Pivotal

Distributed Tracing & Zipkin

Reshmi Krishna

Agenda

- Distributed Tracing
- Zipkin
- Spring Cloud Sleuth Stream
- Roadmap
- Demo

Pivotal

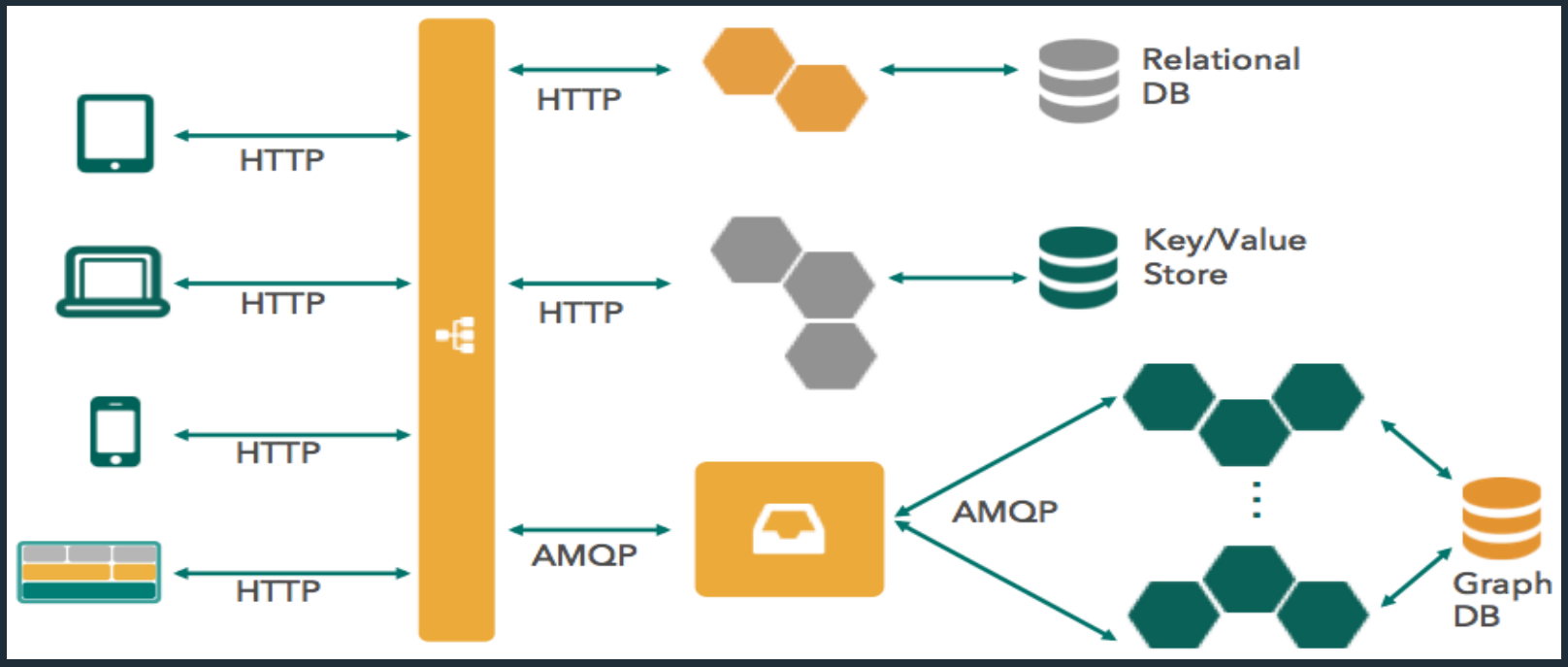
Challenges Of Microservices

- Configuration Management
- Service Registration & Discovery
- Routing & Load Balancing
- Fault Tolerance
- Monitoring
- Latency Analysis and tracing

Pivotal

Distributed Tracing

Microservice architectures are often a graph of components distributed across a network



Pivotal

Dapper Paper By Google

This paper described Dapper, which is Google's production distributed systems tracing infrastructure

Design Goals :

