VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SLEDOVÁNÍ APLIKAČNÍCH SLUŽEB A SERVERŮ

APPLICATION SERVICE AND SERVER MONITORING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **FILIP PLEŠKO**
AUTHOR

VEDOUCÍ PRÁCE — **prof. RNDr. MAREK RYCHLÝ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav informačních systémů (UIFS) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Pleško Filip**
Program: Informační technologie
Název: **Sledování aplikačních služeb a serverů**
Application Service and Server Monitoring
Kategorie: Informační systémy

Zadání:

1. Seznamte se s přístupy, technologiemi a nástroji pro zaznamenávání, sledování a analýzu běhu aplikačních služeb a serverů různých úrovní (např. webové a aplikační kontejnery, databáze, middleware, atd.). Dostupná řešení vyhodnoťte a porovnejte.
2. Navrhněte vhodné řešení pro sledování a analýzu běhu vícevrstvé aplikace (např. klient a web-aplikační-databázový server). Zvolte vhodnou architekturu a implementační technologie.
3. Po konzultaci s vedoucím navržené řešení implementujte, včetně návodu a ukázky jeho nasazení.
4. Výsledek popište, vyhodnoťte a zveřejněte jako open-source.

Literatura:

- James Turnbull. The Logstash Book. James Turnbull, 2013. ISBN 0988820218.
- Massimiliano Rak, Salvatore Venticinque, Tams Mhr, Gorka Echevarria, Gorka Esnal. "Cloud Application Monitoring: The mOSAIC Approach," 2011 IEEE Third International Conference on Cloud Computing Technology and Science, Athens, 2011, pp. 758-763. ISBN 978-1-4673-0090-2. [https://doi.org/10.1109/CloudCom.2011.117]

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Rychlý Marek, RNDr., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 16. října 2019

Abstrakt

V bakalárskej práci je riešená problematika monitorovania serverov a serverových aplikácií. Cieľom práce je zjednodušiť a zefektívniť toto monitorovanie. Na vyriešenie problému, bol zvolený prístup pomocou ELK stacku. Jeho použitím sme dosiahli sprehľadnenie log súborov a vizualizáciu dát z nich získaných.

Abstract

This bachelor thesis addresses an issue of monitoring servers and server aplications. The goal is to simplify this monitoring and make it more efficient. To do so, we used ELK stack. With the use of this stack we achieved to easily read from log files and show usefull informations from them.

Kľúčové slová

ELK, elasticsearch, logstash, kibana, filebeat, metricbeat, logovacie súbory, server, webové aplikácie, Yii, PHP

Keywords

ELK, elasticsearch, logstash, kibana, filebeat, metricbeat, log files, server, web aplications, Yii, PHP

Citácia

PLEŠKO, Filip. *Sledování aplikačních služeb a serverů*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Marek Rychlý, Ph.D.

Sledování aplikačních služeb a serverů

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána profesora RNDr. Mareka Rychlého, PH.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Filip Pleško
28. mája 2020

Poďakovanie

Rád by som poďakoval pánovi prof. RNDr. Marekovi Rychlému, Ph.D. za cenné rady počas vedenia mojej bakalárskej práce.

Obsah

1 Úvod **3**

2 Analýza dát **5**
2.1 Čo to je analýza dát? 5
2.1.1 Položenie otázky 5
2.1.2 Získavanie dát 6
2.1.3 Analýza 6
2.1.4 Interpretácia 6
2.2 Sledovanie výkonu aplikácie 6

3 Návrh **9**
3.1 Posielanie dát 9
3.1.1 Filebeat 9
3.1.2 Metricbeat 9
3.2 Spracovanie dát 10
3.3 Ukladanie dát 10
3.4 Zobrazovanie dát 10

4 Použité technológie **11**
4.1 Na strane serveru 11
4.1.1 Beats 11
4.1.2 Logstash 14
4.1.3 Elasticsearch 15
4.1.4 PHP 16
4.1.5 Yii framework 17
4.2 Na strane klienta 17
4.2.1 JavaScript 17

5 Implementácia **19**
5.1 Inštalácia aplikácií 19
5.1.1 Aplikácie pre získavanie dát 19
5.1.2 Filtrovanie dát 20
5.1.3 Databáza 20
5.1.4 Zobrazovanie dát 20
5.2 Konfigurácia aplikácií 20
5.2.1 Filebeat 21
5.2.2 Metricbeat 21
5.2.3 Logstash 22

5.2.4 Elasticsearch . . . 23
5.3 Implementácia aplikácie pre zobrazovanie dát . . . 23
5.3.1 Migrácia . . . 23
5.3.2 Active Record . . . 24
5.3.3 Controller . . . 24
5.3.4 View . . . 25
5.3.5 Cron . . . 27

6 Nasadenie **28**
6.1 Nastavenie serverov . . . 28
6.2 Spustenie aplikácie . . . 28
6.3 Vytvorenie dashboardu . . . 29
6.4 Spustenie Cron plánovača . . . 31

7 Testovanie **32**
7.1 Docker . . . 32
7.2 Aplikácia . . . 32
7.3 Grok modul . . . 34

8 Záver **35**

Literatúra **36**

A Obsah CD **37**

Kapitola 1

Úvod

V dnešnom modernom svete každá, aj malá spoločnosť, generuje veľké množstvo rôznorodých informácií. Tieto informácie sú často veľmi zmätočné a neprehľadné, ich vypovedacia schopnosť pre používateľa je na veľmi nízkej úrovni, preto je potrebné ich usporiadať. Pre rôzne informácie existujú rôzne spôsoby triedenia a prezentácie. Táto bakalárska práca sa zameriava na informácie, respektíve dáta, ktoré generujú jednotlivé serveri a na nich bežiace aplikácie.

Dáta z webových serverov sa zaznamenávajú v textovom formáte v štandardnej štruktúre v takzvaných logovacích súboroch. Pri správnej interpretácií informácií z týchto súborov je možné skoré zachytenie a rozpoznanie problémov s výkonom aplikácií a rýchla detekcia chybových stavov. Na základe získaných informácií a po ich správnej analýze nasleduje úprava aplikácie tak, aby k danému problému a problémom jemu podobným viac nedochádzalo. Týmto spôsobom sa neustále zvyšuje kvalita poskytovaných elektronických služieb.

Vzhľadom na rozsiahlosť týchto súborov, je ich manuálne prezeranie a snaha získania užitočných informácií časovo veľmi náročná. Z tohoto dôvodu vznikajú rôzne prístupy, ako interpretáciu informácií z vygenerovaných dát zefektívniť, sprehľadniť a tým zjednodušiť pochopenie problémov, ku ktorým môže dôjsť.

Prvý problém, ktorý je potreba vyriešiť, je získavanie týchto logovacích súborov. Predpokladajme, že sa snažíme monitorovať viac serverov naraz a teda pre prehľadnosť by bolo dobré, aby sme všetky tieto servery vedeli sledovať z jedného miesta. Aby bolo možné toto zabezpečiť, je potrebné dané súbory na určené miesto zasielať, na čo slúžia takzvané *shippre* 4.1.1.

Ako už bolo uvedené vyššie, tieto súbory sú veľmi rozsiahle a nie každá informácia v nich je pre používateľa podstatná. Preto je vhodné získané dáta vytriediť pomocou filtra a tak čiastočne zmenšiť objem dát, v ktorých budeme neskôr vyhľadávať. V našom prípade budeme pre toto filtrovanie používať *Logstash* 4.1.2.

Zredukované dáta je potrebné uložiť do databázy, z ktorej ich budeme podľa potreby vyberať a zobrazovať. Nakoľko sa jedná o fulltextové vyhľadávanie, čo je veľmi náročný proces, je dôležité dať si záležať na výbere databázy, ktorú použijeme. Databáza by mala obsahovať výborné indexové vyhľadávanie pre zefektívnenie vyhľadávania. Dobrým príkladom je *Elasticsearch* 4.1.3.

Po výbere vhodných nástrojov a ich správnej konfigurácií, by zobrazovanie dát už nemal byť problém. Aj tu máme na výber z niekoľkých možností. Môžeme využiť niektoré zo

zobrazovacích nástrojov, ako je napríklad *Kibana*, ale rovnako môžeme sami prehľadávať databázu a zobrazovať informácie podľa vlastných predstáv.

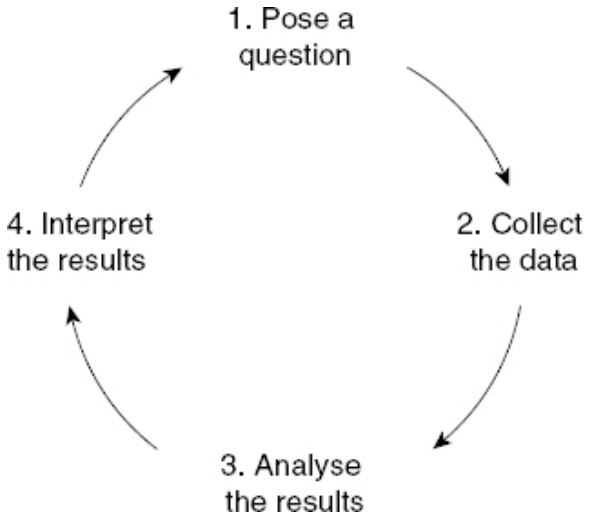
Kapitola 2

Analýza dát

V tejto kapitole si vysvetlíme, čo si predstaviť pod pojmom analýza dát. K čomu sa používa v spojení s monitorovaním serverov a aplikácií, a aké sú jej výhody a prínosy.

2.1 Čo to je analýza dát?

Jedná sa o proces získavania, očistenia, transformovania a modelovania dát, s cieľom objavenia nových informácií o preskúmavaných dátach. Pre lepšiu čitateľnosť údajov sa používa vizuálne zobrazovanie, ktoré zlepšuje možnosť odhalenia užitočných vzorcov. Ná základe takýchto informácií je možné robiť rozhodnutia, ktoré sa snažia vyvarovať predchádzajúcich chýb a tak zlepšiť funkčnosť systému, z ktorého dáta pochádzajú.



Obr. 2.1: Cyklus analýzy dát

2.1.1 Položenie otázky

Na začiatku je potrebné si premyslieť, na aký účel chceme robiť dátovú analýzu. Je potrebné si vybrať, ktorý druh dátovej analýzy chceme použiť. Aby sme sa správne rozhodli, musíme si určiť, čo chceme analyzovať a ako to odmerať, uvedomiť si, prečo to skúmame a aké ukazovatele použiť, aby sme získali očakávané výsledky.

2.1.2 Získavanie dát

Po zodpovedaní otázky máme presnejšiu predstavu o tom, čo hľadáme. Teraz môžeme začať so zbieraním dát ná základe požiadaviek, ktoré by mali byť postavené tak, aby získané dáta boli odpoveďou na zadanú otázku. Dáta je ďalej potrebné spracovať a usporiadať pre analýzu. Ak sú dáta z viacerých zdrojov, je potrebné uchovávať čas a miesto, z ktorého dáta pochádzajú. Po zozbieraní dát existuje stále možnosť, že nie všetky získané informácie sú potrebné pre našu analýzu, informácie môžu obsahovať chyby, duplicity... Dáta, ktoré vstupujú do analýzy musia byť bezchybné, lebo od toho závisí dôveryhodnosť výstupu celkovej analýzy.

2.1.3 Analýza

Po úspešnom získaní a očistení dát, sú dáta pripravené pre dátovú analýzu. V tejto fáze môžeme zistiť, že sme získali odpoveď, ktorú sme hľadali alebo je ešte potrebné získať dodatočné dáta. Pre lepšiu prácu so získanými informáciami môžeme použiť rôzne nástroje a softvéry pre analýzu, ktoré nám pomôžu pochopiť a zobraziť hľadané informácie.

2.1.4 Interpretácia

Po ukončení analýzy je potrebné, aby získané znalosti z nej boli transformované do podoby, z ktorej je vidieť, čo sa danou analýzou dosiahlo. Výsledky môžeme popísať slovne alebo pomocou použitia grafov a tabuliek. Z prehľadne usporiadaných výsledkov je jednoduchšie rozhodnúť sa, akým smerom ďalej napredovať.

