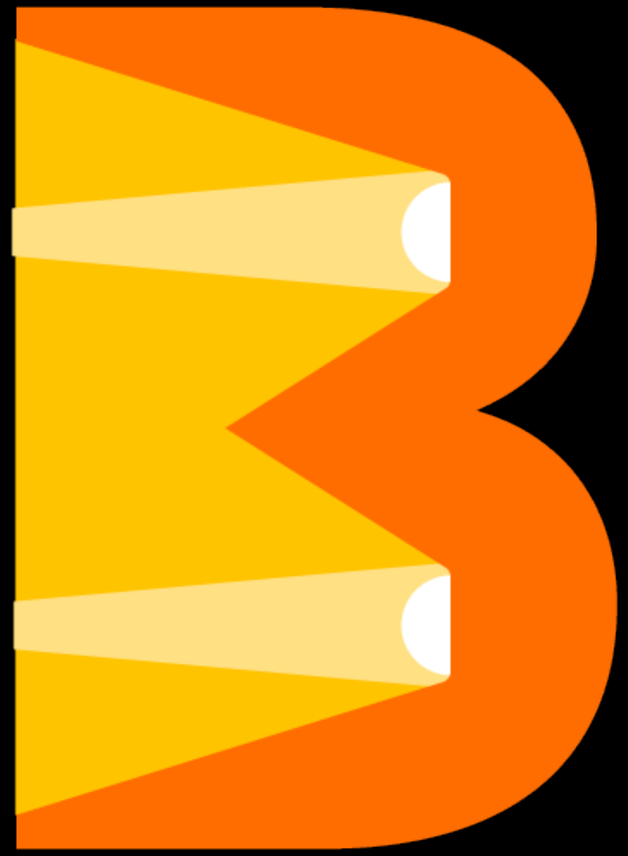
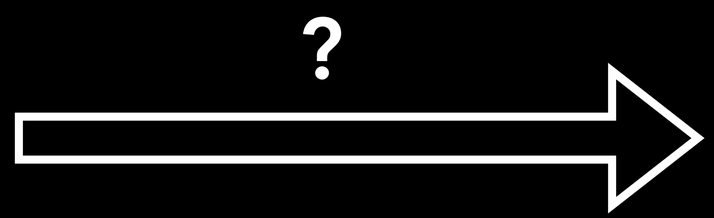
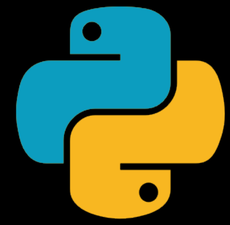
FROM ZERO TO PORTABILITY







APACHE BEAM'S JOURNEY TO CROSS-LANGUAGE DATA PROCESSING

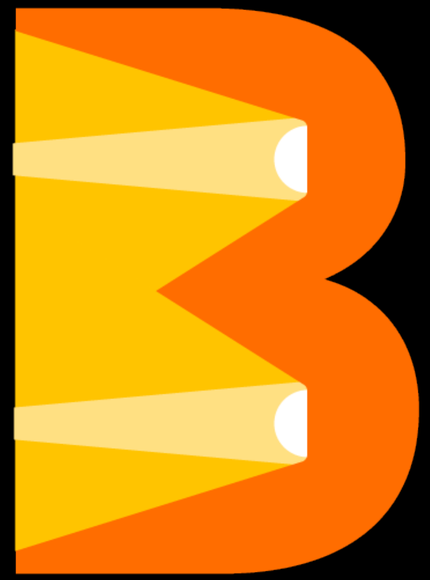
FOSDEM 2019

Maximilian Michels
mxm@apache.org
@stadtlegende
maximilianmichels.com

- What is Beam?
- What does portability mean?
- How do we achieve portability?
- Are we there yet?

WHAT IS BEAM?

- Apache open-source project
- Parallel/distributed data processing
- Unified programming model for batch/stream processing
- Execution engine of your choice ("Uber API")
- Programming language of your choice



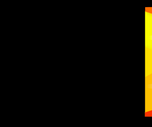
Apache Beam

BEAM VISION

Write Pipeline

Runners

Translate



Direct

Apache Samza

Apache Flink

Google Cloud Dataflow

Apache Apex

Apache Spark

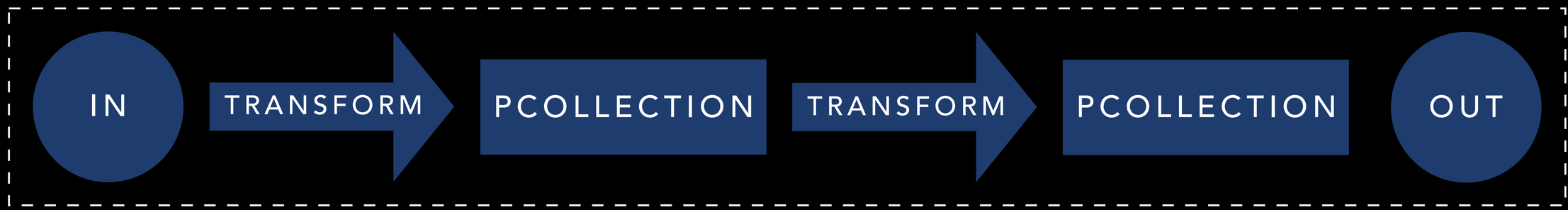
Apache Gearpump

Apache Nemo (incubating)

SDKs

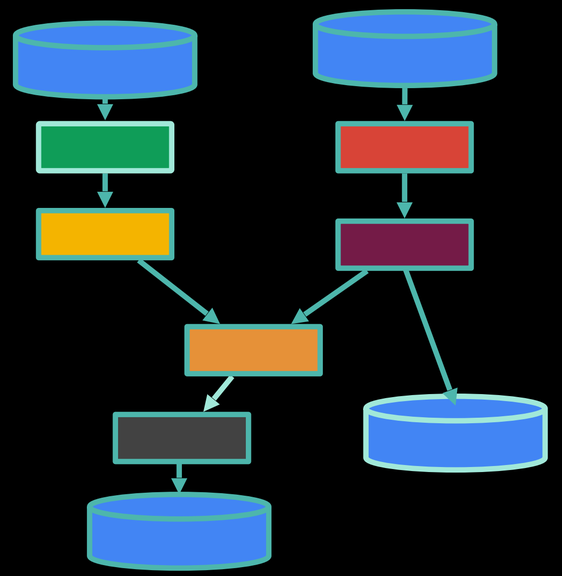
Execution Engines

THE BEAM API



Pipeline

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
- Only small set of primitive transforms
- Runners can support specialized translation of composite transforms, but don't have to

