

# Unlocking Observability in **WebAssembly** with **OpenTelemetry**

Caleb Schoepp

# About me



- Caleb Schoepp
- Software Engineer @ Fermyon
- Spin maintainer
- Proud Canadian 🇨🇦🏒🍁
- Online @ calebschoepp.com



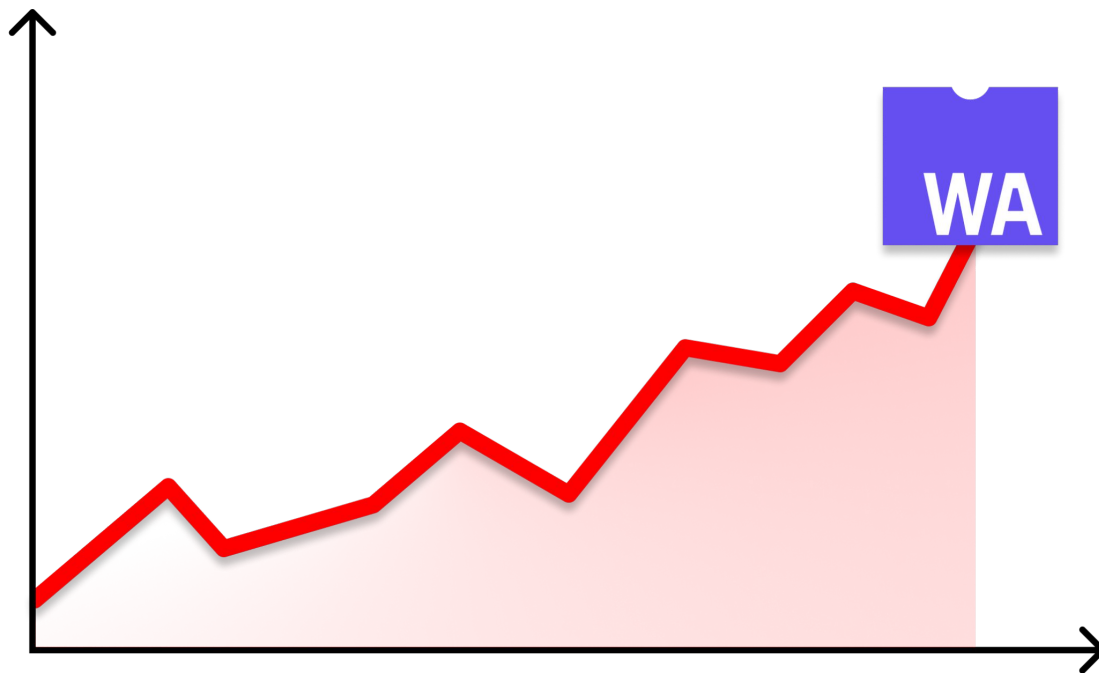
# What we'll cover today

- A quick introduction to observability and OpenTelemetry
- Three places we can collect telemetry in WebAssembly
- WebAssembly runtime auto instrumentation and its future
- How to emit traces from your WebAssembly components
- Design and progress of the wasi-otel proposal

# What is observability ?

- Understand a system from the outside... 
- By asking arbitrary questions... ?
- Without understanding the inside! 
- It's all about the unknown unknowns of our complex systems
- Observability  $\equiv$  o11y

Why does **o11y** matter to **Wasm**?



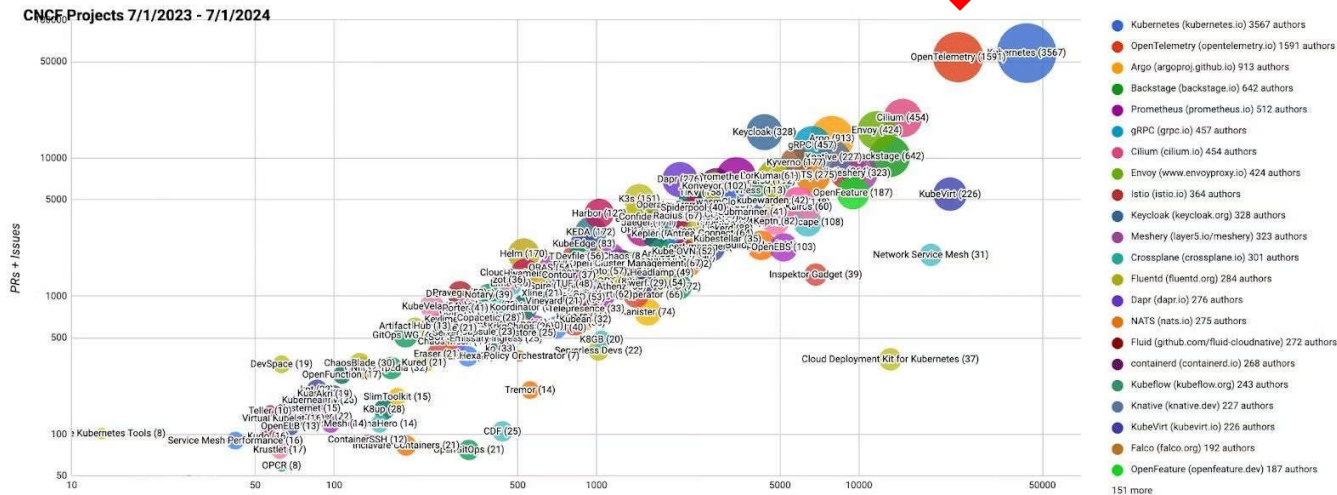
# What is **OpenTelemetry** ?

- Open source observability framework
- OpenTelemetry  $\equiv$  OTel



# What is OpenTelemetry ?

- Standardization and no vendor lock in
- A CNCF project since 2019



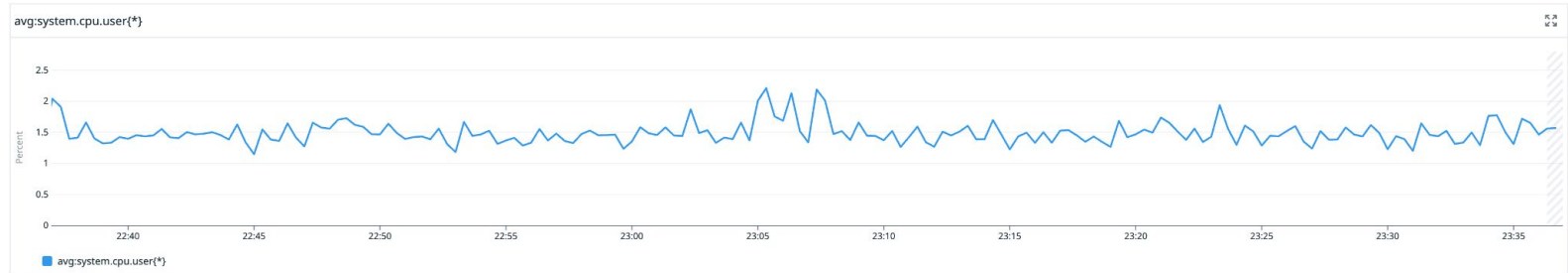
**OTel** helps instrument your code to  
produce telemetry



