

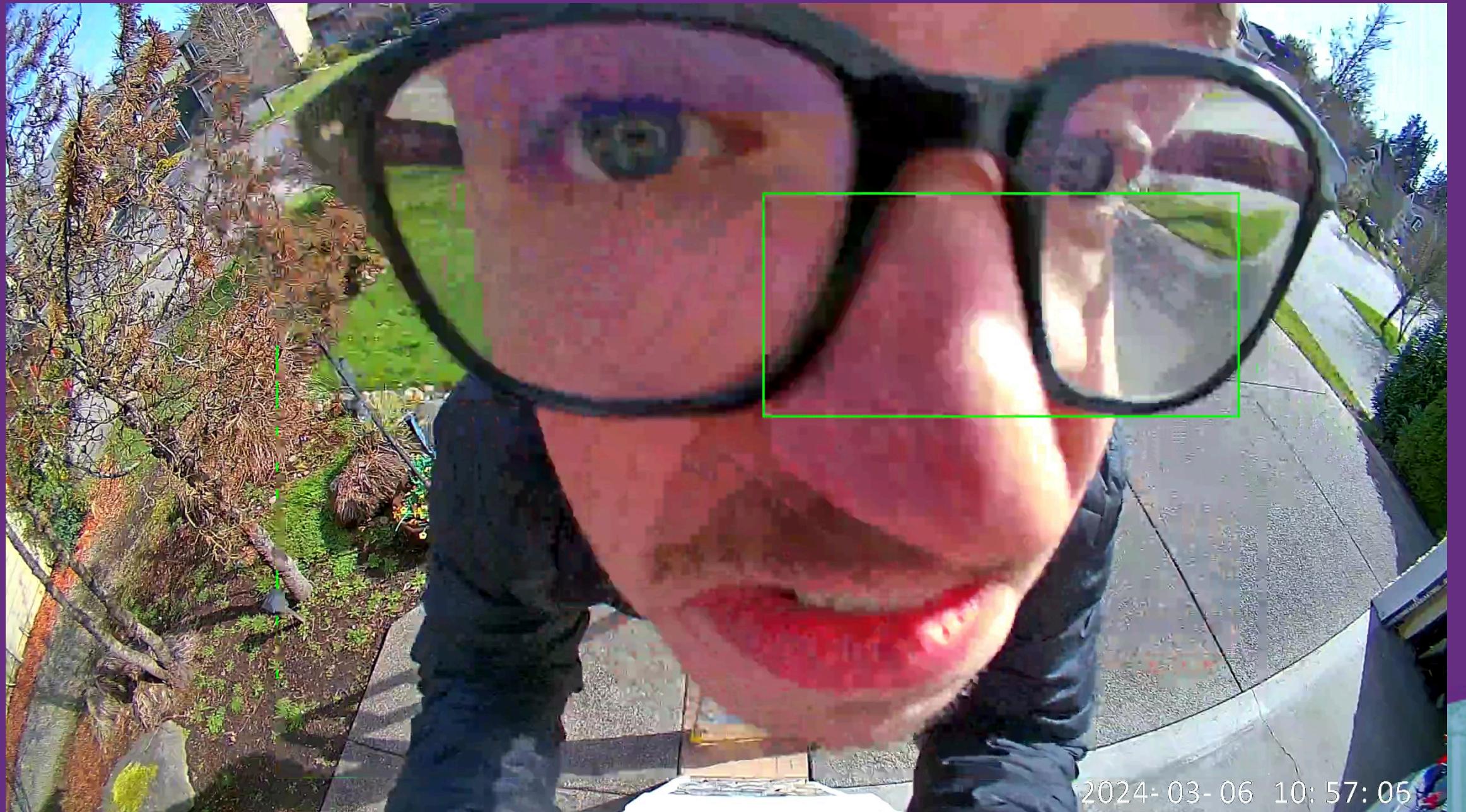


Telemetry- Driven Development

“The Purpose Of the System Is What It Does”

Noah Betzen (@SmartRent)

```
iex(1)> Node.list(:this)  
[:"noah@nezteb.net"]
```



"I have approximate knowledge of many things" - Demon Cat
(Adventure Time)

SPONSORS

Thanks to our amazing sponsors who are helping make the forum and our community the best it can be.

