Użycie nowoczesnych bibliotek do logowania w środowisku mikroserwisów

Łukasz Soszyński

Agenda

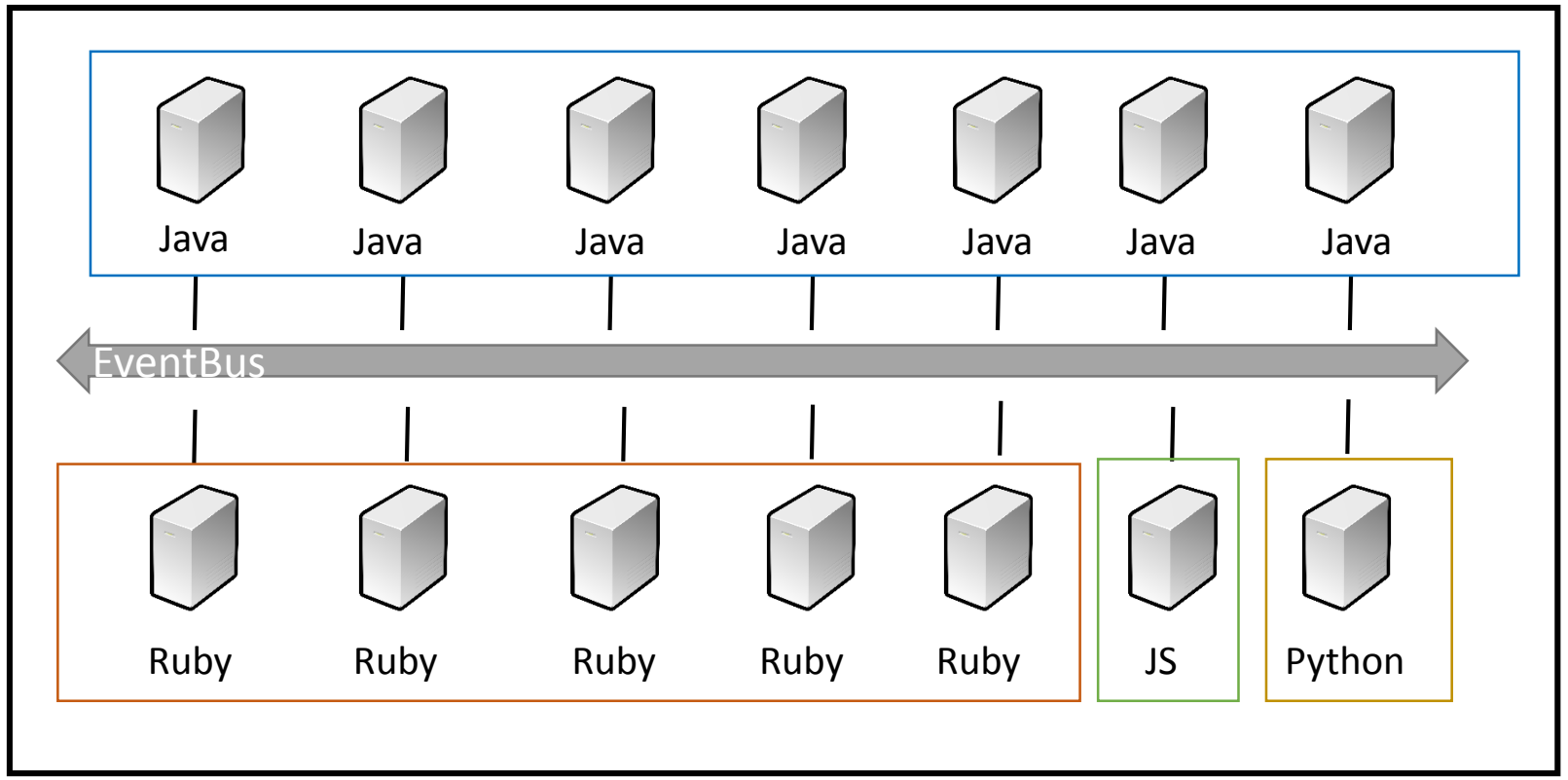
- System zbudowany w architekturze mikroserwisów
- Nowoczesne biblioteki służące do logowana informacji (Log Back)
- Logowanie w systemie rozproszonym (ELK)

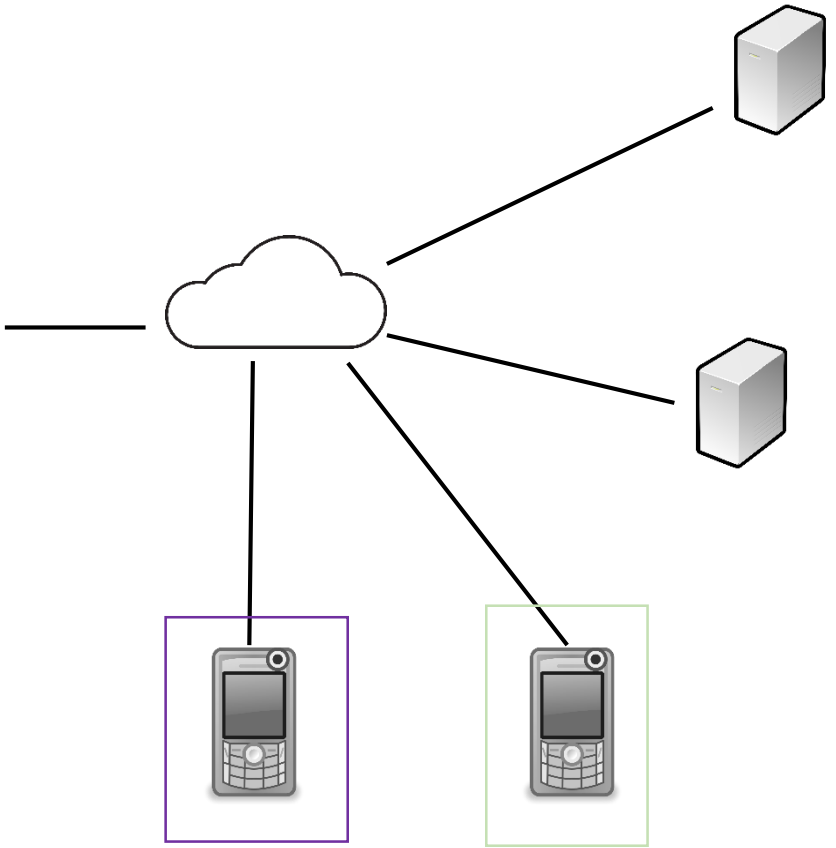
O prowadzącym

- Łukasz Soszyński
  - Spring (różne rodzaje ;)
  - Clean Code
  - TDD
  - DDD
  - Programowanie funkcyjne
- Zawodowo programuję od 8 lat w Javie
- M2M & CC
  - Java / Ruby
  - DevOps

Architektura Mikroserwisów

Architektura Mikroserwisów





Języki programowania używane do budowy systemu

- Java
- Ruby
- Javascript
- Python
- Objective C

- W projekcie bierze udział około 50 osób
- 7 rozproszonych zespołów pracujących w różnych krajach
- Projekt wykorzystuje metodologię SCRUM

Skalowanie

- Środowiska
  - Dev
    - Chmura prywatna
  - Integration
    - Chmura prywatna
  - QA
    - Chmura publiczna
    - HA
  - Prod
    - Chmura publiczna
    - HA
- Skalowanie systemu
  - Horyzontalne
    - Dodawanie kolejnych serwerów
    - Oprogramowanie musi być przygotowane do tego typu operacji
  - Wertykalne
    - Dodawanie zasobów do serwerów
    - Zmiany w oprogramowaniu nie są wymagane
- HA: High availability
- Continuous deployment

Technologie użyte do budowy systemu

- Java 8
  - Lombok
  - Javaslang
- Spring
  - Framework
  - MVC (Rest)
  - Boot
  - Security
  - Data
  - AMQP
- JPA
  - Hibernate
- Querydsl
- JUnit
- Mockito
- AssertJ
- Client driver
- Cucumber for JVM (BDD)
- Swagger

Zapewnianie jakości

- Testy jednostkowe
  - JUnit + Mockito
  - Testy pojedynczych klas
- Testy integracyjne
  - Cucumber
  - Uruchamiają kontekst Springowy
  - Korzystają z docelowej bazy danych
  - Używają kolejki
- Testy systemowe
  - Cucumber
  - Działają na środowisku developerskim

Zapewnianie jakości, promocja wersji oprogramowania

- Dostępne środowiska
  - Developerskie
    - Przeznaczone głównie dla developerów
    - Służy do testowania współdziałania serwisów między sobą
    - Wykorzystywany do uruchamiania testów systemowych
    - Dostępne dla działu QA, który weryfikuje działanie nowych funkcji
  - QA
    - Dostępne dla testerów
    - Odbywają się na nim testy regresji
    - Nie odbywa się na nim development nowych funkcji. **Dostarczane są jedynie poprawki błędów**
  - Produkcyjne
    - Praktycznie niedostępne dla developerów
    - Zarządzane przez dział Operations

Diagnostyka systemu

- Monitoring
  - Działa w czasie rzeczywistym
  - Monitoruje stan maszyn wirtualnych (CPU, HDD, itd.)
  - Wykrywa niedostępność usług
  - Wykrywa niedziałające funkcjonalności
  - Odpowiedzialny za wysyłanie alarmów
- KPI (ang. Key Performance Indicators, Kluczowe wskaźniki efektywności)
  - Monitorują w czasie rzeczywistym procesy biznesowe
    - Ilość zakupionych pakietów w ciągu ostatniej godziny.
  - Pozwalają wykrywać anomalie
- Logi z systemów
  - Zawierają informacje na temat błędów
  - Pozwalają odnaleźć przyczynę błędów
  - Na ich podstawie można wyliczać KPIe

Nowoczesne biblioteki służące do logowana informacji

Logback

Dlaczego warto używać Logger zamiast sysoutów

O wyższości Loggera nad System.out.println

Zapis informacji diagnostycznych: Logger vs. System.out.println

Logger

LOGGER.info(”Activation code was successfully sent to the phone number {}”, msisdn);

System.out.println

System.out.println(”Activation code was successfully sent to the phone number ” + msisdn);

Zapis informacji diagnostycznych: Logger vs. System.out.println

- Do not use System.out.println in server side code
  - http://stackoverflow.com/questions/8601831/do-not-use-system-out-println-in-server-side-code
- Logger vs. System.out.println
  - http://stackoverflow.com/questions/2750346/logger-vs-system-out-println
- Why is System.out.println bad?
  - http://stackoverflow.com/questions/19656387/why-is-system-out-println-bad
- Why is using System.out.println() so bad?
  - http://softwareengineering.stackexchange.com/questions/161194/why-is-using-system-out-println-so-bad

