TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Jürgen Lätte 175721IDAR

TELEKOMIETTEVÕTTE SERVERITE KESKHALDUSLAHENDUSE VÄLJATÖÖTAMINE

Bakalaureusetöö

Juhendaja: Kaido Kikkas
Doktor

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: Jürgen Lätte

18.05.2020

Annotatsioon

Käesolevas diplomitöös on vaatluse all ühes Eesti telekomiettevõttes olevad protsessid. Käsitluse alla tulevad töövõtted hõlmavad endast igapäevaseid haldustegevusi, mida süsteemiadministraatorid teevad ettevõtte infrastruktuuri haldamiseks. Töö eesmärgiks on valida võrdluse abil tööriist, millega on võimalik igapäevaseid tööülesandeid automatiseerida.

Töö sissejuhatavas osas on tutvustatud ettevõtet üldiselt ning paika pandud nõuded, mis esitatakse tulevasele tööriistale. Järgnevalt on kirjeldatud erinevaid keskhaldusvahendeid ning võrreldud nende eripärasid ja sarnasusi. Viimases etapis testitakse valituks osutunud lahendusi.

Töö tulemusena valitakse üks keskhaldusvahend, mis lihtsustab ja automatiseerib süsteemiadministraatori tööd. Uus tööriist vastab püstitatud nõuetele ning aitab infotehnoloogia üksusel olla parem ja kiirem tugi ettevõtte äri poolele.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 4 peatükki, 2 joonist, 4 tabelit.

Abstract

Development of a Central Management Solution for Servers at a Telecom Enterprise

In this diploma thesis the author is looking into work processes of a telecommunications enterprise in Estonia. The selection of the topic of the thesis was mainly driven by the author's desire to improve the life of systems administrators and automate their daily tasks as much as possible and also to have a more sustainable managed infrastructure.

The thesis is divided into 4 main parts: an overview of the enterprise, a comparison of tools needed for centrally managing an infrastructure and testing of the few selected software solutions. In the overview the author describes the organization and the requirements for a solution needed.

The middle section of the thesis focuses on the different aspects of central management systems and explains in detail the pros and cons of each one. The last part of the thesis focuses on testing the selected software and choosing one final solution.

The result of the thesis is a decision on a central management solution that will be further tested and implemented in the current IT infrastructure. The selected new tool matches the requirements set and helps IT to be a better partner for the business development unit.

The thesis is in Estonian and contains 24 pages of text, 4 chapters, 2 figures, 4 tables.

Lühendite ja mõistete sõnastik

API	Rakendusliides. Arvuti operatsioonisüsteemi või rakendusprogrammiga määratud reeglistik, mille alusel rakendusprogramm kasutab operatsioonisüsteemi või teise rakendusprogrammi teenuseid.
Ansible	Tarkvara, mille abil on võimalik automatiseerida ja hallata võrku ühendatud arvutite konfiguratsiooni.
Chef	Tarkvara, mille abil on võimalik automatiseerida ja hallata võrku ühendatud arvutite konfiguratsiooni.
Controller	Juhtmasin, mis haldab teisi sõlmi.
Cron	Unixil (sh Linux ja BSD) põhinevate operatsioonisüsteemide utiliit, millega määratakse tööde ajakava töötlemiseks tulevikus
Docker	Avatud lähtekoodiga platvorm konteinerrakenduste loomiseks, juurutamiseks ja haldamiseks.
Github	Koodihoidla
GUI	Graafiline kasutajaliides.
IP	Interneti põhiprotokoll.
ESXi	Hüperviisor, mis paigaldatakse vastava serveri peale.
Kubernetes	Avatud lähtekoodiga konteineri orkestreerimisesüsteem, millega saab automatiseerida rakenduste juurutamist ja haldamist.
Puppet	Tarkvara, mille abil on võimalik automatiseerida ja hallata võrku ühendatud arvutite konfiguratsiooni.

Playbook	Reeglistik, mis sarnaselt skriptile täidab ülesandeid etteantud loogika järgi.
Python	Interpreteeritav objektorienteeritud programmikeel.
Ruby	Programmeerimiskeel, mida muu hulgas kasutatakse ka Chef keskhaldusvahendis süsteemide seadistuste kirjeldamiseks.
Saltstack/Salt	Tarkvara, mille abil on võimalik automatiseerida ja hallata võrku ühendatud arvutite konfiguratsiooni.
Salt Master ja Salt Minion	Ülem-alluv süsteem.
Shell	Kest. UNIX süsteemiperekonna termin, mille all mõistetakse operatsioonisüsteemi välimist kihti ehk kasutajaliidest, mis korraldab kasutaja ja süsteemiga suhtlemist.
Snapshot	Seisu/oleku salvestamise võimalus, mis võimaldab tagasipöördumist eelmise oleku juurde.
SSH	UNIX süsteemiperekonnapõhine käsuliides ja protokoll, mis võimaldab turvalist sisselogimist kaugarvutisse.
Sudo	Programm Unix tüüpi arvuti operatsioonisüsteemide jaoks, mis võimaldab kasutajatel vaikimisi superkasutajana käivitada programme teise kasutaja turbeõigustega.
Vault	Ansible funktsioon, mis võimaldab turvaliselt hoiustada tundlikku infot krüptitud failides.
VMware	Hüperviisor, mille peal on võimalik teenindada virtuaalseid servereid.
YAML	YAML on inimesesõbralik andmete jadaesituse standard kõigi programmeerimiskeelte jaoks. Seda kasutatakse tavaliselt konfiguratsioonifailides ja rakendustes, kus andmeid hoitakse või edastatakse.

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Tänasel päeval, kus arvutusvõimsus ja serverite hulk on üha kasvav, on süsteemiadministraatoril järjest enam tööd, et hoida kõik süsteemid ajakohased ja serverid turvalised. Süsteemihalduri tegemiste lihtsustamiseks on nüüdisajal loodud mitmeid vahendeid.

Käesolev diplomitöö keskendub keskhalduslahenduse valikule ning valitud lahenduse juurutamisele ühes Eestis tegutsevas telekomiettevõttes. Töö eesmärgiks on leida sobiv lahendus, millega saaks tsentraalselt hallata mitmeid servereid korraga.

Teema on ettevõtte jaoks aktuaalne, kuna tänasel hetkel ei ole firmas kasutusel Linuxi serverite jaoks ühtset keskhaldussüsteemi. Ühtne keskhaldussüsteem lihtsustab süsteemiadministraatori tööd ning muudab selle efektiivsemaks korduvate tegevuste puhul. Näide korduvast haldustegevusest on uue konfiguratsiooni või tarkvarauuenduste paigaldamine.

Diplomitöö jaguneb neljaks osaks. Esimeses osas teeb autor ülevaate ettevõtte vajadustest keskhaldusvahendi järele. Töö teises osas antakse ülevaade võimalikest keskhalduse võimalustest, mis autori tehtud uurimustöö alusel on olemas Linux serverite administreerimiseks. Kolmandas osas tehakse haldusvahendite võrdlev analüüs ja vastav valik edasiste sammude jaoks. Töö neljandas osas käsitletakse tarkvara paigaldamist ja testimist vastavatel teenusserveritel, et näha, milline variant sobib näidisettevõttesse kõige paremini.

Lõputöö üheks lähtetingimuseks on olemasolev *Unixi laadsete (sealhulgas Linux ja BSD) serverite taristu, millel pole ühtset keskhaldussüsteemi. Teisalt on oluline silmas pidada, et kasutusele võetav töövahend oleks avatud lähtekoodiga ning eelarvest lähtudes ei tohiks olla rahalisi väljaminekuid.

2 Ettevõttest ja nõuded süsteemile

2.1 Ettevõtte taustsüsteem

Diplomitöö autor alustas vaatluse all olevas telekomiettevõttes tööd süsteemi-administraatorina 2018. aastal. Sel ajahetkel sai käesoleva töö kirjutaja enda vastutusalasse ligikaudu 100 serverit, millest enamus on virtuaalsed. Osad neist serveritest toetavad üksteist ja on üsna sarnase konfiguratsiooniga. Arhitektuurilised muudatused on tihti toonud kaasa väga palju manuaalset tööd serverites, sest serverite seadistus on olnud samalaadne. Serverites konfiguratsioonifailide käsitsi muutmine suurendab hooletusvigade esinemist.

Selleks, et tulevikus vältida võimalikult palju käsitsitööd ja võimalikke teenusekatkestusi, tuleks kasutusele võtta ühtne keskhaldussüsteem, mis sobiks ennekõike virtuaalserverite haldamiseks, mida peaks olema võimalik rakendada ka füüsilistele masinatele. Diplomitööga seotud telekommunikatsiooni ettevõttes on Linuxi administraatoreid 5 ning servereid umbes 1000. Sealjuures on serverite kogus erinevate teenuste ja äri kasvu tõttu pigem suurenemas kui vähenemas.

2.2 Töö lähtetingimused

Koostöös infrastruktuuri juhi ja teiste süsteemiadministraatoritega lepiti kokku järgmised nõuded soovitavale lahendusele:

- Keskhaldussüsteem peab aitama lihtsamini teele saata konfiguratsiooni või rakenduste muudatusi.
- Tööriist peab võimaldama saada täpset ülevaadet masinate olukorrast, erinevatest konfiguratsioonidest või rakendustest.