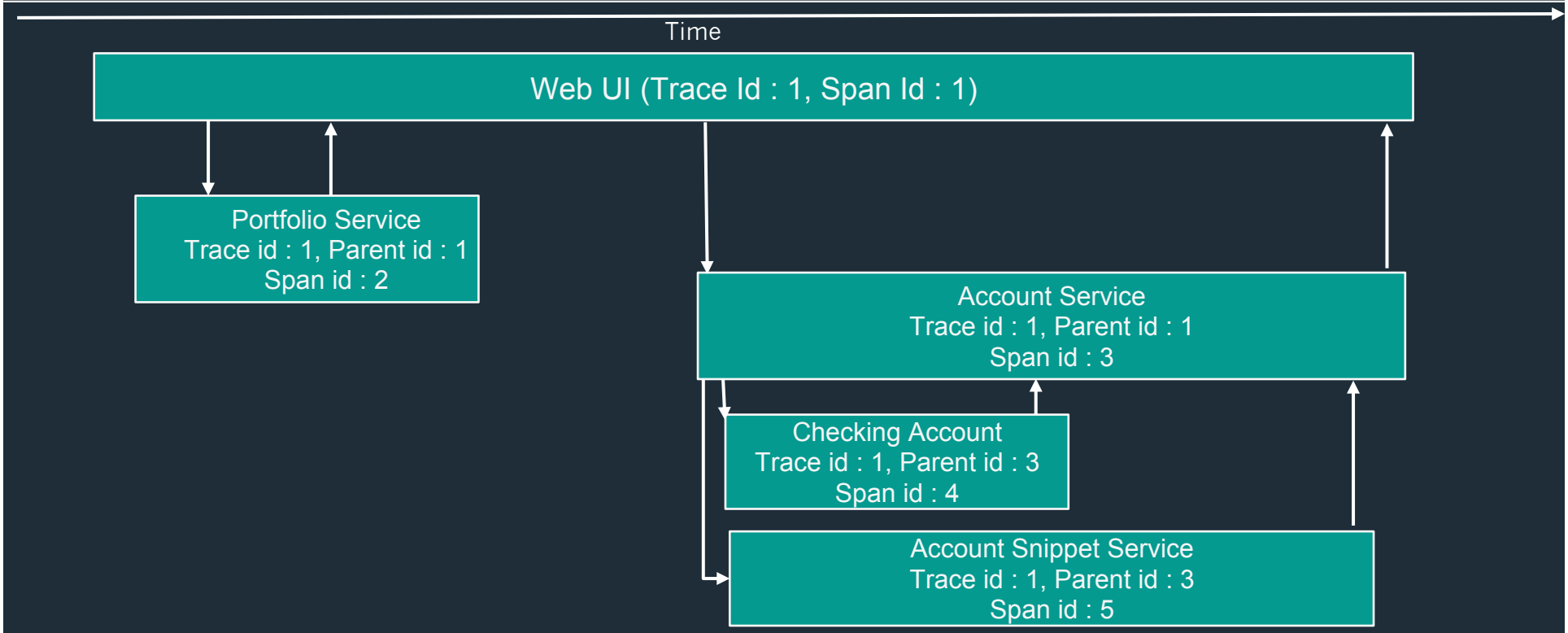
- Low overhead
- Application-level transparency
- Scalability

Based on :

