OBJECT COMPUTING

WEBINAR

Distributed Tracing with Micronaut

Zachary Klein, Senior Software Engineer

objectcomputing.com

Speaker

Zachary Klein is a Senior Software Engineer at OCI. He has been practicing web development since 2010 and frontend development since 2015. He's a contributor to both the Grails and Micronaut frameworks, a conference speaker and an instructor in OCI's training practice. Zachary's home base is in St Louis, MO, along with his wife, Beth, and their three children.

What is Micronaut?

"A modern, JVM-based, full-stack framework for building modular, easily testable microservice and serverless applications."

- A **full-featured & lightweight** JVM application framework
- **Polyglot** - support for Java, Groovy, and Kotlin languages
- Based on **Ahead Of Time compilation** vs runtime reflection
- **Reactive** HTTP layer based on Netty
- "Natively" **Cloud Native**

Micronaut: Controllers & Clients

```
@Controller("/")
class HelloController {

  @Get("/hello/{name}")
  String hello(String name) {
    return "Hello " + name;
  }

}
```

```
@Client("/")
interface HelloClient {

  @Get("/hello/{name}")
  String hello(String name);

  // Implementation generated
  // at compile time
}

@Inject HelloClient helloClient;

helloClient.hello("Bob")
// returns "Hello Bob”
```

Micronaut: Dependency Injection

```
@Singleton //Bean definition generated at compile time
class WeatherService {
  Integer currentTemp() { //... }
}

@Controller('/weather')
class WeatherController {

    @Inject WeatherService weatherService
    //DI computed at compile time

    @Get("/")
    HttpResponse<Integer> currentTemp() {
      HttpResponse.ok(weatherService.currentTemp())
    }
}
```

Micronaut: Cloud Native

SERVICE DISCOVERY

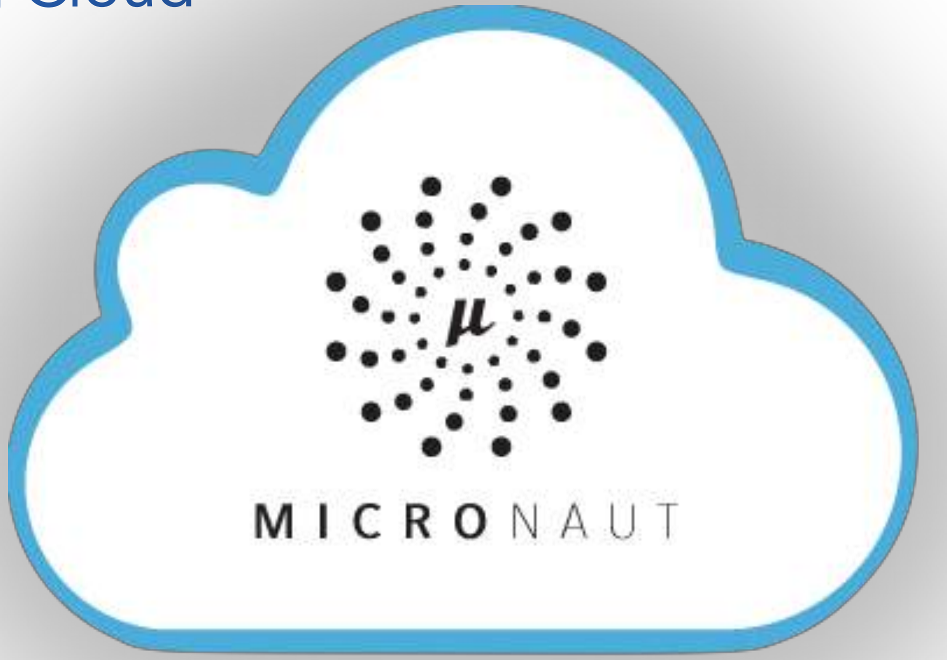
RETRY/FALLBACKS

CIRCUIT BREAKERS

```
//Lookup client from service-discovery registry
@Client(id="billing", path="/billing")
interface BillingClient { ... }

//Automatically retry failing calls
@Client("https://api.external.service")
@Retryable(attempts = '3', delay = '5ms')
interface ExternalApiClient { ... }


//Immediately fail after set number of failures
//Begin accepting calls after `reset` interval
@Singleton
@CircuitBreaker(attempts = '5', reset = '300ms')
class MyService { ... }
```

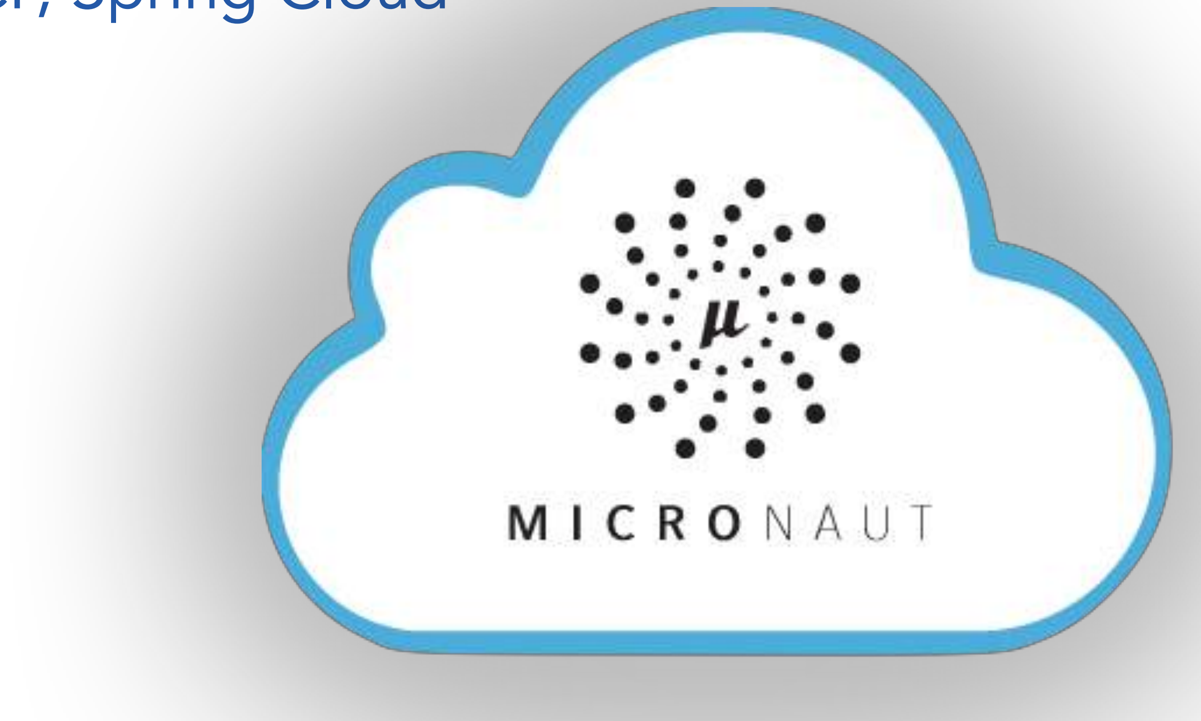
Micronaut: Cloud Native

- Cloud-Aware Environment Detection
- Cloud Provider Integration - AWS, GCP, Spring Cloud
- Metrics & Monitoring
- Distributed Configuration
- Distributed Tracing



Micronaut: Cloud Native

- Cloud-Aware Environment Detection
- Cloud Provider Integration - AWS, GCP, Spring Cloud
- Metrics & Monitoring
- Distributed Configuration
- **Distributed Tracing**



