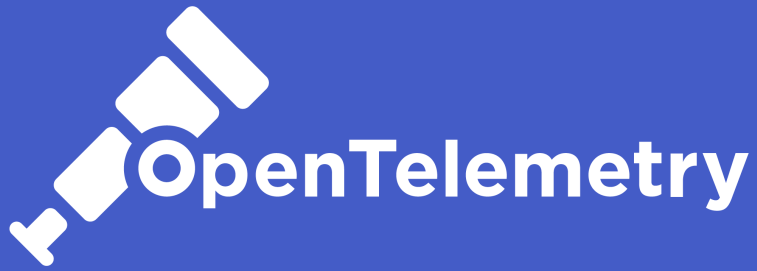
Log Support in



KubeCon Europe 2021

What is OpenTelemetry?

OpenTelemetry makes robust, portable telemetry a built-in feature of cloud-native software.

OpenTelemetry provides a single set of APIs, libraries, agents, and collector services to capture distributed traces and metrics from your application. You can analyze them using Prometheus, Jaeger, and other observability tools.

OpenTelemetry

is the second most active project in CNCF today!

(per CNCF DevStats)

Everyone is Contributing and Adopting

Cloud Providers

AWS | Azure | GCP

Vendors

Every major vendor!

End-users

Mailchimp (PHP)

Postmates (Erlang)

Shopify (Ruby)

Other

Jaeger > OtelCol

Fluent-bit <3 log SIG

Envoy roadmap

OpenMetrics roadmap

Spring roadmap

https://github.com/open-telemetry/community/blob/master/ADOPTERS.md
https://medium.com/jaegertracing/jaeger-embraces-opentelemetry-collector-90a545cbc24

OpenTelemetry

Traces: Stable
Metrics: Beta
Logs: Alpha

@smflanders

flands

https://sflanders.net

Steve Flanders

Director of Engineering, Splunk
OpenTelemetry Website Maintainer
OpenTelemetry Collector Triager
CNCF SIG-Observability Member

Previously:

- Head of Product, Omnition
- Senior Manager for Logs, VMware

Components

OpenTelemetry

Cloud Native Telemetry

Telemetry "verticals"

Telemetry "layers"

	Tracing	Metrics	Logs, etc
Instrumentation APIs		foreach(language)	
Canonical implementations		foreach(language)	
Data infrastructure		collectors, sidecars, etc	
Interop formats		wire protocol, data formats for trace data, metrics, logs, etc	

OpenTelemetry

OpenTelemetry Components

Specification

API, SDK, Data

Instrumentation Libraries

Single library per language for all signals

Collector

Receive, process, and export data

Specification

- Organized into “signals”:
  - Traces
  - Metrics
  - Logs
- Each signal has
  - Data model
  - Instrumentation API
  - Instrumentation SDK
  - Collector support
  - Contrib packages
  - Context
  - Resources
  - Semantic conventions

OpenTelemetry Log Data Model

https://github.com/open-telemetry/opentelemetry-specification/blob/main/specification/logs/data-model.md

Log and Event Record Definition (all optional):

- Field: Timestamp
- Trace Context Fields: TraceId, SpanId, TraceFlags
- Severity Fields: SeverityText, SeverityNumber
- Other Fields: Name, Body, Resource, Attributes

OpenTelemetry Log Data Model

https://github.com/open-telemetry/opentelemetry-specification/blob/main/specification/logs/data-model.md

Log and Event Record Definition (all optional):

- Field: **Timestamp**
- Trace Context Fields: TraceId, SpanId, TraceFlags
- Severity Fields: **SeverityText**, **SeverityNumber**
- Other Fields: Name, **Body**, Resource, Attributes

```
{
  "Timestamp": 1586960586000,
  "Body": {
    "i": "am",
    "an": "event",
    "of": {
      "some": "complexity"
    }
  }
}
```

```
{
  "Timestamp": 1586960586000,
  "SeverityText": "INFO",
  "Body": {
    "i": "am",
    "an": "event",
    "of": {
      "some": "complexity"
    }
  }
}
```

OpenTelemetry Log Data Model

https://github.com/open-telemetry/opentelemetry-specification/blob/main/specification/logs/data-model.md

Log and Event Record Definition (all optional):

