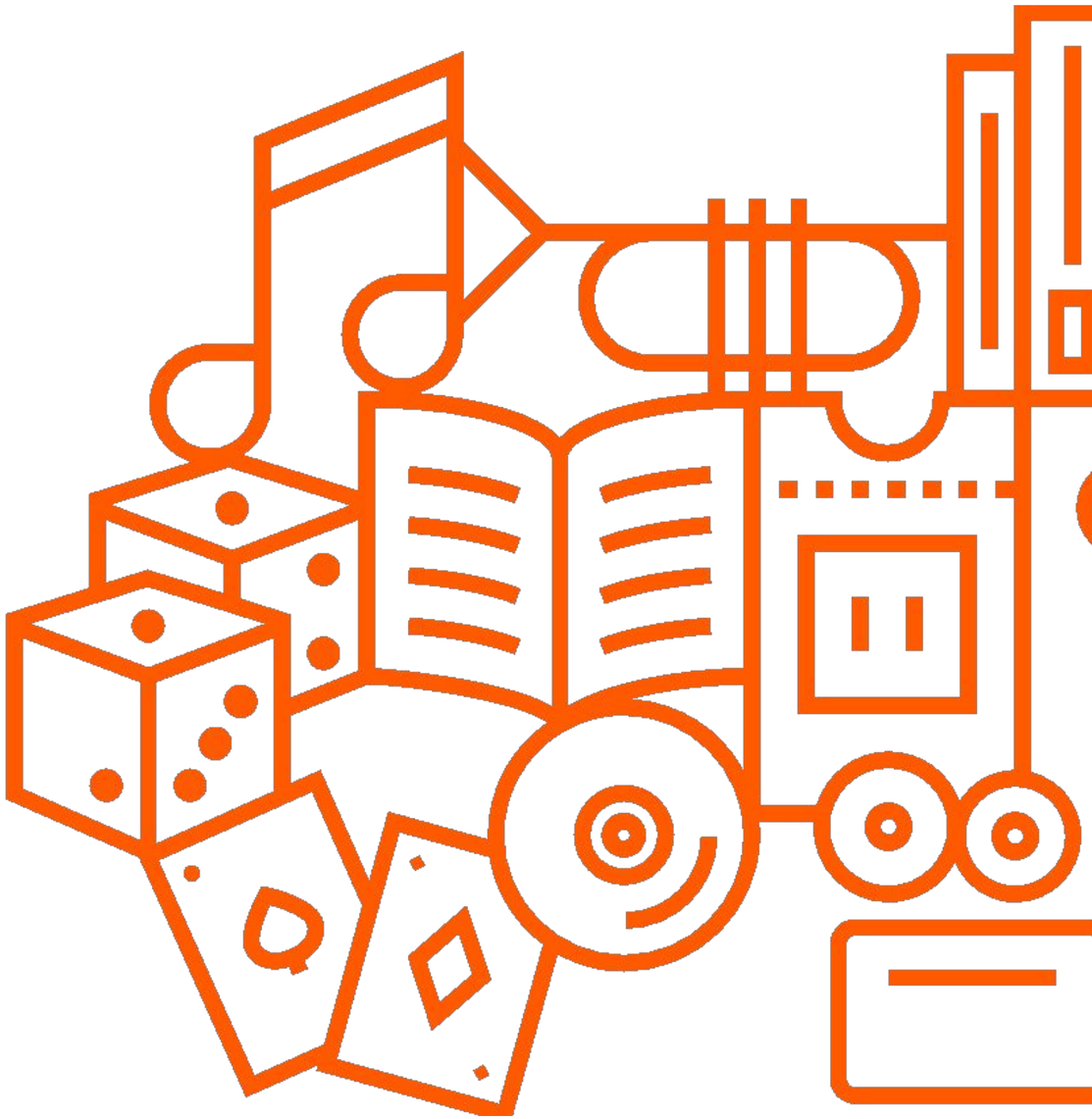
Monitoring aplikacji

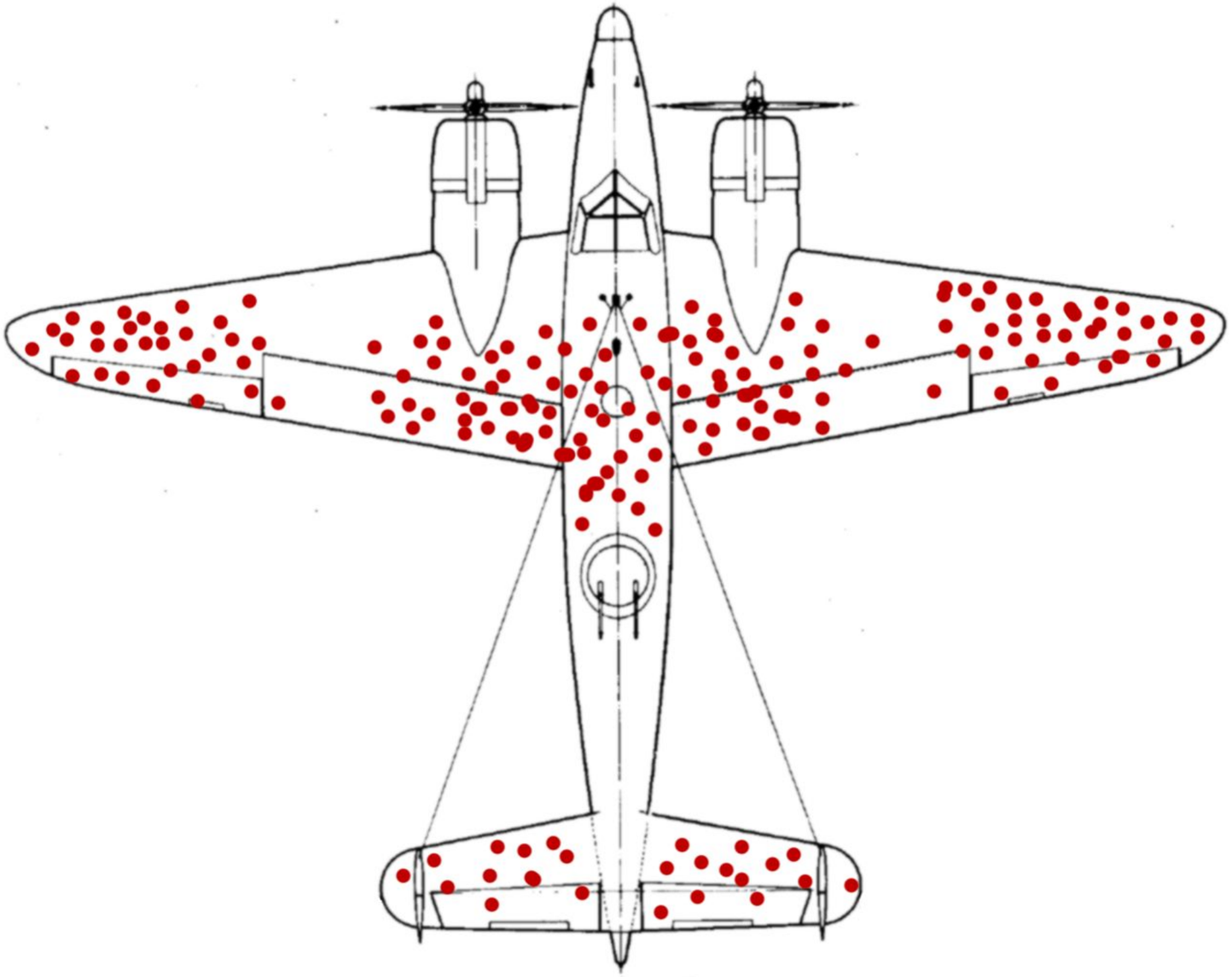
Bartosz Balukiewicz
Software Engineer, Allegro.Tech

