TRACING

IN ELIXIR

LET'S MEET ALEXANDER

Smallcentrix

We need more diagnostics!

– Pretty much everybody


Google

Look at OpenTracing, you scrub!

— 

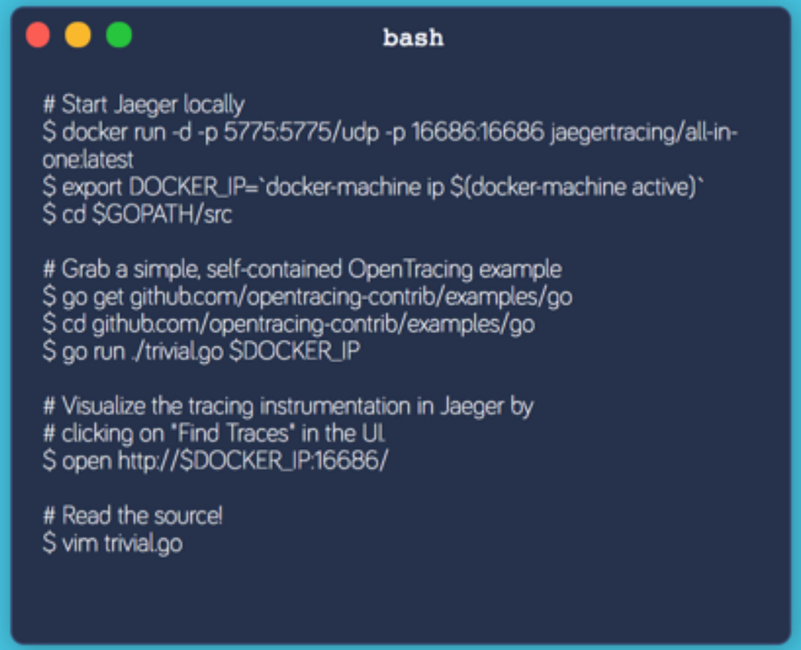
WHAT EXACTLY IS TRACING?

ANALYZING TRANSACTIONS

TRACING IS HARD

- Contexts need to be passed around
  - Within and between processes
  - Through OSS packages (ORM etc.)
  - Self-contained services (NGINX, Redis etc.)
  - Arbitrary glue code and business logic

VENDOR LOCK-IN



Vendor-neutral APIs and instrumentation for distributed tracing

Libraries available in 9 languages

Go, JavaScript, Java, Python, Ruby, PHP, Objective-C, C++, C#

February 2018: OpenTracing Project Update

Supported Tracers

LIGHTSTEP appdash TRACER JAEGER Ha

Supported Frameworks

Go kit django DROPWIZARD MOTAN

TIMED OPERATIONS

CALLED SPANS

RELATIONS BETWEEN SPANS

CONTEXT PROPAGATION

- Span management (start, finish, decorate)
- Inter-process propagation (overcome process boundaries)
- Active span management (store, retrieve)

WHAT ABOUT ELIXIR?

