OpenTelemetry v .NET

Logy, metriky a tracing bez kompromisů

Tomáš Jecha

Head of Engineering at cbData

LinkedIn /in/jechtom | X @jechtom

Why observability sucks?

- Vendor lock-in
  - Proprietary tools and instrumentation libraries
  - Switching is expensive and painful
  - Every tool requires custom SDKs and configuration
- Tools fragmentation
  - Different tools, formats, protocols → no standardization
- Lack of clear best practices
  - No clear guidelines → Every team does observability differently
  - Inconsistent telemetry data structure

What is OpenTelemetry?

- A unified, vendor-neutral observability framework
- Semantic rules – naming (log severity, etc.)
- Protocol – serialization and transport
- APIs & SDKs – C++, .NET, Go, Java, PHP, Python, Rust, Swift, …
- Ecosystem of libraries and tools – instrumentation, exporters, etc.
- OpenTelemetry Collector – receive, process and export telemetry data
- Massive industry support – https://opentelemetry.io/ecosystem/vendors/

OpenTelemetry Protocol (OTLP)

- OTLP/gRPC (Protobuf) or OTLP/HTTP (Protobuf or JSON)
- Protocol specs and protobuf definitions at https://github.com/open-telemetry/opentelemetry-proto
- Defines services:
    - Logs collector
    - Metrics collector
    - Trace collector

OpenTelemetry Protocol (OTLP) Design Goals

- All signal types over single protocol
- For instrumented apps, telemetry backends and proxies
- Reliable, low CPU and memory usage
- High throughput, backpressure signalling
- Load-balancer friendly

.NET Aspire Dashboard

- Part of the .NET Aspire project
- Receives OpenTelemetry data via the OTLP protocol
- Has a standalone mode

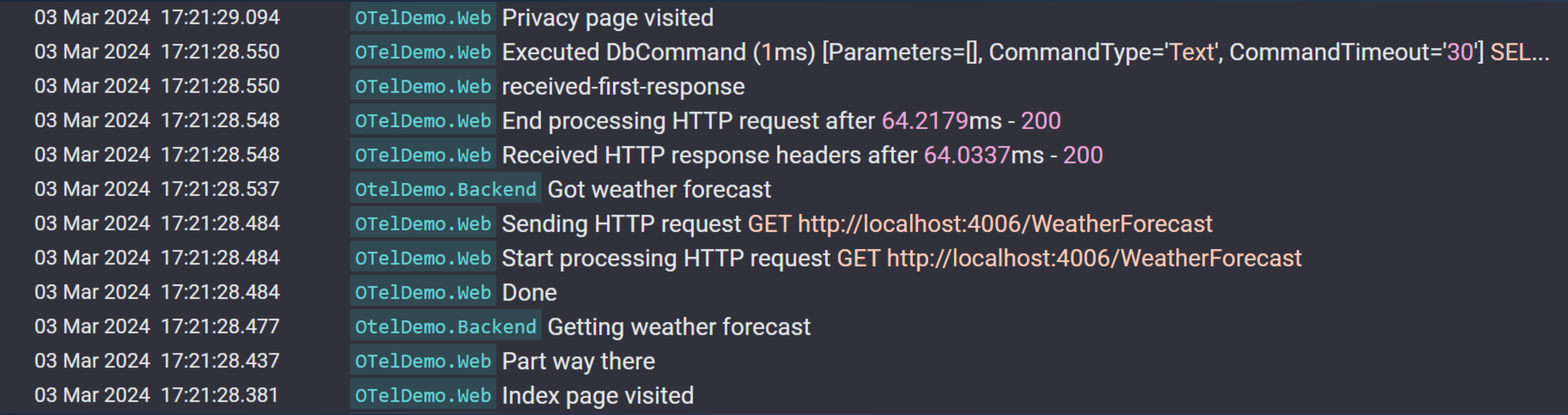
DEMO

.NET Aspire Dashboard
+ Jaeger, Zipkin, Prometheus, SEQ, …

OpenTelemetry Signals

- Logs
- Tracing
- Metrics

Observability Signals – Logs



```
03 Mar 2024  17:21:29.094   OTelDemo.Web Privacy page visited
03 Mar 2024  17:21:28.550   OTelDemo.Web Executed DbCommand (1ms) [Parameters=[], CommandType='Text', CommandTimeout='30'] SEL...
03 Mar 2024  17:21:28.550   OTelDemo.Web received-first-response
03 Mar 2024  17:21:28.548   OTelDemo.Web End processing HTTP request after 64.2179ms - 200
03 Mar 2024  17:21:28.548   OTelDemo.Web Received HTTP response headers after 64.0337ms - 200
03 Mar 2024  17:21:28.537   OtelDemo.Backend Got weather forecast
03 Mar 2024  17:21:28.484   OTelDemo.Web Sending HTTP request GET http://localhost:4006/WeatherForecast
03 Mar 2024  17:21:28.484   OTelDemo.Web Start processing HTTP request GET http://localhost:4006/WeatherForecast
03 Mar 2024  17:21:28.484   OTelDemo.Web Done
03 Mar 2024  17:21:28.477   OtelDemo.Backend Getting weather forecast
03 Mar 2024  17:21:28.437   OTelDemo.Web Part way there
03 Mar 2024  17:21:28.381   OTelDemo.Web Index page visited
```

SEQ logging UI (proprietary)

OpenTelemetry Log Record

- Timespan → When?
- Resource attributes → Who?
- Tracing → Trace Id, Span Id
- Severity → Trace, Debug, Info, Warn, Error, Fatal
- Structured Content → *“Order 32 has been Delivered”*

```
Message = "Order {OrderId} has been {State}"
OrderId = "32"
State   = "Delivered"
```

- Additional Attributes → Scope, Client IP, Identity, …

Resource represents the **entity** producing **telemetry**.

