

Monitoring, in the cloud





01. **Monitoring & Observability**

02. **App Monitoring**

03. **Custom Metrics**

04. **Grafana Datasource**

05. **Observability**

06. **Demo**



Monitoring & Observability



Monitoring & Observability



Monitoring

Understanding the present/past state of the system.

Predefined
metrics/logs



Observability

“Debug” the system in real-time.

Undefined
properties/patterns

Why should SWEs do the monitoring?

🤔 Isn't it a job for DevOps Engineers?

Trend Analysis

Trends in MAU, cache key growth rate, and system growth/decline trends

Change Analysis

Understanding the impact of recent deployments and infrastructure changes

Impromptu Debugging

Identify key metric trends for system failure/anomaly events

Alerting

Alert the on-call engineer in the event of system failure.
Proactive measures can be taken before detection at the infrastructure level.

3 Monitoring Targets

Business Metrics

- Unique visitors
- File upload size
- Auth success/fails
- Page views

App Metrics

- DB connections
- JVM Heap memory
- Thread count
- Response time by endpoint

Platform Metrics

- RDB queries/sec
- EBS volume usage
- Redis memory usage
- EKS scheduler status

App Monitoring



Application Monitoring Tech Stack



Application

- ✓ Spring Boot Actuator



Data Collection/Storage

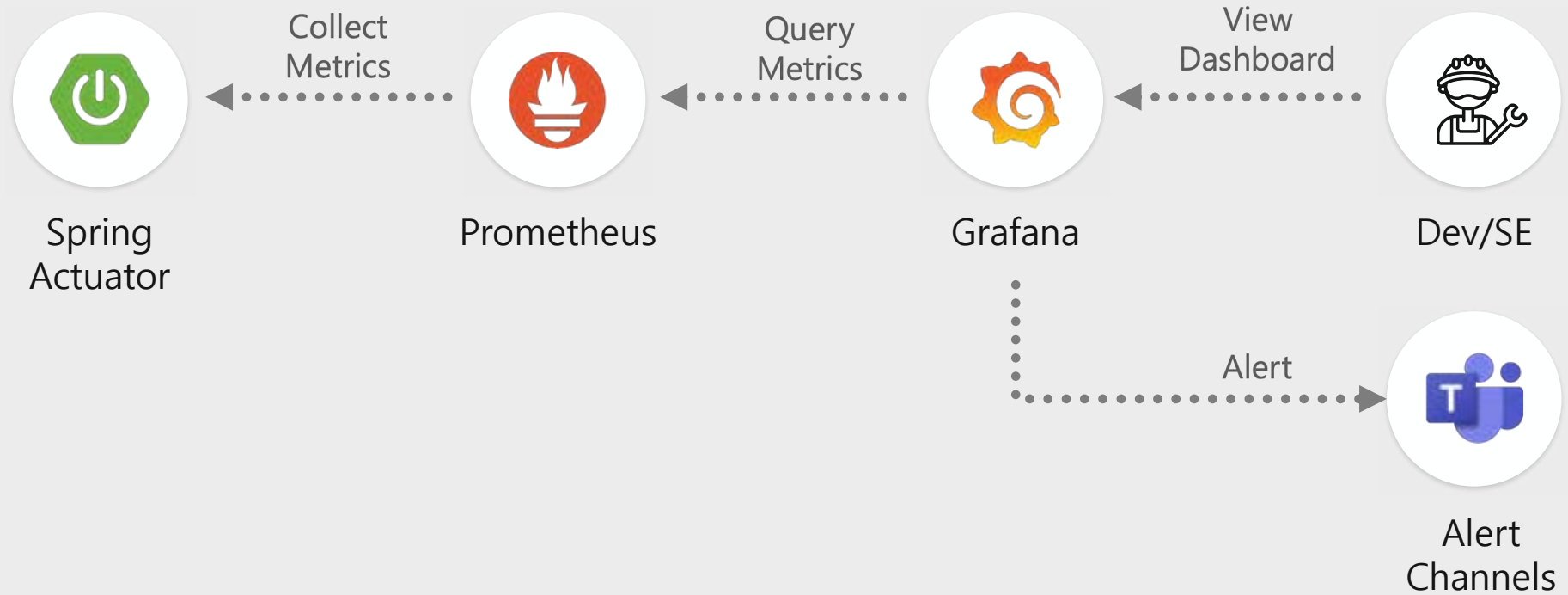
- ✓ Prometheus
- ✓ Elasticsearch
- ✓ RDB/Redis
- ✓ AWS CloudWatch
- ✓ Sentry



Data Visualization

- ✓ Grafana
- ✓ Sentry

Application Monitoring Tech Stack



What is Spring Actuator?