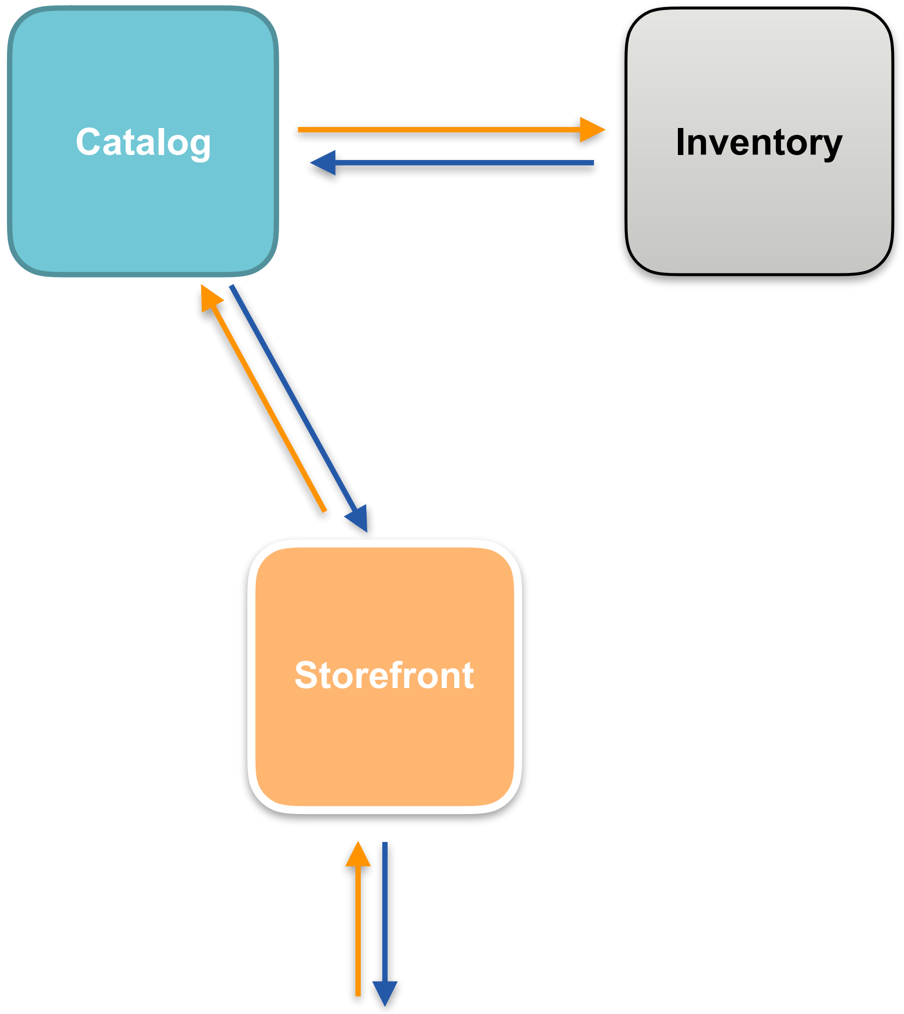
Agenda

1. Introduction to Distributed Tracing
2. Micronaut's Tracing Support
3. Setting up Tracing
4. Demo!

OBJECT
COMPUTING

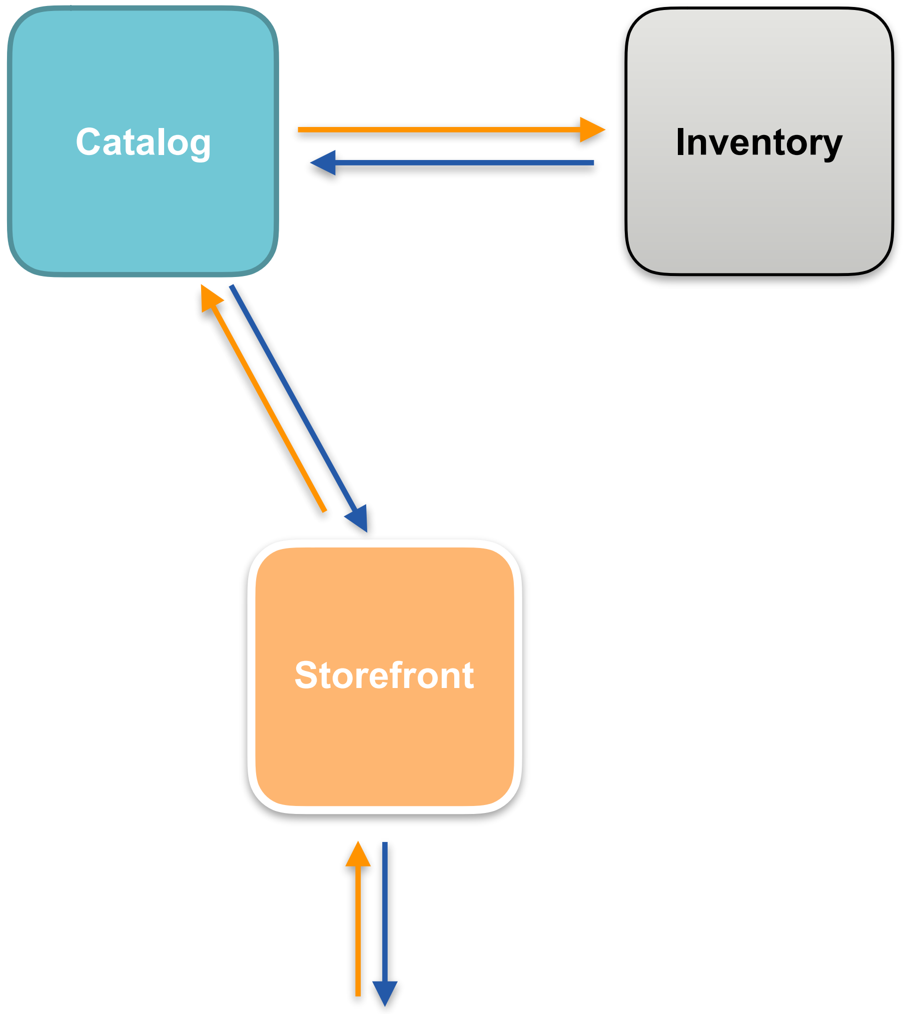
Introduction to Distributed Tracing

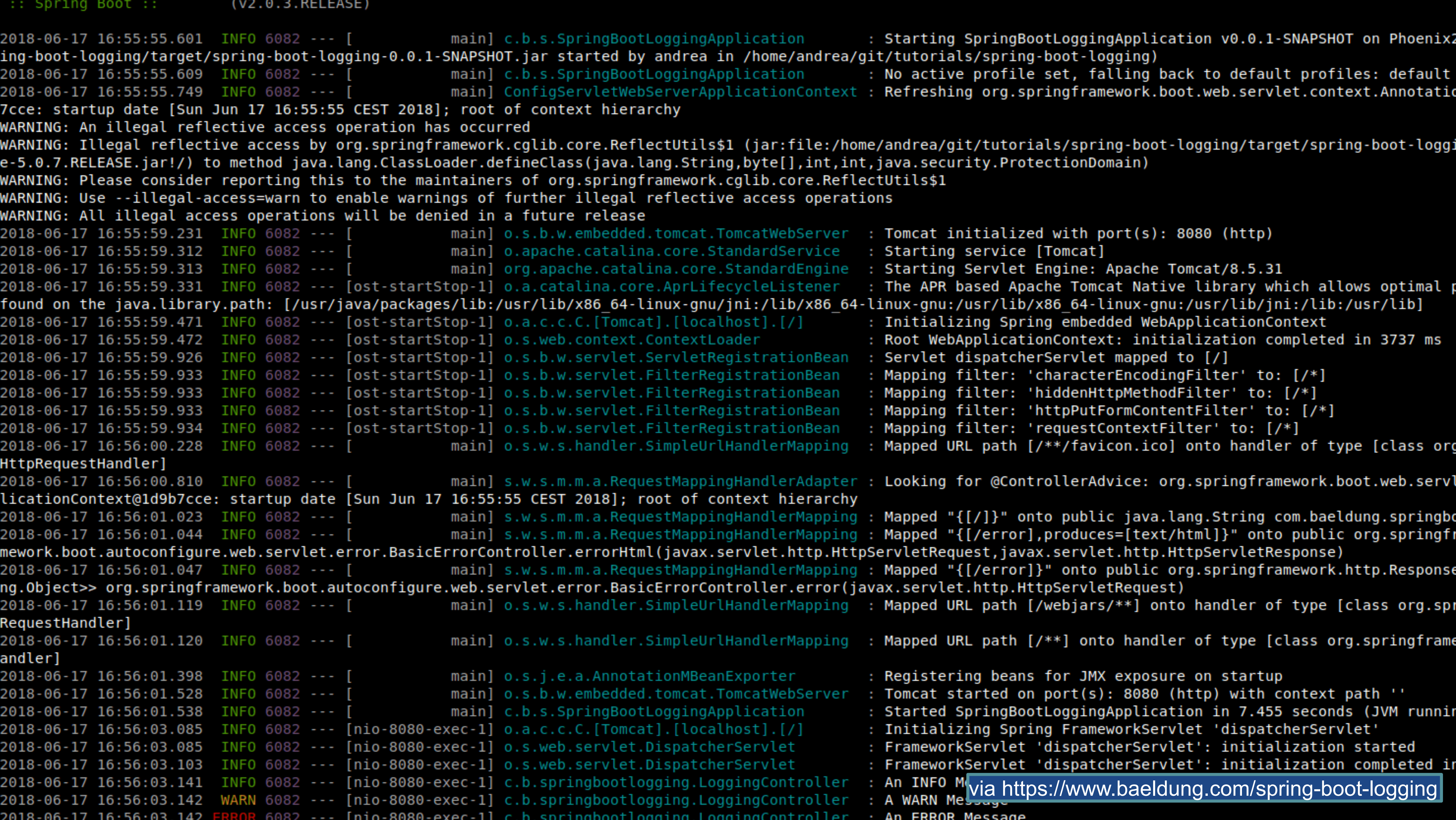
Why Distributed Tracing?



