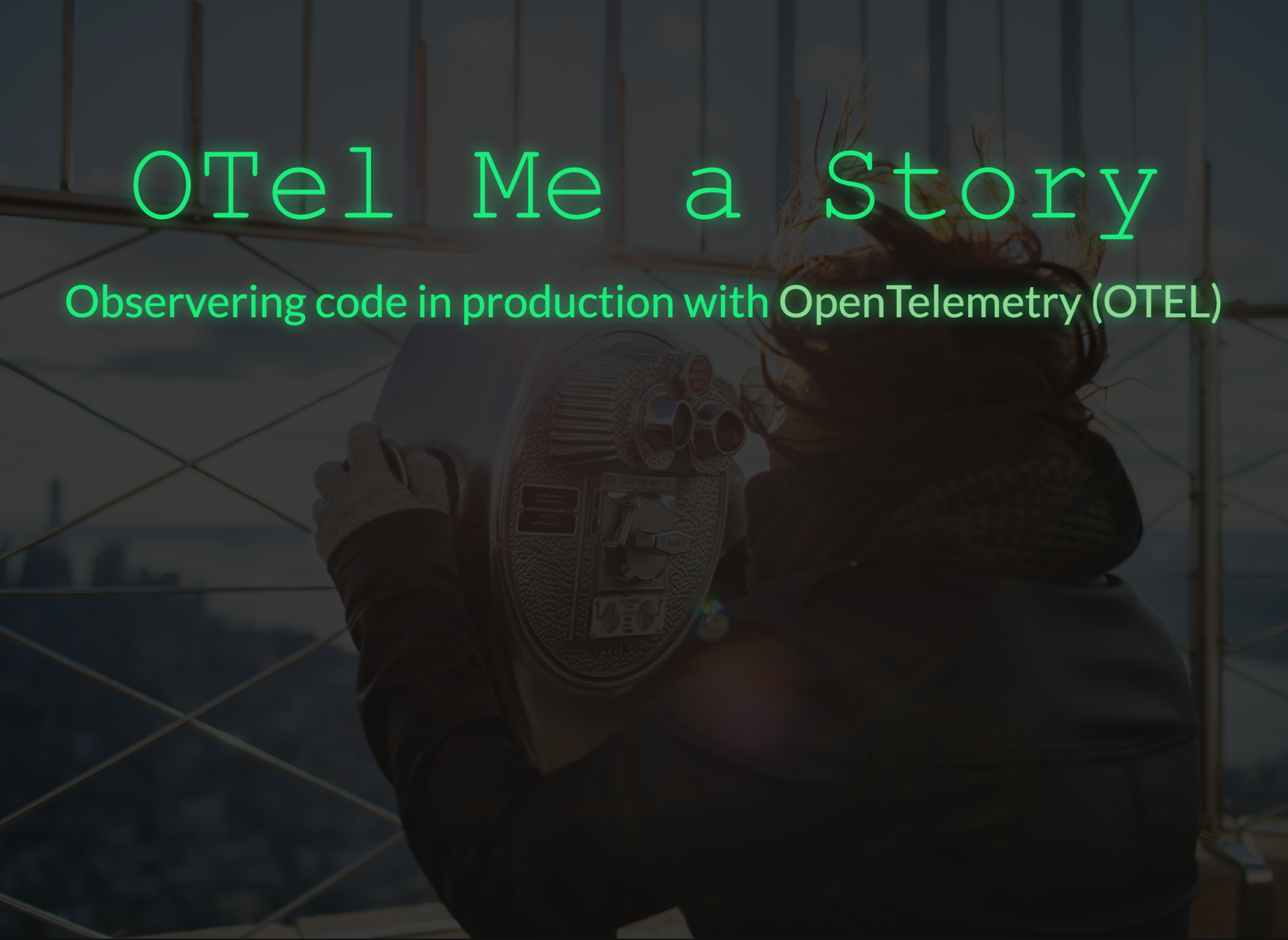
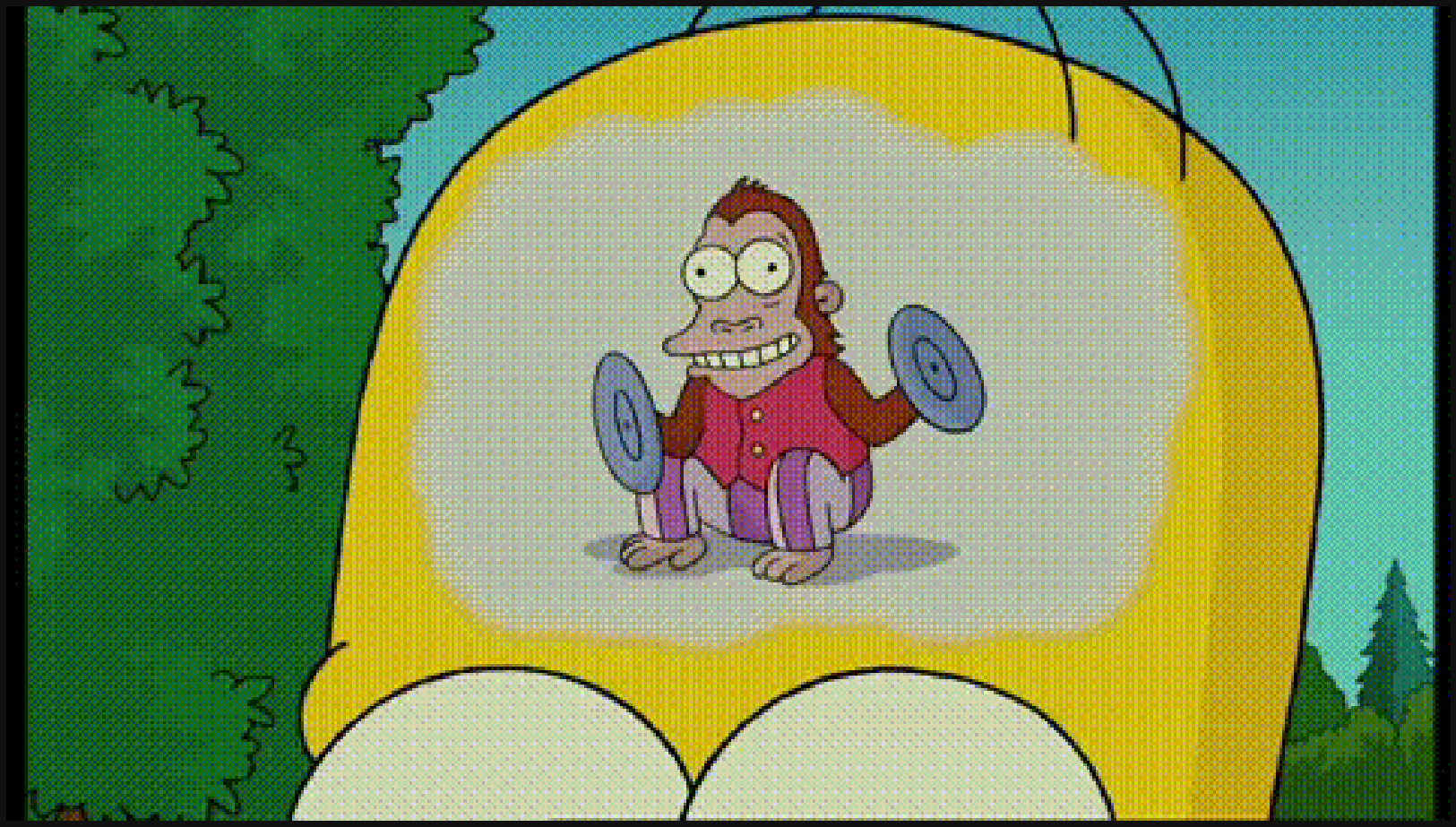


OTel Me a Story

Observering code in production with OpenTelemetry (OTEL)

```python
while True:
    try:
        x = int(input("Enter a number: "))
        break
    except ValueError:
        print("Oops!")
```



Why Intuitive TroubleShooting has Stopped Working for You

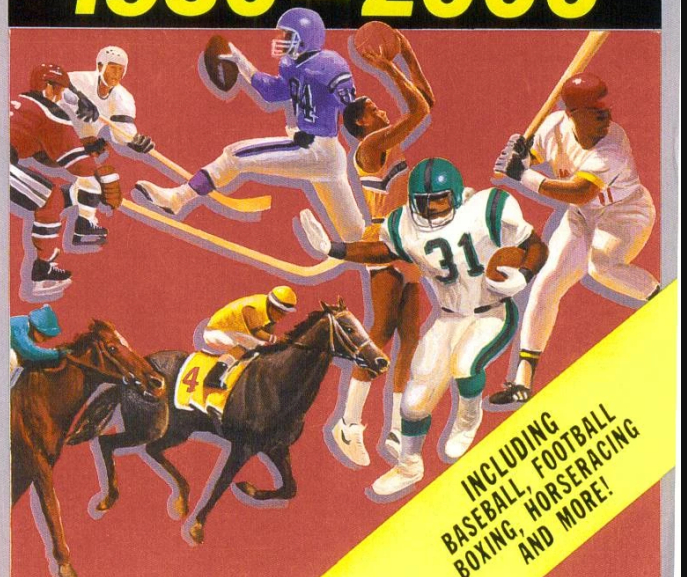
BTTF2: Old Biff Gives The Grays Sports Almanac to His 1955-Self

