Distributed Tracing, OpenTracing & Elastic APM

Adam Quan | Solutions Architect

April 16th, 2019

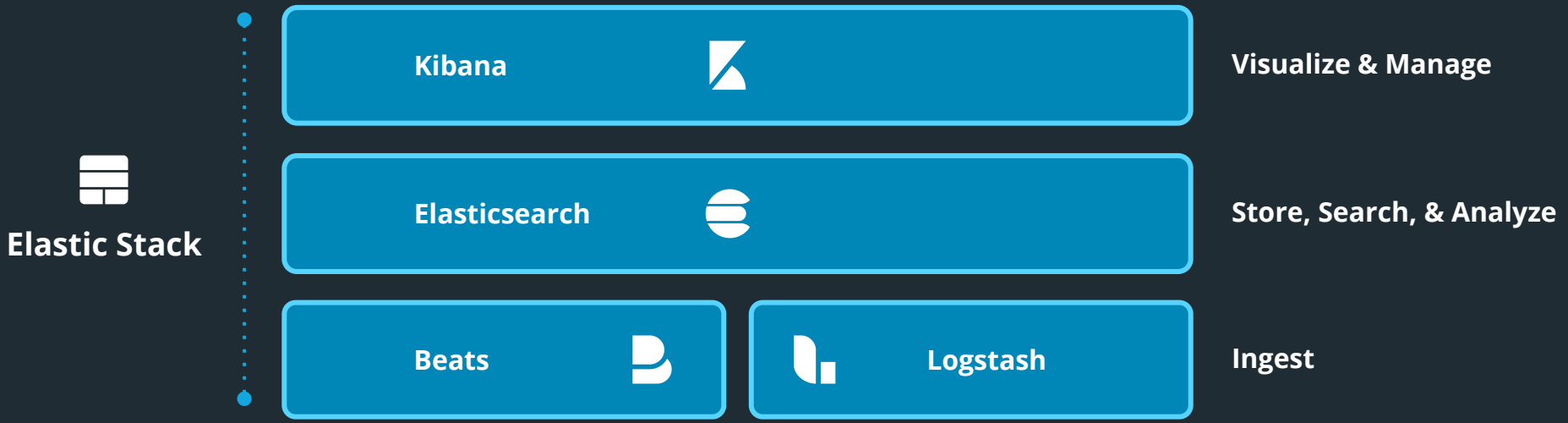
Adam Quan

Solutions Architect, Elastic Certified Engineer

Elastic

Housekeeping & Logistics

- **Recording** will be available following the webinar
- Chat via IRC #elastic-webinar
  - #elastic-webinar @ Freenode
  - Click "Join the Chat" link, create an IRC account
- Please select high resolution in the YouTube video player


Elastic Stack
Kibana
Visualize & Manage
Elasticsearch
Store, Search, & Analyze
Beats
Logstash
Ingest