JUGADEMY 19.02.2020 Poznań

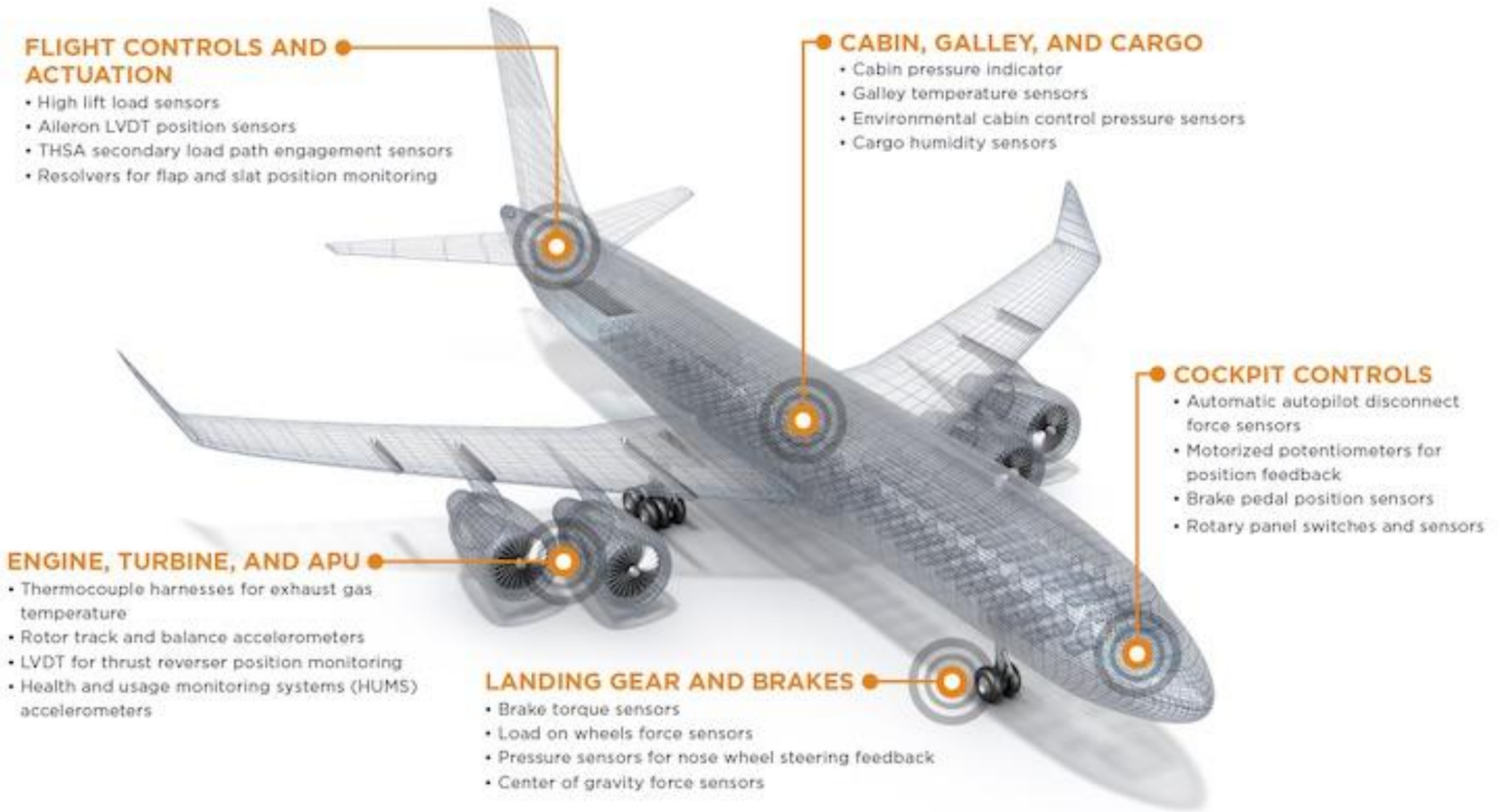
O mnie:

- Software Engineer
- 2 lata w Allegro.Tech
- Smart! i czasy dostawy
-

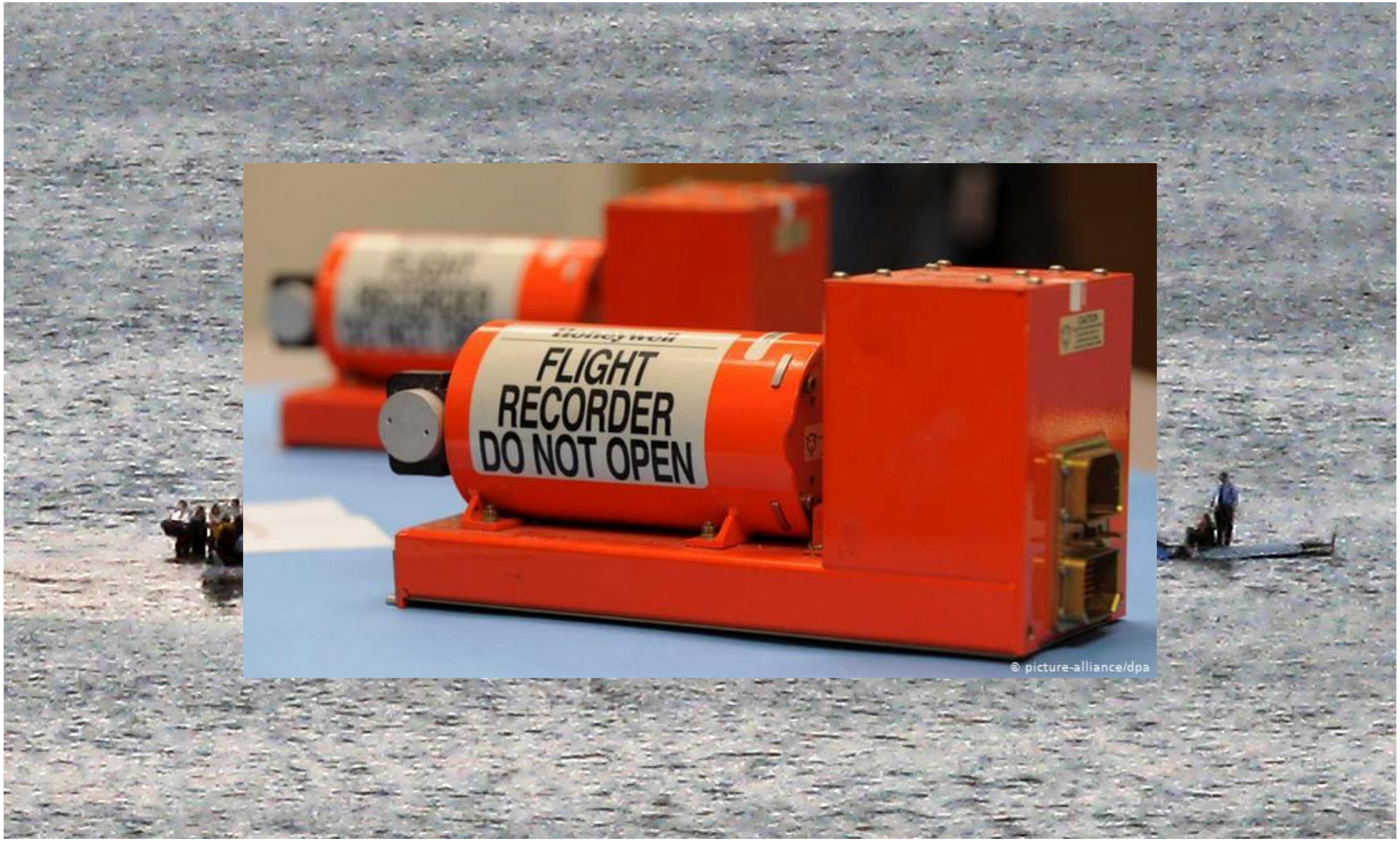


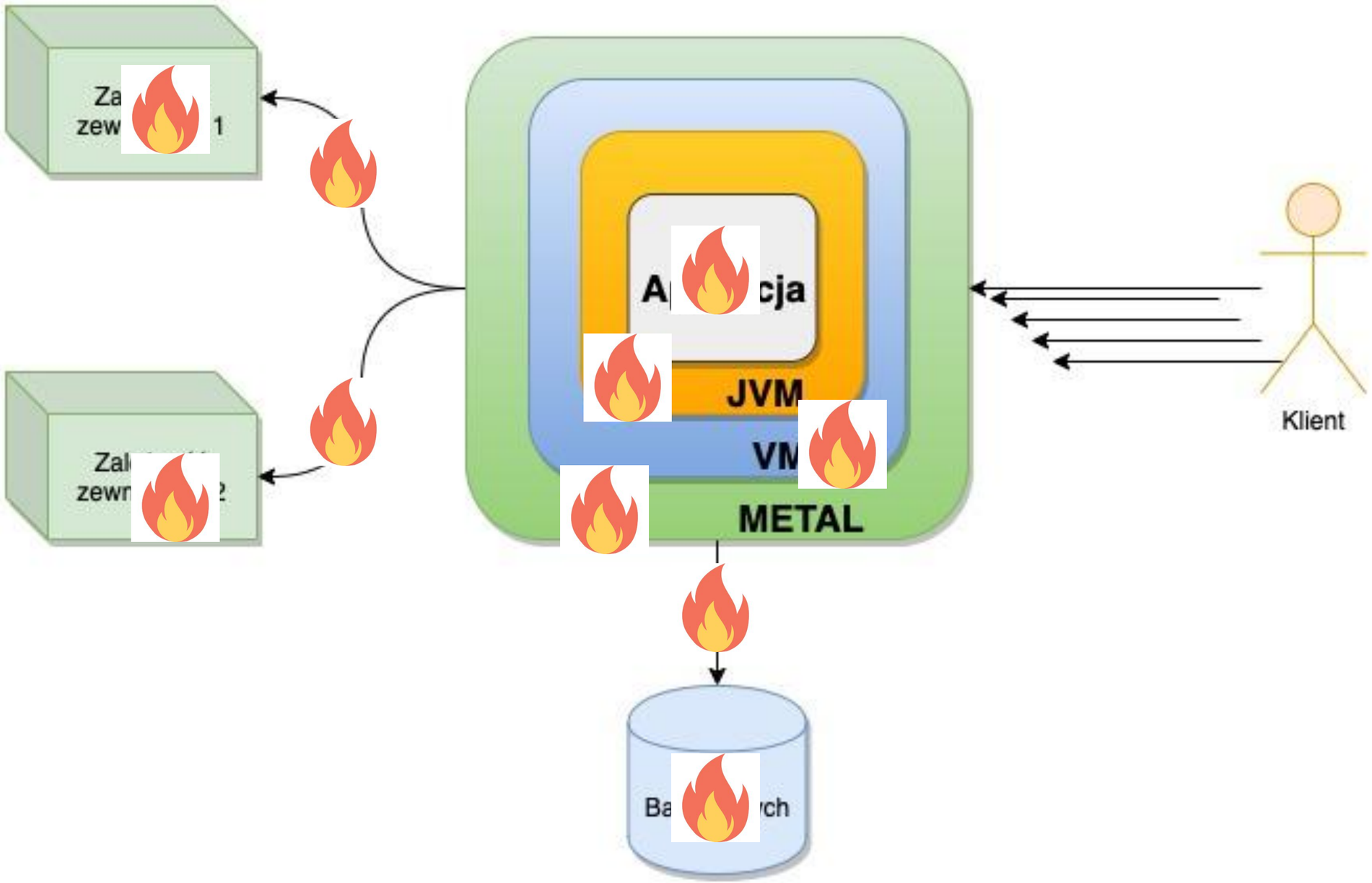


Survivorship Bias



https://roboticsandautomationnews.com/2019/04/15/boeing-737-crash-could-lead-to-planes-being-fitted-with-thousands-more-sensors/21846/


