TALLINNA TEHNIKAÜLIKOOL
INSENERITEADUSKOND
Tartu Kolledž

E-KOOLIKOTI SERVERITE EHITAMINE JA UUENDAMINE

BUILDING AND UPGRADING E-KOOLIKOTT SERVERS

RAKENDUSKÕRGHARIDUSTÖÖ

Üliõpilane: Kristi Ploomipuu

Üliõpilaskood: 178552EDTR

Juhendaja: Sten Aus, taristu büroo juhataja

Kaasjuhendaja: Ago Rootsi, lektor

Tartu 2020

(Tiitellehe pöördel)

AUTORIDEKLARATSIOON

Olen koostanud lõputöö iseseisvalt.
Lõputöö alusel ei ole varem kutse- või teaduskraadi või inseneridiplomit taotletud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

"....." ..... 202…..

Autor: /allkirjastatud digitaalselt/

Töö vastab bakalaureusetöö/magistritööle esitatud nõuetele

"....." ..... 202…..

Juhendaja: /allkirjastatud digitaalselt/

Kaitsmisele lubatud

"....."..... 202… .

Kaitsmiskomisjoni esimees .....
/ nimi ja allkiri /

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina Kristi Ploomipuu (a*utori nimi*) (sünnikuupäev: 15.04.1998)

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
E-koolikoti serverite ehitamine ja uuendamine,

(*lõputöö pealkiri*)

mille juhendaja on Sten Aus,

(*juhendaja nimi*)

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

---

[1] *Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil.*

/allkirjastatud digitaalselt/

_____ (*kuupäev*)

TalTech Tartu Kolledž

LÕPUTÖÖ ÜLESANNE

Üliõpilane: Kristi Ploomipuu, 178552EDTR

Õppekava, peaeriala: EDTR17/17 - Telemaatika ja arukad süsteemid

Juhendajad: Taristu büroo juhataja, Sten Aus
Haridus- ja Noorteamet, 730 2117, sten.aus@harno.ee
Lektor, Ago Rootsi

Lõputöö teema:

(eesti keeles) E-koolikoti serverite ehitamine ja uuendamine

(inglise keeles) Building and upgrading E-koolikott servers

Lõputöö põhieesmärgid:

1. Infosüsteemi viimine uuemale operatsioonisüsteemile
2. Parandada serveri hooldusvõimekust
3. Testida läbi infosüsteemi taastamine
4. Varukoopiate tegemise ajakohastamine

Lõputöö etapid ja ajakava:

Nr	Ülesande kirjeldus	Tähtaeg
1.	Uuendusplaani valmistamine, läbirääkimiste lõpetamine	9.07
2.	Testkeskkonna ehitamine, uuendamine ja testimine	29.08
3.	Toodangukeskkonna ehitamine, uuendamine ja testimine	18.10

Töö keel: eesti keel **Lõputöö esitamise tähtaeg:** "....."..........202….a

Üliõpilane: Kristi Ploomipuu /allkirjastatud digitaalselt/ "....."..........202….a

Juhendaja: Sten Aus /allkirjastatud digitaalselt/ "....."..........202….a

Kaasuhendaja: Ago Rootsi /allkirjastatud digitaalselt/ "....."..........202….a

Kinnise kaitsmise ja/või lõputöö avalikustamise piirangu tingimused formuleeritakse pöördel

SISUKORD

EESSÕNA

Rakenduskõrgharidustöö teema algatajaks on Haridus- ja Noorteameti tehnoloogia juhtimise osakonna taristu büroo, mis enne 2020. aasta augustit kandis nime Hariduse Infotehnoloogia Sihtasutuse struktuuriüksus EENet (Eesti Hariduse ja Teaduse Andmesidevõrk). Rakenduskõrgharidustöö on osa ameti arendus- ja haldustegevustest ning on koostatud Tartus 2018.-2020. aastal.

Lõputöö koolipoolseks kaasjuhendajaks on Tallinna Tehnikaülikooli Tartu Kolledži lektor Ago Rootsi ning ettevõttepoolseks juhendajaks tehnoloogia juhtimise osakonna taristu büroo juhataja Sten Aus.

Töö autor tänab kõiki, kes on olnud töö valmimisel abiks, eelkõige tehnoloogia juhtimise osakonna taristu büroo nii varasemaid kui ka praeguseid kaastöötajaid eesotsas Sten Ausiga, kes kõik olid alati valmis aitama nõu ja jõuga ning lektor Ago Rootsit, kes väsimatult oli nõus abistama ka kesköö ning oma kogemustest rääkides suutis isegi kõige igavamad teoreetilised probleemid huvitavateks pöörata.

Mõistete ja lühendite loetelu

Arenduskeskkond	Keskkond (liivakast), mis on EENetis loodud ainult süsteemiadministraatoritele testimiseks (ingl k *development/ dev server või sandbox*)
AUR	Arch Linux operatsioonisüsteemi pakihalduse repositoorium, mida haldab ja hooldab kogukond
GIT	koodivaramu ehk repositoorium
HTTP	hüperteksti edastusprotokoll (ingl k *Hypertext Transfer Protocol*)
HTTPS	turvaline hüperteksti edastusprotokoll (ingl k *Hypertext Transfer Protocol Secure*)
ISKE	infosüsteemide kolmeastmeline etalonturbe süsteem [1]
Juurkasutaja	süsteemi haldamiseks kõige suuremate õigustega kasutaja (ingl k *root/superuser*)
Nimeviit	teisele failile või kaustale viitav fail (ingl k *symlink/symbolic link*)
Proksi	tarkvara, mis vahendab välisliiklust
SSH	võimaldab turvalist kaugligipääsu võrku ühendatud seadmega (ingl k *Secure Shell*)
Testkeskkond	Keskkond, kus testivad süsteemiadministraatorid koos arendajate ja projektijuhtidega, üldjuhul viiakse muudatused hiljem otse toodangukeskkonda (ingl k *test server*)
Tõmmis	varundusotstarbeline koopia, mis ei sisalda kogu süsteemi sisu

SISSEJUHATUS

Vananenud tarkvara on tänapäeval üheks põhiliseks turvaõnnetuste põhjustajaks ning sellest tulenevalt on ülimalt oluline hoida nii infosüsteemide tarkvarad kui ka operatsioonisüsteem alati ajakohastel versioonidel [2]. Paratamatult on suuremate infosüsteemide serverite ajakohastamine tihti ebameeldiv ja mahukas töö, mida nii mõnedki ettevõtted meeleldi teha ei soovi. Uuendamine võib tuua kaasa mitmeid probleeme, eelkõige vanemate funktsioonide kadumise või asendumisega, mis omakorda võib infosüsteemi tööd häirida või selle üldse peatada, parandamine nõuab uut arendustööd, mis on aeganõudev ja kulukas. Sellegipoolest aitab halbu üllatusi vältida eeluurimine, õigeaegne planeerimine ja kindlasti uuendamine, sest uus arendustöö on küll kulukas, kuid andmeleke kulukam.

Autori valis lõputöö eesmärgiks E-koolikoti serverite ehitamise ja uuendamise, kuna tegemist oli vananenud ülesehitusega serveritega ning seetõttu oli projekt asutuse jaoks küllaltki prioriteetne. Lisaks vastse praktikandina oli eeldatav maht piisavalt pikk, et võimaldada tudengil omandada iseseisvalt Unixilaadsete süsteemide baasteadmised saades kogemusi nii erinevate operatsioonisüsteemide kui ka tarkvarade tööpõhimõtete kohta.

Lõputöö põhieesmärkidest kõige olulisemaks on uuendada operatsioonisüsteem ning tarkvara, mis tagab turvaaukude paikamise. Ainult uuendamise asemel ehitatakse täielikult uued serverid, mis võimaldab infosüsteemi puhastada ebavajalikest tarkvaradest ja failidest, tagades parema jõudluse ning hallatavuse. Viimaseks oluliseks eesmärgiks on teostada taastetestimine ning seeläbi tagada vajalikud juhendid juhuks, kui server või virtuaalserver(id) peaksid mingil juhul hävima.

Lõputöö koosneb kolmest põhiosast. Esimene põhiosa annab lühikese ülevaate vanade serverite ajaloost, ülesehitusest ning põhikomponentidest. Lisaks selgitatakse uue ülesehituse põhieesmärke. Teises peatükis tutvustatakse EENetis kasutusel olevat platvormi serverite poolautomaatseks ehitamiseks ning kolmandas peatükis on kokku võetud E-koolikoti infosüsteemi keskkondade ehitamine ning seadistamine, uuenduse planeerimine ja läbiviimine, mille käigus kolitakse nii test- kui ka toodangukeskkond uuetele virtuaalserveritele.

