TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Gert Kosenkranius

TARKVARA AUTOMAATPAIGALDUSE SEADISTAMINE PÕHJA-EESTI REGIONAALHAIGLA NÄITEL

diplomitöö

Juhendajad: Kristjan Hinn
Bakalaureus

Siim Vene
Magister

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Gert Kosenkranius

16.01.2020

Annotatsioon

Käesoleva töö eesmärgiks on luua skriptid, mida kasutada Bamboo paigaldusraamistikus, et teostada paigaldusi erinevatesse keskkondadesse. Selleks tuli analüüsida erinevaid vajadusi ning nendele toetudes panna paika tingimused, millele loodavad skriptid peavad vastama.

Skriptide loomise käigus selgus vajadus Bamboo seadistuste ja lisafailide järele. Loodi kaks skripti ja abifailid. Seadistati Bamboo paigaldusraamistiku.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 24 leheküljel, 6 peatükki, 10 joonist, 0 tabelit.

Abstract

Configuring Automatic Software Deployment Based on the Example of North Estonian Medical Centre

The purpose of this thesis ist o create scripts, which can be used in Bamboo deployment framework to deploy software in different enviroments. For that, there was an analysis regarding different requirements, and leaning on that, there was established a set of conditions, on wich the script to be creaded, had to answer.

In the process of creating scripts, the need for configuring Bamboo and additional files, arose.Two diferent scripts were created along additional files. Bamboo was configured.

The thesis is in estonian and contains 24 pages of text, 6 chapters, 10 figures, 0 tables.

Lühendite ja mõistete sõnastik

Artefakt	*Lähtekoodi kokku ehitamisel tekkiv failide kogumik.*
Bamboo agent	*Programm, mis asub serveril ning teostab Bamboo keskse serveri poolt saadetavaid tegevusi.*
Bitbucket	*Atlassian toode, koodirepositoorium. Sellest Bamboo võtab koodi, mida kokku ehitada.*
Instants	*Ühe süsteemi esinemine samas keskkonnas mitmel serveril*
JFrog	*Atlassiani toode, mõeldud artifaktide hoidmiseks. Siin hoitakse Bamboo poolt kokku ehitatud koodi.*
Keskkond	*Serverite kogumik, mis on vajalik PERH-i infosüsteemi käitamiseks.*
Koormusjaotur	*Server, mis jaotab temale suunatud pöördumised teenusserverite vahel ära.*
Mikroteenused	*Meetod teenuste arhitektuuriks, kus igal teenusel on kanda võimalikult väike äriloogiline ülesanne.*
Paigaldus	*Artefakti viimine keskkonda.*
PowerShell	*Windowsi skriptimiskeel, milles skriptid kirjutatud*
Repositoorium	*Koht, kus hoitakse koodi või artefakte. Hoiab ise muudatuste ajalugu. Sealt on võimalik mistahes ajahetke koodi või artefakte välja võtta.*
Robocopy.exe	*Windowsi käsurea programm, kasutatakse failide kopeerimiseks ja sünkroniseerimiseks. Olemas igas Windowsi keskkonnas vaikimisi.*
SC.exe	*Windowsi käsurea programm, mida kasutatakse teenuste juhtimiseks. Olemas igas Windowsi keskkonnas vaikimis*
Teenus	*Osa infosüsteemist, mis täidab kindlat ülesannet.*

Sisukord

1 Sissejuhatus 9
2 Probleemi püstitus 10
2.1 Probleem 10
2.1.1 Lahendus 10
2.2 Teenused ja veebid 11
2.2.1 Teenused 11
2.2.2 Veebid 11
2.3 Kasutusel olevad tooted 12
2.3.1 Koodihoidla 12
2.3.2 Projektijuhtimise vahend 12
2.3.3 Appdynamics 12
2.4 Keskkonnad 13
2.5 Konfiguratsiooonifailid 14
2.6 Teenuste nimetamine 14
3 Analüüs 16
3.1 Vajadused 16
4 Lahenduskäik 19
4.1 Bamboo 19
4.1.1 Süsteemi arhitektuur 19
4.1.2 Integratsioonid 20
4.1.3 Agendid 21
4.1.4 Muutujad 21
4.1.5 Seadistused 21
4.2 Loodud skriptid 24
4.2.1 Süsteemide loend 24
4.2.2 Teenuste paigaldamise skript 24
4.2.3 Veebi paigaldamise skript 29
4.3 Tulemused 30
5 Edasised tegevused 31

6 Kokkuvõte 32
Kasutatud kirjandus 33
Lisa 1 – Teenuste paigaldamise script 34
Lisa 2 – Veebi paigalduse script 43

Jooniste loetelu

Joonis 1. Erinevate keskkondade seis 20
Joonis 2. Ühe teenuse paigalduskeskkonnad Bamboos 22
Joonis 3. Tegevused 23
Joonis 4. Agentide määramine 23
Joonis 5. Ühe teenuse näide XML failis 24
Joonis 6. Skripti parameetrid 25
Joonis 7. Muutujate algväärtustamine 25
Joonis 8. Koodi osa, mis otsustab kumba funktsiooni rakendada 26
Joonis 9. Näide abifailist New-to-Old.ps1 28
Joonis 10. Funktsioonide UML diagramm 29

1 Sissejuhatus

Põhja-Eesti Regionaalhaiglas (edaspidi PERH) on töötajatel kasutamiseks 120-st teenusest ja 29-st veebist koosnev infosüsteem, mis on arendatud PERH-i arendajate või arenduspartnerite poolt. Kuna selle süsteemi arendamine on igapäevane töö, siis selle raames peab toimuma ka sagedasti esinevaid paigaldusi, mis parandavad tekkinud vigu või lisavad uut funktsionaalsust. Varasemalt viisid paigaldusi läbi arenduspartnerid ja arendajad ning nende paigalduste käigus tuli tihti ette olukordi, kus peale paigaldust tekkisid infosüsteemis tõrked. Lisaks puudus ka ülevaade, milline versioon koodist millises keskkonnas on.

Antud olukorra parandamiseks võeti kasutusele Bamboo paigaldusrakendus, mille abil saab paigaldusi läbi endisest kiiremini ja väiksema arvu vigadega. Lisaks annab Bamboo selge ülevaate sellest, kes, millal ja millise koodi ning millisesse keskkonda paigaldanud on. Tagamaks, et Bamboo viiks korrektselt läbi paigaldusi, on vaja see õigesti seadistada ning luua skriptide kogumik, mis vastavalt vajadusele paigaldusi teostaks.

Töö lõppeesmärgiks on luua vajalikud skriptid ja seadistused, mille abil Bamboo saab artefakte erinevatesse keskkondadesse paigaldada. Töö tulemit on võimalik kasutada ka teistes ettevõtetes, kus soovitakse Bambood kasutusele võtta. Loodud skriptid ja seadistused on mugandatavad teistsuguste keskkondade tarbeks.

Eesmärgi saavutamiseks uuriti PERH-is kasutusel olevaid tehnoloogiad ja lahendusi ning tehti analüüs, millised tingimused on paigalduste edukaks läbiviimiseks vaja täita. Sellele järgnes skriptide kirjutamise ja testimise faas, kuni jõuti soovitud tulemuseni. Hetkel on PERH-is olukord, kus saab kiirelt ja mugavalt läbi viia paigaldusi erinevatesse keskkondadesse ja vajadusel ka neid tagasi võtta.

2 Probleemi püstitus

Selles peatükis kirjeldatakse paigalduste olukorda ja antakse ülevaade PERH-is olevatest tingimustest ja toodetest, millega tuleb arvestada antud töö eesmärki täites.

2.1 Probleem

PERH-is olev infosüsteem on mahukas ja keeruline. Selle töös hoidmiseks ja parendamiseks on vajalik teha süsteemides muudatusi. Muudatuste haldus on puudulik. Ei ole võimalik kiirelt ja mugavalt aru saada, missugune kood parajasti kuhu paigaldatud on. Parimaks meetodiks versioonide tuvastamisel on failide modifitseerimisaja jälgimine. See pole aga üheselt mõistetav ning põhjustab asjasse puutuvatele osapooltele liigset ajakulu. Samuti on raske, kui mitte võimatu, tuvastada, kas mõni konkreetne koodimuudatus on juba rakendunud või mitte.

Paigaldusi teostavad arendajad käsitsi, kopeerides manuaalselt kokku ehitatud ja segaste nimekujudega paigalduspakkidest faile ise serveritele süsteemide asukohta. Selle käigus tekivad vead, kus näiteks mõni fail jääb kopeerimata või kopeeritakse failid vale süsteemi asukohta, sest kaustade nimed on sarnased. Puudub korralik ülevaade, kes ja millal paigaldusi teostab ning, mis koodimuudatused paigalduspakkidesse kokku ehitatakse. Enne paigaldust ei tehta konkreetsest süsteemist alati varukoopiat ning tagasivõtmise vajaduse ilmnemisel võtab protsess aega 5-15 minutit, mille ajal on rikkega süsteem töös. Puudub ajalugu, millest oleks konkreetselt näha, millised failid kuhu kopeeriti. Ei ole konkreetselt määratud, kes tohib paigaldusi teostada. Arendajatel on ligipääs tootmiskeskkonna serveritele ning administraatoritel ei ole ole alati teada, mis nende poolt hallatavatel serveritel toimub, sest infovahetus on puudulik.

2.1.1 Lahendus

Eelpool loetletud probleemide lahenduseks otsustas PERH-i süsteemiarhitekt kasutusele võtta Bamboo paigaldusraamistiku, mis integreerub teiste PERH-is kasutusel olevate süsteemidega, millega koodikirjutamist, -ehitamist ja -paigaldamist juhitakse. Mainitud

süsteemid on kirjeldatud alamjaotuses 4.1.2 Autori töö oli koostöös arendusmeeskonnaga välja valitud lahenduse juurutamine. Selleks oli vaja välja töötada metoodika, kuidas juhtida paigaldusi erinevatesse keskkondadesse vastavalt vajadustele ning paigaldusskriptide loomine, mis teostaks vajalikke tegevusi. Valitud lahendus teostab automaatika abil tegevusi paigalduse käigus, mida varasemalt tehti käsitsi. Autor lähtus oma töös Martin Fowleri artiklist „Continuous Integration“ [1] ja Bamboo dokumentatsioonist [2].

