TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Marco Sepp 179858IADB

KOLMANDA OSAPOOLE TARKVARA PAIGALDUSE JA UUENDAMISE AUTOMATISEERIMINE UNIX-LIIKI OPERATSIOONISÜSTEEMIDEL KASUTADES INFRASTRUKTUURI KOODI FINANTSETTEVÕTTE NÄITEL

bakalaureusetöö

Juhendaja: German Mumma
Magister

Tallinn 2020

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: Marco Sepp

30.04.2020

Annotatsioon

Käesoleva lõputöö eesmärk oli leida lahendus, mis hoiaks kokku finantsettevõtte andmetöötlusplatvormi arendajate ja haldajate tööd platvormi halduselt ning tarkvarade paigaldamise ja uuendamise pealt ning vastaks ettevõtte turvapoliitikale.

Töökäigus võrreldi erinevaid lahendusi, millega on võimalik automatiseerida kolmanda osapoolte tarkvarade paigaldamise ja uuendamise protsessi ning leiti lahendus, mis sobib töös kirjeldatud finantsettevõtte andmetöötlusplatvormile. Seejärel juurutati lahendus finantsettevõtte näitel ning kirjeldati andmetöötlusplatvormi kahe tarkvara paigalduse ja uuendamise protsessi.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 33 leheküljel, 3 peatükki, 15 joonist, 1 tabelit.

Abstract

Automating third-party software deployment and upgrades on UNIX-like operating systems using infrastructure as code in an example of a financial company

The purpose of this bachelor's thesis is to find a solution to save developers and administrators time on the administration of the platform and to save time on the deployment and upgrading of software. As a part of this thesis, the author plans to find and implement a solution that would allow developers to save time on administration and other repetitive activities and allow them to perform these actions automatically.

The bachelor's thesis is divided into three parts: background information, analysis and technical implementation. The first part describes the current situation in one of the Baltic and Nordic financial companies. The author will describe possible solutions which could solve said problems.

The analysis part of the thesis compares possible software solutions and analyses whether and how these solutions would suit the company. As a result, one software is selected that would meet the criteria set by the author and the company.

In the technical part of the thesis, the author introduces the chosen solution in the example of the financial company and describes the methods of implementing it. In addition, this section also describes future possibilities that could be implemented but will not be covered in this thesis.

During the bachelor's thesis, the solution was implemented in a financial company and the process of deploying and upgrading two data processing platform software was described.

The thesis is in Estonian and contains 33 pages of text, 3 chapters, 15 figures, 1 table.

Lühendite ja mõistete sõnastik

Agent	Funktsionaalüksus, mis on võimeline sooritama haldusobjektidega haldusoperatsioone.
API	*Application programming interface* elik rakendusliides või programmiliides, mis on eri tarkvarakomponentide vahelise selgelt määratletud sidevahendite kogum.
DevOps	*Development and operations* elik kultuur, mis ühendab endas arenduse ja halduse metoodikad.
Dockeri konteiner	Lahendus, mis kasutab UNIX-liiki operatsioonisüsteemides isoleerimisfunktsioone, et tekitada niinimetatud virtuaalne konteiner, mis käitub nagu virtuaalne server, kuid ei oma isiklikku operatsioonisüsteemi. Konteinerid kasutavad selle arvuti operatsioonisüsteemi kus konteiner käivitati.
DSL	Domeeni spetsiifiline keel (ingl. *domain-specific language*).
Git	Vaba, hajutatud, versioonihaldustarkvara, mis loodi Linus Torvaldsi poolt Linuxi tuuma arendamiseks. [1]
HCL	*Hashicorp Configuration Language* elik Hashicorp konfigureerimiskeel.
Hüperviisor	Tarkvara, mis võimaldab virtuaalmasina tööd ja haldamist (ingl. *Hypervisor).*
IaC	*Infrastructure as code* elik infrastruktuur koodina.
INI	Mitteametlik standard konfiguratsioonifailide kirjeldamiseks.
IT	Infotehnoloogia
Kaugserver	Eraldiseisev server millesse on võimalik eemalt ühenduda kasutades erinevaid liidestusvahendeid (ingl. *Remote server*).
Plugin	Tarkvara laiend andmete eritüüpide käsitluseks.
Serveripark	Kõrgelt turvatud hoonete kompleks, kus hoitakse ja käivitatakse füüsilisi servereid.
SSD	*Solid state drive* elik pooljuht andmesalvestusseade.
SSH	Turvakest elik krüptograafiline võrguprotokoll turvaliseks võrguteenuste opereerimiseks turvamata võrgu kaudu. (ingl. *Secure Shell*).

UNIX-liiki operatsioonisüsteem	Operatsioonisüsteem. UNIX-liiki operatsioonisüsteem on ülesehituselt sarnane UNIX-i operatsioonisüsteemiga, kuid kuna viimane on registreeritud kaubamärk, siis nimetatakse teisi UNIX-liiki.
Veebikonks	*Webhook* on versioonihaldustarkvara moodul, mis võimaldab seadistatud tegevuste järel informeerida teisi tarkvarasid nendest tegevustest.
YAML	Andemete serialiseerimiskeel disainitud inimestele lihtsaks lugemiseks. [2]

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänapäeva infoühiskonnas on väga tähtsal kohal andmete kogumine ja töötlemine. Platvormid, mis neid andmeid hoiustavad, muutuvad iga päevaga suuremaks ja võimsamaks. Nende platvormide haldamine muutub sama kiirelt üha keerulisemaks. Veel hiljuti toimus serverite haldamine ja tarkvarade paigaldamine manuaalselt. Protsess oli aeganõudev ning infotehnoloogiasüsteeme haldavad firmad pidid otsustama, kas nõuda administraatoritelt ületundide tegemist või värvata rohkem tööjõudu. Samal ajal muutuvad karmimaks ka ettevõtetele kehtivad infotehnoloogia süsteemide turvameetmed. Suuremates ettevõtetes otsustatakse palgata rohkem tööjõudu, kuid jagada nende inimeste ülesandeid nii süsteemide arenduse kui ka halduse vahel.

Käesoleva diplomitöö eesmärgiks on leida lahendus, kuidas hoida kokku arendajate ja haldajate tööd platvormi halduselt ning tarkvarade paigaldamise ja uuendamise pealt. Töö raames plaanib autor leida ja juurutada lahenduse, mis võimaldaks arendajate aega kokku hoida halduselt ning muudelt korduvatelt tegevustel ning võimaldada neil neid tegevusi teha automatiseeritult.

Bakalaureusetöö on jagatud kolme ossa: taustainfo, analüüs ning tehniline teostus. Töö taustainfo osas kirjeldatakse hetke olukorda ühes Baltikumi ning põhjamaade finantsettevõttes, mille põhjal koostatakse lähteülesanne. Seejärel kirjeldatakse võimalikke lahendusi selle ülesande lahendamiseks.

Analüüsi osas võrreldakse võimalikke tarkvaralisi lahendusi ning analüüsitakse kas ja kuidas need lahendused sobiksid eelmainitud ettevõttele. Analüüsi tulemusena valitakse välja üks tarkvara, mis rahuldaks autori ja ettevõtte poolt püstitatud kriteeriumeid.

Töö tehnilises osas juurutab autor välja valitud lahenduse ettevõtte näitel ning kirjeldab lahenduse kasutuselevõtu võimalusi. Lisaks kirjutatakse selles osas ka tuleviku võimalustest, mida käesoleva lõputöö raames kasutusele ei võeta.

2 Taust

Selles peatükis kirjeldab autor käesoleva bakalaureusetöö tausta. Kirjeldatakse töös kasutatavaid tehnoloogiad ning termineid.

2.1 DevOps kultuur

DevOps on tarkvaraarendusmetoodikate kogum, mille eesmärk on ühendada tarkvaraarendus (*Dev*) ja tarkvaraoperatsioonid (*Ops*). Termin "DevOps" jõudis avalikkuse ette 2008. aastal Torontos Agile'i konverentsil, kus Andrew Shafer ja Patrick Debois kasutasid seda terminit nende paindliku ehk agiilse taristu teemalises esitluses. [3] [4]

DevOps ei ole standard, tehnoloogia ega protsess. Paljud DevOps fanaatikud ja pühendunud DevOps inimesed nimetavad seda pigem kultuuriks või liikumiseks. [5]

Gartner Glossary veebilehekülg pakub DevOpsile välja järgneva selgituse:

„DevOps esindab muutust IT-kultuuris (elik infotehnoloogia kultuuris), keskendudes kiirele IT-teenuste pakkumisele, kasutades süsteemikeskse lähenemisviisi kontekstis paindlikke ja lahkeid tavasid. DevOps rõhutab inimesi (ja kultuuri) ning selle eesmärk on parandada operatsioonide ja arendusmeeskondade vahelist koostööd. DevOps-i rakendused kasutavad tehnoloogiat - eriti automaatikatööriistu, mis saavad kasutada olelustsükli perspektiivist üha programmeeritavamat ja dünaamilisemat infrastruktuuri.“ [6]

2.2 Infrastruktuur koodina

Infrastruktuur koodina (ingl. *Infrastructure as code.* lühend *IaC*) on protsess, mille käigus hallatakse infrastruktuuri ja paigaldatakse tarkvara kasutades masinloetavaid faile, mitte ei tehta seda manuaalselt ja käsitsi. See arenes välja pärast seda, kui tekkisid jagatud arvutusressursi pakkujad nagu Amazon AWS ja Microsoft Azure. Nende teenuste haldamine ja mastaabi muutmine manuaalselt oli tülikas. Infrastruktuuri

koodina hoidmine pakkus paljudele süsteemiadministraatoritele ja -arendajatele huvi. See avas võimaluse disainida ja ehitada platvorme ja süsteeme koodina, pakkuda välja parimaid praktikaid ning hiljem kasutada seda koodi, et korduvalt paigaldada samasugust infrastruktuuri. [7] [8]

Infrastruktuuri koodina on võimalik kasutada nii füüsilise kui ka virtuaalse infotehnoloogiataristu haldamiseks. Üldiselt hoitakse infrastruktuuri koodi versioonihaldustarkvarades, mis võimaldab koodi versioonimise.

Infrastruktuuri kirjutamine koodina väärtus seisneb kõige enam kolmes mõõdetavas kategoorias: hind, kiirus ja riskid. Kui kasutada infrastruktuuri koodina, siis on võimalik kokku hoida raha töötundide pealt, sest sarnast infrastruktuuri on võimalik juurutada juba olemas oleva koodi põhjal. Konfigureerimis- või juurutamiskoodi käivitamine võtab vähem aega kui manuaalne töö. See võimaldab pakkuda infrastruktuuri muudatusi või uut tarkvara kiiremini ja efektiivsemalt. Lisaks maandab koodi kasutamine riske ja vigu, mis võivad tekkida manuaalsel konfigureerimisel või tarkvara paigaldamisel.

Infrastruktuur koodina võimaldab arendajatel kirjutada koodi millega arendatakse platvorme ja infrastruktuuri. See annab arendajatele võimaluse kiiremini üles seada platvorm, mis on just nende tööks vajalik. Lisaks pakub infrastruktuuri kood ka konfiguratsioonihaldust, mis hoiab kogu infrastruktuuri ühtsena.

2.3 Lähteülesanne

2018. aasta kevadel soetas üks Baltikumi ja põhjamaade finantsettevõte omale 99-st füüsilisest serverist koosneva andmetöötlusplatvormi. Iga server omab 72 protsessori tuuma, 768GB operatiivmälu ning ligikaudu 120TB SSD elik pooljuht andmesalvestuspinda andmete salvestamiseks. 99-st serverist on moodustatud 7 serveriklastrit, mis on jagatud arendus-, testimis- ja toodangukeskkondade vahele. Parima jõudluse saavutamiseks pole kasutatud hüperviisoreid, vaid kõik teenused on paigaldatud füüsilise serveri operatsioonisüsteemile. See muudab serverite halduse keerulisemaks kuna teenuste paigaldamine ja konfigureerimine tuleks korraga teha 99-s serveris ning pole võimalik kasutada eelnevalt valmis ehitatud ja konfigureeritud virtuaalseid servereid.