Zapis informacji diagnostycznych: Logger vs. System.out.println

- System.out.println
  - Jest operacją WE/WY
    - Są wolne
    - Proces musi czekać na zakończenie operacji
  - Nie przypisujemy poziomu ważności do poszczególnych komunikatów
  - Brak konfiguracji
    - Czy zawsze i ze wszystkich modułów oprogramowania potrzebujemy szczegółowe informacje diagnostyczne na systemie produkcyjnym?
    - Szum informacyjny
- Logger
  - Jest konfigurowalny
    - Możemy dzięki temu uzyskać odpowiednią wydajność
    - Zapisywane są jedynie komunikaty dla nas istotne
  - Zapis komunikatu może zostać oddelegowany do dedykowanego wątku
    - Wątek zapisujący komunikat oczekuje jedynie na wstawienie danych do bufora

Podstawowe pojęcia związane z bibliotekami do zapisu informacji diagnostycznych

- Level
  - Poziom logowania. Określa jak istotny jest komunikat
- Logger/Kategoria
  - Służy do zapisu komunikatów diagnostycznych na pewnym poziomie
  - Jest powiązany z konkretną klasą, która używa go do zapisu komunikatów diagnostycznych
    - Loggery tworzą hierarchię odwzorowującą sposób pakietowania programu
- Appender/Handler
  - Powiązany z loggerami
  - Służy do zapisu informacji diagnostycznych
  - Dostarcza abstrakcji miejsca w którym są składowane komunikaty
- Konfiguracja

Określanie ważności poszczególnych komunikatów diagnostycznych

Poziomy logowania

Poziomy logowania

JDK (java.util.logging.Level)

- SEVERE
- WARNING
- INFO
- CONFIG
- FINE
- FINER
- FINEST

Logback (ch.qos.logback.classic.Level)

- ERROR
- WARN
- INFO
- DEBUG
- TRACE

Poziomy logowania

- Do każdego komunikatu przypisany jest pewien poziom logowania
- Dla każdego loggera również jest przypisywany pewien poziom
  - Poprzez konfigurację
- Komunikat zostanie zapisany tylko wówczas gdy jego poziom jest wyższy lub równy od poziomu przypisanego do loggera, który został użyty do zapisania komunikatu

Poziom komunikatu	**Poziom przypisany do loggera**				
	TRACE	*DEBUG*	*INFO*	*WARN*	*ERROR*
TRACE	Tak	Nie	Nie	Nie	Nie
DEBUG	Tak	Tak	Nie	Nie	Nie
INFO	Tak	Tak	Tak	Nie	Nie
WARN	Tak	Tak	Tak	Tak	Nie
ERROR	Tak	Tak	Tak	Tak	Tak

Co logować na danym poziomie?

- ERROR
  - Wszystkie wyjątki wraz ze stacktrace'ami
  - Wszystkie odpowiedzi HTTP wskazujące na błąd serwera (HTTP 5XX status code)
- WARN
  - Odpowiedzi HTTP wskazujące na błąd klienta (HTTP 4XX status code)
- INFO
  - HTTP żądania i odpowiedzi HTTP
  - Otrzymywane wiadomości AMQP
  - Wszystkie zdarzenia istotne z punktu widzenia logiki biznesowej aplikacji
  - Podsumowanie po wykonaniu zadania przez scheduler
- DEBUG
  - Zapytania bazodanowe
  - Wszystkie informacje pomocne przy debugowaniu informacji
- TRACE
  - Komunikaty, które w krótkim czasie mogą wielokrotnie pojawić się w pliku logów.

Logger, podstawowy interfejs służący do zapisu komunikatów diagnostycznych

Pobieranie loggera

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
public class AttachmentService {
    private final static Logger LOGGER = LoggerFactory.getLogger(AttachmentService.class);
}
```

```
import java.util.logging.Logger;
public class AttachmentService {
    private final static Logger LOGGER = Logger.getLogger(AttachmentService.class.getName());
}
```

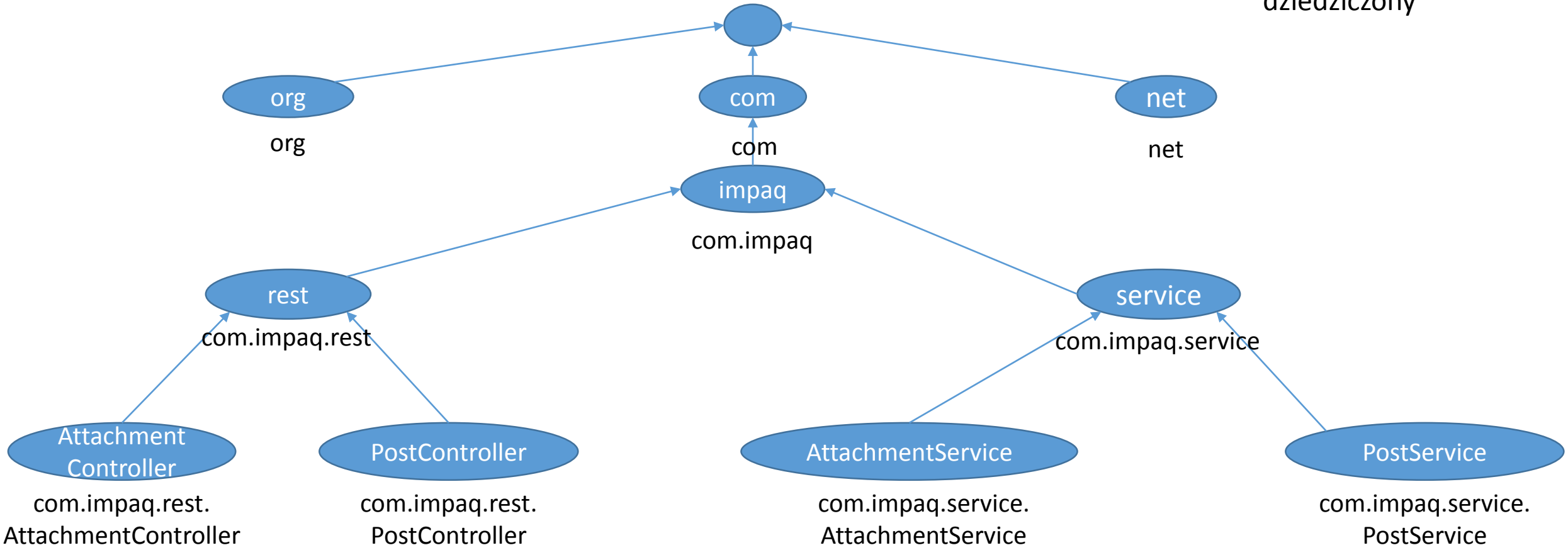
- Wywołanie metody *getLogger* z tym samym argumentem zawsze zwraca ten sam obiekt loggera

Hierarchia loggerów

Logger jest przodkiem innego loggera jeśli jego nazwa po dodaniu kropki jest prefiksem nazwy loggera potomnego. Jeśli dany logger jest bezpośrednim przodkiem loggera potomnego wówczas nazywamy go ojcem, a potomka dzieckiem.

Hierarchia loggerów

Poziom przypisany do loggerów jest dziedziczony



Podstawowy interfejs loggera służący do zapisu komunikatów

```
public interface Logger {
  public void trace(String message);
  public void debug(String message);
  public void info(String message);
  public void warn(String message);
  public void error(String message);
}
```

Zapisywanie informacji diagnostycznych

- Interfejs loggera
  - JDK: java.util.logging.Logger
  - Logback: org.slf4j.Logger

```
LOGGER.log(Level.INFO, "New attachment created {0}", new Object[]{attachmentResponseDto});
LOGGER.info("Message without parameters");
LOGGER.info(() -> "New attachment created " + attachmentResponseDto);
```

```
LOGGER.info("New attachment created {}", attachmentResponseDto);
if(LOGGER.isInfoEnabled()) {
        LOGGER.info("New attachment created " + attachmentResponseDto);
}
```

Ukryty narzut wydajnościowy podczas zapisywania komunikatów

- Konkatenacja stringów zachodzi zawsze nawet jeśli komunikat nie zostanie zapisany:

```
LOGGER.debug("Index: " + i + " is equal to " + array[i]);
```

- Konkatenacja stringów zachodzi tylko wówczas gdy komunikat zostanie zapisany

```
LOGGER.debug("Index: {} is equal to {}", i, array[i]);
```

```
If(LOGGER.isDebugEnabled()){
  LOGGER.debug("Index: " + i + " is equal to " + array[i]);
}
```

Zapis komunikatów m.in. do plików

A także w dowolne inne miejsce

Apender/Handler

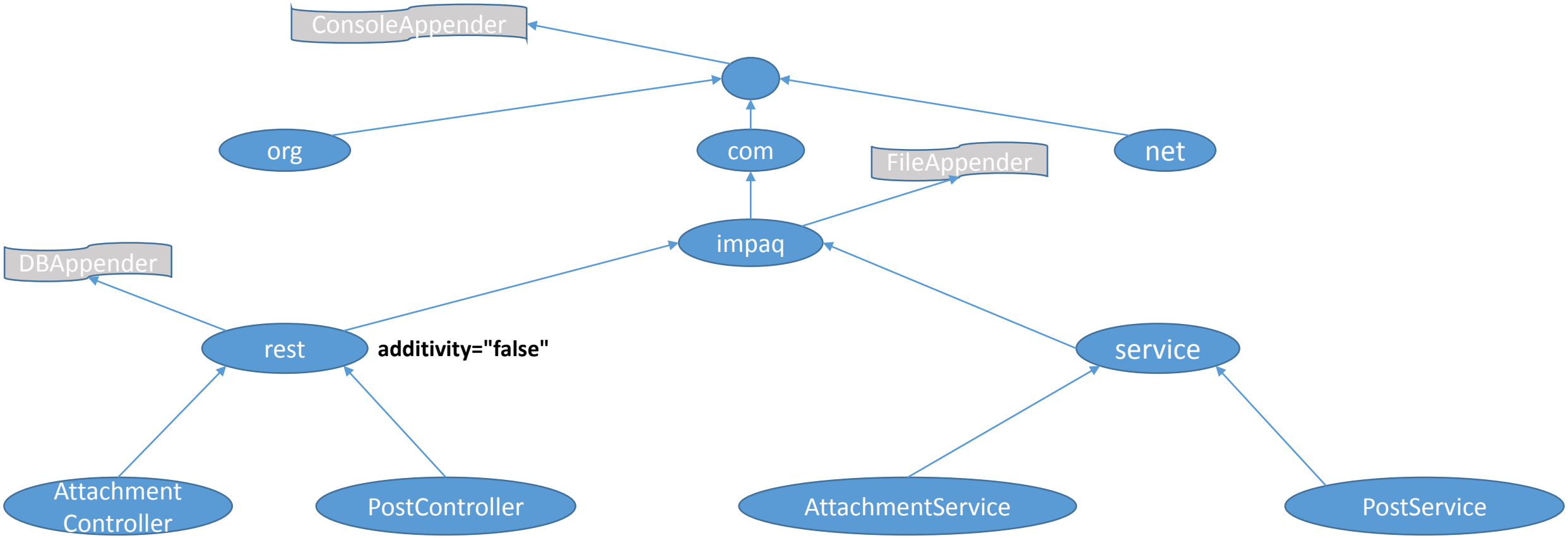
- Jest odpowiedzialny za „zapis” komunikatów
- Dostarcza abstrakcji miejsca, w którym zostaną „zapisywane” komunikaty
- Logger jest powiązany z Appenderem przez nazwę
- Umożliwia filtrowanie komunikatów
- Umożliwia formatowanie komunikatów
  - Za pomocą Layoutów
- Zdefiniowany interfejs appendera: ch.qos.logback.core.Appender
- Dostępne appendery: OutputStreamAppender, ConsoleAppender, FileAppender, **RollingFileAppender**, SocketAppender, SMTPAppender, DBAppender, SyslogAppender, AsyncAppender, itd.

Podstawowy interfejs Appendera