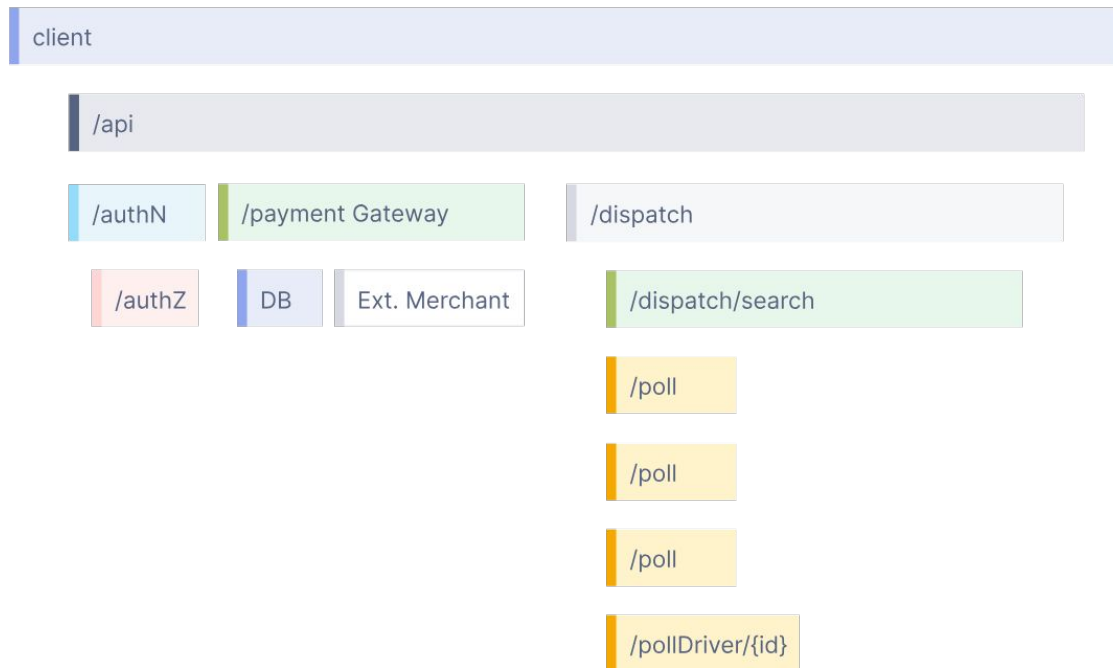
# Logs

```
127.0.0.1 - [04/Aug/2024:12:34:56 -0400] "POST /api/v1/login HTTP/1.1" 200 1234
192.168.1.5 - [04/Aug/2024:12:36:12 -0400] "GET /dashboard HTTP/1.1" 200 5678
10.0.0.23 - [04/Aug/2024:12:38:45 -0400] "GET /images/logo.png HTTP/1.1" 200 3456
172.16.254.1 - [04/Aug/2024:12:40:21 -0400] "PUT /api/v1/users/profile HTTP/1.1" 204 0
8.8.8.8 - [04/Aug/2024:12:45:32 -0400] "GET /products?category=electronics HTTP/1.1" 200 8901
203.0.113.7 - [04/Aug/2024:12:47:18 -0400] "GET /about HTTP/1.1" 200 2345
198.51.100.42 - [04/Aug/2024:12:52:30 -0400] "DELETE /api/v1/cart/item/12345 HTTP/1.1" 200 567
157.240.22.35 - [04/Aug/2024:12:58:11 -0400] "POST /api/v1/orders HTTP/1.1" 201 4321
104.198.14.52 - [04/Aug/2024:13:01:45 -0400] "GET /support/faq HTTP/1.1" 200 6789
45.33.32.156 - [04/Aug/2024:13:05:03 -0400] "PATCH /api/v1/settings HTTP/1.1" 200 890
```

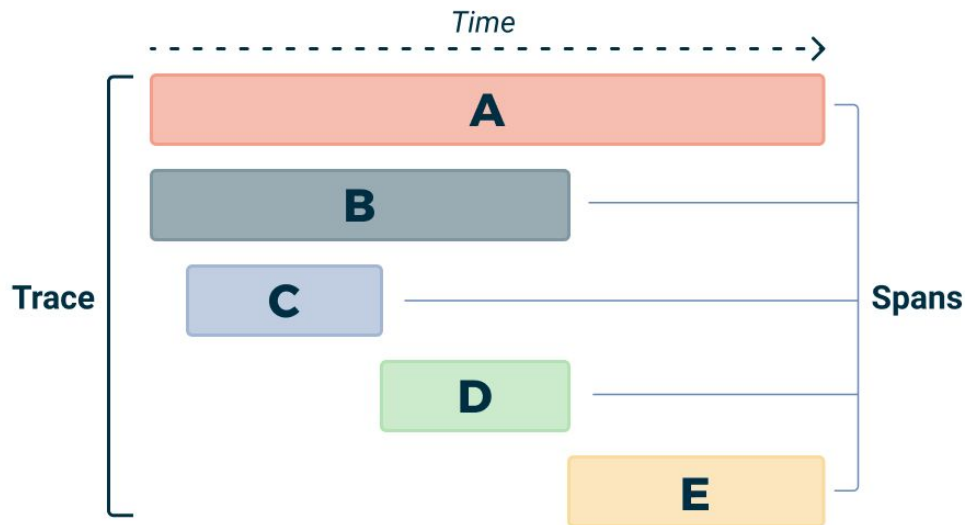
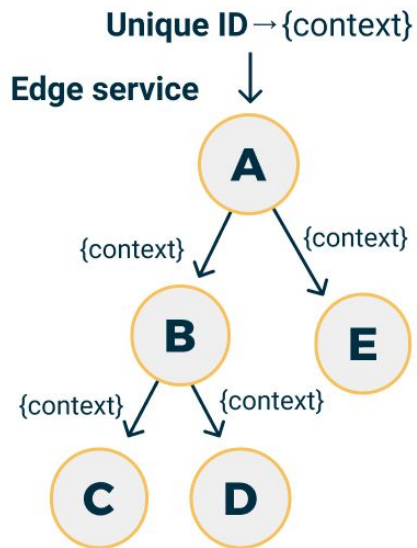
# Metrics



