Observability Made Painless: Go, OTel & LGTM Stack

Introduction

- Haseeb Majid
  - Backend Software Engineer at Nala
  - https://haseebmajid.dev
- Loves cats 🐱
- Avid cricketer 🏏 #BazBall

Who is this for?

- New to OpenTelemetry
- Instrument an existing app

What is Observability?

- What is going on with our app
- Is something wrong?

What is Observability?

- Logs
- Traces
- Metrics


MONITORING?
OBSERVABILITY!
makeameme.org

Pizza Shop

Pizza Shop

Logs: “Order #123: Large veggie pizza burned at 8:05 PM due to oven failure.”

Pizza Shop

Logs: “Order #123: Large veggie pizza burned at 8:05 PM due to oven failure.”

Traces: “Order #123 took 30 mins: 5 mins prep → 20 mins cooking (delay) → 5 mins delivery.”

Pizza Shop

Logs: “Order #123: Large veggie pizza burned at 8:05 PM due to oven failure.”

Traces: “Order #123 took 30 mins: 5 mins prep → 20 mins cooking (delay) → 5 mins delivery.”

Metrics: “We sold 50 pizzas/hour (avg cook time: 8 mins).”

Me before pizza



Me after pizza



Why Observability Matters?

- Provide context to issues
- Bottlenecks in the system


WORKED FINE IN DEV
OPS PROBLEM NOW
imgflip.com

What is OTel?

- OpenTelemetry
- Open Standard
  - Solves vendor lock-in

Why use OTel?

- Open Standard
- Unify logs, metrics & traces

“Only half of programming is coding. The other 90% is debugging” - Anonymous

What is Tracing?

- Caused by a single action
- Components:
  - Services
  - DBs
  - Events

Span

- Operation name
- Start and finish timestamp
- Span context
- Attributes

Span (Cont...)

- A set of events
- Parent span ID
- Links to other spans
- Span Status

Span Context

```
traceparent: 00-d4cda95b652f4a1592b449d5929fda1b-6e0c6325
tracestate: mycompany=true
```

```
traceparent:
00-d4cda95b652f4a1592b449d5929fda1b-6e0c63257de34c92-01
```

```
version: 00
trace-id: d4cda95b652f4a1592b449d5929fda1b
span-id: 6e0c63257de34c92
trace-flags: 01 (sample this trace)
```

Span Links

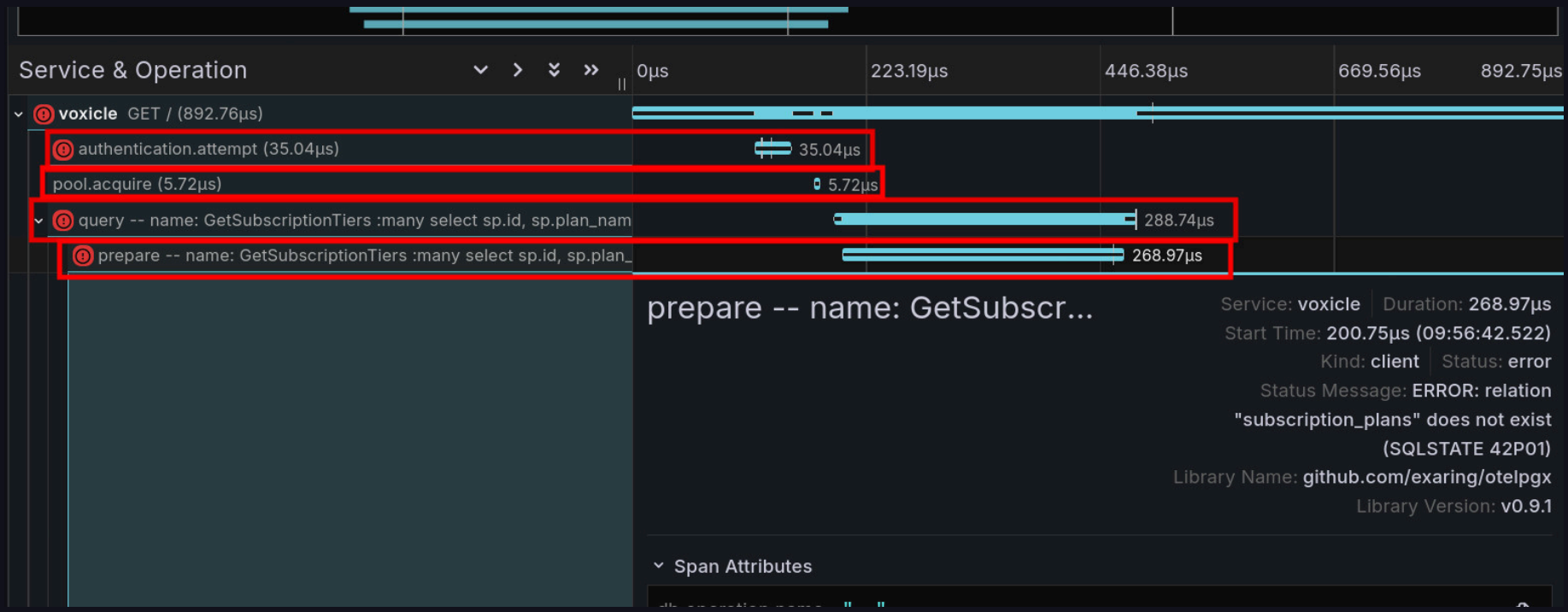
- Connect two spans
  - event-driven systems

Span Events

- Denote point in time
- Page becomes interactive

Span Kind

- One of:
  - Client, Server, Internal, Producer, or Consumer
- Default is internal
- Producer -> Consumer
- Client -> Server


Service & Operation
0µs
223.19µs
446.38µs
669.56µs
892.75µs
voxicle GET / (892.76µs)
authentication.attempt (35.04µs)
35.04µs
pool.acquire (5.72µs)
5.72µs
query -- name: GetSubscriptionTiers :many select sp.id, sp.plan_nam
288.74µs
prepare -- name: GetSubscriptionTiers :many select sp.id, sp.plan_
268.97µs
prepare -- name: GetSubscr...
Service: voxicle
Duration: 268.97µs
Start Time: 200.75µs (09:56:42.522)
Kind: client
Status: error
Status Message: ERROR: relation "subscription_plans" does not exist (SQLSTATE 42P01)
Library Name: github.com/exaring/otelpgx
Library Version: v0.9.1
Span Attributes

prepare -- name: GetSubscr...

Service: voxicle | Duration: 268.97µs
Start Time: 200.75µs (09:56:42.522)
Kind: client | Status: error
Status Message: ERROR: relation "subscription_plans" does not exist (SQLSTATE 42P01)
Library Name: github.com/exaring/otelpgx
Library Version: v0.9.1

Span Attributes

db.operation.name "--"

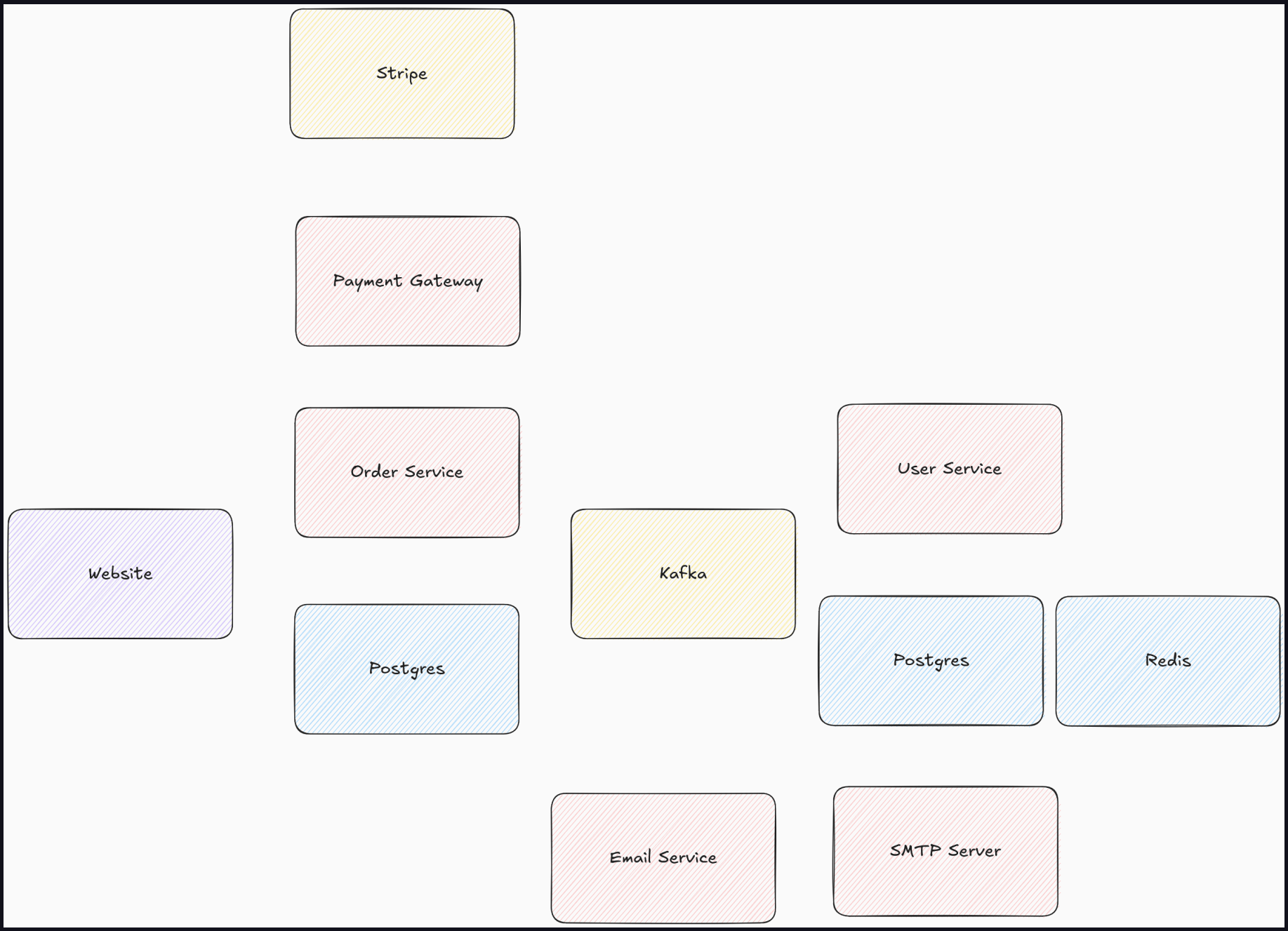
db.query.text

```
"-- name: GetSubscriptionTiers :many
select
    sp.id,
    sp.plan_name,
    sp.monthly_price,
    sp.annual_price,
    sp.currency,
    sp.payment_processor_monthly_plan_id,
    sp.payment_processor_annual_plan_id,
    sp.enabled,
    sp.created_at,
    sp.updated_at,
    coalesce(
        json_agg(
            json_build_object(
                'feature_key',
                pf.feature_key,
                'feature_text',
                pf.feature_text,
                'limit_value',
                pf.limit_value::integer,
                'created_at',
                pf.created_at,
                'updated_at',
```

Resource Attributes: container.id = 4640 | host.name = workstation | service.name = voxicle | servi...

Events (1)

SpanID: 484feb0f7eb25341


Stripe
Payment Gateway
Order Service
User Service
Website
Kafka
Postgres
Postgres
Redis
Email Service
SMTP Server

Example service

```
 1 package main
 2
 3 import (
 4   "encoding/json"
 5   "fmt"
 6   "log"
 7   "net/http"
 8   "time"
 9 )
10
11 func main() {
12     h := &Handler{
13         // ...
```

Example service

```
13          // ...
14      }
15
16      mux := http.NewServeMux()
17      mux.HandleFunc("GET /user/{id}", h.userHandler)
18
19      log.Println("Server starting on port 8080...")
20      log.Fatal(http.ListenAndServe(":8080", mux))
21  }
22
23  func (h *Handler) userHandler(
24      w http.ResponseWriter,
25      r *http.Request,
26  ) {
```

Example service

```
23 func (h *Handler) userHandler(
24     w http.ResponseWriter,
25     r *http.Request,
26 ) {
27     id := r.PathValue("id")
28     // Validation logic ...
29
30     // Interact with the DB.
31     user := h.store.getUser(id)
32
33     w.Header().Set("Content-Type", "application/json")
34     json.NewEncoder(w).Encode(user)
35 }
```

