Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Fakulta ekonomická
Katedra aplikované matematiky a informatiky

Bakalářská práce

Zpracování logů pomocí ELK stacku

Vypracoval: David Slavík
Vedoucí práce: doc. Ing. Ladislav Beránek CSc., MBA
České Budějovice 2023

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH

Ekonomická fakulta

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**David SLAVÍK**
Osobní číslo:	**E20035**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Zpracování logů pomocí ELK stacku**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Práce se bude zabývat zpracováním logů po teoretické i praktické stránce. Sběr logů bude probíhat za účelem auditování, monitoringu a dohledávání chyb. Logy, které budou sbírány z několika zdrojů napříč celou firmou mohou obsahovat důležité informace pro široké spektrum lidí. Pomocí centralizované správy logů pomocí ELK stacku se mohou k logům lidé dostat bez nutnosti přístupu přímo k dané aplikaci nebo serveru, kde aplikace běží. V této práci bude popsáno řešení od samotného sběru dat ze serverů a aplikací, jejich transformace, uložení až po samotnou práci s logy.
Cílem bakalářské práce je tedy navrhnout a popsat zpracování, správu logů pomocí zvoleného nástroje. Součástí práce bude popsání jednotlivých komponent aplikačního stacku, vazby a jejich praktické nasazení včetně konfigurace.

Metodický postup:

1. Studium odborné literatury.
2. Úvod do problematiky správy logů a obecný popis zvoleného řešení.
3. Teoretický popis jednotlivých komponent.
4. Popis implementace řešení.
5. Zhodnocení a možnosti alternativ.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Gormley, C., & Tong, Z. (2015). *Elasticsearch: the definitive guide.* O'Reilly.
2. Holubová, I., Kosek, J., Minařík, K., & Novák, D. (2015). *Big Data a NoSQL databáze.* Praha: Grada.
3. Richardson, L., Amundsen, M., & Ruby, S. (2013). *RESTful Web APIs.* O'Reilly.
4. Dokumentace <https://www.elastic.co/elasticsearch/>

Vedoucí bakalářské práce: **doc. Ing. Ladislav Beránek, CSc., MBA**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **11. ledna 2022**
Termín odevzdání bakalářské práce: **14. dubna 2023**

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Tomáš Mrkvička, Ph.D.
vedoucí katedry

V Českých Budějovicích dne 23. února 2022

Prohlašuji, že svou bakalářskou práci jsem vypracoval/a samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to – v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. Zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

Datum Podpis studenta

Abstrakt

Tato práce popisuje problémy s počítačovými logy a jak s nimi pracovat. Hlavním cílem této práce je ukázat, jak shromažďovat, transformovat, ukládat a zkoumat logy vytvořené aplikací nebo systémem. Představuje řešení pro správu logů s názvem ELK stack, které se skládá ze tří hlavních komponent: Elasticsearch, což je NoSQL databáze a vyhledávací engine, Kibana používaná jako grafické uživatelské rozhraní pro prohlížení, analýzu dat a také pro vytváření dashboardů. Logstash je součástí stacku, který transformuje a připravuje strukturu zpráv pro Elasticsearch. Další komponentou jsou beaty, které slouží ke sběru dat ze serverů. Součástí této práce je také návod na instalaci, konfiguraci a vytvoření pracovního prostoru s daty. Výsledek této práce by měl být užitečný pro společnosti, které se potýkají s logy, při implementaci vlastního řešení pro správu logů.

Abstract

This work describes problems with computer logs and how to treat them. The main goal of this work is to show how to collect, transform, save, and explore logs created by an application or system. It presents the log management solution named ELK stack that consists of three main components Elasticsearch which is a NoSQL database and search engine, Kibana used as a graphical user interface for viewing, analyzing data also as creating dashboards. Logstash is a part of the stack that transforms and prepares the message structure for the Elasticsearch. An additional component is a beat used for collecting the data from servers. Part of this work also includes instructions on how to install, configure and create a workspace with data. The result of this work should be helpful for companies struggling with logs to implement their own log management solution.

Key words: elasticsearch, kibana, logstash, beat, log, management

Obsah

1. Úvod

Log soubory jsou textové soubory obsahující záznamy o činnosti konkrétní IT aplikace. V případě webových serverů jsou do logů ukládány veškeré požadavky, které byly na server vzneseny. Tyto data se mohou hodit při zpětném zjišťování informací o tom, co se na serveru stalo. V rámci velké IT infrastruktury je časově náročné řešit logy individuálně na úrovni jednotlivých serverů. Pokud by správce systému byl nucen pracovat s těmito soubory jednotlivě, je pravděpodobné, že některé důležité informace přehlédne. Tyto informace mohou být důležité a někdy je potřeba provést na jejich základě adekvátní reakci. Přehlédnutí této informace může vést k problémům v chodu této aplikace, která může ovlivnit chod celé firmy. Se stále rostoucím počtem aplikací a serverů, je výhodné mít centralizované místo pro uchovávání a prezentaci log záznamů. Centralizované řešení by mělo obsahovat automatizované řešení pro ulehčení práce se správou logů.

1.1. Cíle práce

Tato bakalářská práce má za úkol představit komplexní řešení Log Managementu od společnosti Elasticsearch za použití ELK stacku. Toto řešení je vhodné pro snadné auditování a ke zpětnému trasování chyb, které se mohou v IT systému vyskytnout. Kvůli centralizaci logů a možnosti pracovat s daty pomocí jednoduchého rozhraní, je možné tato data zpřístupnit i kolegům s menšími technickými znalostmi.

V praktické části je cílem práce popsat komplexně celé řešení pomocí daného log managementu a co vše je možné tímto způsobem řešit. Zvolené řešení je detailně popsáno a obsahuje implementaci jednotlivých komponent včetně jejich instalace a konfigurace. Dále je zde uvedeno, jak efektivně pracovat v grafickém rozhraní a jak toto rozhraní využít pro analýzu dat. Na závěr je uvedeno zhodnocení včetně možných alternativních řešení.

2. Log management

2.1. Význam log managementu

V dnešním světě inteligentních zařízení, dochází ke generování mnoha záznamů, metrik a informací o zařízení a uživateli. Aby tyto data vedla k zpětné vazbě s kontinuálním rozvojem, je vhodné s nimi pracovat a postupně vyhodnocovat. Jelikož se v každém jednom případě jedná o formátem i typem jiné zprávy, je potřeba nástroje pro jednoduchou manipulaci s nimi a jednoduché zacházení.

Log management jako takový je ucelený soubor činností související se sběrem, transformací, zpracováním, ukládáním, archivováním a analýzou log zpráv. Obsahuje širokou škálu nástrojů pro práci s daty. Mezi tyto nástroje zejména patří možnost vizualizace dat, tvorba přehledů s grafy a tvorbu notifikací při námi definované situaci. Nově jsou též používány možnosti umělé inteligence pro detekci neobvyklých jevů např. při řešení bezpečnostních incidentů, kdy dochází k opakovanému pokusu o přihlášení i v nočních hodinách.

Pomocí těchto nástrojů je možné danou chybu zaznamenat, analyzovat a včas reagovat, dokud nedojde k vážnému problému, který by měl za vinu ohrožení chodu celé společnosti.
Mezi takové problémy můžeme řadit výskyt chyby v nepracovní dny, které by za normálních okolností mohly zůstat bez pozornosti, nebo upozornění před nezvykle častým přístupem k systému.
Další případ praktického použití je při již nastalých chyb. V tomto případě je možné nahlédnout do archivu a přesně definovat, kde se stala chyba a jakým způsobem lze ujednat nápravu. Namísto mnohdy pracného a časově náročného debuggování se tak jedná o jednoduché elegantní řešení.

Hlavní benefit je také pro management firmy, který nemusí být dostatečně technicky zdatný pro přístup k jednotlivým serverům, ale přesto má zájem o náhled do situace. Těm je možné zpřístupnit data pomocí grafů a přehledů v lidsky čitelné podobě.

2.2. Podstata logů

2.2.1. Z čeho se skládá log zpráva

Log je jednoduchá zpráva, která obsahuje strukturovaná data, kterou vytváří systém, aplikace podle daných pokynů/předpisů definovaných vývojáři. Logovaná data jsou primárně ukládána do souborů. Měla by nám říct za jakým účelem došlo k jejich zalogování. Dochází k nim zpravidla při změně stavu procesu, přihlášení se do aplikace či naražení na chybu v chodu aplikace.

Log jako takový se skládá z timestampu, log levelu, zdroje a dat.

V následujícím případě se jedná o událost, která je datována na 27 srpna 2022 v čase 21:22:15. Dále ze zprávy vyčteme, že se jedná pouze o informační zprávu ze zdroje, který má za úkol správu paměti. Samostatná data nás poté informují o tom, jaká část paměti je použita.

2022-08-27 21:22:15, Info CSetupFilesCleanup::GetSpaceUsed - Space used : 503476224

2.2.2. Základy logovaných dat

Log data jsou jednoduše log zprávy, logy. Log zpráva je to, co počítačový systém, zařízení nebo software generuje jako výstup na danou akci. Logovaná data pak tedy můžeme brát jako specifická data, která vyčítáme z log zprávy. Například web server vytvoří log pokaždé, když se uživatel přihlásí na webovou stránku. Tento log bude obsahovat například jméno uživatele, který se přihlásil na stránku, čas, kdy se přihlásil a zda bylo přihlášení úspěšné či nikoliv. Můžeme tedy použít jméno uživatele z logu pro jednoznačné určení osoby, která se přihlásila. Tímto získáme přehled o tom, kdo se kdy přihlásil a v případě problému ho můžeme kontaktovat.

2.3. Kdo s logy pracuje

Vývojář aplikací používá logy pro debuggování svého kódu a případně pro sledování chování aplikace.

Specialista na IT bezpečnost užívá logy v rámci kontroly zabezpečení interní sítě. Právě díky logům má přehled o tom, co se kde děje a je informovaný, pokud se objeví neobvyklé podezřelé chování.

Administrátor aplikace pomocí logu zajišťuje chod aplikace. To obsahuje veškeré sledování chodu včetně provozních událostí. Kontroluje, zda nedošlo k aplikační chybě, zda je vše v pořádku, zda není zatížen disk, či není jiný problém s hardwarem.

2.4. Druhy logů

V praxi rozeznáváme několik typů logů pro zpřehlednění určitých typů zpráv.

Základní rozdělení je level logu:

INFO – Používá se pro základní logování zpráv a stavů aplikace.

DEBUG – Užívaný v rámci hledání chyb, využívaný vývojáři. Obsahuje podrobnější informace než ostatní typy. Defaultně není zapnutý z důvodu vyšší produkce dat.

WARN – Poukazuje na možné problémové zprávy. Tato zpráva neohrožuje nutně chod aplikace.

ERROR – Označuje neobvyklé události, které by se neměly dít. Nemusí ohrozit chod dané aplikace, ale je vhodné těmto stavům věnovat určitou pozornost.

FATAL – Fatální pro chod aplikace. Značí situace, které jsou fatální pro aplikaci a ve většině případech dojde k zastavení aplikace. Oproti ERROR typu, kde je nefunkční pouze daná funkcionalita, zde nelze vykonávat žádnou aktivitu. [3]

V ideálním případě by mělo mít největší % zastoupení typ INFO.

Typy log zpráv:
Autentizace, Autorizace a přístupy (AAA – Authentication, Authorization, Access) – Úspěšné a neúspěšné pokusy o autentizaci nebo autorizaci. Obsahuje přístupy k serverům, aplikacím, databázím, vzdálené přístupy k datům.
Změnové události – systémové a aplikační změny. Jedná se například o datové změny včetně vytvoření, odstranění dat. Pak také instalaci a aktualizaci aplikací a komponent.
Chyby a hrozby – Špatně zadané vstupy a jiné ohrožující neobvyklé zprávy.
Problematika zdrojů – Aktivita zdrojů, například překročení určité meze paměti na disku, vyšší zatížení CPU, problémy s připojením.
Problémy kombinované dostupnosti (Mixed availability Issues) – Zprávy zaznamenávající spuštění a vypnutí aplikace, systému a jiné doplňující moduly. [3]

2.5. Knihovny pro zaznamenávání logů

Každý programovací jazyk/systém používá svoji knihovnu pro práci s log záznamy. Mezi nejpoužívanější jazyky patří stále Java, která nejčastěji používá knihovnu Log4j, či méně používanou Logback. Podobně jako jazyk Java je tomu i u jazyku Perl, C/C++ za použití Log4Perl případně tedy Log4C. Unixové systémy, Shell nejčastěji používá Logger. [3]

V následujících kapitolách se budeme věnovat převážně jazyku Java, a proto jsem se rozhodl popsat zde více knihovnu Log4j.

2.5.1. Knihovna Log4J

Jedná se o velice populární balíček, který slouží pro správu logů v aplikacích psaných za použití jazyku Java. Balíček je distribuován pod Apache Software licencí a je celý poskytován v rámci open source řešení. Přístup k log4j je zprovozněn pomocí rozhraní API, přes které prochází komunikace. [6]

2.6. Problematika a užití logů

Logy jsou široce podceňované v mnoha organizacích. Často se s logy nepracuje a připomínka o jejich výskytu se objeví až při zaplnění místa na disku. Logy ale mohou pomoct při řešení problémů a není správné s nimi nepracovat. Samotná analýza logů je náročná a chaotická a zabere nějaký čas. Je náročná z důvodu rozdílných formátů dat, velikostí dat. Pro efektivní práci s logy jsou také potřebné znalosti daného prostředí. Musíme znát co je správné a co je nesprávné, co je podezřelé a co je normální. Pro někoho již zkušeného se normální situace jeví jako běžná, kdy naopak stejná situace, může být pro neznalého člověka speciální událost, a to by mohlo při vyhodnocování

vést k nesprávným výstupům. Logy mohou být prospěšné ke zpětnému hledání příčin vzniklého incidentu. [3]

2.7. Náležitosti logů

2.7.1. Podoba kvalitního logu

Nestačí pouze události logovat, ale je potřeba stanovit si nějaká pravidla a dodržovat řád pro kvalitní užitečný log.

