Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.738.1:766
DOI: https://doi.org/10.24867/19BE04Stanarevic

UPOTREBA ELK STEKA TEHNOLOGIJA ZA OBRADU I ANALIZU SISTEMSKIH LOGOVA

USING ELK TECHNOLOGY STACK TO PROCESS AND ANALYZE SYSTEM LOGS

Mihailo Stanarević, *Fakultet tehničkih nauka, Novi Sad*

Oblast – PRIMENJENE RAČUNARSKE NAUKE I INFORMATIKA

Kratak sadržaj – *U radu je predstavljen sistem za parsiranje, obradu i vizualizaciju sistemskih logova. Sistem preuzima logove klijentskih aplikacija, parsira ih i strukturira pomoću Logstash-a, čuva ih u Elasticsearch-u i omogućava njihov prikaz preko Kibana korisničkog interfejsa. Klijentske aplikacije su sistemi koji se „pretplaćuju" na sistem za obradu logova kako bi koristili usluge istog. Pored same obrade logova implementirane su i funkcionalnosti čuvanja sistemskih logova na Amazonov S3 (Simple Storage Service) baket, kao i skidanje istih sa S3.*

Ključne reči: *Sistemski logovi, Elasticsearch, Logstash, Kibana, Amazon*

Abstract – *This paper describes and presents a system for parsing, processing and visualisation of system logs. This system receives logs from client applications, parses and structures them using Logstash, persists them in Elasticsearch and visualises them using the Kibana user interface. Client applications are systems that „subscribe" to this system to be able to use its functionalities. Along with the before mentioned functionality this system is able to provide client applications the possibility to upload and download their system logs to/from a system dedicated S3 bucket hosted on Amazon.*

Keywords: *System logs, Elasticsearch, Logstash, Kibana, Amazon*

1. UVOD

S obzirom da se sistemi danas razvijaju velikom brzinom, njihovi obimi se povećavaju itd. povećava se i broj procesa koji se dešavaju unutar njih a koji su veoma bitni po onoga čiji je sistem. Statistike, ishodi i druge metrike procesa od značaja se vrlo često zapisuju kako bi se vodila određena evidencija, nadgledalo stanje sistema u realnom vremenu i reagovalo na potencijalne, nepredviđene probleme u trenutku kada se oni dešavaju.

Ovi sistemski podaci obično se zapisuju u fajlovima koji se zovu log fajlovi. Nedostatak proces loginga i nadgledanja sistema nalazi se u OWASP Top 10 listi potencijalnih rizika zašto određeni sistemi bivaju kompromitovani a neretko i osetljivi podaci unutar njih. Zbog toga sistemi moraju implementirati određena rešenja kako bi vodili evidenciju stvari koje se dešavaju unutar njih kao i logiku koja će se primeniti kada se prepozna neželjena situacija u kojoj se sistem našao. Takozvana third party rešenja su vrlo često način na koji se ovi problemi izbegnu. Tačno to je tema ovog rada. Ovaj rad će predstaviti rešenje koje, upotrebnom ELK stack-a tehnologija (Elasticsearch-Logstash-Kibana), će sistem uz koji se integriše, učiniti sigurnijim kao što će i razvijačima sistema proces analitike, statistike, pretrage učiniti lakšim.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Dragan Ivanović, red. prof.

2. TEORIJSKE OSNOVE TEHNOLOGIJA ELK STEKA

Kako bi najbolje razumeli kako funkcioniše sistem potrebno je detaljno opisati uloge koje su imale tehnologije ELK steka u realizaciji ovog rešenja. Naravno pored opisivanja uloga u ovom odeljku će biti objašnjen način funkcionisanja ovih tehnologija.

2.1. Elasticsearch

Elasticsearch je distribuiran softver za pretragu i analitiku, otvorenog koda, koji je zasnovan na Apache Lucene biblioteci i razvijen u Java programskom jeziku. Elasticsearch je počeo kao skalabilna verzija Lucene radnog okvira za pretragu podataka, koji je takođe otvorenog koda, međutim pomoću njega je razvijena mogućnost da se horizontalno skaliraju Lucene indeksi. Elasticsearch omogućava korisniku da sačuva, pretraži i analizira velike količine podataka veoma brzo i u skoro realnom vremenu vrati (reda milisekundi) odgovore na odgovarajući upit [1].

