



splunk>

Zipkin & Splunk: Tracing Transactions Across Your ecosystem

Tom Martin | Staff Practitioner, Splunk
tomm@splunk.com

October 2018



Zipkin & Splunk: Tracing Transactions Across Your Ecosystem

- ▶ Session ID: 1256
 - ▶ Title: Zipkin & Splunk: Tracing Transactions across your ecosystem
 - ▶ Abstract:

Containers, microservices and serverless computing are here to stay, but how do you track performance across these complex, short-lived ecosystems? Zipkin is an open source, open tracing library that enables you to trace transactions from system to system in a standard way. Those traces can be collected and visualized in Splunk! Come see how to use Zipkin and Splunk together to see how transactions navigate your systems and services.

Forward-Looking Statements

During the course of this presentation, we may make forward-looking statements regarding future events or the expected performance of the company. We caution you that such statements reflect our current expectations and estimates based on factors currently known to us and that actual events or results could differ materially. For important factors that may cause actual results to differ from those contained in our forward-looking statements, please review our filings with the SEC.

The forward-looking statements made in this presentation are being made as of the time and date of its live presentation. If reviewed after its live presentation, this presentation may not contain current or accurate information. We do not assume any obligation to update any forward-looking statements we may make. In addition, any information about our roadmap outlines our general product direction and is subject to change at any time without notice. It is for informational purposes only and shall not be incorporated into any contract or other commitment. Splunk undertakes no obligation either to develop the features or functionality described or to include any such feature or functionality in a future release.

Splunk, Splunk>, Listen to Your Data, The Engine for Machine Data, Splunk Cloud, Splunk Light and SPL are trademarks and registered trademarks of Splunk Inc. in the United States and other countries. All other brand names, product names, or trademarks belong to their respective owners. © 2018 Splunk Inc. All rights reserved.

Agenda

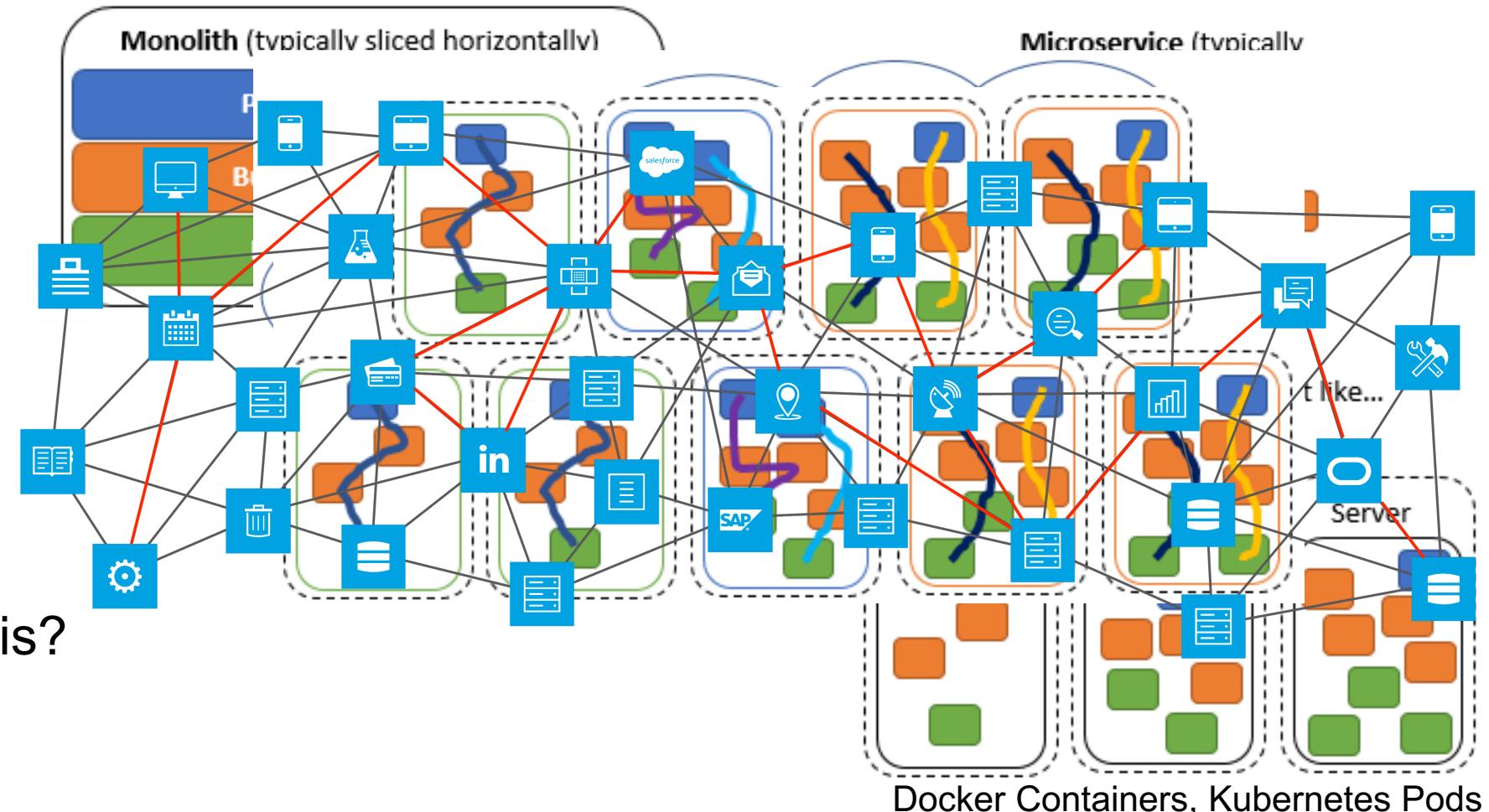
1. App Dev Flashback & Future
2. Observability vs. Monitoring
3. Distributed Tracing
4. Open Tracing
5. Zipkin + Splunk
6. Demo