Taková zpráva by v nejlepším případě měla obsahovat podle pravidla 5 W:
What happened – Podrobně popsaný děj, stav, změna, ke které došlo
When did it happen – Časové razítko s vyvolanou akcí. Ideálně mít zalogovaný začátek i konec
Where did it happen – Na jaké stanici, serveru, případně síťovém rozhraní
Who was involved – Kdo byl dotknut
Where he, she or it came from – Odkud pochází

V některých případech pracujeme i s
How did it happen – Jak se to stalo
[3]

V návaznosti na log události, bude na správci daného prostoru, aby zhodnotil a případně podnikl další kroky. Mezi ně patří určit co daná událost ovlivní a co se bude dít dále, kde najdu více doplňujících informací pro zasazení události do kontextu. Stanovit postup co dělat v případě, že je další postup nežádoucí.

V reálném světě se všech 6 kritérií nejspíše nesetká. Ve velkém množství případů bude naše zpráva obsahovat pouze Co se stalo, kde se to stalo a kdy se to stalo.
Na zbylé otázky bude potřeba nalézt odpověď za pomocí dalších dodatečných informací.

V dnešní době je velmi důležité logovat významné situace, správně. Špatně nastavený log je také log, ale nic podstatného nám neřekne a jen zabírá místo.

2.7.2. Podoba špatného logu

Nevhodný log se vyznačuje tím, že mu chybí časové razítko a časové pásmo, bez tohoto je těžké určit kdy nastal a u vážnějších situacích nemůžeme jednoznačně určit co čemu předcházelo. Chybějící zdrojová/cílová IP nebo host adresa. Toto nemusí být u všech logů, ale u těch, které jsou orientované na komunikaci klient server je to potřebné. Nejednoznačný unikátní identifikátor zprávy. Při generování více zpráv napříč komunikací je vhodné zvolit identifikátor podle kterého lze spárovat ty zprávy, které spolu souvisí. To je zajištěno většinou pomocí generování ID přímo v aplikaci pomocí UUIDs – universal unique message identifiers. [3]

3. Elastic

Firma Elastic vznikla jako startup v roce 2000, kdy později její produkty hojně využívá řada společností, včetně těch nejznámějších jako je Netflix, Uber, Slack nebo Microsoft. Jejich řešení je dostupné jako open source produkt. Klíčový produkt, na který je navázaná veškerá infrastruktura se jmenuje Elasticsearch, je to vyhledávací a analytický engine postaven nad Apache Lucene. [5]

Apache Lucene je v současnosti dominantní nástroj pro fulltextové vyhledávání. Samotný Lucene je knihovna, která poskytuje rozhraní v jazyce Java. Pro jednodušší používání v ostatních programovacích jazycích se používají různé nadstavby a rozšíření jako již námi zmiňovaný Elasticsearch. [2]

V rámci jejich řešení nabízí využití pro mnoho okruhů. Od použití jako full textového vyhledavače s možnostmi scoringu při tvorbě e-shopů, SIEM modulu a IT bezpečnosti, tak i možnost Log managementu.

Nabízí se zde možnosti instalace v rámci vlastního prostředí v režimu on-premise, nebo možnost mít vše pod správou přímo od firmy elastic na jejich cloudovém prostředí v režimu elastic cloud. Ceny pro cloudové řešení se pohybují od 95$ za Standard verzi až po 175$ pokud se jedná o Enterprise verzi. Kromě těchto režimů jsou v nabídce dále Gold a Platinum. [5]

Využití cloudového řešení pak nabízí výhody bezstarostného provozu a automatickou aktualizaci verzí. Je však potřeba brát v potaz, že se jedná o placenou službu a nemusí se tak vždy vyplatit menším firmám. [5]

3.1. Licencování

Jak již bylo zmíněno výše, produkt je v základní verzi zcela zdarma. Jedná se tak o Basic licenci, která je platná neomezeně. Tato verze poskytuje základní set potřebný pro práci s daty. Za pokročilé možnosti a nadstavby, mezi které patří například napojení na LDAP, SSO či využití machine-learningu na detekci anomálií, alerting pomocí emailů je potřeba již zaplatit a povýšit základní licenci na vyšší stupeň dle konkrétní potřeby. Tyto funckionality jsou zpřístupněny až po zakoupení licence Platinum, Enterprise. Některé z těchto funkcionalit lze doplnit v rámci modulů třetích stran, které jsou vyvíjeny komunitou okolo produktu elasticsearch. Není však zajištěna funkčnost a udržitelnost do budoucna. Mezi známé doplňky patří alerting od ElastAlert [9] a LDAP připojení od OpenDistro [10]

4. ELK stack

ELK stack je akronym pro tři produkty Elasticsearch, Logstash, Kibana. Kdy Elasticsearch zastupuje vyhledávací a analytický engine, Logstash slouží pro zpracování dat na serveru, jejich transformaci a poté ukládání do Elasticsearche. Kibanu pak použijeme jako grafické rozhraní nad Elasticsearchem. V kibaně je možné prohledávat data, tvořit grafy a vizualizace, ale také spravovat a ovládat celý stack pomocí managementu. V našem případě budeme k těmto třem produktům používat také Beats. Beats zastupují potřebu získávání surových dat ze souborů. [5]

Pomocí takto komplexního řešení jsme schopni jednoduše a rychle zpracovávat velké počty dat a mít vše pod kontrolou.
Jednotlivé komponenty budou blíže popsány v následujících kapitolách.

4.1. High level řešení

Pro ideální případ se používá složení Elasticsearch, Kibana, Logstash a řada beatů. Lze používat i Logstash bez beatů, ale z důvodu vyšších hardwarových požadavků na logstash to není vhodné. Používá se tedy centrální logstash pro správu všech dat dodávaných z beatů, které jsou nasazeny na různé servery. Beaty dodávají surová data, a logstash je poté uzpůsobí daným potřebám, to znamená, že data obohatí či upraví. Data lze posílat také přímo z beatů do elasticsearche, přijdeme pak ovšem o možnost tato data upravovat a kontrolovat toky dat.
Pro produkční řešení je také vhodné využít clusterového řešení kdy se použije více nodů pro zaručení stále dostupnosti. Při implementaci clusterového řešení je však potřeba brát ohled na split-brain problém.

Pokud máme v jeden okamžik jeden master node a s ním také několik potenciálních master nodů, může nastat problém, kterému se říká split brain problém. Tento jev vzniká tehdy, pokud máme v danou dobu více než jeden master node v clusteru a tyto dva nody o sobě navzájem neví. Tato situace nastává, pokud tyto dva nody mezi sebou přeruší komunikaci, žádný jiný node v clusteru není a každý z těchto dvou nodů sám sebe prohlásí za master node. V tento moment máme cluster se dvěma nody a split brain situaci. Tato situace vede k tomu, že každý node si spravuje „svůj“ cluster a vede tak k nekonzistenci dat. Tomuto jevu se můžeme vyhnout nastavením minimálního počtu master nodů použitím tohoto vzorce. [12]

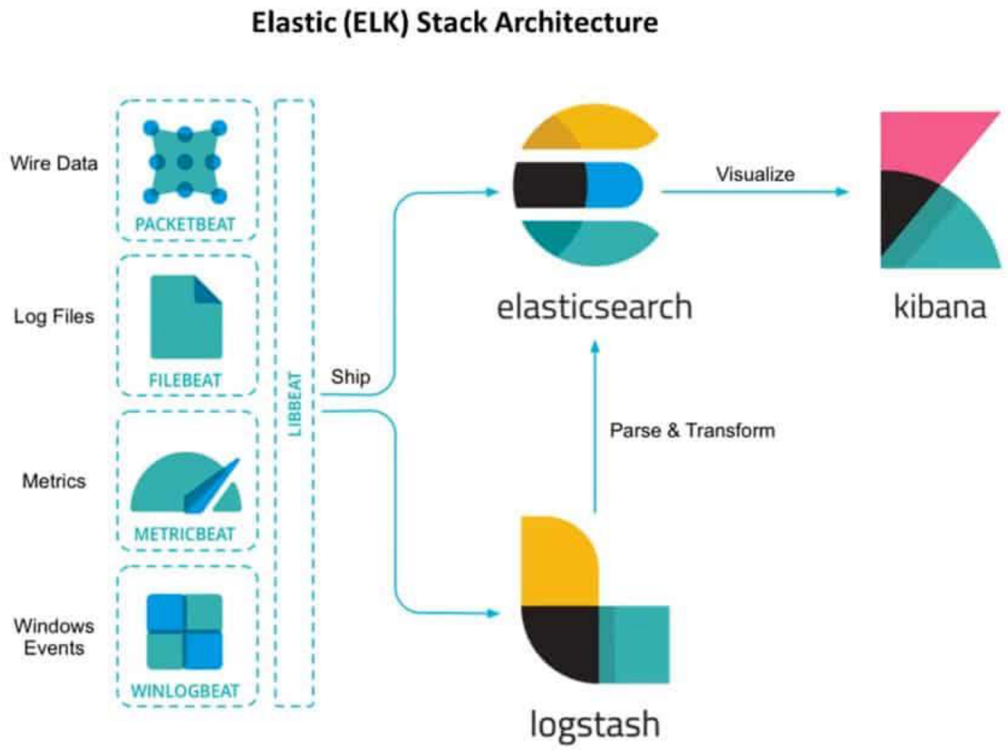
Minimální počet master nodů = (N/2) + 1

Kdy N je celkový počet možných master nodů v clusteru. Pokud tedy máme počet možných master nodů 3, tak minimální počet nodů bude 2. [12]

Představme si tedy, že máme 3 nody A, B a C. A je master node. Pokud z nějakého důvodu dojde k výpadku nodu A, node B a node C si mezi sebou vymění informaci o tom, že je node A nedostupný a určí si mezi sebou nový master node. Při zapojení nodu A zpět do clusteru, převezme tento node roli standardního nodu. To samé platí i pro

výpadek ostatních nodů. Vždy je zde více jak 50 % nodů, které se vzájemně ujistí o nedostupnosti nedostupného nodu a přizpůsobí se tomu.

4.1.1. High level pohled



Obrázek 1:Architektura ELK stacku [13]

4.1.2. Vysoce dostupný cluster

Pro dosažení stálé dostupnosti našeho řešení je potřeba navrhnout robustní architekturu odolnou vůči možným problémům, které by ovlivnily chod log managementu.
Pro použití v menších clusterech se jedná o pokrytí výpadku jednoho nodu z clusteru.
Při zasílání dotazů na dvou a více nodový cluster je vhodné vyspecifikovat všechny adresy nodů tak, aby v případě výpadku jednoho z nich nedošlo ke kompletnímu přerušení komunikace, ale bylo zajištěno pokračování komunikace na zbylé nody v clusteru. [5]

Menšími clustery rozumíme poté:

4.1.2.1. Jedno nodový cluster

Základní provedení, kdy jeden node provádí všechny operace. Toto řešení je náchylné na výpadek. Pokud tento node selže, dojde k zastavení chodu a je nutné problém vyřešit, než zase bude možné pokračovat. Toto nastavení není vhodné používat v produkčním prostředí. [5]

4.1.2.2. Dvou nodový cluster

Při práci se dvěma nody je doporučováno nastavit oba nody jako „data nodes“ a uchovávat každý shard na obou nodech pomocí nastavení repliky. Nastavením obou nodů jako data nodes je docíleno toho, že oba nody vykonávají stejné úkony a jsou defacto kopie. Pokud selže jeden node, druhý node může okamžitě převzat jeho roli a fungovat dočasně jako jedno nodový cluster. [5]

4.1.2.3. Tří nodový cluster

I u clusteru složeného ze tří nodů je doporučeno nastavit všechny nody jako „data nodes“ a u každého indexu ponechávat alespoň jednu repliku. Také je vhodné nastavit každý node jako „master-eligible“ tak aby vždy 2 nody si mohly zvolit svého mastera bez komunikace se třetím nodem. Takový cluster bude odolný proti ztrátě nodu Při výpadku jednoho nodu spolu dále komunikují zbylé dva nody, které si vzájemně potvrdí, že nemohou komunikovat na třetí chybový node a zvolí si tak mezi sebou nového mastera. Po připojení nefunkčního nodu dále pokračuje cluster jako tří nodový. [5]

4.1.2.4. Více nodový cluster

Při správě více nodového clusteru je možné jednotlivé nody specializovat na danou roli. Data node, Ingest node, machine learning node a jiné. Při více nodech je vhodné používat daný node pro danou funkci k využití maximálního užitku při vykonávání činností. Master nodes by neměly být škálovány a měl by být použit maximální počet 3 master nodů. Při více master nodů může docházet k zdlouhavému výběru master nodu. Také je vhodné zvolit některé z těchto nodů jako pevně dané tak aby nemusely být voleny. [5]

4.1.2.5. Vetší clustery

V rámci větších clusterů pracujeme s clustery rozdělenými do různých zón. Ve většině standardních clusterů je obvyklé, že nody v tomto clusteru sdílejí stejnou infrastrukturu a při ohrožení této infrastruktury je ohrožen také chod celého clusteru. Pro předejetí tohoto problému můžeme náš cluster rozdělit do několika zón, tím tak ochráníme celý cluster, pokud nastane výpadek jedné z lokalit. Je však důležité brát v potaz latenci a celkovou rychlost mezi těmito zónami. Jelikož v rámci práce s daty probíhá mnoho operací, tak není ideální mít velké prodlevy ve výměně dat. Můžeme použít model 2 zón s tiebreakerem, který se stará, aby nedošlo k problému split-brain mezi distribuovanými clustery nebo tři a více zónový clustery. [5]

Cluster bude odolný vůči ztrátě dat při výpadku jakékoliv zóny, pokud bude cluster health status jako zelený, jsou zde alespoň 2 zóny, které obsahují data nody. Každý index, který není searchable snapshot index má alespoň jednu repliku každého shardu. Cluster má alespoň tři master-eligible nody. Alespoň 2 z těchto nodů nejsou voting-only master-eligible-nodes a jsou rozprostřeny alespoň mezi 3 zóny. Je nakonfigurováno, že klienti posílají své požadavky do více než jedné zóny, nebo je nastaven load balancer, který rozhazuje zprávy mezi daný seznam nodů. [5]

4.2. Elasticsearch

4.2.1. Popis

Elasticsearch je hlavní komponenta celého stacku. Jedná se o fulltextový vyhledávací engine, NoSQL databázi. Používá se pro aplikační vyhledávání, logování, metriky, bezpečnostní a business analýzu. [5]

4.2.2. Základní pojmy

Index je synonymem pro databázi v relačně orientovaném světě databází. Jedná se tedy o soubor stejných dat seskupených do jedné množiny. Každý index je tvořen minimálně jedním shardem a žádným nebo několika repliky podle velikosti provozovaného řešení. [5]

Každý index obsahuje několik záznamů, kterým se říká dokumenty, jedná se o data ve formátu JSON.

