

{ Tracing performance with some help of Sleuth, Zipkin & ELK }

Rafaëla Breed



Rafaëla Breed

- Software engineer @ Luminis Amsterdam
- Java developer
- Spring(Boot)
- Photography, traveling

Hmmm... breakfast



New company, new application



#jussayin

Let's just start developing our application



cucumber

Everything works fine, but...

...there comes the tester/business/client: 

“It does not work” 

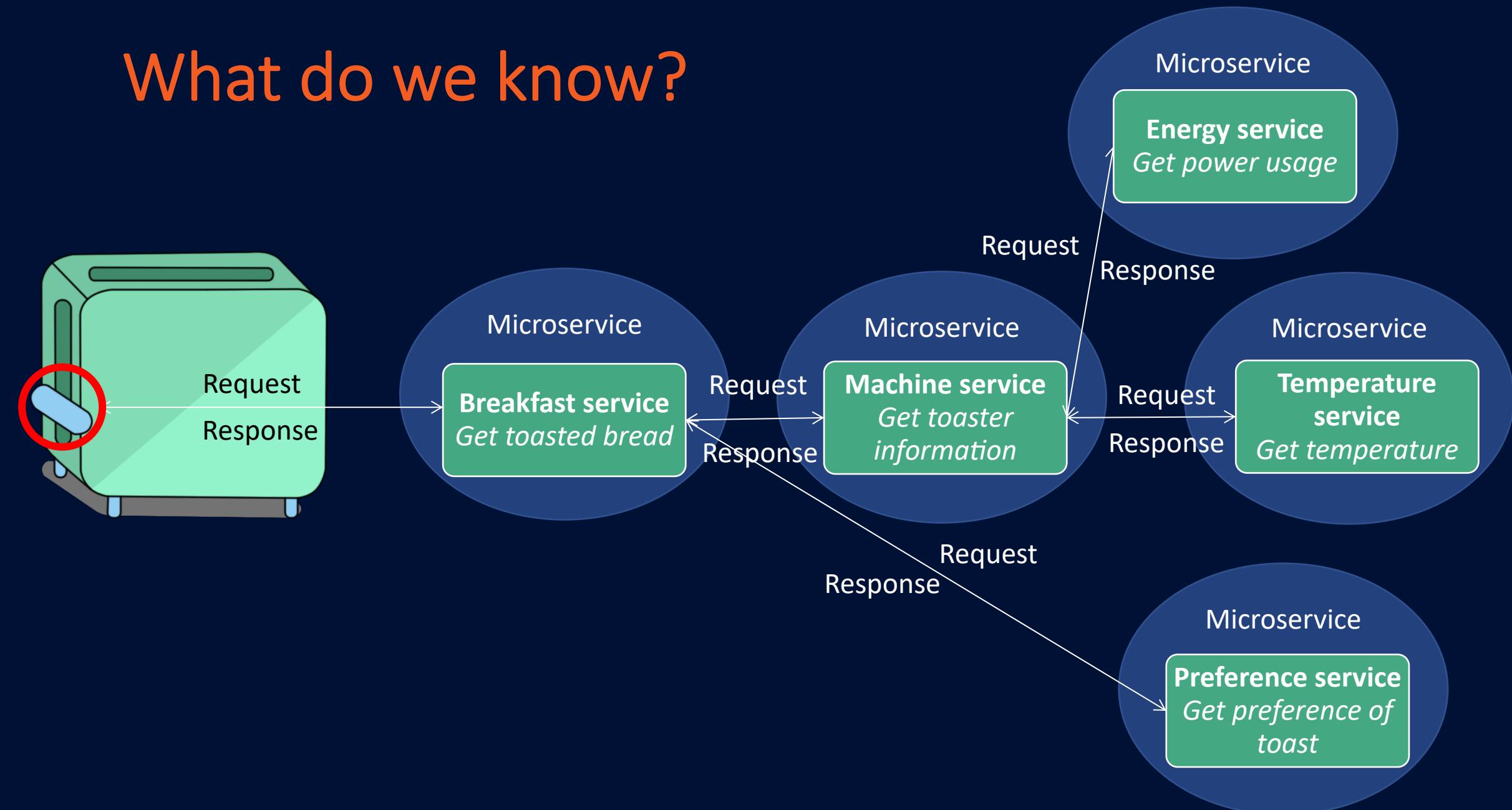
“So, what does not work?” 

“Well, it is slow” 

“All right, do you know which step is being slow exactly?” 

“When I push the toast button, it takes 5 seconds to get the results. That is way too slow. We have to service our customers as fast as possible and” 