2.2 Sledovanie výkonu aplikácie

Sledovanie výkonu aplikácie (APM - aplication performance monitoring) je jeden z procesov manažmentu výkonu aplikácie (application performance management). Zaoberá sa skúmaním, či výkon aplikácie zodpovedá očakávaným výsledkom. Umožňuje správcom odhaľovať a opravovať chyby a potencionálne problémy ešte pred tým, než zasiahnu koncového používateľa. K problémom pri používaní aplikácií môže dôjsť na rôznych úrovniach a preto je potrebné sledovať všetky úrovne.

- **Úroveň serveru** – Na serverovej úrovni pomocou APM sledujeme zaťaženie procesoru, využitie pamäte, rýchlosť zápisov a čítaní z disku aby sa zaistilo, že využitie hardvéru nespomaľuje výkon aplikácie.
- **Úroveň softvéru** – Na softvérovej úrovni sledujeme ako často dochádza k spomaleniu alebo zlyhaniu výkonu aplikácie. Napríklad môže dôjsť k zlyhaniu, ak požiadavky zo strany aplikácie končia chybou alebo k spomaleniu v dôsledku pamäťovo náročného procesu ako je prehľadávanie v databáze.

Pre získanie informácií sa využívajú metriky ako napríklad:

- **Čas odozvy** – miera ktorá určuje čas, ako dlho trvalo aplikácií zobraziť požadované dáta od zaslania požiadavku.
- **Miera požiadavkov** – znázorňuje, aké množstvo používateľov používa aplikáciu v rôznych časoch naraz. Kedy je špička a kedy nečinnosť

- **Dostupnosť aplikácie** – metrika ktorá znázorňuje kedy je aplikácia online a dostupná v pomere kedy je nedostupná. Používa sa pre dodržiavanie SLA.

Trh s APM je plný rôznych možností, pri čom každé z nich má vlastnú sadu nástrojov určených na monitorovanie aplikácií. Pri výbere toho správneho, je potrebné sa zamerať na to, ako a aké dáta zobrazuje a či toto zobrazenie vyhovuje našim potrebám. Ďalej je dobré vedieť, aké skúsenosti s daným nástrojom majú ostatný koncový používatelia, aká je náročnosť na jeho použitie. Nie je zanedbateľná ani jeho cena, ktorá je dôležitý údaj pre analýzu ekonomickej efektívnosti a návratnosti vstupnej investície. Pri rozhodovaní, ktorý z dostupných nástrojov bude použitý počas vypracovania práce sme sa rozhodovali medzi niekoľkými. Nasledujúca tabuľka porovnáva, ktoré softvéry ponúkajú potrebné funkcionality.

Softvér	centrálne monitorovanie	parsovanie a filtrovanie	Indexovanie	API
Loggly	✓	✓	✓	X
Graylog	✓	✓	✓	✓
ELK stack	✓	✓	✓	✓
Splunk	✓	✓	✓	✓

- **Centrálne monitorovanie -** umožňuje monitorovanie viacerých vzdialených serverov
- **Parsovanie a filtrovanie -** použité pre rozpoznanie získaných udalostí, poprípade odstránenie nežiaducich
- **Indexovanie -** umožňuje rýchle prehľadávanie veľkého množstva získaných dát
- **API -** použité pre prístup k informáciám z externej aplikácie

Ďalším faktorom, ktorý rozhoduje pri vybratí aplikácie je rozsiahlosť a kvalita dokumentácie a spokojnosť používateľov.

Softvér	Dokumentácia	Hodnotenie používateľov
Loggly	stredná	8,8/10
Graylog	veľmi dobrá	9/10
ELK stack	veľmi dobrá	8/10
Splunk	veľmi dobrá	8,5/10

Hodnotenia aplikácií Graylog, Splunk a ELK stack boli získané z online aplikácie pre porovnávanie softvérov [11]. Z dôvodu, že Loggly v danej aplikácii nemá žiadne hodnotenia, bol použitý zdroj z oficiálnych stránok softvéru [12].

Pri hodnotení jednotlivých dokumentácií nás zaujímalo v akom časovom úseku sme schopný nájsť požadovanú informáciu. Pri softvéroch Graylog, ELK stack a Splunk bola hľadaná informácia nájdená pomerne jednoducho a intuitívne preto boli ohodnotené na veľmi dobre zdokumentované. Na druhú stranu v dokumentácii pre Loggly bol problém sa zo začiatku zorientovať a nájsť potrebné informácie trvalo značne dlhšiu dobu a preto bola ohodnotená za stredne dobre zdokumentovanú.

Zároveň je potrebné brať ohľad aj na cenu, koľko daný softvér stojí. Nie všetky sú open-source a niektoré, menšie firmy, by mohli mať problém s platením drahých licencií. Cena sa môže líšiť v závislosti od rozsahu licencie.

Softvér	cena za jednotku	jednotka
Loggly	79-279$	mesiac
Graylog	open-source/4500-20000$	rok
ELK stack	open-source	-
Splunk	1800$	rok

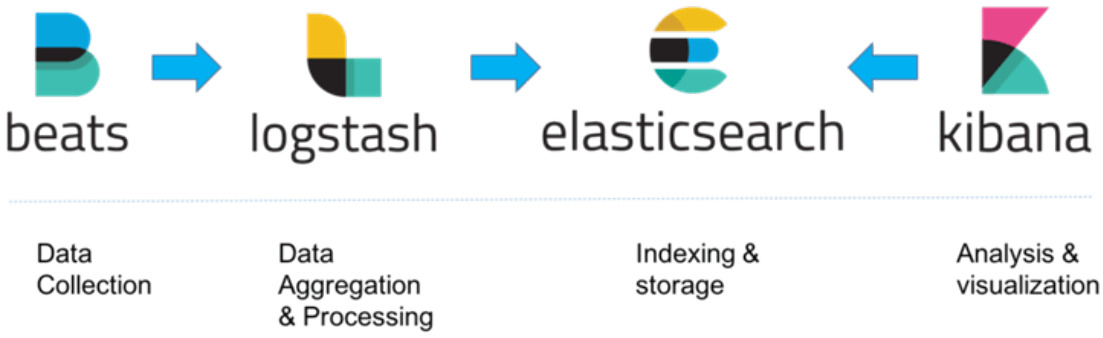
Jednotlivé ceny softvérov boli získané z online zdrojov. Splunk [7], Loggly [10] a ELK stack [3] ponúkajú prehľad ceny za produkt priamo na oficiálnych stránkach. Cena za Graylog sa na oficiálnych stránkach nenachádza a z tohoto dôvodu ju bolo potrebné nájsť na užívateľských fórach, kde sa užívateľ podelil s cenovou kalkuláciou, ktorú mu firma ponúkla [8].

Na základe vyššie uvedených tabuliek bolo rozhodnuté, že bude v práci využitý ELK stack. Všetky potrebné funkcionality, pre dokončenie práce, sú v ELK stack dostupné v open-source verzii. Loggly v open-source verzii neponúka prístup k API a jeho dokumentácia nie je na takej úrovni ako pri ostatných možnostiach. Graylog by bola možná náhrada za ELK stack, ale z dôvodu že Graylog používa taktiež ako ELK, Elasticsearch databázu, tak by sme sa tu pri vývoji stretli s dvojitým štýlom dokumentovania, na rozdiel od použitia ELK, kde sú všetky potrebné aplikácie dokumentované rovnakým štýlom. Splunk je dlhodobo jednotkou na trhu, čo sa týka monitorovania serverov, čo ale taktiež prináša skutočnosť, že jeho cena odráža kvalitu a z dôvodu, že neponúka voľne dostupnú verziu nebolo možné tento softvér použiť.

Kapitola 3

Návrh

Táto kapitola popisuje, ako pomocou ELK stacku získavať dáta zo serverov, ako prebieha ich posielanie, spracovanie a ukladanie a následné grafické zobrazovanie.



Obr. 3.1: ELK stack

3.1 Posielanie dát

Pre získavanie dát z viacerých zdrojov sa používajú shippre, ktoré sú nainštalované na zdroji a pravidelne zasielajú nové údaje na spracovanie. ELK v tomto prípade používa rodinu programov nazvaných beats. Jedná sa o skupinu takzvaných lightweight [1] programov. My budeme používať práve dva z tejto rodiny a to sú *Filebeat* a *Metricbeat*.

3.1.1 Filebeat

Agent nainštalovaný na všetkých serveroch, ktoré chceme monitorovať. Stará sa o sledovanie logovacích súborov alebo lokácií, ktoré mu špecifikujeme, zbiera logovacie udalosti a zasiela ich ďalej do Logstashu.

3.1.2 Metricbeat

Rovnako ako Filebeat, je to agent nainštalovaný na našich serveroch. Ale na rozdiel od Filebeatu nezbiera informácie z logovacích súborov, ale zameriava sa na získavanie metrík

[1] program, navrhnutý tak, aby pri prevádzke využíval minimum zo systémových zdrojov ako je CPU, RAM

z operačného systému a zo služieb bežiacich na serveri. Aj tento agent zasiela získane informácie do Logstashu.

3.2 Spracovanie dát

Dáta v ELK stacku sú spracované pomocou Logstashu. Je to nástroj, ktorý dokáže zhromažďovať dáta z viacerých zdrojov, zjednocovať ich, filtrovať a posielať ďalej na definovanú destináciu.

3.3 Ukladanie dát

Pri ukladaní dát si treba uvedomiť, že ukladáme veľké množstvo textových log správ. Snažiť sa ukladať a potom vyhľadávať v takýchto dátach by bolo s obyčajnou databázou veľmi náročné a zdĺhavé. ELK v tomto prípade používa Elasticsearch, čo je databáza, ktorá umožňuje prehľadávanie dát všetkých typov. Dáta ukladá a indexuje a vďaka tomu je vyhľadávanie veľmi efektívne.

3.4 Zobrazovanie dát

Keď už máme všetky dáta zozbierané, vytriedené a uložené, je na čase začať dáta zobrazovať. K tomuto nám poslúži Kibana. Kibana je platforma určená pre vizualizáciu dát. Ponúka širokú škálu grafov, máp a tabuliek, pomocou ktorých môžeme jednoducho zobrazovať požadované štatistiky. Veľkou výhodou programu Kibana je aj ponúkaná možnosť zdieľať grafy. Zdieľanie grafov nám umožňuje zobrazovať grafy v nej vytvorené na iných webových stránkach.

Kapitola 4

Použité technológie

V tejto kapitole si popíšeme, aké technológie boli použité pri tvorbe aplikácie.

4.1 Na strane serveru

Okrem vytvorenia aplikácie, ktorá bude umiestnená na serveri a zobrazovať získané dáta, bolo taktiež potrebné na serveroch nainštalovať a spustiť aplikácie, ktoré nám umožnia získavať dáta z týchto serverov. Z dôvodu finančného obmedzenia sme sa zamerali na voľne dostupné (open-source) aplikácie.

4.1.1 Beats

Beats sú voľne dostupné lightweight aplikácie, ktoré je možné nainštalovať ako agentov na servery, z ktorých budú zasielať dáta. V rodine Beats aplikácií nájdeme aplikácie pre monitorovanie rôznych dát zo serveru. V našej práci sa zameriame hlavne na monitorovanie pomocou Filebeat a Metricbeat.

Filebeat [4] sa skladá z dvoch hlavných komponentov, ktorými sú vstupy a zberače. Tieto dva komponenty spolupracujú, zbierajú informácie z koncov súborov a odosielajú ich na výstup, ktorý im bol špecifikovaný.

