

Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky



Implementace unifikovaného logovacího systému

BAKALÁŘSKÁ PRÁCE

Studijní program: Aplikovaná informatika

Studijní obor: Aplikovaná informatika

Autor: Pavel Horák

Vedoucí bakalářské práce: Ing. Tomáš Bruckner, Ph.D.

Praha, květen 2019

Prohlášení

Prohlašuji, že jsem bakalářskou práci Implementace unifikovaného logovacího systému vypracoval samostatně za použití v práci uvedených pramenů a literatury.

V Praze dne 2. května 2019

.....

Pavel Horák

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Tomáši Brucknerovi, Ph.D. za vedení práce, čas a vřelý přístup. Mé díky patří i kolegům z práce za možnost realizovat práci a cenné rady.

Abstrakt

Bakalářská práce se zabývá problémem decentralizovaného logování a nemožnosti efektivního vyhledávání v logových záznamech aplikace ESB, která se nachází v prostředí informačního systému virtuálního mobilního operátora. Tento problém se snaží vyřešit výběrem a implementací sady nástrojů, které logování centralizují a umožňují full-textové vyhledávání. Problém byl v práci vyřešen a cíl byl tak naplněn. Použité vybrané nástroje pro konkrétní prostředí jsou Elasticsearch, Elasticsearch Appender pro Logback a Kibana.

Klíčová slova

Elasticsearch; Kibana; Logback; Elasticsearch Appender; logování; vyhledávání

Abstract

The bachelor's thesis deals with a problem of decentralized logging and the inability to search effectively in log files of an ESB application, which is located in an environment of a mobile virtual network's information system. It resolves this problem by selecting and implementing such set of tools that centralize logging and allow full-text search. The goal of this work has been fulfilled. The selected tools used for the specific environment are Elasticsearch, Elasticsearch Appender for Logback and Kibana.

Keywords

Elasticsearch; Kibana; Logback; Elasticsearch Appender; logging; search

Obsah

Úvod	11
1 Technologie	12
1.1 Enterprise service bus (ESB)	12
1.2 Maven	12
1.3 Logback	13
1.3.1 Logger	13
1.3.2 Appender	13
1.3.3 Layout, Encoder	14
1.3.4 Konfigurace Logbacku	14
1.4 ELK Stack	14
1.4.1 Elasticsearch	15
1.4.1.1 Lucene	15
1.4.1.2 Cluster	15
1.4.1.3 Shard	15
1.4.1.4 Konfigurace	16
1.4.1.5 Životní cyklus indexů	17
1.4.1.6 Šablony	18
1.4.2 Logstash	18
1.4.3 Kibana	18
1.5 Fluentd	19
1.5.1 Fluentbit	19
2 Stávající řešení	20
2.1 Architektura aplikace ESB	20
2.2 Nastavení logování	20
2.3 Dostupné prostředky serveru	21
2.4 Zhodnocení stavu	22
3 Výběr vhodných nástrojů pro implementaci a návrh řešení	23
3.1 Nástroj indexace a vyhledávání dokumentů	23
3.2 Nástroj pro přenos dokumentů do indexačního nástroje	24
3.2.1 Logstash	24
3.2.2 Fluentd	24
3.2.3 Fluent bit	24
3.2.4 Elasticsearch appender	25

3.2.5 Porovnání využití paměti	26
3.2.6 Porovnání řešení	26
3.3 Grafické rozhraní pro vyhledávání	27
3.4 Návrh řešení.....	27
4 Implementace navrženého řešení	31
4.1 Instalace	31
4.2 Konfigurace serveru.....	31
4.2.1 Konfigurace Elasticsearch	32
4.2.2 POM	33
4.2.3 Konfigurace Logbacku.....	34
4.2.4 Konfigurace Kibany.....	36
4.3 Spuštění	36
4.3.1 Elasticsearch	36
4.3.2 Kibana	37
4.4 Doplnková konfigurace.....	37
4.4.1 Politika životního cyklu indexů	38
4.4.2 Index patterns	40
4.5 Použití	42
Závěr	44
Použitá literatura.....	46

Seznam výpisů programového kódu

Výpis z kódu 1: Ukázka textu, který se přidáním do souboru pom.xml přidá závislost do projektu. Dostupné z https://mvnrepository.com/artifact/net.logstash.logback/logstash-logback-encoder/5.3	12
Výpis z kódu 2: Ukázka konfigurace Elasticsearche v souboru elasticsearch.yml	33
Výpis z kódu 3: Ukázka konfigurace Elasticsearche v souboru jvm.options.....	33
Výpis z kódu 4: Ukázka konfigurace Mavenu v souboru pom.xml	34
Výpis z kódu 5: Ukázka konfigurace Elasticsearch Appenderu v Logbacku v souboru logback.xml.....	34
Výpis z kódu 6: Ukázka konfigurace loggeru ELASTIC-ERROR v Logbacku v souboru logback.xml.....	35
Výpis z kódu 7: Ukázka chybové hlášky z Elasticsearch Appenderu.....	35
Výpis z kódu 8: Ukázka výsledku na dotaz na Elasticsearch pomocí příkazu curl	37
Výpis z kódu 9: Úryvek z příkazu curl k vytvoření nové šablony s mapováním.....	39

Seznam obrázků

Obr. 1 Deployment diagram aplikace ESB (vlastní zpracování)	20
Obr. 2 Dataflow diagram: Reprezentace datového toku mezi komponentami při ukládání nových záznamů (vlastní zpracování)	21
Obr. 3 Deployment diagram návrhu řešení (vlastní zpracování).....	28
Obr. 4 Dataflow diagram: Reprezentace datového toku mezi komponentami při vytvoření a indexování nových záznamů (vlastní zpracování).....	29
Obr. 5 Dataflow diagram: Reprezentace datového toku mezi komponentami při vyhledávání dokumentů v Elasticsearchi (vlastní zpracování)	30
Obr. 6 Menu pro upravení politik životního cyklu indexů.	38
Obr. 7 Přidání nové politiky k šabloně.	40
Obr. 8 Vybrání správné šablony k přiřazení k politice.....	40
Obr. 9 Vytváření vzoru pro indexy, ve kterých má vyhledávání probíhat.....	41
Obr. 10 Výběr časového pole indexu.	42
Obr. 11 Zobrazení možností vyhledávání v nástroji Kibana.	43

Seznam zkratek

JVM	Java Virtual Machine
SOA	Service Oriented Architecture – architektura orientovaná na služby
ESB	Enterprise service bus
SLF4J	Simple Logging Facade for Java
API	Application Programming Interface – rozhraní pro programování aplikací
JAR	Java Archive
POM	Project Object Model
RPM	Red Hat Package Manager – balíčkovací systém
RAM	Random Access Memory – operační paměť

Úvod

V informatice existují různé aplikace. Některé zprostředkovávají služby a některé jsou samostatné produkty. Je vhodné, aby tyto aplikace tvořily záznamy o své činnosti a stavu, ve kterém se nachází. Takové záznamy se mohou sledovat, a na jejich základě se mohou vytvářet analýzy, pomocí kterých může docházet k vylepšení nebo k úpravám aplikace. Vylepšení jsou přínosná, ale ještě důležitější je, aby aplikace dokázala vyhovět zákazníkům a mohla zajistit nepřetržitý provoz. Pokud se aplikace dostane do nežádoucího stavu, správci nebo podpora se vždy obrátí právě na tyto záznamy o činnostech aplikace, ze kterých se mohou dozvědět více o kontextu problému. Je tedy jasné, že záznamy jsou důležité nejen pro budoucí vývoj, ale i pro retrospektivní pohled, který umožňuje aplikaci dostát závazkům funkčnosti. Těmto záznamům se říká také „logové záznamy“, nebo zkráceně „logy“.

Teď už není pochyb o důležitosti logových záznamů, avšak druhým krokem je zajistit jednoduchý přístup k jednotlivým záznamům. Logy, ke kterým se správce nedostane, neplní ani jednu ze svých funkcí. Pokud tedy záznamy existují a odpovědné osoby k nim mají přístup, bude vše probíhat, jak má? Nemusí. Může nastat situace, kdy aplikace běží na více místech (serverech) najednou a záznamy se ukládají na místě, na kterém příslušná aplikace funguje. Takové řešení značně omezí rychlost, se kterou může podpora reagovat na problém, protože se musí podívat do záznamů na více místech. Další situace, která může nastat, je použití různých formátů záznamů, kdy se ke každému musí přistupovat odlišně. Jiným způsobem se bude pracovat se souborem MS Excel a jinak s jednoduchým souborem „.txt“. Tato situace by práci také neusnadnila, spíše naopak. Takže se nabízí ještě další podmínka efektivního využívání logových záznamů, a to jejich centralizované skladování a jednotný formát.

Cílem této práce je vybrat a implementovat sadu nástrojů, které pomohou v prostředí informačního systému virtuálního mobilního operátora vyřešit problém s decentralizovaným logováním. V prostředí běží aplikace typu Enterprise Service Bus, a je třeba sjednotit logové záznamy z různých výstupů této aplikace. Zároveň musí být nástrojem umožněno v záznamech full-textové vyhledávání.

Práce provede čtenáře výběrem vhodných nástrojů pro konkrétní prostředí a následně i samotnou implementací takového řešení. Nejdříve se představí jednotlivé technologie, které v práci figurují. Poté je krátce zhodnoceno aktuální řešení aplikace ESB a její konfigurace logování. V návaznosti na to je popsán výběr jednotlivých nástrojů řešení. Po výběru je popsána samotná implementace zvolených nástrojů a jejich konfigurace k ideálnímu použití pro konkrétní prostředí. Na závěr je ukázka funkcí, které řešení umožňuje.

1 Technologie

V této kapitole je popsán souhrnný základ znalostí pro porozumění a snadnější orientaci v následujícím textu.

1.1 Enterprise service bus (ESB)

Enterprise Service Bus je standardizovaná platforma, která kombinuje práci se zprávami, webové služby, transformaci dat a určování směru toku dat. To vše za účelem propojení a koordinace interakce více heterogenních aplikací. Také se dá uvažovat o modelu softwarové architektury, která implementuje dorozumívání mezi vzájemně komunikujícími softwarovými aplikacemi v architektuře orientované na služby (SOA). Bez ESB mohou být aplikace propojeny přímo, ale toto řešení se velmi obtížně škáluje a udržuje. Proto existuje ESB, které jednotlivé aplikace nepřímo propojuje skrze své rozhraní. ESB je tak zodpovědné za logiku interakce a integrace jednotlivých systémů. (Sanchit Agrawal 2016; Laliwala et al. 2005)

1.2 Maven

Apache Maven, známý jen jako Maven, je nástroj, který se používá pro sestavování a správu jakýkoliv projektů, které jsou postavené nad Javou. Pomáhá s organizací, plánováním a správou zdrojů projektu. V praxi to znamená, že Maven umožňuje stáhnout potřebné zdrojové soubory, soubory tříd a JAR soubory pomocí jednoduché konfigurace v *pom* (Project Object Model) souboru (Shah 2014).

Knihovny, které jsou dostupné pomocí Mavenu, lze najít na <https://mvnrepository.com>, kde se nachází i část kódu, kterou je možné přidat do *pom* souboru. Následující příklad umožňuje přidat do projektu potřebné soubory JAR pro využití Logstash Encoder pro Logback:

```
<dependency>
  <groupId>net.logstash.logback</groupId>
  <artifactId>logstash-logback-encoder</artifactId>
  <version>5.3</version>
</dependency>
```

Výpis z kódu 1: Ukázka textu, který se přidáním do souboru *pom.xml* přidá závislost do projektu. Dostupné z <https://mvnrepository.com/artifact/net.logstash.logback/logstash-logback-encoder/5.3>

1.3 Logback

Logback je konkrétní implementace abstrakce SLF4J (Simple Logging Facade for Java). SLF4J nabízí dostatečně obecné rozhraní tak, aby samotné logování bylo nezávislé od samotné implementace.(QOS.ch 2019)

Logback je nástupce Log4j. Je to knihovna pro Javu pro práci s logováním. Skládá se ze tří hlavních modulů: logback-core, logback-classic a logback-access. Core obsahuje základy pro fungování celého Logbacku, a classic představuje vylepšenou verzi Log4j, která implementuje SLF4J. Logback-access integruje funkcionalitu Java Servlet, takže umožňuje pracovat s HTTP rozhraním. (Ceki Gülcü et al. 2017a)

Logback pracuje se třemi komponenty: Logger, Appender a Layout. Kombinací těchto komponent je možné specifikovat, jaké typy zpráv bude Logback zaznamenávat, jaký bude jejich formát a kam se mají posílat.

Pracuje také s úrovněmi záznamů. Úrovní je pět:

- Error – pokud událost, která záznam vyvolala, vyžaduje bezprostřední lidský zásah, je na této úrovni.
- Warn – neočekávaná událost, která sice vyžaduje lidský zásah, ale nemusí být bezprostřední
- Info – do této kategorie spadají události životního cyklu, jako jsou např. spuštění nebo zastavení systému. Jsou to události, které popisují stav systému, a tak se využívají především pro diagnostiku případných problémů.
- Debug – v této kategorii jsou především události, které sledují tok skrze systém – může to být vstup nebo výstup ze složitějších metod nebo označení neobvyklých událostí uvnitř metody.
- Trace – tato kategorie obsahuje detailní přehled všech událostí, které v systému nastanou.(Ceki Gülcü et al. 2017a)

1.3.1 Logger

U komponenty loggeru můžeme specifikovat, jaký typ události se má zaznamenávat. Možnosti jsou: trace, debug, info, warn, error. Pokud není specifikován jeden z typů události, logger automaticky přijme typ svého předka. Aby se zajistila možnost, že každý logger eventuálně zdědí nějaký typ, existuje kořen (root), který je předek všech loggerů, a který má ve výchozím nastavení typ debug.(Ceki Gülcü et al. 2017a)

1.3.2 Appender

V komponentě appender se oproti loggeru vyhodnotí, kam mají události pokračovat, resp. do jakého výstupu. Výstupů je pro Logback naprogramováno mnoho, od výstupů do konzole až po výstupy do databáze. Navíc existuje možnost naprogramovat si vlastní appender. (Ceki Gülcü et al. 2017c)

1.3.3 Layout, Encoder

Při nastavení výstupů je vhodné nastavit i formát, v jakém data opustí logger. I když je appender zodpovědný za správný výstupní formát, lze delegovat tuto činnost na komponentu Layout nebo Encoder. Layout komponenta převezme data vstupní události a přetransformuje je na String. Encoder je oproti Layout komponenta, která data transformuje do pole bytů. Výstup lze ještě specifikovat pomocí tzv. patterns neboli vzorů, které určí, jaké informace se mají zaznamenávat. (Ceki Gülcü et al. 2017d; 2017a)

Vzory vždy začínají „%“ a poté následuje klíčové slovo. Vzorů, které jsou pro Logback dostupné, je mnoho. Pro tuto práci jsou tu vypsána klíčová slova těch, které jsou přímo použité.

