Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Indexace a vizualizace dat za účelem monitorování jejich informační hodnoty

Bakalářská práce

Autor: Vladislav Poverin
Studijní obor: Informační management im3-p

Vedoucí práce: Ing. Pavel Blažek, Ph.D.

Odborný konzultant: Ing. Petr Mikeš

MATERNA, a. s.

Hradec Králové Duben 2020

Prohlášení

Prohlašuji, že jsem bakalářskou/diplomovou práci zpracoval/zpracovala samostatně a s použitím uvedené literatury.

V Hradci Králové dne 20. 04. 2020 . . . . . .

Jméno a příjmení studenta

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Pavlovi Blažkovi, Ph.D. a odbornému konzultantovi Ing. Petrovi Mikešovi ze společnosti MATERNA, a. s. za metodické vedení práce, cenné rady a trpělivost.

Anotace

POVERIN, V. Indexace a vizualizace dat za účelem analýzy jejich informační hodnoty. Hradec Králové, 2019/20. Bakalářská práce na fakultě Informatiky a managementu Univerzity Hradec Králové. Vedoucí bakalářské práce Ing. Pavel Blažek, Ph.D.

Bakalářská práce „Indexace a vizualizace dat za účelem analýzy jejich informační hodnoty“ se zabývá zkoumáním možnosti aplikace open source software na podporu analýzy a vizualizace dat, konkrétně systémových logových zpráv ve společnosti MATERNA, a. s. V teoretické části práce vysvětluje základní pojmy a koncepty týkající se tohoto tématu. Obsahuje také analýzu dostupných nástrojů pro práci s nestrukturovanými daty a odůvodnění výběru konkrétního řešení. Praktická část je věnována testování a nasazení konkrétních nástrojů, které usnadňují práci s daty a poskytují tak cenné informace. Bakalářská práce se snaží poukázat jak na důležitost a výhody, které analýza nestrukturovaných dat přináší, tak na konkrétní východiska.

Klíčová slova: open-source, log, syslog, logging, správa logů, databáze, SQL, NoSQL, analýza logů, indexace, vizualizace

Annotation

POVERIN, V. Data Indexing and Visualization in Order to Monitor its Information Value. Hradec Králové, 2019/20. Bachelor thesis at the Faculty of Information Technologies and Management University Hradec Králové. Supervisor Ing. Pavel Blažek, Ph.D.

This Bachelor thesis examines the different types of open source software that support analysis and visualization of unstructured data, system log messages at the company MATERNA, a. s. in particular. The theoretical part of the thesis covers the explanation of basic notions and concepts relating to the topic. It also contains analysis of available tools for working with unstructured data and justification behind choosing a specific solution. The practical section focuses on testing and deployment of specific tools which imprive efficiency when working with unstructured data while also proving valuable information. The Bachelor thesis intends to show importance and advantages of analysis of unstructured data and introduce some specific solutions.

Key words: open-source, log, syslog, logging, log management, database, SQL, NoSQL, log analysis, indexing, visualization

Obsah

Úvod

Analýza dat je v dnešní společnosti stále důležitějším pojmem z hlediska podnikové informatiky. Informovanost představuje jednu z hlavních konkurenčních výhod a jejich kvalita a způsob využití nám umožňují pokrok nejen z pohledu konkurenčního boje, ale i z pohledu vývoje společnosti. Bohužel není snadné přidat datům obohacující informační hodnotu. Je to způsobeno zejména tím, že často nejsme schopni datům, se kterými přijdeme do styku, porozumět a patřičně je zpracovat.

Každý, ať už se jedná o podnik či jedince, přijímá nepřetržitě větší množství informací. Je pro nás tedy stále složitější informace rozlišovat na podstatné a nepodstatné. Dochází tak k růstu potřeby znalostí, které mohou být nápomocny při rozlišování důležitosti informace. Při současném objemu dat to však není jediná věc, kterou potřebujeme k tomu, abychom získávaným datům porozuměli. Vytváří se proto různé podpůrné softwary, kterých je možné využít k porozumění a práci s daty.

V této bakalářské práci bude několika takovým softwarům zaměřeným zejména na analýzu a vizualizaci systémových logových zpráv věnována pozornost. Práce poskytne vhled nejenom do principů fungování jednotlivých softwarů, ale i jejich analýzu a způsoby nasazení do provozu.

1 Log a logování

Log je obecně záznam o nějaké činnosti ve výpočetním systému. Jedná se o logovou zprávu, kterou nám může poskytnout výpočetní systém, přístroj, software atd., jako reakci na nějaký podnět. Podnětem může v Unixovém systému být například přihlášení či odhlášení uživatele, zprávy generované firewallem, které nám poskytují informace o blokování, povolení připojení aplikací, ale i v případě, kdy hrozí nějaké selhání systému.

Logová zpráva je obvykle řádek textu, který obsahuje informace jako jsou například:

- Čas, ve kterém loghost obdržel zprávu.
- Zdroj události, o které nás zpráva informuje. Může se jednat např. o IP adresu routeru.
- Data, která daná zpráva obsahuje. Nachází se zde i čas vygenerování zprávy.

Anton Chuvakin ve své knize Logging and Log Management [1] rozděluje logové zprávy do pěti kategorií na:

- Information

  Tyto zprávy uživateli či administrátorovi výpočetního systému poskytují informaci o nějakém dění v daném systému. Například, jak už bylo zmíněno, přihlášení či odhlášení uživatele. V případě, že se do systému přihlásí nějaký uživatel mimo jeho pracovní dobu, jedná se o důvod administrátora informovat o tomto dění.
- Debug

  Debug zprávy jsou většinou generovány zejména softwarovými systémy za účelem pomoci programátorům při řešení problémů, které mohou nastat při vykonávání kódu daným softwarem.
- Warning

  Tento typ zpráv se objevuje v situacích, kdy se v systému nachází výjimky, které však neovlivňují jeho běh.
- Error

  Zprávy s označením error jsou využívány k přijímání a předávání chyb, ke kterým dochází na různých úrovních počítačového systému.
- Alert

  Alarmové zprávy nás informují o neobvyklé situaci, která v systému, programu či stroji nastala. Tyto zprávy řadíme spíše do odvětví bezpečnosti zařízení a s ním spojeným operačním systémem.

1.1 Transport logových zpráv

Zprávy nesoucí informaci o stavech systémů k transportu využívají syslog protokolu. Jedná se o standard, který je pro daný druh komunikace hojně aplikován. Najdeme ho jak na Unixových systémech, tak i na Windows a dalších. [2]

Komunikace probíhá prostřednictvím vztahu klient – server, pro kterou se z počátku využíval User Datagram Protocol (UDP). Dnes je však díky požadavkům na spolehlivost doručování zpráv možnost využít Transmission Control Protocol (TCP).

1.1.1 Transmission Control Protocol (TCP)

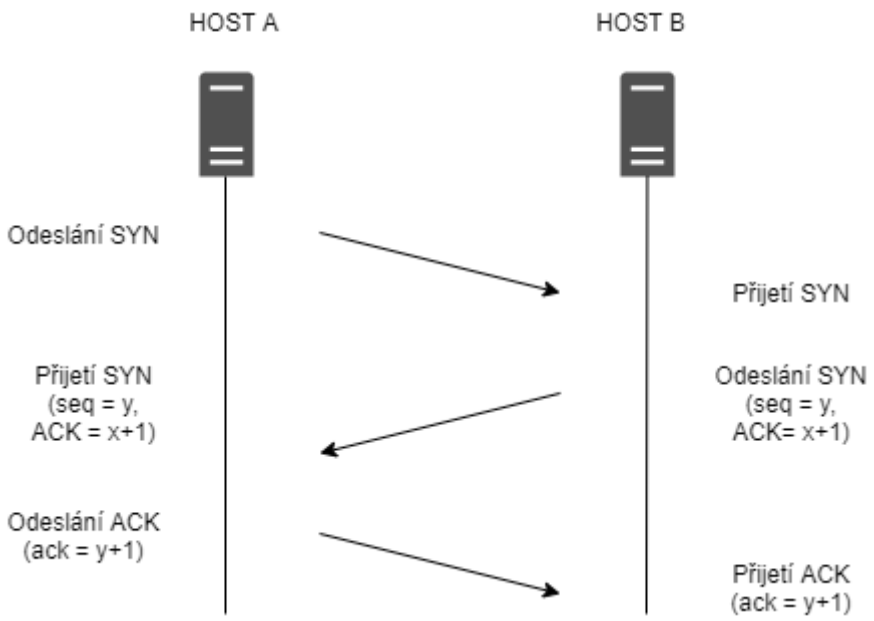
Transmission Control Protocol neboli TCP je přenosový protokol transportní vrstvy. Zatím co protokol IP přenáší data mezi jednotlivými počítači v Internetu a řadí se mezi protokoly síťové vrstvy, TCP je protokol vyšší vrstvy, a zprostředkovává komunikaci mezi aplikacemi běžícími na daných počítačích. Aplikace posílá proud bitů TCP protokolu k doručení, který ho rozděluje na přiměřeně velké segmenty. Segment se skládá ze záhlaví, které slouží zejména pro kontrolní účely a přenášeného bloku dat. Protokol TCP dokáže vytvořit spojení mezi dvěma systémy a umožnit tak potvrzovaný přenos dat. Toto spojení je obousměrné, jinými slovy, data mohou putovat nezávisle oběma směry. Správnost doručení dat je monitorována kontrolním součtem odeslaných a přijatých bitů. Příjemce potvrzuje příjem dat, tzn. poškozená či ztracená data mohou být vyžádaná znovu a opětovně doručená. Při použití TCP protokolu se proto nemusíme bát o ztrátu ani poškození dat v průběhu přenosu. [3]

1.1.2 Transmission Control Protocol Three-way handshake

Pro navázání obousměrného spolehlivého spojení mezi klientem a serverem TCP protokol využívá princip tzv. Three-Way Handshake, které umožňuje jedné straně zahájit spojení a druhé straně spojení buď přijmout či odmítnout. Každá ze stran odesílá SYN (Synchronize Sequence Number) a přijímá ACK (Acknowledgement) od protější strany. SYN a ACK jsou součástí pole řídících bitů (Flags), které se nachází v záhlaví (hlavičce) TCP segmentu a obsahuje šest jednobitových polí používaných ke správě TCP spojení. Při vytváření TCP spojení obě strany vygenerují ISN (Initial Sequence Number). ISN je náhodně vygenerované číslo od 0 do 2^{32}-1 přidané do hlavičky SYN.

Při zahájení spojení je ze všeho nejdříve zapotřebí uvedení serverové strany do režimu naslouchání (LISTEN) na určitém portu. Server dále vytvoří transmission control block (TCB), který obsahuje veškeré informace týkající se spojení. To znamená, že server je připraven pro přijetí SYN od klienta a navázání spojení.

- Prvním krokem na straně klienta je vytvoření (TCB) a následně odeslání SYN zprávy serveru.
- Server od klienta příjme SYN a odešle SYN + ACK zprávu zpátky klientovi. SYN + ACK zpráva obsahuje potvrzení klientova SYN a SYN serveru.
- Klient obdrží SYN + ACK od serveru a zašle ACK SYN serveru. Ze strany klienta je spojení navázáno.
- Server obdrží ACK na svůj SYN, čímž dochází k navázání spojení i ze strany serveru. [4]



Obrázek 1: Příklad TCP three-way handshake

1.2 User Datagram Protocol (UDP)

User Datagram Protocol neboli UDP je stejně jako TCP přenosový protokol transportní vrstvy a využívá IP protokol jako síťový protokol. UDP však nenavazuje spojení mezi systémy. Odesílatel odešle UDP datagram příjemci a už se nezajímá, zda byla data doručena či nikoliv. UDP protokol nepodporuje kontrolní součet. Data se tak v průběhu přenosu mohou poškodit bez toho, aniž by o tom odesílatel či příjemce věděli. UDP protokol je tak považován za protokol nespolehlivý, avšak úsporný. Využívá se v takové komunikaci, ve které má rychlost a jednoduchost doručení dat přednost před spolehlivostí doručení. [3]

1.3 Další protokoly pracující na transportní vrstvě

1.3.1 UDP Lite

The Lightweight User Datagram Protocol (UDP-Lite) je protokol podobný UDP. Nepodporuje však kontrolu checksum. Proto byl navržen spíše pro aplikace, které preferují doručení poškozených dat než nedoručení dat žádných. Protokol byl navržen zejména pro multimédiové protokoly, jako jsou Voice over IP (VoIP) nebo streamování videí. [5]

1.3.2 SCTP

Stream Control Transmission Protocol v sobě spojuje vlastnosti TCP a UDP. Jedná se o spolehlivý a nespojový protokol transportní vrstvy, který je často využíván pro přenos telefonní signalizace po IP. SCTP je na rozdíl od TCP (který je orientován na přenos proudu jednotlivých bitů) orientován na přenos zpráv. Výhodou SCTP nad TCP protokolem je například schopnost přenosu navzájem nezávislých proudů. Případné potíže v jednom proudu neovlivní chod proudů ostatních. Velkým rozdílem mezi SCTP a TCP je tzv. multi-homing. SCTP je navržen tak, aby byl schopen poskytnout robustní komunikaci mezi dvěma endpointy, kdy každý z nich může být dostupný z více než jedné transportní adresy. Dostupné adresy si partneři vyměňují při vytvoření spojení a může se jednat o libovolnou směs IPv4 a IPv6 adres. Během komunikace je jedna z nich brána jako primární a na ni jsou odesílána data. [6]

2 Software pro správu logových zpráv

Software pro správu a analýzu logových dat by měli zahrnovat veškeré funkce, které umožní získat potřebné informace z nestrukturovaných logových zpráv.

Tyto funkce můžeme rozdělit na:

- Dolování logových zpráv ze zdroje
- Syntaktickou analýzu logových zpráv
- Indexace logové zprávy
- Vizualizace dat

2.1 Dolování logových zpráv ze zdroje

Dolování logových zpráv zajišťují „programy“, které jsou schopny logové zprávy z jejich zdroje dopravit do prostředí, kde dohází k jejich dalšímu zpracování. Odlišné logové systémy mají odlišný způsob, jakým zachází se zprávami, které generují. Dělí se na dvě základní kategorie dle používaných protokolů:

Push-based – znamená, že zdroje logových zpráv jsou schopny ukládat zprávy na lokální disk, nebo je samostatně dopravit po síti tam kam je potřeba.

Pull-based – jsou software, které dokážou získat logové zprávy přímo ze zdroje.

Mezi tři hlavní push-based můžeme zařadit Syslog, SNMP, Windows Event Log. Veškeré zdroje logových zpráv k přepravě po síti využívají protokoly TCP nebo UDP.

2.1.1 Syslog

Syslog je rozsáhlý logovací systém a standardizovaný protokol RFC5424. Jeho základní funkcí je ukládání logových zpráv, které jsou zasílány různými programy v systému do souborů. Tímto je možné dát administrátorům systémů větší kontrolu nad děním v daném systému.

Syslog se skládá ze syslog démona (syslogd). Ten je spuštěn při zavádění systému a stará se o sběr, filtrování a ukládání logových zpráv. Syslogd získává log zprávy od kernelu, systémových procesů a aplikací prostřednictvím schránky (Unix domain socket). Syslogd může také dostávat data ze vzdálených zdrojů za použití UDP na portu 514. Logové zprávy jsou dále ukládané, v UNIX systémech do adresáře /var/log/*.log a mohou mít následující tvar.

```
Nov 20 20:04:37 bak2 sudo: pam_unix(sudo:session): session opened for
user root by vladbak2(uid=0)
```

Dnes se pro přesun logových zpráv využívá „rozšířené“ verze syslogu označované rsyslog. Ten má totožný konfigurační soubor jako syslog, poskytuje však nové funkce jako je například podpora RELP protokolu pro komunikaci se vzdálenými zdroji, naslouchání TCP protokolu či nová omezení při filtrování logů. [7]

2.1.2 Filebeat

V roce 2015 společnost Elastic zahájila nový projekt s názvem The Beats. Jedná se o open source data kolektory pro přenos dat založené na libbeat, společné knihovně napsané v programovacím jazyce Go. Tím Elastic podporuje vývoj nových Beats na míru. Beats zajišťují sběr, zpracování a zasílání dat o různých procesech v systému. Data jsou pak přenášena do Elasticsearch. Instalují se na server a monitorují soubory nebo prostředí definované v konfiguračních souborech. Postupem času byly společností Elastic vyvíjeny nové druhy Beats, každý pro sběr specifického typu informace.

Mezi nejznámější Beats patří:

- Filebeat (přeprava logových zpráv)
- Metricbeat (přeprava metrických údajů)
- Packetbeat (přeprava informací o výměně paketů mezi aplikacemi v síti)
- Winlogbeat (přeprava informací o softwarových či hardwarových událostech operačního systému Windows)

Dále v této práci je pozornost zaměřena na Filebeat. Jak už bylo zmíněno, jde o nástroj pro sběr a předávání logových zpráv. Je instalován na serveru, ze kterého mají být logové zprávy transportovány. Filebeat je schopen posílat data buď přímo do databáze, nebo do software, který poskytuje doplňující úpravu dat před jejich indexací.

