

Diana Todea - DX Engineer @VictoriaMetrics

# De cero a desarrollador: mi viaje de un año de serendipia con OpenTelemetry



Cloud Native  
LATAM *Summit*

Nov 27, 4:00 PM – Nov 29, 4:00 PM

GMT-5 Evento Virtual



Colaboradora de OTel

---

Organizadora Cloud Native Days Romania

---

Co-lead CNCF Merge-Forward Neurodiversity

---



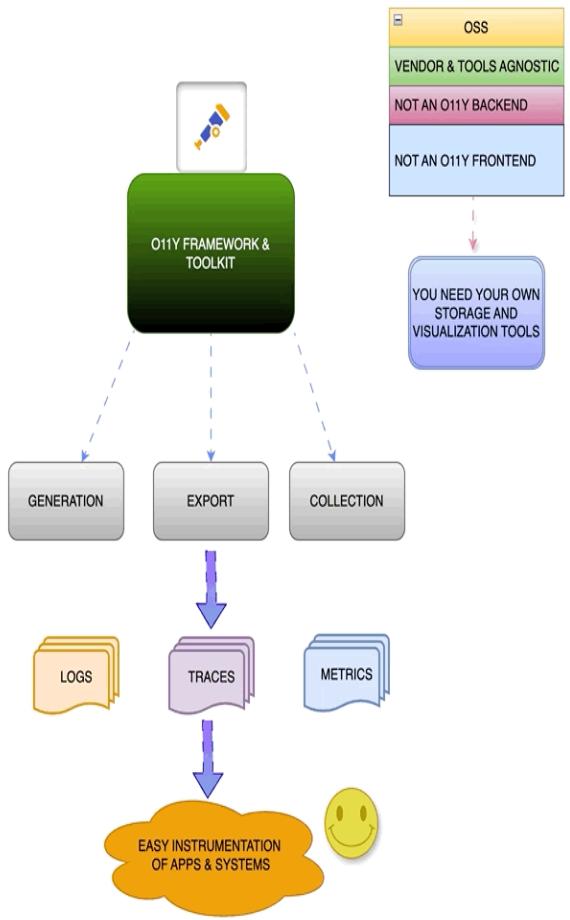
Diana Todea  
DX@VictoriaMetrics | Observability  
SME | OpenTelemetry docs localizatio...



# La telemetría tiene que ser

- fácil
- universal
- proveedor neutral
- débilmente acoplado (dependencias minimas)
- incorporada

La misión de OpenTelemetry: permitir una observabilidad efectiva haciendo que la telemetría portátil y de alta calidad sea universal



# Tener la curiosidad para empezar

La comunidad Linux Foundation buscaba especialistas en observabilidad para crear el primer examen y la primera certificación de OpenTelemetry.



CERTIFICATION

## OpenTelemetry Certified Associate (OTCA)

As cloud native systems grow more complex, the demand for professionals who can leverage telemetry data is growing rapidly. Open new career paths – prove your expertise in OpenTelemetry – the industry standard for tracing, metrics & logs.

OTCA includes:

- ✓ 12-months to schedule & take the exam
- ✓ Two exam attempts

## I played with OTEL and I liked it

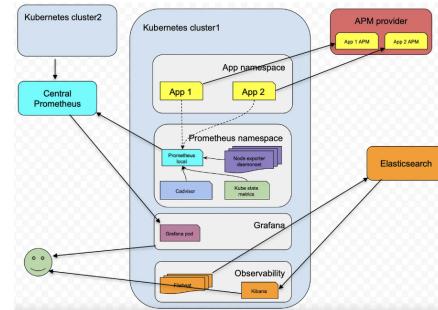
 Diana Todea · 4 min read · Jan 7, 2025

As mentioned in my previous [article](#), in 2024 I had the opportunity to work on an observability project where I wrote a proof of concept on [OpenTelemetry](#). As an SRE I saw the window to experiment with one of the [most popular](#) CNCF projects on distributed tracing and provide an open source alternative to monitor the performance of my client's applications. I had the chance to give a more in depth talk on this topic at [KCD Accra Ghana](#) as well as at [OSMC 2024](#). You can access my presentations directly on my [Github repo](#).