- Kasutusele võetav instrument peab aitama kaasa paremale kasutajate haldamisele – kuvama kasutajate õiguste infot, võimaldama kiiresti luua ja kustutada kasutajaid ning hallata kasutajate õigusi ja nendega seotud informatsiooni.
- Administreerimisvahend peab võimaldama kiiresti paigaldada turvauuendusi. Turvapaikade rohkus on tingitud sellest, et erinevaid turvaauke avastatakse võrdlemisi palju [1].
- Tarkvara kasutamise ja haldamise kogemused peavad olema administraatorile kergelt omandatavad. Sealjuures oleks kindlasti abiks hea dokumentatsioon.
- Tegemist peab olema vabavaralise programmiga. Lisaks ei ole ette nähtud eraldi investeerimisvahendeid tarkvara kasutuselevõtuks.
- Keskhalduslahendust peab olema võimalik laiendada ilma et tekiksid täiendavad kulud arendusele.
- Kasutusele võetaval tarkvara peab toetama võimalust tõmmiste ja mallide tegemiseks VMware keskkonnas. Samuti peab tööriist võimaldama tagasi pöördumist tehtud tõmmise juurde.
- Otsitav keskhaldusvahend peab võimaldama automatiseerimist. Näiteks rakenduste või turvauuenduste automaatset uuendamist.
- Tööriistal peab olemas olema toetav ning aktiivne kogukond ning on oluline, et tarkvara on toetatud arendustiimi poolt ning lahendatakse esinenud tarkvara probleeme ja turvaauke.
- Arvestama peab emaettevõttes kasutatavate protsessidega ning seal olevate töövahenditega.
- Täiendavalt võiks kasutusele võetav keskhalduslahendus toetada erinevaid konteinersüsteeme.
- Lisaks *Unix (sealhulgas Linux ja BSD) operatsioonisüsteemi toele, võiks vastava vahendiga vajadusel olla võimalik hallata ka Windowsi masinaid.

Kõiki neid kriteeriume arvesse võttes, asuski diplomitöö autor uurima, milliseid võimalusi olemas on ning milline võiks kõige paremini sobida antud ettevõttesse.

3 Analüüs

3.1 Ülevaade keskhaldusvahenditest

Vastust küsimusele, milline on parim keskhalduslahenduse tarkvara UNIX-laadsete (muuhulgas Linux ja BSD) serveri süsteemide jaoks, väga lihtsalt ei leia. Lahendus taandub sellele, mis on kõige parem variant otsijale. Ettevõtte vajadus oli ennekõike vabavaraliste tööriistade järgi. Märksõnadest kõige enam tulid esile järgnevad neli [2,3,4,5]: Chef, Puppet, Ansible ja Saltstack, Lisaks tuleb ühe märksõnana välja ka Cfengine. Lugedes, miks tudeng Noormäe [2] oma töös Cfengine´ist loobus, ei pidanud ka käesoleva diplomitöö autor oluliseks minna edasi antud tööriista analüüsimisega. Noormäe sõnul on Cfengine`i õppimine ja uuendamine liiga keeruline. Lisaks on Noormäe [2] arvates Cfengine` näol pigem tegemist raamistikuga hulgale *shelli* skriptidele kui haldusvahendiga.

3.1.1 Saltstack

Saltstacki või lühemalt Salti Githubi lehekülg on mõeldud igas suuruses infrastruktuuri või rakenduse halduse ja konfigureerimise automatiseerimiseks [6]." Colton Myers kirjeldab raamatus [7] Saltstacki, kui avatud lähtekoodiga projekti, millele pani alguse Thomas Hatch 2011. aastal. Myersi sõnul [7] on vaja teada seda, et Saltstacki arhitektuur koosneb kahest peamisest komponendist. Nendeks on Salt Master ja Salt Minion. Käesoleva töö kontekstis tähendab see seda, et tegemist on klient-server arhitektuuriga, kus Master on serveri rollis ja Minion kliendi rollis. Raamatu autori sõnul on Salt lihtsaim viis servereid valitseda. Saltstackiga on kirjaniku enda sõnul võimalik keskselt saata käsurealt käske või ülesandeid tuhandetele serveritele korraga. Haldusvahend on kirjutatud Pythoni programmeerimiskeeles ning Colton Myersi kirjutatu kohaselt [*Ibid*] peaks olema võimalik konkreetset tööriista lihtsasti kohandada vastavalt oma vajadustele.

Salt ei võimalda tasuta versioonis kasutada ei graafilise liidese ega ka *SecOps* mooduleid [8]. Kui ilma graafilise liideseta võiks veel toimetada, siis raporteerimine ning S*ecOps*

moodul turvalisuse ning vastavuse kontroll on need põhjused, miks võiks kaaluda tasulist versiooni [8]. Tutorialspoint [9] toob välja selle, et Saltstacki tasuline versioon maksab orienteeruvalt 150$ masina eest aastas. Tootja enda kodulehel [10] on välja toodud, et kommertslahendus toetab erinevaid platvorme nagu näiteks VMware, Docker ja Kubernetes. Ehkki Colton Myersi kirjutatu [7] põhjal on Salti kõige suuremat potentsiaali võimalik ära kasutada klient-server lahenduse puhul, siis on võimalik ka [10] klienditu või API-põhine lahendus oma tootmiskeskkonna haldamiseks.

3.1.2 Puppet

Teine sarnane keskne administreerimise vahend on Puppet. Franceschi [11] toob välja selle, et Puppet on konfiguratsioonihalduseks ja automatiseerimiseks mõeldud tööriist, mis on kirjutatud Ruby programmeerimiskeeles. Autori sõnul saab seda platvormi kasutada, et juhtida erinevate operatsioonisüsteemidega servereid nagu näiteks Linux, Solaris, BSD, MacOS ja Windows. Turnbull [12] kirjeldab, et Puppet arhitektuur põhineb klient-server mudelil nagu ka Saltstack. Raamatus on mainitud fakti, et Puppetile pani aluse Luke Kanies. Tuuakse veel välja [*Ibid*], et kuna Puppet on avatud lähtekoodiga, siis oskuste olemasolul saab igaüks vajadusel koodi muuta või täiustada. Lisaks on tugev kogukond, kes pidevalt toodet arendab selliselt, et erinevaid koodilõike või -pakette saab lisada oma vajadustest lähtuvalt. Näiteks on Turgulainen [5] aastal 2010 pidanud kõige sobivamaks lahenduseks IT Kolledži arvutipargi haldamise seisukohalt just Puppetit. Tema arvates oli toona see valik parem Cfengine ´ist ja Chefist just seetõttu, et Puppetil oli tol ajahetkel olemasolev veebiliides, laialdlane konfiguratsioonifailide valmiskogu ning lai kasutajate baas.

Firma enda kodulehel on kirjas [13], et Puppetil on olemas VMware tugi ja ühtlasi ka võimekus konteinersüsteemide haldamiseks. Sarnaselt Saltstackile on Puppetil olemas nii tasuta kui ka tasuline versioon [9]. Sama veebileht toob välja, et kui võtta kommertseesmärgiga tööriist, siis tuleb tasuda ligikaudu 100$ masina eest aastas. Vabavaralise vahendi paigaldamisel tuleb siis loobuda järgnevast [14]:

- raportite koostamisest ja standardsusest
- graafilisest veebiliidesest
- toest koormusjaoturitele

- paigaldamist toetavatest moodulitest
- orkestreerimise API-st
- reaalajalisest visualiseerimisest
- virtuaalmasinate haldamise võimekusest (konteinerlahenduste tugi on vabavaralises versioonis olemas)
- osast koodi haldamise võimekusest
- nii sõlme eluea haldamise võimalustest kui ka rollipõhisest juurdepääsu kontrollist
- ettevõtte toest ja koolitusetst

Seega on Puppeti puhul pikk loetelu, millest vabavaralise versiooni kasutamisel tuleb loobuda.

3.1.3 Chef

Chef on klient-server tüüpi lahendus nagu ka eelmised kaks tööriista – Saltstack ja Puppet. Taylor ja Vargo [15] peavad Chefi ennekõike konfiguratsiooni haldusvahendiks. Veel toovad raamatu kirjutajad Chefi eelistena välja järjepidevuse, muutuste juhtimise, serverite taasehitamise lihtsuse ning jälgitavuse. Nad rõhuvad sellele, et Chefi puhul on mõeldav, et infrastruktuuri erinevaid versioone saab nummerdada. Taristut ennast on võimalik lihtsasti uuesti üles ehitada ning koheselt ka testida. Taylor ja Vargo [15] mainivad, et Chefi arendusvahendid on kirjutatud Rubys. Nende sõnul vähendab see arendusele minevat aega ning ühtlasi tagatakse erinevate arendusplatvormide toetus. Rahman [16] rõhutab eraldi seda, et kui tahta rakendada Chefi, siis oleks hea, kui seda teeks inimene, kellel on vähemalt mingi arusaam tootmiskeskkonnast. Rakenduste puhul oleks kasulik, kui arendustiim kirjeldaks ära nõuded, mida vajatakse.