```
public interface Appender<E> extends LifeCycle, ContextAware,
FilterAttachable<E> {
    String getName();
    void doAppend(E event) throws LogbackException;
    void setName(String name);
}
```

Addytywność apenderów

Domyślna wartość parametry additivity to true



Filtry

- Filtr jest zdefiniowany poprzez klasę bazową ch.qos.logback.core.filter.Filter
- Do appendera może zostać przypisanych wiele filtrów
- Filtry przypisane do apendera tworzą uporządkowaną listę
- Bazują na logice ternarnej
- Rezultat działania filtra to enum: ch.qos.logback.core.spi.FilterReply
  - DENY (przetwarzanie komunikatu jest kończone)
  - NEUTRAL (Wywoływany jest następny filtr na liście, jeśli nie ma kolejnego filtra komunikat jest przetwarzany normalnie przez appender)
  - ACCEPT (komunikat jest natychmiast zapisywany. Kolejne filtry nie są wywoływane)

Filtry

- Dostępne implementacje
  - ThresholdFilter
  - LevelFilter
  - EvaluatorFilter
    - Deleguje swoją odpowiedzialność do ch.qos.logback.core.boolex.EventEvaluator

```
public abstract class Filter<E> extends ContextAwareBase implements LifeCycle {
    public abstract FilterReply decide(E event);
}
```

Layout

- Jest przypisywany do apendera
- Odpowiedzialny za przetworzenie komunikatu na string
- Zdefiniowany przez interfejs ch.qos.logback.core.Layout

```
public interface Layout<E> extends ContextAware, LifeCycle {
  String doLayout(E event);
  String getFileHeader();
  String getPresentationHeader();
  String getFileFooter();
  String getPresentationFooter();
  String getContentType();
}
```

Layout

- Dostępne Layouty
  - EchoLayout
  - HTMLLayout
  - **PatternLayout**
  - XMLLayout

PatternLayout

- Elastyczna implementacja layoutu
- Przykładowy pattern:
  %d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} %mdc{user_name} - %msg%n
  - %d – opisuje format daty
  - %thread – nazwa wątku
  - %-5level – poziom wyjustowany do lewej do długości 5 znaków
  - %logger{36} – nazwa loggera składająca się maksymalnie z 32 znaków z ciekawy, algorytmem skracania
  - %mdc{user_name} – pole „user_name” z kontekstu MDC
  - %msg – komunikat
  - %n – znak nowego wiersza
  - Rezultat użycia powyższego patternu:
    21:54:09.349 [http-nio-8080-exec-4] DEBUG org.hibernate.SQL bartek - delete from forum.attachment where id=?

- Dokumentacja: http://logback.qos.ch/manual/layouts.html#conversionWord

System przestał działać!

- Nikt go nie modyfikował od kilku miesięcy
- Do tej pory działał bez zarzutów
- Nikt niczego nie zmieniał
- Co się stało?

Miejsce na dysku twardym jest skończone

- Powoduje to pewne problemy przy zapisie komunikatów diagnostycznych na dysk twardy (FileAppender, OutputStreamAppender)
  - Jeśli system działa wystarczająco długo logi zajmą całe dostępne miejsce na dysku twardym
- RollingFileAppender
  - Odpowiedzialny za „przewijanie plików”
    - SizeBasedTriggeringPolicy
  - Umożliwia skonfigurowanie tego ile i jak długo przetrzymujemy pliki z komunikatami
    - TimeBasedRollingPolicy – Pliki przetrzymujemy przez pewien czas
    - FixedWindowRollingPolicy – Przechowujemy określoną liczbę plików

Algorytm zapisu komunikatów diagnostycznych

- Podejmij decyzję na poziomie loggera m.in. na podstawię poziomów logowania czy komunikat ma zostać zapisany
- Utwórz obiekt zawierający wszystkie informacje o komunikacie: *ch.qos.logback.classic.spi.LoggingEvent*
- Wywołaj apendery (potencjalnie wiele)
  - Wywołaj filtry przypisane do apendera
  - Wykonaj formatowanie komunikatu
  - Zapisz komunikat

Konfiguracja systemu logowania

Konfiguracja

- Podczas inicjalizacji Logback wczytuje konfigurację z:
  - Pliku *logback.groovy* na classpathie
  - Jeśli powyższy plik nie został odnaleziony przetwarzany jest plik *logback-test.xml*
  - Jeśli powyższy plik nie został odnaleziony przetwarzany jest plik *logback.xml*
  - Jeśli powyższy plik nie został odnaleziony używany jest java.util.ServiceLoader aby odnaleźć implementację interfejsu com.qos.logback.classic.spi.Configurator
  - Zostanie wczytana domyślna konfiguracja: *ch.qos.logback.classic.BasicConfigurator*
- Plik konfiguracyjny można przekazać jako properties *logback.configurationFile*

```
java -Dlogback.configurationFile=/path/to/config.xml chapters.configuration.MyApp1
```

Prosta konfiguracja w formacie pliku XML

```
<configuration>

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

  <root level="debug">
    <appender-ref ref="STDOUT" />
  </root>
</configuration>
```

Źródło: logback-examples/src/main/resources/chapters/configuration/sample0.xml

Prosta konfiguracja w formacie pliku XML

```
<configuration>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} %mdc{user_name} - %msg%n</pattern>
    </encoder>
  </appender>
  <logger name="org.hibernate.SQL" level="DEBUG" />
  <logger name="org.hibernate.type" level="WARN" />
  <logger name="org.springframework.web.filter.CommonsRequestLoggingFilter" level="DEBUG" />
  <logger name="com.impaqgroup.training.logging" level="DEBUG" />
  <root level="WARN">
    <appender-ref ref="STDOUT" />
  </root>
</configuration>
```

Prosta konfiguracja w formacie pliku XML

```
<configuration>
  <property name="LOG_FILE" value="/var/log/forum/forum.log" />
  <appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <encoder>
      <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} %mdc{user_name} - %msg%n</pattern>
    </encoder>
    <file>${LOG_FILE}</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
      <fileNamePattern>${LOG_FILE}.%i</fileNamePattern>
    </rollingPolicy>
    <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
      <MaxFileSize>10MB</MaxFileSize>
    </triggeringPolicy>
  </appender>
  <logger name="com.impaqgroup.training.logging" level="DEBUG" />
  <root level="WARN">
    <appender-ref ref="FILE" />
  </root>
</configuration>
```

Zmiana konfiguracji w trakcie działania programu

```
<configuration scan="true" scanPeriod="30 seconds" >
  ...
</configuration>
```

- Za przeładowywanie konfiguracji odpowiedzialny jest filtr ch.qos.logback.classic.turbo.ReconfigureOnChangeFilter
- Jeśli plik konfiguracyjny zostanie zmieniony Logback uaktualni swoją konfigurację gdy:
  - Upłynie czas wyspecyfikowany w parametrze *scanPeriod*
  - Nastąpi kilka, bądź kilkanaście wywołań metod służących do zapisu komunikatów
- Uwaga, podczas rekonfiguracji mogą zostać utracone komunikaty

Zmienne w pliku konfiguracyjnym

- Predefiniowane zmienne:
  - HOSTNAME
  - System property wirtualnej maszyny Javy
- Definicja zmienne za pomocą elementu *property:*
  <property name="USER_HOME" value="/home/sebastien" />
- Wczytanie zmiennych z pliku
  - Obecnego w systemie pliku:
    <property file="src/main/java/chapters/configuration/variables1.properties" />
  - Na classpathie:
    <property resource="resource1.properties" />
- Odwołanie się do zmiennych następuje przy pomocy wrażenia:
  ${NAZWA_ZMIENNEJ}

Logback

- Składa się z dwóch modułów
  - Access
  - Core
- Logback classic do działania wymaga biblioteki SLF4J, która definiuje interfejs jego loggerów

```
<dependency>
   <groupId>ch.qos.logback</groupId>
   <artifactId>logback-classic</artifactId>
   <version>1.1.7</version>
</dependency>
```