### Architecture

This particular observability architecture was mainly composed of [Prometheus metrics](#), [Victoria Metrics](#), [Grafana](#) for visualizations and alerting, [ELK](#) for logging and a commercial APM vendor. A fragmented observability architecture leads to a confusing developer and user experience, working in observability for almost 5 years now it's easy to say that most people prefer having a unified monitoring pane of glass, which is easier to onboard and manage.

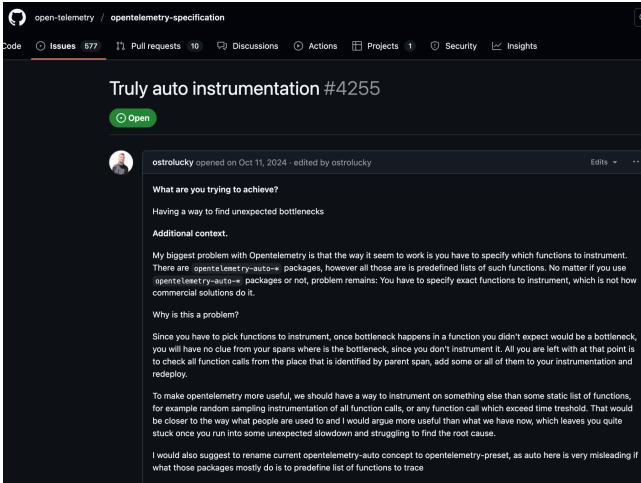


Observability architecture explaining how the K8s clusters ship metrics, traces and logs



# Proporcionar comentarios de los desarrolladores

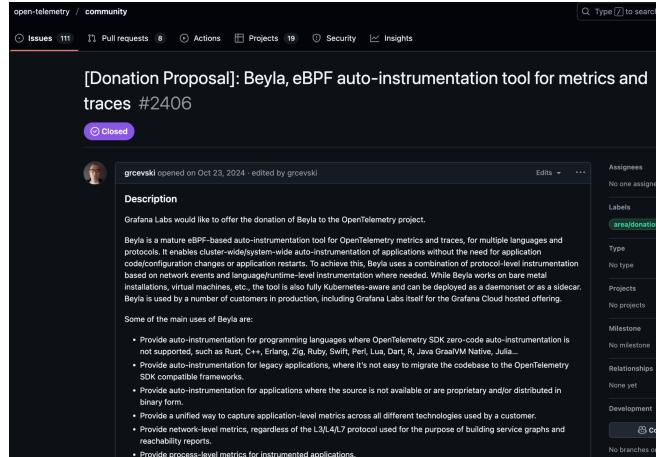
<https://github.com/open-telemetry/opentelemetry-specification/issues/4255>



This screenshot shows a GitHub issue page for a pull request. The title is "Truly auto instrumentation #4255". The issue is labeled as "Open". The description discusses the problem with OpenTelemetry's auto-instrumentation, noting that it requires specifying functions to instrument and that there are "opentelemetry-auto" packages. It suggests a way to find unexpected bottlenecks by checking function calls from parent spans. The author also suggests renaming the "opentelemetry-auto" concept to "opentelemetry-preset".

What are you trying to achieve?  
Having a way to find unexpected bottlenecks  
Additional context.  
My biggest problem with OpenTelemetry is that the way it seems to work is you have to specify which functions to instrument. There are `opentelemetry-auto` packages, however all those are predefined lists of such functions. No matter if you use `opentelemetry-auto` packages or not, problem remains: You have to specify exact functions to instrument, which is not how commercial solutions do it.  
Why is this a problem?  
Since you have to pick functions to instrument, once bottleneck happens in a function you didn't expect would be a bottleneck, you will have no clue from your spans where is the bottleneck, since you don't instrument it. All you are left with at that point is to check all function calls from the place that is identified by parent span, add some or all of them to your instrumentation and redeploy.  
To make OpenTelemetry more useful, we should have a way to instrument on something else than some static list of functions, for example random sampling instrumentation of all function calls, or any function call which exceed time threshold. That would be closer to the way what people are used to and it would argue more useful than what we have now, which leaves you quite stuck once you run into some unexpected slowdown and struggling to find the root cause.  
I would also suggest to rename current `opentelemetry-auto` concept to `opentelemetry-preset`, as `auto` here is very misleading if what those packages mostly do is to predefine list of functions to trace