- Windows IIS AppPool
- Process inside Kubernetes pod
- Linux daemon
- …

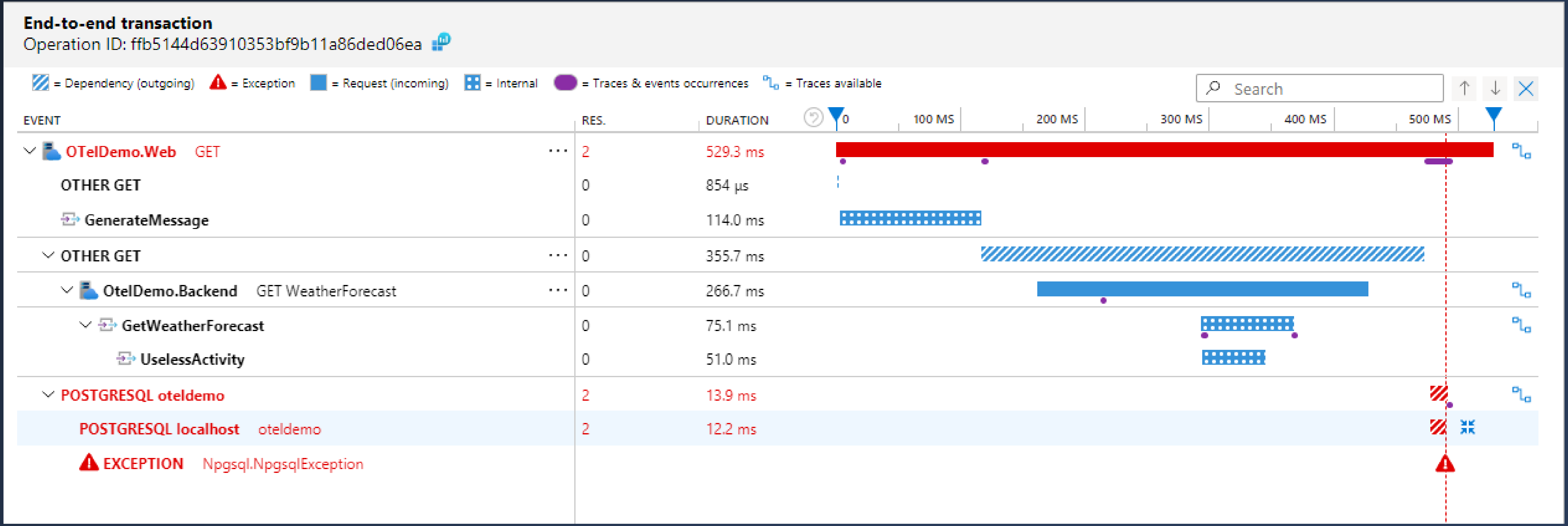
Observability signals like:
logs, metrics, tracing

https://opentelemetry.io/docs/specs/semconv/resource/

Resource Attributes – Examples

```
service.name=ShoppingCart
service.instance.id=627cc493-f310-47de-96bd-71410b7dec09
service.version=3.4.5; a01dbef8a
deployment.environment.name=staging
…
telemetry.sdk.language=dotnet
telemetry.sdk.name=opentelemetry
telemetry.sdk.version=1.2.3
…
process.pid=1234
process.executable.path=D:\apps\ShoppingCart\ShoppingCart.exe
os.type=windows
cloud.platform=azure_container_apps
k8s.pod.name=kubernetes-pod-name
…
```