PRIMITIVE TRANSFORMS

ParDo

GroupByKey

AssignWindows

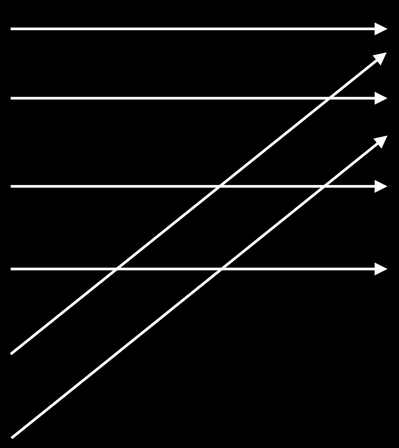
Flatten

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>



"Map/Reduce Phase"

GroupByKey

KV<k,v>… -> KV<k, [v…]>

KV<“to”, [1,1]>
KV<“be”, [1,1]>
KV<“or”, [1]>
KV<“not”,[1]>

"Shuffle Phase"

WORDCOUNT — RAW

```
pipeline
    .apply(Create.of("hello", "hello", "fosdem"))
    .apply(ParDo.of(
        new DoFn<String, KV<String, Integer>>() {
          @ProcessElement
          public void processElement(ProcessContext ctx) {
            KV<String, Integer> outputElement = KV.of(ctx.element(), 1);
            ctx.output(outputElement);
          }
        }))
    .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;
            }
            KV<String, Long> outputElement = KV.of(ctx.element().getKey(), count);
            ctx.output(outputElement);
          }
        }))
```

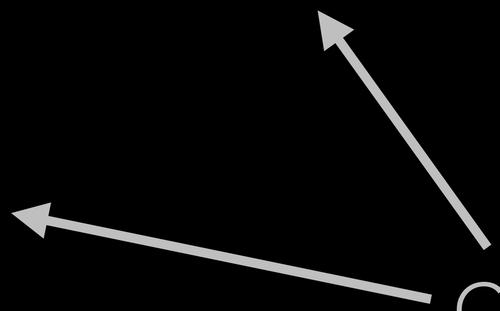
EXCUSE ME,
THAT WAS UGLY AS HELL

WORDCOUNT — COMPOSITE TRANSFORMS

```
pipeline
    .apply(Create.of("hello", "fellow", "fellow"))

    .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("hello", "fellow", "fellow"))
    .apply(Count.perElement());
```

Composite Transforms

PYTHON TO THE RESCUE

```
(p
 | beam.Create(['hello', 'hello', 'fosdem'])

 | beam.Map(lambda word: (word, 1))

 | beam.GroupByKey()

 | beam.Map(lambda kv: (kv[0], sum(kv[1])))
)
```

PYTHON TO THE RESCUE

```
(p
 | beam.Create(['hello', 'hello', 'fosdem'])

 | beam.Map(lambda word: (word, 1))

 | beam.CombinePerKey(sum)
)
```

THERE IS MUCH MORE TO BEAM

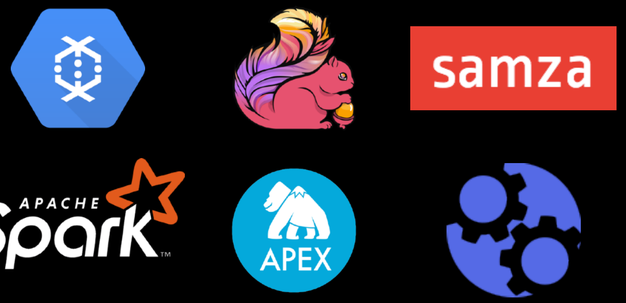
- Flatten/Combine/Partition/ CoGroupByKey (Join)
- Define your own transforms!
- IOs / Splittable DoFn
- Windowing
- Event Time / Processing Time
- Watermarks
- Side Inputs
- Multiple Outputs
- State
- Timers
- ...

- [x] What is Beam?
- [>] What does portability mean?
- [ ] How do we achieve portability?
- [ ] Are we there yet?

