TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Ceisi Peik, 155177

LOGIDE HALDAMISE SÜSTEEMI SEADISTAMINE TALLINNA TEHNIKAÜLIKOOLI NÄITEL

Bakalaureusetöö

Juhendaja: Ago Luberg,
Tarkvarateaduse instituut, MSc

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Ceisi Peik

19.05.2019

Annotatsioon

Töös käsitletakse kolme tüüpi logisid: Moodle'i poolt andmebaasi tabelisse genereeritavad logid, GitLabi poolt JSON formaadis faili genereeritavad logid ja Testeri poolt tekstifaili genereeritavad Java logiformaadis logid. Põgus ülevaade on tehtud kahest logide haldamise süsteemist ja põhjendatud kumma kasuks antud töös otsustati.

Töö raames seadistatakse logide haldamise süsteem, mille keskseks osaks on Graylogi rakendus. Logide saatmiseks algsest asukohast Graylogi kasutatakse Logstashi ja Filebeati. Töös kirjeldatakse süsteemi üldist seadistust ja eelnimetatud formaadis logide lugemiseks ning Graylogi saatmiseks vajalikku konfiguratsiooni. Lisaks konfigureeritakse logide visualiseerimise võimaluste demonstreerimiseks konkreetsetele logi väärtustele vastavad sektordiagrammid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 32 leheküljel, 8 peatükki, 27 joonist, 0 tabelit.

Abstract

Configuration of Log Management System for Tallinn University of Technology

The purpose of this thesis is to create a log management system that can be used with three different type of logs: logs that are stored in database table, logs that are stored in JSON formatted file and logs that are in Java format and stored in a text file.

Thesis includes a little overview of two log management systems: Graylog and ELK Stack, alongside with the explanation of which one was chosen for the thesis. Logs generated by Moodle, GitLab and Tester are used for setting up and testing the configured log management system.

Central application of the configured log management system is Graylog. Graylog uses MongoDB and ElasticSearch for its work, but minimum configuration is required regarding those two applications. Logstash needs to be configured for reading the logs from database table, modifying them accordingly and shipping them over to Graylog. Two Filebeat instances need to be configured, one for reading the logs from JSON formatted file and shipping them to Graylog, other for reading logs in Java log format from text file and shipping them to Graylog. In Graylog three inputs need to be configured, each for different log type. Input configuration for JSON and TXT logs include appropriate data extractor: Graylog's JSON extractor for the first one and grok extractor for the latter. To demonstrate visual representation of log data in Graylog, three different pie charts are configured.

The thesis is in Estonian and contains 32 pages of text, 8 chapters, 27 figures, 0 tables.

Lühendite ja mõistete sõnastik

Moodle	Avatud lähtekoodiga õpingute haldamise süsteem
GitLab	Avatud lähtekoodiga ja sisseehitatud versioonihaldusega tarkvara arendamise platvorm
JSON	*JavaScript Object Notation*, andmete edastamiseks loodud tekstiformaat
Tester	Rakendus programmikoodi valideerimiseks
Bitnami Moodle Stack	Pakett Moodle'i lihtsaks installeerimiseks
MariaDB	Avatud lähtekoodiga relatsiooniline andmebaas, MySQL-i haru
Docker	Operatsioonisüsteemi tasemel virtualiseerimist võimaldav programm
Grep	Käsurea programm tekstist regulaaravaldise vastete leidmiseks
GitHub	Veebipõhine versioonihaldusrakendus
bug	Rakenduses esinevad tootevead
ELK Stack	Järgmiste rakenduste akronüüm: ElasticSearch, Logstash, Kibana
Graylog	Logide salvestamist ja reaalajas analüüsi võimaldav logide haldamise rakendus
Lucene	Terve teksti indekseerimist võimaldav teksti otsingu mootor
ElasticSearch	ELK Stacki kuuluv Lucene'i teegil põhinev JSON dokumentidest koosnev analüüsi- ja otsingumootor
Logstash	ELK Stacki kuuluv reaalajas andmete kogumist võimaldav rakendus
Beats	ELK Stacki kuuluv spetsiaalsete logide saatmist võimaldavate rakenduste grupp
Filebeat	Avatud lähtekoodiga programm logide edastamiseks
Kibana	ELK Stacki kuuluv andmete visualiseerimise rakendus
Vega	Grammatika interaktiivsete visuaalsete graafikate loomiseks
Syslog	Standardne logisõnumi formaat logide saatmiseks erinevatest seadmetest kesksesse logiserverisse
CSV	Failiformaat, milles väärtused on komadega eraldatud
endpoint	Võrgu lõppsõlm teatud protsessi või programmi poole pöördumiseks

Nginx	Peamiselt veebiserverina kasutatav vabavaraline tarkvara
Nmap	*Network Mapper*, võrgu kaardistamiseks kasutatav rakendus
MongoDB	Avatud lähtekoodiga mitte-relatsiooniline andmebaas
JDBC	*Java Database Connectivity*, Java programmiliides andmebaasiga ühendumiseks
Docker Compose	Tööriist mitmetest konteineritest koosnevate Dockeri rakenduste defineerimiseks ja käivitamiseks
HTTP	*Hypertext Transfer Protocol*, protokoll arvutivõrkudes andmete edastamiseks
X-Pack	ELK Stacki laiendus, mis hõlmab turvalisuse, teavitamise, monitooringu ja raporteerimise võimalusi
TCP	*Transmission Control Protocol*, transpordikihi võrguprotokoll, mis garanteerib, et saadetud sõnum jõuab kohale
UDP	*User Datagram Protocol*, transpordikihi võrguprotokoll, mis ei garanteeri, et kõik sõnumid jõuavad kohale
GELF	*Graylog Extended Log Format*, JSON sõnest koosnev logiformaat rakenduste poolt genereeritud logide Graylogi saatmiseks
SHA2	*Secure Hash Algorithm 2*, krüptograafiliste räsi funktsioonide jada
URI	*Uniform Resource Identifier*, tähemärkide jada, mis tähistab teatud ressurssi
Dockerfile	Fail, mis koosneb Dockeri tõmmise ehitamiseks vajalikest juhistest
JAR	*Java Archive*, üks kokku pakitud fail, mis koosneb agregeeritud Java klassi failidest, metaandmetest ja muudest ressursifailidest
JDBC	*Java Database Connectivity*, Java programmeerimiskeele programmiliides andmebaasiga suhtlemiseks
Cron	Ajal põhinev planeerija automaatsete tööde käivitamiseks
TLS	*Transport Layer Security*, transpordikihi turbeprotokoll saadetavate andmete krüpteerimiseks
Grok	Regulaaravaldistest koosnev lause avalistele vastavatele sõnedel nimede andmiseks
Ping	Arvutivõrgu võrguühenduse diagnostikaprogramm

Sisukord

Jooniste loetelu

Sissejuhatus

Lõputöö raames seadistatakse logide haldamise süsteem Tallinna Tehnikaülikoolis kasutatava kolme rakenduse näitel: ained.ttu.ee ehk edaspidi Moodle, GitLab ja Tester. Moodle on veebipõhine õpihaldussüsteem, GitLab on sisseehitatud versioonihaldusega tarkvara arendamise platvorm ning Testerit kasutatakse tudengite loodud programmikoodi valideerimiseks. Eelnimetatud rakendused genereerivad erinevat tüüpi logisid, seega lõputöös näidatakse kuidas seadistada logide haldamise süsteemi, mis saab logisid nii andmebaasi tabelist, JSON formaadis failist kui ka tekstifailist.

Töö eesmärgiks on seadistada logide haldamise süsteem, mille keskseks rakenduseks on Graylog. Andmebaasi tabelist logide lugemiseks ja edasi Graylogi saatmiseks konfigureeritakse Logstashi rakendus. JSON ja TXT formaadis failidest logide lugemiseks ja edasi Graylogi saatmiseks konfigureeritakse kaks eraldi Filebeati instantsi. Logide vastuvõtmiseks konfigureeritakse Graylogi poolsed sisendid, mis lisaks tükeldavad saadetud logisõnumid paremini analüüsitavateks väiksemateks osadeks. Demonstreerimaks logide haldamise süsteemi otsingut ja visuaalseid võimalusi konfigureeritakse kolm erinevat andmeid kokkuvõtvat sektordiagrammi.

Vajadus logide haldamise süsteemi seadistamiseks tuleneb asjaolust, et erinevad rakendused genereerivad oma logid erinevatesse kohtadesse, keerulisemaks muudab asjaolu ka see, kui logid salvestatakse erinevates formaatides. Sellisel juhul on üpriski ebamugav logide hulgast vajalikku informatsiooni leida rääkimata agregeeritud ülevaate saamisest.

Töös tutvustatakse kahte logide haldamise süsteemi ja põhjendatakse, mille kasuks antud töö puhul otsustati. Ülevaade antakse Moodle'i, Gitalbi ja Testeri poolt genereeritavate logide struktuurist ja lõputöös algandmetena kasutatavatest logidest. Põhjalikult on kirjeldatud logide haldamise süsteemi seadistust ja süsteemi komponentide konfiguratsiooni.

1 Käsitletavate rakenduste logimise hetkeolukord

Antud töös käsitletavad Tallinna Tehnikaülikoolis kasutatavad rakendused logivad kõik erineval viisil. Moodle'i kasutamise info logitakse andmebaasi tabelisse, GitLabi logid talletatakse JSON formaadis tekstifaili ja Testeri logid salvestatakse samuti tekstifaili.

1.1 Logide struktuur

Järgnevates peatükkides kirjeldatakse täpsemalt käsitletavate rakenduste logide struktuuri, mis on tulenevalt logimise viisist erinev.