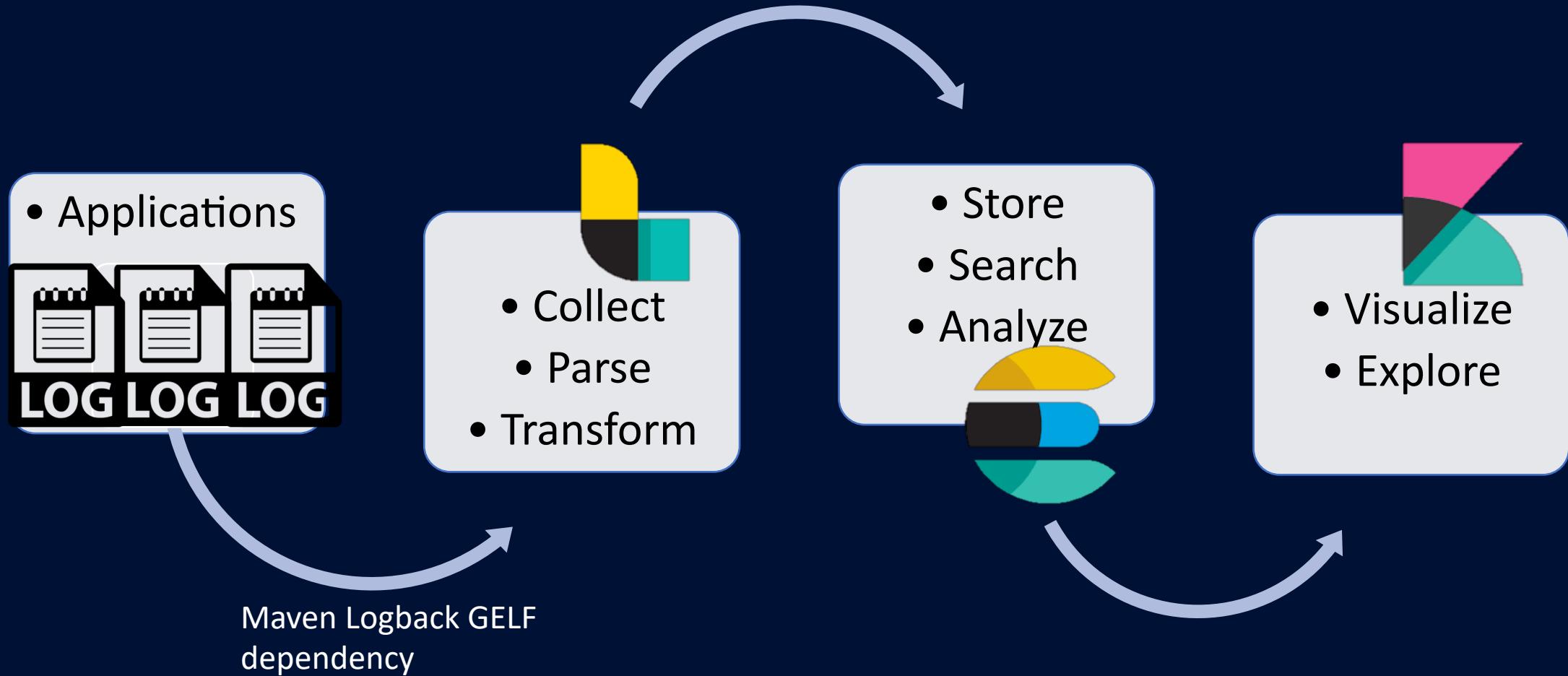
What do we know?



What do we have?

OR Elastic
Stack

ELK: ElasticSearch, Logstash, Kibana



Kibana

7 hits

Search... (00 AND extension:PHP)

levelName: "INFO" traceld: "37f496118c1b5c03" Add a filter +

Selected Fields

- t applicationName
- t levelName
- t message
- t parentId
- t spanId
- t traceld

Available Fields

Popular

- t X-B3-Traceld
- @versi
- @ Fields used in logs
- t X-B3-Curvid
- t X-B3-ParentSpanId
- t X-B3-SpanId
- t X-Span-Export
- t _id
- t index

Filters

November 2nd 2018, 00:00:00.000 - November 2nd 2018, 23:59:59.999 — Auto

Count

01:00 04:00 07:00 10:00 13:00 16:00 19:00 22:00

@timestamp per 30 minutes

Time

	applicationName	levelName	message
▶ November 2nd 2018, 12:01:43.043	BreakfastService	INFO	Returning toasted bread
▶ November 2nd 2018, 12:01:38.013	PreferenceService	INFO	Getting information how the bread should be toasted
▶ November 2nd 2018, 12:01:37.795			
▶ November 2nd 2018, 12:01:37.749	EnergyService	INFO	Get energy
▶ November 2nd 2018, 12:01:37.384	TemperatureService	INFO	Getting temperature of toaster
▶ November 2nd 2018, 12:01:37.259	MachineService	INFO	Getting information about power used & temperature of toaster
▶ November 2nd 2018, 12:01:37.078	BreakfastService	INFO	Getting information about power used & temperature of toaster

Log messages of the applications

Let's try & dive in our logging

Summary of demo

- Easy search & filter by ELK
- But... Not easy to track specific events... yet

What we would like to have

- Correlation between operations
- Each starting operation => unique id
- Each next operation => unique id

What Sleuth can offer

- Depicted from **Dapper** Infrastructure, **Zipkin**, **Htrace**
- Implementable via **Maven/Gradle**
- **Brave** is used as tracing library
- Every **first remote call** starts a **trace** (aka uber-span)
- Every **following remote call** starts a **span** (aka child span)
- Traces are passed to *next service(s)*

