<"be", [1,1]>
KV<"or", [1  ]>
KV<"not",[1  ]>
```

"Map/Reduce Phase"

"Shuffle Phase"

Wordcount - Raw version

```
pipeline
   .apply(Create.of("to", "be", "or", "not", "to", "be"))
   .apply(ParDo.of(
       new DoFn<String, KV<String, Integer>>() {
         @ProcessElement
         public void processElement(ProcessContext ctx) {
           ctx.output(KV.of(ctx.element(), 1));
         }
       }))
   .apply(GroupByKey.create())
   .apply(ParDo.of(
       new DoFn<KV<String, Iterable<Integer>>, KV<String, Long>>() {
         @ProcessElement
         public void processElement(ProcessContext ctx) {
           long count = 0;
           for (Integer wordCount : ctx.element().getValue()) {
             count += wordCount;
           }
           ctx.output(KV.of(ctx.element().getKey(), count));
         }
       }))
```

EXCUSE ME,
THAT WAS UGLY AS HELL

Wordcount — Composite Transforms

```
pipeline
  .apply(Create.of("to", "be", "or", "not", "to", "be"))
  .apply(MapElements.via(
    new SimpleFunction<String, KV<String, Integer>>() {
      @Override
      public KV<String, Integer> apply(String input) {
        return KV.of(input, 1);
      }
    }))
  .apply(Sum.integersPerKey());
```

Composite
Transforms

Wordcount - More Composite Transforms

```
pipeline
    .apply(Create.of("to", "be", "or", "not", "to", "be"))
    .apply(Count.perElement());
```

Composite Transforms

Python to the Rescue

```
pipeline
    | beam.Create(['to', 'be', 'or', 'not', 'to', 'be'])
    | beam.Map(lambda word: (word, 1))
    | beam.GroupByKey()
    | beam.Map(lambda kv: (kv[0], sum(kv[1])))
```

Python to the Rescue

```
pipeline
    | beam.Create(['to', 'be', 'or', 'not', 'to', 'be'])
    | beam.Map(lambda word: (word, 1))
    | beam.CombinePerKey(sum)
```

There is so much more on Beam

IO transforms – produce PCollections of timestamped elements and a watermark.

Filesystems

Amazon S3
Apache HDFS
Google Cloud Storage
Local Filesystems

File Formats

Text
Avro
Parquet
TFRecord
Xml
Tika

Databases

Amazon DynamoDB
Apache Cassandra
Apache Hadoop InputFormat
Apache HBase
Apache Hive (HCatalog)
Apache Kudu
Apache Solr
Elasticsearch
Google BigQuery
Google Bigtable
Google Datastore
Google Spanner
JDBC
MongoDB
Redis

Messaging

Amazon Kinesis
Amazon SNS / SQS
Apache Kafka
AMQP
Google Cloud Pub/Sub
JMS
MQTT
RabbitMQ

There is so much more on Beam

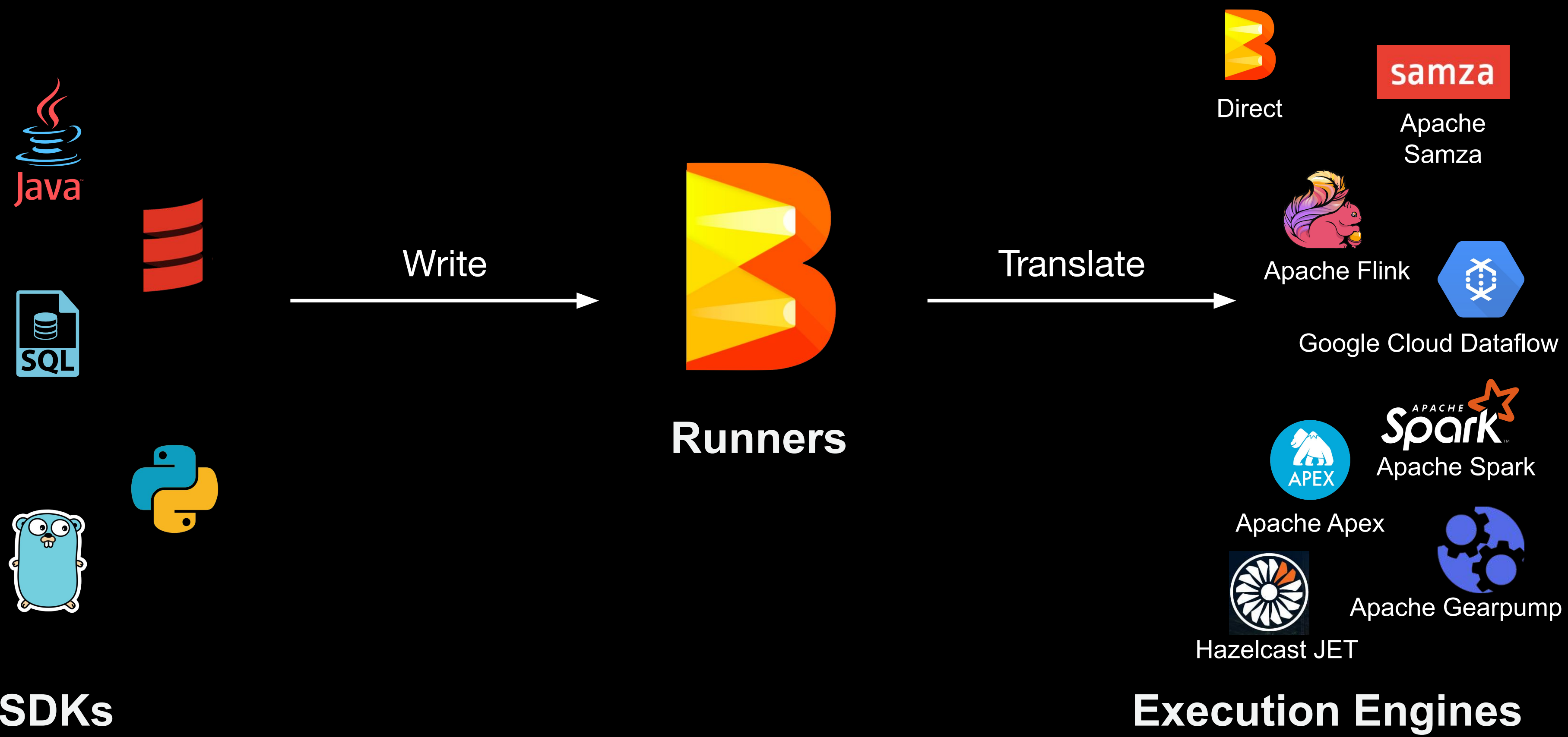
- **More transforms** – Flatten/Combine/Partition/CoGroupByKey (Join)
- **Side inputs** – global view of a PCollection used for broadcast / joins.

Latency / Correctness

- **Window** – reassign elements to zero or more windows; may be data-dependent.
- **Triggers** – user flow control based on window, watermark, element count, lateness.
- **State & Timers** – cross-element data storage and callbacks enable complex operations

What Does Portability Mean?

The Vision



Java
SQL
Write
Runners
Translate
Direct
samza
Apache Samza
Apache Flink
Google Cloud Dataflow
APACHE Spark
Apache Spark
APEX
Apache Apex
Hazelcast JET
Apache Gearpump
SDKs
Execution Engines

Portability

Engine Portability

- Runners can translate a Beam pipeline for any of these execution engines

Language Portability

- Beam pipeline can be generated from any of these language

Engine Portability

1. Write your Pipeline
2. Set the Runner

```
options.setRunner(FlinkRunner.class);
```

or

```
--runner=FlinkRunner / --runner=SparkRunner
```

