Everything You Wanted to Know About Distributed Tracing.

Strange Loop Conference

September 2019

AJUA

Who Am I

- Hungai Amuhinda
- Twitter: @Hungai
- Nairobi, Kenya
- Work: Ajua, Infra Team
- Website: hungaikev.in

AJUA

How to use this talk

- **Please don’t just follow it blindly:** There are often times when you will need to do things differently.
- **Software is all about trade-offs**: Very few decisions are about right and wrong.
- **Try things for yourself**: Why not make Friday afternoons a time to play and experiment?

AJUA

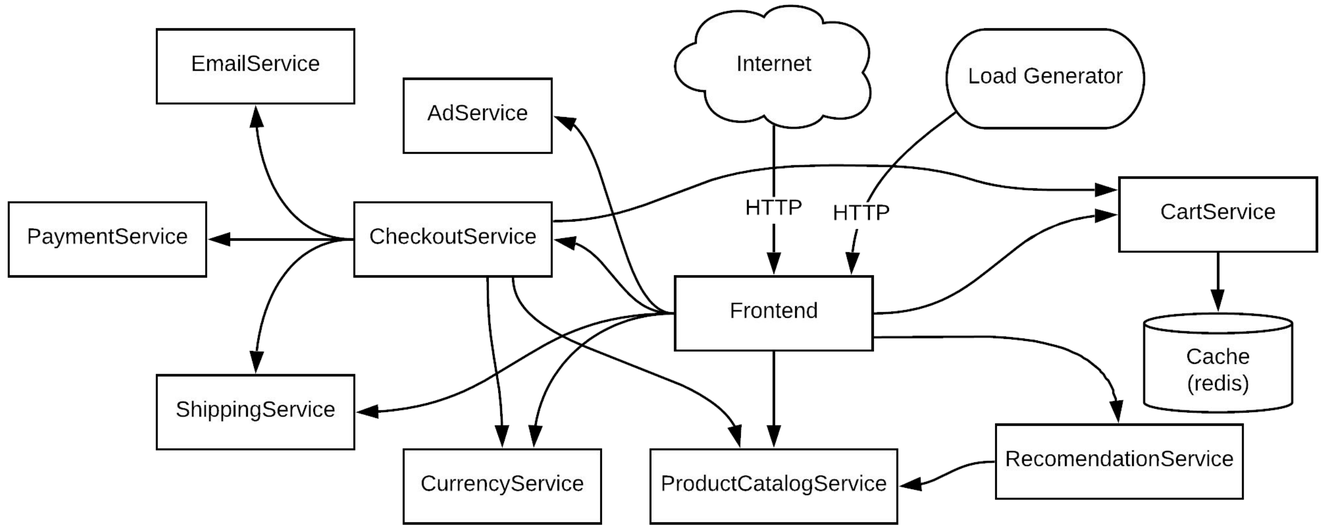
Software Evolution.

- Monolith
- On Prem
- Single Language
- Single Stack
- Virtual Machines

AJUA

Software Evolution.

- Microservices
- Containers
- Multi Cloud/ Hybrid
- Polyglot
- Containers
- Serverless/ Cloud Functions



AJUA

New Architectures/ New Challenges.

- **Observability**
- Deployment / Packaging
- Configuration Management
- Debugging
- Secrets Management

AJUA

Meet: Distributed Tracing

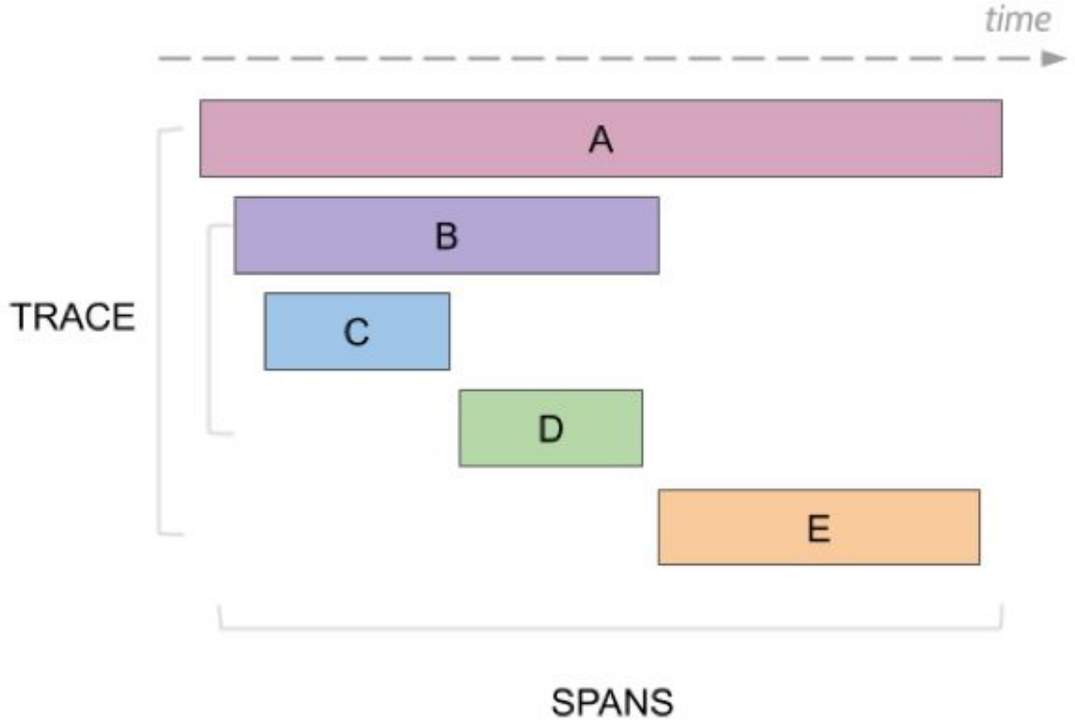
“**Distributed Tracing**, also called distributed request tracing, is a method used to profile and monitor applications, especially those built using a microservices architecture. Distributed tracing helps pinpoint where failures occur and what causes poor performance.”