Technology **differentiation**

SCALE

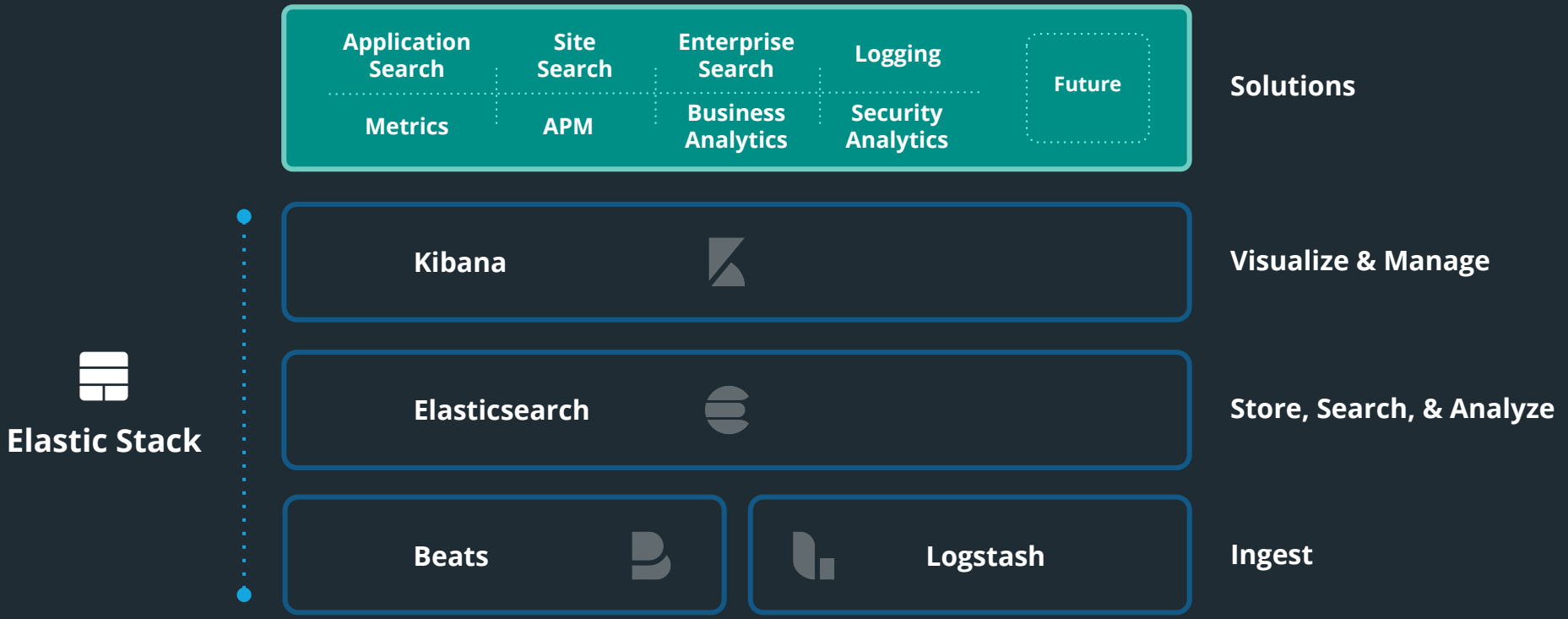
Distributed by design

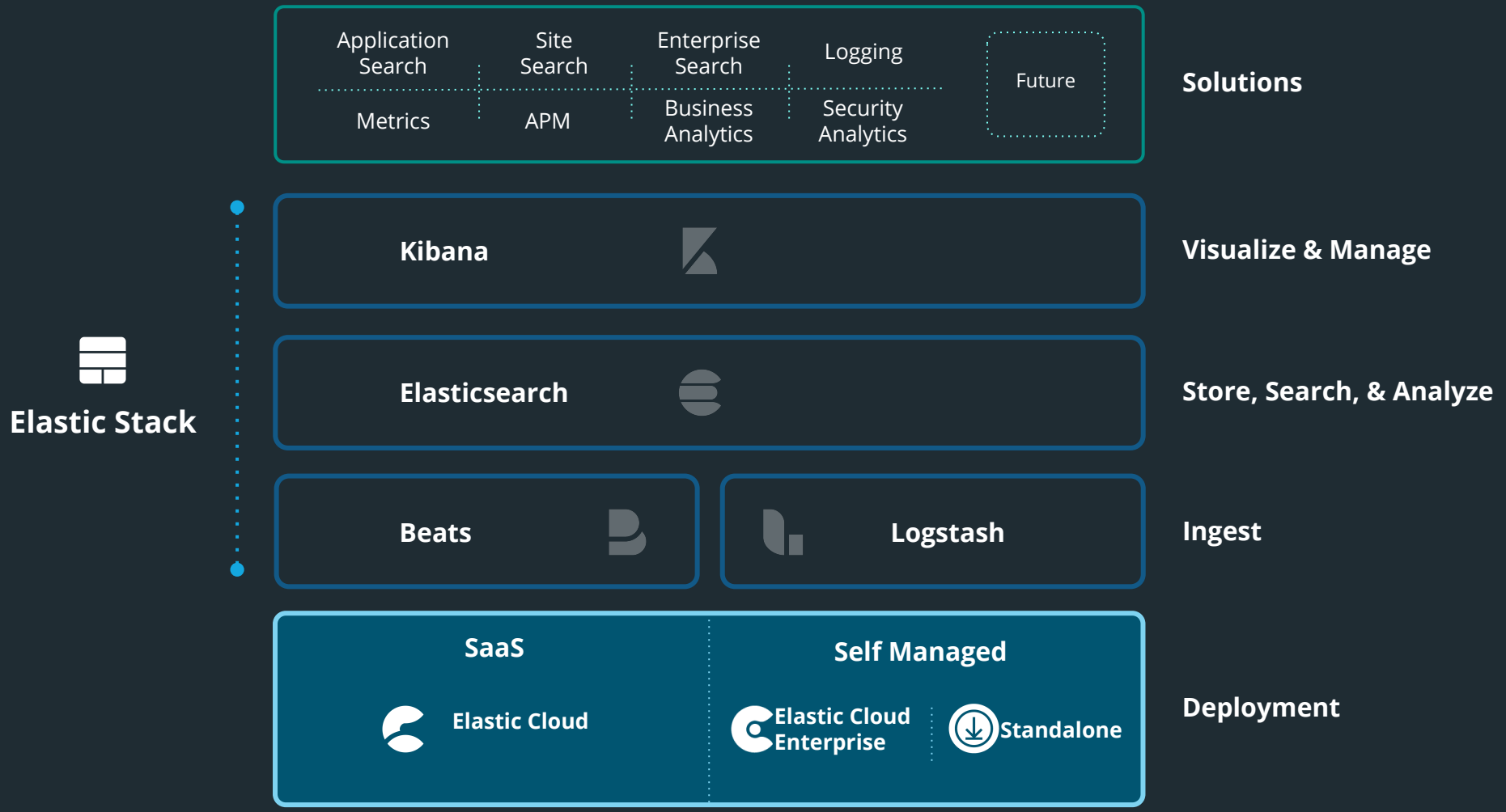
SPEED

Find matches in milliseconds

RELEVANCE

Get highly relevant results

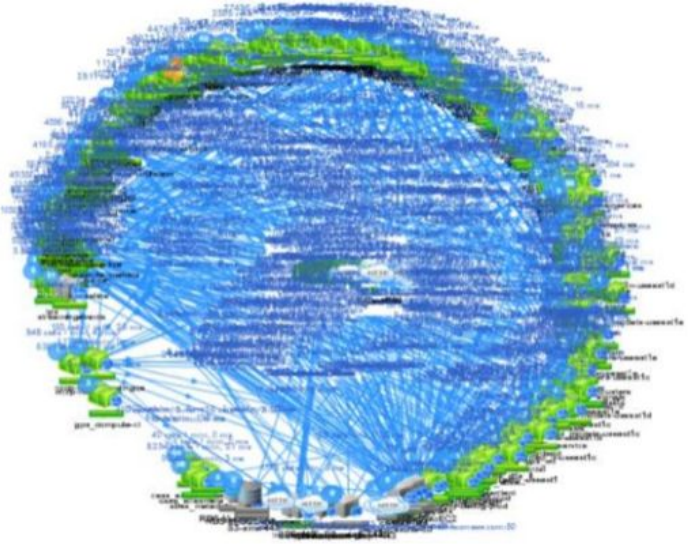

Application Search
Site Search
Enterprise Search
Logging
Metrics
APM
Business Analytics
Security Analytics
Future
Solutions
Kibana
Visualize & Manage
Elasticsearch
Store, Search, & Analyze
Beats
Logstash
Ingest
