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.

splunk> .conf18

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> .conf18

Agenda

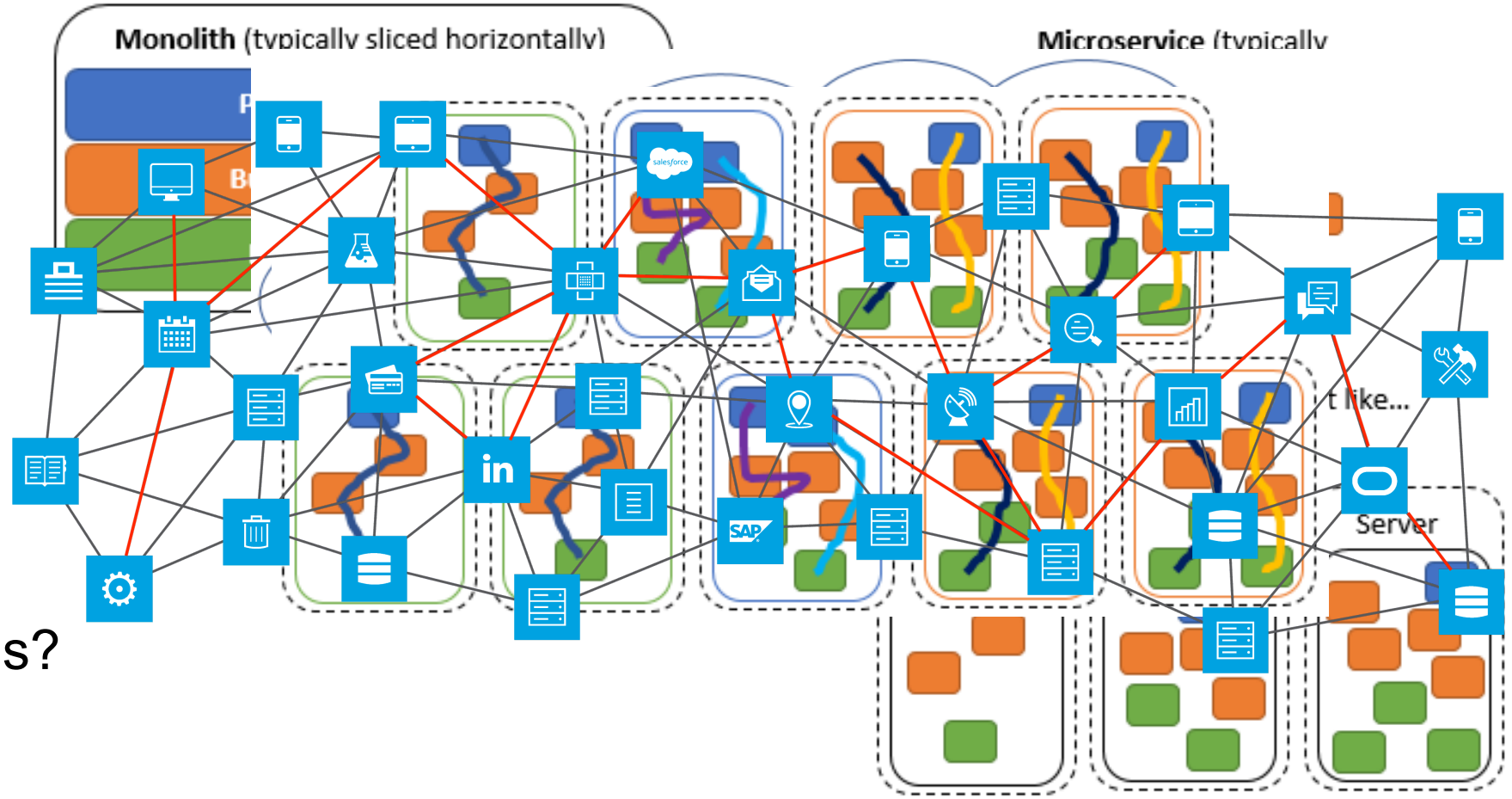
1. App Dev Flashback & Future
2. Observability vs. Monitoring
3. Distributed Tracing
4. Open Tracing
5. Zipkin + Splunk
6. Demo

Let's Start at the Beginning

Containers, Microservices and Serverless…Oh my!