- Field: **Timestamp**
- Trace Context Fields: **TraceId**, **SpanId**, **TraceFlags**
- Severity Fields: **SeverityText**, **SeverityNumber**
- Other Fields: Name, **Body**, **Resource**, **Attributes**

```
{
  "Timestamp": 1586960586000,
  "Attributes": {
    "http.status_code": 500,
    "http.url": "http://example.com",
    "my.custom.application.tag": "hello",
  },
  "Resource": {
    "service.name": "donut_shop",
    "service.version": "2.0.0",
    "k8s.pod.uid": "1138528c-c36e-11e9-a1a7-42010a800198",
  },
  "TraceId": "f4dbb3edd765f620", // this is a byte sequence
                                 // (hex-encoded in JSON)
  "SpanId": "43222c2d51a7abe3",
  "SeverityText": "INFO",
  "Body": "20200415T072306-0700 INFO I like donuts"
}
```

```
{
  "Timestamp": 1586960586000,
  "Attributes": {
    "http.status_code": 500,
    "http.url": "http://example.com",
    "my.custom.application.tag": "hello",
  },
  "Resource": {
    "service.name": "donut_shop",
    "service.version": "2.0.0",
    "k8s.pod.uid": "1138528c-c36e-11e9-a1a7-42010a800198",
  },
  "TraceId": "f4dbb3edd765f620", // this is a byte sequence
                                 // (hex-encoded in JSON)
  "SpanId": "43222c2d51a7abe3",
  "SeverityText": "INFO",
  "Body": "20200415T072306-0700 INFO I like donuts"
}
```

```
{
  "Timestamp": 1586960586000,
  "Attributes": {
    "http.status_code": 500,
    "http.url": "http://example.com",
    "my.custom.application.tag": "hello",
  },
  "Resource": {
    "service.name": "donut_shop",
    "service.version": "2.0.0",
    "k8s.pod.uid": "1138528c-c36e-11e9-a1a7-42010a800198",
  },
  "TraceId": "f4dbb3edd765f620", // this is a byte sequence
                                 // (hex-encoded in JSON)
  "SpanId": "43222c2d51a7abe3",
  "SeverityText": "INFO",
  "Body": "20200415T072306-0700 INFO I like donuts"
}
```

```
{
  "Timestamp": 1586960586000,
  "Attributes": {
    "http.status_code": 500,
    "http.url": "http://example.com",
    "my.custom.application.tag": "hello",
  },
  "Resource": {
    "service.name": "donut_shop",
    "service.version": "2.0.0",
    "k8s.pod.uid": "1138528c-c36e-11e9-a1a7-42010a800198",
  },
  "TraceId": "f4dbb3edd765f620", // this is a byte sequence
                                 // (hex-encoded in JSON)
  "SpanId": "43222c2d51a7abe3",
  "SeverityText": "INFO",
  "Body": "20200415T072306-0700 INFO I like donuts"
}
```

```
{
  "Timestamp": 1586960586000,
  "Attributes": {
    "http.status_code": 500,
    "http.url": "http://example.com",
    "my.custom.application.tag": "hello",
  },
  "Resource": {
    "service.name": "donut_shop",
    "service.version": "2.0.0",
    "k8s.pod.uid": "1138528c-c36e-11e9-a1a7-42010a800198",
  },
  "TraceId": "f4dbb3edd765f620", // this is a byte sequence
                                 // (hex-encoded in JSON)
  "SpanId": "43222c2d51a7abe3",
  "SeverityText": "INFO",
  "Body": "20200415T072306-0700 INFO I like donuts"
}
```

OpenTelemetry Log Data Model

https://github.com/open-telemetry/opentelemetry-specification/blob/main/specification/logs/data-model.md

Log and Event Record Definition (all optional):

- Field: **Timestamp**
- Trace Context Fields: **TraceId**, **SpanId**, **TraceFlags**
- Severity Fields: **SeverityText**, **SeverityNumber**
- Other Fields: Name, **Body**, **Resource**, **Attributes**

Can be converted from/to:

- Open standards: Apache HTTP Server, Elastic Common Schema, Log4j, Syslog (RFC5424), Zap
- Vendor standards: Amazon CloudTrail, Google Cloud Logging, Splunk HEC, Windows Event Viewer

OpenTelemetry Log Data Model

https://github.com/open-telemetry/opentelemetry-specification/blob/main/specification/logs/data-model.md

Summary

- OTel **Log Record** = “Log” and “Event” = Structured JSON payload
- Definition is made up of one or more fields (all optional)
- Supports both OTel and W3C Context
- Offers error semantics
- Can be converted from/to many open source or vendor specific formats
- May be embedded (span events) or standalone (all other)
- Currently experimental so may change in the future!

Instrumentation Libraries

- API: How an application is instrumented to generate telemetry data
- SDK: How telemetry data is processed (e.g., batching) and exported (e.g., otlp)
- Also takes advantage of: Context, Resources, Semantic Conventions, etc.