★★★GRAYS★★★

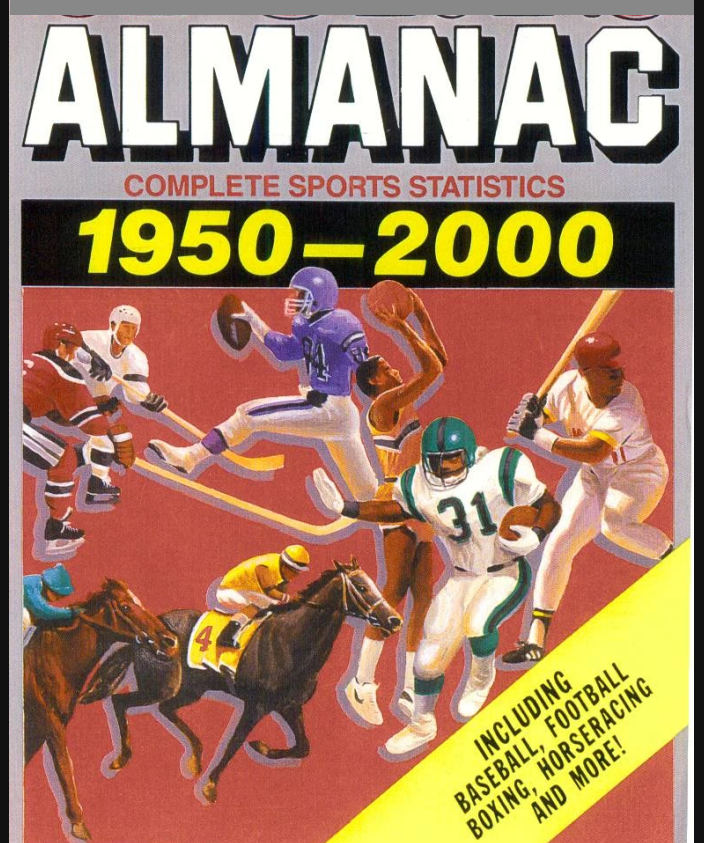
SPORTS ALMANAC

COMPLETE SPORTS STATISTICS

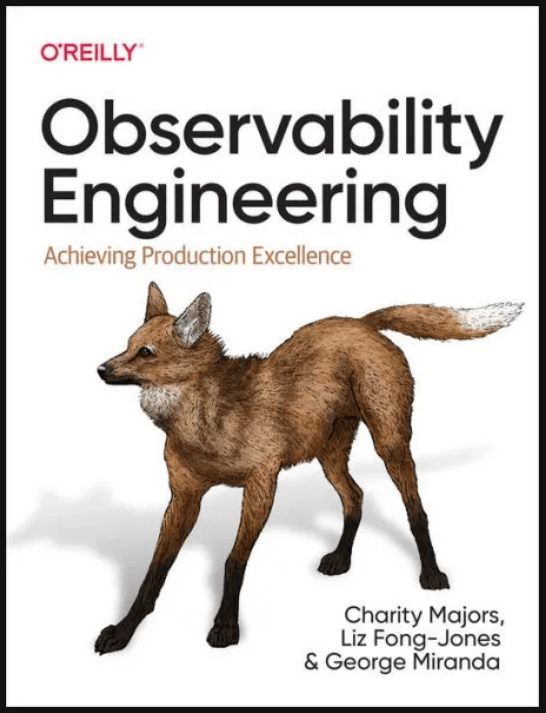
1950–2000

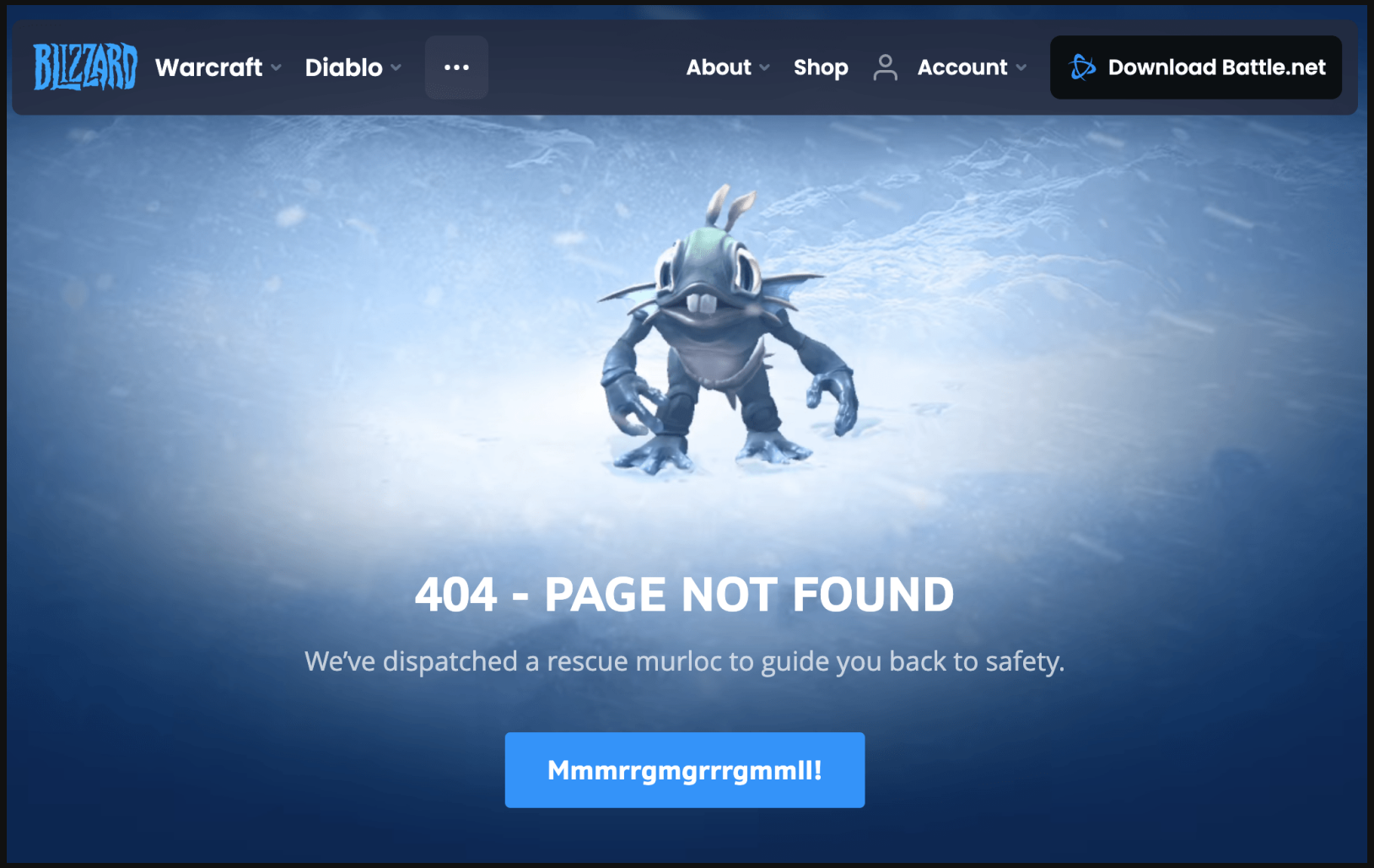


Observability



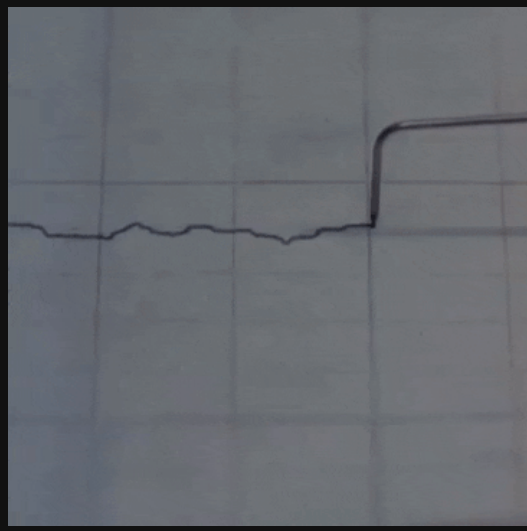
Observability





www.blizzard.com/en-us/404

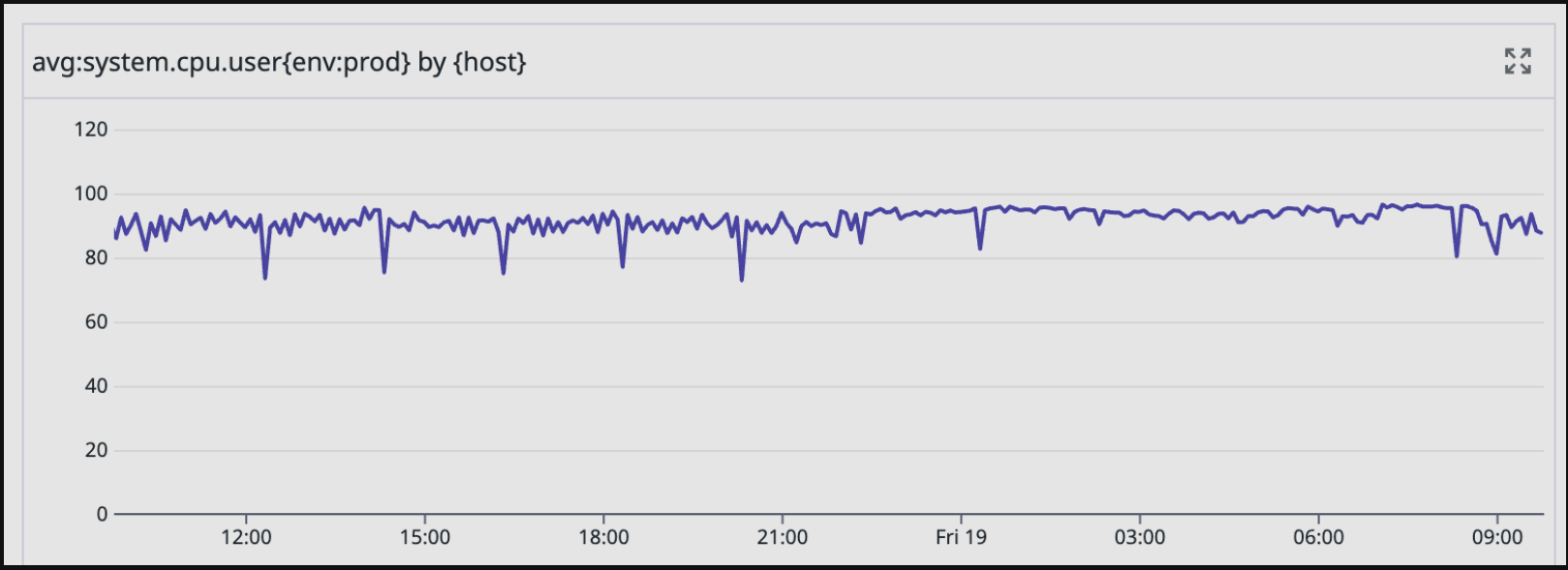
Metrics, Logs and Traces



Metrics

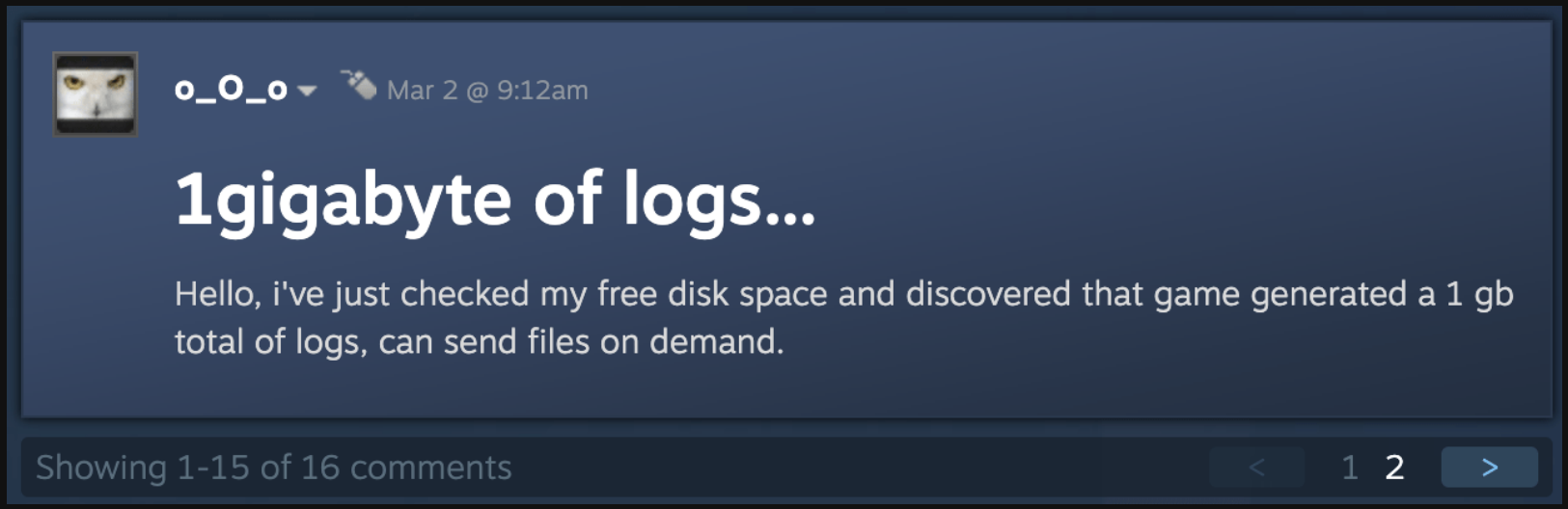
Photo by Altered Vision on Unsplash

Metrics

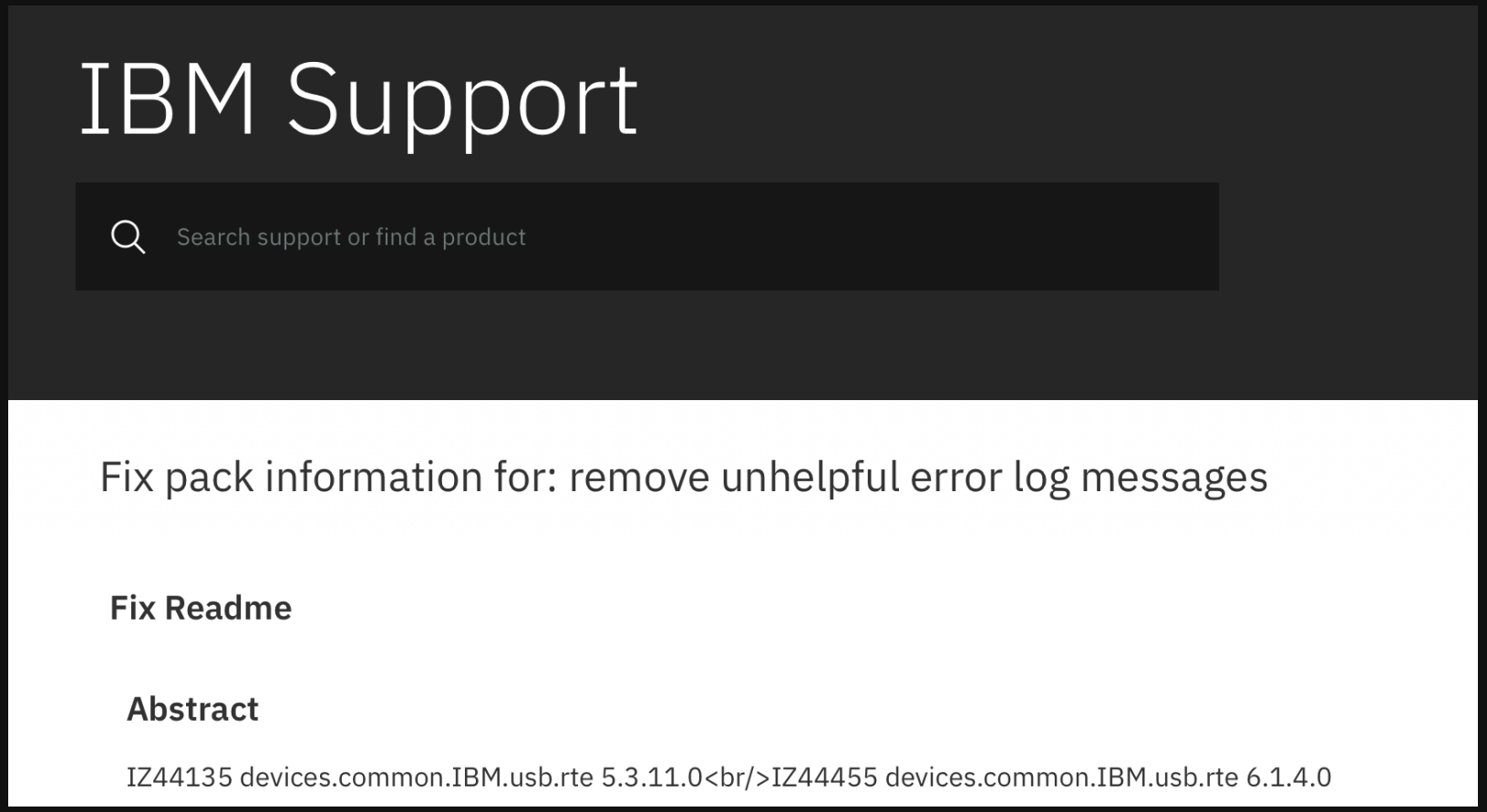

avg:system.cpu.user{env:prod} by {host}
120
100
80
60
40
20
0
12:00
15:00
18:00
21:00
Fri 19
03:00
06:00
09:00


Logs
Photo by Oliver Paaske on Unsplash

Logs


o_O_o
Mar 2 @ 9:12am
1gigabyte of logs...
Hello, i've just checked my free disk space and discovered that game generated a 1 gb total of logs, can send files on demand.
Showing 1-15 of 16 comments
1 2

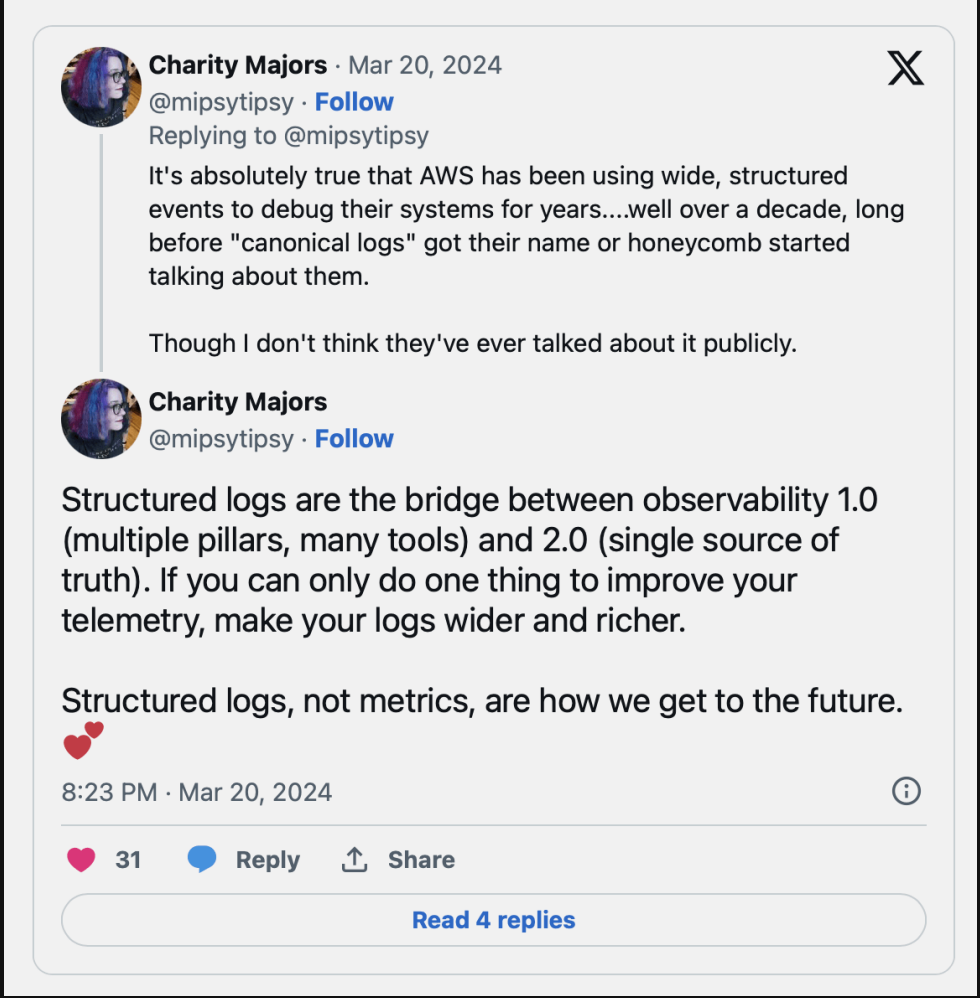
Logs


IBM Support
Search support or find a product
Fix pack information for: remove unhelpful error log messages
Fix Readme
Abstract
IZ44135 devices.common.IBM.usb.rte 5.3.11.0
IZ44455 devices.common.IBM.usb.rte 6.1.4.0

Logs

```
E0419 06:59:50.681167       1 reflector.go:138] pkg/mod/k8s.io/client-
go@v0.22.4/tools/cache/reflector.go:167: Failed to watch *v1.Endpoints: the server has asked for
the client to provide credentials (get endpoints)

time="2024-04-17T20:10:41Z" level=info msg="successfully synced configuration to kong."
subsystem=proxy-cache-resolver
```

Logs



Charity Majors · Mar 20, 2024
@mipsytipsy · **Follow**
