



KubeCon



CloudNativeCon

India 2026

What Did My Agent Do? Observability and Accountability for AI Agents

Ishan Jain
Grafana Labs






Why Observability?

 LLMs are Black Boxes

 Cost & Performance Issues

 Hallucinations & Trust Issues

 **Margaritas At The Genius Bar**
@GBarEscapee

Awesome future we're making for ourselves here

Search: cheese not sticking to pizza

All Images Videos Forums Shopping News We

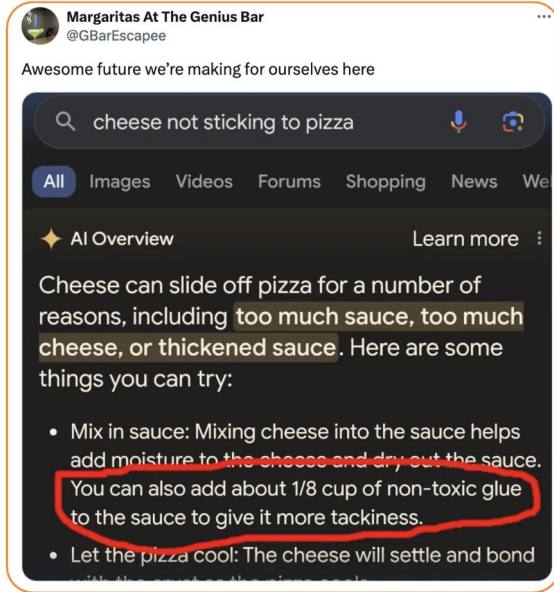
AI Overview [Learn more](#)

Cheese can slide off pizza for a number of reasons, including **too much sauce, too much cheese, or thickened sauce**. Here are some things you can try:

- Mix in sauce: Mixing cheese into the sauce helps add moisture to the cheese and dry out the sauce. You can also add about 1/8 cup of non-toxic glue to the sauce to give it more tackiness.
- Let the pizza cool: The cheese will settle and bond



What went wrong?



❓ Wrong Context?

🧠 LLM hallucinated?

☰ Bad chunking?

✉️ Bad system prompt?

➔ WE NEED TO SEE THE TRACE



**You don't know what your
agent will do until your
users use it**



LLMs are stochastic

HOW DO I BUILD
AN ALERT?

STAGING



HOW DO I BUILD
AN ALERT BASED
ON XYZ
DATASOURCE

PRODUCTION





LLMs cost a lot

Your included usage

US\$20 / US\$20

Resets 23 Jun 2026 ⓘ

On-Demand Usage

US\$1,680.57 / US\$3,000

Pay for extra usage beyond your plan limits.

US\$2,000.00 per user



Which steps cost the most?



How much does Opus 4.8 cost?



Did my last change increase my cost?



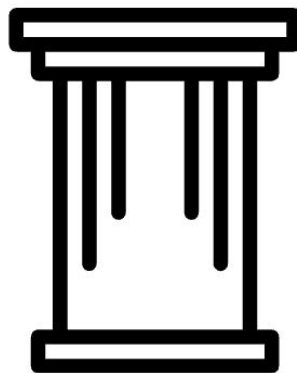
Which users cost the most?



Pillars of Observability



Metrics



Traces



Logs

What is a Trace?

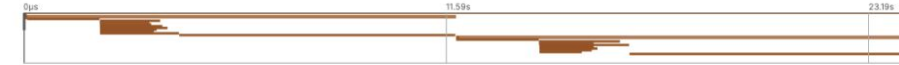
172981728931hj23b1hg31y287398198

Trace

default: invoke_workflow crew POST 200

Trace ID 797aa2699a0fe599ebdac6725511c832 Start time 2026-04-17 09:55:29.826 (7 minutes ago) Duration 46.37s Services 1 URL https://api.openai.com/v1/