Dokumenti su osnovne jedinice informacija koji se indeksiraju u Elasticsearch u JSON formatu, format koji je svetski korišćen kao format za međurazmenu podataka. Jedan dokument se može posmatrati kao jedna torka ili jedan red/vrsta u relacionim bazama podataka. Oni predstavljaju jedan određeni entitet tj. ono što se pretražuje. U Elasticsearch-u dokument može predstavljati mnogo više nego samo skup teksta. Može se strukturirati na bilo koji način takav da se može enkapsulirati u JSON format. Ti podaci mogu biti brojevi, karakteri, datumi, objekti koji predstavljaju geo lokacije, boolean vrednosti itd. Svaki dokument poseduje svoj jedinstveni ID kao i svoj format podataka tj. indikator koji ukazuje na to koji entitet taj dokument zapravo predstavlja.

Indeks u Elasticsearch-u predstavlja kolekciju dokumenata koji imaju slične karakteristike. On je entitet najvećeg nivoa apstraktnosti koji je moguće pretražiti u Elasticsearch-u. Indeks se može posmatrati kao jedna tabela u relacionim bazama podataka. Indeks se identifikuje imenom koje se koristi kada se referencira indeks pri operacijama kao što su indeksiranje, pretraga, ažuriranje ili brisanje dokumenata unutar njega.

Invertovani indeks predstavlja mehanizam koji većina današnjih pretraživača koriste. To je struktura podataka koja čuva i mapira sadržaj, kao što su reči ili brojevi, na njihove lokacije u dokumentu ili u setu dokumenata. Prosto rečeno, to je jedna velika struktura poput heš mape koja direktno dovodi od pojedine reči do dokumenta u kojem se pojavljuje. Invertovani indeks zapravo ne čuva stringove direktno već umesto toga razdvaja dokumente na individualne term-ove (reči koje se pojavljuju u dokumentu a mogu biti predmet pretrage) i tada mapira svaki term na dokument unutar kojeg se taj term pojavio.

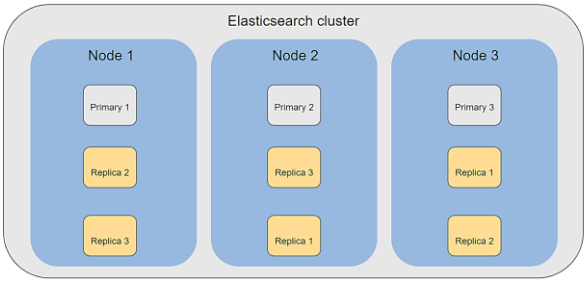
Što se tiče same „fizičke" organizacije Elasticsearch-a kao i njegove mogućnosti za horizontalno i vertikalno skaliranje bitni su sledeći pojmovi: **Klaster (Cluster)**, **Čvor (Node)**, **Shard** i **Replika (Replica)**.

U Elasticsearch-u jedan **Klaster** predstavlja grupu od jednog ili više instanci **Node**-ova (biće reči kasnije o njima) koji su povezani međusobno. Snaga Elasticsearch klastera ogleda se u njegovom distribuiranju zadataka, pretrazi i indeksiranju, kroz sve Node-ove u celom klasteru.

Čvor u Elasticsearch-u predstavlja jedan od servera koji je deo klastera. Jedan čvor čuva podatke i učestvuje u indeksiranju i sposobnosti pretrage klastera unutar kojeg se nalazi. Jedan čvor u Elasticsearch-u može biti konfigurisan kao master čvor (odgovoran za sve operacije koje se unutar klastera dešavaju), kao data čvor (čuva podatke i obavlja odgovarajuće pretrage i agregacije ukoliko ima potrebe) i kao klijentski čvor (služi za omogućavanje komunikacije između klastera i master čvora i komunikaciju sa data čvorom kako bi se podaci mogli učiniti dostupnim).