PORTABILITY

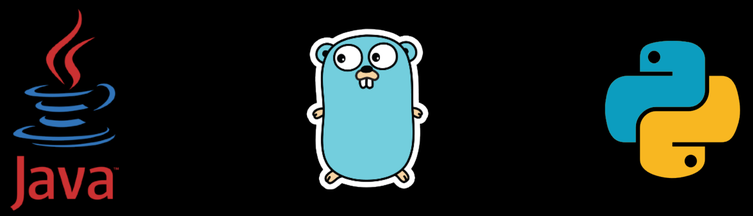
Engine Portability

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



Language Portability

- Beam pipeline can be generated from any of these language



BEAM VISION

Write Pipeline

Runners

Translate

SDKs

Execution Engines

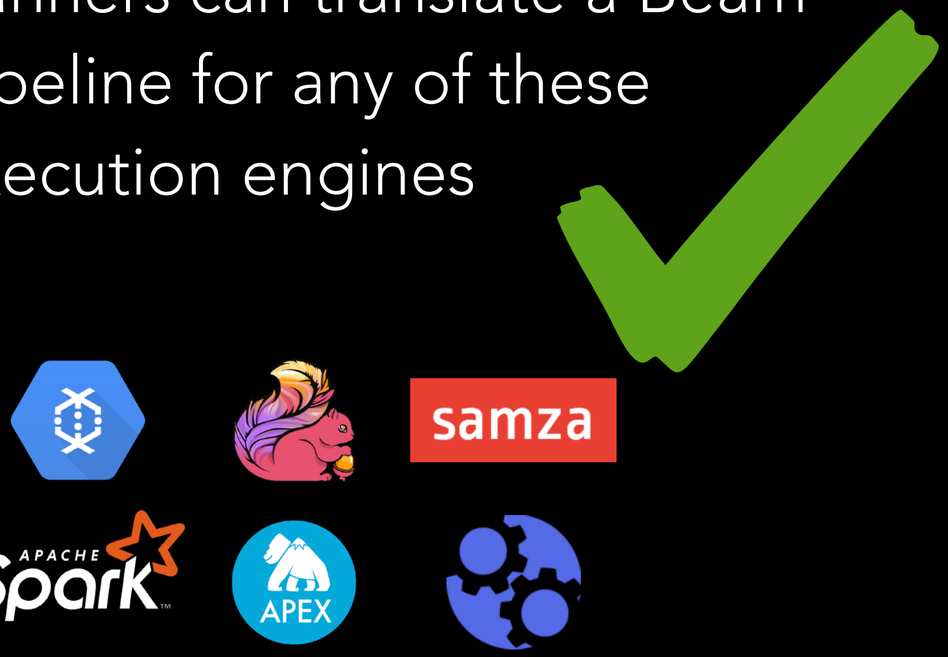
CROSS-ENGINE PORTABILITY

1. Set the Runner
   - `options.setRunner(FlinkRunner.class)`
   - `--runner=FlinkRunner`
2. Run!
   - `p.run()`

PORTABILITY

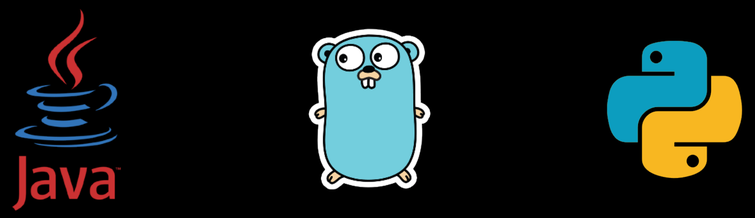
Engine Portability

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



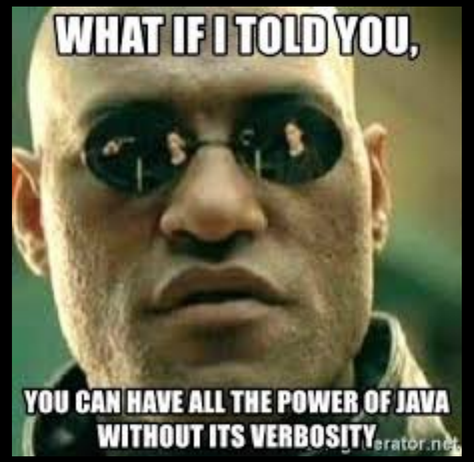
Language Portability

- Beam pipeline can be generated from any of these language



WHY WE WANT TO USE OTHER LANGUAGES

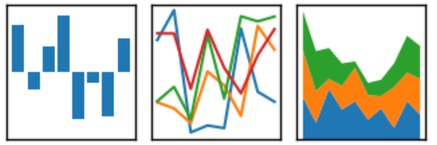
- Syntax / Expressiveness
- Communities (Yes!)
- Libraries (!)



