Scaling up the OpenTelemetry Collector with Beam Go

Alex Van Boxel
Collibra
@alexvb

BEAM SUMMIT
NYC 2023

Alex Van Boxel

Principal System Architect

Collibra

Apache Beam

Committer (but you have to forgive me, it's been a while…)

Google Developer Expert

OpenTelemetry

The future is now

OpenTelemetry

Industry changing specification and implementation

The first time community members and industry partners come together to make a unified standard for telemetry

OpenTelemetry

Early Adopers are gamblers

We were early adapters. Jumping early on a specification is a gamble. But an informed gample:

- Merging OC and OT
- Industry support

OpenTelemetry

The telemetry signals: metrics, traces, and logs

The specification focuses on Traces, Metrics and Log with further expansion to Profiling

Collibra

Collibra Telemetry backbone

Owning your telemetry data

Vendor Independence

Removing lock-in at the collection side

We should always have the possibility of easily switching backend vendors. Without rolling out vendor dependent agents. OpenTelemetry collector promises vendor independent collection.

Owning our own telemetry data

Only when the protocol is open, can you own the data

OpenTelemetry has an open protocol (defined in Protobuf) and well defined semantic conventions. Only through this openness can you start building on top of the data.

Use Cases

Analytics- and operational use-cases

Operational use-cases

Operational use-cases we're moving off of Beam

Although Apache Beam is great for a lot of streaming use-cases… it is not a hammer.

Analytical use-cases

Use Apache Beam for what it's good for

Apache Beam is great at Analytics use-cases.