Observability Signals – Tracing



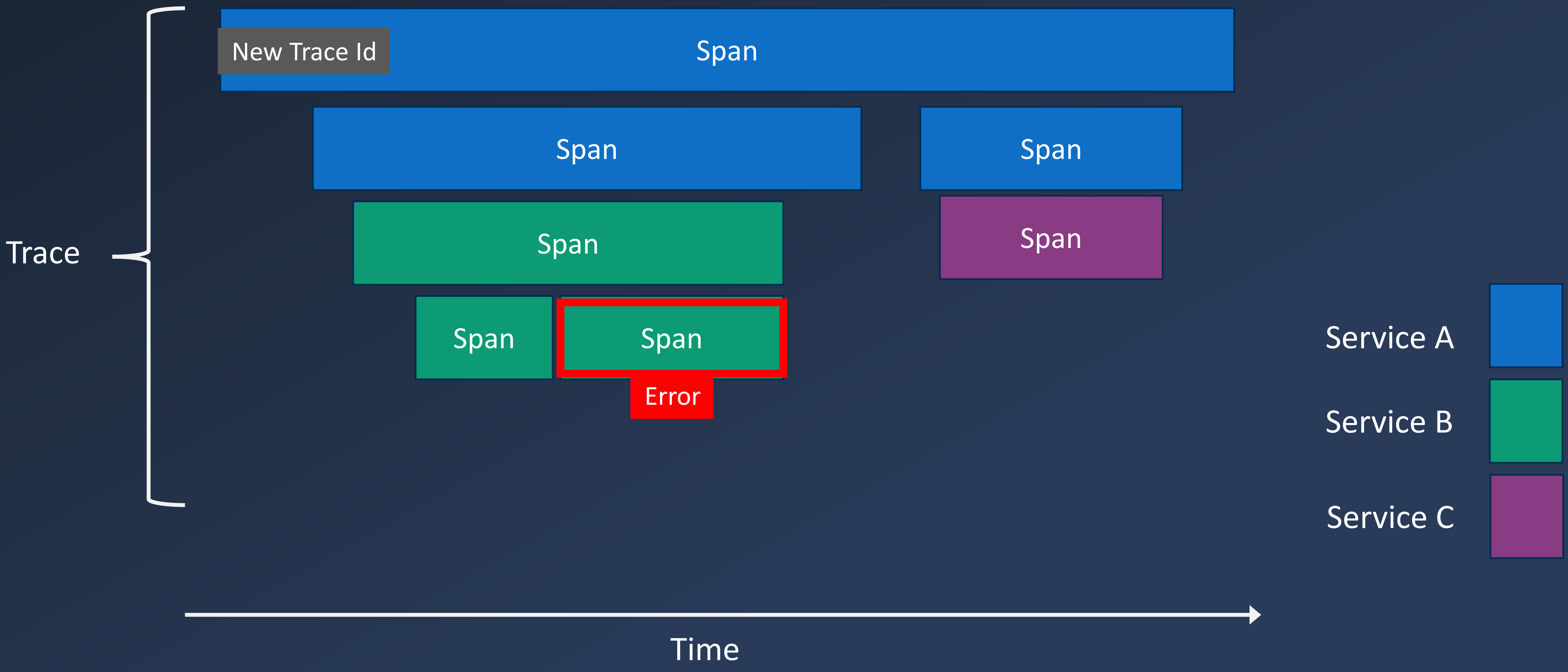
End-to-end transaction
Operation ID: ffb5144d63910353bf9b11a86ded06ea

▨ = Dependency (outgoing) ⚠ = Exception 🟦 = Request (incoming) ⊞ = Internal ● = Traces & events occurrences ⎘ = Traces available

EVENT	RES.	DURATION
OTelDemo.Web GET	2	529.3 ms
OTHER GET	0	854 µs
GenerateMessage	0	114.0 ms
OTHER GET	0	355.7 ms
OtelDemo.Backend GET WeatherForecast	0	266.7 ms
GetWeatherForecast	0	75.1 ms
UselessActivity	0	51.0 ms
POSTGRESQL oteldemo	2	13.9 ms
POSTGRESQL localhost oteldemo	2	12.2 ms
EXCEPTION Npgsql.NpgsqlException		

Azure Application Insights (proprietary)

Observability Signals – Tracing

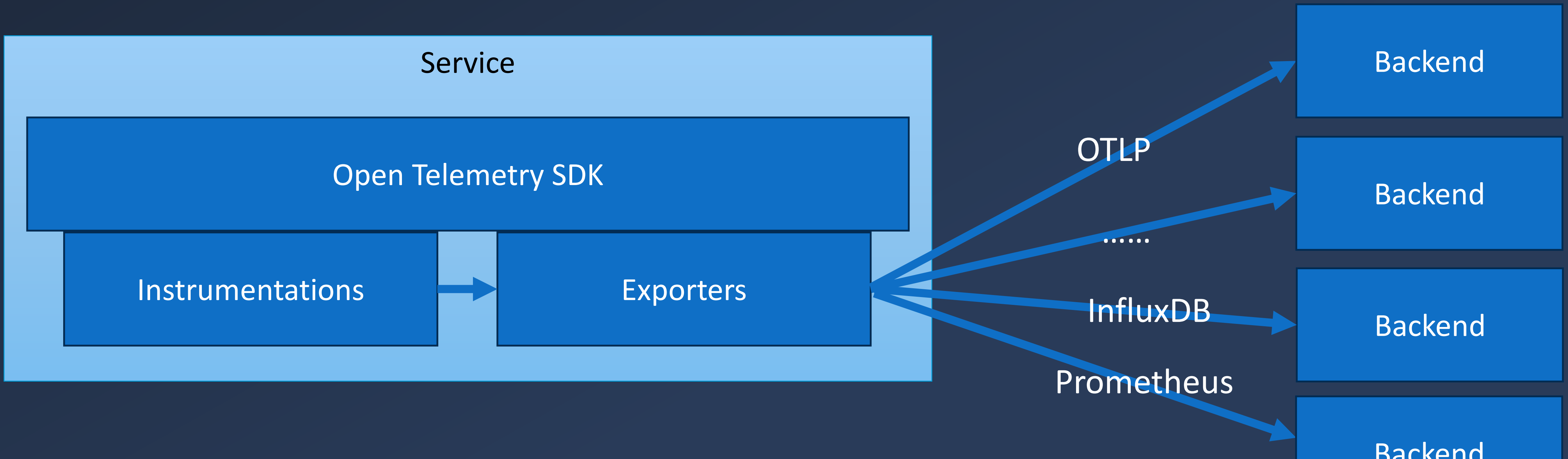


How tracing works?