Zberač je zodpovedný za čítanie obsahu jednotlivých súborov. Číta každý súbor po jednotlivých riadkoch a získané informácie odosiela na výstup. Pre každý súbor je vytvorený jeden zberač. Tento zberač je taktiež zodpovedný za otváranie a zatváranie súboru, čo znamená že deskriptor súboru ostane otvorený pokiaľ zberač pracuje. Ak je súbor presunutý alebo premenovaný, pokiaľ na ňom zberač pracuje, tak Filebeat naďalej pokračuje v jeho čítaní. Toto má vedľajší efekt, že je na disku rezervované miesto, až pokiaľ zberač neukončí svoju prácu. Filebeat drží súbor otvorený, až pokým nedôjde k dovŕšeniu časového limitu, počas ktorého súbor nebol zmenený.

Ukončenie zberača má nasledovné dôsledky:

- Zatvorenie obsluhy súboru a uvoľnenie základných zdrojov, ak bol súbor zmazaný počas jeho obsluhy.
- Obnovenie kontroly súboru začne až po dovŕšení hodnoty po ktorej Filebeat znovu skontroluje dostupnosť nových súborov pre zbieranie dát.

- Ak bol súbor presunutý alebo odstránený, jeho kontrola ďalej nebude obnovená.

Vstup je zodpovedný za správu zberačov a nájdenie všetkých zdrojov, z ktorých bude čítať. Ak typ vstupu je `log`, tak vstup nájde všetky súbory na disku, ktoré sa zhodujú s definovanou cestou a naštartuje zberač pre každý z nich. Filebeat aktuálne podporuje niekoľko vstupných typov a každý z nich môže byť definovaný viac krát. `Log` vstup skontroluje každý súbor, aby zistil, či je potrebné naštartovať nový zberač alebo či už nejaký nebeží alebo či súbor môže byť ignorovaný. Ak sa veľkosť súboru zväčšila od zatvorenia zberača, tak len nové riadky budú zozbierané.

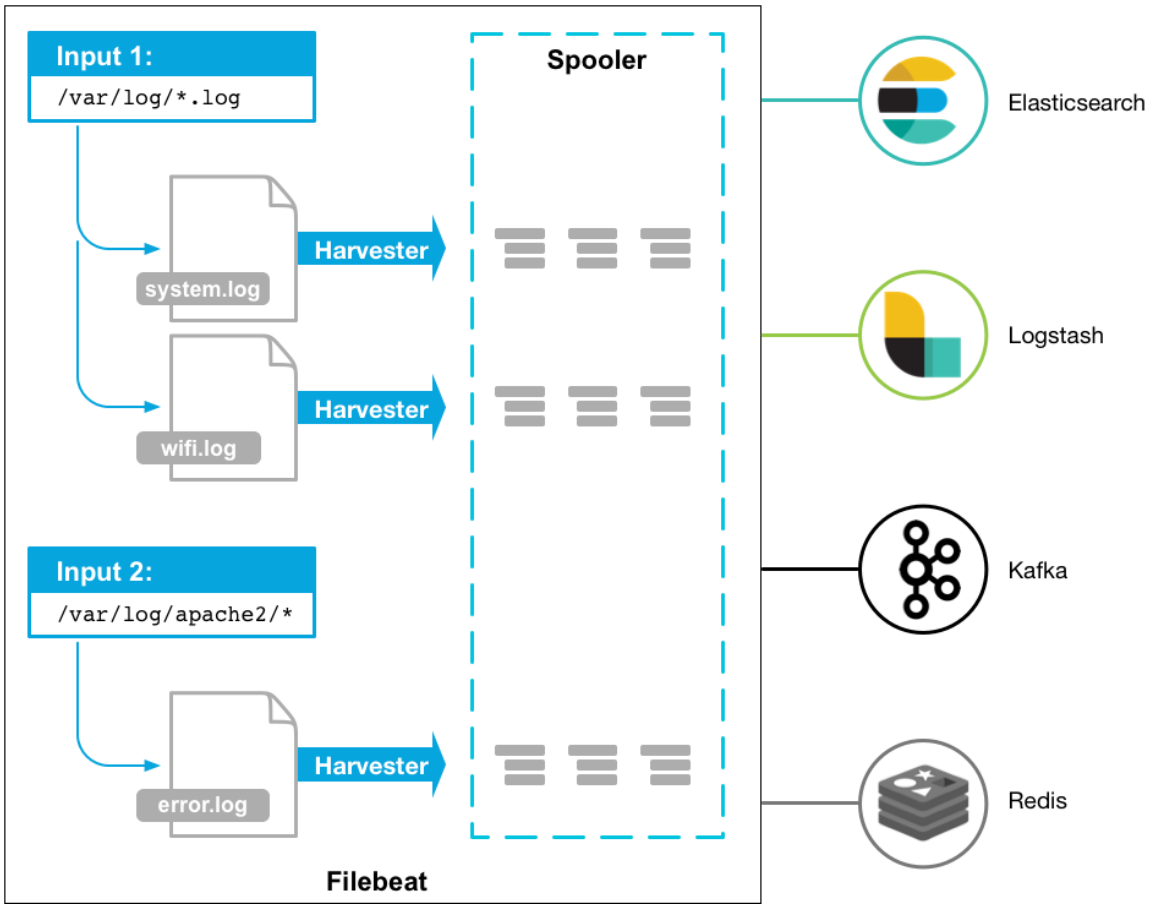
Filebeat udržuje stav každého súboru a často obnovuje stav na disku v registri súbore. Stav sa používa na zapamätanie posledného ofsetu odkiaľ zberač naposledy čítal a na zabezpečenie, že všetky riadky zo súboru boli odoslané. Ak výstup nie je dostupný, Filebeat si zapamätá, ktorý riadok bol odoslaný ako posledný a znovu začne odosielať, akonáhle je výstup znovu dostupný. Pokiaľ je Filebeat spustený, informácie o stave pre každý súbor sú taktiež uchované v pamäti. Keď sa reštartuje Filebeat, tak sa využijú dáta z register súboru na znovu vytvorenie stavu a Filebeat naštartuje zberač na poslednej známej pozícii.

Pre každý vstup, je uložený stav každého súboru. Z dôvodu, že súbory môžu byť premenované alebo presunuté, názov súboru a cesta k nemu nie je dostačujúci identifikátor. Preto Filebeat vytvára a ukladá pre každý súbor špeciálny identifikátor, ktorý umožňuje detekciu, či bol súbor predtým čítaný.

Filebeat zaručuje doručenie každej udalosti na definovaný výstup minimálne raz a bez akejkoľvek straty dát. Toto je možné v ďaka tomu, že si pre každú udalosť ukladá stav doručenia v register súbore.

V situácii, keď je výstup blokovaný, Filebeat sa bude pokúšať posielať dáta na výstup až pokiaľ nedostane odpoveď, že dáta boli úspešne doručené.

Ak je Filebeat vypnutý počas zasielania, tak nečaká na odpoveď od výstupu o úspešnom doručení všetkých dát. Všetky záznamy, ktoré boli odoslané pred vypnutím, ale neboli potvrdené výstupom o prijatí budú po reštarte Filebeatu odoslané znovu. Toto zaručuje, že každý záznam je odoslaný aspoň raz, ale môže to spôsobiť, že na výstupe sa objavia duplicity. Aby sme sa tomuto aspoň čiastočne vyhli, je možné nastaviť timeout, počas ktorého po vypnutí bude Filebeat ešte čakať na odpoveď o doručení.



Obr. 4.1: Filebeat

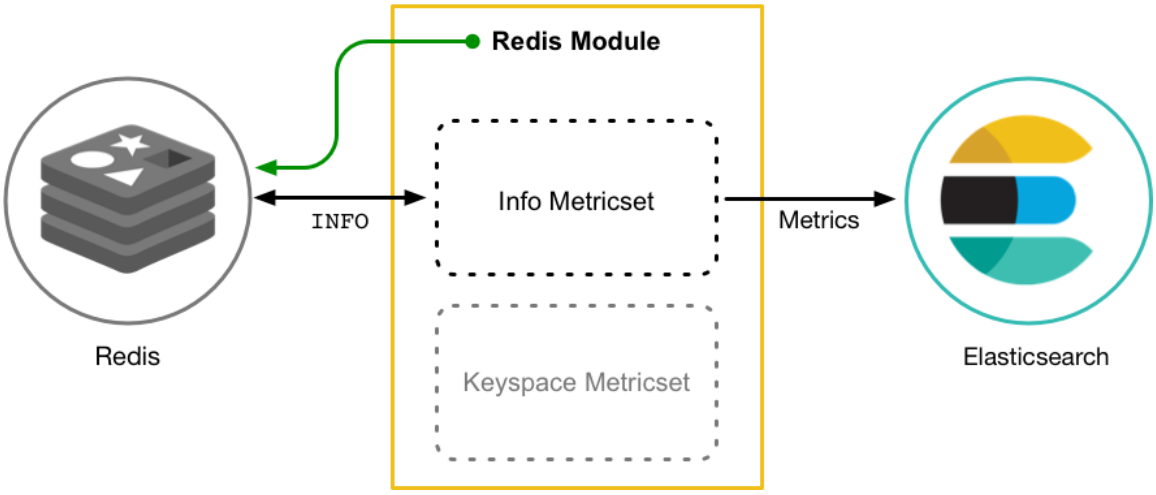
Metricbeat [6] sa skladá z modulov a metrík. Metricbeat modul definuje základnú logiku pre zbieranie dát z konkrétnej služby, ako napríklad Redis, MySQL, a ďalších. Modul špecifikuje detaily o službe vrátane informácií ako sa pripojiť, ako často a ktoré metriky zbierať.

Každý modul má minimálne jeden set metrík. Metriky sú súčasťou modulu, ktorý zbiera dáta a dáva im štruktúru. Namiesto toho aby zbieral každú metriku ako oddelenú udalosť, zozbiera zoznam viacerých metrík, ktoré so sebou súvisia, v jednej požiadavke a spolu ich odošle. Rôzne metriky, ktoré spolu súvisia, sú pospájané do setov a odosielané v jednej požiadavke, čo nám uľahčuje prácu. Väčšina modulov má základné skupiny metrík, ktoré sú povolené, ak si používateľ nenastavil vlastné.

Metricbeat pravidelne získava metriky tak, že vypočúva host systém na základe periódy, ktorá bola špecifikovaná v konfigurácii modulu. Z dôvodu, že môže byť odoslaných viacero požiadaviek na metriky v jednom čase na tú istú službu, Metricbeat sa snaží znovu použiť to isté pripojenie vždy keď môže. Ak sa nemôže pripojiť na službu počas doby špecifikovanej v timeout, tak vráti oznámenie o chybe (error). Metricbeat posiela udalosti asynchrónne, čo znamená, že ich obdržanie nie je potvrdené. Ak teda nastavený výstup nie je dostupný, dáta budú stratené.

Keď Metricbeat narazí na error (napríklad ak sa nemôže pripojiť k hostu), tak odošle error o udalosti na definovaný výstup. Z toho vyplýva, že udalosť je vždy odoslaná, aj

napriek tomu, že sa vyskytol problém. Toto nám umožňuje monitorovať chyby a debugovať správy, pomocou čoho ľahšie identifikujeme chybu.



Obr. 4.2: Metricbeat module

4.1.2 Logstash

Logstash [5] je program určený na zber dát, ktoré zároveň aj spracováva. Dokáže dynamicky zjednocovať dáta z odlišných zdrojov a smerovať ich do destinácie, ktorá bola nakonfigurovaná. Pomocou Logstashu môžeme očisťovať všetky dáta a pripravovať ich na pokročilé analytické a zobrazovacie prípady použitia. Pôvodne bol určený pre zbieranie log záznamov no postupom času sa prepracoval na mnoho ďalších možností použitia. Každý typ udalosti môže byť premenený pomocou širokej škály vstupných, výstupných a filtrovacích pluginov.

Proces spracovanie udalostí v Logstashi má tri fázy. Vstup -> Filter -> Výstup. Vstup generuje udalosti, filter ich upravuje a výstup ich podľa potreby posiela ďalej. Vstupy a výstupy podporujú kodeky, ktoré nám umožňujú dekódovať a zakódovať dáta pri vstupe alebo výstupe zo spracovania, bez toho aby bolo potrebné použiť filter.