Replying to @mipsytipsy

It's absolutely true that AWS has been using wide, structured events to debug their systems for years....well over a decade, long before "canonical logs" got their name or honeycomb started talking about them.

Though I don't think they've ever talked about it publicly.

Charity Majors
@mipsytipsy · **Follow**

Structured logs are the bridge between observability 1.0 (multiple pillars, many tools) and 2.0 (single source of truth). If you can only do one thing to improve your telemetry, make your logs wider and richer.

Structured logs, not metrics, are how we get to the future. 💕

8:23 PM · Mar 20, 2024

31 Reply Share

Read 4 replies

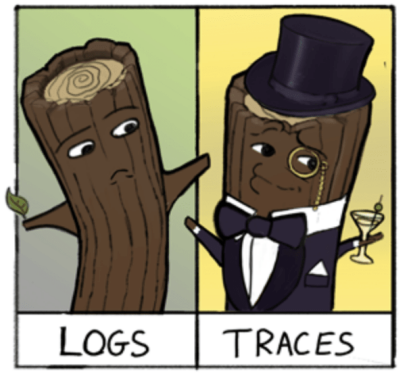
Logs

Where do logs fit in?

My honest opinion is that they don't. Logs relate to point-in-time data, and as such, lack the context we get from traces. **In almost any use case for logs, a span would be a better construct as it has more data and more utility.**

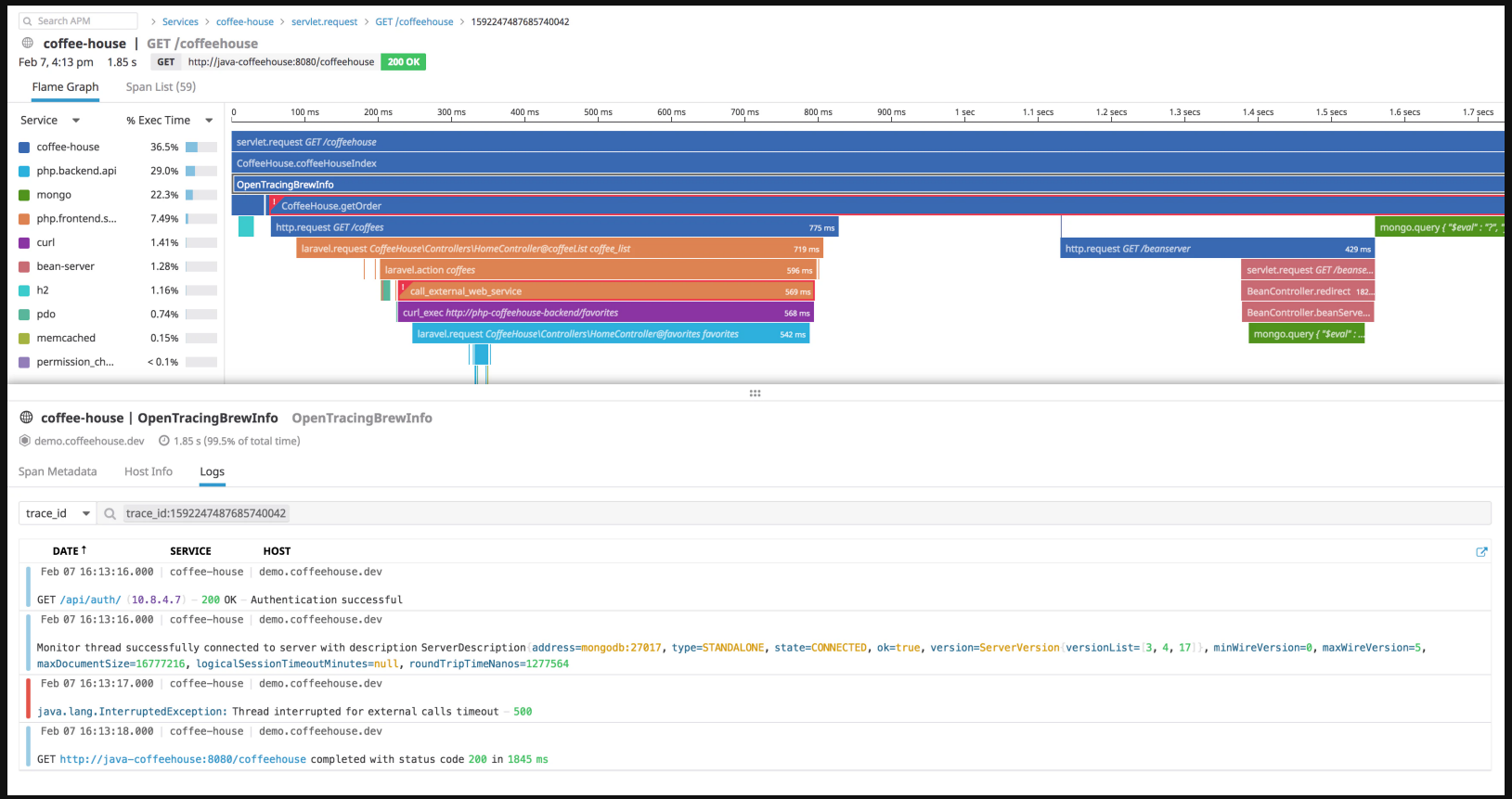
If you do have point-in-time events, these are better represented as span events than logs. This means that they're part of the trace, and inherently, you have all the surrounding context. A good example of this is exceptions in code, as they happen at a point inside a span, and have contextual data like the message, stack trace, etc. Keeping exceptions inside the trace, as events that are queryable, means that when you're looking at them you have all the surrounding context to understand the entire flow that led to—and followed—the error.