OTTER

OPENTRACING TOOLKIT FOR ERLANG

PARTIAL OPENTRACING IMPLEMENTATION

- Span management (start, finish, decorate)
  - Kinda: Active span management

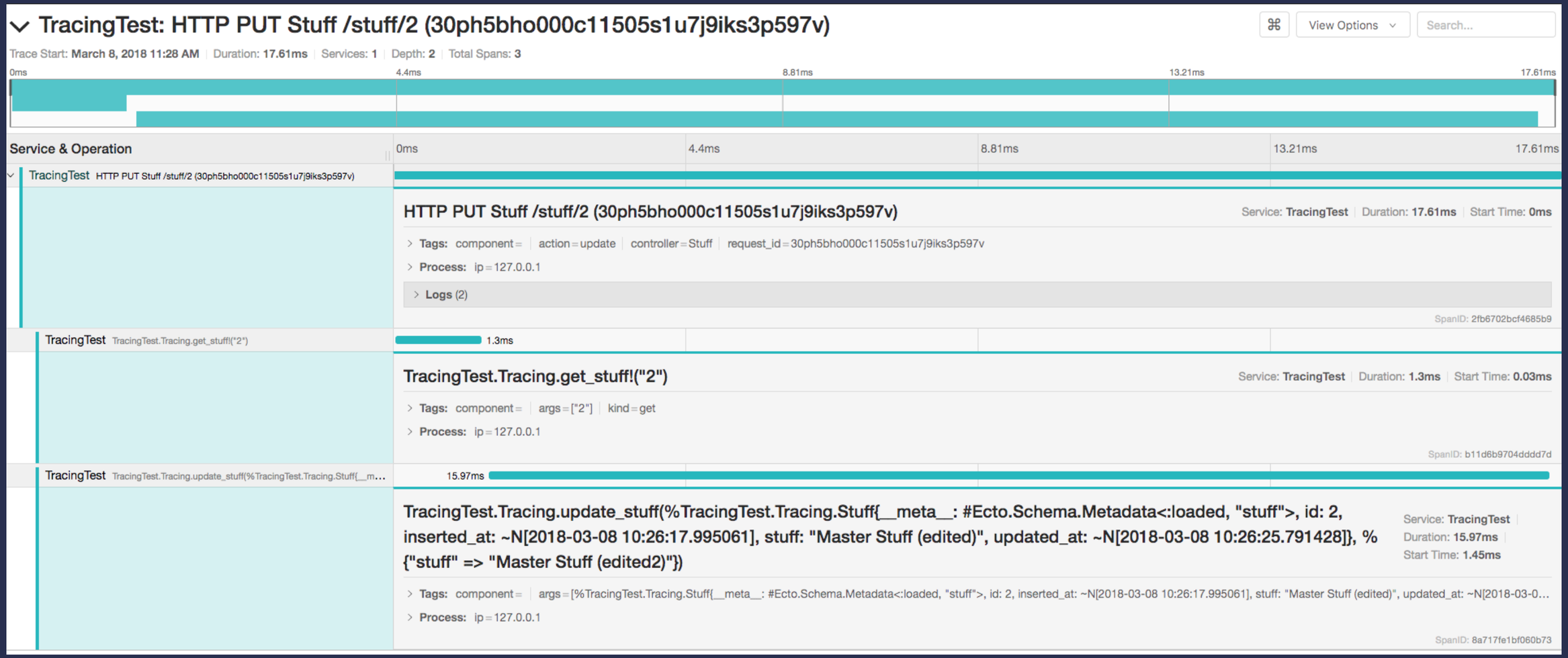
EXRAY

TRACING ANNOTATIONS BUILT WITH OTTER

```elixir
defmodule Stuff do
  use ExRay, pre: :start_span, post: :finish_span

  defp start_span(context), do: ...
  defp finish_span(context, span, result), do: ...

  @trace kind: "list"
  def list_stuff do
    Stuff.Repo.all(Stuff)
  end

  ...
end
```

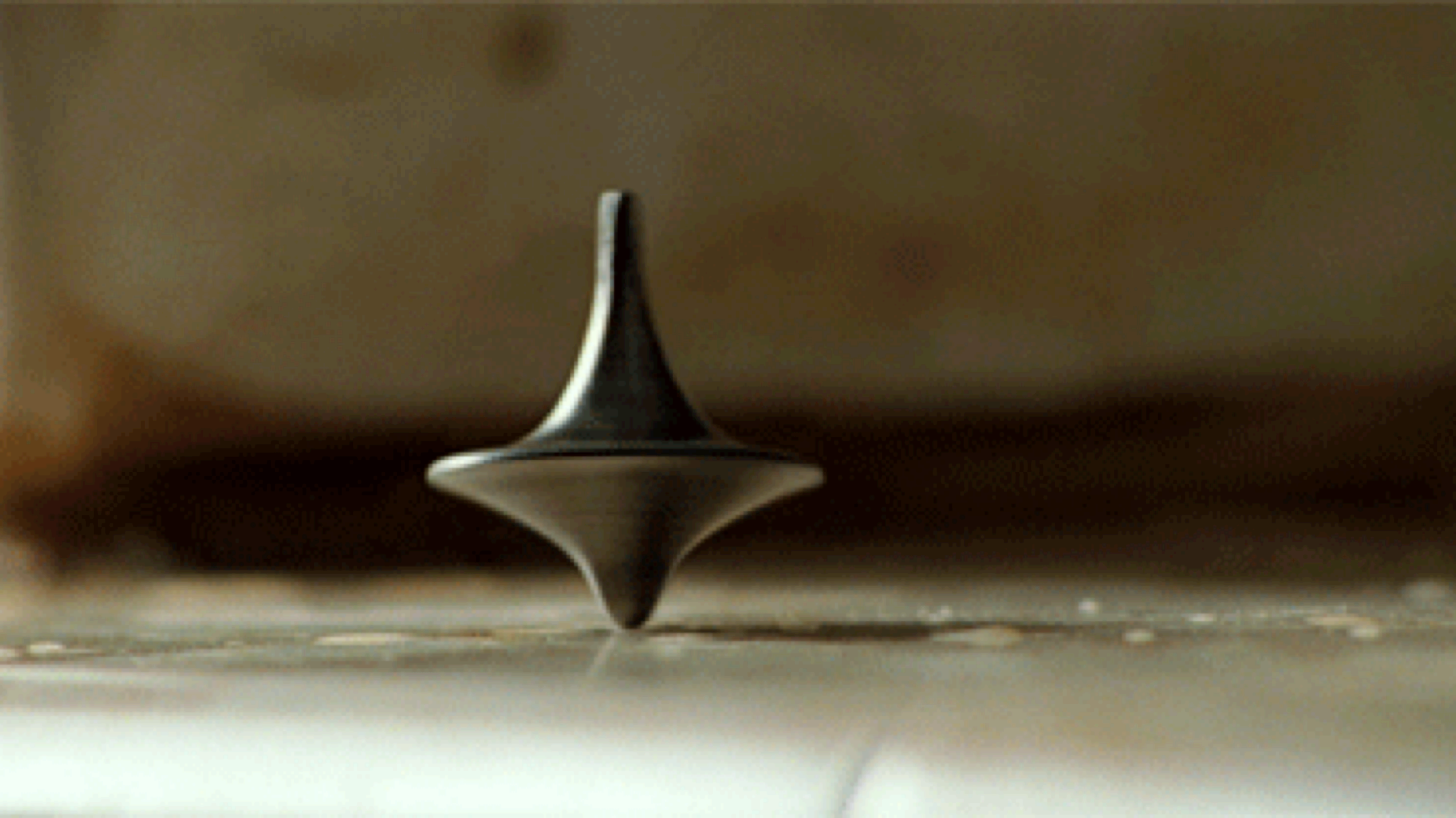
REMEMBER  ?

