

# Own your software by using Micrometer and Spring boot



Ko Turk

**You're part of a racing team**



But what if we don't have this kind of information?

Sounds silly right?



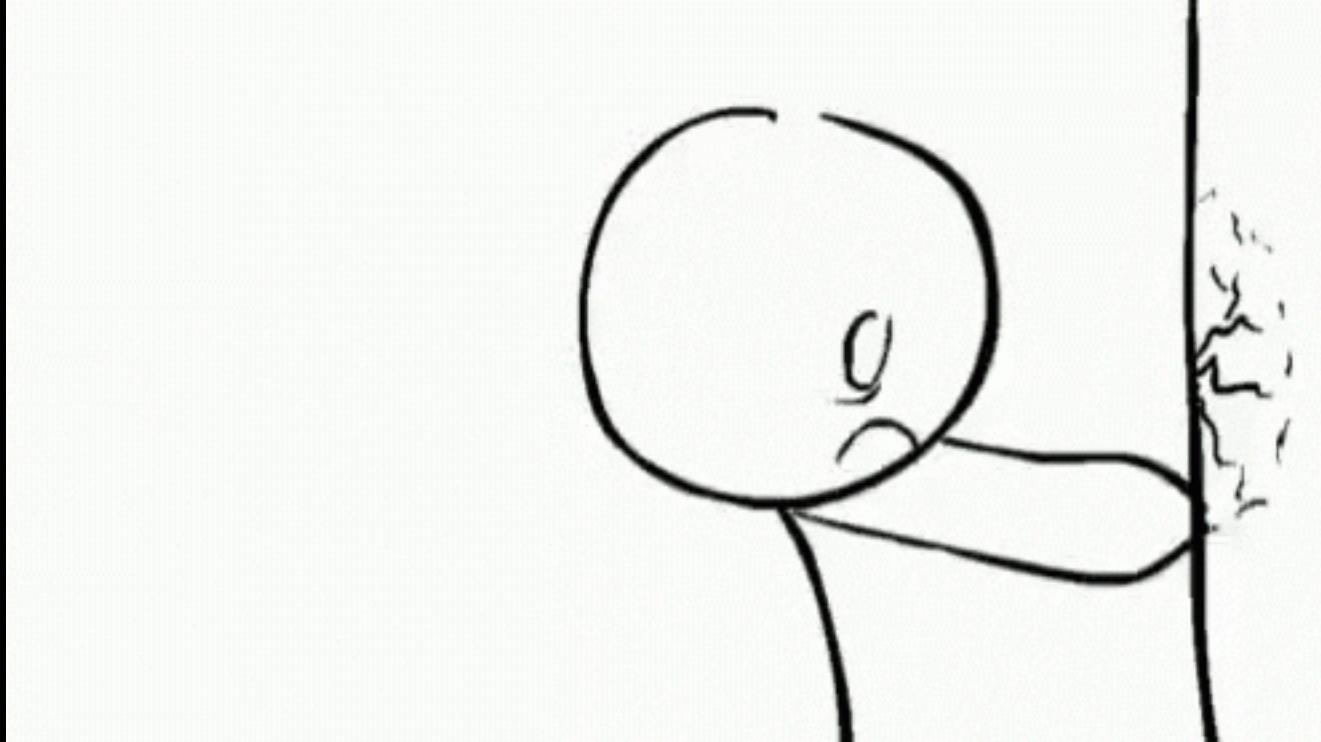
But why don't we (developers) think about proper monitoring?

(still driving but crashing)



Application Y

Backend



If that's happening.....

```
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>log4j-over-slf4j</artifactId>
    <scope>provided</scope>
</dependency>
```

# Of course you can log it but.....

```
} catch (Exception exception) {
    log.error("WTF is going wrong here:", exception);
    return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
}
```

```
log.debug("You never going to read this, logging level is INFO", e);
```