2.2 Teenused ja veebid

2.2.1 Teenused

PERH-i infosüsteem on loodud mikroteenuste põhimõttel, mis tähendab seda, et süsteem koosneb paljudest väikestest osadest, millest igaüks täidab enda kindlat rolli ning koondab enda alla võimalikult väikse osa funktsionaalsusest [3]. Suurim eelis selle süsteemi juures on võimalus paigaldada erinevaid infosüsteemi osasid teineteisest sõltumatult, tuleb vaid tagada süsteemi osade omavaheliste liideste muutumatus. Teine eelis on üldine süsteemi vastupidavus. Kui osad teenused ei ole võimelised oma ülesandeid rikke korral täitma, siis teised saavad oma tööd jätkata ning kogu infosüsteem ei ole sellest rikkest häiritud.

PERH-is on infosüsteem, mis koosneb 120-st teenusest. Valdav enamus teenustest on paigaldatud mitme instantsiga, ehk sama teenus paikneb mitmel serveril. Need on teenused, mis pakuvad funktsionaalsust. Iseseisvalt need teenused oma tegevusi ei tee, vaid vajavad, et nende poole pöördutakse ning funktsionaalsus välja kutsutakse. Puudub võimalus, et korraga kirjutavad andmebaasi samale väljale mitu teenust. Ülejäänud teenused on intervallteenused, mida tohib käitada korraga vaid ühel serveril, sest nende tegevused käivituvad kindlate intervallide tagant. Kui need oleks paigaldatud mitmele serverile, eksisteeriks võimalus, et mitu teenust kirjutab andmebaasis samale väljale samaaegselt. Tulemuseks on vead ja andmete terviklikkuse kadu.

2.2.2 Veebid

Erinevaid veebikeskkondi on 29. Töötajate põhiliseks töövahendiks on erinevad veebikeskkonnad, kus saab läbi viia erinevaid patsientidega seotud tegevusi nt. proovide registreerimine, patsientide voodikohtade määramine, söökide tellimine, analüüside

tellimine, konsiiliumi otsuste koostamine, visiitide planeerimine ja registreerimine. Kuna veebid on töötajatele esimeseks „aknaks“ infosüsteemi, on väga kriitiline, et nende paigaldused kulgeksid sujuvalt ja tõrgeteta. Veebid paiknevad ühel veebiserveril, nende ette pole koormusjaoturit pandud, sest hetkel ei sattu ükski veebidest sellise koormuse alla, mis selle tööd märgatavalt mõjutaks. Tulevikus on plaan ka nende serverite ette implementeerida koormusjaotur, et vähendada riski ohtu ja teha paigaldused ja süsteemiuuendused mugavamaks.

2.3 Kasutusel olevad tooted

2.3.1 Koodihoidla

Rakenduste lähtekoodi hoidmiseks on ühine koodihoidla. See tagab selle, et arendajatel on võimalikult mugav ja lihtne lähtekoodiga tööd teha ning ei teki olukorda, kus erinevad arendajad teineteise muudatusi üle kirjutaks. Selleks tooteks on osutunud firma Atlassian toode Bitbucket, mis omab liidestust projektijuhtimise ja paigalduste rakendustega. Kuna kõik koodiga seotud rakendused on ühe tootja omad, tagab see võimalikult hea integratsiooni erinevate rakenduste vahel.

2.3.2 Projektijuhtimise vahend

Projekte juhitakse PERH-is firma Atlassian tootega Jira. Sinna kirjeldatakse kõik vead ja arendusvajadused, millest tekivad arendajatele ülesanded. Ülesannete juurde saavad arendajad üles märkida oma töötunde ja koodimuudatusi. Iga ülesande juures on näha, millised koodimuudatused on selle ülesande raames tehtud. Koodimuudatustest kompileeritakse peale nende tegemist paigalduspakid, mis tuleb erinevatesse keskkondadesse paigaldada. Paigalduspakke hoitakse JFrog artefakti repositooriumis.

2.3.3 Appdynamics

Teenuste, veebide ja andmebaaside vahelise suhtluse kaardistamiseks on kasutusele võetud firma Appdynamics samanimeline toode. See võimaldab administraatoritel, tootejuhtidel ja arendajatel näha süsteemide vahelist liiklust ning tekkinud vigu. Appdynamics-i agent haakub kooditasandil rakendustega ning annab ülevaate tehtavatest tegevustest. Näha on iga pöördumine, mille süsteem mõne teise süsteemi poole teeb ning see aitab kaardistada PERH-i infosüsteemi. Kuna eraldi on välja toodud vead ning aeganõudvad pöördumised, siis on sellest abi ka äriloogika- ja programmeerimisvigade

tuvastamisel. Suurima tõhususe kasvu annab Appdynamicsi võimalus kuvada, millised teised süsteemid ühe konkreetse süsteemi poole pöörduvad. Milliste teiste süsteemide poole konkreetne süsteem ise pöördub, on tuvastatav koodist või konfiguratsioonifailidest, kuid vastupidise nägemiseks peab kõikide teiste süsteemide koodi ja konfiguratsioonifailid läbi käima ning see on ajamahukas. Pöördumiste nägemisest on abi näiteks siis, kui soovitakse süsteemi ümber kolida teisele serverile, sest siis on näha, millistes teiste süsteemides tuleb muudatusi teha, et nad oleksid võimelised süsteemile tema uues asukohas pöörduma.

2.4 Keskkonnad

Keskkonna all peetakse silmas kogu haigla infosüsteemi tööks vajalike serverite kogumit. Tuginedes levinud parimatele praktikatele [4] on testimiseks ja vigade vältimiseks on LIVE keskkonnast tehtud kaks klooni. PERH-is on kasutusel kolm erinevat keskkonda, kuhu paigaldusi tehakse. Neid nimetakse TEST, PRELIVE ja LIVE.

Esimeseks keskkonnaks, kuhu paigalduspakke paigaldatakse, on testkeskkond (edaspidi TEST). Sinna on ligipääs nii arendajatel kui administraatoritel. TESTIS olevad andmebaasid sisaldavad endas LIVE keskkonnast pärinevaid andmeid, mida on anonümiseeritud, sest neid näevad ka arendajad ja teised kolmandad isikud. TEST keskkonna serverid on väiksema võimekusega, kui teistes keskkondades, sest TESTIS ei viida läbi koormusteste. Lisaks on TESTI jaoks loodud eraldi domeen ning serverid asuvad oma alamvõrgus, mis on tulemüüriga teistest eraldatud. Sedasi on tagatud, et kui seadistamisel tehakse viga, ei saa siiski ükski TESTI teenus pöörduda LIVE andmebaaside poole, sest liiklus on juba võrgu tasandil piiratud.

Edasi liiguvad samad paigalduspakid PRELIVE keskkonda. PRELIVE on identne LIVEGA, välja arvatud andmebaasis olevad andmed. PRELIVES kasutatakse LIVEST pärinevaid poole aasta taguseid andmeid. Kuna neid anonümiseeritud pole, siis on sinna keskkonda pääsemise õigused tunduvalt rangemalt kontrollitud. Kuna PRELIVE ja LIVE serverid on samade parameetritega, on võimalik PRELIVES läbi viia ka koormusteste. PRELIVE asub LIVEGA samas domeenis, aga samuti erinevas alamvõrgus, sarnastel põhjustel TESTIGA.

LIVE keskkond on see, kus toimub kasutajate igapäevane töö. Sellel keskkonnal on kõige rohkem kasutajaid. LIVE paigaldused on keerukad just seetõttu, et tihti kaasneb paigaldusega katkestus. Seetõttu tuleb kasutajaid paigaldustest ette teavitada. Osade rakenduste puhul on paigalduseks määratud aeg väga lühike, sest tegemist on kriitiliste süsteemidega, mis peavad töötama ööpäevaringselt. Näiteks EMO ja labor. Nende puhul on kriitiline, et paigaldus sujuks tõrgeteta ja võimalikult väikse katkestusega, sest pikemaaegsete katkestuste puhul võivad ohtu sattuda reaalsed inimelud.

2.5 Konfiguratsiooonifailid

Rakenduste üheks tähtsaks osaks peale programmifailide on konfiguratsioonifailid. Need on failid, kus on ära kirjeldatud rakenduse tööks vajalikud parameetrid, mis peavad olema lihtsalt muudetavad. Näiteks TCP port, millel teenus liiklust ootab ja intervallid, mille möödudes teenus tegevusi teostab. Seoses lisafunktsionaalsuse tekkimisega rakendusse on vaja aegajalt parameetreid lisada või muuta, mis tingib omakorda vajaduse muuta konfiguratsioonifaile. Selleks et näha, millised muudatused ja millal on sisse viidud, peab eksisteerima versiooniajalugu. Seepärast on PERH-is loodud süsteem, kus konfiguratsioonifaile hoitakse sarnaselt koodile koodihoidlas. Erinevatesse keskkondadesse võetakse konfiguratsioonifailid vastavatest koodiharudest ning on loodud ka võimalus, et arendajad saavad TEST haru konfiguratsioonifaile ise muuta. Konfiguratsioonifaile hoitakse programmifailidest eraldi seepärast, et sama programmikood peab liikuma keskkonnast keskkonda, aga konfiguratsioonifailid peavad olema erinevad.

2.6 Teenuste nimetamine

Kuna erinevaid teenuseid on palju, siis on tähtis, et ei tekiks segadust, millise teenusega on hetkel vaja tegeleda. Ühe teenuse poole annab viidata kasutades erinevaid nimetusi, mistõttu on tõenäosus vigade tekkimiseks suur. Näiteks: Koodihoidla repositooriumi nimi, lahenduse nimi repositooriumis, .exe nimi kompileeritud koodis, teenuse installikausta nimi, rakenduse Services.msc alt teenuse „display name“. Varem olid sagedased olukorrad, kus erinevates kohtades ei olnud nimed identsed. Selle tulemusena tekkis arusaamatusi administraatorite, tootejuhtide ja arendajate vahel, millisele teenusele

hetkel viidatakse. Kuna eksisteeris ka sarnaste nimedega teenuseid, võis ette tulla vale teenuse arendamist või paigaldamist.

3 Analüüs

Selles peatükis analüüsitakse, millised vajadused on paigalduste läbiviimiseks.

3.1 Vajadused

Paigalduste läbiviimiseks on vaja teada kolme asja: mida paigaldatakse, kuhu paigaldatakse ja kuidas paigaldatakse. Nendest esimesele ei pea vastust otsima. See antakse ette paigaldusülesandega, milles on viide konkreetsele Bamboo koodiväljalaskele. Koodiväljalaskes on kirjeldatud artefakt, mis sisaldab endas kõiki faile, mis on ehitusprotsessi käigus tekkinud. Failide asukoht salvestatakse Bamboo muutujas ning selle saab autor skriptile parameetrina ette anda.

