FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Návrh a nasazení infrastruktury pro sběr, analýzu, vyhledávání a prezentaci serverových dat v rámci podniku
Student:	Bc. Michal Junek
Vedoucí:	Ing. Matyáš Zeipelt
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2020/21

Pokyny pro vypracování

Cílem práce je analyzovat potřeby pro fulltextové vyhledávání například v systémových a chybových log souborech v organizaci, zvolit vhodnou platformu, nasadit ji do podnikové infrastruktury a implementovat prezentaci takto získaných dat.

1. Analyzujte požadavky a specifikujte přesnější zadání.
2. Proveďte rešerši vhodných platforem (např. Elastic Stack, Graylog, atd.) a zvolte vhodné řešení.
3. Vybrané řešení nasaďte v prostředí zadavatele.
4. Navrhněte a implementujte typické postupy a způsoby vizualizace výsledků.
5. Zhodnoťte realizované řešení.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 2. února 2020

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Návrh a nasazení infrastruktury pro sběr, analýzu, vyhledávání a prezentaci serverových dat v rámci podniku

Bc. Michal Junek

Katedra softwarového inženýrství
Vedoucí práce: Ing. Matyáš Zeipelt

28. května 2020

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Matyáši Zeipeltovi za možnost a prostor se podílet na rozvoji firemní infrastruktury a Andrei Romanovskému za pomoc při vývoji a úspěšné nasazení na produkční server. Dále bych rád poděkoval za cenné rady svých blízkých, rodině, kolegů, a jmenovitě své přítelkyni Tereze Lubijové za pomoc při dodatečné stylizaci a úpravě textu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 28. května 2020 . . . . . .

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci

Junek, Michal. *Návrh a nasazení infrastruktury pro sběr, analýzu, vyhledávání a prezentaci serverových dat v rámci podniku.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2020.

Abstrakt

Práce se zabývá návrhem a implementací kompletního řešení správy logových zpráv do existující firemní infrastruktury. Řešení je postavené na službách Elasticsearch, Kibana a Logstash (souhrně Elastic Stack) a využívá jednoduchých služeb, tzv. Beats, pro čtení logovacích souborů z Dockeru. Systém se bude používat pro jednodušší nalezení chyb a monitoring požadavků v nových a existujících aplikacích spravovaných organizací.

Klíčová slova logování,monitoring,elastic,stack,docker

Abstract

Aim of this work is to design and implent a complete solution for log message management to an existing company infrastructure. The solution is based on Elasticsearch, Kibana and Logstash (collectively called Elastic Stack) and uses simple services called Beats for reading log files from Docker. The system will be used to easily find problem and monitor requests in new and existing applications.

Keywords logging,monitoring,elastic,stack,docker

Obsah

Seznam obrázků

Úvod

Správa mnoha aplikací se v IT firmách, bez kvalitní infrastruktury vybudované už od začátku, brzy stane tak trochu noční můrou. Jedním z aspektů je kontrola logovacích souborů, které nemusí být vždy jednoduše dostupné. Jednou je návod pro přístup v dokumentaci, podruhé se člověk musí ptát ostatních, a někdy dokonce nikdo neví a tak přichází na řadu vlastní iniciativa a několik hodin pokoušení se k souborům dostat. Aplikace často běží i na více prostředích, a některé z nich jsou spuštěné vícekrát.

Kapitola 1

Cíl práce

Cílem práce je analyzovat potřeby pro fulltextové vyhledávání, například v systémových a chybových log souborech v organizaci, zvolit vhodnou platformu, nasadit ji do podnikové infrastruktury a implementovat prezentaci takto získaných dat.

Výstupem je systém, který dokáže sbírat logové zprávy z více zdrojů, vyhledávat v nich, vizualizovat je a zabezpečit, aby k nim neměla přístup nepovolaná osoba.

Výsledek bude realizován v organizaci MEDIA FACTORY Czech Republic a.s. (dále pouze organizace) na již fungujících aplikacích. Je potřeba se zaměřit na technické požadavky a způsob implementace, aby řešení mělo co nejmenší zásah do existující infrastruktury.

1.1 Požadavky organizace

Aplikace v organizaci jsou strukturované do několika kategorií a prostředí. Každá aplikace má standardně více prostředí, a to je lokální, kde se jedná o aplikaci spuštěnou na lokálním zařízení každého vývojáře určené pro vývoj, poté tzv. dev prostředí, kde jsou jednotlivé instance aplikace spouštěny automaticky dle větví v systému pro správu verzí `GIT`, dále je tu staging pro testování klientem a ostré produkční prostředí pro provoz. Aplikace se řadí do kategorií dle způsobu spuštění na daném prostředí. Možnými způsoby je spuštění přes Docker[1] containery[2] jako samostatná skupina několika dílčích částí – většinou samotná aplikace, databáze a ostatní podpůrné nástroje – a v poslední řadě samostatné spuštění na dedikovaném nebo virtuálním serveru jako izolovaný celek. Produkční prostředí se navíc může nacházet na infrastruktuře klienta, ke které nutně nemusí mít organizace přístup.

[1] jednotné rozhraní pro izolaci aplikací do containerů[1]

[2] izolované prostředí pro běh aplikace

Hlavním požadavkem je centralizovaný přístup k logovým souborům samotných aplikaci a webových serverů, protože není jednoduché uchovávat přístupy pro jednotlivé zaměstnance na všechny kombinace prostředí a kategorií aplikací. Úkolem je shromažďovat logové zprávy ze všech aplikací v jednom systému pro jednoduchý přístup a dostatečnou správu zabezpečení, aby šlo analyzovat a monitorovat jednotlivé aplikace bez nutnosti se připojovat na vzdálený server nebo žádat o výstupy klienta.

Vývojáři se budou moci přihlásit pod svým doménovým uživatelským jménem a heslem, které se bude ověřovat pomocí LDAP.

Vedlejším požadavkem je minimální zásah do existující infrastruktury a kódu aktuálně spravovaných aplikací.

Kapitola 2

Analýza požadavků

2.1 Problémová doména

Správa pár aplikací nemusí být pro zajetou organizaci těžká, ale pokud jejich číslo naroste, a infrastruktura na to není připravena, nastane chaos. Je těžké se dostat k podstatným informacím, každý úkol trvá déle a oprava chyb se dostaví ke koncovému zákazníkovi o tolik později.

To ale není jediný problém. Poslední chybové hlášky jdou poměrně jednoduše najít, ale na první pohled není vidět, co potencionálně zpomaluje běh aplikace nebo kdy se aplikace opravdu zpomalí. V těchto chvílích se spoléhá, že koncoví zákazníci pomalý běh nahlásí, všimne si ho klient nebo ho při testování odhalí programátor. Mnohdy ale zpomalení nikdo nezaznamená a potencionální zákazník odejde.

Z pohledu organizace také není jednoduché zajistit přístup k logovacím souborům tak, aby je viděli pouze pověření vývojáři a zaměstnanci organizace. Hlavní vývojáři by měli mít přístup k logům z produkčních prostředí, kdežto juniorní nebo vývojáři na zaučení či externisté by měli mít možnost číst nanejvýš data ze stage prostředí.

Často se přístup k logovým souborům řeší přes SSH nebo (S)FTP, ale jejich čtení nemusí být jednoduché. Vyžaduje znát umístění souboru na serveru, nainstalování specializovaného softwaru nebo spuštění shellového příkazu, které nemusí nováček v oboru ovládat.

2.2 Možná řešení

V rámci této kapitoly bude zanalyzováno několik existujících řešení, jejichž implementace by pokrývala požadavky organizace (viz. sekce 1.1). Kritéria výběru byla zaměřena na jednoduchost používání, výkon, rozšiřitelnost a konfigurovatelnost.

2.2.1 Graylog

Oficiální web: `https://www.graylog.org/`

Klady:

- Kompletně zdarma, open source
- Jednoduchá konfigurace
- Jednotné řešení
- Výkon
- Rozsáhlá dokumentace

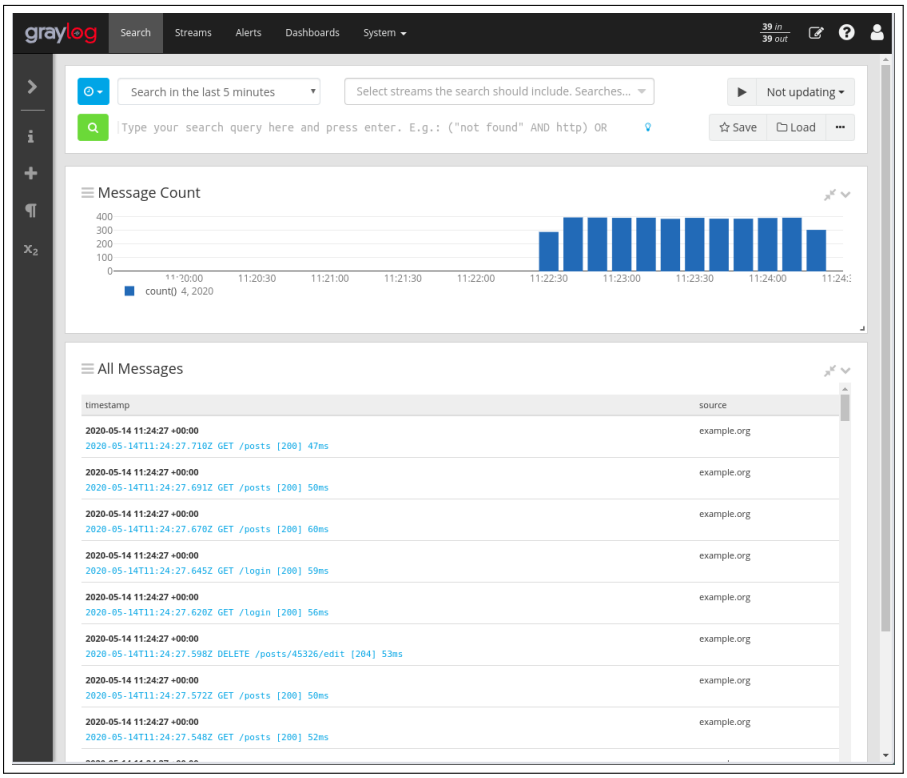
Zápory:

- Nelze jednoduše škálovat
- Podpora pouze jednoduchých vizualizací

Graylog je centralizované řešení pro správu logových zpráv postaveno na otevřených standardech pro získávání, ukládání, a analýzu velkého množství dat v reálném čase[2]. Pracuje s databázemi MongoDB pro ukládání konfigurace a dat.

Hlavní stavební články jsou tzv. `inputs`, které se starají o vkládání dat do systému. Vyšší možnost konfigurace poté nabízejí tzv. `content packs`, kompletní balíčky dostupné v základní konfiguraci a na online obchodu, které se starají o kompletní logiku získávání logových zpráv, jejich transformaci a vizualizaci z konkrétního zdroje, např. Windows Active Directory. Vstupy jsou konfigurovatelné přímo z rozhraní.

Další podstatnou částí jsou tzv. `extractors`, které umí z přijaté zprávy dostat relevantní data, např. IP adresy, čísla, URL adresy aj. Vyhledávání funguje na bázi rozšířeného jazyka Lucene.



Obrázek 2.1: Ukázka rozhraní pro vyhledávání v Graylogu

2.2.2 Elastic Stack

Oficiální web: `https://www.elastic.co/products/`

Klady:

- Lze škálovat jednotlivé části
- In-depth analýza
- Rozšířená konfigurace

Zápory:

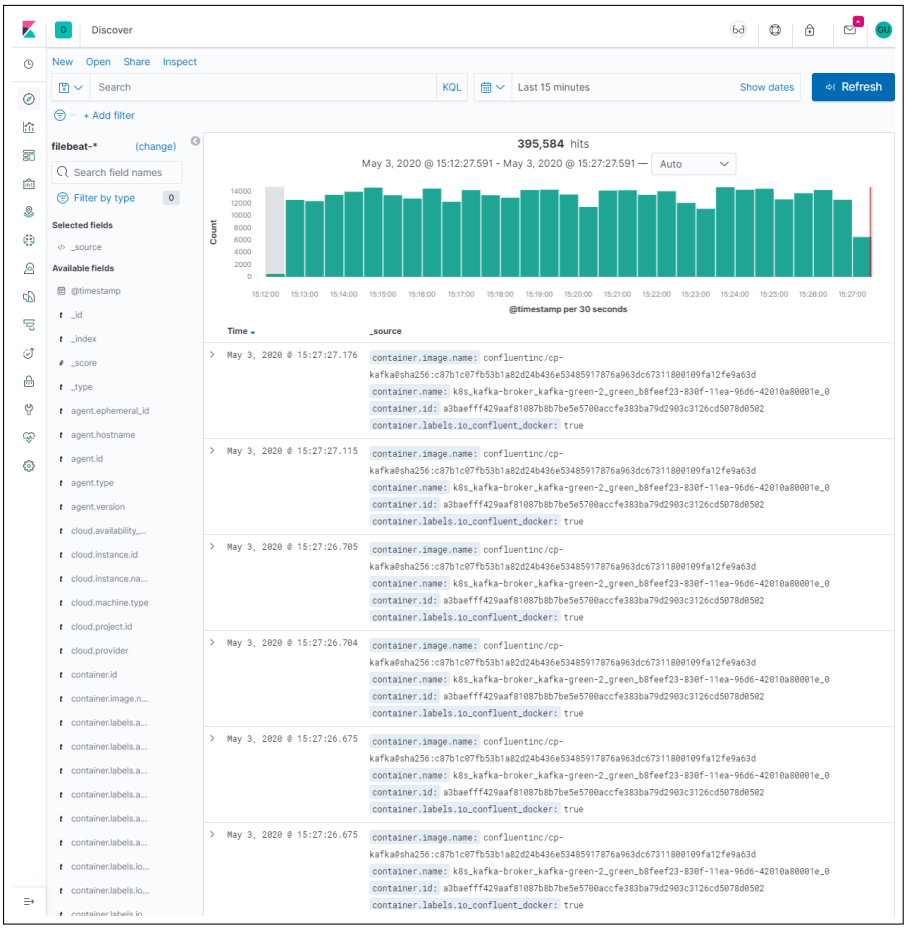
- Některá rozšíření jsou placená
- Žádné defaultní nastavení

Elastic Stack (dále pouze ELK) je soubor aplikací, které spolupracují na sběru, uložení, analýze a vizualizaci logových zpráv. Hlavním úložištěm je no-SQL databáze Elasticsearch, která data ukládá a indexuje k rychlejšímu přístupu. Kibana zde slouží jako uživatelské rozhraní k vizualizaci a analýze logových zpráv, nastavení přístupů a nastavení upozornění. Lze v ní vytvářet pohledy, grafy, reporty atp.

Pro přijímání, transformaci a odeslání dat do úložiště je použit Logstash, který má nakonfigurované pipelines pro rozbor logových zpráv a správné rozčlenění do indexů v databázi. Na rozdíl od Elasticsearche a Kibany, Logstash není nedílnou součástí, ale práci s přijímáním logových zpráv výrazně ulehčuje.

ELK poskytuje velmi rozšířenou konfiguraci pro nastavení na míru. Některá oficiální rozšíření jsou placená, ale existují i neplacené, open source alternativy vytvářené komunitou, které se chovají a vypadají stejně, jako ta oficiální.

K získávání logových zpráv se používají tzv. „Beats“, jednoduché samostatné programy napojené na Logstash, které sbírají logové zprávy ze souborů (Filebeat), dostupnost aplikace (Heartbeat) nebo vytížení systému (Metricbeat).