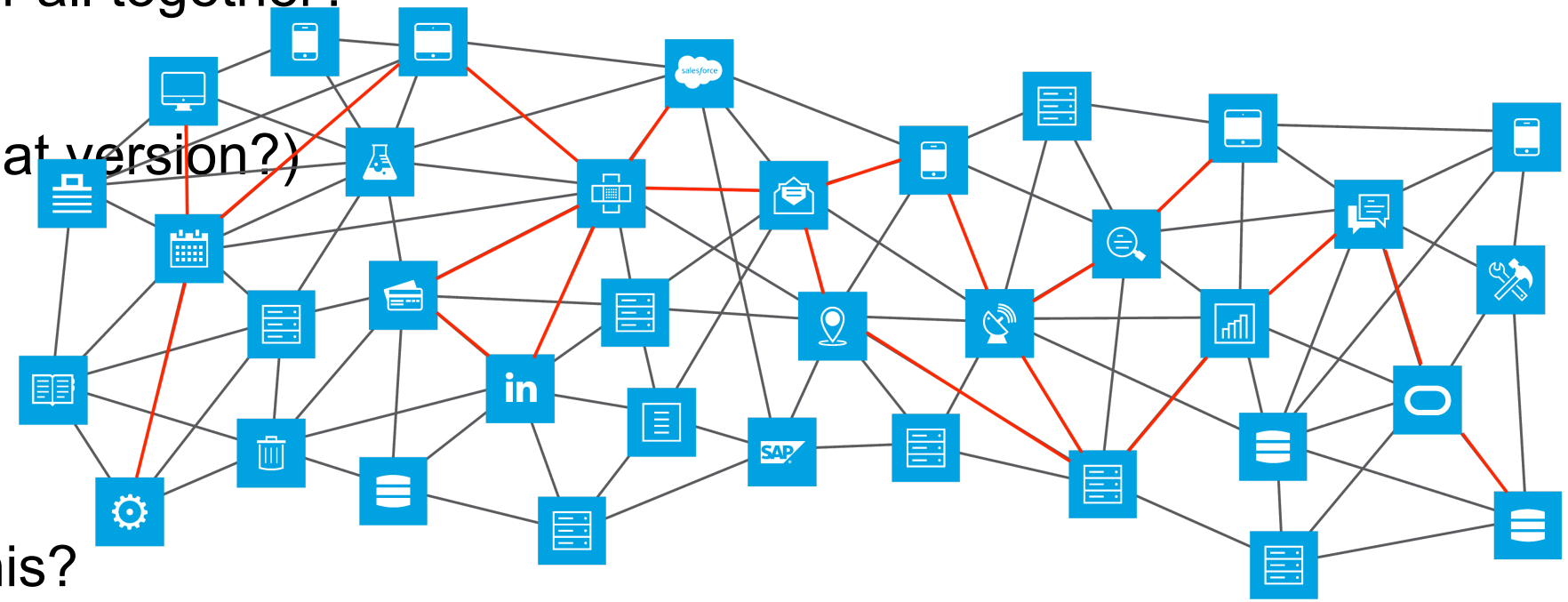
- Monolith apps
- Distributed apps
- Microservices
- Serverless
- Transactions
- External Systems?

