TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Silver Viljaste 174191IDAR

KONTEINERRAKENDUSTE AUTOMAATNE JUURUTAMINE AEGRIDA ANDMETE ANALÜÜSIL TELIA EESTI AS NÄITEL

Diplomitöö

Juhendaja
Edmund Laugasson
MSc

Tallinn 2021

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:	Silver Viljaste
Kuupäev:	17.05.2021

Annotatsioon

Käesolev diplomitöö on valminud IT-süsteemide administreerimise eriala lõputööna, mille eesmärgiks on automatiseerida konteinerrakenduste väljalaske protsess ettevõttes Telia Eesti AS. Automatiseerimise nõue tuli tarkvaraarendusmeeskondade soovist optimeerida juurutamise töövoog toodangus, eesmärgiga minimeerida ajakulu, mis tekib inimressursi kasutamisel tarkvara uuenduste valideerimisel ja tõrgete lahendamisel. Teise ajendina sooviti parandada infosüsteemide tõrkekindlust tarkvara muudatustega kaasnevates ohtudes.

Töös keskendutakse konteinerrakenduste automaatsele juurutamisele. Selleks kavandas autor tarkvara juurutamise töövoo, mis sisaldab tarkvarasüsteemi väljalaske või tagasi pööramisega seotud ülesandeid ja tegevusi. Automatiseeritud töövoo kasutuselevõtuks ja vajalike eelduste kaardistamiseks viis autor ettevõttes läbi küsitluse. Kogutud andmete põhjal analüüsiti ettevõtte tarkvaraarendusmeeskondade kompetentside taset automatiseerimisega seotud valdkondades. Tasemete analüüsil selgus, et organisatsioon vastab tarkvara tarneahela automatiseerimise tingimustele.

Tööprotsessi optimeerimiseks võrdles autor töös kolme juurutusvahendit, millest ühe abil visualiseeriti juurutamise töövoog ning kirjeldati mõõdustikud aegrida andmete analüüsil tehtavate otsustusprotsesside automatiseerimiseks. Platvormiteenuse valikul lähtus autor neljast põhikriteeriumist: integreerimine teiste ettevõtte teenustega, hübriidpilve tugi, isemajutatav ja vabavaraline. Sobiv platvormiteenus võeti tarkvaraarendusmeeskondade poolt kasutusele pideva juurutuse raamsüsteemina.

Töös dokumenteeritud tehniline lahendus on täielikult reprodutseeritav ning kasutatav teistes ettevõtetes, kus on kasutusel konteinertehnoloogiad ja hajusrakendused ning kus plaanitakse tarkvara tarneprotsesse automatiseerida.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 45 leheküljel, 7 peatükki, 21 joonist, 4 tabelit.

Abstract

Automatic deployment of containerized applications based on time series data analysis by the example of Telia Eesti AS

This diploma thesis has been completed as a dissertation in the field of IT systems administration. The aim of the thesis is to automate release process for containerized applications in Telia Eesti AS. The need for automation came from the interest of software development teams to improve continuous delivery pipeline for production deployments. The goal was to minimize the time spent using human resources to validate deployment cycles and resolve issues after software release. Another purpose was the need to improve the resilience of software systems to the risks associated with the changes in code.

The work focuses mainly on automatic deployment of containerized applications. Therefore, a canary release pipeline was designed with corresponding tasks describing software release or rollback activities. In order to achieve a fully automated deployments and meet the necessary criteria, a survey was conducted. The received data was used in organizational evaluation for the maturity analysis in the field of automation. The maturity assessment showed that software development teams meet the required level for automating different stages in software delivery chain.

In order to select a suitable tool for continuous delivery automation, three platform services were compared, one of which was used to visualize canary workflow. Also, some trivial metrics were defined to support automation of decision-making processes based on time series data analysis. The automation framework was selected based on four main criteria: support for hybrid cloud, integration capability with other DevOps tools, self-contained and the existence of a public license. Finally, a selected platform service was adopted by development teams as a framework for continuous automation.

The technical solution documented in the work is fully reproducible and can be implemented in other companies where container technologies and distributed applications are used in continuous delivery processes.

The thesis is in Estonian language and contains 45 pages of text, 7 chapters, 21 figures, 4 tables.

Lühendite loetelu

CD	Ingl.k. Continous Delivery - tarkvaraarendusprotsessi osa. Hulk tarkvara arendamise tavasid ja põhimõtteid, mille eesmärk on parandada tarkvara tarnimise protsessi
CI	Ingl.k. Continuous Integration - pidev tarkvarakoodi integreerimine tarkvaraarendusprotsessis. Pideval integreerimisel ühendatakse kõik eri tarkvaraarendajate poolt tehtud koodimuudatused mitmel korral päevas suuremas koodibaasis
DevOps	Ingl.k. Developement and Operations – arendus ja haldus
DevOps tools	Komplekt arendus- ja haldustööriistadest, mis toetavad arendusprotsessi efektiivistamist ja tarneahela automatiseerimist
Halyard	Käsurea tööriist Spinnakeri konfigureerimiseks, juurutamiseks ja paikamiseks
Kanaarianalüüs	Ing.k. Canary Analysis - protsess, mille käigus kogutakse ja võrreldakse kahe tarkvara versiooni mõõteandmeid
Kanaarijuurutus	Ingl.k. Canary Deployment - tarkvara väljalaske praktika, mis vähendab tarkvara uue versiooni tootmisse viimise riski. Uus versioon juurutatakse piiratud kasutajagrupile ning stabiilse versiooni kõrvale
Marketplace	Elektrooniline turg, kus müüjad pakuvad ja ostjad tellivad tarkvaratooteid
On-Premises	Tarkvara, mis on juurutatud ettevõtte IT-taristus
Prometheus	Serveritarkvara, mida kasutatakse IT-süsteemide monitoorimiseks
SaaS	Ingl.k. Software As A Service - tarkvara kui teenus
Spinnaker	Töövoo haldussüsteem, mis aitab automatiseerida tarkvara juurutamist
YAML-fail	Ingl.k. YAML Ain't Markup Language - YAML formaadis loodud fail, mis on inimeste loetav andmevorming ja mida kasutatakse andmete järjestamiseks

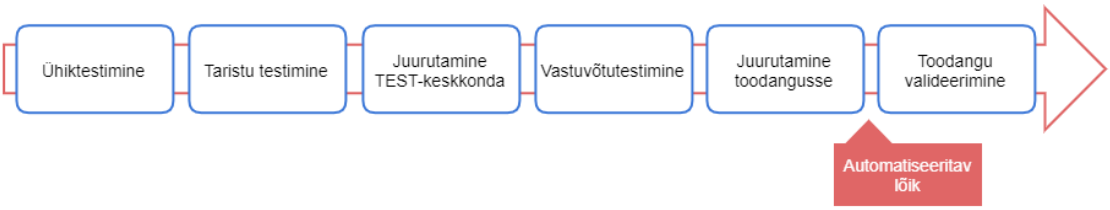
Sisukord

Jooniste loetelu

Tabelite loetelu

1. Sissejuhatus

Erinevad tehnoloogilised lahendused võimaldavad ettevõtetel optimeerida ja parendada tööprotsesse. Tööprotsesside pidev kujundamine on tingitud ettevõtete vajadusest täita kokku lepitud eesmärke: kulude, inim- ja ajaressursi kokkuhoid; konkurentsieelise saavutamine läbi uute tehnoloogiate kasutuselevõtu või näiteks tööprotsessi täpsuse ja kvaliteediomaduste tõstmine. Nende eesmärkide saavutamise üheks vahendiks on töölõikude või kogu tööprotsessi rekonstrueerimine läbi automatiseerimise. Käesolev diplomitöö keskendub Telia Eesti ASi tarkvara arendustsükli realisatsiooni etapis tarkvara valideerimise ja teostamise tööprotsessi optimeerimisele. Täpsemalt automatiseeritakse uue tarkvaraversiooni juurutamise ja toodangu valideerimise töövoog (Joonis 1). Autor lähtub töövoo kavandamisel ja tarkvarasüsteemi paigaldamisel pideva juurutamise (ingl.k. *continuous deployment*) põhimõtetest.



Joonis 1. Tarneahel

Tarkvara paigaldamise printsiibid on aja jooksul konteinertehnoloogia ja juurutusvahendite populariseerimisel palju muutunud. Traditsiooniliselt toimus tarkvarasüsteemi paigaldamine käsitööna, mis oli tihti ressursimahukas ja veaohtlik protsess. Automatiseeritud tarneahela puhul lähtutakse tööprotsesside ja -töövoogude pidevast optimeerimisest ning parendamisest. Pidev automatiseerimine on praktika, mis võimaldab tarkvara arendamist, testimist ning juurutamist teha kiiremini ja sagedamalt, eesmärgiga reageerida kiiresti muutuvatele turunõuetele, tehes seda odavamalt ning vähemate katkestustega. [1] Käesoleva töö praktilises osas näidatakse, mil moel on see erinevate tarkvaratükkide koostoimel saavutatav.

Tarneprotsessi automatiseerimine on DevOps metoodikas olulise tähtsusega valdkond. Täpsemalt seostatakse DevOps metoodikat peamiselt IT-süsteemide pideva integratsiooni ja -tarnega kui vahendiga optimeerimaks töö voogu rakenduse elutsükli vältel arendusest toodangusse. [1] Kui pideva integratsiooni ja -tarne mõisteid käsitletakse erinevate

praktikatena, arvatakse tarkvara paigaldamine ja kasutuselevõtt tavaliselt pideva tarne osaks. Lähiaastatel on kujunenud uus arvamus, mille kohaselt on pidevat juurutamist hakatud käsitlema eraldi praktikana. Seda mõtet jagavad mitmed tuntud tarkvarahiiud nagu Google LLC ja Netflix Inc., kelle koostööl alustati 2014. a. uue tarkvaraprojektiga. Projekti väljundina valmis tööriist, mis järgib tarkvara paigaldamise parimaid tavasid ning sisaldab funktsioone nende rakendamiseks. Tarkvaratoote lähtekood avalikustati aasta hiljem, eesmärgiga võimendada ja populariseerida seesuguste platvormiteenuste arengut. Tarkvaratoote nimeks sai Spinnaker, mis keskendub üksnes tarkvara juurutamisele.

Netflix on kirjeldanud pidevat juurutamist järgmiselt: "Pidev juurutamine on pideva tarne edasiarendus, mille eesmärk on tarkvarafunktsionaalsuse või -täienduse kandmine tarbekeskkonda automaatselt alles siis kui sellele on eelnenud automaattestide, näiteks koormustestide või kanaarianalüüsi (ingl.k. *canary analysis*) edukas läbimine. [2]

Endine Atlassian Corporation Plc töötaja Sten Pittet on kirjeldanud pidevat juurutamist sõnadega: „Pidev juurutamine on tarkvara väljaandmise protsess, mille valideerimiseks ja süsteemi toimepidevuse kontrollimiseks kasutatakse automaattestimise põhimõtteid, eesmärgiga võimaldada tarkvara koodi paigaldamist toodangusse igal ajahetkel.“ [3]

Antud töö koondab endas töövoo automatiseerimisvahendi Spinnaker paigaldamist, häälestamist ja liidestamist ettevõtte teenustega. Spinnakeri kasutuselevõtu ja tööprotsesside kujundamise eelduseks on tarkvaraarendusmeeskondade DevOps küpsustaseme hindamise ning valmisoleku analüüs. Seejärel konstrueeritakse ühe tarkvaraprojekti näitel kobarsüsteem, mis võimaldab tarkvara uue versiooni juurutamist isoleerituna töötava süsteemi kõrvale ning kandidaatversiooni edukuse kontrollimist võrdlusanalüüsi meetodil.

1.1 Taust ja probleem

Ettevõtte tarkvaraarendusmeeskonnad annavad täna tarkvara tarneprotsessi lõppfaasis hinnanguid, kas käiku lastav tarkvara versioon on stabiilne või mitte, tehes otsuseid versiooni tagasi pööramise vajalikkusest. Sõltuvalt meeskondade dünaamikast või isikkoosseisu pädevustest, teostab tarkvara versiooni tagasi pööramise kas tarkvaraarendaja, IT-arhitekt või IT-haldur. Sageli kannab seda rolli IT-haldur, mille põhjuseks on arendajate ligipääsuõiguste või tehnilise lahenduse puudumine, mis aitaks tarkvara juurutusprotsesse visualiseerida.

Iga minut, mis kulub tõrke lokaliseerimisele, selle lahendamisele ja reageerimisele, võib avaldada mõju teenuste kasutamisel lõpptarbijale. Selle tagajärjeks võib olla kliendi rahulolu langemine, ettevõttele saamata jäänud tulu või mainekahju teenustaseme lepingute rikkumisel. Telia Eesti ühe allüksuse näitel 2020 a. perioodil mõõdetud

intsidentide kogu kestvusajaks arvutati 1517 minutit. Intsidentide retroanalüüsis märgiti põhjuseks vead uues tarkvara versioonis. Arvutuskäik sisaldas perioodi tõrke tuvastamisest kuni lahenduse leidmiseni ja rakendamiseni. Automatiseeritud juurutusmeetodi kasutamine koos vigade tuvastamise lahendusega oleks andnud kindlasti teistsugust efekti. Tarkvaratoote lõpuni automatiseeritud tarneahel võimaldaks tarkvaraarendusmeeskondadel keskenduda ärieesmärkidele ning olla vähem hõivatud tõrkeotsingu ning probleemide lahendamisega.

Diplomitöö teema on ajendatud autori isiklikust huvist pidevjuurutuse automatiseerimise vastu ning suuremast haldusvastutuse jagamisest meeskondade poolt arendatavates süsteemides.

Käesolev töö on koostatud Tallinna Tehnikaülikooli IT-süsteemide administreerimise eriala lõputööna 2021. aasta kevadel Tallinnas. Töös kajastatud uurimisküsitlus viidi Telia Eesti AS tarkvaraarendusmeeskondade seas läbi vahemikus veebruar-märts 2021.

1.2 Ülesande püstitus

Diplomitööle on seatud 3 põhieesmärki:

1. Analüüsida ja määratleda Telia Eesti ASi arendusmeeskondade kompetentside tase ning valmisolek tarkvara väljalaske automatiseerimiseks;
2. Paigaldada ja häälestada CD platvormiteenus Spinnaker;
3. Kavandada töövoog konteinerrakenduste automaatseks juurutamiseks koos tarkvara versiooni valideerimisega.

Põhieesmärkide täpsustamiseks on seatud 5 lähtetingimust:

1. Prototüübi lõpplahendus peab lihtsustama ja tõhustama töö- ja arendusprotsessi;
2. Prototüüp peab minimeerima konteinertarkvara juurutamisel tõrgete riski;
3. Tarkvara juurutusvahend peab olema asutusesiseselt (ingl.k. *on-premises*) juurutatav, vabavaraline ja omama hübriidpilve tuge;
4. Juurutusvahend ja selle komponendid peavad ühilduma ettevõtte põhiliste platvormiteenustega: Jenkins, Prometheus, Kubernetes ja Bitbucket;
5. Tarkvarauuenduste paigaldamiseks või tagasi pööramiseks kogutud ja töödeldavad andmed ning teabevahetuse sisu peavad olema struktureeritud.

1.3 Metoodika

Diplomitöös kasutab autor uurimisobjekti ja lähtetingimustega määratud nõuete vastavuse kontrollimiseks erinevaid metoodikaid: küsitlus ja eksperiment.

Uuringu käigus viib autor ettevõttes läbi DevOpsi küpsustaseme hindamise, millele tuginedes ja näidislahenduse analüüsile toetudes saab organisatsioon teha samme DevOpsi praktikate paremaks rakendamiseks. Uuringu eesmärk on välja selgitada tarkvaraarendusmeeskondade peamised takistused töös seatud põhieesmärkide saavutamiseks.

Eksperimendi käigus viib autor ettevõttes läbi katse, mille eesmärk on kontrollida, kas autori poolt loodav prototüüp parandab tarkvara tarne kvaliteeti ja tõrkekindlust.

1.4 Ülevaade tööst

Käesolev töö on jagatud seitsmeks peatükiks, mille järjestamisel on lähtutud akadeemilise töö loogilisest ülesehitusest.

Töö teoreetilises osas antakse ülevaade juurutusmeetoditest ja kolmest tarkvara juurutusvahendist, milles võrreldakse nende funktsionaalseid ja mittefunktsionaalseid omadusi.

Töö analüüsi osas annab autor hinnanguid ettevõtte tarkvaraarendusmeeskondade DevOps küpsustaseme ja eksperimendi tulemuste kohta.

Töö praktilise osa moodustab platvormiteenuse Spinnaker paigaldamine ja häälestamine, juurutamise töövoo kavandamine ning tarkvara kobarsüsteemi projekteerimine.

Viimases peatükis esitab autor kokkuvõtte töö tulemuste ja võimalike edasiarenduste kohta.

2. Juurutusmeetodid

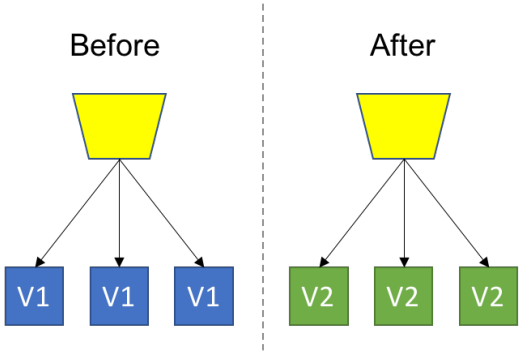
Tarkvara väljalaske variante on mitmeid, kuid käesolevas diplomitöös antakse ülevaade viiest levinumast meetodist. Kaasaegsed juurutusmeetodid pretendeerivad kõik ühele ja samale eesmärgile, milleks on tarkvara uuendamine reaalajas ilma süsteemi seiskamata. Süsteemi seisakuid võivad mõjutavad mitmed tegurid, mistõttu tuleb võimalike ohtudega arvestada juba tarkvara planeerimise faasis, mida toetab läbi mõeldud tarkvarasüsteemi paigaldusplaan. Nüüdisaegsed meetodid põhinevad hajussüsteemidel ja konteinertehnoloogiatel, kuid mille põhimõtted on üle kantavad ka traditsioonilistele paigaldusmudelitele.

Väljalaske ajal peab tarkvara olema tarbijatele alati kättesaadav ning juurutamise mõju tarkvara käideldavusele minimeeritud. Käideldavuse tagamiseks tuleks valida sobiv, tarkvara arhitektuuri, meeskonna pädevusi ja organisatsiooni tehnoloogilist valmisolekut arvestav juurutusstrateegia. Igal strateegial on omad eelised ja puudused, mille valimisel tuleks lähtuda organisatsiooni poolt seatud teenuste ärilisest kriitilisustasemest.

2.1 Asendusjuurutus

Ingl.k. *recreate strategy* või *basic deployment* on juurutusmeetod, mille puhul tarkvara vana versioon eemaldatakse kohe peale uue versiooni juurutamist. Toodangusse paigaldamisel tehakse versiooni asendus ehk alati on kasutusel üks versioon tarkvarasüsteemist. Selle meetodi juures toimub tarkvara muudatuste valideerimine tarneprotsessi testimise etapis. Versiooni väljalaskel süsteemi toimimist ei kontrollita ega muud tesimist läbi ei viida. [4] Protsessi illustreerib joonis 2.

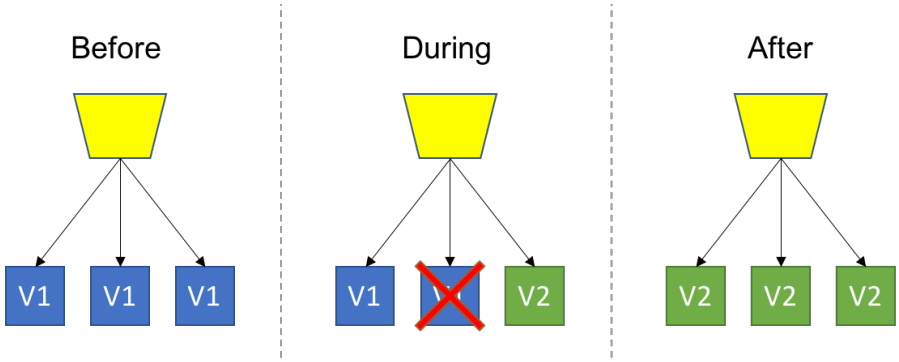
Asendusjuurutuse puhul on teenuse katkemine vältimatu. Katkestus võib olla lühiajaline, kuid on sõltuvuses süsteemi laadimise kiirusest. Juhul kui tarkvarasüsteem tegeleb käivitamisel näiteks andmebaasist andmete puhverdamise või valideerimisega, võib teenuse mittekäideldavus kujuneda minutite pikkuseks. Samuti muutub aeganõudvaks süsteemi versiooni tagasi pööramine, kuna puudub meede tarbijate HTTP-liikluse dünaamiliseks juhtimiseks. Meetod ei sobi kasutamiseks selliste süsteemide juures, mis klassifitseeruvad äri- või missioonikriitilisteks. Ettevõttesiseste rakenduste uuendamiseks, mille käideldavuse tase töövälisel ajal ei ole oluline, sobib meetod hästi kuna protsess on tehniliselt lihtsasti teostatav ja automatiseeritav.