Why Distributed Tracing?

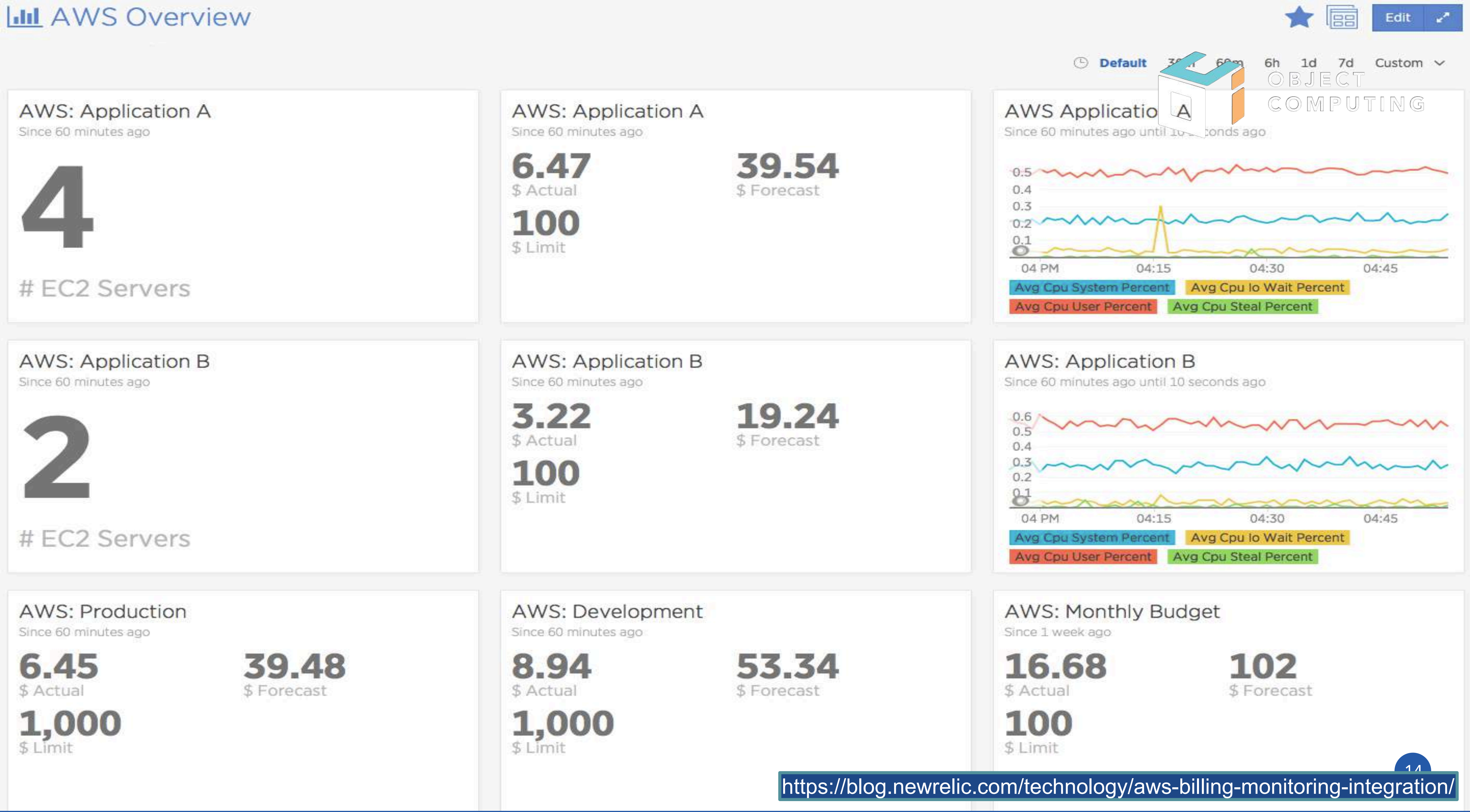
- Logging & Metrics only tell part of the story





```
 :: Spring Boot ::        (v2.0.3.RELEASE)

2018-06-17 16:55:55.601  INFO 6082 --- [           main] c.b.s.SpringBootLoggingApplication       : Starting SpringBootLoggingApplication v0.0.1-SNAPSHOT on Phoenix2
ing-boot-logging/target/spring-boot-logging-0.0.1-SNAPSHOT.jar started by andrea in /home/andrea/git/tutorials/spring-boot-logging)
2018-06-17 16:55:55.609  INFO 6082 --- [           main] c.b.s.SpringBootLoggingApplication       : No active profile set, falling back to default profiles: default
2018-06-17 16:55:55.749  INFO 6082 --- [           main] ConfigServletWebServerApplicationContext : Refreshing org.springframework.boot.web.servlet.context.Annotatio
7cce: startup date [Sun Jun 17 16:55:55 CEST 2018]; root of context hierarchy
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.springframework.cglib.core.ReflectUtils$1 (jar:file:/home/andrea/git/tutorials/spring-boot-logging/target/spring-boot-loggi
e-5.0.7.RELEASE.jar!/) to method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int,java.security.ProtectionDomain)
WARNING: Please consider reporting this to the maintainers of org.springframework.cglib.core.ReflectUtils$1
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
2018-06-17 16:55:59.231  INFO 6082 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2018-06-17 16:55:59.312  INFO 6082 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2018-06-17 16:55:59.313  INFO 6082 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/8.5.31
2018-06-17 16:55:59.331  INFO 6082 --- [ost-startStop-1] o.a.catalina.core.AprLifecycleListener   : The APR based Apache Tomcat Native library which allows optimal p
found on the java.library.path: [/usr/java/packages/lib:/usr/lib/x86_64-linux-gnu/jni:/lib/x86_64-linux-gnu:/usr/lib/x86_64-linux-gnu:/usr/lib/jni:/lib:/usr/lib]
2018-06-17 16:55:59.471  INFO 6082 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2018-06-17 16:55:59.472  INFO 6082 --- [ost-startStop-1] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 3737 ms
2018-06-17 16:55:59.926  INFO 6082 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Servlet dispatcherServlet mapped to [/]
2018-06-17 16:55:59.933  INFO 6082 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2018-06-17 16:55:59.933  INFO 6082 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2018-06-17 16:55:59.933  INFO 6082 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter' to: [/*]
2018-06-17 16:55:59.934  INFO 6082 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2018-06-17 16:56:00.228  INFO 6082 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**/favicon.ico] onto handler of type [class org
HttpRequestHandler]
2018-06-17 16:56:00.810  INFO 6082 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.web.servl
licationContext@1d9b7cce: startup date [Sun Jun 17 16:55:55 CEST 2018]; root of context hierarchy
2018-06-17 16:56:01.023  INFO 6082 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/]}" onto public java.lang.String com.baeldung.springbo
2018-06-17 16:56:01.044  INFO 6082 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springfr
mework.boot.autoconfigure.web.servlet.error.BasicErrorController.errorHtml(javax.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)
2018-06-17 16:56:01.047  INFO 6082 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.Response
ng.Object>> org.springframework.boot.autoconfigure.web.servlet.error.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2018-06-17 16:56:01.119  INFO 6082 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/webjars/**] onto handler of type [class org.spr
RequestHandler]
2018-06-17 16:56:01.120  INFO 6082 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping  : Mapped URL path [/**] onto handler of type [class org.springframe
andler]
2018-06-17 16:56:01.398  INFO 6082 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2018-06-17 16:56:01.528  INFO 6082 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2018-06-17 16:56:01.538  INFO 6082 --- [           main] c.b.s.SpringBootLoggingApplication       : Started SpringBootLoggingApplication in 7.455 seconds (JVM runnir
2018-06-17 16:56:03.085  INFO 6082 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring FrameworkServlet 'dispatcherServlet'
2018-06-17 16:56:03.085  INFO 6082 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet': initialization started
2018-06-17 16:56:03.103  INFO 6082 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : FrameworkServlet 'dispatcherServlet': initialization completed in
2018-06-17 16:56:03.141  INFO 6082 --- [nio-8080-exec-1] c.b.springbootlogging.LoggingController  : An INFO M
2018-06-17 16:56:03.142  WARN 6082 --- [nio-8080-exec-1] c.b.springbootlogging.LoggingController  : A WARN Message
2018-06-17 16:56:03.142 ERROR 6082 --- [nio-8080-exec-1] c.b.springbootlogging.LoggingController  : An ERROR Message
```