3. Run!

```
p.run();
```

Portability

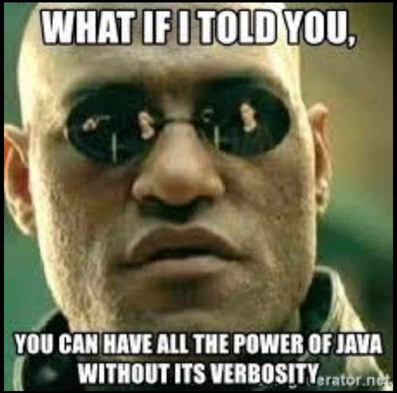
Engine Portability

- Runners can translate a Beam pipeline for any of these execution engines

Language Portability

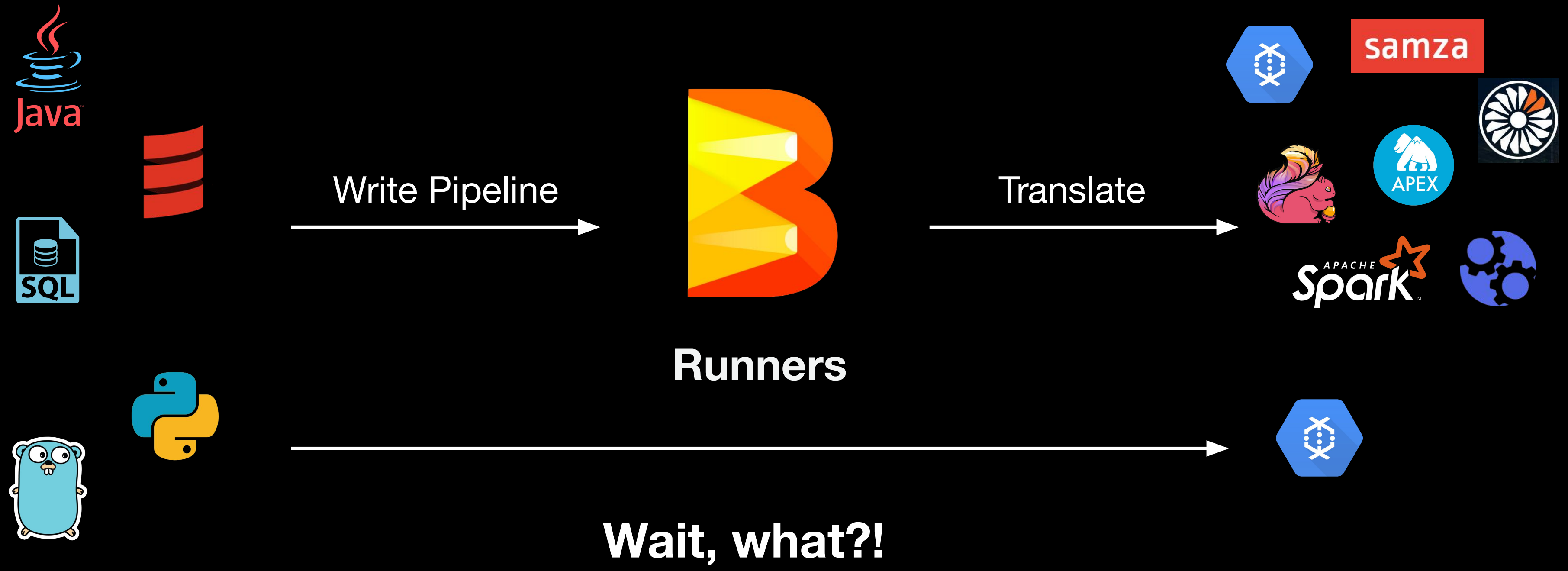
- Beam pipeline can be generated from any of these language

Why Use Another Language?



- Syntax / Expressiveness
- Code reuse
- Ecosystem: Libraries, Tools (!)
- Communities (Yes!)

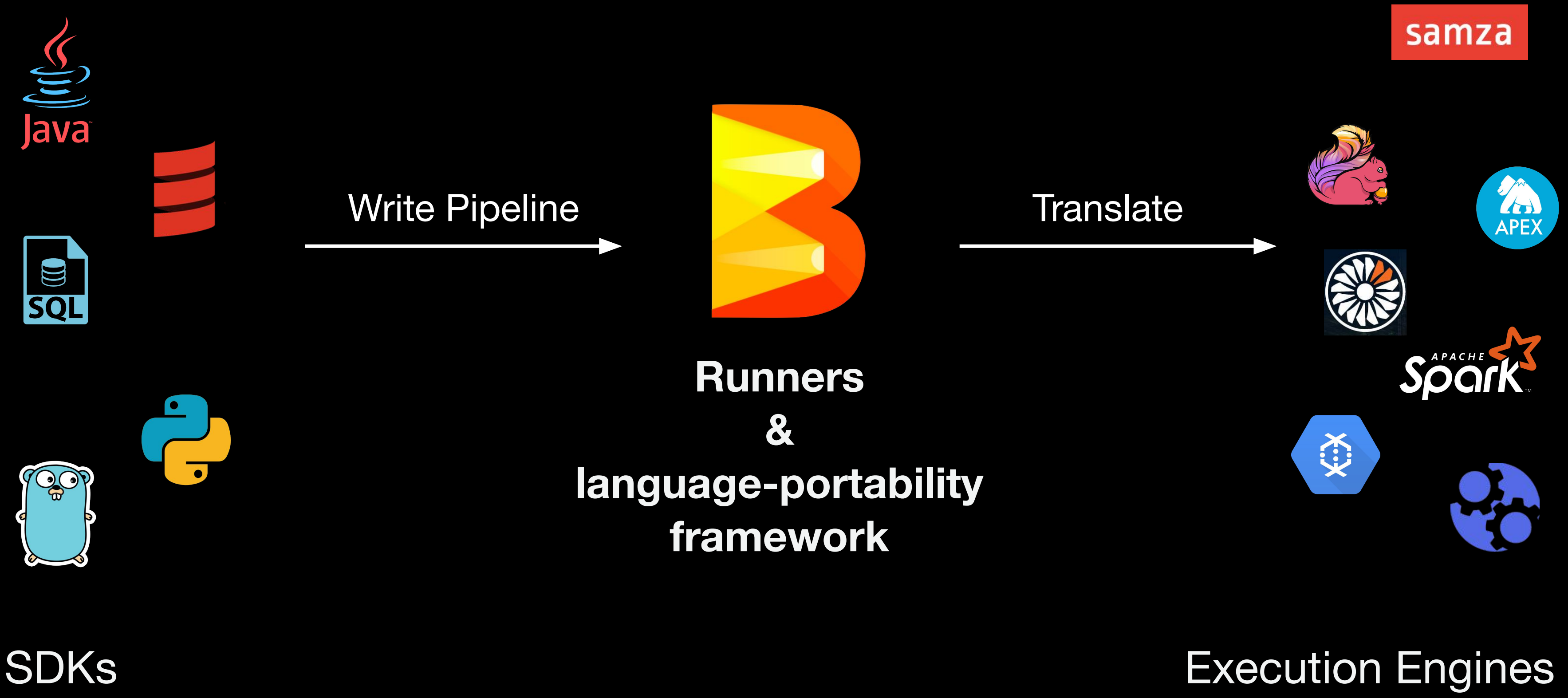
Beam without Language-Portability



SDKs

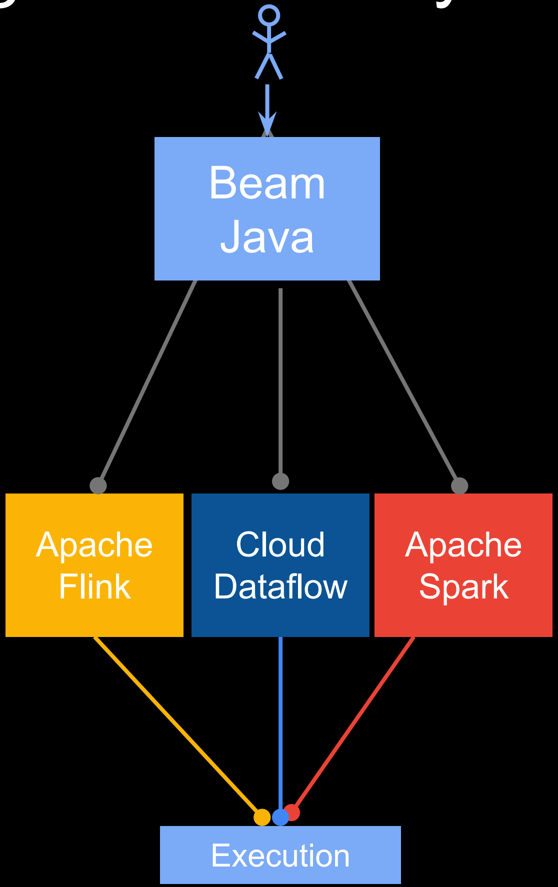
Execution Engines

Beam with Language-Portability



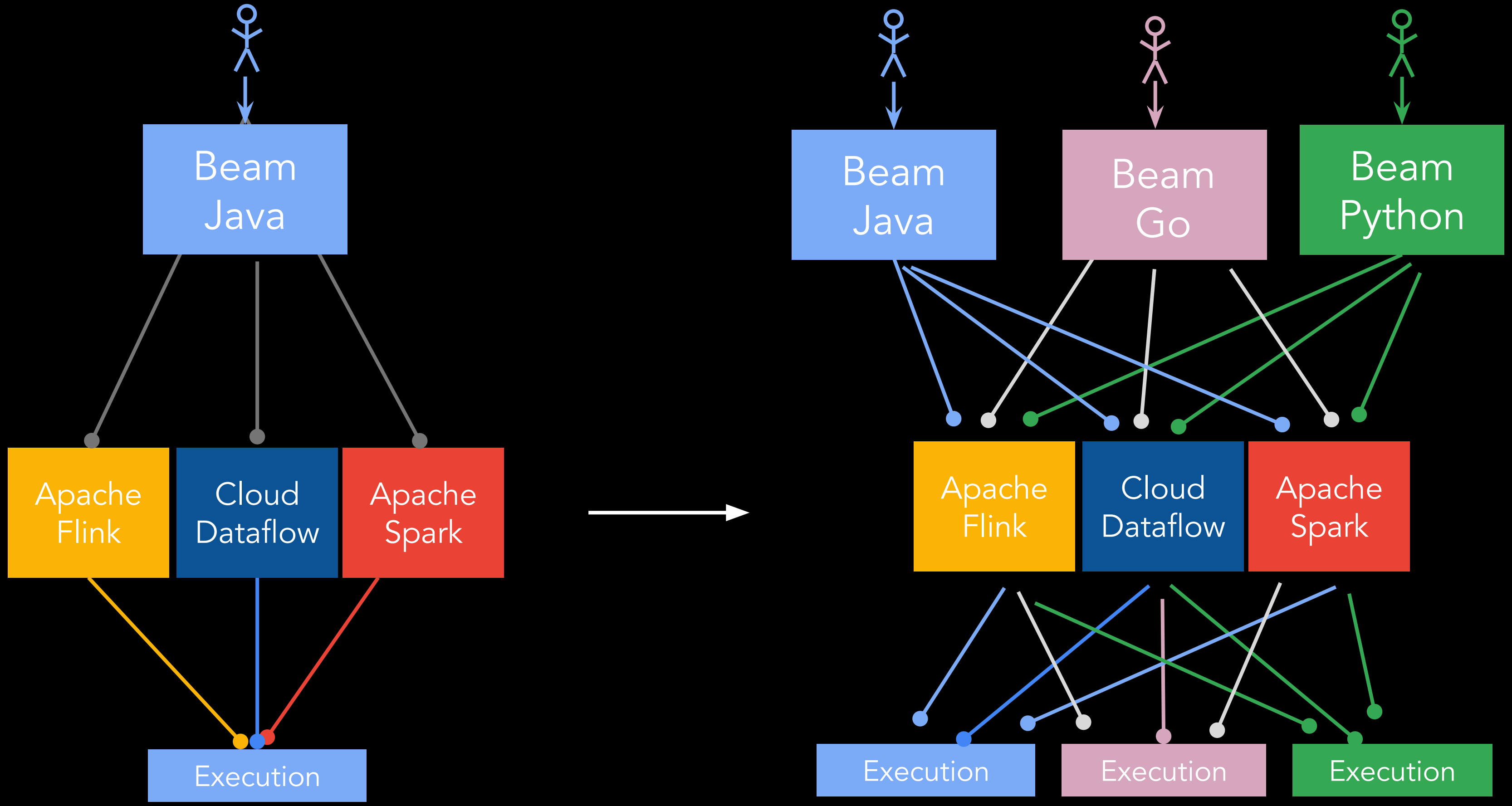
How Does It Work?

