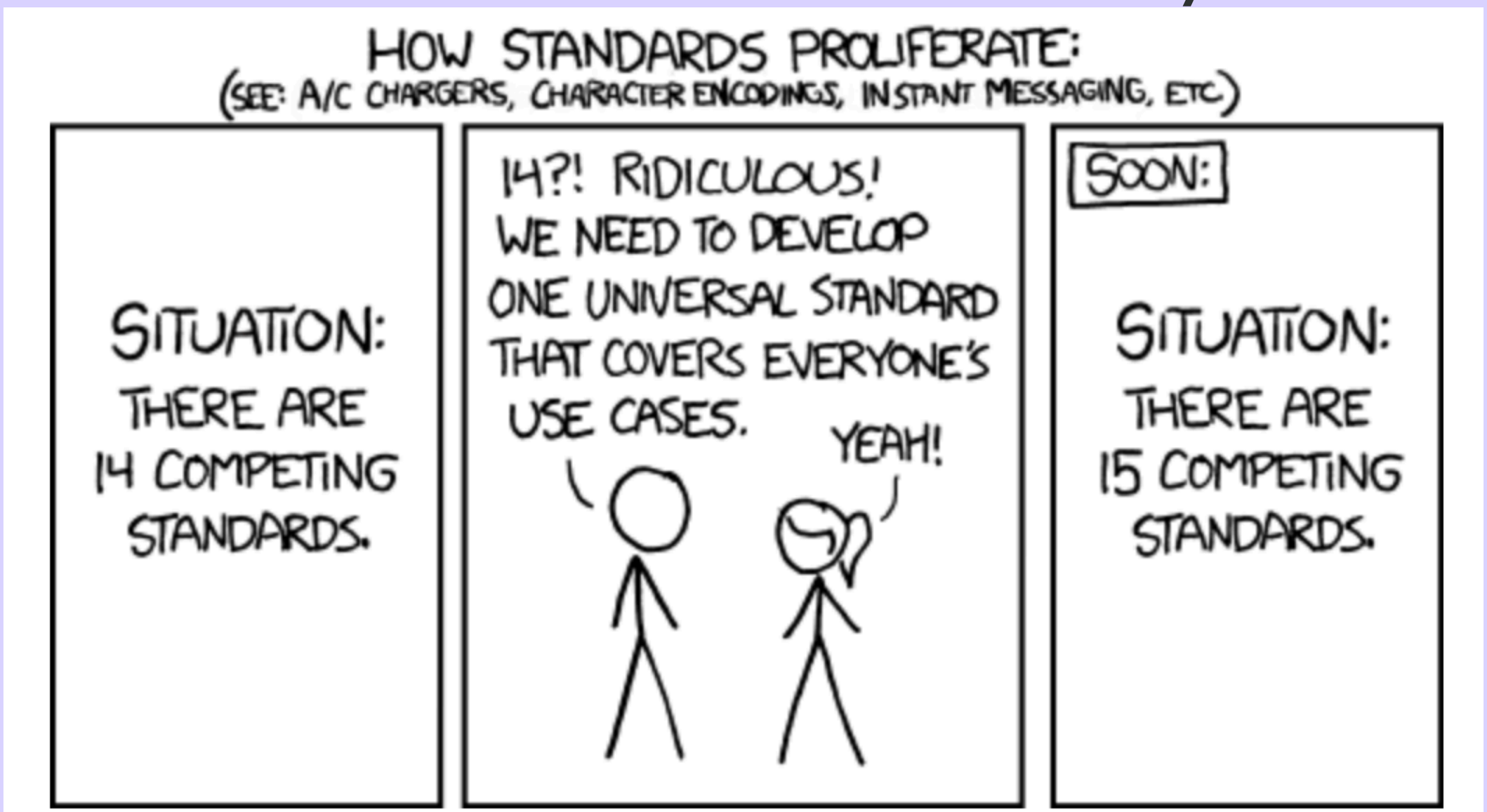
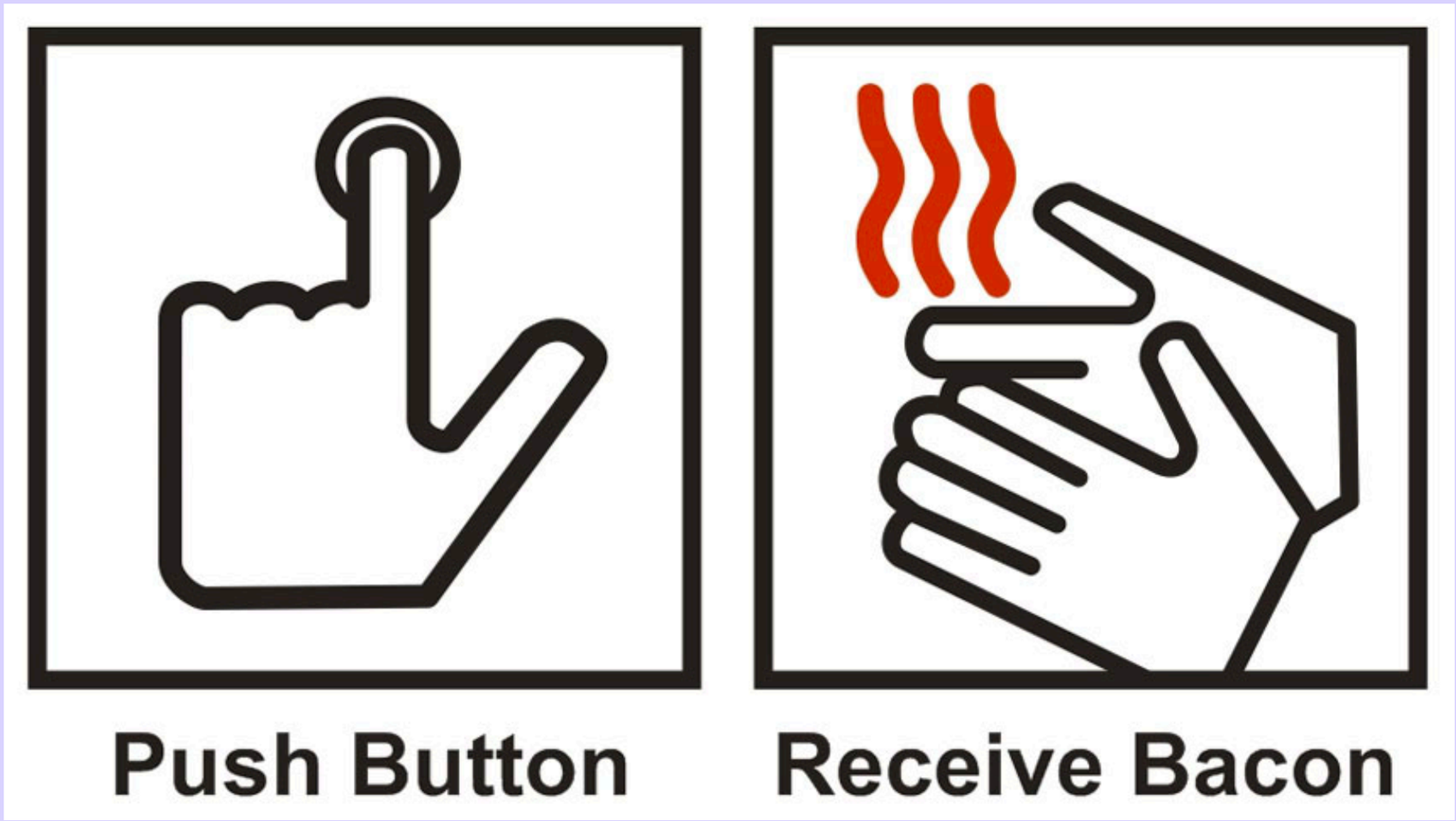
OpenTelemetry

an XKCD 927 Success Story



Observability



https://knowyourmeme.com/memes/push-button-receive-bacon

Observables

Observables

- No bacon.

Observables

- No bacon.
- Warm air coming out.

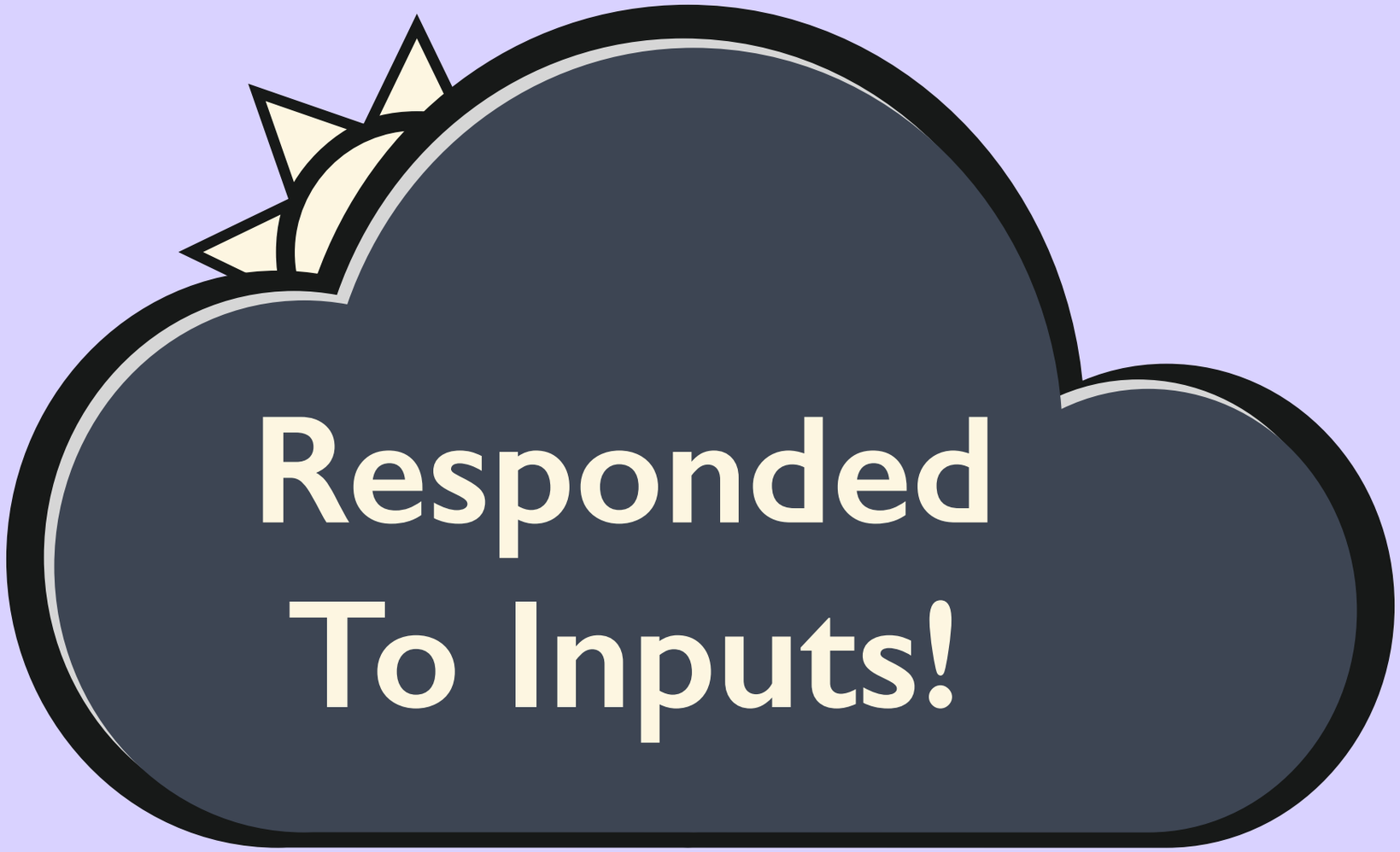
Observables

- No bacon.
- Warm air coming out.
- Lots of noise.