Upguard [17] toob välja, et Chefil on klientidele pakkuda põhimõtteliselt kolme hinnastusmudelit. Tasuta vabavaraline pakett ning kaks tasulist paketti – pilvepõhine ja kliendi enda serverites asuv. Tasuta versioonil on puudu näiteks graafiline liides, analüütika töölaud ja kohandatavad vaated. Lisaks pole ettevõttel võtta tuge mujalt, kui kogukonnalt endalt. Sama artikkel [*Ibid*] viitab ka sellele, et on olemas tõsised kahtlused

selles osas, kas Chef Inc tegelikult tahab jätkata avatud lähtekoodiga tarkvara arendamist. Seda väidet toetab ka Chef Inci enda poolt välja antud artikkel [18], kus nad toovad välja, et nad muudavad oma toodete litsentseerimise tingimusi ning keskenduvad pigem korporatiivklientidele. Upguard [17] kirjutab, et oma infrastruktuurile üles pandav seadistus on keeruline, kuna peab väga hästi taristut tundma. Veelgi enam, iseenda infrastruktuuril kogu süsteemi käigus hoida ilma tootjapoolse toeta, maksab aastas põhimõtteliselt sama palju, kui hoida süsteeme teenusepakkuja enda pilves koos toe ja koolitustega.

3.1.4 Ansible

Ansible sai alguse 2012. aastal ning alusepanijaks peetakse Micheal DeHaani [19]. Ansible kasutab YAML faile info edastamiseks. YAML meenutavat oma olemuselt JSONI või XML-i. Ansible ise on kirjutatud Pythoni baasil. Raamatu autori [19] sõnul on see suureks eeliseks, sest Pythoni kasutamine ei too kaasa sõltuvusi teistest tarkvarapakettidest. SSH ühendust saab kasutada autentimismeetodina ja taustal ei pea eraldi tegutsema ketta- ja täitmismonitorid. Hochstein [20] toob lisaks välja põhjuseid, miks on hea Ansible’t kasutada. Nimelt klient-masinas ei ole vaja midagi paigaldada (on vaja lihtsalt Python pakettide olemasolu), saab ise kiiresti peale lasta uuendusi, võimalus kohandada ka väiksematele masinaparkidele ja tuleb kaasa oma moodulitega [20].

Ansible on saadaval nii tasuta kui tasulise versioonina. Tasulise versiooni puhul algab tugi alates 100$ masin ja kallineb vastavalt sellele, millist tuge vajatakse [9]. Ansible’i puhul tuleb, aga silmas pidada seda, et projekt ise on jagunenud kaheks Ansible Awx ja Red Hat Ansible Tower. Kui vaadata Red Hat korporatsiooni kodulehel [21] välja toodud erinevusi, siis taandub kogu otsustusprotsess peamisel kahele asjale - stabiilsus ja tugi. Tuleb arvestada seda, et avatud lähtekoodiga tasuta versiooni arendatakse pidevalt edasi ning kõiki uusi muudatusi ei pruugita jõuda testida. Ühtlasi puudub, nii nagu teistegi haldusvahendite puhul, otsene tugi toodet arendavalt ettevõttelt ning alati ei saada viimaseid turvauuendusi. Viimane tingib selle, et peab olema valmis kiiremini kasutusele võtma uusimaid tarkvara versioone. Seevastu mainib Hochstein [20] , et Ansible on avatud lähtekoodiga programm ja seda saab lihtsalt kasutada ka käsurealt. Ta räägib sellest, et näiteks Ansible Tower on veebipõhine keskkond Ansible’i haldamiseks. Tuuakse välja seda, et enamus kogukonna poolt kirjutatud mooduleid on lihtsasti kättesaadavad ning kasutatavad.

3.1.5 Keskhalduslahenduse tarkvara lahenduste kokkuvõte

Kõige keerulisem kõikide platvormide juures on adekvaatselt hinnata kogukonna tuge. Kuigi enamikul artiklite kirjutajatel on olemas oma arvamus, millist tööriista kasutada või eelistada, ei ole sageli välja toodud erinevate kommuunide toetusega seonduvat või näiteks toetuse enda puudumine. Yevgeniy Brikman [22] toob oma artiklis välja numbrilised näitajad, mida vaadata.

Tabel 1 Tarkvarade kogukondade võrdlus aastate lõikes. Koostanud:Autor

	Aasta	Aktiivsed panustajad	Koodi versiooni haldusesse laadimine	Teegid	Stack Overflow postitused	Tööpakkumised Indeed.com lehel
Saltstack	2016	2 237	608	318	1 062	1 622
	2019	3 132	1 088	423	1 837	5 792
Puppet	2016	515	94	6 110	3 585	4 200
	2019	613	112	8 432	4 876	3 402
Chef	2016	562	435	3 832	5 982	4 378
	2019	663	1 040	4 828	8 554	3 415
Ansible	2016	4 386	506	20 677	11 746	8 787
	2019	8 553	754	53 140	37 940	19 771

Tabel 1 on koostatud Brikman [22] poolt välja toodud andmetel. Kolme aasta jooksul on Ansible suutnud aktiivsete panustajate arvu kasvatada 195%. Samal ajal, kui Saltstacki aktiivsete panustajate kasv jääb pigem 40% ligidusse ning Puppeti puhul on kasv 19 ja Chef on kasvatanud vastavat numbrit ligikaudu 18%. Sealjuures on Saltstacki koodi versiooni haldusesse laadimine kasvanud 79%, Puppetil 19%, Chefil 239% ja Ansible ´il

49%. Tabelis 1 tulevad suuremas erinevused sisse aga teekide osas. Nimelt Saltstacki teekide arv on kasvanud 33%, Puppetil on vastava näitaja kasvanud 38% ning Chefil on kasv olnud 26% ja Ansible´l 257%. Kasvanud on ka Stack Overflow postituste arv. Nimelt Saltstackil on tõusu 73%, Puppetil vastavalt 36%, Chefil 43% ja Ansible´l 323%. Lahknevused ilmnevad, aga töökuulutuste arvu osas. Brikman [22] toob välja, et indeed.com lehekülje järgi on tööpakkumiste arv kolme aastaga vähenenud Puppetil 19% ja Chefil 22%. Samas tõusu on näidanud nii Saltstack 357% kui ka Ansible 225%. Nende viimaste numbrite järgi on vajadus eelkõige Saltstack ja Ansible spetsialistide järele.

Tabelis 2 on kokku võetud diplomitöö autori poolt kokku kogutud info erinevate tarkvaraliste lahenduste kohta. Koostatud tabel annab võrdlusmomenti otsustajatele ja ühtlasi on ka indikaatoriks sellele, millistele algtingimustele tööriistad vastavad.

Tabel 2 Erinevate keskhaldussüsteemide võrdlus. Koostanud: Autor

	Saltstack	**Puppet**	**Chef**	**Ansible**
Õppimiskõver (1)	Keskmine	Keeruline	Keeruline	Lihtne
Keel, mida kasutatakse haldamiseks	YAML (2)	Ruby	Ruby	Python
Võimalik hallata UNIX (sh Linux ja BSD) süsteeme, Windowsi süsteeme	Jah	Jah	Jah	Jah
Tugi konteinersüsteemide haldamisseks	Jah	Jah	Jah	Jah
VMware tugi	Jah	Jah	Jah	Jah
Tõmmiste ja mallide tegemise võimalus VMware’ s	Jah olemas	Jah olemas (5)	Jah olemas (9)	Jah olemas

	Saltstack	**Puppet**	**Chef**	**Ansible**
Võimalik koostada raporteid riist – ja tarkvara kohta.	Jah	Jah	Jah	Jah
Kasutajate haldamise võimekus	Jah	Jah	Jah	Jah
Kogukondlik tugi	Olemas (3)	Olemas (6)	Olemas (8)	Tugev (10)
Tasuta/Tasuline	Mõlemad variandid olemas (4)	Mõlemad variandid olemas (7)	Mõlemad variandid olemas (9)	Mõlemad variandid olemas (11)

Lisakommentaarid Tabelis 2 oleva info kohta:

1. Hinnang antud diplomitöö autori poolt tuginedes erinevatele artiklitele ja foorumipostitustele.
2. Saltstacki GUI on Pythoni baasil [8].
3. Saltstacki kogukond kasvab andmete järgi mõõdukalt. Aktiivsete panustajate arv on kolme aastaga suurenenud 40% võrra [22].
4. Saltstackil on olemas väiksema funktsionaalsusega tasuta versioon [8]. Kommertslikuks kasututuseks mõeldud tarkvara maksab aastas 100 klienti hinnaga 15 000$ aastas [9].
5. Puppeti puhul on olemas VMware’ s tõmmiste tegemiste moodul ainult Puppet Enterprise ´ga [14].
6. Puppeti puhul on kogukondliku toe kasv pigem aeglane. Aktiivsete panustajate arv on kolme aasta jooksul suurenenud 19% [22].
7. Puppeti puhul on olemas väiksema funktsionaalsusega tasuta versioon [14]. Kommertslitsentsiga tarkvara 100 klienti hinnaga 10 000$ aastas [9].
8. Chefil on kogukondlik tugi olemas, aga edasine kogukondlik panus kahtluse all [17]. Kasv kolme aastaga on olnud aktiivsete panustaje osas 18% [22].

9. Chefil, nagu eelmistelgi haldusvahenditel, on tasuta versioonil oluliselt väiksem funktsionaalsus ja tugi. Kommertsversioon on alates 7 200$ aastas 100 klienti[14].
10. Ansible'i tööriista jaoks on panustajate arv teistest oluliselt kõrgem. Nimelt on kasv olnud kolme aastaga lausa 195% [22].
11. Ansible'i haldusvahendi tasuta versioonil on olemas täisfunktsionaalsus [21]. Tasuline tugi alates 10 000$ aastas 100 klienti [9].

