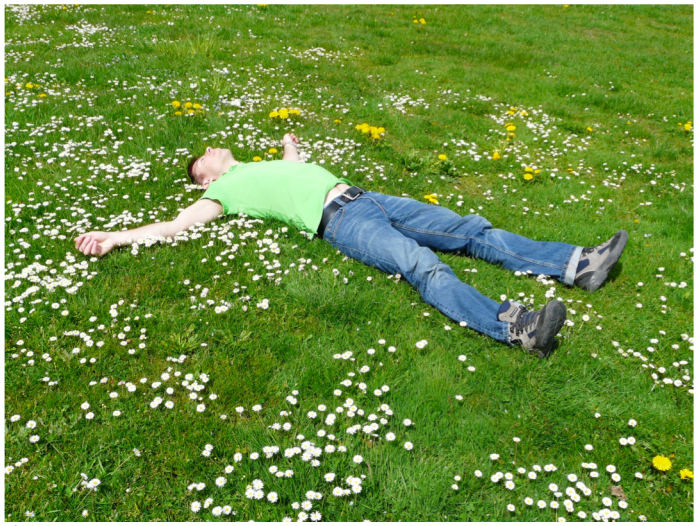
Tracing performance of your service calls with some help of Sleuth, Zipkin & ELK

by Rafaëla Breed

12th of April, 2018

devCON



Tracing performance of your service calls with some help of Sleuth, Zipkin & ELK

by Rafaëla Breed

12th of April, 2018

devCON

Rafaëla Breed

Software Engineer at Luminis /Amsterdam

Solve problems with creative solutions
Spring(Boot)
Photography
Traveling

#DevCon18

New company, new application

#jussayin

Let's just start developing our application

Everything works fine, but…

…there comes the tester/business/client:

"It does not work"

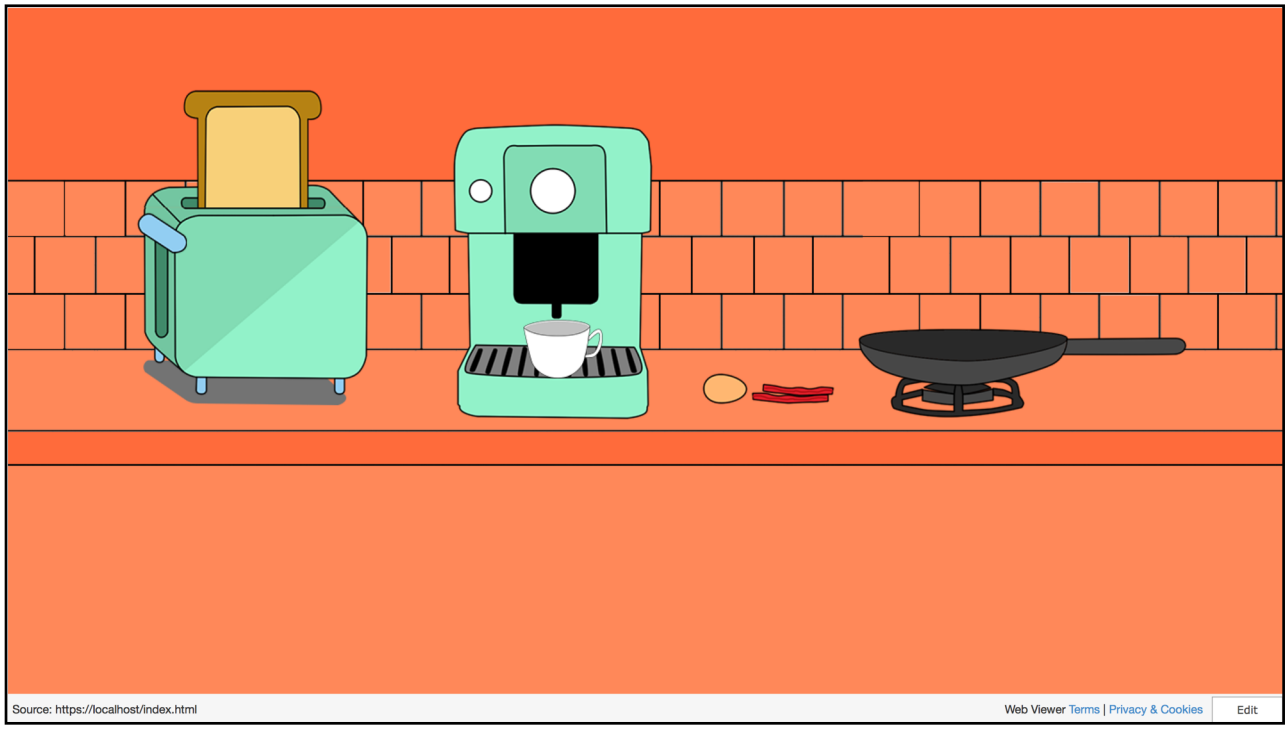
"So, what does not work?"