Obrázek 2.2: Ukázka rozhraní pro vyhledávání v Kibaně

2.2.3 Splunk

Oficiální web: `https://www.splunk.com/`

Klady:

- Mnoho oficiálních rozšíření a integrací
- Kompletní řešení pro organizaci i se zákaznickou podporou

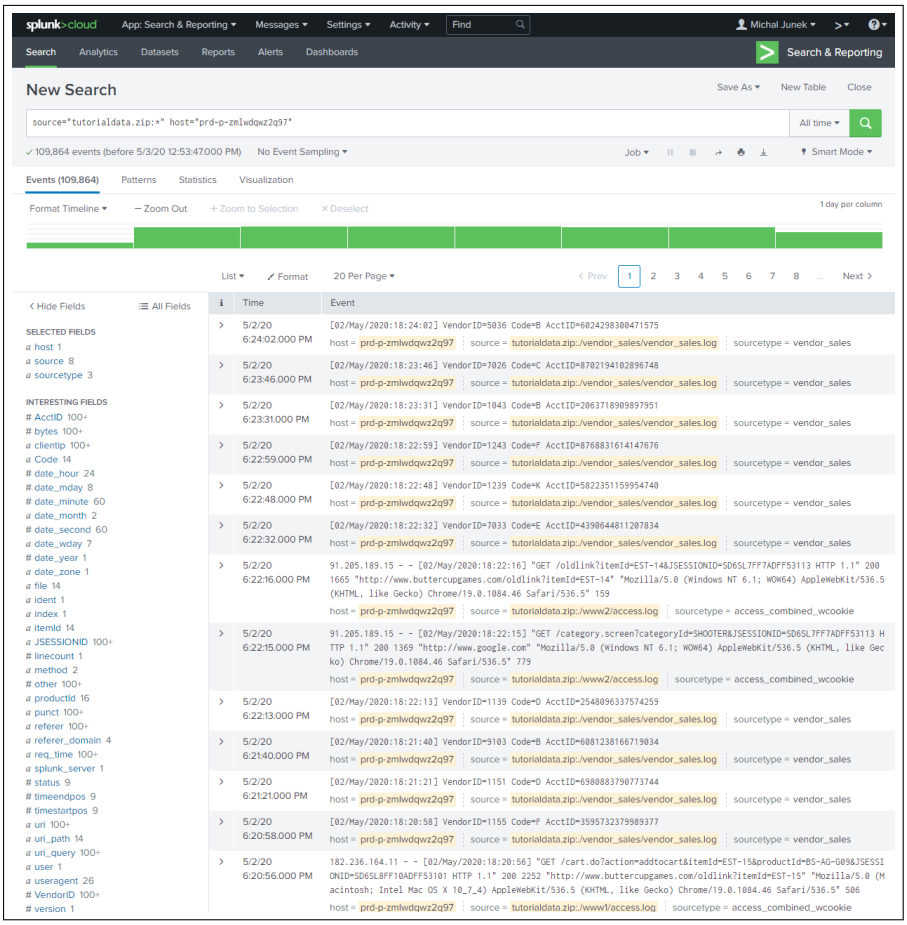
Zápory:

- Vyšší cena
- Vendor lock-in
- Nelze hostovat zdarma na vlastní infrastruktuře

Splunk je software postavený na systému sběru a analýze strojově generovaných dat. V uživatelském rozhraní je možná kompletní správa v podobě nahrávání dat, nastavení zdrojů, vyhledávání a vytváření dashboardů. Hlavní verzí, která se musí hostovat na vlastní infrastruktuře, je Enterprise. Ta obsahuje 60denní trial verzi a poté roční nebo měsiční platbu navyšující se podle počtu uložených dat. Levnější, ale nativně hostovaná cloudu, je verze Cloud, která je sice mnohonásobně levnější, ale sdílí stejnou finanční politiku jako Enterprise.

Díky tomu, že Splunk není open source, nepodporuje tolik rozšíření a přizpůsobení jako ELK nebo Graylog. Na druhou stranu vše, co by kterákoliv organizace potřebovala, je buďto obsaženo v základní verzi nebo se dá zdarma stáhnout či dokoupit na oficiálním obchodě, který poskytuje více jak tisíc rozšíření[3].

Způsob, jak dostat logové zprávy z koncových zařízení, je podobný jako u předchozích dvou řešení. Na koncových zařízeních běží tzv. `universal forwarder`, což jsou konfigurovatelné programy odesílající logové zprávy přímo do Splunku.



Obrázek 2.3: Ukázka rozhraní pro vyhledávání ve Splunku

2.3 Shrnutí

Bylo zanalyzováno několik řešení, které svými vlastnostmi pokrývají všechny hlavní i vedlejší požadavky organizace.

Po konzultaci s vedoucím práce a ostatními vývojáři v organizaci bylo rozhodnuto pro použití Elastic Stacku z důvodu konfigurovatelnosti, ceny i předchozími osobními zkušenostmi s touto technologií.

Kapitola 3

Návrh

V přechozí kapitole bylo rozhodnuto pro použití Elastic Stacku jako technologie pro sběr, vyhledávání, analýzu a monitorování logových zpráv.

V následující kapitole bude navrhnuta implementace do existující firemní infrastruktury a způsob sběru logových zpráv ve všech kategoriích a prostředích.

3.1 Části

Organizace požaduje minimální zásah do infrastruktury, proto všechny následující aplikace budou spuštěny v Dockeru jako nástroje pro izolaci a zprávu jednotlivých instancí. V organizaci se používá Traefik[3], návrh tedy počítá s jeho použitím při implementaci.

Hlavní části jsou Elasticsearch, Kibana a Logstash, vedlejší částí bude Filebeat. Pro každou část bude sestaven vlastní Docker image[4].

3.1.1 Elasticsearch

Elasticsearch je no-SQL databáze pro perzistenci dat, podporuje fulltextové vyhledávání, je vysoce konfigurovatelná, škálovatelná a nabízí distribuovanost na více dílčích databází s možností replikace pro zálohu dat v případě výpadku.[4]

Data jsou ukládány jako dokumenty v indexech dle textového řetězce, kde jeden index pojme libovolný počet dokumentů. S rostoucím počtem dokumentů roste také náročnost na vyhledávání a mizí možnost rozdělení dat dle dalších podkategorií. Proto se data budou ukládat do indexů pojmenovaných dle data a názvu projektu, například `project-x-2020-04-23`, aby bylo na

[3] software pro jednoduchou správu propojení Docker containerů a umožnění dostupnosti přes URL adresy

[4] Šablona pro vytváření Docker containerů

první pohled možné rozeznat, o který projekt se jedná, nebo identifikovat stará nepotřebná data.

3.1.2 Kibana

Kibana je uživatelské rozhraní, které umožňuje vyhledávat ve strukturovaných datech uložených v Elasticsearchi (dále „databáze“). Lze v ní vytvářet dashboardy, vizualizace dle zadaných kritérií, nastavovat práva, upozornění atp. Data budou čteny z databáze, ale než budou vidět v Kibaně, je nutné vytvořit vzory indexů sloužící jako agregace dat z indexů odpovídající zadanému vzoru.

3.1.3 Logstash

Logstash se stará o přijímání, transformaci a přeposílání logových zpráv do databáze. Způsobů přijímání je mnoho[5], ale pro implementaci bude hlavní `http` pro sběr zpráv přímo z aplikací a `beats` pro přijímání z různých Beatů.

Interně je spuštěna jedno nebo více pipelines, které mají přesně definované vstupy, transformace a výstupy. V transformační části je možné data libovolně měnit, obohacovat či klasifikovat pomocí tagů a všechny změny, kromě speciálního pole `metadata`, se projeví na výstupu. Data jsou standardně přijímány a odesílány ve formátu JSON. Všechny zprávy jsou interně konvertovány na tzv. eventy, které jsou dále strojově zpracovatelné.

Výstup bude vždy směrován na instanci databáze a všechny zprávy půjdou přes Logstash, nikoliv přes jiné systémy nebo jinou cestou.

3.1.4 Filebeat

Základní návrh počítá s jediným typem Filebeatu. Ten bude pomocí funkce `autodiscover` prohledávat spuštěné Docker containery a ty, které mají správně nastavené `labels`, bude sledovat a posílat všechny logové zprávy do Logstashe. To bude docíleno pomocí funkce `hint-based autodiscover`[6]. Tato funkcionalita bude rozšířena pro specifikaci názvu projektu.

Pokud má container nastavený label `co.elastic.logs/enabled` na `false`, nebudou sledován a jeho zprávy se nebudou zpracovávat. Název projektu je možné přidat pomocí labelu `co.elastic.logs/project`, který je specifický pro organizaci a není oficiální.

Organizace nyní používá stejný způsob ukládání logových zpráv pro všechny aplikace spuštěné v Dockeru, a to pomocí `json-file` driveru, který zapisuje výstup containeru do logového souboru na disku po dobu, kdy je spuštěn. Tento soubor pak Filebeat sleduje na změny a všechny nové zprávy přeposílá do Logstashe.

[5] více způsobů přijímání na `https://www.elastic.co/guide/en/logstash/current/input-plugins.html`

[6] `https://www.elastic.co/guide/en/beats/filebeat/current/configuration-autodiscover.html`

3.1.5 Zabezpečení a upozornění

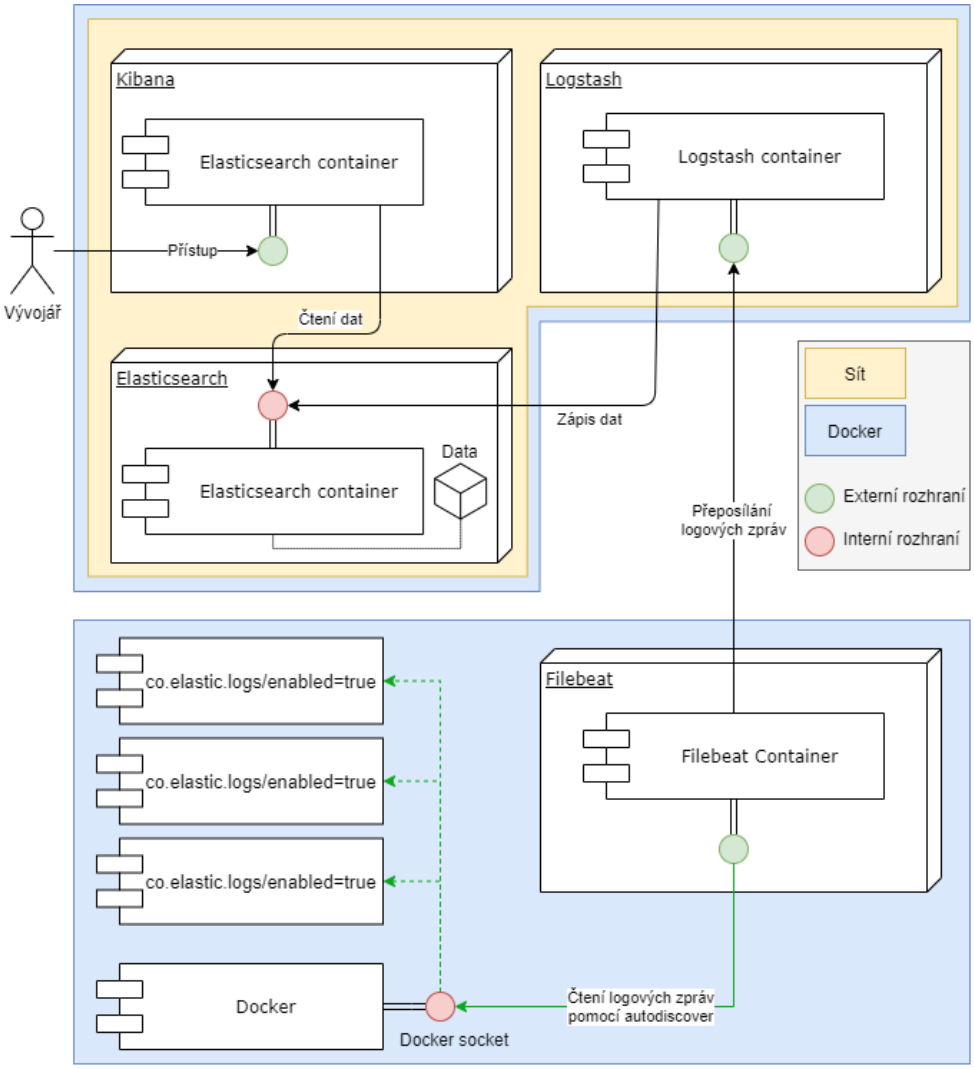
Základní verze Kibany ani Elasticsearche neposkytuje možnost zabezpečení přihlášením ani konfiguraci upozornění. Je nutné buďto dokoupit licenci na X-Pack (oficiální rozšíření), pokusit se o vlastní implementaci nebo nalézt řešení zdarma. Díky otevřenému kódu[5] je možné vytvářet vlastní rozšíření, a proto vzniklo mnoho komunitních projektů s vizí nahradit placená rozšíření alternativami zdarma.

Jedním z takových projektů je Open Distro for Elasticsearch. Poskytuje jak možnost zabezpečení databáze a Kibany tak možnosti upozornění. Instalace těchto pluginů vyžaduje použití předpřipravených Docker images nebo ruční instalaci do již fungující aplikace. Návrh počítá s vlastním sestavením Docker images pro všechny čtyři části a instalací pluginů.

3.2 Infrastruktura

Je nutné správně zabezpečit celý systém. databáze tedy nebude z veřejné sítě dostupná. Logstash a Kibana musejí být veřejně dostupné, aby vývojáři mohli přistupovat k logovým zprávám a aby systém mohl přijímat zprávy z vnějších zdrojů. Zabezpečené připojení přes HTTPS na všech veřejných rozhraních je samozřejmostí, interní komunikace mezi aplikace může zůstat nezabezpečená, protože k fyzickému stroji nemají nepověřené osoby přístup. Filebeat se bude do Logstashe vždy připojovat přes veřejné rozhraní, i když bude spuštěn na stejném zařízení, aby se zamezilo možným chybám a nutnosti dvojí konfigurace pro interní i externí přístup.

Všechny relevantní soubory budou verzované pomocí Gitu.



Obrázek 3.1: Návrh infrastruktury

3.3 Sběr dat pomocí Logstashe

Sbírat logové zprávy bude možné dvěma způsoby. Jak již bylo zmíněno, všechny zprávy budou přeposílány Logstashem, kde bude využito dvou hlavních možností vstupů `gelf` a `beats`.

3.3.1 Příjem

Plugin gelf přijímá zprávy ve formátu GELF a ty interně konvertuje na Logstash eventy. Podporuje jak protokoly TCP i UDP, dokáže přijímat i částečné zprávy a podporuje kompresi. Knihovna pro odesílání je, díky využití v systémech používajících Graylog, implementována pro mnoho programovacích jazyků.

Plugin beats zpracovává zprávy z různých Beats přes TCP protokol `lumberjack`. Podporuje zabezpečení komunikace pomocí certifikátů.

3.3.2 Zpracování

Každá zpráva projede soustavou pluginů, které data transformují a upraví k finálímu uložení. Zde bude správné rozřazení do indexů, parsování plaintextových zpráv do strukturovaných dat a skrytí citlivých informací.

3.3.3 Odesílání

Všechna data budou odesílána do databáze pomocí pluginu `elasticsearch`. Při odeslání si lze definovat název indexu, kam lze dosadit částí originálního eventu.

3.4 Odesílání dat z PHP

V organizace je většina aplikací naprogramována v jazyce PHP a produkční prostředí nejsou spuštěné přes Docker. Vyskytuje se zde více verzí PHP, od 5.4 po 7.4. Díky mnohaletému vývoji se tyto aplikace zásadně liší, i když jsou postavené na stejném jádru, a proto bude navrhnuta implementace do aplikací, které již mají ucelený logovací systém. Pro ostatní aplikace bude implementace podobná, ale individuálně řešená mimo rozsah práce.