# Traces



# Distributed Tracing



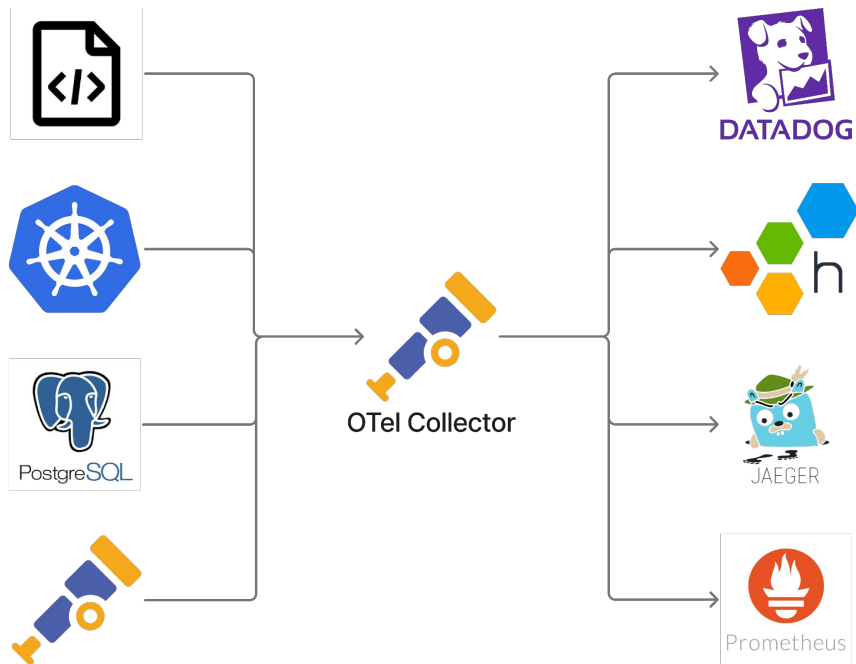


**Tracing** will be our focus



**OTel** helps collect, process, and  
forward your telemetry

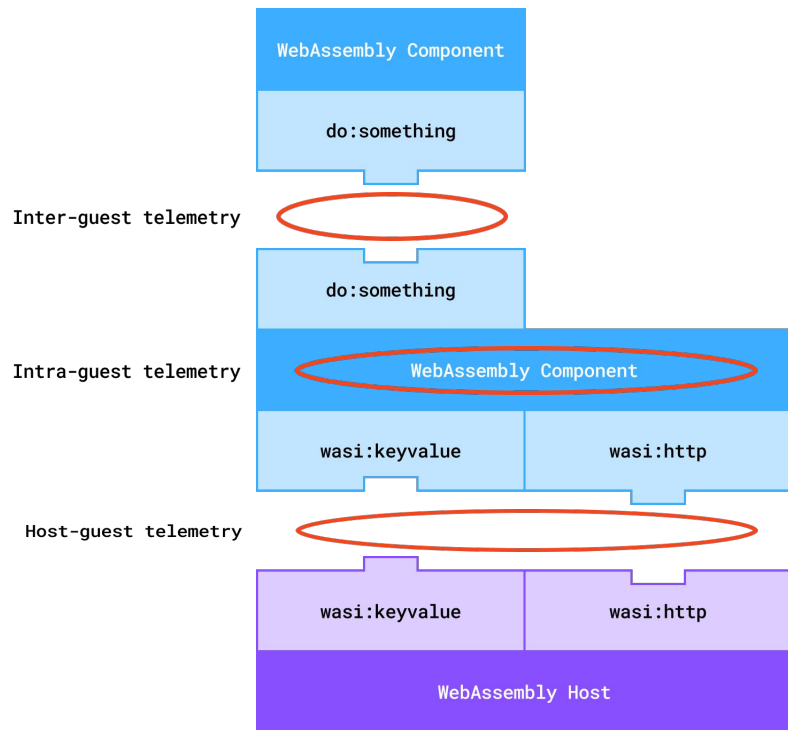
# Via the OTel Collector



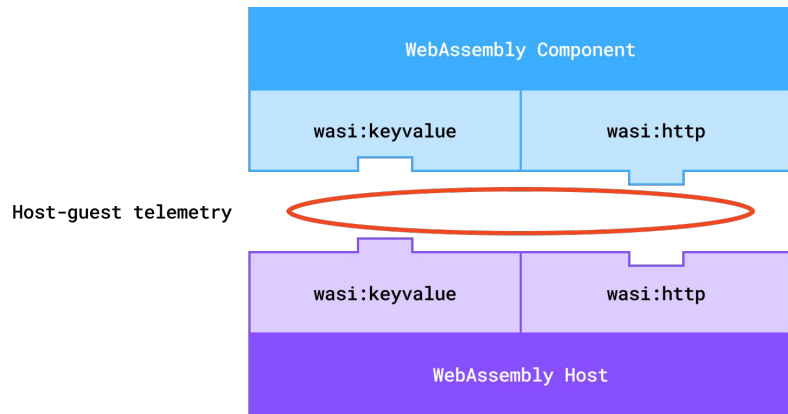
 **OTel** meet **Wasm** 



# Three places we can collect **telemetry**

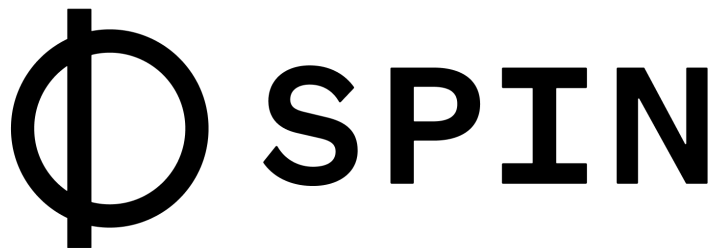


# Wasm runtimes can give us auto instrumentation



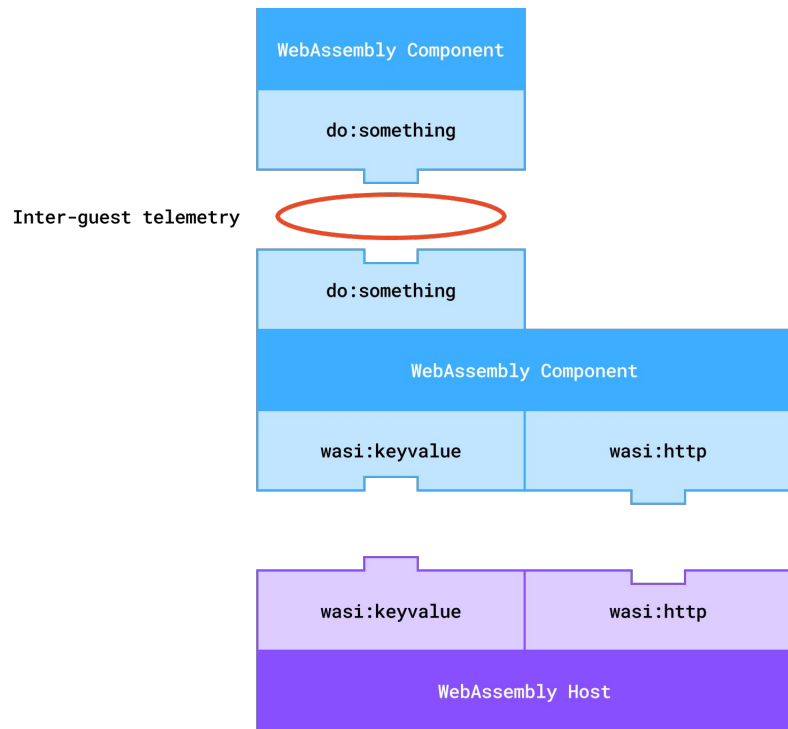
# What is **Spin**?