Elastic Stack


Application Search
Site Search
Enterprise Search
Logging
Metrics
APM
Business Analytics
Security Analytics
Future
Solutions
Elastic Stack
Kibana
Visualize & Manage
Elasticsearch
Store, Search, & Analyze
Beats
Logstash
Ingest
SaaS
Elastic Cloud
Self Managed
Elastic Cloud Enterprise
Standalone
Deployment

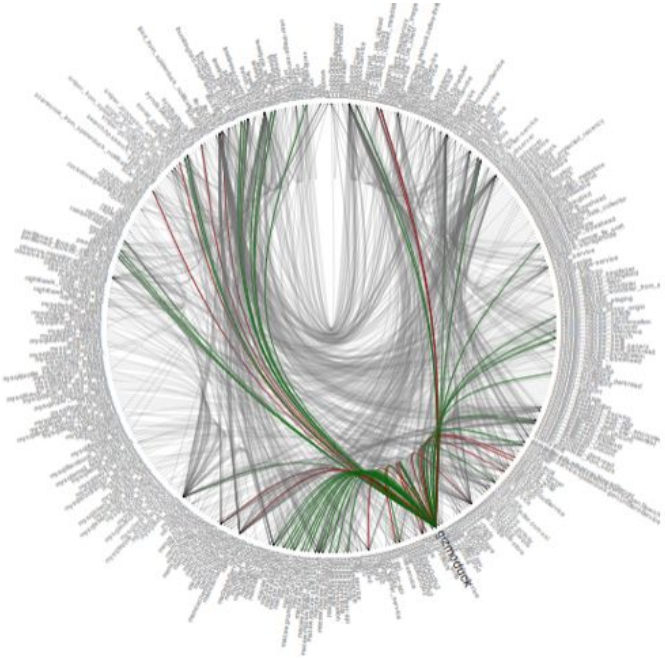
Agenda

- Microservices
- Distributed Tracing
- OpenTracing
- Elastic APM & Distributed Tracing
- Demo
- Q&A

