Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/24BE17Marinkovic

УПРАВЉАЊЕ ДОГАЂАЈИМА У СОФТВЕРСКИМ СИСТЕМИМА ЗАСНОВАНИМ НА ТЕХНОЛОГИЈАМА РАЧУНАРСТВА У ОБЛАКУ

LOG MANAGEMENT IN SOFTWARE SYSTEMS BASED ON CLOUD TECHNOLOGIES

Милан Маринковић, Горан Савић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду представљен је софтверски систем за управљање догађајима заснован на технологијама рачунарства у облаку. При реализацији овог система, коришћен је ELK stack скуп алата. Предложени систем се састоји из агената чији задатак је прикупљање догађаја, Kafka система који прима прикупљене догађаје од агената, Logstash алата који врши процесирање догађаја, Elasticsearch базе података у којој се складиште догађаји и Kibana алата за визуализацију догађаја. При реализацији система је коришћена IaaS категорија рачунарства у облаку. Систем представља push-based тип система. Догађаји који се прикупљају су догађаји оперативног система, апликативни догађаји, те догађаји nginx сервиса, а могуће је прикупљати и произвољне догађаје. Имплементирана је апликација која представља друштвену мрежу за објављивање фотографија, која користи овај систем за управљање догађајима.*

Кључне ријечи: *управљање догађајима, рачунарство у облаку, ELK stack скуп алата*

Abstract – *The paper proposes a software system for log management based on cloud technologies. The proposed architecture uses ELK stack tools. It consists of agents that collect events, a Kafka system for receiving events from agents, a Logstash tool for processing events, an Elasticsearch database for storing events, and a Kibana tool for visualizing events. The system uses the IaaS category of cloud computing. It represents a push-based type of system. Events collected by the system are events of the operation system, application events, and nginx events. The system can fetch custom events. It's implemented social media application for publishing images that use this system for log management.*

Keywords: *log management, cloud computing, ELK stack tools*

1. УВОД

Управљање догађајима је један од важнијих, али и сложенијих елемената при развоју софтверских система. Догађај представља било коју акцију у систему и углавном укључује покушај промјене стања система. Догађај углавном садржи вријеме, акцију, и било који детаљ у вези са догађајем или окружењем, који може помоћи у објашњењу или разумијевању посљедица и ефеката које је тај догађај произвео [1].

Управљање догађајима подразумијева рад са великом количином података коју је потребно прикупити, чувати, анализирати и на адекватан начин одреаговати уколико се уоче догађаји који могу изазвати штету на систему. Управљање догађајима често није само одлука организације да имплементира овај систем, него и законска обавеза, посебно уколико се ради са осјетљивим подацима. Због тога је потребно дизајнирати систем који ће моћи да задовољи све захтјеве како саме организације, тако и правних регулатива. Дизајнирање система за управљање догађајима је сложен и спор процес, те се организације често одлучују на постојећа рјешења, али је њихова цијена веома висока [1, 2].

Рад се бави начинима евиденције догађаја у софтверским системима, које кораке је потребно имплементирати, на који начин, о чему је потребно водити рачуна при имплементацији, ... Посебан фокус је дат на евиденцији догађаја у софтверским системима коришћењем рачунарства у облаку, као модерне и доминантно заступљене технике развоја софтвера. У контексту тога је предложено рјешење архитектуре система за управљање догађајима у софтверским системима коришћењем рачунарства у облаку, које је засновано на алатима отвореног кода, који су бесплатни, *ELK stack* [3] скупу алата, као и додатним алатима који могу помоћи у имплементацији овог система. Коришћењем рачунарства у облаку омогућено је да се на једноставан и брз начин испоручи систем, те изврши једноставно скалирање апликација на захтјев.

2. УПРАВЉАЊЕ ДОГАЂАЈИМА КОРИШЋЕЊЕМ *ELK STACK* СКУПА АЛАТА

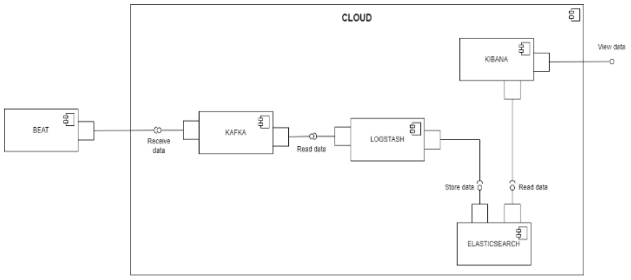
У овом поглављу ће бити детаљније објашњена предложена архитектура система, као и имплементација предложене архитектуре и корака при управљању догађајима.

