Instrumentação com OpenTelemetry

Técnicas e Boas Práticas

Olly GARDEN

Prazer em conhecer!

Juraci Paixão Kröhling
Software Engineer

- Programador @ OllyGarden
- Criador do Dose de Telemetria
- Comitê de Governança do OTel
- Embaixador da CNCF
- Organizador OTel Night Berlin
- Emérito: OTel Collector, Jaeger, ...

Agenda

- OpenTelemetry (OTel) em 1 minuto
- Instrumentação Manual
- Bibliotecas instrumentadas
- Instrumentação “zero code” (agentes e via eBPF)
- Instrumentação em tempo de compilação

OpenTelemetry (OTel)

Juliano Costa
@jcosta.dev

Ou TelemetriaAberta, tb conhecida como TelmA 😂🤣
Gostei mais dessa

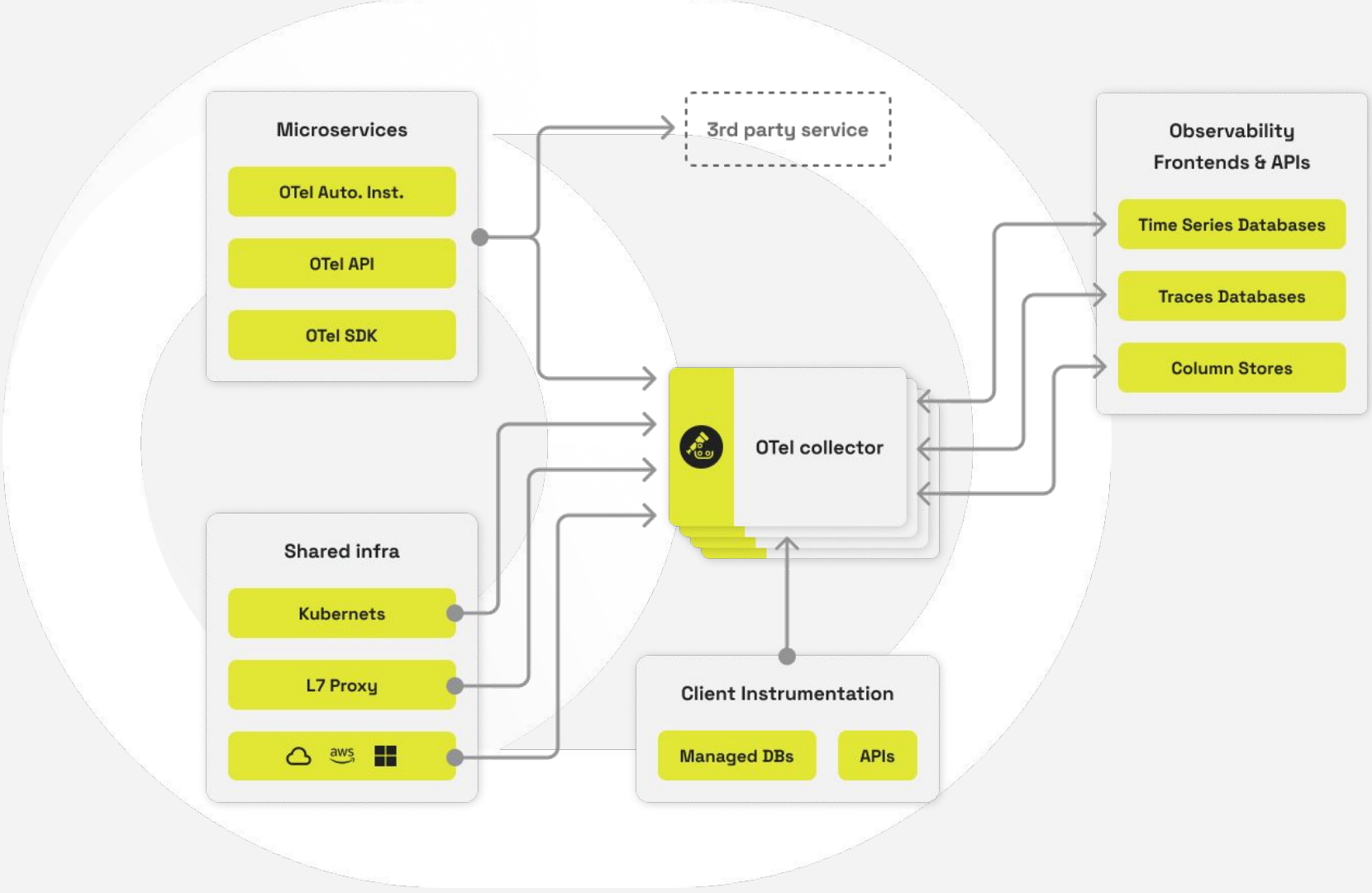
April 5, 2025 at 9:11 PM

TelemetriaAberta (TelmA)

“

O OpenTelemetry é uma coleção de APIs, SDKs e ferramentas. Use-o para instrumentar, gerar, coletar e exportar dados de telemetria (métricas, logs e rastros) para ajudar você a analisar o desempenho e o comportamento do seu software.

