Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Logování, Monitoring a Alerting za použití ELK Stack

Diplomová práce

Autor: Bc. Tomáš Novák

Studijní obor: (N1802) Aplikovaná informatika

Vedoucí práce: doc. RNDr. Petra Poulová, Ph.D.

Odborný konzultant: Ing. Tomáš Burda, Systemart s. r. o.

Hradec Králové duben 2021

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně s použitím uvedené literatury a zdrojů.

V Hradci Králové dne 27. 4. 2021

Bc. Tomáš Novák

Poděkování:

Děkuji vedoucímu diplomové práce doc. RNDr. Petře Poulové, Ph.D. za metodické vedení práce.

Dále bych rád poděkoval Ing. Davidu Petríkovi, Ing. Tomáši Burdovi a celé společnosti Systemart s. r. o. za předané zkušenosti a možnost se na tomto projektu podílet.

Poslední poděkování patří mým rodičům a přítelkyni, za motivaci a podporu při psaní této diplomové práce i během celého magisterského studia.

Anotace:

Práce pojednává o využití nástrojů z rodiny ELK Stack a dalších doprovodných prostředcích, zajišťujících logování v rámci aplikace Logeto. Hlavním cílem je vytvoření logovací infrastruktury v kontejnerizačním prostředí Kubernetes obstarávajícím sběr logů z aplikace Logeto, zasílání varovných zpráv do komunikačního nástroje MS Teams a monitoring interních nástrojů logovací infrastruktury. Řešení fronty zpráv zajišťuje platforma Azure Event Hub. Ke sběru, ukládání a analýze logů je zde využito nástroje Elastic Cloud on Kubernetes a nástroj ElastAlert, ve své kontejnerizované variantě, řeší zasílání varovných zpráv. Kromě popisu jednotlivých komponent a jejich fungování práce obsahuje také konfigurační soubory umožňující replikovat představené řešení. Výsledkem této práce je zavedení funkční logovací infrastruktury nad aplikací Logeto a vyvinutí vlastního ElastAlert modulu pro zasílání personalizovaných varovných zpráv do komunikačního nástroje MS Teams.

Annotation:

Title: Logging, Monitoring and Alerting with ELK Stack

This thesis deals with the use of tools from the ELK Stack family and other accompanying tools, providing logging the application called „Logeto“. The main goal is to create a logging infrastructure in the Kubernetes container environment, providing log collection from the Logeto application, sending warning messages to the MS Teams communication tool and monitoring the internal tools of the logging infrastructure. The message queue solution is provided by the Azure Event Hub. The Elastic Cloud on Kubernetes tool is used to collect, store and analyze logs, and the ElastAlert tool, solves the sending of warning messages, both in its containerized variant. In addition to the description of individual components and their operation, there are also configuration files that allow you to replicate a presented solution. The result of this work is the introduction of a functional logging infrastructure over the Logeto application and the development of its own ElastAlert module for sending personalized warning messages to the MS Teams communication tool.

Obsah

Seznam obrázků

Seznam zdrojových kódů

1 Úvod

Udržovat si přehled nad vlastním produktem, softwarem či infrastrukturou je důležité nejen pro vývojáře, ale i zadavatele projektu. Tento přehled může podat cenné informace ohledně výše nákladů, uživatelském chování nebo o kvalitě samotného vývoje. K zisku těchto cenných informací, je využíváno služeb odborníků z různých odvětví a široké škály analytických nástrojů. V kontextu vývoje softwaru se nejčastěji jedná o specialisty na UI/UX a systémových analytiků, kteří zkoumají kvalitu interakce mezi softwarem a uživatelem a snaží se dosáhnout co nejlepšího uživatelského zážitku. Jejich poznatky jsou dále validovány za použití trasování v rámci hotového softwaru a dalších podpůrných nástrojů.

Nedílnou součástí vývoje softwaru, ať už mobilních, webových nebo desktopových aplikací je proto sledování a analyzování celého procesu. V jeho průběhu je možné sbírat data o chování vývojářů, kvalitě kódu, který vytvářejí i rychlosti vývoje. Cílem je přitom dosáhnout co možná největší efektivity procesu. Kromě zmíněné analytiky probíhá již v rámci vývoje testování doposud vytvořených částí softwaru. Tyto části jsou opět podrobovány důkladné analýze, jejímž cílem je dosažení validity softwaru a jeho infrastruktury.

Poslední částí životního cyklu vývoje softwaru je finální testování, re-validace všech předchozích částí vývoje a zapracování získaných informací, případně opravy objevených chyb do finální verze produktu. Provádí se tak například zátěžové testy, jež simulují nestandardní podmínky při používání aplikace (například pomalejší připojení k internetu) či atypická interakce uživatele s nástrojem.

Všechny výše popsané části vývoje mají jednu společnou část, a tou je sběr informací, mezi které spadá logování událostí a sběr metrik. Existuje řada způsobů, jak k této problematice přistoupit. Tato diplomová práce se zaměřuje na charakteristiku několika vybraných nástrojů spolu s popisem jejich implementace v reálném prostředí nad již existující aplikací Logeto. Popisovanými nástroji je ElastAlert zajišťující notifikace událostí a Beats, Logstash, ElasticSearch a Kibana, které dohromady vytváří ELK Stack.

1.1 Vymezení práce

Cílem této práce je představit doménu logování, monitoringu a alertingu za použití nástrojů ELK Stack a ElastAlert. Dále pak popsat nástroje, které mohou sloužit k tomuto účelu a představit jejich konfiguraci s konkrétní implementací nad produktem Logeto od společnosti Systemart s.r.o.

Výstupem práce je zavedení centralizovaného logování a monitoringu nad aplikací Logeto a vytvoření modulu pro napojení nástroje ElastAlert na webhook komunikátoru MS Teams. To vše konfigurované a nasazené v rámci kontejnerizačním prostředí Kubernetes jako on-premise řešení.

1.1.1 Důvod výběru tématu

Autorem této práce je student oboru Aplikovaná informatika na Univerzitě Hradec Králové. Díky zkušenostem nabytým během studia získal pozici v lokální softwarové firmě Systemart s.r.o., která se zabývá vývojem vlastního docházkového systému Logeto. Autor se zde zabývá manuálním testováním aplikace, vývojem automatického testování aplikací pro operační systém Android, zastává zde také pozici UI/UX designéra a podílí se i na migraci aplikace do kontejnerizačního prostředí nasazeného jako on-premise řešení.

V rámci posledního zmíněného zaměření byla firmou Systemart s.r.o. vyžadována také implementace centralizovaného logování a monitoringu za použití technologií ElasticSearch a Prometheus. Téma diplomová práce tak reflektuje autorovy vlastní zkušenosti nabyté v praxi a popisuje způsob, jakým bylo implementováno zadání zaměstnavatele a zadavatele tématu. Sám autor si pak za cíl stanovuje uvést text práce zajímavou formou, tak aby text práce mohl posloužit jako zdroj pro ty, kteří se chtějí dozvědět víc o tematice logování a monitoringu, případně ho využít na reálných projektech podobného rozsahu.

1.1.2 Struktura práce

První část práce uvádí čtenáře do řešeného problému a nastiňuje, jak může být jeho vyřešení prospěšné pro vývoj aplikací a jejich uživatele. V další části jsou teoreticky

popsány nástroje, které je možné využít k vyřešení stanoveného problému. Jelikož jsou tyto nástroje v části implementace popisovány ve své kontejnerizované variantě, jsou v druhé části popsány také kontejnerizační nástroje a nástroje umožňující kontejnerizované nasazení a škálování.

Třetí část představuje čtenáři záměr zadavatele řešeného problému, popis vlastností a architektury aplikace. To vše s důrazem na pochopení přínosu implementace centralizovaného logování nad danou aplikaci.

Poslední část práce je zaměřena na implementaci konkrétního řešení a zavedení centralizovaného logování nad aplikaci zadavatele. V rámci implementace je řešeno také zasílání varovných zpráv při nevalidním, nebo neočekávaném chování aplikace. Veškerá implementace je zde stavěna nad již zavedeném kontejnerizačním prostředí Kubernetes. V závěru práce je zhodnocení implementovaného přístupu, spolupráce se zadavatelem a představení potenciálních vylepšení.

2 Literární rešerše

Tematikou ELK Stacku se zabývá řada zahraničních knih a příruček. Mezi nejčastěji citované patří *Learning ELK Stack* [1], *Beginning Elastic Stack* [2] a *Mastering Elastic Stack* [3]. První zmíněnou publikaci *Learning ELK Stack* vydal Saurabh Chhajed pod vydavatelstvím Packt Publishing. Publikace se v úvodu zabývá významem různých druhů logování a monitoringu, dále se soustředí na obecné koncepty a architekturu ELK Stacku. Implementační detaily jsou zde zmíněny pouze velmi povrchově a naznačují jen nutné minimum znalostí, potřebné k nasazení ELK Stacku v základním nastavení.

Druhou jmenovanou publikaci – *Beginning Elastic Stack,* napsal Vishal Sharma a vydalo ji nakladatelství Apress. Tato publikace se již více věnuje technickým specifikům a specifikacím ELK Stacku, požadovaným nárokům na hardware a také konkrétním implementacím některých ze základních konceptů. Autor se zde zabývá také konfigurací nástroje Foreman Puppet (nyní pouze The Foreman), který ve starších verzích ElasticSearch obstarával životní cyklus indexů. Správa životního cyklu indexů je však v ElasticSearch od verze 6.6 podporována nativně. Proto tato část publikace již není aktuální.

Poslední publikaci *Mastering Elastic Stack* vydal Yuvraj Gupta pod vydavatelstvím Packt Publishing. Autor se zde velmi podrobně věnuje celé koncepci ELK Stacku. V úvodu zmiňuje informace o jeho historii a základních konceptech. Poté čtenáře provází skrze všechny nástroje obsažené v ELK Stacku. Zabývá se jejich instalací, detailní konfigurací, a dokonce popisem existujících licencí a edicí, které společnost Elastic N.V. nabízí. Poslední kapitolou této publikace je případová studie, jež přináší komplexní ukázku implementace ELK Stacku v praxi.

První dvě jmenované publikace jsou z pohledu komplexnosti ELK Stacku příliš krátké a zabývají se problematikou pouze povrchově. Publikace „*Mastering Elastic Stack*“ byla vydána již v roce 2017, některé její koncepty tedy již nejsou aktuální.

V důsledku těchto skutečností nebylo z prvních dvou jmenovaných publikací čerpáno více než jen základní teoretické koncepty. Z poslední jmenované publikace jsou

pak čerpány jen ty části ELK Stacku, které od roku 2017 neprošli výraznou koncepční změnou.

Přínosným a současně důvěryhodným zdrojem využitým k tvorbě této diplomové práce, je oficiální produktová dokumentace [4], vydávaná společností Elastic N. V. Tato oficiální produktová dokumentace obsahuje aktuální informace a v mnoha případech hovoří také o chystaných změnách, díky čemuž se mohou vývojáři připravit na příchod nové verze a plynule na ni přejít. Tato dokumentace je nejspíš tím nejpodrobnějším a nejspolehlivějším zdrojem, který lze k problematice ELK Stacku nalézt.

Problematikou ELK Stacku, nebo alespoň některou jeho částí, se zabývá i několik akademických prací. V rámci této práce se v úvahu berou pouze akademické práce českých autorů. Přínosným zdrojem informací je diplomová práce z roku 2018 s názvem *Logování pomocí ELK Stacku* [6], v níž se autor věnuje řešení případové studie projektu centralizovaného logování pro společnost Fortuna. První část se zabývá popisem fungování ELK Stacku a využitých doprovodných nástrojů. Hlavním rozdílem oproti této práci je absence zasílání varovných zpráv a také způsob, jakým jsou logované zprávy zasílány z klientských aplikací do interní sítě.

Nejpodobnější této práci je diplomová práce *Application Log Analysis* [7] z roku 2015. Tato práce pochází od české autorky, avšak byla napsána v anglickém jazyce. Autorka zde podrobně popisuje význam logování, monitoringu a analýzy logů. Dále vysvětluje princip fungování ELK Stacku, jeho konfiguraci a použití společně se zasíláním varovných zpráv. Jde o velice přínosnou práci, která se zdá být v kontextu dané problematiky nejrozsáhlejší a nejpodrobnější. Jediným nedostatkem může být rok obhájení, kdy některé implementační postupy již nejsou aktuální, avšak rozsáhlá teoretická část tuto nevýhodu zdatně kompenzuje.

Ostatní akademické práce se problematikou ELK Stacku zabývají již velice povrchově. V některých případech se práce věnují jen teoretickým popisům fungování ELK Stacku, a jemu přidružených nástrojů. Příkladem takovéto práce je *Využití technologie Elastic Stack pro sběr a analýzu logů* [8], jejímž jediným implementačním přínosem je tvorba vizualizací v prostředí Kibana. Jiné práce se naopak zabývají

především analýzou logů, jejich významem a vizualizací. ELK Stack je v tomto případě popisován jako hotové řešení sloužící jako nástroj pro pohodlnou analýzu logů a tvorbu vizualizací v prostředí Kibana. Příkladem takové práce je *Analýza sieťovej prevádzky pomocou nástrojov rodiny Elastic.* [9]

V případě doprovodného nástroje ElastAlert byl taktéž zvolen přístup čerpání informací z produktové dokumentace [5]. U dalších doprovodných nástrojů – Docker a Kubernetes, jsou informace čerpány z odborných článků a akademických prací. Prací, které ke svým výsledkům nějakým způsobem využívají ElasticSearch, je celá řada, ale podobnost tématu nebo cílů s touto prací je velmi malá. Jako zdroj pro pochopení i popis doprovodných nástrojů Docker a Kubernetes, tak sloužila především oficiální dokumentace, odborné články a diplomová práce *Využití Dockeru pro správu aplikací v kontejnerech* [10] od Ing. Tomáše Burdy. Práce se zabývá implementací kontejnerizačního prostředí Kubernetes ve společnosti Systemart s.r.o. a jsou v ní vysvětleny teoretické koncepty kontejnerizace i konkrétní implementační detaily. Jelikož se jedná o práci, která vznikla ve spolupráci se společností, která je zadavatelem i této práce, na ni bude tato diplomová práce částečně navazovat a některé její koncepty dále rozvíjet detailněji.

3 ELK Stack

Open-source nástroj ELK Stack od společnosti Elastic N. V., patří mezi špičku nástrojů určených k logování, monitoringu dat a jejich analytice. ELK je akronym tří hlavních částí architektury této platformy – ElasticSearch, Logstash a grafické rozhraní Kibana. Zřídka se lze setkat i s variacemi tohoto názvu, zohledňujícími další nástroje, které je možné použít. Těmito variacemi jsou EBK a ELBK. V kontextu této práce bude užíváno pouze původního označení ELK, přestože zohledňuje pouze tři hlavní nástroje. [11]

Obrázek 1 *Vizualizace ELK Stacku (převzato z [11])*

3.1 ElasticSearch

Hlavním a nenahraditelným nástrojem ELK Stacku je ElasticSearch. Jedná se o velmi výkonnou dokumentově orientovanou databázi, určenou především pro real-time full-text vyhledávání. Její rychlost vyhledávání je dána tím, že je ElasticSearch postaven nad enginem Apache Lucene. Celou ElasticSearch databázi je možné škálovat vertikálně i horizontálně, přičemž se vyznačuje spolehlivostí, dostupností a možností multitenance. Veškerá komunikace a ovládání ElasticSearch probíhá pomocí JSON skrze RESTful API.

Díky tomuto API je ElasticSearch také schopen přijímat a indexovat data. O indexaci se zde stará engine Apache Lucene, a to ve svém schema-less módu, který

umožňuje indexování dat bez předem známé struktury. V analogii s relační databází je možné do ElasticSearch uložit řádek tabulky, bez toho aniž by bylo předtím nutné tabulku vytvářet a specifikovat datové typy sloupců. Tohoto chování je možné využít v mnoha situacích. Pro přehlednější správu persistovaných dat se však doporučuje použití funkce ILM (index lifecycle management). [1][3]

Jako příklad velikosti či užitečnosti tohoto nástroje je možné uvést několik předních českých i zahraničních společností, jež ElasticSearch k perzistenci dat a full-textovému vyhledávání využívají. Z českých zástupců jsou to například Alza [12] a Zonky. Z těch zahraničních lze jmenovat například Wikipedia, GitHub nebo SoundCloud. [1]

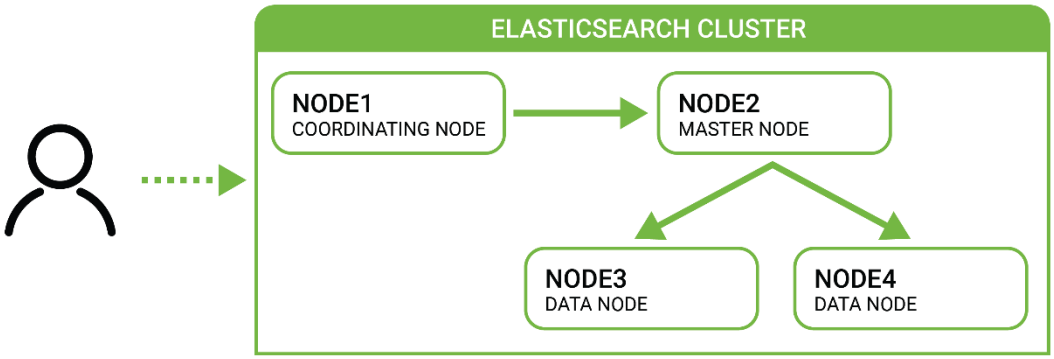
3.1.1 ElasticSearch cluster

Srdcem celého ElasticSearch je node (uzel). Jedná se o jednu běžící instanci, která má za úkol ukládat a indexovat data. Tato instance může zastávat několik rolí, dle své požadované funkcionality. Avšak takováto jedna instance poté pozbývá vysoké dostupnosti a replikace dat, je tedy nutné a vysoce doporučené spustit několik nodů a zavést tak horizontální škálování. Všechny nody se po propojení automaticky postarají o replikaci dat a zaručí tak jejich dostupnost i v případně výpadku jednoho z nodů. Tímto vzniká takzvaný ElasticSearch cluster. [13]

Nody v tomto clusteru však neslouží pouze ke škálování dat. Lze díky nim dosáhnout několika vrstvé architektury clusteru za pomocí již výše zmíněných rolí.

- **Master node** – zodpovědnost za stav, zdraví, správu a konfiguraci celého clusteru. Řídí interní komunikaci a přidávání/odebírání nodů. [14]
- **Data node** – zodpovědný pouze za ukládání dat a datově-relační operace jako je vyhledávání a agregace. [14]
- **Coordinating node** – někdy nazývaný také client node, fungující jako chytrý loadbalancer. Užitečný je především u velmi rozsáhlých clusterů. Tímto typem nodu se uzel stane, pokud má v konfiguraci zakázáno být masterem i datovým nodem. [14]

- **Machine Learn node** – zodpovědný jen a pouze za výpočty strojového učení, dostupný pouze pod placenou licencí. [14]
- **Teransform node** – nejnovější role zavedená ve verzi 7.9, obstarávající výpočty transformací dat před indexací nebo z jiných, již indexovaných dat. [14]



Obrázek 2 *Příklad vícevrstvého ElasticSearch Clusteru (vlastní)*

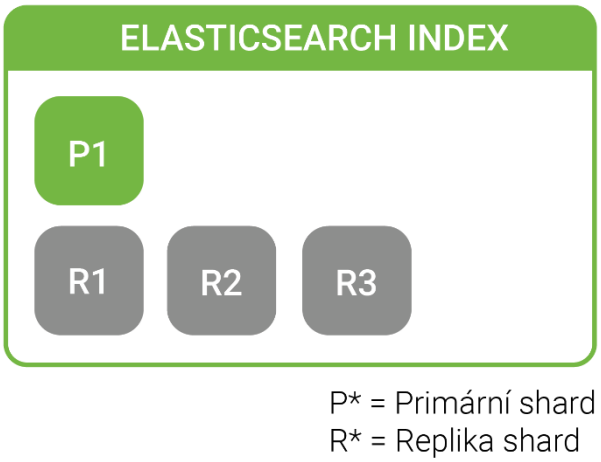
Do verze ElasticSearch 7.8 zde existovaly ještě role Client a Tribe node, ty ale byly odstraněny a nahrazeny funkcí pro mezi clusterovou komunikaci. [14]

V základním nastavení je všem nodům přiřazena role master, data a ingest, s tím, že aktivním masterem, spravujícím cluster, může být jen jeden uzel. Ostatní nody s touto rolí pouze vyčkávají jako náhrada hlavního aktivního mastera v případě výpadku. Obecným doporučením dle dokumentace ElasticSearch je nody co nejvíce specializovat tak, aby každý node, zastával pouze jednu roli. Docílí se tak mnohem stabilnějšího a výkonnějšího clusteru. Umožní to také specializovat nody na hardwarové úrovni, jelikož nody určené pro výpočetní výkon nepotřebují tak velké uložiště a obráceně. [14]

3.1.2 Základní struktura dat

Pro lepší pochopení toho, proč a jak dokáže ElasticSearch nabízet tak rychlé prohledávání logů bez velkých nároků na výkon, je nutné poznat vnitřní strukturu celého nástroje, a to jak jsou data v ElasticSearch uložena.

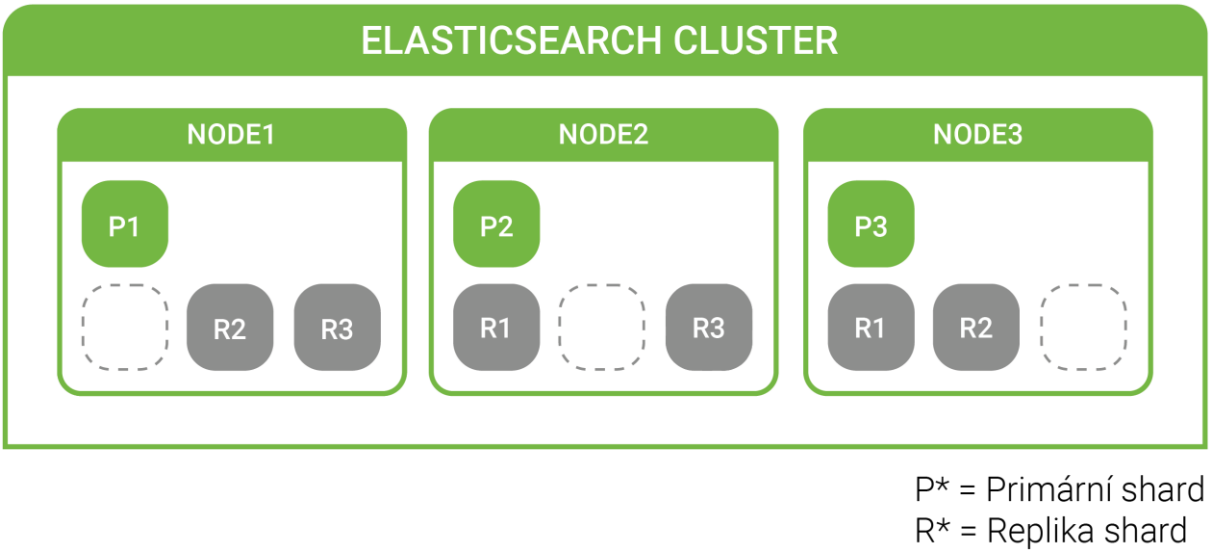
Hlavní datovou jednotkou uvnitř ElasticSearch je index, jedná se o logickou jednotku organizující data do kontextově stejných celků (v analogii s relačními databázemi se jedná o databázi). [15]



Obrázek 3 *ElasticSearch index se svými shardy (převzato z [15])*

Index je složen z shardů (úlomků), kterých je při základním nastavení pět. Shardem je myšlena jedna instance Lucene indexu, jež ukládá samotná data a provádí nad nimi vyhledávání. ElasticSearch si pro tento typ indexu zvolil terminologii shard, aby nedošlo k záměně ElasticSearch indexů s Lucene indexy.

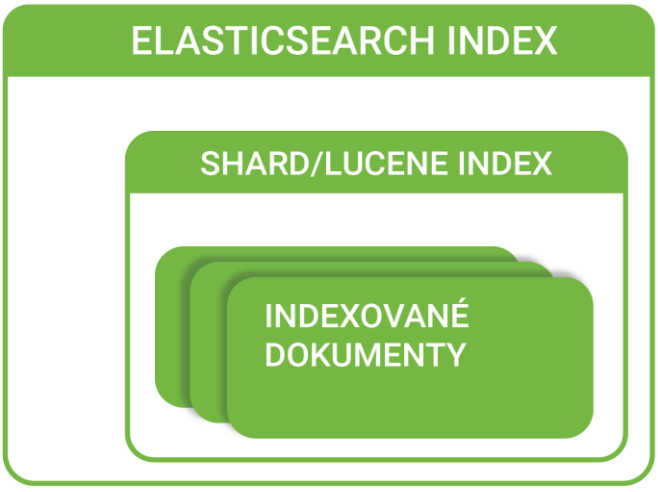
Existují dva typy ElasticSearch shardů, takzvané primární a repliky. Pro každý ElasticSearch index musí existovat alespoň jeden primární shard, přičemž jejich počet již nelze v průběhu existence indexu měnit. Dále může index obsahovat libovolný počet replik shardů, jejichž počty lze během života indexu upravovat a je doporučené aby existovalo alespoň tolik replik shardů, kolik obsahuje ElasticSearch cluster datových nodů. Repliky jsou pak rozmístěny na všechny datové nody, aby bylo možné automaticky obnovit data při výpadku nodu s primárním shardem. [15][16]



Obrázek 4 *Příklad alokace shardů napříč clusterem (převzato z [15])*

Shardy pak obsahují samotné dokumenty, které lze přirovnat k jednotlivým řádkům v tabulce relačních databází. Struktura uložení je ale samozřejmě rozdílná. [17]

ElasticSearch index lze dělit ještě na typy. Typ indexu je v analogii s relačními databázemi něco jako tabulka a umožňuje do jednoho indexu uložit data rozdílných struktur a významů, tento přístup ale není příliš běžný. Nejčastěji mívá index jen jeden typ (tedy v analogii jednu tabulku), k tomuto typu se pak vždy váže konkrétní mapování (v analogii definice sloupců tabulky), určující jaká data jsou v dokumentu obsažena a jak správně reprezentovat jejich datový typ. [17][18]



Obrázek 5 *Vnitřní struktura indexu (převzato [15])*

3.1.3 Mapování dat

Mapování v ElasticSearch určuje, jakým způsobem budou data indexována a jakým datovým typem budou reprezentována. V porovnání s relačními databázemi se jedná v podstatě o definici sloupců tabulky. Mapování tak udržuje informaci o struktuře, názvech a datových typech polí obsažených v dokumentech a také o tom, jak mají být indexována Lucene enginem. [17]

Jak už bylo zmíněno, data jsou do ElasticSearch indexována jako schema-less, tedy není nutné mapování předem uvádět a engine Lucene se na základě vstupních dat pokusí odhadnout správnou strukturu a datové typy. Tento přístup je u základních datových typů a struktur velmi užitečný. To platí především o jednoduchých strukturách klíč-hodnota a základních datových typech, jako jsou String, Date, Numeric nebo Boolean. Avšak u pokročilejších dat je doporučené nespoléhat na tuto automatizaci a mapování vytvářet před vytvořením indexu a první indexací. Zde jde již o více prvkové a úrovňové struktury jako je Array, Nested Array a datové typy jako jsou Geo-point, IPv4 a Attachment (datový typ určený k připojení ostatních binárních souborů). Jelikož špatně namapovaná data by negativně ovlivnila jak výsledky vyhledávání, tak jejich rychlost. [19][20]

```
{
    "name": {
        "first": "John"
    }
}
```

Zdrojový kód 1 *Struktura indexovaného dokumentu (vlastní)*

```
{
  "my_type" : {
    "properties" : {
      "name" : {
        "properties" : {
          "first" : {
            "type" : "string"
          }
        }
      }
    }
  }
}
```

Zdrojový kód 2 *Ukázka mapování pro indexovaný dokument (vlastní)*

Specifikovat vlastní mapování je možné na základě šablon, ty umožňují nastavit, v jaké podobě se index vytvoří a jsou nutné i pro nastavení ILM funkcionality. U šablon lze specifikovat název indexu, verzi (mění se při úpravě šablony), nastavení indexu (počty replik a shardů, alokace indexu, maximální hloubka automatického mapování a další), přiřadit k indexu vlastní metadata (popis, zodpovědnou osobu a další) nebo právě nastavení mapování. [21]

3.1.4 Index lifecycle management (ILM)

Řízení životního cyklu indexu bezesporu patří mezi nejpřínosnější vlastnosti konceptu ElasticSearch. Vzhledem k tomu, že data v indexu nelze mazat ani měnit (smazání a změna pouze invaliduje původní data a případně je nahradí změněnými, avšak paměť přidělená těmto datům stále zůstává obsazena) je výhodné indexy rozdělovat do menších částí. Takové rozdělení lze udělat manuálně pomocí ElasticSearch API. ILM však zvládá i automatické rozdělení, jež lze použít i opakovaně, na základě předem stanovených pravidel. Takto rozdělené indexy lze poté jednotlivě zálohovat, nebo alokovat na méně výkonné ElasticSearch nody (tento postup se nazývá How-Warm-Cold architektura). [22]

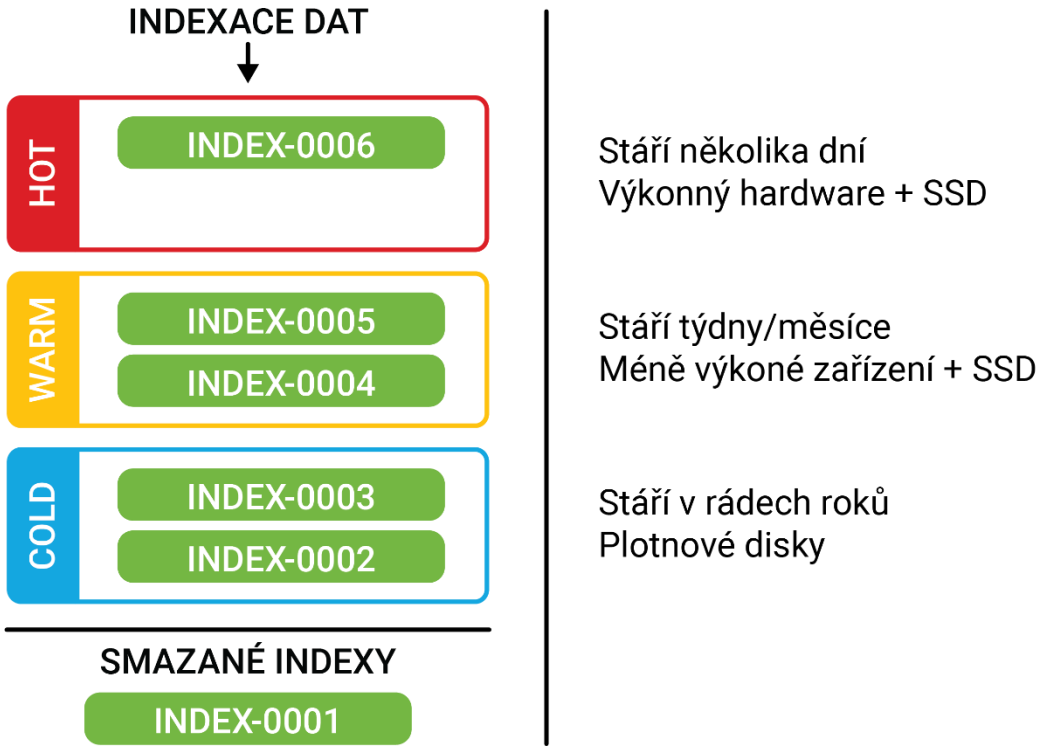
ILM pravidla jsou dělena do čtyř fází: hot, warm, cold a delete, přičemž nově vytvořený index běží vždy ve fázi hot, i bez využití ILM. Pro každou z těchto fází lze definovat sadu pravidel. Pravidla určují, kdy index vstupuje do dané fáze, a jak se v dané fázi index chová. Pořadí fází je striktně dané, ale není nutné definovat všechny fáze a lze využít i zkráceného postupu, kdy se index již při svém vytvoření nachází ve fázi hot a po splnění určitých pravidel přejde ihned do fáze delete – v té je nenávratně smazán. [22][23]

Další důležitým požadavkem pro ILM pravidla jsou šablony, jelikož při přechodu z fáze hot do jakékoliv jiné fáze je vytvořen nový index ve fázi hot, který nahradí ten původní. Vytvoření probíhá na základě šablon, které je nutné k ILM pravidlům přiřadit. Bez šablon nebude ILM fungovat správně, v důsledku čehož v aplikačním logu ElasticSearch zobrazí příslušnou chybovou hlášku. [22]

U fází ILM lze nastavit například, kdy má dojít k rollover (převrácení), zda se má změnit počet shardů nebo jestli se má index stát read-only. Rollover je operace, při které

dojde k tvorbě nového indexu dle šablony, přesměrování toku dat z původního indexu do nového a následnému převedení původního indexu z hot fáze do další předem definované fáze. Po této operaci dojde k provedení operací dle definovaných pravidel warm, cold nebo delete fáze. [22]

Jak přesně tato ILM pravidla fungují, a co u nich lze nastavit, bude nastíněno v praktické části této diplomové práce.