Aktuálně jsou na lokálním, dev i stagining všechny aplikace spuštěné pomocí Dockeru a výstup se automaticky loguje do výstupu containeru, zde tedy není potřeba nic upravovat či přidávat. Práce počítá s návrhem odesílání logových zpráv z produkčních prostředí bez využití kombinace Dockeru a Filebeatu.

Nejvíce rozšířeným způsobem logování v existujících projektech je PHP balíček Monolog ve verzi 1, která se ale od verze 2 použitím příliš neliší.

3.4.1 Monolog

Monolog je balíček pro PHP (instalovatelný pod názvem `monolog/monolog`), který nabízí rozhraní pro odesílání logových zpráv do externích i lokálních služeb. Pro odesílání zpráv do Logstashe bude využito rozšíření pro Monolog s názvem Gelf, které není součástí originálního balíčku, proto je nutné nainstalovat balíček `graylog2/gelf-php`.

Zde se naskýtají dva problémy. Logstash input plugin `gelf` neumí SSL ověřerní. To by mohl být problém, ale díky použití Traefiku můžeme ověření certifikátu delegovat na loadbalancer. Druhým problémem se jevila nemožnost odesílání klientského certifikátu, pouze souboru certifikační autority, který je v tomto případě k ničemu a ověřuje pouze identitu serveru, nikoliv klienta. Tento problém byl opraven vytvořením nové třídy `ClientSslOptions`, která rozšiřuje originální třídu o funkcionalitu připojení klientského certifikátu.

Zdrojový kód 3.1: Rozšíření třídy `SslOptions`

```php
<?php
class ClientSslOptions extends Gelf\Transport\SslOptions
{
  private $certKey;
  private $certFile;

  public function setCertKey($certKey) {
    $this->certKey = $certKey;
  }

  public function setCertFile($certFile) {
    $this->certFile = $certFile;
  }

  public function toStreamContext($serverName = null) {
    $context = parent::toStreamContext($serverName);
    $sslContext = &$context['ssl'];

    if (null !== $this->certFile)
      $sslContext['local_cert'] = $this->certFile;
    if (null !== $this->certKey)
      $sslContext['local_pk'] = $this->certKey;

    return $context;
  }
}
```

Celý systém odesílání zpráv je založen na kanálech (channels), kde každá instance loggeru má právě jeden definovaný kanál a několik handlerů jako pole objektů. Každá zpráva, kterou kanál přijme, prochází handlery vždy od prvního, kde zpráva může být zpracována a její životnost ukončena, zpra-

cována a poslána dál nebo pouze poslána dál. Zprávy lze kategorizovat dle vážnosti[6]. Je tedy možné například odesílat e-maily pro všechny zprávy ERROR a výše a ostatní zprávy pouze zapisovat do souboru či jinak zpracovávat.

Zprávy lze dodatečně v jednotlivých handlerech formátovat pomocí `formatters`, aby ji koncový systém dokázal zpracovat nebo k ucelení její struktury pro jednotný výstup.

Na handlery nebo celý logger lze ještě aplikovat processory, které zprávu obohacují o dodatečné informace, jako například aktuální využití CPU nebo místo v kódu, odkud byla zpráva odeslána. Toho bude využito zejména u propojení požadavku a logové zprávy pomocí jedinečného identifikátoru. K realizaci bude využit processor `WebProcessor`, který je součástí základního balíčku a ke zprávě přidá informace o požadavku včetně žádaného pole `UNIQUE_ID`.

Do existujících projektů bude přidán do Monologu handler Gelf, jehož účelem bude posílat zprávy do Logstashe.

Zdrojový kód 3.2: Příklad implementace Gelf handleru do Monologu

```php
<?php
// Setup SSL
$ssl = new ClientSslOptions();
$ssl->setCertFile(__DIR__ . '/cert.pem');
$ssl->setCertKey(__DIR__ . '/cert-key.pem');
$ssl->setCaCert(__DIR__ . '/root-ca.pem');

// Setup logger
$host = $_ENV['LOGSTASH_HOST'];
$port = $_ENV['LOGSTASH_PORT'];

$transport = Gelf\Transport\TcpTransport($host, $port, $ssl);
$publisher = new Gelf\Publisher($transport);
$handler = new Monolog\Handler\GelfHandler($publisher);

// Send message
$logger = new Monolog\Logger('main', [$handler]);
$logger->log(Monolog\Logger::INFO, 'Test message');
```

Kapitola 4

Implementace

V následující kapitole bude připraveno lokální prostředí pro testování a následně bude celý systém nasazen do firemní infrastruktury. Dále budou nastaveny přístupy vývojářům a realizováno odesílání logových zpráv z projektů.

4.1 Struktura projektů

Všechny hlavní částí budou spouštěny jako celek a budou tedy sdílet jeden repozitář, vedlejší části budou odděleně. Hesla a další údaje pro propojení jednotlivých částí budou vždy nastavené zvlášť pomocí proměnných prostředí v souboru `.env`.
Dle politiky organizace bude struktura repozitáře hlavních částí následující (struktura vedlejších částí bude podobná):

- `/`
  - `bin` ..... příkazy pro spuštění, zastavení a restartování
  - `extras` ..... složka s konfiguračními soubory pro jednotlivé části
    - `elasticsearch`
    - `kibana`
    - `logstash`
    - `traefik`
    - `root-certs` ..... složka s kořenovým certifikátem
  - `.env.dist` ..... proměnné prostředí, šablona pro `.env`
  - `docker-compose.local.yml` soubor pro spuštění aplikace na lokálním prostředí
  - `docker-compose.prod.yml` ..... soubor pro spuštění aplikace na produkčním prostředí
  - `Makefile` ..... podpůrné příkazy
  - `README.md` ..... instrukce pro spuštění

4.2 Docker images

Každá část je jeden izolovaný celek, který bude napojen na ostatní části. K vytvoření Docker image pro každou část bude použita vestavěná funkcionalita Dockeru na sestavení image pomocí souboru `Dockerfile`. To je sada příkazů ovlivňující výsledné chování containeru spuštěným přes takto sestavenou image. Lze tak definovat základní image, kopírovat soubory z lokální složky, otevírat porty nebo spouštět příkazy jejichž efekt se ve výsledném image projeví.

Verze všech částí by měly být stejné a co nejnovější pro zajištění maximální kompatibility. Je třeba brát v potaz i verze pluginů pro zabezpečení a upozornění a pro jaké verze nadřazených aplikací jsou určené. V době vzniku této práce je nejnovější dostupná stabilní verze pluginů 1.6.0.0, které je kompatibilní s verzemi jednotlivých částí 7.6.1.[7] Žádná část nebude využívat placené funkce, proto bylo rozhodnuto pro využití tzv. OSS verze pro menší výslednou velikost, nižší nároky na výkon a zaručenou kompatibilitu s pluginy.

Pro jednotlivé části bude využitu následujících Docker images jako základ:

- Elasticsearch
  `docker.elastic.co/elasticsearch/elasticsearch-oss:7.6.1`
- Kibana
  `docker.elastic.co/kibana/kibana-oss:7.6.1`
- Logstash
  `docker.elastic.co/logstash/logstash-oss:7.6.1`
- Filebeat
  `docker.elastic.co/beats/filebeat-oss:7.6.1`

Budou přidány příkazy na instalaci pluginů pomocí vestavěného scriptu s přepínačem pro potvrzení žádosti o udělení práv, aby se vše dalo zautomatizovat.

Zdrojový kód 4.1: Příklad `Dockerfile` pro Elasticsearch

```
FROM docker.elastic.co/elasticsearch/elasticsearch-oss:7.6.1

# Install plugins
RUN bin/elasticsearch-plugin install --batch \
https://d3g5vo6xdbdb9a.cloudfront.net/downloads/elasticsearch-plugins\
/opendistro-security/opendistro_security-1.6.0.0.zip

RUN bin/elasticsearch-plugin install --batch \
https://d3g5vo6xdbdb9a.cloudfront.net/downloads/elasticsearch-plugins\
/opendistro-security/opendistro_security-1.6.0.0.zip
```

4.3 Konfigurace

Prvním konfiguračním souborem bude bezpochyby `docker-compose.yml`. Tento soubor, ve formátu YAML[7], slouží pro definování chování Dockeru při orchestraci projektu. Hlavními částmi konfigurace je nastavení síťování `networks`, definice úložiť `volumes` a jednotlivých služeb `services`.

Tajné proměnné a proměnné závislé na prostředí budou uložené v souboru `.env`, které lze používat v souboru `docker-compose.yml` a dále předávat jednotlivých containerům.

4.3.1 Docker Compose

Pro každou službu bude její část v `docker-compose.yml` popsána v její vlastní sekci.

4.3.1.1 Verze a logování

Verze Docker Compose jsou rozdělené do dvou hodnot. Verze samotného programu udává, jaká nejvyšší verze souboru je dostupná. Verze souboru udává dostupnost jednotlivých funkcí, povolené hodnoty odrážek a postup orchestrace. Je dobré používat nejnovější verzi, ale best practice v rámci organizace je používat všude stejnou, aby se předešlo rozdílům v definici a funkcionalitě.

Bude využita verze `3.7`, která se používá v celé organizaci a mimo jiné podporuje i YAML Anchors. Díky této funkcionalitě je možné používat jednu část nastavení na více místech a zabraňuje opakování (princip DRY[8]).

Samotná konfigurace bude výstup aplikace ukládat jako strukturované JSON zprávy s informacemi o času, typu výstupu (`stdout`, `stderr`) a obsahuje i samotou zprávu. V případě, že soubor dosáhne maximální specifikované velikosti, vytvoří se nový – sekvenční – soubor a původní se zachová. Konfigurace bude omezovat velikost těchto souborů na 10 MB a nastaví maximální počet sekvenčních souborů na 10.

Zdrojový kód 4.2: Konfigurace verze logování

```
version: '3.7'

x-defaults: &defaults
  restart: ${DOCKER_RESTART_POLICY}
  logging:
    driver: "json-file"
    options:
      max-file: "10"
      max-size: "10m"
```

[7] textový, lidsky čitelný formát, nejčastěji používaný pro konfigurační soubory
[8] Don‘t Repeat Yourself

4.3.1.2 Síťování

Jednoduchá část pro síťování slouží k vytvoření virtuálních sítí mezi containery při spuštění projektu. Externí sítě jsou typicky vytvořené pro jinou aplikaci, a specifikace, že je síť externí, nebude při spuštění vytvořena ale pokusí se připojit k existující. Containery se na sebe v síti mohou odkazovat pomocí jejich `hostname`.

Zde bude zápis jedné externí sítě `webapp` vytvořenou Traefikem a jedné interní sítě `intnet` pro spojení containerů.

Zdrojový kód 4.3: Konfigurace sítí

```
networks:
  webapp:
    external: true
  intnet:
    external: false
```

4.3.1.3 Úložiště

Všechny změny v containeru jsou při jeho vypnutí ztraceny a při dalším spuštění je znovu vytvořen z jeho šablony (image). Containery lze pouze pozastavit, v tom případě se změny zachovají. Pokud chceme přenášet stav aplikace mezi jednotlivými spuštěními, je důležité relevantní data ukládat na disk. Úložiště slouží jako virtuální disky určené k perzistenci souborů, které je možné pomocí odrážky `volumes` v jednotlivých službách připojovat do souborového systému containeru. Využívá se hlavně u databází, aby se změny neztratili při opakovaném spouštění. Po jejich deklaraci stačí místo názvu složky v deklaracích jednotlivých služeb použít název úložiště.

V původním návrhu se počítalo s jejich využitím, ale po konzultaci s hlavním vývojářem bude pro perzistenci dat využita lokální složka.

Zdrojový kód 4.4: Příklad konfigurace úložišťě

```
volumes:
  elasticsearch-data:
    driver: local
```

4.3.2 Elasticsearch

Elasticsearch vyžaduje několik souboru navíc. Jedná se hlavně o interní certifikáty, které zajišťují zabezpečenou komunikaci na transportní ISO vrstvě mezi samotným databází a jeho pluginy. Tyto certifikáty se také využívají na komunikaci mezi jednotlivými uzly. V základu bude databáze zapnuta v módu dvou uzlů s možností rozšíření na další. Dále je zde připojeno úložiště pro perzistenci dat, hlavní konfigurační soubor a podpůrná konfigurace pro nastavení základních uživatelů, skupin a pravomocí.

Počáteční certifikáty pro lokální prostředí budou generovány z kořenového certifikátu pomocí Makefile příkazem `create-certs`.

Zdrojový kód 4.5: Definice služby - uzel Elasticsearch

```
app-elasticsearch-node1:
  <<: *defaults
  image: ${APP_ELASTICSEARCH_IMAGE}
  container_name: "${APP_ID}-elasticsearch-node1"
  hostname: "${APP_ID}-elasticsearch-node1"
  environment:
    - cluster.name=${APP_ID}-cluster
    - node.name=${APP_ID}-elasticsearch-node1
    - discovery.seed_hosts=
    ↪  ${APP_ID}-elasticsearch-node1,${APP_ID}-elasticsearch-node2
    - cluster.initial_master_nodes=
    ↪  ${APP_ID}-elasticsearch-node1,${APP_ID}-elasticsearch-node2
    - bootstrap.memory_lock=true # along with the memlock settings
    ↪  below, disables swapping
    - "ES_JAVA_OPTS=${ES_JAVA_OPTS}"
    - "TZ=Europe/Prague"
    - "LDAP_HOST=${LDAP_HOST}"
    - "LDAP_PASSWORD=${LDAP_PASSWORD}"
    - "LDAP_BIND_DN=${LDAP_BIND_DN}"
    - "LDAP_USERBASE=${LDAP_USERBASE}"
    - "LDAP_USERSEARCH=${LDAP_USERSEARCH}"
    - "LDAP_USERSNAME_ATTRIBUTE=${LDAP_USERSNAME_ATTRIBUTE}"
  labels:
    - "traefik.enable=false"
    - "co.elastic.logs/enabled=false"
  networks:
    - intnet
```

V příkladu konfigurace byly vynechány sekce pro nastavení paměťových limitů a napojení úložišť. Úložiště v tomto případě spojují složky a soubory pro perzistentní úložiště, certifikáty, hlavní konfiguraci a nastavení zabezpečení. Kompletní konfigurace je dostupná na přiloženém médiu.

4.3.2.1 Hlavní konfigurace

Hlavní konfigurační soubor má více možností konfigurace[9], ale většina bude ponechána v defaultním nastavení. Změny budou hlavně v konfiguraci pro plugin zabezpečení[10]. Hlavními změnami bude pojmenování certifikačních souborů, nastavení zabezpečení, založení indexu pro uložení zabezpečení při startu a nastavení auditu přístupů.