Concept of Traces and Spans

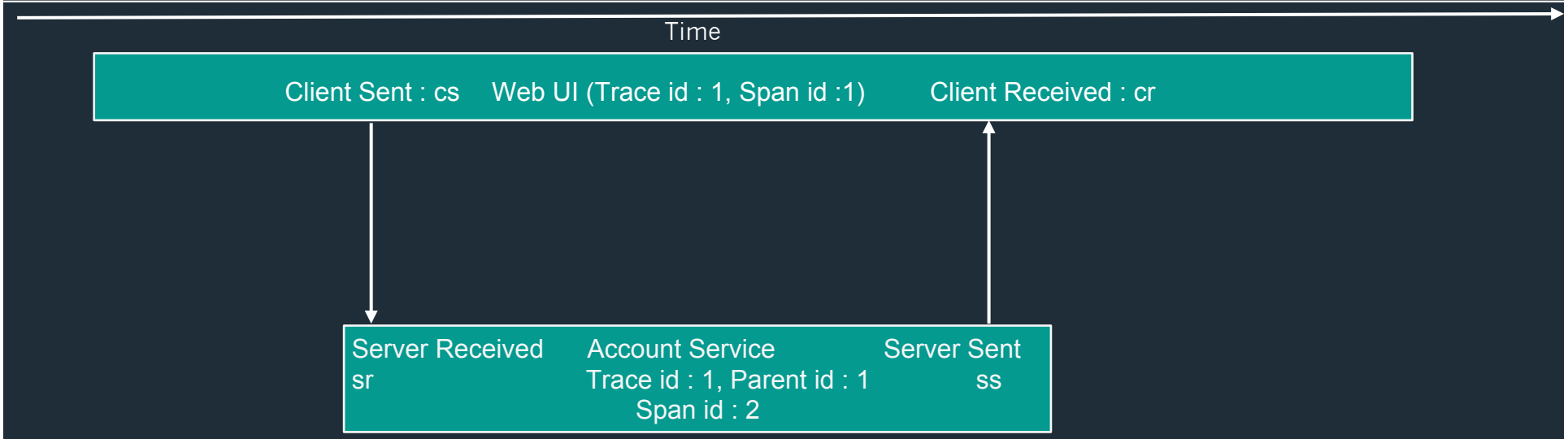
Pivotal

Traces And Spans



Pivotal

Traces And Spans Annotations

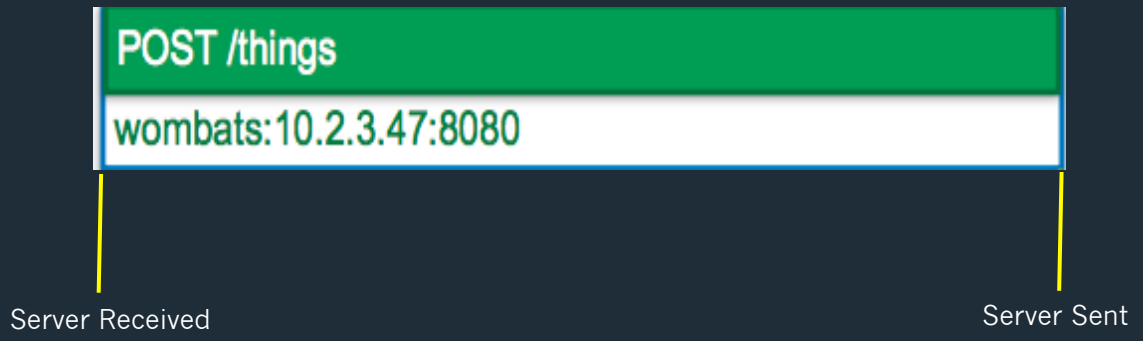


Pivotal

Example Span – An individual operation

Operation

Events



Tags

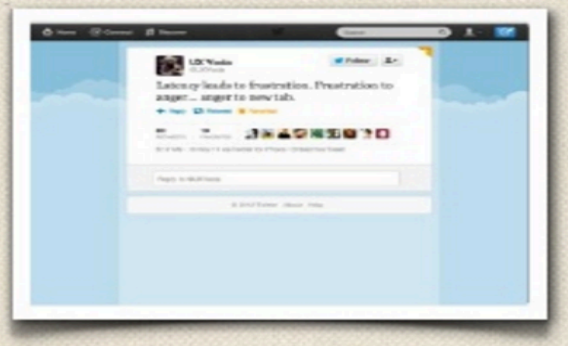
peer.ipv4	1.2.3.4
http.request-id	abcd-ffe
http.request.size	15 MiB
http.url	…&features=HD-uploads

