Finding Your Balance: Observability with Elastic and OpenTelemetry

A Hybrid Approach to Automatic and Manual Instrumentation in Python

Vancouver Cloud Summit - May 2025

Justin Castilla

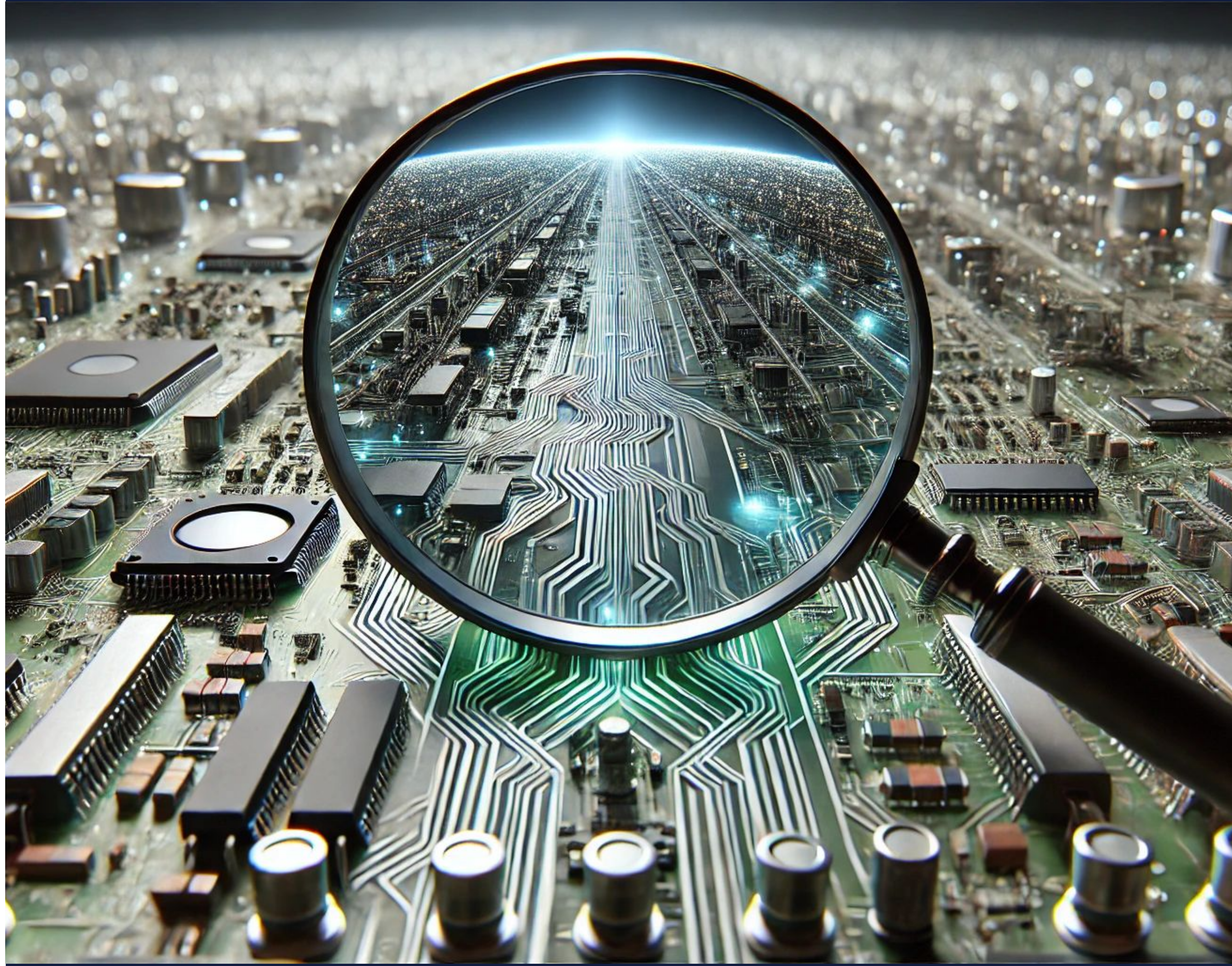
Senior Developer Advocate @ Elastic

Bluesky: @justincastilla.bsky.social

justin.castilla@elastic.co

Talk Agenda:

- Observability Concepts
- OpenTelemetry Framework
  - Automatic Instrumentation
  - Manual Instrumentation
- A Case for a Hybrid Approach
- Hybrid Observability in Practice
- Questions Answered



Observability Concepts

What is Observability?

- Observability is the understanding of what is really happening to a system
- It’s about the who, what, where, when, and why of your application’s internal travel - not just state you are in at the moment
- Observability helps you determine what's happening inside your system by analyzing the data it outputs

Observability Concepts

Who is using your application or visiting your site?

What can be done to fix detected anomalous behavior?

Where is the bottleneck in performance?

When is your service running the slowest?

Why is your site running slow in EMEA?