Let's Start at the Beginning

Containers, Microservices and Serverless...Oh my!

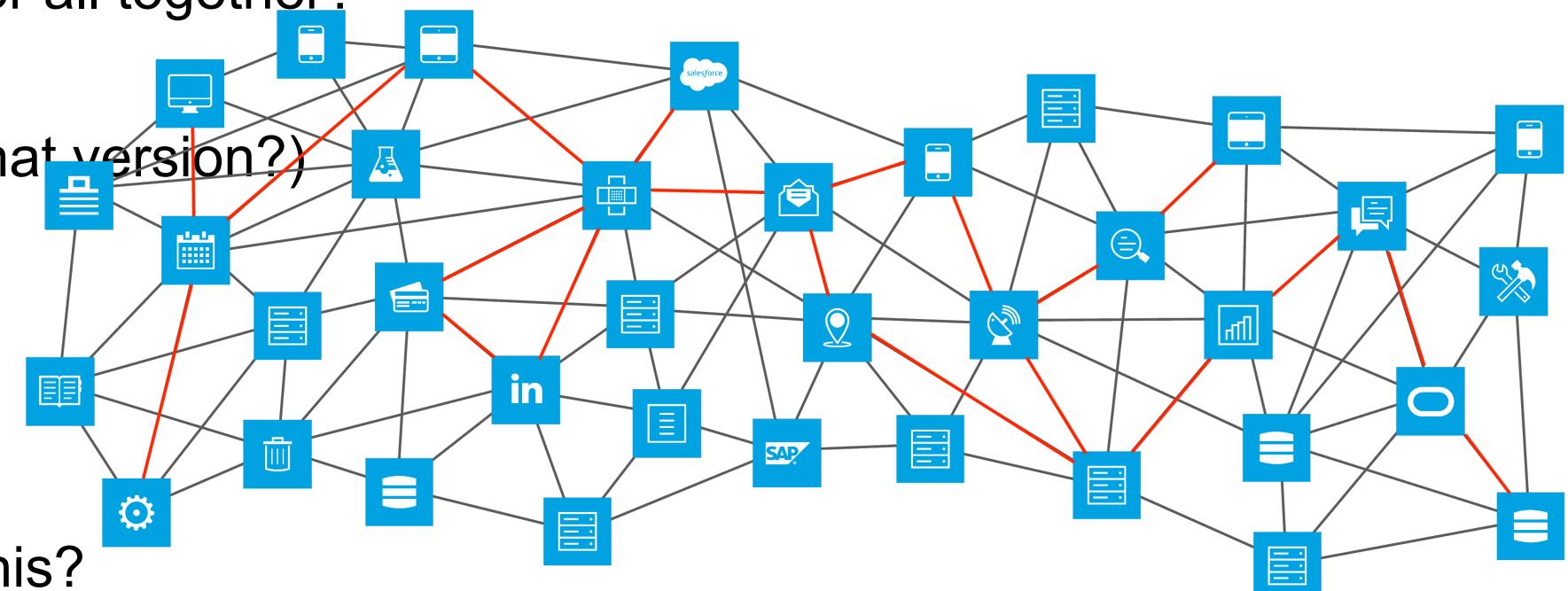
- ▶ Monolith apps
- ▶ Distributed apps
- ▶ Microservices
- ▶ Serverless
- ▶ Transactions
- ▶ External Systems?

- ▶ How do I monitor this?



But Wait, There's More Complexity

- ▶ Dockerless Containers (OCI), Kubernetes or Mesos?
 - ▶ AWS, GCP, Azure or all together?
 - ▶ IaaS, PaaS
 - ▶ DevOps, CI/CD (what version?)



- ## ► How Do I monitor this? Logs, Metrics & Tracing

“*Observability*”

Monitoring or Observability?

Both!