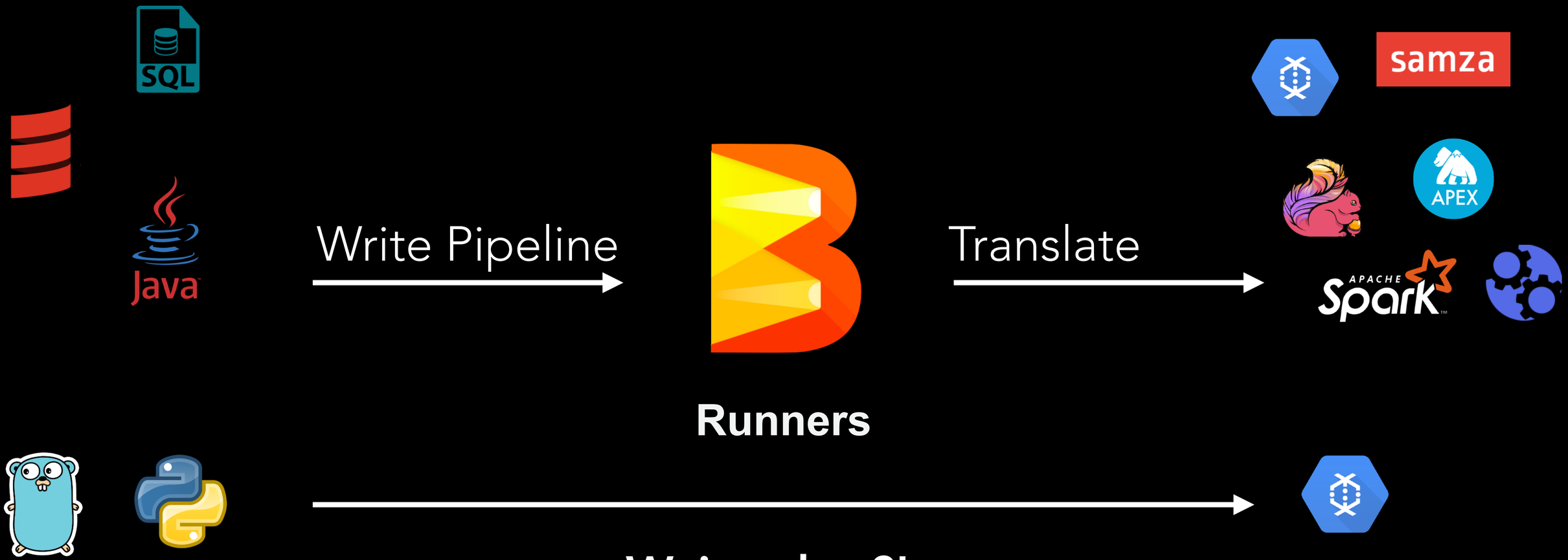
matplotlib

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



BEAM WITHOUT LANGUAGE-PORTABILITY



BEAM WITH LANGUAGE-PORTABILITY

Write Pipeline

Runners

&

language-portability framework

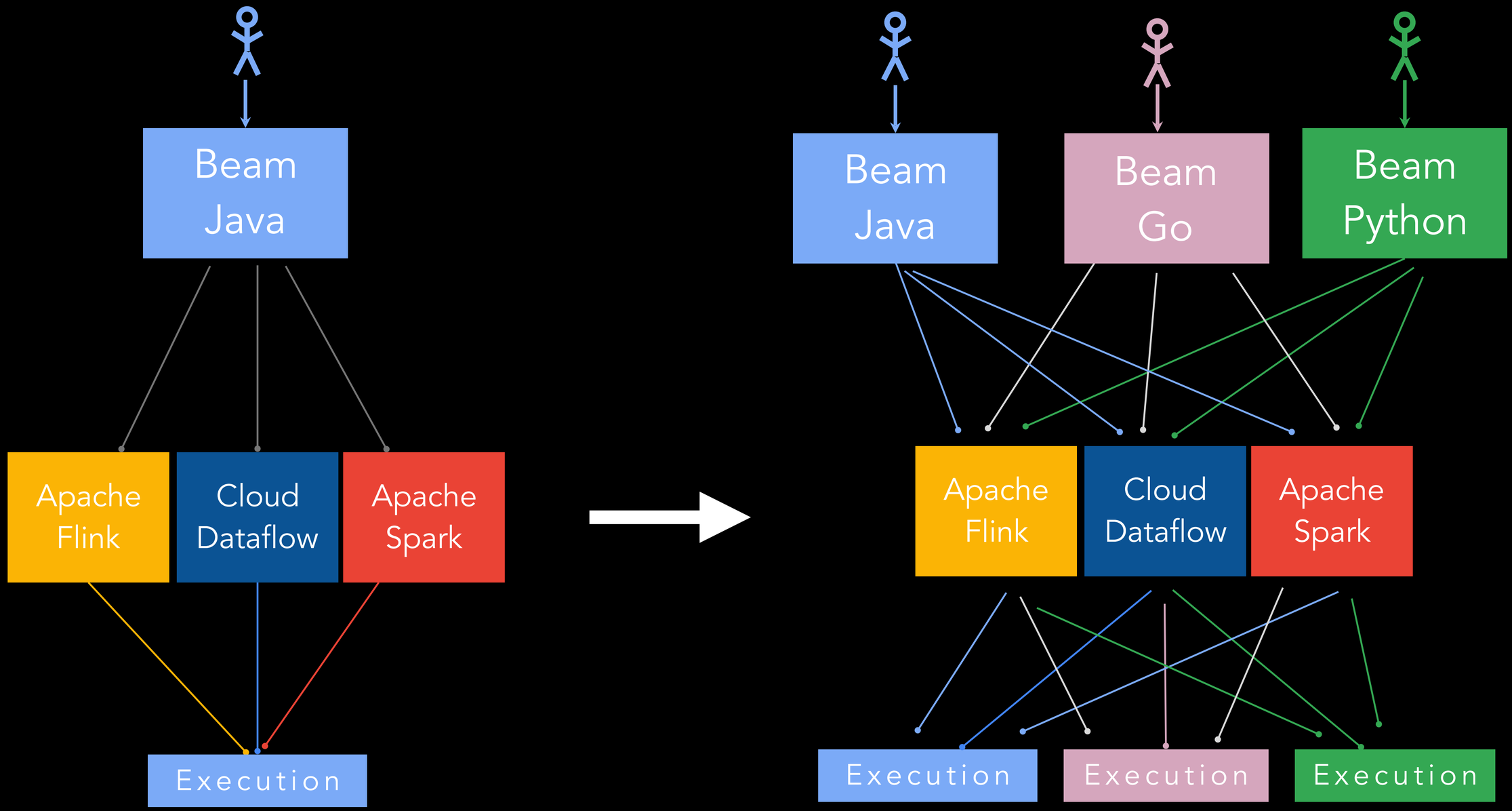
Translate

SDKs

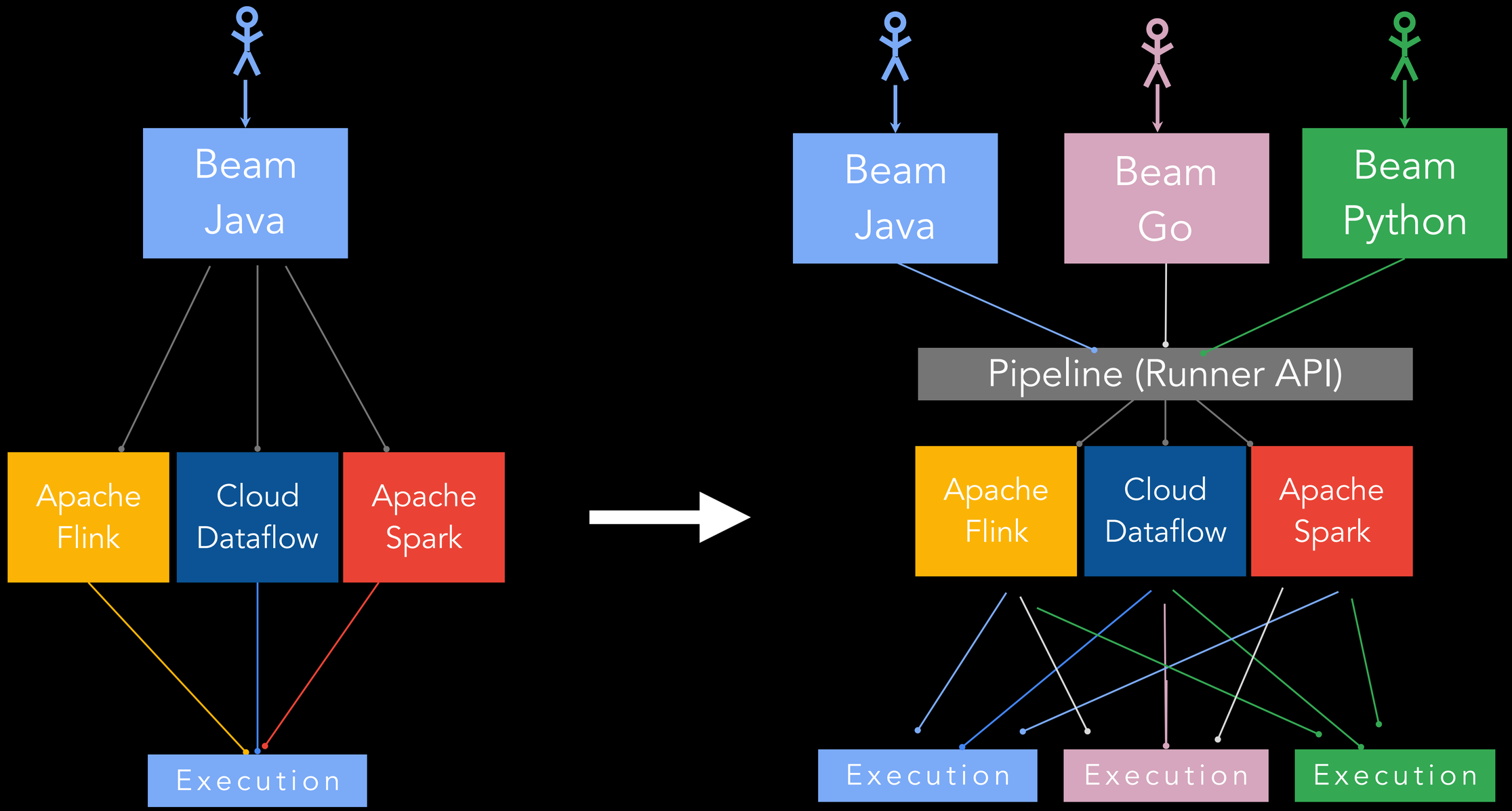
Execution Engines

- [x] What is Beam?
- [x] What does portability mean?
- [ ] How do we achieve portability?
- [ ] Are we there yet?