- Span = .NET System.Diagnostics.Activity
- W3C Trace Context HTTP headers
- TraceId, SpanId, ParentSpanId
- .NET distributed tracing concepts: https://learn.microsoft.com/en-us/dotnet/core/diagnostics/distributed-tracing-concepts

OpenTelemetry Libraries

- Registry: https://opentelemetry.io/ecosystem/registry
- Instrumentation libraries – generates relevant telemetry data
- Exporter libraries – sends telemetry (via OTLP or other protocols)

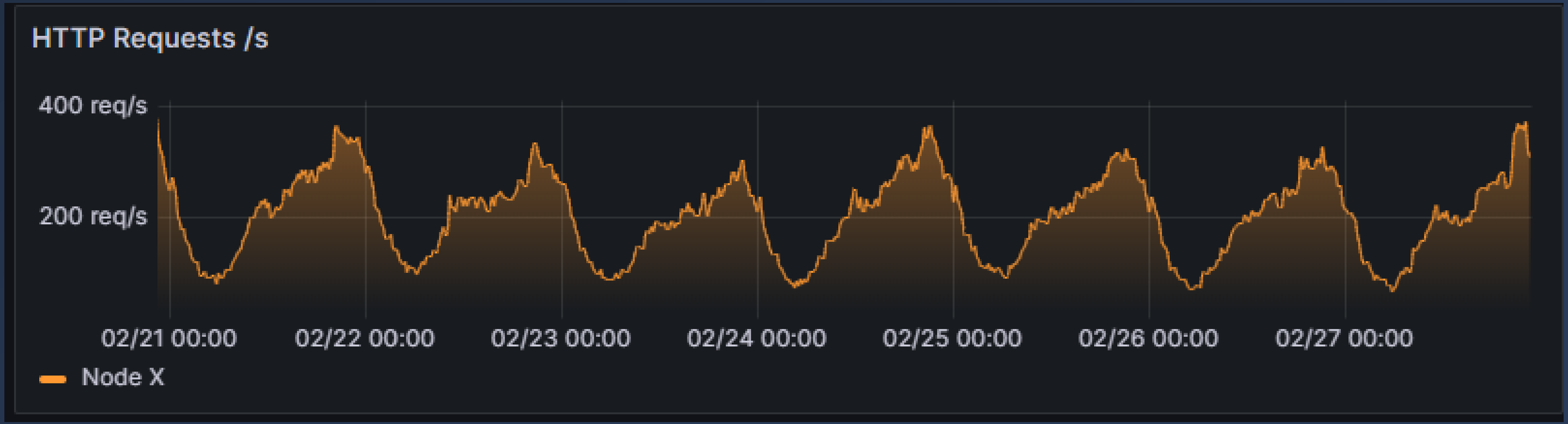


DEMO

OpenTelemetry .NET SDK

Observability Signals – Metrics

- .NET: System.Diagnostics.Metrics
  - .NET6+, designed to integrate well with OpenTelemetry
  - Replaces EventCounters (.NET Core 3+) and PerformanceCounters (Win only) – see https://learn.microsoft.com/en-us/dotnet/core/diagnostics/compare-metric-apis