Monitoring

- ▶ Logs & Metrics
 - ▶ Passive (poll)
 - '*I observe you*'
 - ▶ Measure machines, networks, apps
 - ▶ Overall Health
 - ▶ Use Cases
 - Monitoring
 - Alerting
 - Troubleshooting
 - Capacity Planning

Observability

- ▶ Logs, Metrics & Traces
 - ▶ Active (push/code)
 - ‘*Make yourself observable*’
 - ▶ Measure interactions
 - ▶ Granular Insights
 - ▶ Use Cases ++:
 - Debugging
 - Distributed Tracing
 - Performance Analysis
 - Behavioral Analytics

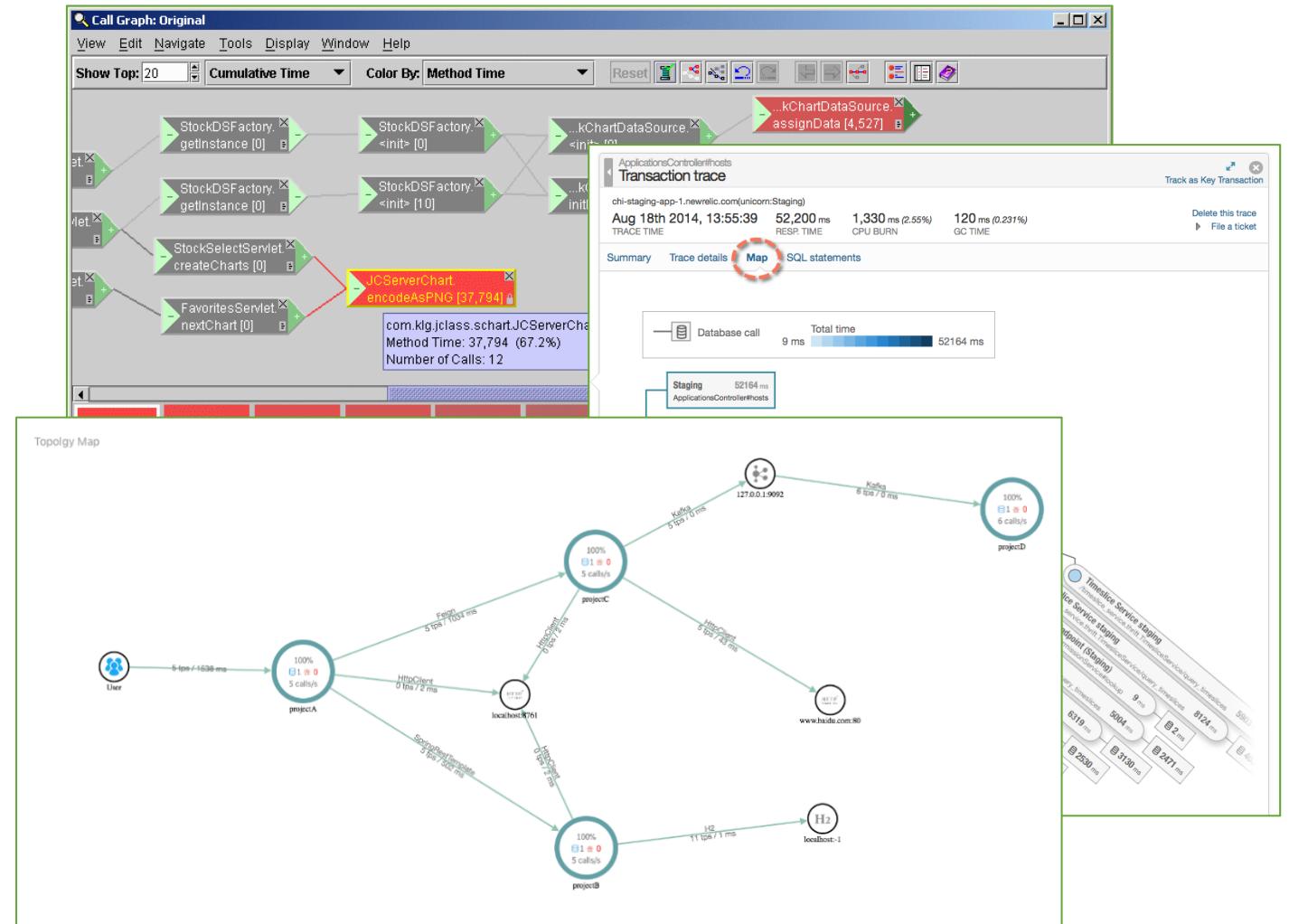
Distributed Tracing

► A long history...

- Tag-and-Follow
- Application Tracing
- Transaction Tracing
- Cross Process Tracing

► Conceptually easy, but difficult to implement

- GUID? SessionID? UserID?
- Wait...I can't change that system
- Multiple Vendors
- No consistency



