TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Margus Laanem 194111IAAB

Keskse seireteenuse loomine rahvusvahelises tarkvaraarenduse ettevõttes

Bakalaureusetöö

Juhendaja: Siim Vene
Magister
Kaasjuhendaja: Rein Remmel
Rakenduskõrgharidus

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: Margus Laanem

15.05.2023

Annotatsioon

Antud lõputöös käsitletakse ühe rahvusvahelise tarkvaraarenduse ettevõtte seirepinu parendamist. Fookuses on aja jooksul tekkinud probleemid või puudused. Olemasoleva lahenduse killustatus on muutnud kasutajate kogemuse seire teostamisel ebamugavaks. Lisaks ei võimalda tänane lahendus säilitada seire andmeid piisavalt kaua.

Antud lõputöö eesmärgiks on analüüsida ja juurutada antud ettevõttes süsteemiseire komponendid mille tulemusena tekiks üks keskne seirepinu mis võimaldab teostada seiret keskkondade üleselt ning vastaks kõigile ettevõtte poolt seatud nõuetele.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 34 leheküljel, 26 peatükki, 12 joonist, 3 tabelit.

Abstract

Central Observability Stack for a Global Software Development Company

This thesis is focusing on improving systems monitoring for a global software development company. Several deficits or requirements have emerged over time that need addressing. Current solution is fragmented and is unconvenient to use. Furthermore the solution is not suitable for long term storing of monitoring data.

The goal of this thesis is to analyse and implement different monitoring components so that a Central observability stack is formed. The new solution should provide central observability over all environments and must be compliant to all the requirements set by the company.

The thesis is in Estonian and contains 34 pages of text, 26 chapters, 12 figures, 3 tables.

Lühendite ja mõistete sõnastik

AWS	Amazon Web Services
EKS	Elastic Kubernetes Service
S3	Simple Storage Service
SaaS	Software as a Service, tarkvara teenusena
exporter	Tarkvara mis kogub mõõdustikku ja väljastab kogutud andmed Prometheusele kogumiseks
CNCF	Cloud Native Computing Foundation
API	Application Programmable Interface, rakendusliides
UDP	User Datagram Protocol, transpordiprotokoll TCP/IP-protokollistikus
GitOPS	Raamistik mis võimaldab hallata rakenduste ja infrastruktuuri elutsüklit kasutades tarkvaraarenduse tööriistasid.
git	Versioonihaldus süsteem
YAML	Inimloetav andmete jadastamise keel
JSON	Lihtne andmevahetusvorming, mis põhineb JavaScripti alamhulgal, on hõlbus inimlugemiseks ja -kirjutuseks
sidecar	Eriotstarbeline konteiner mis laiendab kauna funktsionaalsust

Sisukord

1 Sissejuhatus 10
1.1 Probleemid 10
1.2 Eesmärk 12
2 Metoodika 13
3 Nõuded 14
4 Mõõdustiku talletamise tehnoloogiate analüüs 16
4.1 Cortex 16
4.2 Mimir 18
4.3 Thanos 19
4.4 Analüüsi kokkuvõte 20
5 Logide talletamise tehnoloogiate analüüs 23
5.1 Elasticsearch 23
5.2 Loki 25
5.3 Analüüsi kokkuvõte 28
6 Hajutatud jälgitavuse tehnoloogiate analüüs 30
6.1 Jaeger 31
6.2 Tempo 32
6.3 Zipkin 33
6.4 Analüüsi kokkuvõte 35
7 Valitud tehnoloogiate juurutamine ja valideerimine 37
7.1 Keskse Grafana ja Mimiri paigaldus 38
7.2 Loki paigaldus 40
7.3 Tempo paigaldus 41
7.4 Valideerimise kokkuvõte 41
8 Finantsanalüüs 43
9 Kokkuvõte 44
Kasutatud kirjandus 45
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 48
Lisa 2 – Terraformi koodinäited 49

Lisa 3 – Helm väärtusfailide näidised 51
Lisa 4 – Logstashi seadistuse koodinäidis 54

Jooniste loetelu

Joonis 1. Cortexi arhitektuurijoonis. 16
Joonis 2. Mimiri joonis. 18
Joonis 3. Mimiri arhitektuuri joonis. 19
Joonis 4. Thanose arhitektuuri joonis. 20
Joonis 5. Elasticsearchi arhitektuuri joonis 24
Joonis 6. Loki arhitektuuri joonis. 27
Joonis 7. Hajutatud jälgitavuse joonis. 30
Joonis 8. Jaegeri arhitektuuri joonis 32
Joonis 9. Tempo arhitektuuri joonis. 33
Joonis 10. Zipkini arhitektuuri joonis. 34
Joonis 11. Terraformi joonis. 37
Joonis 12. ArgoCD joonis. 38

Tabelite loetelu

Tabel 1. Mõõdustiku salvestamistehnoloogiate võrdlustabel. 21
Tabel 2. Logimislahenduste võrdlustabel. 28
Tabel 3. Hajutatud jälgitavuse lahenduste hindamistabel. 35

1 Sissejuhatus

Entigo OÜ kliendiks on tarkvaraarendusega tegelev ettevõte kes pakub oma arendatavat teenust klientidele üle maailma. Selleks, et pakutav teenus oleks kvaliteetne ja kättesaadav kõikjal on antud ettevõte valinud oma tarkvara alusplatvormiks Amazon Web Services (edaspidi AWS) pilveteenuse. AWS-ist kasutatakse Kubernetese kobarate käitamiseks AWS Elastic Kubernetes service (edaspidi EKS) hallatud teenust. Ettevõttel on kasutusel kümme EKS klastrit erinevates AWS kontodes ja regioonides. Pooled nendest on arenduseks ja teine pool toodanguks. Antud lõputöö skoobiks on nende keskkondade süsteemi seire. Seire hulka kuuluvad infrastruktuuri ning rakenduste mõõdustik, logid ja hajutatud jälgitavuse(inglise keeles distributed tracing) andmed.

Aastate jooksul on ettevõttes välja arendatud töötav süsteemiseire lahendus kasutades erinevaid vabavaralisi kui ka AWS hallatud teenuseid. Meetrikat kogutakse kasutades erinevaid exportereid mille väljastatud mõõdustikku kogutakse kokku ja talletatakse Prometheuse andmebaasi. Mõõdustiku visualiseerimiseks kasutatakse Grafanat.

Logide kogumiseks kasutatakse Filebeati mis saadab logid edasi Logstashi. Logstashis pannakse logidele külge sildid ja talletatakse logid AWS Opensearch teenusesse. Logide visualiseerimiseks on igasse keskkonda paigaldatud Kibana.

Eelnevalt kirjeldatud komplekt seirevahendeid on täitnud oma eesmärki üsna hästi, küll aga on aja jooksul ka tekkinud mõningad puudused või nõuded mis vajavad lahendamist, et tagada seire jätkusuutlikus kas siis vastavalt tehnoloogia arengule või tekkinud ärilistele nõuetele.

1.1 Probleemid

Antud lõputöös keskendutakse järgnevate probleemide lahenduste leidmisele:

1. Seirepinu kasutajateks on arendajad ja insenerid kes tegelevad igapäevaselt infrastruktuuri ja rakenduste seire teostamisega. Seda siis kas parenduste tegemiseks või probleemide tuvastamiseks. Hetkel kasutusel oleva lahenduse

puhul on kasutajate peamiseks probleemiks seirepinu killustatus. Igal keskkonnal on oma komplekt seirevahendeid. Tihti tekib vajadus võrrelda meetrikat või logisid erinevate keskkondade vahel ja see on üsna keeruline arvestades, et graafikuid ei ole võimalik kõrvutada, sest igal keskkonnal on oma Grafana isend. Ettevõtte soov on konsolideerida kogu mõõdustiku ja logide visualiseerimine ühte kesksesse keskkonda selliselt, et oleks võimalik luua vaateid ja koondpaneele keskkondade üleselt.

2. Mõõdustiku kogumiseks ja talletamiseks kasutatakse täna Prometheust. Paraku Prometheus ei ole mõeldud meetrika talletamiseks pikema perioodi vältel. Tootja enda väitel on ta sobilik kuni paari kuu andmete hoiustamiseks. Selleks, et talletada mõõdustikku pikemalt on Prometheuse arendajad lisanud toe saata mõõdustikku edasi rakendustesse mis on võimelised talletama andmeid palju pikema aja vältel. Ettevõtte soov oleks talletada mõõdustikku 13 kuud [1].

3. Logide puhul on probleemiks AWS Opensearch teenus. Ettevõtte hinnangul ei ole antud teenuse kulud mõistlikud ning logimise kulude proportsioon kogukuludest on liiga kõrge [2]. Hetkel hoitakse Opensearchis logisid 30 päeva. Ettevõtte soov on talletada logisid aga 13 kuud ning seda kuluefektiivsemalt.

   Nagu mõõdustiku puhul on ka logide puhul probleemiks killustatus kuna igal keskkonnal on kasutusel oma Opensearch ja Kibana. Sarnaselt mõõdustikule on ettevõtte soov koguda logisid keskselt ühte kohta kokku ja visualiseerida neid kasutades Grafanat.

4. Viimaseks vaadeldavaks probleemiks on soov hakata koguma ka päringute jälgesid. Kuna ettevõte arendab oma rakendusi kasutades mikroteenuseid siis on päringute jälgede ja nendega seotud mõõdustiku kogumine äärmiselt oluline, et proaktiivselt tuvastada võimalikke probleeme [3]. Hajutatud jälgitavuse puhul pannakse igale päringule külge ID ja selle abil saab hõlpsasti tuvastada kui kaua kulus ühel või teisel teenusel päringu teenindamisele aega terve läbitud teekonna vältel.

1.2 Eesmärk

Lõputöö eesmärgiks on täiendada olemasolevat seirepinu komponentidega mis võimaldab täita mõõdustiku, logide ja hajutatud jälgitavuse andmete keskse kogumise, pikaaegse säilitamise ja kasutusmugavuse eesmärgi. Töö tulemusena peaks kliendile tekkima keskne seirepinu mis suudab koguda ja visualiseerida mõõdustikku, logisid ja jälgesid kõikidest keskkondadest. Kõik komponendid peavad moodustama ühtse kokku töötava terviku. Lahendus peab suutma säilitada mõõdustikku, logisid ja hajutatud jälgitavuse andmeid 13 kuud.

2 Metoodika

Sissejuhatuses kirjeldatud probleemide ja eesmärkide täitmiseks teostatakse kolm eraldi analüüsi.

- Võrreldakse tooteid mis võimaldavad keskselt vastu võtta ja talletada Prometheuse kogutud mõõdustikku.
- Võrreldakse tooteid millega asendada Opensearch logide talletamiseks.
- Võrreldakse tooteid millega koguda ja talletada hajutatud jälgitavuse andmeid keskselt.