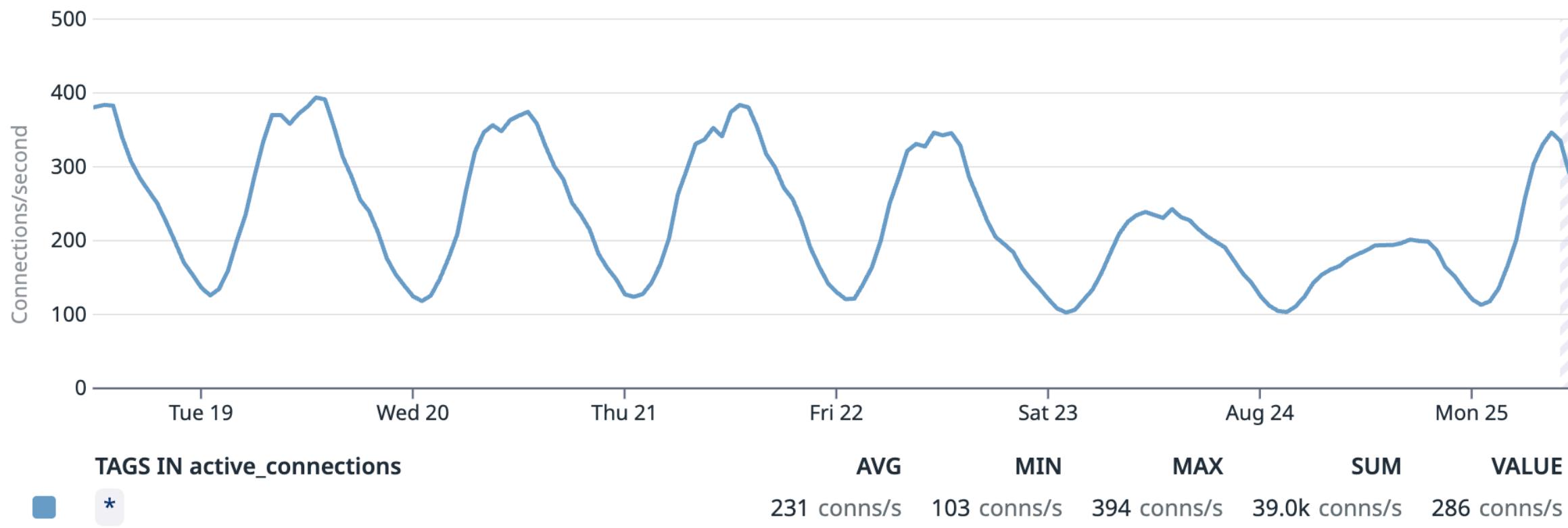
D I A M O N D



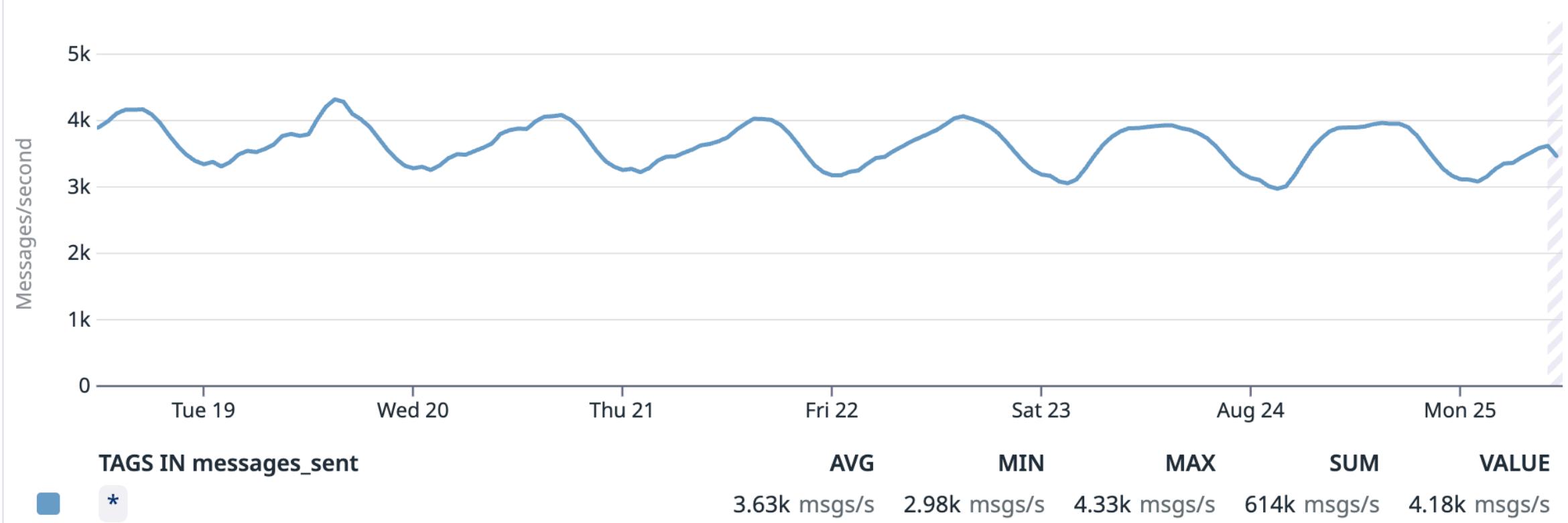
SmartRent (SMRT) is an enterprise smart building automation company developing software solutions that integrate with third-party hardware to empower property owners, managers and homebuilders to effectively manage, protect and automate daily operational processes. Elixir powers our platform from our Phoenix backend to our Nerves devices. See our engineering job openings at smartrent.com/careers.

Devices in prod

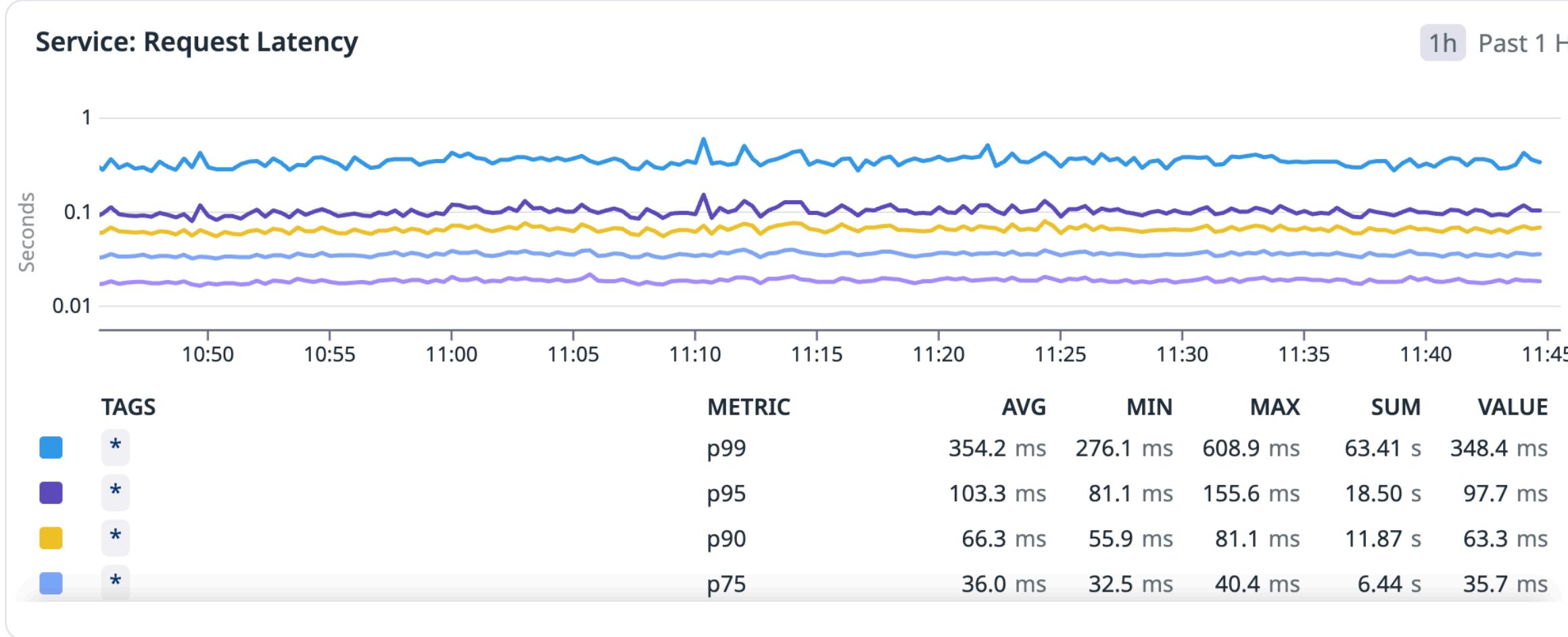
Load Balancer: Active Connections Per Second



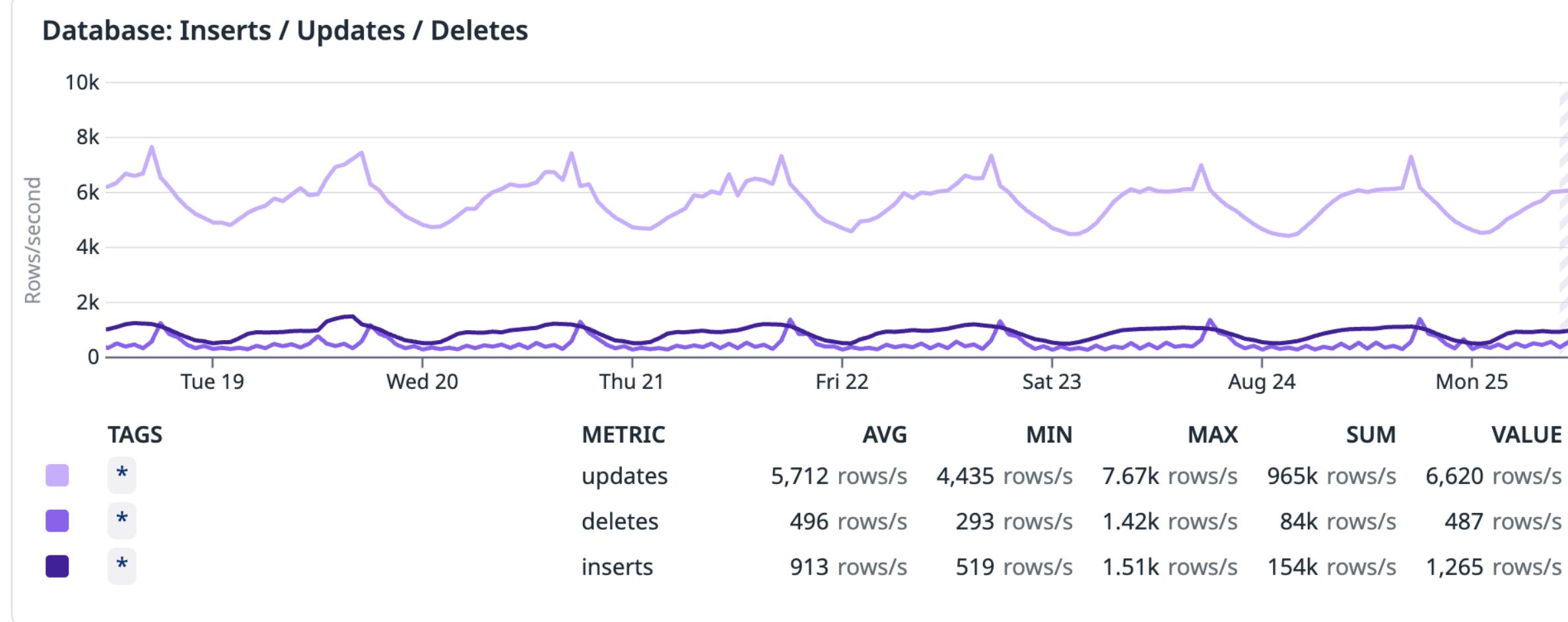
Queue: Messages Sent Per Second



Service: Request Latency



Database: Inserts / Updates / Deletes



Telemetry-driven development?