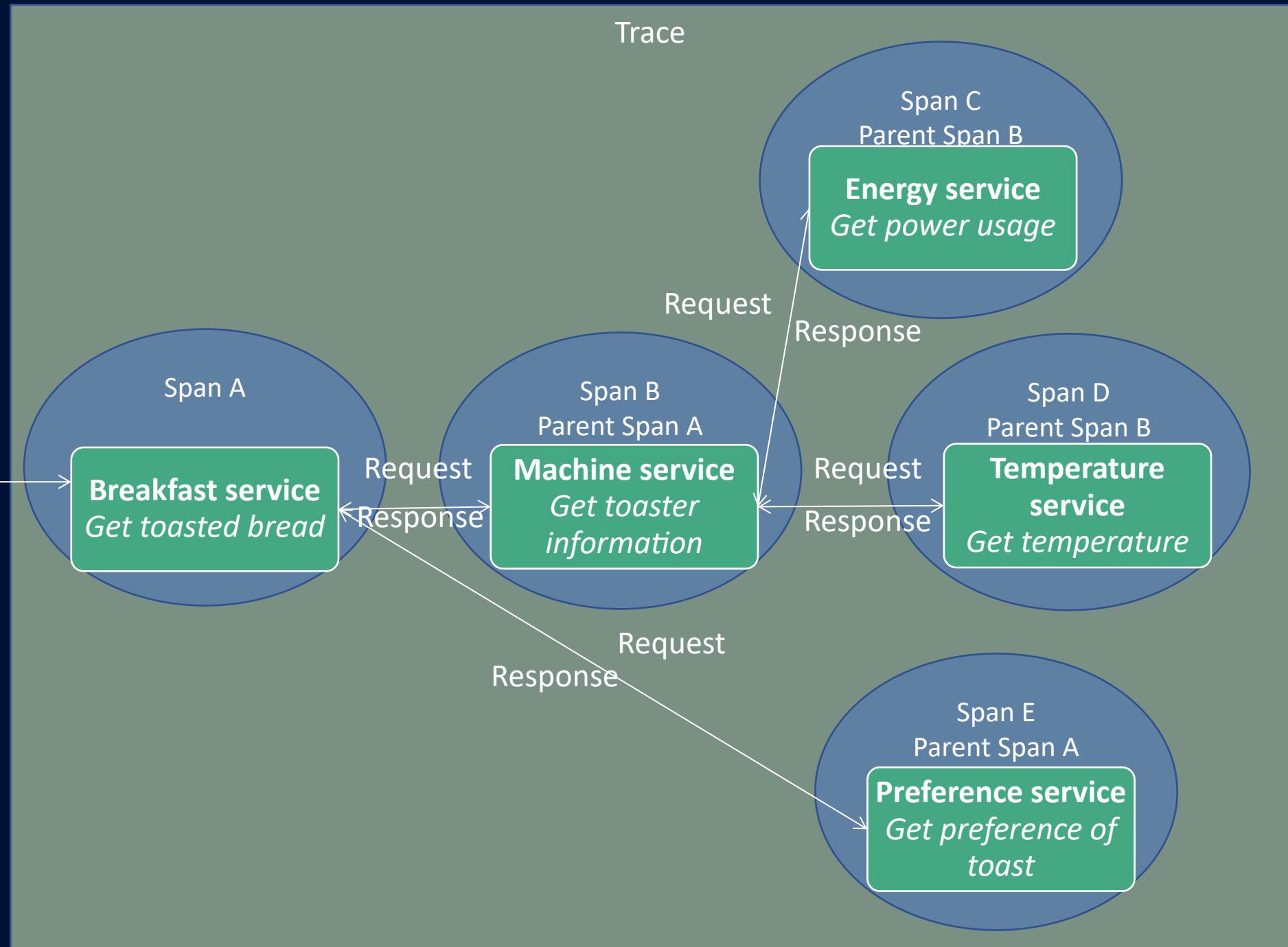
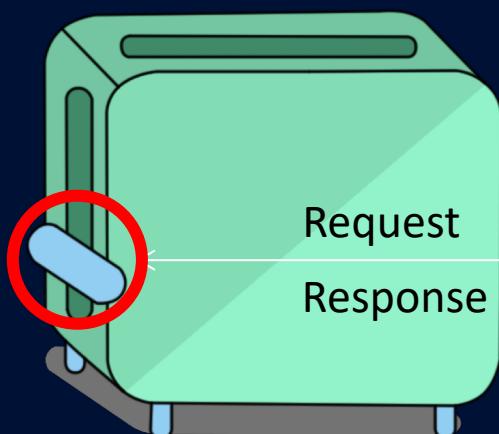
Adding Sleuth to your project

- To every SpringBoot app we add:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-sleuth</artifactId>
      <version>${spring.cloud.version}</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
<dependencies>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-sleuth</artifactId>
  </dependency>
</dependencies>
```

- Or take a look at <https://cloud.spring.io/spring-cloud-sleuth/>
- Using custom log appender(s)? Add the properties X-B3-TracId and X-B3-SpanId to your configuration!

Summary of demo



Summary of demo

Time ▾	applicationName	levelName	message	traceId	spanId	parentId
▶ November 2nd 2018, 12:01:43.043	BreakfastService	INFO	Returning toasted bread	37f496118c1b5 c03	37f496118c1b5 c03	-
▶ November 2nd 2018, 12:01:38.013	PreferenceService	INFO	Getting information how the bread should be toasted	37f496118c1b5 c03	fe5f2b53e2cbf d16	37f496118c1b5 c03
▶ November 2nd 2018, 12:01:37.795	MachineService	INFO	Returning information about power used & temperature	37f496118c1b5 c03	361368a541a66 a2a	37f496118c1b5 c03
▶ November 2nd 2018, 12:01:37.749	EnergyService	INFO	Get energy	37f496118c1b5 c03	a38d6c2adfb77 40f	361368a541a66 a2a
▶ November 2nd 2018, 12:01:37.384	TemperatureService	INFO	Getting temperature of toaster	37f496118c1b5 c03	131d778df7fa2 ef2	361368a541a66 a2a
▶ November 2nd 2018, 12:01:37.259	MachineService	INFO	Getting information about power used & temperature of toaster	37f496118c1b5 c03	361368a541a66 a2a	37f496118c1b5 c03
▶ November 2nd 2018, 12:01:37.078	BreakfastService	INFO	Getting a toasted bread with information	37f496118c1b5 c03	37f496118c1b5 c03	-

Zipkin

- Developed by Twitter in 2012
- Adapted and open-sourced in 2015 as OpenZipkin
- Distributed tracing system
- Trace down your system(s)

Adding Zipkin client to your project

- To every SpringBoot app we add:

```
<!-- Assuming you already added the spring cloud sleuth dependency in  
your dependencyManagement -->
```

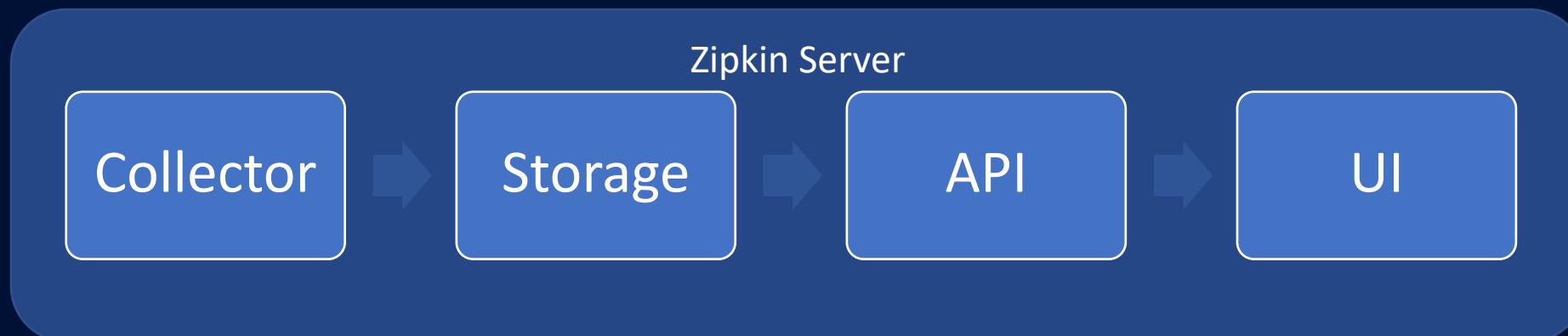
```
<dependency>  
  <groupId>org.springframework.cloud</groupId>  
  <artifactId>spring-cloud-starter-zipkin</artifactId>  
</dependency>
```

Sampler for Zipkin

- *Do you want to put each request into Zipkin?*
- Up-front decision
- Probability 0...1 → chance 0 to 100%
- Default = ProbabilityBasedSampler 0.1
- ALWAYS_SAMPLE
- Custom declarative sampling
- Custom sampling on request
- Rate-limiting sampler
- X-B3-Flags overrides (client request header)

Getting a Zipkin server

- Github project
- Docker images (production ready)
- Cloud solutions for AWS/Azure available

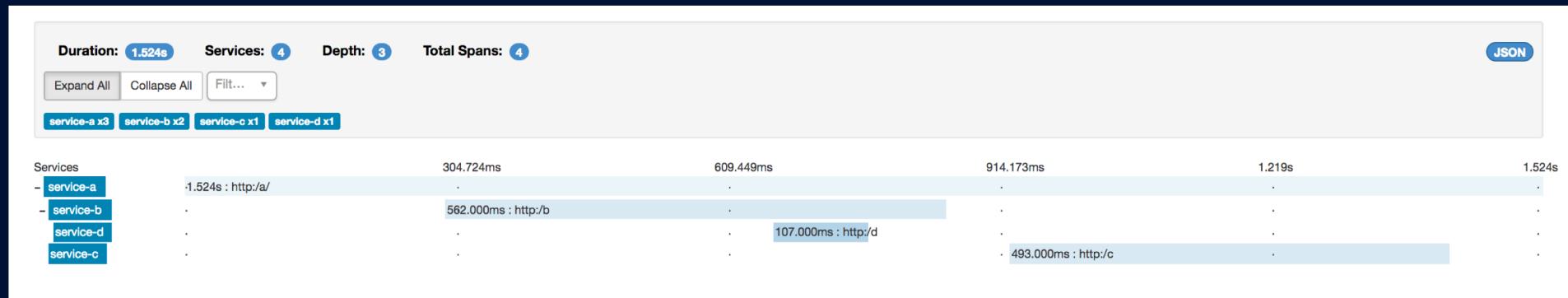


Summary of demo

- The **Zipkin client** sends enriched information of Sleuth to your **Zipkin Server**
- The Zipkin Server provides a **user interface** in which you can overview the details of the events logged by Sleuth/Zipkin Client

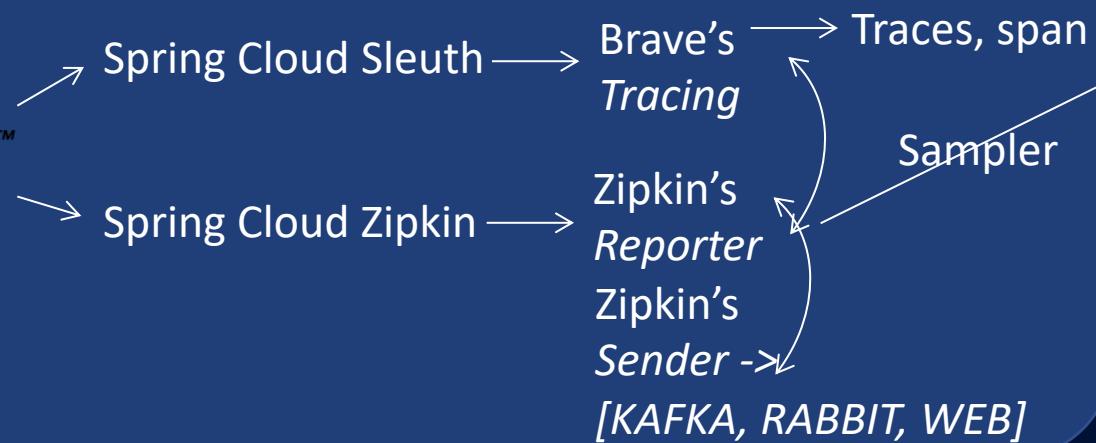
Summary of demo

- The user interface with clear **graphics** of events with times



Overview what will happen (a bit under the hood)

Spring boot application



Zipkin server







Our Cucumber tests

- TDD, business scenario's
- Logs tests & apps available

Feature: get a breakfast

As a customer

I want to get a breakfast

So that I will start the day well

Scenario:

Given a customer

When a customer want a toasted bread

And a good cup of coffee

And a fried egg

And some bacon

Then he gets a wonderful breakfast

Our test log