Zdrojový kód 4.6: Hlavní konfigurace - Elasticsearch

```
network.host: 0.0.0.0
node.name: ${node.name}
node.max_local_storage_nodes: 3
http.compression: true
discovery.seed_hosts: ${discovery.seed_hosts}
cluster.name: ${cluster.name}
cluster.initial_master_nodes: ${cluster.initial_master_nodes}
cluster.routing.allocation.disk.watermark.low: 10gb
cluster.routing.allocation.disk.watermark.high: 7gb
cluster.routing.allocation.disk.watermark.flood_stage: 5gb
cluster.routing.allocation.disk.threshold_enabled: false
opendistro_security.compliance.salt: 34p2dmw5b3hx3t36
opendistro_security.ssl.transport:
  enabled_protocols: ["TLSv1.3"]
  pemcert_filepath: node.pem
  pemkey_filepath: node-key.pem
  pemtrustedcas_filepath: root-ca.pem
  enforce_hostname_verification: false
opendistro_security.ssl.http.enabled: true
opendistro_security.ssl.http.pemcert_filepath: node.pem
opendistro_security.ssl.http.pemkey_filepath: node-key.pem
opendistro_security.ssl.http.pemtrustedcas_filepath: root-ca.pem
opendistro_security.authcz.admin_dn:
  - 'CN=ADMIN,OU=MF DEV,O=MEDIA FACTORY Czech Republic a.s.,C=CZ'
opendistro_security.nodes_dn:
  - 'CN=mfelk-elasticsearch-node1,OU=MF DEV,O=MEDIA FACTORY Czech
  ↪  Republic a.s.,C=CZ'
  - 'CN=mfelk-elasticsearch-node2,OU=MF DEV,O=MEDIA FACTORY Czech
  ↪  Republic a.s.,C=CZ'
opendistro_security.allow_default_init_securityindex: true
opendistro_security.audit.type: internal_elasticsearch
opendistro_security.enable_snapshot_restore_privilege: true
opendistro_security.check_snapshot_restore_write_privileges: true
opendistro_security.restapi.roles_enabled: ["all_access",
↪  "security_rest_api_access"]
opendistro_security.audit.config.disabled_rest_categories: NONE
opendistro_security.audit.config.disabled_transport_categories: NONE
```

[9] více na https://git.io/JflRo
[10] více na https://git.io/JflRK

4.3.3 Kibana

Kibana bude přístupná z veřejné sítě, proto musí být definováno několik položek navíc týkajících se Traefiku. Jedná se hlavně o `labels`, aby fungovalo směrování z URL definované v `.env` souboru na adresu aplikace. V tom je zahrnuto přesměrovaná z `http` na zabezpečené `https`. Dále je třeba zajistit interní přístup k databázi, aby mohla Kibana získávat data – to je realizováno pomocí interní sítě.

Zdrojový kód 4.7: Definice služby - Kibana

```
app-kibana:
  <<: *defaults
  image: ${APP_KIBANA_IMAGE}
  container_name: "${APP_ID}-kibana"
  hostname: "${APP_ID}-kibana"
  environment:
    - "server.name=${KIBANA_SERVER_NAME}"
    - "ELASTICSEARCH_HOST=
    ↪  https://${APP_ID}-elasticsearch-node1:9200"
    - "ELASTICSEARCH_USER=${KIBANA_ELASTICSEARCH_USER}"
    - "ELASTICSEARCH_PASSWORD=${KIBANA_ELASTICSEARCH_PASSWORD}"
    - "TZ=Europe/Prague"
  labels:
    - "traefik.enable=true"
    - "traefik.docker.network=webapp"
    - "co.elastic.logs/enabled=false"
  networks:
    - webapp
    - intnet
  volumes:
    - ./extras/root-certs/root-ca.pem:
    ↪  /usr/share/kibana/certs/root-ca.pem:ro
    - ./extras/kibana/kibana.yml:/usr/share/kibana/config/kibana.yml
```

V příkladu konfigurace byly vynechány sekce pro nastavení Traefiku pomocí labels. Jedná se o standardní přidělení URL adresy a přesměrování na HTTPS. Kompletní konfigurace je dostupná na přiloženém médiu.

4.3.3.1 Hlavní konfigurace

Díky instalaci pluginu pro zabezpečení databáze se musí Kibana umět přihlásit pod účtem s dostatečnými právy na čtení a zápis jak dat tak nastavení. Na to bude vyhrazen samostatný účet, který fyzické osoby nebudou používat. Z pohledu databáze se Kibana jeví jako další uživatel, proto je záhodno dát tomuto účtu co nejvyšší stupeň pravomocí. Práva přihlášeného uživatele se odvíjejí od jeho nastavených práv, nikoliv jaká práva má Kibana samotná.

Kibana podporuje také tzv. `multitenancy`. Tenants jsou virtuální prostory (dále „prostory“), do kterých je možné vytvářet y, vizualizace a ty jsou poté automaticky sdílené s ostatními uživateli, kteří mají do prostoru přístup. V základu existují prostory Private, který má každý uživatel sám pro sebe a nejde sdílet, a poté Global, do kterého mají přístup všichni uživatelé a je vybrán defaultně.

Zdrojový kód 4.8: Hlavní konfigurace - Kibana

```
server.name: ${server.name}
server.host: 0.0.0.0

elasticsearch.hosts:
  - ${ELASTICSEARCH_HOST}

elasticsearch.requestHeadersWhitelist: ["securitytenant",
↪  "Authorization"]
elasticsearch.username: ${ELASTICSEARCH_USER}
elasticsearch.password: ${ELASTICSEARCH_PASSWORD}
elasticsearch.ssl.certificateAuthorities: [
↪  "/usr/share/kibana/certs/root-ca.pem" ]

opendistro_security.multitenancy.enabled: true
opendistro_security.multitenancy.tenants.enable_private: true
opendistro_security.multitenancy.tenants.enable_global: true
opendistro_security.multitenancy.tenants.preferred: ["Global",
↪  "Private"]
opendistro_security.readonly_mode.roles: ["kibana_read_only"]

telemetry.enabled: false
```

4.3.4 Logstash

Stejně jako u Kibany, i Logstash bude veřejně přístupný. Zde bude navíc definice zpřístupněn protokol TCP z důvodu přijímání logových zpráv z Filebeatu a Gelfu (viz. 3.3.1).

Zdrojový kód 4.9: Definice služby - Logstash

```
app-logstash:
  <<: *defaults
  image: ${APP_LOGSTASH_IMAGE}
  container_name: "${APP_ID}-logstash"
  hostname: "${APP_ID}-logstash"
  environment:
    - "ELASTICSEARCH_URL=https://${APP_ID}-elasticsearch-node1:9200"
    - "ELASTICSEARCH_USER=${ADMIN_ELASTICSEARCH_USER}"
    - "ELASTICSEARCH_PASSWORD=${ADMIN_ELASTICSEARCH_PASSWORD}"
    - "TZ=Europe/Prague"
  labels:
    - "co.elastic.logs/enabled=false"
    - "traefik.enable=true"
    - "traefik.docker.network=webapp"
    # logstash filebeats
    - "traefik.tcp.routers.${APP_ID}-logstash.rule=
    ↪  HostSNI(`${APP_LOGSTASH_HOST}`)"
    - "traefik.tcp.routers.${APP_ID}-logstash.service=
    ↪  ${APP_ID}-logstash"
    - "traefik.tcp.routers.${APP_ID}-logstash.entrypoints=https"
    - "traefik.tcp.routers.${APP_ID}-logstash.tls=true"
    - "traefik.tcp.routers.${APP_ID}-logstash.tls.passthrough=true"
    - "traefik.tcp.services.${APP_ID}-logstash
    ↪  .loadbalancer.server.port=5044"
  networks:
    - webapp
    - intnet
```

Konfigurace Traefiku pro Gelf a sekce `volumes` byly vynechány. Kompletní konfigurace je dostupná na přiloženém médiu.

4.3.4.1 Hlavní konfigurace

Hlavní konfigurační soubor neobsahuje mnoho nastavení relevantních pro potřeby organizace[11]. Zde bude pouze adresa pro naslouchání požadavků.

Zdrojový kód 4.10: Konfigurace Logstash

```
http.host: 0.0.0.0
```

[11] více na `https://git.io/JflSX`

4.3.4.2 Pipelines

Stěžejní částí konfigurace Logstashe jsou pipelines (viz. 3.1.3). Pro přehlednost budou vytvořeny celkem tři pipelines. První dvě budou fungovat primárně jako vstupy a připraví zprávu pro další zpracování zjištěním prostředí, názvu projekty a výsledný event odešlou do společné pipeline. Důvodem je rozdílná struktura zprávy díky jinému typu vstupu. Filebeat vidí informace o Docker containeru, proto dokáže odvodit název projektu a prostředí. Na druhou stranu logové zprávy z projektů, které nevyužívají Dockeru, musí obsahovat název projektu a prostředí přímo ve zprávě.

První částí konfigurace pipeline jsou její vstupy. Pro účely organizace bude přijímat zprávy z Beats na portu 5044 a GELF požadavků na portu 5045. Konfigurace posílání zpráv mezi pipelines je dostupná v oficiální dokumentaci[12].

Zdrojový kód 4.11: Logstash pipeline - vstup pro Beats

```
input {
  beats {
    port => 5044
    ssl => true
    ssl_certificate => "/usr/share/logstash/certs/logstash.pem"
    ssl_key => "/usr/share/logstash/certs/logstash-key.pem"
  }
}
```

U druhé části – filtrace nebo také transformace – vstupních pipelines je prvním krokem uložení názvu projektu do metadat pro jednoduchou kategorizaci do správného indexu. Jak již bylo zmíněno, metadata se neukládají do databáze, ale lze je využívat jako proměnné v celém pipeline. Každý vstup bude mít tuto sekci rozdílnou, ale efekt bude stejný – uložení názvu projektu a prostředí do metadat.

Zdrojový kód 4.12: Logstash pipeline - nastavení metadat

```
if [container][labels][co_elastic_logs/project] {
  mutate { add_field => { "[@metadata][project]" =>
  ↪  "%{[container][labels][co_elastic_logs/project]}" } }
} else if [container][labels][com_docker_compose_project] {
  mutate { add_field => { "[@metadata][project]" =>
  ↪  "%{[container][labels][com_docker_compose_project]}" } }
}

if [fields][env] {
  mutate { add_field => { "[@metadata][runenv]" => "%{[fields][env]}"}
  ↪  }
}
```

[12] https://www.elastic.co/guide/en/logstash/7.6/pipeline-to-pipeline.html

Společná pipeline využívá vzory pro `grok` na rozbor Nginx, Apache a Traefik logových zpáv. Grok má na vstupu definovanou složku se vzory, kde každý soubor může obsahovat libovolný počet vzorů využitelné pro rozbor samotné zprávy. Pokud se rozbor nepovede, `grok` přidá ke zpávě štítek `_grokparsefailure`. Tohoto principu je využitu při dalších podmínkách, které testují existenci tohoto štítku a na jeho základě se pokouší rozebrat zprávu pomocí dalších vzorů a posléze přejmenovávají pole.

Při úspěšném rozboru se proměnné uloží do nových polí existující zprávy, ale protože `grok` nedokáže ukládat hodnoty do vnořených polí, je nutné využití pluginu `mutate` a jeho funkce `rename` pro jejich přesunutí.

Z důvodu lepší kontroly nad rozborem zpráv, lepšího pojmenování výstupních polí a nestandardního formátu logových zpráv v organizaci byly definovány vzory pro jednotlivé typy zpráv přestože na tyto typy zpráv již vzory existují.

Kompletní definice vzorů je dostupná na přiloženém médiu ve složce `src/mf-elk/extras/logstash/patterns`.

Zdrojový kód 4.13: Logstash pipeline - rozbor Apache chyby

```
grok {
  patterns_dir => "/etc/logstash/patterns"
  match => {
    "message" => [
      "%{APACHE_ERROR_EXTENDED_MF_L1}",
      "%{APACHE_ERROR_EXTENDED_MF_L2}",
      "%{APACHE_ERROR_EXTENDED_MF_L3}"
    ]
  }

  add_tag => ["apache_error"]
  remove_tag => ["_grokparsefailure"]
}
```

Zdrojový kód 4.14: Logstash pipeline - přejmenování polí

```
if "_grokparsefailure" not in [tags]
  if "apache_error" in [tags] {
    mutate {
      rename => {
        "apache_connection_timestamp" =>
        ↪  "[apache][connection_timestamp]"
        "apache_connection_id" => "[apache][connection_id]"
        "apache_log_id" => "[apache][log_id]"
        "apache_process_id" => "[apache][process_id]"
        "apache_client_ip" => "[apache][client_ip]"
        "apache_file" => "[apache][file]"
        "apache_os_text" => "[apache][os_text]"
```

```
        "apache_level" => "[apache][level]"
        "apache_message" => "[apache][message]"
      }
    }
  }
}
```

Součástí společné pipeline bude i nastavení defaultních hodnot, pokud se je na vstupu nepodařilo zjistit a nejsou vyplněné.

Zdrojový kód 4.15: Logstash pipeline - nastavení projektu

```
if not [@metadata][runenv] {
  mutate {
    add_field => { "[@metadata][runenv]" => "unknown-env" } }
}
if not [@metadata][project] {
  mutate {
    add_field => { "[@metadata][project]" => "unknown-project" } }
}
```

Poslední část definuje způsoby výstupu. Hlavním bude odesílání zprávy do databáze na dané adrese. Je zde definován i název indexu ve formátu `PROJEKT-PROSTŘEDÍ-YYYY-MM-DD`. Logstash se připojí pod určeným účtem a data uloží. Každá zpráva je navíc odeslána na výstup jako logová zpráva containeru pro možnost zpětného prohlédnutí si celého výstupu včetně metadat.

Zdrojový kód 4.16: Logstash pipeline - výstupy

```
output {
  elasticsearch {
    hosts => ["${ELASTICSEARCH_URL}"]
    index =>
    ↪  "%{[@metadata][project]}-%{[@metadata][runenv]}-%{+YYYY.MM.dd}"
    user => "${ELASTICSEARCH_USER}"
    password => "${ELASTICSEARCH_PASSWORD}"
    ilm_enabled => false
    ssl => true
    ssl_certificate_verification => true
    cacert => "/usr/share/logstash/certs/root-ca.pem"
  }
  stdout {
    codec => rubydebug { metadata => true }
  }
}
```

4.3.5 Filebeat

Filebeat bude jako vedlejší část oddělená od hlavních částí. Deklarace verze a logování bude stejná jako u hlavních částí (viz. 4.3.1.1). Zde je ale nutné připojit i složku, kam Docker ukládá výstupy z aplikací, a samotný Docker socket pro zjišťování stavu spuštěných containerů.

Zdrojový kód 4.17: Definice služby - Filebeat

```
app-filebeat-1:
  <<: *defaults
  image: ${APP_FILEBEAT_IMAGE}
  entrypoint: "filebeat --strict.perms=false"
  container_name: "${APP_ID}-beats-1"
  environment:
    - TZ=Europe/Prague
    - strict.perms=false
    - LOGSTASH_HOST=${LOGSTASH_HOST}
    - LOGSTASH_PORT=${LOGSTASH_PORT}
    - SHIPPER_NAME=${SHIPPER_NAME}
    - SHIPPER_ENV=${SHIPPER_ENV}
    - SHIPPER_ADDITIONAL_TAGS=${SHIPPER_ADDITIONAL_TAGS}
  user: root
  volumes:
    - /var/lib/docker/containers:/usr/share/dockerlogs/data:ro
    - /var/run/docker.sock:/var/run/docker.sock
    - ./extras/filebeat/filebeat.yml:/usr/share/filebeat/filebeat.yml:ro
    - ./extras/filebeat/config:/usr/share/filebeat/config.extra
  extra_hosts:
    - "${LOGSTASH_HOST}:${DOCKER_HOST_MACHINE_IP}"
```

Připojení certifikátů bylo vynecháno, kompletní konfigurace je dostupná na přiloženém médiu.

4.3.5.1 Hlavní konfigurace

Hlavní konfigurační soubor udává, jak a kde bude Filebeat číst logové soubory. Dále definuje přídavné chování, rozhraní výstupu a jednoduchý preprocessing, než se výsledek odešle.

Detekce containerů, které mají být sledovány, probíhá navázáním se na Docker socket a sledováním spuštění a vypnutí containerů. Pro každý úspěšně detekovaný container je spuštěn worker pro sledování změn v jeho logovém souboru.

Cílem je číst logové soubory z Dockeru, proto bude deklarován provider typu `docker` se zapnutým hints-based autodiscover (viz. 3.1.4). Je třeba vypnout defaultní konfiguraci, pokud chceme nastavit vlastní. To ale bohužel není možné bez duplikace originálního pravidla pro objevování containerů

jako podmínku v sekci `condition`. Ta udává, že container musí mít label `co.elastic.logs/enabled` nastaven na `false`, aby container nebyl sledován.