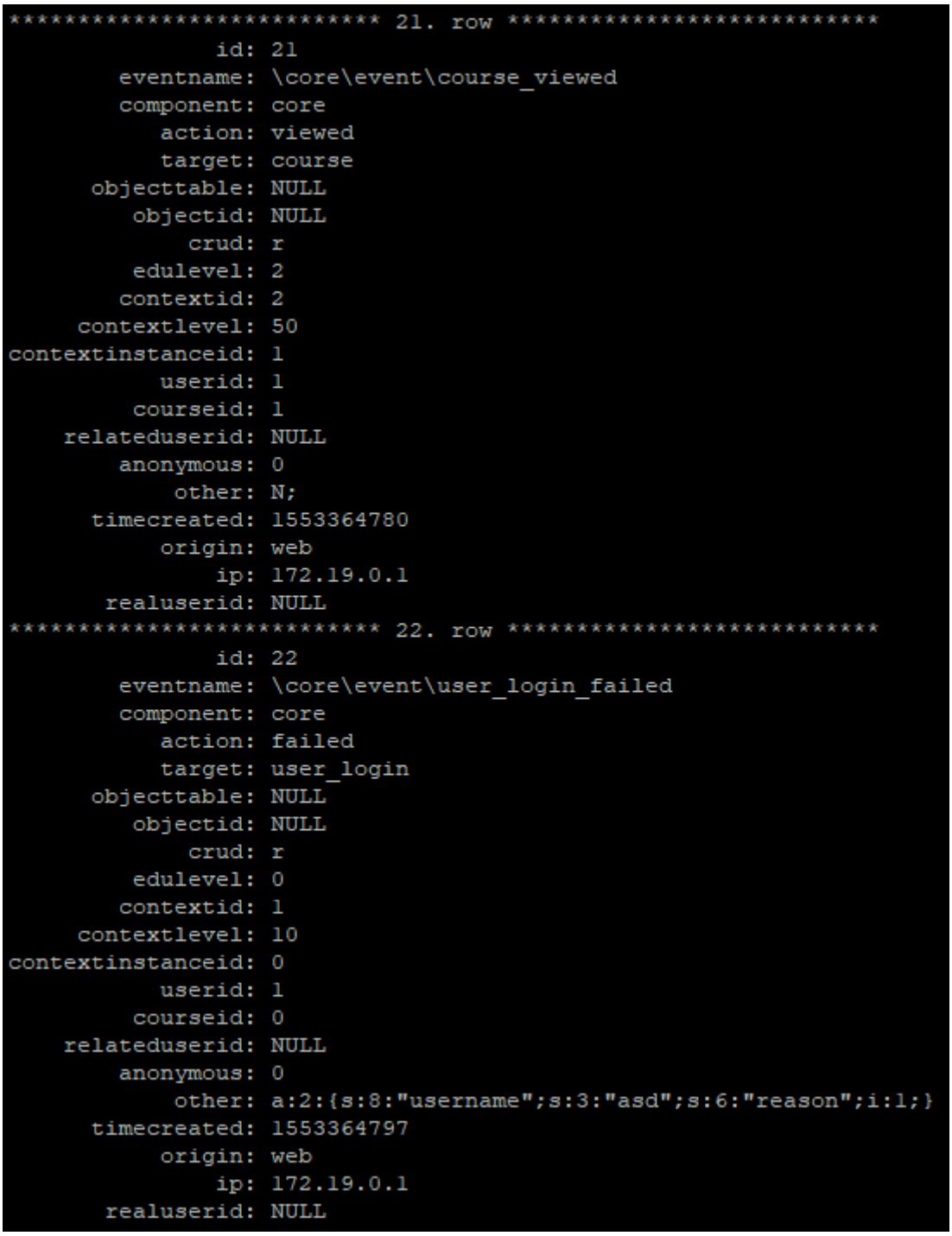
1.1.1 Moodle'i logid

Bitnami Moodle Stacki puhul salvestatakse Moodle'i poolt genereeritud logid MariaDB andmebaasis olevasse `mdl_logstore_standard_log` nimelisse tabelisse. Antud töö puhul jookseb MariaDB andmebaas Docker konteineris. Seega on andmebaasis olevaid logisid võimalik vaadata järgnevalt: tuleb minna konteinerisse, avada seal andmebaas ja teha päring logisid hoidva tabeli pihta. Eelnimetatud tegevusteks vajalikud käsud on välja toodud joonisel 1.

```
$ docker exec -it mariadb bash
$ mysql --user=bn_moodle --password=bitnami_moodle
$ select * from mdl_logstore_standard_log\G
```

Joonis 1. Käsud MariaDB-s olevate Moodle'i logide kuvamiseks.

Joonisel 2 on eelnevalt kirjeldatud käskude kaudu saadud väljavõte Moodle'i logidest.



Joonis 2. Väljavõte Moodle'i logitabelist.

Moodle'i logitabelis on 21 andmevälja, täpsem ülevaade tabeli väljadest, nende andmetüüpidest ja kohustuslikkusest on joonisel 3.

```
MariaDB [bitnami_moodle]> DESCRIBE mdl_logstore_standard_log
    -> ;
+-------------------+-------------+------+-----+---------+----------------+
| Field             | Type        | Null | Key | Default | Extra          |
+-------------------+-------------+------+-----+---------+----------------+
| id                | bigint(10)  | NO   | PRI | NULL    | auto_increment |
| eventname         | varchar(255)| NO   |     |         |                |
| component         | varchar(100)| NO   |     |         |                |
| action            | varchar(100)| NO   |     |         |                |
| target            | varchar(100)| NO   |     |         |                |
| objecttable       | varchar(50) | YES  |     | NULL    |                |
| objectid          | bigint(10)  | YES  |     | NULL    |                |
| crud              | varchar(1)  | NO   |     |         |                |
| edulevel          | tinyint(1)  | NO   |     | NULL    |                |
| contextid         | bigint(10)  | NO   | MUL | NULL    |                |
| contextlevel      | bigint(10)  | NO   |     | NULL    |                |
| contextinstanceid | bigint(10)  | NO   |     | NULL    |                |
| userid            | bigint(10)  | NO   | MUL | NULL    |                |
| courseid          | bigint(10)  | YES  | MUL | NULL    |                |
| relateduserid     | bigint(10)  | YES  |     | NULL    |                |
| anonymous         | tinyint(1)  | NO   |     | 0       |                |
| other             | longtext    | YES  |     | NULL    |                |
| timecreated       | bigint(10)  | NO   | MUL | NULL    |                |
| origin            | varchar(10) | YES  |     | NULL    |                |
| ip                | varchar(45) | YES  |     | NULL    |                |
| realuserid        | bigint(10)  | YES  |     | NULL    |                |
+-------------------+-------------+------+-----+---------+----------------+
21 rows in set (0.01 sec)
```

Joonis 3. Moodle'i logitabeli väljad.

1.1.2 GitLabi logid

GitLabi poolt genereeritavad logid salvestatakse JSON formaadis tekstifaili, mida roteeritakse igal ööl. Üks logikirje objekt paikneb tervenisti ühel real [1]. Joonisel 4 on välja toodud ühel real paiknev logi objekt, mis on vormindatud kergesti loetavale kujule.

```
{
    "method": "GET",
    "path": "/akuusm/planetsystem.git/info/refs",
    "format": "*/*",
    "controller": "Projects::GitHttpController",
    "action": "info_refs",
    "status": 401,
    "duration": 19.82,
    "view": 0.64,
    "db": 2.76,
    "time": "2019-01-16T00:58:31.120Z",
    "params": [
        {
            "key": "service",
            "value": "git-upload-pack"
        },
        {
            "key": "namespace_id",
            "value": "akuusm"
        },
        {
            "key": "project_id",
            "value": "planetsystem.git"
        }
    ],
    "remote_ip": "80.235.15.134",
    "user_id": null,
    "username": null
}
```


Joonis 4. Vormindatud GitLabi logi objekt.

1.1.3 Testeri logid

Testeri puhul salvestatakse Java rakenduse poolt genereeritud logid tekstifaili, mida roteeritakse igal ööl. Tekstifailis on nii standardseid logikirjeid, Java programmikoodi kui ka tavalist teksti. Logikirjetel on kellaaeg, kuid puudub kuupäev, see kajastub aga logifaili nimes. Kuna genereeritavad logid on värvilised kajastub see tekstifailis värvikoodidena. Joonisel 5 on väljavõte Testeri logifailist, mis kujutab ühel real paiknevat Java logikirjet.

```
14:58:53 [33mDEBUG[0;39m [35m[       961065737][0;39m
e.t.c.a.s.p.BaseSubmissionProcessor - Tests:
GitTestResource(repositoryUrl=git@gitlab.cs.ttu.ee:iti0202-2019/ex.git,
commits=[GitCommit(hash=4555104cdd8f522496d417c7d8c3c1cac2e64b83,
message=Update modernbakery.rst, timestamp=1556617672, added=null,
modified=[PR14/modernbakery.rst], removed=null)],
repository=ee.ttu.cs.arete.data.resource.git.TemporaryGitRepository@3a0d433b)
```

Joonis 5. Logikirje Testeri logifailist.

1.2 Logimise problemaatika

Vajadusel on andmebaasist logisid üpriski kerge pärida ja otsitava informatsiooni leiab suhteliselt kiiresti üles. Tekstifaili talletatavate logidega see nii lihtne pole. Üks võimalus teksti otsimiseks on kasutada `grep` käsurea käsku, kuid see on juba tunduvalt ebamugavam ja nõuab omajagu aega. Veel tülikamaks muudab logide hulgast vajaliku informatsiooni otsimise asjaolu, et nii Moodle'i, GitLabi kui ka Testeri logid on erineva struktuuri ja erinevate väljadega. Eelneval põhjusel puudub võimalus üldise monitooringu teostamiseks üle kõigi kolme rakenduse.

2 Logide haldamise tarkvara valik

Logide haldamise tarkvara valikul on lähtutud eelkõige sellest, et tarkvara oleks tasuta ja hästi dokumenteeritud. Olulisel kohal on kogukonna toetus toote foorumi aktiivsuse näol ja avatud lähtekoodiga toote puhul lisaks ka GitHubis hallatavate *bug*-de näol. Eelnimetatud omadustele toetudes jäid valitute hulka ELK Stack ja Graylog.

2.1 ELK Stack

ELK Stacki hulka kuuluvad mitmed tooted, millest igal on erinev tööülesanne. ElasticSearch on laias laastus kui andmebaas, kus indekseeritakse ja salvestatakse kokku kogutud logid. Logstashi kasutatakse logide kokku korjamiseks, struktuuri ja sisu muutmiseks ning edasi saatmiseks järgmisele logide haldamise rakendusele. Beats hõlmab endas erinevaid minimaalseid rakendusi, mis on mõeldud andmete edastamiseks otspunktidest kesksesse haldussüsteemi. Kibanat kasutatakse kokku kogutud andmete visualiseerimiseks [2].

ELK Stacki puhul on plussiks Kibana visualiseerimise võimalused. Kibana võimaldab andmete põhjal luua astmikdiagramme, joondiagramme, sektordiagramme, tabeleid, suhtediagramme ja ajast sõltuvaid graafikuid. Lisaks võimaldab Kibana kasutada Vega visualiseerimiskeeles loodud graafikuid [3].

Teiseks suureks plussiks on Logstash, mis võimaldab andmeid koguda väga erinevatest allikatest alustades Syslog sõnumitega ja lõpetades andmebaasi tabelitega [4]. Sarnaselt on Logstashist võimalik andmeid saata väga erinevatesse sihtpunktidesse, olgu selleks CSV formaadis fail, HTTP *endpoint* või meiliaadress [5].