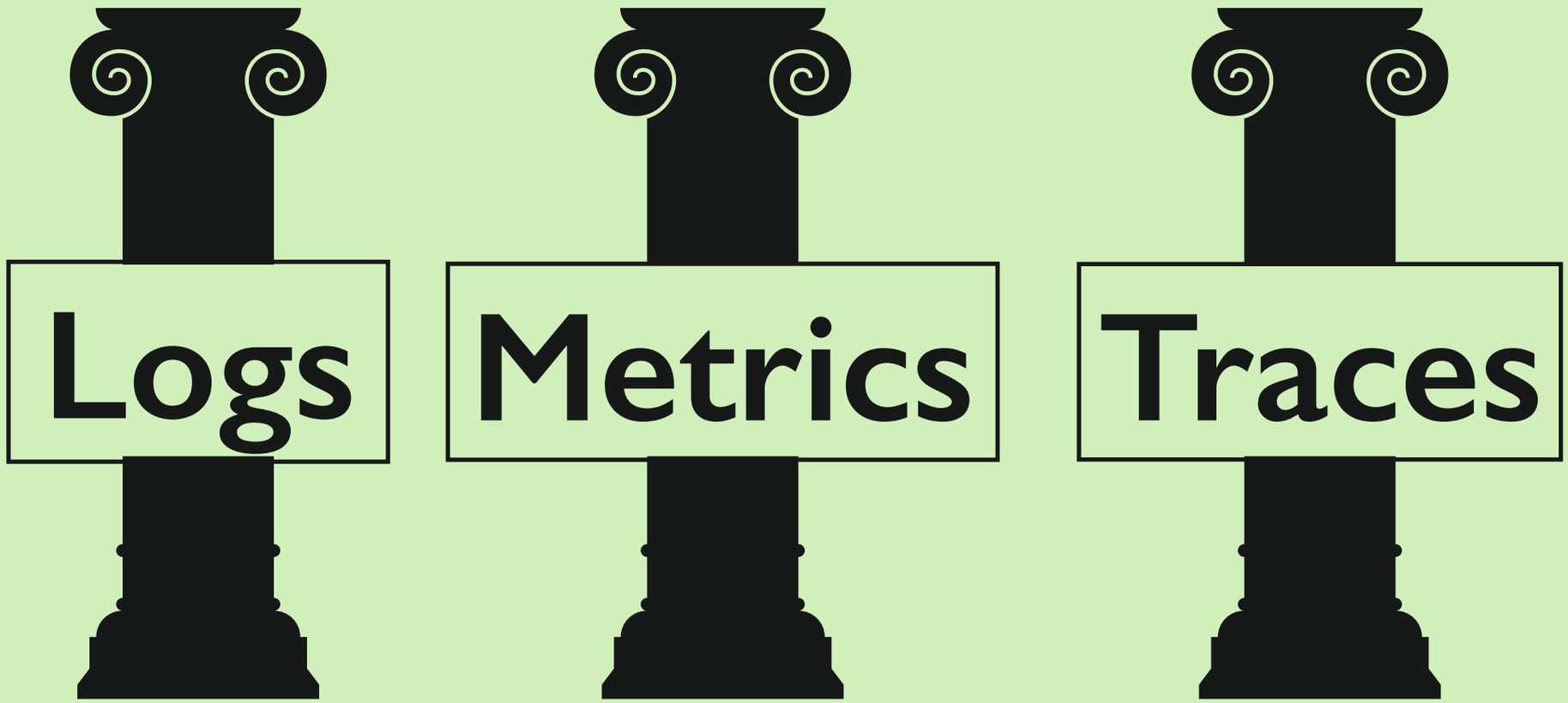
Observables

- No bacon.
- Warm air coming out.
- ~~Lots of noise.~~

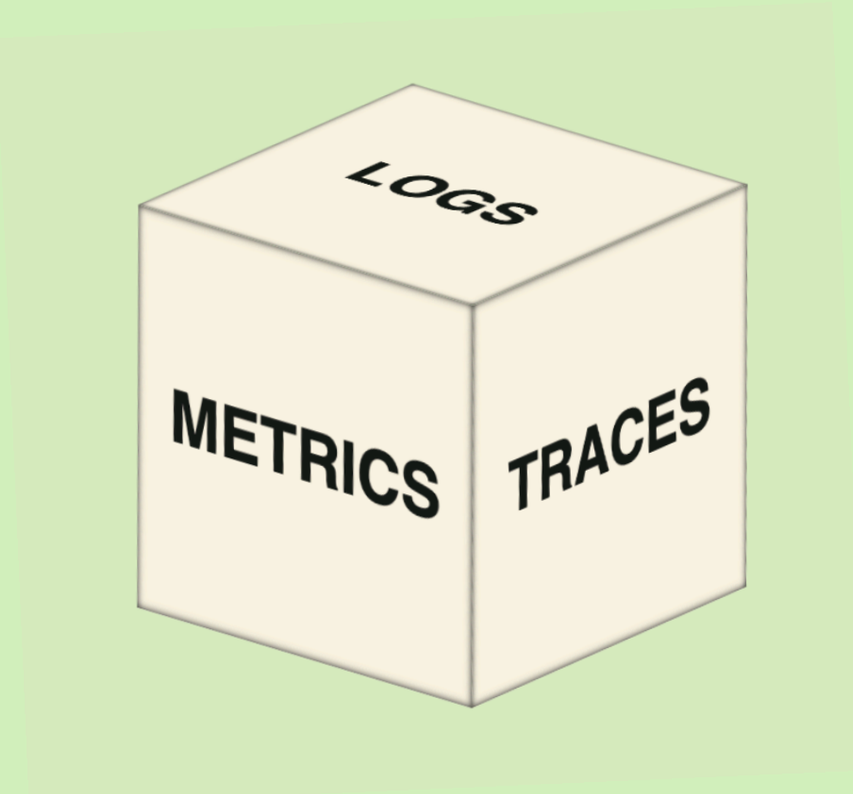
Observables



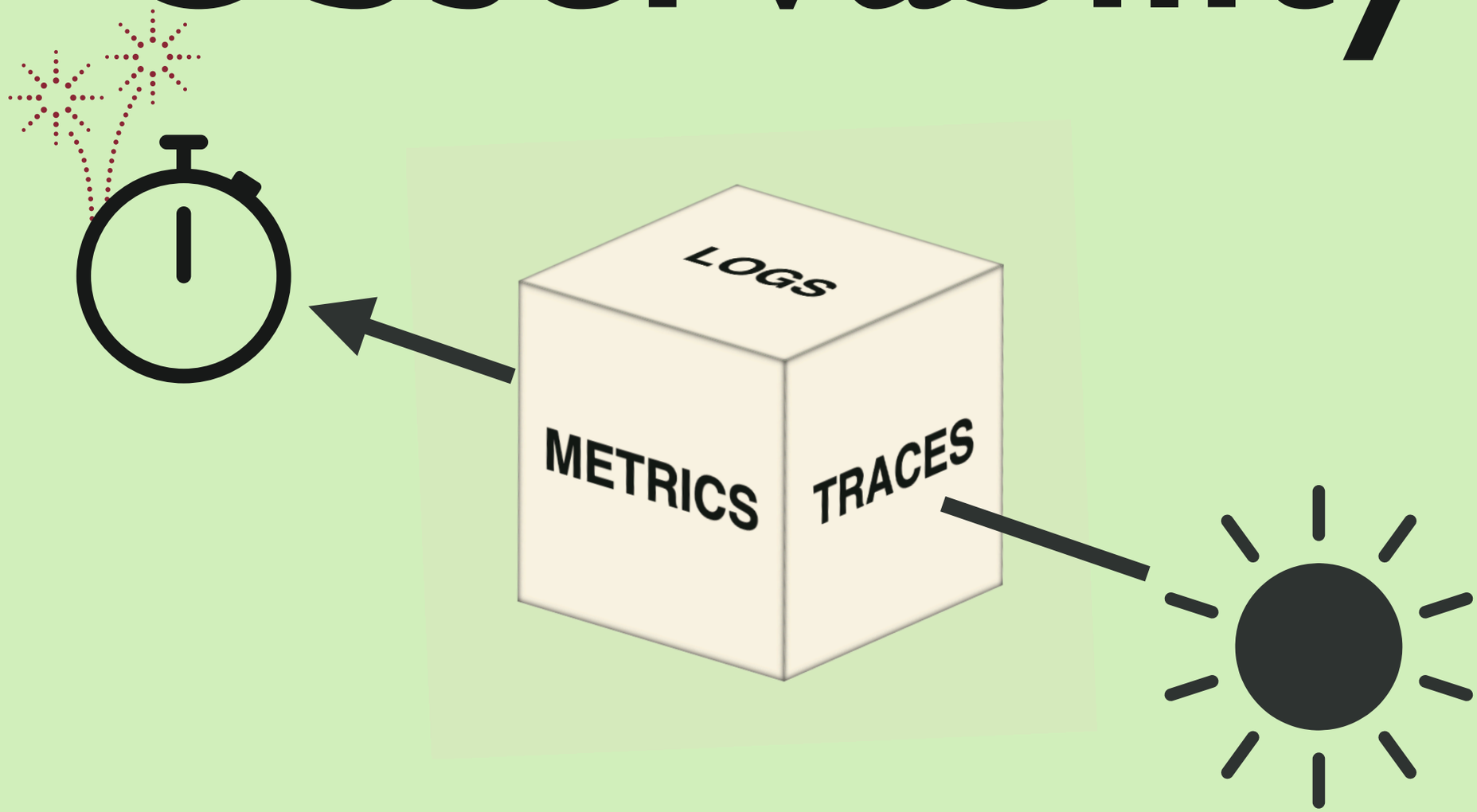
Observability



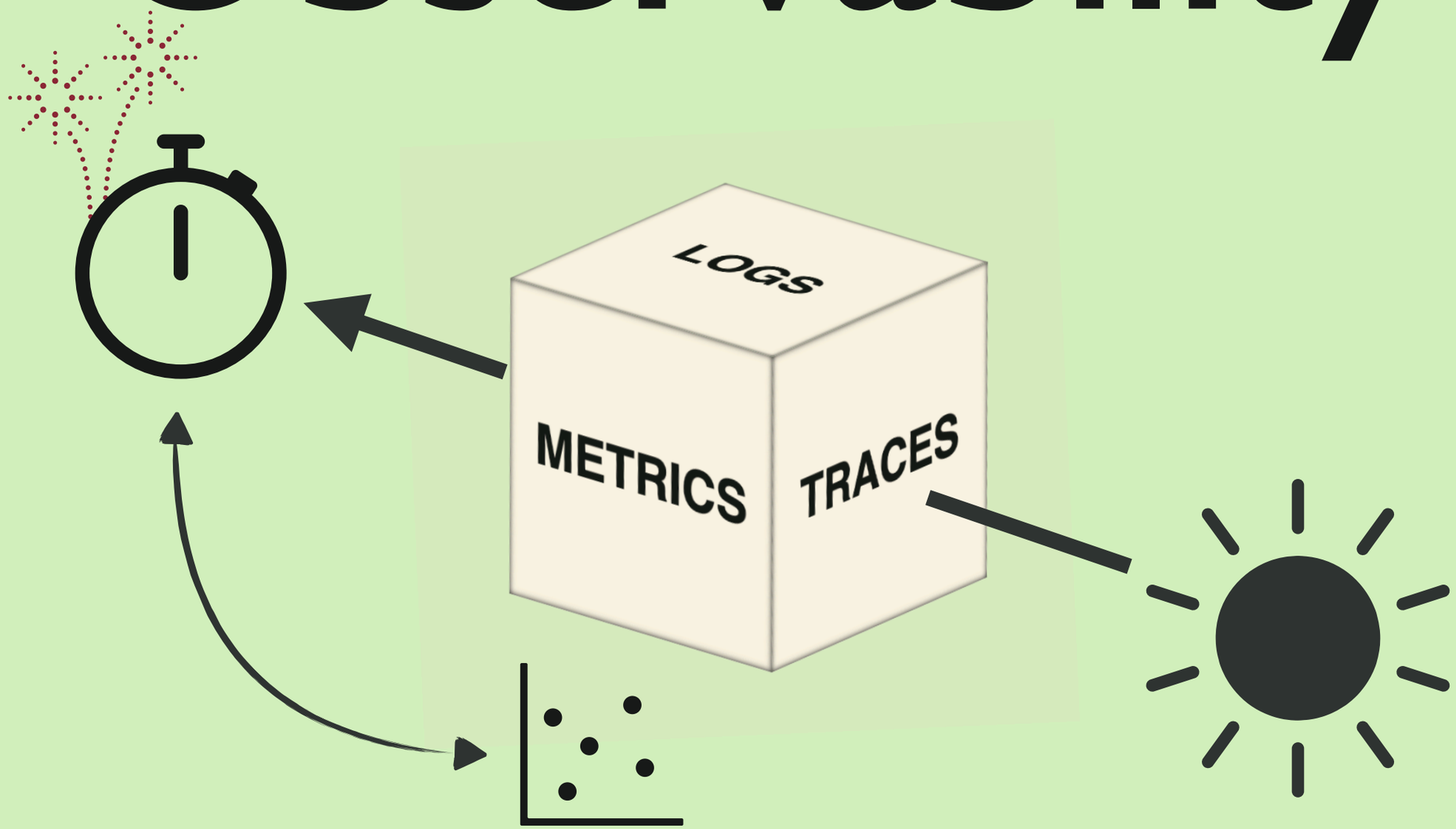
Observability



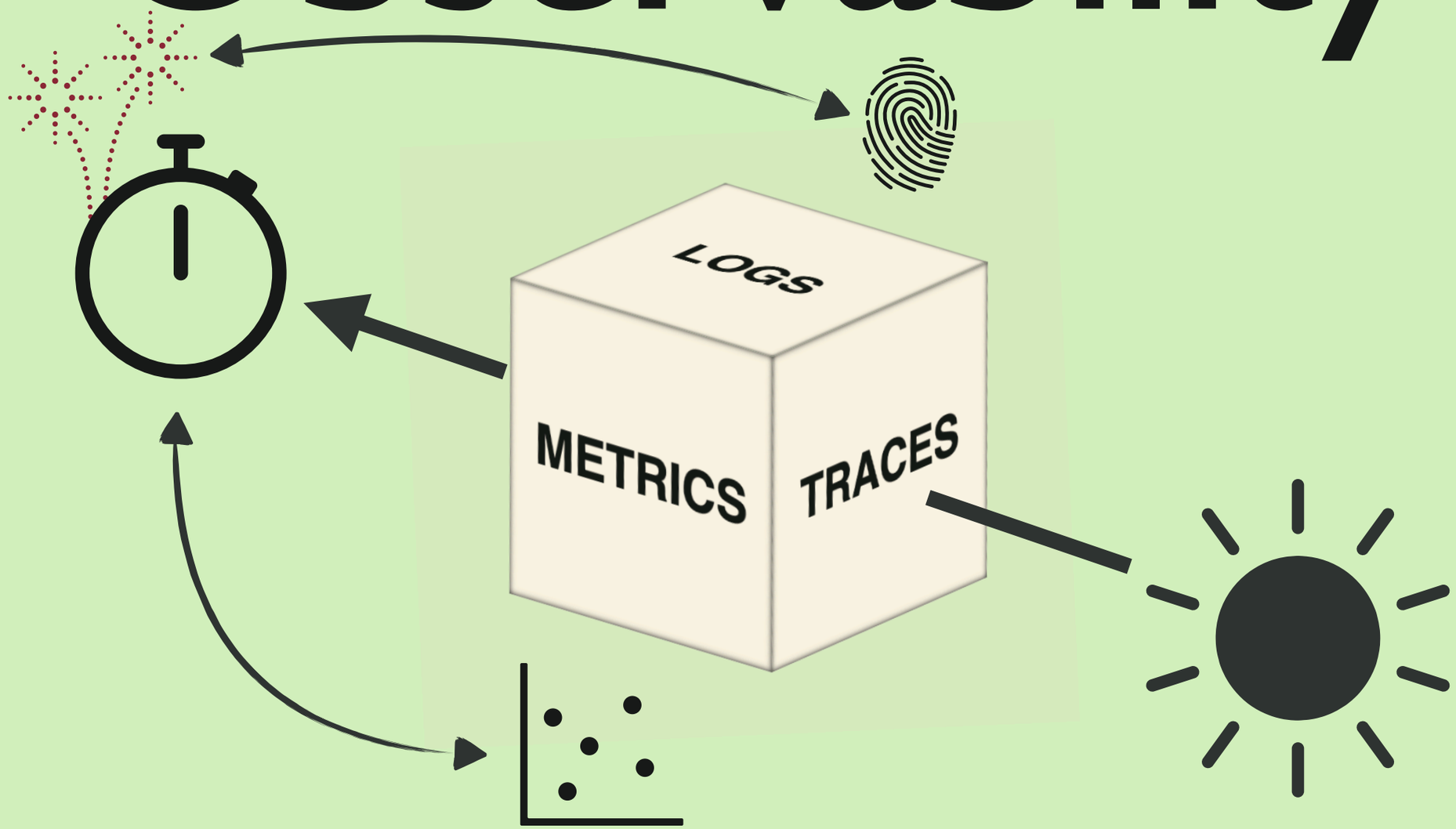
Observability



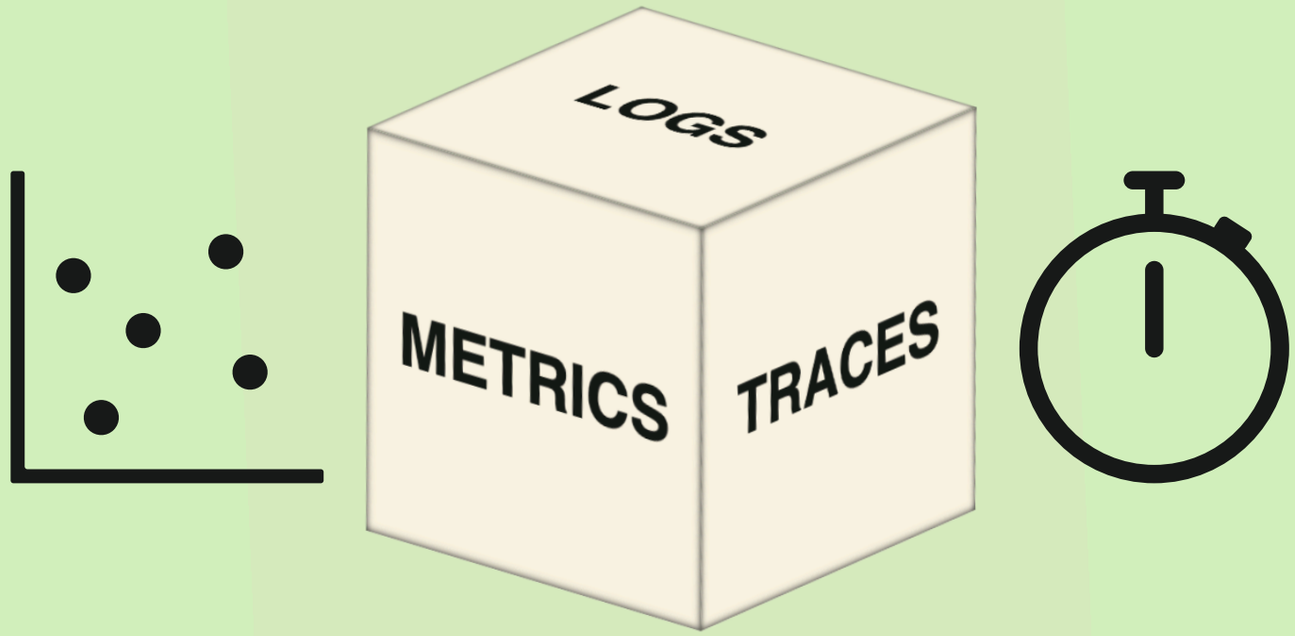
Observability



Observability



Observability



Traces

- Tell a Story
- Particular Requests
- Tags & Data
- Distributed, Sampled

Metrics

- Tell a Story
- *Type*s of Operations
- Time and Tags

Logs

- Tell a Story
- Fine Details
- Structured Data
- Collected/Indexed

(Distributed)

Tracing Theory

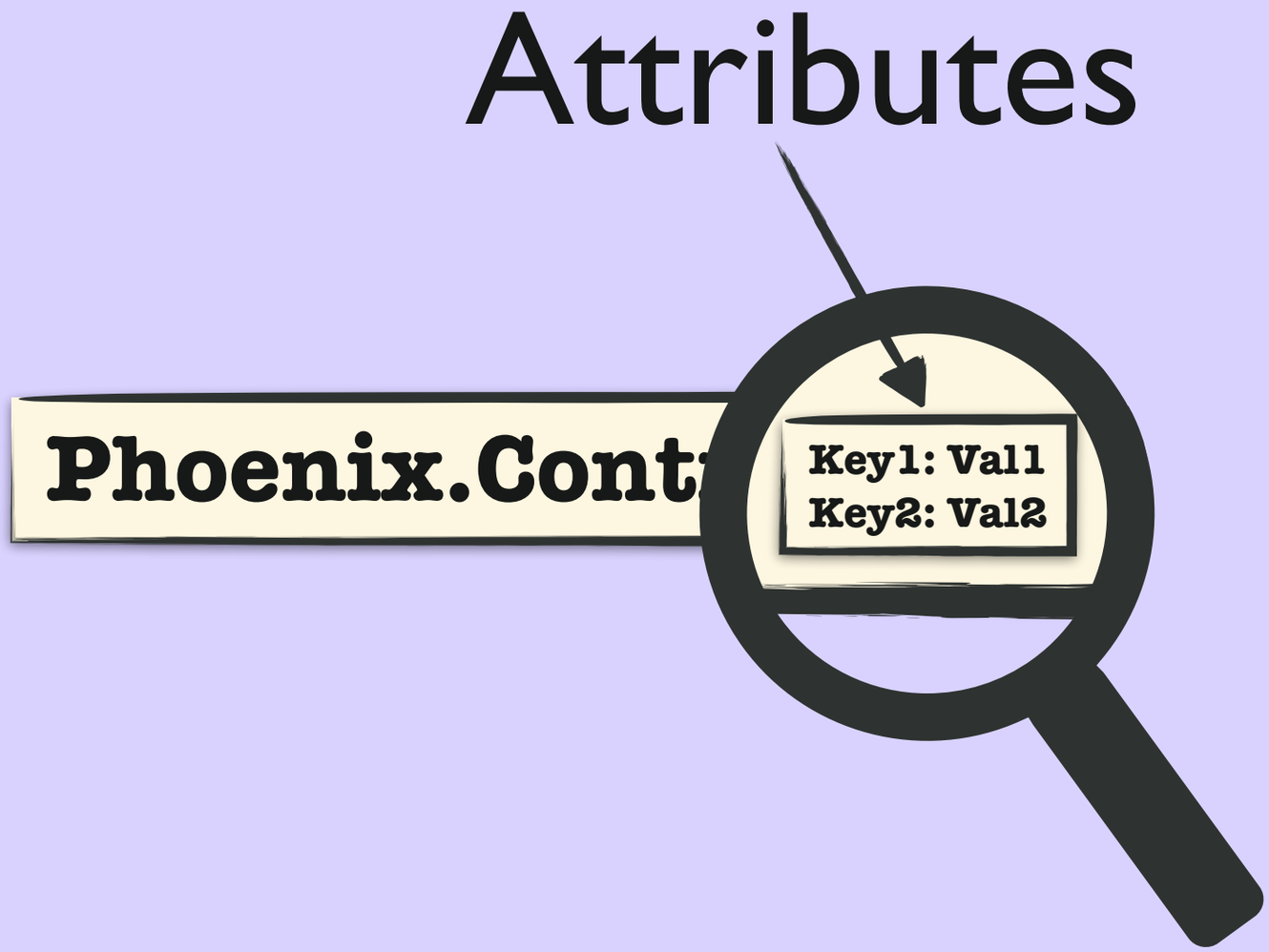
Span

StartTime

EndTime

Phoenix.Controller

Name


Attributes
Phoenix.Cont
Key1: Val1
Key2: Val2

Phoenix.Endpoint

Phoenix.Router

Phoenix.Controller

Ecto.Query

Phoenix.View

Jason.encode

Trace

Phoenix.Endpoint

Phoenix.Router

Phoenix.Controller

Ecto.Query

Phoenix.View

Jason.encode

Root Span

Trace

Phoenix.Endpoint

Phoenix.Router

Phoenix.Controller

Ecto.Query

Phoenix.View

Jason.encode

Trace

Phoenix.Controller

HTTP.Get

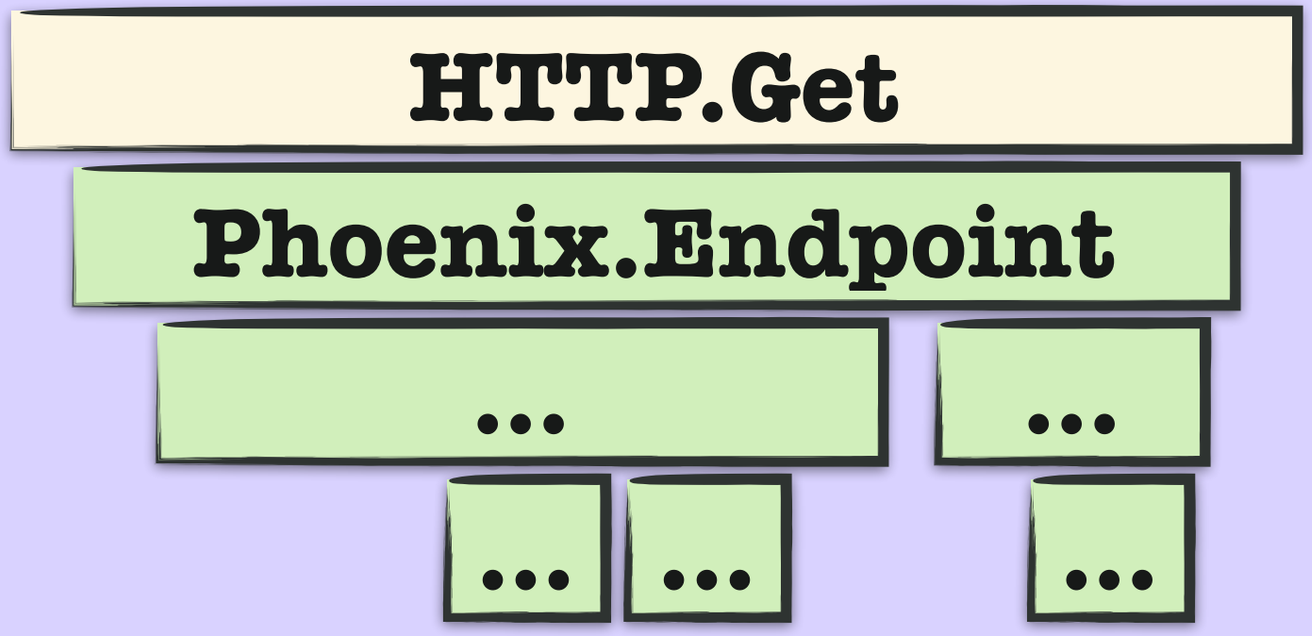
handle_responses

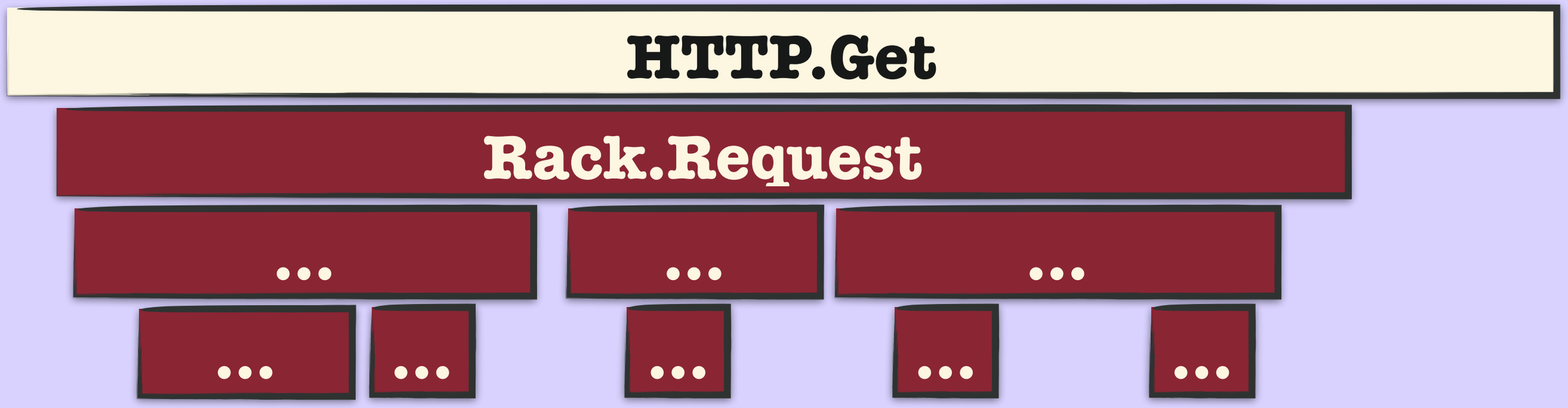
...