1. UUE SÜSTEEMI VAJALIKKUS

E-koolikott on digitaalse õppevara portaal (enne 2018. aastat kandis nime e-Koolikott või Koolielu Waramu/õppevara), mis koosneb kahest virtuaalmasinast, millest üks kuulub testkeskkonnale ja teine toodangukeskkonnale [3]. Mõlemad serverid töötavad Debian 8 operatsioonisüsteemil, mille tugi lõppeb 30. juunil 2020. Kasutusel olevad virtuaalmasinad omavad nelja põhilist süsteemikomponenti – proksi, rakendus, andmebaas ja indekseerija.

Serverite algne tellija ja haldaja oli Haridus- ja Teadusministeerium, EENeti teenuste osakond võttis infosüsteemi haldamise üle 2017. aastal, millele järgnes serverite majutuse ületoomine virtualiseerimisklastrisse [3]. Serverites kasutatav tarkvara ja operatsioonisüsteem on jõudmas oma eluea lõppu. Käesolev peatükk keskendub serverite uuendamise eesmärkidele.

1.1 Üks server üheks eesmärgiks

Algses olukorras on kõik süsteemi komponendid paigaldatud ühte serverisse, millest tulenevalt peavad komponendid jagama virtuaalmasina ressursse üksteisega. Selline lähenemine omab positiivseid ja negatiivseid külgi. Süsteemi haldav süsteemiadministraator peab tegelema ainult ühe serveri ülalhoidmisega ning kõik serveri komponendid on ligipääsetavad, kuid kõikide osade hoidmine ühes kohas suurendab turvariske. Ebaturvaline on hoida andmebaasi serveris, mis on ühendatud avaliku internetiga. EENetis on tavaks saanud jagada teenused serveritesse eesmärgi alusel, millest tulenevalt soovituslikuks (olenevalt siiski infosüsteemi funktsioonist) ülesehituseks oleks hoida nii proksi, rakendus kui ka andmebaas täiesti eraldiseisvates masinates. Eraldiseisvus tagab parema süsteemi komponentide jälgitavuse, selge ülevaate serveri eesmärgist, turvalisuse hoides süsteemide komponente eraldi ning parema ressursside kasutamise.

Iga infosüsteemi ehitamisel tuleb läheneda serveri ülesehitusele individuaalselt. Kui süsteem on ainult asutusesiseseks kasutuseks, siis üldjuhul jäetakse komponendid ühte serverisse, kuna koormus on tavapäraselt kuni 100 töötajat korraga ning ligipääs on piiratud, kuid kuna E-koolikott on välist kasutust saav infosüsteem, siis sellest tulenevalt peab arvestama parima jõudluse saavutamisega. Turvalisuse eesmärkidel paigaldatakse

E-koolikoti andmebaas eraldiseisvasse virtuaalmasinasse, mis ei ole ühendatud avaliku internetiga. Lisaks tagab baasi eraldi hoidmine parema jõudluse ja jälgitavuse. Rakendus ja proksi paigaldatakse ühte virtuaalmasinasse, kuna proksi on antud juhul väikese ressursitarbega ning tihedalt seotud rakendusega.

1.2 Ajakohastamine

Tulenevalt operatsioonisüsteemi vanusest tuleb valida sobiv lähenemine serveri ajakohastamisele, kõige kiirem lahendus on uuendada Debian 8 operatsioonisüsteem uuele versioonile, kuid puhtaim lähenemine on liigutada kõik süsteemi komponendid täiesti uude serverisse. See võimaldab süsteemist sorteerida välja ebavajalikud failid ja komponendid, mis võivad olla serverisse jäänud vana omaniku haldusajast või vananenud versioonidest. Lisaks on EENet ajakohastamisele lähenenud kui Debiani ja teistest sarnastest operatsioonisüsteemidest vabanemisega, kuna nimetatud operatsioonisüsteemid omavad tihti eelpaigaldatud komponente, mis ei ole ilmtingimata vajalikud süsteemi tööks ja hõivavad seeläbi virtuaalmasina ressursse [4]. Debiani pakirepositooriumitest paigaldatavad programmid omavad tihti küllaltki aegunud versioone ning operatsioonisüsteemi piirangute tõttu ei ole alati võimalik ametlikust repositooriumist paigaldada uuemaid saadaolevaid tarkvaraversioone, vananenud pakkide paigaldamine võib endaga kaasa tuua turvaprobleeme [5].

Debiani väljavahetamise kaalumine uue operatsioonisüsteemi vastu võimaldab survestada arendusmeeskonda ja projektijuhti mõtlema asjaolule, et uue operatsioonisüsteemi valimine võib võimaldada süsteemile paremat ressursikasutust, uuemaid tarkvaraversioone. Süsteemiadministraatorid annavad oma ideed üle ning projektijuht koostöös peaarendajaga saab otsustada pakutud ideede poolt või vastu.

Alati survestamine ei toimi, kuna projektis on näiteks käsil tähtsamate funktsioonide väljaarendamine, sest projektijuhi eesmärgiks on tagada infosüsteemi kasutajate maksimaalne meeleha, kuid samal ajal peab olema tähelepanelik turvalisusega, et turvaaukude tõttu ei toimuks süsteemist infoleket. Lisamõjutajateks on arendusmeeskonna suurus, sest uue tarkvara/operatsioonisüsteemi testimisprotsess on üldjuhul küllaltki aeganõudev ja ka kulukas. Meeskond peab olema piisavalt suur, et tagada põhjalik testimine ja vajadusel parandamine/täiustamine. Seetõttu võib

takistavaks teguriks jääda rahalise ressursi puudumine ning seetõttu kolimine võib viibida või ära jääda.

Arch Linux (edaspidi ka Arch) on EENetis eelistatud operatsioonisüsteem, sest süsteem võtab andmeladestuspinnal vähe ruumi. Operatsioonisüsteem on mugandatud sobivamaks ja vaikepaigaldusest on välja võetud mitmed ebavajalikud tarkvarad (süsteemiadministraatorite poolt optimeeritud Arch võtab ruumi 567 MB, Debiani miinimumpaigaldus üldjuhul ~2 GB) [4]. Arch Linuxi pakihaldusel on positiivsed ja negatiivsed küljed, positiivsest küljest operatsioonisüsteem paigaldab alati kõige uuema versiooni tarkvarast ning enamasti on see paar versiooni uuem kui teistel operatsioonisüsteemidel [6]. See tagab tavaliselt parema töövõime ning turvalisuse vaatest on paigatud turvaaugud, mis võisid esineda vanematel versioonidel. Negatiivseks küljeks on asjaolu, et ametlikus pakihalduses on küllaltki vähe väga spetsiifilisi pakke, mille tõttu peab pöörduma teise pakihalduse (AUR, *Arch Linux User Repository*) poole, mis paraku ei oma otsest pakihalduspaigaldust.

1.3 Varundus ja taastamine

Infosüsteemidel, mis omavad andmekogusid (näiteks EHIS, mis sisaldab haridusandmeid või EIS, mis sisaldab lõpueksamite hindeid ja tunnistusi), peavad läbima ISKE (infosüsteemide kolmeastmeline etalonturbe süsteem) auditi, mille põhjal määratakse süsteemile üks kolmest turvaastmest – kõrge, keskmine või madal [1]. Kuna E-koolikott sisaldab ainult õppematerjale, siis süsteemi enda jaoks ei ole ISKE auditi läbimine kohustuslik, kuid kuna EENeti seadmeruum vastab andmekeskuste keskmisele turvaastmele, siis süsteemiadministraatorid töötavad selle nimel, et tagada sama tase kõikidele võimalikele infosüsteemidele.

Suur osa talituspidevuse tegevuskavast on taastetestimine, mis on dokumenteeritud meetod, mille abil testitakse infosüsteemi taastamist katkestuse/hävingu puhul. Kõrge turbeastmega infosüsteeme on tarvis üle vaadata ning testida iga-aastaselt, kuid auditile mittekuuluvaid süsteeme testitakse läbi kord kolme aasta jooksul ning dokumentide ülevaatamine toimub peale igat suuremat versiooniuuendust [7].

Kuna E-koolikott jõudis EENeti haldusesse 2017. aastal, siis polnud süsteemil täielikku taasteplaani, kuid oli olemas esialgne paigaldusjuhend. E-koolikoti täielik uuendamine

võimaldas viia läbi samaaegselt kõik vajalikud tarkvarauuendused, taastetestimise ja dokumentatsiooni korrastamise ning täiendamise. Uus taasteplaan omas vastavalt ISKE talituspidevuse poliitikale: uut süsteemi konfiguratsiooni, riistvara andmeid, varukoopiate asukohta, taastetestija andmeid ja märkuseid ning taastamise detailset protseduuride dokumentatsiooni koos tavapäraste uuendusjuhenditega [7].