Microservices Can Be Very Complex



Netflix



Twitter

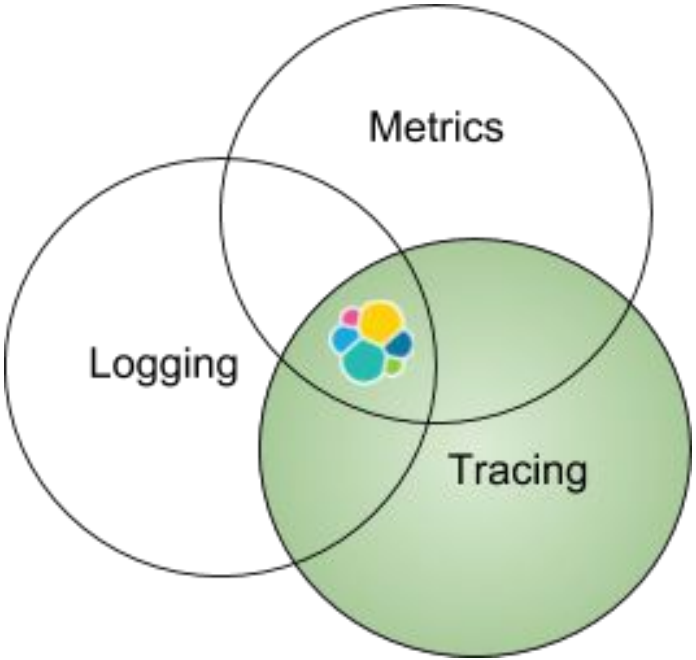
Why Distributed Tracing?

Distributed tracing provides a solution for describing and analyzing the cross-process transactions.

- Latency tracking
  - Where & why things are slow?
  - Slow code, slow network?
- Root cause analysis
  - Where & why did it fail?
  - Bad code?
  - Network failure?

elastic

The 3rd Pillar of Observability

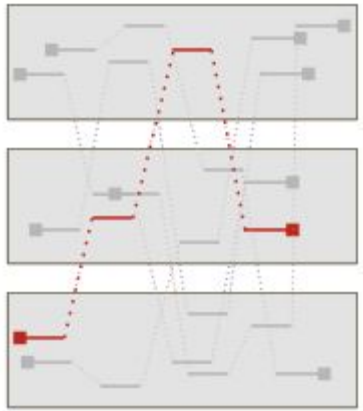


elastic

Distributed Tracers

A fast growing ecosystem

- **Dapper** (Google) - 2010.
  - Google's production distributed systems tracing infrastructure
  - A self-contained tracing tool evolved into a monitoring platform
- **Zipkin** (Twitter)
  - Developed by **Twitter** based on the Google Dapper paper
  - Written in **Java**
  - Uses Cassandra or **Elasticsearch** as a scalable backend, MySQL too
- **Jaeger** (Uber)
  - Newer project from **Uber**, CNCF adopted as an Incubating project
  - Written in **Golang**
  - Supports Cassandra and **Elasticsearch** as scalable storage backends
- **Appdash**
- Others ...



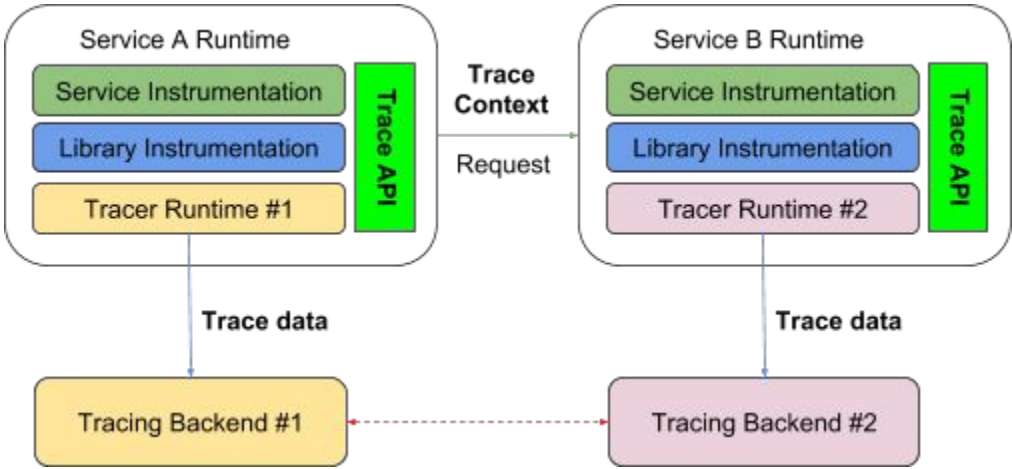
Challenges for Application Developers

- Add the tracer in the application code
- Which one to use?
- What if I need to change my tracer?
- No, I don’t want to change my entire source code
- What do I do with shared libraries?
- What if my third party services use a different tracer?

We Need Distributed Tracing Standardization

Four Components

- **Tracing API**
  - The OpenTracing API
- **Wire Protocol**
  - What gets sent alongside application data in RPC requests
  - Distributed Context Propagation
  - Eg. The trace-context HTTP headers, AMQP message headers
- **Data Definition & Format**
  - What gets sent asynchronously to the analysis system
- **Analysis System**
  - A service for extracting insights from trace data



elastic

Enter OpenTracing

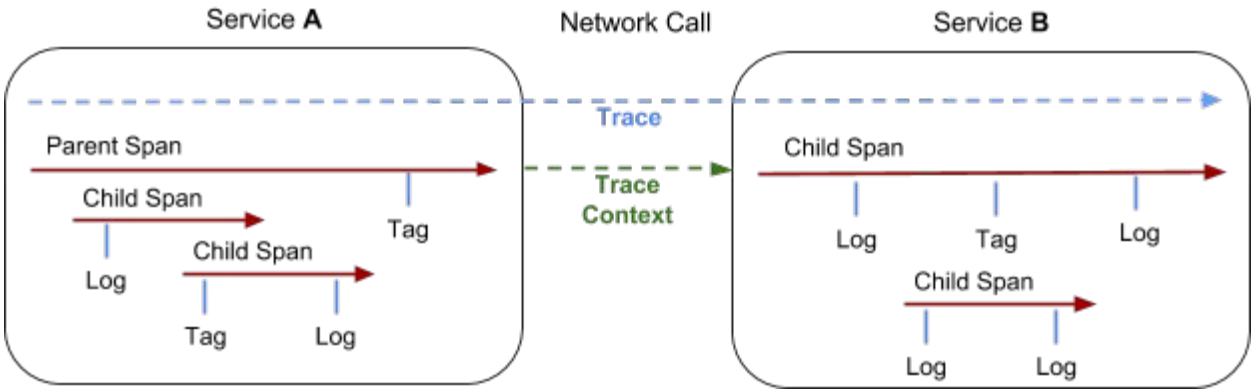
Emerging Industry Standard for Distributed Tracing