```
org.awaitility.core.ConditionTimeoutException: Condition with lambda expression in eu.luminis.breed.sleuthzipkin.selenium.BreakfastPage was not fulfilled within 5 seconds.
  at org.awaitility.core.ConditionAwaiter.await(ConditionAwaiter.java:145)
  at org.awaitility.core.CallableCondition.await(CallableCondition.java:79)
  at org.awaitility.core.CallableCondition.await(CallableCondition.java:27)
  at org.awaitility.core.ConditionFactory.until(ConditionFactory.java:902)
  at org.awaitility.core.ConditionFactory.until(ConditionFactory.java:860)
  at eu.luminis.breed.sleuthzipkin.selenium.BreakfastPage.fryEgg(BreakfastPage.java:47)
  at eu.luminis.breed.sleuthzipkin.selenium.PageStepDefinitions.aFriedEgg(PageStepDefinitions.java:48)
  at *.a fried egg(file:src/test/resources/breakfast-service.feature:10)

[ERROR] Tests run: 1, Failures: 0, Errors: 1, Skipped: 0, Time elapsed: 19.43 s <<< FAILURE! - in eu.luminis.breed.sleuthzipkin.selenium.BreakfastPageTest
[ERROR] EMPTY_NAME(get a breakfast)  Time elapsed: 18.885 s  <<< ERROR!
org.awaitility.core.ConditionTimeoutException: Condition with lambda expression in eu.luminis.breed.sleuthzipkin.selenium.BreakfastPage was not fulfilled within 5 seconds.

[INFO]
[INFO] Results:
[INFO]
[ERROR] Errors:
[ERROR]   Condition with lambda expression in eu.luminis.breed.sleuthzipkin.selenium.BreakfastPage was not fulfilled within 5 seconds.
[INFO]
[ERROR] Tests run: 1, Failures: 0, Errors: 1, Skipped: 0
[INFO]
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 24.414 s
[INFO] Finished at: 2019-04-01T13:42:38+02:00
[INFO] Final Memory: 41M/339M
[INFO] -----
```

Our test report

▼ Feature: get a breakfast

As a customer

I want to get a breakfast

So that I will start the day well

▼ Scenario:

Given a customer

When a customer want a toasted bread

And a good cup of coffee

And a fried egg