- Level – úroveň události viz kapitola 1.3
- X – tento vzor umožňuje přidat hodnotu z kontextu vlákna, které událost vytvořilo.
- Thread – produkuje jméno vlákna, které událost vytvořilo.
- xEx{7} – produkuje výstup ze stack tracu z výjimky, která je spojená s událostí, která nastala. Číslo reprezentuje počet řádků, kolik se má ze stack tracu zaznamenat. xEx zaznamenává i soubor *jar*, který chybu vyvolal.
- logger{36} – produkuje jméno loggeru, který je u zdroje události. Číslo reprezentuje počet písmen loggeru, které se mají zaznamenat.
- msg – zaznamenání dodatečné zprávy aplikace, která je spojená s událostí.(Ceki Gülcü et al. nedatováno)

1.3.4 Konfigurace Logbacku

Logback se může konfigurovat přímo v kódu nebo pomocí výrazů v souboru XML. Logback při spuštění vyhledá soubor *logback.xml*, ve kterém se samotná konfigurace nachází. (Ceki Gülcü et al. 2017b)

1.4 ELK Stack

ELK je kompletní řešení analýzy logových záznamů, které je postaveno na třech open source nástrojích, a to Elasticsearch, Logstash a Kibana. Elasticsearch představuje zdroj analýzy dat a vyhledávání, Logstash je část, která má na starosti řízení logových záznamů. To může být přenos záznamů do více destinací, jejich transformace a parsování. Pokud je Logstash nástroj, kterým se data dostávají do Elasticsearche, Kibana je nástroj, kterým se informace dostanou v přehledné formě, ať už jsou to grafy či lehce upravitelné přehledy, ke koncovému uživateli. Celý proces je tedy následující: data z různých zdrojů se posílají do Logstashe, který je shromáždí a případně transformuje, následně je Logstash pošle do Elasticsearche, kde se data zaindexují a uloží, poté koncový uživatel zadá svůj dotaz pomocí Kibana nástroje, který data zobrazí a zkonstruuje přehledy. (Saurabh Chhajed 2015)

1.4.1 Elasticsearch

Elasticsearch je distribuovaný systém, který umožňuje analytické funkce a vyhledávání téměř v reálném čase. Především se využívá pro full-textové vyhledávání, strukturované vyhledávání, analýzu dat, případně kombinaci těchto tří věcí. Velké společnosti, jako je např. Wikipedia, The Guardian či Stack Overflow používají Elasticsearch. Výhoda je velká škálovatelnost. Využitý může být na malých serverech, kde data nepřesáhnou 1 GB, nebo naopak lze Elasticsearch škálovat přes více serverů s tisíci GB dat. (Gormley a Tong 2015)

1.4.1.1 Lucene

Elasticsearch je postaven na rozhraní Apache Lucene. Apache Lucene je full-textový vyhledávací engine, který je považován za jeden z nejrozvinutějších, vysoce výkonných a plně vybavených vyhledávacích enginů, který lze použít, ať už je řešení open source či proprietární. Lucene existuje jako knihovna, která se musí pomocí jazyka Java integrovat do aplikace, aby se využil potenciál, který nabízí. (Gormley a Tong 2015) Jediná její funkce je indexace a vyhledávání. Při integrování s aplikací komunikuje Lucene skrze otevřené API, takže je možné definovat business procesy v aplikaci a skrýt tak samotnou logiku indexace. (McCandless et al. 2010)

1.4.1.2 Cluster

Jedna instance běžícího Elasticsearch se nazývá *node*. *Cluster* obsahuje jeden nebo více nodů, kteří mají shodné jméno clusteru v konfiguraci. Jednotlivé nody spolu v clusteru spolupracují, sdílejí svá data i práci. Elasticsearch sám rozpozná kolik nodů je v clusteru a při změně přeorganizuje svou strukturu. Jeden z nodů musí být zvolen jako *master node*, který má kromě své funkce, jako instance Elasticsearch, ještě funkci řízení clusteru. Jakýkoliv node může být *master node* a Elasticsearch vyhodnotí, jaký node je nejvhodnějším kandidátem. Ten node, se kterým uživatel komunikuje, rozešle dotaz v clusteru, shromáždí odpovědi z nodů a poté vrátí odpověď klientovi, který dotaz zadal. (Gormley a Tong 2015)

1.4.1.3 Shard

Abychom mohli data v Elasticsearch uchovávat, musíme ukládat data do *indexů*. Index ukazuje na jeden nebo více *shardů*. Shard je jednotka, která obsahuje část informací, na které index odkazuje. Zároveň, každý shard je jedna instance enginu Lucene. Naše dokumenty jsou indexovány a uloženy ve shardech, ale Elasticsearch komunikuje pouze s indexem. Elasticsearch zároveň používá shardy pro distribuci dat v clusteru. Při změně velikosti clusteru Elasticsearch automaticky přesune shard mezi nody tak, aby byl cluster v rovnováze. (Gormley a Tong 2015)

Shard se dělí na dva druhy – *primary*, *replica*. Každý dokument pak patří do jednoho z primárních shardů. Replica je jen kopie primary shardu. V případě výpadku primary shardu je možné vyhledávat v replikách. Počet replik se může za běhu měnit, oproti tomu počet primary shardů je pevně daný u vzniku indexu. (Gormley a Tong 2015)

1.4.1.4 Konfigurace

Konfigurace Elasticsearch se primárně určuje v souboru *Elasticsearch.yml*. Soubor se upravuje u každého nodu zvlášť. Nastavuje se zde:

- `Cluster.name` – název clusteru, ve kterém tento konkrétní node existuje.
- `Node.name` – název nodu
- `Path.data` – cesta, kam si má ukládat své data, v tomto případě jsou to indexy a shardy.
- `Path.logs` – cesta k vlastním logovým záznamům. (Elasticsearch B.V nedatováno)
- `Network.host` – na jaké síťové adrese se má služba publikovat. Výchozí nastavení je localhost (127.0.0.1).
- `http.port` – na jakém síťovém portu se Elasticsearch zpřístupní. (Elasticsearch B.V nedatováno)
- `Discovery.zen.ping.unicast.hosts` – obsahuje adresy ostatních nodů, které se mají nacházet v clusteru. Na těchto adresách se bude Elasticsearch snažit navázat komunikaci s ostatními nody, se kterými očekává spolupráci v clusteru. Komunikaci se bude snažit navázat na portu TCP 9300. (Elasticsearch B.V nedatováno; nedatováno)
- `Discovery.zen.minimum_master_nodes` – Aby se předešlo situaci, kdy jeden cluster má více než jeden master node, musí se určit, kolik nodů může být master nodem v clusteru. Tato situace by byla problémem, protože master node má na starosti řízení celého clusteru a s tím souvisí vytváření nových indexů, jak se pohybují shardy apod. (Elasticsearch B.V nedatováno)
- `Bootstrap.system_call_filter` – Elasticsearch se při spouštění pokusí izolovat do bezpečného prostředí neboli bezpečného módu *seccomp* (Secure Computing Mode) (Tyler Hicks et al. nedatováno; Elasticsearch B.V nedatováno). Ten zaručuje, že proces bude volat jen určité systémové funkce, a v případě zvolání jiných, jádro Linuxu proces automaticky ukončí. Je možné, že konkrétní Linuxové jádro použití funkcionality *seccomp* nepodporuje nebo není povoleno, proto je možné kontrolu, zdali se podařilo Elasticsearchi izolovat, vypnout. (Jason Todor nedatováno)

Pro nastavení prostředí Javy, ve kterém Elasticsearch běží, se využívá soubor *jvm.options*. V tomto souboru se primárně určí velikost paměti, která je Elasticsearch procesu vyhrazena a může ji použít (Gormley a Tong 2015). Pokud jsou hodnoty maximální a minimální velikosti paměti rozdílné, může docházet ke zpomalení procesu. Zpomalení vychází ze změny velikosti heap paměti za běhu procesu. Proto je lepší nakonfigurovat minimální a maximální velikost paměti na stejné hodnoty. Existuje ještě nastavení *bootstrap.memory_lock*, které přikáže Elasticsearchi přiřadit a zamknout si paměť při spuštění (Elasticsearch B.V nedatováno). Nastavení konkrétní hodnoty se odvíjí od dostupné paměti zařízení, na kterém Elasticsearch poběží. Čím více paměti bude mít proces k dispozici, tím více paměti může využít k provozu. Na druhou stranu, není vhodné nastavit maximální paměť na více než 50 % dostupné paměti. Je to z důvodu zajištění dostatečného místa pro systémové operace. Druhé omezení, kterému se musí při konfiguraci přihlídnout, je omezení heap paměti samotného JVM, které je přibližně 32GB (Elasticsearch B.V nedatováno).

1.4.1.5 Životní cyklus indexů

Pro správu životního cyklu indexů (ILM – Index Lifecycle Management) existuje API, které umožňuje správu automatizovat. Nabízí automatizovat činnosti na základě pevného rozvrhu, velikosti shardů nebo i dokonce podle výkonu jednotlivého hardwaru, který je pro daný index dostupný. To může být například přesun méně používaných shardů na méně výkonný hardware a více používané shardy na výkonnější hardware.

Životní cyklus indexu má čtyři fáze:

- Hot – Index je aktivně využíván, to znamená, že nad ním probíhá aktualizace nových záznamů a vyhledávání odpovídek na dotazy.
- Warm – V indexu je vyhledáváno, ale už není doplňován o nové záznamy.
- Cold – Index není doplňován o nové záznamy, ale informace jsou potřeba držet pro občasná nebo případná vyhledávání.
- Delete – Index není potřeba a je možné ho smazat.

Funkce, které správu umožňují, jsou následující:

- Vytvoření nového indexu na základě velikost nebo věku stávajícího.
- Body, kdy index už nemá být aktualizován a může proběhnout snížení počtu jeho shardů tzv. *index shrinking*.
- Kdy se index smaže.
- Kdy může být index přesunutý na hardware, který není tak výkonný.
- Bod, kdy není dostupnost dat už tak důležitá a počet *replic* se může snížit.

Správu indexů je možné nastavit se všemi funkcemi skrze grafické rozhraní Kibany. Nicméně, je možné i využít již zmíněné API (Elasticsearch B.V nedatováno).

U snížení shardů pro index (*index shrinking*) je nutné dbát zvýšené opatrnosti. Výsledkem je zmenšení existujícího indexu do nového, s menším počtem shardů. Počet nových shardů ale musí být celočíselným dělitelem původního počtu shardů. Pokud je počet shardů prvočíslo, musí nový počet shardů být stejný nebo se rovnat jedné. Proces je následovný:

- Vytvoří se nový index s menším počtem shardů, který má stejné nastavení jako původní index.
- Poté vytvoří pevné odkazy souborů (*hard-link*) z původního indexu do nového. Pokud systém, na kterém Elasticsearch běží, nepodporuje vytváření pevných odkazů souborů, všechna data jsou nakopírována.
- Poté se nový index zpřístupní.

Při pohledu na proces zmenšení počtu shardů indexu vyplývá, že musí existovat další požadavky na index, které musí splnit před zahájením zmenšení. Index může být zmenšen jen po splnění následujících požadavků:

- Nový index musí existovat – proces vytváření nového indexu musí být úspěšný.
- Původní index musí mít stejně nebo více shardů než nový.

- Jak již bylo zmíněno, počet shardů nového indexu musí být celočíselným dělitelem původního počtu shardů.
- Při zmenšování shardů není možné, aby nějaký výsledný shard měl více než 2 147 483 519 (limit Integer, od kterého je odečteno 128) indexovaných dokumentů, neboť tento limit je dán Lucene indexem, který může pojmout právě takové množství dokumentů.
- Na úložišti musí být dostatečně místa k duplikování celého indexu (Elasticsearch B.V nedatováno).

1.4.1.6 Šablony

Šablony se mohou použít pro definování nastavení nebo mapování dat. Každá šablona má ve svém nastavení vzory (patterns), které fungují jako filtr pro její aplikování na vytvářený index. Šablony se používají jen při vytváření indexů, a nelze tedy upravit šablonu pro index, který již vytvořen byl. Pro vytvoření šablony se musí specifikovat vzor a počet shardů (Elasticsearch B.V nedatováno).

1.4.2 Logstash

Logstash je nástroj napsaný v Javě, který představuje datový tok. Jeho funkcí je sbírat, parsovat a analyzovat strukturovaná i nestrukturovaná data a události, které mohou pocházet z více různých zdrojových systémů. (Saurabh Chhajed 2015)

Tok se dělí na tři fáze – vstup, filtr a výstup. Vstup přijímá data z jednoho nebo více zdrojů v jakémkoliv formátu. Pro vstup existují zásuvné moduly, které usnadňují nástroji zpracování vstupních dat. Filtr umožňuje nástroji parsování a transformaci každé události, která tokem prochází. Nástroj nabízí i např. anonymizaci dat nebo dešifrování lokace dle IP adresy. Výstup obstarává distribuci dat v konečné formě do jednoho nebo více systémů. I zde jsou k dispozici různé zásuvné moduly, které tak umožňují efektivnější komunikaci s výstupními systémy. (Elasticsearch B.V nedatováno)

Logstash tedy nabízí různé zásuvné moduly, které umožňují připojení k různým typům zdrojů a platform, a jsou navrženy tak, aby efektivněji zpracovaly záznamy, události a nestrukturovaná data k distribuci do různých výstupních modulů. (Saurabh Chhajed 2015)

1.4.3 Kibana

Kibana je open source platforma pro grafickou reprezentaci strukturovaných i nestrukturovaných dat, která jsou uložena v Elasticsearch indexech. Využívá otevřenou REST API od Elasticsearch, aby umožnila koncovému uživateli vyhledávat a graficky znázorňovat data. Skrze své rozhraní nabízí dynamické přehledy, které zobrazují dotazy na Elasticsearch v reálném čase. (Saurabh Chhajed 2015)

1.5 Fluentd

Fluentd je nástroj pro sběr logových záznamů, který je zdarma a open source. Pracuje se záznamy ve formátu JSON a je napsaný převážně v jazyce C v kombinaci s Ruby. Ruby a C nejsou tak náročné na využití paměti jako je Java, a proto je vhodnější pro systémy, kde je paměti nedostatek.