Kirjeldatud platvormi infrastruktuuri ning teenuseid haldavad ja opereerivad hinnanguliselt 20 DevOps inseneri päevas. Lõppkasutajaid, kes töötlevad platvormil pakutavate teenustega platvormis hoitavaid andmeid, on ligikaudu 400. Ettevõte soovib haldusmeeskonna jaotada kaheks: infrastruktuuri- ja teenuste haldajad. Plaanitakse anda äriteenuste haldamise kohustus tiimidele, kes vahendavad platvormil pakutavaid teenuseid lõppkasutajateni. Neid inimesi on ligikaudu 20. Lõpptulemusena soovitakse, et 40-st insenerist umbes pooled tegelevad ainult platvormi infrastruktuuri ja infrastruktuuri teenuste arenduse ja haldamisega ning ülejäänud tegelevad ainult platvormil käivitatavate andmetöötlusteenuste ehk äriteenuste arenduse ja haldamisega.

Ettevõtte turvapoliitika on lubanud juurdepääsu toodangu keskkondade serveritele vaid infrastruktuuri inseneridele ning teenuskontodele. See tähendab, et äriteenuste insenerid ei saa serveritesse sisse logida, et enda halduses olevaid teenuseid paigaldada või uuendada. Äriteenuste haldajad peaksid turvapoliitika järgi kasutama tööriistu, mis ühenduvad serveritesse ning paigaldavad ja uuendavad tarkvara. Neid tööriistu pole juurutatud.

Käesoleva bakalaureuse töö eesmärk on leida lahendus, kuidas automatiseerida kirjeldatud platvormil kolmandate osapoolte tarkvara paigaldamise ja uuendamise protsessi, et võimaldada äriteenuste haldajatel paigaldada ja uuendada enda halduses olevat tarkvara toodangu keskkondades. Autor kirjeldab, kuidas juurutatavad tehnoloogiad vähendavad haldajate tööaega halduselt ning võimaldavad neil keskenduda rohkem platvormi arendusele.

2.4 Finantsettevõtte poolsed piirangud

Ettevõttes on kasutusel versioonihalduslahendusena Atlassian Bitbucketi tarkvara.

Bitbucket on Giti baasil arendatud tööriist, mis võimaldab arendajatel majutada oma koodi ning seda teiste arendajatega jagada. Bitbucket ühildub projektihaldustarkvaradega nagu Jira ja Trello, mis võimaldab arendajatel lihtsalt oma tööd hallata ühest tarkvarast. Ettevõttes on samuti kasutuses Jira projektijuhtimise tarkvara. [9]

Ettevõttes on kasutusel turvapoliitika, mis keelab kasutada pilvepõhised lahendusi ning kohustab majutama kõiki võimalikke tarkvaralisi tööriistu ettevõtte serveriparkides.

Ettevõttes on kasutusel Jenkinsi automatiseerimistarkvara. Jenkins on tasuta tarkvara, mis on mõeldud tarkvara arenduse automatiseerimiseks. See võimaldab arendajatel automaatselt käivitada teste ja ehitada kirjutatud koodi põhjal tarkvara lõplikke paigalduspakette. [10]

Lisaks nõuab ettevõte, et kasutatavad tarkvaralised tööriistad oleksid teenusepakkuja poolt arendatud toodangu keskkondadele ning teenusepakkuja peab olema võimeline pakkuma kasutajatuge vähemalt toodangu keskkonna jaoks.

2.5 Võimalikud infrastruktuuri koodi automatiseerimise lahendused

Järgnevalt kirjeldab autor võimalikke lahendusi infrastruktuuri koodi automatiseerimiseks ja juurutamiseks.

2.5.1 Ansible

Ansible on avatud lähtekoodiga tasuta tarkvaraline tööriist, mis on mõeldud tarkvara haldamiseks, konfiguratsioonide haldamiseks ning tarkvara juurutamiseks. Tööriist on loodud 2012. aastal Michael DeHaani poolt, kes püüdis lahendada probleemi IT-süsteemiadministraatorite jaoks, kus süsteemiadministraatorid pidid kasutama erinevaid tööriistu, et hallata süsteemide konfiguratsioone ning juurutada tarkvara. [11]

Tarkvara müüdi RedHat ettevõttele 2015. aasta oktoobris. [12]

Ansible'i on võimalik käivitada UNIX'i-liiki operatsioonisüsteemides, kuid Ansible suudab hallata nii UNIX'i-liiki süsteemides kui ka Windowsi operatsioonisüsteemides olevat konfiguratsiooni ja tarkvara. Tööriist ei kasuta oma töö tegemiseks agente vaid ühendub kaugserveriga kasutades SSH (*Secure Shell*) teenust UNIX'i-liiki süsteemides või Windows'i kaughaldus teenust Windows'i süsteemides. Ansible tarkvara toimib lükke meetodil. See tähendab, et Ansible server saadab muudatused kaugserveritesse.

Ansible kasutab oma töös Python'i, Windows PowerShell'i, Shell'i ja Ruby süntakseid. Kogu automatiseerimine ning käsud Ansible'i jaoks kirjutatakse YAML või INI failides. YAML failid võimaldavad inimloetavat süntaksit, mille Ansible muudab masinloetavaks. Ansible on ülesehitatud moodulitele, mis kasutavad oma töös Shell'i, Windows PowerShell'i või Python'i mooduleid. Moodulid võimaldavad kirjutada

arendajatel koodi, mida Ansible kasutab konfiguratsioonide haldamiseks, tarkvara juurutamiseks või orkestreerimiseks.

Ansible'l on tugev kogukond, kes arendavad mooduleid ning erinevaid valmis lahendusi sageli tehtavate operatsioonide jaoks.

2.5.2 Chef

Chef (eesti keeles „kokk") on avatud lähtekoodiga tasuta tarkvaraline tööriist, mis loodi 2009. aasta jaanuaris eelkõige tarkvara konfiguratsiooni haldamiseks. Chef on tasuta, aga ettevõttetel on võimalik osta tasuline tugi Chef teenusepakkujalt. [13]

Chef kasutab oma tööks klient-server lahendust. See tähendab, et halduses olevatesse masinatesse tuleb enne lahenduse kasutamist paigaldada agent. Chef tarkvara toimib tõmbe meetodil, mis tähendab, et agent küsib määratud aja tagant Chef'i serverist muudatusi. Kui konfiguratsiooni kirjeldavas koodis on toimunud muudatus, siis agent implementeerib need muudatused selles serveris, milles ta töötab.

Chef on ülesehitatud niinimetatud retseptide peale, mis moodustavad niinimetatud kokaraamatuid. Iga retsept kirjeldab ühte osa konfiguratsioonist, mida tuleb ajakohasena hoida. Kõik käsud ja tegevused on Chefi tegutsemiseks kirjeldatud DSL (*Domain specific language*) failidega. [14]

Chef oli mõeldud ainult konfiguratsiooni haldamiseks. See tähendab, et süsteemi administraatorid pidid tarkvara installeerimise automatiseerimiseks kasutama ka teisi tööriistu. Algselt oli Chef tasuline, kuid tänaseks on see Apache litsents 2.0 all vabalt kasutatav. Lisaks sellele on Chef'i edasi arendatud ning toetab tänaseks ka tarkvara installeerimise automatiseerimist. [15]

Chef'i jaoks kirjutatud retsepte ja muud koodi on võimalik versioonida kasutades Git lahendusi.

2.5.3 Puppet

Puppet on avatud lähtekoodiga vabavaraline tarkvaraline tööriist, mis loodi 2005. aastal Luke Kanies'i poolt eelkõige konfiguratsioonihaldustööriistana. [16]

Puppet on klient-server lahendus, mis toimib samuti tõmbe meetodil nagu Chef. See tähendab, et kaugserveritesse tuleb paigaldada agent enne, kui tööriist suudab oma tööga alustada. Tööriist suudab opereerida nii UNIX-liiki süsteemidega kui ka Windows'i süsteemidega. Kogu konfiguratsioon kirjutatakse manifestidena iga funktsionaalse objekti kohta. Manifeste on võimalik kirjutada Puppeti deklaratiivses keeles või Ruby DSL keeles.

Puppeti jaoks kirjutatud manifestfaile on võimalik versioonida kasutades Git lahendusi.

2.5.4 Terraform

Terraform on avatud lähtekoodiga tarkvaraline tööriist, mis on loodud põhiliselt infrastruktuuri haldamiseks serveriparkides. Terraform loodi 2014. aasta juulis ettevõtte HashiCorp poolt. [17]

Terraform loodi eelkõige infrastruktuuri haldamiseks serveriparkides. Infrastruktuuri mallid kirjutatakse koodina ning neid on võimalik kasutada, et ehitada sarnast infrastruktuuri. Terraform suudab paigaldada teenuseid ning tarkvara etteantud mallide põhjal.

Terraform'i on võimalik installeerida nii UNIX-liiki operatsioonisüsteemidesse kui ka Windows'i operatsioonisüsteemidesse. Tööriist kasutab oma töö tegemiseks HCL (*Hashicorp Configuration Language*) süntaksit, mis tõlgendatakse ümber API (*application programming interface*) päringuteks.

Kõik Terraformi tegevused toimuvad üle API päringute teenusepakkujate poole. Teenusepakkujateks võivad olla kõik tarkvarad ja infrastruktuuri pakkujad, kes on ehitanud omale API liidestuse, mis suudab vastu võtta päringuid. Terraform ei vaja oma tööks agente.

Terraform'i on kõige mugavam kasutada pilveteenuse pakkujate juures oma infrastruktuuri ehitamiseks või virtuaalserverite haldamiseks kohalikes serveriparkides. Füüsiliste serverite haldamine Terraformiga on võimatu kui halduses olevates serverites pole tarkvara, mis suudaks suhelda kasutades API päringuid.

Terraformi jaoks kirjutatud koodi on võimalik versioonida kasutades Git lahendusi.

3 Analüüs

Selles peatükis võrdleb autor eelnevalt kirjeldatud tarkvarasid ning leiab protsessi, mis lahendaks püstitatud probleemi.

3.1 Lahenduste võrdlus ja analüüs

Peatükis 2.5 kirjeldatud lahendustest kaks kasutavad oma töö tegemiseks agente, mis tähendaks, et enne lahenduse kasutamist tuleks paigaldada kõikidesse halduses olevatesse kaugserveritesse agent. See aga tähendaks tulevikus lisa tööd uute serverite liitmisel olemasolevasse halduslahendusse.

Terraformi hetkeversioon on 0.12.24, mis viitab sellele, et lahendus pole saavutanud valmisolekut toodangu keskkondade jaoks. Ettevõtte piirangutest tulenevalt saaks Terraformi kasutusele võtta alles siis, kui versioon oleks vähemalt 1.0.0. Samuti on Terraform'iga peaaegu võimatu hallata füüsilisi servereid. Autor välistab oma edasisest analüüsist Terraformi. [18]

Autor lähtub lahenduste võrdlemisel neljast püstitatud põhikriteeriumist:

1. lahenduse hind;
2. lahenduse juurutamise keerukus;
3. lahenduse liidestamine olemasolevate ettevõttes kasutuses olevate lahendustega;
4. lahenduse kasutamise keerukus.