Obrázek 6 *Životní cyklus hot-warm-cold architektury (vlastní)*

3.2 Logstash

Logstash je vstupním nástrojem do celého ELK Stacku. Jedná se o nástroj určený ke správě, sběru, obohacování a předzpracování logů. Jeho cílem je sesbírat logy z požadovaných zdrojů a transformovat je do normalizované podoby, kterou je možné indexovat do ElasticSearch. Centrálně tak shromažďuje a zpracovává data různého typu a z různých zdrojů. Tento nástroj ELK Stacku se dá přirovnat k určitému typu trychtýře,

přes který projde každý log, než je indexován a uložen do ElasticSearch databáze nebo do jiného analytického uložiště. [24][25]

Logstash je složen z několika částí, lze jej tak rozdělit na pipeliny, jejichž obsahem jsou inputy, outputy, filtry a kodeky. Všechny tyto části se dále skládají z pluginů rozšiřujících základní funkcionalitu. [25] Logstash slouží především k realtime zpracování logů. V základním nastavení jsou všechna data zpracovávána jako in-memory. Je zde ovšem také možné zapnout i persistentní frontu. [26]

Obrázek 7 *Vnitřní architektura nástroje Logstash (vlastní)*

3.2.1 Pipeline

Architektura nástroje Logstash je založena na principu „pluggable pipeline architecture“. Stavební prvky programu jsou napojovány vzájemně na sebe a tvoří tak jakési potrubí určené pro průchod logů od vstupu až na finální výstup. [25] Logstash má pro tyto účely vytvořenou hlavní pipeline. Umožňuje ovšem vytvořit i více průchodů, které na sebe lze napojovat, slučovat nebo rozvětvovat. Ty pak utváří velice robustní nástroj pro průchod logů od zdroje až po zaindexování do ElasticSearch databáze. [27]

3.2.2 Input a Output pluginy

Jak už označení těchto pluginů napovídá, slouží jako vstup a výstup z jednotlivých pipeline. Nejedná se o celkový vstup a výstup z nástroje Logstash (i když tomu tak může být například při využití pouze hlavní pipeliny). Z tohoto důvodu je Logstash vybaven vstupně-výstupním pluginem **pipeline** – sloužící ke komunikaci mezi pipeliny. Na základě tohoto přístupu je poté možné větvit tok logů v rámci nástroje Logstash. [25][28][29]

Mezi další vstupní pluginy jsou například řazeny:

- **stdin** – určený pro vstup z příkazové řádky (primárně využíván pro odladění),
- **file** – pro čtení logů přímo z logovacího souboru,
- **azure_event_hub** – zajišťující proudové čtení přímo z nástroje Azure Event Hub. [28]

Zástupcem výstupních pluginů je pak především:

- **elasticsearch** – určený pro zaslání normalizovaných logů dále do nástroje ElasticSearch,
- **exec** – sloužící pro vyvolání příkazu, pokud log dojde až do fáze výstupu. [29]

3.2.3 Filtr plugin

Plugin typu filtr se v nástroji Logstash využívá při normalizaci dat do požadovaného formátu nebo pro obohacení dat. [25] Jedna z věcí, pro kterou lze pluginy typu filtr využít je i jednoduchá rozhodovací logika nebo pokročilejší skriptování v jazyce Ruby. K tomuto účelu lze využít plugin **ruby**. [30]

Mezi filtrační pluginy sloužící pro normalizaci dat lze zařadit například pluginy:

- **json**, **csv** – Umožňují parsovat data z formátu JSON a CSV do formátu jednotlivých datových polí.
- **byte** – Slouží pro převod číselné hodnoty datové velikosti (41 472 B), na lidsky čitelnou reprezentaci (40.5 KB) a opačně. U tohoto pluginu lze také nastavit jeho chování, zda převod probíhá s binární (1 KB = 1024 B), nebo metrickou (1 KB = 1000 B) přesností. [30]

Do kategorie filtračních pluginů pro obohacení dat pak spadá například plugin **geoip**, který využívá geolokační databázi GeoLite2 k nalezení dodatečných informací o vstupní IP adrese. Díky němu lze získat z IP adresy informace, jako jsou GPS souřadnice, stát, město a časová zóna, ke které je daná veřejná IP adresa registrována. [30]

3.2.4 Persistentní fronta

Při základní konfiguraci je Logstash nastaven na zpracovávání veškerých zpráv jako in-memory. Toto nastavení má vliv především na vyrovnávací frontu mezi vstupy (input pluginy) a zpracováním (filter + kodek pluginy). In-memory fronta uvnitř Logstashe má fixní velikost a není konfigurovatelná. Pokud dojde k nečekanému vypnutí nebo chybě Logstashe, má toto nastavení in-memory fronty za následek ztrátu všech momentálně zpracovávaných a na zpracování čekajících zpráv.

Z důvodu ochrany dat před nenadálými událostmi a abnormálnímu ukončení obsahuje Logstash funkci persistentní fronty. Persistentní fronta umožňuje ukládat na disk veškeré zprávy čekající na zpracování. Persistentní fronta je také jedinou možností zajišťující trvanlivost dat v rámci Logstashe.

Druhou vlastností persistentní fronty je využití pro vyrovnávání zátěže při abnormálních dávkách dat. Namísto nasazení a spravování zprostředkovatelů zpráv, jako jsou Redis, RabbitMQ nebo Kafka, je možné povolit persistentní frontu uvnitř Logstashe a vyhnout se tak nutnosti využívat externí zprostředkovatele zpráv jako vyrovnávací paměť. Persistentní frontu je na rozdíl od in-memory fronty možné plně konfigurovat. Jediným omezením persistentní fronty je absence replikace dat. Perzistovaná data tak nejsou chráněna proti chybě disku a mechanickému poškození. [31]

3.3 Beat

Dalším nástrojem z rodiny ELK Stacku je Beat, ten slouží jako jednoúčelový sběrač a poskytovatel zpráv. Jeho účelem je sloužit jako agent nainstalovaný na zdrojovém serveru, který sbírá data z různých zdrojů daného serveru a zasílá je do cílové destinace. Jako cílová destinace může sloužit přímo ElasticSearch, avšak nezřídka se jako cílová destinace využívá také Logstash pro předzpracování zaslaných zpráv, nebo externí zprostředkovatelé zpráv (Redis, RabbitMQ a Kafka).

Je důležité v tomto kontextu nezaměňovat Logstash a Beat jako rovnocenné nástroje. Při určitých požadavcích je možné je zaměnit, avšak při složitějších nárocích je již jejich hlavní význam patrný. Logstash slouží především jako vstupní brána pro veškerá

data a jejich předzpracování. Data do něj tak musejí být zasílána nebo stahována ve streamech. Naopak Beat slouží jako sonda stojící po boku sledované aplikace nebo zařízení a stará se pouze o zasílání dat do cílové destinace. Jelikož má být Beat jednoúčelovým agentem, je jeho funkcionalita rozdělena do několika samostatně instalovatelných instancí. Každá z těchto instancí umožňuje sběr a přeposílání jen striktně určených dat. [32][33]

3.3.1 Auditbeat

Auditbeat komunikuje přímo s frameworkem audit obsaženém v operačním systému Linux. Auditbeat obstarává sběr stejných dat, jako je možné získat voláním Linuxového démona auditd. Data je poté možné v aktuálním čase analyzovat v prostředí ElasticSearch. [34]

3.3.2 Functionbeat

Functionbeat slouží jako bezserverový agent pro monitorování cloudových služeb. Jeho hlavní výhodou je nutnost nasazení jako Function as a Service (FaaS), to umožňuje nasazení agenta, bez nutnosti spravování dalších vrstev softwaru a hardwaru. Pro účely takového nasazení existují platformy jako Amazon Lambda nebo Google Cloud Functions. [35]

3.3.3 Heartbeat

Heartbeat je sondou zajišťující zjišťování dostupnosti sledované služby. Exaktní funkcionalita spočívá v dotazování na aktivitu konkrétní URL. Toto dotazování funguje za pomoci síťového nástroje ping. Dotazování je možné skrze síťové protokoly ICMP, TCP a HTTP, a to i s podporou TLS, autentizace a proxy. Díky jednoduchému mechanismu rozlišování DNS záznamů je možné sledovat také jednotlivé služby umístěné za load-balancery. [36]

3.3.4 Journalbeat

Journalbeat obstarává sběr žurnálových zpráv z Linuxového démona journald, který je součástí démona systemd. Data je poté možné v aktuálním čase analyzovat v prostředí ElasticSearch. [37]

3.3.5 Metricbeat

Nástroj Metricbeat slouží pro sledování dostupných prostředků na hostujících zařízeních, podporovány jsou všechny typy operačních systémů Linux, Windows i Mac. Z těchto hostujících zařízení sbírá Metricbeat informace o využití CPU, pamětích, souborovém systému, discích, síťovém provozu i všech běžících procesech. Metricbeat obsahuje také moduly pro sběr metrik z hostovaných služeb jako je Apache, NGINX, MongoDB, MySQL nebo Prometheus. [38]

3.3.6 Packetbeat

Packetbeat poskytuje komplexní sběr síťového provozu. Díky nasazení agenta na hostující zařízení, nebo k hostované aplikaci lze získat data o veškeré síťové komunikaci vně i dovnitř. Sledování síťových protokolů, jako je HTTP umožňuje sledovat odezvu, dobu obsloužení síťového dotazu nebo vzory a trendy uživatelských přístupů. Při nasazení několika instancí Packetbeat skrze sledovanou infrastrukturu lze dosáhnout také komplexního náhledu na proudění komunikace skrze celou síť. [39]

3.3.7 Winlogbeat

Winlogbeat je určen pouze pro operační systém Windows a lze pomocí něj získávat data obsažená v protokolu událostí tohoto systému. [40]

3.3.8 Filebeat

Jednou z nejpoužívanějších instancí nástroje Beat je Filebeat. Popularitu si vydobyl především svou všestranností. Filebeat umožňuje číst jakákoliv logovaná data z jakéhokoliv souboru na hostujícím zařízení. Z předem specifikovaného souboru jsou

data vždy čtena po řádcích, případně je možné využít předpřipravených modulů pro logování podporovaných nástrojů, jako je nástroj Kafka nebo Docker.

Jednou z výhod, které Filebeat nabízí, je udržování ukazatele do souboru. Pokud dojde k neočekávanému krátkodobému výpadku sledovaného souboru, Filebeat agenta, nebo hostujícího zařízení, je Filebeat schopen navázat na čtení logů z místa, kdy byl soubor logů naposledy dostupný. [41]

3.4 Kibana

Kibana je posledním důležitým nástrojem z rodiny ELK Stacku. Jedná se o analytický a vizualizační nástroj. Slouží především jako grafické rozhraní pro pohodlné vyhledávání, zobrazování a práci s daty uloženými a zaindexovanými v ElasticSearch. Na základě těchto dat lze vytvářet rozmanité vizualizace v grafech, tabulkách a na mapách. Veškeré analýzy nad daty lze provádět bez nutnosti kódování a vyhledávání lze provádět pomocí Kibana Query Language (KQL). Dále lze prostředí Kibana využít pro správu ElasticSearch a monitorování celé infrastruktury ELK Stacku. Tuto funkcionalitu lze využívat také přímo přes API nástroje ElasticSearch. [42][43]

3.4.1 Kibana Query Language

KQL je proprietární dotazovací jazyk vytvořený společností Elastic N. V. za účelem jednoduchého dotazování nad daty indexovanými v ElasticSearch. Jelikož uložiště v rámci nástroje ElasticSearch je vytvořeno nad enginem Apache Lucene, který disponuje vlastní dotazovací syntaxí Lucene Query Syntax (LQS), je možné pro dotazování dat indexovaných v ElasticSearch použít také tuto syntaxi. Nevýhodou LQS je její složitost pro laické uživatele a nemožnost prohledávat vnořené objekty a skriptovaná pole.

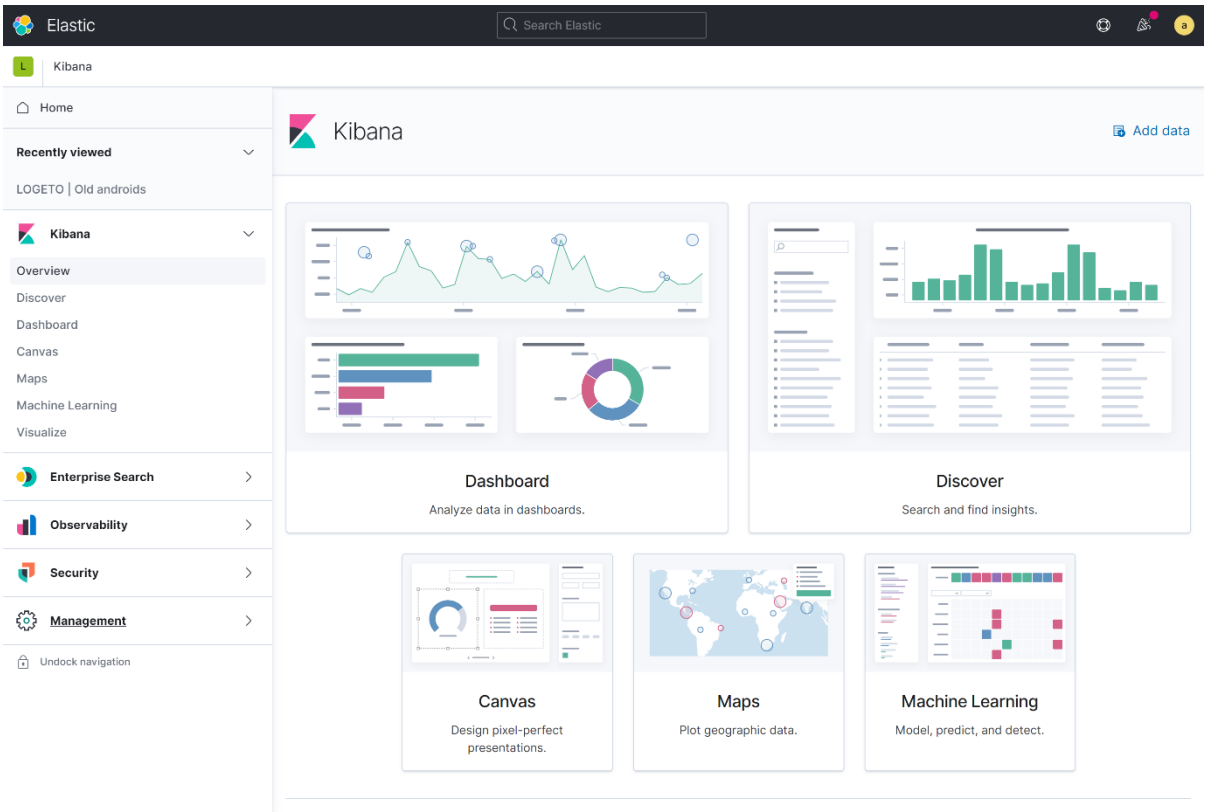
Právě z tohoto důvodu vznikl KQL, který používá výrazně jednodušší syntax, dokáže uživateli našeptávat zamýšlenou strukturu, umožňuje wildcard a odstraňuje některé nevýhody LQS. Naopak omezením KQL, je nemožnost vyhledávání na základě regulárních výrazů a fuzzy výrazů, tuto funkcionalitu nabízí pouze LQS, na který se lze kdykoliv v rámci prostředí Kibana přepínat. [44] [45]

3.4.2 Grafické rozhraní Kibana

Grafické rozhraní Kibana je rozděleno do několika logických celků. Každý z těchto celků obstarává určitou agendu logů a monitoringu, poslední z částí pak slouží pro správu celého ElasticSearch Clusteru a nastavení nástroje Kibana. Toto rozdělení neslouží jen v kontextu nástroje Kibana. Společnost Elastic N. V. jej využívá i v rámci své produktové nabídky, jako logické rozdělení svých produktů. Produkty jsou tak rozděleny na obecné sledování logů (Kibana), databázi a vyhledávání (Enterprise Search), monitoring aplikací (Observability) a bezpečnostní auditování (Security). [43]

Kibana

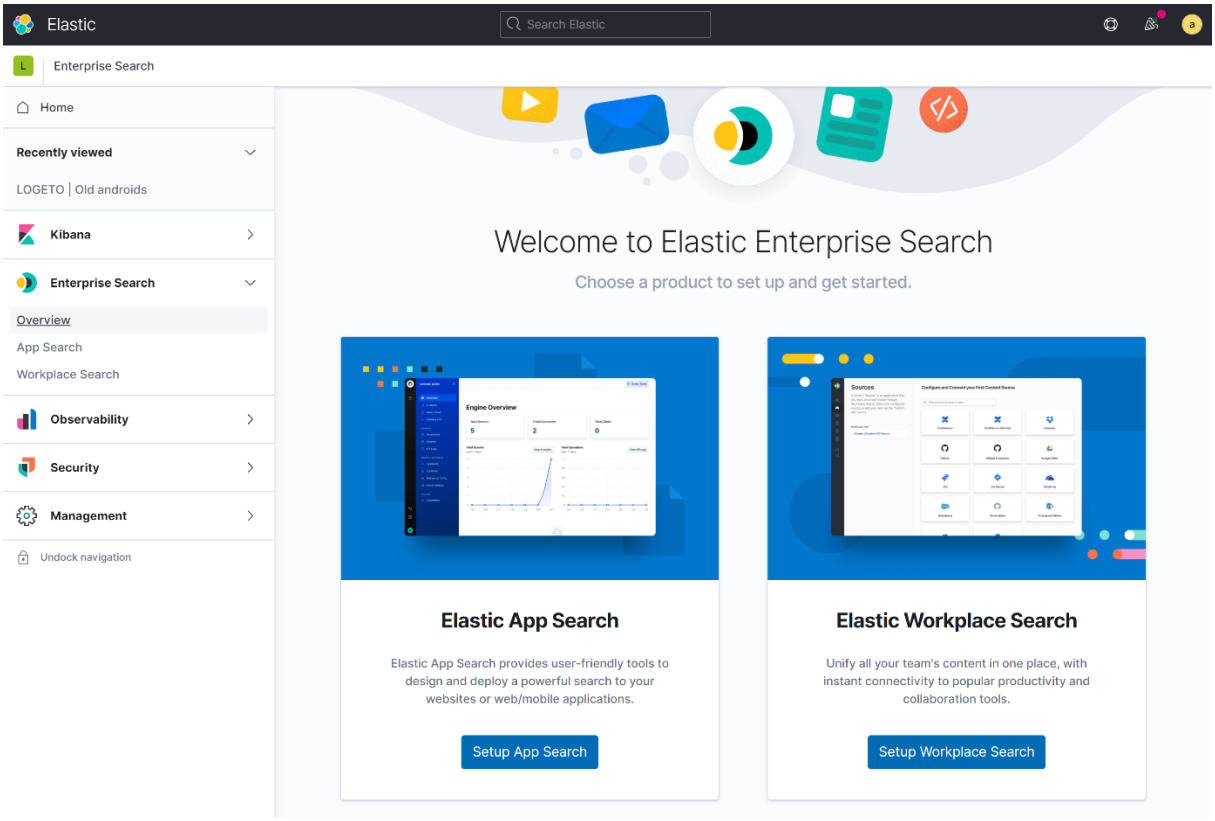
Část Kibana (neplést s celým nástrojem Kibana) je prostředí určené pouze pro vyhledávání zaindexovaných dat, tvorbu vizualizací, grafů a map. Zároveň je zde možné využít nástroj Elastic Machine Learning, který umožňuje nalézt anomálie, odlehlé hodnoty a předpovídat trendy na základě indexovaných dat. Toho lze využít pro účely rozhodování a plánování. [46][47][48]



Obrázek 8 *Záložka Kibana v grafickém prostředí Kibana (vlastní)*

Enterprise Search

Pokud je ElasticSearch využit jako databázový server pro určitý typ aplikací, pak nástroj Enterprise Search slouží jako nástroj pro jednoduchou integraci vyhledávání do aplikace. V prostředí Kibana pak část Enterprise Search slouží pro nastavování tohoto vyhledávání. Lze zde navrhnout strukturu vyhledávaných dat, určit datové typy dat, nastavit relevanci vyhledávaných položek, vytvořit přístupové údaje pro požadovanou aplikaci a především sledovat metriky o využívanosti vyhledávání. Jakékoliv vyhledávání v požadované aplikaci pak probíhá pomocí Enterprise Search. [49]

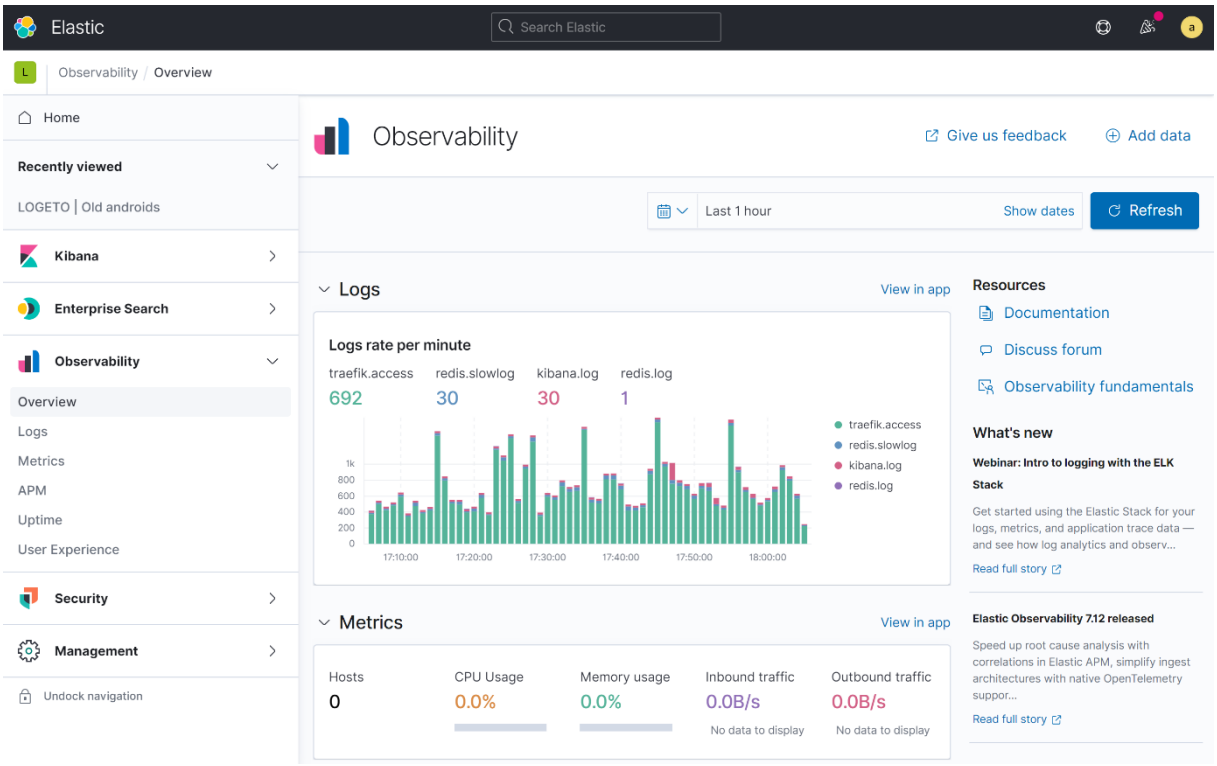


Obrázek 9 *Záložka Enterprise Search v grafickém prostředí Kibana (vlastní)*

Observability

V části Observability je možné monitorovat logy, metriky a vyhodnocovat výkon aplikací v rámci Application Performance Management (APM). Uživateli je tak umožněno sledovat události, ke kterým v prostředí monitorované aplikace dochází, a adekvátně na ně reagovat. Tato část prostředí Kibana pracuje především s daty posbíranými pomocí nástrojů Beat. U logů je to například Filebeat, u metrik pak Metricbeat a pro uptime data

je to Heartbeat. Výjimkou jsou APM data, která je nutné sbírat pomocí balíčku ElasticApmAgent dostupného pro několik programovacích jazyků. Tento balíček musí být implementován přímo v monitorované aplikaci. Balíček slouží ke sběru a auditu běžícího programového kódu a volání, které kód produkuje. V rámci javascriptových frameworků pak ElasticApmAgent sbírá také Real User Monitoring (RUM) data. [50][51]