3.2 Metoodika

Hea ülevaade olemasolevatest infotehnoloogilistest ressurssidest aitab aimu saada, kus parasjagu analüüsi hetkel ollakse, kuhu saab vajadusel edasi minna ning mis on tänased takistused ning riskid, mis ei pruugi lähitulevikus võimaldada äri korraldada nii, nagu seda seni on tehtud. Lacy ja Norfolk [23] toovad välja, et juba eelmisel kümnendil oli IT teenuse pakkujatel vaja silmas pidada järgnevat:

- Teenuste arhitektuur on muutumas.
- Suureneb fookus turvalisusele ja riskide juhtimisele.
- Komplekssetel teenustel ja süsteemidel on vaja teha sagedamini mõjuanalüüse.
- Vajadus pidevalt balansseerida stabiilsuse ja muutuste vahel.

Selleks, et valida õige tööriist, peab silmas pidama mitmeid punkte. Lacy ja Norfolk [23] toovad välja kolm olulist asjaolu.

- Esmalt tuleb teostada turu-uuring, mille tulemusena valitakse välja vaid mõningad teenusepakkujad.

- Järgmiseks tuleb hinnata töövahendit ennast, otsides ja küsides võimalikult palju informatsiooni. Olles saanud vajaliku teabe, tuleb teha analüüs, kas paarisvõrdluse meetodil või MoSCoW meetodil.

- Viimase sammuna kogu protsessis, tuleb alustada pilootjärguga ehk tuleb testida võimalikke tarkvarasid.

MoSCoW meetod on tähtsuse järjekorda seadmise tehnika, mis aitab vajadustest paremini aru saada [24]. Veebilehel [24] välja toodud info kohaselt on see metoodika kõige parem just juhtimaks inimeste ootuseid projekti valmimise osas.

Tabelis 3 välja toodud analüüs on teostatud kasutades MoSCoW meetodit.

Tabel 3 Moscow meetodi kasutamine. Koostanud: Autor

Peab olema (Must have)	• Võimekus hallata *UNIX tüüpi süsteeme. • Võimalus hallata kasutajad ning erinevaid konfiguratsioone. • Võimalik riist- ja tarkvaralise info kogumine. • Lai kogukondlik tugi. • Tasuta versioonil peab olema võimalikult sarnane funktsionaalsus tasulisele versioonile. • Tööriista kasutuselevõtt peab olema võimalikult lihtne.
Peaks olema (Should have)	• Tõmmiste ja mallide tegemise võimekus.
Seekord võiks olla (Could have)	• -
Seekord ei ole (Won´t have this time)	• Käesoleva töö raames jääb välja võimekus hallata Windowsi süsteeme. • Ühtlasi jäävad vaatluse alt välja ka konteinersüsteemide haldamise testimine

3.3 Lahenduste võrdlev testimine

Valiku tegemine konkreetsema keskhaldusvahendi suhtes sõltub kokklepitud tingimustest ja eeldustest. Üheks kõige olulisemaks parameetriks on ettevõtte enda kontekst. Kui lähtuda ainult nõudmistest (MoSCoW tabeli alusel), siis on ettevõtte jaoks kõige sobilikum keskhalduse tarkvara Ansible. Peamised põhjused sellise valiku tegemiseks on järgnevad:

- Tööriista on lihtne kasutama õppida.
- Vabavaraline versioon on piisavalt võimekas, ehk ei ole vajadust kommertslitsentsi soetamiseks.
- Konkreetset tööriista kasutatakse lõputöös käsitletava telekomi emaettevõttes. Seega on mingisugused teadmised ja oskused olemas ning paika pandud protsesse on võimalik Skandinaavia kogemusest üle võtta.
- Tööriista tasuta versioon sisaldab teadaolevalt kõige enam funktsionaalsust võrreldes teiste võrdluses olevate toodetega.
- Kõige suurem kogukonna kasv ning ka haldusvahendi koodi arengusse panustamiste arvuline tõus.

Ansible vastab kõigile töö lähtetingimustele. Seejuures jälgides MoSCoW meetodit saab kaetud kõik vajalik. Sellise tarkvara kasutuselevõttu toetab ka 2016. aastal Dmitri Tsumaki poolt kaitstud diplomitöö [3]. Tsumak on vaadanud ennekõike veebilehe paigaldamist Wordpressi ja monitooringu lahenduse paigaldamist Zabbixi näol, kuid oma töö vaadeldavas ja võrdlevas osas tõstab ta Ansible ettepoole nii Puppetist, Chefist ja Saltstackist. Ainsaks negatiivseks asjaks Ansible'i puhul mainib Tsumak seda, et pole võimalik pidada järge selle üle, mis versiooniga *playbook*'ist parasjagu tegu on. Lahendusena on välja pakutud see, et enamus ettevõtteid kasutab koodihoidlad, et pidada järge ajaloo ja muudatuste osas. Negatiivse poole lahendamiseks teeb autor ettepaneku juurutada koodihoidla kasutamine näidisettevõttes. Valikut, milline operatsioonisüsteem valida Ansible Controller masina jaoks, ei olnud keeruline teha. Shah [25] näited põhinevad enamasti Ubuntul ning ka Myers [7], Heap [19] ja Hochstein [20] on toonud üsna palju näited Ubuntu baasil. Näidete paljususe tõttu tundus diplomitöö autorile just see operatsioonisüsteem kasutusele võtmiseks olevat kõige sobivam. Shah [25] on välja toonud miinumnõuded, mida masinale võiks eraldada. Nõuded uuele masinale on: 2 protsessorit, 2 GB muutmälu ja 20 GB jagu kettaruumi. Lisaks langes diplomitöö autori enda valik operatsioonisüsteemi puhul Ubuntu 18.04 LTS kasuks.

Testimise jaoks loodud masin sai nime ansible-controller.domeen.ee ning IP 10.10.10.5. Lisaks sai testvõrku pandud 3 Ubuntu 16.04 LTS masinat, kus peal töötab MySql andmebaasi klaster. Nimetused määrati järgnevalt jdb1.domeen.ee, jdb2.domeen.ee,

jdb3.domeen.ee ning vastavad IP-d 10.10.10.6, 10.10.10.7 ja 10.10.10.8. Ainult Shah [25] ja Geerling [26] toovad oma näidetes välja, et Ansible tuleks pigem paigaldada ametlikust koodihoidlast. Teised autorid on samas oma näidete jaoks laadinud Ansible'i arenduses oleva versiooni otse Githubi koodihoidlast. Diplomitöö autor otsustas stabiilsust silmas pidades laadida see alla ametlikust koodihoidlast ning mitte paigaldada kõige viimast eksperimentaalset versiooni.

Ansible'i tarkvara paigaldamiseks käivitas diplomitöö autor käsud kasutaja ansiblecontroller alt vastavalt Geerlingu [26] näpunäidetele:

```
$ sudo apt-add-repository -y ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install -y ansible
```

Seejärel kontrollis lõputöö kirjutaja üle, milline versioon sai alla laetud ning käsule

```
$ ansible –version
```

tuli vastuseks: `$ ansible 2.9.4.`

Shah [25] toob välja, et vaikimisi luuakse *hosts* fail /etc/ansible/hosts asukohta. Muuta tuleb vaid konfiguratsioonifaili sisu, mis meie jaoks tähendab järgnevat olukorda:

```
[all]
jdb1.domeen.ee
jdb2.domeen.ee
jdb3.domeen.ee
```

Loetud teoste põhjal eeldatakse paroolita serveritesse sisselogimise võimalust. SSH ühendus sai hallatavatesse serveritesse tekitatud nii nagu on välja toodud Linuxsize [27] näites. Enne seda sai tekitatud sudo õigustega kasutaja ansiblecontroller, mis on sama kasutaja, kes on *Master*rollis oleva masina peal. See on vajalik selleks, et saaks vajalikke käske käivitada hiljem ühest kesksest kohast. Lähtudes juhendist [28], sai muudetud reeglistikku [Lisa 1] vastavalt oma parameetritele, et saaks lisada kasutaja serverisse koos kasutaja enda ssh võtmega ning samas eemaldades sisse logimise võimaluse parooliga.

Selleks, et käsurealt saaks reeglistikku käivitada, on vaja, et asutakse kaustas

```
$ /home/ansiblecontroller/ansible_playbooks
```

Käsk

```
$ ansible-playbook add.user-ssh.yml -i
/etc/ansible/hosts
```

käivitab vastava reeglistiku ning paigaldab *sudo* õigustega kasutaja devops kõigile kolmele serverile. Skripti toimivust saab kontrollida, kui logida vastava kasutajaga sisse näiteks masinasse *jdb2.domeen.ee* ja vaadata, et /etc/ssh/ kaustas on tehtud koopia eelmisest konfiguratsioonist märkega

sshd_config.28943.2020-03-10@10:10:36

Kui on soov eemaldada kasutaja *ssh* ligipääs, siis on seda võimalik teha lihtsalt, muutes näidet [29] nii nagu ta on välja toodud Lisa 2 all. Selleks kasutame käsku

```
$ ansible-playbook remove-user-ssh.yml
```

kus skript eemaldab vastava näite puhul masinast jdb2.domeen.ee kasutaja *ssh* võtme.