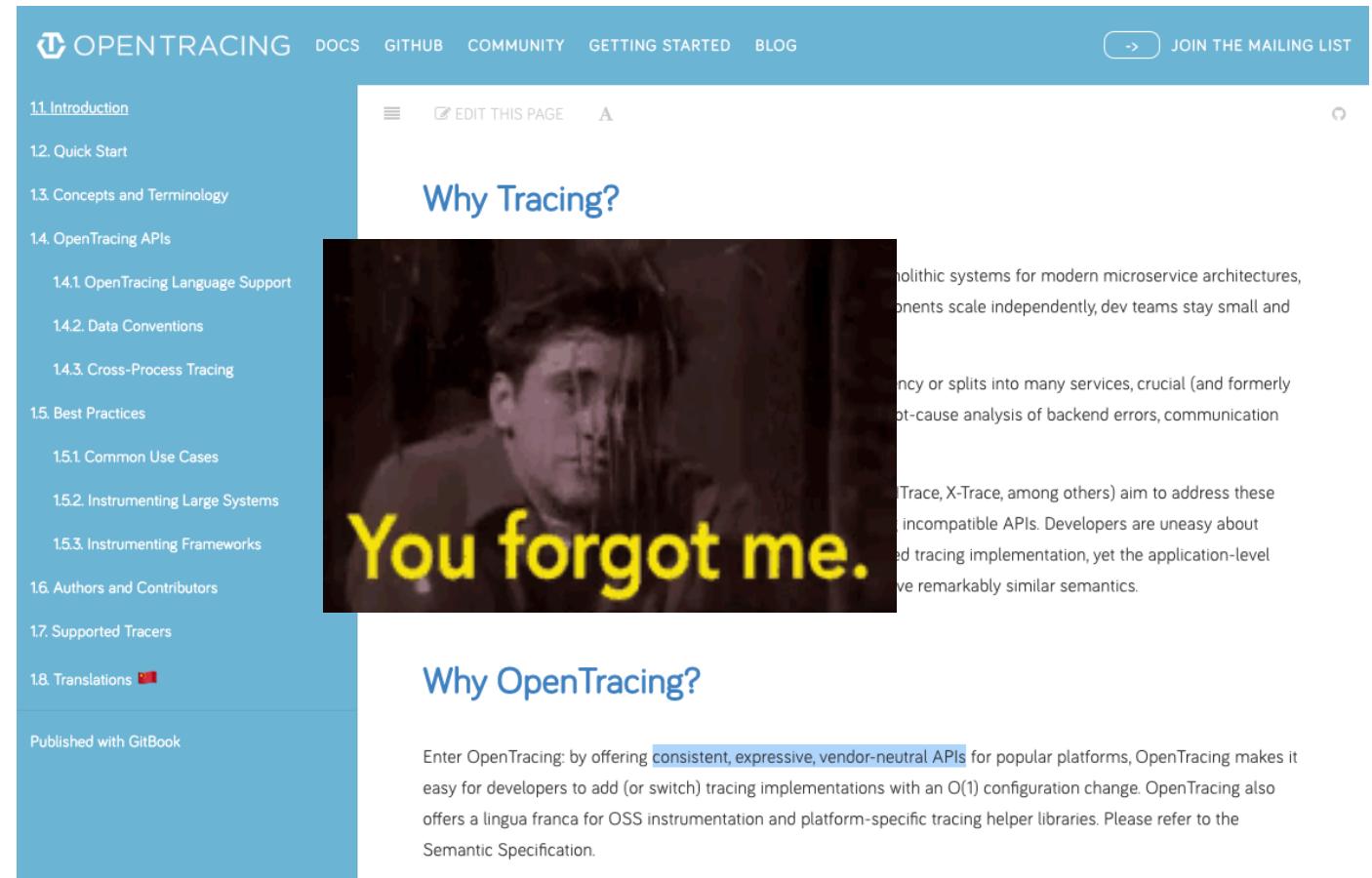
Enter Open Tracing

- ▶ In the beginning...
 - There was [Dapper](#) (2010)

- ▶ Great if you're Google 😞
 - Consistent hardware
 - Application templates
 - Zero involvement to instrument

- ▶ Enter OpenTracing
 - Consistent, vendor-neutral APIs
 - Multiple implementations
 - Plug-n-Play

- ▶ Zipkin, Jaeger, Lightstep



The screenshot shows the official OpenTracing website at opentracing.io. The header includes the OpenTracing logo, navigation links for DOCS, GITHUB, COMMUNITY, GETTING STARTED, and BLOG, and a JOIN THE MAILING LIST button. The main content area features a large image of a man looking slightly off-camera with the text "You forgot me." overlaid in yellow. To the left is a sidebar with navigation links: 1.1 Introduction, 1.2 Quick Start, 1.3 Concepts and Terminology, 1.4 OpenTracing APIs, 1.4.1 OpenTracing Language Support, 1.4.2 Data Conventions, 1.4.3 Cross-Process Tracing, 1.5 Best Practices, 1.5.1 Common Use Cases, 1.5.2 Instrumenting Large Systems, 1.5.3 Instrumenting Frameworks, 1.6 Authors and Contributors, 1.7 Supported Tracers, and 1.8 Translations (with a Chinese flag icon). Below the sidebar, it says "Published with GitBook". The main content area has two sections: "Why Tracing?" and "Why OpenTracing?". The "Why Tracing?" section discusses the evolution from monolithic to microservices architectures and the challenges of tracing across multiple services. The "Why OpenTracing?" section explains the benefits of having a consistent, vendor-neutral API for tracing.

Why Tracing?

Holistic systems for modern microservice architectures, components scale independently, dev teams stay small and...
ency or splits into many services, crucial (and formerly...
ot-cause analysis of backend errors, communication...