Joonis 2. Asendusjuurutus [5]

2.2 Astmeline juurutus

Joonis 3 kirjeldab astmelist juurutusmeetodit ehk ingl.k. *gradual deployment* või *rolling deployment*, mis näeb ette uue versiooniga eksemplari paigaldamist kobarasse. Juurutamise õnnestumisel eemaldatakse kobarast koheselt varasem liige, mis evis eelmist versiooni. Seejärel korratakse tsüklit kuniks kõik kobara liikmed on uuendatud. Iga uue eksemplari juurutamisel tuleb veenduda süsteemi toimimises, näiteks REST-teenuste kaudu. Juhul kui uuel eksemplaril esineb paigaldamisel tõrkeid, katkestatakse protsess ning kobara esialgsete liikmete arv koos varasema versiooniga taastatakse. [6]



Joonis 3. Astmeline juurutus [5]

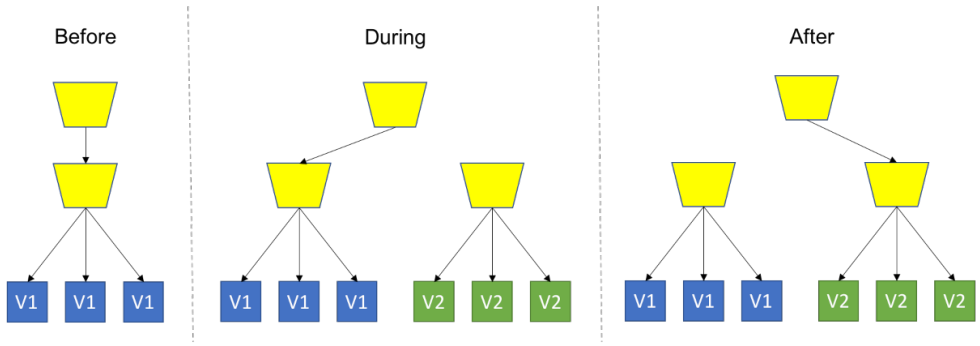
Astmelise juurutusmeetodi puhul töötavad rakenduse uus ja vana versioon paralleelselt. Seepärast peab tarkvara olema stabiilsuse ja toimimise seisukohalt vanema versiooniga alati tagasiühilduv. [6]

Kubernetese pilveplatvorm toetab konteinerrakenduste astmelist juurutamist, mis on sobiv ettevõttesiseste rakenduste ja kliendile suunatud vähemkriitiliste teenuste uuendamiseks. Antud juurutusmeetodi edasiarendusi on mitmeid, millest üheks on kanaarijuurutus.

2.3 Siniroheline juurutus

Ingl.k. *blue/green deployment* järgi töötavad kaks võimalikult ühesugust tootmiskeskkonda kõrvuti, kuid teineteisest sõltumatuna ja eraldiseisvalt. Isoleeritus võimaldab erineva versiooninumbriga tarkvarasüsteemi juurutada ja majutada mistahes viisil või pilvetaristul. Süsteeme seob omavahel koormusjaotur, milles juhitakse tarbijate liiklus uue ja vana versiooni vahel. [7]

Protsessi järgi eelneb uue versiooni väljalaskel kandidaatversiooni põhjalik testimine. Kui testimine on edukas, suunatakse koormusjaoturis kogu sissetulev liiklus vastu uut versiooni. Vana versioon aga jääb ootereˇziimile juhuks kui probleemide ilmnemisel on vaja tarkvara versioon tagasi pöörata. Järgmises juurutustsüklis kordub kõik täpselt samadel põhimõtetel. [7] Meetodist annab ülevaate joonis:

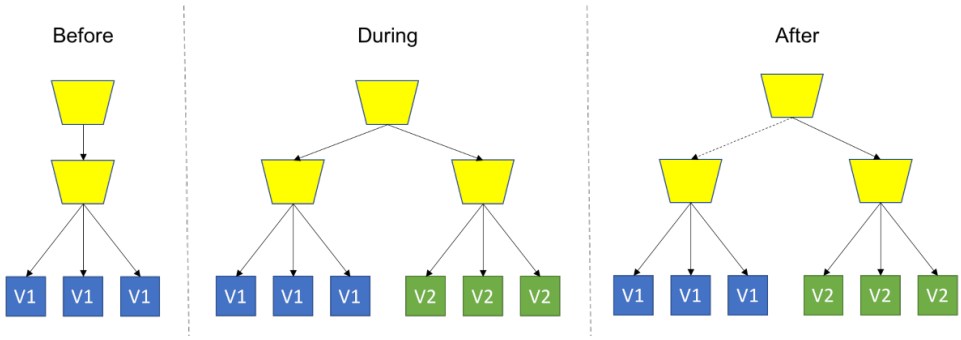


Joonis 4. Siniroheline juurutus [5]

Meetod sobib kasutamiseks nii äri- kui missioonikriitiliste süsteemide juurutamiseks, võimaldades versiooni väljalaskel seda põhjalikult toodangulaadses keskkonnas testida. Koormusjaoturi abil saab tarkvara testimist viia läbi näiteks töötajate seas. Meetod ei sobi kasutamiseks tiheda arendustsükliga projektides, kus tarkvara tarnitakse toodangusse mitu korda päevas.

2.4 Kanaarijuurutus

Ingl.k. *canary deployment* käigus kantakse tarkvara muudatus toodangusse koheselt, kuid tehakse kättesaadavaks vaid osaliselt. Selleks juurutatakse tarkvara viimane ehk uusim versioon põhirakenduse kõrvale. Seda eksemplari tarkvarasüsteemist nimetatakse kanaarirakenduseks. Kanaariversiooniga paralleelselt paigaldatakse etalonversioon, mis evib põhirakendusega sama versiooni. Seejärel analüüsitakse kanaari- ja etalon versioonide erinevusi, mille tulemusel tehakse otsus põhirakenduse uuendamiseks või versiooni tagasi pööramiseks. [5] Kanaarijuurutust kirjeldab joonis 5 ning täpsem ülevaade on antud peatükis 5.



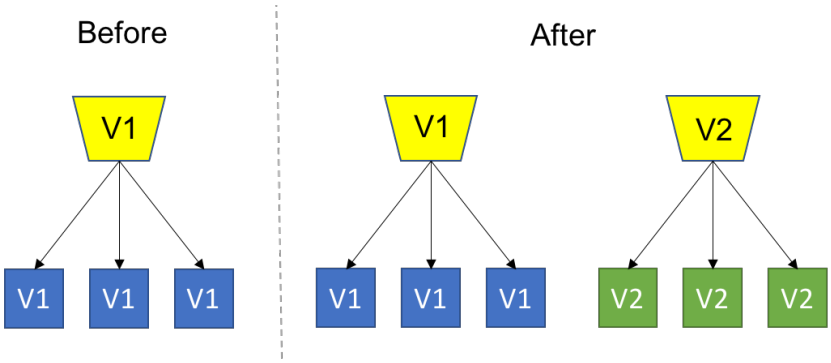
Joonis 5. Kanaarijuurutus [5]

Kanaarijuurutus on sobiv meetod sellistes tarkvaraprojektides, kus ühes päevas tarnitakse kümneid muudatusi. Juhul kui automaattestimisel on vaja koguda andmeid kauem, saab analüüsi kestust pikendada, mis on vajalik näiteks missioonikriitiliste teenuste funktsionaalsuse valideerimiseks enne lõpliku kasutuselevõttu.

2.5 Versioneeritud juurutus

Antud meetodi juures kasutatakse ka nimetust A/B testimine (ingl.k. *A/B testing* või *versioned deployment*, mille puhul tarkvara funktsionaalsus või -täiendus tarnitakse lõppkasutajatele kahel võimalikul viisil: ühe võimalusena saab kasutaja ise uue funktsionaalsuse kasutusele võtta ehk aktiveerida ning teisel juhul rakendab tootja funktsionaalsuse sobiva kasutajate valimi põhjal. Valimi määramisel on abiks näiteks kasutajate geograafiline asukoht, ekraani suurus, veebilehitseja tüüp jmt. [8].

Versioneeritud meetodi puhul juurutatakse põhirakenduse kõrvale eraldiseisev tarkvara eksemplar, kuhu täiendus tarnitakse (Joonis 6).



Joonis 6. Versioneeritud juurutus [5]

Seejärel juhitakse koormusjaoturis HTTP-liiklus eelpool kirjeldatud omaduste abil vastu soovitud versiooni. Protsessi tulemusel moodustub tarkvarasüsteem mitmest paralleelsest töötavast versioonist. [8]

Meetod võimaldab kaardistada tarkvaratoote parimad omadused, eesmärgiga rahuldada vahetul kasutajate tagasisidel turu vajadusi ning nõudmisi. Kui muudatus ei too katsel soovitud tulemust, eemaldatakse versioon tarkvarasüsteemist. Positiivse tagasiside korral tarnitakse muudatus kogu kliendibaasile. Meetod sobib kasutamiseks missioonikriitiliste teenuste arendamisel.

3. Tarkvara juurutusvahendite võrdlus

Telia Eesti AS lähtub tehnoloogiliste vahendite valikul arhitektuurinõukogu otsusest. IT-teenuste ja -süsteemide valik toimub üksmeele põhimõttel, millele eelneb iga komponendi valiku põhjendus ja analüüs. Analüüsi käigus võrreldakse omavahel turul pakutavaid alternatiive, hinnatakse tarkvara funktsionaalsust ja sobivust olemasolevate IT-teenustega. Lisaks arvestatakse turvaosakonna soovitusi ja tähelepanekuid. Seal, kus võimalik, eelistatakse vabavaralisi ja isemajutatavat tarkvara. Uute tehnoloogiate puhul kasutatakse nõuete kontrollimiseks ja parema ülevaate saamiseks või tehnilise võimekuse ja funktsionaalsuse hindamiseks prototüüpimist. Juhul kui tarkvara osutub valituks, avaneb ülejäänud tehnoloogiaüksusel võimalus selle kasutuselevõtuks.

Ettevõte peab lisaks teistele tehnoloogilistele vahenditele eraldi arvestust arendusprotsesse toetavate DevOps tööriistade üle (ingl.k. *DevOps tools*). Sinna kuuluvad erinevad platvormiteenused nagu Jira, Bitbucket, Artifactory, Bamboo jt. Lisaks kesksetele teenustele on tarkvaraarendusmeeskonnad juurutanud ja haldavad ise lahendusi, mis arvestaksid nende arengu ja vajadustega. Üheks selliseks vajaduseks on töövahendi olemasolu, mille abil saaks juhtida konteinerrakenduste ehitamist ja juurutamist.

Ettevõtte tarkvaraarendusmeeskonnad, mille koosseisu autor kuulub, on avaldanud soovi lahenduse järele, mis võimaldaks visualiseerida ja juhtida tarkvara väljalaskeid Kubernetese platvormile. Teise ajendina sooviti parandada infosüsteemide tõrkekindlust tarkvara muudatustega kaasnevates ohtudes ehk rakendada juurutusmeetodit, mis automatiseeriks tarkvara uuendamise või tagasi pööramise otsustusprotsessid. Kuna ettevõtte lähtub tarkvara arendamisel välearenduse (ing.k *agile development*) põhimõtetest, kantakse muudatusi toodangusse mitu korda päevas. Sellist mustrit toetab kanaarijuurutuse meetod.

Täna on tarkvara paigaldamisega seotud haldusrutiinid peamiselt IT-haldurite ülesanne. Vastutuse jagamise eesmärgil, mida toetaks platvormiteenuse funktsionaalsus, uuris ja võrdles autor kolme juurutusvahendit: Bamboo, Jenkins X ja Spinnaker. Autor tegi otsuse eelnimetatud tööriistade võrdlemiseks kolmel põhjusel:

1. Bamboo on ettevõttes keskselt juurutatud platvormiteenus, mis on arhitektuurinõukogu poolt heaks kiidetud. Uue ja alternatiivse lahenduse valikul tuleb arvestada varasemate otsustuskriteeriumidega ning funktsionaalsusega, mis on hetkel kehtiv ja vajalik ehk kas alternatiivse lahenduse kaalumiseks on alust.

2. Jenkins & Jenkins X – Jenkins on tarkvaraarendusmeeskondade seas populaarne alternatiiv Bamboole. Jenkinsi populaarsus kajastub ka erinevate võrgu- ja tehnikaväljaannete CI/CD lahenduste võrdlustes. Jenkins X keskendub tarkvara pidevale tarnele ja juurutamisele vaid Kubernetese platvormile, mis vastab hästi meeskondade poolt kirjeldatud ootustele. Jenkins X kompenseerib funktsionaalsust, mis Jenkinsil on täna puudulik, moodustades Jenkinsiga CI/CD terviklahenduse.
3. Spinnaker – pidevjuurutuse automatiseerimisvahend, mis lõimub nii majasiseste kui pilvepõhiste platvormiteenustega. Teiste alternatiividega võrreldes on Spinnaker vabavaraline ja lihtsa, ent funktsionaalse haldusliidesega platvormiteenus, mille arengus osalevad vaba tarkvara kommuun ja äriettevõtted.

3.1 Bamboo

Austraalias registreeritud ettevõtte Atlassian Corporation Plc on tuntud erinevate tarkvaraarendustööriistade arendamise ja turundamise poolest. Ettevõtte toodete nimekirja kuuluvad näiteks sellised tehnoloogiad nagu Jira projektihalduse tarkvara, Bitbucket koodivaramu, Confluence teabehaldussüsteem ja Bamboo tarkvara integreerimise- ja tarneahela juhtimissüsteem. [9] Kõiki neid tooteid saab juurutada ettevõtesiseselt, ent Atlassian pakub võimalust teenuseid tarbida ka SaaS (ingl.k. *software as a service*) pilvelahendusena. Atlassian tuli CI/CD tootega turule 2007. aastal, mil tutvustati selle esimest versiooni. [10] Sellest hetkest alates sai sisuliselt alguse töövoo haldussüsteemide võidujooks, mille tooteedetabelite võrdluses on Bamboole põhiliselt konkurentsi pakkunud vaba tarkvara etalon Jenkins.

3.2 Jenkins & Jenkins X

Jenkins (varasemalt Hudson) on vabavaraline avatud lähtekoodiga serveritarkvara, mis võimaldab automatiseerida tarkvara tarneahela erinevaid etappe - tarkvara integreerimine, ehitamine, testimine ja paigaldamine. Jenkinsi projekt sai alguse 2004. aastal, mil ettevõtte Sun Microsystemsi endine töötaja Kohsuke Kawaguchi otsis mooduseid oma tarkvarakoodi valideerimiseks, eesmärgiga parandada tarkvarakoodi kvaliteeti. Mainitud kitsaskohtade parendamise tulemusel sündis CI tööriist Jenkins. 2005. aastal tegi Kohsuke Jenkinsi toote lähtekoodi avalikuks, mis andis tõuke kogukondliku tarkvaraarenduse projekti loomisele. [11] Sellest hetkest alates on paljud era- ja avaliku sektori organisatsioonid suurel või vähesel määral pidevalt panustanud Jenkinsi tootearengusse. 2014. aastal liitus Kohsuke Kawaguchi ettevõttega CloudBees Inc., mis on täna üks suurimaid Jenkinsi platvormiteenuse ja selle pistikprogrammide edasiarendajaid ja populariseerijaid. Sealhulgas on CloudBees Inc. välja tulnud oma kommertsliku pilvepõhise CI/CD platvormiteenusega, mis on loodud Jenkinsi vabavara põhjal.

3.3 Spinnaker

Spinnaker on avatud lähtekoodiga platvormiteenus, mis on mõeldud tarkvara tarneprotsesside juhtimiseks ja haldamiseks. Spinnaker valmis tarkvaraettevõtete Netflix Inc. ja Google LLC ühisprojektina, mille esimene versioon avaldati 2017. aastal. Tarkvarasüsteemi lähtekood tehti Netflixi poolt avalikuks juba aastal 2015. [12] Kuigi CI/CD tarkvaraturul on Spinnakeri näol tegemist uue tootega, on see hõivanud lühikese ajaga arvestatava turuosa. Spinnakeri on organisatsioonis juurutanud näiteks sellised kaubamärgid: Adobe, SAP, Airbnb jt. Spinnakeri edu põhjuseid ei pea kaugelt otsima - kommertsettevõtete ja vaba tarkvara kogukonna koostööl positsioneerivad kasusaajad mõlemal poolel. Vaba tarkvara projektis osalemine aitab tõsta ettevõtete tootlikust vabatahtliku arendusressursi abil ja võimaldab ligipääsu laiemale intellektuaalsele varamule. Vabavara entusiastid saavad vastu kiirelt ja stabiilselt areneva toote, mille levitamist ja kasutamist ei piirata.

Hoolimata sellest, et Spinnaker keskendub peamiselt lõimimisele pilvepõhiste platvormiteenustega (AWS, Google Cloud, Microsoft Azure, Oracle Cloud jpt), on tootja tarkvara kavandamisel ja nõuete kirjeldamisel lähtunud tarbijate soovist isemajutatavate platvormiteenuste lõimimise vajaduse järele.

3.4 Juurutusvahendite funktsioonide võrdlus

Töövoo automatiseerimisvahendite funktsioonide võrdlemisel on aluseks võetud tootja tehniline dokumentatsioon. Hindamine on subjektiivne ning lähtub kolmeastmelisest numbrisüsteemist. Tarkvaratoote omadusele on antud hindeks 0 kui tootel puudub kirjeldatud funktsioon. Hinne 0,5 antakse juhul, kui tarkvarasüsteem vastab funktsiooni kirjeldusele osaliselt või kui funktsioon on kasutatav alternatiivsel meetodil, näiteks lisatarkvara kasutuselevõtul. Hinne 1 antakse omadustele, mis on tootja poolt süsteemi tarnitud.

Funktsioonide paremaks jälgimiseks on karakteristikud eraldi välja kirjutatud. Alapeatüki lõppu on lisatud toote omadusi ja hindeid koondav tabel (tabel 1).

1. **Hübriidpilve tugi**
   Hübriidpilv on tervik kahest või enamast pilvetaristust (privaatpilv, kogukonnapilv ja avalik pilv), mis toimivad iseseisvalt, kuid on omavahel tihedalt seotud [13]. Telia Eesti AS kasutab oma teenuste majutamisel kombineeritud (privaat ja avaliku) pilvetaristut.
   - **Bamboo** - ei lõimu Kubernetese privaatse ega avaliku pilvetaristuga. Toetab pistikprogrammi lisamisel AWS pilvetaristut.

- **Jenkins X** - lõimub vaid pilvepõhise (Google Kubernetes Engine) kui ettevõttesisese Kubernetese platvormiteenusega.
- **Spinnaker** - toetab erinevaid pilvetaristuid: AWS, GCP, Open Stack, App Engine, Kubernetes (sh isemajutatav), Oracle, Docker v2 Registry.

2. **Veebihaakide tugi**

   Ingl.k. *webhooks* on staatiline HTTP-tagasikutse, mis põhjustab protseduuri automaatse käivitumise süsteemis, kus veebihaak on kirjeldatud. [14] Kasutame ettevõttes veebihaake autonoomsetes süsteemides protsesside käivitamiseks.
   - **Bamboo** - veebihaakide kirjeldamine on lubatud süsteemis vaid halduri õigustes.
   - **Jenkins X** - veebihaakide kirjeldamine on lubatud projekti omaniku õigustes.
   - **Spinnaker** - veebihaakide kirjeldamine on lubatud projekti omaniku õigustes.
3. **Rakendusliidese tugi**

   API on üht tarkvara teisega ühendav liides, mis võimaldab kahel masinal, arvutil, seadmel, rakendusel andmeid vahetada või käske edastada. [15] Kasutame ettevõttes rakendusliideseid peamiselt tarkvarasüsteemide funktsionaalsuse laiendamiseks seal, kus see on piiratud.
   - **Bamboo** - REST-liidese tugi on olemas.
   - **Jenkins X** - REST-liidese tugi on olemas.
   - **Spinnaker** - REST-liidese tugi on olemas.
4. **Ainulogimise tugi**

   Ühekordne sisselogimine (ingl.k. *Signle Sign-On*) on pääsu reguleerimise funktsioon, mis võimaldab kasutajal pöörduda üheainsa logimisega mitme eri ressurssi poole. [16]
   - **Bamboo** - toetab LDAP-logimist. Tasulise pistikmooduli abil võimaldab kasutada erinevaid logimise tüüpe
   - **Jenkins X** - toetab pistikprogrammi lisamisel erinevaid logimise tüüpe: SAML, OAuth 2.0, LDAP.
   - **Spinnaker** - toetab funktsiooni aktiveerimisel erinevaid logimise tüüpe: OAuth 2.0/OIDC, SAML, LDAP, X.509.
5. **Rollipõhine pääsukontroll**
   - **Bamboo** - kasutab tarkvarasisest pääsureguleerimise loogikat.
   - **Jenkins X** - kasutab tarkvarasisest pääsureguleerimise loogikat.
   - **Spinnaker** - kasutab tarkvarasisest pääsureguleerimise loogikat.
6. **Pistikprogrammide tugi**
   - **Bamboo** - toetab pistikprogrammide juurdearendust.
   - **Jenkins X** - toetab pistikprogrammide juurdearendust.
   - **Spinnaker** - toetab pistikprogrammide juurdearendust.
7. **Töövoode kirjeldamine koodina**

Sarnaste karakteristikute järgi koodina kirjeldatud töövood (ing.k. *pipelines*) lihtsustavad ettevõttes arendus- ja halduskeerukust.

