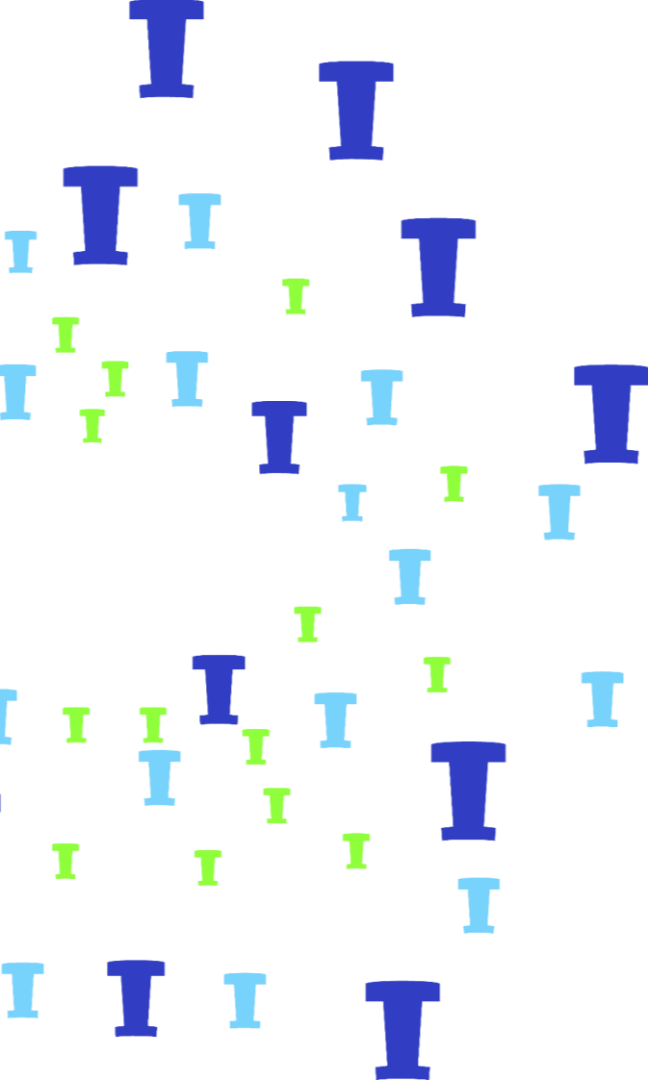


UNIVERSIDAD ICESI

TU FUTURO A OTRO NIVEL

# Seguimiento Distribuido: Jaeger

Jhon Robert Quintero

# Intro to OpenTelemetry

- 🚀 How OpenTelemetry can help me
- 🚀 Overview of the architecture
- 🚀 Basic deployment

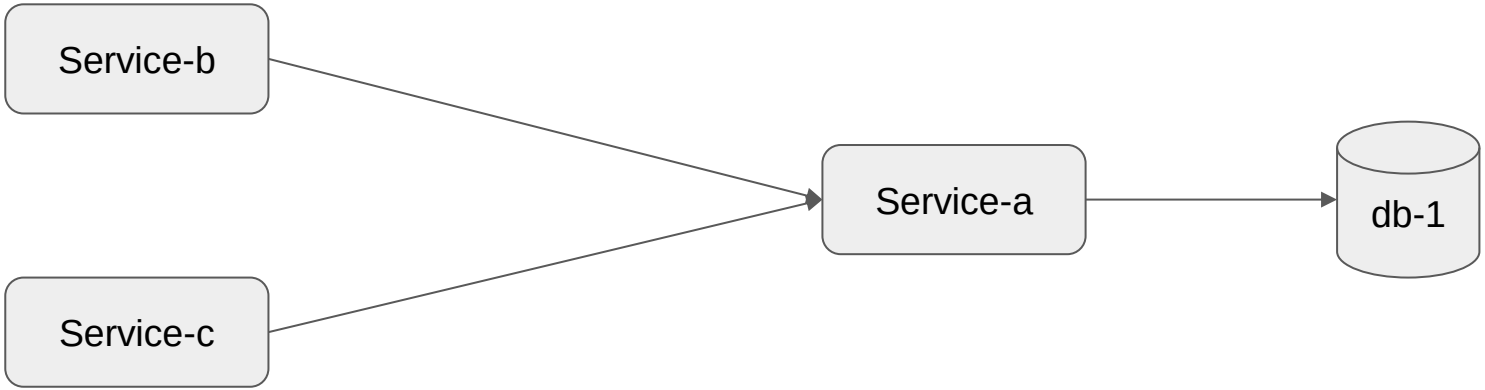
# Let’s start!