- 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

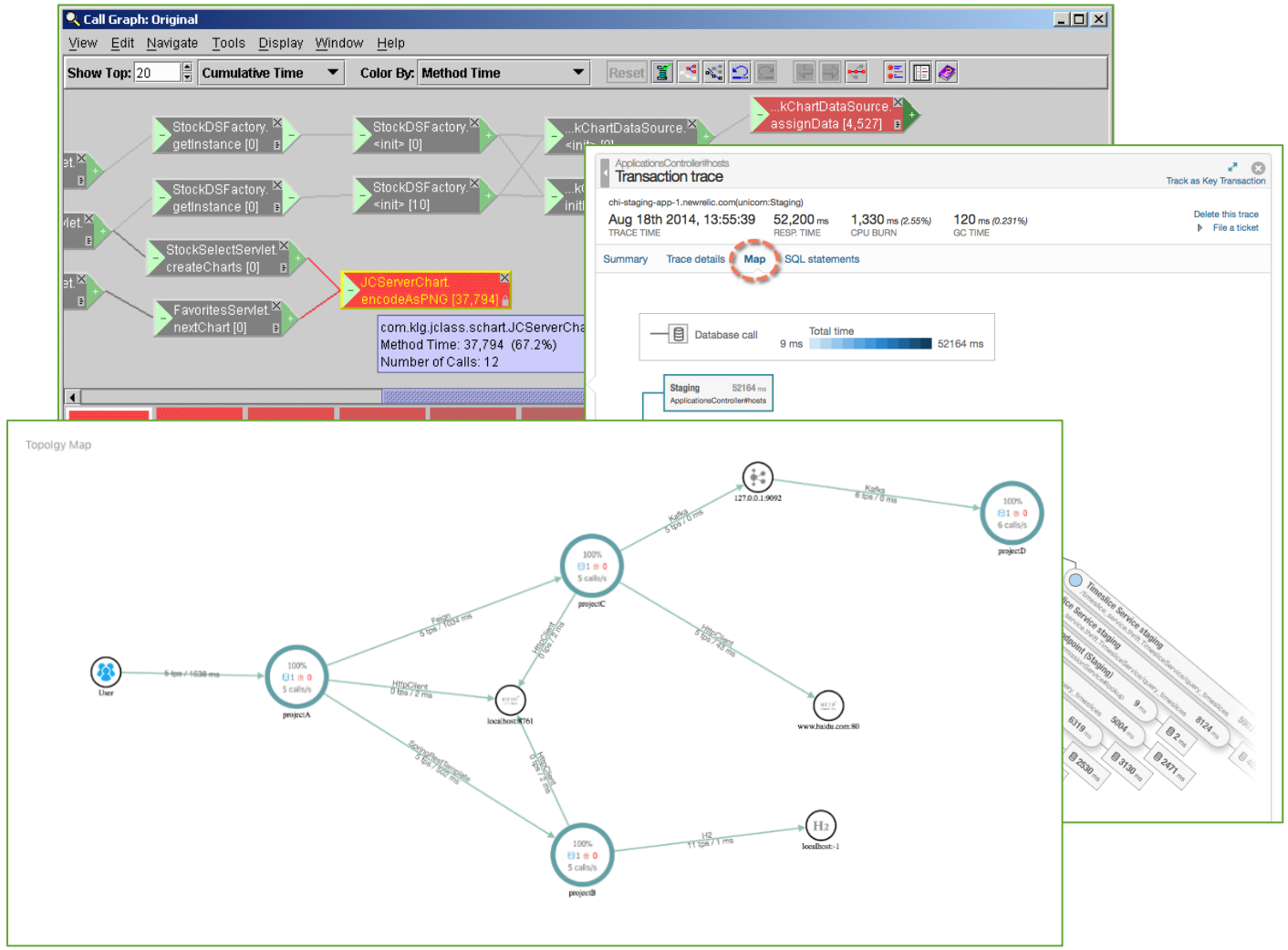
- Logs & Metrics
- Passive (poll)
  - '*I observe you*'
- Measure machines, networks, apps
- Overall Health
- Use Cases
  - Monitoring
  - Alerting
  - Troubleshooting
  - Capacity Planning