Essentially, logs are just spans without context—or said another way, spans are just *fancy logs*.



Ask Miss O11y: To Metric or to Trace?

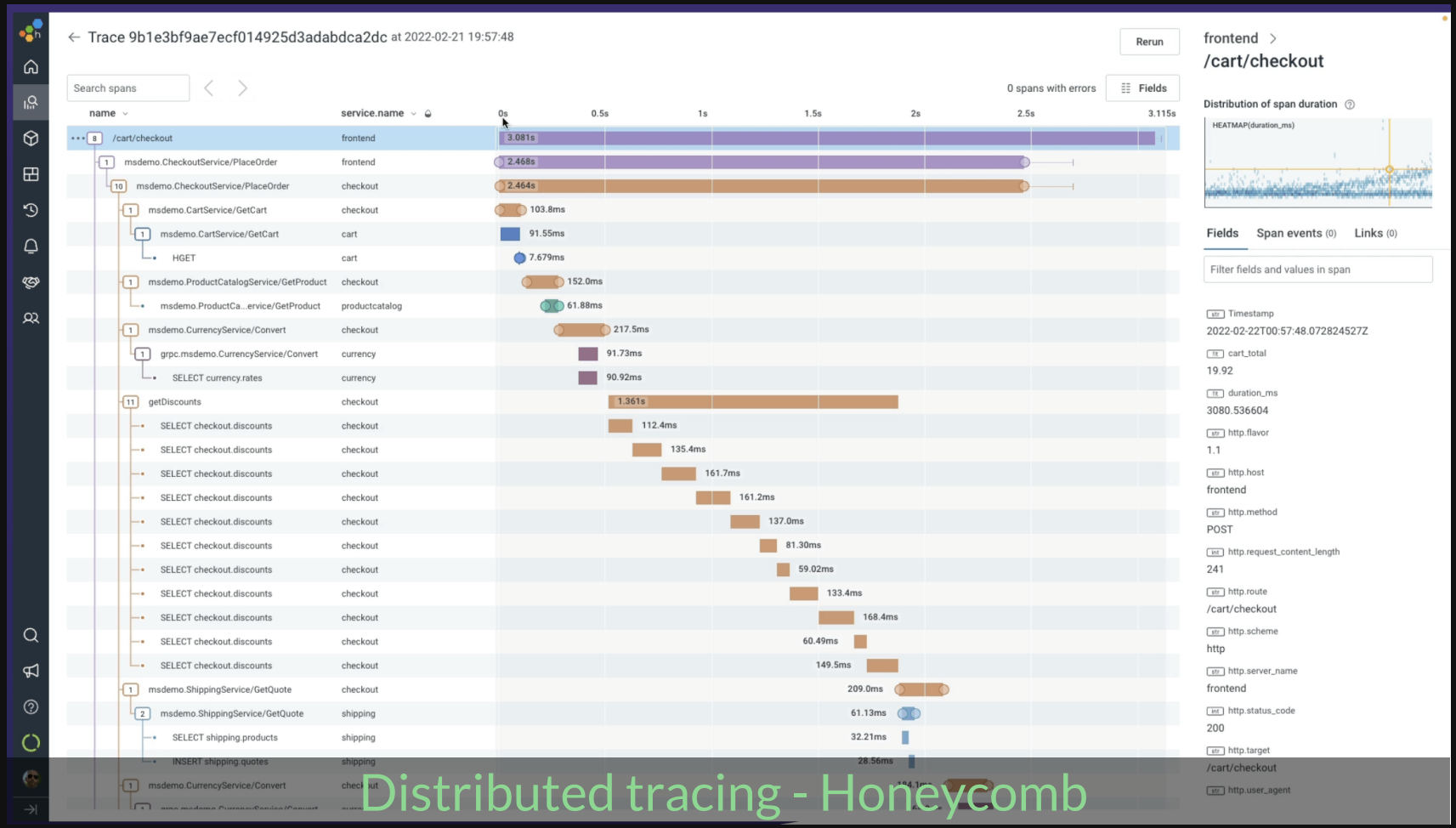
Traces


Search APM
Services > coffee-house > servlet.request > GET /coffeehouse > 1592247487685740042
coffee-house | GET /coffeehouse
Feb 7, 4:13 pm 1.85 s GET http://java-coffeehouse:8080/coffeehouse 200 OK
Flame Graph
Span List (59)
Service
% Exec Time
coffee-house 36.5%
php.backend.api 29.0%
mongo 22.3%
php.frontend.s... 7.49%
curl 1.41%
bean-server 1.28%
h2 1.16%
pdo 0.74%
memcached 0.15%
permission_ch... < 0.1%
0 100 ms 200 ms 300 ms 400 ms 500 ms 600 ms 700 ms 800 ms 900 ms 1 sec 1.1 secs 1.2 secs 1.3 secs 1.4 secs 1.5 secs 1.6 secs 1.7 secs
servlet.request GET /coffeehouse
CoffeeHouse.coffeeHouseIndex
OpenTracingBrewInfo
CoffeeHouse.getOrder
http.request GET /coffees 775 ms
laravel.request CoffeeHouse\Controllers\HomeController@coffeeList coffee_list 719 ms
laravel.action coffees 596 ms
call_external_web_service 569 ms
curl_exec http://php-coffeehouse-backend/favorites 568 ms
laravel.request CoffeeHouse\Controllers\HomeController@favorites favorites 542 ms
http.request GET /beanserver 429 ms
servlet.request GET /beanse...
BeanController.redirect
BeanController.beanServe...
mongo.query { "$eval" : ...
coffee-house | OpenTracingBrewInfo OpenTracingBrewInfo
demo.coffeehouse.dev 1.85 s (99.5% of total time)
Span Metadata
Host Info
Logs
trace_id
trace_id:1592247487685740042
DATE
SERVICE
HOST
Feb 07 16:13:16.000 | coffee-house | demo.coffeehouse.dev
GET /api/auth/ {10.8.4.7} – 200 OK – Authentication successful
Feb 07 16:13:16.000 | coffee-house | demo.coffeehouse.dev
Monitor thread successfully connected to server with description ServerDescription{address=mongodb:27017, type=STANDALONE, state=CONNECTED, ok=true, version=ServerVersion{versionList=[3, 4, 17]}, minWireVersion=0, maxWireVersion=5, maxDocumentSize=16777216, logicalSessionTimeoutMinutes=null, roundTripTimeNanos=1277564
Feb 07 16:13:17.000 | coffee-house | demo.coffeehouse.dev
java.lang.InterruptedException: Thread interrupted for external calls timeout – 500
Feb 07 16:13:18.000 | coffee-house | demo.coffeehouse.dev
GET http://java-coffeehouse:8080/coffeehouse completed with status code 200 in 1845 ms

Traces

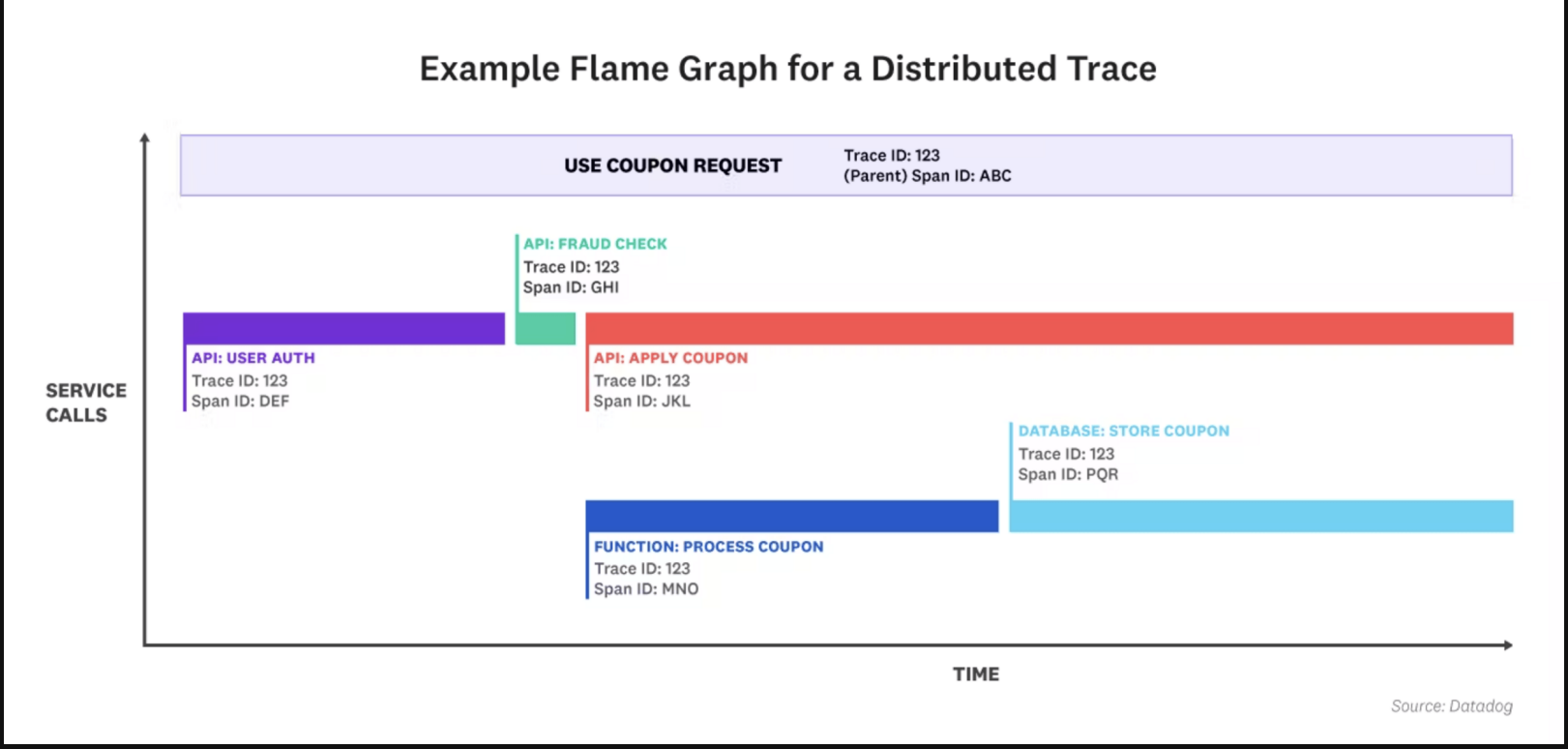
```ruby
get '/posts' do
  Datadog::Tracing.trace('web.request', service: 'my-blog', resource: 'GET /posts') do |span|
    # Trace the activerecord call
    Datadog::Tracing.trace('posts.fetch') do
      @posts = Posts.order(created_at: :desc).limit(10)
    end

    # Add some APM tags
    span.set_tag('http.method', request.request_method)
    span.set_tag('posts.count', @posts.length)

    # Trace the template rendering
    Datadog::Tracing.trace('template.render') do
      erb :index
    end
  end
end
```