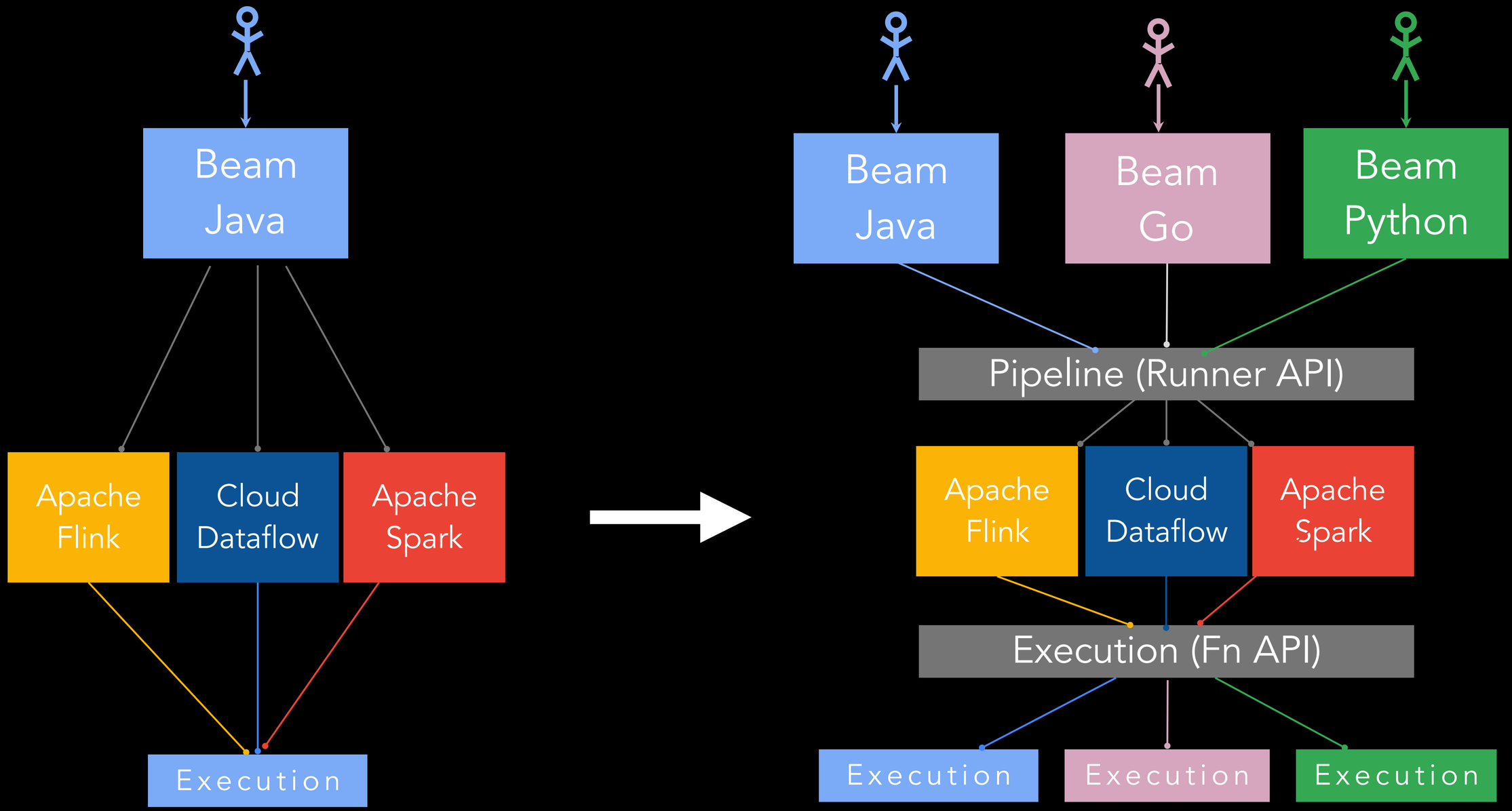
LANGUAGE-PORTABILITY



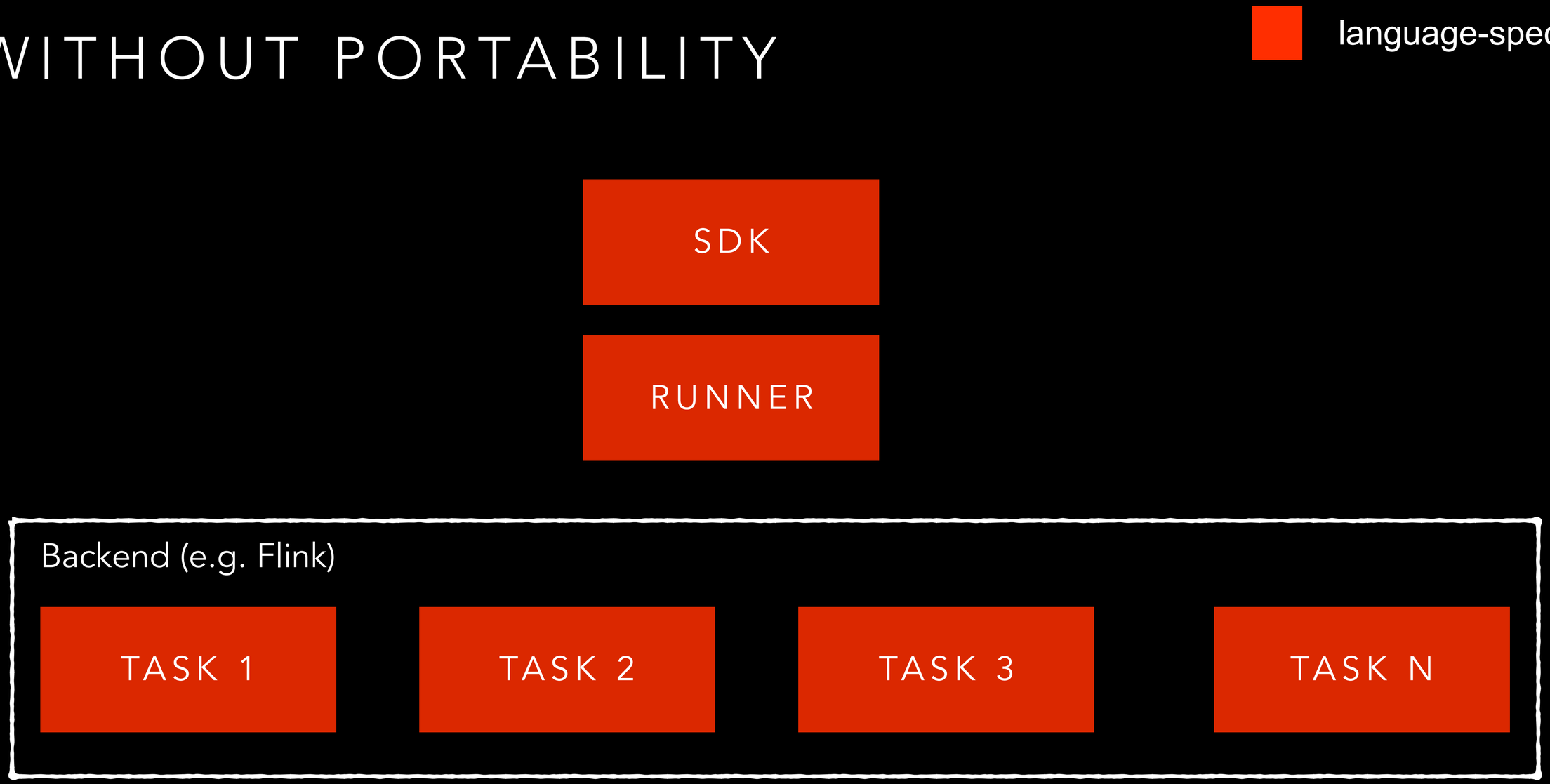
LANGUAGE-PORTABILITY



LANGUAGE-PORTABILITY

