TRACING
IN ELIXIR

Asked myself what this talk should be about ...

- Technical details?
- Scaling?
- Issues?

-> Story

LET'S MEET ALEXANDER

Backend Developer at a random compomany, let's say

Smallcentrix

We need more diagnostics!

– Pretty much everybody

- elixir diagnostics (Healthcare)
- elixir tracing

- Where to start? What makes sense?
- Ask experienced colleague, Senior Developer, if you want

Look at OpenTracing, you scrub!

Pretty much the exchange "Before we look at OT, let's ask ourselfs ..."

WHAT EXACTLY IS TRACING?

Question!

"Tracing is about ..."

ANALYZING TRANSACTIONS

"But ..."

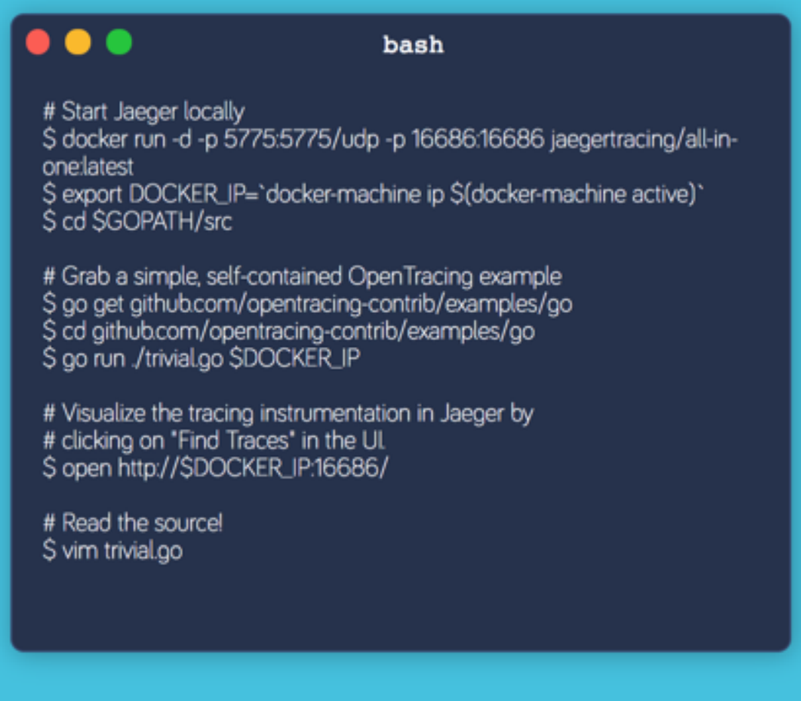
TRACING
IS HARD

Question: Why?

- 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

Unreasonable to abide to single vendor specifications

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

Not going to explain OT in detail, couldn't do it anyway

TIMED OPERATIONS

CALLED SPANS

RELATIONS BETWEEN SPANS

CONTEXT PROPAGATION

Getting Span information from one system into another

"For this OpenTracing standardizes ..."

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

"Ok, so this is OpenTracing ..."

WHAT ABOUT ELIXIR?