Tuginedes artiklitest ja foorumitest saadavale infole on ELK Stacki üpriski keeruline seadistada, suur osa keerukusest seisneb erinevate osade omavahelisel suhtlusel ning veel keerulisemaks läheb siis kui andmemahud kasvavad [6]. ELK Stacki puhul jääb mõnevõrra soovida ka turvalisusest, puudub sisseehitatud autentimine. Üks lihtsamaid viise selle realiseerimiseks on kasutades Nginx pöördproksit [7].

2.2 Graylog

Graylog on logide haldamiseks mõeldud rakendus, kuid ainult Graylogist endast ei piisa. Graylog kasutab sarnaselt ELK Stackile ElasticSearchi logide indekseerimiseks ja talletamiseks. Lisaks on Graylogi tööks vajalik MongoDB, sinna salvestatakse konfiguratsioon ja muud metaandmed. Graylog ise on vajalik saadetud logide vastuvõtmiseks ja ElasticSearchi paigutamiseks, saadud andmete analüüsimiseks ja filtreerimiseks. Lisaks on Graylogis võimalik luua kasutajaid ja ligipääsuõigusi ning seada üles teavitusi [8].

Üldiselt on Graylog-i lihtsam seadistada kui ELK Stacki, sest palju funktsionaalsust saab muuhulgas konfigureerida veebiliidese kaudu. Tihti kasutatakse Graylogi koos osade ELK Stacki toodetega, mistõttu võib seadistamise keerukus kasvada. Siiski lihtsustab Graylogi seadistamist ja kasutamist brauseris avatav kasutajaliides. Graafilises kasutajaliideses on näiteks võimalik seadistada erinevaid sisendeid logide vastuvõtmiseks [9], parsimisreegleid logide struktureerimiseks [10] ja filtreid logidest vajaliku info otsimiseks [11].

Suureks plussiks on sisseehitatud kasutajate ja rollide haldamise süsteem. Graylogis on võimalik luua erinevaid kasutajaid määrates neile sobivaid rolle ja sessiooni aja piiranguid [12]. Kasutajatele rakendatavaid kasutajaliidese ligipääsu rolle on samuti võimalik luua ja hallata veebiliidese kaudu [13].

Graylogi veebiliideses saab küll paljusid seadistusi lihtsamini konfigureerida, kuid andmete visualiseerimise võimalused on võrreldes Kibanaga kesisemad. Graylogis on võimalik luua tavapäraseid astmik-, sektor- ja joondiagramme, lisaks on võimalik leida ja kujutada väljade statistilisi väärtusi nagu maksimum ja standardhälve [14].

2.3 Lõplik valik

Toetudes eelkõige seadistamise keerukusele on logide haldamise süsteemiks valitud Graylog. Kuna antud lõputöö üheks eesmärgiks on võimalus logide hulgast lihtsasti vajalikku informatsiooni otsida, mitte niivõrd andmeid visualiseerida, sobib selleks Graylog väga hästi. Kui hiljem peaks tekkima vajadus andmeid visualiseerida rohkematel viisidel kui Graylog iseseisvalt võimaldab, saab Graylogile külge seadistada ka Kibana.

Antud lõputöö raames on logisid vaja lugeda andmebaasi tabelist, kuid Graylog seda ei võimalda. Andmete andmebaasist Graylogi saamiseks on vaja kasutada Logstashi. Logstashis on võimalik seadistada JDBC sisend, kuhu loetakse määratud ajavahemiku tagant tabelisse tekkinud uued kirjed, mis seejärel saadetakse GELF väljundit kasutades Graylogi.

Lisaks on tekstifailis talletatavate logide edastamiseks vaja kasutada Filebeati, mis paigutatakse kõikidesse masinatesse, kuhu logid algselt genereeritakse. Filebeati abil saab lugeda tekstifaili salvestatavaid logi kirjeid, olenemata nende formaadist ning seejärel edastada kirjed sobivasse sihtpunkti, antud juhul Graylogi.

3 Seadistamine

Järgnevates peatükkides on kirjeldatud lõputöö raames loodava logide haldamise süsteemi üldist seadistust.

3.1 Algandmed

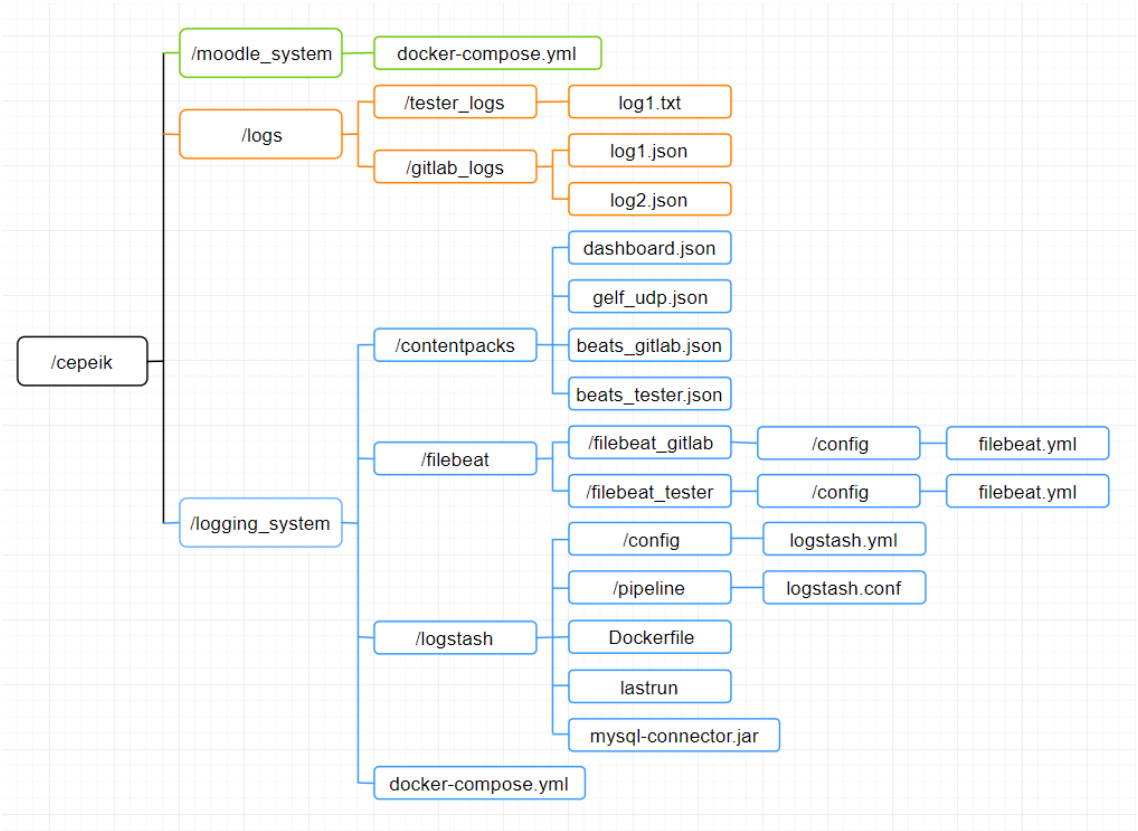
Algandmed on vajalikud logimissüsteemi seadistamiseks ja seadistuse valideerimiseks. Algandmetena on kasutatud logide väljavõtteid reaalselt kasutuses olevast süsteemist ja ka ise süsteemi kasutamise käigus tekitatud logisid.

Moodle'i logide saamiseks on kasutatud Bitnami Moodle Stacki. Stacki on minimaalse seadistusega võimalik Docker Compose'iga käivitada ja kasutada. Avades brauseris Moodle'i veebirakenduse tekitatakse kasutaja tegevuste käigus Moodle'i poolt kasutatavasse MariaDB andmebaasi, `mdl_logstore_standard_log` tabelisse, sellekohased logi kirjeid.

JSON formaadis GitLabi ja TXT formaadis Testeri tekstifailis talletatavate logide lugemise ja saatmise seadistuse kontrollimiseks on kasutatud staatilisi logifaile, milles on väljavõtted reaalse süsteemi logidest. Antud lõputöö raames Testeri logi väljavõtteid mõnevõrra kohandatud. Kuna Testeri logifailides on väga erinevas formaadis kirjeid, alustades mitmel real paiknevast programmikoodist ning lõpetades ühel real paikneva Java logiga, on lõputöös kasutatud vaid ühel real paiknevaid Java logikirjeid.

3.2 Keskkond

Lõputöö raames seadistatud logide haldamise süsteem on püsti pandud Tallinna Tehnikaülikooli serverisse lõputöö koostaja alamkataloogi. Kataloogis on lisaks veel Bitnami Moodle Stack ja tekstiformaadis logid. Logide haldamise süsteemi osad on eraldi Docker konteinerites, mille käivitamiseks kasutatakse Docker Compose'i, sama kehtib ka Bitnami Moodle Stacki komponentide kohta. Joonisel 6 on kujutatud lõputöös kasutatavate komponentide failipuu.



Joonis 6. Failipuu.

3.3 Docker Compose

Lõputöös loodav logide haldamise süsteem koosneb mitmest komponendist, ka seadistamiseks ja testimiseks kasutatav Bitnami Moodle Stack koosneb rohkem kui ühest komponendist. Sellest tulenevalt on kummagi süsteemi jooksutamiseks kasutatud Docker Compose tarkvara, mis võimaldab ühe käsuga korraga tööle panna mitu Dockeri konteinerit.

Eelnimetatust järeldub, et lõputöö raames käsitletakse kahte Docker Compose kogumikku. Vaikimisi luuakse ühe *Compose* faili raames üks võrk, mida saavad omavaheliseks suhtluseks kasutada kõik antud failis defineeritud konteinerid [15]. Siit tuleneb aga probleem, sest logimissüsteemi konteiner peab pääsema lugema Moodle'i poolt genereeritud logisid, mis salvestatakse Bitnami Moodle Stacki *Compose* failis defineeritud andmebaasi. Lahendusena tuleb luua võrk, mida saab kasutada üle mitme *Compose* faili, selleks on logimissüsteemi *Compose* failis defineeritud võrk nimega `custom_logging_nw`. Moodle'i andmebaasi konteiner saab eelnimetatud võrgu nime järgi ühilduda enda *Compose* failist väljaspool defineeritud võrku ja suhelda teiste seal

võrgus olevate konteineritega [16]. Antud näide on kujutatud joonisel 7 ja joonisel 8 väljavõtetena lõputöö raames loodud *Compose* failidest.