FLIGHT
RECORDER
DO NOT OPEN
© picture-alliance/dpa


Za
zew
1
Zal
zewn
2
A
cja
JVM
VM
METAL
Klient
Ba
rch

Obserwowalność

(Observability)

Logowanie

Logowanie

```java
try {
    doingSomethingVeryRisky();
} catch(MyVeryOwnException exception) {
    exception.printStackTrace();
}
```

```
com.example.demo.MyVeryOwnException
    at com.example.demo.DemoApplication.main(DemoApplication.java:11)
```

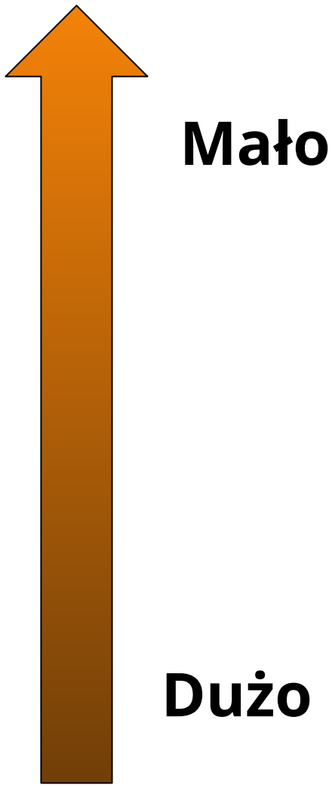
Logowanie

```java
try {
    doingSomethingVeryRisky();
} catch(MyVeryOwnException exception) {
    Logger logger = LoggerFactory.getLogger(DemoApplication.class);
    logger.error("Got very bad exception :(", exception);
}
```

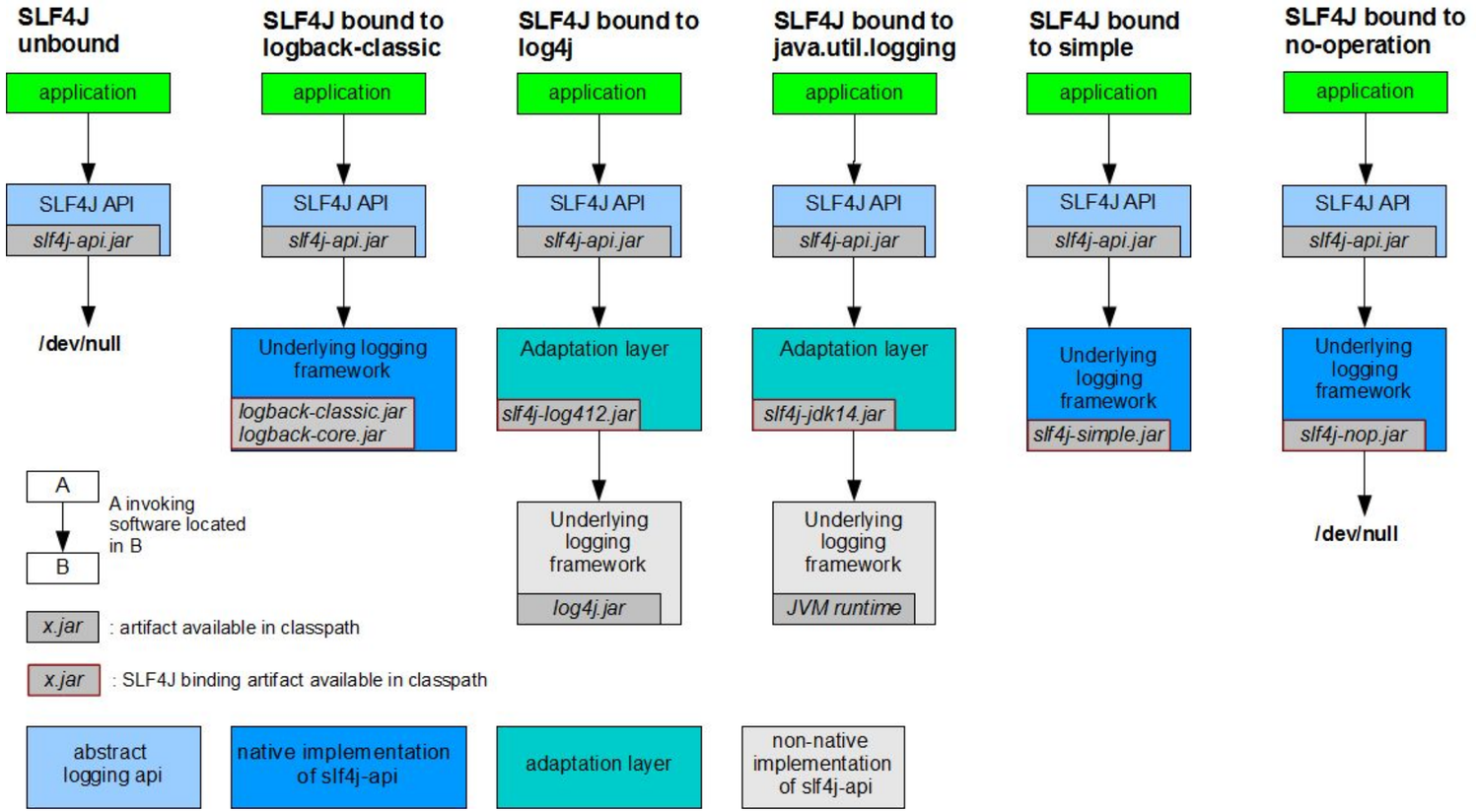
```
18:14:56.130 [main] ERROR com.example.demo.DemoApplication - Got very bad exception :(
com.example.demo.MyVeryOwnException: We've got a problem!
    at com.example.demo.DemoApplication.doingSomethingVeryRisky(DemoApplication.java:22)
    at com.example.demo.DemoApplication.main(DemoApplication.java:15)
```

Poziomy logowania (co logujemy?)

- ***FATAL - ???***
- ***ERROR*** - jest źle, trzeba działać
- ***WARN*** - jest źle
- ***INFO*** - normalna informacja
- ***DEBUG*** - tylko dewelopment
- ***TRACE***



Simple Logging Facade for Java (SLF4J)



- lepsza konfiguracja
- lepsze API (np. poziomy logowania)
- pisanie do konsoli, plików, internetu?
- jak na przykładzie - większy kontekst

http://www.slf4j.org/manual.html // http://nurkiewicz.github.io/talks/2014/jinkubator/#/4

Logback

```xml
<configuration>

    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
            <pattern>
                %d{dd-MM-yyyy HH:mm:ss.SSS} %magenta([%thread]) %highlight(%-5level) %logger{36}.%M - %msg%n
            </pattern>
        </encoder>
    </appender>

    <root level="info">
        <appender-ref ref="STDOUT"/>
    </root>

    <logger name="com.example.demo" level="debug">
        <appender-ref ref="STDOUT" />
    </logger>

</configuration>
```

```
15-02-2020 16:35:24.642 [main] ERROR com.example.demo.DemoApplication.main - Got very bad exception :(
Up the stack trace ⌥⌘↑ o.MyVeryOwnException: We've got a problem!
    at com.example.demo.DemoApplication.doingSomethingVeryRisky(DemoApplication.java:23)
    at com.example.demo.DemoApplication.main(DemoApplication.java:15)
```

https://www.codingame.com/playgrounds/4497/configuring-logback-with-spring-boot

Logback

```
<appender name="SAVE-TO-FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">

    <file>${LOG_PATH}/log.log</file>

    <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
        <Pattern>
            %d{dd-MM-yyyy HH:mm:ss.SSS} [%thread] %-5level %logger{36}.%M - %msg%n
        </Pattern>
    </encoder>

    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
        <fileNamePattern>
            ${LOG_PATH}/archived/log_%d{dd-MM-yyyy}.log
        </fileNamePattern>
        <maxHistory>10</maxHistory>
        <totalSizeCap>100MB</totalSizeCap>
    </rollingPolicy>

</appender>
```

http://logback.qos.ch/manual/appenders.html

Dobre logowanie 101

1. Opisowe logi

```
18-02-2020 11:18:25.384 [main] INFO  com.example.demo.DemoApplication.main - Created transaction!

15-02-2020 17:17:59.693 [main] INFO  com.example.demo.DemoApplication.main - Created transaction with id: 123456 for user:
 bartek123
```