App Monitoring & Control

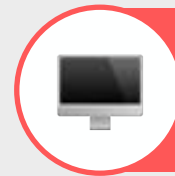
Query internal metrics, Shutdown, heap dump creation, etc

- Simply add a dependency to an existing Spring project
- Access metrics and features through HTTP Endpoints
- Supports variety of metric formats through Micrometer



JVM

Memory Usage, Thread count, GC metrics



System

CPU Usage, File descriptor, Uptime



Logger

Event count of each log level



Web MVC/WebFlux

Call count of each endpoint, Response time



Datasource

Hikari DB pool size, Connection speed

Spring Actuator

Prometheus Endpoint

<http://localhost:8080/actuator/prometheus>

```
# TYPE jvm_memory_committed_bytes gauge
jvm_memory_committed_bytes{area="heap",id="Tenured Gen",} 6.2713856E7
jvm_memory_committed_bytes{area="nonheap",id="CodeHeap 'profiled nmethods'",} 2.7262976E7
jvm_memory_committed_bytes{area="heap",id="Eden Space",} 2.523136E7
jvm_memory_committed_bytes{area="nonheap",id="Metaspace",} 1.09748224E8
jvm_memory_committed_bytes{area="nonheap",id="CodeHeap 'non-nmethods'",} 2555904.0
jvm_memory_committed_bytes{area="heap",id="Survivor Space",} 3080192.0
```

What is Prometheus?



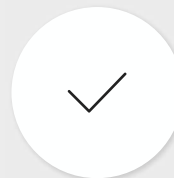
Time Series DB

Handling all events with INSERT for easy system change monitoring. Optimizing time-series data with long-term data retrieval and automatic data expiration.



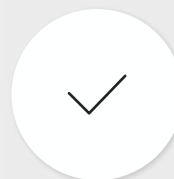
Label-based Data Model

e.g. Group by HTTP Method + URI or just by URI



All-in-one Solution

Metric collection, storage, querying, and alerting can all be addressed using Prometheus



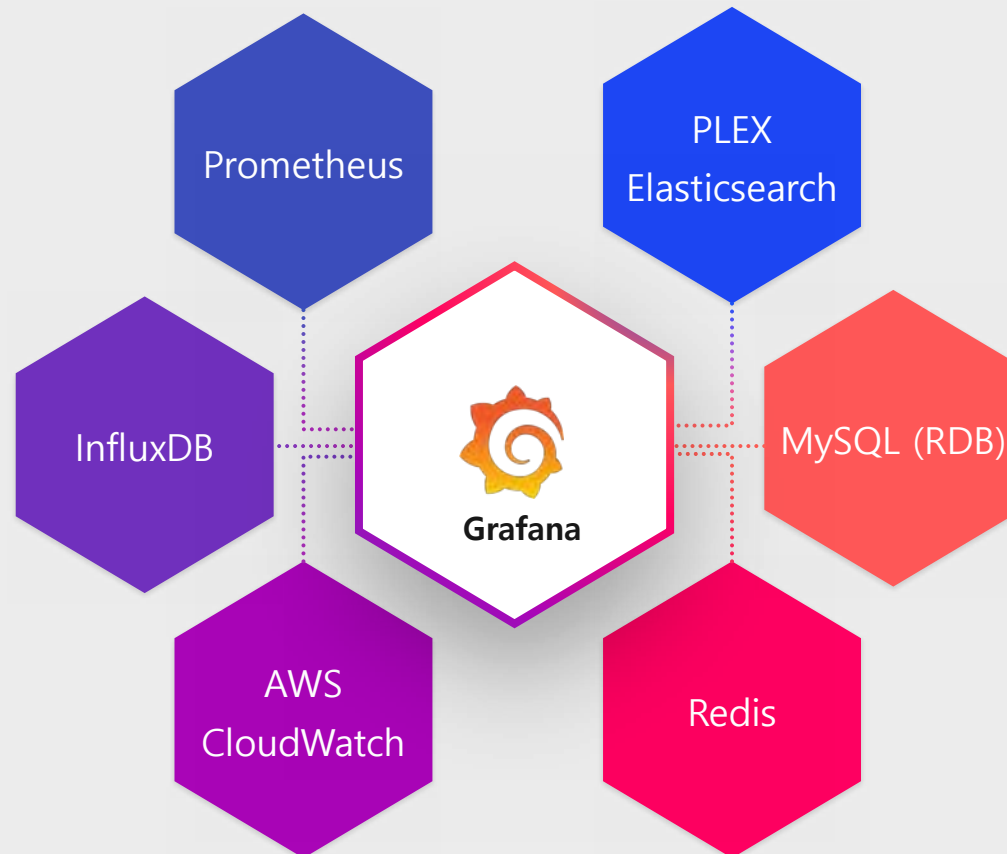
Works well with Kubernetes

Most Kubernetes modules provide metrics in the Prometheus format

What is Grafana?

Data Visualization

- Able to connect diverse sets of data sources
- Create Monitoring Dashboards
- Supports alerts based on dashboards





- ✓ Data origin: Spring Actuator
- ✓ Data Collection/Storage: Prometheus
- ✓ Data Visualization: Grafana