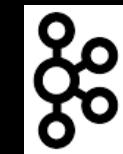
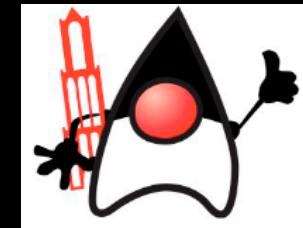
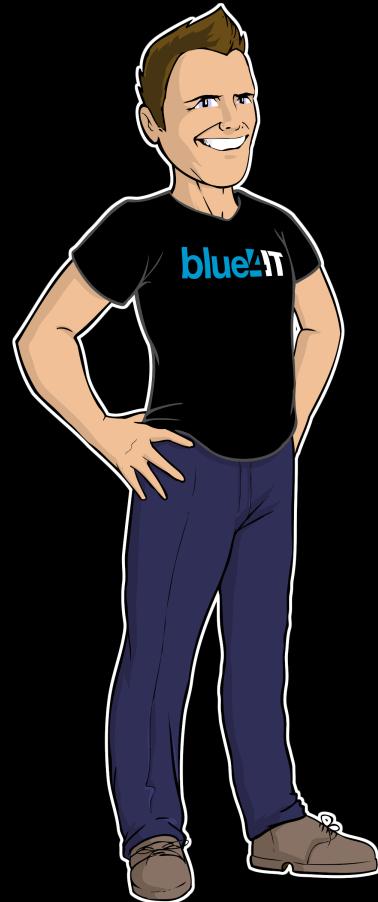
**A log is an event that happened**

**A metric is a measurement of the health of a system.**

That's where  
Micrometer comes in

# Introduction

# MARRIED



@KOTURK77

# Logging? Metrics?



# What do you learn today (goals)?

Define

Create

Save

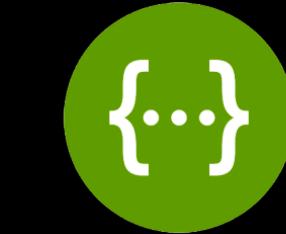
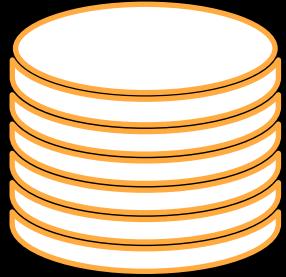
And visualise

# metrics!

# *Let's define the teams strategy*



# **What do we want to monitor?**



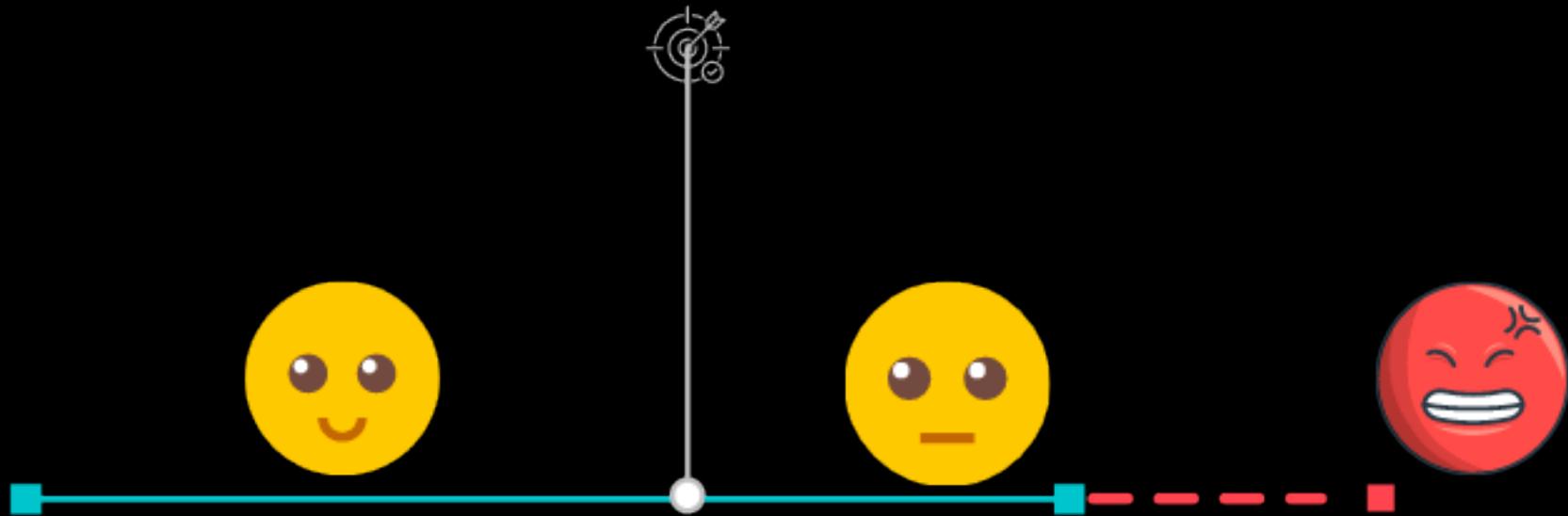
# How to be good in DevOps?