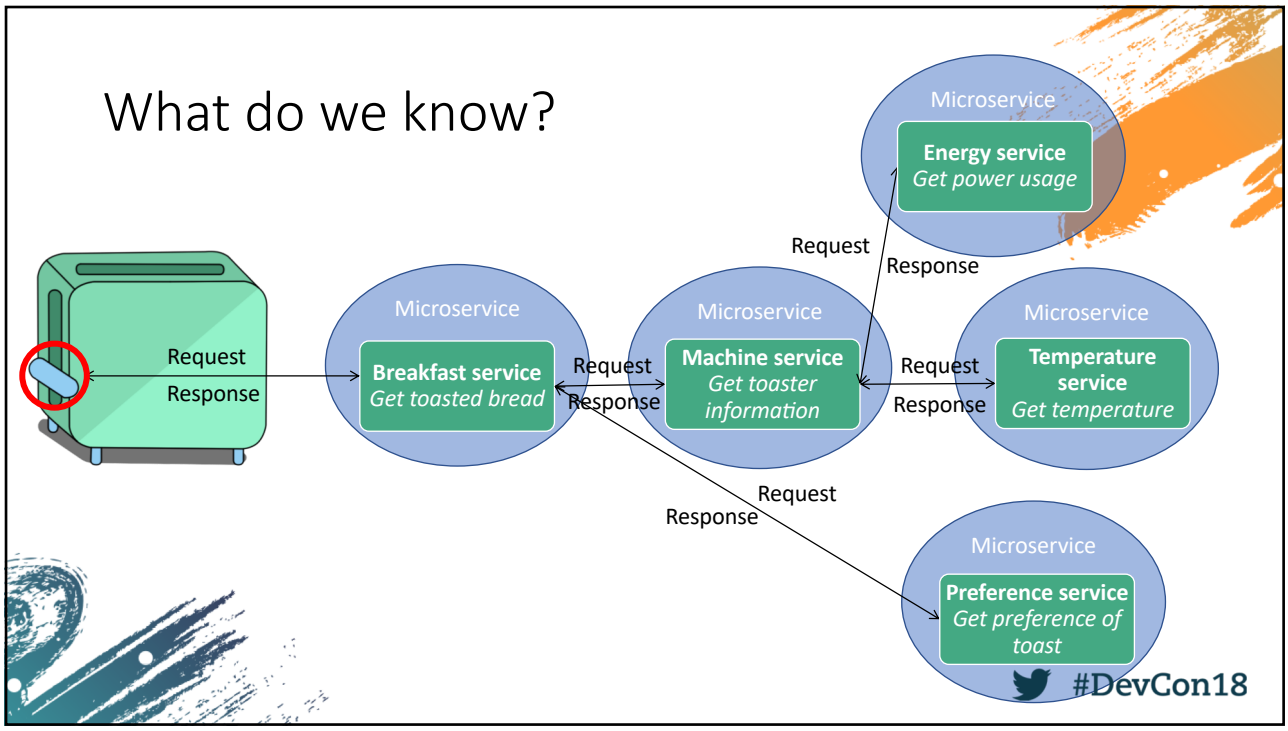
"Well, it is slow"

"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 ……"