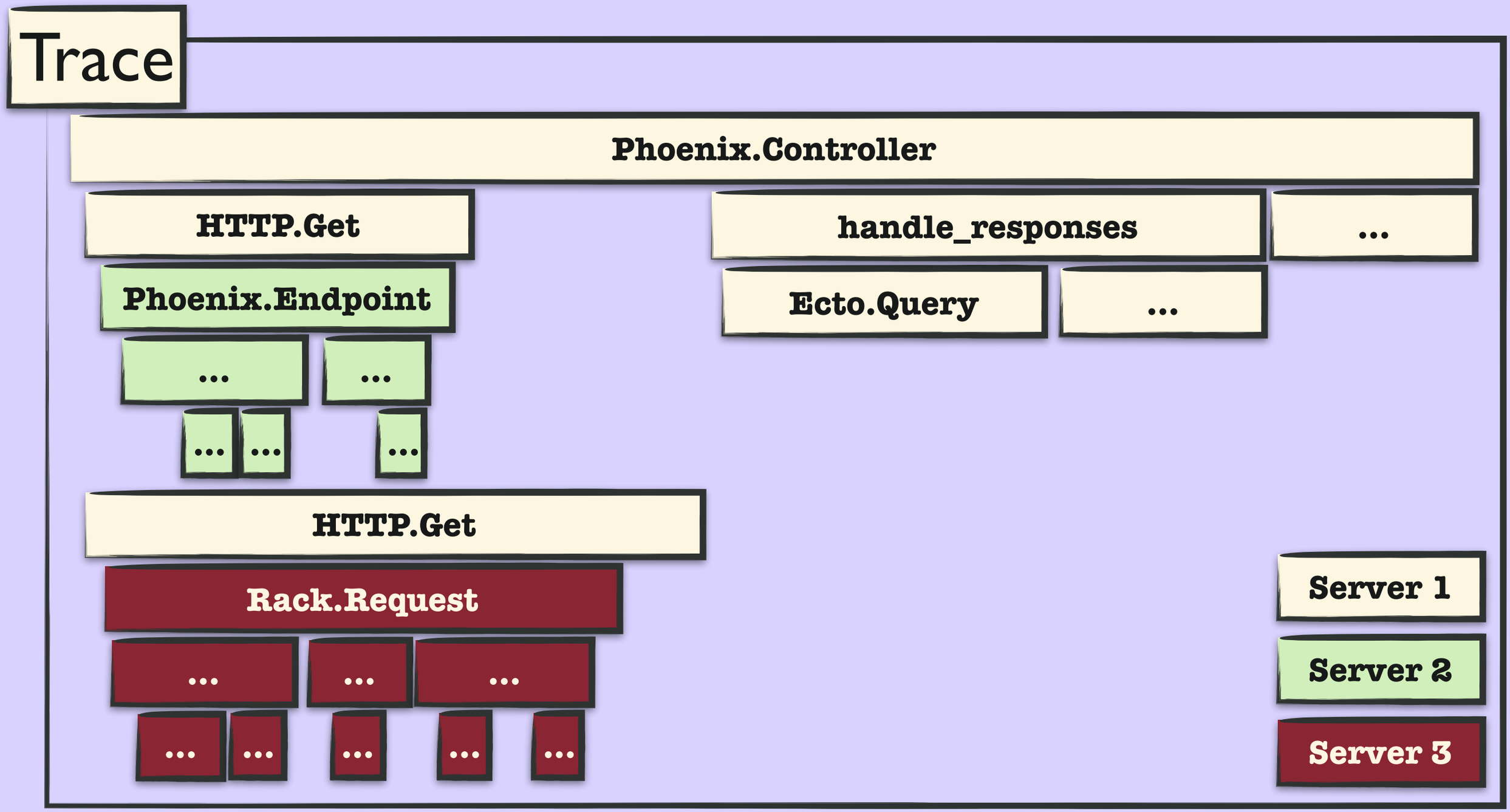
Ecto.Query

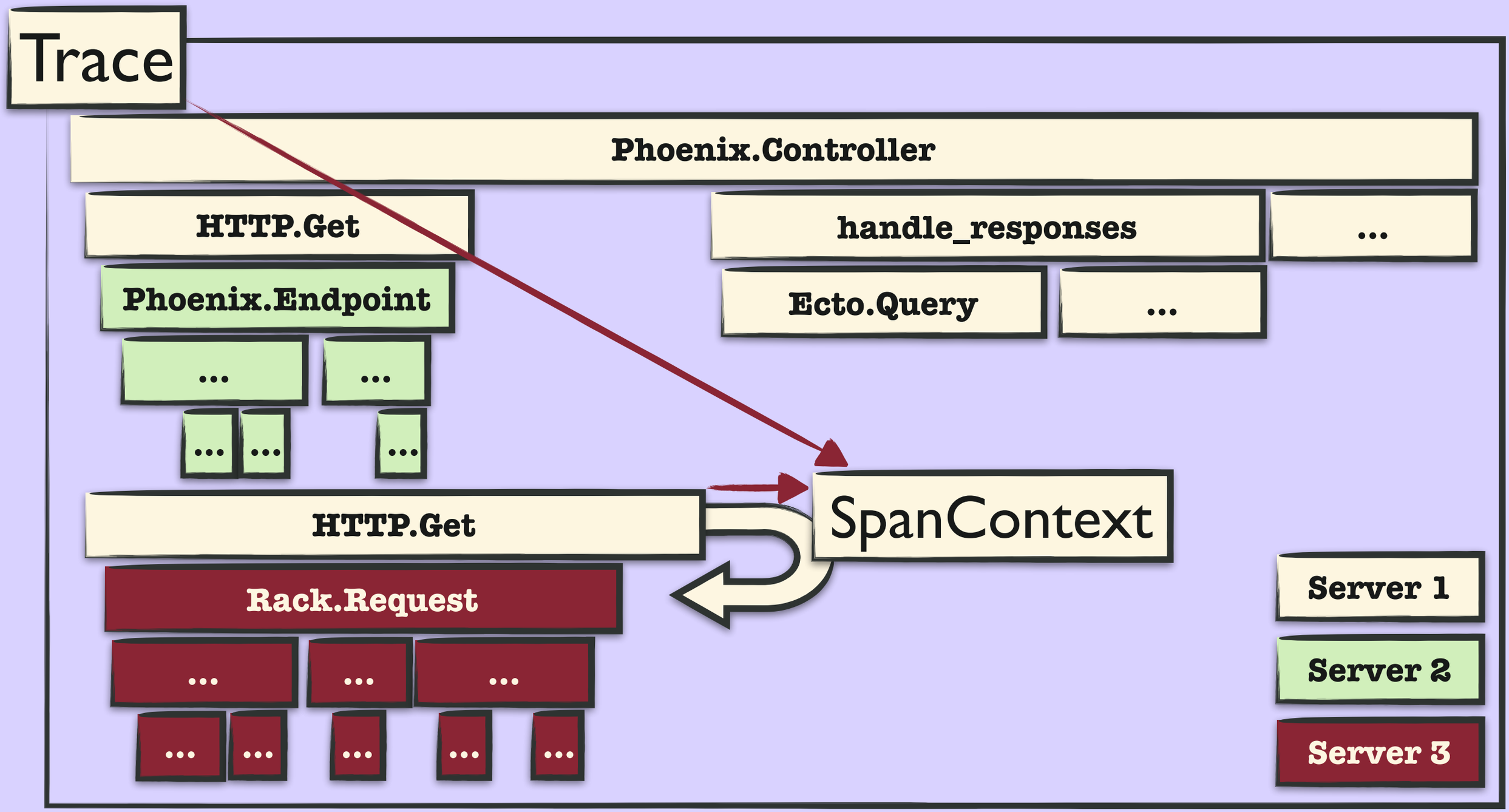
...

HTTP.Get


HTTP.Get
Phoenix.Endpoint
...
...
...
...
...


HTTP.Get
Rack.Request
...
...
...
...
...
...
...


Trace
Phoenix.Controller
HTTP.Get
Phoenix.Endpoint
...
...
...
...
...
HTTP.Get
Rack.Request
...
...
...
...
...
...
...
handle_responses
...
Ecto.Query
...
Server 1
Server 2
Server 3


Trace
Phoenix.Controller
HTTP.Get
Phoenix.Endpoint
...
...
...
...
...
handle_responses
...
Ecto.Query
...
HTTP.Get
Rack.Request
...
...
...
...
...
...
...
...
SpanContext
Server 1
Server 2
Server 3

Trace

Phoenix.Controller

HTTP.Get

Phoenix.Endpoint

...

...

...

...

...

handle_responses

...

Ecto.Query

...

HTTP.Get

Rack.Request

...

...

...

...

...

...

...

Sp

Sampled: ✓
State: [...]

Server 1

Server 2

Server 3

Context Propagation

https://www.w3.org/TR/trace-context/

§ 2.2.2 Field value

This section uses the Augmented Backus-Naur Form (ABNF) notation of RFC5234, including the HEXDIG rules from that document.

```
value           = version "-" version-format
version         = 2HEXDIG   ; this document assumes version 00. Version 255 is forbidden
```

The value is US-ASCII encoded (which is UTF-8 compliant). Character – is used as a delimiter between fields.

Version (`version`) is a 1 byte representing an 8-bit unsigned integer. Version 255 is invalid. Current specification assumes the `version` is set to `00`.

The following `version-format` definition is used for version `00`.

```
version-format   = trace-id "-" span-id "-" trace-flags

trace-id         = 32HEXDIG  ; 16 bytes array identifier. All zeroes forbidden
span-id          = 16HEXDIG  ; 8 bytes array identifier. All zeroes forbidden
trace-flags      = 2HEXDIG   ; 8 bit flags. Currently only one bit is used. See below for de
```

Trace

Phoenix.Controller

HTTP.Get

¯_(ツ)_/¯

handle_responses

...

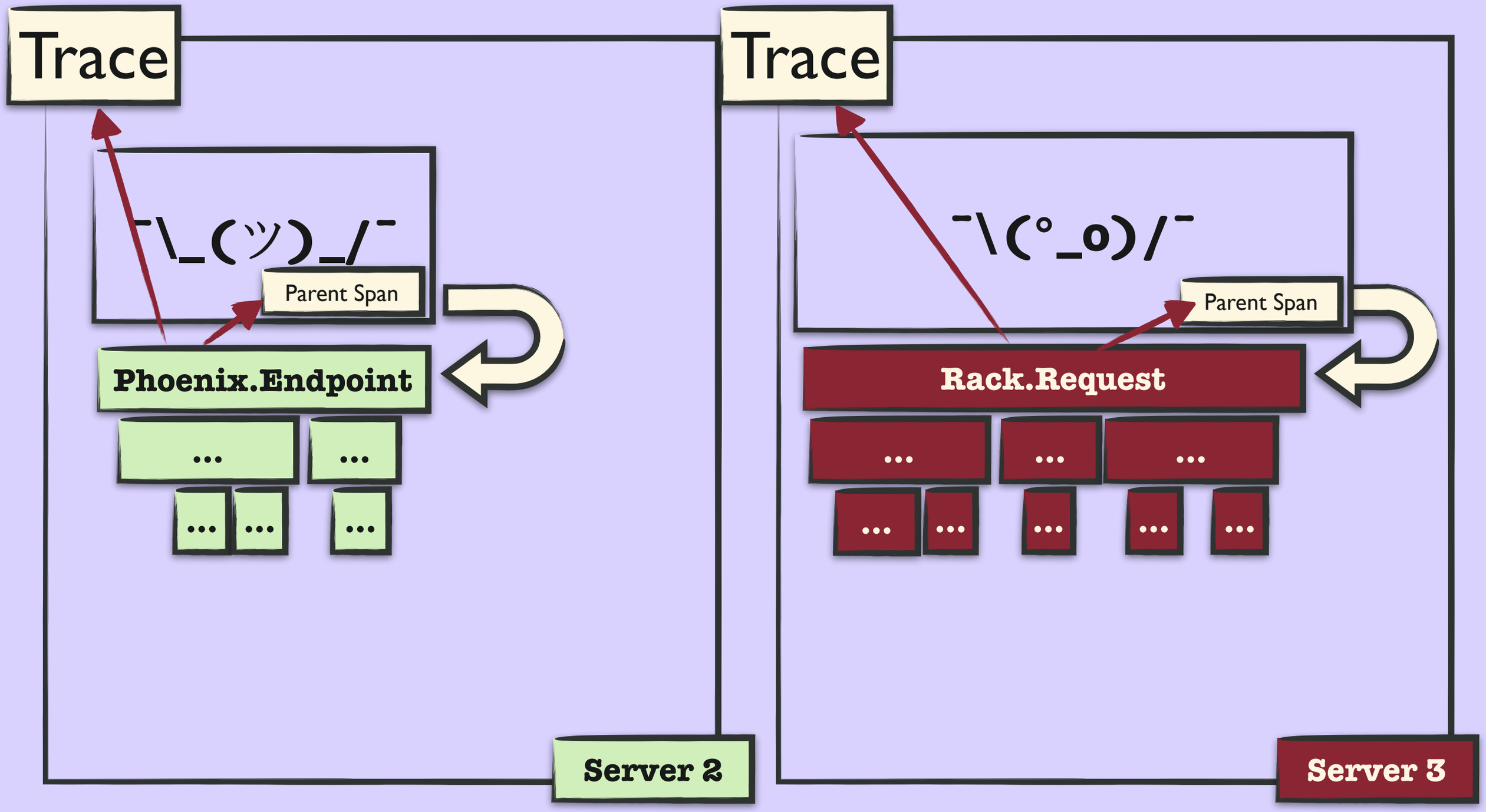
Ecto.Query

...

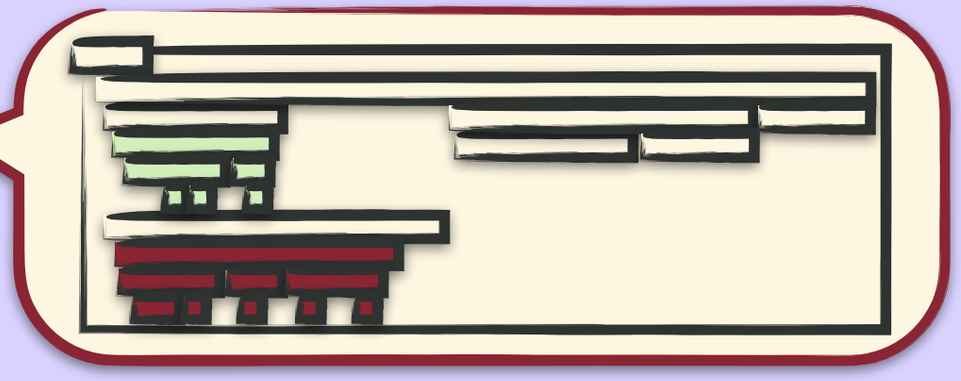
HTTP.Get

¯\(°_o)/¯

Server 1


Trace
¯_(ツ)_/¯
Parent Span
Phoenix.Endpoint
Server 2
Trace
¯\(°_o)/¯
Parent Span
Rack.Request
Server 3

Collector 🤔



Trace

Phoenix.Endpoint

...

...

...

...

...

Server 2

Trace

Rack.Request

...

...

...

...

...

...

...

...

Server 3

Trace

Phoenix.Controller

HTTP.Get

handle_responses

...

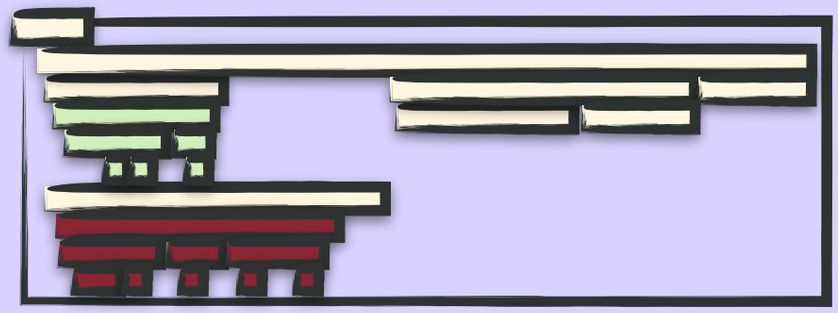
Ecto.Query

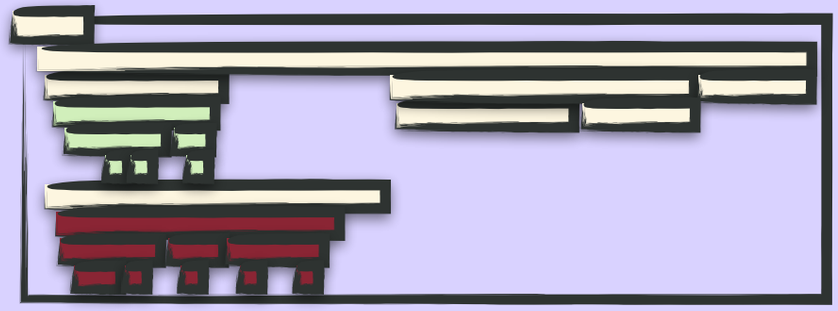
...

HTTP.Get

Server 1

Collector 🤔

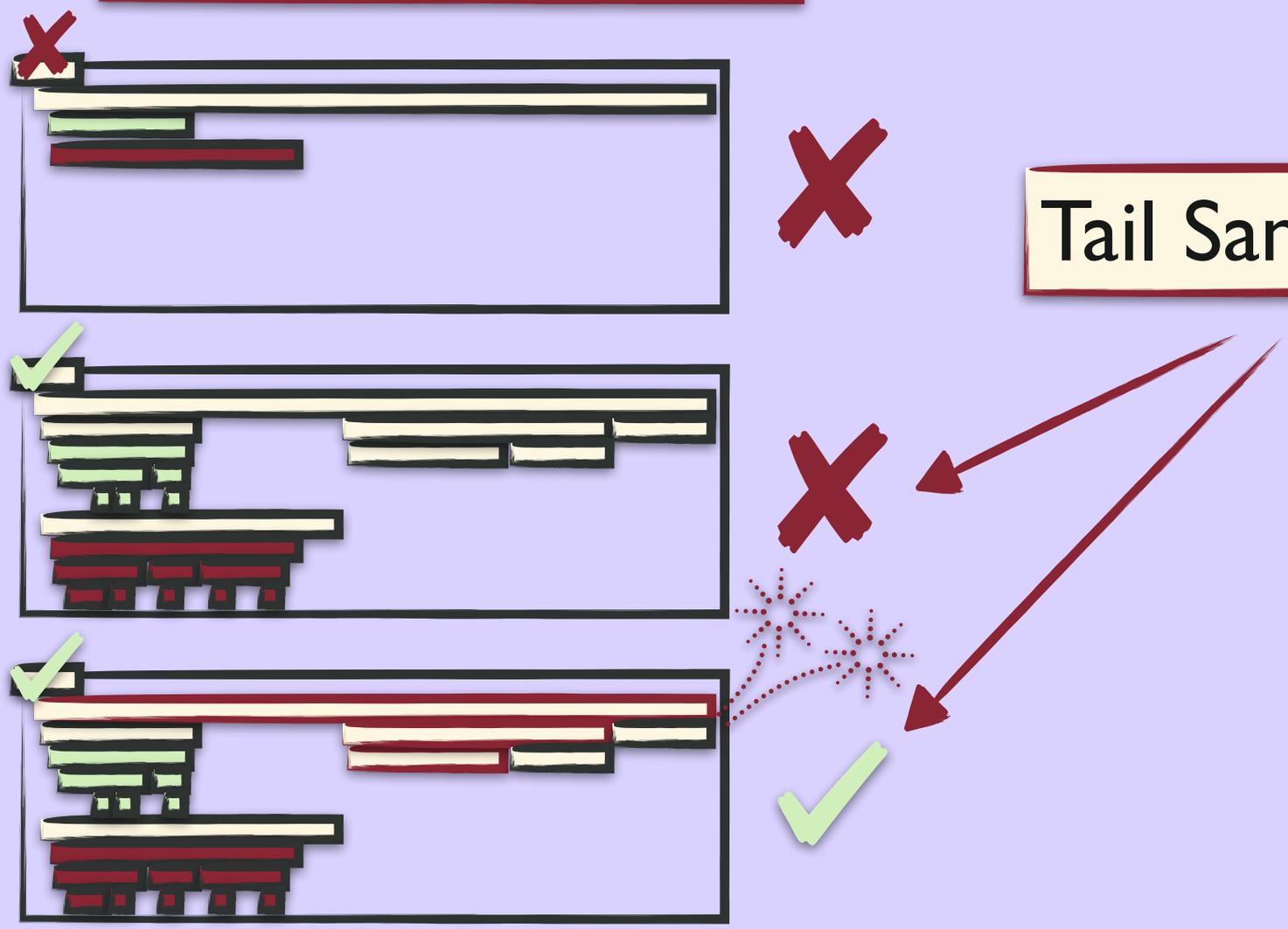




Tracer:
Head Sampling

Collector 🤔

Collector 🤔



Tail Sampling

BEAM Built-In Tracing

(*not* used for distributed tracing)

erlang:trace_pattern

http://erlang.org/doc/man/erlang.html#trace_pattern-2

ERLANG

Erlang Run-Time System Application (ERTS)
Reference Manual
Version 10.3

- User's Guide
- Reference Manual
- Release Notes
- PDF
- Top

- Expand All
- Contract All

Table of Contents

- erl_prim_loader
- erlang
  - Top of manual page
  - abs/1
  - adler32/1
  - adler32/2
  - adler32_combine/3

```
erlang:trace_pattern(MFA, MatchSpec) -> integer() >= 0
```

Types

```
MFA = trace_pattern_mfa() | send | 'receive'
MatchSpec =
    (MatchSpecList :: trace_match_spec()) |
    boolean() |
    restart |
    pause
trace_pattern_mfa() = {atom(), atom(), arity() | '_'} |
on_load
trace_match_spec() =
    [{[term()] | '_' | match_variable(), [term()],
[term()]}]
match_variable() = atom()
 Approximation of '$1' | '$2' | '$3' | ...
```

The same as `erlang:trace_pattern(Event, MatchSpec, [])`, retained for backward compatibility.

```
erlang:trace_pattern(MFA :: send, MatchSpec, FlagList :: []) ->
                        integer() >= 0
```

OTP 19.0

Types

```
MatchSpec = (MatchSpecList :: trace_match_spec()) |
boolean()
trace_match_spec() =
    [{[term()] | '_' | match_variable(), [term()],
[term()]}]
match_variable() = atom()
 Approximation of '$1' | '$2' | '$3' | ...
```

recon_trace

http://ferd.github.io/recon/recon_trace.html

Recon index recon recon_alloc recon_lib recon_rec recon_trace

Module recon_trace

- Description
- Data Types
- Function Index
- Function Details

`recon_trace` is a module that handles tracing in a safe manner for single Erlang nodes, currently for function calls only.