From Wikipedia, the free encyclopedia

Test-driven development (TDD) is a way of writing [code](#) that involves writing an [automated unit-level test case](#) that fails, then writing just enough code to make the test pass, then [refactoring](#) both the test code and the production code, then repeating with another new test case.

Alternative approaches to writing automated tests is to write all of the production code before starting on the test code or to write all of the test code before starting on the production code. With TDD, both are written together, therefore shortening debugging time necessities.^[1]

TDD is related to the test-first programming concepts of [extreme programming](#), begun in 1999,^[2] but more recently has created more general interest in its own right.^[3]

Programmers also apply the concept to improving and [debugging legacy code](#) developed with older techniques.^[4]

Part of a series on
Software development

“Purpose”

The screenshot shows a Wikipedia page titled "The purpose of a system is what it does". The page is in English and has 3 languages available. The main content discusses the heuristic POSIWID (Positive Invariant) coined by Stafford Beer. It states that the purpose of a system is what it does, rather than what it was designed to do. The page includes sections for Article, Talk, Read, Edit, View history, and Tools.

The purpose of a system is what it does (POSIWID) is a heuristic in systems thinking coined by the British management consultant [Stafford Beer](#),^[1] who stated that there is "no point in claiming that the purpose of a system is to do what it constantly fails to do".^[2] It is widely used by [systems theorists](#), and is generally invoked to counter the notion that the purpose of a system can be read from the intentions of those who [design](#), operate or promote it. When a system's [side effects](#) or [unintended consequences](#) reveal that its behaviour is poorly understood, then the POSIWID perspective can balance political understandings of system behaviour with a more straightforwardly [descriptive view](#).

The screenshot shows a Wikipedia page titled "Systems thinking". The page is in English and has 3 languages available. It defines systems thinking as a way of making sense of the complexity of the world by looking at it in terms of wholes and relationships rather than by splitting it down into its parts. It has been used as a way of exploring and developing effective action in complex contexts, enabling systems change.^{[4][5]} Systems thinking draws on and contributes to systems theory and the system sciences.^[6]

Systems thinking is a way of making sense of the complexity of the world by looking at it in terms of wholes and relationships rather than by splitting it down into its parts.^{[1][2]} It has been used as a way of exploring and developing effective action in complex contexts,^[3] enabling systems change.^{[4][5]} Systems thinking draws on and contributes to systems theory and the system sciences.^[6]



“The best time to utilize telemetry in your Elixir application was July 3rd, 2021^{*}; the second best time is now.

If you (or an AI/LLM) really want to understand the purpose of your system, you need some form of telemetry.”

- Noah Betzen

Versions (12)

- [1.3.0](#) Aug 22, 2024
- [1.2.1](#) Jan 12, 2023
- [1.2.0](#) Jan 03, 2023
- [1.1.0](#) Apr 01, 2022
- [***1.0.0**](#) Jul 03, 2021

[Show All Versions](#)

Outline

1. Recap Telemetry + OpenTelemetry
2. Introduce basic local telemetry setup (demo!)
3. MIX_ENV=dev
4. MIX_ENV=test
5. MIX_ENV=prod
6. Challenges combining Elixir + OpenTelemetry
7. Future of telemetry-driven development

“Telemetry”