Filebeat sestává ze dvou základních částí:

- Vstupy (inputs)

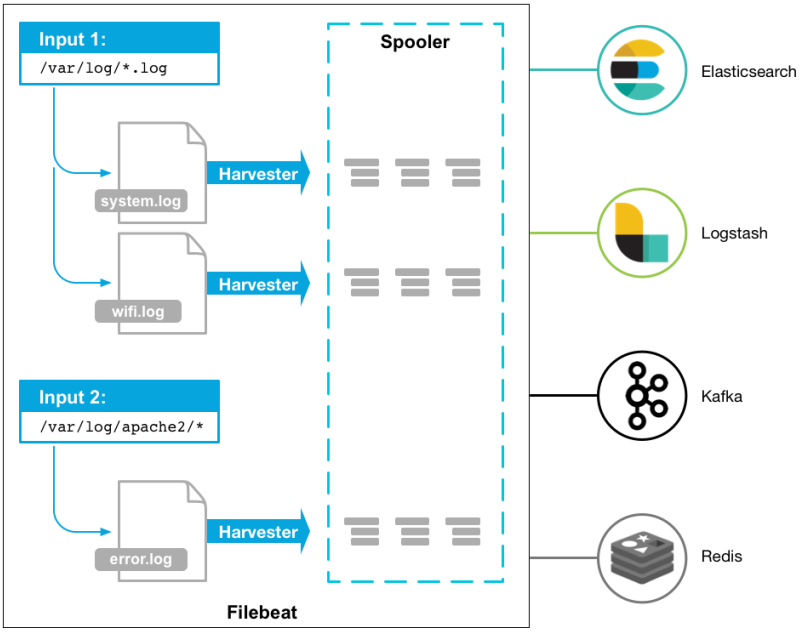
  Vstupy, respektive inputy, jsou definovány cestou a typem v konfiguračním souboru Filebeatu. Inputy řídí sběr dat a odpovídají za nalezení správného zdroje, ze kterého má

čtení dat probíhat. Jelikož mohou být soubory a cesty k nim přejmenovány či přemístěny, Filebeat pro každý input ukládá identifikační údaje, které mu pomohou zjistit, zda byl soubor již definován jako zdroj dat.

Filebeat podporuje několik input datových typů jako jsou například log, stdin, UDP, TCP, kafka, docker, syslog atd.

- Sběrač (harvester)

Sběrač je odpovědný za čtení obsahu specifického souboru. Ten čte řádek po řádku a odesílá obsah na výstup. Pro každý soubor je spuštěn jeden sběrač, který se stará jak o otevření, tak uzavření souboru po dokončení čtení.



Obrázek 2: Schéma software Filebeat [7]

Z obrázku č. 2 je vidět, že pro sběr dat má Filebeat definovánu cestu ve svém konfiguračním souboru v sekci input. Pro každý adresář s logovými zprávami, který je nalezen přes definovanou cestu, spustí sběrač neboli harvester (viz obrázek 2). Ten pak čte obsah logových zpráv a přeposílá nová data libbeatu, který agreguje události a odesílá agregovaná data na výstup, který je definován v konfiguračním souboru.

Filebeat je naprogramován v jazyce Go a je navržen tak, aby měl nízké nároky na operační paměť a zvládal odbavit velké množství dat. Díky ukládání dodávkových stavů do registrů zajišťuje doručení bez ztráty dat. Filebeat zaznamenává poslední řádek, který byl úspěšně uložen v registrech a v případě nedostupnosti příjemce či sítových problémů si pamatuje v jaké části souboru došlo k potížím. Při obnovení spojení pokračuje tam, kde skončil. [7]

2.2 Syntaktická analýza logových zpráv

Před indexací logové zprávy je zapotřebí provést syntaktickou analýzu logové zprávy. Logové zprávy jsou dolovány v nestrukturované podobě, které není vždy snadné porozumět. Je tedy potřeba zprávě dát strukturu, která bude jednotná pro veškeré zprávy daného typu a usnadní nám další manipulaci s logovou zprávou.

Syntaktická analýza neboli parsing je nezbytnou součástí procesu analýzy logových zpráv. Jedná se o techniku, při které jsou textové řetězce uspořádány dle předem definovaných vzorů. Příchozí text je porovnáván se vzory a v případě shody je možné s textem manipulovat různými způsoby. Tyto vzory nám dovolují mimo jiné například odstranit nehodící se znaky z obdržené zprávy či přeměnit strukturu textu.

Nejznámějšími nástroji pro parsing a analýzu logových zpráv jsou:

- Splunk – Placený softwarový produkt, který poskytuje nástroje pro sběr a analýzu dat v sobě zahrnuje i funkci parcingu.
- Fluentd – Open source kolektor dat, který v sobě zahrnuje i určité filtry a pluginy, kterých se dá využít k lepší strukturaci příchozích dat.
- Graylog – Open source robustní řešení pro management logových zpráv s možností parsingu a úpravy vstupních logových zpráv.

V této bakalářské práci je využíváno k syntaktické analýze logových zpráv software Logstash. Jedná se o open source software, který poskytuje společnost Elastic.

2.2.1 Logstash

Logstash je open source modul, který dokáže sbírat data z více zdrojů a následně je odesílat tam, kam je potřeba. Alespoň pro to byl původně navržen. Byl však rozšířen o spoustu jiných funkcí. Nabízí tak mimo přepravy logových zpráv také jejich úpravu na vstupu. Je tak možné změnit příchozí zprávu do potřebného formátu, který bude vhodný pro uložení do databáze a další analýzu. Logstash k parsování dat využívá parsovacího jazyku Grok, který je založen na principu regexu. Mimo syrový Grok, který je ne každému hned srozumitelný a práce s ním není jednoduchá, ovládá také několik již nadefinovaných vzorů a filtrů, jež je možné využít jak na vstupu, tak na výstupu. To umožňuje větší flexibilitu a zásadně usnadňuje syntaktickou analýzu.

Logstsash byl primárně navržen pro zasílání výstupních dat do Elasticsearch. Nyní však disponuje celkem rozsáhlým množstvím output pluginů a strukturovaná data mohou být zasílána do jiných úložišť jako například MongoDB nebo Solr. [8] [9]

2.3 Ukládání a indexace logové zprávy

Poté co je příchozí logová zpráva uspořádána do námi požadované struktury, přichází na řadu uložení této logové zprávy do databáze. Nejpoužívanějšími databázovými systémy jsou stále relační databáze, jejichž principy budou v této práci stručně popsány pro pochopení základní myšlenky těchto systémů. Další možnosti uchovávání dat je princip tzv. NoSQL databází, který je využíván v praktické části této práce a bude mu tedy věnována větší pozornost.

2.3.1 Relační databáze

Data v relační databázi (relational database management systems, RDBMS) jsou reprezentována v n-ticích organizovaných do relací a vyhledávání v nich je zprostředkováno pomocí operací relační algebry. Nejjednodušší reprezentace n-tice v relační databázi je pomocí řádku v tabulce. Je to z důvodu nepřehlednosti zápisu pomocí množin. Jeden řádek tabulky představuje jeden záznam v databázi. Řádek je složen z několika polí neboli sloupců a každý z nich má pevně daný datový typ. Pojmy relace a tabulka však nemůžeme vzájemně zaměňovat a je třeba je mezi sebou rozlišit. Každá relace může být zobrazena pomocí tabulky, ale ne každá tabulka může být zobrazena pomocí relace. Pro komunikaci s relační databází se využívá Structured Query Language (SQL). Jedná se o standardizovaný vyhledávací jazyk pro relační databáze a dovoluje nám data do databáze ukládat a zobrazovat.

Jedním z nejdůležitějších pojmů týkajících se relačních databází jsou transakce a jejich tzv. ACID architektura. Transakce se může skládat z několika procesů. Ty jsou provedeny všechny, nebo ani jeden z nich. Výsledkem vykonání transakce je změna v databázi například v podobě vložení nových dat do databáze.

Po úspěšném provedení všech kroků v transakci dochází k tzv. commitu. Ten slouží jako potvrzení aktivity v databázi, která se udála od posledního commitu. V případě jakékoliv chyby v průběhu transakce nastává tzv. rollback a veškeré změny jsou vráceny do stavu posledního úspěšného commitu. Jako příklad transakce Mullins ve své knize Database administration [10] uvádí výběr z bankomatu. Předpokládejme, že si chceme vybrat peníze z bankomatu nebo u

pokladny. Když banka obdrží požadavek na výběr peněz, je zapotřebí vykonat několik úkolů, které povedou k úspěšnému obdržení požadované částky.

1. Kontrola zůstatku na účtu.
2. Pokud není na účtu dostatek peněz, zastavit transakci, v opačném případě pokračovat.
3. Odečíst požadovanou částku z účtu klienta.
4. Vystavit doklad o provedení transakce.
5. Vydání dané částky a dokladu.

Všechny kroky musí být splněny, aby mohla být transakce považována za úspěšnou a oba účastníci dané transakce byli spokojeni.

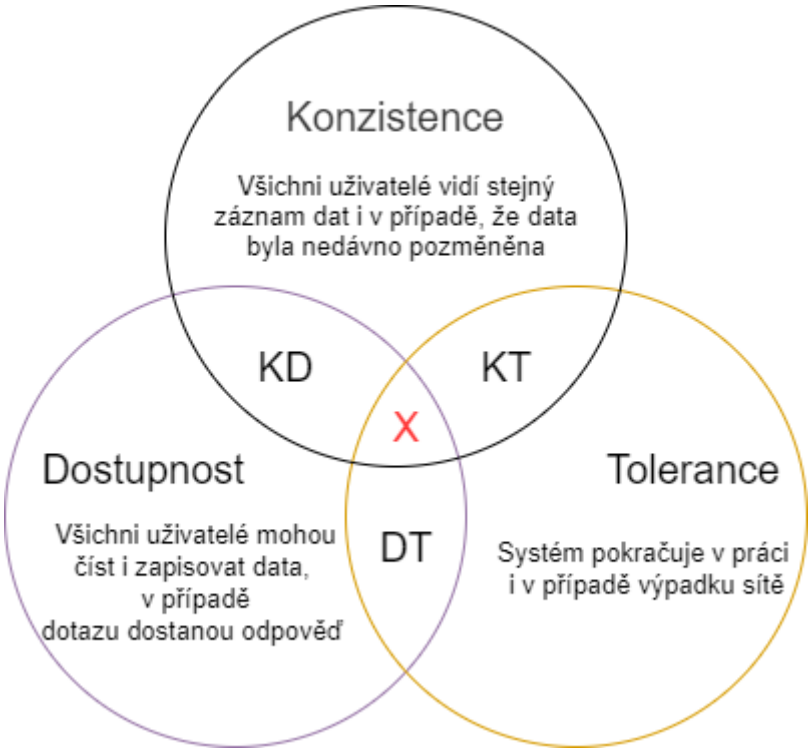
Pravidla, která musí transakce splňovat, v sobě zahrnuje zkratka ACID (Atomicity - atomicita, Consistency - konzistence, Isolation - izolovanost, Durability - trvalost).

- Atomicita – transakce je jeden celek, který je dále nedělitelný a buď se provedou všechny procesy anebo žádný.
- Konzistence – transakce může databázi změnit z jednoho validního stavu na druhý validní stav. Jinými slovy, data v databázi musí splňovat předem daná pravidla jak před provedením transakce, tak po jejím provedení.
- Izolovanost – každá transakce je oddělená od ostatních a průběh jedné transakce nijak neovlivňuje průběh té druhé.
- Trvalost – výsledek transakce je trvalý. Z výše uvedeného příkladu výběru peněz z bankomatu, je zřejmé, že peníze po výběru byly odepsány z účtu. Ani v případě kolapsu systému není možné, aby se tento stav, jakkoliv změnil.

Vlastností ACID není v reálném světě tak jednoduché dosáhnout, obzvlášť když se jedná o rozsáhlé databázové systémy, a to zejména s ohledem na distribuci, replikaci a výpadky sítě. Řídit se neustálým striktním dodržováním těchto pravidel tak často vede k zpomalování distribuovaného databázového systému. V praxi je tedy často využíván tzv. princip CAP teorému (Consistency – konzistence, Availability – dostupnost, Partition tolerance – tolerance vůči ztrátě spojení). Eric Brewer, autor teorému, říká, že databázový systém může splnit pouze dvě ze tří uvedených požadavků.

- Konzistence – princip tohoto požadavku je založen na existenci pouze jedné verze dat v daný okamžik.

- Dostupnost – služby databáze, respektive požadavky na čtení či zápis dat jsou v každém případě systémem uspokojeny a nemůže dojít k jakékoliv závadě.
- Tolerance vůči ztrátě spojení – schopnost provozu databáze i v případě ztráty spojení mezi jejími částmi.



Obrázek 3: CAP teorém

Teorém na obrázku č. 3 nám ukazuje, že pokud chceme horizontálně škálovat, tedy zvyšovat výkon databáze přidáváním nového hardware jako uzlů v clusteru a spravovat tak čím dál větší objem dat s možností zálohování, potom je nutné vybrat pouze dvě pravidla (dva požadavky), kterými se budeme řídit a dodržovat.

Pro vyšší výkonnost databáze a rychlejší vyhledávání v ní je potřeba data při ukládání také indexovat. Index je alternativní cesta k datům uloženým v databázi. Jeho struktura díky nižšímu počtu I/O operací umožňuje jednodušší přístup k datům. [10] Při výběru dat z relační databáze, nad kterými jsou definovány indexy, proto není zapotřebí procházet všechna data v databázi, dokud nenarazíme na data splňující zvolená kritéria, ale využít indexů, ve kterých jsou data organizována tak, aby bylo možné vybrat pouze relevantní záznamy. Mezi zástupce relačních databázových systémů patří například Oracle a Microsoft SQL Server.

2.3.2 NoSQL databáze

S nárustem množství dat nám standardní databázové systémy nestačí a je zapotřebí nových přístupů. Ty jsou dnes souhrnně označovány jako NoSQL databáze. Většina těchto databázových systémů v dnešní době vzniká pro podporu efektivního zpracování Big Data. Tento pojem definuje například společnost IBM takto: „V závislosti na odvětví a organizaci zahrnují Big Data informace z interních a externích zdrojů, jako jsou transakce, sociální média, podniková data, senzory a mobilní zařízení. Firmy mohou tato data využívat, aby lépe přizpůsobily své výrobky a služby potřebám zákazníka, dále optimalizovaly provoz a infrastrukturu a/nebo nalezly zcela nové zdroje příjmů“. [11]

Pojem NoSQL databáze byl poprvé použit pro označení open source relační databáze, která pro přístup k datům nevyužívala jazyka SQL kolem roku 1998. Dnes můžeme za NoSQL databáze označit všechny databáze, které využívají nejenom relačního přístupu nebo ho nevyužívají vůbec, jsou škálovatelné s podporou replikace a open source přístupem.

Většina NoSQL databází oproti relačním databázím nevyužívají pro své transakce ACID pravidel, nýbrž přístup BASE. Jedná se opět o zkratku, která je složená ze začátečních písmen následujících pojmů Basically available, Soft-state, Eventually consistent.

- Basically available – Většina dat je dostupná po většinu času, nemůže se stát, že systém přestane fungovat úplně.
- Soft-state – Oproti ACID commitu, kdy jsou všechny úpravy v databázi úspěšně potvrzeny a uloženy v BASE principu tomu tam nemusí být vždy. Může se tak stát, že různým uživatelům ve stejnou dobu můžou být zobrazena jiná data.
- Eventually consistent – Databáze se převážně nachází v konzistentním stavu, avšak dovoluje i částečnou nekonzistenci dat za účelem zvýšit dostupnosti dat a rychlosti dotazování. Může se tak stát, že různí uživatelé v určitém čase (inconsistency window) vidí odlišná data. [12]

Typologie NoSQL databází

Oba přístupy, jak relační databáze, tak NoSQL databáze, využívají podobné datové struktury, jež jsou známy delší dobu a slouží pro podporu vyššího výkonu databázového systému. Těmi jsou například stromy (b-trees), fronty, pole atd. Oba typy databází od sebe odlišuje to, jak dané

techniky pro ukládání dat předkládají uživateli k použití. V literatuře a mezi vývojáři jsou NoSQL databáze rozděleny na čtyři typy v závislosti na datových modelech jednotlivých systémů. [14]

- Databáze klíč-hodnota – jedná se o velmi jednoduchou organizaci dat, která disponuje velmi svobodným datovým modelem. Tyto databáze jsou schopny ukládat v podstatě jakýkoliv typ dat, kde je každý záznam opatřen unikátním klíčem. S daty tak není možné manipulovat nebo je vyhledávat na základě jejich obsahu, nýbrž pouze prostřednictvím unikátního klíče.
- Grafické databáze – jsou databáze, které podporují organizaci dat v grafech. Tento typ NoSQL databáze se využívá pro data, která je vhodné dotazovat jako grafy. Datový model grafické databáze je rozdělen do dvou entit – uzly a hrany, které je mezi sebou propojují. Uzly tak tvoří objekty, které obsahují atributy a hrany jsou vztahy mezi těmito objekty.