2. Mądrze używajmy poziomów logowania
3. Zastanówmy się co logujemy...

```
15-02-2020 17:19:33.867 [main] DEBUG com.example.demo.DemoApplication.main - Reset password for user: bartek123. New password: happy!@#cats
```

4. Trace (rzecz do doczytania: distributed tracing)

```
15-02-2020 17:23:19.791 [main] DEBUG com.example.demo.DemoApplication.main - Transaction: uuid-uuid2-uuid 3 Creating account for user: bartek123
15-02-2020 17:23:19.794 [main] DEBUG com.example.demo.DemoApplication.main - Transaction: uuid-uuid2-uuid 3 Account created for user: bartek123
15-02-2020 17:23:19.794 [main] DEBUG com.example.demo.DemoApplication.main - Transaction: uuid-uuid2-uuid 3 Processing finished for user: bartek123
```

Dobre logowanie 101

6. Szanujmy zasoby - AsyncAppender

7. Leniwa ewaluacja Stringów (lazy evaluation :))

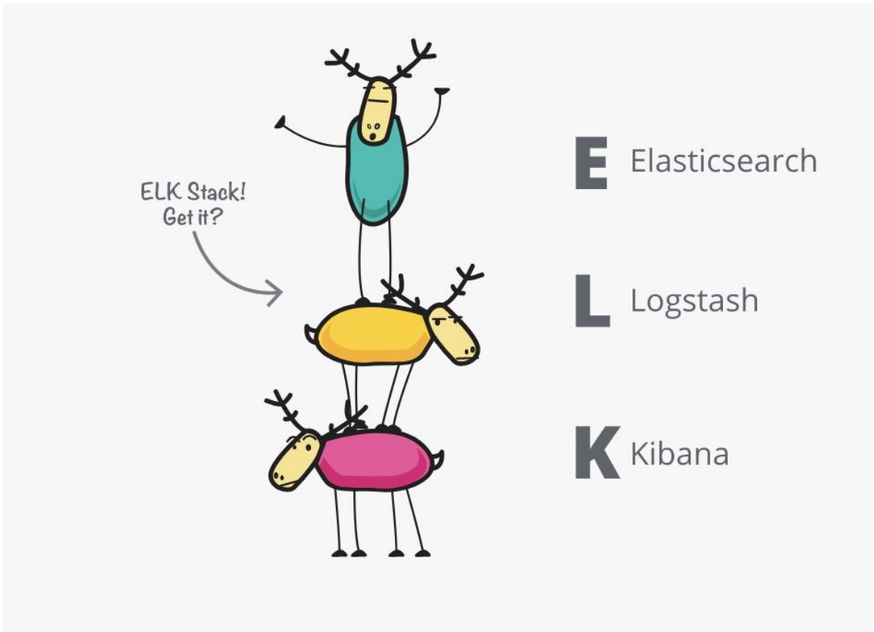
```
logger.debug("Transaction:" + transaction.getId() + "Creating account for user:" + user.getId());
```

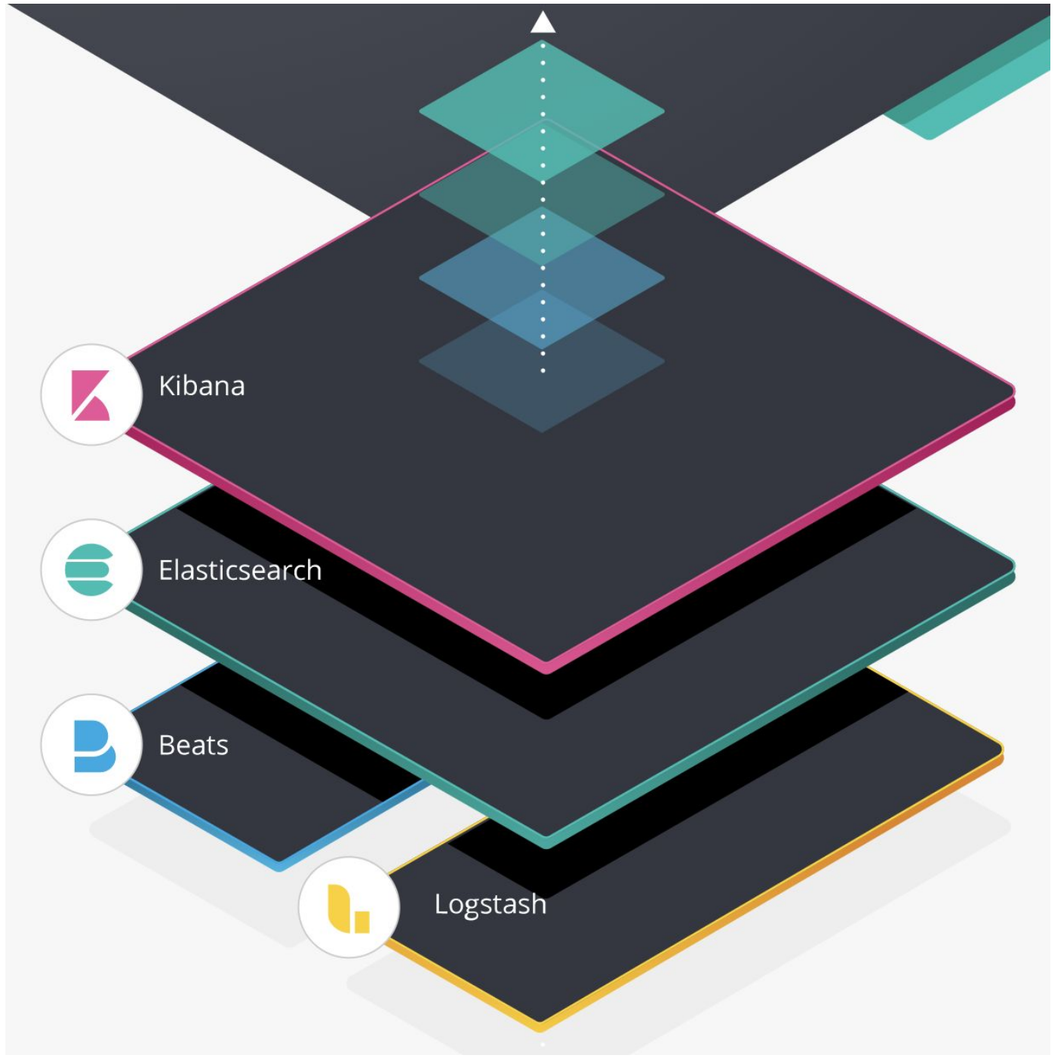
```
logger.debug("Transaction: {} Creating account for user: {}", transaction.getId(), user.getId());
```

```
logger.atDebug()
        .addArgument(transaction.getId())
        .addArgument(user.getId())
        .log("Transaction: {} Creating account for user: {}");
```

Mam te logi i co dalej

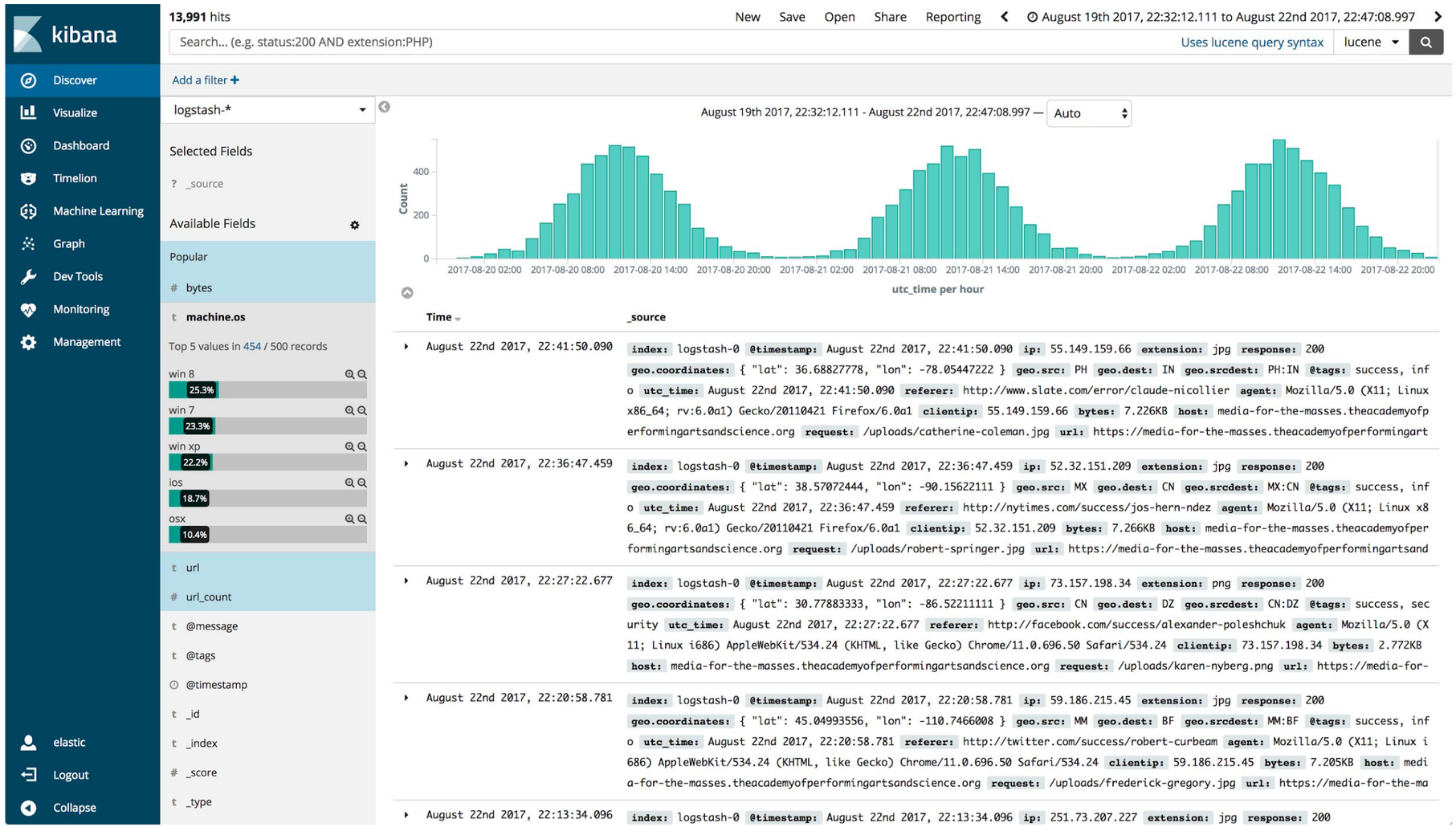
1. Mała skala - *tail* i *grep*
2. ...?