- **Bamboo** - toetab alates versioonist 6.0 töövoode kirjeldamist YAML-vormingus või kasutades Java programmeerimiskeelt.
- **Jenkins X** - toetab töövoode kirjeldamist YAML-vormingus.
- **Spinnaker** - toetab töövoode kirjeldamist JSON-vormingus.

8. **Haldusliidese olemasolu**
   - **Bamboo** - on olemas.
   - **Jenkins X** - on olemas eraldi pistikprogrammina.
   - **Spinnaker** - on olemas.
9. **Tarkvaratoote juurutamine konteinertehnoloogial**
   Lähtume ettevõttes tarkvara planeerimisel ja arendamisel mikroteenuste arhitektuurimudelist. Lisaks kasutame konteinertehnoloogiat tarkvarasüsteemi juurutamisel seal, kus see on otstarbekas.
   - **Bamboo** - toetab süsteemi konteinerdamist ja juurutamist Dockeri tehnoloogial.
   - **Jenkins X** - toetab nii Dockerit kui konteinerite orkerstreerimisplatvormi Kubernetest.
   - **Spinnaker** - toetab alusplatvormina ainult Kubernetest ja virtuaalmasinat.
10. **Jenkins CI platvormiteenuse püsiv lõimimine**
    - **Bamboo** - ei toeta. Lõimimine on tagatud ainult tarkvaraliidese (API) kaudu.
    - **Jenkins X** - toetab autoriseerimisliidese kaudu.
    - **Spinnaker** - toetab autoriseerimisliidese kaudu.
11. **Bitbucket koodivaramu püsiv integratsioon**
    - **Bamboo** - toetab.
    - **Jenkins X** - toetab pistikprogrammi kasutamisel.
    - **Spinnaker** - toetab.
12. **Kubernetese orkestreerimisplatvormi püsiv integratsioon**
    Kasutame ettevõttes konteinerrakenduste alusplatvormina põhiliselt Kubernetest.
    - **Bamboo** - ei toeta. Lõimimine on tagatud skriptimisvõtete abil.
    - **Jenkins X** - toetab. Lõimimine on tagatud Helm (Kubernetese paketihaldur) laiendusega.
    - **Spinnaker** - toetab.
13. **Prometheuse monitooringuteenuse püsiv integratsioon**
    Kasutame ettevõttes Prometheusi serveritarkvara erinevate süsteemi mõõteandmete salvestamiseks ja monitoorimiseks.
    - **Bamboo** - toetab eraldi pistikprogrammi kasutamisel.
    - **Jenkins X** - toetab eraldi pistikprogrammi kasutamisel.
    - **Spinnaker** - toetab. Lisaks on toetatud sellised monitooringutööriistad nagu

Stackdriver, Datadog, Signalfx ja New Relic.

14. **Kanaarianalüüsi tugi**
    Struktureeritud andmed võimaldavad tarkvarasüsteemi analüüsil kasutada matemaatilisi meetodeid statistiliste järelduste tegemiseks.
    - **Bamboo** - ei toeta.
    - **Jenkins X** - toetab eraldi pistikprogrammi kasutamisel, ent nõuab toimiva lahendusena Kubernetese platvormiteenuse häälestamist.
    - **Spinnaker** - toetab sisse ehitatud tarkvaramooduli *Kayenta* abil.

Juurutusvahendite funktsioonide hindamisel selgus, et sobivaim kandidaat töö praktilises osas loodava prototüübi ühe komponendina osutus Spinnaker. Spinnaker kogus võrdlusanalüüsil 14 punkti, millele järgnes Jenkins X 12,5 punktiga ning kolmandaks tuli Atlassiani CI/CD tööriist 10,5 punktsisummaga. Spinnaker kogus hindamisel maksimumpunktid, milleks oli 14, edastades konkurente vastavalt 3,5 (Bamboo) ja 1,5 (Jenkins X) punktiga.

Spinnakeri tugevusteks konkurentide ees võib pidada kanaarianalüüsi ja hübriidpilve tuge, mis konkurentidel on kasin või puudulik. Jenkins X paistis positiivselt silma samuti kõigis kategooriates, kogudes vähemalt 0,5 punkti iga karakteristiku kohta. Bamboo seevastu kogus võrdlusanalüüsil kõige vähem punkte, mis kajastub Bamboo väheses integratsioonitoes. Selle põhjuseks võib pidada tootjalukustust ehk tootja soovi suunata tarbijaid kasutama enda tooteid ja teenuseid.

Spinnakeri tugevusi varjutab platvormiteenuse arhitektuuriline keerukus aga ka juurutamise ja häälestamise raskustase. Süsteem moodustub ligi kümnest eraldiseisvast hajusüsteemist ja andmebaasist, mille kavandamisel tuleb lähtuda arvestatava alusplatvormi ja IT-taristu ressursiga.

Bamboo tugevuseks võib pidada platvormiteenuse pistikprogrammide arvu, mida leiab töö kirjutamise hetkel Atlassiani E-poest (ingl.k. *marketplace*) 191 tükki, millest veidi üle 1/3 on tasulised. [17] Kaasaegne ja lihtne kasutajaliides ning lõimumine teiste Atlassiani toodetega moodustab sellest hästi toimiva terviksüsteemi. Bamboo kahjuks räägib 2020. aasta sügisel ilmunud uudis isemajutatava tootetoe lõppemisest 2024. aastal. See otsus on ajendanud erinevaid IT- ja telekommunikatsioonieттevõtteid otsima alternatiivseid isemajutatavaid lahendusi. Lisaks räägib alternatiivsete lahenduste kasuks Bamboo kõrge litsentsitasu ja pistikprogrammide soetusmaksumus.

Jenkins on erinevate paigaldusstsenaariumide lihtsustamiseks andnud välja uue toote Jenkins X, kuid millel puudub tootja poolt arendatav haldusliides. Haldusliides on lisatav eraldi pistikprogrammina. Samuti puudub haldusliideses võimalus lihtsamate operatsioonide täitmise võimalus. Jenkins X lähtub tarneprotsesside kirjeldamisel GitOps

praktikatest, seades tehnilisele oskusteabele kõrgemad nõudmised. GitOps keskendub tarkvaraprojekti ja IT-taristu reprodutseerimise printsiibile, mille järgi süsteemide ja rakenduste kood salvestatakse koodivaramusse, võimaldades selle hilisemat kasutamist. [2]

Tabel 1. Juurutusvahendite funktsioonide hindamismaatriks

Funktsioon	**Bamboo**	**Jenkins X**	**Spinnaker**
Hübriidpilve tugi	0,5	0,5	1
Veebihaakide tugi	1	1	1
Rakendusliidese tugi	1	1	1
Ainulogimise tugi	1	1	1
Rollipõhine pääsukontroll	1	1	1
Pistikprogrammide tugi	1	1	1
Töövoogude kirjeldamine koodina	1	1	1
Tarkvaratoote juurutamine konteinertehnoloogial	1	1	1
Haldusliidese olemasolu	1	0,5	1
Jenkins CI platvormiteenuse püsiv lõimimine	0	1	1
Bitbucket koodivaramu püsiv lõimimine	1	1	1
Isemajutatava Kubernetese orkestreerimisplatvormi püsiv lõimimine	0	1	1
Prometheuse monitooringuteenuse püsiv lõimimine	1	1	1
Kanaarianalüüsi tugi	0	0,5	1
Punktisumma	**10,5**	**12,5**	**14**

3.5 Mittefunktsionaalsete nõuete võrdlus

Lisaks tarkvarasüsteemi funktsionaalsetele omadustele ehk tarkvarasüsteemi võimekusele on soovitatav toote valikul lähtuda erinevatest mittefunktisonaalsetest nõuetest, mis iseloomustavad süsteemi kasutatavust. Selles alapeatükis on kirjeldatud võrreldavate toodete omadused, millele vastavust kontrollitakse iga tarkvaratoote juures eraldi. Hinnangute andmisel lähtus autor subjektiivsest arvamusest. Kriteeriumid on toodud välja tabelis:

Tabel 2. Juurutusvahendite mittefunktsionaalsed nõuded

Kriteerium	**Bamboo**	**Jenkins X**	**Spinnaker**
Õigused ja litsents	Omanduslik litsents	Apache 2.0 litsents	Apache 2.0 litsents
Toote hind	Tasuline - hind sõltub hinnastamise tingimustest	Tasuta	Tasuta
Toote seadistamine	Graafilise kasutajaliidese kaudu	Käsurealiidese kaudu	Käsurealiidese abil
Toote dokumentatsioon	Põhjalik	Keskmiselt ülevaatlik	Keskmiselt ülevaatlik
Laiendatavus	Atlassiani E-poest leiab üle 100 tasuta pistikprogrammi.	Pistiprogrammide arv on kasin ehk alla 50	Ametlik pistikprogrammide varamu puudub
Süsteemi jälgimine	Prometheuse tugi ja logiinfo	Prometheuse tugi ja logiinfo	Prometheuse tugi, ja logiinfo
Süsteemi turvamine	Toetab rollipõhist ligipääsumudelit ressurssidele ja funktsionaalsusele	Toetab rollipõhist ligipääsumudelit ressurssidele ja funktsionaalsusele	Toetab rollipõhist ligipääsumudelit ressurssidele ja funktsionaalsusele

Võrdluses selgus, et Spinnaker ja Jenkins X vastavad võrdselt kirjeldatud omadustele. Mõlema toote litsents lubab tarkvara kasutamist ärilisel eesmärgil ning seda vabalt levitada. Bamboo seevastu on tasuline, mille hind sõltub hinnastamise tingimustest ja tarbija konkreetsetest vajadustest.

Spinnakeri ja Jenkins X tarkvarasüsteemi paikamist lihtsustab selleks spetsiaalselt loodud käsurearakendus. Rakendus võimaldab platvormiteenuse uuendamist, muutmist, paigaldamist ja muul viisil juhtimist, mis kasutajaliideses on ebamugav. Bamboo konfigureerimisel ja häälestamisel tuleb leppida kasutajaliidesega, mis võib olla ajaliselt mahukas ettevõtmine.

Autori hinnangul on Spinnakeri ja Jenkins X toote dokumentatsioon võrreldes Bambooga kasin ja üldsõnaline. Bamboo dokumentatsiooni hindab autor põhjalikuks - teave on hästi struktureeritud ja lihtsasti leitav.

Bamboo puhul hakkab silma pistikprogrammide lai valik, millest 1/3 on tasulised. Jenkins X moodulite koguarv jääb veidi alla viiekümne. Spinnakeril seevastu puudub ametlik pistikprogrammide varamu. Funktsioonide võrdlusanalüüsis aga selgus, et kõikidel toodetel on olemas pistikprogrammide arendamiseks ja liidestamiseks vajalik tugi.

Süsteemi töö jälgimiseks pakuvad kõik tootjad ühtemoodi sarnaseid võimalusi - logimine, mõõteandmete esitusviis REST-teenustena jne. Kõik platvormiteenused omavad pääsukontrolli mehhanismi, mis võimaldab lõppkasutajate rollipõhist piiramist funktsionaalsusele ja ressurssidele, mille kaitsmine on süsteemi toimimise seisukohalt vajalik.

Mittefunktsionaalsete omaduste võrdluses selgus, et Jenkins X ja Spinnakeri kasutatavuse näitajad on peaaegu identsed. Mõlemad tooted on tasuta, kuid sellisel juhul peab organisatsioon pakkuma tarbijatele ise tootetuge ja lahendusi probleemide korral. Jenkins X ja Spinnakeri haldamise ja konfigureerimise teeb mugavaks käsurearakenduse olemasolu, mis võimaldab süsteemi mugavat reprodutseerimist ja ehitamist. Bamboo kahjuks räägib litsentsitasu, mis võib olla sõltuvalt kriteeriumidest väga kõrge. Lisaks teeb Bamboo haldamise ebamugavaks konfiguratsiooni koondamine haldusliidesesse. Bamboo tugevusteks võib pidada tasuta pistikprogrammide arvu ja kasutajasõbralikku dokumentatsiooni. Antud võrdluses kujuneks parimaks tooteks Jenkins X, mis edestas napilt Spinnakeri pistikprogrammide kriteeriumide poolest.

4. Tarkvaraarendusmeeskondade küpsustaseme analüüs

Autor viis ettevõttes läbi küsitluse, mille abil selgitati välja meeskondade tehniline küpsustase ja valmisolek automatiseeritud juurutuse töövoo kasutuselevõtuks. Tehniliste kompetentside kaardistamiseks kasutati J. M. Radstaak poolt 2019. a. avaldatud magistritöös "DevOps maturity model: A validated model to evaluate the maturity of DevOps in organizations" välja töötatud arenduse ja halduse küpsustaseme hindamise raamistikku. Hindamisel võeti vaatluse alla vaid tarkvara tarneprotsesse kirjeldavad kategooriad.

Kogutud andmete põhjal analüüsiti ettevõtte meeskondade pädevuste taset kategooriates: 1) tarkvara tarne 2) taristu ehitamine 3) tarkvara ehitamine 4) tarkvara juurutamine 5) automatiseerimisvahendite kasutamine ja 6) monitooring. Iga kategooria juures arvutati vastuste põhjal tulemus, mis tähistab konkreetset pädevuste taset. Seejärel pakkus autor välja juurutusprotsessi automatiseerimise eelduste põhjal soovitusliku taseme. Analüüsi tulemusel tekkis ülevaade ettevõtte tasemest üle valdkondade, konkreetsemalt tarkvaraarendusmeeskondade juures. Lisaks näitas analüüs, kas tarkvaraarendusmeeskonnad on valmis käesoleval hetkel rakendama diplomitöös välja töötatud juurutusmudelit, mida toetab valminud prototüüp.

Küsitluse valim moodustati Telia Eesti AS erinevate funktsioonidega üksustest, kuhu kuulusid meeskonnad, kelle põhitegevuseks on tarkvara arendamine aga ka ettevõtte tuumiktaristu haldamisega tegelevad osakonnad. Küsimustik jaotati kaheks osaks. Esimene osa keskendus hetkeolukorrale, kus mõõdeti meeskondade pädevuste taset tarkvara tarneahela automatiseerimise eeldustel ning teine osa uuris, millised on meeskondade seas praktikas levinumad juurutusmeetodid ja takistused protsesside automatiseerimiseks. Pädevuste kategooriad esitati viietasemelisel skaalal, kus iga kategooria juures tuli teha hetkeolukorda kirjeldav parim valik. Taseme valikul kehtis pärimuse nõue ehk meeskonna olukorra kirjeldus pidi vastama ka eelnevate tasemete kriteeriumidele. Küpsusmudeli järgi vastas tasemele 1 algtase ehk väga nõrk, tasemele 2 juhitud, tasemele 3 defineeritud, tasemele 4 mõõdetud ning tasemele 5 optimeeritud ehk väga hea. Kõikide tasemete täpsem selgitus on lisatud tabelisse 3. Hindamise tulemused ja järeldused on lisatud peatüki lõppu. Küsimustiku teises pooles sõnastati kaks valikvastustega küsimust kompetentside hindamise lõppresultaadi täiendusena.

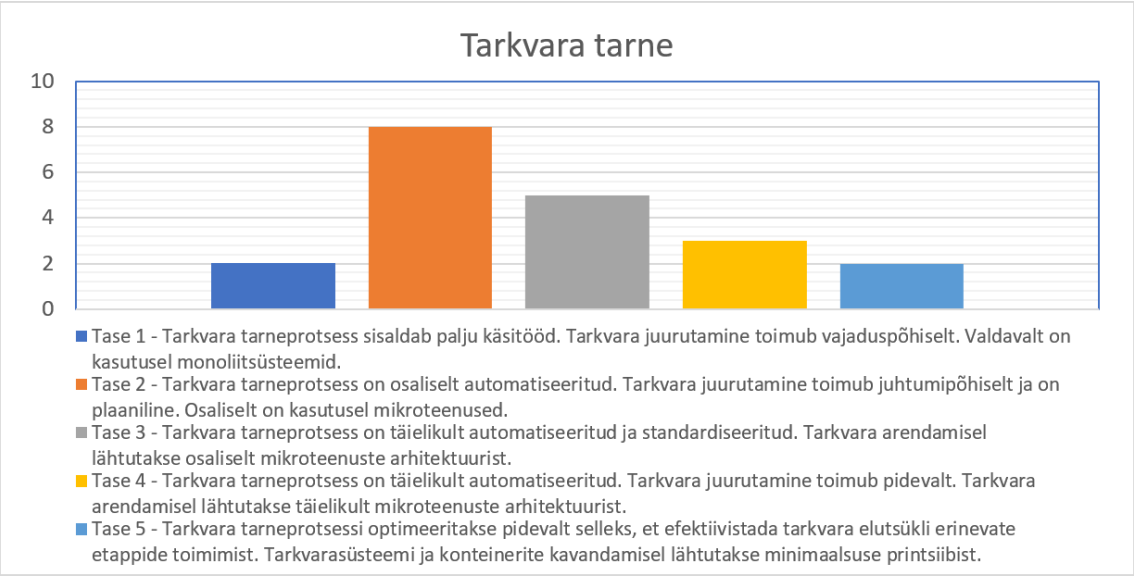
Kõik küsimused esitati elektroonilises Google Formsi keskkonnas eesti keeles, millele vastas 20 meeskonda, moodustades 43,5% kogu valimist. Küsitlus viidi läbi perioodil 22.02 - 31.03.2021.

Tabel 3. Hindamise tasemed

Tase	**Kirjeldus**
algtase	Meeskond pole tegelenud tööprotsesside automatiseerimisega või puudub vajadus selle rakendamise järele. Süsteemide paikamine, juurutamine ja konfigureerimine toimub käsitsi. Valdavalt on kasutusel monoliitsüsteemid. [18]
juhitud	Tööprotsessid on osaliselt automatiseeritud, kuid suur osakaal on endiselt käsitööl. Tarkvara ehitamine ja juurutamine on osaliselt standardiseeritud. [18]
määratletud	Tarkvara tarneprotsessi kõik etapid on standardiseeritud ja automatiseeritud. Juurutamine toodangusse toimub endiselt käsitsi. Tarkvaraprojekti taristu juures kasutatakse võimalusel virtualiseerimist. [18]
mõõdetud	Tööprotsessid on täielikult automatiseeritud ja monitooritud, mis võimaldab tarkvarasüsteemi juurutamist toodangukeskkonda automaatselt. Süsteemi uuendamisel on tagatud täielik käideldavus. Tarkvara arendamisel lähtutakse mikroteenuste arhitektuurist. [18]
optimeeritud	Meeskond tegeleb tööprotsesside pideva optimeerimisega, eesmärgiga kiirendada tarkvara tarnetsüklite kestust ja sagedust. Tarkvara ja selle taristu on esitatud koodina ehk programmeeritavalt. [18]

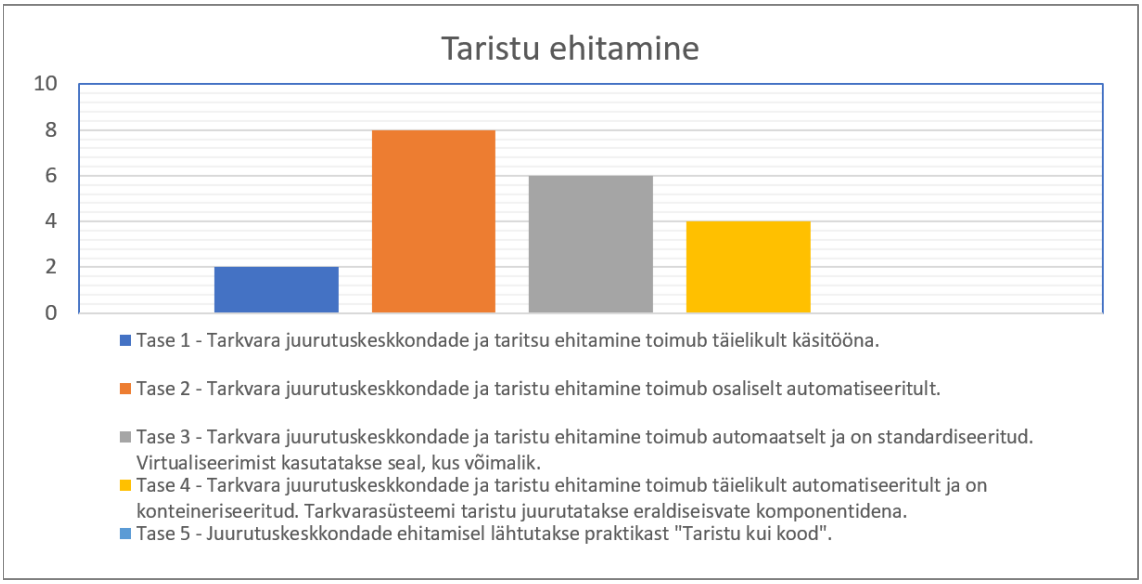
4.1 Hindamiskategooriate ülevaade

Esmalt paluti hinnata olukorda tarkvara tarne automatiseerimise üldvaates. Selle kategooria eesmärgiks oli tuvastada kui mitu meeskonda on automatiseerinud kogu tarkvara tarneahela s.o tarkvara tarnepaki loomisest kuni toodangukeskkonda paigaldamiseni välja. 50% vastanutest hindas end vähemalt tasemele 3, mis viitab asjaolule, et umbes pooled meeskonnad kasutavad tarkvara arendamisel moodsaid tehnoloogiaid ning lähtuvad tarkvara arendamisel arhitektuuri mudelist, mis toetab virtualiseerimist. Nendest omakorda pooled (vastavalt tase 4 ja 5) kasutavad rakenduste virtualiseerimisel põhiliselt konteinertehnoloogiaid, moodustades $1/4$ kõikidest vastanutest. Ligi pooled rõhutasid seost pärandsüsteemidega ning tõid välja tarnete olematu või osalise automatiseerituse. Tarneprotsessi üldtulemusi kirjeldab jooni 7.