Overview



Filters

Filter by attribute or text

Critical path Errors High latency

Service & Operation

Service & Operation	Duration
default invoke_workflow crew (46.37s)	46.37s
- Environment Context (1.66ms)	1.66ms
- Crew Created (1.07ms)	1.07ms
invoke_agent Travel Research Specialist (11.85s)	11.85s
- Task Created (107µs)	107µs
- Task Execution (11.85s)	11.85s
chat gpt-4o (2.07s)	2.07s
- POST (1.97s)	1.97s
execute_tool Search the internet with Serper (1.66s)	1.66s
- POST (1.65s)	1.65s
execute_tool Search the internet with Serper (1.36s)	1.36s
- POST (1.36s)	1.36s
execute_tool Search the internet with Serper (1.41s)	1.41s
- POST (1.41s)	1.41s
execute_tool Search the internet with Serper (1.72s)	1.72s
- POST (1.71s)	1.71s
execute_tool Search the internet with Serper (1.86s)	1.86s
- POST (1.85s)	1.85s
execute_tool Search the internet with Serper (1.55s)	1.55s
- POST (1.55s)	1.55s
execute_tool Search the internet with Serper (1.47s)	1.47s
- POST (1.47s)	1.47s
execute_tool Search the internet with Serper (2.17s)	2.17s
- POST (2.16s)	2.16s
chat gpt-4o (7.57s)	7.57s



What is a span?

Core Definition

A span represents a **single operation** within a trace.

Attributes

Key-value pairs providing metadata about the operation.

```
chat gpt-4o
Service: default  Duration: 2.07s  Start Time: 22.53ms (09:55:29.848)  Child Count: 1  Kind: client  Status: ok  Library Name: openlit.instrumentation.openai

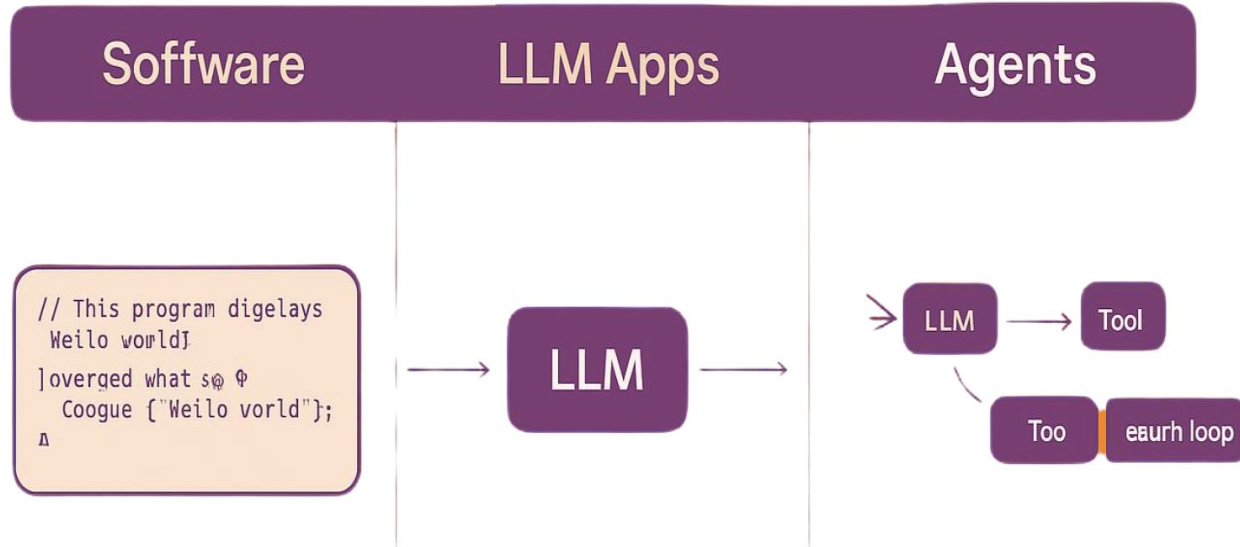
Span attributes
deployment.environment      "default"
gen_ai.client.token.usage   634

[
  {
    "role": "system",
    "parts": [
      {
        "type": "text",
        "content": "You are Travel Research Specialist. You are an experienced travel researcher with extensive knowledge of global destinations. You excel at finding the best attractions, local customs, weather patterns, and travel requirements for any destination. Your research is thorough, accurate, and focuses on providing practical information that helps create amazing travel experiences. Your personal goal is: Research destinations, attractions, and travel logistics to provide comprehensive information"
      }
    ]
  },
  {
    "role": "user",
    "parts": [
      {
        "type": "text",