<https://github.com/open-telemetry/community/issues/2406>



This screenshot shows a GitHub issue page for a donation proposal. The title is "[Donation Proposal]: Belya, eBPF auto-instrumentation tool for metrics and traces #2406". The issue is labeled as "Closed". The description explains that Grafana Labs would like to offer the donation of Belya to the OpenTelemetry project. Belya is described as a mature eBPF-based auto-instrumentation tool for OpenTelemetry metrics and traces, supporting multiple languages and protocols. It enables cluster-wide/system-wide auto-instrumentation of applications without the need for application code/configuration changes or application restarts. The tool uses a combination of protocol-level instrumentation based on network events and language/runtime-level instrumentation where needed. Belya works on bare metal installations, virtual machines, etc., and can be deployed as a daemonset or as a sidecar. It is used by a number of customers in production, including Grafana Labs itself for the Grafana Cloud hosted offering.

Some of the main uses of Belya are:

- Provide auto-instrumentation for programming languages where OpenTelemetry SDK zero-code auto-instrumentation is not supported, such as Rust, C++, Erlang, Zig, Ruby, Swift, Perl, Lua, Dart, R, Java GraalVM Native, Julia...
- Provide auto-instrumentation for legacy applications, where it's not easy to migrate the codebase to the OpenTelemetry SDK compatible frameworks.
- Provide auto-instrumentation for applications where the source is not available or are proprietary and/or distributed in binary form.
- Provide a unified way to capture application-level metrics across all different technologies used by a customer.
- Provide network-level metrics, regardless of the L3/L4/L7 protocol used for the purpose of building service graphs and reachability reports.
- Provide process-level metrics for instrumented applications.

# Colaborador individual en Cloud Native

No depende  
de la  
organización

Contribuciones  
CON/SIN  
Código

Múltiples  
proyectos  
OSS



# Apostrar las comunidades locales

#otel-docs-localization  
#otel-localization-es  
#cloud-native-valencia  
#cnd-romania



# | Localización

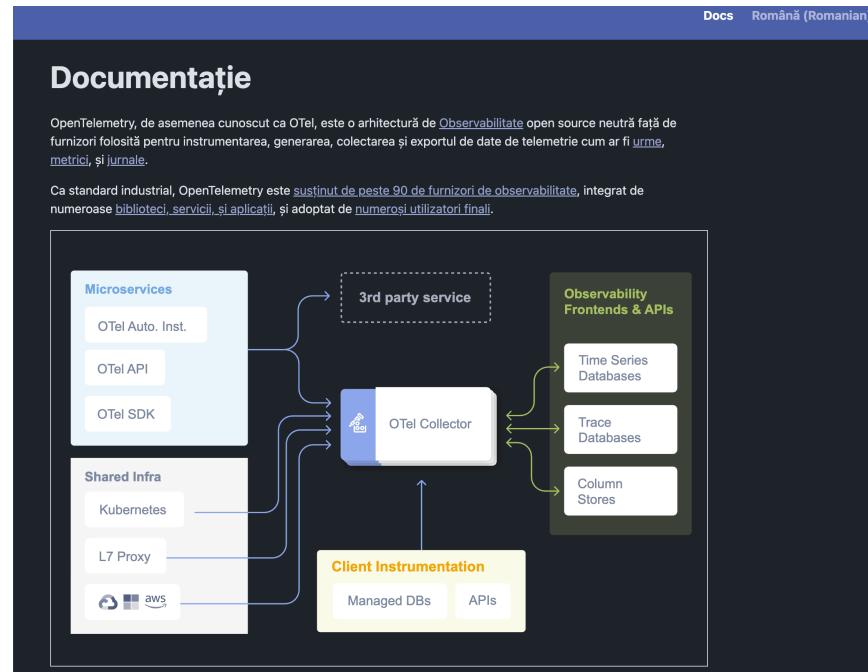
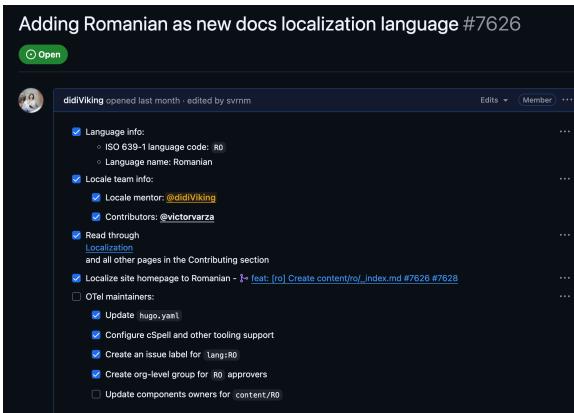


