TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Maris Salk 202851IADB

Mikroteenustel põhineva toote monitoorimise parandamise prototüüp

Bakalaureusetöö

Juhendaja: Meelis Antoi
Magistrikraad

Tallinn 2024

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: Maris Salk

04.01.2024

Annotatsioon

Antud lõputöö eesmärk on koostada SiteMinderi Guest Engagement tootele monitoorimise prototüüp, mille põhjal otsustada, millised tööriistad lahendavad kõige edukamalt Guest Engagement tootega seotud monitoorimise probleeme. Töö käigus analüüsitakse ning rakendatakse prototüübil kaks erinevat hajutatud jälgitavuse tööriista ning kaks erinevat mõõdikute visualiseerimise tööriista ja võrreldakse neid omavahel.

Hajutatud jälgitavuse tööriistade AWS X-Ray ning Grafana Tempo võrdlusest selgus, et Guest Engagement monitoorimise nõudeid täidab paremini AWS X-Ray. Mõõdikute visualiseerimise tööriistadest Amazon CloudWatch ja Grafana täidab nõudeid edukamalt Grafana.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 7 peatükki, 22 joonist, 3 tabelit.

Töö on vormistatud kasutades LaTeX märgistuskeelt ja Infotehnoloogia teaduskonna poolt soovitatud LaTeX malli lõputöö vormistamiseks.

Abstract

Observability Improvement Prototype for Microservice Based Product

The purpose of this thesis is to create a monitoring prototype for SiteMinder's Guest Engagement product, which helps to decide which tools most successfully solve monitoring issues related to the Guest Engagement product. Throughout the thesis two different distibuted tracing tools and two metric visualization tools will be analysed, implemented on the prorotype and compared with each other.

The comparison of distributed monitoring tools AWS X-Ray and Grafana Tempo revealed that AWS X-Ray better fulfills the requirements of Guest Engagement monitoring. Among the metric visualization tools Grafana meets the requirements more effectively than Amazon CloudWatch.

The thesis is written in Estonian and is 28 pages long, including 7 chapters, 22 figures and 3 tables.

This thesis is styled using LaTeX markup language and uses a LaTeX template recommended by the School of IT.

Lühendite ja mõistete sõnastik

ADOT	Amazoni OpenTelemetry andmete koguja (*AWS Distro for OpenTelemetry*)
API	Rakendusliides (*Application Programming Interface*)
ASP.NET	Serveripoolne veebirakenduse raamistik
AWS	Amazoni veebiteenused (*Amazon Web Services*)
EC2	Amazoni veebipõhine teenus, mis võimaldab jooksutada rakendusi virtuaalmasinatel AWS pilves (*Amazon Elastic Compute Cloud*)
ECS	Amazoni täielikult hallatav konteinerite orkestreerimisteenus (*Amazon Elastic Container Service*)
HTTP	Hüperteksti edastusprotokoll (*The Hypertext Transfer Protocol*)
NATS	Sõnumitele orienteeritud vahevara
NuGet	.NET'i paketihaldur
OTLP	OpenTelemetry protokoll (*OpenTelemetry Protocol*)
PromQL	Prometheusi päringukeel (*Prometheus Query Language*)
SDK	Tarkvaraarenduskomplekt (*Software Development Kit*)
SQL	Struktuurpäringukeel (*Structured Query Language*)

Sisukord

Joonistе loetelu

Tabelite loetelu

1. Sissejuhatus

Antud lõputöö eesmärk on koostada mikroteenustel põhineva monitoorimise prototüüp Guest Engagement tootele. Prototüüp võimaldab langetada informeeritud otsuse, millised monitooringu tööriistad lahendavad kõige edukamalt Guest Engagement monitoorimise probleeme. Guest Engagement on SiteMinderi toode, mis võimaldab hotellidel külastajatega vahetult suhelda nii majutusasutuses viibimise ajal kui ka enne ja pärast viibimist [1].

Töö vajadus tekkis sellest, et Guest Engagement tootel on olulisi puudujääke teenuste seisukorra ülevaates ning tõrkeotsingu efektiivsuses, mida on süvendanud integratsioonisüsteemi üleminek monoliitselt arhitektuurilt mikroteenustele.

Lõputöö koostamisel lähtutakse vaaldeldavuse kolmest sambast, milleks on logid, mõõdikud ning jäljed. Kuigi ligipääs logidele, mõõdikutele ning jälgedele üksinda ei tee süsteemi rohkem vaadeldavaks, on need siiski võimsad tööriistad, mis annavad võimaluse teha süsteemi vaadeldavust paremaks, kui neid hästi mõistetakse [2]. Käesoleval hetkel on Guest Engagement tootel täielikult rakendatud ainult logid ning süsteemi tasemel mõõdikud.

Hästi ülesehitatud monitooringusüsteem on eriti tähtis hajutatud süsteemide puhul, mil rakenduse kompleksuse tõttu muutub vigade täpsete põhjuste tuvastamine äärmiselt keeruliseks. Põhjuseid, miks reaalaja monitooringu rakendamine toob kasu, on mitu [3]:

- Suurenenud nähtavus süsteemide jõudlusesse.
- Kiirem ja paranenud tõrkeotsing.
- Täiustatud turvalisus (tekib parem arusaam, kuidas erinevad komponendid omavahel suhtlevad ning see teeb potentsiaalsete turvaohtude tuvastamise lihtsamaks).
- Kulude kokkuhoid (väheneb vajadus manuaalsele tõrkeotsingule ning logide analüüsimisele, mis säästab aega ja raha).

Töö käigus analüüsitakse kahte erinevat hajutatud jälgitavuse toodet ja kahte erinevat mõõdikute kogumise ja visualiseerimisega seotud toodet. Töö praktilises osas koostatakse prototüüp, millele rakendatakse kõik neli monitooringutööriista ning võrreldakse neid omavahel.

Lõputöö visiooniks on, et prototüübi põhjal on võimalik langetada informeeritud valik, milline hajutatud jälgitavuse tööriist ning milline mõõdikute visualiseerimise tööriist täidab kõige edukamalt Guest Engagement monitooringu nõudeid.

1.1 Metoodika

Lõputöö eesmärgini jõudmiseks kirjeldatakse kõigepealt Guest Engagement olemasolevat monitooringu lahendust ning tuuakse välja eksisteeriva monitooringuga seotud probleemid. Samuti kirjeldatakse töö skoop.

Seejärel määratakse nõuded prototüübile ning uutele monitooringu tööriistadele. Nõuded määratakse koostöös tulevaste kasutajatega ning sarnaseid lahendusi uurides. Samuti määratakse tehnoloogia, millega prototüüp tuleb koostada ning viiakse läbi praktilises osas rakendavate monitooringutööriistade omaduste analüüs.

Nõuete põhjal koostatakse prototüüp rakendus, millele rakendatakse eelnevalt analüüsitud monitooringu tehnoloogiaid. Rakendatud tehnoloogiaid analüüsitakse ning võrreldakse omavahel.

2. Probleemi ülevaade

Traditsiooniliste monoliitsete rakendustega tõrkeotsing on lihtsustatud: tõrkeotsinguks ning analüüsiks koguvad infot logi failid, probleeme tuvastatakse läbi staatiliste graafikute ning hoiatuste ning rakendus ise on üldiselt hästi mõistetav. Mikroteenustega on aga rohkem liikuvaid osasid, süsteem on dünaamiline ja selle osad on lõdvalt seotud ning lühiealised. Süsteem võib ebaõnnestuda paljudel erinevatel ja ettearvamatutel viisidel. Vajalik on jälgida rakendust, võrku ja infrastruktuuri [4]. Traditsioonilised tõrkeotsingu meetmeted ei ole mikroteenuste arhitektuuril piisavad ja tuleb kautada lisameetmeid.

Paremat vaadeldavust mikroteenuste arhitektuuriga tootel aitavad saavutada vaadeldavuse kolm sammast [5]:

- Mõõdikud - mõõdikud viitavad kvantitatiivsetele mõõtudele, mis annavad ülevaate süsteemi käitumisest, jõudlusest ja tervisest. Mõõdikuid tavaliselt kogutakse ja salvestatakse aja jooksul.
- Logid - logid on sündmuste, tegevuste ja sõnumite kirjed, mis on genereeritud tarkvarasüsteemi erinevate komponentide poolt. Need sõnumid võivad hõlmata veateateid, hoiatusi, teabesõnumeid ja muid asjakohaseid andmeid, mis on kasulikud süsteemi seisundi ja jõudluse jälgimiseks ja analüüsimiseks.
- Jäljed - jälg on sündmuste jada, mis kujutab otsast lõpuni päringu voolu läbi hajutatud süsteemi. Jäljed on hajutatud jälgimise oluline osa, mis aitab monitoorida ning teostada tõrkeotsingut kompleksetele süsteemidele

2.1 Eksisteeriv lahendus

Lõputöö koostamise hetkel on Guest Engagement tootel vaadeldavuse kolmest sambast (logid, jäljed ja mõõdikud) rakendatud Amazon CloudWatch logid ning Prometheus süsteemi tasemel mõõdikute kogumiseks. Amazon CloudWatch võimaldab monitoorida kasutaja AWS ressursse ja rakendusi reaalajas ning Prometheus on tööriist mõõdikute kogumiseks ja hoiustamiseks aeg-seeria andmetena [6][7]. Jälgede kogumiseks on osaliselt rakendatud OpenTelemetry, mis on reaalaja monitooringu raamistik ja tööriist, mis võimaldab koguda telemeetria andmeid nagu jäljed, mõõdikud ning logid [8].

Lõputöö koostamise hetkel kogub Prometheus ainult süsteemi tasemel mõõdikuid nagu protsessori ja mälu kasutus. Graafikute esitlemiseks kasutatakse Grafanat. Kuigi osaliselt on rakendatud OpenTelemetry jälgede kogumiseks, siis puudub tööriist, mis aitab kogutud jälgedest aru saada ja probleemi põhjuseid lihtsamini tuvastada. Seega kolmest sambast on täielikult rakendatud ainult üks.

Seoses hetkel kasutusel oleva monitooringusüsteemiga on tuvastatud järgnevad probleemid:

- Arendajad ei saa ajakohast teavet teenustes tekkivate vigade kohta. Ajutise lahendusena on seadistatud Slacki teavitused, mis kord päevas pärivad Cloudwatchi logidest eelmise päeva vigade kogusumma teenuste lõikes.
- Puuduvad hoiatused jõudlusega seotud probleemide ja intsidentide puhul.
- Juhul kui on vigu, siis arendajad peavad iga päev minema manuaalselt logisid läbi vaatama, et näha, mis teenustes juhtus.
- Tavaliselt läheb vaja mitut arendajat, et aru saada, mis on läinud katki ja tõrkeotsing on väga aeganõudev.
- Tihti peab vea põhjuse tuvastamiseks teoritiseerima, sest käesolevad vahendid ei võimalda täielikult vea põhjust üles leida.
- Väga raske on tuvastada anomaaliaid (näiteks järsk tõus imporditavate andmetes).
- Raske on tuvastada, kui midagi jääb juhtumata.
- Pole võimalik tuvastada pudelikaelu.
- Jõudlusprobleemide puhul on väga raske leida nende põhjust.
- Puudub arusaam, kui palju andmeid läbi teenuste käib.

2.2 Lahenduse skoop

Lõputöö raames käsitletakse just Guest Engagement tagarakenduse vaadeldavust, sest see on toote osa, millega töö autor igapäevaselt kõige rohkem kokku puutub. Kasutajaliidese ja eessüsteemi monitoorimisega seotud osad jäävad töö skoobist välja.