Data se snaží převážně transformovat do formátu JSON, se kterým poté pracuje. Protože mají data jednotnou strukturu díky JSONu, je Fluentd schopen aplikovat různé filtry nebo transformace dat do jiných formátů. Zároveň je Fluentd flexibilní nástroj, který má díky své komunitě k dispozici více než 500 zásuvných modulů pro vstup nebo výstup dat. (Fluentd Project 2019)

1.5.1 Fluentbit

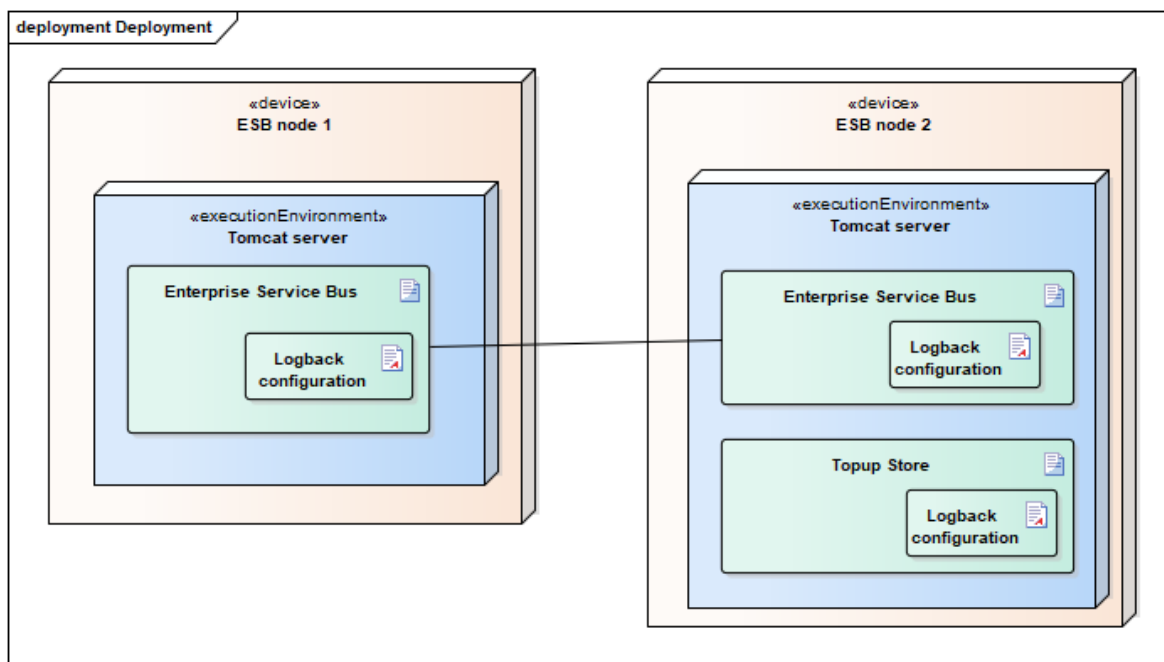
Pokud jsou desítky MB paměti, které Fluentd využívá, moc, je možné využít nástroj Fluentbit. Tento open source nástroj je dostupný pro Linux, OSX a BSD (Eduardo Silva et al. 2018). Podobně jako Fluentd i Fluentbit umožňuje řídit tok dat záznamů z jedné lokace do druhé. Fluentd se zaměřuje především na sběr, zpracování a agregaci záznamů, Fluentbit se zaměřuje na sběr a zpracování záznamů s omezenou funkcí agregace. Oproti Fluentd je napsán čistě v C, a tak je jeho využití paměti v řádech KB. Na druhou stranu nabízí mnohem méně zásuvných modulů než Fluentd, kolem 35. (Eduardo Silva et al. 2019b)

2 Stávající řešení

Pro výběr vhodných nástrojů je důležité seznámit se nejprve s prostředím, do kterého se budou nástroje implementovat. V této kapitole je tedy popsána architektura aplikace ESB a její konfigurace logování. Je zde i popsán datový tok vytváření a ukládání záznamů a na závěr kapitoly i zhodnocení dostupných prostředků serveru.

2.1 Architektura aplikace ESB

Pro reprezentaci architektury je využit Deployment diagram (viz obr. 1). Na obrázku je možné vidět, že i když aplikace běží na více serverech, komunikují mezi sebou. Každá aplikace pracuje v prostředí Tomcat serveru. Aplikace mají svou vlastní konfiguraci Logbacku (viz kapitola 1.3), která definuje způsob ukládání záznamů.



Obr. 1 Deployment diagram aplikace ESB (vlastní zpracování)

2.2 Nastavení logování

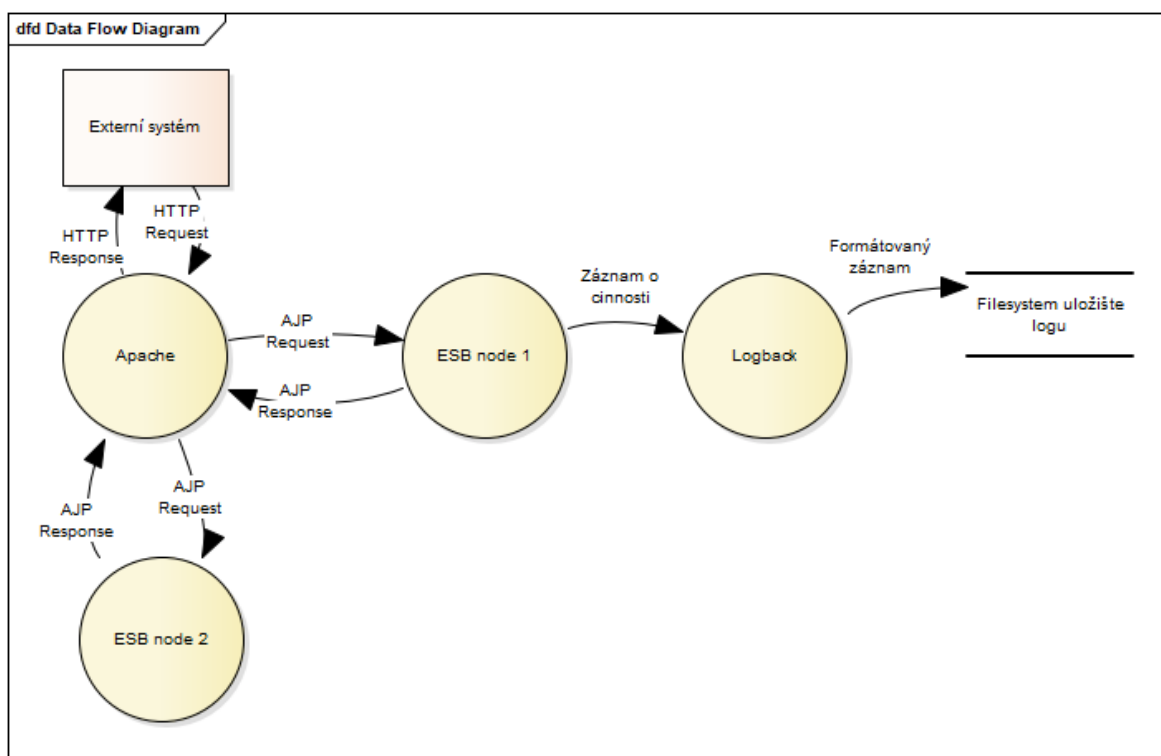
V aplikaci ESB je implementován Logback (viz kapitola 1.3) a je zde i nakonfigurován skrze soubor logback.xml. Stávající nastavení definuje následující Appendery (viz kapitola 1.3.2) pro Logback:

- Konzole – vypsání záznamů jen do konzole.
- Soubor – výstup do souboru. Pokud je soubor větší než 10 MB, zkomprimuje se. Soubory se drží maximálně 7 dní, poté se odstraní. Druhé omezení existence

souborů je velikost. Pokud součet velikostí souborů překročí hranici 250 MB, nejstarší soubor se odstraní. Do těchto souborů se ukládají všechny záznamy, které aplikace ESB produkuje.

- Alert – v případě, že je aplikace zaneprázdněna nějakým procesem příliš dlouho, záznam o této události se uloží do souboru skrze tento Appender. Tyto záznamy jsou ve své vlastní složce.
- Method duration – ukládání doby trvání Java metod v některých třídách do souborů. Tyto záznamy jsou ve své vlastní složce.

Z nastavení lze ověřit, že tvořené záznamy jsou ukládány do souborů, v jednom případě jen vypsány na konzoly, tzn. neukládány. Pro analýzu logů je proto třeba procházet jednotlivé soubory. Reprezentace datového toku (viz obr. 2) je udělána pomocí Dataflow diagramu. Na obrázku je vidět způsob, jakým data proudí, kde se transformují a ukládají. Aplikace ESB je provolána externím systémem. Požadavek i odpověď prochází Apache serverem, který rozhoduje o tom, jaký node požadavek zpracuje a odpoví na něj. Při zpracování požadavku se tvoří záznamy o činnosti aplikace ESB. Záznamy o těchto činnostech jsou dle nastavení Logbacku naformátovány do určité formy pomocí Patterns (viz kapitola 1.3.3), a dále ukládány do souborů nebo vypsány na konzoly.



Obr. 2 Dataflow diagram: Reprezentace datového toku mezi komponentami při ukládání nových záznamů (vlastní zpracování)

2.3 Dostupné prostředky serveru

V této části se práce zaměří na dostupné prostředky serveru jen z pohledu dostupného místa v úložišti a v dostupné paměti RAM. Místo v úložišti serveru je důležité z pohledu ukládání

záznamů. Tato informace také říká, zdali je dostatek dostupného místa po instalaci dalších nástrojů. Paměť RAM je třeba pro běh všech aplikací. Od dostupné paměti se také odvíjí, jaká aplikace se může na server nainstalovat. Některé mají větší nároky na paměť než jiné. Pomocí příkazu `free` s parametrem `-m` je možné zjistit informace o paměti – celková paměť, využití, dostupnost aj.

Tab. 1 Využití paměti serveru v jednotkách MB (vlastní zpracování)

Druh paměti	Celková	Využívaná	Dostupná
RAM	4 842	3293	1548
Buffer/Cache	-	2501	2340

Dostupná paměť je okolo 1548 MB. Pokud se k tomuto číslu přičte paměť, která je ve vyrovnávací paměti nebo v cache, výsledná volná paměť je přibližně 3888 MB. Více než půl volné operační paměti je v cache, proto je možné říci, že prostředky server jsou z pohledu RAM omezené. Navrhovaná řešení budou muset toto omezení zohlednit a v případě dvou vyhovujících řešení vybrat to, které má nároky na paměť menší.

Informace o dostupném úložišti serveru se dají zjistit pomocí příkazu `df`. Pro lepší čitelnost je použit parametr `-h`, který výstup převede do jednotek MB nebo GB tak, aby byl čitelný. V tab. 2 je možné vidět, že místa je na serveru dostatečně, neboť prozatím je využito jen 26 % z celkového úložiště. Návrh řešení tedy nemusí stanovit velkou váhu nárokům na úložiště.

Tab. 2 Využití úložiště serveru (vlastní zpracování)

Úložiště	Celková	Využívaná	Dostupná	Využití v %
root	35 GB	8.3 GB	25 GB	26

2.4 Zhodnocení stavu

Nastavení ukládání záznamů není ideálním řešením, neboť se při analýze logových záznamů musí procházet soubory ručně. Momentálně neexistuje na serveru nástroj, který by umožnil vyhledávání například skrze servery nebo po vláknech aplikace. Z tohoto důvodu je vhodné najít nástroj, který tuto činnosti umožní a nevyžaduje po uživatelích speciální vzdělání.

Prostředky serveru jsou dostačující pro stávající řešení i s rezervou, co se týče úložiště. Z pohledu využití RAM jsou prostředky sice dostačující, ale nenabízí mnoho prostoru pro další využití. Je proto vhodné najít nástroje, u kterých je při výběru nízké využití RAM jako priorita.

3 Výběr vhodných nástrojů pro implementaci a návrh řešení

V předchozí kapitole byl zhodnocen stav a konfigurace prostředí. Na základě získaných informací je možné postoupit k výběru nástrojů, které jsou pro konkrétní situaci vhodné. Po výběru následuje i návrh řešení, jak tyto nástroje implementovat do prostředí.

Dosavadní řešení splňuje informační potřebu vývojářů, a tak není žádoucí měnit obsahovou stránku záznamů. Avšak existují nástroje, které umí shromažďovat logové záznamy a pro snazší práci s nimi umožňují v záznamech vyhledávat. Nejdříve je nutné vybrat nástroj, který bude záznamy zpracovávat. Poté najít spojení mezi ESB a nástrojem.

Během výběru je důležité zohlednit požadavky zadávajícího. Požadavky byly stanoveny následovně:

- Řešení je zdarma
- Open-source
- Jednoduše horizontálně škálovatelný
- Podporuje full-textové vyhledávání
- Filtrace podle různých klíčů

3.1 Nástroj indexace a vyhledávání dokumentů

Podle metrik, které zohledňují např. počet zmínění na webových stránkách, frekvence technických diskuzí a počet nabídek práce, ve kterých je systém zmíněn, vychází s nejlepším skóre Elasticsearch. S kritériem Open-source je na druhém místě s méně než polovičním skóre řešení Solr.(solid IT gmbh 2019)

Oba nástroje splňují požadavky a jsou Open-source. Je tedy možné se podívat na aktivitu příslušného repositáře, a usoudit tak, zda je nástroj stále aktivně vyvíjen či vývoj stagnuje. Pro porovnání je použit volně dostupný nástroj GitHub Compare, který sběrem dat z příslušných repositářů dokáže tyto repositáře porovnat. Při porovnání počtu uzavřených problémů a počtu lidí, kteří se podílí na vývoji, Elasticsearch vítězí. (Brian Payne 2014)

Aby se mohla porovnat aktivita v čase, je nutné porovnat čísla z repositářů v časovém horizontu. Na tuto činnost je vybrán volně dostupný nástroj od společnosti Sematext Group, který dokáže porovnat data z repositářů v tabulce i graficky v závislosti na čase. V tomto porovnání je Elasticsearch aktivnějším, i když mladším nástrojem.(Sematext Group 2019)

Optimálním nástrojem indexace a vyhledávání ve zpracovaných dokumentech je pro tento případ Elasticsearch.

3.2 Nástroj pro přenos dokumentů do indexačního nástroje

Jak již bylo zmíněno v kapitole 2, dosavadní řešení vytváření a formátování logových záznamů je pomocí Logbacku (viz kapitola 1.3). Pro zachování konzistence tohoto řešení je vhodné najít takový nástroj, který je s tímto kompatibilní. Je tedy nutné najít nástroj, který zároveň implementuje Appender (viz kapitola 1.3.2) pro Logback. A nejen to, Elasticsearch využívá ve výchozím nastavení 1GB pro JVM heap pro svou práci s indexováním a držení cache posledních dotazů. Vzhledem k celkové paměti a volné paměti na serveru, je nutné při výběru zohlednit nároky na paměť ostatních komponent. Kromě paměti už nejsou jiné speciální požadavky na přesun logů z aplikace do indexovacího nástroje.

3.2.1 Logstash

Kvůli existenci ELK stacku, který představuje kompletní řešení, a Appenderu k Logbacku, je Logstash potenciálně první volba ve výběru. Musí se ovšem zohlednit i nárok na paměť. Tím, že je Logstash napsán v jazyce Java, je nástroj náročnější na paměť a přiřadí si ke svému procesu celou část vymezené heap paměti, která se určuje v konfiguračním souboru Logstashu. V tomto ohledu není nástroj ideálním řešením.