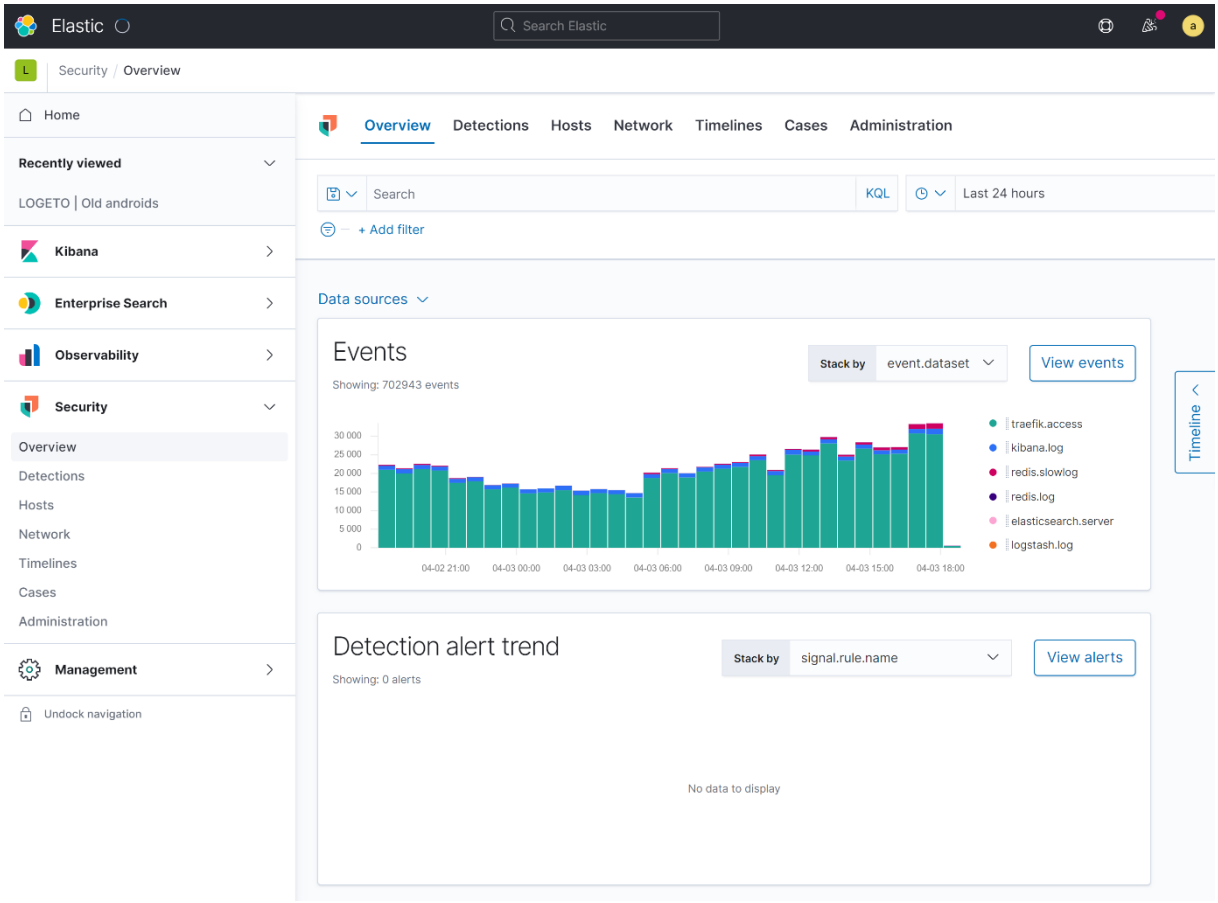


Obrázek 10 *Záložka Observability v grafickém prostředí Kibana (vlastní)*

Security

Část Security poskytuje analýzu umožňující prevenci, detekci a reakci na bezpečnostní hrozby. Lze zde sledovat a auditovat data proudící skrze koncové body infrastruktury, události z operačního systému hostitelských počítačů, autentizační logy a nahlížet na proudění dat v infrastruktuře v rámci k tomu uzpůsobeného UI. Data pro

tuto část jsou opět sbírána pomocí nástroje Beat, v tomto případě je to konkrétně Filebeat, Auditbeat a Packetbeat. [52][53]

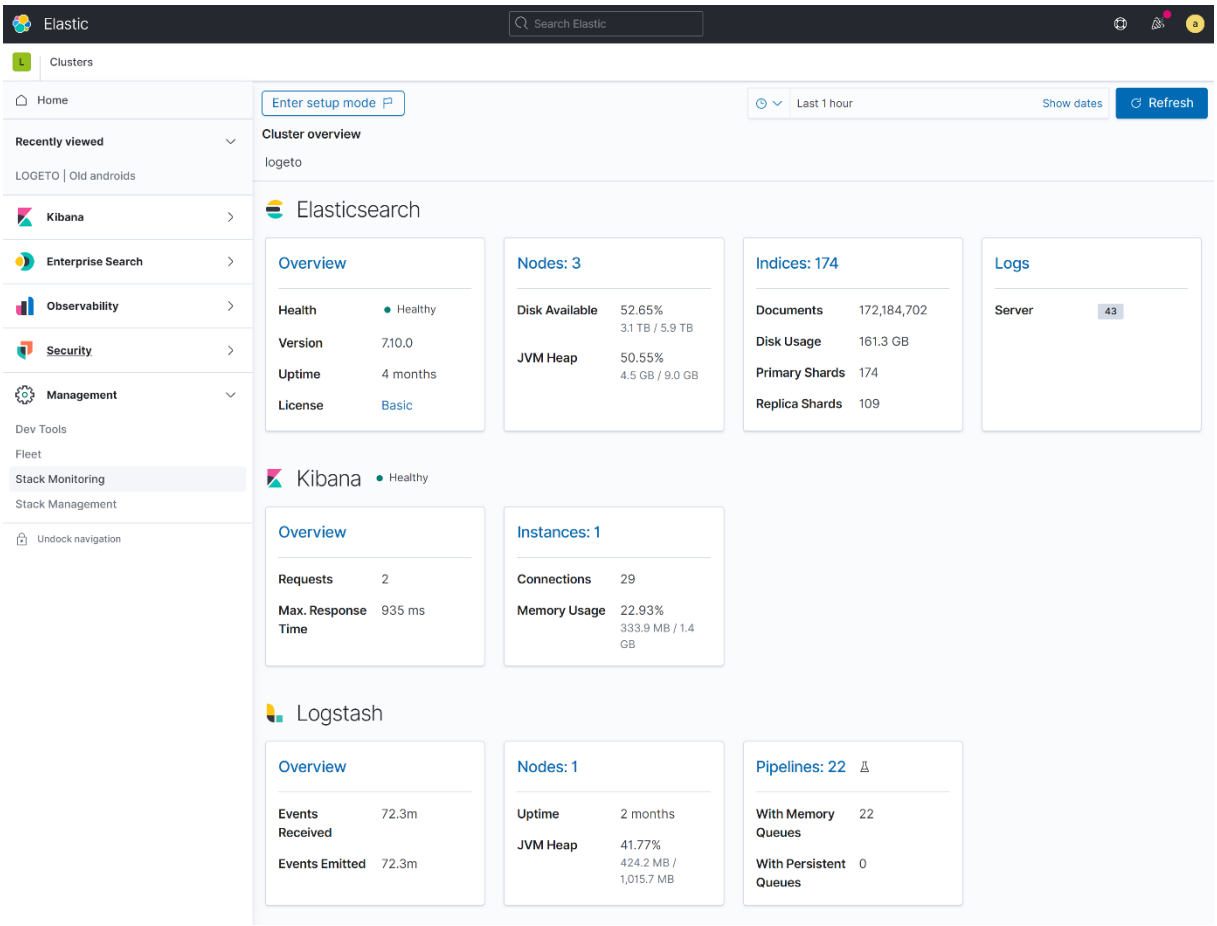


Obrázek 11 *Záložka Security v grafickém prostředí Kibana (vlastní)*

Management

V rámci Managementu lze sledovat stav celého ElasticSearch clusteru, spravovat mapování dat, zasílání zpráv, provádět zálohy, obnovy a aktualizace celého ElasticSearch clusteru nebo nastavovat samotné prostředí Kibana. Zároveň se v části Management nachází vývojářské nástroje jako je konzole, profiler a sandboxové prostředí (Painless Lab). Pro následné účely implementační části budou využívány především části pro mapování dat, ILM a zálohování. V požadavcích implementace je také zasílání zpráv (alerting), které se taktéž nachází v části Management. Slouží pro účely zasílání varovných správ ze všech výše popsaných částí. Bohužel funkcionalita zasílání zpráv je dostupná

pouze s placenou licencí ELK Stacku, proto byl pro tyto účely vybrán jiný nástroj, kterému je věnována následující kapitola. [54]



Obrázek 12 *Záložka Management v grafickém prostředí Kibana (vlastní)*

4 ElastAlert

ElastAlert je open-source nástroj od společnosti Yelp, vytvořený v jazyce Python. Je určený k zasílání varovných zpráv z nástroje ElasticSearch. Na základě tohoto nástroje lze z ElasticSearch získávat indexovaná data, sledovat jejich přírůstky a využívat tato data k vyhodnocování anomálií, špiček nebo jiných typů zajímavých vzorců. Na základě takto vyhodnocených dat lze zasílat varovné zprávy do různých typů komunikačních nástrojů, jako je e-mail, Slack či obecné webhooky.

Vnitřní struktura tohoto nástroje je tvořena službou obstarávající dotazy na ElasticSearch a dvěma typy komponent, pravidla (rules) a varovné zprávy (alerts). Každá z těchto komponent, je tvořena Python moduly, které je možné v případě potřeby rozšířit o vlastní funkcionalitu. Této problematice se věnuje celá jedna kapitola praktické části této diplomové práce.

Veškeré konfigurace nástroje ElastAlert probíhají prostřednictvím konfiguračních souborů yaml, které inicializují konkrétní nastavení při startu. Avšak pro laické uživatele existují také pluginy do prostředí Kibana, jež umožňují konfiguraci nástroje ElastAlert za běhu, v rámci GUI. Tyto pluginy však bohužel nejsou spravovány přímo společností Yelp a často se jedná pouze o nástroje od anonymních členů komunity. [55][56]

4.1 ElasticSearch querying

Jelikož ElastAlert není integrován v rámci ElasticSearch, ale jedná se o samostatně běžící službu, je nutné data načítat. Toto načítání probíhá pomocí dotazování na ElasticSearch API. To, na jaký koncový bod a jak často se ElastAlert má dotazovat, je možné nastavit v základní konfiguraci. Lze zde specifikovat, zda má komunikace s ElasticSearch probíhat šifrovaně, přiřazovat certifikáty nebo přihlašovací údaje pro autentizaci. Kromě toho je zde nutné nastavit složku, ve které se nacházejí pravidla (rules).

Kompletní soupis všech konfigurovatelných parametrů je možné nalézt v rámci rozsáhlé dokumentace (viz [55]). Níže jsou blíže specifikovány ty zásadní v kontextu praktické části této diplomové práce:

- **rules_folder** – Cesta ke složce v hostitelském kontejneru, ve které se nacházejí konfigurace pravidel. ElastAlert načte všechny konfigurační yaml soubory nacházející se v této složce. Složka musí obsahovat alespoň jedno definované pravidlo, jinak se ElastAlert nespustí.
- **run_every** – Jedná se o parametr definující frekvenci dotazů na data indexovaná v ElasticSearch. Hodnota je určena vnořeným parametrem **seconds**, **minutes**, **hours** a další, dle požadované časové jednotky a její hodnoty.
- **es_host, es_port, es_username, es_password** – Tyto parametry slouží k určení přístupového bodu k ElasticSearch, jeho portu a přihlašovacích údajů pro autentizaci uživatele.
- **use_ssl, ca_certs** – Těmito parametry pak lze zapnout šifrovanou komunikaci a cestu k certifikátu, která má být pro tuto komunikaci využita.
- **writeback_index** – Za zmínku stojí jistě také parametr writeback_index, jenž určuje název indexu v rámci ElasticSearch, který si ElastAlert vytvoří pro vlastní potřeby zaznamenávání metadat. Tento index a data v něm lze v prostředí Kibana dále využít pro analýzu chování nástroje ElastAlert a odhalení chyb.

4.2 Rules

Rules (pravidla) tvoří hlavní logiku nástroje ElastAlert. Všechna pravidla jsou definována jako yaml konfigurace. Uložena mohou být v libovolné složce v hostitelském kontejneru. Cesta k této složce musí být uvedena v konfiguraci nástroje ElastAlert pod parametrem **rules_folder**. Pro každou provedenou query se vždy vyhodnocují všechna specifikovaná pravidla, jejichž nastavení lze ovlivnit velkým množstvím parametrů. Mezi několik parametrů, které stojí za zmínku, patří například name, index, filter, type a alert. [57]

- **name** – Tento parametr slouží k identifikaci pravidla dle názvu (lze jej vidět v rámci dat, indexovaných do writeback indexu).

- **index** – Index určuje, ke kterému indexu uvnitř ElasticSearch, mají být dotazy tohoto pravidla směřovány.
- **alert** – Parametr určující typ alertu, který má být použit k zaslání zprávy, a jeho nastavení pomocí vnořených parametrů.
- **type** – Type slouží k nastavení logiky samotného pravidla. Lze nastavit, zda se má pravidlo chovat jako **blacklist/whitelist** (vyvolávat alert pouze pokud se objeví/neobjeví výskyt dané zprávy), **new_term** (vyvolává alert pouze pokud přijde zpráva, která se ještě neobjevila) nebo **percentage_match** (vyvolává alert, pokud je v daném časovém okně zvýšen výskyt určených zpráv, překračujících stanovenou procentuální hranici). Takovýchto typů pravidel existuje mnohem více a lze je všechny nalézt v rámci oficiální dokumentace nástroje ElastAlert.
- **filter** – Poslední popisovaný parametr nastavuje samotný dotaz zasílaný do ElasticSearch. Lze si opět vybrat mezi velkou řadou typů, které filtry poskytují. Například **query** (jehož obsahem musí být validní ElasticSearch Query), **range** (umožňující dotaz na data obsahující atribut v daném rozsahu) nebo **term** (obsahující název a hodnotu atributu, který má být obsahem zprávy dotazované z ElasticSearch).

4.3 Alerts

Alerty již řeší pouze samotné zasílání vyhodnocených zpráv na koncové komunikační nástroje. Pro každé pravidlo lze určit několik alertů, jejichž počet není omezen a pro každý z nich je možné nastavit jiný formát zasílaných zpráv, dle možností konkrétního alertu. ElastAlert poskytuje celkem 26 typů alertů pro různé komunikační nástroje. [58]

- **command** – Tento typ alertu umožňuje spouštět příkazy v rámci hostitelského systému. Na základě vyhodnocených pravidel například vyvolat script, který provede požadovanou funkcionalitu. Avšak při zvolení kontejnerizované varianty nástroje ElastAlert je použití tohoto alertu velmi omezené.
- **email** – Alert email slouží pouze k jednoduchému zaslání obsahu vyhodnoceného pravidla na určenou e-mailovou adresu. K zaslání je využito SMTP serveru

určeného pomocí parametrů **smtp_host** a **smtp_port**, pokud nejsou tyto parametry specifikovány je využita přednastavená hodnota localhost.

- **jira** – Alert typu jira umožňuje otevřít nový úkol v issue tracking nástroji Jira na základě vyhodnoceného pravidla. Tento typ alertu je společností Yelp pravidelně udržován a obsahuje již velké množství parametrů ovlivňujících, jaká data a v jakém formátu budou do úkolu zapsána. Mezi základní parametry patří například **jira_server** a **jira_project,** určující konkrétní Jira server a projekt, na kterém se má úkol otevřít. Zároveň je potřeba přihlášení k danému Jira serveru, což zprostředkovává parametr **jira_account_file**, v němž musí být uvedena cesta k yaml souboru obsahujícím přihlašovací údaje (username uživatele a heslo).
- **slack** – Dalším velice používaným a udržovaným alertem je slack. Ten, jak název napovídá, slouží k zasílání zpráv do komunikačního nástroje Slack. Zde zasílání probíhá pomocí webhook url, která obsahuje informace o cílovém serveru, autentizační data a identifikační číslo komunikačního kanálu, pro něhož je zpráva určena. Díky udržovanosti tohoto alertu jsou opět dostupné rozsáhlé možnosti úprav zasílané zprávy.
- **ms_teams** – Posledním typem alertu je ms_teams, což je rovněž typ alertu, jímž se detailněji zabývá praktická část této diplomové práce. Možnosti úprav u tohoto typu alertu jsou bohužel velice omezené, jediným parametrem určeným pro personalizaci zasílané zprávy je zde **ms_teams_theme_color,** upravující barvu oznámení u dané zprávy. Zbylé parametry slouží pouze k nastavení komunikace s MS Teams, jediným povinným parametrem je zde pak **ms_teams_webhook_url**, který musí obsahovat webhook url, skládající se ze stejných informací, jako u alertu pro nástroj Slack.

5 Elastic Cloud on Kubernetes

Společně se zavedením logování a monitoringu byla ve společnosti Systemart s. r. o. implementována také vlastní cloudová infrastruktura založená na technologii Kubernetes. Díky této skutečnosti bylo rozhodnuto využít pro logování a monitoring taktéž cloudové varianty všech nástrojů. V tomto konkrétním případě se jedná o Elastic Cloud on Kubernetes, variantu ELK Stacku složenou z nástrojů (Logstash, Beat, ElastiSearch a další) upravených pro fungování v kontejnerizačním prostředí.

ELK Stack je možné využívat v několika variantách. Hlavní propagovanou variantou je Elastic Cloud. Jedná se o kontejnerizovanou variantu aplikace, která je spravována přímo společností Elastic v rámci cloudových služeb Google Cloud Services a Amazon Web Services, případně lze využít varianty správy přímo společnostmi Google a Amazon, taktéž v rámci jejich cloudových služeb. Tato varianta je ve všech svých licencích placená, v ceně jsou zahrnuty i náklady na cloudové služby. [59]

Další variantou je Elastic Cloud on Kubernetes, jedná se v podstatě o stejnou variantu, avšak uživatel je nucen si zařídit vlastní cloudovou infrastrukturu pomocí nástroje Kubernetes. To má za následek nutnost najmutí osob se znalostmi těchto technologií a vyšší náklady na provoz vlastní cloudové infrastruktury. Výhodou je však možnost využití bezplatné licence. Bezplatná licence nabízí veškerou základní funkcionalitu. Mezi funkce nedostupné v této licenci patří například mezi clusterová replikace nebo Kibana Actions (zasílání událostí do komunikačních platforem, e-mailů a webhooků). [60]

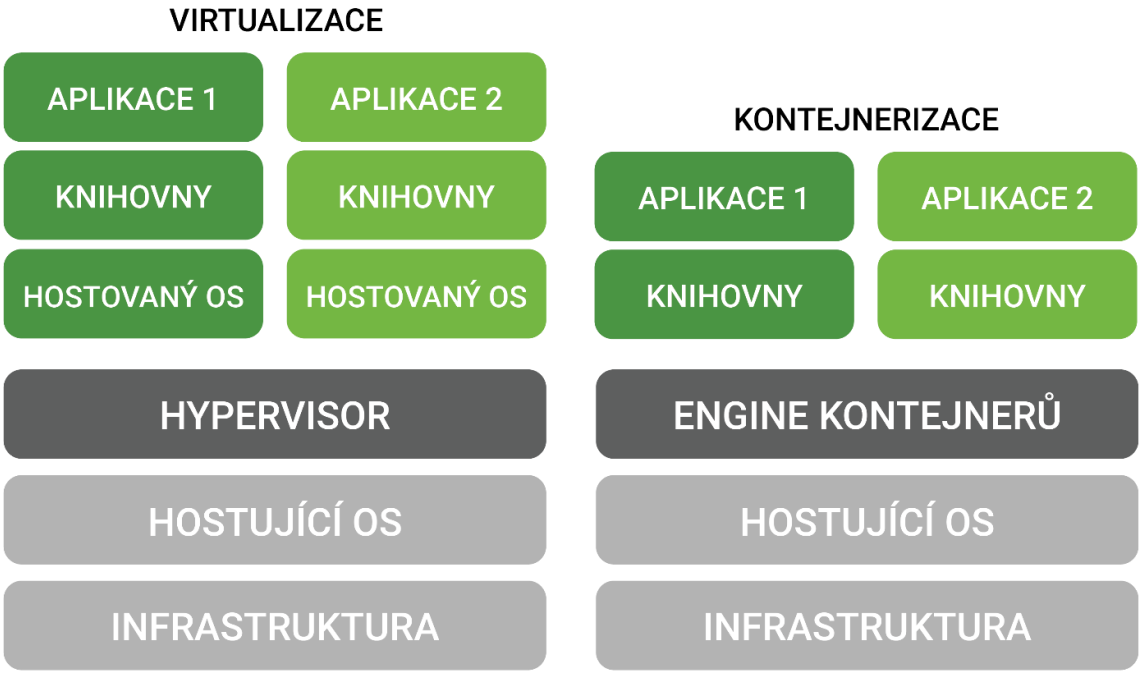
Poslední dostupnou variantou jsou samostatně šiřitelné nástroje z rodiny ELK Stacku. Všechny tyto nástroje lze využít buď v kontejnerizované podobě, nebo jako spustitelné JAR (Java ARchive) soubory zajišťující určitou formu multiplatformnosti díky JRE (Java Runtime Environment). Licencování u těchto variant je shodné s variantou Elastic Cloud on Kubernetes. [11]

Z důvodu využití kontejnerizované varianty ELK Stacku je vhodné popsat jak kontejnerizace a samotná orchestrace kontejnerů v rámci nástroje Kubernetes probíhá. Tomuto popisu jsou věnovány následující kapitoly.

5.1 Kontejnerizace

Kontejnery jsou softwarové balíčky poskytující prostředí pro nasazení a spuštění aplikace. Jejich obsahem jsou zdrojové kódy, systémové knihovny a systémové nástroje, potřebné pro běh kontejnerizované aplikace. Běžné aplikace je nutné instalovat a spouštět přímo v operačním systému hostitelského počítače. Kontejnerizované aplikace se oproti tomu spouštějí v kontejnerizačním prostředí. Díky tomu lze spouštět různé verze kontejnerizovaných aplikací souběžně. Kontejnerizované aplikace si taktéž vzájemně neblokují stejný prostor v souborovém systému, síťové komunikaci a podobně.

Tato izolace od okolního softwaru a nezávislost na konkrétním prostředí umožňuje provádět jednoduché přesuny kontejnerizovaných aplikací do jiného prostředí a také usnadňuje práci vývojářům. V případě přesunů do jiného prostředí je možné kontejnerizované aplikace přesouvat mezi hostitelskými počítači, z lokálního prostředí na produkční nebo mezi cloudovými platformami.



Obrázek 13 *Rozdíl mezi virtualizací a kontejnerizací (převzato z [61])*

Na první pohled by se mohlo zdát, že kontejnery fungují jako virtuální stroje, avšak jedná se o velice rozdílné architektury. Obě sice slouží pro virtualizaci prostředí. Virtuální stroje však pro svůj chod vyžadují nejen aplikace, ale i celý operační systém. Oproti tomu kontejnery sdílejí jádro hostitelského operačního systému s dalšími kontejnery a procesy běžícími v tomto jádře. Každý kontejner je reprezentován samostatným procesem v uživatelském prostoru. Díky tomu jsou kontejnerizované aplikace nezávislé na prostředí a velice úsporné jak na výpočetní prostředky, tak na uložiště.

Virtuální stroje navíc často dosahují velikosti několika gigabytů, zato kontejnery svou velikostí příliš nepřevyšují obsaženou aplikaci, tedy několik stovek megabytů (zdrojové kódy, knihovny a další). Nevýhodou kontejnerů je pak nižší zabezpečení. Díky sdílení jádra hostitelského operačního systému není zajištěna taková úroveň izolace přístupu k prostředkům jako v případě virtuálních strojů. [61][62]

5.2 Docker

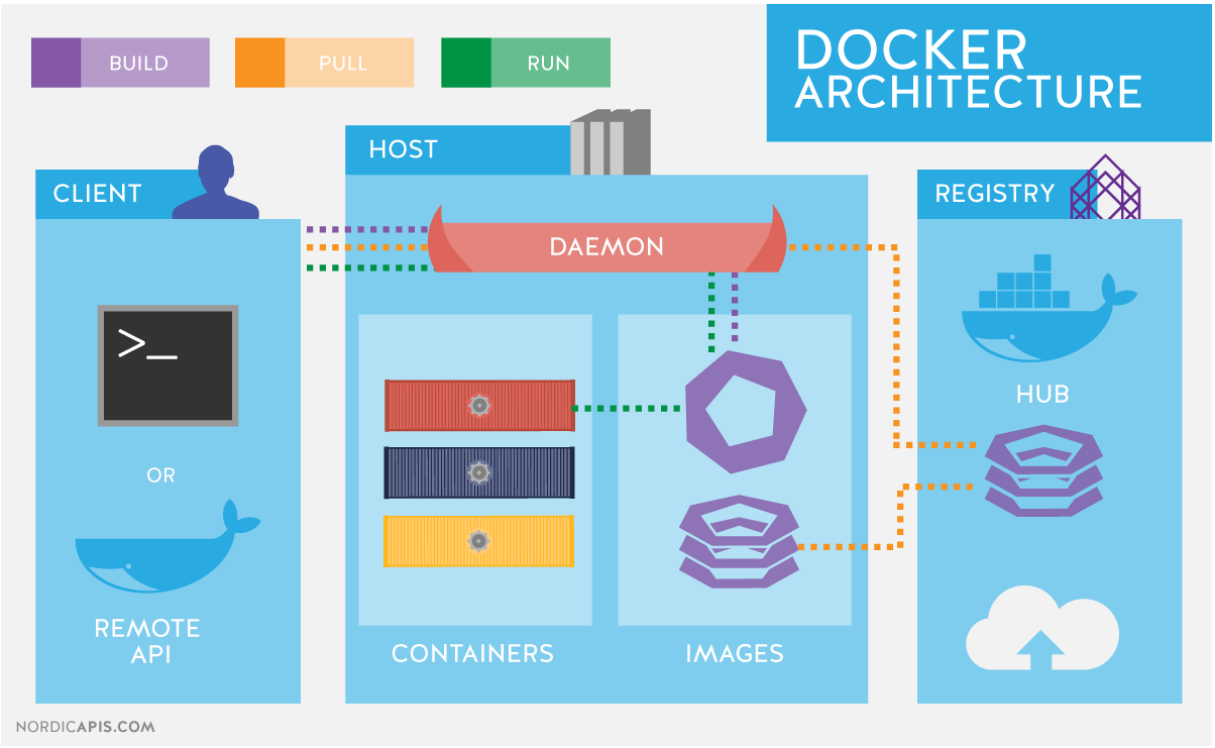
Docker je open-source platforma určená pro vývojáře a administrátory k vývoji, nasazení a spouštění kontejnerizovaných aplikací. Platforma Docker je vyvíjena stejnojmennou společností ve spolupráci s předními technologickými společnostmi jako je Microsoft, Rad Had, IBM, Google nebo Amazon. [63] Pro tyto společnosti je problematika kontejnerizace velice významná z důvodu škálování vlastních aplikací. Podle výsledků každoročního průzkumu portálu Stack Overflow je Docker nejoblíbenější, nejpoužívanější a současně nejžádanější platformou pro správu kontejnerů. Tohoto ročníku průzkumu se v uplynulém období (2020) zúčastnilo téměř 65 000 uživatelů z celého světa, a to především z řad programátorů. [63]

Platforma Docker se skládá z klienta, registru a služby Docker deamon (dockerd). Klient umožňuje uživateli prostřednictvím příkazové řádky komunikovat se službou dockerd. Touto cestou je možné například spouštět kontejnery, sestavovat obrazy a vyvolávat příkazy uvnitř kontejnerů. [64]

Registr je služba zabezpečující ukládání a distribuci sestavených Docker obrazů (Docker image). Registr je dále dělen do veřejných a soukromých repozitářů. Stejné obrazy mohou být v rámci jednoho repozitáře děleny do více verzí na základě značek

(tagů). Sestavené obrazy je dále možné díky vzdáleným repozitářům sdílet mezi uživateli. Příkladem takového repozitáře je Docker Hub, provozovaný společností Docker. Docker Hub obsahuje desítky tisíc obrazů od různých uživatelů a společností a je nastaven také jako výchozí vzdálený repozitář pro Docker. [64]

Dockerd je služba běžící na hostitelském operačním systému. Úlohou této služby je spravovat veškeré komponenty a prostředky Dockeru. Mezi tyto komponenty a prostředky patří obrazy, kontejnery, sítě a uložiště. [64]



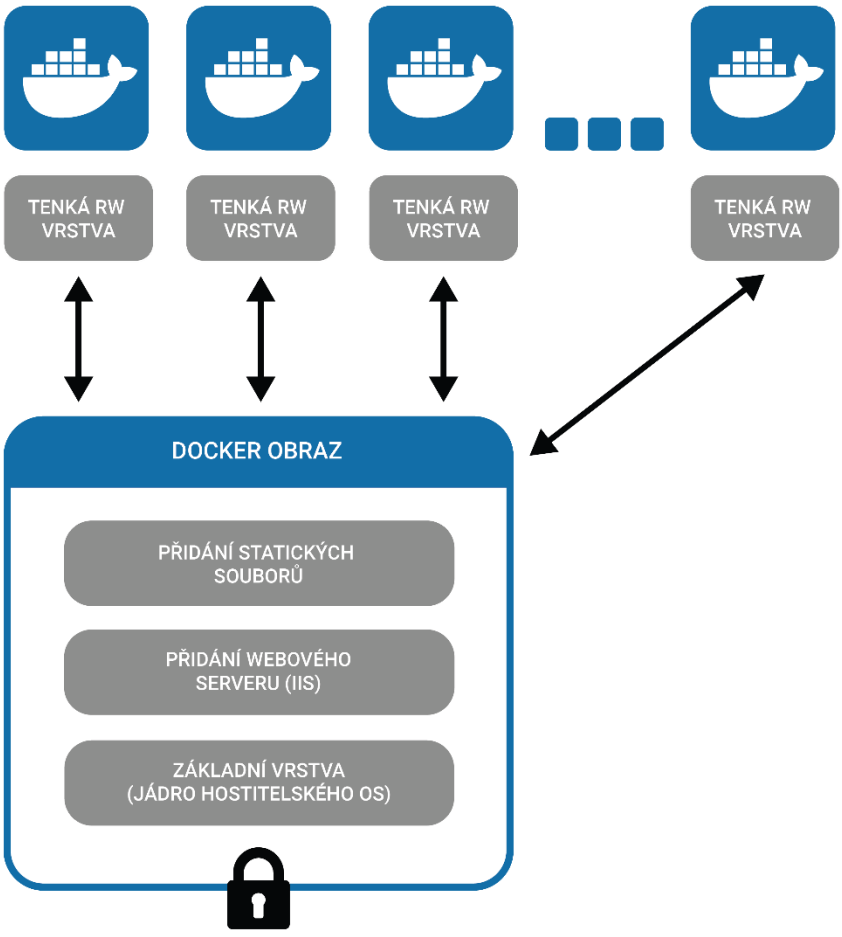
Obrázek 14 *Architektura Dockeru (převzato z [65])*

5.2.1 Docker obraz

Docker obraz (Docker image) je šablona, na základě které jsou spouštěny nové kontejnery. Každý Docker obraz se skládá z několika vrstev určených jen ke čtení. Každý obraz je tvořen první vrstvou, která vychází ze základního obrazu. Jako základní obraz často slouží oficiální obrazy, jako jsou Windows Server Core, Ubuntu nebo CentOS. Pro základní obraz lze využít jakýkoliv jiný obraz obsahující i další vrstvy. Druhou variantou

je tvorba obrazu od nuly. Každá další vrstva přidává určitou požadovanou funkcionalitu a jedná se pouze o množinu rozdílů souborového systému od předchozí vrstvy.