Joonis 7. Küsimustik - tarkvara tarne

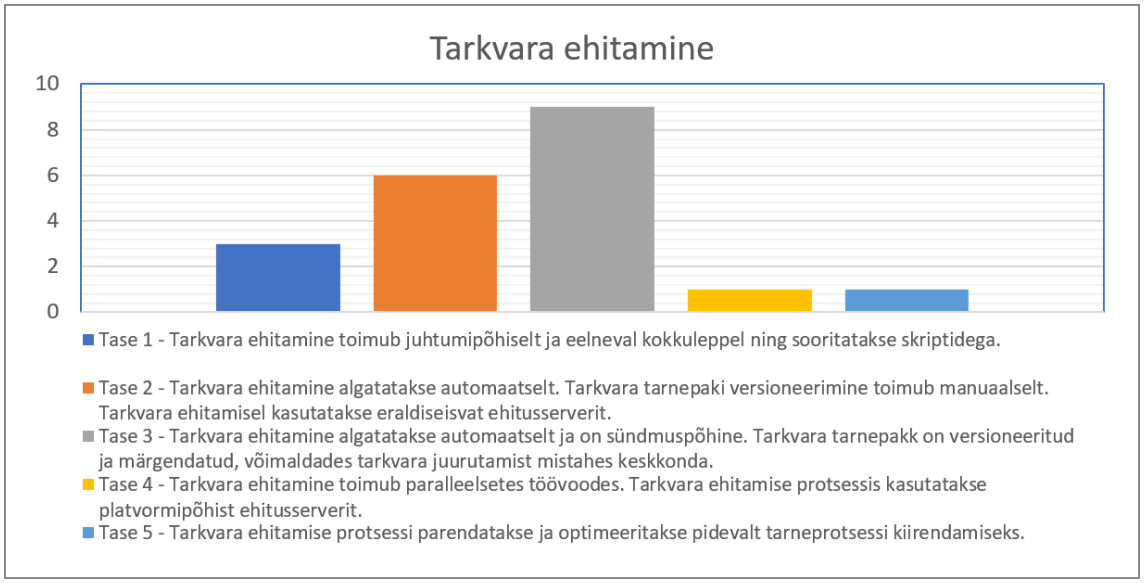
Tarneprotsessi planeerimisel on oluline, et tarkvara toetav taristu oleks taasloodav ja parameetritega juhitav. Sellele tingimusele vastas 50% kõikidest küsitluses osalejatest, vastavalt tase 3 ja 4. Tase 3 seab nõudeks tarkvara taristu osalise virtualiseerituse, ent tase 4 seevastu eeldab konteinertehnoloogiate kasutamist taristu loomeprotsessis. 10 meeskonda ehk 50% on märkinud enda tasemeks algtase või juhitud, mis viitab püsiva ja tarkvara jaoks kohandatud taristu kasutamisele.



Joonis 8. Küsimustik: taristu ehitamine

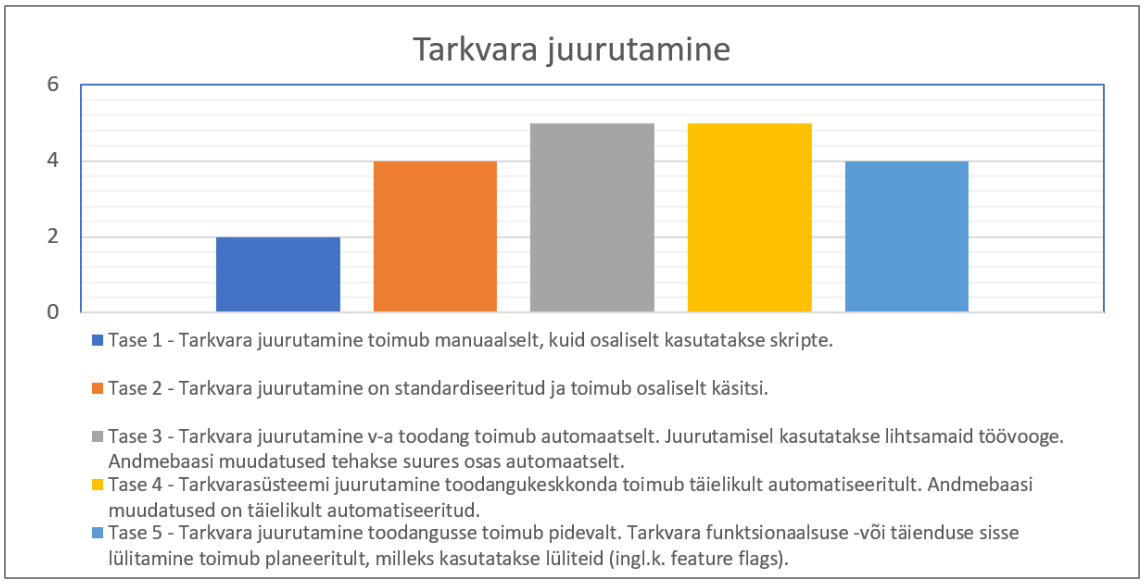
Tarkvara ehitamise kategoorias hinnati meeskondade võimet toota tarkvara efektiivselt ja viisil, mis võimaldaks selle pidevat juurutamist mistahes töökeskkonda. 15% vastanutest

valmistab tarkvara tarnepaki ette käsitsi või skriptide abil. Teisele tasemele määratles kuus (s.o 30%) meeskonda, kes kasutavad tarkvara ehitamiseks automatiseerimisvahendeid, kuid ei ole protsessi lõpuni automatiseerinud. Üle poole ehk 55% vastanutest on suutelised tarkvara tarnima efektiivselt ning pidevalt. Umbes 20% nendest pooltest on tegelenud tarkvara tootmisprotsessi järjepideva edasiarendamisega, kasutades selleks erinevaid tehnoloogiaid ja töövõtteid.



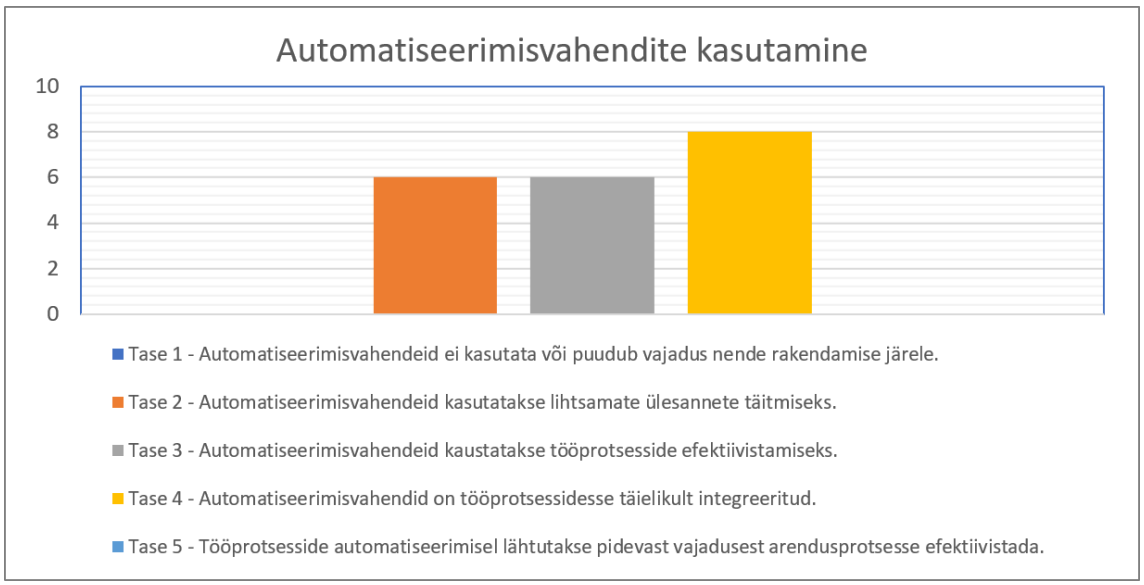
Joonis 9. Küsimustik: tarkvara ehitamine

Neljandas kategoorias hinnati meeskondade küpsust tarkvara väljalaske protsessi automatiseerimises. 30% vastanutest kasutavad tarkvarasüsteemi juurutamiseks peamiselt skripte ning teise poole moodustab käsitöö (vastavalt tase 1 ja 2). 25% hinnatavatest on automatiseerinud tarkvara juurutamise arendus- ning testkeskkonnas, kuid on teevad seda toodangus manuaalselt. $1/4$ s.o 25% on täielikult automatiseerinud juurutamise töövoo nii arendus-, test-, kui ka toodangukeskkonda ning ülejäänud 20% kasutab seejuures täiendavalt tarkvara funktsionaalsuse või -täienduse kasutuselevõtu meetodina lüliteid (ing. k. *feature flags*).



Joonis 10. Küsimustik: tarkvara juurutamine

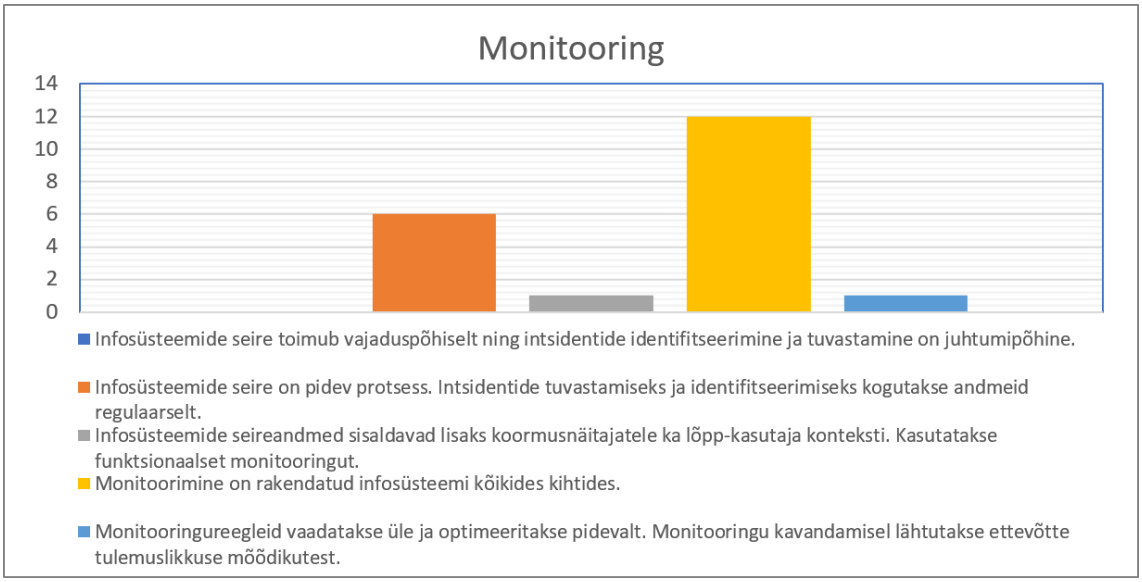
Viies kategoorias hindas erinevate DevOps tööriistade ja automatiseerimisvahendite kasutamist arendusprotsessis. Nendeks vahenditeks on näiteks koodivaramu, töövoo haldussüsteemid, konteinertehnoloogiad, monitooringulahendused jne. 30% kasutab seesuguseid IT-teenuseid lihtsamate ülesannete täitmiseks. Ülejäänud 70% (tase 3 ja 4) on lõiminud automatiseerimisvahendid tarneprotsessi sisse.



Joonis 11. Küsimustik: automatiseerimisvahendite kasutamine

Viimases kategoorias paluti meeskondadel hinnata seirevahendite kasutamist tarneprotsessi erinevates etappides. Mitte ükski vastanu ei lähtu süsteemide ja teenuste monitoorimisel vajaduspõhisusest. Andmeid kogutakse reaalajas ja on jälgitavad tagasiulatuvalt. Kõik

meeskonnad on teadvustanud monitooringu tähtsust. 60% vastanutest kogub seireandmeid lisaks tarkvarasüsteemidele ka taristu erinevate komponentide kohta. Vaid üks vastaja positsioneerib end tasemele 5 ning tegeleb monitooringu pideva edasiarendamisega, selleks et täita äri ees kokkulepitud eesmärke, näiteks intsidentide kiire ja proaktiivne lahendamine. 5% määratleb end tasemele 3 ning 30% tasemele 2.



Joonis 12. Küsimustik: monitooring

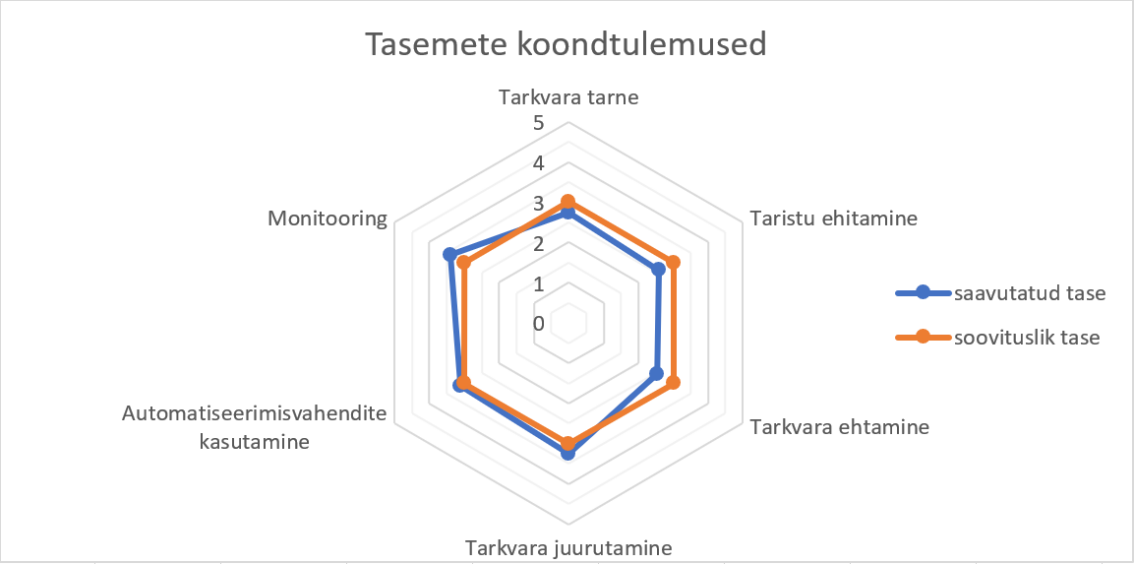
4.2 Kokkuvõte

Soovitud automatiseerimistaseme saavutamiseks tarkvara tarneprotsessi kategoorias tuleb sõnastada tegevused ja algatused, mille abil eesmärgini jõutakse. Küpsustaseme hindamismudel on üheks vahendiks hetkeolukorra kaardistamiseks ja eesmärkide seadmiseks tähtsuse järjekorda. Täpsema taseme määramisel tuleks vaadata igat meeskonda eraldi. Seejuures peaks hindamine toimuma koos meeskonnaga ja võimalusel iga tarkvaraprojekti juures. Hindamisel on soovitatav arvestada teguritega nagu pärandsüsteemide osakaal tarkvaraprojektide portfellis või tehnoloogiate ja töövahendite kättesaadavus, kuna valdkonniti võib meeskondade tehniline tase erineda. Põhjus võib peituda äriliste eesmärkide ebaühtlases tähtsuse järjekorras, arhitektuurilises keerukuses või pädevuste puudumises.

Tarkvara väljalaskeprotsessi täieliku automatiseerimise eeldusena peab autor vähemalt taset 3. Tase kolm seab kasutatavale tehnoloogiale, tarkvara ja taristu arhitektuurile ning tööpõhimõtete ja vahendite kasutamisele kõige sobivamad kriteeriumid, ent ei nõua automaatse juurutamise valmidust toodangukeskkonda. Viimase puhul on mitu võimalust, kuid mille ühe variandina võib kasutada autori poolt diplomitöös kirjeldatud juurutamise töövood ning prototüübi kasutamist. Kanaarijuurutuse kasutuselevõtu eelduseks on

pädevuste soovituslik ehk miinimumtase.

Selleks, et näha, kas organisatsioon on soovituslikul tasemel, arvutati vastuste põhjal iga kategooria kohta tulemus. Tulemuste ja soovitusliku küpsustaseme suhet kirjeldab joonis 13. Joonisel on tähistatud oranžiga soovituslik küpsustase ning sinisega iga kategooria summeeritud keskväärtus. Arvutus näitab, et soovituslikule tasemele vastavad meeskonnad vaid kolmes kategoorias: monitooring, automatiseerimisvahendite kasutamine ja tarkvara juurutamine. Allapoole taset jäid: taristu ehitamine, tarkvara ehitamine ja tarkvara tarne.

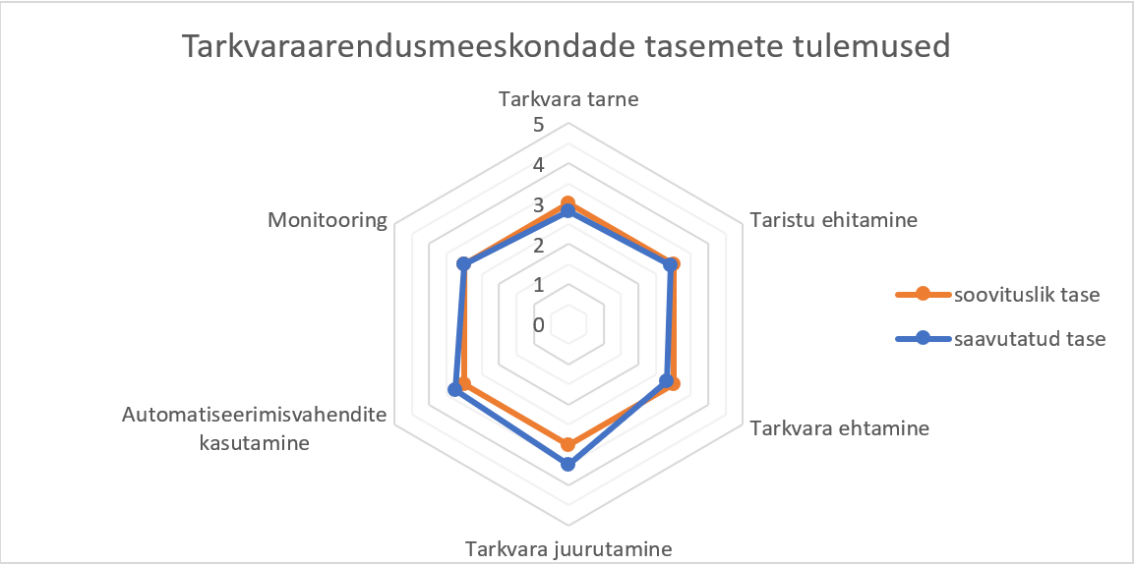


Joonis 13. Küsimustik: tasemete koondtulemused

Koondtulemuste põhjal ei vasta ettevõte tasemele. Põhjuseks võib pidada valdkonnapõhiste osakondade osakaalu küsimustiku vastustes, kus arendus- ja halduspraktikad ning tehnoloogiad erinevad DevOps meeskondade omast. Kui eemaldada valimist tuumiktaristu haldusega tegelevad üksused, mis moodustas kõikidest vastanutest 20%, näeme, et tarkvaraarendusmeeskondade tase nihkus kesktasemele lähemale (Joonis 14). On võimalik, et ülejäänud meeskonnad, kes on täna jäänud valimist välja, võivad lõpliku tulemust mõjutada veelgi. Küll aga näitab valimi kitsendamine, et tarkvaraarendusmeeskonnad asetsevad soovitud tasemel või selle piirjoonel.