https://en.wikipedia.org/wiki/Telemetry

WIKIPEDIA
The Free Encyclopedia

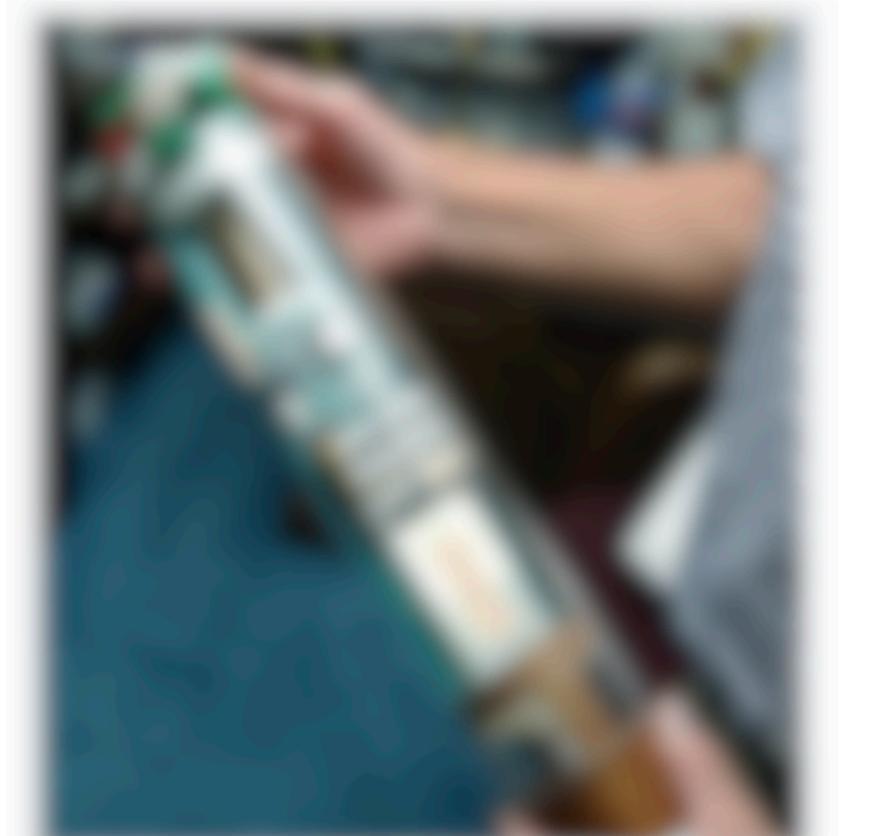
Telemetry

From Wikipedia, the free encyclopedia

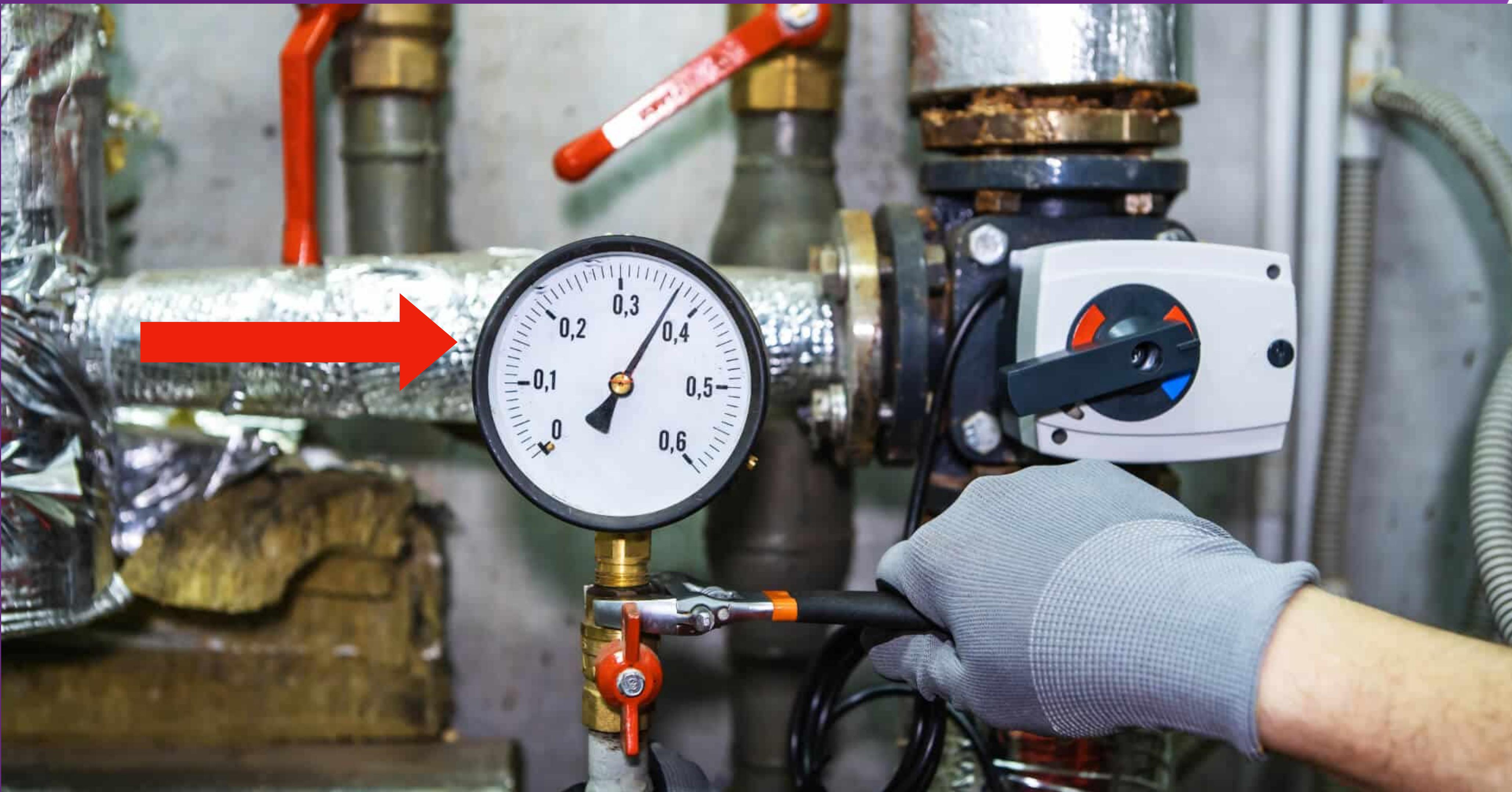
Not to be confused with [Telemetry \(company\)](#) or [Telemetry \(software\)](#).

Telemetry is the [in situ collection of measurements](#) or other data at remote points and their automatic [transmission](#) to receiving equipment ([telecommunication](#)) for [monitoring](#).^[1] The word is derived from the [Greek roots](#) *tele*, 'far off' and *metron*, 'measure'. Systems that need external instructions and data to operate require the counterpart of telemetry: [telecommand](#).^[2]

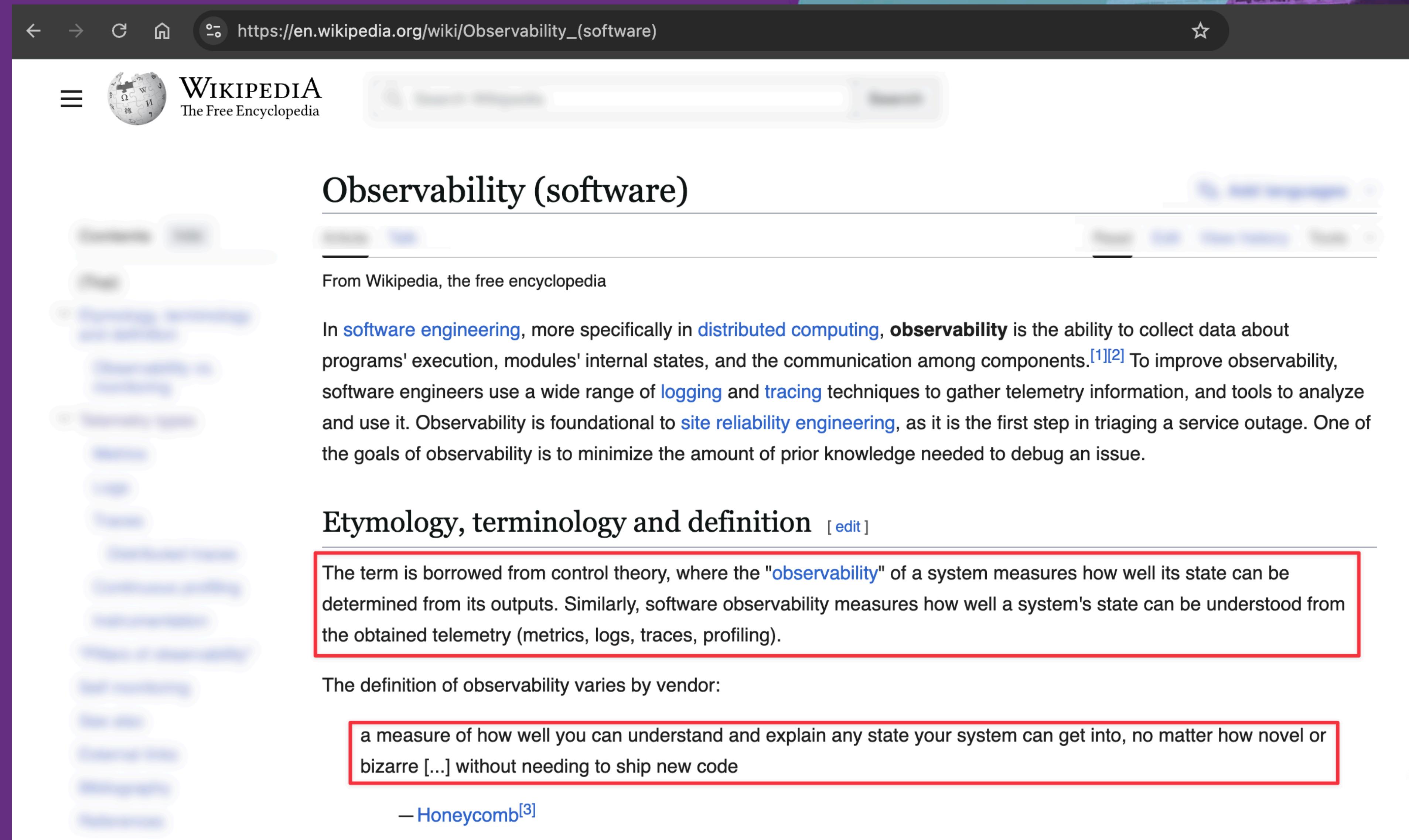
Although the term commonly refers to [wireless](#) data transfer mechanisms (e.g., using [radio](#), ultrasonic, or [infrared](#) systems), it also encompasses data transferred over other media such as a telephone or [computer network](#), optical link or other wired communications like power line carriers. Many modern telemetry systems take advantage of the low cost and ubiquity of [GSM](#) networks by using [SMS](#) to receive and transmit telemetry data.



