Making Big Data Processing Portable.
The Story of Apache Beam and gRPC

Ismaël Mejía. **@iemejia**

KubeCon | CloudNativeCon

Europe 2018

Who am I?

@iemejia

Software Engineer

Apache Beam PMC / Committer

ASF member

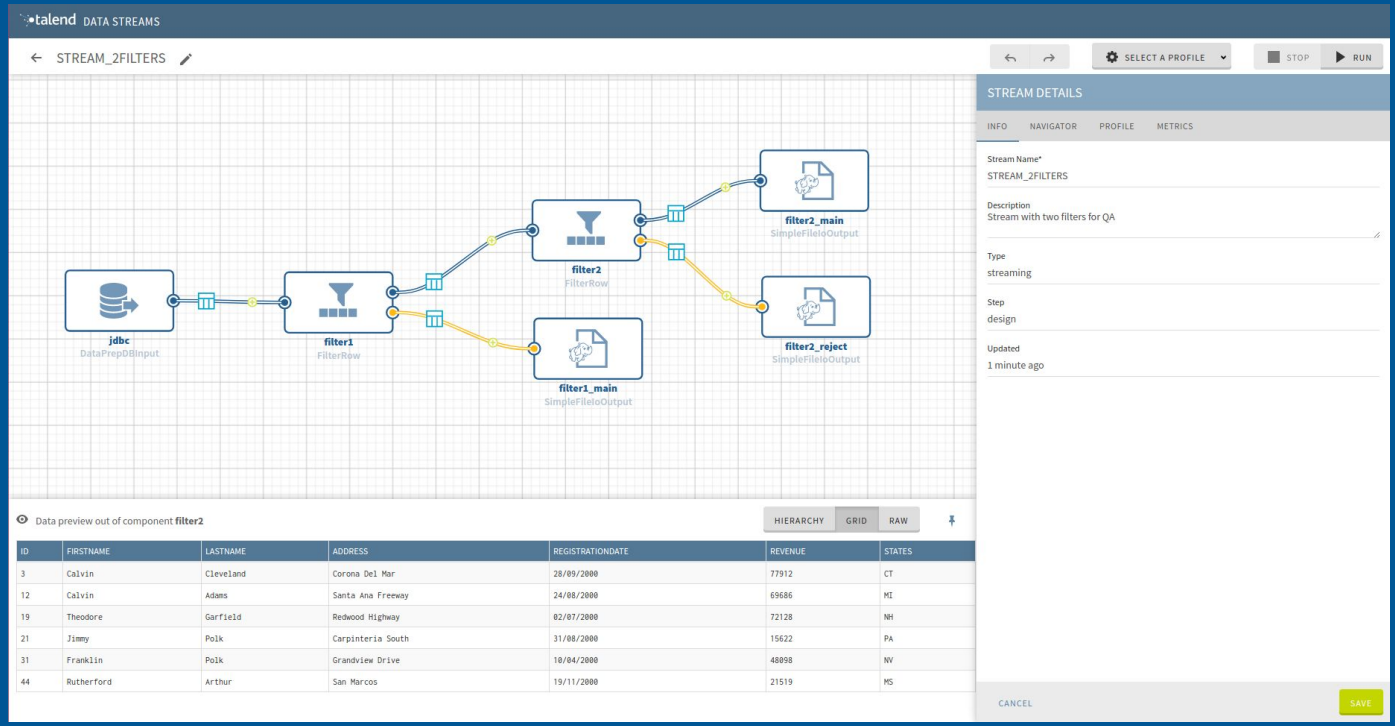
talend

Integration Software

Big Data / Real-Time

Open Source / Enterprise

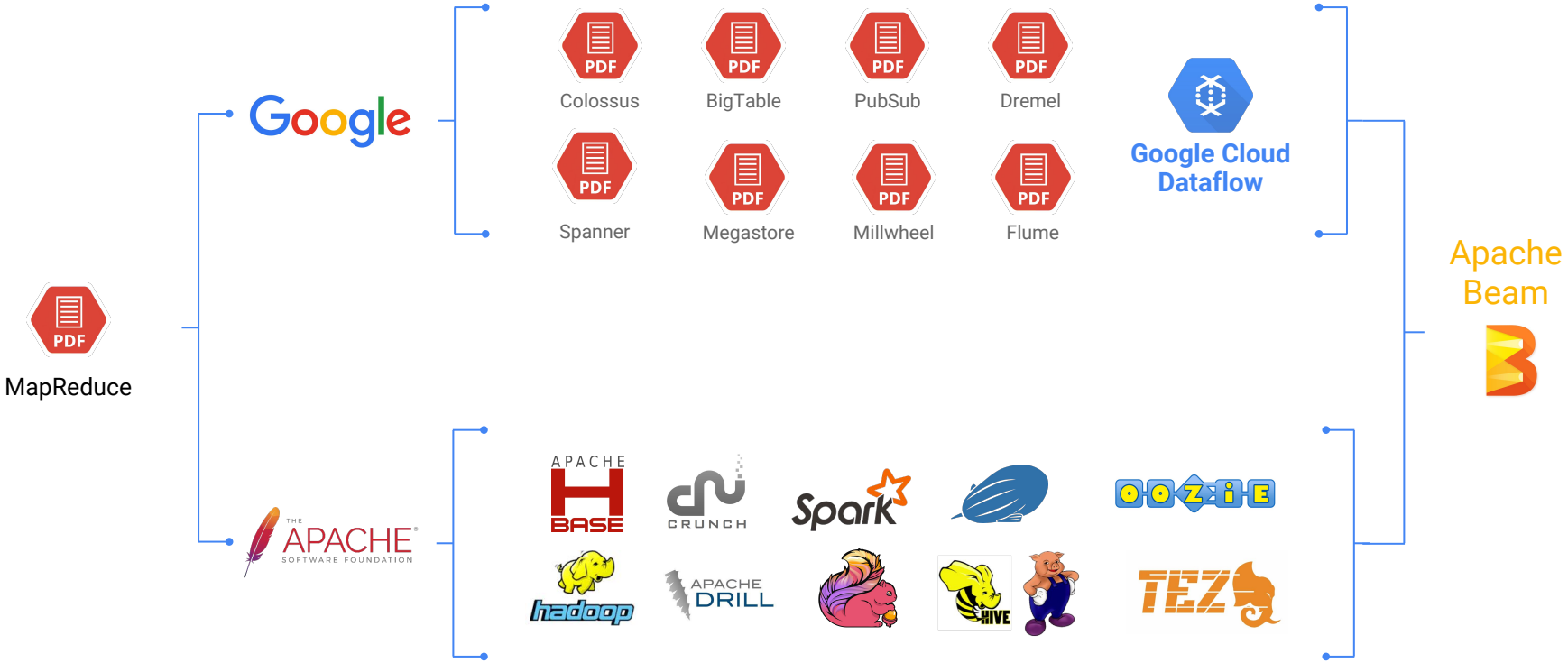
New products



We are hiring !

Introduction: Apache Beam

Apache Beam origin



What is Apache Beam?

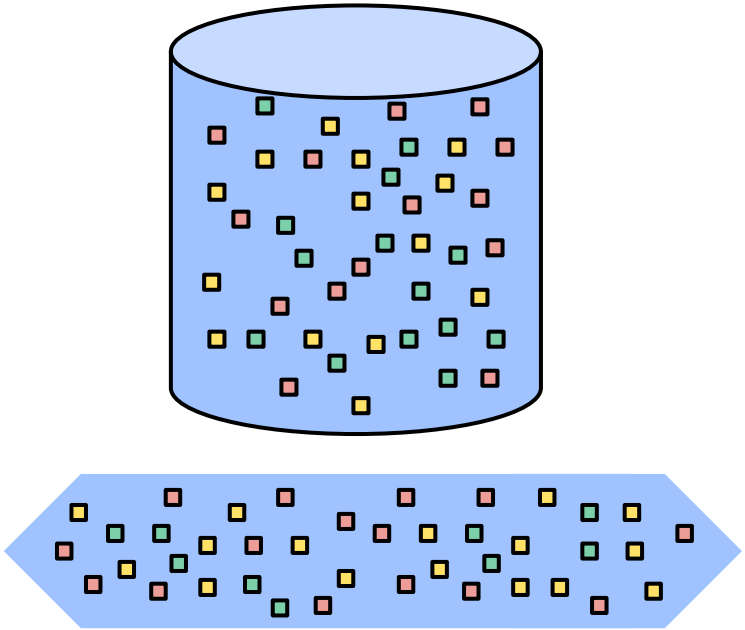
Apache Beam is a **unified** programming model designed to provide **efficient** and **portable** data processing pipelines

Beam Model: Generations Beyond MapReduce

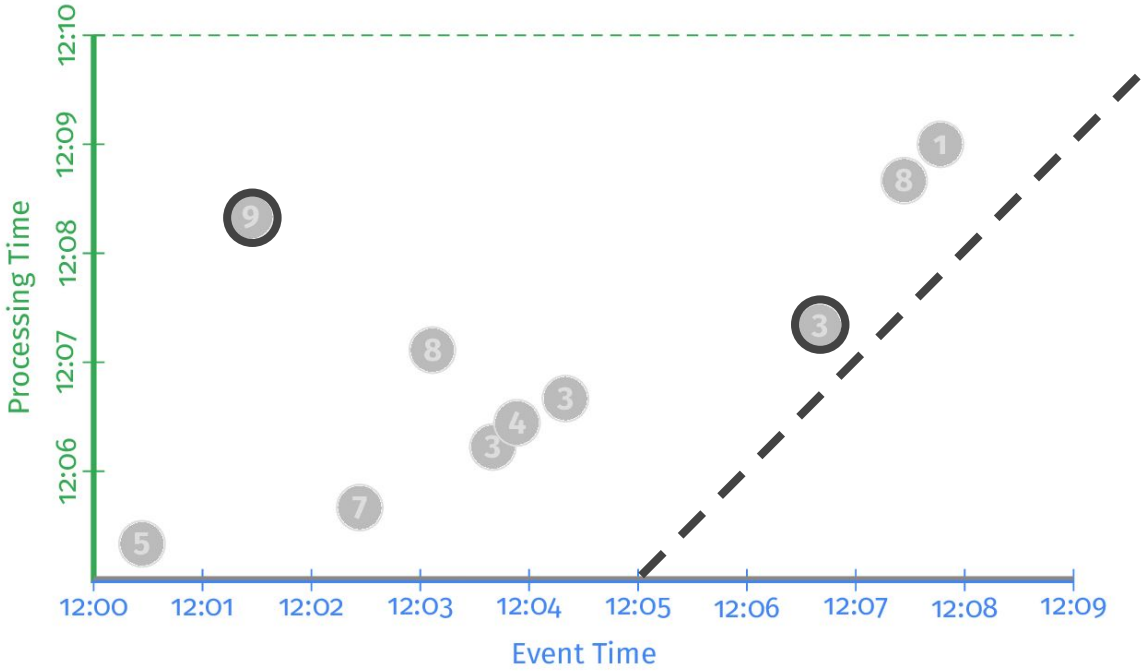
Improved abstractions let you focus on your application logic

Batch and stream processing are *both* first-class citizens -- no need to choose.