Engine Portability

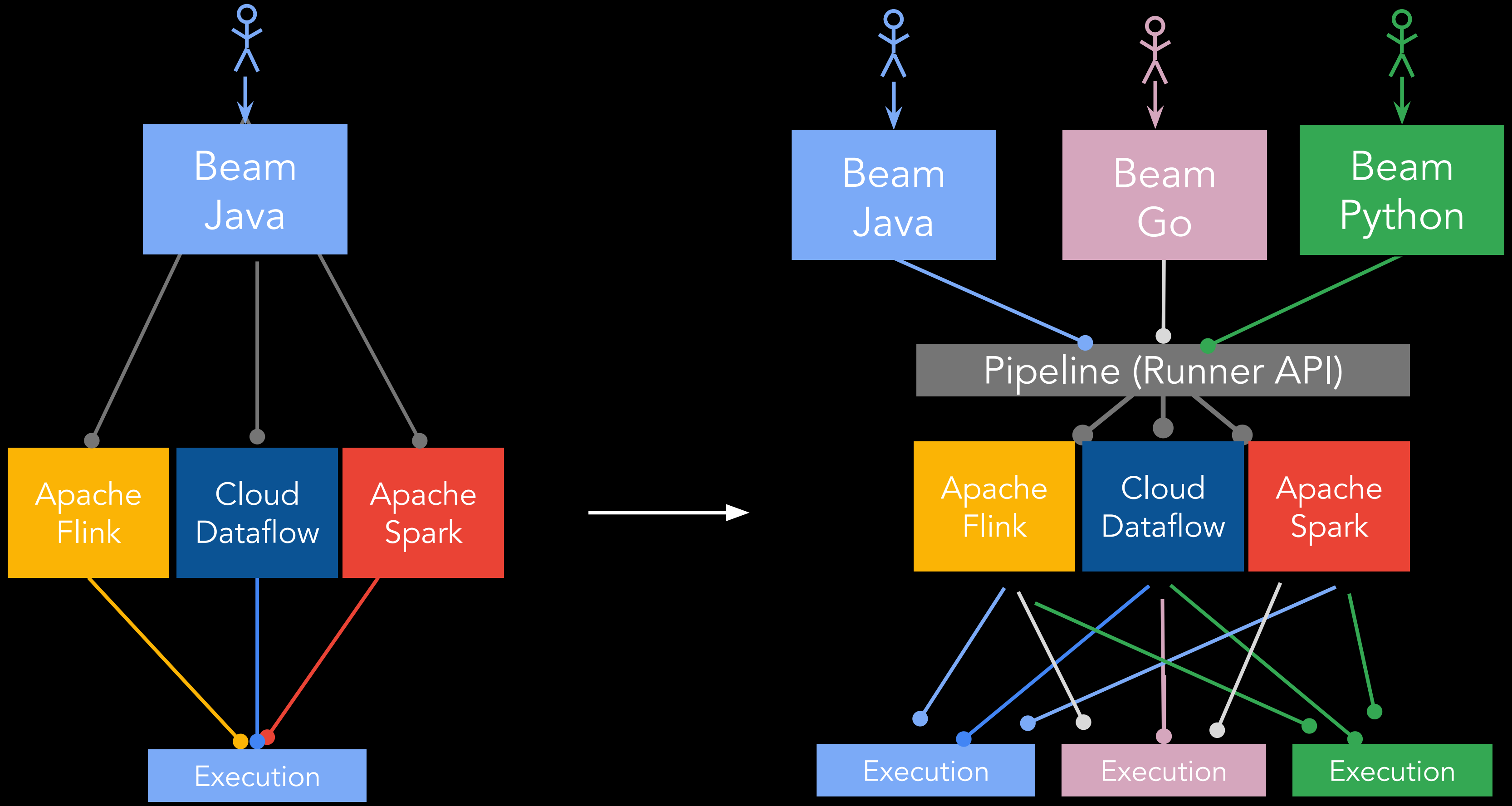


Primitive Transforms
ParDo
GroupByKey
Assign Windows
Flatten
Sources

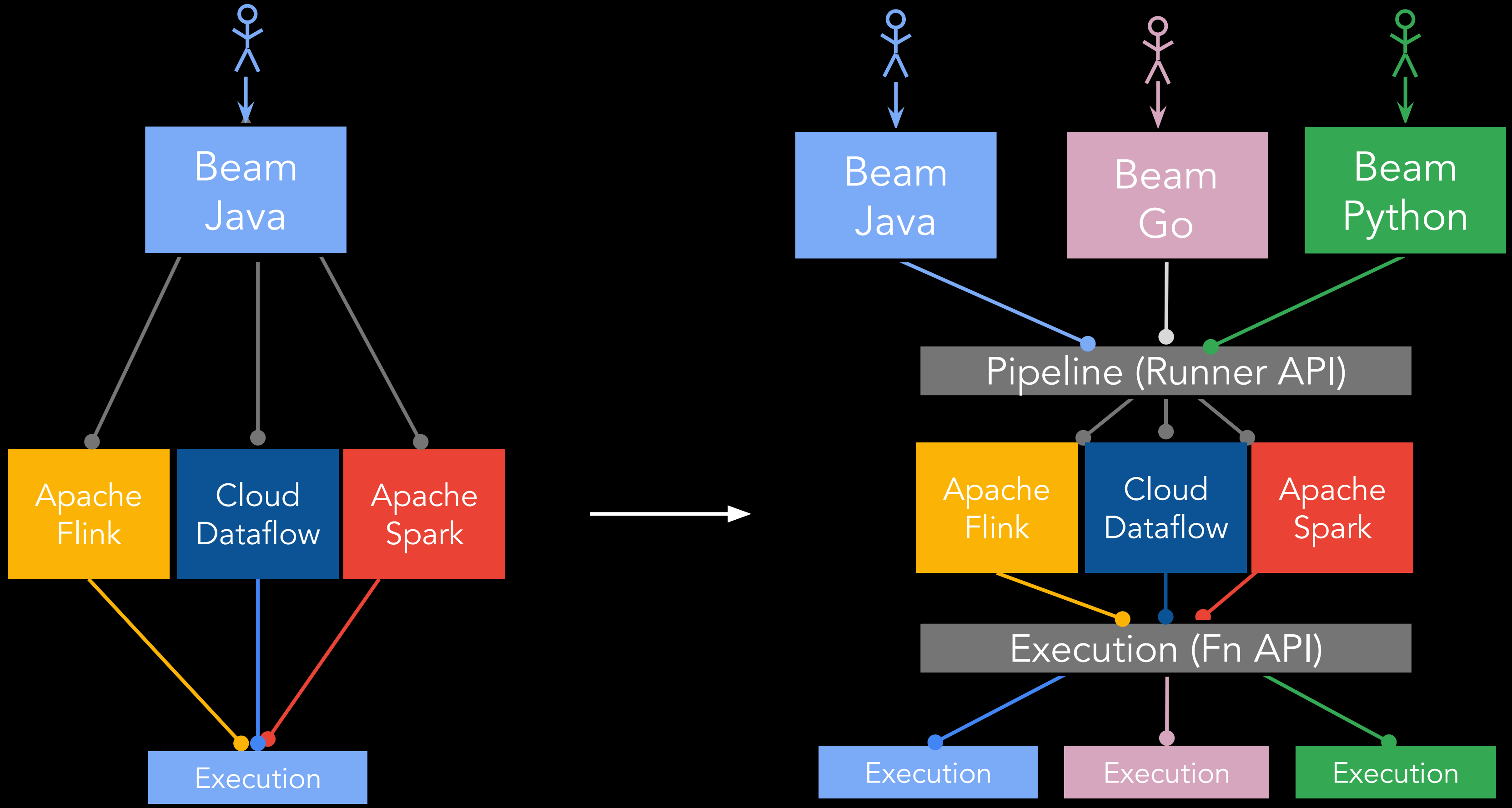
Language Portability



Language Portability

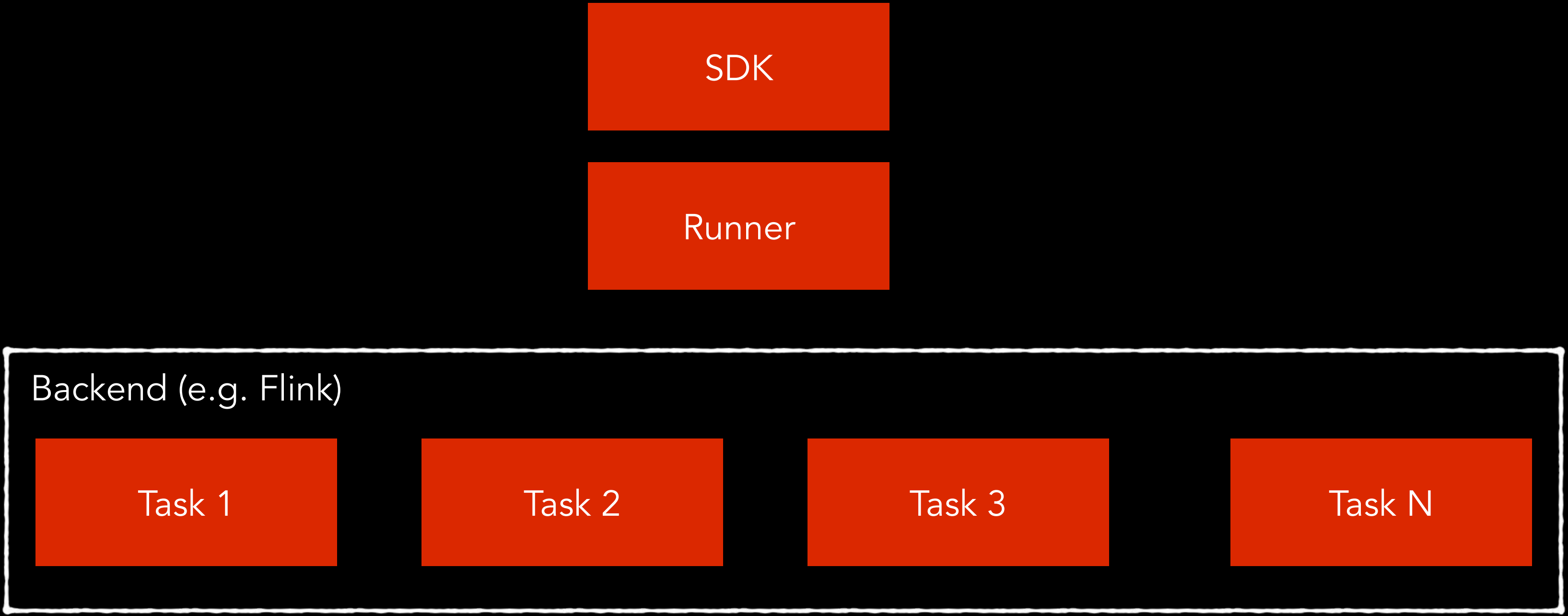


Language Portability



Engine Portability

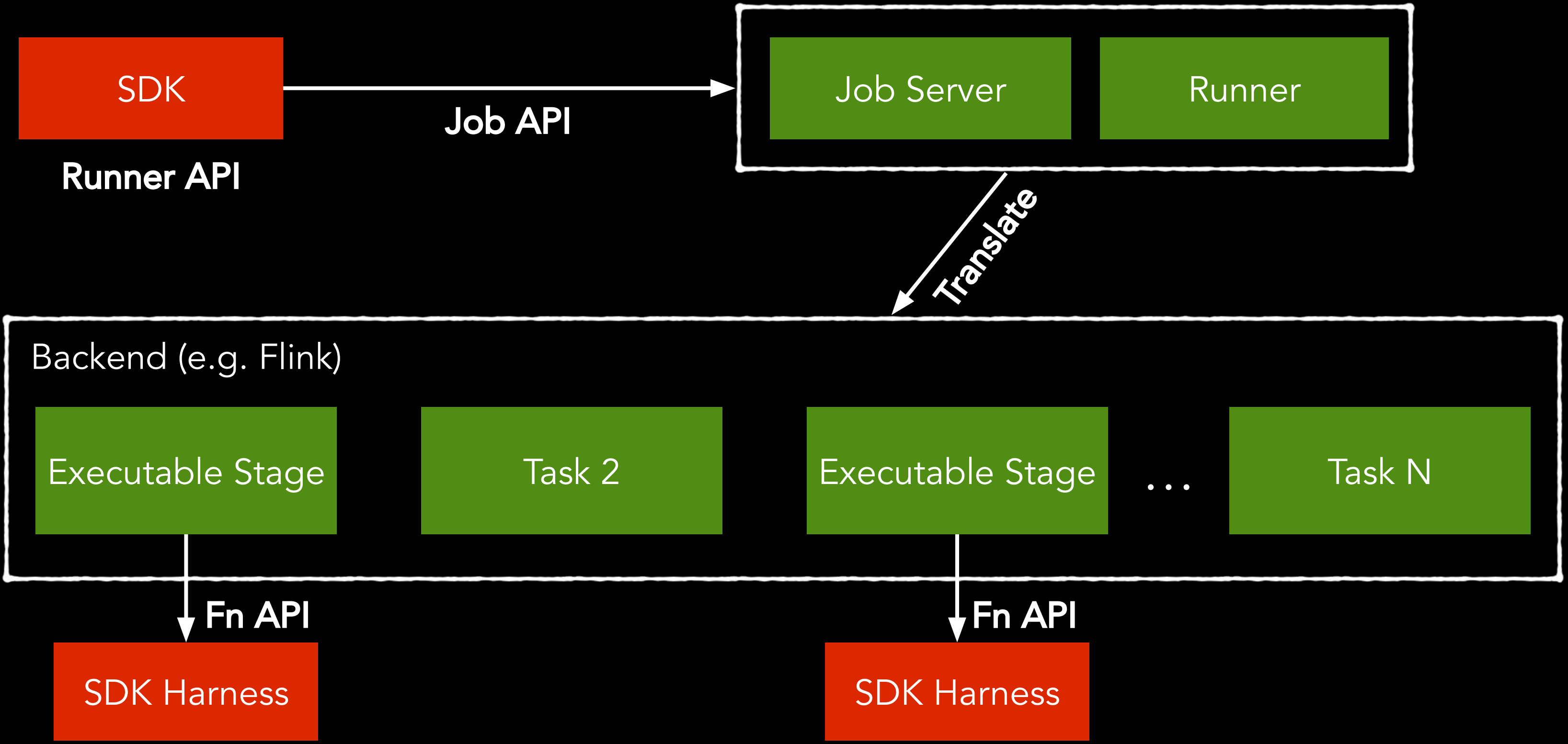
🟥 language-specific



All components are tight to a single language

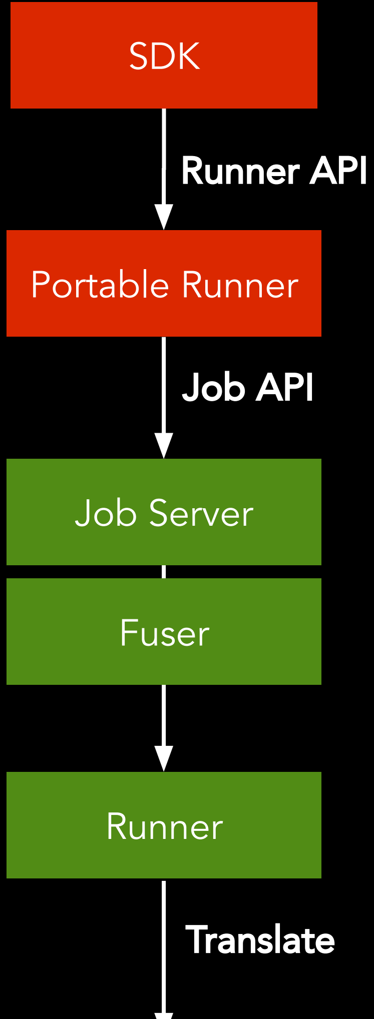
Language Portability Architecture

🟥 language-specific
🟩 language-agnostic



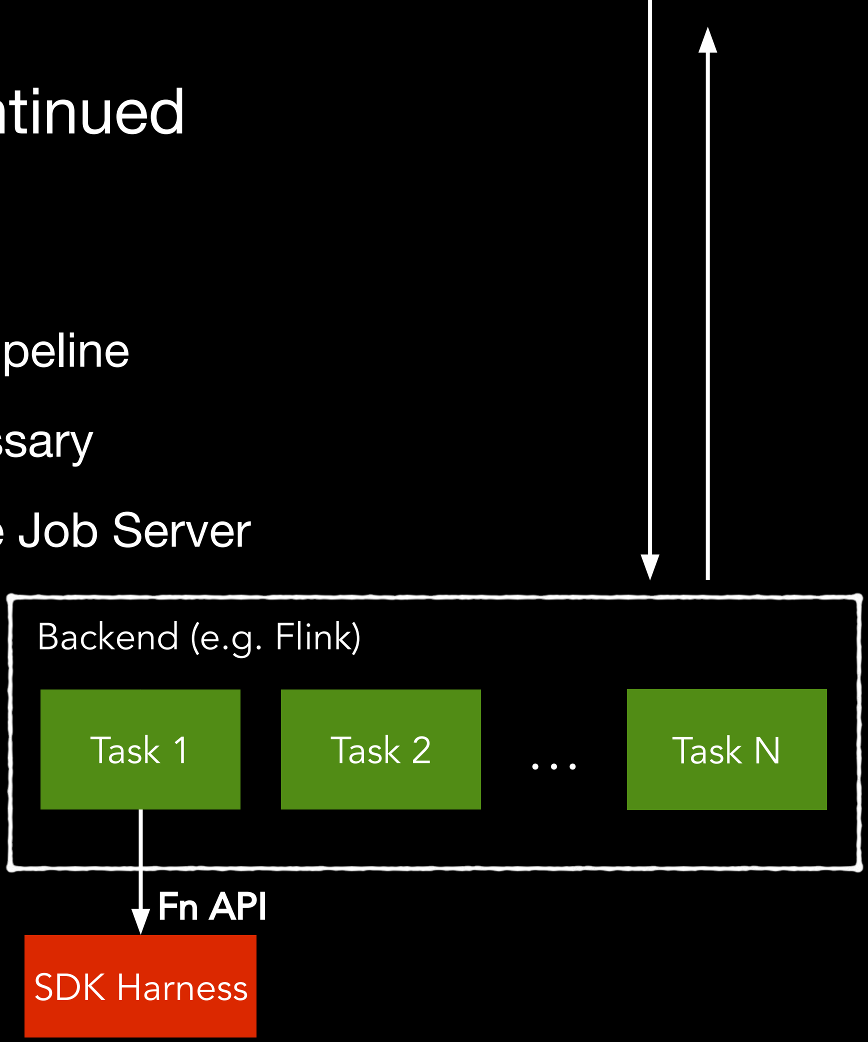
From Pipeline to Execution

1. Pipeline is serialized to the ProtoBuf Runner API