Install traces

```
go get go.opentelemetry.io/otel \
go.opentelemetry.io/otel/trace \
go.opentelemetry.io/contrib/instrumentation/net/http/\
otelhttp \
go.opentelemetry.io/otel/exporters/otlp/otlptrace/\
otlptracehttp \
go.opentelemetry.io/otel/sdk/resource \
go.opentelemetry.io/otel/sdk/trace \
go.opentelemetry.io/otel/semconv/v1.26.0
```

```
OTEL_EXPORTER_OTLP_ENDPOINT="http://localhost:4318"
```

```go
package main

import (
    // ... existing imports ...
      "fmt"
    "context"
    "log"
    "os"
    "time"

    "go.opentelemetry.io/otel"
    "go.opentelemetry.io/otel/attribute"
    "go.opentelemetry.io/otel/exporters/otlp/otlptrace/o
```

```
19
20
21 func newTracerProvider(ctx context.Context)
22     (*sdktrace.TracerProvider, error) {
23     // Create OTLP exporter
24     exporter, err := otlptracehttp.New(ctx)
25     if err ≠ nil {
26         return nil, err
27     }
28
29     // Create trace provider
30     tp := sdktrace.NewTracerProvider(
31         sdktrace.WithBatcher(exporter),
32     )
```

```
25      if err != nil {
26          return nil, err
27      }
28
29      // Create trace provider
30      tp := sdktrace.NewTracerProvider(
31          sdktrace.WithBatcher(exporter),
32      )
33
34      // Set global tracer provider
35      otel.SetTracerProvider(tp)
36      otel.SetTextMapPropagator(
37          propagation.NewCompositeTextMapPropagator(
```

```
29      // Create trace provider
30      tp := sdktrace.NewTracerProvider(
31          sdktrace.WithBatcher(exporter),
32      )
33
34      // Set global tracer provider
35      otel.SetTracerProvider(tp)
36      otel.SetTextMapPropagator(
37          propagation.NewCompositeTextMapPropagator(
38              propagation.TraceContext{},
39          )
40      )
41
```

```
32      )
33
34      // Set global tracer provider
35      otel.SetTracerProvider(tp)
36      otel.SetTextMapPropagator(
37          propagation.NewCompositeTextMapPropagator(
38              propagation.TraceContext{},
39          )
40      )
41
42      // Return shutdown function
43      return tp, nil
44  }
```

```
32      )
33
34      // Set global tracer provider
35      otel.SetTracerProvider(tp)
36      otel.SetTextMapPropagator(
37          propagation.NewCompositeTextMapPropagator(
38              propagation.TraceContext{},
39          )
40      )
41
42      // Return shutdown function
43      return tp, nil
44  }
```

Trace Context

```
traceparent: 00-d4cda95b652f4a1592b449d5929fda1b-6e0c6325
tracestate: mycompany=true
```

```go
func main() {
    ctx := context.Background()
    tp, err := newTracerProvider(ctx)
    if err != nil {
        log.Fatalf("failed to setup tracer: %v", err)
    }
    defer tp.Shutdown(ctx)

    // Previous code ...

    // Add OpenTelemetry middleware
  m := otelhttp.NewMiddleware("user-service",
        otelhttp.WithSpanNameFormatter(
```

```go
func main() {
    ctx := context.Background()
    tp, err := newTracerProvider(ctx)
    if err ≠ nil {
        log.Fatalf("failed to setup tracer: %v", err)
    }
    defer tp.Shutdown(ctx)

    // Previous code ...

    // Add OpenTelemetry middleware
  m := otelhttp.NewMiddleware("user-service",
        otelhttp.WithSpanNameFormatter(
```

```
10
11      // Add OpenTelemetry middleware
12      m := otelhttp.NewMiddleware("user-service",
13          otelhttp.WithSpanNameFormatter(
14          func(operation string, r *http.Request) string
15              return fmt.Sprintf("%s %s",
16                      r.Method,
17                      r.URL.Path,
18              )
19          }),
20      ),
21
22      log.Fatal(http.ListenAndServe(":8080", m(mux)))
```

```
12    m := otelhttp.NewMiddleware("user-service",
13          otelhttp.WithSpanNameFormatter(
14          func(operation string, r *http.Request) string
15              return fmt.Sprintf("%s %s",
16                      r.Method,
17                      r.URL.Path,
18              )
19          }),
20      ),
21
22      log.Fatal(http.ListenAndServe(":8080", m(mux)))
23
24  }
```

ctx

```
1 func (h *Handler) userHandler(
2     w http.ResponseWriter,
3     r *http.Request,
4 ) {
5
6     ctx := r.Context()
7     span := trace.SpanFromContext(ctx)
8     baggage := baggage.FromContext(ctx)
9 }
```

ctx

```
1 func (h *Handler) userHandler(
2     w http.ResponseWriter,
3     r *http.Request,
4 ) {
5
6     ctx := r.Context()
7     span := trace.SpanFromContext(ctx)
8     baggage := baggage.FromContext(ctx)
9 }
```

ctx

```
1 func (h *Handler) userHandler(
2     w http.ResponseWriter,
3     r *http.Request,
4 ) {
5
6     ctx := r.Context()
7     span := trace.SpanFromContext(ctx)
8     baggage := baggage.FromContext(ctx)
9 }
```

Custom Trace

```go
func getUser(ctx context.Context, userID string) (*Use
  ctx, span := otel.Tracer("user-service")
                   .Start(ctx, "getUser")
  defer span.End()
    span.SetAttributes(
        attribute.String("user.id", userID),
    )

  user, err := dbFetch(ctx, userID)
  if err != nil {
    span.SetStatus(codes.Error, "database error")
    span.RecordError(err)
    return nil, err
```

Custom Trace

```go
func getUser(ctx context.Context, userID string) (*Use
  ctx, span := otel.Tracer("user-service")
                   .Start(ctx, "getUser")
  defer span.End()
    span.SetAttributes(
        attribute.String("user.id", userID),
    )

  user, err := dbFetch(ctx, userID)
  if err != nil {
    span.SetStatus(codes.Error, "database error")
    span.RecordError(err)
    return nil, err
```

Custom Trace

```
 4    defer span.End()
 5      span.SetAttributes(
 6          attribute.String("user.id", userID),
 7      )
 8
 9    user, err := dbFetch(ctx, userID)
10    if err ≠ nil {
11      span.SetStatus(codes.Error, "database error")
12      span.RecordError(err)
13      return nil, err
14    }
15
16    if user.Premium {
```

Custom Trace

```
 6          attribute.String("user.id", userID),
 7      )
 8
 9      user, err := dbFetch(ctx, userID)
10      if err != nil {
11          span.SetStatus(codes.Error, "database error")
12          span.RecordError(err)
13          return nil, err
14      }
15
16      if user.Premium {
17          span.SetAttributes(
18              attribute.Bool("user.premium", true),
```

Custom Trace

```
10    if err != nil {
11      span.SetStatus(codes.Error, "database error")
12      span.RecordError(err)
13      return nil, err
14    }
15
16    if user.Premium {
17          span.SetAttributes(
18              attribute.Bool("user.premium", true),
19          )
20    }
21    return user, nil
22  }
```

Trace JSON

```
{
  "name": "hello",
  "context": {
    "trace_id": "5b8aa5a2d2c872e8321cf37308d69df2",
    "span_id": "051581bf3cb55c13"
  },
  "parent_id": null,
  "start_time": "2022-04-29T18:52:58.114201Z",
  "end_time": "2022-04-29T18:52:58.114687Z",
  "attributes": {
    "http.route": "some_route1"
  },
  "events": [
```

Trace JSON

```
{
  "name": "hello",
  "context": {
    "trace_id": "5b8aa5a2d2c872e8321cf37308d69df2",
    "span_id": "051581bf3cb55c13"
  },
  "parent_id": null,
  "start_time": "2022-04-29T18:52:58.114201Z",
  "end_time": "2022-04-29T18:52:58.114687Z",
  "attributes": {
    "http.route": "some_route1"
  },
  "events": [
```

Trace JSON

```
{
  "name": "hello",
  "context": {
    "trace_id": "5b8aa5a2d2c872e8321cf37308d69df2",
    "span_id": "051581bf3cb55c13"
  },
  "parent_id": null,
  "start_time": "2022-04-29T18:52:58.114201Z",
  "end_time": "2022-04-29T18:52:58.114687Z",
  "attributes": {
    "http.route": "some_route1"
  },
  "events": [
```

Trace JSON

```
    "span_id": "051581bf3cb55c13"
  },
  "parent_id": null,
  "start_time": "2022-04-29T18:52:58.114201Z",
  "end_time": "2022-04-29T18:52:58.114687Z",
  "attributes": {
    "http.route": "some_route1"
  },
  "events": [
    {
      "name": "Guten Tag!",
      "timestamp": "2022-04-29T18:52:58.114561Z",
      "attributes": {
```

Trace JSON

```
10    "attributes": {
11      "http.route": "some_route1"
12    },
13    "events": [
14      {
15        "name": "Guten Tag!",
16        "timestamp": "2022-04-29T18:52:58.114561Z",
17        "attributes": {
18          "event_attributes": 1
19        }
20      }
21    ]
22  }
```

Postgres

```
go get github.com/exaring/otelpgx
```

```go
func NewPool(ctx context.Context, uri string) (*pgxpoo
  pgxConfig, err := pgxpool.ParseConfig(uri)
  if err != nil {
    return nil, fmt.Errorf("failed to parse db URI: %w
  }
  pgxConfig.ConnConfig.Tracer = otelpgx.NewTracer()

  pool, err := pgxpool.NewWithConfig(ctx, pgxConfig)
  if err != nil {
    return nil, fmt.Errorf("failed to setup database: 
  }

  return pool, err
```

prepare -- name: GetSubscr...

Service: voxicle | Duration: 268.97µs
Start Time: 200.75µs (09:56:42.522)
Kind: client | Status: error
Status Message: ERROR: relation "subscription_plans" does not exist (SQLSTATE 42P01)
Library Name: github.com/exaring/otelpgx
Library Version: v0.9.1

Span Attributes

db.operation.name `"--"`

db.query.text

```
"-- name: GetSubscriptionTiers :many
select
    sp.id,
    sp.plan_name,
    sp.monthly_price,
    sp.annual_price,
    sp.currency,
    sp.payment_processor_monthly_plan_id,
    sp.payment_processor_annual_plan_id,
    sp.enabled,
    sp.created_at,
    sp.updated_at,
    coalesce(
        json_agg(
            json_build_object(
                'feature_key',
                pf.feature_key,
                'feature_text',
                pf.feature_text,
                'limit_value',
                pf.limit_value::integer,
                'created_at',
                pf.created_at,
                'updated_at',
```

Resource Attributes: container.id = 4640 | host.name = workstation | service.name = voxicle | servi...

Events (1)

SpanID: 484feb0f7eb25341

Redis

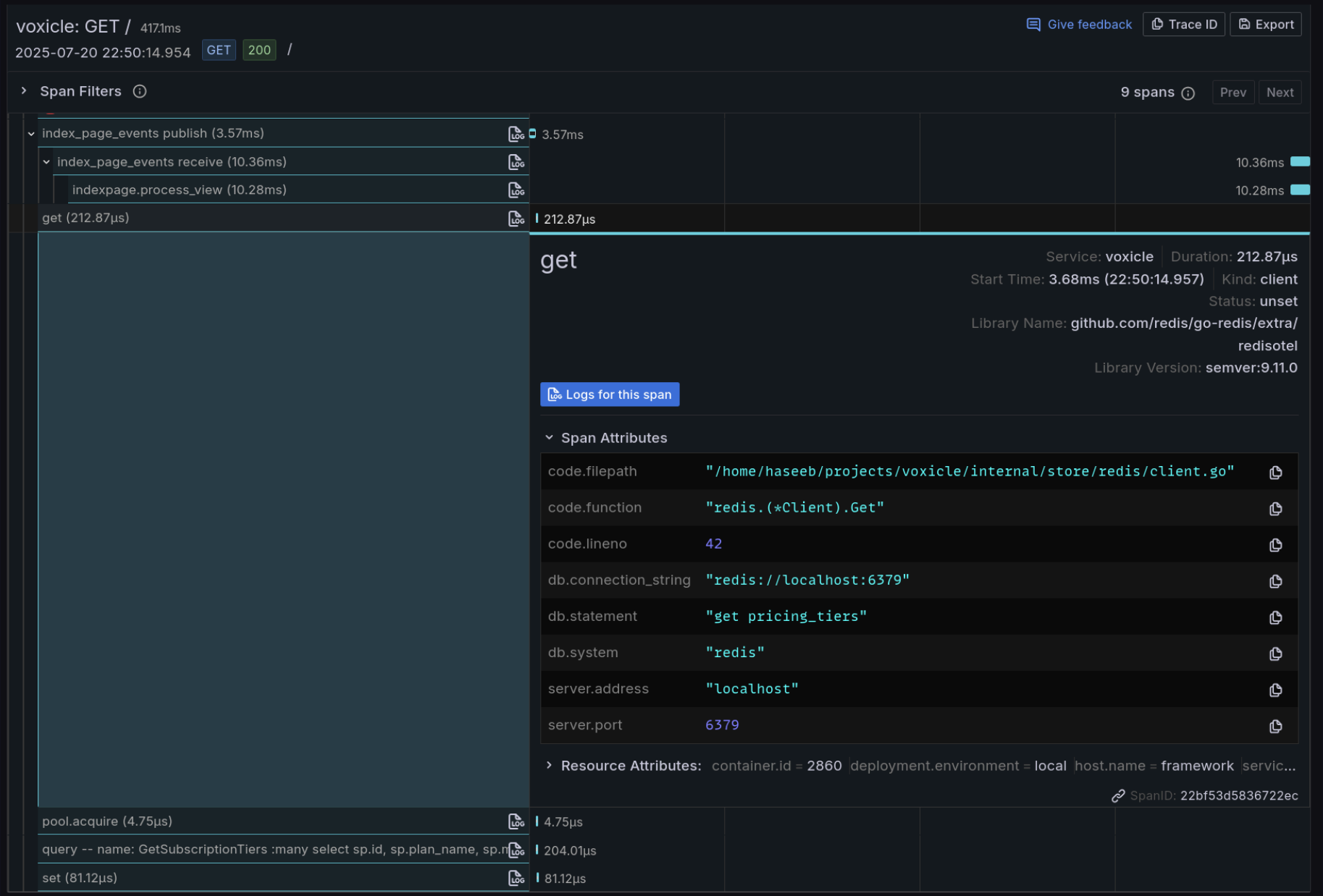
```
go get github.com/redis/go-redis/extra/redisotel/v9
```

```go
func NewRedisClient(address string, retries int) (Clie
    r := redis.NewClient(&redis.Options{
        Addr:       address,
        Password:   "",
        DB:         0,
        MaxRetries: retries,
    })

    err := redisotel.InstrumentTracing(r)
    if  err != nil {
        return Client{}, err
    }