AJUA

Distributed Tracing - Terminology

Trace - a trace is a tree of spans that follows the course of a request or system from its source to its ultimate destination.

Each **trace** is a narrative that tells the requests story as it travels through the system.

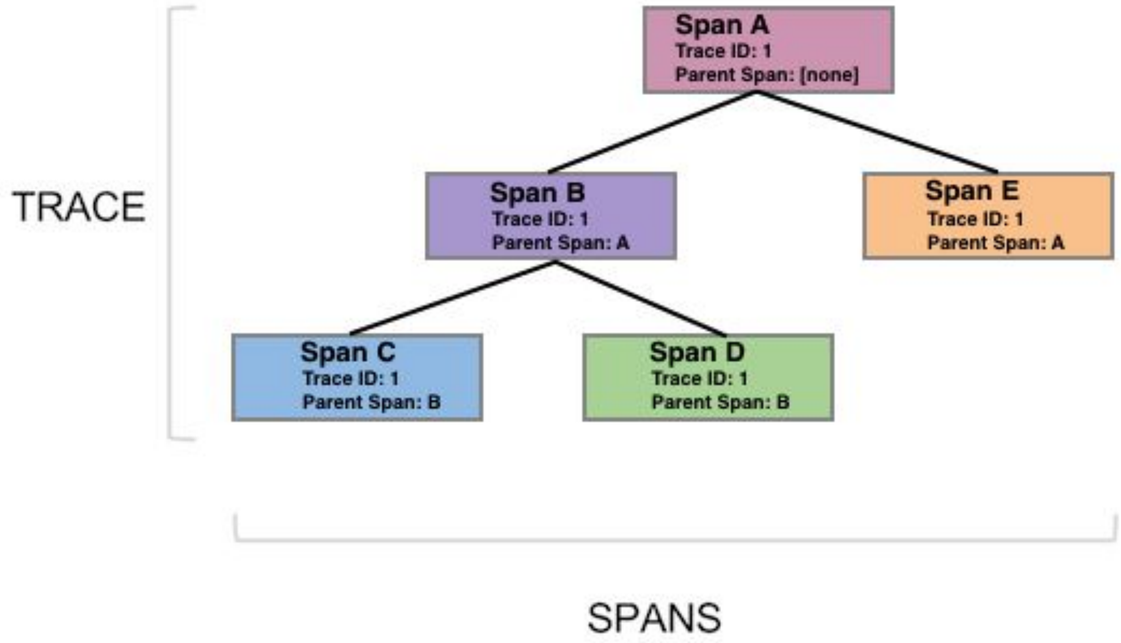


AJUA

Distributed Tracing - Terminology

Span - are logical units of work in a distributed system. They all have a name, a start time, and a duration.

Each **Span** captures important data points specific to the current process handling the request.