https://www.elastic.co/what-is/elk-stack

ELK Stack



https://www.elastic.co/blog/making-kibana-accessible

Monitorowanie

Fallacies of distributed computing

1. **The network is reliable**
2. **Latency is zero**
3. **Bandwidth is infinite**
4. **The network is secure**
5. **Topology doesn't change**
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

Średnia i percentyle

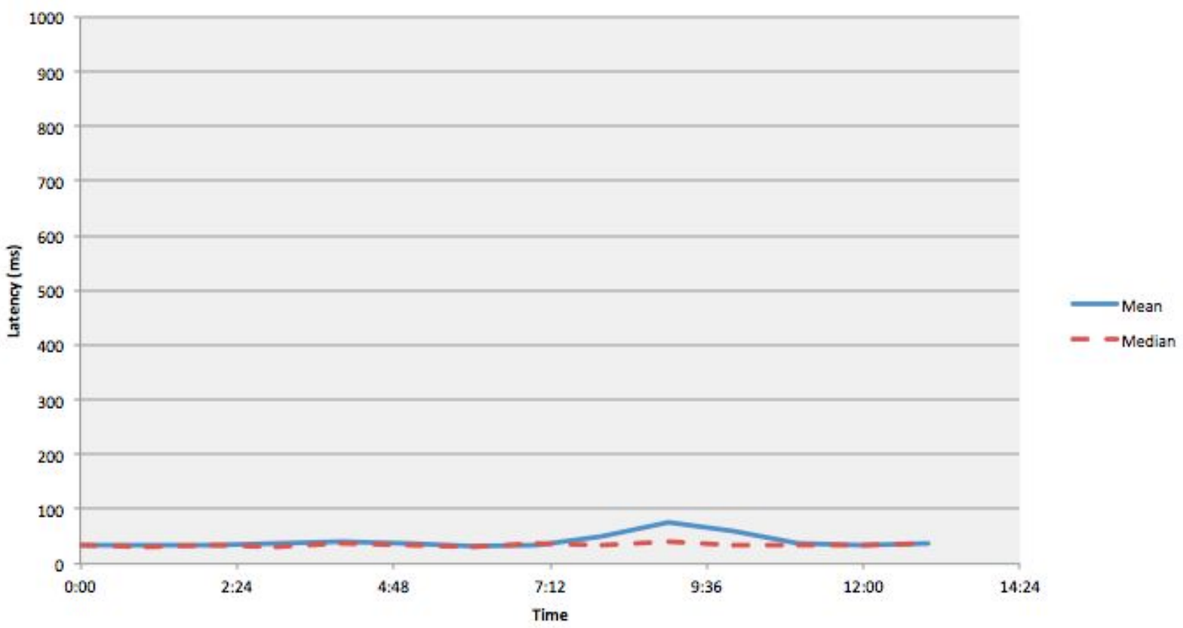
100, 100, 100, 100, 1_000, 100, 100, 100, 100, 1_000, 100, 100