Iga analüüs lähtub seatud eesmärkidest ja nõuetest. Analüüsi käigus valitakse taustauuringu käigus sobivaimad tooted ja võrreldakse neid omavahel. Taustauuringut teostatakse otsingumootoreid kasutades ja otsingu tulemusel leitud materjalide hulgas enim kajastust leidnud tooted lähevad omavahel võrdlemisse. Valimist jäävad välja tarkvara teenusena(edaspidi SaaS) teenused, sest antud ettevõttel on varasemast kogemus New Relicu nimelise SaaS tootega. Olgugi, et tegu on väga hea teenusega ei ole nende hinnangul see kulu õigustatud arvestades, et saada on alternatiivseid lahendusi. Sobivuse hindamiseks koostatakse tabel kus hinnatakse iga toodet vastavalt esitatud nõuetele. Valituks saab toode mis kogub analüüsi käigus enim punkte.

Analüüside käigus välja valitud tooted paigaldatakse seejärel kliendi keskkonda ja valideeritakse nende sobivus koostöös arendajate ja teiste inseneridega.

3 Nõuded

Kesksele seirepinule on seatud järgnevad nõuded:

- Kogu visualiseerimine peab toimuma ühest kohast kasutades Grafanat. Nagu sissejuhatuses manitud siis on ettevõte juba varasemalt investeerinud kõvasti aega oma seire ülesse ehitamisele Grafana baasil. Nii infrastruktuurile kui ka rakendustele on loodud oma vaated ning koondpaneelid. Lisaks on arendajad harjunud kirjutama päringuid PromQL keeles, et luua endale sobivaid vaateid ja koondpaneel.

- Mõõdustik, logid ja jäljed peavad olema salvestatud kesksesse asukohta. See võimaldab keskkondade üleselt teha päringuid ja vähendab killustatust seirepinus.

- Mõõdustik, logid ja jäljed peavad olema kättesaadavad tagasiulatuvalt kuni 13 kuu ulatuses. Tegemist on ärilise nõudega, et tagada piisavalt tagasiulatuvalt seire andmed kui kliendid peaksid seda nõudma.

- Kõik paigaldatavad komponendid peavad olema võimelised töötama Kubernetese platvormil. Ettevõte on ehitanud oma infrastruktuuri ülesse selliselt, et kõik mitte hallatud teenused töötaksid Kubernetesel baseeruval platvormil. See tagab ühtse halduse kasutades juba olemasolevaid haldustööriistu.

- Eelistatud on lahendused mis ei vaja täiendavate tarkvara litsentside soetamist. Ettevõtte plaan selleks aastaks on pigem kulude optimeerimine ja ei ole ette nähtud uute tarkvarade litsentside hankimist.

- Mõõdustiku kogumise vahendid, milleks on erinevad exporterid ja Prometheus peavad jääma samaks, sest nendele tööriistadele tuginedes on seatud ülesse juba vaated ja koondpaneelid ning seadistatud tõrgetest teavitamine kasutades Alertmanageri ja Pagerduty-t. Ettevõtte jaoks on ka kasulikum kui arendajad saavad keskenduda tootearendusele ja kasutada edasi juba omale tuttavaid töövahendeid mitte tegeleda uute tööriistade kasutusele võtust tingitud suuremahuliste koolitamistega.

- Logide kogumise ja edastamise vahendid peavad jääma samaks, sest suur hulk rakendusi kasutavad Filebeati sidecarina kauna küljes, et lugeda logisid failidest. Filebeat saadab logid edasi Logstashi ja sealt on võimalik saata logid omakorda mitmesse kohta paralleelselt mis aitab ühelt süsteemilt ülemineku kasutajatele teha sujuvamaks.

- Eelistatud on lahendused mis võimaldavad andmeid salvestada AWS S3 objekt andmesalvestus pinnale. AWS S3 on oma loomult juba kõrgkäideldav ja skaleeruv ning seda 3-4 korda odavama hinnaga kui sama mahu juures blokk andmesalvetus pind. S3 Standard gigabaidi hind on 0,023€ versus gp3 gigabaidi hind mis on 0.088€. Hinnad on võetud AWS hinnakirjast 1.04.2023 seisuga [4].

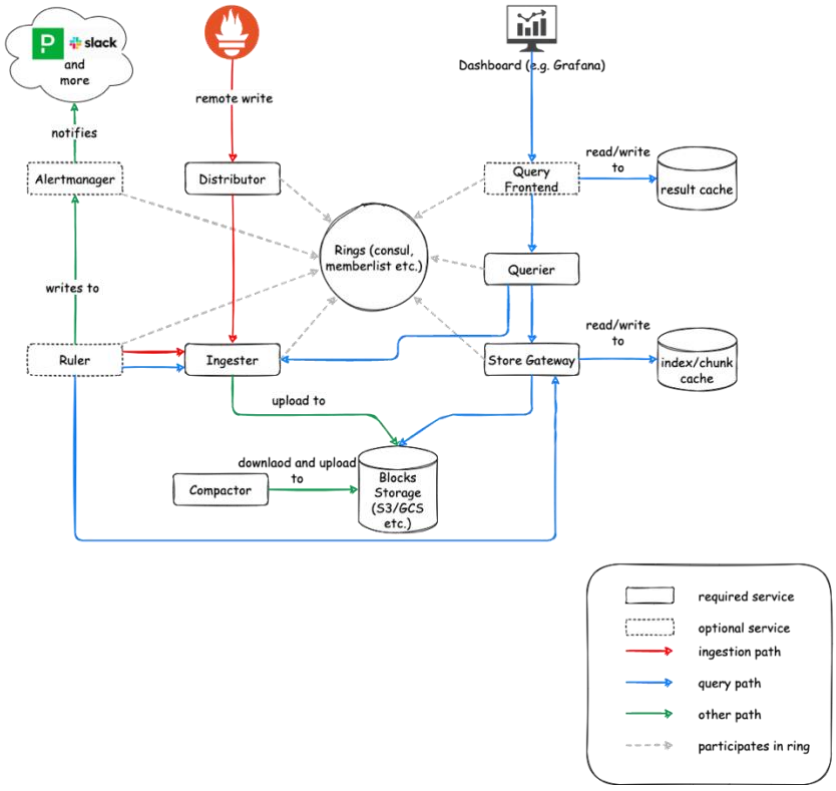
- Tooted peavad olema kõrgkäideldavad, et tagada seire toimivus kas hooldustegevuste käigus või mõne Kubernetese sõlme rikke korral.

4 Mõõdustiku talletamise tehnoloogiate analüüs

Selles peatükis analüüsitakse ja võrreldakse taustauuringu käigus leitud võimalikke lahendusi mõõdustiku keskseks talletamiseks. Taustauuringust välja tulnud toodetest valiti analüüsimiseks Cortex, Mimir ja Thanos. Järgnevalt kirjeldatakse nende kolme lahenduse omadusi ja arhitektuuri.

4.1 Cortex

Cortex on CNCF inkubatsiooni projekt mis lubab Prometheusele horisontaalselt skaleeruvat, kõrgkäideldavat ja mitmik kliendi toega pikaajalist salvestustehnoloogiat [5]. Cortexi üks suurimaid arendusse panustajaid aastatel 2018-2022 oli Grafana Labs, kes kasutas Cortexit oma SaaS teenuse Grafana Cloud tagateenusena mõõdustiku salvestamiseks. 2022 aasta märtsis aga otsustas Grafana Labs hakata edasi arendama oma toodet ja lahkus panustajate hulgast [6]. Githubi andmetel on pärast Grafana Labsi lahkumist arendustegevus oluliselt langenud [7]. Cortexi arhitektuur on illustreeritud all oleval joonisel:



Joonis 1. Cortexi arhitektuurijoonis [8].

Cortex tööpõhimõte seisneb selles, et Prometheus kogub mõõdustiku kokku ja kasutades remote_write API-t edastatakse mõõdustik Cortexisse. Cortex salvestab vastuvõetud andmed maha objekt andmesalvestus pinnale. Remote_write API võimaldab lisada igale päringule kaasa HTTP päise kus on defineeritud kliendi ID mille alusel saab isoleerida erinevate klientide andmed. Cortex koosneb minimaalselt viiest komponendist mis kõik on kõrgkäideldavas režiimis eraldiseisvalt skaleeritavad vastavalt vajadusele. Kohustuslikud komponendid paigalduseks on [8]:

- Distributor – Esimene peatuspunkt vastuvõetud andmetele. See komponent teostab vastuvõetud andmete valideerimise. Valideeritud andmed saadetakse seejärel edasi Ingesterile

- Ingester – Vastutab andmete kirjutamise eest lõplikku andmesalvestuspinnale. Tegelikult ei kirjutata andmeid kohe maha lõpp asukohta vaid säilitatakse esialgu Ingesteri mälus. See tagab kiirema päringu värskematele andmetele. Vaikimisi kirjutatakse mälust kettale maha andmed iga kahe tunni tagant. Andmesalvestuseks on toetatud AWS, Azure ja Google Cloud objekt andmesalvestus pinnad. Mitte kõrgkäideldavas režiimis on toetatud ka kohalikule kettale salvestamine.

- Querier – Komponent mis kasutab PromQL päringukeelt andmete pärimiseks kas siis lõpp asukohast või Ingesterist vastavalt päringus esitatud perioodile.

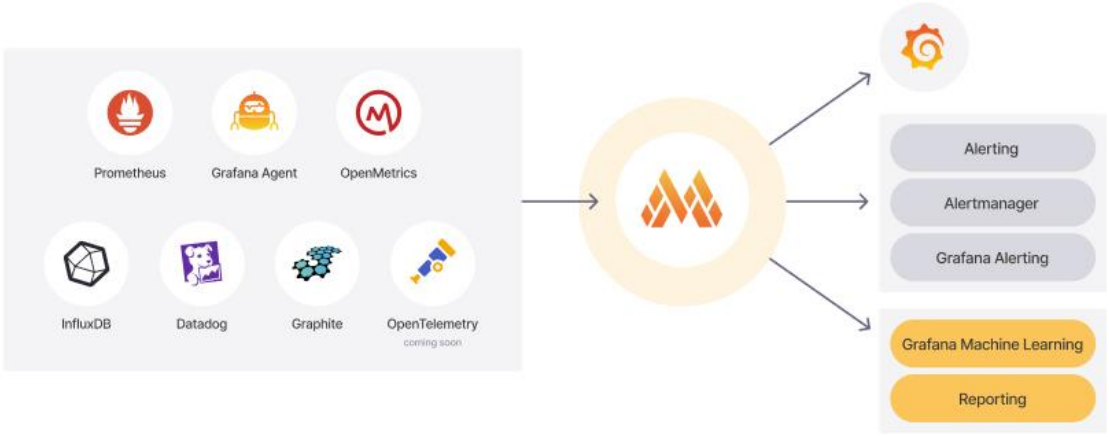
- Compactor – Teenus mis optimeerib andmeplokkide kasutust, pakib väiksemad plokid ühte suuremasse blokki kokku mis aitab kaasa hiljem päringute kiirusele ning võtab vähem mahtu kettal.

- Store gateway – Teenus mis vastutab päringute kättesaadavuse eest blokkidest. Store gateway laeb perioodiliselt alla talletatud andmete indeksit et omada ülevaadet millises blokis mis andmed asuvad.

Cortexi saab lisada Grafanasse andmeallikana ning teha päringuid kasutades PromQL päringukeelt samamoodi nagu Prometheuse puhul. Seega kasutajate jaoks ei tähenda see lisakoormust uue päringukeele õppimise näol.

