

Três Sinais e um Destino para o Sucesso da Observabilidade

Quem sou eu?

Nome: Daniel Nascimento

Cargo: Engenheiro de Plataforma

Experiência: DevOps, SRE, Plataformas (playing)

Vida: Pai, Filho, Amigo, Corredor amador



O que vamos ver

Os 3 pilares da observabilidade

OpenTelemetry como padrão universal

Problema da observabilidade moderna

Desafios:

- Falta de visibilidade entre serviços
 - Internos
 - Externos
 - Banco de dados
 - Sistemas de filas
- Ferramentas desconectadas
- Correlação difícil (cada um implementa de uma maneira)
- Vendor lock-in (problema bem comum)

Os Três Sinais da Observabilidade

O que é Otel

É um mecanismo padrão (standard) para geração de dados de telemetria de maneira organizada e estruturado.

Três pilares: logs, métricas e traces

Suporta outras coisas... como definição de profile, syntactic data, para coletar dados de telemetria dos ambientes, clientes e

Vendor agnostic - enviar os dados para onde quiser

Open telemetry entende tudo como sinais: logs, métricas e traces são sinais. Correlaciona os sinais de maneira contextualizada.

Os Três Sinais da Observabilidade



LOGS: Eventos discretos no tempo - Debug, auditoria, níveis de informação



MÉTRICAS: Valores agregados - Dashboards, alertas



TRACES: Jornada da requisição - Performance e debugging

OpenTelemetry: o padrão universal

SDK único e vendor-neutral

Suporte a logs, métricas e traces

 Vantagens:

Sem vendor lock-in

Padronizado

Comunidade ativa

Instrumentação Automática

- ✗ Antes (manual): Muito código boilerplate
 - ✓ Depois (auto): SDK @opentelemetry/sdk-node
- Código mais limpo e com menos fricção

Instrumentação Manual

 Quando usar:

- Atributos de negócio
- Child spans
- Propagação de contexto

Configuração de Métricas



Contador de requisições: `createCounter()`



Latência: `createHistogram()`



Gauge de conexões: `createObservableGauge()`

Métricas coletadas via middleware

Demo Time!

1. Executar Workload (docker-compose up)
2. Gerar chamadas (curl para gerar tráfego)
3. Visualizar no Datadog, Jaeger/Prometheus
4. Simular error
5. Correlacionar logs
6. Investigar
7. Repetir os passos 3 a 6

Análise no Jaeger

✓ Informações capturadas:

- Latência detalhada
- Dependências entre serviços
- Stack trace e atributos de contexto
- Correlação de logs via trace ID

Configuração de Produção



Docker Compose



OpenTelemetry Collector



Exportadores para Jaeger e Prometheus

Volumes, ports e env vars configurados

Boas Práticas



Faça:

- Auto-instrumentação
- Atributos relevantes
- Sampling adequado
- Naming padronizado



Evite:

- Tracear demais
- Dados sensíveis
- Traces órfãos

Métricas-chave para monitorar



Golden Signals (SRE):

- Latência
- Erros
- Tráfego
- Saturação



RED Method:

- Requests
- Errors
- Duration

Troubleshooting Distribuído

1. Identificar erro via métricas
2. Buscar no Datadog, Grafana Loki, Jaeger, ou outra ferramenta
3. Analisar trace
4. Corrigir causa-raiz
5. Correlacionar com logs estruturados

ROI da Observabilidade



Melhorias quantificadas:

- MTTR (Tempo Médio para Reparo): 4h → 30min
- MTBF (Tempo Médio Entre Falhas): 2 dias → 2 semanas
- Deploy Confidence: 60% → 95%
- Debug Time: 2h → 15min

Roadmap de Implementação



Fase 1: Fundação



Fase 2: Expansão



Fase 3: Otimização



Fase 4: Maturidade

Q&A

? Perguntas Frequentes:


- Overhead? <5%
- É possível implementar no Kubernetes? Sim
- Compatível com ferramentas existentes
- LGPD? Sim, via configuração

Obrigado!

 OpenTelemetry é o presente / futuro da observabilidade

 Três sinais trabalham melhor juntos

 Implementação gradual é a chave do sucesso

 ROI visível em semanas (time / empresa)

A observabilidade é sobre cultura!

 [danielcn](mailto:deo.daniel@gmail.com) | deo.daniel@gmail.com | [Daniel Nascimento](#)