HE BUILT A PROOF OF CONCEPT


TracingTest: HTTP PUT Stuff /stuff/2 (30ph5bho000c11505s1u7j9iks3p597v)
View Options
Search...
Trace Start: March 8, 2018 11:28 AM | Duration: 17.61ms | Services: 1 | Depth: 2 | Total Spans: 3
0ms
4.4ms
8.81ms
13.21ms
17.61ms
Service & Operation
TracingTest HTTP PUT Stuff /stuff/2 (30ph5bho000c11505s1u7j9iks3p597v)
HTTP PUT Stuff /stuff/2 (30ph5bho000c11505s1u7j9iks3p597v)
Service: TracingTest | Duration: 17.61ms | Start Time: 0ms
Tags: component = | action = update | controller = Stuff | request_id = 30ph5bho000c11505s1u7j9iks3p597v
Process: ip = 127.0.0.1
Logs (2)
SpanID: 2fb6702bcf4685b9
TracingTest TracingTest.Tracing.get_stuff!("2")
1.3ms
TracingTest.Tracing.get_stuff!("2")
Service: TracingTest | Duration: 1.3ms | Start Time: 0.03ms
Tags: component = | args = ["2"] | kind = get
Process: ip = 127.0.0.1
SpanID: b11d6b9704dddd7d
TracingTest TracingTest.Tracing.update_stuff(%TracingTest.Tracing.Stuff{__m...
15.97ms
TracingTest.Tracing.update_stuff(%TracingTest.Tracing.Stuff{__meta__: #Ecto.Schema.Metadata<:loaded, "stuff">, id: 2, inserted_at: ~N[2018-03-08 10:26:17.995061], stuff: "Master Stuff (edited)", updated_at: ~N[2018-03-08 10:26:25.791428]}, %{"stuff" => "Master Stuff (edited2)"})
Service: TracingTest
Duration: 15.97ms
Start Time: 1.45ms
Tags: component = | args = [%TracingTest.Tracing.Stuff{__meta__: #Ecto.Schema.Metadata<:loaded, "stuff">, id: 2, inserted_at: ~N[2018-03-08 10:26:17.995061], stuff: "Master Stuff (edited)", updated_at: ~N[2018-03-0...
Process: ip = 127.0.0.1
SpanID: 8a717fe1bf060b73

INTEGRATE IT INTO THE PROJECT!