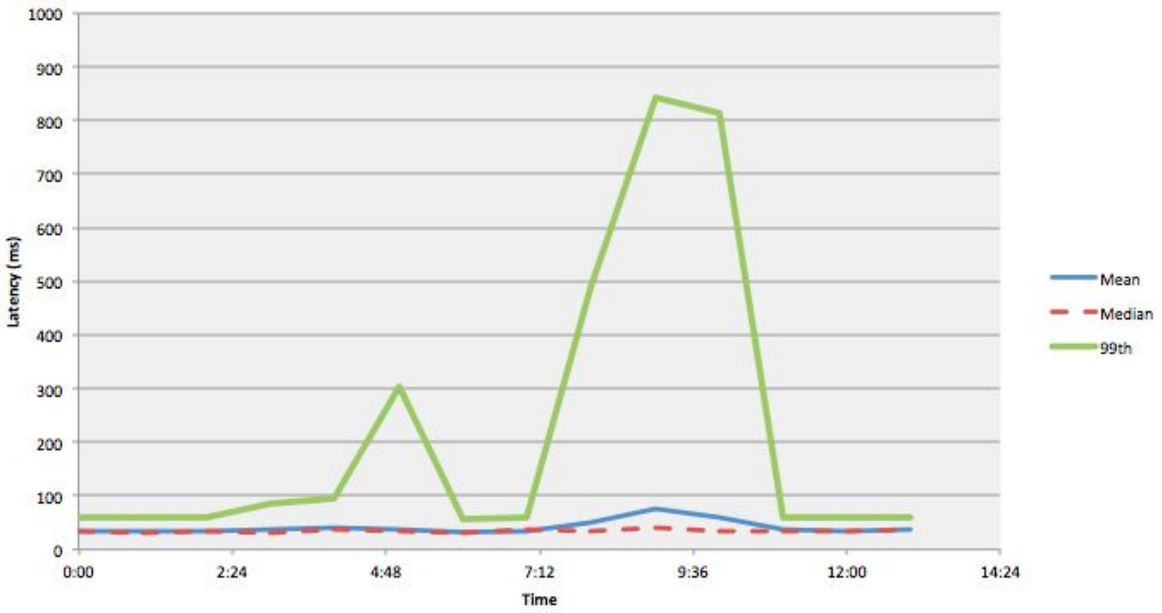
Średnia: **250**

Mediana (50 percentyl) : **100**

99 percentyl: **1000**

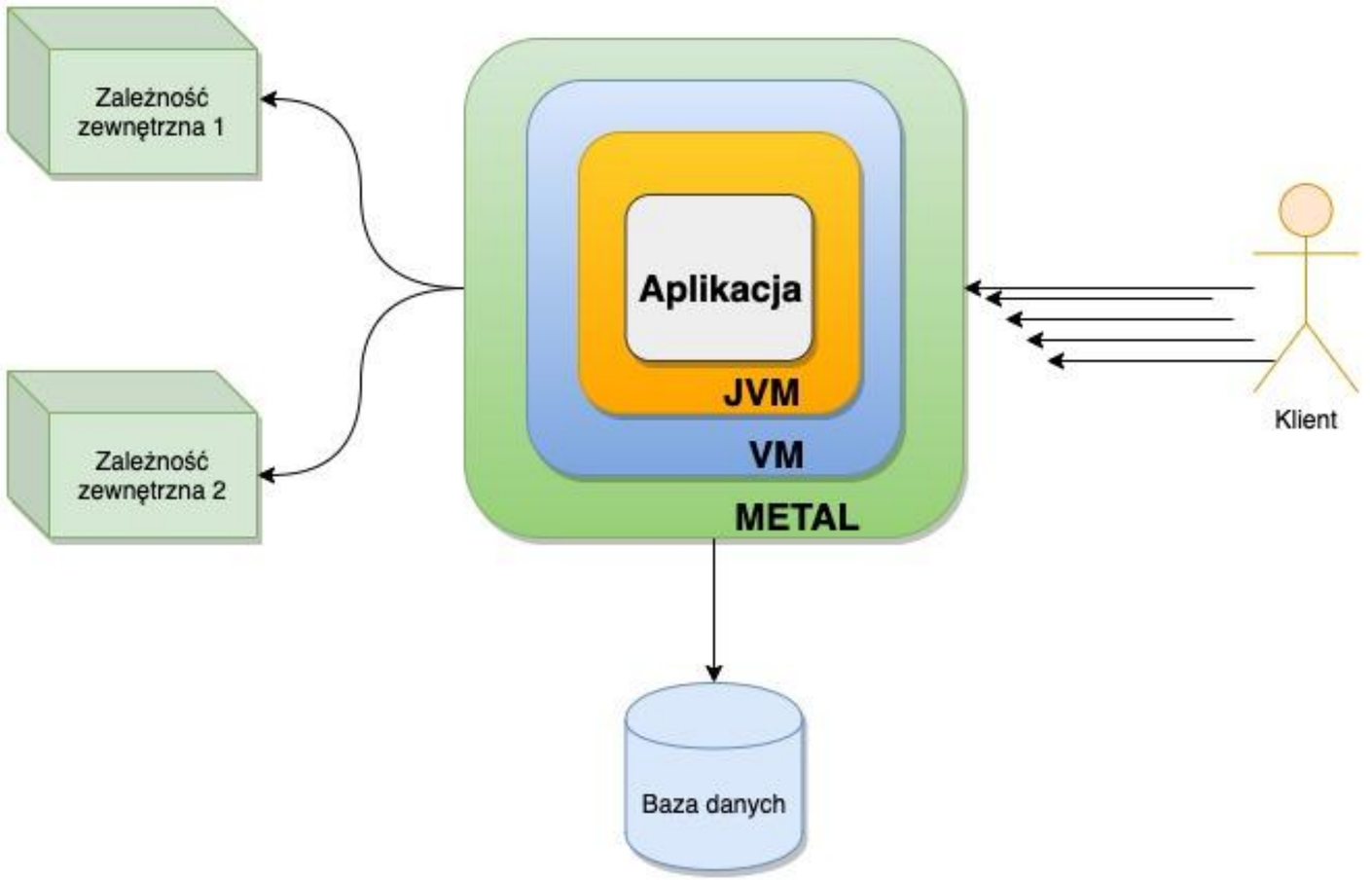
Średnia i percentyle





https://www.elastic.co/blog/averages-can-dangerous-use-percentile

Co monitorować?



Co monitorować?

- Healthcheck
- Czas odpowiedzi (p99, p999?...)
- Kody odpowiedzi (2xx, 4xx, 5xx)
- Liczba zapytań na sekundę/minutę (rps, rpm)
- Czasy odpowiedzi zależności (+ liczba zapytań / liczba błędów)
- Zużycie CPU, pamięci, pamięci JVM, liczba odpalonych GC, czas GC ...
- **metryki biznesowe, np. liczba rejestracji na minutę, liczba transakcji o kwocie powyżej 1000 PLN itp.**

Kontrakty

- Key Performance Indicator (**KPI**)

  Stosunek liczby rejestracji do liczby wejść na stronę ma wynosić 20%.

- Service-level Agreement (**SLA**)

  99 percentyl czasów odpowiedzi nie może przekraczać 500 milisekund.

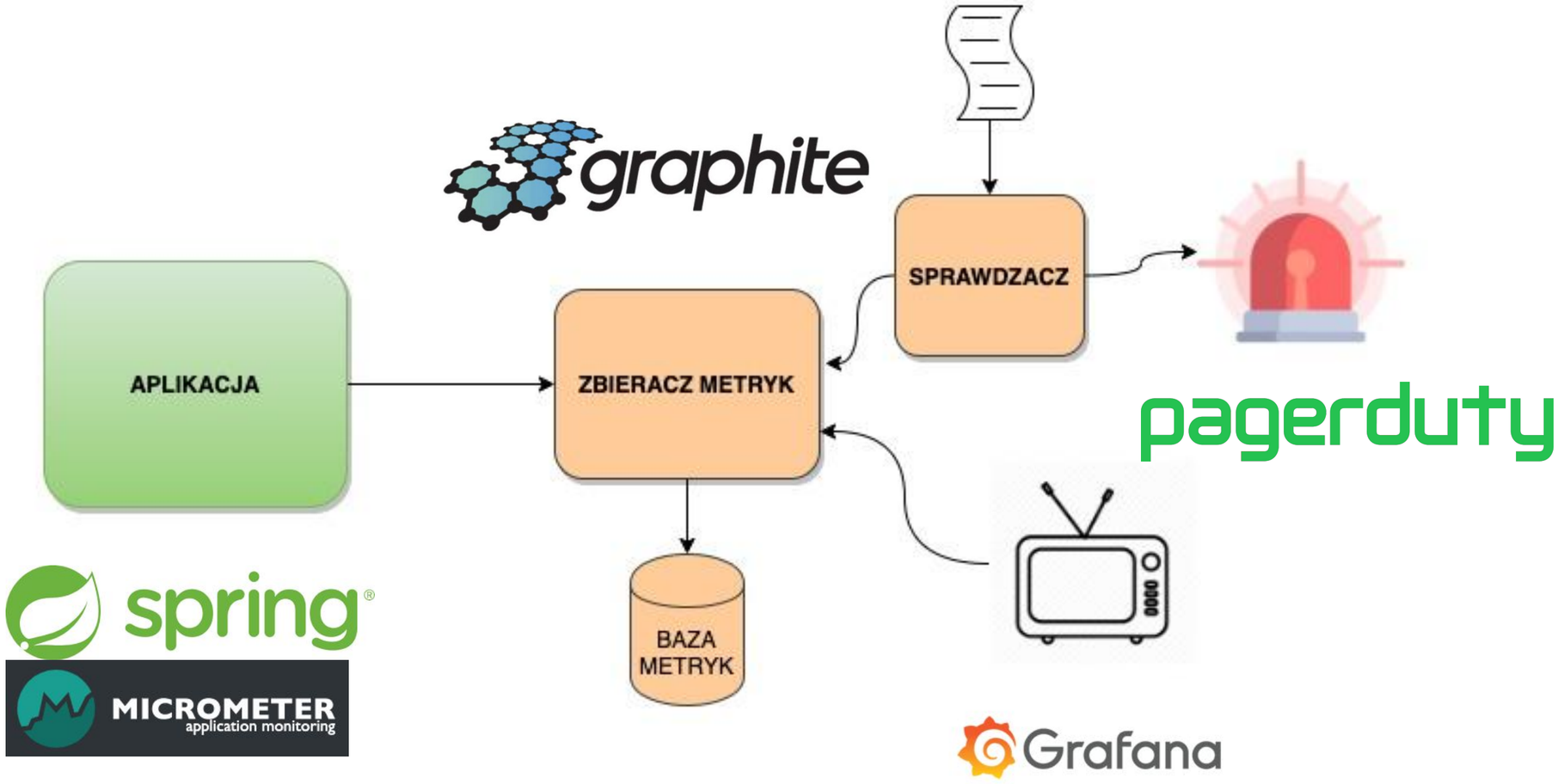
Prawdziwy monitoring

- p99 <= 500ms
- liczba 5xx w ciągu ostatniej minuty < 10
- healthcheck == 200
- ...

Ogólny monitoring we wszystkich aplikacjach

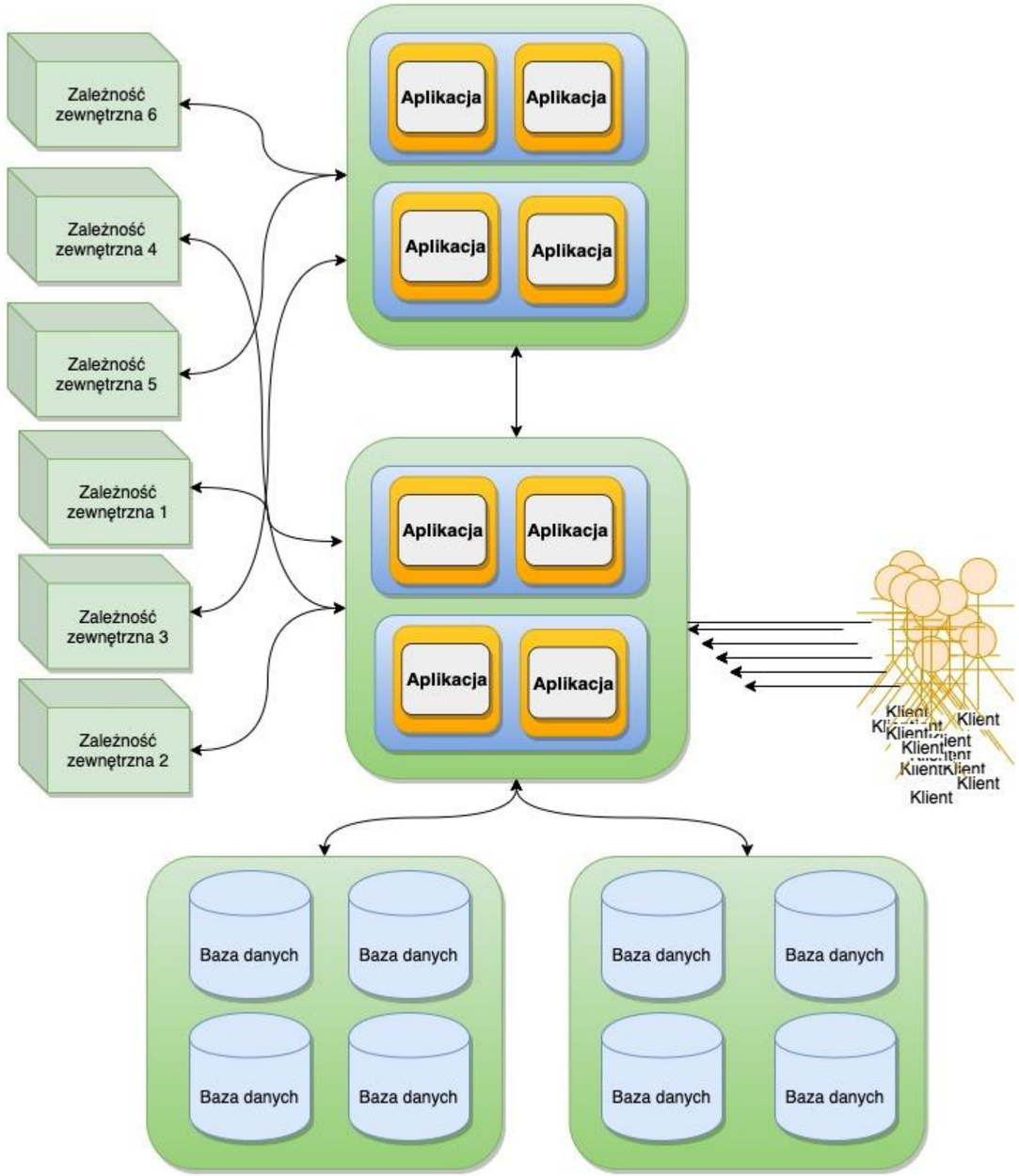
Liczby eksperckie, a pomiary.