```

```
 3          Addr:        address,
 4          Password:    "",
 5          DB:          0,
 6          MaxRetries: retries,
 7      })
 8
 9      err := redisotel.InstrumentTracing(r)
10      if  err ≠ nil {
11          return Client{}, err
12      }
13
14      return Client{
15          Redis:       r,
```


voxicle: GET / 417.1ms
2025-07-20 22:50:14.954 GET 200 /
Give feedback
Trace ID
Export
Span Filters
9 spans
Prev
Next
index_page_events publish (3.57ms)
3.57ms
index_page_events receive (10.36ms)
10.36ms
indexpage.process_view (10.28ms)
10.28ms
get (212.87µs)
212.87µs
get
Service: voxicle
Duration: 212.87µs
Start Time: 3.68ms (22:50:14.957)
Kind: client
Status: unset
Library Name: github.com/redis/go-redis/extra/redisotel
Library Version: semver:9.11.0
Logs for this span
Span Attributes
code.filepath "/home/haseeb/projects/voxicle/internal/store/redis/client.go"
code.function "redis.(*Client).Get"
code.lineno 42
db.connection_string "redis://localhost:6379"
db.statement "get pricing_tiers"
db.system "redis"
server.address "localhost"
server.port 6379
Resource Attributes: container.id = 2860 deployment.environment = local host.name = framework servic...
SpanID: 22bf53d5836722ec
pool.acquire (4.75µs)
4.75µs
query -- name: GetSubscriptionTiers :many select sp.id, sp.plan_name, sp.n
204.01µs
set (81.12µs)
81.12µs

HTTP Client

```
go get \
go.opentelemetry.io/contrib/instrumentation/net/http/\
otelhttp
```

```go
func NewHTTPClient() *http.Client {
    transport := otelhttp.NewTransport(
        http.DefaultTransport,
        otelhttp.WithSpanNameFormatter(
        func(operation string, r *http.Request)
        string {
            return fmt.Sprintf("%s %s",
                r.Method,
                // INFO: From /user/123 → /user/id
                sanitizePath(r.URL.Path),
            )
        }),
    )
```

```go
func NewHTTPClient() *http.Client {
    transport := otelhttp.NewTransport(
        http.DefaultTransport,
        otelhttp.WithSpanNameFormatter(
        func(operation string, r *http.Request)
        string {
            return fmt.Sprintf("%s %s",
                r.Method,
                // INFO: From /user/123 → /user/id
                sanitizePath(r.URL.Path),
            )
        }),
    )
```

```go
func NewHTTPClient() *http.Client {
    transport := otelhttp.NewTransport(
        http.DefaultTransport,
        otelhttp.WithSpanNameFormatter(
        func(operation string, r *http.Request)
        string {
            return fmt.Sprintf("%s %s",
                r.Method,
                // INFO: From /user/123 → /user/id
                sanitizePath(r.URL.Path),
            )
        }),
    )
```

```
        http.DefaultTransport,
        otelhttp.WithSpanNameFormatter(
        func(operation string, r *http.Request)
        string {
            return fmt.Sprintf("%s %s",
                r.Method,
                // INFO: From /user/123 → /user/id
                sanitizePath(r.URL.Path),
            )
        }),
    )

    return &http.Client{
        Transport: transport
```

```
        http.DefaultTransport,
        otelhttp.WithSpanNameFormatter(
        func(operation string, r *http.Request)
        string {
            return fmt.Sprintf("%s %s",
                r.Method,
                // INFO: From /user/123 → /user/id
                sanitizePath(r.URL.Path),
            )
        }),
    )

    return &http.Client{
        Transport: transport
```

```
15      return &http.Client{
16          Transport: transport,
17          Timeout:   5 * time.Second,
18      }
19  }
20
21  func (s *Service) callExternalAPI(ctx context.Context)
22      client := NewHTTPClient()
23      req, _ := http.NewRequestWithContext(
24          ctx,
25          "GET",
26          "https://api.example.com/data",
27          nil,
```

```
        Transport: transport,
        Timeout:   5 * time.Second,
    }
}

func (s *Service) callExternalAPI(ctx context.Context)
    client := NewHTTPClient()
    req, _ := http.NewRequestWithContext(
        ctx,
        "GET",
        "https://api.example.com/data",
        nil,
    )
```

```
20
21 func (s *Service) callExternalAPI(ctx context.Context)
22     client := NewHTTPClient()
23     req, _ := http.NewRequestWithContext(
24         ctx,
25         "GET",
26         "https://api.example.com/data",
27         nil,
28     )
29
30     // Trace context automatically injected!
31     resp, err := client.Do(req)
32
```

```
22      client := NewHTTPClient()
23      req, _ := http.NewRequestWithContext(
24          ctx,
25          "GET",
26          "https://api.example.com/data",
27          nil,
28      )
29
30      // Trace context automatically injected!
31      resp, err := client.Do(req)
32
33      // ... handle response ...
34  }
```

Kafka

```
go get github.com/twmb/franz-go \
    github.com/twmb/franz-go/plugin/kotel
```

```go
import (
  "github.com/twmb/franz-go/pkg/kgo"
  "github.com/twmb/franz-go/plugin/kotel"
)

func NewKafkaClient(brokers []string, group string) (*
    tracer := kotel.NewTracer(
        kotel.WithTracerProvider(
            otel.GetTracerProvider(),
        ),
    )

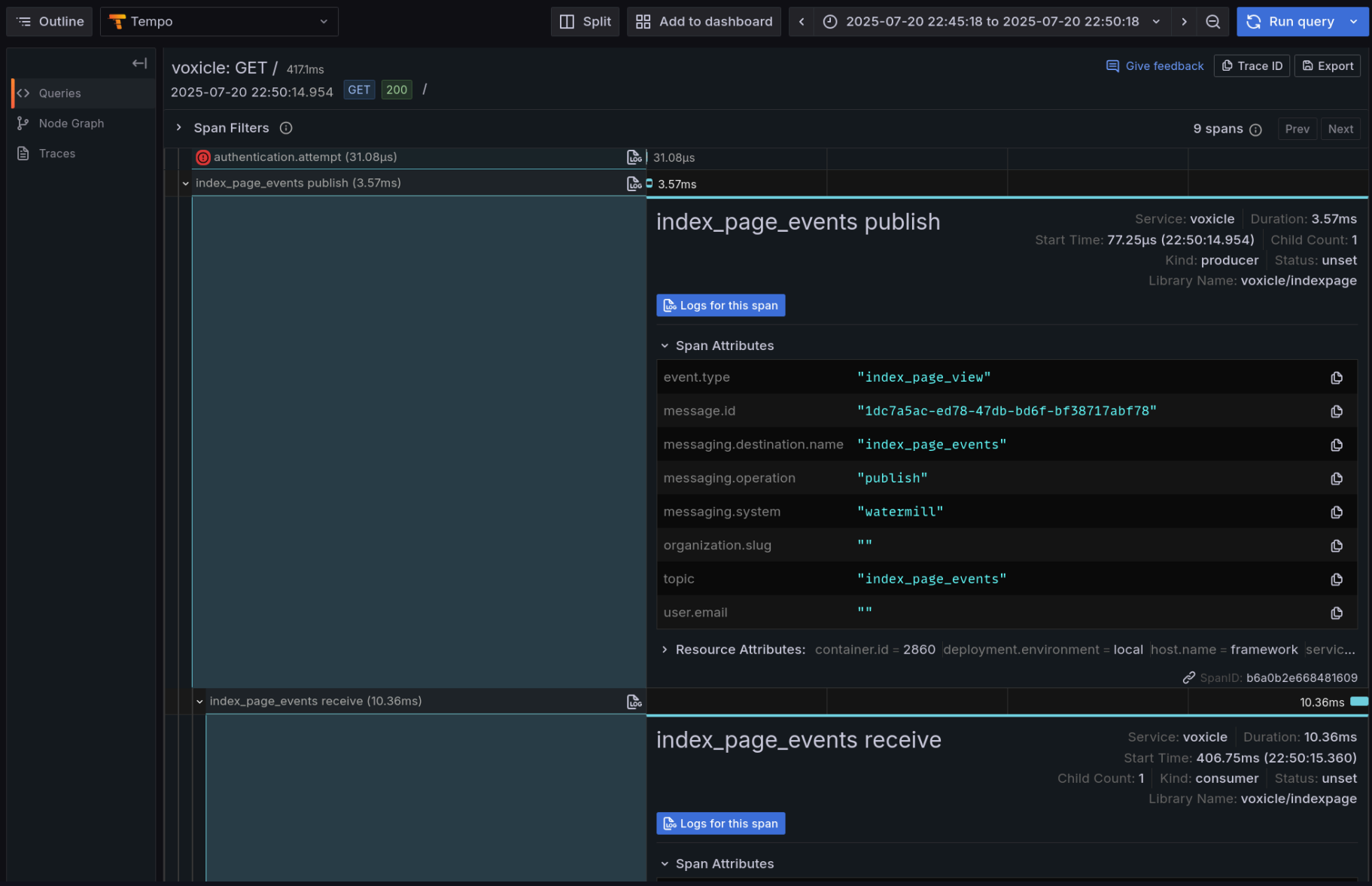
    opts := []kgo.Opt{
```

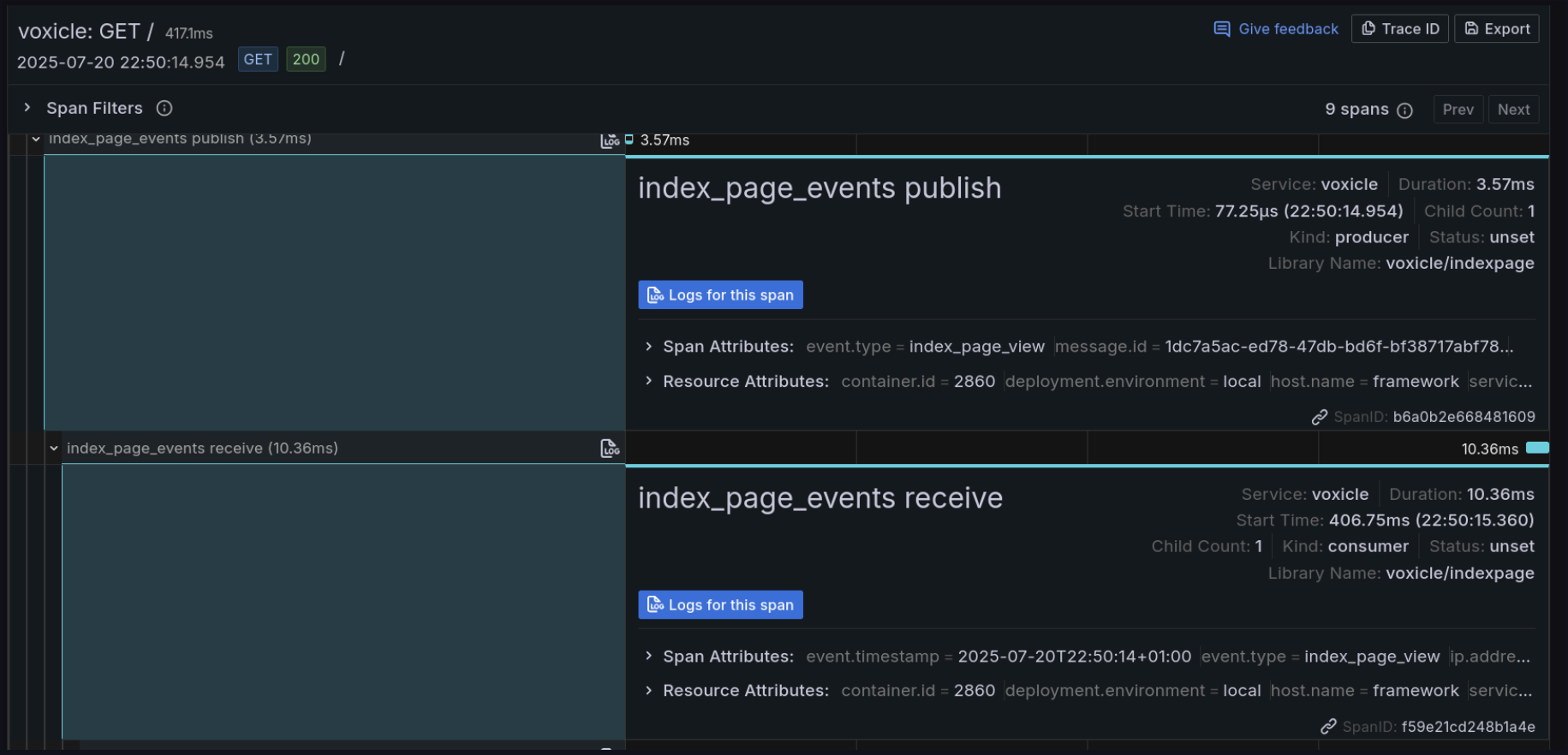
```
10          ),
11      )
12
13      opts := []kgo.Opt{
14          kgo.SeedBrokers(brokers...),
15          kgo.ConsumerGroup(group),
16          kgo.WithHooks(tracer.Hooks()),
17      }
18
19      opts = append(opts, kgo.WithProduceBatchIntercepto
20          kotel.NewProduceBatchInterceptor(tracer),
21      ))
22
```

```
24 }
25
26 func (s *Service) produceMessage(ctx context.Context,
27     record := &kgo.Record{
28         Topic: topic,
29         Value: []byte(msg),
30         Headers: []kgo.RecordHeader{},
31     }
32
33     s.kafkaClient.Produce(ctx, record,
34         func(r *kgo.Record, err error) {
35             // Do stuff ...
36         }
```

```
29          Value: []byte(msg),
30          Headers: []kgo.RecordHeader{},
31      }
32
33      s.kafkaClient.Produce(ctx, record,
34          func(r *kgo.Record, err error) {
35              // Do stuff ...
36          }
37      )
38  }
39
40  func (s *Service) consumeMessages(ctx context.Context)
41      for {
```

```
35                // Do stuff ...
36            }
37        )
38    }
39
40    func (s *Service) consumeMessages(ctx context.Context)
41        for {
42            fetches := s.kafkaClient.PollFetches(ctx)
43            fetches.EachRecord(func(r *kgo.Record) {
44                processMessage(ctx, r)
45            })
46        }
47    }
```