Metrics

Photo by Mitchel Boot on Unsplash

Traces

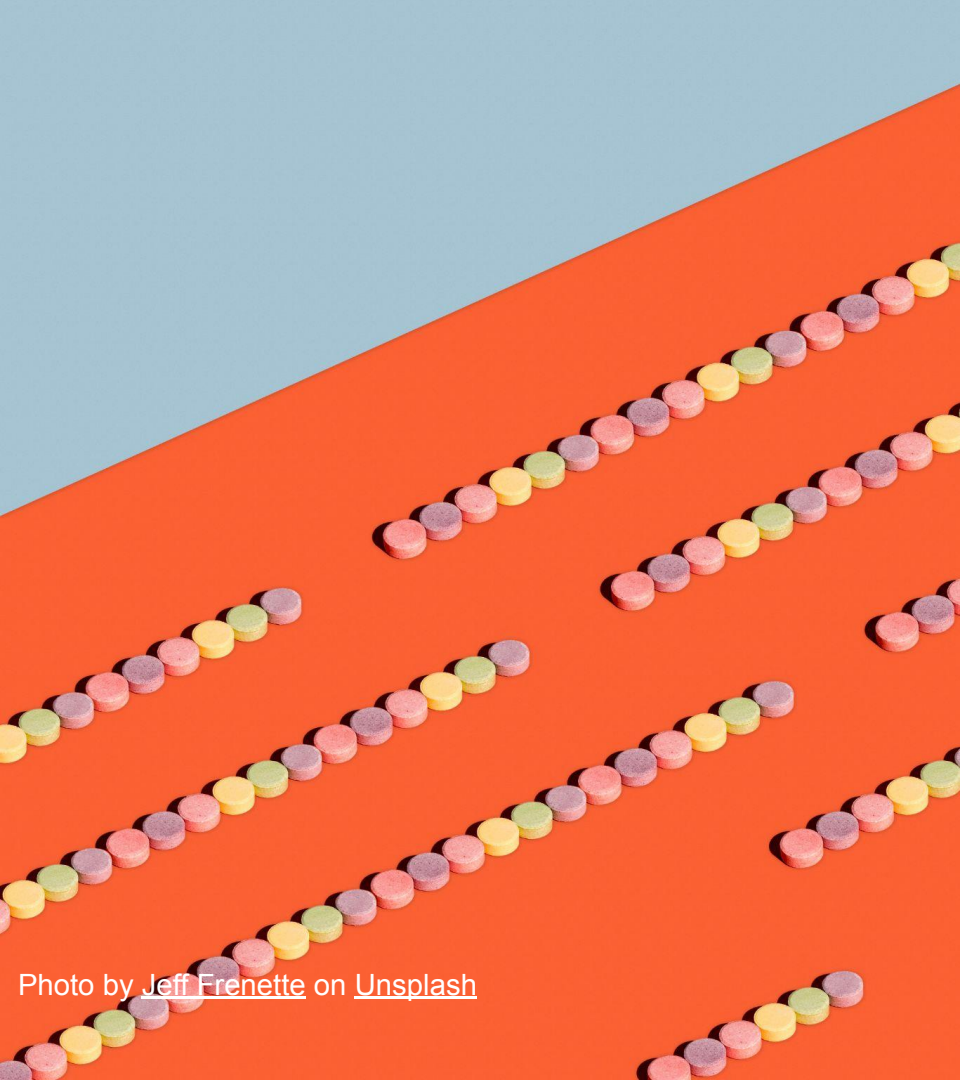


Photo by Jeff Frenette on Unsplash

Logs

Logs as analytics input

Classic logs are still the best source for analytics

Logs are being used since the beginning of time. Logs of the Big Data frameworks have their origins in processing logs:

- Hadoop
- BigQuery

Logs as analytics input

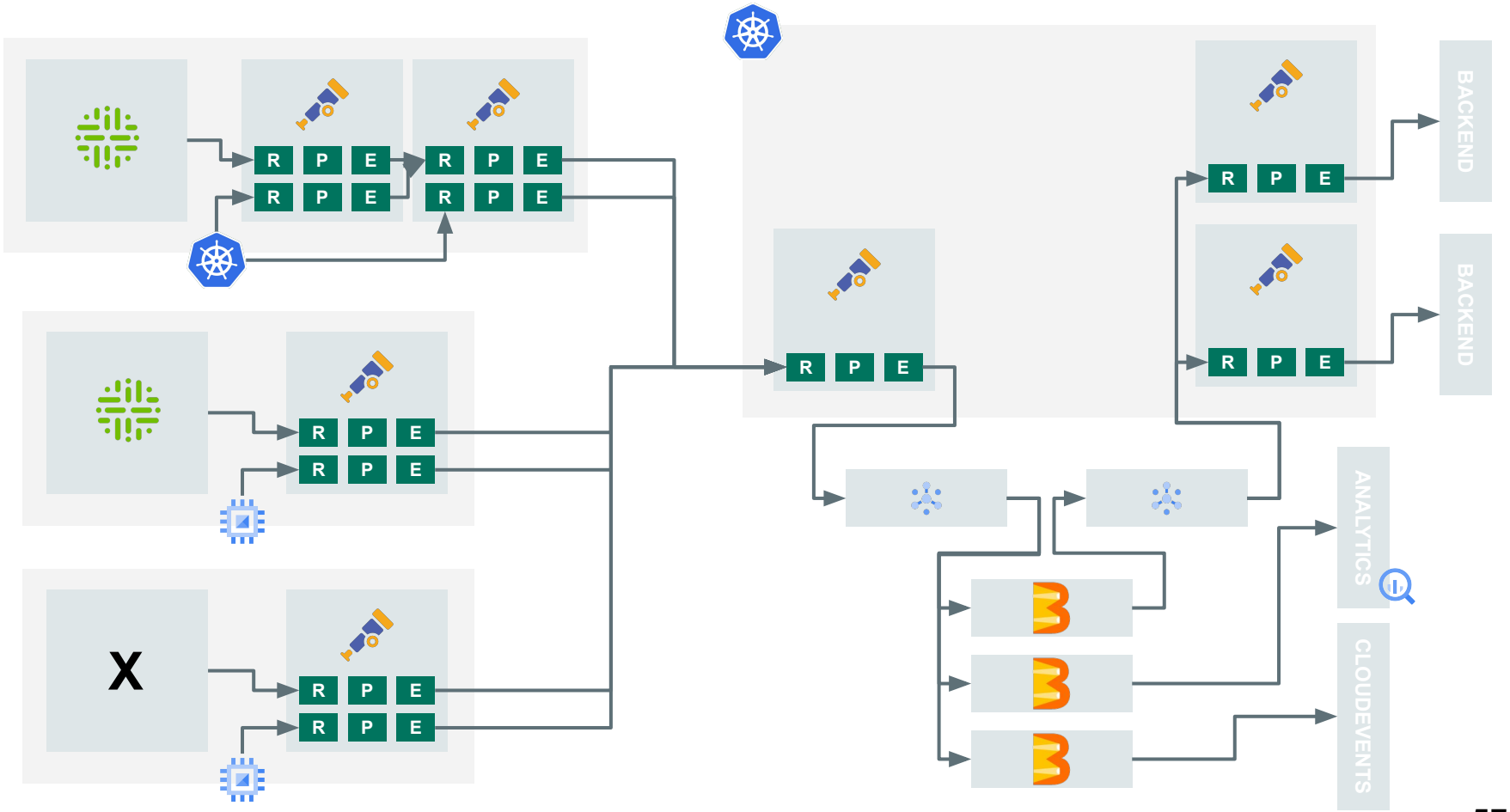
Structured Logs and Governance

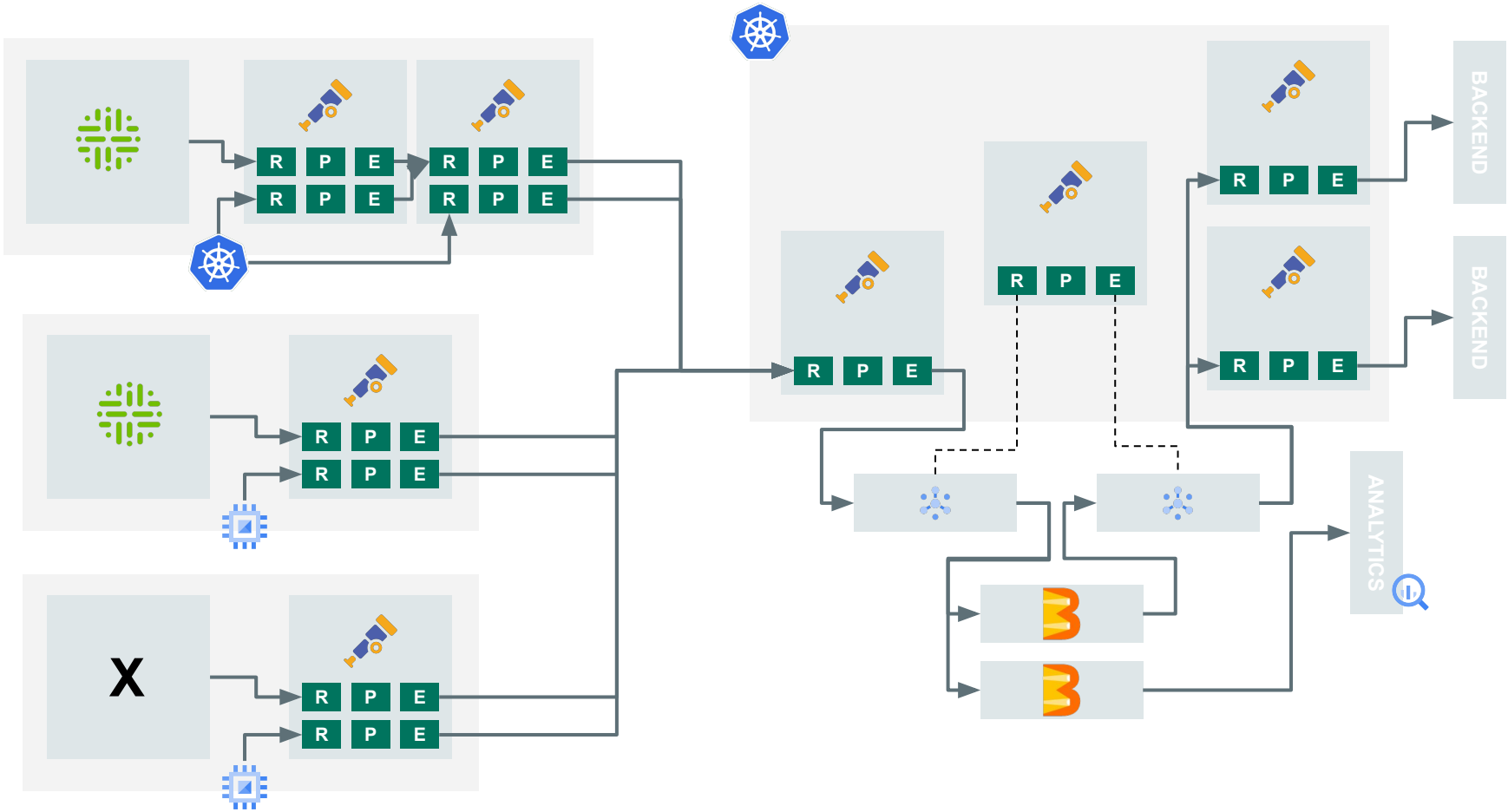
Don’t do naïeve logging. Go for structured logs. And try todo governance around it, you will get way more value out of your logs.