Clearly separates event time from processing time.



Processing Time vs. Event Time



Beam Model: Asking the Right Questions

What results are calculated?

Where in event time are results calculated?

When in processing time are results materialized?

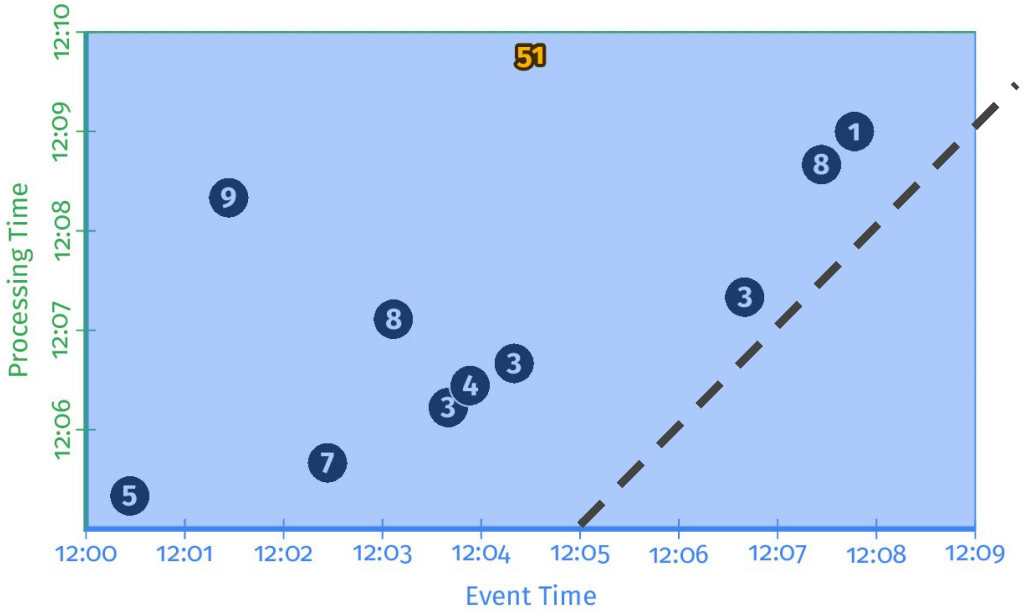
How do refinements of results relate?

The Beam Model: What is Being Computed?

```
PCollection<KV<String, Integer>> scores = input
    .apply(Sum.integersPerKey());
```

```
scores = (input
    | Sum.integersPerKey())
```

The Beam Model: What is Being Computed?



Event Time: Timestamp when the event happened
Processing Time: Absolute program time (wall clock)

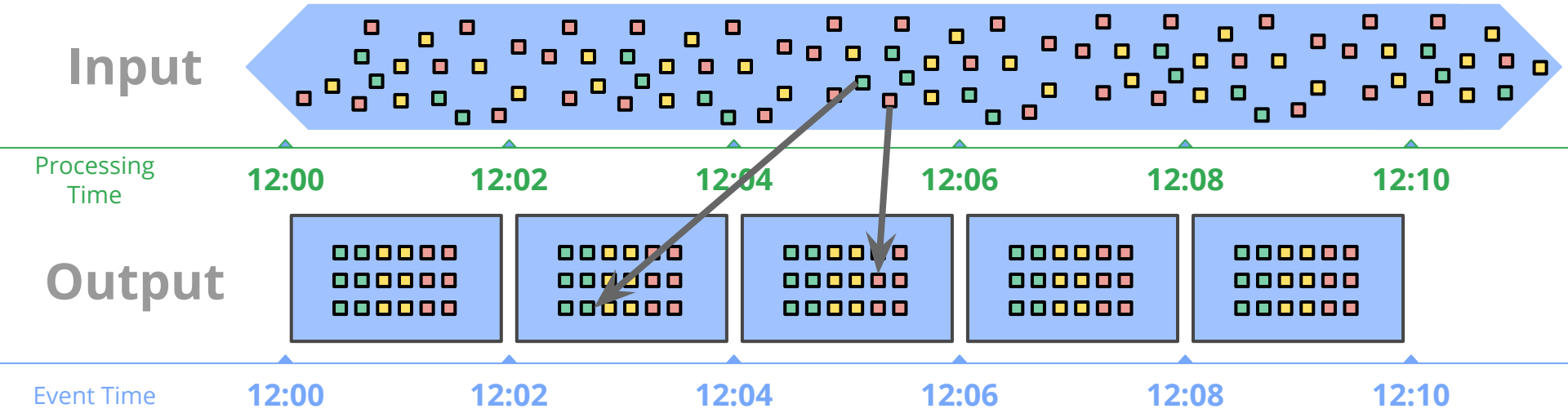
The Beam Model: Where in Event Time?

```
PCollection<KV<String, Integer>> scores = input
    .apply(Window.into(FixedWindows.of(Duration.standardMinutes(2))))
    .apply(Sum.integersPerKey());
```

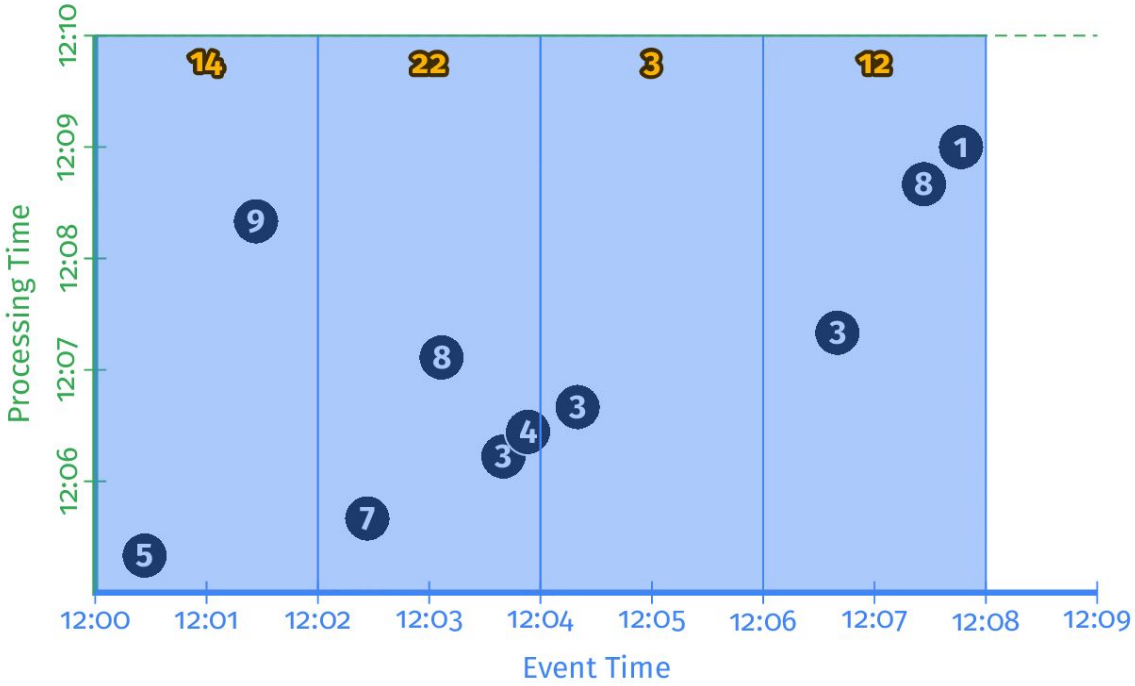
```
scores = (input
    | beam.WindowInto(FixedWindows(2 * 60))
    | Sum.integersPerKey())
```

The Beam Model: Where in Event Time?

- Split infinite data into finite chunks



The Beam Model: Where in Event Time?

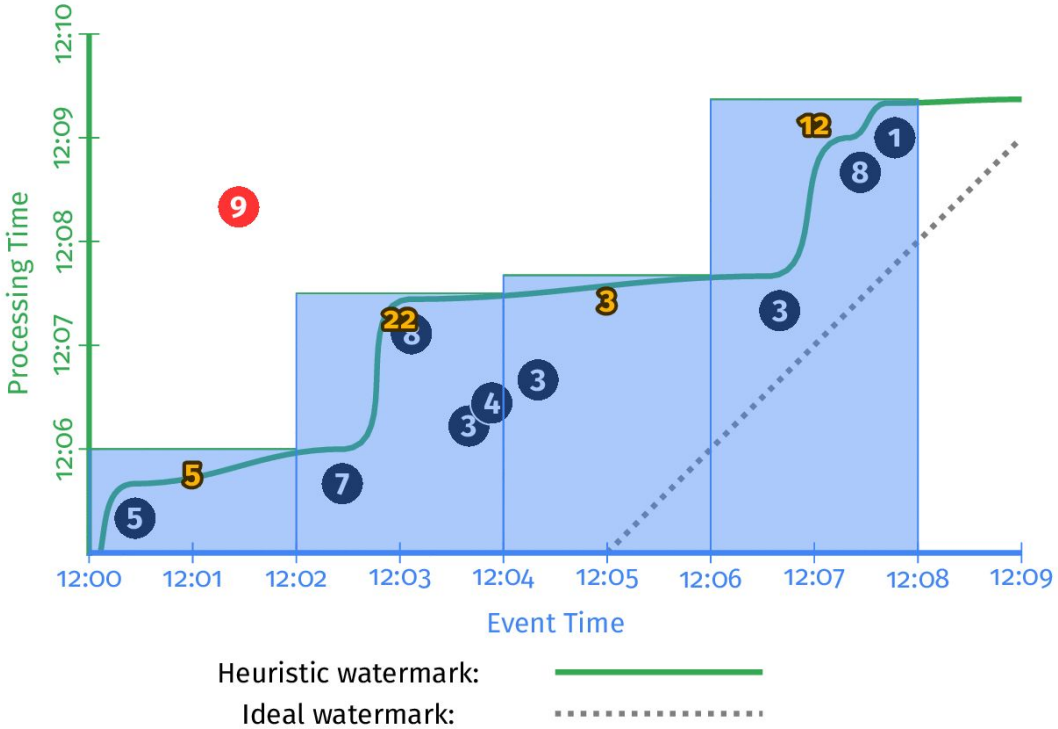


The Beam Model: **When** in Processing Time?

```
PCollection<KV<String, Integer>> scores = input
    .apply(Window.into(FixedWindows.of(Duration.standardMinutes(2))
           .triggering(AtWatermark()))
    .apply(Sum.integersPerKey());
```

```
scores = (input
    | beam.WindowInto(FixedWindows(2 * 60)
        .triggering(AtWatermark())
    | Sum.integersPerKey())
```

The Beam Model: When in Processing Time?