Custom Metrics



Custom Metrics

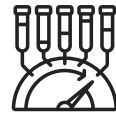
Meters



Counter

A single metric that can only increase

e.g. cumulative page views



Gauge

Metric that can go up or down

e.g. number of active sessions



Timer

Duration or count

e.g. average time taken to call an external API

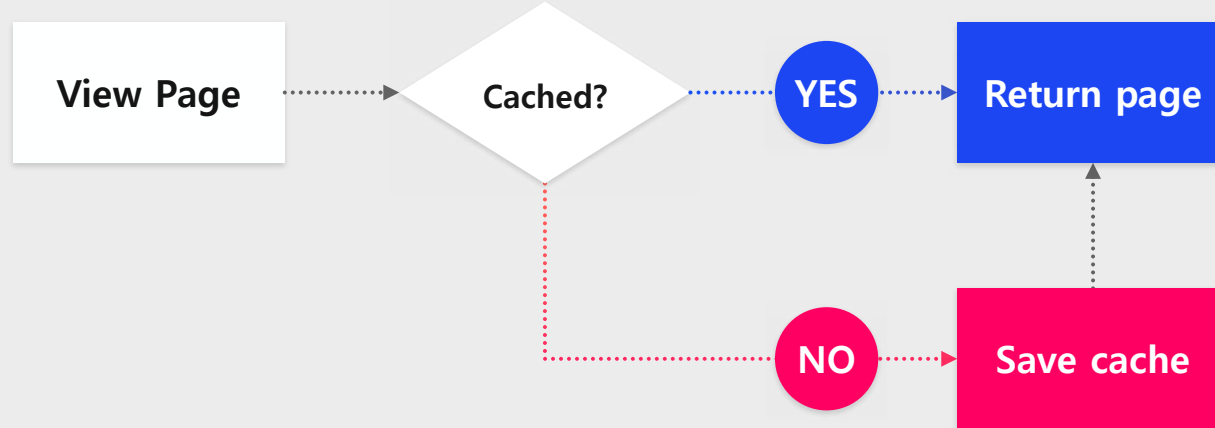
Time Taken to Call an External API

```
@Timed(  
    value = "realprob.gettime",  
    description = "Time taken to get realprob pro  
public List<RealProbDto> getProbtableRealprobs(  
    String serviceid, String contentid, String tr  
    return webClient  
        .get() WebClient.RequestHeadersUriSpec<...>  
        .uri(  
            uri: "/services/{serviceId}/contents/{con  
            serviceid,  
            contentid,  
            trialid, | 김유홍, 2021-07-13 09:26 • Op  
            probtableid) capture of?  
        .exchangeToMono(  
            response -> {
```

@Timed annotation

- Record execution count and total time taken by using it on a method
- Avg. Execution Time = (Total Time Taken) / (Execution Count)
- Needs Spring AOP dependency
- Only usable on public methods

Limitations of @Timed



- Only 1 timer per method
- No dynamic labeling
- Gauge, Counter not supported

```
public ReleasedPage getReleasedPage(String slug, String pageUuid) {  
    if (  
  
        Va  
        Re  
        if  
  
        if (!serviceSnapshotRedisRepository.pageCacheTableExists(slug, isPreview: false)) {  
            snapshotService.cacheSnapshot(service.getServiceId());  
            pageDbReadCounterBuilder.tags(SLUG_TAG, slug, PAGE_UUID_TAG, pageUuid).register(meterRegistry).increment();  
            page = optPage.get();  
            pageCacheReadCounterBuilder.tags(SLUG_TAG, slug, PAGE_UUID_TAG, pageUuid).register(meterRegistry).increment();  
        }  
    }  
}
```