Instalace je možná skrze repositáře apt nebo yum. Jsou ovšem dostupné i balíčky tar.gz, deb, zip i rpm. Instalace je tedy možná bez administrátorského přístupu. A to stáhnutím balíčku pomocí příkazu `wget`, a poté rozbalením do adresáře (Elasticsearch B.V nedatováno).

3.2.2 Fluentd

Fluentd nabízí využití vyrovnávací paměti ve formě RAM i ve formě souborů na disku, nebo v kombinaci těchto dvou. Tím může zaručit dodání záznamů i v případě, kdy destinace není dostupná. Také nabízí optimalizovaný zásuvný modul pro výstup do Elasticsearche.

Pro logback existuje open source knihovna *more-appenders*, která implementuje dva Appendery pro Fluentd. Oba je možné nainstalovat pomocí Mavenu, kdy se do *pom* souboru přidají příslušné závislosti.

Instalace je možná skrze použití RPM nebo DEB repositářů pro Linux, .dmg pro MacOS X, .msi pro Windows. Další možnosti jsou použitím nástrojů Ruby Gem nebo Chef, nebo je možná instalace přímo ze zdrojových souborů. Instalace ze zdrojových souborů ale vyžaduje další nástroje, jako je – Ruby, Rake, Gem. Druhý způsob je přes nástroj `td-agent`. Ten pouze vyžaduje přidání záznamu o `td-agent` do Yum repositáře, a poté přes příkaz `yum` tento nástroj nainstalovat. Tím se nainstalují všechny potřebné závislosti i samotný `fluentd`. Všechny způsoby vyžadují administrátorský přístup, a pokud uživatel tento přístup nemá, je téměř nemožné tento nástroj nainstalovat. (Cloud Native Computing Foundation 2019)

3.2.3 Fluent bit

Aby byl uživatel schopný nainstalovat nástroj Fluentbit, je nutné, aby jeho systém disponoval dvěma nástroji: překladač (kompilátor) pro jazyk C, nástroj CMake. CMake je

nástroj spravovaný společností pro open source software Kitware, Inc. Tento nástroj lze nainstalovat stažením balíčku zdrojových souborů, a poté spustit skript, který je v adresáři k dispozici. Instalace vyžaduje překladač, který podporuje C i C++11.

Další závislosti Fluentd se odvíjí od konkrétních zásuvných modulů. Stažení zdrojových souborů je možné přes `wget`. Pro instalaci už se využije nástroj `CMake`. Druhý způsob instalace může být přes nástroj `td-agent-bit`. `Td-agent-bit` je nástroj spravovaný společností Treasure Data, Inc. Pro instalaci tak stačí přidat záznam o `td-agent-bit` do Yum repozitáře a poté přes příkaz `yum` tento nástroj i s `fluent-bit` a všemi závislostmi nainstalovat. Pokud zařízení nedisponuje překladačem na C++ a C++11, je téměř nemožné tento nástroj nainstalovat bez administrátorských práv k systému (Eduardo Silva et al. 2019a).

3.2.4 Elasticsearch appender

Jak bylo zmíněno v kapitole 1.3.2, je možné vytvořit vlastní implementaci Appenderu. Tato možnost dala vzniku Appenderu, který je napsaný přímo pro Elasticsearch. Je volně dostupný z GitHubu na <https://github.com/internetitem/logback-elasticsearch-appender>. Tento appender umožňuje asynchronně posílat záznamy z Logbacku přímo do Elasticsearche. Tím, že je Appender implementován pomocí Logbacku, má své omezení. Omezení nastane v případě, kdy Elasticsearch není spuštěný nebo se fronta naplní nedostatečným výkonem Elasticsearche, což způsobí ztrátu záznamů.

Pro použití tohoto Appenderu v Logbacku je nutné přidat závislost do souboru *pom* v příslušném projektu. Pak už jen stačí odkázat se v souboru *logback.xml* v definici Appenderu na třídu *com.internetitem.logback.elasticsearch.ElasticsearchAppender*. Appender pro připojení na Elasticsearch používá REST API Elasticsearche „*_bulk*“. Toto rozhraní umožňuje jediným voláním API provést více operací.

- *url* – adresa pro bulk API. Pro výchozí nastavení Elasticsearch by tato adresa byla: `http://localhost:9200/_bulk`
- *index* – index, do kterého se mají odeslané záznamy přidat
- *type* – každý zaindexovaný dokument v Elasticsearchi má určitý typ mapování. Tento typ mapování určuje, jak bude dokument zaindexovaný podle předchozích dokumentů.
- *errorLoggerName* – tímto je možné připojit další logger, který bude agregovat chybové hlášky Appenderu. Obvyklou událostí, která se dostane do tohoto loggeru, je chyba v připojení k Elasticsearchi.
- *Properties* – toto je předek k elementům property, který existuje pro lepší orientaci v konfiguraci Appenderu
- *Property* – zde je možné pomocí *logback patterns* (viz kapitola 1.3.3) definovat, jaké informace se budou přidávat do záznamu. Vždy je nutné *value* (hodnotu), kam se vkládá jeden z *patterns*, a jméno, pod kterým bude poté možné najít hodnotu v Elasticsearchi.
- *Headers* – Elasticsearch přijímá data ve formátu JSON. Je proto nutné ještě doplnit přenos dat o hlavičku, která dodá informaci o formátu přenesených dat. (Adam Batkin 2019)

3.2.5 Porovnání využití paměti

Hodnoty využití paměti se budou porovnávat s počátečním stavem paměti, tzn. před spuštěním log shipperu. Hodnota počátečního stavu využití paměti je: 147 MB.

Tab. 3 Záznam využití paměti jednotlivými log shippery (vlastní zpracování)

Log shipper	Po spuštění	Rozdíl
Logstash	760 MB	613 MB
Fluentd (td-agent)	309 MB	162 MB
Fluent bit (td-agent-bit)	156 MB	9 MB

Elasticsearch Appender je součástí Logbacku, a tak není samostatnou částí řetězce, která by pracovala se záznamy. Proto je možné ho vynechat z porovnání využití paměti.

Z pohledu využití paměti je tedy nejvhodnějším řešením Elasticsearch Appender. Problémy mohou nastat v případě, kdy instance Elasticsearche nebude dostatečně výkonná na zpracování záznamů, a Logback tak bude tvořit frontu v paměti. Tomu by se dalo předejít navyšováním paměti pro Elasticsearch, dokud problémy nezmizí. Frontu bude tvořit i v případě, kdy instance Elasticsearche není spuštěná. Pokud fronta dosáhne velikosti 200 MB, další záznamy se nebudou přidávat do fronty a budou vyřazeny (Adam Batkin 2019). Může tak potenciálně dojít ke ztrátě záznamů. Tato skutečnost by se zaznamenala v Error loggeru, který tak musí být nakonfigurovaný v Elasticsearch Appenderu. Na druhou stranu, denně je vytvořeno přibližně 3,2 GB záznamů, to je 2,25 MB za minutu, a to znamená, že by Elasticsearch nesměl být spuštěný déle než hodinu. Velikost fronty lze naštěstí upravit, musí se ale vzít v úvahu paměť procesu logování, kterému nesmí dojít paměť.

3.2.6 Porovnání řešení

Z pohledu paměti je nejvhodnější využít implementaci Appenderu pro Elasticsearch. Další způsob porovnání může být z pohledu jednoduchosti nastavení či instalace. Jak je vidět v tab. 4, k instalaci Fluentd a Fluentbitu je nutné mít administrátorský přístup k systému. To může v některých případech být problém, neboť není vždy možné tento přístup zajistit.

Tab. 4 Nutnost administrátorských práv k instalaci log shipperu (vlastní zpracování)

Log shipper	Administrátorská práva
Logstash	Ne
Fluentd (td-agent)	Ano
Fluent bit (td-agent-bit)	Ano
Elasticsearch Appender	Ne

Pokud by výběr pokračoval v jednoduchosti instalace nebo nastavení řešení, možnosti by byly dvě: Logstash, Elasticsearch Appender. Bohužel pro Logstash, musel by se nakonfigurovat výstup z Logbacku pro Logstash, a poté ještě nastavit samotný Logstash. Přitom u Elasticsearch Appenderu jen stačí správně nastavit samotný Appender a řešení je hotové. V tomto ohledu tedy také vítězí Elasticsearch Appender.

Pokud by se abstrahovalo od administrátorských práv a jednoduchosti instalace, byl by jednoznačný rozdíl v nabízených funkcích, možnostech filtrů, výstupů nebo vstupů a nakonec i v efektivnosti využití zdrojů – RAM. Protože jsou funkce Fluentd a Logstash velmi podobné, rozhodování by bylo ve prospěch Fluentd právě kvůli využití paměti. Pokud by složitější agregace logových záznamů nebylo předmětem výběru, dalo by se uvažovat i o Fluentbitu, jakožto konkurentovi pro Fluentd.

3.3 Grafické rozhraní pro vyhledávání

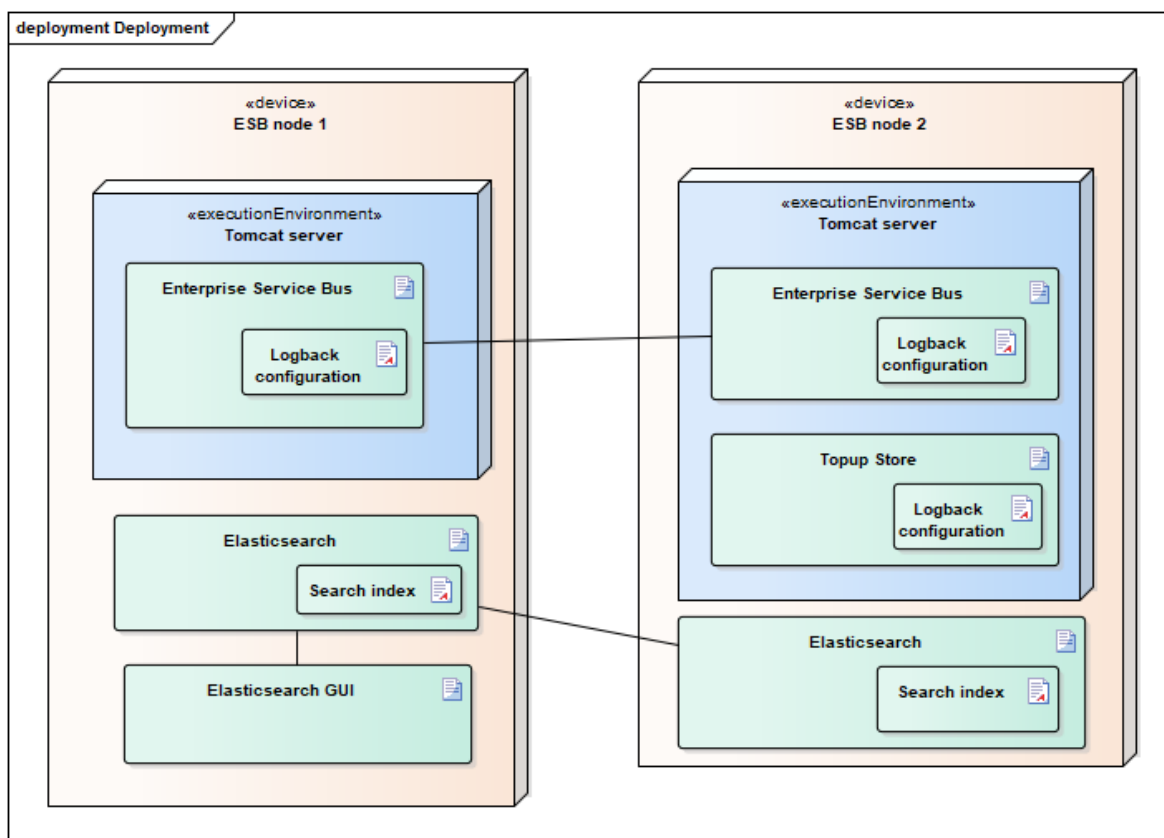
Pro Elasticsearch existuje optimalizovaný nástroj Kibana, který společně s Logstashem tvoří tzv. ELK Stack. Kibanu není třeba při čerstvé instalaci nijak nastavovat a funguje s Elasticsearchem ve výchozím nastavení. Proto se nabízí jako první potenciální řešení.

Alternativou by mohl být jeden ze zdarma nástrojů – Grafana, Graylog, Prometheus. Grafana je spíše nástroj pro vizualizaci a monitorování systémů a jako takový ani nenabízí full-textové vyhledávání (Asaf Yigal 2018). Graylog představuje kompletní řešení podobné kombinaci Elasticsearchi a Logstashi, často je instalován spolu s Grafanou. Prometheus je nástroj, který je spíše využíván pro metriky než vyhledávání. Díky přehlednému porovnání nástrojů Graylog, Prometheus a Kibana pomocí volně dostupného nástroje společnosti StackShare, Inc. (<https://stackshare.io/stackups/graylog-vs-kibana-vs-prometheus>), je možné usoudit, že Kibana je opravdu nejvhodnějším nástrojem.

3.4 Návrh řešení

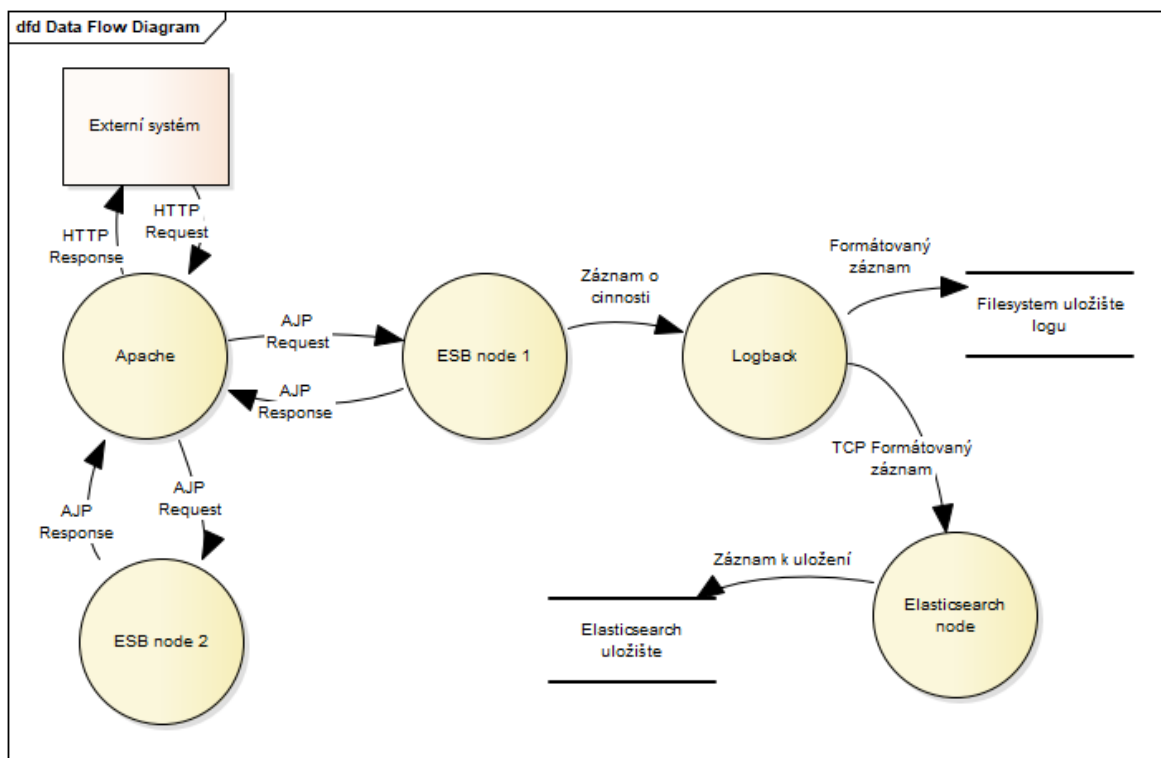
Pro indexaci dokumentů bude využitý nástroj Elasticsearch, pro přepravu záznamů z aplikace ESB bude vyžítý Elasticsearch Appender pro Logback, a pro grafické rozhraní a vyhledávání bude použitý nástroj Kibana.