Elasticsearch omogućava da se odgovarajući indeks podeli na više delova koji se nazivaju shard-ovi. Svaki **Shard** je sam za sebe potpuno funkcionalni i nezavnisni indeks koji se može čuvati na bilo kom čvoru unutar klastera.

Elasticsearch dozvoljava da se kreiraju jedna ili više kopija shard-ova sa odgovarajućim indeksom koji se nazivaju **replika** shard-ovi ili samo replike. U suštini, replika shard predstavlja kopiju primarnog shard-a. Svaki dokument u okviru jednog indeksa pripada primarnom shard-u. Replike omogućavaju redundantne kopije podataka kako bi štitile od hardverskih grešaka i time služile kao backup ukoliko primarni shard ne bude u funkciji.



Slika 2.1. *Odnos backend komponenti u Elasticsearch-u*

Kao što se vidi na slici 2.1. u okviru ovog Elasticsearch klastera postoje 3 čvora u kojem svaki poseduje svoj primarni shard i po 2 replike primarnih shard-ova susednih čvorova. Ovo rešenje je skalabilno i vertikalno i horizontalno. Ukoliko se desi potreba da se kapacitet klastera poveća moguće je dodati novi čvor u klaster i primarni shard u okviru njega (horizontalno skaliranje). Tada se postiže povećanje kapaciteta klastera kao i raspodela narednih indexiranih dokumenata indexa koji se nalaze na tim shard-ovima. Samim tim što je dodat novi shard i novi node postiže se jos veća paralelizacija prilikom pretrage određenih indexa.

Veći broj shard-ova pretražuje manji obim podataka u isto vreme što rezultuje boljim performansama pretrage. Vertikalnim skaliranjem (kreiranjem više instanci replika podataka) postižemo veću redundansu podataka ali i mogućnost restauracije podataka ukoliko node ili neki od primarnih shard-ova padne, kao i veću dostupnost klastera. Što je klaster vertikalno veći, to je mogućnost gubitka podataka manji.

2.2. Logstash

Logstash je centralna komponenta za protok podataka u Elastic Stack-u koji služi za sakupljanje, obogaćivanje i spajanje svih podataka nevezano za format podataka ili šemu podataka.

Logstash-ovo procesiranje u realnom vremenu je poprilično moćna karakteristika kada se poveže sa Elasticsearch-om za perzistenciju podataka i Kibanom za pregled podataka. Te tri komponente zajedno čine poznati i vrlo rasprostranjeni ELK stack. Logstash dinamički uzima, transformiše, pakuje i šalje podatke ma koliko oni bili kompleksnog formata. Logstash-ova konfiguracija pipeline-a ya procesiranje event-ova se sastoji iz 3 glavne komponente: Inputa, Filtera i Outputa [2].

Uloga **inputa** je da definiše ulazni izvor podataka i da organizuje ulazne podatke u vidu niza događaja (event-ova) koje će, kasnije u toku Logstash-ovog pipeline-a, procesuirati i filtrirati filter stadijum pipeline-a [3].

Pomoću **file input plugin**-a, obrađuju se event-ovi iz fajlova, obično tako što se prati sadržaj u istim, ili ukoliko je takvo rešenje potrebno, čitanjem sadržaja fajla od početka. Kako je generalno implementirano na nivou file input plugin-a, jedan događaj (event) se smatra kao jedna linija sadržaja (jednom linijom se smatra sadržaj dok se ne pojavi delimiter za novu liniju ("\n").

Kako podaci prolaze kroz Logstash-ov pipeline od izvora do medijuma gde će se sačuvati, Logstash **filteri plugin-i** parsiraju svaki event, determinišu imenovana polja kako bi kreirali određenu strukturu iz inače nestrukturiranih podataka [4].