Authors: Fred Hebert (`mononcqc@ferd.ca`) [*web site:* `http://ferd.ca/`].

Description

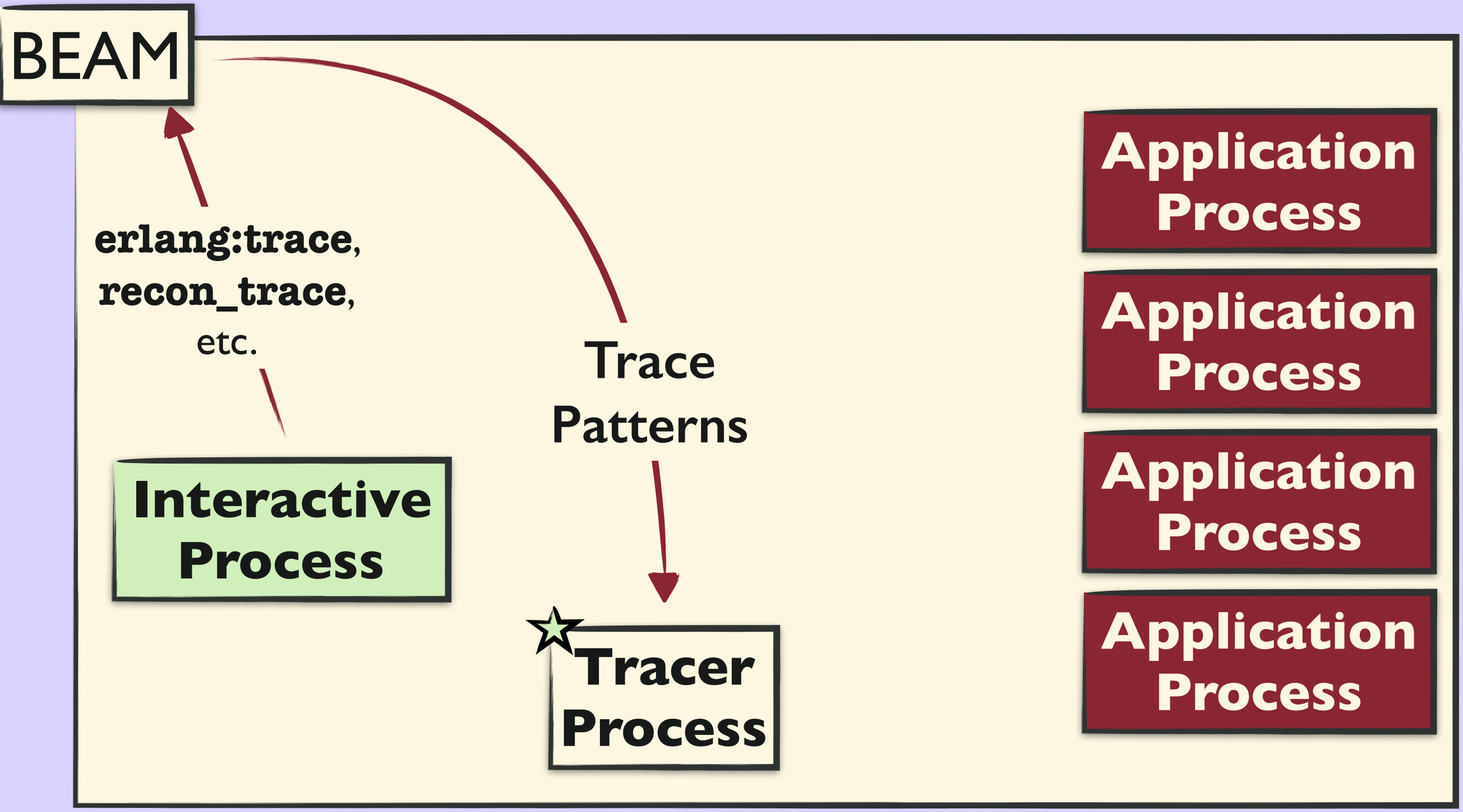
`recon_trace` is a module that handles tracing in a safe manner for single Erlang nodes, currently for function calls only. Functionality includes:

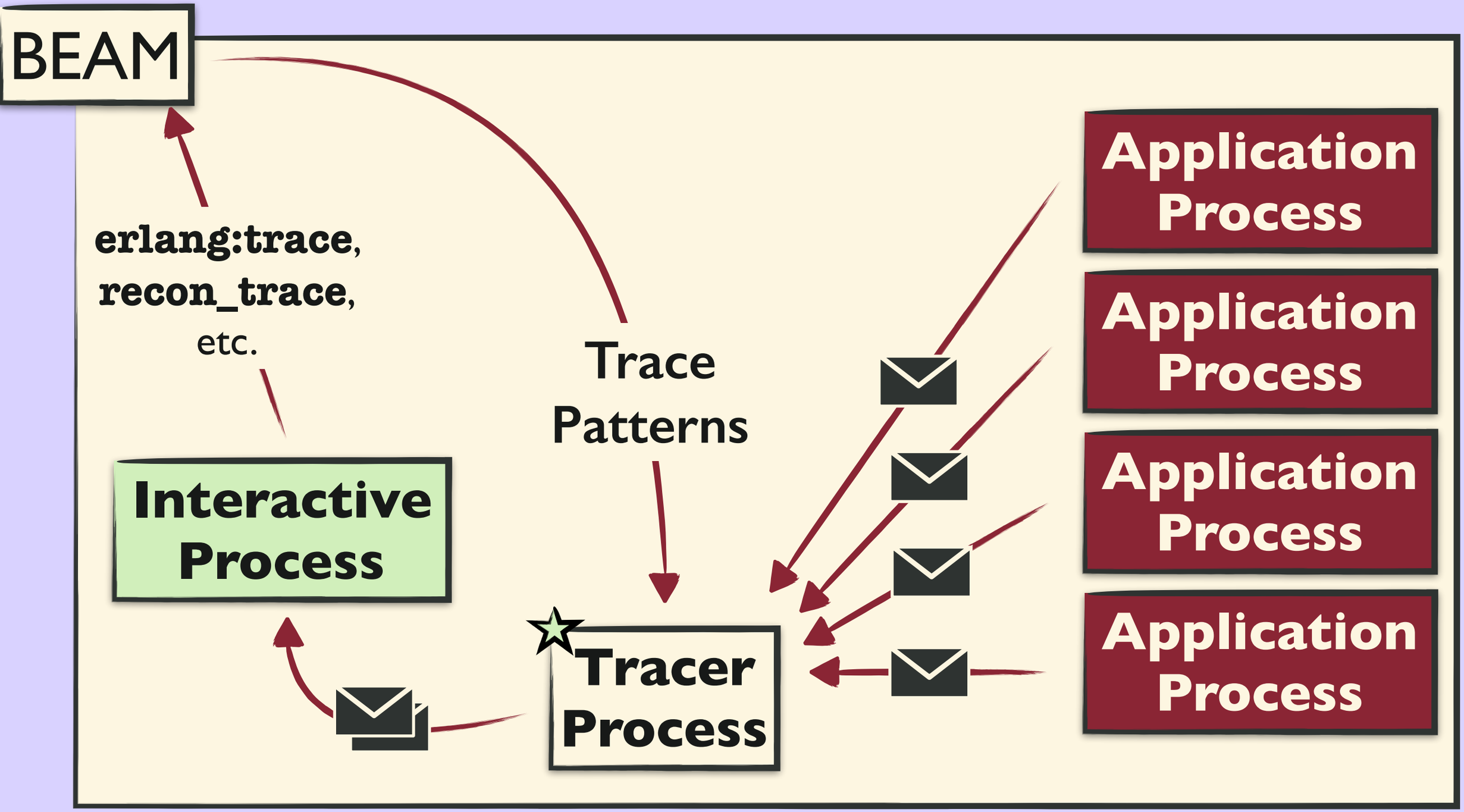
- Nicer to use interface (arguably) than `dbg` or trace BIFs.
- Protection against dumb decisions (matching all calls on a node being traced, for example)
- Adding safe guards in terms of absolute trace count or rate-limitting
- Nicer formatting than default traces

Tracing Erlang Code

The Erlang Trace BIFs allow to trace any Erlang code at all. They work in two parts: pid specifications, and trace patterns.

Pid specifications let you decide which processes to target. They can be specific pids, `all` pids, `existing` pids, or `new` pids (those not spawned at the time of


BEAM
erlang:trace,
recon_trace,
etc.
Interactive Process
Trace Patterns
Tracer Process
Application Process
Application Process
Application Process
Application Process


BEAM
erlang:trace,
recon_trace,
etc.
Trace
Patterns
Application
Process
Application
Process
Application
Process
Application
Process
Interactive
Process
Tracer
Process

✅ Prod-Safe

👩‍💻 Interactive

1️⃣ Per BEAM

⭕ Local Only

Recon — index · recon · recon_alloc · recon_lib · recon_rec · recon_trace

Module recon_trace

- Description
- Data Types
- Function Index
- Function Details

`recon_trace` is a module that handles tracing in a safe manner for single Erlang nodes, currently for function calls only.

Authors: Fred Hebert (`mononcqc@ferd.ca`) [*web site:* `http://ferd.ca/`].

Description

`recon_trace` is a module that handles tracing in a safe manner for single Erlang nodes, currently for function calls only. Functionality includes:

- Nicer to use interface (arguably) than `dbg` or trace BIFs.
- Protection against dumb decisions (matching all calls on a node being traced, for example)
- Adding safe guards in terms of absolute trace count or rate-limitting
- Nicer formatting than default traces

Tracing Erlang Code

The Erlang Trace BIFs allow to trace any Erlang code at all. They work in two parts: pid specifications, and trace patterns.

Pid specifications let you decide which processes to target. They can be specific pids, `all` pids, `existing` pids, or `new` pids (those not spawned at the time of the function call).

The trace patterns represent functions. Functions can be specified in two parts: specifying the modules, functions, and arguments, and then with Erlang match specifications to add constraints to arguments (see `calls/3` for details).

ERLANG

Erlang Run-Time System Application (ERTS)
Reference Manual
Version 10.3

User's Guide
Reference Manual
Release Notes
PDF
Top

Expand All
Contract All

Table of Contents

- erl_prim_loader
- erlang
  - Top of manual page
  - abs/1
  - adler32/1
  - adler32/2
  - adler32_combine/3
  - append_element/2
  - apply/2
  - apply/3
  - atom_to_binary/2
  - atom_to_list/1

```
erlang:trace_pattern(MFA, MatchSpec) -> integer() >= 0

  Types

    MFA = trace_pattern_mfa() | send | 'receive'
    MatchSpec =
        (MatchSpecList :: trace_match_spec()) |
        boolean() |
        restart |
        pause
    trace_pattern_mfa() = {atom(), atom(), arity() | '_'} |
    on_load
    trace_match_spec() =
        [{[term()] | '_' | match_variable(), [term()],
    [term()]}]
    match_variable() = atom()
     Approximation of '$1' | '$2' | '$3' | ...
```

The same as `erlang:trace_pattern(Event, MatchSpec, [])`, retained for backward compatibility.

```
erlang:trace_pattern(MFA :: send, MatchSpec, FlagList :: []) ->
                        integer() >= 0

  Types

    MatchSpec = (MatchSpecList :: trace_match_spec()) |
    boolean()
    trace_match_spec() =
        [{[term()] | '_' | match_variable(), [term()],
    [term()]}]
    match_variable() = atom()
     Approximation of '$1' | '$2' | '$3' | ...
```

Sets trace pattern for **message sending**. Must be combined with `erlang:trace/3` to set the `send` trace flag for one or more processes. By default all messages sent from `send` traced processes are traced. To limit traced send events based on the message content, the sender and/or the receiver, use `erlang:trace_pattern/3`.

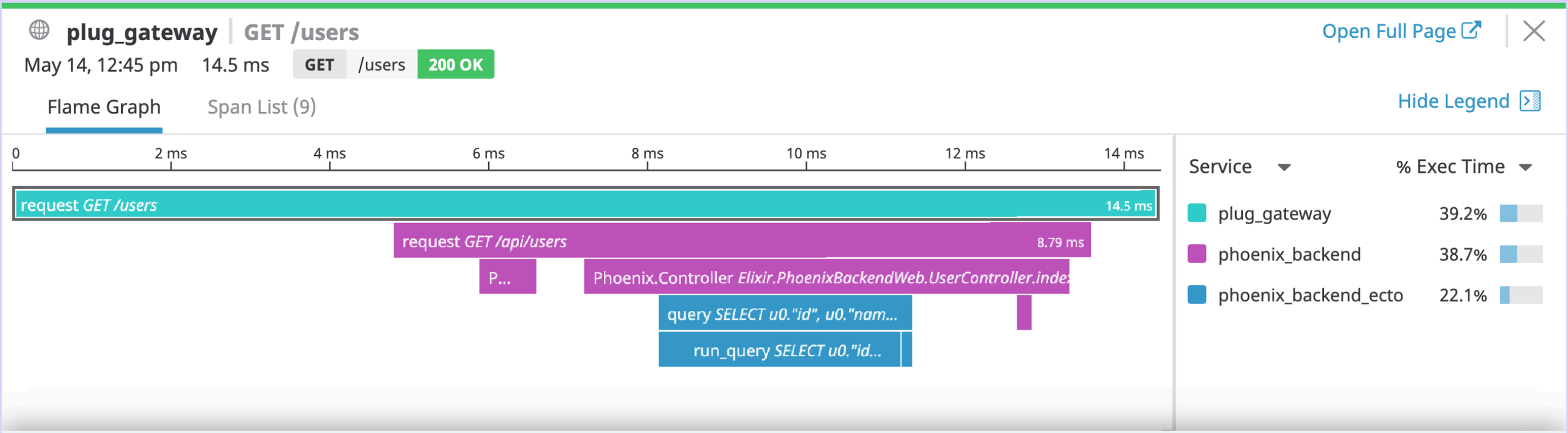
Greg Mefford (@ferggo) #FOSDEM

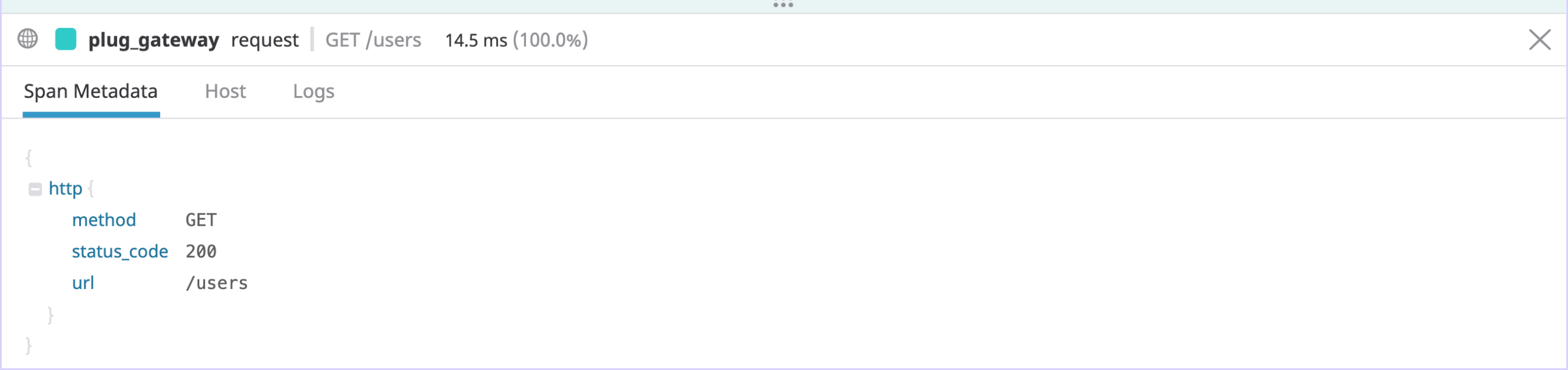
Observability

Super-Powers

Greg Mefford (@ferggo) #FOSDEM

Single-Request Flame Graphs


plug_gateway
GET /users
May 14, 12:45 pm
14.5 ms
GET
/users
200 OK
Open Full Page
Flame Graph
Span List (9)
Hide Legend
0
2 ms
4 ms
6 ms
8 ms
10 ms
12 ms
14 ms
request GET /users
14.5 ms
request GET /api/users
8.79 ms
P...
Phoenix.Controller Elixir.PhoenixBackendWeb.UserController.index
query SELECT u0."id", u0."nam...
run_query SELECT u0."id"...
Service
% Exec Time
plug_gateway 39.2%
phoenix_backend 38.7%
phoenix_backend_ecto 22.1%


plug_gateway request
GET /users
14.5 ms (100.0%)
Span Metadata
Host
Logs
{
http {
method GET
status_code 200
url /users
}
}

Stack-Traces

In Context

Search APM > Services > [redacted]

[redacted] | GET /[redacted]

Apr 23, 5:13 pm 54.6 ms GET /[redacted] 200 OK

Flame Graph Span List (43) Hide Legend

0 5 ms 10 ms 15 ms 20 ms 25 ms 30 ms 35 ms 40 ms 45 ms 50 ms

request *GET* /[redacted] 54.6 ms
Phoenix.Controller [redacted] 54.2 ms
async *Task.async* 26.1 ms
26.0 ms
ExternalRequest.get[redacted] ExternalRequest.post[redacted]
request *GET* /[redacted]
request *POST* /[redacted] 11.0 ms
Phoenix.Con...
Phoenix.Controller [redacted]
Phoenix.View
jiffy_encode
async *Task.async* 43.7 ms
43.7 ms
[redacted]ExternalRequest.get[redacted] 43.6 ms
request *GET* /[redacted] 10.7 ms
request *GET* /[redacted] 18.9 ms
Phoenix.Controller *Eli...*
Phoenix.Controller [redacted] 18.5 ms
jiffy_decode [redacted]
Phoenix.View
Phoenix.View [redacted] 10....