Lahendus peab olema tasuta või võimalikult soodne. Juurutamine peab olema lihtne, et juurutamisprotsess oleks kiire ning lahenduse saaks kasutusele võtta nii kiiresti kui võimalik. Lahendus peab olema võimeline liidestuma ettevõttes juba kasutusel olevate tarkvaradega, et kasutamisel ei peaks hakkama juba olemasolevaid lahendusi uuesti arendama. Lahendus peab olema lihtsalt kasutatav. Kuna lahendust hakkavad kasutama arendajad, kes pole varem sarnaseid lahendusi kasutanud, siis peab õppeprotsess olema

võimalikult lühike ja lihtne, et ettevõte saaks lahenduse juurutamisest võimalikult kiiresti kasu.

Järgnevas tabelis (Tabel 1Tabel 1. Analüüsitavate lahenduste võrdlus.) kirjeldab autor lahenduste vastavust eelnimetatud kriteeriumitele.

Tabel 1. Analüüsitavate lahenduste võrdlus.

	Ansible	**Puppet**	**Chef**
Hind	Tasuta. RedHat ettevõte pakub tasulist lahendust, mis pakub rohkem võimalusi ning kasutajaliidest oma koodi käivitamiseks. [19]	Tasuta. Puppet ettevõte pakub tasulist lahendust koos kasutajatoega.	Tasuta. Chef ettevõte pakub tasulist lahendust koos lisa võimaluste ja kasutaja toega.
Juurutamise keerukus	Lihtne. Tuleb alla laadida Ansible pakett ning paigaldada see operatsioonisüsteemi. Seejärel tuleb konfigureerida manuaalselt Ansible tarkvara. Lisaks tuleb avada võrgupordid Ansible serverist kaugserveritesse ning lubada Ansible teenuskontol sisselogimine halduses olevatesse serveritesse. Seejärel saab alustada infrastruktuuri koodi kirjutamisega.	Keeruline. Puppeti kasutamiseks tuleb alla laadida, paigaldada ja konfigureerida Puppeti server ja andmebaas. Seejärel tuleb igas kaugserveris alla laadida ja paigaldada Puppeti agent ning konfigureerida suhtlus serveriga. Lisaks tuleb avada vajalikud võrgupordid kõikidest kaugserveritest Puppeti serverisse. Pärast seda on võimalik alustada infrastruktuuri koodi kirjutamisega.	Keeruline. Tuleb alla laadida Chef Workstation pakett, mis sisaldab vajalikke komponente Chef'i tarkvara käivitamiseks. Kõik paketid tuleb paigaldada ja konfigureerida vastavalt vajadustele. Igas kaugserverisse tuleb alla laadida ja paigaldada Chef'i agent, see konfigureerida ning avada tuleb võrgupordid kaugserverist Chef'i serverisse. Pärast seda on võimalik alustada infrastruktuuri koodi kirjutamisega.

	Ansible	**Puppet**	**Chef**
Liidestusvõimekus olemasolevate lahendustega	Ansible on võimalik liidestada Jenkinsi tarkvaraga. See võimaldab käivitada Ansible't Jenkinsist, mis annab võimaluse näha käivitatud koodi ning näha seda, kes seda koodi käivitas. Samuti võimaldab automatiseerida koodi käivitusprotsessi. [20]	Puppet on võimeline saatma Jenkinsisse infot paigaldatud konfiguratsioonide ja muudatuste kohta. Lisaks on võimalik Puppeti logisid saata ettevõtte kesksesse logihaldus tarkvarasse. [21]	Chef'i kliendi ja serveri logisid on võimalik saata läbi lisamooduli ettevõtte kesksesse logihaldus tarkvarasse.
Kasutuskeerukus	Lihtne kasutada. Kogu kood kirjutatakse YAML failides mis tagab loetavuse. Ansible kasutab oma tööks mooduleid, mis on põhjalikult dokumenteeritud Ansible kodulehel. Lisaks pakub Ansible valmis kirjutatud koodi sageli tehtavate toimingute jaoks. [22]	Pole väga lihtne kasutada. Puppet kasutab spetsiaalset keelt Puppet DSL, mis on rohkem suunatud süsteemi-administraatoritele ning mida pole väga lihtne õppida. [22]	Pole väga lihtne kasutada. Chef kasutab Ruby DSL keelt, milles kirjutatakse kõik vajalikud failid. Ruby DSL on rohkem suunatud tarkvara-arendajatele. [22]

Kõik analüüsitavad lahendused on avatud lähtekoodiga ning tasuta kasutamiseks. See tähendab, et iga lahendus katab esimese, lahenduse hinna, kriteeriumi. Igale lahendusele on võimalik juurde osta teenusepakkuja poolne tugi vähemalt toodangu keskkondade jaoks.

Puppet ja Chef toimivad tõmbe meetodil, mis tähendab, et juurutamise korral tuleb kaugserverites paigaldada ja konfigureerida agendi. Lisaks tuleb paigaldada server ja avada võrgupordid. Ansible töötab lükkemeetodil, mis tähendab, et paigaldada tuleb ainult Ansible server, lubada teenuskontodel sisselogimine kaugserveritesse ning avada vajalikud võrgupordid kaugserveritesse. Sisselogimine kaugserveritesse on lahendatud domeeni gruppide tasandil, mis tähendab, et uue teenuskonto lubamine serveritesse tähendaks teenuskonto lisamist kindlasse domeeni gruppi. Juurutamise lihtsuse kriteeriumile vastab kõige enam Ansible tarkvara.

Kuna Ansible toimib lükke meetodil, siis see tähendab, et töö käivitamine nõuab arendaja sekkumist. Ansible ühildub ettevõttes kasutusel oleva Jenkinsi automatiseerimistarkvaraga, mis võimaldab käivitada Ansible töid. See võimaldab pakkuda arendajatele veebiliidest, mille kaudu on neil võimalik organiseerida oma paigaldamis- või uuendamistöid. Jenkins võimaldab automaatset töö käivitamist, kui on toimunud koodi muudatus versioonihaldustarkvaras. Sellest kirjutab autor peatükis 4.3 Tuleviku võimalused.

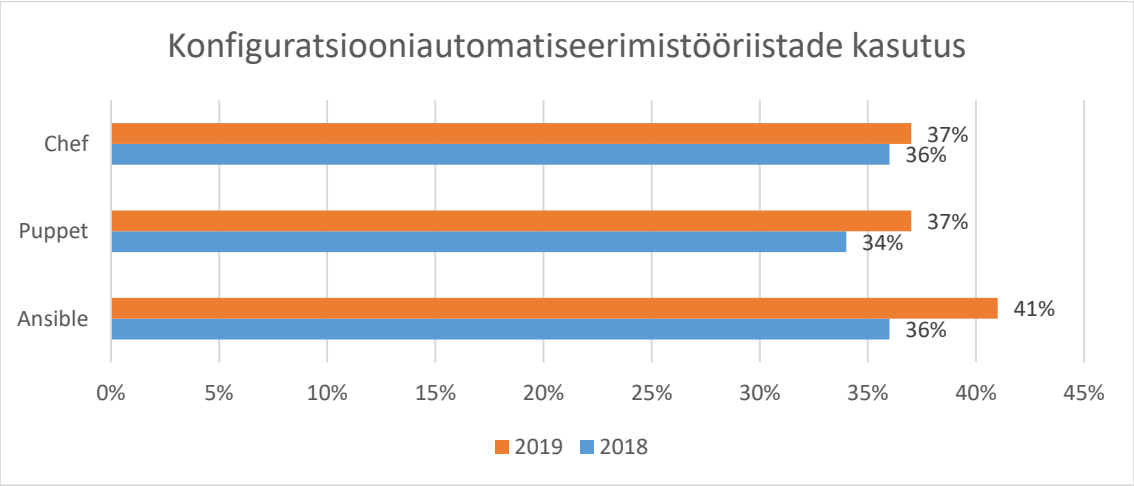
Puppet ja Chef ei vaja eraldi töö käivitamist, kuna neil on agendid, kes toimivad tõmbe meetodil, ehk agendid käivitavad töö iseseisvalt ettemääratud intervalli põhjal. Puppet ei paku tasuta veebiliidest tööde seireks. See tähendaks, et paigaldus ja uuendamis staatuste seireks tuleks lugeda agentide ja serveri logifaile. See tähendaks haldajatele lisakoormust. Puppet on võimalik liidestada ettevõtte logihaldustarkvaraga, mis pakub arendajale võimaluse logihalduse veebiliidese kaudu logisid lugeda. Chef pakub baaspaketiga koheselt veebiliidese, mis võimaldab staatuse seire veebiliidese kaudu.

Oluline aspekt on versioonihaldustarkvara liidestus. Kõik eelnimetatud lahendused toetavad koodi versioonimist ettevõtte versioonihaldustarkvaras.

Liidestust vajavad kõige enam Ansible ja Puppet, kuna neil puudub vabavaraline veebiliides tööde seireks.

Kasutusmugavus on kõige parem Ansible lahendusel, sest see kasutab lihtsalt loetavat ja kirjutatavat keelt YAML. Ansible kogukond jagab koodi, mis on kirjutatud sageli tehtavate toimingute jaoks. Puppet ja Chef kasutavad spetsiaalseid DSL keeli, mis on pigem suunatud süsteemiadministraatoritele või tarkvara arendajatele ning mida pole väga lihtne õppida.

Maailmas on tarkvara kasutusele võtmise trend Ansible kasuks. Alljärgnev joonis (Joonis 1) näitab, et üha enam hakatakse kasutusele võtma Ansible tarkvara. [23]

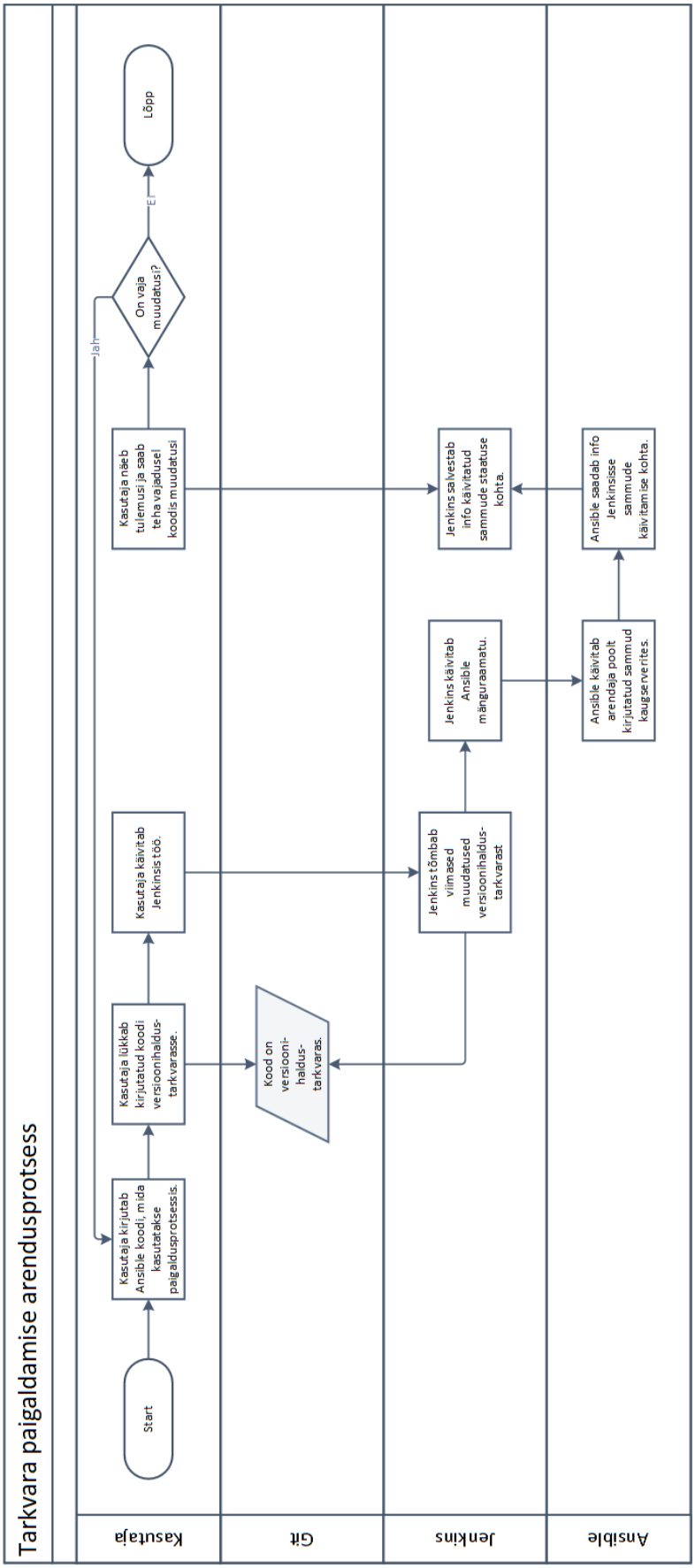


Joonis 1. Flexera raport 2019 "*RightScale 2019 State of the Cloud Report*".