- Open source tool for serverless WebAssembly
- A CNCF Sandbox project
- spin new → spin build → spin up

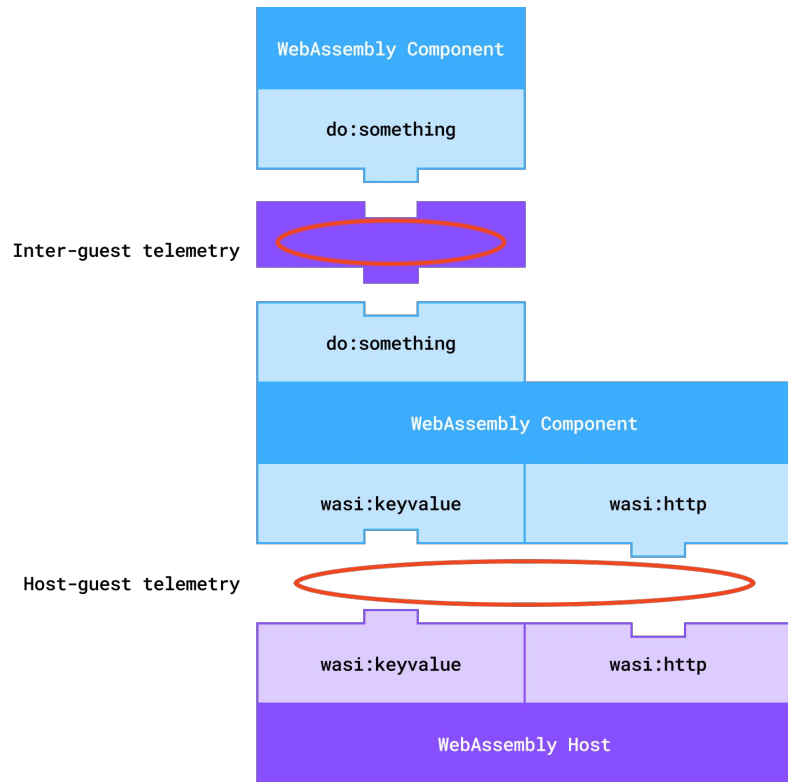




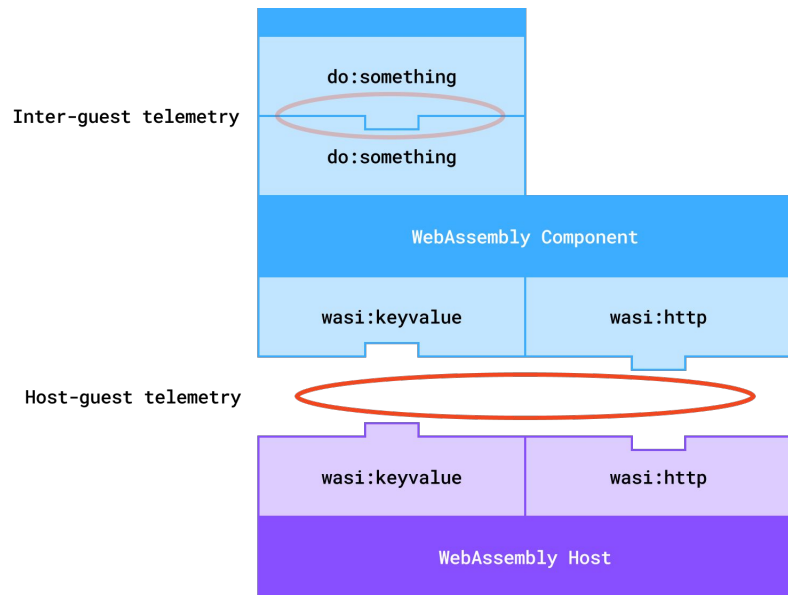
# Inter-guest telemetry is runtime dependent