Service % Exec Time

52.5%
37.3%
8.84%
1.27%
0.16%

[redacted] request | GET /[redacted] 54.6 ms (100.0%)

Span Metadata Host Logs Error (1)

```
{
  http {
    method      GET
    status_code 200
    url         /[redacted]
  }
}
```

Greg Mefford (@ferggo) #FOSDEM

Search APM > Services >

GET /

Apr 23, 5:13 pm 54.6 ms GET / 200 OK

Flame Graph Span List (43) Hide Legend

0 5 ms 10 ms 15 ms 20 ms 25 ms 30 ms 35 ms 40 ms 45 ms 50 ms

request *GET /* 54.6 ms
Phoenix.Controller 54.2 ms
async *Task.async* 26.1 ms
26.0 ms
ExternalRequest.get ExternalRequest.post
request *GET /* request *POST /* 11.0 ms
Phoenix.Con... Phoenix.Controller
Phoenix.View
jiffy_encode
async *Task.async* 43.7 ms
43.7 ms
ExternalRequest.get 43.6 ms
request *GET /* 10.7 ms request *GET /* 18.9 ms
Phoenix.Controller *Eli...* Phoenix.Controller 18.5 ms
Phoenix.View Phoenix.View 10....
jiffy_decode

Service % Exec Time

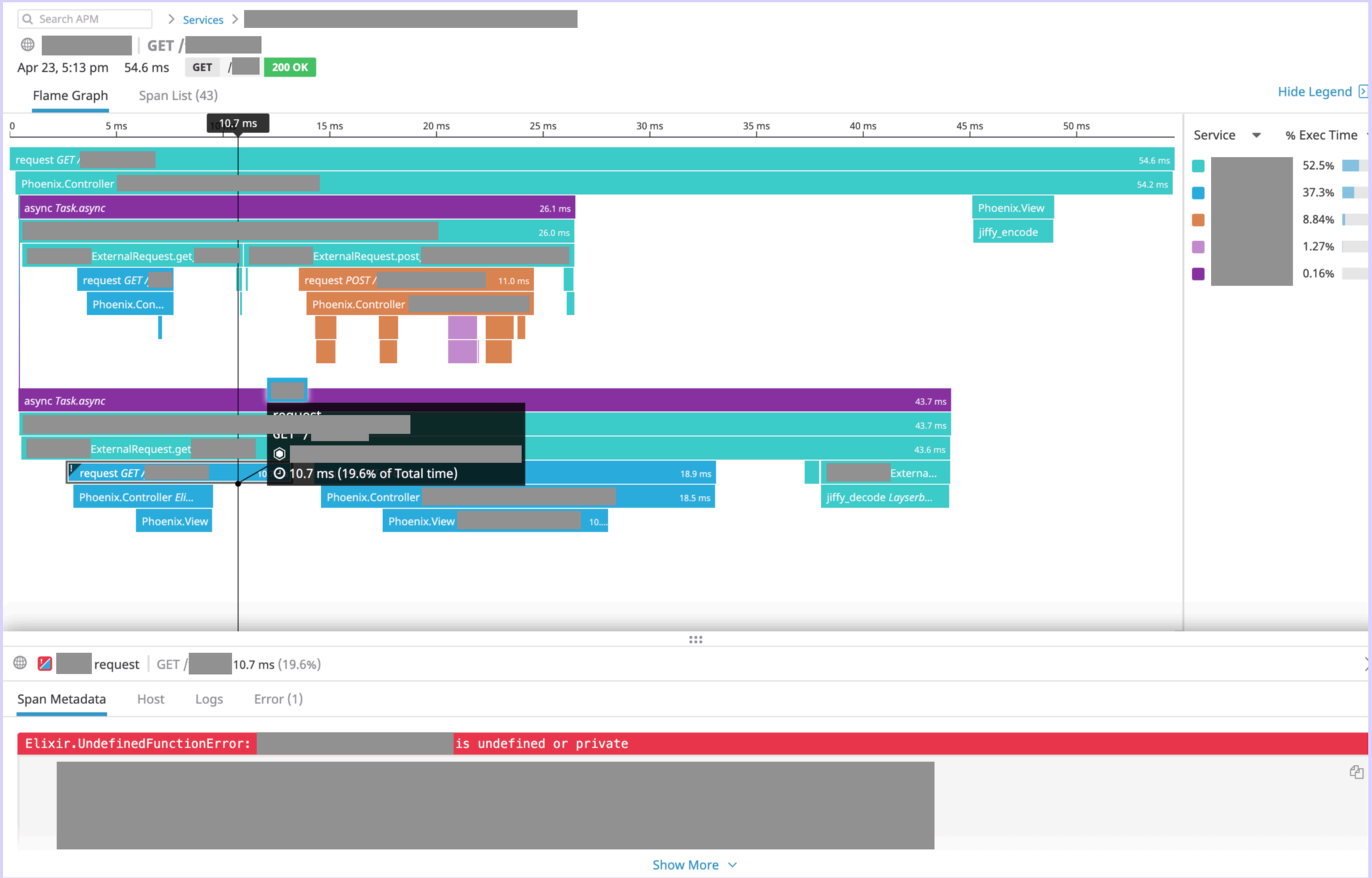
52.5%
37.3%
8.84%
1.27%
0.16%

request | GET / 54.6 ms (100.0%)

Span Metadata Host Logs Error (1)

```
{
  http {
    method      GET
    status_code 200
    url         /
  }
}
```

Greg Mefford (@ferggo) #FOSDEM



Greg Mefford (@ferggo) #FOSDEM

N+1 Queries

Greg Mefford (@ferggo) #FOSDEM

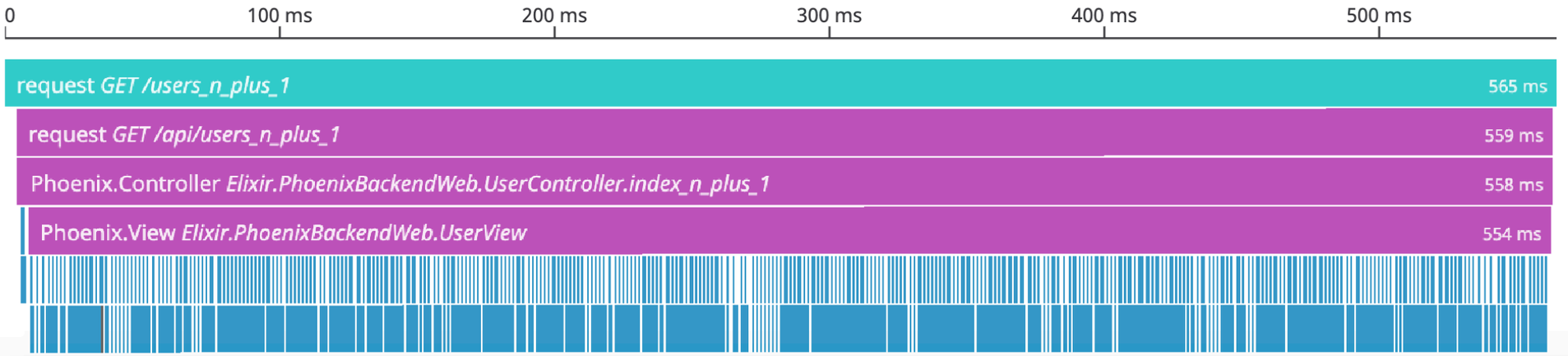
plug_gateway | GET /users_n_plus_1

May 14, 12:42 pm 565 ms GET /users_n_plus_1 200 OK

Open Full Page

Flame Graph Span List (1589)

Hide Legend



Service % Exec Time

phoenix_backend	63.5%
phoenix_backend_ecto	35.5%
plug_gateway	0.96%

phoenix_backend_ecto decode | 15.5 µs (< 0.1%)

Span Metadata Host Logs

```
SELECT p0."title" FROM "posts" AS p0 WHERE (p0."id" = $1)
```


```
{
  param_count  1
  sql {
    db    phoenix_backend_repo
    rows  1
```

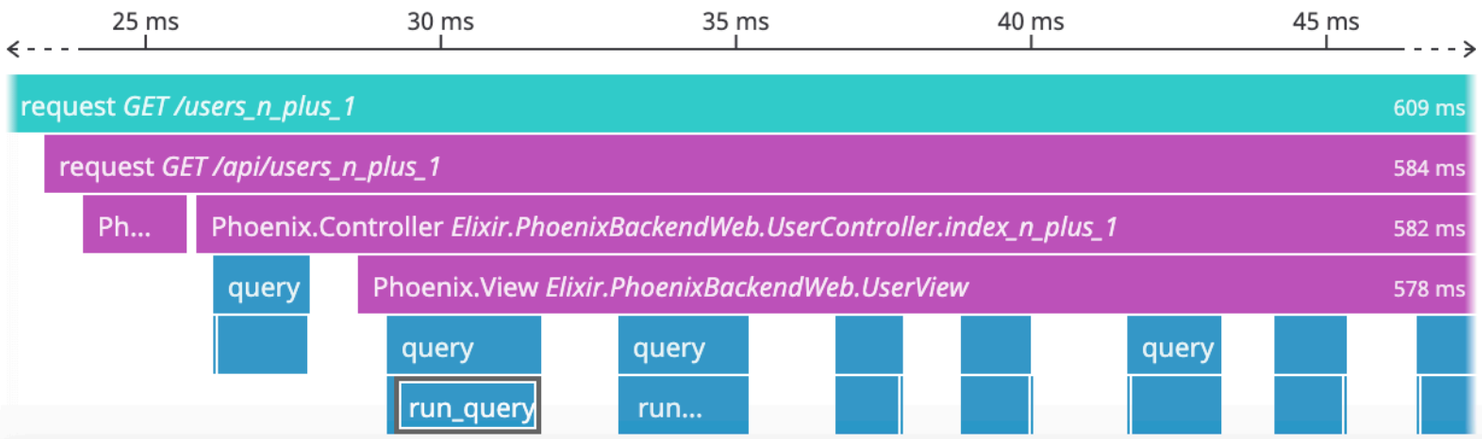

plug_gateway | GET /users_n_plus_1

Open Full Page

May 15, 3:06 pm 609 ms GET /users_n_plus_1 200 OK

Flame Graph Span List (1589)

Hide Legend



Service	% Exec Time
phoenix_backend	63.3%
phoenix_backend_ecto	32.7%
plug_gateway	3.96%

phoenix_backend_ecto run_query | 2.50 ms (0.41%)

Span Metadata Host Logs

```
SELECT p0."id" FROM "posts" AS p0 WHERE (p0."user_id" = $1)
```

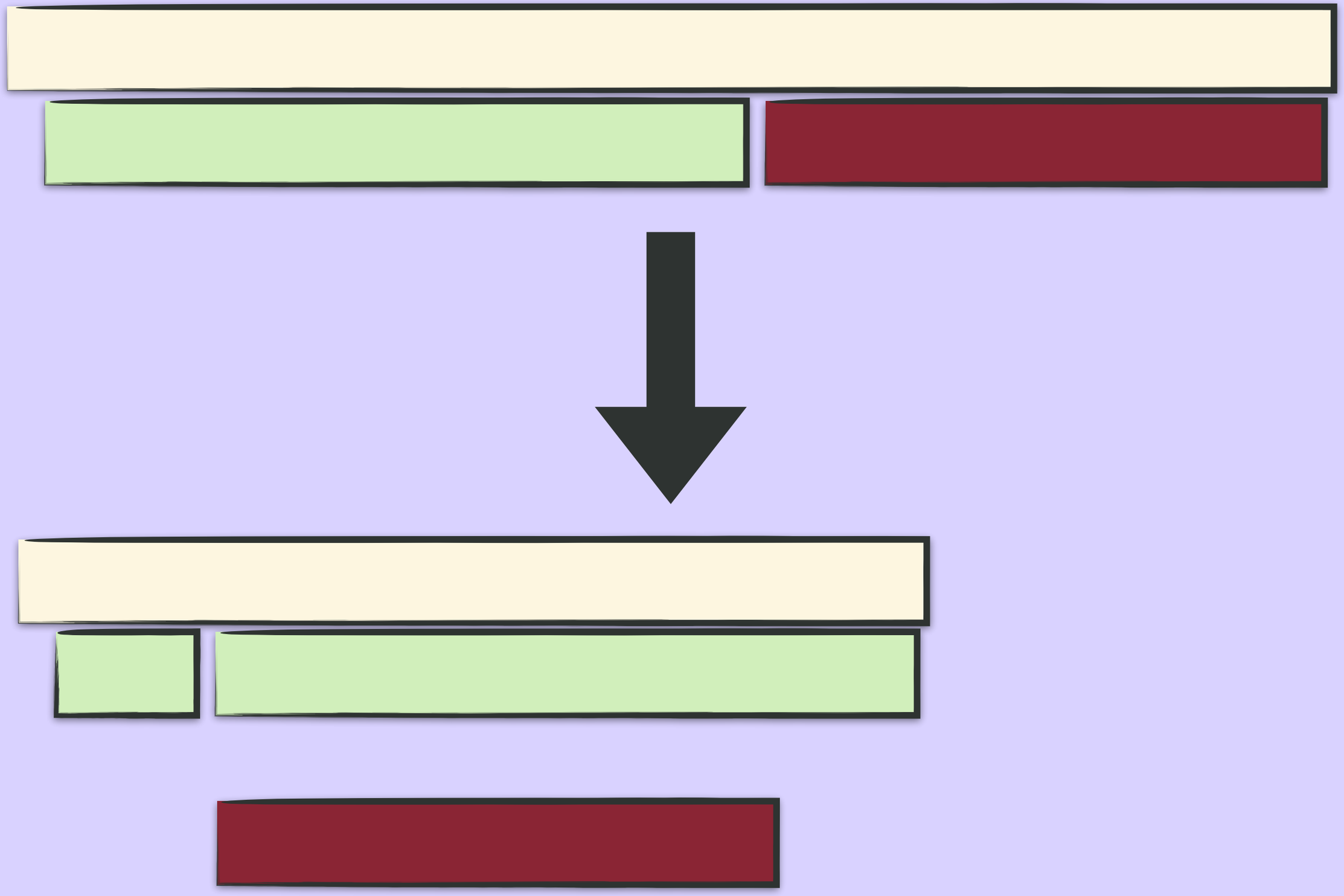
```
{
  param_count  1
  sql {
    db    phoenix_backend_repo
    rows  1
```

Calls to the Same Service


Flame Graph
Span List (206)
0
20 ms
40 ms
60 ms
80 ms
100 ms
120 ms

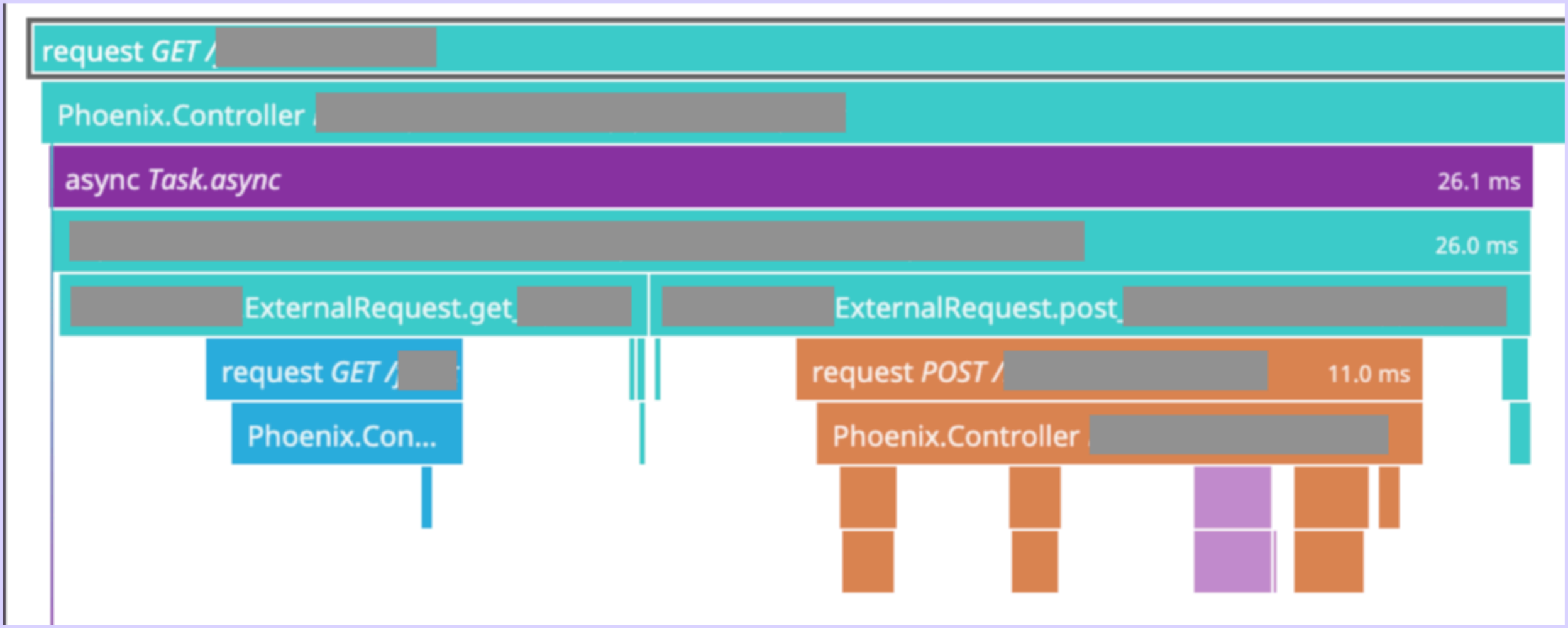
Async

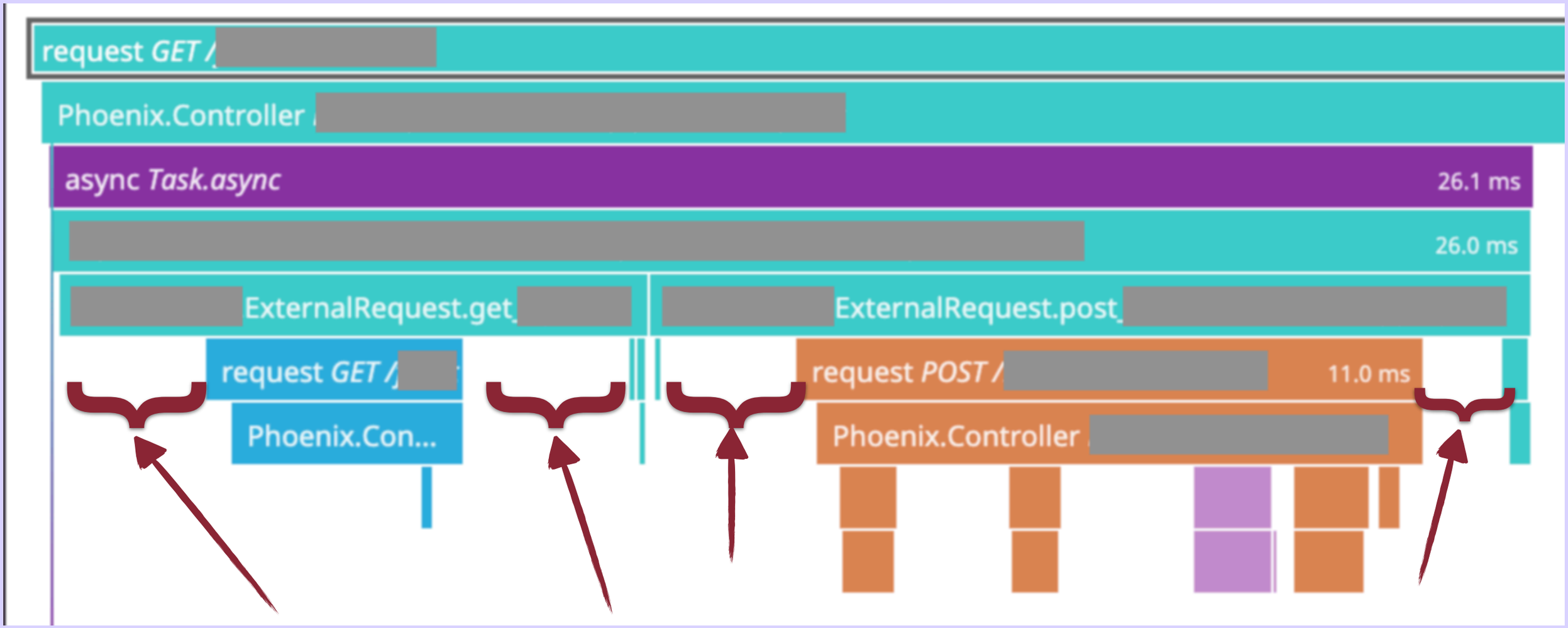
Opportunities



Network

Latency


request GET /
Phoenix.Controller
async Task.async
26.1 ms
26.0 ms
ExternalRequest.get
ExternalRequest.post
request GET /
request POST /
11.0 ms
Phoenix.Con...
Phoenix.Controller


request GET /
Phoenix.Controller
async Task.async
26.1 ms
26.0 ms
ExternalRequest.get
ExternalRequest.post
request GET /
Phoenix.Con...
request POST /
11.0 ms
Phoenix.Controller