Et otsustada, millised tooted lahendavad ülaltoodud probleemid kõige paremini, koostatakse Guest Engagement tootele sarnase keskkonnaga lihtsustatud prototüüp, millele rakendatakse kaks erinevat hajutatud jälgitavuse tööriista ning kaks erinevat mõõdikute visualiseerimise tööriista ja analüüsitakse tulemust.

Võrreldavate monitooringu tööriistade valik võeti SiteMinderi poolt eelnevalt heaks kiidetud tehnoloogiate seast ning nendeks on Amazon ja Grafana. Nii Amazon kui ka Grafana pakuvad nii hajutatud jälgitavuse tööriistu (AWS X-Ray ja Grafana Tempo) kui ka mõõdi-

kute visualiseerimise tarkvara (AWS Cloudwatch ja Grafana) ning mõlemad on ka väga levinud ja tuntud tööriistad, mistõttu on need head kandidaadid. Samuti on mõlemad tooted mingil määral ka Guest Engagement tootele juba rakendatud, mis aitab hoida monitooringu tööriistu vähem hajutatuna.

3. Lahenduse analüüs

Käesolevas peatükis määratakse nõuded prototüübile ning monitooringu tööriistadele. Nõuded määrati koostöös tulevaste kasutajatega ning sarnaseid lahendusi uurides. Samuti tuuakse välja tehnoloogiad, millega prorotüüp koostatakse.

3.1 Prototüübi nõuded

Prototüübi peamine eesmärk on tekitada olukordi, millega testida monitooringu tööriistade funktsionaalsust. Seega kasutatakse prototüübis pseudoandmeid ning tekitatakse kuntslikke olukordi, et näha, kuidas erinevad tööriistad samade olukordade puhul käituvad.

Monitooringu prototüübile on määratud järgmised funktsionaalsed nõuded:

- Http päringu kaudu peab saama käivitada ahela, mis läbib mõlema mikroteenuste kõik punktid (http päringud, andmebaasi päringud, NATS sõnumid).
- Http päringu kaudu peab saama tekitada veaolukorra.
- Http päringu kaudu peab saama tekitada pudelikaela.

Monitooringu prototüübile on määratud järgmised mittefunktsionaalsed nõuded:

- Peab kasutama sarnast tehnoloogiat Guest Engagement tootele (tehnoloogia on kirjeldatud peatükis 3.3).

3.2 Nõuded monitooringu tööriistadele

Käesoleva lõputöö raames on jälgitavuse tööriistale seatud järgmised funktsionaalsed nõuded:

- Jälgitavuse tööriistad peavad integreeruma OpenTelemetry'ga.
- Toetatud on sündmuste esitamine visualiseeritud kujul.
- Valitav tööriist peab aitama tuvastada pudelikaelu.
- Tööriist peab esitama selgelt veaolukordi.
- Jäljed peavad olema esitatud lihtsasti eristavana ning arusaadavana.

Olulisi funktsionaalseid nõudeid jälgitavuse tööriistale on veelgi, kuid neid käesoleva töö raames ei verifitseerita. Nendeks on:

- Uued tööriistad ei tohi põhjustada jõudlusprobleeme.
- Tööriistad peavad olema skaleeritavad (toetama praegust ja tulevast koormust).
- Võiks olla ka anomaaliate tuvastamise funktsionaalsus.

Mõõdikute esitamise tööriistale on seatud järgmised funktsionaalsed nõuded:

- Süsteemi tasemel mõõdikute esitamise funktsionaalsus.
- Kohandatud mõõdikute esitamise funktsionaalsus.
- Peab olema võimalus seadistada hoiatusi ja teavitusi.

Nii jälgitavuse kui ka mõõdikute esitamise tööriistadele on seatud järgmised mittefunktsionaalsed nõuded:

- Tööriistad peaks olema lihtsasti õpitavad ja kasutatavad.
- Uutel tööriistadel võiks olla põhjalik dokumentatsioon ning tugi.

3.3 Prototüübi tehnoloogia

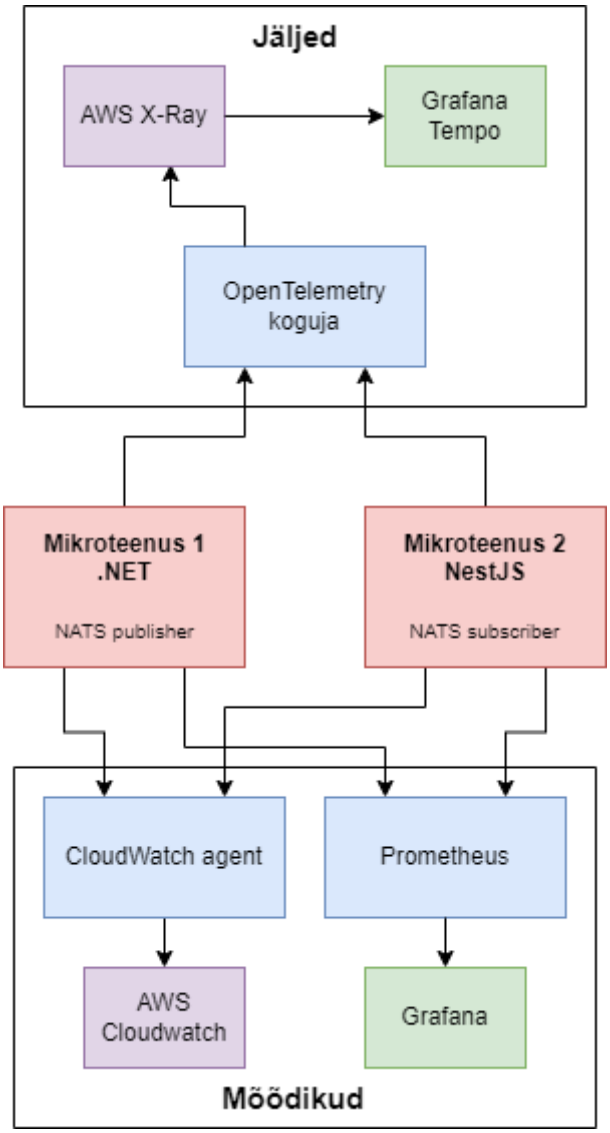
Prototüübi tehnoloogia valikul võetakse arvesse, et see peab olema tehnoloogiliselt võimalikult sarnane päris Guest Engagement toote tehnoloogiale.

Prototüübiks koostatakse kaks mikroteenust. Üks mikroteenus koostatakse .NET raamistikus ja C# programmeerimiskeeles. .NET on Microsofti loodud avatud lähtekoodiga arendajate platvorm, millega saab luua erinevaid tüüpi rakendusi [9]. Teine mikroteenus luuakse NestJS raamistikus ning Typescript programmeerimiskeeles. NestJS on raamistik, mis võimaldab ehitada efektiivseid ja skaleeritavaid Node.js serveripoolseid rakendusi [10]. Node.js on avatud lähtekoodiga serverikeskkond [11].

Mikroteenused suhtlevad omavahel NATS tehnoloogia abil. NATS võimaldab tarkvararakenduste ja -teenuste vahelist andmevahetust segmenteeritud sõnumite kujul [12]. Andmebaasiks kasutatakse mitterelatsioonilist MongoDB andmebaasi ning jälgede kogumiseks rakendatakse OpenTelemetry.

Prototüübi mikroteenused käitakse Amazoni EC2 serveritel. EC2 on veebiteenus, mis pakub turvalist ja skaleeritavat arvutusmahtu pilves. EC2 kaudu saab käivitada virtuaalseid servereid, konfigureerida turavlisusseadeid ja võrgundust ning hallata salvestusruumi [13].

Ülevaade prototüübi planeeritavast monitooringu arhitektuurist on toodud joonisel 1.



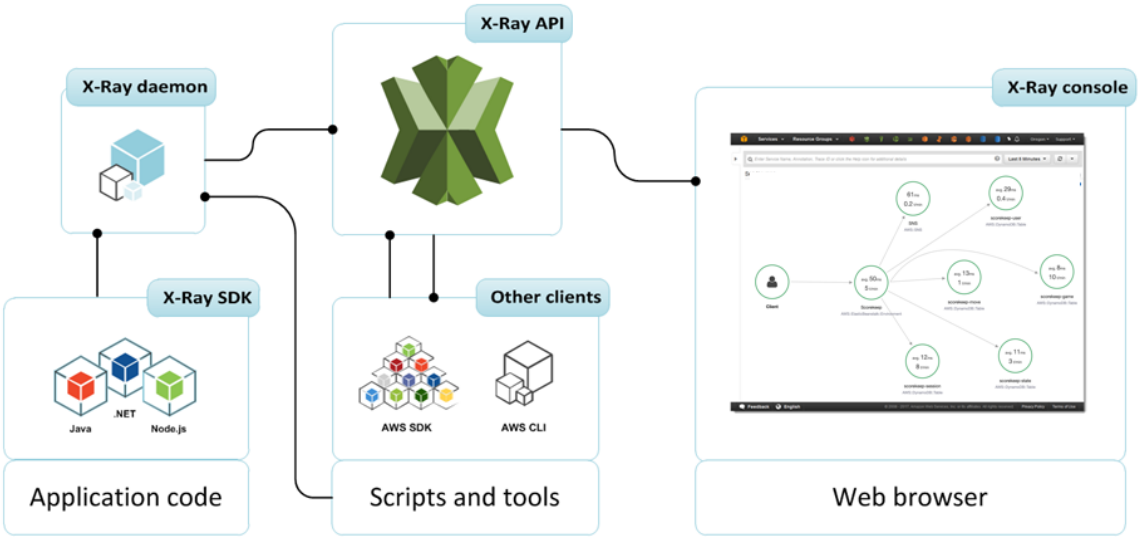
Joonis 1. Prototüübi monitooringu arhitektuur.

4. Monitooringu tööriistade omaduste analüüs

Järgenvalt toimub prototüübil rakendavate monitooringu toodete omaduste analüüs, mille eesmärk on selgitada, mida iga toode võimaldab teha ning kuidas need toimivad.

4.1 AWS X-Ray

AWS X-Ray on Amazoni teenus, mis kogub andmeid rakendustes tehtavate päringute kohta ning pakub tööriistu, millega saab vaadata ja filtreerida kogutud andmeid ning võimaldab saada teenuse ülevaade, et tuvastada probleeme ja optimiseerimisvõimalusi. AWS X-Ray kaudu näeb detailset infromatsiooni iga jälitatud (inglise keeles *traced*) päringu ja nende vastuste kohta [14]. Joonisel 2 on näidatud AWS X-ray arhitektuur ning andmete liikumine.



Joonis 2. AWS X-Ray arhitektuur [14].

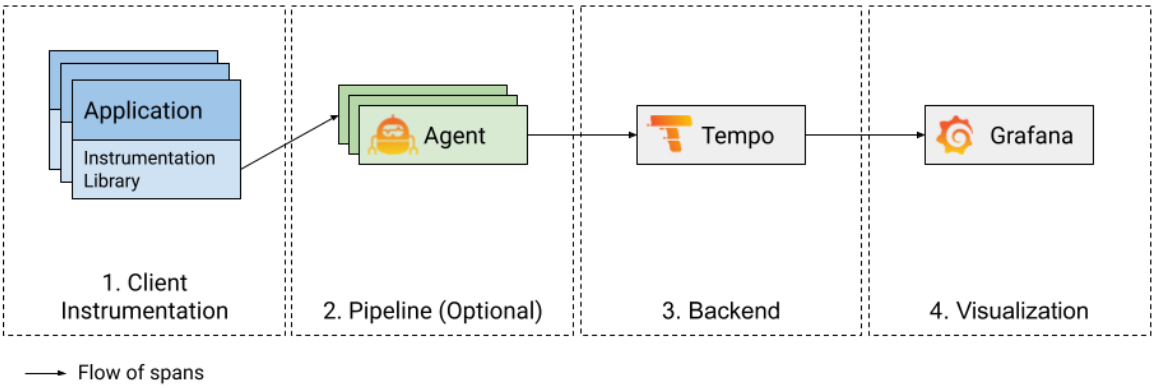
AWS X-Ray toote omadusteks on [15]:

- Otsast lõpuni teenustevaheline jälgitavus.
- Toetab rakendusi, mis töötavad teenustel Amazon EC2, Amazon ECS, AWS Lambda ja AWS Elastic Beanstalk.
- Toetab jälgede kogumist rakendustel, mis on kirjutatud Node.js'is, Javas ja .NET'is.
- Võimaldab seada sobiva jälgede kogumise diskreetimissageduse.

