

---

# Getting Started with OpenTelemetry

A Gateway to Modern Observability



---

# Nice to meet you!



Juraci Paixão Kröhling  
Software Engineer

- Software Engineer @ OllyGarden
- OTel Governance Committee
- CNCF Ambassador
- Organizer OTel Night Berlin
- Emeritus: OTel Collector, Jaeger, ...
- Dose de Telemetria

---

# Agenda

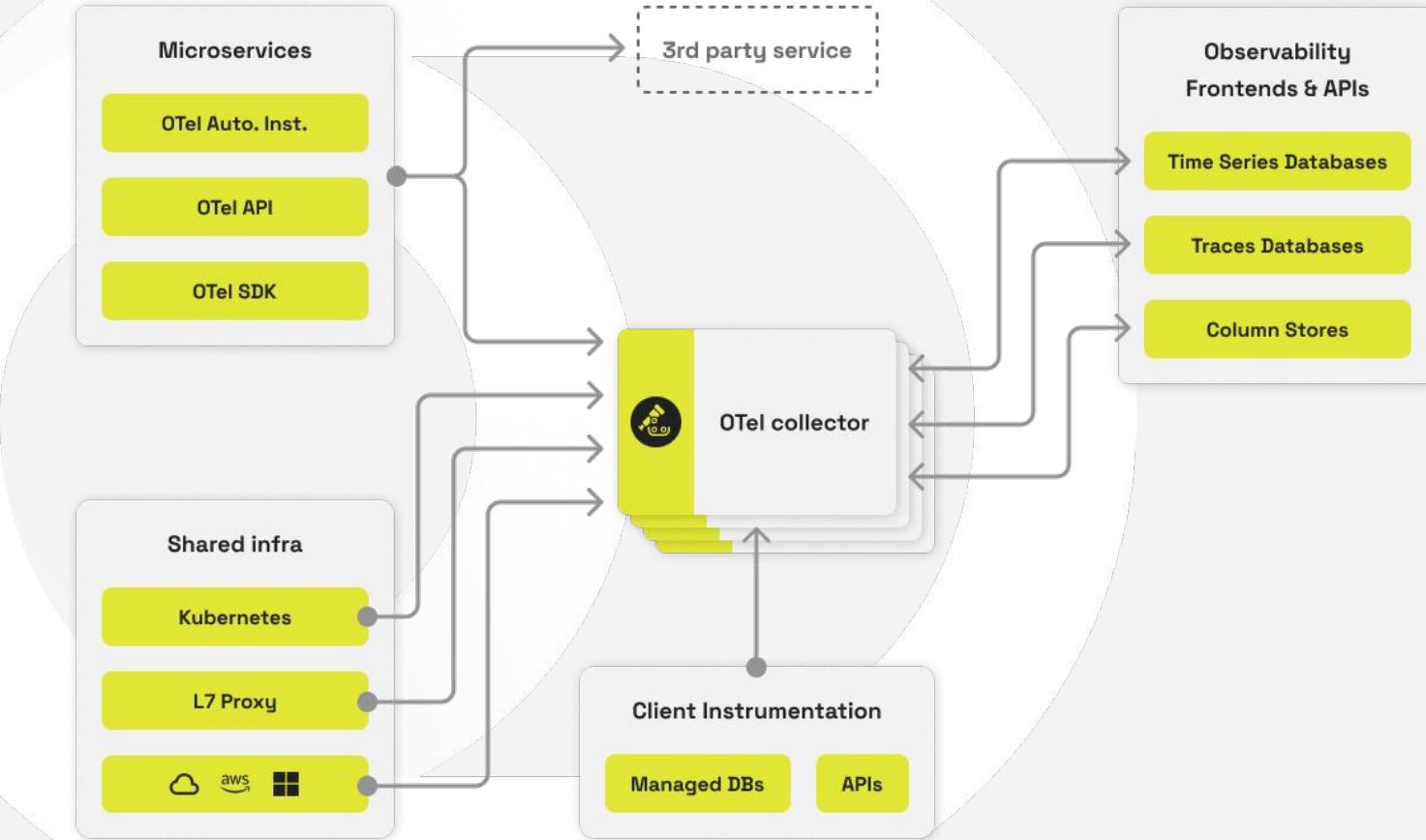
- A Simple Service
- Adding the SDK
- Adding Traces
- Adding Metrics
- Collector Pipeline
- Best Practices
- Q&A Session

---

“

OpenTelemetry is a collection of APIs, SDKs, and tools. Use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyze your software's performance and behavior.