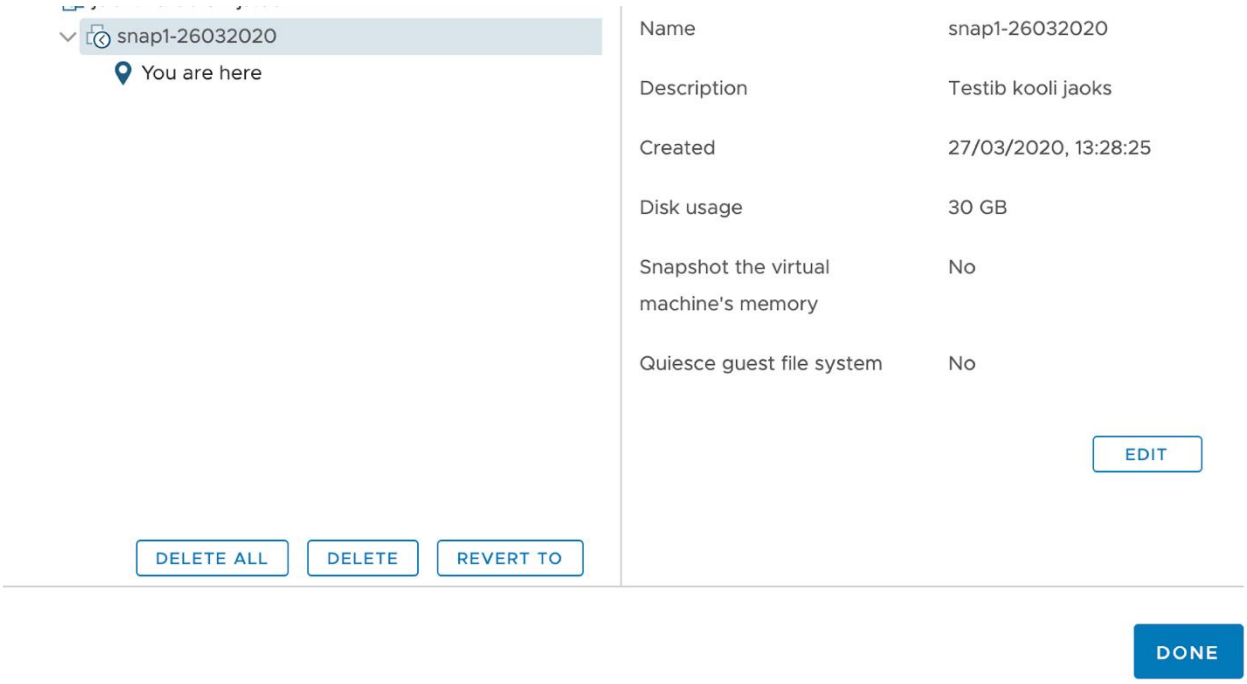
Kui on soov saada üldist infot operatsioonisüsteemi kohta kindlal serveril, siis võib käivitada käsu:

```
$ ansible jdb3.domeen.ee -m setup | less
```

mis siis annab väljundi vastava serveri parameetrite [Lisa 3] kohta. Lisas 4 on välja toodud serveril jdb3.domeen.ee olevad teenused, mille jälgimisest on diplomitöö autor huvitatud. Väljavõtte puhul kõige olulisemateks ehk jälgimist vajavaks, peab käesoleva töö kirjutaja järgmist: **mysql, networking, rsyslog, ssh ja ufw** teenuseid. Reeglistik, millega kontrollida oluliste teenuste seisu, on välja toodud Lisas 5. Väljund, mis reeglistiku käivitamisest tuli, on välja toodud Lisas 6. Väljavõttest selgub, et kontrollitakse, kas teenus toimib või mitte. Keerukust, mis hõlmaks teenuste automaatset käivitamist ja teenusnimekirja läbikäimist, on plaanis diplomitöö autoril aja jooksul juurde lisada.

Selleks, et oleks peale muudatuste tegemist võimalus pöörduda tagasi esialgse muutmata oleku juurde, tuleb kasutada VMware võimalust nimega *snapshot*. Selleks, et meie

ansible-controller server pääseks esmalt VMware' le ligi, peab jälgima juhendit [30] ja alles siis saab hakata kasutama reeglistikku [31], millega saaks hetkeolukorra tõmmiseid teha. Diplomitöö autor testis skripte [Lisa 7, 8, 9], mis kõik toimisid ja pakuvad võimalust hallata tõmmiseid VMware keskkonnas. Enne seda, kui neid skripte käivitada, peab ESXi serveri peal paika panema kasutaja vastavate õigustega, nagu on ära toodud VMware [32] enda dokumentatsioonis. Turvalisuse huvides ei ole vaja anda kõiki õiguseid kasutajale, kes haldab hetkeolukorra tõmmiseid.



Joonis 1. VMware keskkonnas tehtud hetketõmmis. Allikas: Autor

Jooniselt 1 võib välja lugeda, et tõmmise tegemine toimis. Siin ja edaspidi tuleb meeles pidada seda, et see toimis käsurealt, kuna oli teada vastav ESXi serveril, kuhu peale tehakse tõmmis. Sealjuures, kui tagasiminek [Lisa 8] toimus sama serveri peale, siis kustutada ettevõtte infrastuktuuris enam sama serveri pealt ei saanud. Nimelt liigutas VMware enda automaatika masina teise ESXi serveri peale ning Ansible'i reeglistiku toimimiseks tuli ka seal muuta vastavaid kirjeid. Ettevõtte kontekstis peab meeles pidama, et on palju ESXi servereid, mistõttu käsurealt tõmmiste tegemine ei pruugi olla nii aegasäästev kui võib-olla lihtsalt veebiliidest kasutades. Turvapaikade ja rakenduste uuenduste automaatseks paigaldamiseks kasutatakse värskelt paigaldatud masinat *jdb4.domeen.ee* , mille peal töötab operatsioonisüsteem Ubuntu 16.04 ning mis koheselt tahab saada 157 tarkvara paketi uuendust, millest 105 on turvauuendused. Muuhulgas

tuleb ära uuendada vana *kerneli versioon 4.4.0-142-generic*. Turvauuenduste ja ka näiteks uue tarkvara paigaldamine Apache näol, on ära toodud [Lisas 10]. Siinkohal peab meeles pidama seda, et seda skripti võib kasutada ka koos [Lisas 7] välja toodud skriptiga ning ajastada automaatselt käima näiteks *cron*'iga. Põhjus, miks kombineerida omavahel [Lisa 7] ja [Lisa 10] skripte on see, et mõningate rakenduste toimivust peab kontrollima käsitsi ning peab olema võimalus pöörduda tagasi eelneva ning varasemalt toimiva olukorra juurde. Seega tänasel päeval võib öelda, et Ansible katab ära põhivajadused ettevõtte jaoks, aga arvestama peab ettevõtte infrastruktuuriga seotud eripärasid alustades rakendustest ja lõpetades ESXi serveritega.

Teisalt on vajalik aru saada, kuidas või kui keeruliselt tegelikult töötab teine tarkvara samades oludes. Seetõttu otsustas diplomitöö autor testkeskkonnas üles panna ka Saltstack haldusvahendi. Nimelt on see oma kogukonna kasvult teisel kohal hetkel analüüsitavatest haldusvahenditest. Üldjoontes vastab Saltstack ka etteseatud lähtetingimustele. Ainuke vastuolu, mis tähelepanu köidab, on väide, et seda tööriista on keerukam kasutama õppida kui teisi. Selle miinuse on välja toonud Tsumak [3]. Lisaks toob ta miinustena antud tööriista puhul välja selle, et puudub versiooni halduse võimekus, ametliku dokumentatsiooni järgimine on keeruline, esineb probleeme mitte *UNIX- laadsete (sealjuures Linux ja BSD) platvormide haldamisega ning isegi kui kasutada GUI-d, siis see ei ole optimeeritud ega intuitiivne kasutada. Täiendavalt peab käesoleva diplomitöö autor mainima seda, et Ansible'ga võrreldes on Saltstacki tasuta versioon kommertstootest rohkem kärbitud.

Vastavalt Colton Myersi [7] soovitusele, paigaldab diplomitöö autor Saltstacki kontrolliva osa Salt-Masteri Ubuntu 18.04 LTS platvormile ning esialgseks testimiseks kasutame masinaid *jdb1-jdb4*, kuhu paigaldatakse Salt-Minioni kliendid. Samal ajal, kui Salt-Masteri paigaldamine juhendi järgi [33] läks peale õige operatsioonisüsteemi valimist võrdlemisi lihtsalt, siis Salt-Minioni paigaldamisega jäid ikkagi üles tarkvaralised sõltuvused, mis ei lasknud korrekselt programmi paigaldada. Seetõttu otsustas diplomitöö autor valida teise juhendi [34] ja nii sai ka klient paigaldatud vajalikesse masinatesse. Teise juhendi järgmine on ka sellepoolest lihtsam, et kui ei olda teadlik, millisele operatsioonisüsteemile tarkvara parasjagu paigaldatakse, siis õpetuses olev skript otsib ise õige versiooni üles ja paigaldab vajalikud programmijupid.