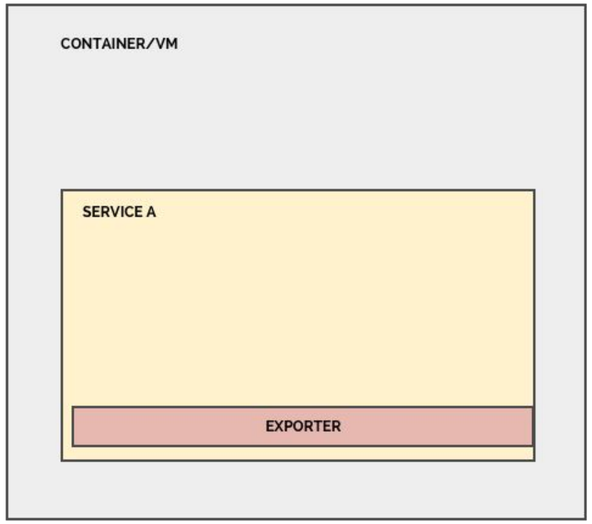


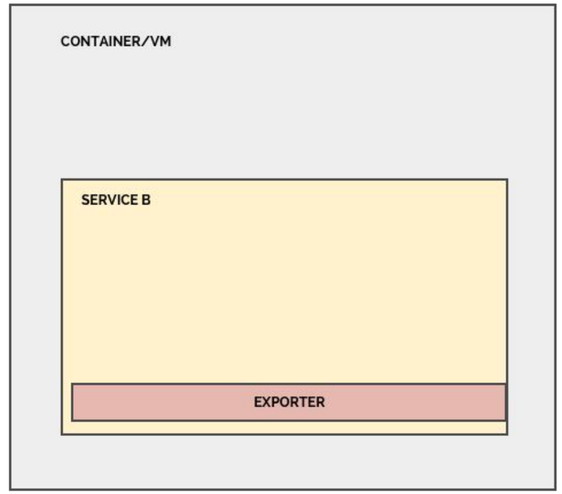
AJUA

Distributed Tracing - Terminology

Context Propagation:

Incoming Request





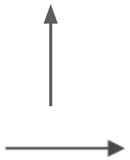
AJUA

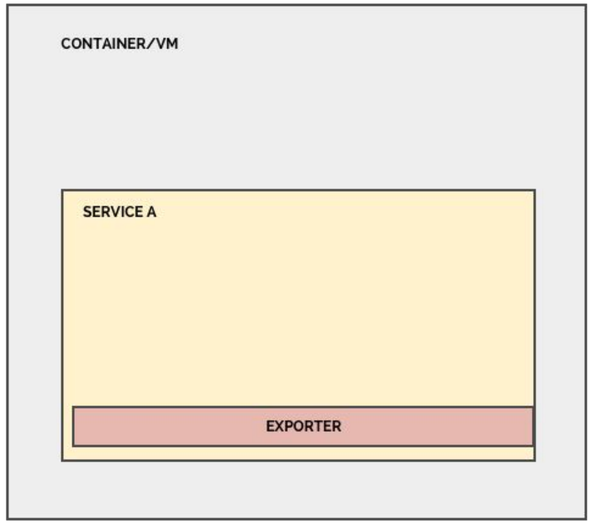
Distributed Tracing - Terminology

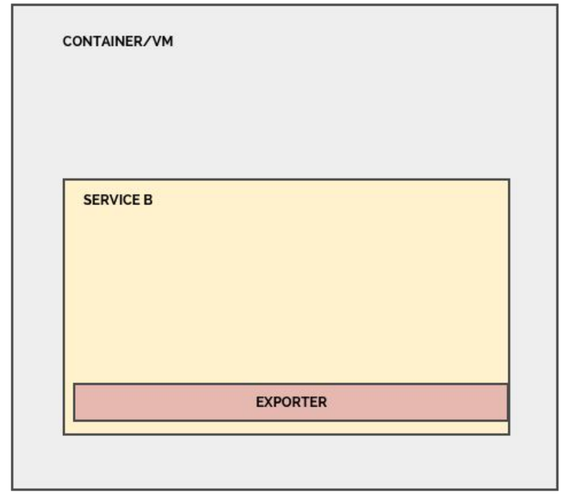
Context Propagation:

trace-id = 123
parent-d = nil
span-id = 1

Incoming Request



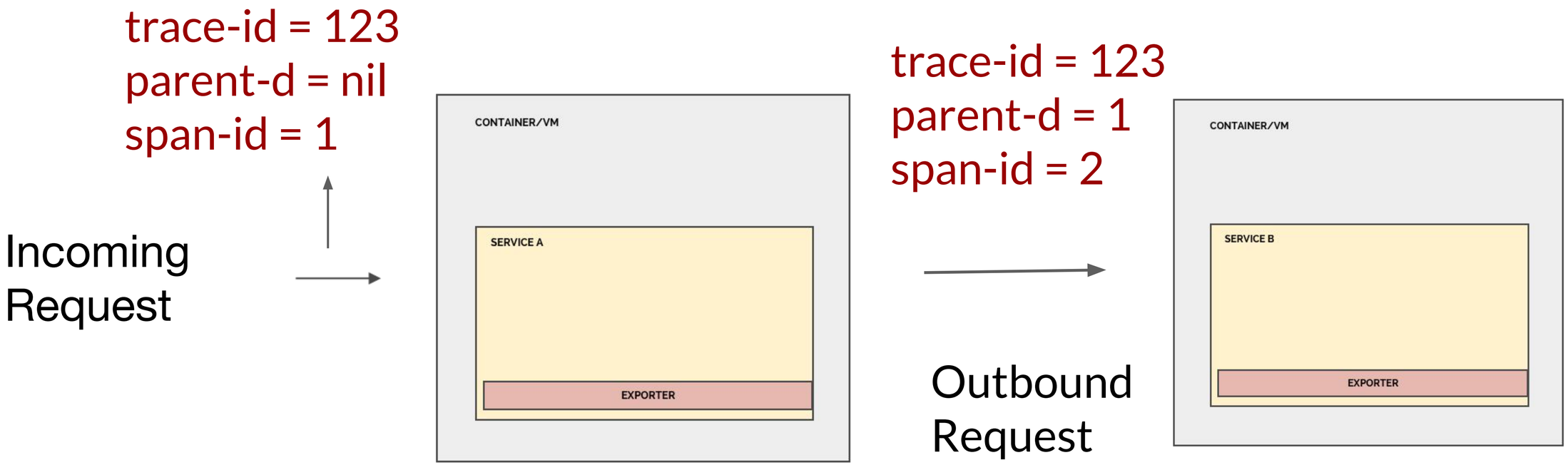




AJUA

Distributed Tracing - Terminology

Context Propagation:



Distributed Tracing - Terminology

Tags & Logs: both annotate the span with some contextual information.

- Tags typically apply to the whole span, while logs represent some events that happened during the span execution.
- A log always has a timestamp that falls within the span's start-end time interval.
- The tracing system does not explicitly track causality between logged events the way it keeps track of causality relationships between spans, because it can be inferred from the timestamps.

AJUA

What questions can tracing help us answer?

AJUA

Distributed Tracing:

- What services did a **request pass through**?

AJUA

Distributed Tracing:

- What services did a **request pass through**?
- What occured in **each service** for a given request?

AJUA

Distributed Tracing:

- What services did a r**equest pass through**?
- What occured in **each service** for a given request?
- Where did the **error** happen?

AJUA

Distributed Tracing:

- What services did a r**equest pass through**?
- What occured in **each service** for a given request?
- Where did the **error** happen?
- Where are the **bottlenecks**?

AJUA

Distributed Tracing:

- What services did a r**equest pass through**?
- What occured in **each service** for a given request?
- Where did the **error** happen?
- Where are the **bottlenecks**?
- What is the **critical path** for a request?

AJUA

Distributed Tracing:

- What services did a r**equest pass through**?
- What occured in **each service** for a given request?
- Where did the **error** happen?
- Where are the **bottlenecks**?
- What is the **critical path** for a request?
- Who should I **page**?

AJUA

If tracing is so good why isn’t everyone using it?

AJUA

If tracing is so good why isn’t everyone using it?

- Not much education or not many publicized case studies on the benefits.

AJUA

If tracing is so good why isn’t everyone using it?

- Not much education or not many publicized case studies on the benefits.
- Vendor Lock in is unacceptable: Instrumentation must be decoupled from vendors

AJUA

If tracing is so good why isn’t everyone using it?

- Not much education or not many publicized case studies on the benefits.
- Vendor Lock in is unacceptable: Instrumentation must be decoupled from vendors .
- Inconsistent APIs: Tracing semantics must not be language dependent.

AJUA

If tracing is so good why isn’t everyone using it?

- Not much education or not many publicized case studies on the benefits.
- Vendor Lock in is unacceptable: Instrumentation must be decoupled from vendors .
- Inconsistent APIs: Tracing semantics must not be language dependent.
- Handoff woes: Tracing libs in Project X do not handoff to tracing libs in Project Y.

AJUA

Meet OpenTelemetry

OpenTelemetry

Open Telemetry is made up of an integrated set of APIs and libraries as well as a collection mechanism via a agent and collector. These components are used to generate, collect, and describe telemetry about distributed systems.

Problems OpenTelemetry solves:

- Vendor neutrality for tracing, monitoring and logging
- Context Propagation.