Analüüsi põhjal otsustas autor oma töös edasi liikuda Ansible tarkvara juurutamisega.

3.2 Arendusprotsessi kirjeldus

Infrastruktuuri koodi arendamisprotsess on sarnane tarkvara arendusprotsessiga: koodi kirjutamine, versioonimine, testimine, vabastamine toodangusse. Arendaja kirjutab infrastruktuuri koodi, millega tehakse muudatusi infrastruktuuris või rakenduste konfiguratsioonis. Seejärel arendaja versioonib koodi kasutades versioonihaldustarkvara. Pärast seda arendaja käivitab tööriista, mis kasutab versioonitud koodi ja millega tehakse muudatused infrastruktuuris. Arendaja näeb töö käivitamise tulemust, kus kuvatakse tehtud muudatused. Seejärel saab arendaja manuaalselt testida infrastruktuuri ja veenduda, et kõik muudatused tehti korrektselt. Kui arendaja koodis on olnud viga, siis saab arendaja selle vea parandada, koodi uuesti versioonida ning uuesti testida. Joonis 2 kirjeldab eelmainitud protsessi käes olevas töös kasutatavate tarkvarade näitel.



Joonis 2. Tarkvara paigaldamise arendusprotsess kasutades Ansiblet ja Jenkinsit.

4 Praktiline osa

Selles peatükis kirjeldab autor, kuidas ta juurutas infrastruktuuri koodi automatiseerimislahenduse töös kirjeldatud finantsettevõtte näitel. Lisaks kirjeldab autor kahe tarkvara paigaldus protsessi kasutades juurutatud lahendust

4.1 Lahenduse juurutamine ja arhitektuur

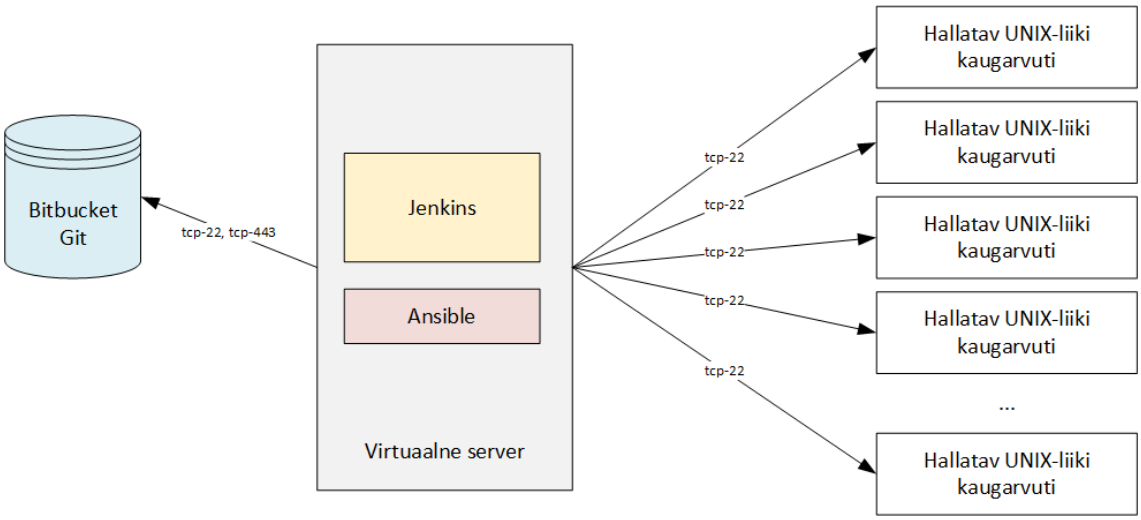
Lahenduse juurutamisel lähtus autor eelkõige ettevõtte turvapoliitikast tulenevate reeglitega ning peatükis 3 saadud analüüsi tulemusega. Samuti püüdis autor juurutamisel kasutada võimalikult palju lahendusi, mis olid juba ettevõtte siseselt juurutatud.

Autor kasutas Ansible serverina sama serverit kuhu oli paigaldatud Jenkinsi tarkvara. See server saab ligi versioonihaldustarkvarale. Ansible tuli alla laadida ja paigaldada serverisse */usr/local/bin* kausta. Seejärel paigaldati Jenkinsisse serverisse Ansible plugin, mis kasutab Ansible tarkvara just eelnimetatud kaustast. Järgnevalt tuli avada võrgupordid vastavalt Ansible dokumentatsioonile kõikidesse kaugserveritesse mida oli plaanis halduse alla võtta. Kuna käesolev bakalaureusetöö keskendub UNIX-liiki operatsioonisüsteemidega kaugserveritele, siis tuli avada võrgupordid SSH liikluse jaoks. SSH teenus kasutab edastusohje protokolli võrguporti 22 (*tcp*-22) ning krüptograafilisi lahendusi liikluse turvamiseks. Seejärel tuli lubada teenuskontol sisselogimine kaugserverisse.

Teenuskontol serverisse sisselogimise lubamisseks tuli lisada teenuskonto vastavasse ettevõtte domeeni gruppi. Teenuskontol on oma kasutajanimi ja parool. Ansible suudab ühenduda kaugserverisse kasutades nii kasutajanime ja parooli kui ka kasutades asümmeetrilise krüptoalgoritmiga võtmepaari. Autor kasutas esmase ühendumise jaoks teenuskonto kasutajanime ja parooli, mis salvestati Jenkinsi võtmehoidlasse. Tulevikus plaanib autor kasutada avaliku võtmega krüptoalgoritmi võtmepaari. Selleks tuleb teenuskonto jaoks kaugserveris seadistada avalik võti ning privaatne võti salvestada

Jenkinsi võtmehoidlasse. Jenkinsis olev Ansible moodul kasutab võtme identifikaatorit, et pärida Jenkinsi võtmehoidlast vajalik võti ühenduse loomiseks.

Järgnev joonis (Joonis 3) näitab üldist arhitektuuri juurutatava lahenduse kohta.



Joonis 3. Ansible arhitektuur finantsettevõtte näitel.

Autor paigaldas Ansible tarkvara Jenkinsi serverisse ning paigaldas vajalikud pluginad Jenkinsisse. Lisaks avas autor võrgupordid hallatavatesse serveritesse ning lubas teenuskontol neisse sisselogimise. Sellega oli juurutamisprotsess lõppenud.

4.2 Arendatud moodulite näited

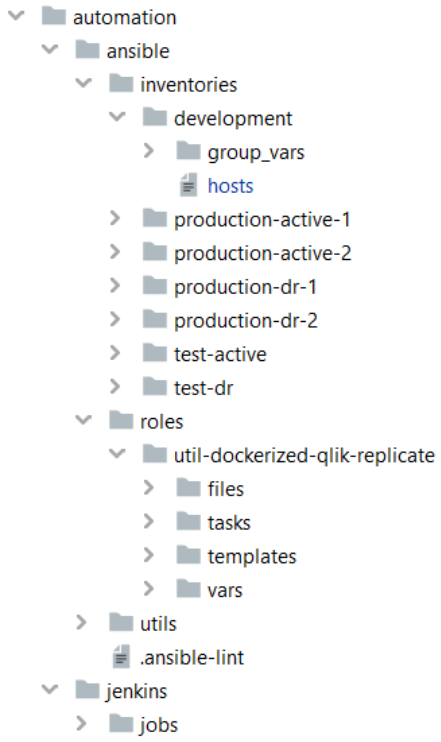
Bakalaureusetöö eesmärk on juurutada lahendus, mis võimaldaks teenuste haldajatel automatiseeritult installeerida ja uuendada tarkvara ning aega kokku hoida haldustoimingutelt. Autor valis välja kaks tarkvara, millele ta kirjutas koodi, mis võimaldab tarkvara paigaldada ning uuendada.

4.2.1 Konteineriseeritud Qlik Replicate paigaldamine

Andmetöötlusplatvormil tahetakse kasutada Qlik ettevõtte poolt pakutavat tarkvara Replicate, mille eesmärk on koguda muudatusi andmebaasidest ning need muudatused edasi saata teistele tarkvaradele. Teised tarkvarad saavad selle põhjal reaalajas uuendada raporteid ja muud infot. Teenuse talituspidevusplaani kohaselt peab tarkvara olema kätte saadav 99.9% aastas. Maksimaalne mitteplaneeritud järjestikune häire võib olla kuni 4 tundi.

Tulenevalt eelnimetatud tarkvara omadustest ja ettevõtte poolsetest nõuetest otsustas autor kasutada tarkvara käivitamiseks Dockeri konteinerite lahendust. Docker võimaldab UNIX-liiki operatsioonisüsteemides tekitada niinimetatud virtuaalseid konteinereid, mis käituvad nagu virtuaalsed serverid, kuid ei oma isiklikku operatsioonisüsteemi. Selleks kasutab ta UNIX-liiki operatsioonisüsteemides olevaid isoleerimisfunktsioone. Dockeri tarkvara on kaugserveritesse juba paigaldatud.

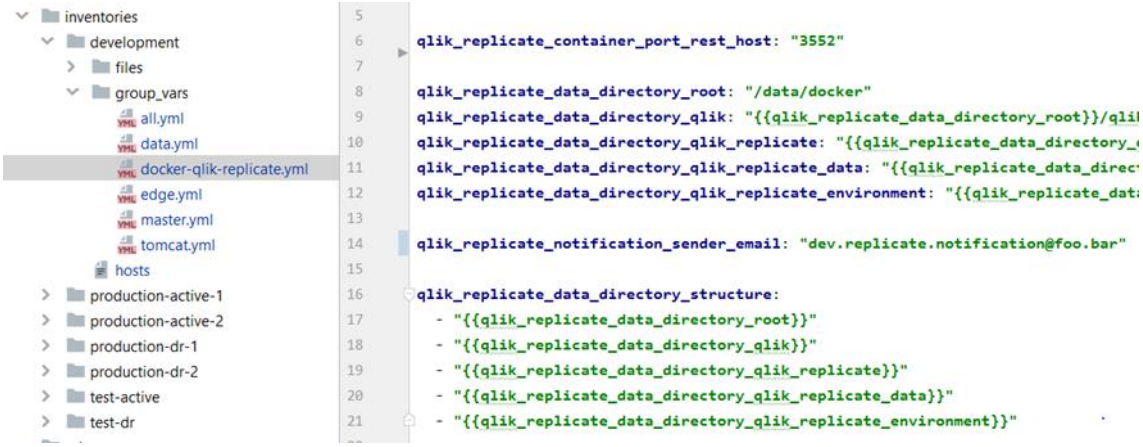
Autor valmistas ette vajaliku kaustade struktuuri vastavalt Ansible dokumentatsioonile. Joonis 4 näitab loodud kaustade struktuuri.