= GRAN IMPACTO



# Potenciar la localización de OTel

Presentar un problema, crear una primera solicitud  
Construir una comunidad en torno a la localización  
Nuevo SIG: #otel-localization-ro



# Compartir con audiencias amplias

## Contribuciones a la experiencia del desarrollador

### [Online] Women in Cloud Era @ Cloud Native Kuala Lumpur Meetup - September 2025

DIANA TODEA  
Developer Advocate at Aircall  
Diana is a Developer Advocate at Aircall. She has worked as a Senior Site Reliability Engineer focused on Observability. She is an active contributor to the OpenTelemetry open source project and supports women in tech.

From zero to developer: my one year serendipity journey with OpenTelemetry

Sessions

Last Wednesday at 1:50 PM - 15 min - Room 4  
From zero to developer: my one year serendipity journey with OpenTelemetry  
Diana Todea

FROM ZERO TO DEVELOPER: MY ONE YEAR SERENDIPITY JOURNEY WITH OPentelemetry

Becoming a contributor to an open-source project is a transformative step in any developer's career. This session explores the journey from first-time contributor to active developer, covering best practices for navigating project communities, understanding codebases, and making meaningful contributions. Learn strategies for selecting the right project, mastering collaboration tools, and embracing the culture of open-source development. The audience will be inspired by one year journey with the open source project OpenTelemetry, how I have built a proof of concept for it and achieved developer status for this project. By the end of this talk, the public will gain insights into the tools to become a better developer and how to build more engagement with the community.

SPEAKER

Diana Todea  
Aircall

Cloud Native Days Romania

The Kubernetes & Cloud Native community will gather at the Radisson Blu in Bucharest, Romania, for one day to learn, share, and connect. This is a great opportunity for Cloud Native Days Romania is aimed at developers, system administrators, and professionals with an interest in cloud-native technologies.

### From zero to developer: my one year serendipity journey with OpenTelemetry

Last Wednesday at 1:50 PM - 15 min - Room 4  
From zero to developer: my one year serendipity journey with OpenTelemetry  
Diana Todea

DEVOPTSDAYS AMSTERDAM

Becoming a contributor to an open-source project is a transformative step in any developer's career. This session explores the journey from first-time contributor to active developer, covering best practices for navigating project communities, understanding codebases, and making meaningful contributions. Learn strategies for selecting the right project, mastering collaboration tools, and embracing the culture of open-source development. The audience will be inspired by one year journey with the open source project OpenTelemetry and how I have built a proof of concept for it and achieved developer status for this project. By the end of this talk, the public will gain insights into the tools to become a better developer and how to build more engagement with the community.

Schedule Sessions Speakers

Virginia Diana Todea  
Diana is a Developer Advocate and a Senior Site Reliability Engineer. She is passionate about observability, machine learning and open source projects. She is an OpenTelemetry contributor and loves sharing her knowledge with the community by participating to tech conferences and writing articles.

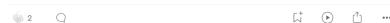
Open ObservabilityCon + Tel Community Day

June 26  
Denver, Colorado  
#OpenOtelCon #TelDay

# Educar a través de contenidos escritos

## Cloud Native Days Romania 2025-tales from the field

Diana Todea · 4 min read · May 8, 2025



I had the honor to participate at Cloud Native Days Romania's 2nd edition, hosted in Bucharest, Romania, on the 5th and 6th of May and honestly I was amazed. Meeting with a vibrant open source community, talented organizers and an overall curious audience that made me feel that I belonged, what a dream come true.



Formerly known as Kubernetes Community Days Romania, CND Romania has reinvented itself this year adding an extra day for workshops. From CI/CD pipelines with Dagger, to GitOps with FluxOperator, Ray and Kubernetes for ML projects, Istio Service Mesh and many other exciting topics, no wonder the workshop day reached its full capacity.

## Why feedback matters for open source

Diana Todea · 3 min read · Jan 7, 2025



My 2024 has been a year where I finally immersed myself in the open source realm. It has been on my mind for a while to start contributing to an open source project, however I wanted to pick the right project that resonated with my current on-going work.

