TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Alen Siilivask 210747 IAAB

Seiresüsteemi püstitamine ettevõttele Küehne-Nagel

Bakalaureusetöö

Juhendaja: Margus Sumla
MSc

Tallinn 2023

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: Alen Siilivask

15.05.2023

Annotatsioon

Selle bakalaurusetöö eesmärk on luua efektiivne ja paindlik seiresüsteemi lahendus, mis tagab Küehne-Nageli projektile SwiftLOG sujuva toimimise. Bakalaurusetöö teoreetiline osa koosneb probleemi püstitamisest ja projekti eesmärkide seadmisest ning tingimustest, millele lõpptulemus peab lõpuks vastama. Samuti tehakse ülevaade populaarsetest seiresüsteemidest ja valitakse sobivaimad neist.

Praktilises osas kirjeldatakse samm-sammult lahenduse loomist ja seadistamist, pöörates erilist tähelepanu Prometheus ja Grafana integreerimisele, Jenkins serveri ja Nexus repositooriumi jälgimisele. Lisaks käsitletakse Docker'i kasutamist tarkvara paigaldamise ja haldamise hõlbustamiseks.

Lõpuks esitatakse lahenduse analüüs, sealhulgas seiretulemused, mis käsitlevad Jenkinsi ja Nexus *repository* toimimist. Projekt tõestab valitud jälgimislahenduse efektiivsust ja paindlikkust ning näitab, kuidas selline süsteem aitab tagada projektile SwiftLOG sujuva toimimise.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 35 leheküljel, 6 peatükki, 37 joonist, 2 tabelit.

Abstract

Implementing a Monitoring Solution for Küehne-Nagel Company

The aim of this bachelor thesis is to create an effective and flexible monitoring solution that ensures the smooth operation of the Küehne-Nagel project SwiftLOG. The theoretical part of the bachelor thesis consists of setting the problem and defining the objectives of the project, as well as the conditions that the final outcome must ultimately meet. An overview of popular monitoring systems is provided and the most suitable ones are selected.

In the practical part, the creation and configuration of the solution are described step-by-step, with particular attention paid to the integration of Prometheus and Grafana, as well as the monitoring of the Jenkins server and Nexus repository. The use of Docker for the installation and management of the software is also discussed.

Finally, an analysis of the solution is presented, including the monitoring results concerning the performance of the Jenkins and Nexus repository. The project proves the effectiveness and flexibility of the chosen monitoring solution and demonstrates how such a system helps ensure the smooth operation of the SwiftLOG project.

The thesis is in Estonian and contains 35 pages of text, 6 chapters, 37 figures, 2 tables.

Lühendite ja mõistete sõnastik

DevOps	Tarkvaraarenduse ja IT-tegevuse automaatimise ja koordineerimise metoodika
DNS	*Domain Name System*, võrguprotokoll domeeninimede ja IP-aadresside vahetamiseks
HTTP	*HyperText Transfer Protocol,* võrguprotokoll andmete edastamiseks klientide ja veebiserverite vahel
HTTPS	*HyperText Transfer Protocol Secure,* krüpteeritud võrguprotokoll andmete turvaliseks edastamiseks klientide ja veebiserverite vahel
ICMP	*Internet Control Message Protocol*, võrguprotokoll, mida kasutatakse sõnumite edastamiseks ja võrguteenuste seisundi jälgimiseks
IP	*Internet Protocol,* võrguprotokoll, mis pakub internetiühenduse kaudu aadresside määratlemist ja pakettide edastamist
QA	*Quality Assurance,* tarkvara kvaliteedi tagamist või lihtsalt kvaliteedikontrolli
SSL	*Secure Sockets Layer*, krüpteerimisprotokoll andmete turvaliseks edastamiseks võrgus
TLS	*Transport Layer Security*, krüpteerimisprotokoll andmete turvaliseks edastamiseks võrgus
URL	*Uniform Resource Locator,* veebiaadress, mis näitab, kus asub konkreetne veebileht

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva bakalaureusetöö eesmärk on luua Küehne-Nagel ettevõtte SwiftLOG projektile tõhus ja usaldusväärne seiresüsteem, mis keskendub sellele, kuidas luua ja integreerida seirelahendus, mis tagaks projektile vajaliku järelevalve ja hoiatussüsteemi. Selle saavutamiseks antakse töös ülevaade ettevõttest ja SwiftLOG projektist, mis tegeleb laosüsteemide lahendustega, ning uuritakse praeguseid probleeme ja puudujääke, mis tulenevad seiresüsteemi puudumisest Nexus'i *repository* ja Jenkins'i serveri jaoks.

Lahenduse nõuete analüüs keskendub nõuete ja eelistuste esitamisele seiresüsteemile, mis tagaks sobivuse SwiftLOG projekti jaoks ning oleks lihtsalt seadistatav ja tasuta. Seejärel analüüsitakse ja võrreldakse erinevaid seiretarkvarasid, nagu Zabbix, Nagios, Prometheus ja Grafana, et valida sobivaimad lahendused projekti jaoks.

Töö lõppfaasis kirjeldatakse, kuidas integreerida valitud seirelahendus projektiga, alates tarkvara paigaldamisest ja seadistamisest kuni lahenduse jõudluse analüüsini. Seirelahenduse rakendamisel pööratakse erilist tähelepanu sellele, kuidas jälgida ainult Jenkins ja Nexus *repository* süsteeme, kasutades Prometheus ja Grafana tehnoloogiaid.

Lõpptulemusena pakub käesolev bakalaureusetöö põhjalikku ülevaadet seiresüsteemi püstitamise protsessist Küehne-Nagel SwiftLOG projektile, aidates parandada süsteemi järelevalvet ja hoiatusi ning suurendades projektis osalevate meeskondade efektiivsust ja tootlikkust. Töö toob välja nii praeguste probleemide ja puuduste analüüsi kui ka lahenduse nõuete analüüsi, tagades tervikliku ja põhjaliku lähenemise seiresüsteemi loomisele ja rakendamisele.

2 Ettevõtte kirjeldus

Kühne-Nagel on ülemaailmne logistika- ja tarneahela juhtimise ettevõte, millel on rohkem kui 130 aastane kogemus logistikasektoris. Asutatud 1890. aastal Saksamaal, on Kühne-Nagel laienenud enam kui 100 riiki ning pakub laia valikut teenuseid, nagu mere, õhu-, raudtee- ja maanteetransport, ladustamine, tollivormistus ja tarneahela konsultatsioonid.

Projekt SwiftLOG on Kühne-Nagel'i uuenduslik algatus, mis tegeleb laotööstuse protsesside ja süsteemide optimeerimisega. SwiftLOG’i eesmärk on pakkuda oma klientidele kõrgetasemelist teenust, tõhusat tööprotsessi ja kulude vähendamist, kasutades uusimaid tehnoloogiaid, nagu tehisintellekt, andmeanalüüs ja pilvepõhised lahendused.

Hetkel on SwiftLOG projektis probleem, et Nexus repositoril ja Jenkins serveritel pole püstitatud seiresüsteem. Selle tagajärjel võivad tekkida probleemid süsteemi jõudluse, töökindluse ja turvalisuse osas.

2.1 Praegune seiresüsteemi lahendus ettevõttes

Kühne-Nagel'i SwiftLOG projektis puudub hetkel seiresüsteem Nexus *repository* ja Jenkins serverite jaoks, mis tekitab olulisi probleeme igapäevases tööelus. Seiresüsteemi puudumine tähendab, et süsteemide jõudluse, töökindluse ja turvalisuse tagamine võib olla keeruline ja aeganõudev. Selle tulemusena peavad töötajad sageli otsima ja lahendama probleeme käsitsi, mis võib olla aeglasem, ebaefektiivsem ja vähem täpne kui automatiseeritud seirelahendus.

Üks näide probleemidest, mis võivad tekkida seiresüsteemi puudumisel, on Jenkinsi *pipeline'* ebaõnnestumine, mis võib olla tingitud Nexus *repository* serveri ootamatust taaskäivitamisest. Sellises olukorras peavad arendajad ja ops-tiimid kulutama aega ja ressursse, et välja selgitada probleemi põhjus, analüüsida logisid ning leida ja rakendada sobiv lahendus. See võib põhjustada viivitusi projektide ajakavas, suurendada tööjõukulusid ja vähendada töötajate tootlikkust.

Lisaks võib seiresüsteemi puudumine põhjustada probleeme süsteemi koormuse ja ressursikasutuse juhtimisel. Kuna Nexus *repository* ja Jenkins serveri töökoormus võib varieeruda erinevatel aegadel ja sõltuvalt projektide vajadustest, on oluline tagada, et süsteemide ressursid oleksid optimeeritud ja piisavad, et toetada kõiki töövooge. Seiresüsteemi puudumine võib muuta selle ülesande keeruliseks ja vähem täpseks, suurendades süsteemi ülekoormuse ja jõudluse probleemide riski.

2.2 Probleemi lahenduse võimalused

Seiresüsteemi püstitamine Nexus *repository* ja Jenkins serveri jaoks on oluline samm probleemide lahendamisel, mis on seotud süsteemide jõudluse, töökindluse ja turvalisusega. Seiresüsteem võimaldab ettevõttel Kühne-Nagel jälgida süsteemide olekut ja jõudlust reaalajas, hõlbustades kiiret reageerimist ja probleemide lahendamist.

Üks võimalik lahendus on valida ja rakendada sobiv seiresüsteemi tarkvara, mis on spetsiaalselt loodud IT-infrastruktuuri jälgimiseks ja haldamiseks. Selline tarkvara peaks olema võimeline koguma ja analüüsima erinevaid süsteemide meetrikaid, nagu koormus, ressursikasutus, töökindlus ja vastuseajad. Tarkvara peaks olema ka paindlik ja skaleeritav, et toetada ettevõtte kasvavat ja muutuvat IT-infrastruktuuri.

Teiseks on oluline luua teavitussüsteem, mis hoiatab töötajaid automaatselt, kui süsteemides ilmneb probleem või kui server on maas. Teavitused võivad hõlmata e-kirju, tekstsõnumeid, mobiilirakenduste märguandeid või integratsioone kolmandate osapoolte tööriistadega, nagu Zoom. Teavitussüsteem peaks olema konfigureeritav, et töötajad saaksid seadistada hoiatusi vastavalt oma vajadustele ja eelistustele.

3 Lahenduse nõuete analüüs

Seiresüsteemi valimisel Küehne-Nageli projektile SwiftLOG tuleb arvestada mitmete oluliste tegurite ja nõuetega. Järgnevalt on välja toodud mõned peamised nõuded, mida seiresüsteem peaks täitma:

1. Kaasaegsus: Seiresüsteem peab olema kaasaegne, kasutades uusimaid tehnoloogiaid ja lähenemisviise, et tagada parim võimalik jõudlus ja tõrgeteta integreerimine olemasolevate süsteemidega, nagu Jenkins ja Nexus *repository*.

2. Paindlikkus ja skaleeritavus: Seiresüsteem peab olema hõlpsasti kohandatav, et rahuldada projekti erinevaid vajadusi ning kasvada ja areneda koos ettevõttega. See peaks võimaldama lisada uusi jälgitavaid komponente või muuta olemasolevaid ilma suurema vaevata.

3. Lihtne seadistamine: Lahendus peab olema lihtsasti seadistatav ja hooldatav. See peaks võimaldama kiiret seadistamist ja kasutuselevõttu, et vähendada IT-meestkonna töökoormust ja tagada kiirem kasutuselevõtt.

4. Tasuta ja avatud lähtekoodiga: Seiresüsteem peaks olema tasuta ja avatud lähtekoodiga, et vähendada kulusid ja võimaldada ettevõttel kohandada lahendust vastavalt oma vajadustele. Samuti peaks see tagama tugeva kogukonna ja dokumentatsiooni, mis toetab lahenduse kasutamist ja arengut.

5. Hoiatused ja teavitused: Lahendus peaks pakkuma hoiatuste ja teavituste süsteemi, mis võimaldab meeskonnal kiiresti tuvastada probleeme ja tegutseda nende lahendamiseks. Teavitused peaksid olema konfigureeritavad, et saaks määrata sobivad läveväärtused ja reageerida vastavalt vajadusele.

6. Aruandlus ja visualiseerimine: Seiresüsteem peaks pakkuma tõhusaid aruandlus- ja visualiseerimisvõimalusi, mis võimaldavad meeskonnal paremini mõista süsteemi toimimist ja tuvastada võimalikke probleeme.

Ülaltoodud nõuete arvessevõtmine aitab tagada, et valitud seiresüsteem on sobiv Küehne-Nageli projektile SwiftLOG, toetades ettevõtte eesmärke ning pakkudes tõhusaid ja usaldusväärseid lahendusi süsteemide jälgimiseks ja haldamiseks. Seiresüsteemi valikul tuleks kaaluda, kuidas see toetab meeskonna tööd ja suurendab projekti edukuse tõenäosust.

7. Integreerimine olemasolevate süsteemidega: Valitud seiresüsteem peaks olema võimeline integreeruma sujuvalt ettevõtte olemasolevate süsteemidega, nagu Jenkins ja Nexus *repository*.

8. Turvalisus ja privaatsus: Seiresüsteem peab tagama andmete turvalisuse ja privaatsuse, kaitstes neid volitamata juurdepääsu ja lekete eest. Lahendus peaks sisaldama turvameetmeid, nagu autentimine, krüptimine ja auditilogide säilitamine, et tagada ettevõtte andmete kaitse.

9. Dokumentatsioon ja õppematerjalid: Seiresüsteem peaks pakkuma põhjalikku ja ajakohast dokumentatsiooni, mis aitab meeskonnal kiiresti õppida ja kasutada lahendust. See peaks hõlmama nii tehnilist dokumentatsiooni kui ka juhiseid ja näiteid, mis aitavad meeskonnal rakendada lahendust oma keskkonnas.

10. Kasutajasõbralikkus: Seiresüsteem peaks olema intuitiivne ja kasutajasõbralik, et hõlbustada meeskonna liikmete tööd ning vähendada koolitusaega. Süsteemi tuleks olema võimalik hõlpsalt navigeerida ja selle funktsioonid peaksid olema selgelt mõistetavad, hõlbustades seeläbi kasutajatele süsteemi kasutuselevõttu ja selle funktsioonide kasutamist.

Arvestades neid nõudeid, tuleb seiresüsteemi valik teha hoolikalt, et tagada, et see toetab projekti eesmärke ja aitab meeskonnal tõhusalt jälgida ja hallata IT-infrastruktuuri.

4 Seiresüsteemi vahendite ülevaade

Selles peatükis antakse põhjalik ülevaade erinevatest olemasolevatest seiresüsteemidest, mis võiksid olla potentsiaalsed kandidaadid seirelahenduse loomiseks Küehne-Nagel SwiftLOG projektile. Eesmärk on tutvustada laialdaselt kasutatavaid seiretarkvarasid, mis aitavad tagada IT-süsteemide stabiilsust ja jõudlust.

Ülevaade keskendub erinevate tarkvarade eelistele ja puudustele, sealhulgas nende paindlikkusele, laiendatavusele, kohandatavusele, kasutusmugavusele, haldusvõimalustele ja hinnale. Lisaks uuritakse nende sobivust projekti konkreetsete vajaduste ja eelistuste põhjal, arvestades Küehne-Nageli SwiftLOG projekti spetsiifilisi nõudmisi ja ootusi.

Töö selles osas võetakse arvesse mitmeid seiresüsteeme, mille hulka kuuluvad Zabbix, Nagios, Prometheus ja Grafana. Nende süsteemide võrdlus võimaldab paremini mõista nende potentsiaalset mõju Küehne-Nageli projektile, aidates otsustada, milline neist sobib kõige paremini projekti jaoks.

Peatükk toob välja ka iga süsteemi eripärad ja funktsioonid ning annab ülevaate nende rakendusvõimalustest ja piirangutest. Lõpuks aitab see ülevaade teha teadliku otsuse seiretarkvara valiku osas, mis vastab kõige paremini Küehne-Nageli SwiftLOG projekti nõuetele ja eelistustele.

4.1 Ülevaade olemasolevatest seiresüsteemi vahenditest

Olemasolevate seiresüsteemi vahendite valik on mitmekesine, pakkudes erinevaid funktsioone ja eeliseid, mis aitavad jälgida ja hallata Nexus *repository* ja Jenkins serveri tööd. Allpool on esitatud mõned populaarsed seiresüsteemi vahendid, mis on levinud IT-infrastruktuuri jälgimiseks ja haldamiseks.