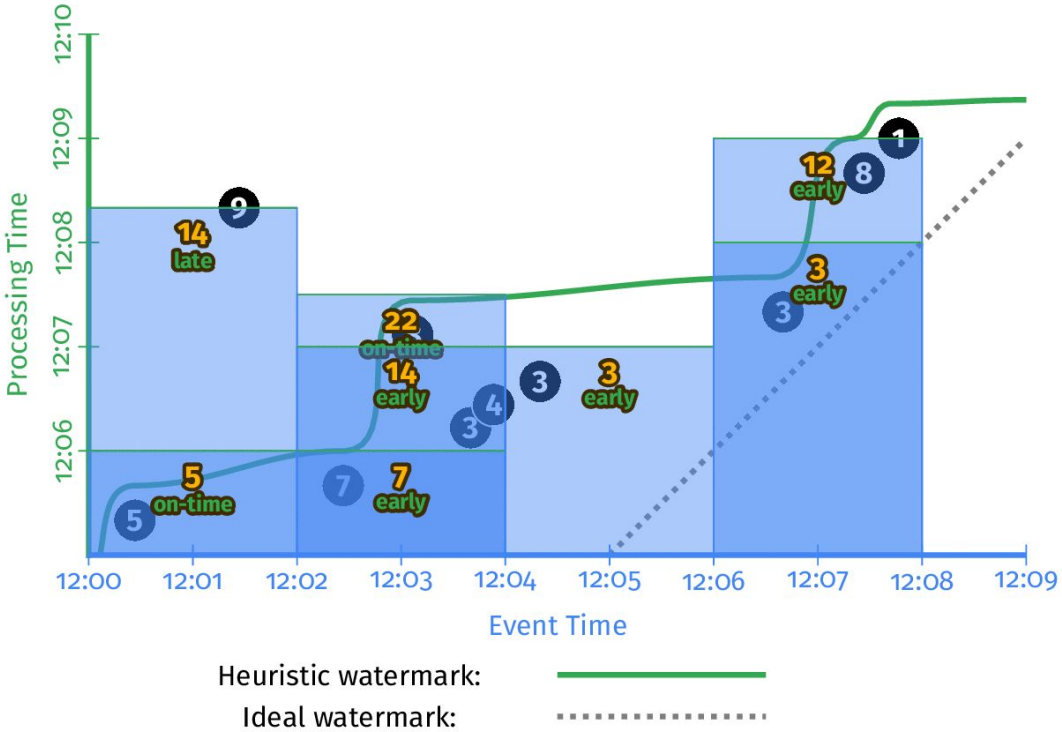


The Beam Model: How Do Refinements Relate?

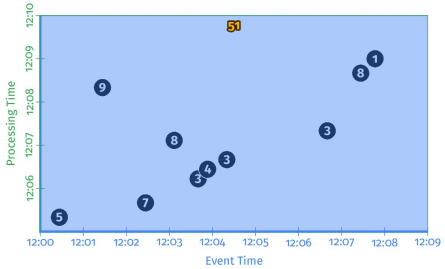
```
PCollection<KV<String, Integer>> scores = input
    .apply(Window.into(FixedWindows.of(Duration.standardMinutes(2))
           .triggering(AtWatermark()
              .withEarlyFirings(AtPeriod(Duration.standardMinutes(1)))
              .withLateFirings(AtCount(1)))
           .accumulatingFiredPanes())
    .apply(Sum.integersPerKey());

scores = (input
    | beam.WindowInto(FixedWindows(2 * 60)
        .triggering(AtWatermark()
            .withEarlyFirings(AtPeriod(1 * 60))
            .withLateFirings(AtCount(1))
        .accumulatingFiredPanes())
    | Sum.integersPerKey())
```

The Beam Model: How Do Refinements Relate?

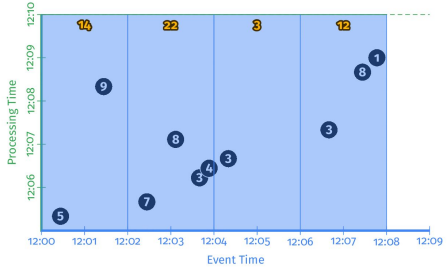


Customizing What Where When How



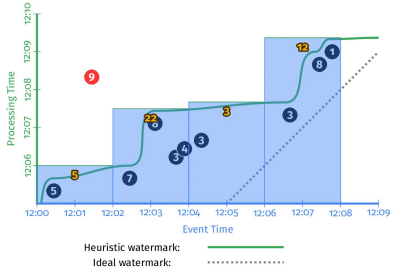
1

Classic Batch



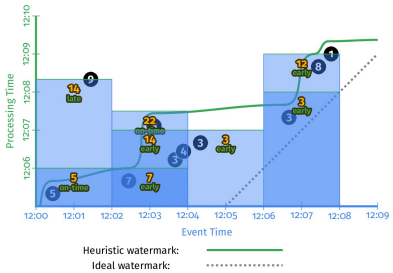
2

Windowed Batch



3

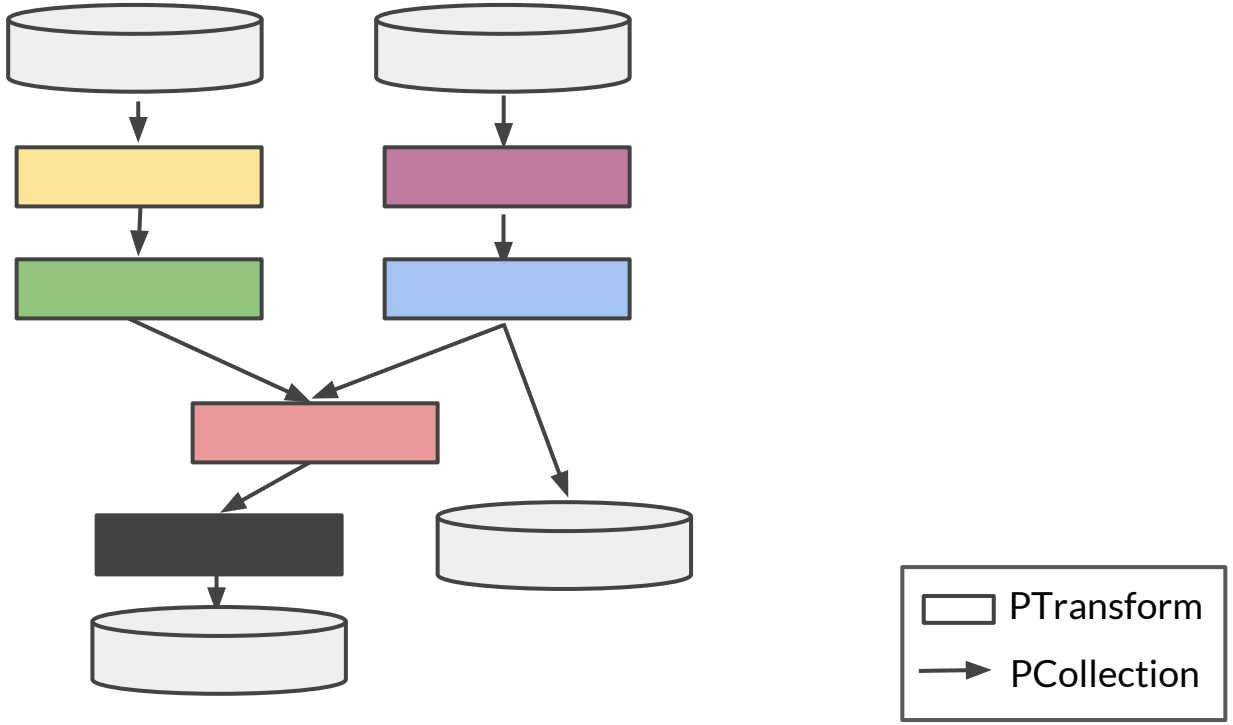
Streaming



4

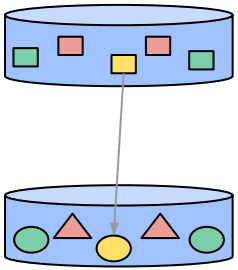
Streaming + Accumulation

Beam Pipeline



Apache Beam - Programming Model

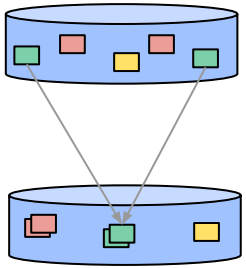
Element-wise



ParDo -> DoFn
MapElements
FlatMapElements
Filter

WithKeys
Keys
Values

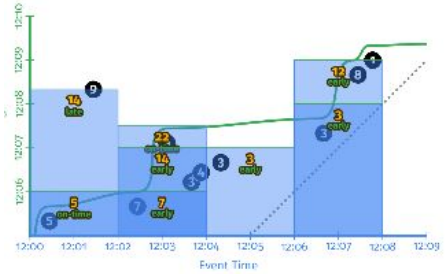
Grouping



GroupByKey
CoGroupByKey

Combine -> Reduce
Sum
Count
Min / Max
Mean
...