Putting doubts aside, I was really impressed with OpenTelemetry's popularity and the fact that so many organizations already adopted it into their production environments. When I started drafting my GenerativeAI talk, I interacted with OpenTelemetry, APM and Elastic's Observability stack. Seeing how everything blended together was a great experience, so when the chance opened up to contribute to OpenTelemetry project, I jumped right into it.

OTCA certification from Linux Foundation has been released on the market just very recently. It is the first exam and certification covering the OpenTelemetry topic at its foundational level. I got the chance to contribute as a developer to OTCA exam, writing questions, double checking the documentation sources and putting myself in the shoes of a person who wants to get certified on OpenTelemetry. For me it has been the perfect opportunity to get closer to an open source project I admired from afar and interacting with the open source community.

OpenTelemetry project has a vast documentation with a tendency to get complicated really fast. Some parts are still developing and definitely are in need of developers' constant feedback. It's hard to perceive OpenTelemetry's level of difficulty from afar, you actually have to put your working gloves and deploy it. Later on in 2024, I had the chance to get my hands "dirty" with such an observability project and build the proof of concept for OpenTelemetry with several SDKs, documenting my experience in this article.

## OpenObservabilitySummit and OTEL Community Day 2025 recap

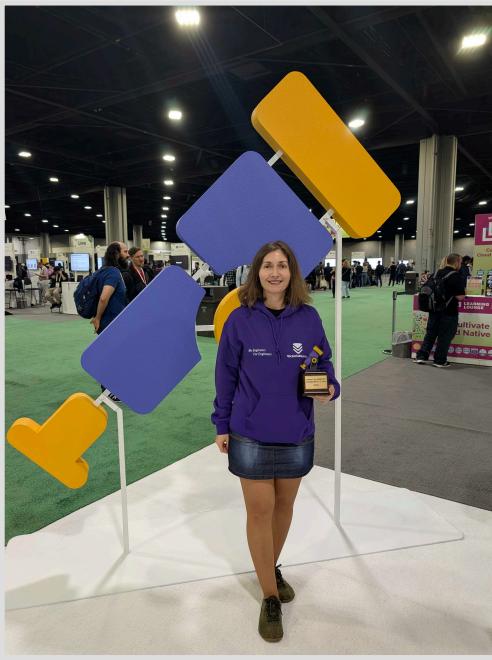
Diana Todea · 6 min read · Jun 30, 2025



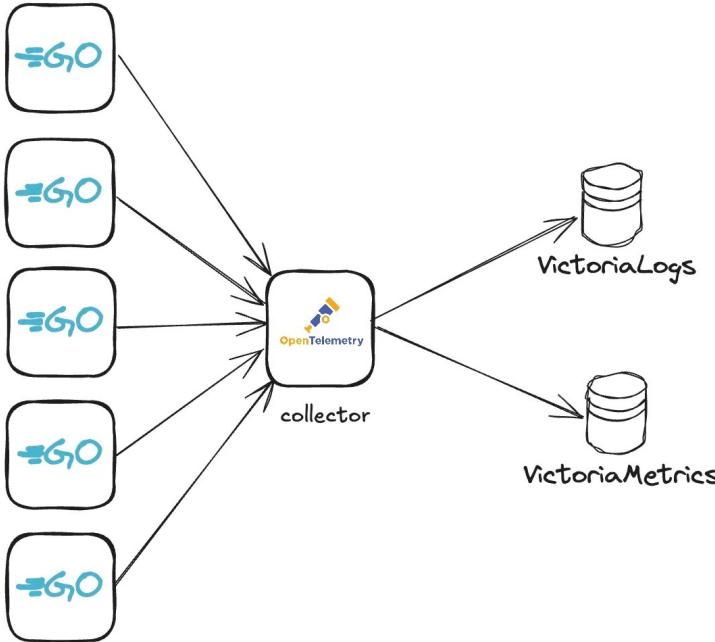
Last Thursday in Denver, US, OTEL Community Day event took place as part of the OpenObservabilitySummit and it was a blast. I had the chance to participate with a lightning talk and learn about all the latest releases within the Open Observability realm.

Below, you can find the summary based on the sessions I participated at, the official videos will be released on the CNCF YouTube channel in a couple of weeks. The OTEL Community Day had two parallel tracks and some of the presentations are already uploaded on the schedule page.