Observability

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

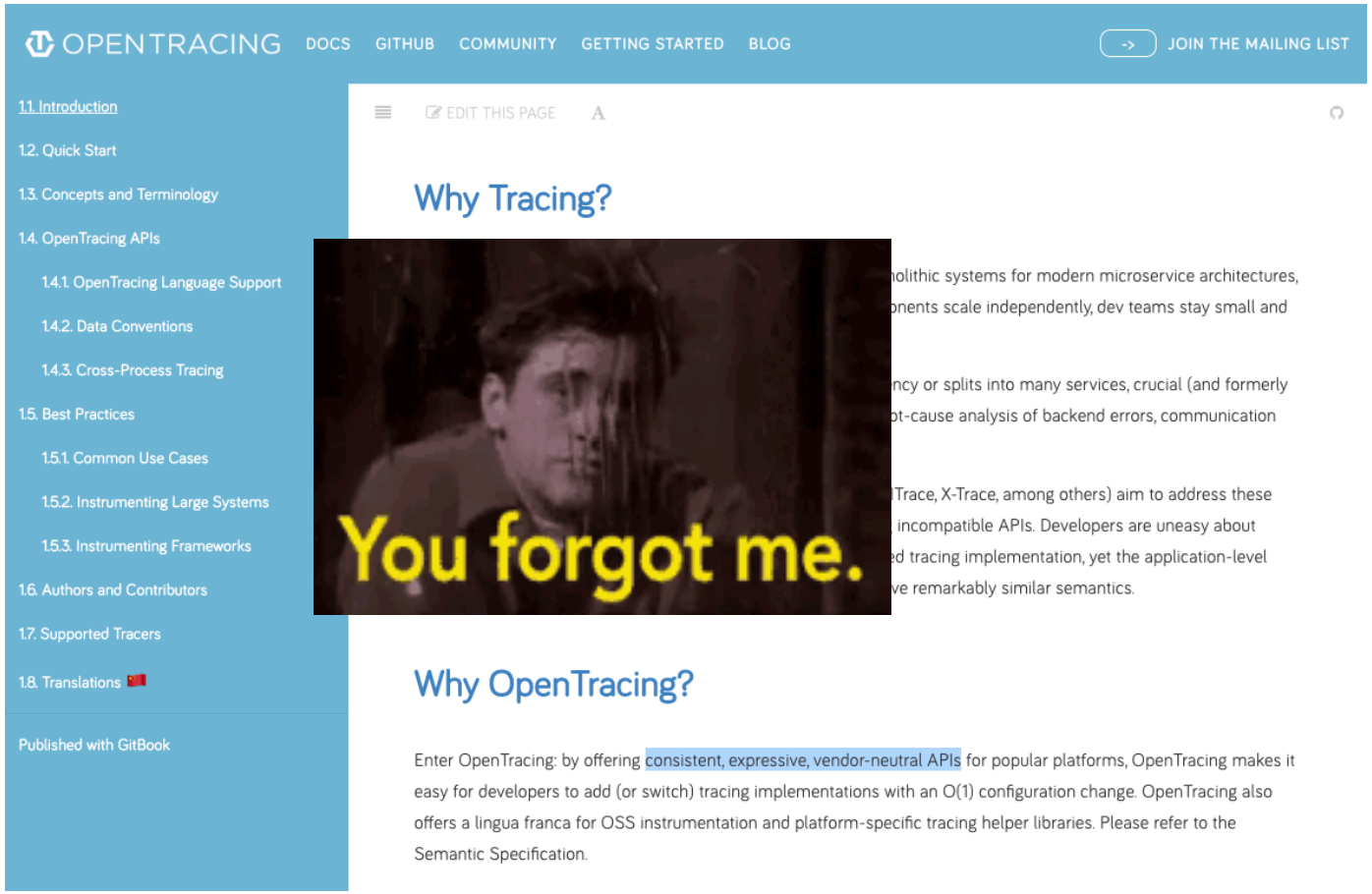
Distributed Tracing

- 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

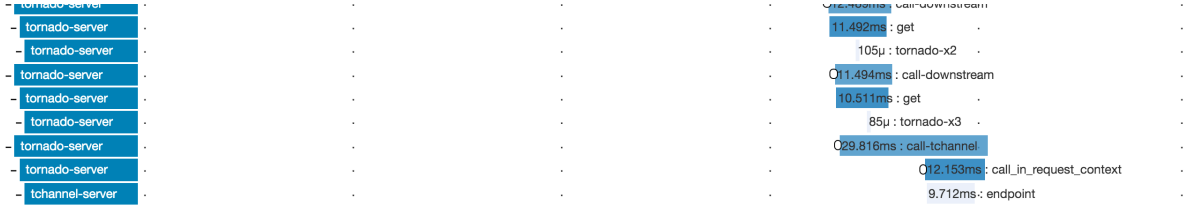
- In the beginning…
  - There was Dapper (2010)
- Great if you're Google ☹
  - Consistent hardware
  - Application templates
  - Zero involvement to instrument
- Enter OpenTracing
  - Consistent, vendor-neutral APIs
  - Multiple implementations
  - Plug-n-Play
- Zipkin, Jaeger, Lightstep



Zipkin?