Windowing/Triggers



Windows
FixedWindows
GlobalWindows
SlidingWindows
Sessions

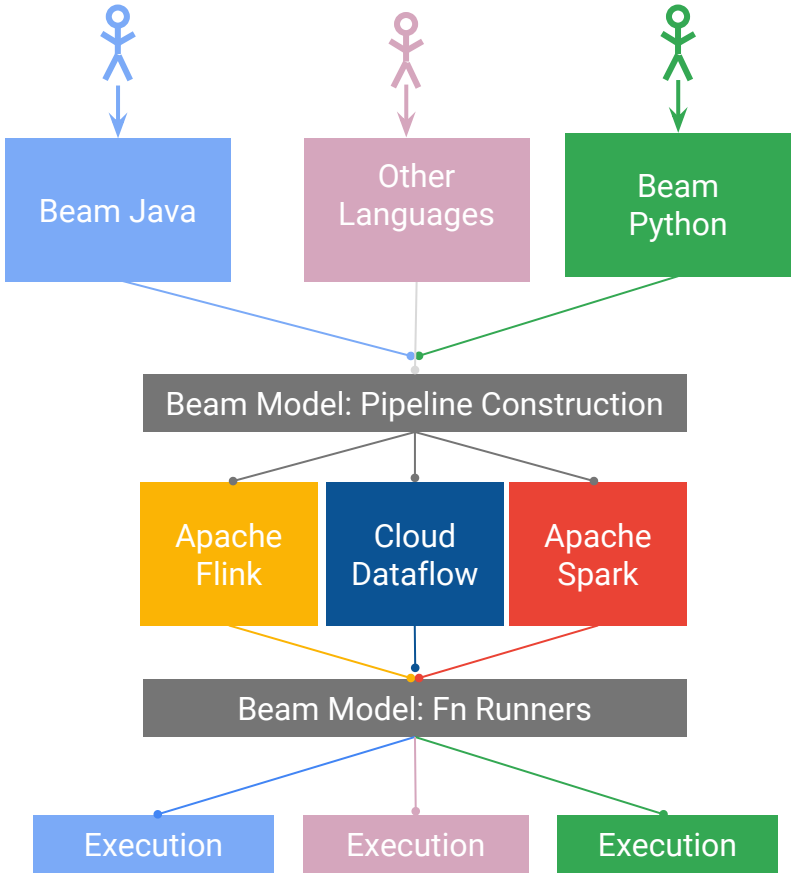
Triggers
AfterWatermark
AfterProcessingTime
Repeatedly

The Apache Beam Vision

1. **End users:** who want to write pipelines in a language that's familiar.

2. **SDK writers:** who want to make Beam concepts available in new languages.

3. **Runner writers:** who have a distributed processing environment and want to support Beam pipelines



Runners

Runners "**translate**" the code into the target runtime

Apache Beam
Direct Runner

Apache Apex

Apache Spark

Apache Flink

Apache Gearpump

Google Cloud
Dataflow

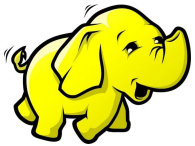
IBM Streams

Apache Storm

WIP

Ali Baba
JStorm

Apache Samza



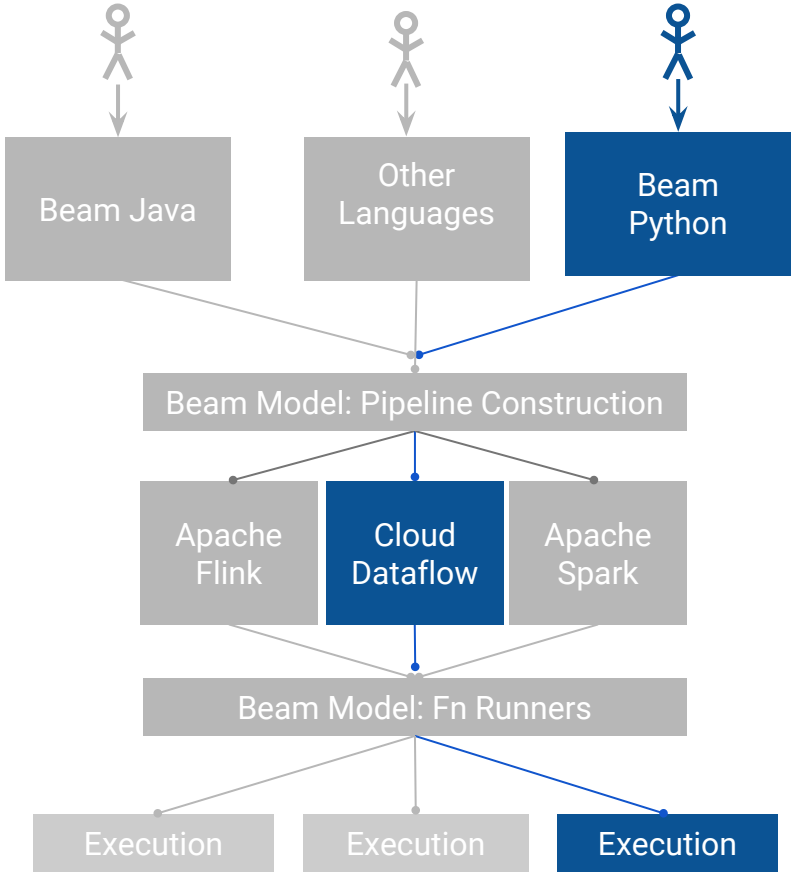
Hadoop
MapReduce

* Same code, different runners & runtimes

Awesome but...

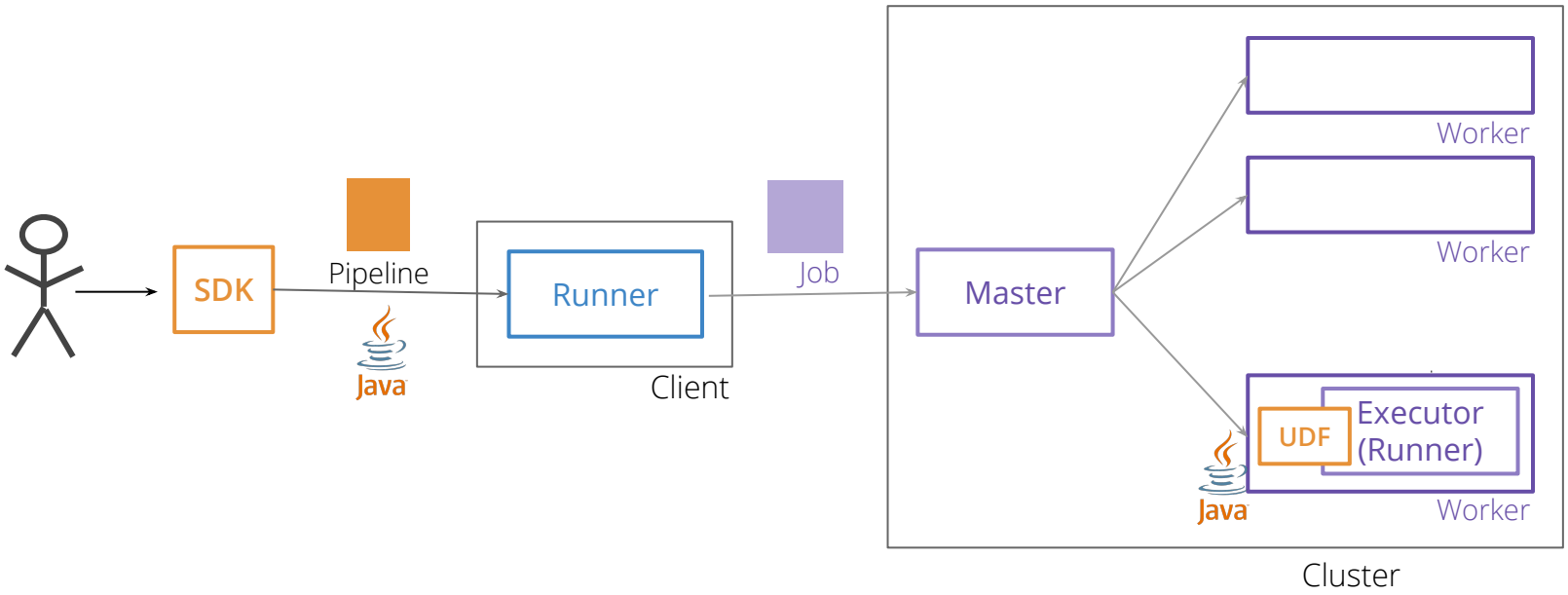
- If I run a Beam python pipeline on the Spark runner, is it translated to PySpark?
- Wait, can I execute python on a Java based runner?
- Can I use the python Tensorflow transform from a Java pipeline?
- I want to connect to Kafka from Python but there is not a connector can I use the Java one?

No



Portability API

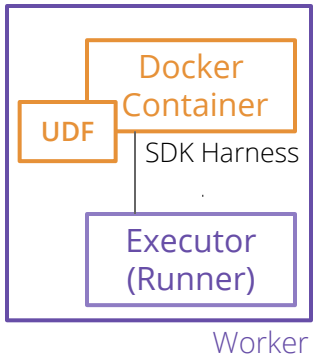
How do Java-based runners do work today?



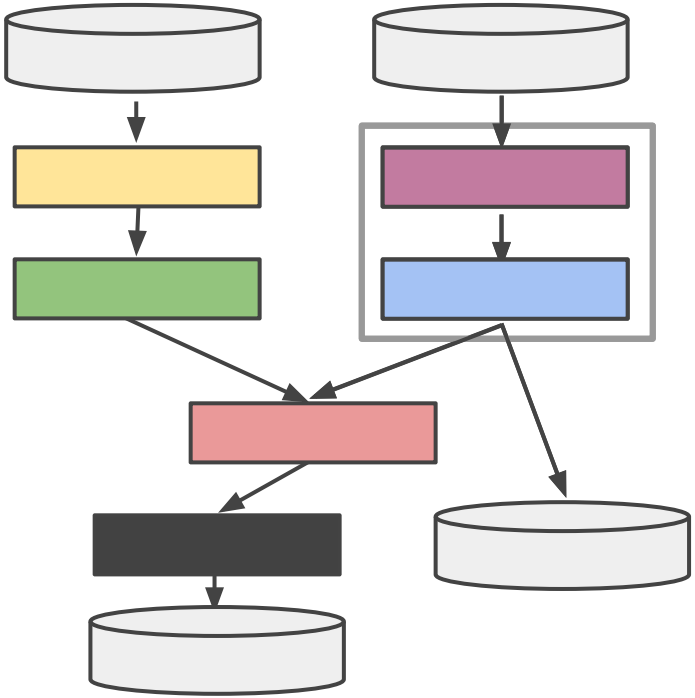
Portability API Design

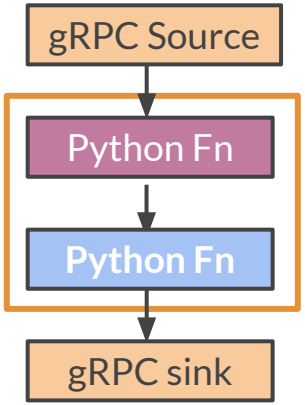
Goal:
Execute user code from 'any' language in every runner.