```
org.awaitility.core.ConditionTimeoutException: Condition with lambda expression in eu.luminis.breed.sleuthzipkin.selenium.BreakfastPage was not fulfilled
    at org.awaitility.core.ConditionAwaiter.await(ConditionAwaiter.java:145)
    at org.awaitility.core.CallableCondition.await(CallableCondition.java:79)
    at org.awaitility.core.CallableCondition.await(CallableCondition.java:27)
    at org.awaitility.core.ConditionFactory.until(ConditionFactory.java:902)
    at org.awaitility.core.ConditionFactory.until(ConditionFactory.java:860)
    at eu.luminis.breed.sleuthzipkin.selenium.BreakfastPage.fryEgg(BreakfastPage.java:47)
    at eu.luminis.breed.sleuthzipkin.selenium.PageStepDefinitions.aFriedEgg(PageStepDefinitions.java:48)
    at *.a fried egg(file:src/test/resources/breakfast-service.feature:10)
```

And some bacon

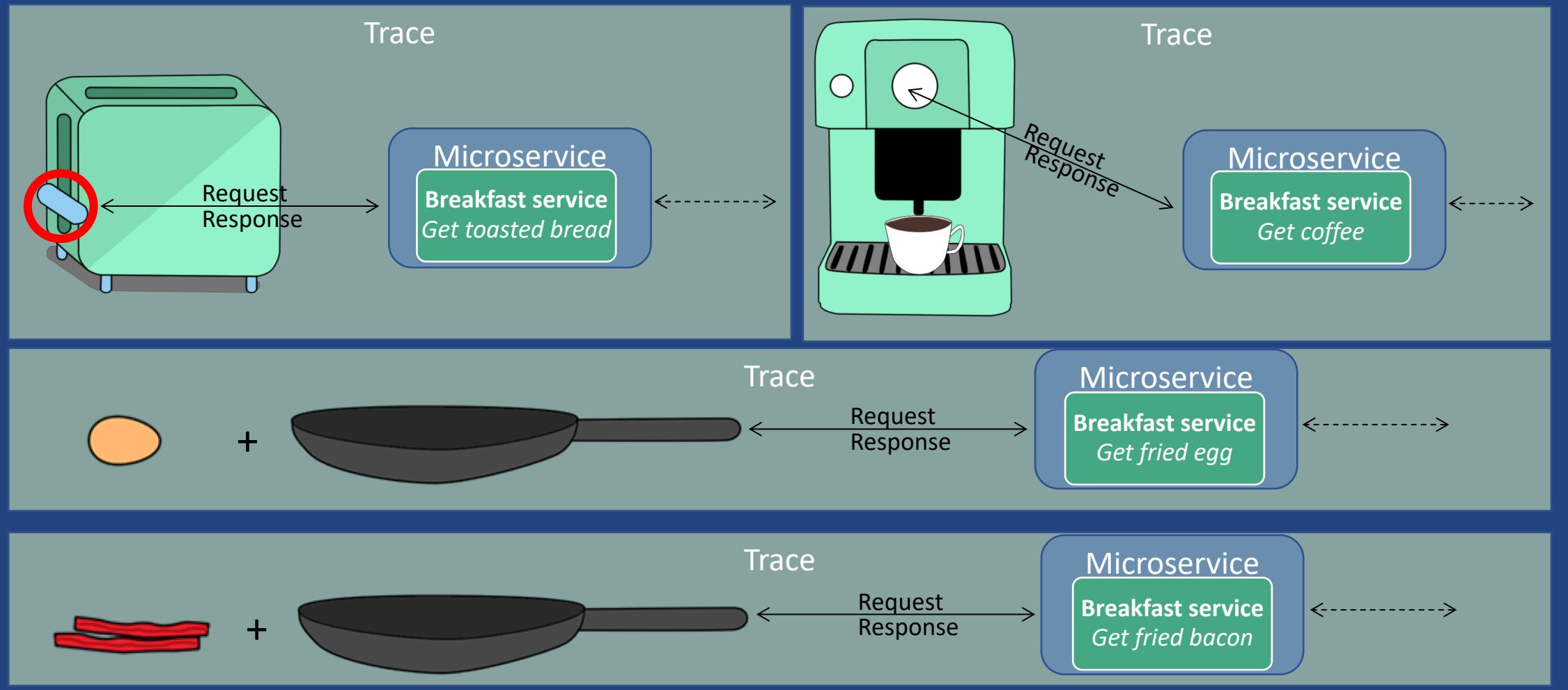
Then he gets a wonderful breakfast

Our application logs

▶ April 2nd 2018, 21:20:46.628	Breakfast Service	Getting a fried egg with some information	f8bf9b99012e95e1	f8bf9b99012e95e1
▶ April 2nd 2018, 21:20:46.555	Breakfast Service	Returning toasted bread	ab03e10dc971a506	ab03e10dc971a506
▶ April 2nd 2018, 21:20:41.549	PreferenceService	Getting information how the bread should be toasted	ab03e10dc971a506	c7d82cdf6d4fa034
▶ April 2nd 2018, 21:20:41.548	PreferenceService	FrameworkServlet 'dispatcherServlet': initialization completed in 20 ms	-	-
▶ April 2nd 2018, 21:20:41.528	PreferenceService	FrameworkServlet 'dispatcherServlet': initialization started	-	-
▶ April 2nd 2018, 21:20:41.526	PreferenceService	Initializing Spring FrameworkServlet 'dispatcherServlet'	-	-

Our future wishes

“Über-trace” aka conversation



Conversation ID specs

- Client may pass **request header** as **X-B3-ConVID**
- If not passed → application **creates** it
- **Subsequent** calls of services receive it & use it
- **Loggable** & passed to **Zipkin** → searchable
- Client **response header** has X-B3-ConVID

▼ Request Headers [view source](#)

Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: nl-NL,nl;q=0.9,en-US;q=0.8,en;q=0.7
Connection: keep-alive
Host: localhost:8081
Origin: https://localhost:8000
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_1)
x-b3-convid: d2131670-a4af-4a11-931f-ea53e4e5cc04

▼ Response Headers [view source](#)

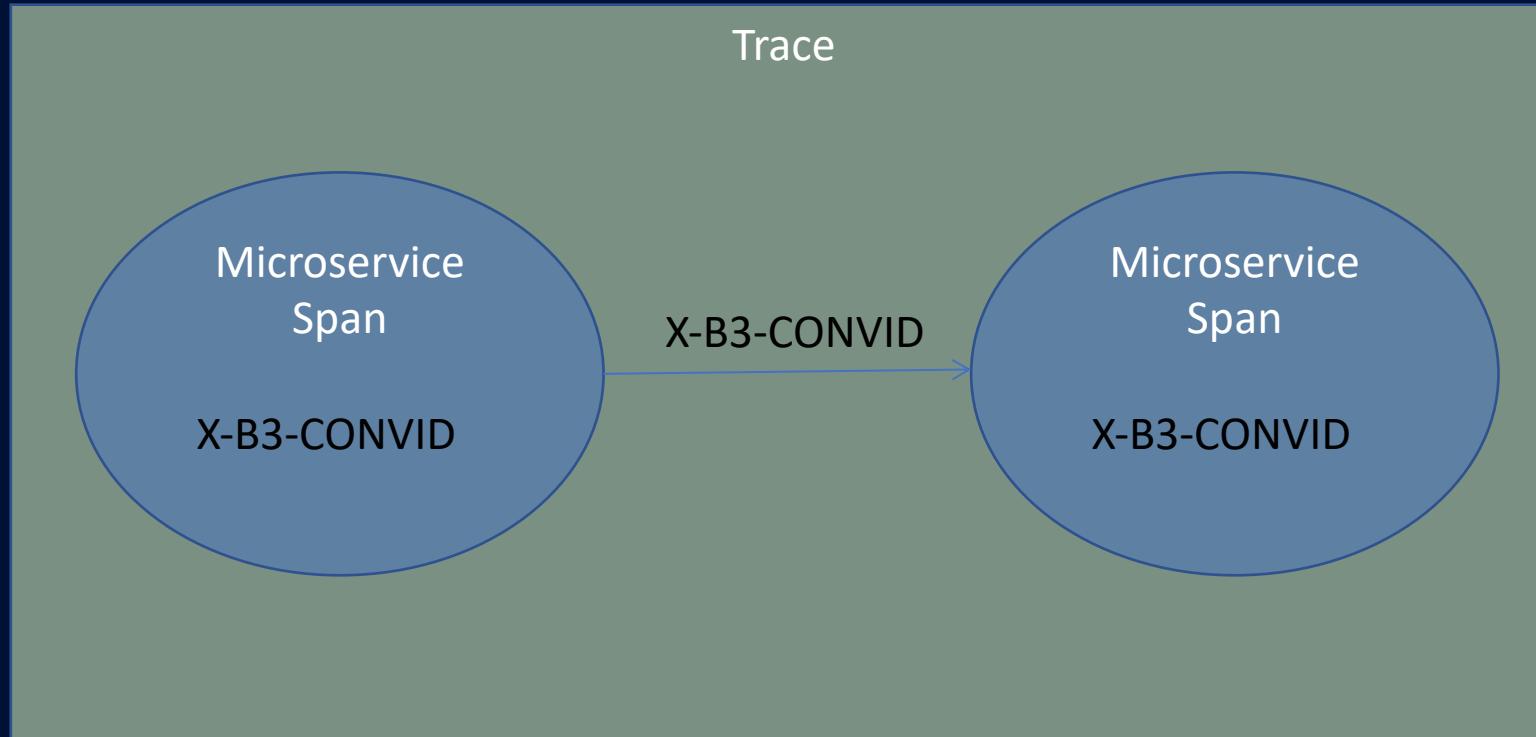
Access-Control-Allow-Origin: https://localhost:8000
Access-Control-Expose-Headers: X-B3-CONVID
Content-Type: application/json; charset=UTF-8
Date: Fri, 30 Mar 2018 20:36:16 GMT
Transfer-Encoding: chunked
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
X-B3-CONVID: d2131670-a4af-4a11-931f-ea53e4e5cc04