Traces



Distributed tracing - Honeycomb

Traces


Example Flame Graph for a Distributed Trace
USE COUPON REQUEST
Trace ID: 123
(Parent) Span ID: ABC
API: FRAUD CHECK
Trace ID: 123
Span ID: GHI
API: USER AUTH
Trace ID: 123
Span ID: DEF
API: APPLY COUPON
Trace ID: 123
Span ID: JKL
DATABASE: STORE COUPON
Trace ID: 123
Span ID: PQR
FUNCTION: PROCESS COUPON
Trace ID: 123
Span ID: MNO
SERVICE CALLS
TIME
Source: Datadog

Traces

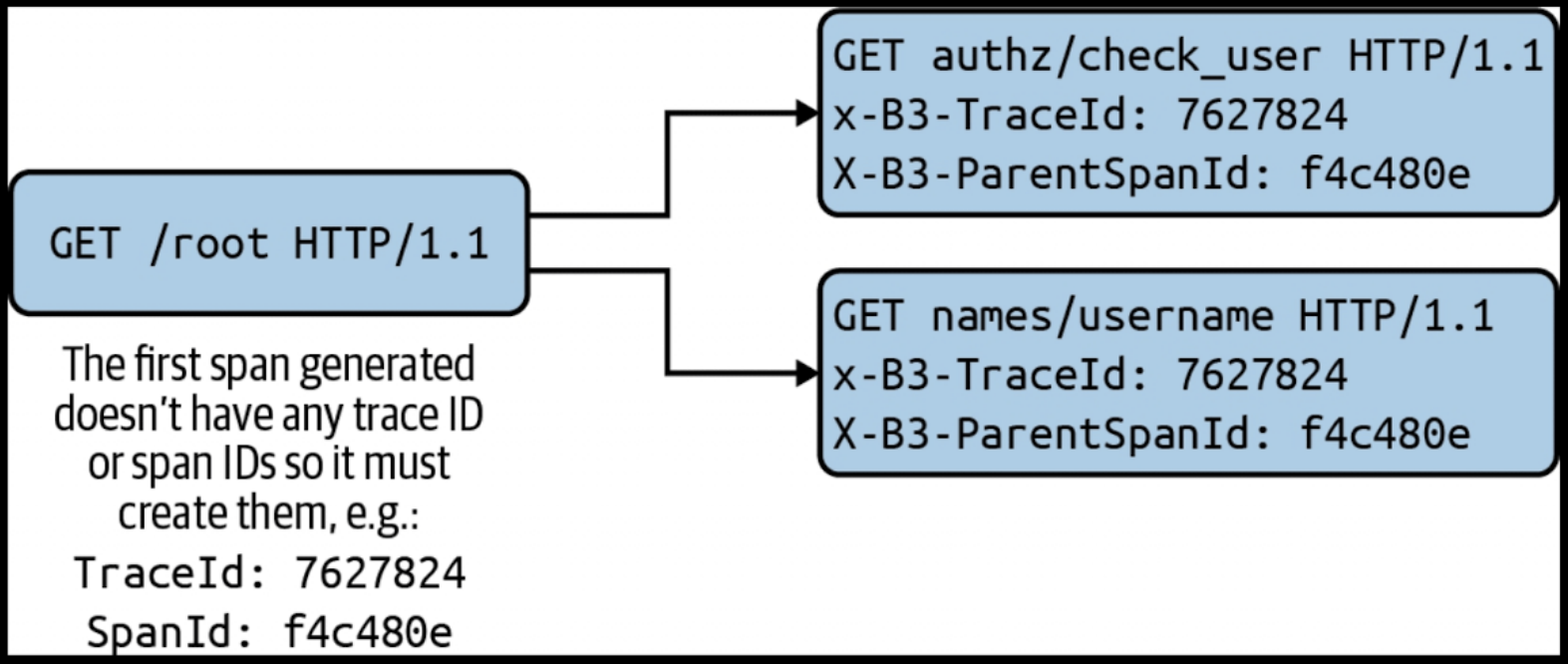


Figure 6-3. Our example app would now propagate `traceID` and `parentID` to each child span

Traces

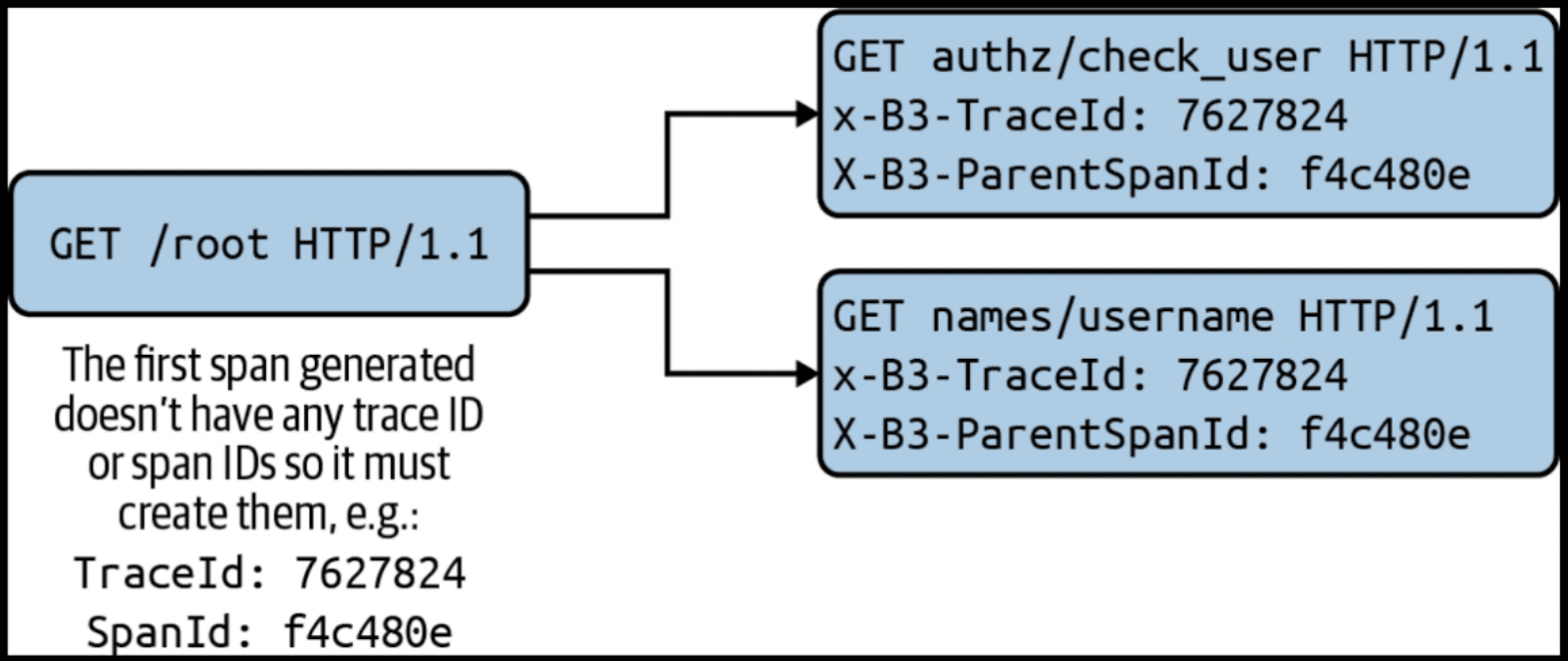


Figure 6-3. Our example app would now propagate `traceID` and `parentID` to each child span

Traces

1.

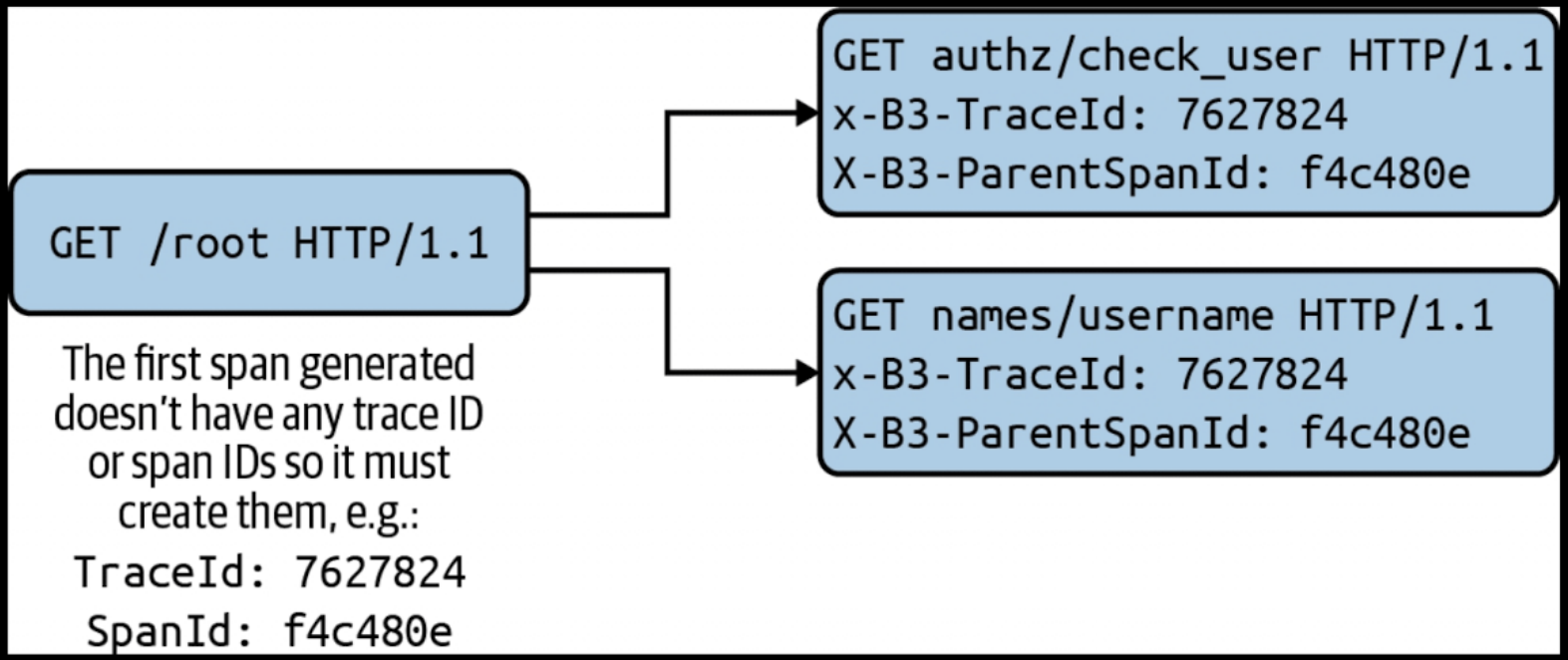


Figure 6-3. Our example app would now propagate `traceID` and `parentID` to each child span