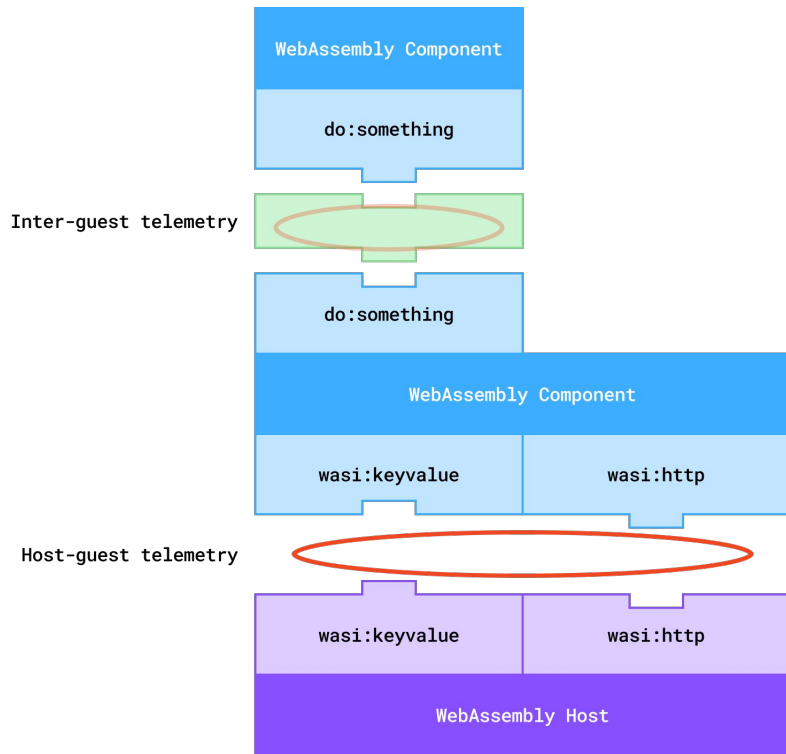
# Dynamically composed components



# Statically composed components



# Statically composed components

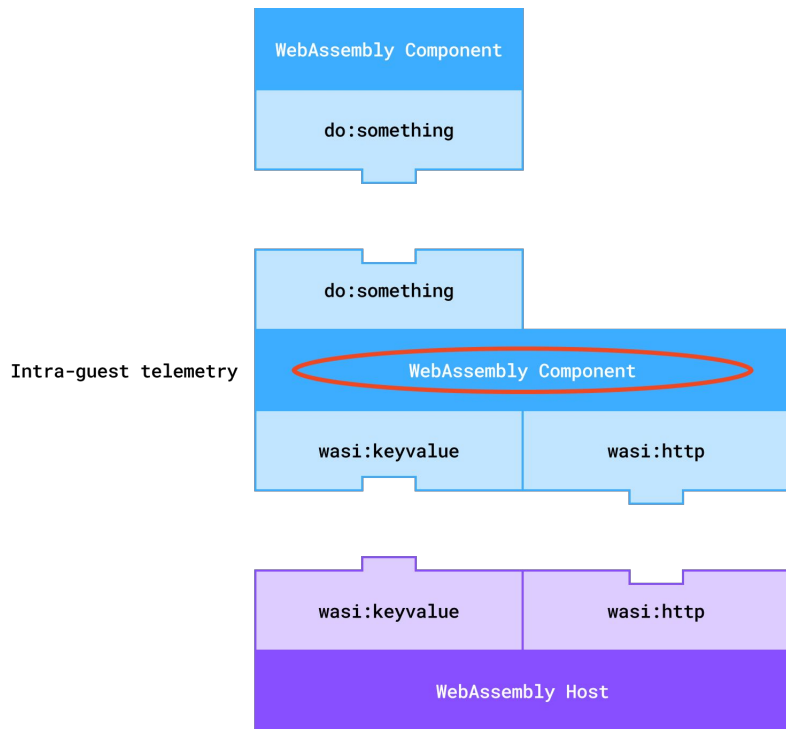




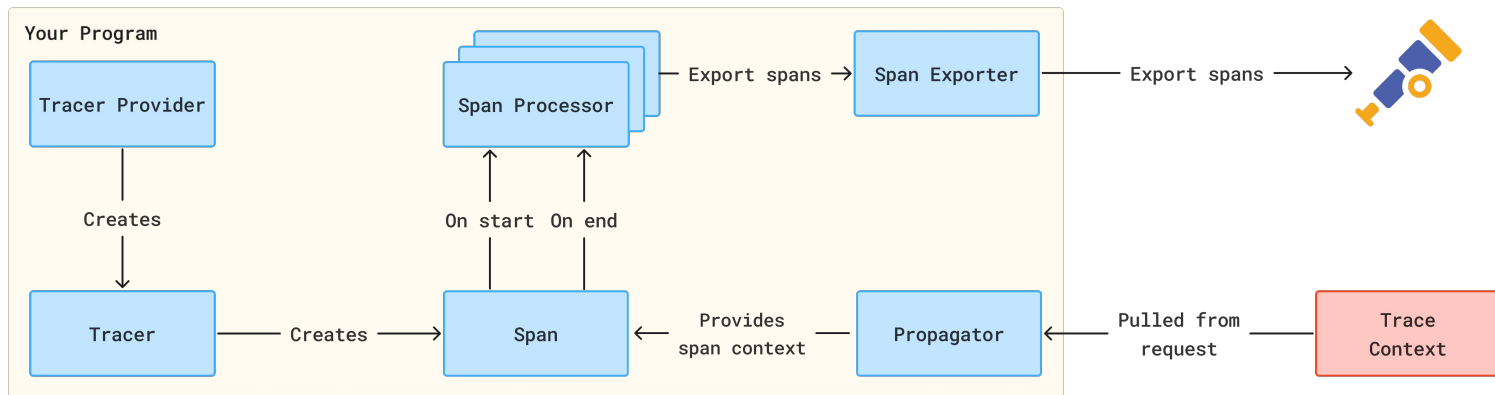
# Wasm runtime auto instrumentation

- No effort required from developer ✓
- Cross-language ✓
- A bright future with inter-guest telemetry ✓
- Not customizable by the developer ✗

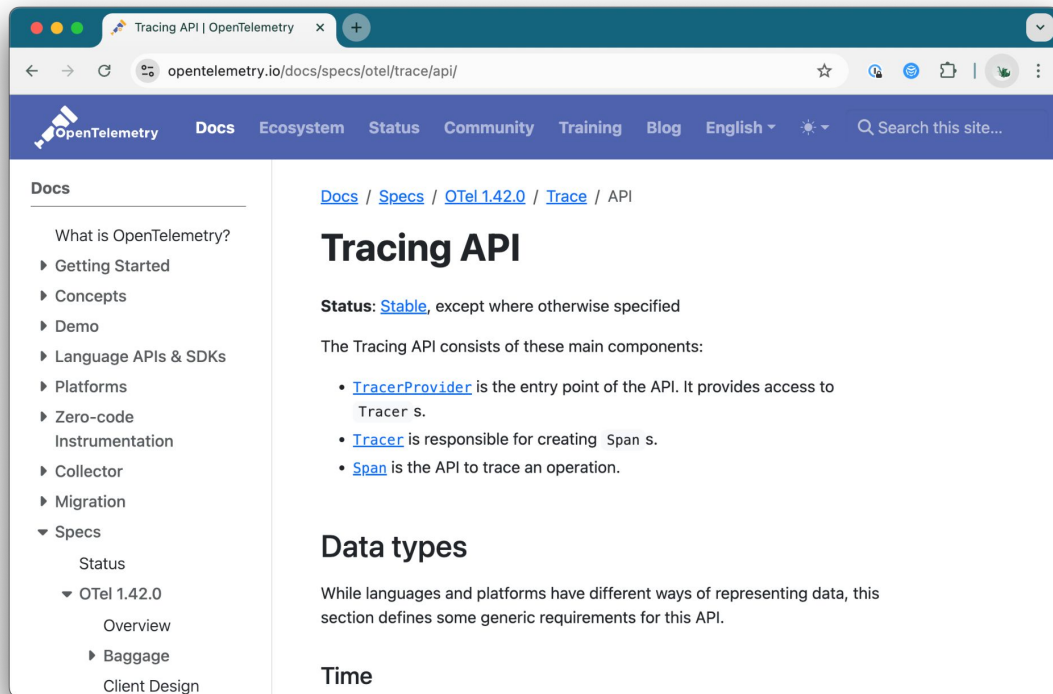
# Let's emit **traces** from the **component**



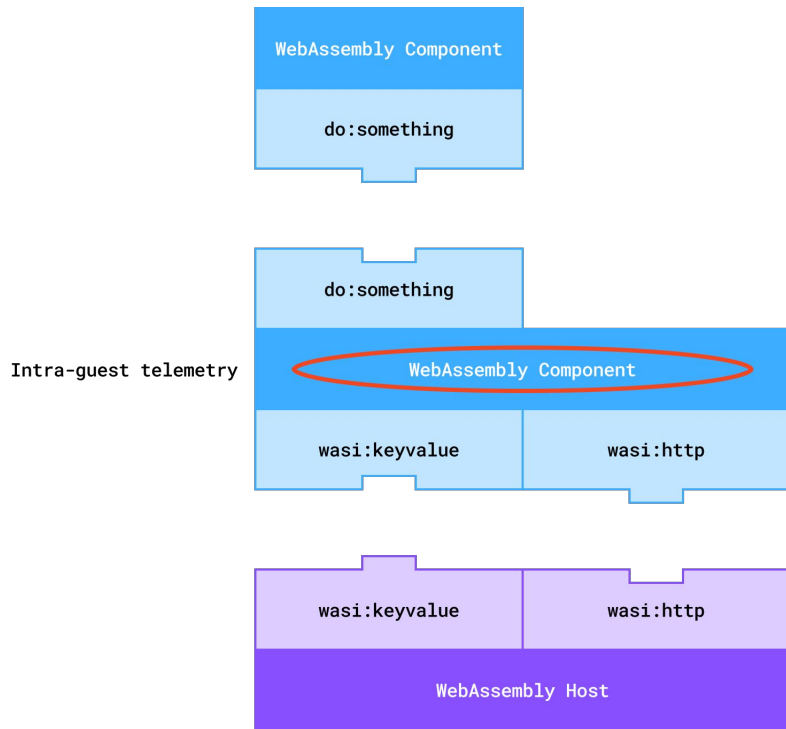
# First we must understand **tracing instrumentation**