4.1.1 Zabbix

Zabbix on avatud lähtekoodiga jälgimistööriist, mida saab kasutada serverite, võrkude, IT-komponentide, pilveteenuste ja virtuaalmasinate jälgimiseks. See pakub

jälgimismõõdikuid ja jälgib võrgu kasutust, ketta ruumikasutust ja CPU koormust. Zabbix on ettevõtteklassi hajutatud jälgimislahendus serverite jälgimiseks [1]. Seda saab kasutada võrgu mitmete parameetrite ja serverite, virtuaalmasinate, rakenduste, teenuste, andmebaaside, veebisaitide, pilve jne tervise ja terviklikkuse jälgimiseks. Zabbix on loodud võrgu infrastruktuuri erinevate aspektide jälgimiseks, sealhulgas serverite, rakenduste, võrguseadmete, andmebaaside, virtualiseerimisplatvormide (VMware, HyperV) jne jälgimiseks. Zabbixiga saab jälgida praktiliselt kõike, millel on IP-aadress [2].

Zabbixil on palju erinevaid viise andmete kogumiseks, nende töötlemiseks, analüüsimiseks ja visualiseerimiseks. Samuti pakub see administraatoritele suurt paindlikkust sobiva valiku valimisel igas olukorras. Zabbix pakub ettevõtte tasemel jälgimist ja tuge klientidele üle maailma [3] [4] [5].

4.1.2 Nagios

Nagios on avatud lähtekoodiga jälgimissüsteem arvutisüsteemidele, mis käivitab perioodilisi kontrollimisi rakenduste, võrgu ja serveriressursside kriitiliste parameetrite osas. See oli mõeldud töötama Linuxi operatsioonisüsteemil ja saab jälgida seadmeid, mis töötavad Linuxi, Windowsi ja Unixi operatsioonisüsteemidel [6]. Nagiose tarkvara käivitab perioodilisi kontrollimisi rakenduste, võrgu ja serveriressursside kriitiliste parameetrite osas. See suudab jälgida serverit ja leida jõudluse kitsaskohti. Nagios on võimas jälgimissüsteem, mis võimaldab organisatsioonidel tuvastada ja lahendada IT-infrastruktuuri probleeme enne nende mõju kriitilistele äriprotsessidele [7]. See pakub põhjalikke jälgimisvõimalusi rakenduste, teenuste, operatsioonisüsteemide, võrguprotokollide, süsteemi mõõdikute ja infrastruktuuri komponentide jälgimiseks [8].

4.1.3 Prometheus

Prometheus on avatud lähtekoodiga süsteemide jälgimis- ja hoiatuskomplekt, mis loodi algselt SoundCloudis 2012. aastal. Sellest ajast alates on paljud ettevõtted ja organisatsioonid selle omaks võtnud ning sellel on väga aktiivne arendajate ja kasutajate kogukond. Prometheus kogub mõõdikuid sihtmärkidelt, kraapides mõõdikute HTTP-lõpp-punkte. Kuna Prometheus avaldab andmeid iseenda kohta samal viisil, saab see jälgida ja kontrollida ka omaenda tervist. Kuigi ainult iseenda andmeid koguv Prometheus'i server pole eriti kasulik, on see hea algne näide [9].

Prometheus pakub võimsat päringukeelt nimega PromQL, mis võimaldab teil reaalajas valida ja koguda aegridade andmeid. Lisaks pakub see veebiliidest mõõdikute andmete pärimiseks ja visualiseerimiseks. Prometheus't saab kasutada kõige alates serverite, rakenduste, andmebaaside ja võrguseadmete jälgimisest. Seda saab kasutada ka Kubernetes'i klasterite ja nende peal töötavate teenuste jälgimiseks [10].

4.1.4 Grafana

Grafana on avatud lähtekoodiga vaatlusplatvorm, mis võimaldab kasutajatel oma andmeid graafikutena ja diagrammidena näha, mis on ühendatud ühe juhtpaneeliga lihtsamaks tõlgendamiseks ja mõistmiseks [11]. See on populaarne komponent jälgimisvarudes ning seda kasutatakse sageli koos ajatempel andmebaasidega nagu InfluxDB, Prometheus ja Graphite [12].

4.2 Vahendite analüüs ja võrdlus

Järgnevas jaotises analüüsitakse ja võrreldakse nelja erinevat seirevahendit: Zabbix, Nagios, Prometheus ja Grafana. Vaadeltakse nende omadusi, eeliseid ja puudusi, et hinnata nende sobivust SwiftLOG projektile (Tabel 1).

Tabel 1. Seiresüsteemide võrdlus

Seiresüsteem	Eelised	Puudused
Zabbix	Avatud lähtekood, tasuta, ulatuslik seire, võimas seadistus, hea jõudlus, graafilised paneelid, aktiivne kogukond, hea dokumentatsioon	Keeruline seadistamine, vajab lisa tarkvara
Nagios	Avatud lähtekood, laialt kasutusel, tugev jälgimisvõimekus, palju lisamooduleid, aktiivne kogukond, hea dokumentatsioon	Keeruline seadistamine, graafiliste paneelide puudumine
Prometheus	Avatud lähtekood, tõhus andmekogumine, paindlik, võimas päringukeel, integreerub Grafanaga	Pole mõeldud võrguseireks, vajab täiendavat teavitussüsteemi
Grafana	Avatud lähtekood, laialt kasutusel, kohandatud armatuurlauad, tugev visualiseerimisvõimekus, ühildub paljude andmebaasidega, kasutajasõbralik	Ei paku seirefunktsioone, mõned tasulised funktsioonid

Kõik neli tööriista pakuvad erinevaid võimalusi ja eeliseid seiresüsteemi loomiseks Küehne-Nageli projektile SwiftLOG. Zabbix ja Nagios on vanemad ja hästi tuntud lahendused, mis sobivad erinevate IT-komponentide, nagu riistvara, teenuste ja rakenduste jälgimiseks. Kuid nende seadistamine ja kohandamine võib olla keeruline ning nende graafilised paneelid ja hoiatused ei ole alati kõige intuitiivsemad.

Prometheus ja Grafana on uuemad ja kiiremini arenevad tööriistad, mis pakuvad suuremat paindlikkust, laiendatavust ja kohandatavust. Prometheus on eriti tugev konteinerite seireks, samas kui Grafana võimaldab luua kohandatud seirevaadeid ja visualiseerida andmeid erinevatest allikatest. Nende tööriistade integreerimine võimaldaks luua tervikliku seire- ja hoiatussüsteemi, mis vastab Küehne-Nageli projektile SwiftLOG esitatavatele nõudmistele.

4.3 Tehnoloogia valik

Pärast analüüsi ja arvessevõtmist Küehne-Nageli projektile SwiftLOG esitatavaid nõudmisi otsustati kasutada Prometheus andmete kogumise süsteemina ja Grafana kui

jälgimislahendust. See kombinatsioon võimaldab luua tervikliku seire- ja hoiatussüsteemi, mis on paindlik, laiendatav ning vastab projektinõuetele ja eelistustele. Selles lahenduses keskendutakse peamiselt Jenkinsi ja Nexus *repository* süsteemide jälgimisele.

Nii Prometheus kui ka Grafana pakuvad laialdasi kohandamisvõimalusi, mis võimaldavad luua kohandatud seirelahendusi vastavalt projekti erivajadustele ja eelistustele. See hõlmab näiteks hoiatusreeglite loomist ja graafikute kujundamist. Grafana kasutajaliides on intuitiivne ja lihtne kasutada, võimaldades projektitiimil kiiresti ja hõlpsasti luua ja hallata seirelahendusi ilma keeruliste õppimiskõverateta.

Prometheus ja Grafana on ühilduvad paljude erinevate operatsioonisüsteemide, andmebaaside ja muude IT-infrastruktuuri komponentidega. See tagab, et neid saab integreerida erinevatesse keskkondadesse ja neid saab kasutada mitmesuguste rakenduste ja teenuste jälgimiseks, sealhulgas Jenkinsi ja Nexus *repository* süsteemidele. Lisaks on mõlemad tehnoloogiad optimeeritud ressursside tõhusaks kasutamiseks, pakkudes madalat mälutarbimist ja suurt jõudlust isegi suurte andmekogumite puhul. See võimaldab projektil SwiftLOG pakkuda kvaliteetset seirelahendust minimaalse ressursikuluga.

Prometheus ja Grafana on loodud tagama kõrget kättesaadavust ja töökindlust, mis on oluline projekti SwiftLOG jaoks, kus süsteemi järjepidev jälgimine on kriitilise tähtsusega. Selle saavutamiseks kasutavad mõlemad tehnoloogiad erinevaid strateegiaid, nagu andmete replikatsioon ja tõrkekäsitlus.

5 Valitud lahenduse juurutamine

Peatükk käsitleb valitud seirelahenduse juurutamisele Küehne-Nagel SwiftLOG projekti raames. Selle peatüki eesmärk on selgitada, kuidas valitud lahendust rakendatakse ja integreeritakse projektiga, pakkudes ülevaadet erinevatest etappidest ja protsessidest, mis selles protsessis osalevad.

Esiteks käsitletakse peatükis valitud lahenduses kasutatavaid tarkvarakomponente, nagu Docker, Linux-server, Prometheus, Grafana ja BlackBox exporter. Selles osas antakse ülevaade nende komponentide funktsioonidest ja eelistest seirelahenduse jaoks.

Peatükis kirjeldatakse protsessi, mis hõlmab valitud seirelahenduse komponentide paigaldamist ja seadistamist. Selles jaotises käsitletakse erinevaid samme, mis on vajalikud Prometheus'i ja Grafana õigeks tööks seadistamiseks.

Lõpuks keskendutakse valitud lahenduse jõudluse ja efektiivsuse hindamisele. Selles osas uuritakse, kuidas lahendus vastab projekti nõuetele ja eesmärkidele ning kuidas see aitab lahendada seirega seotud probleeme.

5.1 Lahenduses kasutatav tarkvara

Docker, Linux server kus docker on püstitatud. Prometheus docker konteiner, Grafana docker konteiner, BlackBox docker konteiner. Prometheus ja BlackBox exporteri konteiner on ühendatud ühte konteinerisse. Seirelahenduse põhikomponendid hõlmavad Dockerit, Linux-serverit, Prometheus Docker konteinerit, Grafana Docker konteinerit ja ühendatud Prometheus-BlackBox konteinerit.

Linux-server, millele Docker on paigaldatud, toimib seirelahenduse tugipunktina, pakkudes jõudlust, töökindlust ja turvalisust. Linuxi valimine serveri operatsioonisüsteemiks tagab mitmeid eeliseid, nagu madalamad litsentsikulud, avatud lähtekood ja suur hulk kättesaadavaid tööriistu ja ressursse.

BlackBox Docker konteiner töötab koos Prometheusiga, võimaldades lisaseiret ja andmeid seirelahendusele. Prometheus ja BlackBox exporter konteinerite ühendamine

ühte konteinerisse suurendab haldamise lihtsust ja parandab ressursikasutuse tõhusust. Selles lahenduses kasutatakse Prometheus BlackBox exporter spetsiaalselt Nexus *repository* monitoorimiseks.

Grafana Docker konteiner pakub sarnaseid eeliseid nagu Prometheus konteiner, lihtsustades Grafana paigaldamist ja haldamist. See võimaldab projektimeeskonnal keskenduda seirelahenduse kohandamisele ja optimeerimisele, ilma et peaks muretsema Graafana aluseks oleva infrastruktuuri pärast.

5.2 Prometheus ja BlackBox exporter tarkvara paigaldamine ja seadistamine

Joonis 1. näitab Dockeri faili, kus on kirjeldatud Prometheus ja BlackBox exporter konteinerite loomine ning kohaliku masina kataloogide sidumine konteineritega. See tagab vajalike seadistuste ja andmete säilimise ning konteinerite sujuva töö projekti raames.

Dockerfile sisu on järgmine:

```
  GNU nano 2.3.1                    File: Dockerfile

# Filename: Dockerfile
FROM prom/prometheus

ADD prometheus.yml /etc/prometheus/
RUN mkdir -p /etc/prometheus/blackbox/

ADD blackbox/blackbox.sh /etc/prometheus/blackbox/
ADD blackbox/blackbox_exporter /etc/prometheus/blackbox/
ADD blackbox/blackbox.yml /etc/prometheus/blackbox/
ADD blackbox/targets /etc/prometheus/blackbox/

ENTRYPOINT ["sh", "/etc/prometheus/blackbox/blackbox.sh"]
```

Joonis 1. Faili Dockerfile sisu.

See Dockerfile kirjeldab Prometheus ja BlackBox exporter konteinerite loomist. Fail sisaldab juhiseid tarkvara komponentide lisamiseks konteinerisse, kataloogide loomiseks ja sidumiseks ning konteineri käivitamiseks vajaliku käivituspunkti määratlemiseks.

Pärast Dockerfile loomist saab luua Docker *image*, kasutades järgmist käsku:

```
docker build -t prometheus_blackbox .
```

Käsk loob Docker *image* nimega "prometheus_blackbox", kasutades praeguses kaustas olevat Dockerfile'i, mis sisaldab määratletud konfiguratsioone.

Kui Docker *image* on loodud, siis saab Docker konteineri käivitada käskuga:

```
docker run --network host --name prometheus_blackbox -d --volume
/home/influx_test/PrometheusBlackBox /AlenPrometheusTest:/etc/prometheus/ -p
9090:9090 -p 9115:9115 prometheus_blackbox
```

See käsk käivitab Dockeri konteineri järgmiste parameetritega:

- **--network host**: kasutab hosti võrguühendust konteineri jaoks.
- **--name prometheus_blackbox**: määrab konteinerile nimeks "prometheus_blackbox".
- **-d**: käivitab konteineri taustal.
- **--volume /home/influx_test/PrometheusBlackBox/AlenPrometheusTest:/etc/prometheus/**: kaardistab kohaliku kausta **/home/influx_test/PrometheusBlackBox/AlenPrometheusTest** konteineri kausta **/etc/prometheus/**.
- **-p 9090:9090**: edastab konteineri pordi 9090 kohalikku pordi 9090.
- **-p 9115:9115**: edastab konteineri pordi 9115 kohalikku pordi 9115.
- **prometheus_blackbox**: kasutab "prometheus_blackbox" *image* konteineri loomiseks.

Järgnevalt toimub paigaldusprotsess, mille käigus Dockeri konteinerid luuakse ja käivitatakse vastavalt Dockerfile'is kirjeldatud seadistustele. Seejärel seadistatakse Prometheus ja BlackBox exporter.

Joonis 2. kujutab faili "http_2xx.yml", mis on Prometheus BlackBox Exporteri konfiguratsioon. Selles failis määratletakse mooduli nime ja sihtmärgid (Nexus *repository* lingid).

Joonis 2. Faili http_2xx.yml sisu.

Joonis 3. kujutab faili "blackbox.yml", mis kasutab eelnevalt loodud "http_2xx" moodulit. Selles konfiguratsioonis on määratletud, et BlackBox Exporter kasutab HTTP-päringuid Nexus *repository* sihtmärgi kontrollimiseks. Konfiguratsioonifailis on seatud järgmised parameetrid:

Joonis 3. Faili blackbox.yml sisu.

- Eksporter kasutab HTTP-päringuid, mis võimaldab tal jälgida sihtmärki, pöördudes selle veebiteenuse poole.