via https://www.baeldung.com/spring-boot-logging



AWS Overview

Edit

Default 30m 60m 6h 1d 7d Custom

AWS: Application A
Since 60 minutes ago
4
EC2 Servers

AWS: Application A
Since 60 minutes ago
6.47
$ Actual
39.54
$ Forecast
100
$ Limit

AWS: Application A
Since 60 minutes ago until 10 seconds ago

AWS: Application B
Since 60 minutes ago
2
EC2 Servers

AWS: Application B
Since 60 minutes ago
3.22
$ Actual
19.24
$ Forecast
100
$ Limit

AWS: Application B
Since 60 minutes ago until 10 seconds ago

AWS: Production
Since 60 minutes ago
6.45
$ Actual
39.48
$ Forecast
1,000
$ Limit

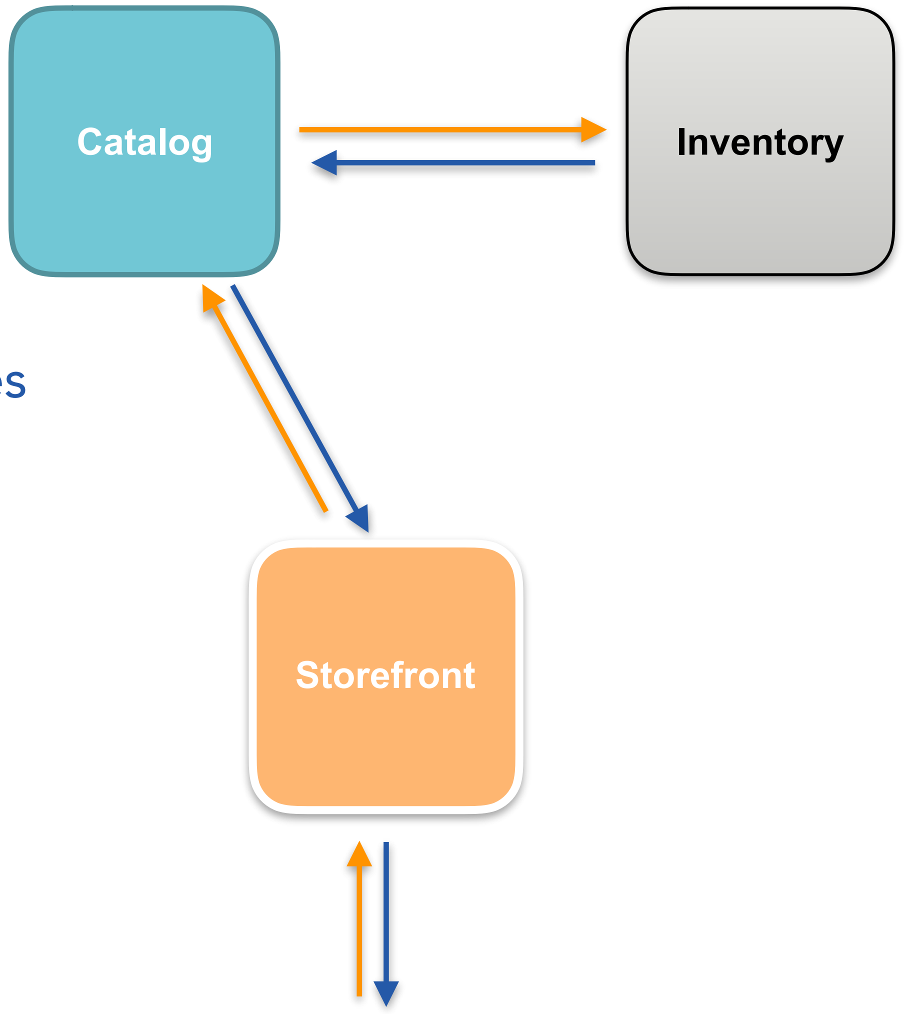
AWS: Development
Since 60 minutes ago
8.94
$ Actual
53.34
$ Forecast
1,000
$ Limit

AWS: Monthly Budget
Since 1 week ago
16.68
$ Actual
102
$ Forecast
100
$ Limit

https://blog.newrelic.com/technology/aws-billing-monitoring-integration/

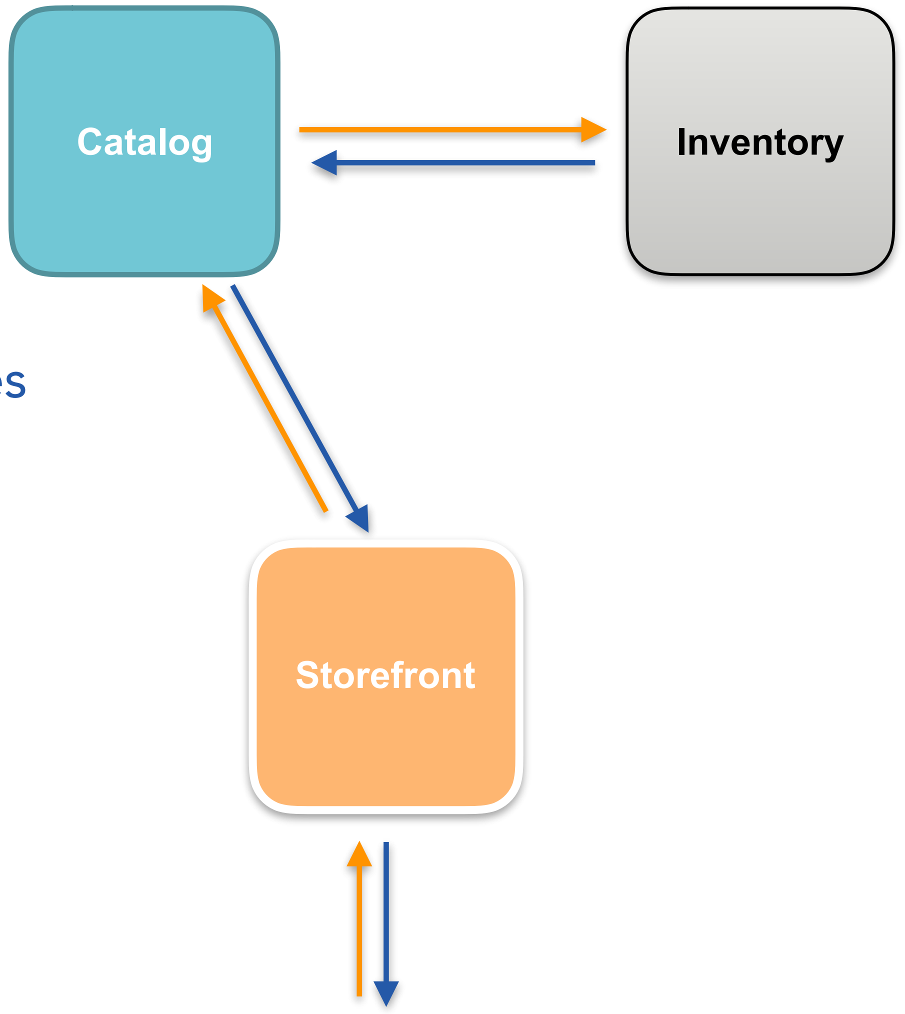
Why Distributed Tracing?