4.2 Mimir

Grafana Labs tuli 2022 aasta märtsis välja oma lahendusega pärast Cortexi arendamise loobumisest [6]. Toote nimeks sai Mimir ja seda kutsuti kui Grafana mõõdustiku kogumise tagateenuseks. Mimirisse lisati lisaks Prometheusele veel mitme toote tugi mõõdustiku vastuvõtmiseks.

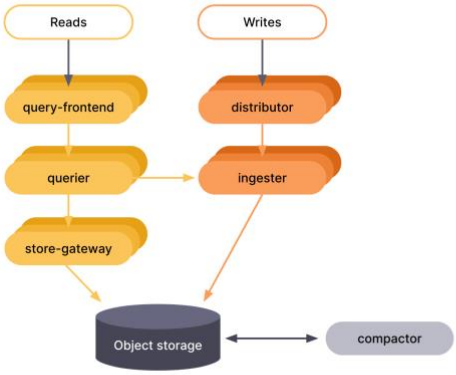


Joonis 2. Mimiri joonis.

Lisaks Prometheusele on toetatud Grafana Agent, Opentelemetry collector, Datadog, Graphite ja InfluxDB [9].

Oma arhitektuurilt ja töö põhimõttelt on Mimir väga sarnane Cortexile [10]. Seda seetõttu, et aluseks võeti just nimelt Cortexis tehtud töö. Tehti palju enda jaoks sobivaid muudatusi ning lisati varasemalt kommerts lahendustes kasutatud omadused mis lubavad paremat andmete pakkimist ja kiiremat otsingut. Githubi andmetel arendatakse Mimirit väga aktiivselt ja iga kahe kuni kolme nädala järel tuleb välja uus reliis mis lisab uut funktsionaalsust ja parandab vigu [6].

Mimiri kohustuslikud komponendid on samad mis Cortexi puhul ja isegi dokumentatsioon on suures osas sõna sõnalt sama mis Cortexil [11]. Mimiri arhitektuur on illustreeritud all oleval skeemil:



Joonis 3. Mimiri arhitektuuri joonis [12].

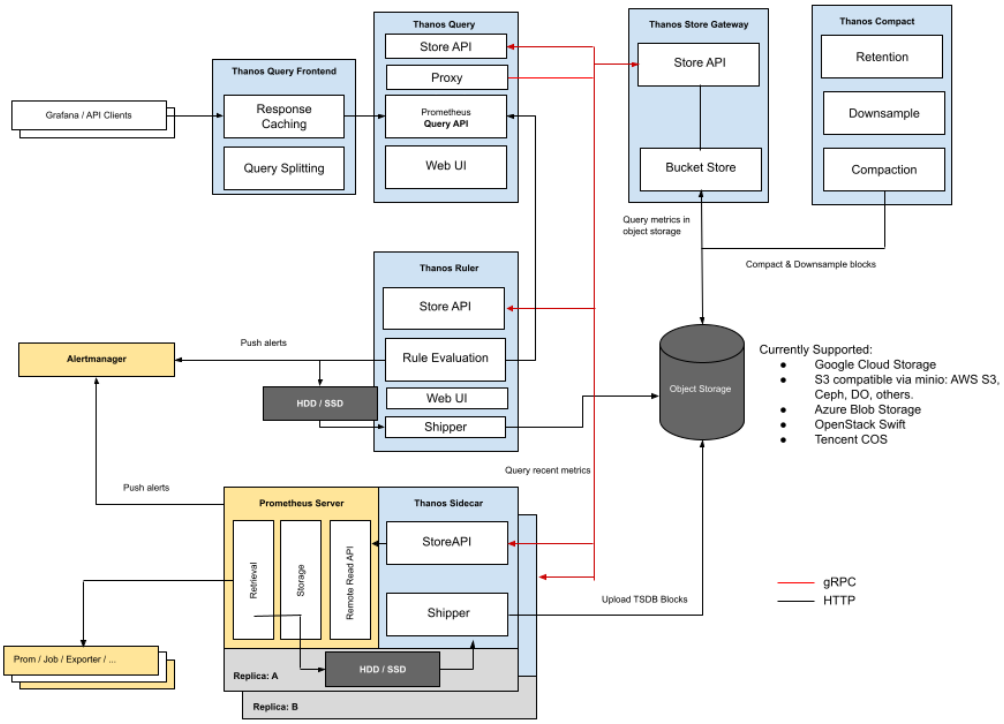
4.3 Thanos

Thanost hakati arendama umbes samal ajal kui Cortexit aga nende lähenemine sellele kuidas andmeid Prometheusest saadakse erineb märkimisväärselt. Kui Cortex ja Mimir kasutavad Prometheuse sisse ehitatud remote_write API-t siis Thanos liidab Prometheusele enda sidecari ja kogub andmeid otse Prometheuse aeg-seeria andmebaasist ning edastab siis kogutud andmed sobivale objekt andmesalvestus pinnale [13].

Kui andmete kogumine välja arvata siis ei erine ka siin Thanos arhitektuuriliselt suures osas kuidagi eelnevalt vaadeldud toodetest. Kasutusel on sama funktsionaalsusega komponendid mille nimetus on veidi erinev. Kõik komponendid on kõrgkäideldavad ja eraldiseisvalt skaleeritavad. Thanos toetab andmete hoiustamiseks järgnevaid tooteid:

- Google Cloud Storage
- AWS S3 ja kõik teised S3 ühilduvad andmesalvestus pinnad
- Azure Storage Account
- Openstack Swift
- Lokaalne failisüsteem
- Oracle Cloud Infrastructure Object Storage

Thanose arhitektuur on illustreeritud alloleval joonisel.



Joonis 4. Thanose arhitektuuri joonis [13].

Vaatamata sellele, et Thanost on arendatud peaaegu sama kaua kui Cortexit ja oluliselt kauem kui Mimirit ei ole siiani jõutud oma arenduses versioon 1.0-ni. Antud lõputöö kirjutamise ajal on värskeim versioon 0.31.0 [14].

4.4 Analüüsi kokkuvõte

Võrreldes eelnevalt tutvustatud kolme toodet on selgelt näha, et tegu on väga sarnaste toodetega nii funktsionaalsuselt kui ka arhitektuurilt. Esmaspilgul sobivad kõik kolm toodet lahendamaks Prometheuse andmesalvestuse probleeme ning neid saab kasutada keskse andmeallikana Grafanas andmete pärimiseks. Kõik tooted on oma arhitektuurilt skaleeruvad, kõrgkäideldavad, toetavad mitmikkliendi puhul eraldatust ning toetavad andmete salvestamist väga pikaks ajaks.

Allolevas tabelis on tooted kõrvutatud nõuetega, et neid hinnata.

Tabel 1. Mõõdustiku salvestamistehnoloogiate võrdlustabel.

Nõuded	**Cortex**	**Mimir**	**Thanos**
Kõrgkäideldav	1	1	1
Saab lisada Grafanasse andmeallikaks	1	1	1
Võimaldab andmete kogumist kesksesse asukohta	1	1	1
Toetab andmete säilitamist vähemalt 13 kuud	1	1	1
Toetatud on paigaldus Kubernetesesse	1	1	1
Litsentside hind	1	1	1
Toetab AWS S3 andmete talletamiseks	1	1	1
Arendus	0	1	0
Punkte kokku	7	8	7

Antud tabelist on näha, et kõik valikus olnud tooted vastasid enamustele esitatud nõuetele. Valiku tegemisel sai määravaks siiski toote arendus. Nagu analüüsi käigus selgus siis Cortexi arendus on märkimisväärselt langenud peale Grafana Labsi lahkumist ja Thanos pole veel jõudnud oma arendusega versioon 1.0-ni. Seega antud juhul osutus valituks Mimir ning sellega jätkatakse valideerimist.

5 Logide talletamise tehnoloogiate analüüs

Antud peatükis analüüsitakse erinevaid võimalusi logide talletamiseks ja visualiseerimiseks kasutades Grafanat. Taustauuringut tehes saigi aluseks võetud esialgu Grafana dokumentatsioonist võetud nimekiri andmeallikatest mis toetavad logide salvestamist [15]. Antud nimekirjast jäi sõelale kolm kandidaati kuhu on võimalik saata logisid Logstashi vahendusel. Elasticsearch, Loki ja InfluxDB. Alles jäänud toodetest selgus lähemal uurimisel, et InfluxDB avatud lähtekoodiga versioon ei toeta töötamist kõrgkäideldaval rešiimil [16]. Selleks on vajalik soetada enterprise versioon. Seega on taustauuringu käigus alles jäänud ainult Elasticsearch ja Loki.

5.1 Elasticsearch

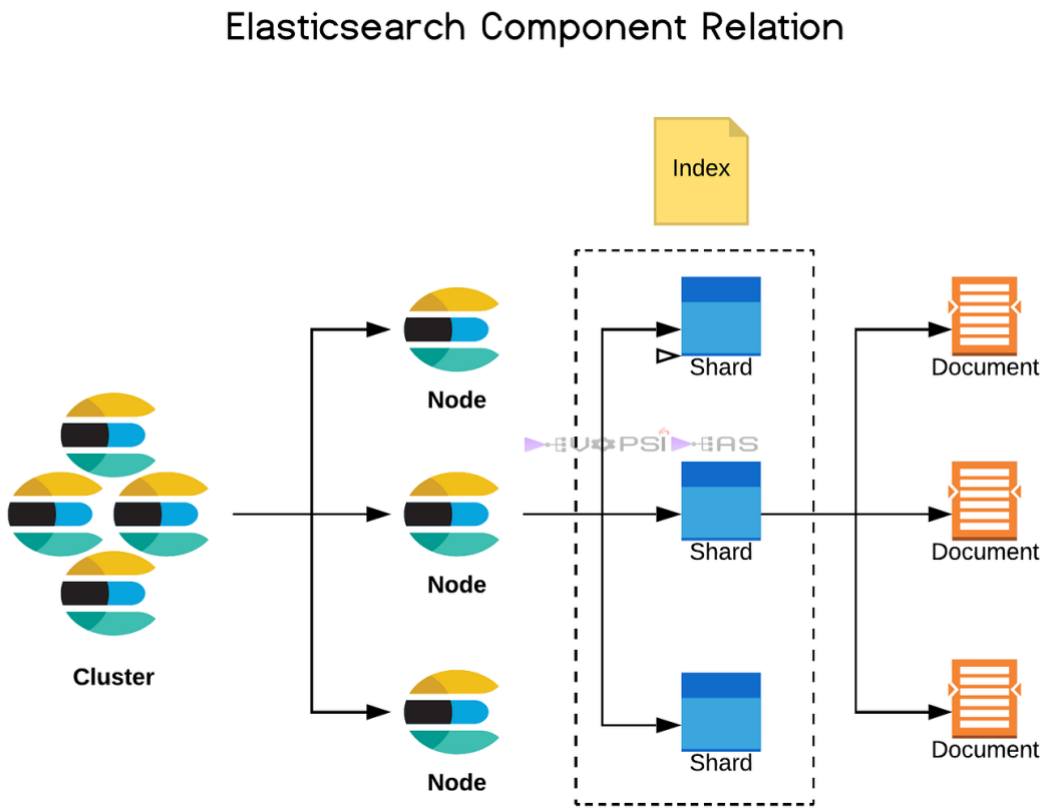
Elasticsearch on kõrgkäideldav erinevate andmete ja suurte andmemahtude salvestamiseks ja analüüsimiseks ehitatud otsingumootor [17]. Elasticsearch on ehitatud Apache Lucene peale. Elasticsearchi hakati arendama 2010 aastal sama nimelise firma poolt. Tegu on avatud lähtekoodiga tarkvaraga.