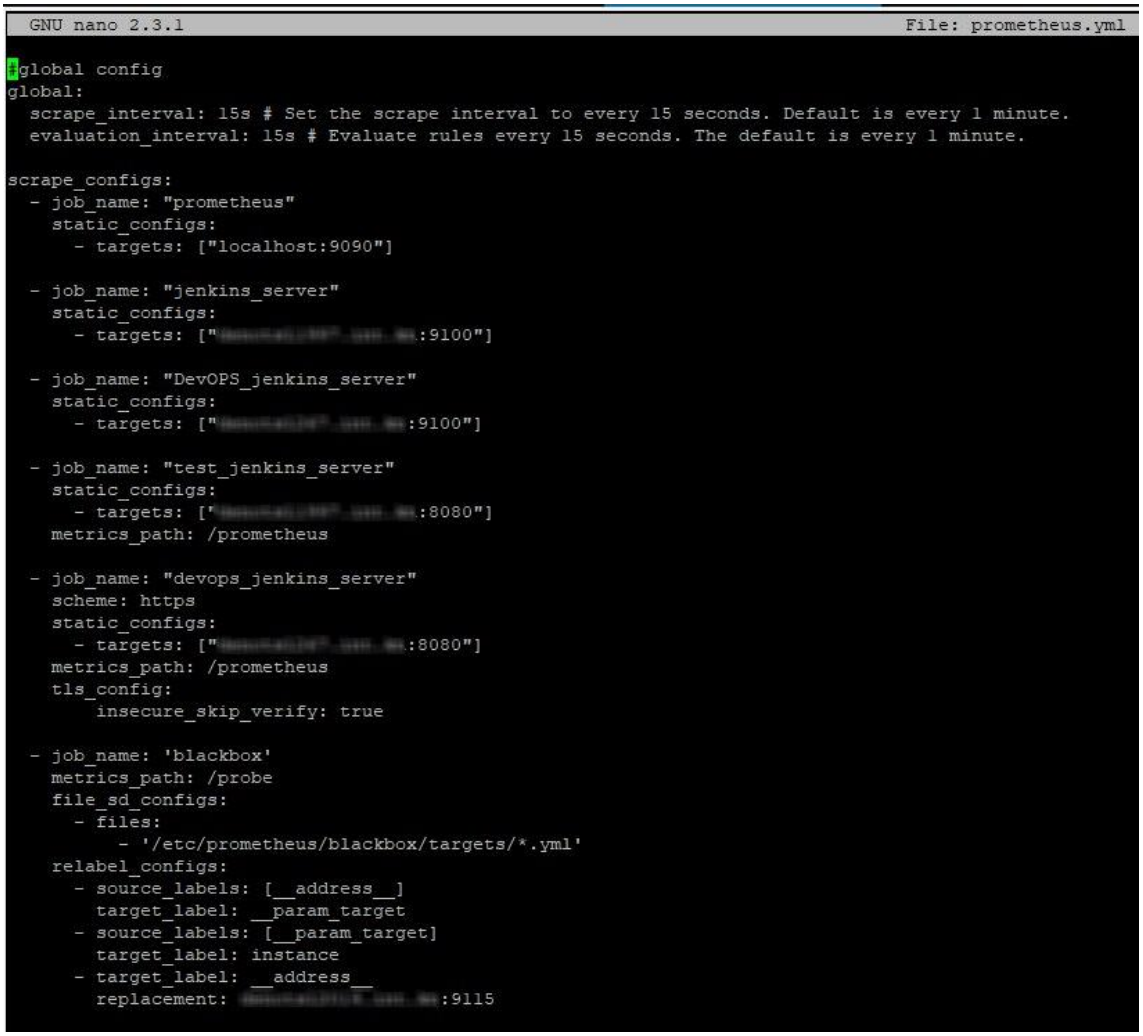
- HTTP-päringute meetodina on valitud GET. See tähendab, et eksporter küsib sihtmärgilt teavet, kasutades standardset HTTP GET-päringut.

Kui kõik seadistused on valmis, saab konteineri taaskäivitada, et laadida uued konfiguratsioonid ja lõpuks koguda Nexus *repository* meetmeid. Lisa 1, 2 ja 3 näitab Nexus *repository'*st saadud meetmete näited.

Selles protsessis konteineri taaskäivitamine võimaldab uute konfiguratsioonidega töötada ning Nexus *repository* olekut jälgida. Tulemuseks on kogutud meetmed, mis kajastavad Nexus repositooriumi jõudlust ja tööd.

Kui kogu andmete kogumine toimub pidevalt, saab seadistada Prometheus'i. Selleks tuleb luua konfiguratsioonifail (Joonis 4.), milles määratakse intervall, millega Prometheus võtab kogutud andmeid BlackBox'ist. Samuti tuleb määrata töö nimed, et need kuvataks Prometheus'is. Töö nimi on "BlackBox", kus on määratletud asukoht ja kus kogutud meetmed asuvad.

Selles etapis on oluline määrata õige konfiguratsioon, et tagada BlackBox'i ja Prometheus'i korrektne andmete edastamine ja kuvamine. Töö nimede määratlemine aitab jälgida erinevaid meetmeid ja neid Prometheus'is hõlpsamini hallata.

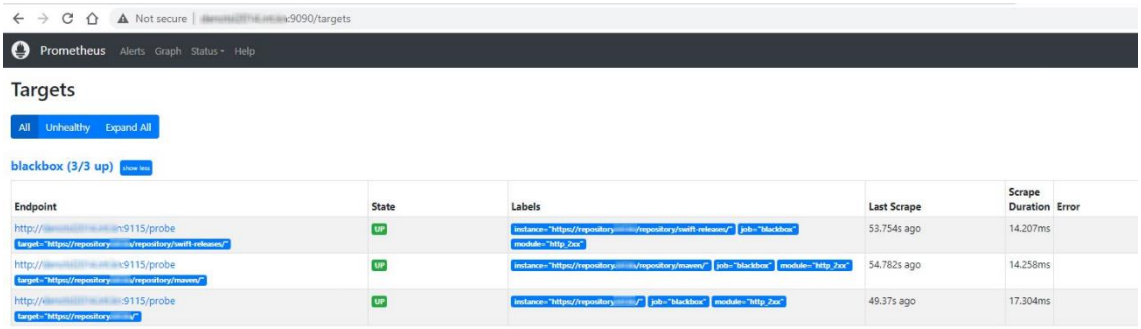


Joonis 4. Faili prometheus.yml sisu.

Pärast kõigi konfiguratsioonietappide lõpetamist taaskäivitatakse konteineri uuesti, et kontrollida, kas kõik töötab nagu peab. Kontrollimiseks navigeeritakse veebilehele (IP:9090). Port 9090 on vaikimisi port, kus Prometheus töötab. Joonis 5. näitab, et Prometheus töötab korrektselt.

Joonis 5. Prometheuse algusleht.

Joonis 6. näitab, et BlackBox *Exporter*'i kogutud meetmed on Prometheus jaoks edukalt kättesaadavad. Sihtmärkide olekut saab nüüd jälgida, näidates, et need on aktiivsed ja töötavad korrektselt. See võimaldab süsteemi jõudlust jälgida ja tagada, et Nexus'i *repository* töötab korrektselt.



Joonis 6. Prometheuse sihtmärkide ekraan koos BlackBox Exporteri kogutud meetmetega.

Kui kõik meetmed on edukalt kogutud ja Prometheus jälgib neid, on aeg jätkata Grafana seadistamisega, mis võimaldab luua visuaalseid graafikuid ja juhtpaneeli, et jälgida süsteemi seisundit ja hoiatada võimalike probleemide korral. See muudab süsteemi järelevalve tõhusamaks ja aitab kiiremini tuvastada ja lahendada Jenkins'i ja Nexus'i *repository* seotud probleeme.

5.3 Grafana tarkvara paigaldamine ja seadistamine

Grafana paigaldamiseks ja seadistamiseks Docker'i abil tehakse neid samme:

1. Tõmmakse Gragana Docker'i *image* kasutades seda käsku:

```
docker pull grafana/grafana
```

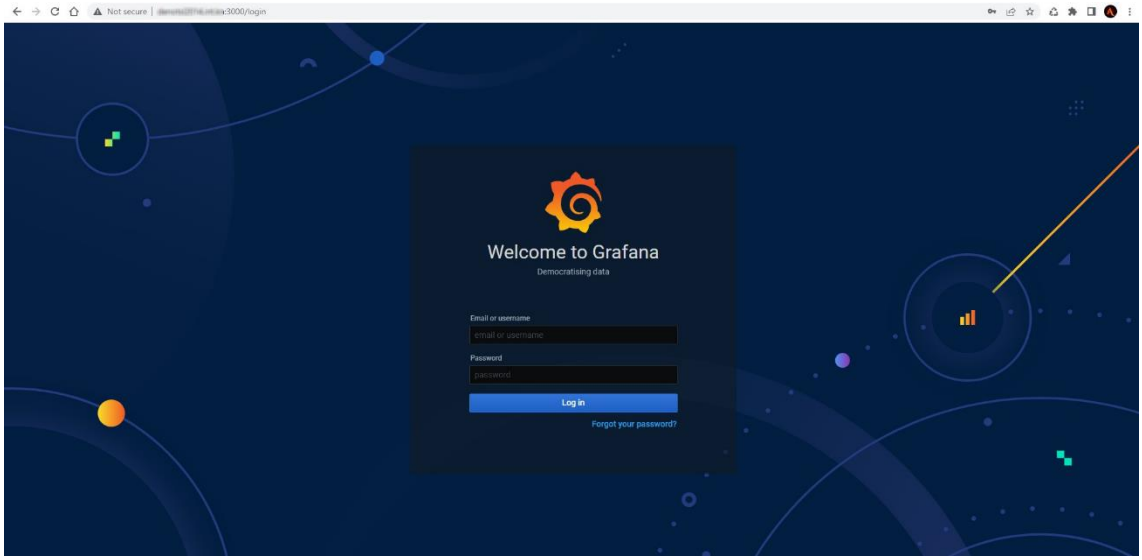
2. Käivitatakse Grafana Dockeri konteiner käskuga:

```
docker run -d -p 3000:3000 --name grafana -e
"GF_SECURITY_ADMIN_PASSWORD=mysecurepassword" grafana/grafana
```

See käsk käivitab Dockeri konteineri järgmiste parameetritega:

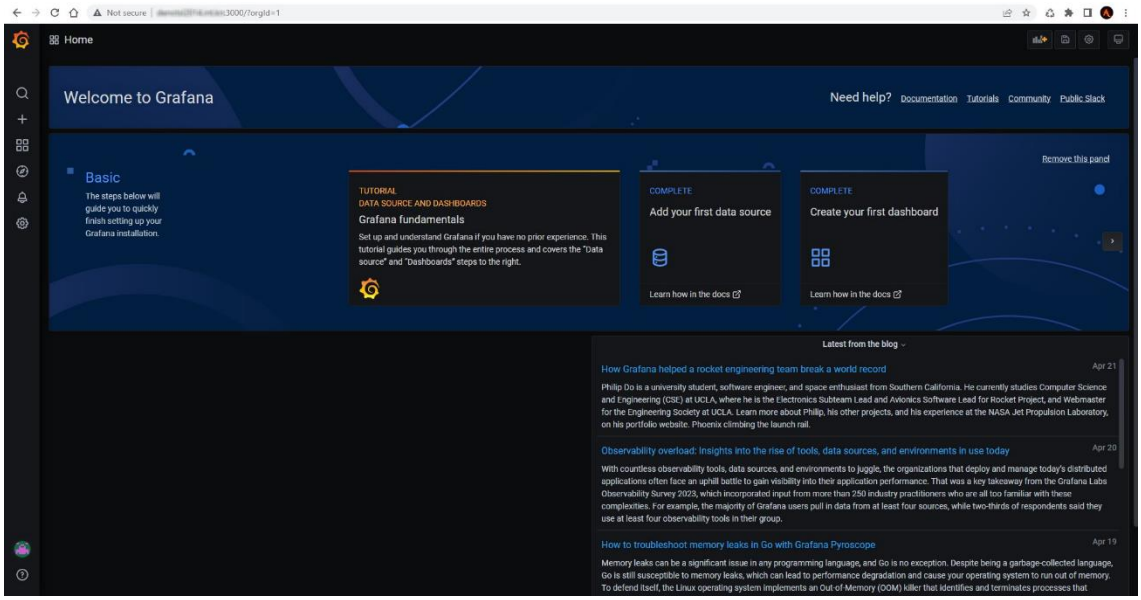
- **-d** käivitab konteineri taustal eraldi protsessina
- **-p 3000:3000** seob kohaliku masina pordi 3000 konteineri pordiga 3000
- **--name grafana** määrab konteinerile nimeks "grafana"

3. Pärast konteineri käivitamist saab veebibrauseris avada aadress http://ip:3000 ja sisse logida kasutades vaikimisi seadistatud kasutajani ja parooliga. Vaikimisi Grafana Kasutajanimi ja Parool on **admin** (Joonis 7.).



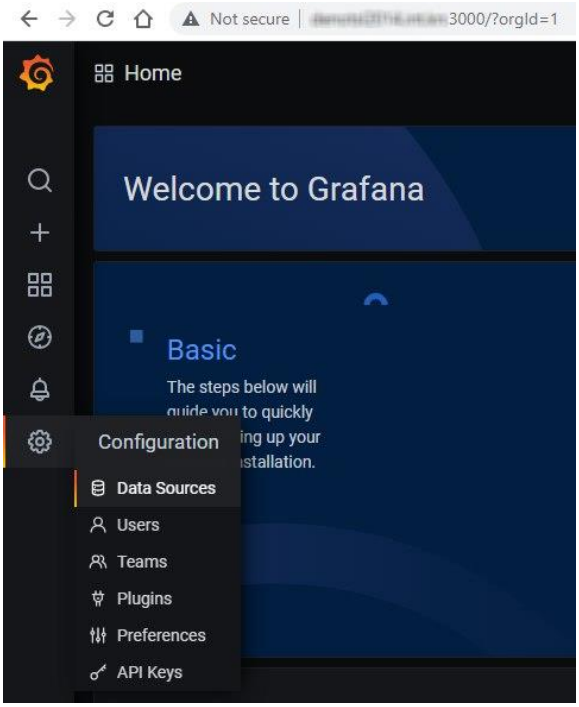
Joonis 7. Grafana sisselogimise algusleht.

Pärast edukat sisselogimist saab näha Tervituslehe (Joonis 8.) kus on kirjeldatud peamised tegevused mida peaks tegema. Näiteks lisada admeallika (Data Source) ja seadistada esimene seire vaade (*Dashboard*).



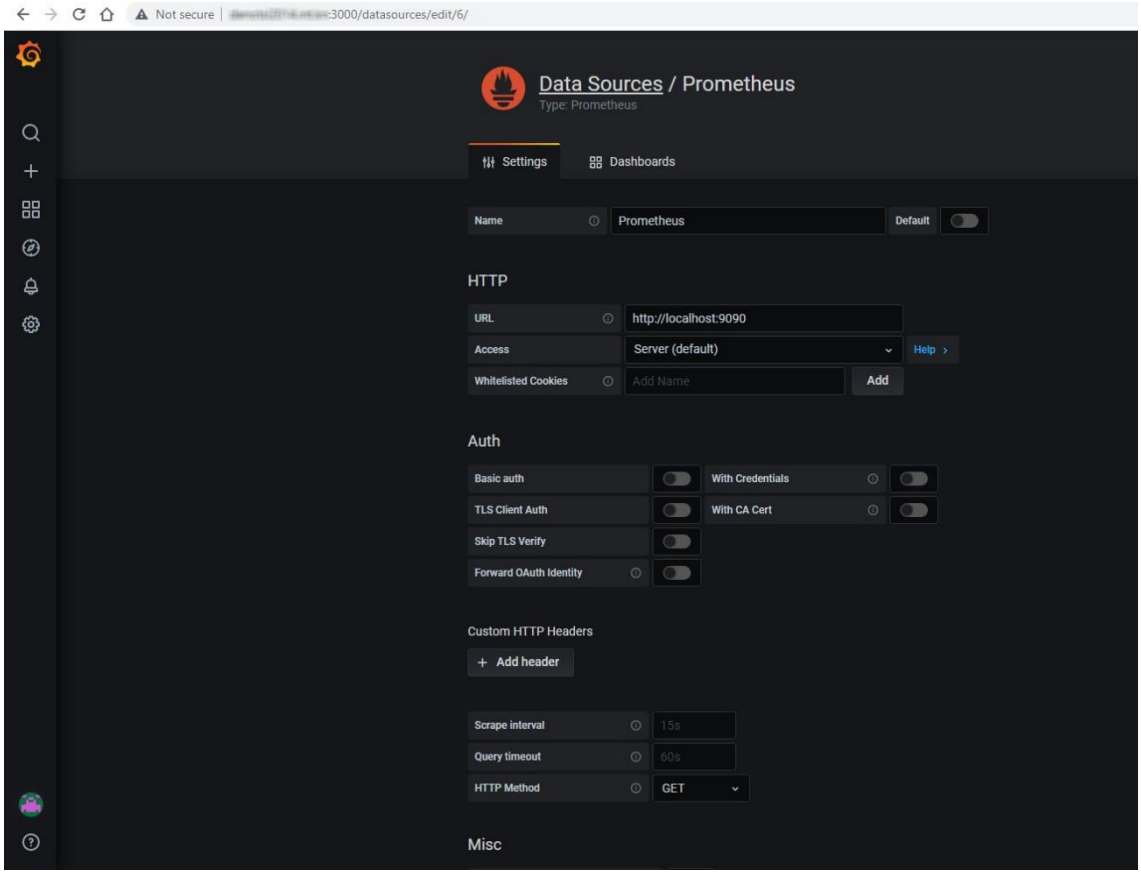
Joonis 8. Grafana algusleht.