2. Protobuf message is send over via the Job API
3. Staging prepares execution dependencies for Fn API
4. Job Server fuses the pipeline and calls the actual Runner
5. Runner translates and submits to its execution engine



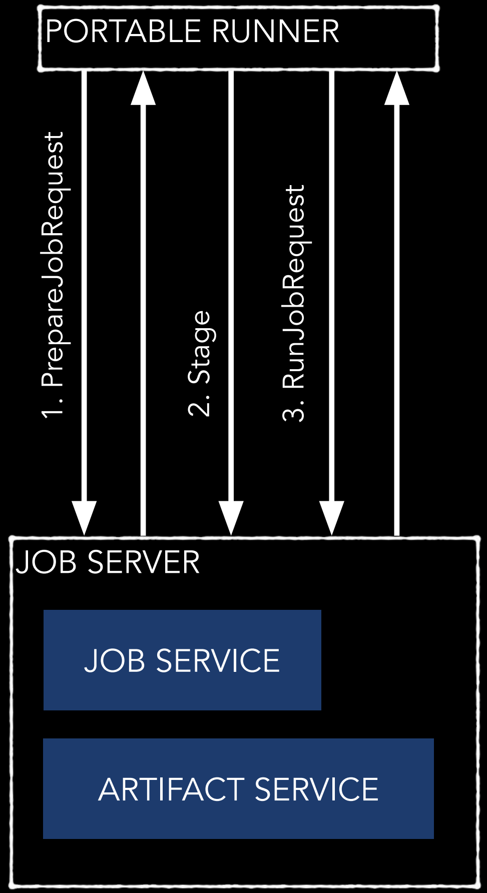
From Pipeline to Execution / continued

6. Execution engine executes translated pipeline
7. SDK harness is utilized whenever necessary
8. Execution status is reported back to the Job Server



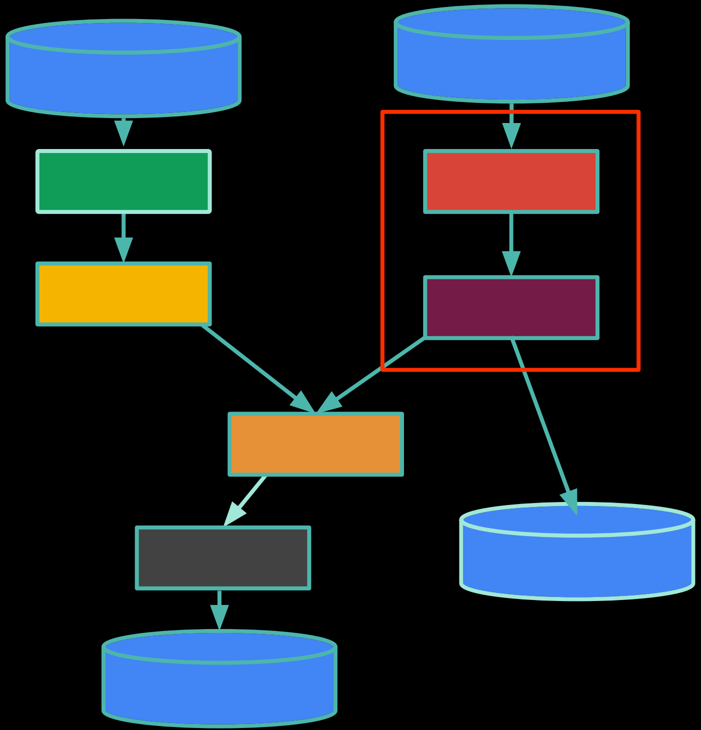
Portable Runner / Job Server

- Each SDK has an additional Portable Runner
  - Portable Runner takes care of talking to the JobService
- Each backend has its own submission endpoint