Elasticsearch on Elasticu poolt arendatava seirepinu üks komponentidest koos Kibana, Beatsi ja Logstashiga. Hetkel kasutuses olev AWS Opensearch lahendus baseerub samal Elasticsearch tehnoloogial [18].

Elasticsearchi klaster koosneb ühest või mitmest sõlmest. Sõlmed hoiustavad andmeid ja osalevad andmete indekseerimises ning päringutele vastamises. Elastisearchi sõlmedel on kolm erinevat rolli [19]:

- Pea sõlm- Kontrollib klastri tööd ja on vastutav indeksite loomise ning kustutamise eest.
- Andmete sõlm – Hoiustab andmeid ja käitab otsinguid ning andmete agregeerimist.
- Klient sõlm – edastab klastri päringud pea sõlmele ja andmetega seotud päringud andmete sõlmedele.

Logide kogumise puhul saadab klient Elasticsearchi andmed JSON formaadis. Elasticsearch indekseerib kogu logi sisu ja kirjutab kettale maha. Kogu logi sisu indekseerimisega on võimalik kiirelt väga suuri ja keerulisi päringuid väga kiirelt töödelda. Alloleval joonisel on illustreeritud Elasticsearchi arhitektuur ja omavahelised seosed.



Joonis 5. Elasticsearchi arhitektuuri joonis [19].

Andmete salvestamiseks kasutab Elasticsearch iga andme sõlme lokaalset kettapinda. Kõrgkäideldavuse tagamiseks kopeeritakse andmeid vastavalt seadistusele mitme lõime peale. Kuigi tegu on väga kiire ja laialt kasutuses oleva logide agregeerimise süsteemiga on tema kasutamine suurte andmemahtude korral väga kulukas. Peamiseks kuluallikaks on arvutusressurss. Nimelt selleks, et tagada kiire otsimine laeb Elasticsearch indeksid muutmällu. Kui muutmälust indekseid ei leita siis hakatakse neid alles kettalt otsima ja sellisel juhul on päring aeglane. Seega mida rohkem andmeid Elasticsearchi salvestatakse seda rohkem tarbib ta muutmälu indeksite hoidmiseks. Isegi kui andmeid ei pärita on tarvis süsteemi kiiruse tagamiseks neid ikkagi muutmälus hoida.

5.2 Loki

Loki on kõrgkäideldav, horisontaalselt skaleeruv ja mitmik kliendi toega logide agregeerimise süsteem [20]. Loki arendus algas 2018 aastal Grafana Labsi poolt. Toote arenduses on Grafana Labsi väitel võetud aluseks, et toode oleks kuluefektiivne ja lihtne kasutada. Kuigi Lokit on võimalik kasutada monoliitse rakendusena testimiseks mis sisaldab kõiki komponente on ta disainitud töötama mikroteenustena ja iga komponent on eraldi skaleeritav vastavalt vajadusele. Loki on võimeline talletama andmeid väga pika aja jooksul. Loki koosneb järgnevatest komponentidest [21]:

- Distributor – komponent mis tegeleb sisse tulevate andmevoogudega. Iga andmevoog valideeritakse. Distributor tükeldab seejärel andmevoo tükkideks ja saadab tükid laiali erinavetele Ingesteritele paralleelseks töötlemiseks.

- Ingester – Ingester vastutab logide maha kirjutamise eest püsivale objekt andmesalvestus pinnale. Andmete maha kirjutamine ei toimu koheselt vaid mingi kindla perioodi vältel, mis on administraatori poolt seadistatav. Mälus olevad andmed on kopeeritud vaikimisi kolme Ingesteri vahel, et vähendada andmekadu juhul kui mõne Ingesteriga peaks tekkima probleem.

- Query-frontend – teenus mis haldab logide lugemisel päringute järjekorda.

- Querier – teenus mis küsib Query-frontend teenuselt järjekorrast päringuid, teostab päringu vastu andmeid mis asuvad maha kirjutatatult või veel Ingesteri mälus ja tagastab tulemuse Query-frontendile. Querier kasutab päringuteks LogQL päringukeelt.

- Compactor – teenus mis taustal tegeleb väiksemate andmeblokkide pakkimiseks suurematesse blokkidesse, et säästa andmesalvestusruumi.

Lisaks eelmainitud kohustuslikele komponentidele on võimalik lisada ka puhverserver, mis aitab puhverdada päringuid mis korduval pärimisel säästab päringule kuluvat aega ja võrguliiklust.

Loki toetab andmete salvestamiseks järgnevaid lahendusi:

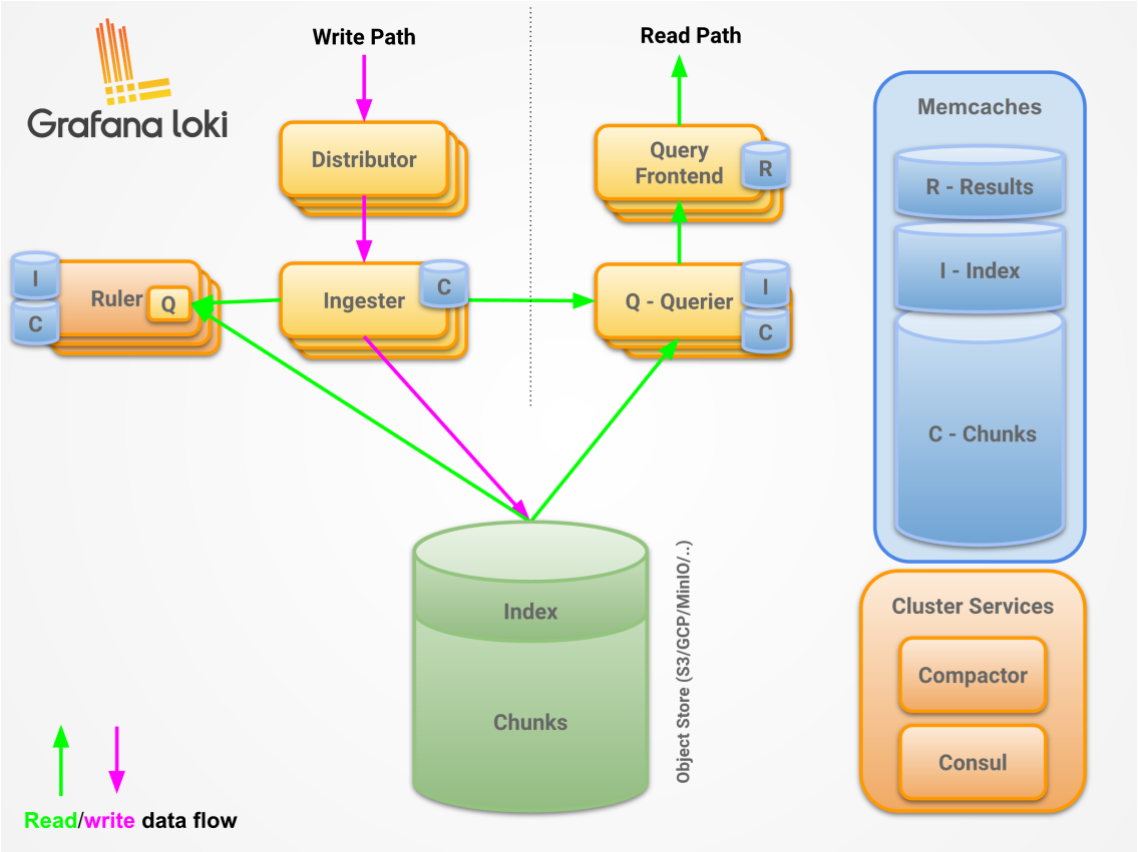
- Amazon DynamoDB

- Google Bigtable
- Apache Cassandra
- Amazon S3
- Google Cloud Storage
- Failisüsteem - toetatud ainult monoliitrešiimis
- Baidu Object Storage

Lisaks logidele hoiab Loki eraldi indexit kus on kirjas logil küljes olnud sildid ja viide millises blokis ta asub. Erinevalt Elasticsearchist mis indekseerib kõik andmeväljad on Loki eeliseks madalam kulu indeksitele, samas ei võimalda see nii head teksti põhist otsingut [22]. Indeksi hoidmiseks on toetatud järgnevad lahendused:

- Single Store (boltdb-shipper) – võimaldab Indeksit hoida objekt andmesalvestus pindadel.
- Amazon DynamoDB
- Google Bigtable
- Apache Cassandra
- BoltDB – toetatud ainult monoliitrešiimis

Loki arhitektuur on illustreeritud järgneval joonisel:



Joonis 6. Loki arhitektuuri joonis [21].

Lokisse on võimalik logisid saata järgmiste klientide poolt [23]:

- Promtail
- Docker Driver
- Fluentd
- Fluent Bit
- Logstash
- Lambda Promtail

5.3 Analüüsi kokkuvõte

Elasticsearch on sisuliselt sama lahendus mis täna kasutusel olev AWS Opensearch. Oma arhitektuurilt on tegemist väga võimsa ja kiire otsingumootoriga. Samas on sellel ka omad miinused. Kuna Elasticsearch laeb indeksid muutmällu, on tarvis väga suure muutmälu mahuga sõlmesid, et tagada võimalikult kiired otsingutulemused. Lisaks kasutatakse andmete salvestamiseks blokk andmesalvetus pinda.

Loki samas läheneb logide talletamisele teistmoodi. Rõhku on pandud pigem kulude kokkuhoiule, samas pole loobutud kõrgest käideldavusest ja skaleeritavusest. Madalamad nõuded arvutusressursile ja andmete hoiustamine AWS S3-s võimaldab märkimisväärset kulude kokkuhoidu võrreldes Elasticsearchiga.

Allolevas tabelis on tooted kõrvutatud nõuetega, et neid hinnata.

Tabel 2. Logimislahenduste võrdlustabel.