Adding a custom tracefilter

- Extend your class with **GenericFilterBean**
- Ensure that the `@Order` is *before* or *after*
`TraceWebServletAutoConfiguration.TRACING_FILTER_ORDER`,
dependent on what you want to provide

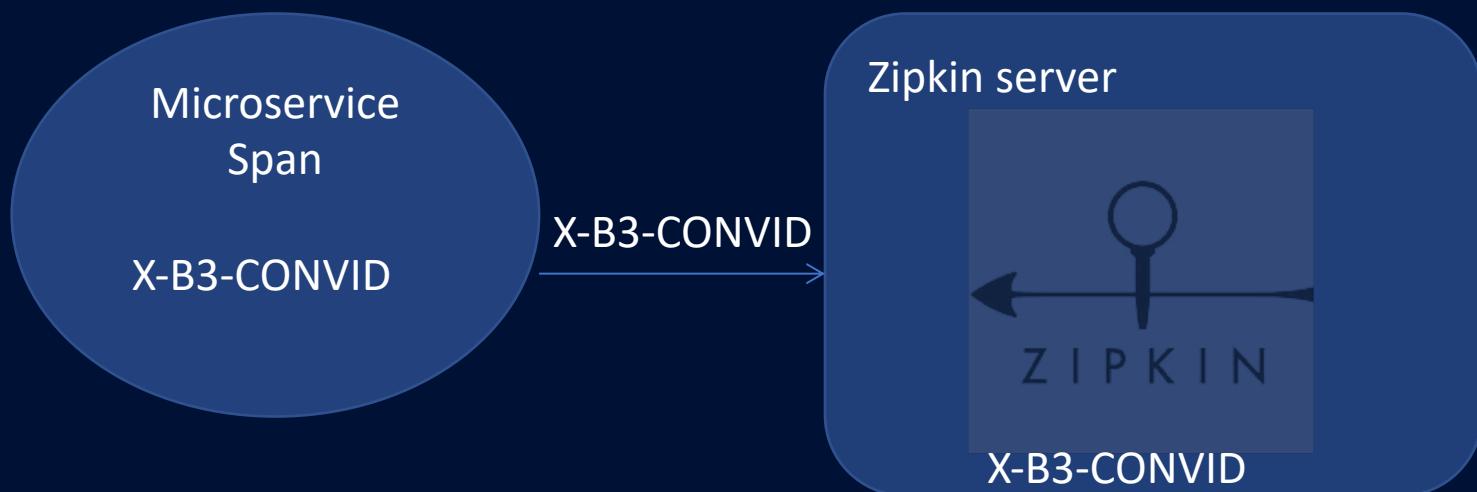
Propagation fields

- Or bagage
- Addable fields passed through services with trace and every span



Tags

- Addable fields for specific span for details / lookup
- Will be passed to Zipkin



Summary of demo

- Manipulate pass through of Sleuth with a custom **filter**
- **Propagation fields** will be passed to the whole trace
- A **tag** is used to tag a span & is also passed to Zipkin
- Make conversation id or trace id available in your reports!

But...

...what about applications without Spring Boot?

...what about applications without Java?

...are there alternatives to Zipkin?

Distributed tracing

opencensus.io & opentracing.io

The image shows two screenshots of developer websites side-by-side, with orange circles highlighting specific features on both pages.

OpenCensus (Left):

- Header:** OpenCensus
- Navigation:** Introduction, Quickstart, Tracing, Stats/Metrics, Tags, Exporters, zPages, Service, Advanced Concepts (highlighted), Guides, Language Support, Integrations, FAQ, Feature Matrix, Code of Conduct, Community, Blogs.
- Section: What is OpenCensus?**

OpenCensus is a single distribution of libraries that collect metrics and distributed traces from your services.
- Buttons:** OVERVIEW, QUICKSTART
- Section: How can I use OpenCensus in my project?**

We provide libraries for Go, Java, C#, Node.js, C++, Ruby, Erlang/Elixir, Python, Scala and PHP.
- Text:** Supported backends include Azure Monitor, Datadog, Instana, Jaeger, SignalFx, Stackdriver, and Zipkin. You can also [add support for other backends](#).
- Buttons:** LANGUAGE SUPPORT, SUPPORTED BACKENDS
- Section: Who is behind it?**

OpenCensus originates from Google, where a set of libraries called Census are used to automatically capture traces and metrics from services. Since going open source, the project is now composed of a group of cloud providers, application

OpenTracing (Right):

- Header:** OPENTRACING DOCS, GUIDES, PROJECT, GET INVOLVED, GITHUB, BLOG, REGISTRY, SAY HI ON GITTER
- Section: Vendor-neutral APIs and instrumentation for distributed tracing**
- Text:** Libraries available in 9 languages: Go, JavaScript, Java, Python, Ruby, PHP, Objective-C, C++, C#.
- Text:** The latest from the community: Merging OpenTracing and OpenCensus
- Section: Supported Tracers:** ZIPKIN, LIGHTSTEP, JAEGER
- Section: Supported Frameworks:** GRPC, Flask, Go kit, django
- Code Snippet:** bash