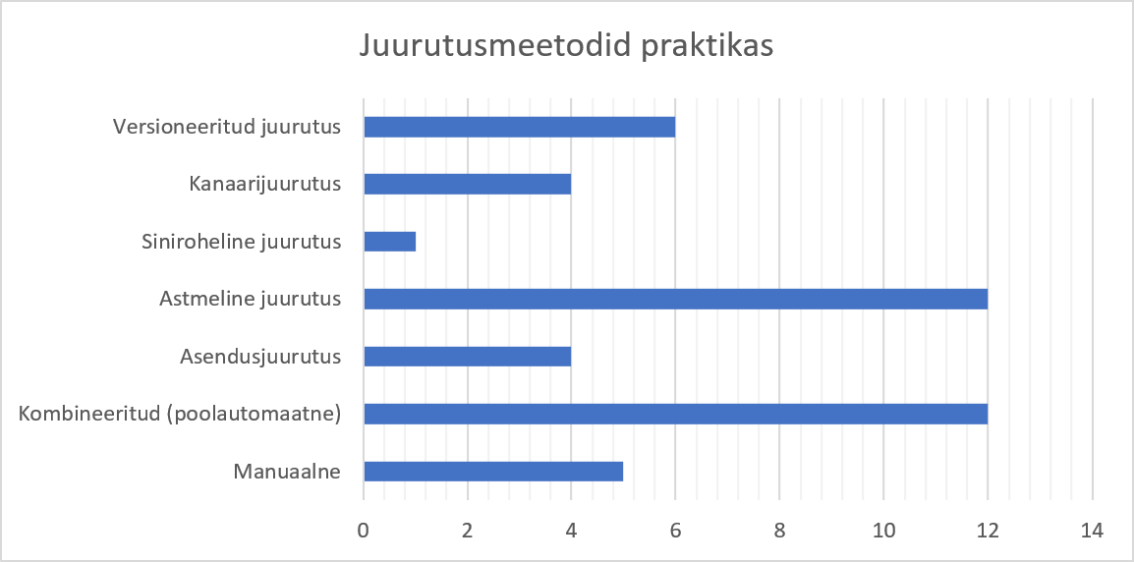
Joonis 14 näitab, et tarkvara tarne põhietapid v-a juurutamine on kesktaseme piiril ja tarne üldvaatega ühtlasel joonel. Juurutamise ja ülejäänud kategooriate tasemete vahe ei ole suur, kuid siiski arvestatav. Tarkvara juurutamise graafikul hakkab silma kõrge vastanute arv tasemel 4 ja 5, mis võib toodangu puhul viidata astmelise juurutusmeetodi kasutamisele ning on automatiseerimise seisukohalt kõige lihtsamini rakendatav meetod. Taseme 5 juures on meeskonnad rakendanud funktsionaalsuse tarnimise ja kasutuselevõtu meetodina lüliteid ehk *feature flags*, mis tingimata ei nõua tarkvara väljalaske protsessi

automatiseerimist.



Joonis 14. Küsimustik: tarkvaraarendusmeeskondade tasemete tulemused

Meeskondade seas levinumate juurutusmeetodite kasutamise kohta esitas autor täpsustava küsimuse "Millised juurutusmeetodid on teie meeskonnas kasutusel?", mille vastused on näha joonisel 15.

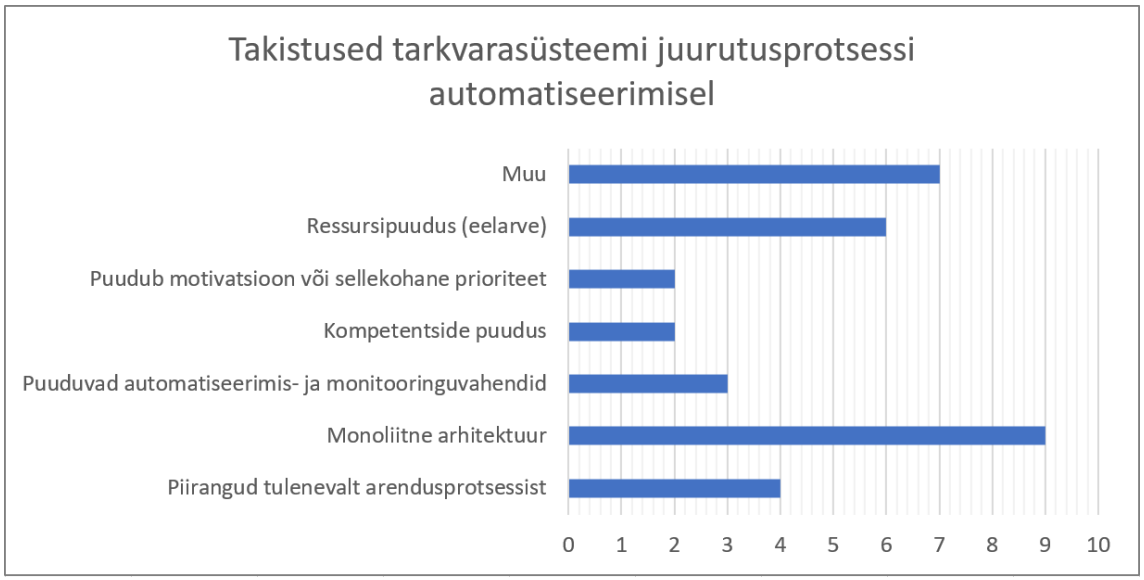


Joonis 15. Küsimustik: juurutusmeetodid praktikas

Vastustest selgus, et kõige populaarsem meetod tarkvara uuendamisel on astmeline juurutusmeetod. Sellele järgnes kombineeritud ehk poolautomaatne ning lisandus manuaalne juurutusmeetod. Automatiseeritud juurutusprotsessi seisukohalt võib vastuste seast leida veel versioneeritud, siniroheline ja kanaarimeetodid. Versioneeritud

juurutusmeetodi puhul võib tegemist olla nii automaatse kui manuaalse strateegiaga, kuid selles uurimuses arvatakse see automaatsete meetodite kategooriasse. Kanaarimeetodi puhul uuris autor vastanute tehnilise lahenduse tausta, millest selgus, et kasutusele võetud lahendus ei lähtunud 100% kanaarijuurutuse tööpõhimõtetest. Lahendus nägi ette osalist HTTP-liikluse suunamist kanaarirakendusse, ent uuenduse järgse süsteemi valideerimisega tegeles tarkvaraarendusmeeskond käsitsi, toetudes logidele ja monitooringuandmetele. Seega on tegemist küll kanaarijuurutusega, ent mille põhifunktsioon s.o versiooni valideerimine toimub käsitööna.

Kuna tarkvara väljalaskel võib meeskondade juures kohata arvestataval määral käsitööd, esitas autor lisaküsimuse, kus palus teha valiku enamlevinud takistuste vahel juurutusprotsessi automatiseerimisel. Ülevaadet kõikidest vastustest leiab jooniselt 16.



Joonis 16. Küsimustik: takistused juurutusprotsessi automatiseerimisel

Selgus, et peamise piiranguna näevad meeskonnad tarkvara arhitektuurist tulenevaid puudujääke. Monoliitsüsteemid on põhjusena välja toodud 12 korral. Teisel kohal figureerib eelarveliste vahendite puudumine, kokku 6 vastust. Mõlema põhjuse leevendamisel tuleb organisatsiooni poolt seada konkreetsed initsiatiivid ning tegevuskava olukorra parandamiseks. Esimese puhul on tegemist pikaajalise projektiga, mis nõuab ettevõtteülese strateegia formuleerimist. Kolmandana toodi välja piirangud arendusprotsessis, mille põhjuseid peavad meeskonnad analüüsima iseseisvalt. Teiste takistustena mainiti veel motivatsiooni, automatiseerimisvahendite ja vajalike pädevuste puudumist. Nende puhul on tegemist taas vajadusega seada ettevõtteülesed prioriteedid, mille täitmist tuleb järjepidevalt jälgida ning koordineerida. Ühe tõukena eesmärkide kirjeldamisel võib kasutada DevOps küpsustaseme hindamist. Ülejäänud põhjendused, mis esitati valikvastustena, kirjeldasid ärinõuete puudumist või valdkondlike iseärasusi,

kus tarkvara arendusprotsessi sellisel kujul ei järgita.

5. Prototüübi ülevaade ja juurutamine

Järgnevas peatükis teeb autor ülevaate kanaarijuurutuse töövoost ning praktilise tööna valminud prototüübist. Tarkvara juurutusvahendite võrdluses selgus, et sobivaimaks platvormiteenuseks prototüübi kavandamisel osutus Spinnaker. Spinnaker on prototüübi raamsüsteem, mis seob ja juhib erinevaid konteinerrakenduste paigaldamisega seotud ülesandeid. Prototüübi tehnilise osa moodustab kobarsüsteem, mille juurutamist töös automatiseeritakse. Tarkvara kobarsüsteem on ökosüsteem erineva rolliga konteinerrakendustest, mille eesmärk on toetada versiooniuuendust aegrida analüüsil. Tarkvara juurutamisel lähtus autor kanaarijuurutuse tööpõhimõtetest.

Enne tarkvara kobarsüsteemi kirjeldamist tuleb selgitada kanaarijuurutuse tööpõhimõtet. Niisiis on tegemist tarkvara juurutamise ühe meetodiga, mille käigus juurutatakse tarkvaraprojekti viimane ehk uusim versioon põhirakenduse kõrvale. Uus versioon (ingl.k. *canary version*) paigaldatakse eraldiseisva tarkvarasüsteemina. Kanaariversiooniga paralleelselt paigaldatakse tarkvaraprojekti etalonversioon, mis omab põhirakendusega identset versiooni. Seejärel võrreldakse kanaari- ja etalonversioonis tekkinud vigade ja muude anomaaliate esinemissagedust, mille analüüsi tulemusel tehakse otsus põhirakenduse uuendamiseks kandidaatversioonile või versiooni tagasi pööramiseks. [5] Võrdlusanalüüsil kasutatavad tarkvarasüsteemi valideerimise viisid võivad olla erinevad, näiteks jõudlus- või koormustestimine, aga ka näiteks teenuste funktsionaalne testimine.

Kobarsüsteemi moodustavad:

1) Kanaarirakendus (ingl.k. *canary application*) - põhirakenduse tuletis ehk tarkvaraprojekti uue versiooni väljalaske kandidaat;
2) Etalonrakendus (ingl.k. *baseline application*) - põhirakenduse tuletis, mida kasutatakse kandidaatversiooni testimiseks ja valideerimiseks;
3) Põhirakendus (ingl.k. *main application*) - tarkvaraprojekti hetkeseis;
4) Envoy koormusjaotur - kasutatakse sissetuleva HTTP-liikluse suunamiseks kanaari-, etalon-, ja põhirakendusse.
5) Envoy vaheserver - kasutatakse konteinerrakenduse (kanaari ja etalon) liidesena (ingl.k. *sidecar*) HTTP-päringute struktureeritud kujul esitamiseks.

Praktilise töö kavandamisel tutvus autor Spinnakeri ülesseadmiseks vajalike juhiste ja spetsifikatsiooniga. Seejärel kirjeldati platvormiteenuse ja selle tugiteenuste (näiteks

andmebaasiserver ja pöördpuhverserver) paigaldamiseks ja häälestamiseks vajalikud tegevused. Kõik konfiguratsiooniks vajaminevad käsud on lisatud käesoleva töö lisadesse. Töö ei sisalda Dockeri teenuse, virtuaalmasinate või Kubernetese platvormiteenuse seadistamiseks vajaminevaid käske ja selgitusi. Tegevuste kirjeldustes on töökäskudes ja kuvatõmmistes hägustatud teave, mille abil on võimalik tuvastada ettevõttes kasutusel olevaid teenuseid, aadresse, kasutajaid, versiooninumbreid jm info, mida ettevõtte ei soovi avaldada.

Kobarsüsteemi projekteerimisel ja komponentide valikul lähtus autor varem analüüsitud kriteeriumidest. Süsteemi visualiseerimiseks kirjeldas autor paigaldusdiagrammi ning dokumenteeris ülesseadmiseks vajalikud tegevused.

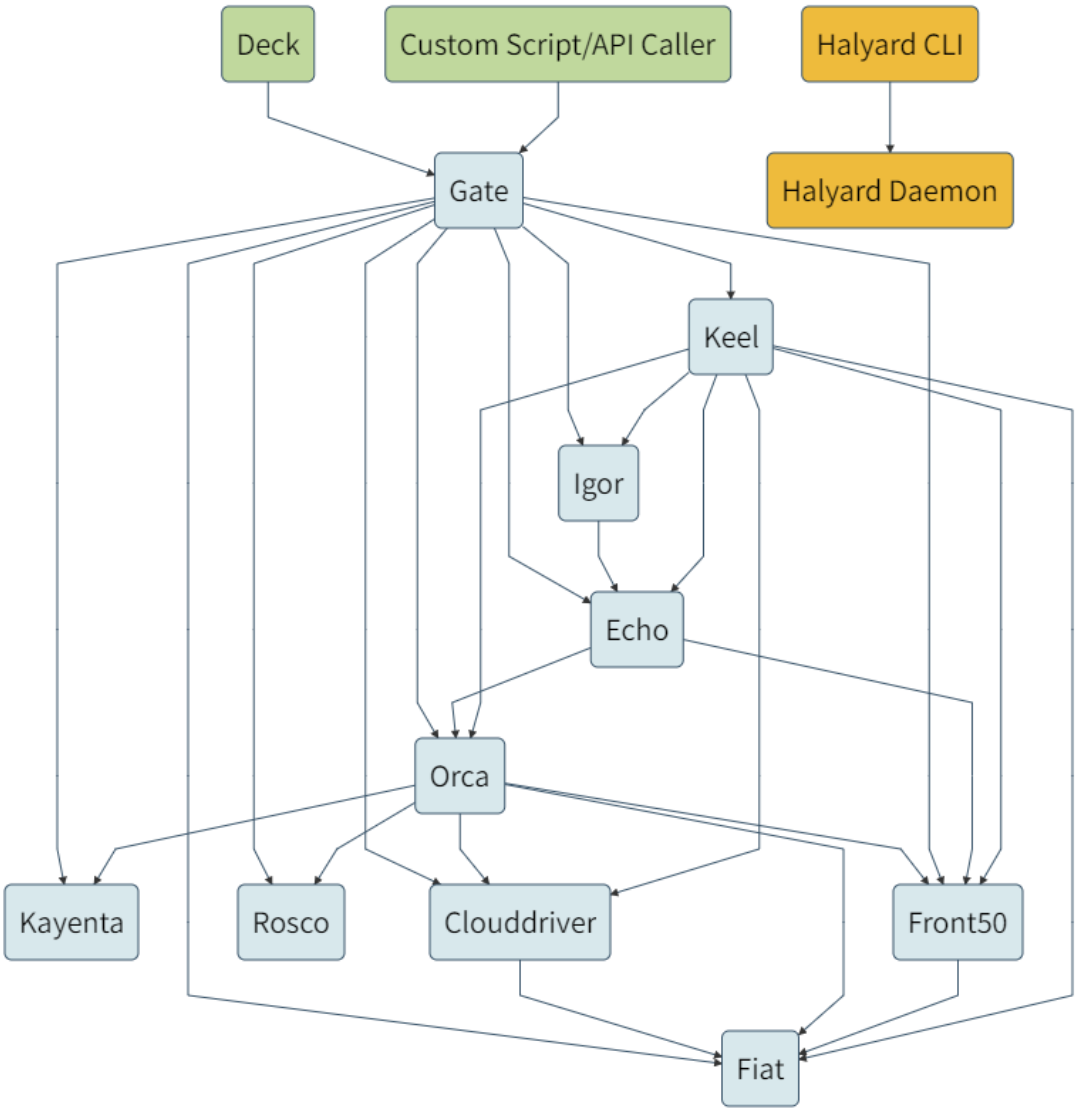
5.1 Spinnakeri juurutusviisid ning nõuded riist- ja tarkvarale

Tootja soovitab Spinnakeri juurutada kahel viisil: hajussüsteemina Kubernetese orkestraatorile või virtuaalmasinale. Mõlema meetodi puhul on toetatud nii pilvepõhine juurutamine kui ka isemajutatav juurutusviis. Tarkvarasüsteemi paigaldamise, konfigureerimise ja uuendamise lihtsustamiseks on tootja loonud eraldi käsurearakenduse Halyard. Seal kus võimalik, tuleks eelistada pilvetaristut. Operatsioonisüsteemina on toetatud Ubuntu 14.04 või 16.04 ning riistvara ressursi planeerimisel seatakse nõudeks vähemalt 4 protsessori tuuma ja 16 GB operatiivmälu. [19]

Spinnakeri juurutamist ja seadistamist on võimalik sooritada ka ilma Halyard käsurearakenduseta. Kuna Spinnakeri puhul on tegemist tarkvarasüsteemiga, mis koosneb erinevatest eraldiseisvatest Spring Boot Java rakendustest (Joonis 17), on võimalik mõningasel planeerimisel juurutada platvormiteenus mistahes viisil. Näiteks konteinertehnoloogia abil hajutatult üle mitme peremeesserveri (*Docker swarm*) või ühe kobarana füüsilisele serverile. Sellegipoolest on soovituslik teha seda Halyardi abil pilvetaristule, mis võimaldab teenuse hilisemat paikamist katkestuseta.

Halyardi käsureatööriista ülesseadmisele on tootja esitanud samuti nõuded. Operatsioonisüsteemi valikul soovitab tootja kasutada Ubuntu versiooni 14.04, 16.04 või 18.04. Lisaks on toetatud Debian 8 või 9 ning MacOS versioon 10.13. Käsureatööriista paigaldamise asukohana võib valida nii serveri kui ka klient-arvuti. Operatiivmälu osas on tootja soovitanud vähemalt 12 GB. [20]

Käesoleva töö raames juurutati Spinnakeri CD platvormiteenus hajussüsteemina ettevõtte privaatsesse pilveplatvormile. Spinnakeri juhtsüsteem paigaldati Dockeri konteineril eraldi virtuaalmasinasse.



Joonis 17. Spinnakeri juurutusdiagramm [21]

5.2 Halyard käsurearakenduse ülesseadmine

Halyard vajab töötamiseks Java kompilaatorit (OpenJDK versioon 11) ning Spinnakeri paigaldamiseks pilvetaristule Linux käsurea haldusvahendeid "*kubectl*", mida kasutatakse Kubernetese platvormiteenusega suhtlemiseks ja "*packer*", mille abil luuakse konteinerite kettakujutisi (*ingl.k. image*). Töös kasutati Halyardi viimast versiooni 1.41.0.

Halyardi ülesseadmiseks kasutas autor varasemalt ette valmistatud virtuaalmasinat, kus olid kriteeriumid Kubernetese platvormiteenusega suhtlemiseks ning Halyardi rakenduse konteinerdamiseks täidetud. Kubernetese platvormiteenusega lõimimisel kasutab Halyard *kubectl* eraldiseisvat käsureatööriista. Selleks tuleb tulemüüris avada ligipääs serverist, kuhu Halyard on paigaldatud, vastu IP-aadressit, mis defineerib Kubernetese kobara asukoha. Kubernetese rakendusliidesega suhtlemiseks kasutatakse tavaliselt TCP pordi

numbreid 80 või 443. Pilvetaristuga ühenduse loomiseks tuleb *kubectl* käsureatööriista jaoks defineerida sellekohane sätetefail. Fail sisaldab infot kobara aadressi ja autoriseeritud kasutaja kohta. Fail tuleb salvestada linuxi kasutaja kodukataloogi *$HOME/.kube/config*. Halyardi käsurearakenduse välja kutsumiseks tuleb kasutada käsku "*hal*", millele järgnevad sisendparameetrid.

Halyardi juurutamiseks konteineris ja rakenduse häälestamiseks kirjeldas autor sätetefaili (*Dockerfile*), mida Dockeri teenus kasutab kettakujutise loomiseks ja konteineri käitamiseks. [Lisa 2] Halyard töötab eraldi Linuxi kasutaja alt "spinnaker".

Tootja dokumentatsiooni järgi on Halyard asendumas uue tööriistaga *kleat* [22], mis tähendab, et Halyardi tootetugi võib lähitulevikus lõppeda.