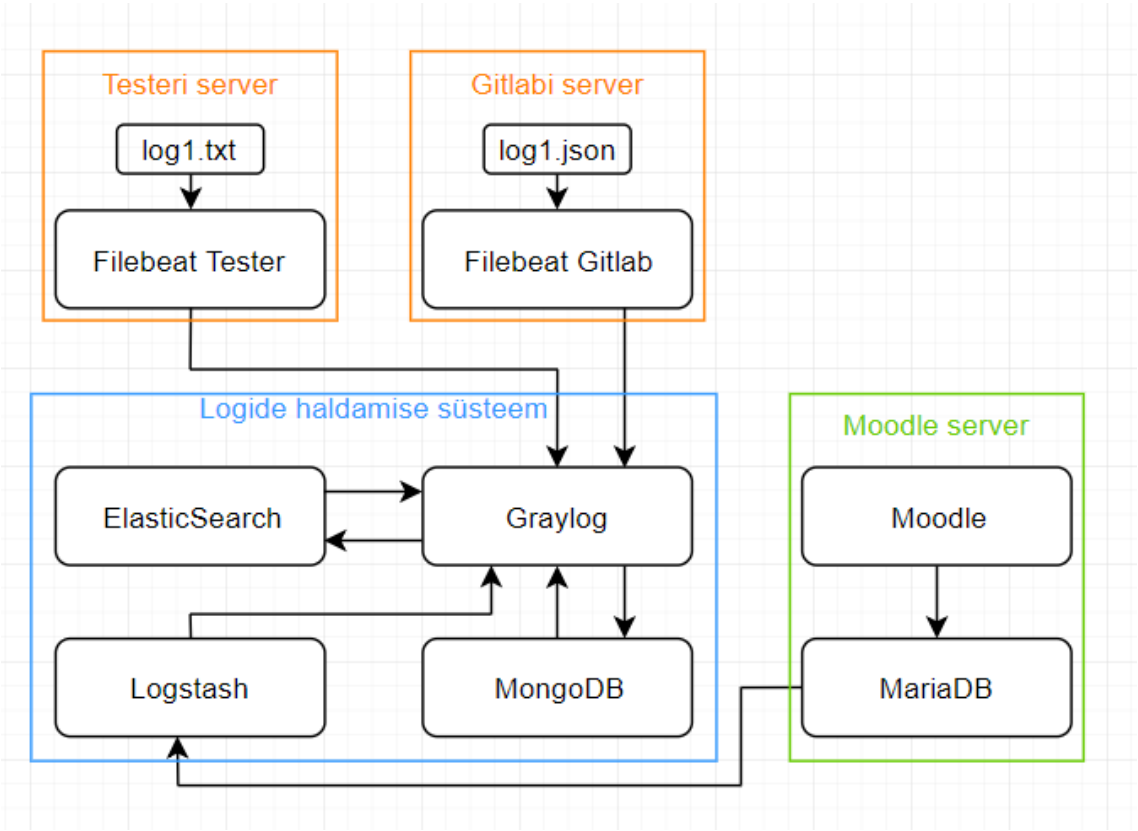
```
services:
  logstash:
      networks:
        - logging-network
networks:
  logging-network:
    name: custom_logging_nw
```

Joonis 7. Väljavõte logide haldamise süsteemi *Compose* failist.

```
services:
  mariadb:
    networks:
      - logging_nw
networks:
  logging_nw:
    external:
      name: custom_logging_nw
```

Joonis 8. Väljavõte Bitnami Moodle Stacki *Compose* failist.

Compose failis defineeritakse teenused, mis käivitatakse korraga ühe käsuga ning iga teenuse jaoks ehitatakse eraldi konteiner. Bitnami Moodle Stacki teenused on Moodle'i rakendus ja selle poolt kasutatav MariaDB andmebaas. Andmebaasi teenusel on keskkonnamuutujatena vaja määrata andmebaasi nimi ja kasutaja ning Moodle'i teenusel andmebaasi kirjutamiseks vajalikud keskkonnamuutujad. Logide haldamise süsteemi teenused on MariaDB, ElasticSearch, Logstash, kaks Filebeati instantsi ja Graylog. Kuna antud teenused moodustavad tähtsa osa lõputööst, on nende konteinerite seadistust pikemalt kirjeldatud järgmistes peatükkides. Joonisel 9 on kujutatud lõputöös kasutatavate komponentide vaheline andmete liikumine.



Joonis 9. Andmete liikumine seadistatavas logide haldamise süsteemis.

3.4 Logide haldamise süsteemi teenused

Logide haldamise süsteem koosneb mitmetest teenustest, millest igal on oma funktsioon. Järgnevates peatükkides on kirjeldatud nende teenuste konteinerite seadistust.

3.4.1 ElasticSearch

Joonisel 10 on kujutatud ElasticSearch teenuse väljavõte logide haldamise süsteemi *Compose* failist. ElasticSearchis indekseeritakse ja salvestatakse kõik Graylogi saadetud logid. Seetõttu peab ElasticSearch asuma Graylogiga samas võrgus, et ta oleks võimeline vastu võtma Graylogi poolt saadetud logisid. *Compose* failis ei ole vaja ElasticSearchi porti määrata, kui pole vajadust selle muutmiseks, kuna vaikimisi kasutab Graylog ElasticSearchiga suhtlemiseks porte 9200/tcp ja 9300/tcp [17].

ElasticSearch konteineri puhul kasutatakse `docker.elastic.co/elasticsearch/elasticsearch:6.5.1` *image*-t ehk pilti, mis on avalikult kättesaadav tõmmis konteinerist. Kuna mahukamaid lisaseadistusi ei ole hetkel vaja teha ja ka *host* ehk konteineri peremeesmasinasse ei ole

vaja konteineri töö jooksul tekkinud andmeid salvestada, ei pea teenuse `volume` ehk jagatud kettaruumi välja väärtustama.

Mõned ElasticSearch konteineri jaoks vajalikud väärtused tuleb seada `environment` suvandi ehk keskkonnamuutujate kaudu. Järgenvad punktid kirjeldavad täpsemalt *Compose* failis kasutatavate `elasticsearch` konteineri keskkonnamuutujate sisu [18] - [22].

- `http.host` muutujat kasutatakse `http.bind_host` ja `http.publish_host` väärtuste seadmiseks. Esimene on sisuliselt servermasina aadress, mille külge seotakse antud HTTP teenus. Teine on HTTP klientidele avaldatav servermasina aadress, mille kaudu kliendid saavad servermasinas paikneva teenuse poole pöörduda.

- `transport.host` muutujat kasutatakse `transport.bind_host` ja `transport.publish_host` väärtuste seadmiseks. Esimene on servermasina aadress kuhu seotakse transporditeenus. Teine on klastrisse kuuluvatele sõlmedele avaldatav aadress, mille kaudu sõlmed saavad suhelda servermasinaga.

- `network.host` muutuja väärtusega seotakse antud teenus ja aadressi avaldatakse ka teistele klastris olevatele sõlmedele, mille kaudu nad saavad teenuse poole pöörduda. Nii eelnimetatud HTTP kui ka transpordi muutujad tulenevad antud võrgu moodulist. Transpordi mooduli muutujaid kasutab sõlmede vaheliseks suhtluseks Java transpordi klient ja HTTP mooduli muutujaid kõik ülejäänud kliendid, kes soovivad suhtlemiseks kasutada JSON-it üle HTTP.

- `xpack` algusega keskkonnamuutujaid saab kasutada üldise turvalisuse ja monitooringuga seotud muutujate väärtustamiseks. Kuna suur osa X-Packi funktsioonidest kuulub tasulisse versiooni ei ole lõputöös ühtegi X-Packi funktsionaalsust kasutatud, vajadusel on võimalik tasuta versiooni kuuluvad funktsionaalsused sisse lülitada.

- `ES_JAVA_OPTS` kaudu on võimalik muuta ElasticSearchi poolt kasutatava Java virtuaalmasina muutujate väärtusi. Vaikimisi määrab ElasticSearch Java virtuaalmasina *maximum heap size* ehk mälu ülempiiri ja *miinimum heap size* ehk

esialgselt reserveeritud mälu suuruseks 1 GB. Vastavaid väärtusi on võimalik muuta seades sobivad Xmx ja Xms väärtused.

```
elasticsearch:
    container_name: elasticsearch
    image: docker.elastic.co/elasticsearch/elasticsearch:6.5.1
    environment:
      - http.host=0.0.0.0
      - transport.host=localhost
      - network.host=0.0.0.0
      - xpack.security.enabled=false
      - xpack.watcher.enabled=false
      - xpack.monitoring.enabled=false
      - xpack.security.audit.enabled=false
      - xpack.ml.enabled=false
      - xpack.graph.enabled=false
      - "ES_JAVA_OPTS=-Xmx512m -Xms512m"
    networks:
      - logging-network
```

Joonis 10. elasticsearch teenuse väljavõte logide haldamise süsteemi *Compose* failist.

3.4.2 MongoDB

Joonisel 11 on MongoDB teenuse väljavõte logide haldamise süsteemi *Compose* failist. MongoDB andmebaas on vajalik Graylogi konfiguratsiooni ja meta-andmete salvestamiseks. Selleks peab MongoDB olema Graylogiga samas võrgus. *Compose* failis ei ole MongoDB teenuse jaoks vaja porti määrata, kui pole vajadust selle muutmiseks, kuna Graylog kasutab MongoDB-ga suhtlemiseks viimase vaikimisi porte, milleks on 27017, 27018, 27019 [23].

MongoDB konteineri puhul kasutatakse Docker Hubis avalikult kättesaadavat ametlikku `mongo:3` Dockeri tõmmist.

```
mongodb:
    container_name: mongodb
    image: mongo:3
    networks:
      - logging-network
```

Joonis 11. mongodb teenuse väljavõte logide haldamise süsteemi *Compose* failist.

3.4.3 Graylog

Joonisel 12 on Graylogi teenuse väljavõte logide haldamise süsteemi *Compose* failist. Graylogi üheks ülesandeks on temani jõudnud logide saatmine ElasticSearchi ning sealt nende pärimine, kui kasutaja soovib salvestatud logidega teostada mingeid tegevusi,

näiteks otsida mustrile vastavaid kirjeid. Graylog asub `logging_network` võrgus, mida kasutavad ka teised temaga seotud teenused. Graylogi veebiliides ja API on kätte saadavad pordil 9000. Lisaks kasutab Graylog teatud porte erinevate sisendite jaoks, pordil 514 asub Syslog TCP, pordil 514/udp asub Syslog UDP, pordil 12201 asub GELF TCP ning pordil 12201/udp asub GELF UDP [24].