# KubeCon Atlanta 2025: la encuentra con la comunidad OTEL



# VictoriaMetrics + OpenTelemetry



VictoriaMetrics and VictoriaLogs tienen soporte nativo para la ingestión de métricas y registros en el formato OTEL.

Necesitas:

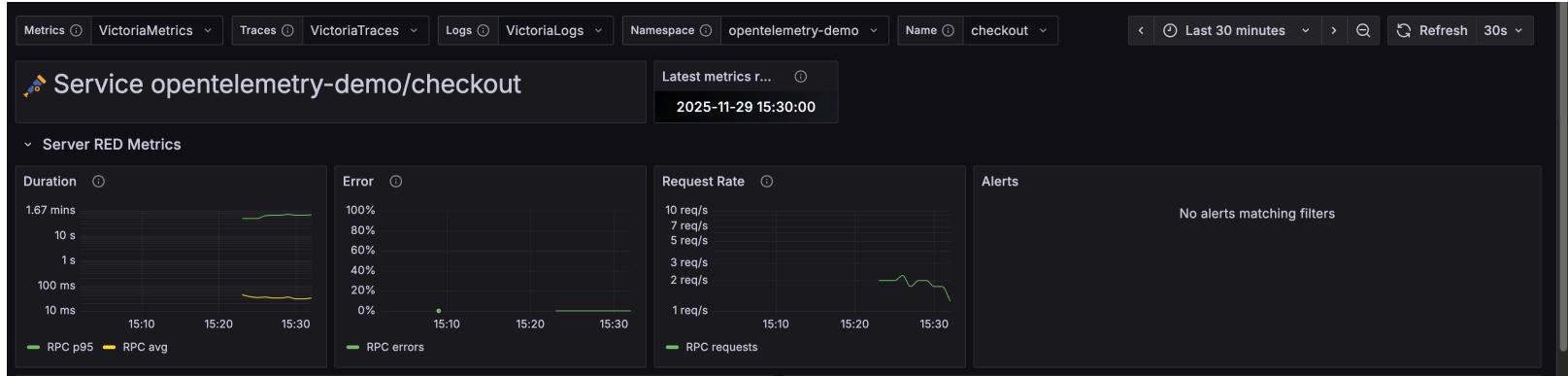
k8s cluster, kubectl, helm



# VictoriaMetrics OTel Demo fork

★<https://github.com/VictoriaMetrics-Community/opentelemetry-demo>

The screenshot shows the GitHub repository page for 'opentelemetry-demo' under the 'VictoriaMetrics-Community' organization. The repository has a dark theme. At the top, there's a navigation bar with links for 'README', 'Contributing', and 'License'. Below the navigation, there's a header with the repository name 'OpenTelemetry Demo' and a small icon of a telescope. The repository has a green badge for Slack (@cncf/otel/demo), a purple badge for release v2.1.3, and a blue badge for commits since 2.1.3 (99). It also has a blue badge for docker pulls (639k), a red badge for License (Apache 2.0), and a green badge for Integration Tests (passing). There's a blue badge for Artifact Hub (opentelemetry-demo) and a yellow badge for openssf best practices (in progress 88%). A section titled 'Version for VictoriaMetrics Stack' explains that it's a fork of the OpenTelemetry demo using the VictoriaMetrics Stack, listing three differences: VictoriaMetrics instead of Prometheus, VictoriaLogs instead of OpenSearch, and VictoriaTraces instead of Jaeger. Another section, 'Welcome to the OpenTelemetry Astronomy Shop Demo', describes the repository as a microservice-based distributed system for demonstrating OpenTelemetry. It lists three goals: providing a realistic example of a distributed system, building a base for vendors and tooling authors, and creating a living example for contributors. It also mentions ongoing development and future language support.