- Dokumentové databáze – data jsou ukládána v dokumentech. Typickými formáty pro ukládání dat jsou JSON a XML. Dokumenty neobsahují pouze data, nýbrž i metadata. Jedná se tedy o samo-popisnou datovou strukturu. Dokumenty jsou využívány nejenom pro ukládání dat, ale i pro komunikaci s klienty a aplikacemi. Nejvýznamnějším představitelem této datové struktury je NoSQL databáze MongoDB.

- Sloupcové databáze – databáze tohoto typu mají stejně jako relační databáze strukturu založenou na tabulkách, do kterých je možné přidávat sloupce nezávisle na řádcích. Databáze funguje na principu klíč-hodnota. Klíčem je zde název sloupce. NoSQL databáze Cassandra je databáze sloupcového typu.

Mezi nejznámější představitelé NoSQL databází patří například

- Redis
- Cassandra
- MongoDB

Výhody NoSQL databází:

1. Flexibilní škálovatelnost: relační databázové systémy využívají tzv. vertikální škálovatelnost, což znamená nasazení výkonnějšího hardware ve snaze navýšit kapacitu databáze. NoSQL databáze škálují horizontálně. Zpracování každé úlohy může být distribuováno v rámci množiny serverů, které je možné rozšiřovat přidáváním dalších serverů v rámci clusteru.
2. Flexibilní datový model: NoSQL databáze nevyžadují určení žádného databázového modelu anebo je schéma dat volné.
3. Efektivní čtení: NoSQL databáze nevyžadují pevné datové schéma, což nám dovoluje vytvořit datovou strukturu tak, aby nám dovolovala co nejlépe odpovídat na často kladené dotazy.
4. Ekonomický aspekt: Značnou výhodou je samozřejmě i finanční náročnost správy databázového systému. Díky horizontální škálovatelnosti NoSQL databází není od serverů, na které ukládáme data, požadováno velkého výpočetního výkonu. Je tedy možné v clusteru využít relativně levných počítačů.

Nevýhody NoSQL databází:

1. Standardizace přístupů k datům: Relační databáze mají jasně standardizovaný přístup k datům prostřednictvím jazyka SQL. S každou nově implementovanou NoSQL databází je potřeba se učit novou syntaxi.
2. Robustnost: NoSQL databáze mezi námi nejsou tak dlouho jako relační databáze a na rozdíl od relačních databází nedosahují léty prověřené robustnosti.
3. Uživatelská podpora: Myšlenka open source nám poskytuje neustálý vývoj v odvětví NoSQL databází. Bohužel ne všechny projekty nabízí dostačující uživatelskou podporu ve srovnání s relačními databázemi.

S postupem času přicházejí nové problémy, které vyžadují nová řešení. Neustálý nárůst objemu dat s sebou nese nové výzvy a NoSQL přístup nám umožňuje na tyto výzvy reagovat. NoSQL databáze nám zjednodušují práci s nestrukturovanými daty o velkém objemu a dovolují nám je analyzovat na nové úrovni. NoSQL databáze nemají nahradit relační databáze, mohou dokonce být jejím doplňkem. NoSQL databáze v sobě často obsahují i moduly, které podporují SQL syntaxi.

Relační databáze je vhodné použít tam, kde není potřeba horizontální škálování a nepočítáme s markantním nárůstem objemu dat. Naproti tomu jsou NoSQL vhodné pro zpracování velkého množství dat, které nemusí mít jasně danou strukturu a nabízejí mnoho možností jejich další analýzy jako například grafové funkce, fulltextové vyhledávání aj. NoSQL databáze mají smysl tam, kde je kladen důraz na výkon, rychlou odezvu a kdy nejsou kladeny vysoké nároky na konzistenci.

2.3.3 Time series databáze

Time series databáze (v češtině také používán výraz „databáze časových řad“) jsou databáze, které jsou navrženy pro uchovávání velkého objemu dat s časovou souvislostí. Tyto databáze se liší od relačních i NoSQL databází způsobem, jakým data ukládají. Zatímco klasické relační databáze ukládají data do řádků a sloupců tabulek, které jsou často navrženy pro ukládání určitého typu dat, time series databáze ukládají data do „kolekcí“, které sdílí jednotný atribut a tím je čas, přes který jsou veškerá data agregována. To znamená, že vše, co je ukládáno do databáze je spojeno s časovým údajem timestamp. [15]

Time series databáze jsou nejčastěji využívány pro sledování metrických údajů serverů, cen akcií, výkonu aplikací, monitoring síťového provozu, dat ze senzorů v chytrých domácnostech (změny teplot či vlhkosti vzduchu v čase), a mnoho dalších dat, které je možné analyzovat.

Mezi výhody time series databází patří například možnost klást dotazy na velké časové období nebo porovnávat nová data se staršími záznamy v databázi. Dalšími výhodami je rychlost vyhledávání či jednodušší vytváření dotazů pro spojení dat na základě shody atributu timestamp.

Dle portálu [15] jsou nejpoužívanějšími time series databázemi:

1. InfluxDB
2. Kdb+
3. Prometheus
4. Graphite
5. RRDTool

V praktické části této bakalářské práci se setkáme s použitím software Prometheus. Jde o open source software vytvořený v roce 2012 na platformě SoundCloud, který je využíván pro monitoring výpočetních systémů. Jedná se o time series databázi, jenž disponuje mnoha dalšími

funkcionalitami a je navržena tak, aby byla schopna monitorovat servery, databáze, virtuální stroje apod. Monitoring Prometheus provádí prostřednictvím sběru dat z uživatelem určeného zdroje. Ke sběru metrických dat dochází prostřednictvím tzv. exportérů nebo pushgateway komponenty. Exportéry jsou myšleny knihovny a servery, které zprostředkovávají přepravu metrických dat z různých systémů do formátu, který je dále Prometheus schopen zpracovávat. Pushgateway způsob sběru dat je určen pro data aplikací, které nevytváří metrická data přímo (tzv. batch jobs, dávkové soubory) a Prometheus je tak není schopen z důvodu příliš krátkého času trvání získat klasickým dotazováním. [16]

Prometheus je napsaný převážně v programovacím jazyce Go, což ho dělá výkonným a hardwarově nenáročným softwarem. Mimo sběr metrických dat je Prometheus schopen ve svém webovém rozhraní, nebo za použití jiných nástrojů vizualizovat a generovat alarmy, které je poté možné zasílat například prostřednictvím e-mailu.

2.4 Vizualizace logových zpráv

Aby bylo možné z dat získat v co nejkratším čase co nejvíce informací je zapotřebí data, která máme k dispozici, vizualizovat. Je snazší, a pro člověka srozumitelnější, pohled na data už nějak vizuálně zpracovaná. Může se jednat o grafy, barevné rozlišení odlišných typů logových zpráv, rozdělení logových zpráv podle času, místa výskytu, důležitosti apod. tak, aby bylo možné z dané vizualizace v reálném čase sledovat specifické charakteristiky jednotlivých druhů logových zpráv.

Dalším aspektem je také tvorba daných vizualizací. Je zapotřebí, aby aplikace byla schopna nabídnout uživateli dostatečnou svobodu při práci s vizualizačními nástroji a umožnila uživateli vytvořit si vizualizaci na míru. Metody tvorby vizualizací by měly být uživatelsky přívětivé a zacházení s nimi by mělo být pro uživatele intuitivní.

Systém, jenž nám poskytuje vizualizaci, by měl být také vybaven uživatelským rozhraním, které bude schopno ovládat více uživatelů na různých úrovních. Administrátor serverů potřebuje získávat jiná data než například manager prodeje nebo zaměstnanec na oddělení zákaznické podpory.

3 Porovnání software pro správu logových zpráv

Tato část bakalářské práce je věnována popisu a analýze, v daný moment dostupných řešení v oblasti správy logových zpráv. Na jejím základě bylo vybráno konkrétní řešení, jehož implementace je popsána v praktické části.

Při výběru níže uvedených nástrojů byl kladen důraz na open source distribuce a goodwill společností, které je nabízí.

3.1 Elastic Stack

Elastic Stack je open source projekt od společnosti Elastic, dříve také známý jako akronym začátečních písmen software, které společně tvoří robustní monitorovací systém (Elasticsearch, Logstash a Kibana). Jelikož ale došlo k vytvoření nového oddílu nástrojů pro přepravu dat s názvem Beats, tato zkratka ztrácí na významu a upřednostňuje se název Elastic Stack. V předchozích částech této práce byly popsány dva nástroje, ze kterých se Elastic Stack skládá, a to Logstash a Filebeat. Ze zmíněného skupiny software, které v sobě Elastic Stack obsahuje, zbývá Elasticsearch a Kibana. [17]

3.1.1 Elasticsearch

Elasticsearch je srdcem celého projektu Elastic Stack a společnost Elastic ho definuje jako open source nástroj, který slouží k vyhledávání a analýze různého typu dat od textu, čísel až po geologické souřadnice. Data mohou být strukturovaná či nikoli. Elasticsearch je napsán v jazyce Java a vychází z Apache Lucene, což je tzv. vyhledávací systém, jenž využívá pro ukládání dat invertovaného indexu. Invertovaný index je možné si představit jako databázi se strukturou určenou zejména pro ukládání textu a efektivní vyhledávání v něm. [18]

Pojem, se kterým se nejčastěji při styku s invertovanými indexy a Elasticseach setkáváme, je full-text search (full-textové vyhledávání). Elasticsearch ukládá data jako JSON dokumenty, ze kterých se skládá index. V jazyce relačních databází by index představoval databázi a dokument záznam v ní. Dokument je propojen se sadou klíčů jako např. název pole a jeho datový typ v dokumentu, které napomáhají efektivnímu vyhledávání. Při indexaci Elasticsearch rozděluje text na jednotlivé části a ukládá je s identifikátory tohoto textu do vyhledávacího indexu. Při zadání dotazu tak Elasticsearch otevře vyhledávací index, do kterého uložil jednotlivé části, a vrátí shodu téměř v reálném čase. Díky možnosti horizontálního škálování, které umožňuje vytvoření clusteru, je Elasticsearch velmi rychlým a výkonným nástrojem pro analýzu a vyhledávání dat.

Pro vyhledávání a práci s daty Elasticsearch poskytuje REST API. To umožňuje uživateli prostřednictvím http metod přistupovat k datům vzdáleně z více míst.

Zde je příklad dotazu z konzole, který zobrazí všechny indexy:

```
curl -X GET 'http://localhost:9200/_cat/indices?v'
```

3.1.2 Kibana

Kibana je platforma pro vizualizaci a analýzu dat, navržena pro kooperaci s Elasticsearch. Nabízí několik funkcí a možností, jak data vizualizovat a dále s nimi pracovat. Kibana slouží také jako Elasticsearch GUI a disponuje konzolí, pomocí které je interakce s Elasticsearch jednodušší a přehlednější. Součástí Kibany jsou různé grafy, filtry pro podporu vyhledávání v datech anebo například Grok debugger. Ten slouží k ověřování správnosti shody vzorů grok filtrů s předloženým textem, díky kterému je vytváření konfiguračních souborů pro Logstash příjemnější. [19]

Elastic nabízí jako součást Kibana platformy ALERT funkce prostřednictvím X-Pack placeného řešení, kterého je možné využít v rámci 30 dnů zdarma.

Elastic Stack má velmi dobrou uživatelskou podporu. Poskytuje přehlednou dokumentaci veškerých svých produktů, aktivní vývojáře, kteří jsou ochotní poradit s případným problémem či nabídku kurzů pro zdokonalení znalostí uživatelů.

Logstash – dokáže přijímat data z více zdrojů, indexovat a parsovat je do potřebného formátu, ve kterém se poté data ukládají do elasticsearch. Logstash k parsování záznamů využívá parsovací jazyk Grok.

3.2 Graylog

Graylog je open-source log management software, který slouží ke sběru, ukládání a analýze logových zpráv. Pro komunikaci mezi serverem a Graylog není potřeba dalšího software, který by se staral o syntaktickou analýzu příchozích dat, jako je tomu u Elastic Stacku, kde tuto funkci zastává Logstash. Parsovací pravidla jsou řešena přímo v samotném webovém rozhraní Graylogu. Software využívá MongoDB pro ukládání konfiguračních dat. Do databáze jsou ukládána metadata, jako například informace o uživateli nebo stream konfigurace. Žádné

logové zprávy nejsou uloženy v MongoDB, a proto tato NoSQL databáze nemá velký vliv na hardware.

Graylog podporuje příjem dat prostřednictvím různých protokolů jako jsou například:

- Syslog (TCP, UDP, AMQP, Kafka)
- GELF (TCP, UDP, AMQP, Kafka, HTTP)
- AWS (AWS Logs, FlowLogs, CloudTrail)
- Beats/Logstash
- CEF (TCP, UDP, AMQP, Kafka)
- JSON Path from HTTP API
- Netflow (UDP)
- Plain/Raw Text (TCP, UDP, AMQP, Kafka) [20]

Graylog mimo jiné nabízí ALERT funkce v open-source řešení, uživatelsky přívětivé webové rozhraní poskytující grafické zobrazení logových zpráv, ukládání dashboardů, funkce geologace, která je schopna zobrazit mapu, na níž je možné vidět místo výskytu logové zprávy nebo plugin Aggregates, jež zprostředkovává zasílání upozornění na události prostřednictvím e-mailu v HTML. Graylog využívá pro ukládání a vyhledávání dat Elasticsearch. Tím je zajištěna funkce full-textového vyhledávání.

Sidecar je funkcionalita Graylog, která nám dovoluje jednoduché zobrazení informací o našich zdrojích logů. Je zde vidět například jaký software je používán pro sběr logů z daného serveru a jaký má daný server operační systém. [21]

Dokumentace a uživatelská podpora je na dobré úrovni. Graylog má vlastní diskusní fórum, není však tak aktivní jako je tomu u Elastic. Jednou z vlastností Graylog webové stránky je, že tvůrci poskytují video ukázky Graylog funkcí.

Protože zde není využíváno funkcionality Logstash, má toto řešení nižší požadavky na hardware. Bohužel možnost parsování přijímaných dat nenabízí takové funkce jako je schopen poskytnout software od Elasticu.

3.3 Fluentd MongoDB

Řešení s využitím Fluentd, které poskytuje řešení pro sběr logových zpráv z několika serverů a přeposílat je v JSON formátu do MongoDB. Dalším nástrojem je MongoDB Compass, GUI pro tuto NoSQL databázi, které nám dovolí analyzovat a vizualizovat data v ní uložená.

MongoDB:

- Vhodné pro úkládání dat
- Nevhodné pro analýzu dat
- Flexibilní v porovnání s relační DBS
- Nepodporuje full-textové vyhledávání
- Pomalé pro analýzu dat

Je možné použít ElasticSearch namísto MongoDB a Kibanu pro vizualizaci dat s tím, že narušíme architekturu ELK a vyměníme logstash za log collector Fluentd.

3.4 LOGALYZE

LOGalyze nabízí komplexní open source řešení pro sběr logů z více zdrojů, jejich indexaci, analýzu či systém ALERT.

Pro přenos logových zpráv využívá rsyslog. Pracuje s Simple Object Access Protocol (SOAP).

Poskytuje jednoduché uživatelsky přívětivé webové rozhraní. Bohužel uživatelská podpora je na velmi nízké úrovni. Poslední verze softwaru byla vypuštěna v roce 2018 a o další verzi není žádné zmínky. Dokumentace k software je pro uživatele nedostačující. [22]

3.5 Závěr analýzy

Na základě provedené analýzy a informací získaných v průběhu vytváření porovnání, bylo vybráno k testovací a implementaci řešení nabízené společností Elastic – Elastic Stack.

Mezi rozhodující aspekty patří zejména vzájemná součinnost Elasticsearch a Kibany. Funkcionality, jako jsou horizontální škálování a fulltextové vyhledávání, nám díky uživatelsky přívětivému prostředí v Kibaně umožňují získat přehled a kontrolu nad našimi daty. Dalším

významným faktorem je zapojení Logstash jakožto prostředníka mezi zdrojem dat a Elasticsearch, který je schopen přijímat data z více zdrojů a z nepřehledných řádků logových zpráv vytvořit systematický obsah, se kterým můžeme dále pracovat. Jednou z nevýhod tohoto řešení jsou vysoké požadavky na hardware. Elastic dokumentace uvádí 64 GB RAM jako „sweet spot“, ale dá se pracovat i s 32 GB a 16 GB RAM. Co se týká CPU, Elastic doporučuje upřednostnit počet jader před rychlostí procesoru.

4 Praktická část

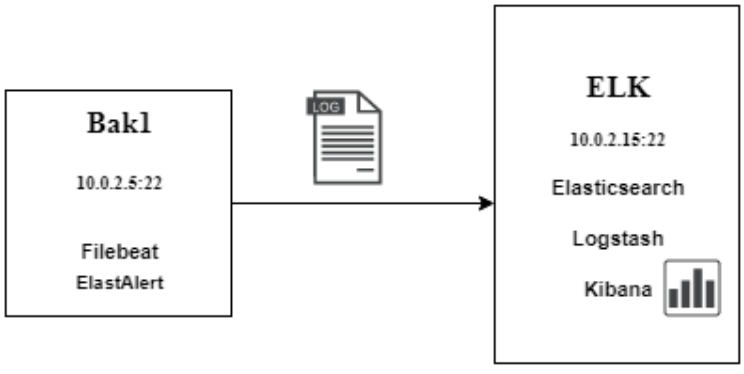
Tato část bakalářské práce je zaměřena na praktické využití představených systémů a jejich podrobnější popis z pohledu implementace. Praktická část je rozdělena na tři samostatné, na sobě nezávislé projekty. Každý z těchto projektů se snaží nalézt řešení konkrétního problému.