Graylogi puhul on kasutatud `graylog/graylog:2.5` Dockeri tõmmist. Andmesisendeid saab määrata Graylogi veebiliideses, kuid Graylogi teenuse konteineri taaskäivitamisel kaovad eelnevalt veebiliidese kaudu lisatud sisendid ära. Nii juhtub sellepärast, et sisendid seotakse konteineri ID-ga, kuid taaskäivitamisel genereeritakse konteinerile uus ID. Selleks, et sisendeid ei peaks peale igat taaskäivitust uuesti veebiliideses seadistama, on sisendite konfiguratsioon kirjutatud JSON formaadis faili, mille sisu loetakse Graylogi käivitamisel. Et konteineri sees oleks sisendite konfiguratsioonifaile võimalik lugeda, tuleb *host* masinas paiknev konfiguratsioonifailide kaust teha konteinerile kättesaadavaks. Selleks kasutatakse *Compose* faili `volumes` välja, mille kaudu saab jagada servermasina kettaruumi konteineriga ning vastupidi.

Kuna Graylogi tööks on vaja nii MongoDB kui ka ElasticSearch, tuleb nende teenuste konteinerid käivitada enne Graylogi konteinerit, selleks kasutatakse *Compose* faili `depends_on` välja. Lisaks on vaja väärtustada `links` väli, koos aliasega MongoDB pihta, kuna Graylogi konteineri konfiguratsioonis pöördutakse MongoDB poole kui `mongo`. Graylogi puhul on vaja defineerida ka mõned keskkonnamuutujad, vaikimisi on kasutajaliideses kasutatav kasutajanimi „admin“ ning hetkel ei ole seda muudetud. Järgnevad punktid kirjeldavad täpsemalt Graylogi keskkonnamuutujate sisu [25].

- `GRAYLOG_PASSWORD_SECRET` väärtust kasutatakse parooli krüpteerimiseks ja soola lisamiseks. Väärtus on kohustuslik ning ilma selleta keeldub server käivitumast.

- `GRAYLOG_ROOT_PASSWORD_SHA2` on SHA2 räsi Graylogi veebiliideses kasutatavast paroolist. Lõputöö raames on sarnaselt kasutajanimele parooli väärtus „admin“.

- `GRAYLOG_WEB_ENDPOINT_URI` on URI, mille kaudu pääseb Graylogi veebiliideseni.

- `GRAYLOG_CONTENT_PACKS_LOADER_ENABLED` väärtus võimaldab määrata, et Graylog loeb käivitamisel automaatselt määratud kataloogis olevaid konfiguratsioonifaile.

- `GRAYLOG_CONTENT_PACKS_DIR` kataloog, mis sisaldab Graylogi esmasel käivitamisel loetavaid konfiguratsioonifaile.

- `GRAYLOG_CONTENT_PACKS_AUTO_LOAD` komadega eraldatud list Graylogi esmasel käivitamisel loetavatest konfiguratsioonifailidest.

```
graylog:
    container_name: graylog
    image: graylog/graylog:2.5
    environment:
      - GRAYLOG_PASSWORD_SECRET=somepasswordpepper
      # Password: admin
      - GRAYLOG_ROOT_PASSWORD_SHA2=
8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918
      - GRAYLOG_WEB_ENDPOINT_URI=http://127.0.0.1:9000/api
      - GRAYLOG_CONTENT_PACKS_LOADER_ENABLED=true
      - GRAYLOG_CONTENT_PACKS_DIR=data/contentpacks
      - GRAYLOG_CONTENT_PACKS_AUTO_LOAD=gelf-udp.json,beats-
gitlab.json,beats-tester.json
    volumes:
      - ./contentpacks:/usr/share/graylog/data/contentpacks
    links:
      - mongodb:mongo
    depends_on:
      - mongodb
      - elasticsearch
    networks:
      - logging-network
    ports:
      - 9000:9000
      - 514:514
      - 514:514/udp
      - 12201:12201
      - 12201:12201/udp
```

Joonis 12. graylog teenuse väljavõte logide haldamise süsteemi *Compose* failist.

3.4.4 Logstash

Loodava logide haldamise süsteemi puhul on Logstashi vaja kasutada andmebaasi tabelist logide lugemiseks. Kuna Logstash saadab logid edasi Graylogi, peab ta olema viimasega samas võrgus. Logstashi pordiks on määratud 5001, kuid hetkel see otsest rakendust ei

leia, sest kasutuses olevad Filebeati teenused saadavad kogutud logid otse Graylogi ning Logstash kasutab andmebaasist logide lugemiseks MariaDB porti 3306.

Logstashi Dockeri tõmmis luuakse `/logging_system/logstash` kataloogis asuva Dockerfile'i põhjal, kuna antud lõputöö raames loodava logide haldamise süsteemi puhul ei piisa ainult Logstashi ametlikust Dockeri tõmmisest. Dockerfile'i `FROM` klauslis on defineeritud alusprojekt `docker.elastic.co/logstash/logstash:6.5.4`. Lisaks vajab Logstash Moodle'i MariaDB andmebaasiga ühendumiseks MySQL Connectorit, mis on Dockerfile'i lisatud JAR kujul. Logide andmebaasist lugemiseks vajab Logstash JDBC sisendi pistikprogrammi ja Graylogi saatmiseks GELF väljundi pistikprogrammi. Kuna enne logide edasi saatmist on vaja neid Graylogile vastuvõetavaks muuta, tuleb kasutada vastavat filtreerimise pistikprogrammi. Logstashi Dockerfile'i sisu on kujutatud joonisel 13.

```
FROM docker.elastic.co/logstash/logstash:6.5.4
USER root
COPY mysql-connector-java-8.0.13.jar /opt/
RUN logstash-plugin install --no-verify logstash-input-jdbc
RUN logstash-plugin install logstash-output-gelf
RUN logstash-plugin install logstash-filter-mutate
```

Joonis 13. Logstashi Dockerfile'i sisu.

Logstashi konteineri jaoks on vaja defineerida mitu `volumes` väärtust. Konteineris peab olema kättesaadav serverimasinas paiknev Logstashi konfiguratsioonifail ja kataloog milles paikneb nii-öelda torustiku konfiguratsioon. Viimane on vastutav andmebaasist logide lugemise eest. Lisaks sellele on vaja andmebaasist lugemisel salvestada viimati loetud kirje indikaator. Kuna indikaator peab kättesaadav olema ka peale konteineri peatamist või taaskäivitamist, tuleb see salvestada jagatud kettaruumile.

`env_file` suvandiga on määratud *Compose* failiga samas kataloogis asuv keskkonnamuutujaid sisaldav fail, antud juhul on seal Moodle'i andmebaasiga seotud muutujad. Logstashi puhul on *Compose* failis defineeritud üks keskkonnamuutuja, mille sisu on kirjeldatud järgnevas punktis.

- `LS_JAVA_OPTS` kaudu on võimalik muuta Logstashi poolt kasutatava Java virtuaalmasina muutujate väärtusi. Mälu ülempiiri suurust saab seada Xmx muutuja kaudu ning esialgselt reserveeritud mälu suurust Xms muutuja kaudu.

Joonisel 14 on kujutatud Logstashi teenuse väljavõte logide haldamise süsteemi *Compose* failist.

```
logstash:
    container_name: logstash
    build: logstash/
    volumes:
      - './logstash/config/logstash.yml:
        /usr/share/logstash/config/logstash.yml'
      - './logstash/pipeline:/usr/share/logstash/pipeline'
      - './logstash/lastrun:/root/.logstash_jdbc_last_run'
    environment:
      LS_JAVA_OPTS: '-Xmx1g -Xms1g'
    env_file:
      - .env
    networks:
      - logging-network
    ports:
      - '5001:5001'
```

Joonis 14. logstash teenuse väljavõte logide haldamise süsteemi *Compose* failist.

3.4.5 Filebeat GitLab

Joonisel 15 on GitLabi jaoks kasutatava Filebeati instantsi väljavõte logide haldamise süsteemi *Compose* failist. Ühte Filebeati instantsi on vaja GitLabi poolt tekitatud logide lugemiseks ja saatmiseks. Antud konteiner peab olema Graylogiga samas võrgus, et tal oleks võimalik sinna logisid saata. GitLabi Filebeati jaoks on kasutusel port 10006, kuid hetkel see reaalset kasutust ei leia, sest Filebeat saadab ise logid Graylogi pordile 5044.

Filebeati teenuse jaoks kasutatakse `docker.elastic.co/beats/filebeat:6.5.4` Dockeri tõmmist. Filebeati konteinerile on *Compose* faili `volumes` välja kaudu kättesaadavaks tehtud konfiguratsioonifail, mis on seotud logide lugemise ja saatmisega. Kuna Filebeat töötab samas masinas kus on logid, mida ta edasi saadab, peab Filebeati konteinerile ligipääsetav olema servermasinas paiknev logide asukoht.

```
filebeat_gitlab:
    container_name: filebeat_gitlab
    image: docker.elastic.co/beats/filebeat:6.5.4
    volumes:
      - './filebeat/filebeat_gitlab/config/filebeat.yml:
        /usr/share/filebeat/filebeat.yml'
      - '../logs/gitlab_logs:/gitlab_logs'
    networks:
      - logging-network
    ports:
      - '10006'
```

Joonis 15. filebeat_gitlab teenuse väljavõte logide haldamise süsteemi *Compose* failist.

3.4.6 Filebeat Tester

Joonisel 16 on Testeri jaoks kasutatava Filebeati instantsi väljavõte logide haldamise süsteemi *Compose* failist. Teist Filebeati instantsi on vaja Testeri logide lugemiseks ja saatmiseks. Antud konteiner peab samuti olema Graylogiga samas võrgus, et ta saaks sinna logisid saata. Testeri Filebeati jaoks on määratud port 10007, kuid hetkel ei leia see rakendust, sest Filebeat saadab ise logid Graylogi pordile 5045.

Sarnaselt GitLabile kasutatakse ka Testeri Filebeat teenuse jaoks `docker.elastic.co/beats/filebeat:6.5.4` Dockeri tõmmist. Filebeati konteinerile on `volumes` välja kaudu servermasina kettalt kättesaadavaks tehtud konfiguratsioonifail, mis on seotud logide lugemise ja edasi saatmisega. Lisaks sellele peab Filebeati konteinerile ligipääsetav olema temaga samas servermasinas olevate logide asukoht, vastasel juhul ei ole tal võimalik leida faile, mida lugeda.

```
filebeat_tester:
    container_name: filebeat_tester
    image: docker.elastic.co/beats/filebeat:6.5.4
    volumes:
      - './filebeat/filebeat_tester/config/filebeat.yml:
        /usr/share/filebeat/filebeat.yml'
      - '../logs/tester_logs:/tester_logs'
    networks:
      - logging-network
    ports:
      - '10007'
```

Joonis 16. filebeat_tester teenuse väljavõte logide haldamise süsteemi *Compose* failist.