AJUA

OpenTelemetry (opentelemetry.io) Is:

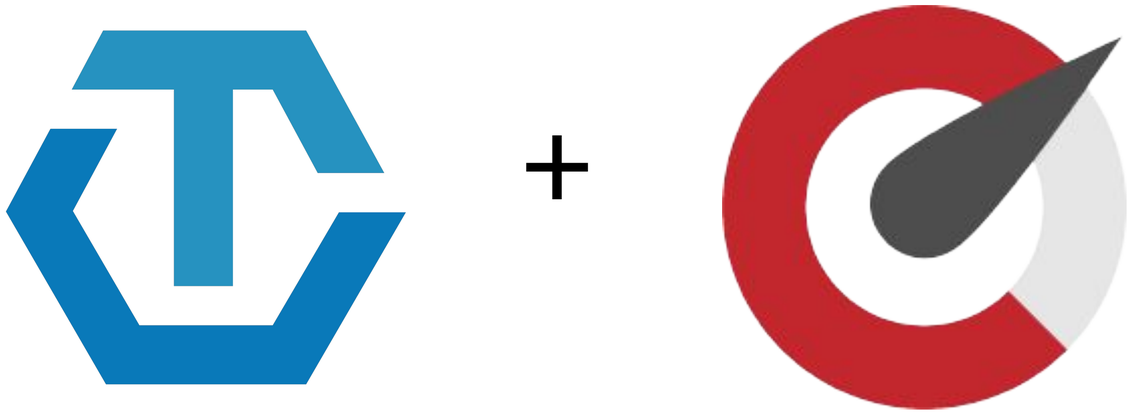
- **Single set of APIs** for tracing and metrics collection.
- **Standardized Context Propagation**.
- **Exporters** for sending data to backend of choice.
- **Collector** for smart traces & metrics aggregation.
- **Integrations** with popular web, RPC and storage frameworks.

AJUA

OpenTelemetry (opentelemetry.io) Is:

Next major version of the OpenTracing and OpenCensus projects.

OpenTelemetry Roadmap:



Merging OpenTracing and OpenCensus: Goals and Non-Goals

Ben Sigelman Follow

Mar 29 · 3 min read

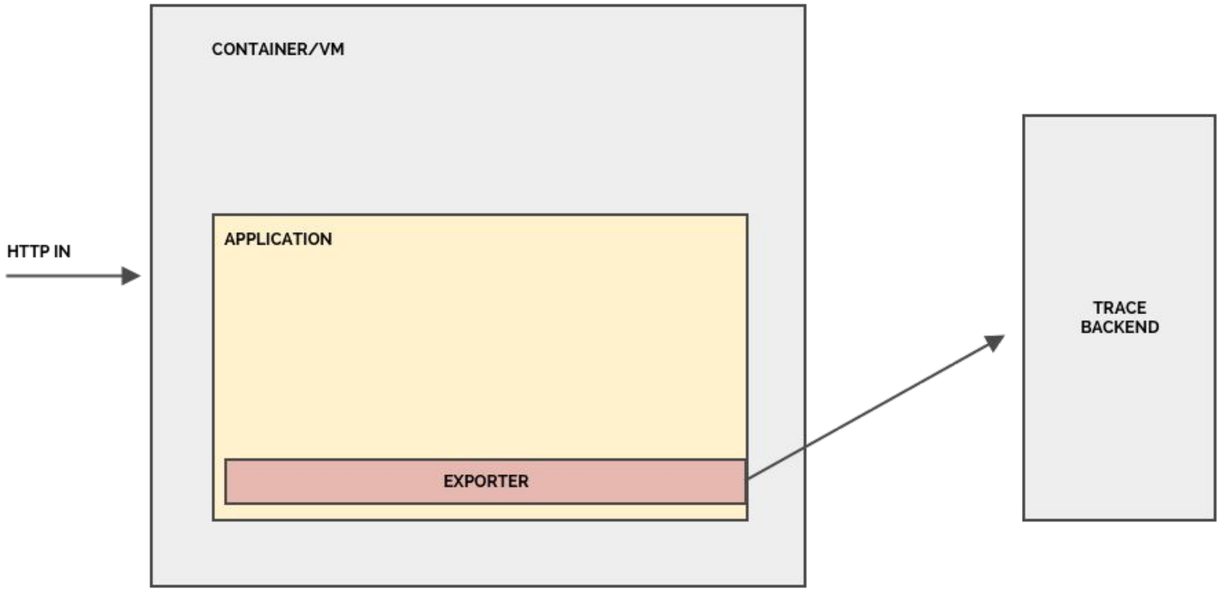
Announcement: https://medium.com/opentracing/merging-opentracing-and-opencensus-f0fe9c7ca6f0