4.1 Implementace Elastic Stack

Cílem tohoto projektu je představit některé možnosti a funkce monitorovacího systému, který nám nabízí softwary zahrnuty v Elastic Stack, testovat a poté implementovat řešení, které by bylo schopné zaznamenávat z logových zpráv informace o doručené, nedoručené a ztracené zprávy. Vizualizovat tyto získané informace v koláčovém grafu, jenž by nám poskytoval procentuální rozdělení těchto tří druhů zpráv s podmínkou výskytu logové zprávy s informací o doručení do 72 hodin. V případě výskytu nedoručené či ztracené zprávy informovat o tomto výskytu prostřednictvím alarmů.

K implementaci a testování Elastic Stack bylo využito dvou virtuálních počítačů. K jejich vytvoření posloužil známý software pro virtualizaci – VM VirtualBox od společnosti Oracle. Na obou virtuálních počítačích byl nainstalován open-source operační systém Linux Ubuntu verze 18.04.

	Bak1	ELK
RAM	4GB	6GB
Počet CPU	2	4



Obrázek 4: Schéma testování Elastic Stack

- Konfigurace společné sítě pro zajištění komunikace mezi VMs
- Nastavení předávání logových zpráv pomocí rsyslogu z jedné VM na druhou

4.1.1 Instalace Elastic Stack

Instalaci Elasticsearch je možné provést prostřednictvím apt repositáře což zjednoduší i instalaci Kibany. Repositář je ošetřen bezpečnostním klíčem, konkrétně PGP key (Pretty Good Privacy). Ten je možné získat následujícím příkazem:

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add –
```

Poté následuje uložení repositáře a instalace

```
echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-7.x.list
```

Kibana je instalována ze zmíněného repositáře a v konfiguračním souboru se pouze zkontroluje správnost cesty k Elasticsearch.

Před instalací Logstash je nejdříve zapotřebí zkontrolovat verzi Javy. Logstash vyžaduje verzi 8 nebo 11. V případě absence Javy je třeba její instalace. Instalace poté probíhá za použití PGP klíče stejně jako u Elasticsearch.

Dalším krokem je konfigurace Logstash a to vytvořením souboru .conf v repositáři conf.d. Ten se na Linux operačním systému zpravidla nachází v adresáři `/etc/logstash`. Konfiguraci můžeme uložit do několika souborů zvlášť pro input, output a případně filter či sloučit všechny tři části do jednoho souboru.

```
input {
  beats {
    port => 5044
  }
}
filter {
}
output {
  elasticsearch {
   hosts => ["localhost:9200"]
   index => "new_index"
  }
}
```

Pro konfiguraci filtrů, prostřednictvím kterých je možné např. parsovat zprávy do námi potřebného formátu a následně je ukládat do Elasticsearch, se využívá jazyka Grok.

4.1.2 Instalace Filebeat na VM bak1 a testování sběru logových zpráv

U Filebeatu je v našem případě nutné změnit jeho konfigurační soubor filebeat.yml. Ten se stejně jako konfigurační soubory Logstash v Linux operačním systému nachází v adresáři /etc. V konfiguračním souboru je defaultně nastaven output na Elasticsearch. My však chceme naše nestrukturovaná data parsovat před uložením do Elasticsearch, tudíž před část s outputem pro Elasticsearch přidáme symbol #, tím vytvoříme komentář. Následně upravíme logstash output na IP adresu, na které je logstash nakonfigurován. Logstash defaultně naslouchá na portu 5044.

```
#------------------------ Logstash output -------------------

output.logstash:

 hosts: ["IP:5044"]
```

Dále je zapotřebí povolit Filebeat modul pro zasílání potřebných logových zpráv. Nejdříve byl testován transport systémových logových zpráv, a proto došlo k povolení modulu s názvem system. Tyto Filebeat moduly pomáhají zjednodušit sběr, parsování a vizualizaci typických logových formátů. Každý modul obsahuje jeden nebo více „filesetů“ ve kterých se nachází například Filebeat input konfigurace s defaultní cestou k logovým zprávám daného formátu. Samozřejmě záleží na operačním systému, který je využíván. Fileset obsahuje také definici uzlu příjmu pro Elasticsearch, díky kterému je Elasticsearch schopen parsovat zprávu do řádků. Dále filesety obsahují definici atributů jako je například host.id, @timestamp, _id, _index, které je možné vidět v Kibaně, či vzorové Dashboardy v Kibaně, jež je možné využít pro zobrazení logových zpráv.

```
filebeat modules enable system
```

Konkrétní cestu k logovým zprávám, které chceme sbírat, je možné upravit v následujícím konfiguračním souboru:

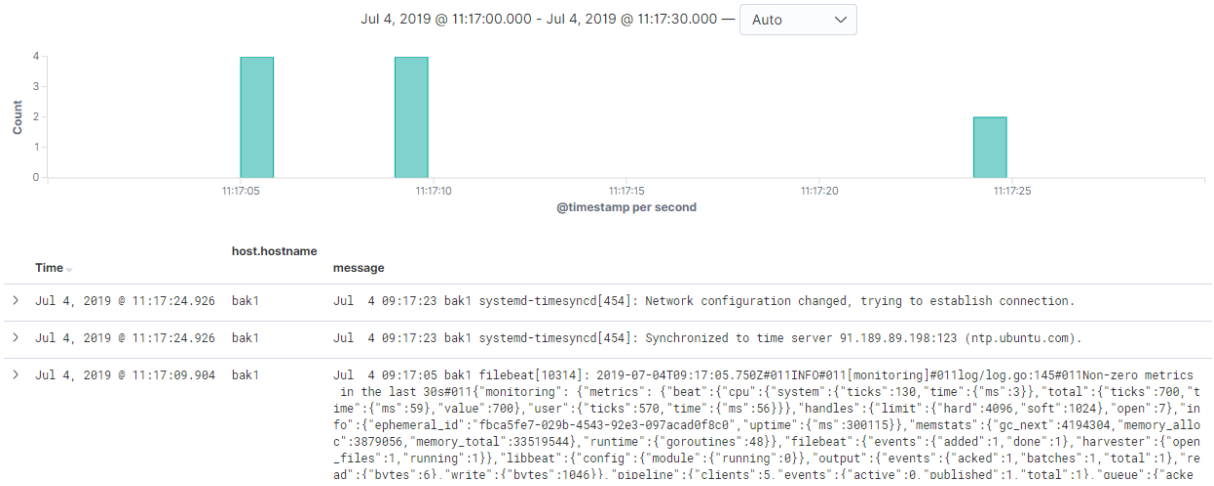
```
/etc/filebeat/modules.d/system.yml
```

Poslední věc, kterou je potřeba udělat pro spuštění našeho logovacího řešení je vytvoření indexu („databáze“) v Elasticsearch kam se logy budou posílat a ukládat.

Pro založení indexu je potřeba zajistit spojení mezi Filebeatem a Elasticsearchem, které je možné ověřit prostřednictvím telnetu. Následující příkaz na moment přeruší spojení Filebeat – Logstash a povolí spojení Filebeat – Elasticsearch a poskytne tak možnost založení indexu.

```
filebeat setup --index-management -E output.logstash.enabled=false -E
'output.elasticsearch.hosts=["elasticsearch:9200"]'
```

Jako poslední krok je třeba najít námi vytvořený index v Kibaně v Settings -> Index Patterns -> Create index pattern a pro zobrazení naších dat vybereme příslušný index v rubrice Discover v ovládacím panelu Kibany.



Obrázek 5: Webové rozhraní vizualizace Kibana

4.1.3 Nasazení open source alarmovacího software ElastAlert na bak1.

ElastAlert je open source alarmovací software od firmy Yelp. Tento software je založen na periodickém dotazování, analýze dat v Elasticsearch a generování alarmů na základě jednoduchých pravidel jako je například: Zašli e-mail v případě, že určitá hodnota má určitý počet výskytů nebo v případě růstu počtu chybových zpráv.

Elasticsearch je v určitém časovém rámci nastaveném v jedno z pravidel periodicky dotazován (defaultně každou minutu) a data jsou porovnávány se stanovenými pravidly. V případě shody je spuštěn alarm, který nám je schopen zaslat zprávu prostřednictvím e-mailu, Slacku, Telegramu, HTTP POSTU atd.

Jelikož je ElastAlert napsán v pythonu, pro jeho instalaci je zapotřebí mít nainstalován Python a jeho package management pip prostřednictvím kterého je ElastAlert instalován. Není to však jediná možnost instalace software. [23]

Je také možné získat repositář Git, který obsahuje nejnovější změny. Je vyžadována instalace modulů „setuptools=>11.3“, setup.py a elasticsearch.py.

ElasAlert adresář obsahuje soubor s pravidly jejíchž konfigurací charakterizujeme, při jaké příležitosti a jaký typ alarmu se má spustit. V pravidle musí pod atributy es_host a es_port být

uvedena cesta k Elasticsearch a jeho indexu, který má být dotazován. Abychom mohli mít přehled o dění v ElastAlert, je nutné pro něj vytvořit index v Elasticsearch, který bude využit pro ukládání metadat a informací o dotazování.

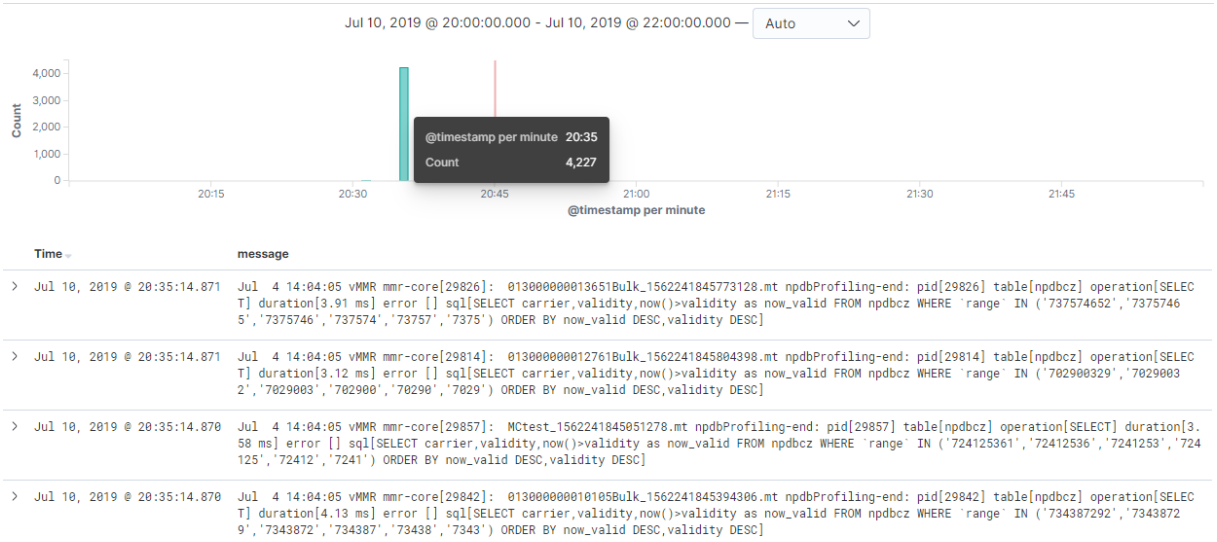
```
elastalert-create-index
```

Po otestování našeho pravidla můžeme spustit ElastAlert jako daemon proces.

```
$ python -m elastalert.elastalert --verbose --rule rule.yaml
```

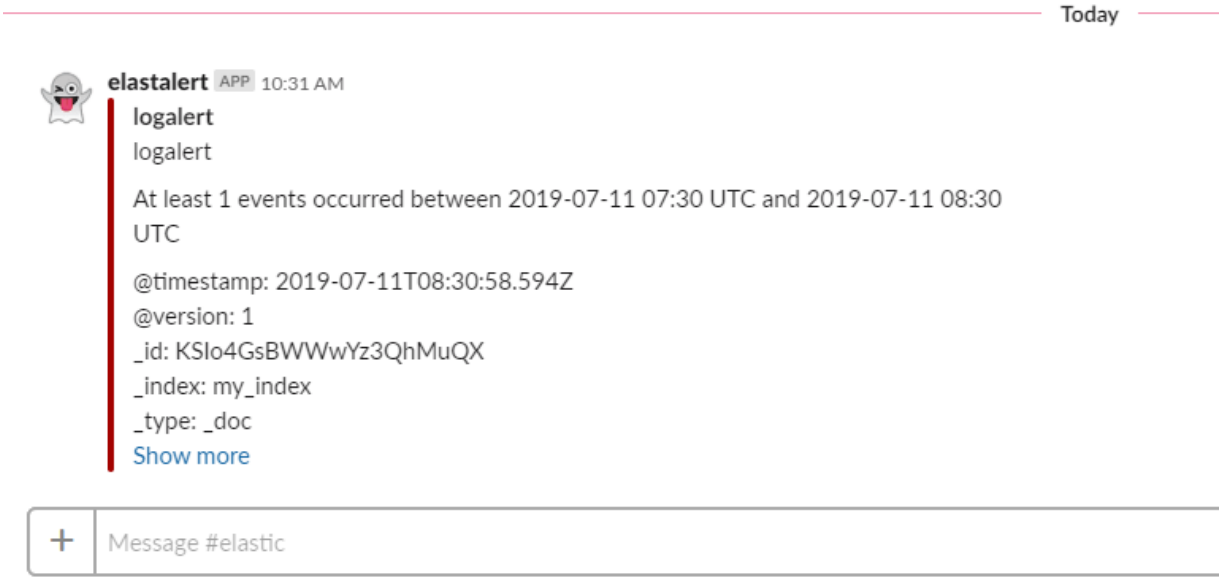
4.1.4 Nahrání textového souboru s nestrukturovanými záznamy z logových zpráv

Pro nahrání textového souboru se vzorovými daty bylo využito parsovacího software Logstash, který dokáže provést úpravu nestrukturovaného textu na námi potřebný formát. V tom se dále uloží do Elasticsearch. Při nahrávání však došlo k problému, kdy si Logstash nebyl schopen poradit s nahráním souboru na dříve Filebeatem založený index z důvodu odlišných verzí Filebeatu a Logstashe. Bylo tedy za potřebí vytvořit nový index, na který mohl být soubor nahrán. U všech software od společnosti Elastic je pro jejich bezproblémovou spolupráci zapotřebí stejná verze daných software.



Obrázek 6: Vizualizace vzorových dat z textového souboru (nestrukturované)

Dále následovalo testování možnosti vyhledávání v Elasticsearch jak ze strany console tak prostřednictvím Kibany. Testování ElastAlert na výskyt určité duration hodnoty, v případě které, se spustil alarm v podobě odeslání zprávy na Slack. Zpráva obsahuje časové údaje, ID logové zprávy s výskytem vysoké hodnoty pole duration a index, ve kterém se zpráva nachází.



Obrázek 7: Zpráva odeslaná prostřednictvím ElastAlert na Slack

4.1.5 Parsování a indexace dat v textovém souboru prostřednictvím Logstash

Pro testování práce s Logstash a Elasticsearch software bylo použito textového souboru se vzorovými logovými zprávami., které bylo zapotřebí dále zpracovávat a analyzovat.

Logstash za pomocí konfiguračního souboru s filtrem psaném v jazyce Grok pro parsování jednotlivých řádků v textovém souboru získával data z daného souboru a rozděloval je na části, které dále byly ukládány do Elasticsearch jako atributy. Pomocí těchto atributů probíhají námi potřebné operace vyhledávání a vizualizace v Kibaně – webovém rozhraní pro práci s Elasticsearch.

Formát jednoho řádku v textovém souboru:

```
Jul  4 13:56:17 vMMR mmr-core[29839]:
GtsAwegAOMTbez_1562241377271986.mt npdbProfiling-end: pid[29839]
table[npdbcz] operation[SELECT] duration[6.27 ms] error []
sql[SELECT carrier,validity,now()>validity as now_valid FROM npdbcz
WHERE `range` IN
('606339842','60633984','6063398','606339','60633','6063') ORDER BY
now_valid DESC,validity DESC]
```

Jazyk Grok je vhodný pro práci s nestrukturovanými daty a je ho možné využít pro část filter v konfiguračních souborech Logstashe. Pro parsování logové zprávy má následující syntaxi: `%{SYNTAX:SEMANTIC}.` Část, jež chceme upravit, se nachází ve složených závorkách a začíná procentem. SYNTAX je typ vzoru, který se shoduje s textem, s jimž pracujeme a SEMANTIC je tzv. identifikátor konkrétní části textu pod syntaxí. Grok filter plugin má 120 výchozích vzorů a v případě nutnosti je možné vytvořit vlastní. [24]

Obsah filtru pro parsování jednoho řádku:

```
filter {
 grok {
  match => {“message” => “%{NOTSPACE:timestamp} %{SPACE}
%{NOTSPACE:field_1} %{NOTSPACE:field_2} %{SYSLOGHOST:hostname}
%{DATA:type} %{SPACE} %{DATA:file_id} %{DATA:file_name}
%{DATA:syslog_pid} .*table\s*\[%{WORD:table}\]
.*operation\s*\[%{WORD:operation}\]
.*duration\s*\[%{NUMBER:duration:float} ms\] %{GREEDYDATA:rest}”}
 }
}
```

4.1.6 Alternativa k open-source alarmovacímu software ElastAlert

Jelikož se ukázalo, že alarmovací funkce ElastAlert nejsou pro naše potřeby dostačující, bylo zahájeno hledání jiného řešení.

První na řadu přišel zpoplatněný balíček s rozšiřujícími funkcemi od společnosti Elastic X–Pack, který je možné po dobu 30 dnů zdarma vyzkoušet. Ten mimo jiné obsahuje i alarmovací funkci jménem Watcher. Tato funkce se ovládá prostřednictvím Kibany a můžeme pomocí ní vytvářet alarmy založené na podmínkách, které jsou vyhodnocovány pomocí dotazů na data nacházející se v Elasticsearch indexech. Alarmy je možné zasílat prostřednictvím e-mail, webhook, Slack, PagerDuty a HipChat.

Na základní úrovni je ovládání Watcher funkce jednoduché a intuitivní. V Kibaně rozklikneme záložku Management, kde je odkaz na Watcher. Ten nabízí dvě možnosti vytvoření alarmu:

- Threshold alert – umožňuje nám jednoduché nastavení alarmu, který se spustí v případě překročení zadané hranice. Může se jednat například o jednoduchou podmínku jako je „pošli zprávu v případě, že hodnota daného pole bude vyšší než 5“. Po nastavení podmínky alarmu vybereme metodu zaslání zprávy a alarm uložíme.

- Advanced watch – jedná se o pokročilou tvorbu alarmu, kdy nás Watcher dokáže upozornit na cokoliv na co se můžeme dotázat. Pokročilý alarm se tvoří pomocí jazyka pro vyhledávání v Elasticsearch – Query DSL. Alarm se zpravidla skládá ze čtyř částí: schedule, query, condition a actions. První, co je potřeba definovat je schedule neboli v jakých intervalech má Watcher spouštět náš dotaz a kontrolovat Elasticsearch. Následně definujeme query část alarmu, kterou řekneme Watcher na co se má dotazovat. Condition část slouží k vytvoření podmínky pro spuštění akce, která je formulována na konci celého příkazu.