Deployment diagram návrhu řešení je graficky znázorněno na obrázku 3. Existují dvě zařízení pojmenovaná ESB node 1 a 2. Na každém zařízení běží Tomcat server, jakožto prostředí pro Enterprise Service Bus a jeho další aplikace. Mezi ESB aplikacemi probíhá komunikace, konkrétně o rozdělování požadavků a zátěže. Každá aplikace ESB disponuje svou vlastní konfigurací Logbacku. Na obou nodech poběží ve svém vlastním prostředí instance Elasticsearche. Protože jsou instance Elasticsearche v Clusteru, budou mezi sebou komunikovat. Každá instance Elasticsearche bude mít své Indexy a Shardy. Na jednom zařízení bude běžet Elasticsearch GUI – Kibana. Kibana bude komunikovat jen s jednou instancí Elasticsearche. Elasticsearch bude pro odpověď na dotaz komunikovat s ostatními instancemi.



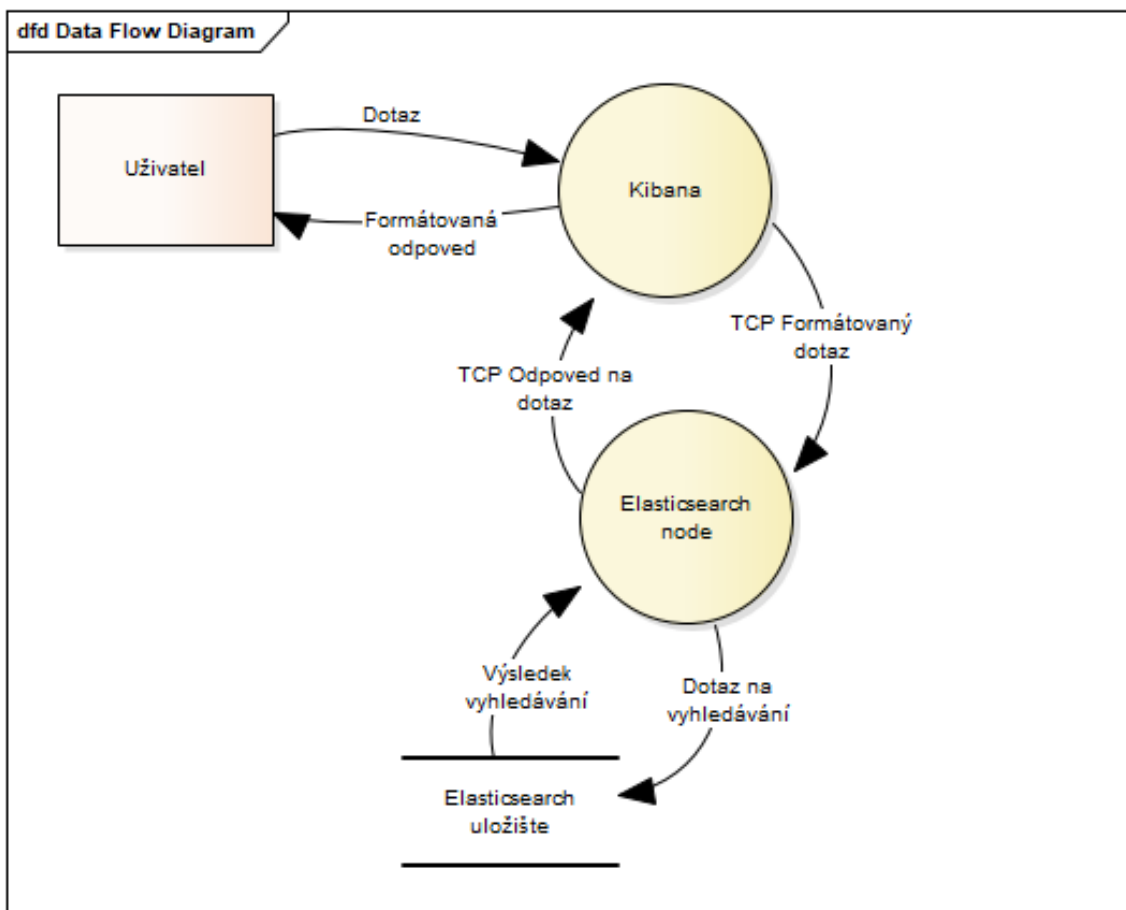
Obr. 3 Deployment diagram návrhu řešení (vlastní zpracování)

Pro přehledné zobrazení komunikace mezi komponentami je využito znázornění pomocí Dataflow diagramu. Na obrázku 4 je možné vidět datový tok při vytvoření nového záznamu a jeho následné zaindexování nástrojem Elasticsearch. Z externího systému přijde požadavek na Enterprise Service Bus. Apache server rozhodne, kterému nodu ESB tento požadavek přiřadí. Po přiřazení požadavku se požadavek dostane k samotnému ESB. Ten na požadavek odpoví, a odpověď projde opět Apache serverem, který se postará o doručení odpovědi. Při zpracovávání požadavku dochází k vytváření záznamů o probíhající činnosti ESB. Tyto záznamy se přes Logback ukládají do úložiště zařízení a zároveň se posílají na rozhraní Elasticsearche. Elasticsearch požadavek přijme, uloží a zaindexuje.



Obr. 4 Dataflow diagram: Reprezentace datového toku mezi komponentami při vytvoření a indexování nových záznamů (vlastní zpracování)

Na obrázku 5 je znázorněná komunikace mezi komponentami při poslání dotazu na vyhledávání v Elasticsearchi. Uživatel pomocí vyhledávacího pole Kibany zadá svůj dotaz. Kibana dotaz přetransformuje do formátu pro komunikaci s Elasticsearchem. Tento dotaz přes TCP pošle k Elasticsearchi, který dotaz přijme. Elasticsearch poté vyhledá ve svém úložišti odpovídající záznam a odešle ho zpět Kibaně. Kibana odpověď, která je ve formátu JSON, transformuje do čitelné podoby a zobrazí uživateli.



Obr. 5 Dataflow diagram: Reprezentace datového toku mezi komponentami při vyhledávání dokumentů v Elasticsearchi (vlastní zpracování)

4 Implementace navrženého řešení

Pokud existuje návrh řešení, který odpovídá požadavkům na funkcionalitu a omezením prostředí, je možné toto řešení implementovat. Každý z nástrojů se musí určitým způsobem nainstalovat a nakonfigurovat. Poté se musí i otestovat správnost nastavení jednotlivých komponent. Po splnění těchto dílčích úkolů je možné začít nástroje používat. Tato kapitola provede čtenáře instalací, konfigurací a na závěr kapitoly i samotným použitím navrženého řešení.

4.1 Instalace

Instalace Appenderu pro Logback je jednoduchá, jen se do závislostí projektu Maven přidá knihovna Elasticsearch Appender. Poté už jen zbývá nakonfigurovat správné využití Appenderu v konfiguračním souboru *logback.xml*.

Instalace Elasticsearche probíhá pomocí volně dostupných balíčků tar.gz. Bezplatná verze obsahuje veškerou funkcionalitu, která je potřeba k řešení. Pomocí příkazů *wget* je možné stáhnout balíček obsahující všechny soubory Elasticsearche. Je vhodné dodržovat bezpečnostní doporučení, a proto je třeba stáhnout pomocí *wget* i kontrolní součet SHA a poté porovnat tento součet se součtem ze stáhnutého balíčku. Pokud se shodují, zbývá už jen rozbalit tar.gz pomocí příkazu *tar*. Balíčky i součty jsou volně dostupné z adres, které se nachází v oficiálním návodu na instalaci Elasticsearche a jsou aktualizované na aktuální verzi, na <https://www.elastic.co/guide/en/elasticsearch/reference/current/zip-targz.html>. Před spuštěním je nutné upravit konfiguraci serveru, na kterém instance Elasticsearche poběží, a poté i samotný Elasticsearch.

Instalace Kibany je obdobná, jako u Elasticsearche. Jen se liší názvy balíčků a adresa, kde je možné aktualizované odkazy na správné verze najít. Pro Kibanu je to oficiální návod k instalaci dostupný na <https://www.elastic.co/guide/en/kibana/current/targz.html>.

4.2 Konfigurace serveru

Po instalaci je třeba upravit konfiguraci každého serveru, na kterém bude běžet instance Elasticsearche. Tyto úpravy vyžadují administrátorská práva. Pokud nemá uživatel dostatečné oprávnění, je nutné, aby požádal správce serveru o provedení těchto změn.

Pokaždé, když je v systému otevřen nějaký soubor, operační systém si vytvoří záznam, který soubor reprezentuje a obsahuje informace o daném souboru. Tomuto záznamu se říká deskriptor souboru. Elasticsearch pracuje s mnoha soubory najednou, a proto využívá mnoho deskriptorů. Na zařízení je tedy nutné nastavit vyšší hodnotu maximálního počtu deskriptorů minimálně na 65536. Nastavení lze upravit pomocí příkazu *ulimit -n 65536* nebo lze upravit soubor */etc/security/limits.conf*, a přidat dva řádky - jeden ve formátu *[uživatel]*

`hard nofile 65536` a druhý podobně, jen místo typu `hard` je typ `soft`. Pokud bychom chtěli nastavení upravit pro všechny uživatele, je možné za uživatele dosadit `*`.

Jeden node Elasticsearch obsahuje několik fondů vláken, aby mohl lépe spravovat využití paměti jednotlivými vlákny. Elasticsearch vykoná požadavek tím, že požadavek rozdělí do několika fází a tyto fáze rozdělí do různých fondů. Protože existuje více fondů vláken pro různé účely, je nutné, aby mohl Elasticsearch vytvářet mnoho vláken, které do těchto fondů přiřadí. Na Linuxu existuje kontrola, jestli má daný proces oprávnění tyto vlákna vytvářet. Proto se musí upravit nastavení maximálního počtu vláken, které může proces Elasticsearch vytvořit. A to na hodnotu minimálně 4096 vláken. Nastavení lze upravit jednoduše pomocí příkazu `ulimit -u 4096`, nebo obdobně jako u deskriptorů lze upravit soubor `/etc/security/limits.conf`, do něhož se přidají dva řádky - jeden ve formátu `[uživatel] hard nproc 4096` a druhý místo typu `hard` s typem `soft`.

Elasticsearch využívá úložiště `mmapfs` pro ukládání svých indexů. `Mmapfs` ukládá index shardů tím, že mapuje soubor do paměti. Mapování do paměti využívá tak velkou část virtuálního místa, jako je velikost mapovaného souboru. V případě manuální instalace ze souborů `zip` nebo `tar.gz`, a ne z repositáře `RPM` nebo `Debian`, je nutné zvýšit maximální počet mapování souborů v paměti, který by jinak byl příliš malý. A to na hodnotu 262144. Při instalaci z balíčků z jednoho z repositářů je toto nastavení upraveno automaticky. Proces Elasticsearch by mohl bez tohoto nastavení vyhazovat výjimky z nedostatku paměti. Toto nastavení lze upravit příkazem `sysctl -w vm.max_map_count=262144`.

Jak už bylo zmíněno v kapitole 1.4.1.2, Elasticsearch při vytvoření Clusteru komunikuje. Tato komunikace probíhá přes `TCP` port 9300. Proto je třeba upravit nastavení firewallu. Toto nastavení je třeba provést u každého serveru, na kterém běží node Elasticsearch, který má být v clusteru. Nastavení musí umožnit komunikaci mezi servery na tomto portu. U firewallu je třeba ještě jednoho nastavení. Na serveru, na kterém běží nástroj Kibana, musí být umožněn přístup přes port 5601 zvenčí, nebo z určité domény. Navíc je třeba zajistit přístup mezi serverem, na kterém běží Kibana, a serverem, na kterém běží node Elasticsearch. Tento přístup je třeba umožnit na portu `TCP` 9200, na kterém Elasticsearch komunikuje skrze své `REST` rozhraní.

Konfigurační soubory umožňují konkrétní adresy a porty upravovat. Je tak možné nedržet se výchozích nastavení a nastavit vlastní hodnoty. Ať už bude nastavení jakékoliv, je třeba umožnit komunikaci nodům Elasticsearch v clusteru, a komunikaci mezi jedním nodem a Kibanou. V případě přístupu na Kibanu mimo server, na kterém běží, je třeba ještě umožnit tento přístup.

4.2.1 Konfigurace Elasticsearch

Konfigurace Elasticsearch byla vysvětlena v kapitole 1.4.1.4. V této části už budou jednotlivé hodnoty nastavení zadány.


```
cluster.name: cleverbus
node.name: esbdev02
bootstrap.system_call_filter: false
path.data: /srv/cleverbus/home/elasticsearch-6.6.0/data/
path.logs: /srv/cleverbus/home/elasticsearch-6.6.0/logs/
network.host: 192.168.0.24
http.port: 9200
discovery.zen.ping.unicast.hosts: ["192.168.0.24:9300", "192.168.0.23:9300"]
discovery.zen.minimum_master_nodes: 2
```

Výpis z kódu 2: Ukázka konfigurace Elasticsearche v souboru `elasticsearch.yml`

Kernel CentOS 6 nepodporuje *seccomp* (viz kapitola 1.4.1.4), tudíž je nutné kontrolu izolace procesu Elasticsearch vypnout. Pro přehlednost umístění souborů, které Elasticsearch vytváří, jsou umístěna data a logové záznamy přímo do složky Elasticsearche. Takto je pak možné najít nastavení, data i logy pod jednou složkou.

TCP port 9200 byl dostupný. Pro větší přehlednost byla hodnota nastavena explicitně, i když na stejnou hodnotu jako je výchozí. Síťové rozhraní, na kterém Elasticsearch běží, bylo nastaveno na jedno z dostupných na serveru - `192.168.0.24`.

Pro vyhledávání nodů do clusteru jsou pak zadány adresy, na kterých nody Elasticsearche běží. První se uvádí adresa lokálního nodu, a poté následují další adresy. V tomto případě jde o adresu druhého serveru.