E-koolikoti virtuaalservereid on praeguses olukorras raske varundada, kuna puudub kontroll kuhu server talletab vajaminevaid andmeid. Näiteks EENetis kasutusel oleva lahenduse puhul talletab server kõik oma failid /srv kaustapuule, tavapärase Debiani puhul on failid laiali /srv, /etc või /var kaustapuudel, kusjuures iga uuendusega võivad asukohad uuesti muutuda.

2. UUS PLATVORM

EENetis on tavaks vanad süsteemid võimalusel ja vajadusel (näiteks süsteeme, mis on plaanitud lähiajal sulgeda ei oleks ajakulu mõttes tark uuendama hakata, kui just ei avastata turvaprobleeme) likvideerida ja tõsta Arch Linux operatsioonisüsteemile. Arch on EENetis üles seatud ainult lugemisõigustesse, millest tulenevalt virtuaalmasin võtab endale opsüsteemi külge üle võrgu. Ainult lugemisõiguste kasutamine tagab suurema turvalisuse, kuna süsteemi tööfaile ei saa serverist muuta. Kuigi operatsioonisüsteem on ainult lugemisõigustes, siis sellegipoolest tuleb seda varundada nagu kõiki teisi süsteemi osasid. Kui peaksid tekkima probleemid serveris, kus ladustatakse operatsioonisüsteemi, siis ei oleks võimalik sooritada mitte ühegi serveri taastamist. Lisaks tagab opsüsteemi varundamine tagavaraplaani juhuks, kui süsteemiadministraator peaks tegema muudatuse, mis päädib probleemidega süsteemi töös. Varunduse olemasolu tagab sellistel juhtudel kiire ülemineku eelnevale ülesehitusele süsteemi töö taastamiseks.

2.1 Serverivabrik

Operatsioonisüsteemi ehitamiseks on EENetis kasutusel Serverivabrik, mis on BASH ja Ruby skriptide kogum võimaldades kettale paigaldada eelseadistatud operatsioonisüsteeme, omades nii Preseedi kui konfiguratsioonihaldustööristade (näiteks Puppet) funktsionaalsusi. Seda kasutatakse virtuaalmasinate operatsioonisüsteemide loomiseks ja versioonihalduseks. Serverivabrikut hoitakse EENeti gitiserveris, tagades töötajatele ligipääsu loodud serverite seadistusele. Serverivabrik võimaldab uue virtuaalmasina laadimiseks vajalikke ressursse (operatsioonisüsteemi tuumasid, pakke, konstruktoreid ja spetsiifilisi konfiguratsioonifaile) hoida ühes asukohas, kasutades efektiivselt andmemahtu ja lihtsustades virtuaalmasinate loomist ning haldust. Serverivabriku peamisteks eelisteks on kiire ehitamine, põhjalik süsteemiadministraatorite enda poolt loodud dokumentatsioon, versioonitud ülesehitud (ehk võimalus varasemale versioonile tagasi minna), turvaline keskkond ning võimalus vähese vaevaga ehitada nii toodangu-, test- kui ka liivakasti keskkondasid [8][1].

* Märkus 1. Tegemist on kollektiivselt loodud KKK rubriigi tekstiga, lõputöö autor on üks teksti loojatest.

Iga serveri jaoks on Buildfile (e.k. konstruktor), milles defineeritakse sellele omistatavad väärtused. Konstruktor käivitatakse läbi skripti, mis loob Ehitaja kettale serveri nimelise kausta kuupäevaga, pakib sellesse lahti operatsioonisüsteemi põhja ja konstruktori alusel teeb konfiguratsioonimuudatused, paigaldab pakid, loob vajalikud sümbolnimed. KVM virtualiseerimist kasutav VM sooritab alglaadimise, kasutades PXEd ja saab TFTP protokolliga Linuxi tuuma. Seejärel haagitakse konstruktoriga tekitatud operatsioonisüsteemi juurfailisüsteem NFS jaona külge. Käivitamiseks vajalikud seaded saab VM PXELinux konfiguratsioonifailist, mis on VMi virtuaalse võrguseadme MAC aadressi nimeline. Fail sisaldab lisaks Kerneli versioonile ka parameetreid nagu külgehaagitavad kettad ja IP aadressid. VM on alglaadimise sooritanud, võimaldades külgehaagitud ketaste initsialiseerimist, näiteks selle partitsioneerimist, failisüsteemi loomist ja failide kopeerimist [8][1].

2.1.1 Operatsioonisüsteem

Arch Linuxi üheks eeliseks on operatsioonisüsteemi eelnev kokkuehitamise võimalus ilma virtuaalmasina töölepanekuta, mida näiteks Debiani või Ubuntuga on raske saavutada, kuna nimetatud opsüsteemid ehitavad süsteemi osasid alles *boot*imise ehk käivitamise käigus. Lisaks aitab Arch Linux kaasa serverite ajakohasena püsimisel, kuna iga serveri operatsioonisüsteemi taasehitamisega paigaldatakse kõige uuemad pakid. Kohati kõige uuemate pakkide paigaldamine võib kaasa tuua ka probleeme, näiteks olukordades, kus soovitud programmi versioon on uuem kui infosüsteemis toetatud versioon, sellistes olukordades tuleb kasutusele võtta kõrvaline pakkide haldus AUR.

Arch Linux on ainuke operatsioonisüsteem, mida on võimalik valida Serverivabrikus serveri ehitamisel. Kõrvalharuna on proovitud arendada valikuvõimalustesse Alpine Linuxi (edaspidi Alpine) rakendamist. Alpine on sarnaselt Archiga väga väikese mahuga, mis on üheks põhiliseks teguriks EENetis operatsioonisüsteemi valikul [9]. OSi arendusmeeskonna üheks eesmärgiks oli varem mahutada kogu operatsioonisüsteem diskett andmekandjale. Tulenevalt oma suurusest on Alpine sama hea kanditaat nagu Arch, kuid vähese kasutuse tõttu EENetis ei ole ajaliselt mõistlik hakata rakendama uut operatsioonisüsteemi, kuna näidiseid sarnasest süsteemist on vähe ning ajakulu kujuneks ebamõistlikult suureks toomata seejuures suurt kasu.

2.1.2 Pakkide paigaldamine

Loodavas serveris on vaja infosüsteemi töötamiseks mitmeid erinevaid tarkvarasid. Kõik tarkvarad paigaldatakse pakkidest, mis pärinevad Archi pakihaldusest ja paigaldatakse Serverivabriku automatiseeritud ülesehituse abil, tänu millele ei pea süsteemiadministraator käsitsi pakke alla laadima ning paigaldama.

Konstruktori faili algul määratud operatsioonisüsteemi väärtuse põhjal teab Serverivabrik, millise pakihalduse repositooriumi poole pöörduda (koodilõik 2.1), asukohad on eelnevat käsitsi seadistatud vabriku tarkvaras. Paki paigaldamise esilekutsumiseks peab sisestama koodilõigus 2.2 toodud näite põhjal *sf_install_pkg* käsu. *sf_install_pkg* käsu teiseks väärtuseks on soovitud tarkvara nimi. Vajaliku tarkvara otsimine käib repositooriumis nime põhjal ning kõige kindlam on leida õige nimi Archi ametlikul lehel otsingumootori abil.

```
# OS: arch
# BUILD_VER: 3
```

Koodilõik 2.1. Operatsioonisüsteemi defineerimine

```
sf_install_pkg          <soovitud tarkvara nimi>
```

Koodilõik 1.2. Tarkvara paigalduse esilekutsumine

2.1.3 AUR pakkide paigaldamine

AUR on kogukonna poolt hooldatav repositoorium, kuhu kasutajad ise arendavad tarkvarasid, mis on ametlikus Arch Linuxi repositooriumis puudu või põhinevad vanemal versioonil. AURi puhul ei ole Arch Linuxi „tuge" või „kinnitust", et tarkvara on kontrollitud, mis tähendab, et iga sealt laetud pakk on tehniliselt mitteametlik ning alla tuleks neid laadida ainult siis, kui ollakse pakkide turvalisuses veendunud.