```
  "trigger": {
    "schedule": {
      "interval": "30m"
    }
  },
  "input": {
    "search": {
      "request": {
        "body": {
          "size": 0,
          "query": {
            "match_all": {}
          }
        },
        "indices": [
          "*"
        ]
      }
    }
  },
  "condition": {
    "compare": {
      "ctx.payload.hits.total": {
        "gte": 10
      }
    }
  },
  "actions": {
    "my-logging-action": {
      "logging": {
        "text": "There are {{ctx.payload.hits.total}} documents in
your index. Threshold is 10."
      }  }}} [25]
```

4.1.7 Logstash parsování dat různého typu

Pro případ zjištění počtu odeslaných, doručených a ztracených SMS zpráv z logových zpráv serveru bylo potřeba každou logovou zprávu rozložit na části, které by bylo možné snadno zobrazit ve webovém rozhraní Kibana a vyhledat potřebnou informaci. Každá logová zpráva je jiného tvaru, a proto je potřeba vytvořit filter, který dokáže rozeznat jakýkoliv formát příchozího logu a správně ho rozložit do logických celků.

K tomuto účelu bylo využito Logstash dissect filter plugin. Tento plugin nám umožňuje rozdělit každý řádek na části, se kterými je dále možné pracovat.

```
Jul 23 09:24:16 mmr mmr-core[5147]:  Aweg3AOMTs_1563866656876839.mt
NPDB::query(111=npdbcz,123456789): no match [2.05 ms]
Jul 23 09:24:16 mmr mmr-core[5147]:  Aweg3AOMTs_1563866656876839.mt
GetDestination(2,+31112223344,1,1,0):    MATCH:205    "PR+420603"
(0.031ms prefix match, 2.261ms NPDB)
```

Tyto dva řádky mají společný tvar pouze pro datum, název hosta a ID zprávy, zbytek je odlišný. Díky Logstash dissect plugin můžeme využít tohoto totožného tvaru a rozdělit tak zprávu na čtyři části – datum, host, process a zbytek zprávy.

```
dissect  {  mapping  =>  {  "message"  =>  "%{[@metadata][ts]->}
%{+[@metadata][ts]}   %{+[@metadata][ts]}   %{host}   %{process}:
%{restOfLine}" } }
date { match => [ "[@metadata][ts]", "MMM dd HH:mm:ss" ] }
```

Následně se využije parsovacího jazyka Grok pro přístup do jednotlivých celků a definici parsingu, která nám umožní získat na výstupu tvar, který potřebujeme.

```
    grok {
    match => {
        "restOfLine" => [
            "^Processing file \[%{NOTSPACE:messageId}\]",
            "^ %{NOTSPACE:messageId}"
        ]
    }
    }

    grok {

        pattern_definitions => {"SOMETEXT" => "[[:alnum:]]+"}
                match => {
                  "messageId" => [

"^%{SOMETEXT:text}_%{INT:num1}\.%{INT:num2}\.%{DATA:suffix}$",

"^%{SOMETEXT:text}_%{INT:num1}\.%{DATA:suffix}$"
                     ]

                }
    }
```

Zajímavá funkce jazya Grok je pattern_definitions, které bylo využito k rozdělení `Aweg3AOMTs_1563866656876839.mt` na části [text]_[num1]_[suffix]. Na začátku definice tohoto vzoru je definice celého objektu jako SOMETEXT a všechny znaky, jak číselné, tak písemné (alnum). Dalším krokem je nalezení messageId, které je dvakrát definováno předem v processing file, z důvodu dvou typů těchto messageId.

```
Aweg3AOMTs_1563866656876839.mt
Aweg3AOMTs_1563866656876839.0.dn
```

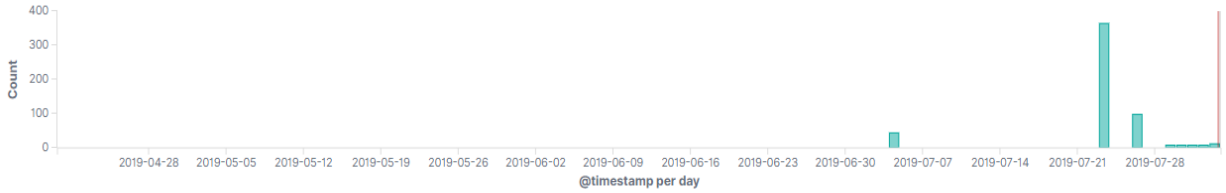
Dále následuje použití vzoru pattern_definitions, který byl stanoven výše

```
"^%{SOMETEXT:text}_%{INT:num1}\.%{DATA:suffix}$"
"^%{SOMETEXT:text}_%{INT:num1}\.%{INT:num2}\.%{DATA:suffix}$"
```

Logstash umožňuje do konfiguračního souboru vložit podmínku jako například:

```
if "realtime" in [message] {
grok { match => { "restOfLine" => "^ %{NOTSPACE} %{DATA:type}
realtime: %{NUMBER:realtime}" } }
}
```

Tahle podmínka nám říká, že v případě výskytu slova realtime v naší zprávě, porovnej v celku restOfLine následující formát a pokud je výskyt v tomto formátu, tak vytvoř oddělené pole pro hodnotu realtime.



	Time	text	num1	num2	suffix	realtime
›	Jul 26, 2019 @ 19:44:14.000	EuroSMSBulk	1563867854325555	-	mt	4.961
›	Jul 26, 2019 @ 19:44:18.000	EuroSMSBulk	1563867854323333	0	dn	1.84
›	Jul 23, 2019 @ 11:24:16.000	Aweg3AOMTs	1563866656871111	-	mt	5.684
›	Jul 23, 2019 @ 11:24:18.000	Aweg3AOMTs	1563866656872222	0	dn	1.382

Obrázek 8: Vytvoření nového atributu realtime na základě podmínky

4.1.8 Vyhledávání prostřednictvím Elasticsearch query DSL

K definici dotazů pro vyhledávání prvků dokumentů v indexech Elasticsearch využívá jazyka Query DSL (Domain Specific Language) založeného na JSON. Query DSL nám dovoluje vytvářet dotazy na určitou hodnotu v určitém poli. Mezi tyto dotazy řadíme tzv. Leaf query clauses match, term a range příkazy. Dalším druhem dotazů jsou složené dotazy jako například bool či dis_max do kterých jsou zabaleny příkazy z Leaf query.

Jednou z nejužitečnějších vyhledávacích funkcí Elasticsearch je agregace. Agregace sdružují informace na základě našich požadavků a dělí se na tři druhy. Můžeme je přirovnat k příkazu GROUP BY v relačních databázích.

Metric aggregation – je nejjednodušší typ agregací a dovoluje nám provádět různé logické operace nad námi určenými daty.

Bucket aggregation – vytvoří tzv. buckety neboli množiny dokumentů. Každý bucket je definován kritérii, které musí dokument splňovat, aby se stal součástí daného bucketu. Na rozdíl od metrických agregací, bucket agregace nám dovolují vytvářet sub-agregace. Vkládáním jedné agregace do druhé vzniká situace, kdy vnořená agregace uplatňuje svá pravidla v rámci rodičovské agregace. [26]

Pipeline aggregation – tyto agregace pracují s výstupy jiných agregací. Pro svůj input využívají parametr `buckets_path`, který definuje cestu k agregaci, nad kterou chceme provést námi zvolenou operaci. Pipeline agregace nemohou mít sub agregace, jako je tomu u bucket agregací. Mohou však odkazovat na jinou pipeline agregaci v `buckets_path`, což nám dovoluje tzv. řetězení agregací. [27]

Každá logová zpráva má svůj sufix, jehož tvar má dvě podoby, buď mt nebo dn. Query DSL pro vyhledání všech zpráv v indexu my_index3, které mají oba sufixy jak dn tak mt by mohl vypadat následovně:

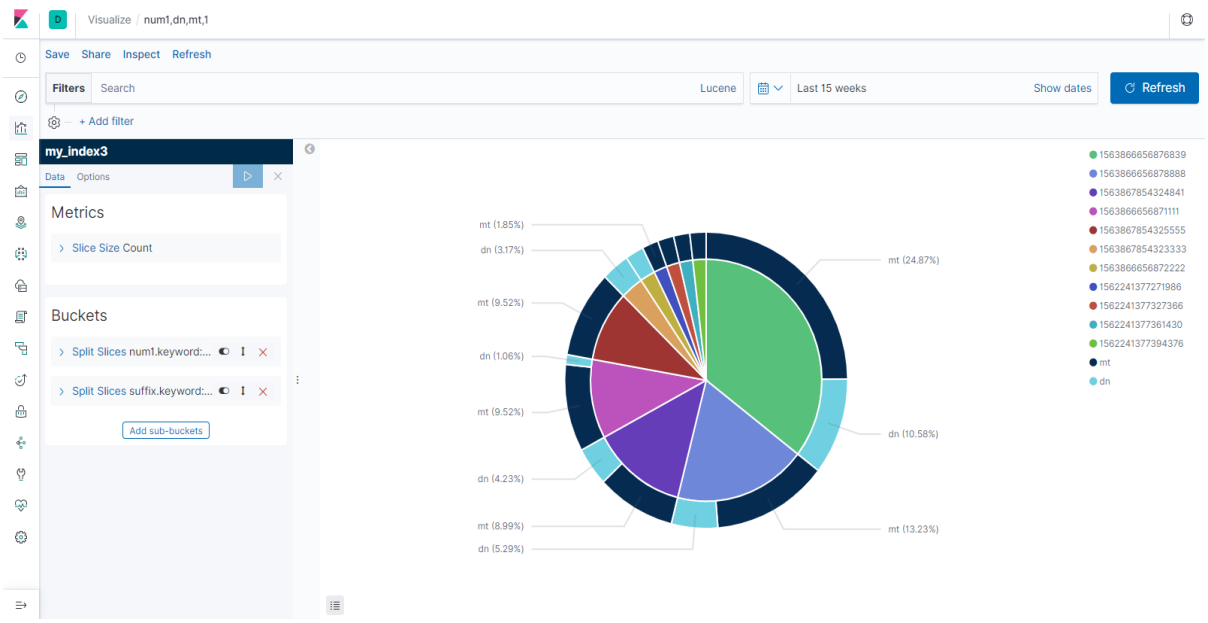
```
GET /my_index3/_search
{
  "size": 0,
  "aggs": {
    "num1": {
      "terms": {
        "field": "num1.keyword",
        "order": {
          "_count": "desc"
        }
      },
      "aggs": {
        "count_of_suffix": {
          "cardinality": {
            "field": "suffix.keyword"
          }
        },
        "my_filter": {
          "bucket_selector": {
            "buckets_path": {
              "count_of_suffix": "count_of_suffix"
            },
            "script": "params.count_of_suffix == 2"
          }
        }
      }
    }
  }
}
```

4.1.9 Vizualizace dat prostřednictvím Kibana

Za předpokladu, že logová zpráva vypovídající o odeslané SMS zprávě má určité ID a sufix ve tvaru mt. Logová zpráva se sufixem dn nám říká o úspěšném doručení dané zprávy. Na základě toho, dokážeme pomocí dotazu výše zjistit počet úspěšně doručených zpráv.

Cílem tohoto dotazu však není pouze zjistit daný počet úspěšně doručených zpráv, ale poskytnout vizualizaci v podobě koláčového grafu. Bohužel naše přesvědčení o tom, že je vizualizace založená na Query DSL, bylo mylné.

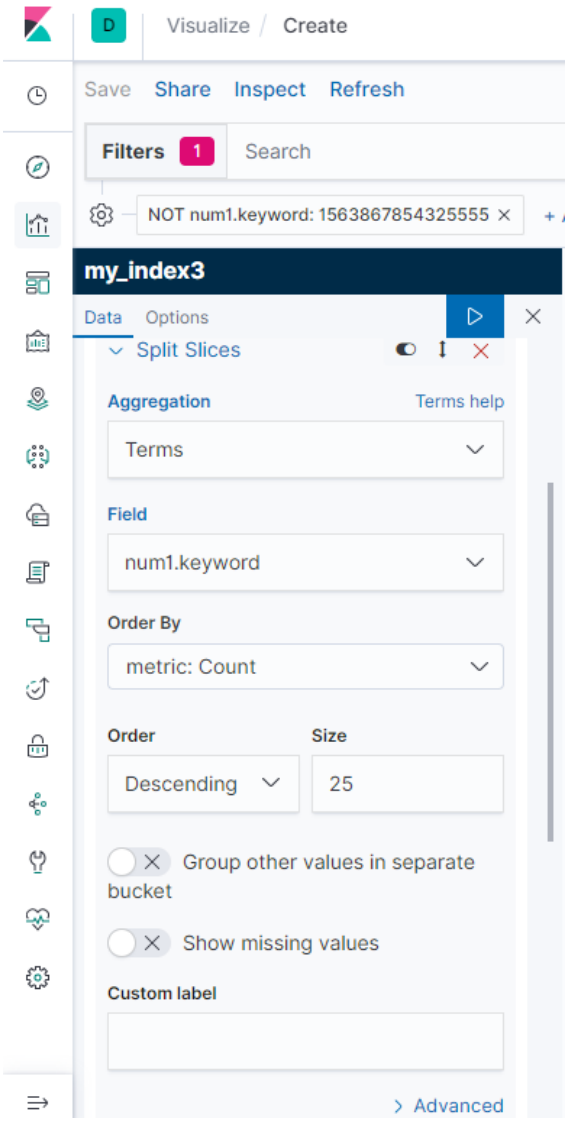
Vizualizace dat probíhá prostřednictvím webového rozhraní Kibana, kde v ovládacím panelu v záložce Visualization vybereme index, který chceme použít pro vizualizaci a typ vizualizačního grafu. Dále Kibana nabízí jednoduché metrické agregace a sub-agregace na vytvoření součtů, průměrů, počtů atd. Ve snaze zobrazení úspěšně doručených zpráv byl vygenerován následující koláčový graf.



Obrázek 9: Koláčový graf vytvořen ve webovém rozhraní Kibana

Graf ve vnitřní části zobrazuje id čísla každé logové zprávy, která byla přidána do indexu my_index3. Vnější část grafu ukazuje sufix, který každá zpráva obsahuje. Pokud jsou číslu z vnitřní části grafu přiřazeny obě barvy z vnější části grafu, zpráva byla v pořádku doručena. Samozřejmě v případě tisíce různých zpráv denně tato vizualizace nemá smysl.

Zde je zobrazena jedna z agregací, kterou lze jednoduše vytvořit v Kibana V pokročilém nastavení agregace je dále možné přidání „omezení“ polí include, exclude jež nám říkají, která pole mají být zahrnuta do agregace či nikoliv. V pokročilém nastavení agregace nalezneme i pole pro JSON script. Ten pracuje tak, že nám dovolí přizpůsobit požadavek posílaný do Elasticsearch pomocí přidání parametrů v konfiguraci agregace. Parametr je definován jako inline script, který je vykonáván nad každým dokumentem, a ne nad výsledkem agregace.



Obrázek 10: Koláčový graf vytvořen ve webovém rozhraní Kibana

NoSQL databáze jsou dnes často rozšířené o funkcionality SQL jazyka, kterého je možné využít k vytváření dotazů. Elasticsearch poskytuje možnost použití SQL dotazování na velmi vysoké úrovni ve své placené verzi. Jelikož byla zaměřena pozornost zejména na využití open source systémů, nebylo možné těchto možností plně využít. Nicméně i v bezplatné licenci Elasticsearch disponuje ohraničenou verzí tohoto SQL dotazování.

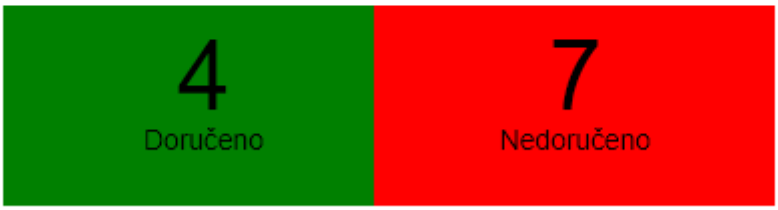
Jednou z dalších zajímavých funkcionalit, které nám Elastic nabízí je tzv. dataframe (datový rámec). Tato funkce byla vytvořena na základě myšlenky o entity-centric indexes Marka Harwooda, který je členem Elastic teamu a tuto myšlenku vysvětluje na příkladu sledování návštěvnosti na webu. Výše zmíněné agregace, které využíváme v jazyce DSL Query pro vyhledávání v databázích Elasticsearch, nám nedokážou říct nic moc víc o tom, co samotná logová zpráva neobsahuje. Pokud by nás zajímalo, kolik času konkrétní uživatel strávil na naší webové stránce, agregace nám k tomu nestačí. Je potřeba zaznamenat logovou zprávu, která obsahuje čas příchodu uživatele na stránku a čas, ve kterém uživatel stránku opustil. Tyto časové informace zároveň musí být svázány s identifikačním číslem uživatele. Entity-centric indexing nám pomáhá vyřešit daný problém ukládáním proudu logových zpráv do separátních indexů, kde jeden index je přiřazen jednomu uživateli. [28] Datové rámce fungují tak, že nám k vytvoří nadstavbu indexu (rámec), jež je založena na definici dotazu, který chceme položit a další jeho strukturu, například s jakou další informací bychom dotaz chtěli spojit. Tím dochází k zjednodušení další práce s těmito daty, protože už jsou strukturované v určitém rámci.

Datový rámec byl využit k agregování identifikačního čísla zprávy a sufixu, který říká, zda logová zpráva vypovídá o odeslání nebo o doručení dané zprávy. Pro vizualizaci doručených a nedoručených zpráv byla také testována funkce Canvas. Jak název napovídá, jedná se o prostředí pro tvorbu různých vizuálních interpretací dat. Canvas se nachází v Kibaně a jeho výhodou je, že umožňuje pokládání dotazů v SQL jazyce a na základě těchto dotazů je schopen zobrazit grafy, tabulky a jiné vizualizační elementy.