- Jälgede põhjal koostatakse teenuste kaart, mis näitab teenustevahelisi ühendusi ning agregeeritud andmeid iga teenuse kohta, sealhulgas latentsusaeg ning tõrgete arv.
- Võimaldab lisada annotatsioone rakenduse konkreetsetest komponentidest või teenustest väljastatud andmetele. Ärispetsiifilise info lisamine aitab paremini diagnoosida probleeme.

4.2 Grafana Tempo

Grafana Tempo on avatud lähtekoodiga laiaulatuslik hajutatud jälgimise tööriist. Tempo on kuluefektiivne ning vajab ainult objektisalvestit, et opereerida [16]. Joonisel 3 on näidatud Grafana Tempo andmevoo diagramm.



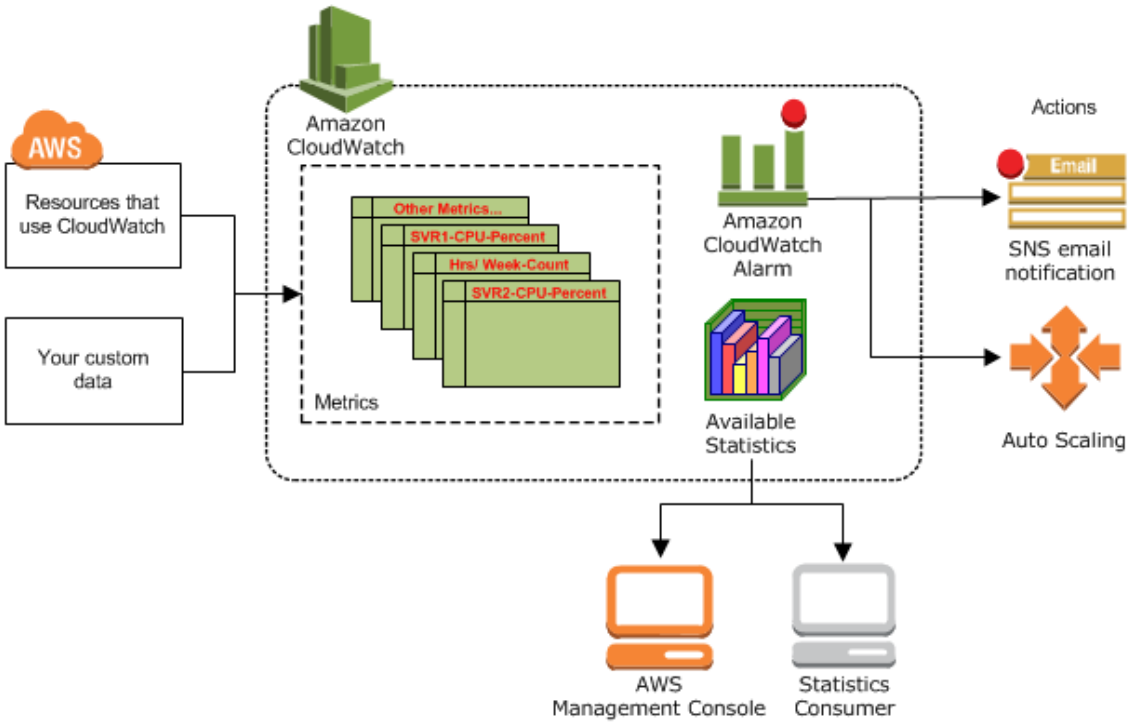
Joonis 3. Grafana Tempo andmevoo diagramm [17].

Grafana Tempo toote omadusteks on [16], [17], [18]:

- Jälgede otsimine.
- Mõõdikute genereerimine jälgedest.
- Jälgede ühendamine logide ja mõõdikutega.
- Integreeritud sügavalt Grafana, Mimiri, Prometheusi ja Lokiga (nt võimaldab hüpata logidest või mõõdikutest otse jälgede vaatesse).
- Tempo suudab töödelda levinumaid avatud lähtekoodiga jälgede protokolle nagu näiteks Jaeger, Zipkin ja OpenTelemetry.
- Ainus sõltuvus on objektisalvesti, mis võimaldab taskukohast pikaajalist jälgede säilitamist.

4.3 Amazon Cloudwatch mõõdikud

Amazon CloudWatch on põhimõtteliselt mõõdikute hoidla. AWS-i teenus (näiteks Amazon EC2) paneb mõõdikuid hoidlasse ning kasutaja saab statistikat nende mõõdikute põhjal. Kui panna hoidlasse oma kohandatud mõõdikud, siis saab ka nende kohta statistikat hankida [19]. Joonisel 4 on toodud CloudWatch'i arhitektuur.



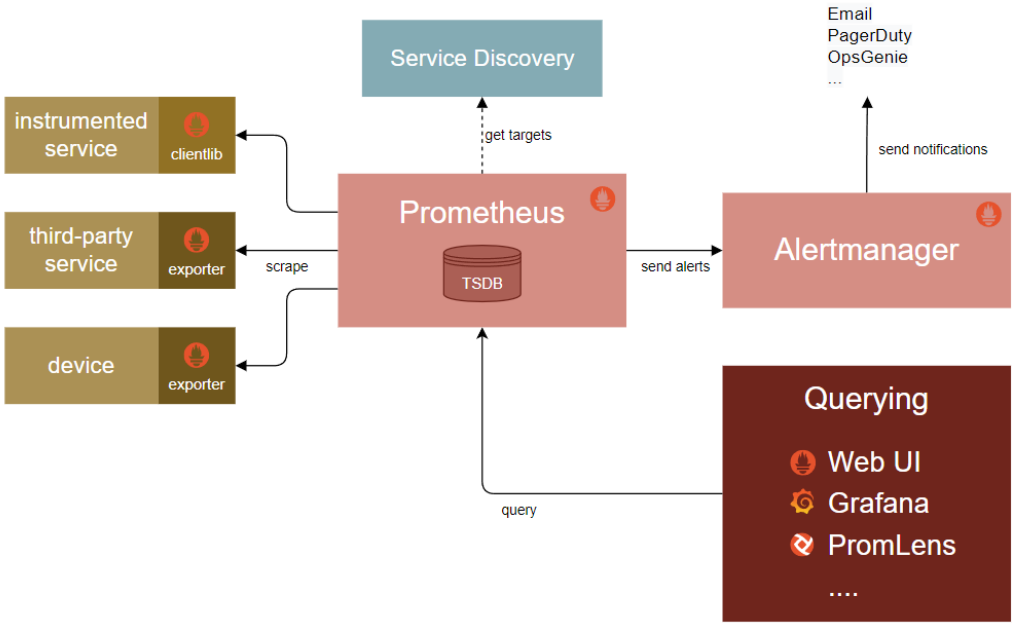
Joonis 4. CloudWatch arhitektuur [19].

Amazon CloudWatch mõõdikute omadusteks on [20]:

- Paljud Amazoni teenused pakuvad vaikimisi tasuta mõõdikuid ressursside jaoks ning see on automaatselt sisse lülitatud.
- Võimaldab rakenduse tasemel kohandatud mõõdikuid avaldada.
- Mõõdikuid hoitakse 15 kuud.
- Graafikute loomiseks konsoolis kasutades CloudWatch Metrics Insights'i, mis on suure jõudlusega SQL-päringumootor.

4.4 Grafana mõõdikud

Grafana on avatud lähtekoodiga tarkvara, mis võimaldab pärida, visualiseerida, seada hoiatusi ning uurida mõõdikuid. Joonisel 5 on toodud mõõdikute saatmine Grafana Cloudi kasutades Prometheusi.



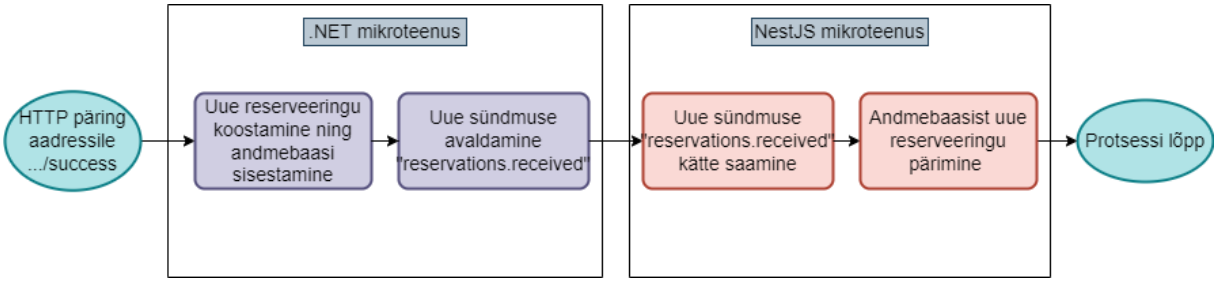
Joonis 5. Mõõdikute saatmine Prometheusist Grafanasse [21].

Grafana Cloud mõõdikute omadusteks on [22]:

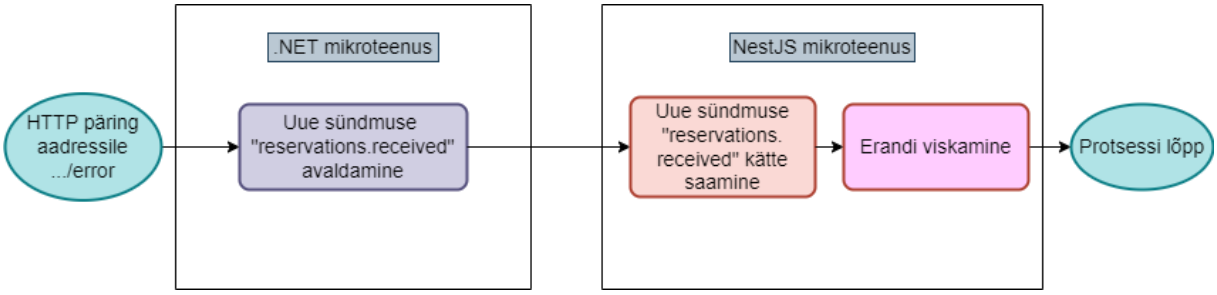
- Skaleeritav pärimine ning hoiustamine.
- Globaalne vaade - saab teha päringuid üle mitmete andmekeskuste ühest kohast.
- Palju erinevaid intergratsioone, millega saab eelkonfigureeritud Prometheusil ja Grafanal põhineva vaadeldavuse pinu üles seada väga kiirelt.
- Võimaldab visualiseerida välisallikates hoiustatavaid andmeid.
- Päringuid tehakse PromQL päringukeelega.