```
<dependency>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-api</artifactId>
   <version>1.7.21</version>
</dependency>
```

Zadanie 1: Uruchomienie aplikacji

- Uruchom maszynę wirtualną
  - Użytkownik: user
  - Hasło: alamakota
- Uruchom IDE
  - ./idea.sh
- Uruchom przykładowy program
  Główna klasa: *com.impaqgroup.training.logging.Application*
- W plikach projektu znajdź konfigurację biblioteki Logback
- Znajdź konfigurację aplikacji
  - Pliki application.properties, application-derby.properties, application-postgresql.properties

Wskazówka, git

- Pobranie nowej wersji z repozytorium:
  git pull
- Sprawdzenie statusu
  git status
- Sprawdzenie roboczej gałęzi:
  git branch
- Zmiana gałęzi:
  git checkout master
  git checkout training-start
- Dodanie zmian
  git add .
- Zatwierdzenie zmian
  git commit -m "My description"

Lokalizacja przykładowego programu:
~/loging-training

Zadanie 2: Konfiguracja Logbacka

- Ustaw poziom głównego loggera na WARN
- Ustaw poziom logowania dla loggerów:

  org.hibernate.SQL: DEBUG
  org.hibernate.type: WARN
  com.impaqgroup.training.logging: DEBUG
- Dodaj *ConsoleAppender* o nazwie STDOUT
- Przypisz do appender'a PatternLayout z formatem:
  %d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n
- Przypisz apender o nazwie STDOUT do root loggera
  - W ten sposób wszystkie loggery „odziedziczą" appender STDOUT

Zadanie 2: Zapis komunikatów diagnostycznych

- Dodaj według uznania komunikaty diagnostyczne w każdej metodzie klas:
  - com.impaqgroup.training.logging.service.AttachmentService
  - com.impaqgroup.training.logging.service.PostService
- Sprawdź czy dodane komunikaty pojawiają się na konsoli
  - Poprzez ręczne wysłanie kilku requestów RESTowych do aplikacji (wskazówki na następnym slajdzie
  - Przez uruchomienie testuów (uwaga, testy nigdy się nie kończą):
    - com.impaqgroup.training.logging.rest.PostControllerIT#stressTest
    - com.impaqgroup.training.logging.rest.AttachmentControllerIT#stressTest

Wskazówka: Operacje CRUD na poście

- Pobranie wszystkich postów:
  curl -i -XGET -u ala:makota localhost:8080/post
- Dodanie posta
  curl -i -XPOST -u ala:makota -H "content-type:application/json" localhost:8080/post -d '{"title":"Question", "content":"I need asnwer!"}
- Pobranie posta o id 1:
  curl -i -XGET -u ala:makota localhost:8080/post/1
- Aktualizacja posta o id 1:
  curl -i -XPUT -u ala:makota -H "content-type:application/json" localhost:8080/post/1 -d '{"title":"Question", "content":"I need asnwer right now!"}
- Usunięcie posta
  curl -i -XDELETE -u ala:makota localhost:8080/post/1

Wskazówka: Operacje CRUD załączniki

- Pobranie wszystkich załączników
  curl -i -XGET -u ala:makota localhost:8080/attachment
- Utworzenie załącznika
  curl -i -XPOST -u ala:makota -H "content-type:application/json" localhost:8080/attachment -d '{"name":"screenshot", "content":"YWxh"}
- Pobranie załącznika o identyfikatorze 1:
  curl -i -XGET -u ala:makota localhost:8080/attachment/1
- Aktualizacja załącznika o identyfikatorze 1:
  curl -i -XPUT -u ala:makota -H "content-type:application/json" localhost:8080/attachment/1 -d '{"name":"image", "content":"YWxh"}
- Usunięcie załącznika o identyfikatorze 1:
  curl -i -XDELETE -u ala:makota localhost:8080/attachment/1

Zadanie 3: Zapis komunikatów diagnostycznych do pliku

- Dodaj do konfiguracji Logbacka FileAppender o nazwie FILE
- Użyj PatternLayoutu z formatowaniem:
  %d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n
- Plik tworzony przez apender powinien nazywać się forum.log i znajdować się w katalogu domowym
- Dołącz apender do głównego loggera

Zadanie 4: Logowanie żądań HTTP

- Zarejestruj filtr *CommonsRequestLoggingFilter* w konfguracji *com.impaqgroup.training.logging.Application*
- Skonfiguruj filtr *CommonsRequestLoggingFilter* aby zapisywał:
  - Informacje o kliencie
  - Query String
  - Payload
- Filtr powinien zostać zarejestrowany w kolejności zdefiniowanej w stałej *FILTER_ORDER_COMMONS_REQUEST_LOGGING_FILTER*
- Ustaw poziom logowania z loggera *org.springframework.web.filter.CommonsRequestLoggingFilter* na DEBUG
- Sprawdź czy w logu aplikacji są zapisywane żądania HTTP

Dostępne biblioteki do zapisu komunikatów.

I problemy powodowane przez ich „dużą” liczbę.

Biblioteki służące do zapisu informacji diagnostycznych

- Log4j
  - 1999-10-15
  - 2000-11-20 – v. 0.9.0, pierwszy wersja udostępniona publicznie
  - 2001-01-08 – v. 1.0
  - 2015-08-05 – koniec życia projektu
- JUL (java.util.logging)
  - 2002-06-02
- Logback
  - 2006-07-20
  - 2011-11-01 – v. 1.0.0
- Log4j2
  - 2012-07-29
  - 2014-07-12 – v. 2.0 GA

Której biblioteki do logowanie użyć?

- W moim projekcie używam następujących bibliotek:
  - ORM, która wykorzystuje Log4j 1.2
  - Depenency Injection, która używa Logbacka
- Czy to znaczy, że muszę dostarczyć pliki konfiguracyjne dla
  - Log4j 1.2
  - Logbacka
- Oraz każda z tych bibliotek będzie zapisywała dane do innego pliku logu?

Biblioteki dostarczające abstrakcji systemy logowania

- Najpopularniejsze dostępne biblioteki:
  - JCL - Apache Common Logging
  - SLF4J - Simple Logging Facade for Java
- Biblioteki te umożliwiają wybór docelowego systemu logowania podczas wdrożenia
- Wybór systemu logowania odbywa poprzez dodanie odpowiednich bibliotek do classpathu
- Biblioteki te dostarczają:
  - API, przeznaczone do zapisu komunikatów (zawiera m.in. *org.slf4j.Logger)*
  - Wiele implementacji API, służące do zapisu komunikatów przy pomocy konkretnej biblioteki (Log4j, Logback, JUL)

SLF4J i dostarczane implementacje

- Implementacje loggera *org.slf4j.Logger* pozwalające delegować wywołania do następujących bibliotek:
  - Log4j (slf4j-log4j12-1.7.21.jar)
  - JUL (slf4j-jdk14-1.7.21.jar)
  - JCL (slf4j-jcl-1.7.21.jar)
- Inne implementacje loggera *org.slf4j.Logger*
  - NOP (slf4j-nop-1.7.21.jar) – nie zapisuje komunikatów :o
  - Simple (slf4j-simple-1.7.21.jar) – zapisuje komunikaty do System.err
  - Logback (logback-classic-1.0.13.jar)
    - Logger *ch.qos.logback.classic.Logger* implementuje *org.slf4j.Logger*

Biblioteki zależące bezpośrednio od JUL i Log4j

- Jeśli biblioteki, których chcemy użyć logują komunikaty diagnostyczne bezpośrednio za pomocą loggerów:
  - JUL: java.util.logging.Logger
  - Log4j: org.apache.log4j.Logger
- Czy możemy je przekierować do SLF4J, a tym samym do dowolnego innej biblioteki do logowania komunikatów diagnostycznych ???

Przekierowanie Log4j i JUL do SLF4J

- log4j-over-slf4j
  - Dostarcza własnej implementacji klas z biblioteki Log4j, która deleguje zapis komunikatów do SLF4J
  - „Oryginalną” bibliotekę Log4J należy usunąć z classpatha
- jul-to-slf4j
  - JUL deleguje proces tworzenia loggerów do metody: *java.util.logging.LogManager#demandLogger*
  - LogManager jest singletonem, referencja do niego jest przechowywana w polu: *java.util.logging.LogManager#manager*
  - Podczas tworzenia LogManagera odbywa się odczyt własności systemowej **java.util.logging.manager**
    - Jeśli jest równa null zostanie utworzony domyślny LogManager java.util.logging.LogManager
    - Jeśli jest różna od null wówczas zamiast domyślnego LogManagera zostanie utworzony za pomocą refleksji obiekt, którego nazwa klasy była zapisana we własności systemowej **java.util.logging.manager**

Zaawansowane funkcje

Kontekst MDC

Kontekst MDC

- Umożliwia dodanie informacji kontekstowych do serii logowanych komunikatów
- Dane zapisane w MDC są dostępne podczas zapisu każdego komunikatu
  - Od momentu wstawiania danych do kontekstu MDC
  - Do momentu usunięcia danych z kontekstu MDC
- Dane, które zostaną zapisane do pliku logu z kontekstu MDC wyznacza Pattern przypisany do Appendera
- Dane najczęściej dodawane do kontekstu
  - Nazwa lub identyfikator użytkownika
  - Ścieżka przetwarzanego requestu HTTP
  - Dane pozwalające skorelować komunikat z innymi
  - Nazwa procesu biznesowego

Interfejs kontekstu MDC

- Definiowany przez klasę: *org.slf4j.MDC*
- Dane w kontekście MDC są przypisane do wątku (ThreadLocal)

```
public class MDC {
    public static void put(String key, String val) throws IllegalArgumentException {
        //...
    }
    public static String get(String key) throws IllegalArgumentException {
        //...
    }
    public static void remove(String key) throws IllegalArgumentException {
        //...
    }
    public static void clear() {
        //...
    }
    public static Map<String, String> getCopyOfContextMap() {
        //...
    }
}
```

Zapis danych z kontekstu MDC

```
 <appender name="STDOUT"
class="ch.qos.logback.core.ConsoleAppender">
      <encoder>
        <pattern>%-5level user:%mdc{user_name} - %msg%n</pattern>
      </encoder>
   </appender>
```

Przykład użycia kontekstu MDC

```
@Test
public void mdcContextExample(){
    LOGGER.info("Message without any data in context");
    MDC.put("user_name", "Budzigniew");
    LOGGER.debug("Data added to MDC context");
    LOGGER.info("Data still present in context");
    LOGGER.warn("Data will be removed from MDC context");
    MDC.remove("user_name");
    LOGGER.info("Data removed from context");
}
```

Przykład wyjścia zawierającego dane z kontekstu MDC

INFO user: - Message without any data in context

DEBUG user:Budzigniew - Data added to MDC context

INFO user:Budzigniew - Data still present in context

WARN user:Budzigniew - Data will be removed from MDC context

INFO user: - Data removed from context

TurboFiltry

TurboFiltry

- Są przypisane bezpośrednio do kontekstu logowania (a nie do apendera)
- Są wywoływane w momencie gdy logger ustala (na podstawie Leveli) decyzję czy dany komunikat ma być zapisany
- Pozwalają wpłynąć na standardowe zachowanie loggera. Mogą zostać użyte do:
  - wymuszenia przetworzenia komunikatu przez Logger
  - Do zaniechania dalszego przetwarzania komunikatu przez Logger
- Bazują na logice ternarnej (DENY, NEUTRAL, ACCEPT)
- Są wywoływane w kolejności ich zdefiniowania w kontekście
- Rozszerzają klasę: ch.qos.logback.classic.turbo.TurboFilter

Przykładowy TurboFiltr