- Let’s start by an example when would OpenTelemetry be beneficial
- You got an alert (based on logs) that service-a isn’t able to write to db-1

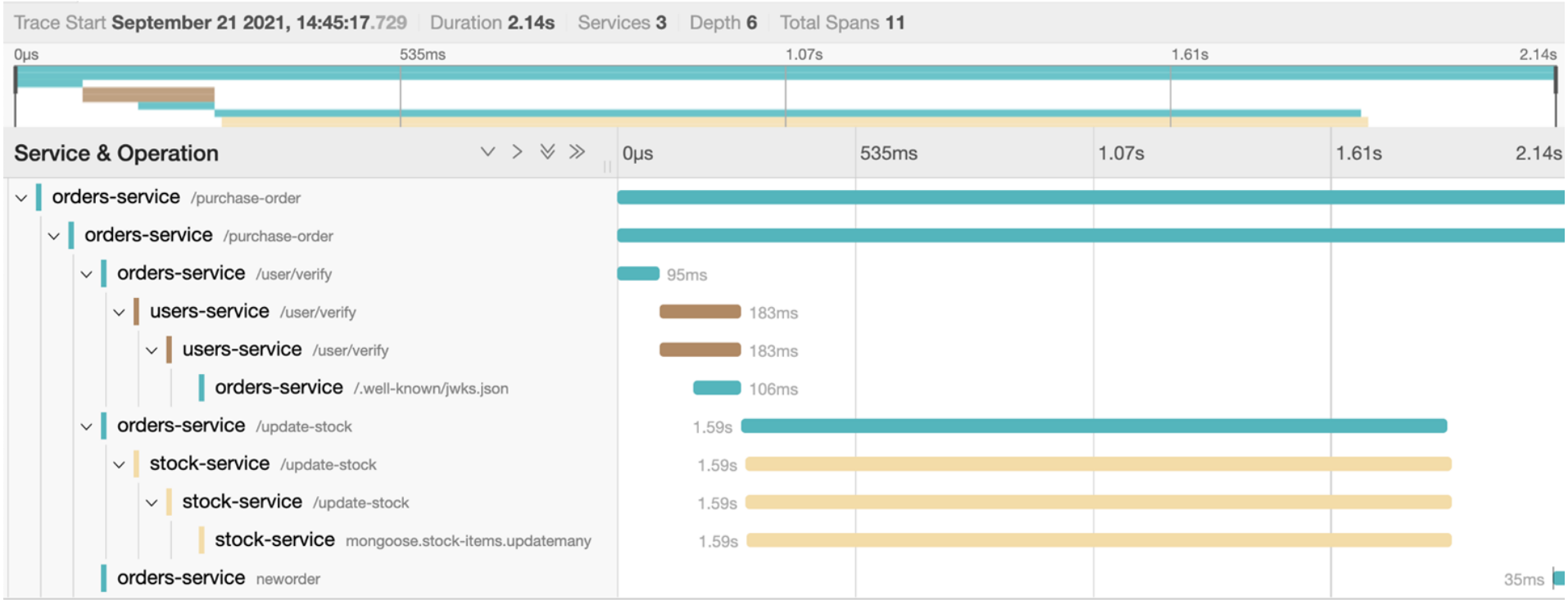
# Service-a can’t write to db-1

- Error logs show exception
- Metric show high CPU in db1
- Maybe an increase in traffic?
  - Which HTTP Route is causing query to db-1?
  - Other types of communication?