Challenges:
Support existing SDKs (Java / Python)
Provision of expected execution environment
Performance. Low overhead
Support Multiple language data representation
Easy to evolve



Portability API Design





Portability APIs

Well-defined, language-neutral data structures and protocols between the SDK Harness and runner

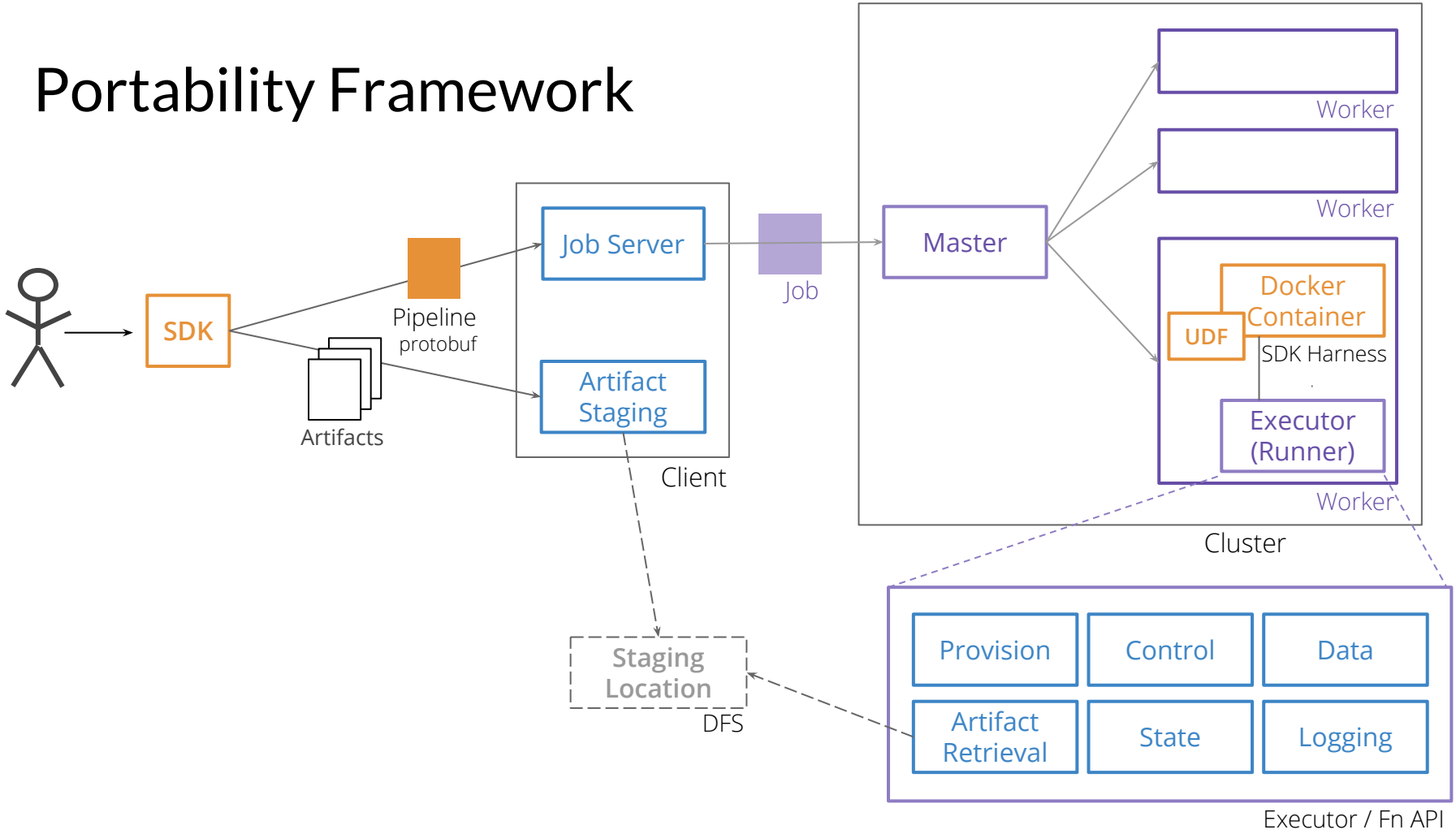
Runner API: Pipeline language agnostic representation

Job API: Job submission and management protocol.

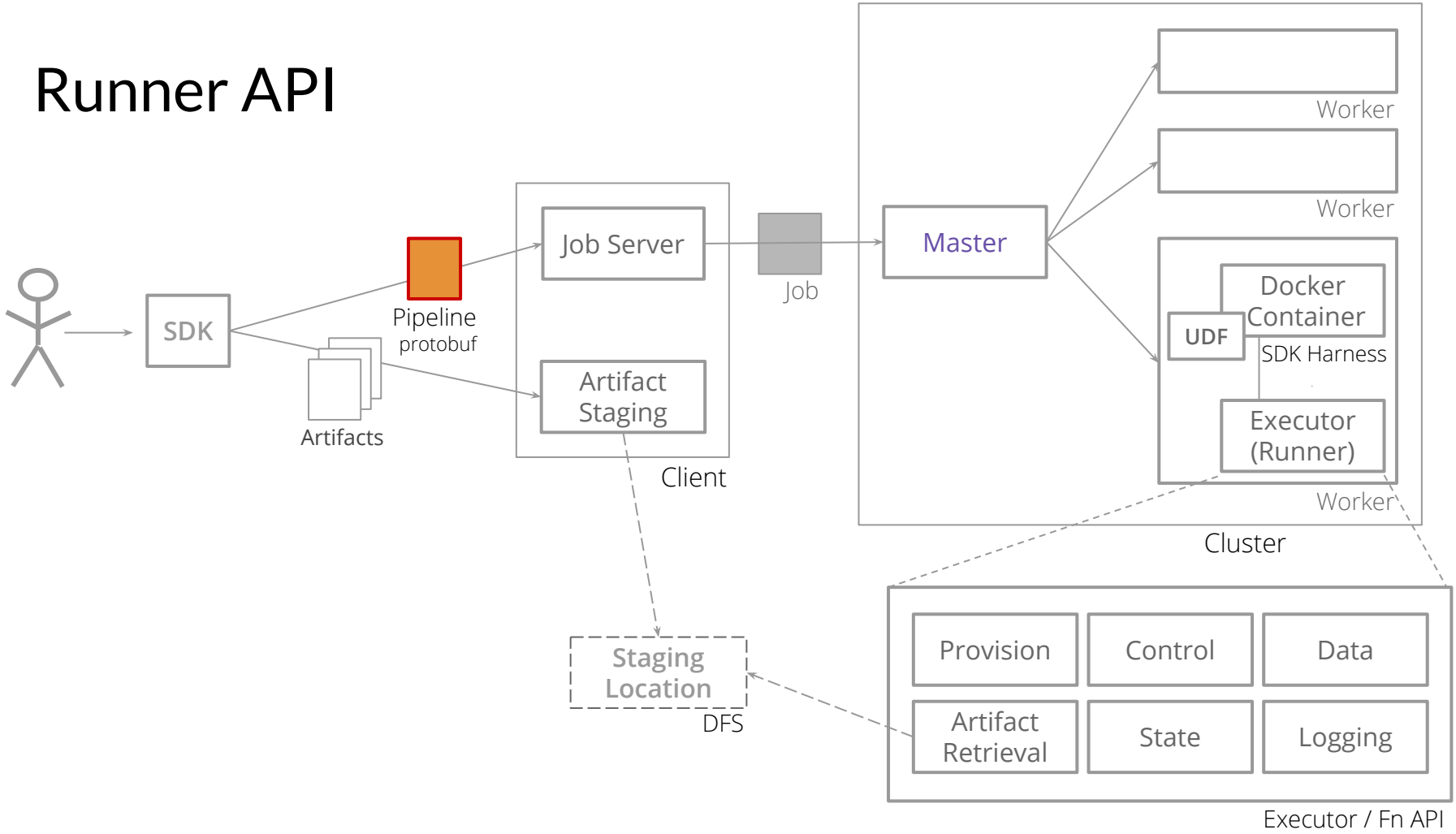
Fn API: Protocols between runner and SDK harness

- Efficient serialization format (Protocol buffers)
- Multiple language support
- Simple service definition
- Network performance
- Multiplexing (via HTTP/2)
- Rich communication models:
  - Subscriptions
  - Bidirectional streaming

Portability Framework



Runner API


SDK
Pipeline
protobuf
Artifacts
Job Server
Artifact Staging
Client
Job
Master
Worker
Worker
Docker Container
UDF
SDK Harness
Executor (Runner)
Worker
Cluster
Staging Location
DFS
Provision
Control
Data
Artifact Retrieval
State
Logging
Executor / Fn API

Runner API

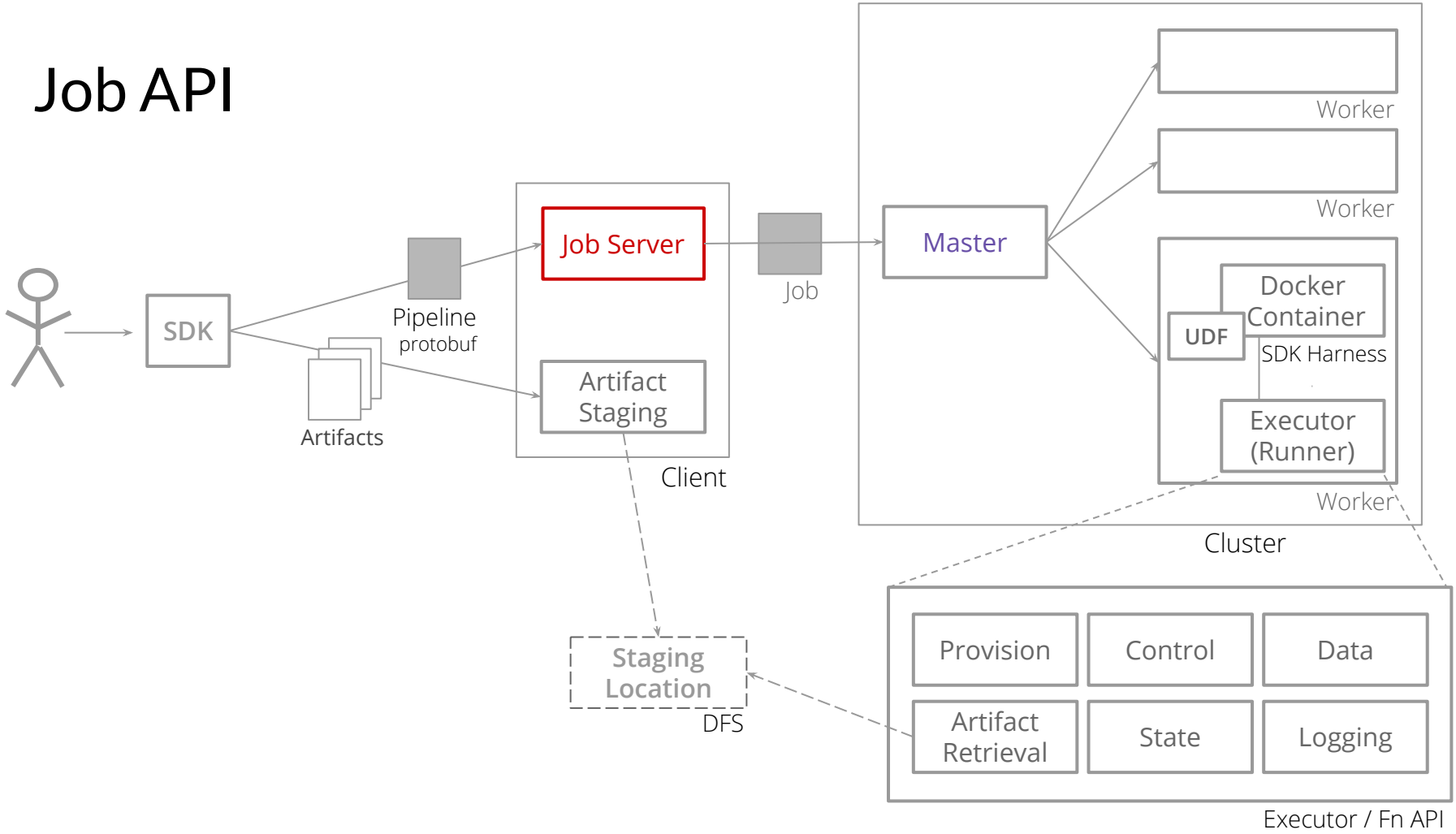
Represent Beam model via Protobuf

```
message Pipeline {
  Components components = 1;
  ...
}
```

```
oneof root {
  Coder coder = 2;
  CombinePayload combine_payload = 3;
  SdkFunctionSpec sdk_function_spec = 4;
  ParDoPayload par_do_payload = 6;
  PTransform ptransform = 7;
  PCollection pcollection = 8;
  ReadPayload read_payload = 9;
  SideInput side_input = 11;
  WindowIntoPayload window_into_payload = 12;
  WindowingStrategy windowing_strategy = 13;
  FunctionSpec function_spec = 14;
}
```