OTTER

OPENTRACING TOOLKIT FOR ERLANG

PARTIAL OPENTRACING IMPLEMENTATION

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

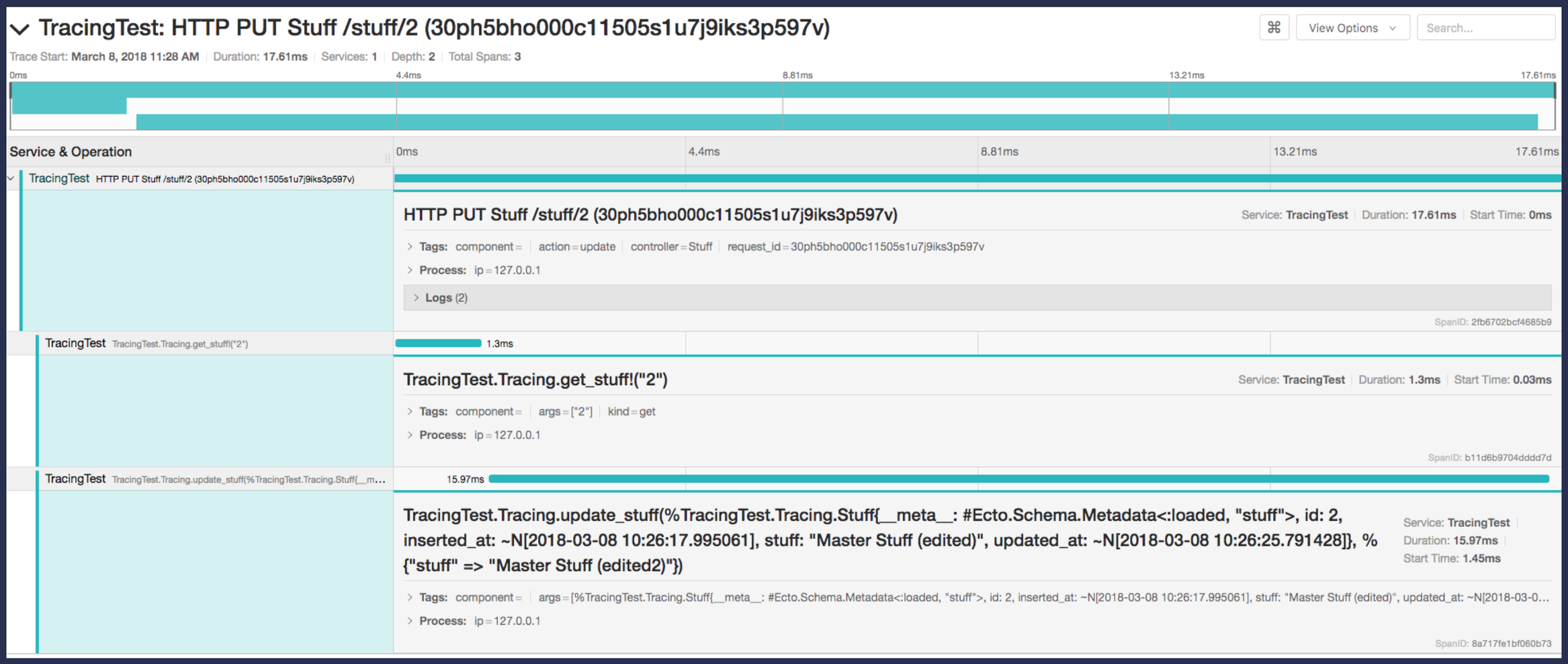
Active: Per-Process span "caching", not crossing process boundaries


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



Screenshot from Jaeger UI, distributed tracing system by Uber

Everybody was like
"Yeeeaaaah"

INTEGRATE IT INTO THE PROJECT!

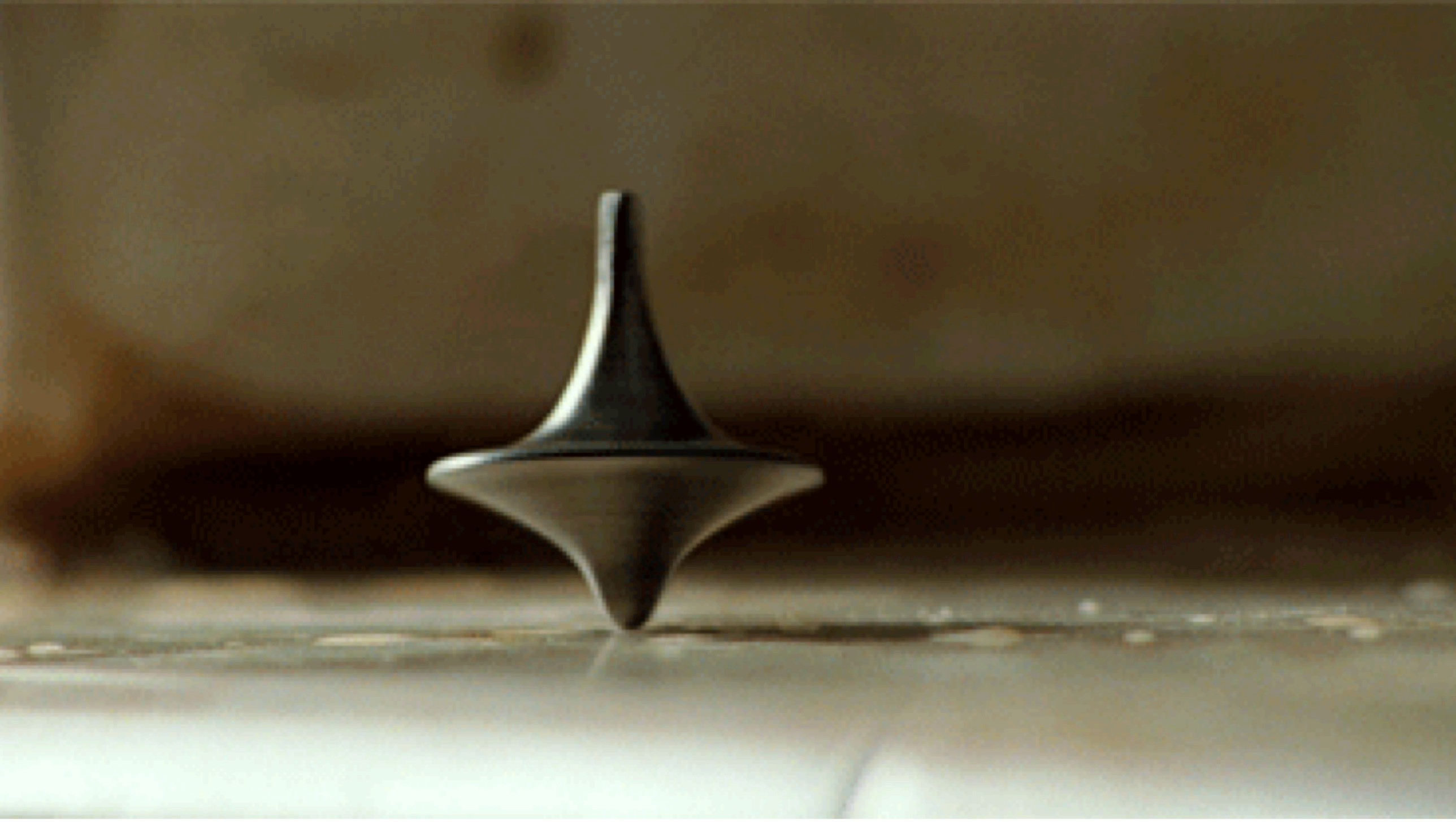
Must be easy now, right?