# The OTel specification



# We have two options for intra- **guest** telemetry

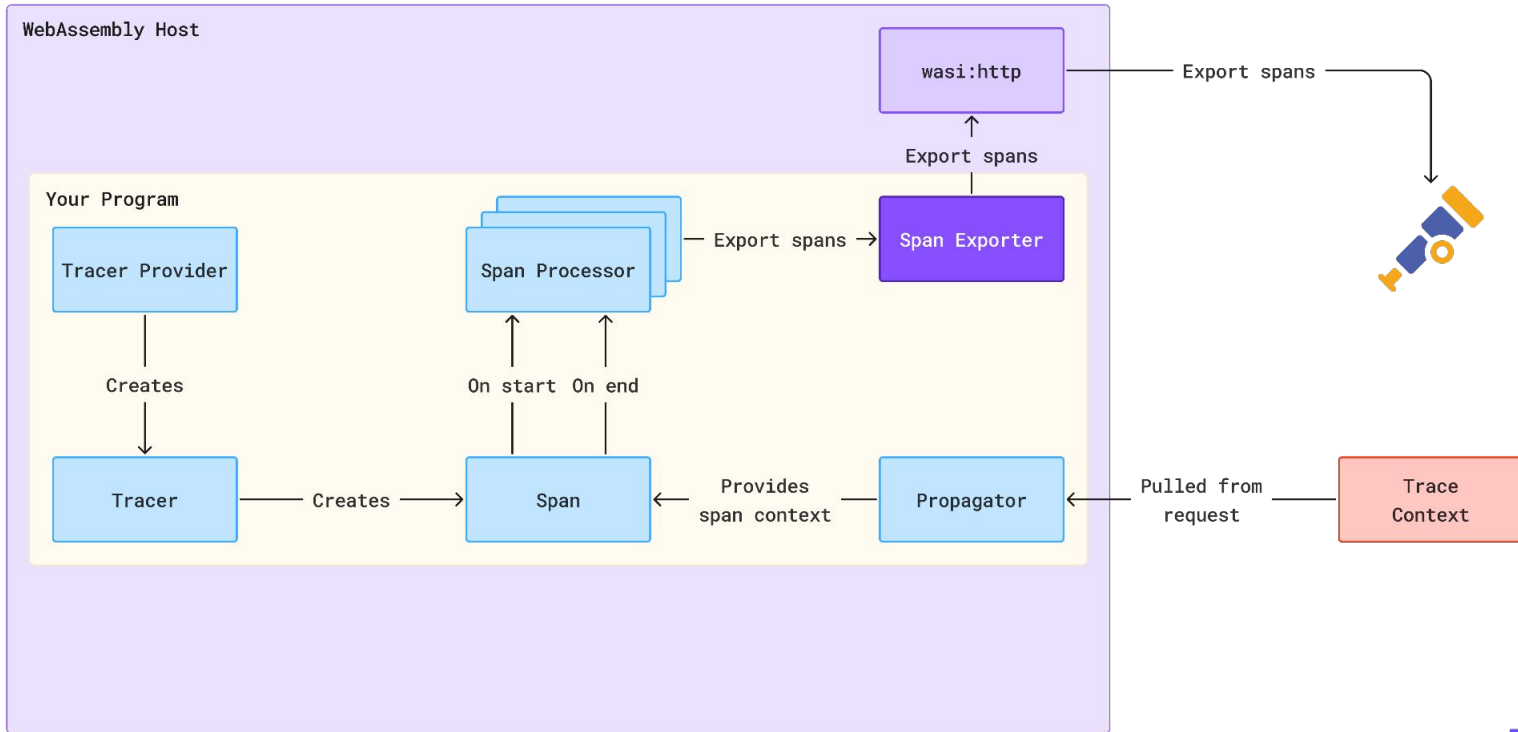


# **Option #1**

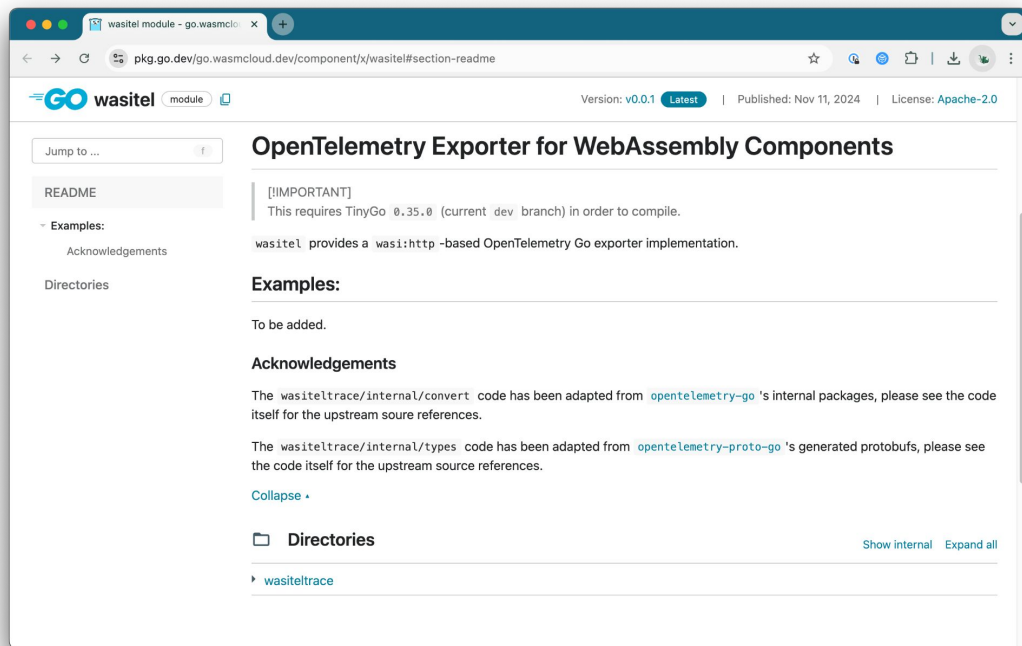
## **Standard HTTP exporter**

### **backed by wasi-http**

# Standard HTTP exporter backed by wasi-http





# Standard HTTP exporter backed by wasi-http





# Standard HTTP **exporter** backed by **wasi-http**

- Using an HTTP exporter is familiar 
- Still have to do some work to back it with wasi-http 