Grok filter plugin parsira nestrukturirani tekst i strukturira ga na način takav da se može i izvršavati smisleni upiti nad njim. Ti logovi su obično vrlo često raznih struktura i formata jer se obično kreiraju za ljudske oči, a od velikog su informativnog značaja za dijagnostiku rada sistema. Logstash sadrži otprilike 120 različitih podrazumevanih šablona (dostupni su na sledećem linku: https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns).

Grok funkcioniše tako što kombinuje šablone teksta u nešto što odgovara logovima koji su mu predstavljeni. Sintaksa koja definiše ove šablone se sastoji iz sledećih komponenti u sledećem formatu: %{SYNTAX:SEMANTIC}. Syntax predstavlja ime šalbona koji će odgovarati tekstu koji parsira. Na primer

brojevi će odgovarati NUMBER sintaksi, IP adrese IP sintaksi itd. SEMANTIC, sa druge strane, predstavlja jedinstveni identifikator vrednosti teksta koji je prepoznat i kao takav se čuva na izabrani medijum koji se definiše kroz output deo Logstash-ovog pipeline-a. Sintaksa specificira način na koji će se prepoznati određeni podatak a semantik ga imenuje za lakšu prezentaciju, čuvanje kao i mogućnost izvršavanja upita nad istim.

Output deo Logstash-ovog pipeline-a nagoveštava gde naši parsirani i, sada, strukturirani treba da idu kako bi se sačuval. Iako je Elasticsearch najrasprostranjeniji i otvara put ka boljoj pretrazi i analitičkim mogućnostima, postoje mnoge druge mogućnosti kuda podaci mogu potencijalno da se rutiraju [5].

2.3. Kibana

Kibana je besplatna i otvorena za sve korisnike frontend aplikacija koja se nalazi na vrhu Elastic stack-a, omogućavajući pretragu i vizualizaciju podataka koji su indeksirani u Elasticsearch-u.

Opštije poznatija kao alat za razne grafike za Elastic stack, Kibana takođe predstavlja korisnički interfejs za nadzor, upravljanje i obezbeđivanje Elastic Stack klastera kao i centralizovani čvor za već kreirana rešenja razvijena na Elastic steku [6].

3. Specifikacija i arhitektura sistema

Sistem za obrađivanje logova je servis koji ima otvoren API za sve klijentske aplikacije koje se na njega „pretplate". Klijentske aplikacije koje se pretplate dobijaju svoj jedinstven *clientId* i *clientSecret* koje moraju da šalju pri svakom zahtevu. Ukoliko pošalju odgovarajuće informacije, klijentske aplikacije imaju mogućnost da obave sledeće akcije:

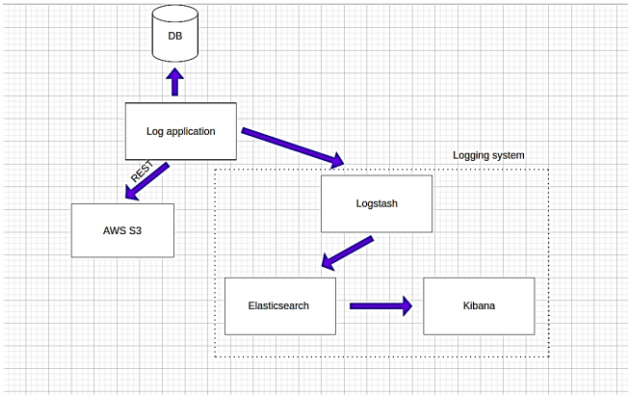
- Obrađivanje svojih logova i njihov prikaz na Kibana korisničkom interfejsu
- Otpremanje svojih logova i zipovanih logova na svoj posvećen medijum (S3) za čuvanje podataka te prirode
- Skidanje, na zahtev, fajlova koji su prethodno otpremljeni na S3

Arhitektura sistema se može videti na slici 3.1. Ona se neformalno sastoji iz 3 celine:

- Aplikacije koja ima otvoren API sa kojim klijentske aplikacije mogu komunicirati
- Cloud deo aplikacije kojeg predstavlja S3 baket sistema u kojem svaka pretplaćena aplikacija ima svoje mesto gde može da čuva svoje logove i zipovane verzije logova
- Pozadinski sistem za logovanje koji se sastoji iz ELK steka tehnologija koji obavljaju glavnu logiku parsiranja, strukturiranja i čuvanja strukturiranih logova

4. IMPLEMENTACIJA SISTEMA

Celokupan sistem je implementiran pomoću već pomenutih tehnologija sa već utvrđenom arhitekturom. U ovom poglavlju ćemo, međutim, proći detaljno kroz izvorni kod (dostupan na linku: https://github.com/mihailostanarevic/master-rad) i implementaciju svih navedenih funkcionalnosti u 3. poglavlju.