```
# Start Jaeger locally
$ docker run -d -p 6831:6831/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

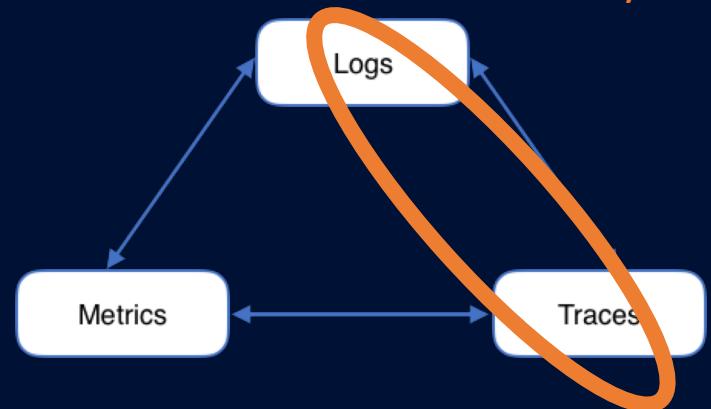
# Visualize the tracing instrumentation in Jaeger by
# clicking on 'Find Traces' in the UI.
$ open http://$DOCKER_IP:16686/
$ vim trivial.go

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

Summary

- Easy **review** of application calls with ELK, Sleuth, Zipkin
- More clear **separation** of events by traces & spans with Sleuth
- More easy search for **performance latencies** by visualization in Zipkin
- **Extra information** by extra custom interceptors/filters
- **Logging and tracing** will go hand-in-hand

Golden circle of observability



Thank you for your time!

- Questions?
- Please give feedback in the app

- Presentation & code at

<https://github.com/RDBreed/sleuthzipkindemo>



Observability & serverless?
14:10-14:50 by Yan Cui @ room 2