Sarnaselt ametlikust repositooriumist saadud pakkide paigaldamiseks on AURi pakkide paigaldamiseks eraldi *sf* käsk (koodilõik 2.3). AURi pakkide puhul on tarkvarade jõudmine operatsioonisüsteemi keerulisem ja ajakulukam. Vabrik on võimeline sf käsku täitma, kuid eelnevalt peab süsteemiadministraator vastava tarkvara paki käsitsi kokku ehitama.

```
sf_arch_install_aur_pkg          <soovitud tarkvara nimi>
```

Koodilõik 2.3. AURist pärineva tarkvara esilekutsumine

Serverivabrikus on eraldi operatsioonisüsteem AURi pakkide ehitamiseks ning peale esmakordset käsitsi kokku ehitamist suudab vabrik pakki automaatpaigaldusena operatsioonisüsteemi initsialiseerida. Enne paki ehitamist on vaja soovitud tarkvara leida AURi repositooriumist ning laadida alla kokku pakitud versioon wget käsuga. Peale allalaadimise sooritamist tuleb pakk lahti pakkida ning konsoolis edasi liikuda tarkvara kausta. Järgmiseks sammuks on paki ehitamine makepkg käsuga – tegu on skriptiga, mis automatiseerib pakkide ehitamist (koodilõik 2.4). Peale paki ehitusprotsessi lõppu on tarkvara valmis paigaldamiseks. Viimaseks tegevuseks jääb tarkvara liigutamine ettenähtud kaustapuusse ning omandi muutmine.

```
wget https://aur.archlinux.org/cgit/aur.git/snapshot/solr6.tar.gz

tar -zxf solr6.tar.gz

cd solr6

makepkg
```

Koodilõik 2.4. AURist pärit tarkvara paki ehitamine

2.1.4 Moodulite paigaldamine

Tarkvarad, mis leiavad tihti kasutust, on korjatud kokku ning lisatud eraldi moodulisse (inglise k *partial*). Moodul on standardiseeritud kogum kindlaksmääratud tegevustest, mida on vaja mitmel serveril/teenusel taaskasutada ja millele on võimalik rakendada standardkonfiguratsiooni (näiteks veebiserver, varundus või monitooring) [10]. Moodul on ülesehituselt sarnane tarkvara paigaldamisele, kuid omab enamasti rohkem lisasamme.

Tarkvara paigaldamine lugemisõigustes olevasse operatsioonisüsteemi ei ole alati kõige kergem, kuna paigalduse käigus pakitakse lahti failid, kuid programm loob tavaliselt käivitudes uusi faile. Töötamise käigus loodud failid tuleb suunata ladustusseadmele, mis on eraldi kaustapuus võrreldes operatsioonisüsteemi kaustadega. Tihti rakenduse

failide suunamine kõvakettale tuleb teha operatsioonisüsteemi seadistamise hetkel kasutades nimeviiteid ehk sümboolseid viiteid.

Põhilisteks liigutusteks on algseadistuse kõrvale tõstmine ning selle asendamine optimeeritud seadistusega, mis on mooduli kaustas või nimeviidete tegemine. Sümboolsed viited võimaldavad operatsioonisüsteemi paigaldatud tarkvara jätta samaks vihjates kausta või faili asukohale sümboolselt. Lõppasukohas on enamasti /srv kaustapuu, mille aluseks on virtuaalmasina kaustapuu.

Kõikide tarkvarade puhul ei ole vaja rakendada mooduli ülesehitust, kuna kõik programmid ei vaja lisaseadistamist lugemisõigustes oleva operatsioonisüsteemi tõttu. Kui rakendus on kasutuses ainult ühes serveris siis on süsteemiadministraatoril õigus langetada isiklik otsus, kas liigutada tarkvara paigaldamine moodulisse või mitte. Nagu tarkvarade puhul, kasutatakse moodulite paigaldamisel samuti sf käsku, kuid teistsuguse ülesehitusega – lisaks tavapärasele sf_build ning tarkvara osadele tuleb ette lisada moodulite kaustale vihjav partial (koodlõik 2.5).

```
sf_build        partial/apache
```

Koodilõik 2.5. Mooduli paigaldamine

2.1.5 Tüüpilised konstruktori osad

Iga serveri tegevuskavas on alati administraatoritele kasutajate loomine, et tagada ligipääs administraatoritele ka serveri esialgse ehitaja haigestumise või töölt lahkumise puhul. Vaikeseadistusena paigaldatakse serverisse ainult süsteemiadministraatorite võtmed ning erandjuhtude korral on võimalik manuaalselt lisada juurde kolmandaid osapooli, näiteks arendajad või projektijuhid (koodilõik 2.6 ja koodilõik 2.8). Kasutajate ligipääs serveritesse on ainult läbi SSH võtme, mitte kontodega seotud paroolidega. Serverites seadistatakse parool ainult juurkasutajale (mille kasutamine peale kasutajate loomist keelatakse vaikeseadena), sest selline käitumine tagab turvalisuse, et serveri kasutajate paroolid ei lekiks tahtmatult avalikku internetti – paraku on laialdaseks veaks sama salasõna kasutamine mitmetes, kui mitte kõikides masinates ja ka igapäevaselt kasutatavatel veebilehtedel [11].

```
# Administrators
sf_build partial/admins
```

Koodilõik 2.6. Süsteemiadministraatorite paki paigaldamine

Kõik serverid, mis ei ole süsteemiadministraatorite arenduskeskkonnad on kohustuslik seadistada varundama (koodilõik 2.7). Virtuaalmasinaid varundatakse iga öö ning vastavalt süsteemi ülesehitusele tehakse vajadusel mitu varundust, kui süsteemis on andmebaas, siis tehakse sellest tõmmis ning kogu kirjutatava ketta sisust tehakse lisaks veel eraldiseisev koopia.

```
# Backup
sf_build partial/borg
```

Koodilõik 2.7. Varunduspaki paigaldamine

Kõik serverid, mis omavad avalikku liidest vajavad tulemüüri (EENetis on kasutusel FireHOL tarkvara), et hoida ära turvaintsidente ja rünnakuid. Tulemüüri paigaldamisega piirtakse ära kõik tegevused avalikul liidesel, kuid sisene liides jääb avatuks. Tulemüüri rakendamisega lubatakse avaliku liidese pihta (SSH) ainult kontorivõrgu vahemik. Kõige põhilisemad pordid, mis infosüsteemidel avalikul liidesel avame on 80 ja 443 ehk HTTP ja HTTPS, mis on vajalikud veebiliikluseks. Vahel harva lisatakse avalikule liidesele arendaja või mõne kolmanda osapoole ligipääs pordile 22 ehk SSH võimalus (näiteks e-Koolikoti puhul) või mõni muu vajalik port infosüsteemi ja/või rakenduse jaoks (koodilõik 2.8). Kolmandatele osapooltele ligipääsu jagamiseks luuakse lisafail FireHOL kausta, kuhu lisatakse nende IP-aadressid või nende vahemikud ning Serverivabrik kopeerib faili(d) operatsioonisüsteemi. Samuti on väljuvad ühendused piiratud vaid vajalike portidega.

```
# Firehol setup and allow ssh access for tunnel, for developers only
sf_build partial/firehol
cp -v $BD/ssh.conf /etc/firehol/public/
cp -v $BD/koolitaja.conf /etc/firehol/public/
```

Koodilõik 2.8. Tulemüüri paki paigaldamine ja ligipääs arendajatele

Veebiserveri proksina on e-Koolikotis kasutusel Apache, mis suunab liikluse Java rakendusele. Apache on seadistatud tegema SSL *offload*'i ehk sertifikaatide edastamist, kontrolli ja liikluse edasisuunamist Java rakenduse suunas HTTP kujul selleks, et rakendus saaks ülesandeid kiiremini hallata ja ei peaks tegelema päringute lahti krüpteerimisega.