- Abstraction from different tracers
- Vendor-neutral APIs and instrumentation
- Switching between tracers
- OpenTracing and OpenCensus are merging
  - Make observability a built-in feature in every modern application
  - Not just “tracing” - merged project includes data sources beyond distributed transaction traces, like metrics, traces and logs

OpenTracing

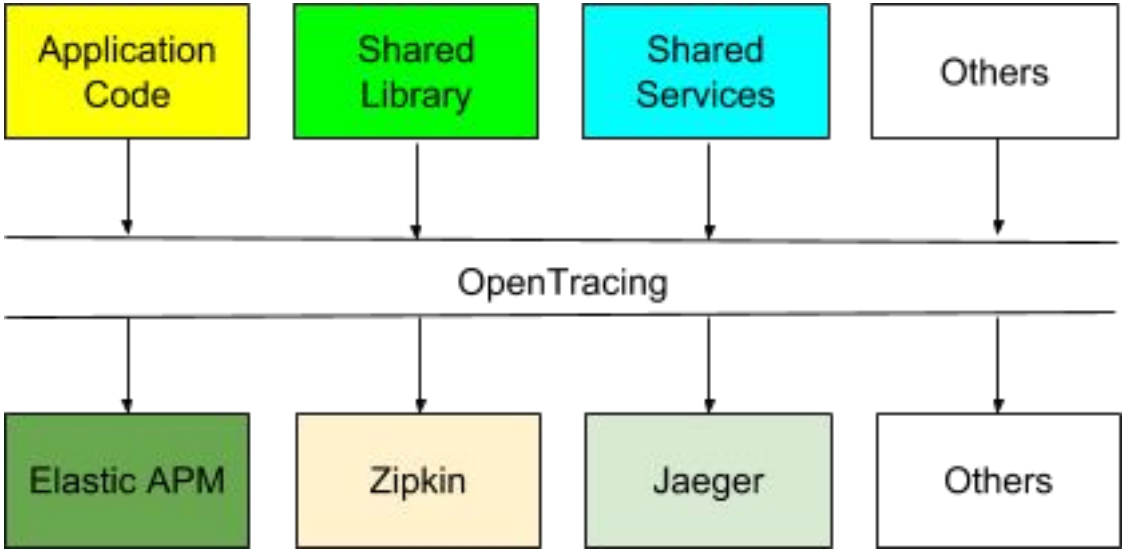
A Mental Model

- Distributed Tracing is a process of collecting end-to-end transaction graphs in near real time
- A **trace** represents the entire journey of a request
- A **span** represents single operation call
- As a request is flowing from one microservice to another, tracers add logic to create unique trace Id, span Id



elastic

How Does It All Fit In?



elastic

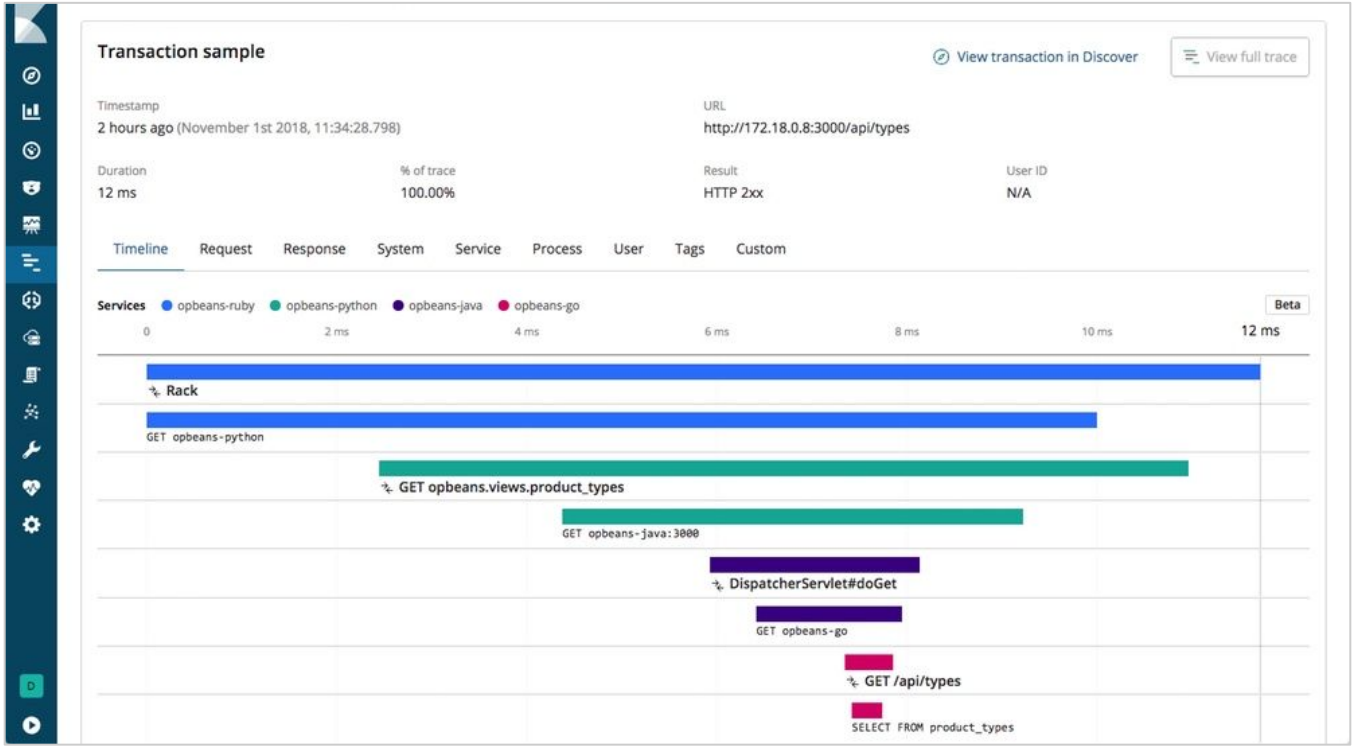
The Unfortunate Facts - Today

No Runtime Compatibility among Tracers, but there is hope

- The only thing standardized by OpenTracing is the programmatic **Tracing API**
  - `io.opentracing.Span, io.opentracing.Tracer`
- **Wire Protocol** is not standardized yet.
  - For example, HTTP headers for context propagation:
    - Jaeger: **uber-trace-id**: 118c6c15301b9b3b3:56e66177e6e55a91:18c6c15301b9b3b3:1
    - Elastic: **elastic-apm-traceparent**: 00-f109f092a7d869fb4615784bacefcfd7-5bf936f4fcde3af0-01
  - HTTP header name is different & ID format is different
  - Trace-Context is a W3C working group for standardizing the HTTP headers for distributed tracing https://github.com/w3c/distributed-tracing
  - OpenTracing will most likely adopt and require Trace-Context for distributed tracing HTTP headers as part of its specification
- **Data Protocol** is not standardized
  - Zipkin, Jaeger and Elastic APM use different data format for Span data and others