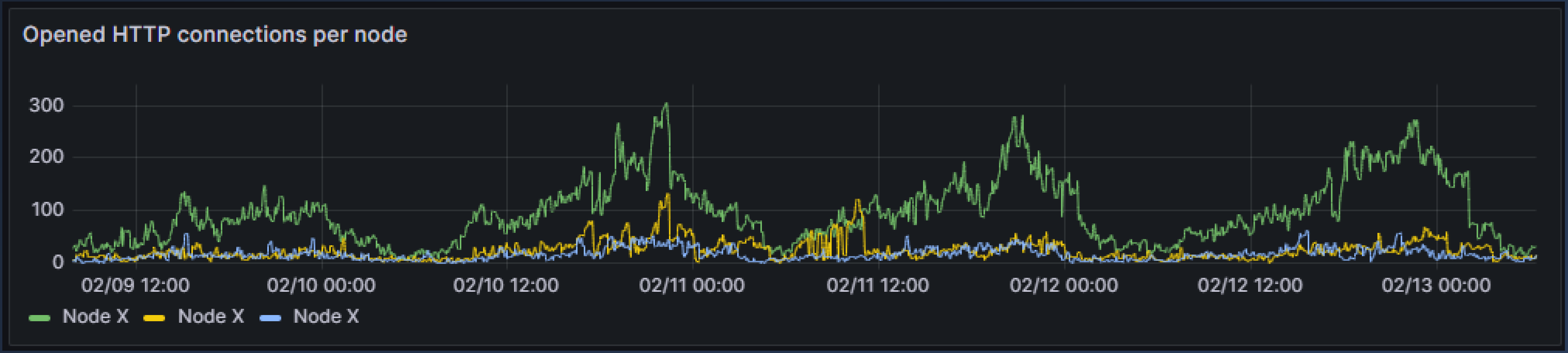
OpenTelemetry Metric Types

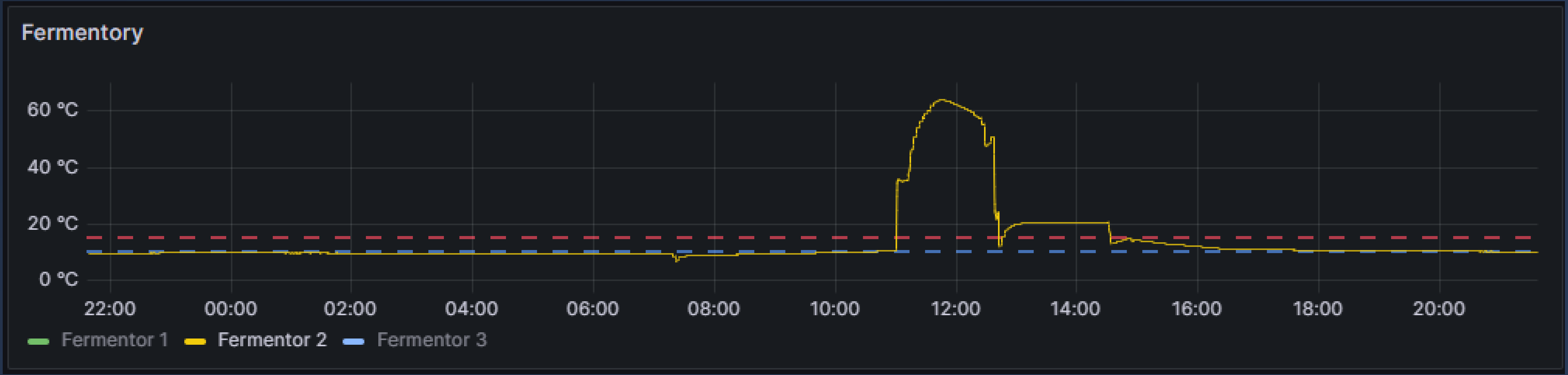
- Gauge
- Counter
- Histogram

Metric Types - Gauge

- Usage: Set instant value
- Examples:
  - CPU usage
  - Allocated threads
  - Open connections
  - Longest running task
  - Timespan of last backup
  - Free disk space
  - Queue length
  - ….

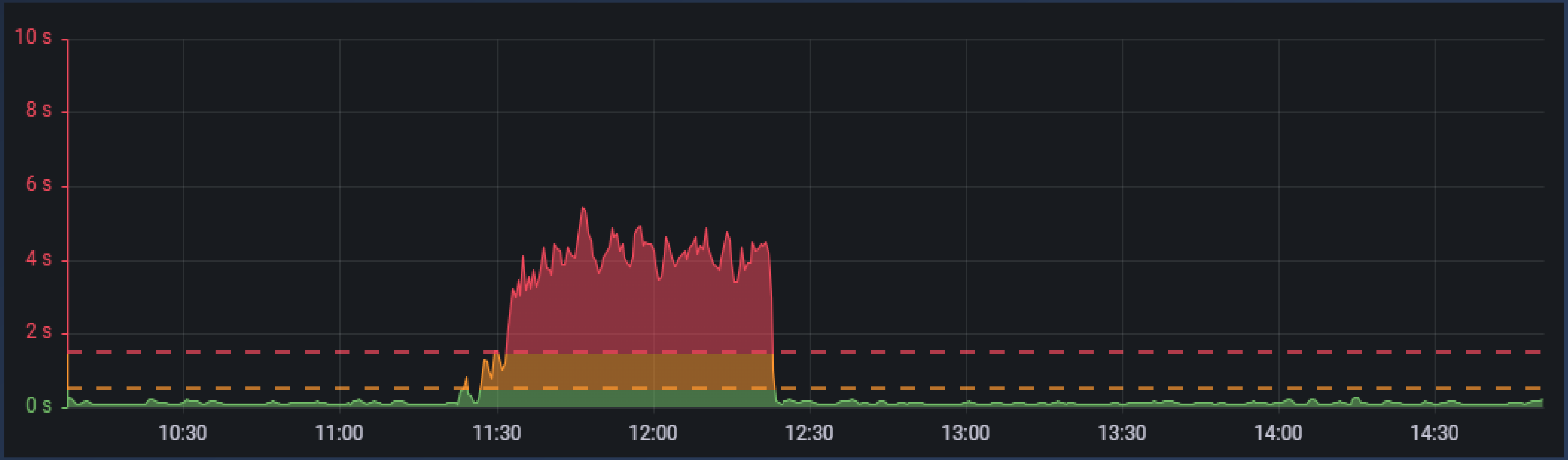
Metric Types - Gauge





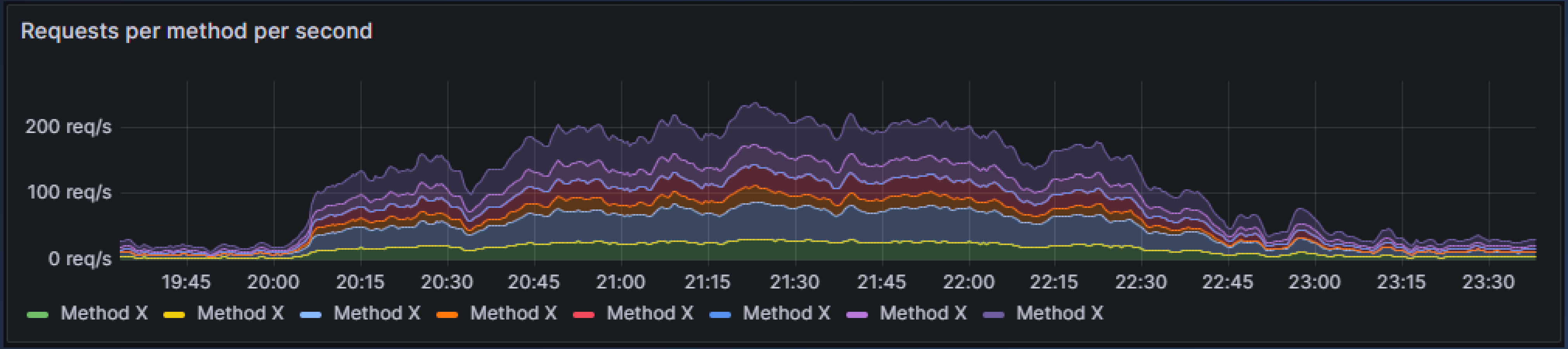
Metric Types – Counter (Sum)