Pivotal

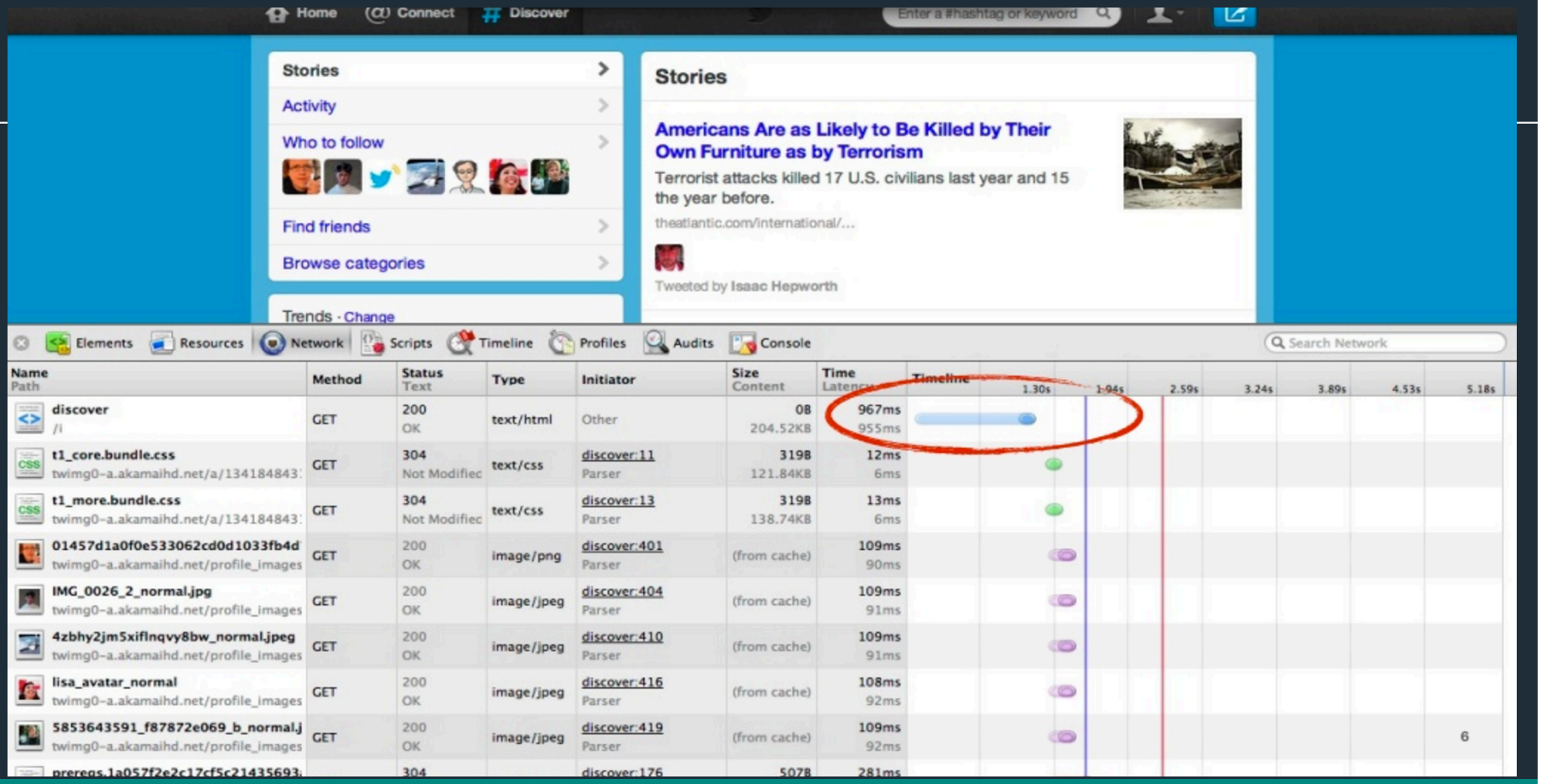
Twitter

Clip slide

Front end



Back end



Pivotal

Zipkin

- Zipkin is a tracing system and was created by Twitter in 2012
- Implementation is based on Dapper Paper by Google
- Adrian Cole (Spring Cloud) is the main contributor
- In 2015, OpenZipkin became the primary fork

Goals :