5.3 Andmebaasiserveri Minio ülesseadmine

Spinnaker vajab süsteemi ja töövoogude konfiguratsiooniparameetrite, mõõteandmete ehk meetrikate ning märgisandmete salvestamiseks eraldiseisvat andmebaasi. Toetatud on erinevad pilvepõhised andmebaasi teenused nagu Azure Storage, Google Cloud Storage, Oracle Object Storage ning isemajutatavad andmebaasiserverid nagu Minio ja Redis. Redis ei ole tootja sõnul sobiv andmebaasiserver konfiguratsiooniparameetrite salvestamiseks toodangus. [23]

Kuna prototüübi loomise nõudeks oli selle juurutamine ettevõtte sisevõrku ning lähtudes Spinnakeri spetsifikatsioonist, võeti andmete salvestusteenusena kasutusele Minio objektorienteeritud andmebaasiserver. Objektorienteeritud andme mudelit kasutatakse andmehalduses keerukamate andmestruktuuride korral, kus ei piisa ainult omaduste loetelude haldusest, vaid kus on vaja sooritada salvestatud andmetega seotud tegevusi või toiminguid. [24]

Minio ülesseadmisel lähtus autor tootja dokumentatsioonist. Töö käigus juurutati Minio viimane versioon, mis oli avalikus GitHubi koodivaramus saadaval. Tootja lubab süsteemi juurutamist nii riistvarale kui konteinerrakendusena. Toetatud on erinevad operatsioonisüsteemid nagu macOS, Linux ja Windows. [24] Autor paigaldas Minio tootja poolt soovitatud vaikesätetega ja konteinerdatult varasemalt ette valmistatud virtuaalmasinasse. Andmebaasiserveri ja haldusliidese kasutamiseks tuli avada tulemüüris port numbriga 8446. Selleks kirjeldas autor ettevõtte pöördpuhverserveris vastava saidi.

Andmete järjepidevuse tagamiseks sidus autor omavahel konteinerrakenduse ja peremeesserveri kettajaod, mis võimaldab andmete hilisemat varundamist otse peremeesserveris. Konteinerrakenduse paigaldamiseks vajalik manifest-fail (*Dockerfile*) on toodud välja töö lisades. [Lisa 3] Dockerfile sisaldab infot Spinnakeri kasutaja ja

salasõna kohta, mida kasutatakse andmebaasi kasutajakonto määratlemiseks. Seda kasutajakontot kasutati hiljem Spinnakeri häälestamisel.

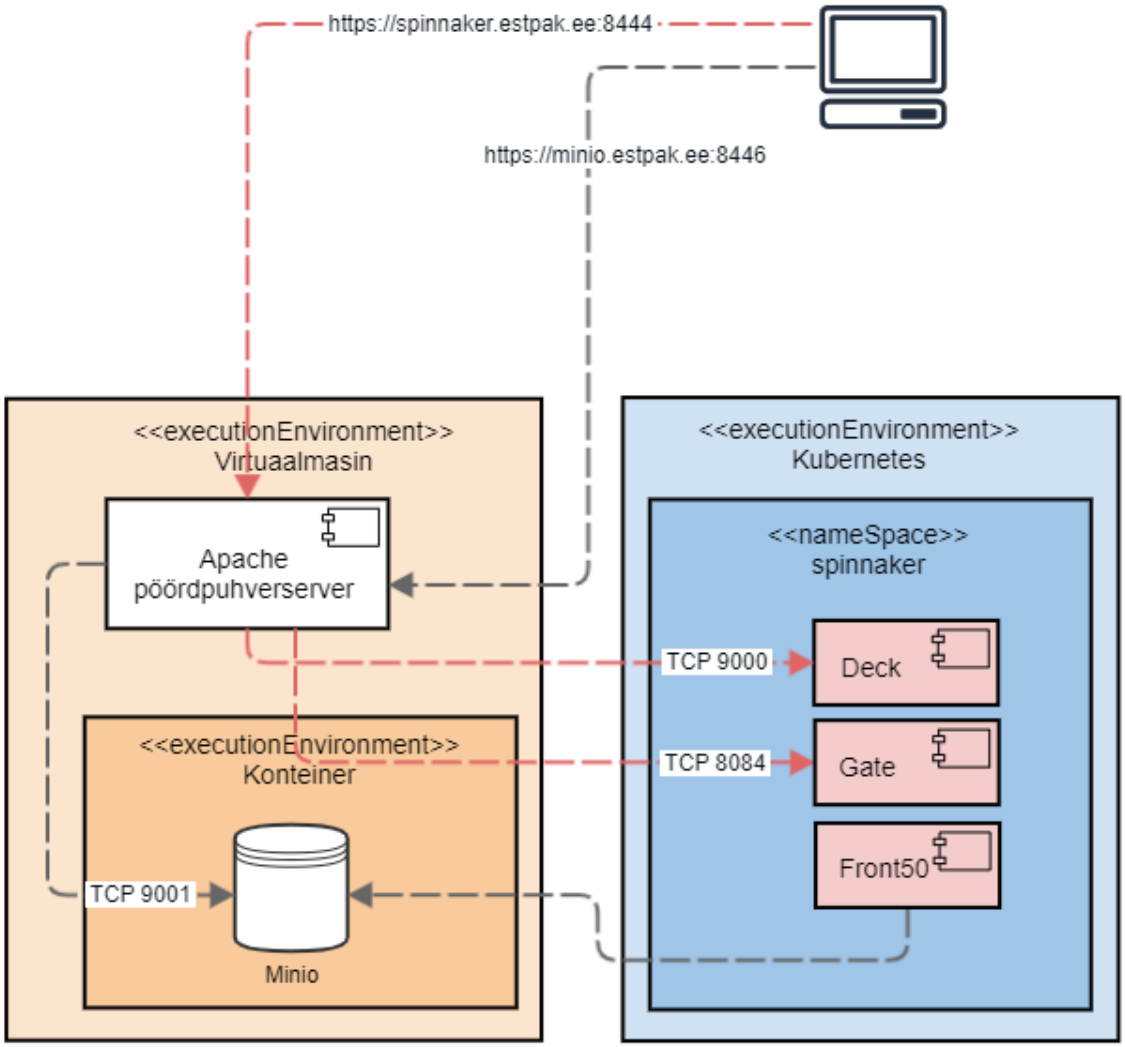
5.4 Spinnaker CD platvormiteenuse paigaldamine ja häälestamine

Selles alampeatükis on kirjeldatud Spinnakeri platvormiteenuse juurutamise ja konfigureerimisega seotud tegevused. Kõik tegevused on esitatud kronoloogilises järjekorras ja eeldavad, et lõimeteenused on võrgus leitavad ja kättesaadavad. Töös juurutati Spinnakeri versioon 1.25.4, mis oli sel hetkel tootja dokumentatsiooni järgi viimane stabiilne versioon.

Spinnakeri ülesseadmise eelduseks on töötav käsurearakendus Halyard, andmebaasiserver Minio ja Kubernetese platvormiteenus. Halyard juurutati konteinerrakendusena eraldi virtuaalmasinas. Selleks, et Halyardi kasutada, tuleb konteinerrakendusse sisse logida. Sisse logimine on saavutatav Dockeri teenuse abil spetsiaalse käsuga.

Spinnakeri seadistamisel Halyardiga salvestatakse sätted eraldi YAML-faili asukohaga "*$HOME/.hal/config*". See fail on Spinnakeri erinevate komponentrakenduste konfiguratsiooni põhjaks. Faili salvestatud võtmeid ja väärtusi kasutatakse Halyardi poolt Spinnakeri konteinerrakenduste hilisemaks seadistamiseks. Niisiis tuleb Spinnakeri häälestamisel lähtuda vaid eelnimetatud faili muutmisest ja salvestamisest. Antud failist tasub teha varukoopia vajadusel süsteemi hilisemaks reprodutseerimiseks.

Spinnakeri haldusliidese kasutamiseks tehti Kubernetese platvormiteenuses täiendus, mis lubab tarkvarasüsteemi privaatse (konteinerpõhise) IP-aadressi ja ettevõte kohtvõrgu IP-aadressi sidumist. Sidumisel kirjeldati pordi numbrid 8084 ja 9000, mis määratlevad haldusliidese pääsukohtasid. Teenuse avamiseks ettevõtte sisevõrgus lubati tulemüürist ligipääs eraldiseisvast Apache'i pöördpuhverserverist Spinnakeri kohtvõrgu IP-aadressi pihta. Lahenduse lõpptulemuse illustreerimiseks on lisatud pilt:



Joonis 18. Spinnakeri haldusliidese paigaldusdiagramm Telia Eesti AS taristul

Kõik juurutamisega seotud käsud ja kommentaarid on lisatud töö lisadesse. [Lisa 4]

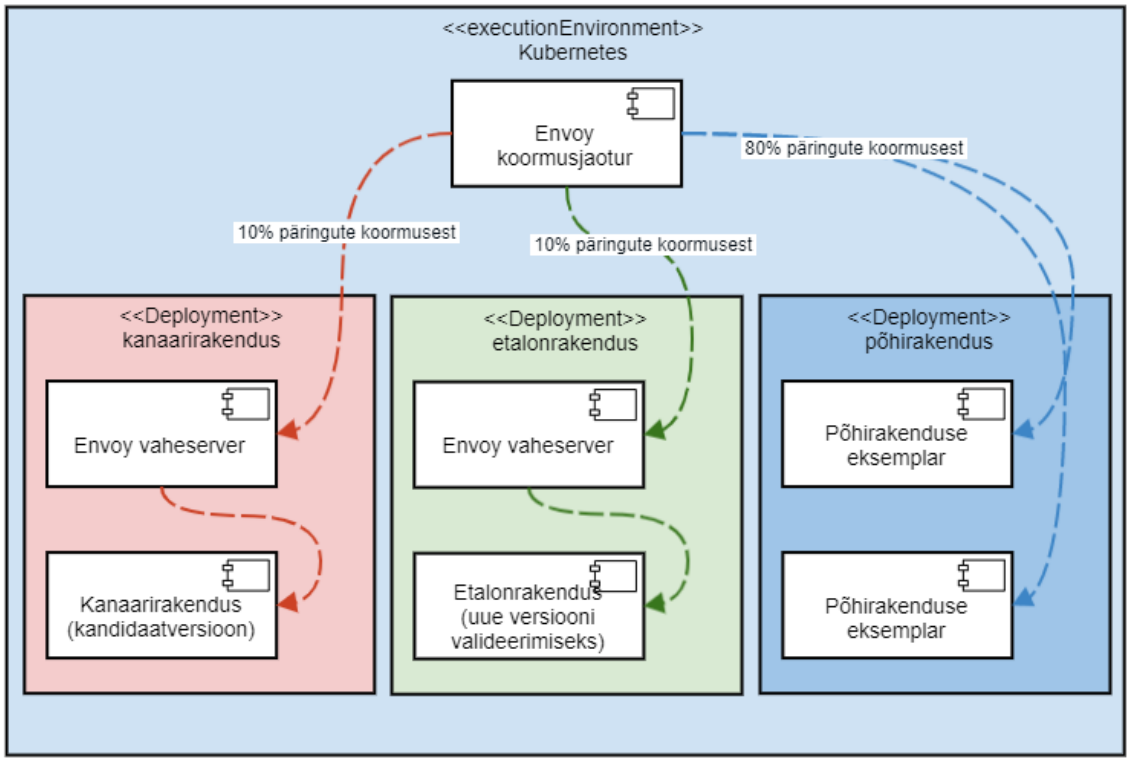
5.5 Apache pöördpuhverserveri seadistamine

Spinnakeri kasutajaliidese avalikuks tegemine vajas pöördpuhverserver seadistamist, mis võimaldab kliendil turvalise (HTTPS) ühenduse loomist serveriga. Selleks kirjeldati Apache teenuses vastav sait (*ingl.k. virtualhost*). Kuna täiendused tehti jagatud teenuses, toob autor töös välja vaid sätete kitsendatud osa. [Lisa 5]

Minio andmebaasiserveri kasutajaliidese saidi kirjeldus määratleti samuti pöördpuhverserveris. [Lisa 5]

5.6 Tarkvara kobarsüsteemi projekteerimine ja kirjeldamine

Tarkvara kobarsüsteem on ökosüsteem erinevatest konteinerrakendustest, mis seob omavahel kanaarirakenduse, etalonrakenduse, põhirakenduse, Envoy vaheserverid ja koormusjaoturi. Envoy on L7 vaheserver, mis loodi põhieesmärgiga toetada mikroteenuste arhitektuuri mustrite kasutamist andmevahetuskihis. [25] Kobarsüsteemi ülesehitust kirjeldab joonis 19.



Joonis 19. Tarkvara kobarsüsteem

Kanaarirakendus on joonisel tähistatud punasega, rohelisega on tähistatud etalonrakendus ning tumesinisega põhirakendus. Etalon- ja kanaarirakenduse ette on paigaldatud Envoy vaheserverid (*Envoy sidecar*), mis võtavad vastu koormusjaoturist tulevad päringud ning suunavad need rakendusserverisse. Vaheserverid on kasutusel klient-server teabevahetuse sisu jälgimiseks ja produtseerimiseks mõõdustikena, mis on tarbitav eraldi REST-teenusena "*/stats/prometheus*". Prometheuse monitooringusüsteem salvestab REST-teenuse kaudu mõõteandmeid nende hilisemaks analüüsiks.

Koormusjaoturi puhul on tegemist eriotstarbelise vaheserveriga, mis vahendab HTTP-liiklust konteinerrakenduste ja tarbijate vahel. Paigaldusdiagrammist on näha konteinerrakenduste seos Envoy koormusjaoturiga. Koormusjaotur suunab liiklust süsteemi sätetes kirjeldatud reeglite abil, mille tulemusel 10% kõikidest päringutest

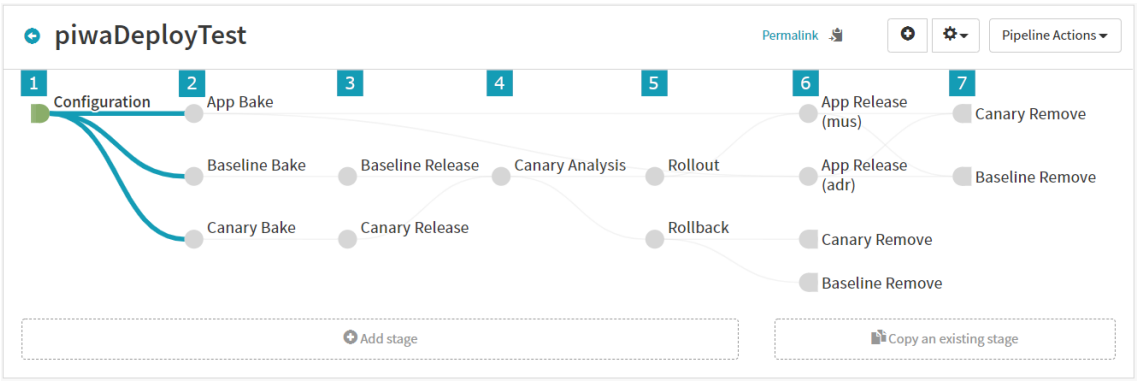
edastatakse kanaarirakendusse ja teine 10% etalonrakendusse. Ülejäänud 80% juhitakse põhirakendusse. Selle näite järgi suunatakse iga kümnes HTTP-päring vastu uut töötavat versiooni, mis võimaldab tarkvara uuendamisel maandada tõrgete esinemissagedust ning parandada muudatustest tingitud rikke korral süsteemi üleüldist tõrkekindlust. Juhul kui rikke tulemusel peaks toimuma seisak süsteemi töös, ei mõjuta see ülejäänud tarkvarasüsteemi käideldavust kuna vigane konteinerrakendus eemaldatakse koormusjaoturi poolt aktiivsete liikmete hulgast. Selleks kasutab koormusjaotur liikmete elusoleku kontrollimise mehhanismi "*heartbeat*". Lisaks pakub selline lahendus tõrke kiiremat lokaliseerimist kuna tarkvarasüsteemis tehtavad muudatused on kanaarirakenduse näol isoleeritud.

Põhirakendus on paigaldatud sõltumatu süsteemina ülejäänud konteinerrakenduste kõrvale. Põhirakendus on tavaliselt juurutatud rohkem kui ühe samaväärse asendajana. Kanaari- ja etalonrakendus on juurutatud ühekordsete eraldiseisvate eksemplaridena. Envoy koormusjaoturi ja vaheserverite seadistused on lisatud uurimistöö lisadesse. [Lisa 6]

5.7 Süsteemi juurutamise töövoo kavandamine

Praktilise töö viimases etapis loob autor Spinnakeri kasutajaliideses tarkvaraprojekti juurutamise töövoo. Töövoo kirjeldamiseks tuleb kasutajaliideses luua uus projekt (*ing.k. project*) ning loogiline rakenduse (*ing.k. "application"*) keha. Rakenduse keha sisaldab konkreetse tarkvaraprojekti juurutamiseks vajaminevaid elemente nagu töövoo kirjeldus, andmeanalüüsi kirjeldus, tarkvarasüsteemi mõõdustikud jmt. Projekt seevastu sisaldab infot tarkvarasüsteemi käideldavuse, paigaldamise asukoha ja töövoode staatuste kohta.

Töövoog on üksteisest sõltuvate unikaalsetest töökäskudest moodustuv jada. Ka töövood võivad olla teineteisest sõltuvad. Näiteks käivitatakse tarkvaraprojekti juurutamise töövoog testkeskkonda peale seda kui juurutamise töövoog arenduskeskkonnas on oma töö lõpetanud. Töövood ja töökäsud võivad teineteise suhtes asetseda rööbiti või jadamisi. Vahel tekib vajadus käivitada töökäske paralleelselt, näiteks etalon- ja kanaarirakenduse paigaldusfailide samaaegseks ettevalmistamiseks. Selleks kirjeldame töökäsud, mis paigutuvad teineteise suhtes rööbiti. Autori poolt loodud töövoogu demonstreerib joonis 20.



Joonis 20. Süsteemi juurutamise töövoog

Joonise järgi jaguneb juurutamise töövoog seitsmeks etapiks:

1) Töövoo esimeses etapis "*Configuration*" on kirjeldatud päästik, mis vallandab kõne all oleva töövoo käivitamise. Päästik kontrollib sellele eelneva töövoo (juurutamine arenduskeskkonda) lõpetamist. Arenduskeskkonna paigaldamise töövoos on samuti päästik, ent see kontrollib tarkvaraprojekti ehitamise töövoo lõppemist Jenkins CI platvormil.
2) Töökäskude "*App Bake*", "*Baseline Bake*" ja "*Canary Bake*" puhul valmistatakse ette konteinerrakenduste juurutamiseks vajalikud paigaldusfailid. Selleks laetakse koodivaramust alla igat rakenduse töökeskkonda ehk konteinerit kirjeldav *Helm chart*. *Helm chart* on arhiivifail, mis sisaldab seadeid konteinerrakenduse paigaldamiseks Kubernetese pilvetaristule. [26]
3) Kolmandas etapis paigaldatakse kanaari- ja etalonrakendus koos Envoy vaheserveritega Kubernetese pilvetaristule.
4) Neljandas etapis käivitatakse kanaarianalüüs, mis võrdleb kanaari- ja etalonrakenduse versioonis teikkinud vigade esinemissagedust. Tulemused esitatakse eraldi graafikuna Spinnakeri kasutajaliideses.
5) Viiendas etapis toimub kanaarianalüüsi tagasisidestamine. Juhul kui eelnev etapp lõppes edukalt, algatatakse põhirakenduse juurutamine kandidaatversioonile (*Rollout*). Negatiivse tulemuse korral eemaldatakse kobarast kanaari- ja etalonrakendus ehk kandidaatversioon võetakse tagasi (*rollback*).
6) Töövoo kuues järk kirjeldab põhirakenduse uuendamist kandidaatversioonile. Versiooni uuendamine toimib mõlemas Kubernetese kobaras. ("*App release (mus)*" ja "*App release (adr)*"). Seejärel eemaldatakse pilvetaristult kanaari- ja etalonrakendus (vastavalt "*Canary Remove*" ja "*Baseline Remove*").
7) Töövoo viimases ehk seitsmendas etapis taastatakse protsessi eelne olukord s.t tarkvara kobarast eemaldatakse kanaari- ja etalonrakendus.

5.8 Mõõdustikud

Tarkvarasüsteemi valideerimisel peale versiooni uuendamist tuleb lähtuda süsteemi juurutamise edukusest. Selleks peavad juurutamise järgsed tulemused olema mõõdetavad. Tulemuste hindamiseks saab kasutada erinevaid mõõdustikke, mis kirjeldavad süsteemi toimeprotsesse. Toimeprotsesside aktsepteeritavuse kontrollimiseks tuleks kasutada lävendeid, mille saavutamisel loetakse juurutamine edukaks või ebaõnnestunuks.

Tarkvaraprojekti juurutamise lähtekriteeriumid ja valideerimise viisid võivad olla erinevad. Kanaarijuurutuse kavandamisel ja tarkvarasüsteemi kvaliteedinäitajate kontrollimiseks kasutas autor kanaarianalüüsi meetodit. Kanaarianalüüs on vaid üks meede tarkvarasüsteemi toimimise kontrollimiseks. Teisteks meetoditeks on näiteks koormustestimine, teenuste funktsionaalne testimine, turvatestimine, läbistustestimine jmt. Lisaks kanaarianalüüsile on usaldusväärsema tulemuse saamiseks soovitatav kasutada eelnimetatud meetodite kombineeritud vormi.