2.1. Архитектура система

Архитектура система се састоји од *Elasticsearch* базе података, за коју је креиран кластер, *Logstash* и *Kibana* алата, те *Kafka* система. За реализацију овог система је одабрана *IaaS* [4] категорија рачунарства у облаку. На овом серверу се налазе све компоненте које су у вези са управљањем догађајима, како би систем био потпуно изолован и независан од остатка

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Горан Савић, ванр. проф.

система. Док се на свим осталим серверима, на којима се налазе апликације, налазе и *beat* алати. Слика 1 је приказ архитектуре система.



Слика 1. Архитектура система

Сваки сервер са апликацијом, садржи *Metricbeat, Packetbeat, Filebeat, Heartbeat* и уколико је сервер са *Windows* оперативним системом, налази се још и *Winlogbeat* алат. Обзиром да *beat* алати прикупљају податке и шаљу ка централној јединици, ово представља *push-based* архитектуру система, која нам омогућава да на једноставан начин додајемо нове сервере, без да се мора вршити измјена на централној процесорској јединици. Један од главних проблема код *push-based* архитектуре система јесте потенцијална загушеност, јер централна јединица не може да контролише количину података коју прима.

Logstash може да обради око пар десетина хиљада података у секунди, а обзиром да не подржава кластер и троши доста ресурса, није једноставно извршити скалирање. Као алтернатива *Logstash* алату, одлучено је да се користи *Kafka* систем. *Kafka* систем је платформа за стримовање догађаја која има способност да обради преко милион догађаја у секунди. Састоји се од кластера, издавача (енг. *publisher*) и претплатника (енг. *subscriber*). Издавачи шаљу поруке на *Kafka* систем и уписују их у теме (енг. *topic*). Тако ће сви подаци који долазе од *beat* алата ићи на *Kafka* систем и у зависности од самог *beat* алата уписивати у одређену тему.

Након *Kafka* система долази *Logstash* алат који се претплаћује на теме и чита податке из њих, претпроцесира их и онда шаље ка *Elasticsearch* бази података. Ова архитектура нема стандардизован формат логова, него се користи више различитих формата логова. У свакој теми се налази посебан формат података, те је за сваку тему дефинисан посебан *Logstash* софтверски ток (енг. *pipeline*), који на одговарајући начин процесира податке из тема.

Процесирани подаци се даље шаљу ка *Elasticsearch* бази података у којој се складиште, након чега те податке користи *Kibana* алат у ком се врши анализа и визуализација података.

Компоненте *Kafka, Elasticsearch, Kibana* и *Logstash* су покренуте у *Docker* контејнерима. *Docker* је алат који омогућава брзу изградњу, тестирање и испоруку кода. *Docker* пакује софтвер у стандардизоване јединице које се зову контејнери. Контејнер је покренута инстанца софтвера која се креира од *Docker* слика. Обзиром да је потребно покренути више апликација у *Docker* контејнерима, за ту сврху се користи алат *compose*.

2.2. Складиштење логова

За складиштење логова се користи *Elasticsearch* база података. За сваки формат лога је креиран посебан индекс. За логове који долазе са *beat* алата, осим *Filebeat* алата, користе се предефинисане одговарајуће схеме које су креиране од стране *Elastic.co* организације. За све логове које прикупља *Filebeat* алат креиране су посебне схеме које одговарају формату логова које *Filebeat* алат добавља. Сваки индекс има један оригинални фрагмент (енг. *shard*) и по једну копију. На овај начин је обезбијеђена резервна копија података у случају да дође до отказа на једном од чворова. Проблем који постоји је то што су оба чвора на истој машини, ако дође до отказа на самој машини, трајно се губе сви подаци.