#DevCon18


Source: https://localhost/index.html
Web Viewer Terms | Privacy & Cookies
Edit

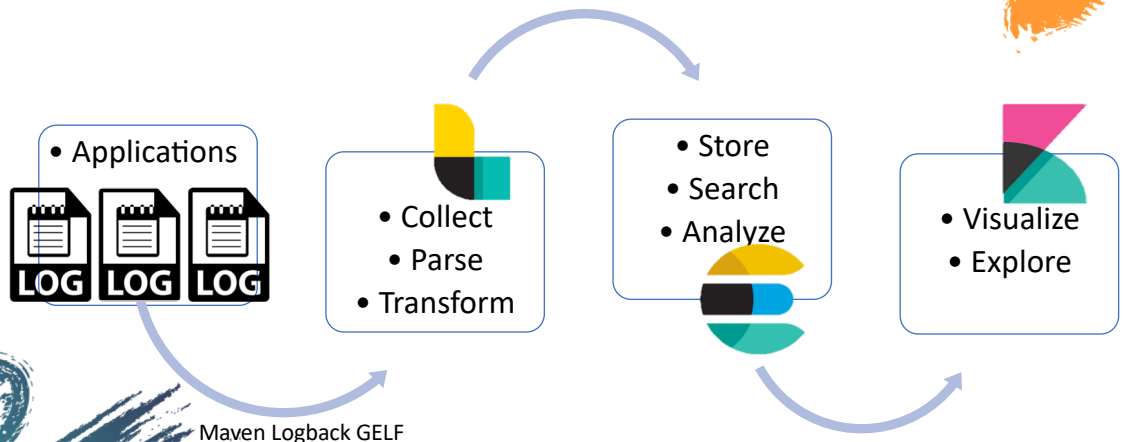

What do we know?
Microservice
Energy service
Get power usage
Request
Response
Request
Response
Microservice
Breakfast service
Get toasted bread
Request
Response
Microservice
Machine service
Get toaster information
Request
Response
Microservice
Temperature service
Get temperature
Request
Response
Microservice
Preference service
Get preference of toast
#DevCon18

What do we have?