Increment RDB read
counter

Increment Cache read
counter

= Grafana Datasources



Grafana Datasources

AWS CloudWatch



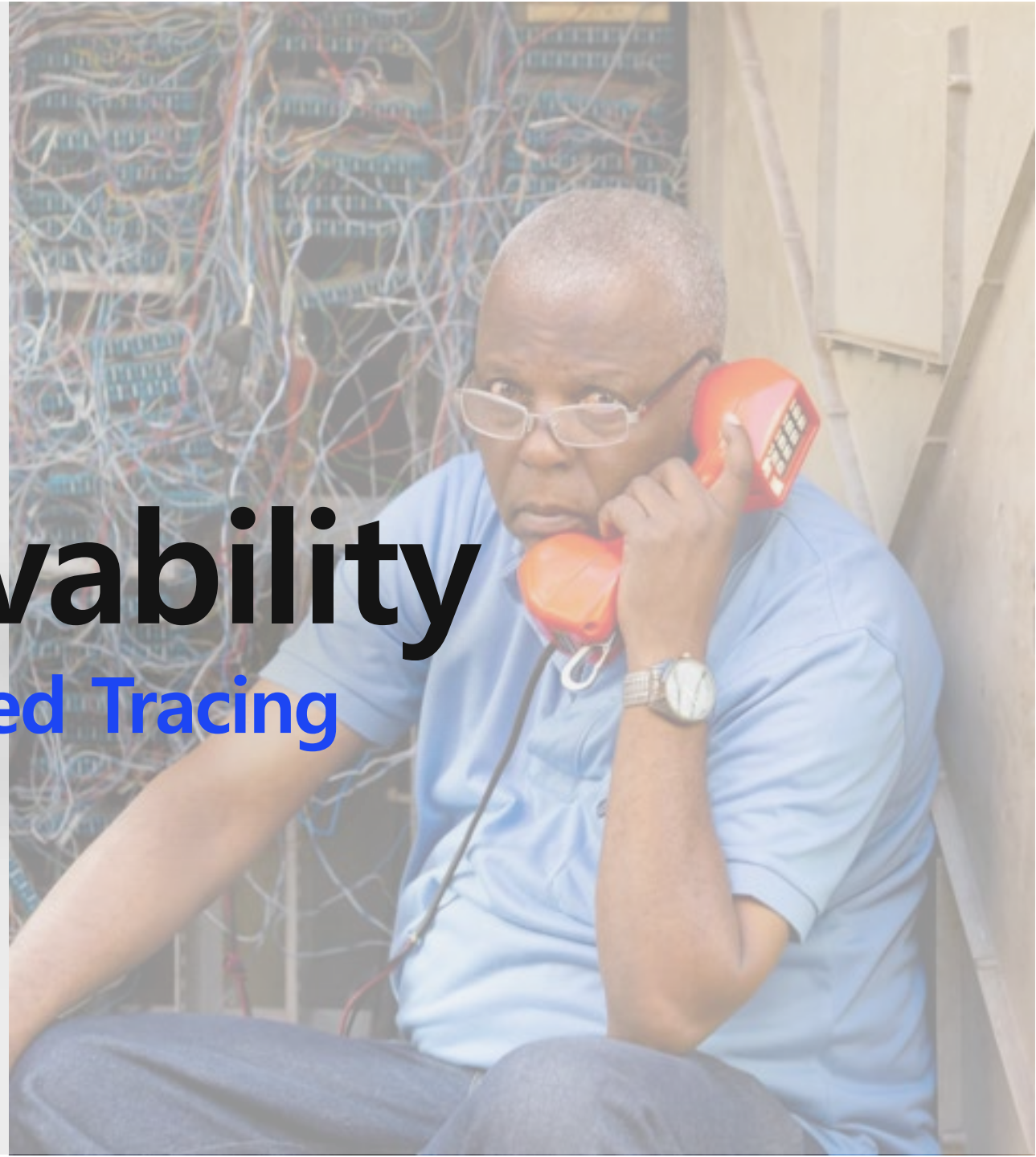
Grafana Datasources

MySQL (RDB)



