Pivotal

How to Properly Blame Things for Causing Latency

- An Introduction to Distributed Tracing and Zipkin

@Adrian Cole

works at Pivotal

works on Zipkin

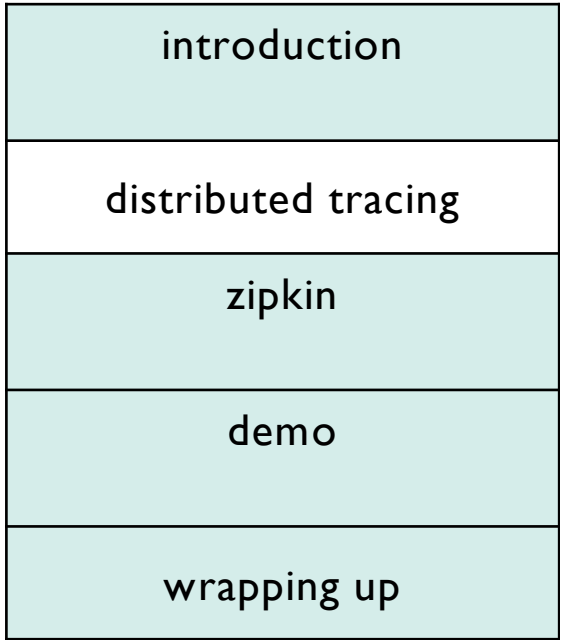
Introduction

introduction
understanding latency
distributed tracing
zipkin
demo
wrapping up

@adrianfcole

spring cloud at pivotal
focused on distributed tracing
helped open zipkin

Distributed Tracing



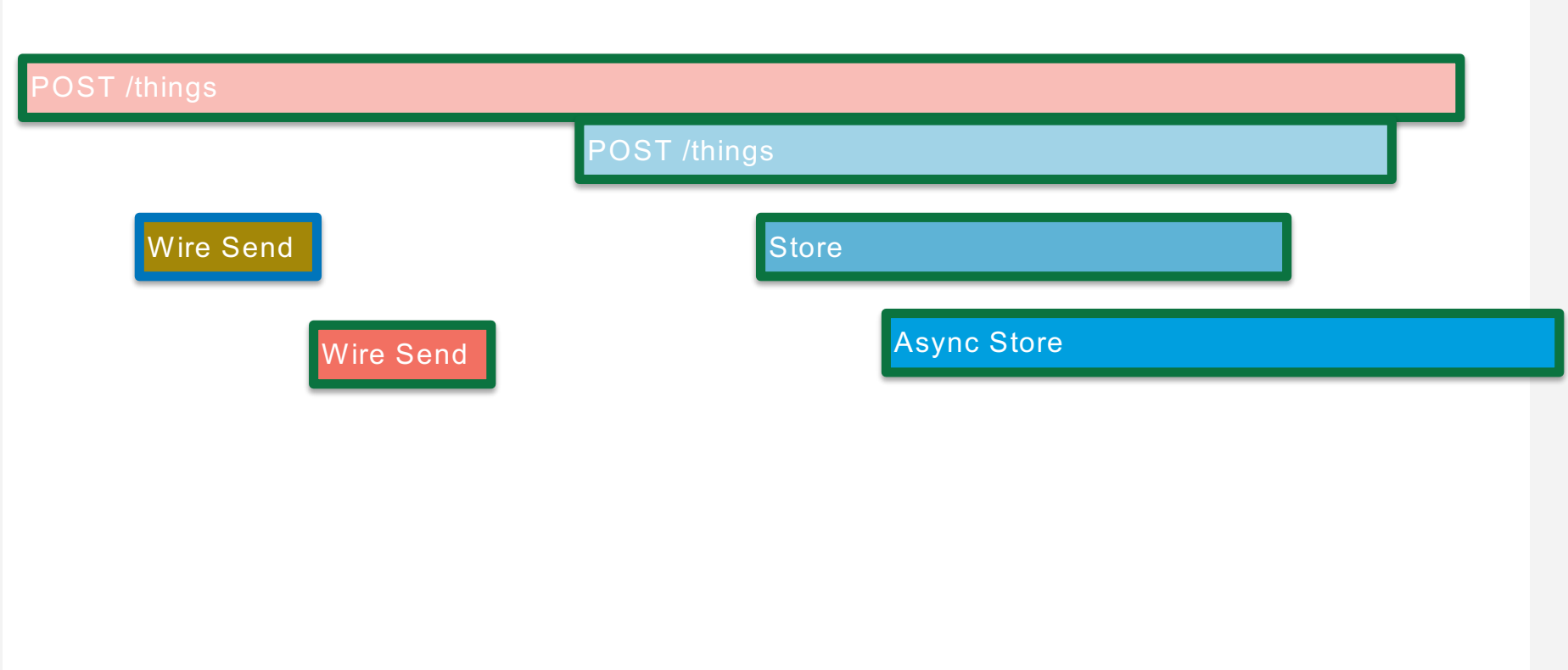
What is Distributed Tracing?