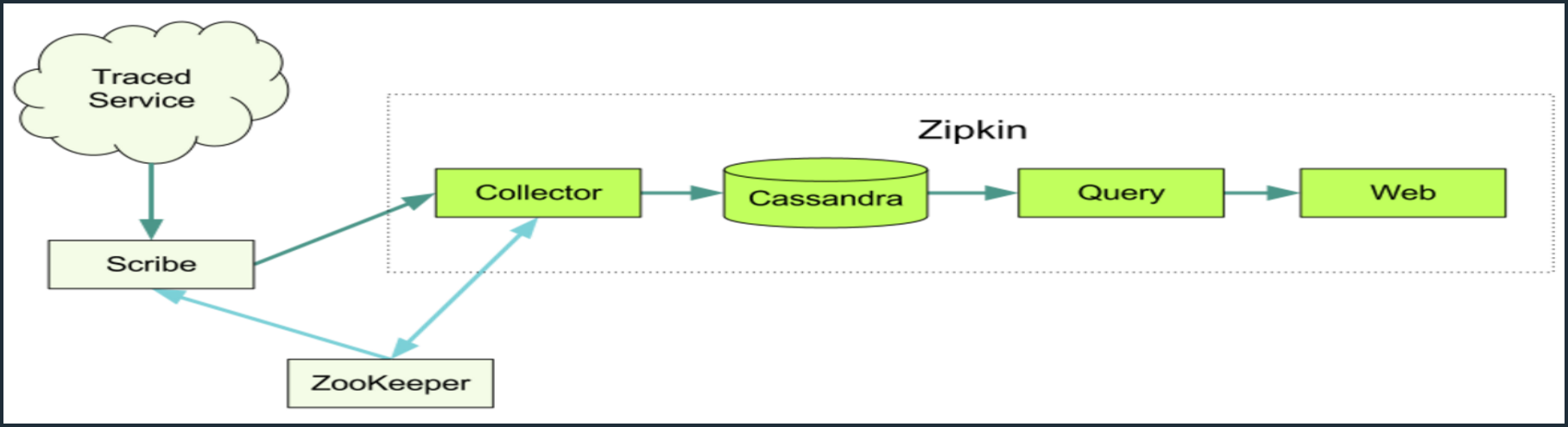
Aggregate spans into trace trees

Provide query and visualization for latency analysis

Have retention policy

Pivotal

Zipkin Architecture



- Tracers collect timing data and transport it over HTTP, RabbitMq, Kafka
- Collectors store spans in MySQL or Cassandra
- Users query for traces via Zipkin's Web UI or Api

Pivotal

Tracers

- **Tracers** add logic to create unique trace ID
- **Trace ID** is generated when the first request is made
- **Span Id** is generated as the request arrives at each microservice
- Example tracer is Spring Cloud Sleuth
- Tracers execute in your production apps! They are written to not log too much
- Tracers have instrumentation or sampling policy to manage volumes of traces and spans

Pivotal

Spring Cloud Sleuth is a Zipkin-Compatible tracer

- Spring Cloud Sleuth sets up useful log formatting for you that prints the trace ID and the span ID
- It includes instrumentation for Spring Boot, and a streaming collector
- Reports to zipkin via HTTP/stream by depending on spring-cloud-sleuth-zipkin / stream
- Transparent to application developer
  1. Include starter BOM on app build system

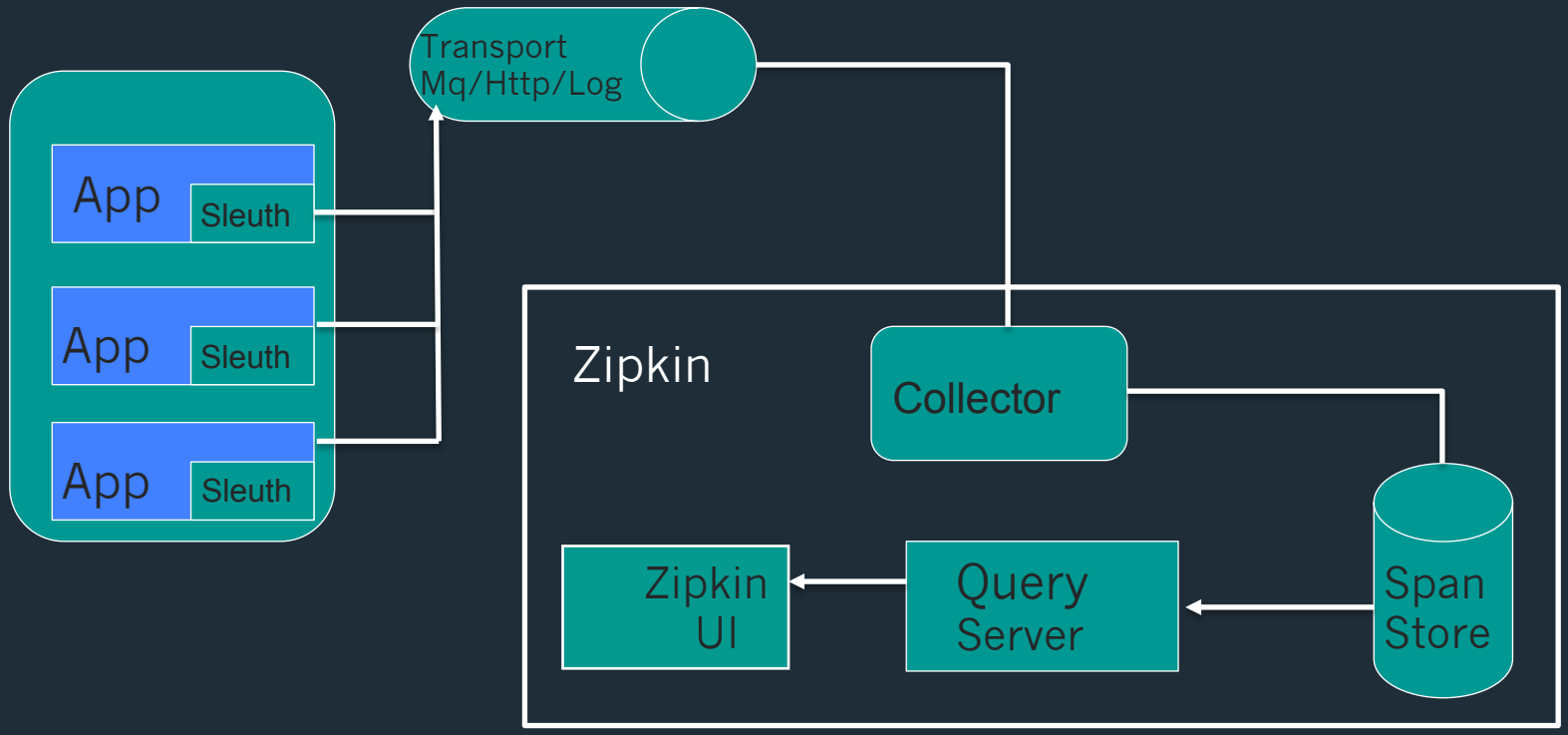
```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-sleuth-stream</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-stream-rabbit</artifactId>
</dependency>
<dependency>
```