Traces

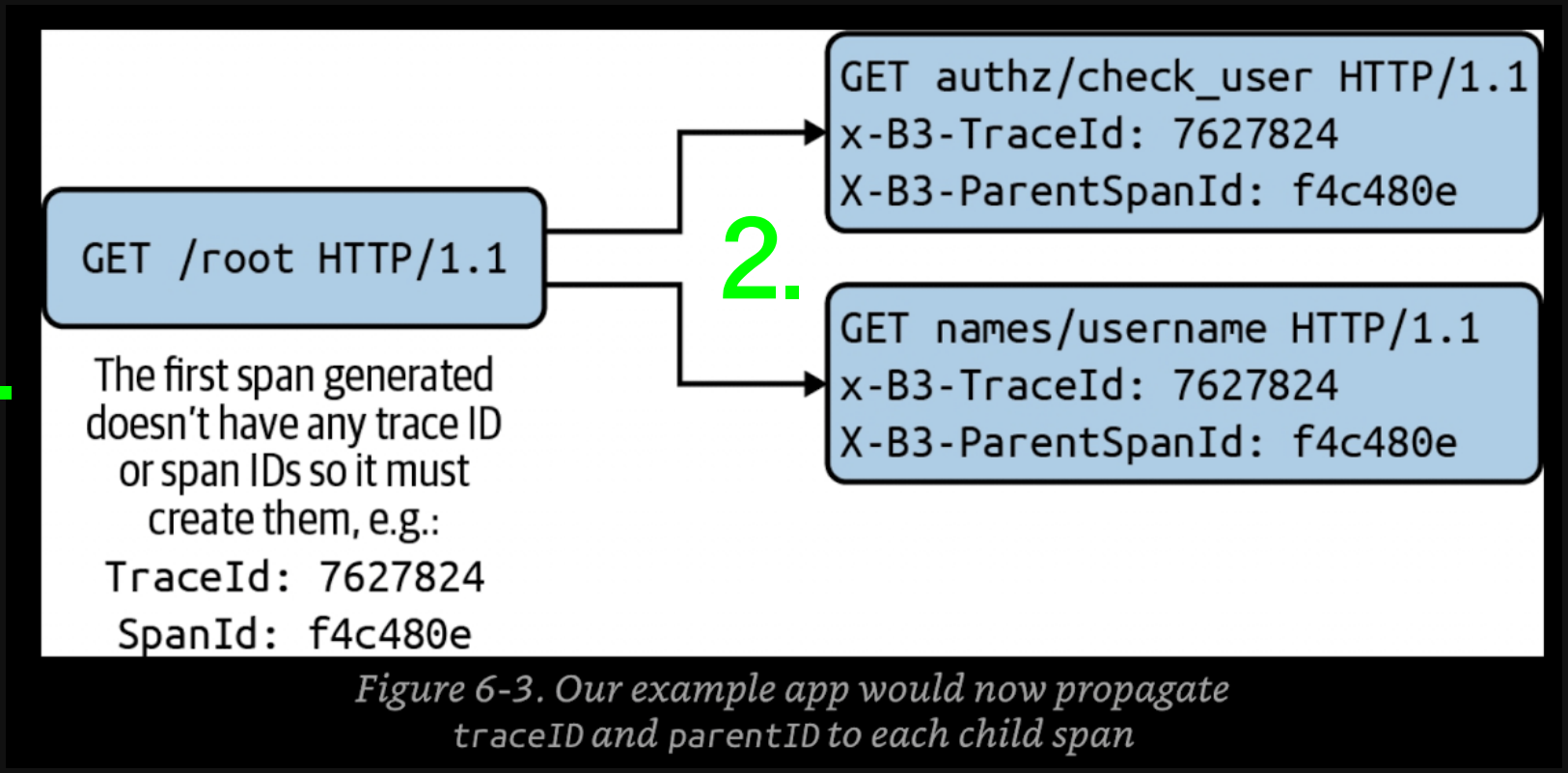


Figure 6-3. Our example app would now propagate `traceID` and `parentID` to each child span

OpenTelemtry (OTEL)

> *OTel captures traces, metrics, logs, and other application telemetry data and lets you send it to the backend of your choice.*

OTEL ~= Observability api + spec

OTel me, code, why are you slow?

opentelemetry.io/docs/languages/go/getting-started/

github.com/virtualandy/otel-simple-example

```go
// main.go
func main() {
    http.HandleFunc("/rolldice", rolldice)
    log.Fatal(http.ListenAndServe(":8000", nil))
}
```

```go
// rolldice.go
func rolldice(w http.ResponseWriter, r *http.Request) {
    roll := 1 + rand.Intn(6)

    // Sleep for a few seconds, so we can see longer traces
    // And breakout the slow api call versus other work
    // that might be done in this function, i.e. a DB query
    time.Sleep(time.Millisecond * time.Duration(roll*100))

    log.Println("simulating a call to slow api")
    err := simulateSlowAPI(500 * roll)
    if err != nil {
        log.Printf("Failed to make call to simulate slow api")
    }

    resp := strconv.Itoa(roll) + "\n"
```

```go
// main.go
func main() {
    http.HandleFunc("/rolldice", rolldice)
    log.Fatal(http.ListenAndServe(":8000", nil))
}
```

```go
// rolldice.go
func rolldice(w http.ResponseWriter, r *http.Request) {
    roll := 1 + rand.Intn(6)

    // Sleep for a few seconds, so we can see longer traces
    // And breakout the slow api call versus other work
    // that might be done in this function, i.e. a DB query
    time.Sleep(time.Millisecond * time.Duration(roll*100))

    log.Println("simulating a call to slow api")
    err := simulateSlowAPI(500 * roll)
    if err != nil {
        log.Printf("Failed to make call to simulate slow api")
    }

    resp := strconv.Itoa(roll) + "\n"
```

```go
// rolldice.go
// A function for simulating slowness with an external http ca
func simulateSlowAPI(sleepInMilliseconds int) error {
    // Construct the URL with the sleep parameter
    baseURL := "https://fakeresponder.com/"
    sleepParam := "?sleep=" + strconv.Itoa(sleepInMilliseconds

    // Create the full URL
    fullURL := baseURL + sleepParam

    // Make the GET request
    resp, err := http.Get(fullURL)
    if err != nil {
        log.Println("Error making the GET request:", err)
        return err
    }
```

```go
// rolldice.go
// A function for simulating slowness with an external http ca
func simulateSlowAPI(sleepInMilliseconds int) error {
    // Construct the URL with the sleep parameter
    baseURL := "https://fakeresponder.com/"
    sleepParam := "?sleep=" + strconv.Itoa(sleepInMilliseconds

    // Create the full URL
    fullURL := baseURL + sleepParam

    // Make the GET request
    resp, err := http.Get(fullURL)
    if err != nil {
        log.Println("Error making the GET request:", err)
        return err
    }
```

```
// > go run .

2024/04/24 22:03:11 simulating a call to slow api
2024/04/24 22:03:16 simulating a call to slow api
2024/04/24 22:03:20 simulating a call to slow api
2024/04/24 22:03:23 simulating a call to slow api

// > watch -n 2 'curl -s localhost:8000/rolldice
Every 2.0s: curl -s localhost:8000/rolldice

4
```

```
## TODO: find out what's slow

_Assuming we didn't already know_ :)

- [ ] Add more logs
- [ ] Add metrics (response time, number of calls, etc)

*OR*

- [x] Add OTel
```

```go
// main.go
    import "go.opentelemetry.io/contrib/instrumentation/net/ht

    // Set up OpenTelemetry.
    otelShutdown, err := setupOTelSDK(ctx)
    if err != nil {
        return
    }
    // Handle shutdown properly so nothing leaks.
    defer func() {
        err = errors.Join(err, otelShutdown(context.Backgroun
    }()

    ...

    handleFunc := func(pattern string, handlerFunc func(http.
    // Configure the "http.route" for the HTTP instrumentatio
    handler := otelhttp.WithRouteTag(pattern, http.HandlerFun
    mux.Handle(pattern, handler)

    handler := otelhttp.NewHandler(mux, "/")
```

```go
// main.go
    import "go.opentelemetry.io/contrib/instrumentation/net/ht

    // Set up OpenTelemetry.
    otelShutdown, err := setupOTelSDK(ctx)
    if err != nil {
        return
    }
    // Handle shutdown properly so nothing leaks.
    defer func() {
        err = errors.Join(err, otelShutdown(context.Backgroun
    }()

    ...

    handleFunc := func(pattern string, handlerFunc func(http.
    // Configure the "http.route" for the HTTP instrumentatio
    handler := otelhttp.WithRouteTag(pattern, http.HandlerFun
    mux.Handle(pattern, handler)

    handler := otelhttp.NewHandler(mux, "/")
```

```go
// otel.go

    // Set up trace provider.
    tracerProvider, err := newTraceProvider()
    if err != nil {
        handleErr(err)
        return
    }
    shutdownFuncs = append(shutdownFuncs, tracerProvider.Shutd
    otel.SetTracerProvider(tracerProvider)

    case "stdout":
      exporter, err = stdouttrace.New(
        stdouttrace.WithPrettyPrint()
      )


```

```go
// rolldice.go
var (
    tracer  = otel.Tracer("rolldice")
    meter   = otel.Meter("rolldice")
    rollCnt metric.Int64Counter
)

func rolldice(w http.ResponseWriter, r *http.Request) {
    ctx, span := tracer.Start(r.Context(), "rolling dice")
    defer span.End()

    roll := 1 + rand.Intn(6)

    // Sleep for a few seconds, so we can see longer traces
    time.Sleep(time.Millisecond * time.Duration(roll*100))

    span.AddEvent("simulating a call to slow api")
    err := simulateSlowAPI(500*roll, ctx)
    if err != nil {
        log.Printf("Failed to make call to simulate slow api")
    }
```

```go
// rolldice.go
var (
    tracer  = otel.Tracer("rolldice")
    meter   = otel.Meter("rolldice")
    rollCnt metric.Int64Counter
)

func rolldice(w http.ResponseWriter, r *http.Request) {
    ctx, span := tracer.Start(r.Context(), "rolling dice")
    defer span.End()

    roll := 1 + rand.Intn(6)

    // Sleep for a few seconds, so we can see longer traces
    time.Sleep(time.Millisecond * time.Duration(roll*100))

    span.AddEvent("simulating a call to slow api")
    err := simulateSlowAPI(500*roll, ctx)
    if err != nil {
        log.Printf("Failed to make call to simulate slow api")
    }
```

```go
// rolldice.go

func simulateSlowAPI(sleepInMilliseconds int, ctx context.Conte
    ctx, childSpan := tracer.Start(ctx, "simulatedSlowApiSpan")
    defer childSpan.End()
    // Construct the URL with the sleep parameter
    baseURL := "https://fakeresponder.com/"
    sleepParam := "?sleep=" + strconv.Itoa(sleepInMilliseconds)
```

```
(base) ➜  otel-dice-example git:(otel-with-stdout) ✗ go run .

{"Resource":[
  {"Key":"service.name","Value":
   {"Type":"STRING","Value":"dice"}
  },
  {"Key":"service.version","Value":
   {"Type":"STRING","Value":"0.1.0"}
  },
  {"Key":"telemetry.sdk.language","Value":
   {"Type":"STRING","Value":"go"}
  },
  {"Key":"telemetry.sdk.name","Value":
   {"Type":"STRING","Value":"opentelemetry"}
  },
  {"Key":"telemetry.sdk.version","Value":
   {"Type":"STRING","Value":"1.25.0"}
  }
],"ScopeMetrics":[]}

{"Resource":[{"Key":"service.name","Value":{"Type":"STRING","V
```

```
{
    "Name": "/",
    "SpanContext": {
        "TraceID": "5a844fd338c4c00ec001fb28ea352c64",
        "SpanID": "4908426582a52298",
    },
    "Parent": {
        "TraceID": "00000000000000000000000000000000",
        "SpanID": "0000000000000000",
    },
    "SpanKind": 2,
    "StartTime": "2024-04-25T00:21:00.179788-06:00",
    "EndTime": "2024-04-25T00:21:01.471713625-06:00",
    "Attributes": [
        {
            "Key": "http.route",
            "Value": {
                "Type": "STRING",
                "Value": "/rolldice"
            }
        },
```

```
{
    "Name": "/",
    "SpanContext": {
        "TraceID": "5a844fd338c4c00ec001fb28ea352c64",
        "SpanID": "4908426582a52298",
    },
    "Parent": {
        "TraceID": "00000000000000000000000000000000",
        "SpanID": "0000000000000000",
    },
```