```
filters

| essql

  query="SELECT num1.keyword as ahoj FROM \"dataframelast\" where suffix.keyword in ('mt','dn') group by num1.keyword having count (distinct suffix.keyword) = 1"

| math "unique(ahoj)"

| metric "Nedoručeno"

  metricFont={font size=48 family="'Open Sans', Helvetica, Arial, sans-serif" color="#000000" align="center" lHeight=48}

  labelFont={font size=14 family="'Open Sans', Helvetica, Arial, sans-serif" color="#000000" align="center"}

| render containerStyle={containerStyle padding="12px" backgroundColor="#FF0000"}


filters

| essql

  query="SELECT num1.keyword as ahoj FROM \"dataframelast\" where suffix.keyword in ('mt','dn') group by num1.keyword having count (distinct suffix.keyword) = 2"

| math "unique(ahoj)"

| metric "Doručeno"

  metricFont={font size=48 family="'Open Sans', Helvetica, Arial, sans-serif" color="#000000" align="center" lHeight=48}

  labelFont={font size=14 family="'Open Sans', Helvetica, Arial, sans-serif" color="#000000" align="center"}

| render containerStyle={containerStyle padding="12px" backgroundColor="#008000"}
```

Dotazem zobrazeným výše bylo dosaženo vizualizace, která jen z části vyhovuje požadavkům, jež byli uvedeny jako cílové. Umožňuje nám však téměř v reálném čase sledovat počet doručených a nedoručených SMS zpráv.

Doruceno	Nedoruceno
1563866656871111	1562241377271986
1563866656876839	1562241377327366
1563866656878888	1562241377361430
1563867854324841	1562241377394376
	1563866656872222
	1563867854323333
	1563867854325555



Obrázek 11: Vizualizace doručených a nedoručených zpráv v Canvas

4.1.10 Shrnutí

Ačkoliv představené způsoby řešení nesplňuji všechny požadované cíle uvedené v úvodu tohoto projektu, seznamuje nás se základními funkcemi softwarů z řady Elastic Stack. Ukazuje návrhy pro vytváření alarmů, parsování příchozích logových zpráv a možnosti následné vizualizace dat. Cílem zadání bylo zobrazení doručených, nedoručených a ztracených zpráv v koláčovém grafu, jenž by nám poskytoval procentuální rozdělení těchto tří druhů zpráv s podmínkou výskytu logové zprávy s informací o doručení do 72 hodin.

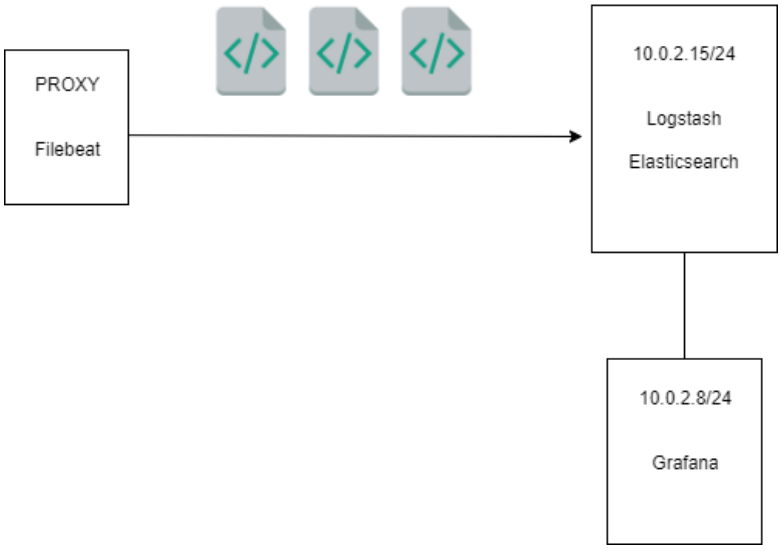
V relační databázi, kde se pro dotazování nad daty využívá jazyk NoSQL, by dotaz mohl vypadat následovně:

```
SELECT num1
FROM table4
WHERE suffix IN ('mt', 'dn')
GROUP BY num1
HAVING MIN(CASE WHEN suffix = 'dn' THEN date END) < MAX(CASE WHEN
suffix = 'mt' THEN date END) + INTERVAL 72 HOUR;
```

Přeformulovat tento dotaz do dotazovacího jazyku, který využívá Elasticsearch (Query DSL) bohužel nebylo úspěšné.

4.2 Analýza stavových kódů

Tento projekt má za cíl transport logových zpráv z proxy serveru do NoSQL databáze Elasticsearch. Přenos bude zprostředkován Filebeatem a Logstashem, který data strukturalizuje. Následuje indexace přenesených dat a jejich vizualizace prostřednictvím software Grafana. Logové zprávy z proxy serveru obsahují tzv. stavové kódy. Každý stavový kód má svůj význam. Tento význam bude monitorován, a pomůže nám sledovat stav procesů jenž probíhají da daném serveru.



Obrázek 12: Vizualizace doručených a nedoručených zpráv v Canvas

4.2.1 Elasticsearch – indexace a parsování příchozích logových zpráv

Prvním krokem na cestě k vizualizaci stavových kódů je vhodná indexace zpráv v Elasticsearch.
Příklad příchozí logové zprávy:

```
111.222.3.11 - - [15/Oct/2019:08:40:49 +0200] "asaa /ssss/asf sss/saaa" 200 12833 "-" "sada(asd; fffff sff ff f asdasd) ssssss/6sadasf (ddddd, sssssss) ddd/sdasds/2222"
```

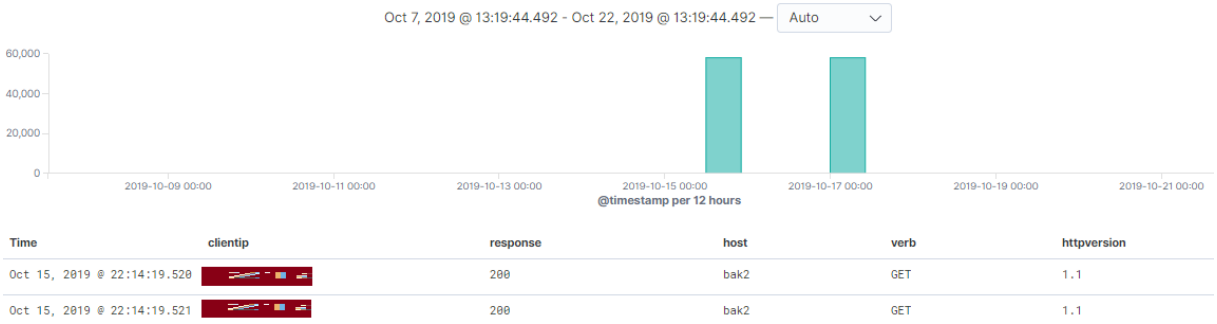
Nás zajímá především IP adresa, ze které bychom dále byli schopni určit polohu zdroje stavového kódu a jeho typ 200, 403, 500 atd.
Jako první způsob parsování těchto zpráv byl testován podporovaný Logstashem parsovací jazyk GROK, který prostřednictvím filtrů ve svém konfiguračním souboru dokáže převést řádek do požadované struktury. Pomocí GROK Debuggeru v Kibaně byl navržen filtr na rozklad řádku do několika skupin. Bohužel každá zpráva je jinak dlouhá a má jiný formát. Proto parsování řádku slovo od slova bylo ihned zamítnuto. Logstash má však sadu vzorů, které významně zjednoduší rozklad logové zprávy na potřebné části.

```
filter {
  grok { match => { "message" => [ "%{HTTPD_COMBINEDLOG}",
      "%{HTTPD_COMMONLOG}" ] }
  }
}
```

Filtr obsahuje vzory HTTPD_COMBINEDLOG a HTTPD_COMMONLOG, které jsou schopny vyhledat v každém řádku, mimo jiné následující atributy:

```
t clientip
t response
 @timestamp
t host
t httpversion
```

Tyto atributy můžeme vidět a dále s nimi pracovat v grafickém prostředí Kibana.

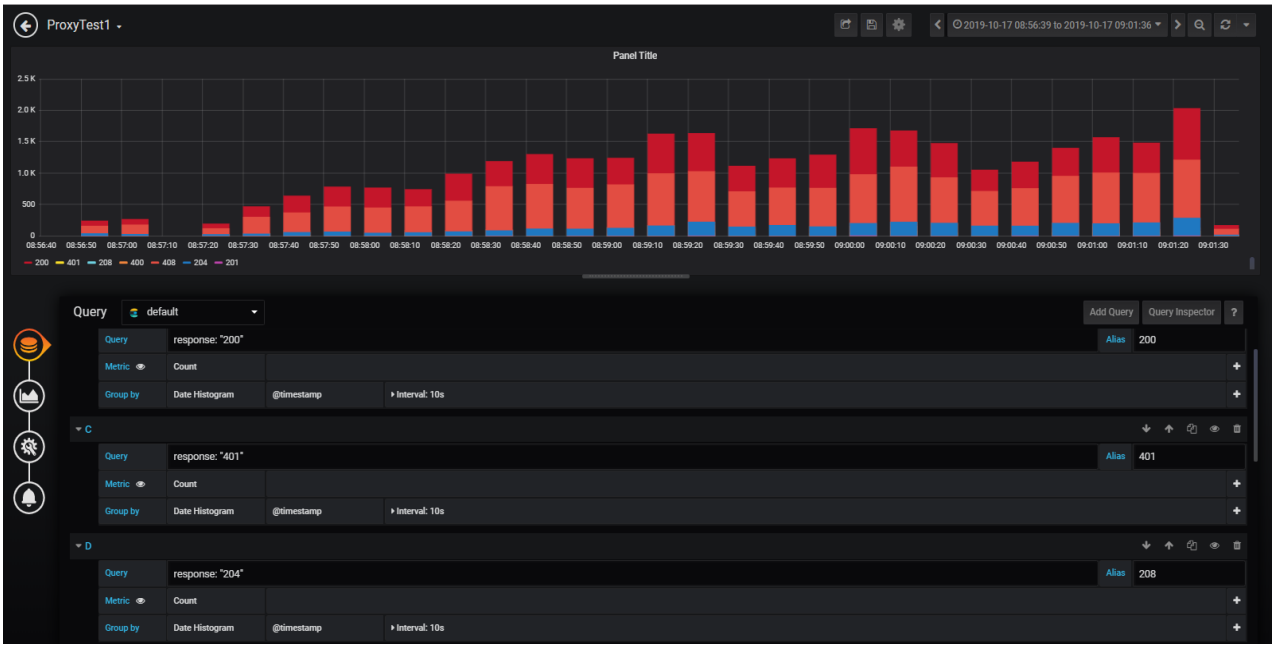


Obrázek 13: Zobrazení rozparsovaných logových zpráv v Kibana

4.2.2 Grafana – vizualizace stavových kódů pomocí histogramu

Ze všeho nejdřív je nutné v Grafaně nastavit Data Source na odpovídající Elasticsearch index a následně vytvořit nový dashboard s vizualizací typu Graph.

Dotazy jsou nastaveny tak, aby vyhledali námi požadované stavové kódy v deseti sekundových intervalech. Na ose Y je zobrazen počet výskytů dané zprávy a na ose X časové pásmo.



Obrázek 14: Vizualizace stavových kódů v Grafana

Orientaci v grafu nám zjednoduší vytvoření jednoho dotazu s využitím funkce „Group by“ podle názvu pole, které máme nadefinované v Elasticsearch, a tím je response.keyword. Je

důležité, aby pole bylo typu keyword a bylo ho možné najít v rozbalovacím menu. Dalším nastavením dotazu pouze přidáme Date Histogram podle časového údaje v poli @timestamp a intervalem, který nám vyhovuje.

4.2.3 Elasticsearch – geolokace zdrojů zpráv pomocí IP adres

Lokalizace IP adres je možno vyřešit prostřednictvím Logstash, který disponuje pluginem Geoip plugin. Ten je schopen přidat informaci o geografické poloze IP adresy na základě Maxmind GeoLite2 veřejné databáze, která obsahuje zeměpisnou šířku a délku konkrétní IP adresy.

Pro použití této funkce bylo zapotřebí změnit obsah Logstash konfiguračního souboru na:

```
filter {

  grok { match => { "message" => "%{COMBINEDAPACHELOG}" } }

    geoip { source => "clientip" }

}
```

Bohužel to není tak jednoduché, jak to na první pohled může vypadat. Tento filtr je schopen pouze získat geografické údaje o IP adrese z databáze. Ke správnému uložení do Elasticsearch je zapotřebí „přemapovat“ index, do kterého se dané informace chystáme uložit. Důvodem je, že defaultní cíl pro geoip filter, který je použit v konfiguračním souboru Logstash, se jmenuje geoip. Toto defaultní nastavení pro indexy spojuje základní logstash -* index s geo_point, který sám Logstash nedokáže definovat.
Je tedy zapotřebí změnit “mapping” našeho indexu z defaultního na náš vlastní a vytvořit atribut (field), který bude datového typu “geo_point” a Logstash ho dokáže identifikovat. Do tohoto pole se poté uloží informace o geografické poloze IP adresy.

Základní kód pro vytvoření indexu s takovým mapováním je dostupný v dokumentaci Elasticu [29] a vypadá následovně

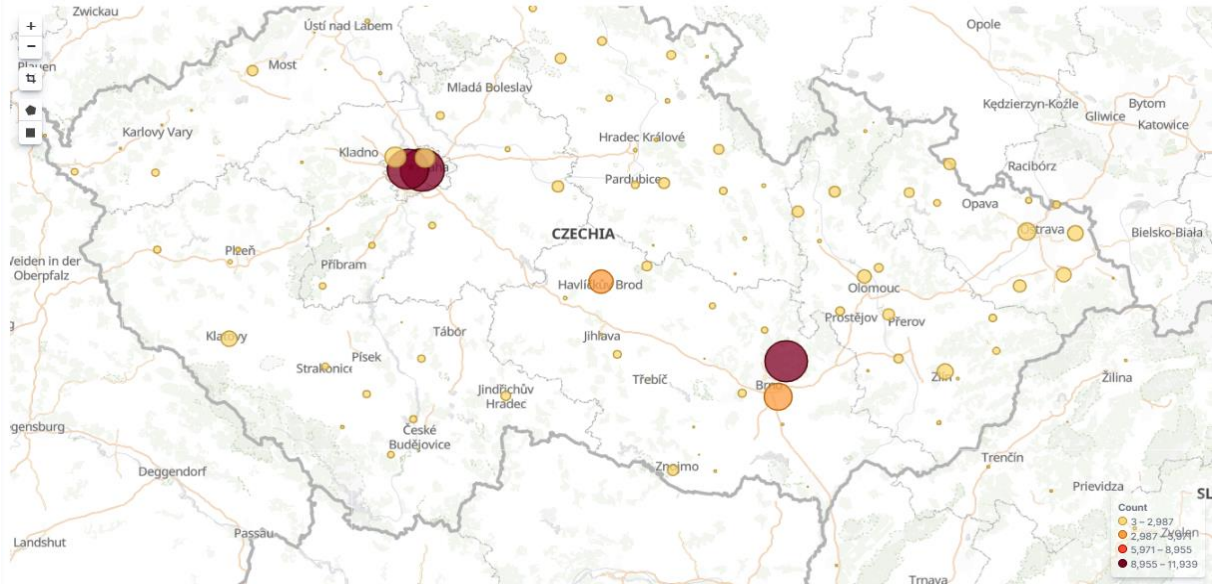
```
PUT my_index
  {
    "mappings": {
      "properties": {
        "location": {
          "type": "geo_point"
        }
      }
    }
 }
```

Z kódu je patrné, že pole location je nastaveno na datový typ geo_point. Bohužel Logstash ho po tomto nastavení stále není schopen identifikovat a vložit do něho geografické údaje.

Problém je v tom, že je zapotřebí vytvořit atribut/field s předponou geoip.location a datovým typem geo_point. Tak bude Logstash schopen spojit své pole s geografickými údaji s polem, které máme připraveno v Elasticsearch indexu. Je také nutné vytvořit pole pro zeměpisné délky a šířky typu float, jelikož jsou defaultně vytvářeny jako String.

Následující kód ukazuje vytvoření indexu s patřičnou konfigurací. Pro nás je důležitá zejména část properties.

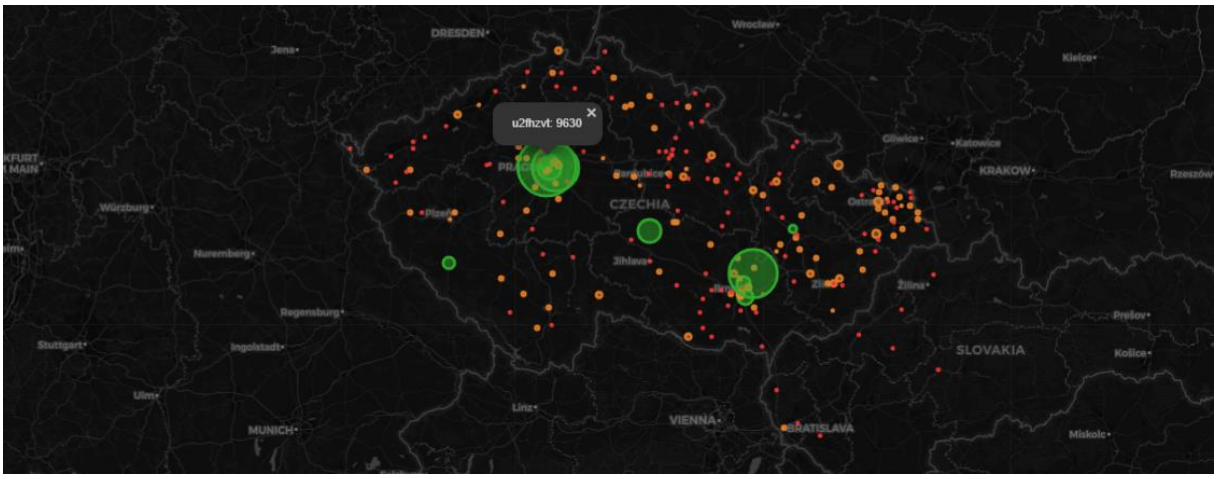
```
PUT /my_index5
{
  "mappings" : {
    "dynamic_templates" : [ {
      "message_field" : {
        "path_match" : "message",
        "match_mapping_type" : "string",
        "mapping" : {
          "type" : "text",
          "norms" : false
        }
      }
    }, {
      "string_fields" : {
        "match" : "*",
        "match_mapping_type" : "string",
        "mapping" : {
          "type" : "text", "norms" : false,
          "fields" : {
            "keyword" : { "type": "keyword", "ignore_above": 256 }
          }
        }
      }
    } ],
    "properties" : {
      "@timestamp": { "type": "date"},
      "@version": { "type": "keyword"},
      "geoip"  : {
        "dynamic": true,
        "properties" : {
          "ip": { "type": "ip" },
          "location" : { "type" : "geo_point" },
          "latitude" : { "type" : "half_float" },
          "longitude" : { "type" : "half_float" }
        }
      }
    }
  }
}
```



Obrázek 15: Zobrazení zdrojů stavových kódů v Kibana

4.2.4 Grafana – geolokace zdrojů zpráv pomocí IP adres

Pro zobrazení stejné mapy v Grafana je zapotřebí implementace Grafana pluginu World map, který je k dispozici na platformě Grafana, a nastavit požadované parametry.



Obrázek 16: Zobrazení zdrojů stavových kódů v Grafana

Při nastavení Query byl vybrán Metric Count, Group by Geo Hash Grid na pole geoip.location, které je typu geo_point. Dále byla nastavena přesnost od 1 do 7, kdy 7 je nejvyšší možná hodnota.

Při konfiguraci Vizualizace je v první řadě potřeba nastavit Map Data Options/Location Data na geohash a následně vybrat naše nadefinované pole geoip.location jako geo_point/geohash Field ve Field Mapping. Metric Field a Treshold Options dále můžeme vybrat dle našich preferencí.

4.2.5 Shrnutí

Při testování bylo využito příkladů logových zpráv pouze jednoho formátu pocházejícího pouze z jednoho zdroje. Přidání dalších zdrojů stavových kódů mělo za následek neschopnost filtru využívajícího jednoduchého vzoru `"%{COMBINEDAPACHELOG}"` najít shodu v příchozích řádcích logových zpráv. Bylo tedy nutné vytvoření customizovaných filtrů, které budou schopny správně parsovat veškerá příchozí data.

```
filter {
  grok {
    match => { "message" => "%{IPORHOST:clientip} %{HTTPDUSER:ident}
%{USER:auth} \[%{HTTPDATE:timestamp}\] \[%{NOTSPACE:referrer}\]
\"(?:%{WORD:verb} %{NOTSPACE:request}(?:
HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})\"
%{NUMBER:response} (?:%{NUMBER:bytes}|-)"
    }
  }
grok {
  match => { "message" => "%{HTTPDUSER:ident} %{USER:auth}
\[%{HTTPDATE:timestamp}\] \"%{DATA:zprava}\" (%{NOTSPACE:bordel})
\"%{DATA:buhvi}\" \(h=%{NOTSPACE:referrer} %{GREEDYDATA:rest}"
  }
}
  grok{
    match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
%{HOSTNAME:host} %{HTTPDUSER:ident}\[%{INT:num}\]:
%{IPORHOST:clientip} - - \[%{HTTPDATE:timestamp}\] \"(?:%{WORD:verb}
%{NOTSPACE:request}(?:
HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})\"
%{NUMBER:response} (?:%{NUMBER:bytes}|-) (%{NOTSPACE:bordel})
\"(?:%{DATA:nevim})\" \(h=%{NOTSPACE:referrer} %{GREEDYDATA:rest}"
    }
  }
```

```
  grok {
    match => { "message" => "%{IPORHOST:clientip} - -
\[%{HTTPDATE:timestamp}\] \"(?:%{WORD:verb} %{NOTSPACE:request}(?:
HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})\"
%{NUMBER:response} (?:%{NUMBER:bytes}|-)"
    }
  }
  grok {
    match => { "message" => "%{IPORHOST:clientip} %{HTTPDUSER:ident}
%{USER:auth} \[%{HTTPDATE:timestamp}\] \"(?:%{WORD:verb}
%{NOTSPACE:request}(?:
HTTP/%{NUMBER:httpversion})?|%{DATA:rawrequest})\"
%{NUMBER:response} (?:%{NUMBER:bytes}|-) (%{NOTSPACE:bordel})
\"%{NOTSPACE:first} %{NOTSPACE:second} %{NOTSPACE:version1}
\(h=%{NOTSPACE:referrer} %{GREEDYDATA:rest}"
    }
 }
  grok {
    match => { "referrer" =>
"%{WORD:protocol}://%{WORD:domain1}.%{WORD:domain2}.%{WORD:domain3}:
%{INT:port}"
    }
  }
  geoip { source => "clientip" }
}
```

4.3 Analýza aktivity uživatelů databáze

Cílem tohoto projektu je zajistit monitorování aktivity uživatelů databází MySQL a MariaDB Monitoring by měl poskytnout informace, jako například který uživatel je momentálně připojen a k jaké databázi, graf nejaktivnějších uživatelů a metrické statistiky databáze jako např. QPS (Queries per seconds), počet aktivních vláken databáze atd.

Obrázek 17: Schéma systému pro analýzu aktivity uživatele databáze

4.3.1 Konfigurace MariaDB databáze

Prvním krokem byla instalace a konfigurace open-source databáze MariaDB na testovací virtuální PC. Nejdříve proběhlo nastavení ukládání slow_query_log, což jsou logové zprávy o dotazech na databázi, které trvaly déle než určitý časový limit. Nastavení slow_query_log je na linuxových operačních systémech možné v adresáři `/etc/mysql/mariadb.cnf` nebo `/etc/mysql/my.cnf`. Povolení ukládání slow_query_logů bylo dosaženo přidáním následujícího kódu pod sekci mysqld:

```
slow_query_log = 1
log-slow-queries = /var/log/mysql-slow.log
long_query_time = 2
```

Uvedený adresář je nutné založit adresář a zpřístupnit ho mysql:

```
touch /var/log/mysql-slow.log
# chown mysql:mysql /var/log/mysql-slow.log
```

Po restartování servisů MariaDB a MySQL bylo dosaženo ukládání slow_query_logů do námi vytvořeného adresáře `/var/log/mysql-slow.log` v následující podobě:

```
# Time: 2019-11-12T08:51:04.006781Z
# User@Host: vlad[vlad] @ localhost []  Id:     2
#   Query_time:   0.000621     Lock_time:   0.000135   Rows_sent:   2
Rows_examined: 2
SET timestamp=1573548664;
select * from Persons;
```

Z obsahu této logové zprávy můžeme získat informace nejenom o tom, který uživatel a v jakém čase byl aktivní, ale i to, jak dlouho trvalo vyřízení položeného dotazu či samotný dotaz.

4.3.2 Filebeat, Elasticsearch a Grafana konfigurace

Filebeat disponuje modulem pro sběr a částečný parsing slow logů a error logů. Tento modul je možné aktivovat příkazem `filebeat modules enable mysql`. Modul vyžaduje MySQL verzi 5.5 a novější a MariaDB 10.1 a vyšší, Percona 5.7 a vyšší.

V konfiguračním souboru `/etc/filebeat/modules.d/mysql.yml` je pak možné nastavit konkrétní cestu k adresářům, ze kterých má být sběr dat prováděn.

Samotný modul nám však není schopen zajistit parsing námi požadovaných dat, a proto je zapotřebí implementace Logstashe jako mezičlánku pro parsing a filtorvání informací, které nás ze slow_query_logu zajímají. Konfigurace Filebeatu by tak měla obsahovat Logstash v sekci output.

Jelikož slow_query_log není tvořen jedním řádkem jako je tomu zvykem u většiny aplikačních logů, je zapotřebí před odesláním zprávy do Logstash nastavit rozpoznávání víceřádkového textu Filebeatem. Ten disponuje vzorem pro víceřádkový text s názvem multiline.pattern, který je možné nastavit v konfiguračním souboru filebeat.yml v sekci filebeat.inputs.

```
multiline.pattern: '^#'
multiline.negate: true
multiline.match: after
```

- multiline.pattern je symbol, na základě kterého, je rozpoznáván nový řádek zprávy.

- multiline.negate slouží pro případnou negaci multiline.patternu.

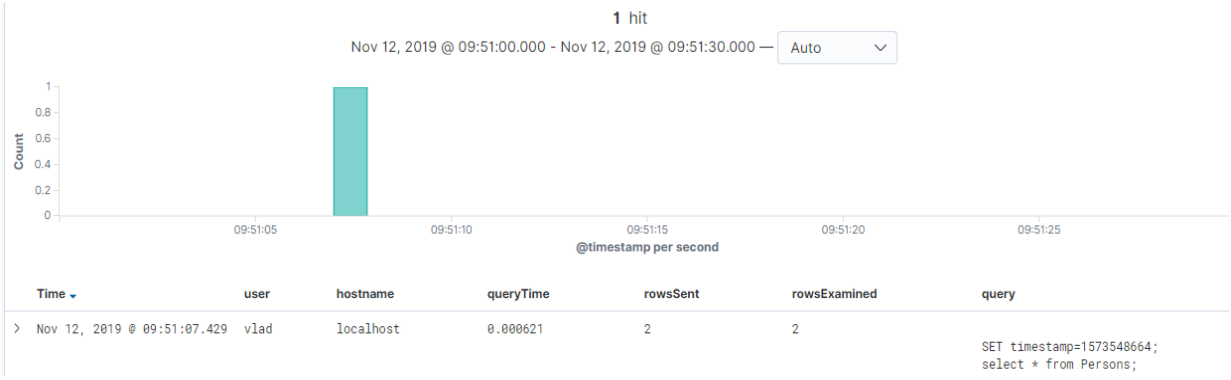
- multiline.match říká, zda spojování řádků probíhá po výskytu symbolu pro spojování nebo před. Je závislé na hodnotě multiline.negate.

4.3.3 Logstash slow_query_log filter

Pokud je rozpoznávání víceřádkového textu připraveno a Logstash ví, že k němu přichází víceřádkový obsah, je možné přistoupit ke konfiguraci Logstash filteru pro parsing příchozí zprávy a indexace do Elasticsearch ve formátu, který je preferován.

Logstash filter by mohl vypadat následovně:

```
filter {
    grok {
        break_on_match => false
        match => {
            "message" => [
                "Time: %{NOTSPACE:mysqlTime}",
                "User@Host: %{USER:user}\[%{USER}\] @
%{HOSTNAME:hostname}",
                "Query_time: %{NUMBER:queryTime:float}",
                "Rows_sent: %{NUMBER:rowsSent:int}",
                "Rows_examined:
%{NUMBER:rowsExamined:int}%{GREEDYDATA:query}"
            ]
        }
    }

}
```



Obrázek 18: Zobrazení obsahu slow_query_log v Kibana

Z důvodu velkého množství databází a vysoké aktivity je však toto řešení velmi náročné jak na výkon, tak na úložiště. Při představě neustálého dotazování bychom ztratili přehled o uživatelích, kteří je provádějí.

4.3.4 User Statistics feature by Percona

Percona je drop-in náhrada MySQL projektu a zaměřuje se na poskytnutí lepšího výkonu, konzistence a škálování na hardware. Díky zavedení proměnné userstat u MySQL verze 5.1 a MariaDB verze 5.2.0 nám umožňuje ukládat uživatelské statistiky přímo do tabulky databáze.

Percona Server for MySQL	MySQL	MariaDB
1. verze 5.1 – zavedení userstat_running variable	verze 5.1	5.2.0 – zavedení User statistics, userstat variable
2. verze 5.5 – přejmenování userstat_running na userstat	verze 5.5	
3. verze 5.6 – přidání INFORMATION_SCHEMA tables a userstat variable z verze 5.5	verze 5.6	
4. verze 5.7 – feature User statistics beze změn	verze 5.7	
5. verze 8.0 – feature User statistics beze změn	verze 8.0	

4.3.5 Prometheus a Grafana Monitoring

Prometheus je time series databáze a využívá se zejména pro sběr a ukládání metrických dat. Princip jejich fungování je detailněji popsán na straně 14.

Prometheus však poskytuje spoustu dalších funkcionalit a je navržen tak, aby byl schopen monitorovat servery, databáze, virtuální stroje apod. V našem případě je Prometheus schopen prostřednictvím tzv. mysql_exporteru získávat informace z databázové tabulky, která obsahuje údaje o uživatelské aktivitě. Tyto údaje budou shromažďovány a vizualizovány.

Konfigurační soubor Prometheus:

```
[Unit]
Description=MySQL Exporter
User=prometheus

[Service]
Type=simple
Restart=always
ExecStart=/usr/local/bin/mysqld_exporter \
--config.my-cnf /etc/.exporter.cnf \
--collect.auto_increment.columns \
--collect.binlog_size \
--collect.engine_innodb_status \
--collect.engine_tokudb_status \
--collect.global_status \
--collect.info_schema.userstats \
--web.listen-address=0.0.0.0:9104