Elastic & Distributed Tracing

Elastic APM is OpenTracing Compliant and Supports Distributed Tracing



elastic

Elastic OpenTracing Bridge

Reuse Existing Instrumentations

```
<dependency>
    <groupId>co.elastic.apm</groupId>
    <artifactId>apm-opentracing</artifactId>
    <version>${elastic-apm.version}</version>
</dependency>
```

```
import co.elastic.apm.opentracing.ElasticApmTracer;
import io.opentracing.Tracer

Tracer tracer = new ElasticApmTracer();
```

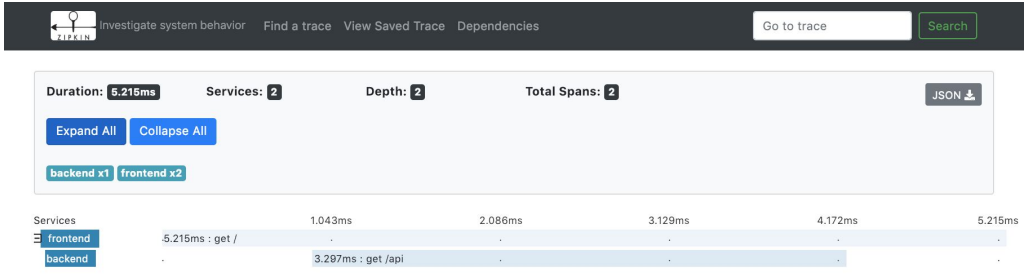
Allows you to replace other OpenTracing-compliant tracers like Jaeger without having to change the rest of the tracing code.

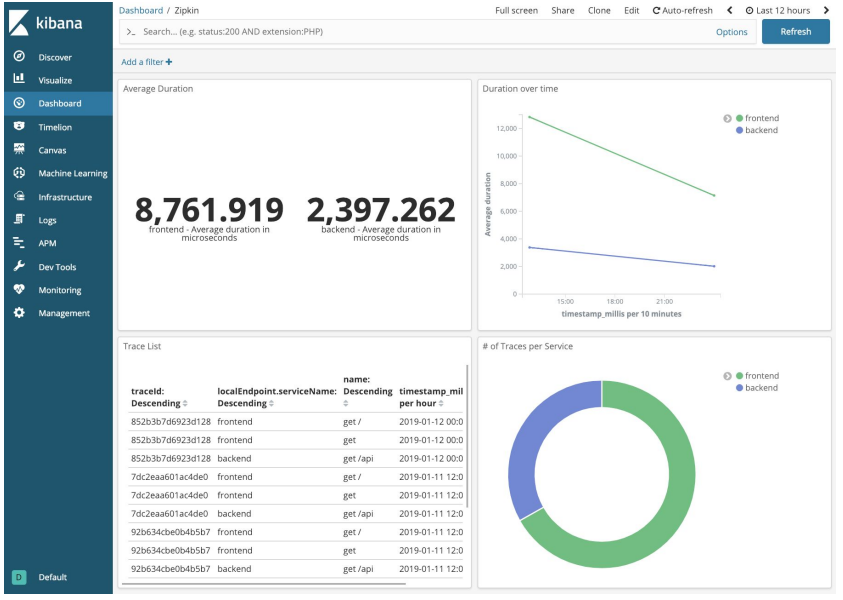
See: https://www.elastic.co/guide/en/apm/agent/java/current/opentracing-bridge.html

Elastic & Distributed Tracing

Integrating Zipkin, Jaeger with Elastic

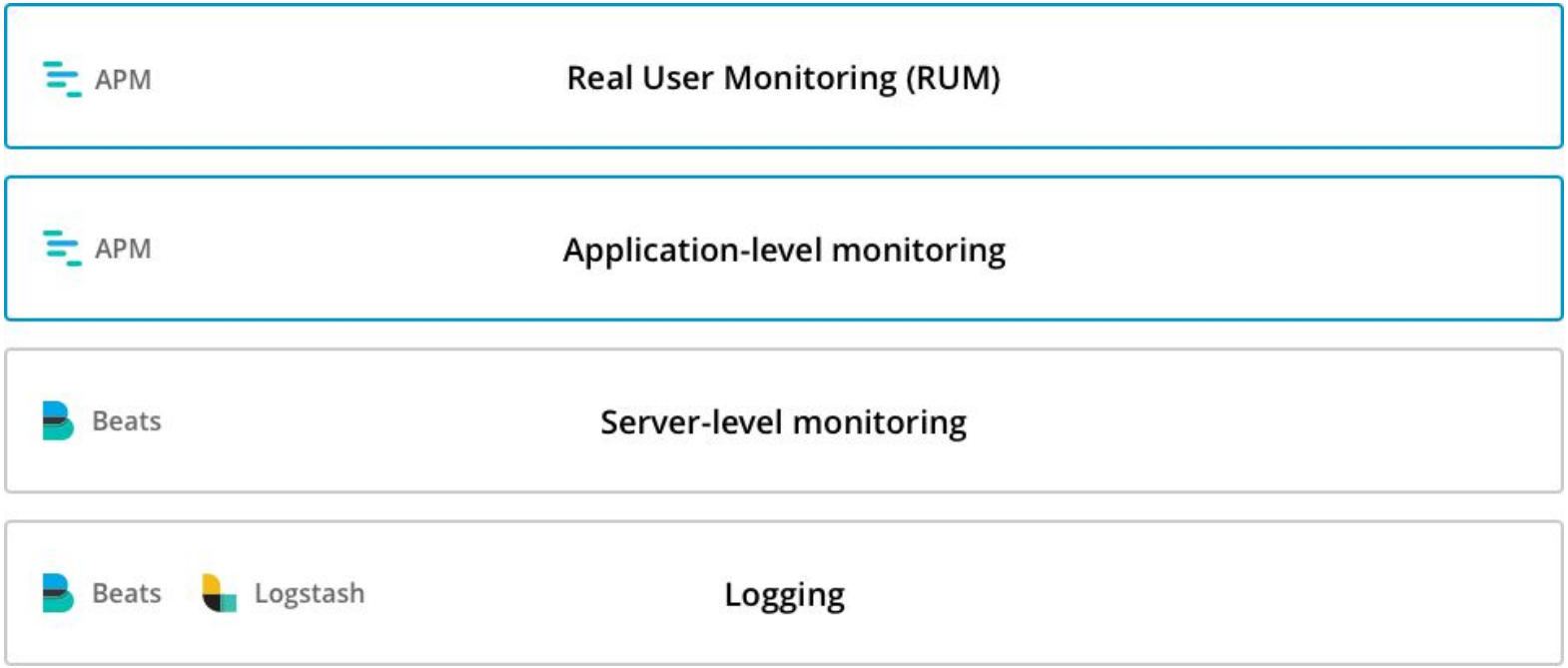
- Zipkin & Jaeger
  - Basic tracing information
  - Dependency diagram
  - Out-of-the-box integration with Elasticsearch
- Elasticsearch
  - Rich trace analysis
  - Long-term trace data storage
  - Correlation with logs & metrics