Distributed tracing tracks production requests as they touch different parts of your architecture.

Requests have a unique trace ID, which you can use to lookup a trace diagram, or log entries related to it.

Causal diagrams are easier to understand than scrolling through logs.

Example Trace Diagram



Why do I care?

- **Reduce time in triage** by contextualizing errors and delays
- **Visualize latency** like time in my service vs waiting for other services
- **Understand complex applications** like async code or microservices
- **See your architecture** with live dependency diagrams built from traces

Example Service Diagram

A tracing system can draw your service dependencies!

It might resemble your favorite noodle dish!

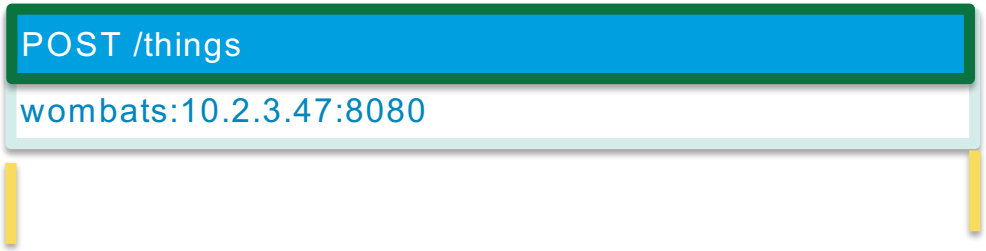
Distributed Tracing Vocabulary

A **Span** is primarily the duration of an operation.

A **Trace** links all spans in a request together by cause.