Source: <https://opentelemetry.io>



# Planting the seed

```
package main
|
import (
    "encoding/json"
    "fmt"
    "log"
    "net/http"
    "time"
)
|
type Payment struct {
    ID      string `json:"id"`
    Amount  float64 `json:"amount"`
    Status  string `json:"status"`
    Date    string `json:"date"`
}
|
var payments []Payment
|
func main() {
    http.HandleFunc("/api/payment", paymentHandler)

    fmt.Println("Server starting on :8080")
    log.Fatal(http.ListenAndServe(":8080", nil))
}
|
func paymentHandler(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")

    switch r.Method {
    case http.MethodGet:
        handleGetPayments(w, r)
    case http.MethodPost:
        handleCreatePayment(w, r)
    default:
        w.WriteHeader(http.StatusMethodNotAllowed)
        json.NewEncoder(w).Encode(map[string]string{"error": "Method not allowed"})
    }
}
```



## Recap

- We configured the SDK using an external configuration file
- We tied our logging framework with OTel
- We added a span, telling us more information about the incoming HTTP request

# First Sprouts - adding spans

```
func paymentHandler(w http.ResponseWriter, r *http.Request) {
    _, span := telemetry.Tracer().Start(r.Context(), "paymentHandler")
    defer span.End()

    // ...
}
```

# Growing roots - adding metrics

```
type Metrics struct {
    requestCounter      metric.Int64Counter
    responseDuration    metric.Float64Histogram
    errorCounter        metric.Int64Counter
    paymentAmount       metric.Float64Histogram
    paymentsByStatus   metric.Int64Counter
    paymentsByCurrency metric.Int64Counter
}
```

# Growing roots - adding metrics

```
203     paymentsByCurrency, err := meter.Int64Counter(
204         "payments_by_currency_total",
205         metric.WithDescription("Total number of payments by currency"),
206         metric.WithUnit("1"),
207     )
208     if err != nil {
209         return err
210     }
211 }
```

# Growing roots - adding metrics

```
metrics.paymentsByCurrency.Add(r.Context(), 1, metric.WithAttributes(  
    attribute.String("currency", payment.Currency),  
))
```

return StatusCode(HttpStatusCode.StatusCreated)



## Recap

- We declared, initialized, and used (business) metrics in our code

# Full Bloom - OTel Collector

```
! otelcol.yaml
1   extensions:
2     receivers:
3       otlp:
4         protocols:
5           http:
6             endpoint: "localhost:4318"
7
8     processors:
9       transform:
10         error_mode: ignore
11         trace_statements:
12           - set(resource.attributes["pipeline.name"], "local-collector")
13
14     exporters:
15       otlphttp:
16         endpoint: "http://localhost:5318"
17
18     service:
19       extensions: [ ]
20       pipelines:
21         traces:
22           receivers: [ otlp ]
23           processors: [ transform ]
24           exporters: [ otlphttp ]
```



## Recap

- We created an OTel Collector configuration file
- We configured a pipeline that adds a resource attribute to all spans
- We configured the Collector to send data to our backend



## Tending the garden

- Semantic conventions
- Instrumentation via libraries
- Purposeful instrumentation
  - Manual? Libraries? Zero-code?

---

“

Purposeful instrumentation is the  
difference between telemetry that  
exists and telemetry that's  
**useful.**