```
public class NiceNameTurboFilter extends TurboFilter {

    private String name;

    public FilterReply decide(Marker m, Logger l, Level le, String format, Object[] params, Throwable t) {
        boolean nice = isNice(format, params == null ? emptySet() : new HashSet<>(asList(params)));
        return nice ? ACCEPT :  NEUTRAL;
    }

    private boolean isNice(String format, Set<Object> params) {
        return format.contains(name) || params.contains(name);
    }

    public void setName(String name) {this.name = name;}
}
```

Standardowe TurboFiltry

- DuplicateMessageFilter
- DynamicThresholdFilter
- MDCFilter
- MarkerFilter

Konfiguracja TurboFiltrów

```
<configuration>
  <turboFilter class="com.impaqgroup.training.logging.turbo.NiceNameTurboFilter">
    <name>Łukasz</name>
  </turboFilter>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <encoder>
      <pattern>%-5level user:%mdc{user_name} - %msg%n</pattern>
    </encoder>
  </appender>
  <logger name="com.impaqgroup" level="DEBUG" />
  <root level="WARN"><appender-ref ref="STDOUT" /></root>
</configuration>
```

Test TurboFiltra

```
@Test
public void turboFilterTest(){
    LOGGER.trace("{} is a very nice name", "Dobromysław");
    LOGGER.trace("{} is a very nice name", "Łukasz");
    LOGGER.trace("{} is a very nice name", "Kociebor");
}
```

Rezultat:
TRACE user: - Łukasz is a very nice name

Markery

Markery

- Definiowany przez klasę *org.slf4j.Marker*
- Tworzone za pomocą metody fabrykującej: *org.slf4j.MarkerFactory#getMarker(String)*
- Każda z metod służących do zapisu komunikatów w interfejsie loggera przyjmuje parametry typu Marker
- Wraz z TurboFiltrami służą do tworzenia zaawansowanych reguł przeznaczonych do filtrowania komunikatów

Interfejs Markera

```
public interface Marker extends Serializable {
    public String getName();
    public void add(Marker reference);
    public boolean remove(Marker reference);
    public boolean hasReferences();
    public Iterator<Marker> iterator();
    public boolean contains(Marker other);
    public boolean contains(String name);
}
```

Asynchroniczny zapis komunikatów

AsyncAppender

- Komunikaty są zapisywane asynchronicznie
  - Przez dedykowany do tego celu wątek
  - Przy wykorzystaniu niewielkiego bufora na komunikaty
- Wątek wywołujący logger nie oczekuje na zapis komunikatu
- Duże ilości danych są przetwarzane znacznie efektywnie
- Umożliwia „gubienie mało istotnych komunikatów gdy bufor jest pełny
- http://blog.takipi.com/how-to-instantly-improve-your-java-logging-with-7-logback-tweaks/

AsyncAppender wydajność

	Discarding Threshold	Queue size	Lines / Minute avg.	% of top performer
AsyncAppender	0	500	6,132,908	100%
AsyncAppender	0	1,000,000	5,909,417	94%
FileAppender	-	-	1,644,635	27%
AsyncAppender (default)	0.2	256	1,249,118	20%

Źróło: http://blog.takipi.com/how-to-instantly-improve-your-java-logging-with-7-logback-tweaks/

AsyncAppender – przykładowa konfiguracja

```
<appender name="STDOUTSync" class="ch.qos.logback.core.ConsoleAppender">
   <encoder>
         <pattern>%-5level user:%mdc{user_name} - %msg%n</pattern>
   </encoder>
</appender>
<appender name="STDOUT" class="ch.qos.logback.classic.AsyncAppender">
   <queueSize>500</queueSize>
   <discardingThreshold>0</discardingThreshold>
   <appender-ref ref="STDOUTSync" />
</appender>
```

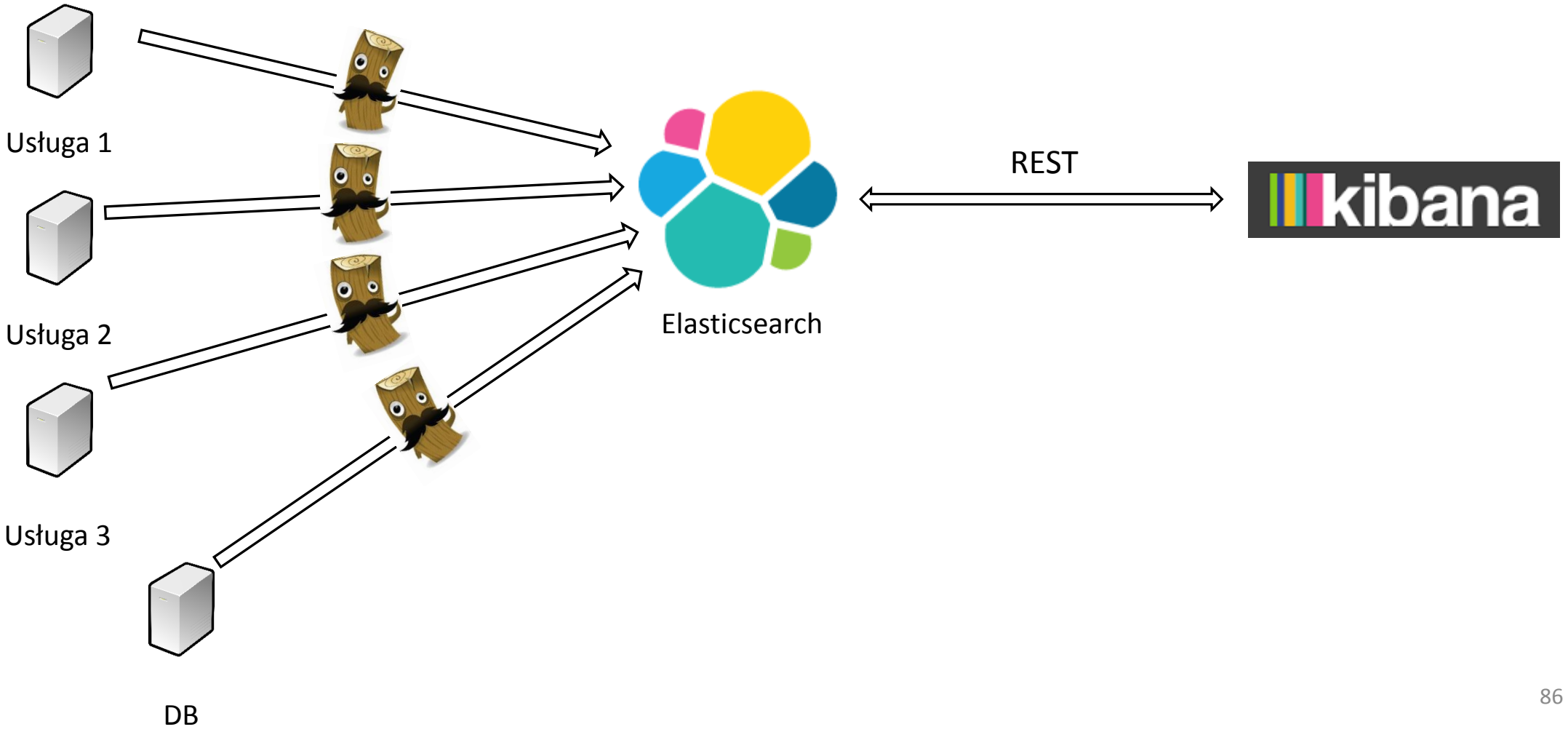
Logowanie w systemie rozproszonym

Elasticsearch Logstash Kibana

System rozproszony i logowanie komunikatów

- Podejście tradycyjne
  - Każda usługa zapisuje komunikaty do lokalnego pliku logu
  - Komunikaty można przeglądać dzięki narzędziom takim jak ssh i grep
  - Podejście to nie sprawdza się w systemach rozproszonych z maszyn wirtualnych/kontenerów
- Podejście wygodne
  - Wszystkie logi składujemy w jednym centralnym miejscu (Elasticsearch)
  - Logi z wszystkich maszyn wirtualnych/kontenerów są przesyłane do jednej centralnej bazy danych (Za pomocą Logstash’a)
  - Logi można efektywnie przeszukiwać za pomocą dedykowanej do tego aplikacji (Kibana)

System rozproszony i logowanie komunikatów



Logstash

Logstash

- Wczytuje dane za pomocą pluginu wejściowego
- Przekształca dane
  - Filtry
- Wysyła dane za pomocą pluginu wyjściowego

- Pluginy wejściowe/wyjściowe:
  - file
  - rabbitmq
  - elasticsearch
  - syslog
  - redis

Plik konfiguracyjny Logstasha

- Uruchomienie Logstasha: *logstash -f logstash.conf*
- Dokumentacja: https://www.elastic.co/guide/en/logstash/current/index.html

- Plik konfiguracyjny Logstasha
  - Zawiera (wiele) sekcji input
  - Zawiera (wiele) sekcji output