Observability

Distributed Tracing

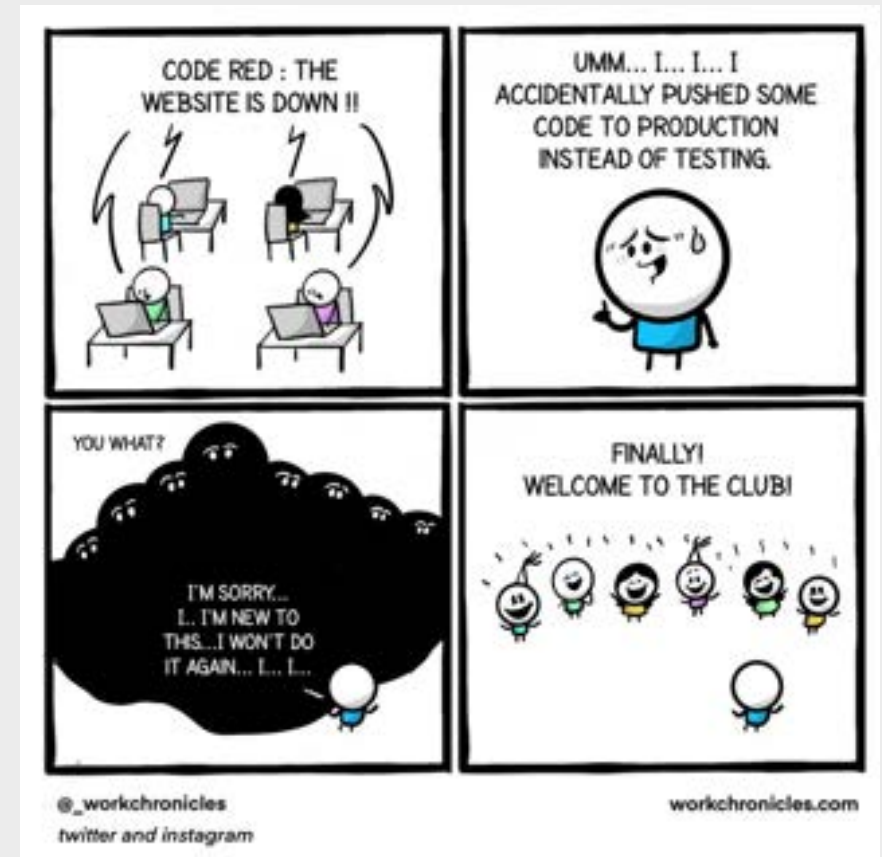


4 Quadrants of Risk

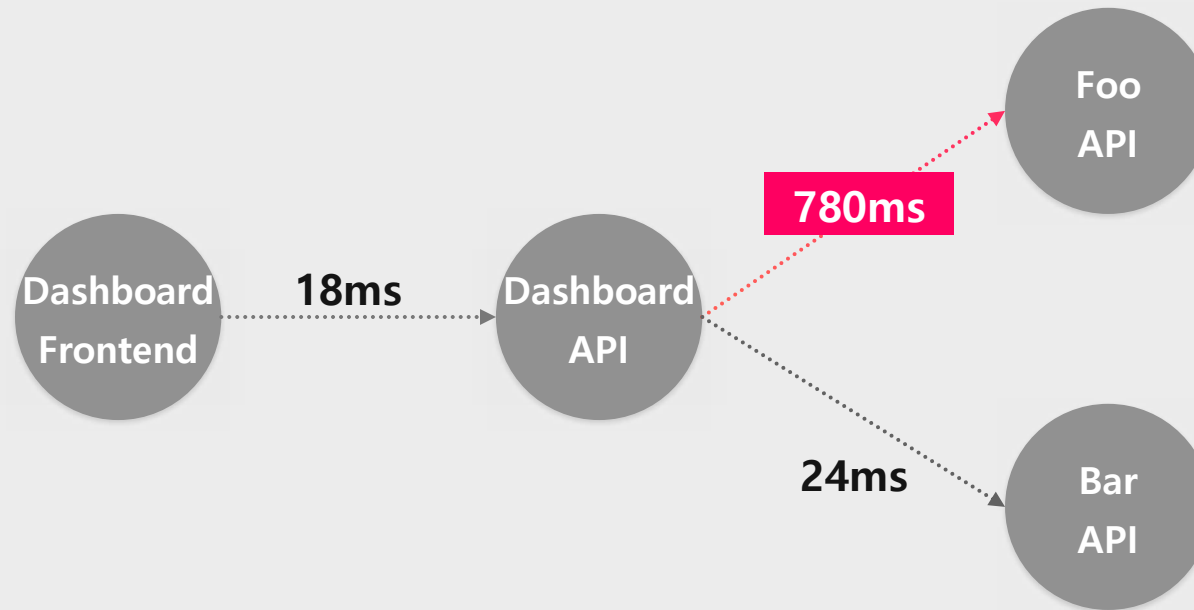
		CAUSE identified?	
		Yes	No
PROBLEM identified?	Yes	Internal Server Error Unchecked input is the cause	Memory Leak Cause Unknown
	No	AWS Failure leads to service failure Don't know which AWS service failed	Don't know what errors will happen, Don't know what will cause errors

How Can We Prevent the Unpredictable?

- You Can't.
Incidents will occur
NO MATTER WHAT
- But, we can prepare:
 - 🩹 Rapid **Incident Response** & Recovery
 - 💀 **Postmortem** & Root Cause Analysis
 - 🚧 Mitigate **Failure Recurrence**

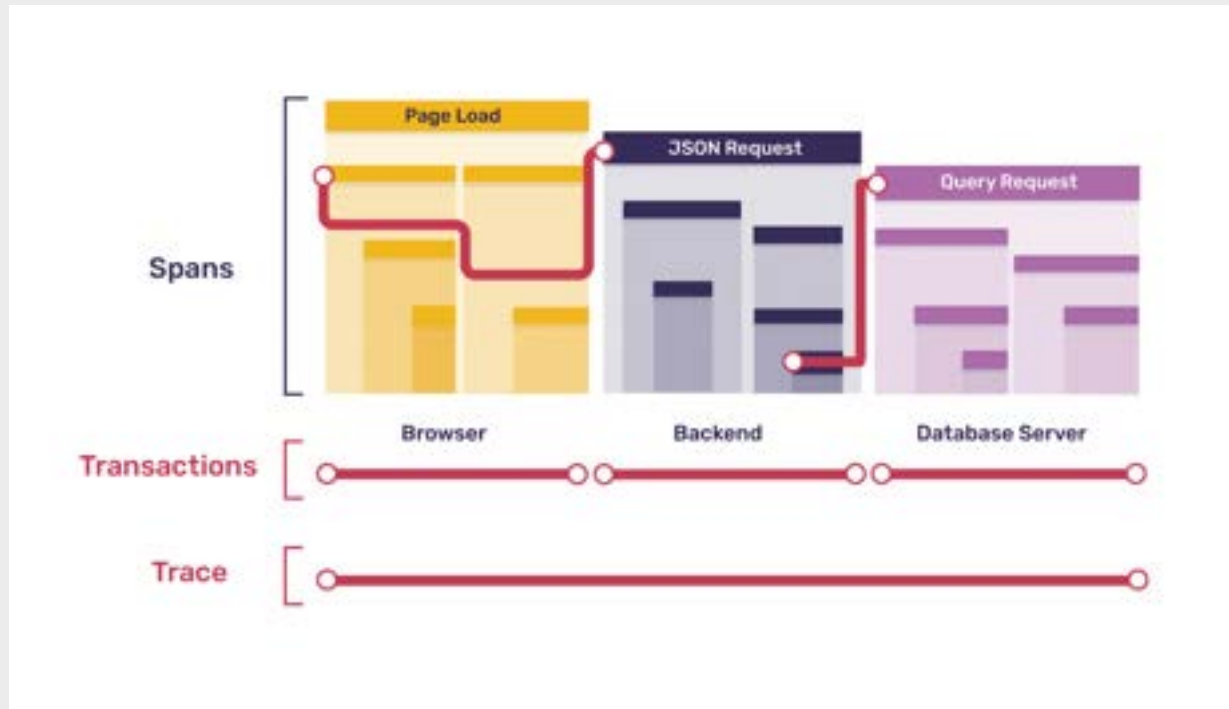


Distributed Tracing



- Hard to identify delays and errors in a distributed microservices topology
- Even harder to identify short-lived errors only occurring under specific conditions
- **No more ambiguous problem reports**
 - ✗ Dashboard is slow today
 - ✗ Page view takes too long
 - ✓ EgRepository#findByld method used by page view operation shows an average of 3,000ms response time and therefore caused page view delays for 2 hours

Distributed Tracing



- Multiple spans under a single Transaction
- Usually 1 Method : 1 Span
- SDKs for almost all mainstream frameworks
Vue, React, Spring, .NET, Express, etc