Roadmap: https://medium.com/opentracing/a-roadmap-to-convergence-b074e5815289

AJUA

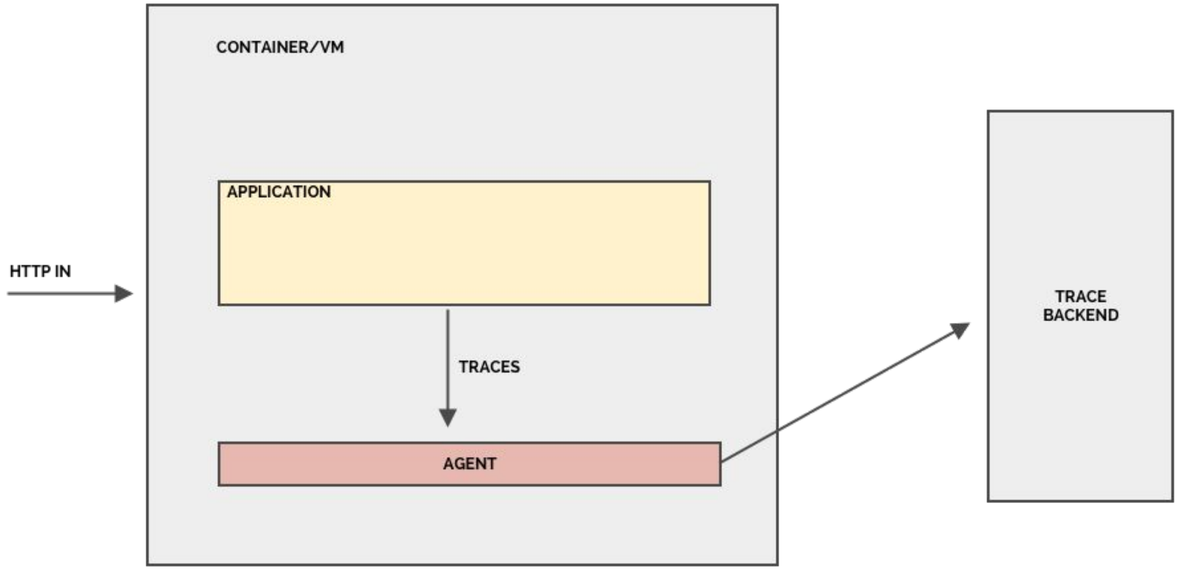
Tracing with OpenTelemetry - The Options

Agentless



Initialize exporter in app code

Using an Agent



Install the agent alongside the app

AJUA

OpenTelemetry: How to get Involved

Github: https://github.com/open-telemetry

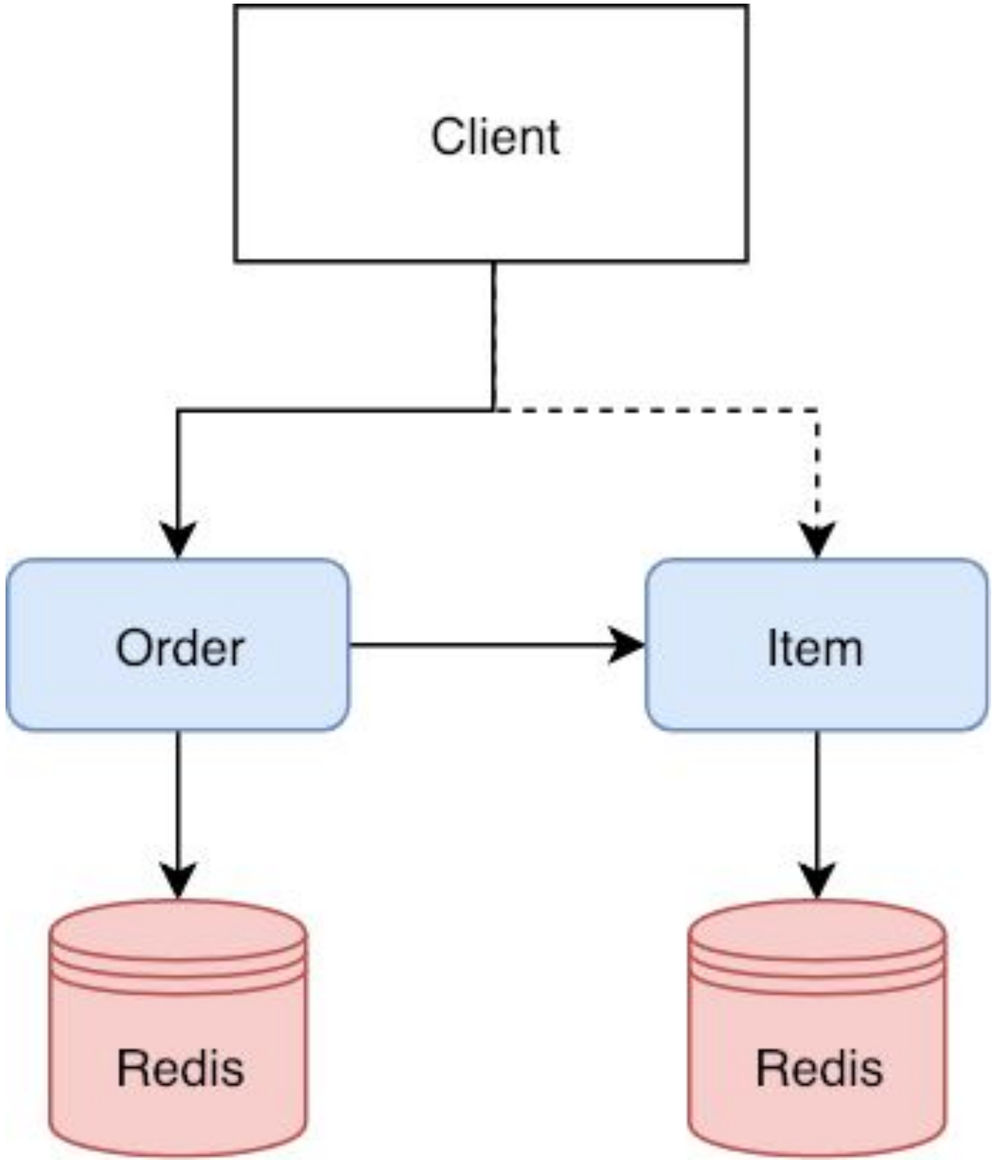
Gitter: https://gitter.im/open-telemetry