TRACING PLUG

1. Create span and add some tags
2. Wait for response
3. Finish span

Now he actually wants to see what a request DOES!

ALEX WANTED TO GO DEEPER!



"But for that, he had to ..."

CROSS PROCESS BOUNDARIES

"Why? Because, you see, the project was ..."

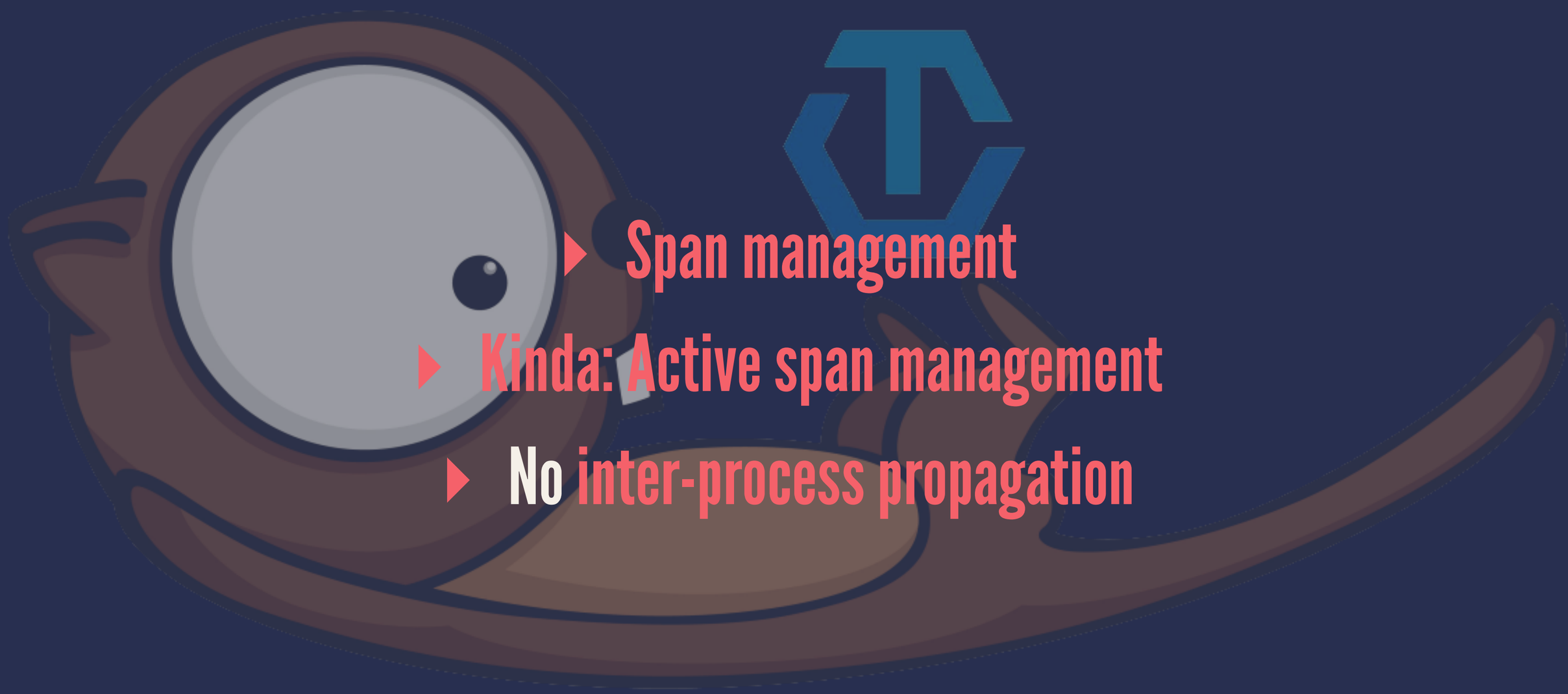
EVENT-DRIVEN

He wanted to see what an event triggered!