“Telemetry”



“Observability”



The screenshot shows a web browser displaying the English Wikipedia article on "Observability (software)". The URL in the address bar is [https://en.wikipedia.org/wiki/Observability_\(software\)](https://en.wikipedia.org/wiki/Observability_(software)). The Wikipedia logo and the text "The Free Encyclopedia" are visible at the top left. The main title "Observability (software)" is centered above a horizontal line. Below the title, a short summary states: "From Wikipedia, the free encyclopedia". The main content begins with a paragraph explaining that observability is the ability to collect data about programs' execution, modules' internal states, and communication among components. It notes that software engineers use logging and tracing techniques to gather telemetry information and tools to analyze it. Observability is described as foundational to site reliability engineering. A red box highlights a quote from control theory: "The term is borrowed from control theory, where the "observability" of a system measures how well its state can be determined from its outputs. Similarly, software observability measures how well a system's state can be understood from the obtained telemetry (metrics, logs, traces, profiling)." Another red box contains a quote from Honeycomb: "a measure of how well you can understand and explain any state your system can get into, no matter how novel or bizarre [...] without needing to ship new code". The bottom right corner of the slide features a small image of a bridge.

From Wikipedia, the free encyclopedia

In [software engineering](#), more specifically in [distributed computing](#), **observability** is the ability to collect data about programs' execution, modules' internal states, and the communication among components.^{[1][2]} To improve observability, software engineers use a wide range of [logging](#) and [tracing](#) techniques to gather telemetry information, and tools to analyze and use it. Observability is foundational to [site reliability engineering](#), as it is the first step in triaging a service outage. One of the goals of observability is to minimize the amount of prior knowledge needed to debug an issue.

Etymology, terminology and definition [edit]

The term is borrowed from control theory, where the "observability" of a system measures how well its state can be determined from its outputs. Similarly, software observability measures how well a system's state can be understood from the obtained telemetry (metrics, logs, traces, profiling).

The definition of observability varies by vendor:

a measure of how well you can understand and explain any state your system can get into, no matter how novel or bizarre [...] without needing to ship new code

— Honeycomb^[3]

Elixir Telemetry / Metrics

<https://hex.pm/packages/telemetry>

telemetry 1.3.0

Dynamic dispatching library for metrics and instrumentations

License
Apache-2.0

Downloads

Last 30 days, all versions

Period	Downloads
this version	13 336 310
yesterday	20 122
last 7 days	421 405
all time	143 724 943

Versions (12)

- 1.3.0 Aug 22, 2024
- 1.2.1 Jan 12, 2023
- 1.2.0 Jan 03, 2023
- 1.1.0 Apr 01, 2022
- 1.0.0 Jul 02, 2021

Dependencies (0)

Owners

- josevalim
- wojtekmach
- arkgil

Publisher

- wojtekmach

https://hex.pm/packages/telemetry_metrics

telemetry_metrics 1.1.0

Provides a common interface for defining metrics based on Telemetry events.

License
Apache-2.0

Downloads

Last 30 days, all versions

Period	Downloads
this version	3 125 875
yesterday	10 717
last 7 days	280 440
all time	38 915 638

Versions (14)

- 1.1.0 Jan 24, 2025
- 1.0.0 Mar 18, 2024
- 0.6.2 Jan 11, 2024

Dependencies (1)
telemetry ~> 0.4 or ~> 1.0

Owners

- josevalim
- arkgil

Publisher

- josevalim

https://opentelemetry.io/docs/

Documentation

OpenTelemetry, also known as OTel, is a vendor-neutral open source [Observability](#) framework for instrumenting, generating, collecting, and exporting telemetry data such as [traces](#), [metrics](#), and [logs](#).

```
graph LR; subgraph Microservices [Microservices]; direction TB; A[OTel Auto. Inst.] --> C[OTel Collector]; B[OTel API] --> C; D[OTel SDK] --> C; end; subgraph SharedInfra [Shared Infra]; direction TB; E[Kubernetes] --> C; F[L7 Proxy] --> C; G[aws] --> C; end; subgraph ClientInstrumentation [Client Instrumentation]; direction TB; H[Managed DBs] --> C; I[APIs] --> C; end; C[OTel Collector] --> J[3rd party service]; C --> K[Observability Frontends & APIs]; subgraph ObservabilityFrontends [Observability Frontends & APIs]; direction TB; L[Time Series Databases] --> K; M[Trace Databases] --> K; N[Column Stores] --> K; end;
```

OpenTelemetry “Signals”

Signals

Learn about the categories of telemetry supported by OpenTelemetry

The purpose of OpenTelemetry is to collect, process, and export [signals](#). Signals are system outputs that describe the underlying activity of the operating system and applications running on a platform. A signal can be something you want to measure at a specific point in time, like temperature or memory usage, or an event that goes through the components of your distributed system that you'd like to trace. You can group different signals together to observe the inner workings of the same piece of technology under different angles.

OpenTelemetry currently supports:

- [Traces](#)
- [Metrics](#)
- [Logs](#)
- [Baggage](#)

Also under development or at the [proposal](#) stage:

- [Events](#), a specific type of [log](#)
- [Profiles](#) are being worked on by the Profiling Working Group.



Elixir + OpenTelemetry

The current status of the major functional components for OpenTelemetry Erlang/Elixir is as follows:

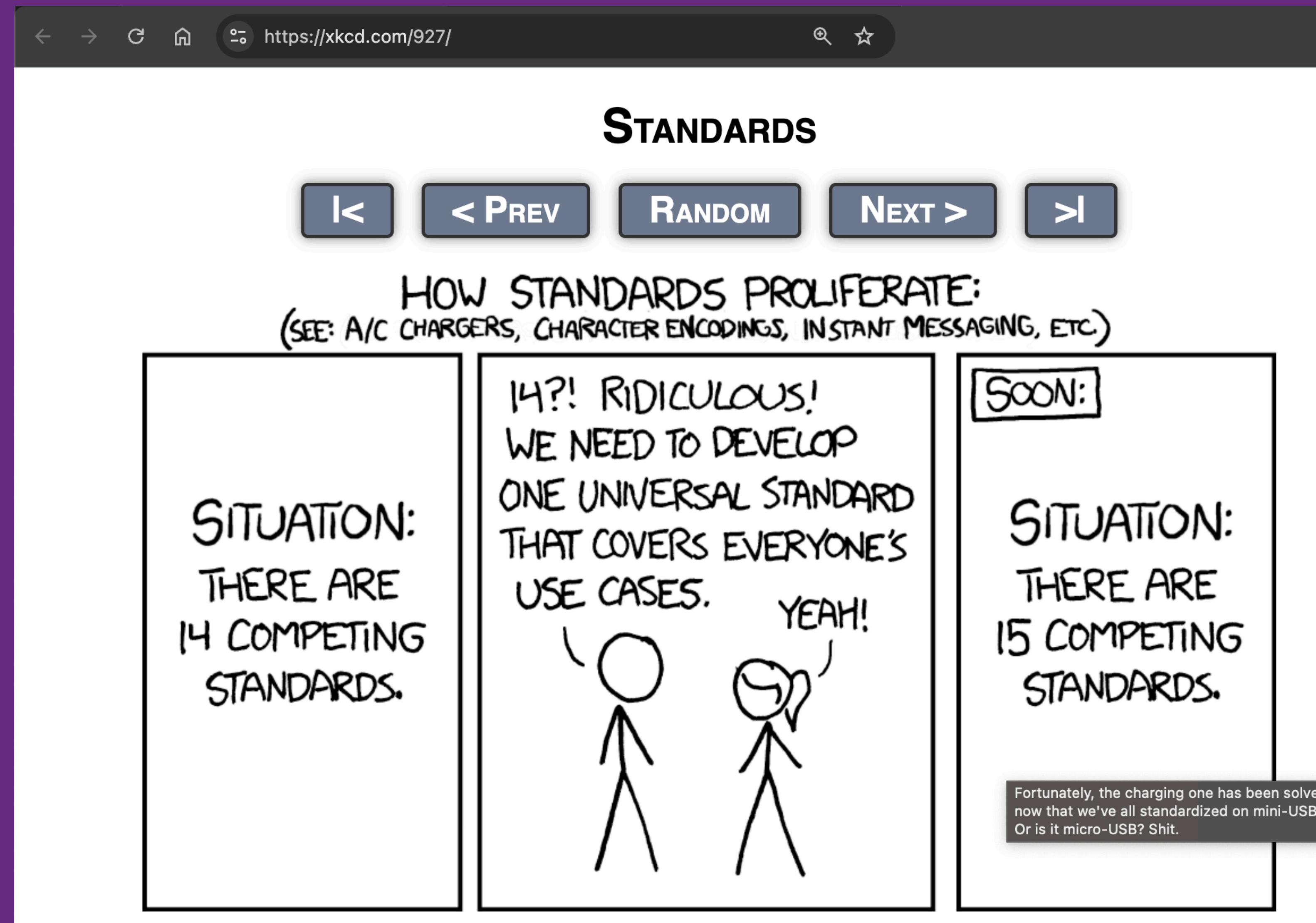
Traces	Metrics	Logs
Stable	Development	Development