```

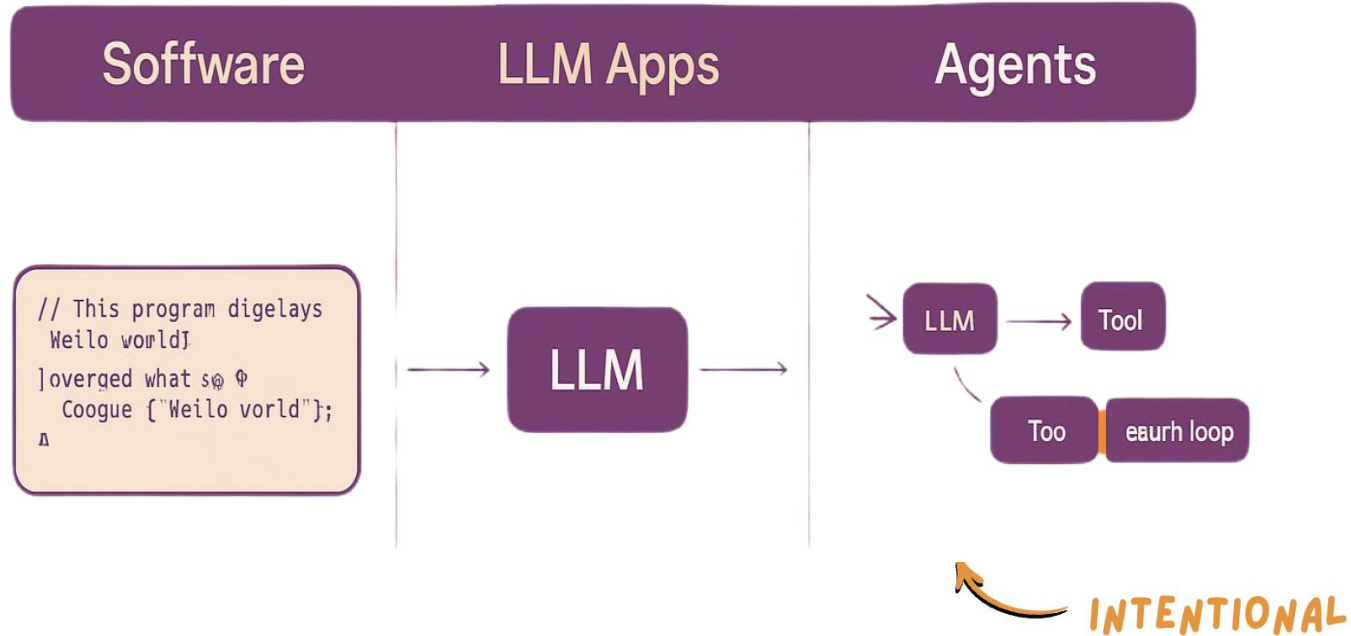


Building with AI is different





Building with AI is different





Primitives of LLM Observability

Single Turn

A single LLM request with Input and Outputs

- Captures the core interaction between the user and the model.
- Includes prompt metadata, model parameters, and response tokens.
- Essential for debugging individual model performance issues.

```
chat gpt-4o
Service: default | Duration: 2.07s | Start Time: 22.53ms (09:55:29.848) | Child Count: 1 | Kind: client | Status: ok | Library Name: openlit.instrumentation.openai

Span attributes
deployment.environment      "default"
gen_ai.client.token.usage   634

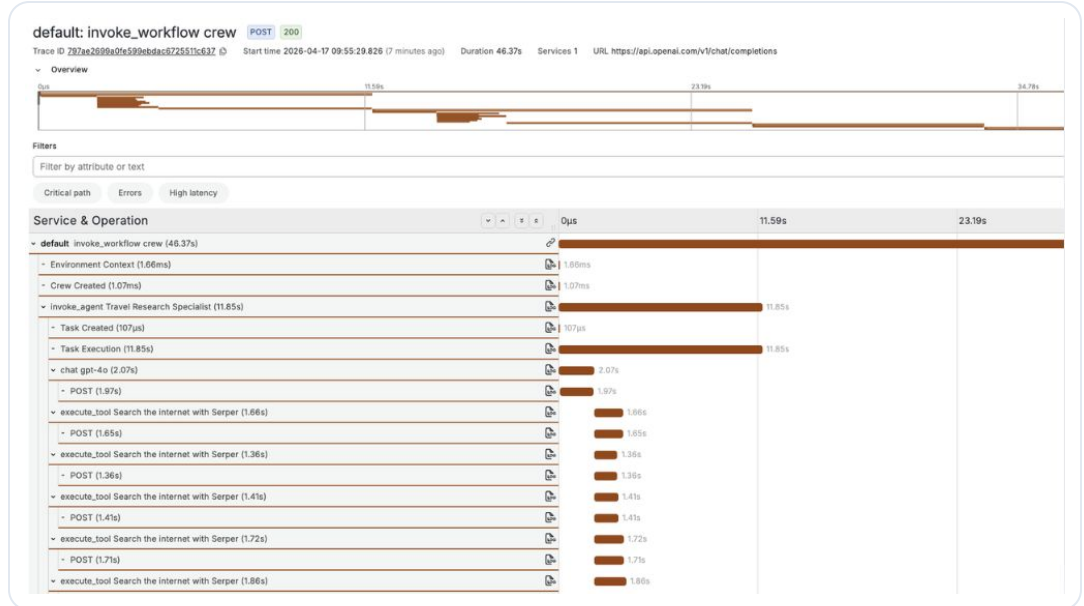
[
  {
    "role": "system",
    "parts": [
      {
        "type": "text",
        "content": "You are Travel Research Specialist. You are an experienced travel researcher with extensive knowledge of global destinations. You excel at finding the best attractions, local customs, weather patterns, and travel requirements for any destination. Your research is thorough, accurate, and focuses on providing practical information that helps create amazing travel experiences. Your personal goal is: Research destinations, attractions, and travel logistics to provide comprehensive information"
      }
    ]
  },
  {
    "role": "user",
    "parts": [
      {
        "type": "text",
```

Primitives of Agent Observability

Multi Turn

A complete Agentic run showing all LLM requests and Tools usage

- Visualizes the full lifecycle of an autonomous agent's execution.
- Tracks sequential and parallel LLM calls alongside external tool invocations.
- Crucial for identifying bottlenecks and "loops" in complex agentic workflows.





The Role of



CNCF Momentum

Second-fastest growing project in CNCF ecosystem.

Industry Standard

De facto standard for Observability across Traces, Metrics, and Logs.