Artefakti paigaldamise asukoht on problemaatilisem. Vastavalt süsteemile, mida paigaldatakse ning keskkonnale, kuhu paigaldatakse, on asukoht vaja täpselt ära määratleda. Kuna paigaldamiseks on vajadus kasutada võimalikult vähe skripte, mida saaks keskselt muuta, on autoril vaja luua loend, kus saaks hoida iga süsteemi kohta tema erinevates keskkondades olevate asukohtade infot. Seda infot on võimalik hoida ka Bamboos iga süsteemi küljes olevas muutujas, kuid hiljem muudatuste tegemine oleks aeganõudev ja ebamugav kasutajale. Seepärast on otstarbekam luua fail, mis sisaldab kogu infot, sest seda on hiljem võimalik mugavalt ja kiiresti muuta. Näiteks kui peaks lisanduma koormusjaoturi taha veel üks teenusserver, siis annab ühes failis asuvat loendit muuta skriptiga, mis vastava kirje igale süsteemile juurde lisab. Selleks et hoida tekkinud loendit võimalikult selgena, loob autor kaks erinevat faili, kus ühes on kirjeldatud teenused ja nende asukohad ning teises veebid ja nende asukohad. Skript peab tuvastama, mitu instantsi teenusel on vastavas keskkonnas ning õigetele serveritele vastavalt teenuse paigaldama.

Nagu eelnevalt mainitud, siis teenuste erinevad nimekujud erinevates asukohtades tekitavad olukorra, kus ei saa teenusele viidata programmaatiliselt. Sellise olukorra edaspidiseks vältimiseks otsustas autor koostöös arendajatega läbi viia teenuste ümbernimetamise. See tähendab, et autor mõtles välja nomenklatuurieeglid, millele peab teenuse viide vastama ning koostas vastavustabeli, kus on kirjeldatud teenuste endised ja

uued nimekujud. Nomenklatuuri loomisel võeti arvesse, et nimekuju saaks kasutada igas kohas, kus on vaja teenusele viidata. Seetõttu ei tohi nimekuju sisaldada endas näiteks erisümboleid. Vastav info edastati ka arendajatele, kes viisid muudatuse koodi sisse.

Sellega seoses tekkis paigaldusskriptidele uus vajadus. Skript peab tuvastama, kas serveril on teenus paigaldatud uue või vana nimekujuga. Kui teenus on uue nimekujuga paigaldatud, peab skript läbi viima tavapaigalduse, ehk peatama teenuse, kopeerima failid ja teenuse uuesti käivitama. Kui teenus aga on serveril vana nimega, peab skript paigaldama uue teenuse, vana nimega teenuse peatama ning uue nimega teenuse käivitama. Paigaldusskripti selliseks toimimiseks on vaja luua vastavustabel, kus on omavahel üks-ühele seoses vana ja uus nimekuju.

Kuna uuele nimekujule üleminek toimub erinevates keskkondades erinevatel aegadel, peab olema võimalus juhtida paigaldusi keskkondade kaupa.

Teenus peab enda tööst jätma inimloetava logi. Seetõttu on vajalik skriptides kõikvõimalikesse hargnemiskohtadesse ja tegevuste juurde lisada logimine. Kuna Bamboo salvestab standardväljundisse suunatava teksti iga paigalduse juurde ise logiks, piisab sellest, kui kasutada *PowerShelli* meetodit Write-Host ning autoril ei ole vaja eraldi logide salvestamise ja roteerimise meetodit välja töötama hakata.

Skript peab tuvastama, millisesse keskkonda parasjagu paigaldust soovitakse teostada. Selleks on kõige mõistlikum kasutada Bamboost tulenevat muutujat *${bamboo.deploy.environment},* sest selles on vajalik info juba talletatud. See tekitab omakorda vajaduse, et iga süsteemi kõik paigalduskeskkonnad nimetatakse Bamboo paigaldusplaanides üheselt, kuid see on loetavuse ja ühtse arusaama tekkimise jaoks niigi vajalik. Keskkondasid nimetatakse Bamboos TEST, PRE ja LIVE.

Paigaldatavaid teenuseid on kahte tüüpi. On vanema põlvkonna nn „legacy“ teenused ning uuema põlvkonna, nn „non-legacy“ teenused. Nende vahe seisneb selles, kuidas teenus käsitseb enda konfiguratsioonifaile. Vanema põlvkonna teenustel on vajalik hoida kogu konfiguratsioonifailis sisalduv info nn. „sisemises“ konfiguratsioonifailis, mis asub teenuse binaarfailiga samas kaustas. Uuema põlvkonna teenustes on lisatud välise konfiguratsiooni tugi, mis tähendab, et teenuse sisemises konfiguratsioonifailis on vaid viide välisele konfiguratsioonifailile ning korraga võib viidata mitmele erinevale konfiguratsioonifailile. See annab võimaluse hoida varasemalt igas konfiguratsioonifailis

identse oleva info eraldi ühes failis ning sellel muutumisel peab seda kirjeldama vaid ühes kohas, mitte igaühes eraldi. Vastavalt sellele, mis sorti on teenus, peab skript oskama esmakordsel paigaldusel teha tegevusi, et teenus saaks omale konfiguratsioonifaili.

Skript peab tuvastama, milline on teenuse staatus enne paigaldust ning jätma teenuse samasse staatusesse ka peale paigaldust. Siinkohal peab autor silmas seda, kas teenus töötab või mitte. On olukordi, kus teenused serveril ei tööta, aga neid on vaja siiski paigaldada. Teenuste paigalduseks on aga vajalik teenuse töö peatamine, et failid ei oleks kopeerimise ajal kasutuses. Ning vastavalt varasemale staatusele, peab skript peale paigaldust siis teenuse uuesti käivitama või mitte. Veebid on arendatud sedasi, et kasutuses olevaid faile pole ning nende faile võib ka töö käigus üle kopeerida.

Kuna konfiguratsioonifailid liiguvad teenusefailidest eraldi, on tähtis, et kogemata artefakti kaasa tulnud konfiguratsioonifailid ei liiguks teenuse kausta paigalduse käigus. Arenduse käigus tekkinud konfiguratsioonifailid võivad endas sisaldada valesid väärtuseid ning kuna konfiguratsioonid on keskkondade kaupa erinevad, ei tohi teenuse kaustas olevaid konfiguratsioonifaile kunagi üle kirjutada paigalduse käigus. Seetõttu tuleb enne kopeerimist kõik konfiguratsioonifailid, mis artefaktis on, eemaldada.

Skript peab vigade tekkimisel need tuvastama ning näitama kasutajale infot, mille abil on võimalik olukord lahendada, et uuesti paigaldamisel paigaldus õnnestuks. Kasutajale ei tohi mitte mingil juhul jääda muljet, et kõik õnnestus, kui seda tegelikult ei juhtunud. Bamboo märgib paigalduse ebaõnnestunuks, kui skripti väljumiskood ei võrdu väärtusega 0. Seepärast peabki skript iga kasutaja sekkumist vajava vea korral tagastama väljumiskoodiks nullist erineva väärtuse.

Teenuste ja veebide paigaldus on erinev. Teenuste puhul tuleb leida serverilt teenus, mis paigalduse ajaks peatada, veebide puhul mitte. Et hoida skripti võimalikult lihtsana, on mõistlikum luua kaks eraldi skripti veebide ja teenuste jaoks.

4 Lahenduskäik

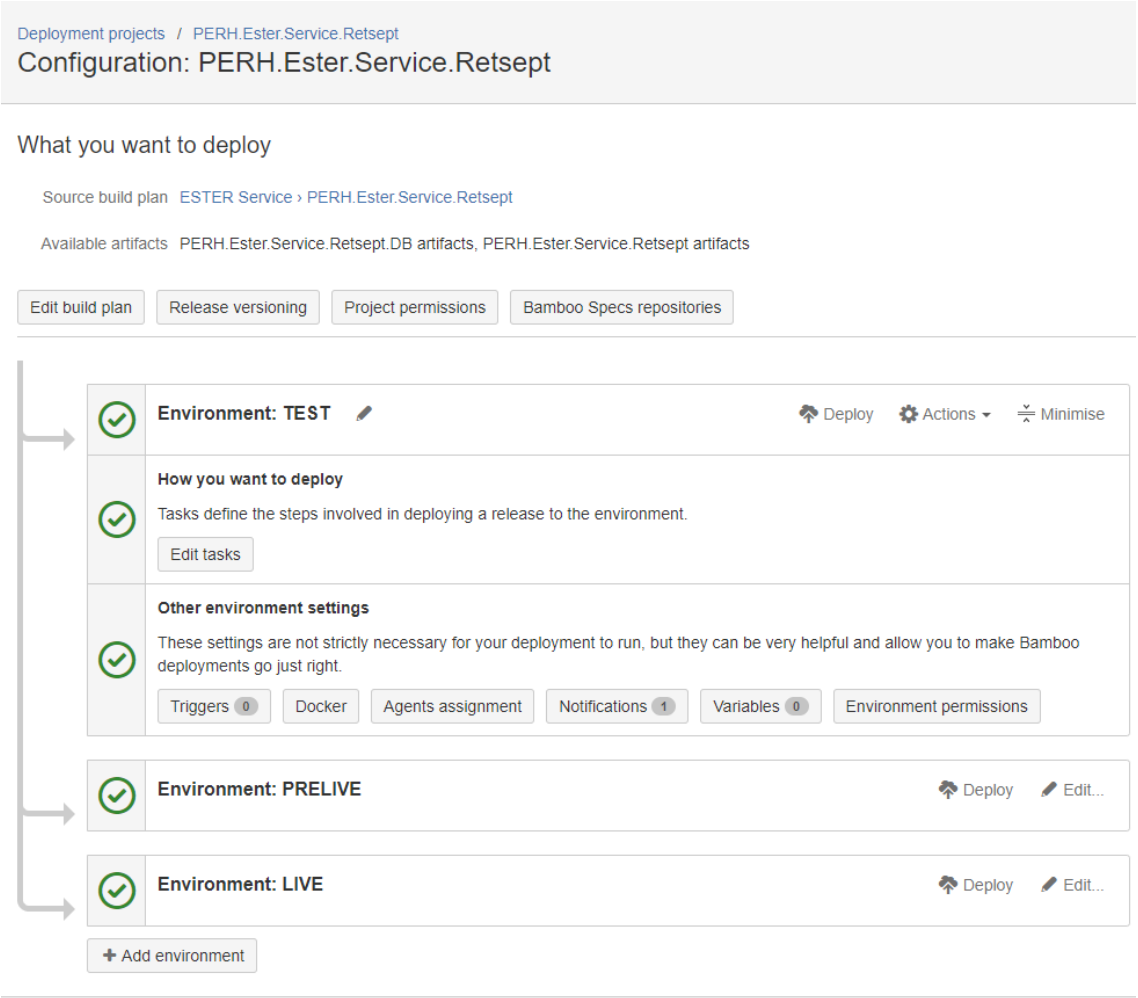
Selles peatükis kirjeldatakse Bamboo paigaldusraamistikku ja selles tehtud seadistusi ning loodud skripte.