Při vytváření jednotlivých kontejnerů je k vrstvám obrazu přidána nová vrstva, do které lze i zapisovat. Tato vrstva je jedinečná pro každá běžící kontejner a ukládají se do ní změny provedené za běhu. Pokud dojde k odstranění kontejneru je tato vrstva smazána včetně jejích dat, ostatní vrstvy obrazu zůstávají nezměněny. Díky této architektuře je možné sdílet jeden obraz mezi více kontejnerů a je tak dosaženo nízkých výpočetních nároků, rychlého spouštění a nízkých nároků na místo v uložišti. [10]



Obrázek 15 *Architektura Docker obrazu (převzato z [10])*

5.3 Kubernetes

Kubernetes (zkráceně k8s) je open-source platforma určená pro správu velkého množství kontejnerů a jejich orchestraci. Důvodem vzniku byla potřeba zjednodušit

správu velkého množství kontejnerů v rámci infrastruktury společnosti Google a usnadnit tak škálování. Původní návrh této platformy vycházel z nástroje Borg, jednalo se o interní nástroj společnosti Google, na kterém si společnost vyzkoušela specifické koncepty správy kontejnerů a nabrané zkušenosti poté použila k tvorbě platformy Kubernetes. V roce 2014 si společnost Google začala uvědomovat význam platformy Kubernetes pro odvětví kontejnerizace a rozhodla se ho darovat organizaci Cloud Native Computing Foundation (CNCF) pod open-source licencí. CNCF spolu s početnou komunitou uživatelů dále tuto platformu rozvíjí a stojí za její popularizací v rámci celého odvětví. [65]

Kubernetes nabízí API rozhraní, skrze které se popisuje požadovaný stav kontejnerizovaných aplikací a okolní infrastruktury. Kubernetes následně obstará potřebné kroky pro přizpůsobení infrastruktury do požadovaného stavu. Dokáže tak nasadit skupinu kontejnerů, replikovat je dle potřeb a obstarávat jejich automatické zotavení při selhání některých kontejnerů. [10]

Požadovaný stav infrastruktury je nutné specifikovat pomocí definovaných konfiguračních souborů (tzv. manifestů) ve formátu YAML, nebo JSON.

5.3.1 Resources

Pro základní pochopení principu, jak Kubernetes funguje, a také pro účely implementační částí této diplomové práce, je dobré vysvětlit, co jsou to prostředky (tzv. resources) z pohledu Kubernetes. Mezi základní a nejpoužívanější prostředky patří Pods, Services, Namespaces, Secrets atd. [67]

Pod

Pod je tím nejzákladnějším prostředkem v Kubernetes, který lze vytvořit a spravovat. Jedná se o logickou jednotku jedné běžící aplikace nebo běžícího procesu v rámci Kubernetes. Tato jednotka se skládá z jednoho či více kontejnerů vzájemně sdílících stejnou konfiguraci a veškeré zdroje (uložiště, IP adresace a další). Vnitřní komunikace v rámci podu probíhá na základě síťových portů. Pomocí IP adresace probíhá pouze externí komunikace mimo pod. Kubernetes se automaticky stará o vytváření a ničení podů a současně udržuje jejich aktuální konfiguraci. Tato automatizace s sebou

přináší nevýhodu, v důsledku které jsou pody ničeny i při selhání, nebo dochází k nedostatku prostředků na Kubernetes nodu. Z toho důvodu je doporučené a mnohem častější využití controllerů pro správu, škálování a aktualizací jednotlivých podů. Tyto controllery zabezpečují obnovování a přesuny podů napříč několika nody Kubernetes, pokud dojde k selhání, nebo nastane nedostatek prostředků na jednotlivých nodech Kubernetes. [10][67]

Mezi příklady controllerů, které obsahují jeden nebo více podů patří:

- **Deployment –** Tento typ controlleru obstarává udržování aktuálního stavu konfigurace a uvedení podů do nového stavu konfigurace, případně vrácení změn v případě selhání nové konfigurace.
- **StatefulSet –** StatefulSet je specializovaný controller určený pro správu podů, které vyžadují persistentní uložiště. Často se využívá například pro databázové aplikace nebo v rámci ElasticSearch.
- **Job a CronJob –** Tyto controllery umožňují spouštění naplánovaných úloh, jako jsou zálohy či sady testů nad aplikacemi.

Service

Jelikož jsou pody vytvářeny a ničeny dle potřeby, není zaručeno přidělení stejné IP adresace podu. Z tohoto důvodu postrádá smysl využívat pro komunikaci mezi pody, nebo mimo cluster IP adresy podů. Pro tyto účely tak vznikly služby (Services), které pody seskupují a zajišťují komunikaci mezi pody a do veřejné sítě. IP adresa služby je tak stejná po celou dobu existence služby. To, zda jde o veřejnou, či privátní IP adresu pak určuje konkrétní typ služby a její konfigurace. Pody jsou ke službě přiřazovány pouze na základě štítků a selektorů (Labels and Selectors) ve formátu klíč-hodnota. Interní komunikace mezi pody v rámci clusteru pak může probíhat i pouze na základě názvů služeb a příslušných portů podů přiřazených ke službě. Služby jsou dále dle svého využití děleny na ClusterIP, NodePort, LoadBalancer a ExternalName. [10][68]

- **ClusterIP –** Je defaultním typem služby, který se používá k vystavení služby na interní IP adrese clusteru. Tato adresa není dostupná z vnější sítě.

- **NodePort –** NodePort slouží k otevření síťových portů na nodech clusteru. Služba tak může přes tyto porty komunikovat s vnější sítí.
- **LoadBalancer –** LoadBalancer plní stejnou funkcionalitu jako ClusterIP, avšak navíc umožňuje propojení s externím load balancerem, firewallem a směrovacími pravidly, které vždy vytváří poskytovatel cloudového řešení, například AWS, Azure nebo GCP.
- **ExternalName –** Tento typ služby pomáhá mapovat doménové jméno externí služby (například externě hostovaná databáze mimo cluster) na jméno služby. Ostatní služby a pody v clusteru poté nesměrují komunikaci na cílové URL, ale na jméno této služby.

Mezi další prostředky, které jsou důležité pro pochopení konfigurací uvedených v praktické části této diplomové práce, patří:

- **Namespace –** Tento prostředek slouží jako virtuální prostor clusteru. Umožňuje cluster rozdělit na prostory dle týmů, projektů nebo také k oddělení vývojových a produkčních prostředí.
- **Volume –** Svazek (Volume) je typ prostředku, který vyhradí určené místo na hostitelském serveru, přičemž může být toto místo následně připojeno k jednotlivým podům. Využívá se především jako persistentní uložiště, které není smazáno při restartu kontejnerů obsažených v podech.
- **Secret –** Obsahuje citlivé informace, jako jsou hesla, TLS certifikáty, autentizační tokeny a SSH klíče.

6 Aplikace Logeto

Iniciálním impulzem pro tvorbu této práce byl požadavek společnosti Systemart s.r.o. vyvíjející docházkovou aplikaci Logeto (na českém a slovenském trhu známá pod názvem Výkaz práce). Společnost v době iniciování požadavku začínala s přechodem na cloudové řešení. Požadavkem bylo, aby v rámci tohoto přechodu bylo zavedeno také centralizované sbírání a analýza logů ze všech částí aplikace.

Samotnému přechodu na cloudové řešení se věnoval Ing. Tomáš Burda, ve své diplomové práci *Využití Dockeru pro správu aplikací v kontejnerech*, na tuto práci a již fungující cloudovou infrastrukturu navazuje tato práce s názvem *Logování, Monitoring a Alerting za použití ELK Stack*, zabývající se implementací centralizovaného sběru a analýzy logů. Několik následujících odstavců je věnováno struktuře aplikace Logeto a průběhu jejího vývoje, tak aby měl čtenář lepší představu o požadavcích na sběr logů.

6.1 Popis aplikace

Aplikace Logeto slouží k jednoduché evidenci práce a docházky na pracovišti, na služebních cestách nebo z domova. Jednotlivé části aplikace jsou postaveny na technologiích ASP.NET, Xamarin, Universal Windows Platform a pro persistenci dat je využíván Microsoft SQL server. Vyvíjena je společností Systemart s.r.o., která byla založena v roce 2003. Společnost nyní sídlí v Hradci Králové s dalšími pobočkami v Praze a Brně.

Aplikace je rozdělena do několika agend. Každá z agend slouží k zadávání a vyhodnocování určitého typu zadávaných údajů. To, které agendy se uživatel rozhodne využívat, je plně na jeho preferencích. Aplikace obsahuje velice rozsáhlé možnosti personalizace, tak aby vyhovovala téměř každému typu pracovního úvazku. V následujících kapitolách jsou jednotlivé agendy popsány detailněji. [10][69][70][71]

6.1.1 Výkaz práce

Agenda Výkaz práce je hlavní agendou celé aplikace a slouží pro evidenci odpracovaných hodin každého z pracovníků s možností rozepisování na jednotlivé

zakázky a jejich části. Dále lze záznamy Výkazu práce propojovat s dalšími agendami díky zakázkám, fakturovatelnosti nebo fakturačním a nákladovým sazbám. K záznamům Výkazu práce je možné připojovat také uživatelsky konfigurovatelná pole. Jako výstup z této agendy slouží přehledné exporty a tiskové sestavy, které poskytují přehled odpracovaných hodin, pracovní výkazy a podklady pro výpočet mezd. [10][70][71]

6.1.2 Docházka

Agenda Docházka umožňuje v aplikaci zaznamenávat přítomnost na pracovišti a případné absence s přestávkami. Při zaznamenávání v této agendě dochází k automatickému ověřování momentální lokality, aplikace tak zamezuje podvržení přítomnosti na pracovišti. Díky těmto vlastnostem lze získat přehled o přítomnosti pracovníků na pracovišti a věrohodně je spojit s vykázaným podkladem práce. [10][70][71]

6.1.3 Kniha jízd

Kniha jízd umožňuje zaznamenávat jízdy s jednotlivými vozidly. Stejně jako u agendy Výkaz práce lze záznamy z Knihy jízd propojovat s ostatními agendami za pomoci zakázek, fakturovatelnosti nebo fakturační a nákladové sazby. Kniha jízd navíc disponuje evidencí prostojů a ujeté vzdálenosti. Další důležitou funkcí Knihy jízd je automatické generování záznamů, s využitím Výkazu práce, Docházky, výpisu navštívených lokalit a pomocných záznamů, sloužících jako průjezdní body. [10][70][71]

6.1.4 Výdaje

Agenda výdajů slouží pro evidování ostatních výdajů a výnosů, které nelze jednoznačně zařadit do agend Výkazu práce nebo Docházky. K záznamům výdajů lze opět doplnit velké množství dodatečných informací, jako jsou zakázky, sazby nebo přiřazení k daným klientům. [10][70][71]

6.1.5 Zakázky

Zakázky slouží pro strukturování vykázaných záznamů z Výkazu práce, Knihy jízd a Výdajů do srozumitelných celků, na základě jejich vykázaného kontextu. U jednotlivých zakázek lze nastavovat rozpočty nebo oprávnění umožňující využívat dané záznamy zakázek jen určeným osobám a skupinám. Jako výstup z této agendy slouží přehledné exporty a tiskové sestavy, které poskytují přehled odpracovaných hodin na jednotlivých zakázkách a informace o ziskovosti zakázek. [10][70][71]

6.1.6 Fakturace

Agenda Fakturace slouží pro vytváření podkladů k fakturaci, na základě vykázaných odpracovaných hodin, vykázaných jízd nebo jednotlivých výdajů. Prostřednictvím nastavených fakturačních a nákladových sazeb aplikace automaticky vypočítává celkovou částku k úhradě u vyfakturovaných záznamů. [10][70][71]

6.2 Klientské platformy

Aplikace Logeto je primárně vyvíjena jako webová aplikace, k přístupu do aplikace je možné využít většinu moderních webových prohlížečů. Mezi aktuálně podporované patří například Google Chrome, Firefox nebo Microsoft Edge. K webové aplikaci dále existuje několik klientských aplikací sloužících především k pohodlnému zadávání ověřených záznamů v terénu nebo u vstupu na pracoviště. V současné době jsou k tomuto účelu využívány platformy Android, iOS a Universal Windows Platform. Pro vývojáře třetích stran je poté dostupné REST API, které je využíváno pouze při napojení na systémy třetích stran (nejčastěji interní systémy zákazníků).

Celý systém funguje jako architektura klient-server, kde webová aplikace zastává roli serveru. Na serveru jsou centralizovaně shromažďována veškerá data a prováděny výpočty nad daty. Úlohou klientů je sběr záznamů, ověřování lokalit na základě lokalizačních modulů zařízení a synchronizace záznamů směrem k serveru.

Pro jednotlivce je aplikace Logeto dostupná bezplatně, v případě větších společností s více pracovníky je aplikace po dvouměsíčním bezplatném období

zpoplatněna dle aktuálně platného ceníku. Vyúčtování probíhá na základě kreditového systému, přičemž jsou na začátku každého kalendářního měsíce z kreditového účtu strženy kredity za každého aktivního pracovníka, který v předchozím měsíci zadal alespoň jeden záznam do systému. Kredity lze pohodlně dobíjet různými platebními metodami přímo ve webové aplikaci, kde se také nachází přehled jednotlivých vyúčtování a plateb. V následujících kapitolách jsou stručně popsány všechny platformy a jejich hlavní účel.

6.2.1 Webová aplikace

Webová aplikace slouží jako hlavní část celého systému, jsou zde uložena veškerá data z klientských aplikací. Bez jejího používání nelze využívat další části systému ani v offline režimu. Uživatelé ji mohou využít k zadávání záznamů, ale jejím hlavním smyslem je především správa či vyhodnocování zaměstnanců a jimi vykázaných pracovních hodin a jízd. Z tohoto důvodu jsou zde dostupné rozsáhlé možnosti nastavení, filtrování a exportů.

6.2.2 Mobilní aplikace

Účelem mobilních aplikací je primárně zadávání záznamů a ověřování lokality, avšak v poslední době se aplikace dále rozšiřuje a začíná se do ní implementovat také administrativní funkcionalita. Nejnovějším přírůstkem je správa uživatelů, která byla prozatím dostupná pouze ve webové aplikaci. Mobilní aplikace je vhodná pro všechny uživatele, kteří při práci disponují mobilním telefonem s operačním systémem iOS (11 a vyšší) a Android (4.0.3 a vyšší). Pro uživatele bez možnosti zadávání na mobilním telefonu je určena aplikace pro trvalá pracoviště.

6.2.3 Aplikace pro osobní počítače

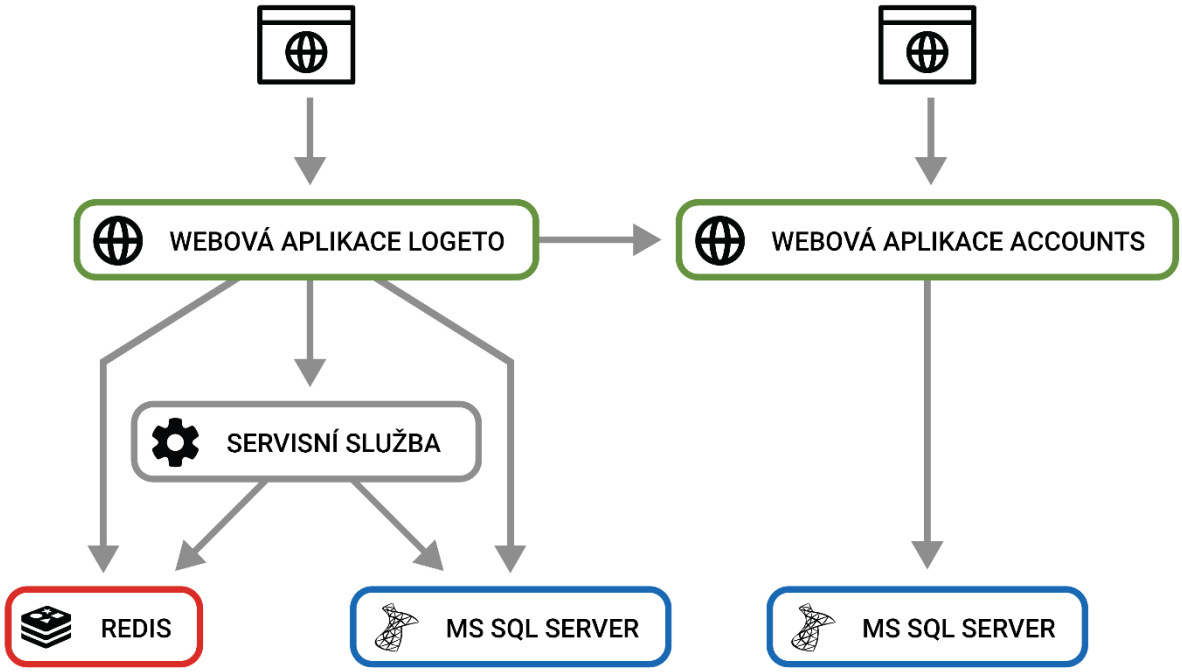
Aplikace pro osobní počítače je postavena na platformě Universal Windows Platform, je tedy vhodná pouze pro použití na osobních počítačích s operačním systémem Windows 10. Tato klientská aplikace slouží především pro kancelářské typy zaměstnání, kde mají uživatelé přístup k osobním počítačům na vlastním pracovišti. Funkcionalitou je tato aplikace totožná s aplikací pro mobilní telefony.

6.2.4 Aplikace pro trvalá pracoviště

Poslední klientská aplikace je taktéž založena na platformě Universal Windows Platform, avšak aplikace je dodávána především jako all-in-one řešení. Zákazník si tak objednává celé zařízení sloužící pro trvalé umístění na pracovišti. Funkce této aplikace jsou ještě více omezené, než je tomu u aplikace pro mobilní telefony. V aplikaci pro trvalá pracoviště je možné znamenávat příchody a odchody z pracoviště, absence a přestávky. Z důvodu zabezpečení a důvěryhodnosti vykázaných záznamů je aplikace spuštěna pouze ve výhradním režimu (Assigned access), ve kterém pracovníci nemohou zařízení používat pro jiný účel než k vykazování práce. Přístup k zařízení je pracovníkům umožněn na základě několika vyměnitelných modulů. Jedná se například o modul čtečky NFC (Near Field Communication) tagů nebo čtečky otisků prstů, sloužící pro přihlášení do aplikace.

6.3 Architektura aplikace

Architektura celé aplikace je rozdělena do několika částí. Následující diagram ilustruje architekturu aplikace a jejích částí.



Obrázek 16 *Architektura aplikace Logeto (vlastní)*

- **Webová aplikace –** Webová aplikace má monolitickou architekturu, založenou na technilogii ASP.NET Web Forms. Aplikace je nasazena na webovém serveru Internet Information Services.
- **Webová aplikace Accounts –** Aplikace Accounts je založena na stejné architektuře i technologii jako výše popsaná hlavní webová aplikace. Úkolem aplikace Account je provádět platby, sledovat platební transakce a generovat vyúčtování za dobité kredity. Dále se aplikace stará o správu uživatelských účtů. Všechny funkce aplikace Account poskytují statistiky o uživatelích, sloužící k podpoře vývoje a dalšímu plánování.
- **Servisní služba –** Komponenta servisní služby provádí jen dlouhotrvající úlohy, jako jsou aktualizační skripty databází, přepočty uživatelských záznamů, zálohy databází a udržování transakčního kontextu pro synchronizaci klientů. Tato část systému je založena na technologii Windows Communication Foundation (WCF).
- **Redis –** Redis je open-source distribuovaná databáze typu klíč-hodnota. Databáze je v tomto případě využívána jako vyrovnávací paměť (cache paměť) pro některá data uživatelů a servisní služby. K datům lze přistupovat mnohem rychleji, než kdyby byly ukládány do relačních databází. [72]
- **Microsoft SQL Server –** MSSQL Server je relační databázový systém od společnosti Microsoft. Databázový systém je využíván webovou aplikací, aplikací Accounts a servisní službou za účelem persistence dat. Aplikace account má navíc vlastní separátní MSSQL Server. Na každém z těchto serverů je umístěno přes 9 000 databází, každý uživatelský účet má založenou separátní databázi. Tato architektura databází poskytuje určité výhody, avšak také velkou řadu nevýhod. Krátkým výčtem lze zmínit například výhodu možnosti obnovení celé databáze na požádání zákazníka nebo odstínění jednotlivých zákazníků na úrovni databáze, což zabraňuje únikům dat mezi zákazníky, v případě chyby v softwaru. Naopak mezi velké nevýhody patří pomalá rychlost načítání, protože MSSQL Server není stavěn na takto velké počty relačních databází. Mezi nevýhody patří také obtížná záloha a obnovování po havárii, jelikož při této architektuře databází nelze v MSSQL Serveru využít vestavěných funkcí pro zálohování a obnovu. [73]

6.4 Vývoj, testování a distribuce

Společnost Systemart s.r.o. se snaží aplikaci vyvíjet agilním vývojem. Na celém projektu nyní participuje třináct zaměstnanců v rámci dvou poboček. Hlavní pobočkou a sídlem společnosti jsou kanceláře v Hradci Králové, na této pobočce probíhá plánování, analýza a vývoj mobilních aplikací. Sídlí zde také oddělení testerů společně se zákaznickou podporou. Druhou pobočkou jsou kanceláře v centru Brna, tato pobočka je malá, ale probíhá zde veškerý vývoj webové aplikace a aplikace pro trvalá pracoviště.

Nová verze aplikace je vydávána pravidelně alespoň jednou měsíčně, tyto pravidelné aktualizace obsahují primárně funkcionální vylepšení a zásadní vylepšení celého systému. Dále jsou během celého měsíce vydávány i menší aktualizace obstarávající především opravu chyb a drobná vylepšení. Tomuto postupu je uzpůsoben také proces vývoje, který by v ideálních případech neměl trvat déle než měsíc, během tohoto procesu je nutné vykonat plánování, analýzu, implementaci, testování a nasazení. Pro podporu celého procesu jsou využívány nástroje Jira, Teamcity a Bitbucket. [74][75][76]

7 Logování aplikace Logeto

Následující kapitoly se zabývají implementací centralizovaného logování aplikace Logeto a všech jejích komponent. Veškerá konfigurace popisovaná v této kapitole je založena na souborech ve formátu YAML, které jsou poté nasazeny do vlastního Kubernetes clusteru. Ten se již postará o spuštění aplikace dle předložených konfiguračních souborů.

Do nynější doby byly všechny komponenty aplikace Logeto odkázány pouze na logování do souborů uložených na hostitelských serverech. Tato roztroušenost znepříjemňovala dostupnost logů a u některých klientských aplikací jejich dostupnost dokonce kompletně znemožňovala. V důsledku toho nebylo možné reagovat na některé chyby v aplikacích, jiné chyby pak zůstaly dokonce neodhalené. Malá část chyb byla i částečně ignorována, jelikož neexistoval způsob centralizovaného upozornění na nastalou chybu, a tak bylo velice obtížné až nemožné odhalit některé sporadické chyby a jejich periodické výskyty.

Výjimkou v tomto původním fungování byly klientské aplikace pro mobilní telefony. Ty díky vývoji nad platformou Xamarin a napojením na Visual Studio App Center shromažďovaly data o pádech aplikací a chybách přímo do Visual Studio App Center. Zároveň byly tyto klientské aplikace schopné na žádost uživatele odeslat logy aplikace na e-mail uživatelské podpory. I tak byl tento způsob nedokonalý především z pohledu zmíněné roztroušenosti informací a nutnosti delegovat řešený problém na několik vývojářů a zákaznickou podporu s různými přístupy do konkrétních nástrojů.

Tyto skutečnosti vedly k rozhodnutí implementovat centralizované logování, tak aby všechny zainteresované osoby měly vždy dostupné veškeré informace při řešení chyb a pádů aplikací.

7.1 Nástroje a architektura logovací infrastruktury

Ještě před započetím implementace požadavku, bylo nutné vybrat vhodné prostředky pro vývoj a navrhnout infrastrukturu pro logování. Na základě

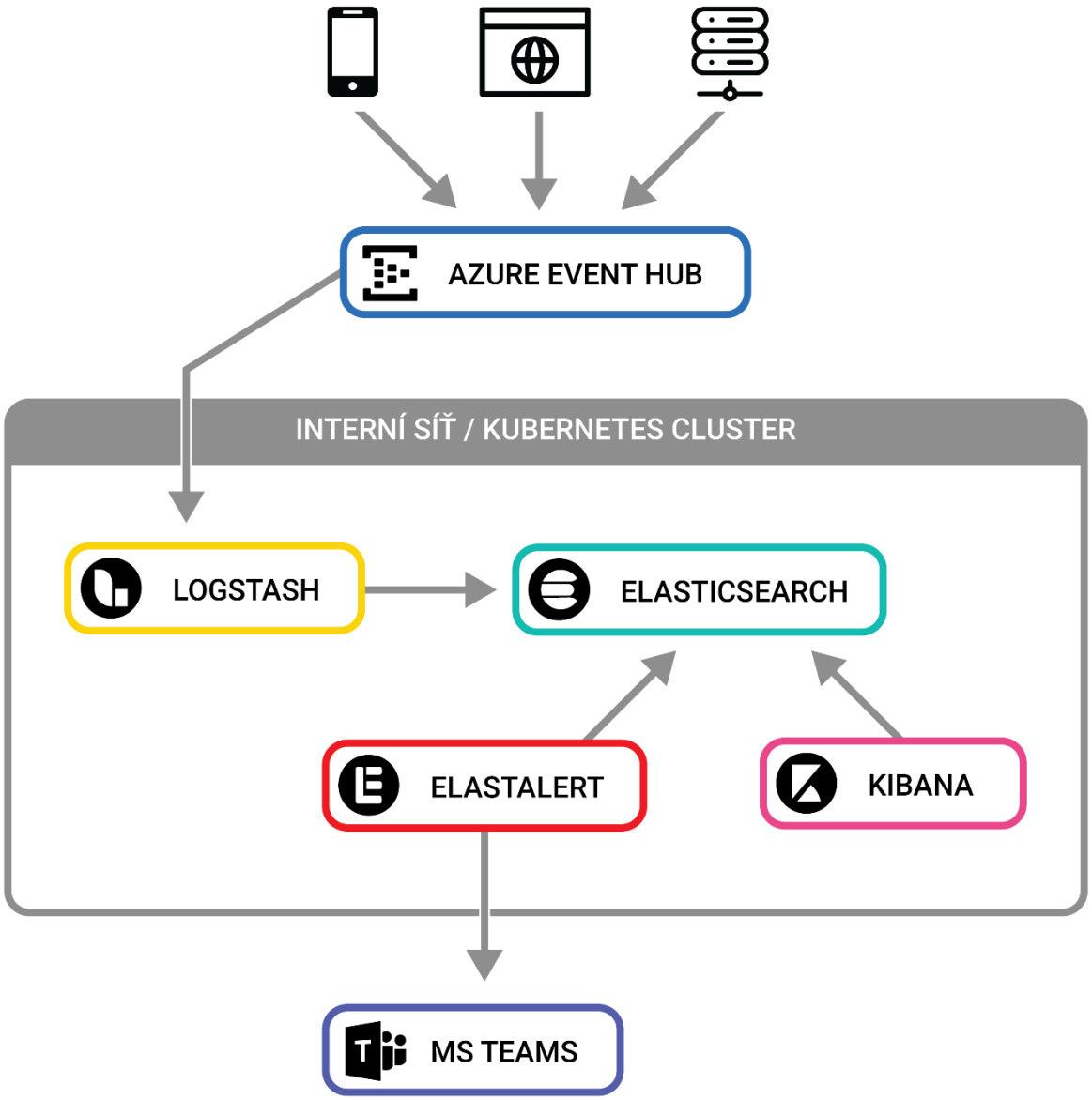
prostudovaných vlastností, jejichž popisem se zabývá teoretická část této diplomové práce, byl k tomuto účelu zvolen ELK Stack od společnosti Elastic N.V. Mezi podporované funkce ELK Stacku patří mimo jiné požadované zasílání varovných zpráv do komunikačních nástrojů, avšak dle nabízených edicí se jedná o placenou funkcionalitu. Po krátké komunikaci s obchodním oddělením Elastic N.V. a obdržení konkrétní cenové nabídky na licenci obsahující zasílání varovných zpráv, bylo rozhodnuto o využití pouze open-source licence ELK Stack v kombinaci s nástrojem třetích stran, který se o zasílání varovných zpráv postará namísto původně zamýšlené placené verze. Příčinou tohoto rozhodnutí byla příliš vysoká cena licence, odpovídající několika násobkům přiděleného měsíčního rozpočtu.

K účelu zasílání varovných zpráv byla zvolena platforma ElastAlert. Důvodem výběru byla velikost komunity, která se kolem ní shlukuje a pravidelné aktualizace. Příjemným bonusem u tohoto nástroje je také existence kontejnerizované varianty. Použití ostatních nástrojů, které by byly napojitelné na ElasticSearch a současně schopné zasílat potřebné varovné notifikace, je opět dostupné pouze pod placenou licencí, nebo nejsou dlouhodobě udržovány, v důsledku čehož je nelze na aktuální verzi ElasticSearch napojit.

Jako poslední část logovací infrastruktury bylo nutné navrhnout způsob sběrů logů z několika stovek až tisíců klientských zařízení fungujících mimo interní síť a infrastrukturu, bez nutnosti vystavení některého z nástrojů ELK Stacku pro vnější síť. Společnost Systemart s.r.o. se obávala možného bezpečnostního rizika při vystavení nástrojů ELK Stacku pro veřejnou síť. Bylo tak rozhodnuto, že veškerá logovaná data budou zasílána do určitého typu buffer queue (vyrovnávací fronty) ležící ve vnější síti a na tuto buffer queue bude dále napojen Logstash ležící ve vnitřní síti. Jeho úkolem bude přijímat data ze zmíněné fronty. Kvůli dodržování zásad firemní kultury, která prosazuje využívání produktů od společnosti Microsoft a zkušeností s cloudovou službou Azure, byl pro účely buffer queue, zvolen nástroj Azure Event Hub.