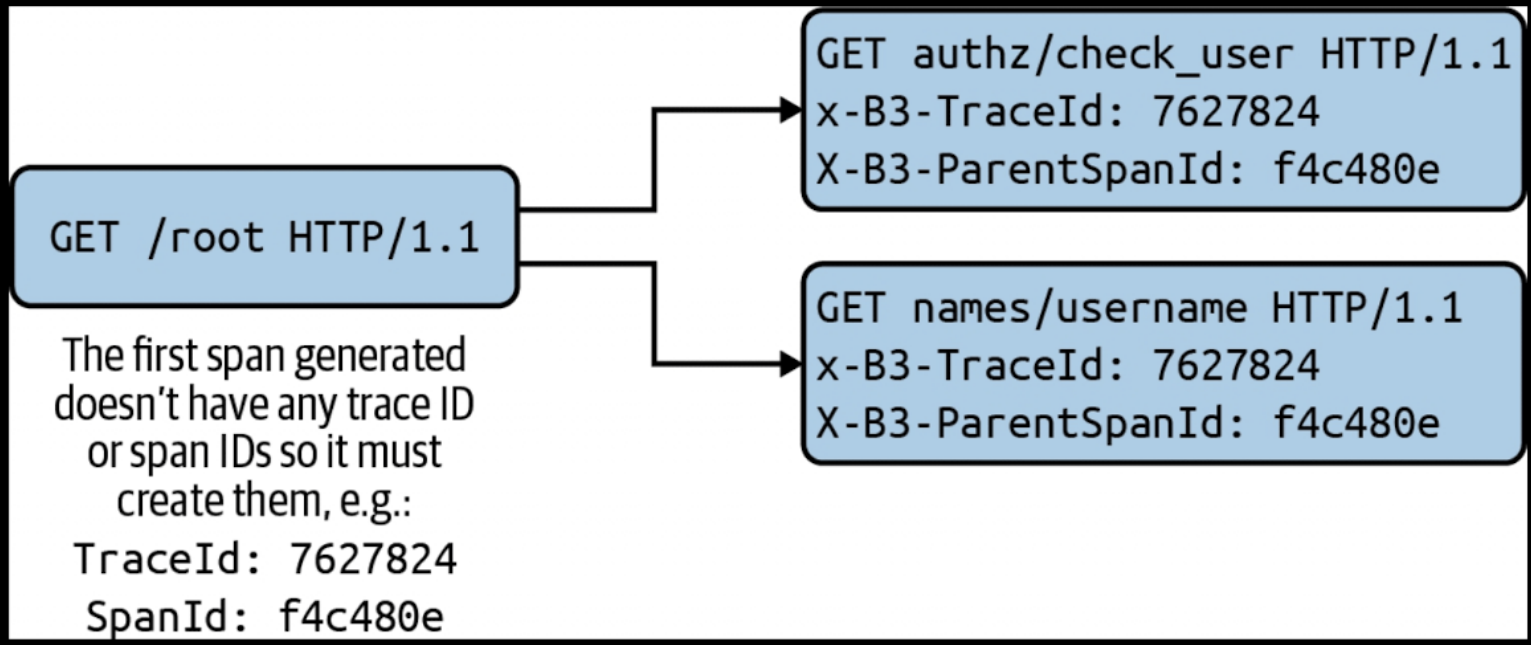
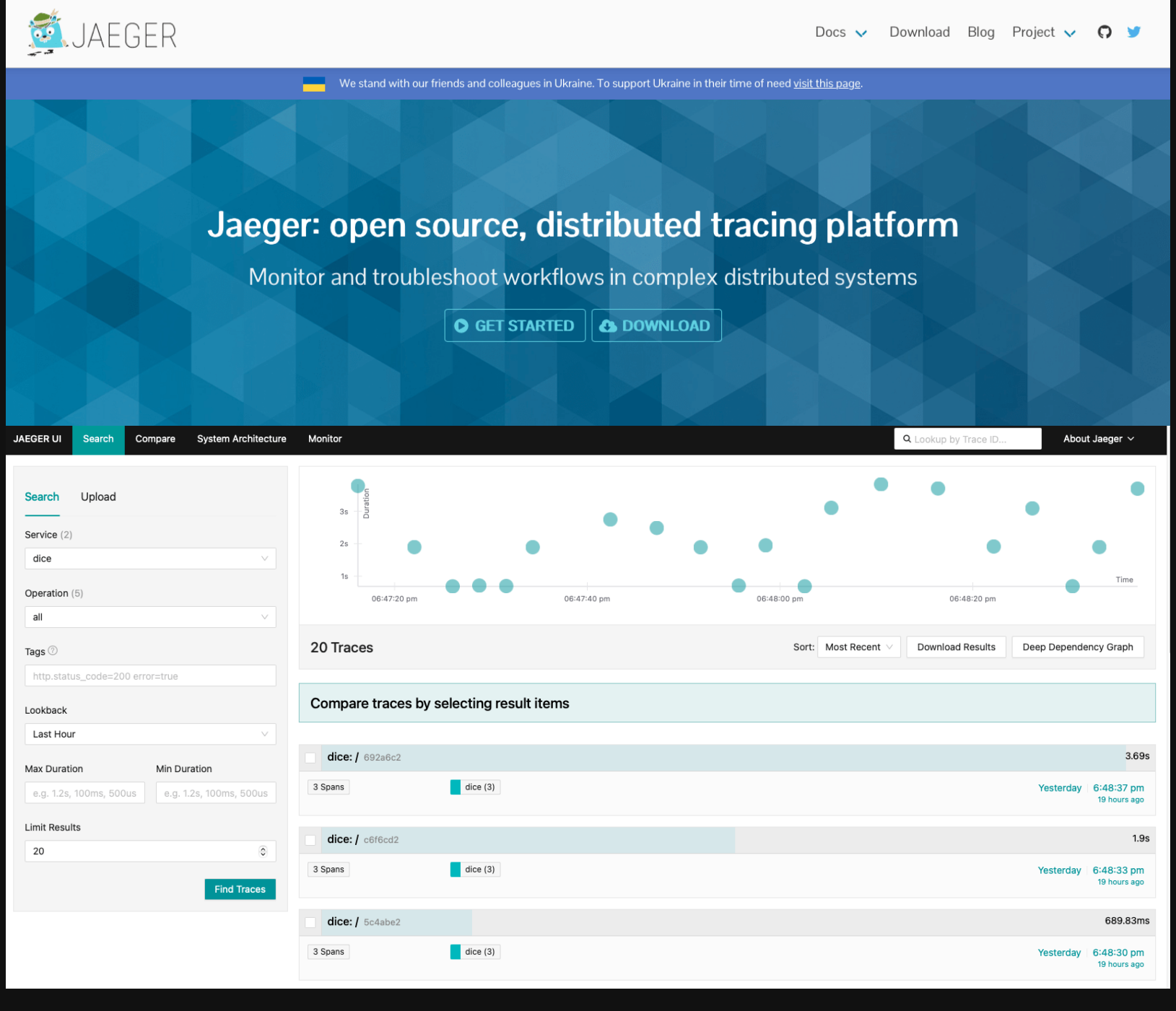


Figure 6-3. Our example app would now propagate

```
{
    "Name": "rolling dice",
    "SpanContext": {
        "TraceID": "5a844fd338c4c00ec001fb28ea352c64",
        "SpanID": "7e394450317027c5",
    },
    "Parent": {
        "TraceID": "5a844fd338c4c00ec001fb28ea352c64",
        "SpanID": "4908426582a52298",
    },
    "Attributes": [
        {
            "Key": "roll.value",
            "Value": {
                "Type": "INT64",
                "Value": 2
            }
        }
    ],
```


JAEGER
Docs
Download
Blog
Project
We stand with our friends and colleagues in Ukraine. To support Ukraine in their time of need visit this page.
Jaeger: open source, distributed tracing platform
Monitor and troubleshoot workflows in complex distributed systems
GET STARTED
DOWNLOAD
JAEGER UI
Search
Compare
System Architecture
Monitor
Lookup by Trace ID...
About Jaeger
Search
Upload
Service (2)
dice
Operation (5)
all
Tags
http.status_code=200 error=true
Lookback
Last Hour
Max Duration
e.g. 1.2s, 100ms, 500us
Min Duration
e.g. 1.2s, 100ms, 500us
Limit Results
20
Find Traces
Duration
3s
2s
1s
Time
06:47:20 pm
06:47:40 pm
06:48:00 pm
06:48:20 pm
20 Traces
Sort:
Most Recent
Download Results
Deep Dependency Graph
Compare traces by selecting result items
dice: / 692a6c2
3.69s
3 Spans
dice (3)
Yesterday 6:48:37 pm
19 hours ago
dice: / c6f6cd2
1.9s
3 Spans
dice (3)
Yesterday 6:48:33 pm
19 hours ago
dice: / 5c4abe2
689.83ms
3 Spans
dice (3)
Yesterday 6:48:30 pm
19 hours ago

```
> docker run -d --name jaeger \
  -e COLLECTOR_OTLP_ENABLED=true \
  -p 16686:16686 \
  -p 4317:4317 \
  -p 4318:4318 \
  jaegertracing/all-in-one:latest


> OTEL_EXPORTER_OTLP_ENDPOINT=http://localhost:4318 go run .
```

JAEGER UI | Search | Compare | System Architecture | Monitor | Lookup by Trace ID... | About Jaeger

Search | Upload

Service (2)

dice

Operation (5)

all

Tags

http.status_code=200 error=true

Lookback

Last Hour

Max Duration

e.g. 1.2s, 100...

Min Duration

e.g. 1.2s, 100...

Limit Results

20

Find Traces

15 Traces

Sort: Most Recent | Download Results | Deep Dependency Graph

Compare traces by selecting result items

dice: / 1e25936 — 3.69s

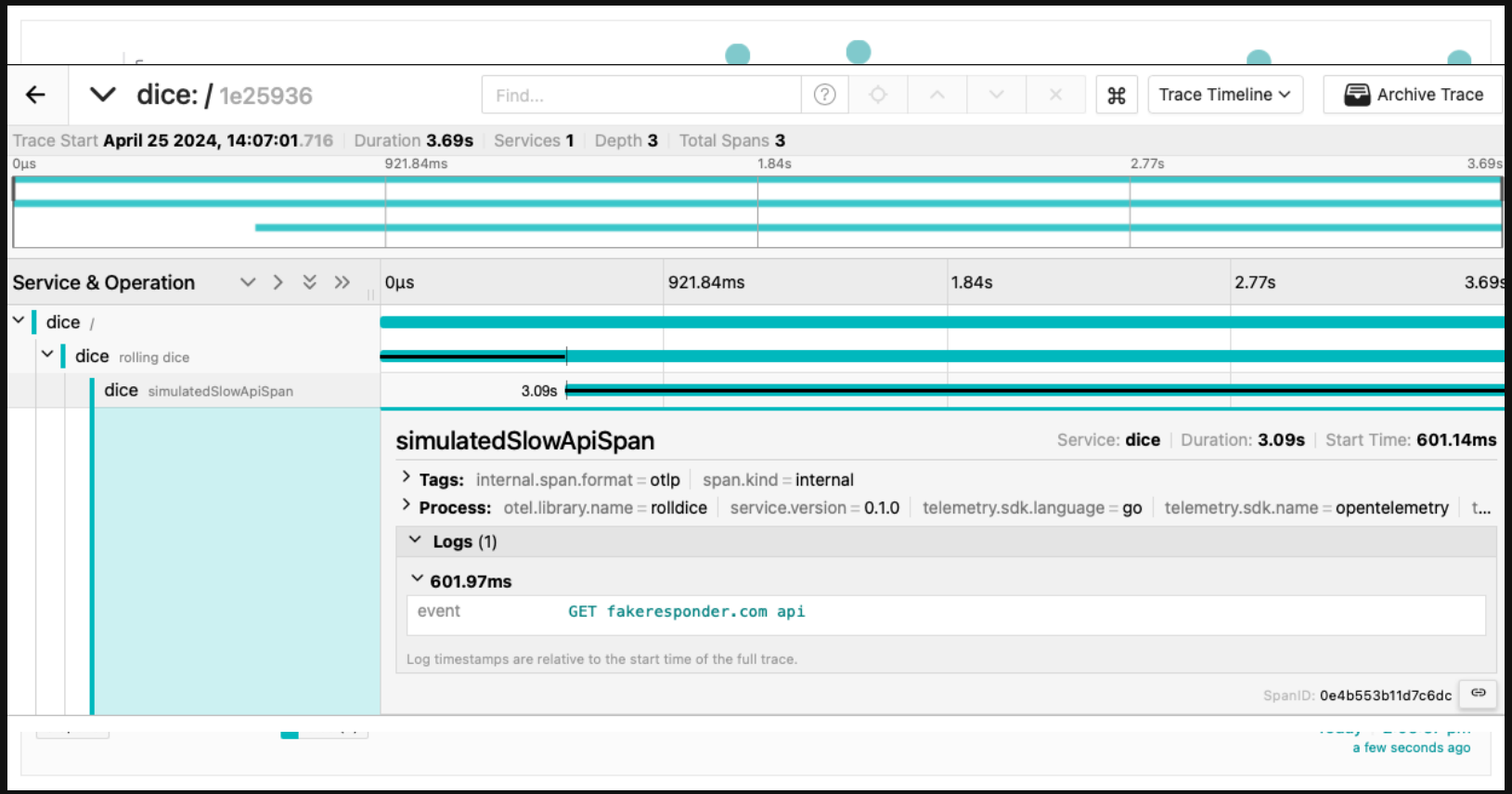
3 Spans | dice (3)