4 Teenuste konfiguratsioon

Järgnevates peatükkides on kirjeldatud logide haldamise süsteemi logide edastamise teenuste konfiguratsiooni, lisaks sellele ka Graylogi konfiguratsiooni, mis on vajalik logide vastuvõtmiseks.

4.1 Logstash

Logstashi esimeseks ülesandeks on ühenduda andmebaasiga, milles talletatavad logid tahetakse Graylogi saata. Selleks on Logstashil vaja teada andmebaasi nime, andmebaasi kasutajanime ja parooli. Eelnimetatud muutujad on defineeritud konteineri käivitamisel loetavas `.env` keskkonnamuutujate failis, mille sisu on kujutatud joonisel 17.

```
DB_CONNECTION=mysql
DB_HOST=db
DB_PORT=3306
DB_DATABASE=bitnami_moodle
DB_USERNAME=bn_moodle
DB_PASSWORD=
```

Joonis 17. Moodle'i andmebaasiühenduse keskkonnamuutujad.

Andmebaasiga ühenduse loomiseks on kasutatud `logstash_input_jdbc` pistikprogrammi, mis võimaldab määratud aja tagant lugeda JDBC liidese kaudu andmebaasi kirjeid. Et juba loetud kirjeid uuesti ei loetaks, salvestab pistikprogramm määratud `sql_last_value` parameetri meta-andmete faili. Väärtust uuendatakse vastavalt päringuga saadud kirjetele [26].

JDBC sisendi puhul on kõigepealt vaja viidata sobivale andmebaasidraiveri teegile, milleks antud juhul on MySQL Connector/J 8.0.13. Lisaks sellele on vaja määrata draiveri klass, milleks on `mysql` ning JDBC ühenduse sõne, mis on sisuliselt andmebaasi URL. Seejärel peavad olema määratud andmebaasi kasutajanimi ja parool. Kahe andmebaasipäringu vahelist ajavahemikku saab määrata Cron formaadis, antud töö puhul on selleks kaks minutit. Kui `schedule` suvandiga ei ole ajavahemikku määratud, jooksutatakse päringut vaid ühe korra [26].

`statement` välja väärtuseks on andmebaasi pihta tehtav päring. Antud juhul on päritud kõik tabeli väljad, mille puhul „id“ on suurem eelnevalt loetud viimase kirje „id“-st. Nii saab vältida juba loetud kirjete korduvat pärimist. Selleks, et viimati loetud kirje

indikaatorina kasutataks andmebaasi välja tuleb `use_column_value` määrata tõeseks ja `tracking_column` väärtuseks määrata antud veeru nimi. Kuna „edulevel“ ja „anonymous“ väljad on `tinyint` tüüpi ja Logstash ei ole võimeline sellist andmetüüpi lugema, tuleb nende väljade väärtused teisendada `char` tüüpi väärtusteks. Eelnevalt kirjeldatud konfiguratsioon on kujutatud joonisel 18.

```
input {
  jdbc {
    jdbc_driver_library => "/opt/mysql-connector-java-8.0.13.jar"
    jdbc_driver_class => "com.mysql.jdbc.Driver"
    jdbc_connection_string => "jdbc:mysql://mariadb:3306/${DB_DATABASE}"
    jdbc_user => "${DB_USERNAME}"
    jdbc_password => "${DB_PASSWORD}"
    statement => "SELECT id, eventname, component, action, target,
objecttable, objectid, crud, CAST(edulevel AS char) AS edulevel, contextid,
contextlevel, contextinstanceid, userid, courseid, relateduserid,
CAST(anonymous AS char) AS anonymous, other, timecreated, origin, ip,
realuserid FROM mdl_logstore_standard_log WHERE id > :sql_last_value ORDER BY
id"
    use_column_value => true
    tracking_column => "id"
    schedule => "*/2 * * * *"
  }
}
```

Joonis 18. Logstashi JDBC sisendi konfiguratsioon.

Enne andmebaasist loetud andmete Graylogi saatmist tuleb neile Logstashis rakendada mõningaid filtreid. Kõigepealt konverteeritakse „edulevel“ ja „anonymous“ väljade väärtused tagasi numbriks, antud juhul täisarvuks. Seejärel lisatakse `short_message` ja `host` väljad, mis on GELF sõnumi puhul kohustuslikud. Esimene on sõnumit kirjeldav lühike lause, teine sõnumi saatmise allikas. Seejärel on vaja kõik andmebaasist päritud väljad ümber nimetada nii, et välja nime ette lisatakse alakriips, vastasel juhul ei oska Graylog saadetud lisa välju vastu võtta ja viskab nad kõrvale [27]. Andmetüüpide muutmine on eraldi `mutate` ploki sees sellepärast, et olla kindel, et andmetüübid muudetakse enne väljade ümbernimetamist, kuna ühe `mutate` ploki sees olevaid tegevusi ei tehta alati tegevuste plokis esinemise järjekorras. Eelnimetatud filtrite konfiguratsioon on kujutatud joonisel 19.

```
filter {
  mutate {
    convert => {
      "edulevel" => "integer"
      "anonymous" => "integer"
    }
  }
  mutate {
    add_field => { "short_message" => "moodle GELF message" }
    add_field => { "host" => "logstash" }
    rename => { "id" => "_mdl_id" }
    rename => { "eventname" => "_eventname" }
    rename => { "component" => "_component" }
    rename => { "action" => "_action" }
    rename => { "target" => "_target" }
    rename => { "objecttable" => "_objecttable" }
    rename => { "objectid" => "_objectid" }
    rename => { "crud" => "_crud" }
    rename => { "edulevel" => "_edulevel" }
    rename => { "contextid" => "_contextid" }
    rename => { "contextlevel" => "_contextlevel" }
    rename => { "contextinstanceid" => "_contextinstanceid" }
    rename => { "userid" => "_userid" }
    rename => { "courseid" => "_courseid" }
    rename => { "relateduserid" => "_relateduserid" }
    rename => { "anonymous" => "_anonymous" }
    rename => { "other" => "_other" }
    rename => { "timecreated" => "_timecreated" }
    rename => { "origin" => "_origin" }
    rename => { "ip" => "_ip" }
    rename => { "realuserid" => "_realuserid" }
  }
}
```

Joonis 19. Logstashi filtri konfiguratsioon.

Kõige viimasena on vaja lisada Logstashi väljund, mis saadaks kokku kogutud andmed sobivasse sihtpunkti. Antud juhul on kasutatud GELF väljundit, mis võimaldab lisada erinevaid välju säilitades seejuures nende andmetüübi. Väljundile on soovitatav lisada „id“, kuid see ei ole kohustuslik. Sihtkohaks on Graylogi konteiner pordil 12201, mis on GELF sõnumi vastuvõtmiseks kasutatav port. Väljundi konfiguratsioon on joonisel 20.

```
output {
  gelf {
    id => "moodle_gelf_message"
    host => "graylog"
    port => 12201
  }
}
```

Joonis 20. Logstashi väljundi konfiguratsioon.

4.2 Filebeat GitLab ja Filebeat Tester

GitLabi ja Testeri logide saatmiseks on kasutatud Filebeati, mis paigutatakse nii GitLabi poolt genereeritud logidega kui ka Testeri poolt genereeritud logidega samasse masinasse. Filebeatis peab logifailide lugemiseks määrama sisendi. `log` tüüpi sisend tähendab seda, et logifaili loetakse ridade kaupa ehk iga rida moodustab ühe Graylogi saadetava sündmuse. Antud sisendi tüübi puhul tuleb `paths` suvandi kaudu määrata logifailide asukoht. Lõputöö raames on nii GitLabi kui ka Testeri puhul viidatud ainult ühele logifailile, kuid vajadusel saab viidata alamkataloogidele ja nendes paiknevatele failidele. Näiteks `/var/log/*/*.log` puhul loetakse kõiki `/var/log` alamkataloogides paiknevaid `.log` faile. `fields` suvandiga saab igale loetud logikirjele lisada täiendavaid välju, antud juhul on lisatud `gitlab_log` ja `tester_log` väljad, et Graylogis oleks võimalik lihtsasti eristada kummagi rakenduse logisid [28].

Filebeatist saab logisid üle TCP Graylogi saata kasutades Logstashi väljundit, kuna eraldi Graylogi väljund puudub. See ei tähenda, et logisid saadetaks Graylogi läbi Logstashi, vaid, et Logstashi väljund on sobilik ka andmete Graylogi saatmiseks. `hosts` välja väärtusena määratakse serverid ja vastavad pordid, millele on konfigureeritud sissetulevaid Beats ühendused [29]. Antud juhul on kuulavaks serveriks Graylogi konteiner, mis ootab pordil 5044 GitLabi logisid ja pordil 5045 Testeri logisid. Filebeati GitLabi instantsi konfiguratsioon on joonisel 21 ja Testeri instantsi konfiguratsioon joonisel 22.

```
filebeat.inputs:
- type: log
  paths:
  - /gitlab_logs/json_log.log.2019-01-05.json
  fields:
    gitlab_log: true

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

Joonis 21. GitLabi Filebeati konfiguratsioon.

```
filebeat.inputs:
- type:
  paths:
  - /tester_logs/txt_log_2019-01-05.log
  fields:
    tester_log: true

output.logstash:
  hosts: ["graylog:5045"]
```

Joonis 22. Testeri Filebeati konfiguratsioon.

4.3 Graylog Content Pack

Content Pack-ks nimetatakse JSON faile, mis sisaldavad Graylogi komponentide konfiguratsiooni. Antud juhul on konfiguratsioonifailid vajalikud selleks, et Graylogi käivitamisel oleks vajalikud sisendid olemas ja ka vastavalt konfigureeritud. Järgnevates peatükkides on kirjeldatud Moodle'i, GitLabi ja Testeri logide vastuvõtmiseks vajalikku Graylogi konfiguratsiooni.

4.3.1 Moodle'i sisendi Content Pack

Graylogis kasutatakse Moodle'i logide lugemisel GELF sisendit. Kuna väärtused loetakse juba Logstashis vastavatele väljadele ja saadetakse samal kujul Graylogi, ei pea sisendi konfiguratsioonis sõnumit uuteks osadeks tükeldama.

Kõige tähtsamad sisendi konfiguratsiooni väljad on `bind_address` ja `port` ehk aadress ja port, millel kuulatakse sissetulevaid sõnumeid. Lisaks saab määrata antud sisendi internetiühenduses kasutatava vastuvõetavate andmete puhvri suuruse ja ka sõnumi suuruse peale lahti pakkimist. Mõlema eelnimetatud välja väärtused on baitides ja nende väljade väärtustamine ei ole kohustuslik. Vajadusel saab `override_source` suvandiga üle kirjutada paketi saatnud serveri nime mingi muu soovitud väärtusega. Lisas 1 on kuvatõmmis Graylogi saabunud Moodle'i logi kirje sõnumist. Joonisel 23 on Moodle'i logide vastuvõtmiseks kasutatava sisendi konfiguratsioon.