Java rakendus käitab serveris eraldisesva kasutaja halduses, et maandada turvariske, tagada parem hallatavus ning vajaduse korral kergendada vigade otsimist (koodilõik 2.9). Kuna rakenduse kasutaja piiratud õigustega saab see kirjutada ainult etteantud kaustadesse ning kontoga ei ole seotud SSH privaatvõtit, st puudub võimalus identiteeti kasutades teistesse serveritesse edasi liikuda.

```
# kott user
useradd -c Kott -d /srv/kott -U -o -u 501 -s /bin/bash kott

# kott.service
cp -v $BD/kott.service /etc/systemd/system/
systemctl enable kott.service
```

Koodilõik 2.9. Kott kasutaja ja teenuse seadistamine

Rakenduse serveris asub Solr, mis on kasutusel kui otsingumoor Java rakenduses. Java rakenduse käitamiseks on tarvis paigaldada Java Runtime Environment ning rakenduse komponentide allalaadimiseks Giti pakk. Rakenduse seadistusfailid ja Java programm on internetis avalikult üleval Giti koodirepositooriumis (koodilõik 2.10).

```
# Solr
sf_install_pkg solr
ln -vs /srv/solr/dop /usr/share/solr/server/solr/dop
chmod +x /usr/share/solr/bin/solr
systemctl enable solr

# JDK
sf_install_pkg jre8-openjdk

# Git
sf_install_pkg git
```

Koodilõik 2.10. Vajalikud süsteemi lisad

2.1.6 Serveri ehitamine

Tulenevalt vabriku eripärast on serveri ehitamiseks vaja käivitada ainult üks käsk (koodilõik 2.11), millel tuleb määrata ehitatava serveri nimi ja tüüp, mille märkimine on informatsiooniallikaks süsteemiadministraatorile andes märku, kas tegu on arendus- või toodangukeskkondadega. Serveri ehitamise protsess on enamjaolt automaatne kui välja arvata protsessi lõpp, kus operatsioonisüsteemi ehitamise skript küsib kasutajalt, kas on soov asendada kasutuses oleva virtuaalmasina operatsioonisüsteem uuega. Uus operatsioonisüsteem võetakse kasutusele alles peale virtuaalmasina taaskäivitamist.

```
./build ekoolikott dev
```

Koodilõik 2.11. Operatsioonisüsteemi ehitamine

2.2 Virtuaalmasina käivitamine ja VIPS

Virtuaalmasina käivitamiseks on tarvis läbida mitmeid tegevusi, esimeste sammude seas on operatsioonisüsteemi kasutavate programmide ja seadistuse kirjutamine konstruktorisse ning OSi kokku ehitamine. Lõppliku serveri käivitamiseks on vaja teha tegevusi nii käsureal kui ka graafilistes liidestes (võimalus on teha kõik käsurealt, kuid see raskendab ülesehitust ning graafilisel liidesel korduvalt sama nupu vajutamine on mugavam ja kiirem.

EENetis on virtuaalmasinate haldamiseks kasutusel Virtuaalprivaatserveri teenus VIPS, mis põhindeb Proxmox vabavaral [13]. Virtualiseerimisklastris on kokku 23 serverit ning 281 virtuaalserverit. Süsteemiadministraatoril on vaja kindalt ülevaadet, kui palju on kindel virtualiseerimisserver parasjagu koormatud ning Proxmox võimaldab tabelitena jägida ressursikasutust ning seeläbi leida parim server, kuhu uus infosüsteem või virtuaalserver paigutada. Iga server on piiratud ressurssidega, mistõttu on oluline jälgida, et ühtegi serverit ei koormataks liigselt üle. Kui süsteemiadministraator peaks näiteks andma virtuaalmasinale rohkem mälu, kui serveril jagada on, siis võib olukord lõppeda serveris probleemidega. Protsessori tuumade jagamisega võib olla leebem, kuna on väga vähe virtuaalmasinaid, mis kasutavat reaalsuses üle 80% neile määratud jõudlusest. Sellest tulenevalt võib olla ühes virtualiseerimisklastris kasutuselolevate virtuaalmasinate protsessorite tuumade arv suurem kui füüsiline kogus kuni kahekordselt. Kergemaks jälgimiseks on joonisel värvikoodid, must viitab vabale

ruumile, sinine on hoiatav (kasutus on piiril peal või juba üle) ning punane viitab kriitilisele olukorrale, mis vajaks sekkumist (joonis 2.12). Süsteemiadministraator peab jagama ressursside kasutust mõstulikult ning arvestama teiste virtuaalmasinatega, mis on samas klastris – mõni klaster jäetakse teadlikult tühjemaks, kuna seal võib olla majutatud mõni tähtis infosüsteem, mida ei tasukus koormata teiste masinatega või mille ressursid langevad ja tõusevad hooajaliselt (näiteks Eksamite Infosüsteem, või Sisseastumiste Infosüsteem, mis saavad suurt koormust kas eksamite ajal paar korda aastas või sisseastumiste hooajal).

	Memory	Allocated		CPU	Allocated		VM-s	
	MB	running	stopped		running	stopped	R	S
koonerdaja	128878	36864	27072	32	28	13	8	3
majandaja	128863	20480	16384	24	20	20	3	7
mikrotaja1	193324	98304	64512	40	50	42	8	5
mikrotaja2	193324	146432	0	40	73	0	15	0
mikrotaja3	193324	129028	12288	40	75	8	10	1
mikrotaja4	193324	136192	18432	40	63	14	20	3
mikrotaja5	193330	124928	0	40	54	0	10	0
odaviskaja1	192091	132100	0	48	64	0	18	0
odaviskaja2	192092	101376	22528	48	84	16	15	7
odaviskaja3	192092	113656	9728	48	61	11	19	6
odaviskaja4	192092	45056	0	48	30	0	6	0
rabaja	128878	16384	0	32	4	0	1	0
riisuja1	192076	100352	0	56	56	0	15	0
riisuja2	192076	126976	0	56	34	0	7	0
riisuja3	192076	49152	0	56	40	0	3	0
riisuja4	192076	74752	4096	56	44	2	4	2
seikleja1	192074	73732	18432	96	65	22	17	4
seikleja2	192074	83968	2048	96	20	2	5	2
seikleja3	192074	94208	0	96	46	0	12	0
seikleja4	192074	73728	0	96	26	0	4	0
tasuja	128878	88064	0	32	48	0	17	0
tasuja2	128878	24576	0	32	12	0	3	0
valitseja	128822	111616	0	32	85	0	21	0
TOTAL	4044790	2001924	195520	1184	1082	150	241	40

Joonis 2.12. Klastri koormustabel

Peale VIPS keskkonnas virtuaalserveri loomist (ressursside määramist) määrab Proxmox virtuaalmasinale automaatselt sisevõrgu MAC-aadressi, mis on põhiliseks lüliks virtuaalmasina ressursside sidumisel operatsioonisüsteemiga. Server, mis ühendatakse avaliku võrguga vajab lisategevusi kasutajaliideses, et lisada peale sisevõrgu seadmele tugi välisvõrgu vastuvõtmiseks võrguseade. IP-aadressid seadistatakse DHCP abil samas serveris, kus ehitati operatsioonisüsteem. Lisaks märgitakse kasutuses olevale operatsioonisüsteemi fail kõvaketta ülesehitusest ja vajadusel lisatakse väline IP-aadress. Peale vajalike linkide loomist operatsioonisüsteemi ja virtuaalmasina ressursside vahel tuleb masin taaskäivitada. Viimaseks tegevuseks seadistatakse ketas ning tehakse viimane taaskäivitus.

3. UUENDUSE LÄBIVIIMINE

Enne uuenduste läbiviimist suheldi aktiivselt projektijuhi ja arendajaga, kõik eelnevad testimised sai läbi mängitud arenduskeskkondades, mis tagas võimaluse luua uuendusjuhendi, et uuendamise ajal läheks töö sujuvamalt ning süsteemi töö katkestuse kestvus oleks minimaalne. Kogu süsteemi kolimise vältel peavad nii arendaja(d) kui ka projektijuht olema kättesaadavad. Arendaja olemasolu on oluline siis, kui süsteemi kolimisel peaks tekkima tarkvarade rakendamisel probleeme ning projektijuhi poolt on otsustada olukorrad, kui kolimise käigus ei peaks mõned komponendid töötama eesmärgipäraselt ning tuleb langetada otsus, kas liikuda vanale operatsioonisüsteemile tagasi ja proovida uuendust mõnel teisel ajahetkel või jätta süsteem probleemselt tööle ning üritada neid jooksvalt parandada.

3.1 Testkeskkonna ehitamine

E-koolikoti uued masinad koosnevad kahest serverist – mõlemal keskkonnal (test ja toodang) on eraldi rakendus ja andmebaas. Sellest lähtudes tuleb luua kaks konstruktorit ehk *Buildfile*'i, kus on süsteemile ette kirjutatud kõik vajalikud komponendid. Rakenduse ja andmebaasi ainukeseks sarnasuseks on süsteemiadministraatorite kasutajate lisamine ja serveri varunduse seadistamine.

Nii test- kui ka toodangusüsteemi jaoks kasutatakse ühte konstruktorit, mille tõttu on kohati vaja kasutada *if* ehk kui lauseid, et määrata ära millised tegevused tehakse testis ja millised toodangus (koodilõik 3.1). Sellist tegevust on kasutatud kui on mingeid määravaid kõrvalekaldeid test- või toodangusüsteemi vahel (näiteks kui peaks olema vaja määrata spetsiifiline varunduse tegemise kellaaeg või kasutada lisatarkvara ühes keskkonnas).

```
# Kui tegemist on TEST süsteemiga
if [ "$SF_SERVER_XNAME" = 'ekoolikotttestdb' ]; then
   # Käsud mis käivitatakse kui on tegu test süsteemiga
fi

# Kui tegemist on LIVE süsteemiga
if [ "$SF_SERVER_XNAME" = 'ekoolikottdb' ]; then
   # Käsud mis käivitatakse kui on tegu live süsteemiga
fi
```

Koodilõik 3.1. Kui-lause (*if*-lause) ülesehitus

3.1.1 Proxy

Vanad Apache'i seadistusfailid olid pikad ja koosnesid mitmetest failidest, mistõttu oli nende lugemine ja haldamine keerukas. Parema ülevaate saamiseks oli tarvis seadistusfailid ebavajalikust ja võimalikust turvaohtlikust sisust puhastada, tarvilik osa üheks failiks sobitada ning ülesehitus arusaadavaks ja loogiliseks seada.