4.1 Bamboo

Bamboo on firma Atlassian toode, mis on mõeldud abistamaks koodi kokku ehitamist ja paigaldust. Bambooga on võimalik erinevatest koodihoidlatest võtta kood, see kokku ehitada/kompileerida ning seejärel saadud artefaktid paigaldada erinevatesse keskkondadesse. Bamboo lisaväärtus on see, et selle abil annab näha, millised koodimuudatused on millistes keskkondades, millal need sinna paigaldati ja kelle poolt. Ülevaadet on võimalik näha jooniselt Joonis 1. Erinevate keskkondade seis. Lisaks on võimalused automaatseteks testideks ja paigaldusteks, mida PERH-is veel ei rakendata. See tähendab selles, et peale paigaldust saab Bamboo käivitada automaattestid, millega kontrollitakse äsja paigaldatud koodi funktsionaalsust ning probleemide mitteesinemise korral tehakse automaatselt paigaldus järgmisesse keskkonda. Selline lahendus vähendaks inimeste töökoormust ning välistaks inimlikke vigu.

4.1.1 Süsteemi arhitektuur

PERH-is koosneb Bamboo neljast serverist. Üks Ubuntul põhinev Linuxi server ning kolm Windows Server operatsioonisüsteemil põhinevat serverit. Bamboo veebirakendus paikneb Linuxi serveril, mis on ka ise Bamboo paigaldusagent. Seda küll hetkel ei kasutata, sest paigaldused toimuvad ainult Windowsi operatsioonisüsteemiga serveritele. Ülejäänud 3 on Bamboo paigaldusagendid, millest tuleb juttu allpool. Tööks vajalik andmebaas paikneb Linuxi serveril.



Joonis 1. Erinevate keskkondade seis

4.1.2 Integratsioonid

Bambood on võimalik mugavalt ja lihtsalt integreerida teiste, sama tootja toodetega. [5] Sellise integratsiooni puhul tekivad erinevatesse rakendustesse lisavõimalused ja lisainfo, mis muudavad arendamise ja projektijuhtimise tunduvalt lihtsamaks. Parim näide sellest on projektijuhtimistarkvara Jira, kus on arendajatele tehtud ülesannete juures selgelt näha, kas ja millistesse keskkondadesse soovitud veaparandus või lisafunktsionaalsus jõudnud on. See annab tootejuhtidele selgema ülevaate ning võimaldab neil tõhusamalt oma tööd planeerida. Artefaktide hoidmiseks on Bamboo liidestatud artefakti repositooriumi JFrog-iga. Sisselogimise jaoks on Bamboo liidestatud Active Directory-ga, mis annab võimaluse kasutajate õigusi juhtida AD gruppide kaudu.

4.1.3 Agendid

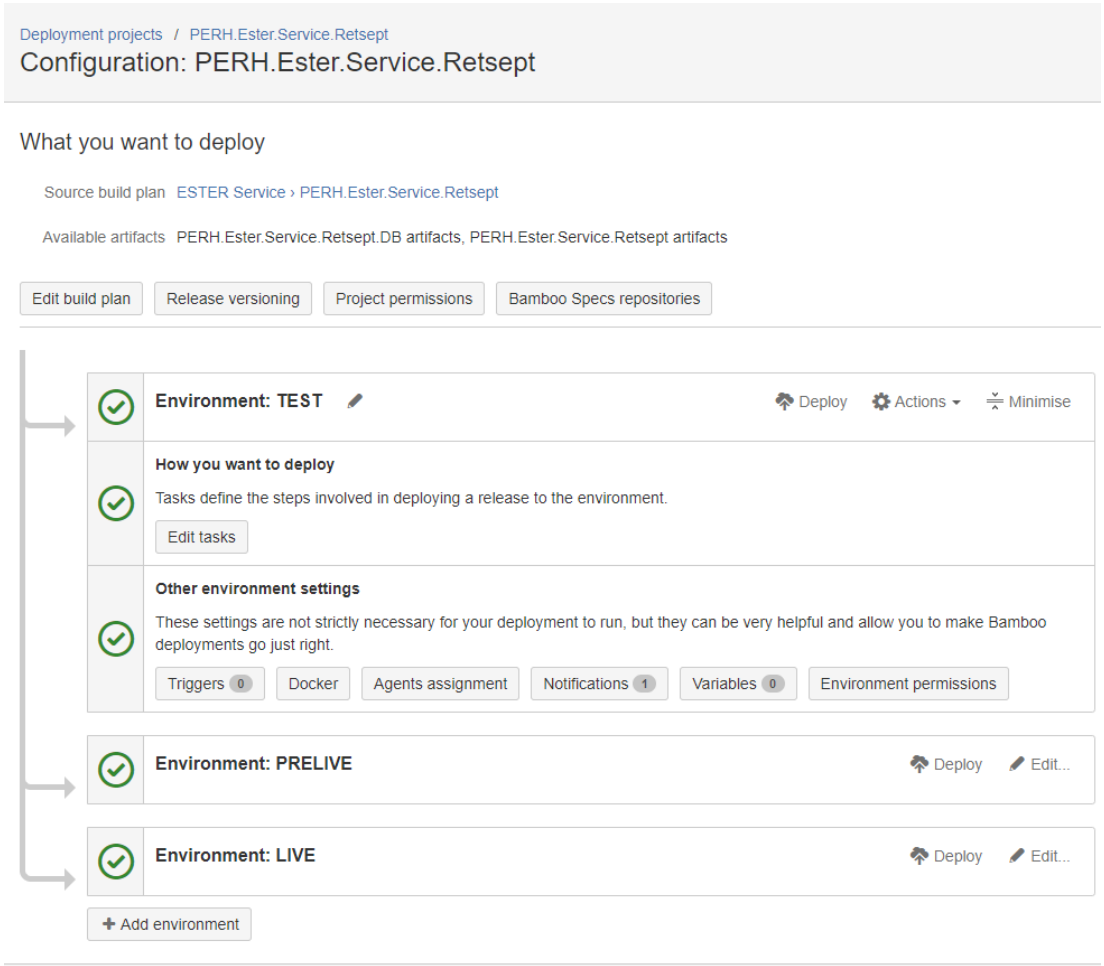
PERH-is juurutatud Bamboo lahendus on agendipõhine, mis tähendab, et rakendus vajab töötamiseks lisaks põhilisele serverile ka operatsioonisüsteemipõhiseid agente. Vastavalt selle serveri operatsioonisüsteemile, kuhu soovitakse artefakte paigaldada, peab Bambool olema sama operatsioonisüsteemiga agent-server, kuna agendil käivitatakse operatsioonisüsteemipõhiseid skripte, mis on kirjutatud nt *Bash* või *PowerShell* skriptimiskeeltes. Kuna agente kasutatakse ka koodi kokku ehitamiseks, siis vältimaks olukorda, kus kõik agendid on hõivatud, on Windowsi põhiseid agente kolm.

4.1.4 Muutujad

Bamboo-s on võimalik töö läbiviimiseks kasutada keskkonnamuutujaid, mida saab näiteks skriptidele argumentidena kaasa anda. Osad muutujad väärtustatakse töö käigus dünaamiliselt Bamboo poolt, teised aga on võimalik väärtustada Bamboo kasutajal ning need on staatilised. Dünaamiline muutuja, mida kasutatakse on näiteks artefakti asukoht agendil, sest selle asukoht genereeritakse koodi kokku ehitamise käigus. Staatiline muutuja on näiteks paigaldusprojekti nimi, mida kasutatakse õige teekonna leidmisel abifailist.