  2. Specify “sampler”

```
@Bean
public Sampler sampler(){
return new AlwaysSampler();
}
```

Pivotal

Spring Cloud Sleuth Stream & Zipkin Architecture



Pivotal

Road Map (Distributed Tracing and PCF)

- Zipkin as a PCF Spring Cloud Service expected on Q4
- Apps running on PCF can integrate Spring Cloud Sleuth and bind to Zipkin
- Currently supporting only rabbit MQ, future plans to support kafka
- Future versions might merge with Metrix

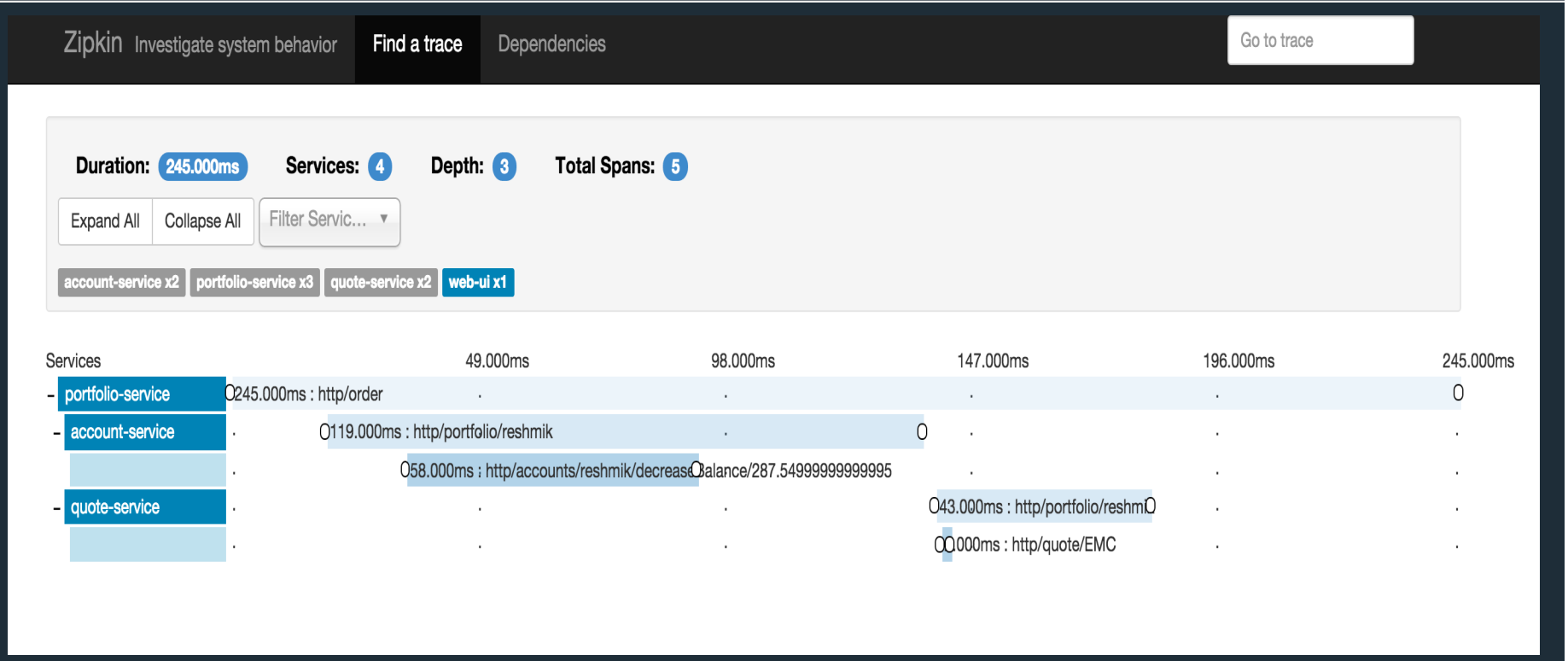
Pivotal

Summary

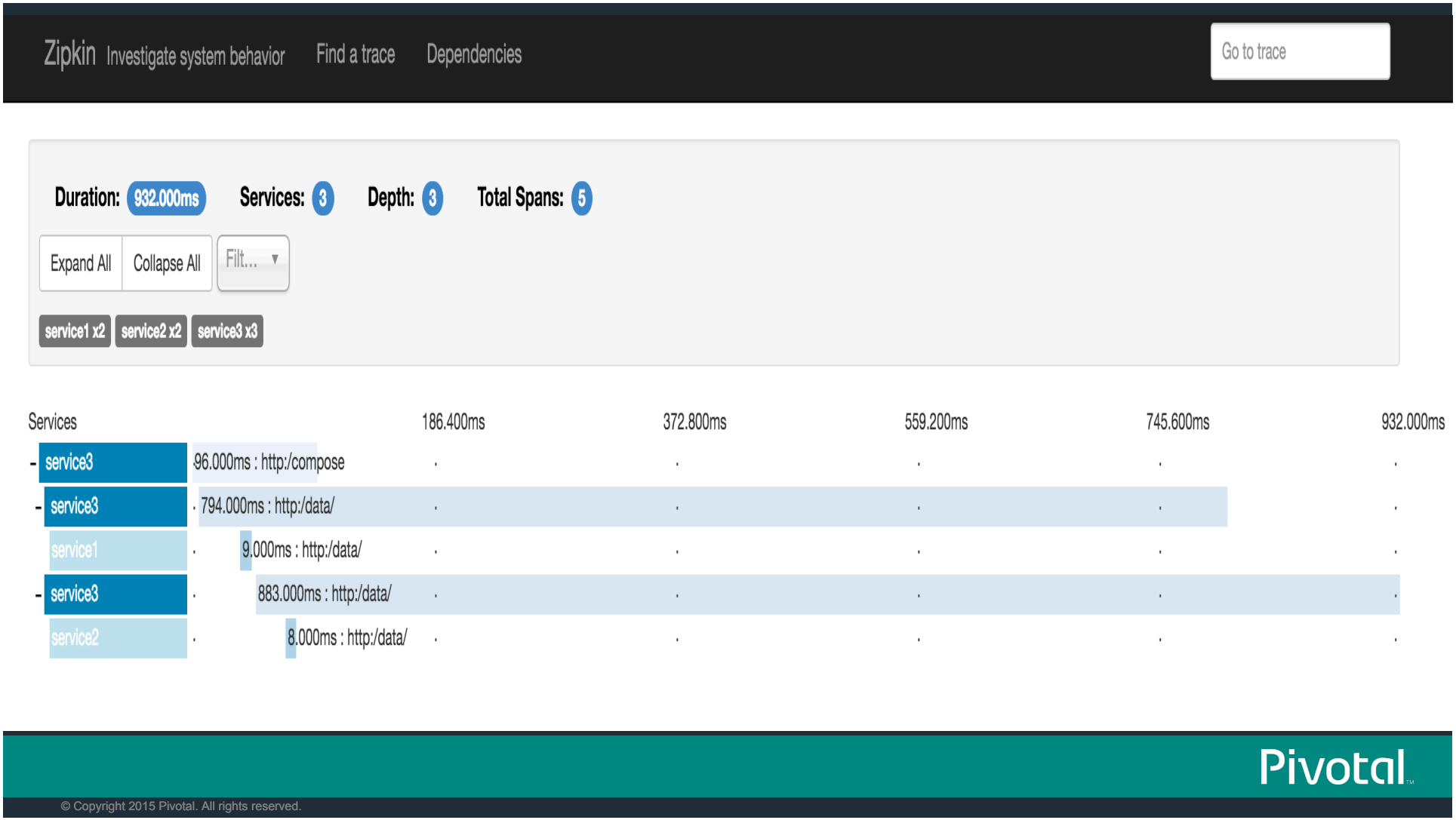
- Tracing requests in a microservice ecosystem is hard
- Distributed tracing to the rescue, by using traces and spans you can trace your requests end-end
- Zipkin is a distributed tracing system that collect, process and present data reported by tracers
- Spring Cloud Sleuth is a Zipkin Compatible tracer