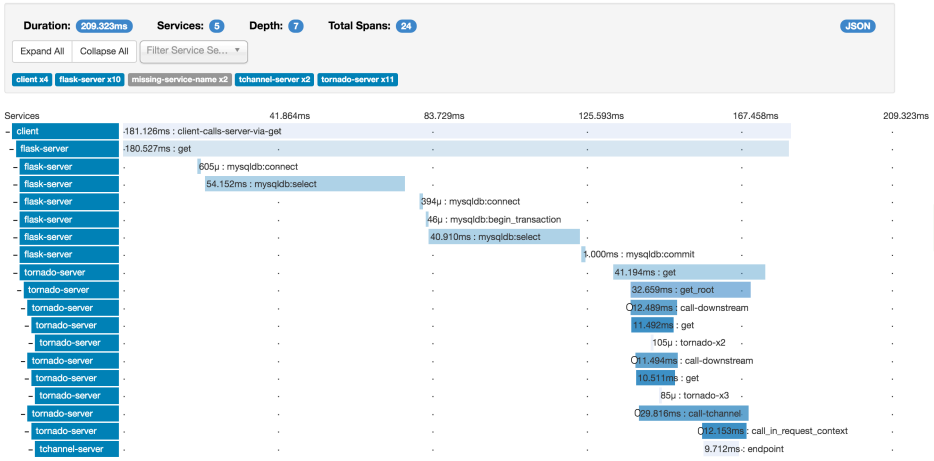
An OpenTracing Implementaion