4.1.5 Seadistused

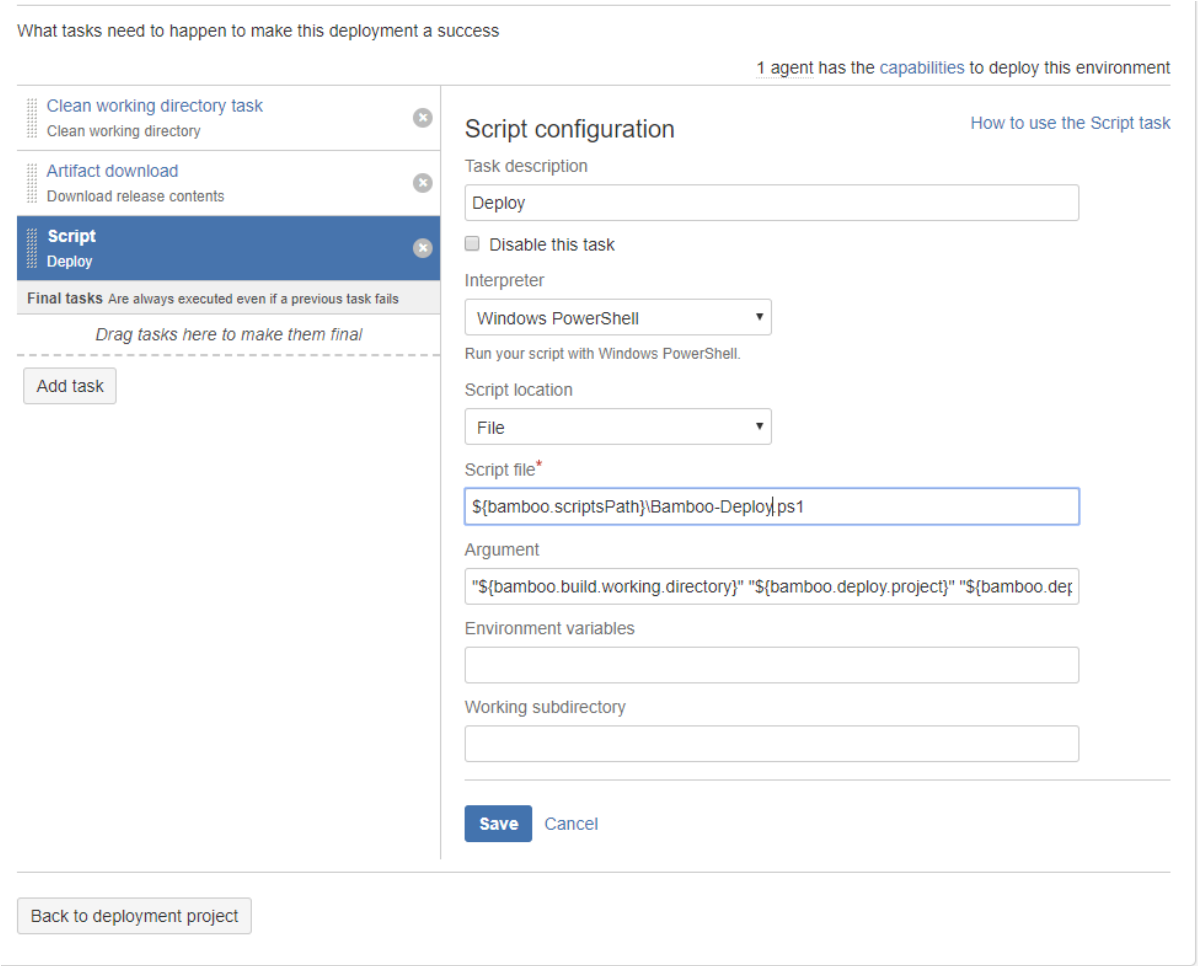
Lisaks skriptide loomisele, tuleb seadistada ka Bamboo paigaldusraamistikku. Iga teenuse puhul tuleb kõigepealt luua TESTI paigalduskeskkond, seal teha vajalikud seadistused ning seejärel kloonida ümber teiste keskkondade jaoks. Ainuke erinevus on paigalduskeskkonna nimi, mis on vastavalt keskkonnale kas TEST, PRELIVE või LIVE. Ülevaade keskkondadest on toodud joonisel Joonis 2. Ühe teenuse paigalduskeskkonnad Bamboos.



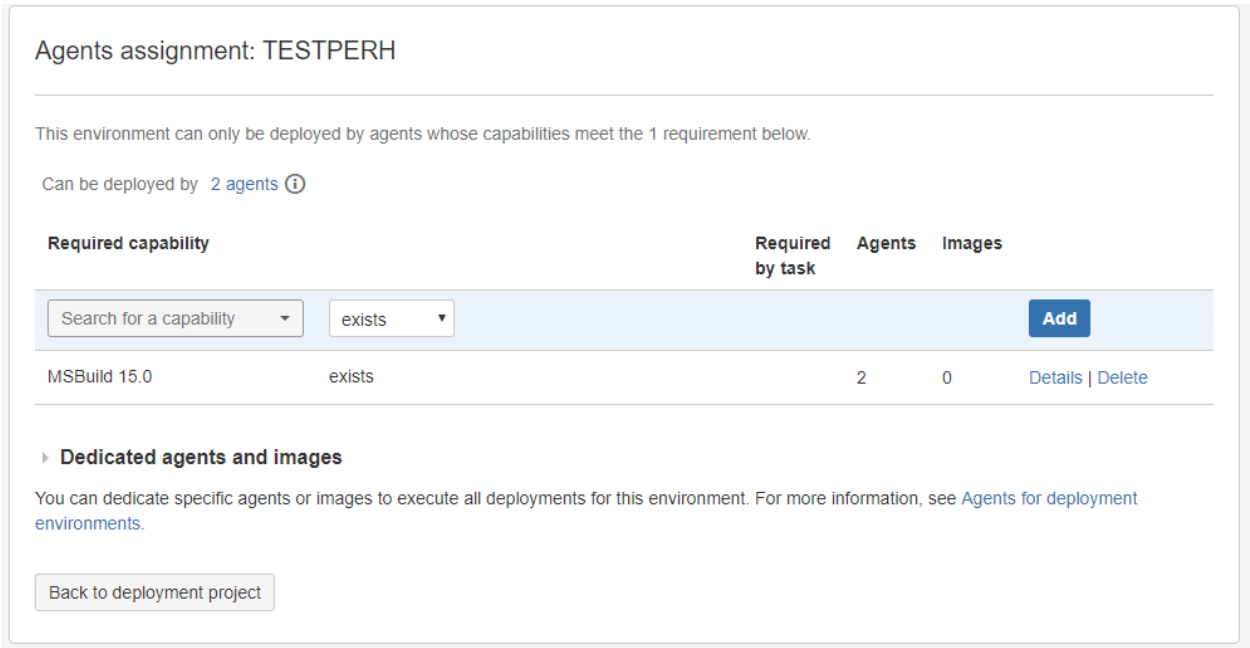
Joonis 2. Ühe teenuse paigalduskeskkonnad Bamboos

Iga paigalduskeskkonna juures on tegevused (task), millega kirjeldadakse ära Bamboo poolt tehtavad tegevused. Ühe keskkonna tegevused on välja toodud joonisel Joonis 3. Tegevused. Jooniselt on näha vajalikud seadistused, mida autor tegi tegevuste juures. Tegevuses on ära kirjeldatud skriptifaili asukoht Bamboo agentidel ning skriptile ette antavad parameetrid, mis on pikemalt lahti seletatud allpool.

Veel on ära määrata, millist Bamboo agenti paigaldusplaanis tehakse. Vastasel juhul üritab Bamboo *PowerShell* käske käivitada Linuxi keskkonnas, mis on teoreetiliselt küll võimalik, aga jääb hetkel antud töö skoobist välja. Windowsi agentide määramiseks peab Bamboo kasutama agenti, millel on defineeritud MSBuild 15.0 võimekus. Seda on näha jooniselt Joonis 4. Agentide määramine.



Joonis 3. Tegevused



Joonis 4. Agentide määramine

4.2 Loodud skriptid

Siinkohal annab autor ülevaate loodud skriptidest ja abifailidest.

4.2.1 Süsteemide loend

Selleks, et läbi viia paigaldusi, lõi autor XML-failid, kus on ära kirjeldatud kõikide teenuste ja veebide asukohad erinevates keskkondades. Ühe sissekande näide on joonisel Joonis 5.

```
<System value="PERH.Teenuse.Nimi.Service">
    <LegacyValue value="Legacy" />
    <Enviroments>
        <Enviroment value="LIVE">
            <Paths>
                <Path>\\server1.regionaalhaigla.ee\E$\WCFServices\PERH.Teenuse.Nimi.Service</Path>
                <Path>\\server2.regionaalhaigla.ee\E$\WCFServices\PERH.Teenuse.Nimi.Service</Path>
            </Paths>
        </Enviroment>
        <Enviroment value="PRE">
            <Paths>
                <Path>\\server1-pre.regionaalhaigla.ee\E$\WCFServices\PERH.Teenuse.Nimi.Service</Path>
                <Path>\\server2-pre.regionaalhaigla.ee\E$\WCFServices\PERH.Teenuse.Nimi.Service</Path>
            </Paths>
        </Enviroment>
        <Enviroment value="TEST">
            <Paths>
                <Path>\\server1.testperh.ee\E$\WCFServices\PERH.Teenuse.Nimi.Service</Path>
                <Path>\\server2.testperh.ee\E$\WCFServices\PERH.Teenuse.Nimi.Service</Path>
            </Paths>
        </Enviroment>
    </Enviroments>
</System>
```

Joonis 5. Ühe teenuse näide XML failis

Kasutades sellist XML faili ülesehitust, saab autor ära kirjeldada ühe süsteemi puhul ükskõik palju keskkondi ning igas keskkonnas võib süsteem paikneda ükskõik mitmel serveril. Uue serveri lisandumisel mingisse keskkonda on võimalik XML faili muuta skriptiga, lisades vastavale keskkonnale uue *Path* XML elemendi. Lisaks on võimalus ka süsteemidele lisada parameetreid, nagu seda hetkel on *LegacyValue.* Autor lõi kaks XML faili, ühes on kirjeldatud veebid, teises teenused. Autori arvates tagab selline lähenemine faili inimloetavuse, sest vahel on vaja failidesse lisada uusi süsteeme või sisse viia korrektuure käsitsi ning sellisel juhul on loetavus vajalik.

4.2.2 Teenuste paigaldamise skript

Teenuste paigaldamiseks lõi autor *PowerShell* skripti, mis käivitub Bamboo Windowsi agentidel. Skript vajab korrektseks tööks kolme argumenti, mida saab seadistada Bamboos paigaldusplaani ülesandes skripti argumentide real. Need kolm argumenti on välja toodud joonisel Joonis 6:

```
param(
[string]$pakk,
[string]$bambooteenus,
[string]$enviroment
)
```

Joonis 6. Skripti parameetrid

Nendest esimene viitab artefakti asukohale, see on teekond agendil asuva kaustani, kus sees asuvad paigalduseks vajalikud failid.

Teine viitab Bamboo paigaldusplaani nimele. See on sama nimekuju, mis autor nomenklatuuri välja mõeldes paika pani. Sama nimekujuga on ka teenuse paigalduskausta asukoht, teenuse „display name“ teenuste nimekirjas, teenuse binaarfaili nimi, teenuse konfiguratsioonifaili nimi ja teenuse koodihoidla nimi. See tagab selle, et alati on aru saada, millisele teenusele soovitakse viidata. Lisaks tekib nüüd võimalus kasutada paigaldusskriptis seda muutujat.

Kolmas viitab keskkonnale, kuhu paigaldust soovitakse teha. See muutuja tuleb iga süsteemi iga keskkonna kohta eraldi kirjeldada.

```
$service_start_timeout = 40
$switcher = 0

$system = $sisu.systems.system | where {$_.value -eq
$bambooteenus}

$paths = $system.Enviroments.Enviroment | where {$_.value -eq
$enviroment}

$paths = $paths.Paths.path

$legacyvalue= $system.legacyvalue.value
```

Joonis 7. Muutujate algväärtustamine

Joonisel Joonis 7. väljatoodud koodis väärtustatakse muutujad *$service_start_timeout*, mis määrab ära, mitu sekundit skript ootab peale katset käivitada teenus, enne kui ta loeb teenuse mittekäivitunuks, *$switcher*, mida kasutatakse otsustamiseks, millisesse keskkonda ja missugust paigaldust teostada ja muutujad *$system*, *$paths* ja *$legacyvalue*, kuhu loetakse väärtused XML failist, kus on kõik süsteemid kirjeldatud.