5. Prototüübi lahendus

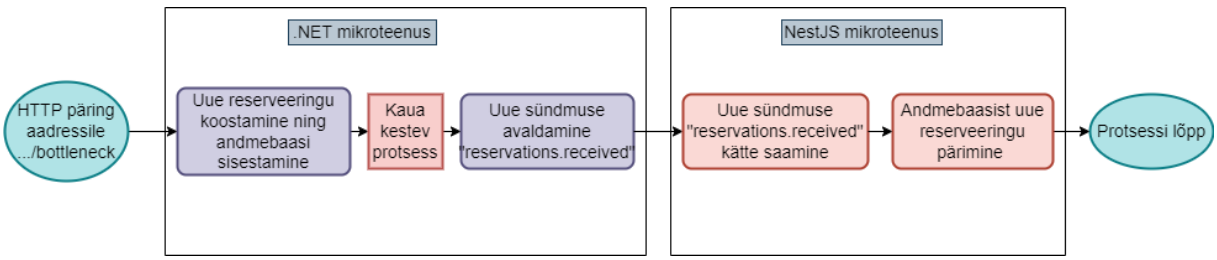
Prototüübi jaoks koostatakse kaks mikroteenust: üks .NET'is ning teine NestJS'is. Hajutatud jälgitavuse nõuete verifitseerimiseks tekitatakse prototüüpidesse kolm põhilist kasutusjuhtu: edukas protsess, protsess, mis tekitab veaolukorra ning protsess, mis tekitab pudelikaela. Eduka protsessi voodiagramm on toodud joonisel 6, veaolukorraga voodiagramm joonisel 7 ning pudelikaelaga voodiagramm joonisel 8.



Joonis 6. Prototüübi eduka protsessi voodiagramm.



Joonis 7. Prototüübi veaolukorraga protsessi voodiagramm.



Joonis 8. Prototüübi pudelikaelaga protsessi voodiagramm.

5.1 OpenTelemetry rakendamine

Jälgede ning mõõdikute kogumiseks tuleb rakendada mõlemale mikroteenusele OpenTelemetry. OpenTelemetry on kollektsioon API-dest, SDK-dest ning tööriistadest, millega

instrumenteerida, genereerida, koguda ning eksportida telemeetria andmeid (mõõdikud, jäljed ja logid) [8].

5.1.1 .NET OpenTelemetry rakendamine

.NET rakenduses OpenTelemetry rakendamiseks laetakse alla vajalikud OpenTelemetry NuGet teegid ning seadistatakse koodis OpenTelemetry. Osade teekide puhul saab kasutada automaatset instrumenteerimist, mis automaatselt loob jälgesid ning mõõdikuid (näiteks ASP.NET Core loob jälgi ja mõõdikuid sissetulevate HTTP päringute kohta) [23]. Käesoleva prototüübi puhul rakendatakse ASP.NET Core, AWS, HTTP kliendi ning MongoDB instrumenteerimise teegid.

Kõikide tehnoloogiate jaoks pole automaatse instrumenteerimise teeke loodud ning sellisel juhul luuakse jäljed käsitsi. Nii on näiteks NATS tehnoloogia puhul, millega prototüübi mikroteenused omavahel suhtlevad. Et NATS tehingud oleks osa jäljest, siis luuakse sõnumi avaldamisel uus tegevus (inglise keeles *activity*), mille tulemusel tekib uus alamjälg (inglise keeles *span*). Et jälg jätkuks järgmises teenuses, mis sõnumi kätte saab, pannakse NATS sõnumi päisesse kaasa jälje kontekst, et järgmine teenus oskaks jälge jätkata.

Telemeetria andmete visualiseerimiseks ning analüüsimiseks eksporditakse andmed OpenTelemetry kogujasse. Käesoleva prototüübi raames võetakse kasutusele OpenTelemetry Protocol (OTLP) eksportija, mis on üldotstarbeline telemeetriaandmete edastamise protokoll, mis on loodud OpenTelemetry projekti raames [24].

5.1.2 NestJS OpenTelemetry rakendamine

NestJS rakenduses OpenTelemetry rakendamiseks kasutatakse Node SDK'd ning automaatse instrumenteerimise teeke sarnaselt .NET'ile. Prototüübi raames kasutatakse HTTP, AWS, Expressi, Nesti ning MongoDB automaatse instrumenteerimise teeke. Express on minimalistlik veebi raamistik node.js'ile. [25] Telemeetria eksportimiseks kasutatakse OTLP jälgede eksportijat nagu .NET rakenduses.

OpenTelemetry rakendamisel on oluline, et jälg jätkuks teenusest teenusesse. HTTP päringute puhul teeb selle instrumenteerimise teek automaatselt ära, kuid NATS sõnumite puhul tuleb jälje kontekst sõnumi päisest kätte saada ning propageerida.

5.2 Telemeetria andmete koguja seadistamine

OpenTelemetry edastamiseks X-Ray'sse ning Grafanasse, on kõigepealt vaja seadistada OpenTelemetry andmete koguja, mis saadab telemeetria andmed monitooringu tööriistadele edasi. Selleks kasutatakse ADOT Collector'it (*AWS Distro for OpenTelemetry Collector*). ADOT Collector pannakse tööle EC2's ning seadistatakse AWS mandaat, et AWS X-Ray teaks, et andmete saatja on autoriseeritud seda tegema.

Telemeetria andmete saatmiseks Grafanasse on palju võimalusi. Üheks võimaluseks on X-Ray plugina kaudu seadistada Grafanas andmeallikaks AWS X-Ray, mille kaudu saab telemeetria andmed otse X-Rayst. Et Grafana saaks AWS'ist telemeetria andmed kätte, seadistatakse Grafanas AWS mandaat.

5.3 Prometheusi seadistamine

Mõõdikute ilmumiseks Grafanasse seadistatakse Prometheus, mis korjab rakenduste avaldatud mõõdikuid. Prometheus pannakse jooksma eraldi EC2 serverile ning seadistatakse korjama mõõdikuid aadressilt "/metrics" iga 5 sekundi tagant mõlemalt mikroteenuselt. Seejärel lisatakse Prometheus Grafanasse andmeallikaks.

.NET mikroteenus pannakse avaldama kolme kohandatud mõõdikut prometheus-net NuGet paketi abiga ning NestJS mikroteenuses pannakse avaldama kahte kohandatud mõõdikut prom-client paketi abiga.

5.4 Cloudwatch agendi seadistamine

Kohandatud mõõdikute ilmumiseks CloudWatchi tuleb seadistada CloudWatch agent, mis edastab andmed CloudWatchi. CloudWatchi agent pannakse jooksma mikroteenuste EC2 serveritele Linux teenusena.

Erinevalt Prometheusist CloudWatch agent ei korja mõõdikuid ise, vaid need tuleb sinna ise lükata. Seda on võimalik teha kasutades StatsD või collectd protokolle. StatsD on populaarne monitooringu lahendus koodi instrumenteerimiseks kohandatud mõõdikutega [26]. Collectd on taustaprogramm, mis kogub perioodiliselt süsteemi ja rakenduse mõõdikuid [27]. Prototüübi raames võetakse kasutusele StatsD. Mõlemas mikroteenuses on vaja alla laadida ning kasutusele võtta spetsiaalsed StatsD paketid, mis aitavad avaldada mõõdikuid ning saata need CloudWatch agendile.

5.5 Mõõdikute seadistamine

Mõõdikute nõuetele vastavuse verifitseerimiseks avaldatakse kohandatud loendur mõõdikuid iga .NET mikroteenusesse tehtud päringu kohta. Iga joonistel 6-8 toodud protsessi kohta on oma mõõdik. Samuti avaldatakse kohandatud mõõdik iga NestJS mikroteenuses kätte saadud NATS sõnumi kohta ning iga tekkinud erandi kohta.

Lisaks seadistatakse CloudWatchis ning Grafanas süsteemi tasemel mõõdikute kogumine. CloudWatchis esitatakse süsteemi tasemel mõõdikuid AWSis hoitavatele serveritele automaatselt. Kuna loodud mikroteenused jooksevad AWSi EC2 serveritel, siis ei pea nende üles seadmiseks ekstra midagi tegema. Grafanas süsteemi tasemel mõõdikute üles seadmine on samuti väga lihtne - tuleb lisada CloudWatch Grafana andmeallikaks ning seejärel on võimalik importida eelkonfigureeritud graafikud erinevatele CloudWatchi teenustele.

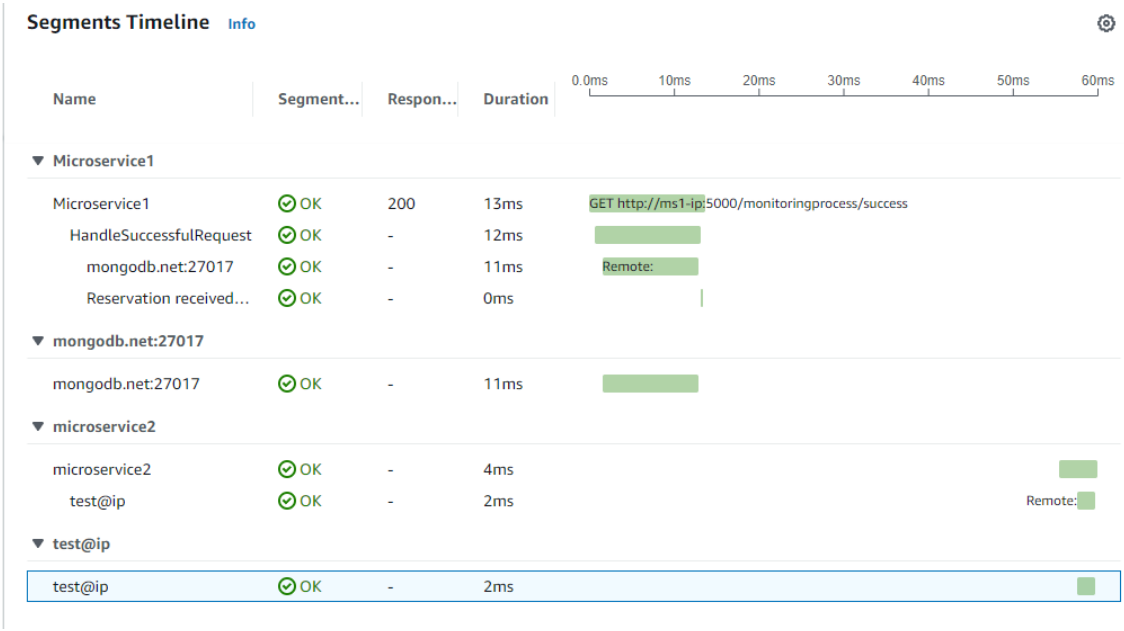
Seejärel seadistatakse hoiatus CloudWatchis ning Grafanas NestJS mikroteenuses erandeid loeandavale mõõdikule - hoiatus saadetakse kui minuti jooksul on toimunud rohkem kui viis erandit.

6. Monitooringu tööriistade võrdlus

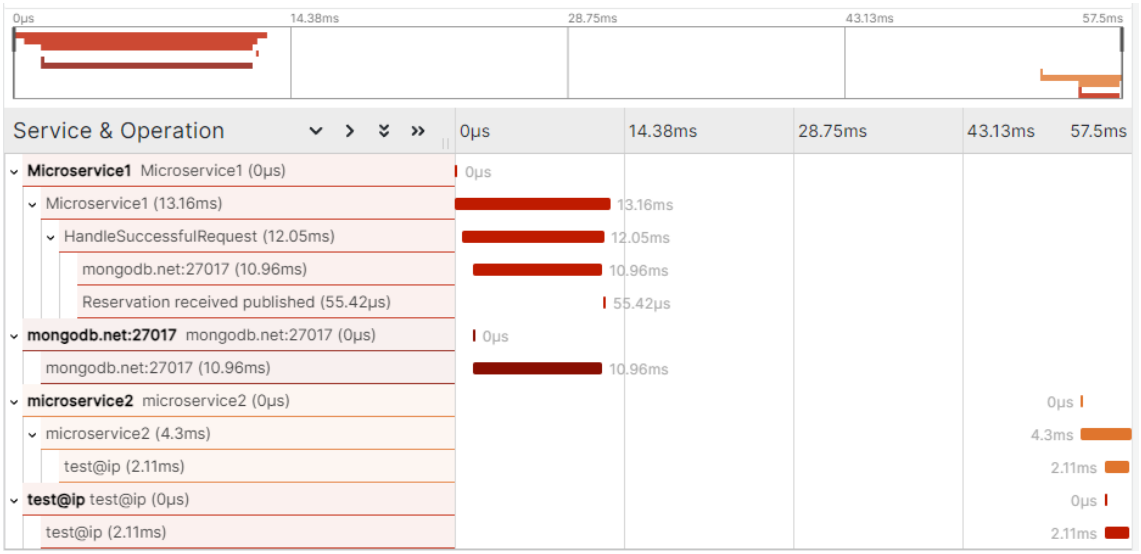
Hajutatud jälgitavuse tööriistu ning mõõdikute visualiseerimise tööriistu võrreldakse peatükis 5 toodud protsesside põhjal, mis tekitavad jälgesid ning kohandatud mõõdikuid. Võrdluseks kasutatakse parema loeatvuse huvides süsteemide heledaid taustasid.