Vstup používame pre zozbieranie dát do procesu ich spracovania. Najčastejšie používané vstupu sú:

- **file:** číta zo súboru uloženom na disku. Podobne ako UNIX príkaz `tail -0f`.
- **syslog:** počúva na známom porte 514 pre syslog správy a analyzuje ich podľa RFC3164 formátu.
- **redis:** číta z redis serveru, s použitím redis kanálov a zoznamu. Redis je taktiež používaný ako "broker"v centralizovanej Logstash inštalácii, kde ukladá udalosti od vzdialených odosielateľov do poradia.
- **beats:** spracováva udalosti odoslané pomocou beats aplikácií.

Filtre sú zariadenia v procese spracovanie, ktoré toto spracovanie vykonávajú. Je možné kombinovať filtre spolu s podmienkami, a tak uplatniť špeciálne filtre na udalosti, ktoré splnili dané podmienky. Užitočné filtre sú napríklad:

- **grok:** analyzuje ľubovolný text a pridáva mu štruktúru. Grok je aktuálne najlepšia možnosť, ako v Logstash parsovať neštruktúrované log dáta, a tak ich premeniť na štruktúrované informácie, ktoré je možné uchovávať, prehľadávať a filtrovať. Logstash obsahuje 120 vzorov, pomocou ktorých je možné rozpoznávať prichádzajúce dáta.
- **mutate:** vykonáva všeobecnú premenu nad poliami udalostí. Umožňuje premenovať, odstrániť, upraviť a nahradiť jednotlivé polia.
- **drop:** celkovo zahodí neželanú udalosť.
- **clone:** vytvorí kópiu udalosti a môže jej pridať alebo odstrániť polia.
- **geoip:** pridá geografické informácie týkajúce sa lokácie IP adries.

Výstup je finálnou časťou spracovania udalostí v Logstash. Udalosť môže byť odoslaná na viacero výstupov, ale v momente, keď sú všetky výstupné procesy splnené, spracovanie udalosti je ukončené. Príklady výstupov sú:

- **elasticsearch:** posiela udalosti do elasticsearch. Ak máme záujem ukladať dáta v efektívnom, praktickom a ľahko vyhľadávacom formáte, tak toto je ten správny výstup.
- **file:** zapisuje dáta do súboru na disku.
- **graphite:** posiela dáta do graphite, známeho, voľne dostupného nástroja pre ukladanie a zobrazovanie metrík.
- **statsd:** odosiela dáta do statsd. Zbiera štatistiky ako napríklad počítadla a časovače odosielané cez UDP a odosiela agregované dáta ďalším backend doplnkovým službám.

Kodeky sú v podstate filtre, ktoré môžu fungovať ako súčasť vstupov alebo výstupov. Umožňujú nám jednoduché oddelenie správ od serializácie. Medzi známe kodeky patria `json`, `msgpack` and `plain` (text).

- **json:** zakóduje alebo dekóduje dáta do JSON formátu.
- **multiline:** spojí viacriadkový text ako napríklad stacktrace správy do jednej udalosti.

4.1.3 Elasticsearch

Elasticsearch [2] je distribuovaný softvér pre vyhľadávanie a analýzu. Ponúka real-time vyhľadávanie a analýzu pre všetky tipy dát. Ak máme štrukturovaný alebo neštrukturovaný text, číselné hodnoty alebo geopriestorové dáta, Elasticsearch umožňuje ich efektívne ukladanie a indexovanie tak, aby bolo zabezpečené rýchle vyhľadávanie. Môžeme s ním vykonávať zložité vyhľadávanie a agregáciu dát. Aj keď nie každý problém, je problém vyhľadávania, Elasticsearch ponúka rýchlosť a flexibilitu pre zvládnutie širokej škály prípadov použitia:

- Pridáva box pre vyhľadávanie do aplikácie alebo web stránky.
- Ukladá a analyzuje logy, metriky a bezpečnostné dáta.

- Používa strojové učenie pre automatické modelovanie chovania sa dát v skutočnom čase.
- Automatizuje obchodné, pracovné toky s použitím Elasticsearch ako softvér pre ukladanie.
- Správa, integrácia a analýza priestorových informácii s použitím Elasticksearchu ako geografického informačného systému GIS.

4.1.4 PHP

PHP [13] je skriptovací jazyk na strane servera. Podobne ako ďalšie skriptovacie jazyky, umožňuje vývojárovi vložiť logiku do vytvárania webového obsahu a spravovať dáta vrátené z webového prehliadača. PHP taktiež obsahuje množstvo rozšírení, ktoré uľahčujú prácu s databázou, získavanie dát ktoré chceme zobraziť na web stránke alebo naopak ukladanie dát, ktoré vytvorí používateľ.

PHP sa skladá zo skriptovacieho jazyka a interpreta. PHP definuje správanie a logiku webovej stránky. Tieto skripty sú vložené do HTML dokumentov a sú spracované webovým serverom. Interpret je vo forme modulu, ktorý je integrovaný na serveri a konvertuje skripty na príkazy, ktoré počítač potom vykoná tak, aby dosiahol výsledky definované vývojárom.

Pre správne pochopenie ako PHP funguje, je potrebné najskôr pochopiť čo sa stane, keď sa zobrazuje web stránka v prehliadači. Keď používateľ navštívi web stránku alebo klikne na odkaz na stránke, prehliadač odošle požiadavku na webový server ktorý poskytuje danú stránku, aby mu poslal kópiu stránky. Web server obdrží požiadavku, vo svojom systéme súborov nájde požadovaný súbor s obsahom vyžiadanej web stránky a odošle ho naspäť do používateľovho prehliadača.

Zvyčajne sa webový server nezaujíma o obsah súboru, ale iba ho odošle webovému prehliadaču, pokiaľ si je istý, že sa obsah súboru zobrazí používateľovi v prehliadači tak, ako to zamýšľal vývojár.

Tu je potrebné špecifikovať, akému typu obsahu rozumie prehliadač. V dnešnej dobe je normálne, že webové stránky pozostávajú z HTML, XHTML a JavaScriptu. Ak sa takýto obsah dostane do prehliadača, tak prehliadač vie ako s daným obsahom narábať a zobrazí ho tak, ako bolo plánované. Rozumie štruktúre HTML a zvládne vyrenderovať stránku a obsahuje JavaScript prekladač, pomocou ktorého rozumie inštrukciám z JavaScript skriptu. Avšak vzhľadom na to, že prehliadač vôbec nerozumie PHP skriptu, ktorý môže byť vložený do HTML dokumentu, nevedel by čo s daným dokumentom urobiť, ak by ho dostal ako odpoveď od serveru.

Keďže prehliadač nevie ako narábať s HTML súborom, ktorý obsahuje PHP skript, musíme takýto súbor upraviť ešte na strane servera. Tu prichádza na radu PHP modul na predbežné spracovanie. Ako už bolo spomenuté, PHP modul je integrovaný na servery. Tento modul hovorí serveru, že ak sa snaží obslúžiť dokument, ktorý obsahuje PHP skript (identifikovaný podľa špeciálnej značky), tak má tento skript najprv poslať do modulu pre predbežné spracovanie a počkať pokiaľ modul nevráti obsah, ktorý je náhrada daného skriptu. PHP modul rozumie PHP skriptu, vyhodnotí ho a na základe inštrukcii v ňom napísaných vráti obsah, ktorému bude prehliadač rozumieť. PHP server nahradí získaný výstup na miesta, kde predtým bol skript a upravenú web stránku odošle do prehliadača, ktorý teraz už bez problémov zobrazí obsah.

Vďaka tejto mechanike vieme vytvárať šablóny stránok ako napríklad profil. Každý profil má rovnakú štruktúru, ale obsahuje iné údaje. Pomocou HTML by sme museli údaje vložiť

staticky, a vytvoriť stránku pre každý jeden profil. S využitím šablóny a PHP môžeme na miesta, kde sa budú dáta meniť v závislosti od toho ktorý profil prezeráme, vložiť premenné a teda budeme mať jeden súbor pre rôzne profily.

4.1.5 Yii framework

Yii [9] je vysoko výkonný framework založený na komponentoch, určený pre rýchly vývoj moderných webových aplikácií. Meno Yii s výslovnosťou [ji:] pochádza z Čínštiny a v preklade znamená jednoduchý a evolučný.

Yii je typický framework pre programovanie webov, čo znamená, že môže byť použitý pre vývoj rôznych druhov web aplikácií, ktoré používajú PHP. Vďaka architektúre založenej na komponentoch a sofistikovanom ukladaní do vyrovnávacej pamäte, je obzvlášť vhodný pre vývoj rozsiahlych aplikácií ako sú napríklad portály, fóra, systémy pre správu obsahu (CMS), online obchody, ...

Podobne ako ostatné frameworky, Yii je založené na MVC (Model-View-Controller) architektúre a propaguje organizáciu kódu založenú na tomto modele. Vychádza z filozofie, že kód by mal byť písaný jednoduchou a zároveň elegantnou formou a z tohoto dôvodu sa nikdy nebude snažiť pretvárať veci len preto, aby sa striktne držalo nejakého vzoru. Yii má taktiež v sebe zabudovaných mnoho overených a užitočných funkcií, ktoré sú rovno pripravené na použitie, ako napríklad ActiveRecord pre oboje relačné aj noSQL databázy, podporu pre viac úrovňové ukladanie do vyrovnávacej pamäte, podporu pre RESTfull API vývoj a mnoho ďalších. Výhodou tohoto frameworku je, že je ho možné veľmi jednoducho rozširovať. Je možné nahradiť a upraviť takmer každý kúsok kódu z jadra frameworku. Jeho súčasťou je aj solídna architektúra pre rozšírenia, vďaka ktorej je možné vytvárať nové rozšírenia.

Vývojový tím si drží prehľad o najnovších trendoch, osvedčených postupoch a funkcionalitách čo sa týka webového vývoja a snaží sa ich implementovať do jadra Yii frameworku a predstaviť cez jednoduché a elegantné rozhrania.

Pre správny chod Yii frameworku je potrebná verzia 5.4.0 PHP a najlepšie výsledky ma s verziou 7. Yii je vytvorený čisto pomocou objektovo orientovaného programovania.

4.2 Na strane klienta

4.2.1 JavaScript

JavaScript [1] je skriptovací jazyk, ktorý nám umožňuje implementovať komplexné funkcie na webových stránkach. Vždy, keď web stránka robí niečo viac, ako len statické zobrazenie informácii, zobrazuje obsah, ktorý sa v čase aktualizuje, interaktívne mapy, animované 2D/3D grafiky... Môžeme si byť istý, že je to spôsobené použitím JavaScriptu. Jedná sa o tretiu vrstvu z vrstiev štandardných webových technológií. Prvé dve sú HTML a CSS.

Jadro jazyku JavaScript sa skladá z niektorých základných programovacích funkcionalít, ktoré nám umožňujú robiť veci ako ukladanie hodnôt do premenných, operácie s textom (v programovacej terminológii string), spúšťať určitú časť kódu v závislosti na činnosti používateľa, napríklad po kliknutí na určitú časť webovej stránky.

JavaScript bol vytvorený v roku 1995 pre prehliadač Netscape Navigator. Umožňoval na tomto prehliadači pridávať programy do webových stránok. Postupom času sa však vyvinul a v dnešnej dobe je podporovaný všetkými modernými prehliadačmi. Pomocou JavaScriptu sú dnes vytvárané všetky moderné webové aplikácie a vďaka nemu môže používateľ in-

teraktívne pracovať s danou aplikáciou bez toho aby zakaždým musel čakať kým sa mu stránka znovu načíta. Toto viditeľne zrýchľuje prácu používateľa. Vďaka veľkej rozšírenosti JavaScriptu postupne začali vznikať knižnice, ktoré ho využívajú a vďaka týmto knižniciam uľahčujú vývojárom vývoj nových, moderných aplikácii. Príkladom takejto knižnice je **ReactJS**. Jedná sa o knižnicu, ktorá je určená na stavanie znovu-použiteľných komponentov, ktoré obsahujú dáta čo sa menia s časom. Veľká časť vývojárov využíva React ako V vo MVC (model view controller) aplikáciach. Renderovať React je možné nie len na strane klienta ale aj na strane serveru a to pomocou Node.