Každý dokument obsahuje několik polí. Pole je dvojice klíč-hodnota zobrazující určité data.

Shard obsahuje určitou část dat v clusteru a používá se v rámci tzv. shardingu, který rozděluje data do několika částí skrze všechny nody napříč clusterem. Sharding je často spojován také s replikací dat. Replika je pak kopie shardů a slouží jako záloha, pokud ztratíme přístup k shardům. [2]

4.2.3. Správa dat a jejich úschova

Elasticsearch přistupuje k datům jako k jednotlivým dokumentům, které obsahují několik polí. Tyto dokumenty jsou pak v JSON formátu uloženy do indexů. Index je obdoba tabulky ve světě relačních databází a obsahuje dokumenty stejného typu.

Námi použité typy polí:
Binary – binární hodnota zakódována do Base64 stringu
Boolean – true/false
Keywords – obsahuje keyword, constant_keyword a wildcard. Je vhodný na použití opakujících se hodnot. Například log_level (INFO, WARN, ERROR, DEBUG). Keyword datové typy lze agregovat a řadit
Numbers – Číselný datový typ jako long a double
Dates – datumový datový typ obsahující date a date_nanos
Text – řetězec, obsahuje text a match_only_text, analyzovaný či nestrukturovaný text [5]

Elasticsearch podporuje i další typy polí, které však nejsou v rámci této práce použity, a tudíž nejsou ani zmíněny.

4.2.4. Záloha a obnovování dat

Pro zálohu elasticsearch clusteru používáme snapshoty, kterými poté můžeme obnovit chod do původní podoby. Snapshoty mohou být užitečné v případě, kdy dojde k potřebě již smazaných dat nebo dojde k poškození hardwaru, přenos dat mezi clustery, zredukování nákladů na úschovu dat, kdy použijeme searchable snapshots v cold nebo frozen fázi. [5]

Pro využívání snapshotů, je potřeba definovat repozitář, kam se budou tyto data ukládat. Repozitář je vhodné zvolit na jiném serveru než na tom, kde běží daný cluster, aby právě při poškození daného hardwaru bylo možné použít tuto zálohu. [5]

By default snapshoty obsahují clusterové nastavení, indexové templaty, legacy indexové templaty, ingest pipeliny, ILM policies v rámci stavu clusteru, všechny datové streamy, všechny regulérní indexy. Obsah snapshotů lze však modifikovat a ukládat pouze požadované položky. [5]

Toto je jediná podporovaná a ověřená záloha clusteru. Není možné zálohovat cluster vytvářením kopií adresářů daného nodu. Není podporována žádná metoda na filesystem úrovni. Při pokusu o obnovu pomocí takového řešení může dojít k selhání z důvodu nekonzistence nebo poškození dat. [5]

4.2.5. Rozdělení nodů

Master-eligible node – je zodpovědný za clusterové akce jako je vytváření a mazání indexů, hledání, které nody jsou součástí clusteru, rozhodování o tom, jaké shardy budou alokovány, na jakém nodu. Pro správně fungující cluster je důležité mít stabilní master node. Master node může být buď pevně daný node, což nám zaručí, že daný node bude sloužit pouze pro řízení clusteru a nebude zatěžován jinými aktivitami, nebo master node, který je možné zvolit.
Data node – drží shardy, které obsahujou dokumenty. Tyto nody se používají k vykonávání operací vázaných na data. Jedná se tak například o vykonávání CRUD operacích, vyhledávání, agregování. Data nody lze rozdělit podle životního cyklu dat na hot, warm, cold, frozen.
Ingest node – provádí předzpracování dat. Ve větším clusteru může být prospěšné mít tyto nody vyčleněné pouze na tuto činnost.
Remote-eligible node – tváří se jako cross-cluster klient a připojuje se k vzdáleným clusterům. Pokud je připojen, můžeme ho využít pro hledání ve vzdáleném clusteru pomocí cross-cluster vyhledávání, nebo také pro synchronizaci dat mezi clustery.
Machine learning node – obsluhuje API požadavky na strojové učení.
Transform node – obsluhuje transformační API požadavky. [5]

Každý data a master-eligible node požaduje přístup k adresáři, kde jsou shardy a indexy uloženy, zrovna tak jako metadata o clusteru. [5]

4.2.6. Správa indexů

Pro správu našich dat v indexů je možné použít ILM – Index lifecycle management neboli správu životního cyklu indexu. Použití je vhodné pro automatizovanou správu dat na základě vstupních požadavků na dostupnost, rychlost.

Definuje 5 fází, kterých může index nabývat. [5]
Hot – index je aktivně updatován a používán pro query dotazy
Warm – index už není updatován, má už stálá data, ale je stále používán pro query dotazy
Cold – index už není updatován a query dotazy jsou prováděny jen zřídka. Informace potřebují být stále vyhledatelná, ale uživatelé se smíří s nižším výkonem při práci s nimi
Frozen – index už není updatován a query dotazy se už téměř nepoužívají. Informace musí být stále dostupné, ale uživatelům nevadí si počkat při hledání v nich
Delete – index už není více potřeba a může být smazán

Přechod mezi jednotlivými fázemi se řídí podle „věku“ indexu. Pro přechod se základně používají dny od dosažení daného cyklu, ale může být využito i dalších pravidel, například dosažení zaplnění indexu na danou úroveň paměti. Před odstraněním indexu lze nastavit pravidlo, kdy dojde k uložení indexu jako snapshotu a až poté bude index odstraněn. To nám zajistí zálohu dat a možnost při problému obnovit daný snapshot. [5]

4.2.7. NoSQL vs SQL

Relační databáze	**NoSQL databáze**
Integrita dat je zásadní.	Stačí, pokud je většina dat většinu času v pořádku.
Datový formát je konzistentní a dobře definovaný.	Datový formát nemusí být známý nebo konzistentní.
Předpokládáme dlouhodobé uložení dat.	Vzhledem k velkému množství dat často ukládáme pouze určité „časové okno“.
Aktualizace dat jsou časté.	„Write-once/read-many“, tedy vložená data už typicky nejsou dále modifikována. Obvykle data neustále přibývají.
Předvídatelný nárust velikost dat.	Nepředvídatelný nárust velikosti dat.
Nástroje pro dotazování dat umožňují přístup i ne-programátorům.	Typicky pouze programátoři píšou zpracování dat.
Probíhají pravidelné zálohy dat.	Pro řešení výpadků je využívána replikace dat.
Přístup k datům zajišťuje jediný server.	Data jsou umístěna na více serverech, přistupujeme tedy ke clusteru uzlů (nodů).

Tabulka 1:Relační vs. NoSQL database – předpoklady o datech a aplikaci [2]

4.2.8. Rozhraní

Ke komunikaci s klienty používáme rozhraní pomocí REST API. Předávání pokynů podáváme pomocí metod GET, POST, PUT, DELETE zprávou v podobě JSON formátu.

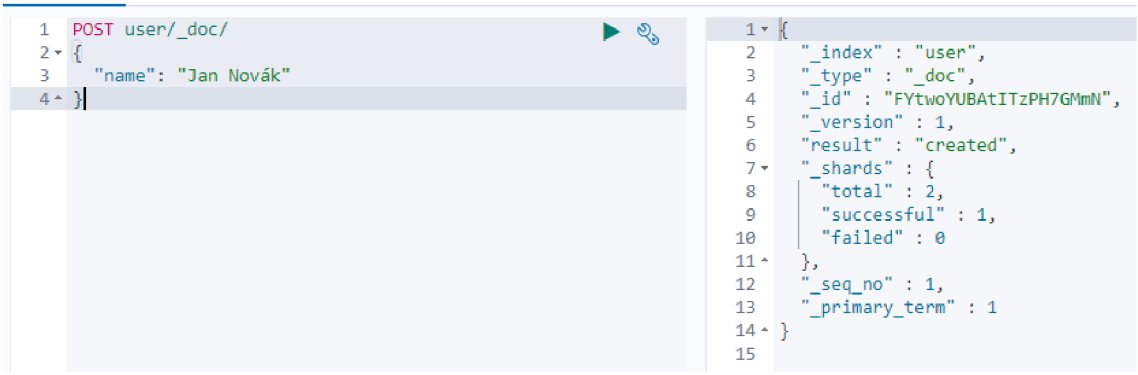
GET – metoda, kdy klient pošle dotaz po vzoru URL pro reprezentaci dat na tomto endpointu. Slouží jako doptání se po určitých datech. Tato metoda je označována jako bezpečná metoda, která neovlivní data na serveru. Lze pouze přetížit server častým dotazováním. Proti tomuto je však předpokládáno určité nastavení throttlingu, které tomu zabrání nastavením politik počtu volání z jednotlivé adresy. Nejčastěji tato metoda vrací návratový kód 200 – OK [1]

POST – vhodná metoda pro založení zdroje. Request pro tuto metodu obsahuje zprávu. Zpráva je nejčastěji ve tvaru JSON, XML těla. Návratový kód očekáváný na výstupu je 201 – Created. Po obdržení tohoto kódu je zdroj založen a je možné k němu přistupovat pomocí předchozí metody GET. [1]

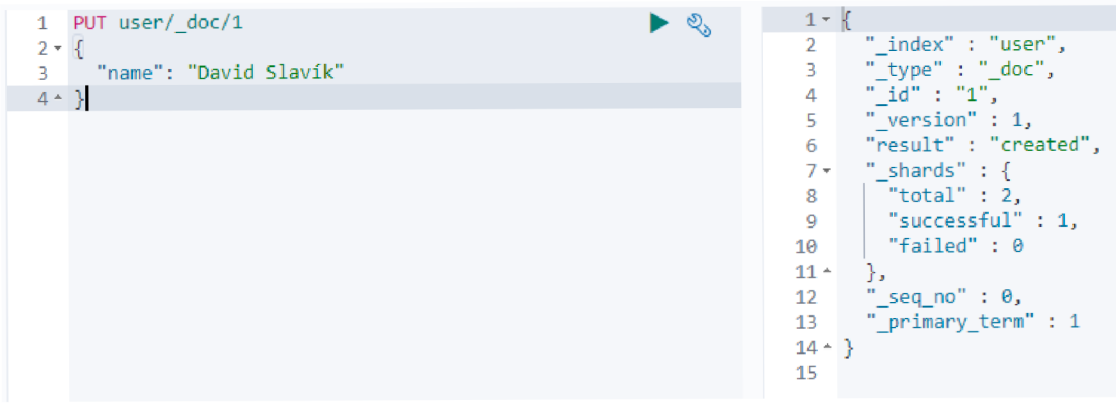
PUT – používá se při potřebě upravit již existující zdroj. Podobně jako POST musí obsahovat payload zprávu. Návratové kódy využívané u této metody jsou 200 – OK nebo 204 – No Content. [1]

DELETE – nebezpečná metoda, která nenávratně vymaže daný zdroj. Opakované volání metody DELETE nemá na zdroj žádný vliv, jelikož je už smazán. Této funkci, kdy nemá opakované volání metody vliv na stav zdroje se říká „Idempotence“. Je společná pro metody GET, POST, DELETE. Při úspěšném smazání zdroje vrací návratový kód 204 – No Content (zdroj byl smazán a není nic víc co se dá vrátit) nebo 200 – OK, či 202 – Accepted (vím o pokynu a provedu smazání až se na tento pokyn dostane řada) [1]

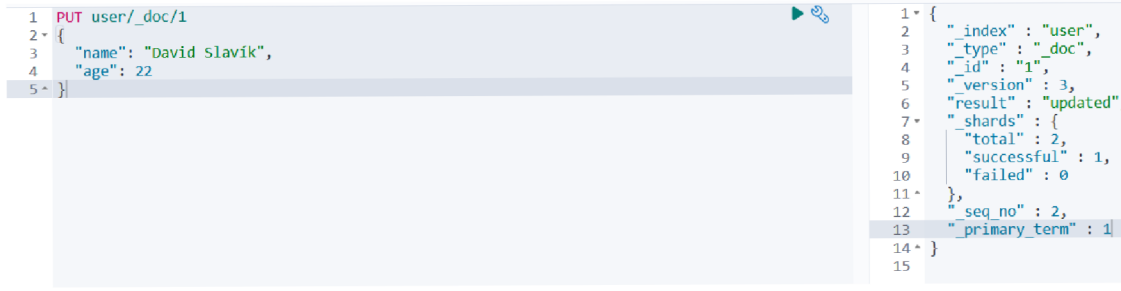
V následující části si ukážeme použití těchto metod na volání lokálního elasticsearche pomocí nástroje dev tools zakomponovaného v kibaně. Pro volání REST API můžeme také použít klientský nástroj zvaný Postman [11]



Obrázek 2:Založení dokumentu pomocí metody POST



Obrázek 3:Založení dokumentu pomocí metody PUT



Obrázek 4:Aktualizace dokumentu pomocí metody PUT

Obrázek 5:Získání dokumentů pomocí metody GET

Obrázek 6:Odstranění dokumentu pomocí metody DELETE

4.3. Logstash

4.3.1. Popis

Logstash je open source engine pro datové kolekce s možností zpracování toku dat v reálném čase. Je užíván k dynamickému unifikování dat z několika zdrojů a normalizování dat před uložením do cílových destinací. Transformuje naše data pro kvalitní využití analýz a vizualizací. [5]

Používá tzv. pipeline. Pipeline je termín užívaný pro zobrazení tok událostí skrze logstash. Pipeline se skládá ze vstupu, filtru a výstupu. [5]

Logstash je server-side řešení a je tudíž vhodné mít logstash pouze jako prostředníka pro zpracování dat nikoliv pro sběr surových dat ze serveru. Pro sběr surových dat ze serveru by nám měli sloužit beaty, které mají výrazně nižší požadavky na hardware. [5]

Pokud je potřeba pracovat s různými daty, lze použít multiple pipelines, tedy každá jedna flow bude mít svojí pipeline. [5]

4.3.2. Průchod událostí

Zprocesování událostí se skládá ze tří částí vstupy -> filtry -> výstupy. Na vstupu registrujeme nové události, pomocí filtrů je klasifikujeme, upravíme a výstupy je odešleme na požadovaný výstup. [5]