A Span is an individual operation

Operation



Events

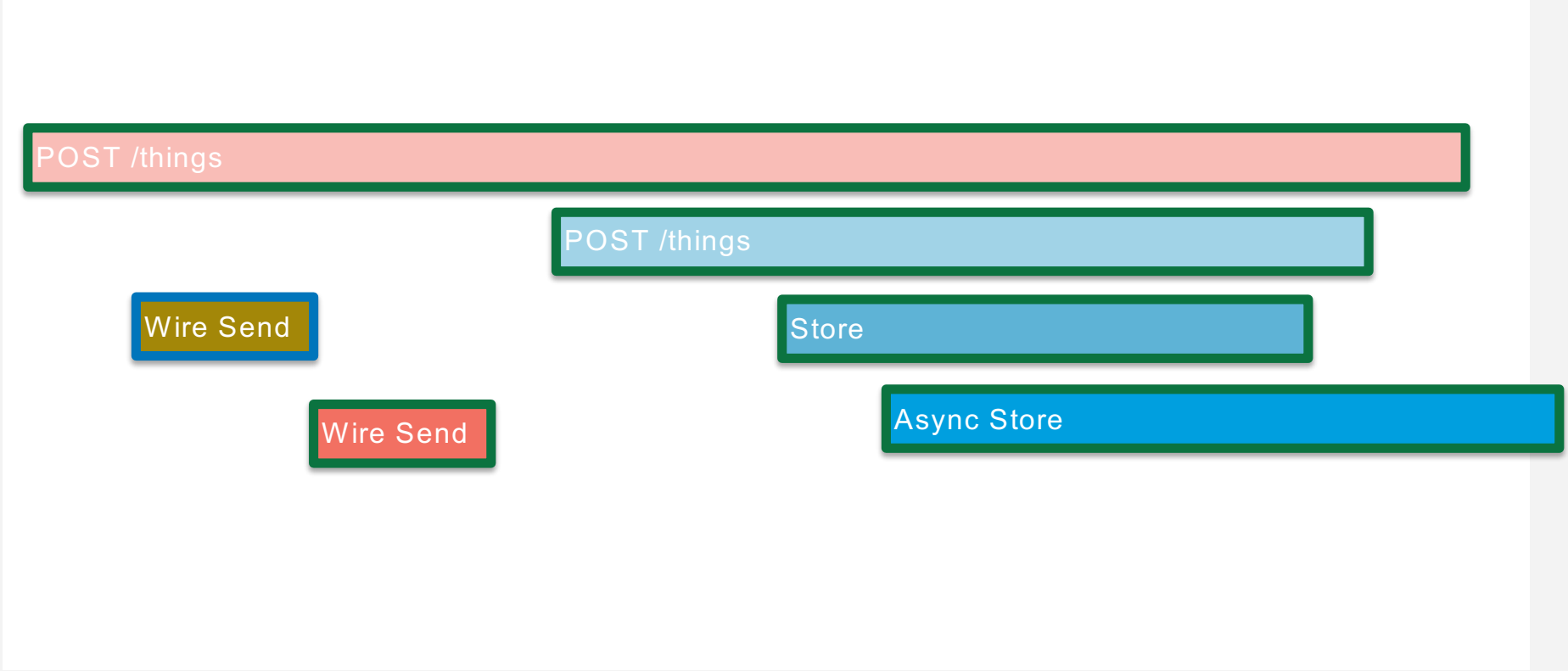
Server Received a Request

Server Sent a Response

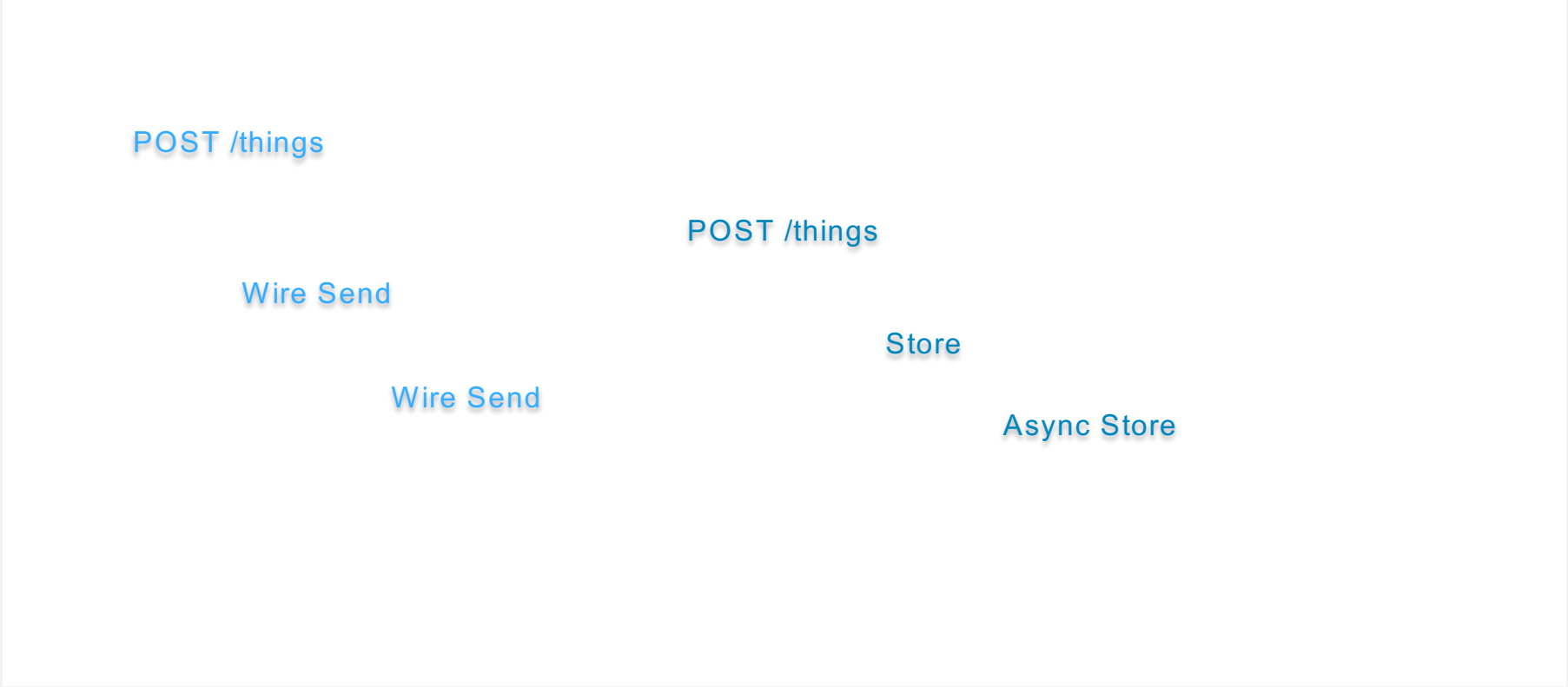
Tags