[Install]
WantedBy=multi-user.target
```

Při konfiguraci je důležité přidání collectoru pro sběr uživatelských statistik s názvem collect.info_schema.userstats. Právě ten zajišťuje dolování uživatelských statistik z tabulky INFORMATION_SCHEMA.USER_STATISTICS. Tato tabulka obsahuje metrické údaje o počtu připojení, počtu odeslaných a přijatých řádků, dobu, po kterou byl uživatel připojen apod. Veškeré tyto údaje jsou užitečné, ale neposkytují informaci o tom, kterou databázi na daný moment uživatel využívá.

V relačních databázových systémech jako jsou MySQL a MariaDB, se informace o spuštěných instancích databáze ukládají do tabulky INFORMATION_SCHEMA.PROCESSLIST. Tato tabulka vypadá následovně:

```
SHOW FULL PROCESSLIST;

+-----+------+-----------+------+---------+------+-------+----------------------+---------
| Id  | User | Host      | db   | Command | Time | State | Info                 | Progress
|+-----+------+-----------+------+---------+------+-------+----------------------+--------

| 126 | root | localhost | DEV | Query   |    0 | NULL  | SHOW FULL PROCESSLIST |    0.000
|+-----+------+-----------+------+---------+------+-------+----------------------+--------
```

Tabulka ukazuje ID, jméno uživatele a databázi, ke které je v danou chvíli připojen. Abychom těchto informací mohli využít a sledovat aktivitu uživatele v konkrétní databázi, bylo rozhodnuto přenést data z této tabulky do Elasticsearch a následně vizualizovat prostřednictvím Grafany.

K transportu záznamů z tabulky processlist bylo použito softwaru Logstash. Ten disponuje JDBC input pluginem, který umožňuje přenos dat z kterékoli databáze pomocí JDBC (Java Database Connectivity). JDBC je množina tříd, které v sobě implementují rozhraní definované v JDBC API. To umožňuje programům napsaným v Java komunikovat s databázovými servery. Přístup k databázovým serverům je provádět prostřednictvím JDBC ovladačů, které jsou vydávány jednotlivými databázovými servery. [30]

JDBC input plugin se instaluje příkazem `bin/logstash-plugin install logstash-input-jdbc` jehož defaultní umístění je v adresáři `/usr/share/logstash.` Tento plugin nezahrnuje knihovny s JDBC ovladači. Pro získání JDBC ovladačů na Linux operačním systému Ubuntu bylo využito příkazu `libmariadb-java - Java database driver for MariaDB and MySQL`. Ovladače je dále nutné umístit do adresáře

`/usr/share/logstash/logstash-core/lib/jars/.` Tímto bylo dosaženo příprav pro testování přenosu dat z relační databáze MariaDB do Elasticsearch pomocí Logstash. Konfigurační soubor Logstash pro spuštění transportu dat by mohl vypadat následovně:

```
input {
  jdbc {
     jdbc_validate_connection => true
     jdbc_driver_library=>"/usr/share/logstash/logstash
core/lib/jars/mariadb-java-client-2.4.2.jar"
     jdbc_driver_class => "Java::org.mariadb.jdbc.Driver"
     jdbc_connection_string                                    =>
"jdbc:mariadb://localhost:3306/test_database"
     jdbc_validate_connection => true
     connection_retry_attempts => 3
     jdbc_user => "root"
     jdbc_password => "root"
     schedule => "* * * * *"
   statement => "SELECT * FROM INFORMATION_SCHEMA.PROCESSLIST;"
   }
}
```

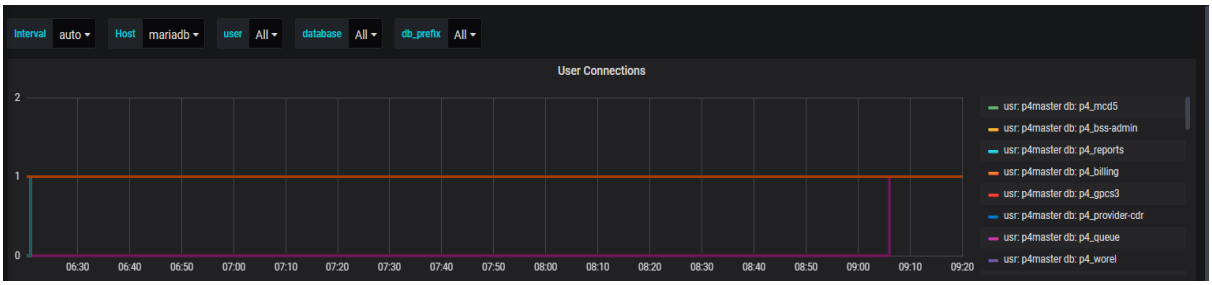
Tato jednoduchá konfigurace poskytne informace o aktivitě uživatele v konkrétní databázi v reálném čase. Parametry jdbc_user a jdbc_password definují údaje uživatele, přes kterého se Logstash přihlašuje k databázi. Schedule říká, jak často má Logstash pokládat dotaz neboli statement. Schedule funguje na základě stejné syntaxe jako cron (5 hvězdiček položí dotaz každou minutu). Procceslist však přiřazuje každému uživateli jedno identifikační číslo. To znamená, že pokud se například uživatel root připojí k databázi database1, dostane identifikační číslo 1. Poté se odpojí a připojí se k databázi database2 se stejným identifikačním číslem. Tím pádem je ztracen záznam o prvním vstupu do databáze database1 a je nahrazen vstupem do databáze database2.

Z výše uvedeného důvodu je nutné vytvoření filtru v konfiguračním souboru. Tento filtr bude přiřazovat každému záznamu specifické identifikační číslo v závislosti na čase v poli timestam a bude ho ukládat do námi definovaného nového pole, například doc_id. Pro změny nad jednotlivými poli Logstash využívá tzv. Mutate filter plugin. Tento plugin umožňuje provádět změny jako například přejmenování pole, odstranění, přemístění apod. viz. [31]

```
filter{
 mutate {add_field => {"doc_id" => "%{id}-%{user} %{@timestamp}"}
 }
 grok {
   match => {
     "db" => ["%{DATA:db_prefix}_%{GREEDYDATA:db_name}"]
   }
 }
 grok {
   match => {
      "db" => ["%{DATA:db_prefix}-%{GREEDYDATA:db_name}"]
   }
 }
}
```

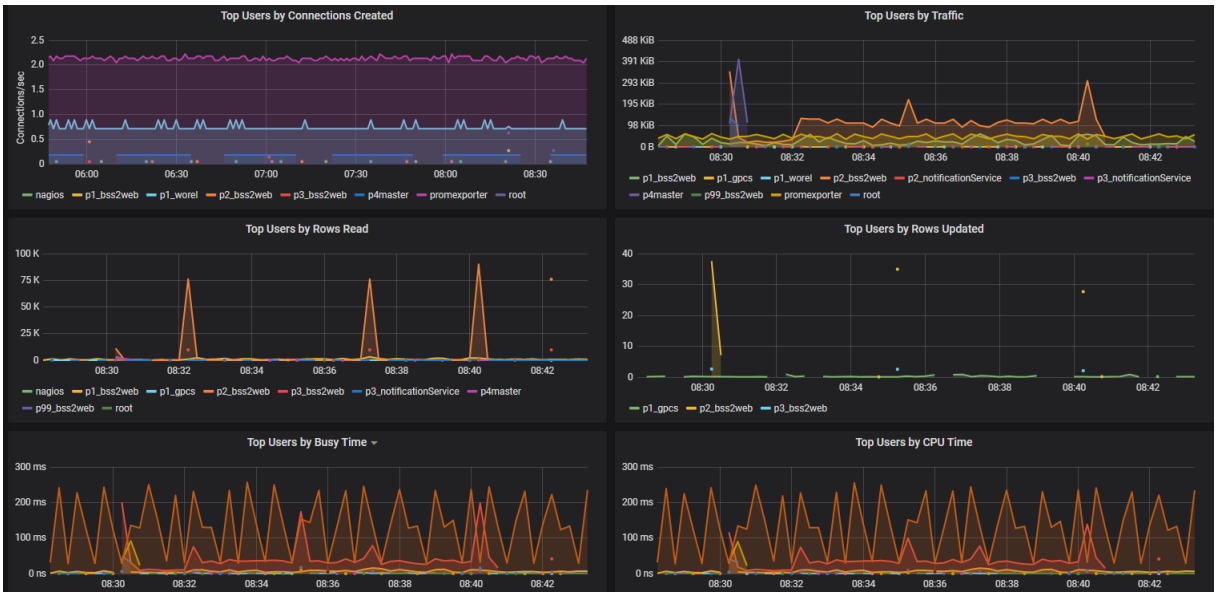
Jelikož jsou názvy databází, se kterými pracujeme, ve tvaru prefix_jméno databáze či prefix-jméno databáze a je potřeba vyhledávat i podle jména databáze i prefixu, byli přidány dva grok filtry, které nám rozdělí název databáze na prefix a jméno databáze. Tyto údaje pak zvlášť uloží do jednotlivých polí.

K vizualizaci statistik bylo využito Grafany, která disponuje Percona pluginem s dashboardy. Tyto dashboardy nabízí velmi přehlednou vizualizaci dat, dolovaných prostřednictvím Prometheus. V Grafaně je možné vidět statistiky top uživatelů seřazených podle počtu připojení do databáze, podle aktivity v databázi, nebo například top uživatele podle CPU Time.



Obrázek 19: Graf aktivních uživatelů databází v Grafana

Na ose Y jsou hodnoty 0 a 1, kdy hodnota 1 znamená, že uživatel je připojen a 0 opak. V pravé části grafu můžeme vidět jméno uživatele (usr) a databázi, ke které je momentálně připojen (db). V horní části Je možné filtrovat zobrazení grafu podle uživatele, databáze a prefixu.



Obrázek 20: Grafy zobrazující metrické údaje získané z time series databáze Prometheus

4.3.6 Shrnutí

Tento projekt ukazuje postup k nasazení konkrétního řešení pro sledování uživatelské aktivity v relačních databázích a získávání tak cenných informací nejenom o tom, který uživatel je v danou dobu aktivní, ale i to, jak vytěžuje databázi či jak často ji využívá. Jak je vidět, první myšlenkou bylo získávání dat ze slow_query_logů, které se při testování prokázalo jako nevhodné. Další možností byla time series databáze Prometheus, která splňovala většinu požadavků. Optimálním řešením nakonec byla kombinace time series databáze a Logstash input jdbc pluginu.

5 Závěr

Cílem této práce bylo navrhnout komplexní řešení pro analýzu dat. K tomu bylo třeba prostudovat dostupné, zejména open source nástroje pro zpracování, indexaci, analýzu a vizualizaci dat s důrazem na logové zprávy. V práci bylo představeno několik, na daný moment aktuálně nejpoužívanějších přístupů k analýze nestrukturovaných dat a s nimi svázanými softwarové produkty. Porovnáním nástrojů pro práci s daty byl vybrán konkrétní systém, který byl následně implementován a testován v praxi.

V teoretické části bylo nutné nastudovat principy různých technologií, jako jsou například protokoly transportní vrstvy, jenž se využívají pro přenos dat či nové možnosti v ukládání velkého objemu nestrukturovaných dat.

Na základě praktické části byl učiněn závěr, že i při současném trendu využívání NoSQL přístupů při zpracovávání velkého objemu nestrukturovaných dat, jsou relační databáze mnohdy nenahraditelnou součástí. Často se tedy setkáváme s tím, že NoSQL databáze rozšiřují své funkce o možnost dotazování se v SQL jazyce. Jedním z poznatků je také ten, že v současné době nejsou dostupné žádné open source projekty, které by zajišťovaly vytváření složitějších alarmů.

Navržené systémy a postupy představené v projektech se opírají o technologie, jež jsou momentálně dostupné na open-source licencích a řeší monitoring velkého objemu dat. Díky tomu je možné mít přehled o dění na jednotlivých serverech a v případě nežádoucích okolností pohotově zasáhnout, což ve finále vedlo k jejich ostrému nasazení.

Realizace této bakalářské práce, zejména praktické části, kterou jsem zpracovával ve společnosti MATERNA, a. s. pro mě byla velkým přínosem. Dva ze tří projektů uvedených v praktické části jsou nyní implementovány a využívány v provozu, což osobně považuji za úspěch. Tvorba této bakalářské práce mi tak dala nejenom nespočet nových znalostí a zkušeností v oblasti informatiky, ale i v oboru řízení podniku.

6 Seznam použité literatury

[1] Anton Chuvakin, Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management, Syngress Media, 2013, ISBN 9781597496353

[2] Evi Nemeth, Garth Snyder, Hein Trent, UNIX® and LINUX® system administration handbook 5th edition, Addison-Wesley Educational Publishers, 2017, ISBN-13: 978-0-13-427755-4

[3] Kabelová, Alena a Libor Dostálek. Velký průvodce protokoly TCP/IP a systémem DNS. 5., aktualiz. vyd. Brno: Computer Press, 2008. ISBN 978-80-251-2236-5

[4] Dostálek, Libor a Alena Kabelová. Understanding TCP/IP A clear and comprehensive guide to TCP/IP protocols. 1. 32 Lincoln Road Olton Birmingham, B27 6PA, UK.: Packt Publishing, 2006. ISBN 1-904811-71-X

[5] RFC 3828 - The Lightweight User Datagram Protocol. IETF Tools [online]. 2004 [cit. 2020-02-22]. Dostupné z: https://tools.ietf.org/html/rfc3828/

[6] RFC 4960 - Stream Control Transmission Protocol. IETF Tools [online]. 2004 [cit. 2020-02-22]. Dostupné z: https://tools.ietf.org/html/rfc4960

[7] Filebeat overview. Elastic [online]. 2019 [cit. 2020-02-22]. Dostupné z: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html

[8] Logstash. Elastic [online]. 2019 [cit. 2020-02-22]. Dostupné z: https://www.elastic.co/logstash

[9] Logstash. Elastic [online]. 2019 [cit. 2020-02-22]. Dostupné z: https://www.elastic.co/guide/en/logstash/current/introduction.html

[10] Mullins, Craig. Database administration: the complete guide to practices and procedures. Boston: Addison-Wesley, c2002. ISBN 0-201-74129-6

[11] Big Data. IBM [online]. 2015 [cit. 2020-02-22]. Dostupné z: http://www.ibm.com

[12] Holubová, Irena, Jiří KOSEK, Karel Minařík a David Novák. Big Data a NoSQL databáze. Praha: Grada, 2015. Profesionál. ISBN 9788024754666.

[14] Hills, Ted. NoSQL and SQL data modeling: bringing together data, semantics, and software. Basking Ridge, NJ: Technics Publications, [2016]. ISBN 9781634621090.

[15] InfluxDatabase. Influxdata [online]. 2020 [cit. 2020-02-22]. Dostupné z: https://www.influxdata.com/time-series-database/

[16] Prometheus. Prometheus [online]. 2020 [cit. 2020-02-22]. Dostupné z: https://prometheus.io/docs/introduction/overview/

[17] ELK Stack. Elastic [online]. 2020 [cit. 2020-02-22]. Dostupné z: https://www.elastic.co/what-is/elk-stack/

[18] Elasticsearch. Elastic [online]. 2020 [cit. 2020-02-22]. Dostupné z: https://www.elastic.co/what-is/elasticsearch/

[19] Kibana. Elastic [online]. 2020 [cit. 2020-02-22]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/introduction.html/

[20] Graylog. Graylog [online]. 2020 [cit. 2020-02-22]. Dostupné z: https://docs.graylog.org/en/3.2/pages/getting_started/planning.html

[21] Sidecar. Graylog [online]. 2020 [cit. 2020-02-22]. Dostupné z: https://docs.graylog.org/en/3.2/pages/sidecar.html

[22] LOGalyze. Logalyze [online]. 2020 [cit. 2020-02-22]. Dostupné z: http://www.logalyze.com/

[23] ElastAlert. ElastAlert [online]. 2014 [cit. 2020-03-14]. Dostupné z: https://elastalert.readthedocs.io/en/latest/elastalert.html#overview

[24] Grok filter plugin. Elastic [online]. 2019 [cit. 2020-03-20]. Dostupné z: https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html

[25] Watcher. Elastic [online]. 2020 [cit. 2020-03-20]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/watcher-ui.html#watcher-create-advanced-watch

[26] Bucket Aggregations. Elastic [online]. 2020 [cit. 2020-03-23]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket.html

[27] Pipeline Aggregations. Elastic [online]. 2020 [cit. 2020-03-23]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-pipeline.html

[28] Entity-centric indexing. Elastic [online]. 2015 [cit. 2020-03-25]. Dostupné z: https://www.elastic.co/elasticon/2015/sf/building-entity-centric-indexes

[29] Geo-point datatype. Elastic [online]. 2020 [cit. 2020-03-28]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/geo-point.html

[30] Java 6: JDBC and database applications: Software Development [online]. London: bookboon.com, 2017 [cit. 2020-04-02]. ISBN 978-87-403-1736-7. Dostupné z: https://bookboon.com/cs/java-6-jdbc-and-database-applications-ebook

7 Seznam obrázků