Mezi nejvíce používané vstupy patří:
Soubory – dokáže číst ze souboru podobně jako UNIX příkaz tail
Syslog – poslouchá na portu 514 pro syslog zprávy a parsuje je podle RFC3164 formátu
Redis – čte z redis serveru jak redis kanály, tak redis seznamy
Beats – zpracovává záznamy odeslané z beatů [5]

Filtry slouží pro zprocesování událostí. Je možné tyto filtry kombinovat.
Grok – parsuje a strukturuje text. Je nejlepší možností jak zestrukturalizovat data
Mutate – vykonává hlavní transformaci ohledně jednotlivých polí dané zprávy. Používá se pro přejmenování, odstranění nebo modifikace pole

Drop – zahazuje události, vhodné při debugování
Clone – vytváří kopii události
Geoip – přidá informaci o geografické lokaci IP adresy [5]

Výstupy jsou finální fáze v procesu přenosu událostí. Událost může projít několika výstupy, kdy v dokončení přenosu na poslední výstup dojde k ukončení zpracování události.
Elasticsearch – odešle události do elasticsearche, kde je s nimi dále možné pracovat v rámci stacku
File – zapíše data na soubor na disku
Graphite – odešle data na graphite, populární open source nástroj pro ukládání a zobrazení metrik
Statsd – odešle data do statsd [5]

4.3.3. Možnosti ochrany dat

Základně se používá udržování zpráv v paměti mezi jednotlivými fázemi. Díky tomu je průchod rychlý, jednodušší na správu. V případě nenadálého výpadku však přijdeme o data bez možnosti jejich opětovného zpracování. Dále má problémy při zpracování velkého počtu zpráv.
Pro zabránění ztrát událostí během jejich zpracování je možné využít perzistované fronty nebo „Dead letter queues“. [5]

Perzistence zprávy probíhá ještě před započetím filtrovací fáze, kdy dochází k jejímu uložení na disk. Zabrání se tak neobvyklému chování a předejde se ztrátě událostí.
V této frontě pak jsou zprávy drženy, dokud minimálně jednou úspěšně neprojde zpráva do cíle.
Tato metoda sama o sobě neřeší chyby jako poškození disku, chybu disku, jelikož data perzistována na disku nejsou replikována. Pokud se „checkpoint“ neprovede dříve, než dojde k nečekanému výpadku, je možné, že se data neuloží. Lze použít víceméně pouze pro HTTP protokol za použití beatů, kde se užívá response-request, na základě čehož je možné ověřit, zda došlo skutečně k úspěšnému odeslání zprávy do cíle. Protokoly jako tcp, udp nedisponují tímto mechanismem potvrzující odeslání. [5]

Dead Letter queues je navrženo pro dočasný zápis událostí, které neprojdou do cíle. Tyto zprávy pak tedy neblokují frontu a ostatní zprávy mohou dále procházet úspěšně. Zprávy uložené v DLQ však stále musí být manuálně prošetřeny a musíme docílit odeslání do cíle. DLQ je podporován pouze pro výstup Elasticsearch a je užíván pro dokumenty s návratovým kódem 400 a 404. [5]

4.4. Kibana

4.4.1. Popis

Komponenta slouží jako grafické rozhraní, které reprezentuje data z elasticsearche. Pomocí kibany získáme jednodušší přehled o svých datech a rychlejší administraci

napříč celým stackem. Poskytuje větší přehlednost nad jednotlivými daty a umožňuje vizualizaci a tvorbu grafů a diagramů. Grafy a diagramy lze skládat do objemných dashboardů, ze kterých lze poté generovat reporty například ve formátu csv.
Díky intuitivnímu rozhraní tak lze přidávat data, nastavovat přístupy a oprávnění, řízení pipeline, záloh a obnov. Další funkcí je zde zabudované vývojářské prostředí, ve kterém lze volat jednotlivá API oproti lokálnímu elasticu, testovat grok patterny, či testovat jednotlivé query dotazy nad daty na úrovni výkonosti daného dotazu. Hodnotí se zde jak dlouho daný dotaz trvá a jaký je výsledek. [5]

4.4.2. Spaces – rozdělení do skupin

Do centrálního stacku sbíráme data z mnoha zdrojů, které ale nelze poskytnout všem. Pro tento důvod se používá Spaces nebo také prostředí, které oddělují jednotlivé týmy od toho, jaké data mají přístupné a s čím mohou pracovat. Je tak možné veškerá data, která jsou v defaultním prostoru distribuovat do samotných prostorů dle práv. Jednotlivá data jsou distribuována pomocí uložených objektů z hlavního prostoru, tak aby nebyla narušena vazba mezi jednotlivými objekty. Dashboardy a query jsou vázané na indexy pomocí ID, a tak je priorita zachovat tuto vazbu. Prostředí lze modifikovat i na úrovni jaké sekce a podsekce z grafického zobrazení budou viditelné. Pro pouhé nahlížení do dashboardů a tvorbu analytiky nad daty nemusí uživatelé mít přístup do jiných sekcí a jejich prostředí je pak jednoduché, zaměřené jen na dané nástroje.

4.4.3. Vývojářské prostředí

Kibana sama o sobě též disponuje paletou vývojářských nástrojů k testování a optimalizaci řešení. Zde se pak nachází nástroje jako Konzole, která slouží jako klient k provolávání lokálních REST API daného elasticsearche. Vyhledávácí profil sloužící na analyzování naších query dotazů nad daty, kde je možné správně upravit dotaz pro co nejrychlejší dotaz. Grok debugger vhodný na otestování našich grok vzorů užívaných pro parsování dat na úrovni logstashe. Můžeme si poté být 100% jistý, že je daný grok napsaný správně a je podporován. [5]

4.5. Beats

4.5.1. Popis

Beaty jsou open source datový agenti, které nainstalujeme na server odkud chceme sbírat data. Tito agenti mají poté za úkol sběr dat ze zdroje a jejich doručení buď přímo do elasticsearche, nebo k dalšímu zpracování do logstashe. Všechny Beaty jsou založeny na libbeat frameworku.
Elastic základně poskytuje řadu beatů, které jsou aktualizovány. Obsahuje Auditbeat pro sběr auditních dat, Filebeat používaný pro čtení logů ze souborů, Functionbeat, který sbírá data z cloudu, Heartbeat užívaný pro kontrolu dostupnosti služeb, Metricbeat pro sběr metrik například i v rámci sběru metrik pro samotný ELK stack, Packetbeat pro vytíženost sítě, Winlogbeat vyčítací data z windows event logů. Z těchto výše uvedených beatů se blíže zaměříme a představíme si pouze filebeat, winlogbeat a metricbeat, které si následně ukážeme i v rámci praktické části.

Při řešení nestandardního použití lze použít komunitní beaty mezi které patří například amazonbeat, který čte data ze specifického Amazon produktu, dockbeat, který čte statistiky Docker kontejneru či twitterbeat, který vyčítá tweety. [5]

4.5.2. Filebeat

Filebeat se skládá z dvou hlavních komponent Inputs a Harvesters. Tyto dvě komponenty spolu spolupracují pří procházení souborů a odesílání dat na výstup. [5]

4.5.2.1. Harvester

Harvester je zodpovědný za vyčítání dat ze souboru. Čte každý soubor řádek po řádku a odesílá data na výstup. Pro každý soubor je jeden harvester. Pro sběr dat rozeznáváme na úrovni Filebeatu dva důležité parametry, kterými jsou „close_inactive“ a „scan_frequency“. Skenovací frekvence značí, jak často se bude kontrolovat přírůstek dat (defaultně 10 sekund). Časová frekvence je definována časem, kdy došlo k vyčtení poslední řádky ze souboru nikoliv zápisem systému do souboru. Pokud po určitou dobu nedojde k sběru dat dojde u tohoto souboru k uzavření pro neaktivitu. Uzavření pro neaktivitu dochází defaultně po 5 minutách. [5]

4.5.2.2. Inputs

Inputs je zodpovědný za správu harvesterů a hledání zdrojů odkud se bude vyčítat. Pokud je type „log“ pak input nalezne všechny soubory které jsou definovány. Pro každý soubor pak začne pracovat harvester. Každý vstup má svou vlastní Go rutinu. Filebeat podporuje řadu input typů jako Kafka, Syslog, TCP. Každý typ může být definován několikrát. Log vstup kontroluje každý soubor, aby věděl, zda má spustit harverster nebo zda už běží či zda má být soubor ignorován. Nové data jsou vyčteny, když se velikost souboru změnila od té doby, co byl harvester uzavřen. [5]

4.5.2.3. Udržování stavu

Udržování stavu probíhá opakovaným nahráním stavu na disk do registr souboru. Stav je použit pro poslední offset, který harvester četl a pro zajištění, že se logy odeslaly. Pokud je výstup elasticsearch nebo logstash nedostupný, dojde na straně filebeatu k udržení, co naposled bylo odesláno a pokračuje ve vyčítání po obnovení dostupnosti. Při běhu filebeatu je stav také udržován v paměti pro každý vstup. Při restartu služby se data nahrají z registry souboru pro obnovení předchozího stavu.
Pro každý vstup si filebeat drží stav ke každému souboru, který podle filteru nalezne. Z důvodu možné změny názvu souboru nebo změně cesty k souboru si filebeat sám ukládá unikátní identifikátory k odhalení jaký soubor už byl vyčten. [5]

4.5.2.4. Doručení zpráv

Filebeat garantuje doručení zpráv na zvolený výstup bez ztráty dat. Na základě registry souboru se udržuje stav každé události a při výpadku výstupu dochází k opětovnému provolávání výstupu až do té doby, než je výstup znovu dostupný pro přijímání dat. Při přerušení odesílání zpráv, filebeat nečeká na přijmutí všech událostí před ukončením filebeatu. Veškeré události, které byly poslány, ale nebyly potvrzeny budou odeslány

znovu po restartování služby, to zaručí, že každá správa bude odeslána minimálně jednou. Pro zamezení duplicit můžeme použít parametr „shutdown_timeout“, který bude čekat určitou dobu před vypnutím. V takovém momentě se bude po vypnutí čekat nastavený čas po který bude filebeat čekat na přijmutí odesílaných zpráv a poté se filebeat vypne. Defaultně je tento parametr deaktivován a není doporučeno nastavovat tento parametr z důvodu špatného odhadu času v rámci prostředí, nebo stavu zvoleného výstupu. [5]

4.5.3. Winlogbeat

Winlogbeat dodává log události na výstup podobně jako Filebeat. Tento beat lze nainstalovat jako Windows službu. Vyčítání probíhá pomocí Windows APIs. Pozice každého vyčítaného logu je ukládána na disku obdobně jako u filebeatu. Je to z toho důvodu, aby bylo možné při restartování této služby winlogbeatu možno opět pokračovat tam, kde výčet skončil před restartováním.

Winlogbeat dokáže zachytit data z jakýkoliv událostních logů běžícím na systému. Například aplikační události, hardware události, bezpečnostní události a systémové události. [5]

4.5.4. Metricbeat

Metricbeat jako předchozí beaty jsou jen odesílatelé dat a můžeme je nainstalovat na všechny servery, kde potřebujeme sbírat metriky z operačního systému a služeb běžících na serveru. Sbírá pomocí modulu metriky a statistiky například z Apache, HAProxy, MongoDB, MySQL. Je také použit při sledování stavu celého clusteru a sleduje tak i komponenty Elasticsearch, Kibana, Logstash, Beats. [5]

5. Alternativní řešení

Podle betterstack [14] se mezi alternativy k ELK stacku řadí například Logtail či Splunk. V rámci samotné Kibany se pak může jednat o konkurenci ve formě Grafany.

5.1. Logtail

Nabízí mnohem větší efektivitu zdrojů. Logtail umožňuje sběr logů napříč stackem, škálovat data. Je možné využít živé procházení logů a jejich analýzu. Mezi hlavní benefity autoři udávají počáteční cenu 0.25$/GB, použití ClickHouse uložiště. [14]

5.2. Splunk

Splunk je relativně nový a moderní log management a monitorovací nástroj. Poskytuje možnosti vyhledávání, filtrování, indexování a reportování. Dále nabízí uživatelsky přívětivé dashboardy. Mezi hlavní benefity se řadí flexibilní GUI, komplexní řešení vhodné pro podnikové použití a podpora mnoho funkcionalit jako jsou S3 zálohy, Heroku, Github. [14]

5.3. Grafana

Grafana stejně jako Kibana je nástroj pro vizualizaci dat. Grafana jako taková umožňuje číst data z mnoha zdrojů mezi něž patří také Elasticsearch. Na druhou stranu Kibana může číst data pouze z Elasticsearche jakožto součást ELK stacku. [15]

Parametr hodnocení	Grafana	Kibana
Instalace	Podporuje Linux, MacOS, Windows. Jednoduché na instalaci.	Podporuje Linux, MacOS, Windows. Jednoduché na instalaci.
Procesování dat	Je používána převážně na monitoring a metriky při porovnávání v čase. Analýza dat v reálném čase. Nabízí tedy funkcionality v rámci tohoto zaměření.	Nabízí velmi efektivní možnosti vyhledávání nad daty a jejich filtrování a vizualizaci. Hodí se spíše pro vizualizaci a prohlížení uložených dat například logů. Pracuje s širokou škálou zobrazení dat pomocí mnoho nástrojů.
Procházení dat	Pro procházení dat a query dotazy používá jazyk zvaný Grafana DSL. Tento jazyk je založen na PromQL (Prometheus Query Language).	Používá jazyk Elasticsearch Query DSL. Jeho nevýhodou je, že je kompatibilní pouze v ELK stacku.
Integrace datových zdrojů	Popdoruje databáze jako Prometheus, InfluxDB, Elasticsearch, SQL databáze, CSV soubory, AWS CloudWatch.	Podporuje pouze Elasticsearch.