Výsledná logovací infrastruktura se tak skládá z nástroje Azure Event Hub, do kterého všechny aplikace zasílají svá logovaná data. Z tohoto místa data putují do nástroje Logstash, který se již nachází ve vnitřní síti. Následně Logstash data upraví do požadované

podoby a zaindexuje je do nástroje ElasticSearch. Data uložená v ElasticSearch je pak možné manuálně prohlížet a analyzovat v grafickém prostředí Kibana. Pro zasílání varovných zpráv a automatické upozornění na určitý typ logů je zde nástroj ElastAlert komunikující přímo s ElasticSearch. Celá architektura je pak znázorněna na **Obrázek 17**.

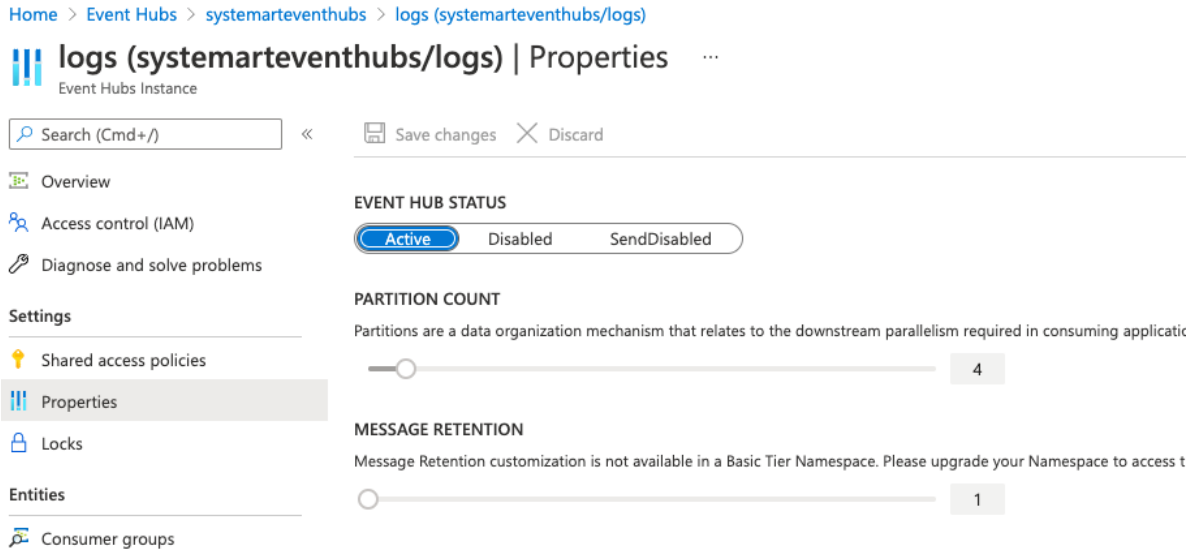


Obrázek 17 *Diagram logovací infrastruktury (vlastní)*

8 Azure Event Hub

Azure Event Hub je streamovací platforma pro big data, pomáhá se sběrem a zpracováním miliónů událostí za sekundu. Platforma funguje jako Platform-as-a-Service (PaaS) nástroj v rámci cloudové služby Azure. Možnosti využití této platformy jsou velice rozsáhlé, v tomto konkrétním případě však slouží pouze jako sběrný bod logovaných dat. Jelikož je platforma uzpůsobená zpracování velkého množství dat a umí data dočasně uchovat, je její využití velice výhodné. Pokud by nástroj Logstash krátkodobě nemohl data obsloužit, zůstanou tato data zachována v Azure Event Hub. Zároveň nebude problém zapisovat data do Azure Event Hub z celého světa a několika stovek až tisíců zařízení současně.

Konfigurace Azure Event Hub, spočívá pouze ve vytvoření požadovaného počtů instancí a jejich nastavení v rámci jednoduchého grafického prostředí cloudové služby Azure (viz **Obrázek 18**). Zároveň jsou tímto způsobem vytvořeny přístupové údaje složené z přístupového klíče a klíče účtu. Konfigurace těchto klíčů dále popisována nebude a ve všech konfiguracích budou tyto klíče cenzurovány z důvodu zachování bezpečnosti a non-disclosure agreement (NDA).



Obrázek 18 *Nastavení Azure Event Hub (vlastní)*

V kontextu této diplomové práce, se jedná o konfiguraci jedné instance Azure Event Hub obsahující čtyři oddíly. Oddíl je jednotkou Azure Event Hub, jedná se o frontu udržující všechny zprávy v pořadí, jak byly obdrženy. Na jednu instanci Azure Event Hubu, lze napojit jen tolik odběratelských služeb, kolik obsahuje oddílů. Azure Event Hub se zároveň sám stará o rovnoměrné rozdělení zpráv mezi všechny oddíly dané instance. Je tedy zachováno časové pořadí zpráv v rámci jednoho oddílu, avšak nemusí být dodrženo, v rámci celé instance Azure Event Hubu. Z tohoto důvodu jsou všechna logovaná data vybavena také časovým razítkem a unikátním identifikátorem založeném na časovém razítku. Čtyři oddíly jsou pro typický objem dat aplikace dostačující a především odpovídají konfiguraci input pluginu nástroje Logstash a výkonnostním prostředkům, kterými Logstash disponuje. [77]

9 Popis konfigurace nástrojů ELK Stack

V následujících kapitolách jsou popsány konkrétní postupy a konfigurace využité při implementaci nástrojů ELK Stacku v rámci projektu Logeto. Všechny zde popsané principy odpovídají skutečnému použití, avšak některé konfigurační soubory jsou částečně pozměněny kvůli zachování NDA a pro účely koncepčního zasazení do této diplomové práce. I přesto zůstává zachován význam těchto konfigurací, tak aby bylo možné pochopit daný princip a replikovat jej na vlastním řešení.

9.1 Logstash

Logstash slouží v rámci popisované logovací infrastruktury k odběru dat z Azure Event Hub, jejich úpravě do požadované podoby a následnému indexování do ElasticSearch.

9.1.1 Nasazení nástroje Logstash

Nasazení tohoto nástroje je postaveno na základě následujícího konfiguračního souboru (viz **Zdrojový kód 3**). Na tomto konfiguračním souboru lze vidět základní informace důležité pro Kubernetes, jako je název, značky, namespace nebo konkrétní docker image. V tomto případě se jedná o aktuální konfiguraci využívající docker image **logstash: 7.10.0** přímo z registru společnosti Elastic N.V.

Dále lze v tomto konfiguračním souboru nalézt také proměnné prostředí, která jsou dále propsána do běžícího kontejneru. Logstash je také využívá pro připojení k ElasticSearch. Jedná se o **ELASTICSEARCH_HOST** obsahující název Kubernetes servisy vytvořené pro ElasticSearch, **ELASTICSEARCH_PORT** s číslem portu na kterém ElasticSearch naslouchá a nakonec **LOGSTASH_USER** a **LOGSTASH_PWD** obsahující přihlašovací údaje do ElasticSearch. Heslo je do těchto údajů dosazováno pomocí Kubernetes, a není tak nutné jej uvádět přímo do konfiguračního souboru.

Poslední částí konfiguračního souboru pro nasazení jsou mapované adresáře. Tato část slouží k namapování konfiguračních souborů uvedených v Kubernetes prostředku ConfigMap do vnitřní souborové struktury běžícího kontejneru. V tomto případě se jedná

o konfiguraci samotného nástroje Logstash, konfiguraci pluggable pipeline architektury a následně složky obsahující konfigurační soubory jednotlivých pipeline. Všem těmto konfiguračním souborům jsou věnovány následující odstavce.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: logstash
  namespace: elasticsearch
spec:
  template:
    spec:
      nodeSelector:
        kubernetes.io/os: linux
      containers:
      - image: docker.elastic.co/logstash/logstash:7.10.0
        name: logstash
        ports:
          - containerPort: 9600
            name: logstash
        env:
        - name: ELASTICSEARCH_HOST
          value: "logeto-es-http"
        - name: ELASTICSEARCH_PORT
          value: "9200"
        - name: LOGSTASH_USER
          value: "elastic"
        - name: LOGSTASH_PWD
          valueFrom:
            secretKeyRef:
              name: logeto-es-elastic-user
              key: elastic
        volumeMounts:
        - name: config
          mountPath: /usr/share/logstash/config/pipelines.yml
          subPath: pipelines.yml
          readOnly: true
        - name: config
          mountPath: /usr/share/logstash/config/logstash.yml
          subPath: logstash.yml
          readOnly: true
        - name: pipelines
          mountPath: /usr/share/logstash/pipeline
          readOnly: true
      volumes:
      - name: pipelines
        configMap:
          name: logstash-pipelines
      - name: config
        configMap:
          name: logstash
```

Zdrojový kód 3 *Deployment.yml pro nasazení nástroje Logstash (vlastní)*

9.1.2 Konfigurace nástroje Logstash

Konfigurace nástroje je řízena třemi následujícími konfiguračními soubory ve formátu YAML souboru pro Kibernetes prostředek ConfigMap.

První ConfigMap obsahuje konfigurační soubor **logstash.yml** (viz **Zdrojový kód 4**) sloužící pro nastavení hlavních vlastností nástroje. Zde se jedná pouze o konfiguraci automatického obnovování konfigurace v případě změny konfiguračních souborů, vypnutí self-monitoringu (monitoring je následně řešen pomocí nástroje Beat) a vypnutí podpory Elastic Common Schema (ECS).

```
http.port: 9600
http.host: 0.0.0.0
config.reload:
  automatic: true
  interval: 60
monitoring.enabled: false
monitoring.cluster_uuid: gKWkMgkfQb6YlKbxGxhd2A
pipeline.ecs_compatibility: disabled
```

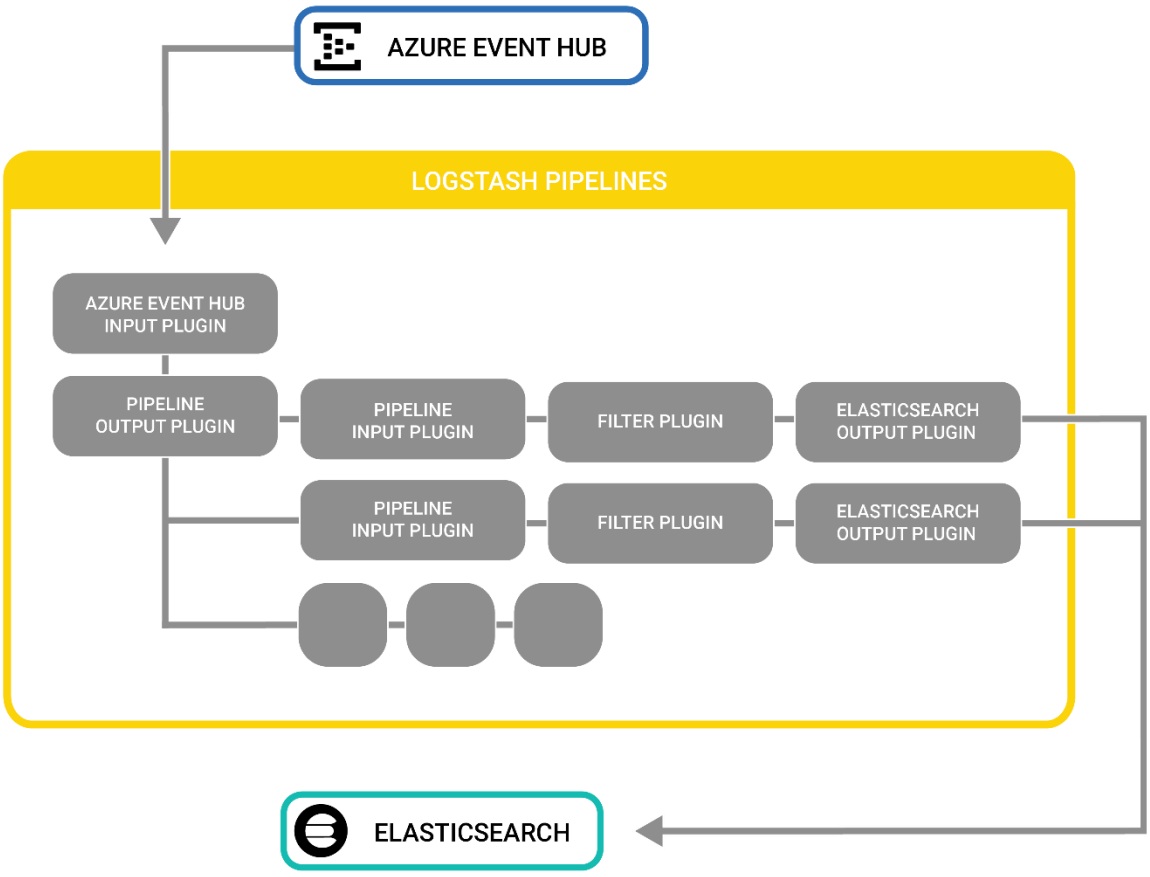
Zdrojový kód 4 *Logstash.yml pro konfiguraci nástroje Logstash (vlastní)*

9.1.3 Konfigurace Pipelines

Následuje ConfigMap s konfiguračním souborem **pipelines.yml** pro pluggable pipeline architekturu (viz **Zdrojový kód 5**). Jedná se o výčet identifikátorů jednotlivých pipeline a cesty k jejich konfiguračním souborům. V tomto případě byla vybrána architektura zohledňující typ a zdroj vstupních dat. Pro každou část aplikace Logeto je tak vytvořena pipeline, která obstará úpravu dat a následně je zašle do ElasticSearch k indexaci. Zároveň je zde jedna hlavní pipeline, která se stará o příjem dat z Azure Event Hub a rozřazení dat do správné pipeliny ke zpracování. Rozřazení probíhá na základě atributu **Source** obsaženém v každé zprávě. Vizuální znázornění této architektury je naznačeno na **Obrázek 19**.

```
- pipeline.id: event-hub-input
  path.config: "/usr/share/logstash/pipeline/event-hub-input.conf"
- pipeline.id: uwp
  path.config: "/usr/share/logstash/pipeline/uwp.conf"
- pipeline.id: terminal
  path.config: "/usr/share/logstash/pipeline/terminal.conf"
- pipeline.id: appservice
  path.config: "/usr/share/logstash/pipeline/appservice.conf"
- pipeline.id: web
  path.config: "/usr/share/logstash/pipeline/web.conf"
- pipeline.id: webrequests
  path.config: "/usr/share/logstash/pipeline/webrequests.conf"
- pipeline.id: psmodulepublic
  path.config: "/usr/share/logstash/pipeline/psmodulepublic.conf"
- pipeline.id: psmoduleinternal
  path.config: "/usr/share/logstash/pipeline/psmoduleinternal.conf"
- pipeline.id: feedback
  path.config: "/usr/share/logstash/pipeline/feedback.conf"
...
..
.
```

Zdrojový kód 5 *Pipelines.yml pro konfiguraci nástroje Logstash (vlastní)*



Obrázek 19 *Architektura pipeline uvnitř nástroje Logstash (vlastní)*

9.1.4 Konfigurace jednotlivých Pipeline

Poslední ConfigMap obsahuje konfigurační soubory **event-hub-input.conf**, **uwp.conf** a **webrequests.conf** pro jednotlivé pipeliny uvnitř nástroje Logstash. Z důvodu rozsáhlosti tohoto konfiguračního souboru je zde zobrazena jen ukázková konfigurace obsahující hlavní pipeline pro vstup dat, pipeline pro Universal Windows Platform aplikaci a pipeline zpracovávající dotazy na webovou aplikaci.

Event-hub-input.conf

Konfigurační soubor pro vstup dat do nástroje Logstash má název **event-hub-input.conf** (viz **Zdrojový kód 6**) a sestává ze tří částí. První část je vstupní a obsahuje vstupní plugin **azure_event_hubs**. Konfigurace tohoto vstupního pluginu sestává z uvedení připojovacího řetězce, uživatelského jména a přístupového klíče. Dále je zde nastavena doba, po jejímž uplynutí má být v Azure Event Hubu vytvořen checkpoint, což slouží pro případ znovu navázání komunikace při výpadu jedné ze stran nebo restartu nástroje Logstash. A nakonec je zde uveden počet vláken, které má Logstash vytvořit. Je doporučené, aby počet vláken odpovídal počtu oddílů v Azure Event Hub plus jedna a zároveň by tento počet měl být stejný jako počet jader hostitelského počítače opět plus jedna. Z toho důvodu je počet vláken nastaven na pět. Nastavení vláken je důležitým parametrem, jelikož jeden oddíl v Azure Event Hub může mít pouze jednoho konzumenta v daném čase, avšak pokud vlákno přejde ze čtení na zpracování v části filtrování, je tento čtecí kanál uvolněn a může jej nahradit volné vlákno, v tomto případě páté dodatečné. Je tak docíleno optimálního využití propustnosti Azure Event Hubu i nástroje Logstash a omezení nečinných vláken na minimum.

Druhá část je tvořena filtrem využívajícím filtrační plugin **json**. Jedná se o jednoduchou část, která parsuje obdržená data ve formátu JSON do formátu jednotlivých datových polí Logstash zprávy. Takováto zpráva je následně postoupena do poslední části.

Poslední část je výstupní a je v ní využito jednoduché rozhodovací logiky společně s výstupním pluginem **pipeline**. Jednoduché podmínky zde rozhodují, která zpráva obsahuje řetězec znaků určující o jakou zdrojovou aplikaci nebo část aplikace Logeto se

jedná. Bohužel Logstash nabízí jen základní řídící struktury a lze zde vytvořit pouze strukturu If-Else, při větším větvení tak již dochází k nepřehlednosti této struktury. Z tohoto důvodu je také uvedena ukázka jen pro dva zdroje.

Dále je zde využit výstupní plugin **pipeline** u kterého stačí uvést parametr **send_to** a výčet identifikátorů jednotlivých pipeline, které mají sloužit jako další vstup.

```
input {
  azure_event_hubs {
    event_hub_connections => [
      "Endpoint=sb://*****.servicebus.windows.net/;
      SharedAccessKeyName=*****;
      EntityPath=logs"
    ]
    threads => 5
    decorate_events => true
    checkpoint_interval => 60
    storage_connection =>
      "DefaultEndpointsProtocol=https;
      AccountName=*****;
      AccountKey=*****;
      EndpointSuffix=core.windows.net"
  }
}
filter {
  json {
    source => "message"
  }
}
output {
  if [Source] == "WEB" {
    pipeline { send_to => uwp }
  }
  else if [Source] == "WEB_REQ" {
    pipeline { send_to => webrequests }
  }
  else {
    stdout {}
  }
}
```

Zdrojový kód 6 *Vstupní pipeline z Azure Event Hub (vlastní)*

Uwp.conf

Konfigurační soubor s názvem **uwp.conf** (viz **Zdrojový kód 7**) pro zdroj z Universal Windows Platform aplikace, obsahuje pouze vstupní a výstupní část. Vstupní část obsahuje pouze vstupní plugin **pipeline,** u kterého je uveden parametr **address**

a identifikátor, pod kterým je tato pipeline rozpoznatelná. Výstupní část se skládá z výstupního pluginu **elasticsearch**, který obsahuje adresu a port, na němž je dostupný ElasticSearch server a přihlašovací údaje k němu. Dále je zde vypnuta podpora Elastic Common Schema (ECS) a nastaven název indexu, do kterého mají být data indexována. O ostatní obsluhu indexace se Logstash a ElasticSearch postarají automaticky.

```
input {
  pipeline { address => uwp }
}
output {
  elasticsearch {
    hosts => ["http://${ELASTICSEARCH_HOST}:${ELASTICSEARCH_PORT}"]
    index => "uwp-"
    document_id => "%{UniqueId}"
    user => "${LOGSTASH_USER}"
    password => "${LOGSTASH_PWD}"
    ecs_compatibility => disabled
  }
}
```

Zdrojový kód 7 *Uwp.conf konfigurace pipeline pro UWP klienta (vlastní)*

Webrequests.conf

Posledním popisovaným konfiguračním souborem pro nástroj Logstash je **webrequests.conf** (viz **Zdrojový kód 8**)určený pro dotazy na webovou aplikaci. Tato konfigurace opět obsahuje tři části: vstupní, filtrační a výstupní. Vstupní a výstupní část jsou analogicky shodné s konfigurací **uwp.conf**. Filtrační část zde využívá filtrační plugin **geoip**, tento plugin umožňuje z IP adresy získat geolokační informaci za pomoci geolokační databáze GeoLite2. K účelům této diplomové práce je zde vyžito atributu **IpAddress** obsahující IP adresu. Plugin se následně automaticky postará o získání informací a přidá je mezi ostatní datová pole Logstash zprávy.

```
input {
  pipeline { address => webrequests }
}
filter {
  geoip {
    source => "IpAddress"
  }
}
output {
  elasticsearch {
    hosts => ["http://${ELASTICSEARCH_HOST}:${ELASTICSEARCH_PORT}"]
    index => "webrequests-"
    document_id => "%{UniqueId}"
    user => "${LOGSTASH_USER}"
    password => "${LOGSTASH_PWD}"
    ecs_compatibility => disabled
  }
}
```

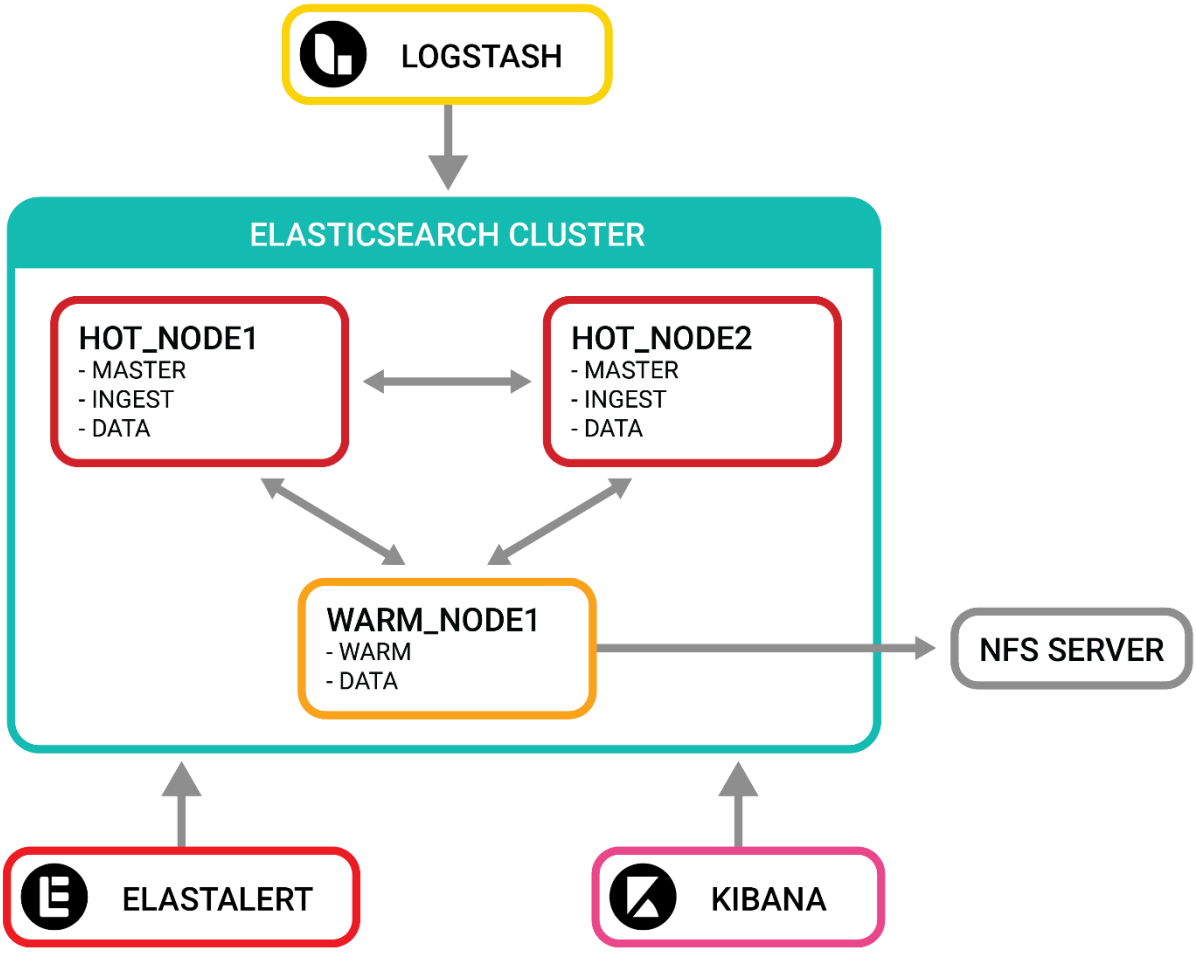
Zdrojový kód 8 *Webrequests.conf konfigurace pro dotazy webové aplikace (vlastní)*

9.2 ElasticSearch

Dalším a současně tím nejdůležitějším nástrojem celé logovací infrastruktury je ElasticSearch, přesto je jeho nastavení pomocí konfiguračních souborů jedno z nejjednodušších. Příčinou toho je potřeba převážnou část nastavení konfigurovat až skrze ElasticSearch API nebo grafické rozhraní Kibana. Z tohoto důvodu jsou konfigurace funkcionalit, jako je správa uživatelů nebo ILM, představeny až v kapitole popisující grafické rozhraní Kibana.

Na uvedeném příkladu není využíváno pouze jedné instance ElasticSearch, naopak se jedná o menší ElasticSearch Cluster sestávající se ze tří instancí konfigurovaných pro spolupráci v rámci hot-warm architektury. První dvě instance jsou anotovány jako **hot** nody s přiřazenými rolemi **master**, **data** a **ingest**. Pro tento typ instancí je v Kubernetes clusteru přiřazeno rychlé polovodičové uložiště o velikosti 100 GB na jednu instanci a data jsou na něm replikována. Naopak třetí instance je anotována jako **warm** node pouze s rolí **data**. K této instanci je připojeno NFS uložiště osazené běžným pevným diskem o velikosti 150 GB. **Warm** instance slouží pro data starší než určený počet dní, neboť objem obdržených dat a indexy na ní uložené obsahují pouze primární repliku. Vyhledávání v těchto indexech je tak výrazně pomalejší než u indexů uložených na **hot** nodu a zároveň

není zajištěna jejich dostupnost a obnovení při chybě nebo výpadku tohoto **warm** nodu. Vnitřní strukturu ElasticSearch Clusteru je možné vidět na následujícím obrázku.



Obrázek 20 *Vnitřní architektura ElasticSearch clusteru (vlastní)*

9.2.1 Persistentní oddíly

Výše popsaná konfigurace je řízena pomocí dvou konfiguračních souborů. Prvním z nich je **persistentvolume.yml** (viz **Zdrojový kód 9** a **Zdrojový kód 10**), tento konfigurační soubor obsahuje tři Kubernetes zdroje PersistentVolume, mapující persistentní oddíly pro Kubernetese.

Všechny tři PersistentVolume si jsou velice podobné, jediným významným rozdílem a také významnou konfigurací je zde část **nfs** u oddílu pro **warm** node, a část **nodeAffinity** obsažená ve všech třech konfiguracích oddílů.

U části **nfs** je možné vidět adresu v interní síti, na které běží NFS server a cestu ke které je tento oddíl mapován. Část **nodeAffinity** pak určuje, ke kterému hostitelskému zařízení se oddíl připojí. V případě této implementace Kubernetes cluster běží nad dvěma nody **k8s-w01-lin** a **k8s-w02-lin**, jedná se o servery s operačním systémem linuxového typu. Konfigurace tedy zaručuje, že rychlá uložiště jsou vždy rovnoměrně mapována dle názvu hostitelského zařízení a pomalé NFS uložiště je mapováno vždy k nodu s názvem **k8s-w02-lin**.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-data-nfs-1
spec:
  storageClassName: elasticsearch-nfs
  capacity:
    storage: 150Gi
  accessModes:
  - ReadWriteOnce
  mountOptions:
    - vers=3
  nfs:
    path: /volume1/Elastic/es-warm-1
    server: 192.168.2.2
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - k8s-w02-lin
```

Zdrojový kód 9 *PersistentVolume pro warm node (vlastní)*

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-data-0
spec:
  storageClassName: elasticsearch
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/elastic/es0"
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - k8s-w01-lin
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: elasticsearch-data-1
spec:
  storageClassName: elasticsearch
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/elastic/es0"
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: In
          values:
          - k8s-w02-lin
```

Zdrojový kód 10 *PersistentVolume pro hot nody (vlastní)*

9.2.2 Nasazení a konfigurace nástroje ElasticSearch

Druhým konfiguračním souborem je **elasticsearch.yml** (viz **Zdrojový kód 13** a **Zdrojový kód 14**). ElasticSearch nelze nasadit v prostředí Kubernetes pomocí docker image, tak jak tomu bylo v případě nástroje Logstash. K nasazení je nejdříve nutné nainstalovat Elastic Cloud on Kubernetes (ECK). Jedná se edici ElasticSearch složenou z několika aplikací, služeb a prostředků určených pro prostředí Kubernetes. Instalace ECK

je možná buď pomocí Helm, nebo aplikováním konfiguračního souboru dodávaného společností Elastic N.V.

Helm je správcem balíčků v prostředí Kubernetes a umožňuje spravovat, instalovat a aktualizovat komplexnější aplikace složené z více částí. Samotná instalace poté sestává z následujících tří příkazů.