Uue admeallika saab lisada vajutades nuppu konfiguratsioon ja seal nupp *Data Sources* (Joonis 9.)



Joonis 9. Andmeallika lisamine.

Vajutades nupule "*Add data source*" ja valides loendist "Prometheus", avaneb uus aken, kus saab seadistada Prometheus andmeallika (Joonis 10.). Siin tuleb seadistada andmeallika nimi ja URL, kus Prometheus on püstitatud.



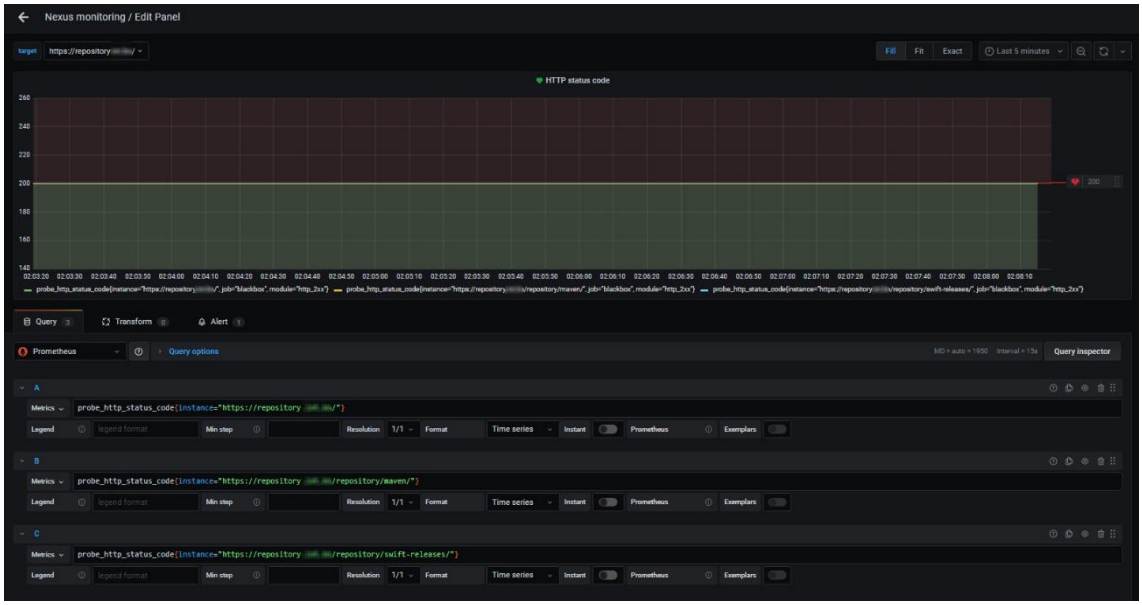
Joonis 10. Prometheuse andmeallika seadistamine.

Selleks, et kontrollida, kas andmed on õigesti sisestatud ja andmeallikas töötab korrektselt, tuleb vajutada nuppu "Save & Test". Kui ilmub roheline teavitus "Data source is working" (Joonis 11.), siis võib alustada seire vaade (*Dashboard*) püstitamisega.

Joonis 11. Adnmeallika kontrollimine.

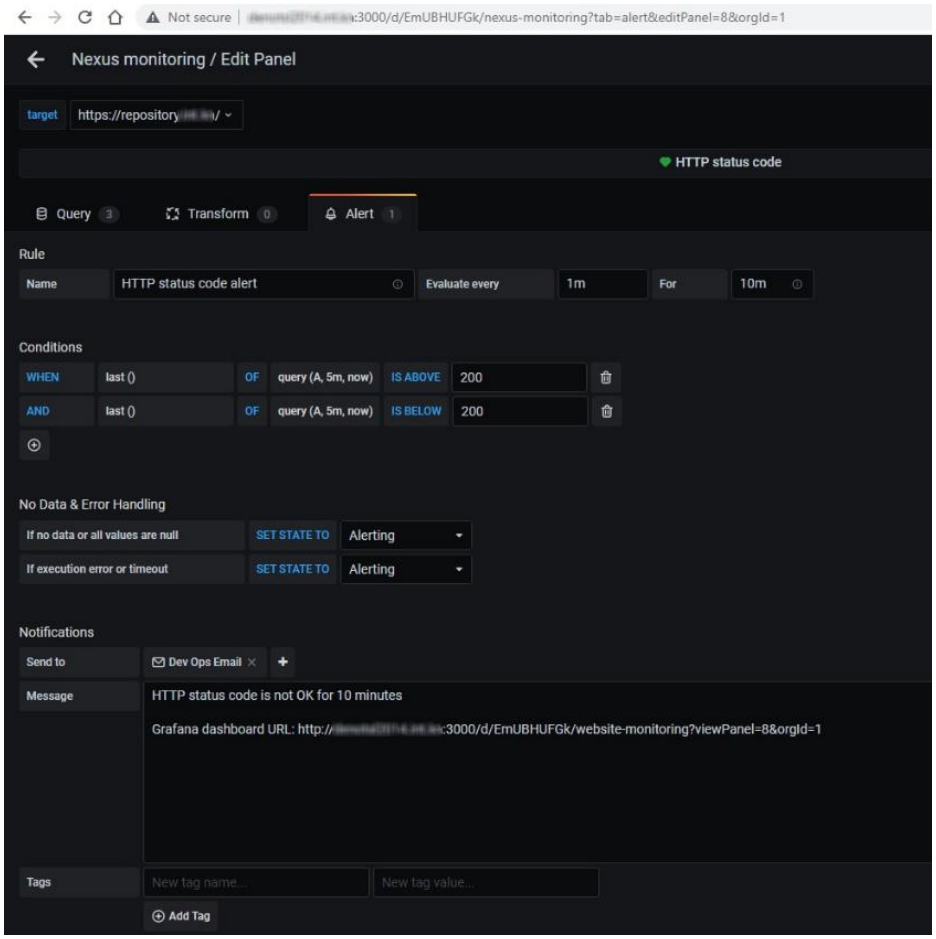
5.4 Grafana seire vaade seadistamine Nexus repository jaoks

Grafana seire vaade seadistamiseks luuakse uue seire vaadet vajutades *"Create"* -> *"Dashboard"*. Kasutades andmeid, mida Prometheus kogub BlackBox exporterist, tehakse esimese päringu, mis kuvab HTTP staatuse kolmest repositooriumist. Staatus 200 näitab, et HTTP-päringud on korrektsed ja Nexus repositooriumid töötavad (Joonis 12.).



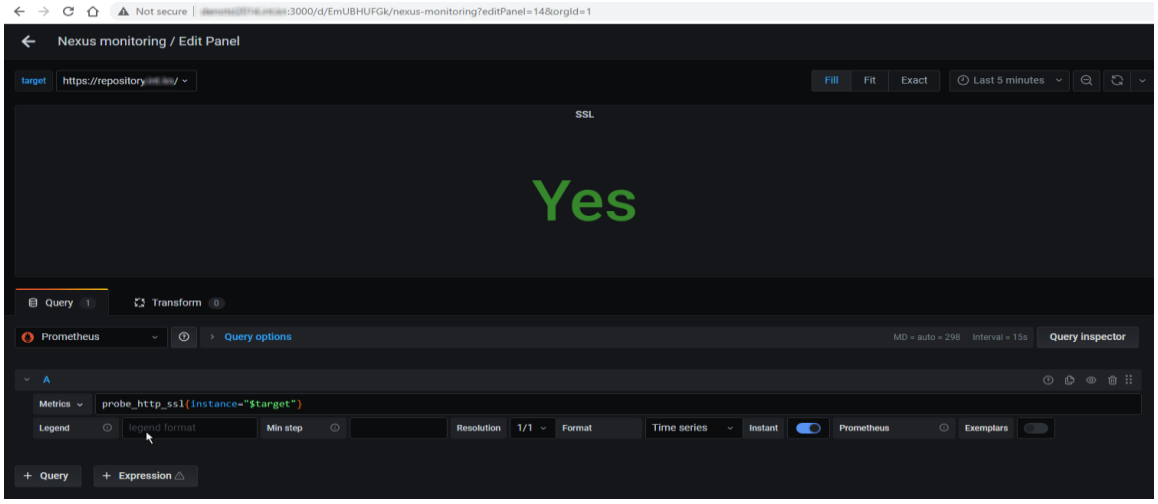
Joonis 12. Nexus repository HTTP päringu staatuse paneel.

Lisaks sellele paneelile lisatakse ka teavitussüsteemi, mis annab teada, kui serveril on teine staatuskood. Sellisel juhul saadetakse teavitus DevOps tiimile e-posti teel (Joonis 13.).



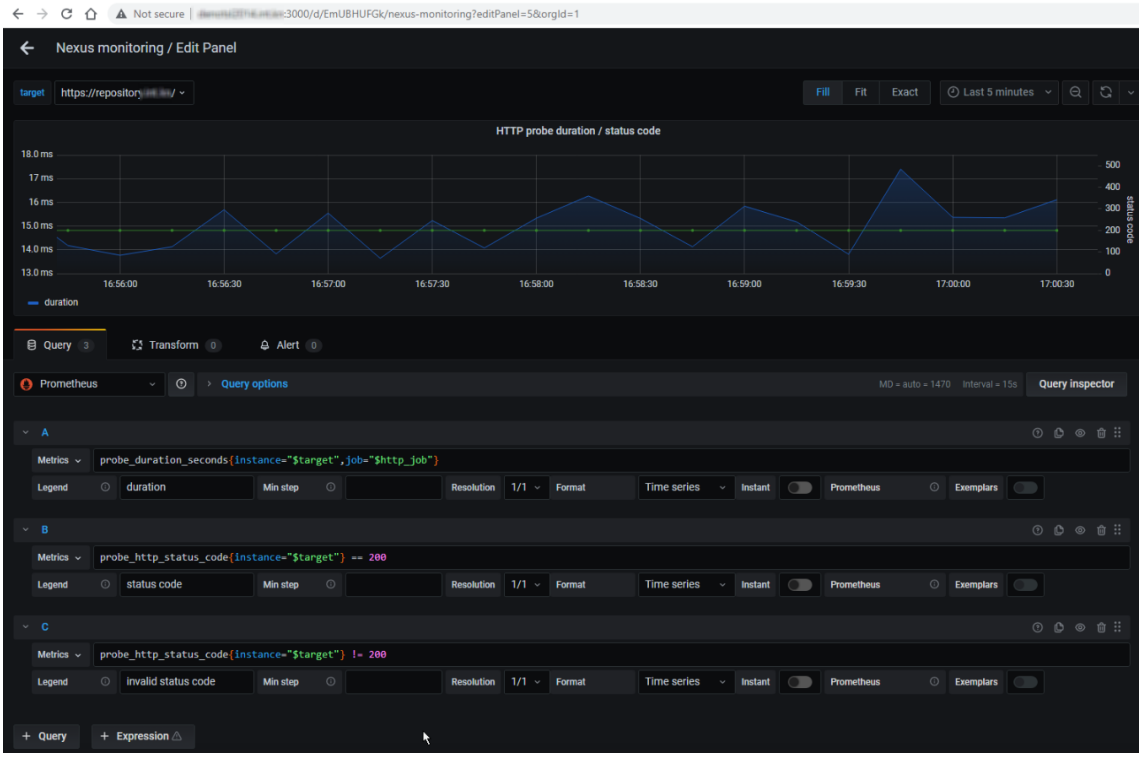
Joonis 13. Teavitussüsteemi lisamine HTTP staatuse paneeli peale.

Grafana paneel kasutab päringut *probe_http_ssl*, et kontrollida SSL-i (Secure Sockets Layer) toimimist. See päring saadab HTTPS-ühenduse, et kontrollida, kas see on õigesti seadistatud ja töötab korrektselt. Joonis 14. kuvab tulemused, mis näitavad, kas SSL töötab või mitte, võimaldades seeläbi hõlpsalt jälgida süsteemi turvalisust.



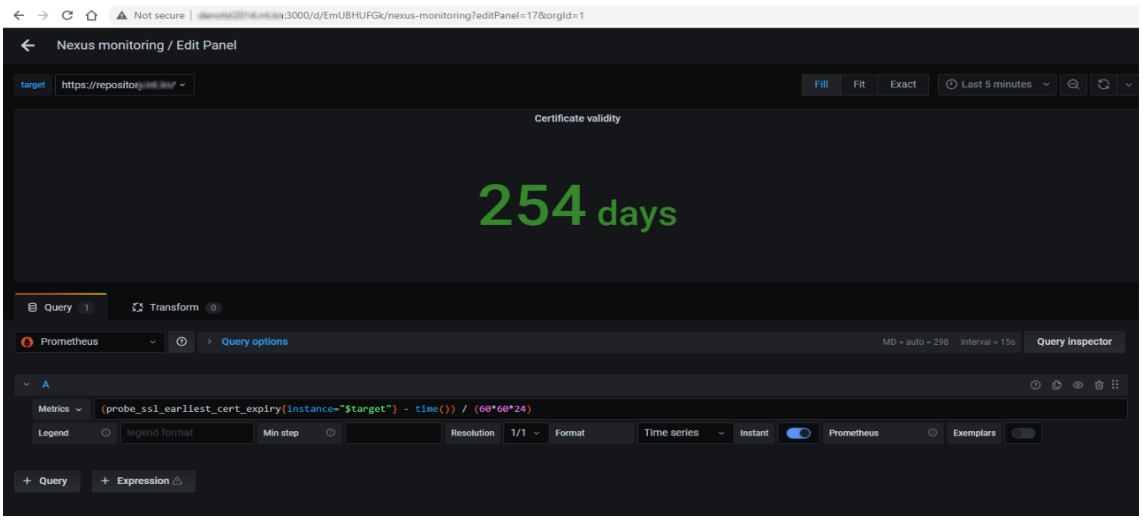
Joonis 14. Grafana Nexus repository SSL staatuse paneel.

Joonis 15. kuvab paneeli, mis kasutab päringut *probe_duration_seconds*. See paneel aitab jälgida Nexus serverite kättesaadavuse aega ja nende vastuseid HTTP-päringutele. Päring mõõdab aega, mis kulub vastava teenuse või serveri pingimiseks ning seejärel kuvab selle graafikul sekundites.



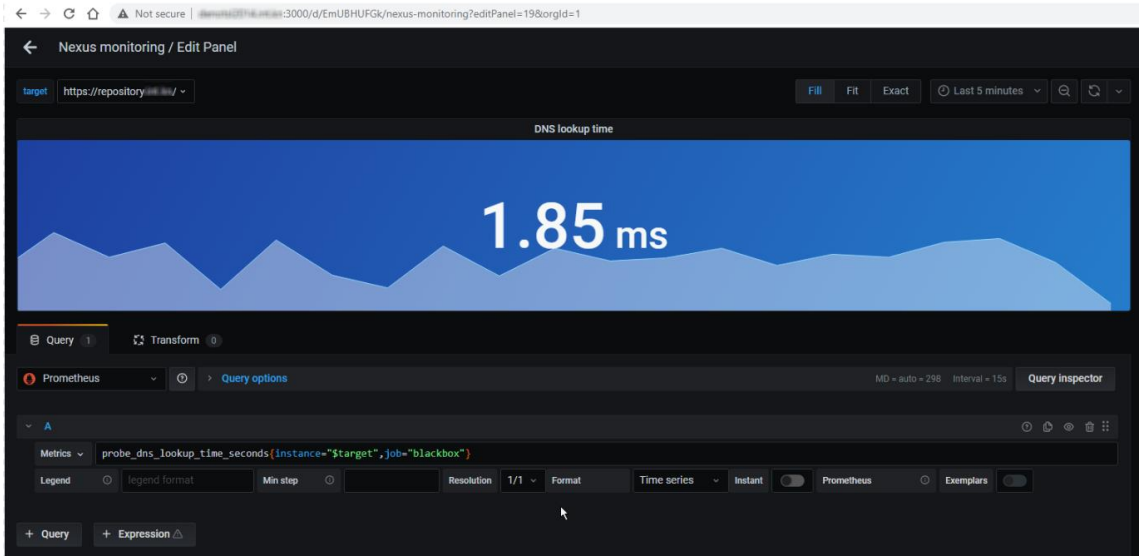
Joonis 15. Grafana Nexus repository HTTP päringute vastuse aja paneel.