- Logging & Metrics only tell part of the story
- Microservice architecture - one request, many services



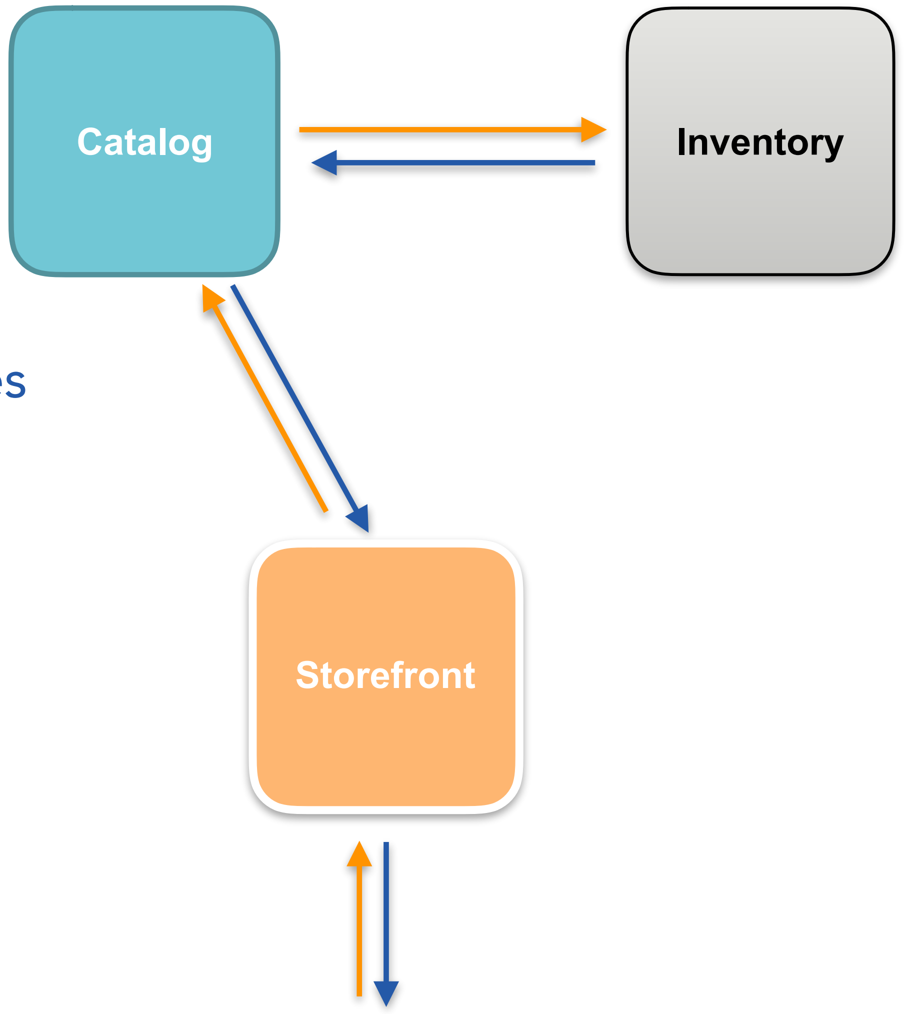
Why Distributed Tracing?

- Logging & Metrics only tell part of the story
- Microservice architecture - one request, many services
- Request-first monitoring



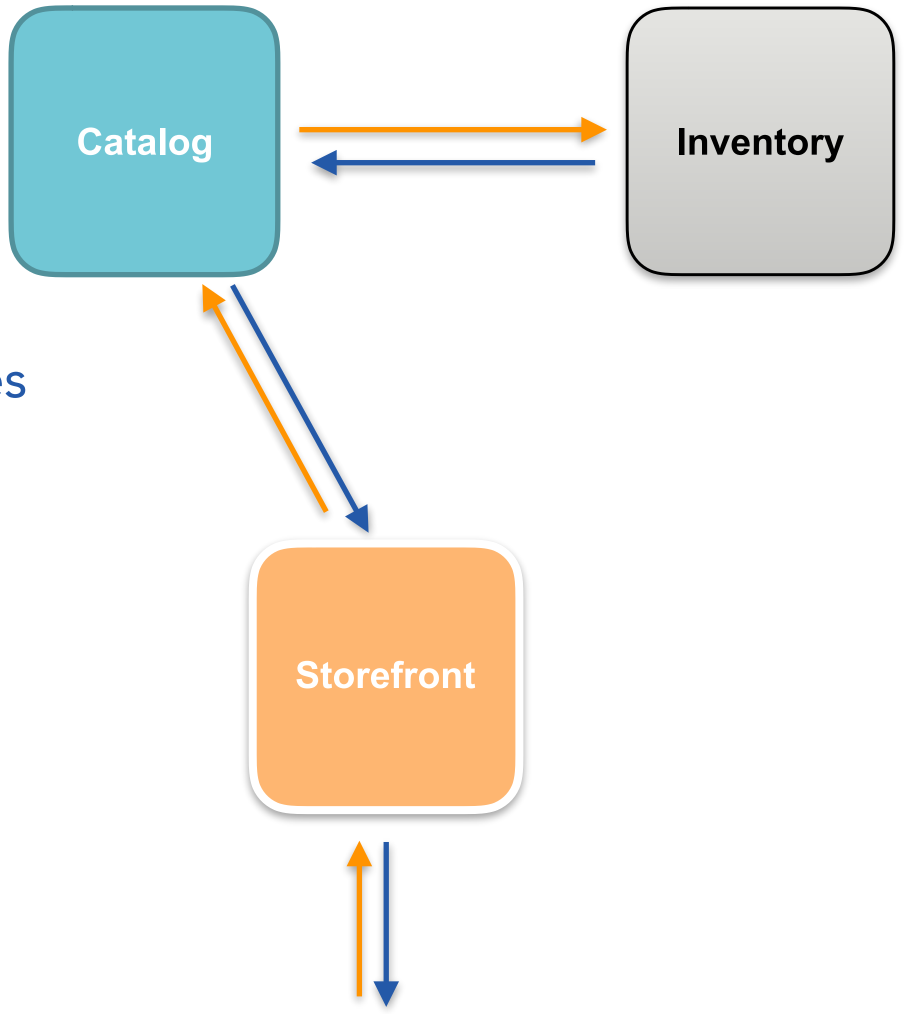
Why Distributed Tracing?