Nõuded	**Elasticsearch**	**Loki**
Kõrgkäideldav	1	1
Saab lisada Grafanasse andmeallikaks	1	1
Võimaldab andmete kogumist kesksesse asukohta	1	1
Toetab andmete säilitamist vähemalt 13 kuud	1	1
Toetab logide kogumist kasutades Logstashi	1	1

Toetatud on paigaldus Kubernetesesse	1	1
Litsentside hind	1	1
Toetab AWS S3 andmete talletamiseks	0	1
Punkte kokku	7	8

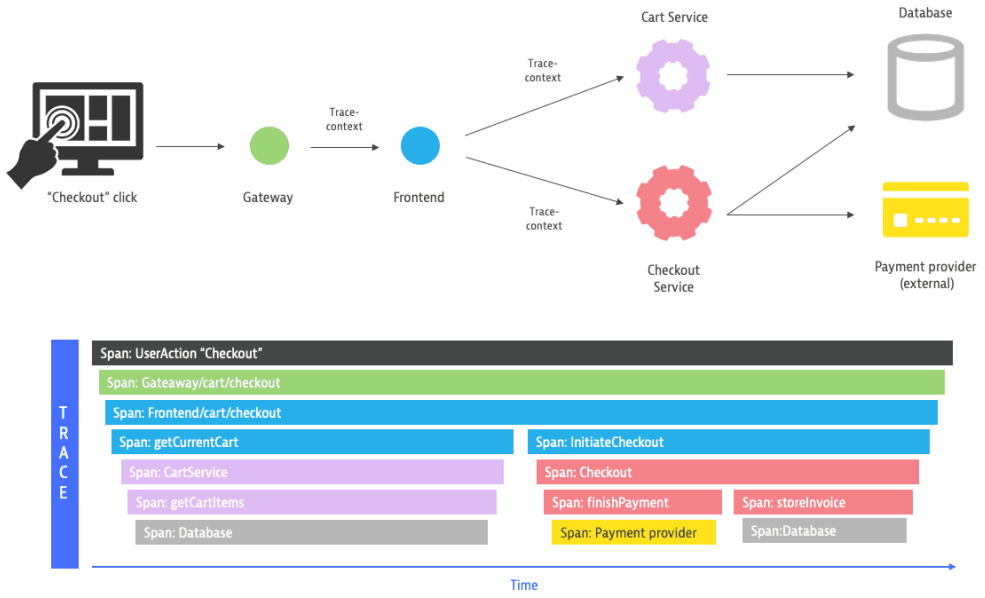
Tulenevalt sellest kuidas andmeid hoiustatakse, mis on selle hoiustamise kulu ja arvestades fakti, et Elasticsearch on juba tegelikult kasutuses oleva AWS Opensearch teenuse ise hallatav versioon, jätkatakse valideerimise faasis Lokiga.

Kuna Filebeat võimaldab andmeid saata nii Opensearchi kui Lokisse üheaegselt siis on võimalik valideerimisel võrrelda mõlema lahenduse jõudlust ja kogu lahenduse kulu ning seeläbi teha lõplik otsus kas jätkata olemasoleva lahendusega või asendada see Lokiga.

6 Hajutatud jälgitavuse tehnoloogiate analüüs

Antud peatükis analüüsitakse hajutatud jälgitavuse mõõdustiku ja jälgede talletamise tehnoloogiaid. Kui mõõdustik ja logid on seire puhul üsna iseenesest mõistetavad siis hajutatud jälgitavus võib olla paljude jaoks uus termin. Siinkohal oleks ilmselt mõistlik lahti seletada mida hajutatud jälgitavus endast kujutab.

Hajutatud jälgitavus võimaldab jälgida päringute elutsüklit mis läbivad erinevaid mikroteenuseid, rakendusi või andmebaase [3] [24]. Päringud tekitavad igal sammul jälgesid mis sisaldavad olulist infot antud ajahetkel selle kohta kuidas päringut serveeritakse ja kas esineb mingeid anomaaliaid. Igale päringule lisatakse unikaalne ID mis päringu edenedes seotakse tekkinud jälgedega. Ühe päringu jäljed moodustavad kokku span-i. Antud jälgesid analüüsides koos mõõdustiku ja logidega on võimalik tuvastada või ennetada erinevaid võimalikke tõrkeid funktsiooni täpsusega. See kui täpselt on võimalik tuvastada sõltub paljuski sellest kui täpselt on arendajad oma rakendust instrueerinud välja andma mõõdustikku. Allolev pilt illustreerib kuidas hajutatud jälgitavus töötab:



Joonis 7. Hajutatud jälgitavuse joonis [24].

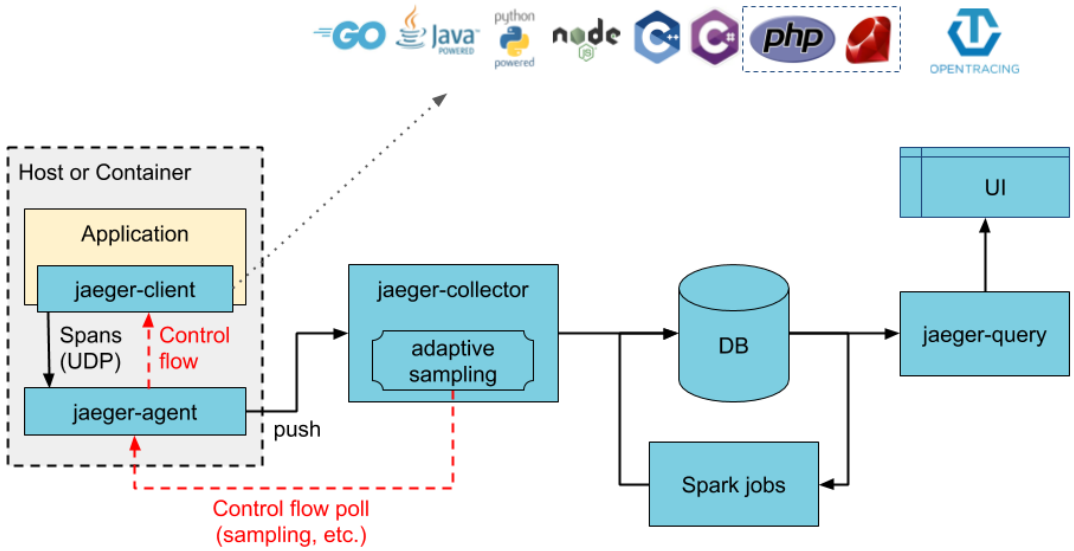
Nagu selgub esitatud nõuetest siis valitud tehnoloogia peab võimaldama vastu võtma mõõdustikku ja jälgi mis on kogutud kasutades Opentelemetry tööriistasid. Taustauuringu põhjal leiti kolm võimalikku sobivat tarkvara. Nendeks on Jaeger, Tempo ja Zipkin.

6.1 Jaeger

Jaeger on spetsiaalselt mikroteenuste jälgede kogumiseks ja visualiseerimiseks loodud tarkvara. Algselt loodud Uberis, hiljem anti ta üle CNCF-ile inkubatsiooniprojektiks. Jaeger on skaleeruv ja kiirusele orienteeritud [25]. Jaegeril on analüüsimiseks oma veebiliides aga Jaegerit on võimalik liidestada ka Grafanaga, et teostada päringuid ja analüüsi. Jaeger koosneb järgmisest komponentidest [26]:

- Jaeger-client – Liidestatud otse rakenduse sisse. Kogub rakendusest jälgesid ja saadab need üle UDP protokolli edasi jaeger-agentile.
- Jaeger-agent – Rakendus mis on lisatud rakenduse konteinerile külgkorvina. Võtab jaeger-clientilt jäljed vastu ja saadab edasi jaeger-collectorile.
- Jaeger-collector – võtab jäljed vastu ja saadab edasi kas Kafka sõnumivahetusteenusesse või kirjutab andmed otse andmebaasi.
- Jaeger-query – Teostab UI kaudu tehtud päringuid andmebaasist.

Jaegeri arhitektuur on illustreeritud alloleval skeemil.



Joonis 8. Jaegeri arhitektuuri joonis [26].

Jaeger toetab andmete hoiustamiseks järgnevaid andmebaase:

- Cassandra
- Elasticsearch

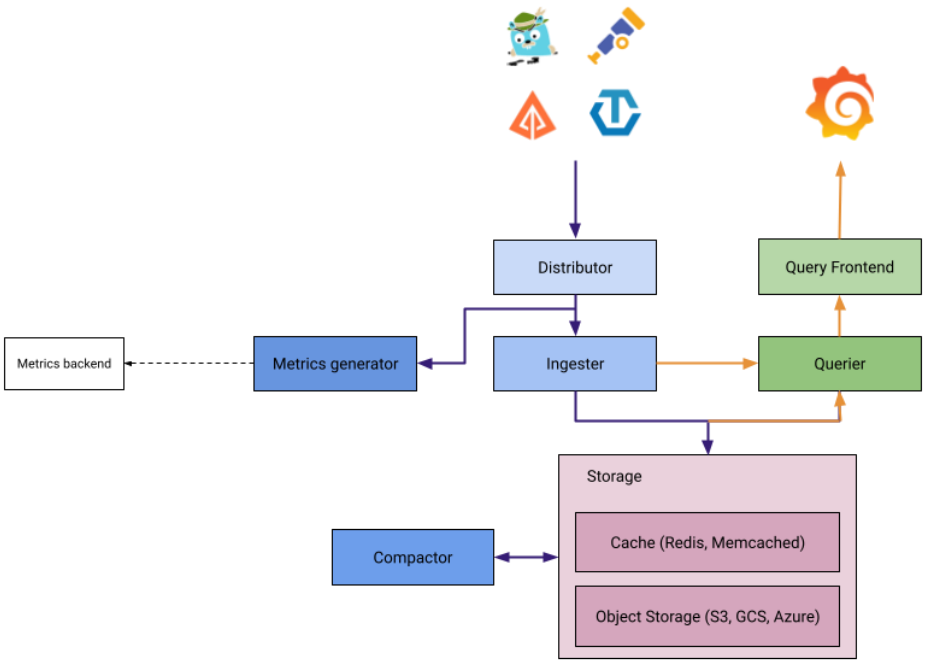
6.2 Tempo

Tempo on Grafana Labsi poolt välja antud toode mis võimaldab koguda jälgi kasutades erinevaid agente. Grafana sõnutsi on tegemist kuluefektiivse ja skaleeruva alternatiiviga Jaegerile ja Zipkinile. Erinevalt Jaegerist ja Zipkinist kasutab Tempo andmete talletamiseks objekt andmesalvestustehnoloogiaid. See võimaldab koguda ja salvestada väga suure hulga andmeid ilma neid eelnevalt töötlemata. Tempo integreerub Grafana, Mimiri ja Loki-ga ning võimaldab analüüsida jälgi koos logide ja muu mõõdustikuga mis annab väga hea vaadeldavuse [27]. Tempo koosneb järgnevates komponentidest [28]:

- Distributor – võtab vastu talle saadetud span-e. Hashib traceID ja saadab edasi Ingesterile.
- Ingester – pakib vastuvõetud andmed blokkideks, indekseerib need ning kirjutab maha andmesalvestus pinnale.

- Query frontend – vastutab päringute järjekorra eest.
- Querier – Võtab Query frontendi järjekorrast päringu ja otsib päringule vastuse kas Ingesterist või objekt andmesalvestus pinnalt.
- Compactor – Pakib perioodiliselt andmeid et blokkide arv ei kasvaks üleliia suureks.

Allolev skeem illustreerib Tempo arhitektuuri:



Joonis 9. Tempo arhitektuuri joonis [27].

Tempo on võimeline andmeid vastu võtma kasutades erinevaid agente. Eelistatud on Grafana agent aga toetatud on ka Jaeger, Zipkin ja Opentelemetry agent.