Joonis 16. kuvab paneeli, mis kasutab päringut *probe_ssl_earliest_cert_expiry*. See paneel aitab jälgida SSL-sertifikaatide kehtivusaegu. Päring kontrollib, millal sertifikaadid aeguvad ja kuvab selle graafikul.



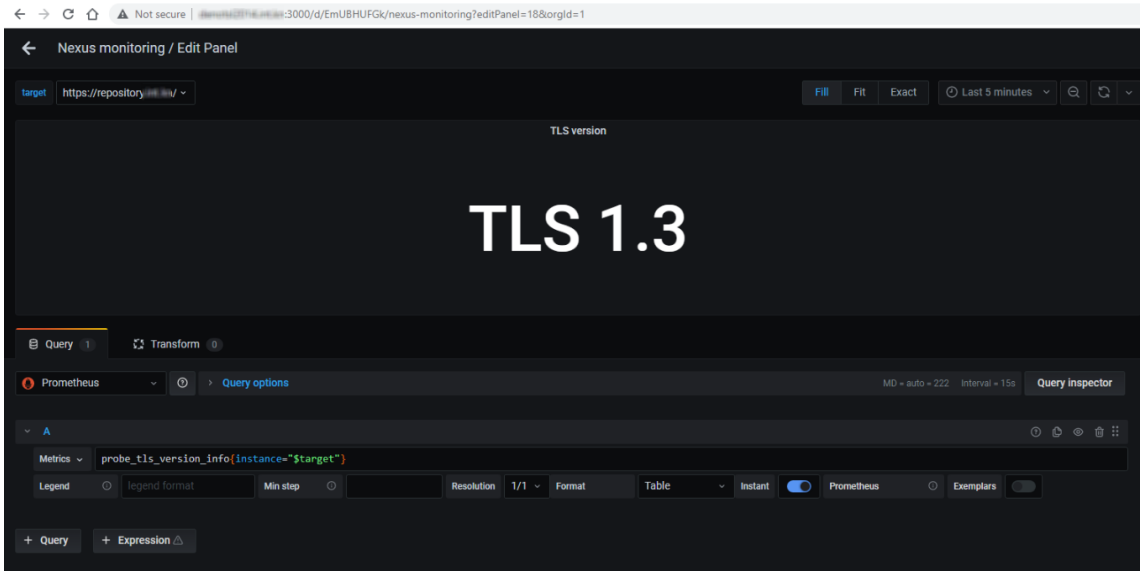
Joonis 16. Grafana Nexus repository HTTP sertifikaati kehtivusaja paneel.

Joonis 17. kuvab paneeli, mis kasutab päringut *probe_dns_lookup_time_seconds*. See paneel jälgib DNS-päringute aega ja nende vastuseid. Päring saadab DNS-päringu, et kontrollida, kui kiiresti vastus saadakse. See paneel on oluline, kuna DNS-päringute aeg võib mõjutada veebisaitide laadimise kiirust ja üldist tulemust.



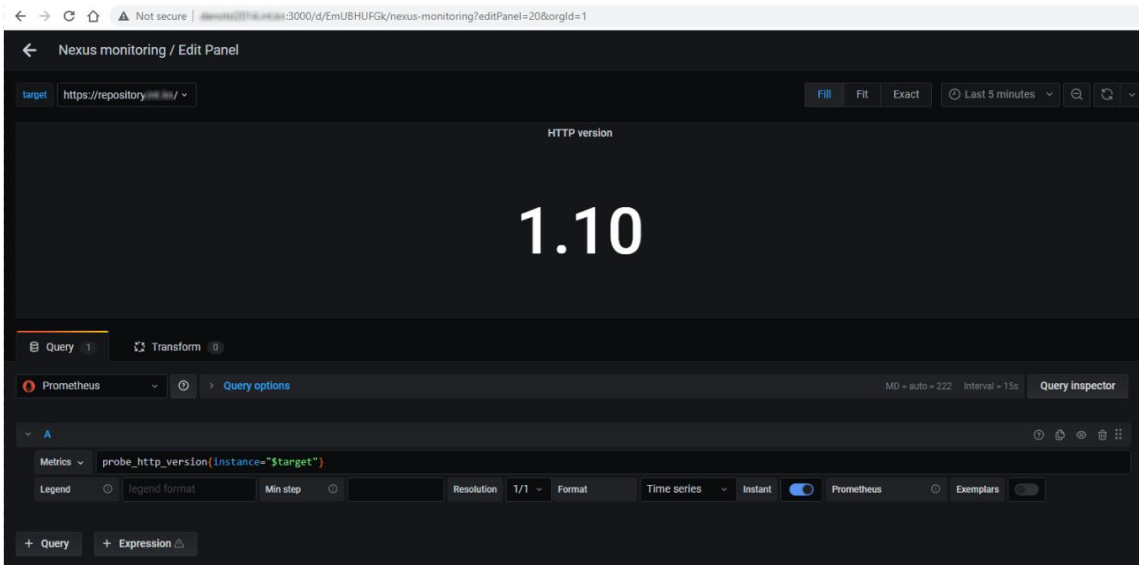
Joonis 17. Grafana Nexus repository DNS-päringute aja ja nende vastuste paneel.

Joonis 18. kuvab paneeli, mis kasutab päringut *probe_tls_version_info*. See paneel jälgib SSL/TLS versioone, mida kasutatakse HTTPS-ühenduse loomiseks. Paneel aitab tagada süsteemi turvalisust, tuvastades SSL/TLS versioonide vananemise ja vajaduse nende uuendamiseks vastavalt turvanõuetele.



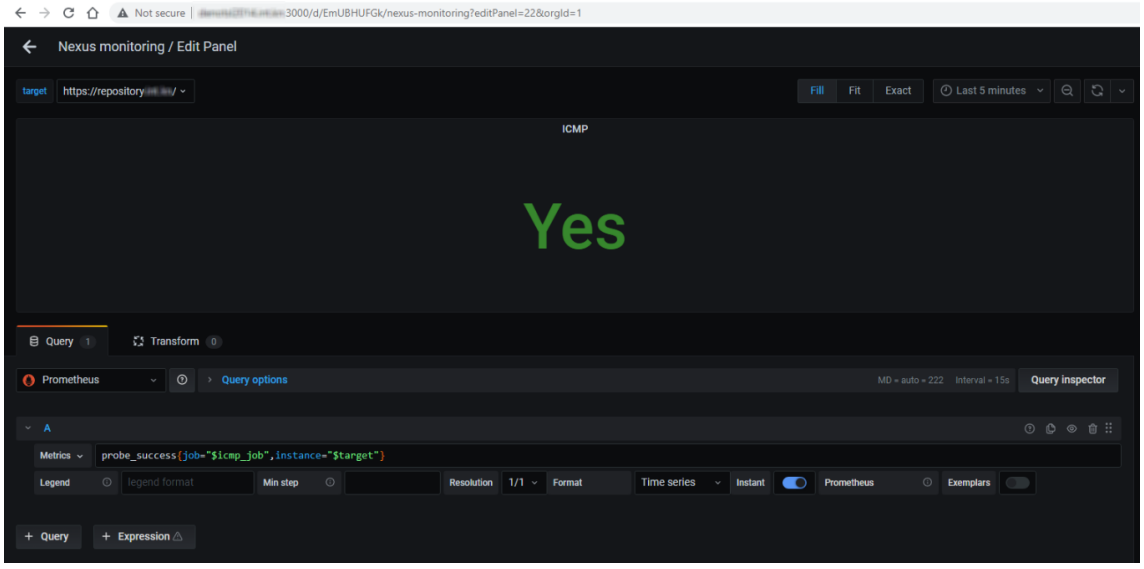
Joonis 18. Nexus repository TLS vesioon.

Joonis 19. kuvab paneeli, mis kasutab päringut *probe_http_version*. Paneel jälgib HTTP-protokolli versiooni.



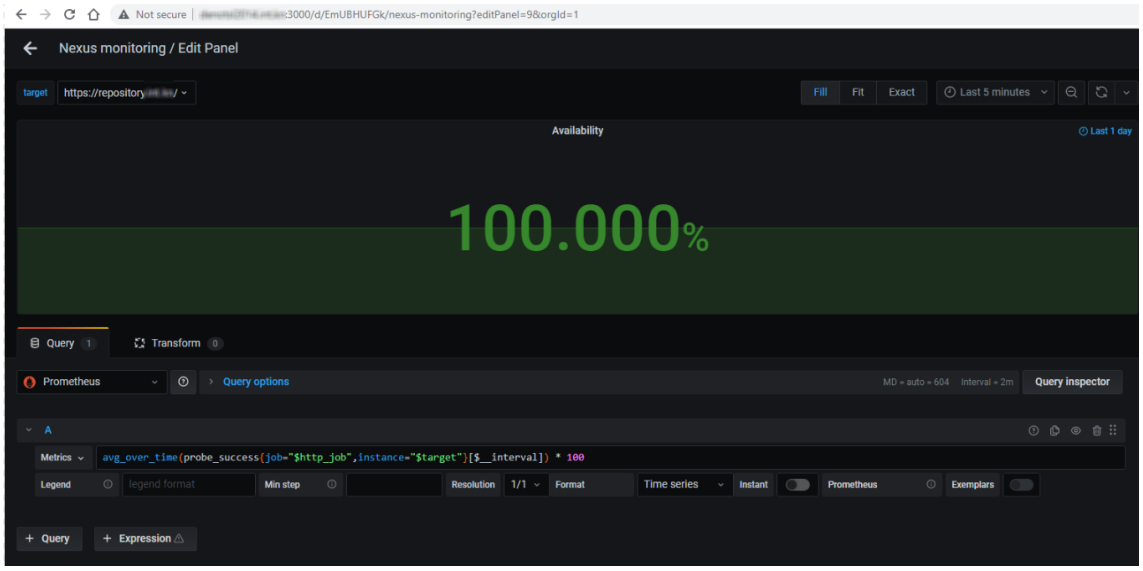
Joonis 19. Nexus repository HTTP versioon.

Joonis 20. kuvab paneeli, mis näitab ICMP päringute edukust. Päring saadab ICMP-paketi, et kontrollida, kas Nexus *repository* server on kättesaadav või mitte, ning seejärel kuvab selle graafikul. See paneel on oluline, kuna ICMP-päringud on sageli kasutatavad võrguprobleemide tuvastamiseks ja võrgu tervise jälgimiseks.



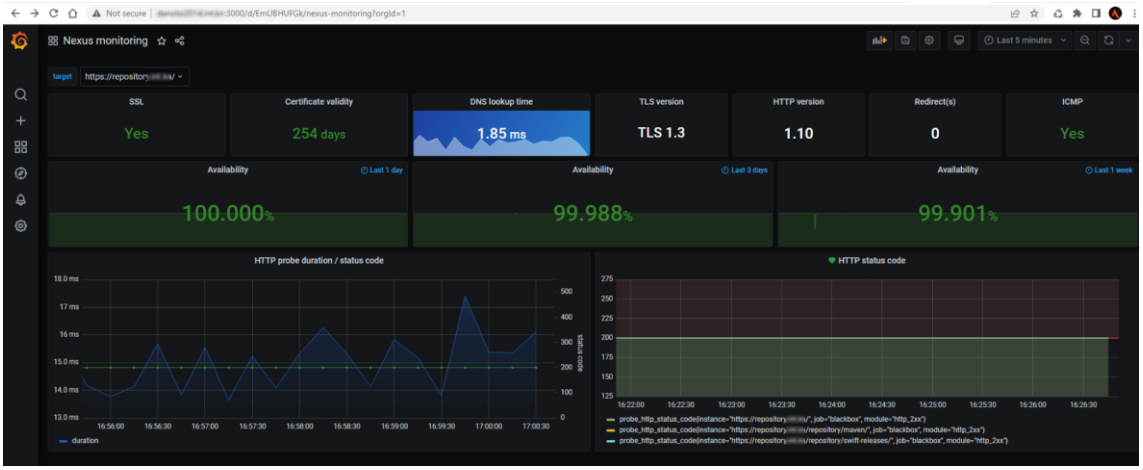
Joonis 20. Nexus repository ICMP päringute paneel.

Joonis 21. kuvab paneeli, mis kasutab funktsiooni *avg_over_time()* koos prometheus päringuga *probe_success{job="$http_job",instance="$target"}[$__interval]) * 100*. See paneel näitab Nexus'i kättesaadavuse keskmist määra protsentides ühe päeva jooksul. Päring kontrollib, milliseid HTTP-päringuid tehti Nexus'i suhtes ja milline oli nende edukuse määr. Funktsioon *avg_over_time()* arvutab selle keskmise ajavahemiku jooksul ja seejärel kuvab selle protsentides graafikul.



Joonis 21. Nexus serveri kättesaavadus ühe päeva jooksul.

Joonis 22. kujutab endast tervet seire vaadet, kus on kõik erinevad paneelid ühes ekraanis. See võimaldab projekti liikmetel jälgida kõiki olulisi graafikuid korraga ning tuvastada kiiresti võimalikke probleeme.



Joonis 22. Nexus seiresüsteemi täisvaade kõikide paneelidega.

5.5 Node exporter paigaldamine ja seadistamine

Node Exporter [13] on vajalik süsteemi olekuandmete edastamiseks. See on Prometheus'i *exporter*, mis kogub teavet serveri koormuse, mälu kasutamise, võrguühenduse, kettakasutuse ja paljude teiste süsteemi oleku näitajate kohta. Node Exporter'it kasutatakse süsteemi jõudluse jälgimiseks ja optimeerimiseks, mis võimaldab kiiresti tuvastada ja lahendada võimalikke probleeme.

Node Exporter'i paigaldamine:

1. Laadida Node Exporter alla GitHub repositooriumist käskuga:

   ```
   wget
   https://github.com/prometheus/node_exporter/releases/download/v1.3.1/n
   ode_exporter-1.3.1.linux-amd64.tar.gz
   ```

2. Lahti arhiveeritakse allalaaditud tar.gz arhiiv järgmise käsu abil:

   ```
   sudo tar -xzvf node_exporter-1.3.1.linux-amd64.tar.gz
   ```

   -xzvf on **tar** käsurea programmi lipud, mis määravad arhiveerimise või lahtipakkimise käitumise. Siin on nende lipude tähendused:

   - **-x** - see lipp ütleb programmile, et arhiivist tuleb faile välja pakkida.
   - **-z** - kasutab gzip'i tihendust arhiivi lahtipakkimiseks või pakendamiseks.
   - **-v** - lipp "verbose" mode'iga, mis annab rohkem teavet käsu käivitamise kohta. See näitab ekraanil pakendatud või lahtipakitud failide nimekirja.
   - **-f** - see lipp näitab, et järgmine argument on arhiivifaili nimi, mida kasutatakse arhiveerimiseks või lahtipakkimiseks (antud juhul **node_exporter-1.3.1.linux-amd64.tar.gz**).

3. Lahtipakitud teenusfaili liigutamine:

   Selle käskuga liigutakse teenusfaili kausta */usr/local/bin*

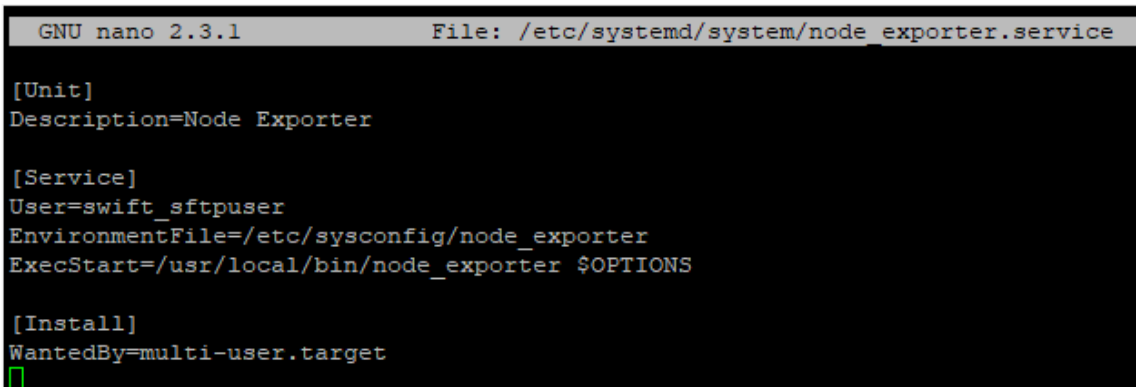
   ```
   sudo mv node_exporter-1.3.1.linux-amd64/node_exporter /usr/local/bin
   ```

4. Node Exporter'i teenusfaili loomine */etc/systemd/system/* kaustas:

Käskuga luuakse teenusfail mis asub */etc/system/system/* ja selle nimi on *node_exporter.service*.

```
sudo nano /etc/systemd/system/node_exporter.service
```

Loodud teenusfaili sisu kus käsitletakse mis saab teenuse nimeks, millise kasutaja nimelt seda teenust jooksutakse ja kõige olulisem on vaja õigesti määrada lahtipakitud faili asukoha (Joonis 23.)