<https://landing.google.com/sre/sre-book/toc/index.html>

## Pitstop

Shouldn't take more than 4 seconds



## Pitstop

It should be less than 6 seconds

# What should you monitor?

# What do you learn today (goals)?

Define

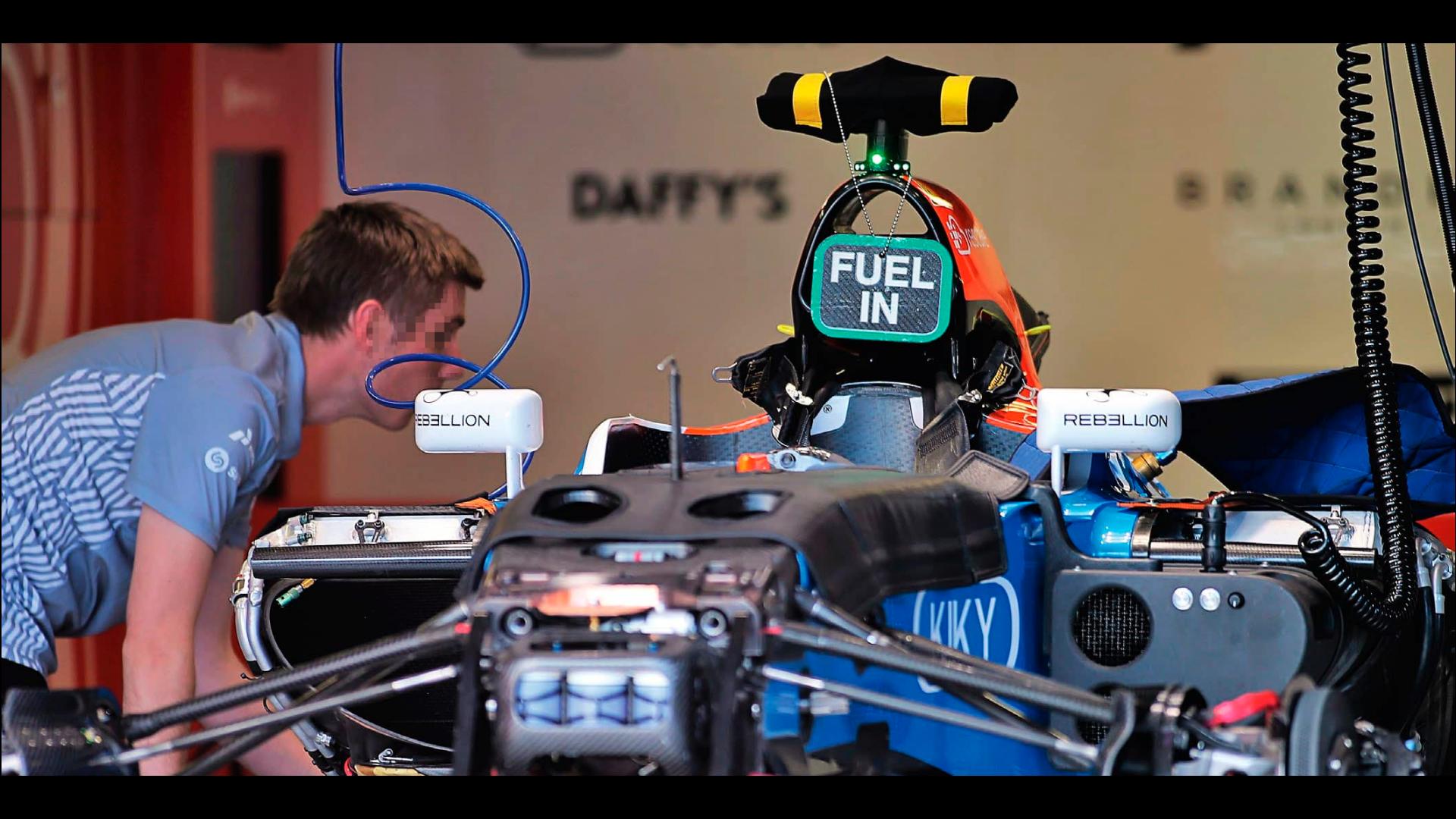


Create

Save

And visualise

# metrics!



FUEL  
IN

REBELLION

REBELLION

KIKY

Micrometer for the win!