Languages:

- .NET SDK
- GoLang SDK
- Java SDK
- JavaScript SDK
- Python SDK
- Ruby SIG
- Erlang/Elixir SDK

AJUA

EXAMPLE



AJUA

Create a Tracer

```
125     // Creating tracer
126     var tracer ot.Tracer
127     var closer io.Closer
128     tracer, closer, err = util.InitTracer("item", s.logger)
129     if err != nil {
130         s.logger.Warnw("unable to initialize tracer",
131             "error", err,
132         )
133     } else {
134         defer closer.Close()
135         ot.SetGlobalTracer(tracer)
136     }
137
```

AJUA

Create a Tracer

```
14
15  // InitTracer returns an instance of Jaeger Tracer that samples 100% of traces and logs all spar
16  func InitTracer(serviceName string, logger *Logger) (ot.Tracer, io.Closer, error) {
17      cfg, err := config.FromEnv()
18      if err != nil {
19          return nil, nil, err
20      }
21
22      cfg.Sampler.Type = "const"
23      cfg.Sampler.Param = 1
24      cfg.Reporter.LogSpans = false
25
26      tracer, closer, err := cfg.New(
27          serviceName,
28          config.Logger(logger),
29          config.Metrics(prometheus.New()),
30      )
31      if err != nil {
32          return nil, nil, err
33      }
34      return tracer, closer, nil
35  }
```

AJUA

Tracers

JAEGER

DATADOG

honeycomb.io

ZIPKIN

THE APACHE SOFTWARE FOUNDATION

Skywalking

LIGHTSTEP

AJUA

Instrument

```
37  // TracerMiddleware adds a Span to the request Context ready for other handlers to use it.
38  func TracerMiddleware(inner http.Handler, route Route) http.HandlerFunc {
39      return http.HandlerFunc(func(w http.ResponseWriter, r *http.Request) {
40          var ctx context.Context
41          var span ot.Span
42          tracer := ot.GlobalTracer()
43
44          // If possible, extract span context from headers
45          spanCtx, _ := tracer.Extract(ot.HTTPHeaders, ot.HTTPHeadersCarrier(r.Header))
46          if spanCtx == nil {
47              span = tracer.StartSpan("request")
48              defer span.Finish()
49              ctx = ot.ContextWithSpan(r.Context(), span)
50          } else {
51              span = tracer.StartSpan("request", ext.RPCServerOption(spanCtx))
52              defer span.Finish()
53              ctx = ot.ContextWithSpan(r.Context(), span)
54          }
55          for k, v := range r.Header {
56              span.SetTag(fmt.Sprintf("header.%s", k), v)
57          }
58
59          // TODO: capture return code as tag in root trace
60          span.SetTag("method", r.Method)
61          span.SetTag("url", r.URL.Path)
62          span.SetTag("handler", route.Name)
63
64          r = r.WithContext(ctx)
65          inner.ServeHTTP(w, r)
66      })
67  }
```

AJUA

Instrument

```
204  // getItem retrieves a single Item by ID from Redis.
205  func (s *Server) getItem() http.HandlerFunc {
206      return func(w http.ResponseWriter, r *http.Request) {
207          span, ctx := ot.StartSpanFromContext(r.Context(), "getItem")
208          defer span.Finish()
209
210          pr := mux.Vars(r)
211          key := pr["id"]
212
213          item, err := s.RedisGetItem(ctx, key)
214          if err != nil {
215              s.logger.Errorw("unable to get key from redis",
216                  "key", key,
217                  "error", err,
218              )
219              s.Respond(ctx, http.StatusInternalServerError, "unable to retreive item", 0, nil, w)
220              return
221          }
222          if item == nil {
223              s.Respond(ctx, http.StatusNotFound, fmt.Sprintf("item with ID %s doesn't exist", key
224              return
225          }
226          s.Respond(ctx, http.StatusOK, "item retrieved", 1, []*Item{item}, w)
227      }
228  }
```