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

Job API


SDK
Pipeline
protobuf
Artifacts
Job Server
Artifact Staging
Client
Job
Master
Worker
Worker
Docker Container
UDF
SDK Harness
Executor (Runner)
Worker
Cluster
Staging Location
DFS
Provision
Control
Data
Artifact Retrieval
State
Logging
Executor / Fn API

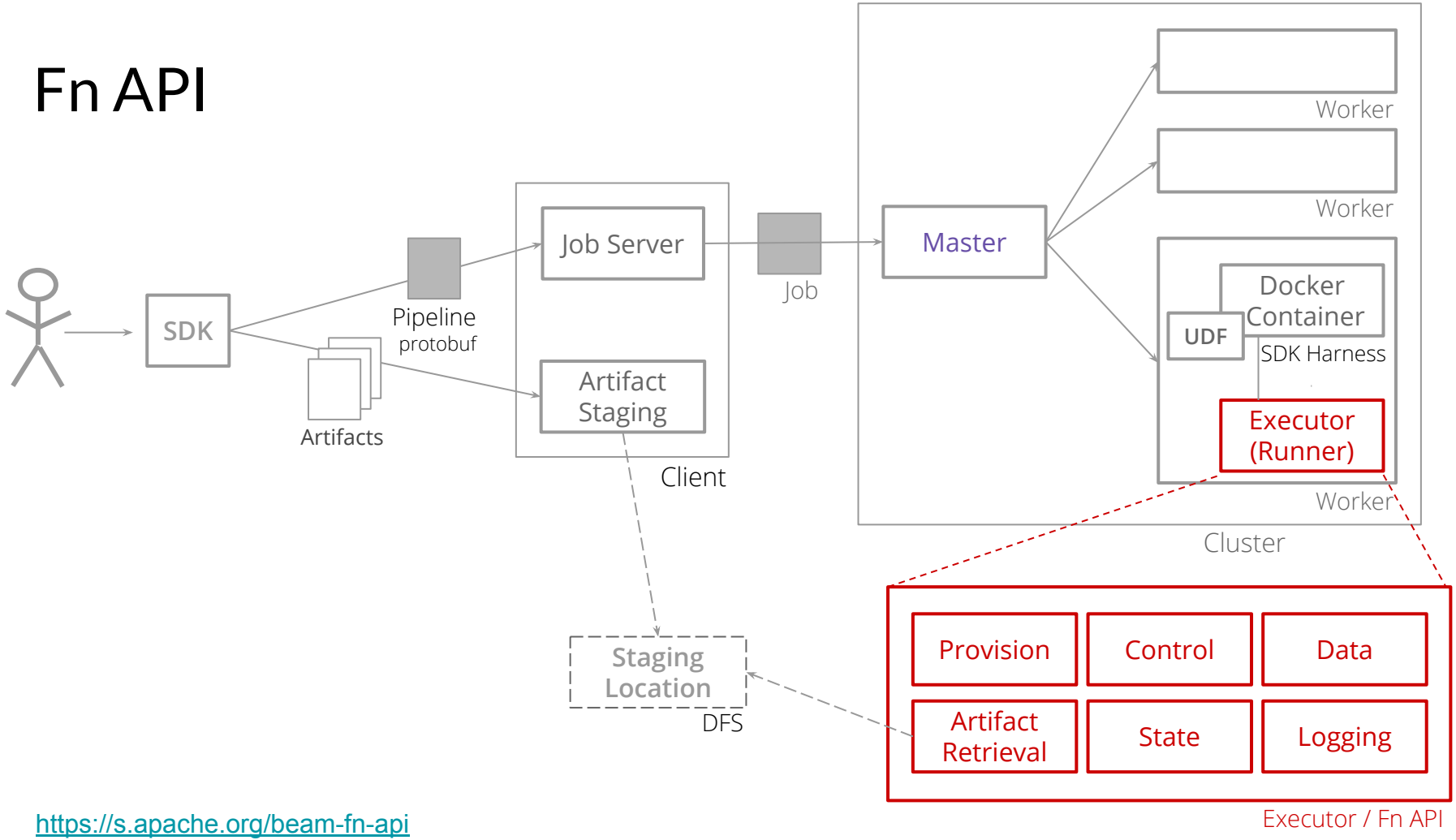
Job API

Job submission and management protocol

```
service JobService {
  rpc Prepare (PrepareJobRequest) returns (PrepareJobResponse);
  rpc Run (RunJobRequest) returns (RunJobResponse);
  rpc GetState (GetJobStateRequest) returns (GetJobStateResponse);
  rpc Cancel (CancelJobRequest) returns (CancelJobResponse);
  // Subscribe to a stream of state changes of the job
  rpc GetStateStream (GetJobStateRequest) returns (stream GetJobStateResponse);
  // Subscribe to a stream of state changes and messages from the job
  rpc GetMessageStream (JobMessagesRequest) returns (stream JobMessagesResponse);
}
```

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

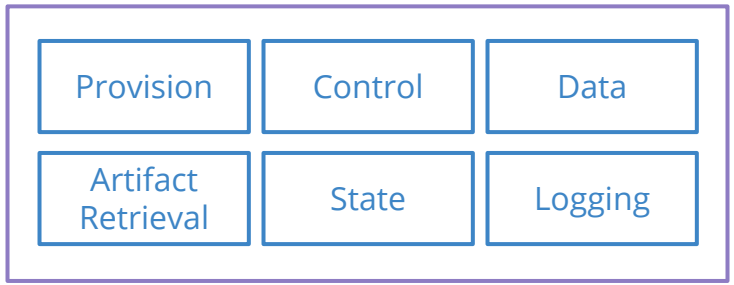
Fn API



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

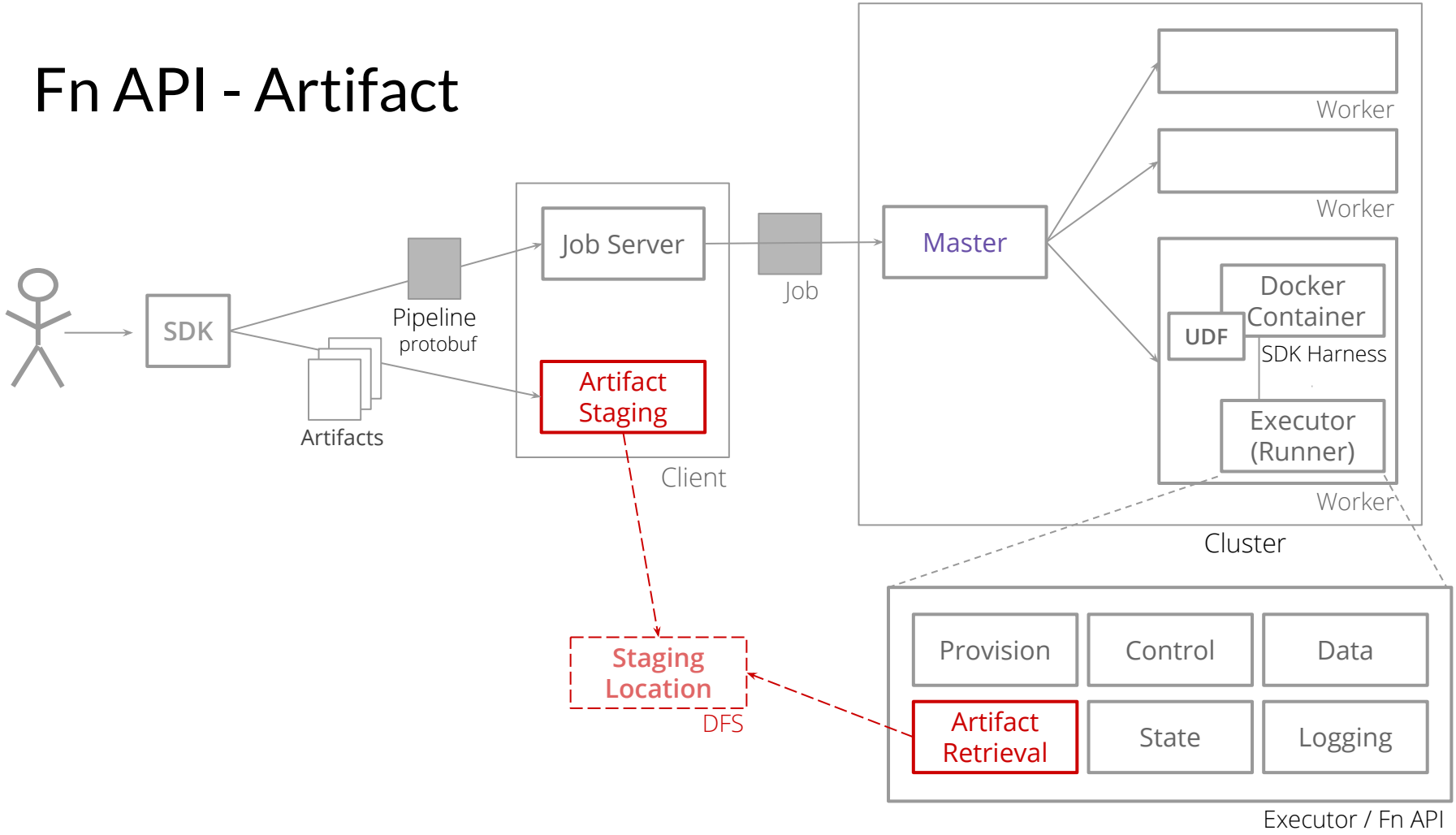
Fn API

Fn API allows a runner to invoke SDK-specific user-defined functions
Interaction between Runner and SDK Harness