# Service-a can’t write to db-1




Logs - there is an error, can’t write to db-1
Metrics - high CPU, the “technical” why
Trace - the context, the “path” within the system
---
Checking the right metric / code

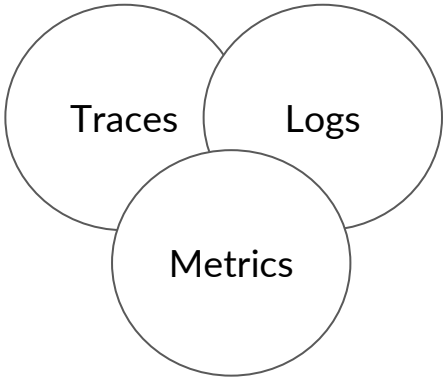

Trace Start September 21 2021, 14:45:17.729
Duration 2.14s
Services 3
Depth 6
Total Spans 11
0µs
535ms
1.07s
1.61s
2.14s
Service & Operation
orders-service /purchase-order
orders-service /purchase-order
orders-service /user/verify
95ms
users-service /user/verify
183ms
users-service /user/verify
183ms
orders-service /.well-known/jwks.json
106ms
orders-service /update-stock
1.59s
stock-service /update-stock
1.59s
stock-service /update-stock
1.59s
stock-service mongoose.stock-items.updatemany
1.59s
orders-service neworder
35ms

# Distributed systems monitoring

- Logs - The application story
- Metrics - Numbers telling statistical facts about the system
- Traces - The context of why things are happening
  - Trace events

Those would be the three pillars of Observability

# OpenTelemetry three pillars




Trace can have logs inside it

Logs can point to trace

Metrics be correlated via time to both.

# OpenTelemetry

- The glue to collect the three pillars together under a unified SDK.
- Under CNCF
- One specification, Implementation for every programming language
- ???

# OpenTelemetry

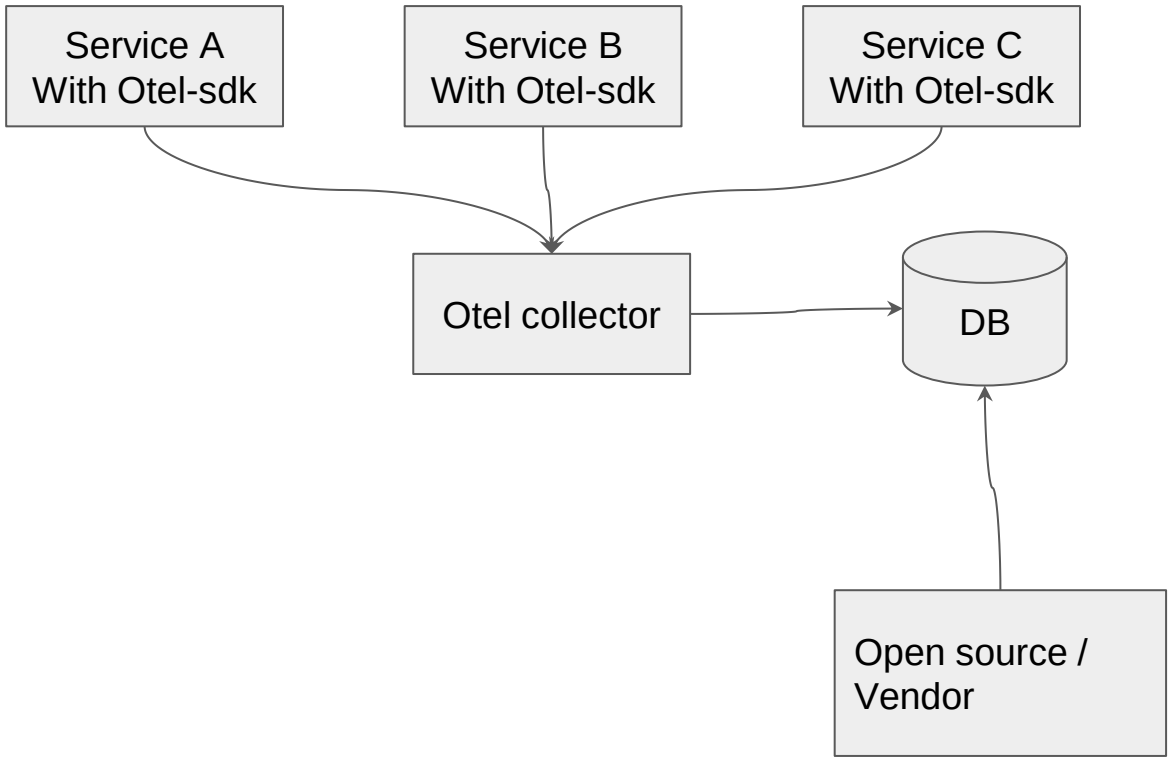
- An SDK that collects the three pillars
- It will ship it to a backend, then a DB
- We will need a visualization layer to display traces, logs and metrics