AJUA

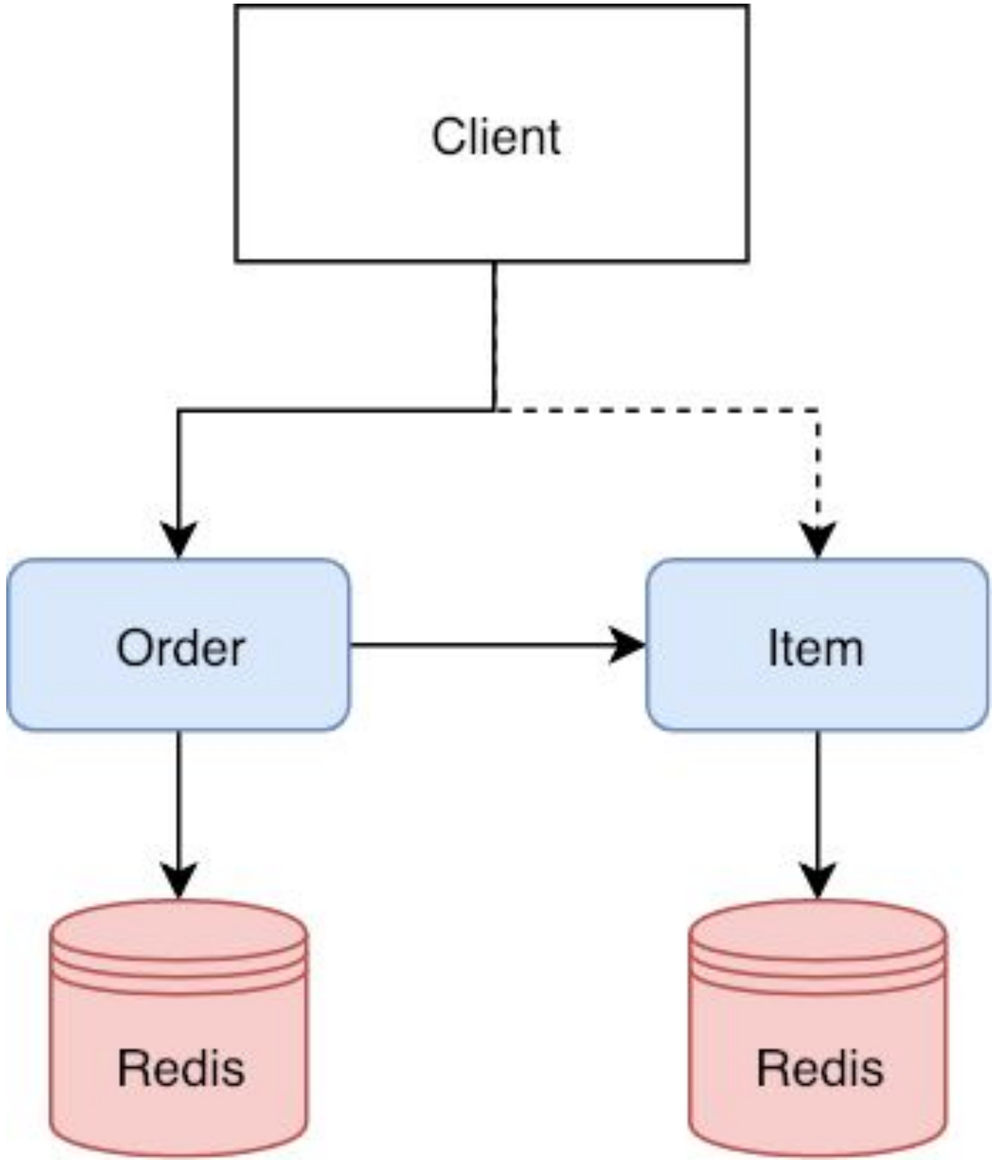
Introducing Jaeger:

- Open source distributed tracing platform.
- Inspired by Google Dapper and OpenZipkin
- Created by Uber in 2015 and donated to CNCF in 2017.
- Compliant with both OpenTracing and OpenCensus.
- Supports multiple storage options (Cassandra, ElasticSearch, In-Memory)
- Compatible with Apache Kafka for backpressure management.

AJUA

DEMO

obitech/micro-obs



AJUA

Conclusion

- Tracing is crucial for understanding complex, microservices applications.
- Distributed tracing provides a base view of the system that can drastically shorten feedback loops and the number of people involved incidents.
- Tracing provides much more context, allowing an on call responder to better understand the system and get further on their own before involving more people.

AJUA

Thank You

Twitter: @Hungai

Email: hungaikevin@gmail.com

AJUA

AJUA