Nastavení nodu na druhém serveru je velmi podobné. Co se liší, je nastavení síťového rozhraní a pořadí v nastavení adres pro vyhledávání nodů v clusteru.

V konfiguračním souboru *jvm.options* jsou změněny jen dvě hodnoty, oproti výchozímu souboru.

```
-Xms1g
-Xmx1g
```

Výpis z kódu 3: Ukázka konfigurace Elasticsearche v souboru `jvm.options`

Jedná se o hodnoty minimální a maximální hodnoty místa pro paměť heap, nastavené na 1 GB. Podle doporučení, zmíněné v kapitole 1.4.1.4, je velikost minima i maxima nastavená na stejné hodnoty.

4.2.2 POM

Aby se mohl využít Elasticsearch Appender, je nutné přidat potřebné soubory z knihovny do projektu. K tomuto účelu se použije soubor *pom.xml*, kterým se přikáže Mavenu tyto soubory stáhnout. Konkrétně se do souboru musí přidat následující řádky:

```
<dependency>
  <groupId>com.internetitem</groupId>
  <artifactId>logback-elasticsearch-appender</artifactId>
  <version>1.6</version>
</dependency>
```

Výpis z kódu 4: Ukázka konfigurace Mavenu v souboru pom.xml

Pokud jsou v projektu přítomny soubory z knihoven Logbacku nebo SLF4J, není nutné tyto dependence znovu přidávat. Pokud však v projektu nejsou, je nutné je přidat. V případě ESB jsou tyto knihovny už používány, stačí proto jen přidat Elasticsearch Appender. Poté už se může konfigurovat samostatný soubor *logback.xml*, ve kterém se specifikuje užití Appenderu.

4.2.3 Konfigurace Logbacku

Pro konfiguraci Logbacku je využit souboru *logback.xml*. Ten pro ESB vypadá následovně.

```
<appender name="ELASTIC"
class="com.internetitem.logback.elasticsearch.ElasticsearchAppender">
  <url>http://192.168.0.24:9200/_bulk</url>
  <index>logs-%date{yyyy-MM-dd}</index>
  <type>cleverbus-2.7.0</type>
  <errorLoggerName>ELASTIC-ERROR</errorLoggerName>
  <properties>
    <property>
      <name>logger</name>
      <value>%logger{36}</value>
    </property>
    .
    .
    .
  </properties>
  <headers>
    <header>
      <name>Content-Type</name>
      <value>application/json</value>
    </header>
  </headers>
</appender>
```

Výpis z kódu 5: Ukázka konfigurace Elasticsearch Appenderu v Logbacku v souboru logback.xml

V kapitole 1.3.2 byl Appender popsán. Zde je mu přiřazeno jméno „ELASTIC“ a je odkázán na třídu s konkrétní implementací (viz kapitola 3.2.4). Poté následují jednotlivé části nastavení konkrétního Appenderu. Nastavení *url* se Appender odkazuje na adresu, na které běží Elasticsearch. Je nutno poznamenat, že jde o adresu s příponou „_bulk“, na které je REST aplikační rozhraní Elasticsearche pro zpracování více operací v jednom volání na rozhraní. Záznamy se mají přiřadit do indexu, který začíná *logs* a poté následuje datum. Je

zde i nastavený logger pro případ, kdyby se nepodařilo doručit logové záznamy do Elasticsearche. V souboru poté následuje sekce *properties*. Každá *property* má své jméno a hodnotu. Celé properties jsou následující:

- `ServerId` – `serverId` se nenachází ve specifikaci Appenderu, ale je to proměnná prostředí, ve kterém ESB běží. Jedná se o jméno serveru, na kterém je aplikace ESB spuštěna.
- Dále `level`, `thread`, `xEx{7}`, `logger{36}` (viz kapitola 1.3.3)

V případě problémů s Appenderem se využije Logger (viz kapitola 1.3.1) *elastic-error*, který o takových událostech záznamy uloží. Nenahradí funkcionalitu Elasticsearche, ani funkcionalitu log shipperu, který by umožnil záznamy dočasně uložit, ale pouze uloží záznam o události, který říká, jaká chyba se vyskytla. Nejčastěji to pak může být právě nemožnost připojení k instanci Elasticsearche, pokud Elasticsearch proces neběží nebo jinak není schopen přijmout požadavky. V následující ukázce je úryvek ze souboru *logback.xml*, který určuje chování *elastic-error* loggeru.

```
<logger name="ELASTIC-ERROR" level="INFO" additivity="false">
  <appender name="ELASTIC-FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
    <file>/srv/cleverbus/logs/j2ee/elastic-error.${nodeid}.log</file>
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
      <fileNamePattern>/srv/cleverbus/logs/j2ee/elastic-
error.${nodeid}.log.%d{yyyy-MM-dd}_%i.log</fileNamePattern>
      <maxHistory>15</maxHistory>
      <timeBasedFileNamingAndTriggeringPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">
        <maxFileSize>10MB</maxFileSize>
      </timeBasedFileNamingAndTriggeringPolicy>
    </rollingPolicy>
    <encoder>
      <pattern>%d{ISO8601} - %msg%n</pattern>
    </encoder>
  </appender>
</logger>
```

Výpis z kódu 6: Ukázka konfigurace loggeru ELASTIC-ERROR v Logbacku v souboru logback.xml

Záznamy se ukládají do souboru, který začíná *elastic-error* a pokračuje s názvem nodu, který záznam vyprodukoval. Záznamy se drží po dobu 15 dní a pokaždé, když dosáhnout velikosti 10 MB se zkomprimují. Obsah záznamů má podobu data, kdy se událost stala, ve formátu ISO 8601, a poté následuje zpráva záznamu. Ukázka záznamu z toho souboru:

```
2019-03-25 10:51:43,844 - Failed to send events to Elasticsearch: Connection refused
```

Výpis z kódu 7: Ukázka chybové hlášky z Elasticsearch Appenderu

4.2.4 Konfigurace Kibany

Konfigurační soubor *kibana.yml* se nachází v podsložce *config*. Budou upraveny následující hodnoty:

- `Server.host` - Zde je třeba vyplnit adresu síťového rozhraní, pro které je nakonfigurovaný přístup pro komunikaci s Elasticsearchem. Na této adrese pak instance Kibany poběží.
- `Elasticsearch.hosts` – Zde se vyplní adresa nodu Elasticsearche, který chceme pro dotazování využívat.
- `Elasticsearch.pingTimeout` – Hodnota v ms, jak dlouho bude Kibana čekat na odpověď od Elasticsearche při zahajování komunikace.
- `Elasticsearch.requestTimeout` – Hodnota v ms, jak dlouho bude Kibana čekat na odpověď na dotaz od Elasticsearche.
- `Logging.dest` – Zde se může nastavit cesta k souboru, do kterého má Kibana ukládat záznamy o své aktivitě.
- `Logging.quiet` – Hodnota `true`, nebo `false`. V případě `true` bude Kibana ukládat záznamy jen o erorech, jiné záznamy vynechá.

4.3 Spuštění

Po konfiguraci je možné otestovat správnost nastavení spuštěním jednotlivých komponent.

4.3.1 Elasticsearch

Nejdříve je nutné spustit Elasticsearch, který musí být připraven přijímat data a zároveň musí povolit připojení Kibaně. Spuštění se provede z adresáře Elasticsearche pomocí příkazu `./bin/Elasticsearch -d -p pid`. Přepínač „d“ přikáže procesu spustit se v pozadí, „p“ pak uloží ID procesu (pid). Ukončit proces lze následujícím způsobem:

1. Jako uživatel, který spustil Elasticsearch, zadat příkaz `jps`, který zobrazí spuštěná JVM prostředí.
2. Najít proces Elasticsearch a zapsat si nebo zapamatovat příslušné ID procesu.
3. Pomocí příkazu `kill -SIGTERM` s parametrem číslo procesu Elasticsearche se Elasticsearch ukončí.