```
helm repo add elastic https://helm.elastic.co
helm repo update
helm install elastic-operator elastic/eck-operator
```

Zdrojový kód 11 *Instalace ECK pomocí správce balíčků Helm (vlastní)*

Výhodou tohoto přístupu je možnost většího přizpůsobení instalace pomocí několika parametrů. Seznam všech instalačních parametrů je pak možné získat následujícím příkazem.

```
helm show values elastic/eck-operator
```

Zdrojový kód 12 *Příkaz pro zobrazení konfigurovatelných parametrů (vlastní)*

Instalace pomocí konfiguračního souboru je jednodušší na počet kroků, avšak postrádá možnost přizpůsobení. Samotný konfigurační soubor obsahuje několik separátních konfigurací a skládá se téměř ze 4 000 řádků konfigurace. Celý konfigurační soubor je možné nalézt na oficiálních webových stránkách Elastic. Tento soubor poté stačí aplikovat jako konfiguraci prostředí Kubernetes a to se postará o nasazení všech jeho částí.

Jakmile je Elastic Cloud on Kubernetes nainstalován v prostředí Kubernetes, je možné přistoupit k jeho konfiguraci a spuštění. Spuštění i konfigurace probíhá pouhým aplikováním konfiguračního souboru **elasticsearch.yml** (viz **Zdrojový kód 13** a **Zdrojový kód 14**) obsahujícího konfiguraci ElasticSearch Clusteru. Konfigurační soubor se skládá ze dvou částí. První část obsahuje konfiguraci hlavních nodů anotovaných jako **hot** s rolí **master**, **data** a **ingest**. Druhá část pak obsahuje konfiguraci nodu anotovaného jako **warm** pro uložení starších dat.

Za zmínku stojí i další části konfigurace, například skutečnost, že počet hlavních nodů je v tomto případě nastaven na dva a warm node existuje pouze v jedné instanci. Dále je jistě důležitá část **config**, která obstarává přiřazení anotací a rolí či zabezpečení. Vnitřní zabezpečení je nastaveno na **native**, což je jediná varianta autentizace uživatelů, kterou ElasticSearch nabízí v neplacené edici. Autentizace je následně řízena pomocí interní správy uživatelů, kterou si ElasticSearch uchovává ve svých systémových indexech. Její nastavení je detailněji představeno a okomentováno v kapitole zabývající se popisem grafického rozhraní Kibana.

```yaml
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: logeto
  namespace: elasticsearch
spec:
  version: 7.10.0
  http:
    tls:
      selfSignedCertificate:
        disabled: true
    service:
      spec:
        selector:
          common.k8s.elastic.co/type: elasticsearch
          elasticsearch.k8s.elastic.co/cluster-name: logeto
          elasticsearch.k8s.elastic.co/statefulset-name: logeto-es-master
  secureSettings:
  - secretName: snapshot-secret
  nodeSets:
  - name: master
    count: 2
    podTemplate:
      spec:
        containers:
        - name: elasticsearch
        initContainers:
        - name: sysctl
          securityContext:
            privileged: true
          command: ['sh', '-c', 'sysctl -w vm.max_map_count=262144']
    config:
      node.attr.data: hot
      node.roles: ["master", "data", "ingest"]
      xpack.security.authc.realms:
        native:
          native1:
            order: 0
    volumeClaimTemplates:
    - metadata:
        name: elasticsearch-data
      spec:
        storageClassName: elasticsearch
        accessModes:
        - ReadWriteOnce
        resources:
          requests:
            storage: 10Gi
```

Zdrojový kód 13 *Elasticsearch.yml část I. (vlastní)*

```
- name: data-warm
  count: 1
  podTemplate:
    spec:
      initContainers:
      - name: sysctl
        securityContext:
          privileged: true
        command: ['sh', '-c', 'sysctl -w vm.max_map_count=262144']
  config:
    node.attr.data: warm
    node.roles: ["data"]
    xpack.security.authc.realms:
      native:
        native1:
          order: 0
  volumeClaimTemplates:
  - metadata:
      name: elasticsearch-data
    spec:
      storageClassName: elasticsearch-nfs
      accessModes:
      - ReadWriteOnce
      resources:
        requests:
          storage: 10Gi
```

Zdrojový kód 14 *Elasticsearch.yml část II. (vlastní)*

9.3 Kibana

Dalším nástrojem, jehož nasazení a konfigurace je nedílnou součástí této diplomové práce, je grafické prostředí Kibana. Tato kapitola se zabývá jednak principem jednoduchého nasazení či konfigurace správy uživatelů apod. a jednak popisem samotného grafického rozhraní.

9.3.1 Nasazení a konfigurace prostředí Kibana

Díky tomu, že je součástí Elastic Cloud on Kubernetes i předinstalované grafické rozhraní, není potřeba UI Kibana nasazovat pomocí docker image. Postačí jen krátký základní konfigurační soubor **kibana.yml** (viz **Zdrojový kód 15**). Tento konfigurační soubor nenabízí žádné konkrétní nastavení, které bych stálo za zmínku. Jedná se o doporučenou konfiguraci, kterou je možné nalézt v oficiální dokumentaci. Jedinou nezvyklostí je zde nastavení dvou instancí z důvodu redundance a vypnutí self monitoringu, který je řešen pomocí nástroje Beat.

```
apiVersion: kibana.k8s.elastic.co/v1
kind: Kibana
metadata:
  name: logeto
  namespace: elasticsearch
spec:
  version: 7.10.0
  count: 2
  elasticsearchRef:
    name: logeto
  config:
    monitoring.kibana.collection.enabled: false
  http:
    tls:
      selfSignedCertificate:
        disabled: true
  podTemplate:
    spec:
      nodeSelector:
        kubernetes.io/os: linux
  secureSettings:
  - secretName: kibana-secret-settings
```

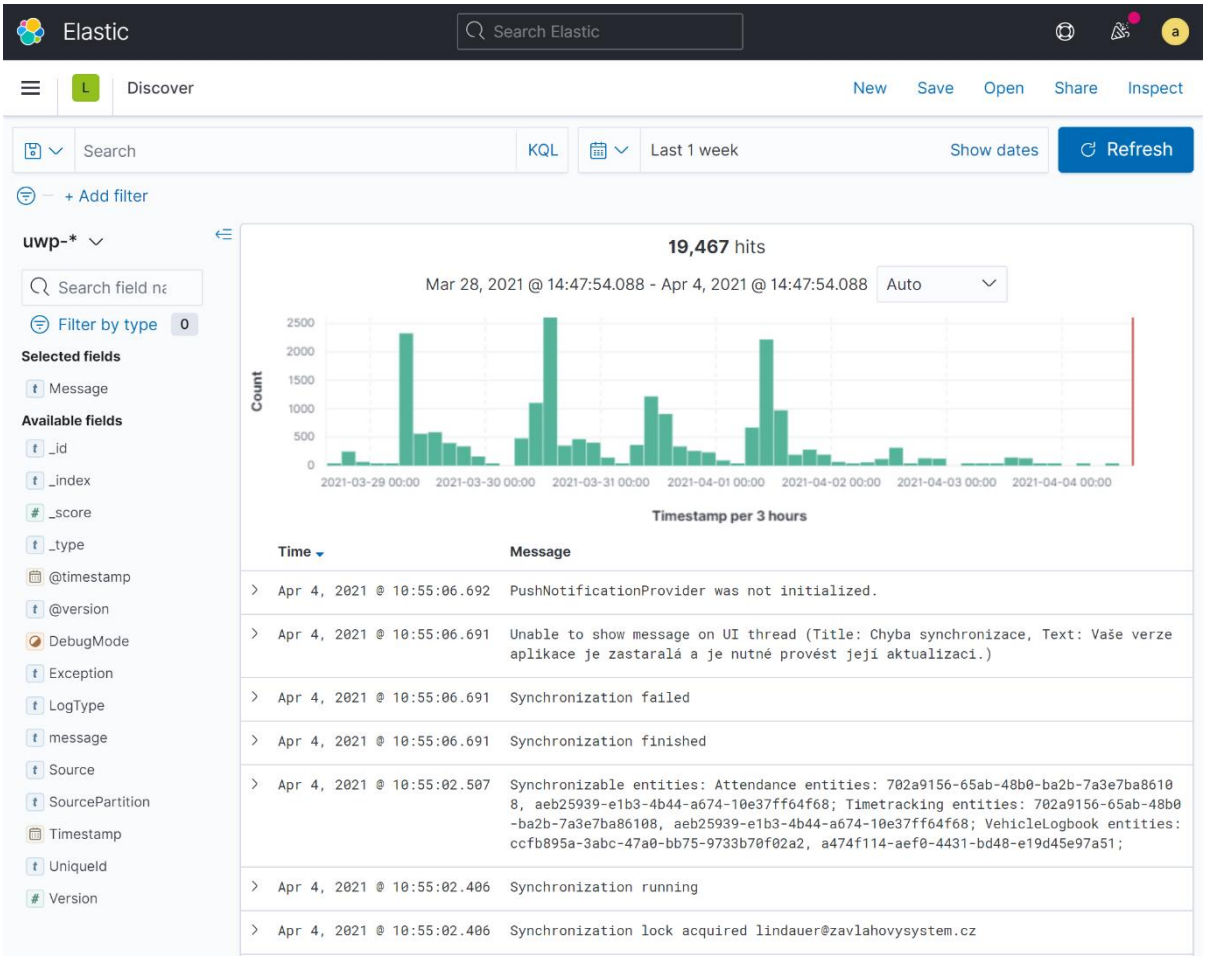
Zdrojový kód 15 *Kibana.yml pro nasazení grafického prostředí Kibana (vlastní)*

9.3.2 Kibana Discovery

Sekce Kibana Discovery je v případě společnosti Systemart s.r.o. využívána především vývojáři, ti k ní mají volný přístup a mohou zde prohlížet surová indexovaná data skrze všechny indexy. Zároveň zde mají volný přístup k filtrování a agregaci dat, tak aby mohli dle svých potřeb analyzovat vzniklé problémy a chyby v aplikaci.

Filtrování a agregace zde probíhá pomocí horního panelu, ve kterém se nachází vstupní pole pro KQL dotazy a pole pro nastavení momentálně zobrazeného časového rámce. Pravá část sekce je věnována filtrování polí, která jsou pro daná data indexována. V analogii s relačními databázemi se jedná o filtrování sloupců tabulky. Ve středu se pak nachází graf pro zvolený časový rámec zobrazující histogram výskytů a pod ním již seřazený seznam indexovaných dat. Každý řádek zde reprezentuje jeden indexovaný log. Tyto řádky lze dále rozbalit pro podrobnější náhled. V případě, že se jedná o pole

s ordinálním datovým typem, lze řazení seznamu měnit na základě momentálně zobrazených polí. Vše výše popisované lze vidět na následujícím obrázku.



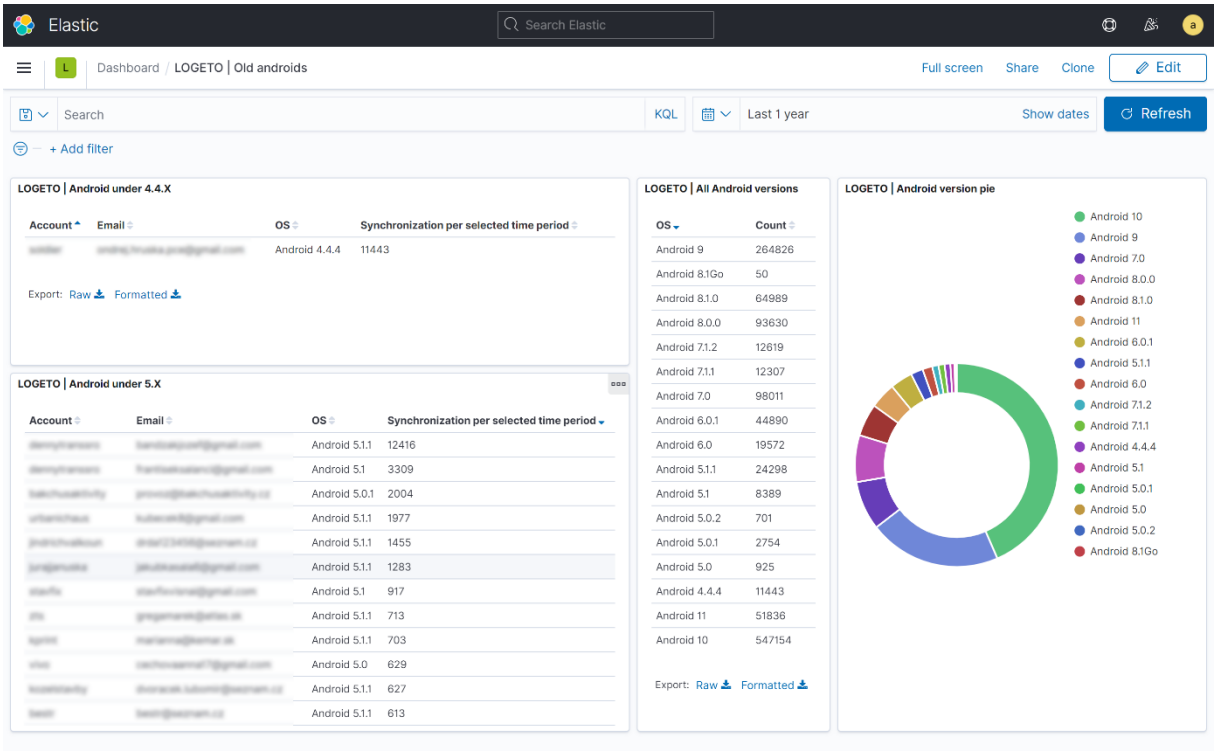
Obrázek 21 *Kibana Discovery pro index UWP (vlastní)*

9.3.3 Využívané vizualizace

Vizualizaci využívá především uživatelská podpora a analytici jakožto podporu při rozhodování. Mezi nejpoužívanější vizualizace v kontextu projektu Logeto patří přehled mobilních operačních systémů, které uživatelé využívají a počty dotazů na webovou aplikaci.

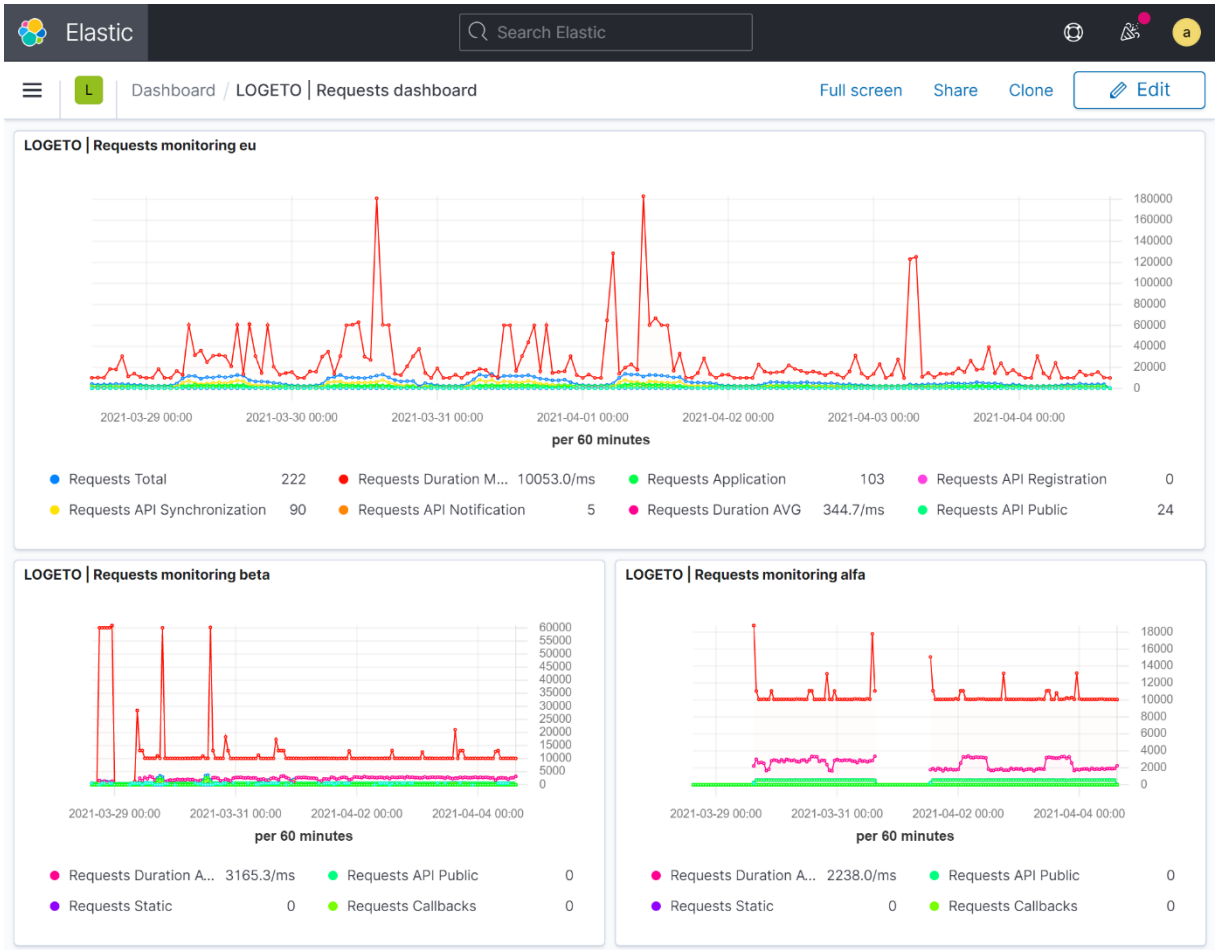
První zmíněná vizualizace pomáhá stanovit verzi mobilních operačních systémů, na kterou cílit finální aplikace. Díky tomuto ukazateli lze například odstavit starší verze operačního systému Android a vyhnout se tak potížím se zpětnou kompatibilitou. Levá část této vizualizace je věnována konkrétním uživatelům s operačním systémem Android

5 a 4. Pravá část je pak souhrnně zobrazuje počty zalogovaných dotazů za jednotlivé verze operačního systému Android. Tuto vizualizaci je možné vidět na **Obrázek 22**.



Obrázek 22 *Vizualizace pro analýzu používaných verzi OS Android (vlastní)*

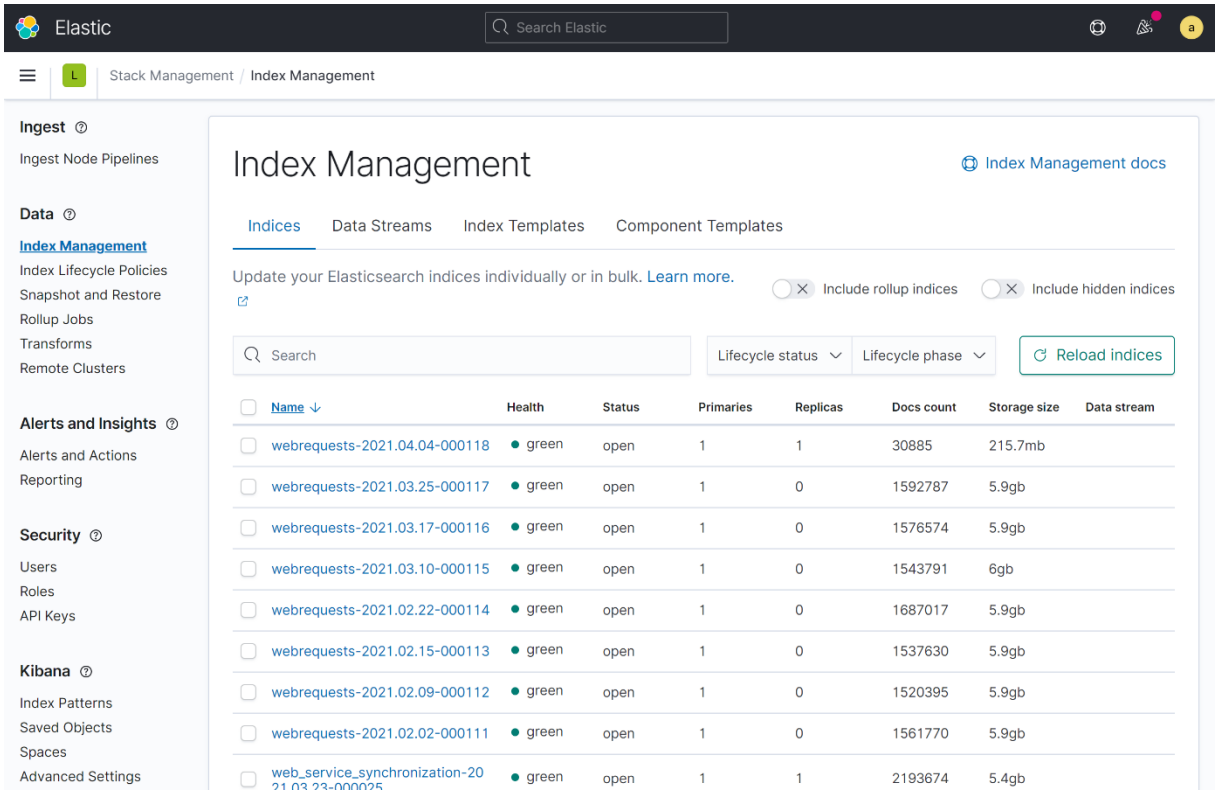
Druhá vizualizace slouží jako podpora vývoje a užitečná pomůcka pro vývojáře. Tato vizualizace umožňuje stanovit počty a délku trvání síťových dotazů na jednotlivé části webové aplikace. Obsahuje například počty, průměrnou a maximální dobu databázových dotazů, počty dotazů na veřejné API nebo počty dotazů na statický obsah, jako jsou obrázky a další. Na této vizualizaci je možné vidět tři oddělené sekce, kdy každá z nich zobrazuje data pouze pro jedno prostředí aplikace. Horní část zobrazuje data produkčního prostředí, zato spodní dvě sekce zobrazují vývojové a testovací prostředí. Vše je opět názorně viditelné na **Obrázek 23**.



Obrázek 23 *Vizualizace dotazů na webovou aplikaci (vlastní)*

9.3.4 Správa indexu

V sekci správa indexů se nachází všechny indexy. V kontextu této diplomové práce jich je zde momentálně téměř 100, přesto že data jsou sledována pouze z 20 částí aplikace. Tento počet je způsoben především ILM, které se stará o oddělení starších indexů a jejich přesun na warm node. Další využívaná funkcionalita v této sekci jsou šablony indexů. Ty udávají strukturu dat v indexu a jeho samotné nastavení. Zároveň jsou tyto šablony nutné pro správné fungování ILM. Tyto sekce lze vidět na **Obrázek 24**.



Obrázek 24 *Sekce správy indexů (vlastní)*

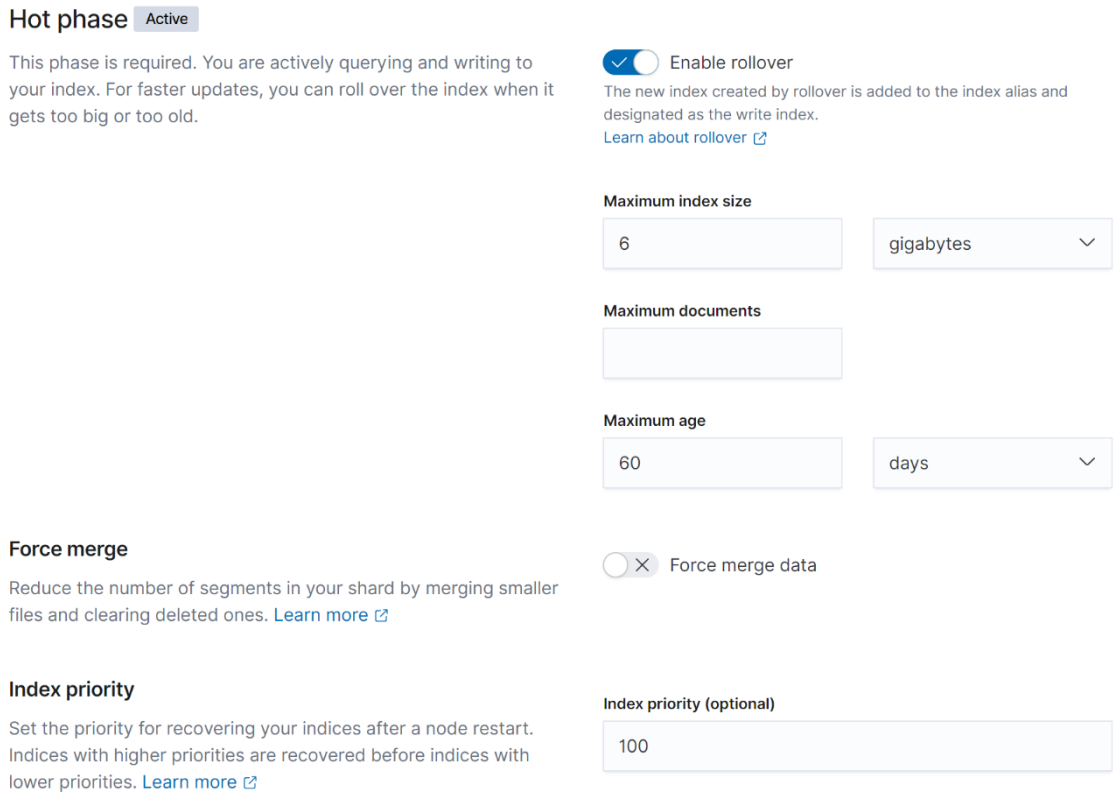
9.3.5 Nastavení ILM

Sekce ILM slouží k nastavení hot-warm architektury indexů. Tato architektura zprostředkovává indexy se staršími daty, na něž není kladen tak vysoký nárok na dostupnost. Takové indexy se přesunou na méně výkonné nody a uložiště. Lze tak ušetřit významnou část nákladů.

Uvedený příklad využívá ILM pro všechny definované indexy. Tím nejvíce vytíženým indexem z pohledu ILM je **webrequests**, sloužící pro logování dotazů na webovou aplikaci. Do tohoto indexu je každý den indexováno přes 250 000 logů, což odpovídá přibližně objemu dat o velikosti 1,5 GB. Tímto tempem by při provozu i přísunu dalších indexů, byla velice brzy vyčerpána kapacita přiřazených oddílů odpovídající velikosti pouze 100 GB (celková mapovaná velikost hot nodů odpovídá 200 GB, avšak všechny hot indexy zde obsahují primární shard i jeho repliku, zabírají tedy dvojnásobek místa). Z toho důvodu je index **webrequests** jeden z nejvíce vytížených a je nutné u něj

ILM nastavit. Zároveň je to index, na jehož ILM nastavení lze ukázat většinu možností, které tato funkcionalita nabízí.

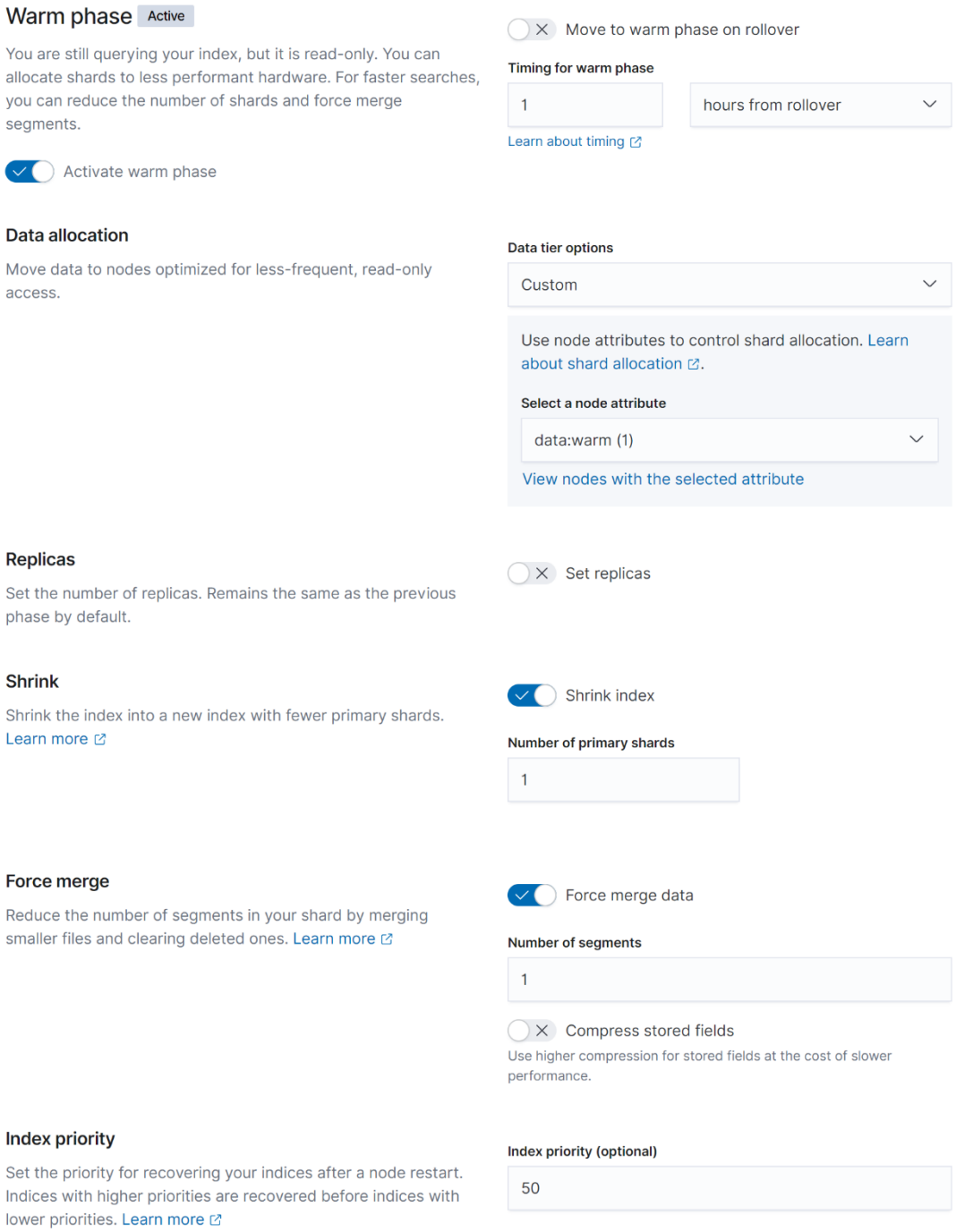
Nastavení ILM je řízeno separátními pravidly pro každý index a je rozděleno do čtyř fází: **hot**, **warm**, **cold** a **delete**. Fáze **hot** je základní fází, ve které se index nachází ihned po svém vzniku. Na **Obrázek 25** je možné vidět využívané nastavení, na kterém je patrné, že je rovněž použita funkce **rollover** obstarávající oddělení indexu při přechodu z fáze **hot** do fáze následující. K tomuto oddělení u níže uvedeného nastavení dochází při dosažení jedné z následujících podmínek. Buď index dosáhne velikosti 6 GB, nebo je starší než 60 dní (pro vyvolání oddělení postačuje dosažení jen jedné z podmínek). Funkcionalota **force merge** v této fázi není využívána a priorita indexu je nastavena na hodnotu 100, což odpovídá nejvyšší stanovené prioritě. Priorita indexu určuje, v jakém pořadí jsou indexy obnovovány při restartu nebo chybě ElasticSearch nodů.



Obrázek 25 *Hot fáze ILM pravidla (vlastní)*

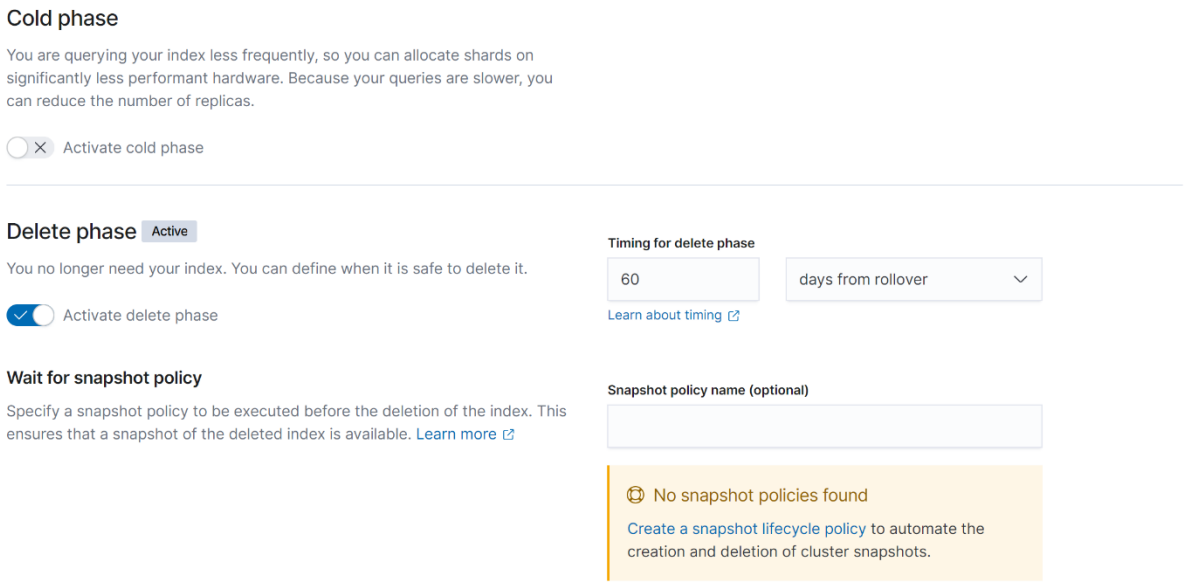
Další fází ILM je fáze **warm**. Nastavení této fáze je vidět na **Obrázek 26**. Indexy, které se oddělí, jsou dle tohoto nastavení po hodině od oddělení převedeny do fáze **warm**.

Zároveň se tímto převedením index alokuje na **warm** ElasticSearch node. Repliky jsou v této fázi již vypnuty a všechny shardy i segmenty jsou sloučeny do jednoho primárního shardu. Díky tomu se velikost indexu zmenší přibližně na polovinu. Priorita je pak v tomto případě nastavena na poloviční hodnotu, což zaručí, že tyto indexy budou při obnově, či restartu obslouženy jako poslední.



Obrázek 26 *Warm fáze ILM pravidla (vlastní)*

Ani fáze **cold** není v tomto nastavení využívána. Avšak při její aktivaci by možnosti nastavení odpovídaly fázi **warm**. Lze tak docílit mnohem škálovatelnějšího oddělování indexů. Poslední fází, do které se index může dostat, je fáze **delete**. Jakmile je index přesunut do této fáze, je automaticky nenávratně smazán. Jedinou možností návratu je obnovení celého clusteru z bodů obnovy. Pro tento účel je zde i nastavení, které vyčká na provedení nastavených záloh a až poté index nenávratně smaže. Pro účely této diplomové práce jsou **warm** indexy převedeny do fáze **delete** po 60 dnech od oddělení z fáze **hot**. Nastavení je opět znázorněné na následujícím obrázku.



Obrázek 27 *Cold a Delete fáze ILM pravidla (vlastní)*

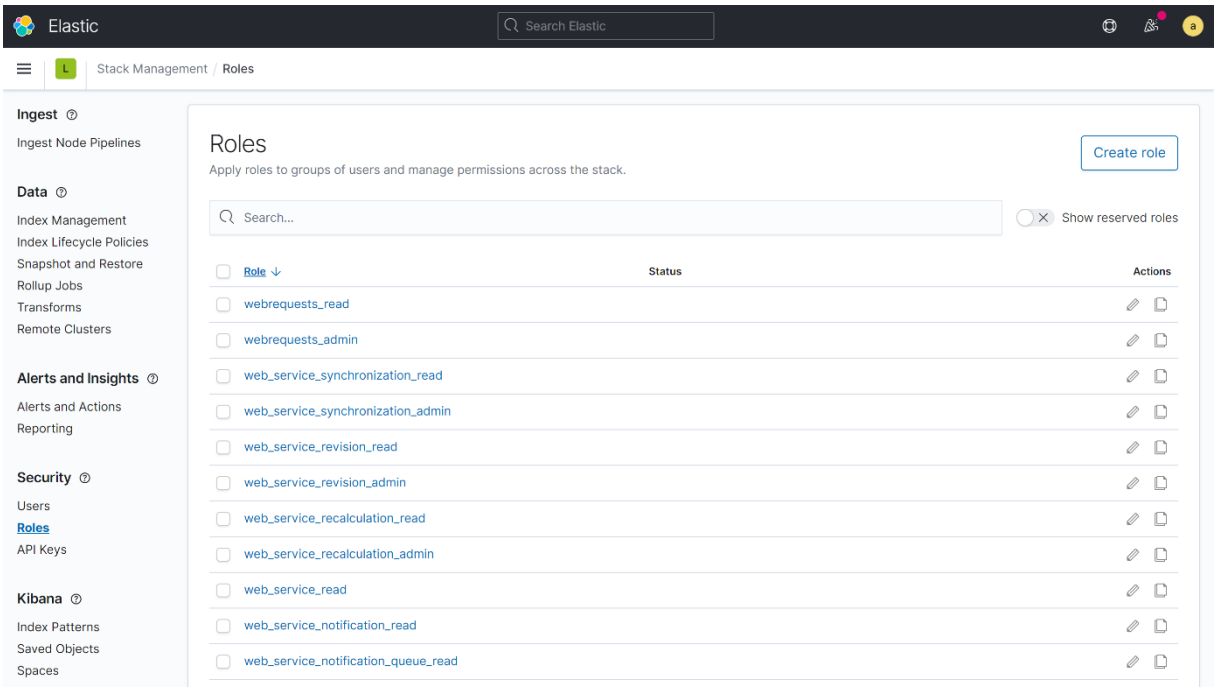
9.3.6 Správa uživatelů

Přístupové údaje uživatelů jsou spravovány v části zabezpečení. Jelikož ElasticSearch ve své open-source variantě nenabízí možnost autentizace třetích stran nebo napojení na Microsoft Active Directory, je v případě tohoto sestavení nutné využít pouze nativní zabezpečení. Pro tento typ zabezpečení lze vytvářet přístupové klíče, uživatelské role a uživatelské účty, které jsou poté perzistentně uloženy v systémových indexech. Ukázku sekce uživatelských rolí lze vidět na **Obrázek 28**.

Pro přístup vývojářů, analytiků a uživatelské podpory je využíván uživatelský účet **reader**, tento účet má přiřazené role umožňující přístup pouze do sekce Discovery

a k vizualizacím. V sekci s vizualizací pak uživatelé mohou vizualizace pouze prohlížet a filtrovat, nemohou je však měnit. Správce ElasticSearch Clusteru má přidělený vlastní uživatelský účet **admin** s přiřazenou rolí **superuser** zpřístupňující veškerou správu ElasticSearch Clusteru.

K dalšímu přístupu je využit separátní uživatelský účet pro instanci nástroje Logstash. Tento uživatelský účet umožňuje pouze indexaci dat, aby nedošlo k bezpečnostnímu riziku, při úniku přihlašovacích údajů obsažených v konfiguračních souborech nástroje Logstash.



Obrázek 28 *Sekce uživatelských rolí (vlastní)*

9.4 Beat

Všechny výše popsané nástroje z rodiny ELK Stack je nutné také monitorovat. Jejich monitoring přináší důležité údaje o vzniklých chybách a využití. V případě nástroje Logstash je možné sledovat metriky jednotlivých pipeline a propustnost zpráv. Naopak u nástroje ElasticSearch lze prostřednictvím monitoringu získat i údaje o probíhajících úlohách na pozadí, jako jsou ILM rollovery, zálohování ElasticSearch Clusteru nebo komunikace mezi jednotlivými ElasticSearch nody.

ELK Stack nabízí dvě možnosti jak tohoto monitoringu dosáhnout. První z možností je řízená přímo pomocí ElasticSearch Clusteru a jednotlivých nástrojů z rodiny ELK Stacku. Tato varianta se označuje jako self-monitoring. Pro její nastavení stačí v konfiguracích jednotlivých nástrojů zapnout parametr **self-monitoring** a ElasticSearch Cluster, ke kterému se dané nástroje připojují, vyřídí veškerou správu a sběr metrik. Bohužel tento způsob monitoringu byl v jedné z posledních verzí ElasticSearch označen jako deprecated (zastaralý) a společnost Elastic N.V. plánuje jeho kompletní odstranění v jedné z následujících majoritních verzí.

Druhou variantou pro sběr metrik a jejich zasílání do ElasticSearch, je využití nástroje Metricbeat. Jedná se o doporučený způsob. Jeho výhodou je nepřerušované monitorování při chybě monitorovaného nástroje, a také možnost jednotné konfigurace pro monitoring všech nástrojů ELK Stacku, ale i jiných nástrojů běžících uvnitř prostředí Kubernetes, jako je Redis nebo Kafka. Vzhledem k tomu, že tato metoda zůstane v budoucnu jediným způsobem, jak zajistit monitoring ELK Stacku, bylo rozhodnuto implementovat tuto metodu. Konkrétní konfigurací pro nasazení nástroje Metricbeat a jeho nastavením se zabývá následující kapitola.

9.4.1 Nasazení a konfigurace Metricbeat

Stejně jako při nasazení a konfiguraci nástroje Kibana lze i v tomto případě využít již nainstalovaného Elastic Cloud on Kubernetes, který obsahuje předinstalovaný Beat. Konfigurace je však o něco komplikovanější, přestože lze využít konfigurační soubor, který automaticky vytvoří instanci nástroje Beat, v tomto případě konkrétně typ Metricbeat. Složitost do této konfigurace přináší ta skutečnost, že na rozdíl od nástroje Kibana, neexistuje grafické rozhraní pro Metricbeat a je nutné jej nastavit uvnitř této konfigurace. Z toho důvodu zde budou uvedeny jen nejdůležitější částí konfigurace.

Nejdůležitější částí této konfigurace je **metricbeat.autodiscover** (viz **Zdrojový kód 16**). Tato část určuje, že Metricbeat bude sledovat ElasticSearch Cluster. V podsekci **templates** je pak uveden výčet jednotlivých monitorovaných nástrojů, prvním z nich je Kibana, následuje ElasticSearch, Beat (zajišťující monitoring sebe sama) a Logstash. U každého nástroje je pomocí proměnných nastavena jeho host adresa, přihlašovací údaje

a parametr **metricsets**. Tento parametr svým výčtem určuje, které typy metrik bude Metricbeat sbírat z uvedeného nástroje.

Dále je v konfiguračním souboru patrné nastavení přístupových údajů do ElasticSearch, ILM pro index, který si Metricbeat vytvoří, obecný předpis nasazení pro Kubernetes a mapování oddílů (viz **Zdrojový kód 17**). ILM je zde nastavováno pomocí separátního konfiguračního souboru (viz **Zdrojový kód 18**). Tímto způsobem lze zajistit konfiguraci ILM i bez grafického rozhraní Kibana, proto je zde uvedena i tato alternativní varianta nahrazující způsob popsaný pro ostatní spravované indexy, zmíněné v předchozí kapitole.