Eelnevalt väljatoodud vajaduste hulgas tõi autor välja selle, et eksisteerib vajadus juhtida paigaldusi vastavalt keskkondadele ja sellele, kas tegemist on „Legacy“ või „NonLegacy“ teenusega. Selle realiseerimiseks otsustas autor kasutusele võtta kahendsüsteemi järkude

kaaludel põhineva eraldusmeetodi. Lahtiseletatult tähendab see seda, et vastavalt teatud tunnuse esinemise puhul liidetakse muutujale *$switcher* kahendsüsteemi järkude kaale. Näiteks, kui süsteem on „NonLegacy“, siis liidetakse 0, kui aga „Legacy“ siis 1, vastavalt keskkonnale, liidetakse TEST puhul 4, PRE puhul 8 ja LIVE puhul 16. Iga tunnusega liidetav väärtus on kahe aste ning need on järjest suurenevad. Sedasi omistatakse muutujale *$switcher* lõpuks väärtus, mis määrab unikaalselt ära tunnuste komplekti, mis selle väärtusega kaasas käib. Peale muutuja *$switcher* väljaarvutamist saab siis välja kutsuda vastava funktsiooni, mis sõltub sellest, mida konkreetses keskkonnas konkreetse teenusega teha soovitakse. Joonisel Joonis 8 on välja toodud kood, mis otsustamist teostab ja millelt on näha kõik erinevad kombinatsioonid, mida muutuja *$switcher* omandada võib. Autor lisas ka veahalduseks vaikimisi tegevuse, mis lõpetab skripti veakoodiga 1, kui muutujale omistatud väärtust ei leidu sobivate kombinatsioonide hulgas. Hetkel on teostatud hinnanguliselt 500 paigaldust ja sellist olukorda, kus esineks veakood 1, pole veel ette tulnud.

```
switch ($switcher)
{
    '36' {Regulaarpaigaldus $bambooteenus} #NonLegacy, TEST, Vana
    '37' {Regulaarpaigaldus $bambooteenus} #Legacy, TEST, Vana
    '40' {Regulaarpaigaldus $bambooteenus} #NonLegacy, PRE, Vana
    '41' {Regulaarpaigaldus $bambooteenus} #Legacy, PRE, Vana
    '48' {Regulaarpaigaldus $bambooteenus} #NonLegacy, LIVE, Vana
    '49' {Regulaarpaigaldus $bambooteenus} #Legacy, LIVE, Vana
    '68' {Konversioonpaigaldus $bambooteenus} #NonLegacy, TEST, Uus
    '69' {Konversioonpaigaldus $bambooteenus} #Legacy, TEST, Uus
    '72' {Regulaarpaigaldus $bambooteenus} #NonLegacy, PRE, UUS
    '73' {Regulaarpaigaldus $bambooteenus} #Legacy, PRE, UUS
    '80' {Konversioonpaigaldus $bambooteenus} #NonLegacy, LIVE, Uus
    '81' {Konversioonpaigaldus $bambooteenus} #Legacy, LIVE, Uus
    Default {Write-Host "Vastava kombinatsiooniga teenust ei leitud,
väljun."; exit 1}
}
```

Joonis 8. Koodi osa, mis otsustab kumba funktsiooni rakendada

Funktsiooni *Regulaarpaigaldus* kasutatakse sellisel puhul, kui ei ole vaja serveril olevat teenust uuele nimekujule üle viia. Funktsioon teostab järgnevad tegevused:

- Kustutab ära paigalduspakist .config lõpuga failid.
- Loeb teenuse asukohast välja serveri, kus teenus asub ning küsib serverilt, kas selline teenus töötab serveril.

- Kui teenus töötab, peatab skript selle, kopeerib failid ja käivitab uuesti teenuse.
- Kui teenus ei tööta, kopeerib skript failid.

Failide kopeerimiseks otsustas autor kasutada Windowsi töövahendit robocopy.exe, kuna selle poolt antav tagasiside on sisukam, kui *PowerShelli* käsu Copy-Item tagasiside. Lisaks on robocopy-l rohkem võimalusi seadistamiseks. Skriptis kasutab autor robocopy.exe käivitusvõtmetega */is /e*, millest esimene tõstab üle kõik paigalduspakis olevad failid, hoolimata sellest, kas sihtkohas on need olemas või mitte ning teine käsib kopeerida kõik paigalduspaki alamkaustad ja –failid. Robocopy.exe on olemas vaikimisi igas Windowsi operatsioonisüsteemis.

Teenuste failide kopeerimiseks on vajalik, et paigaldatav teenus oleks paigalduse ajal peatatud, kuna robocopy.exe ei saa üle kirjutada faile, mis on hetkel kasutuses teiste protsesside poolt. Lisaks esineb olukordi, kus teenus on küll serverile paigaldatud, aga pole käivitatud. Skript peab tuvastama, mis seisus teenus enne paigaldust on ning vastavalt peale paigaldust teenuse käivitama või mitte. Teenuse staatuse kontrollimiseks kasutas autor käsurea tööriista SC.exe, mille üheks väljundiks on teenuse staatuse kontroll. See tööriist on vaikimisi igas Windowsi operatsioonisüsteemis olemas. Sama tööriista kasutas autor ka teenuste peatamiseks ja käivitamiseks.

Funktsiooni *Konversioonpaigaldus* kasutatakse juhul kui on vaja paigalduse käigus teenus viia üle uuele nimekujule. Funktsioon teostab järgmised tegevused:

- Kustutab ära paigalduspakist .config lõpuga failid
- Loeb teenuse asukohast välja serveri, kus teenus asub ning küsib serverilt, kas paigaldatav uue nimekujuga teenus töötab serveril.
- Kui töötab, suunab töövoo edasi *Regulaarpaigaldus* funktsiooni
- Kui ei tööta, aga teenus on serverile juba paigaldatud, suunab töövoo *Regulaarpaigaldus* funktsiooni
- Kui ei tööta ja teenust pole uue nimekujuga serveril, otsib vastavustabelist uuele nimekujule vastava vana nimekuju ning küsib serverilt, kas selline teenus on serverile paigaldatud.

- Kui on, siis peatab vana teenuse, seadistab vana teenuse enam mitte automaatselt serveri taaskäivitumisel käivituma, paigaldab uue teenuse kasutades funktsiooni *PaigaldaUusTeenus* ja käivitab uue teenuse.

- Kui ei ole vana nimekujuga teenust serveril, siis paigaldab uue teenuse kasutades funktsiooni *PaigaldaUusTeenus*.

Uuele nimekujule vana nimekuju vaste leidmiseks koostas autor abifaili New-to-Old.ps1, mille näide on joonisel Joonis 9. Oma olemuselt on see *PowerShelli* räsitabel, kus hoitakse võtme ja väärtuse paare.

```
$new_to_old = @{
        ...
        "PERH.Service.Teenus1" = "[PERH] VanaTeenus1"
        "PERH.Service.Teenus2" = "[PERH] VanaTeenus2"
        ...
}
```

Joonis 9. Näide abifailist New-to-Old.ps1

Funktsiooni *PaigaldaUusTeenus* kasutatakse juhul, kui on vaja paigaldada serverile uus teenus. Funktsioon teostab järgmised tegevused:

- Loob teenuse asukohta teenuse nimekujuga kausta

- Kopeerib teenuse failid loodud kausta

- Loob teenuse paigaldamiseks ja eemaldamiseks vajalikud .cmd failid

- Kontrollib, kas tegemist on „Legacy“ või „NonLegacy“ teenusega ning kutsub välja vastava konfiguratsioonifaili paigaldamise funktsiooni.

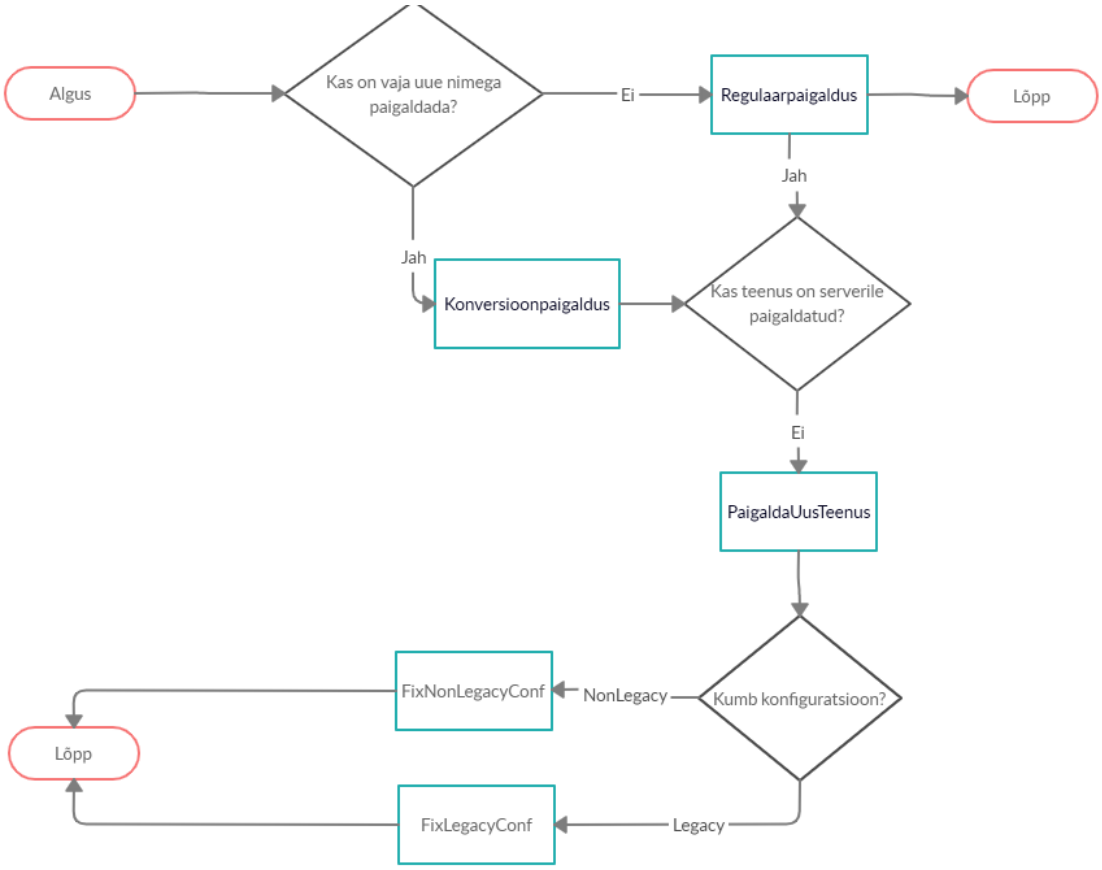
- Käivitab loodud installifaili, mis paigaldab teenuse serverile, kasutades selleks .NET raamistiku tööriista InstallUtil.exe

Funktsiooni *FixLegacyConf* kasutatakse siis, kui on vaja paigaldada „Legacy“ teenuse konfiguratsioonifail. Funktsioon teostab järgmised tegevused:

- Koostab teenuse sisemise konfiguratsioonifaili ja välise konfiguratsioonifaili viite asukoha, kasutades selleks teenuse uut nimekuju.