Observability Concepts

3 Pillars of Observability:

Logs	Traces	Metrics
Records of events in a system, documenting operations, errors, and activities	Tracks the path and interactions of a request through a system.	Quantitative measurements that track the performance and health of a system

We draw from all three individually and in combination to tell a story about your application.

Observability Concepts

Observability provides *context*.

- Raw data alone *isn't* enough—context turns noise into actionable insights.
- Without context, it's like reading a list of random events with no understanding of how they're connected.
- Knowing a request failed is good; knowing why it failed is better.
- Observability is *not* about data volume but about providing meaningful answers through context.

The OpenTelemetry Framework

OpenTelemetry as an open standard

- A merger of two existing open source projects: OpenTracing and OpenCensus.
- An observability framework that is open source designed to work with any backend system.
- It provides standardized APIs, libraries, and tools to collect telemetry data.
- You may point your data from any supported platform to another with reasonably high confidence of 1:1 portability thanks to ECS (Elastic Common Schema)

The OpenTelemetry Framework

Instrumentation

The process of adding observability features to your application to collect telemetry data, such as traces, metrics, and logs.

The OpenTelemetry Framework

Automatic Instrumentation

- A script is invoked *before* your application is started and the service monitors activity of the available libraries. Supported modules can be found at the [OpenTelemetry Registry]
- This process typically covers HTTP, gRPC, database calls, and other common package interactions, depending on the instrumentation libraries enabled in your setup.

The OpenTelemetry Framework

Automagic is that easy!

- Very fast setup - most frameworks simply require that you run the wrapper script before your server startup
- Most common frameworks are supported (Django, Flask, FastAPI)
- Reference an options object or define environment variables containing sample frequency, packages to ignore, resource limits, etc.
- Demo: http://localhost:4999

The OpenTelemetry Framework

Automatic Challenges

- Lack of application-specific context - we only see the travel patterns, but not the data being sent
- Noise from irrelevant spans - this has the potential to create a LOT of unnecessary chatter
- Doesn't always align with your business context - creates uninformative, unactionable data

The OpenTelemetry Framework

Manual Instrumentation

- OTel code segments are written *into* your application to collect and transmit telemetry data.
- A collector receives telemetry data and sends them via the OTel protocol to a specific destination.
- You decide where, when, and what telemetry data is sent to the data platform, as well as how frequently and in what conditions.

The OpenTelemetry Framework

Manual Advantages

- Adds application-specific detail to traces
- Enables visibility into critical business workflows for your specific situation
- Low data transfer potential
- Filter attributes via processors.
- Demo: http://localhost:5000

The OpenTelemetry Framework

Manual Challenges

- Requires effort and discipline to implement
- Planning is necessary to write code to trace the paths and attributes that matter to you
- Risk of missing critical spans consequence of poor planning from the above
- Updates to your codebase may require updates to your manual implementation as well

The OpenTelemetry Framework

A hybrid approach

- Automatic provides a broad, sweeping view of your telemetry data as it traverses your application's landscape. You are able to see everything from one platform
- Manual allows you to select specific data to track that are of particular importance to your logic workflow. You choose the data to observe.

The OpenTelemetry Framework

A hybrid approach

By configuring your automatic instrumentation to *only* cover what paths and libraries you want covered and with a set interval rate, you can reduce noise and ensure high performance.

You can also explore logs, metrics, and traces *specific* to your observability needs

Bringing it All Together

Open Source for a reason

- You can use Elastic or other data platforms!
- Most modern platforms adhere to the OTel ECS standard
- It's an observability market, not a data platform market
- You can create your own instrumentation client for your own clients and libraries!

Bringing it All Together

Key Questions for a Balanced Approach

- What is your business-critical functionality?
    - What do you need to know, how often, and where does that info come from?
- What parts of your app already have rich telemetry?
    - Can you add to it?
    - How can you integrate that data into Open Telemetry?
- What's the overhead vs. value?
    - How important is this information to you temporally?
    - How much computation are you willing to allot to telemetry?

Best Practices for Hybrid Instrumentation

Strike your own balance

- Start with auto-instrumentation and layer manual spans incrementally.
  - This prevents duplication of efforts
- Evolve instrumentation as your application evolves.
  - Can you add to it?
  - Check in with your observability to identify any blindspots
- Use Elastic dashboards to validate and refine your telemetry strategy.
  - The data is there to be used!

Hybrid Instrumentation for the win!

Strike your own balance

- Observability *isn't* one-size-fits-all.
- Use automatic instrumentation for *quick* wins, and manual for *depth*.
- Elastic bridges raw data and actionable *insights*.
- Find your *balance* and focus on what matters most to your users.

Hybrid Instrumentation for the win!

Resources

- OpenTelemetry Demo (Elastic fork)
- OpenTelemetry Python SDK.
- Elastic Observability docs.
- GitHub repo with demo code.

Thank you!

Justin Castilla

Senior Developer Advocate @ Elastic

Bluesky: @justincastilla.bsky.social

justin.castilla@elastic.co