UI a UX	Uživatelsky přívětivé prostředí, kde je možné vytvářet dashboardy, grafy a metriky. Spravovat konfiguraci.	Uživatelsky přívětivé prostředí, kde je možné vytvořit dashboardy, grafy. Lze zde také spravovat nastavení. Více uživatelsky přívětivé, jelikož umožňuje také drag and drop.
Vizualizace	Obsahuje mnoho různých typů vizualizace jako grafy, metriky a mapy.	Obsahuje mnoho různých typů vizualizace jako grafy, metriky, koláčové grafy, heat mapy, mapy, liniové grafy.
Alerting	Podporuje alerting a incident management.	Podporuje alerting a incident management.
Týmová spolupráce	Grafana umožňuje vytvářet uživatele, skupiny a jednoduše sdílet data v týmu. Je možné sledovat změny.	Kibana může pracovat s prostředím pro určité skupiny lidí, kde sdílí informace. Týmy si mohou nastavit alerty a notifikace pro sdílení informací o změnách. Kibana také nabízí nástroje jako je Slack pro komunikování v reálném čase.
Dokumentace a podpora	Velice přehledně a podrobně napsaná dokumentace. Větší podpora komunity.	Velice přehledně a podrobně napsaná dokumentace.
Licencování	Pro self-managed řešení se jedná o open source řešení zcela zdarma. Je možné provozovat základní cloudové řešení zcela zdarma. Poté jsou zde další 2 řešení Pro a Advanced. Ty nabízí funkcionality jako SSO, LDAP. Cloudové SLA a podporu. Reporty a exporty.	V rámci Basic licence je řešení zdarma. Toto řešení je nabízeno v self-managed režimu. Při potřebě více funkcionalit je možné povýšit licenci na placenou. Zde se cena odvíjí od potřeby funkcionalit. V rámci cloudového řešení spravované od Elasticu se též platí dle preferencí.

Tabulka 2:Porovnání Grafana a Kibana

6. Příprava clusteru

V rámci praktické části zde bude popsána konfigurace, instalace a práce s log managementem. Pro tyto účely bude představen pouze 1 nodový cluster, tudíž tento node bude zastupovat všechny funkce výše popsané. V produkčním nasazení však takové řešení není úplně vhodné a bylo by dobré volit například 3 nodový cluster.

Nejprve si všechny komponenty nakonfigurujeme a poté postupně spustíme a představíme. Při započetí sběru dat je vhodné nejprve spustit elasticsearch a připravit potřebné věci jako jsou například Index template, ILM policy, jednotlivé uživatelské účty a jejich role. Zde se řadí i separovaný účet pro logstash a kibanu, který bude s elasticsearchem komunikovat pro výměnu specifických informací pro fungování clusteru. Poté je možné spustit logstash a jednotlivé beaty, které budou již posílat data do elasticu.

6.1. Instalace clusteru

Při instalaci ELK stacku je vhodné připomenout, že by všechny komponenty napříč stackem měly mít stejnou verzi. V tomto případě je použita verze 7.17.5. Instalace by měla probíhat v následujícím pořadí.

1. Elasticsearch
2. Kibana
3. Logstash
4. Beats
5. APM
6. Elasticsearch Hadoop

[5]

Komponenty jsou dostupné pro platformy Windows, MacOS, Linux, RPM, DEB, Linux.

6.1.1. Elasticsearch

Po konfiguraci dané komponenty je potřeba si ověřit, zda je funkční a je možné na ni směřovat dotazy. To si můžeme jednoduše ověřit zavoláním REST API http://localhost:9200/_cluster/health?pretty

Po zavolání tohoto endpointu se nám vrátí data ve formátu.

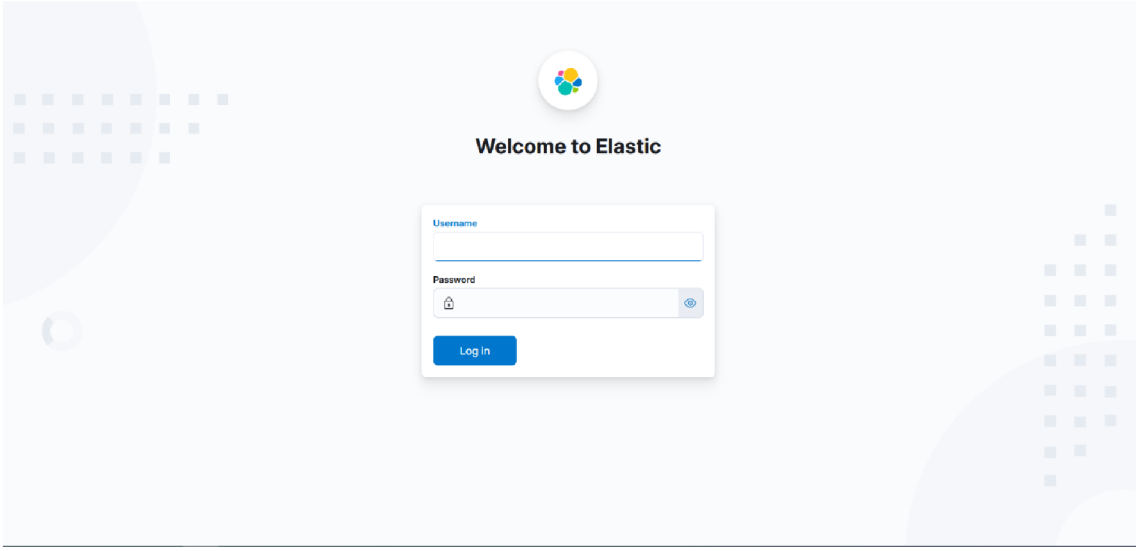
```
{
    "cluster_name" : "jcu",
    "status" : "green",
    "timed_out" : false,
    "number_of_nodes" : 1,
    "number_of_data_nodes" : 1,
    "active_primary_shards" : 28,
    "active_shards" : 28,
    "relocating_shards" : 0,
    "initializing_shards" : 0,
    "unassigned_shards" : 0,
    "delayed_unassigned_shards" : 0,
    "number_of_pending_tasks" : 0,
    "number_of_in_flight_fetch" : 0,
    "task_max_waiting_in_queue_millis" : 0,
    "active_shards_percent_as_number" : 100.0
}
```


Obrázek 7:Kontrola stavu elastic clusteru

Data nás informují o tom, o jaký cluster se jedná, jaký je jeho stav, počet nodů, shardů a probíhajících úkolů.
Nejdůležitější je pro nás kontrola, zda se náš cluster nachází ve stavu „green“. Pokud není ve stavu green je něco špatně a je potřeba to vyřešit. Existují zde další stavy „yellow“ a „red“, kdy ke stavu yellow může dojít například pokud nejsou dostupné všechny shardy/repliky. Red pak značí nefunkční cluster.

6.1.2. Kibana

Po konfiguraci Kibany máme možnost spravovat elasticsearch pomocí grafického rozhraní. Odtud můžeme jednoduše spravovat veškeré nastavení, prohlížet data. Defaultně je grafické rozhraní dostupné na adrese http://localhost:5601. Po zadání této adresy se dostaneme na přihlašovací stránku.



Obrázek 8:Přihlašovací stránka do Kibany

6.1.3. Logstash

Použití logstashe nám umožní transformovat data a přidělit je do daného indexu.
V rámci konfigurace tak musíme definovat na jakém portu nová pipeline naslouchá požadavky, určit grok pattern, který danou zprávu zachytí a případně provede úpravy na

zprávě. To znamená přidání nových polí, či odebrání stávajících a následně kam data zašle.

6.1.4. Beats

Nastavení jednotlivých Beatů pro sběr dat a posílání do Logstashe. V našem případě se jedná o Filebeat a Metricbeat. Filebeat bude sbírat data z log souboru aplikace a metricbeat sbírá metriky ELK stacku. Jedná se o správné nakonfigurování zdrojové cesty k souboru a poté správné cesty na Logstash, kde se budou data zpracovávat.

6.2. Sběr surových dat

Je zprostředkováno pomocí jednotlivých Beatů. Je potřeba definovat o jaké data se jedná, na úrovni beatů vymezit sběr těchto dat. Například je vhodné použít Multiline pattern, který bude rozdělovat log zprávy do dokumentů na základě tohoto patternu. Tím zapříčiníme toho, že se data sjednotí do jednoho dokumentu, ale bude každá zpráva brána jako jedna zpráva. Takto načtená zpráva bude odeslána do logstashe, kde se zpracuje na úrovni pipeline.

```
# ============================== Filebeat inputs ===============================

filebeat:
  config:
    inputs:
      enabled: true
      path: C:\Program Files\elastic\filebeat-7.17.5\inputs.d\*.yml



# ------------------------------ Logstash Output -------------------------------
output.logstash:
  hosts: ["localhost:5044"]
  tail_files: true
  tail: true
```

Obrázek 9: Konfigurace filebeat

Ve filebeat konfiguraci odkážeme na adresář, kde se nachází jednotlivé konfigurace pro výčet jednotlivých dat. Každá konfigurace odkazuje na 1 zdroj dat či aplikaci.

```
- type: log
  enabled: true
  scan_frequency: 10s
  multiline.pattern: '^\[[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2},[0-9]{3}\]'
  multiline.negate: True
  multiline.match: after
  fields_under_root: True
  paths: C:\Program Files\wso2ei-6.6.0\repository\logs\wso2carbon.log
  fields: {"event.type": "bezpecnost"}
```

Obrázek 10: Konfigurace sběru bezpečnostních dat

Zde se jedná o konfiguraci pro výčet dat ohledně bezpečnosti. V tomto případě používáme pole event.type podle kterého pak na úrovni logstashe identifikujeme o jaká

data se jedná. To je z toho důvodu, že pokud na jednom serveru vyčítá filebeat více souborů, je potřeba je rozdělit na této úrovni. Na straně logstashe pak můžeme filtrovat zdroje dat pomocí tohoto pole.

6.3. Zpracování pomocí pipeline

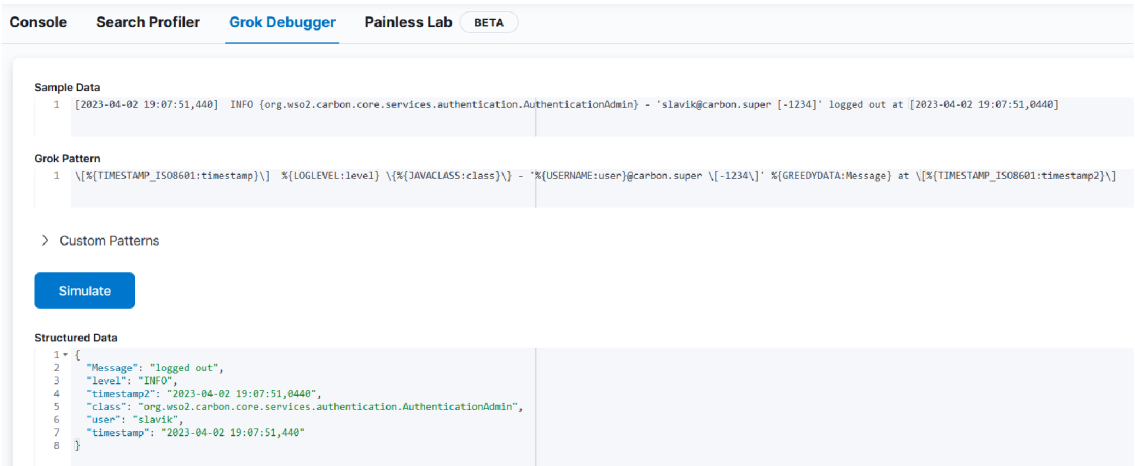
Jedná se o konfiguraci pipeline, která poslouchá na vstupu beaty na portu 5044. Jako výstup slouží lokální elasticsearch. Pomocí filtru identifikujeme data, které nás zajímají, na základě grok patternu rozparsujeme surová data do požadovaných polí a poté se data odešlou do elasticsearche. Výstup má dynamický název indexu, který se odvíjí od názvu pole event.type.

```
input {
  beats {
    port => 5044
  }
}

output {
  elasticsearch {
    hosts => [ "http://localhost:9200"]
    user => "logstash_push"
    password => "logstashpush"
    sniffing => true
    manage_template => false
    index => "jcu_%{[event][type]}-alias"
  }
}

filter {
  if [event][type] == "bezpecnost" {
    if "logged" in [message]{
    grok {
      match => [ "message", "\[%{TIMESTAMP_ISO8601:timestamp}\]  %{LOGLEVEL:level} \{%{JAVACLASS:class}\} - '%{USERNAME:user}@carbon.super \[-1234\]' %{GREEDYDATA:Message} at \[%{TIMESTAMP_ISO8601:timestamp2}\]"]
      }

      } else {
    grok {
      match => [ "message", "\[%{TIMESTAMP_ISO8601:timestamp}\]  %{LOGLEVEL:level} \{%{JAVACLASS:class}\} - %{GREEDYDATA:Message} '%{USERNAME:user}\[-1234\]' at \[%{TIMESTAMP_ISO8601:timestamp2}\]"]
        }
    }

    date {
      match => [ "timestamp", "ISO8601", "YYYY-MM-dd HH:mm:ss,SSSS" ]
      target => "timestamp"
      locale => "en"
    }

    mutate
    {
     remove_field => ["message", "timestamp2"]
    }
  }
}
```

Obrázek 11: Konfigurace Logstash pipeline

6.3.1. Tvorba grok patternu pomocí Dev tools

Při zpracování dat pomoci Logstashe potřebujeme definovat strukturu dat. Toho docílíme tak, že z přijímané zprávy vyparsujeme jednotlivé pole na základě grok patternu. Pro tvorbu tohoto vzorce můžeme použít naše vývojářské prostředí zabudované v Kibaně.