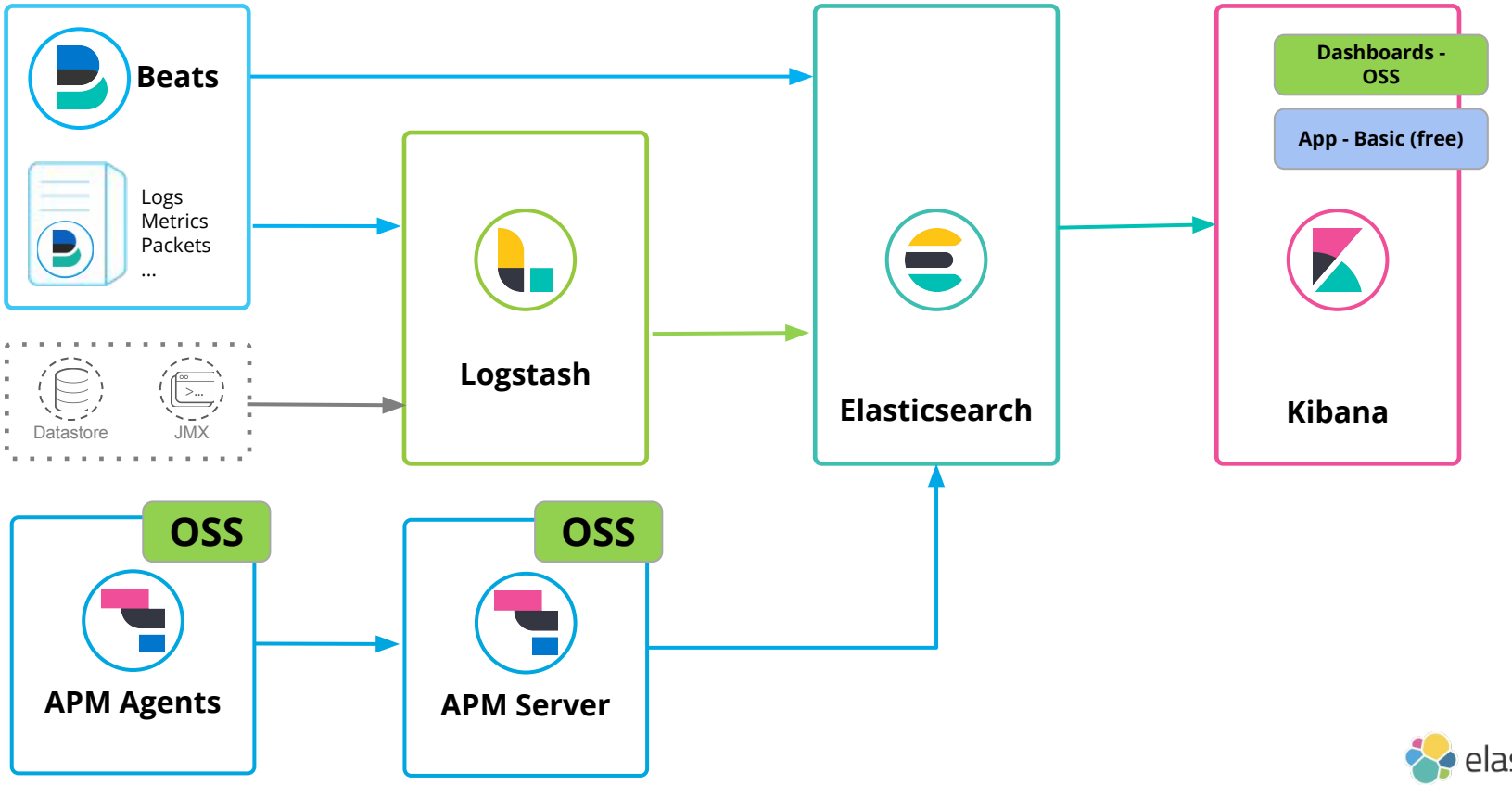
Full stack monitoring in a single place

Adding end-user experience and application-level monitoring to the Elastic Stack



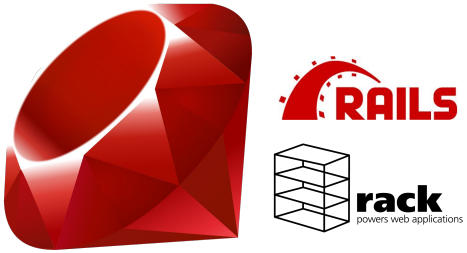
elastic

Where APM fits in the Elastic Stack



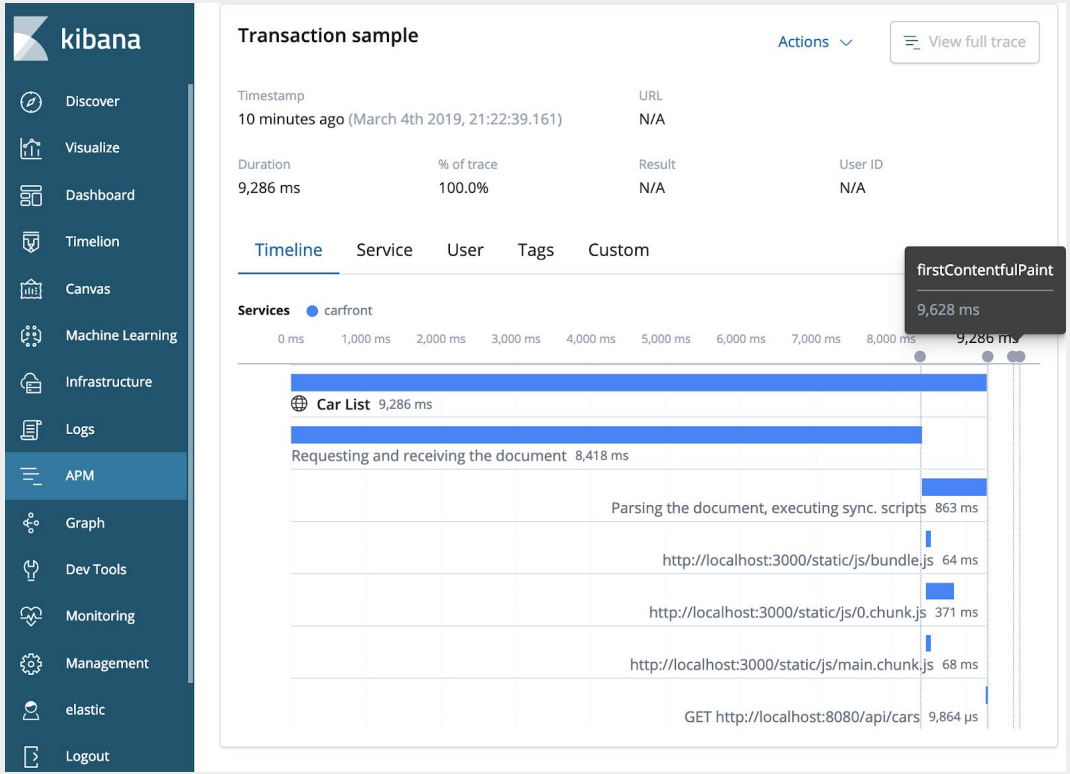
elastic

Supported Languages & Frameworks

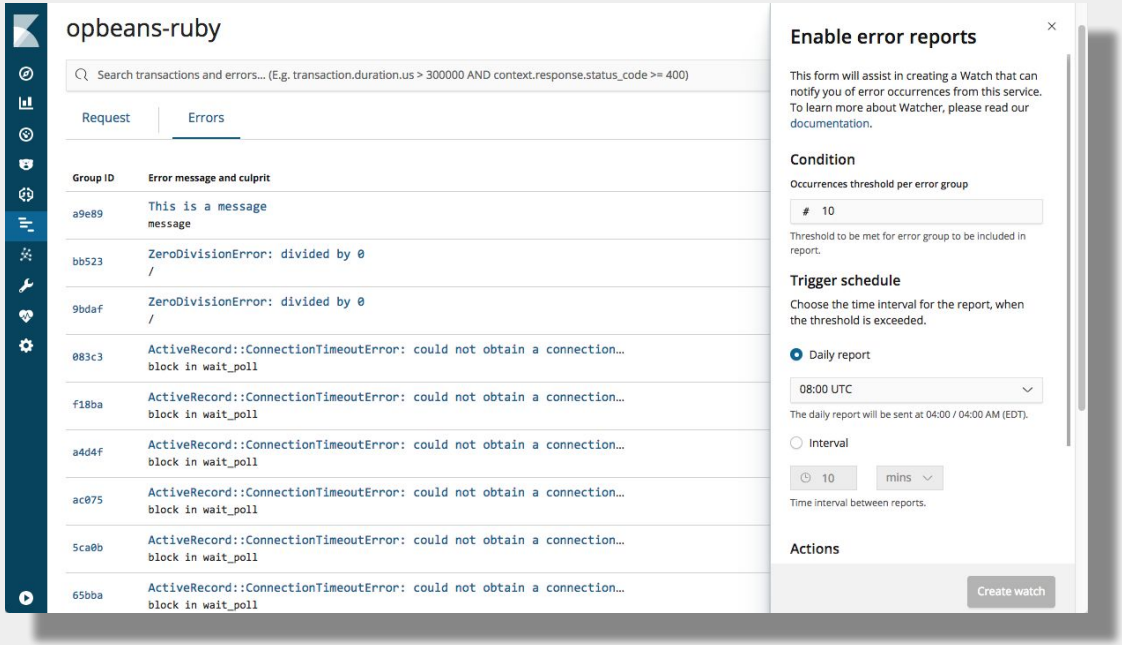


RUM

- Measures actual end-user experience
- See where the browser spends its time
- Similar waterfall view
- Annotations at key DOM events