#DevCon18

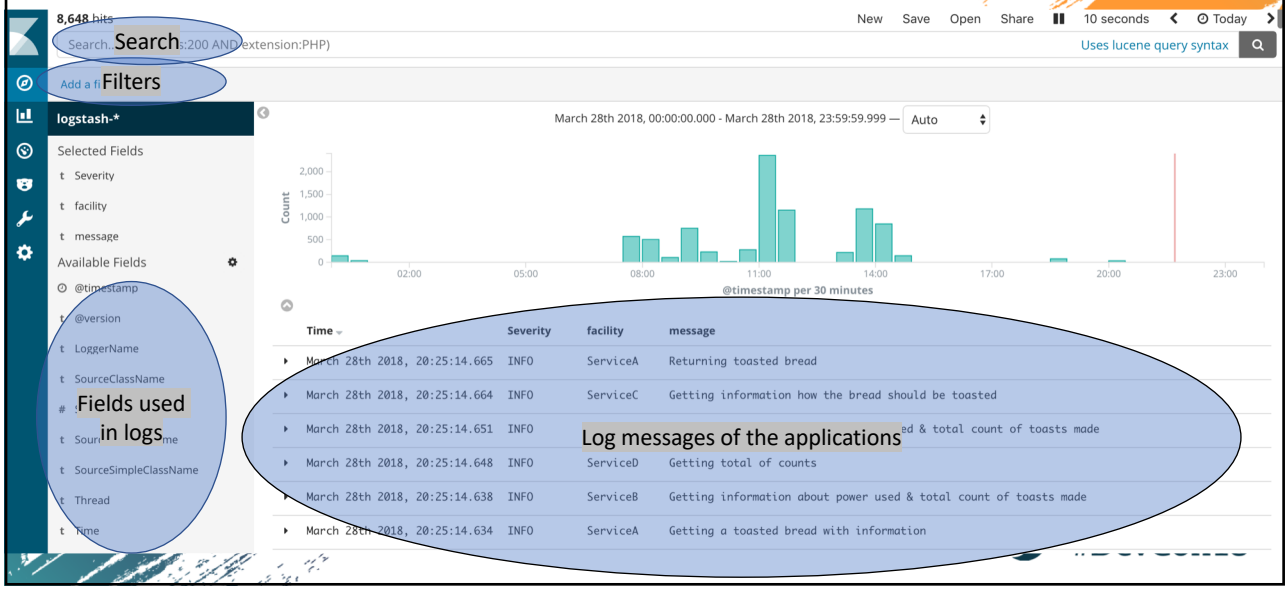
ELK: ElasticSearch, Logstash, Kibana

or Elastic Stack



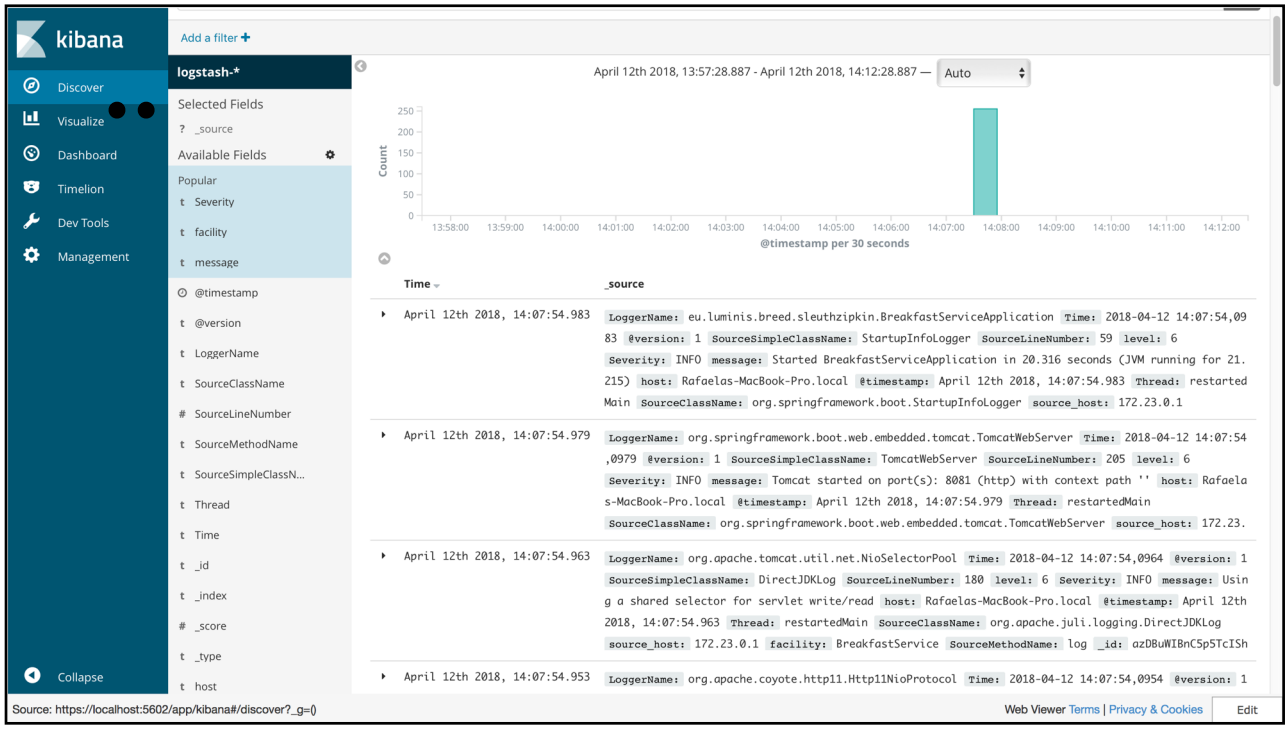
#DevCon18

Kibana



Let's try & dive in our logging

#DevCon18



Summary of demo

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

What we would like to have

- Each operation => unique id (**trace**)
- Each inner-event => unique id (**span**)

What Sleuth can offer

- Depicted from **Dapper** Infrastructure, **Zipkin**, **Htrace**
- Implementable via **Maven**/**Gradle**
- **Brave** is used as tracing library
- Every event starts a **trace**
- Every internal event starts a **span**
- Traces are passed to *next service(s)*