Outline
Tempo
Split
Add to dashboard
2025-07-20 22:45:18 to 2025-07-20 22:50:18
Run query
Queries
Node Graph
Traces
voxicle: GET / 417.1ms
2025-07-20 22:50:14.954 GET 200 /
Give feedback
Trace ID
Export
Span Filters
9 spans
Prev
Next
authentication.attempt (31.08µs)
31.08µs
index_page_events publish (3.57ms)
3.57ms
index_page_events publish
Service: voxicle
Duration: 3.57ms
Start Time: 77.25µs (22:50:14.954)
Child Count: 1
Kind: producer
Status: unset
Library Name: voxicle/indexpage
Logs for this span
Span Attributes
event.type "index_page_view"
message.id "1dc7a5ac-ed78-47db-bd6f-bf38717abf78"
messaging.destination.name "index_page_events"
messaging.operation "publish"
messaging.system "watermill"
organization.slug ""
topic "index_page_events"
user.email ""
Resource Attributes: container.id = 2860 deployment.environment = local host.name = framework servic...
SpanID: b6a0b2e668481609
index_page_events receive (10.36ms)
10.36ms
index_page_events receive
Service: voxicle
Duration: 10.36ms
Start Time: 406.75ms (22:50:15.360)
Child Count: 1
Kind: consumer
Status: unset
Library Name: voxicle/indexpage
Logs for this span
Span Attributes


voxicle: GET / 417.1ms
2025-07-20 22:50:14.954 GET 200 /
Give feedback
Trace ID
Export
Span Filters
9 spans
Prev
Next
index_page_events publish (3.57ms)
3.57ms
index_page_events publish
Service: voxicle
Duration: 3.57ms
Start Time: 77.25µs (22:50:14.954)
Child Count: 1
Kind: producer
Status: unset
Library Name: voxicle/indexpage
Logs for this span
Span Attributes: event.type = index_page_view message.id = 1dc7a5ac-ed78-47db-bd6f-bf38717abf78...
Resource Attributes: container.id = 2860 deployment.environment = local host.name = framework servic...
SpanID: b6a0b2e668481609
index_page_events receive (10.36ms)
10.36ms
index_page_events receive
Service: voxicle
Duration: 10.36ms
Start Time: 406.75ms (22:50:15.360)
Child Count: 1
Kind: consumer
Status: unset
Library Name: voxicle/indexpage
Logs for this span
Span Attributes: event.timestamp = 2025-07-20T22:50:14+01:00 event.type = index_page_view ip.addre...
Resource Attributes: container.id = 2860 deployment.environment = local host.name = framework servic...
SpanID: f59e21cd248b1a4e

Sampling

- Head vs Tail

Metrics

- Numerical Data
  - Low Context
  - Fast to query
- Time series data

Metric Types

- Counter: for tracking ever-increasing values
- Gauge: for measuring fluctuating values

Metric Types (Cont...)

- Histogram: for observing the distribution of values within predefined buckets.
- UpDownCounter: for values that go up and down

Metric Model

- Name: A descriptive name like http.server.request_count
- Attributes: Key-value pairs that provide context

Metric Model (Cont...)

- Timestamp: The time at which the data point was collected
- Value: The actual numerical value of the metric at that timestamp

Install metrics

```
go get go.opentelemetry.io/otel/metric \
go.opentelemetry.io/otel/sdk/metric \
go.opentelemetry.io/otel/exporters/otlp/ \
otlpmetric/otlpmetrichttp \
go.opentelemetry.io/contrib/instrumentation/runtime \
go.opentelemetry.io/contrib/instrumentation/host
```

Instrument metrics

```go
func newMeterProvider(ctx context.Context)
    (*sdkmetric.MeterProvider, error) {

    exporter, err := otlpmetrichttp.New(ctx)
    if err != nil {
        return nil, err
    }

    mp := sdkmetric.NewMeterProvider(
        sdkmetric.WithReader(
            sdkmetric.NewPeriodicReader(exporter)
        ),
    )
```

Instrument metrics

```
 1 func newMeterProvider(ctx context.Context)
 2     (*sdkmetric.MeterProvider, error) {
 3
 4     exporter, err := otlpmetrichttp.New(ctx)
 5     if err ≠ nil {
 6         return nil, err
 7     }
 8
 9     mp := sdkmetric.NewMeterProvider(
10         sdkmetric.WithReader(
11             sdkmetric.NewPeriodicReader(exporter)
12         ),
13     )
```

Instrument metrics

```
 5      if err ≠ nil {
 6          return nil, err
 7      }
 8
 9      mp := sdkmetric.NewMeterProvider(
10          sdkmetric.WithReader(
11              sdkmetric.NewPeriodicReader(exporter)
12          ),
13      )
14
15      err = runtimeMetrics.Start(
16          runtimeMetrics.WithMeterProvider(mp)
17      )
```

Instrument metrics

```
11              sdkmetric.NewPeriodicReader(exporter)
12          ),
13      )
14
15      err = runtimeMetrics.Start(
16          runtimeMetrics.WithMeterProvider(mp)
17      )
18      if  err ≠ nil {
19          return nil, err
20      }
21
22      err = hostMetrics.Start(
23          hostMetrics.WithMeterProvider(mp)
```

Instrument metrics

```
18      if err ≠ nil {
19          return nil, err
20      }
21
22      err = hostMetrics.Start(
23          hostMetrics.WithMeterProvider(mp)
24      )
25      if err ≠ nil {
26          return nil, err
27      }
28
29      // Set global meter provider
30      otel.SetMeterProvider(mp)
31      return mp, nil
```

Instrument metrics

```
20      }
21
22      err = hostMetrics.Start(
23          hostMetrics.WithMeterProvider(mp)
24      )
25      if  err ≠ nil {
26          return nil, err
27      }
28
29      // Set global meter provider
30      otel.SetMeterProvider(mp)
31      return mp, nil
32  }
```

Instrument metrics

```go
func main() {
    ctx := context.Background()

    // Previous code ...

    // Setup meter
    mp, err := newMeterProvider(ctx)
    if err != nil {
        log.Fatalf("failed to setup meter: %v", err)
    }
    defer mp.Shutdown(ctx)

    // Rest of the code ...
```

Middleware

```go
package middleware

import (
  "net/http"
  "strings"
  "sync"
  "time"

  "go.opentelemetry.io/otel"
  "go.opentelemetry.io/otel/attribute"
  "go.opentelemetry.io/otel/metric"
  semconv "go.opentelemetry.io/otel/semconv/v1.26.0"
)
```

Middleware

```
78      if statusCode == 0 {
79        statusCode = http.StatusOK
80      }
81
82      attrs := []attribute.KeyValue{
83        semconv.HTTPRequestMethodKey.String(r.Method),
84        semconv.HTTPResponseStatusCode(statusCode),
85        semconv.HTTPRoute(r.URL.EscapedPath()),
86      }
87
88      ctx := r.Context()
89      if requestCount != nil {
90              requestCount.Add(ctx, 1,
```

Middleware

```
85          semconv.HTTPRoute(r.URL.EscapedPath()),
86      }
87
88      ctx := r.Context()
89      if requestCount ≠ nil {
90              requestCount.Add(ctx, 1,
91                  metric.WithAttributes(attrs...)
92              )
93      }
94      if latencyHist ≠ nil {
95          latencyHist.Record(ctx, float64(duration), metr
96      }
97      if requestSize ≠ nil && r.ContentLength ⩾ 0 {
```

High Cardinality

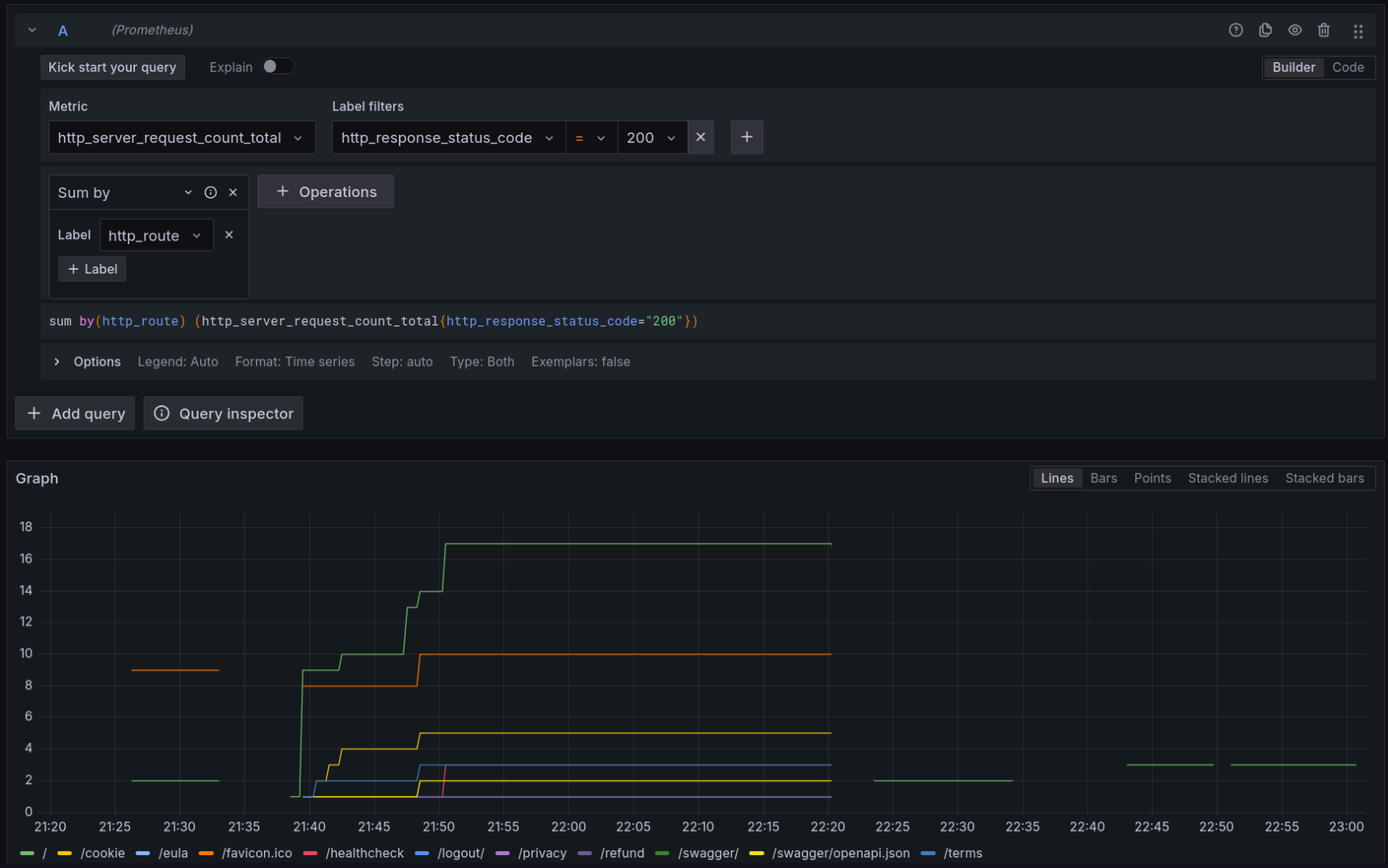
```
requestCounter.Add(ctx, 1, metric.WithAttributes(
    // 1M possible values
    attribute.String("user_id", "user_12345"),
    // 100 endpoints
    attribute.String("endpoint", "/api/users"),
    // 5 methods
    attribute.String("method", "GET"),
    // 20 status codes
    attribute.String("status", "200"),
))

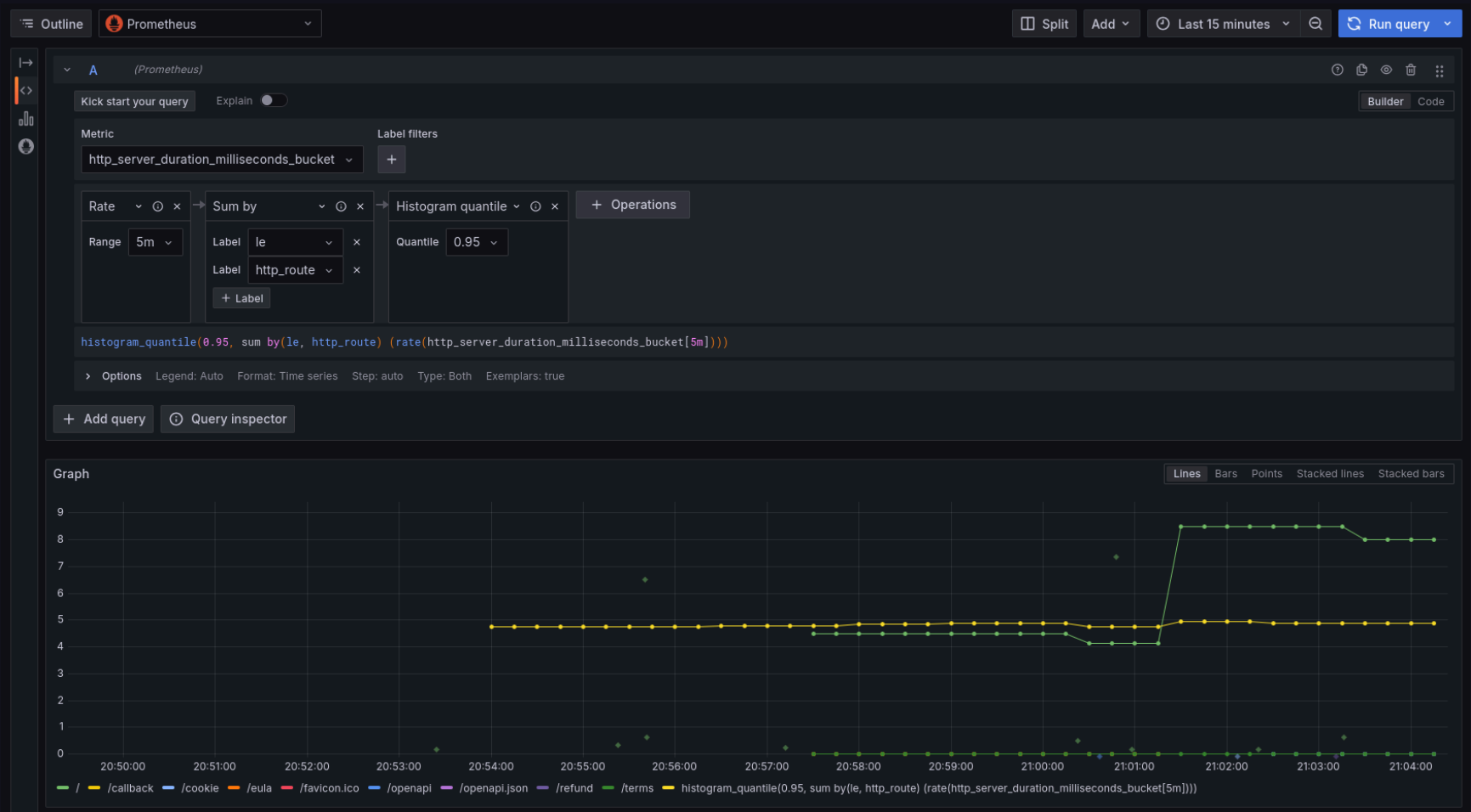
// Total time series = 1M × 100 × 5 × 20
// = 10 BILLION time series! 💥
```