remote.ipv4	1.2.3.4
http.request-id	abcd-ffe
http.request.size	15 MiB
http.url	...&features=HD-uploads

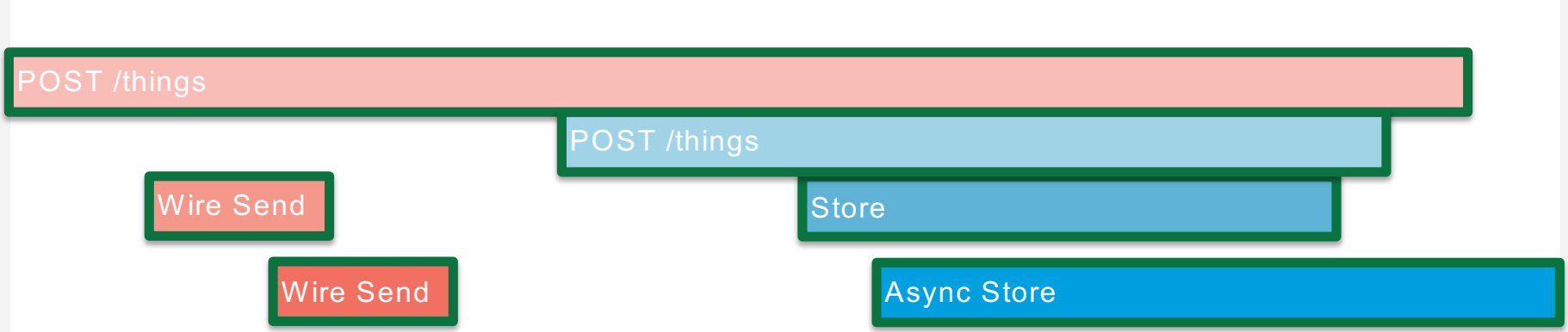
Trace shows each operation the request caused