Jak to zrobić?

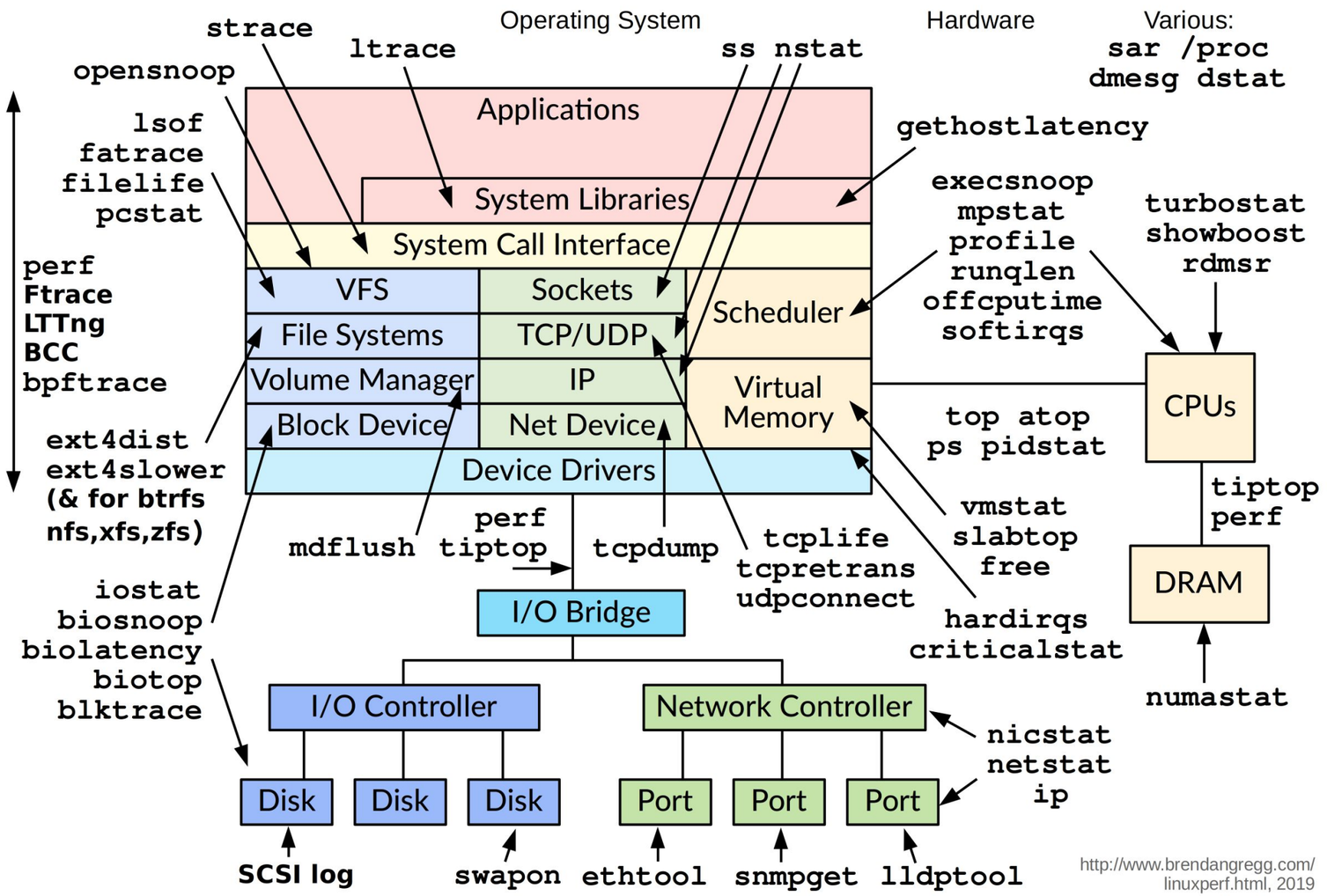


Centrum sterowania :)

A tak jest naprawdę



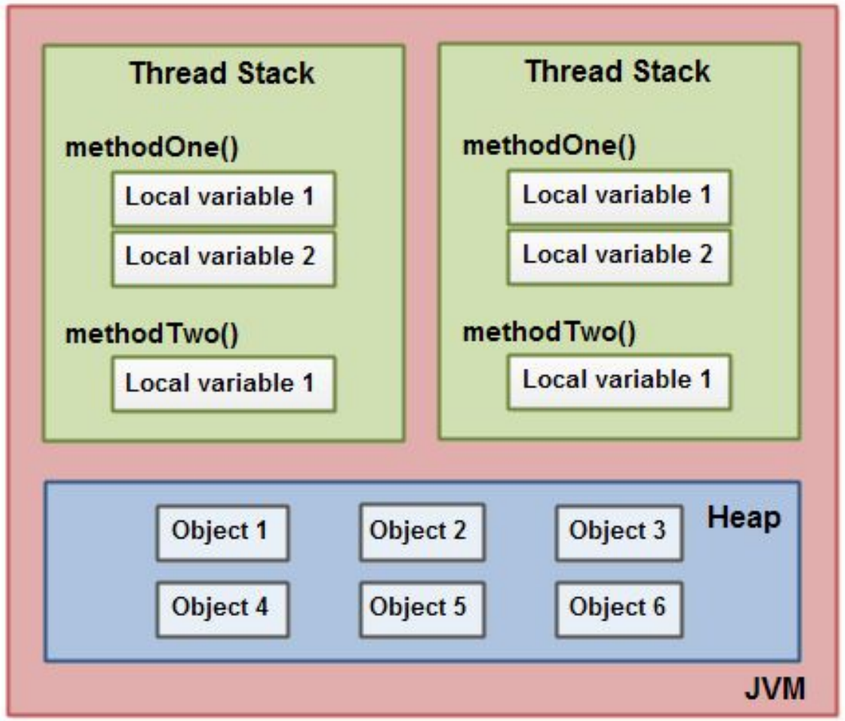
VM i System

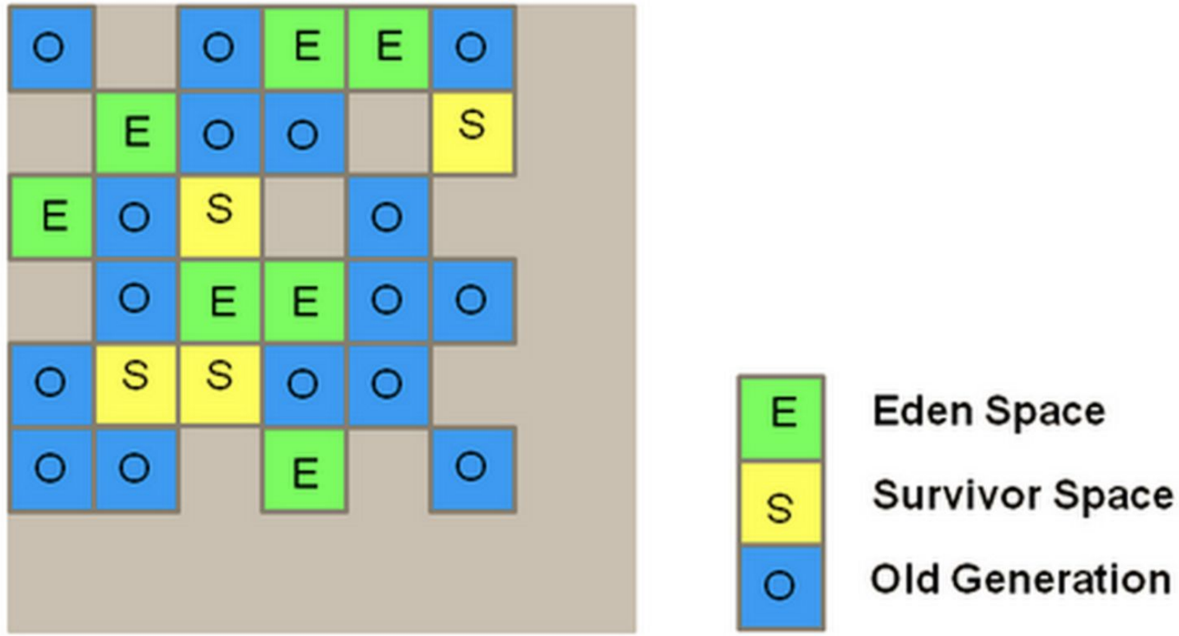


http://www.brendangregg.com/linuxperf.html via http://nurkiewicz.github.io/talks/2014/jinkubator/#/57