Joonis 23. Node Exporter service fail.

5. Laaditakse süsteemi deemonid uuesti, kasutades käsku:

```
sudo systemctl daemon-reload
```

6. Loatakse Node Exporter'i teenus, kasutades käsku:

```
sudo systemctl enable node_exporter.service
```

7. Käivitatakse Node Exporter'i teenus, kasutades käsku:

```
sudo systemctl start node_exporter.service
```

8. Kontrollitakse Node Exporter'i teenuse olekut, kasutades käsku:

```
sudo systemctl status node_exporter.service
```

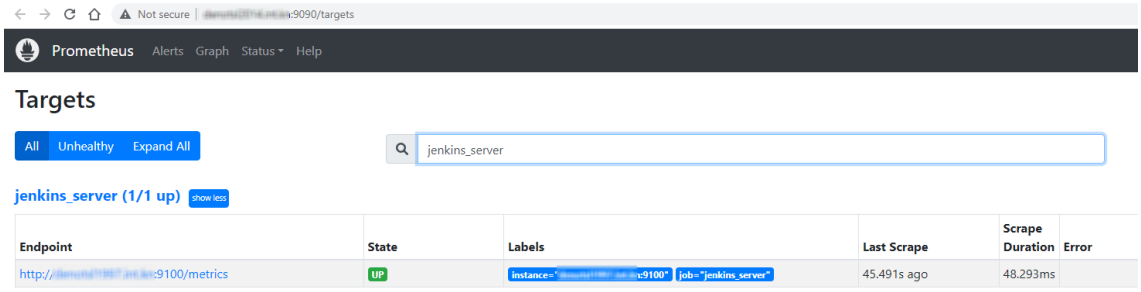
Kui Node Exporter on edukalt paigaldatud, saab vaadata milliseid meetmeid ta süsteemist kogub, külastades veebilehte *http://ip:9100/metrics* (Lisa 5.). Port 9100 on Node Exporter'i vaikimisi port.

5.6 Jenkins serveri Grafana vaade seadistamine

Esmalt tuleb lisada Node Exporter Prometheus'ele, et see saaks lugeda meetmeid Node Exporter'ist. Seda saab teha, muutes faili prometheus.yml, lisades Node Exporteri töö nime ja URL-i (Joonis 24.).

Joonis 24. Node Exporteri lisamine prometheus konfiguratsiooni.

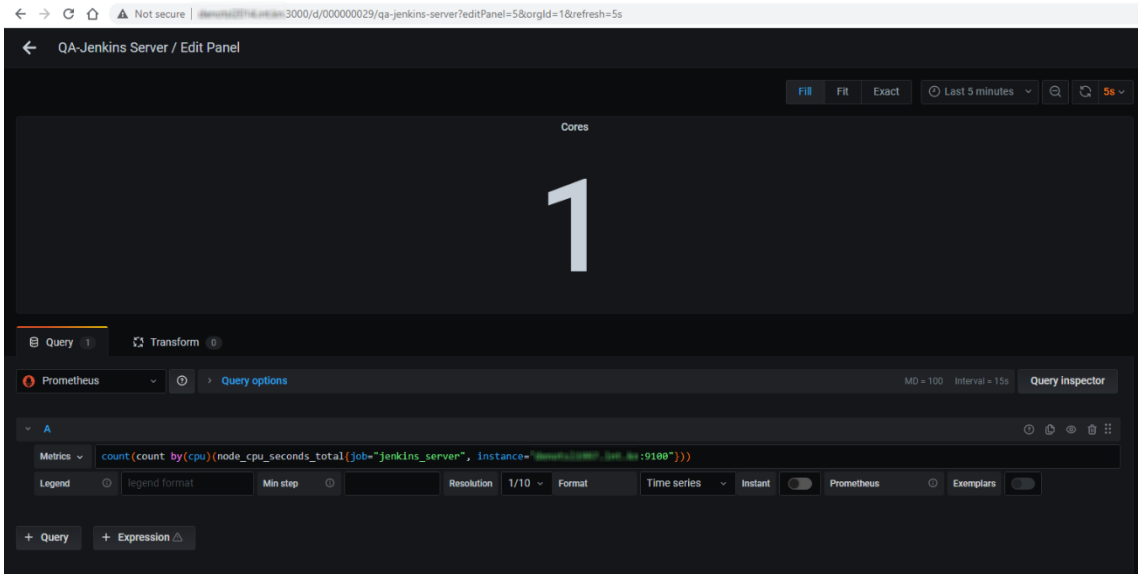
Pärast seda, kui on kontrollinud, et Prometheus suudab edukalt lugeda Node Exporter'i andmeid (Joonis 25.), on aeg luua Grafana's esimene juhtpaneel. Nimetakse selle QA-Jenkins Serveriks, kuna see konkreetne Jenkins'i serverit kasutab QA meeskond SwiftLOG projektis.



Joonis 25. Jenkins serveri Node Exporter meetmed.

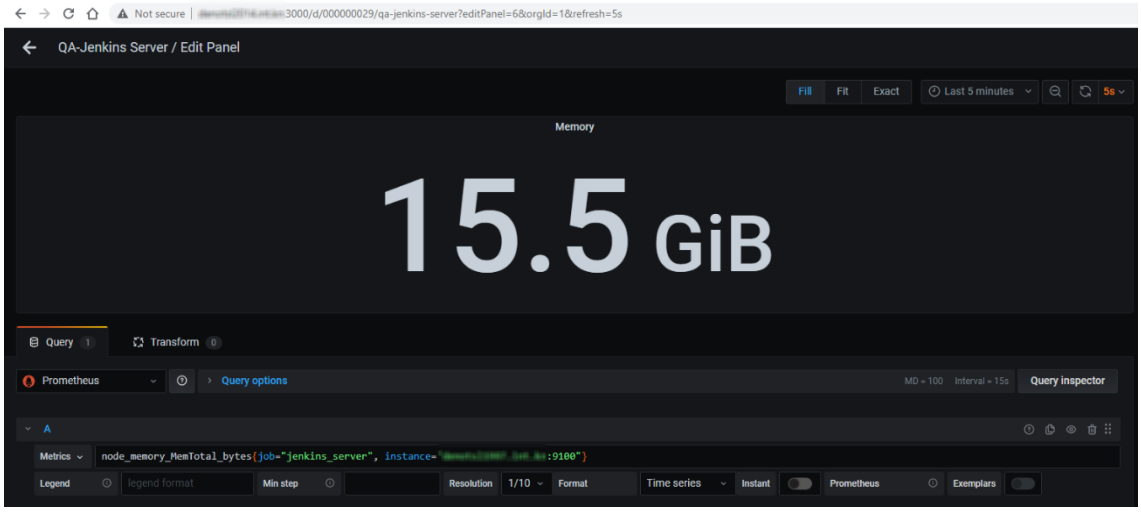
Jenkins'i serveri seirevaate loomiseks Grafana's tuleb esmalt kasutada Prometheus andmeallikas mis oli juba varem tehtud, mis põhineb Node Exporter'i andmetel. Seejärel saab luua erinevaid paneele, mis kuvavad Jenkins'i serveri erinevaid meetrikaid, nagu protsessori kasutamine, mälu kasutamine ja võrguliiklus. Need andmed aitavad meeskonnal hinnata serveri jõudlust ja tuvastada võimalikke kitsaskohti, mis võivad mõjutada projekt SwiftLOG sujuvat toimimist.

Joonis 26. kuvab graafiku mis näitab reaalajas kui palju protsessori tuuma on hetkel kasutusel Jenkins serveri peal.



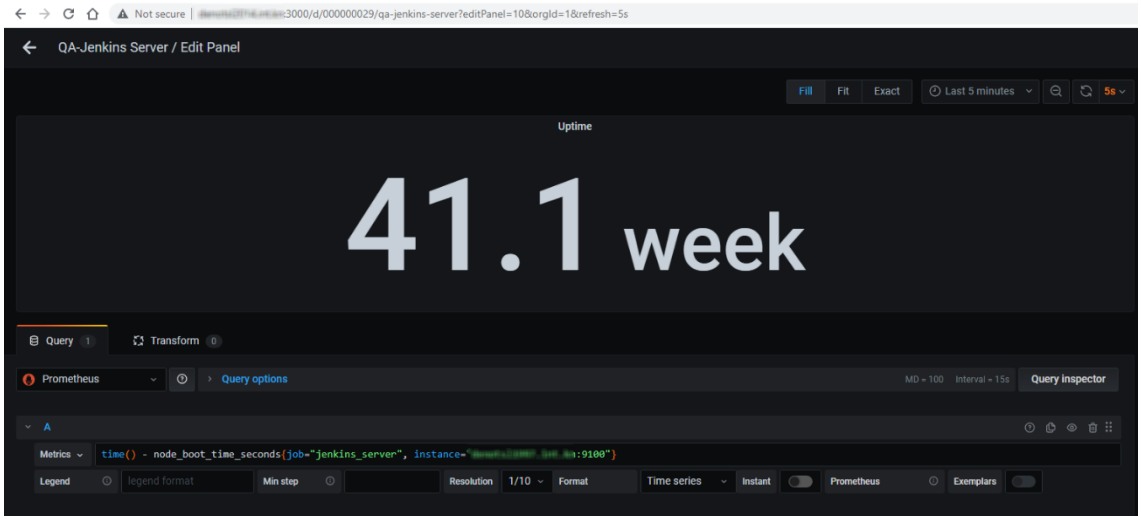
Joonis 26. Protsessori tuumade kasutus.

Joonis 27. kuvab graafiku mis näitab operatiivmälu suurust serveris.



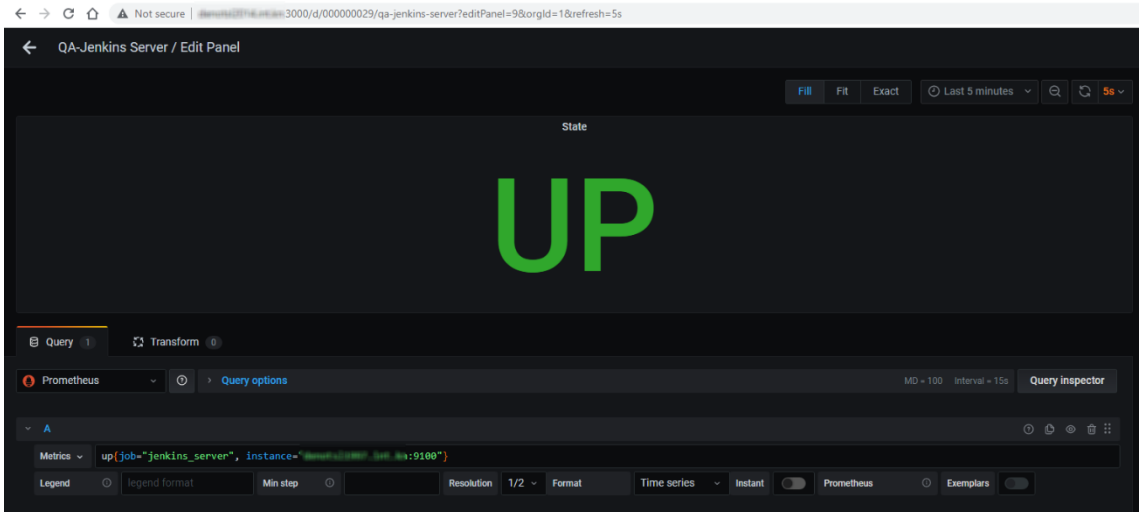
Joonis 27. Operatiivmälu maht Jenkins serveris.

Joonis 28. kuvab graafiku mis näitab kui kaua on Jenkins server töötanud eelmisest taaskäivitamisest.



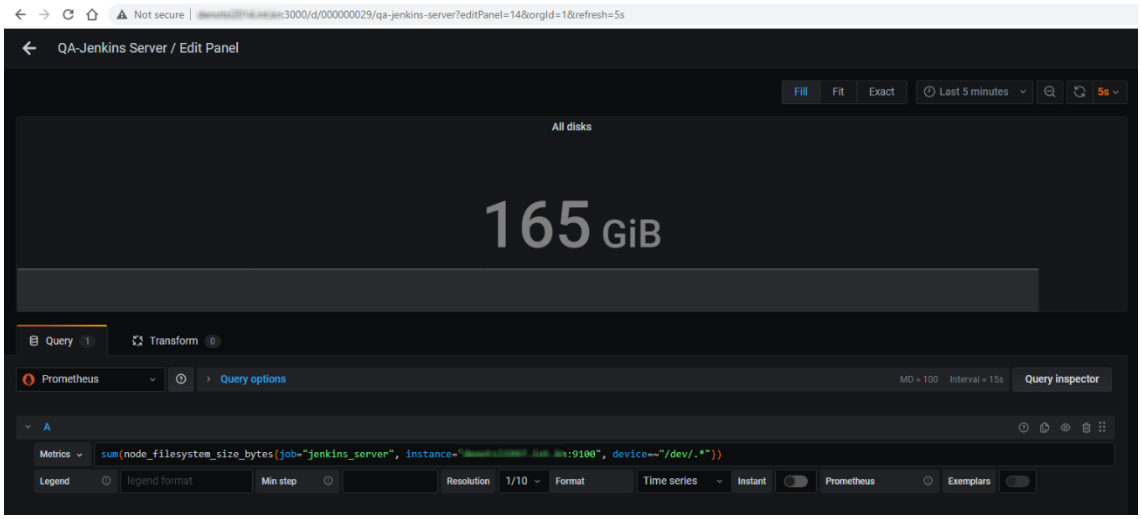
Joonis 28. Jenkins serveri tööaeg.

Joonis 29. kuvab Jenkins serveri staatuse, kui server on maas, siis seiresüsteem näitab et Jenkins server on maas.



Joonis 29. Jenkins serveri staatus.

Joonis 30. kuvab Jenkins ketta mälu suuruse.



Joonis 30. Jenkins serveri kettamälu.

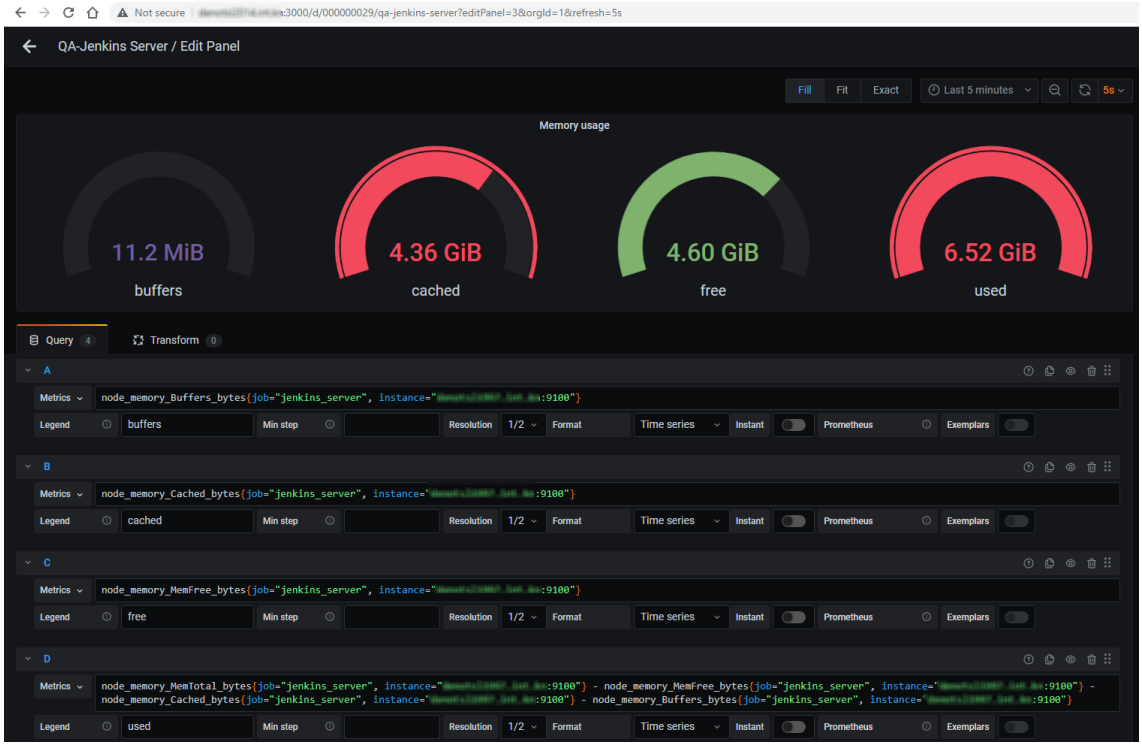
Joonis 31. kuvab Jenkins serveri operatiivmälu kasutuse graafikuna.