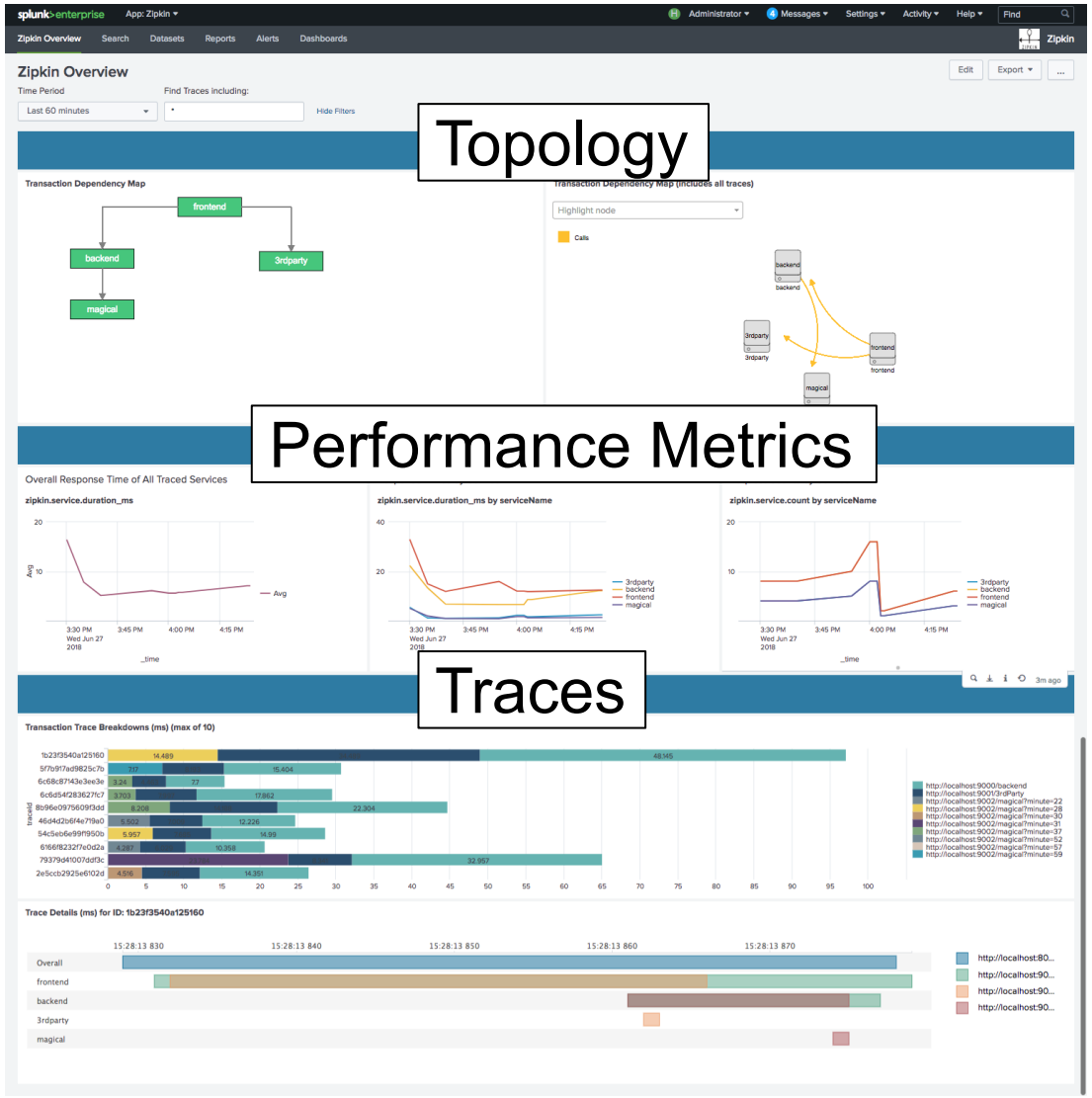
ZIPKIN



splunk> .conf18

Zipkin OpenTracing with Splunk!