Fonte: https://opentelemetry.io/pt


Microservices
OTel Auto. Inst.
OTel API
OTel SDK
3rd party service
Observability
Frontends & APIs
Time Series Databases
Traces Databases
Column Stores
OTel collector
Shared infra
Kubernets
L7 Proxy
aws
Client Instrumentation
Managed DBs
APIs

APIs

- O que nós, programadores, usamos para instrumentar nosso software
- Cada linguagem tem sua API
- Todas seguem a especificação para APIs
- Algumas fornecem pacotes de instrumentação “contrib”
    - nós instrumentamos algumas coisas para que você não precise fazer tudo do zero!

SDKs

- Define o comportamento da API
- Cada linguagem tem uma SDK
- Todas seguem a especificação para SDKs

Especificações

- OTLP
- Convenções semânticas
    - como você deve instrumentar (nomes, propriedades, ...)

Ferramentas

- Collector
- Operator
- Ferramentas de instrumentação
    - em tempo de execução (Java Agent, eBPF, ...)
    - em tempo de compilação (Go Compile-Time instrumentation)

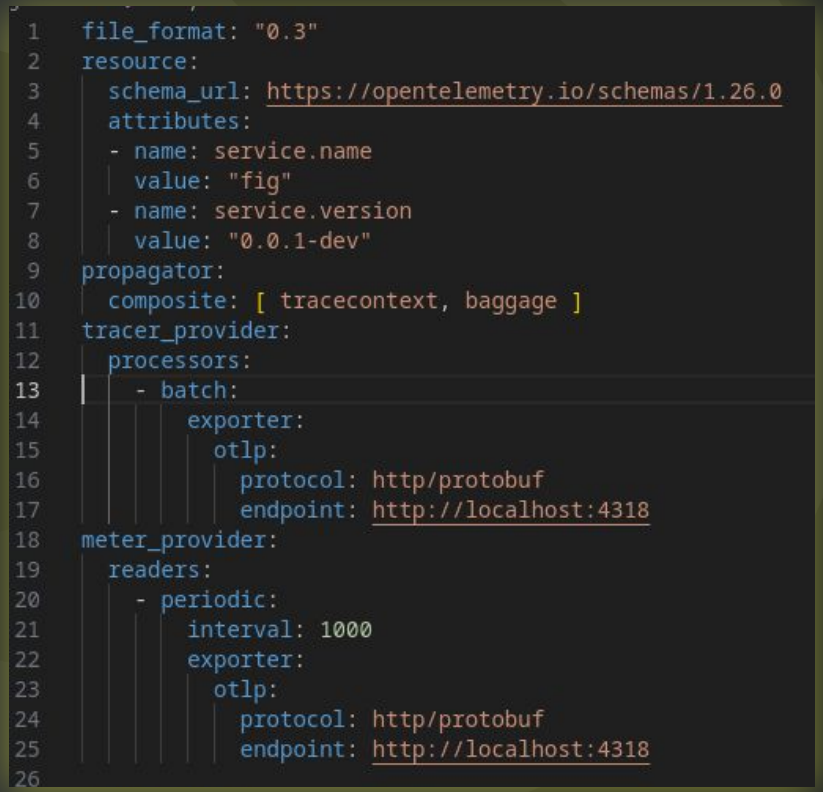
Instrumentação

Instrumentação

- Inicialização da SDK
- Instrumentação das partes interessantes
- ???
- Profit!

Demonstração!

Arquivo de configuração



```yaml
file_format: "0.3"
resource:
  schema_url: https://opentelemetry.io/schemas/1.26.0
  attributes:
  - name: service.name
    value: "fig"
  - name: service.version
    value: "0.0.1-dev"
propagator:
  composite: [ tracecontext, baggage ]
tracer_provider:
  processors:
    - batch:
        exporter:
          otlp:
            protocol: http/protobuf
            endpoint: http://localhost:4318
meter_provider:
  readers:
    - periodic:
        interval: 1000
        exporter:
          otlp:
            protocol: http/protobuf
            endpoint: http://localhost:4318
```