6.1 AWS X-Ray vs Grafana Tempo jäljed

AWS X-Ray ja Grafana Tempo jälgede kujutamist ning kasutatavust hinnatakse eduka protsessi põhjal, mis on kujutatud joonisel 6. Pudelikaelaga ja erandiga protsessi analüüs toimub eraldi. X-Ray ning Tempo eduka protsessi jäljed on toodud vastavalt joonistel 9 ja 10.



Joonis 9. AWS X-Ray eduka protsessi jäljed.



Joonis 10. Grafana Tempo eduka protsessi jäljed.

Esmapilgul peale vaadates tunduvad Grafana ja AWS X-Ray kasutajaliidesed jälgede kujutamisel paigutuse poolest väga sarnased, kuid X-Ray jäljed näevad puhtamad välja kui Tempo jäljed. Mõlema tööriista puhul on jäljed kujutatud samas järjekorras ning mikroteenuste protsessid on selgesti eristatud, samuti on eristatud ka andmebaasi päringud. Mõlemad tööriistad kujutavad kõiki vajalikke jälgesid piisava selgusega.

Erinevus tuleb jälgede peale klikates, kus X-Ray puhul avanevad jälgede metadata ning muud detailid jälgedest paremal, kuid Grafanal avanevad detailid otse jälje all, mistõttu saab neid kõiki ka paralleelselt avada. X-Ray jälgede metadata on selgemalt segmenteeritud ning info on kujutatud alati samas kohas. Tempo puhul on detailid kujutatud üksteise all ja välja tuuakse ainult need alamosad, kus on detaile.

Mõlemad tööriistad võimaldavad vaadelda ka jälgede tooreid andmeid, mille põhjal koostatakse ajajoon, kuid X-Ray puhul näeb andmeid kasutajaliidesest ning Tempos tuleb need alla laadida. Omadusi võrdlev kokkuvõttev tabel on esitatud tabelis 1.

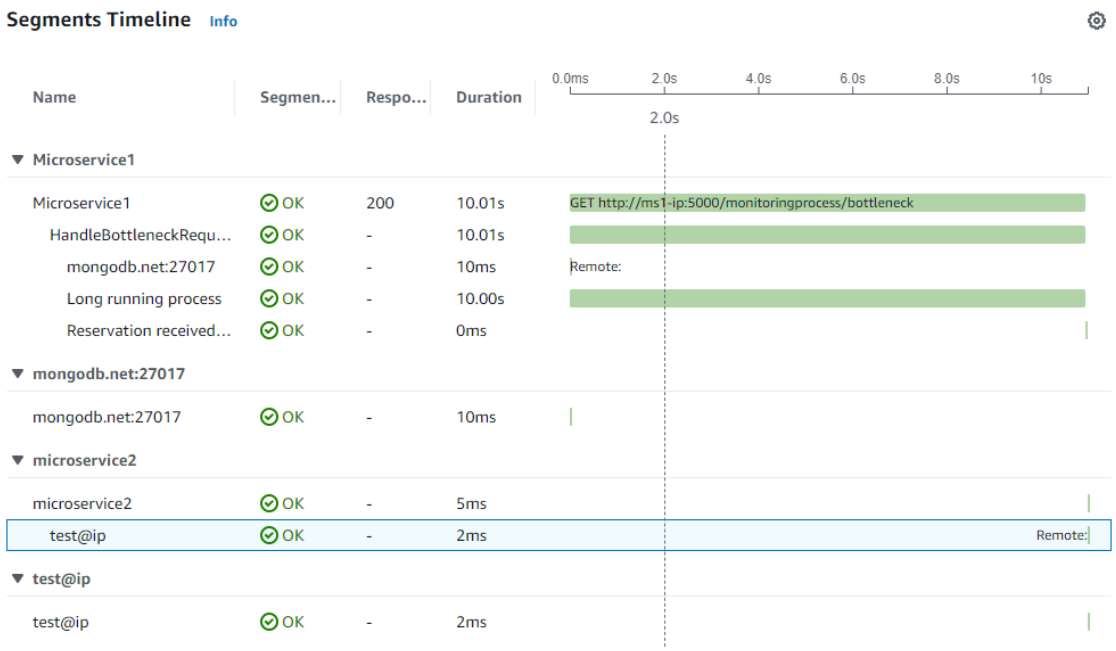
Omadus	AWS X-Ray	Grafana Tempo
Jäljed rohkem eristatavad teenuste lõikes		
Puhtam kasutajaliides		
Saadaval rohkem jälgede detaile		
Erinevate jälgede metadatat saab paralleelselt avada		
Saab valida info, mida kuvatakse jälje ajajoonel		
Võimalik filtreerida ühe jälje lõikes		
Võimalik näha teenuste kaarti		

Tabel 1. AWS X-Ray ja Grafana Tempo võrdlemine.

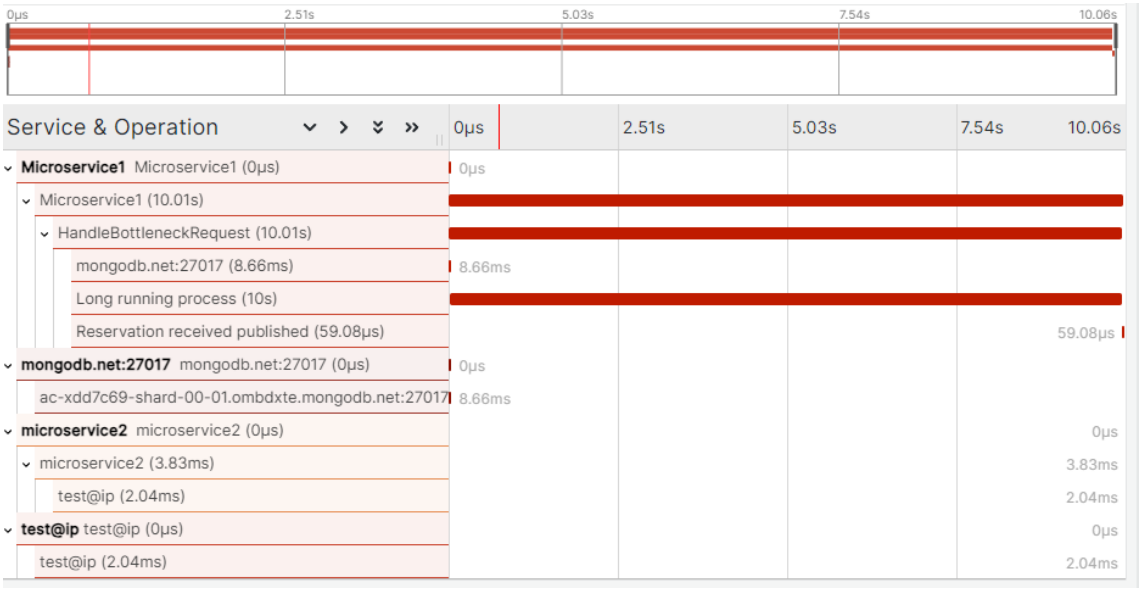
6.1.1 Pudelikaelaga protsessi kujutamine

X-Ray pudelikaelaga protsessi jälje ajajoone kuvatõmmis on joonisel 11 ning Tempo kuvatõmmis joonisel 12. Jäljed on genereeritud joonisel 8 esitatud pudelikaela tekitava protsessiga.

Nii X-Ray kui ka Grafana puhul on selgelt näha protsess, mis võttis kauem aega ning mille taha programm toppama jäi. Samuti on mõlema puhul näha ka, kui kaua protsess aega võttis. Vaikimisi kumbki tööriist ei hoiata pikalt jooksnud protsessi eest, kuid AWS X-Ray teenustekaart näitab, et .NET mikroteenuses on pikem latentsusaeg kui teistes teenustes. X-Ray teenustekaart on toodud joonisel 15.



Joonis 11. AWS X-Ray pudelikaelaga protsessi jäljed.



Joonis 12. Grafana Tempo pudelikaelaga protsessi jäljed.

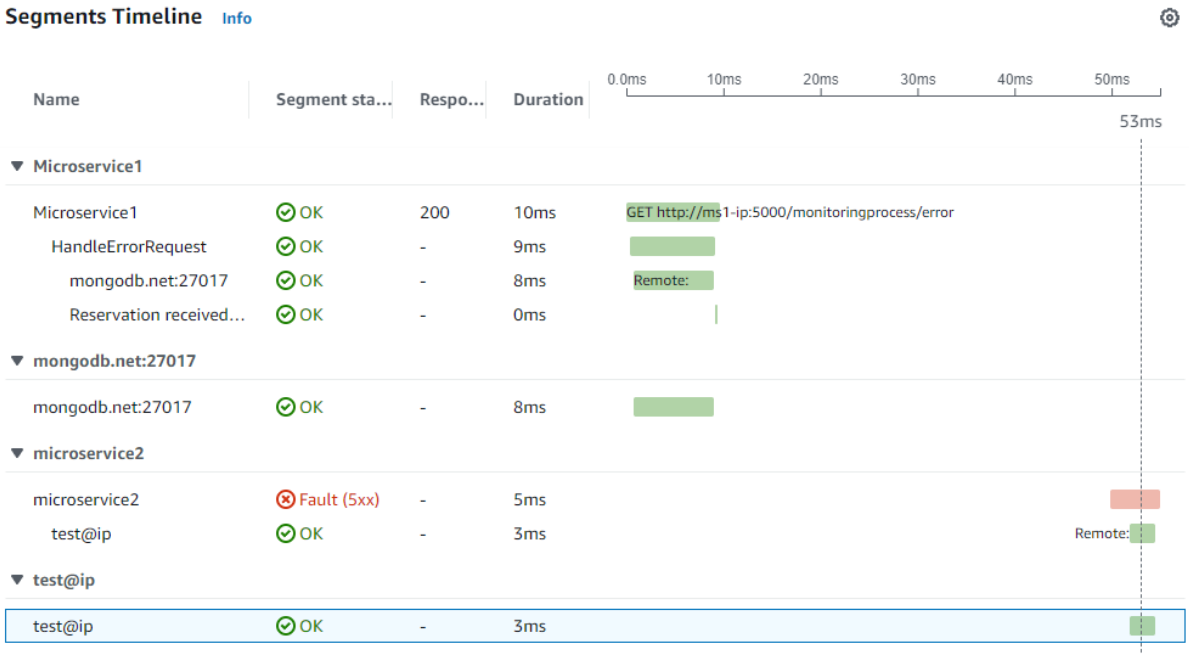
6.1.2 Erandiga protsessi kujutamine

X-Ray erandiga protsessi jälje ajajoone kuvatõmmis on joonisel 13 ning Tempo kuvatõmmis joonisel 14. Jäljed on genereeritud joonisel 7 esitatud erandit tekitava protsessiga.