```
{
  "name": "Moodle UDP GELF input",
  "description": "Adds a global UDP GELF input on port 12201",
  "category": "Inputs",
  "inputs": [
    {
      "title": "moodle udp gelf input",
      "configuration": {
        "override_source": null,
        "recv_buffer_size": 262144,
        "bind_address": "graylog",
        "port": 12201,
        "decompress_size_limit": 8388608
      },
      "static_fields": {},
      "type": "org.graylog2.inputs.gelf.udp.GELFUDPInput",
      "global": true,
      "extractors": []
    }
  ]
}
```


Joonis 23. Moodle'i sisendi konfiguratsioonifaili sisu.

4.3.2 GitLabi sisendi Content Pack

Graylogis kasutatakse GitLabi logide lugemisel Beats sisendit. Suurem osa väljadest kattub GELF sisendi väljadega. Erinev on sisendi tüüp, milleks on BeatsInput ja lisaks on Beats sisendil mitmeid saadetavate andmete krüpteerimisega seotud TLS välju. Lõputöö puhul TLS-i ei kasutata.

GitLabi logid saabuvad Graylogi JSON objektina, mis tähendab, et logide paremaks hilisemaks analüüsimiseks tuleb sõnumist väljad lahti pakkida. Antud juhul on selleks kasutatud Graylogi poolt pakutavat JSON *extractor* tööriista. Lahti pakkimist vajav sõne loetakse `message` väljalt ja selleks kasutatakse `COPY` strateegiat, mis tähendab, et algne kirje jäetakse samuti alles. `condition` väljadega saab määrata millistele sõnumitele tükeldamist rakendatakse ning peamine konfiguratsioon koosneb erinevate eraldusmärkide defineerimisest. Lisas 2 on kuvatõmmis Graylogi saabunud GitLabi logi kirje sõnumist. Sisendi tükeldamiseks vajalik konfiguratsioon on joonisel 24.

```
{
    "title": "gitlab_extractor",
    "type": "JSON",
    "cursor_strategy": "COPY",
    "target_field": "",
    "source_field": "message",
    "configuration": {
        "list_separator": ", ",
        "kv_separator": "=",
        "key_prefix": "",
        "key_separator": "_",
        "replace_key_whitespace": false,
        "key_whitespace_replacement": "_"
    },
    "converters": [],
    "condition_type": "NONE",
    "condition_value": "",
    "order": 0
}
```


Joonis 24. GitLabi logide JSON *extractor*.

4.3.3 Testeri sisendi Content Pack

Testeri puhul sarnaneb sisendi konfiguratsioon samuti Moodle'i sisendi omale. Erinev on sisendi tüüp, milleks on BeatsInput ja ka siin on lisaks andmete krüpteerimisega seotud TLS väljad, mille tõeväärtused on antud töö puhul väärad. Testeri logid saabuvad Graylogi pika sõnena, mille osad tuleb tükeldada erinevate väljade väärtusteks, et logide filtreerimist oleks mugavam teostada. Lisas 3 on kuvatõmmis Graylogi saabunud Testeri logi kirje sõnumist.

Pikast sõnest väärtuste eraldamiseks on sisendi alla kuuluvas `extractors` loendis defineeritud *grok* tüüpi väärtuste eraldaja, mis võimaldab sõnumist korraga välja võtta mitmeid väljasid. Sisuliselt on *grok* hulk regulaaravaldisi, millest saab moodustada mustri, et seejärel alam-mustritele vastavad osad tekstist välja võtta, andes neile seejuures sobivad nimed. Joonisel 25 on Testeri logist vajalike andmete välja võtmiseks kasutatud *grok* muster [30].

```
%{TIME:time;date;HH:mm:ss}\\s..\\d\\d\\w\\s*%{LOGLEVEL:loglevel}..\\d.\\d\\d\
\w\\s..\\d\\d\\w.\\s*%{SESSIONKEY:sessionkey}...\\d.\\d\\d\\w\\s%{CLASS:class
}\\s.\\s%{MESSAGE:class_message}
```

Joonis 25. Testeri logi *grok extractor*.

Mustris on rakendatud topelt längkriipsu, et teisena esinevat längkriipsu tõlgendataks kui tavalist tähemärki. Ebavajaliku sisu kõrvale jätmiseks on kasutatud järgmisi meta-märke:

- `\s` tähistab suvalist tühimärki
- `\d` tähistab suvalist numbrimärki
- `\w` tähistab suvalist tähemärki
- `.` (punkt) tähistab suvalist üksikut kirjamärki
- `*` (asterisk) näitab, et tema ees paiknev märk võib puududa või teda võib olla suvaline arv kordi

Alam-mustritena on defineeritud väljad, mida soovitakse sõnumist välja võtta ja need koosnevad regulaaravaldisest. Kasutatud süntaksi puhul tähistab loogeliste sulgude sees olev esimene sõna mustri nime, teine sõnumist eraldatud väärtuse väljale antavat nime ja semikoolonile järgnevad väärtused andmetüüpi ning formaati. Joonisel 26 on alam-mustrite puhul kasutatavad regulaaravaldised [31].

```
{ "name": "SECOND", "pattern": "(?:(?:[0-5]?[0-9]|60)(?:[:.,][0-9]+)?)" },
{ "name": "MINUTE", "pattern": "(?:[0-5][0-9])" },
{ "name": "HOUR", "pattern": "(?:2[0123]|[01]?[0-9])" },
{ "name": "TIME", "pattern": "(?!<[0-9])%{HOUR}:%{MINUTE}(?::%{SECOND})(?![0-
9])" },
{ "name": "SESSIONKEY", "pattern": "\\d+" },
{ "name": "LOGLEVEL", "pattern":
"([A|a]lert|ALERT|[T|t]race|TRACE|[D|d]ebug|DEBUG|[N|n]otice|NOTICE|[I|i]nfo|
INFO|[W|w]arn?(?:ing)?|WARN?(?:ING)?|[E|e]rr?(?:or)?|ERR?(?:OR)?|[C|c]rit?(?:
ical)?|CRIT?(?:ICAL)?|[F|f]atal|FATAL|[S|s]evere|SEVERE|EMERG(?:ENCY)?|[Ee]me
rg(?:ency)?)" },
{ "name": "MESSAGE", "pattern": "[a-zA-Z\\s\\d\\W\\D]*" },
{ "name": "CLASS", "pattern": "[a-zA-Z.]*" }
```

Joonis 26. Alam-mustrite regulaaravaldised.

5 Logide analüüsimise päringud

Graylog võimaldab `search` ehk otsingu vahekaardi kaudu avataval kuval teostada päringuid kogutud logide pihta. Päringu süntaks sarnaneb Lucene'i süntaksile, kuid tuleb arvestada, et AND, OR ja NOT operaatorid tuleb päringus kirjutada suuri tähemärke kasutades. Kiirotsingu puhul on võimalik kasutada kolme erinevat ajamääratlust. Relatiivne ajamääratlus võimaldab otsingu nupule vajutamise ja määratud aja vahel saabunud kirjeid pärida. Absoluutse ajaga otsing võimaldab määrata täpse alguse ja lõppaja. Märksõnadega ajamääratlus võimaldab loomulike ingliskeelsete märksõnade kaudu defineerida otsitavate kirjete ajavahemikku, näiteks *4 hours ago* puhul leitakse kirjed mis jäävad praeguse hetke ja nelja tunnise aja tagusesse aega [32].

Kasutades otsingut ning andmeväljadega seotud vidinaid saab Graylogi infopaneelile luua otsingutulemusi visualiseerivaid plokke. Näiteks kõige rohkem esinevaid väärtusi saab sektordiagrammina kujutada `QUICKVALUES` vidina abil. Antud töö puhul on loodud kolm otsingutulemusi visualiseerivat plokki:

- viimase viie päeva jooksul kõige sagedasem Testeri kasutamise kellaaeg sektordiagrammina
- viimase viie päeva jooksul kõige enam esinenud Moodle'i tegevus sektordiagrammina
- viimase viie päeva jooksul kõige enam esinevad GitLabi kontrollerid sektordiagrammina. Lisa 4 kujutab antud sektordiagrammi infopaneelil ning joonisel 27 `dashboard.json` failis paiknevat komponendi konfiguratsiooni.

```
{
  "description": "top_gitlab_controllers",
  "type": "QUICKVALUES",
  "cache_time": 10,
  "configuration": {
    "timerange": {
      "type": "relative",
      "range": 432000
    },
    "field": "controller",
    "query": "gitlab_log:true",
    "show_data_table": true,
    "limit": 5,
    "show_pie_chart": true,
    "sort_order": "desc",
    "stacked_fields": "",
    "data_table_limit": 50
  },
  "col": 0,
  "row": 0,
  "height": 0,
  "width": 0
}
```


Joonis 27. Sektordiagrammi komponendi konfiguratsioon.

Infopaneeli kuvatavate graafiliste komponentide konfiguratsioon on samuti salvestatud *content pack*-i ehk JSON formaadis konfiguratsioonifaili. Fail loetakse küll automaatselt Graylogi, kuid infopaneeli aktiveerimiseks tuleb `system` rippmenüüst minna `content packs` lehele, valida `dashboards` alt loodud konfiguratsioon, antu juhul „General Dashboard Pack“ ning aktiveerida see nupust `apply content`. Seejärel on `dashboards` vahekaardilt võimalik lisatud konfiguratsiooniga infopaneeli kuva vaadata.

6 Testimine

Logide haldamise süsteemi seadistamise puhul oli esmalt vaja testida komponentide omavahelist ühendust. Näiteks Logstashil peab olema võimalus lugeda andmeid MariaDB andmebaasist, see tähendab, et ühes konteineris paiknev rakendus peab suutma ühenduda teise konteineriga. Selle testimiseks saab kasutada Ping võrguühenduse diagnostikaprogrammi, jooksutades ühe konteineri siseselt `ping` käsku teise konteineri pihta.

Logikirjet sisaldava sõnumi saatmist on võimalik testida Nmapiga. Graylog ei pruugi sõnumeid vastu võtta puuduva välja tõttu või jõuavad sõnumid Graylogi vigasel kujul. Sellise juhtumite puhul on kõige lihtsam saata Nmapiga erinevaid sõnumeid näiteks Logstashi või Filebeati konteinerist Graylogi ning vaadata, kas ja millisel kujul nad Graylogi jõuavad. Selline testimisviis võimaldab kergesti tuvastada, kui Graylog ei võta näiteks puuduva kohustusliku välja tõttu sõnumit vastu. Joonisel 28 on kujutatud käsk, mis sisaldab Logstashist Graylogi saadetavat näite sõnumit.