But what is it?

# **SLF4J only for metrics**

# Who is using Spring Boot?

# Micrometer is included in Spring Boot actuator

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

# Extra dependency for Spring Boot 1.5.x

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-spring-legacy</artifactId>
</dependency>
```

## And also for Micronaut

```
<dependency>
    <groupId>io.micronaut.configuration</groupId>
    <artifactId>micronaut-micrometer-core</artifactId>
</dependency>
```

```
<dependency>
    <groupId>io.micronaut</groupId>
    <artifactId>micronaut-management</artifactId>
</dependency>
```

## And for Quarkus

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-micrometer</artifactId>
</dependency>
```

# Live coding



<https://i.ytimg.com/vi/eVpRNi5VEDo/maxresdefault.jpg>

# Pre-configured Bindings

- Instrumentations of:
  - caches
  - class loader
  - garbage collection
  - processor utilisation
  - thread pools
  - and more

[Go to actuator metrics](#)

# Timer

[https://s1.cdn.autoevolution.com/images/news/gallery/red-bull-racing-beats-record-to-set-new-fastest-pit-stop-in-the-history-of-f1\\_2.jpg](https://s1.cdn.autoevolution.com/images/news/gallery/red-bull-racing-beats-record-to-set-new-fastest-pit-stop-in-the-history-of-f1_2.jpg)



# Tags

<https://www.formula1.com/content/dam/fom-website/manual/Misc/PORGPMANUALADDS/GettyImages-1306846855.jpg.transform/9col/image.jpg>