Packages of the API, SDK and OTLP exporter are published to [hex.pm](#) as [opentelemetry_api](#), [opentelemetry](#) and [opentelemetry_exporter](#).

Repositories

- [opentelemetry-erlang](#): Main repository containing the API, SDK and OTLP Exporter.
- [opentelemetry-erlang-contrib](#): Helpful libraries and instrumentation libraries for Erlang/Elixir projects like [Phoenix](#) and [Ecto](#).

Elixir + OpenTelemetry



Elixir + OpenTelemetry

https://hex.pm/packages?search=opentelemetry&sort=recent_downloads

hex opentelemetry

Packages

Exact Match:

opentelemetry 1.5.1	Implementation of stable OpenTelemetry signals	1 467 307 recent downloads
64 Results Found		
Sort: Recent downloads ▾		
Search Results:		
opentelemetry_api 1.4.1	OpenTelemetry API	1 575 433 recent downloads
opentelemetry_exporter 1.8.1	OpenTelemetry Protocol Exporter	1 417 803 recent downloads
opentelemetry_semantic_conventions 1.27.0	OpenTelemetry Semantic Conventions	1 335 905 recent downloads
opentelemetry_telemetry 1.1.2	Bridge library between Telemetry events and OpenTelemetry Erlang	1 215 909 recent downloads
opentelemetry_process_propagator 0.3.0	Tools for opentelemetry context propagation across process boundaries	1 177 519 recent downloads
opentelemetry_phoenix 2.0.1	OpenTelemetry tracing for the Phoenix Framework	1 071 076 recent downloads



<https://knowyourmeme.com/memes/mr-bones-wild-ride>



Deep breath



Water break

Context / Demo

Timeline / Context

- [January 19, 2022: "Monitoring Elixir With OpenTelemetry"](#)
- [June 22, 2022: "BEAM + Prometheus + Grafana = Observability Heaven"](#)
- [November 14, 2022: "Elixir, OpenTelemetry, and the Infamous N+1"](#)
- [November 3, 2023: Tweet: "Webinar: Test driven development with OpenTelemetry"](#)
- [February 15, 2024: "All you need is Wide Events, not "Metrics, Logs and Traces"](#)
- [February 17, 2024: "Observability for Phoenix using the Grafana Stack in Dev"](#)
- [April 4, 2024: Cory O'Daniel DMs me a link to `tracetest`](#)
- [April 16, 2024: "Using OpenTelemetry To Troubleshoot & Monitor Production Applications"](#)
- [December 11, 2024: "Instrumenting Erlang and Elixir code with open telemetry"](#)
- [December 12, 2024: "Telemetry & Observability for Elixir Apps"](#)
- [February 6, 2025: "Let the code to test emit telemetry events of what you are interested in. Let the test be a handler for those telemetry events. " - LostKobrakai](#)
- [February 11, 2025: Initial wildfires local telemetry demo](#)
- [March 24, 2025: German Velasco asks about tracing in production](#)
- [April 27, 2025: "OpenTelemetry: From Desire to Dashboard"](#)
- [May 6, 2025: Posts about Elixir pains working with OpenTelemetry](#)
- [June 23, 2025: Lars Wikman asks about opinions on Grafana](#)
- [June 23, 2025: Łukasz Niemier mentions OpenTelemetry + DuckDB](#)
- [June 30, 2025: Jacob Swanner's proposed telemetry talk](#)

The screenshot shows a GitHub repository page for the user 'Neztek' with the repository name 'telemetry-driven-development'. The page includes navigation icons, a search bar, and links for Code, Issues, Pull requests, Actions, Security, Insights, and Settings. A profile picture of the user is displayed next to the repository name. The repository is marked as 'Private'. The background features a large, abstract graphic of a city skyline at night with a yellow diagonal shape.

Context / Demo

https://hub.docker.com/r/grafana/otel-lgtm

hub Search Docker Hub

Explore / grafana / otel-lgtm

grafana/otel-lgtm Verified Publisher

By Grafana Labs · Updated 1 day ago

An OpenTelemetry backend in a Docker image.