Pro ty containery, které jsou povolené ke sběru logových zpáv, bude do paměti zkopírována celá konfigurace, která musí obsahovat identifikátor sledovaného containeru (odrážka `containers.ids`). Proměnou dokáže Filebeat substituovat za ID containeru.

Ke každé odeslané zprávě bude také automaticky připojena informace o prostředí, které je definovaná v souboru `.env` (viz. `fields.env`).

Zdrojový kód 4.18: Konfigurace Filebeat - hlavní konfigurace a detekce containerů

```
name: ${SHIPPER_NAME}
fields:
  env: ${SHIPPER_ENV}
filebeat.config_dir: /usr/share/filebeat/config.extra
filebeat.autodiscover:
  providers:
    - type: docker
      hints.enabled: true
      templates:
        - condition:
            not.equals:
              docker.container.labels.co.elastic.logs/enabled: 'false'
          config:
          - type: docker
            tags: ${SHIPPER_ADDITIONAL_TAGS}
            containers:
              path: '/usr/share/dockerlogs/data'
              stream: 'all'
              ids:
                - '${data.docker.container.id}'
              cri.parse_flags: true
              combine_partial: true
              exclude_files: ['\.gz$']
```

Processory slouží k dalšímu zpracování logových zpráv před odesláním do Logstashe. Díky tomu, že je využíván vstup typu `docker`, jsou ke každé zprávě připojeny informace o Dockeru jako je název image, identifikátor containeru a jeho labels. Některé Docker informace ale chybí, ty sou poté přidány pomocí processoru `add_docker_metadata`. Tím vzniknout duplicitní data, které je záhodno odstranit, k čemuž slouží processor `drop_fields`.

Zdrojový kód 4.19: Konfigurace Filebeat - processory

```
processors:
  - add_docker_metadata:
      host: 'unix:///var/run/docker.sock'
  - drop_fields:
      fields: ['docker']
```

Poslední částí konfigurace je nastavení výstupu. Jak již bylo řečeno, Filebeat bude odesílat data do Logstashe k dalšímu zpracování, což je zajištěno správným napojením na Logstash a připojení certifikátů pro ověření identity.

Zdrojový kód 4.20: Konfigurace Filebeat - výstup

```
output.logstash:
  hosts: ['${LOGSTASH_HOST}:${LOGSTASH_PORT}']
  ssl.enabled: true
  ssl.certificate_authorities:
    - '/usr/share/filebeat/certs/root-ca.pem'
  ssl.certificate: '/usr/share/filebeat/certs/logstash.pem'
  ssl.key: '/usr/share/filebeat/certs/logstash-key.pem'
```

4.3.6 Zabezpečení

Pro zabezpečení celé aplikace bude použito pluginu Security pro Elasticsearch, který plní vlastní funkci zabezpečení pomocí systému rolí, uživatelů a práv. Pro Kibanu slouží ke skrytí částí, ke kterým uživatelé nemají přístup. Dále přidává menu k ručnímu nastavení práv bez nutnosti měnit konfigurační soubory.

Všechny konfigurační soubory týkající se zabezpečení budou uloženy ve složce projektu hlavních částí v `/extras/elasticsearch/security` (viz. 4.1).

4.3.6.1 Elasticsearch

Prvním krokem pro zabezpečení databáze je definice všech potřebných objektů.

Základním stavebím kamenem jsou uživatelé. Systém bude mít několik přednastavených uživatelů s vygenerovaným náhodným heslem – tzv. interní – a poté možnost se přihlásit pomocí doménového uživatelského jména a hesla z organizačního LDAP – externí.

Interní uživatelé budou `admin` pro správu nastavení a separátně `kibanaserver` pro možnost přístupu Kibany samotné k datům z databáze (viz. 4.3.3.1).

Zdrojový kód 4.21: Zabezpečení - uživatelé
`internal_users.yml`

```
admin:
  hash: "$2a$12$VcCDgh2NDk07JGN0rjGbM.Ad41qVR/YFJcgHp0UGns5JDymv..TOG"
  reserved: true
  backend_roles:
  - "admin"
  description: "Admin user"

kibanaserver:
  hash: "$2a$12$4AcgAt3xwOWadA5s5blL6ev39OXDNhmOesEoo33eZtrq2N0YrU3H."
  reserved: true
  description: "Kibana server user"
```

Rolím mohou být přiřazeny jakékoliv kombinace pravomocí včetně jejich omezení na ty nejmenší části – jednotlivá pole –, přes zprávy, indexy až po celý systém a prostory.

V základu je vytvořená role `default`, která se následně přiřadí všem uživatelům. Jsou zde i dvě systémové role, kterým jsou pravomoce přiřazeny automaticky, ale je nutné je definovat. `kibana_read_only` má práva pouze na zobrazování dashboardů a vizualizací, `security_rest_api_access` poté pro možnost měnit nastavení zabezpečení přes REST API. Tyto dvě role se používají primárně v interní komunikaci.

Zdrojový kód 4.22: Zabezpečení - role
`roles.yml`

```
default:
  reserved: true
  cluster_permissions:
    - "read"
    - "search"
  tenant_permissions:
    - tenant_patterns:
      - "global_tenant"
      allowed_actions:
      - "kibana_all_write"

kibana_read_only:
  reserved: true

security_rest_api_access:
  reserved: true
```

Pojivem mezi rolemi a uživateli jsou tzv. role mappings – mapování rolí. Místo obecně standardního systému přiřazení role uživateli to zde funguje

obráceně. Jedné roli je definováno, kteří uživatelé dle uživatelského jména, hostname nebo backendových rolí[13] dostanou tuto roli.

Zde je definováno přiřazení základních pravomocí pomocí role `default` pro všechny uživatele, přístup ke všemu pro administrátora a nakonec role `kibana_server` pro potřebné pravomoce pro Kibanu.

Zdrojový kód 4.23: Zabezpečení - mapování rolí `roles_mapping.yml`

```
default:
   reserved: true
   users:
     - "*"

all_access:
  reserved: false
  backend_roles:
  - "admin"
  description: "Maps admin to all_access"

kibana_server:
  reserved: true
  users:
  - "kibanaserver"
```

Pro napojení přihlašování do domény organizace je zapotřebí nakonfigurovat authentikaci pomoci LDAP v souboru `config.yml`. Pokud soubor neexistuje, používá se pouze interní databáze uživatelů (viz. začátek této sekce). Tento soubor ale přepisuje stávající, proto je třeba definovat všechny používané typy přihlašování.

Pro LDAP backend je potřeba nakonfigurovat přihlašovací údaje a parametry pro vyhledávání uživatelů. Přihlášení probíhá tak, že se systém nejdříve sám přihlásí do LDAP, a posléze se pokusí vyhledat uživatele dle vyplněných údajů. To je z důvodu toho, jak LDAP funguje. Typicky se uživatel přihlašuje pomocí uživatelského jména a hesla. U LDAP je ale uživatelským jménem řetězec zvaný `bind`, který obsahuje celou cestu k objektu uživatele v systému. Aby mohlo být realizováno přihlašování pouze pomocí kombinace uživatelského jména a hesla, musí se systém nejdříve napojit do LDAP a uživatelský účet vyhledat ve správné složce. K tomu slouží další parametry, které určují, jakým způsobem má systém uživatele vyhledat a kterou vlastnost objektu použít jako uživatelské jméno dále v systému.

[13] textové řetězce, primárně využívané při authorizaci uživatelů externími systémy

Možné konfigurační hodnoty:

- `http_enabled` – Zda je metoda povolená pro HTTP přístup
- `transport_enabled` – Zda je metoda povolená pro TLS přístup
- `order` – Priorita (1 je nejvyšší)
- `http_authenticator.type` – Typ authentikace, může být basic, proxy, kerberos, pomocí certifikátu nebo JWT tokenu
- `http_authenticator.challenge` – Zda se zobrazí okno pro přihlášení nebo se systém pokusí údaje najít přímo v HTTP požadavku
- `type: ldap` – Typ backendu, lze použít interní nebo LDAP
- `bind_dn: "${env.LDAP_USERNAME}"` – Dotaz k nalezení přihlašovací identity
- `password: "${env.LDAP_PASSWORD}"` – Heslo k nalezené identitě
- `userbase: "OU=SBSUsers,OU=Users,OU=MyBusiness, DC=mediafactory,DC=cz"` – Základní uzel pro vyhledávání
- `usersearch: "(sAMAccountName={0})"` – Vyhledávací query, dosazuje uživatelské jméno zadané při přihlášení za {0}
- `username_attribute: "sAMAccountName"` – Název pole s uživatelským jménem pro identifikaci po přihlášení
- `hosts` – Seznam adres LDAP serveru

Zdrojový kód 4.24: Konfigurace zabezpečení - authentikace `config.yml`

```
config:
  dynamic:
    authc:
      internal_users:
        http_enabled: true
        transport_enabled: true
        order: 2
        http_authenticator:
          type: basic
          challenge: false
        authentication_backend:
          type: intern
      mf_ldap:
        http_enabled: true
        transport_enabled: true
        order: 1
        http_authenticator:
          type: basic
          challenge: true
        authentication_backend:
          type: ldap
          config:
            bind_dn: "${env.LDAP_USERNAME}"
            password: "${env.LDAP_PASSWORD}"
            userbase: "OU=SBSUsers,OU=Users,OU=MyBusiness,
            ↪  DC=mediafactory,DC=cz"
            usersearch: "(sAMAccountName={0})"
            username_attribute: "sAMAccountName"
            hosts:
              - "${env.LDAP_HOST}"
```

Další soubory nebudou obsahovat skupiny akcí ani prostory navíc, ale budou zaneseny do Gitu pro budoucí využití.

4.3.7 Upozornění

Upozornění nevyžaduje žádné konfigurační soubory. Je zde možnost konfigurace, ale pro účely organizace není potřeba tyto hodnoty upravovat.

4.3.8 Shrnutí

Byl sestaven kompletní projekt připravený ke spuštění na lokálním prostředí pomocí Dockeru. Dalším krokem je úspěšné nasazení aplikace do organizační infrastruktury. To bude realizováno systémovým administrátorem. Díky existujícímu systému continuous integration za využití Jenkins je zajištěné automatické nasazení aplikace.

Kapitola 5

Postupy

5.1 Analýza případů použití

5.1.1 Aktéři

Návštěvník Nepřihlášený uživatel, nemá přístup k žádné funkcionalitě kromě přihlášení samotného.

Vývojář Ve většině případů vývojář, ale může to být kdokoliv, kdo má k systému přístup, tedy standarní uživatel, který může vyhledávat a vytvářet vizualizace či dashboardy.

Administrátor Administrátor má navíc pravomoce ke správě systému a řídí, kdo má k čemu přístup.

5.1.2 Případy užití

„Use case, česky případ užití, je termín označující v oborech jako Human–Computer Interaction (interakce člověka s počítačem) či interakční design metodu popisu sekvence kroků, který vykonává uživatel při plnění konkrétního úkolu za použití interagujícího systému (konkrétního softwarového či hardwarového produktu). Takovým úkolem může být například vytvoření „určitého typu dokumentu v textovém editoru, nákup v internetovém obchodě, přečtení si článku na zpravodajském serveru i například výběr z bankomatu. Ve zvláštních případech může jít o interakci dvou strojů, z nichž jeden plní konkrétní úkol (například synchronizace zařízení).

Use case je složen ze sledu kroků, které uživatel vykonává, aby daný úkol splnil. Pro popis use case bývá užíván specializovaný jazyk Unified Modeling Language (UML), který formou diagramu popisuje jednotlivé akce uživatele a reakce systému, ale celou interakci je možné popsat i jednoduše slovně.“ [8]

5.1.2.1 U1 - Přihlášení doménovým účtem

Návštěvník se přihlašuje otevřením aplikace Kibana, která zobrazí formulář pro zadání uživatelského jména a hesla. Po odeslání údajů a ověření správnosti této kombinace proti organizačnímu LDAP severu je uživatel přihlášen.

Hlavní scénář

1. Návštěvník otevře aplikaci Kibana.
2. Návštěvník zadá své doménové uživatelské jméno a heslo.
3. Systém zkontroluje správnost zadaných údajů.
4. Návštěvník je přihlášen a přesměrován na úvodní obrazovku.

5.1.2.2 U2 - Zobrazení vzorů indexů

Vývojář si může zobrazit momentálně vytvořené vzory indexů.

Hlavní scénář

1. Vývojář se naviguje pomocí postranního panelu do sekce `Management` (⚙).
2. Vývojář se naviguje do správy vzorů indexů pomocí tlačítka `Index Patterns`.

5.1.2.3 U3 - Vytvoření a úprava vzoru indexů

Administrátor může vytvářet nové vzory indexů a z detailu upravovat stávající.

Požadavky Administrátor se nachází v menu `Index Patterns` (viz. **U2**).

Hlavní scénář - vytvoření

1. Administrátor vybere možnost pro vytvoření nového vzoru `Create index pattern`.
2. Administrátor vyplní vzor pro indexy, kde pomocí znaku `*` může indikovat libovolný počet znaků.
3. Administrátor vidí, které indexy budou obsaženy ve výsledném vzoru.
4. Administrátor potvrdí formulář.
5. Administrátor vybere pole, které se použije pro časovou značku.
6. Vzor indexů je vytvořen.

Vedlejší scénář - úprava

1. Administrátor vybere existující vzor kliknutím na jeho název
2. Administrátor může upravovat typy jednotlivých polí a přidávat nové

5.1.2.4 U4 - Obnova polí indexu

Administrátor může z detailu vzoru indexů obnovit indexovaná pole.

Požadavky Administrátor se nachází v menu `Index Patterns` (viz. **U2**).

Hlavní scénář

1. Administrátor vybere požadovaný vzor.
2. Administrátor obnoví pole vzoru pomocí tlačítka `Refresh field list` (🔄).

5.1.2.5 U5 - Zobrazení seznamu dashboardů

Vývojář může zobrazit seznam dashboardů.

Hlavní scénář

1. Vývojář se naviguje pomocí postranního panelu do sekce `Dashboard` (▦).

5.1.2.6 U6 - Správa dashboardu

Vývojář může vytvářet nové a upravovat nebo mazat stávající dashboardy. Jdou odtud upravovat i jednotlivé vložené vizualizaci.

Požadavky Vývojář se nachází v menu `Dashboard` (viz. **U5**).

Hlavní scénář - vytvoření

1. Vývojář vybere možnost pro vytvoření nového dashboardu `Create dashboard`.
2. Vývojář přidá existující nebo vytvoří nové vizualiace (viz. **U8**).
3. Vývojář dashboard uloží pomocí tlačítka `Save`.

Vedlejší scénář - úprava

1. Vývojář vybere možnost pro úpravu dashboardu vedle jeho jména v seznamu (✎).
2. Scénář pokračuje od kroku **2** hlavního scénáře.

5.1.2.7 U7 - Zobrazení seznamu vizualizací

Vývojář si může zobrazit seznam vizualizací.

Hlavní scénář

1. Vývojář se naviguje pomocí postranního panelu do sekce `Visualize` (🏠).

5.1.2.8 U8 - Správa vizualizace

Vývojář může po výběru typu vytvářet nové a upravovat nebo mazat stávající vizualizace.

Požadavky Vývojář se nachází v menu `Visualize` (viz. **U7**).

Hlavní scénář - vytvoření

1. Vývojář vybere možnost pro vytvoření nové vizualizace `Create visualization`.
2. Vývojář vybere typ vizualizace.
3. Vývojář vybere zdroj dat.
4. Vývojář nastaví vlastnosti vizualizace.
5. Vývojář vizualizaci uloží pomocí tlačítka `Save`.

Vedlejší scénář - úprava

1. Vývojář vybere možnost pro úpravu vizualizace vedle jejího jména v seznamu (✎).
2. Scénář pokračuje od kroku **4** hlavního scénáře.