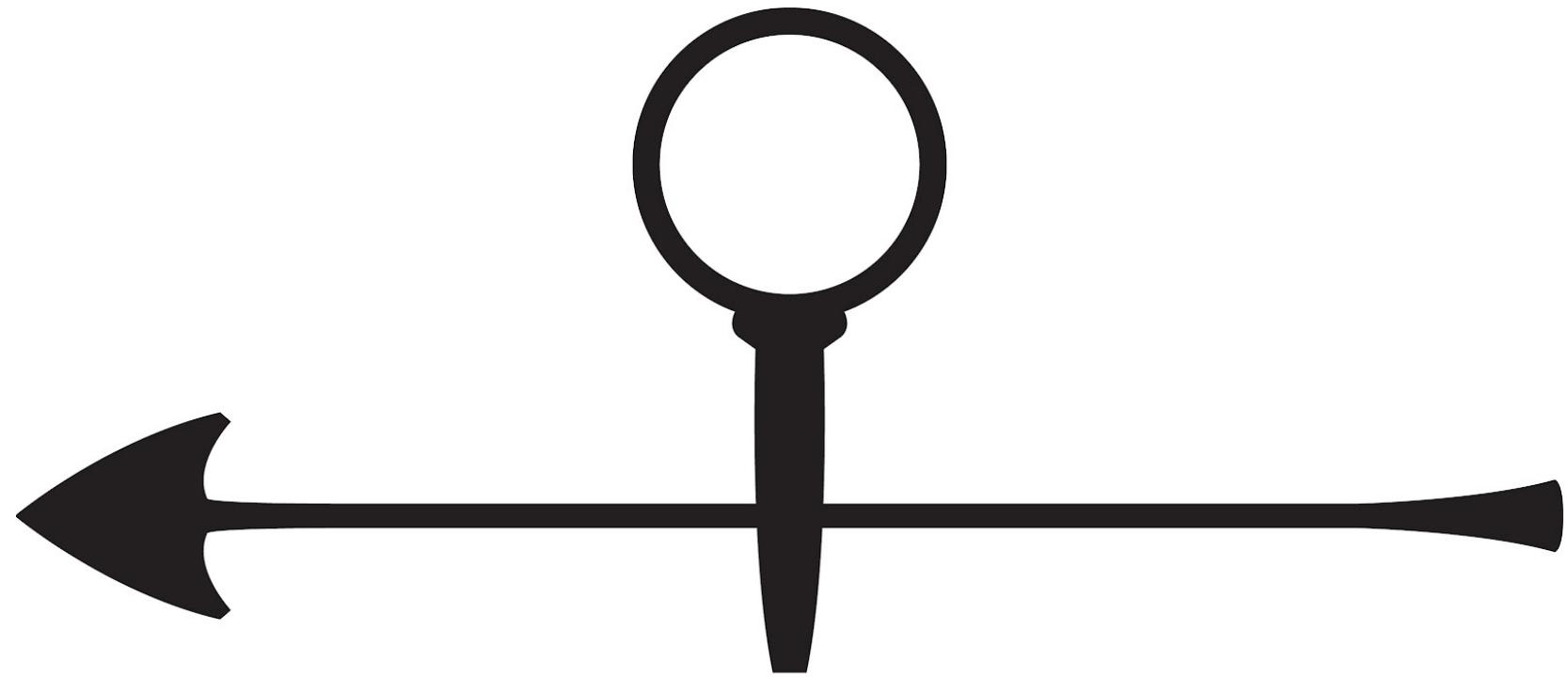
You forgot me.

Why OpenTracing?

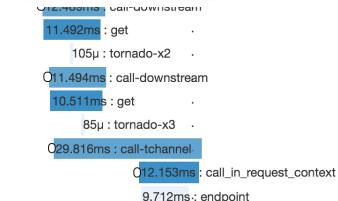
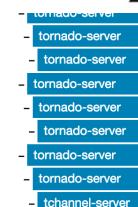
Enter OpenTracing: by offering [consistent, expressive, vendor-neutral APIs](#) for popular platforms, OpenTracing makes it easy for developers to add (or switch) tracing implementations with an O(1) configuration change. OpenTracing also offers a lingua franca for OSS instrumentation and platform-specific tracing helper libraries. Please refer to the Semantic Specification.

Zip**Zipkin**What?