Ротација логова је имплементирана на дневном нивоу. Ово је реализовано тако што се у *Logstash* софтверског тока приликом слања података у *Elasticsearch* базу података наводи назив индекса, међутим сви називи су у формату име-ГГГГ.ММ.ДД (ГГГГ – представља годину, ММ – представља мјесец, ДД – представља дан), при чему ће сваки дан индекс имати нови назив, те ће *Elasticsearch* то третирати као посебан индекс. Међутим, проблем који се овдје јавља је велика количина индекса која може значајно да троши ресурсе система. Како би се број индекса смањио, дефинисана је скрипта која се позива сваких седам дана и која позива *Elasticsearch API* који врши реиндексирање, тако што све индексе настале у тих седам дана спаја у један већи индекс, те се на тај начин број индекса смањује седам пута. Поред овога, путем *Kibana* алата је дефинисана и политика животног вијека индекса. Сви индекси имају исту политику животног вијека, која се састоји из четири фазе: врућа фаза, топла фаза, хладна фаза и фаза брисања.

У врућој фази се налазе најновији индекси који ће се највише претраживати и индексирати. Због тога је у сврху ових индекса извршена оптимизација која пружа најбоље перформансе за претраживање и индексирање докумената, али и троши највише ресурса. Индекси у овој фази буду 30 дана од дана креирања.

У топлој фази се налазе индекси старији од 30 дана који се и даље често претражују, али се њихов садржај јако ријетко модификује, те су оптимизовани тако да пруже најбоље перформансе при претраживању, али су перформансе индексирања доста лошије у односу на врућу фазу. У овој фази се налазе индекси стари између мјесец дана и два мјесеца.

Хладна фаза садржи индексе који су старији од два мјесеца и који се ријетко модификују, али и ријетко претражују, те је њихова употреба веома ријетка и чувају се углавном ради задовољавања правних регулатива или *retention* периода. У овој фази, индекси проводе три мјесеца и заузимају најмање меморијског простора.

Обзиром да у овом раду није имплементирано архивирање логова, након неког временског периода, сви логови који изађу из хладне фазе, прелазе у фазу брисања, у којој се врши трајно брисање свих података у индексима и они се заувијек губе. *Elasticsearch*

подржава могућност архивирања логова, тако што ће се дефинисати локација, на којој ће се чувати подаци, и период након ког се врши архивирање логова. *Elasticsearch* подржава архивирање логова на *Azure, Google Cloud Storage, AWS S3* и на фајл систему сервера на ком се налази *Elasticsearch.*

2.3. Анализа и визуализација података

Анализа података у овом систему се ради ручно. Прате се визуализовани подаци, те уколико се уоче одређени проблеми или потенцијалне опасности извршавају се додатни кораци, који би углавном представљали додатно истраживање података постављањем одговарајућих упита ка бази података.

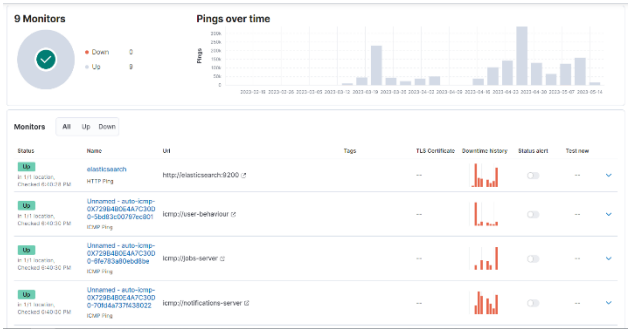
Визуализација података је одрађена путем *Kibana* алата. Визуализовани су сви подаци свих индекса. Подаци који су се визуализовали су учитавани из *Elasticsearch* базе података. Приликом креирања визуализације за *Metricbeat, Packetbeat, Auditbeat* и *Winlogbeat* алате, искоришћени су постојећи темплејти креирани од стране *Elastic.co* организације, који су укључени приликом дефинисања самог *beat* алата. Приликом дефинисања темплејта, може се користити подразумијевани темплејт или навести путања до жељеног темплејта, с тим да схема података мора да одговара схеми коју очекује дати темплејт.

Metricbeat има могућност праћења метрика машине на којој се извршава, *Docker* контејнера, *Kubernetes, AWS, Azure, Nginx* сервиса, и многих других сервиса. *Winlogbeat* је задужен за праћење догађаја на машини са *Windows* оперативним системом.