IMAGE

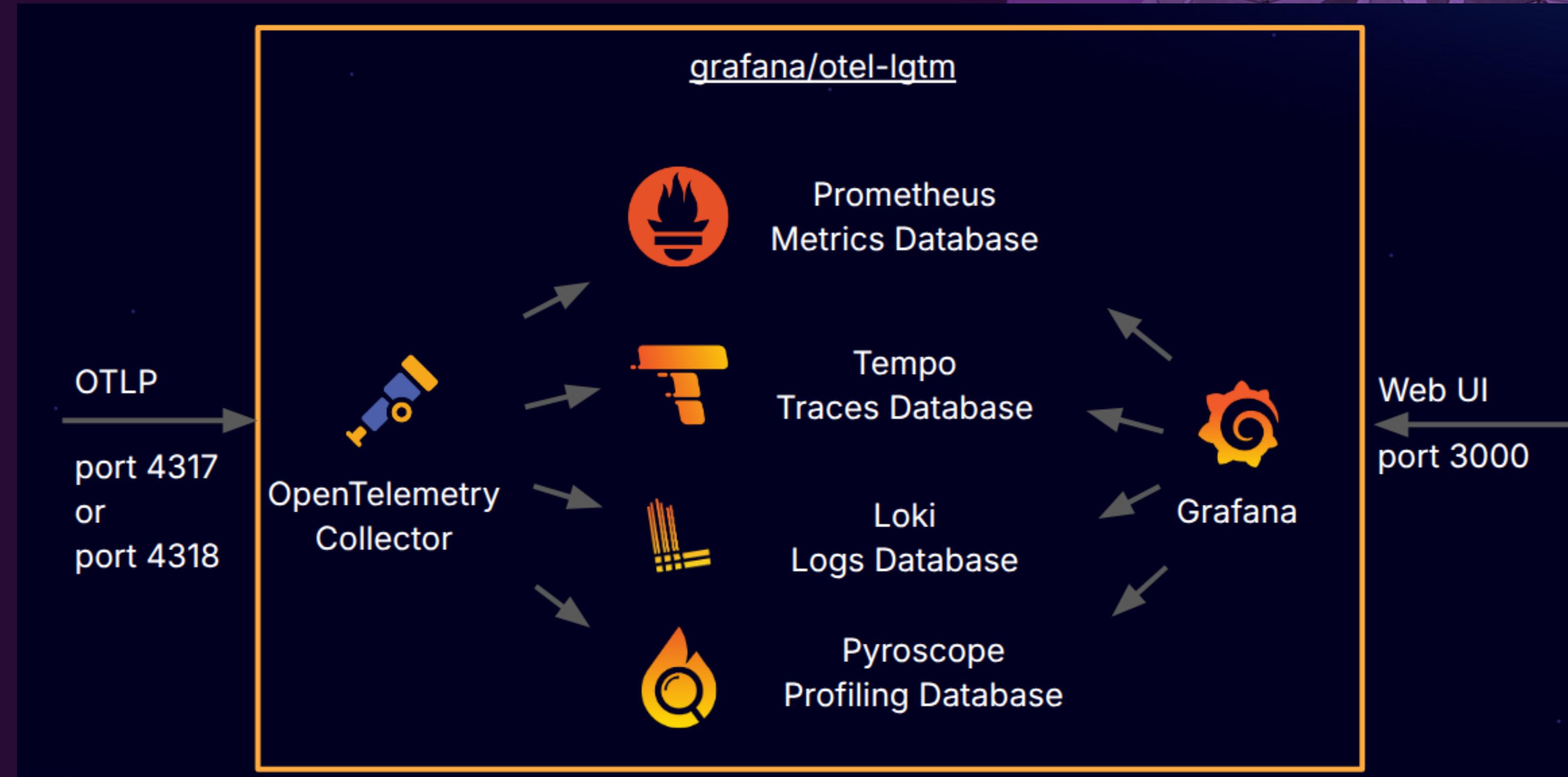
500K+ stars

Overview Tags

An OpenTelemetry Backend in a Docker Image

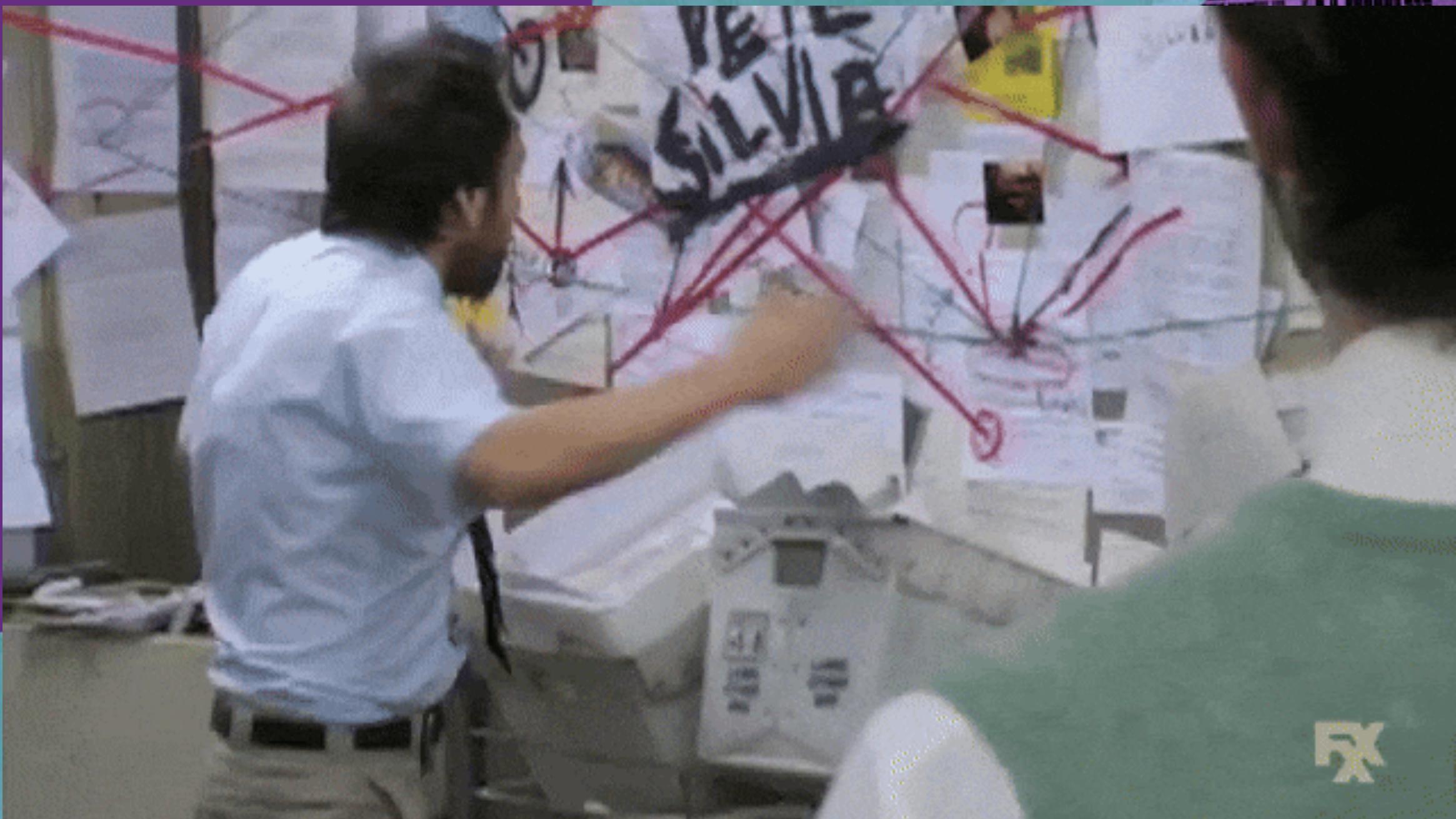
grafana/otel-lgtm contains a complete OpenTelemetry backend in a single Docker image:

- [OpenTelemetry collector](#) for receiving OpenTelemetry data.
- [Prometheus](#) metrics database.
- [Tempo](#) trace database.
- [Loki](#) logs database.
- [Grafana](#) for visualization.



Environments

1. MIX_ENV=dev
2. MIX_ENV=test
 1. Local
 2. CI
3. MIX_ENV=prod
 1. “dev”
 2. “test” / “qa”
 3. “prod”



<https://knowyourmeme.com/memes/pepe-silvia>

A black and white photograph of a rocket launching from a launch pad. The rocket is positioned vertically in the center of the frame, with its nose cone pointing upwards. A large plume of white smoke and fire is visible at the base of the rocket, indicating the start of the launch. The background shows the dark sky above the launch pad.

MIX_ENV=dev

A black and white photograph of a rocket launching from a launch pad. The rocket is positioned vertically in the center of the frame, with its nose cone pointing upwards. A large plume of white smoke and fire is visible at the base of the rocket, indicating the start of the launch. The background shows the dark sky above the launch pad.

MIX_ENV=test

A large, white text overlay is positioned in front of a background image of a rocket launching. The rocket is shown from a low angle, rising vertically against a dark sky. A thick plume of white smoke and fire erupts from its base. The background features a complex network of metal structures and scaffolding of the launch pad.

MIX_ENV=prod

“development” “test”, “UAT”, “QA”, “demo”, “prod-us”, “prod-eu”, etc.