Culture Shift

Distributed Tracing Pitfalls

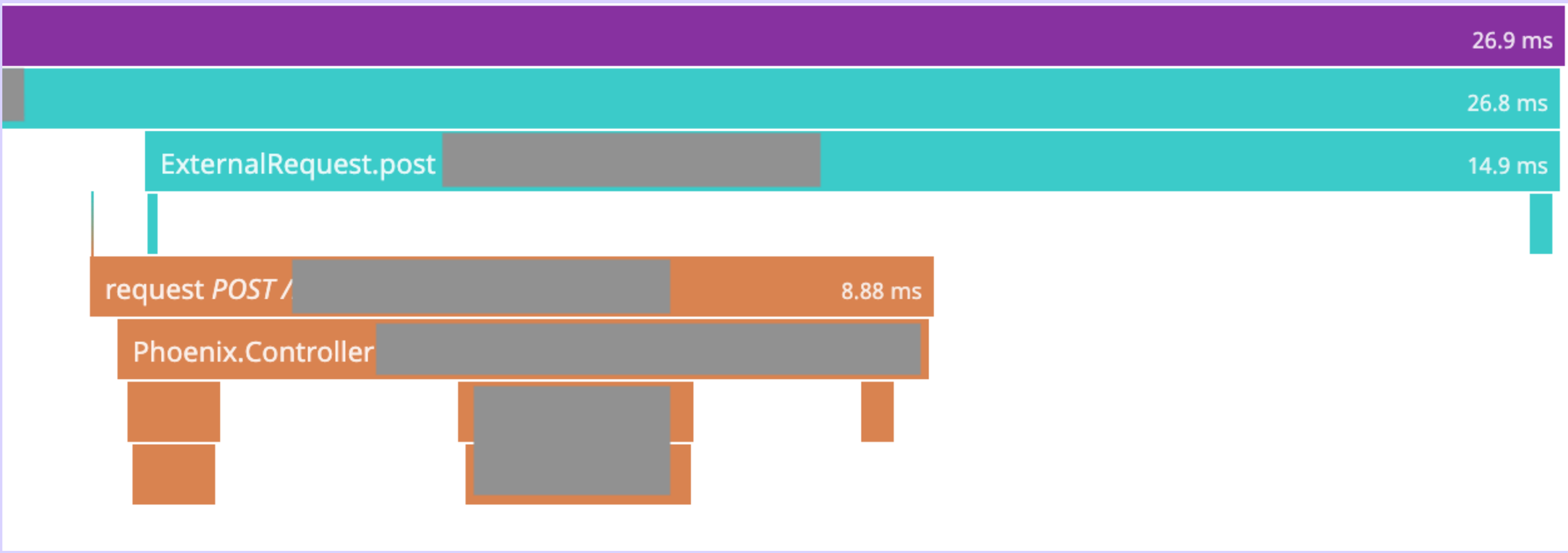
Greg Mefford (@ferggo) #FOSDEM

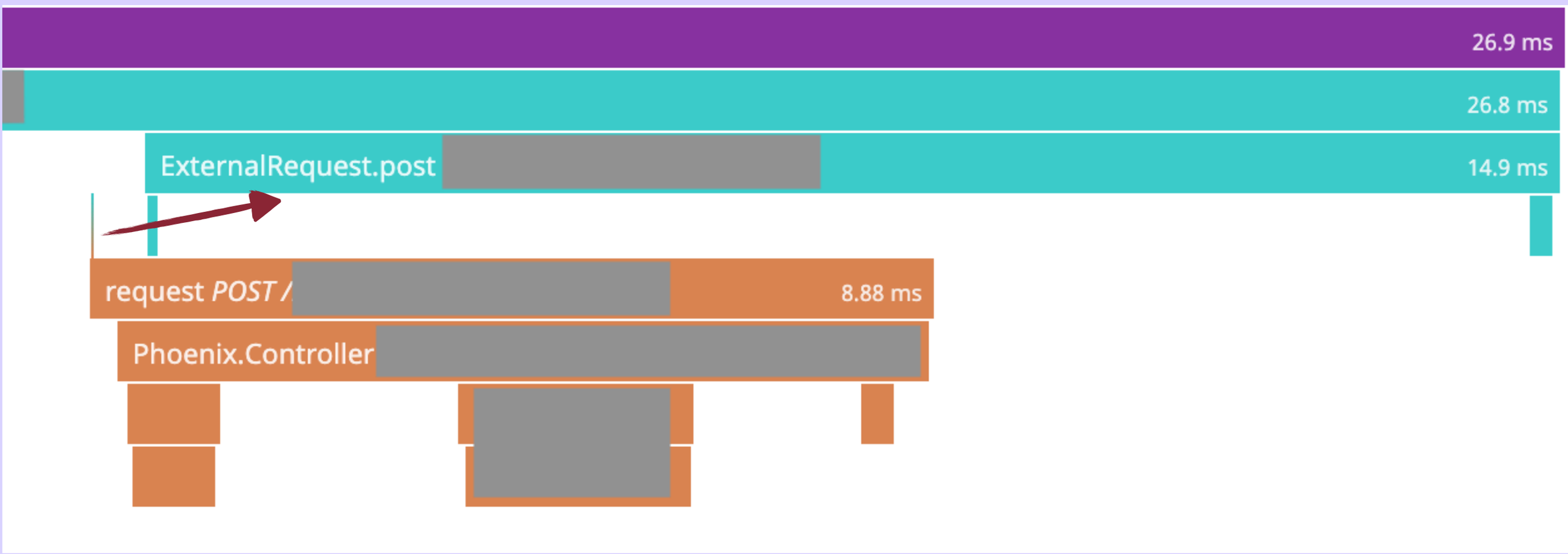
Sampling

Incomplete

Traces

Clock Skew


26.9 ms
26.8 ms
ExternalRequest.post
14.9 ms
request POST /
8.88 ms
Phoenix.Controller


26.9 ms
26.8 ms
ExternalRequest.post
14.9 ms
request POST /
8.88 ms
Phoenix.Controller

OpenTelemetry

(and Open{Census,Tracing})

OpenTracing

Cloud Native Computing Foundation (CNCF)

Observability and Analysis · Tracing

OpenTracing API for Go

Website	https://opentracing.io/
Repository	https://github.com/opentracing/opentracing-go
	★1,691
Crunchbase	https://www.crunchbase.com/organization/cloud-native-computing-found…
LinkedIn	https://www.linkedin.com/company/cloud-native-computing-foundation
Twitter	@opentracing
First Commit	3 years ago
Contributors	37
Headquarters	San Francisco, California

Latest Tweet	3 weeks ago
Latest Commit	about a month
Latest Release	about a month
Headcount	11-50

Cloud Native Incubating

Open Source Software

License Apache License 2.0

No CII Best Practices

Tweet 593

https://opentracing.io

What is OpenTracing?

It is probably easier to start with what OpenTracing is NOT.

- OpenTracing is not a download or a program. Distributed tracing requires that software developers add instrumentation to the code of an application, or to the frameworks used in the application.
- OpenTracing is not a standard. The Cloud Native Computing Foundation (CNCF) is not an official standards body. The OpenTracing API project is working towards creating more standardized APIs and instrumentation for distributed tracing.

OpenTracing is comprised of an API specification, frameworks and libraries that have implemented the specification, and documentation for the project. OpenTracing allows developers to add instrumentation to their application code using APIs that do not lock them into any one particular product or vendor.

https://opentracing.io/docs/overview/what-is-tracing/

What is OpenTracing?

It is probably easier to start with what OpenTracing is NO

- OpenTracing is not a download or a program. Distri … e developers add instrumentation to the code of an application, or to … ation.
- OpenTracing is not a standard. The Cloud Native Computin … official standards body. The OpenTracing API project is wor … rdized APIs and instrumentation for distributed tracing.

OpenTracing is comprised of an API specification, frameworks a… libraries that have implemented the specification, and documentation for the project. OpenTracing allows developers to add instrumentation to their application code using APIs that do not lock them into any one particular product or vendor.

It's not a Download

It's not a Standard

https://opentracing.io/docs/overview/what-is-tracing/

What is OpenTracing?

It is probably ing is NOT.

- OpenTr am. Distributed add instrumentation to the code of an application, or to the fr

- OpenTracing is not a standard. The Cloud Native Compu official standards body. The OpenTracing API project is working towards creating more standardized APIs and instrumentation for distributed tracing.

OpenTracing is comprised of an API specification, frameworks and libraries that have implemented the specification, and documentation for the project. OpenTracing allows developers to add instrumentation to their application code using APIs that do not lock them into any one particular product or vendor.

It's an "API Spec."

It's Various Implementations

https://opentracing.io/docs/overview/what-is-tracing/

What is OpenTracing?

It is probably ing is NOT.

- OpenTr am. Distributed add instrumentation to the code of an application, or to the fr
- OpenTracing is not a standard. The Cloud Native Compu official standards body. The OpenTracing API project is v ed APIs and instrumentation for distributed tracing.

OpenTracing is comprised of an API specification, framework specification, and documentation for the project. OpenTracing to their application code using APIs that do not lock them into ar

https://opentracing.io/docs/over

It's an "API Spec."

It's Various Implementations

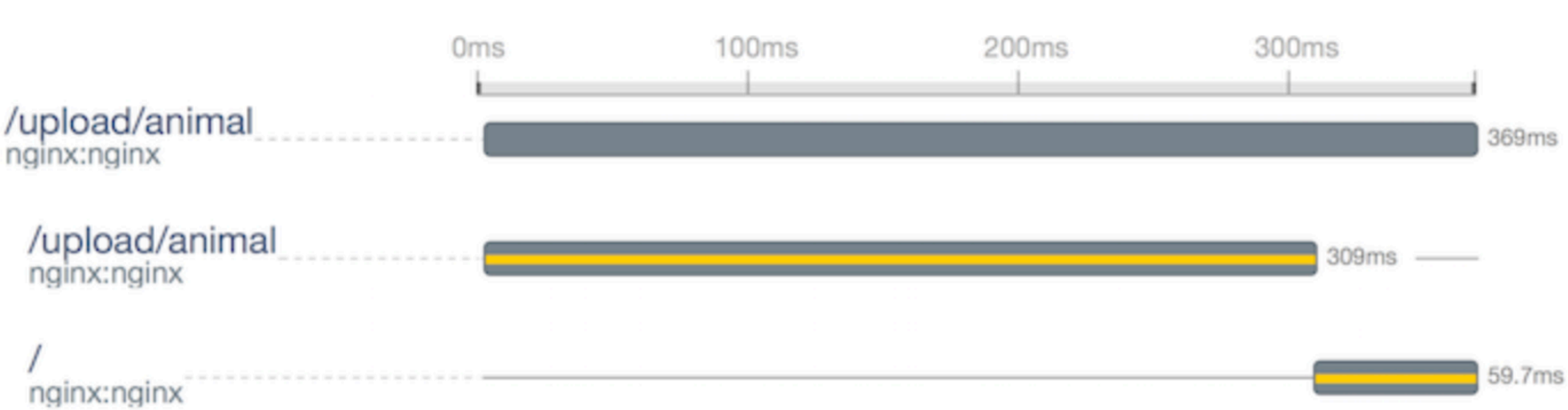
Spandex,
Otter,
ExRay,
(various others)

Enabling OpenTracing for NGINX

We can tell NGINX to trace every request by adding these two lines to `nginx.conf`:

```
http {
  opentracing_load_tracer /path/to/tracer/plugin /path/to/tracer/config;
  opentracing on;
...
```

Now, we'll see the following when admitting a new animal into the zoo:



https://github.com/opentracing-contrib/nginx-opentracing/blob/master/doc/Tutorial.md

OpenCensus

Easily collect telemetry like metrics and distributed traces from your services

What is OpenCensus?

OpenCensus is a single distribution of libraries that collect **metrics** and **distributed traces** from your services

https://opencensus.io

OpenCensus

Eas… …rics an… …ices

Single Set of Libraries

Metrics and Traces

What is OpenCensus?

OpenCensus is a single distribution of libraries that collect **metrics** and **distributed traces** from your services

https://opencensus.io

OpenCensus

Eas… …rics an… …ices

Single Set of Libraries

Metrics and Traces

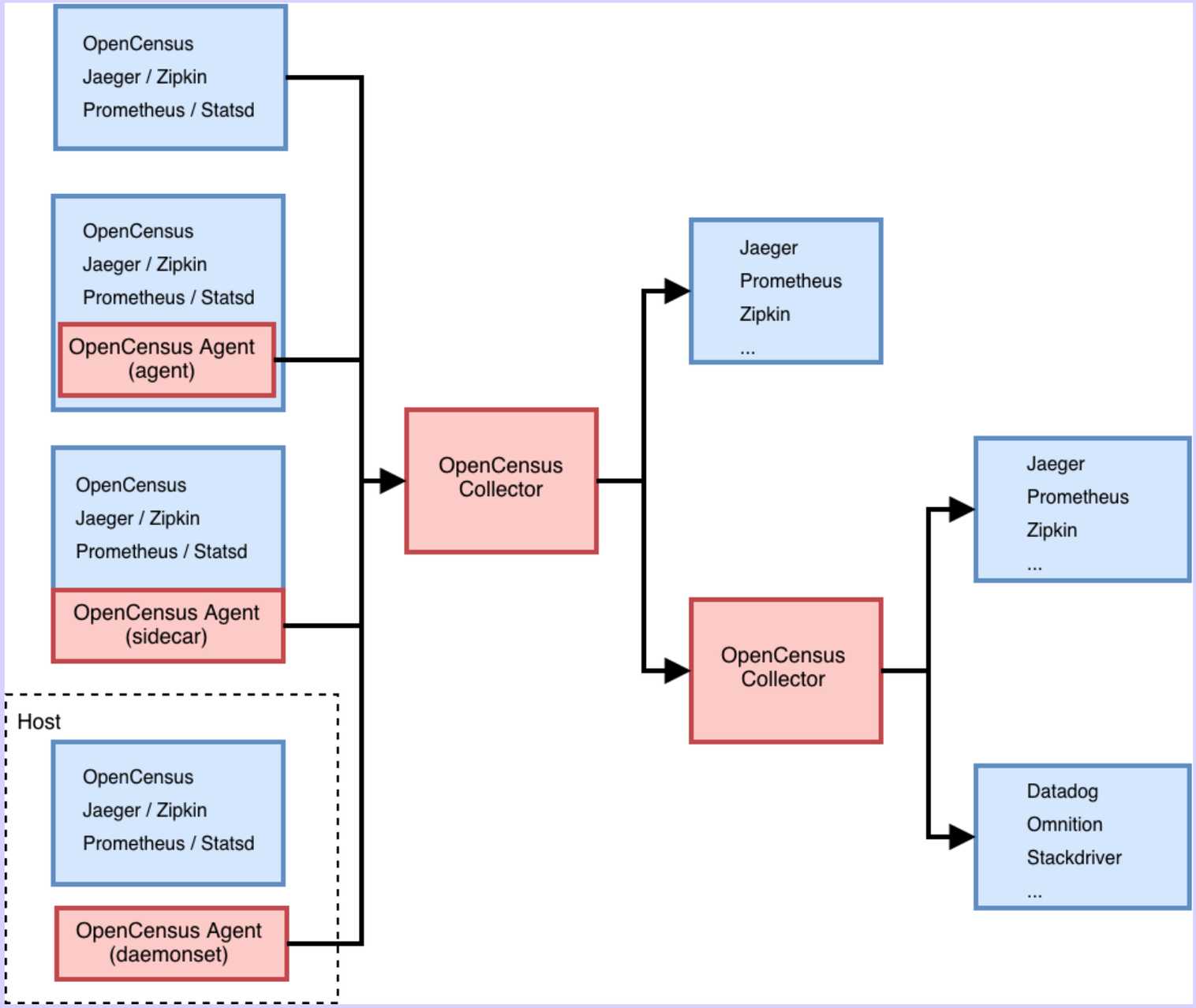
… and logs in the future

What is OpenCensus?

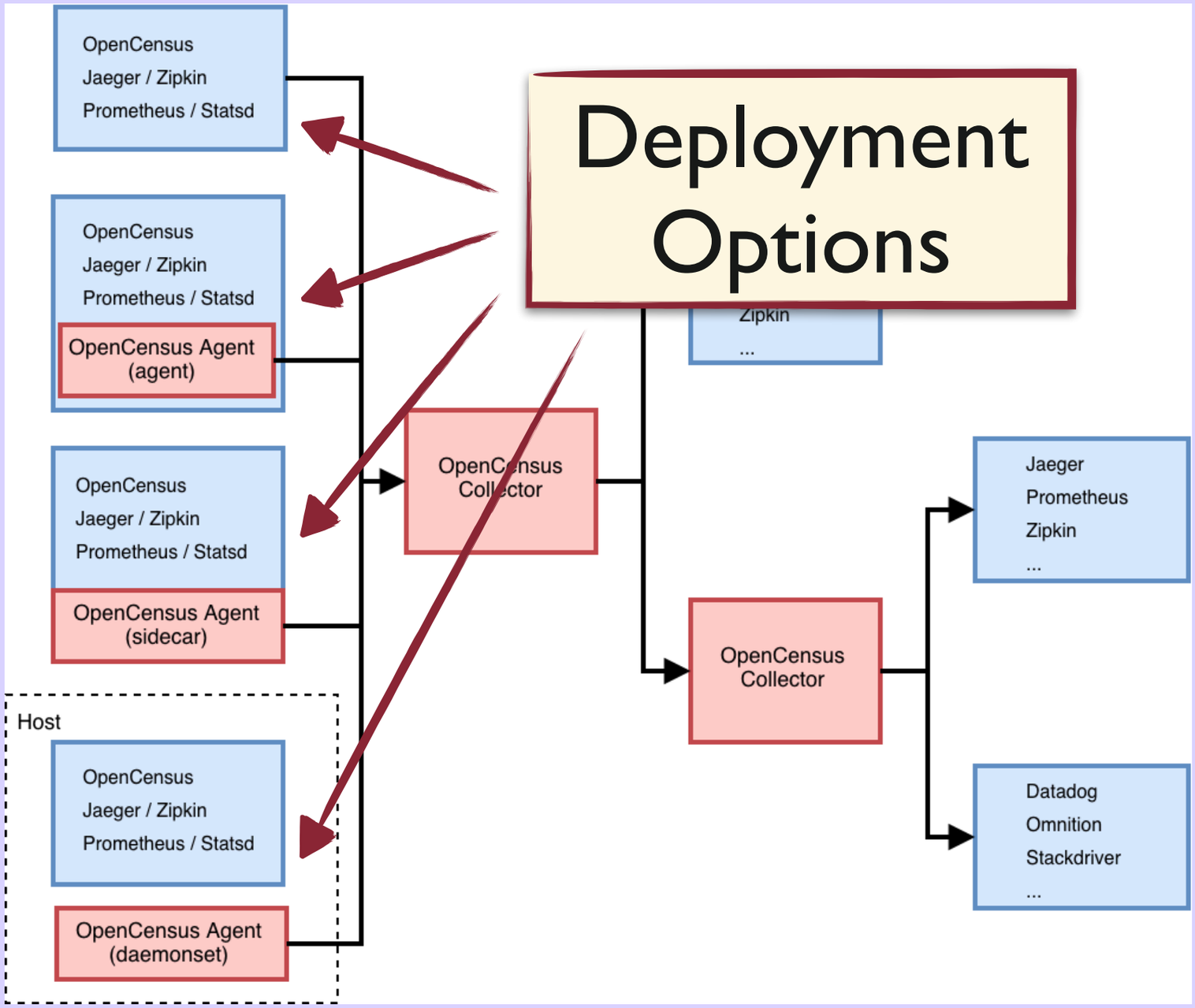
OpenCensus is a single distribution of libraries tha…