Collibra

Collector or Beam

Why not both?!


R P E
R P E
R P E
R P E
R P E
R P E
R P E
R P E
R P E
X
BACKEND
BACKEND
ANALYTICS
CLOUDEVENTS


R P E
R P E
R P E
R P E
R P E
R P E
R P E
R P E
R P E
R P E
X
BACKEND
BACKEND
ANALYTICS

Consolidating programming language

Go is the language of infrastructure

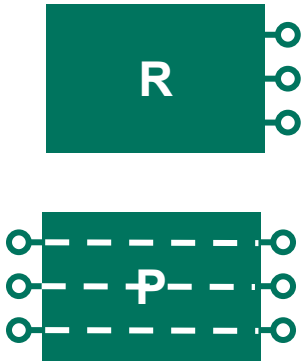
Go Beam SDK is a great addition to the Beam ecosystem. It's also the language uses in the infrastructure space.

Collector pipelines on Beam

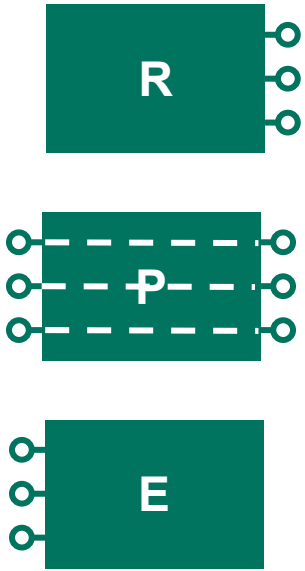
Why re-implement, when you can reuse

OpenTelemetry collector building blocks

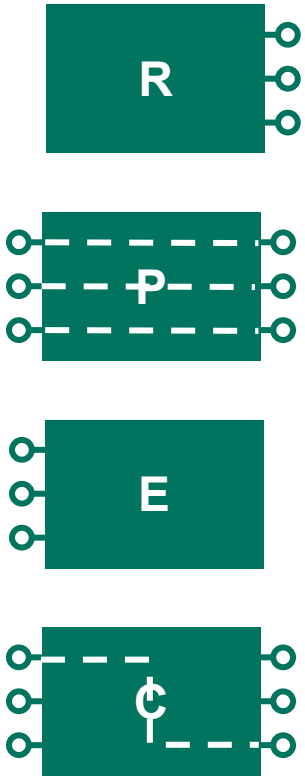
OpenTelemetry collector building blocks



OpenTelemetry collector building blocks



OpenTelemetry collector building blocks



What if we could natively run the components as DoFns

If we can reuse the components we have better code reuse

We have lots of awesome component. Why not reuse them. Even basic filters could be useful in analyticals pipelines.

Why build the pipeline in code if you already have a way to build pipelines

```
processors:
  attributes/example:
    actions:
      - key: copy_key
        from_attribute:
key_original
      - key: account_password
        action: delete
      - key: account_email
        action: hash
      - key: http.status_code
        action: convert
        converted_type: int
```

Processors are suited. Neither receivers, nore exporters are suited

Processors pretty easy to support, you wrap them with a DoFn and you are done. Some you can ignore.

Exporters and Receivers are less easy and you need custom implementations that make use of Beam I/O.