Configurando a SDK

```go
func Setup(ctx context.Context, cfgFile string) (func(context.Context) error, error) {
    b, err := os.ReadFile(cfgFile)
    if err != nil {
        if errors.Is(err, os.ErrNotExist) {
            logger = zap.Must(zap.NewProduction())
            logger.Info("otel config file not found")
            return func(ctx context.Context) error { return nil }, nil
        }

        return nil, err
    }
    // parse the config
    conf, err := otelconf.ParseYAML(b)
    if err != nil {
        return nil, err
    }
    sdk, err := otelconf.NewSDK(otelconf.WithContext(ctx), otelconf.WithOpenTelemetryConfiguration(*conf))
    if err != nil {
        return nil, err
    }
    otel.SetTracerProvider(sdk.TracerProvider())
    otel.SetMeterProvider(sdk.MeterProvider())
    global.SetLoggerProvider(sdk.LoggerProvider())

    // needs to be manually set due to https://github.com/open-telemetry/opentelemetry-go-contrib/issues/6712
    otel.SetTextMapPropagator(propagation.NewCompositeTextMapPropagator(propagation.TraceContext{}, propagation.Baggage{}))
```

Novo trecho (span)

```
ctx, span := otel.Tracer("fig").Start(ctx, "fig.event.new_service.onMessage")
defer span.End()
```

Erros

```
if err := trellis.OnRawTelemetry(ctx, msg, sd.onTraces); err != nil {
    span.SetStatus(codes.Error, err.Error())
    span.RecordError(err)
}
```

Atributos e eventos

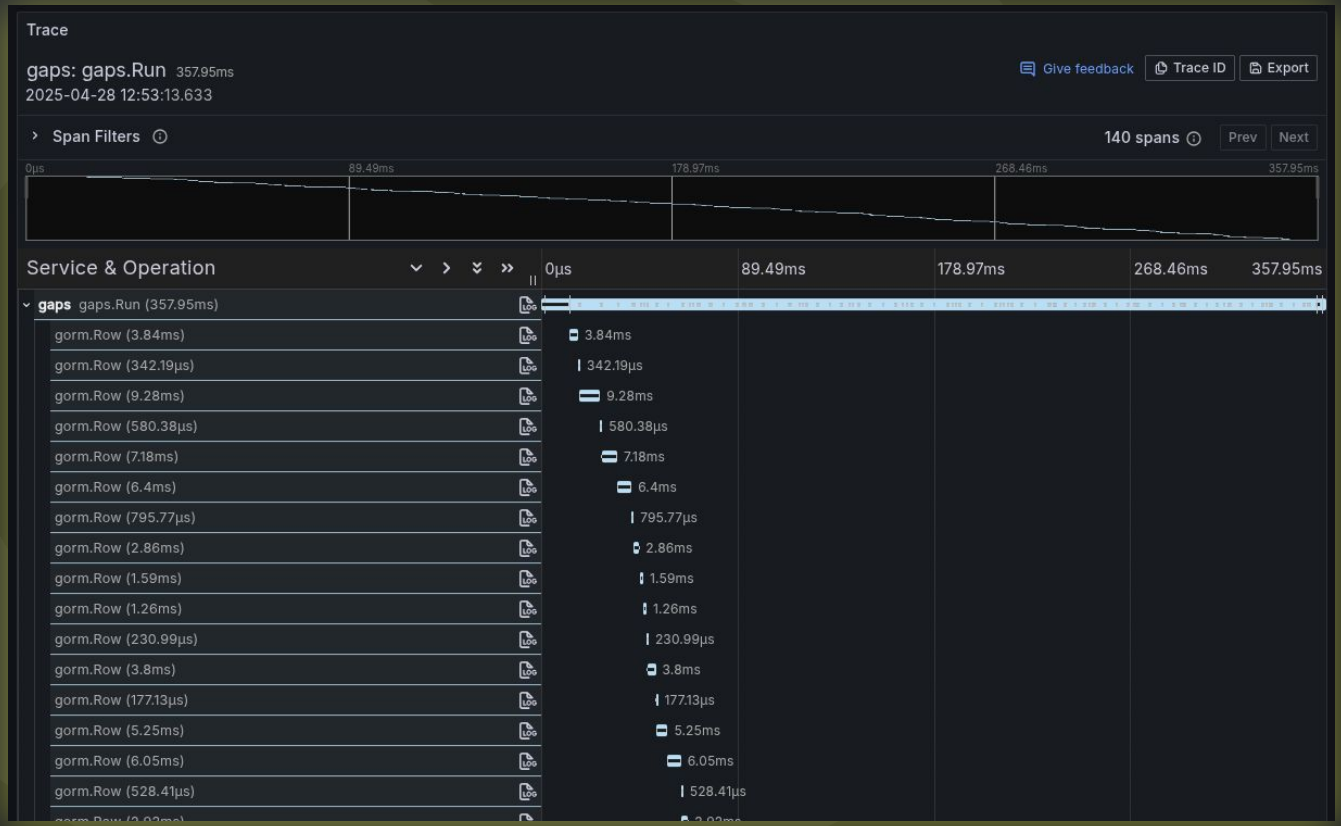
```
span.SetAttributes(attribute.Int("accepted_resource_records", accepted))

if accepted == 0 {
    span.AddEvent("no resource records left after filtering")
    return nil
}
```