  - Consistent language-independent way for pipeline submission and monitoring
  - Stage files for SDK harness



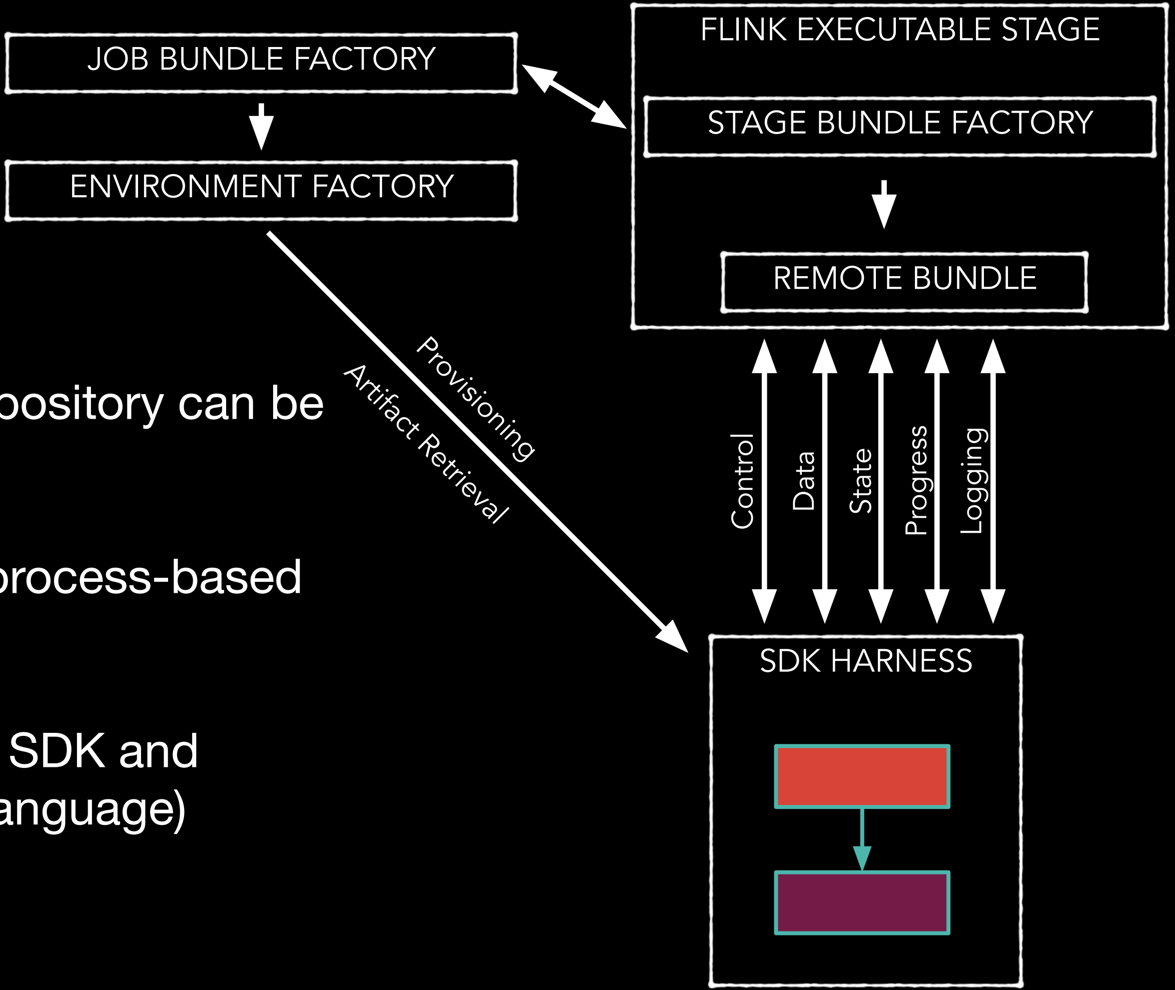
Pipeline Fusion

- SDK Harness environment comes at a cost
  - Serialization step before and after processing with SDK harness
- User defined functions should be chained and share the same environment



SDK Harness

- SDK Harness runs
  - in a Docker container (repository can be specified)
  - in a dedicated process (process-based execution)
  - embedded (only works if SDK and Runner share the same language)



Primitive Transforms

- Did we have to rewrite the old Runners?