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

Fn API - Artifact



Fn API - Artifact

Two services: Stage/Retrieve artifacts for use in a Job

```
service ArtifactStagingService {
 rpc PutArtifact(stream PutArtifactRequest) returns (PutArtifactResponse);
 // Commit the manifest for a Job. All artifacts must have been uploaded
 rpc CommitManifest(CommitManifestRequest) returns (CommitManifestResponse);
}
```

```
service ArtifactRetrievalService {
 rpc GetManifest(GetManifestRequest) returns (GetManifestResponse);
 rpc GetArtifact(GetArtifactRequest) returns (stream ArtifactChunk);
}
```

```
message ArtifactMetadata {
 string name = 1;
 uint32 permissions = 2;
 string md5 = 3;
}
```

Fn API - Provision

Provide runtime provisioning information to the SDK harness

```
service ProvisionService {
   rpc GetProvisionInfo(GetProvisionInfoRequest) returns (GetProvisionInfoResponse);
}
```

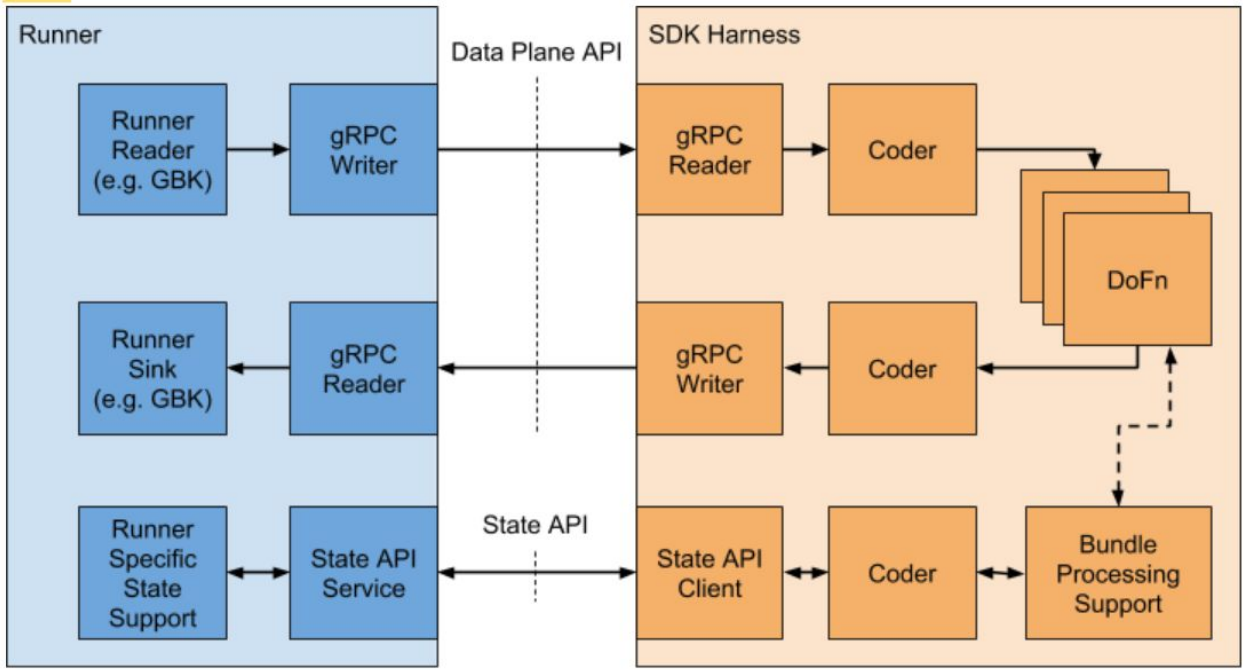
```
message ProvisionInfo {
   string job_id = 1;
   string job_name = 2;
   string worker_id = 5;

   google.protobuf.Struct pipeline_options = 3;
   Resources resource_limits = 4;
}
```

```
message Resources {
   Memory memory = 1;
   Cpu cpu = 2;
   Disk semi_persistent_disk = 3;
}
```

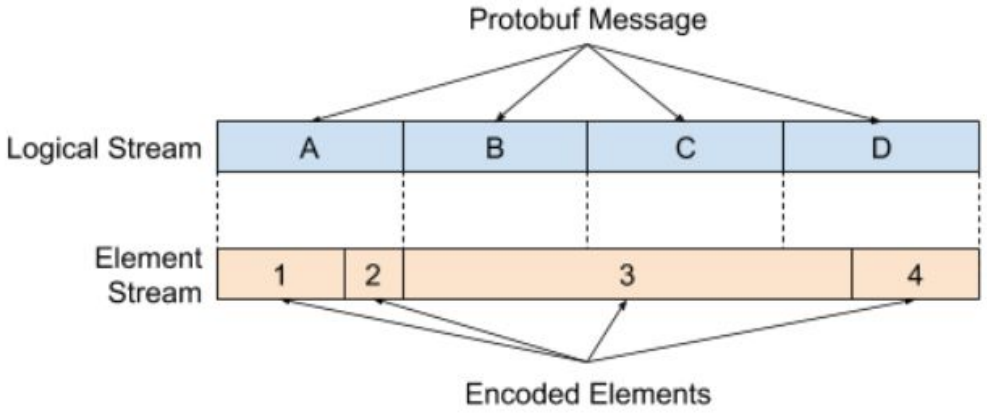
Fn API - Data

Moves data between the runner and the SDK Harness



Fn API - Data

```
service BeamFnData {
  rpc Data(stream Elements) returns (stream Elements);
}
```

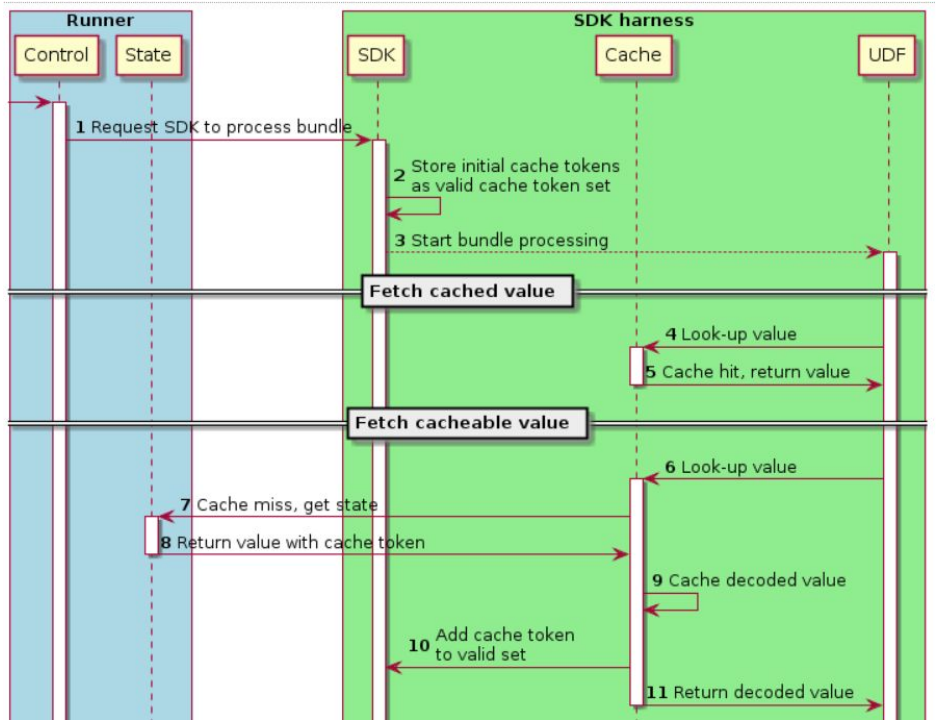



```
message Elements {
  message Data {
    string instruction_reference = 1;
    Target target = 2;
    bytes data = 3;
  }
  repeated Data data = 1;
}
```

Protobuf message limitation 2 or 4 GB depending on language. Beam asumes < 2GB

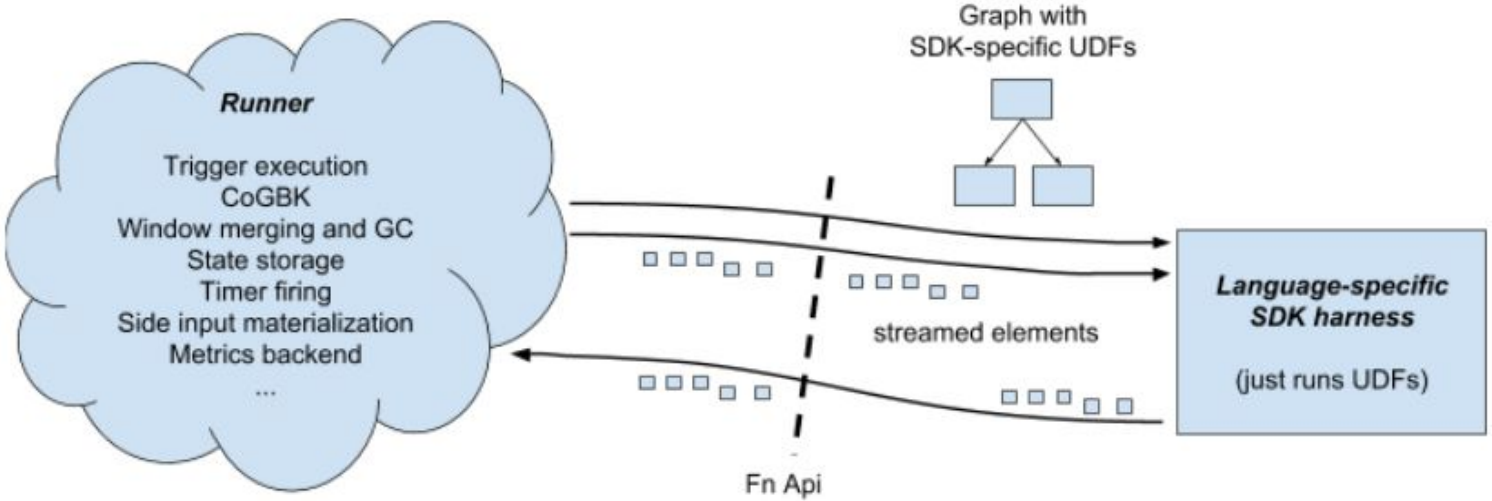
Fn API - State

Supports user state, side inputs, and Group by Key re-iteration



Fn API - Control

Tell SDK Harness what UDFs to execute and when to do it.