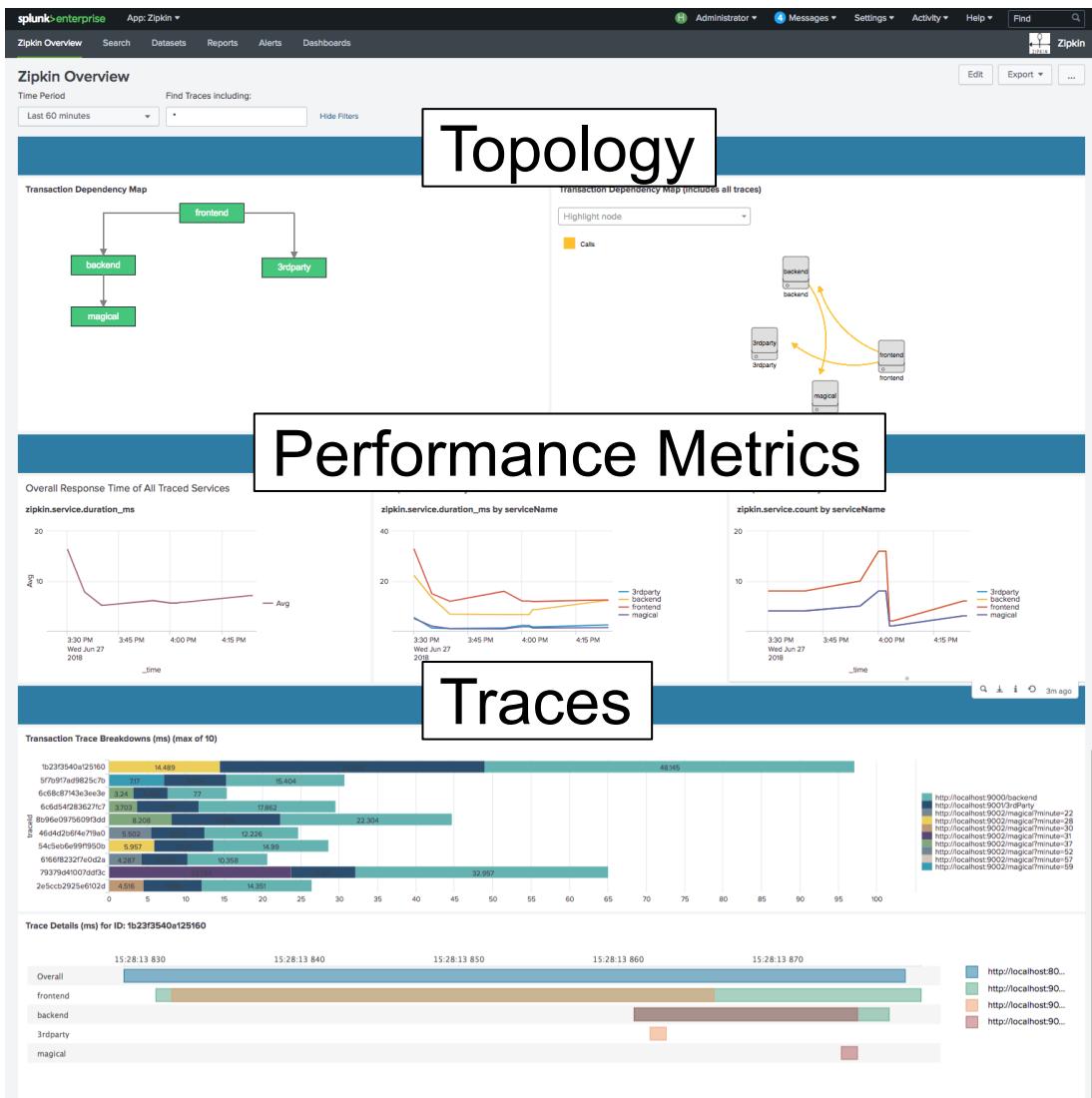
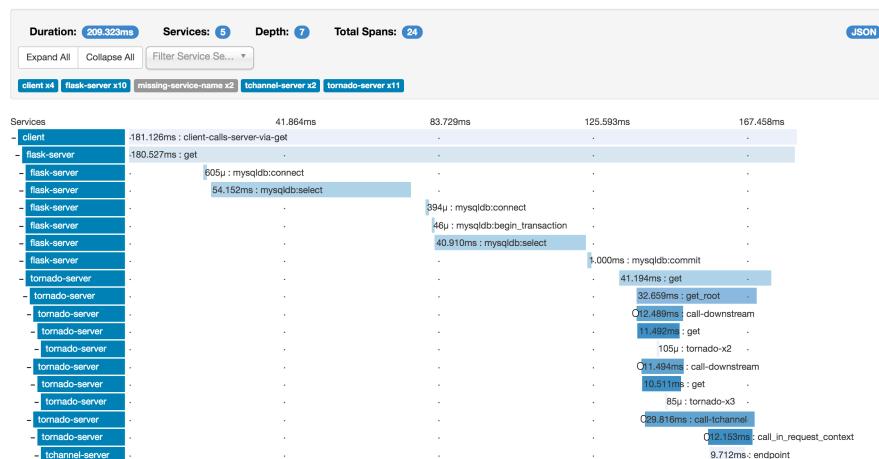
An OpenTracing
Implementation



Z I P K I N



Zipkin OpenTracing with Splunk!



OK, But How?

- ▶ HTTP Event Collector
- ▶ Zipkin ‘Recorder’
- ▶ Application Code

```
// Create an instance of the Splunk recorder
const {recorder} = require('./recorder');

// initialize tracer
const tracer = new Tracer({ctxImpersonate: true});

// instrument the client
const zipkinRest = rest.wrap(restInterface, tracer);
```