Erinevalt Ansible'st ei ole Saltstacki puhul vaja võimaldada ligipääsu port 22-le. Nimelt kui programm paigaldada Salt-Master serveril, siis Ubuntu [7] puhul aktiveeritakse automaatselt pordid 4505 ja 4506, mille kaudu hakkab suhtlus toimima. Teistel opratsioonisüsteemidel tuleb tulemüüridest eraldi liiklus vastavatele portidele lubada. Lisa sammuna tuleb seadistada vastavates serverites ***/etc/salt/minion*** *ja* ***/etc/salt/master*** konfiguratsioonid, et liiklus hakkaks serverite vahel käima. Eeldusel, et serverid suhtlevad omavahel, tuleb veel Salt-Master serveril lubada Salt-Minion klientidel ühenduda *Masteriga*. Seda saab teha kui uurida käsuga ***salt-key***, mis kliendid üldse olemas on. Käsk

```
$ salt-key -a jdb1.domeen.ee
```

aktsepteerib selle kliendi võtme *Master* serveris ja nüüd on viimasest võimalik esimesse juba erinevaid käskluseid saata.

Soovides saada ülevaadet näiteks ühest *minionist,* võib käivitada käsu

```
$ sudo salt 'jdb4' grains.items
```

Saame sellise väljundi nagu on [Lisa 11]. See väljund annab esialgset infot masina üldisema seisundi kohta. Selleks, et teada saada, millised teenused mingis masinas olemas on, saab kasutada käsku

```
$ salt 'jdb4.domeen.ee' service.get_all
```

Nii nagu Lisas 12 lühendatult näha võib. Teenuseid võib samuti kontrollida sama mooduli abil. Näiteks

```
$ salt 'jdb4.domeen.ee' service reload apache2
```

abil saab taaskäivitada apache2 programmi.

Peale mõnda aega otsimist, leidis käesoleva diplomitöö kirjutaja efektiivse viisi, kuidas kasutajat tekitada koheselt koos SSH võtmega. Aluseks sai võetud [35] kood, mida sai muudetud vastavalt [Lisa 13] oma vajadustele. Lisast 13 võib järeldada seda, et kui kasutaja loomiseks võtmega peab looma skripti vaikimisi faili ***/srv/salt/users/*** alla ning käivitama selle käsuga

```
$ salt 'jdb4.domeen.ee' state.apply users kasutaja123
```

siis eemaldamine on vastupidiselt lihtsam.

Näiteks käsuga

```
$ salt '*' ssh.rm_auth_key kasutaja123 *
```

saab eemaldada kõikidest *Minion*-serveritest antud kasutaja kõik SSH võtmed. Peale selle, kui on soov hallata kindlaid faile, siis näidete [36] järgi tuleb lihtsalt muuta faile, mis asuvad *Master* serveris. Üks võimalik variant on järgnev:

```
deploy the http.conf file:
    file.managed:
      - name: /etc/http/conf/http.conf
      - source: salt://apache/http.conf
```

See koodilõik otsib üles *Master* serveris asuvast kaustast */srv/salt/apache/http.conf* konfiguratsiooni faili ja käivitamisel kontrollitakse, et *Minion* serveris olev fail oleks samasugune nagu peaarvutis olev fail. Juhul, kui nii pole, siis võetakse juhtarvutis olev fail ja asendatakse see uuema versiooniga konfiguratsioonist.

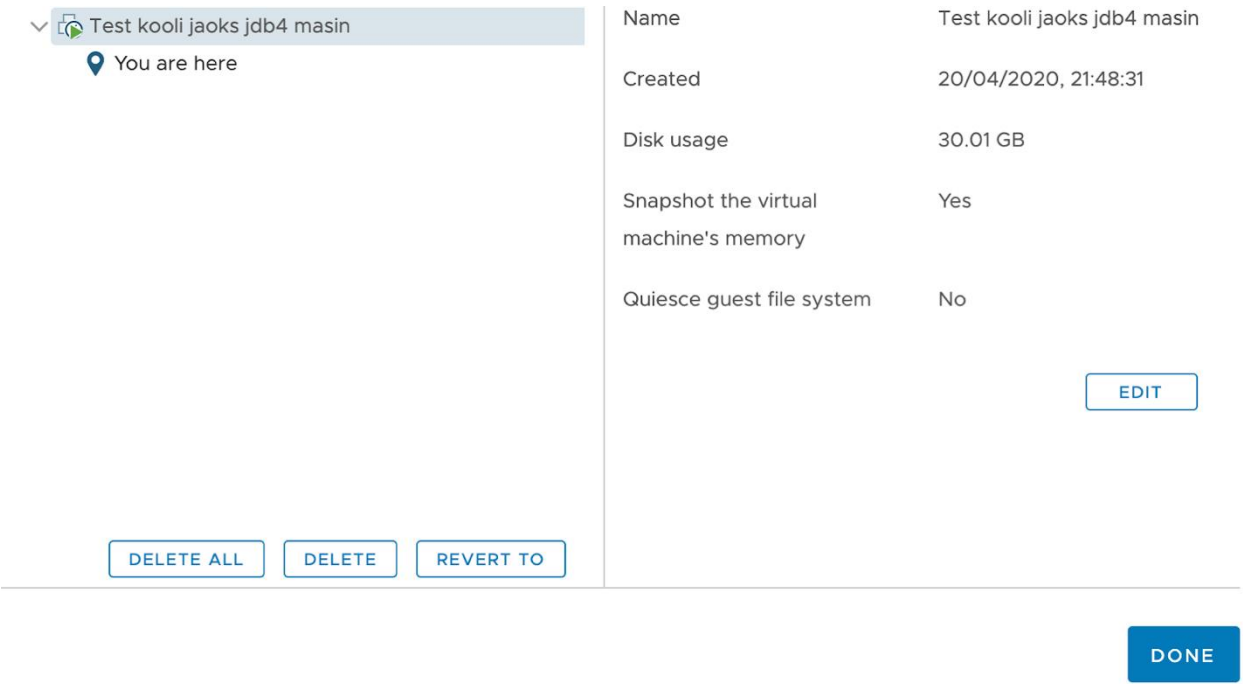
Suhtlemiseks VMware keskkonnaga piisab ametliku dokumentatsiooni [37] jälgimisest. Nimelt selleks, et suhelda tõmmistega on vaja, et *Master* serveri failis ***/etc/salt/cloud.providers.d/VMware.conf*** oleks kirjas minimaalselt järgnevad read:

```
vcenter01:
  driver: VMware
  user: 'DOMAIN\user'
  password: 'verybadpass'
  url: 'vcenter01.domain.com'
```

Oluline siinjuures on silmas pidada seda, et vastaval kasutajal peaks olema turvalisuse mõttes ainult tõmmiste tegemise õigused VMware keskkonnas. Käsuga

```
$ salt-cloud -a create_snapshot jdb4
snapshot_name="Test kooli jaoks jdb4 masin"
[description="jdb4 Test kooli jaoks"]
[memdump=False] [quiesce=True]
```

luuakse vastavast virtuaalmasinast hetketõmmis.



Joonis 2. Saltstackiga tehtud hetketõmmis. Allikas: Autor

Seda näitab ka Joonis 2. Lisast 14 saame näha ka lühikest väljundit, mis tuleb peale mõnda aega ootamist, kuna tõmmise tegemine võtab aega. [Lisad 15 ja 16] koos käskude ja väljunditega annavad mõista, et tõmmistega toimetamine on võrdlemisi lihtne. Juhul, kui on soov mõne tööriistaga automatiseerida ka operatsioonisüsteemi uuendusi, siis peale [Lisa 14]. oleva käsu käivitamist tuleks käivitada käsk

```
$ salt 'jdb4' pkg.upgrade
```

mis siis uuendab ära masinas vastavad tarkvarad. Siinkohal tuleb siiski rõhutada, et sellist uuendamist tasub teha ainult läbiproovitud ja testitud masinatega. Tänase lahenduse juures on miinuseks see, et rakenduse toimivust peab ikkagi käsitsi kontrollima.

4 Lahenduse kasutusele võtmine

Diplomitöö eesmärgiks oli tutvustada uut lahendust ettevõttele infrastruktuuri paremaks haldamiseks ning läbi selle paremaks muuta ja kiirendada oma infrastruktuuri haldamist. Peale erinevate lahenduste toimivuse demonstreerimist töökollektiivile ja lõpuks ühisele otsusele jõudes, et edasi minnakse Ansible keskhaldusprogrammiga, jääb ainult üle see kasutusele võtta.

Edukuse mõõdupuuks võttis diplomitöö autor ajalise säästu, mis peaks tulenema automatiseerimise võimekusest. Seda enam, et käsitsi asjade tegemine peaks kõrvale jääma ning tänases olukorras peaks inimese kontrolli nõudma ainult rakenduse toimimine peale uuenduste tegemist. Irina Ivanova [4] oma töös on kasutanud stopperit mõõtmaks protsessi kestvuse aega. Sama meetodit saab kasutada käesolevas diplomitöös, et määrata erinevused käsitsi tööde tegemise ja sama tööde automatiseeritud variantide vahel. Vaatluse alla lähevad kasutaja loomine, kasutaja ligipääsu eemaldamine, hetketõmmise tegemine ja uuenduste paigaldamine.