JVM

Java Memory Model





http://tutorials.jenkov.com/java-concurrency/java-memory-model.html

https://www.journaldev.com/2856/java-jvm-memory-model-memory-management-in-java

GC LOGI

https://dzone.com/articles/disruptive-changes-to-gc-logging-in-java-9

>= JAVA 9

-Xlog:gc*:verbose_gc.log:time

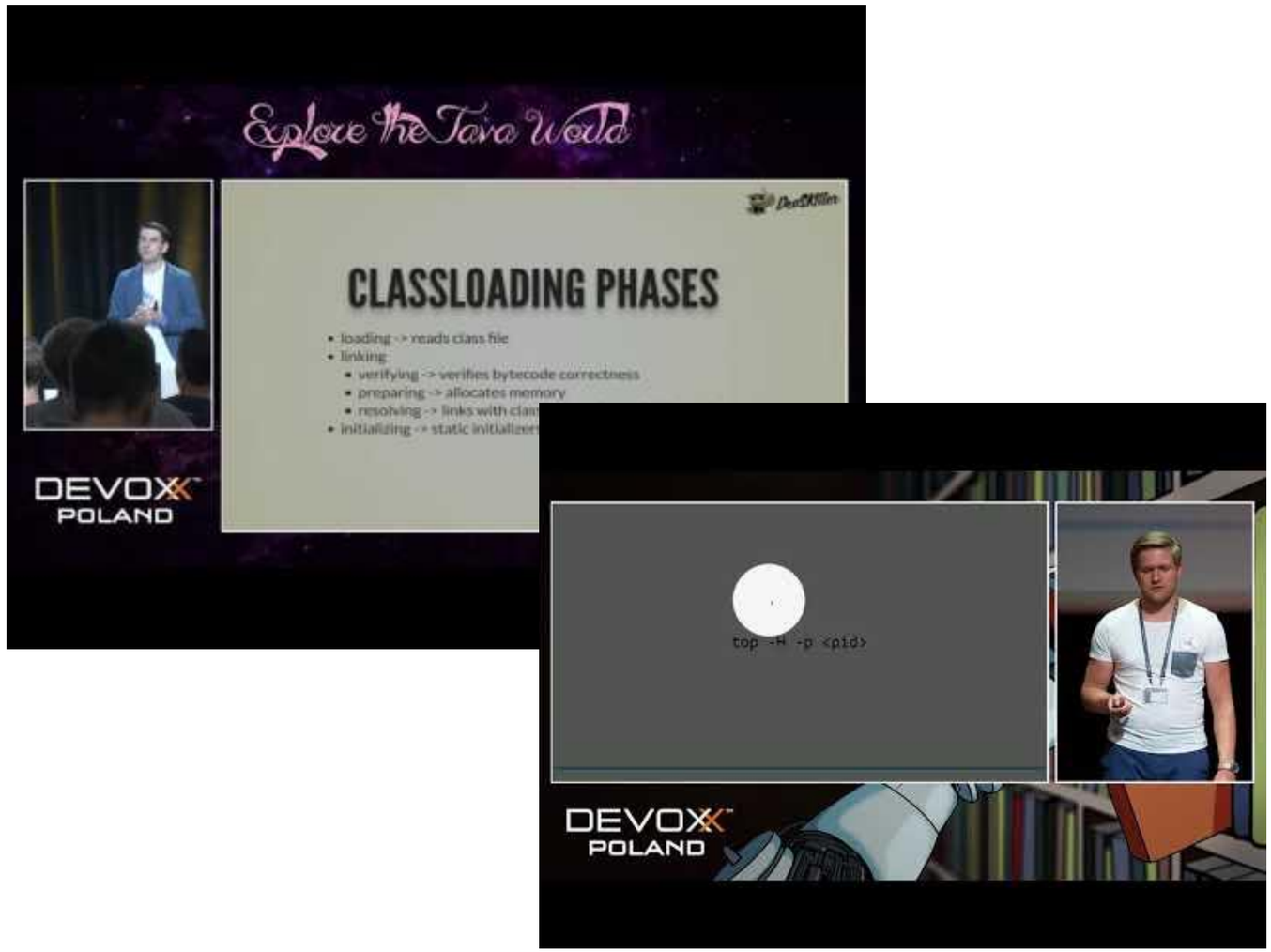
<= JAVA 8

-Xloggc:gc.log -XX:+PrintGCDetails -XX:+PrintGCDateStamps

-XX:+PrintGCCause

Narzędzia

- thread-dump
- head-dump
- Profilowanie aplikacji
- Monitoring JVM
- przeglądanie JITa

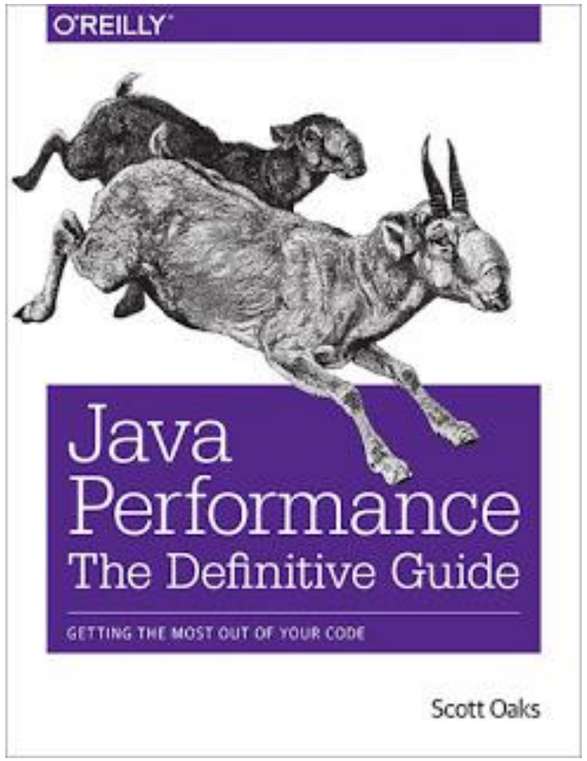


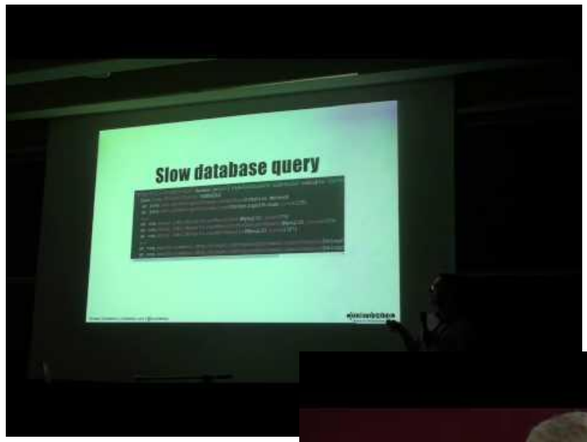
Demo

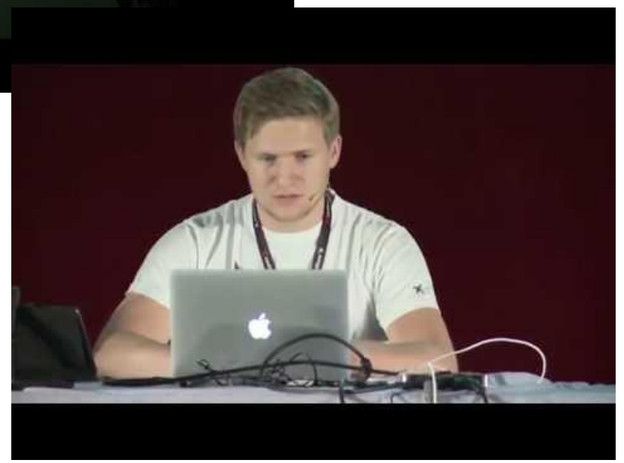
https://github.com/spooz/pjug-monitoring-demo

Co dalej?


The Pragmatic Programmers
Release It!
Second Edition
Design and Deploy
Production-Ready Software
Michael T. Nygard
Edited by Katharine Dvorak


O'REILLY
Java
Performance
The Definitive Guide
GETTING THE MOST OUT OF YOUR CODE
Scott Oaks


Slow database query



Dziękuję za uwagę
i czekam na pytania :)

https://allegro.pl/praca/staze
https://allegro.pl/praca
https://allegro.tech/braincode/