OpenTelemetry

OpenTelemetry Log Instrumentation

Java Options

- Java manual logging reference implementation:
  https://github.com/open-telemetry/opentelemetry-java/tree/main/sdk-extensions/logging
- Java logger MDC auto-instrumentation:
  https://github.com/open-telemetry/opentelemetry-java-instrumentation/blob/main/docs/logger-mdc-instrumentation.md

```
logging.pattern.console = %d{yyyy-MM-dd HH:mm:ss} - %logger{36} - %msg trace_id=%X{trace_id}
span_id=%X{span_id} trace_flags=%X{trace_flags} %n
```

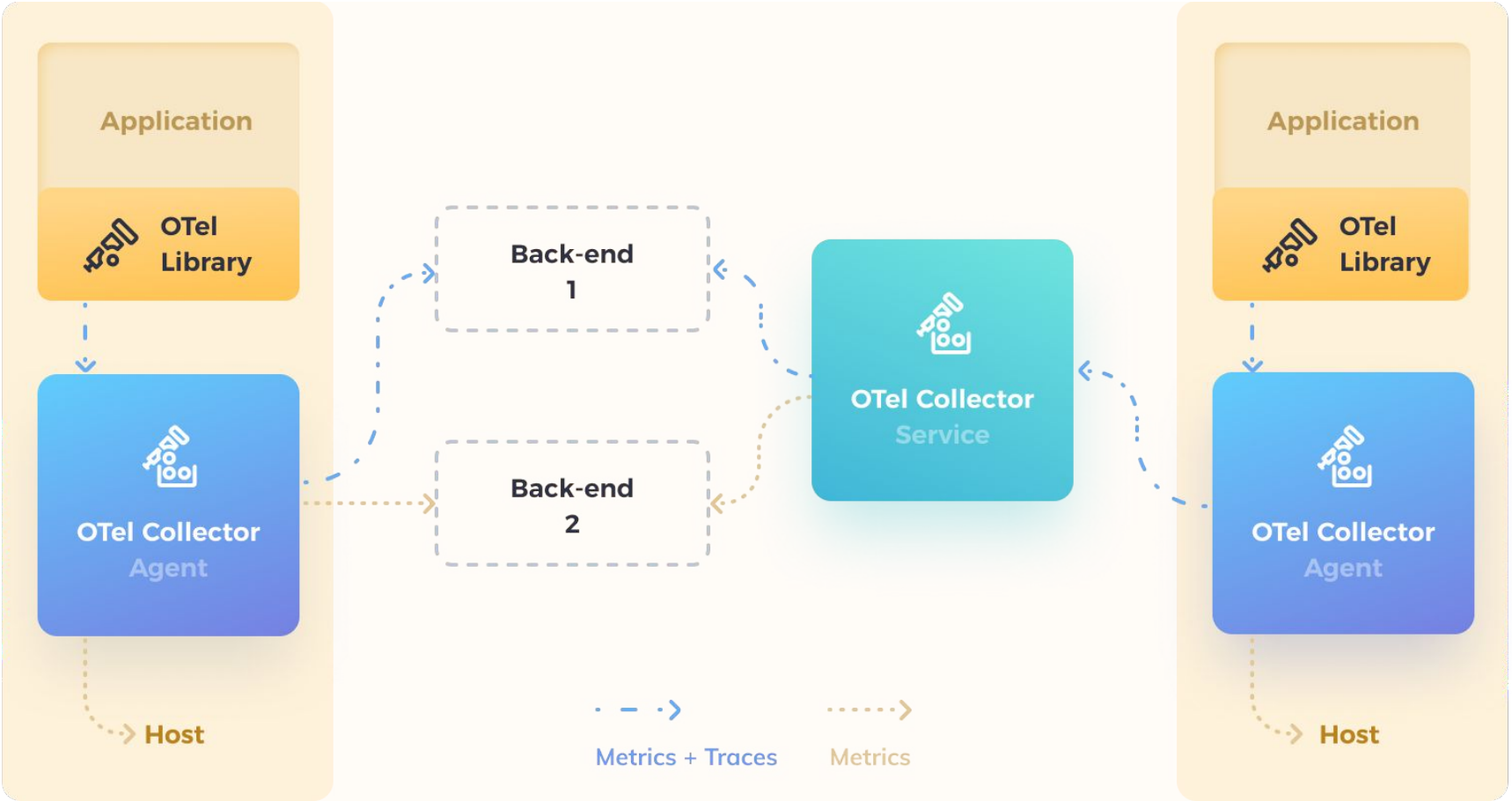
OpenTelemetry Log Instrumentation

Summary

- Some instrumentation libraries offer automatic trace injections into logs
  - Java: https://github.com/open-telemetry/opentelemetry-java-instrumentation/blob/main/docs/logger-mdc-instrumentation.md
  - Python: https://github.com/open-telemetry/opentelemetry-python-contrib/tree/main/instrumentation/opentelemetry-instrumentation-logging
- Manual log instrumentation considered in the future
- Without a stable data model, manual log instrumentation not ready for adoption yet
- Remember 1) traces 2) metrics 3) logs; PRs welcomed!