- Usage: Increment +1
  - Examples: Counter of HTTP requests, executions, cache hit/miss, …
- Usage: Add +delta
  - Examples: Bytes transferred, rows processed, request duration, …

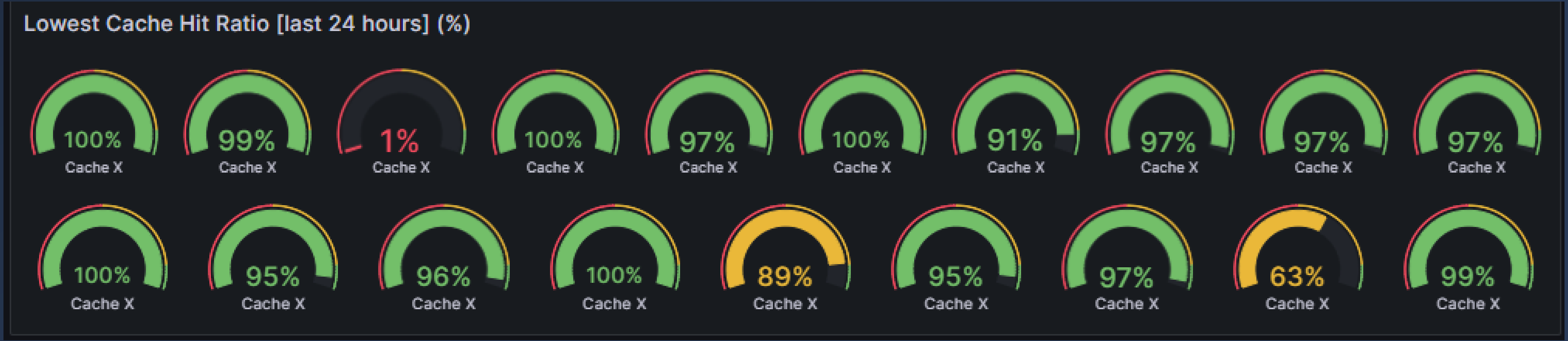


Metric Types – Counter (Sum)

- Visualized as rate
  - Examples: Requests/second, MB/s bandwidth, orders per hour, …

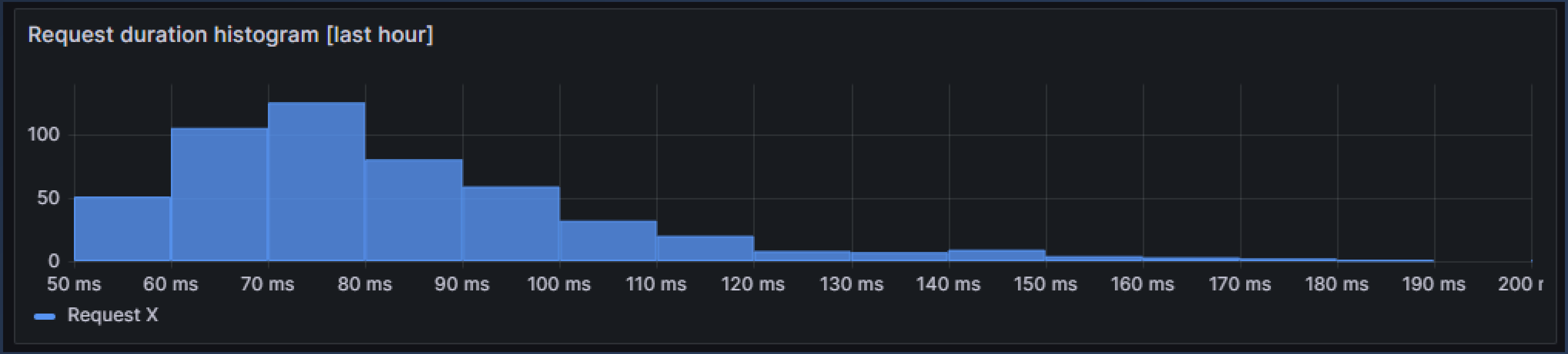


- or as ratio (multiple metrics)
  - Examples: cache hit/miss ratio, success/failure rate, …