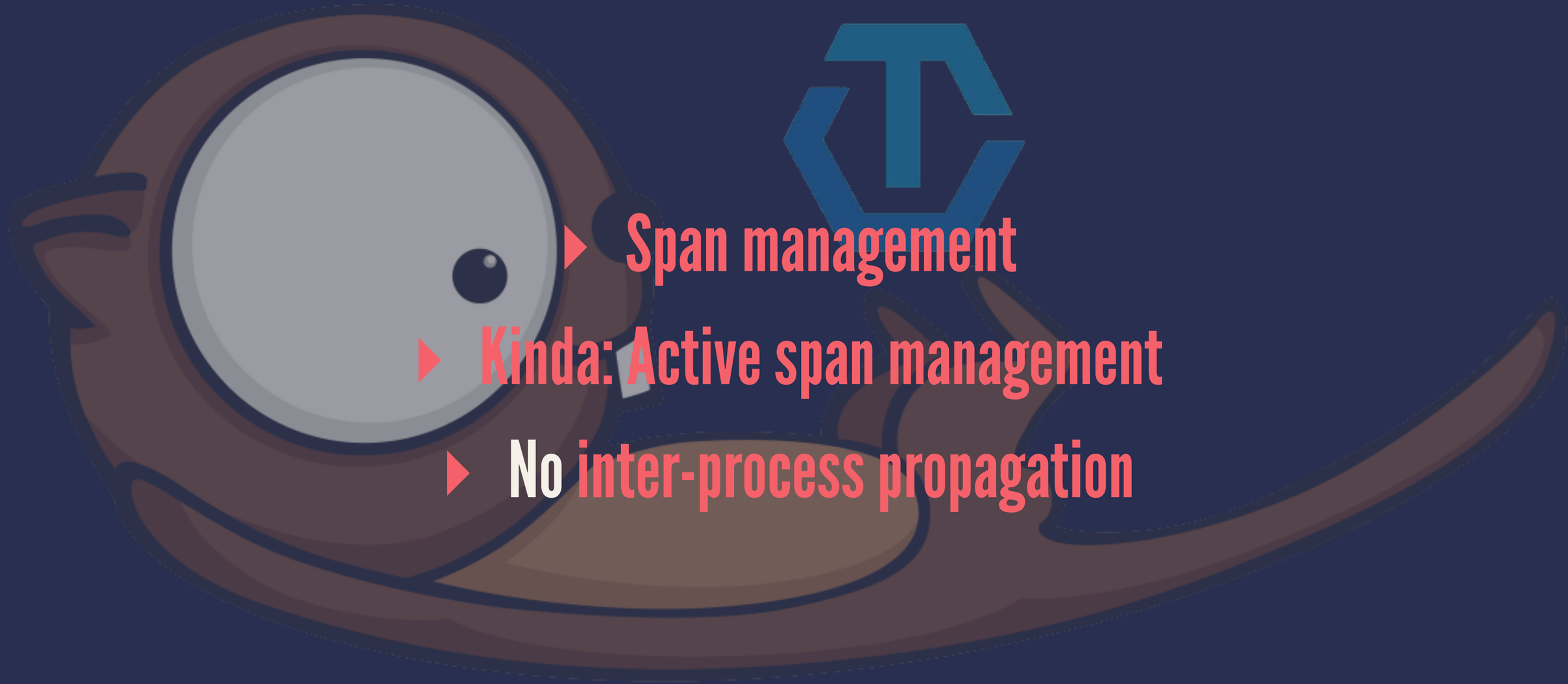
TRACING PLUG



CROSS PROCESS BOUNDARIES

EVENT-DRIVEN

INTER-PROCESS PROPAGATION



- Span management
- Kinda: Active span management
- No inter-process propagation


panicked screeching

OPTIONS?