splunk> .conf18

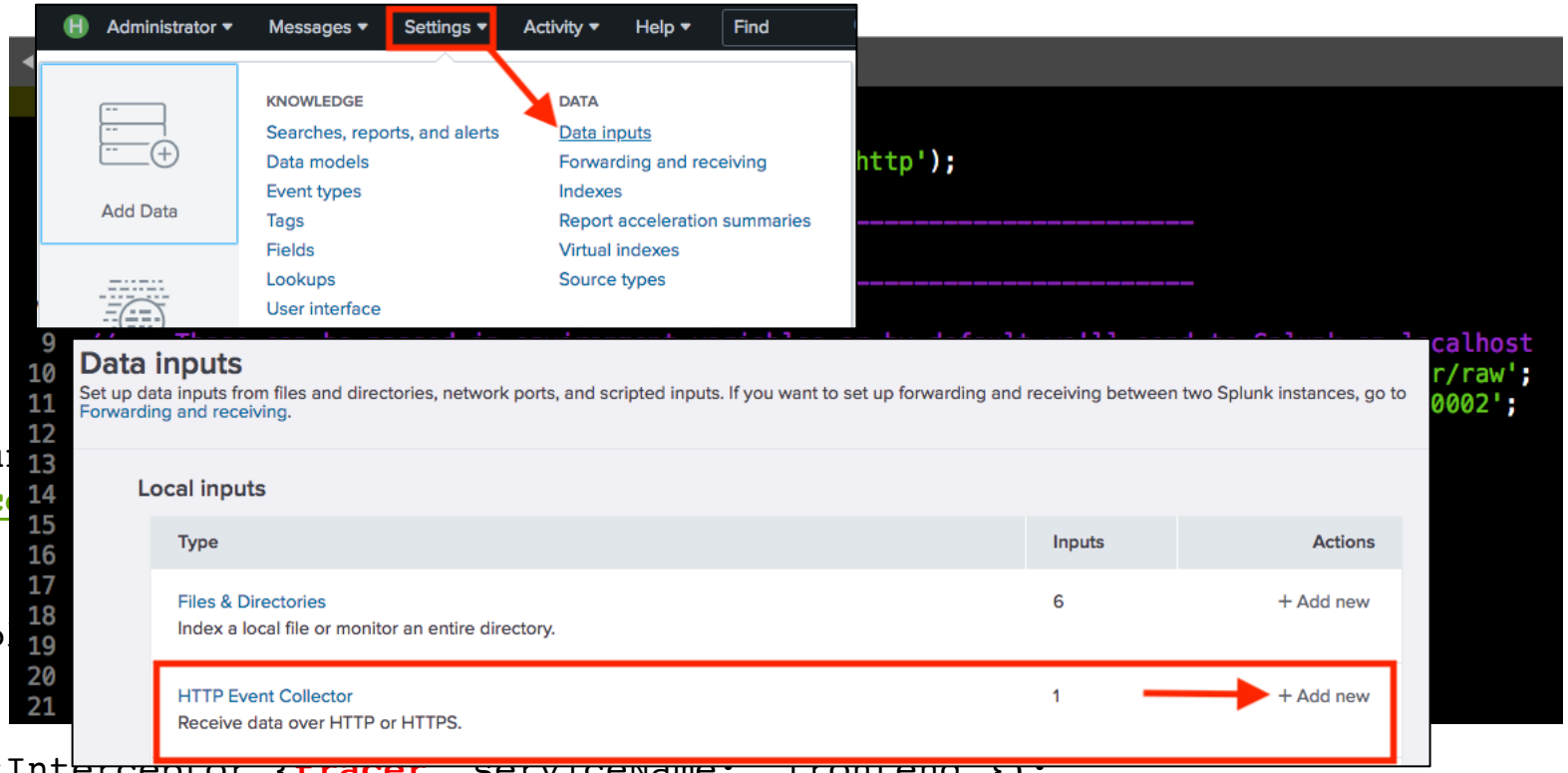
OK, But How?

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

```
// Create an instance of the Splu
const {recorder} = require('./rec

// initialize tracer
const tracer = new Tracer({ctxImp

// instrument the client
const zipkinRest = rest.wrap(restInterceptor,{tracer, serviceName: 'frontend'});
```