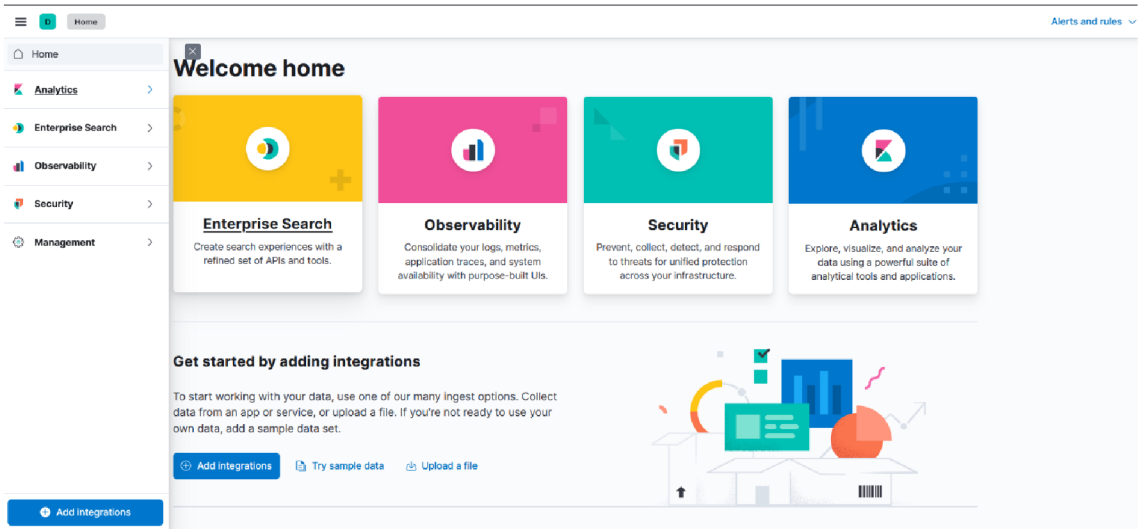


Obrázek 12:Tvorba grok pattern

Jak můžeme vidět jako první máme vzorové data nad kterými chceme vytvořit grok pattern. Pod daty máme námi vytvořený grok pattern pro rozdělení celé zprávy na několik polí, podle kterých bude později možné filtrovat. Nově vytvořené pole vidíme dole pod strukturovanými daty. Takto budou finálně vypadat pole našeho indexu. V tomto případě se jedná o data ohledně bezpečnosti zaznamenávající přihlášení jednotlivých uživatelů do aplikace.

7. Grafické rozhraní

Pomocí grafického rozhraní je možné spravovat celý cluster, a to jednoduše bez nutnosti použití API volání napřímo. V liště se nachází 5 sekcí, Analytics, Enterprise Search, Observability, Security a Management. V každé sekci máme paletu nástrojů, se kterými je možné pracovat.



Obrázek 13: Úvodní stránka pro grafické rozhraní nad Elasticsearchem Kibana

7.1.1. Analytics

V analytické části Kibany se nachází celý přehled dat, možnost procházení dat a jejich filtrování. Nedílnou součástí analytiky jsou také dashboardy a vizualizace. Jednotlivé vizualizace je možné skládat do rozvinutých dashboardů znázorňující danou problematiku. Tato část slouží pro uživatele k procházení jimi používaných dat. Zde lze také vytvářet geografické mapy podle polohy zprávy. Je tedy možné sledovat informace i dle lokality.

7.1.2. Enterprise Search

Slouží pro vývojáře a vývojářské týmy, kteří mají zájem o to vytvořit aplikaci s vyhledáváním pomocí elasticsearche.

7.1.3. Observability

Obsahuje nástroje pro alerty, APM, monitoring. Zde můžeme pracovat s funkcemi na pozorování stavu komponent. Sekce observability nám umožňuje přidat a monitorovat logy, systémové metriky, data chodu aplikace. Získáme tím centrální místo pro přidávání a konfiguraci datových zdrojů.

7.1.4. Security

ELK stack nemusí obsahovat pouze správu logů, ale nabízí také použití v rámci kybernetické bezpečnosti a monitoringu nad servery. Tato část tak představuje přehled nad zařízeními napojené v rámci bezpečnosti. Pomocí jednoho z dalších beatů, konkrétně Heartbeatu, lze sledovat aktivitu jednotlivých endpointů a zaznamenávat tak

jejich stav. Je možné nastavit pravidla nad logy a při abnormálním chování upozornit privilegované osoby.

7.1.5. Management

Sekce pro správce a vývojáře ELK stacku. V této části se nachází již zmiňované vývojářské prostředí, kde můžeme provozovat jednotlivé REST API volání na lokálním nodu, či vymýšlet grok patterny. Dále se zde nachází celková správa stacku, která obsahuje správu uživatelů, indexů, rolí, uložené objekty, snapshoty a jejich obnovu, ILP, alerting. Monitoring celého stacku s metrikami o počtu indexů, využití paměti a status jednotlivých komponent stacku a administrační vlákno sloužící pro vytváření uživatelů, nastavování práv,
tvorbu index patternů a přehled uložených objektů do kterých patří jednotlivé indexy, query dotazy, vizualizace, dashboardy. Tyto jednotlivé uložené objekty lze poté sdílet mezi jednotlivá prostředí napříč stackem pro zviditelnění určitým lidem.

7.2. Příprava pro příjem dat

Před samotným přijímáním dat je vhodné správně navrhnout strukturu zprávy a vymyslet, jak budeme k indexům přistupovat. Pokud budeme chtít využít všechny funkcionality jako jsou ILM tedy automatizovanou správu indexů, je vhodné použít automatické rollování indexů, které bylo vysvětleno v teoretické části. To zajistíme tak, že pro index vytvoříme alias a data budou posílány na tento alias nikoliv na jednotlivé indexy. Ty se totiž budou v čase měnit a škálovat.

7.2.1. Vytvoření aliasu

Pomocí následujícího REST API volání vytvoříme první index jcu_bezpecnost-000001 s aliasem jcu_bezpecnost-alias. Nová data budeme zasílat na tento alias jcu_bezpecnost-alias. Alias indexů jcu_bezpecnost si pak sám hlídá jaký index je zrovna aktuální a do jakého indexu budou nová data ukládána. To nám zaručí bezpracný chod clusteru, v případě, že jsou všechny fáze životního cyklu optimálně nastaveny.

```
PUT jcu_bezpecnost-000001
{
  "settings": {
    "index.lifecycle.name": "jcu_bezpecnost_policy",
    "index.lifecycle.rollover_alias": "jcu_bezpecnost-alias"
  },
  "aliases": {
    "jcu_bezpecnost-alias": {
      "is_write_index": true
    }
  }
}
```

Obrázek 14: Vytvoření aliasu

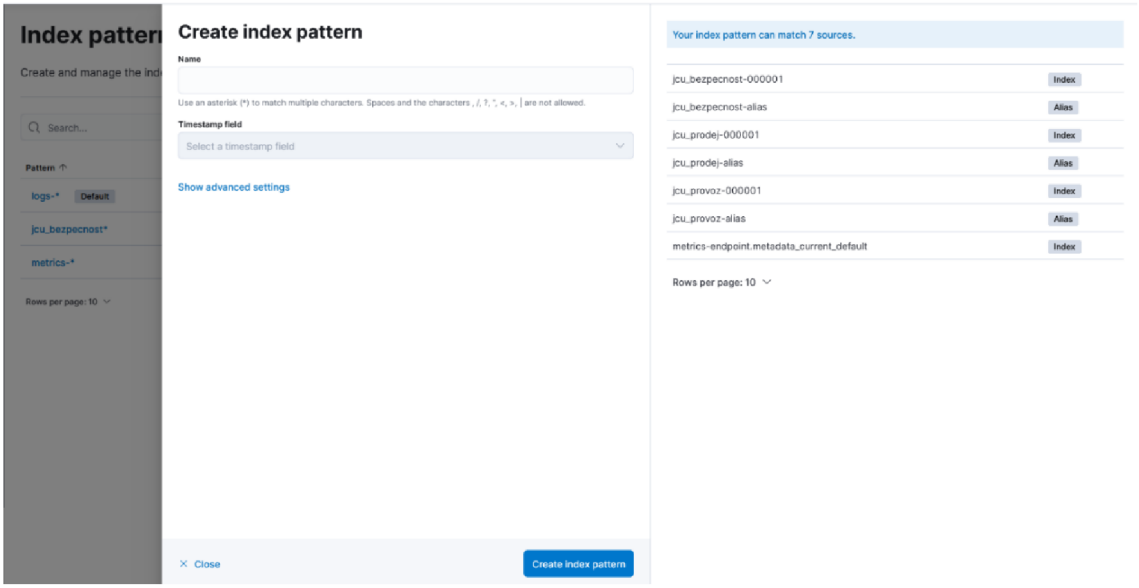
7.2.2. Vytvoření index template a index patternu

Tímto voláním vytvoříme index template, který se nám stará o nastavení shardů a replik. Dále definujeme mapování atributů a přiřadíme do ILM policy a určíme alias pro rollování jednotlivých indexů.

```
PUT _index_template/jcu_bezpecnost_template
{
  "template": {
    "settings": {
      "index": {
        "lifecycle": {
          "name": "jcu_bezpecnost_policy",
          "rollover_alias": "jcu_bezpecnost-alias"
        },
        "number_of_shards": "1",
        "number_of_replicas": "0"
      }
    },
    "mappings": {
      "properties": {
        "level": {
          "type": "keyword"
        },
        "message": {
          "type": "text"
        },
        "class": {
          "type": "keyword"
        },
        "user": {
          "type": "keyword"
        },
        "timestamp": {
          "type": "date"
        }
      }
    }
  },
  "index_patterns": [
    "jcu_bezpecnost-*"
  ]
}
```

Obrázek 15: Vytvoření index template

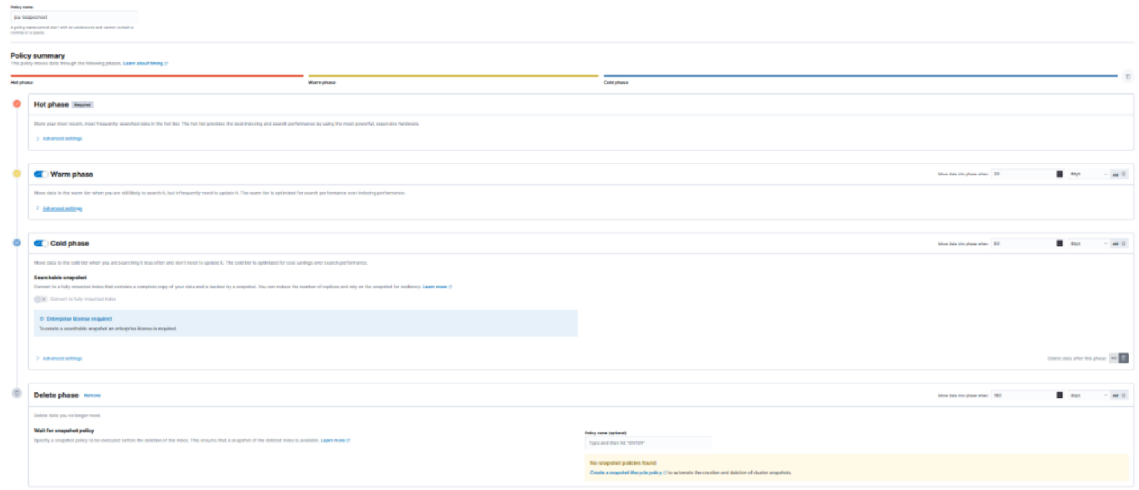
Vytvoření index patternu je možné až po vytvoření indexu. Tento pattern mapuje veškeré indexy, které definujeme. V tomto stacku je například použit index pattern jcu_bezpecnost-*, pomocí kterého zachytíme všechny indexy začínající právě názvem jcu_bezpecnost-. Pomocí těchto patternů poté můžeme vyhledávat mezi tímto typem indexů a použijeme je například při analýze dat.



Obrázek 16: Správa index patternů

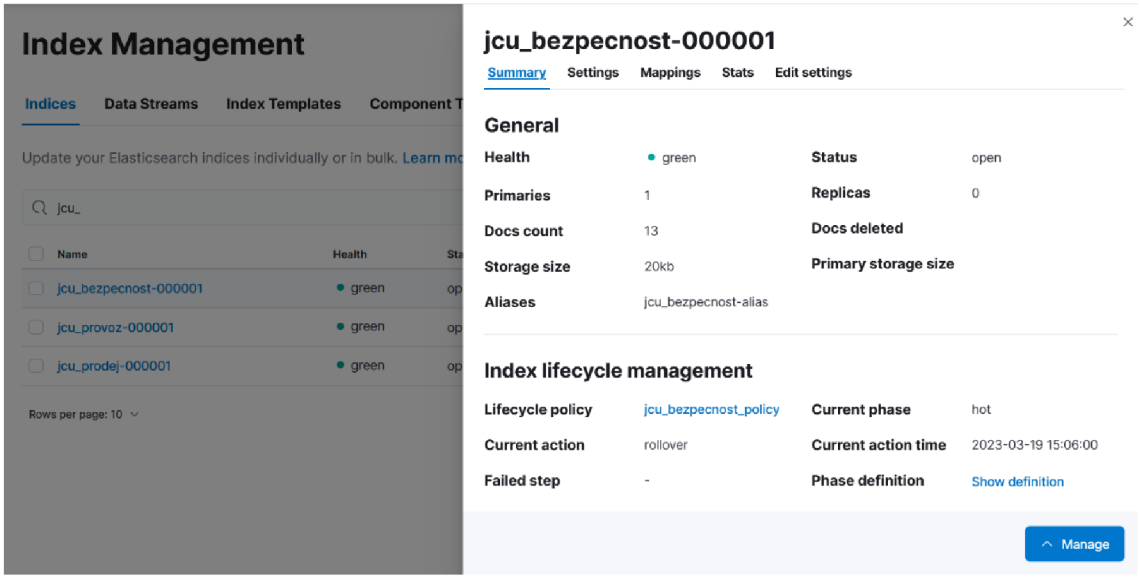
7.2.3. Vytvoření ILM

Pro automatizovaný chod stacku je důležité automatizovaný koloběh indexů. To zařídíme pomocí index lifecycle managementu, který se nám stará o životní cyklus indexu. Nadefinujeme pro náš index pattern jcu_bezpecnost nastavení, které se skládá ze 4 fází. Hot fáze je první z fází, do které se index zařadí vždy po založení nového indexu. V této fázi bude daný index buď po dobu 30dnů nebo do dovršení 50 GB. Po splnění jednoho z těchto požadavků se přesune index do další fáze Warm, kde setrvá dalších 30 dnů. Souběžně s tím se nová data zařazují již do nového indexu. Standardně se při této operaci přesune index s příponou 000001 do této fáze a vytvoří se nový index v hot fázi s příponou 000002. Toto vše je zajištěno pomocí aliasu. Po Warm fázi se přesune index do Cold fáze, kdy po určité době dochází k přesunu do Delete fáze a odtud je následně smazán. Pokud máme zájem o uchování těchto dat pro účely auditování, můžeme nastavit tvorbu snapshotu před odstraněním. Před odstraněním tak dojde k vytvoření snapshotu, který je možné kdykoliv obnovit.