Deep breath



Water break

MIX_ENV	dev	test	prod
Logs?	Yes	As needed	Yes
Metrics?	🤔	🤔	Yes
Tracing?	🤔	🤔	Yes

Environment	“Develop”	“Test”	“US Prod”	“EU Prod”
Logs?	🤔	🤔	Yes	Yes
Metrics?	🤔	🤔	Yes	Yes
Tracing?	🤔	🤔	Yes	Yes

Telemetry Advice

“If you can ingest a hundred percent of your traces, I would just go crazy there and focus all my energy on traces. I wouldn't worry about metrics. I **really** wouldn't worry about logs.



My ultimate hot take is, ‘**logs are garbage**’. They're super expensive and really low-utility in my opinion.”

- Ethan Gunderson

The screenshot shows a web browser displaying a podcast episode page. The URL in the address bar is <https://smartlogic.io/podcast/elixir-wizards/s13-e09-observability-telemetry-elixir-cars-commerce/>. The page header features the SmartLogic logo and navigation links for Services, Technologies, Method, Case Studies, and Podcast. The main content area has a teal header with the title "Telemetry & Observability for Elixir Apps at Cars.com with Zack Kayser & Ethan Gunderson". Below the title is a thumbnail image of two men, Zack Kayser and Ethan Gunderson, wearing headphones. The episode details include the title "Elixir Wizards a SmartLogic podcast", the subtitle "Telemetry & Observability for Elixir Apps at Cars.com with Zack Kayser & Ethan Gunderson", and the episode number "S13E09 INSTRUMENTING Elixir: TELEMETRY AT CARS.COM WITH ZACK KAYSER & ETHAN GUNDERSON". There are also buttons for PLAY, SHARE, DOWNLOAD, SUBSCRIBE, and TRANSCRIPT, along with a progress bar showing the duration as 00:30:38 / 00:42:39 and a 1.0x speed setting. Below the player, the text "About this Episode" is followed by "Published December 12, 2024 | Duration: 42:39 | RSS Feed | Direct download" and "Transcript: English".

“Telemetry isn’t free”

- Business value: signal vs noise
 - “Signals”
 - Logging
 - Metrics
 - Traces
- Effort: Encoding, transmitting, parsing, storing, etc.
- Goodhart's Law
 - "When a measure becomes a target, it ceases to be a good me
- Some ad hoc telemetry data could just be application data
 - `SELECT count(*) FROM <table>`

Future

Status and Releases

The current status of the major functional components for OpenTelemetry Erlang/Elixir is as follows:

Traces Metrics Logs

[Stable](#) [Development](#) [Development](#)

For releases, including the [latest release](#), see [Releases](#).

Packages of the API, SDK and OTLP exporter are published to [hex.pm](#) as [opentelemetry_api](#), [opentelemetry](#) and [opentelemetry_exporter](#).

Status and Releases

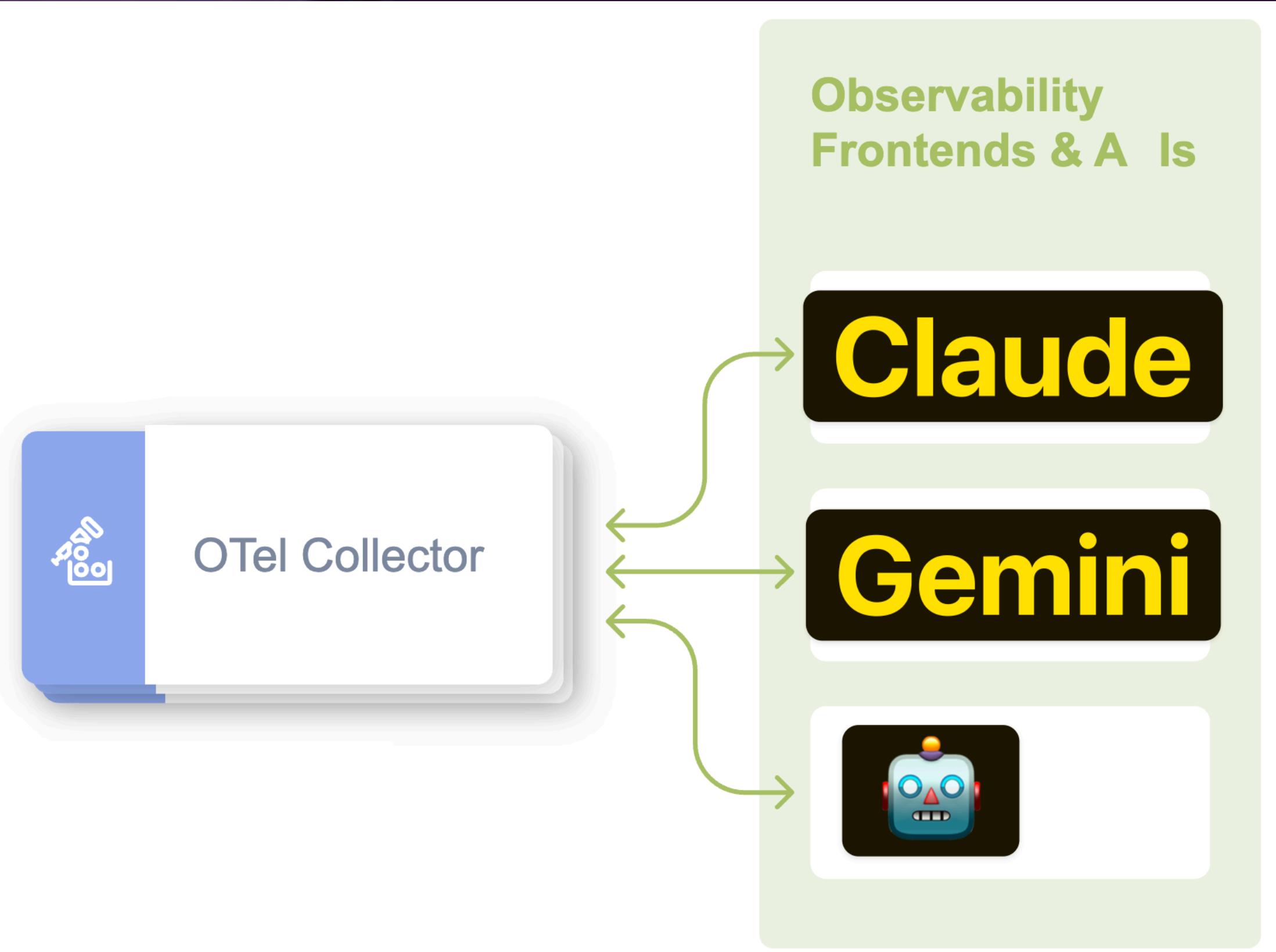
The current status of the major functional components for OpenTelemetry Erlang/Elixir is as follows:

Traces Metrics Logs

[Stable](#) [Stable](#) [Stable](#)

For releases, including the [latest release](#), see [Releases](#).

Packages of the API, SDK and OTLP exporter are published to [hex.pm](#) as [opentelemetry_api](#), [opentelemetry](#) and [opentelemetry_exporter](#).



Recap

1. Recap Telemetry + OpenTelemetry
2. Introduce basic local telemetry setup (demo!)
3. MIX_ENV=dev
4. MIX_ENV=test
5. MIX_ENV=prod
6. Challenges combining Elixir + OpenTelemetry
7. Future of telemetry-driven development

A photograph of a young man with short brown hair and black-rimmed glasses, wearing a dark t-shirt. He is standing in front of a microphone stand, clapping his hands. In the background, there are other people seated at tables, suggesting a conference or event setting.

Q&A

Secret Spicy Take

“If you decide to track product-specific analytics using typical engineering telemetry tooling, you will die.”

There are two hard problems in Computer Science:

0. Naming things
1. Cache invalidation
2. Off-by-one errors