Apache'i seadistusfaili korrastamine kujunes ajakulukaks. Vältimaks korduvaid ridu monteeriti failide kogum üheks suureks failiks, mis paigutati ilma muudatusteta uude prooviserverisse. Seadistusfail jäi algul puutumata, et tagada võimalikult väike erinevus algse klooniga, sest kui oleks asutud kohe muudatusi sisse viima oleks võinud tahtmatult juhtuda mõne olulise rea kustutamine, mis oleks toonud kaasa lisatööd hiljem vea otsimisel. Suurem osa muudatustest tulenesid süsteemi kolimisest uude operatsioonisüsteemi, mis töötab ainult lugemisõigustes. Sellest tulenevalt toimus faili korrastamine järk-järgult ning viimased muudatused said paika paar nädalat peale tootekeskkonna uuendust. Muudatuste sisseviimisel oli kõige olulisem jälgida, et veebi kaustapuu ei satuks välismaailmale nähtavaks, sest kaustapuus asuvad failid, mis annavad informatsiooni süsteemi ülesehituse ning kasutavate skriptide kohta.

3.1.2 Rakendus

Et tagada rakenduse kõige turvalisem ülesehitus, paigaldati rakendused eraldiseisva kasutaja alla töötama (koodilõik 3.2). Kui rakendus töötab *root* ehk juurkasutaja õigustes, võib see iseendale anda liigseid volitusi ning kui rakendusele peaks rünnaku tagajärjel ligi pääsema kolmas osapool, oleks tal maksimaalsete õiguste tõttu võimalus liikuda serveris vabamalt ringi ja saada ligipääs konfidentsiaalsetele failidele.

```
[Unit]
Description=e-Koolikott


[Service]
User=kott
WorkingDirectory=/srv/kott
ExecStart=/usr/bin/java -Xmx2048M -jar kott.jar --spring.config.location=custom.yaml
ExecStop=/bin/kill -9 $MAINPID


[Install]
WantedBy=multi-user.target
```

Koodilõik 3.2. Kott teenuse ülesehitus

Rakenduse piiramine ei hoia ära andmete lekke kui see peaks juhtuma, kuid kõige tähtsamaks on säästa edasist kahju. Tänu piiratud õigustele ei saa rakendus kontrollida teiste protsesside tööd nagu veebiserver, SSH server ja palju muud.

3.1.3 Solr

Solr tarkvara rakendamine oli uute serverite ehitamisel kõige keerulisemaks protsessiks. Keerukus tulenes enamasti asjaolust, et uus operatsioonisüsteem on kasutuses ainult lugemisõigustes ning Solr üritab kirjutada asukohtadesse, kuhu tarkvara kirjutada ei saa. Operatsioonisüsteemi eripärast tulenevalt tuli tarkvara jaoks luua palju erinevaid nimeviite kaustade ja failide vahel, mis võimaldas vajalikud failid suunata kaustadesse, kuhu Solr sai kirjutada

Serverite kolimise hetkel ei olnud Solri võimalik leida ametlikust Archi pakihoidlast, kuna tarkvara oli eemaldatud kasutuslitsentside rikkumise tõttu kuu aega varem. Seetõttu oldi kohustatud kasutama AURi. Solri päritolus AURist tähendas seda, et programm töötas hoopis teistsugustes kaustades võrreldes Debiani operatsioonisüsteemiga, mis raskendas rakenduse seadistamist märkimisväärselt.

Rakenduse kolimise tegi veelgi keerukamaks tarkvara esialgne seadistus ja andmebaasiga liitmine. Luues uue sissekande ei suutnud tarkvara ühenduda andmebaasiga ning tuues üle seadistusfaili vanast serverist edastas programm vigu.

Ainsaks lahenduseks oli iga võimaliku ülesehituse läbikatsetamine, mis oli ajakulukas. Takistavaks teguriks osutus ka arendaja poolt loodud juhis, mis erines oma ülesehituse poolest omakorda aktuaalse ja uue operatsioonisüsteemi ülesehitusest.

3.1.4 Andmebaas

Andmebaasid paigaldatakse rakendusserverist eraldi virtuaalmasinasse (koodilõik 3.3), et tagada parem jõudlus, käideldavus ja monitoorimine [12]. Avalikku liidest andmebaasidele ei seata, kuna baasile vajab ligipääsu ainult rakendus läbi sisevõrgu ning kolmandatele osapooltele ei pea andmebaas avalikul liidesel kättesaadav olema. Kuna server on sisevõrgus, siis on võimalik lubada rakenduse ja andmebaasi vahel krüpteerimata liiklus, kuid see ei tähenda, et ei rakendataks turvameetmeid. Andmebaasidele on piiratud IP-aadresside vahemikkudega ligipääs ning baasi kasutajad on parooliga kaitstud.

```
# MariaDB
sf_build partial/mariadb
```

Koodilõik 3.3. Andmebaasi paki paigaldamine

3.1.5 Üle võrgu jagamine

Kõrvalise lisana on lisatud rakendusse *Network File System* (eesti k üle võrgu failide jagamine) ehk NFS, mis on vajalik süsteemiadministraatoritele süsteemi uuenduste läbiviimiseks (koodilõik 3.4). NFS võimaldab rakendusserveril ligipääsu andmebaasi kettale, et teostada andmebaasi tõmmis enne iga uuenduse läbiviimist. Tõmmis on vajalik eriti olukordades, kus uuendus ebaõnnestub ning on vaja minna tagasi vanemale versioonile. Andmebaasi konstruktorisse tuleb lisada NFS pakk, et võimaldada vastava kasuta väljajagamist rakendusele. Turvalisuse tagamiseks on kaust välja jagatud ainult rakendusserverile piirates ligipääsu IP-põhiselt.

```
# NFS, andmete vastuvõtmiseks
sf_install_pkg nfs-utils

# NFS, andmete väljajagamiseks
sf_build partial/nfs-server
cat >> /etc/exports <<'EOT'
/srv/mysqldump x.x.x.x/32(rw,sync,no_root_squash,no_subtree_check)
EOT
```

Koodilõik 3.4. NFSi seadistamine

3.2 Keskkondade kolimine

Test- ja tootekeskkonna uuendusjuhendid on peaaegu täielikult identsed, omades paari erinevust kaustanimedes ja IP-aadressides (koodilõik 3.5). Testkeskkonna uuendusega alustati hommikul kell üheksa, kui siseneti vanasse rakendusserverisse ja peatati veebining rakendusserver. Tootekeskkonna uuendus tehti infosüsteemi versiooniuuendusega koos.

```
# Vanasse serverisse sisse
ssh 193.40.55.130
sudo -i
/etc/init.d/apache2 stop
/etc/init.d/dop stop
```

Koodilõik 3.5. Vanades serverites teenuste peatamine

Andmebaasi tõmmis on vajalik uue baasi ülesseadmiseks (koodilõik 3.6). Kõige riskivabama tõmmise saavutamiseks peab seiskama nii rakenduse kui ka veebiserveri, et andmebaasi suunas poleks avatud ühtegi ühendust ning rakendus ei oleks samaaegselt kasutuses. Kui kasutaja peaks tõmmise loomise ajal infosüsteemis sissekanded tegema, võivad sisestatud andmed jääda puudu uuest keskkonnast või mõjutada edaspidist süsteemi tööd. Lisaks hõivab tõmmis suure osa kasutatavast ressursist endale muutes süsteemi kasutaja jaoks ebameeldivalt aeglaseks.

```
# DB dump käima (võtab aega ~15 min)
mysqldump --all-databases > /srv/mysqldump/$(date +%Y%m%d)-dump.sql
```

Koodilõik 3.6. Andmebaasi tõmmise loomine

Uus server oli eelnevalt liivakast infosüsteemi tööle saamiseks, kus katsetati läbi süsteemi uuendamine ja komponentide koostöö. Selleks, et uude serverisse saaks hakata andmeid ja faile üle tooma, oli oluline eelnevad testimisel kasutatud failid eemaldada ja rakenduse programmid peatada (koodilõik 3.7).

```
# Uue serveri asjad seisma
ssh 193.40.55.70
sudo -i
sc stop kott
sc stop httpd

# Uues serveris vanade kaustade eemaldamine
rm -r /srv/kott/reviews /srv/kott/uploads
rm -r /srv/kott/frontend
rm /srv/kott/kott.jar
```