Obrázek 17: Definování Index Lifecycle Policy

7.3. Správa indexů

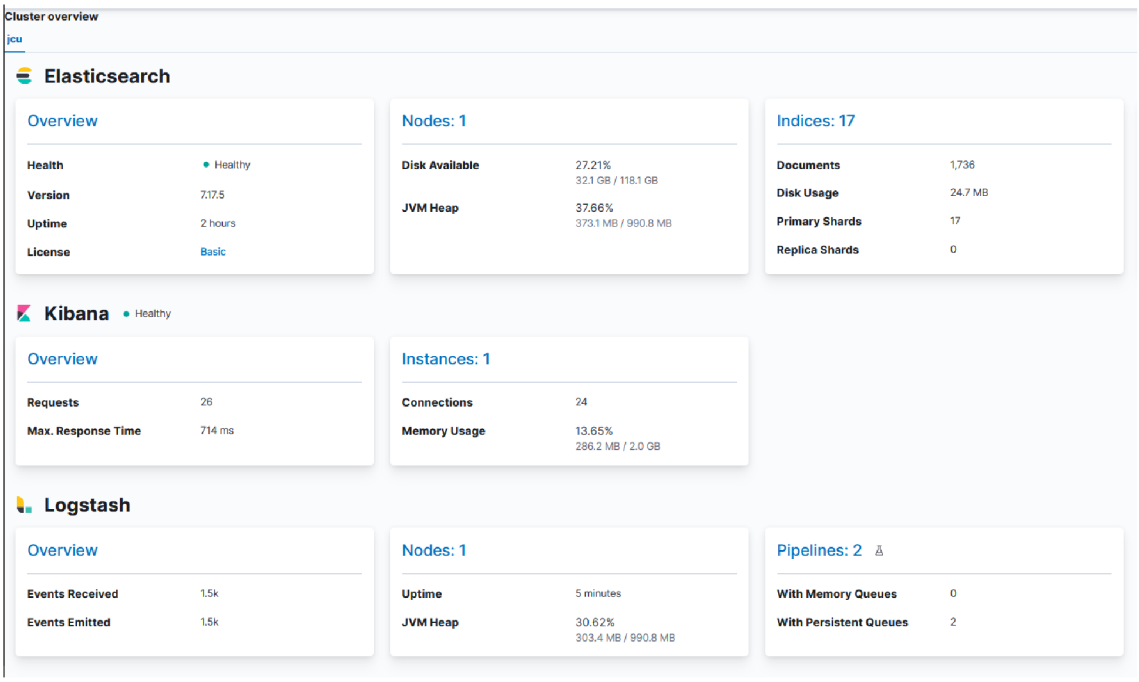


Obrázek 18: Index management

Na pozadí můžeme vidět všechny indexy začínající na jcu_. Jedná se o námi definované indexy pro bezpečnost, provoz a prodej. Každý z těchto indexů je první index v rámci rollování a má tedy příponu 000001. Po zobrazení více dat zobrazené na pravé straně vidíme stav indexu, počet shardů a replik a počet dokumentů v indexu. Dále máme přehled o jeho životním cyklu a kdy se do daného cyklu dostal. Pokud potřebujeme upravit životní cyklus z důvodu rychlého hromadění dat, je možné ihned pomocí odkazu na policu a to jcu_bezpecnost_policy. Zde je možné upravit nastavení pro lepší optimalizaci.

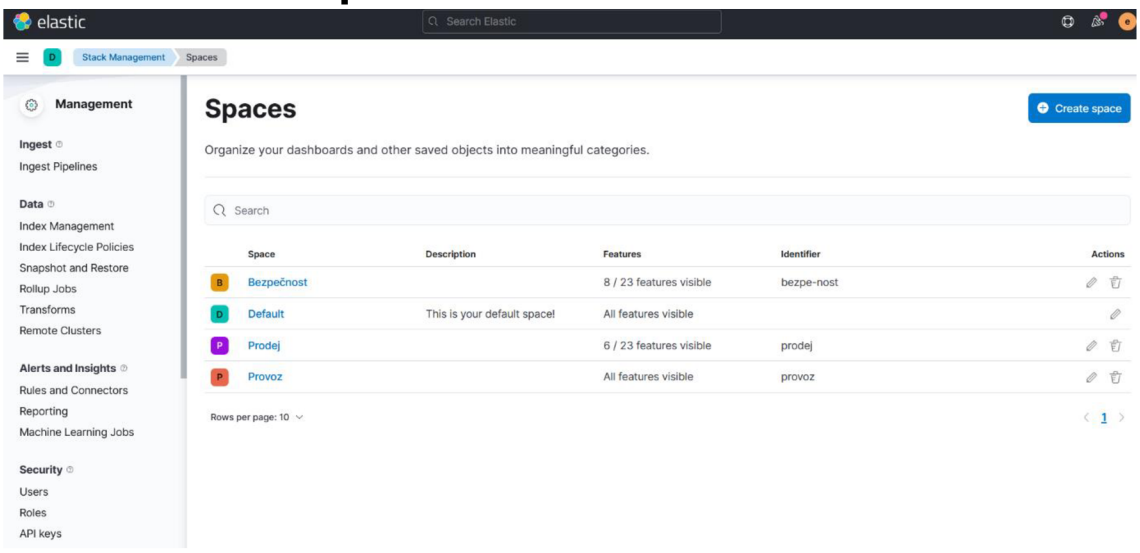
7.4. Cluster overview

Po konfiguraci metricbeatu, kdy jsme nastavili sbírání metrik našeho clusteru, a to jmenovitě komponent Elasticsearch, Kibana a Logstash lze sledovat jejich stav.
V kibaně je přístupný přehled těchto komponent v rámci Stack monitoringu. Na této úrovni máme přehled o stavu clusteru, verzi, počtu indexů (včetně systémových), počtu nodů, vytížení, kapacitu disku. Na úrovni Logstashe pak vidíme počet příchozích zpráv. Dále zde vidíme všechny pipeliny i s informací o jaký typ se jedná.



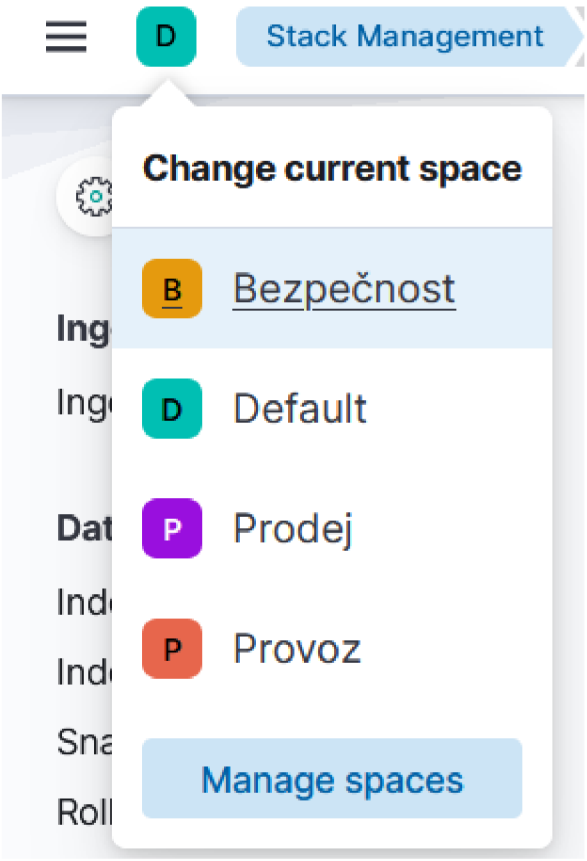
Obrázek 19:Cluster overview

7.5. Rozdělení prostředí



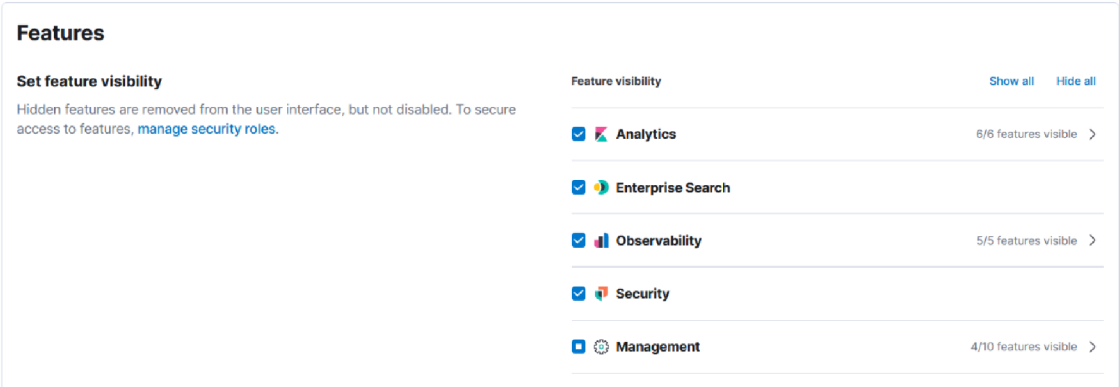
Obrázek 20:Správa prostředí

Při správě velkého množství dat je vhodné si celý cluster rozdělit do několika oblastí dle logických celků. V tomto případě máme data rozdělena do 3 prostředí dle zaměření jednotlivých oddělení ve firmě, a to jsou Bezpečnost, Prodej, Provoz. Default space slouží jako hlavní prostředí pro správce ELK stacku. Z tohoto základního prostředí můžeme distribuovat uložené objekty jako pohledy, dashboardy, indexy do těchto prostředí. Tímto zajistíme, že má privilegovaná osoba přístup pouze ke svým datům.



Obrázek 21:Přepínání prostředí

V případě přístupu do více než jednoho prostředí lze mezi těmito prostředími jednoduše přepínat a pracovat s daty v daném prostředí.



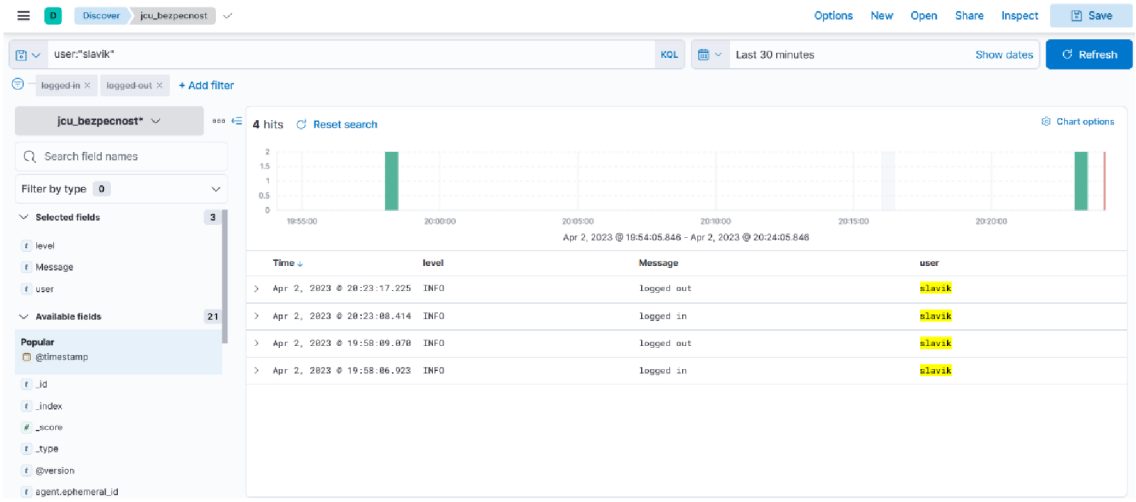
Obrázek 22:Nastavení funkcionalit v prostředí

V daném prostředí vydefinujeme uživatele, jejich role a také možnosti přístupu k jednotlivým funkcionalitám. Pro prostředí provozu tedy dává smysl, aby měli přístup k analytice, vyhledávání, metrikám, bezpečnosti i k určité části managementu. Toto se však netýká skupiny prodej, které stačí pouze přístup k datům, a proto má povolené pouze analytiku. Na uživatelském rozhraní se pro dané prostředí nachází pouze nastavené sekce. To zajistí větší přehlednost při práci pouze z daty.

8. Analýza dat

Analýza logovaných dat je velmi důležitá část celé práce s logy a taky důvod proč zvolit řešení jako ELK stack. Pomocí nástrojů na analýzu můžeme rychle a efektivně procházet tato data, které máme vhodně připravena k procházení a analyzování. Je možné hledat příčiny problémů chyb a případné dohledání problému v čase. To se může hodit při zkoumání problému. V rámci analýzy můžeme procházet data, vytvářet přehledy a ukládat do CSV ve formě reportů.

8.1. Vyhledávání dokumentů



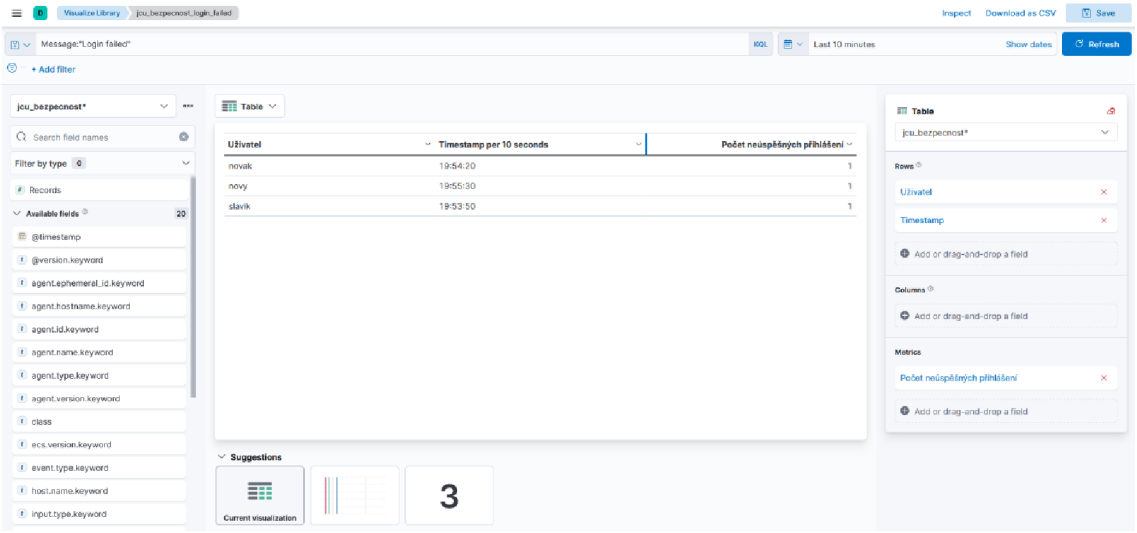
Obrázek 23: Vyhledávání dokumentů v indexu

Takto vypadá plocha pro hledání v indexu. V levém sloupci máme přehled všech polí v indexu, které můžeme zobrazit na hlavní ploše. Mimo pole vytvořené parsováním zprávy na úrovni logstashe, se zde nachází i defaultní pole vytvořené filebeatem. Pro tento účel zobrazení dat jsou použity pouze pole level, Message a user. Nad výběrem pole se nachází výběr, z jakého index patternu čerpáme data. Index pattern je vzor pro indexy. To znamená, že když máme 2 indexy jcu_bezpecnost-000001, který je již ve fázi warm či cold a nový index jcu_bezpecnost-000002, probíhá vyhledávání nad oběma těmito indexy. V horní levé nabídce dále nalezneme řádek pro zadávání query dotazů nad daty a možnost přidat stálé filtry. Momentálně vidíme, v jakých časech se uživatel slavik přihlásil a odhlásil za posledních 30minut. V případě zjištění jen přihlášení je možné si zapnout filtr logged in. Tento pohled je možné si uložit a ulehčí nám tak definování nových podmínek. To samé platí i pro definici query dotazů, které lze také ukládat.