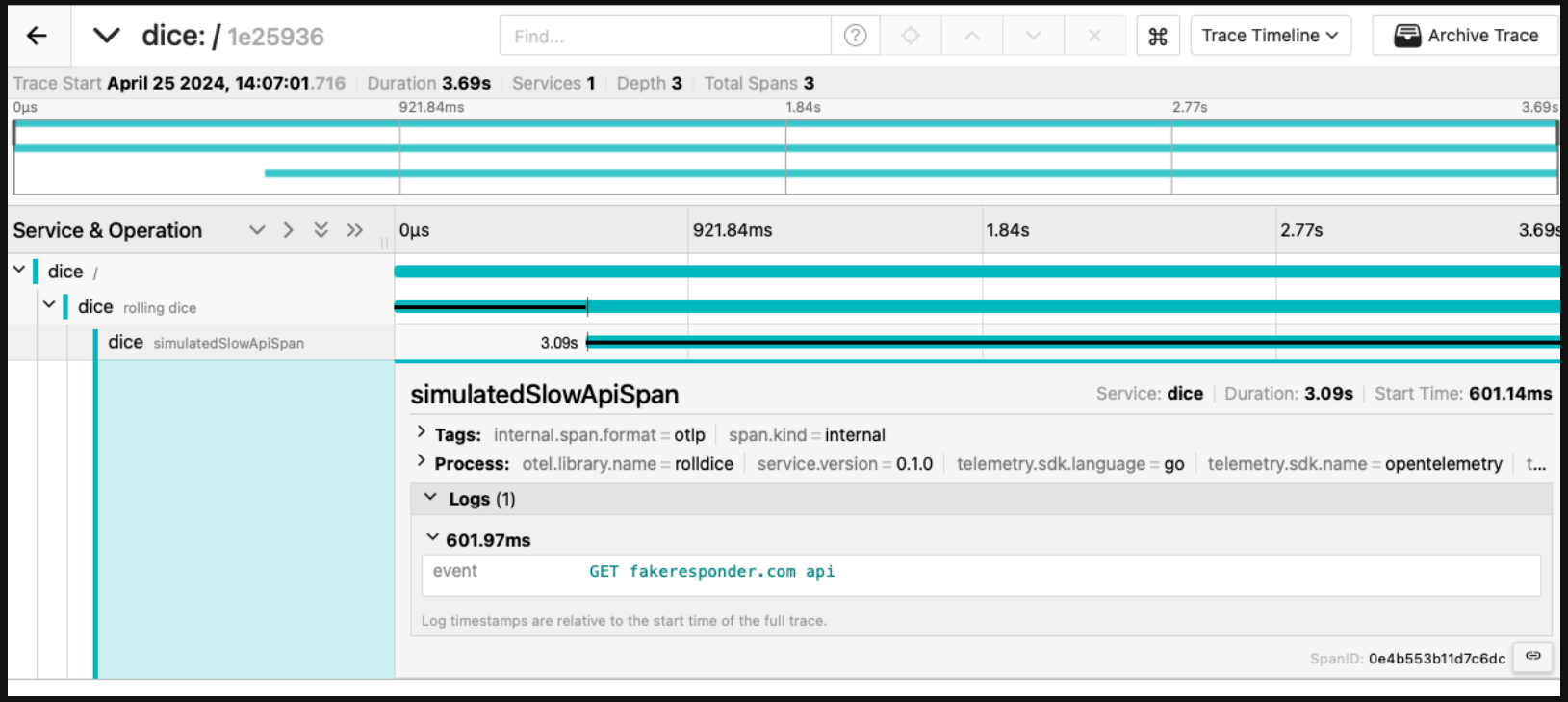
Today | 2:07:01 pm
a few seconds ago

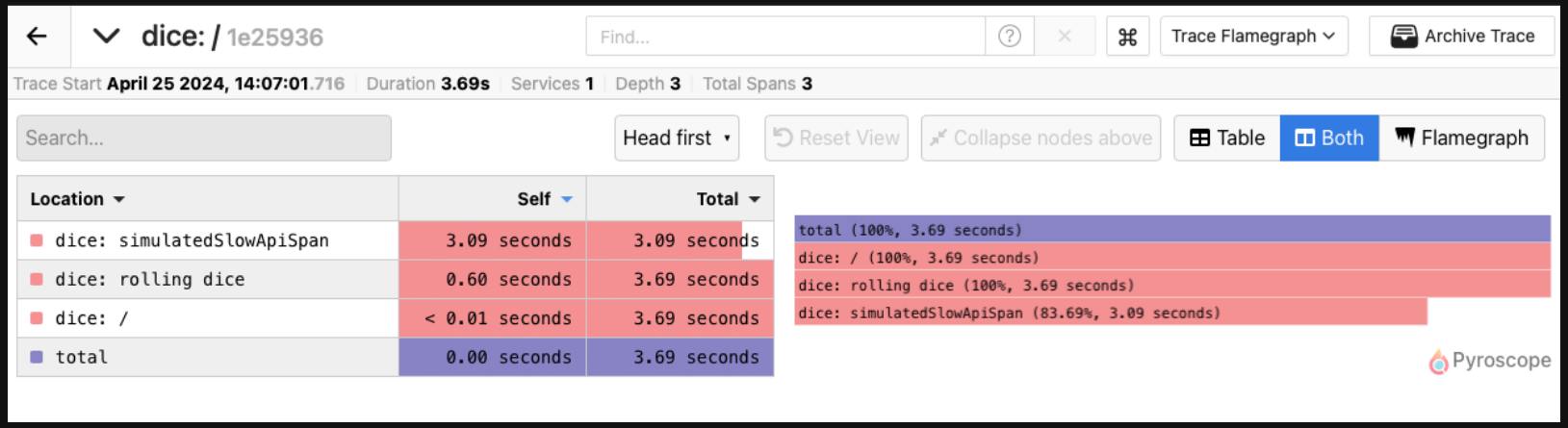
dice: / 0d24b7e — 1.95s

3 Spans | dice (3)

Today | 2:06:57 pm
a few seconds ago


dice: / 1e25936
Find...
Trace Timeline
Archive Trace
Trace Start April 25 2024, 14:07:01.716 | Duration 3.69s | Services 1 | Depth 3 | Total Spans 3
0µs
921.84ms
1.84s
2.77s
3.69s
Service & Operation
dice /
dice rolling dice
dice simulatedSlowApiSpan
3.09s
simulatedSlowApiSpan
Service: dice | Duration: 3.09s | Start Time: 601.14ms
Tags: internal.span.format = otlp | span.kind = internal
Process: otel.library.name = rolldice | service.version = 0.1.0 | telemetry.sdk.language = go | telemetry.sdk.name = opentelemetry
Logs (1)
601.97ms
event GET fakeresponder.com api
Log timestamps are relative to the start time of the full trace.
SpanID: 0e4b553b11d7c6dc
a few seconds ago

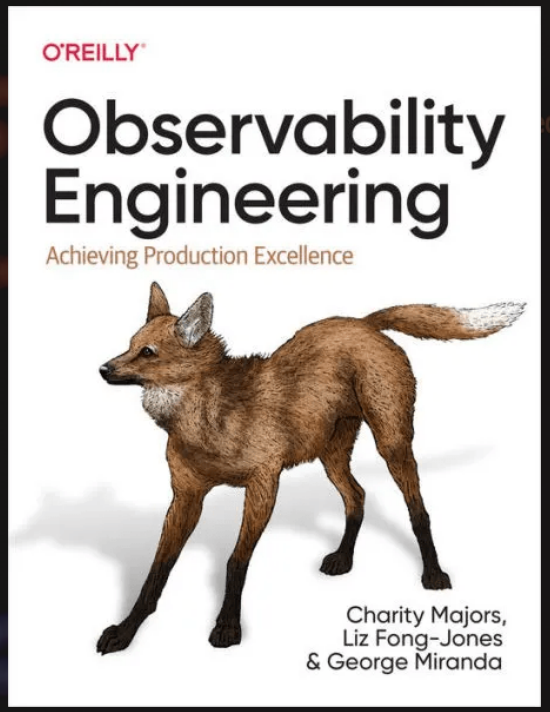

dice: / 1e25936
Find...
Trace Timeline
Archive Trace
Trace Start April 25 2024, 14:07:01.716 | Duration 3.69s | Services 1 | Depth 3 | Total Spans 3
0µs
921.84ms
1.84s
2.77s
3.69s
Service & Operation
dice /
dice rolling dice
dice simulatedSlowApiSpan
3.09s
simulatedSlowApiSpan
Service: dice | Duration: 3.09s | Start Time: 601.14ms
Tags: internal.span.format = otlp | span.kind = internal
Process: otel.library.name = rolldice | service.version = 0.1.0 | telemetry.sdk.language = go | telemetry.sdk.name = opentelemetry | t...
Logs (1)
601.97ms
event GET fakeresponder.com api
Log timestamps are relative to the start time of the full trace.
SpanID: 0e4b553b11d7c6dc


dice: / 1e25936
Find...
Trace Flamegraph
Archive Trace
Trace Start April 25 2024, 14:07:01.716 | Duration 3.69s | Services 1 | Depth 3 | Total Spans 3
Search...
Head first
Reset View
Collapse nodes above
Table
Both
Flamegraph
Location
Self
Total
dice: simulatedSlowApiSpan
3.09 seconds
3.09 seconds
dice: rolling dice
0.60 seconds
3.69 seconds
dice: /
< 0.01 seconds
3.69 seconds
total
0.00 seconds
3.69 seconds
total (100%, 3.69 seconds)
dice: / (100%, 3.69 seconds)
dice: rolling dice (100%, 3.69 seconds)
dice: simulatedSlowApiSpan (83.69%, 3.09 seconds)
Pyroscope

OTel your future self

Win a free print copy of *Observability Engineering*

https://forms.gle/RFurH6QUahkFTsXFA



Thank you!

@virtualandy

slides.com/virtualandy/otel-me-a-story/