Joonis 4. Ansible kaustade struktuur Qlik Replicate tarkvara haldamiseks.

Struktuur on jaotatud põhiliselt kaheks: Ansible osa ja Jenkinsi osa. Ansible osas hoitakse kogu koodi Ansible jaoks ning Jenkinsis koodi Jenkinsi jaoks. Ansible kaustad jagunevad *inventories*, *roles* ja *utils*. *Inventories* kaust jaguneb keskkondadeks, kus hoitakse keskkonnamuutujaid iga keskkonna kohta. Lisaks hoitakse seal infot keskkonna kaugserverite kohta. Autor valmistas ette igale keskkonnale eraldi kausta, kus hoitakse Ansible faile, mis on muutuvad vastavalt keskkonnale (Joonis 5). *Roles* kaustas hoitakse Ansible rolle, mis jagunevad alamfailideks, kus hoitakse koodi iga eraldiseisva tegevuse kohta. Rollid on mõeldud ühe funktsionaalse objekti haldamiseks. *Roles* kaust jaguneb üldiselt rolli tegevusteks, mallideks ja rollipõhisteks muutujateks. Lisaks võib seal hoida faile, mis on vajalikud rolli tööks. Näiteks sertifikaadid või muud

failid, mis ei muutu keskkonna põhiselt. *Utils* kaustas hoitakse Ansible niinimetatud mänguraamatuid (ingl. *Playbook*), mis käivitavad rolle või üksikuid alamülesandeid. [24]



Joonis 5. Qlik Replicate tarkvara arenduskeskkonnapõhised muutujad.

Dockeri kasutamiseks tuleb luua Dockeri fail (ingl. *Dockerfile*). Dockeri fail on tekstifail, mis kirjeldab, kuidas tarkvara paigaldatakse ja konfigureeritakse. Docker failist luuakse Dockeri konteinertõmmis. Dockeri konteinertõmmis on failide kogum, mille põhjal on võimalik luua Dockeri konteinereid, milles on kogu kirjeldatud tarkvara paigaldatud ja konfigureeritud. Qlik Replicate tarkvara installeerimine on kirjeldatud ainult Docker faili sees (Joonis 6). Dockeri fail on kirjutatud Ansible mallina *util-qlik-replicate* rolli juures, millesse on võimalik sisestada muutujaid, mis saavad väärtuse vastavalt keskkonnale sel hetkel, kui koodis malli kasutatakse.

Joonis 6. Docker faili mall Qlik Replicate konteinermalli loomiseks.

Ansible kasutab tööde kogumiku kirjeldamiseks mänguraamatuid, milles kirjeldatakse kõik sammud, mida soovitakse kaugserveris teha. Alljärgneval joonisel (Joonis 7) on näha kood, millega käivitatakse kaugserveris Dockeri konteiner. Seda koodi on võimalik mänguraamatu kaugu välja kutsuda.

```
- name: create replicate data directory structure
  file:
    path: "{{item}}"
    state: directory
  with_items: "{{qlik_replicate_data_directory_structure}}"

- name: create replicate environment file
  template:
    src: "environment_conf.j2"
    dest:
"{{qlik_replicate_data_directory_qlik_replicate_environment}}/{{env}}.json"

- name: run Qlik Replicate container
  docker_container:
    name: "{{qlik_replicate_container_name}}"
    image: "{{qlik_replicate_image}}"
    state: started
    restart: true
    restart_policy: on-failure
    ports: "{{qlik_replicate_container_ports}}"
    volumes: "{{qlik_replicate_container_volumes}}"
    env: "{{qlik_replicate_container_env}}"
    network_mode: host
```

Joonis 7. Qlik Replicate Dockeri konteineri käivitamine kasutades Ansible docker_container moodulit.

Ansible tarkvara omab erinevaid mooduleid, millega on võimalik kirjutatav kood muuta lihtsalt loetavaks. Autor kasutas konteineri käivitamiseks *docker_container* moodulit. Eelneva koodi saaks ka kirjutada kasutades teisi mooduleid. Sellisel juhul pole kood lihtsasti loetav ja vajab rohkem teadmisi koodist arusaamiseks. Alljärgneval joonisel (Joonis 8) on näha samad sammud, kasutades mooduleid nimedega *shell* ja *template*.

```
- name: create replicate data directory structure
  shell: |
    mkdir -p {{qlik_replicate_data_directory_root}} \
    {{qlik_replicate_data_directory_qlik}} \
    {{qlik_replicate_data_directory_qlik_replicate}} \
    {{qlik_replicate_data_directory_qlik_replicate_data}} \
    {{qlik_replicate_data_directory_qlik_replicate_environment}}

- name: create replicate environment file
  template:
    src: "environment_conf.j2"
    dest:
"{{qlik_replicate_data_directory_qlik_replicate_environment}}/{{env}}.json"

- name: run Qlik Replicate container
  shell: |
    docker run -p {{qlik_replicate_container_ports}} \
    -v
{{qlik_replicate_container_volume_data_host}}:{{qlik_replicate_container_volu
me_data_container}} \
    -v
{{qlik_replicate_container_volume_environment_host}}:{{qlik_replicate_contain
er_volume_environment_container}} \
    -e ReplicateRestPort={{qlik_replicate_container_port_rest_container}} \
    -e ReplicateAdminPassword={{qlik_replicate_admin_password_encrypted}} \
    -e ReplicateMasterPassword={{qlik_replicate_master_password_encrypted}} \
    -e ReplicateEnvironment={{env}} \
    --network host \
    --name {{qlik_replicate_container_name}} \
    --restart on-failure \
    {{qlik_replicate_image}}
```

Joonis 8. Qlik Replicate Dockeri konteineri käivitamine kasutades Ansible shell moodulit.

Eelkirjeldatud Ansible koodi on võimalik käivitada Ansible serveri käsurealt või kasutades muid tööriistu, mis suhtlevad serveri käsureaga. Autor kirjutas oma töös, et ettevõte kasutab Jenkinsi automatiseerimislahendust ning Ansible on võimalik ühendada Jenkinsiga. Peatükis 4.1 kirjutas autor Ansible juurutamisest ning Jenkinsi mooduli paigaldamisest.

Jenkins kasutab oma tööks niinimetatud torustiku skripti (ingl. *Pipeline script*), mis koondab endasse tegevuste automatiseerimiseks vajalikud sammud. Autor kirjutas torustiku skripti Qlik Replicate konteineri haldamiseks (Lisa 1). Torustiku skripti lugedes on näha, et autor määrab versioonihaldustarkvara aadressi, kus hoitakse kirjutatud koodi ning lubab töö käivitajal määrata parameetreid töö tegemiseks:

1) Giti haru, kus koodi kirjutati;

2) keskkond, kus soovitakse koodi käivitada;

3) mänguraamat, mida soovitakse käivitada;

4) keskkonnas olevad serverid, mille peal on võimalik konteinerit käivitada.

Neid parameetreid kasutatakse Jenkinsi skripti teises sammus, kus kasutatakse *ansiblePlaybook* moodulit. Selles moodulis määratakse käivitamise keskkond, käivitatav kood ja vajalik autentimisinfo mida Ansible kasutab kaugserverisse sisselogimiseks.

Kõige lõpus kasutab autor *cleanWS* meetodit, mis kustutab Gitist alla laetud failid Jenkinsi serveri ajutistest kaustadest. Sellega hoiab autor kokku Jenkinsi serveri salvestuspinda.