High Cardinality

```
requestCounter.Add(ctx, 1, metric.WithAttributes(
    // 1M possible values
    attribute.String("user_id", "user_12345"),
    // 100 endpoints
    attribute.String("endpoint", "/api/users"),
    // 5 methods
    attribute.String("method", "GET"),
    // 20 status codes
    attribute.String("status", "200"),
))

// Total time series = 1M × 100 × 5 × 20
// = 10 BILLION time series! 💥
```


A (Prometheus)
Kick start your query
Explain
Builder Code
Metric
http_server_request_count_total
Label filters
http_response_status_code = 200
Sum by
Operations
Label http_route
Label
sum by(http_route) (http_server_request_count_total{http_response_status_code="200"})
Options Legend: Auto Format: Time series Step: auto Type: Both Exemplars: false
Add query
Query inspector
Graph
Lines Bars Points Stacked lines Stacked bars
18 16 14 12 10 8 6 4 2 0
21:20 21:25 21:30 21:35 21:40 21:45 21:50 21:55 22:00 22:05 22:10 22:15 22:20 22:25 22:30 22:35 22:40 22:45 22:50 22:55 23:00
/ /cookie /eula /favicon.ico /healthcheck /logout/ /privacy /refund /swagger/ /swagger/openapi.json /terms


Outline
Prometheus
Split
Add
Last 15 minutes
Run query
A
(Prometheus)
Kick start your query
Explain
Builder
Code
Metric
http_server_duration_milliseconds_bucket
Label filters
Rate
Range
5m
Sum by
Label
le
Label
http_route
+ Label
Histogram quantile
Quantile
0.95
+ Operations
histogram_quantile(0.95, sum by(le, http_route) (rate(http_server_duration_milliseconds_bucket[5m])))
Options
Legend: Auto
Format: Time series
Step: auto
Type: Both
Exemplars: true
+ Add query
Query inspector
Graph
Lines
Bars
Points
Stacked lines
Stacked bars
9
8
7
6
5
4
3
2
1
0
20:50:00
20:51:00
20:52:00
20:53:00
20:54:00
20:55:00
20:56:00
20:57:00
20:58:00
20:59:00
21:00:00
21:01:00
21:02:00
21:03:00
21:04:00
/
/callback
/cookie
/eula
/favicon.ico
/openapi
/openapi.json
/refund
/terms
histogram_quantile(0.95, sum by(le, http_route) (rate(http_server_duration_milliseconds_bucket[5m])))

```
# ≤ 100ms: 100 requests
http_request_duration_bucket{le="0.1"} 100
# ≤ 500ms: 250 requests
http_request_duration_bucket{le="0.5"} 250
# ≤ 800ms: 290 requests
http_request_duration_bucket{le="0.8"} 290
# ≤ 1000ms: 310 requests
http_request_duration_bucket{le="1.0"} 310
# All requests: 320 total
http_request_duration_bucket{le="+Inf"} 320

# ~ 0.94 seconds
```


CAN U
SPOT THE CAT?
imgflip.com

Logs

Why OTel & Logging?

- Context Propagation: Attach trace context to logs
- Correlation: Link logs directly to traces

Why OTel & Logging (Cont...)?

- Unified Schema: Consistent attributes across signals
- Reduced Overhead: Single instrumentation pipeline

Example Log

```
{
  "time": "2023-10-05T12:34:56Z",
  "level": "ERROR",
  "msg": "Failed to get user",
  "trace_id": "d4cda95b652f4a1592b449d5929fda1b",
  "span_id": "6e0c63257de34c92",
  "user_id": "12345",
  "service.name": "user-service",
  "error": "record not found"
}
```

Install logs

```
go get go.opentelemetry.io/otel/log \
 go.opentelemetry.io/otel/log/global \
 go.opentelemetry.io/otel/sdk/log \
 go.opentelemetry.io/otel/exporters/otlp/ \
 otlplog/otlploghttp \
 go.opentelemetry.io/contrib/bridges/otelslog \
 github.com/lmittmann/tint \
 github.com/samber/slog-multi
```

Instrument logs

```go
func newLoggerProvider(
    ctx context.Context,
    logLevel minsev.Severity,
) (*log.LoggerProvider, error) {
    exporter, err := otlploghttp.New(ctx)
    if err ≠ nil {
        return nil, err
    }

    p := log.NewBatchProcessor(exporter)
    processor := minsev.NewLogProcessor(p, logLevel)
    lp := log.NewLoggerProvider(
        log.WithProcessor(processor),
```

Instrument logs

```
 1 func newLoggerProvider(
 2     ctx context.Context,
 3     logLevel minsev.Severity,
 4 ) (*log.LoggerProvider, error) {
 5     exporter, err := otlploghttp.New(ctx)
 6     if err ≠ nil {
 7         return nil, err
 8     }
 9
10     p := log.NewBatchProcessor(exporter)
11     processor := minsev.NewLogProcessor(p, logLevel)
12     lp := log.NewLoggerProvider(
13         log.WithProcessor(processor),
```

Instrument logs

```
 6      if err ≠ nil {
 7          return nil, err
 8      }
 9
10      p := log.NewBatchProcessor(exporter)
11      processor := minsev.NewLogProcessor(p, logLevel)
12      lp := log.NewLoggerProvider(
13          log.WithProcessor(processor),
14      )
15
16      global.SetLoggerProvider(lp)
17      return lp, nil
18  }
```

Instrument logs

```
 6      if err != nil {
 7          return nil, err
 8      }
 9
10      p := log.NewBatchProcessor(exporter)
11      processor := minsev.NewLogProcessor(p, logLevel)
12      lp := log.NewLoggerProvider(
13          log.WithProcessor(processor),
14      )
15
16      global.SetLoggerProvider(lp)
17      return lp, nil
18  }
```

Instrument logs

```go
func main() {
    ctx := context.Background()

    // Previous code ...

    // Setup logger
    lp, err := newLoggerProvider(ctx)
    if err != nil {
        log.Fatalf("failed to setup lp: %v", err)
    }
    defer lp.Shutdown(ctx)

    // Rest of the code ...
```

Instrument logs

```go
package telemetry

import (
  "log/slog"
  "os"
  "time"

  "github.com/lmittmann/tint"
  slogmulti "github.com/samber/slog-multi"
  "go.opentelemetry.io/contrib/bridges/otelslog"
)

func NewLogger() *slog.Logger {
```

Instrument logs

```
12
13 func NewLogger() *slog.Logger {
14     var handler slog.Handler
15     if os.Getenv("ENVIRONMENT") == "local" {
16         stdoutHandler := tint.NewHandler(os.Stdout,
17             &tint.Options{
18                 AddSource:  true,
19                 TimeFormat: time.Kitchen,
20             }
21         )
22         otelHandler := otelslog.NewHandler(
23             "user-service",
24             otelslog.WithSource(true),
25         )
```

Instrument logs

```
17              &tint.Options{
18                  AddSource:  true,
19                  TimeFormat: time.Kitchen,
20              }
21          )
22          otelHandler := otelslog.NewHandler(
23              "user-service",
24              otelslog.WithSource(true),
25          )
26          handler = slogmulti.Fanout(
27              stdoutHandler,
28              otelHandler,
29          )
30      } else {
```

Instrument logs

```
        )
        otelHandler := otelslog.NewHandler(
            "user-service",
            otelslog.WithSource(true),
        )
        handler = slogmulti.Fanout(
            stdoutHandler,
            otelHandler,
        )
    } else {
        handler = otelslog.NewHandler(
            "user-service",
            otelslog.WithSource(true),
        )
```

Instrument logs

```
26          handler = slogmulti.Fanout(
27              stdoutHandler,
28              otelHandler,
29          )
30      } else {
31          handler = otelslog.NewHandler(
32              "user-service",
33              otelslog.WithSource(true),
34          )
35      }
36
37      logger := slog.New(handler)
38      return logger
39  }
```

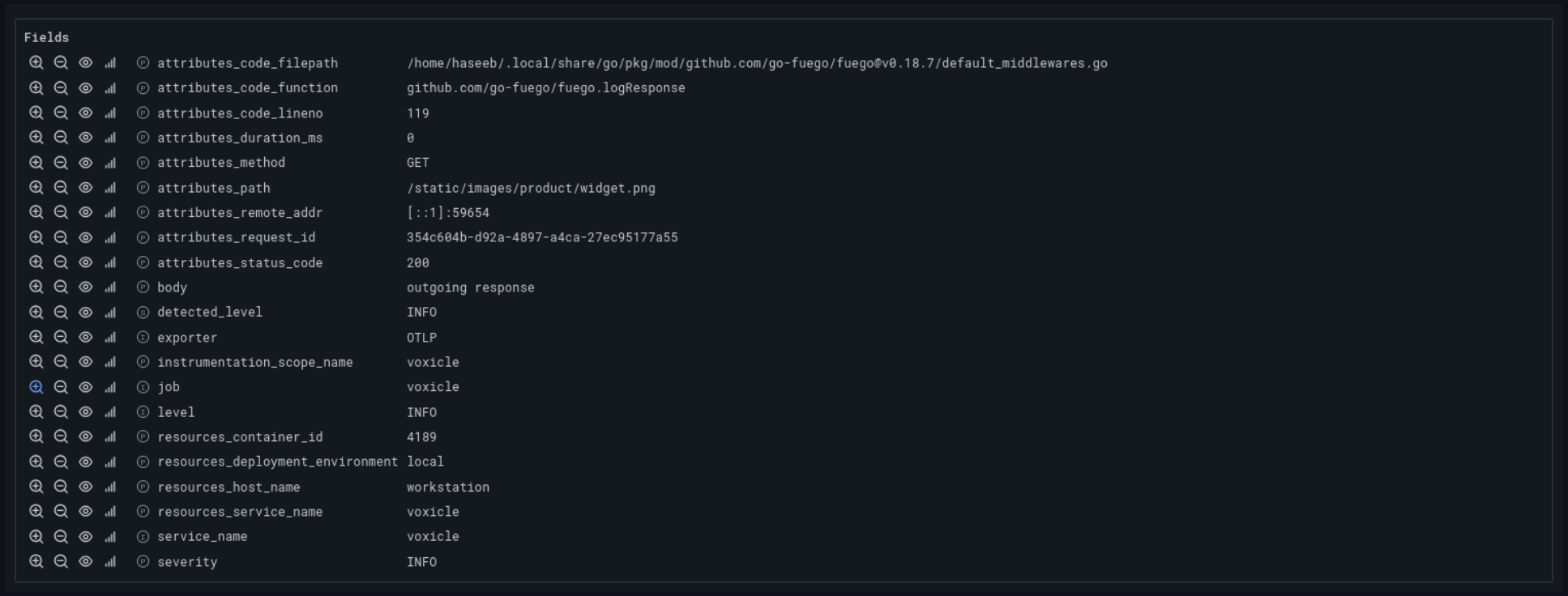
Instrument logs

```
27                stdoutHandler,
28                otelHandler,
29            )
30        } else {
31            handler = otelslog.NewHandler(
32                "user-service",
33                otelslog.WithSource(true),
34            )
35        }
36
37        logger := slog.New(handler)
38        return logger
39    }
```

```
logger.InfoContext(
    ctx,
    "starting server",
    slog.String("host", conf.Server.Host),
    slog.Int("port", conf.Server.Port)
)
```

```
{
  "time": "2023-10-05T12:34:56Z",
  "level": "INFO",
  "msg": "starting server",
  "trace_id": "d4cda95b652f4a1592b449d5929fda1b",
  "span_id": "6e0c63257de34c92",
  "host": "localhost",
  "port": 8080
}
```

```
[0] 10:04PM INF watermill@v1.4.6/slog.go:58 Starting handler topic=subscriptions subscriber_name=subscription-handler
[0] 10:04PM INF server/main.go:246 starting server host=0.0.0.0 port=8080
[0] 10:04PM INF http/handlers.go:178 starting server host=0.0.0.0 port=8080
[0] 10:04PM INF fuego@v0.18.7/engine.go:221 JSON spec: http://0.0.0.0:8080/swagger/openapi.json
[0] 10:04PM INF fuego@v0.18.7/engine.go:221 OpenAPI UI: http://0.0.0.0:8080/swagger/index.html
[0] 10:04PM INF fuego@v0.18.7/serve.go:74 Server running ✅ on http://0.0.0.0:8080 "started in"=1.092766ms
[0] 10:04PM INF fuego@v0.18.7/engine.go:221 JSON file: doc/openapi.json
[1] Done in 3ms
```