Joonis 31. Jenkins serveri operatiivmälu kasutus.

Joonis 32. kuvab Jenkins serveri protsessori kasutust protsentides graafikuna.

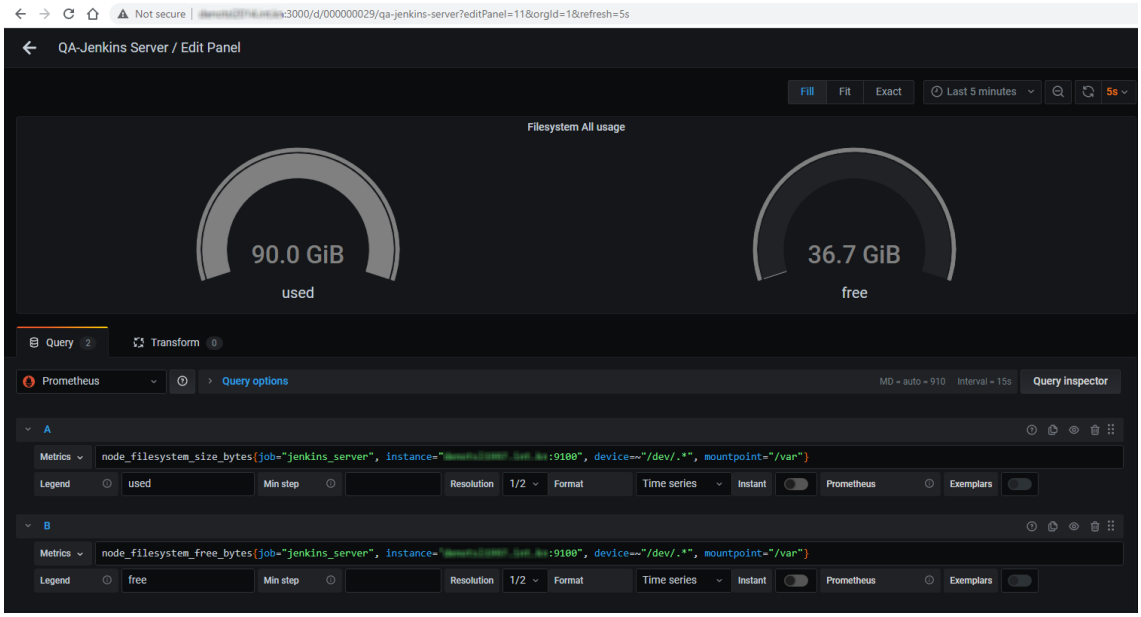
Joonis 32. Jenkins serveri protsessori kasutus.

Joonis 33. kuvab Jenkins serveri operatiivmälu põhjalikumalt mis näitab kui palju on bufferis, vahemälu suurus, kui palju on operatiivmälu vaba ja kui palju on kasutatud.



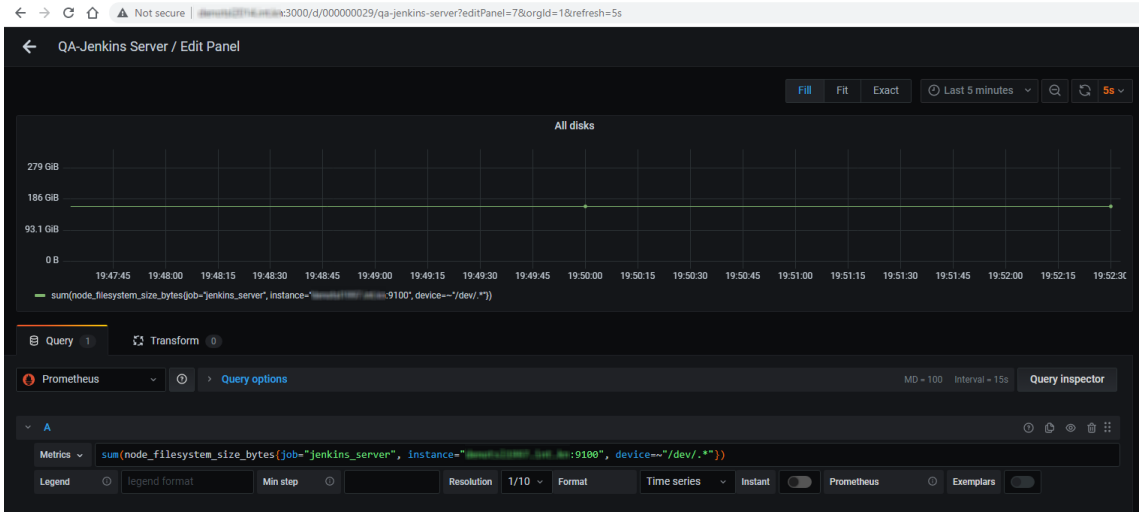
Joonis 33. Jenkins serveri operatiivmälu põhjalikum paneel.

Joonis 34. kuvab failisüsteemi mälu infot, näidates kui palju kettaruumi on kasutatud ja kui palju on vaba.



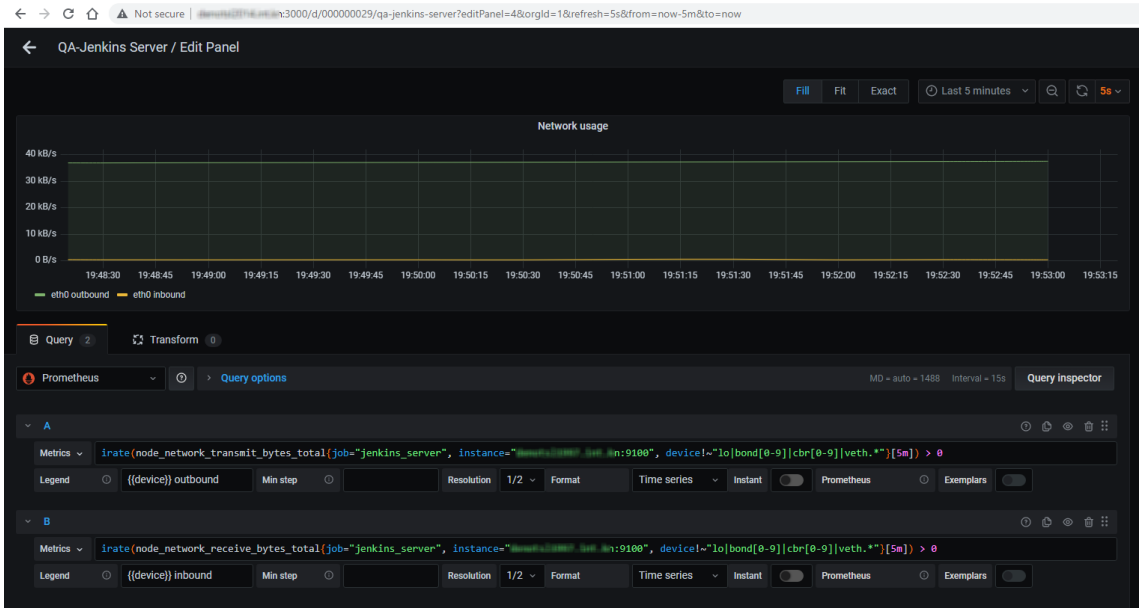
Joonis 34. Ketta ruumi põhjalik vaade.

Joonis 35. kuvab Jenkins serveri kettaruumi kasutuse graafikuna.



Joonis 35. Kõikide kettade ruumi kasutuse graafik.

Joonis 36. kuvab Jenkins serveri võrgu kasutuse, näidates kui suur on sisenev ja väljaminev traafik.



Joonis 36. Jenkins serveri võrgu kasutus.

Joonis 37. kuvab täisvaadet seiresüsteemist kus on näha korraga kõik graafikud.

Joonis 37. Jenkins serveri seiresüsteemi täisvaade.

5.7 Lahenduse analüüs

Lahenduse analüüsi käigus hinnatakse, kuidas loodud seirelahendus vastab seatud eesmärkidele ja kuidas see aitab SwiftLOG projekti sujuva toimimise tagamisel. Selleks vaadeldakse järgmisi punkte:

1. Prometheus ja Grafana integreerimine: Kuidas integreeritud seiresüsteemid töötavad koos ning kuidas nad suudavad pakkuda vajalikku teavet süsteemi oleku kohta.

2. Seiretulemuste analüüs: Vaadeldakse erinevaid meetmeid, mis on kogutud kasutades Prometheus, Grafana, Jenkins'i serveri ja Nexus *repository*. See hõlmab süsteemi jõudlust, ressursikasutust ja erinevate komponentide töökindlust.

3. Teavitussüsteemi töökindlus: Analüüsitakse, kuidas teavitussüsteem suudab tuvastada probleeme ja edastada need DevOps-tiimile e-posti teel, võimaldades neil probleemidele kiiresti reageerida.

4. Lahenduse rakendamine ettevõttes: Lõplik lahendus on juba kasutusel ettevõttes, kus see aitab töötajatel kiiresti tuvastada ja lahendada süsteemiga seotud probleeme. See omakorda parandab süsteemi töökindlust ja tagab projektile SwiftLOG sujuva toimimise.

5. Lahenduse paindlikkus: Uuritakse, kuidas loodud seirelahendus on paindlik ja kohandatav, võimaldades selle kohandamist vastavalt ettevõtte vajadustele ja kasvule.

Pärast tehtud tööd korraldati demo-koosolek, kus autor tutvustas süsteemi toimimist ning selgitas, mida näitab iga seire vaade. Koosolekul osalesid mitmed meeskonnad erinevatest asukohtadest: QA-meeskond Ukrainast, QA-meeskond Tallinnast, DevOps-meeskond Inglismaalt ja DevOps-meeskond Tallinnast. Peale esitlust andsid kõik meeskonnad oma arvamuse seiresüsteemi kohta ja esitasid küsimusi.

Tabel 2. Demo-koosoleku tagasiside ja hinnangud seiresüsteemi kohta

	Demokoosolek	Lisamärkused	Hinnang
QA Ukraina	Andsid positiivset tagasisidet ja hindasid monitooringu süsteemi	Püstitasid küsimusi süsteemi kohta	5/5
QA Tallinn	Avaldasid heameelt monitooringu efektiivsuse üle	Täiendavad ideed süsteemi täiustamiseks	5/5
DevOps Inglismaa	Kiitsid süsteemi ja andsid tunnustust tehtud tööle	Jagasid oma kogemusi sarnaste süsteemidega	5/5
DevOps Tallinn	Hindasid kõrgelt monitooringu süsteemi ja andsid positiivset tagasisidet	Pakkusid välja potentsiaalseid parandusi ja koostöövõimalusi	5/5

Tabel 2. annab ülevaate demo-koosolekul osalenud meeskondadest, nende positiivsest tagasisidest, lisamärkustest ning nende antud hinnangust seiresüsteemile.

Kokkuvõtteks võib öelda, et loodud seirelahendus vastab seatud eesmärkidele ja aitab SwiftLOG projekti sujuva toimimise tagamisel. Lahendus on edukalt integreeritud ettevõtte töövoogudesse, aidates töötajatel kiiresti tuvastada ja lahendada süsteemiga seotud probleeme.

6 Kokkuvõte

Bakalaurusetöö "Seiresüsteemi püstitamine ettevõttele Küehne-Nagel" eesmärgiks oli luua efektiivne ja paindlik jälgimislahendus, mis tagab Küehne-Nagel'i projektile SwiftLOG sujuva toimimise. Töö teoreetiline osa keskendus probleemi püstitamisele, projekti eesmärkide seadmisel ning tingimustele, millele lõpptulemus peab vastama. Samuti tehti ülevaade populaarsetest seiresüsteemidest ning valiti sobivaimad neist.

Praktilises osas kirjeldati samm-sammult lahenduse loomist ja seadistamist, pöörates erilist tähelepanu Prometheus ja Grafana integreerimisele, Jenkins'i serveri ja Nexus *repository* jälgimisele. Lisaks käsitleti Docker'i kasutamist tarkvara paigaldamise ja haldamise hõlbustamiseks.

Lahenduse analüüs näitas, et loodud seirelahendus vastab seatud eesmärkidele ning on edukalt integreeritud ettevõtte töövoogudesse, aidates töötajatel kiiresti tuvastada ja lahendada süsteemiga seotud probleeme.

Kokkuvõttes võib öelda, et töös loodud jälgimislahendus on efektiivne ja paindlik, pakkudes Küehne-Nageli projektile SwiftLOG vajalikku toe ja turvalisust. Lahendus on näidanud end kui olulist vahendit projektide edukal juhtimisel, pakkudes reaalajas teavet süsteemi oleku ja jõudluse kohta.

Kasutatud kirjandus

[1] A. Prakash, „Zabbix: An Introduction To The Server Monitoring Tool,“ GeekyAnts Tech Blog, 11 November 2021. [Võrgumaterjal]. Available: https://techblog.geekyants.com/zabbix-an-introduction-to-the-server-monitoring-tool. [Kasutatud 05 Aprill 2023].

[2] Z. LLC, „Explore Zabbix features,“ Zabbix LLC, 2023. [Võrgumaterjal]. Available: https://www.zabbix.com/features. [Kasutatud 05 Aprill 2023].

[3] A. S. Gillis, „Zabbix,“ TechTarget, Juuni 2018. [Võrgumaterjal]. Available: https://www.techtarget.com/searchitoperations/definition/Zabbix. [Kasutatud 30 Märts 2023].

[4] A. Prakash, „Zabbix: An Introduction To The Server Monitoring Tool,“ GeekyAnts, 11 November 2021. [Võrgumaterjal]. Available: https://techblog.geekyants.com/zabbix-an-introduction-to-the-server-monitoring-tool. [Kasutatud 10 Aprill 2023].

[5] R. KUMAR, „What is Zabbix and How it works? An Overview and Its Use Cases,“ DevOpsSchool, 24 Märts 2022. [Võrgumaterjal]. Available: https://www.devopsschool.com/blog/what-is-zabbix-and-how-it-works-an-overview-and-its-use-cases/. [Kasutatud 10 Aprill 2023].

[6] A. S. Gillis, „Nagios,“ TechTarget, Mai 2018. [Võrgumaterjal]. Available: https://www.techtarget.com/searchitoperations/definition/Nagios. [Kasutatud 2 Aprill 2023].

[7] D. Taylor, „Nagios Tutorial: What is Nagios Tool? Architecture & Installation,“ GURU99, 11 March 2023. [Võrgumaterjal]. Available: https://www.guru99.com/nagios-tutorial.html. [Kasutatud 2 Aprill 2023].

[8] P. Gupta, „Nagios Monitoring Tool,“ EDUCBA, 2023. [Võrgumaterjal]. Available: https://www.educba.com/nagios-monitoring-tool/. [Kasutatud 5 Aprill 2023].

[9] S. Singh, „Prometheus : The Observability Tool,“ talent500, 17 Veebruar 2023. [Võrgumaterjal]. Available: https://talent500.co/blog/prometheus-the-observability-tool/. [Kasutatud 12 Aprill 2023].

[10] C. Hailey, „An Introduction to Prometheus Monitoring (2021),“ Sensu, 1 Juuni 2021. [Võrgumaterjal]. Available: https://sensu.io/blog/introduction-to-prometheus-monitoring. [Kasutatud 12 Aprill 2023].