Примарно се прате успјешне и неуспјешне пријаве на систем, број покренутих и стопираних процеса, активни процеси, догађаји и ко је креирао дате догађаје, ниво догађаја (грешка, информативни или догађаји упозорења), ... *Packetbeat* је искоришћен за прикупљање података о саобраћају на машини. *Packetbeat* подржава различите протоколе, као што су *TLS, HTTP, ICMP, DNS, MySQL,...,* те за сваки од тих протокола има предефинисан темплејт за визуализацију. За *Heartbeat* не постоји такав темплејт, него *Kibana* има посебну страницу у дијелу *Observabilities* > *Uptime* > *Monitors,* на којој могу да се прате стања свих сервиса. Овдје се налазе информације о сервисима који се прате, њиховом тренутном статусу, адреси на којој се налазе, протоколу путем којих се провјерава њихово стање, историји неактивности, итд.

Поред претходно наведених предефинисаних визуализација, креиране су и двије ручне визуализације чији је циљ да визуализују логове апликација и логове са *nginx* сервиса.

На слици 2 је приказан изглед странице на којој се прате стања сервиса. У овом примјеру се налазе четири сервиса, од којих је један и сам *Elasticsearch* чије се стање провјерава путем *HTTP* протокола, док се стања осталих сервиса провјеравају путем *ICMP* протокола. Такође, на слици се могу видјети и периоди неактивности одређених сервиса.



Слика 2. *Приказ статуса сервиса који се прате путем Heartbeat алата*

3. СТУДИЈА СЛУЧАЈА

Ради демонстрације претходно написаног, урађена је студија случаја. Имплементирана је апликација која представља друштвену мрежу на којој се могу објављивати фотографије корисника, лајковати и коментарисати фотографије. Апликација је заснована на микросервисној архитектури при чему апликација има реализован и систем за управљање догађајима који је претходно објашњен.

Микросервиси који постоје у апликацији су *users-ms, posts-ms, medias-ms, notifications-ms, messages-ms* и *user-activities-ms.* Сви микросервиси осим *user-activities-ms* микросервиса имају исту архитектуру која се састоји од базе података, апликације и *filebeat* алата који прикупља логове апликације и просљеђује их ка *Kafka* систему на тему *application-logs. User-activities-ms* микросервис се састоји само од апликације која је писана у програмском језику *Node.js.* Сва комуникација између микросервиса иде путем *RabbitMQ* система. Такође, сваки микросервис је покренут у *Docker* контејнеру. Сви захтјеви који пристижу од клијентске апликације иду преко *api gateway* микросервиса, гдје се налази *nginx* сервис, који пристигле захтјеве даље просљеђује одговарајућем микросервису. *Api gateway* микросервис такође посједује *Filebeat* алат који прикупља логове генерисане од стране *nginx* сервиса и просљеђује их ка *Kafka* систему на тему *nginx-logs.*

Систем за управљање догађајима је реализован употребом *IaaS* категорије рачунарства у облаку. Самом серверу се приступа путем *ssh* протокола, на порту 22, док су остали портови, осим портова 4500 и 5601, гдје се порт 4500 користи за приступ *Kafka* систему, а порт 5601 се користи за приступ *Kibana* алату, затворени како би се повећала безбједност сервера и апликација које се на њему извршавају.

Kafka систем подразумијевано поруке у темама чува 168 сати, међутим, у овом систему је подешено да се поруке чувају 24 сата, односно један дан, како би се сачувао меморијски простор потребан за складиштење података. У *Elasticsearch* бази података, одабран је интервал освјежавања од 30 секунди, како би се добило на перформансама приликом индексирања, јер претрага у реалном времену није примарна у случају управљања догађајима.

Креиран је *user-activities-ms* микросервис чија је сврха да прати понашање корисника у апликацији путем догађаја које креира. Догађај се креира сваки пут када корисник изврши једну од акција. Акције које се прате су ажурирање профила, објављивање фотографије, слање поруке, запраћивање и отпраћивање корисника, лајковање слике и коментарисање слике.

Клијентска апликација, када корисник изврши акцију, шаље догађај до *api-gateway* микросервиса, који затим то просљеђује до *user-activities-ms* микросервиса. *User-activities-ms* микросервис примљени догађај просљеђује даље до *Kafka* система на тему *user-activities*. Обзиром да је додат нови топик на *Kafka* систем, који не постоји иницијално у систему за управљање догађајима, било је потребно креирати и посебан софтверски ток у *Logstash* алату, као и посебан индекс и схему у *Elasticsearch* бази података.