```
input {
	stdin {
	}
}

output {
	stdout {
	}
}
```

Plik konfiguracyjny Logstasha przykład

```
input {
   file {
         path =>  [ "/var/log/forum/forum.json" ]
         codec => json {}
         add_field => {
            "source" => "logging-training"
            "tenant" => "dev"
         }
      }
}

output {
   elasticsearch {
      index => "logging-training-%{+YYYY.MM.dd}"
   }
}
```

Zadanie 5: Plik logu w formacie JSON

- Dodaj do konfiguracji Logbacka FileAppender o nazwie JSON_FILE
- Dodaj do konfiguracji apendera enkoder: *net.logstash.logback.encoder.LogstashEncoder*
- Plik tworzony przez apender powinien nazywać się forum.json i znajdować się w katalogu domowym
- Dołącz apender do głównego loggera

Elasticsearch

Elasticsearch

- Rozproszona dokumentowa baza danych (NoSQL)
  - Dokumenty są reprezentowane przy użyciu notacji JSON
- Pełnotekstowy silnik wyszukiwania danych
  - Zbudowany w oparciu o Apache Lucene
- Wspiera bezproblemowe klastrowanie
- Komunikacja z Elasticsearchem odbywa się za pomocą REST API
- Domyślny port: 9200

Struktura bazy danych

http://elasticsearch/index/type/documentId

- Index – baza danych
- Typ – tabela
- Dokument – rekord
  - Dokument wewnątrz danego indeksu o danym typie jest identyfikowany przez documentId

Identyfikator dokumentu

- Dokument jest jednoznacznie identyfikowany przez:
  - Indeks
  - Typ
  - Identyfikator

Indeksy

- Pobranie listy indeksów:
  curl -i elasticsearch:9200/_cat/indices
- Tworzenie indeksu:
  curl -XPUT -i elasticsearch:9200/my-index-name
- Elasticsearch utworzy indeks automatycznie w momencie gdy dodamy do niego pierwszy dokument
- Usuwanie indeksu
  curl -XDELETE http://elasticsearch:9200/my-index-name

Operacje na dokumentach

- Tworzenie z wyspecyfikowanym identyfikatorem
  curl -i -XPOST elasticsearch:9200/my-app-index/users/007 -d '{"firstname":"James","surname":"Bond"} '
- Tworzenie dokumentu z automatycznie wygenerowanym identyfikatorem
  curl -i -XPOST elasticsearch:9200/my-app-index/users -d '{"firstname":"Marian","surname":"Zacharski"},
- Pobieranie dokumentu po identyfikatorze
  curl -i -XGET elasticsearch:9200/my-app-index/users/007

Operacje na dokumentach

- Sprawdzanie czy dokument istnieje:
  curl -i -XHEAD elasticsearch:9200/my-app-index/users/007
- Aktualizacja dokumentu:
  curl -i -XPUT elasticsearch:9200/my-app-index/users/007 -d '{"firstname":"Jamie","surname":"Bond"}
- Aktualizacja części dokumentu:
- *curl -i -XPOST elasticsearch:9200/my-app-index/users/007/_update -d '{"doc":{"firstname":"Jamie"}}'*
- Usuwanie dokumentu:
  curl -i -XDELETE elasticsearch:9200/my-app-index/users/007

Typy danych

- string
- byte, short, integer, long
- float, double
- boolean
- date

- Elasticsearch próbuje odgadnąć prawidłowy typ danych podczas wstawiania pierwszego dokumentu zawierającego dane pole
- Pobranie typów danych związanych z typem:
  curl -XGET http://elasticsearch:9200/my-app-index/_mapping/users

Indeksowanie stringów

Elasticsearch może indeksować stringi na następujące sposoby:

- no
  Pole nie jest indeksowane. Pole nie będzie mogło zostać użyte w kryteriach wyszukiwania
- not_analyzed
  Dosłowna wartość pola może zostać użyta w kryteriach wyszukiwania
- analyzed
  Pole jest najpierw analizowane po czym indeksowanie. Pełno tekstowe wyszukiwanie jest dostępne na wartościach pola

Test analizy stringów można wykonać za pomocą:
curl -i "http://elasticsearch:9200/_analyze?analyzer=english" -d 'Text to test'

Wyszukiwanie: Query String Query

- curl http://elasticsearch:9200/index/type/_search?q=text:java