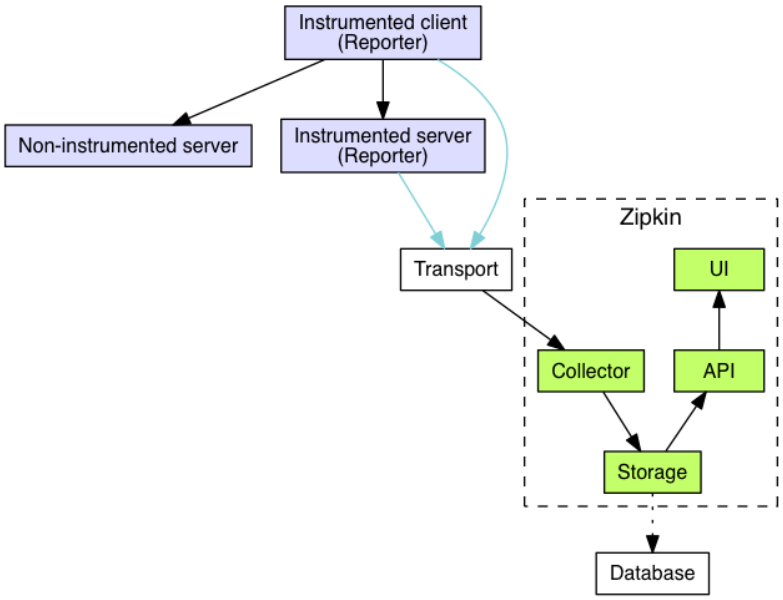
6.3 Zipkin

Zipkini puhul instrumenteeritakse andmete saatmine rakendusest kas üle http protokolli või kasutades mõnda sõnumivahetuse lahendust. Nendeks võivad olla Kafka, ActiveMQ või RabbitMQ. Zipkin ise koosneb järgnevatest komponentides [29]:

- Collector – võtab saadetud jäljed vastu ja kirjutab maha andmebaasi.

- API – teostab andmebaasist päringuid vastavalt sellele mida UI kaudu on päritud.
- UI – kasutajaliides mis võimaldab andmeid visualiseerida ja analüüsida. Lisaks on võimalik näha antud kasutajaliidese kaudu erinevate mikroteenuste omavahelisi seoseid.

Zipkini arhitektuur on illustreeritud alloleval skeemil:



Joonis 10. Zipkini arhitektuuri joonis [29].

Zipkin toetab andmete talletamiseks järgnevaid andmebaase:

- Cassandra
- Elasticsearch
- MySQL

Lisaks on Zipkiniga võimalik edastada kogutud andmeid mõnda sõnumivahetuse tarkvarasse nagu Kafka, RabbitMQ jne. ning siis sealt juba omale sobivasse kohta andmed maha kirjutada.

6.4 Analüüsi kokkuvõte

Allolevas tabelis on tooted kõrvutatud nõuetega, et neid hinnata.

Tabel 3. Hajutatud jälgitavuse lahenduste hindamistabel.

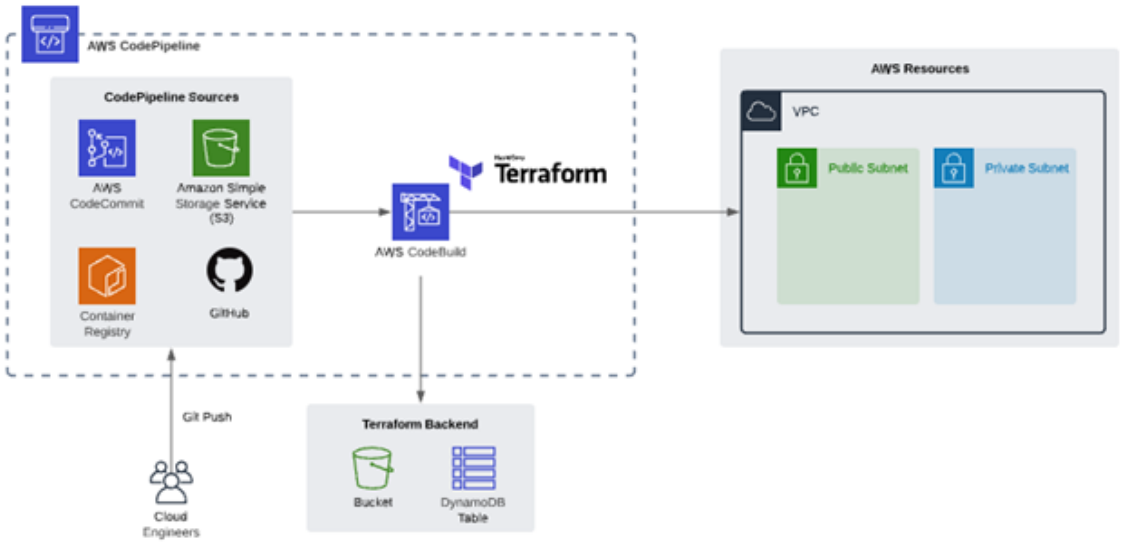
Nõuded	**Jaeger**	**Tempo**	**Zipkin**
Kõrgkäideldav	1	1	1
Saab lisada Grafanasse andmeallikaks	1	1	1
Võimaldab andmete kogumist kesksesse asukohta	1	1	1
Toetab andmete säilitamist vähemalt 13 kuud	1	1	1
Toetatud on paigaldus Kubernetesesse	1	1	1
Litsentside hind	1	1	1
Toetab AWS S3 andmete talletamiseks	0	1	0
Toetab andmete kogumist kasutades Opentelemetry tööriistasid	1	1	1
Punkte kokku	7	8	7

Antud analüüsist järeldub, et kõikidele seatud tingimustele vastab ainult Tempo. Kuigi Jaeger ja Zipkin on olnud turul kõige kauem vajavad nad andmete talletamiseks mõnda välist andmebaasi. Kuna Cassandra või Elasticsearchi klastri haldamine ei ole kliendi jaoks hetkel eelistatud ja pigem on soov S3 toetavaid lahendusi kasutada siis jätkatakse antud lõputöö raames Tempo paigaldamise ja valideerimisega.

7 Valitud tehnoloogiate juurutamine ja valideerimine

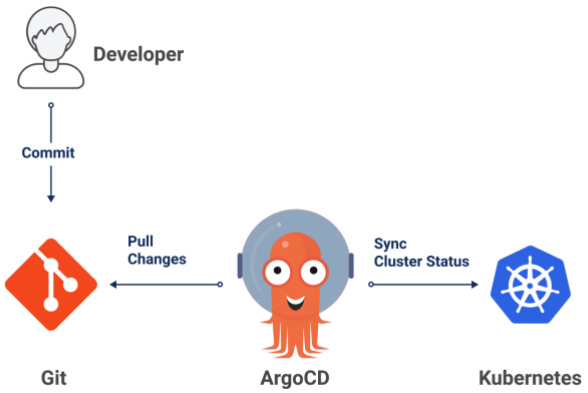
Antud peatükis kirjeldatakse kuidas analüüsi käigus välja valitud tooteid juurutati ning nende toimimist vastavalt esitatud nõuetele valideeriti.

Ettevõte on juurutanud infrastruktuuri haldamiseks GitOPS metoodika [30]. GitOPS metoodika kasutab kogu infrastruktuuri ja rakenduste haldamiseks infrastruktuur koodina lähenemist. AWS komponentide paigalduseks ja halduseks kasutatakse Terraformi ja AWS Codepipeline teenust. Allolev skeem illustreerib infrastruktuuri komponentide paigaldusprotsessi:



Joonis 11. Terraformi joonis [31].

Rakenduste paigalduseks on kasutusel ArgoCD. Allolev skeem illustreerib kubernetese ressursside paigaldusprotsessi:



Joonis 12. ArgoCD joonis [32].

Mõlemad tööriistad võtavad infrastruktuuri ja rakendusi kirjeldava koodi git-ist ja paigaldavad automaatselt koodis kirjeldatud seisu. Ka antud lõputöös käsitletavate toodete ja nende infrastruktuuri sõltuvused paigaldatakse ja seadistatakse kasutades sama lähenemist, et hoida kogu haldus ühtsena.

Kõik kolm valitud toodet vajavad eeldustena Kubernetesesse vaba ressurssi. Antud juhul leidsime kollektiivselt, et paigaldame igale tootele oma EKS sõlmegrupi kus on vähemalt kaks sõlme, et tagada kõrgkäideldavus. Põhjus miks antud rakendusi ei paigaldata juba olemasolevatele sõlmedele on selles, et pole teada täpselt kui palju lõplikus seadistuses antud rakendused ressurssi tarbivad ning see aitab vältida olukorda kus juba olemasolevatelt rakendustelt võetakse ootamatult vajalik ressurss ära. Lisaks on vaja igale tootele oma AWS S3 ämbrit ja ämbrite pääsupoliitikaid. Nende loomise koodinäited on lisas 2. Kliendil oli eelnevalt olemas juba keskne keskkond mis oligi mõeldud keskkondade üleste tööriistade paigaldamiseks, seega sai ka antud keskkond valitud uue seirepinu komponentide paigalduseks.

7.1 Keskse Grafana ja Mimiri paigaldus

Antud projekti puhul sai peale eelduste täitmist kõigepealt paigaldatud Mimir. Dokumentatsiooni järgi on soovitatud paigalduseks kasutada ametlikku Helmi pakki [33]. Kuna antud ettevõttes senimaani pole kasutatud otse Helmi pakkidest paigaldust vaid hoitakse rakenduste manifeste Yaml kujul siis otsustati sama joont jätkata, et ei tekiks erisusi koodi haldamisel. Selleks, et saada Helmi pakist rakenduse manifestid Yaml kujule on esialgu vaja Helmi pakk alla laadida. Seejärel väärtuste failis paika panna

soovitud paigalduse parameetrid ning siis templiitida Helmi pakk väärtuste faili kasutades valmis rakenduse manifestid. Näidis Helmi paki väärtuste fail on lisas 3. Seejärel sai manifestid git-i pandud ja ArgoCD kaudu paigaldatud rakenduse komponendid. Olles veendunud, et kõik Mimiri komponendid läksid ilma vigadeta käima sai järgmise sammuna suunatud esimese testkeskkonna Prometheus saatma mõõdustikku Mimiri suunas. Seejuures tasub märkida, et Mimiri kaudu oleks võimalik eristada millisesse klastrisse mõõdustik kuulub, tuleb lisada saadetavate andmetega kaasa ka parameeter mis on väärtustatud klastri nimega.

Edasi sai paigaldatud samasse keskkonda kaks isendit Grafanat. Üks toodangu jaoks kus kasutajatel on graafikute vaatamise õigus ja teine testi jaoks kus arendajad saavad toodangu jaoks arendada uusi või juba olemasolevaid graafikuid ja vaateid. Grafana puhul sai taaskasutada juba olemasolevaid rakenduse manifeste.

Grafanasse sai lisatud vastavalt juhendile uus andmeallikas. Olles veendunud, et andmed jõuavad Prometheusest Mimirisse tuli kõikidesse olemasolevatesse Grafana vaadetesse lisada juurde uus filter nimega „cluster“. Selleks, et antud filter töötaks on tarvis kõikide vaadete paneelides kasutatavates päringutes lisada uus parameeter mis väärtustatakse antud filtri valikutega. Olles need muudatused sisse viidud sai kõik vaated eksporditud Grafanast Json formaati ja pandud git-i kust siis ehitatakse uus Grafana konteiner mis sisaldab juba täiustatud vaateid.