Adding Sleuth to your project

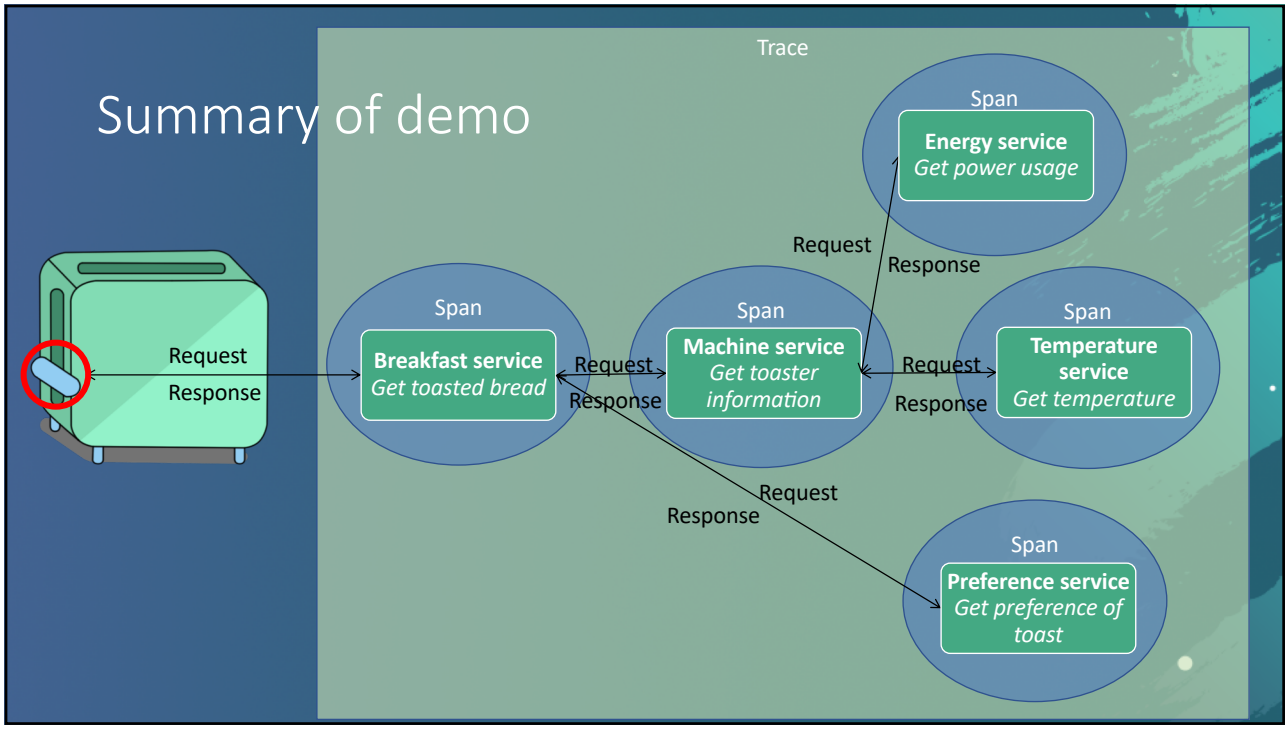
- To every SpringBoot app we add:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-sleuth</artifactId>
            <version>2.0.0.M9</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-TraceId and X-B3-SpanId to your configuration!

#DevCon18

Summary of demo

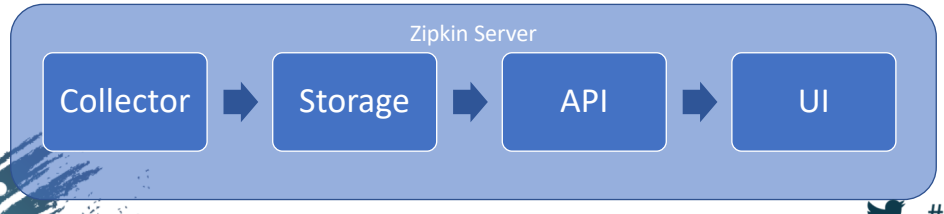


Summary of demo

	Time	facility	message	X-B3-TraceId	X-B3-SpanId
▸	March 28th 2018, 20:25:14.665	ServiceA	Returning toasted bread	430043c1f1cc5b70	430043c1f1cc5b70
▸	March 28th 2018, 20:25:14.664	ServiceC	Getting information how the bread should be toasted	430043c1f1cc5b70	1dc30c4b50c63e0a
▸	March 28th 2018, 20:25:14.651	ServiceB	Returning information about power used & total count of toasts made	430043c1f1cc5b70	4f5987109352211b
▸	March 28th 2018, 20:25:14.648	ServiceD	Getting total of counts	430043c1f1cc5b70	2e94422b354b28b5