Resultado esperado

Resultado


Trace
raincatcher: /v1/traces 48.32s
2025-04-28 12:55:53.556 POST 200 /v1/traces
Give feedback
Trace ID
Export
Span Filters
12 spans
Prev
Next
0µs
12.08s
24.16s
36.24s
48.32s
Service & Operation
raincatcher /v1/traces (3.5ms) 3.5ms
auth.Handler (3.45ms) 3.45ms
HandleTraces (3.42ms) 3.42ms
natsingester.EnqueueRequest (918.7µs) 918.7µs
natsingester.marshal (196.27µs) 196.27µs
natsingester.publish (677.46µs) 677.46µs
fig fig.event.new_service.onMessage (426.76µs) 426.76µs
overstory overstory.onTraceFunc (2.5ms) 2.5ms
gorm.Create (2.4ms) 2.4ms
gaps gaps.onTraceFunc (6.82ms) 6.82ms
gorm.Create (3.94ms) 3.94ms
gorm.Create (2.38ms) 2.38ms

Resultado


Trace
gaps: gaps.Run 357.95ms
2025-04-28 12:53:13.633
Give feedback
Trace ID
Export
Span Filters
140 spans
Prev
Next
0µs
89.49ms
178.97ms
268.46ms
357.95ms
Service & Operation
gaps gaps.Run (357.95ms)
gorm.Row (3.84ms)
gorm.Row (342.19µs)
gorm.Row (9.28ms)
gorm.Row (580.38µs)
gorm.Row (7.18ms)
gorm.Row (6.4ms)
gorm.Row (795.77µs)
gorm.Row (2.86ms)
gorm.Row (1.59ms)
gorm.Row (1.26ms)
gorm.Row (230.99µs)
gorm.Row (3.8ms)
gorm.Row (177.13µs)
gorm.Row (5.25ms)
gorm.Row (6.05ms)
gorm.Row (528.41µs)
3.84ms
342.19µs
9.28ms
580.38µs
7.18ms
6.4ms
795.77µs
2.86ms
1.59ms
1.26ms
230.99µs
3.8ms
177.13µs
5.25ms
6.05ms
528.41µs

Fases da instrumentação de métricas

Inicializando um instrumento (métrica)

```
func NewInsight() *Insight {
    countOrphans, err := otel.Meter("gaps").Int64Counter(
        "gaps.orphan_records",
        metric.WithDescription("Number of orphaned records."),
        metric.WithUnit("{records}"),
    )
```

Fazendo medições

```
i.countOrphans.Add(ctx, int64(len(orphans)))
```

Bibliotecas instrumentadas

Instrumentando chamadas HTTP

```
otelHandler := otelhttp.NewHandler(authHandler, "/v1/traces")
mux.Handle("/v1/traces", otelHandler)
```

Instrumentando chamadas gRPC

```
grpcSrv := grpc.NewServer(grpc.StatsHandler(otelgrpc.NewServerHandler()))
```

Instrumentação em tempo de execução (Agentes)

Instrumentação - Agentes

- Não existe :-)

Instrumentação - Agentes

- Não existe :-)
- (estamos na GopherCon, certo?)

Instrumentação - Agentes

- Java, Python, Ruby, JS, ...


Instrumentação
em tempo de
execução
(eBPF)

Instrumentação - Tempo de Execução

- OTel Go Auto-Instrumentation
  - Foco em rastreamento distribuído, não funciona em kernel mais novos
- Grafana Beyla, sendo doado para o OTel
  - Foco em métricas “RED”
  - Suporte inicial a rastreamento distribuído

Instrumentação em tempo de compilação

Instrumentação - Tempo de Compilação

- Estágio super inicial
- `otel-compile-time-instr go build .`
- Baseado em dois projetos: DataDog Orchestrion, Alibaba ‘otel’

Boas práticas

Boas práticas

- O que eu posso precisar saber às 02:00?
- Pense na telemetria que você precisa
- Pense nos atributos que você precisa
- Consulte as convenções semânticas
- Construa suas próprias convenções semânticas

Veredito

Veredito

- Instrumentação proposital
- Na falta de telemetria, use auto-instrumentação
  - Bibliotecas instrumentadas
  - Agentes (Java, Node, ...)
  - via eBPF

Perguntas e respostas

Obrigado!

Vamos manter contato!

linkedin.com/in/jpkroehling

Olly GARDEN