Kanaarianalüüs jaguneb kuueks etapiks:

1. Analüüsi esimene etapp tegeleb kanaari- ja etalonrakenduse mõõteandmete kogumisega. Mõõteandmed koos märgisandmetega salvestatakse eraldi andmebaasi. Märgisandmete põhjal identifitseeritakse mõõdustike päritolu. [27]
2. Valideerimise etapis veendutakse analüüsiks vajalike mõõteandmete kättesaadavuses ja konsistentsuses. Juhul kui mõõdustik ei tagasta andmeid, asendatakse selle sisu väärtusega *NODATA* ehk puudulik. [27]
3. Andmete ettevalmistamise faas vastutab selle eest, et erinevate süsteemide identsed mõõdustikud oleksid omavahel võrreldavad. Selleks asendatakse näiteks puuduvad väärtused nulliga või eemaldatakse mõõdustik nimekirjast. [27]
4. Selles etapis klassifitseeritakse mõõdustikud, mis kajastuvad etalon- ja kanaarirakenduse mõõteandmete hälbes. Mõõdustike klassifitseerimisel lähtutakse kolmest tasemest: madal, keskmine ja kõrge. Madala klassifikatsiooni korral on võrreldavate andmete hälve väike. Kõrge hälbe korral erineb kanaarirakenduse mõõdustiku tulemus oluliselt etalonrakenduse omast. [27]
5. Pärast mõõdustike klassifitseerimist arvutatakse tulemus, mis kajastub analüüsi järeldusena kanaarirakenduse ja etalonrakenduse mõõdustike erinevuses. Iga mõõdustiku juures arvutatakse tulemus, mille kokku liitmisel esitatakse lõppresultaadina võimalikud juurutamise stsenaarium s.o kas jätkata versiooni uuendamisega põhirakendusse või pöörata kanaarirakenduse versioon tagasi. Näiteks kui 9 mõõdustikku 10st klassifitseeritakse kategooriasse "madal", on lõplikuks punktisummaks 90, mis tähistab tavaliselt madalat negatiivset hälvet kahe süsteemi vahel. [27]

6. Analüüsi viimases etapis esitatakse mõõtetulemused ja mõõdustike punktsisumma graafikuna. [27]

Autor kirjeldas kanaarianalüüsi jaoks mõned tüüpilised mõõdustikud, mida produtseerivad kanaari- ja etalonrakenduse ette paigaldatud Envoy vaheserverid. Mõõdustikud võib liigitada kaheks: a) HTTP-vastuseid kirjeldavad väärtused ja b) serveri käideldavusega seotud väärtused. Kõik mõõdustikud on esitatud tabelis 4.

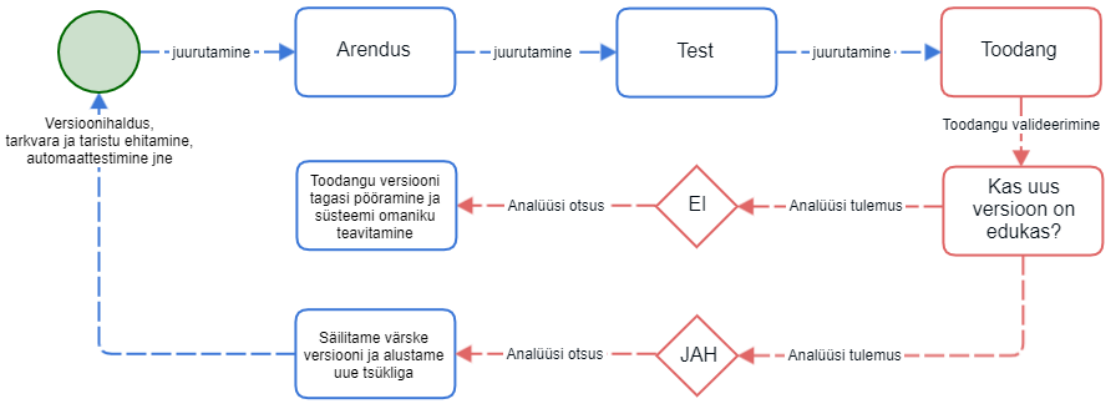
Tabel 4. Mõõdustike ülevaade

Grupp	**Mõõdustik**	**Kirjeldus**
HTTP5XX	httpRequestError500	Mõõdustik serveri HTTP-vastuste (koodiga 500) esinemissageduse mõõtmiseks
HTTP5XX	httpRequestError501	Mõõdustik serveri HTTP-vastuste (koodiga 501) esinemissageduse mõõtmiseks
HTTP5XX	httpRequestError502	Mõõdustik serveri HTTP-vastuste (koodiga 502) esinemissageduse mõõtmiseks
HTTP5XX	httpRequestError503	Mõõdustik serveri HTTP-vastuste (koodiga 503) esinemissageduse mõõtmiseks.
AVAILABILITY	connectFail	Mõõdustik ebaõnnestunud ühenduste esinemissageduse mõõtmiseks.
AVAILABILITY	connectTimeout	Mõõdustik aegunud ühenduste esinemissageduse mõõtmiseks.

6. Tulemused

Tarkvara juurutamise protsess peab olema sujuv, andma tõrgete korral kohest tagasisidet ning premeerima meeskonda hästi läbi mõeldud arenduste korral. Kuid juurutamine ei tähenda ainult inkrementaalset versiooninumbri kasvamist. Hoolitseda tuleb ka selle eest kui kiiresti on võimalik nurjunud versiooni tagasi pöörata ja suunata uuesti arendusse. Selleks kavandas autor kanaarijuurutuse töövoo.

Joonisel 21 on punasega tähistatud protsessi osa, mida tarkvaraarendusmeeskonnad Telias sooritavad käsitööna. Käsitöö alla kuulub toodangu järgne versiooni valideerimine ja otsustusprotsessi vastuvõtmine s.o kas suunata nurjunud versioon tagasi arendusse või säilitada uus versioon tootmiskeskkonnas. Töö tulemusel automatiseeriti punasega tähistatud tööprotsessi etapid.



Joonis 21. Automatiseeritud tööprotsessi ülevaate

Automatiseerimise tulemusi kontrolliti eksperimendiga. Enne töövoo kasutuselevõttu koguti tarkvaraarendusmeeskondadelt tagasisidet tehniliste kompetentside ja automatiseerimise hetkeolukorra kohta. Hindamise aluseks enne uue lahenduse juurutamist oli vajadus veenduda organisatsiooni valmisolekus ja pädevuste küpsusestasemes.

Küpsustaseme analüüs näitas, et Telia Eesti tarkvaraarendusmeeskonnad omavad tarkvara juurutamise automatiseerimiseks soovituslikku taset. Kõikide hindamisele võetud kategooriate keskmiseks tulemuseks kujunes kolm, mis on madalaim küpsustase tarneprotsesside täielikuks automatiseerimiseks. Tase kolm seab konkreetsed nõuded

tarkvara tarneahela erinevate etappide automatiseerimise mahule ning tehniliste kriteeriumide täitmisele.

Vastustest selgus, et levinuma juurutusmeetodina kasutatakse astmelist juurutusmeetodit, mis viitab hajutatud infosüsteemide arhitektuuri ja disaini mustrite kasutamisele tarkvara arenduses, eesmärgiga ehitada vastupidavamaid, efektiivsemaid ja dünaamilisemaid teenuseid. Teisest küljest varjutab meeskondade üldist taset monoliitsüsteemide osakaal tarkvaraarendusprojektides. Sellised süsteemid kuuluvad tihti korralisse versioonivahetusprotsessi, mille väljalasked on reguleeritud ning plaanilised.

Küsitluse tagasiside põhjal vestles autor erinevate meeskondadega, et leida sobiv tarkvaraprojekt eksperimendis osalemiseks. Tarkvaraprojekti valikule eelnes lahenduse põhjalik tutvustus ja tehniliste nõuete selgitus. Vestluste käigus kujunesid põhilisteks osalemise kriteeriumideks tarkvarasüsteemi aktiivne arendamine, äriteenuse madal kriitilisusklass, konteinertehnoloogiate kasutamine, hajusarhitektuuri järgimine ning teenuse aktiivne tarbivus. Kuna olemasolevate tarkvarasüsteemide juurutamise töövoo ümberkujundamine ei olnud teenuste käideldavuse ja sujuva arendusprotsessi seisukohalt otstarbekas, viidi eksperiment läbi äsja alustatud tarkvaraarendusprojektis. Täpsemalt võeti juurutamise töövoog ning kobarsüsteem kasutusele projekti testkeskkonnas kuna toodang ei olnud selleks hetkeks veel valmis. Tarkvarasüsteemi nimeks sai PIWA, mis vastutab klientide tooteandmete tiražeerimise ja muudatuste eest erinevates andmeallikates. Teenuse kaudu hallatakse ligi 400 000 kliendi tooteandmeid. Tarkvarasüsteem on aktiivses arendamises, sellel on üks põhitarbija ning testkeskkond kuulub madalasse kriitilisusklassi. Eksperiment viidi läbi perioodil 05 - 26 aprill 2021.

6.1 Automatiseerimise töövahendid

Eksperimendis ja tööprotsessi automatiseerimiseks võeti kasutusele CD platvormiteenus Spinnaker. Spinnakeris kavandati tarkvara väljalaske protsessi kirjeldav töövoog, mis sisaldab erinevaid juurutamiseks vajalike ülesandeid (vt peatükk 5, alapeatükk 5.7). Selleks, et tarkvara tarneahela erinevad etapid moodustaksid terviku ning oleksid protsessiliselt juhitud, liidestati Spinnaker platvormiteenusega Jenkins. Jenkinsis kirjeldati töövood tarkvara ja taristu ehitamiseks, mis muuhulgas sisaldasid automaattestimise, koodi kvaliteedi kontrollimise ning tarkvara tarnepaki ja töökeskkonna loomise ülesandeid.

Spinnakeri ülesseadmine nõuab head arusaama platvormiteenuse ülesehitusest ning erinevate funktsioonide, sealhulgas lõimivate teenuste tööpõhimõtetest. Spinnaker areneb ja täieneb pidevalt, mistõttu ei ole tootja suutnud dokumentatsiooni hoida kaasajastatuna ning mis süsteemi ülesseadmisel põhjustas alatihti seisakuid. Kompensatsioonimeetmena

kasutavad platvormiteenuse komponendid suhtlemiseks REST-liidest, mille logiinfo abil on probleemid kergesti tuvastatavad. Teisest küljest teeb süsteemi ülesehitus päringuahela jälgimise aga keeruliseks.

Spinnakeri platvormiteenus paigaldati Kubernetese orkestraatorile kahte teineteisest sõltumatusse andmekeskusesse, eesmärgiga tagada teenuse katkematu töö mistahes tõrke korral. Teenus töötab kahe aktiivse isendina, mis on võrguteenuse tasemel tehtud tarbijatele kättesaadavaks. Andmebaasiserver Minio juurutati virtuaalmasinasse, mida duplitseerib server süsteem varuandmekeskuses.

Spinnakeri paigaldamiseks kasutati süsteemi juurutamiseks ja seadistamiseks eraldiseisvat käsurearakendust Halyard. Halyard on platvormiteenuse juhtprogramm, mille töökeskkonna loomiseks kasutati konteinertehnoloogiat Docker. Põhjus konteinerdatud töökeskkonna kasutamiseks tulenes tootja poolt ette määratud operatsioonisüsteemidest, mida virtuaalmasina kujul ettevõtte IT-taristus ei pakuta. Lisaks pakub sellisel kujul teenuse töökeskkonna ehitamine võimaluse selle paigaldamiseks tulevikus mistahes IT-taristule või virtuaalmasinasse. Platvormiteenuse aga ka tugiteenuste seadistamisel ja juurutamisel järgiti praktikaid, mis lubaks keskkonna ehitamist reprodutseerival viisil ja kiiresti.

6.2 Kanaarijuurutuse eksperiment

Eksperimendi esimene faas keskendus kobarsüsteemi erinevate komponentide häälestuse kontrollimisele, eesmärgiga veenduda tehnilise lahenduse toimimises. Kontrollimist vajasid peamiselt koormusjaoturi ja vaheserverite seaded, mida testiti süsteemi erinevatel koormustel ja olukordades. Lisaks kutsuti testimise käigus tahtlikult esile tõrked komponentide töös ning jälgiti süsteemi seisundit selle taastumisel. Testimise käigus avastati mitmeid kriitilise iseloomuga probleeme. Näiteks mõjutas süsteemi stabiilsust etalon- ja kanaarirakenduse eemaldamine kobarast, mille tulemusel jätkas koormusjaotur päringute edastamist mitteaktiivsetele liikmetele. Samuti hakkasid üksikud päringud süsteemi koormuse kasvades ebaõnnestuma, mille tingis koormusjaoturi ebapiisav otsustuskiirus taustsüsteemide eemaldamisel teenuse aktiivsete liikmete hulgast. Mõlemal juhul oli lahenduseks teenuste häälestuse parendamine.

Teises faasis veenduti kanaarianalüüsiks vajaminevate mõõdustike ja nende tehnilise kirjelduse sobivuses. Kasutatud mõõdustike ülevaade on esitatud tabelis 4. Mõõteandmete pärimiseks Prometheuse monitooringusüsteemist kasutatakse päringukeelt PromQL. Mõõdustike ja nende andmete kättesaamiseks kirjeldati PromQL's sobivad päringulaused, mis esitati Spinnakeris kanaarianalüüsi sätete osana. Seejärel kontrolliti analüüsiprotsessi korduval käitamisel etalon- ja kanaarirakenduse vigade mõõtetulemusi. Esialgu kasutati

mõõteandmete esile kutsumiseks testpäringuid, mis konstrueeriti defektsena, eesmärgiga tekitada süsteemis meelega veaolukord. Testimise tagajärjel täiendati päringulauseid sobilike funktsioonidega, mille abil hõrendati mõõteandmete hulka etteantud ajaperioodil. Hõrendamine oli vajalik selleks, et analüüsi perioodi pikkusele vastaks igale ajaühikule N-arv mõõteandmeid.

Eksperimendi sooritamise faasis käitati kogu tarneahel tarkvara ehitamisest kuni testkeskkonna juurutamiseni, eesmärgiga veenduda kogu protsessi tõrgeteta töös. Kuna PIWA tarkvarasüsteem oli eksperimendi hetkeks arenduse algfaasis, ei olnud süsteemil veel aktiivseid tarbijaid. Seepärast pidi töö autor asuma imiteerija rolli, kes päringuid sooritas. Tarneahel käitati kümnel korral. Seitsmel korral kasutati kanaarianalüüsis mõõdustikku *httpRequestError500* ning kolmel korral mõõdustikku *connectFail* (vt Tabel 4). Esimese puhul viidi süsteem neljal korral meelega veaolukorda. Testkeskkonna juurutamise järgsel valideerimisel pöörati tarkvara versioon kanaarimeetodil tagasi neljal korral. Kolmel korral toimus juurutamine edukalt. Kanaarianalüüsi pikkuseks määrati üks tund, mis sisaldas kuut andmete kogumise ja võrdlemise kontrolletappi. Teise mõõdutiku puhul lühendati kanaarianalüüsi pikkust kümnele minutile ning Envoy vaheserveris muudeti kanaarirakenduse pordi number tahtlikult valeks, mille tulemusel ei saanud vaheserver rakendusserveriga ühendust. Seejärel kontrolliti mõõdusiku mõju tarkvarasüsteemi juurutamisele. Kolmel korral määrati kanaarijuurutus ebaõnnestunuks kuna analüüsi etapis arvutatud lõpptulemus ei vastanud õnnestumise lävendile. Õnnestumise lävendile ei vastanud mitte ükski töötsükkel kuna iga mõõdustiku juures tehti seadistus, mis sundis mistahes vigade korral protsessi ebaõnnestuma. Seesugune otsus oli vajalik kanaarianalüüsi ja mõõdustike paremaks testimiseks.

Kuna serveri poolsete vigade ja teenuste kättesaadavusega seotud mõõdustikud olid protsessi toimimise seisukohalt piisavad, ei kasutatud eksperimendis ülejäänud alapeatükis 5.8 kirjeldatud mõõdustikke kuna need sisaldasid testitud mõõdustikega sarnaseid mõõteandmeid. Küll aga on need vajalikud kriitiliste intsidentide tuvastamiseks versioonivahetuse hetkel toodangus.

Pärast töövoo automatiseerimist ei nõua ükski tegevus käsitsi sekkumist. Kõik tööprotsessi etapid on automatiseeritud vastavalt tööprotsessi automatiseerimise skeemile (Joonis 21). Automatiseerimise tulemusena on loodud põhi, mille abil saavad tarkvaraarendusmeeskonnad keskenduda põhiülesandele, milleks on tarkvara pidev tootmine ning ärieesmärkide teostamine, näiteks digitaalse kliendikogemuse parandamine. Toodangu valideerimine, millega varem tegeles IT-haldur või arendaja, toimub peale uue versiooni väljalaset automaatselt. Samuti tehakse võimalike tõrgete korral otsus kandidaatversiooni tagasi pööramiseks, mis kiirendab probleemidele reageerimise aega ning maandab süsteemi käideldavusega seotud riske.

6.3 Võimalikud edasiarendused

Diplomitöös juurutatud CD platvormiteenus ja kanaarijuurutuse töövoog on võetud praktikas kasutusele ühes tarkvaraarendusprojektis. Diplomitöö avaldamise hetkeks kirjeldati juurutamise töövoog testkeskkonnas, kus toimub protsessi järk-järguline optimeerimine ja parendamine. Seejärel planeeritakse juurutusmeetodi kasutuselevõttu toodangus. Lisaks on prototüübi ja tööprotsesside automatiseerimisega loodud põhi tulevaste ja hetkel käigus olevate tarkvaraarendusprojektide tarneprotsessi täielikuks automatiseerimiseks.

Praktilise ja sisulise töö jätkuteemad:

1. Kanaarianalüüsiks ei piisa vaid vaheserverite esitlusandmetest. Otsustuskriteeriumide automatiseerimise seisukohalt tuleks süsteemi hetkeolukorra hindamiseks kasutada ka äriloogilisi tööprotsesse kirjeldavaid mõõdustikke. Kuna iga tarkvaraprojekt on unikaalne, tuleks äriloogilised mõõdustikud kirjeldada koos tarkvaraarendusmeeskonnaga. Seejärel tasub uurimist, millised oleksid mõõdustikud, mida saaks taaskasutada mistahes tarkvaraarendusprojektis.
2. Versiooni väljalaskel etalon- ja kanaarirakendusse võiks kanaarianalüüsi täiendusena uurida erinevaid tarkvarasüsteemi valideerimise viise. Nendeks meetoditeks võivad olla näiteks koormustestimine, teenuste funktsionaalne testimine, turvatestimine või läbistustestimine. Kombineeritud analüüs annab hetkeolukorrast kõige parema ülevaate.
3. Praktilises osas kirjeldatud juurutamise töövoog ja selle seaded on näidisprojekti spetsiifilised ega luba konfiguratsiooni täielikku taaskasutamist mõnes teises projektis. Konfiguratsiooni taaskasutamiseks tuleks leida erinevate tarkvarasüsteemide ühisosa ning kirjeldada reeglid ühtlustatud mallide loomiseks.
4. Prototüübi edukal juurutamisel on autori huvi tutvustada lahendust ülejäänud organisatsioonis ning teha lahendus kättesaadavaks võimalikult paljudele tarkvaraarendusmeeskondadele. Prototüübi juurutamist kitsendab asjaolu, et mitmete tarkvaraarendusprojektide tarneahelad on kirjeldatud alternatiivsetes CI/CD platvormiteenustes (näiteks Bamboo), mida Spinnaker ei toeta. Selleks, et võimendada saadavat kasu kogu tehnoloogiaüksuses, tuleb koostada plaan idee tutvustamiseks erinevates juhtrühmades. Plaan peab sisaldama tulemusi, kuidas tarkvara tarneahel on võrreldes varasemaga paremaks muutunud ja milline on loodav väärtus.
5. Anomaaliate ja probleemide tuvastamine lineaarse protsessimudeli järgi on pikkade analüüsitsüklite juures, kus olukorrast ülevaate saamiseks on n-ö "aega üle", piisav, kuid efektiivsuse seisukohalt kasin meetod. Selliseks meetodiks võib pidada kanaarianalüüsi. Juhul kui eesmärgiks on analüüsi faasi optimeerimine, mis lubaks

ennustada ka võimalike tõrkeid ja anomaaliaid süsteemi töös, tuleks mõelda masinõppe kasutamisele.