Ověření, že nástroj Elasticsearch běží, je možné několika způsoby. Prvním způsobem je příkazem `curl`. Je nutné dosadit IP adresu a port, na kterém má Elasticsearch běžet. Příklad: `curl -XGET '127.0.0.1:9200'`. Pokud je správně vyplněná IP adresa i port, výstup z tohoto příkazu by měl vypadat přibližně takto:

```
{
  "name" : "esbdev01",
  "cluster_name" : "cleverbus",
  "cluster_uuid" : "0oue6zQ2QNaGaFo0JghxWw",
  "version" : {
    "number" : "6.6.0",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "a9861f4",
    "build_date" : "2019-01-24T11:27:09.439740Z",
    "build_snapshot" : false,
    "lucene_version" : "7.6.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Výpis z kódu 8: Ukázka výsledku na dotaz na Elasticsearch pomocí příkazu curl

Druhým způsobem může být pomocí příkazu `top`, nebo již zmíněným `jps`. V obou případech je nutné proces Elasticsearch vyhledat.

4.3.2 Kibana

Po úspěšném spuštění Elasticsearche je možné spustit Kibanu. Kibana bohužel nemá spouštění v pozadí zabudované přímo v sobě, a musí se tak využít funkce spouštění procesů v pozadí, kterou nabízí systém Linux. Příkaz ke spuštění pak může ve složce Kibana vypadat takto: `./bin/kibana & disown`

Parametr „&“ přikáže procesu spustit se v pozadí. Parametr „disown“ pak upraví vlastnosti procesu tak, aby se při ukončení připojení k terminálu přes ssh neukončil i daný proces.

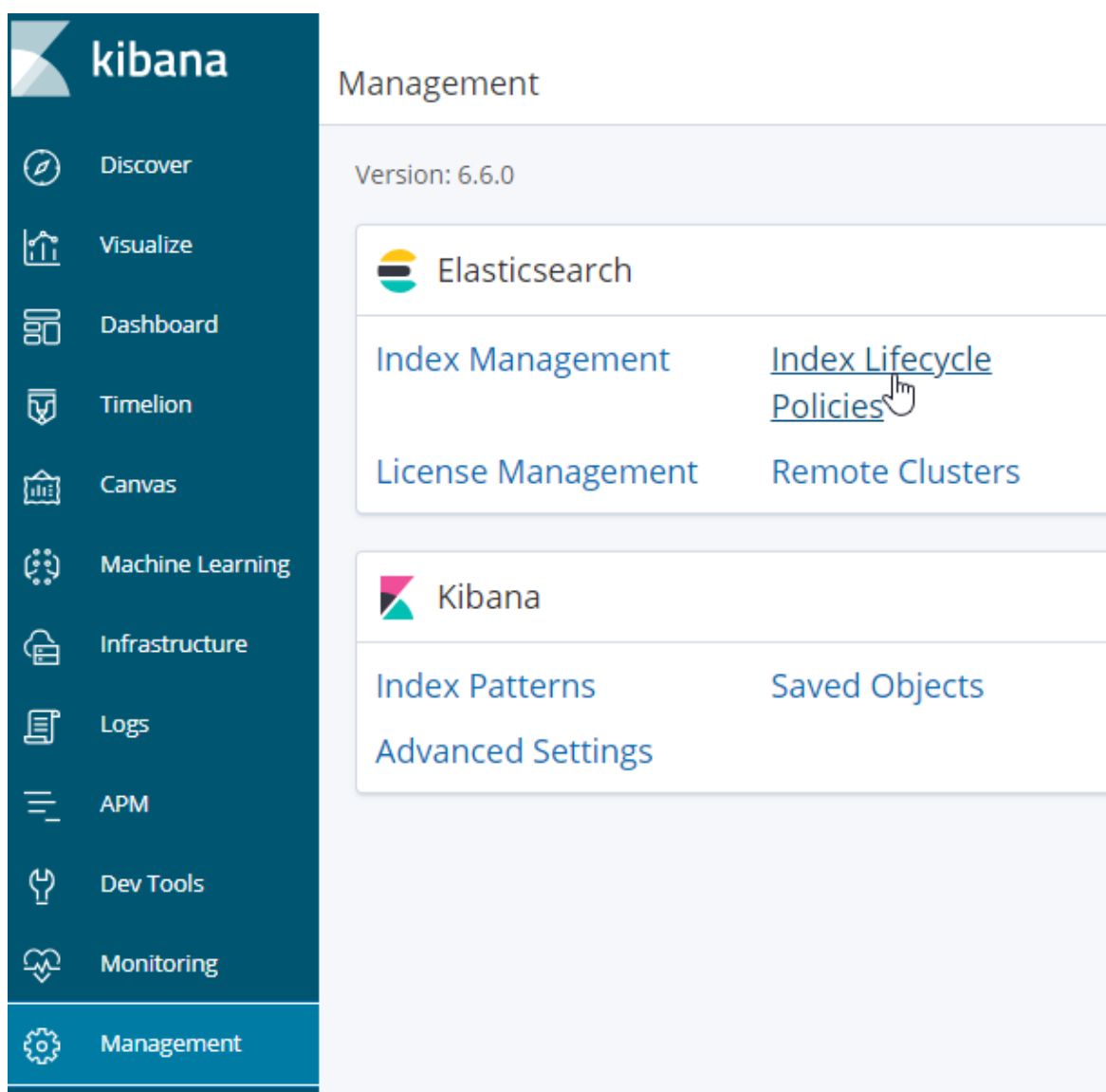
Pokud je nastavený přístup k IP adrese na portu Kibana, může se jednoduše ověřit její úspěšné spuštění. Buď přímo zadáním adresy Kibany do prohlížeče a načtením domovské stránky Kibany nebo opět pomocí příkazu `top` a vyhledání procesu Kibany.

4.4 Doplnková konfigurace

Takto nastavený Elasticsearch a Kibana budou fungovat. Pro zajištění dlouhodobého provozu je vhodné upravit ještě nastavení správy indexů. To se dá udělat pomocí grafického rozhraní Kibany nebo kombinací příkazů `curl`. Jednodušší a intuitivnější nastavení je pomocí grafického rozhraní.

4.4.1 Politika životního cyklu indexů

Pod sekcí managementu v Kibaně je kategorie politika životního cyklu indexů (viz obr. 6). Životní cyklus indexů již byl zmíněn v kapitole 1.4.1.5. Posílání logových záznamů z Logbacku je nastaveno tak, aby se záznamy posílaly do indexu, který v sobě obsahuje informaci o datumu (viz kapitola 4.2.3). Tímto nastavením se zaručí každý den nový index. Není tedy třeba nastavovat denní vytváření indexů. Je ale vhodné změnit životní cyklus indexů starších než jeden den do stavu *warm* (viz kapitola 1.4.1.5). V těchto indexech se bude vyhledávat, ale už se tam nebudou přidávat další záznamy. Je tedy možné těmto indexům snížit počet shardů. Nové shardy se automaticky přejmenují. Nový název nese prefix „shrink-“.



Obr. 6 Menu pro upravení politik životního cyklu indexů.

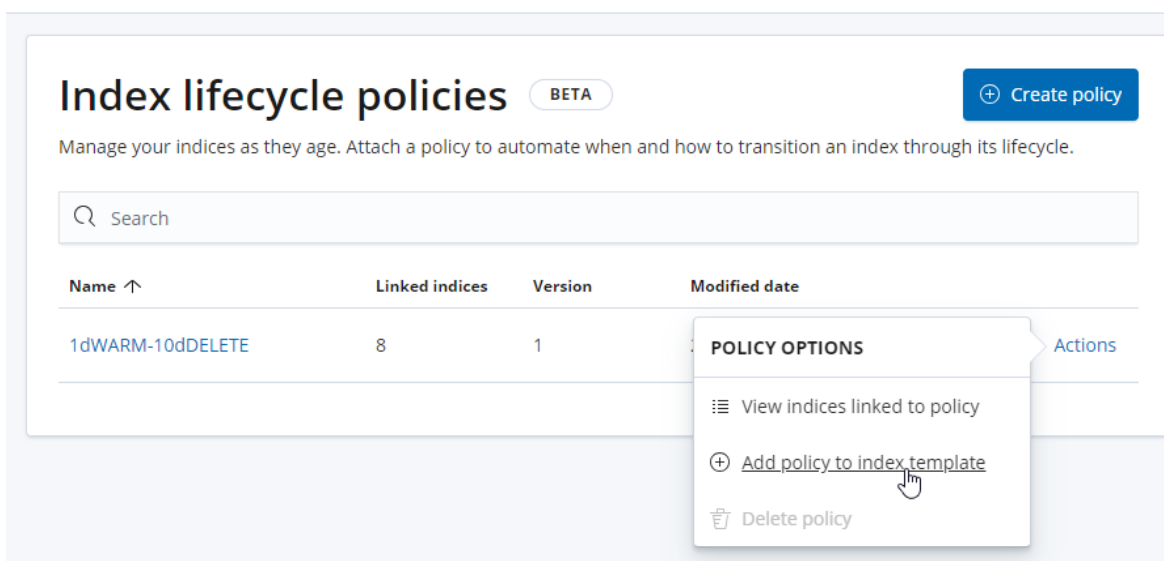
Aby se nehromadily staré indexy do nekonečna, je vhodné nastavit podmínku, kdy se indexy budou mazat. V tomto nastavení je možné automaticky mazat indexy, které jsou starší než zvolený počet dní, a toto nastavení je také využito.

Politiku životního cyklu lze nastavit pro šablonu indexu. Indexy vytvořené použitím této šablony pak budou mít automaticky nastavenou konkrétní politiku. Musí se tedy ještě vytvořit šablona pro indexy. To lze provést v příkazové řádce příkazem curl a využít tak aplikační rozhraní Elasticsearche.

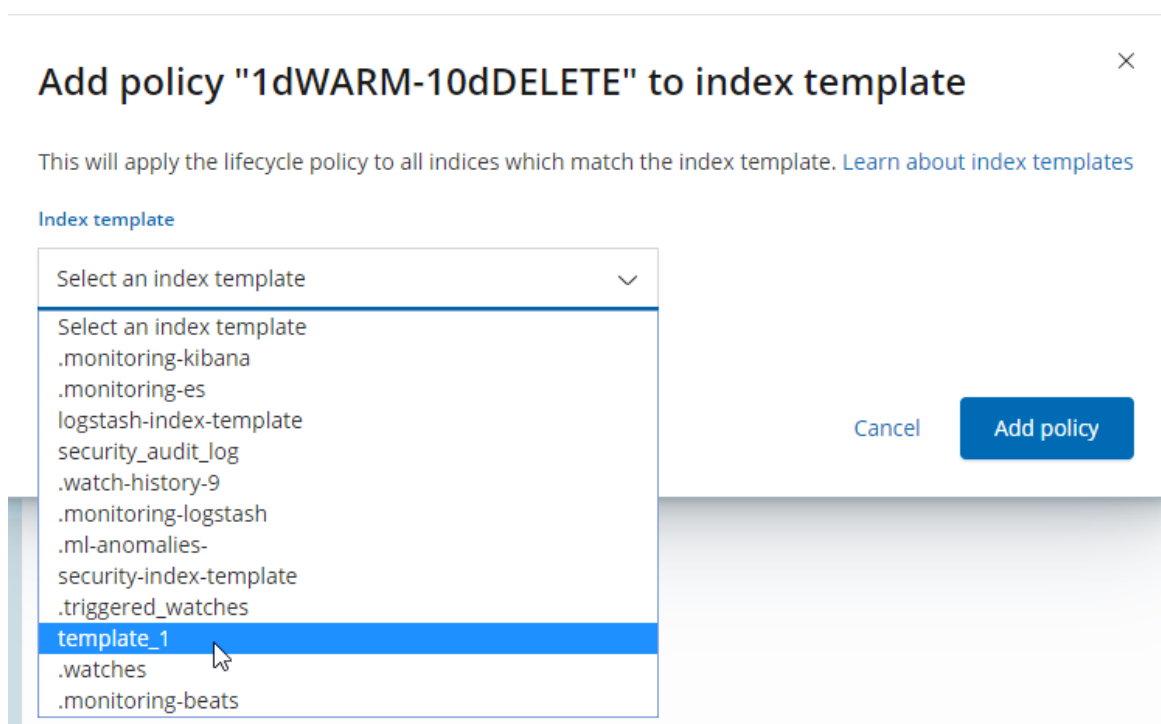
```
curl -XPUT "192.168.0.23:9200/_template/template_1" -H 'Content-Type: application/json' -d'
{
  "index_patterns": ["*logs*"],
  "settings": {
    "number_of_shards": 5
  },
  "mappings": {...
    "level": {
      "type": "text",
      "fields": {
        "keyword": {
          "type": "keyword",
          "ignore_above": 256
        }
      }
    }
  }
}
```

Výpis z kódu 9: Úryvek z příkazu curl k vytvoření nové šablony s mapováním.

Tento příkaz vytvoří novou šablonu `template_1`, která se bude aplikovat pro všechny indexy, které obsahují slovo „logs“. Tímto zajistíme, že se bude nastavená politika aplikovat i na indexy, které začínají slovem „shrink“. Další nastavení obsahuje počet shardů, a poté už jednotlivé mapování dat. Pro demonstrativní účely stačí pouze úryvek z nastavení mapování. Tato část mapuje data se jménem „level“, typu „text“. Dodatečné nastavení keyword pak přikáže Elasticsearchi chovat se k tomuto poli jako ke klíčovému slovu. Tím ho bude zahrnovat pro filtry ve vyhledávání. Další nastavení „ignore_above“ jen říká, aby se cokoli nad 256 znaků ignorovalo. Teď už jen zbývá přiřadit vytvořenou šablonu k politice. Nejdříve se vybere správná politika, poté se u tlačítka „Actions“ vybere z menu „Přidat politiku k šabloně“ (viz obr. 7), a na závěr se vybere šablona, u které se má politika projevit, a výběr se potvrdí (viz obr. 8).



Obr. 7 Přidání nové politiky k šabloně.

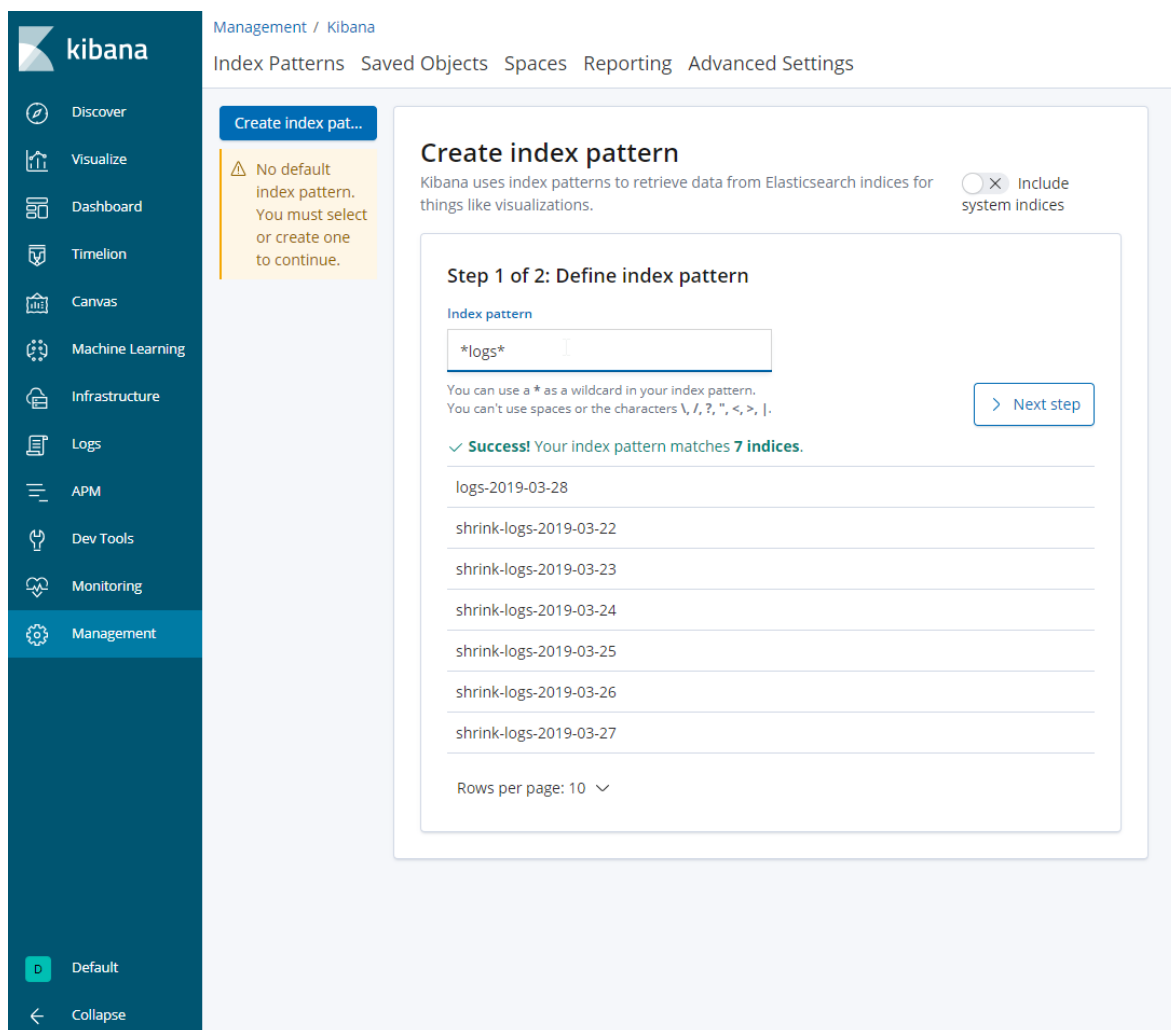


Obr. 8 Vybrání správné šablony k přiřazení k politice

4.4.2 Index patterns

Pro umožnění vyhledávání, je nutné ještě nakonfigurovat vzor indexů, ve kterých se má vyhledávání provádět. To lze jednoduše nastavit pomocí Kibany. Nastavení se nachází opět pod managementem, tentokrát však pod možností „Index Patterns“. Kibana automaticky donutí uživatele zadat výraz, kterým se vybere aspoň jeden index. Ve výrazu je možné použít zástupný znak „*“, který představuje jakoukoliv kombinaci znaků. Pomocí zástupného

znaku lze vybrat vzorec „*logs*“ vybrat všechny indexy, které obsahují slovo „logs“ (viz obr. 9). Vzhledem k nastavení Logbacku a politiky životního cyklu indexů se tímto vzorem vyberou všechny vytvořené indexy. Po zadání vzoru, který vyhovuje alespoň jednomu indexu, následuje další krok. V dalším kroku se vybírá pole, podle kterého se mají třídit záznamy podle času. Tento krok je možné přeskóčit, ale uživatel se tím vzdává možnosti sledovat, filtrovat nebo jinak pracovat se záznamy v časovém horizontu. Není tedy doporučené tento krok přeskakovat. Po vybrání časového pole se pouze klikne na „Create index pattern“, čímž se vzor vytvoří (viz obr. 10).



Obr. 9 Vytváření vzoru pro indexy, ve kterých má vyhledávání probíhat.

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

☐ Include system indices

Step 2 of 2: Configure settings

You've defined ***logs*** as your index pattern. Now you can specify some settings before we create it.

Time Filter field name

Refresh

@timestamp

@timestamp

I don't want to use the Time Filter

> Show advanced options

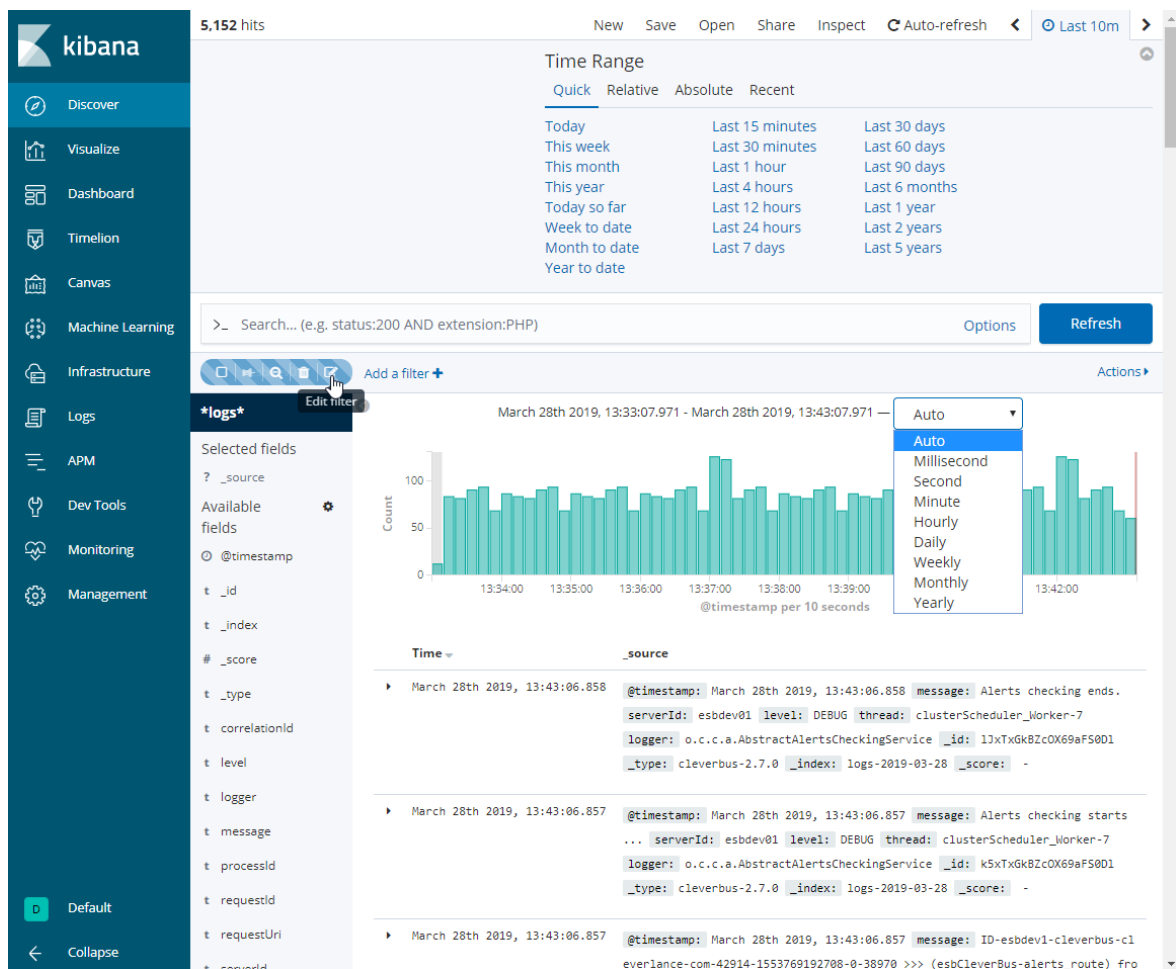
< Back

Create index pattern

Obr. 10 Výběr časového pole indexu.

4.5 Použití

Po dodatečné konfiguraci je možné začít řešení používat. Vyhledávání je v záložce „Discover“. K vyhledávání slouží textové pole (viz obr. 11). Nalevo jsou jednotlivá indexovaná pole, podle kterých je možné filtrovat vyhledávání. Nad nimi je aktuální vzor, který se pro vyhledávání aplikuje. Filtry lze přidávat, odebírat, aktivovat či deaktivovat, upravit. Filtry lze i kombinovat, tím se pak může uživatel dopracovat ke komplikovanějším dotazům. V horní části je možnost časového ohraničení. To nabízí předepsané možnosti jako je např. dnes, tento týden, posledních 15 minut apod., nebo je možné nastavit ohraničení absolutně či relativně. Absolutní ohraničení je dle data v kalendáři a hodin, relativní lze pak nastavovat jako časové omezení od do. Pod vyhledávacím polem je grafické znázornění počtu zaindexovaných dokumentů za vybrané období. Detail zobrazení lze upravit z detailnějšího pohledu na méně detailní – z pohledu na počet záznamů za vteřinu oproti počtu záznamů za hodinu.



Obr. 11 Zobrazení možností vyhledávání v nástroji Kibana.

Hlavní použití pro tuto práci je možnost vyhledávání záznamů podle serveru, přes jednotlivá vlákna a zobrazení konkrétní zprávy. Toho lze docílit právě použitím filtrů. Filtrem se omezí výběr jen na zvolené vlákno a konkrétní server. Kliknutím na konkrétní záznam se zobrazí všechna jeho data i s konkrétní zprávou události.

Závěr

Aplikace ESB v prostředí informačního systému virtuálního mobilního operátora měla problém s decentralizovaným ukládáním logových záznamů. Tato práce měla jako cíl vybrat vhodné nástroje pro toto prostředí, které dokážou tento problém vyřešit, a poté tyto nástroje implementovat. Nástroje musí sjednotit výstupy z aplikace ESB a zároveň umožnit uživatelům full-textové vyhledávání v záznamech.