Fields
attributes_code_filepath /home/haseeb/.local/share/go/pkg/mod/github.com/go-fuego/fuego@v0.18.7/default_middlewares.go
attributes_code_function github.com/go-fuego/fuego.logResponse
attributes_code_lineno 119
attributes_duration_ms 0
attributes_method GET
attributes_path /static/images/product/widget.png
attributes_remote_addr [::1]:59654
attributes_request_id 354c604b-d92a-4897-a4ca-27ec95177a55
attributes_status_code 200
body outgoing response
detected_level INFO
exporter OTLP
instrumentation_scope_name voxicle
job voxicle
level INFO
resources_container_id 4189
resources_deployment_environment local
resources_host_name workstation
resources_service_name voxicle
service_name voxicle
severity INFO

How many errors occurred?	Metrics	Aggregate counts needed for alerting
Why did request ID:123 fail?	Logs	Requires detailed error context (stack trace, inputs)
Is latency increasing system-wide?	Metrics	Track percentiles (P95/P99) across all requests

Where is the bottleneck?	Traces	Breakdown of time spent across services
Why did payment TX-456 timeout?	Traces	Follow call path: Gateway → Auth → Payment → DB
Why did login for user@x fail?	Logs	Authentication details (wrong password? locked account?)

Resources

- Attributes to include in all OTel data
- Describe the “source” of the telemetry data
  - service name
  - instance id

Resources

- Example: Container in k8s
  - process name
  - namespace
  - deployment name

```
OTEL_RESOURCE_ATTRIBUTES="service.namespace=
                tutorial,service.version=1.0"
```

Resources

```
res, err := resource.New(
    ctx,
    resource.WithHost(),
    resource.WithContainerID(),
    resource.WithAttributes(
        semconv.DeploymentEnvironmentKey.String(
            environment
        ),
        semconv.ServiceNameKey.String("user-service"),
    ),
)
```

Resources

```
res, err := resource.New(
    ctx,
    resource.WithHost(),
    resource.WithContainerID(),
    resource.WithAttributes(
        semconv.DeploymentEnvironmentKey.String(
            environment
        ),
        semconv.ServiceNameKey.String("user-service"),
    ),
)
```

```
loggerProvider := log.NewLoggerProvider(
    log.WithProcessor(processor),
    log.WithResource(res),
)

meterProvider := metric.NewMeterProvider(
    metric.WithReader(reader),
    metric.WithResource(res)
)

traceProvider := trace.NewTracerProvider(
    trace.WithBatcher(traceExporter,
        trace.WithBatchTimeout(time.Second),
```

```
    log.WithProcessor(processor),
    log.WithResource(res),
)

meterProvider := metric.NewMeterProvider(
    metric.WithReader(reader),
    metric.WithResource(res)
)

traceProvider := trace.NewTracerProvider(
    trace.WithBatcher(traceExporter,
        trace.WithBatchTimeout(time.Second),
    ),
```

```
 4 )
 5
 6 meterProvider := metric.NewMeterProvider(
 7     metric.WithReader(reader),
 8     metric.WithResource(res)
 9 )
10
11 traceProvider := trace.NewTracerProvider(
12     trace.WithBatcher(traceExporter,
13         trace.WithBatchTimeout(time.Second),
14     ),
15     trace.WithResource(res),
16 )
```

semconv

- Semantic Convention
- Common attributes
  - Across languages and tools

```
attrs := []attribute.KeyValue{
    semconv.HTTPRequestMethodKey.String(r.Method),
    semconv.HTTPResponseStatusCode(statusCode),
    semconv.HTTPRoute(r.URL.EscapedPath()),
}
```

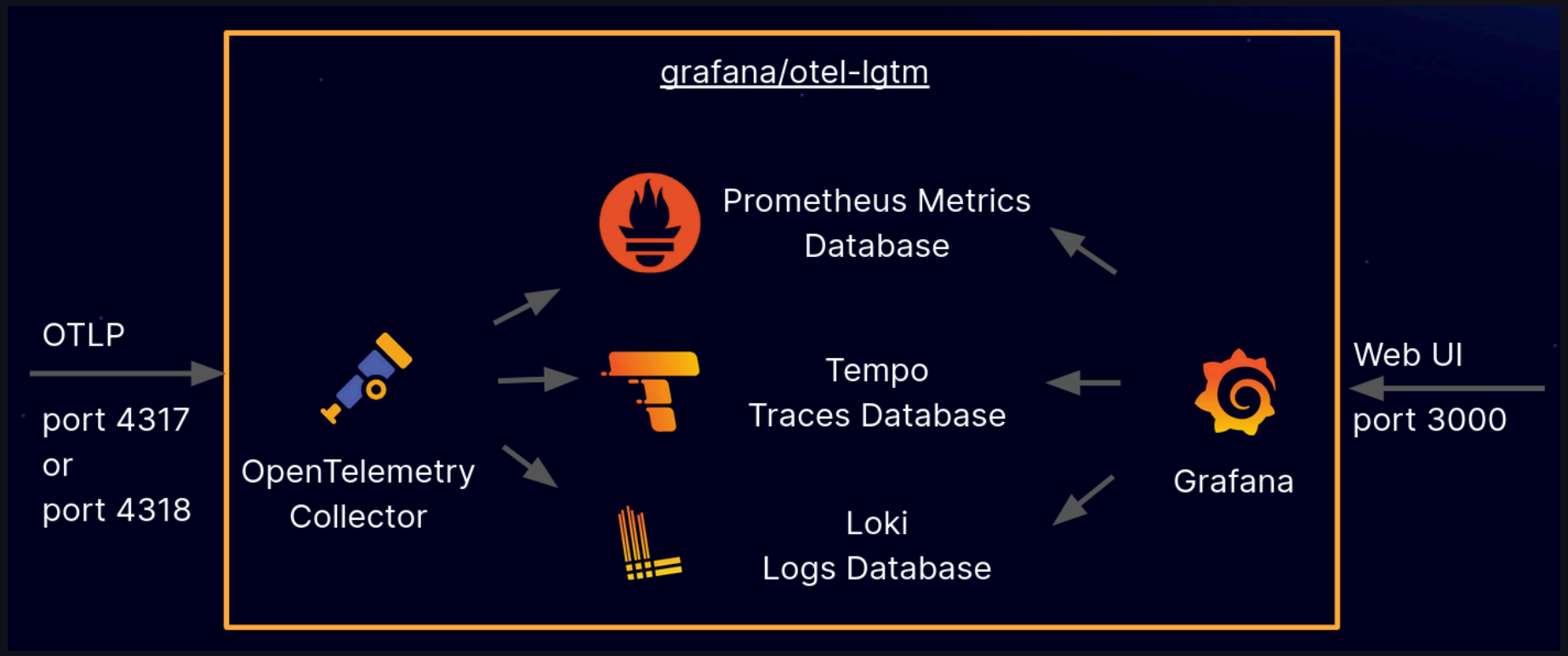
```
http.request.method
http.response.status_code
http.route
```

Anti-Patterns

- Emitting a log for every HTTP request (use metrics instead).
- Trying to capture a user ID in a metric (use logs/traces).
- Logs without TraceId (use OTel context propagation).

LGTM Stack

- Loki: Logs
- Grafana: Visualisation
- Tempo: Traces
- Mimir: Metrics


grafana/otel-lgtm
Prometheus Metrics
Database
OTLP
port 4317
or
port 4318
OpenTelemetry
Collector
Tempo
Traces Database
Loki
Logs Database
Grafana
Web UI
port 3000

docker-compose.yml

```yaml
services:
  alloy:
    image: grafana/alloy:v1.9.1
    profiles:
      - monitoring
    command:
      [
        "run",
        "--server.http.listen-addr=0.0.0.0:12345",
        "--storage.path=/var/lib/alloy/data",
        "/etc/alloy/config.alloy",
      ]
    ports:
```

docker-compose.yml

```
12        ]
13      ports:
14        - 4317:4317
15        - 4318:4318
16        - 12345:12345
17      volumes:
18        - ./docker/config.alloy:/etc/alloy/config.alloy
19      depends_on:
20        - tempo
21        - loki
22        - mimir
23
24    mimir:
```

docker-compose.yml

```
23
24    mimir:
25      image: grafana/mimir:2.11.0
26      profiles:
27        - monitoring
28      command:
29        - "-auth.multitenancy-enabled=false"
30        - "-auth.no-auth-tenant=anonymous"
31        - "-config.file=/etc/mimir/config.yaml"
32      volumes:
33        - ./docker/mimir.yaml:/etc/mimir/config.yaml
34        - mimir-data:/data
35
```

docker-compose.yml

```
35
36    grafana:
37      image: grafana/grafana:11.6.1
38      profiles:
39        - monitoring
40      ports:
41        - 3000:3000
42      environment:
43        - GF_AUTH_ANONYMOUS_ENABLED=true
44        - GF_AUTH_ANONYMOUS_ORG_ROLE=Admin
45      volumes:
46       - ./docker/grafana/datasources.yaml:/etc/grafana/p
47       - grafana-data:/var/lib/grafana
48
```

docker-compose.yml

```
46      - ./docker/grafana/datasources.yaml:/etc/grafana/p
47      - grafana-data:/var/lib/grafana
48
49    tempo:
50      image: grafana/tempo:2.7.2
51      profiles:
52        - monitoring
53      command: ["-config.file=/etc/tempo.yaml"]
54      volumes:
55        - ./docker/tempo.yaml:/etc/tempo.yaml
56        - tempo-data:/var/tempo
57
58    loki:
59      image: grafana/loki:3.5.0
```

docker-compose.yml

```
55        - ./docker/tempo.yaml:/etc/tempo.yaml
56        - tempo-data:/var/tempo
57
58    loki:
59      image: grafana/loki:3.5.0
60      profiles:
61        - monitoring
62      command: ["-config.file=/etc/loki/loki-config.yaml
63      volumes:
64        - ./docker/loki.yaml:/etc/loki/loki-config.yaml
65        - loki-data:/loki
66
67  volumes:
68    grafana-data:
```