  Good news, we can re-use most of the code

- There are, however, four different translators for the Flink Runner
  - Legacy Batch/Streaming
  - Portable Batch/Streaming

- And three different translators for Spark runner
  - Legacy Batch/Streaming
  - Portable Batch

Transforms	
Classic	**Portable**
ParDo	ExecutableStage
GroupByKey	
Assign Windows	ExecutableStage
Flatten	
Sources	Impulse + SDF

The IO Problem

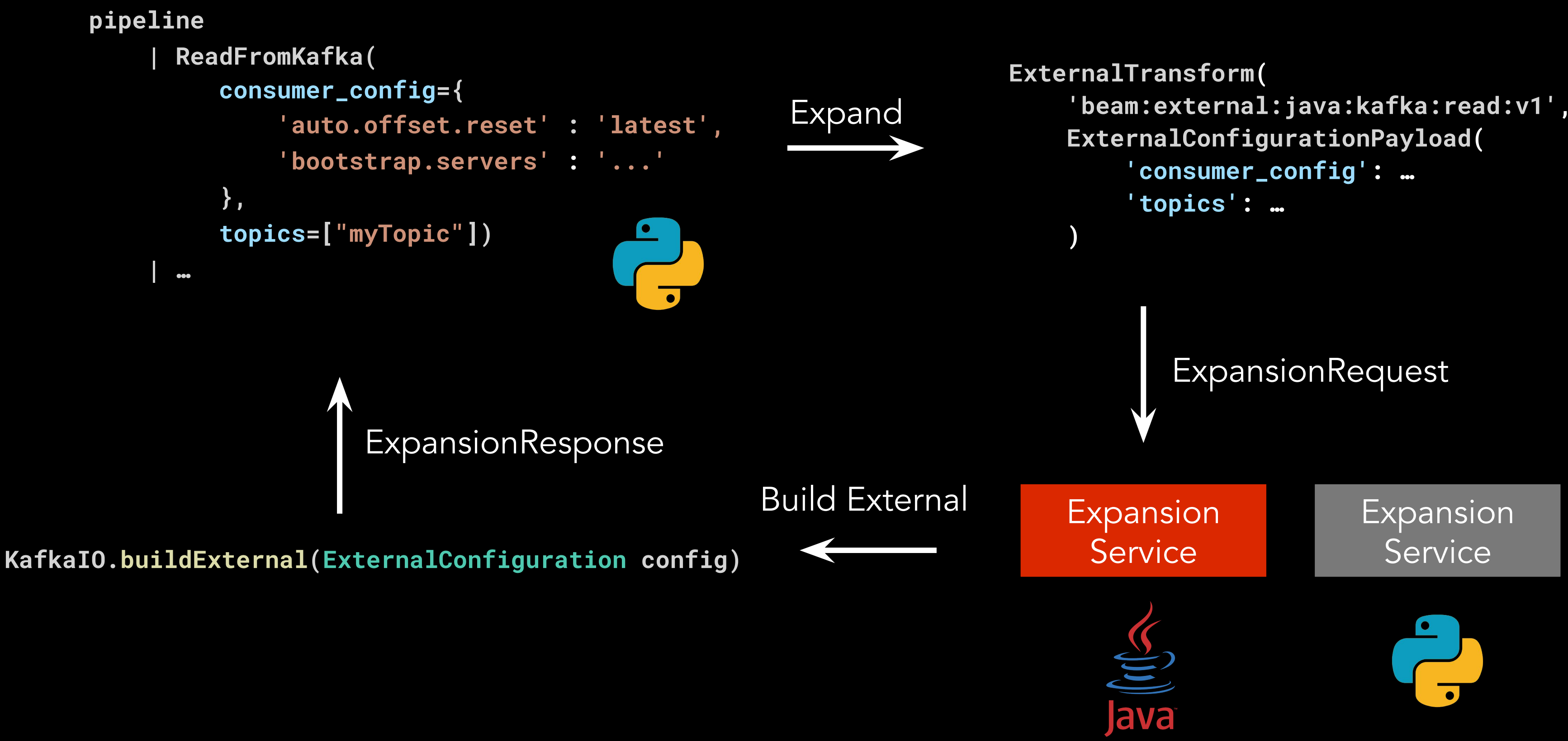
- Java SDK has rich set of IO connectors, e.g. FileIO, KafkaIO, PubSubIO, JDBC, Cassandra, Redis, ElasticsearchIO, …
- Python SDK has replicated parts of it, i.e. FileIO
  - Are we going to replicate all the others?