7. Kokkuvõte

Diplomitöö eesmärgiks oli juurutada automatiseerimisvahend, mis visualiseeriks ja automatiseeriks konteinerrakenduste paigaldamist arendus-, test- ja tootmiskeskkonda. Konteinerrakenduste paigaldamiseks kavandati kanaarijuurutuse töövoog ning tarkvarasüsteemi kirjeldav komponentmudel kandidaatversiooni väljalaseteks.

Tarkvara tarneprotsesside automatiseerimise hetkeolukorra hindamiseks viis autor organisatsioonis läbi tarkvaraarendusmeeskondade küpsustaseme uuringu. Uuring näitas, et meeskonnad omavad nii tehnilist kui kompetensialast valmisolekut tööprotsesside automatiseerimiseks. Seejärel uuris autor erinevaid juurutusmeetodeid ning võrdles kolme vabavaralist ja majasisest juurutusvahendit, mille tulemusel võeti üks neist kasutusele ning tehti tarkvaraarendusmeeskondadele kättesaadavaks. Sobiva tarkvaraprojekti juurutamise automatiseerimiseks kirjeldati tingimused eksperimendi planeerimisel.

Eksperimendis testiti kanaarijuurutuse töövood praktikas, eesmärgiga kontrollida diplomitöö lähtetingimuste vastavust lõpptulemustega. Katse viidi läbi testkeskkonnas, kuna tööprotsessi ümberkujundamine seab ohtu teenuste käideldavuse ja arendusprotsessi tõrgeteta toimimise, mida töö käigus püüti vältida. Lõppresultaadina algatati juurutusprotsess seitsmel korral, mis järgisid ette antud stsenaariumit. Nendest neli lõpetasid tingimusliku veaga ning kolm olid edukad. Tööprotsessi automatiseerimisel ei nõua ükski tegevus käsitsi sekkumist, mis lubab meeskondadel keskenduda teiste tööülesannete täitmisele.

Kasutatud kirjandus

[1] S. Vene. DevOps juurutamine suurettevõttes: magistritöö. Tallinna Tehnikaülikool, Tallinn, 2017.

[2] Medium. *What is Jenkins X and how is it Different from Jenkins?*. [WWW] https://medium.com/edureka/jenkins-x-d87c0271af57 (25.03.2021).

[3] Pittet, S. *Continuous Deployment*. [WWW] https://www.atlassian.com/continuous-delivery/continuous-deployment (24.03.2021).

[4] Grant, C. *Deployment Strategies & Release Best Practices*. [WWW] https://medium.com/cgrant/deployment-strategies-release-best-practices-6e557c3f39b4 (18.04.2021).

[5] Shakury, I. *Deployment Strategies Defined*. [WWW] https://blog.itaysk.com/2017/11/20/deployment-strategies-defined (18.04.2021).

[6] Smith, A. *Rolling Deployment: What This Is and How it De-Risks Software Deploys*. [WWW] https://www.cloudbees.com/blog/rolling-deployment/ (18.04.2021).

[7] Fowler, M. *Blue Green Deployment*. [WWW] https://martinfowler.com/bliki/BlueGreenDeployment.html (18.04.2021).

[8] Tremel, E. *Six Strategies for Application Deployment*. [WWW] https://thenewstack.io/deployment-strategies/ (18.04.2021).

[9] Atlassian. *Overview of all Atlassian products*. [WWW] https://confluence.atlassian.com/confeval/other-atlassian-evaluator-resources/overview-of-all-atlassian-products (25.03.2021).

[10] Chaimungkalanont, M. *Bamboo 1.0 Released*. [WWW] https://www.atlassian.com/blog/archives/bamboo_10_released (25.03.2021).

[11] Cloudbees. *What is Jenkins?*. [WWW] https://www.cloudbees.com/jenkins/what-is-jenkins (25.03.2021).

[12] Sanson, C. *Spinnaker 1.0: a continuous delivery platform for cloud*. [WWW] https://cloud.google.com/blog/products/gcp/spinnaker-10-continuous-delivery (25.03.2021).

[13] Mängel, T. Pilvandmetöötlus Euroopa liidus Eesti näitel: bakalaureusetöö. Tallinna Tehnikaülikool, Tallinn, 2015.

[14] Techterms. *Webhook*. [WWW] https://techterms.com/definition/webhook (25.03.2021).

[15] Nuum, T. *Mis on API?*. [WWW] https://blog.twn.ee/et/Mis_on_API (25.03.2021).

[16] Akit. *Andmekaitse ja infoturbe leksikon*. [WWW] https://akit.cyber.ee/term/780 (25.03.2021).

[17] Atlassian. *Explore apps for Bamboo*. [WWW] https://marketplace.atlassian.com/addons/app/bamboo (25.03.2021).

[18] J. M. Radstaak. Developing a DevOps maturity model: A validated model to evaluate the maturity of DevOps in organizations: magistritöö. University of Twente, Enschede, 2019.

[19] Spinnaker. *Choose your Environment*. [WWW] https://spinnaker.io/setup/install/environment/ (03.04.2021).

[20] Spinnaker. *Install Halyard*. [WWW] https://spinnaker.io/setup/install/halyard/ (03.04.2021).

[21] Spinnaker. *Architecture*. [WWW] https://spinnaker.io/reference/architecture/ (03.04.2021).

[22] Spinnaker. *Install and Configure Spinnaker*. [WWW] https://spinnaker.io/setup/install/halyard/ (03.04.2021).

[23] Spinnaker. *About External Storage*. [WWW] https://spinnaker.io/setup/install/storage/ (03.04.2021).

[24] EUCIP. *Objektipõhine loogiline mudel*. [WWW] https://eopearhiiv.edu.ee/e-kursused/eucip/arendus/234_objektiphine_loogiline_mudel.html (03.04.2021).

[25] Puri, M. *L4 vs L7 Load Balancing*. [WWW] https://levelup.gitconnected.com/l4-vs-l7-load-balancing-d2012e271f56 (05.04.2021).

[26] Bsoul, N. *What is Helm in Kubernetes?*. [WWW] https://jfrog.com/knowledge-base/what-is-helm-in-kubernetes/ (07.04.2021).

[27] Graff, M., Sanden, C. *Automated Canary Analysis at Netflix with Kayenta*. [WWW] https://netflixtechblog.com/automated-canary-analysis-at-netflix-with-kayenta-3260bc7acc69 (07.04.2021).

Lisad

Lisa 1. Lihtlitsents

Mina, Silver Viljaste

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "KONTEINERRAKENDUSTE AUTOMAATNE JUURUTAMINE AEGRIDA ANDMETE ANALÜÜSIL TELIA EESTI AS NÄITEL" , mille juhendaja on Edmund Laugasson,
   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.

17.05.2021

Lisa 2. Halyard konteinerrakenduse Dockerfile

```
FROM ubuntu:16.04

LABEL description="Halyard"

USER root

ENV KUBECTL_VERSION="1.15.10"
ENV PACKER_VERSION="1.7.0"
ENV SPINNAKER_USER="spinnaker"

ENV TZ=Europe/Tallinn

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ >
/etc/timezone

RUN apt-get update && apt-get upgrade -y && apt-get install -y \
    software-properties-common \
    && yes | add-apt-repository ppa:openjdk-r/ppa

RUN apt-get update && apt-get install -y \
    curl \
    sudo \
    vim \
    apt-transport-https \
    wget \
    unzip \
    openjdk-11-jdk \
    && rm -rf /var/lib/apt/lists/*

RUN USERNAME="${SPINNAKER_USER}"; \
    NAME="${USERNAME}"; \
    GROUPS="sudo"; \
    useradd -c "${NAME}" -s /bin/bash -G "${GROUPS}" -d
    /home/${USERNAME} -m "${USERNAME}" && \
    echo "${USERNAME} ALL=(ALL:ALL) NOPASSWD:ALL" >>
    /etc/sudoers && \
    mkdir /home/${USERNAME}/.kube
```

```
RUN curl -f -LO --retry 3 --retry-delay 3
https://storage.googleapis.com/kubernetes-release/release/
v${KUBECTL_VERSION}/bin/linux/amd64/kubectl && \
    chmod +x ./kubectl && \
    mv ./kubectl /usr/local/bin/kubectl

RUN curl -f -LO --retry 3 --retry-delay 3
https://releases.hashicorp.com/packer/${PACKER_VERSION}/
packer_${PACKER_VERSION}_linux_amd64.zip && \
    unzip ./packer_${PACKER_VERSION}_linux_amd64.zip && \
    chmod +x ./packer && \
    mv ./packer /usr/local/bin/packer && \
    rm -f ./packer_${PACKER_VERSION}_linux_amd64.zip

RUN curl -O --retry 3 --retry-delay 3
https://raw.githubusercontent.com/spinnaker/halyard/master/
install/debian/InstallHalyard.sh && \
    mv ./InstallHalyard.sh /home/${SPINNAKER_USER}/

ADD ./config/k8s-profile.txt /home/${SPINNAKER_USER}/.kube/config
ADD ./config/keystore.jks /home/${SPINNAKER_USER}/keystore.jks

RUN chown ${SPINNAKER_USER}:${SPINNAKER_USER}
/home/${SPINNAKER_USER}/ -R

CMD echo "spinnaker" |
bash /home/${SPINNAKER_USER}/InstallHalyard.sh; while true; do
sleep 60; done;

EXPOSE 8084 9000
```

Lisa 3. Minio andmebaasiserveri Dockerfile

```
FROM minio/minio

LABEL description="Minio"

USER root

ARG spinnaker_user
ARG spinnaker_password

ENV MINIO_ROOT_USER="${spinnaker_user}"
ENV MINIO_ROOT_PASSWORD="${spinnaker_password}"
ENV TZ=Europe/Tallinn

RUN ln -snf /usr/share/zoneinfo/$TZ /etc/localtime && echo $TZ >
/etc/timezone

CMD ["server", "--address", ":9001", "/data"]

EXPOSE 9001
```

Lisa 4. Spinnakeri paigaldamise ja häälestamise käsud

```
# Halyardi konteinerrakendusse sisselogimine kasutajaga
"spinnaker".
docker exec -u spinnaker -it halyard bash

# Spinnakeri versiooni määramine.
hal config version edit --version 1.25.4

# Spinnakeri juurutusviisi ja asukoha seadistamine.
hal config deploy edit --type distributed --account-name
k8s-spinnaker-account
hal config deploy edit --location spinnaker-namespace
hal config provider kubernetes enable
hal config provider kubernetes account add k8s-spinnaker-account
--context k8s-cluster

# Pilvetaristuga ühendamiseks vajalike seadete kirjeldamine.
hal config provider kubernetes account add k8s-apps-account-dev
--context k8s-cluster --namespaces dev-namespace
hal config provider kubernetes account add k8s-apps-account-test
--context k8s-cluster --namespaces test-namespace

# LDAP autentimisviisi seadete kirjeldamine.
echo "salasöna" | hal config security authn ldap edit
--url="ldaps://kataloogiteenus.estpak.ee:636/DC=estpak,DC=ee"
--manager-dn="uid=spinnaker,ou=ldap,ou=kasutajad"
--group-search-base="ou=rollid,ou=grupid"
--user-search-filter="(sAMAccountName={0})" --manager-password
hal config security authn ldap enable
hal config security authz enable
echo "salasöna" | hal config security authz ldap edit
--url="ldaps://kataloogiteenus.estpak.ee:636/DC=estpak,DC=ee"
--manager-dn="uid=spinnaker,ou=ldap,ou=kasutajad"
--group-search-base="ou=rollid,ou=grupid"
--user-search-filter="(sAMAccountName={0})" --manager-password

# Andmebaasiserveri ühenduse loomiseks vajalike seadete
kirjeldamine.
hal config storage edit --type s3
```

```
echo "salasöna" | hal config storage s3 edit --endpoint
https://minio.estpak.ee:8446 --access-key-id spinnaker
--secret-access-key --path-style-access=true

# Spinnakeri haldusliidese veebiaadresside kirjeldamine
hüperlinkide loomiseks.
hal config security ui edit --override-base-url
https://spinnaker.estpak.ee
hal config security api edit --override-base-url
https://spinnaker.estpak.ee/gate

# Koodivaramu seadete kirjeldamine.
hal config artifact bitbucket enable
echo "salasöna" | hal config artifact bitbucket account add
bitbucket --username spinnaker --password

# Jenkinsiga ühendamiseks vajalike seadete kirjeldamine.
echo "salasöna" | hal config ci jenkins master add jenkins
--address https://jenkinsmaster.estpak.ee --username spinnaker
--password --trustStoreType jks --trustStore
/home/spinnaker/truststore.jks --trustStorePassword changeit

# Kanaarijuurtuse seadete kirjeldamine.
hal config canary enable
hal config canary edit --show-all-configs-enabled false
hal config canary aws enable
echo "salasöna" | hal config canary aws account add minio
--endpoint https://minio.estpak.ee:8446 --bucket
spin-523da5fc-a3b5-44da-ae16-3c288e8f2911 --access-key-id
spinnaker --secret-access-key --path-style-access=true
hal config canary aws edit --s3-enabled=true
hal config canary edit --default-storage-account minio

# Prometheuse monitooringusüsteemiga ühendamiseks vajalike
seadete kirjeldamine.
hal config canary prometheus enable
hal config canary edit --default-judge JUDGE
hal config canary edit --default-metrics-store prometheus
hal config canary prometheus account add prometheus --base-url
http://meetrikad.estpak.ee --no-validate
```

```
# Spinnakeri paigaldamine Kubernetese orkestraatorile.
hal deploy apply

# Spinnakeri sätete varundamine.
hal backup create
```

Lisa 5. Spinnakeri ja Minio saidi kirjeldus Apache pöördpuhverserveris

```
LISTEN 8444
LISTEN 8446

# Platvormiteenuse Spinnaker sait.
<VirtualHost spinnaker.estpak.ee:8444>

  ServerName spinnaker.estpak.ee

  AllowEncodedSlashes NoDecode

  RequestHeader unset X-Forwarded-For
  RequestHeader unset X-Forwarded-Host
  RequestHeader unset X-Forwarded-Server
  RequestHeader set X-Forwarded-Port "443"
  RequestHeader set X-Forwarded-Proto "https"
  Header always set Strict-Transport-Security "max-age=63072000"
  Header always set Content-Security-Policy
  "upgrade-insecure-requests"

  ErrorLog logs/spinnaker.estpak.ee_ssl_error.log
  CustomLog logs/spinnaker.estpak.ee_ssl.log combined

  SSLProxyEngine On
  SSLEngine On
  SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
  SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-
  GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-
  SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-
  POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
  SSLHonorCipherOrder off
  SSLCertificateChainFile /etc/apache2/ssl/cacert.crt
  SSLCertificateFile /etc/apache2/ssl/spinnaker_2021.crt
  SSLCertificateKeyFile /etc/apache2/ssl/spinnaker_2021.key

  ProxyRequests Off
  ProxyPreserveHost On
  RewriteEngine On
```

```
  <LocationMatch "/">
    Header set ServerHostname "spinnaker.estpak.ee"
    RequestHeader set X-Forwarded-For "%{REMOTE_ADDR}e"
    RequestHeader set Host "spinnaker.estpak.ee"

    ProxyPass
    http://spinnaker-external.spinnaker.tcs.glb.estpak.ee:9000/
    ProxyPassReverse
    http://spinnaker-external.spinnaker.tcs.glb.estpak.ee:9000/
  </LocationMatch>

  <LocationMatch "(?i)(/gate/)(.*)">
    Header set ServerHostname "spinnaker.estpak.ee"
    RequestHeader set X-Forwarded-For "%{REMOTE_ADDR}e"
    RequestHeader set Host "spinnaker.estpak.ee"

    ProxyPass
    http://spinnaker-external.spinnaker.tcs.glb.estpak.ee:8084/$2
    ProxyPassReverse
    http://spinnaker-external.spinnaker.tcs.glb.estpak.ee:8084/
  </LocationMatch>

</VirtualHost>

# Andmebaasiserveri Minio sait.
<VirtualHost minio.estpak.ee:8446>

  ServerName minio.estpak.ee

  Header always set Strict-Transport-Security "max-age=63072000"
  RequestHeader set X-Forwarded-Proto "https"
  AllowEncodedSlashes NoDecode

  ErrorLog logs/minio.estpak.ee_ssl_error.log
  CustomLog logs/minio.estpak.ee_ssl.log combined

  SSLProxyEngine On
  SSLEngine On
  SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
  SSLCipherSuite ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-
```

```
  GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-
  SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-
  POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384
  SSLHonorCipherOrder off
  SSLCertificateChainFile /etc/apache2/ssl/cacert.crt
  SSLCertificateFile /etc/apache2/ssl/minio_2021.crt
  SSLCertificateKeyFile /etc/apache2/ssl/minio_2021.key

  ProxyRequests Off
  ProxyPreserveHost On
  RewriteEngine On

  <LocationMatch "/">
    ProxyPass http://127.0.0.1:9001/
    ProxyPassReverse http://127.0.0.1:9001/
  </LocationMatch>

</VirtualHost>
```

Lisa 6. Envoy koormusjaoturi ja vaheserveri seadete kirjeldus

```
# Envoy koormusjaotur.
static_resources:
  listeners:
    - address:
        socket_address:
          protocol: TCP
          address: 0.0.0.0
          port_value: 10000
      filter_chains:
        - filters:
          - name: http
            typed_config:
              "@type": type.googleapis.com/envoy.extensions.
              filters.network.http_connection_manager.v3.
              HttpConnectionManager
              drain_timeout: 5s
              codec_type: auto
              stat_prefix: ingress_http
              route_config:
                virtual_hosts:
                - name: backend
                  domains: ["*"]
                  routes:
                    - match:
                        prefix: "/"
                      route:
                        weighted_clusters:
                          clusters:
                            - name: main
                              weight: 80
                            - name: baseline
                              weight: 10
                            - name: canary
                              weight: 10
              http_filters:
              - name: health
                typed_config:
                  "@type": type.googleapis.com/envoy.extensions.
```

```
                filters.http.health_check.v3.HealthCheck
                pass_through_mode: true
            - name: envoy.filters.http.router
              typed_config:
                "@type": type.googleapis.com/envoy.extensions.
                filters.http.router.v3.Router
            access_log:
            - name: access_log
              typed_config:
                "@type": type.googleapis.com/envoy.extensions.
                access_loggers.file.v3.FileAccessLog
                path: /dev/stdout
clusters:
- name: main
  connect_timeout: 30s
  type: STRICT_DNS
  dns_lookup_family: V4_ONLY
  lb_policy: ROUND_ROBIN
  load_assignment:
    cluster_name: main
    endpoints:
    - lb_endpoints:
      - endpoint:
          address:
            socket_address:
              address: "piwa-internal"
              port_value: "80"
- name: baseline
  connect_timeout: 30s
  type: STRICT_DNS
  dns_lookup_family: V4_ONLY
  lb_policy: ROUND_ROBIN
  health_checks:
    - timeout: 1s
      interval: 1s
      interval_jitter: 1s
      unhealthy_interval: 1s
      unhealthy_threshold: 3
      healthy_threshold: 3
      tcp_health_check: {}
  load_assignment:
```

```
      cluster_name: baseline
      endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              socket_address:
                address: "piwa-baseline-internal"
                port_value: "80"
  - name: canary
    connect_timeout: 30s
    type: STRICT_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: ROUND_ROBIN
    health_checks:
      - timeout: 1s
        interval: 1s
        interval_jitter: 1s
        unhealthy_interval: 1s
        unhealthy_threshold: 3
        healthy_threshold: 3
        tcp_health_check: {}
    load_assignment:
      cluster_name: canary
      endpoints:
        - lb_endpoints:
          - endpoint:
              address:
                socket_address:
                  address: "piwa-canary-internal"
                  port_value: "80"
admin:
  address:
    socket_address:
      protocol: TCP
      address: 0.0.0.0
      port_value: 9901
  access_log_path: /dev/stdout

# Envoy vaheserver.
static_resources:
  listeners:
```

```
  - name: static_listener
    address:
      socket_address:
        protocol: TCP
        address: 0.0.0.0
        port_value: 10000
    filter_chains:
    - filters:
      - name: envoy.http_connection_manager
        config:
          codec_type: auto
          stat_prefix: ingress_http
          route_config:
            name: local_route
            virtual_hosts:
            - name: vhost
              domains: ["*"]
              routes:
              - match:
                  prefix: "/"
                route:
                  cluster: backend_clu
          http_filters:
          - name: envoy.router
            config: {}
          access_log:
          - name: envoy.access_log
            typed_config:
              "@type": type.googleapis.com/envoy.extensions.
              access_loggers.file.v3.FileAccessLog
              path: /dev/stdout
  clusters:
  - name: backend_clu
    connect_timeout: 30s
    type: STRICT_DNS
    dns_lookup_family: V4_ONLY
    lb_policy: ROUND_ROBIN
    hosts: [
      { socket_address: { address: 127.0.0.1, port_value: 8080 }}
    ]
admin:
```

```
address:
  socket_address:
    protocol: TCP
    address: 0.0.0.0
    port_value: 9901
access_log_path: /dev/stdout
```