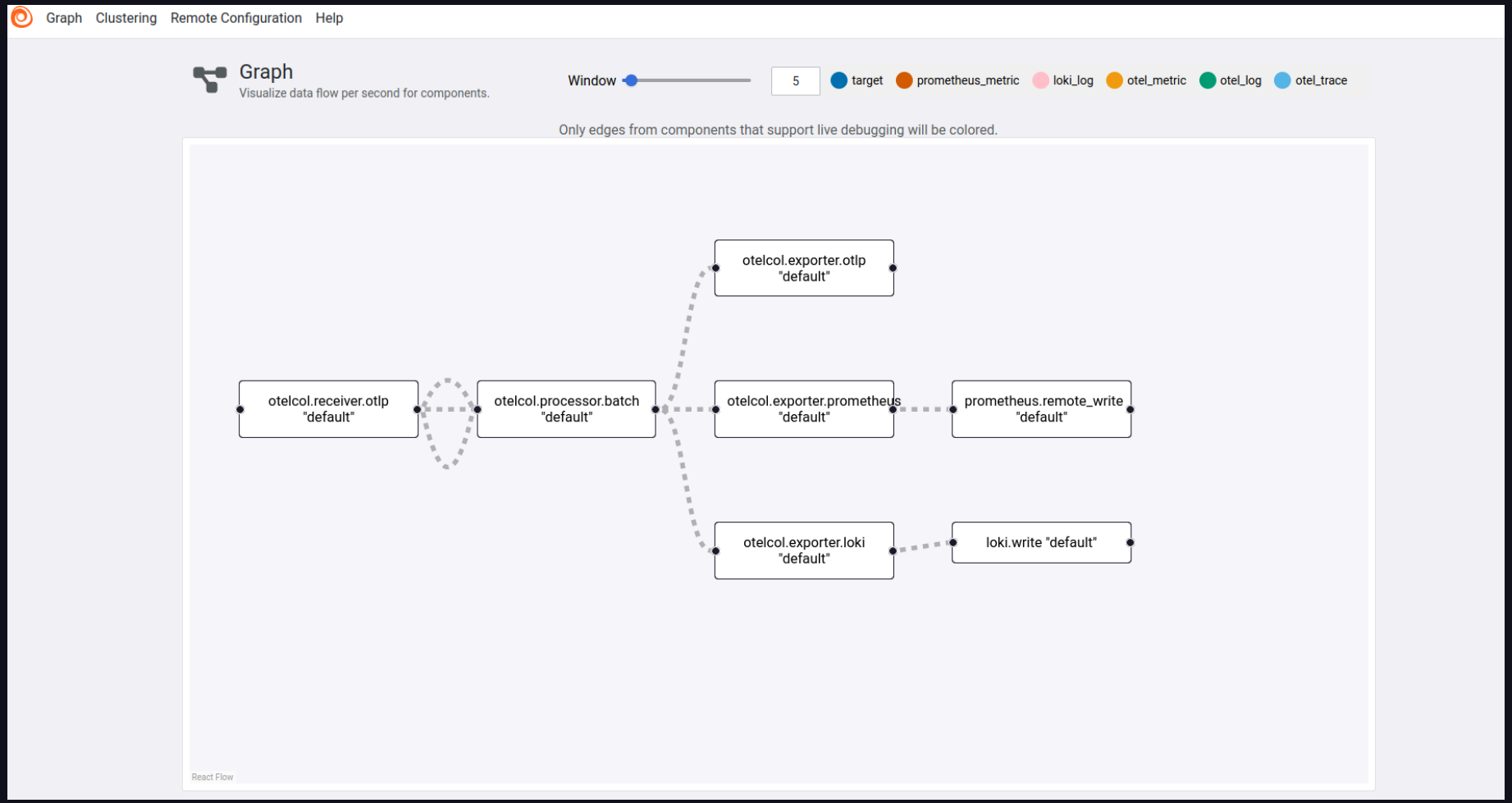
docker-compose.yml

```
59      image: grafana/loki:3.5.0
60      profiles:
61        - monitoring
62      command: ["-config.file=/etc/loki/loki-config.yaml
63      volumes:
64        - ./docker/loki.yaml:/etc/loki/loki-config.yaml
65        - loki-data:/loki
66
67  volumes:
68    grafana-data:
69    mimir-data:
70    tempo-data:
71    loki-data:
```



OTEL COLLECTOR

Alloy


Graph
Clustering
Remote Configuration
Help
Graph
Visualize data flow per second for components.
Window
5
target
prometheus_metric
loki_log
otel_metric
otel_log
otel_trace
Only edges from components that support live debugging will be colored.
otelcol.exporter.otlp "default"
otelcol.receiver.otlp "default"
otelcol.processor.batch "default"
otelcol.exporter.prometheus "default"
prometheus.remote_write "default"
otelcol.exporter.loki "default"
loki.write "default"
React Flow

Setup Grafana

```yaml
apiVersion: 1
datasources:
  - name: Prometheus
    uid: "prometheus"
    type: prometheus
    access: proxy
    url: http://mimir:9009/prometheus
    isDefault: true
    jsonData:
      httpMethod: POST
      exemplarTraceIdDestinations:
        - name: trace_id
          datasourceUid: "tempo"
```

Setup Grafana

```
 5     type: prometheus
 6     access: proxy
 7     url: http://mimir:9009/prometheus
 8     isDefault: true
 9     jsonData:
10       httpMethod: POST
11       exemplarTraceIdDestinations:
12         - name: trace_id
13           datasourceUid: "tempo"
14           url: '${__value.raw}'
15
16   - name: Loki
17     uid: "loki"
18     type: loki
```

Setup Grafana

```
12          - name: trace_id
13            datasourceUid: "tempo"
14            url: '${__value.raw}'
15
16      - name: Loki
17        uid: "loki"
18        type: loki
19        access: proxy
20        url: http://loki:3100
21        isDefault: false
22        jsonData:
23          derivedFields:
24            - name: "🔍 View Trace"
25              matcherRegex: '"traceid":\s*"([0-9a-f]{32})"
```

Setup Grafana

```
19      access: proxy
20      url: http://loki:3100
21      isDefault: false
22      jsonData:
23        derivedFields:
24          - name: "🔍 View Trace"
25            matcherRegex: '"traceid":\s*"([0-9a-f]{32})"
26            url: '$${__value.raw}'
27            datasourceUid: "tempo"
28            matcherType: regex
29
30    - name: Tempo
31      uid: "tempo"
```

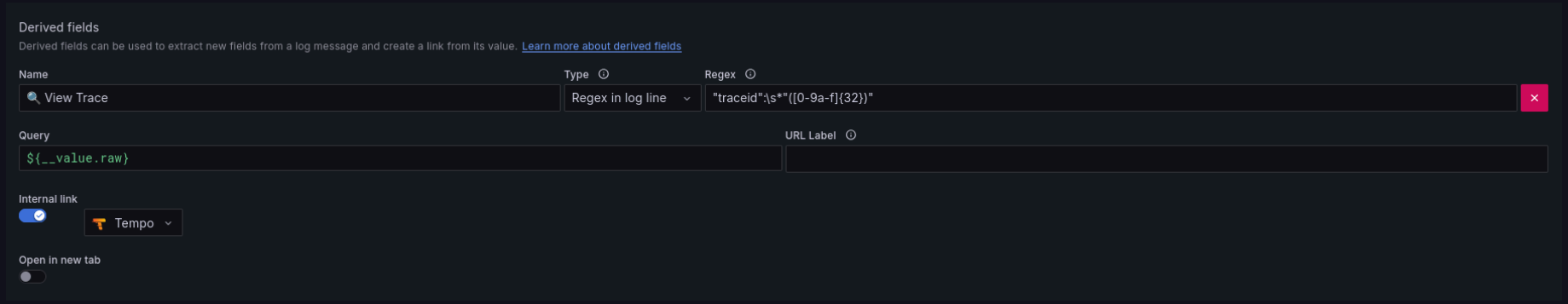
Setup Grafana

```
27            datasourceUid: "tempo"
28            matcherType: regex
29
30    - name: Tempo
31      uid: "tempo"
32      type: tempo
33      access: proxy
34      url: http://tempo:3200
35      isDefault: false
36      jsonData:
37        nodeGraph:
38          enabled: true
39        tracesToLogs:
```

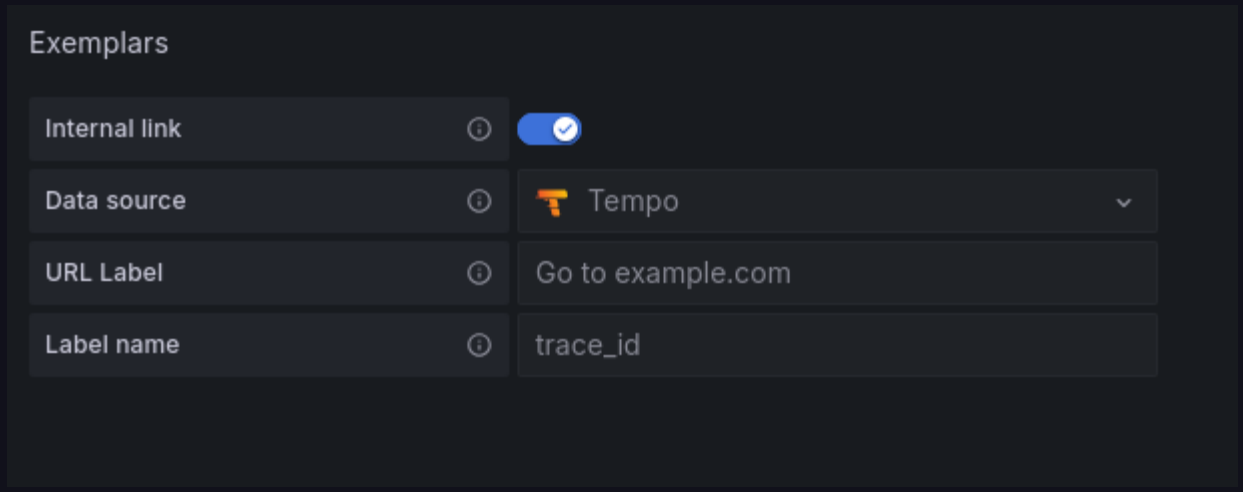
Setup Grafana

```
33      access: proxy
34      url: http://tempo:3200
35      isDefault: false
36      jsonData:
37        nodeGraph:
38          enabled: true
39        tracesToLogs:
40          datasourceUid: "loki"
41          spanStartTimeShift: '-5m'
42          spanEndTimeShift: '5m'
43          filterByTraceID: true
44        tracesToMetrics:
45          datasourceUid: "prometheus"
```

Logs


Derived fields
Derived fields can be used to extract new fields from a log message and create a link from its value. Learn more about derived fields
Name
Type
Regex
View Trace
Regex in log line
"traceid":\s*"([0-9a-f]{32})"
Query
URL Label
${__value.raw}
Internal link
Tempo
Open in new tab

Metrics


Exemplars
Internal link
Data source
Tempo
URL Label
Go to example.com
Label name
trace_id

Exemplars

- OTel context to a metric
  - Connect to a trace

Exemplar Modal

- The trace context
- The time of the observation

Exemplar Modal (Cont...)

- The recorded value
- A set of filtered attributes
  - additional insight into the Context

OTLP Export

```
metrics {
  name: "http_server_duration_milliseconds_seconds"
  histogram {
    data_points {
      buckets {
        count: 12
        exemplars {
          value: 0.382
          time_unix_nano: 1678901234000000000
          // Base64-encoded span ID
          span_id: "APBnoLoJArc="
          // Base64-encoded trace ID
          trace_id: "S/kyNXezTaajzpKdDg5HNg=="
```

OTLP Export

```
      buckets {
        count: 12
        exemplars {
          value: 0.382
          time_unix_nano: 1678901234000000000
          // Base64-encoded span ID
          span_id: "APBnoLoJArc="
          // Base64-encoded trace ID
          trace_id: "S/kyNXezTaajzpKdDg5HNg=="
        }
      }
      explicit_bounds: [0.5, 1, 2]
    }
  }
```

OTLP Export

```
 7           exemplars {
 8             value: 0.382
 9             time_unix_nano: 1678901234000000000
10             // Base64-encoded span ID
11             span_id: "APBnoLoJArc="
12             // Base64-encoded trace ID
13             trace_id: "S/kyNXezTaajzpKdDg5HNg=="
14           }
15         }
16         explicit_bounds: [0.5, 1, 2]
17       }
18     }
19 }
```

Traces

Trace to logs

Navigate from a trace span to the selected data source's logs. Learn more about trace to logs

Data source	Loki
Span start time shift	-5m
Span end time shift	5m
Tags	+
Filter by trace ID	on
Filter by span ID	off
Use custom query	off

Trace to metrics

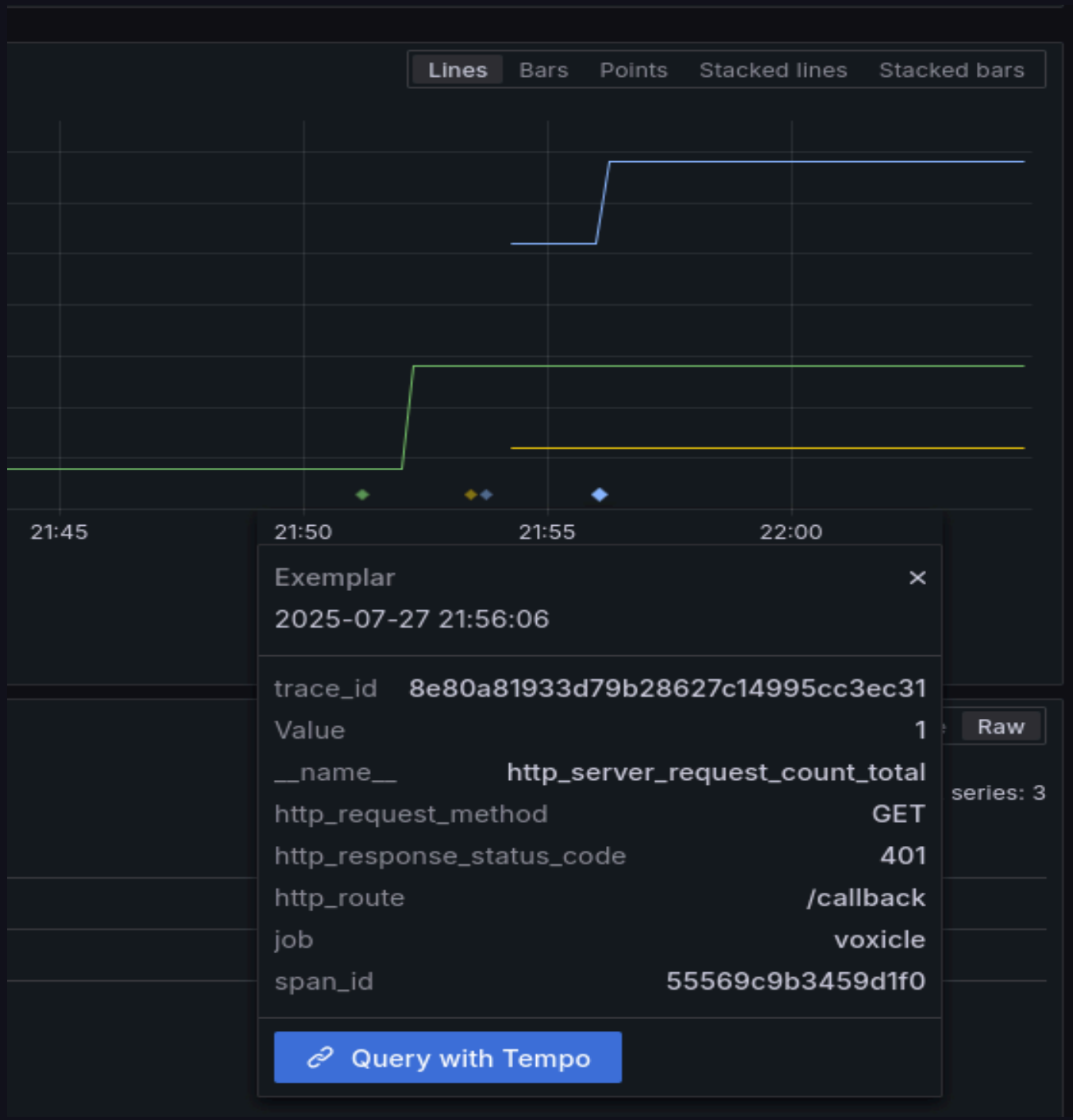
Navigate from a trace span to the selected data source's metrics. Learn more about trace to metrics