# Long Task Timer



<https://i.pinimg.com/originals/fb/5b/98/fb5b98edf68000f44e89d6428a3e2c65.jpg>

# Counter



# Gauge



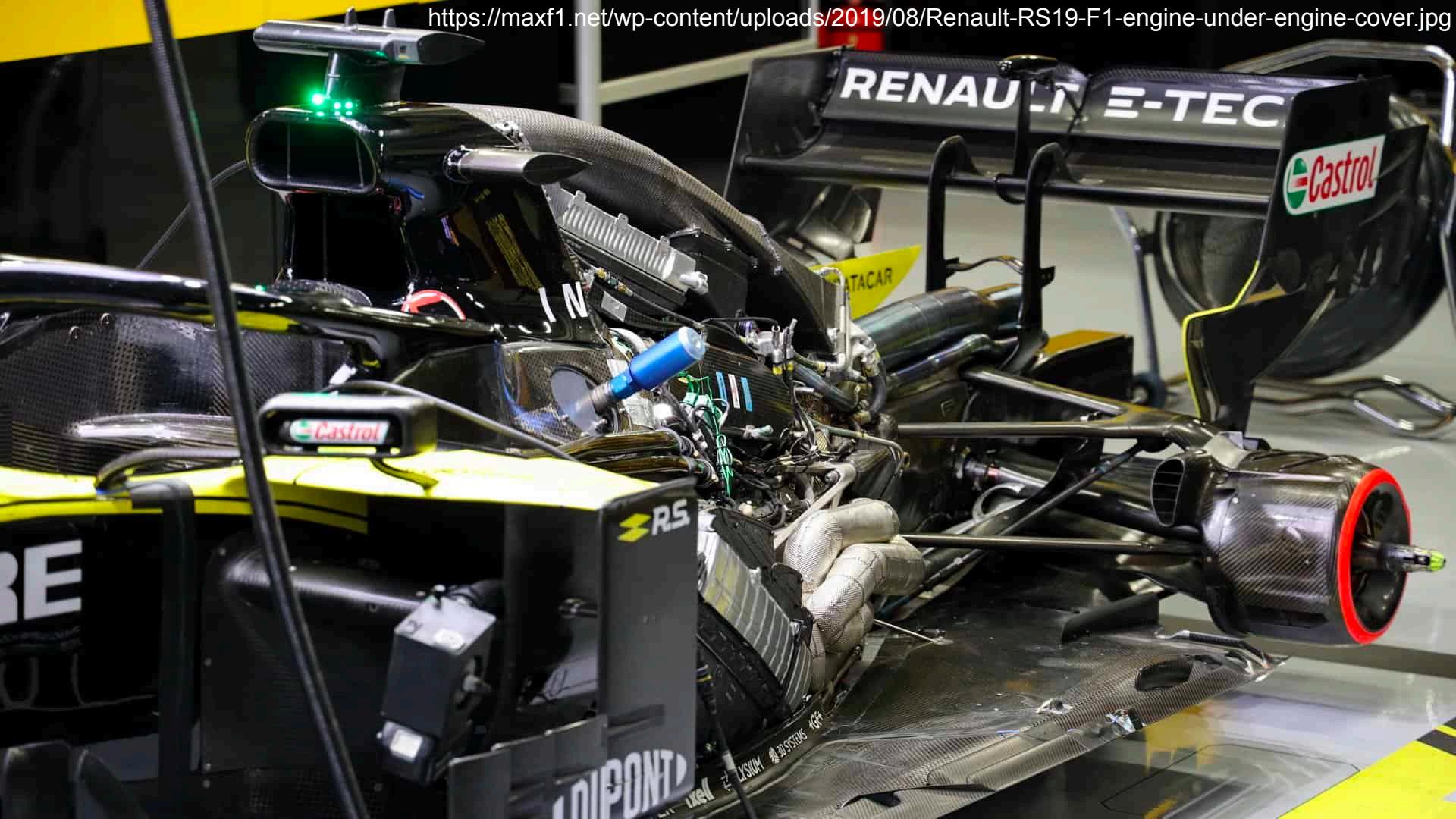
# Exceptions



# Kafka



# Get data out of Actuator



# About security

- Please implement spring security
- Otherwise you will expose your endpoint to others
- And don't forget to add it in your database config!

```
basic_auth:  
    username: "user"  
    password: "pass"
```

<https://www.baeldung.com/spring-security-basic-authentication>

<https://egkazioura.com/2020/05/07/spring-boot-and-micrometer-with-prometheus-part-6-securign-metrics/>

# Sensitive data (!)

- Disable endpoints you don't need

```
management:  
  endpoints:  
    web:  
      exposure:  
        include: 'health'  
    jmx:  
      exposure:  
        exclude: '*'
```

# What do you learn today (goals)?

Define



Create



Save

And visualise

# metrics!

Where to store the metrics?

Which database should I use?

# Databases

- Prometheus
- Graphite
- Atlas
- KairosDB

**Persistent or not persistent?**

## If you want to expose to prometheus

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

## Or Graphite

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-graphite</artifactId>
</dependency>
```