Seejärel seadistati ka ülejäänud Prometheuse keskkonnad saatma mõõdustikku Mimirisse. Siinkohal tasub mainida ära, et Prometheuse mälu kasutus kasvab ca 10-20% kui ta seadistada andmeid välisesse tagateenusesse saatma. Lisaks kasvas iga keskkonna lisamisega Mimiri Ingester komponendi mälu kasutus 3-5GB.

Peale täiendavaid seadistamisi olles veendunud, et uus keskkond töötab stabiilselt ja ilma tõrgeteta sai antud uude keskkonda ligipääs arendajatele. Arendajate tagasisidest selgus, et mõned vaated ei töötanud korralikult aga need vead said operatiivselt likvideeritud ning seejärel võeti antud lahendus üldisesse kasutusse.

Need Grafana isendid mis olid igas keskkonnas varasemalt said skaleeritud 0 jäljendini mitte ei kustutatud ära. Kollektiivne otsus oli need jätta alles igasse keskkonda juhuks kui keskne Grafana või Mimir peaks olema kättesaamatu mõne rikke korral. Selliselt on nad ootel olekus aga ressursse ei tarbi, samas saab need mõne minutiga käima panna.

Prometheus sai seadistatud esialgu hoidma lokaalselt 30 päeva asemel test keskkondades 7 päeva ja toodangu keskkondades 14 päeva mõõdustikku.

7.2 Loki paigaldus

Loki paigaldus on oma loomult sama nagu Mimiri puhul [34]. Helmi pakile on tarvis tekitada õigete parameetritega väärtusfail ja templiitida valmis rakenduse manifestid. Peale rakenduse manifestide git-i panemist ja rakenduse paigaldust sai lisatud Loki Grafanasse andmeallikana vastavalt kasutusjuhendile.

Selleks, et logid jõuaksid Logstashist Lokisse tuli esmalt paigaldada Logstashile loki väljund pistikprogramm. Seejärel sai lisada esimeses testkeskkonnas Logstashi väljundiks Loki. Grafana ootab väga kindlate väljadega logisid mistõttu esmapilgul ei olnud võimalik korralikult logisid filtreerida. Selle tarbeks sai Logstashi seadistuses tehtud andmeväljade vastendamine, et Elasticsearchi jaoks mõeldud andmeväljade nimed vastenduks Grafana filtritele vastavaks. Seadistuse näidis lisas 4. Peale testkeskkonnas vajalike seadistuste paika saamist võis samad seadistused rakendada ka ülejäänud keskkondades. Esialgu sai jäetud paralleelselt käima nii AWS Opensearch kui ka Loki, et oleks võimalik võrrelda jõudlust ja anda arendajatele sujuvama ülemineku ühelt tootelt teisele.

Igale tootemeeskonnale sai loodud Grafanasse nende rakenduse vaadetesse vastava rakenduse logide vaated. Selliselt saab eraldada logidele ligipääsu vastavalt tootemeeskondade kaupa. Ühtlasi lihtsustas see arendajate ülemineku ühest keskkonnast teise.

Ülemineku periood antud lahenduse puhul oli kolm kuud. Peale seda sai hakatud sulgema AWS Opensearch keskkondi. Esialgu suleti test keskkonnad ning seejärel toodangu keskkonnad. Tänaseks on kogu logide haldus üle viidud Loki peale.

Logide puhul edasiste tööde näol on planeeritud järgneva aasta jooksul Filebeat ja Logstash asendamine kas Promtail või Grafana agendiga, et vähendada hallatavate tööriistade hulka. Selleks on planeeritud põhjalikum testimine käesoleva aasta kolmandasse kvartalisse, et veenduda kas selline vahetamine on võimalik või mõistlik.

7.3 Tempo paigaldus

Tempo paigalduseks sai kasutatud samamoodi nagu Mimiri ja Loki puhul Helmi pakki [35]. Helmi väärtuste fail sai paika sätitud, Helmi pakk tempIiiditud ja tekitatud manifestid git-i ülesse laetud ning peale paigaldust sai vastavalt Grafana dokumentatsioonile Tempo ka lisatud andmeallikaks.

Kuna antud lõputöö kirjutamise hetkeks polnud veel ükski arendusmeeskond valmis antud tehnoloogiat valideerima siis ei saa siinkohal ka teha kokkuvõtvat otsust kas antud tehnoloogia on sobilik või mitte. Hetkel teada olevalt on planeeritud esimesed testid käesoleva aasta kolmandasse kvartalisse.

7.4 Valideerimise kokkuvõte

Antud lõputöö raames sai kolmest paigaldatud tehnoloogiast valideeritud kaks. Paigalduse koha pealt on autoril mõned märkused. Nimelt on Grafana Labs teinud Helmi pakkide vaike väärtuste failid väga suureks. Iga toote puhul tuli paigalduseks läbi töödelda mitu tuhat rida Yaml koodi, et saada lõpuks paigalduseks vajalikud väärtused paika. Lisaks tuli peale paigaldust päris mitu korda muuta Kubernetese sõlmede suurusi, et oleks rakendustele piisavalt arvutusressurssi.

Mõõdustiku lahenduse osas on kasutajate tagasiside väga positiivne. Kadunud on killustatus ja on tekkinud võimalus võrrelda keskkondade üleselt rakenduste mõõdustikku. Jõudlus on keskse süsteemi puhul sama kiire kui eelnevalt lokaalselt, lisaks on tähelepanekuid, et Mimir on pika intervalliga päringute puhul stabiilsem kui Prometheus.

Logimise kohapealt on üldine tagasiside positiivne, küll aga oli alguses pikalt osadel arendajatel vaja ümber harjuda logide pärimisega. Loki kasutab LogQL keelt pärimiseks, Opensearch ja Kibana kasutasid KQL päringukeelt.

Logide pärimise kiirusega on tulnud kahte erinevat tagasisidet. Lühikese intervalliga päringud on sama kiired nagu seda oli vanas süsteemis. Küll aga on välja tulnud, et pika intervalliga, siinkohal räägime mitme kuu logid koos täiendavate filtritega, päringud

võtavad mitukümmend sekundit aega. Hetkel on sellised päringud aga ääre juhtumid pigem ja klient on otsustanud, et see ei ole nende jaoks probleem.

Nagu eelnevalt juba mainitud siis kahjuks ei ole antud lõputöö raames olnud võimalik valideerida Tempo toimimist.

8 Finantsanalüüs

Antud projekti raames sai antud ettevõtte keskkondades suletud kümme AWS Opensearch teenust. Enne projekti alustamist moodustas antud teenuste hind 90% süsteemi seirele kulunud infrastruktuuri kulust. Küll aga lisandus uusi teenuseid mis omakorda vajasid eeldustena uusi Kubernetese sõlmi ja AWS S3 ämbreid andmete talletamiseks. Kokku sai lisatud kesksesse asukohta juurde kuus Kubernetese sõlme suurusega r6a.2xlarge mis moodustavad kogu uue lahenduse arvutusjõudluse. Lisaks sai igale teenusele loodud oma AWS S3 ämbrid. Kui kokku arvutada lisandunud AWS teenuste kulu ja võrrelda seda eelneva seisuga siis rahaline sääst antud ettevõttele on ligi 70%.

9 Kokkuvõte

Antud lõputöö raames valmis Entigo OÜ kliendile terviklik seirelahendus mis võimaldab ette antud ajaraami vältel talletada mõõdustikku, logisid ja hajutatud jälgitavuse jälgesid. Kogutud andmeid on võimalik visualiseerida ja analüüsida ühest kesksest Grafanast. Antud lõputöö raames tehtud analüüsi põhjal valitud komponentidest sai paigaldatud kõik välja valitud tooted ning nendest valideeritud said mõõdustiku ja logide kogumise lahendused.

Mimiri näol on tegemist väga stabiilse ja töökindla tagateenusega Prometheuse saadetud andmete talletamiseks. Kasutajate tagasiside põhjal on antud toode saanud kõige paremat tagasisidet. Kasutajate suurim probleem, seire killustatus sai lahendatud Mimiri ja Loki koostöös. Lisaks sellele, et kasutajamugavus seireks paranes märgatavalt andsid antud tooted ka võimaluse teostada seiret selliselt mis varem polnud võimalik. Keskkondade ülene seire annab märksa parema ülevaate süsteemide toimimisest. Lisaks kasutajate probleemide lahendamisele võimaldasid antud tooted ka lahendada seire andmete pikaajalise talletamise probleemi. Kasutajate tagasiside põhjal võib valideeritud tehnoloogiate valiku lugeda õnnestunuks.

Olgugi, et antud lõputöö raames ei õnnestunud valideerida Tempo toimimist on vähemalt algus selles suunas tehtud, et neid andmeid oleks võimalik hakata koguma ja seeläbi süsteemiseiret veel täiustada.

Lisaks eelmainitule on klient hinnanud kõrgelt antud projektiga saavutatud senist 70% kokkuhoidu seirepinu kuludest sealjuures funktsionaalsuses ja jõudluses järele andmata.

Kasutatud kirjandus

[1] Prometheus, „Storage,“ 2023. [Võrgumaterjal]. Saadaval: https://prometheus.io/docs/prometheus/latest/storage/#operational-aspects. [Kasutatud 24.04.2023].

[2] AWS, „Opensearch Pricing,“ 2023. [Võrgumaterjal]. Saadaval: https://aws.amazon.com/opensearch-service/pricing/. [Kasutatud 28.04.2023].

[3] Splunk, „What is distributed tracing,“ 2023. [Võrgumaterjal]. Saadaval: https://www.splunk.com/en_us/data-insider/what-is-distributed-tracing.html. [Kasutatud 28.04.2023].

[4] AWS, „Overview,“ 2023. [Võrgumaterjal]. Saadaval: https://aws.amazon.com/pricing/?aws-products-pricing.sort-by=item.additionalFields.productNameLowercase&aws-products-pricing.sort-order=asc&awsf.Free%20Tier%20Type=*all&awsf.tech-category=*all. [Kasutatud 1.04.2023].

[5] Cortexmetrics, „Docs,“ 2023. [Võrgumaterjal]. Saadaval: https://cortexmetrics.io/docs/. [Kasutatud 28.04.2023].

[6] B. C. Gain, „The Great Grafana Mimir and Cortex Split,“ 2022. [Võrgumaterjal]. Saadaval: https://thenewstack.io/the-great-grafana-mimir-and-cortex-split/. [Kasutatud 28.04.2023].

[7] Cortexmetrics, „Releases,“ 2023. [Võrgumaterjal]. Saadaval: https://github.com/cortexproject/cortex/releases. [Kasutatud 28.04.2023].

[8] Cortexmetrics, „Arhcitecture,“ [Võrgumaterjal]. Saadaval: https://cortexmetrics.io/docs/architecture/.

[9] Grafana Labs, „Grafana Mimir overview,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/oss/mimir/. [Kasutatud 15.05.2023].

[10] Grafana, „About the Grafana Mimir architecture,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/mimir/latest/operators-guide/architecture/about-grafana-mimir-architecture/. [Kasutatud 28.04.2023].