Logs

```
|> 2025-11-29 15:31:36.619 Successful to write message. offset: 0, duration: 357.167µs
|> 2025-11-29 15:31:36.618 sending to postProcessor
|> 2025-11-29 15:31:36.618 order confirmation email sent to "reed@example.com"
|> 2025-11-29 15:31:36.601 order placed
|> 2025-11-29 15:31:36.591 payment went through
|> 2025-11-29 15:31:36.567 [PlaceOrder]
|> 2025-11-29 15:31:20.221 Successful to write message. offset: 0, duration: 345.375µs
|> 2025-11-29 15:31:20.221 sending to postProcessor
|> 2025-11-29 15:31:20.221 order confirmation email sent to "moore@example.com"
|> 2025-11-29 15:31:20.201 order placed
|> 2025-11-29 15:31:20.186 payment went through
|> 2025-11-29 15:31:20.166 [PlaceOrder]
|> 2025-11-29 15:30:50.693 Successful to write message. offset: 0, duration: 328.5µs
|> 2025-11-29 15:30:50.692 sending to postProcessor
|> 2025-11-29 15:30:50.692 order confirmation email sent to "bill@example.com"
|> 2025-11-29 15:30:50.682 order placed
|> 2025-11-29 15:30:50.679 payment went through
|> 2025-11-29 15:30:50.647 [PlaceOrder]
|> 2025-11-29 15:28:36.543 Successful to write message. offset: 0, duration: 84.625µs
|> 2025-11-29 15:28:36.542 sending to postProcessor
|> 2025-11-29 15:28:36.542 order confirmation email sent to "jeff@example.com"
|> 2025-11-29 15:28:36.540 order placed
|> 2025-11-29 15:28:36.538 payment went through
|> 2025-11-29 15:28:36.530 [PlaceOrder]
|> 2025-11-29 15:28:34.424 Successful to write message. offset: 0, duration: 403.458µs
|> 2025-11-29 15:28:34.423 sending to postProcessor
|> 2025-11-29 15:28:34.423 order confirmation email sent to "michael@example.com"
```

## Traces

Outline

VictoriaTraces

Split

Add

Last 1 hour



Run query

load-generator: user\_checkout\_multi POST 200

200

Trace ID 4ba1aeecdacd4f58feb1d588653e2385



Start time 2025-11-29 15:31:36.528 (a minute ago)

Duration 99.76ms

Services 11

Route /oteldemo.CartService/GetCart

Feedback

Share

62 spans

Prev Next

&gt; Span Filters

## Service &amp; Operation

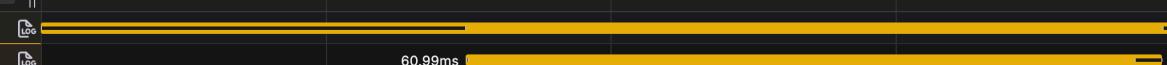
0μs

24.94ms

49.88ms

74.82ms

99.76ms



↳ load-generator user\_checkout\_multi (99.76ms)

↳ POST (60.99ms)

POST

Logs for this span

Span attributes: error unset http.method POST http.status\_code 200 http.url http://frontend-proxy:8080/api/checkout otel.scope.name opentelemetry.instrumentation.requests otel.scope.version 0.59b0 span.kind client

Resource attributes: host.name docker-desktop os.type linux service.namespace opentelemetry-demo service.version 2.1.3 telemetry.sdk.language python telemetry.sdk.name opentelemetry telemetry.sdk.version 1.38.0

↳ frontend-proxy ingress (58.28ms)

58.28ms

↳ router frontend egress (58.19ms)

58.19ms

↳ frontend POST (58.4ms)

58.4ms

↳ POST (56.13ms)

56.13ms

↳ executing api route (pages) /api/checkout (55.71ms)

55.71ms

↳ grpc.oteldemo.CheckoutService/PlaceOrder (53.32ms)

53.32ms

↳ checkout oteldemo.CheckoutService/PlaceOrder (51.75ms)

51.75ms

↳ prepareOrderItemsAndShippingQuoteFromCart (15.38ms)

15.38ms

↳ oteldemo.CartService/GetCart (1.36ms)

1.36ms

↳ cart POST /oteldemo.CartService/GetCart (407μs)

407μs

HGET (289μs)

289μs

↳ checkout oteldemo.ProductCatalogService/GetProduct (411μs)

411μs

↳ product-catalog oteldemo.ProductCatalogService/GetProduct (88μs)

88μs

↳ checkout oteldemo.CurrencyService/Convert (2.6ms)

2.6ms

currency Currency/Convert (925μs)

925μs

↳ checkout oteldemo.ProductCatalogService/GetProduct (238μs)

238μs

↳ product-catalog oteldemo.ProductCatalogService/GetProduct (47μs)

47μs

↳ checkout oteldemo.CurrencyService/Convert (685μs)

685μs

# Tener en cuenta ☺ OTEL Weaver y Injector

[README](#) [Code of conduct](#) [Contributing](#) [Apache-2.0 license](#) [Security](#)

## OpenTelemetry Weaver



**Observability by Design**  
*Treat your telemetry like a public API*

CI [passing](#) [codecov](#) 78% [Security Audit](#) [passing](#) License Apache 2.0 Slack #otel-weaver

OpenTelemetry Weaver helps teams build observability by design, enabling consistent, type-safe, and automated telemetry through semantic conventions. With Weaver, you can define, validate, and evolve your telemetry schemas, ensuring reliability and clarity across your systems.