```
# TYPE jvm_buffer_memory_used_bytes gauge
jvm_buffer_memory_used_bytes{id="direct",} 24576.0
jvm_buffer_memory_used_bytes{id="mapped",} 0.0
# HELP process_start_time_seconds Start time of the process since unix epoch.
# TYPE process_start_time_seconds gauge
process_start_time_seconds 1.581090649742E9
# HELP jvm_buffer_total_capacity_bytes An estimate of the total capacity of the buffers in this pool
# TYPE jvm_buffer_total_capacity_bytes gauge
jvm_buffer_total_capacity_bytes{id="direct",} 24576.0
jvm_buffer_total_capacity_bytes{id="mapped",} 0.0
# HELP gauge
# TYPE gauge gauge
gauge 7.0
# HELP logback_events_total Number of error level events that made it to the logs
# TYPE logback_events_total counter
logback_events_total{level="warn",} 0.0
logback_events_total{level="debug",} 0.0
logback_events_total{level="error",} 0.0
logback_events_total{level="trace",} 0.0
logback_events_total{level="info",} 8.0
# HELP counterBuilder_total
# TYPE counterBuilder_total counter
counterBuilder_total{continent="USA",} 0.0
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for the Java virtual machine to use
# TYPE jvm_memory_committed_bytes gauge
jvm_memory_committed_bytes{area="heap",id="PS Survivor Space",} 1.3631488E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen",} 1.76160768E8
jvm_memory_committed_bytes{area="heap",id="PS Eden Space",} 1.42606336E8
jvm_memory_committed_bytes{area="nonheap",id="Metaspace",} 3.9452672E7