Tracing is capturing important events



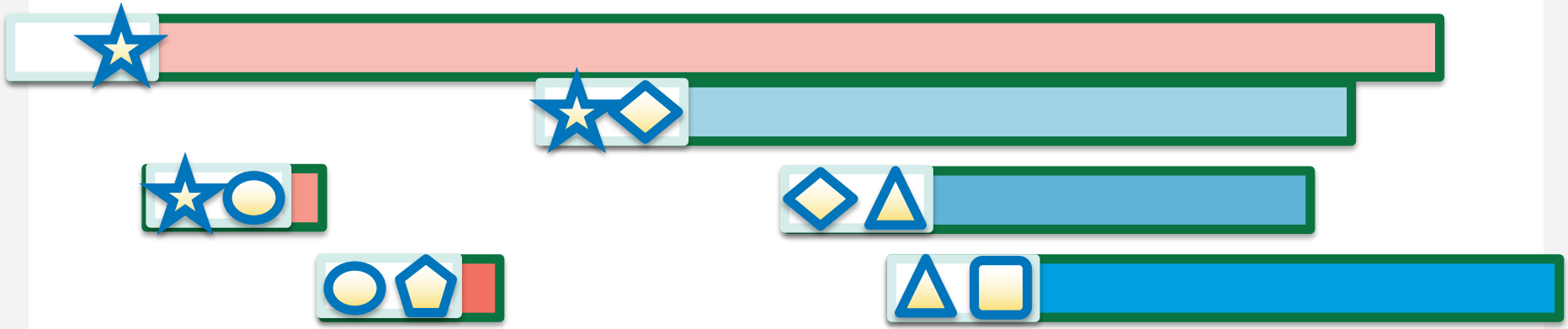
Tracers record time, duration and host



Tracers don’t decide what to record, instrumentation does.. we’ll get to that

Tracers send trace data out of process

Tracers propagate IDs in-band,
to tell the receiver there's a trace in progress



Completed spans are reported out-of-band,
to reduce overhead and allow for batching

Tracer vs Instrumentation

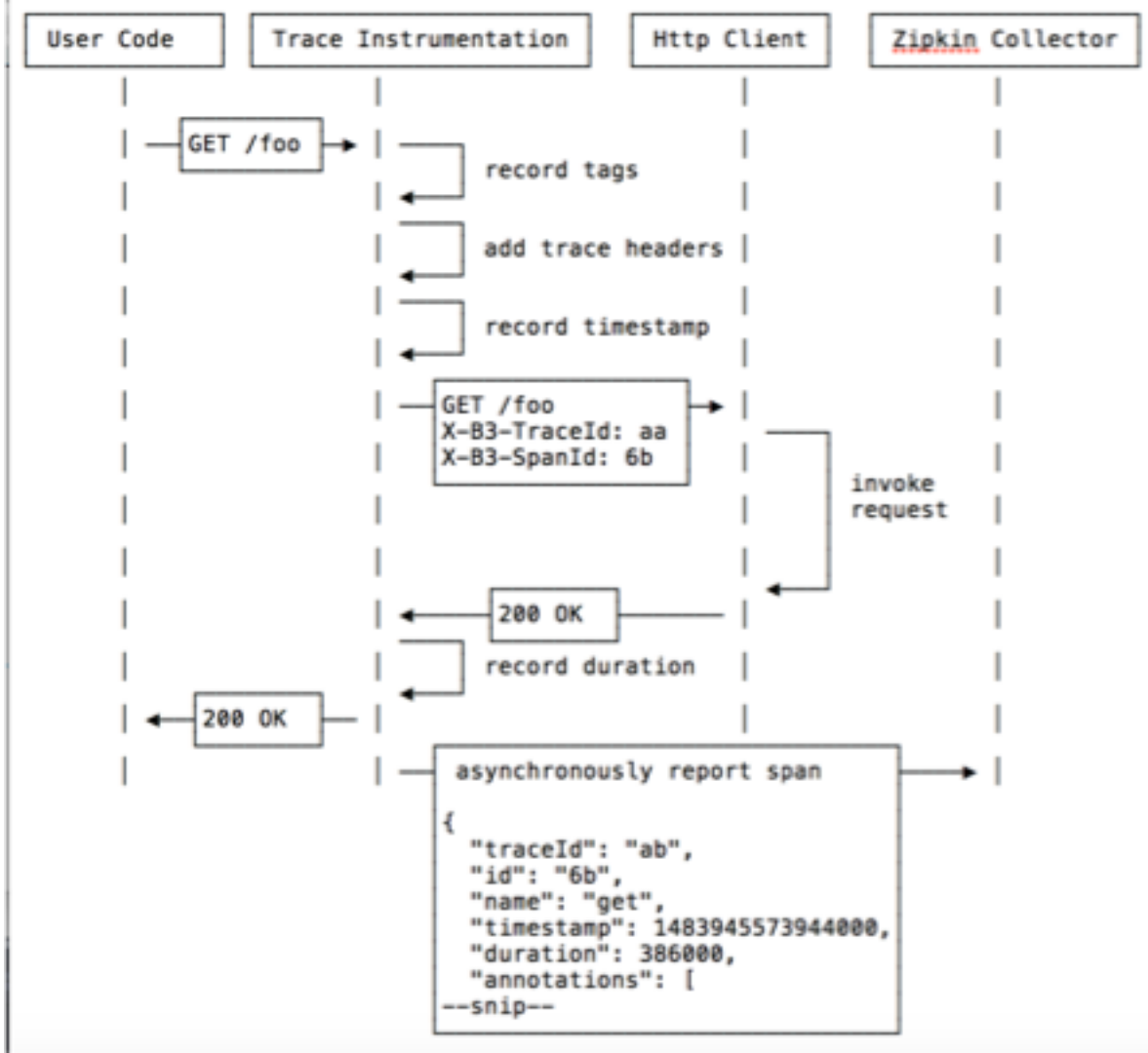
A tracer is a utility library similar to metrics or logging libraries.