- Tekitab sümboolse lingi, mis viitab konfiguratsiooni asukohas olevale failile ning asub teenuse kaustas.

Funktsioonide UML diagramm on toodud joonisel Joonis 10. Funktsioonide UML diagramm



Joonis 10. Funktsioonide UML diagramm

4.2.3 Veebi paigaldamise skript

Veebide paigalduse juures pole vaja tähelepanu pöörata failide kasutusel olekule, seetõttu on veebide paigaldusskript lihtsam. Selles pole teenuste oleku kontrolli. Skript teostab järgnevad tegevused:

- Kustutab ära paigalduspakist .config lõpuga failid.
- Kopeerib veebi failid.

4.3 Tulemused

Peale skriptide ja abifailide koostamist ning Bamboo seadistamist, hakati paigaldusi keskkondadesse teostama eranditult läbi Bamboo. Paigaldus võtab tunduvalt vähem aega, kui varem. Varasemalt kulus ühe paigalduse peale olenevalt rakenduse mahust kuni 7 minutit, suurima osa sellest võttis hetkel olemasolevast rakendusest varukoopia tegemine, mida tuli käsitsi teha. Bambooga pole vaja varukoopiat teha, sest alati on võimalus koodi repositooriumist vajaliku seisu kood uuesti kokku ehitada ning see paigaldada. Bambooga võtab paigaldus aega tavaliselt 40 sekundit. Elimineeritud on ka nimekujude erisusest tingitud arusaamatused, sest nüüd on kõik teenused üle viidud korrektsele, nomenklatuuri reeglitele vastavale nimekujule. Ka uute, lisanduvate teenuste puhul hoitakse rangelt seatud reeglitest kinni. Kadunud on ka administraatorite eksimustest tingitud vead, sest nüüd teostab paigaldusi programm, mitte inimene. Vahel ebaõnnestuvad paigaldused on tingitud vigadest algandmetes, näiteks on abifaili koostamisel tehtud viga. Need vead saavad aja jooksul korrigeeritud ning lõpuks on saadud süsteem robustne ning töötab korrektselt.

Bamboo-s on ka võimalus seadistada erinevatele kasutajatele ja kasutajagruppidele õigusi erinevatesse keskkondadesse. Seda võimalust ära kasutades, töötas autor koostöös süsteemiarhitektiga välja õiguste süsteemi, mis lubab arendajatel teostada paigaldusi TESTI ning tootejuhtidel PRELIVE. Sellisel puhul saavad need paigaldused teostatud nii, et selleks ei vajata üldse administraatorite aega. Esinevate probleemide korral aga on olemas konkreetne jälg, millal mõni paigaldus teostati ning kas see võib probleemi põhjuseks olla. Süsteem on PERH-is kasutusel igapäevaselt ning alates süsteemi kasutuselevõtust on edukalt teostatud üle 500 paigalduse.

5 Edasised tegevused

Tulevikus on loodud süsteemi laiendamine lihtne. Uue teenuse või veebi lisandumisel, tuleb see kirjeldada vastavas abifailis ning Bamboos lisada uus paigaldusplaan. Uue serveri lisandumisel on võimalik abifaile skriptiga muuta ning lisandunud server kirjeldada. On võimalik, et lisaks kahele paigaldusskriptile tuleb juurde lisada teisigi, kui paigaldusprotsess erineb kasutusel olevatest. Näiteks JAVA teenused, kus paigaldusprotsessi osaks on .jar faili kopeerimine ning sellest Windowsi teenuse tegemine.

Hetkel pole veel rakendatud kogu Bamboo poolt pakutav võimekus. Näiteks saab seadistada veel selle, et artefakt paigaldatakse TESTI kohe peale selle valmimist. Hetkel peab selle paigalduse keegi käsitsi käivitama. Veel on võimalus seadistada automaattestid, mis peale paigaldust käivitatakse. Testide õnnestumise korral, paigaldatakse artefakt järgmisesse keskkonda edasi.

Lõpuni seadistamata on teavituste osa, mis annab asjassepuutuvatele isikutele märku õnnestunud koodiväljalasetest ja paigaldustest. Selle jaoks on vaja käsitsi kõik paigaldus- ja ehitusplaanid üle käia ja vajalikud seadistused sisse viia.

Implementeerimata on veel andmebaasi propagaator, mis viiks arenduse käigus tekkinud vajalikud andmebaasi muudatused automaatselt andmebaasidesse. Selleks on vaja arendajatepoolset lisa arendustööd ning administraatorite poolset seadistamist.

Loodud lahendust on võimalik kasutada ka siis, kui PERH-i süsteemide arhitektuur liigub tulevikus Dockerile. See on plaanis, aga täpne ajakava pole veel teada. Sellisel juhul on vaja koostada uued abifailid, ning skripte modifitseerida.

6 Kokkuvõte

Töö lõppeesmärgiks oli luua paigaldusskriptid ja nende juurde kuuluvad abifailid, mille abil saab teostada paigaldusi PERH-i erinevatesse keskkondadesse. Eesmärgi täitmiseks analüüsiti PERH-i keskkondasid ning seati üles tingimused, millele loodavad skriptid peavad vastama. Loodi kaks skripti, üks teenuste, teine veebide paigalduseks. Lisaks selgus skriptide kirjutamise käigus, et on vaja luua ka abifailid, kus hoitakse erinevaid tööks vajalike loendeid. Peale skriptide koostamist implementeeriti need Bamboo paigaldusraamistikku. Lisaks seadistati Bamboo-d, et skriptid tuvastaksid, mis keskkonnas need käivitati ning tegutseksid vastavalt. Peale Bamboo kasutuselevõttu on paigaldusprotsess muutunud kiiremaks ning paigaldusega seotud vigu pole enam esinenud. Lisaks on välja töötatud protsess, kuidas saavad paigaldusi teostada ka arendajad ja tootejuhid, mis aitab säästa administraatorite tööaega.

Kasutatud kirjandus

[1] M. Fowler, „Continuous Integration,“ ThoughtWorks, 01 mai 2006. [Võrgumaterjal]. Available: https://www.martinfowler.com/articles/continuousIntegration.html. [Kasutatud 20 september 2019].

[2] Atlassian, „Bamboo documentation,“ Atlassian, 17 september 2019. [Võrgumaterjal]. Available: https://confluence.atlassian.com/bamboo. [Kasutatud 20 september 2019].

[3] M. Fowler, „Microservices,“ ThoughtWorks, 25 Märts 2014. [Võrgumaterjal]. Available: https://martinfowler.com/articles/microservices.html. [Kasutatud 25 Oktoober 2019].

[4] P. Murray, „Traditional Development/Integration/Staging/Production Practice for Software Development,“ Disruptive Library Technology Jester, 4 detsember 2006. [Võrgumaterjal]. Available: https://dltj.org/article/software-development-practice/. [Kasutatud 08 oktoober 2019].

[5] R. Cutter, „Continuous Integration and Bamboo,“ 2019. [Võrgumaterjal]. Available: http://www.blendedperspectives.com/wp-content/uploads/2014/04/Bamboo-and-CI-cutterryan.pdf. [Kasutatud 20 september 2019].