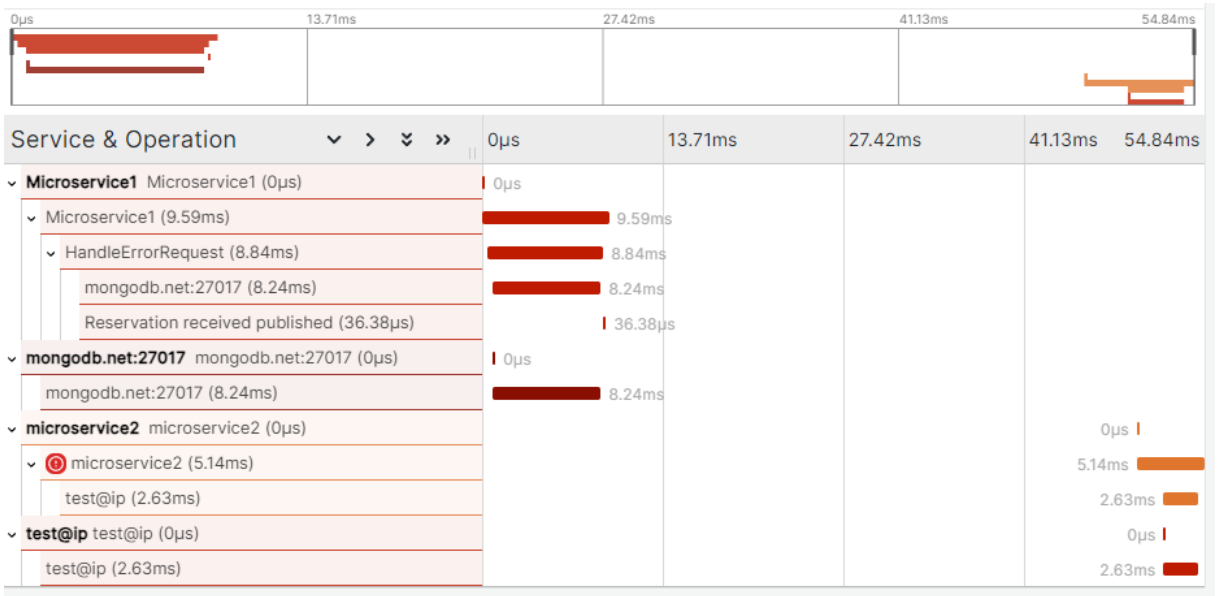
Piltidelt on näha, et erand eristub selgelt rohkem X-Rays. Kuigi Tempos on samuti märgitud väikselt punase hüüumärgiga, et teises mikroteenuses on toimunud erand, siis see ei tule

värvigamma tõttu selgelt välja. X-Rays näeb kohe peale vaadates, mis staatuskoodiga viga teenuses esines, Tempos seda välja pole toodud.

Jälge avades leiab erandi kohta infot X-Rays sektsiooni "*Exceptions*" alt. See sektsioon on igal jäljel alati olemas, kuid kui erandit pole toimunud, siis jäetakse see tühjaks. Tempos jälge avades pole väga selgelt välja toodud, et erand on juhutunud. Ainuke indikatsioon, et erand on toimunud, on sellest, et on tekkinud uus sektsioon "*Stack trace*", kuid seda on väga lihtne mitte märgata, sest see on ilmunud kõige alla ja pole eraldi rõhutatud.



Joonis 13. AWS X-Ray erandiga protsessi jäljed.



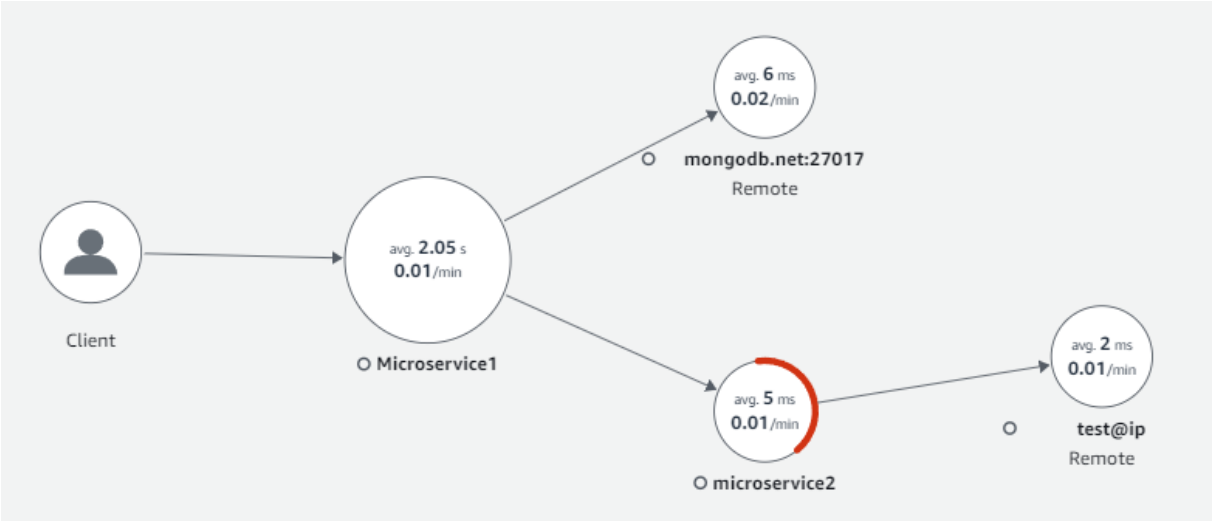
Joonis 14. Grafana Tempo erandiga protsessi jäljed.

6.1.3 Muud funktsionaalsused

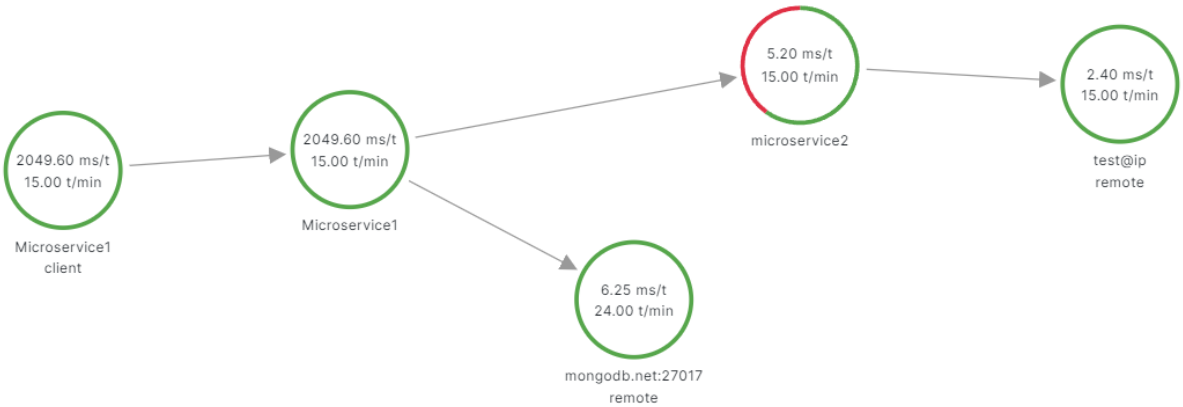
Nii X-Ray kui ka Tempo puhul on võimalik genereerida jälgedest statistikat. Nii X-Ray ku ka Grafana puhul näeb teenuste lõikes statistikat teenuse kaardil. Erinevus on selles, et X-Ray puhul saab valida, kas teenusekaardi sõlme suurus näitab päringute arvu, latentsust või tervist. Joonisel 15 on valitud sõlme suuruse esitamine latsentsuse järgi. Seal on selgelt näha, kuidas esimese mikroteenuse sõlm on kõige suurem, sest seal toimub kauakestev pudelikaela protsess.

Tempo teenusekaardil on ainult üks vaade, kuid seal on samuti näha keskmine reageerimiseaeg ning päringute arv minutis. Tempo teenuse kaart on toodud joonisel 16.

Mõlemal teenuse kaardil on selgelt välja toodud, millises teenuses on esinenud veaolukordi, kuid X-Ray teenuse kaardil on latentsusaeg visuaalselt selgemalt välja toodud.



Joonis 15. AWS X-Ray teenuse kaart, kus sõlme suurus näitab latentsust.



Joonis 16. Grafana teenuse kaart.

Grafana Tempos saab eraldi genereerida graafiku, kus näeb jälgede põhjal näiteks esinenud vigade arvu, keskmist vastamise aega ja kogu jälgede arvu. Grafana statistika pluss on see, et näeb ajas liikumist ning numbreid ning seda saab vastavalt soovile filtreerida teenuste, jälgede jne lõikes. X-Rays on võimalik avada graafikud iga teenuse lõikes, valikus on näiteks latentsusaeg, päringute arv ja vigade arv.

Nii Amazon X-Rayl kui ka Grafana Tempol on olemas põhjalik dokumentatsioon ning kasutusjuhendid, samuti on võimalik raskuste ja küsimuste korral kasutajatoega ühendust võtta. Mõlemad süsteemid on lihtsasti õpitavad ja kasutatavad.

6.1.4 Hinnakujundus

AWS X-Ray iga kuu esimesed 100 000 salvestatud jälge on tasuta ning esimesed 1 000 000 välja otsitud või skaneeritud jälge on tasuta. Edaspidi iga miljoni salvestatud jälje kohta tuleb maksta 5 dollarit ning iga miljoni välja otsitud või skaneeritud jälje kohta tuleb maksta 0,5 dollarit [28]. AWS kodulehel on ka kalkulaator, mis võimaldab hinnangulise igakuise summa välja arvutada.

Tempo tasuta versioon võimaldab koguda 50GB jälgesid, mida hoitakse 14 päeva. Pro versioon võimaldab koguda 100GB jälgesid ning neid hoitakse 30 päeva. Iga järgnev gigabait on 0,5 dollarit. Cloud Advanced versioon võimaldab koguda 100 GB jälgesid, mida hoitakse 13 kuud ning iga järgnev gigabait maksab 0,5 dollarit [29].

6.1.5 Tulemused

AWS X-Ray ning Grafana Tempo nõuetele vastavuse kokkuvõte on toodud tabelis 2. X-Rayl ja Tempol on palju sarnaseid omadusi, kuid on ka olulisi erisusi. X-Ray lahenduse üks suurim positiivne külg võrrledes Tempoga oli see, et erandite esinemine oli selgelt välja toodud ning info nende kohta oli lihtsamini üles leitav. Grafana Tempo peidab värvigamma tõttu olulisi detaile ära, nagu näiteks erandi esinemine, seda nii heleda kui ka tumeda taustaga variandis.