[11] Grafana Labs, „Grafana Mimir components,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/mimir/latest/operators-guide/architecture/components/#grafana-mimir-components. [Kasutatud.28.04.2023].

[12] Grafana Labs, „Microservices mode,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/mimir/latest/operators-guide/architecture/deployment-modes/#microservices-mode. [Kasutatud 28.04.2023].

[13] Thanos, „Quick tutorial,“ 2023. [Võrgumaterjal]. Saadaval: https://thanos.io/tip/thanos/quick-tutorial.md/. [Kasutatud 28.04.2023].

[14] Thanos, „Releases,“ 2023. [Võrgumaterjal]. Saadaval: https://github.com/thanos-io/thanos/releases. [Kasutatud 28.04.2023].

[15] Grafana Labs, „Datasources,“ 2023. [Saadaval]. Available: https://grafana.com/docs/grafana/latest/datasources/. [Kasutatud 28.04.2023].

[16] Influxdata, „InfluxDB Clustering - High Availability and Scalability,“ 2020. [Võrgumaterjal]. Saadaval: https://www.influxdata.com/blog/influxdb-clustering/. [Kasutatud 28.04.2023].

[17] Elastic, „Elasticsearch Guide,“ 2023. [Võrgumaterjal]. Saadaval: https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html. [Kasutatud 28.04.2023].

[18] Elastic, „Amazon OpenSearch Service is not Elasticsearch,“ 2023. [Võrgumaterjal]. Saadaval: https://www.elastic.co/amazon-opensearch-service. [Kasutatud 15.05.2023].

[19] Jin, „Elasticsearch Architecture,“ 2022. [Võrgumaterjal]. Saadaval: https://medium.com/geekculture/elasticsearch-architecture-1f40b93da719. [Kasutatud 28.04.2023].

[20] Grafana Labs, „Overview,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/loki/latest/fundamentals/overview/. [Kasutatud 28.04.2023].

[21] Grafana Labs, „Components,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/loki/latest/fundamentals/architecture/components/. [Kasutatud 28.04.2023].

[22] A. Anand, „Loki vs Elasticsearch,“ 28 02 2023. [Võrgumaterjal]. Saadaval: https://signoz.io/blog/loki-vs-elasticsearch/. [Kasutatud 15.05.2023].

[23] Grafana Labs, „Loki Clients,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/loki/latest/clients/?pg=oss-loki&plcmt=resources#clients. [Kasutatud 15.05.2023].

[24] D. Khan, „Open Observability: Distributed tracing and observability,“ 2021. [Võrgumaterjal]. Saadaval: https://www.dynatrace.com/news/blog/open-observability-distributed-tracing-and-observability/. [Kasutatud 28.04.2023].

[25] K. Wakayama, „Open-Source Tracing Tools: Jaeger Vs. Zipkin Vs. Grafana Tempo,“ 2023. [Võrgumaterjal]. Saadaval: https://codersociety.com/blog/articles/jaeger-vs-zipkin-vs-tempo. [Kasutatud 28.04.2023].

[26] Jaegertracing, „Architecture,“ 2023. [Võrgumaterjal]. Saadaval: https://www.jaegertracing.io/docs/1.18/architecture/. [Kasutatud 28.04.2023].

[27] Grafana Labs, „Tempo documentation,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/tempo/latest/. [Kasutatud 28.04.2023].

[28] Grafana Labs, „Architecture,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/tempo/latest/operations/architecture/. [Kasutatud 28.04.2023].

[29] Zipkin, „Architecture,“ 2023. [Võrgumaterjal]. Saadaval: https://zipkin.io/pages/architecture.html. [Kasutatud 28.04.2023].

[30] Red Hat, „What is GitOps?,“ 2023. [Võrgumaterjal]. Saadaval: https://www.redhat.com/en/topics/devops/what-is-gitops. [Kasutatud 28.04.2023].

[31] „How to use CI/CD to deploy and configure AWS security services with Terraform,“ 2019. [Võrgumaterjal]. Saadaval: https://aws.amazon.com/blogs/security/how-use-ci-cd-deploy-configure-aws-security-services-terraform/. [Kasutatud 28.04.2023].

[32] K. Kapelonis, „Solving configuration drift using GitOps with Argo CD,“ 2020. [Võrgumaterjal]. Saadaval: https://www.cncf.io/blog/2020/12/17/solving-configuration-drift-using-gitops-with-argo-cd/. [Kasutatud 28.04.2023].

[33] Grafana Labs, „Deploy Grafana Mimir on Kubernetes,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/mimir/latest/operators-guide/deploy-grafana-mimir/. [Kasutatud 28.04.2023].

[34] Grafana Labs, „Install Grafana Loki with Helm,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/loki/latest/installation/helm/. [Kasutatud 28.04.2023].

[35] Grafana Labs, „Get started with Grafana Tempo using the Helm chart,“ 2023. [Võrgumaterjal]. Saadaval: https://grafana.com/docs/helm-charts/tempo-distributed/next/get-started-helm-charts/. [Kasutatud 28.04.2023].

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

Mina, Margus Laanem

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Keskse seireteenuse loomine rahvusvahelises tarkvaraarenduse ettevõttes“, mille juhendaja on Siim vene ja kaasjuhendaja Rein Remmel.
   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 – Terraformi koodinäited

```
resource "aws_s3_bucket" "mimir" {
  bucket = "mimir-${terraform.workspace}"
  tags = {
    Name = "mimir"
  }
}

resource "aws_s3_bucket_acl" "mimir" {
  bucket = aws_s3_bucket.mimir.id
  acl    = "private"
}
resource "aws_s3_bucket_public_access_block" "mimir" {
  bucket = aws_s3_bucket.mimir.id
  block_public_acls   = true
  block_public_policy = true
  ignore_public_acls  = true
  restrict_public_buckets = true
}

resource "aws_s3_bucket_policy" "mimir_s3_policy" {
  bucket = aws_s3_bucket.mimir.id
  policy =  <<POLICY

{
    "Version": "2012-10-17",
    "Id": "",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "${aws_iam_role.mimir.arn}"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::mimir-${terraform.workspace}",
                "arn:aws:s3:::mimir-${terraform.workspace}/*"
            ]
        }
    ]
}
POLICY
}

resource "aws_iam_role" "mimir" {
  name = "mimir"
```

```
  assume_role_policy = <<POLICY
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "<oidc provider>:sub":
"system:serviceaccount:monitoring:mimir",
                    "<oidc provider>:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
}
POLICY
}

resource "aws_iam_role_policy" "mimir" {

  name = "mimir-role"
  role = aws_iam_role.mimir.id

  policy = <<POLICY
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt",
        "kms:Encrypt",
        "kms:GenerateDataKey",
        "kms:ReEncrypt*",
        "s3:PutObject",
        "s3:GetObject",
        "s3:DeleteObject*",
        "s3:AbortMultipartUpload"
      ],
      "Resource": [ "${aws_s3_bucket.mimir.arn}",
                    "${aws_s3_bucket.mimir.arn}/*",
                    "${data.aws_kms_alias.s3.target_key_arn}" ]
    }
  ]
}
POLICY
```

Lisa 3 – Helm väärtusfailide näidised

```
global:
  extraEnvFrom:
    - secretRef:
        name: mimir-s3
alertmanager:
  enabled: false
distributor:
  nodeSelector:
    ntype: mimir2
  tolerations:
  - key: "ntype"
    operator: "Equal"
    value: "mimir2"
    effect: "NoSchedule"
ingester:
  replicas: 2
  nodeSelector:
    ntype: mimir
  tolerations:
  - key: "ntype"
    operator: "Equal"
    value: "mimir"
    effect: "NoSchedule"
  persistentVolume:
    size: 100Gi
  zoneAwareReplication:
    enabled: false
overrides_exporter:
  nodeSelector:
    ntype: mimir2
  tolerations:
  - key: "ntype"
    operator: "Equal"
    value: "mimir2"
    effect: "NoSchedule"
ruler:
  enabled: false
querier:
  replicas: 1
  nodeSelector:
    ntype: mimir2
  tolerations:
  - key: "ntype"
    operator: "Equal"
    value: "mimir2"
    effect: "NoSchedule"
query_frontend:
```

```
  nodeSelector:
    ntype: mimir2
  tolerations:
  - key: "ntype"
    operator: "Equal"
    value: "mimir2"
    effect: "NoSchedule"
query_scheduler:
  enabled: true
  replicas: 1
  nodeSelector:
    ntype: mimir2
  tolerations:
  - key: "ntype"
    operator: "Equal"
    value: "mimir2"
    effect: "NoSchedule"
store_gateway:
  nodeSelector:
    ntype: mimir2
  tolerations:
  - key: "ntype"
    operator: "Equal"
    value: "mimir2"
    effect: "NoSchedule"
  persistentVolume:
    size: 50Gi
  zoneAwareReplication:
    enabled: false
compactor:
  nodeSelector:
    ntype: mimir2
  tolerations:
  - key: "ntype"
    operator: "Equal"
    value: "mimir2"
    effect: "NoSchedule"
  persistentVolume:
    size: 100Gi
memcachedExporter:
  enabled: false
rollout_operator:
  enabled: false
minio:
  enabled: false
nginx:
  enabled: false
gateway:
  enabledNonEnterprise: true
  nodeSelector:
    ntype: mimir2
```

```
tolerations:
- key: "ntype"
  operator: "Equal"
  value: "mimir2"
  effect: "NoSchedule"
```

Lisa 4 – Logstashi seadistuse koodinäidis

```
filter {
        mutate {
          add_field => {
           "cluster" => "dev"
            }
          }
          if [kubernetes][container][name] {
            mutate {
              add_field => { "container_name" =>
"%{[kubernetes][container][name]}" }
            }
          }
          if [kubernetes][namespace] {
            mutate {
              add_field => { "namespace" => "%{[kubernetes][namespace]}" }
            }
          }
          if [kubernetes][pod][name] {
            mutate {
              add_field => { "pod" => "%{[kubernetes][pod][name]}" }
            }
          }
          if [kubernetes][node][name] {
            mutate {
              add_field => { "node_name" => "%{[kubernetes][node][name]}" }
            }
          }
          if [container][image][name] {
            mutate {
              add_field => { "image_name" => "%{[container][image][name]}" }
            }
          }
          if [host][name] {
            mutate {
              add_field => { "host_name" => "%{[host][name]}" }
            }
          }
          if [log][file][path] {
            mutate {
              add_field => { "log_file_path" => "%{[log][file][path]}" }
            }
          }
          if [kubernetes][labels][component] {
            mutate {
              add_field => { "component" =>
"%{[kubernetes][labels][component]}" }
            }
          }
```

```
          if [kubernetes][labels][kubernetes_io_instance] {
            mutate {
              add_field => { "instance" =>
"%{[kubernetes][labels][kubernetes_io_instance]}" }
            }
          }
          if [kubernetes][labels][app] {
            mutate {
              add_field => { "app" => "%{[kubernetes][labels][app]}" }
              add_field => { "job" => "%{[kubernetes][namespace]}/%{[app]}" }
            }
          } else {
            mutate {
              add_field => { "app" => "%no-app-label" }
            }
          }
```