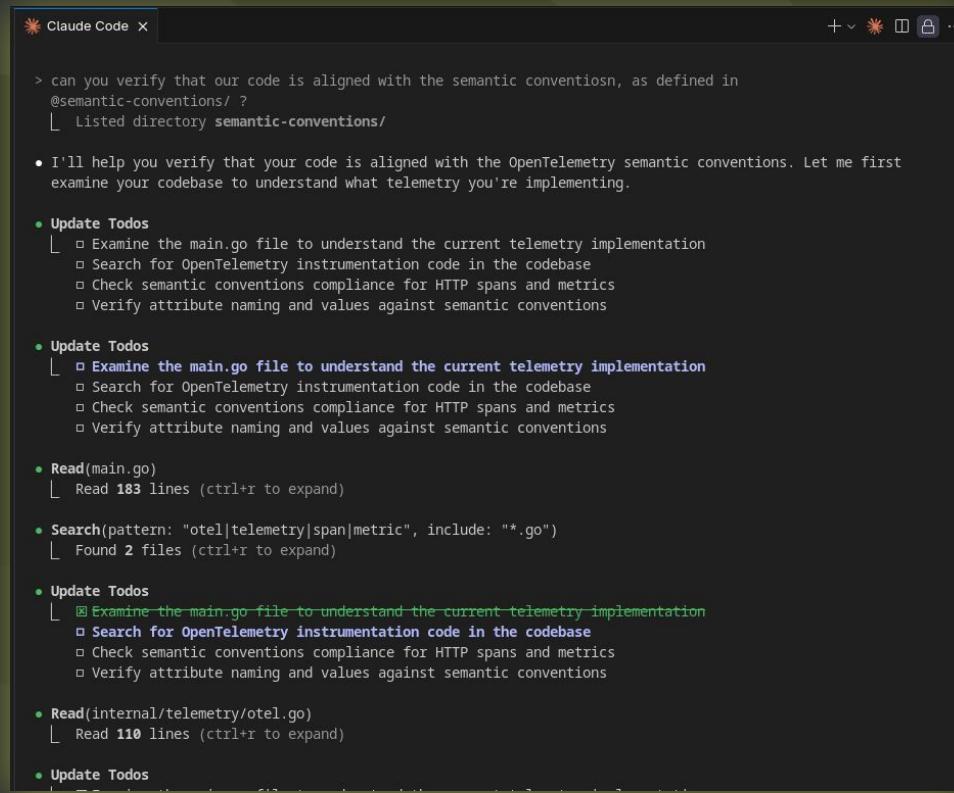
---

## Tending the garden

- Hint: Instrumentation Score
  - <https://instrumentation-score.com/>



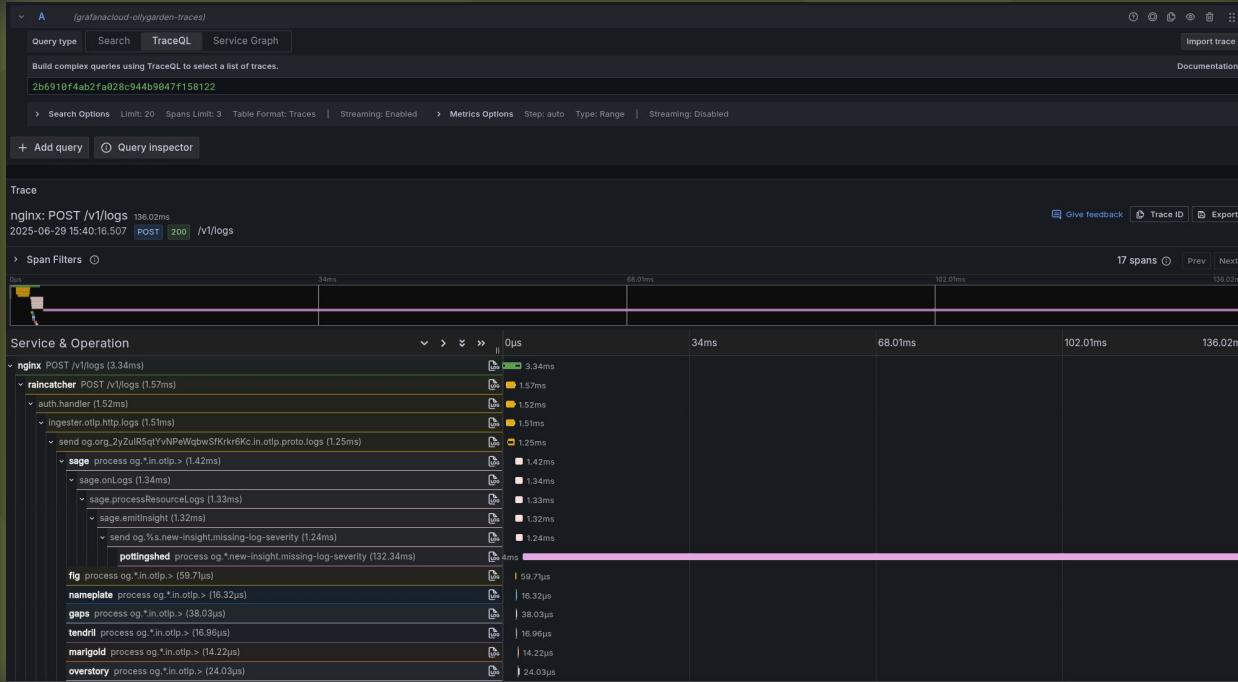
# Well-tended garden (final pro-tip!)



The screenshot shows a terminal window titled "Claude Code" with a dark theme. The window contains a list of tasks and sub-tasks:

- > can you verify that our code is aligned with the semantic conventions, as defined in @semantic-conventions/ ?
  - >Listed directory `semantic-conventions`
- I'll help you verify that your code is aligned with the OpenTelemetry semantic conventions. Let me first examine your codebase to understand what telemetry you're implementing.
- Update Todos
  - Examine the `main.go` file to understand the current telemetry implementation
    - Search for OpenTelemetry instrumentation code in the codebase
    - Check semantic conventions compliance for HTTP spans and metrics
    - Verify attribute naming and values against semantic conventions
  - Examine the `main.go` file to understand the current telemetry implementation
    - Search for OpenTelemetry instrumentation code in the codebase
    - Check semantic conventions compliance for HTTP spans and metrics
    - Verify attribute naming and values against semantic conventions
- Read(`main.go`)
  - Read 183 lines (ctrl+r to expand)
- Search(pattern: "otel|telemetry|span|metric", include: "\*.go")
  - Found 2 files (ctrl+r to expand)
- Update Todos
  - Examine the `main.go` file to understand the current telemetry implementation
    - Search for OpenTelemetry instrumentation code in the codebase
    - Check semantic conventions compliance for HTTP spans and metrics
    - Verify attribute naming and values against semantic conventions
- Read(internal/telemetry/otel.go)
  - Read 110 lines (ctrl+r to expand)
- Update Todos

# Well-tended garden





# Takeaways

- OTel is an observability framework
  - APIs, SDKs, tools (Collector, Operator, ...)
  - Semantic conventions, OTLP, ...
- Pick and choose what you need!
- Apply purposeful instrumentation on your daily routine

---

# Obrigado!



Let's keep in touch!

[linkedin.com/in/jpkroehling](https://linkedin.com/in/jpkroehling)