5.1.2.9 U9 - Fulltextové vyhledávání

Vývojář může vyhledávat ve všech sekcích, kde to dává smysl. Tedy v menu vizualizací, dashboardů a samotného vyhledávání. Vyhledávat lze pomocí jazyka KQL nebo Lucene a mezi těmito způsoby přepínat.

Hlavní scénář

1. Vývojář se naviguje pomocí postranního panelu do sekce `Discover` (🧭).
2. Vývojář může nastavit filtrace, zobrazená pole, časový interval, vzor indexů a další.

5.1.2.10 U10 - Uložení vyhledávání

Vývojář může aktuální vyhledávání uložit.

Požadavky Vývojář se nachází v sekci, ve které je možné vyhledávat.

Hlavní scénář

1. Vývojář klikne na tlačítko pro uložení vyhledávání (🖫).
2. Vývojář klikne na tlačítko `Save current query`.
3. Vývojář vyplní název a potvrdí.
4. Vyhledávání je uloženo.

5.1.2.11 U11 - Zobrazení seznamu monitorů

Vývojář může zobrazit seznam monitorů a odtud zobrazit jejich detail.

Hlavní scénář

1. Vývojář se naviguje pomocí postranního panelu do sekce `Alerting` (A).
2. Vývojář vybere záložku `Monitors`.

5.1.2.12 U12 - Správa monitoru

Vývojář může vytvářet nové a upravovat nebo mazat stávající monitory. Lze také zobrazit seznam triggerů nebo upozornění, které jdou potvrdit.

Požadavky Vývojář se nachází v menu `Alerting` v záložce `Monitors` (viz. **U11**).

Hlavní scénář - vytvoření

1. Vývojář klikne na tlačítko `Create monitor`.
2. Vývojář vyplní název, typ definice metriky monitoru, indexy, časové pole a interval kontroly metriky.
3. Vývojář uloží monitor pomocí tlačítka `Create`.

Vedlejší scénář - úprava

1. Vývojář přejde do detailu monitoru kliknutím na jeho název v seznamu.
2. Vývojář klikne na tlačítko `Edit`.
3. Scénář pokračuje v kroku **2** hlavního scénáře.

Vedlejší scénář - vypnutí, zapnutí, smazání

1. Vývojář klikne na rozšířené možnosti u monitoru v seznamu (⋮).
2. Vývojář vybere možnost `Delete` pro smazání, `Enable` pro zapnutí nebo `Disable` pro vypnutí.

5.1.2.13 U13 - Zobrazení seznamu triggerů

Vývojář může z detailu monitoru zobrazit seznam triggerů.

Požadavky Vývojář se nachází v menu `Alerting` v záložce `Monitors` (viz. **U11**).

Hlavní scénář

1. Vývojář přejde do detailu monitoru kliknutím na jeho název v seznamu.
2. Detail monitoru obsahuje sekci `Triggers`.

5.1.2.14 U14 - Správa triggeru

Vývojář může vytvářet nové a upravovat nebo mazat stávající triggery monitoru.

Požadavky Vývojář se nachází v detailu monitoru (viz. **U13**).

Hlavní scénář - vytvoření

1. Vývojář klikne na tlačítko `Create trigger`.
2. Vývojář vyplní název, vážnost a podmínku výše metriky.
3. Vývojář definuje jednu čí více akcí, u každé vyplní její název, cíl, předmět a obsah zprávy, popřípadě povolí limit.
4. Vývojář vytvoří trigger pomocí tlačítka `Create`.

Vedlejší scénář - úprava

1. Vývojář vybere zaškrtávacím polem trigger, který chce upravit.
2. Vývojář klikne na tlačítko `Edit`.
3. Scénář pokračuje krokem **2** hlavního scénáře.

Vedlejší scénář - smazání

1. Vývojář vybere zaškrtávacím polem jeden či více triggerů, které chce smazat.
2. Vývojář klikne na tlačítko `Delete`.

5.1.2.15 U15 - Zobrazení seznamu cílů

Vývojář může zobrazit seznam cílů a odtud zobrazit jejich detail.

Hlavní scénář

1. Vývojář se naviguje pomocí postranního panelu do sekce `Alerting` (A).
2. Vývojář vybere záložku `Destinations`.

5.1.2.16 U16 - Správa cíle

Administrátor může vytvářet nové a upravovat nebo mazat stávající cíle.

Požadavky Administrátor se nachází v seznamu cílů (viz. **U15**).

Hlavní scénář - vytvoření

1. Administrátor klikne na tlačítko `Add destination`.
2. Administrátor vyplní název a vybere typ.
3. Administrátor vyplní další údaje vzhledem k typu, který vybral.
4. Administrátor uloží cíl kliknutím na tlačítko `Create`.

Vedlejší scénář - úprava

1. Vývojář klikne na tlačítko `Edit` vedle cíle v seznamu cílů.
2. Scénář pokračuje v kroku **2** hlavního scénáře.

Vedlejší scénář - smazání

1. Vývojář klikne na tlačítko `Delete` vedle cíle v seznamu cílů.

5.1.2.17 U17 - Zobrazení zabezpečení

Administrátor může zobrazit menu zabezpečení.

Hlavní scénář

1. Vývojář se naviguje pomocí postranního panelu do sekce `Security` (🔒).

5.1.2.18 U18 - Správa uživatelů

Administrátor může vytvářet nové a upravovat nebo mazat stávající interní uživatele. Nelze jakkoliv měnit detaily uživatelů přihlašujících se přes doménový účet.

Požadavky Administrátor se nachází v menu zabezpečení (viz. **U17**) a sekci `Internal User Database`.

Hlavní scénář - vytvoření

1. Administrátor klikne na tlačítko vytvoření (+).
2. Administrátor vyplní uživatelské jméno, heslo, přiřadí role, backend role a atributy.
3. Administrátor uloží uživatele kliknutím na tlačítko `Submit`.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro editaci (✎) nebo pro kopírování (⧉).
2. Scénář pokračuje v kroku **2** hlavního scénáře.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro smazání (🗑).

5.1.2.19 U19 - Správa prostorů

Administrátor může vytvářet nové a upravovat nebo mazat stávající prostory.

Požadavky Administrátor se nachází v menu zabezpečení (viz. **U17**) a sekci `Tenants`.

Hlavní scénář - vytvoření

1. Administrátor klikne na tlačítko vytvoření (+).
2. Administrátor vyplní identifikátor.
3. Administrátor uloží prostor kliknutím na tlačítko `Submit`.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro editaci (🖉) nebo pro kopírování (🗐).
2. Scénář pokračuje v kroku **2** hlavního scénáře.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro smazání (🗑).

5.1.2.20 U20 - Správa skupin akcí

Administrátor může vytvářet nové a upravovat nebo mazat stávající skupiny akcí.

Požadavky Administrátor se nachází v menu zabezpečení (viz. **U17**) a sekci `Action Groups`.

Hlavní scénář - vytvoření

1. Administrátor klikne na tlačítko vytvoření (+).
2. Administrátor vyplní identifikátor akční skupiny.
3. Administrátor vybere jednu nebo více dalších akčních skupin nebo jednotlivých práv.
4. Administrátor uloží akční skupinu kliknutím na tlačítko `Submit`.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro editaci (🖉) nebo pro kopírování (🗐).
2. Scénář pokračuje v kroku **3** hlavního scénáře.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro smazání (🗑).

5.1.2.21 U21 - Správa mapování rolí

Administrátor může vytvářet nové a upravovat nebo mazat stávající mapování rolí.

Požadavky Administrátor se nachází v menu zabezpečení (viz. **U17**) a sekci `Role Mappings`.

Hlavní scénář - vytvoření

1. Administrátor klikne na tlačítko vytvoření (+).
2. Administrátor vybere roli, přidá libovolný počet vzorů uživatelů, backend rolí a hostitelů.
3. Administrátor uloží mapování role kliknutím na tlačítko `Submit`.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro editaci (✎) nebo pro kopírování (⎘).
2. Scénář pokračuje v kroku **2** hlavního scénáře.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro smazání (🗑).

5.1.2.22 U22 - Správa rolí

Administrátor může vytvářet nové a upravovat nebo mazat stávající rolí.

Požadavky Administrátor se nachází v menu zabezpečení (viz. **U17**) a sekci `Roles`.

Hlavní scénář - vytvoření

1. Administrátor klikne na tlačítko vytvoření (+).
2. Administrátor vyplní identifikátor role.
3. Administrátor přepne záložku na `Cluster Permissions` a nastaví globální práva pomocí skupin akcí a jednotlivých práv.
4. Administrátor přepne záložku na `Index Permissions` a nastaví práva pro odpovídající indexy pomocí vzoru výběrem skupin akcí a jednotlivých práv.
5. Administrátor přepne záložku na `Tenant Permissions` a nastaví práva pro prostory odpovídající vzoru práva výběrem skupin akcí a jednotlivých práv.
6. Administrátor uloží roli kliknutím na tlačítko `Submit`.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro editaci (✎) nebo pro kopírování (⎘).
2. Scénář pokračuje v kroku **3** hlavního scénáře.

Vedlejší scénář - úprava, kopie

1. Administrátor klikne na tlačítko pro smazání (🗑).

5.1.2.23 U23 - Exportování objektů

Vývojář může exportovat uložené objekty, např. dashboard, vizualizace ale i uložená vyhledávání, nastavení a vzory indexů.

Požadavky Vývojář se nachází v menu Management (viz. **U2**).

Hlavní scénář

1. Vývojář se naviguje do správy uložených objektů pomocí tlačítka `Saved Objects`.
2. Vývojář vybere objekty, které chce exportovat.
3. Vývojář klikne na tlačítko `Export`.
4. Prohlížeč automaticky stáhne exportované objekty jako soubor.

5.1.2.24 U24 - Importování objektů

Vývojář může exportované objekty importovat.

Požadavky Vývojář se nachází v sekci uložených objektů (viz. **U25**) a má soubor exportovaných objektů (viz. **U23**).

Hlavní scénář

1. Vývojář se naviguje do správy uložených objektů pomocí tlačítka `Saved Objects`.
2. Vývojář klikne na tlačítko `Import`.
3. Vývojář vybere soubor exportovaných objektů.
4. Vývojář odešle formulář kliknutím na tlačítko `Submit`.
5. Objekty jsou importovány.

5.1.2.25 U25 - Zobrazení seznamu uložených objektů

Vývojář může zobrazit seznam uložených objektů a zobrazit si jejich detail.

Požadavky Vývojář se nachází v menu Management (viz. **U2**).

Hlavní scénář

1. Vývojář se naviguje do uloženách objektůpomocí tlačítka `Saved Objects`.

5.1.2.26 U26 - Přepnutí prostoru

Vývojář může přepnout aktuální prostor, zobrazit si jeho vizualizace a dashboardy.

Hlavní scénář

1. Vývojář se naviguje pomocí postranního panelu do sekce `Tenants` (👤).

Vývojář může zobrazit historii upozornění a potvrdit aktivní.

Hlavní scénář

1. Vývojář se naviguje pomocí postranního panelu do sekce `Alerting` (A).
2. Vývojář vybere záložku `Dashboard`.

5.1.2.27 U28 - Potvrzení upozornění

Vývojář může aktivní upozornění potvrdit z detailu monitoru nebo ze seznamu všech upozornění.

Požadavky - hlavní scénář Vývojář se nachází v sekci `Alerting` (viz. **U11**).

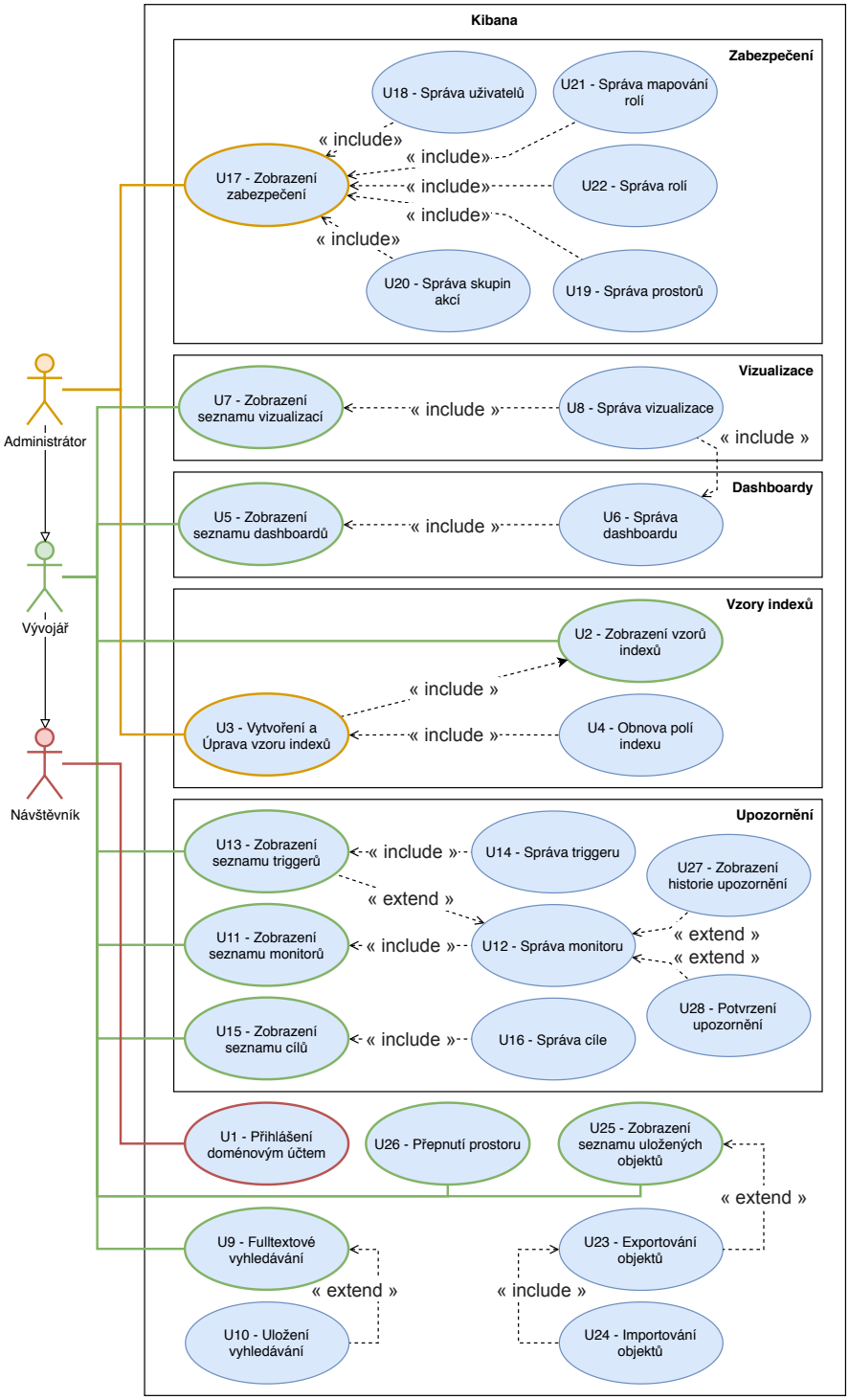
Hlavní scénář

1. Vývojář vybere upozornění, které chce potvrdit.
2. Vývojář klikne na tlačítko `Acknowledge`.
3. Upozornění se označí jako potvrzené.

Požadavky - vedlejší scénář Vývojář se nachází v detailu monitoru (viz. **U27**).