Slika 3.1. *Arhitektura sistema za obradu logova*

4.1. Konfiguracija

Spring Boot kada pokreće aplikaciju on prvobitno skenira sve klase i sve klase koje su adekvatno anotirane instancira i obavlja potrebna uvezivanja zavisnosti. Takođe preuzima konfiguraciju koja mu se nalazi u application.properties fajlu kako bi ih iskoristio ukoliko su u kodu korišćene ili su potrebne za instanciranje određenih klasa. Unutar ovog fajla definisane su vrednosti poput porta na kom server radi, putanja konteksta servleta, podešavanja potrebna za konekciju sa bazom podataka itd.

Za inicijalno punjenje aplikacije sa podacima korišćen je data.sql fajl koji se nalazi u resources paketu projekta. Data.sql se, s obzirom da je u application.properties fajlu definisano da se na podizanje aplikacije rekreira baza, učitava i komande koje se nalaze unutar njega se izvršavaju direktno u shell-u Postgre-a.

Unutar ContextConfig.java klase definisani su svi third party bean-ovi koji su potrebni da se instanciraju. Neki od njih su RestTemplate koji će kasnije služiti za komunikaciju našeg sistema sa eksternim servisima kao i PasswordEncoder bean koji koristi *BcryptPasswordEncoder* kako bi heširao lozinke da bi se takve čuvale u bazi.

4.2. Implementacija funkcionalnosti

Bez registracije u okviru sistema za obradu logova, aplikacije neće biti u mogućnosti da obavljaju preostale tri funkcionalnosti. S obzirom da celokupan API sistema ima relativno mali broj funkcionalnosti u nastavku će biti opisana implementacija svih funkcionalnosti zasebno.

4.2.1. Registracija aplikacija

Registracija aplikacija na sistem za obradu logova je najbitnija funkcionalnost u sistemu s obzirom da je on preduslov za mogućnost korišćenja ostatka API-ja sistema. Sam endpoint se nalazi na putanji */api/register* sa POST metodom. Da bi se aplikacija uspešno registrovala potrebno je da pošalje naziv svoje aplikacije koja mora biti jedinstvena u okviru celog sistema

U *RegistrationService* klasi potrebno je generisati odgovarajuće kredencijale za aplikaciju koja se registruje. ClientId se generiše tako što se izgeneriše novi UUID (Universal Unique Identifier) i razdeli se po „-" delimiteru. Na već postojeći string „APP-" nadodaje se prvi deo UUID-a i tako nastaje clientId.

Što se tiče clientSecret-a potrebna je komunikacija sa otvorenim API-jem koji se nalazi na https://www.passwordrandom.com/. On generiše string fiksne dužine koji se sastoji iz slučajno izabranih karaktera. Tada, ukoliko je ime aplikacije jedinstveno, obavlja se heširanje clientSecret-a i čuvanje aplikacije u bazu. Aplikacija tad dobija svoje kredencijale nazad u HTTP odgovoru kako bi ih koristila za dalje zahteve.

4.2.2. Čuvanje i skidanje logova sa S3

Funkcionalnosti vezane za S3, iako nisu toliko bitne za ovaj rad, predstavljaju funkcionalnosti koje su dodatne pogodnosti za pretplaćene aplikacije pomoću kojih mogu da čuvaju i nakon toga skidaju svoje fajlove sa posvećenog sistemskog S3 baketa. Da bismo mogli integrisati naš sistem sa AWS-ovim servisom S3 potrebno je uvesti AWS-ov SDK (*Software Development Kit*) tako što će se ubaciti odgovarajući dependency u pom.xml fajl aplikacije.