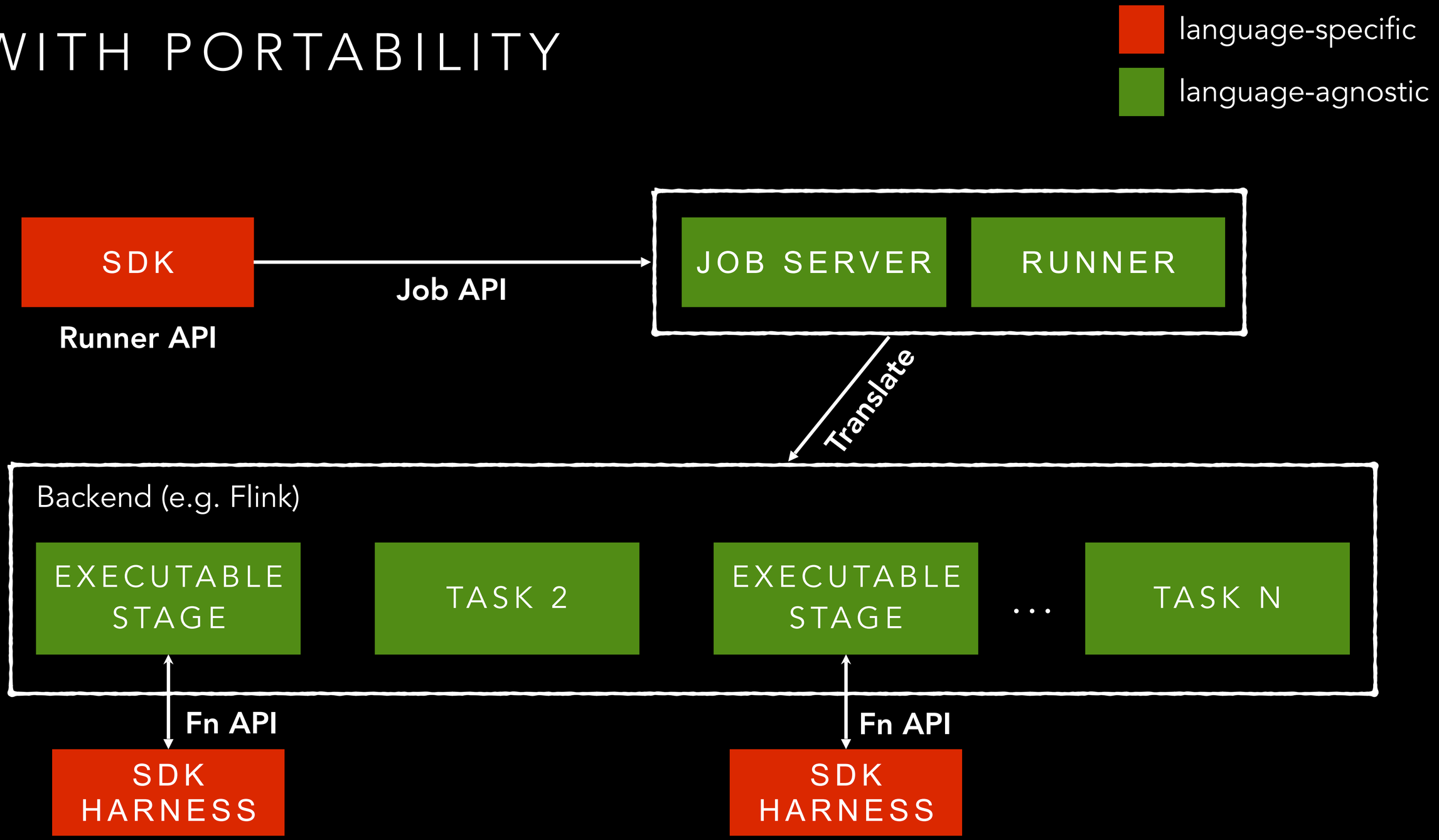


WITHOUT PORTABILITY



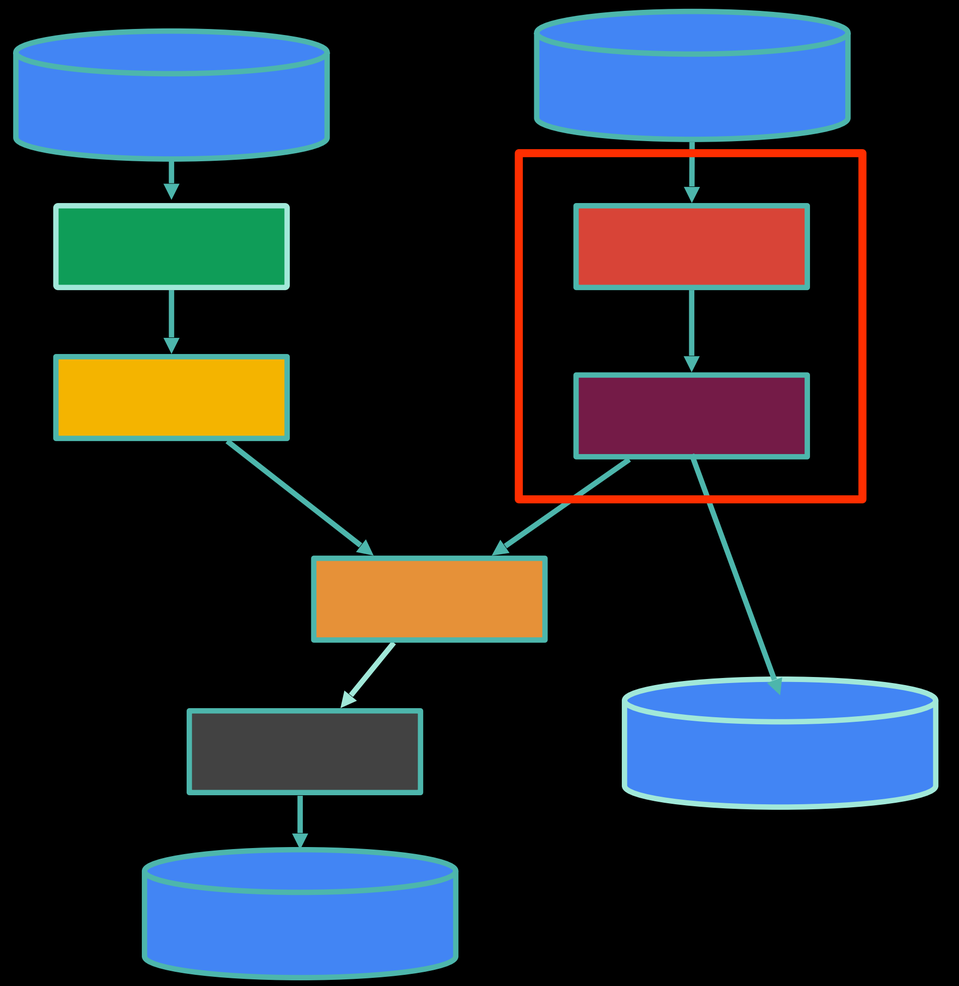
All components are tight to a single language

