
KubeCon
CloudNativeCon
North America 2024

KubeCon

CloudNativeCon

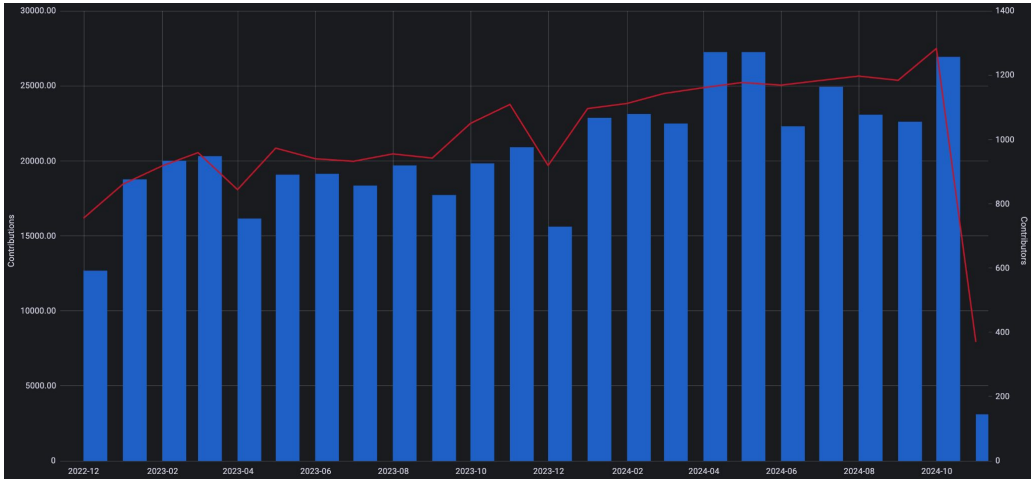
North America 2024

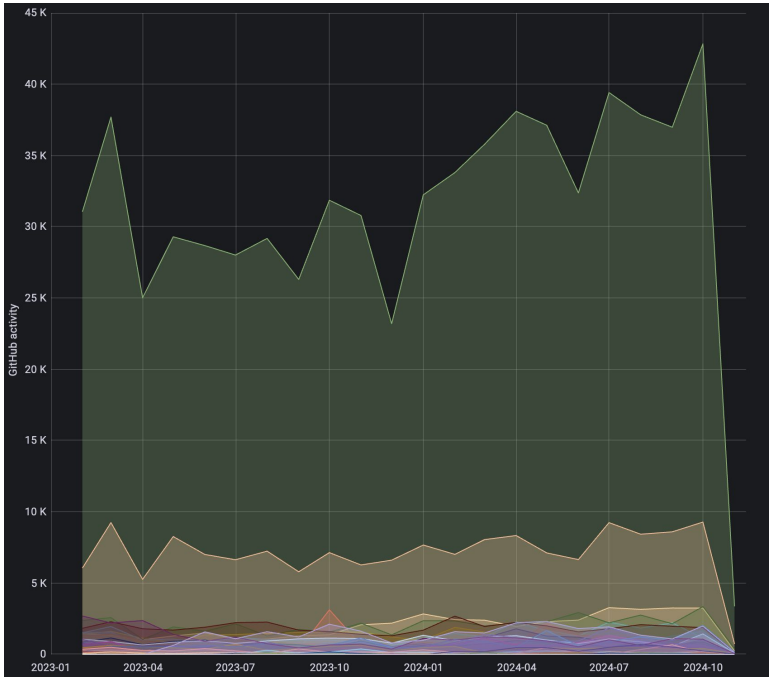
OpenTelemetry Project Update

Alolita Sharma, Apple
Daniel Dyla, Dynatrace
Juraci Paixão Kröhling, Grafana Labs
Morgan McLean, Splunk
Ted Young, ServiceNow

The state of the community is **strong**!