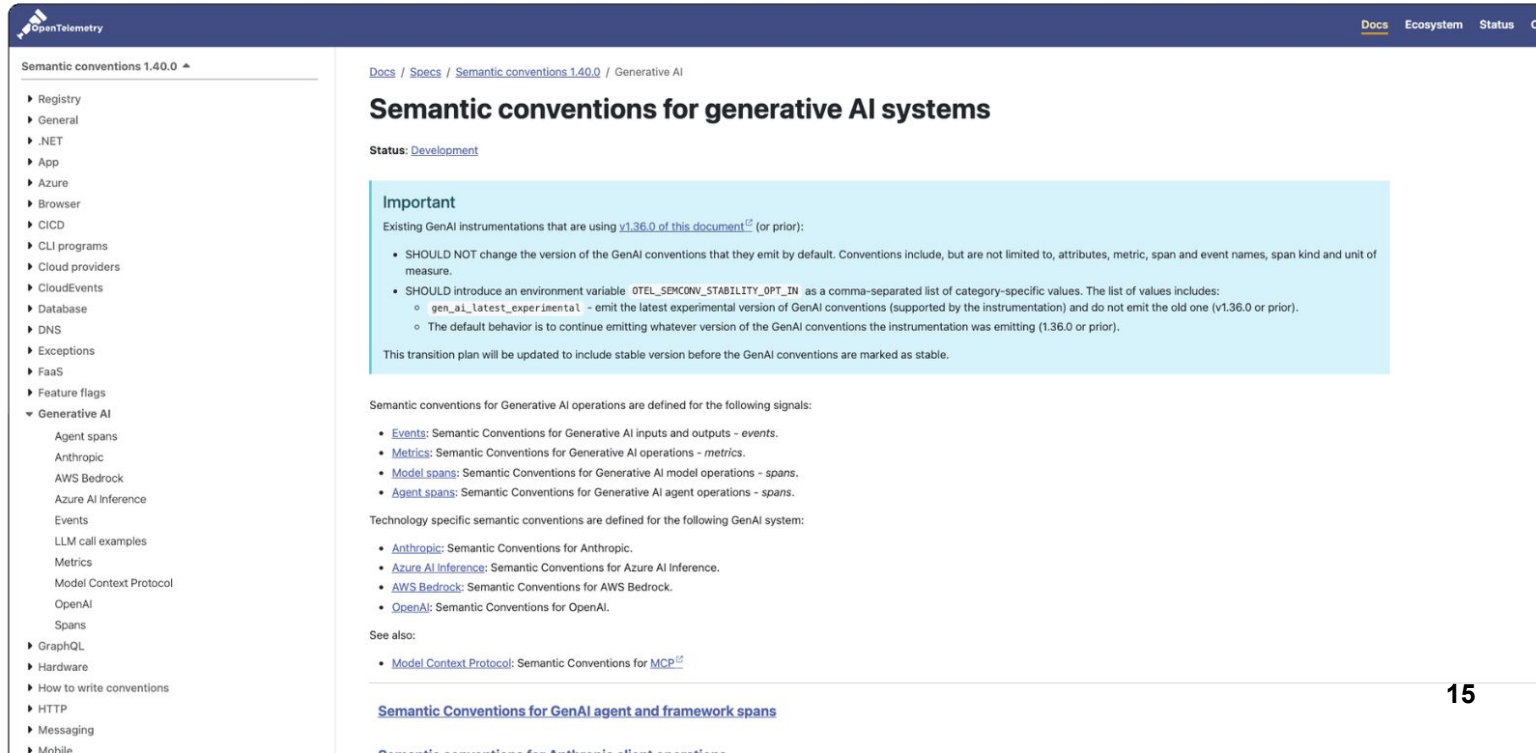
Open & Neutral

Vendor neutral approach ensuring long-term flexibility and no lock-in.

“OTel is the USB-C of observability, one open standard for every signal.”



The Role of OpenTelemetry



The screenshot shows the OpenTelemetry documentation page for "Semantic conventions 1.40.0". The page title is "Semantic conventions for generative AI systems" with a status of "Development". A light blue box highlights an "Important" section stating that existing GenAI instrumentations using v1.36.0 should not change their default conventions and should introduce an environment variable `OTEL_SEMCONV_STABILITY_OPT_IN` to specify the version to emit. Below this, a list of signals is provided: **Events**, **Metrics**, **Model spans**, and **Agent spans**. A section for "Technology specific semantic conventions" lists links for Anthropic, Azure AI Inference, AWS Bedrock, and OpenAI. The page also includes a sidebar with a navigation menu and a footer with logos for KubeCon and CloudNativeCon India 2026.