Koodilõik 3.7. Uute serverite puhastamine

Uute failide liigutamiseks oli eelnevalt uuritud millised kaustad on vajalikud ning suureks abiks oli siinkohal eelseadistatud seadistusfailil, tänu millele sujus uuendus kiiremini. Uude serverisse toodi üle Java rakendus, veebiliidese kaustas olev kasutajaliides ning *uploads* ja *reviews* kaustad, kus asuvad infosüsteemis laetud ja salvestatud failid (koodilõik 3.8).

```
# Uute kaustade kopeerimine plus .jar
scp -r kristi@10.40.0.130:/var/www/dop/uploads /srv/kott
scp -r kristi@10.40.0.130:/var/www/dop/reviews /srv/kott
scp -r kristi@10.40.0.130:/var/www/dop/frontend /srv/kott
scp kristi@10.40.0.130:/var/www/dop/kott.jar /srv/kott
```

Koodilõik 3.8. Failide migreerimine

Kopeerimise käigus võivad muutuda failide ja kaustade õigused, kuid kuna rakendus ja veebiserver töötavad ainult kindlate kasutajate õigustes, siis on tarvis need taas paika saada kasutades *chown* käsku (koodilõik 3.9). Nii veebiserver kui ka rakendus peavad olema võimelised ligi pääsema neile etteantud kohtadele failide salvestamiseks või kasutajale sisu väljastamiseks, kuid õiguste jagamisega peab olema ettevaatlik, et vastavad süsteemi osad saaksid ligi ainult neile ettenähtud kaustadele/failidele, vastasel juhul võivad lekkida näiteks rakenduse seadistusfailid või logid.

```
# Kaustadel panna omanikud paika
chown -r nimi:nimi /srv/kott/uploads
chown -r nimi:nimi /srv/kott/reviews
chown -r nimi:nimi /srv/kott/frontend
chown kott:kott /srv/kott/kott.jar
```

Koodilõik 3.9. Migreeritud failide kasutaja õigused paika

Andmebaasiserverit kasutati samuti eelnevalt süsteemiadministraatorite mängumaana, kus katsetati läbi baasi taastamist, uuendamist ja kohendamist uuele keskkonnale, seetõttu peab olema veendunud, et vana andmebaas oleks eelnevalt täielikult eemaldatud (koodilõik 3.10).

```
# DB kontroll (kui on testimisest jäänud dop DB tuleks see eemaldada)
ssh 10.40.10.76
sudo -i
mysql
drop database dop;
create database dop;
```

Koodilõik 3.10. Andmebaasi puhastamine

Andmebaasi kolimiseks peab kopeerima eelnevalt loodud tõmmise uude andmebaasiserverisse sarnaselt nagu kopeeriti rakendusserverisse vajaminevad kaustad. Peale tõmmise kopeerimist uude serverisse saab sooritada taastamise. Lisaks tuleb täiendada andmebaasis olevaid väljasid uuele ülesehitusele vastavaks (koodilõik 3.11).

```
# Andmebaasi taastamine
mysql < /srv/mysqldump/$(date +%Y%m%d)-dump.sql

# Andmebaasi uuendamine
mysql_upgrade

# Pisikohendus
update mysql.proc set definer =  'kott@%';
```

Koodilõik 3.11. Andmebaasi taastamine ja seadistamine

Kõige viimaseks tegevuseks on rakenduse ja veebiserveri taaskäivitamine uues keskkonnas (koodilõik 3.12), seejärel saab kontrollida, kas esileht kuvab oodatut ülesehitust veebilehes ning kas andmebaasi ühendused töötavad planeeritult, mõlemat

saab kontrollida liikudes lehel ringi. Samal ajal tuleb serveri seest jälgida rakenduste logisid ning otsida veateateid või hoiatusi. Täispikk konstruktor asub lisas 1 ja lisas 2.

```
# Kui DB tehtud, rakenduse osad käima
ssh 193.40.55.70
sudo -i
sc start kott
sc start httpd
```

Koodilõik 3.12. Teenuste käivitamine

Kontrollile järgneb informatiivse kirja kirjutamine arendajale ja projektijuhile eduka uuendamise kohta, kus soovituslikult tuleb mainida nende edasised tegevused ja versioonidega seotud muudatused, et hiljem oleks võimalik kirjade ajaloo põhjal teha kokkuvõtte tegevustest. Loodud uued serverid tuleb lisada monitooringusse ning kõik teadmised ja tegevused, mis said omandatud või tehtud, tuleb dokumenteerida EENeti arendusveebi.

3.2.1 Õnnestumised ja ebaõnnestumised

Rakenduse töö kontrollimisest võtsid osa kõik osapooled - süsteemiadministraatorid, projektijuht ja arendajad. Kontroll on süsteemiadministraatorite joaks pinnapealsem, kuna osakond tegeleb paljude infosüsteemide haldamisega, kuid seda ainult serveri sisupoolelt, seega ei ole administraatoritel teadmiseid hindamaks veebiliidese tööd. Kui esileht tuleb üles, reageerib nupuvajutustele ning sisselogimine toimib, loeb osakond süsteemi kasutuskõlblikuks ning järg antakse üle projektijuhile. Projektijuht haldab üldjuhul vaid ühte infosüsteemi ning on seetõttu ka süsteemi tööga süvitsi kursis, tema vaatab üle täpsemad funktsioonid ning soovi korral kaasab teströhma kontrollimaks, kas kõik töötab vastavalt ülesehitusele, probleemide ilmnemisel kaasatakse arendajad.

Testkeskkonna kolimisele kulus viis tundi, mille käigus vahetati välja baasserver täielikult uue vastu. Keskkonna peatamine uuenduseks lepiti kokku üks päev enne ning planeeritavaks katkestuse pikkuseks planeeriti neli tundi. Uuenduse ebaõnnestumise puhuks oli valmis seatud tagavaraplaan vanadele serveritele tagasikolimiseks. Keskkonna testimist tehakse kuni kõik osapooled nõustuvad, et süsteem töötab nii nagu planeeritud ning seejärel hakatakse planeerima toodangukeskkonna uuendust.

Kehva kommunikatsiooni tõttu läks kaduma väga palju aega, kuna uuendus toimus suvel ja varasügisel, siis paljud töötajad olid puhkusel ning alati ei olnud olemas asendajat, kes teemaga kursis oleks. Lisaks tekkis ajapikku ka mitmeid arusaamatusi, mis suurendasid ajakulu veelgi.

Palju aega kulus ajatsooni ülesehituse erinevusest tingitud probleemidele. Andmebaas, operatsioonisüsteem ja indekseerija olid esmapilgul vaadates seadistatud samasse ajatsooni, kuid kuvatav informatsioon oli tegelikkuses kohalikust ajast nihkes paar tundi, millest tulenevalt ei käitunud süsteem nii nagu vaja. Lõplikuks lahenduseks otsustati lisada Solri teenuse käivituslausele juurde ajatsooni ülesehituse informatsioon.

Toodangukeskkond sujus paigalduse poolest sujuvamalt ning ajakulu oli võrreldes testserveri nelja ja poole tunniga kolm tundi. Küll aga ilmnes toodangu puhul välja palju probleeme, mis testis jäid kõikidel osapooltel märkamata. Sellega seoses tuli uuendusele järgevatel nädalatel nii test- kui ka toodanguserveritele suurtes kogustes kiirpaiku (ingl k *hotfix*), mille abil parandati vigu alates visuaalsetest fondivärvidest kuni Java süntaksite parandusteni.

KOKKUVÕTE

Käesoleva rakenduskõrgharidustöö eesmärgiks oli vahetada vanad infosüsteemi baasserverid uuendatud ja ajakohastatud virtuaalmasinate vastu. Eesmärgi saavutamiseks loodi uuendusplaan, arutati koostöös projektijuhi ja arendajatega uuendamise võimalusi, tarkvaraversioonide sobivust ning eeldatavat aja- ja ressursikulu. Kuigi süsteemiadministraatorite plaaniks oli kohe algusest peale minna uue operatsioonisüsteemi teed, siis lõppude-lõpuks sai antud lõputöö valmida sellisel kujul tänu projektijuhi ja arendusmeeskonna koostöövalmidusele.