Vedlejší scénář

1. Scénář je identický s hlavním scénářem.



Obrázek 5.1: Diagram případů užití

5.2 Kibana

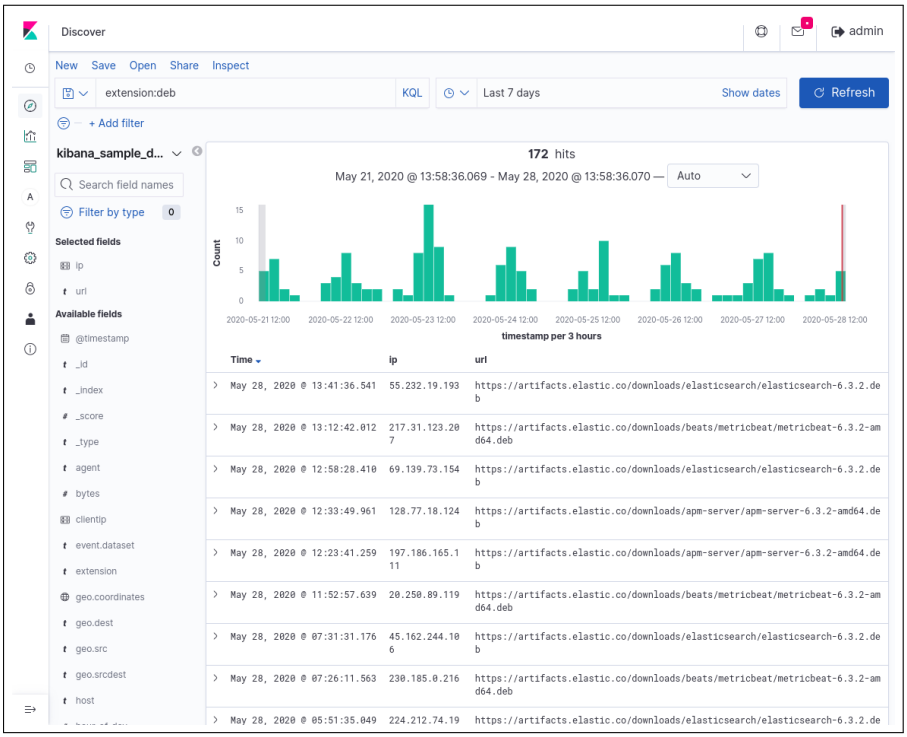
5.2.1 Vyhledávání

Vyhledávání kombinuje několik možností filtrování. Před vyhledáváním je potřeba zkontrolovat, zda je vybraný správný vzor. V základu je vybrán časový rozsah „posledních 15 minut“, což u vzorů, které jsou aktivní, stačí, ale je záhodno tuto hodnotu měnit dle potřeby.

V levé části se poté nachází výběr polí. Pokud není vybráno žádné pole, zpráva se ve vyhledávání ukazuje celá. Často je možných polí více než sto, proto mezi nimi lze vyhledávat dle názvu, typu a dalších vlastností.

Vyhledávání dle hodnot polí je definováno v horní části. Lze využít vyhledávací jazyk KQL[14] nebo – pro zajištění zpětné kompatibility se staršími verzemi – lze přepnout na jazyk Lucene[15].

Relevantní případy použití: **U9**



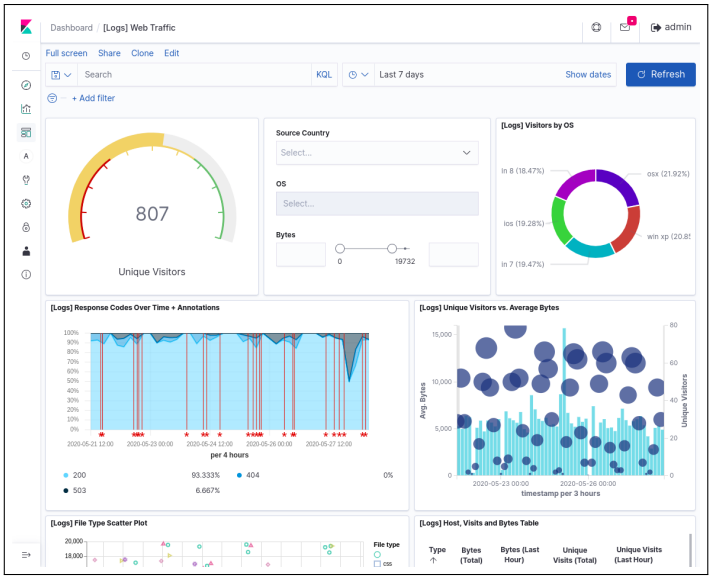
Obrázek 5.2: Příklad vyhledávání zobrazující pole `ip` a `url`, filtruje pouze požadavky s příponou `.dev` sedm dní zpátky

[14] `https://www.elastic.co/guide/en/kibana/7.6/kuery-query.html`
[15] `https://lucene.apache.org/core/2_9_4/queryparsersyntax.html`

5.2.2 Dashboardy

Dashboardy vznikají složením několika uložených vizualizací. Na všechny vizualizace v dashboardu se aplikuje momentálně nastavený filtr na v horní části obrazovky (viz. obrázek 2.2) jako zdroj dat. Dashboardy lze ukládat i s nastaveným datovým filterem, který se po jeho otevření aplikuje.

Relevantní případy pužití: **U5**, **U6**



Obrázek 5.3: Příkladný dashboard zobrazující informace o webových požadavcích.

5.2.3 Vizualizace

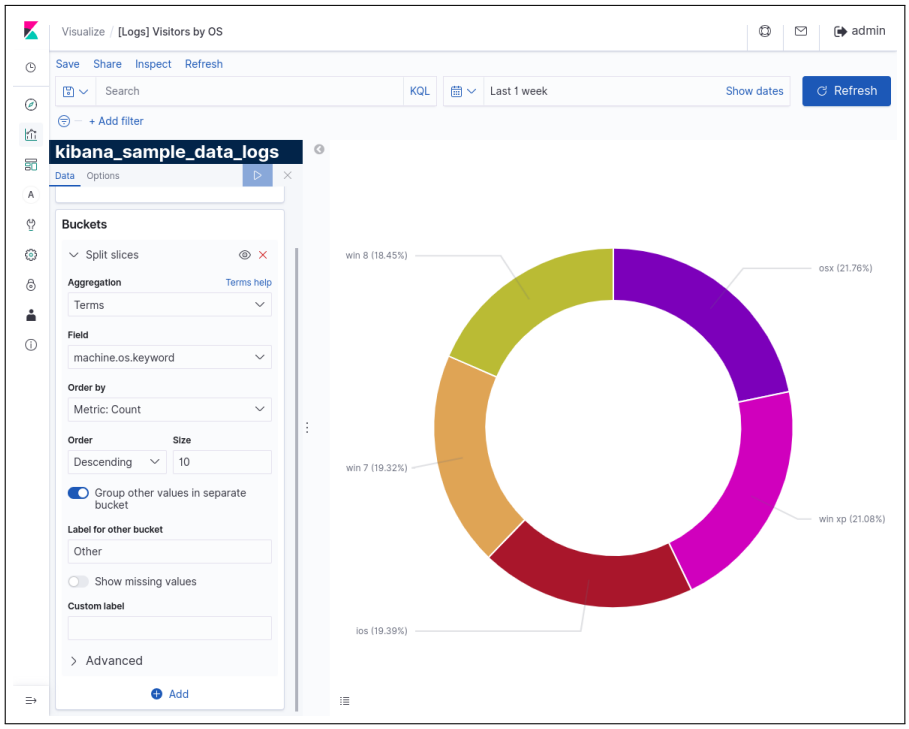
„Vizualizace v Kibaně jsou postavené na agregacích dat z Elasticsearche. Nezáleží na tom, o jaký druh dat jde. Mohou to být geografická data, data o komunikaci v síti, výsledky voleb a další.“[9]

Vizualizace je možné agregovat v libovolném počtu dashboardů.

Je možné si vybrat z následujících typů vizualizací:

- Vertical Bar – Sloupcový graf
- Horizontal Bar – Pruhový graf
- Area – Plošný spojnicový graf
- Line – Spojnicový graf
- Pie – Koláčový graf
- Gauge – Budík zobrazující kam spadá hodnota na určeném spektru
- Goal – Budík zobrazující hodnotu vůči absolutnímu cíli
- Metric – Zobrazení výsledku výrazu jako jedno číslo
- Coordinate Map – Mapa s možností vizualizace geografických dat pomocí zemepisných souřadnic
- Heat Map – Teplotní mapa
- Region Map – Mapa s možností zanesení dat do států, krajů a měst
- Controls – Ovládací prvky pro jednoduchou filtraci vizualizovaných dat, podporuje dropdown a rozsahy
- Data Table – Standardní tabulka s daty
- Markdown – Statický dokument v jazyce Markdown
- Tag Cloud – Skupina dat uspořádaná velikostně dle určené metriky
- TSVB – Zobrazení metriky v čase pomocí vizuálního rozhraní pro správu pipelines
- Timelion – Vizualizace dat pomocí funkčních výrazů
- Vega – Pokročilá vizualizace pomocí deklarativního formátu Vega a Vega-Lite[16]

[16] více na `https://vega.github.io/`



Obrázek 5.4: Příkladná vizualizace typu `Pie`

5.2.4 Prostory

Tenants jsou virtuální prostory (viz. 4.3.3.1), které jsou oddělené od ostatních prostorů a na každý se vztahují vlastní práva.

Každý uživatel má v základu dostupný a vybraný globální prostor a proto vše, co vytvoří, se sdílí ostatním uživatelům. Je možné se přepnout do soukromého prostoru, odkud lze posléze všechny vytvořené objekty sdílet.

Relevantní případy použití: **U19**, **U26**

5.2.5 Sdílení objektů

Uložené vizualizace, dashboardy, vyhledávání a další jsou označovány souhrnným typem `Saved Objects`. Lze je všechny zobrazit přes menu Management dostupné čtvrtou ikonou od konce (⚙) v postranním menu a dále kliknutím na `Saved Objects`. Každý objekt je uložený v prostoru, ve kterým byl vytvořen, a je možné je mezi prostory sdílet.

Relevantní případy použití: **U23**, **U24**, **U25**

5.2.5.1 Export

Pro sdílení objektů mezi prostory je potřeba zaškrtnout objekt, který chceme sdílet. Lze vybrat více objektů a pro urychlení procesu sdílení více objektů najednou. Dalším krokem je kliknutí na tlačítko `Export` v levém horním rohu nad seznamem objektů, kde se doporučuje povolit nastavení `Include related objects`. V případě, že se nějaký objekt odkazuje na další objekty – typicky dashboardy na vyhledávání, vizualizace a indexy – by se v případě, že se v daném prostoru nenachází, neimportoval. Po těchto krocích stačí kliknout na tlačítko `Export` a prohlížeč automaticky stáhne soubor ve formátu JSON.

5.2.5.2 Import

Po úspěšném exportu objektů a získání jejich definice v souboru ve formátu JSON je lze ve stejném menu – po přepnutí prostoru – importovat. To lze učinit kliknutím na text `Import` v levém horním rohu nad seznamem objektů. Po kliknutí se zobrazí postranní menu s výběrem souboru a nastavením, zda přepsat existující objekty, pokud již byly v minulosti importovány. Po vybrání souboru stačí kliknout na modré tlačítko `Import` v pravém dolním rohu, vyčkat na úspěšné vykonání operace a potvrdit modrým tlačítkem `Done`, které se objeví namísto tlačítka pro odeslání importu. Seznam uložených objektů se následně obnoví a bude obsahovat importované objekty.

5.2.5.3 Externí sdílení

Objekty je také možné sdílet externě. Nabízí se zde možnost přímého odkazu na objekt nebo vložení do stránky pomocí iFrame. Prvním krokem ke sdílení objektu touto cestou je zobrazení si objektu, který chceme sdílet. Po kliknutí na modrý text `Share` v levém horním rohu se naskytne několik možností.

Embed code Výsledkem je HTML kód obsahující iFrame, který při zobrazení načte sdílený objekt.

Permalinks Výsledkem je URL adresa odkazující na sdílený objekt.

Oba způsoby sdílení mají na výběr mezi dvěma možnosti.

Snapshot Zobrazí objekt s daty tak, jak vypadal v době sdílení a jeho úpravy vzhled nemění.

Saved object Zobrazí objekt tak, jak momentálně vypadá, a jeho změny se projeví při opakovaném zobrazení.

Je zde i možnost zkrácení URL z důvodu přehlednosti a kompatibility mezi prohlížeči a ostatními softwarem či službami.

5.2.6 Zabezpečení

Zabezpečení se skládá z několika hlavních objektů, které mezi sebou mají složité relace. Všechno nastavení je konfigurovatelné nebo je možné ho změnit z uživatelského rozhraní.

Relevantní případy použití: **U17**.

5.2.6.1 Objekty

Většina objektů v systému zabezpečení je navázána pomocí názvu a jednoznačných identifikátorů. U většiny nastavení pak lze používat wildcards k navázání více objektů jiného typu. Výhodou tohoto systému je jednoduché propojení objektů bez nutnosti se starat o jiné unikátní identifikátory nebo kolize.

Uživatel Uživatel je kdokoliv, kdo má možnost se do systému přihlásit. Systém nemusí o každém uživateli vědět ani si ho pamatovat po tom, co se odhlásí. Všechna práva, vlastnictví a ostatní navázání se vážou na uživatelské jméno, proto je v případě přihlašování pomocí externích systémů nutné zajistit jeho unikátnost a zároveň je díky této funkci možné se přihlásit více způsoby.

Relevantní případy pužití: **U18**.

Action Akce, nebo také pravomoce (permissions), nejsou samostatně definované ani není možné je přidat. Nejsou to objekty jako takové, existují pouze jako textové identifikátory a každá akce povoluje nějakou funkci systému.

Action group Skupina akcí je z pohledu systému samostatná akce, která agreguje více akcí pod jeden textový identifikátor. Povolením této akce se automaticky povolí všechny podřazené.

Relevantní případy použití: **U20**.

Role Role jsou základním stavebním prvkem pro přiřazení pravomocí uživatelům. Každá role má tři možnosti nastavení rolí a každá možnost má přiřazeno více akcí. Akce mohou být základní, definované systémem, nebo lze použít skupiny akcí, vytvořené ručně.

Cluster Práva se aplikují globálně na všechny indexy.

Index Práva platí pouze na ty indexy, jejichž název odpovídá alespoň jednomu vzoru.

Tenants Práva platí pouze na objekty (vizualizace, dashboard atp.) a akce v prostoru odpovídajícímu alespoň jednomu vzoru.

Relevantní případy použití: **U22**.

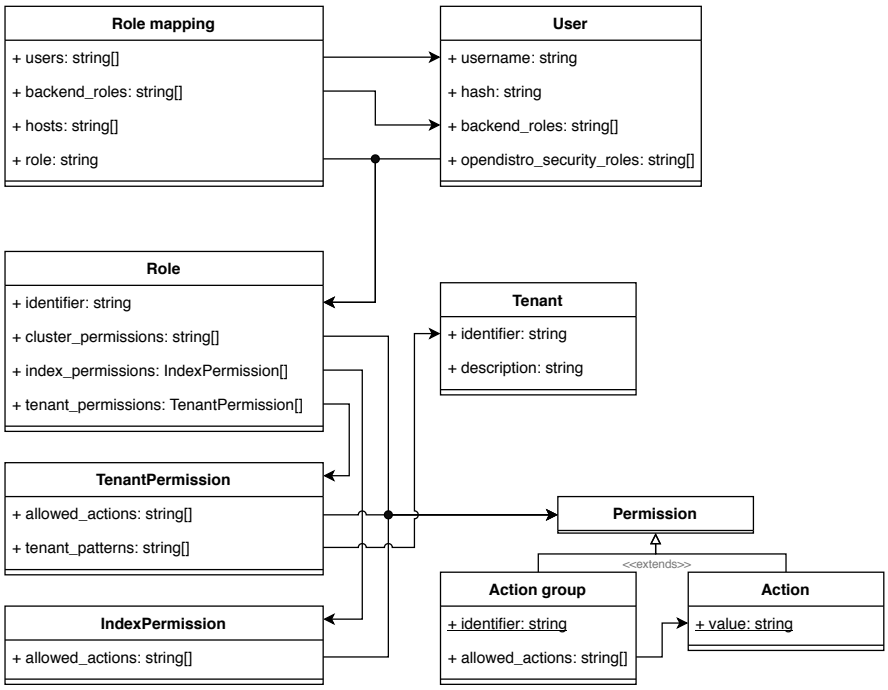
Roles mapping Mapování rolí mapuje role na uživatele dle několika typů. Pro každý typ může být definováno několik vzorů.