Töö käivitamisel Jenkinsist käivitatakse ette antud Ansible mänguraamat, mis käivitab kogu arendaja kirjutatud koodi vastavalt reeglitele. Tulemus käivitatud töö kohta on nähtav Jenkinsi veebiliidesest (Joonis 9). Jooniselt on näha, et kõik kirjutatud sammud konteineri käivitamiseks käivitati edukalt. Esimese sammu juures nähtav *ok* näitab, et serveris olev kaustade struktuur vastas juba kirjeldatud nõuetele ning Ansible ei pidanud midagi muutma. Järgmine samm näitab, et serveris loodi tarkvara jaoks keskkonda kirjeldav fail. Seejärel käivitati Qlik Replicate konteiner ning ülejäänud sammud jäeti vahele, kuna need ei vastanud käivitatava mänguraamatu kriteeriumitele. Kõige lõpus on näha *cleanWs* meetodi käivitamist.

```
TASK [../roles/util-dockerized-qlik-replicate : create replicate data directory structure] ***
ok: [edge2] => (item=/data/docker)
ok: [edge2] => (item=/data/docker/qlik)
ok: [edge2] => (item=/data/docker/qlik/replicate)
ok: [edge2] => (item=/data/docker/qlik/replicate/data)
ok: [edge2] => (item=/data/docker/qlik/replicate/environment)

TASK [../roles/util-dockerized-qlik-replicate : create replicate environment file] ***
changed: [edge2]

TASK [../roles/util-dockerized-qlik-replicate : run Qlik Replicate container] ***
changed: [edge2]

TASK [../roles/util-dockerized-qlik-replicate : include_tasks] *****************
skipping: [edge2]

TASK [../roles/util-dockerized-qlik-replicate : include_tasks] *****************
skipping: [edge2]

TASK [../roles/util-dockerized-qlik-replicate : include_tasks] *****************
skipping: [edge2]

TASK [../roles/util-dockerized-qlik-replicate : include_tasks] *****************
skipping: [edge2]

PLAY RECAP *********************************************************************
edge2                      : ok=11   changed=3    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0

[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Declarative: Post Actions)
[Pipeline] cleanWs
[WS-CLEANUP] Deleting project workspace...
[WS-CLEANUP] Deferred wipeout is used...
[WS-CLEANUP] done
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Joonis 9. Väljavõte Jenkinsi veebiliidese vaatest Ansible tulemuste kohta pärast edukat käivitamist.

Kui arendaja poolt kirjutatud koodis on viga või midagi läks koodi käivitamisel valesti, siis Ansible saadab selle vea Jenkinsisse ning kogu Jenkinsi töö peatub kui Ansible koodi pole arendatud teisiti. Ansible võimaldab ka arendajatel kirjutada koodile loogikat, mis näiteks käivitab teatud vigade korral teistsuguse koodi, millega muudetakse kõik tehtud muudatused tagasi selliseks nagu need olid enne mänguraamatu käivitamist.

Sellise tarkvara manuaalne paigaldamine nõuaks, et arendaja ühenduks serveri käsureaga ja käivitaks käsurealt teatud käske. Automatiseerimine lihtsustab protsessi ning selle tarkvara paigaldamine automatiseeritult võtab aega keskmiselt kolm minutit.

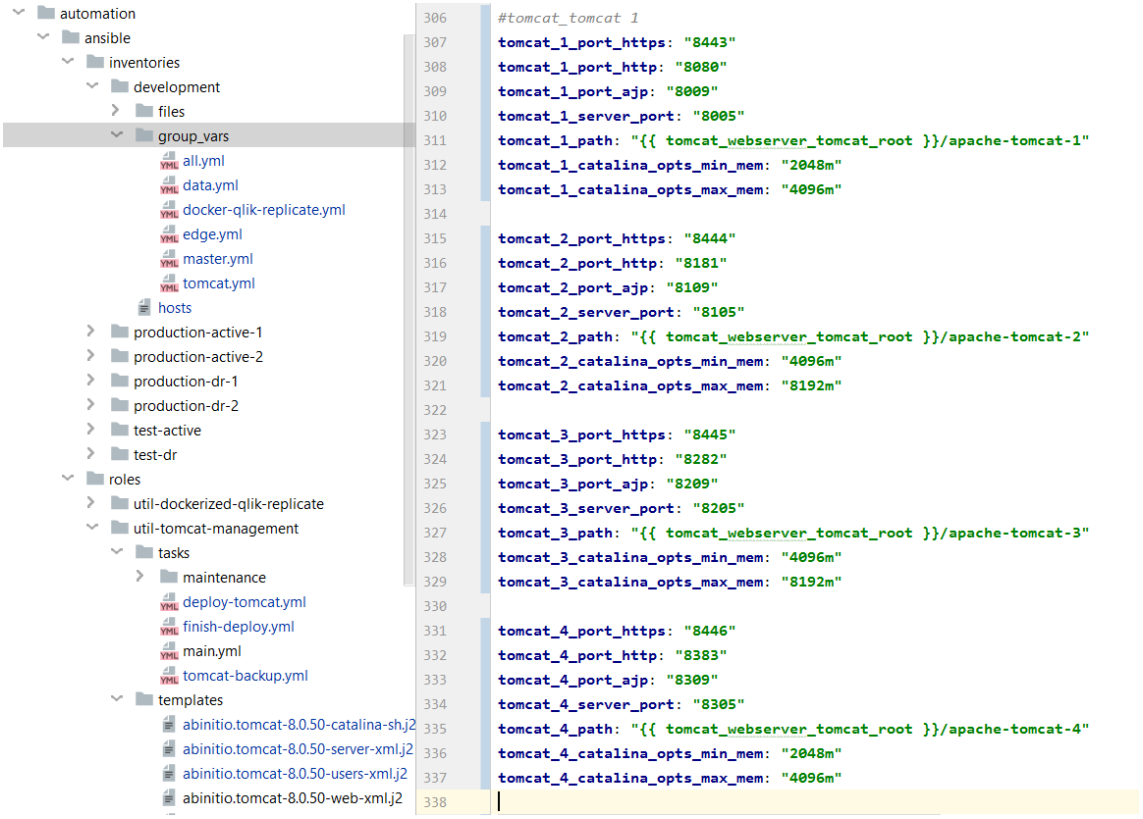
4.2.2 Apache Tomcat serveri paigaldamine ja uuendamine

Apache Tomcat server (vahel nimetatud ka kui „Tomcat“) on avatud lähtekoodiga serveri tarkvara, mis on mõeldud Java veebirakenduste käivitamiseks. Apache Tomcat

tarkvara loodi 1998. aastal James Duncan Davidson'i poolt, kes oli tol ajal tarkvara arhitekt ettevõttes Sun Microsystems. [25] [26]

Käesolevas töös kirjeldatud andmetöötlus platvorm kasutab Tomcat veebiserverite lahendust veebipõhiste andmetöötlustööriistade käivitamiseks. Kokku on platvormil 17 Tomcat serverit. Selleks, et nende paigaldus ja uuendamine oleks lihtsam, otsustas autor kirjutada infrastruktuuri koodi, mis need tegevused automatiseeriks.

Autor lisas eelnevalt kirjutatud Ansible koodile Tomcat serveri paigaldamiseks vajaminevad failid: keskkonnamuutujad ja sammud, mida tuleb teha, et tarkvara paigaldada (Joonis 10).



Joonis 10. Tomcat veebiserveri keskkonnamuutujad koos lisatud kaustade struktuuri ja muude failidega.

Tomcat serveri paigaldamine on triviaalne võrreldes teiste tarkvarade paigaldamisega:

1. Internetist tuleb alla laadida paigalduspaketi arhiveeritud ja tihendatud fail;
2. Alla laetud fail tuleb serveris lahti pakkida;
3. Konfiguratsiooni failides tuleb teha keskkonnale vajalikud muudatused;
4. Tomcat server tuleb käivitada.

Kõik käesolevas töös kirjeldatud platvormi serverid on eraldatud internetist, mis tähendab, et serveritel pole võimalik otse internetist faile alla laadida. Selle probleemi lahendamiseks on ettevõttes kasutusele võetud avalik failide jagamise platvorm, millele on juurdepääs kõikidel ettevõtte serveritel. Autor laadis üles eelnimetatud failide jagamise platvormile Apache Tomcat serveri kaks versiooni: 8.0.50 ja 8.5.47. Paraku ei toeta platvormil kasutatavad andmetöötlustarkvarad uuemaid Tomcat serveri versioone.

Autor kirjutas vajaliku faili alla laadimiseks, lahti pakkimiseks ja konfiguratsiooni tegemiseks Ansible koodi, kasutades Ansible mooduleid *get_url*, *unarchive* ja *template* (Joonis 11). Esimene moodul laeb failide jagamise platvormilt alla vajaliku pakettfaili ning salvestab selle serverisse. Teine moodul pakib pakettfaili lahti ning kolmas moodul muudab konfiguratsioonifaile vastavalt Ansible mallides tehtud muudatustele.

```
- name: get tomcat tar from artifactory
  get_url:
    url: "{{tomcat_artifactory_address}}/apache-tomcat-
{{tomcat_version_nr}}.tar.gz"
    dest: "{{tomcat_webserver_tomcat_temp}}"

- name: unarchive
  unarchive:
    src: "{{tomcat_installation_file_dir}}/apache-tomcat-
{{tomcat_version_nr}}.tar.gz"
    dest: "{{tomcat_webserver_tomcat_temp}}"
    creates:
"{{tomcat_webserver_tomcat_temp}}/{{tomcat_webserver_tomcat_name}}"
    owner: "{{tomcat_admin_user}}"
    group: "{{tomcat_group}}"

- name: copy templates
  template:
    src: "{{item.src}}"
    dest: "{{item.dest}}"
    owner: "{{tomcat_admin_user}}"
    group: "{{tomcat_group}}"
    mode: "{{item.mode}}"
  with_items:
    - { src: "abinitio.tomcat-setenv.j2", dest:
"{{tomcat_webserver_tomcat_temp}}/apache-tomcat-
{{tomcat_instance_nr}}/bin/setenv.sh", mode: '0755' }
    - { src: "abinitio.tomcat-{{tomcat_version_nr}}-web-xml.j2", dest:
"{{tomcat_webserver_tomcat_temp}}/apache-tomcat-
{{tomcat_instance_nr}}/conf/web.xml", mode: '0600' }
    - { src: "abinitio.tomcat-{{tomcat_version_nr}}-server-xml.j2", dest:
"{{tomcat_webserver_tomcat_temp}}/apache-tomcat-
{{tomcat_instance_nr}}/conf/server.xml", mode: '0600' }
    - { src: "abinitio.tomcat-{{tomcat_version_nr}}-users-xml.j2", dest:
"{{tomcat_webserver_tomcat_temp}}/apache-tomcat-
{{tomcat_instance_nr}}/conf/tomcat-users.xml", mode: '0600' }
    - { src: "abinitio.tomcat-{{tomcat_version_nr}}-catalina-sh.j2", dest:
"{{tomcat_webserver_tomcat_temp}}/apache-tomcat-
{{tomcat_instance_nr}}/bin/catalina.sh", mode: '0755' }
    - { src: "abinitio.tomcat-host-manager-context-xml.j2", dest:
"{{tomcat_webserver_tomcat_temp}}/apache-tomcat-
{{tomcat_instance_nr}}/webapps/host-manager/META-INF/context.xml", mode:
'0755' }
    - { src: "abinitio.tomcat-manager-context-xml.j2", dest:
"{{tomcat_webserver_tomcat_temp}}/apache-tomcat-
{{tomcat_instance_nr}}/webapps/manager/META-INF/context.xml", mode: '0755' }
```

Joonis 11. Ansible kood Tomcat veebiserveri paigaldamiseks.

Seejärel tuleb Tomcat server käivitada ning selleks kirjutas autor faili, mida on võimalik käivitada eraldiseisvalt või välja kutsuda teiste sammude keskel (Joonis 12). Nimetatud fail sisaldab koodi (Lisa 2), mis käivitab või lülitab välja kasutaja poolt määratud Tomcat serveri.

```
- name: Startup tomcat-{{tomcat_instance_nr}} instance
  include_tasks: ./maintenance/tomcat-instance-start-stop.yml
  vars:
    tomcat_name: "{{tomcat_instance_nr}}"
    tomcat_start_stop: "startup"
```

Joonis 12. Teiste funktsionaalsete objektide kasutamine Ansible sammude vahel.

Tomcat serveri uuendamine on veidi keerulisem kui serveri paigaldamine:

1. Internetist tuleb alla laadida paigalduspaketi arhiveeritud ja tihndatud fail;
2. Alla laetud fail tuleb serveris lahti pakkida;
3. Konfiguratsiooni failides tuleb teha keskkonnale vajalikud muudatused;
4. Vanast Tomcati serverist tuleb kopeerida kõik veebirakenduste failid uue Tomcat serveri kaustadesse;
5. Uus Tomcat server tuleb tööle panna;
6. Vana Tomcat server tuleb arhiveerida.

Kuna Apache Tomcat serveri uuendamine tähendab uue versiooni Tomcat serverist tööle panemist ning veebirakenduste kopeerimist vanast serverist uuele, siis on autoril võimalus kasutada eelnevalt kirjutatud koodi ning seda mitte kopeerida.

Autor kirjutas Ansible koodi, mis kopeerib kõik veebirakenduste failid vanalt serverilt ajutisse kausta kust need hiljem uuele serverile kopeeritakse. Seejärel arhiveeritakse vana Tomcat server (Joonis 13). Autor paigutas mänguraamatus üks teise järele Tomcat serveri arhiveerimiskoodi ja paigaldamiskoodi ning kogu arendus Tomcat serveri uuendamiseks oli valmis.

```
- set_fact:
    fact1: "apache-tomcat-{{tomcat_instance_nr}}"

- name: check if instance exists
  shell: ls {{tomcat_webserver_tomcat_root}} | grep -x {{fact1}}
  register: tomcat_instance
  become: true
  become_user: "{{tomcat_admin_user}}"
  ignore_errors: yes

- set_fact:
    is_installed: true
  when: fact1 in tomcat_instance.stdout

- block:
  - name: shutdown tomcat-{{tomcat_instance_nr}} instance
    include_tasks: ./maintenance/tomcat-instance-start-stop.yml
    vars:
      tomcat_name: "{{tomcat_instance_nr}}"
      tomcat_start_stop: "shutdown"

  - name: check if any war files exist
    shell: ls {{tomcat_webserver_tomcat_root}}/apache-tomcat-
{{tomcat_instance_nr}}/webapps/ | grep ".war"
    register: tomcat_war_exist
    ignore_errors: yes

  - set_fact:
      war_exist: true
    when: "'.war' in tomcat_war_exist.stdout"

  - block:
    - name: create temp folder for .war files
      file:
        path: "{{tomcat_webserver_tomcat_temp}}/apache-tomcat-
{{tomcat_instance_nr}}"
        state: directory

    - name: copy .war files to temp dir
      shell: cp {{tomcat_webserver_tomcat_root}}/apache-tomcat-
{{tomcat_instance_nr}}/webapps/*.war {{tomcat_webserver_tomcat_temp}}/apache-
tomcat-{{tomcat_instance_nr}}/.
    when: war_exist | bool

  - name: create temp folder for tomcat backup
    file:
      path: "{{tomcat_webserver_tomcat_root}}/apache-tomcat-
{{tomcat_instance_nr}}-backup"
      state: directory