OpenTelemetry Docs Ecosystem Status

Semantic conventions 1.40.0

- Registry
- General
- .NET
- App
- Azure
- Browser
- CICD
- CLI programs
- Cloud providers
- CloudEvents
- Database
- DNS
- Exceptions
- FaaS
- Feature flags
- Generative AI
 - Agent spans
 - Anthropic
 - AWS Bedrock
 - Azure AI Inference
 - Events
 - LLM call examples
 - Metrics
 - Model Context Protocol
 - OpenAI
 - Spans
- GraphQL
- Hardware
- How to write conventions
- HTTP
- Messaging
- Mobile

Docs / Specs / Semantic conventions 1.40.0 / Generative AI

Semantic conventions for generative AI systems

Status: [Development](#)

Important

Existing GenAI instrumentations that are using [v1.36.0 of this document](#) (or prior):

- SHOULD NOT change the version of the GenAI conventions that they emit by default. Conventions include, but are not limited to, attributes, metric, span and event names, span kind and unit of measure.
- SHOULD introduce an environment variable `OTEL_SEMCONV_STABILITY_OPT_IN` as a comma-separated list of category-specific values. The list of values includes:
 - `gen_ai_latest_experimental` - emit the latest experimental version of GenAI conventions (supported by the instrumentation) and do not emit the old one (v1.36.0 or prior).
 - The default behavior is to continue emitting whatever version of the GenAI conventions the instrumentation was emitting (1.36.0 or prior).

This transition plan will be updated to include stable version before the GenAI conventions are marked as stable.

Semantic conventions for Generative AI operations are defined for the following signals:

- [Events](#): Semantic Conventions for Generative AI inputs and outputs - *events*.
- [Metrics](#): Semantic Conventions for Generative AI operations - *metrics*.
- [Model spans](#): Semantic Conventions for Generative AI model operations - *spans*.
- [Agent spans](#): Semantic Conventions for Generative AI agent operations - *spans*.

Technology specific semantic conventions are defined for the following GenAI system:

- [Anthropic](#): Semantic Conventions for Anthropic.
- [Azure AI Inference](#): Semantic Conventions for Azure AI Inference.
- [AWS Bedrock](#): Semantic Conventions for AWS Bedrock.
- [OpenAI](#): Semantic Conventions for OpenAI.

See also:

- [Model Context Protocol](#): Semantic Conventions for [MCP](#)

Semantic Conventions for GenAI agent and framework spans

Semantic conventions for Anthropic client operations

KubeCon CloudNativeCon India 2026

15



Adding Observability to Agents

Basic

```
export OTEL_PYTHON_LOGGING_AUTO_INSTRUMENTATION_ENABLED=true
opentelemetry-instrument \
  --traces_exporter console \
  --metrics_exporter console \
  --logs_exporter console \
  --service_name dice-server \
  python my_ai_agent.py
```



Adding Observability to Agents

SLOW

```
from openai import OpenAI
from opentelemetry import trace
from opentelemetry.sdk.resources import SERVICE_NAME, Resource
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.trace.export import SimpleSpanProcessor, ConsoleSpanExporter

# --- OpenTelemetry setup ---
# Configure the tracer provider with a console exporter for demo purposes.
trace.set_tracer_provider(
    TracerProvider(resource=Resource.create({SERVICE_NAME: "openai-client"}))
)
tracer = trace.get_tracer(__name__)

span_processor = SimpleSpanProcessor(ConsoleSpanExporter())
trace.get_tracer_provider().add_span_processor(span_processor)

# --- OpenAI client ---
client = OpenAI(api_key="YOUR_API_KEY")

def instrumented_chat_completion(messages, model="gpt-4o-mini"):
    # Start a span around the OpenAI call
    with tracer.start_as_current_span("openai.chat_completion") as span:
        span.set_attribute("openai.model", model)
        span.set_attribute("openai.messages_count", len(messages))

        response = client.chat.completions.create(
            model=model,
            messages=messages
        )

        # You can record tokens or usage metrics as attributes too:
        usage = response.usage
        if usage:
            span.set_attribute("openai.prompt_tokens", usage.prompt_tokens)
            span.set_attribute("openai.completion_tokens", usage.completion_tokens)
            span.set_attribute("openai.total_tokens", usage.total_tokens)

        return response.choices[0].message.content

# Example usage:
reply = instrumented_chat_completion([
    {"role": "user", "content": "Write a haiku about clouds."}
])
print("Model reply:", reply)
```