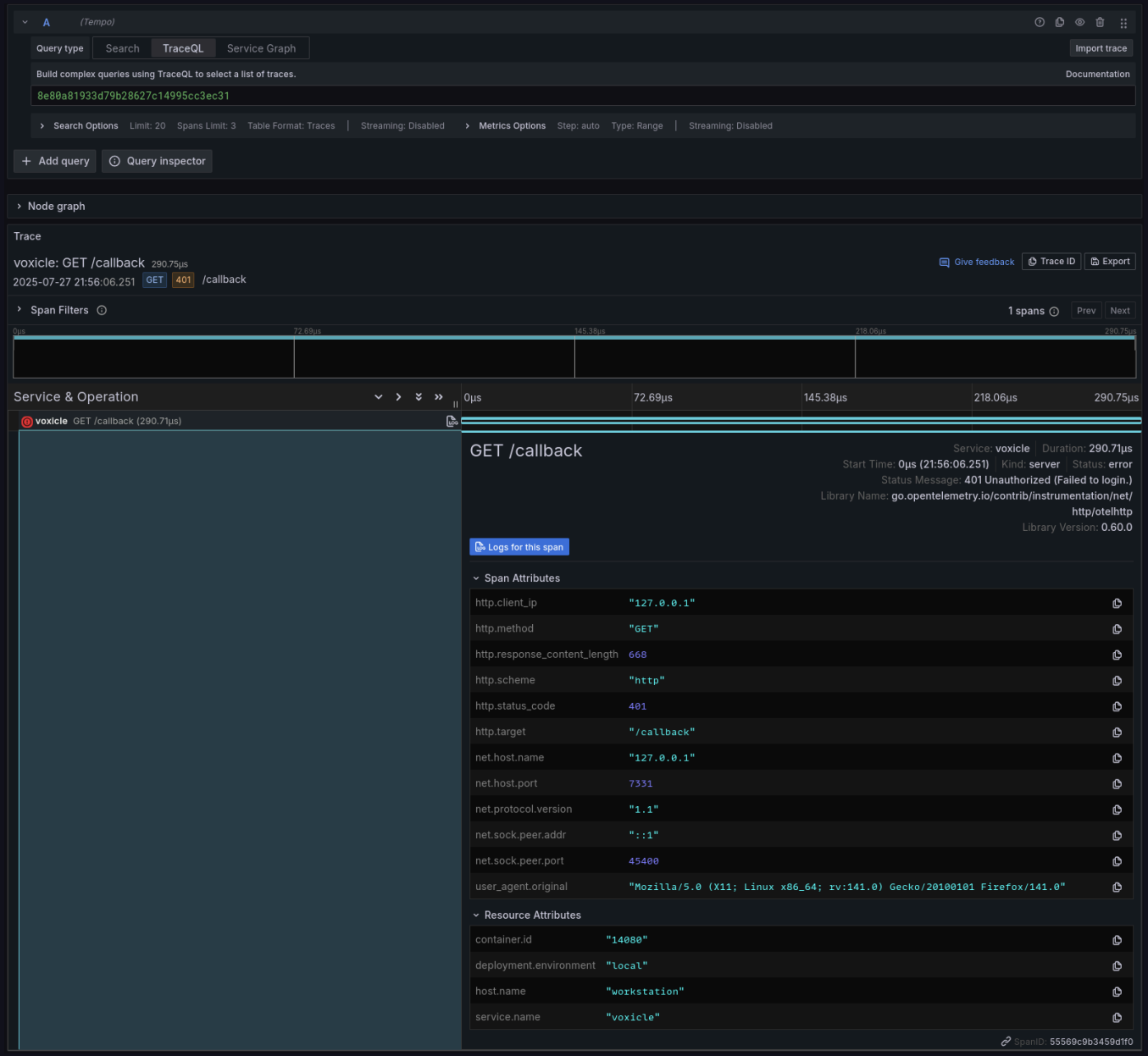
Data source	Prometheus
Span start time shift	-2m
Span end time shift	2m
Tags	+

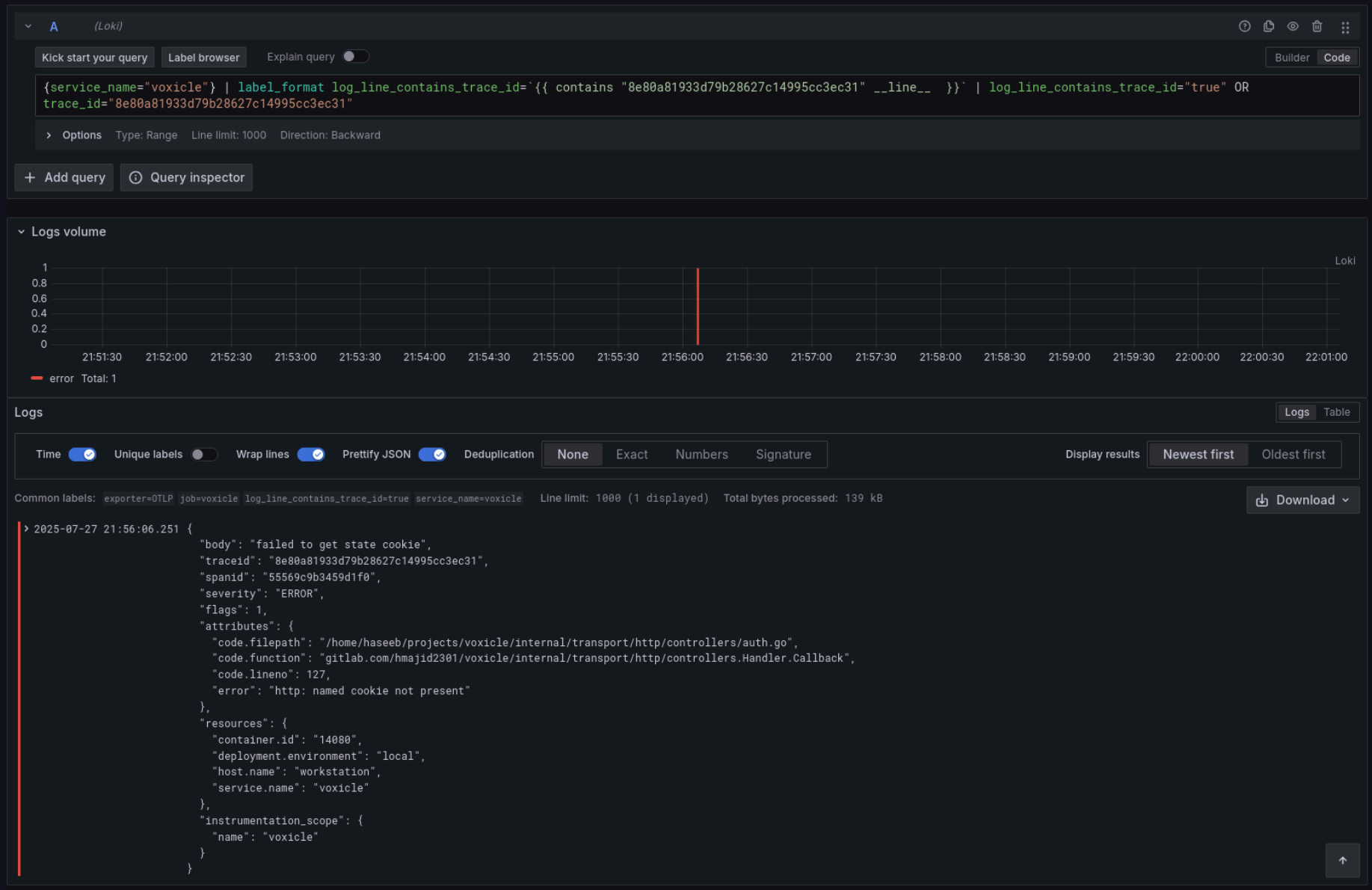
+ Add query

Viewing an error




Lines
Bars
Points
Stacked lines
Stacked bars
21:45
21:50
21:55
22:00
Exemplar
2025-07-27 21:56:06
trace_id 8e80a81933d79b28627c14995cc3ec31
Value 1
__name__ http_server_request_count_total
http_request_method GET
http_response_status_code 401
http_route /callback
job voxicle
span_id 55569c9b3459d1f0
Query with Tempo
Raw
series: 3


A (Tempo)
Query type
Search
TraceQL
Service Graph
Import trace
Build complex queries using TraceQL to select a list of traces.
Documentation
8e80a81933d79b28627c14995cc3ec31
Search Options Limit: 20 Spans Limit: 3 Table Format: Traces | Streaming: Disabled
Metrics Options Step: auto Type: Range | Streaming: Disabled
Add query
Query inspector
Node graph
Trace
voxicle: GET /callback 290.75µs
2025-07-27 21:56:06.251 GET 401 /callback
Give feedback
Trace ID
Export
Span Filters
1 spans
Prev
Next
0µs
72.69µs
145.38µs
218.06µs
290.75µs
Service & Operation
voxicle GET /callback (290.71µs)
GET /callback
Service: voxicle | Duration: 290.71µs
Start Time: 0µs (21:56:06.251) | Kind: server | Status: error
Status Message: 401 Unauthorized (Failed to login.)
Library Name: go.opentelemetry.io/contrib/instrumentation/net/http/otelhttp
Library Version: 0.60.0
Logs for this span
Span Attributes
http.client_ip "127.0.0.1"
http.method "GET"
http.response_content_length 668
http.scheme "http"
http.status_code 401
http.target "/callback"
net.host.name "127.0.0.1"
net.host.port 7331
net.protocol.version "1.1"
net.sock.peer.addr "::1"
net.sock.peer.port 45400
user_agent.original "Mozilla/5.0 (X11; Linux x86_64; rv:141.0) Gecko/20100101 Firefox/141.0"
Resource Attributes
container.id "14080"
deployment.environment "local"
host.name "workstation"
service.name "voxicle"
SpanID: 55569c9b3459d1f0


A (Loki)
Kick start your query
Label browser
Explain query
Builder
Code
{service_name="voxicle"} | label_format log_line_contains_trace_id=`{{ contains "8e80a81933d79b28627c14995cc3ec31" __line__ }}` | log_line_contains_trace_id="true" OR trace_id="8e80a81933d79b28627c14995cc3ec31"
Options Type: Range Line limit: 1000 Direction: Backward
Add query
Query inspector
Logs volume
Loki
1
0.8
0.6
0.4
0.2
0
21:51:30 21:52:00 21:52:30 21:53:00 21:53:30 21:54:00 21:54:30 21:55:00 21:55:30 21:56:00 21:56:30 21:57:00 21:57:30 21:58:00 21:58:30 21:59:00 21:59:30 22:00:00 22:00:30 22:01:00
error Total: 1
Logs
Logs
Table
Time
Unique labels
Wrap lines
Prettify JSON
Deduplication
None
Exact
Numbers
Signature
Display results
Newest first
Oldest first
Common labels: exporter=OTLP job=voxicle log_line_contains_trace_id=true service_name=voxicle
Line limit: 1000 (1 displayed)
Total bytes processed: 139 kB
Download
2025-07-27 21:56:06.251 {
"body": "failed to get state cookie",
"traceid": "8e80a81933d79b28627c14995cc3ec31",
"spanid": "55569c9b3459d1f0",
"severity": "ERROR",
"flags": 1,
"attributes": {
"code.filepath": "/home/haseeb/projects/voxicle/internal/transport/http/controllers/auth.go",
"code.function": "gitlab.com/hmajid2301/voxicle/internal/transport/http/controllers.Handler.Callback",
"code.lineno": 127,
"error": "http: named cookie not present"
},
"resources": {
"container.id": "14080",
"deployment.environment": "local",
"host.name": "workstation",
"service.name": "voxicle"
},
"instrumentation_scope": {
"name": "voxicle"
}
}





Slides:
https://haseebmajid.dev/slides/gophercon-otel/

Useful Links

- Observability vs Monitoring: https://www.youtube.com/watch?v=ytx6jr2TyxI
- OpenTelemetry Collector - Explanation & Setup: https://www.youtube.com/watch?v=4ACQhkJBKyA
- Sick of Digging Through Logs? Find Errors FAST with OpenTelemetry and Go!: https://www.youtube.com/watch?v=gOag7kWHJ5c

- OTel Metrics Spec: https://opentelemetry.io/docs/specs/otel/met
- Traces vs Spans: https://www.youtube.com/watch?v=A-nbuPug