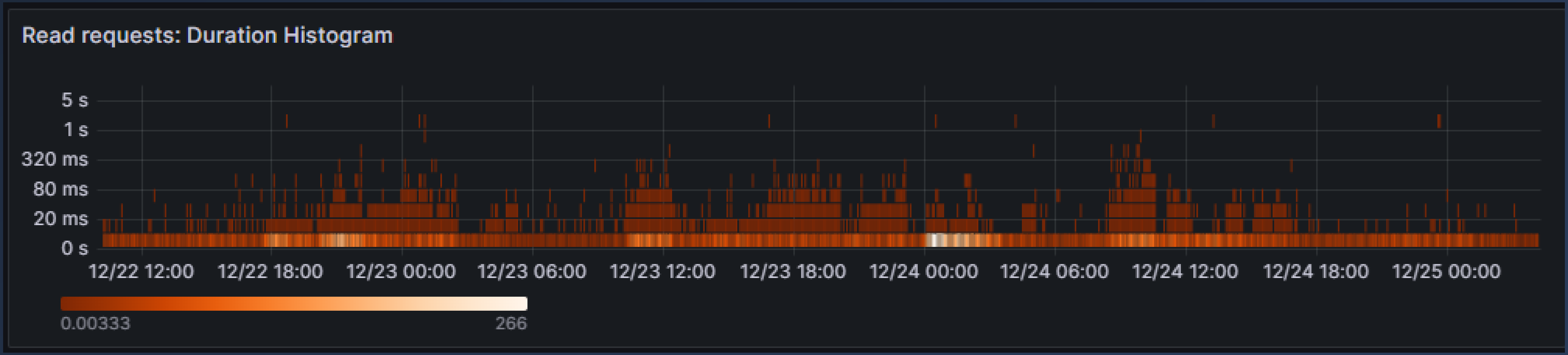
Metric Types – Histogram

- Usage: Record frequency of value (buckets)
- Examples: Request duration, message size, quantity per order
- Visualized as: Histogram, heatmap, percentile, average
- Default buckets for OpenTelemetry:
  [0, 5, 10, 25, 50, 75, 100, 250, 500, 750, 1000, 2500, 5000, 7500, 10000]



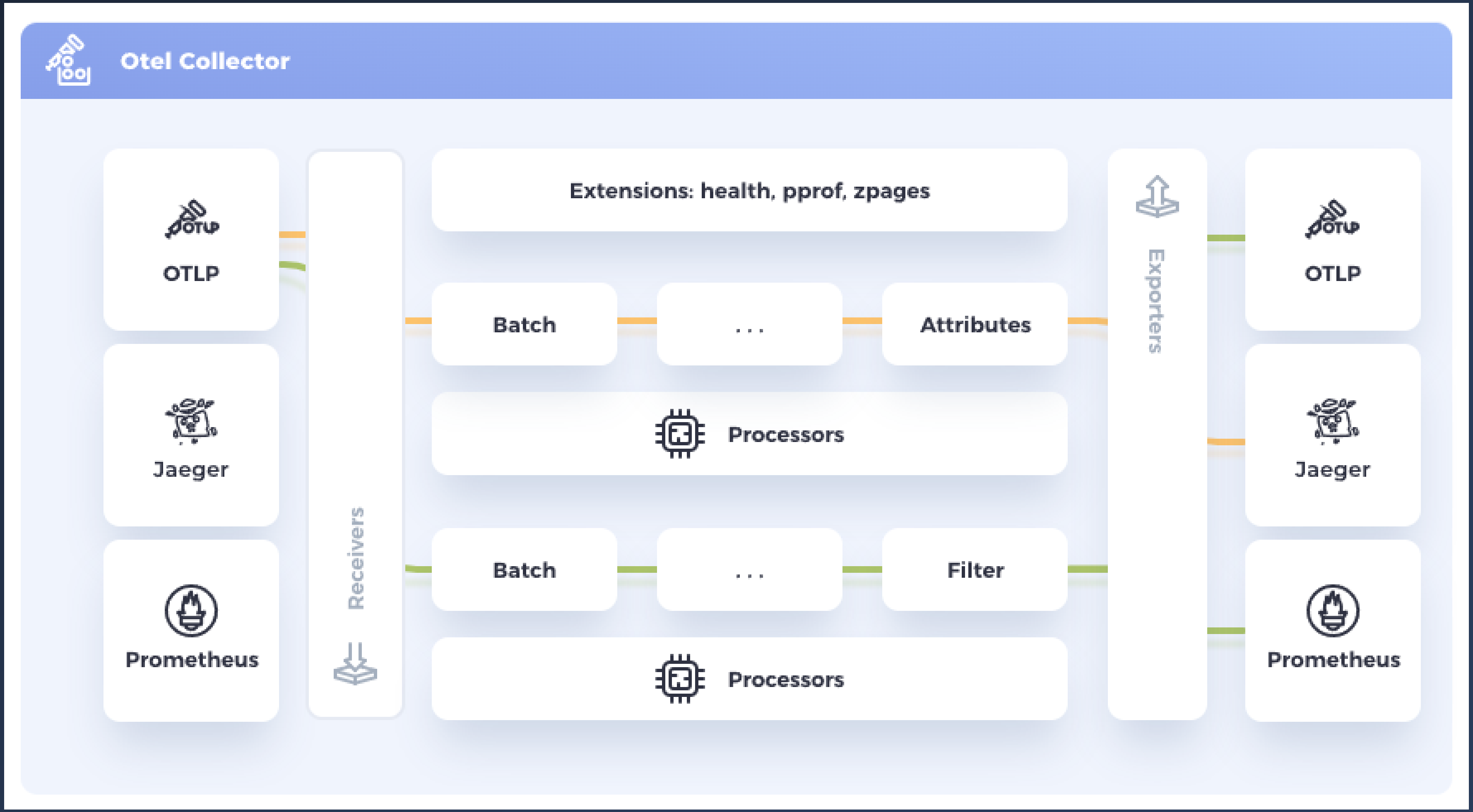
Metric Types – Histogram

Heatmap visualization example (histogram over time)