# OpenTelemetry - The “stack”

Your app

Otel backend

Visualization

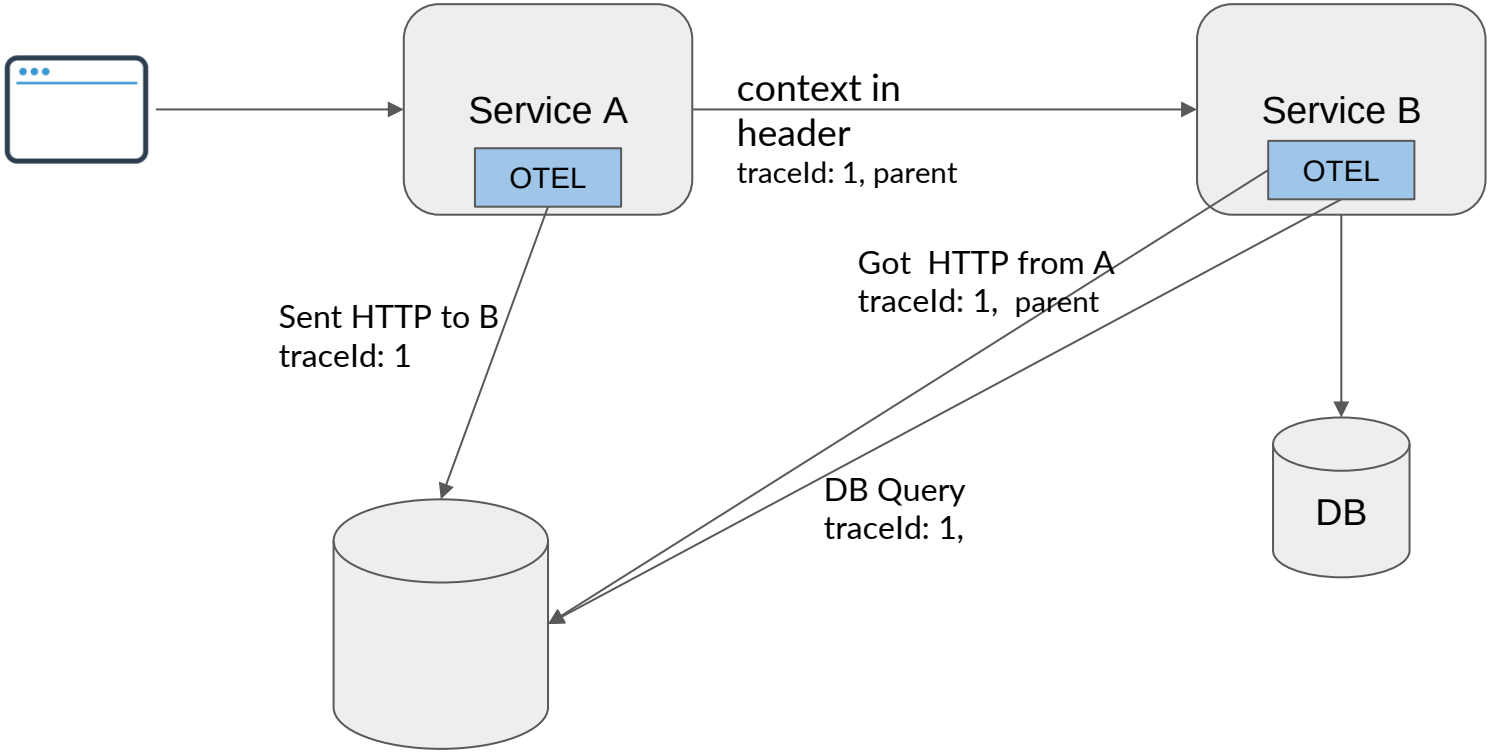


# OpenTelemetry “stack”

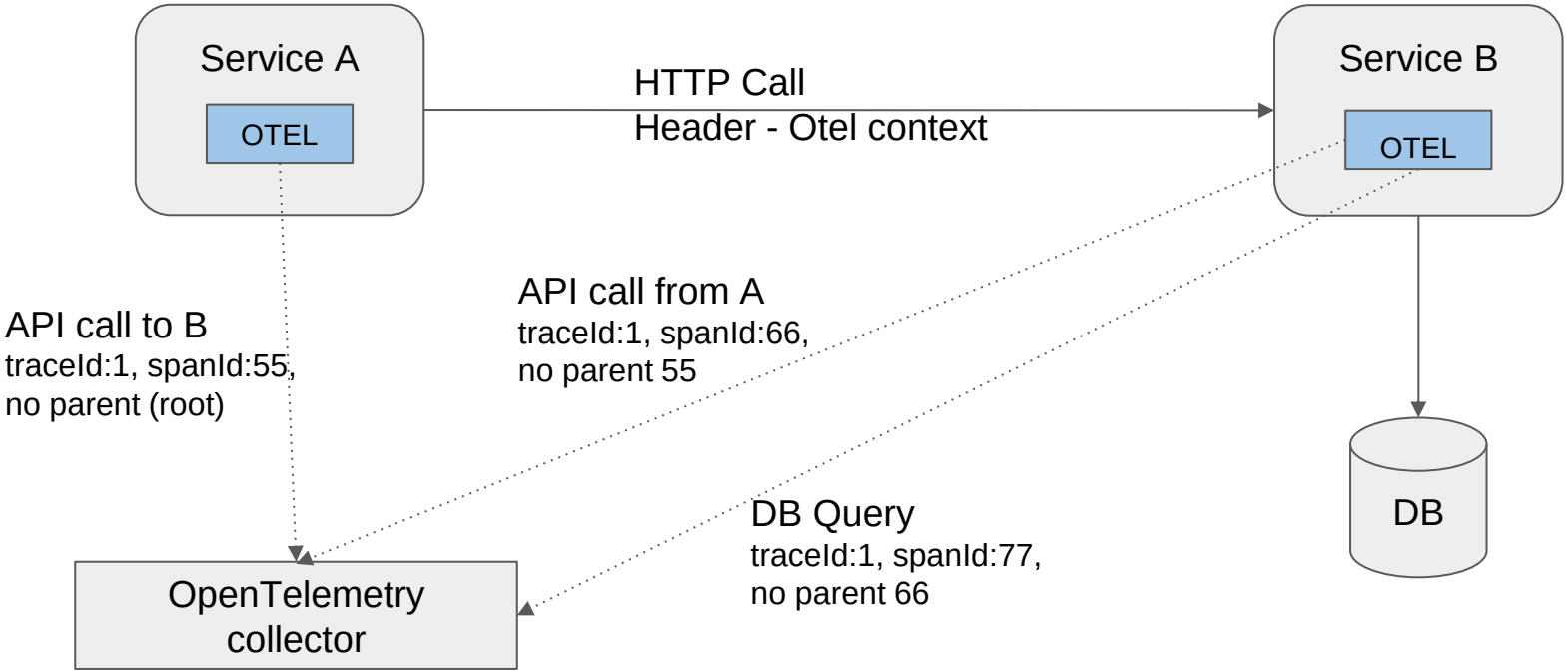
- SDK - Collects traces, logs and metrics and export them
- Collector - receivers telemetry, process it and export it
- A DB to store telemetry data
- Visualization layer