Zipkin

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



#DevCon18

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>
```

#DevCon18

Sampler for Zipkin

- Up-front decision
- Default NEVER
- ALWAYS_SAMPLER
- Customize on annotation (in code)
- Customize on flag
- Probability 0…1 → chance 0 to 100%
- X-B3-Flags (client request header)

#DevCon18

Getting a Zipkin server

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

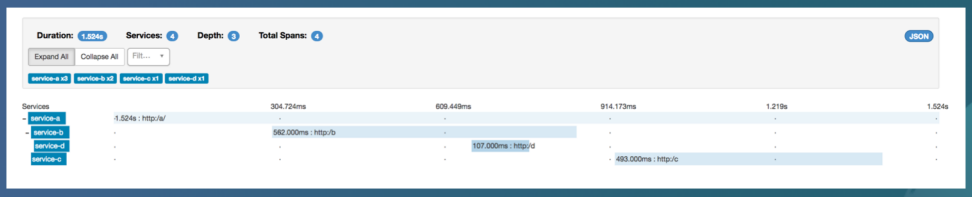
#DevCon18

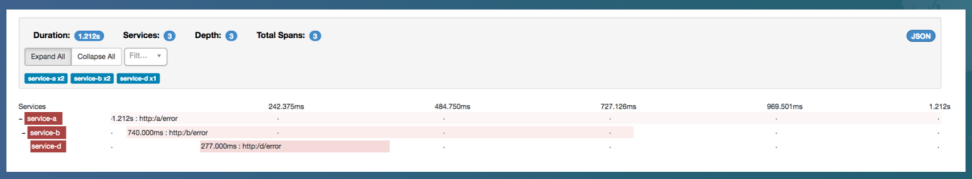
Summary of demo

- 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

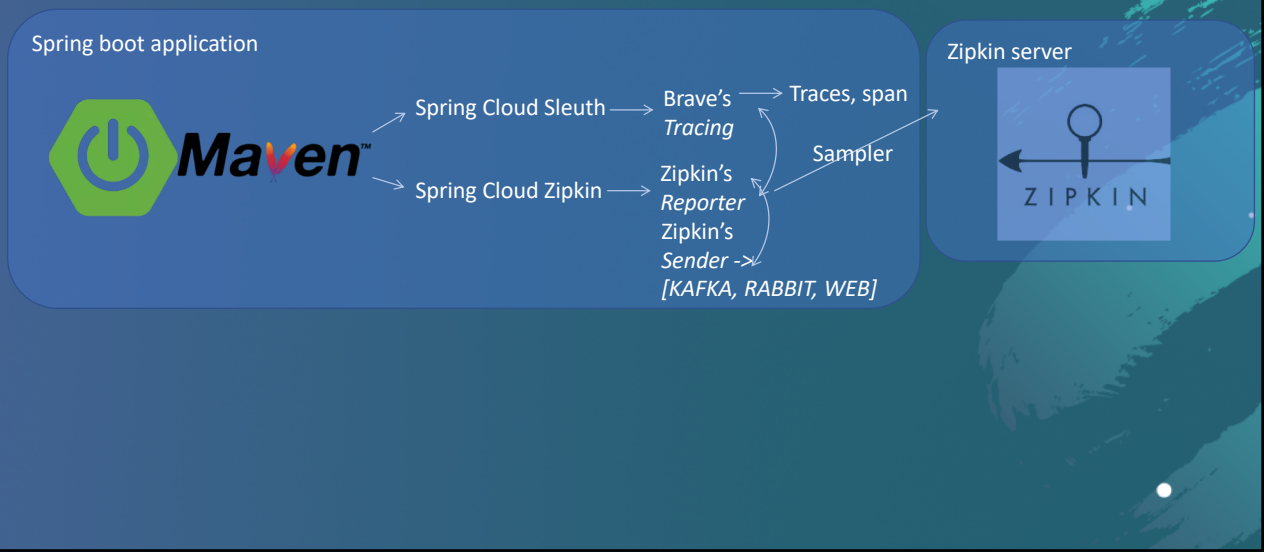
Summary of demo

- The user interface with clear **graphics** of events with times
- Visibility of errors (bonus!)





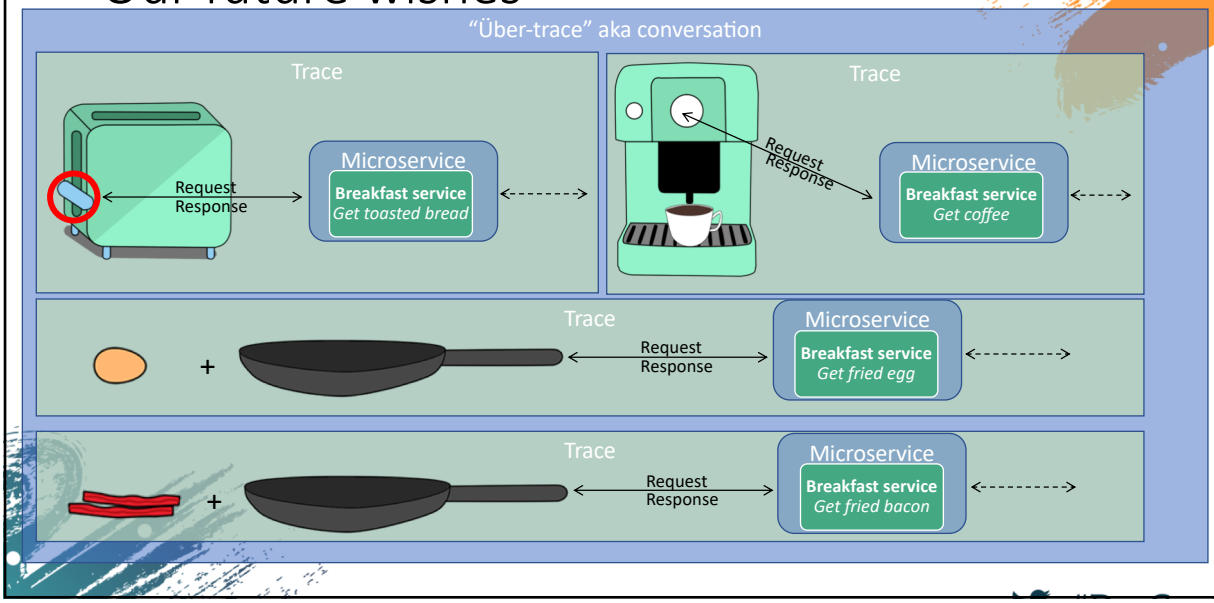