DEMO
Metrics

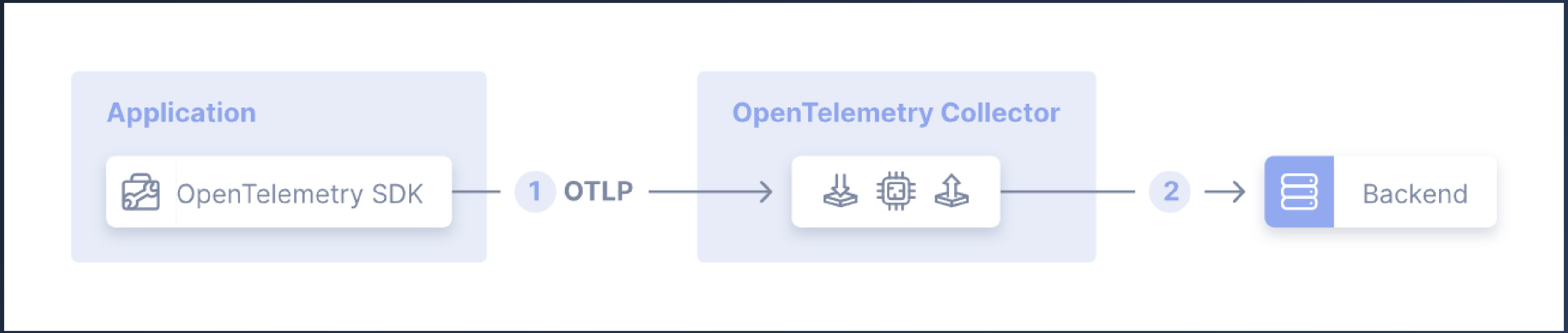
OpenTelemetry Collector

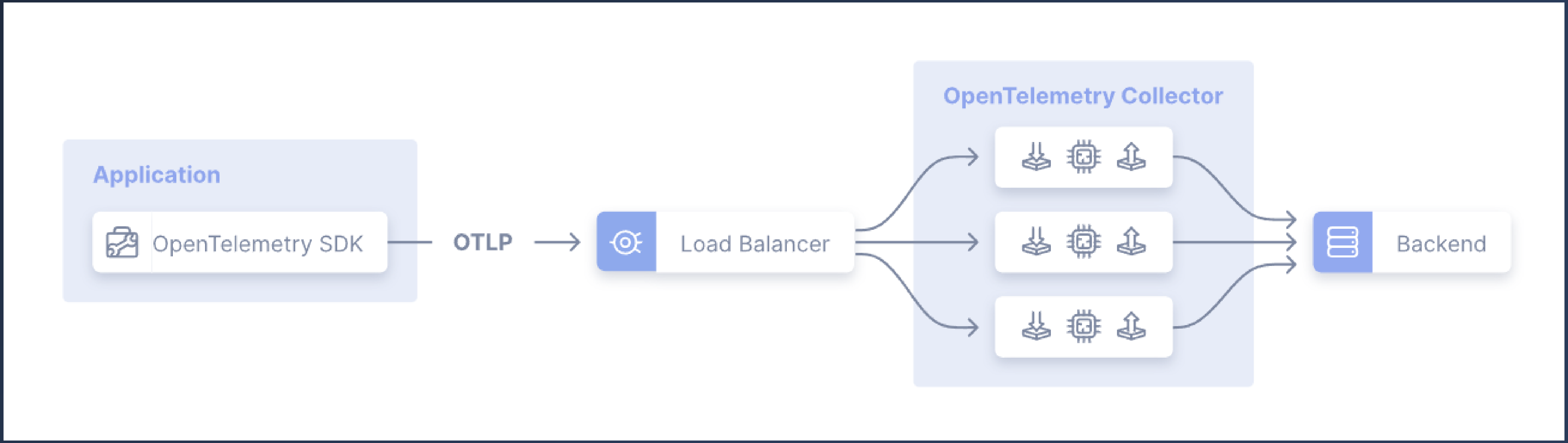


OpenTelemetry Collector

- Receive, process and export telemetry data
- https://opentelemetry.io/docs/collector/
- Alternatives: Logstash, Fluentd, Telegraf (InfluxDB), …
- Registry: https://opentelemetry.io/ecosystem/registry/?language=collector

Collector Deployment Models





DEMO
OpenTelemetry Collector

Zero-code Instrumentation for .NET

- https://opentelemetry.io/docs/languages/net/automatic/
- Steps:
    1. Install auto-instrumentation (once)
    2. Run `.otel-dotnet-auto/instrument.sh`
    3. Configure with env variables (OTEL_EXPORTER_OTLP_ENDPOINT, etc.)
    4. Run your app/service

- Works like *magic** 🦄 🌈

**magic* is limited to .NET 6+

DEMO
Zero Code Instrumentation

https://github.com/jechtom/demo-open-telemetry

Tomáš Jecha

LinkedIn /in/jechtom | X @jechtom