За потребе ових догађаја креирана је визуализација података у *Kibana* алату. Креирана су два графа. Први граф приказује број догађаја током времена, на основу чега се може закључити у којим периодима корисници највише користе апликацију. На другом графу је приказан проценат акција које су корисници извршили, што даје увид у то које акције корисници најчешће извршавају. Такође, комбиновањем визуализације са додатним упитима, може се закључити какво је понашање појединачних корисника у апликацији.

4. ЗАКЉУЧАК

У овом раду је представљен систем за управљање догађајима који је заснован на технологијама рачунарства у облаку. Имплементиран је систем за управљање догађајима по предложеном дизајну за једну апликацију. Сам систем за управљање догађајима је имплементиран коришћењем *ELK stack* скупа алата у комбинацији са *Kafka* системом. Читав систем је покренут користећи *Docker* контејнере и рачунарство у облаку. Одабрана је *IaaS* категорија рачунарства у облаку. Кораци који су реализовани у овом раду су прикупљање догађаја, које се врши путем употребе *beat* алата, процесирање података употребом *Logstash* алата, складиштење података коришћењем *Elasticsearch* базе података, док је визуализација и анализа података имплементирана уз помоћ *Kibana* алата.

Произвољан формат логова, те да се једноставно могу додати нови крајњи сервери чији се догађаји прате, су неке од предности овог система. Такође, скалирање базе података и *Kafka* система је веома једноставно јер се извршава у кластерима. Додатно је могуће по потреби додати и нове визуелне графике или измијенити постојеће. Захваљујући *KQL* упитном језику могуће је једноставно поставити и сложене упите за претрагу и филтрирање података, а захваљујући индексној структури *Elasticsearch* базе података, перформансе претраге су на високом нивоу, чак и када се ради о великој количини података.

Уколико се жели додати нови формат догађаја, потребно је дефинисати нову тему на *Kafka* систему, те дефинисати нови софтверски ток у *Logstash* алату, док је такође потребно креирати и нову схему за дати формат у *Elasticsearch* бази података, што додавање новог формата чини веома сложеним из угла конфигурације новог формата. Додавање новог софтверског тока на *Logstash* алату захтјева да се поново покрене инстанца *Logstash* алата. Такође, уколико дође до промјене формата постојећих логова у тренутку када постоји велика количина података за тај формат, потребно је извршити реиндексирање података, што је процес који при великој количини података троши велику количину ресурса система и може да траје данима

Анализа и надгледање података, у предложеној архитектури система за управљање догађајима, се своде на анализу и посматрање од стране човјека, односно, лица задужених за овај посао. Додатно би се рад могао проширити реализацијом система који би аутоматски могли да надгледају и анализирају податке. Реаговање на догађаје представља још један корак који се извршава мануелно и који би се могао аутоматизовати интеграцијом постојећег система са неким новим системом који би када се деси одређени догађај извршио унапријед дефинисану акцију за тај догађај. Такође, пожељно би било имплементирати и корелацију између података, како би се на лакши начин уочили потенцијални проблеми.

5. ЛИТЕРАТУРА

[1] A. Chuvakin, K. Schmidt and C. Phillips, “Logging and log management: the authoritative guide to understanding the concepts surrounding logging and log management”, 2012.

[2] J. Babbin, “Security log management: identifying patterns in the chaos”, 2006.

[3] P. Shukla, S. Kumar, “Learning Elastic Stack 7.0: Distributed search, analytics, and visualization using Elasticsearch, Logstash, Beats, and Kibana”, 2019.

[4] M. J. Kavis, “Architecting the cloud: design decisions for cloud computing service”, 2014.

Кратка биографија:

Милан Маринковић рођен је у Бањој Луци 1998. год. Мастер студије на Факултету техничких наука из области Електротехнике и рачунарства – Софтверско инжењерство и информационе технологије уписао је 2021. год. контакт: milan_marinkovic98@hotmail.com

Горан Савић је ванредни професор на Факултету техничких наука Универзитета у Новом Саду. На истом факултету завршио је мастер студије 2006. године и докторирао 2011. године из области рачунарских наука и информатике. Ужа област интересовања су му електронска настава и пословни информациони системи.