  - Solution: Use cross-language pipelines!

File-based
Apache HDFS
Amazon S3
Google Cloud Storage
Local Filesystems
AvroIO
TextIO
TFRecordIO
XmlIO
TikaIO
ParquetIO

Messaging
Amazon Kinesis
Amazon SNS / SQS
AMQP
Apache Kafka
Google Cloud Pub/Sub
JMS
MQTT

Databases
Amazon DynamoDB
Apache Cassandra
Apache Hadoop InputFormat
Apache HBase
Apache Hive (HCatalog)
Apache Kudu
Apache Solr
Elasticsearch
Google BigQuery
Google Bigtable
Google Datastore
Google Spanner
JDBC
MongoDB
Redis

Cross-Language Pipelines

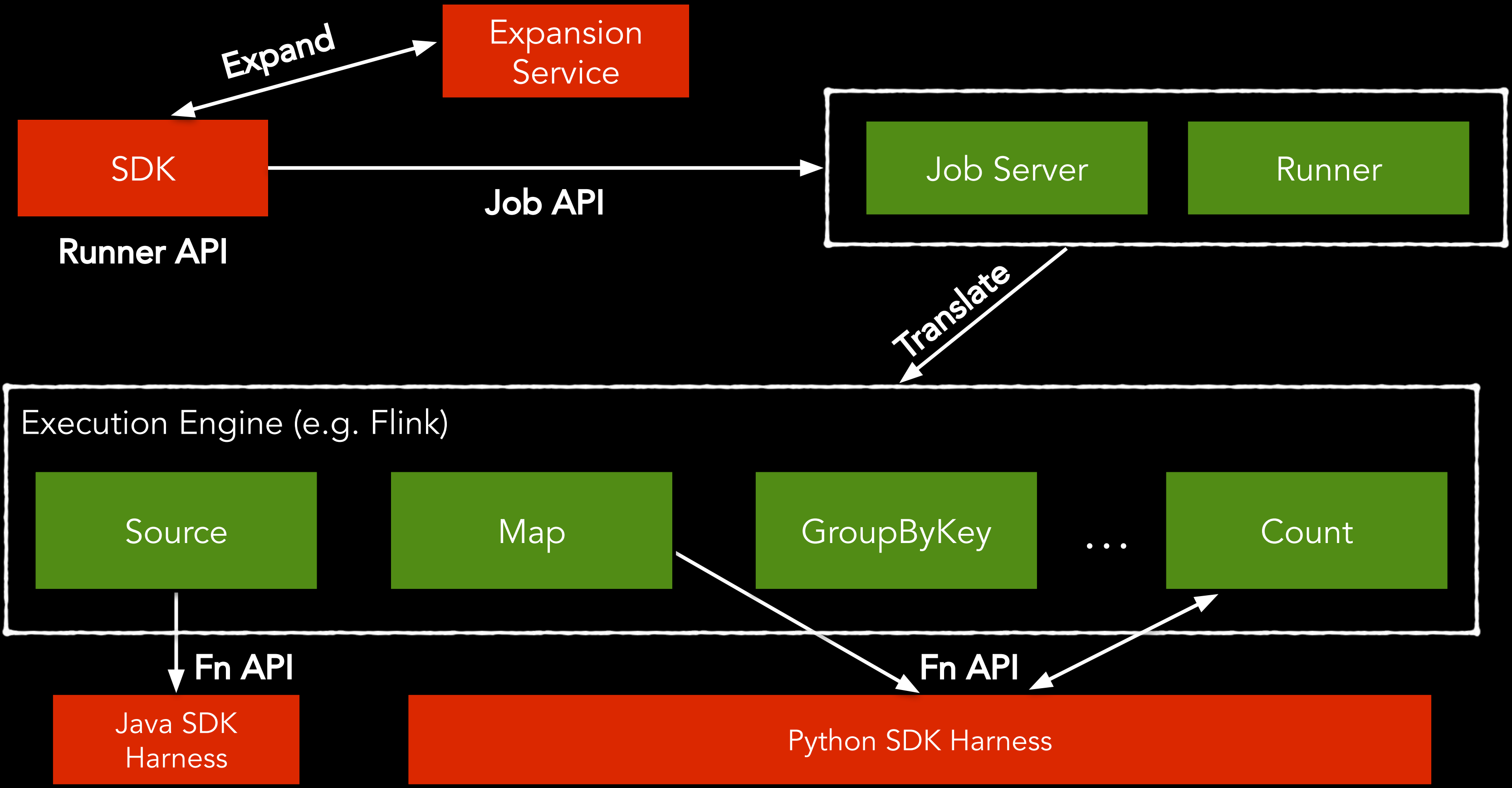


```
pipeline
    | ReadFromKafka(
        consumer_config={
            'auto.offset.reset' : 'latest',
            'bootstrap.servers' : '...'
        },
        topics=["myTopic"])
    | …
```

```
ExternalTransform(
    'beam:external:java:kafka:read:v1',
    ExternalConfigurationPayload(
        'consumer_config': …
        'topics': …
    )
```

```
KafkaIO.buildExternal(ExternalConfiguration config)
```

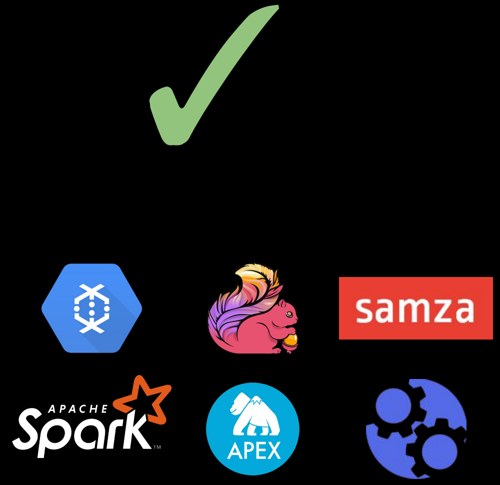
Cross-Language with Multiple Environments



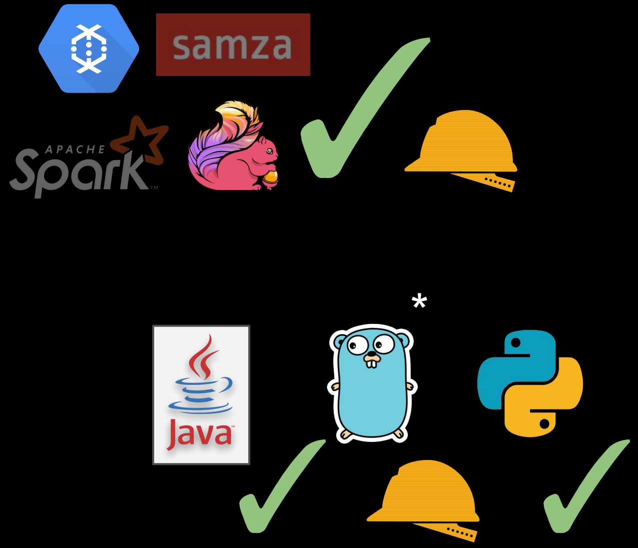
Outlook

Status of Portability

Engine Portability

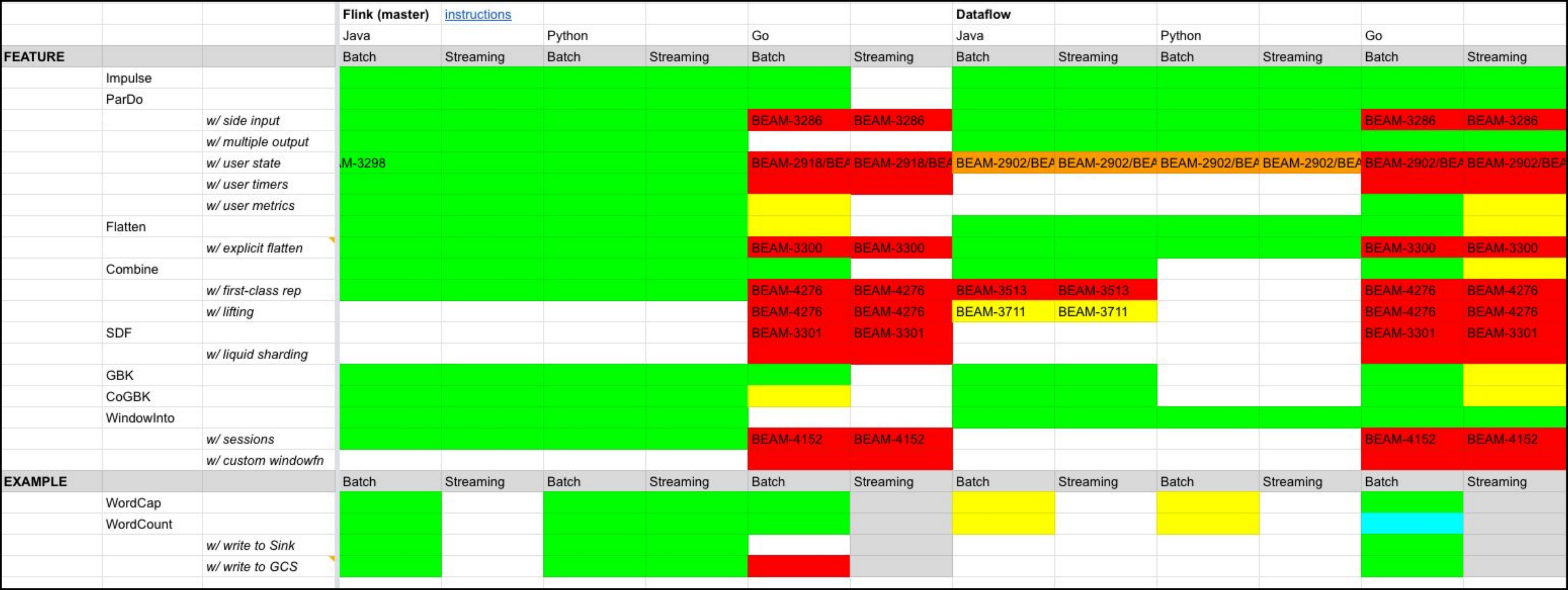


Language Portability



* See Robert Burke's talk directly after this talk

Portability Support Matrix



			Flink (master)	instructions					**Dataflow**					
			Java		Python		Go		Java		Python		Go	
FEATURE			Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming
	Impulse		🟩	🟩	🟩	🟩	🟩		🟩	🟩	🟩	🟩	🟩	🟩
	ParDo		🟩	🟩	🟩	🟩	🟩		🟩	🟩	🟩	🟩	🟩	🟩
		w/ side input	🟩	🟩	🟩	🟩	🟥 BEAM-3286	🟥 BEAM-3286	🟩	🟩	🟩	🟩	🟥 BEAM-3286	🟥 BEAM-3286
		w/ multiple output	🟩	🟩	🟩	🟩			🟩	🟩	🟩	🟩	🟩	🟩
		w/ user state	🟩 M-3298	🟩	🟩	🟩	🟥 BEAM-2918/BEA	🟥 BEAM-2918/BEA	🟧 BEAM-2902/BEA	🟧 BEAM-2902/BEA	🟧 BEAM-2902/BEA	🟧 BEAM-2902/BEA	🟥 BEAM-2902/BEA	🟥 BEAM-2902/BEA
		w/ user timers	🟩	🟩	🟩	🟩	🟥	🟥					🟥	🟥
		w/ user metrics	🟩	🟩	🟩	🟩	🟨						🟩	🟨
	Flatten		🟩	🟩	🟩	🟩	🟨		🟩	🟩	🟩	🟩	🟩	🟨
		w/ explicit flatten	🟩	🟩	🟩	🟩	🟥 BEAM-3300	🟥 BEAM-3300	🟩	🟩	🟩	🟩	🟥 BEAM-3300	🟥 BEAM-3300
	Combine		🟩	🟩	🟩	🟩	🟩		🟩	🟩			🟩	🟨
		w/ first-class rep	🟩	🟩	🟩	🟩	🟥 BEAM-4276	🟥 BEAM-4276	🟥 BEAM-3513	🟥 BEAM-3513			🟥 BEAM-4276	🟥 BEAM-4276
		w/ lifting					🟥 BEAM-4276	🟥 BEAM-4276	🟨 BEAM-3711	🟨 BEAM-3711			🟥 BEAM-4276	🟥 BEAM-4276
	SDF						🟥 BEAM-3301	🟥 BEAM-3301					🟥 BEAM-3301	🟥 BEAM-3301
		w/ liquid sharding					🟥	🟥					🟥	🟥
	GBK		🟩	🟩	🟩	🟩	🟩		🟩	🟩			🟩	🟨
	CoGBK		🟩	🟩	🟩	🟩	🟨		🟩	🟩			🟩	🟨
	WindowInto		🟩	🟩	🟩	🟩			🟩	🟩	🟩	🟩	🟩	🟩