Overview what will happen (a bit under the hood)



#DevCon18

#DevCon18

Our future wishes



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

#DevCon18

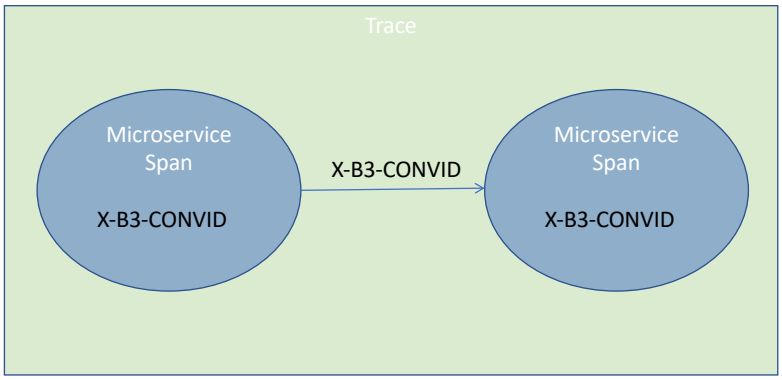
Adding a custom tracefilter

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

#DevCon18

Propagation fields

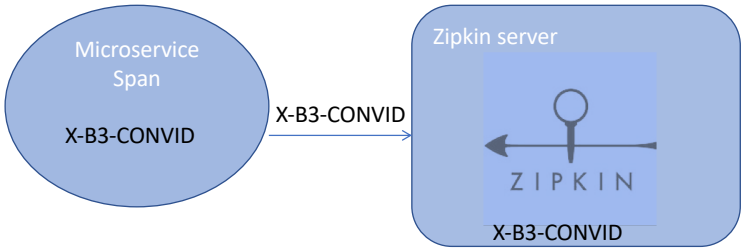
- Addable fields which will be passed through services



#DevCon18

Tags

- Addable fields on spans for details / lookup
- Will be passed to Zipkin



#DevCon18

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
- Other services? Databases, SOAP... → Use interceptors/filters or other implementations of Brave

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 fried egg
    And some bacon
    And a good cup of coffee
    Then he gets a wonderful breakfast
```