Tabel 4 Protsesside ajaline võrdlus Ansible'iga. Koostanud: Autor

Protsess	**Käsitsi tehtav töö**	**Automatiseeritud töö**	**Tegevused keskmiselt 1 kuus**	**Säästetav aeg**
Kasutaja loomisele kuluv aeg	119s	28s	2 kasutajat 5. masinasse	910s
Kasutaja ligipääsu eemaldamine	32s	28s	1 kasutaja 10. masinast	40s

Protsess	**Käsitsi tehtav töö**	**Automatiseeritud töö**	**Tegevused keskmiselt 1 kuus**	**Säästetav aeg**
Hetketõmmise tegemine	22s (159*s)	65s (103*s)	30 tk	420s*
Uuenduste paigaldamine	45s	36s	30 tk	270s

Tabelist 3 selgub, kui palju ajalist säästu iga protsess annab. Olgu mainitud, et diplomitöö autor ei arvesta ajavõtmise hulka seda aega, mis kulub ansible-controller masinasse, vastavasse serverisse või VMware keskkonda sisselogimiseks. Aeg läheb käima sellest hetkeks, mil protsess algab ja lõppeb sellel hetkel, kui tehtud tööl on mõõdetav tulemus olemas.

Testid said tehtud ühe masina peal ja sealt edasi tuletatud ajaline sääst keskmiste tegevuste pealt ühe kuu jooksul. Kasutaja loomise katse tagajärjel selgus, et ühte masinasse ühe kasutaja loomine automaatselt annab 91 sekundit säästu. Sealjuures, kui võtta see arv keskmiselt ühe kuu tegevuste peale, siis on ajaline sääst veelgi suurem. Säästmist võib Tabelis 3 näha ka kasutaja ligipääsu eemaldamisel. Erinevus on tervelt 4 sekundit automaatika kasuks. Huvipakkuv asjaolu selgus hetketõmmiste tegemise juures. Nimelt koheselt ühe masina peal testimisest võitu ei olnud. Parem tulemus diplomitöö autorile tuli välja siis, kui teha tõmmis rohkem kui ühest masinast korraga. Käesolevas näites on tehtud test nelja masinaga, et näha, kas mingit ajalist kokkuhoidu tekib. Ühele masinale taandatuna on võimalik suuremate hulkade korral võita 14 sekundit masina kohta. Juhul, kui on ainult 1 masin, siis tasub tõmmist teha pigem käsitsi, sest ajaline erinevus on ligi 3 korda. Ajaline võit selgub ka uuenduste paigaldamisel, kui erinevus ühe masina kohta on 9 sekundit. Hetkel tuleb ainult nende nelja protsessi automatiseerimise pealt ajalist säästu 1640s, mis on ligikaudu 27 minutit ja 20 sekundit. Terve aasta peale tulev võit on ligilähedaselt 328 minutit, mis omakorda on 5 tundi 30 minutit.

Statistikaameti [38] andmeil oli 2019 aasta süsteemiadministraatori mediaan brutopalk ühes kuus 2091 eurot. Arvestades, et 2019 aastal oli keskmiselt tehtavad töötunnid ühes

kuus [39] 168 tundi. Saab tuletada, et mediaan tunni hinnaks aastal 2019 oli ligikaudu 12.45 eurot. Diplomitöö autor saab tuletada ettevõttele tuleva rahalise säästu, kui võtta arvesse ajaline võit ligikaudu 5 tundi ja 30 minutit ning ligikaudne tunnihind 12.45 eurot. See säästa nende protsesside automatiseerimise eest aastas oleks umbkaudselt 68.45 eurot. See arv kehtib muidugi hetkel ühe süsteemihalduri kohta. Arvestades seda summat tiimi peale, siis tuleb säästetavaks summaks 342.25 eurot. Leidmaks tööandja kulu palgafondile kasutab diplomitöö autor Internetis leiduvat tööriista palgakalkulaatorit [40]. Palgakalkulaatori järgi oleks 2019. aastal olnud selle summa palgafond ehk ettevõtte kulu kokku ligikaudu 457.93 eurot, mida on võimalik meeskonna peale säästa. Lisaks ei saa mainimata jätta, et numbriliselt ei saa väljendada neid tegevusi, mida süsteemiadministraator saab teha samal ajal, kui automaatsed tegevused taustal toimetavad.

Diplomitöö autor ei saa siinkohal lugeda teemat lõpetatuks. On tehtud algus sellele, et automatiseerida ja muuta efektiivsemaks süsteemiadministraatorite tööd. Juurutamisega edasiliikumiseks on vaja läbi mõelda vastavad asjaolud:

- Tuleb tekitada ühtne koodihoidla süsteemiadministraatoritele, et saaks hõlpsasti ning järgitavalt skripte jagada ja muuta.
- Standardiseerida süsteeme, et sarnase sisuga konfiguratsiooni failid asetseksid masinates samades asukohtades.
- Tuleb testida ja kasutusele võtta Ansible Vault [41], et turvaliselt saaks hoida tundlikke andmeid krüptitud failides ja mitte tavalise tekstina mõne skripti sees. Näited sellistest andmetest on paroolid ja SSH võtmed.
- Peab testkeskkonnast ja testserveritelt edasi liikuma ning haldama hakata tootmiskeskkonnas olevaid servereid
- Tarvilik on konsulteerida osakondadega, et läbi rääkida, kuidas keskhaldusvahend aitaks nende tööprotsesse lihtsustada ja automatiseerida. Sealjuures kuidas automaatsete uuendusprotsesside juures tagada see, et rakendused peale uuendusi töötavad. Veelgi enam hoolitseda selle eest, et kõikvõimalikud keskkonnad oleksid terviklikud peale muudatusi.
- Lisaks olemasolevatele õppematerjalide ja tasuta saadaolevatele õpetustele, plaanib diplomitöö autor võtta ka erinevaid tasulisi koolitusi, et arendada enda

oskusi ning seeläbi teada ka mis koolitusi või õppematerjale oma kolleegidele soovitada.

Käesolevat diplomitööd ja lisades leiduvad materjale saavad õppimise eesmärgil kasutada kõik, kes on huvitatud oma infrastruktuuri ja igapäevaste tegevuste automatiseerimisest. Diplomitöö edasiarendusena näeb autor, et testitakse näiteks kahe või enama keskhalduslahendusega, kuidas mõnele uuele platvormile ehitada terve suur infrastruktuur alates virtuaalmasinatest ja lõpetades toimivate rakenduste ja monitooringusüsteemidega. Osaliselt Ansible'i puhul on katnud seda teemat Tsumak [3] tehes skriptid Wordpressi ja Zabbixi paigaldamise jaoks, aga siin annakski uudsust võrdlusmoment erinevate keskhaldusvahendite võimekuse osas terve taristu lahendus toimima saada. Lisaks on käesoleva diplomitöö autoril lootus, et ehk võetakse keskhaldusvahendite tutvustamine ka Tallinna Tehnikaülikooli IT Kolledži õppekavasse, et näidata olemasolevaid lahendusi oma tudengitele.

5 Kokkuvõte

Diplomitöö põhieesmärgid jagunesid kolmeks:

1. Tutvustada erinevaid keskhalduslahenduse võimalusi Linuxi serverite administreerimiseks.
2. Teha valik potentsiaalsete keskhaldusvahendite rakendamise osas.
3. Vastava(te) tarkvara(de) paigaldamine ja testimine.

Ülevaate koostamine erinevatest keskhalduse lahenduse võimalustest ei olnud keeruline. Seda ennekõike seetõttu, et on tekkinud erinevad kogukonnad, kes dokumenteerivad erinevaid võimalikke lahendusi ning potentsiaalselt kasutatav info on hõlpsalt raamatute ja interneti abil leitav. Dokumenteerimise tegi lihtsamaks asjaolu, et otsing sai kitsendatud ainult Linuxi süsteemidele.

Teine suurem eesmärk töö juures oli MoSCoW meetodit kasutades välja valida sobivad tarkvarad, mida saaks testida ja millest siis ühte hakata kasutama serverite haldamiseks. Peale pikemat analüüsi jäid sõelale Ansible ja Saltstack.

Autori siht käesoleva teosega oli välja valida sobiv tarkvara keskhalduse jaoks. Peale kahe tarkvara süvitsi testimist võib väita, et on leitud sobiv lahendus, mida saab kasutada telekommunikatsiooni ettevõtte infrastruktuuri edendamiseks. Selleks lahenduseks on Ansible, mis tõendas oma sobivust ettevõtte infrastruktuuri paremini.

Kasutatud kirjandus

[1] Mitre Corporation, [Võrgumaterjal]. Available: https://www.cvedetails.com/product/47/Linux-Linux-Kernel.html?vendor_id=33. [Kasutatud 10 05 2020].

[2] H. Nooremäe, *ARVUTIKLASSI MITMIKKÄIVITUSE KESKHALDUSSÜSTEEMI RAKENDAMINE JÕGEVA GÜMNAASIUMIS,* 2011.

[3] D. Tsumak, *Large-Scale Provisioning and Configuration Management,* 2016.

[4] I. Ivanova, *Versiooniuuenduse automatiseerimine kasutades skriptimiskeelt Bash,* 2016.

[5] K. Turgulainen, *ESTOBUNTU INSTALLATSIOONITÕMMISE ARENDAMINE EESTI INFOTEHNOLOOGIA KOLLEDŽI ARVUTIKLASSI NÄITEL,* 2010.

[6] Saltstack Inc, [Võrgumaterjal]. Available: https://github.com/saltstack/salt. [Kasutatud 13 10 2019].

[7] C. Myers, Learning Saltstack, Second Edition, 2016.

[8] H. M. Hughes. [Võrgumaterjal]. Available: https://eitr.tech/blog/2019/05/31/ansible-vs-salt.html. [Kasutatud 17 11 2019].

[9] Tutorialspoint, [Võrgumaterjal]. Available: https://www.tutorialspoint.com/saltstack/saltstack_competitors.htm. [Kasutatud 17 11 2019].