distributed traces from your services

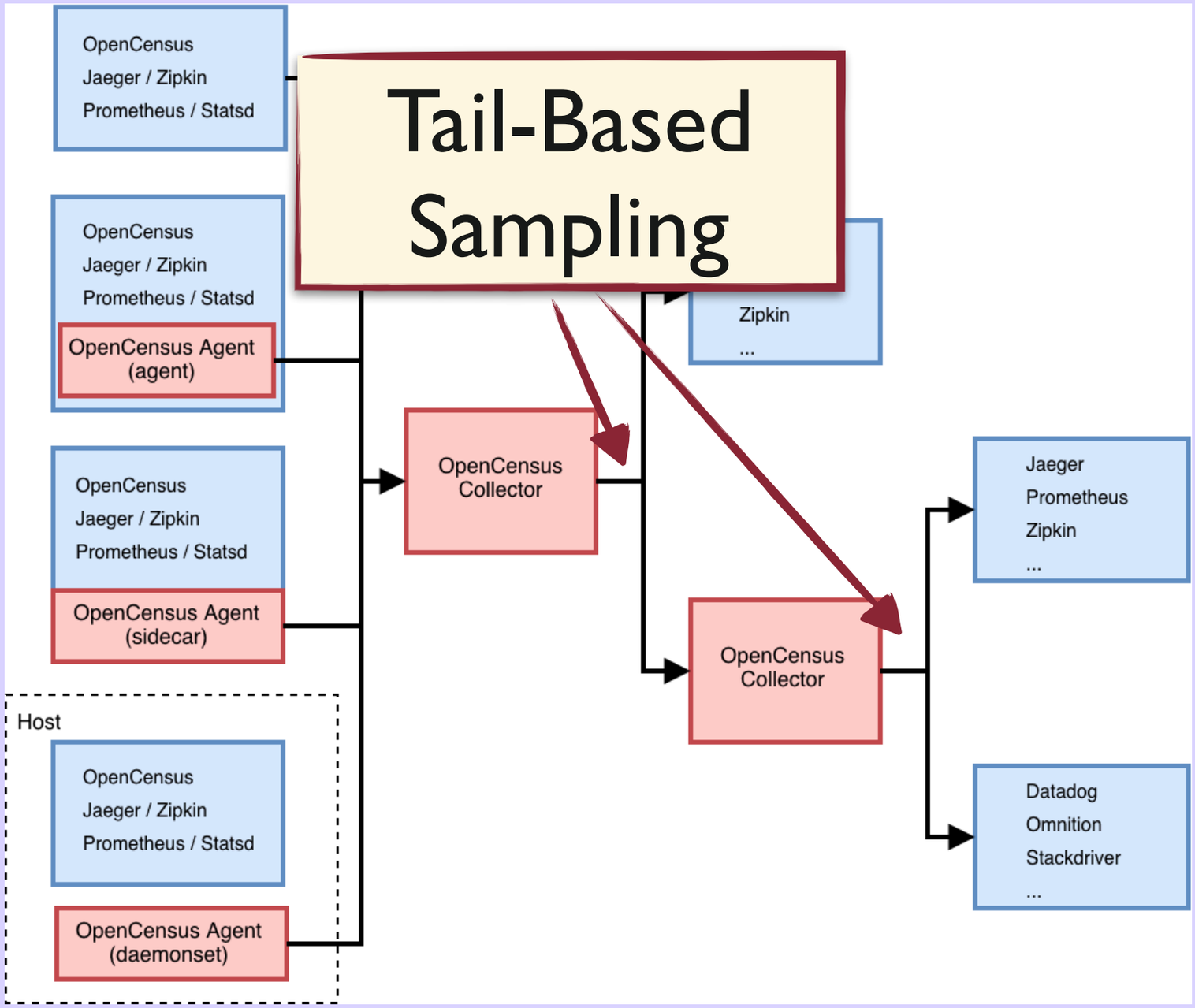
https://opencensus.io



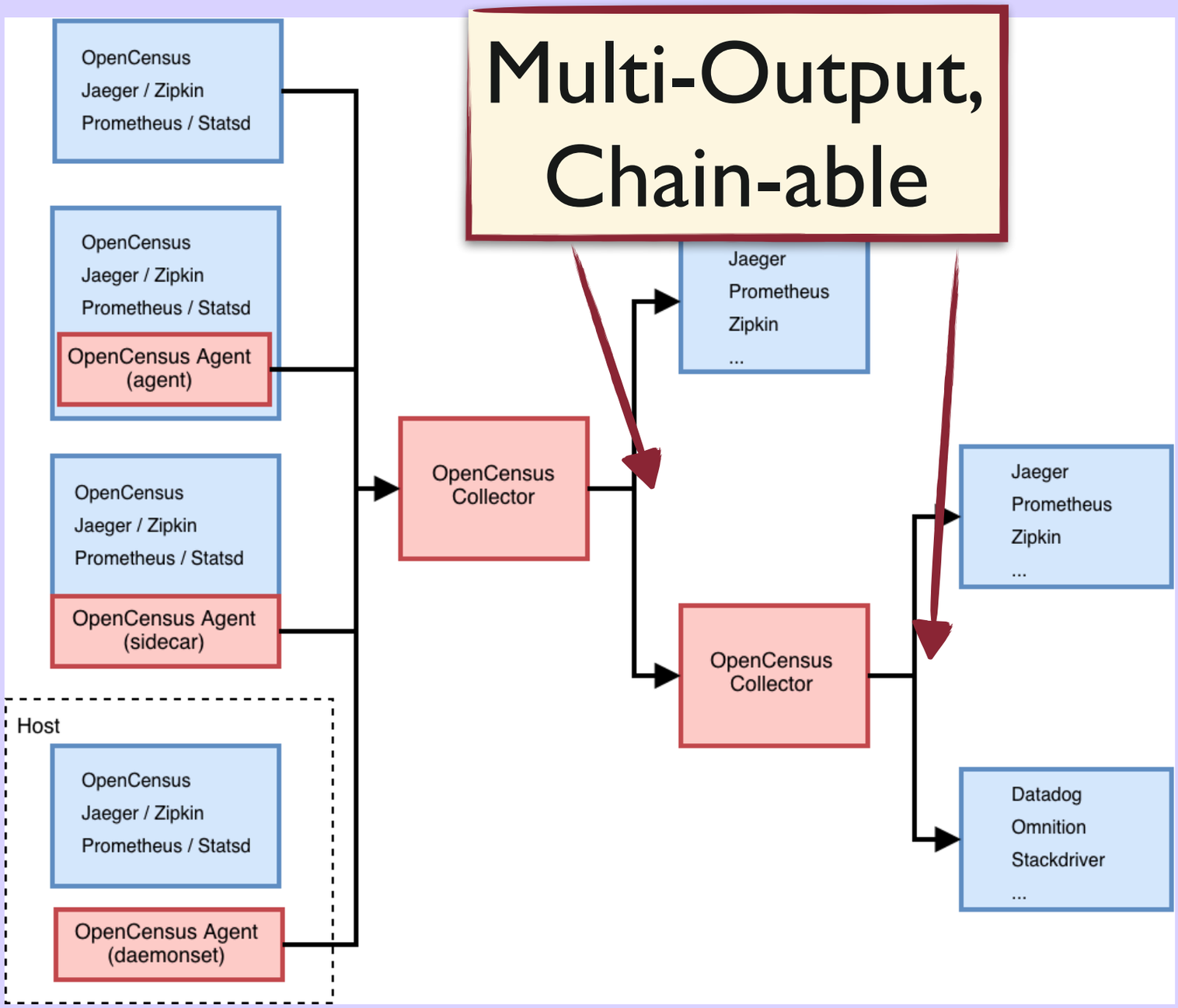
https://opencensus.io/service/



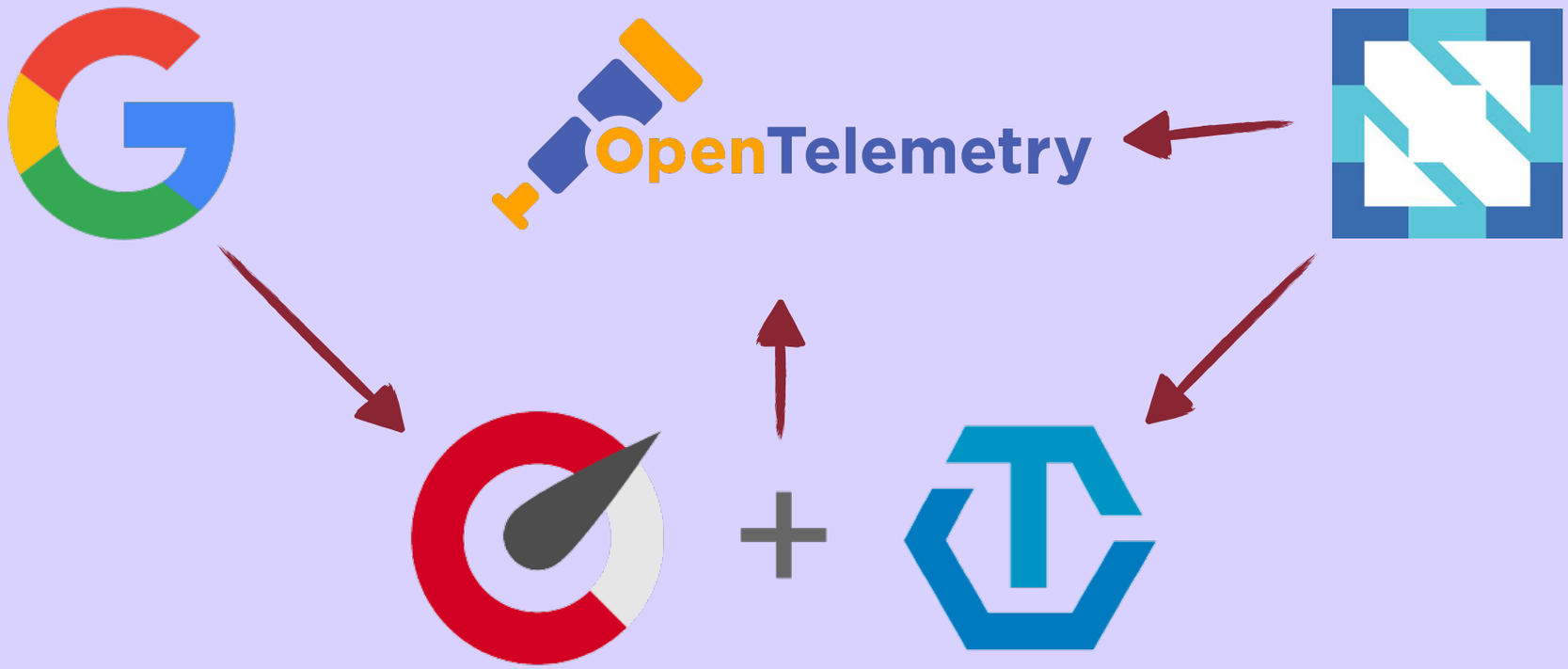
https://opencensus.io/service/



https://opencensus.io/service/



https://opencensus.io/service/



https://medium.com/opentracing/a-roadmap-to-convergence-b074e5815289

Effective observability requires high-quality telemetry

OpenTelemetry makes robust, portable telemetry a built-in feature of cloud-native software.

OpenTelemetry provides a single set of APIs, libraries, agents, and collector services to capture distributed traces and metrics from your application. You can analyze them using Prometheus, Jaeger, and other observability tools.

https://opentelemetry.io/

Effective observability requires high-qu

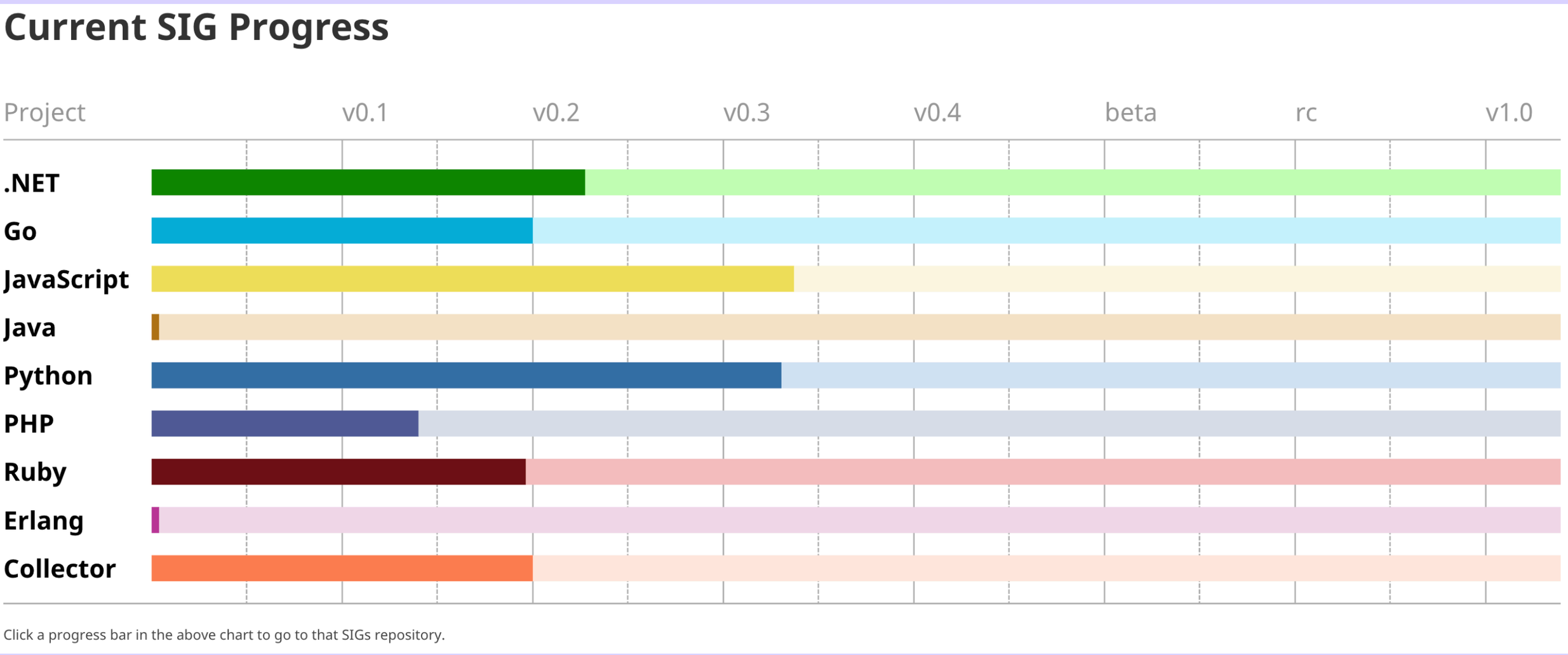
OpenTele portable telemetry ou software.

OpenTelemetry provides a single set of APIs, libraries, agents, and collector services to capture distributed traces and metrics from your application. You can analyze them using Prometheus, Jaeger, and other observability tools.

Single Set of Libraries and Tools

Metrics, Traces
(logs in the future)

https://opentelemetry.io/



https://opentelemetry.io/project-status/

Spandex

A modular/adapter based tracing ecosystem for elixir.

Repositories 5 | People 2 | Teams 1 | Projects 0

Find a repository... | Type: All | Language: All | New

spandex_phoenix

Phoenix Instrumentation tracer

Elixir ★ 12 7 MIT Updated 10 days ago

spandex_datadog

A datadog adapter for the `spandex` library.

Elixir ★ 5 7 MIT Updated 10 days ago

spandex_ecto

Tools for integrating Ecto with Spandex

Elixir ★ 2 5 MIT Updated 24 days ago

Top languages

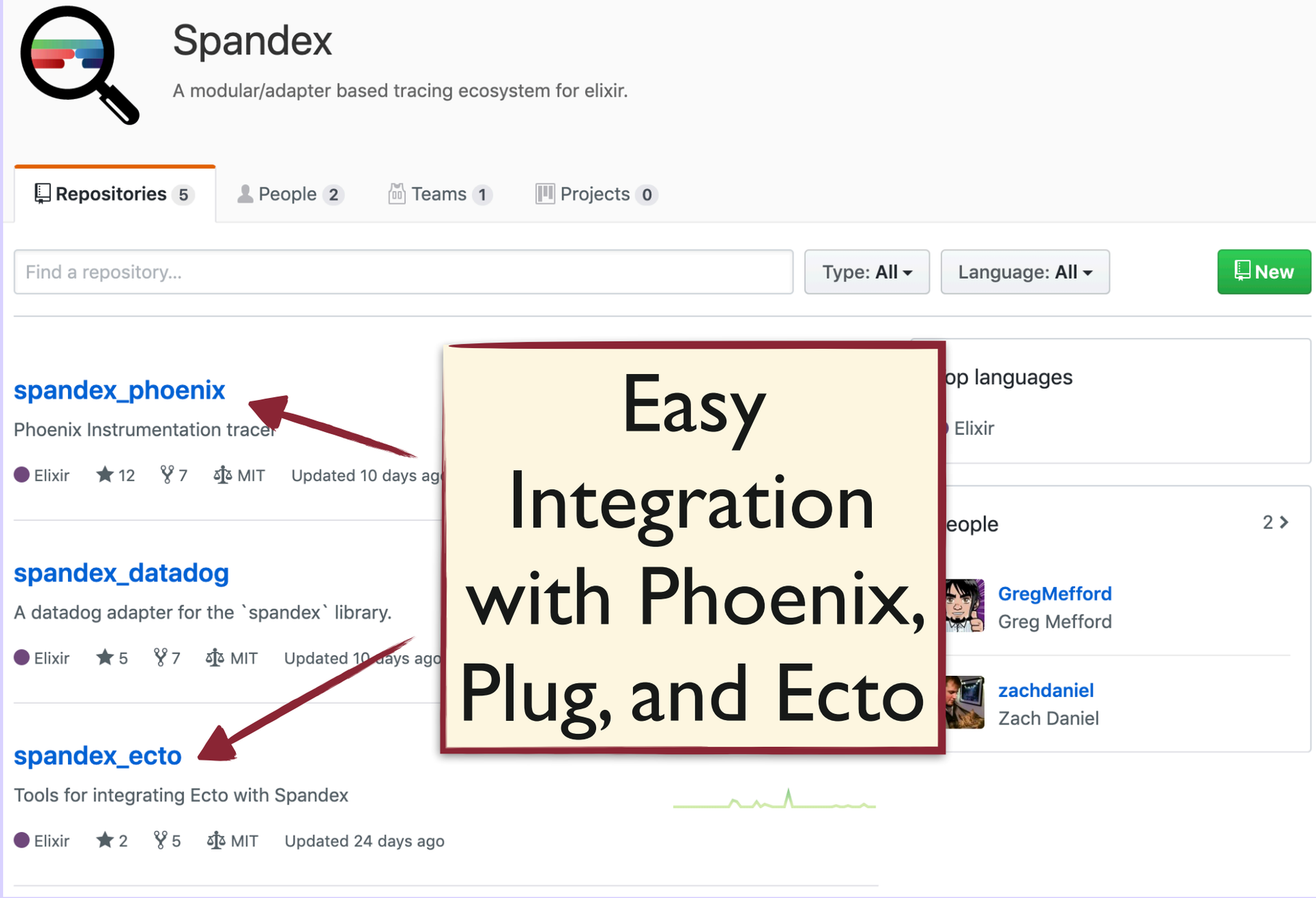
Elixir

People 2

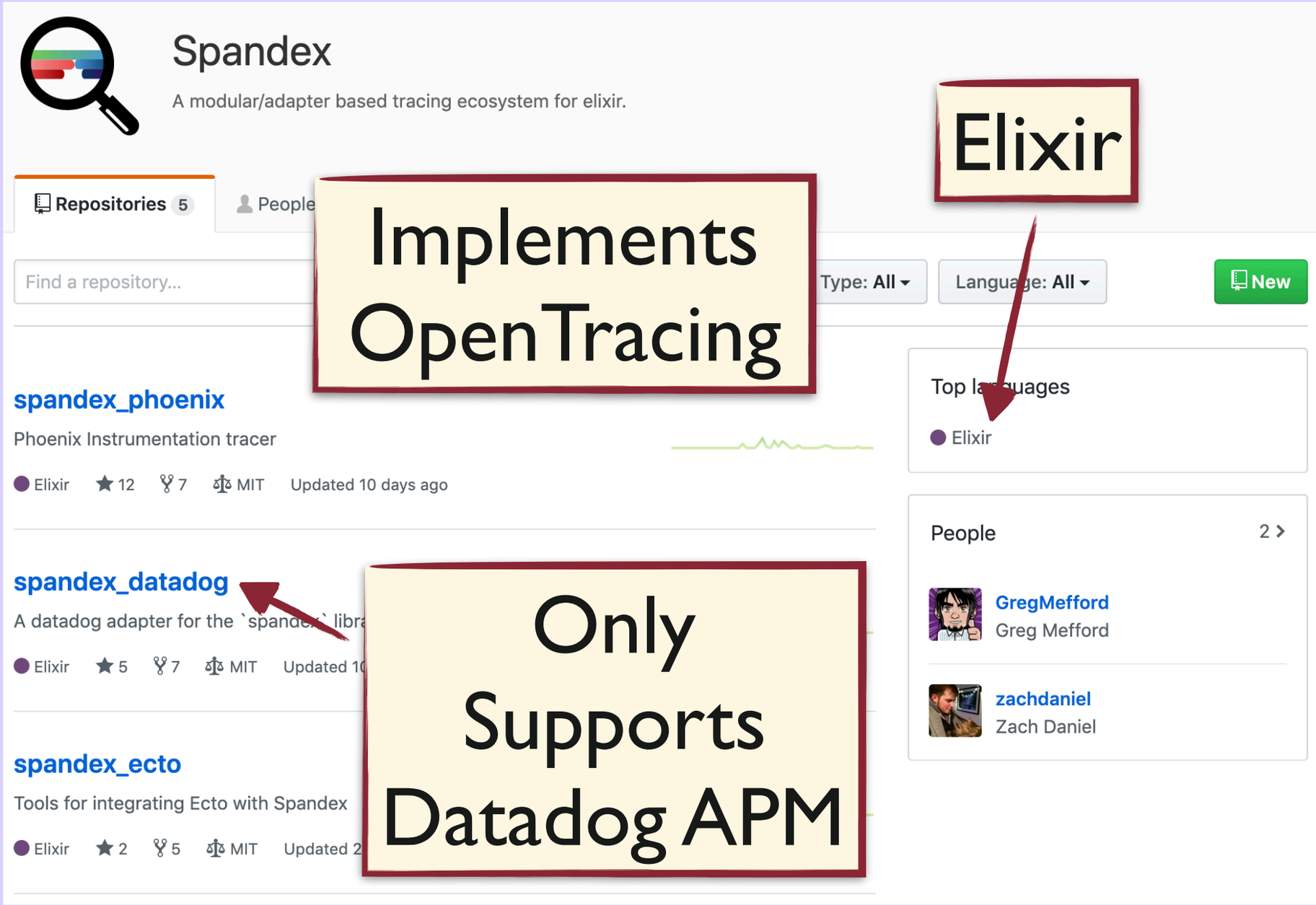
GregMefford
Greg Mefford

zachdaniel
Zach Daniel

https://github.com/spandex-project/



https://github.com/spandex-project/



https://github.com/spandex-project/

Opencensus.io integrations for Erlang, Elixir, and other BEAM languages

Report abuse

Repositories 20 | People 3 | Projects 2

opencensus | Type: **All** | Language: **All**

18 results for repositories matching **opencensus**

Clear filter

Top languages

Erlang Elixir

https://github.com/opencensus-beam/

Opencensus.io integrations for Erlang, Elixir, and other BEAM languages

Report abuse

Repositories 20 | People 3

opencensus | Type: All | Language: All

18 results for repositories matching opencensu

Top languages

Erlang Elixir

Implements OpenCensus

Erlang (and Elixir)

Supports Various Trace Collectors (including Datadog APM)

https://github.com/opencensus-beam/

Opencensus.io integrations for Erlang, Elixir, and other BEAM languages

Repositories 20 | **People 3** | Projects 2

Find a member...