As we progress we will see that not all are mandatory

# OpenTelemetry - How does the SDK works??



# OpenTelemetry - How does the SDK works??



# What data did we collect

Log - text, extra data, time, severity + *traceId*

Metric - numeric value, time

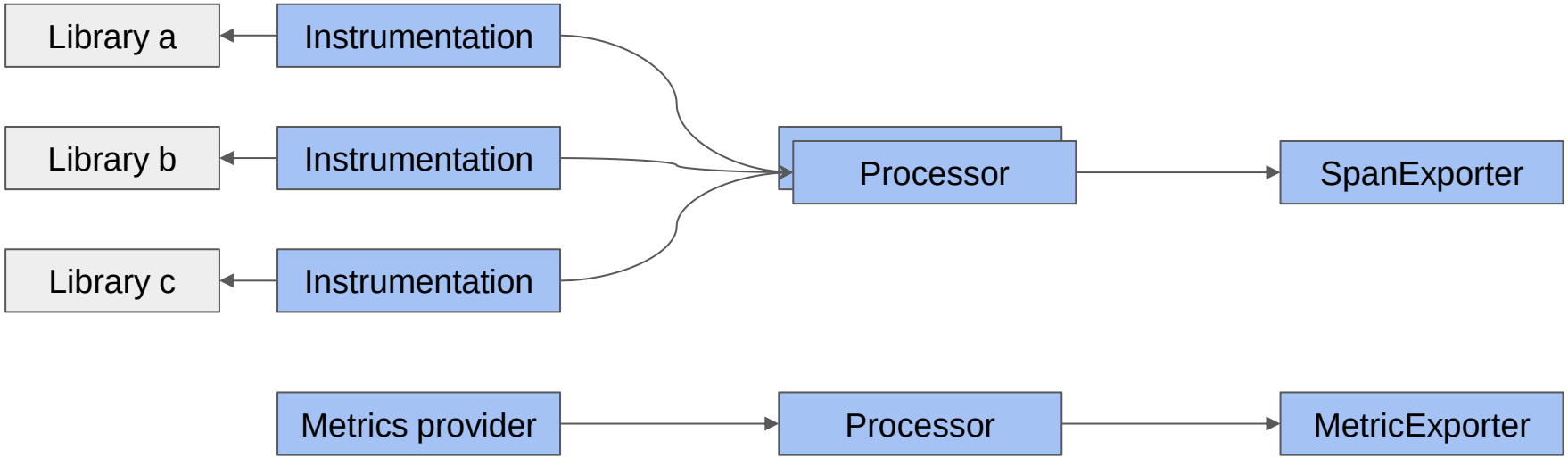
Trace:

- Id
- List of spans
  - Structured data about the span
  - Start time
  - Duration
  - Structured data about errors
  - Structured data about *events* (can be considered as logs)

# What is the purpose of OpenTelemetry SDK

- Collect data about the application
- Propagate the context between services
- Ship it somewhere

# OpenTelemetry SDK - collecting & send data



# About instrumentation

- Automatic
  - Patches / attaches to a library
  - Collect data library activities in runtime
  - Produces spans based on specification and semantic-conventions
  - May offer additional configuration / features
  - List of all auto-instrumentation: https://opentelemetry.io/registry/

- Manual
  - Application developer writes dedicated code
  - Starts and end span, set status
  - Adding attributes and events

# Common use cases for manual instrumentations

- Internal activities (timers)
- Adding data (user id)
- Unsupported auto instrumentation
  - Episode 5 will be about creating your own instrumentation

# About Processors

- Responsibilities
  - Act as pipeline observes, can change the data or omit it
  - Enforce sampling
- Pre-defined span-processors
  - SimpleProcessor
  - BatchProcessor
  - MultipleProcessor
  - NoopSpanProcessor
- Pre-defined metric-processors
  - UngroupedProcessor
  - ExactProcessor (used by prometheus exporter)