Adding Observability to Agents



```
import openlit  
  
openlit.init()
```

```
openlit-instrument \  
  --service-name my-ai-app \  
  --environment production \  
python my_ai_agent.py
```



Adding Observability to Agents





Adding Observability to Agents (2026)

The screenshot shows the OpenTelemetry documentation website. The top navigation bar includes links for Docs, Ecosystem, Status, Community, Training, Blog, and a language selector set to English. The left sidebar contains a 'Docs' section with a tree view: 'What is OpenTelemetry?', 'Getting Started', 'Concepts', 'Demo', 'Language APIs & SDKs', 'Platforms', 'Zero-code Instrumentation', and 'OBI'. Under 'OBI', there are sub-links for 'Configure', 'Network', and 'Setup'. The main content area shows the breadcrumb 'Docs / Zero-code Instrumentation / OBI' followed by the title 'OpenTelemetry eBPF Instrumentation'. The text describes how to use OpenTelemetry eBPF Instrumentation for automatic instrumentation, notes that getting started with distributed tracing can be complex, and explains that OBI is an auto-instrumentation tool that uses eBPF to inspect application executables and the OS networking layer. It also lists features offered by OBI.

OpenTelemetry

[Docs](#) [Ecosystem](#) [Status](#) [Community](#) [Training](#) [Blog](#) [English](#)

Docs

- What is OpenTelemetry?
- ▶ Getting Started
- ▶ Concepts
- ▶ Demo
- ▶ Language APIs & SDKs
- ▶ Platforms
- ▼ Zero-code Instrumentation
 - ▼ **OBI**
 - ▶ Configure
 - ▶ Network
 - ▶ Setup

[Docs](#) / [Zero-code Instrumentation](#) / OBI

OpenTelemetry eBPF Instrumentation

Learn how to use OpenTelemetry eBPF Instrumentation for automatic instrumentation.

OpenTelemetry libraries provide telemetry collection for popular programming languages and frameworks. However, getting started with distributed tracing can be complex. In some compiled languages like Go or Rust, you must manually add tracepoints to the code.

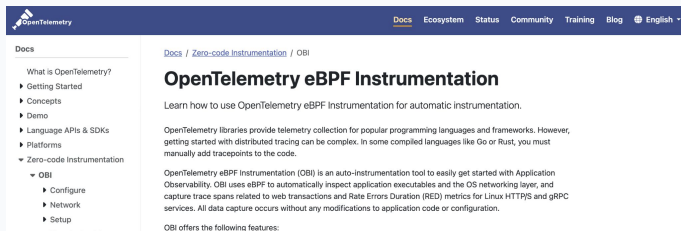
OpenTelemetry eBPF Instrumentation (OBI) is an auto-instrumentation tool to easily get started with Application Observability. OBI uses eBPF to automatically inspect application executables and the OS networking layer, and capture trace spans related to web transactions and Rate Errors Duration (RED) metrics for Linux HTTP/S and gRPC services. All data capture occurs without any modifications to application code or configuration.

OBI offers the following features:



Adding Observability to Agents (2026)

INSTRUMENTATION



BACKEND & VISUALIZATION



Traces



Metrics



Visualization



Demo (Dangerous)





Agent Evaluations

You are testing **reasoning**, not code paths

Single Turn

Did the Agent respond correctly according to the user prompt?

Multi Turn

Did the Agent maintain context across a conversation and perform well in e2e execution?



Agent Evaluations

We are testing **reasoning**, not code paths

Single Turn

Did the Agent respond correctly according to the user prompt?

Multi Turn

Did the Agent maintain context across a conversation and perform well in e2e execution?



KubeCon



CloudNativeCon

India 2026

Thankyou!

Ishan Jain
Grafana Labs

Linked in