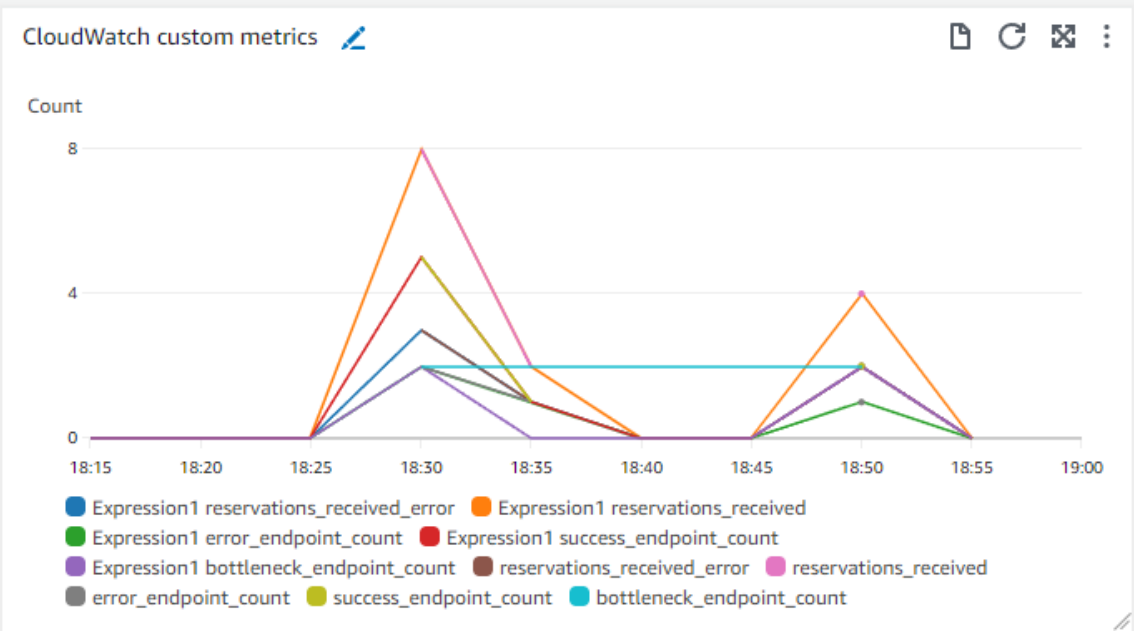
Nõue	AWS X-Ray	Grafana Tempo
Integreerumine OpenTelemetry'ga		
Aitab tuvastada pudelikaelu		
Veaolukordade selge esitamine		
Jäljed lihtsalt eristatavad ja arusaadavad		
Lihtsasti õpitavad ja kasutatavad		
Põhjalik dokumentatsioon ja tugi		

Tabel 2. AWS X-Ray ja Grafana Tempo vastavus nõuetele.

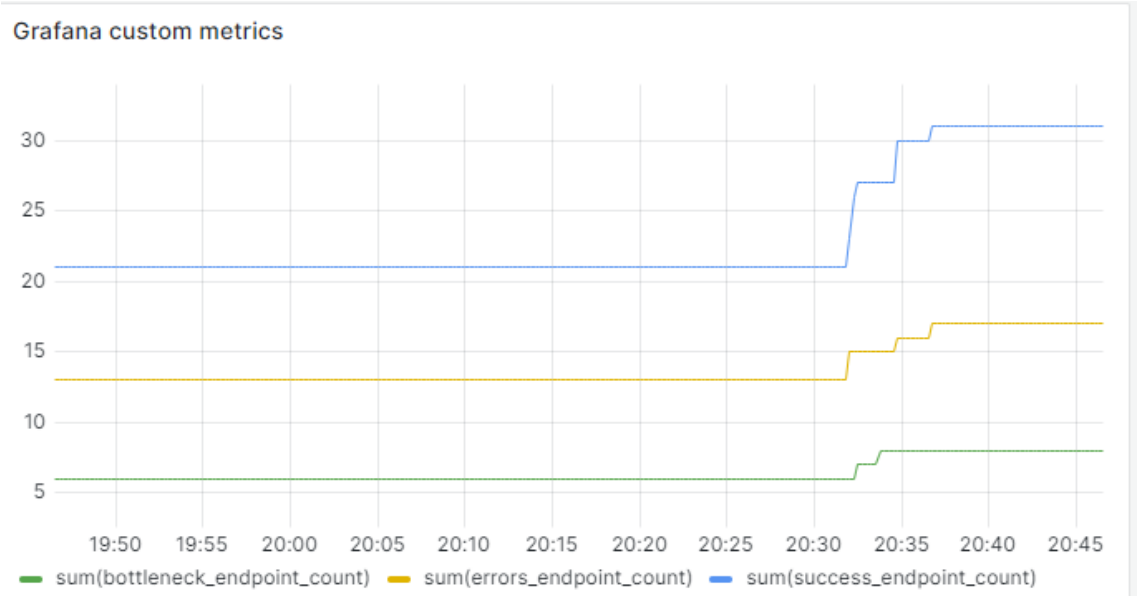
Kokkuvõttes täidaks mõlemad tööriistad autori arvates enamus nõudeid, mõlemal tootel oli oma plussid ja miinused, kuid X-Ray täitis rohkem Guest Engagement toote nõudeid hajutatud jälgitavusele.

6.2 AWS CloudWatch vs Grafana mõõdikud

Mõõdikute visualiseerimise tööriistade võrdlemiseks avaldatakse kohandatud mõõdikuid mõlmast mikroteenusest, mis on kirjeldatud peatükis 5.5. CloudWatchis visualiseeritud kohandatud mõõdikud on joonisel 17 ning Grafana mõõdikud joonisel 18. Järgnevalt analüüsitakse tööriistade vastavust seatud nõuetele.



Joonis 17. CloudWatch mõõdikud.



Joonis 18. Grafana mõõdikud.

6.2.1 Kohandatud mõõdikud

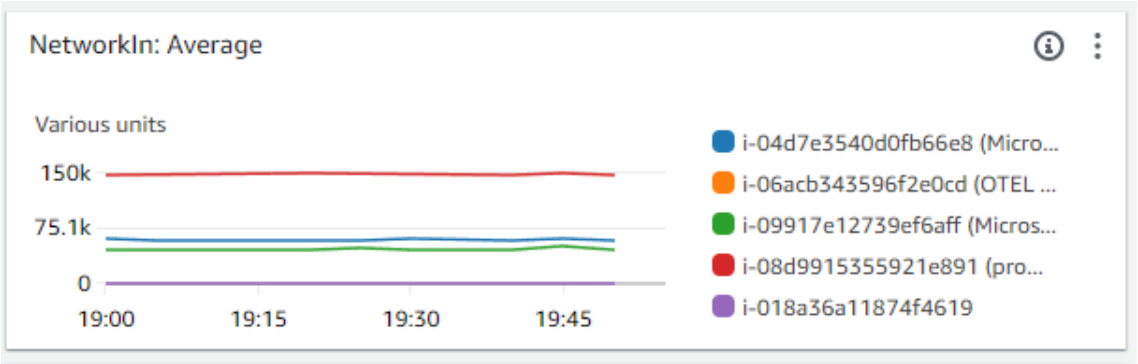
Joonistelt 17 ning 18 on näha, et nii CloudWatchil kui ka Grafanal on olemas kohandatud mõõdikute funktsionaalsus. Küll aga käituvad tööriistad samade mõõdikute puhul vaikimisi erinevalt. CloudWatchis avaldatud loendur mõõdikud esitatakse valitud perioodi summana - kui on valitud perioodiks näiteks üks minut, siis graafik näitab mitu mõõdikut ühe minuti jooksul avaldati. Grafana puhul esitatakse loendur kumuleeritud summana loenduri alustamise hetkest. Seetõttu näevad graafikud selgelt erinevad välja.

Grafanas on lihtsalt võimalik koostada ka sarnane graafik CloudWatchile, kuid lihtsa vaevaga sarnast graafikut Grafanale CloudWatchis võimalik luua ei olnud, seega tundub Grafana kasutamine intuitiivsem, sest kummagi süsteemi päringukeel ei ole autoril igapäevaselt kasutuses.

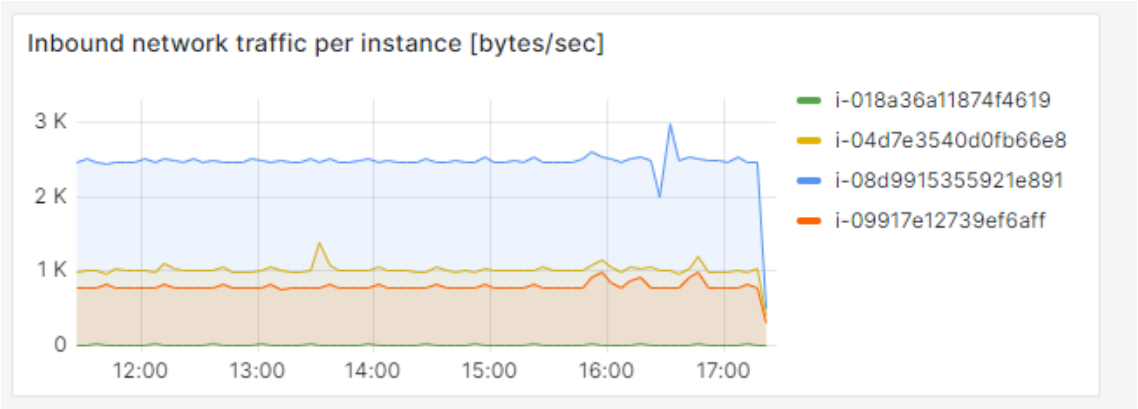
6.2.2 Süsteemi tasemel mõõdikud

Kuna EC2 serverite mõõdikute andmed imporditakse Grafanasse CloudWatchist, siis on süsteemi tasemel mõõdikute graafikud väga sarnased võrreldavates süsteemides. CloudWatchis on vaikimisi EC2 serveritele 12 graafikut ning Grafanas 13. Graafikud on peamiselt samad - protsessori kasutus, sissetulev ning väljaminev võrguliiklus, kettalt lugemine ning kirjutamine ja staatuse kontroll. Grafanas on veel lisaks graafik kogu võrguliikluse kohta (nii sissetulev kui ka väljaminev).

Näide süsteemi tasemel mõõdikutest CloudWatchis on joonisel 19 ning Grafanas joonisel 20. Grafana esitab süsteemi tasemel mõõdikute graafikuid struktureeritumalt ning samuti on Grafanas selgem pealkiri igal graafikul, mis mõõdikutega tegemist on.



Joonis 19. CloudWatch sissetulev võrguliiklus.



Joonis 20. Grafana sissetulev võrguliiklus.

6.2.3 Hoiatused ja teavitused

Nii CloudWatch kui ka Grafana võimaldasid seadistada hoiatused mõõdikute põhjal. Mõlemas süsteemis seati hoiatus kui joonisel 7 toodud veaolukord esines rohkem kui 5 korda ühe minuti jooskul. Mõlemad tööriistad seadistati saatma hoiatus emaili peale. CloudWatchi saadetud teavitus on toodud joonisel 21 ning Grafana hoiatus joonisel 22.