jvm_memory_committed_bytes{area="nonheap",id="Code Cache",} 9568256.0
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space",} 5636096.0
# HELP process_cpu_usage The "recent cpu usage" for the Java Virtual Machine process
# TYPE process_cpu_usage gauge
process_cpu_usage 0.0
# HELP system_cpu_count The number of processors available to the Java virtual machine
# TYPE system_cpu_count gauge
system_cpu_count 16.0
# HELP gaugeSimple
# TYPE gaugeSimple gauge
gaugeSimple 7.0
# HELP jvm_threads_peak_threads The peak live thread count since the Java virtual machine started or peak was reset
# TYPE jvm_threads_peak_threads gauge
jvm_threads_peak_threads 29.0
# HELP http_server_requests_seconds
# TYPE http_server_requests_seconds summary
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/metrics",} 1.0
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/metrics",} 0.017340408
# HELP http_server_requests_seconds_max
# TYPE http_server_requests_seconds_max gauge
http_server_requests_seconds_max{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/metrics",} 0.017340408
# HELP process_files_max_files The maximum file descriptor count
# TYPE process_files_max_files gauge
process_files_max_files 10240.0
# HELP gaugeBuilder
# TYPE gaugeBuilder gauge
gaugeBuilder 7.0
```

# What do you learn today (goals)?

Define



Create



Save



And visualise

# metrics!

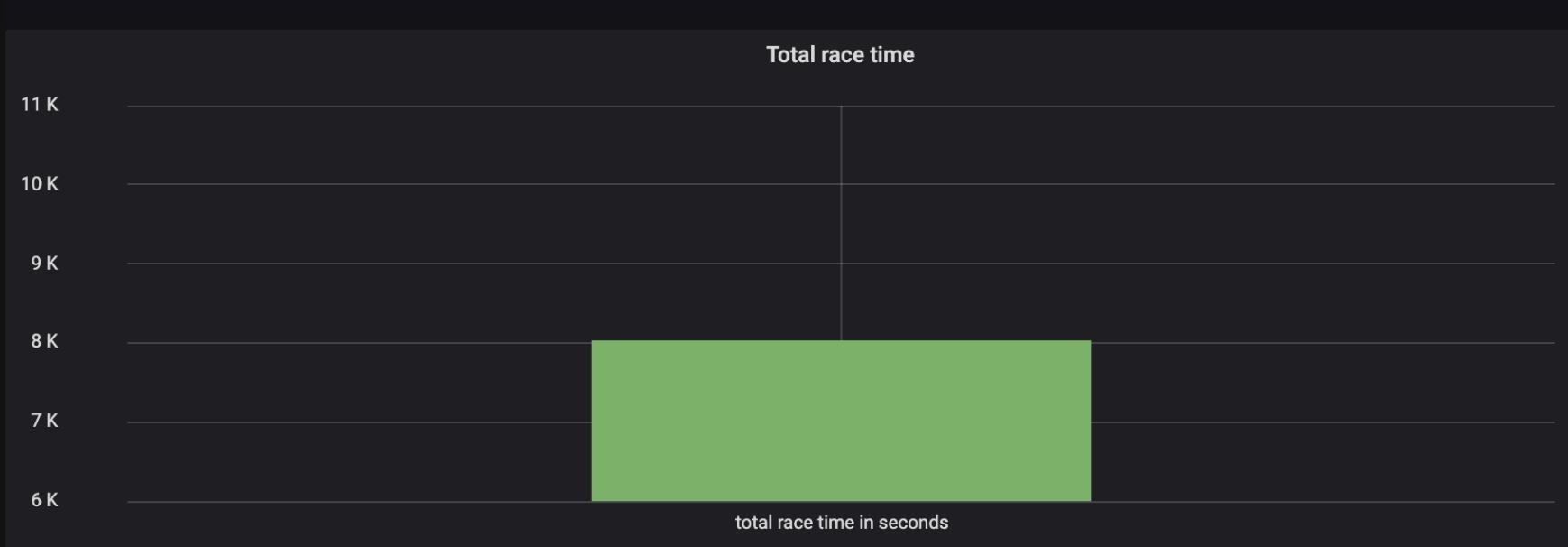