```
metricbeat.autodiscover:
  providers:
    - type: kubernetes
      scope: cluster
      hints.enabled: true
      templates:
        - condition:
            equals.kubernetes.labels.common_k8s_elastic_co/type: kibana
          config:
            - module: kibana
              metricsets:
                - stats
                - status
              period: 10s
              hosts: "http://${data.host}:${data.ports.http}"
              username: ${ES_USERNAME}
              password: ${ES_PASSWORD}
              ssl.verification_mode: "none"
              xpack.enabled: true
        - condition:
            equals.kubernetes.labels.common_k8s_elastic_co/type: beat
          config:
            - module: beat
              metricsets:
                - stats
                - state
              period: 10s
              hosts: "http://${data.host}:${data.ports.monitoring}"
              username: ${ES_USERNAME}
              password: ${ES_PASSWORD}
              ssl.verification_mode: "none"
              xpack.enabled: true
        - condition:
          logstash
          ...
          ..
          .
        - condition:
          elasticsearch
          ...
          ..
          .
```

Zdrojový kód 16 *Konfigurace Metricbeat část I. (vlastní)*

```
deployment:
  podTemplate:
    spec:
      nodeSelector:
        kubernetes.io/os: linux
      serviceAccountName: metricbeat
      automountServiceAccountToken: true
      securityContext:
        runAsUser: 0
      containers:
      - name: metricbeat
        ports:
        - containerPort: 5066
          name: monitoring
        env:
        - name: ES_USERNAME
          value: elastic
        - name: ES_PASSWORD
          valueFrom:
            secretKeyRef:
              key: elastic
              name: logeto-es-elastic-user
        volumeMounts:
        - name: ilm
          mountPath: /etc/ilm.json
          subPath: ilm.json
          readOnly: true
      volumes:
      - name: ilm
        configMap:
          name: metricbeat
          items:
          - key: ilm.json
            path: ilm.json
```

Zdrojový kód 17 *Konfigurace Metricbeat část II. (vlastní)*

```
{
  "policy": {
    "phases": {
      "hot": {
        "min_age": "0ms",
        "actions": {
          "rollover": {
            "max_age": "60d",
            "max_size": "2gb"
          },
          "set_priority": {
            "priority": 100
          }
        }
      },
      "warm": {
        "min_age": "1h",
        "actions": {
          "allocate": {
            "number_of_replicas": 0,
            "include": {},
            "exclude": {},
            "require": {
              "data": "warm"
            }
          },
          "set_priority": {
            "priority": 50
          },
          "shrink": {
            "number_of_shards": 1
          }
        }
      },
      "delete": {
        "min_age": "120d",
        "actions": {
          "delete": {}
        }
      }
    }
  }
}
```

Zdrojový kód 18 *Konfigurační soubor ILM pravidla pro Metricbeat (vlastní)*

10 Popis konfigurace nástroje ElastAlert

Jelikož Elastic Cloud on Kubernetes ve své bezplatné edici nenabízí možnost zasílání varovných zpráv do komunikátorů jako je Slack nebo MS Teams, bylo za účelem realizace této diplomové práce využito open-source nástroje třetích stran. Tímto nástrojem je ElastAlert, nabízející nejširší možnosti nastavení ze všech zdarma dostupných nástrojů podobného typu. V následujících kapitolách je popsáno jeho nasazení a konfigurace související s plněním cílů tohoto textu.

10.1 Nasazení nástroje ElastAlert

Nasazení nástroje ElastAlert je řízeno jedním souborem (viz **Zdrojový kód 19**), jedná se o konfigurační soubor Kubernetes prostředku Deployment. Jde tedy o nasazení obecné aplikace pomocí jejího docker image **elastalert-server:latest**. Tento docker image pochází z komunitního repozitáře **preacoapp**, jelikož společnost Yelp spravující ElastAlert nenabízí kontejnerizovanou variantu nástroje. I přesto lze tento docker image bez problémů použít, jelikož ho autor zakládá na oficiálním Git repozitáři ElastAlert, společnosti Yelp.

Pro nasazení nástroje je nutné prostřednictvím proměnných prostředí kontejneru nastavit přihlašovací údaje do nástroje ElasticSearch, kde je heslo opět díky Kubernetes načítáno ze Secret prostředku **logeto-es-elastic-user**. A následně také namapovat oddíly obsahující konfiguraci nástroje ElastAlert a jeho pravidel. Součástí mapovaných oddílů je také soubor **my_alerts.py** obsahující vlastní Python modul sloužící jako implementace podtřídy Alerter v nástroji ElastAlert.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: elastalert
  namespace: elasticsearch
spec:
  template:
    spec:
      containers:
      - image: praecoapp/elastalert-server:20201109
        name: elastalert
        env:
        - name: ES_USERNAME
          value: "elastic"
        - name: ES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: logeto-es-elastic-user
              key: elastic
        volumeMounts:
        - name: rules
          mountPath: /opt/elastalert/rules
        - name: config
          mountPath: /opt/elastalert/config.yaml
          subPath: config.yaml
        - name: config-json
          mountPath: /opt/elastalert-server/config/config.json
          subPath: config.json
        - name: alerts
          mountPath: /opt/elastalert/elastalert_modules/my_alerts.py
          subPath: my_alerts.py
      volumes:
        - name: config
          configMap:
            name: elastalert
            items:
            - key: config.yaml
              path: config.yaml
        - name: config-json
          configMap:
            name: elastalert
            items:
            - key: config.json
              path: config.json
        - name: alerts
          configMap:
            name: elastalert-alerts
            items:
            - key: my_alerts.py
              path: my_alerts.py
        - name: rules
          configMap:
            name: elastalert-rules
```

Zdrojový kód 19 *Deployment.yml pro nástroj ElastAlert (vlastní)*

10.2 Konfigurace nástroje ElastAlert

Konfigurace nástroje ElastAlert je řízena dvěma konfiguračními soubory. Oba konfigurační soubory jsou součástí Kubernetes prostředku ConfigMap, jako první je zde uveden konfigurační soubor **config.yml** a dalším je **config.json** (viz **Zdrojový kód 20**).

První zmíněný konfigurační soubor řídí, jakým způsobem aplikace vyhodnocuje pravidla. Lze zde nastavit, jak často budou pravidla vyhodnocována (v rámci tohoto intervalu musí vždy proběhnout i dotaz do ElasticSearch), bufferování dat nebo čas po kterém je alert zaslán znovu, v případě, že by došlo k chybě alertu. Dále se v tomto konfiguračním souboru nastavuje také připojení k ElasticSearch a writeback index, do kterého si ElastAlert zapisuje vlastní data, avšak tyto informace jsou duplicitně obsaženy i v následujícím konfiguračním souboru. [78]

Druhý zmíněný soubor slouží ke konfiguraci samotné aplikace běžící uvnitř kontejneru. Tato konfigurace nastaví jak se ElastAlert server chová, na kterém portu naslouchá, ze kterých adresářů čerpá konfiguraci pravidel a writeback index, do něhož si uvnitř ElasticSearch indexuje vlastní informace. Většina informací v tomto konfiguračním souboru je duplicitních, s prvním popisovaným souborem, avšak ElastAlert ve své kontejnerizované variantě neumí tato data přijímat z jednoho souboru a dle oficiální dokumentace je nutné tyto informace uvést v obou souborech.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: elastalert
  namespace: elasticsearch
  labels:
    app: elastalert
data:
  config.yaml: |-
    rules_folder: /opt/elastalert/rules
    scan_subdirectories: false
    run_every:
      seconds: 10
    buffer_time:
      minutes: 5
    es_host: logeto-es-http
    es_port: 9200
    writeback_index: elastalert_status
    alert_time_limit:
      days: 0
  config.json: |-
    {
      "appName": "elastalert-server",
      "port": 3030,
      "wsport": 3333,
      "elastalertPath": "/opt/elastalert",
      "verbose": false,
      "es_debug": false,
      "debug": false,
      "rulesPath": {
        "relative": true,
        "path": "/rules"
      },
      "templatesPath": {
        "relative": true,
        "path": "/rule_templates"
      },
      "es_host": "logeto-es-http",
      "es_port": 9200,
      "es_username": "",
      "es_password": "",
      "es_ssl": false,
      "writeback_index": "elastalert_status"
    }
```

Zdrojový kód 20 *Konfigurace nástroje ElastAlert (vlastní)*

10.3 Konfigurace pravidel

Posledním konfiguračním souborem, který je nutné pro ElastAlert specifikovat, jsou samotná pravidla alertů. Tento soubor je opět Kubernetes konfiguračním prostředkem ConfigMap, v němž se nacházejí konkrétní konfigurační soubory pravidel,

které se následně namapují do vnitřní souborové struktury ElastAlert kontejneru. Pro každý index uvnitř ElasticSearch je nastavené alespoň jedno pravidlo, a to z důvodu, že ke každému indexu mohou dorazit zprávy typu **Report,** což je z pohledu vývoje nejzávažnější typ chyby a je důležité, aby o něm byl vývojář okamžitě informován. Jelikož by soubor obsahující všechna pravidla byl příliš dlouhý, je v příloze vypuštěna většina pravidel konfigurace. Zachována zůstala pouze pravidla **feedback_immedialy**, sloužící k zasílání logů zařízení a aplikace z klientských aplikací pro operační systém Android a iOS a pravidlo **webrequests_immediately** pro dotazy na webovou aplikaci. Obě jsou pak popsána jako separátní zdrojový kód (viz **Zdrojový kód 21** a **Zdrojový kód 22**).

Prvním popisovaným pravidlem je pravidlo **feedback_immedialy**. V jeho konfiguraci lze vidět nastavení názvu a typu, název ElasticSearch indexu, ze kterého pravidlo čte data, realert určující po jaké době má dojít k odeslání nového varování pokud je událost stále aktuální a následně filter, který je v tomto případě typu query a je tedy nutné uvést validní ElasticSearch query. Zde vyvolávaná query zohledňuje jen to, aby data obdržená z ElasticSearch nebyla typu debug, tedy aby data pocházela jen od uživatelů a nedocházelo k zasílání zpráv pocházejících z vývoje. Zbytek pravidel již nastavuje alerter.

V tomto konkrétním případě se jedná o vlastní alerter pro komunikační nástroj MS Teams. Pro tento alerter je nutné uvést řadu vstupních parametrů. Povinným parametrem je zde jen **ms_teams_webhook_url**, obsahující url webhook komunikátoru na který se zpráva zasílá, **ms_teams_alert_summary,** obsahující pole nastavené jako hlavička odesílané zprávy a **ms_teams_index_pattern_url** obsahující Kibana url indexu, ze kterého daná data pochází. Ostatní parametry již specifikují pouze formu a vzhled zaslané zprávy. Konkrétní fungování tohoto alerteru je vysvětleno v následující kapitole.

Druhé pravidlo **webrequests_immediately** se svou formou od předchozího výrazně neliší, analogicky zde jsou doplněny potřebné údaje a jedinou výraznou změnou je index ze kterého pravidlo čerpá, query na ElasticSearch a formát správy pro alerter. Query v tomto případě zohledňuje také to, aby zpráva byla typu **Report**, tedy nejzávažnější typ zalogované chyby aplikace.

```
name: Feedback immediately reports
type: any
index: feedback-*
realert:
  seconds: 0
filter:
- query:
    query_string:
      query: "DebugMode: false"
alert:
- "elastalert_modules.my_alerts.myMsTeamsAlerter"
ms_teams_webhook_url:
- "webhook_url"
ms_teams_alert_summary: "Feedback report immediately"
ms_teams_theme_color: "0076D7"
ms_teams_alert_title: "Text"
ms_teams_index_pattern_url: "kibana_url"
ms_teams_alert_fields:
  - name: OS
    value: OS
  - name: ApplicationVersion
    value: ApplicationVersion
  - name: AccountName
    value: AccountName
  - name: Timestamp
    value: Timestamp
ms_teams_alert_links:
  - name: "Download database"
    value: Database
  - name: "Download device log"
    value: DeviceLog
  - name: "Download app log"
    value: ApplicationLog
```

Zdrojový kód 21 *Konfigurace pravidla feedback_immedialy (vlastní)*

```
name: Webrequests immediately reports
type: any
index: webrequests-*
realert:
  seconds: 0
filter:
- query:
    query_string:
      query: "LogType: Report AND NOT SourcePartition: local AND NOT
DebugMode: true"
alert:
- "elastalert_modules.my_alerts.myMsTeamsAlerter"
ms_teams_webhook_url:
- "webhook_url"
ms_teams_alert_summary: "Webrequests report immediately"
ms_teams_theme_color: "0076D7"
ms_teams_alert_title: "Message"
ms_teams_index_pattern_url: "kibana_url"
ms_teams_alert_fields:
  - name: Message
    value: Message
  - name: ModuleVersion
    value: ModuleVersion
  - name: Function
    value: Function
  - name: Timestamp
    value: Timestamp
```

Zdrojový kód 22 *Konfigurace pravidla webrequests_immediately (vlastní)*

11 Alerter modul pro MS Teams

Poslední část této diplomové práce je věnována implementaci vlastního alerteru pro nástroj ElastAlert. Jelikož ElastAlert obsahuje jen velice generický alerter k zasílání varovných zpráv do komunikačního nástroje MS Teams, bylo zapotřebí vytvořit vlastní alerter, umožňující alespoň základní personalizaci zasílané zprávy. Alerter je nutné programovat jako podtřídu generického alerteru **Alerter**, již obsaženého v nástroji ElastAlert, a je nutné jej programovat v jazyce Python.

Z generické třídy **Alerter** je možné dědit velké množství funkci a parametrů, některé z těchto funkcí je nutné implementovat, jiné jsou volitelné. Vytvořený alerter velkou částí pochází z předepsané struktury obsahující všechny povinné funkce, tuto strukturu lze nalézt v oficiální dokumentaci [80]. Tato dokumentace velice srozumitelně popisuje veškeré části, které je nutné v alerteru implementovat, tak aby validně fungoval v rámci celého nástroje.

Obsahem implementovaného modulu je několik funkcí, mezi ty povinně implementované patří jen funkce **alert** a **get_info**. Funkce **alert** obstarává samotné zaslání alertu, naopak funkce **get_info** slouží jako callback, který ElastAlert vyvolává po odeslání alertu a může být využit například k zapsání výsledného stavu alertu do writeback indexu uvnitř ElasticSearch. Mezi ty nepovinně implementované patří především funkce pro extrakci dat z pravidla a následně jejich dosazení do formátu zprávy pro MS Teams webhook. Všechny funkce jsou detailněji popsány v následujících kapitolách

11.1 Formát zprávy

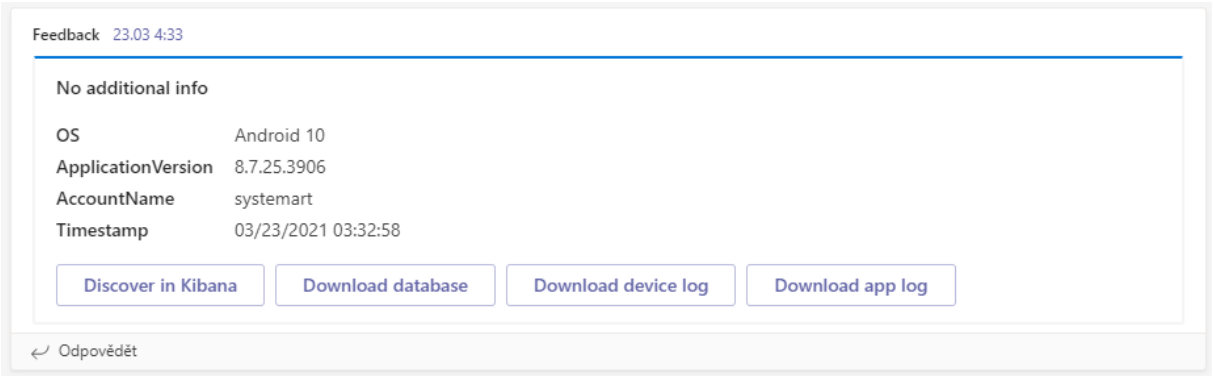
Základní alerter pro komunikátor MS Teams obsahuje předdefinovanou zprávu ve formátu JSON, která obsahuje pouze nadpis a nestrukturovaný obsah, v němž se nachází celý obsah zprávy získané z ElasticSearch indexu. Z toho důvodu bylo nutné vytvořit vlastní alerter a především vlastní formát zasílané zprávy, který by splňoval stanovené požadavky na upravitelnost.

Microsoft nazývá tento formát zprávy Cards, jelikož stejný formát zprávy lze zpracovat téměř všemi aplikacemi, ke kterým společnost Microsoft zprostředkovává webhook. Typů karet (cards) je několik, rozdělených dle jejich účelu. V tomto případě byl použit typ Adaptive card, který může obsahovat libovolnou kombinaci textu, obrázků, tlačítek nebo vstupních polí.

Požadavkem bylo, aby karta zobrazená v komunikátoru MS Teams obsahovala definovaný nadpis. V těle karty pak musí být strukturovaný výpis pouze požadovaných polí ze zprávy indexované uvnitř ElasticSearch. A na závěr musí karta obsahovat akční tlačítka, díky kterým se lze dostat do prostředí Kibana. Při vývoji bylo využito prostředí Message Card Playground [79], které společnost Microsoft nabízí pro vizualizaci vytvořených karet. Finální implementací karty je následující JSON obsahující požadovanou strukturu. To jak karta vypadá již vyplněná v komunikátoru MS Teams je zobrazeno na **Obrázek 29**.