- Logging & Metrics only tell part of the story
- Microservice architecture - one request, many services
- Request-first monitoring
- Understanding interactions



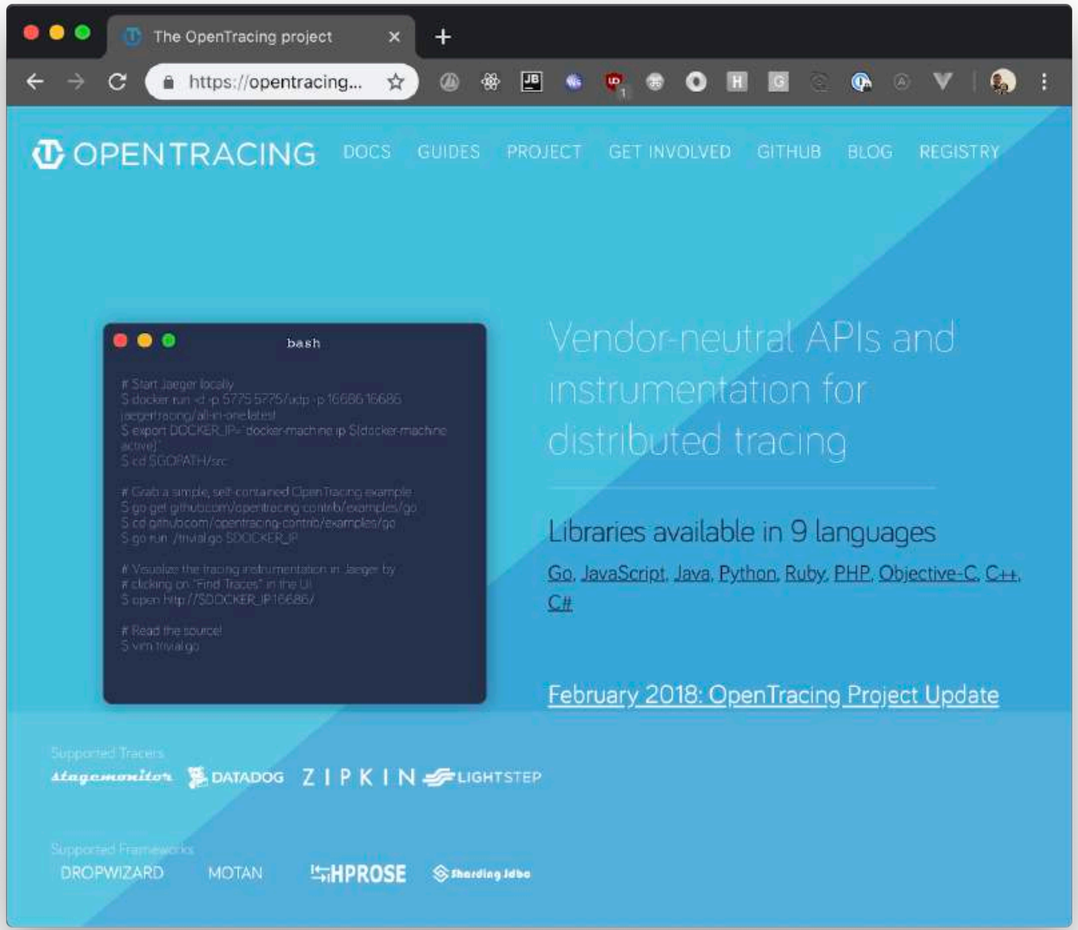
Why Distributed Tracing?

- Logging & Metrics only tell part of the story
- Microservice architecture - one request, many services
- Request-first monitoring
- Understanding interactions
- Identifying inefficiencies



Open Tracing

- Open source specification
- Defines common interface for sending and receiving spans
- Vendor-neutral
- https://opentracing.io



```
# Start Jaeger locally
$ docker run -d -p 5775:5775/udp -p 16686:16686
jaegertracing/all-in-one:latest
$ export DOCKER_IP=`docker-machine ip $(docker-machine
active)`
$ cd $GOPATH/src

# Grab a simple, self-contained OpenTracing example
$ go get github.com/opentracing-contrib/examples/go
$ cd github.com/opentracing-contrib/examples/go
$ go run ./trivial.go $DOCKER_IP

# Visualize the tracing instrumentation in Jaeger by
# clicking on "Find Traces" in the UI
$ open http://$DOCKER_IP:16686/

# Read the source!
$ vim trivial.go
```

Vendor-neutral APIs and instrumentation for distributed tracing

Libraries available in 9 languages

Go, JavaScript, Java, Python, Ruby, PHP, Objective-C, C++, C#

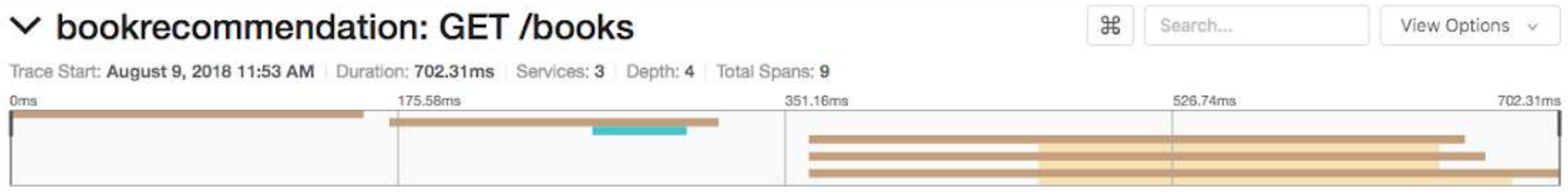
February 2018: OpenTracing Project Update

Distributed 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.

https://opentracing.io/docs/overview/what-is-tracing

Tracing Terminology

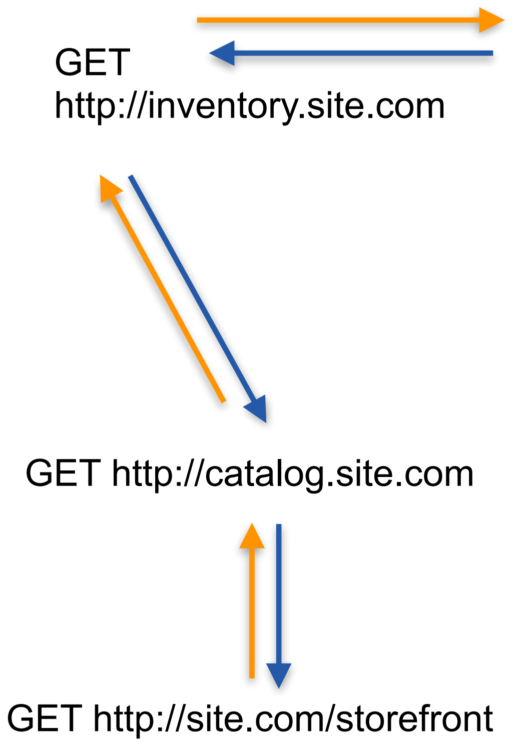
- **Trace** - record of a "request" to the system, from start to finish
    - Each trace has a unique trace ID



bookrecommendation: GET /books

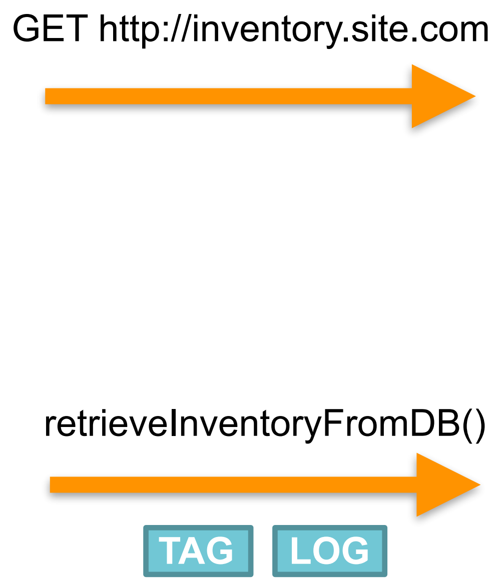
Trace Start: August 9, 2018 11:53 AM | Duration: 702.31ms | Services: 3 | Depth: 4 | Total Spans: 9