Lisa 1 – Teenuste paigaldamise script

```
param(
  [string]$pakk,
  [string]$bambooteenus,
  [string]$enviroment
)


[xml]$sisu = Get-Content "E:\bamboo\PSScripts\Systems2.xml"
 . "E:\bamboo\PSScripts\new-to-old.ps1"


write-host "Pakk on: $pakk"
write-host "Teenus on: $bambooteenus"
write-host "Keskkond on: $enviroment"

$service_start_timeout = 40
$switcher = 0

$system = $sisu.systems.system | where {$_.value -eq $bambooteenus}

$paths = $system.Enviroments.Enviroment | where {$_.value -eq $enviroment}

$paths = $paths.Paths.path

$legacyvalue= $system.legacyvalue.value

if($system -eq $null)
{
    Write-Host "Teenust ei leitud XML-ist."
    exit 1
}

if($paths -eq $null)
{
    Write-Host "Teenusel puuduvad selles keskkonnas teekonnad."
    exit 1
}

switch($legacyvalue)
{
    'NonLegacy' {$switcher+=0}
    'Legacy' {$switcher +=1}
    Default {write-host "Teenuse legacyvalue on kirjeldamata"; exit 1}
```

```
}

switch ($enviroment)
{
    'TEST' {$switcher+=4}
    'PRE' {$switcher+=8}
    'LIVE' {$switcher+=16}
    Default{Write-host "Teenuse keskkonda polnud keskkondade nimekirjas.";
exit 1 }
}

if($bambooteenus.StartsWith("["))
{
    $switcher+=32
}
elseif($bambooteenus.StartsWith("PERH."))
{
    $switcher+=64
}
else
{
    Write-Host "Teenuse nimi ei vasta nõuetele, väljun."
    exit 1
}

function Regulaarpaigaldus ($teenus)
{

    $conffiles = $pakk + "\*.config"
    Write-Host "Eemaldan konfifailid: $conffiles"
    $conffiles | remove-item -force -Verbose

    foreach ($path in $paths)
    {
        Write-host "Teekond on: $path"
        Write-Host "Teenus on: $teenus"

       if(!(Test-Path $path))
       {
            Write-host "Teenuse teekonda ei leitud"
            exit 1
       }

       $temp=$path.split("\")
       $server=$temp[2]
       $vastus = sc.exe \\$server query $teenus
       $copypath= $pakk+"\*"

       if($vastus[3].contains("RUNNING"))
       {
            try
```

```
            {
                do
                {
                    $a = SC.exe \\$server STOP $teenus
                    start-sleep -Seconds 2
                    $vastus = sc.exe \\$server query $teenus
                }
                until (!$vastus[3].Contains("4"))
                Write-Host "Teenus peatatud..."
            }
            catch
            {
                Write-Host "Teenuse peatamine ebaõnnestus."
                exit 1
            }
            try
            {
                robocopy $pakk $path /is /e
                Write-Host "Teenuse failid kopeeritud..."
            }
            catch
            {
                Write-Host "Teenuse failide kopeerimine ebaõnnestus."
                exit 1
            }

            $teenusestart = Get-Date

            try
            {
                do
                {
                    $teenusestart_try = Get-Date
                    $b = SC.exe \\$server START $teenus
                    start-sleep -Seconds 2
                    $vastus = sc.exe \\$server query $teenus
                    $span=$teenusestart_try - $teenusestart

                    if($span.totalseconds -gt $service_start_timeout)
                    {
                        throw "Teenuse käivitamine peale paigaldust
ebaõnnestus."
                    }
                }
                until ($vastus[3].Contains("4"))
                Write-Host "Teenuse käivitamine õnnestus..."
            }
            catch
            {
                Write-Host "$PSItem"
                exit 1
```

```
            }
        }
        elseif($vastus[3].Contains("STOPPED"))
        {
            try
            {
                robocopy $pakk $path /is /e
                Write-Host "Teenuse failid kopeeritud..."
            }
            catch
            {
                Write-Host "Teenuse failide kopeerimine ebaõnnestus."
                exit 1
            }
            Write-host "Teenus oli enne paigaldust STOPPED, ei käivita seda."
        }
        elseif($vastus[0].Contains("1060"))
        {
            Write-host "Teenus puudub serveril, väljun."
            exit 1
        }
        else{Write-host "Teenuse staatus on ebaselge, väljun."; exit 1}
    }
    Write-Host "paigaldus õnnestus"
    exit 0
}

function Konversioonpaigaldus ($teenus)
{
     foreach ($path in $paths)
     {
         Write-host "Teekond on: $path"
         write-host "Teenus on: $teenus"

        $temp=$path.split("\")
        $server=$temp[2]
        $vastus = sc.exe \\$server query $teenus
        $copypath= $pakk+"\*"
        $installifail=""
        $uninstallifail=""
        $InstallFilePath = $path+"\install.cmd"
        $UninstallFilePath = $path+"\uninstall.cmd"

        if($vastus[3].contains("RUNNING"))
        {
            Regulaarpaigaldus $teenus

        }
        elseif($vastus[3].Contains("STOPPED"))
        {
            Regulaarpaigaldus $teenus
```

```
        }
        elseif($vastus[0].Contains("1060"))
        {
             Write-host "Teenus $teenus puudub serveril, viin läbi esmakordse
paigalduse..."
             $vanateenus= $new_to_old[$teenus]
             $vanavastus= sc.exe \\$server query $vanateenus
             if($vanavastus[3].contains("RUNNING") -or
$vanavastus[3].Contains("STOPPED") )
             {
                  PaigaldaUusTeenus
                  Write-host "Teenus paigaldatud..."
                  try
                  {
                     $a = SC.exe \\$server STOP $vanateenus
                  }
                  catch
                  {
                     Write-Host "VANA TEENUSE PEATAMINE EBAÕNNESTUS."
                  }

                  try
                  {
                     $a = SC.exe \\$server config $vanateenus start= disabled
                  }
                  catch
                  {
                     Write-Host "VANA TEENUSE DISABLEMINE EBAÕNNESTUS."
                  }

                  Write-Host "Vana teenus peatatud ja disabletud."

                 $teenusestart = Get-Date

                 try
                 {
                     do
                     {
                         $teenusestart_try = Get-Date
                         $b = SC.exe \\$server START $teenus
                         start-sleep -Seconds 2
                         $vastus = sc.exe \\$server query $teenus
                         $span=$teenusestart_try - $teenusestart

                         if($span.totalseconds -gt $service_start_timeout)
                         {
                             throw "Teenuse käivitamine peale paigaldust
ebaõnnestus."
                         }
                     }
                     until ($vastus[3].Contains("4"))
```

```
                    Write-Host "Uue Teenuse käivitamine õnnestus..."
                }
                catch
                {
                    Write-Host "$PSItem"
                }

            }
            elseif($vanavastus[0].Contains("1060"))
            {
                PaigaldaUusTeenus
            }
            else
            {
            Write-host "Staatus on: $vanavastus"
            Write-host "Teenuse staatus on ebaselge, väljun."
                    exit 1
            }
        }
    }
}

function PaigaldaUusTeenus()
{
    try
    {
    New-item -ItemType Directory -Path $path -force
    Write-Host "Teenuse kaust loodud..."
    }

    catch
    {
        Write-host "Kausta $path loomisel tekkis viga."
        exit 1
    }

    if(Test-path -Path $path)
    {
        try
        {
            robocopy $pakk $path /is /e
            Write-Host "Teenuse failid kopeeritud..."
        }

        catch
        {
            Write-Host "Teenuse failide kopeerimine esmakordsel paigaldusel
ebaõnnestus."
            exit 1
        }
```

```
        try
        {
            $installifail = new-item -ItemType File -Path $InstallFilePath -
force
            $uninstallifail = new-item -ItemType File -Path
$UninstallFilePath -force
            Write-Host "Installifail loodud..."
        }

        catch
        {
            Write-Host "Installifaili loomine ebaõnnestus."
            exit 1
        }

        if($legacyvalue -eq "Legacy")
        {
            FixLegacyConf
        }
        elseif($legacyvalue -eq "NonLegacy")
        {
            FixNonLegacyConf
        }
        else
        {
            Write-host "Ei õnnestunud konfe korda teha." ; exit 1
        }

        try
        {
            $failisisu =
"C:\WINDOWS\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe "+
"E:\WCFServices\$teenus\" + $teenus + ".exe"
            $failisisu2 =
"C:\WINDOWS\Microsoft.NET\Framework64\v4.0.30319\InstallUtil.exe "+
"E:\WCFServices\$teenus\" + $teenus + ".exe /u"
            Write-host "Installifaili sisu on: $failisisu "
            set-content -Path $installifail -Value $failisisu
            set-content -Path $uninstallifail -Value $failisisu2
            Write-Host "Installifaili sisu muudetud..."
            $InstallFilePath = "E:\WCFServices\$teenus\install.cmd"
        }

        catch
        {
            Write-Host "Installifaili sisu muutmine ebaõnnestus."
            exit 1
        }

        try
        {
            write-host "Installifaili teekond: $InstallFilePath"
```

```
            Invoke-Command -ComputerName $server -ScriptBlock {&
$using:installfilepath {$installfilepath  } -wait }
        }
        catch
        {
                Write-Host "Installifaili käivitamine ebaõnnestus."
                Exit 1
        }
    }

}

function FixNonLegacyConf
{
    Write-host "Paigaldame NonLegacy konfi."
}

function FixLegacyConf
{
    $target = "\\$server\e$\WCFServices\$teenus\$teenus.exe.config" #Kuhu
luuakse
    $source = "\\$server\d$\perh\config\$teenus\$teenus.exe.config" #Millele
vidiatakse

    if(test-path $source)
    {
        try
        {
            New-Item -ItemType SymbolicLink -Path $target -Value $source
            Write-host "Symlink asukohaga $target loodud..."
        }
        catch
        {
            Write-host "Symlinki asukohaga $target ei õnnestunud luua."
            exit 1
        }
    }
    else {Write-host "Symlinki sihtkohta $source ei eksisteeri."; exit 1}
}

Write-host $switcher

switch ($switcher)
{
    '36' {Regulaarpaigaldus $bambooteenus} #NonLegacy, TEST, Vana
    '37' {Regulaarpaigaldus $bambooteenus} #Legacy, TEST, Vana
    '40' {Regulaarpaigaldus $bambooteenus} #NonLegacy, PRE, Vana
    '41' {Regulaarpaigaldus $bambooteenus} #Legacy, PRE, Vana
    '48' {Regulaarpaigaldus $bambooteenus} #NonLegacy, LIVE, Vana
    '49' {Regulaarpaigaldus $bambooteenus} #Legacy, LIVE, Vana
    '68' {Konversioonpaigaldus $bambooteenus} #NonLegacy, TEST, Uus
```

```
    '69' {Konversioonpaigaldus $bambooteenus} #Legacy, TEST, Uus
    '72' {Regulaarpaigaldus $bambooteenus} #NonLegacy, PRE, UUS
    '73' {Regulaarpaigaldus $bambooteenus} #Legacy, PRE, UUS
    '80' {Konversioonpaigaldus $bambooteenus} #NonLegacy, LIVE, Uus
    '81' {Konversioonpaigaldus $bambooteenus} #Legacy, LIVE, Uus
    Default {Write-Host "Vastava kombinatsiooniga teenust ei leitud,
väljun."; exit 1}
}
```

Lisa 2 – Veebi paigalduse script

```
param(
  [string]$pakk,
  [string]$teenus,
  [string]$enviroment
)

[xml]$sisu = Get-Content "E:\bamboo\PSScripts\websystems.xml"


$enviromentlist = "TEST", "PRE", "LIVE"

if($enviromentlist -notcontains $enviroment)
{
    Write-host "Veebi keskkonda polnud keskkondade nimekirjas."
    exit 1
}

$system = $sisu.systems.system | where {$_.value -eq $teenus}

if($system -eq $null)
{
    Write-Host "Veebi ei leitud XML-ist."
    exit 1
}

$paths = $system.Enviroments.Enviroment | where {$_.value -eq $enviroment}

$paths = $paths.Paths.path

if($paths -eq $null)
{
    Write-Host "Veebil puuduvad selles keskkonnas teekonnad."
    exit 1
}

$conffiles = $pakk + "\*.config"
Write-Host "Eemaldan konfifailid: $conffiles"
$conffiles | remove-item -force -Verbose


foreach ($path in $paths)
{
    Write-host "Teekond on: $path"

   if(!(Test-Path $path))
   {
        Write-host "Veebi teekonda ei leitud"
```

```
        exit 1
    }

    $copypath= $pakk+"\*"

        try
         {
             robocopy $pakk $path /is /e
             Write-Host "Veebi failid kopeeritud..."
         }

         catch
         {
             Write-Host "Veebi failide kopeerimine ebaõnnestus."
             exit 1
         }

         write-host "Paigaldus õnnestus"

}
```