# About Exporters

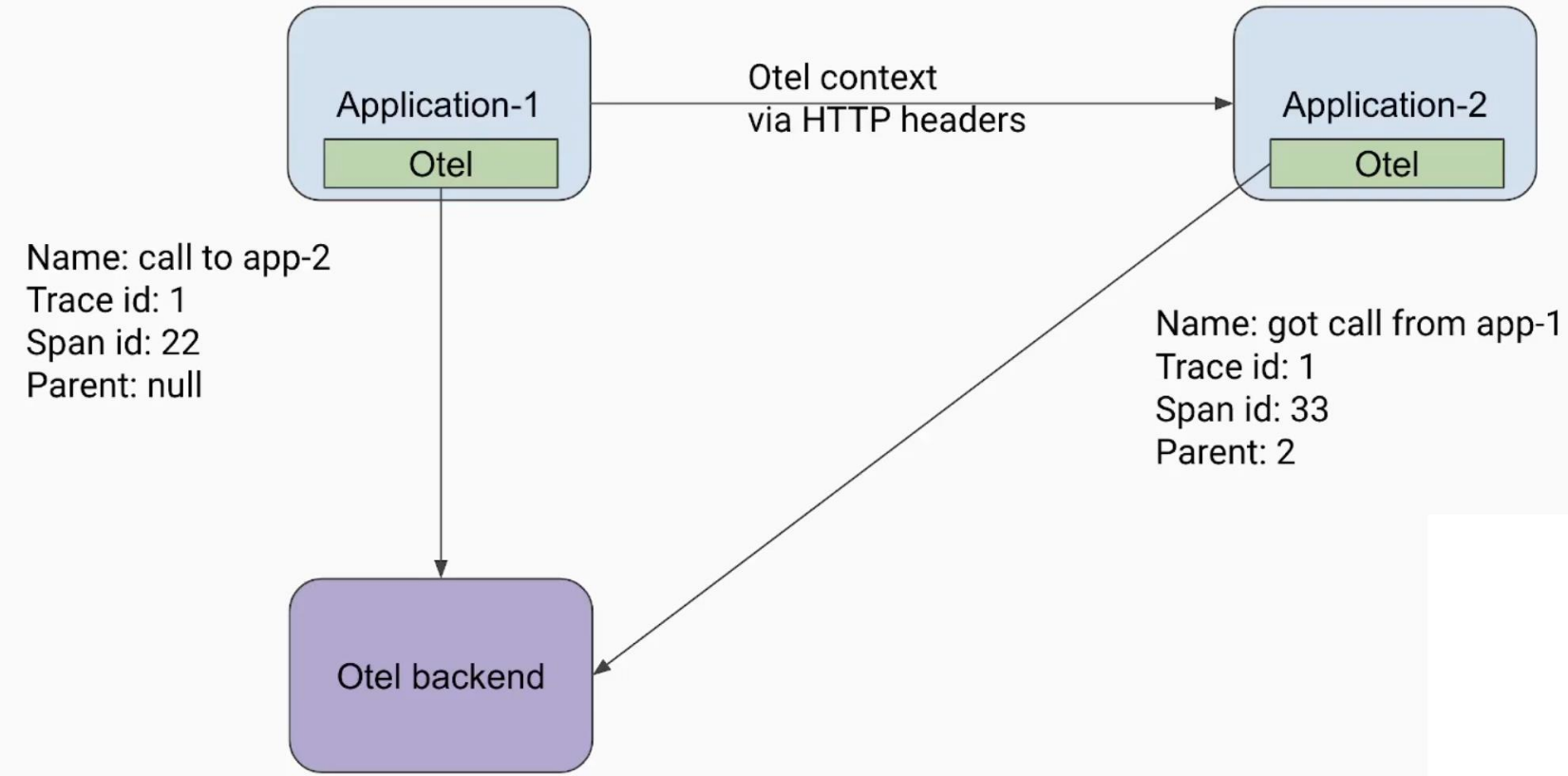
- Responsibilities
  - Send the data somewhere
  - Authentication

- Pre-defined exporters
  - Jaeger / zipkin / collector exporter
  - In memory exporter
  - Console exporter
  - OTLP exporter (metrics + spans)
  - Noop exporter

# About Exporters

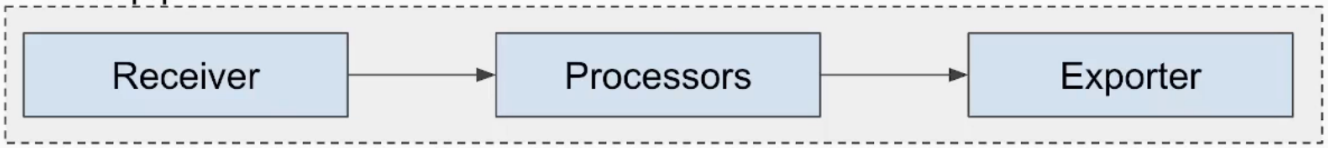
- Protocols:
  - HTTP
  - gRPC
- Formats
  - JSON
  - proto

# Traces are so cool! how does it work?



# Collector flow

Traces pipeline




Metrics pipeline


Logs pipeline

# We are going to work on…

1. Using Opentelemetry Java Agent
2. Using OpenTelemetry API
3. Using OpenTelemetry Collector

# Additional resources

https://www.aspecto.io/blog/why-and-how-to-run-jaeger-locally/

https://www.aspecto.io/blog/how-to-deploy-jaeger-on-aws-a-comprehensive-step-by-step-guide/

https://opentelemetry.io/docs/