- curl http://elasticsearch:9200/index/type/_search?q=lang:(-en)
- curl http://elasticsearch:9200/index/type/_search?q=text:(java javascript ruby)
- curl http://elasticsearch:9200/index/type/_search?q=user.name:(Javascript Digest)
- curl http://elasticsearch:9200/index/type/_search?q=+text:performance
- curl http://elasticsearch:9200/index/type/_search?q=-text:performance
- curl http://elasticsearch:9200/index/type/_search?q=user.friends_count:>2000
- curl http://elasticsearch:9200/index/type/_search?q=text:(java AND javascript AND ruby AND php)
- curl http://elasticsearch:9200/index/type/_search?q=lang:pl AND text:paypal
- curl http://elasticsearch:9200/index/type/_search?q=_missing_:in_reply_to_status_id
- curl http://elasticsearch:9200/index/type/_search?q=_exists_:in_reply_to_status_id
- curl http://elasticsearch:9200/index/type/_search?q=user.friends_count:>=100 AND user.friends_count:<=200
- curl http://elasticsearch:9200/index/type/_search?q=user.friends_count:[100 TO 201}

Query String Query

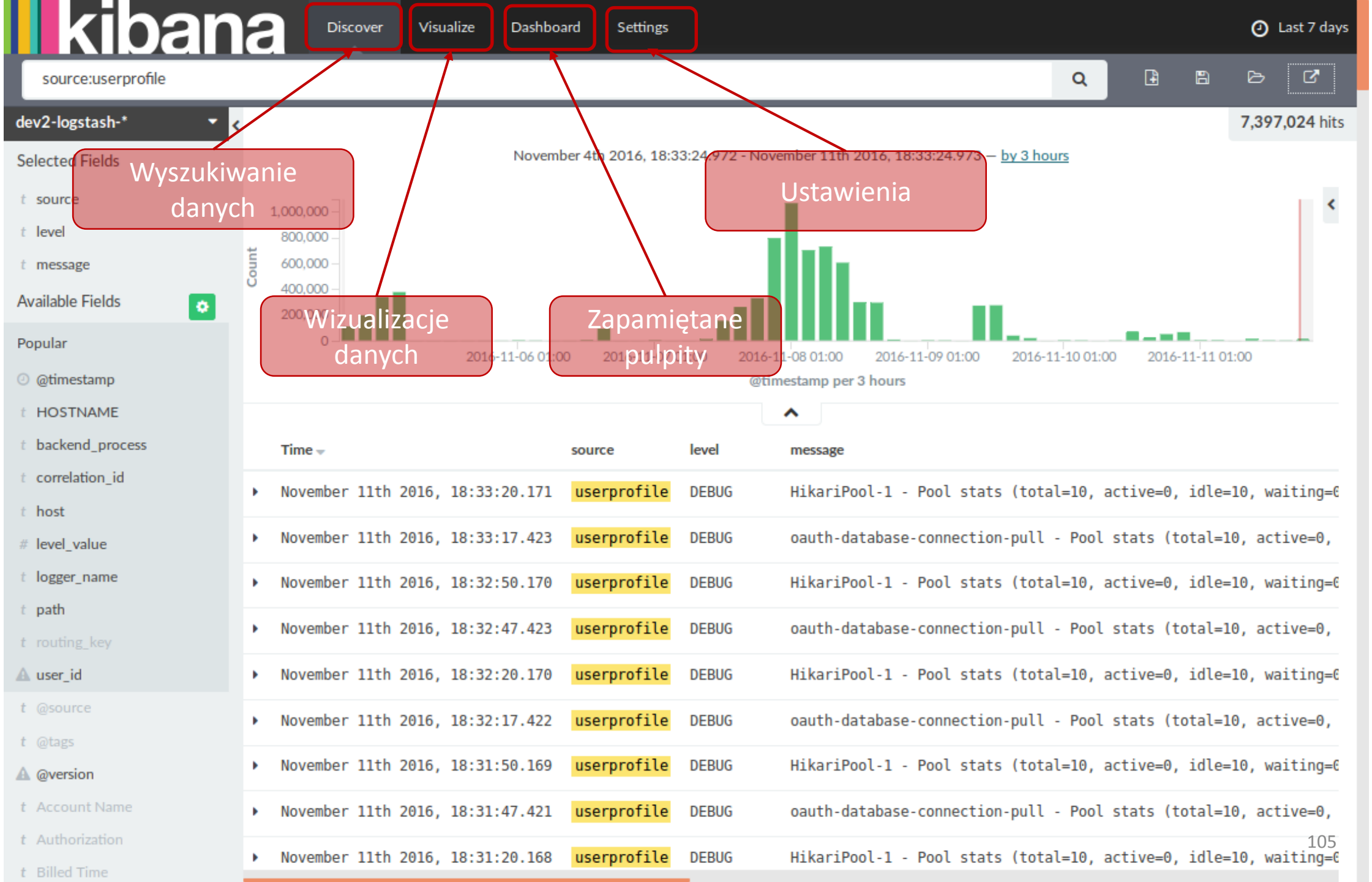
Dokumentacja:

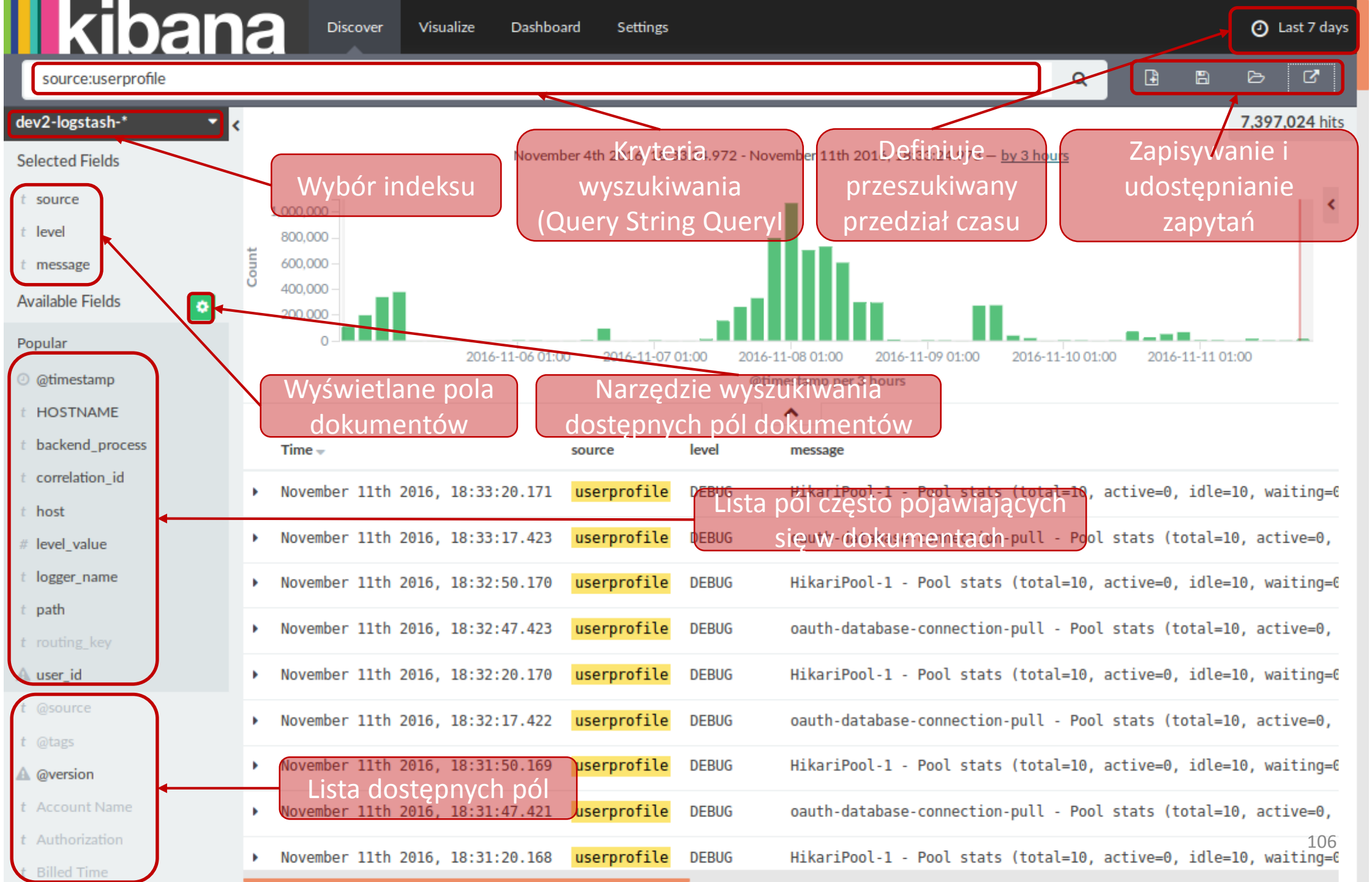
- https://www.elastic.co/guide/en/elasticsearch/reference/current/search-uri-request.html
- https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-query-string-query.html#query-dsl-query-string-query

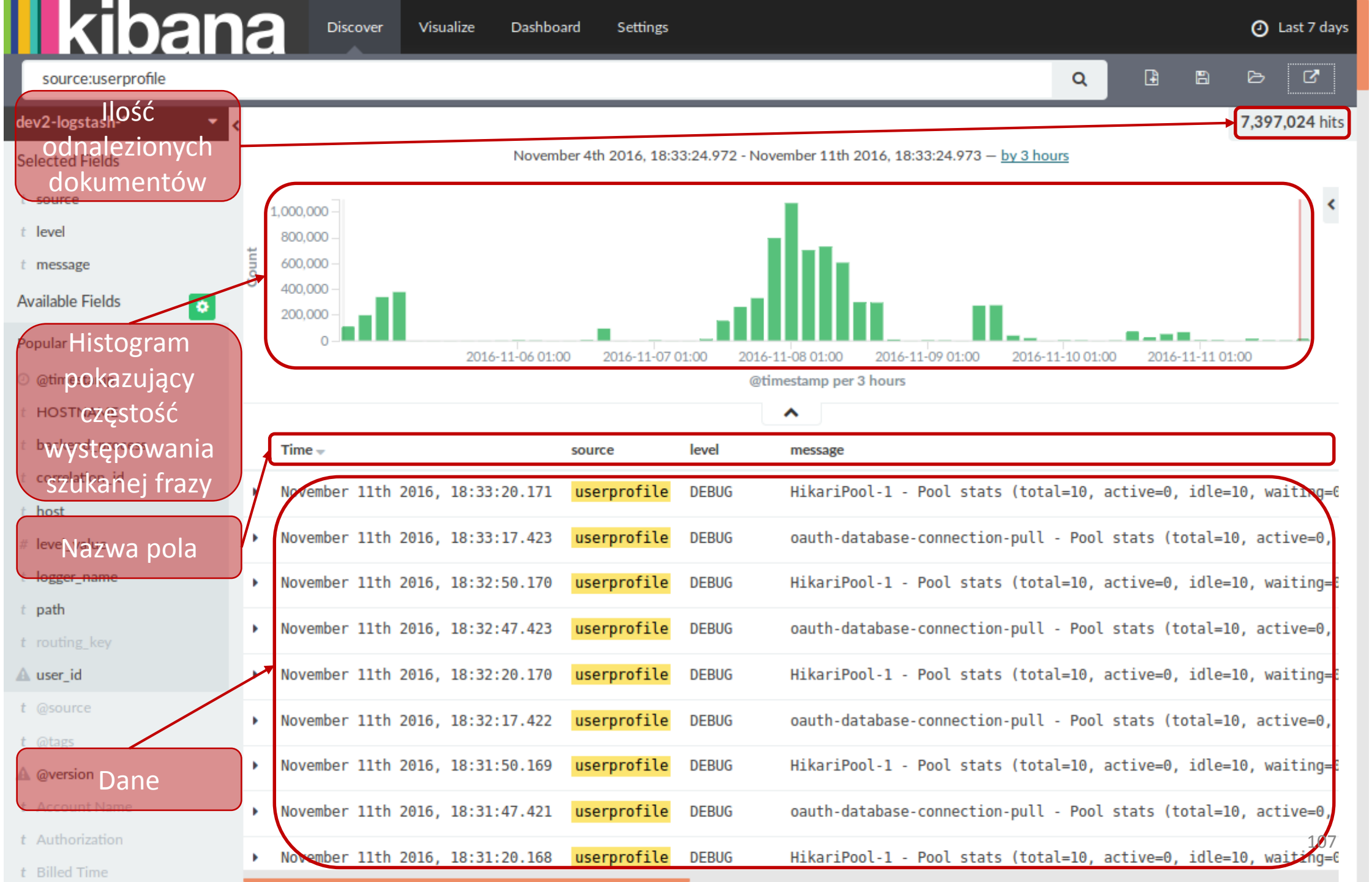
Kibana

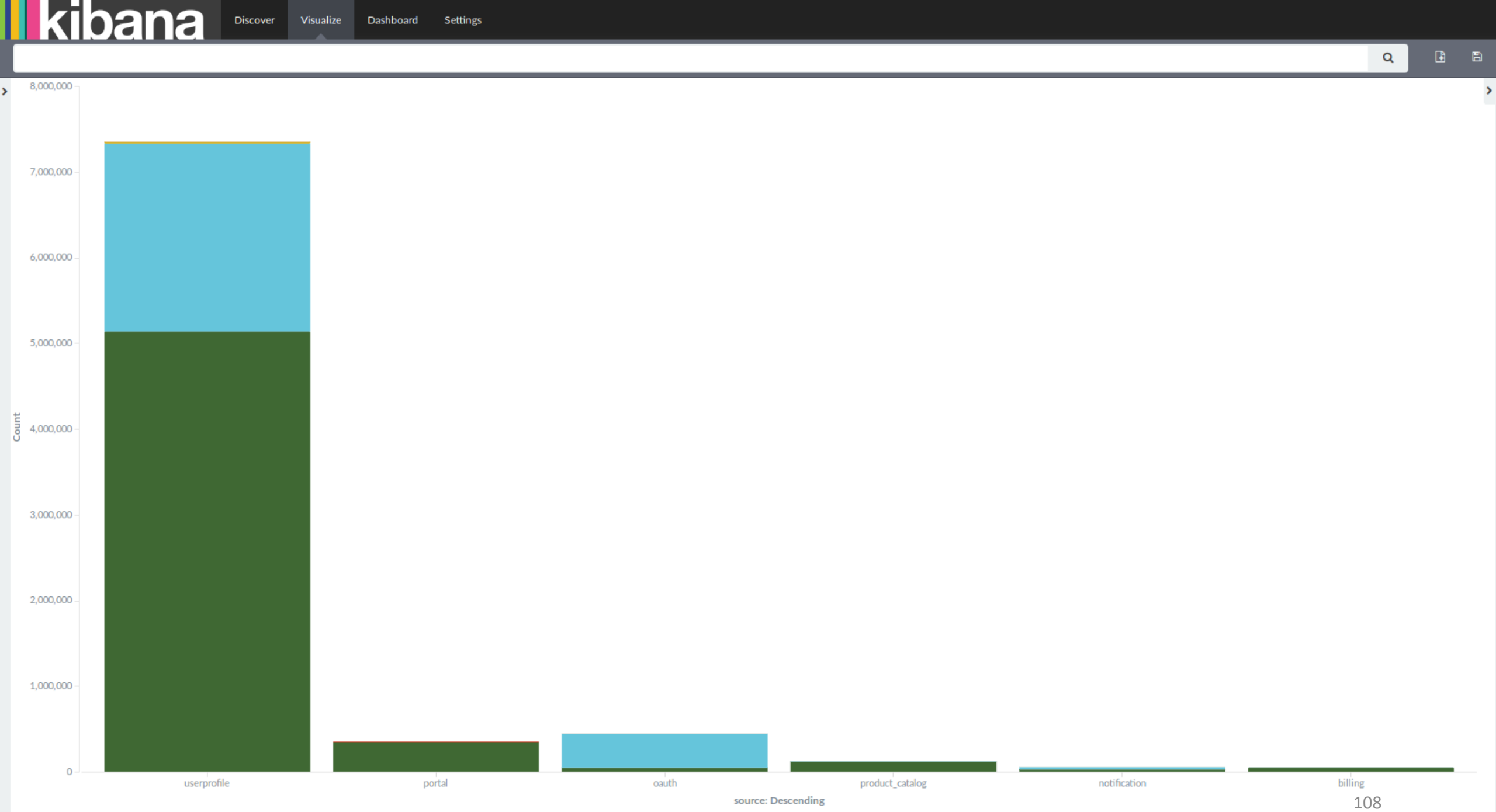
Kibana

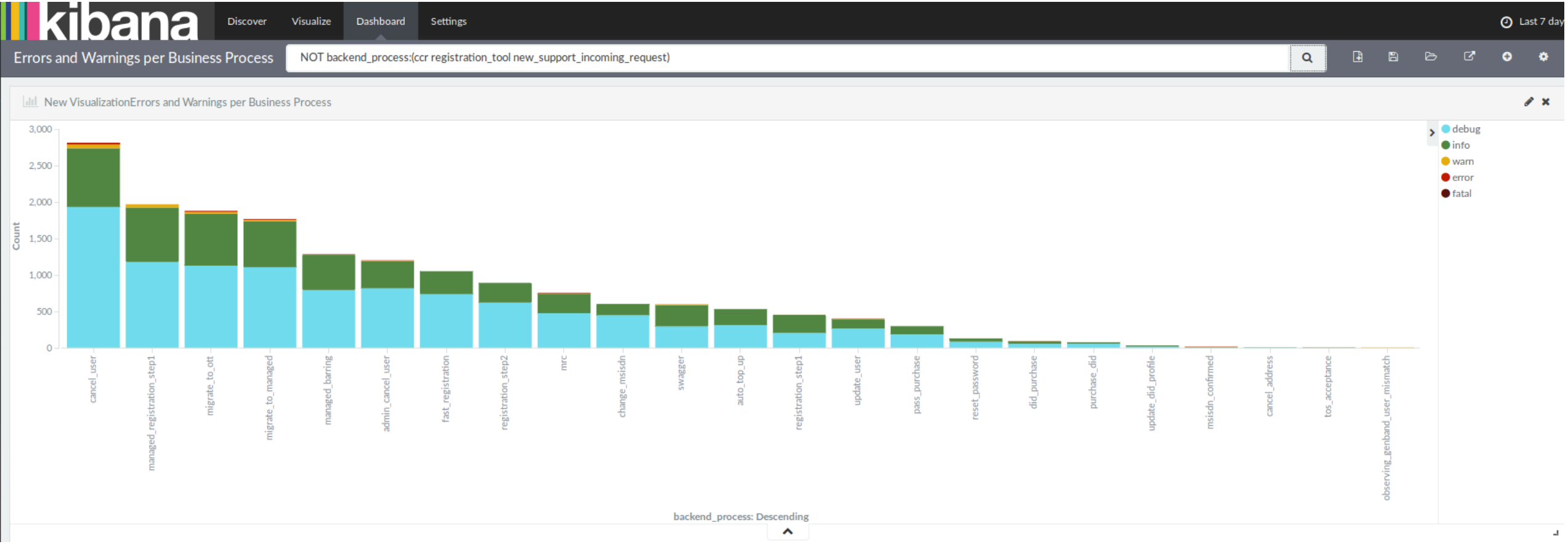
- Narzędzie do odkrywanie informacji zapisanych w danych
- Narzędzie do wyszukiwania danych
- Narzędzie do analizy danych
- GUI dla Elasticsearcha

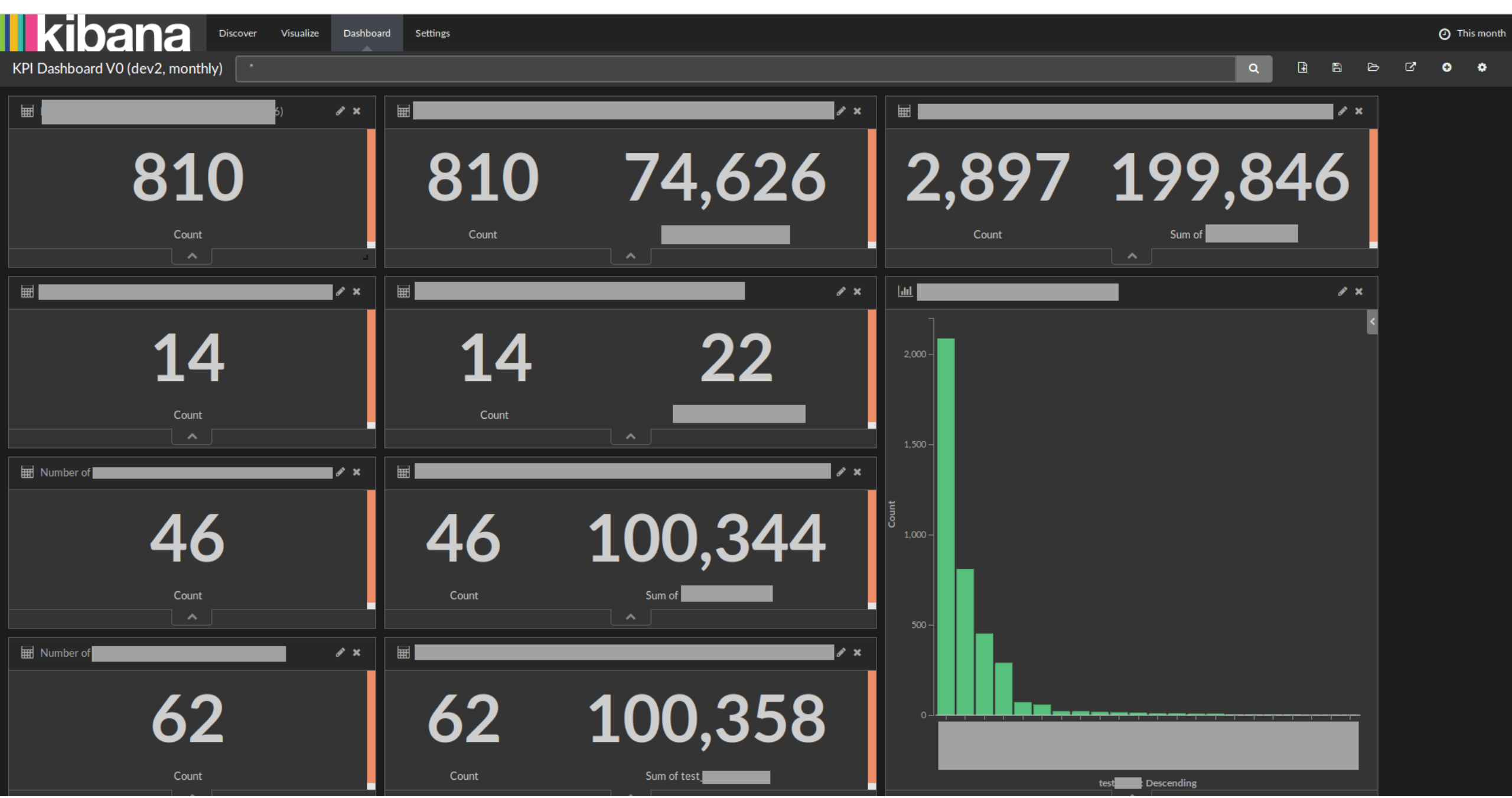

kibana
Discover
Visualize
Dashboard
Settings
Last 7 days
source:userprofile
dev2-logstash-*
7,397,024 hits
Selected Fields
source
level
message
Available Fields
Popular
@timestamp
HOSTNAME
backend_process
correlation_id
host
level_value
logger_name
path
routing_key
user_id
@source
@tags
@version
Account Name
Authorization
Billed Time
November 4th 2016, 18:33:24.972 - November 11th 2016, 18:33:24.973 — by 3 hours
Count
1,000,000
800,000
600,000
400,000
200,000
0
2016-11-06 01:00
2016-11-08 01:00
2016-11-09 01:00
2016-11-10 01:00
2016-11-11 01:00
@timestamp per 3 hours
Time
source
level
message
November 11th 2016, 18:33:20.171 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:33:17.423 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:32:50.170 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:32:47.423 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:32:20.170 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:32:17.422 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:31:50.169 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:31:47.421 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:31:20.168 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
Wyszukiwanie danych
Wizualizacje danych
Zapamiętane pulpity
Ustawienia


kibana
Discover
Visualize
Dashboard
Settings
Last 7 days
source:userprofile
dev2-logstash-*
7,397,024 hits
Selected Fields
source
level
message
Available Fields
Popular
@timestamp
HOSTNAME
backend_process
correlation_id
host
level_value
logger_name
path
routing_key
user_id
@source
@tags
@version
Account Name
Authorization
Billed Time
Count
2016-11-06 01:00
2016-11-07 01:00
2016-11-08 01:00
2016-11-09 01:00
2016-11-10 01:00
2016-11-11 01:00
Time
source
level
message