Instrumentation is framework code that uses a tracer to collect details such as the http url and request timing.

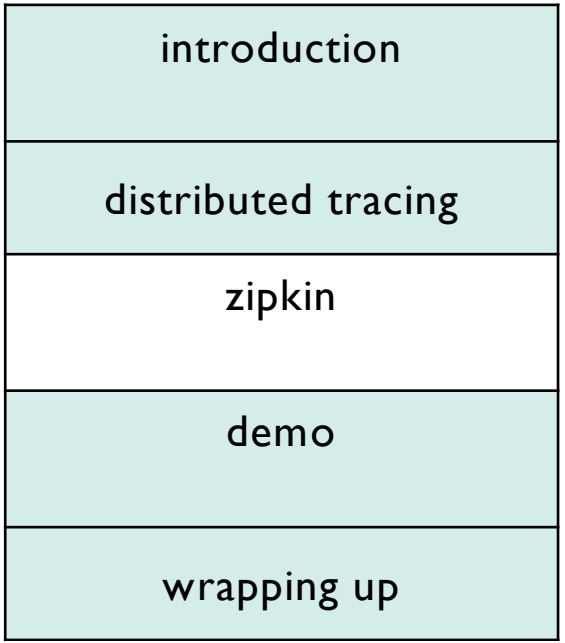
Instrumentation is usually invisible to users