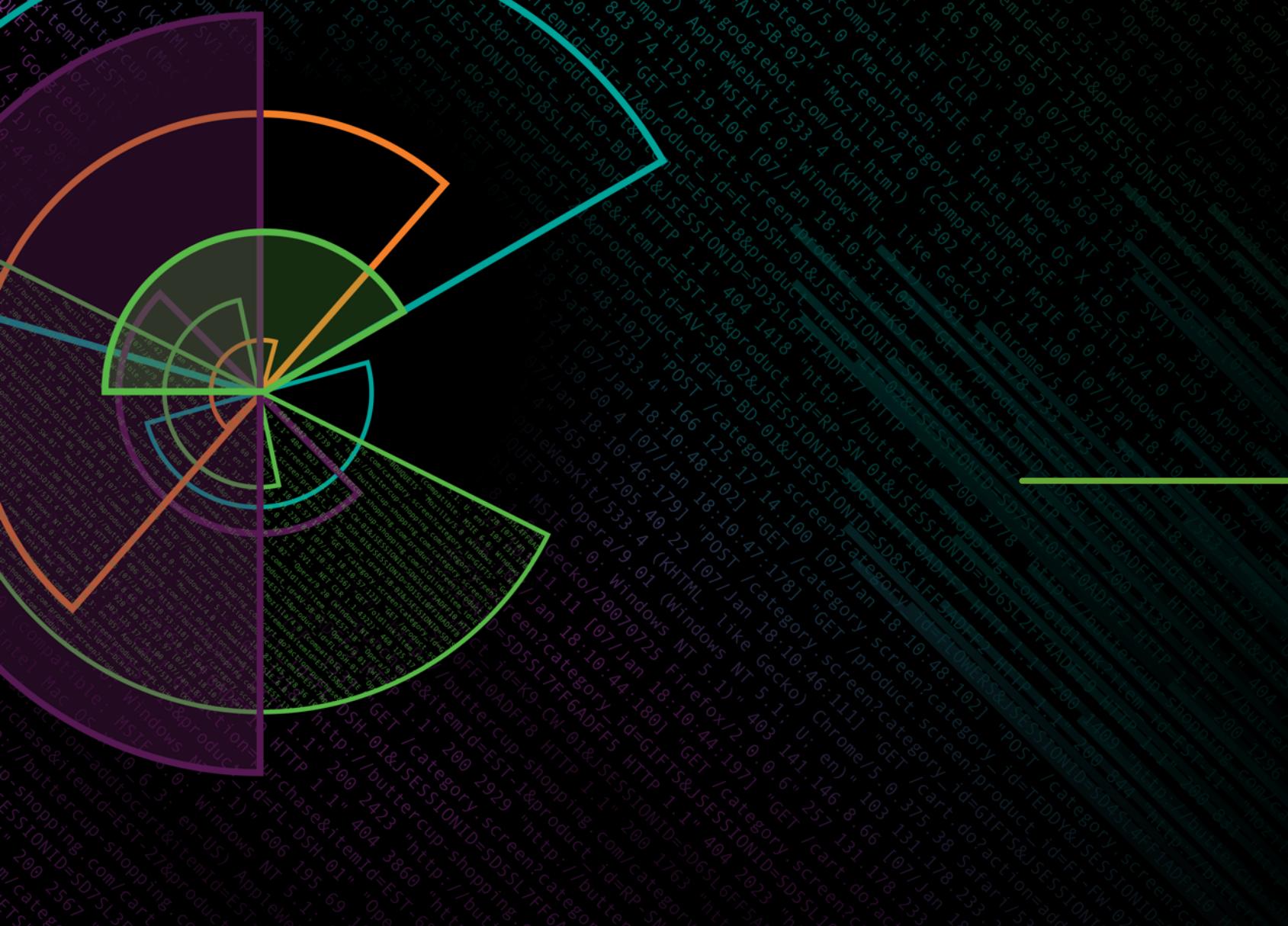
The screenshot shows the Splunk web interface with the following details:

- Top Navigation:** Administrator, Messages, Settings (highlighted), Activity, Help, Find.
- Left Sidebar:** KNOWLEDGE (Searches, reports, and alerts, Data models, Event types, Tags, Fields, Lookups, User interface) and DATA (Data inputs, Forwarding and receiving, Indexes, Report acceleration summaries, Virtual indexes, Source types).
- Main Content Area:**
 - Data inputs:** Sub-section of DATA. Description: Set up data inputs from files and directories, network ports, and scripted inputs. If you want to set up forwarding and receiving between two Splunk instances, go to [Forwarding and receiving](#).
 - Local inputs:** Table view of inputs:

Type	Inputs	Actions
Files & Directories	6	+ Add new
HTTP Event Collector	1	+ Add new
 - HTTP Event Collector Detail View:**
 - Name:** Zipkin
 - Description:** Endpoint for Zipkin Traces
 - Token Value:** 00000000-0000-0000-0000-000000000002 (highlighted with a red box)
 - Actions:** Edit, Disable, Delete
- Bottom Right Corner:** A yellow speech bubble containing the text "splunk.com" with a small arrow pointing towards it.