First Contentful Paint ⓘ

1696ms



0% 100% 0%

Largest Contentful Paint ⓘ

1500ms



100% 0% 0%

First Input Delay ⓘ

6ms



100% 0% 0%

Cumulative Layout Shift ⓘ

Frontend Tracing

LCP p75 ⓘ



LCP Distribution ⓘ



Total Events 21

Display 1

LCP p75 ▾

Display 2

LCP Distribution ▾

★ TRANSACTION

PROJECT

TPM ↓

FCP

LCP

FID

CLS

USERS

USER MISERY

☆ /	0.0003	n/a	n/a	n/a	n/a	1	
☆ /mg/services/43001401/uploader/...	0	1.32s	1.33s	150ms	n/a	1	
☆ /mg/services/12345/folders	0	1.70s	n/a	n/a	n/a	2	
☆ /mg/services/12345/site-config	0	1.61s	1.12s	100ms	n/a	2	
☆ /mg/services/12345	0	1.46s	n/a	n/a	n/a	2	
☆ /mg/services/12345/...	0	2.11s	n/a	20.60ms	n/a	1	

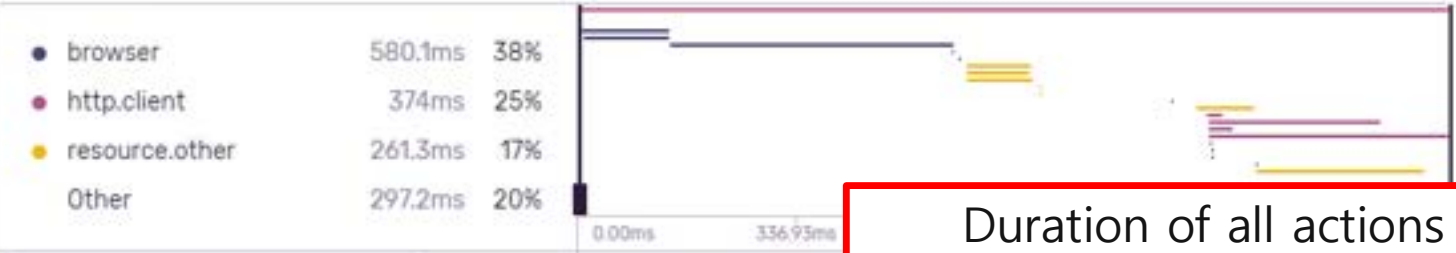
WebVitals

Event ID ? b126853a
Event Duration ? 1.35s
Status ? unknown
10 days ago

Trace Navigator ?
This Event 4 Children 2 Descendants
View Full Trace: cc'