Täielikult uute serverite ehitamine võimaldas jagada põhieesmärgi omakorda kolmeks väiksemaks, kuid siiski väga olulisteks punktideks, mille tulemusena minimaliseeriti turvaaukude olemasolu eemaldades vanu ja ebavajalikke faile/komponente, parandati hallatavust failide asukohtade piiramise näol ning loodi oluline dokumentatsioon ja mängiti reaalselt läbi taastetestimine.

Töö autor on tulemusega väga rahul. Tegemist oli tudengi jaoks teise ülesandega peale praktikale asumist ning kuna tudeng asus praktikale põhimõtteliselt olematute Unixiga seotud teadmistega, siis oli lõputöö kohati üsna keeruline, kuid andis palju väärtuslikke teadmiseid mitmekülgsetel teemadel. Antud projekti raames õppis tudeng tundma MariaDB andmebaase, Debiani ja Arch Linuxi operatsioonisüsteeme, veebiservereid ja palju muud, mis tuli järgnevate ülesannete lahendamisel suureks kasuks. Kõike dokumenteerides tekkis tudengil uus harjumus igat tegevust üles kirjutada, tänu millele tekkis palju juhendeid (näiteks serverite ehitamine nullist, varunduse ja andmebaaside initsialiseerimine), mida on võimalik kasutada tulevikus automatiseeritud skriptide loomisel või lihtsalt korduvate ülesannete kiirema lahenduse nimel. Lisaks isiklikele teadmistele suutis tudeng pakkuda asutusele kasu ehitades turvalised ja optimeeritud serverid ning aidates läbi viia taastetestimist.

Lõputöö projekti tulemusena loodud serverid on praeguseks täismahus kasutusel. Serverite ehitamise käigus loodud juhendeid kasutati uuendustele automaatpaigaldusskripti kirjutamiseks. Alates 2020. aasta sügisest arendatakse E-koolikotti uut visuaalset poolt ning muutuvad mõningad tarkvaranüansid, kuid autori poolt ehitatud serverid jäetakse edasi kasutusele.

SUMMARY

The aim of this thesis was to replace old and outdated information system with new, up to date virtual machines. To fulfill the thesis objective, a basic upgrade plan was made, which was then presented to the project lead and developers to summarize student's intentions and to find the best way to make the upgrade process as smooth as possible, figure out software limitations and plan an estimate necessity for time and resources. Even though the student's team had planned to switch out the old operating system from the very beginning, then this thesis could be presented in this way only thanks to the project lead's and developer's full readiness to cooperate.

Full upgrade for all servers allowed dividing the main objective into three smaller, yet still very important goals - minimize the existence of vulnerabilities by removing old and unnecessary files/components, improve manageability by limiting folders where files can be written and disaster recovery was played through and all necessary documentation was written.

The author is pleased with the outcome. This thesis was the student's second project since becoming an intern at Information Technology Foundation for Education. Student started with an essentially nonexistent skillset for system administration, which made this thesis occasionally quite challenging but gave a substantial amount of expertise in various fields. During this thesis, student gained new experiences regarding MariaDB databases, Debian and Arch Linux operating systems, web servers, and much more, which all became beneficial in future tasks. During this long learning process student also acquired a habit to document every step and command, which resulted in multiple new manuals regarding building servers from the beginning and initialization of backups and databases. These manuals can be applied to automated scripts or to make repetitive tasks less challenging by copy-pasting required commands. Overall, the highlight of this thesis was to provide new, secure, and optimized to the information system.

The servers that were made as the result of this thesis are currently fully in use. The manuals that were made during the process were used to make an automatic script for updating E-koolikott servers. As of autumn, 2020 a new visual side is being built and some software is being switched out, but servers themselves will stay unchanged.

KASUTATUD KIRJANDUSE LOETELU

1. Infosüsteemide turvameetmete süsteem ISKE. [*Online*]
https://www.ria.ee/et/kuberturvalisus/infosusteemide-turvameetmete-susteem-iske.html (10.07.2020)

2. Mark Kedgley. The problem with running outdated software. [*Online*]
https://www.newnettechnologies.com/whitepaper/Outdated-Software-Whitepaper.pdf (07.11.2020)

3. E-koolikoti korduma kippuvad küsimused. [*Online*]
https://e-koolikott.ee/kkk (10.07.2020)

4. Meeting Minimum Hardware Requirements. [*Online*]
https://www.debian.org/releases/stretch/i386/ch03s04.html.en (28.08.2020)

5. Advice For New Users On Not Breaking Their Debian System. [*Online*]
https://wiki.debian.org/DontBreakDebian (07.18.2020)

6. Arch compared to other distributions. [*Online*]
https://wiki.archlinux.org/index.php/arch_compared_to_other_distributions (07.12.2020)

7. Infosüsteemide talituspidevuse poliitika. [*Online*]
https://www.ria.ee/sites/default/files/content-editors/ISKE/lisa1.03.infosusteemide_talitluspidevuse_poliitika.doc (02.09.2020)

8. Serverivabrik. [*Online*]
https://arendus.eenet.ee/w/arendusveeb/serverivabrik/ (30.06.2020)

9. Alpine Requirements. [*Online*]
https://wiki.alpinelinux.org/wiki/Requirements (08.12.2020)

10. Kernel Module Loading. [*Online*]
https://wiki.archlinux.org/index.php/Kernel_module#Loading (08.12.2020)

11. Nineveh Madsen. OpenVPN Study Reveals Employee Behaviors Have A Direct Impact On Corporate Cybersecurity Effectiveness. [*Online*]
https://www.privatetunnel.com/news/employee-passwords-cybersecurity-study/ (14.07.2020)

12. Jake Fellows. Is Splitting off Resources for Your Database Right for You? [*Online*] https://www.liquidweb.com/blog/is-splitting-off-resources-for-your-database-right-for-you/ (04.07.2020)

13. Virtuaalprivaatserveri teenus VIPS. [*Online*] https://www.eenet.ee/EENet/VIPS.html (14.04.2020)

LISAD

Lisa 1 Rakendusserveri täispikkuses konstruktor

```
# OS: arch
# BUILD_VER: 3

# Firehol and allow ssh access for tunnel for developers only
sf_build partial/firehol
cp -v $BD/ssh.conf /etc/firehol/public/
cp -v $BD/koolitaja.conf /etc/firehol/public/

# Backup
sf_build partial/borg

# Adminnid
sf_build partial/admins

# Apache
sf_build partial/apache
cp -v $BD/httpd-kott.conf /etc/httpd/conf/

# kott user
useradd -c Kott -d /srv/kott -U -o -u 501 -s /bin/bash kott

# MariaDB clients
sf_install_pkg mariadb-clients

# Git
sf_install_pkg git

# JDK
sf_install_pkg jre8-openjdk
cp -v $BD/kott.service /etc/systemd/system/
systemctl enable kott.service

# Solr
sf_arch_install_aur_pkg solr
ln -vs /srv/solr/logs /opt/solr/server/logs
ln -vs /srv/solr/dop /opt/solr/server/solr/dop
systemctl enable solr
echo 'SOLR_TIMEZONE="Europe/Tallinn"' >> /opt/solr/bin/solr.in.sh
```

```
# Allow more files to be opened for solr
mkdir -vp /etc/systemd/system/solr.service.d/
cat <<EOT > /etc/systemd/system/solr.service.d/override.conf
[Service]
LimitNOFILE=65000
LimitNPROC=65000
EOT

# NFS, andmete vastuvõtmiseks
sf_install_pkg nfs-utils

# vajame mailcapi, et tekiks /etc/mime.types fail
sf_install_pkg mailcap

# Motd
cat >> /etc/motd <<'EOT'

  e-Koolikott

EOT
```

Lisa 2 Andmebaasiserveri täispikkuses konstruktor

```
# OS: arch
# BUILD_VER: 3

# Proxy
sf_build partial/proxy

# Backup
sf_build partial/borg
cp -v $BD/pre-backup-mariadb.sh /root/borg/

# Adminnid
sf_build partial/admins

# MariaDB
sf_build partial/mariadb

# Kui tegemist on TEST süsteemiga
if [ "$SF_SERVER_XNAME" = 'ekoolikotttestdb' ]; then

# NFS
sf_build partial/nfs-server
cat >> /etc/exports <<'EOT'
/srv/mysqldump 10.40.10.77/32(rw,sync,no_root_squash,no_subtree_check)
EOT

fi

# Kui tegemist on LIVE süsteemiga
eif [ "$SF_SERVER_XNAME" = 'ekoolikottdb' ]; then

# NFS
sf_build partial/nfs-server
cat >> /etc/exports <<'EOT'
/srv/mysqldump 10.40.10.75/32(rw,sync,no_root_squash,no_subtree_check)
EOT

fi

# Motd
cat >> /etc/motd <<'EOT'

  e-Koolikott DB

EOT
```