OTTL: OpenTelemetry Transformation Language

Going a step further, could we make a native OTTL transform

This comes in the theorital spaces, but the OpenTelemetry Transformation Language is a good candidate for going even further.

```yaml
transform:
  error_mode: ignore
  log_statements:
    - context: log
      statements:
        # Parse body as JSON and merge the resulting map with the cache map, ignoring non-json bodies.
        # cache is a field exposed by OTTL that is a temporary storage place for complex operations.
        - merge_maps(cache, ParseJSON(body), "upsert") where IsMatch(body, "^\\{")

        # Set attributes using the values merged into cache.
        # If the attribute doesn't exist in cache then nothing happens.
        - set(attributes["attr1"], cache["attr1"])
        - set(attributes["attr2"], cache["attr2"])

        # To access nested maps you can chain index ([]) operations.
        # If nested or attr3 do no exist in cache then nothing happens.
        - set(attributes["nested.attr3"], cache["nested"]["attr3"])
```

OTTL: Exploring extensions

Why not extend the OTTL with, windows, state and timers

OTTL is build for extensibility so maybe we could extend the OTTL with the windowing functions, just like Beam SQL.

Exploring the analytics side

SQL transform support would be nice

Eventually we will need to integrate Beam SQL in hybrid pipelines. As this is for analytical use-cases no reuse is expected with Collector only use cases.

Collibra

Conclusion

and learnings

What would we do different?

LAST YEARS SLIDE

- As the engineerings in the operations we would now start investigating the **Go SDK** (two years ago it was too early)
- Some parts would be a better fit for the **opentelemetry-collector (pipeline)**, switching to the Go SDK maybe makes it easier to share code.

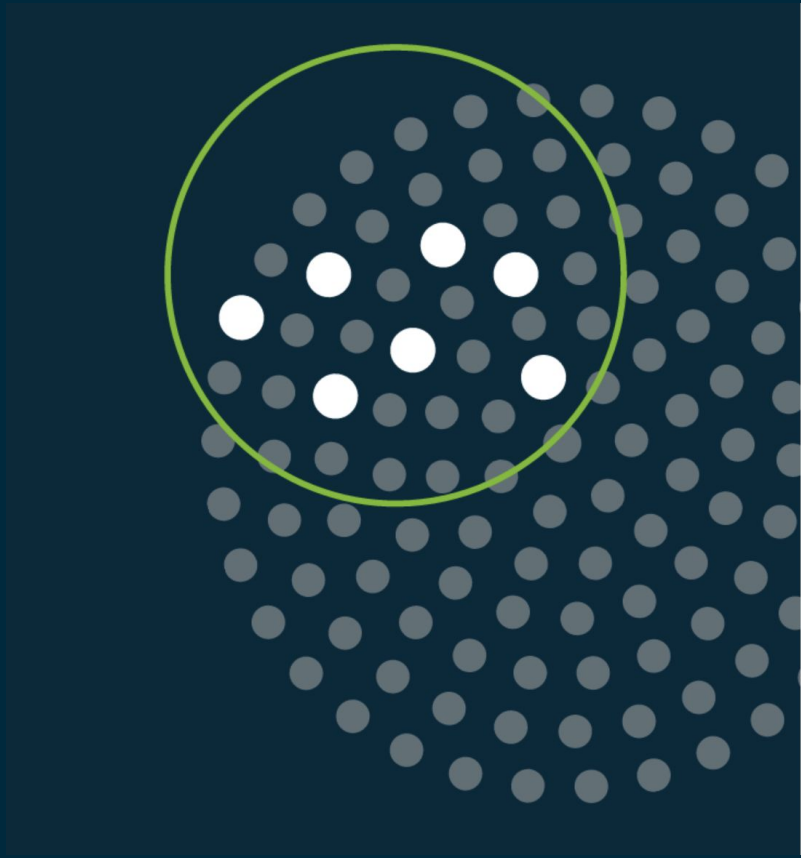
What is happening now?

- The team is focussing on re-implementing code in the collector, it’s easier to migrate between version as it’s nearing General Availability.
- Beam GO and the pipeline config translator should make it even easier to share code and use existing collector processors found in the collector on top of Beam

Thank you

https://github.com/alexvanboxel/opentelemetry-beam

@alexvb




BEAM
SUMMIT