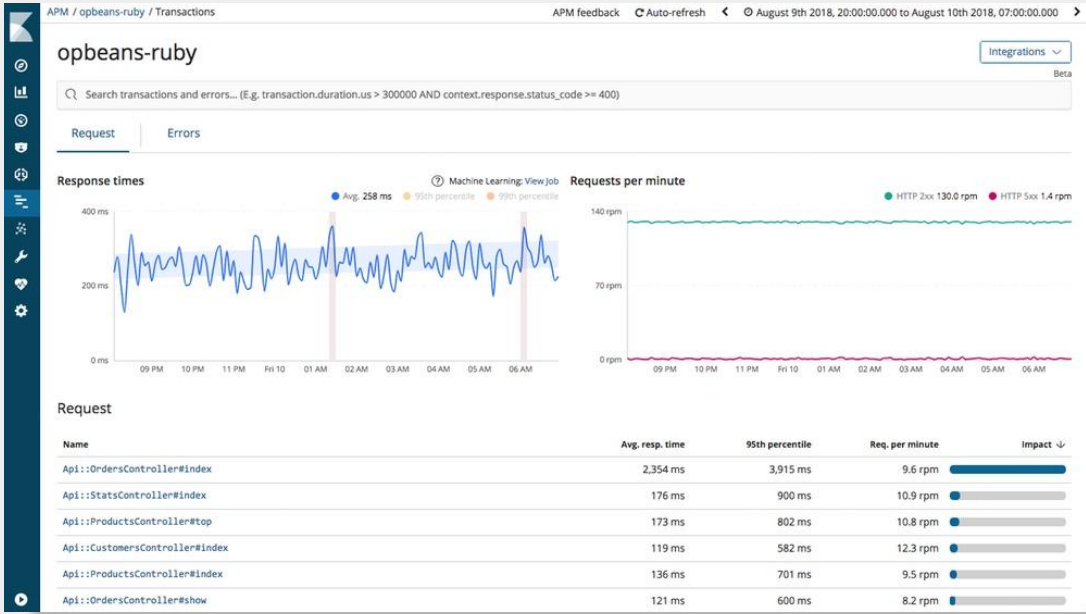
Alerting Integration



- Daily or Interval based
- Configurable Threshold
- Slack or email notification

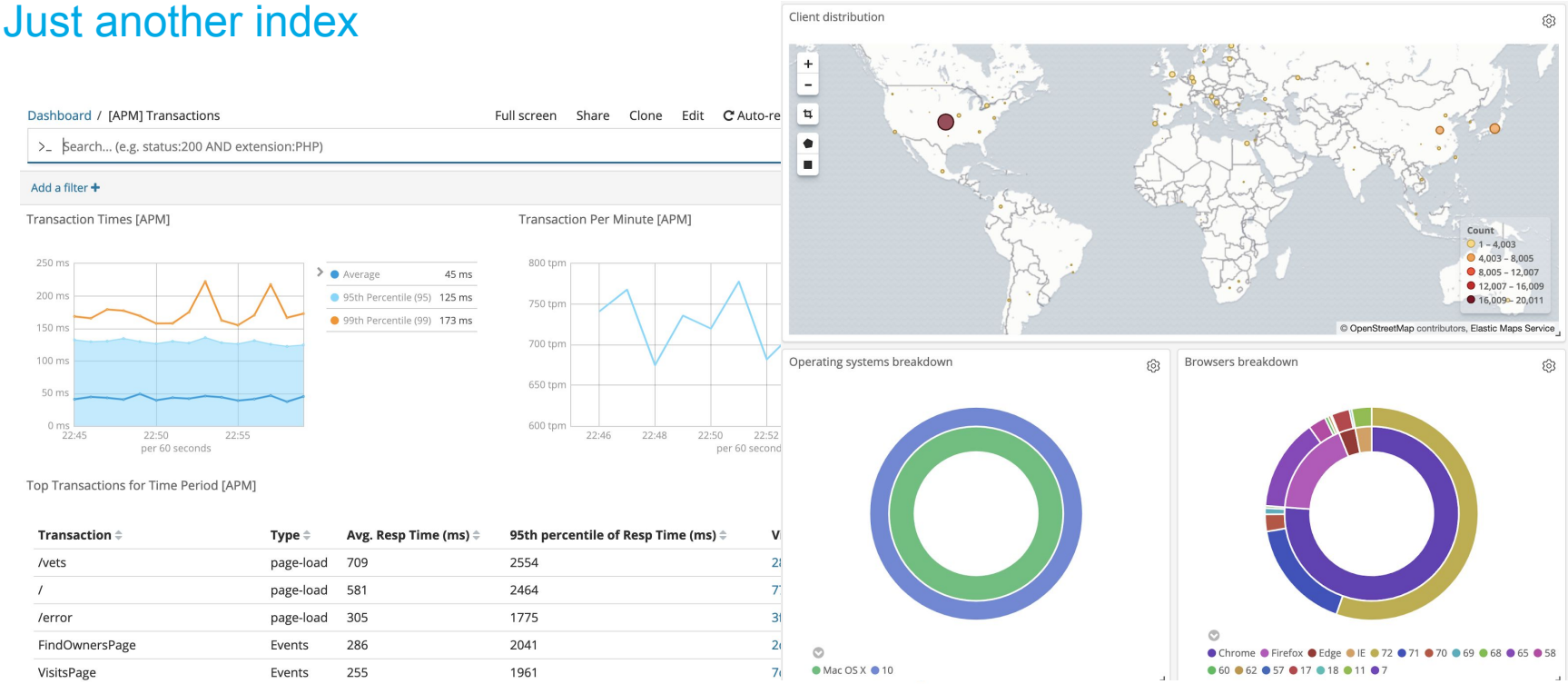
ML Integration

- Calculate anomaly scores on high response times
- Graph annotations when score is over 75



Import Dashboards or Make Custom Visualizations

Just another index

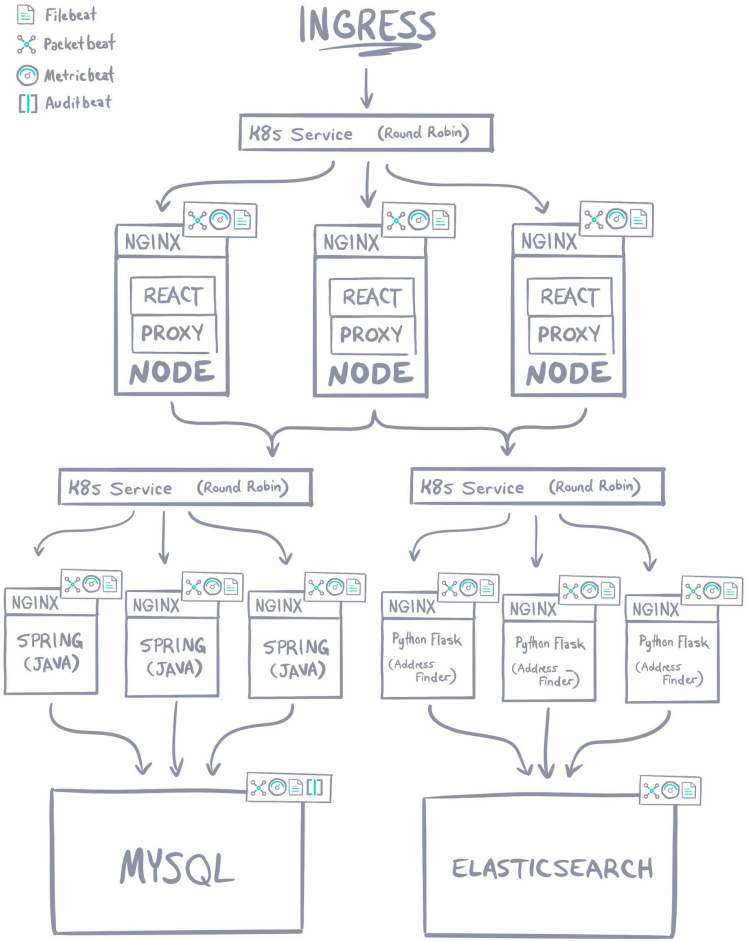


elastic

Demo Time

Application Architecture

- React frontend
- Node.js Express proxy
- Java backend
- MySQL data store
- Python for address lookup
- Elasticsearch backend
- Elastic Cloud:
  - ES
  - Kibana
  - APM



elastic

Key Takeaways

Why Elastic APM?

- Elastic APM is **OpenTracing compliant**
- Elastic OpenTracing bridge allows **instrumentation reuse**
- Elastic Stack is a great **scalable long-term storage** for other tracers
- Elastic provides **rich analytics** for tracing data Elastic or not
- Elastic is a great platform for all **three pillars of observability** - logging, metrics & tracing

Resources

Elastic APM: https://www.elastic.co/solutions/apm

Blog: Distributed Tracing, OpenTracing & Elastic APM

Blog: A Sip of Elastic RUM (Real User Monitoring)

APM Forum: https://discuss.elastic.co/c/apm

Thank You

- Web : www.elastic.co
- Demos: demo.elastic.co
- Products : https://www.elastic.co/products
- Forums : https://discuss.elastic.co/
- Community : https://www.elastic.co/community/meetups
- Twitter : @elastic

Questions?