Related Transactions across different microservices

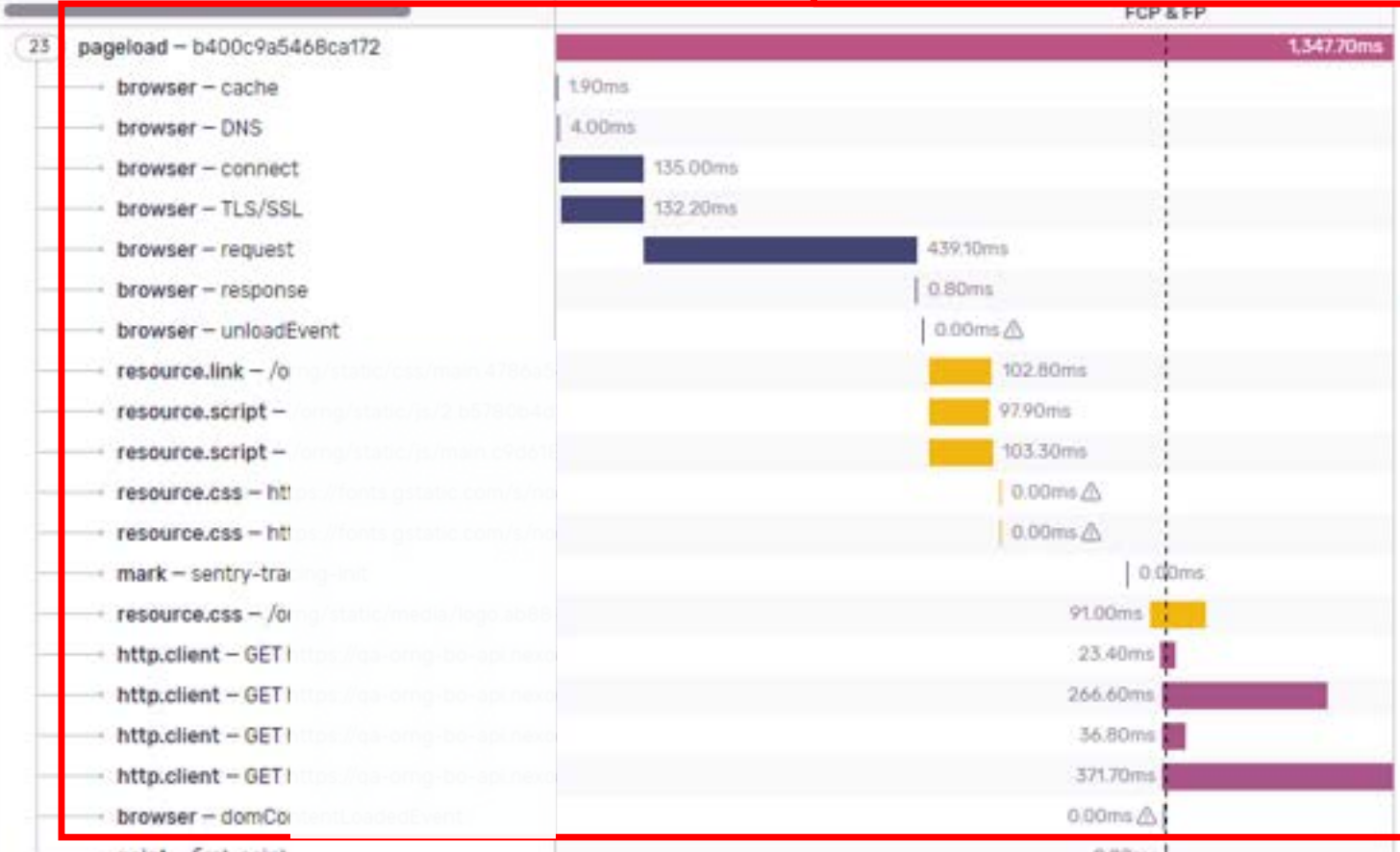
Filter Search for spans



Duration of all actions

Web Vitals

- First Contentful Paint: 980.700ms
- First Paint: 980.700ms
- Time to First Byte: 579.300ms
- Request Time: 438.300ms



Tag Details

- browser: Chrome 92.0.4515
- browser.name: Chrome
- deviceMemory: 8
- effectiveConnectionSpeed: 4g
- environment: chrome
- hardwareConcurrency: 8
- level: page
- os: windows-10
- os.name: Windows
- transaction: 1

Frontend Trace Detailed View

Event ID ⓘ Event Duration ⓘ Status ⓘ
 b9 799.00ms ok
🟢 10 days ago -

Trace Navigator ⓘ
 Parent — This Event — 1 Child
 View Full Trace: 3bd

API Trace Detailed View

Filter

Pro	law	602ms	73%
Emj	ndBySid	113ms	14%
http.client			

DB call, HTTP API call
Time Taken

Tag Details

- browser Chrome 92.0.4515
- browser.name Chrome
- client_os Windows 10
- client_os.name Windows
- environment dev
- level info
- server_name
- transaction
- url



user id:N

User Info

BREADCRUMBS

TYPE	CATEGORY	DESCRIPTION	LEVEL	TIME
>	com.nexon.org...	S	un	info 16:19:19
>	org.springframework...	n	to	info 16:19:20
>	org.springframework...	C	t	info 16:19:20
↔	http	G	su	info 16:24:23
>		[-t	info 16:24:23
>		e	on	Id
>		t	id	info 16:24:23

Application Log

WebClientResponseException\$BadRequest GET /api/realprob...

400 Bad Request from GET https://priva... ExceptionHandler

Resolve Ignore Mark Reviewed Share Open in Discover

Event 077c9 | JSON (30.0 KiB) Older Newer

Tags: Chrome Version: 92.0.4515, Windows Version: 10

transaction GET /api/

API Error Detailed View

MESSAGE: WebClient Exception

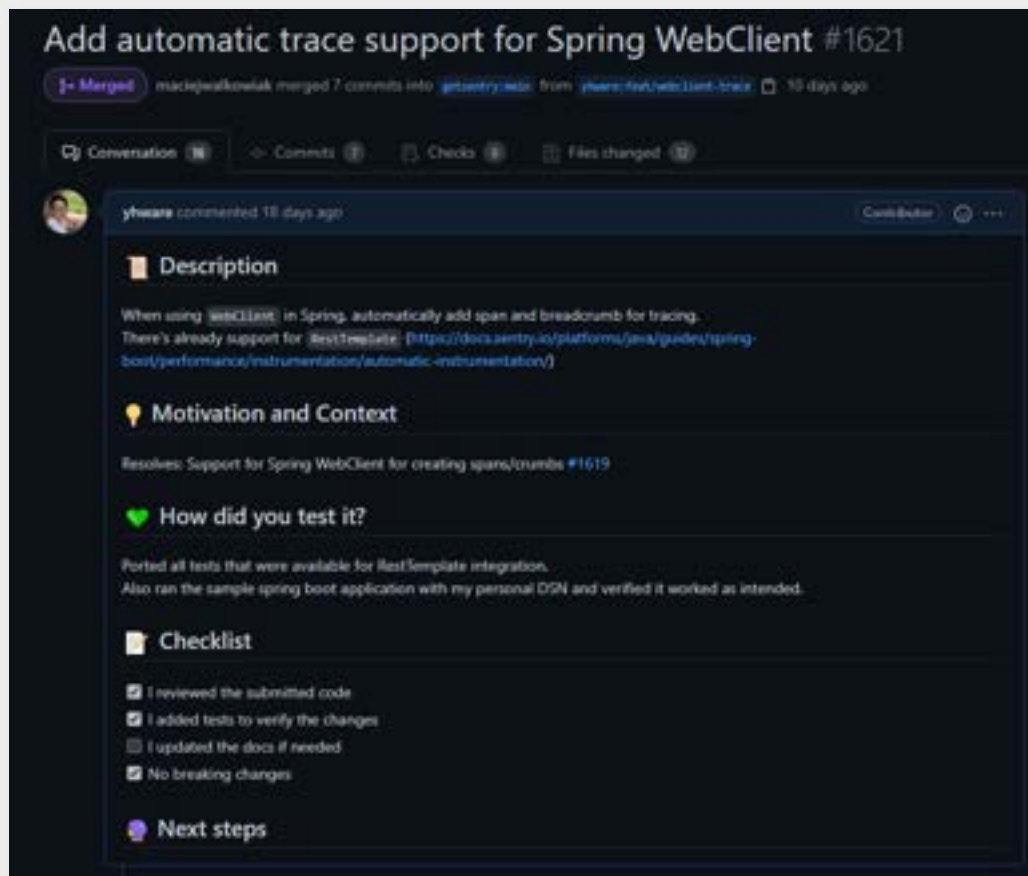
EXCEPTION (most recent call first): WebClientResponseException\$BadRequest