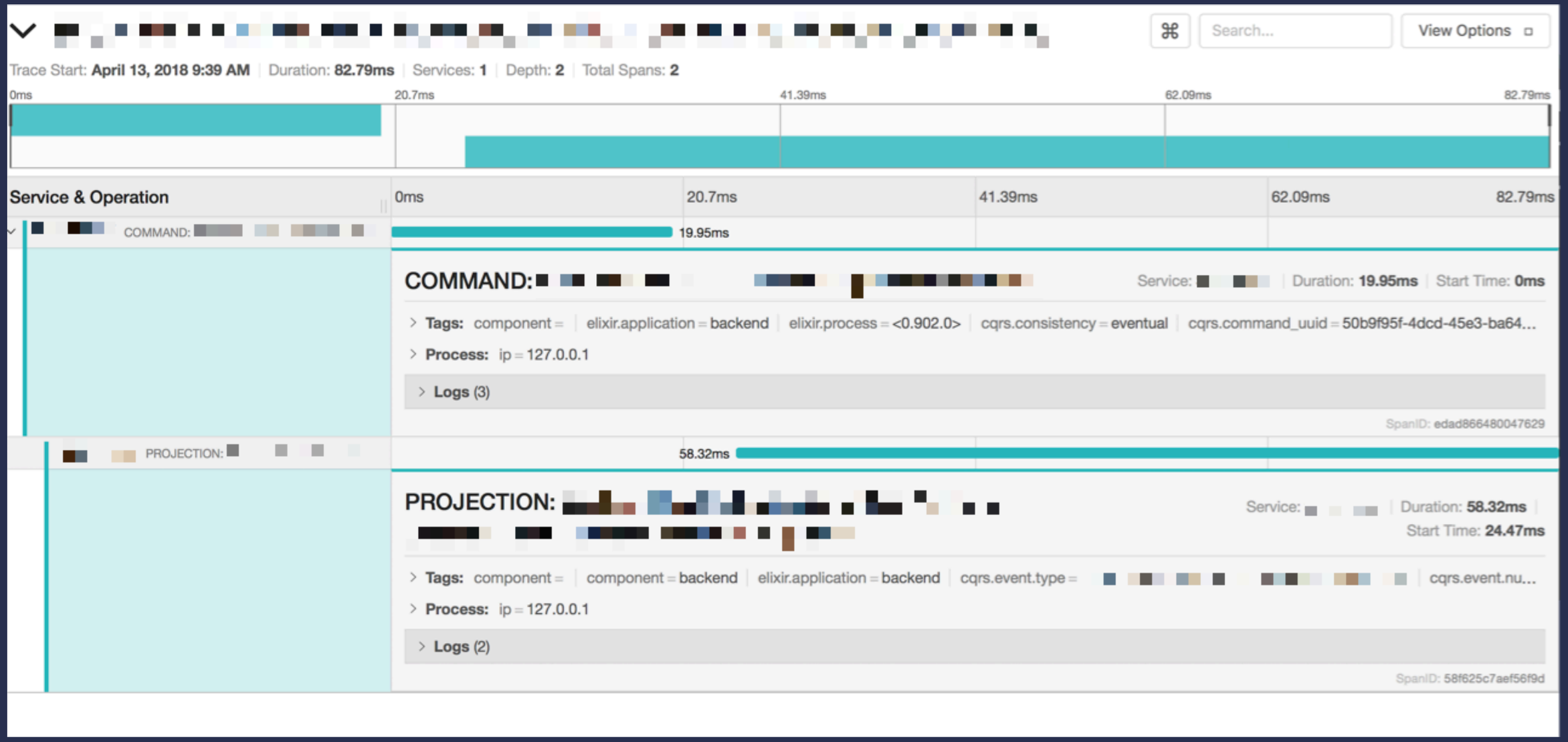
1. Inject tracing contexts "into" events
2. Save tracing contexts globally
3. Generate span ID from input
4. Use unique event data as span IDs

WHAT DID  DO?

INSERTED SPAN IDS INTO EVENT METADATA

USED THESE IDS DOWNSTREAM TO CORRELATE SPANS

CUSTOM INTER-PROCESS PROPAGATION


Search...
View Options
Trace Start: April 13, 2018 9:39 AM | Duration: 82.79ms | Services: 1 | Depth: 2 | Total Spans: 2
0ms
20.7ms
41.39ms
62.09ms
82.79ms
Service & Operation
COMMAND:
19.95ms
COMMAND:
Service:
Duration: 19.95ms
Start Time: 0ms
Tags: component = | elixir.application = backend | elixir.process = <0.902.0> | cqrs.consistency = eventual | cqrs.command_uuid = 50b9f95f-4dcd-45e3-ba64...
Process: ip = 127.0.0.1
Logs (3)
SpanID: edad866480047629
PROJECTION:
58.32ms
PROJECTION:
Service:
Duration: 58.32ms
Start Time: 24.47ms
Tags: component = | component = backend | elixir.application = backend | cqrs.event.type = | cqrs.event.nu...
Process: ip = 127.0.0.1
Logs (2)
SpanID: 58f625c7aef56f9d

TAKE AWAY?

- Tracing is hard
- OpenTracing has neat ideas
- Tooling in Elixir/Erlang still has need for improvement

Questions?

SLIDES ON GITHUB[1]

[1] https://github.com/Zeeker/talks