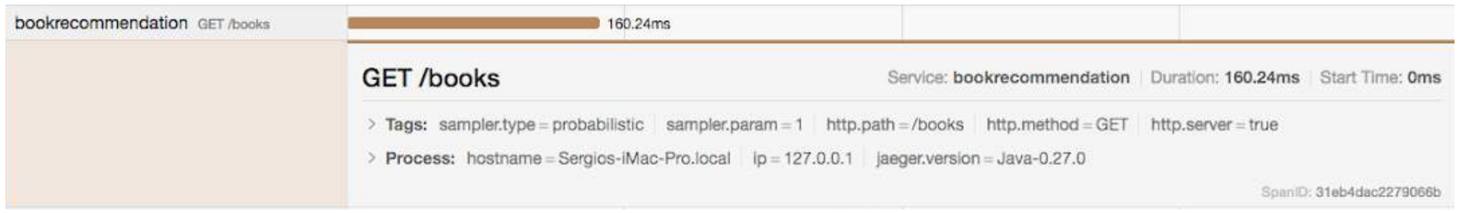
0ms 175.58ms 351.16ms 526.74ms 702.31ms


GET http://inventory.site.com
GET http://catalog.site.com
GET http://site.com/storefront

Tracing Terminology

- **Span** - a segment of a trace within a service
- One service can output **multiple spans** to a single trace
- Typically need at least one span per service to be useful
- Not necessarily network requests - inter-service calls can emit spans

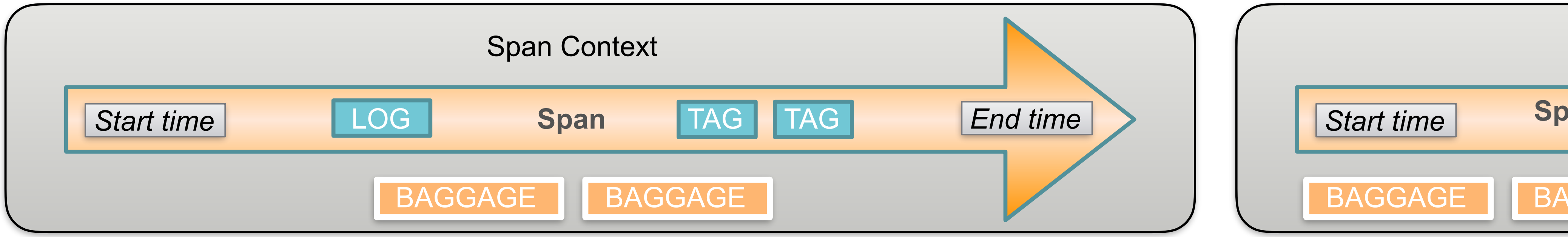




```
bookrecommendation GET /books    160.24ms

GET /books                 Service: bookrecommendation | Duration: 160.24ms | Start Time: 0ms
> Tags: sampler.type = probabilistic | sampler.param = 1 | http.path = /books | http.method = GET | http.server = true
> Process: hostname = Sergios-iMac-Pro.local | ip = 127.0.0.1 | jaeger.version = Java-0.27.0
                                                                          SpanID: 31eb4dac2279066b
```

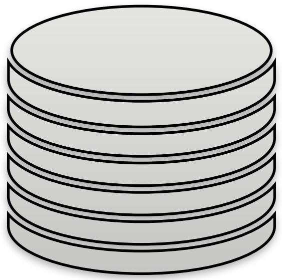
Tracing Terminology

- **Span** -
- Spans can contain **metadata:** tags and logs
- Spans have a **context** that holds state across the trace
  - E.g, **baggage items** are key/value pairs that can be stored/retrieved across all spans in a traces



Tracing Terminology

- **Collector** - agent or runtime that receives and persists tracing data
  - Typically a standalone service, e.g, running in a Docker container

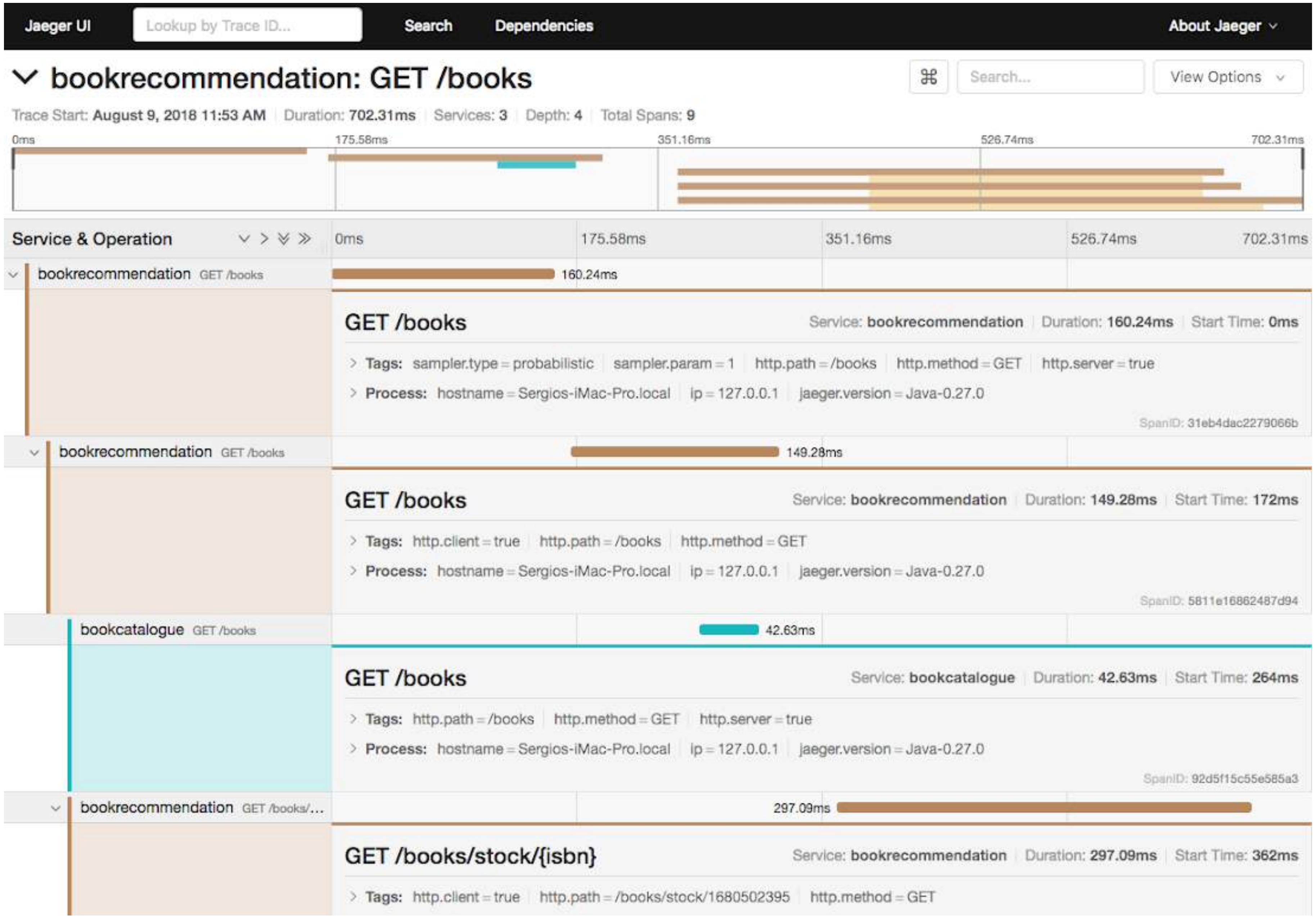


https://www.jaegertracing.io https://zipkin.io

Tracing Terminology

- **Sampling** - how many requests should be traced
  - Impractical to trace every single request; a sampling percentage should be configured

```java
import java.util.Map;
import io.opentracing.mock.MockTracer;
import io.opentracing.mock.MockSpan;
import io.opentracing.tags.Tags;

// Initialize MockTracer with the default values.
MockTracer tracer = new MockTracer();

// Create a new Span, representing an operation.
MockSpan span = tracer.buildSpan("foo").start();

// Add a tag to the Span.
span.setTag(Tags.COMPONENT, "my-own-application");

// Finish the Span.
span.finish();

// Analize the saved Span.
System.out.println("Operation name = " + span.operationName());
System.out.println("Start = " + span.startMicros());
System.out.println("Finish = " + span.finishMicros());

// Inspect the Span's tags.
Map<String, Object> tags = span.tags();
```