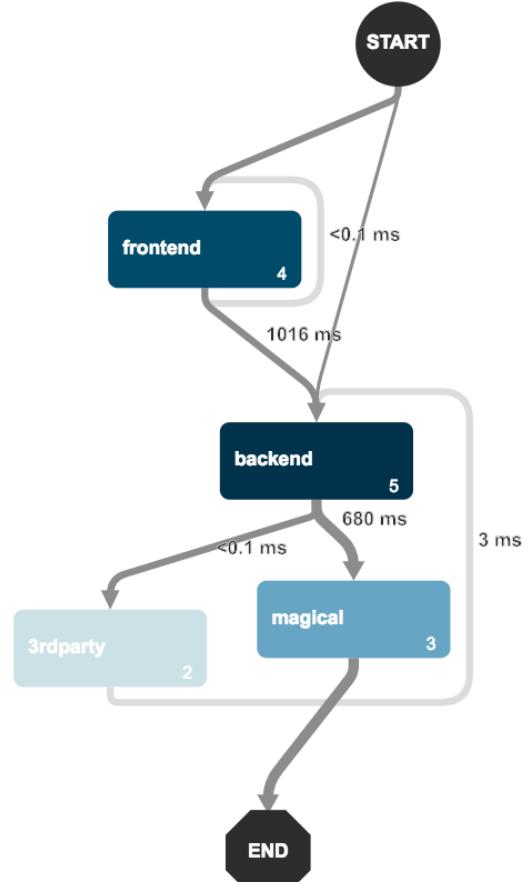
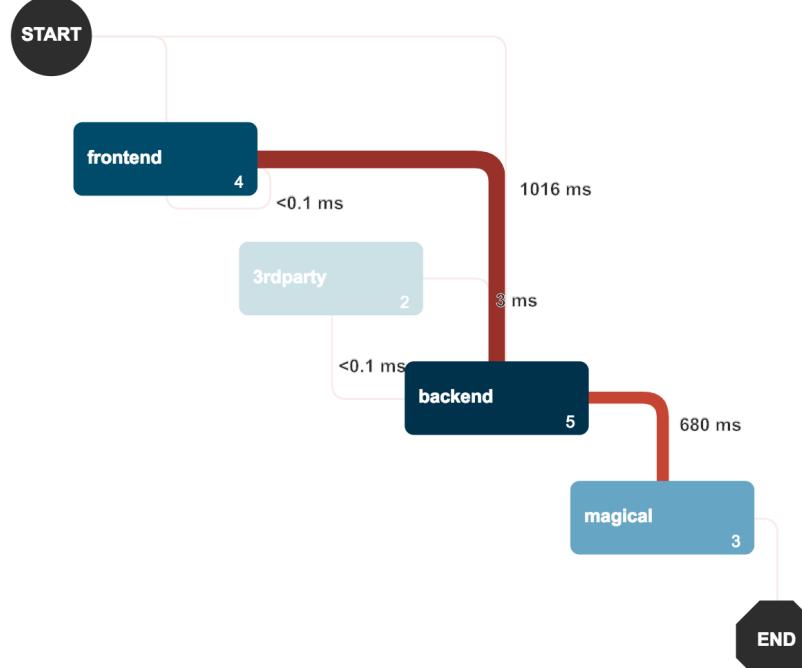
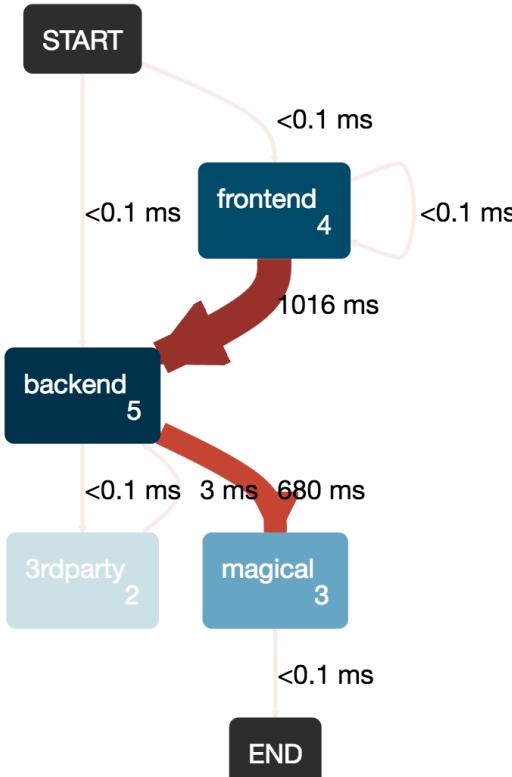
Demo!

Splunk + Zipkin



More Visualizations...

😴 More on this tomorrow 😴



Helpful Links

- ▶ [Dapper Paper](#)
- ▶ OpenTracing - <http://opentracing.io/>
- ▶ Open Source Implementations
 - Zipkin - <https://zipkin.io/>
 - Jaeger - <https://www.jaegertracing.io/>
- ▶ Commercial Implementations
 - LightStep - <https://lightstep.com/>
 - Instana - <https://www.instana.com/>
- ▶ This Demo:
 - [Zipkin-Demo Github Repository](#)
 - [Docker image \(Splunk with Zipkin App\)](#)
- ▶ Jaeger Demos & Tutorials
 - [Distributed Tracing with NGINX](#)
 - [Jaeger Implementations](#)
 - [HotRod Tutorial](#)
 - [MicroDonuts](#)
 - [Case Study: Distributed Tracing @ Uber](#)

Thank You!

Don't forget to rate this session
in the .conf18 mobile app