400 Bad Request from GET https://s/4/fill

```
org.springframework.web.reactive.function.client.WebClientResponseException in create at line 196
org.springframework.web.reactive.function.client.DefaultClientResponse in lambda$createException$1 at line 213
reactor.core.publisher.FluxMap$MapSubscriber in onNext at line 106
reactor.core.publisher.FluxOnErrorResume$ResumeSubscriber in onNext at line 79
```

Stack Trace

A little show off: My OSS contribution 🎉🙈



Add automatic trace support for Spring WebClient #1621

maciejwalkowiak merged 7 commits into getsentry/sentry from yhwang/feat/webClient-trace 10 days ago

Conversation · Comments · Checks · Files changed

yhwang commented 10 days ago

Description

When using `WebClient` in Spring, automatically add span and breadcrumb for tracing. There's already support for `RestTemplate` (<https://docs.sentry.io/platforms/java/guides/spring-boot/performance/instrumentation/automatic-instrumentation/>)

Motivation and Context

Resolves: Support for Spring WebClient for creating spans/trunks #1619

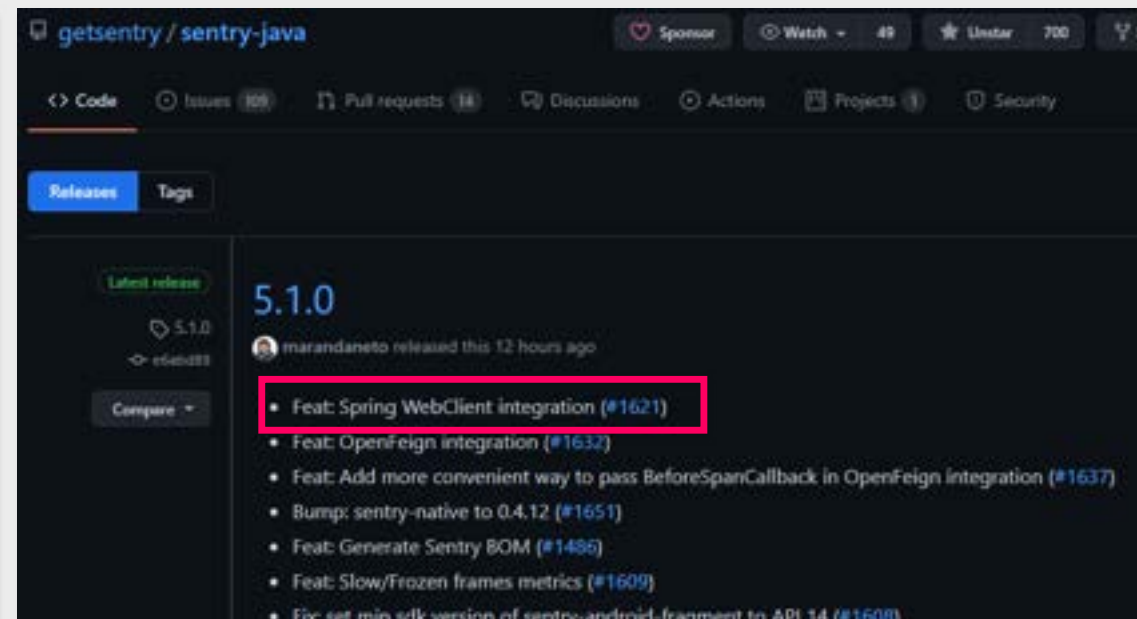
How did you test it?

Ported all tests that were available for `RestTemplate` integration. Also ran the sample spring boot application with my personal OGN and verified it worked as intended.

Checklist

- I reviewed the submitted code
- I added tests to verify the changes
- I updated the docs if needed
- No breaking changes

Next steps



getsentry / sentry-java

Sponsor · Watch 49 · Unstar 700

Code · Issues 109 · Pull requests 14 · Discussions · Actions · Projects 1 · Security

Releases · Tags

Latest release

5.1.0

marandaneo released this 12 hours ago

- Feat: Spring WebClient integration (#1621)
- Feat: OpenFeign integration (#1632)
- Feat: Add more convenient way to pass BeforeSpanCallback in OpenFeign integration (#1637)
- Bump: sentry-native to 0.4.12 (#1651)
- Feat: Generate Sentry BOM (#1486)
- Feat: Slow/Frozen frames metrics (#1609)
- Fix: set min sdk version of sentry-android-fragment to API 14 (#1608)

Thank you.

Daniel Kim