  - name: Backup existing tomcat instance
```

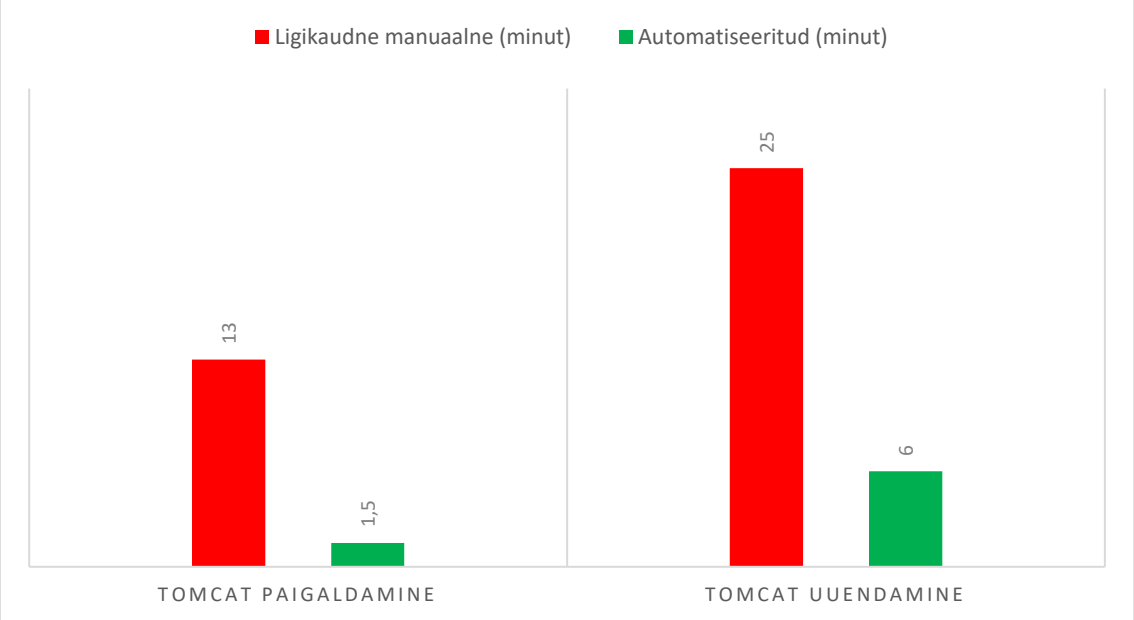
```
    archive:
      path: "{{tomcat_webserver_tomcat_root}}/apache-tomcat-
{{tomcat_instance_nr}}"
      dest: "{{tomcat_webserver_tomcat_root}}/apache-tomcat-
{{tomcat_instance_nr}}-backup/apache-tomcat-{{tomcat_instance_nr}}-
{{lookup('pipe','date +%Y%m%d%H%M%S')}}.tar.gz"

  - name: delete old tomcat
    file:
      path: "{{tomcat_webserver_tomcat_root}}/apache-tomcat-
{{tomcat_instance_nr}}"
      state: absent

  when: is_installed | bool
```

Joonis 13. Ansible kood Tomcat veebirakenduste kopeerimiseks ja arhiveerimiseks.

Kogu Apache Tomcat serveri automatiseeritud paigaldusprotsess võtab aega umbes 90 kuni 110 sekundit. Manuaalselt läheks aega umbes 10-15 minutit. Manuaalne uuendamise protsess võtab keskmiselt aega umbes 20-30 minutit installatsiooni kohta. See tähendab, et 17 serveri uuendamine võtab aega umbes 5-6 tundi. Automatiseeritud Tomcat serveri uuendamine võttis aega umbes 5-7 minutit. See tähendab, et kõigi 17 serveri uuendamine võtab aega umbes 100 minutit. Alljärgnev joonis (Joonis 14) näitab võrdlust Tomcat serveri manuaalse ja automaatse paigaldamise ning uuendamise kohta. Jooniselt on näha, kuidas automatiseerimine on kiirendanud tarkvara paigaldamisprotsessi.



Joonis 14. Tomcat tarkvara manuaalse paigaldamise ja uuendamise protsessi erinevus võrreldes automatiseeritud protsessiga.

Arendajad saavad paigaldada tarkvara automatiseeritult toodangu keskkondadesse ning neil pole tarvis juurdepääsu serverite käsureale. See tähendab võitu ajas ning arendaja võib kindel olla, et kõik paigaldused ja uuendused on ühesugused. Sellega maandatakse võimalikud inimtekkelised vead. Kõik muudatused kajastuvad versioonihalduses ning kogu paigaldatud tarkvara on paigaldatud vastavalt kirjutatud koodile.

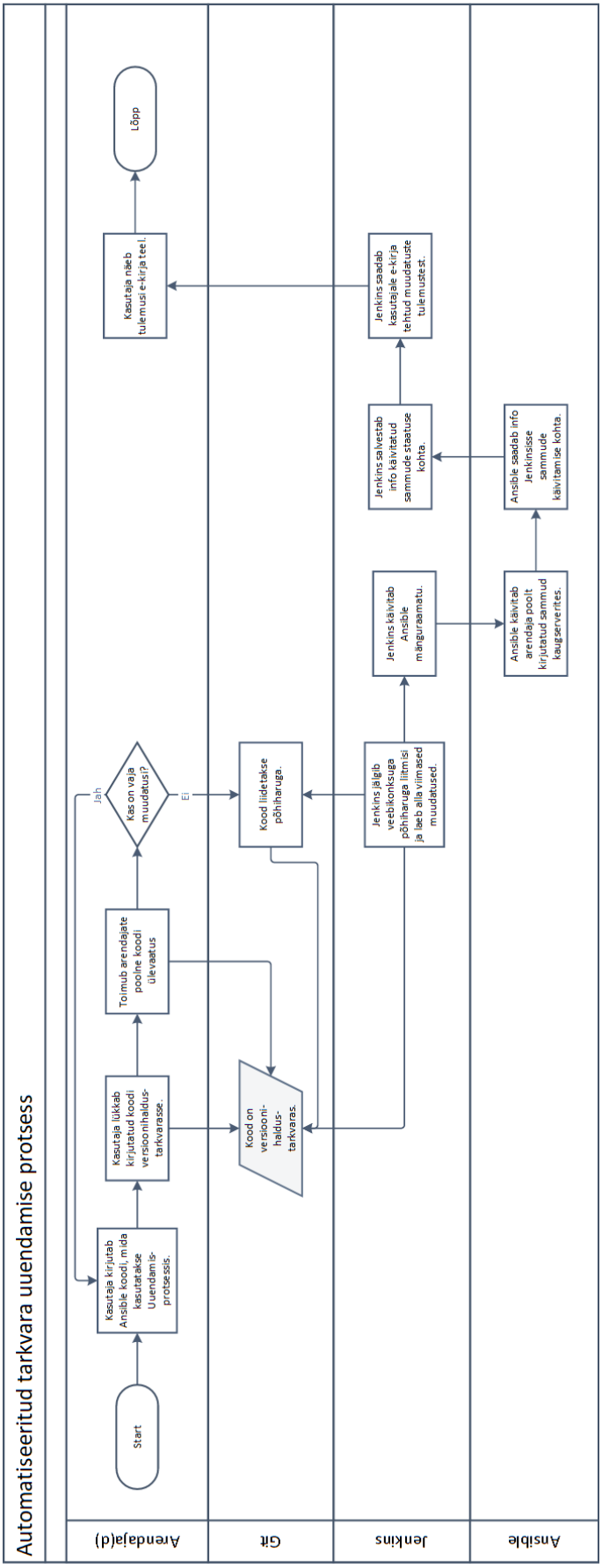
4.3 Tuleviku võimalused

Käesolev bakalaureusetöö eesmärk on leida lahendus ettevõtte arendajatele, kelle halduses on tarkvarad, mida neil pole võimalik hallata otse füüsilise serveri käsurealt. Lisaks on eesmärk automatiseerida tarkvara paigaldamise ja uuendamise protsessi. Järgnevalt kirjeldab autor, kuidas on võimalik protsessi veel automatiseerida, kuid mida käesoleva bakalaureusetöö raames ei juurutata.

4.3.1 Automaatne tarkvara uuendamine kasutades veebikonkse

Praegu juurutatud lahendus tähendab automatiseeritud paigaldus- ja uuendamiskoodi manuaalset käivitamist Jenkinsi veebiliidese kaudu. Jenkins on võimalik ühendada Gitiga kasutades nn. veebikonkse (ingl. *webhook*), mis jälgivad muudatusi versioonihaldustarkvaras ning käivitavad automaatselt Jenkinsi töö.

Protsess algab arendaja poolse muudatuse tegemisest koodis. Näiteks muudab arendaja koodis tarkvara versiooni muutujat ning lükkab muudatused versioonihaldustarkvarasse. Versioonihalduses toimub arendajate poolne koodi ülevaatus ning kui kood on saanud piisava heakskiidu, siis Jenkinsi veebikonks käivitab vastavalt reeglitele vajaliku töö. Käivitatud töö uuendab automatiseeritult tarkvara versioonile mille arendaja koodis määras. Arendajale saadetakse e-kirja teel käivitatud sammude tulemused ning sellega protsess lõppeb (Joonis 15).



Joonis 15. Automatiseeritud tarkvara uuendamise protsess.

Selline protsess hoiab arendajate aega kokku manuaalse töö käivitamise pealt. Lisaks pakub versioonihaldustarkvara võimaluse jälgida tehtavaid muudatusi.

4.3.2 Infrastruktuuri koodi testimine

Samuti nagu iga arendatav programmikood vajab ka infrastruktuuri kood testimist. Eelnevalt kirjeldas autor, kuidas on võimalik koodi manuaalselt testida. Kood kirjutatakse valmis, versioonitakse ning juurutatakse serveritesse. Seejärel peab arendaja manuaalselt testima ja kontrollima paigaldatud koodi funktsionaalsust.

Infrastruktuuri koodi testimine jaguneb kolmeks:

1. Koodi stiili ja koodi terviku testimine;
2. Üksiku funktsionaalse objekti testimine ehk ühiktestimine;
3. Koodi integratsiooni testimine ehk terviktestimine.

Koodi stiili kontrollimine tähendab, et kirjutatud kood käivitatakse vastavate tööriistadega, mis kontrollivad, et kõik keskkonna muutujad ning kirjutatud funktsioonid vastaksid süntaksile. Ansible koodi on võimalik testida kasutades Ansible Lint tarkvara. Ansible Lint kontrollib automaatselt, et kirjutatud kood vastaks YAML süntaksile ning koodis poleks tehtud vigu, mis võivad välja kutsuda muutujaid, mida ei eksisteeri. Stiili kontrollimine ei kontrolli koodi funktsionaalsust vaid kindlustab koodi loetavuse ja vastavuse stiili- ning kvaliteedinõuetele.

Koodi üksiku funktsionaalse objekti testimise korral käivitatakse testitavad objektid isoleeritud keskkonnas ja kontrollitakse nende funktsionaalsust. Kuna infrastruktuuri kood ei koosne ainult ühest failist, vaid mitmetest failidest kus igal koodi osal on oma funktsionaalne eesmärk, siis testitakse väiksemate koodi osade kaupa nende funktsionaalsust. Ühiktestimine võtab vähem aega, kui terviku testimine. See võimaldab vigadele jälile saada koodi kirjutamise varasemates staadiumites.

Koodi integratsiooni testimine infrastruktuuri koodi testimise kontekstis tähendab kirjutatud koodi paigaldamist proovikeskkonda. Proovikeskkond on üldiselt võimalikult sarnane toodangu keskkonnale, mis võimaldab testida, kas kirjutatud kood toimib ka toodangukeskkondades. Terviktestimine tähendab, et testitakse kirjutatud moodulite funktsionaalsust. Tervik testid võtavad tavaliselt kaua aega. [27]

Infrastruktuuri koodi testimine aitab arendajatel valideerida kirjutatud koodi kvaliteedinõudeid ja funktsionaalsust. Käesoleva töö raames autor koodi testimist ei juurutanud.

Kokkuvõte

Käesoleva lõputöö eesmärk oli leida lahendus, mis hoiaks kokku finantsettevõtte andmetöötlusplatvormi arendajate ja haldajate tööd platvormi halduselt ning tarkvarade paigaldamise ja uuendamise pealt ning vastaks ettevõtte turvapoliitikale.

Töö esimeses osas tutvustas autor DevOps kultuuri ning infrastruktuuri koodina haldamise põhimõtteid. Lisaks kirjeldas autor, milliseid lahendused sobiksid probleemi lahendamiseks. Töö teises osas analüüsiti esimeses osas kirjeldatud tarkvarasid ning leiti, et kõike efektiivsem ning kasutajasõbralikum oleks juurutada ettevõttes Ansible lahendus. Töö kolmandas osas kirjeldas autor kahe, ettevõtte andmetöötlusplatvormil kasutatava tarkvara paigalduse ja uuendamise protsessi, testimist ja võimalikke tulevikuvõimalusi. Samuti kirjeldas autor, kuidas automatiseerimis protsess muudab arendajate tööd efektiivsemaks ja veakindlamaks.

Autori hinnangul vastab bakalaureusetöö tulemus töö alguses püstitatud nõuetele ning ettevõtte kriteeriumitele. Töökäigus juurutatud lahendus võimaldab ettevõttel säästa raha ning arendajate aega. Lisaks on tänu töö analüüsi osas olevale võrdlusele võimalik mõista, millist lahendust oleks mõistlik juurutada kui on soov automatiseerida kolmanda osapoolte tarkvarade paigaldamise ja uuendamise protsessi.

Kasutatud kirjandus

[1] A. Scopatz ja K. D. Huff, Effective Computation in Physics. Lk 351, 2015.

[2] B. Ingerson, C. C. Evans ja O. Ben-Kiki, „Yet Another Markup Language (YAML) 1.0,“ YAML, 10 Detsember 2001. [Võrgumaterjal]. Link: https://yaml.org/spec/history/2001-12-10.html. [Kasutatud 12 Aprill 2020].

[3] „DevOps,“ AKIT, [Võrgumaterjal]. Link: https://akit.cyber.ee/term/8926-devops. [Kasutatud 13 Aprill 2020].

[4] P. Debois, „Agile 2008 Toronto: Agile Infrastructure and Operations Presentation,“ 9 Oktoober 2008. [Võrgumaterjal]. Link: http://www.jedi.be/blog/2008/10/09/agile-2008-toronto-agile-infrastructure-and-operations-presentation/. [Kasutatud 13 Aprill 2020].

[5] „What Is DevOps?,“ New Relic, [Võrgumaterjal]. Link: https://newrelic.com/devops/what-is-devops. [Kasutatud 13 Aprill 2020].

[6] „Devops,“ Gartner Glossary, [Võrgumaterjal]. Available: https://www.gartner.com/en/information-technology/glossary/devops. [Kasutatud 13 Aprill 2020].

[7] A. Wittig ja M. Wittig, Amazon Web Services in Action lk 93, Manning Press, 2016.

[8] C. Reiley, „Version your infrastructure,“ Devops, 12 November 2015. [Võrgumaterjal]. Link: https://devops.com/version-your-infrastructure/. [Kasutatud 13 Aprill 2020].

[9] „A brief overview of Bitbucket,“ Atlassian, [Võrgumaterjal]. Link: https://bitbucket.org/product/guides/getting-started/overview#a-brief-overview-of-bitbucket. [Kasutatud 12 Aprill 2020].

[10] K. Kawaguchi, „Use Hutson,“ Hutson, [Võrgumaterjal]. Link: https://web.archive.org/web/20090207151650/http://hudson.gotdns.com/wiki/display/HUDSON/Use%2BHudson#UseHudson-License. [Kasutatud 14 Aprill 2020].

[11] M. Maughan, „An Interview with Ansible Author Michael DeHaan,“ 17 Aprill 2012. [Võrgumaterjal]. Link: https://web.archive.org/web/20121114031927/http://www.coloandcloud.com/editorial/an-interview-with-ansible-author-michael-dehaan/. [Kasutatud 12 Aprill 2020].

[12] J. Novet, „Red Hat is buying Ansible for more than $100M,“ Venturebeat, 15 Oktoober 2015. [Võrgumaterjal]. Link: https://venturebeat.com/2015/10/15/source-red-hat-is-buying-ansible-for-more-than-100m/. [Kasutatud 12 Aprill 2020].

[13] J. Robbins, „Announcing Chef,“ Chef Blog, 15 Jaanuar 2009. [Võrgumaterjal]. Link: https://blog.chef.io/announcing-chef/. [Kasutatud 13 Aprill 2020].

[14] Saurabh, „What Is Chef? – A Tool Used For Configuration Management,“

Eureka, 22 Mai 2019. [Võrgumaterjal]. Link: https://www.edureka.co/blog/what-is-chef/. [Kasutatud 13 Aprill 2020].

[15] B. Crist, „Introducing the New Chef: 100% Open, Always,“ Chef Blog, 2 Aprill 2019. [Võrgumaterjal]. Link: https://blog.chef.io/chef-software-announces-the-enterprise-automation-stack/. [Kasutatud 13 Aprill 2020].

[16] „Crunchbase Puppet Company,“ Crunchbase, [Võrgumaterjal]. Link: https://www.crunchbase.com/organization/puppet-labs#section-overview. [Kasutatud 13 Aprill 2020].

[17] „Github release,“ HashCorp, 5 Märts 2014. [Võrgumaterjal]. Link: https://github.com/hashicorp/terraform/releases. [Kasutatud 13 Aprill 2020].

[18] „HasiCorp Github releases,“ Github, 19 Märts 2020. [Võrgumaterjal]. Link: https://github.com/hashicorp/terraform/releases. [Kasutatud 18 Aprill 2020].

[19] „Red Hat Ansible Automation Platform,“ Red Hat, [Võrgumaterjal]. Link: https://www.ansible.com/products/automation-platform. [Kasutatud 13 Aprill 2020].

[20] „Ansible plugin,“ Jenkins, [Võrgumaterjal]. Link: https://plugins.jenkins.io/ansible/. [Kasutatud 13 Aprill 2020].

[21] „Puppet plugin,“ Jenkins, [Võrgumaterjal]. Link: https://plugins.jenkins.io/puppet/. [Kasutatud 13 Aprill 2020].

[22] N. Jain, „Chef vs Puppet vs Ansible,“ [Võrgumaterjal]. Link: https://www.whizlabs.com/blog/chef-vs-puppet-vs-ansible/. [Kasutatud 14 Aprill 2020].

[23] Flexera, „State of the cloud 2019,“ Flexera, 2019.

[24] „Ansible Documentation,“ Redhat, [Võrgumaterjal]. Link: https://docs.ansible.com/ansible/latest/index.html. [Kasutatud 14 Aprill 2020].

[25] J. Wallen, „How to install Apache Tomcat on Ubuntu Server 16.04,“ TechReplublic, 21 Märts 2017. [Võrgumaterjal]. Link: https://www.techrepublic.com/article/how-to-install-apache-tomcat-on-ubuntu-server-16-04/. [Kasutatud 15 Aprill 2020].

[26] J. D. Davidson, „What was the history of Tomcat inside Sun before it was open-sourced?,“ Quora, 14 November 2014. [Võrgumaterjal]. Link: https://www.quora.com/What-was-the-history-of-Tomcat-inside-Sun-before-it-was-open-sourced. [Kasutatud 15 Aprill 2020].

[27] S. J. Bigelow, „Infrastructure-as-code testing strategies to validate a deployment,“ Search ITOperations, 15 November 2019. [Võrgumaterjal]. Link: https://searchitoperations.techtarget.com/tip/Infrastructure-as-code-testing-strategies-to-validate-a-deployment. [Kasutatud 16 Aprill 2020].

Lisa 1 – Jenkins'i kood Ansible juhtimiseks