Jaeger UI | Lookup by Trace ID... | Search | Dependencies | About Jaeger

bookrecommendation: GET /books

Trace Start: **August 9, 2018 11:53 AM** | Duration: **702.31ms** | Services: **3** | Depth: **4** | Total Spans: **9**

0ms | 175.58ms | 351.16ms | 526.74ms | 702.31ms

Service & Operation

bookrecommendation GET /books — 160.24ms

GET /books — Service: **bookrecommendation** | Duration: **160.24ms** | Start Time: **0ms**

Tags: sampler.type = probabilistic | sampler.param = 1 | http.path = /books | http.method = GET | http.server = true

Process: hostname = Sergios-iMac-Pro.local | ip = 127.0.0.1 | jaeger.version = Java-0.27.0

SpanID: 31eb4dac2279066b

bookrecommendation GET /books — 149.28ms

GET /books — Service: **bookrecommendation** | Duration: **149.28ms** | Start Time: **172ms**

Tags: http.client = true | http.path = /books | http.method = GET

Process: hostname = Sergios-iMac-Pro.local | ip = 127.0.0.1 | jaeger.version = Java-0.27.0

SpanID: 5811e16862487d94

bookcatalogue GET /books — 42.63ms

GET /books — Service: **bookcatalogue** | Duration: **42.63ms** | Start Time: **264ms**

Tags: http.path = /books | http.method = GET | http.server = true

Process: hostname = Sergios-iMac-Pro.local | ip = 127.0.0.1 | jaeger.version = Java-0.27.0

SpanID: 92d5f15c55e585a3

bookrecommendation GET /books/... — 297.09ms

GET /books/stock/{isbn} — Service: **bookrecommendation** | Duration: **297.09ms** | Start Time: **362ms**

Tags: http.client = true | http.path = /books/stock/1680502395 | http.method = GET

OBJECT COMPUTING

Micronaut’s Tracing Support

Micronaut's Distributed Tracing

- The **micronaut-tracing** library provides native support for Open Tracing implementations
- Special configurations included for **Jaeger** and **Zipkin** (same API works for both)
- Annotation-based API for interacting with tracer spans

Distributed Tracing Annotations

@NewSpan - Start a new span at this method

```
@NewSpan("productList")
List<ProductDetails> productList() {
    //Get products
}
```

Distributed Tracing Annotations

`@ContinueSpan` - Continue the current span across this method

```
@ContinueSpan
@Get("/{id}")
HttpResponse<ProductDetails> show(Serializable id) {
    //Get product
}
```

Distributed Tracing Annotations

@SpanTag - Add a metadata “tag” to the active (current or new) span

```
@NewSpan("hits")
@Get("/{productNumber}")
HttpResponse<Integer> hits(
  @SpanTag("product") String productNumber
) {

    //Get result…

    HttpResponse.ok(result)
}
```

OpenTracing.io Tracer Interface

- Standard **interface** for managing/manipulating Open Tracing traces
- Micronaut registers a `io.opentracing.Tracer` **bean** that can be injected into any class
- Tracer bean exposes the entire Open Tracing API, including access to the `SpanContext`, "baggage items", and more

OpenTracing.io Tracer Interface

```java
public interface Tracer {
    ScopeManager scopeManager();
    Span activeSpan();
    SpanBuilder buildSpan(String operationName);
    <C> void inject(SpanContext spanContext, Format<C> format, C carrier);
    <C> SpanContext extract(Format<C> format, C carrier);
    interface SpanBuilder {
        SpanBuilder asChildOf(SpanContext parent);
        SpanBuilder asChildOf(Span parent);
        SpanBuilder addReference(String referenceType, SpanContext referencedContext);
        SpanBuilder ignoreActiveSpan();
        SpanBuilder withTag(String key, String value); //
        SpanBuilder withStartTimestamp(long microseconds);
        Scope startActive(boolean finishSpanOnClose);
        Span start();
    }
}
```

OpenTracing.io Tracer Interface

```java
public interface Tracer {
    ScopeManager scopeManager();
    Span activeSpan();
    SpanBuilder buildSpan(String operationName);
    <C> void inject(SpanContext spanContext, Format<C> format, C carrier);
    <C> SpanContext extract(Format<C> format, C carrier);
    interface SpanBuilder {
        SpanBuilder asChildOf(SpanContext parent);
        SpanBuilder asChildOf(Span parent);
        SpanBuilder addReference(String referenceType, SpanContext referencedContext);
        SpanBuilder ignoreActiveSpan();
        SpanBuilder withTag(String key, String value); //
        SpanBuilder withStartTimestamp(long microseconds);
        Scope startActive(boolean finishSpanOnClose);
        Span start();
    }
}
```

OpenTracing.io Tracer Interface

```
import io.opentracing.Tracer
import javax.inject.Inject
import javax.inject.Singleton

@Singleton
class StorefrontService {

    @Inject
    Tracer tracer

    @NewSpan("productList")
    List<ProductDetails> productList() {

        products = //Get products

        tracer.activeSpan().setTag("count", products.size())

        products
    }
```

OBJECT COMPUTING

Setting up Tracing

Micronaut CLI

- The **Micronaut CLI** includes support for generating projects with tracing pre-configured
- CLI features available for **Zipkin** and **Jaeger**

```
~ mn create-app my-app –features tracing-jaeger //or tracing-zipkin
```

Adding Tracing to Your Project

1. Add **micronaut-tracing** dependency,
   plus desired tracing implementation

```
compile "io.micronaut:micronaut-tracing"
runtime 'io.zipkin.brave:brave-instrumentation-http'
runtime 'io.zipkin.reporter2:zipkin-reporter'
compile 'io.opentracing.brave:brave-opentracing'
```

2. Add configuration to **application.yml** file

```
tracing:
    zipkin:
        http:
            url: http://localhost:9411
        enabled: true
        sampler:
            probability: 1
```

Adding Tracing to Your Project

1. Add **micronaut-tracing** dependency,
   plus desired tracing implementation

```
compile “io.micronaut:micronaut-tracing"
compile 'io.jaegertracing:jaeger-thrift'
```

2. Add configuration to **application.yml** file

```
tracing:
    jaeger:
        enabled: true
        sampler:
            probability: 1
```

Running Tracing Agent via Docker

Jaeger

```
$ docker run -d \
  -e COLLECTOR_ZIPKIN_HTTP_PORT=9411 \
  -p 5775:5775/udp \
  -p 6831:6831/udp \
  -p 6832:6832/udp \
  -p 5778:5778 \
  -p 16686:16686 \
  -p 14268:14268 \
  -p 9411:9411 \
  jaegertracing/all-in-one:1.6
```

http://localhost:16686

Zipkin

```
$ docker run -d
-p 9411:9411
openzipkin/zipkin
```

http://localhost:9411

OBJECT COMPUTING

Demo

OBJECT COMPUTING

Thank you!

LEARN MORE ABOUT OCI EVENTS AND TRAINING

OBJECT COMPUTING

Events:

- objectcomputing.com/events

Training:

- objectcomputing.com/training
- grailstraining.com
- micronauttraining.com

Or email info@ocitraining.com to schedule a custom training program for your team online, on site, or in our state-of-the-art, Midwest training lab.

OBJECT
COMPUTING

CONNECT WITH US

1+ (314) 579-0066

@objectcomputing

objectcomputing.com