Nejdříve proběhlo zhodnocení stavu prostředí. Ze zhodnocení vyplynula určitá omezení na výběr nástrojů. Omezení se týkala především využití RAM. Samotný výběr byl rozdělen do jednotlivých částí dle funkčnosti – indexace záznamů, kolekce dat z výstupů, grafické rozhraní pro vyhledávání. Pro indexaci záznamů byly porovnávány nástroje Solr a Elasticsearch. Porovnávala se aktivita komunity, která nástroj spravuje, vývoj, zralost i podpora nástroje. Lepší variantou byl nástroj Elasticsearch, a byl proto vybrán. Pro kolekci dat bylo více kandidátů – Logstash, Fluentd, Fluent bit, Elasticsearch Appender. Všechny splňovaly funkčnost, která byla požadována. Rozdíly mezi nástroji byly ve využití operační paměti a jednoduchosti instalace. Byl vybrán nástroj Elasticsearch Appender, který byl nejjednodušší na instalaci a zároveň měl nejmenší dopad na využití paměti RAM. Tento nástroj je pouze implementace komponenty Appender pro Logback, a proto nenabízí pokročilé konfigurace agregace a transformace dat. Pro složitější agregaci nebo transformaci logových záznamů není tento nástroj vhodný, a proto je při výběru podobných nástrojů vždy třeba zvážit konkrétní požadavky. Výběr grafického rozhraní nebyl složitý. Z důvodu existence ELK Stacku (Elasticsearch, Logstash, Kibana), byla Kibana díky kompatibilitě a optimalizaci pro Elasticsearch mezi favority. Konkurenty jí byly Grafana, Graylog, Prometheus. Každý nástroj měl ovšem jiné zaměření než bylo vyhledávání s pokročilou filtrací s napojením na Elasticsearch. Vybrané nástroje byly tedy Elasticsearch pro indexaci, Elasticsearch Appender pro sběr záznamů a Kibana pro grafické rozhraní umožňující vyhledávání.

Po výběru přirozeně následovala instalace samotných nástrojů. Instalace Elasticsearche a Kibany proběhla stažením příslušných balíčků z repozitářů a rozbalením do adresářů, nebyly tak třeba pro instalaci administrátorská práva. Elasticsearch Appender se pouze přidal jako závislost do projektu aplikace ESB. Před spuštěním bylo třeba nakonfigurovat server. K tomuto úkonu už byla zapotřebí administrátorská práva. Nastavení se týkala úpravy firewallu na serverech, na kterých nástroje mají běžet, a úprav nastavení samotného operačního systému. Po těchto úpravách už zbývalo jen nakonfigurovat samotné nástroje pro spuštění. U Elasticsearche proběhla větší část konfigurace skrze konfigurační soubor, u Kibany obdobně. U Elasticsearch Appenderu proběhla všechna nastavení v konfiguračním souboru Logbacku, kde se samotný Appender definuje.

Instalace i konfigurace byla úspěšná a práce pokračovala spuštěním jednotlivých nástrojů. Úspěšné spuštění bylo ověřeno a zbývalo jen přidat automatizovanou správu indexovaných záznamů pro dlouhodobý provoz. Automatizovaná správa umožnila zabránit hromadění indexů Elasticsearche. Zároveň bylo možné nastavit Elasticsearch tak, aby hlavní část zdrojů serveru využíval pro indexaci nových záznamů a staré ponechal jen pro čtení. Na závěr práce

bylo představeno samotné použití Kibany pro vyhledávání. Všechny cíle práce byly tedy naplněny.

Možná rozšíření práce mohou spočívat v integraci vyhledávání Kibana do existujícího grafického rozhraní aplikace ESB, nebo pro integraci vyhledávání nahradit Kibanu vývojem vlastního nástroje využitím aplikačního rozhraní Elasticsearche.

Použitá literatura

ADAM BATKIN, 2019. Logback Elasticsearch Appender. Contribute to internetitem/logback-elasticsearch-appender development by creating an account on GitHub [online]. Java. B.m.: internetitem [vid. 2019-03-21]. Dostupné z: <https://github.com/internetitem/logback-elasticsearch-appender>

ASAF YIGAL, 2018. Grafana vs. Kibana: The Key Differences to Know. Logz.io [online]. [vid. 2019-03-26]. Dostupné z: <https://logz.io/blog/grafana-vs-kibana/>

BRIAN PAYNE, 2014. Github Compare [online] [vid. 2019-03-05]. Dostupné z: <https://bayne.github.io/github-compare/#!/compare/elastic/elasticsearch/apache/lucene-solr>

CEKI GÜLCÜ, SÉBASTIEN PENNEC a CARL HARRIS, 2017a. Chapter 2: Architecture [online] [vid. 2019-02-21]. Dostupné z: <https://logback.qos.ch/manual/architecture.html>

CEKI GÜLCÜ, SÉBASTIEN PENNEC a CARL HARRIS, 2017b. Chapter 3: Configuration [online] [vid. 2019-02-21]. Dostupné z: <https://logback.qos.ch/manual/configuration.html>

CEKI GÜLCÜ, SÉBASTIEN PENNEC a CARL HARRIS, 2017c. Chapter 4: Appenders [online] [vid. 2019-02-21]. Dostupné z: <https://logback.qos.ch/manual/appenders.html>

CEKI GÜLCÜ, SÉBASTIEN PENNEC a CARL HARRIS, 2017d. Chapter 5: Encoders [online] [vid. 2019-02-21]. Dostupné z: <https://logback.qos.ch/manual/encoders.html>

CEKI GÜLCÜ, SÉBASTIEN PENNEC a CARL HARRIS, nedatováno. Chapter 6: Layouts [online] [vid. 2019-03-21]. Dostupné z: <https://logback.qos.ch/manual/layouts.html>

CLOUD NATIVE COMPUTING FOUNDATION, 2019. Installing Fluentd from Source | Fluentd [online] [vid. 2019-03-26]. Dostupné z: <https://docs.fluentd.org/v1.0/articles/install-from-source>

EDUARDO SILVA, JON HADFIELD a JUAN OSORIO ROBLES, 2018. Introduction [online] [vid. 2019-03-25]. Dostupné z: <https://docs.fluentbit.io/manual/>

EDUARDO SILVA, JON HADFIELD a JUAN OSORIO ROBLES, 2019a. CentOS Packages [online] [vid. 2019-03-25]. Dostupné z: https://docs.fluentbit.io/manual/installation/redhat_centos

EDUARDO SILVA, JON HADFIELD a JUAN OSORIO ROBLES, 2019b. Fluentd & Fluent Bit [online] [vid. 2019-03-25]. Dostupné z: https://docs.fluentbit.io/manual/about/fluentd_and_fluentbit

ELASTICSEARCH B.V, nedatováno. Applying a policy to our index | Elasticsearch Reference [6.7] | Elastic [online] [vid. 2019a-03-26]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/6.7/_applying_a_policy_to_our_index.html

ELASTICSEARCH B.V, nedatováno. Configuring Elasticsearch | Elasticsearch Reference [6.6] | Elastic [online] [vid. 2019b-03-21]. Dostupné z: <https://www.elastic.co/guide/en/elasticsearch/reference/current/settings.html>

ELASTICSEARCH B.V, nedatováno. Discovery settings | Elasticsearch Reference [6.6] | Elastic [online] [vid. 2019c-03-21]. Dostupné z: <https://www.elastic.co/guide/en/elasticsearch/reference/current/discovery-settings.html>

ELASTICSEARCH B.V, nedatováno. Heap size check | Elasticsearch Reference [6.6] | Elastic [online] [vid. 2019d-03-21]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/current/_heap_size_check.html

ELASTICSEARCH B.V, nedatováno. HTTP | Elasticsearch Reference [6.6] | Elastic [online] [vid. 2019e-03-21]. Dostupné z: <https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-http.html>

ELASTICSEARCH B.V, nedatováno. Index Templates | Elasticsearch Reference [master] | Elastic [online] [vid. 2019f-03-28]. Dostupné z: <https://www.elastic.co/guide/en/elasticsearch/reference/master/indices-templates.html>

ELASTICSEARCH B.V, nedatováno. Installing Logstash | Logstash Reference [6.6] | Elastic [online] [vid. 2019g-03-26]. Dostupné z: <https://www.elastic.co/guide/en/logstash/current/installing-logstash.html>

ELASTICSEARCH B.V, nedatováno. Logstash [online] [vid. 2019h-03-07]. Dostupné z: <https://www.elastic.co/products/logstash>

ELASTICSEARCH B.V, nedatováno. Setting the heap size | Elasticsearch Reference [6.6] | Elastic [online] [vid. 2019i-03-21]. Dostupné z: <https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html>

ELASTICSEARCH B.V, nedatováno. Shrink Index | Elasticsearch Reference [6.7] | Elastic [online] [vid. 2019j-03-26]. Dostupné z: <https://www.elastic.co/guide/en/elasticsearch/reference/6.7/indices-shrink-index.html>

ELASTICSEARCH B.V, nedatováno. System call filter check | Elasticsearch Reference [master] | Elastic [online] [vid. 2019k-03-21]. Dostupné z: https://www.elastic.co/guide/en/elasticsearch/reference/master/_system_call_filter_check.html

ELASTICSEARCH B.V, nedatováno. Zen Discovery | Elasticsearch Reference [6.6] | Elastic [online] [vid. 2019l-03-21]. Dostupné z: <https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-discovery-zen.html>

FLUENTD PROJECT, 2019. What is Fluentd? | Fluentd [online] [vid. 2019-03-19]. Dostupné z: <https://www.fluentd.org/architecture>

GORMLEY, Clinton a Zachary TONG, 2015. Elasticsearch: the definitive guide. First edition. Beijing ; Sebastopol, CA: O'Reilly. ISBN 978-1-4493-5854-9.

JASON TEDOR, nedatováno. SecComp fails on CentOS 6 · Issue #22899 · elastic/elasticsearch. GitHub [online] [vid. 2019-03-21]. Dostupné z: <https://github.com/elastic/elasticsearch/issues/22899>

LALIWALA, Zakir, Abdul SAMAD, Azaz DESAI a U. VYAS, 2005. Mule ESB Cookbook. Olton, UNITED KINGDOM: Packt Publishing Ltd. ISBN 978-1-78216-441-8.

MCCANDLESS, Michael, Erik HATCHER, Otis GOSPODNETIĆ a Otis GOSPODNETIĆ, 2010. Lucene in action. 2nd ed. Greenwich: Manning. ISBN 978-1-933988-17-7.

QOS.CH, 2019. SLF4J [online] [vid. 2019-02-21]. Dostupné z: <https://www.slf4j.org/>

SANCHIT AGRAWAL, 2016. Everything you need to know about Enterprise Service Bus (ESB) | HCL Blogs [online] [vid. 2019-03-25]. Dostupné z: <https://www.hcltech.com/blogs/everything-you-need-know-about-enterprise-service-bus-esb>

SAURABH CHHAJED, 2015. Learning ELK Stack. Birmingham, UNITED KINGDOM: Packt Publishing Ltd. ISBN 978-1-78588-670-6.

SEMATEXT GROUP, 2019. Project Hub [online] [vid. 2019-03-05]. Dostupné z: <https://sematext.com/opensource/report/project/trend?q=ElasticSearch,Solr>

SHAH, Sanjay, 2014. Maven for Eclipse. B.m.: Packt Publishing - ebooks Account. ISBN 978-1-78398-712-2.

SOLID IT GMBH, 2019. DB-Engines Ranking. DB-Engines [online] [vid. 2019-03-05]. Dostupné z: <https://db-engines.com/en/ranking/search+engine>

TYLER HICKS, MICHAEL KERRISK, WILL DREWRY a KEES COOK, nedatováno. seccomp(2) - Linux manual page [online] [vid. 2019-03-21]. Dostupné z: <http://man7.org/linux/man-pages/man2/seccomp.2.html>