"Which meant, Alex needed ..."

INTER-PROCESS PROPAGATION

Remember OTTER? Partial Implementation? What exactly did it gave Alex?


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

"Well …"

"Here Alexander ..."


panicked screeching

OPTIONS?

Question: Ask this!

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

1. Where?
2. scaling issues
3. collision risk (that bad?), obscure
4. UUIDs but we need 64bit integer

WHAT DID 👨‍💻 DO?

INSERTED SPAN IDS INTO EVENT METADATA

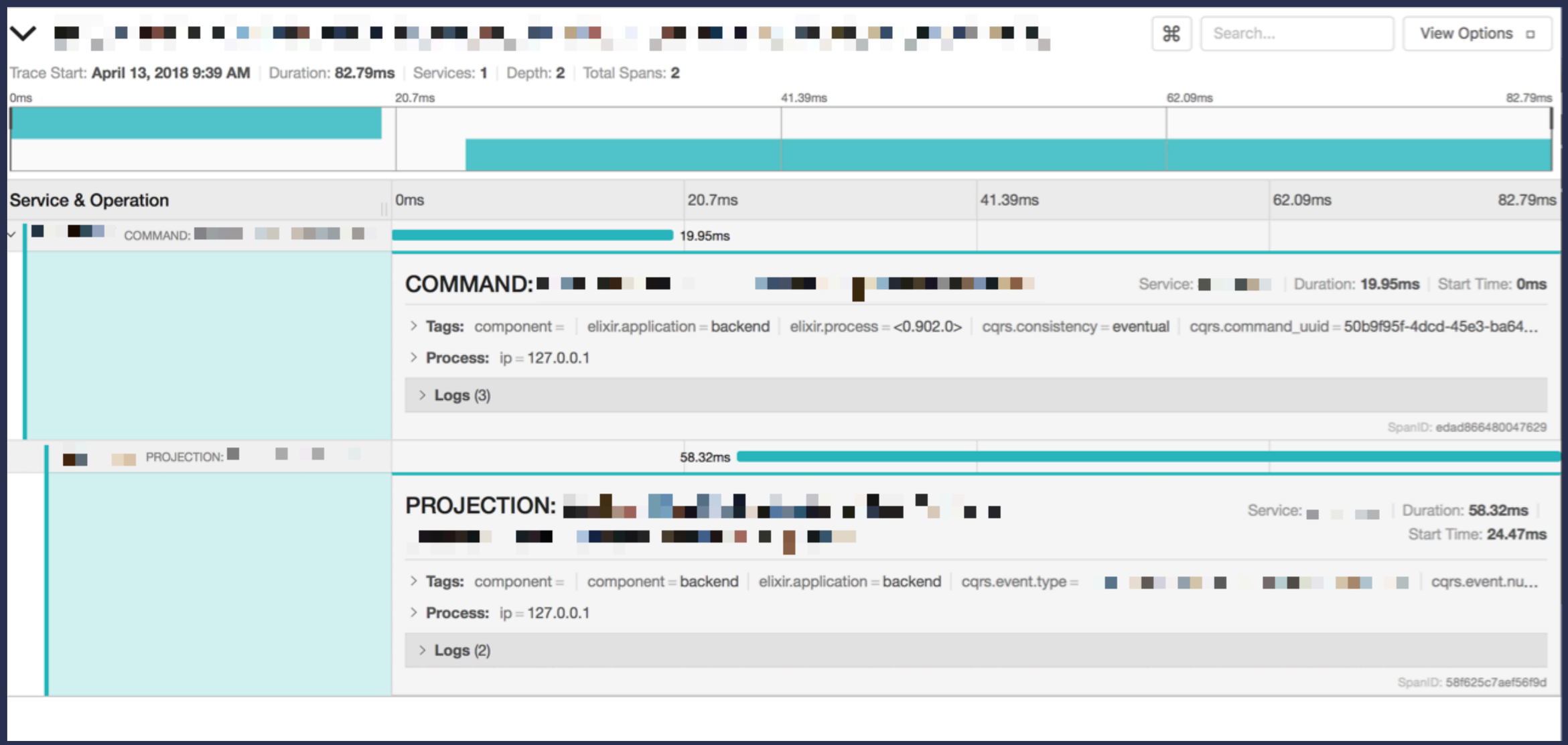
1. Option

USED THESE IDS DOWNSTREAM TO CORRELATE SPANS

Basically built ...

CUSTOM INTER-PROCESS PROPAGATION

"Now finally he could ..."


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

Used those downstream to correlate the spans to each other