```
echo -n '{ "version": "1.1", "host": "logstash", "short_message": "moodle
GELF message", "level": 5, "_some_custom_field": "foo" }' | nc -w1 -u
logging_system_graylog_1 12201
```

Joonis 28. Nmap käsk sõnumi saatmiseks Logstashi konteinerist Graylogi.

7 Kokkuvõte

Lõputöö eesmärgiks oli seadistada logide haldamise süsteem Tallinna Tehnikaülikooli näitel kasutades Moodle'i, Testeri ja GitLabi poolt genereeritud logisid. Eelnimetatud logid on kõik erinevas formaadis, Moodle'i logid asuvad andmebaasi tabelis, GitLabi logid on JSON formaadis failis ning Testeri logid on Java logi formaadis tekstifailis. Lõputöö raames loodi vajalikud seadistused eelnimetatud tüüpi logide haldamiseks.

Logide haldamise süsteemiks valiti Graylog ja sellega kaasnevad MongoDB ning ElasticSearch. Logide lugemiseks ja Graylogi saatmiseks on kasutatud Logstashi ja kahte Filebeati instantsi. Üks osa konfiguratsioonist hõlmab kogu süsteemi üldist seadistust, mille hulka kuulub komponentidevaheline suhtlus võrgus ja komponentide üldine seadistus *Compose* failis. Teine osa konfiguratsioonist hõlmab logide lugemise, edastamise ja vastuvõtmisega seotud seadistusi. Lisaks on konfigureeritud näited logiandmete kuvamise visuaalsetest võimalustest.

Logide haldamise süsteemi seadistamine oli edukas ja süsteemi on võimalik kasutada ka muude lõputöös käsitletud tüüpi logisid genereerivate rakenduste puhul. Selliseid rakendusi saab süsteemi lisada minimaalse seadistusega, muutes loetavate failide asukohta.

Seadistatud logide haldamise süsteemi lähtekood asub TTÜ Giti salve aadressil https://gitlab.cs.ttu.ee/cepeik/log-management-system.

Kasutatud kirjandus

[1] GitLab, „Log system,“ [WWW]. Saadaval: https://docs.gitlab.com/ee/administration/logs.html. (22.03.2019).

[2] Elasticsearch B.V., „The Elastic Stack,“ [WWW]. Saadaval: https://www.elastic.co/products/. (22.03.2019).

[3] Elasticsearch B.V., „Kibana,“ [WWW]. Saadaval: https://www.elastic.co/products/kibana. (22.03.2019).

[4] Elasticsearch B.V., „Input plugins,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/logstash/current/input-plugins.html. (22.03.2019).

[5] Elasticsearch B.V., „Output plugins,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/logstash/current/output-plugins.html. (22.03.2019).

[6] P. Kamat, „DIY Elastic Stack: The 5 stages of grief,“ 2017 07 17. [WWW]. Saadaval: https://www.loggly.com/blog/diy-elastic-elk-stack-grief/. (22.03.2019).

[7] D. Berman, „Securing the ELK Stack with Nginx,“ 20 02 2019. [WWW]. Saadaval: https://logz.io/blog/securing-elk-nginx/. (22.03.2019).

[8] G. R, „What is Graylog?,“ 30 05 2018. [WWW]. Saadaval: https://medium.com/@gouthamr102/graylog-5265748cace0. (22.03.2019).

[9] Graylog, Inc., „Sending in log data,“ [WWW]. Saadaval: http://docs.graylog.org/en/2.5/pages/sending_data.html#what-are-graylog-message-inputs. (26.03.2019).

[10] Graylog, Inc., „Extractors,“ [WWW]. Saadaval: http://docs.graylog.org/en/2.5/pages/extractors.html. (26.03.2019).

[11] Graylog, Inc., „Saved searches,“ [WWW]. Saadaval: http://docs.graylog.org/en/2.5/pages/queries.html#saved-searches. (26.03.2019).

[12] Graylog, Inc., „Users,“ [WWW]. Saadaval: http://docs.graylog.org/en/2.5/pages/users_and_roles/users.html. (26.03.2019).

[13] Graylog, Inc., „Roles,“ [WWW]. Saadaval: http://docs.graylog.org/en/2.5/pages/users_and_roles/roles.html. (26.03.2019).

[14] Graylog, Inc., „Dashboard,“ [WWW]. Saadaval: http://docs.graylog.org/en/2.5/pages/dashboards.html. (27.03.2019).

[15] Docker Inc., „Networking in Compose,“ [WWW]. Saadaval: https://docs.docker.com/compose/networking/. (10.04.2019).

[16] Graylog, Inc., „Use a pre-existing network,“ [WWW]. Saadaval: https://docs.docker.com/compose/networking/#use-a-pre-existing-network. (10.04.2019).

[17] Graylog, Inc., „Configuration of Elasticsearch nodes,“ [WWW]. Saadaval: http://docs.graylog.org/en/3.0/pages/configuration/elasticsearch.html#control-access-to-elasticsearch-ports. (10.04.2019).

[18] Elasticsearch B.V., „HTTP,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/elasticsearch/reference/6.5/modules-http.html. (10.04.2019).

[19] Elasticsearch B.V., „TCP Transport,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/elasticsearch/reference/6.5/modules-transport.html. (10.04.2019).

[20] Elasticsearch B.V., „Commonly Used Network Settings,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/elasticsearch/reference/6.5/modules-network.html#common-network-settings. (10.04.2019).

[21] Elasticsearch B.V., „Setting the heap size,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html. (10.04.2019).

[22] Elasticsearch B.V., „General security settings,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/elasticsearch/reference/6.5/security-settings.html#general-security-settings. (10.04.2019).

[23] Graylog, Inc., „MongoDB,“ [WWW]. Saadaval: http://docs.graylog.org/en/3.0/pages/getting_started/configure.html#mongodb. (18.04.2019).

[24] Graylog, Inc., „Settings,“ [WWW]. Saadaval: http://docs.graylog.org/en/2.5/pages/installation/docker.html?highlight=ports#settings. (20.04.2019).

[25] Graylog, Inc., „server.conf,“ [WWW]. Saadaval: http://docs.graylog.org/en/2.5/pages/configuration/server.conf.html. (20.04.2019).

[26] Elasticsearch B.V., „Jdbc input plugin,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/logstash/6.5/plugins-inputs-jdbc.html. (20.04.2019).

[27] Graylog, Inc., „GELF Payload Specification,“ [WWW]. Saadaval: http://docs.graylog.org/en/2.5/pages/gelf.html#gelf-payload-specification. (20.04.2019).

[28] Elasticsearch B.V., „Log input,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/beats/filebeat/6.5/filebeat-input-log.html. (04.05.2019).

[29] Elasticsearch B.V., „Configure the Logstash output,“ [WWW]. Saadaval: https://www.elastic.co/guide/en/beats/filebeat/6.5/logstash-output.html. (04.05.2019).

[30] Graylog, Inc., „Using Grok patterns to extract data,“ [WWW]. Saadaval: https://docs.graylog.org/en/2.5/pages/extractors.html#using-grok-patterns-to-extract-data. (10.05.2019).

[31] „grok-patterns,“ [WWW]. Saadaval: https://github.com/elastic/logstash/blob/v1.4.0/patterns/grok-patterns. (10.05.2019).

[32] Graylog, Inc., „Searching,“ [WWW]. Saadaval: https://docs.graylog.org/en/2.5/pages/queries.html. (10.05.2019).

Lisa 1 – Moodle'i logi kirje Graylogis

Received by
moodle udp gelf input on
dd7de501 / a0df957172bb

Stored in index
graylog_0

Routed into streams
- All messages

action
viewed

anonymous
0

component
core

contextid
2

contextinstanceid
1

contextlevel
50

courseid
1

crud
r

edulevel
2

eventname
\core\event\course_viewed

facility
gelf-rb

full_message
%{message}

ip
192.168.48.1

level
6

mdl_id
67

message
moodle GELF message

origin
web

other
N;

protocol
0

source
logstash

target
course

timecreated
1558011550

timestamp
2019-05-16T13:00:00.245Z

userid
0

Lisa 2 – GitLabi logi kirje Graylogis

Received by
gitlab beats input on
dd7de501 / a0df957172bb

Stored in index
graylog_0

Routed into streams
- All messages

action
new

controller
SessionsController

db
2.01

duration
50.29

facility
filebeat

file
/gitlab_logs/production_json.log.2019-01-05_short_wo_message.json

format
html

gitlab_log
true

message
{"method":"GET","path":"/users/sign_in","format":"html","controller":"SessionsController","action":"new","status":200,"duration":50.29,"view":27.4,"db":2.01,"time":"2019-01-04T01:16:45.285Z","params":[],"remote_ip":"46.229.168.137","user_id":null,"username":null}

method
GET

name
fdfdbe2b9f2e

offset
5956

path
/users/sign_in

remote_ip
46.229.168.137

source
fdfdbe2b9f2e

status
200

time
2019-01-04T01:16:45.285Z

timestamp
2019-05-16T12:54:55.828Z

type
null

view
27.4

Lisa 3 – Testeri logi kirje Graylogis

Received by
tester beats input on
dd7de501 / a0df957172bb

Stored in index
graylog_0

Routed into streams
- All messages

HOUR
16

MINUTE
18

SECOND
57

class
e.t.c.a.s.e.TimeLimitTestJobExecutor

class_message
Container exited with status 0

facility
filebeat

file
/tester_logs/api_short.log

loglevel
DEBUG

message
16:18:57 □[33mDEBUG□[0;39m □[35m[715823777]□[0;39m e.t.c.a.s.e.TimeL
imitTestJobExecutor - Container exited with status 0

name
44688416a800

offset
27333

sessionkey
715823777

source
44688416a800

tester_log
true

time
1970-01-01T16:18:57.000+0000

timestamp
2019-05-16T12:54:56.743Z

type
null

Lisa 4 – Sektordiagramm GitLabi kontrollerite esinemistihedusest