# Standard HTTP exporter backed by wasi-http

- Can't get the parent trace context in non-http components ✗

my\_channel receive

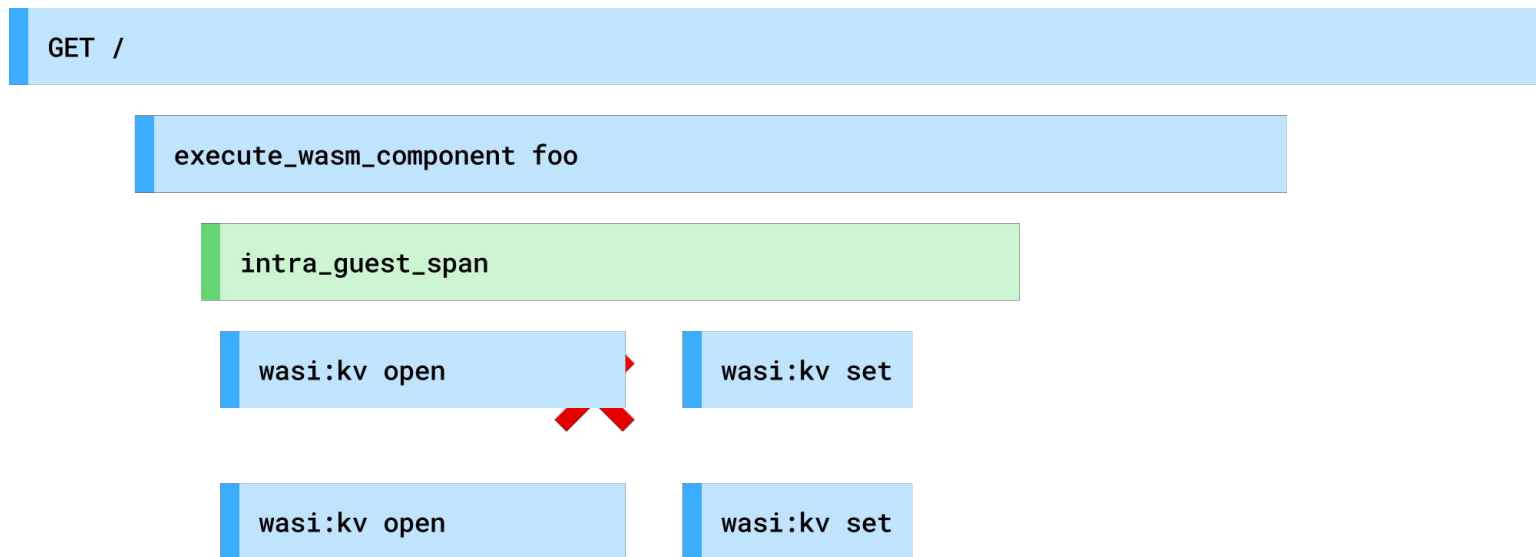
execute\_wasm\_component foo

intra\_guest\_span ✗

intra\_guest\_span

# Standard HTTP exporter backed by wasi-http

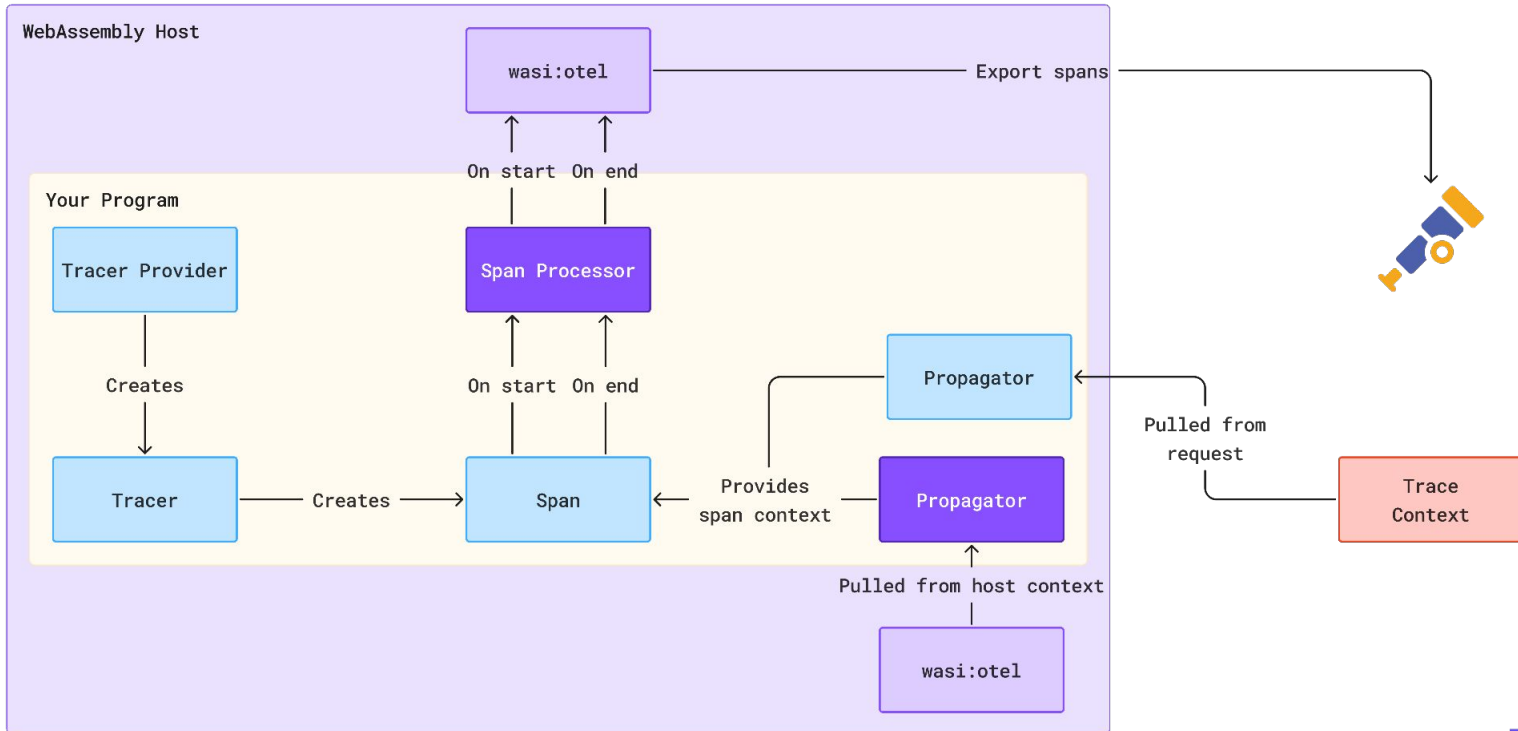
- Can't associate host operations with trace ❌