8.2. Tvorba vizualizací

Tvorba vizualizací lze tvořit pomocí metody drag and drop. Při návrhu tedy zvolíme pole na základě, kterých chceme dělat vizualizace a poté je přesuneme na pracovní plochu. Poté máme několik návrhů na zobrazení dat. Stejně jako u vyhledávání platí i zde, že je možné zakomponovat filtry a query dotazy. Vizualizace lze ukládat do CSV souboru. Je možné pracovat i v tomto formátu pro jiné účely. Pro tvorbu vizualizací nelze použít textové pole, jelikož nejsou agregované. V našem případě jsme použili pole

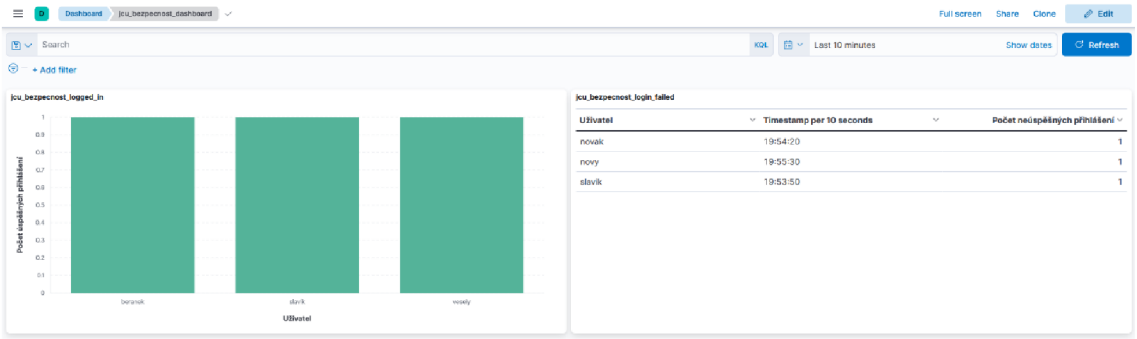
user datového typu keyword a pro hledání jsme si pomohli přidáním query dotazu, který v poli Message hledá text „Login failed“. Finální vizualizace nám pak předává stručný přehled o tom, kdo měl za posledních 10 minut problém s přihlášením se do aplikace.



Obrázek 24: Tvorba vizualizace

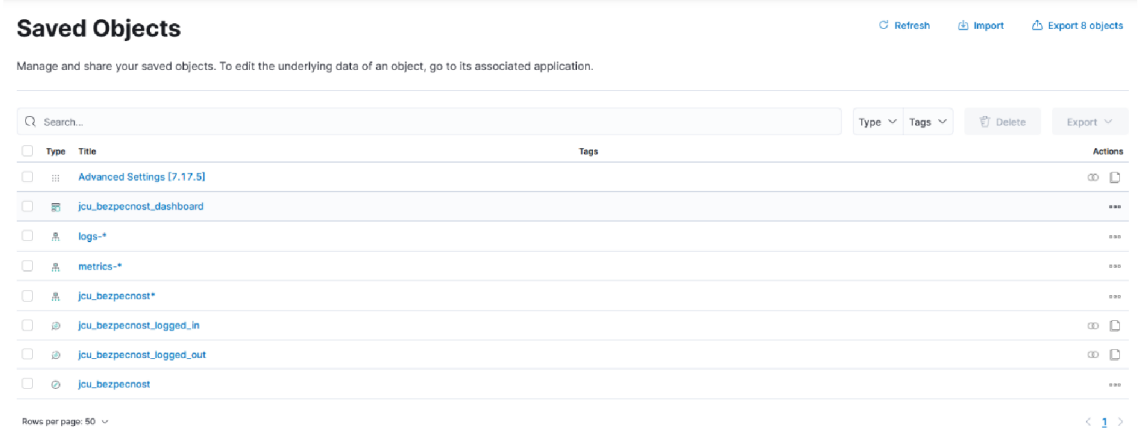
8.3. Dashboard

Tento dashboard se skládá ze dvou vizualizaci logged_in a login_failed. V jedné vizualizaci je znázorněno pomocí tabulky. Na druhé zobrazené vizualizaci vidíme, že však můžeme data zobrazit i pomocí sloupcového grafu či jiných typů grafů.
V dashboardu lze skládat několik vizualizací dohromady a můžeme zde například přiložit i pohled z vyhledávání dokumentů, kde vidíme celé zprávy. Lze tak velice jednoduše sloučit více zdrojů informací do jednoho nad kterými můžeme hromadně v určitém časovém pásmu vykonávat analýzu těchto dat.



Obrázek 25: Dashboard obsahující vizualizace

9. Přenos uložených objektů



Obrázek 26: Všechny uložené objekty ve stacku

Ve správě uložených objektů vidíme všechny objekty v rámci stacku. Nalezneme zde index patterny, pohledy, query dotazy, vizualizace, dashboardy a mnoho dalších. Pokud chceme tyto data sdílet mezi jednotlivými prostředími „spaces“ lze tak učinit vybráním určitých objektů a poté jejich export. Export vytvoří JSON soubor, který definuje veškeré tyto objekty. Pro nahrání do jiného prostředí se přepneme do daného prostředí a pomocí importu veškeré objekty naimportujeme. Tímto tak lze jednoduše sdílet pouze vybraná data do určitých prostředí.

Závěr

Tato Bakalářská práce se zabývala tématem Log managementu a efektivní správě logů v rámci podnikové infrastruktury. Log management je zde řešen na úrovni centrální správy pomocí ELK stacku.

Teoretická část uvedla čtenáře do problematiky logů a proč je dobré podobné řešení použít ve firmě. Jsou zde uvedeny informace o jednotlivých komponentách stacku tak aby bylo srozumitelné, jak celé řešení funguje a jak by bylo vhodné pro konkrétní řešení navrhnout.
Na závěr teoretické části je uvedeno možné použití alternativních řešení.

V praktické části je popsán návod pro zprovoznění podobného clusteru, který se bude skládat z Elasticsearche, Kibany, Logstashe, Filebeatu a Metricbeatu. Tento cluster se v rámci ukázky Log managementu skládá pouze z 1 nodového clusteru. Dále je zde popsáno, jak připravit prostředí pro příjem dat z logstashe, jak zajistit co nejvíce automatizované řešení. V další části je znázorněno, jak pracovat s grafickým rozhraním, například jak pracovat s vývojářským prostředím, jak efektivně spravovat uživatele, prostředí, indexy. Pro kontrolu provozu slouží stack monitoring, který zaznamenává veškerý provoz.

Pojmy a zkratky

Open source je druh softwaru, který je volně distribuován a jeho zdrojový kód je dostupný pro všechny pro užití. Většinou se jedná o kód, který je vyvíjen a udržován spoluprací komunity lidí. [16]

Log je obecné označení pro záznam činnosti či události.

Primární Shard představuje některé nebo všechna data indexu. Po přidání dokumentu do indexu přidá Elasticsearch tento dokument do primárního indexu. [5]

Replika Shard je kopií primárního shardu. Může zlepšit výkon při prohledávání a odolnost ztráty dat distribuováním dat na více nodů. [5]

Autentizace (Authentication) je proces prokázání totožnosti. Využívá se pro přihlašování k aplikacím k osvědčení toho, že jsme těmi, kdo tvrdíme, že jsme. [8]

Nejznámější z metod autentizace je Basic Authentication, která je v jednoduchém formátu typu Jméno, Heslo.

Pomocí jména a hesla se poté vygeneruje Autorizační hlavička, která jméno a heslo převede pomocí algoritmu Base64 do zakódovaného řetězce a odešle je na server. Problém tady spočívá v tom, že přenos musí používat protokol HTTPS pro zašifrování přenášených dat. Pokud by nebyl použit protokol HTTPS, ale jen HTTP, bylo by možné odchytit odesílané packety a dekódováním Base64 algoritmu odhalit přístupové údaje dané osoby. [1]

Autorizace (Authorization) je činnost, která má za úkol přidělit ověřené osobě práva k užívání určitých nástrojů, přístup k datům a tím co s daty může dělat. [8]

Cluster je skupina jednoho nebo více nodů propojených do většího celku tzv. cluster. [5]

API (Application programming interface) je rozhraní pro jednoduchou výměnu dat mezi aplikacemi. Otevírá možnosti dat a funkcionalit pro třetí strany, obchodní partnery a jiné vývojáře. Vývojáři, kteří se chtějí připojit na dané API nemusí znát konkrétní implementaci. Řídí se definicí (schématem) API, která bývá udávána pomocí SWAGGERu a slouží jako jednoznačný závazný kontrakt. Používá protokoly SOAP či REST. [7]

REST (REpresentational State Transfer) protokol, který se užívá při komunikaci v rámci webových služeb. Je nástupcem SOAP protokolu.

JSON (JavaScript Object Notation) je podřazen jazyku Javascript. Obvykle slouží pro výměnu zpráv v komunikaci mezi webovými službami na úrovni klient-server. Je vhodný také pro úschovu dokumentů v NoSQL databázích pro svou jednoduchost a komplexnost. Seskládá se z objektů, které obsahuji dvojice atribut/hodnota. [4]

Tento jazyk používá jednoduché datové typy a těmi jsou řetězec, číslo, logická hodnota a null. A také strukturované datové typy, které jsou objekt a pole. [2]

Vzorová zpráva obsahující všechny možné datové typy:

```
{
  "Nazev_skoly": "Jihočeská univerzita v Českých Budějovicích",
  "Obory": [
    {
      "name": "Ekonomická informatika",
      "Typ_studia": "Bakalářské",
      "Forma_studia": "Kombinované",
      "Fakulta": "EF",
      "Delka_studia": 3,
      "moznost_stipendia": false,
      "pocet_studentu": null,
      "Predmety": [
        "Matematika 1",
        "Mikroekonomie",
        "Informační a komunikační sítě",
        "Základy softwarového inženýrství"
      ]
    },
    {
      "name": "Ekonomika a management",
      "Typ_studia": "Bakalářské",
      "Forma_studia": "Prezenční",
      "Fakulta": "EF",
      "Delka_studia": 3,
      "moznost_stipendia": true,
      "pocet_studentu": null,
      "Predmety": [
        "Matematika 1",
        "Mikroekonomie",
        "Finance podniku",
        "Ekonomika podniku"
      ]
    }
  ]
}
```

Obrázek 27: JSON zpráva

Použité zdroje

1 - Richardson, L., Amundsen, M., & Ruby, S. (2013). *RESTful Web APIs*. Sebastopol: O'Reilly.
2 - Holubová, I., Kosek, J., Minařík, K., & Novák, D. (2015). *Big Data a NoSQL databáze*. Praha: Grada.
3 - Čuvakin, A. A., Schmidt, K. J., & Phillips, C. (2013). *Logging and log management: .: the authoritative guide to understanding the concepts surrounding logging and log management*. Amsterdam: Elsevier.
4 - *Journal of Database Management* [Online]. (2019) (Vol. 30). Retrieved from http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/JDM.2019070103
5 - Https://www.elastic.co/ [Online]. Retrieved October 26, 2022, from https://www.elastic.co/
6 - Log4j – Apache Log4j 2 [Online]. Retrieved October 26, 2022, from https://logging.apache.org/log4j/2.x/index.html
7 - What is an Application Programming Interface (API)? | IBM [Online]. Retrieved October 26, 2022, from https://www.ibm.com/cloud/learn/api
8 - Authentication vs. authorization - Microsoft Entra | Microsoft Learn [Online]. Retrieved October 26, 2022, from https://docs.microsoft.com/en-us/azure/active-directory/develop/authentication-vs-authorization
9 - ElastAlert - Easy & Flexible Alerting With Elasticsearch — ElastAlert 0.0.1 documentation [Online]. Retrieved October 26, 2022, from https://elastalert.readthedocs.io/en/latest/#
10 - About - Open Distro Documentation [Online]. Retrieved October 26, 2022, from https://opendistro.github.io/for-elasticsearch-docs/
11 - Postman API Platform | Sign Up for Free [Online]. Retrieved October 26, 2022, from https://www.postman.com/
12 - Elasticsearch Split Brain Problem - How to Resolve & Avoid it, with FAQ [Online]. Retrieved March 16, 2023, from https://opster.com/guides/elasticsearch/best-practices/elasticsearch-split-brain/
13 - Elk-arch [Online]. Retrieved March 16, 2023, from https://i0.wp.com/sysadminxpert.com/wp-content/uploads/2018/05/elk-arch.jpg?ssl=1
14 - 10 Best ELK Stack Alternatives in 2023 [Online]. Retrieved March 16, 2023, from https://betterstack.com/community/comparisons/elk-stack-alternatives/
15 - Grafana vs Kibana: How to Choose in 2023 [Online]. Retrieved March 16, 2023, from https://betterstack.com/community/comparisons/grafana-vs-kibana/
16 - What is open source software? [Online]. Retrieved March 28, 2023, from https://www.ibm.com/topics/open-source

Seznam obrázků a tabulek