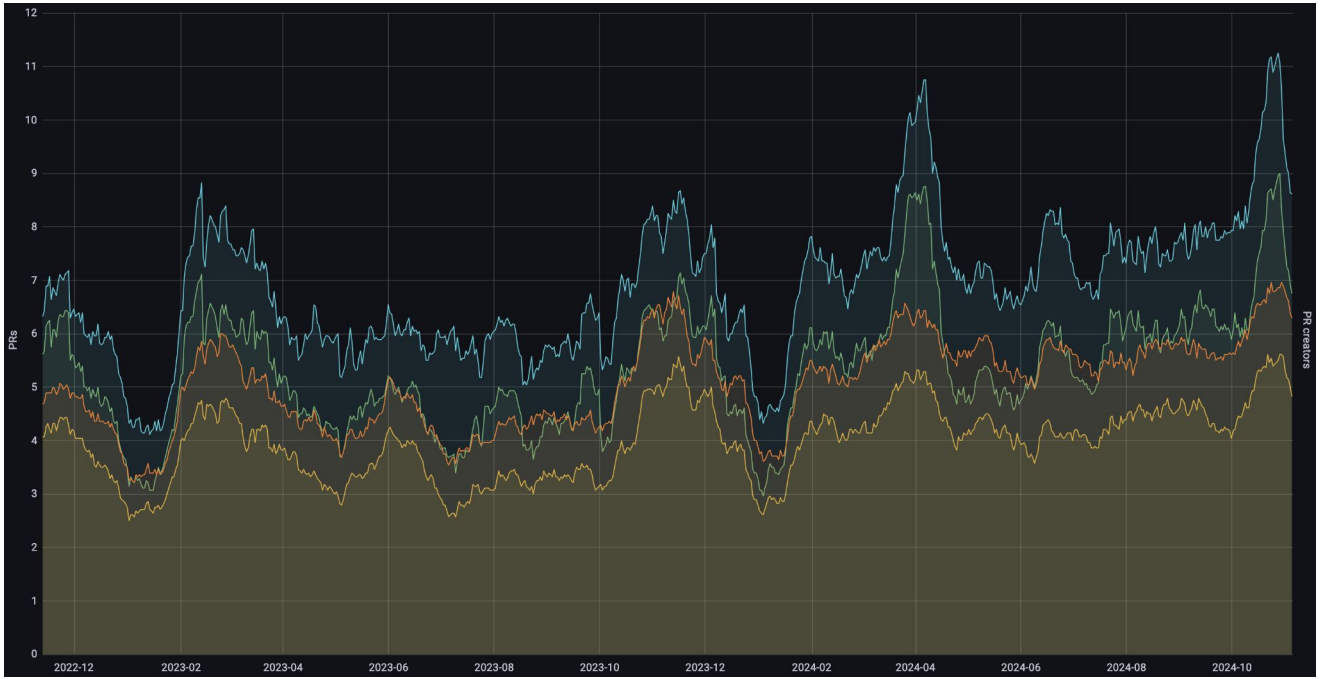
Community Update





Contributions and Activity Remain High

Community Update



We continue to gain new contributors!

Ecosystem Expansion

- New integrations/native OpenTelemetry
  - .NET Aspire 9, Next.JS 15
- Jaeger v2 now runs on the OTel Collector
- Prometheus v3
- Cortex has OTLP support
- OpenTelemetry Certified Associate (OTCA) coming soon!

End User Adoption Is Growing!

Community Stars

A way to recognize community members who have gone above and beyond!

OpenTelemetry Stars

Congratulations to the 2024 OpenTelemetry Community Stars!

- **@serkan-ozal (Serkan Ozal)**
- **@anunarapureddy (Anusha Reddy)**
- **@avillela (Adriana Villela)**
- **@codeboten (Alex Boten)**
- **@emdneto (Emídio Neto)**

Roadmap - Priorities

P0 Continued Investment in OpenTelemetry Artifacts

P1 Logs

P1 Further Stabilizing Semantic Conventions

P2 Client Instrumentation (Browser, Android, Swift)

P2 Profiling

P2 OpenTelemetry Control Plane / File-based Configuration

P2 OpenTelemetry Demo

Source: https://opentelemetry.io/community/roadmap/

Roadmap - Current Projects

Client Instrumentation

Browser support is experimental.

Android support is in beta and needs feedback.

Swift client supports Swift v5.10 and iOS 1.11.0;

- Tracing is stable, Logs is in beta, Metrics in development and needs feedback.

All stable client implementations are gated on other spec work being completed.

Logs

Turns out that OpenTelemetry does need it's own logging API.

Entities and Mutable Resources

Resources are being extended to include Entities – groupings of resources that will allow us to create better topologies.

Resources are also being extended to handle resources that change during the lifespan of the application.

Roadmap - Current Projects

Control Plane

OpAMP support for the Collector is in progress.

But at the moment, Collector v1 is taking priority over this work.

Configuration

The specification for file based configuration for SDK is experimental but ready for use.

Experimental implementations are available in Java, Go, and PHP.

Developer Experience

Improve the SDK installation experience beyond configuration.

Improve the instrumentation experience (API convenience methods).

Seeking feedback from the community!