		w/ sessions	🟩	🟩	🟩	🟩	🟥 BEAM-4152	🟥 BEAM-4152					🟥 BEAM-4152	🟥 BEAM-4152
		w/ custom windowfn					🟥	🟥					🟥	🟥
EXAMPLE			Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming
	WordCap		🟩		🟩	🟩	🟩	▨	🟨		🟨		🟩	▨
	WordCount		🟩		🟩	🟩	🟩	▨	🟨		🟨		🟦	▨
		w/ write to Sink	🟩		🟩	🟩		▨					🟩	▨
		w/ write to GCS	🟩		🟩	🟩	🟥	▨					🟩	▨

https://s.apache.org/apache-beam-portability-support-table

Limitations and Pending Work

- Implement all Fn API in all Runners
- Splittable DoFn
- Improve Go support
- Concurrency model for the SDK harness
- Performance tuning
- Publish Docker Images
- Artifact Staging in cross-language pipelines
-

Getting Started

Getting Started With the Python SDK

1. Prerequisite

   a. Setup virtual env
   ```
   virtualenv env && source env/bin/activate
   ```

   b. Install Beam SDK
   ```
   pip install apache_beam # if you are on a release
   # if you want to use the latest master version
   ./gradlew :sdks:python:python:sdist
   cd sdks/python/build
   python setup.py install
   ```

   c. Build SDK Harness Container
   ```
   ./gradlew :sdks:python:container:docker
   ```

   d. Start JobServer
   ```
   ./gradlew :runners:flink:1.8:job-server:runShadow
   -PflinkMasterUrl=localhost:8081 # Add if you want to submit to a Flink cluster
   ```

See also https://beam.apache.org/contribute/portability/

Getting Started With the Python SDK

2. Develop your Beam pipeline
3. Run with Direct Runner (testing)
4. Run with Portable Runner

```
# required args
--runner=PortableRunner --job_endpoint=localhost:8099

# other args
--streaming
--parallelism=4
--<option_arg>=<option_value>
```

Refs.
https://beam.apache.org/documentation/runners/flink/
https://beam.apache.org/documentation/runners/spark/

Thank You!

- **Visit** beam.apache.org/contribute/portability/
- **Subscribe** to the mailing lists:

  user-subscribe@beam.apache.org

  dev-subscribe@beam.apache.org
- **Join** the ASF Slack channel #beam-portability

Follow @ApacheBeam @stadtlegende @iemejia

- **Attend** Beam Summit Europe June 19-20 (!)

References

https://s.apache.org/beam-runner-api

https://s.apache.org/beam-runner-api-combine-model

https://s.apache.org/beam-fn-api

https://s.apache.org/beam-fn-api-processing-a-bundle

https://s.apache.org/beam-fn-state-api-and-bundle-processing

https://s.apache.org/beam-fn-api-send-and-receive-data

https://s.apache.org/beam-fn-api-container-contract

https://s.apache.org/beam-portability-timers