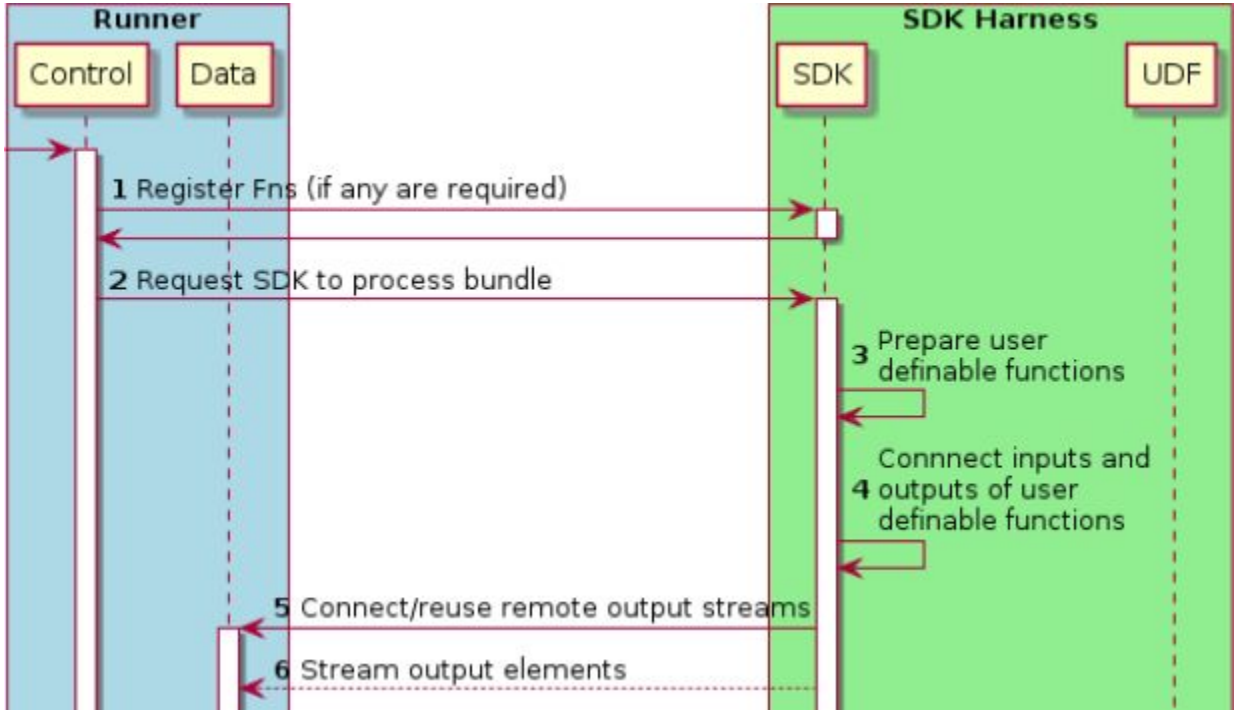
Fn API - Control

Describes the work that a SDK harness is meant to do

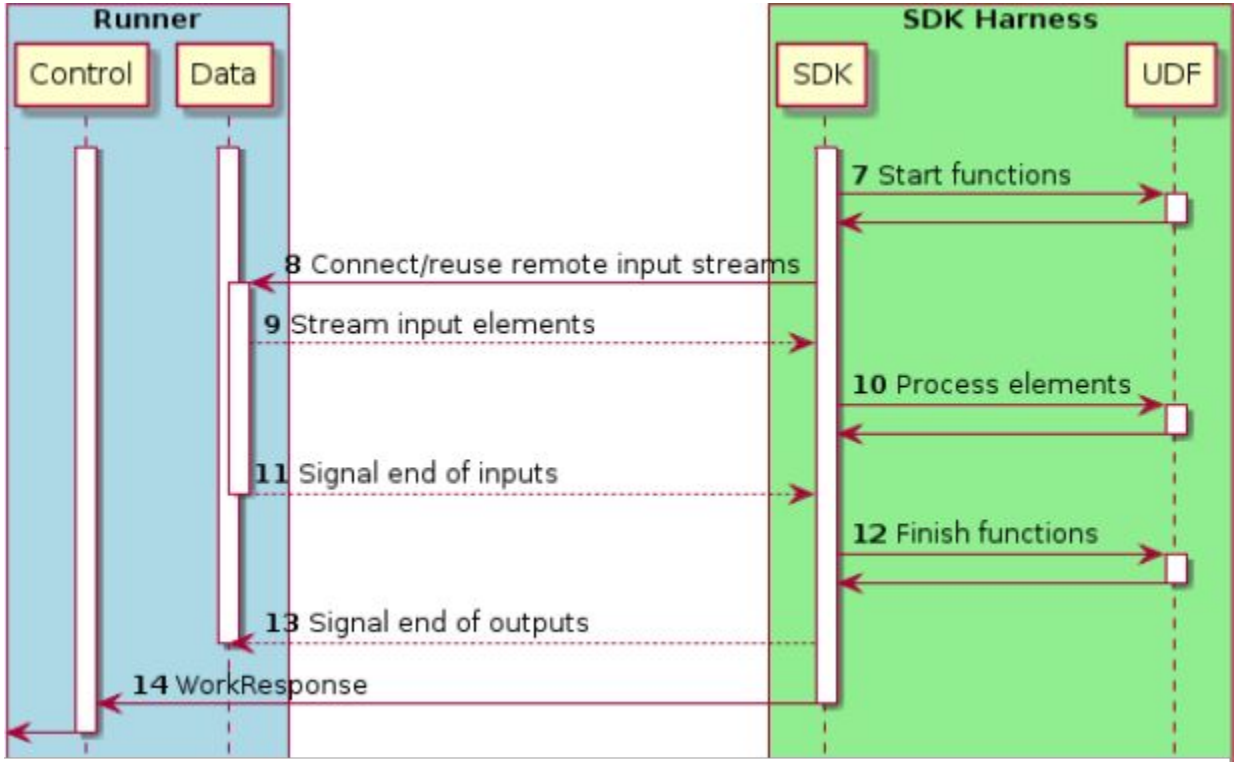
```
service BeamFnControl {
  rpc Control(stream InstructionResponse) returns (stream InstructionRequest);
}
```

```
message InstructionRequest {
  string instruction_id = 1;
  oneof request {
    RegisterRequest register = 1000;
    ProcessBundleRequest process_bundle = 1001;
    ProcessBundleProgressRequest process_bundle_progress = 1002;
    ProcessBundleSplitRequest process_bundle_split = 1003;
  }
}
```

Fn API - Control - Bundle Processing



Fn API - Control - Bundle Processing



Fn API - Logging

Associate SDK Harness log entries with the runner

```
service BeamFnLogging {
  rpc Logging(stream LogEntry.List)
    returns (stream LogControl);
}
```

```
message LogEntry {
  message List {
    repeated LogEntry log_entries = 1;
  }
  message Severity {
    enum Enum {
      UNSPECIFIED = 0;
      TRACE = 1;
      DEBUG = 2;
      //...
    }
  }
  Severity.Enum severity = 1;
  google.protobuf.Timestamp timestamp = 2;
  string message = 3;
  // ...
}
```

Advantages

Isolation of user code
Isolated configuration of user environment
Multiple language execution
Mix user code in different languages

Issues

Performance overhead (15% in early evaluation). via extra RPC + container
Extra component (docker)
A bit more complex but it is the price of reuse

Current status and future work

Current status

- Universal Local Runner (Local runner)
- Rewrite of the Flink runner to support the Portability API
- Python wordcount runs on Apache Flink runner
- and a newcomer also runs on the Apache Flink runner ...

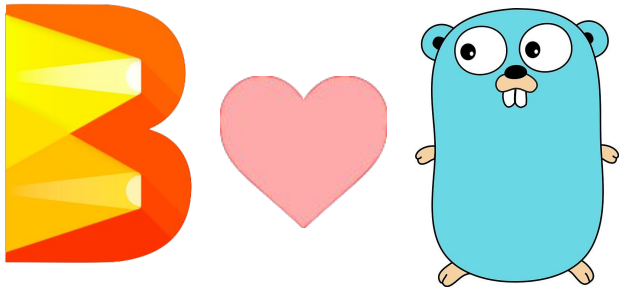
Go SDK

First user SDK completely based on Portability API.

```
func main() {
    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)

    if err := beamx.Run(context.Background(), p); err != nil {
        log.Fatalf("Failed to execute job: %v", err)
    }
}
```



Ongoing / Future work

- Full Beam model support (State, Windows, Triggers, etc)
- Metrics
- Invoke IO connectors between languages
- Multiple language pipelines
- Ergonomics (aka User/Dev eXperience)
- Production-readiness
- Other runners: Spark, ...
- Validation tests *'our TCK'*

Contribute

You are welcome to contribute!

- Try the portability work and help us report and fix issues.
- Multiple Jiras that need to be taken care of.
- Improve documentation
- New feature requests, new ideas.
- More SDKs (more languages) .net anyone please, etc
- More runners, improve existing, a native go one maybe?.

Not only for Portability, **Beam** is in a perfect shape to jump in.

First Stable Release. 2.0.0 API stability contract (May 2017)
Current: 2.5.0 (vote starting soon)

Contribute to Apache Beam (May 2018)

A vibrant community of contributors + companies:

Google, data Artisans, Talend, Ali Baba, Lyft, Yours?

Exciting Upcoming Features:

Portability, been able to run multiple languages on other runners
Go SDK, finally gophers have the right to Big Data
IO Connectors based on Splittable DoFn
Schema-aware PCollections and SQL improvements
New Libraries: Perfect moment to contribute yours !

Greetings

- **Lukasz Cwik**
- **Thomas Groh, Vikas Kedigehalli, Sourabh Baja**
- **Ben Sidhom, Axel Magnuson, Daniel Oliveira**
- **Kenneth Knowles, Henning Rohde, Valentyn Tymofieiev** (Google)
- **Aljoscha Krettek** (data Artisans)
- **Thomas Weise** (Lyft)
- The rest of the **Beam** community in general for being awesome.

* The nice slides with animations were created by Tyler Akidau and Frances Perry and used with authorization.

References

Portability

Portability Framework

Apache Beam

https://beam.apache.org

Join the mailing lists!

user-subscribe@beam.apache.org
dev-subscribe@beam.apache.org

Follow @ApacheBeam on Twitter

Thanks