November 11th 2016, 18:33:20.171 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:33:17.423 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:32:50.170 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:32:47.423 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:32:20.170 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:32:17.422 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:31:50.169 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:31:47.421 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:31:20.168 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
Wybór indeksu
Kryteria wyszukiwania (Query String QueryI
Definiuje przeszukiwany przedział czasu
Zapisywanie i udostępnianie zapytań
Wyświetlane pola dokumentów
Narzędzie wyszukiwania dostępnych pól dokumentów
Lista pól często pojawiających się w dokumentach
Lista dostępnych pól


kibana
Discover
Visualize
Dashboard
Settings
Last 7 days
source:userprofile
dev2-logstash-
Ilość odnalezionych dokumentów
7,397,024 hits
Selected Fields
source
level
message
Available Fields
Popular
November 4th 2016, 18:33:24.972 - November 11th 2016, 18:33:24.973 — by 3 hours
1,000,000
800,000
600,000
400,000
200,000
0
Count
2016-11-06 01:00
2016-11-07 01:00
2016-11-08 01:00
2016-11-09 01:00
2016-11-10 01:00
2016-11-11 01:00
@timestamp per 3 hours
Histogram pokazujący częstość występowania szukanej frazy
Nazwa pola
Dane
Time
source
level
message
November 11th 2016, 18:33:20.171 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:33:17.423 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:32:50.170 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:32:47.423 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:32:20.170 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:32:17.422 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:31:50.169 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0
November 11th 2016, 18:31:47.421 userprofile DEBUG oauth-database-connection-pull - Pool stats (total=10, active=0,
November 11th 2016, 18:31:20.168 userprofile DEBUG HikariPool-1 - Pool stats (total=10, active=0, idle=10, waiting=0


kibana
Discover
Visualize
Dashboard
Settings
8,000,000
7,000,000
6,000,000
5,000,000
4,000,000
3,000,000
2,000,000
1,000,000
0
Count
userprofile
portal
oauth
product_catalog
notification
billing
source: Descending


kibana
Discover
Visualize
Dashboard
Settings
Last 7 day
Errors and Warnings per Business Process
NOT backend_process:(ccr registration_tool new_support_incoming_request)
New VisualizationErrors and Warnings per Business Process
Count
3,000
2,500
2,000
1,500
1,000
500
0
debug
info
warn
error
fatal
cancel_user
managed_registration_step1
migrate_to_ott
migrate_to_managed
managed_barring
admin_cancel_user
fast_registration
registration_step2
mrc
change_msisdn
swagger
auto_top_up
registration_step1
update_user
pass_purchase
reset_password
did_purchase
purchase_did
update_did_profile
msisdn_confirmed
cancel_address
tos_acceptance
observing_genband_user_mismatch
backend_process: Descending


kibana
Discover
Visualize
Dashboard
Settings
This month
KPI Dashboard V0 (dev2, monthly)
810
Count
810
74,626
Count
2,897
199,846
Count
Sum of
14
Count
14
22
Count
Number of
46
Count
46
100,344
Count
Sum of
Number of
62
Count
62
100,358
Count
Sum of test
2,000
1,500
1,000
500
0
Count
test
Descending

Zadanie 6

- Wystartuj Elasticsearcha
  (Znajdziesz go w katalogu domowym w folderze elasticsearch-1.5.2)
- Uruchom Logstasha z konfiguracją:
  ~/logging-training/logstash.conf
  (Pliki wykonywalne Logstasha znajdują się w katalogu domowym w folderze logstash-1.5.0)
- Uruchom i skonfiguruj Kibanę
  (Kibana znajduje się w katalogu domowym w folderze kibana-4.0.2-linux-x64)
- **Wyświetl logi pochodzące z przykładowej aplikacji w Kibanie**