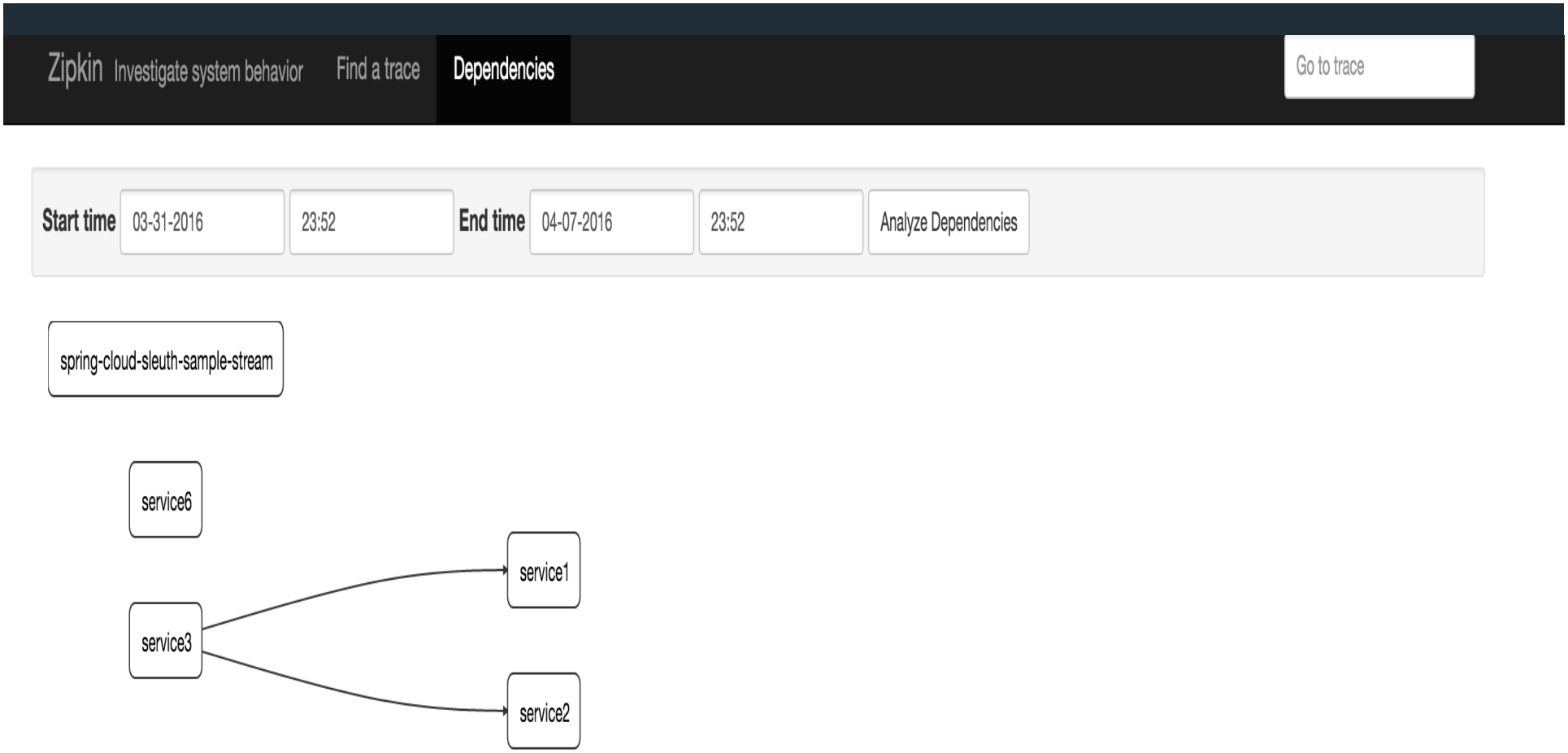
Pivotal

Demo



Pivotal





Resources

SME Content :

Distributed Tracing SME

#sme-pcf-zipkin

Github Content:

spring-cloud-sleuth repo

twitter-zipkin

openzipkin

Dapper Paper By Google

Josh Long's blog : https://spring.io/blog/2016/02/15/distributed-tracing-with-spring-cloud-sleuth-and-spring-cloud-zipkin

Pivotal