Instrumentation decides what to record

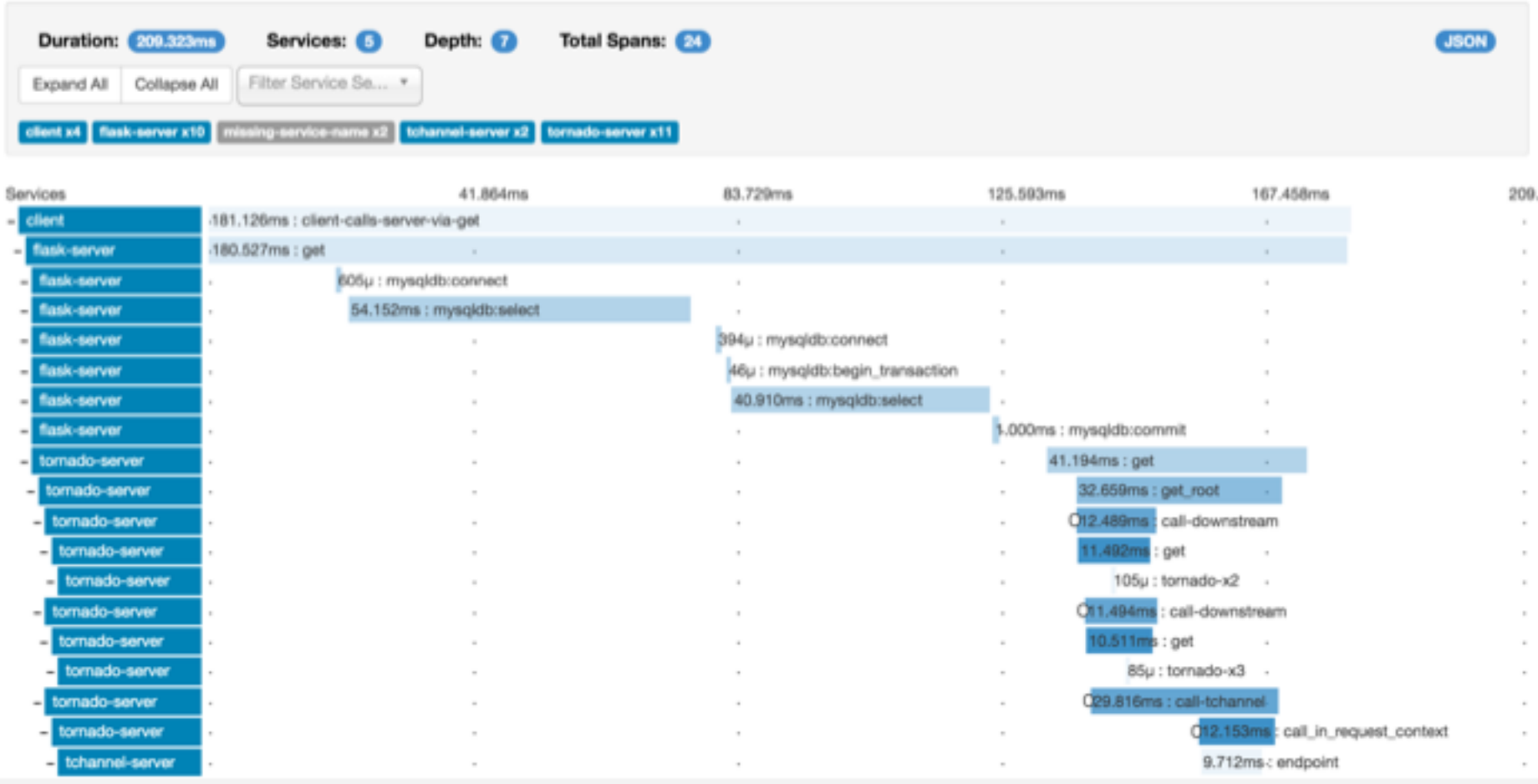
Instrumentation decides how to propagate state



Zipkin


introduction
distributed tracing
zipkin
demo
wrapping up

Zipkin is a distributed tracing system



Zipkin lives in GitHub

Zipkin was created by Twitter in 2012 based on the Google Dapper paper. In 2015, OpenZipkin became the primary fork.

OpenZipkin is an org on GitHub. It contains tracers, OpenApi spec, service components and docker images.