Semantic conventions update

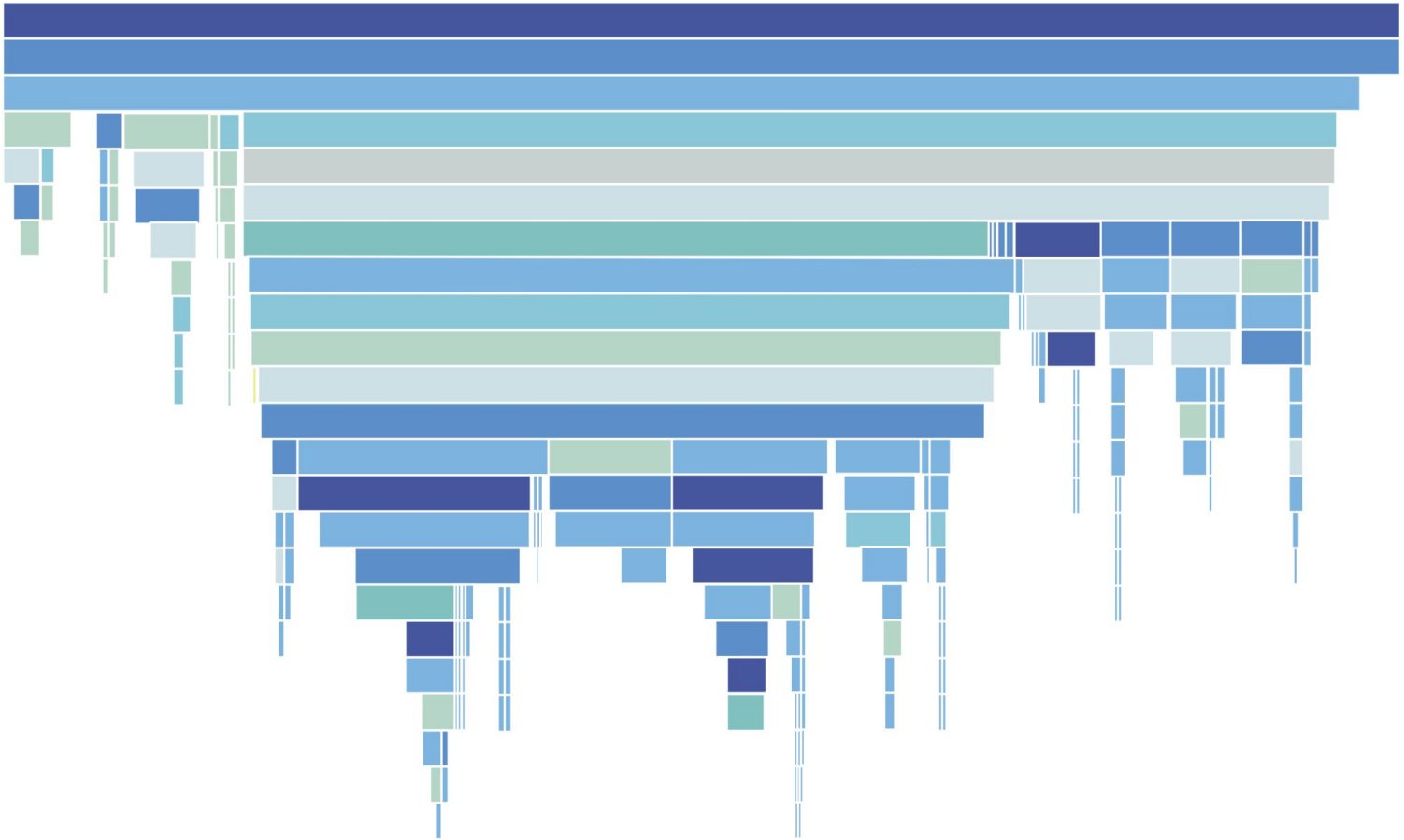
- 186 Merged PRs since KCCNC Paris
- 10 New Components:
  - File
  - Gen-AI
  - Azure
  - User
  - Cloudfoundry
  - Profile
  - Runtime Metrics: Go, NodeJS, V8, .Net
- Nearing Stabilization
  - Database Client (RC)
  - Messaging
- Active SIGs
  - CI-CD
  - Client Instrumentation
  - Feature Flag Evaluation
  - Gen AI
  - Kubernetes
  - Security
  - System Metrics
  - …others

Collector updates

- New distribution to guide v1: otelcol-otlp
- New distribution tailored for Kubernetes workloads: otelcol-k8s
- Progress towards v1
  - Focus on stabilizing base modules and APIs
  - Improving internal telemetry
  - Resiliency improvements
  - Configuration alignment with other parts of OTel
  - Focus on Security (audit, attestations, config changes, …)
  - Preparation for adding profiling support
  - Documentation

Source: https://github.com/orgs/open-telemetry/projects/83

Profiling



Profiling

Introduces Profiling Data Model v2 #239

Edit | Code

Merged jsuereth merged 37 commits into `open-telemetry:main` from `petethepig:profiling-pprofextended` on Feb 23

Conversation 139 | Commits 37 | Checks 1 | Files changed 2 | +1,604 −0

petethepig commented on Nov 1, 2023 • edited — Member

This is second version of the Profiling Data Model OTEP. After we've gotten feedback from the greater OTel community we went back to the drawing board and came up with a new version of the data model. The main difference between the two versions is that the new version is more similar to the original pprof format, which makes it easier to understand and implement. It also has better performance characteristics. We've also incorporated a lot of the feedback we've gotten on the first PR into this OTEP.