### What is Observability by Design?

Have you ever experienced:

- Broken alerts after a deployment because metric names changed?
- Complex, hard-to-understand queries due to inconsistent naming?
- Teams struggling to interpret unclear or undocumented signals?
- Missing critical instrumentation discovered only in production?

**Observability by Design** solves these problems by treating your observability signals (metrics, traces, logs) as a first-class public API that requires the same quality standards as your code.

An introduction to Weaver and Observability by Design is presented in the official blog post: [Observability by Design: Unlocking Consistency with OpenTelemetry Weaver](#)

[README](#) [Code of conduct](#) [Contributing](#) [Apache-2.0 license](#) [Security](#)

## OpenTelemetry Injector

The OpenTelemetry injector is a shared library (written in [Zig](#)) that is intended to be used via the environment variable `LD_PRELOAD`, the `/etc/ld.so.preload` file, or similar mechanisms to inject environment variables into processes at startup.

It serves two main purposes:

- Inject an OpenTelemetry Auto Instrumentation agent into the process to capture and report distributed traces and metrics to the [OpenTelemetry Collector](#) for supported runtimes.
- Set resource attributes automatically, (for example Kubernetes related resource attributes and service related resource attributes in environments where this is applicable).

The injector can be used to enable automatic zero-touch instrumentation of processes. For this to work, the injector binary needs to be bundled together with the OpenTelemetry auto-instrumentation agents for the target runtimes.

Official RPM and DEB packages that contain the injector as well as the auto-instrumentation agents are available, and can be downloaded from the [releases page](#). The OpenTelemetry injector Debian/RPM packages install the OpenTelemetry auto-instrumentation agents, the `libotelinject.so` shared object library, and a default configuration file to automatically instrument applications and services to capture and report distributed traces and metrics to the [OpenTelemetry Collector](#).

The `opentelemetry-injector` deb/rpm package installs and supports configuration of the following Auto Instrumentation agents:

- [Java](#)
- [Node.js](#)
- [.NET](#)

# ¡Construye tu propio viaje!

- ¡Tu motivación es tu superpoder!
- Diversifica tus aportaciones
-  contribuciones de código-sin código
- Apoyar a los demás: ser un mentor





We seek to empower underrepresented groups to become more visible and active community members.

- ➔ The "new home" for Community Groups
- ➔ **#merge-forward** on the CNCF Slack

Join us!

### Merge Forward Groups

- ◆ Deaf and Hard of Hearing
- ◆ Blind and Visually Impaired
- ◆ Stuttering and Speech Diversity
- ◆ Neurodiversity
- ◆ Deep Roots
- ◆ Women in Cloud Native
- ◆ Friends of Dorothee



[community.cncf.io/merge-forward/](https://community.cncf.io/merge-forward/)



# Recursos

[I played with OTel and I liked it](#)

[Why feedback matters for Open Source](#)

<https://opentelemetry.io/community/>

<https://training.linuxfoundation.org/certification/opentelemetry-certified-associate-otca/>

<https://github.com/open-telemetry/opentelemetry-injector>

<https://github.com/open-telemetry/weaver>

<https://play.victoriametrics.com/>

<https://play-vmlogs.victoriametrics.com/>

<https://play-vtraces.victoriametrics.com/>

<https://play-grafana.victoriametrics.com/dashboards>

<https://docs.victoriametrics.com/guides/getting-started-with-opentelemetry/>

<https://github.com/VictoriaMetrics-Community/opentelemetry-demo>

# Gracias!

Bsky: @didiviking.bsky.social

X: @dianavtodea

Github: @didiViking/Conferences\_Talks

LinkedIn: @diana-todea-b2a79968

Diana Todea  
DX@VictoriaMetrics | Observability  
SME | OpenTelemetry docs localizatio...