Kapitola 5

Implementácia

Táto kapitola je zameraná na popis, ako správne nastaviť aplikácie použité pre zhromažďovanie a ukladanie dát. Taktiež popisuje akými spôsobmi je možné tieto dáta prehľadávať a zobrazovať.

5.1 Inštalácia aplikácií

Ako prvý krok je potrebné nainštalovať vhodné aplikácie na server, poprípade servery, pomocou ktorých budú dané servery sledované.

5.1.1 Aplikácie pre získavanie dát

Pre získavanie dát bola využitá rodina beat aplikácii, konkrétne filebeat a metricbeat, ktoré umožňujú jednoduchý spôsob získavania podstatných dát a veľmi dobre fungujú v spolupráci s ELK stack. Pri použití serverov založených na linuxovom operačnom systéme, môžeme pre inštaláciu využiť *Advanced Packaging Tool* v skratke APT, vďaka ktorému môžeme postupovať následovne:

1. Najskôr je potrebné stiahnuť a nainštalovať verejný kľúč a to pomocou príkazu:

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

2. Potom je potrebné nainštalovať *apt-transport-https* príkazom:

   ```
   apt-get install apt-transport-https
   ```

3. Ďalej uložíme definíciu repozitára do */etc/apt/sources.list.d/elastic-7.x.list*:

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

4. Na koniec aktualizujeme APT a môžeme nainštalovať ľubovoľnú beat aplikáciu:

   ```
   apt-get update
   apt-get install filebeat
   apt-get install metricbeat
   ```

5.1.2 Filtrovanie dát

Pre filtrovanie dát a bližšiu špecifikáciu, ako chceme mať dáta uložené v databáze, bol použitý Logstash. Umožňuje zbierať dáta, ktoré sú mu posielané na vstup, spracovať ich a ďalej posielať na ukladanie do databáze. Pre jeho inštaláciu je možné taktiež využiť APT repozitár, ktorého vytvorenie bolo popísané v sekcii 5.1.1 a teda na samotnú inštaláciu stačí jediný príkaz:

```
apt-get install logstash
```

A však pre jeho správnu funkčnosť je potrebné aby v systéme, na ktorom chceme mať Logstash nainštalovaný, bola nainštalovaná *Java 8* alebo *Java 11*. Skontrolovať aktuálnu verziu Java vieme pomocou príkazu:

```
java -version
```

V prípade, že monitorujeme viac serverov naraz, nie je potrebné mať na každom serveri nainštalovaný Logstash osobitne. Stačí jeden, na ktorý budú prichádzať dáta z viacerých zdrojov a už Logstash samotný tieto dáta spracuje a môže im pridať značku, ktorá bude uložená do databázy spolu s ostatnými dátami a podľa ktorej budeme ďalej rozlišovať, z ktorého serveru jednotlivé informácie pochádzajú.

5.1.3 Databáza

Je potrebné uvedomiť si, že budeme ukladať veľké množstvo dát, v ktorých bude potrebné čo najefektívnejšie vyhľadávať. Preto sme si zvolili Elasticsearch databázu a taktiež preto, že vďaka predošlým inštaláciám je jej inštalácia veľmi jednoduchá pomocou jediného príkazu:

```
apt-get install elasticsearch
```

5.1.4 Zobrazovanie dát

Pre zobrazovanie dát, môžeme využiť poslednú vrstvu ELK stacku a tou je Kibana. Umožňuje prehľadávanie Elasticsearch databáze a vytváranie rôznych grafov. Jej inštalácia, rovnako ako v predchádzajúcich prípadoch, je veľmi jednoduchá, stačí zavolať príkaz:

```
apt-get install kibana
```

A však Kibana ako mocný nástroj, ponúkajúci mnoho nastavení a spôsobov ako zobrazovať dáta, ponúka jediný spôsob ako tieto dáta z nej môžeme zobrazovať v iných aplikáciach a to pomocou iframov. Ak by bolo potrebné zobrazovať viacero grafov z Kibany v našej aplikácií tak za použití niekoľkých iframov by došlo k výraznému spomaleniu pri načítaní stránky. Z tohoto dôvodu bolo nutné implementovať vlastný spôsob vyhľadávania v databáze a zobrazovania dát v grafoch. Pre implementáciu bol použitý *php Yii2 framework*, pomocou ktorého bola implementovaná zobrazovacia aplikácia. Bližšie informácie o tom, ako bola aplikácia implementovaná sa nachádzajú v kapitole 5.2.4.

5.2 Konfigurácia aplikácií

Ďalším krokom bolo potrebné správne nastavenie jednotlivých aplikácií tak, aby získané dáta prešli správne celým procesom od získania až po uloženie.

5.2.1 Filebeat

Pre konfiguráciu Fielebeat sa používa súbor **filebeat.yml**, ktorý sa po inštalácií nachádza v zložke **/etc/filebeat/filebeat.yml**. Najprv je potrebné nastaviť vstup, z ktorého má Filebeat čítať. Toto sa nastavuje pomocou **filebeat.inputs**. Je možné nastaviť viacero vstupov a tie sú v súbore uložene ako YAML list a každý z nich začína s "–". Ďalej je potrebné nastaviť kam má Filebeat posielať získané dáta. Filebeat ponúka viacero možností kam posielať dáta, ako napríklad priamo do Elasticsearch, konzole, súboru alebo Logstash... My sme použili možnosť Logstash, keďže dáta ešte chceme upravovať. Tento výstup nastavíme pomocou **output.logstash** a **host**. Príklad ako by mal vyzerať konfiguračný súbor je následovný:

```
filebeat.inputs:
- type: log
  paths:
    - /var/log/system.log
    - /var/log/wifi.log
- type: log
  paths:
    - "/var/log/apache2/*"
  fields:
    apache: true

output.logstash:
  hosts: ["127.0.0.1:5044"]
```

5.2.2 Metricbeat

Pre konfiguráciu Metricbeat sa používa súbor **metricbeat.yml**, ktorý po inštalácií nájdeme v zložke **/etc/metricbeat/metricbeat.yml**. Pre aktiváciu modulov z konfiguračného súboru môžeme pridať položky do **metricbeat.modules** zoznamu. Každý modul začína s "–"po ktorom nasledujú nastavenia pre daný modul. Taktiež je potrebné pomocou **output.logstash** a **host** nastaviť kam sa majú dáta odosielať. Pre zbieranie informácii o systéme sa využíva system modul a konfigurácia s týmto modulom môže vyzerať následovne:

```
metricbeat.modules:
  - module: system
    metricsets:
      - cpu
      - memory
    enabled: true
    period: 5s
    processes: [".*"]
    cpu.metrics: ["normalized_percentages"]

output.logstash:
  hosts: ["127.0.0.1:5044"]
```

5.2.3 Logstash

Pre Logstash konfiguráciu sú podstatné dva súbory a to **logstash.yml**, ktorý sa nachádza v zložke **/etc/logstash/logstash.yml** a **logstash.conf** v zložke **/etc/logstash/conf.d/logstas.conf**. V logstash.yml môžeme nastavovať parametre ako napríklad aké veľké dávky dát chceme zbierať pred tým než ich odošleme na výstup, alebo ako dlho čakať na ďalšie dáta pred tým než odošleme menšiu dávku dát a mnoho ďalších. Nám však stačia defaultne nastavené hodnoty a oveľa viac nás zaujímajú nastavenia v súbore logstash.conf. Tento súbor sa skladá z troch častí a to sú vstup, filter, výstup. Na vstupe máme možnosť použiť veľkú škálu pluginov ako napríklad beats, cloudwatch, file, ... Nám vyhovuje plugin beats, ktorému nastavíme port 5044, na ktorý budú chodiť dáta z beat aplikácií. Ďalej pre filtrovanie dát využívame logstash filter pluginy ako je gork, geoip, mutate, drop. Na dáta môžeme uplatňovať rôzne sady filtrov na základe podmienok. Na koniec, aby sa upravené dáta odoslali na správne miesto je potrebné nastaviť výstup. Tu máme taktiež viacero možností kam môže logstash odosielať zozbierané informácie. My ich chceme ukladať a neskôr k ním pristupovať a preto použijeme možnosť odoslania dát do elasticksearchu. Výsledný logstash.conf súbor by mal vyzerať napríklad takto:

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

filter {
    if [log][file][path] =~ "/var/log/apache2/access.log" {
        mutate {
            add_field => { "log_type" => "access_apache" }
        }
    }
    else if [log][file][path] =~ "/var/log/apache2/error.log" {
        mutate {
            add_field => { "log_type" => "error_apache" }
        }
    } else {
        mutate {
            add_field => {
                "log_type" => "others_logs"
            }
        }
    }
}

output {
        elasticsearch {
                hosts => "elasticsearch:9200"
        index => "%{[@metadata][beat]}-%{[agent][hostname]}-%{+YYYY.MM.dd}"
        }
}
```

5.2.4 Elasticsearch

Konfiguračný súbor Elasticsearch je **elasticsearch.yml** a nachádza sa v zložke **/etc/logstash/elasticsearch.yml**. Hlavným nastavením ktoré potrebujeme nastaviť, ak databázu budeme prehľadávať z iného hostu ako z toho, na ktorom sa nachádza, je povolenie **CORS** čo je skratka pre Cross-origin resource sharing. Nastavíme to následovne:

```
http.cors.enabled: true
http.cors.allow-origin: "*"
http.cors.allow-methods: OPTIONS, HEAD, GET, POST, PUT, DELETE
http.cors.allow-headers: "X-Requested-With, X-Auth-Token, Content-Type,
                      Content-Length, Authorization"
```

5.3 Implementácia aplikácie pre zobrazovanie dát

Posledným krokom, ktorý bol potrebný pre správnu analýzu dát, bolo vytvorenie aplikácie pre ich zobrazovanie. Pre vývoj aplikácie sme sa rozhodli použiť PHP framework Yii, pomocou ktorého sme implementovali základnú logiku aplikácie a pre zobrazovanie grafov bola použitá JavaScript knižnica ReactJS.

5.3.1 Migrácia

V aplikácii by sme chceli mať možnosť monitorovať viac serverov a preto je ideálne použiť šablónu pre detail, v ktorej sa budú meniť údaje v závislosti od toho, ktorý server aktuálne pozeráme. Na to, aby bolo toto možné implementovať, je potrebné vytvoriť databázu, v ktorej sa bude nachádzať tabuľka dashboardov a z tejto tabuľky budeme vyberať potrebné údaje do šablóny. Najlepší spôsob ako vytvárať databázu, je pomocou migrácií. Migrácie nám umožňujú vykonávať všetky možné úkony nad databázou ktoré potrebujeme, ako napríklad pridávanie nových tabuliek, aktualizovanie už existujúcich. Ale hlavnou výhodou migrácií je to, že ak počas vývoja používame viac vývojových prostredí (testovanie, réžia, produkcia), nemusíme na každom vytvárať novú databázu ale stačí spustiť migráciu a databáza sa vytvorí automaticky. V prípade, ak sa niečo počas niektorej z migrácií pokazí, jednoducho ju môžeme vrátiť späť. Novú migráciu v Yii frameworku vytvoríme jednoducho pomocou konzolového príkazu:

```
yii migrate/create <name>
```

Každá migrácia je definovaná ako PHP trieda, rozširujúca triedu `yii\db\Migration`. Uvedený príkaz vytvorí novú PHP triedu v zložke `@app/migrations` pod názvom `m200501_185401_name_of_migration.php`. Názov je zložený z UTC dátumu a času, v ktorom bol príkaz spustený a mena, ktoré bolo v príkaze nastavené. Vygenerovaná trieda obsahuje dve metódy `up` a `down`. Metóda up sa používa pre zapísanie zmien, ktoré chceme vykonať v databáze a je volaná keď updatujeme databázu. Metóda down sa používa pre vrátenie zmien, ktoré boli vytvorené up metódou a je volaná keď degradujeme databázu. Avšak nie vždy je možné vrátiť zmeny, ktoré boli vykonané up metódou, ako napríklad ak pri update mažeme riadok alebo stĺpec. V tom prípade je dobré vracať hodnotu `false` v down metóde, aby sme tak indikovali, že migráciu nie je možné zvrátiť. Príklad migrácie môže vyzerať následovne:

```
<?php
use yii\db\Migration;

class m200501_185401_name_of_migration extends Migration
{
    public function up()
    {
    }

    public function down()
    {
        echo "m200501_185401_name_of_migration cannot be reverted.\n";

        return false;
    }
}
```

V našej aplikácii sme využili migrácie pre vytvorenie databázovej tabuľku dashboard, v ktorej budeme uchovávať všetky potrebné dáta pre monitorovanie serverov ako napríklad na akej host adrese sa nachádza daný server, alebo aké metriky z daného serveru chceme monitorovať poprípade akú aplikáciu na danom serveri chceme sledovať.

5.3.2 Active Record

Databázová tabuľka je zaobalená do PHP triedy Active Record a teda každá inštancia objektu odpovedá jednému riadku tabuľky a atribúty objektu sú mapované na stĺpce v tabuľke. Odkazovanie sa na atribút objektu sa rovná prístupu na zodpovedajúci riadok a stĺpec v tabuľke databázy. Jednotlivé Active Record triedy sú považované za modely a preto sú uložené v zložke `@app/models` a názov triedy odpovedá názvu tabuľky podľa konvencie. Nové záznamy vytvárame pomocou `new` operátora, nastavíme ich atribúty a po zavolaní metódy `save()` sú uložené do databázy. K už vytvoreným záznamom môžeme pristupovať pomocou použitia metódy `find()` pre získanie existujúceho záznamu alebo záznamov. Pre vyhľadávanie v databáze, vytvárame nové search triedy, ktoré rozširujú Active Record a použitím rôznych kombinácií vyhľadávacej metódy `find()` môžeme vrátiť hľadaný záznam alebo pole záznamov poprípade Data Provider. Pred uložením je možné skontrolovať správnosť atribútov, napríklad či ich dátový typ zodpovedá očakávanému dátovému typu v databázovom stĺpci. Pravidlá pre jednotlivé atribúty sú nastavené pomocou metódy `rules()`.

Data Provider nám umožňuje zobrazovať filtrované dáta z databázy a pomocou data widgetou ich môže používateľ interaktívne triediť a stránkovať.

Pre potreby našej aplikácie bol vytvorený Dashboard Active Record, ktorý sa stará o ukladanie a získavanie dát do databázovej tabuľku dashboard. K nemu bola vytvorená DashboardSearch trieda, ktorá ho rozširuje a stará sa o získavanie informácií z databázy.

5.3.3 Controller

Controller je C v MVC architektúre. Sú to objekty triedy, ktorá je rozšírením `yii\base\Controller` a sú zodpovedné za spracovanie požiadaviek a generovanie odpovedí. Presnejšie povedané, po prevzatí kontroly z aplikácie, controller zanalyzuje prichádzajúce

požiadavky, posunie ich modelom 5.3.2 a spolu s naplnenými modelami vygeneruje odpoveď, ktorú vráti do potrebného view 5.3.4.

Controller sa skladá z akcií, čo sú najzákladnejšie metódy, ktorým môže používateľ odoslať požiadaviek na spracovanie. V jednom controlleri sa môže nachádzať viac akcií naraz. Controllere uchovávame v aplikácii v zložke `@app/controllers` a ich názov sa skladá z názvu controlléru, ku ktorému pripojíme `Controller`, napríklad `SpecificNameController`. Príklad controlleru pre vykreslenie view môže vyzerať nasledovne:

```
namespace app\controllers;

use Yii;
use app\models\Post;
use yii\web\Controller;
use yii\web\NotFoundHttpException;

class PostController extends Controller
{
    public function actionView($id)
    {
        $model = Post::findOne($id);
        if ($model === null) {
            throw new NotFoundHttpException;
        }

        return $this->render('view', [
            "model" => $model,
        ]);
    }
}
```

K akcii actionView($id) sa vieme dostať poslaním požiadavky na adresu `http://hostname/index.php?r=post/view&id=123`. Kde **post** určuje ktorý controller sa má použiť a **view** určuje ktorá akcia z daného controlleru sa má vykonať. Hodnota 123 bude vložená do parametra akcie a bude použitá pri vyhľadávaní záznamu s daným ID v databáze. Ak daný záznam neexistuje, tak bude vyvolaná výnimka, inak sa zobrazí náhľad požadovaného príspevku.

V našej aplikácii sme použili controllér pre implementovanie CRUD funkcií. Jedná sa o 4 základné funkcie pomocou ktorých môžeme vytvárať, zobrazovať, upravovať alebo zmazávať dáta z databázy.

5.3.4 View

View je zodpovedné za zobrazovanie dát koncovému používateľovi v MVC architektúre. Vo webových aplikáciách sú views vytvárané vo forme šablón, čo sú PHP skripty obsahujúce prevažne HTML kód a podľa potreby PHP kód. Pre ich správu sa používa view komponent, ktorý ponúka bežne používané metódy pre vykresľovanie.

V nasledujúcom príklade je znázornená šablóna pre prihlasovaciu stránku. Môžeme v nej vidieť spomínanú kombináciu PHP a HTML kódu. PHP sa stará o generovanie dynamických častí ako je titulok alebo formulár a HTML ich organizuje do zoskupenia ako by mali na stránke vyzerať. Pre view je podstatná **$this** premenná, pomocou ktorej pristupujeme

ku view komponentu pre správu vykresľovania. Okrem tejto premennej, je možné používať aj ďalšie ako $form, $model. Tieto premenné reprezentujú dáta, ktoré boli odovzdané do view cez controller alebo iný objekt, ktorý spustil vykreslenie šablóny.

```
<?php
use yii\helpers\Html;
use yii\widgets\ActiveForm;

/* @var $this yii\web\View */
/* @var $form yii\widgets\ActiveForm */
/* @var $model app\models\LoginForm */

$this->title = 'Login';
?>
<h1><?= Html::encode($this->title) ?></h1>

<p>Please fill out the following fields to login:</p>

<?php $form = ActiveForm::begin(); ?>
    <?= $form->field($model, 'username') ?>
    <?= $form->field($model, 'password')->passwordInput() ?>
    <?= Html::submitButton('Login') ?>
<?php ActiveForm::end(); ?>
```

V našom projekte boli použité šablóny pre zobrazenie prehľadu a detailu vytvorených dashboardov a spoločná šablóna pre ich vytvorenie a úpravu. Pre zobrazenie prehľadu dashbordov bol použitý **GridView** widget, ktorý nám umožnil ich stránkovanie a triedenie.

Name	Memory	Cpu	Os	Browsers	Unique Users	
nemeckamista						
Localhost						
Bilovice						
Remax						
Sobriety						

« 1 2 »

Showing **1-5** of **6** items.

Obr. 5.1: Grid view

Formulár v šablóne pre vytvorenie a úpravu dashboardov bol implementovaný pomocou **ActiveForm**. ActiveForm by mal byť použitý pre generovanie formulárov vždy, pokiaľ sa jedná o formulár s napojením na model. Vďaka tomuto prepojeniu sme schopní, pomocou pravidiel nastavených pre model, kontrolovať správnosť odoslaných hodnôt vo formulári. Pre generovanie grafov v detaile dashboardu bola použitá knižnica grafov Recharts, implementovaná pomocou JavaScript knižnice ReactJS. Pomocou tejto knižnice sme mohli implementovať viacero typov grafov, napríklad koláčový graf pre výpis počtu pripojení použitím určitého webového prehliadača alebo operačného systému, stĺpcový graf pre výpis počtu unikátnych prístupov na stránku v jednotlivých dňoch a čiarový graf pre zobrazovanie využitia serverových zdrojov ako CPU a pamäť. Všetky Recharts komponenty sú od seba

viditeľne oddelené. Napríklad LineChart je zložený z x/y os, tooltip, grid, a line položiek. Všetko sú to samostatné React komponenty čo robí z Recharts prehľadnú a jednoducho použiteľnú knižnicu. Príklad použitia LineCharts môže byť nasledovný:

```
<LineChart data={this.state.data}
          margin={{top: 5, right: 30, left: 20, bottom: 5}}>
    <XAxis dataKey="time" reversed={true}/>
    <YAxis domain={[0, 100]}/>
    <Tooltip/>
    <Line type="monotone" dataKey="value" stroke="#8884d8"
          name={`${this.props.monitoringUnit} usage`}
          dot={false}/>
</LineChart>
```

5.3.5 Cron

Cron je plánovač akcií, ktorý sa používa v unixových systémoch. V daných systémoch je spustený ako démon na pozadí a v periodických intervaloch spúšťa akcie, ktoré mu boli definované. V aplikácií je využitý pre kontrolu, či nedošlo k prekročeniu potencionálne kritickej hranice využitia serverových zdrojov. V prípade že áno, aplikácia zaznačí túto udalosť do databázy a emailom upozorní používateľa. Inštalácia Cron prebieha pomocou príkazu:

```
apt-get install cron
```

Pre prístup ku konfigurácii je použitý príkaz:

```
crontab -e [username]
```

konfigurácie príkazu vyzerá následovne:

```
.---------------- min (0 - 59)
| .------------- hod (0 - 23)
| | .---------- den v mesiaci (1 - 31)
| | | .------- mesiac (1 - 12) alebo jan,feb,mar,apr ...
| | | | .---- den v tyzdni (0 - 6) (nedela=0 or 7) alebo sun,mon,tue,...
| | | | |
* * * * * prikaz ktory chceme vykonat
```

Kapitola 6

Nasadenie

Nasledujúca kapitola popisuje postup, ktorý treba dodržať pri nasadení aplikácie, aby aplikácia správne fungovala.

6.1 Nastavenie serverov

Na jednotlivých serveroch je potrebné nainštalovať aplikácie, ktoré umožňujú zhromažďovanie informácií o daných serveroch. Na každý server, ktorý chceme sledovať, je potrebné nainštalovať aplikácie pre zbieranie dát **Filebeat** a **Metricbeat**.

Ďalej je potrebné nainštalovať aplikácie, ktoré sa budú starať o centralizovanie získaných informácií a ich ukladanie. Pre zbieranie dát zo všetkých zberacích aplikácií slúži **Logstash**, ktorý zozbierané dáta odošle na úschovu do databázy **Elasticsearch**. Tieto dve aplikácie stačí mať nainštalované iba na jednom serveri, napríklad jeden z tých ktoré sú sledované alebo nejaký iný.

Postup ako nainštalovať všetky tieto aplikácie je popísaný v sekcii 5.1.

6.2 Spustenie aplikácie

Pre spustenie aplikácie je potrebné nahrať aplikáciu na server, na ktorom bude aplikácia spustená. Po nahratí je nutné nainštalovať dodatočné balíčky, potrebné pre beh aplikácie.

```
composer install
```

Ďalej je potrebné inicializovať aplikáciu podľa toho či je aplikácia určená pre ďalší vývoj, vývojové prostredie alebo pre produkciu, produkčné prostredie.

Keďže aplikácia využíva databázu, taktiež treba nastaviť prístup do nej. Konfigurácia prístupu je uvedená v súbore `config/db.php` a môže vyzerať takto:

```
<?php
return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=db_host;dbname=db_name',
    'username' => 'db_user',
    'password' => 'db_pswd',
    'charset' => 'utf8',
];
```

Po vytvorení pripojenia na databázu, je potrebné v databáze vytvoriť štruktúru, s ktorou bude aplikácia pracovať. Na to slúži príkaz, ktorý spustí databázové migrácie.