Collector

- Components configured via pipelines:
  - Receivers (push or pull-based)
  - Processors
  - Exporters (push or pull-based)
- Offers:
  - Translation between formats
  - CRUD metadata operations
  - Resource support



OTel reference architecture


Otel Collector
OTLP
Jaeger
Prometheus
Receivers
Extensions: health, pprof, zpages
Batch
. . .
Attributes
Processors
Batch
. . .
Filter
Processors
Exporters
OTLP
Jaeger
Prometheus

OpenTelemetry Log Collection

Available today

- Receivers
  - Filelog (tail)
  - Fluent forward
  - OTLP
  - Splunk HEC
- Processors
  - Attributes
  - Batch
  - Resource detection
- Exporters
  - Alibaba Cloud Log Service
  - Elasticsearch (WIP)
  - F5 Cloud Exporter
  - Loki
  - OTLP
  - Splunk HEC
  - Sumo Logic

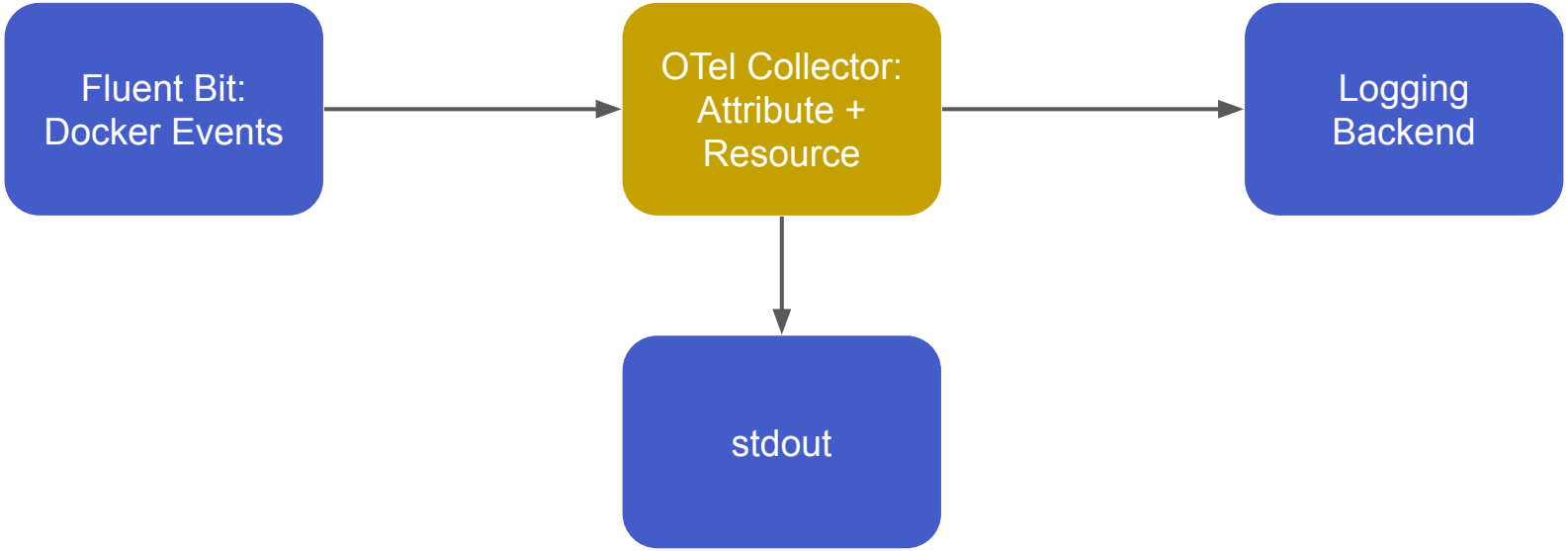
OpenTelemetry Log Collection

Stanza donation by observIQ being integrated now! (https://github.com/open-telemetry/community/issues/605)

- Receivers
  - Filelog (tail)
  - Journald
  - Syslog
  - TCP/UDP
  - Windows eventlog
- Processors
  - Attributes
  - Batch
  - Filter
  - Parser
  - Resource detection
- Exporters
  - Every destination that supports!

DEMO!

OpenTelemetry



OpenTelemetry

Next Steps

- Join a SIG: https://github.com/open-telemetry/community#special-interest-groups
- Join the conversation:
  - Each SIG leverages GitHub discussions
  - CNCF Slack: https://cloud-native.slack.com
- Submit a PR (consider good-first-issue and help-wanted labels)

Thank You!

OpenTelemetry