https://github.com/openzipkin

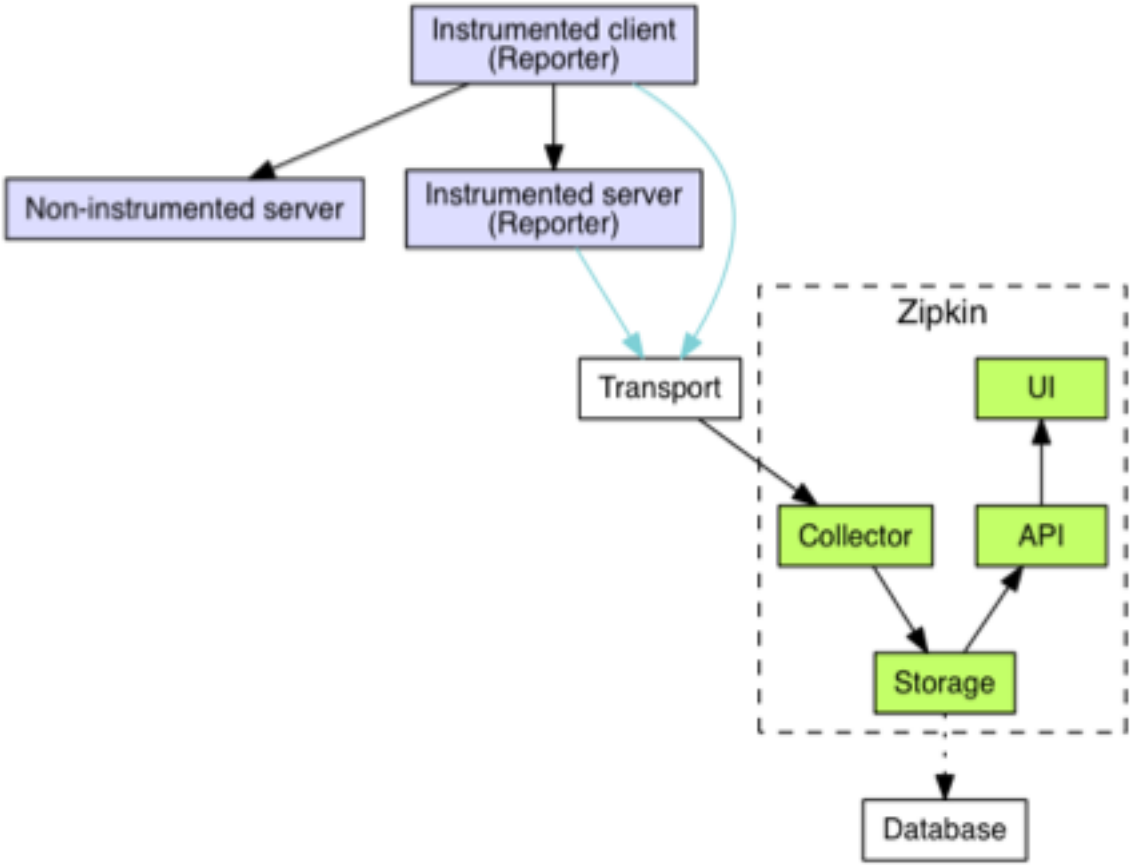
Zipkin Architecture

Tracers **report** spans HTTP or Kafka.

Servers **collect** spans, storing them in MySQL, Cassandra, or Elasticsearch.

Users **query** for traces via Zipkin's Web UI or Api.

Amazon
Azure
Docker
Google
Kubernetes
Mesos
Spark



Zipkin has starter architecture

Tracing is new for a lot of folks.

For many, the MySQL option is a good start, as it is familiar.

```
services:
  storage:
    image: openzipkin/zipkin-mysql
    container_name: mysql
    ports:
      - 3306:3306
  server:
    image: openzipkin/zipkin
    environment:
      - STORAGE_TYPE=mysql
      - MYSQL_HOST=mysql
    ports:
      - 9411:9411
    depends_on:
      - storage
```

Zipkin can be as simple as a single file

```
$ curl -SL 'https://search.maven.org/remote_content?g=io.zipkin.java&a=zipkin-server&v=LATEST&c=exec' > zipkin.jar
$ SELF_TRACING_ENABLED=true java -jar zipkin.jar
```

```
:: Powered by Spring Boot ::          (v1.5.4.RELEASE)

2016-08-01 18:50:07.098  INFO 8526 --- [           main] zipkin.server.ZipkinServer               : Starting ZipkinServer on acole with PID 8526 (/Users/acole/oss/sleuth-webmvc-
example/zipkin.jar started by acole in /Users/acole/oss/sleuth-webmvc-example)
-snip-
```

```
$ curl -s localhost:9411/api/v2/services|jq .
[
  "gateway"
]
```

Brave: the most popular Zipkin Java tracer

- **Brave** - OpenZipkin's java library and instrumentation
  - Layers under projects like Armeria, Dropwizard, Play

- **Spring Cloud Sleuth** - automatic tracing for Spring Boot
  - Includes many common spring integrations
  - Starting in version 2, Sleuth is a layer over Brave!

c, c#, erlang, javascript, go, php, python, ruby, too

Some notable open source tracing libraries

- **OpenCensus** - Observability SDK (metrics, tracing, tags)
  - Most notably, gRPC's tracing library
  - Includes exporters in Zipkin format and B3 propagation format
- **OpenTracing** - trace instrumentation library api definitions
  - Bridge to Zipkin tracers available in Java, Go and PHP
- **SkyWalking** - APM with a java agent developed in China
  - Work in progress to send trace data to zipkin
- **Kamon** - Ak**Ka Mon**itoring: trace and metrics specializing in scala
  - Uses B3 propagation and has a Zipkin export plugin


Demo

Wrapping up

Pivotal

Transforming How The World Builds Software