[11] J. Walker, „How to Set Up a Simple Grafana Cloud Monitoring Dashboard for Your Linux Server,“ 7 Oktoober 2022. [Võrgumaterjal]. Available: https://www.howtogeek.com/devops/how-to-set-up-a-simple-grafana-cloud-monitoring-dashboard-for-your-linux-server/. [Kasutatud 11 Aprill 2023].

[12] S. Team, „Everything You Need to Know about Grafana,“ Skedler, 15 Veebruar 2023. [Võrgumaterjal]. Available: https://www.skedler.com/blog/everything-you-need-to-know-about-grafana/. [Kasutatud 12 Aprill 2023].

[13] D. Olier, „Prometheus monitoring: a new open source generation,“ Pandora FMS, 27 Oktoober 2022. [Võrgumaterjal]. Available: https://pandorafms.com/blog/prometheus-monitoring/. [Kasutatud 7 Aprill 2023].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

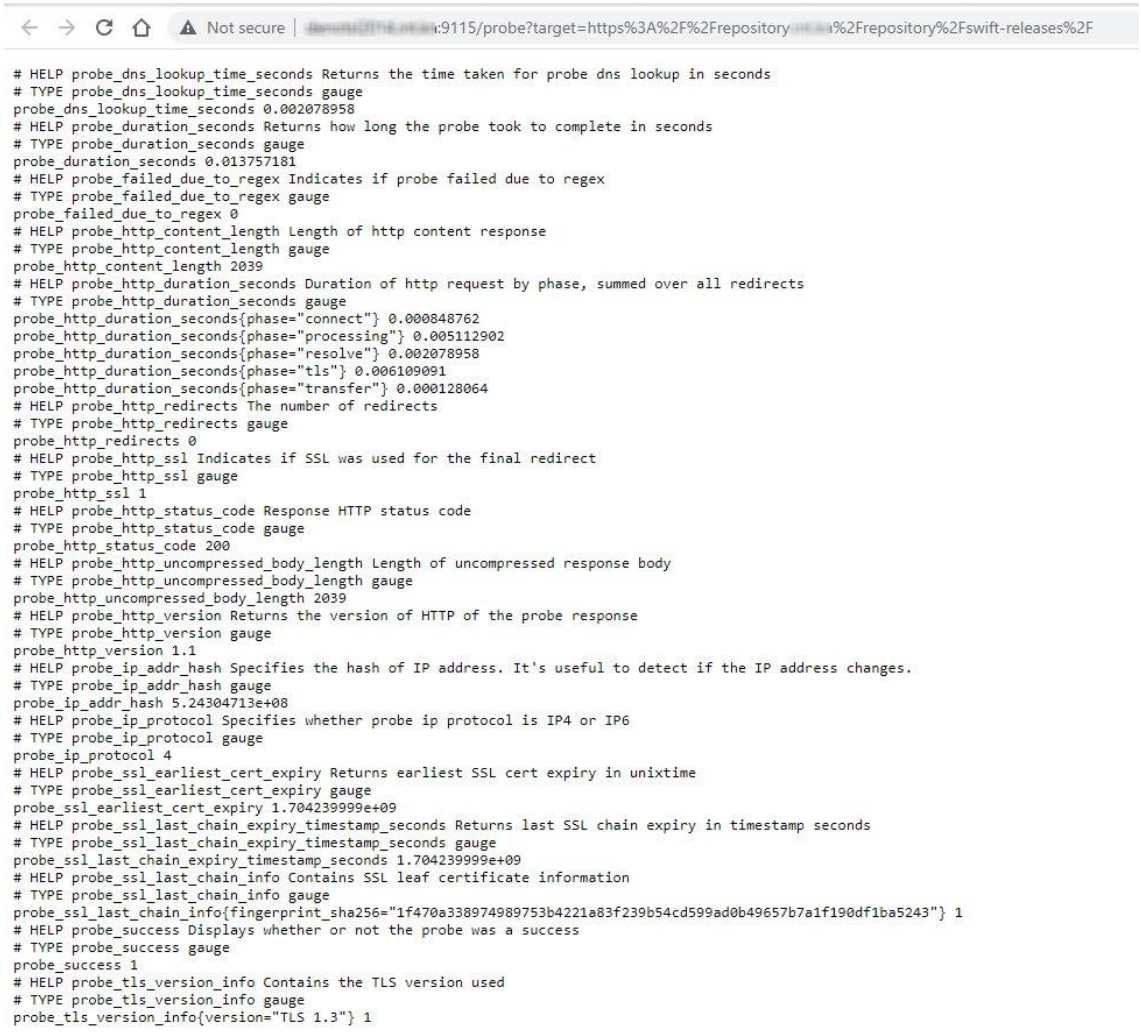
Mina, Alen Siilivask

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Seiresüsteemi püstitamine ettevõttele Küehne-Nagel" , mille juhendaja on Margus Sumla.
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

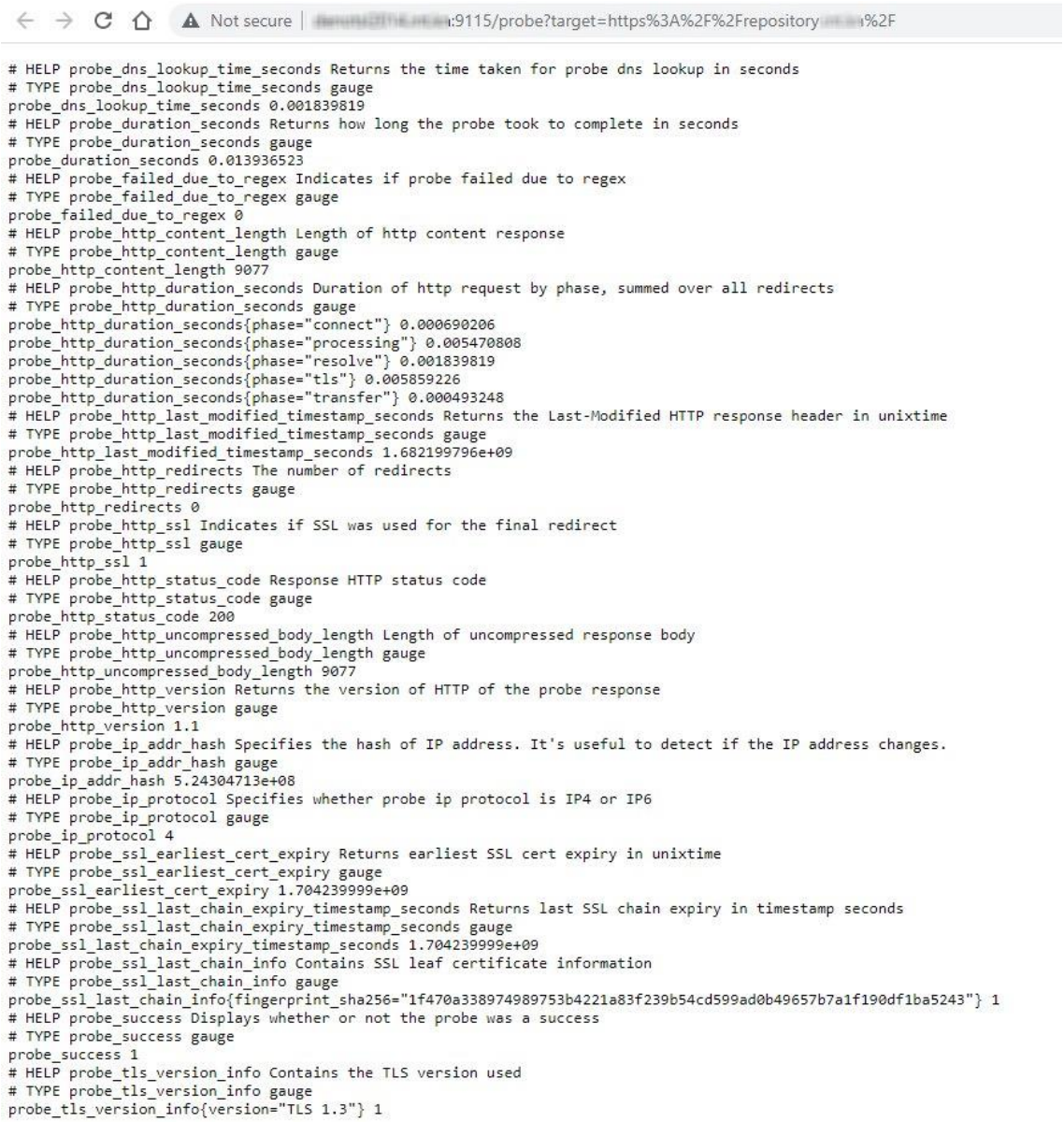
15.05.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

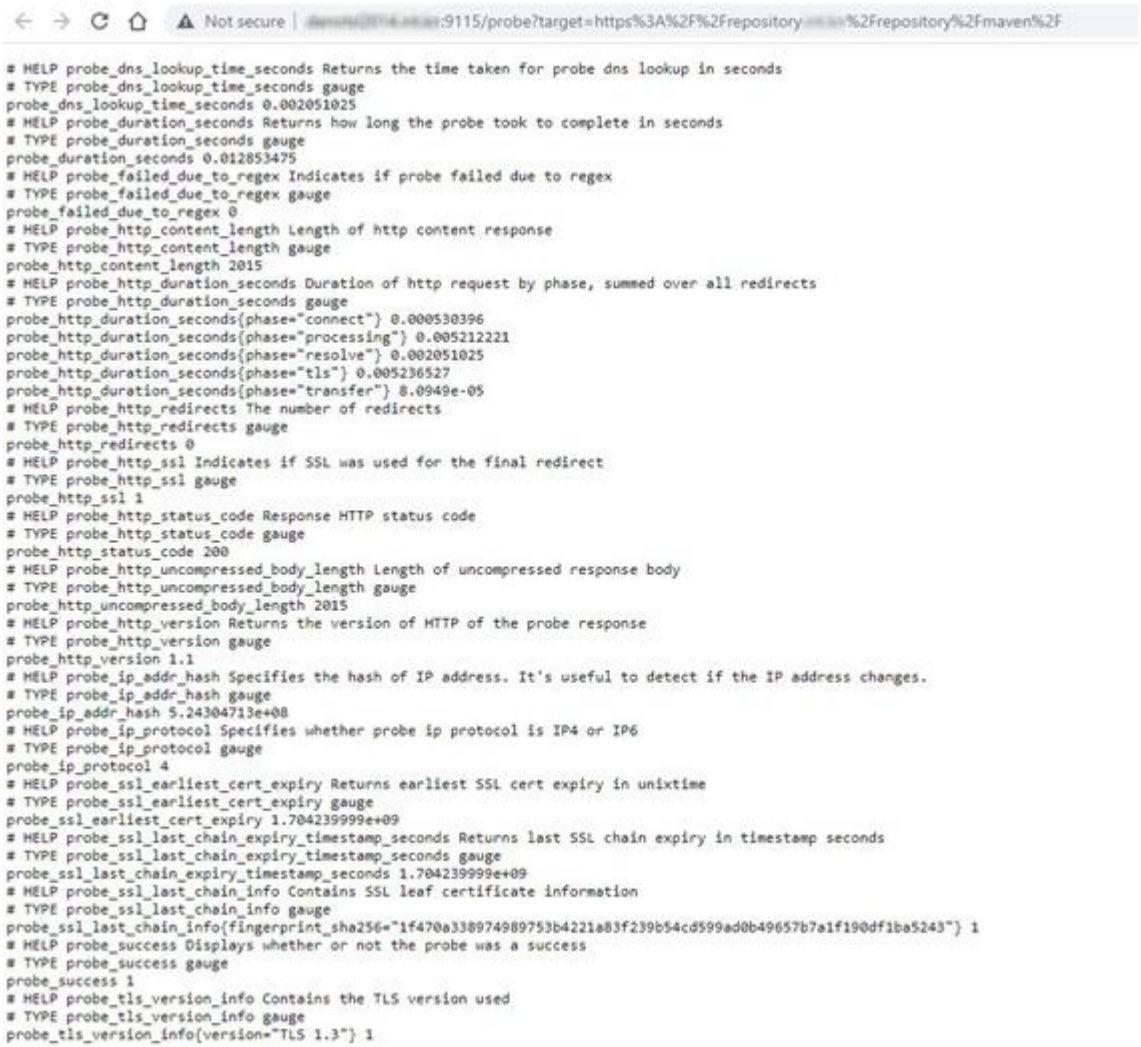
Lisa 2 – Meetmed Nexus Repository "Swift-releases" repositooriumilt



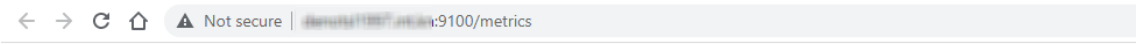
Lisa 3 – Meetmed Nexus Repository oleku kohta



Lisa 4 – Meetmed Nexus Repository "maven" repositooriumilt



Lisa 5 – Jenkins serveri süsteemsed meetmed




```
# HELP go_gc_duration_seconds A summary of the pause duration of garbage collection cycles.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 3.4832e-05
go_gc_duration_seconds{quantile="0.25"} 4.1661e-05
go_gc_duration_seconds{quantile="0.5"} 4.4905e-05
go_gc_duration_seconds{quantile="0.75"} 5.3954e-05
go_gc_duration_seconds{quantile="1"} 0.00236178
go_gc_duration_seconds_sum 103.57170929
go_gc_duration_seconds_count 1.653771e+06
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 8
# HELP go_info Information about the Go environment.
# TYPE go_info gauge
go_info{version="go1.17.3"} 1
# HELP go_memstats_alloc_bytes Number of bytes allocated and still in use.
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 2.443344e+06
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated, even if freed.
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 3.182495416632e+12
# HELP go_memstats_buck_hash_sys_bytes Number of bytes used by the profiling bucket hash table.
# TYPE go_memstats_buck_hash_sys_bytes gauge
go_memstats_buck_hash_sys_bytes 2.041871e+06
# HELP go_memstats_frees_total Total number of frees.
# TYPE go_memstats_frees_total counter
go_memstats_frees_total 4.1796626128e+10
# HELP go_memstats_gc_cpu_fraction The fraction of this program's available CPU time used by the GC since the program started.
# TYPE go_memstats_gc_cpu_fraction gauge
go_memstats_gc_cpu_fraction 0.00010826589348108923
# HELP go_memstats_gc_sys_bytes Number of bytes used for garbage collection system metadata.
# TYPE go_memstats_gc_sys_bytes gauge
go_memstats_gc_sys_bytes 5.466296e+06
# HELP go_memstats_heap_alloc_bytes Number of heap bytes allocated and still in use.
# TYPE go_memstats_heap_alloc_bytes gauge
go_memstats_heap_alloc_bytes 2.443344e+06
# HELP go_memstats_heap_idle_bytes Number of heap bytes waiting to be used.
# TYPE go_memstats_heap_idle_bytes gauge
go_memstats_heap_idle_bytes 3.588096e+06
# HELP go_memstats_heap_inuse_bytes Number of heap bytes that are in use.
# TYPE go_memstats_heap_inuse_bytes gauge
go_memstats_heap_inuse_bytes 4.308992e+06
# HELP go_memstats_heap_objects Number of allocated objects.
# TYPE go_memstats_heap_objects gauge
go_memstats_heap_objects 16333
# HELP go_memstats_heap_released_bytes Number of heap bytes released to OS.
# TYPE go_memstats_heap_released_bytes gauge
go_memstats_heap_released_bytes 2.875392e+06
# HELP go_memstats_heap_sys_bytes Number of heap bytes obtained from system.
# TYPE go_memstats_heap_sys_bytes gauge
go_memstats_heap_sys_bytes 7.897088e+06
# HELP go_memstats_last_gc_time_seconds Number of seconds since 1970 of last garbage collection.
# TYPE go_memstats_last_gc_time_seconds gauge
go_memstats_last_gc_time_seconds 1.6823379519271426e+09
# HELP go_memstats_lookups_total Total number of pointer lookups.
# TYPE go_memstats_lookups_total counter
go_memstats_lookups_total 0
# HELP go_memstats_mallocs_total Total number of mallocs.
# TYPE go_memstats_mallocs_total counter
go_memstats_mallocs_total 4.1796642461e+10
# HELP go_memstats_mcache_inuse_bytes Number of bytes in use by mcache structures.
# TYPE go_memstats_mcache_inuse_bytes gauge
go_memstats_mcache_inuse_bytes 1200
```