```
pipeline {
    agent any

    parameters {
            string(name: 'branch', defaultValue: 'master', description: 'Pass
a git branch name')
            choice(name: 'target_environment', choices: ['(choose
environment)', 'development','test-active', 'test-dr', 'production-2-active',
'production-2-dr'], description: 'Select environment for running the
pipeline.')
            choice(name: 'playbook', choices: ['(choose playbook)', 'qlik-
replicate-build-image','qlik-replicate-run-container', 'qlik-replicate-
remove-container', 'qlik-replicate-remove-image'], description: 'Select
playbook.')
            choice(name: 'host', choices: ['(choose host)', 'edge1','edge2',
'edge3', 'edge4', 'edge5', 'edge6'], description: 'Select host.')

    }

    stages {
        stage('Checkout') {
            steps {
                git branch: params.branch,
                credentialsId: 'jenkins',
                url: 'ssh://git@git.foo.bar:7999/odl/cmf.git'
            }
        }

        stage("Run Playbook"){
            steps {
                ansiblePlaybook(
                    credentialsId: "ansible_credentials",
                    installation: 'ansible-default',
                    inventory:
"automation/ansible/inventories/${params.target_environment}/hosts",
                    playbook:
"automation/ansible/utils/${params.playbook}.yml",
                    vaultCredentialsId:
"${params.target_environment}_private_key_vault_password",
                    extraVars: [
                        J_host_variable: "${params.host}"
                        ]
                )
            }
        }
```

```
    }

    post {
        always {
            cleanWs()
        }
    }
}
```

Lisa 2 – Tomcat serveri käivitamise ja sulgemise kood

```
- set_fact:
    tomcat_started_msg: "Tomcat started."
    tomcat_start_stop: "{{tomcat_start_stop}}"

- name: check if instance is running
  pids:
    name: "apache-tomcat-{{tomcat_name}}"
  register: tomcat_pid
  become: true
  become_user: "{{tomcat_admin_user}}"

- block:
  - name: start or stop tomcat instance
    shell: |
      /sb/env/app/tomcat/apache-tomcat-
{{tomcat_name}}/bin/{{tomcat_start_stop}}.sh
    register: tomcat_started
    become: true
    become_user: "{{tomcat_admin_user}}"
    become_flags: "-i"

  - name: Processing the {{tomcat_start_stop}} request on Tomcat-
{{tomcat_name}}
    pause:
      seconds: 15

  - fail:
      msg: "Tomcat was not started!"
    when: "tomcat_started_msg not in tomcat_started.stdout and 'startup' in
tomcat_start_stop"

  - name: check if instance is still running
    pids:
      name: "apache-tomcat-{{tomcat_name}}"
    register: tomcat_pid2
    become: true
    become_user: "{{tomcat_admin_user}}"
    when: "'shutdown' in tomcat_start_stop"

  - name: kill instance
    shell: |
      kill -9 {{item}}
    become: true
    become_user: "{{tomcat_admin_user}}"
```

```
    with_items: "{{tomcat_pid2.pids}}"
    when: "'shutdown' in tomcat_start_stop and tomcat_pid2.pids != ''"

  - name: Processing the KILL request on Tomcat-{{tomcat_name}}
    pause:
      seconds: 5
    when: "'shutdown' in tomcat_start_stop and tomcat_pid2.pids != ''"

  - name: check if instance is running
    pids:
      name: "apache-tomcat-{{tomcat_name}}"
    register: tomcat_pid3
    become: true
    become_user: "{{tomcat_admin_user}}"
    when: "'shutdown' in tomcat_start_stop"

  - fail:
      msg: "Tomcat was not shutdown!"
    when: "'shutdown' in tomcat_start_stop and tomcat_pid3.pids != ''"

  when: "'shutdown' in tomcat_start_stop and tomcat_pid.pids != '' or
'startup' in tomcat_start_stop and tomcat_pid.pids == ''"

- debug:
    msg: "This Tomcat is already running!"
  when: "'startup' in tomcat_start_stop and tomcat_pid.pids != ''"

- debug:
    msg: "This Tomcat is already shutdown!"
  when: "'shutdown' in tomcat_start_stop and tomcat_pid.pids == ''"
```