```
php yii migrate
```

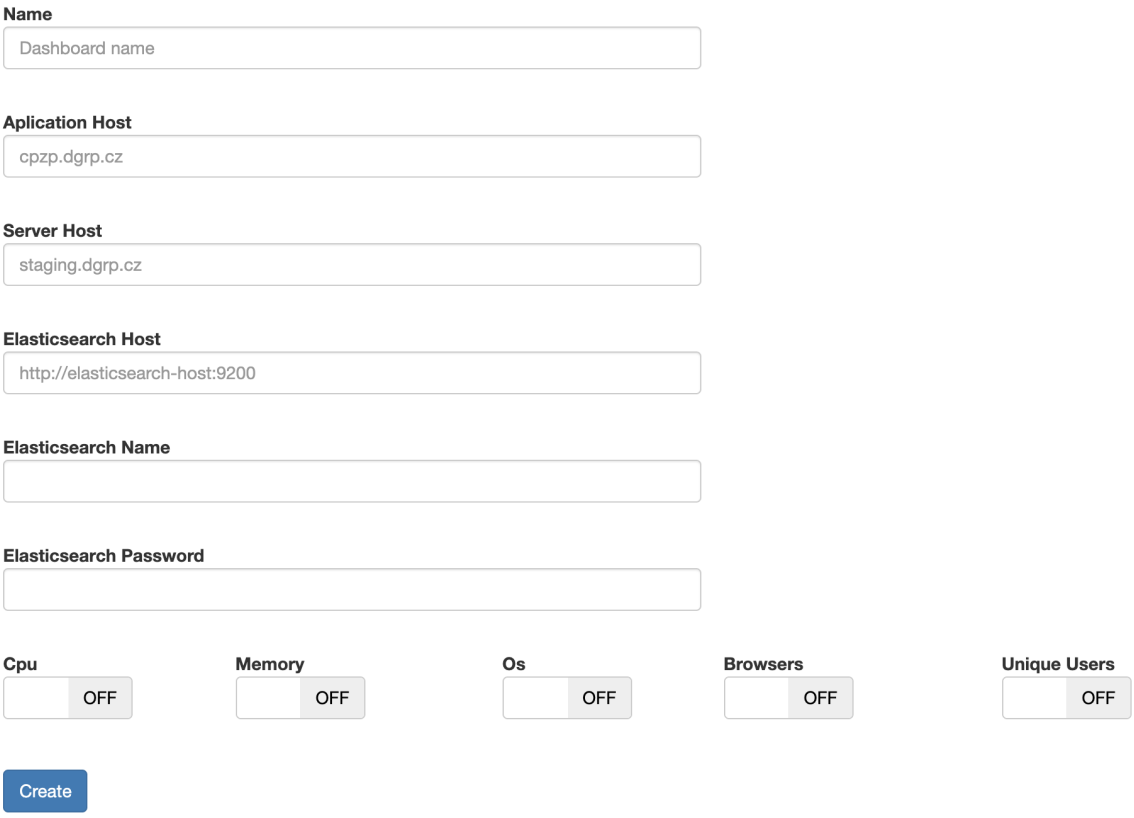
Pre správne posielanie emailov pomocou Cronu na serveri, je potrebné v súbore `config/console.php` správne nastaviť komponent `mailer`. Príklad nastavenia je:

```
'components' => [
    'mailer' => [
        'class' => yii\swiftmailer\Mailer::class,
        'useFileTransport' => false,
        'enableSwiftMailerLogging' => true,
        'transport' => [
            'class' => Swift_SmtpTransport::class,
            'encryption' => 'tls',
            'host' => 'smtp.gmail.com',
            'port' => '587',
            'username' => 'email@gmail.com',
            'password' => 'emailPswd',
        ],
    ],
],
```

6.3 Vytvorenie dashboardu

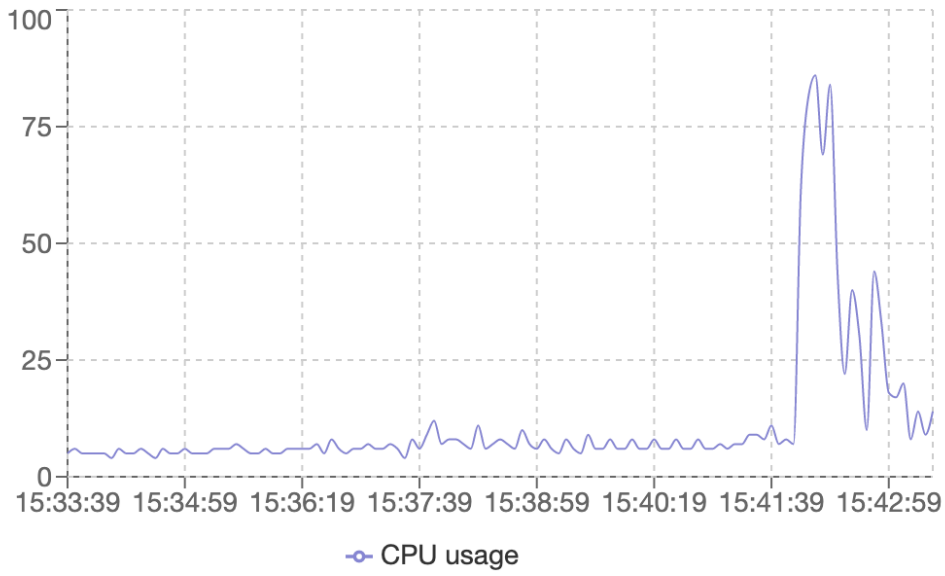
Po spustení aplikácie je možné začať monitorovať servery. Domovská stránka zobrazuje prehľad serverov, ktoré monitorujeme. Pre nastavenie nového serveru je potrebné vytvoriť nový dashboard.

Na obrázku 6.1 vidíme, že jednotlivým serverom je možné nastaviť niekoľko parametrov. **Server host** a **Aplication host** sú host adresy, serveru a aplikácie bežiacej na danom servery. **Elasticsearch host** je adresa, na ktorej sa nachádza databáza Elasticsearch, v ktorej sa nachádzajú získané informácie. **Elasticsearch name** a **Elasticsearch password** sú prístupové údaje do Elasticsearch databázy.



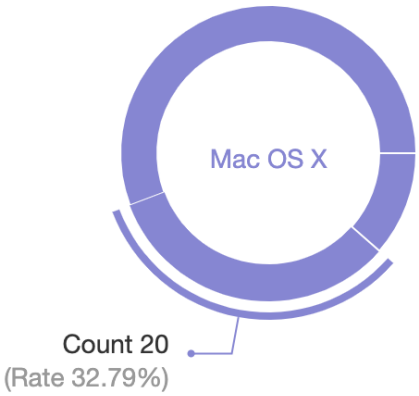
Obr. 6.1: formulár pre vytvorenie/úpravu dashboardu

Po vytvorení Dashboardu je možné v jeho detaile sledovať niekoľko grafov. Pomocou čiarového grafu 6.2 aplikácia znázorňuje vyťaženie procesoru alebo zaplnenie pamäte na serveri.



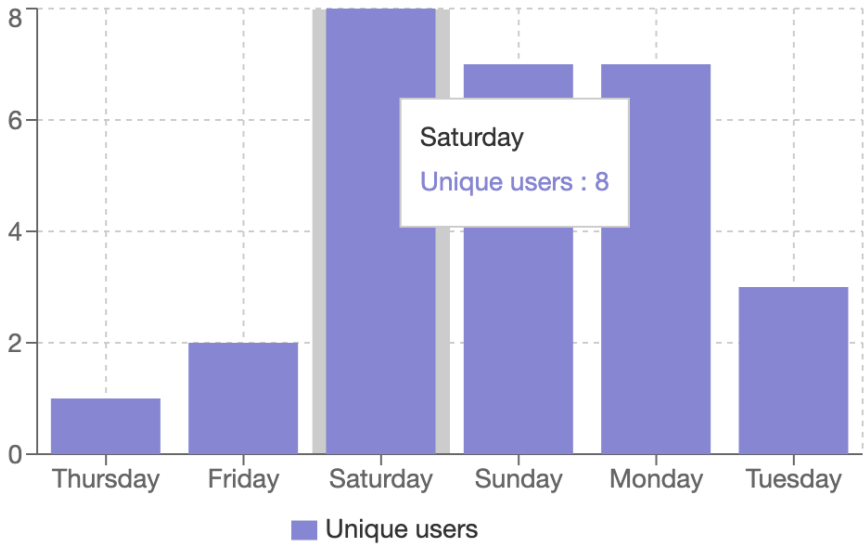
Obr. 6.2: Graf zaťaženia CPU.

Koláčové grafy 6.3 znázorňujú početnosť, koľko požiadaviek na aplikáciu, bežiacu na serveri, bolo poslaných zo špecifického prehliadača alebo operačného systému.



Obr. 6.3: Graf počtu prístupov.

Stĺpcový graf 6.4 zobrazuje počet jedinečných prístupov na stránku v jednotlivých dňoch v týždni.



Obr. 6.4: Graf jedinečných prístupov v dňoch.

6.4 Spustenie Cron plánovača

Aby bolo možné použiť Cron pre sledovanie prekročenia kritickej hranice využitia serverových zdrojov a zaslanie upozornenia používateľovi pomocou emailu, bol vytvorený konzolový controller, ktorý toto zabezpečuje. Pre periodické spúšťanie akcií v controlleri je potrebné vytvoriť akciu v crontab, ktorá bude vyzerať napríklad následovne:

```
* * * * * php /appLocation/yii controllerName/actionName
```

Uvedený príkaz spúšťa definovanú akciu periodicky každú minútu.

Kapitola 7

Testovanie

Kapitola testovanie popisuje, akým spôsobom bola aplikácia testovaná a ako vyzeralo testovacie prostredie.

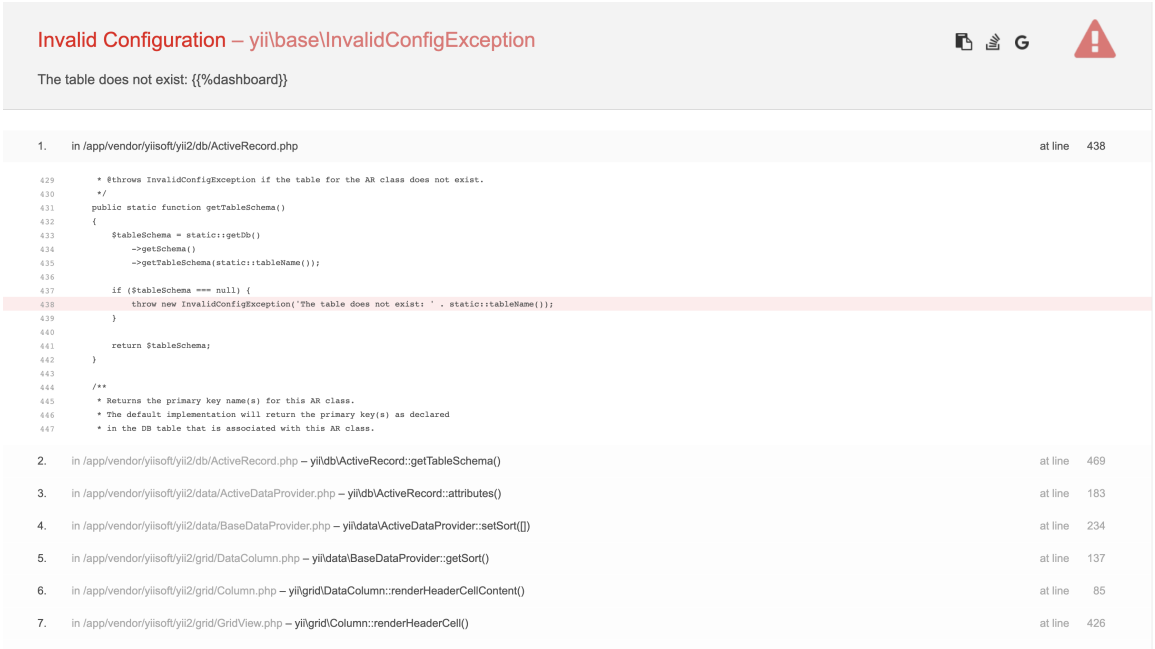
7.1 Docker

Pre vývoj aplikácie a nastavenie testovacieho prostredia bol použitý Docker. Docker umožňuje vytvárať izolované prostredia, nazývané kontajnery. Výhodou kontajnerov je, že ich môže byť spustených viac na jednom zariadení súčasne. Pri použití docker compose, je možné vytvárať aplikácie zložené z viacerých kontajnerov.

S použitím týchto funkcionalít, bol vytvorený `docker-compose.yml` súbor, ktorý obsahuje nastavenia pomocou ktorých vytvorí kontajnery, ktoré simulujú reálny systém. Pre aplikáciu bol vytvorený kontajner, ktorý simuluje Apache server na ktorom bude bežať aplikácia a kontajner pre SQL databázu, ktorú aplikácia používa. Ďalej boli vytvorené kontajnery pre servery, ktoré chceme monitorovať. Na týchto kontajneroch boli spustené aplikácie Metricbeat a Filebeat. Pomocou použitia špeciálnych obrazov (images) pre Elasticsearch, Logstash a Kibanu boli vytvorené 3 kontajnery, ktoré spolu simulovali server, na ktorom sa ukladajú údaje.

7.2 Aplikácia

Po spustení docker compose je aplikácia dostupná na štandardnom porte 80 vďaka mapovaniu docker portov a je k nej možný prístup pomocou internetového prehliadača na adrese `localhost`. Aplikácia bola počas vývoja inicializovaná do vývojového prostredia. Vďaka tomuto prostrediu je generovaných viac log správ a pri chybe v aplikácii sa vypíše stack trace 7.1 priamo do prehliadača, vďaka čomu je hľadanie, kde chyba nastala oveľa jednoduchšie.



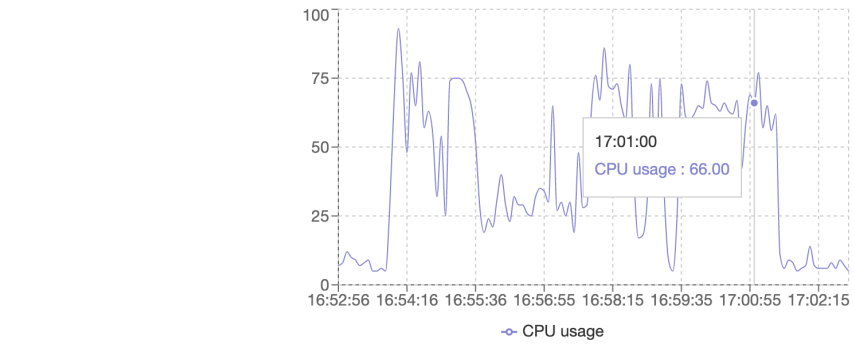
Obr. 7.1: Príklad výpisu stack trace na stránke.

V obrázku 7.1 je možné vidieť, že tabuľka **dashboard** z ktorej sa aplikácia snaží získať informácie, sa v databáze nenachádza. K danému stavu mohlo dôjsť chybou pri zadávaní názvu tabuľky, v ktorej má aplikácia vyhľadávať alebo požadovaná tabuľka ešte nebola vytvorená. Ak k tomuto stavu dôjde, je potrebné vytvoriť novú databázovú migráciu, ak ešte neexistuje a spustiť ju, čím sa v databáze požadovaná tabuľka vytvorí.

V aplikácii sa predpokladalo zobrazovanie grafov vytvorených pomocou Kibany. Kibana umožňuje zdieľanie grafov v nej vytvorených pomocou iframov. Pri testovaní bolo však zistené, že použitím viacerých grafov z Kibany na jednej stránke spôsobuje spomalenie načítavania stránky, ktoré potom trvalo viac ako 30 sekúnd, čo je v dnešnej dobe naozaj veľmi pomalé. Z tohoto dôvodu bolo implementované zobrazovanie grafov priamo v aplikácií, ktorá ich zobrazuje na základe získaných informácií z Elasticsearch databázy. Toto výrazne zrýchlilo načítanie stránky, ktoré aktuálne trvá v rozmedzí od 0.5-2s. Merania boli vykonané pomocou rozšírenia v prehliadači Google Chrome **Page load time**.

Aj keď Kibana v konečnom dôsledku nebola použitá pri zobrazovaní dát, pri testovaní bola využitá jej funkcia **Dev Tools**. Dev Tools umožňuje vytvárať `POST/GET` požiadavky na Elasticsearch databázu, ktoré boli potom použité v aplikácií pri získavaní údajov pre grafy.

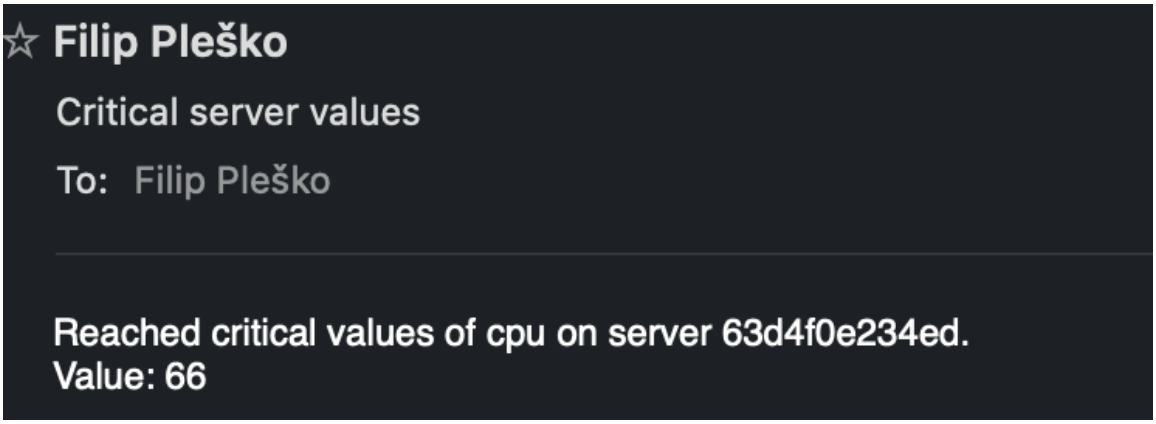
V rámci testovania bolo umelo zvýšené využitie procesoru na serveri, čím sme sledovali správnu funkčnosť monitorovania rizikových hodnôt využitia serverových zdrojov. 7.2



Time	Resource	Value
2020-05-25 17:01:04	cpu	66%

Obr. 7.2: Znázornenie zaznamenania rizikových hodnôt využitia serverových zdrojov.

V prípade prekročenia kritickej hodnoty je taktiež vygenerovaný email, ktorý používateľa upozorní o prekročení hranice 7.3.



Obr. 7.3: Ukážka emailovej pripomienky prekročenia kritickej hranice.

7.3 Grok modul

Grok je modul, použitý pri filtrovaní dát pomocou Logstash. Po aplikovaní zmeny do Logstash konfiguračného súboru bolo potrebné reštartovať composer, čo bolo ale veľmi časovo náročné a ak nebol filter správny, bolo potrebné celý proces opakovať. Na to aby bolo zabezpečené, že do konfiguračného súboru bol vložený správny filter, bol filter najskôr otestovaný v online Grok debugeri [1].

[1] Grok debuger - https://grokdebug.herokuapp.com/

Kapitola 8

Záver

Cieľom práce bolo zoznámiť sa s problematikou monitorovania serverov a na základe získaných znalostí implementovať aplikáciu, ktorá by umožňovala grafické sledovanie stavu servera.

V priebehu tvorenia práce bolo zistené, že použitie ELK stack v spolupráci s beat aplikáciami je ideálne pre získavanie informácii o serveroch a preto boli tieto aplikácie aj v práci využité. Ďalej bola implementovaná aplikácia v PHP frameworku Yii, ktorá zobrazuje informácie ako sú využitie procesoru, zaplnenie pamäte na serveri alebo počet rôznych druhov prístupov do aplikácií, ktoré bežia na danom serveri.

Zobrazovanie grafov v implementovanej aplikácii bolo najprv myslené pomocou zdieľania grafov z Kibany. Tá však ale ponúka zdieľanie grafov iba pomocou iFramov, čo pri použití viacerých grafov na jednej stránke výrazne spomaľovalo načítanie danej stránky a z tohoto dôvodu bola nakoniec použitá React knižnica Recharts pre generovanie vlastných grafov.

Prvé dve kapitoly práce popisujú problematiku monitorovania serverov, porovnávajú rôzne nástroje, ktoré sa v dnešnej dobe používajú v súvislosti s monitorovaním. V nasledujúcej, tretej kapitole, je popísaný návrh riešenia, ktorý bol použitý v tejto práci. Štvrtá kapitola detailne popisuje technológie, ktoré boli použité pri implementácii aplikácie. Postup pri implementácii sa nachádza v piatej kapitole.

Výsledkom práce je aplikácia, ktorá je schopná monitorovať niekoľko serverov spolu s webovými aplikáciami, ktoré na týchto serveroch bežia. Aplikácia v momentálnom stave umožňuje sledovať procesor, pamäť serveru a logovacie správy z aplikácií. Do budúcna je možné aplikáciu rozšíriť o sledovanie viacerých serverových zdrojov, ak by bolo potrebné, poprípade zobrazovať dodatočné štatistiky o webových aplikáciách.

Výsledná aplikácia je zverejnená ako open-source na verzovacom systéme Github, na adrese: https://github.com/Plech0/Application-Service-and-Server-Monitoring

Literatúra

[1] DOCS, M. web. *Javascript* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript.

[2] ELASTIC. *Elasticsearch* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.elastic.co/elasticsearch/.

[3] ELASTIC. *ELK pricing* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.elastic.co/subscriptions.

[4] ELASTIC. *Filebeat* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html.

[5] ELASTIC. *Logstash* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.elastic.co/guide/en/logstash/current/introduction.html.

[6] ELASTIC. *Metricbeat* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.elastic.co/guide/en/beats/metricbeat/current/metricbeat-overview.html.

[7] INC., S. *Splunk pricing* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.splunk.com/en_us/software/pricing/enterprise-and-cloud.html.

[8] JABBADUHNUTT. *Graylog pricing* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.reddit.com/r/graylog/comments/bd4jlt/question_regarding_enterprise_pricing/.

[9] LLC, Y. S. *Yii* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.yiiframework.com/doc/guide/2.0/en/intro-yii.

[10] SOLARWINDS WORLDWIDE, L. *Loggly pricing* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.loggly.com/plans-and-pricing/#.

[11] STATION, I. C. *Software reviews* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.itcentralstation.com/products/comparisons/elk-logstash_vs_graylog_vs_splunk.

[12] STATION, I. C. *TrustRadius* [online]. 2020 [cit. 15. mája 2020]. Dostupné z: https://www.trustradius.com/products/loggly/reviews.

[13] TECHOTOPIA. *PHP* [online]. 2016 [cit. 15. mája 2020]. Dostupné z: https://www.techotopia.com/index.php/An_Overview_of_PHP.

Príloha A

Obsah CD

- `/assets` - assety použité pre dizajn šablón
- `/config` - konfiguračné súbory pre aplikáciu, napr. nastavenie databázy
- `/controllers` - controlleri pre logiku aplikácie
- `/migrations` - databázové migrácie pre inicializáciu databáze
- `/models` - modely použité v aplikácii
  - `/search` - vyhľadávacie modely pre prehľadávanie databáze
- `/view/**/*.php` - šablóny pre jednotlivé stránky aplikácie
- `/web` - zdrojové kódy assetov
- `/elk` - konfiguračné súbory pre elk stack
- `/servers` - konfiguračné súbory pre simulované, sledovacie servery
- `docker-compose.yml` - konfiguračný súbor pre spustenie Docker kontajnerov
- `composer.json` - registrácia rozšírení použitých v aplikácii