[10] Saltstack Inc, [Võrgumaterjal]. Available: https://www.saltstack.com/solutions/continuous-compliance/. [Kasutatud 17 11 2019].

[11] A. Franceschi, Extending Puppet, 2014.

[12] J. Turnbull, Pulling Strings with Puppet, 2007.

[13] Puppet Inc, [Võrgumaterjal]. Available: https://puppet.com/docs/discovery/1.x/pd_vsphere_vm.html#reference-6004. [Kasutatud 17 11 2019].

[14] Puppet Inc, [Võrgumaterjal]. Available: https://puppet.com/products/why-puppet/puppet-enterprise-and-open-source-puppet. [Kasutatud 17 11 2019].

[15] M. Taylor ja S. Vargo, Learning Chef : a guide to configuration management and automation, 2015.

[16] Rahman ja N. U, Configuration Management with Chef-Solo, 2014.

[17] Upguard, [Võrgumaterjal]. Available: https://www.upguard.com/articles/chef-open-source-vs-hosted-chef-vs-on-premises-chef. [Kasutatud 20 11 2019].

[18] Chef Inc, [Võrgumaterjal]. Available: https://blog.chef.io/chef-software-announces-the-enterprise-automation-stack/. [Kasutatud 20 11 2019].

[19] M. Heap, Ansible: From Beginner to Pro, 2016.

[20] L. Hochstein, Ansible: Up and Running, 2015.

[21] Red Hat Inc, [Võrgumaterjal]. Available: https://www.redhat.com/en/resources/awx-and-ansible-tower-datasheet. . [Kasutatud 21 11 2019].

[22] Y. Brikman. [Võrgumaterjal]. Available: https://blog.gruntwork.io/why-we-use-terraform-and-not-chef-puppet-ansible-saltstack-or-cloudformation-7989dad2865c#5176. [Kasutatud 21 11 2019].

[23] S. Lacy ja D. Norfolk, „Configuration management : expert guidance for IT service managers and practitioners,“ 2014.

[24] Agile Business Consortium, [Võrgumaterjal]. Available: https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation . [Kasutatud 13 11 2019].

[25] G. Shah, Ansible Playbook Essentials, 2015.

[26] J. Geerling, Ansible for DevOps, 2017.

[27] Linuxsize, [Võrgumaterjal]. Available: https://linuxize.com/post/how-to-set-up-ssh-keys-on-ubuntu-1804/. [Kasutatud 08 03 2020].

[28] A Medium Corporation, [Võrgumaterjal]. Available: https://medium.com/@khandelwal12nidhi/setup-ssh-key-and-initial-user-using-ansible-playbook-61eabbb0dba4. . [Kasutatud 10 03 2020].

[29] [Võrgumaterjal]. Available: https://gist.github.com/trongnghia203/2d720280d74085d2330776274ee2c887. [Kasutatud 13 03 2020].

[30] Red Hat Inc, [Võrgumaterjal]. Available: https://docs.ansible.com/ansible/latest/scenario_guides/vmware_scenarios/vmware_requirements.html. [Kasutatud 26 03 2020].

[31] Red Hat Inc, [Võrgumaterjal]. Available: https://docs.ansible.com/ansible/latest/modules/vmware_guest_snapshot_module.html. [Kasutatud 26 03 2020].

[32] VMware Inc, [Võrgumaterjal]. Available: https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.security.doc/GUID-4D0F8E63-2961-4B71-B365-BBFA24673FDB.html. [Kasutatud 26 03 2020].

[33] Saltstack Inc, [Võrgumaterjal]. Available: https://repo.saltstack.com/#ubuntu. [Kasutatud 14 04 2020].

[34] Saltstack Inc, [Võrgumaterjal]. Available: https://repo.saltstack.com/#bootstrap. [Kasutatud 14 04 2020].

[35] J. Schroeder. [Võrgumaterjal]. Available: https://gist.github.com/SEJeff/2856687. [Kasutatud 17 04 2020].

[36] Saltstack Inc, [Võrgumaterjal]. Available: https://docs.saltstack.com/en/getstarted/config/files.html. [Kasutatud 18 04 2020].

[37] Saltstack Inc, [Võrgumaterjal]. Available: https://docs.saltstack.com/en/latest/ref/clouds/all/salt.cloud.clouds.vmware.html#salt.cloud.clouds.vmware.create_snapshot. [Kasutatud 20 04 2020].

[38] Statistikaamet, [Võrgumaterjal]. Available: https://andmestikud.stat.ee/ametipalk/. [Kasutatud 24 04 2020].

[39] [Võrgumaterjal]. Available: https://regmas.ee/2019-aasta-kalendaarne-tooajafond/.. [Kasutatud 24 04 2020].

[40] Trinity Capital, [Võrgumaterjal]. Available: https://www.kalkulaator.ee/et/palgakalkulaator. [Kasutatud 24 04 2020].

[41] Red Hat Inc, [Võrgumaterjal]. Available: https://docs.ansible.com/ansible/latest/user_guide/vault.html. [Kasutatud 26 04 2020].

Lisa 1 Ansible'i reeglistik , millega paigaldada kasutaja serverisse

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%201%20%20Ansible%20playbook%20%2C%20millega%20paigaldada%20kasutaja (kättesaadav vähemalt kuni 01.09.2020)

Lisa 2 Ansible’i reeglistik , millega eemaldada kasutaja serverisse

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%202%20%20Ansible%20playbook%20%2C%20millega%20eemaldada%20kasutaja%20serverist (kättesaadav vähemalt kuni 01.09.2020)

Lisa 3 Ansible'i käsu väljund operatsioonisüsteemi info kohta

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%203%20%20%20Ansible%20k%C3%A4su%20v%C3%A4ljund%20operatsioonis%C3%BCsteemi%20info%20kohta (kättesaadav vähemalt kuni 01.09.2020)

Lisa 4 Väljavõtte olemasolevatest teenustest server jdb3.domeen.ee peal

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%204%20%20V%C3%A4ljav%C3%B5tte%20olemasolevatest%20teenustest%20server%20jdb3.domeen.ee%20peal (kättesaadav vähemalt kuni 01.09.2020)

Lisa 5 Ansible’i reeglistik teenuste toimivuse kontrollimiseks

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%205%20Ansible%20Palybook%20teenuste%20toimivuse%20kontrollimiseks (kättesaadav vähemalt kuni 01.09.2020)

Lisa 6 Väljund Ansible'i reeglistikust, mis kontrollib teenuste toimivust

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%206%20V%C3%A4ljund%20Ansible%20Playbookist%2C%20mis%20kontrollib%20teenuste%20toimivust(kättesaadav vähemalt kuni 01.09.2020)

Lisa 7 Ansible’i reeglistik hetkeolukorra tõmmise tegemiseks

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%207%20Ansible%20Playbook%20hetkeolukorra%20t%C3%B5mmise%20tegemiseks(kättesaadav vähemalt kuni 01.09.2020)

Lisa 8 Ansible’i reeglistik tõmmise tegemise aega tagasi pöördumiseks

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%208%20Ansible%20Playbook%20t%C3%B5mmise%20tegemise%20aega%20tagasi%20p%C3%B6%C3%B6rdumiseks(kättesaadav vähemalt kuni 01.09.2020)

Lisa 9 Ansible'i reeglistik tõmmise eemaldamiseks

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%209%20Ansible%20Playbook%20t%C3%B5mmise%20eemaldamiseks(kättesaadav vähemalt kuni 01.09.2020)

Lisa 10 Ansible’i reeglistik turvauuenduste paigaldamiseks

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%2010%20Ansible%20playbook%20turvauuenduste%20paigaldamiseks(kättesaadav vähemalt kuni 01.09.2020)

Lisa 11 Saltstacki väljavõte serveri andmete kohta

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%2011%20Saltstacki%20v%C3%A4ljav%C3%B5te%20serveri%20andmete%20kohta(kättesaadav vähemalt kuni 01.09.2020)

Lisa 12 Saltstack väljavõte teenuste kohta

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%2012%20Saltstack%20v%C3%A4ljav%C3%B5te%20teenuste%20kohta(kättesaadav vähemalt kuni 01.09.2020)

Lisa 13 Saltstackiga kasutajaga võtme loomine ja kahel erineval viisil võtme eemaldamine

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%2013%20Saltstack%20kasutajaga%20v%C3%B5tme%20loomine%20ja%20kahel%20erineval%20viisil%20v%C3%B5tme%20eemaldamine
(kättesaadav vähemalt kuni 01.09.2020)

Lisa 14 Saltstacki väljund kuvatõmmise tegemise kohta

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%2014%20Saltstacki%20v%C3%A4ljund%20kuvat%C3%B5mmise%20tegemise%20kohta(kättesaadav vähemalt kuni 01.09.2020)

Lisa 15 Saltstacki väljund tõmmise juurde pöördumise kohta

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%2015%20Saltstacki%20v%C3%A4ljund%20t%C3%B5mise%20juurde%20p%C3%B6%C3%B6rdumise%20kohta(kättesaadav vähemalt kuni 01.09.2020)

Lisa 16 Saltstacki väljund tõmmise eemaldamise kohta

https://bitbucket.org/jlatte/jurgen-latte-diplomitoo/src/master/Lisa%2016%20Saltstack%20v%C3%A4ljund%20t%C3%B5mmise%20eemaldamise%20kohta(kättesaadav vähemalt kuni 01.09.2020)