Some minor details about the data model are still being discussed and will be flushed out in the future OTEPs. We intend to finalize these details after doing experiments with early versions of working client + collector + backend implementations and getting feedback from the community. The goal of this OTEP is to provide a solid foundation for these experiments.

So far we've done a number of things to validate it:

- we've written a new profiles proto described in this OTEP
- we've documented decisions made along the way in a decision log
- we've done benchmarking to refine the data representation (see Benchmarking section in a collector PR)
- diff between original pprof and the new proto: link

We're seeking feedback and hoping to get this approved.

For (a lot) more details, see:

- OTel Profiling SIG Meeting Notes

👍 35 | 🎉 17 | ❤️ 24 | 🚀 18 | 👀 12

Reviewers

- florianl
- tigrannajaryan
- P403n1x87
- aalexand
- felixge
- beaubelgrave
- Kielek
- jpkrohling
- brianrob
- dashpole
- jsuereth
- yanglong1010
- mtwo
- reyang
- christos68k
- jhalliday
- etep
- Rperry2174
- thomasdullien

Profiling

[Donation Proposal]: Continuous Profiling Agent #1918

Edit | New issue

Closed | AlexanderWert opened this issue on Jan 30 · 25 comments

AlexanderWert commented on Jan 30 • edited | Member

Description

Elastic would like to offer the donation of the Elastic profiling agent to the OpenTelemetry project.

The Elastic profiling agent is an eBPF-based mature multi-runtime/multi-language CPU profiler. It enables fleet-wide and system-wide continuous profiling without the need for any application instrumentation or even application restart. It's currently, to the best of our knowledge, the only existing continuous profiler with no instrumentation/process restart for a broad range of real-world languages. The agent has been used in real large-scale customer production environments since August 2021.

Some of the core features and strengths of the agent are:

- Very low CPU and memory overhead (1% CPU and 250MB memory are our upper limits in testing and the agent typically manages to stay way below that)
- Support for native C/C++ executables without the need for DWARF debug information (by leveraging `.eh_frame` data as described in US11604718B1)
- Support profiling of system libraries **without frame pointers** and **without debug symbols on the host**.
- Support for mixed stacktraces between runtimes - stacktraces go from Kernel space through unmodified system libraries all the way into high-level languages.
- Support for native code (C/C++, Rust, Zig, Go, etc. without debug symbols on host)
- Support for a broad set of HLLs (Hotspot JVM, Python, Ruby, PHP, Node.JS, V8, Perl), .NET is in preparation.
- 100% non-intrusive: there's no need to load agents or libraries into the processes that are being profiled.
- No need for any reconfiguration, instrumentation or restarts of HLL interpreters and VMs: the agent supports unwinding each of the supported languages in the default configuration.
- ARM64 support for all unwinders except NodeJS.
- Support for native `inline frames`, which provide insights into compiler optimizations and offer a higher precision of function call chains.

Assignees
No one—assign yourself

Labels
None yet

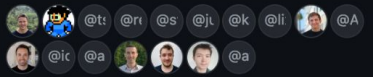
Projects
None yet

Milestone
No milestone

Development
Create a branch for this issue or link a pull request.

Notifications | Customize
Unsubscribe
You're receiving notifications because you're watching this repository.

17 participants



Lock conversation

Profiling - What's Complete / In Progress

Protocol

OTLP now treats profiles as a dedicated signal!

Data model has been merged in; still marked as experimental and changes may come.

Languages

SDK and automatic instrumentation support for languages with runtimes or dedicated profiling APIs.

These integrations are lightweight, simple, and provide the best possible data.

System Agent

eBPF-based agent that can capture profiles from any application.

Especially useful for profiling C++ and Rust applications, and third-party software.

Easy way to get started with fleetwide profiling.

Profiling - What's Next

- Further development of the system profiling agent
- More language implementations
- Profiling signal 1.0 in OTLP
- More correlations with other signals
- More triggers

Graduation progress

Overall Status

- We’re in the queue!
- Once the CNCF TOC picks up our application, more stuff will happen

Specific Notes

- Our security audit went extremely well – thanks to those who helped!
- There will probably be some more work around ensuring we’re following best practices across all SIGs; Stay Tuned.

Join our community!

OpenTelemetry needs you.

We need help in

- Client Instrumentation
- OpAMP in the Collector and SDKs
- OpenTelemetry Demo
- Semantic Conventions

We need more approvers, triagers for OTel language libraries, Collector, semantic conventions, security, CI/CD.

If you are a contributor already, please reach out to the SIG maintainers.

Ask us anything!

Ask OTel experts about any topic!