Ilya Khaprov
deadtrickster

Łukasz Jan Niemier
hauleth

Tristan Sloughter
tsloughter

++

Spandex

Repositories 5 | **People 2**

Find a member...

2 people in the Spandex organization

Greg Mefford
GregMefford

Zach Daniel
zachdaniel

opentelemetry-beam

Repositories 3 Packages

Find a member...

6 people in the opentelemetry-beam


Ilya Khaprov
deadtrickster


Fred Hebert
ferd


Greg Mefford
GregMefford


Łukasz Jan Niemier
hauleth


Irving Reid
irvingreid


Tristan Sloughter
tsloughter

open-telemetry / **opentelemetry-erlang-api**

Watch 9 Star 5 Fork 4

Code Issues 0 Pull requests 3 Actions Projects 0 Wiki Security Insights

Erlang/Elixir OpenTelemetry API

20 commits **2** branches **0** packages **1** release **2** contributors Apache-2.0

open-telemetry / **opentelemetry-erlang**

Watch 15 Star 51 Fork 7

Code Issues 5 Pull requests 2 Actions Projects 0 Security Insights

OpenTelemetry Erlang SDK

24 commits **3** branches **0** packages **1** release **4** contributors Apache-2.0

Telemetry

(Metrics and Events)

Telemetry

https://github.com/beam-telemetry/telemetry

Simple

Standard

Safe*

* ¯_(ツ)_/¯

```erlang
% Erlang

% In a library
telemetry:execute([web, request, done],
                  #{latency => Latency}, #{status_code => Status})

% In a receiver
telemetry:attach(<<"log-response-handler">>,
                 [web, request, done],
                 fun log_response_handler:handle_event/4, [])
```

```elixir
# Elixir

# In a library
:telemetry.execute([:web, :request, :done],
  %{latency: latency},%{status_code: status})

# In a receiver
:telemetry.attach("log-response-handler",
  [:web, :request, :done], &LogResponseHandler.handle_event/4, nil)
```

```erlang
% Erlang

-module(log_response_handler).
-include_lib("kernel/include/logger.hrl")

handle_event([web, request, done],
             #{latency := Latency}, #{status_code := Status}, _Config) ->
  ?LOG_INFO("~p sent in ~p", [Status, Latency]).
```

```elixir
# Elixir

defmodule LogResponseHandler do
  require Logger

  def handle_event([:web, :request, :done], measurements, metadata, _config) do
    Logger.info("#{metadata.status_code} sent in #{measurements.latency}")
  end
end
```

```erlang
% Erlang

-module(log_response_handler).
-include_lib("kernel/include/logger.hrl")

handle_event([web, request, done],
             #{latency := Latency}, #{status_code := Status}, _Config) ->
  ?LOG_INFO("~p sent in ~p", [Status, Latency]).
```

Called Synchronously

```elixir
# Elixir

defmodule LogResponseHandler do
  require Logger

  def handle_event([:web, :request, :done], measurements, metadata, _config) do
    Logger.info("#{metadata.status_code} sent in #{measurements.latency}")
  end
end
```

```erlang
% Erlang

-module(log_response_handler).
-include_lib("kernel/include/logger.hrl").

handle_event([web, request, done],
             #{latency := Latency}, #{status_code := Status}, _Config) ->
  ?LOG_INFO("~p sent in ~p", [Status, Latency]).
```

start / stop Events Used as Tracing Hooks

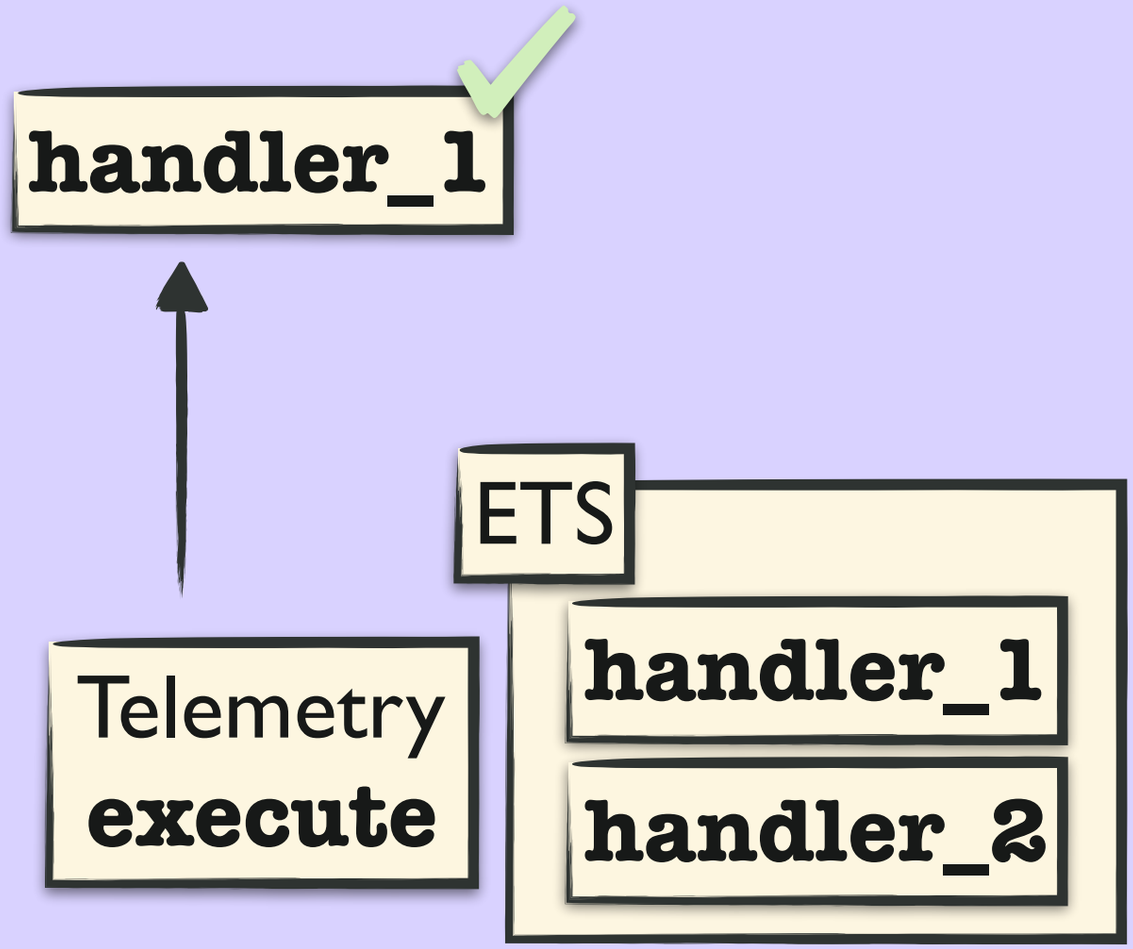
```elixir
# Elixir

defmodule LogResponseHandler do
  require Logger

  def handle_event([:web, :request, :done], measurements, metadata, _config) do
    Logger.info("#{metadata.status_code} sent in #{measurements.latency}")
  end
end
```

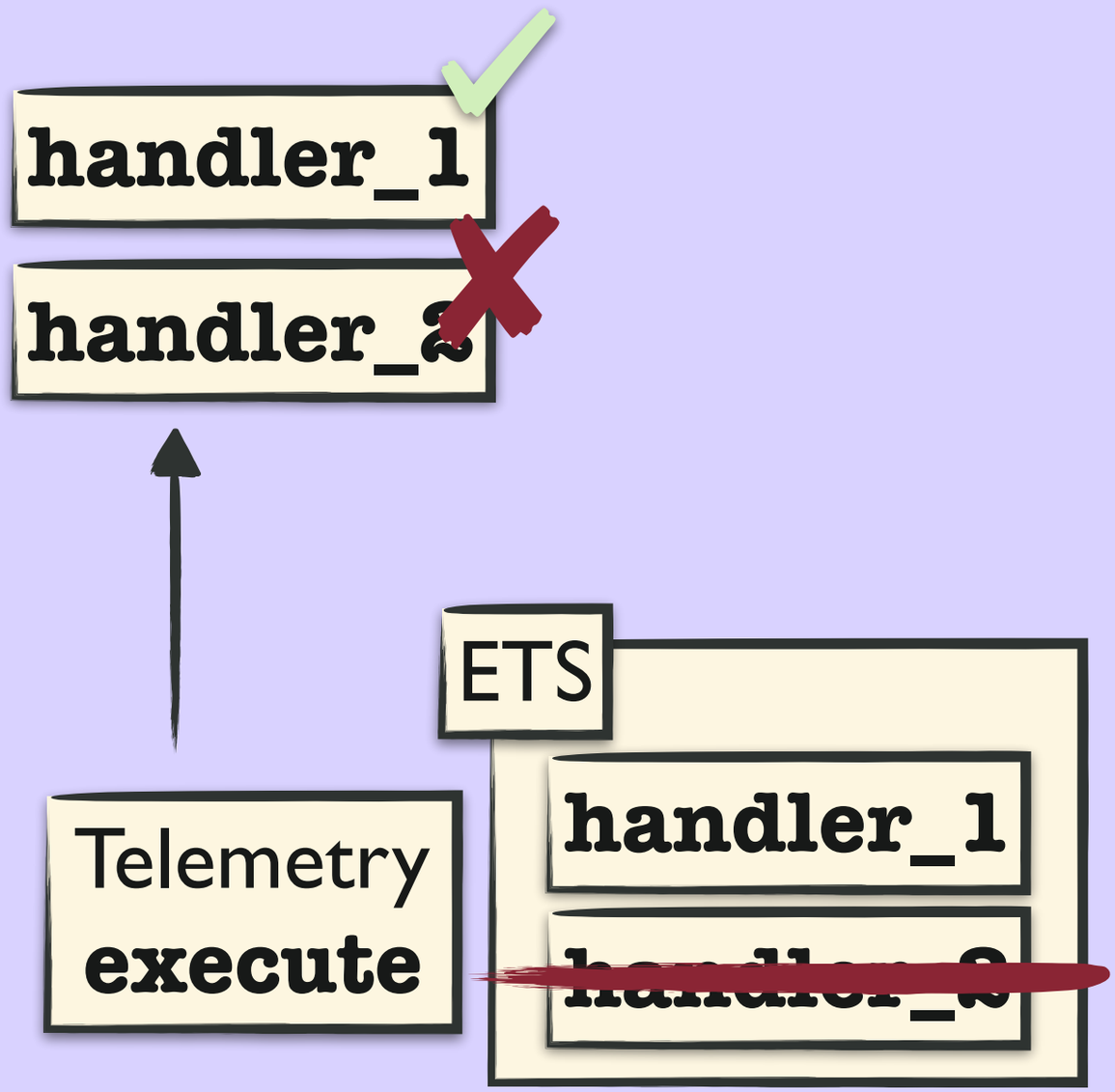
```erlang
% Erlang

-module(log_response_handler).
-include_lib("kernel/include/logger.hrl")

handle_event([web, request, done],
             #{latency := Latency}, #{status_code := Status}, _Config) ->
  ?LOG_INFO("~p sent in ~p", [Status, Latency]).
```

```elixir
# Elixir

defmodule LogResponseHandler do
  require Logger

  def handle_event([:web, :request, :done], measurements, metadata, _config) do
    Logger.info("#{metadata.status_code} sent in #{measurements.latency}")
  end
end
```

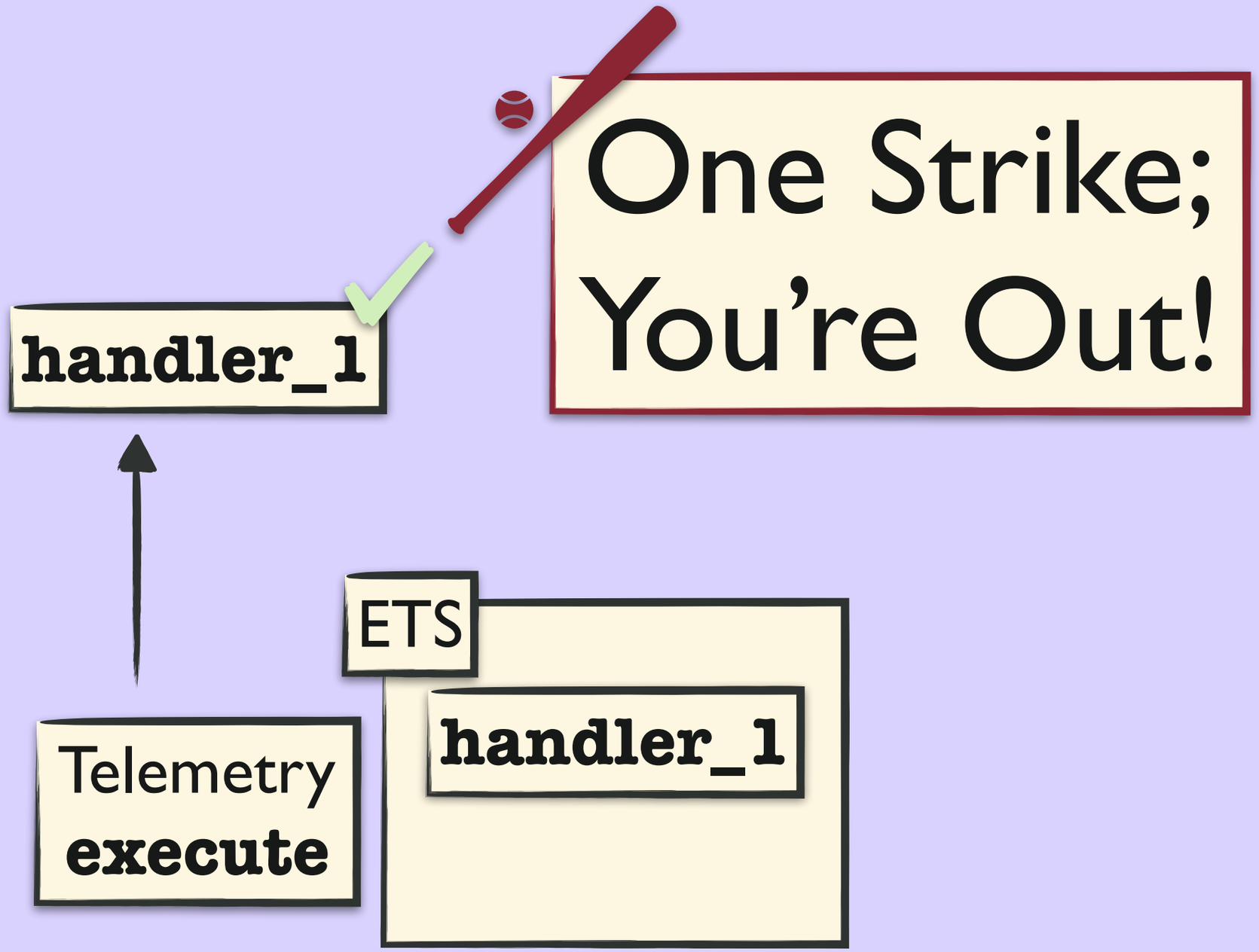
Easy Time-Series Metrics

Simple!


handler_1
Telemetry
execute
ETS
handler_1
handler_2


handler_1
handler_2
Telemetry
execute
ETS
handler_1
handler_2

One Strike; You're Out!



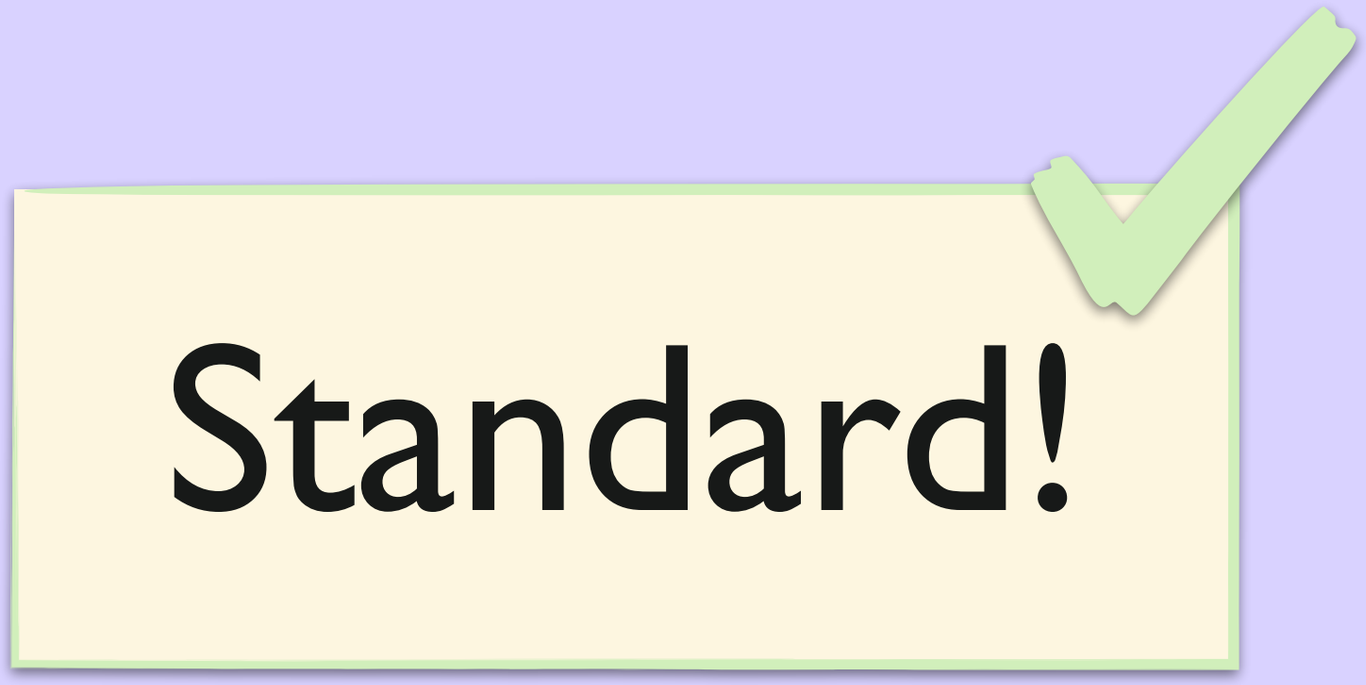
* ¯_(ツ)_/¯

Plug ✓

Phoenix ✓

EctoSQL ✓

Your Library


Standard!

Take-Aways

Instrument your App

Instrument your Libs

Use Telemetry

Learn about OpenTelemetry