User Role je aplikována na všechny uživatele jejichž uživatelské jméno odpovídá alespoň jednomu vzoru.

Backend role Role je aplikována na všechny uživatele jejich alespoň jedna backend role odpovídá alespoň jednomu vzoru.

Host Role je aplikována na všechny uživatele jejichž IP adresa nebo název hostitele odpovídá alespoň jednomu vzoru.

Relevantní případy pužití: **U21**.



Obrázek 5.5: Relace mezi jednotlivými objekty zabezpečení

5.2.7 Upozornění

Pro vytvoření upozornění je třeba několik kroků. Prvním krokem je vytvoření cíle (destination). Ty udávají, kam se výsledná zpráva pošle. Na výběr je Amazon Chime, Slack a Custom Webhook. Pro potřeby organizace bude využit převážně Slack a dle potřeby i Custom Webhook, který dokáže poslat HTTP(S) požadavek na danou URL adresu s určenými parametry. To bude napojeno na jednoduchý server, který bude posílat e-maily na dané adresy.

Po vytvoření cíle je dalším krokem vytvoření monitoru. Zde se definuje dotaz, jehož výstup je vždy číslo – metrika. Je možné použít filtraci dokumentů, vybrat libovolné indexy (zde je výběr ze všech indexů a existující definované index patterns se zde nedají vybrat) a časový rozsah.

Relevantní případy pužití: **U11**, **U12**, **U13**, **U14**, **U15**, **U16**, **U27**, **U28**.

Možné funkce pro výpočet metriky jsou:

- `count()` – počet nalezených dokumentů
- `average()` – aritmetický průměr vybraného pole
- `sum()` – suma vybraného pole
- `min()` – minimální hodnota z hodnot vybraného pole
- `max()` – maximální hodnota z hodnot vybraného pole

Na konci formuláře je výběr časového intervalu, jak často se bude monitor obnovovat a tlačítko na samotné vytvoření.

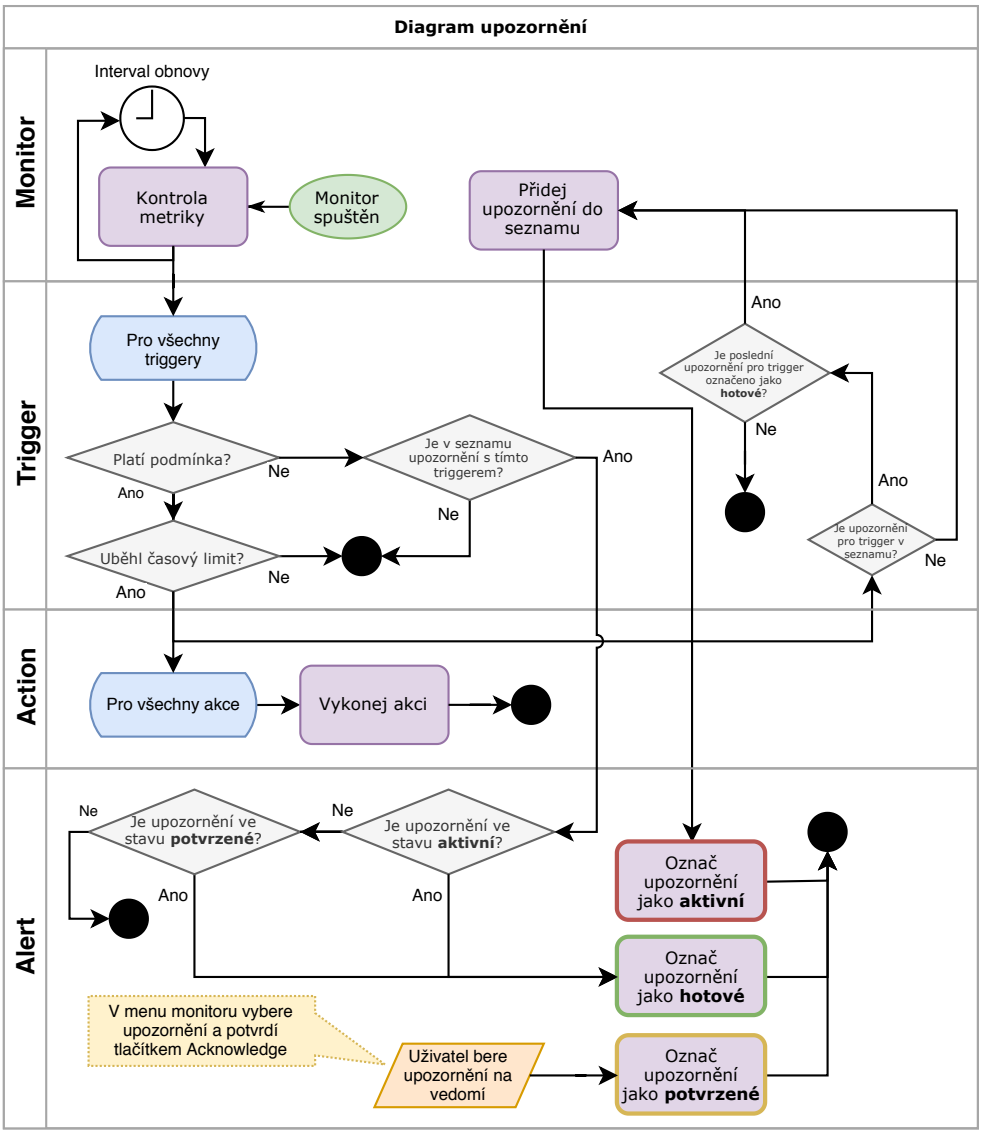
Po vytvoření se při každém obnovení zkontrolují všechny triggery. Ty udávají, co se má stát, když je metrika monitoru větší, menší nebo stejná jako dané číslo (položka Trigger condition – podmínka). Hodnota metriky u triggeru udává vážnost, kde 1 je nejmenší a 5 je největší. To lze využít například pro další zpracování na straně webhooku.

Akce triggeru udávají, co se stane, když je splněna podmínka. Lze si personalizovat předmět i zprávu pomocí templatovacího jazyk Mustache[17], který jednoduše dosazuje hodnoty proměnných do textu pomocí {{...}}.

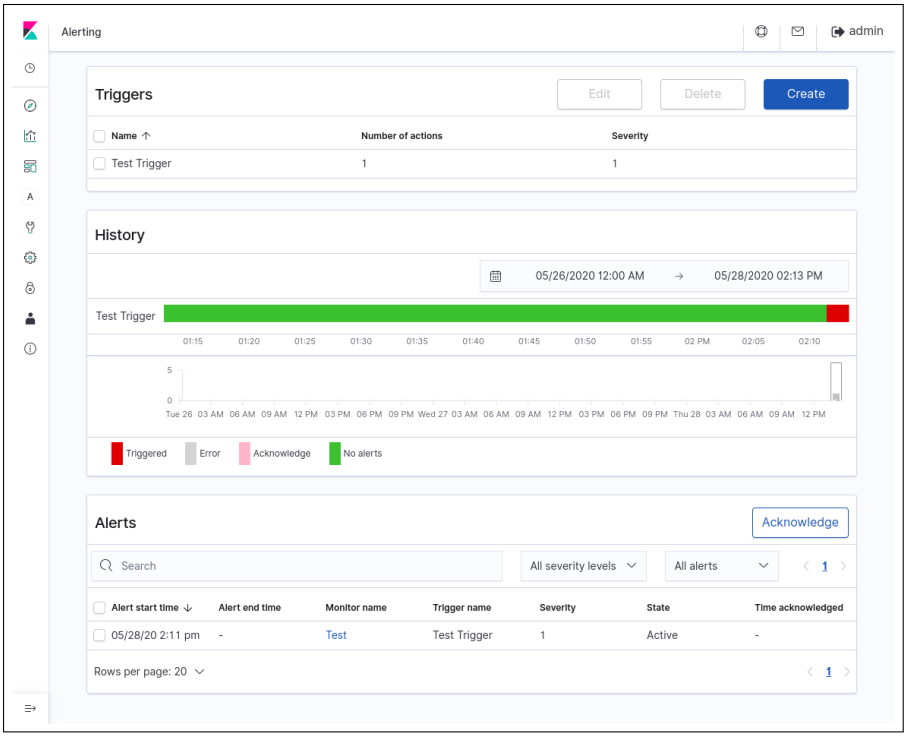
[17] více na `https://mustache.github.io/mustache.5.html`

Ve zprávě je dostupná proměnná `ctx` která obsahuje následující informace:[10]

- `ctx.results` – pole o jednom prvku obsahující výsledek
- `ctx.monitor`
  - `.name` – název
  - `.type` – typ monitoru
  - `.enabled` – zda je monitor zapnut
  - `.enabled_time` – kdy byl monitor zapnut
  - `.schedule` – interval obnovy monitoru
  - `.inputs` – pole vstupů
  - `.triggers` – pole triggerů (struktura každého prvku viz. `ctx.trigger` níže)
  - `.last_update_time` – poslední čas obnovy monitoru
- `ctx.trigger` – trigger, který zprávu spustil
  - `.name` – název
  - `.severity` – vážnost
  - `.condition` – podmínka
  - `.actions` – pole akcí
- `ctx.periodStart` – čas, kdy byl poslední trigger zaznamenán
- `ctx.periodEnd` – čas, kdy byl poslední trigger potvrzen, pokud probíhá tak `null`
- `ctx.error` – v případě chyby obsahuje text chybové hlášky, jinak `null`
- `ctx.alert` – aktivní trigger, jinak `null`
  - `.id` – identifikátor
  - `.version` – verze
  - `.isAcknowledged` – je-li potvrzený



Obrázek 5.6: Flowchart upozornění



Obrázek 5.7: Stav monitoru, kdy je splněna podmínka triggeru a je vytvořeno upozornění

Kapitola 6

Zhodnocení

Výsledný systém je plně funkční a je v souladu s hlavními i vedlejšími požadavky organizace. V průběhu jeho životnosti se bude nadále vyvíjet, přibudou nové funkce, a časem se stane nedílnou součástí při vývoji a monitorování vyvíjených aplikací.

6.1 Centralizovaný přístup a historizace

Centralizovaný přístup s jednoduchou správou pravomocí nahradí dosud používané, zastaralé způsoby pro přístup k logovým zprávám a zajistí možnost vizualizace aspektů aplikací. To ocení nejen vývojáři ale také ostatní zaměstnanci organizace, jako systémoví administrátoři, kteří mohou využít funkcionality ke sledování zatížení systému díky dalším rozšířením a zdroje útoků či zpomalení jednotlivých částí. Správci aplikací dále ocení možnost prezentovat klientům reálná data v přehledné podobě.

Historizace zpráv zajišťuje možnost zpětně dohledat důležité milníky, kdy potencionální či konkrétní chyba nastala, a co jí přecházelo. Jednoduché odstranění historie za účelem uvolnění místa a jednotné úložiště sníží náklady na infrastrukturu a čas vynaložený na uvolnění místa.

6.2 Technologie a dokumentace

Správný výběr technologie zajistil budoucí rozvoj ostatními vývojáři a bude postupně nahrazovat dosavadní způsoby jako nástroj na čtení logových zpráv. Díky rozsáhlému ekosystému lze implementovat mnoho rozšíření pro přidanou hodnotu a potřebné informace.

6.3 Zabezpečení

Systém je zabezpečen vůči přístupu nepovolaným osobám a pouze aplikace, které vlastní certifikát, mohou odesílat zprávy do systému. Tento způsob je ale náchylný na přístup přímo k soubory pomocí FTP, které bývá chráněno pouze heslem. Tomu ale jen těžko zabránit a počítá se, že posílat falešné logové zprávy nikomu radost neudělá.

6.4 Do budoucna

Budoucí pozice systému v rámci organizace je zajištěná na dlouhou dobu. Pomalu se bude upouštět od zastaralých způsobů a čím více budou vývojáři systém používat, tím efektivnější bude jejich práce. Přibudou nové způsoby sběru logových souborů, primárně pro produkční prostředí hostované bez využití Dockeru, jako je například Apache, Nginx, Redis, MySQL či Elasticsearch.

Časem mohou vznikat požadavky na detailnější informace a jednoduchost rozšiřitelnosti systému bude proces zavedení těchto změn výrazně podporovat.

Práce bude sloužit také jako příručka a technická dokumentace, kde se důležité části přepíší do interní dokumentace a budou dále rozšiřovat.

Závěr

Práce splňuje zadání a všechny požadavky organizace. Návrh, způsob, dopad a výsledek implementace se těšil uznání dobře odvedené práce z řad kolegů a v době publikování práce se již používá na nemalém množství aplikací. Implementace dala organizaci kvalitní základ pro další rozvoj a systém se bude rozšiřovat o přídavnou funkcionalitu a integrovat s dalšími systémy jako je Jira nebo Bitbucket.

Literatura

[1] Docker: About. [online], [cit. 3. 5. 2020]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Docker_(software)&oldid=18172774

[2] Graylog About. [online], [cit 2. 5. 2020]. Dostupné z: https://www.graylog.org/about

[3] Splunkbase. [online], [cit. 4. 5. 2020]. Dostupné z: https://splunkbase.splunk.com/

[4] Elasticsearch: About. [online], [cit. 4. 5. 2020]. Dostupné z: https://www.elastic.co/what-is/elasticsearch

[5] Elastic: Why open source? [online], [cit. 5. 5. 2020]. Dostupné z: https://www.elastic.co/about/why-open-source

[6] Gerhards, R.: The Syslog Protocol. RFC 5424, březen 2009. Dostupné z: https://tools.ietf.org/html/rfc5424

[7] OpenDistro for Elasticsearch: Version history. [online], [cit. 6. 5. 2020]. Dostupné z: https://opendistro.github.io/for-elasticsearch-docs/version-history/

[8] Martinek, J.: Use Case. [online], [cit. 11. 5. 2018]. Dostupné z: http://wiki.knihovna.cz/index.php?title=Use_Case&oldid=28204

[9] RÁCEK, T.: *Vizualizace dat pomocí nástroje Kibana [online]*. Diplomová práce, Masarykova univerzita, Fakulta informatiky, Brno, 2016 [cit. 2020-05-14]. Dostupné z: https://theses.cz/id/dad0ba/

[10] Open Distro for Elasticsearch: Monitoring - available variables. [online], [cit. 15. 5. 2020]. Dostupné z: https://opendistro.github.io/for-elasticsearch-docs/docs/alerting/monitors/#available-variables

Příloha A

Seznam použitých zkratek

DRY Don't Repeat Yourself

ELK Elastic Stack

FTP File Transfer Protocol

GELF Graylog Extended Log Format

HTML Hypertext Markup Language

HTTPS Hypertext Transfer Protocol Secure

HTTP Hypertext Transfer Protocol

JSON JavaScript Object Notation

KQL Kibana Query Language

LDAP Lightweight Directory Access Protocol

PHP PHP: Hypertext Preprocessor

SFTP Secure File Transfer Protocol

SSH Secure Shell

TCP Transmission Control Protocol

UDP User Datagram Protocol

URL Universal Resource Locator

YAML YAML Ain't Markup Language

Příloha B

Obsah přiloženého média

/
- src ..... zdrojové kódy implementace
  - elastic-stack ..... zdrojový kód hlavních částí
  - filebeat ..... zdrojový kód vedlejší části
- text
  - thesis.pdf ..... text práce ve formátu PDF
  - tex ..... zdrojové soubory textu práce