#DevCon18

Our test log

```
Feature: get a breakfast
  As a customer
  I want to get a breakfast
  So that I will start the day well
21:05:32.115 [main] DEBUG org.springframework.web.client.RestTemplate - Created GET request for "http://localhost:8081/breakfast/toastedbread"
21:05:32.120 [main] DEBUG org.springframework.web.client.RestTemplate - Setting request Accept header to [text/plain, application/json, application/*+json, */*]
21:05:32.681 [main] DEBUG org.springframework.web.client.RestTemplate - GET request for "http://localhost:8081/breakfast/toastedbread" resulted in 500 (null); invo


  Scenario:                            # breakfast-service.feature:6
    Given a customer                   # GeneralStepDefinitions.aCustomer()
    When a customer want a toasted bread # BreakfastServiceStepDefinition.aCustomerWantAToastedBread()
      java.lang.AssertionError: The response gave an statuscode 500 error with message:
```

#DevCon18

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

```
java.lang.AssertionError: The response gave an statuscode 500 error with message:
{"cause":null,"stackTrace":[{"methodName":"getError","fileName":"GlobalExceptionHandler.java","lineNumber":20,"className":"eu.luminis.breed.sleuthzipkin.configuration.GlobalExceptionHandler","nativ
        at org.junit.Assert.fail(Assert.java:88)
        at eu.luminis.breed.sleuthzipkin.BreakFastServiceApi$MyErrorHandler.handleError(BreakFastServiceApi.java:71)
        at org.springframework.web.client.ResponseErrorHandler.handleError(ResponseErrorHandler.java:63)
        at org.springframework.web.client.RestTemplate.handleResponse(RestTemplate.java:777)
        at org.springframework.web.client.RestTemplate.doExecute(RestTemplate.java:730)
        at org.springframework.web.client.RestTemplate.execute(RestTemplate.java:704)
        at org.springframework.web.client.RestTemplate.exchange(RestTemplate.java:662)
        at eu.luminis.breed.sleuthzipkin.BreakFastServiceApi.getResponse(BreakFastServiceApi.java:60)
        at eu.luminis.breed.sleuthzipkin.BreakFastServiceApi.getToastedBread(BreakFastServiceApi.java:43)
        at eu.luminis.breed.sleuthzipkin.BreakFastServiceStepDefinition.aCustomerWantAToastedBread(BreakfastServiceStepDefinition.java:30)
        at *.When a customer want a toasted bread(breakfast-service.feature:8)
```

And a fried egg
And some bacon
And a good cup of coffee
Then he gets a wonderful breakfast

#DevCon18

Our application logs

▸ April 2nd 2018, 21:20:46.628	Breakfast Service	Getting a fried egg with some information	f8bf9b99012e95e1	f8bf9b99012e95e1	908f7cf8-8318-455e-8ae7-0a760cf2783d
▸ April 2nd 2018, 21:20:46.555	Breakfast Service	Returning toasted bread	ab03e10dc971a506	ab03e10dc971a506	908f7cf8-8318-455e-8ae7-0a760cf2783d
▸ April 2nd 2018, 21:20:41.549	PreferenceService	Getting information how the bread should be toasted	ab03e10dc971a506	c7d82cdf6d4fa034	908f7cf8-8318-455e-8ae7-0a760cf2783d
▸ 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'	-	-	-

#DevCon18

Use Sleuth (& Zipkin)

- Make conversation id or trace id available in your reports!
- Eventually use your own via request header

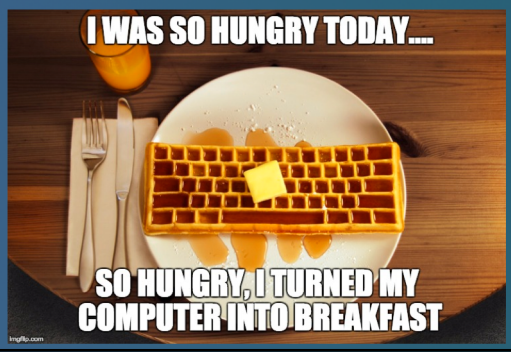
#DevCon18

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 **metrics** will go hand-in-hand

Thank you for your time!

- Questions?
- Presentation & code at https://github.com/RDBreed/sleuthzipkindemo



devCON
CINEMEC, EDE
APRIL 12TH, 2018

is powered by

luminis
Conversing worlds