```
{
    "@type": "MessageCard",
    "@context": "http: //schema.org/extensions",
    "themeColor": "FF0000",
    "summary": "test",
    "sections": [
        {
            "activityTitle": "Message",
            "facts": [
                {
                    "name": "Name",
                    "value": "Value"
                }
            ],
            "markdown": true
        }
    ],
    "potentialAction": [
        {
            "@type": "OpenUri",
            "name": "Discover in Kibana",
            "targets": [
                {
                    "os": "default",
                    "uri": "url"
                }
            ]
        }
    ]
}
```

Zdrojový kód 23 *Předpis karty pro MS Teams (vlastní)*



Obrázek 29 *Vizuální interpretace karty pro MS Teams (vlastní)*

11.2 Extrakce dat

K tomu, aby karta obsahovala relevantní data, je nutné dekomponovat zprávu, kterou ElastAlert obdrží z ElasticSearch, a extrahovaná data dosadit do správných sekcí výše popsané karty. K tomuto účelu bylo implementováno několik funkcí alerteru, které extrakci obstarávají. První a nejjednodušší funkcí je **populate_title**, ta vezme všechna pole, která obsahuje zpráva z ElasticSearch a navrátí pouze hodnotu pole se shodným názvem uvedeným ve vstupní proměnné **ms_teams_alert_title**.

```
def populate_title(self, matches):
        return lookup_es_key(matches[0], self.ms_teams_alert_title)
```

Zdrojový kód 24 *Extrakce nadpisu funkcí populate_title (vlastní)*

Další funkcí je **populate_fields**. Zde je využíváno skutečnosti, že formáty YAML a JSON jsou vždy vzájemně kompatibilní a převoditelné. Funkce tak vezme pole atributů klíč-hodnota uvedené v konfiguraci ElastAlert pravidla (ve formátu YAML) a nahradí hodnoty těchto atributů hodnotami ekvivalentních polí obsažených ve zprávě z ElasticSearch. Následně jsou tato pole ve funkci **alert** dosazena do karty pro MS Teams komunikátor, která je již ve formátu JSON.

```
def populate_fields(self, matches):
    alert_fields = []
    for arg in self.ms_teams_alert_fields:
        arg = copy.copy(arg)
        arg['value'] = lookup_es_key(matches[0], arg['value'])
        alert_fields.append(arg)
    return alert_fields
```

Zdrojový kód 25 *Extrakce datových polí funkcí populate_fields (vlastní)*

Poslední a nejsložitější funkcí je **populate_links** (viz **Zdrojový kód 26**), ta do karty pro MS Teams dosazuje akční tlačítka obsahující externí odkaz. Funkce je rozdělena do dvou částí, které obecně fungují stejně. První část dosazuje akční tlačítko obsažené u každé zprávy. Toto tlačítko obsahuje statický odkaz indexu v prostředí Kibana, ke kterému je připojen identifikátor konkrétní zprávy, na kterou uživatele odkáže. Druhá část slouží pouze pro pravidlo **feedback_immedialy**, jelikož zprávy v ElasticSearch indexu **feeback** obsahují odkazy na Azure Blob Storage, do kterého telefony ukládají celý zaslaný soubor logů, o němž daná zpráva informuje. Díky těmto akčním tlačítkům je tak možné odkázat se na tyto logy a stáhnout si je přímo z Azure Blob Storage. Samotná extrakce poté funguje stejným způsobem jako u předchozích dvou funkcí. U akčních tlačítek je pouze o něco složitější struktura karty pro MS Teams, do které je hodnoty nutné dosadit.

```
def populate_links(self, matches):
    alert_links = []
    if self.ms_teams_index_pattern_url != '':
        document_id = lookup_es_key(matches[0], 'UniqueId')
        my_url = '%s%s' % (self.ms_teams_index_pattern_url,document_id)
        name = "Discover in Kibana"

        current_link_pattern = copy.copy(self.link_pattern)
        current_target_pattern = copy.copy(self.target_pattern)

        current_link_pattern['name'] = name

        target_wrapper = []
        target_wrapper.append(current_target_pattern)
        current_link_pattern['targets'] = target_wrapper
        current_link_pattern['targets'][0]['uri'] = my_url

        alert_links.append(current_link_pattern)
    if self.ms_teams_alert_links != '':
        for arg in self.ms_teams_alert_links:
            link_url = lookup_es_key(matches[0], arg['value'])
            name = arg['name']

            current_link_pattern = copy.copy(self.link_pattern)
            current_target_pattern = copy.copy(self.target_pattern)

            if link_url != '' and link_url is not None:
                current_link_pattern['name'] = name

                target_wrapper = []
                target_wrapper.append(current_target_pattern)
                current_link_pattern['targets'] = target_wrapper
                current_link_pattern['targets'][0]['uri'] = link_url

                alert_links.append(current_link_pattern)
    return alert_links
```

Zdrojový kód 26 *Extrakce odkazů pomocí funkce populate_links (vlastní)*

11.3 Zaslání alertu a doplnění formátu zprávy

Poslední částí vytvořeného alerteru je povinná funkce **alert**. Tato funkce se stará o odesílání zprávy do komunikátoru MS Teams a dosazování extrahovaných dat do šablony karty pro MS Teams. Nachází se zde také část zapisující výsledek sestavené karty do logovacího souboru pro případ odladění chyb. Celou funkci **alert** je možné vidět na **Zdrojový kód 27**.

```
def alert(self, matches):
    headers = {'content-type': 'application/json'}
    proxies = {'https': self.ms_teams_proxy} if self.ms_teams_proxy else None

    payload = {
        '@type': 'MessageCard',
        '@context': 'http://schema.org/extensions',
        'themeColor': self.ms teams theme color,
        'summary': self.ms_teams_alert_summary,
        'sections': [
            {
                'activityTitle': self.create_title(matches),
                'facts': [],
                'markdown': True
            }
        ],
        'potentialAction': []
    }

    if self.ms_teams_alert_title != '':
       payload['sections'][0]['activityTitle'] =
       self.populate_title(matches)

    if self.ms_teams_alert_fields != '':
       payload['sections'][0]['facts'] = self.populate_fields(matches)

    if self.ms_teams_alert_links != '' or self.ms_teams_index_pattern_url != '':
       payload['potentialAction'] = self.populate_links(matches)

    with
       open('/opt/elastalert/
              elastalert_modules/
              alerter_test_output.log', 'a') as outfile:
       json.dump(payload, outfile)

    for url in self.ms_teams_webhook_url:
        try:
            response = requests.post(
              url,
              data=json.dumps(
                     payload,
                     cls=DateTimeEncoder),
              headers=headers,
              proxies=proxies)
            response.raise_for_status()
        except RequestException as e:
            raise EAException("Error posting to ms teams: %s" % e)
    elastalert_logger.info("Alert sent to MS Teams")
```

Zdrojový kód 27 *Odeslání alertu do MS Teams funkcí alert (vlastní)*

11.4 Kompletní Python module

Následující zdrojový kód obsahuje celý funkční Python modul pro zasílání personalizovaných zpráv do komunikátoru MS Teams.

```
import requests
import json
import copy
from elastalert.alerts import Alerter, BasicMatchString, DateTimeEncoder
from elastalert.util import lookup_es_key, elastalert_logger, EAException
from requests.exceptions import RequestException

class myMsTeamsAlerter(Alerter):

    required_options = set([
       'ms_teams_webhook_url',
       'ms_teams_alert_summary',
       'ms_teams_index_pattern_url'])

    link_pattern = {
        '@type': 'OpenUri',
        'name': '',
        'targets': []
    }

    target_pattern = {
        'os': 'default',
        'uri': ''
    }

    def __init__(self, rule):
        super(myMsTeamsAlerter, self).__init__(rule)
        self.ms_teams_webhook_url = self.rule['ms_teams_webhook_url']
        if isinstance(self.ms_teams_webhook_url, str):
            self.ms_teams_webhook_url = [self.ms_teams_webhook_url]
        self.ms_teams_proxy = self.rule.get('ms_teams_proxy', None)
        self.ms_teams_alert_summary = self.rule.get(
              'ms_teams_alert_summary',
              'ElastAlert Message')
        self.ms_teams_alert_title = self.rule.get('ms_teams_alert_title', '')
        self.ms_teams_alert_fields = self.rule.get('ms_teams_alert_fields', '')
        self.ms_teams_theme_color = self.rule.get('ms_teams_theme_color', 'FF0000')
        self.ms_teams_index_pattern_url = self.rule.get(
              'ms_teams_index_pattern_url',
              '')
        self.ms_teams_alert_links = self.rule.get('ms_teams_alert_links', '')

    def populate_fields(self, matches):
        alert_fields = []
        for arg in self.ms_teams_alert_fields:
            arg = copy.copy(arg)
            arg['value'] = lookup_es_key(matches[0], arg['value'])
            alert_fields.append(arg)
        return alert_fields
```

```
    def populate_title(self, matches):
        return lookup_es_key(matches[0], self.ms_teams_alert_title)

    def populate_links(self, matches):
        alert_links = []
        if self.ms_teams_index_pattern_url != '':
            document_id = lookup_es_key(matches[0], 'UniqueId')
            my_url = '%s%s' % (self.ms_teams_index_pattern_url,document_id)
            name = "Discover in Kibana"

            current_link_pattern = copy.copy(self.link_pattern)
            current_target_pattern = copy.copy(self.target_pattern)

            current_link_pattern['name'] = name

            target_wrapper = []
            target_wrapper.append(current_target_pattern)
            current_link_pattern['targets'] = target_wrapper
            current_link_pattern['targets'][0]['uri'] = my_url

            alert_links.append(current_link_pattern)
        if self.ms_teams_alert_links != '':
            for arg in self.ms_teams_alert_links:
                link_url = lookup_es_key(matches[0], arg['value'])
                name = arg['name']

                current_link_pattern = copy.copy(self.link_pattern)
                current_target_pattern = copy.copy(self.target_pattern)

                if link_url != '' and link_url is not None:
                    current_link_pattern['name'] = name

                    target_wrapper = []
                    target_wrapper.append(current_target_pattern)
                    current_link_pattern['targets'] = target_wrapper
                    current_link_pattern['targets'][0]['uri'] = link_url

                    alert_links.append(current_link_pattern)
        return alert_links

    def alert(self, matches):
        headers = {'content-type': 'application/json'}
        proxies = {'https': self.ms_teams_proxy} if self.ms_teams_proxy else None

        payload = {
            '@type': 'MessageCard',
            '@context': 'http://schema.org/extensions',
            'themeColor': self.ms_teams_theme_color,
            'summary': self.ms_teams_alert_summary,
            'sections': [
                {
                    'activityTitle': self.create_title(matches),
                    'facts': [],
                    'markdown': True
                }
            ],
            'potentialAction': []
        }
```

```
        if self.ms_teams_alert_title != '':
            payload['sections'][0]['activityTitle'] = self.populate_title(matches)

        if self.ms_teams_alert_fields != '':
            payload['sections'][0]['facts'] = self.populate_fields(matches)

        if self.ms_teams_alert_links != '' or self.ms_teams_index_pattern_url != '':
            payload['potentialAction'] = self.populate_links(matches)

        with open('/opt/elastalert/
              elastalert_modules/
              alerter_test_output.log', 'a') as outfile:
            json.dump(payload, outfile)

        for url in self.ms_teams_webhook_url:
            try:
                response = requests.post(
                      url,
                      data=json.dumps(
                             payload,
                             cls=DateTimeEncoder),
                      headers=headers,
                      proxies=proxies)
                response.raise_for_status()
            except RequestException as e:
                raise EAException("Error posting to ms teams: %s" % e)
        elastalert_logger.info("Alert sent to MS Teams")

    def get_info(self):
        return {'type': 'ms_teams',
                'ms_teams_webhook_url': self.ms_teams_webhook_url}
```

Zdrojový kód 28 *Modul pro zasílání personalitovaných alertů do MS Teams (vlastní)*

12 Závěr

Práci, lze obecně považovat za zdařilou. Čtenáře srozumitelnou formou seznamuje s použitými nástroji a s koncepty jejich fungování, popisuje postup implementace logovací infrastruktury nad aplikací Logeto a implementuje vlastní Python modul pro zasílání personalizovaných varovných zpráv z nástroje ElastAlert do komunikačního nástroje MS Teams.

Hlavním přínosem práce je funkční logovací infrastruktura přinášející společnosti Systemart s.r.o. možnost centralizovaně nahlížet na logovaná data aplikací z jednotného grafického rozhraní. Implementace logovací infrastruktury, která je předmětem této diplomové práce, objevila množství chyb, s nimiž se doposud museli reální uživatelé aplikace potýkat. V důsledku předchozí absence sběru logů klientských aplikací zůstávaly tyto chyby neodhaleny. Taktéž se podařilo odhalit periodiku některých interních chyb systému, které byly do té doby vnímány pouze jako sporadické anomálie.

Další přínosnou částí práce je vlastní Python modul pro nástroj ElastAlert. Tento modul umožňuje alespoň částečnou personalizaci zasílaných zpráv pro komunikační nástroj MS Teams. Společnost Systemart s.r.o. nyní může sledovat veškeré závažné chyby přímo v interním komunikačním nástroji, bez nutnosti přistupovat do jiných nástrojů. Zároveň jsou informace obsažené ve varovných zprávách srozumitelně formátovány a každý analytik, vývojář i zákaznická podpora jsou ihned informováni o nastalém problému.

I přes tyto přínosy zůstává ještě několik nedořešených úkolů spojených s celou logovací infrastrukturou. Mimo to, že je nutné infrastrukturu dále udržovat a aktualizovat její části (především často aktualizovaný ElasticSearch), zbývá ještě rozšířit infrastrukturu o monitoring Kubernetes clusteru a rozšířit Python modul pro varovné zprávy, tak aby se svou možností personalizace blížil ostatním modulům obsaženým v nástroji ElastAlert.

Monitoring Kubernetes clusteru bude dále řešen pomocí nástroje Prometheus a Grafana. Pro monitorované metriky bude nutné taktéž zasílat varovné zprávy do

komunikačního nástroje MS Teams – k tomuto účelu slouží nástroj PromTeams. Rozšíření Python modulu bude pokračovat na základě dodatečných požadavků. Po dosažení personalizace odpovídající jiným modulům nástroje ElastAlert, budeme usilovat o jeho zařazení mezi ostatní moduly v Git repozitáři projektu.

Ve spojitosti s touto prací je pak důležité si uvědomit, že zvolené postupy a nástroje byly vybrány pro konkrétní společnost a aplikaci. V případě jiné společnosti, nebo aplikace, může dojít k výběru jiných, vhodnějších nástrojů a přístupů, nebo jiné konfigurace odpovídající konkrétním požadavkům.

13 Seznam použité literatury

[1] **CHHAJED, Saurabh**. *Learning ELK Stack* [online]. Birmingham, UK: Packt Publishing, 2015 [cit. 2021-04-01]. ISBN 978-1-78588-715-4. Dostupné z: https://www.amazon.com/Learning-ELK-Stack-Saurabh-Chhajed/dp/1785887157

[2] **SHARMA, Vishal**. *Beginning Elastic Stack* [online]. Berkeley, CA: Apress, 2016 [cit. 2021-04-01]. ISBN 978-1-4842-1693-4. Dostupné z: https://doi.org/10.1007/978-1-4842-1694-1

[3] **GUPTA, Yuvraj**. *Mastering Elastic Stack* [online]. Birmingham, UK: Packt Publishing, 2017 [cit. 2021-04-01]. ISBN 978-1-78646-001-1. Dostupné z: https://www.amazon.com/Mastering-Elastic-Stack-Yuvraj-Gupta/dp/1786460017

[4] *Elastic Stack and Product Documentation* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/index.html

[5] *ElastAlert: Easy & Flexible Alerting With Elasticsearch* [online]. 2016. United States: Yelp, 2015 [cit. 2021-04-01]. Dostupné z: https://elastalert.readthedocs.io/

[6] **BERGER, Jan**. *Logování pomocí ELK stacku* [online]. Praha, 2016 [cit. 2021-04-01]. Dostupné z: https://theses.cz/id/wwlmg2/. Diplomová práce. Vysoká škola ekonomická v Praze. Vedoucí práce Helena Palovská.

[7] **MURÍNOVÁ, Júlia**. *Application Log Analysis* [online]. Brno, 2016 [cit. 2021-04-01]. Dostupné z: https://theses.cz/id/moz8l2/. Diplomová práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce doc. RNDr. Vlastislav Dohnal, Ph.D.

[8] **ŘÁDKOVÁ, Pavla**. *Využití technologie Elastic Stack pro sběr a analýzu logů* [online]. Praha, 2019 [cit. 2021-04-01]. Dostupné z: https://theses.cz/id/hjew5r/. Bakalářská práce. Vysoká škola ekonomická v Praze. Vedoucí práce Jan Pokorný.

[9] **ADÁMIK, Samuel**. *Analýza síťového provozu pomocí nástrojů rodiny Elastic* [online]. Brno, 2017 [cit. 2021-04-01]. Dostupné z: https://is.muni.cz/th/udttf/. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Tomáš JIRSÍK.

[10] **BURDA, Tomáš**. *Využití Dockeru pro správu aplikací v kontejnerech* [online]. Hradec Králové, 2020 [cit. 2021-04-01]. Dostupné z: https://theses.cz/id/7qmy20/. Diplomová práce. Univerzita Hradec Králové, Fakulta informatiky a managementu. Vedoucí práce doc. RNDr. Petra Poulová, Ph.D.

[11] *What is the ELK Stack?* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/what-is/elk-stack

[12] **JEŽEK, Pavel**. *Pohled do zákulisního mozku Alzy aneb jak vypadá vývoj našeho interního informačního systému*. Czechcrunch [online]. 2019 [cit. 2021-04-01]. Dostupné z: https://www.czechcrunch.cz/2019/08/pavel-jezek-pohled-do-zakulisniho-mozku-alzy-aneb-jak-vypada-vyvoj-naseho-interniho-informacniho-systemu/

[13] **BERMAN, Daniel**. *Creating an Elasticsearch Cluster: Getting Started*. Logz.io [online]. 2020 [cit. 2021-04-01]. Dostupné z: https://logz.io/blog/elasticsearch-cluster-tutorial/

[14] *Node* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-node.html

[15] **NATHANI, Ronak**. *Anatomy of an Elasticsearch Cluster: Part I*. Medium [online]. 2016 [cit. 2021-04-01]. Dostupné z: https://blog.insightdatascience.com/anatomy-of-an-elasticsearch-cluster-part-i-7ac9a13b05db

[16] **NATHANI, Ronak**. *Anatomy of an Elasticsearch Cluster: Part II*. Medium [online]. 2016 [cit. 2021-04-01]. Dostupné z: https://blog.insightdatascience.com/anatomy-of-an-elasticsearch-cluster-part-ii-6db4e821b571

[17] *Mapping concepts across SQL and Elasticsearch* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/ mapping concepts across sql and elasticsearch.html

[18] *Index vs. Type* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/blog/index-vs-type

[19] **BERMAN, Daniel**. *Elasticsearch Mapping: The Basics, Updates & Examples*. Logz.io [online]. 2020 [cit. 2021-04-02]. Dostupné z: https://logz.io/blog/elasticsearch-mapping/

[20] *An Introduction to Elasticsearch Mapping* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/blog/found-elasticsearch-mapping-introduction

[21] *Index templates* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/index-templates.html

[22] *Implementing a Hot-Warm-Cold Architecture with Index Lifecycle Management* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/blog/implementing-hot-warm-cold-in-elasticsearch-with-index-lifecycle-management

[23] **GIGASEARCH ENGINEERING**. *Elasticsearch Hot-Warm Architecture*. Medium [online]. 2016 [cit. 2021-04-02]. Dostupné z: https://medium.com/gigasearch/elasticsearch-hot-warm-architecture-8322ed229388

[24] *Logstash: Centralize, transform & stash your data* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/logstash

[25] *Logstash Introduction* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/logstash/current/introduction.html

[26] *Persistent Queues* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/logstash/current/persistent-queues.html

[27] *Introducing Multiple Pipelines in Logstash* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/blog/logstash-multiple-pipelines

[28] *Input plugins* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/logstash/current/input-plugins.html

[29] *Output plugins* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/logstash/current/output-plugins.html

[30] *Filter plugins* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/logstash/current/filter-plugins.html

[31] *Persistent Queues* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/logstash/current/persistent-queues.html

[32] *Beats: Lightweight data shippers* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/beats/

[33] *What are Beats?* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/beats/libbeat/current/beats-reference.html

[34] *Auditbeat: Lightweight shipper for audit data* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/beats/auditbeat

[35] *Functionbeat: Serverless shipper for cloud data* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/beats/functionbeat

[36] *Heartbeat: Lightweight shipper for uptime monitoring* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/beats/heartbeat

[37] *Journalbeat* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/beats/journalbeat/current/journalbeat-overview.html

[38] *Metricbeat: Lightweight shipper for metrics* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/beats/metricbeat

[39] *Packetbeat: Lightweight shipper for network data* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/beats/packetbeat

[40] *Winlogbeat: Lightweight shipper for Windows event logs* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/beats/winlogbeat

[41] *Filebeat: Lightweight shipper for logs* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/beats/filebeat

[42] *Kibana: Your window into the Elastic Stack* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/kibana

[43] *What is Kibana?* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/what-is/kibana

[44] *Kibana Query Language* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/kuery-query.html

[45] *Lucene query syntax* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/lucene-query.html

[46] *Kibana: Discover* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/discover.html

[47] *Kibana: Dashboard* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/dashboard.html

[48] *Kibana: Machine learning* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/xpack-ml.html

[49] *Elastic Enterprise Search: Search everything, anywhere* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/enterprise-search

[50] *Elastic Observability: Unified visibility across your entire ecosystem* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/observability

[51] *Kibana: Observability* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/observability.html

[52] *Elastic Security: Unified protection for everyone, built on the Elastic (ELK) Stack* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/security

[53] *Kibana: Elastic Security* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/xpack-siem.html

[54] *Kibana: Management* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/guide/en/kibana/current/management.html

[55] *ElastAlert: Easy & Flexible Alerting With Elasticsearch* [online]. San Francisco, CA: Yelp, 2021 [cit. 2021-04-01]. Dostupné z: https://elastalert.readthedocs.io/en/latest/elastalert.html

[56] *ElastAlert* [online]. San Francisco, CA: Yelp [cit. 2021-04-01]. Dostupné z: https://github.com/Yelp/elastalert

[57] *ElastAlert: Rule Types and Configuration Options* [online]. San Francisco, CA: Yelp, 2021 [cit. 2021-04-01]. Dostupné z: https://elastalert.readthedocs.io/en/latest/ruletypes.html#

[58] *ElastAlert: Alerts* [online]. San Francisco, CA: Yelp, 2021 [cit. 2021-04-01]. Dostupné z: https://elastalert.readthedocs.io/en/latest/ruletypes.html#alerts

[59] *Elastic Cloud: Enterprise search, observability, and security for the cloud* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/cloud/

[60] *Clastic Cloud on Kubernetes: Elasticsearch & Kibana on Kubernetes* [online]. Mountain View, CA: Elastic N.V., 2021 [cit. 2021-04-01]. Dostupné z: https://www.elastic.co/elastic-cloud-kubernetes

[61] **TURNBULL, James**. *The Docker Book* [online]. 2017 [cit. 2021-04-01]. Dostupné z: https://dockerbook.com/

[62] **KANE, Sean P. a Karl MATTHIAS**. *Docker: up and running. Second edition.* Sebastopol, CA: O'Reilly Media, [2018]. ISBN 978-1-492-03673-9.

[63] *Developer Survey Results 2020* [online]. 2020 [cit. 2021-04-01]. Dostupné z: https://insights.stackoverflow.com/survey/2020

[64] *Docker Documentation: Docker provides a way to run applications securely isolated in a container, packaged with all its dependencies and libraries.* [online]. Docker, 2020 [cit. 2021-04-01]. Dostupné z: https://docs.docker.com/

[65] *Docker Architecture* [online]. Tel Aviv-Yafo, Izrael: Aqua Security Software, 2021 [cit. 2021-04-01]. Dostupné z: https://www.aquasec.com/cloud-native-academy/docker-container/docker-architecture/

[66] *Borg: The Predecessor to Kubernetes* [online]. San Francisco, CA: The Linux Foundation, 2021 [cit. 2021-04-01]. Dostupné z: https://kubernetes.io/blog/2015/04/borg-predecessor-to-kubernetes/

[67] *Kubernetes Documentation* [online]. The Linux Foundation, 2020 [cit. 2021-04-01]. Dostupné z: https://kubernetes.io/docs/home/

[68] *Kubernetes Services* [online]. Palo Alto, CA: VMware, 2021 [cit. 2021-04-01]. Dostupné z: https://www.vmware.com/topics/glossary/content/kubernetes-services

[69] *Logeto - Attendance and Time tracking* [online]. Czech Republic: Systemart, 2021 [cit. 2021-04-01]. Dostupné z: https://www.logeto.com

[70] *Dokumentace Výkazu práce* [online]. Hradec Králové: Systemart, 2020 [cit. 2021-04-01]. Dostupné z: https://dokumentace.vykazprace.cz

[71] **NOVÁK, Tomáš**. *Automatické testování mobilních aplikací* [online]. Hradec Králové, 2019 [cit. 2021-04-02]. Dostupné z: https://theses.cz/id/sxui56/. Bakalářská práce. Univerzita Hradec Králové, Fakulta informatiky a managementu. Vedoucí práce doc. Mgr. Tomáš Kozel, Ph.D.

[72] *Redis* [online]. Redis Labs [cit. 2021-04-01]. Dostupné z: https://redis.io/

[73] *SQL Server technical documentation* [online]. Microsoft, 2020 [cit. 2021-04-01]. Dostupné z: https://docs.microsoft.com/en-us/sql/sql-server/?view=sqlserver-ver15

[74] *Jira* [online]. Atlassian, 2020 [cit. 2021-04-01]. Dostupné z: https://www.atlassian.com/cs/software/jira

[75] *TeamCity: the Hassle-Free CI and CD Server by JetBrains* [online]. JetBrains, 2020 [cit. 2021-04-01]. Dostupné z: https://www.jetbrains.com/teamcity/

[76] *Bitbucket* [online]. Atlassian, 2020 [cit. 2021-04-01]. Dostupné z: https://bitbucket.org/product/

[77] *Azure Event Hubs: A big data streaming platform and event ingestion service* [online]. Redmond, WA: Microsoft, 2021 [cit. 2021-04-01]. Dostupné z: https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-about

[78] *Praeco ElastAlert* [online]. San Francisco, CA: Praeco [cit. 2021-04-01]. Dostupné z: https://github.com/johnsusek/praeco

[79] *Legacy actionable message card reference* [online]. Redmond, WA: Microsoft [cit. 2021-04-01]. Dostupné z: https://docs.microsoft.com/en-us/outlook/actionable-messages/message-card-reference

[80] *Adding a New Alerter* [online]. San Francisco, CA: Yelp [cit. 2021-04-01]. Dostupné z: https://elastalert.readthedocs.io/en/latest/recipes/adding_alerts.html

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor: **Bc. Tomáš Novák**

Studium: I1900284

Studijní program: N1802 Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Název diplomové práce: **Logování, Monitoring a Alerting za použití ELK Stack**

Název diplomové práce AJ: Logging, Monitoring and Alerting with ELK Stack

Cíl, metody, literatura, předpoklady:

Cílem práce je zavedení logování, monitoring a alerting pro celé portfolio aplikací od společnosti Systemart s.r.o. za použití technologie ELK Stack a ElastAlert. V teoretické části budou popsány principy logování, monitoringu a alertingu, budou představeny teoretické koncepty, nástroje a zdůvodnění výběru konkrétní technologie/nástroje. V prvním úseku praktické části budou nastíněny konkrétní konfigurace využité ve společnosti Systemart s.r.o a vysvětleny principy jejich fungování a důvodu konkrétního nastavení. Druhý úsek praktické části bude věnovaná pouze nástroji ElastAlert a tvorbě Python modulu pro zasílání varovných správ a jejich stylování pro komunikační nástroj Microsoft Teams.

Struktura práce:

1. Úvod
2. Teoretický část
    1. Logování
    2. Monitoring
    3. Alerting
    4. Využití ve společnosti Systemart s.r.o
3. Praktická část
    1. ELK Stack
        1. Elasticsearch
        2. Logstash
        3. Beats
        4. Kibana
    2. ElastAlert
4. Závěr

1. CHHAJED, Saurabh. Learning elk stack. 2015. Mumbai: Packt, 2015. ISBN 978-1-78588-715-4.
2. SHARMA, Vishal. Beginning Elastic Stack. 2016. New Delhi: Apress, 2016. ISBN 978-1-4842-1694-1.
3. Elastic Stack and Product Documentation [online]. 2016. United States: Elasticsearch B.V., 2012 [cit. 2020-01-22]. Dostupné z: https://www.elastic.co/guide/index.html
4. ElastAlert: Easy {\&
Flexible Alerting With Elasticsearch [online]. 2016. United States: Yelp, 2015 [cit. 2020-01-22]. Dostupné z: https://elastalert.readthedocs.io/

Garantující pracoviště: Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu

Vedoucí práce: doc. RNDr. Petra Poulová, Ph.D.

Datum zadání závěrečné práce: 26.1.2021