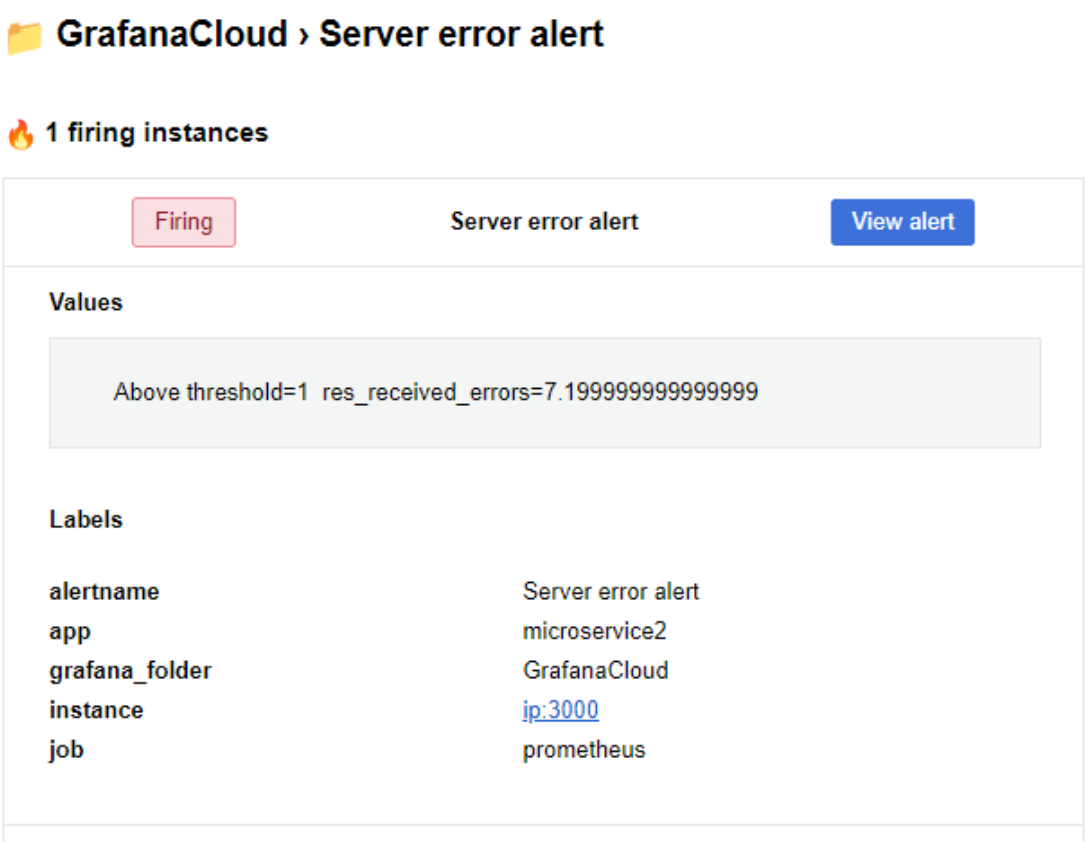
Kuigi mõlemas süsteemis on võimalik seadistada kohandatud mõõdikute pealt hoiatusi, siis näeb Grafana saadetud hoiatus selgelt parem välja. Samuti on informatsioon Grafana hoiatusel lihtsamini leitav puhtama ja selgema struktuuri tõttu. CloudWatchi hoiatus on väga üleujutatud ning pole selgelt aru saada, et tegemist on hoiatusega, seetõttu võib seda olla lihtne ignoreerida. Mõlemad hoiatused võimaldavad lingi kaudu minna otse hoiatuse põhjuseid uurima minna.

You are receiving this email because your Amazon CloudWatch Alarm "Microservice2_server_errors_alarm" in the EU (Stockholm) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [5.005420508188807 (27/12/23 14:49:00)] was greater than the threshold (5.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Wednesday 27 December, 2023 14:50:55 UTC".

View this alarm in the AWS Management Console:
https://eu-north-1.console.aws.amazon.com/cloudwatch/deeplink.js?region=eu-north-1#alarmsV2:alarm/Microservice2_server_errors_alarm

Alarm Details:
- Name: Microservice2_server_errors_alarm
- Description:
- State Change: INSUFFICIENT_DATA -> ALARM
- Reason for State Change: Threshold Crossed: 1 out of the last 1 datapoints [5.005420508188807 (27/12/23 14:49:00)] was greater than the threshold (5.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp: Wednesday 27 December, 2023 14:50:55 UTC

Joonis 21. CloudWatchi hoiatus.



Joonis 22. Grafana hoiatus.

6.2.4 Muud funktsionaalsused

CloudWatchis on võimalik esitada mõõdikuid 7 erineval viisil, samas Grafans on defineeritud 30 erinevat visualiseerimise võimalust, seega Grafana on palju mitmekülgsem ja kohandatavam.

Nii Grafanal kui ka CloudWatchil on olemas põhjalik dokumentatsioon ning kasutusjuhendid, samuti on võimalik raskuste ja küsimuste korral kasutajatoega ühendust võtta. Kasutatavuse poolest oli autori arvates Grafana pisut intuitiivsem ja lihtsamini õpitav kui CloudWatch, kuid mõlemaga oli õppimiskõver.

6.2.5 Tulemused

CloudWatchi ning Grafana nõuetele vastavuse kokkuvõte on toodud tabelis 3. Kokkuvõttes täitsid mõlemad mõõdikute visualiseerimise tööriistad kõik vajalikud nõuded, kuid Grafanas on selgelt rohkem võimalusi ning paindlikkust. Samuti ei olnud CloudWatchi saadetud hoiatused autori arvates kuigi head ning seetõttu on märgitud nõuetele vastavuse tabelis CloudWatchi hoiatus oranžilt.

Nõue	CloudWatch	Grafana
Süsteemi tasemel mõõdikud		
Kohandatud mõõdikud		
Hoiatused ja teavitused		
Lihtsasti õpitav ja kasutatav		
Põhjalik dokumentatsioon ja tugi		

Tabel 3. CloudWatch ja Grafana vastavus nõuetele.

Nii CloudWatchil kui ka Grafanal on omad plussid ja miinused, kuid autori arvates täidab paremini Guest Engagement toote mõõdikutele seatud monitooringu nõudeid Grafana.

7. Kokkuvõte

Antud lõputöö eesmärgiks oli koostada monitoorimise prototüüp Guest Engagement tootele, mille põhjal otsustada, millised tööriistad lahendavad kõige paremini Guest Engagement tootega seotud monitoorimise probleeme. Vaatluse alla võeti kaks erinevat hajutatud jälgitavuse tööriista ning kaks mõõdikute visualiseerimise tööriista, mis Guest Engagement monitoorimise lahendusest puudusid. Hajutatud jälgitavuse tööriistadest vaadeldi AWS X-rayd ning Grafana Tempot ning mõõdikute visualiseerimise tööriistadest Amazon CloudWatchi ja Grafanat.

Lõputöö eesmärgini jõudmiseks kirjeldati kõigepealt Guest Engagement toote olemasolevat monitooringu lahendust, toodi välja sellega seotud probleemid ning määrati töö skoop. Seejärel määrati funktsionaalsed ja mittefunktsionaalsed nõuded nii prototüübile kui ka analüüsitavatele monitooringu tööriistadele, samuti kirjeldati prototüübi tehnoloogiat. Lisaks analüüsiti prototüübil rakendatavaid monitooringu tehnoloogiaid.

Töö praktilises osas kirjeldati prototüübi koosatamise lahendust ning monitooringu tööriistade rakendamist prototüübil. Seejärel võrreldi omavahel monitooringu tööriistu ning selgitati välja nende vastavus nõuetele.

Hajutatud jälgitavuse tööriistade AWS X-Ray ning Grafana Tempo võrdlusest selgus, et kuigi tööriistad käitusid paljudes olukordades sarnaselt ning täidaks enamus nõudeid ära, siis paremaks osutus siiski X-Ray. X-Ray tugevused Tempo ees olid puhas ja järjekindel disain ning vigade selge väljatoomine.

Mõõdikute visualiseerimise tööriistadest Amazon CloudWatch ning Grafana jäi aga peale Grafana. Grafanal oli selgelt rohkem kohandamise võimalusi ning saatis ka arusaadavamaid ja parema disainiga hoiatusi.

Antud töö järgmisteks sammudeks on parema hajutatud jälgitavuse tööriista ning mõõdikute visualiseerimise tarkvara rakendamine ning kasutuselevõtt Guest Engagement toote raames.

Kasutatud kirjandus

[1] *Guest Engagement | SiteMinder*. [Accessed: 23-09-2023]. URL: https://www.siteminder.com/guest-engagement/.

[2] Cindy Sirdharan. *Distributed Systems Observability | O'Reilly Media, Inc.* [Accessed: 30-09-2023]. 2018. URL: https://learning.oreilly.com/library/view/-/9781492033431/ch04.html#e43ec9bf-77d0-4dc5-a27a-44d784e3db2b.

[3] *What is Observability and Why is it Important? | IoT Business News*. [Accessed: 24-09-2023]. 2022. URL: https://iotbusinessnews.com/2022/12/28/39577-what-is-observability-and-why-is-it-important/.

[4] *Microservices Observability | Devopedia*. [Accessed: 10-10-2023]. URL: https://devopedia.org/microservices-observability.

[5] *Three Pillars of Observability | DNSstuff*. [Accessed: 01-01-2024]. URL: https://www.dnsstuff.com/three-pillars-of-observability.

[6] *What is Amazon CloudWatch? | AWS*. [Accessed: 11-12-2023]. URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html.

[7] *OVERVIEW | Prometheus*. [Accessed: 30-09-2023]. URL: https://prometheus.io/docs/introduction/overview/.

[8] *OpenTelemetry | OpenTelemetry*. [Accessed: 10-11-2023]. URL: https://opentelemetry.io/.

[9] *What is .NET? | Microsoft*. [Accessed: 07-10-2023]. URL: https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet.

[10] *Introduction | NestJS*. [Accessed: 07-10-2023]. URL: https://docs.nestjs.com/.

[11] *Node.js Introduction | W3Schools*. [Accessed: 11-12-2023]. URL: https://www.w3schools.com/nodejs/nodejs_intro.asp.

[12] *What is NATS | NATS Docs*. [Accessed: 07-10-2023]. URL: https://docs.nats.io/nats-concepts/what-is-nats.

[13] *What is Amazon EC2? | AWS*. [Accessed: 02-12-2023]. URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html.

[14] *What is AWS X-Ray?* | *AWS*. [Accessed: 08-11-2023]. URL: https://docs.aws.amazon.com/xray/latest/devguide/aws-xray.html.

[15] *AWS X-Ray features* | *AWS*. [Accessed: 08-11-2023]. URL: https://aws.amazon.com/xray/features/.

[16] *Tempo documentation* | *Grafana Labs*. [Accessed: 08-11-2023]. URL: https://grafana.com/docs/tempo/latest/.

[17] *Get started with Grafana Tempo* | *Grafana Labs*. [Accessed: 08-11-2023]. URL: https://grafana.com/docs/tempo/latest/getting-started/.

[18] *About Grafana Tempo* | *Grafana Labs*. [Accessed: 08-11-2023]. URL: https://grafana.com/oss/tempo/.

[19] *How Amazon CloudWatch works* | *AWS*. [Accessed: 03-12-2023]. URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch_architecture.html.

[20] *Use Amazon CloudWatch metrics* | *AWS*. [Accessed: 03-12-2023]. URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/working_with_metrics.html.

[21] *System Architecture* | *PromLabs*. [Accessed: 16-12-2023]. URL: https://training.promlabs.com/training/introduction-to-prometheus/prometheus-an-overview/system-architecture.

[22] *Metrics* | *Grafana Labs*. [Accessed: 03-12-2023]. URL: https://grafana.com/docs/grafana-cloud/send-data/metrics/.

[23] *Using instrumentation libraries* | *OpenTelemetry*. [Accessed: 10-11-2023]. URL: https://opentelemetry.io/docs/instrumentation/net/libraries/.

[24] *Exporters* | *OpenTelemetry*. [Accessed: 10-11-2023]. URL: https://opentelemetry.io/docs/instrumentation/net/exporters/.

[25] *First steps* | *NestJS*. [Accessed: 04-01-2024]. URL: https://docs.nestjs.com/first-steps.

[26] *StatsD: What Is It and How To Monitor It* | *MetricFire*. [Accessed: 11-12-2023]. 2023. URL: https://www.metricfire.com/blog/statsd-what-is-it-and-how-to-monitor-it/.

[27] *collectd – The system statistics collection daemon* | *collectd*. [Accessed: 11-12-2023]. URL: https://collectd.org/.

[28] *AWS X-Ray pricing* | *AWS*. [Accessed: 08-11-2023]. URL: https://aws.amazon.com/xray/pricing/.

[29] *Pricing | Grafana Labs*. [Accessed: 08-11-2023]. URL: https://grafana.com/pricing/.

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

Mina, Maris Salk

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Mikroteenustel põhineva toote monitoorimise parandamise prototüüp", mille juhendaja on Meelis Antoi
   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.

04.01.2024

[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.