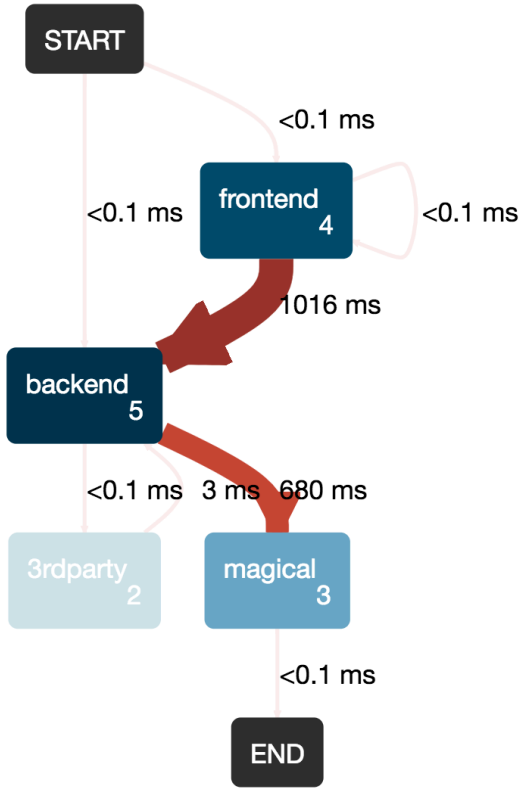


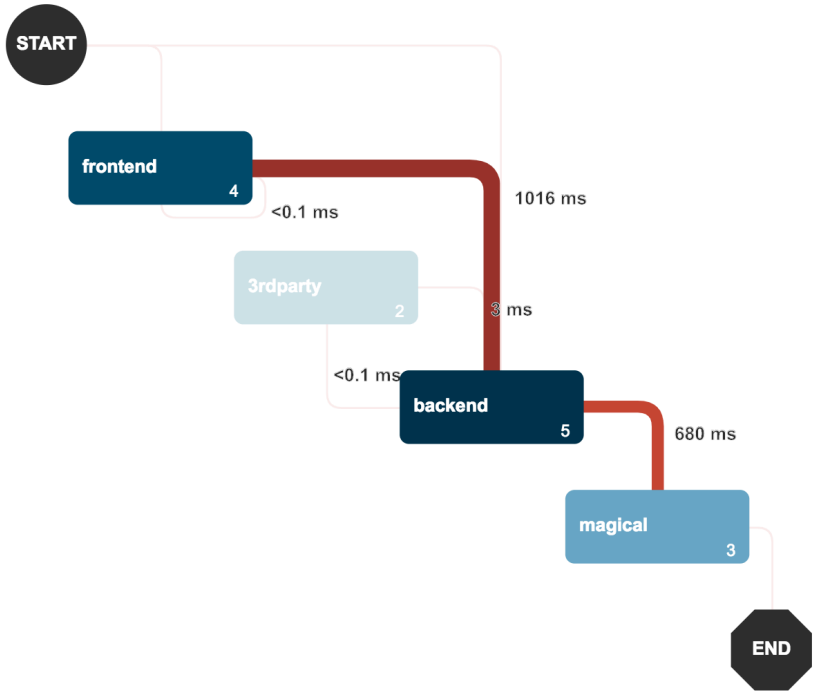
Demo!

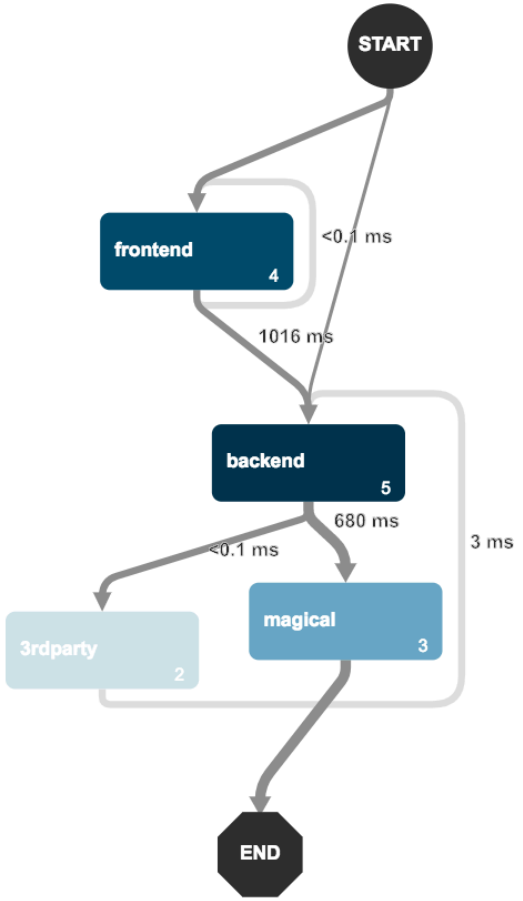
Splunk + Zipkin

More Visualizations…

😉 More on this tomorrow 😉







splunk> .conf18

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