Funkcionalnost za otpremanje fajlova je definisana POST metodom na */storage/upload* URL-u. Metoda prima niz fajlova koje korisnik želi da sačuva na S3 a vraća indikator uspešnosti/neuspešnosti procesa. Od klijentske aplikacije se traži da pošalje svoje kredencijale u okviru zaglavlja zahteva kako bi se predstavio sistemu. Nakon što se obavi potrebna provera kredencijala poziva se metoda uploadFile za svaki fajl koji korisnik želi da sačuva. Tada klijentska aplikacija, kao odgovor, dobija poruku o ishodu procesa čuvanja fajla.

Skidanje fajla sa S3 je dostupno na */storage/download* URL-u. Na klijentskoj aplikaciji je da pošalje svoje kredencijale i naziv fajla koji želi da skine. Tada servis generiše takozvani PreSigned URL koji je jedinstven i traje određeni vremenski period (koliko developer odluči). Klijentska aplikacija dobija nazad taj PreSigned URL koji može da poseti i u zavisnosti od tipa fajla će ili započeti skidanje fajla ili će probati da prikaže u browser-u.

4.2.3. Obrada i prikaz logova

Funkcionalnost obrade i prikaza logova dostupna je na endpoint-u */logs/index*. U okviru zahteva potrebno je da klijentska aplikacija pošalje svoje log fajlove i kredencijale. Prvenstveno se vrši autentifikacija aplikacije na serveru.

Nakon uspešne autentifikacije kopira se sav sadržaj log fajlova na fiksnu lokaciju na serveru. Logstash za to vreme konstantno sluša na promene na toj istoj lokaciji i, kada do nje dođe, on pokušava da iskoristi svoj pipeline da parsira i strukturira (pomoću Grok-a) podatke koji se nalaze u tom log fajlu. Nakon što strukturira podatke, zapakuje ih u event-ove i šalje u svoj konfigurisani output plugin za Elasticsearch, gde se strukturirani dokumenti indeksiraju. Nakon što se podaci indeksiraju, klijentska aplikacija dobija informaciju da su uspešno indeksirani logovi i da su dostupni na linku koji vodi do Kibana Discover screen-a. Tada je klijentska aplikacija u mogućnosti da pregleda svoje logove, da obavlja kompleksnije pretrage i analize nad njima i još mnogo toga.

5. ZAKLJUČAK

U ovom radu opisano je rešenje koje će potencijalno mnogobrojnim klijentskim aplikacijama omogućiti napredno praćenje, pretragu, analitiku svojih sistemskih logova. Takođe korisnici će sa ovim rešenjem biti u mogućnosti da se rešavaju svojih zipovanih fajlova sa sistemskim logovima sa svojih mašina na posvećeni *cloud* prostor u okviru sistema koji je tema ovog rada.

6. LITERATURA

[1] Elasticsearch https://www.knowi.com/blog/what-is-elastic-search/
[2] Logstash https://www.elastic.co/logstash/
[3] Logstash Input Plugins https://www.elastic.co/guide/en/logstash/current/input-plugins.html
[4] Logtash Filter Plugins https://www.elastic.co/guide/en/logstash/current/filter-plugins.html
[5] Logstash Output Plugins https://www.elastic.co/guide/en/logstash/current/output-plugins.html
[6] Kibana https://www.elastic.co/what-is/kibana

Kratka biografija:

Mihailo Stanarević rođen je 12.05.1997. godine u Novom Sadu. Godine 2016. upisao je Fakultet Tehničkih Nauka u Novom Sadu, odsek Računarstvo i automatika. 2018. godine upisuje usmerenje Primenjene računarske nauke i informatika, a 2019. godine se opredeljuje za modul Internet i elektronsko poslovanje. U septembru 2020. godine je diplomirao i upisao Master akademske studije.

Kontakt: stanarevic.mihailo10@gmail.com