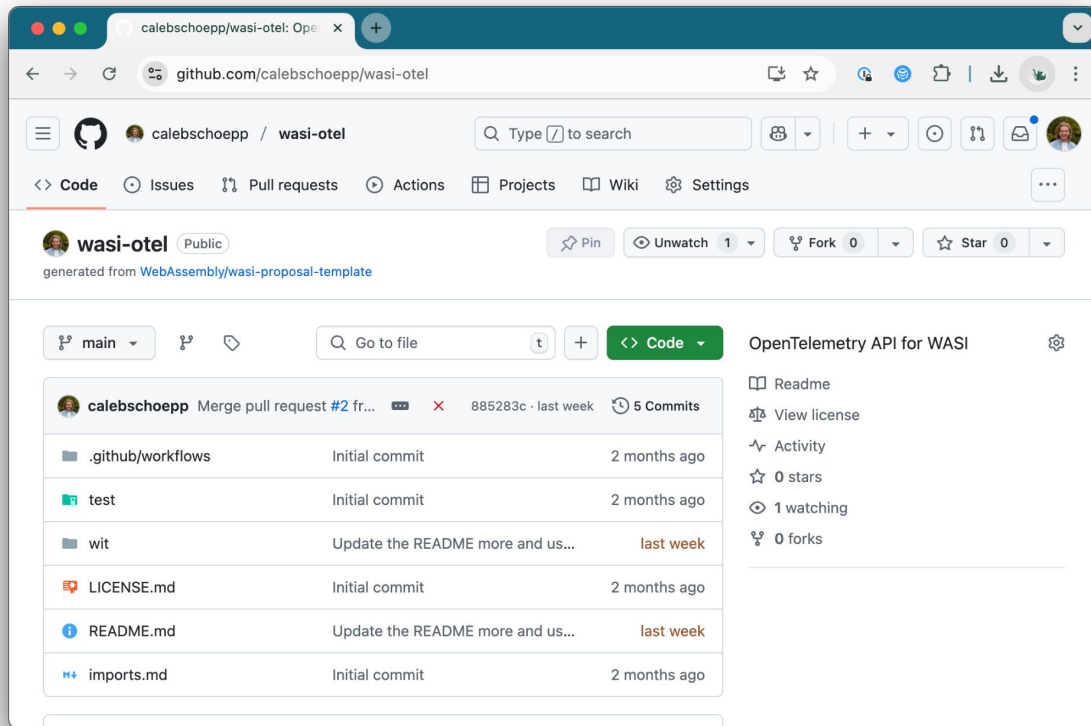
# Option #2

## A special WASI processor backed by wasi-otel

# Special WASI Processor backed by **wasi-otel**



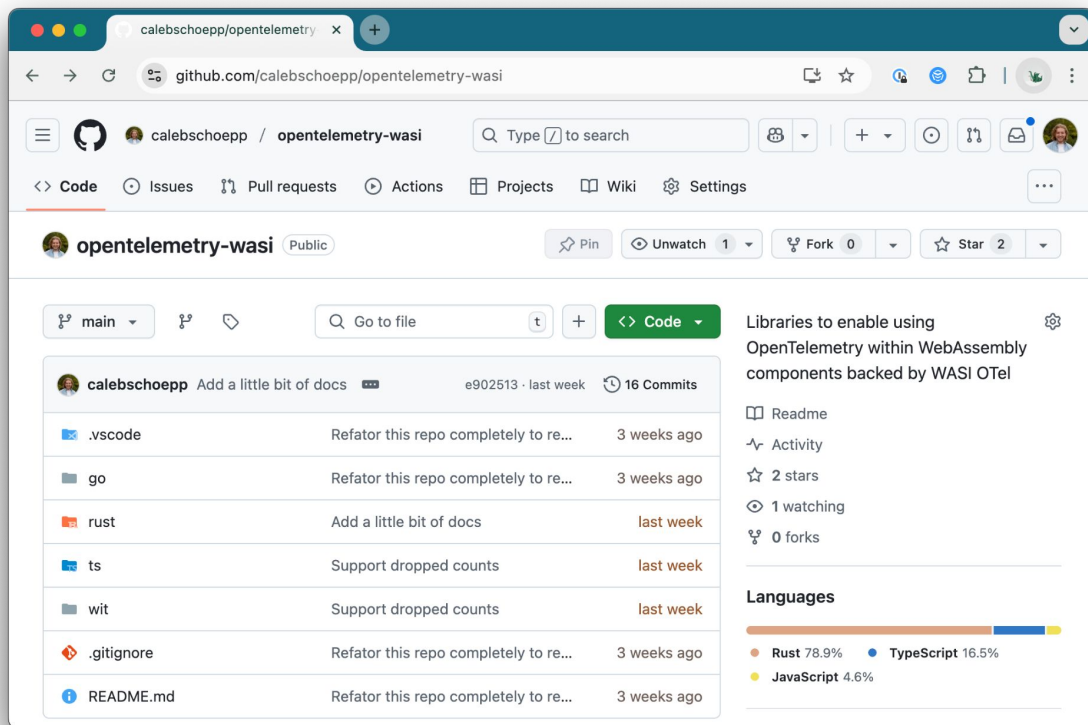
# wasi-otel



# wasi-otel

```
1 interface tracing {
2     /// Called when a span is started.
3     on-start: func(context: span-context);
4
5     /// Called when a span is ended.
6     on-end: func(span: span-data);
7
8     /// Returns the span context of the host.
9     outer-span-context: func() → span-context;
10 }
```

# opentelemetry-wasi



The screenshot shows the GitHub repository page for `calebschoepp/opentelemetry-wasi`. The repository is public and has 2 stars, 0 forks, and 1 watch. The main branch is `main`. The repository description is "Libraries to enable using OpenTelemetry within WebAssembly components backed by WASI OTel".

The repository contains the following files:

File	Description	Last Commit
<code>.vscode</code>	Refator this repo completely to re...	3 weeks ago
<code>go</code>	Refator this repo completely to re...	3 weeks ago
<code>rust</code>	Add a little bit of docs	last week
<code>ts</code>	Support dropped counts	last week
<code>wit</code>	Support dropped counts	last week
<code>.gitignore</code>	Refator this repo completely to re...	3 weeks ago
<code>README.md</code>	Refator this repo completely to re...	3 weeks ago

The repository is categorized by languages:

- Rust 78.9%
- TypeScript 16.5%
- JavaScript 4.6%



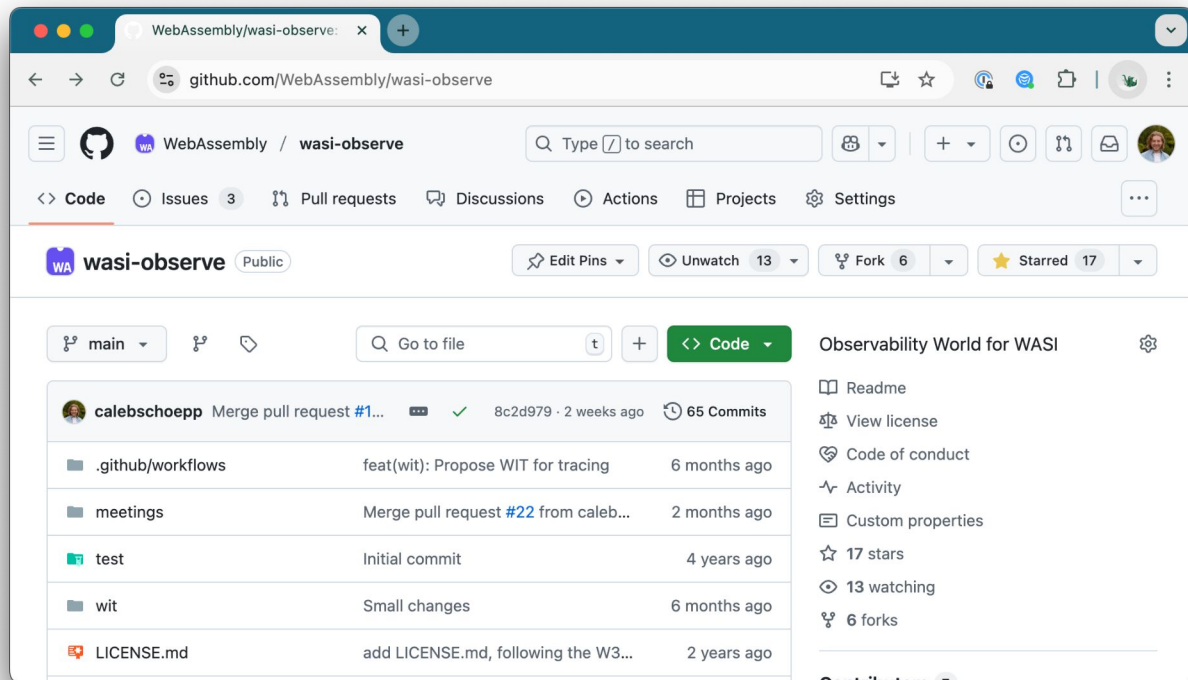




**Demo**



# wasi-otel vs wasi-observe



# Next steps for **wasi-otel**

- Drive wasi-otel through the WASI Subgroup phase process 🚗
- Involve the OpenTelemetry community in the design 💡
- Implement wasi-otel support in more Wasm runtimes than Spin ⚙️
- Implement opentelemetry-wasi SDKs in more languages than Rust 💻
- Design metrics and logs support for wasi-otel 📊
- Play around with it as end user ▶️

# Get involved!

Chat on [Zulip](#)



Play with the [code](#)



Or come talk to me



Thank you to everyone who contributed to this work!  
Especially Shekar Bommas, Victor Adossi, and Lann Martin.