How to visualise the application metrics?  
Which monitoring system to use?

# Systems

- Grafana
- Datadog
- Dynatrace
- Instant
- WaveFront
- and many more

# Let's deep dive into Grafana

# Long Task Timer (duration)



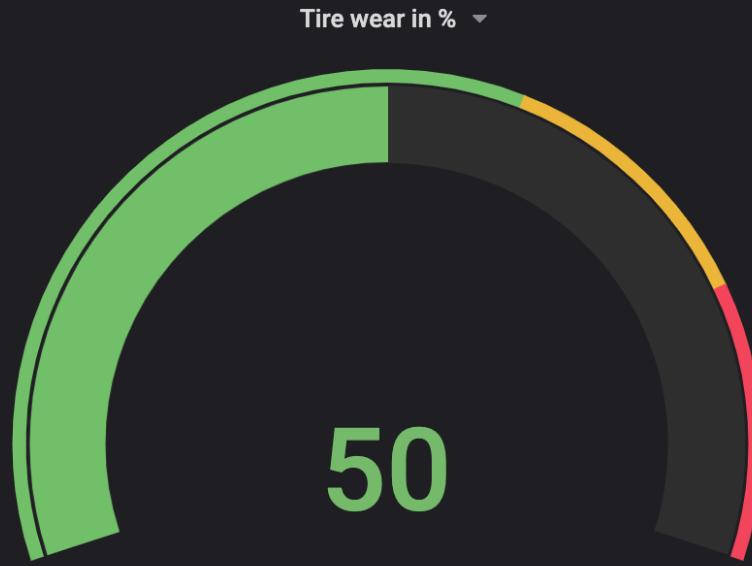
# Timer (count)



# Timer (max)



# Gauge



# Bindings

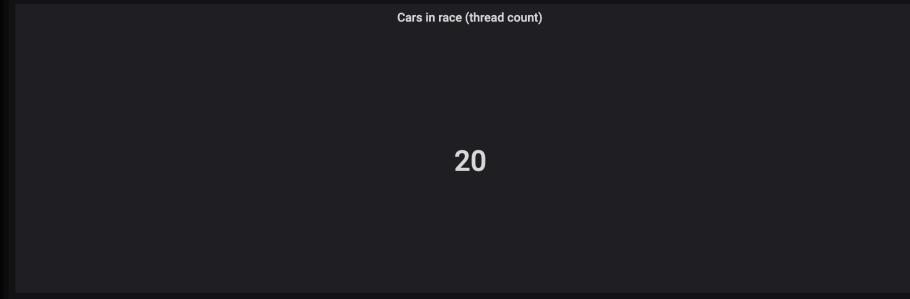
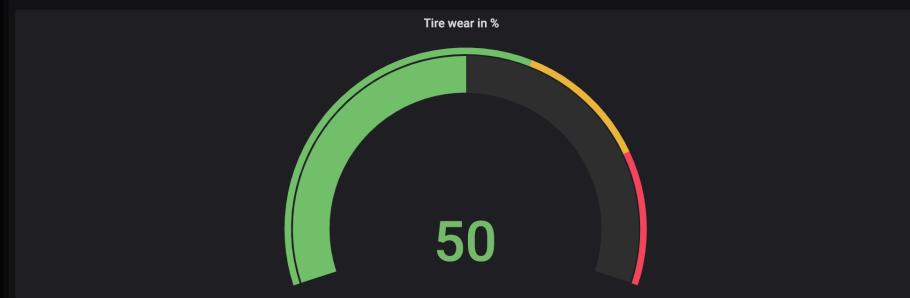
Cars in race (thread count)

20



Race-Demo -

>Last 6 hours ▾



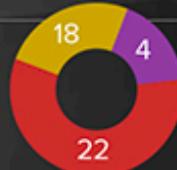
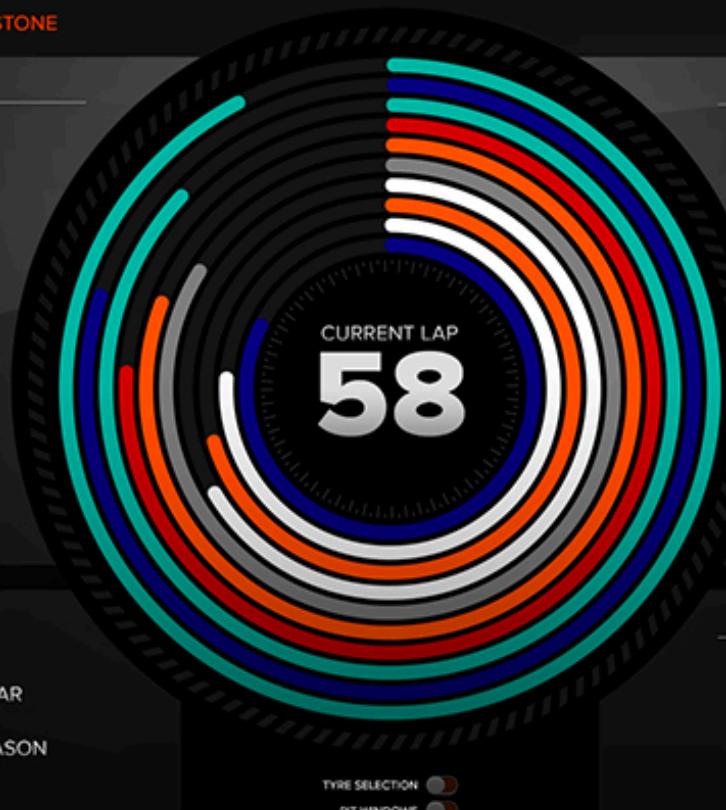
# PURE PITWALL

2016 FORMULA 1 BRITISH GRAND PRIX // SILVERSTONE

TRACK TEMP: 29°C AIR TEMP: 20°C

## PIT STOP TIMES

ROS	FASTEAST
VET	+0.51
BUT	+0.74
SOT	+0.78
RIC	+0.90
HUL	+1.12
HAM	+1.19
MAS	+1.34
ALO	+1.95
RAI	+2.40



## TYRE COMPOUNDS

### SETS USED

- SUPER SOFT
- ULTRA SOFT
- SOFT

ROS 1:40.418

FASTEST LAPS: 10/67



TRACK RECORD: 1:39.768

## SAFETY CAR LAPS

<b>07</b>	LAST YEAR
<b>4</b>	THIS SEASON
<b>28</b>	

TYRE SELECTION  
 PIT WINDOWS

## TOTAL OVERTAKES

WITH DRS	<b>28</b>
FOR LEAD	<b>2</b>
	<b>35</b>



Take your responsibility

Start creating metrics

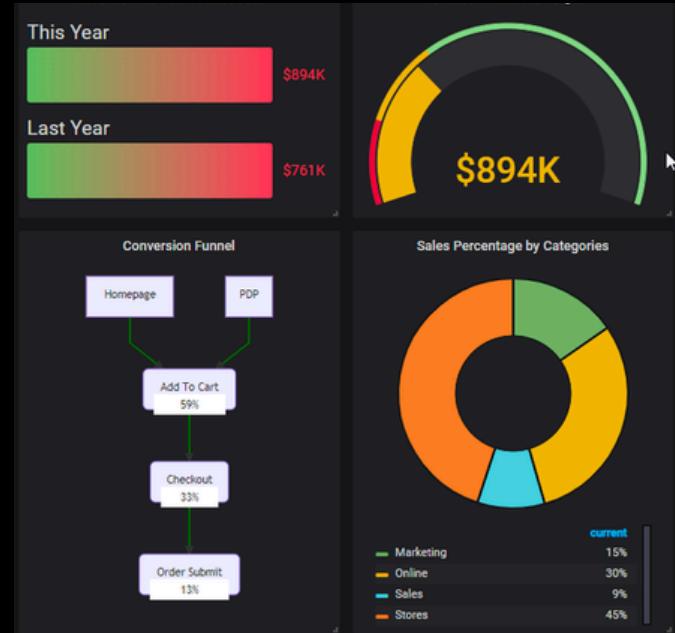


# My three (simple) points

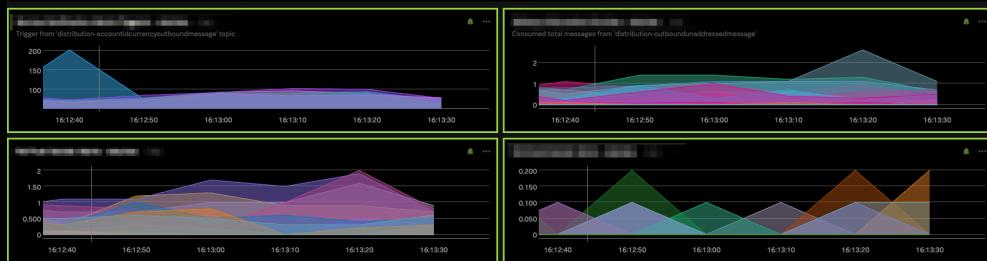
## Exceptions



## Own logic



## Performance





**AND NOW IT'S UP TO YOU**



All examples you can find at:  
**<https://github.com/KoTurk/Micrometer/OpenWeb>**



@KoTurk77