WITH PORTABILITY



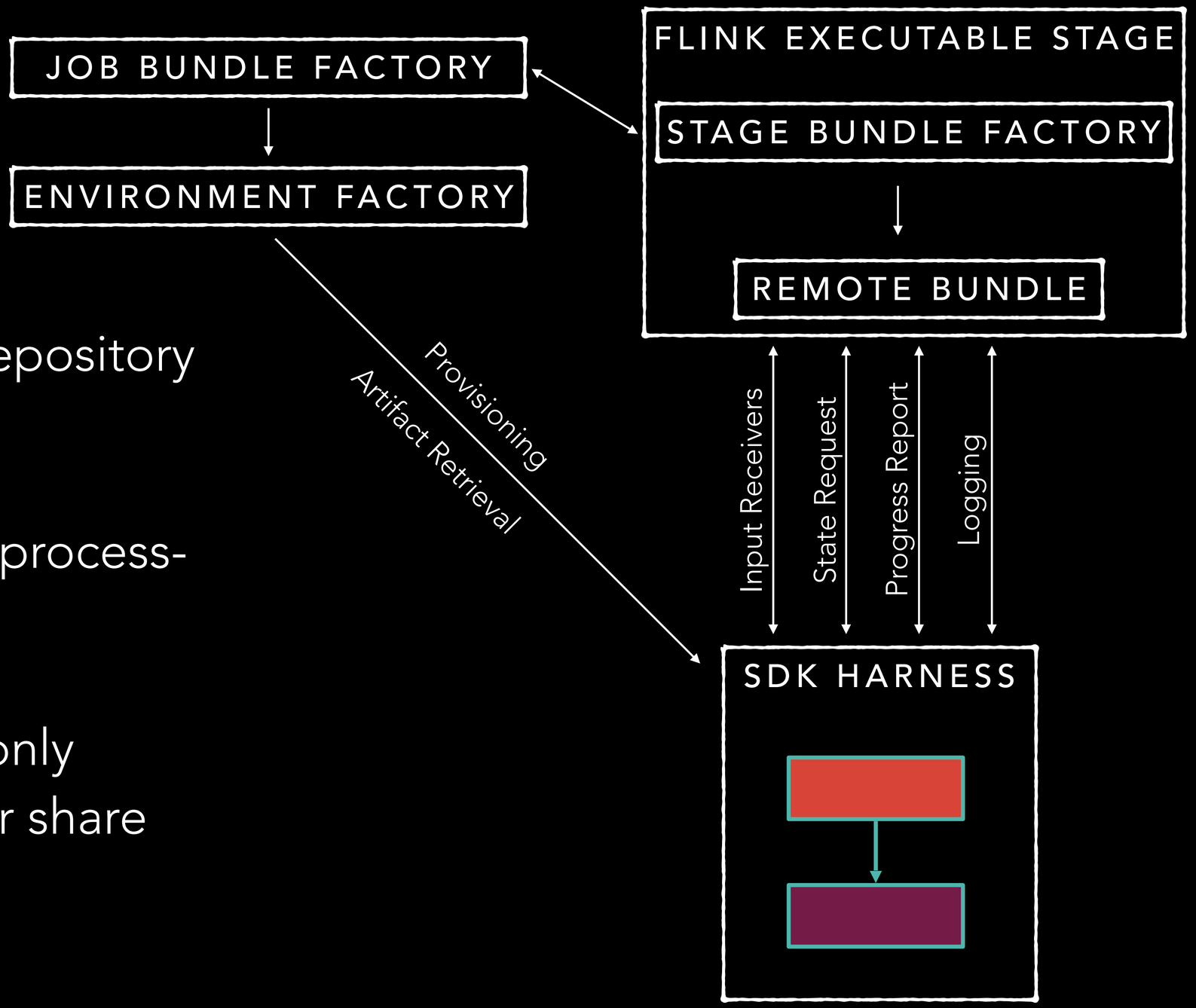
PIPELINE FUSION

- 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)
  - directly in the process (only works if SDK and Runner share the same language)



CROSS-LANGUAGE PIPELINES

- 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!

Files-Based
Apache HDFS
Amazon S3
Google Cloud Storage
local filesystems
AvroIO
TextIO
TFRecordIO
XmlIO
TikaIO
ParquetIO

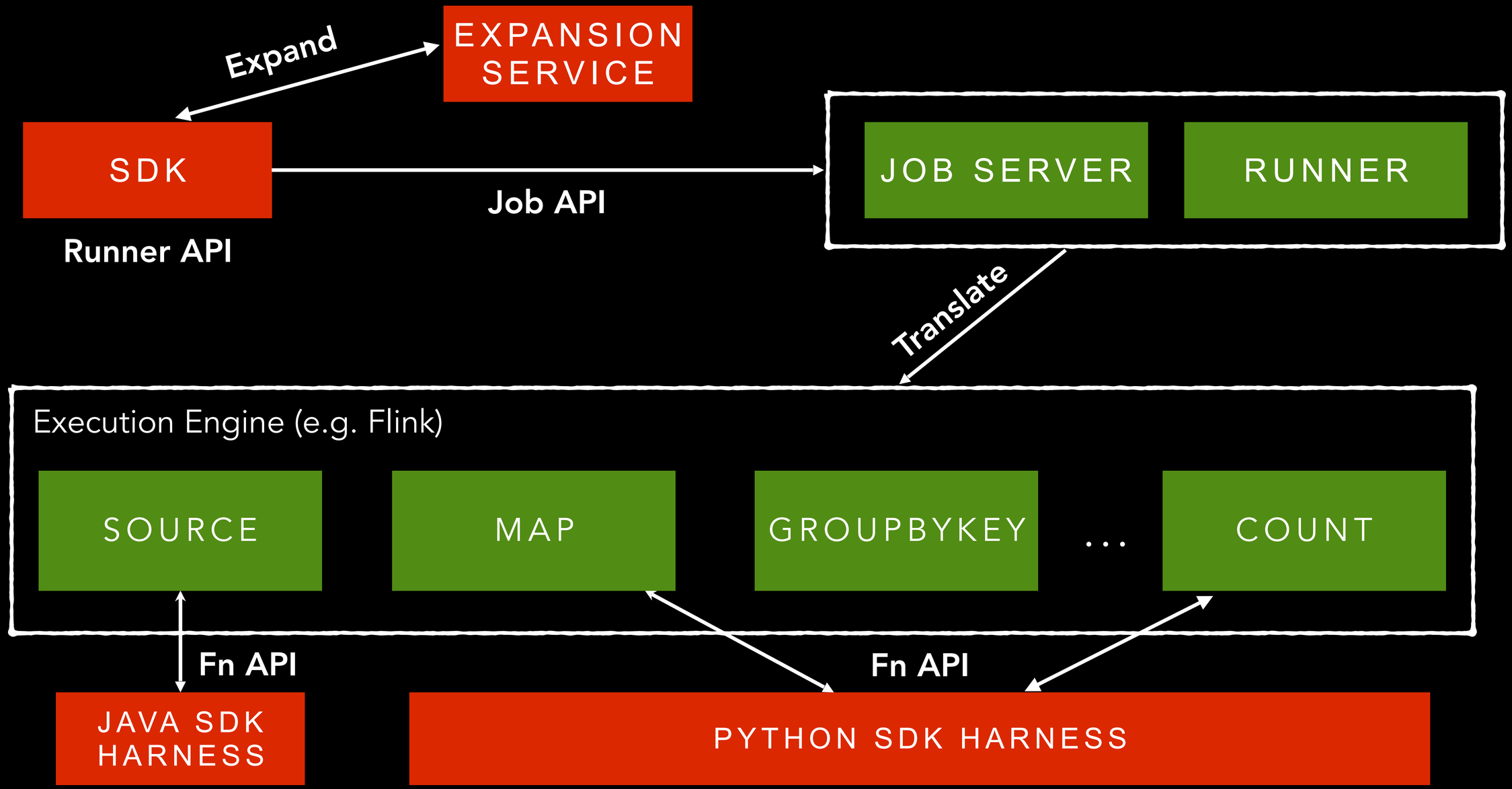
Messaging
Amazon Kinesis
AMQP
Apache Kafka
Google Cloud Pub/Sub
JMS
MQTT

Databases
Apache Cassandra
Apache Hadoop InputFormat
Apache HBase
Apache Hive (HCatalog)
Apache Kudu
Apache Solr
Elasticsearch (v2.x, v5.x, v6.x)
Google BigQuery
Google Cloud Bigtable
Google Cloud Datastore
Google Cloud Spanner
JDBC
MongoDB
Redis

CROSS-LANGUAGE PIPELINES

```
p = Pipeline()
(p
  | IoExpansion(io='KafkaIO',
                  configuration={
                    'topic'  : 'fosdem',
                    'offset' : 'latest'
                  })
  | …
)
```

CROSS-LANGUAGE VIA MIXED ENVIRONMENTS



- ✓ What is Beam?
- ✓ What does portability mean?
- ✓ How do we achieve portability?
- > Are we there yet?

PORTABILITY

Engine Portability

samza

Language Portability

pretty darn close

ROADMAP

- P1 [MVP]: Implement the **fundamental plumbing** for portable SDKs and runners for batch and streaming, including containers and the **ULR** [BEAM-2899]. Each SDK and runner should use the portability framework at least to the extent that **wordcount** [BEAM-2896] and **windowed wordcount** [BEAM-2941] run portably.

- P2 [Feature complete]: Design and implement portability support for remaining execution-side features, so that any pipeline from any SDK can run portably on any runner. These features include **side inputs** [BEAM-2863], **User state** [BEAM-2862], **User timers** [BEAM-2925], **Splittable DoFn** [BEAM-2896] and more. Each SDK and runner should use the portability framework at least to the extent that the **mobile gaming examples** [BEAM-2940] run portably.

- P3 [Performance]: Measure and **tune performance** of portable pipelines using benchmarks such as Nexmark. Features such as **progress reporting** [BEAM-2940], **combiner lifting** [BEAM-2937] and **fusion** are expected to be needed.

- P4 [Cross language]: Design and implement **cross-language pipeline support**, including how the ecosystem of shared transforms should work.

PORTABILITY COMPATIBILITY MATRIX

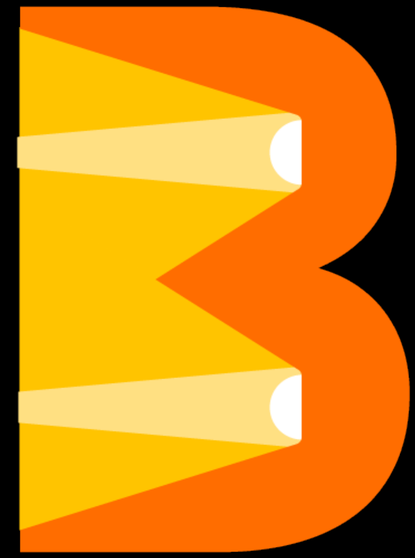
			Flink (master)	instructions					Dataflow					
			Java		Python		Go		Java		Python		Go	
FEATURE			Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming	Batch	Streaming
	Impulse													
	ParDo													
		w/ side input					BEAM-3286	BEAM-3286					BEAM-3286	BEAM-3286
		w/ 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

- ✓ What is Beam?
- ✓ What does portability mean?
- ✓ How do we achieve portability?
- ✓ Are we there yet?

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 or @stadtlegende



Maximilian Michels
mxm@apache.org
@stadtlegende
maximilianmichels.com

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

GETTING STARTED WITH 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
    python setup.py install # if you use the master version
    ```

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

    d. Start JobServer
    ```
    ./gradlew :beam-runners-flink_2.11-job-server:runShadow
    -PflinkMasterUrl=localhost:8081
    ```

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

GETTING STARTED WITH 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
--input=gs://path/to/data* --output=gs://path/to/output
```