TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Roman Vilu 212991IAAB

Kubernetesel põhineva taristu haldusmooduli arendamine ja testimine organisatsiooni sisemiseks kasutamiseks

Bakalaureusetöö

Juhendaja: Siim Vene
Magistrikraad
Kaasjuhendaja: Jevgeni Družkov
Magistrikraad

Tallinn 2024

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: Roman Vilu

13.05.2024

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on arendada ja testida Kubernetesel põhineva taristu haldusmooduli, mis võimaldab püstitada töövalmis Kubernetese kobarat koos selle vajalike virtuaalse IT-taristu ja kobarasiseste taristuteenustega. Mooduli arendamine on võtmeülesandeks organisatsioonis toimuva Kubernetese kasutuselevõtu protsessi juures ning on kutsutud lihtsustama süsteemiadministraatori tööd.

Lõputöö analüüsi käigus kirjeldati mooduli tehnilised nõuded ja taristu arhitektuur, mille põhjal koostati tehnoloogiline mudel. Mooduli töö põhineb ‘taristu kui koodi’ kontseptsioonil, millega tagatakse taristu elutsükli halduse peamiselt deklaratiivsete konfiguratsioonide kaudu. Arendatud lahenduse näol tegemist on Terraformi mooduliga, mis rakendab automatiseeritud voogu taristu püstitamisel ja haldamisel. Moodul on kasutatav operatsioonisüsteemi tasemel ning selle käivitamine nõuab minimaalset sisendit süsteemiadministraatori poolt.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 30 leheküljel, 5 peatükki, 6 joonist, 6 tabelit.

Abstract

Development and testing of Kubernetes-based infrastructure management module for organization’s internal use

The aim of this bachelor's thesis is to develop and test a Kubernetes-based infrastructure management module, which allows setting up a ready-to-use Kubernetes cluster with the necessary virtual IT infrastructure and cluster internal infrastructure services. The development of the module is a key task of the Kubernetes adoption process in the organization and is called to simplify the work of the system administrator.

During the analysis of the thesis, the technical requirements of the module and the infrastructure architecture were described, based on which a technological model was prepared. The work of the module is based on the concept of 'infrastructure as code', which ensures infrastructure lifecycle management mainly through declarative configurations. In the form of the developed solution, it is a Terraform module that implements an automated flow in setting up and managing the infrastructure. The module is usable at the operating system level and requires minimal input from the system administrator to run.

The thesis is in Estonian and contains 30 pages of text, 5 chapters, 6 figures, 6 tables.

Lühendite ja mõistete sõnastik

AKS	Azure Kubernetes Service
API	*Application programming interface*, rakendusliides
AWS	Amazon Web Services
CNCF	Cloud Native Computing Foundation
DNS	*Domain Name System*, domeeninimede süsteem
DSL	*Domain-specific language*, domeenispetsiifiline keel
E2E	*End-to-end testing*, läbivtestimine
EKS	Elastic Kubernetes Service
FURPS	tarkvara kvaliteediatribuutide klassifitseerimise mudel
GCP	Google Cloud Platform
GPL	*General-purpose language*, üldotstarbeline keel
HCL	HashiCorp Configuration Language
HTTP	*Hypertext Transfer Protocol*, hüperteksti edastusprotokoll
HTTPS	*Hypertext Transfer Protocol Secure*, turvaline hüperteksti edastusprotokoll
IaaS	*Infrastructure as a service*, taristu kui teenus
IaC	*Infrastructure as code*, taristu kui kood
IDE	*Integrated development environment*, integreeritud arenduskeskkond
IP	*Internet Protocol*, internetiprotokoll
IT	Infotehnoloogia
LTS	*Long-term support*, pikaajaline toetus
RKE	Rancher Kubernetes Engine
SLA	*Service level agreement*, teenusetaseme leping
SSH	*Secure Shell*, turvakest
TLS	*Transport Layer Security*, transpordikihi turbeprotokoll
VRRP	*Virtual Router Redundancy Protocol*, virtuaalse ruuteri liiasuse protokoll
YAML	märgistuskeel

Sisukord

1 Sissejuhatus....10
1.1 Ülevaade probleemist....11
1.2 Metoodika....12
2 Analüüs ja arhitektuur....13
2.1 Nõuete kirjeldus....13
2.1.1 Nõuded moodulile....14
2.1.2 Nõuded arhitektuurile....16
2.2 Arhitektuur....16
2.2.1 Virtuaalne taristu....17
2.2.2 Kobarateenused....18
3 Tehnoloogiline mudel....20
3.1 Haldustööriistad....20
3.1.1 Taristuhaldus....20
3.1.2 Konfiguratsioonihaldus....22
3.1.3 Kubernetese kobarahaldus....23
3.1.4 Kobarateenuste haldus....24
3.2 Kobarateenused....25
3.2.1 Nimeserver....25
3.2.2 Veebihaldusliides....25
3.2.3 Sissevoolu kontroller....25
3.2.4 Sertifikaadihaldur....25
3.2.5 Salvestusruumihaldur....26
3.2.6 Jälgitavuse teenused....26
3.3 Virtuaalse taristu tarkvara....27
3.3.1 Serverite operatsioonisüsteem....27
3.3.2 Konteinerimootor....28
3.3.3 Koormusjaotur....29
4 Mooduli arendamine ja testimine....30

4.1 Terraformi moodul....30
4.2 Ansible mänguraamat....33
4.3 RKE kobar....34
4.4 Helmi pakendid....35
4.5 Testimine....36
5 Hinnang....37
5.1 Saavutatud tulemus....37
5.2 Aktuaalsus....37
5.3 Edasiarendus....38
6 Kokkuvõte....40
Kasutatud kirjandus....41
Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks....48
Lisa 2 – Püsilink mooduli koodibaasile....49
Lisa 3 – Testitud funktsionaalsuste nimekiri....50
Lisa 4 – Tulemuste võrdlev analüüs....54

Jooniste loetelu

Joonis 1. Uurimuse käik.....12
Joonis 2. Mooduli hallatavad süsteemi tasemed.....17
Joonis 3. Virtuaalse taristu arhitektuur.....18
Joonis 4. Kobarateenuste arhitektuur.....19
Joonis 5. Terraformi pistikprogrammi arhitektuur.....31
Joonis 6. Evituse töövoog.....32

Tabelite loetelu

Tabel 1. Mooduli konfiguratsiooni parameetrid.....14
Tabel 2. Mooduli kataloogi struktuur.....31
Tabel 3. Ansible mänguraamatu rollid.....34
Tabel 4. Kobarateenuste Helmi pakendid.....35
Tabel 5. Mooduli testitud funktsionaalsus.....50
Tabel 6. Tulemuste võrdlev analüüs.....54

1 Sissejuhatus

Aina rohkem tänapäeva infosüsteeme arendatakse Kubernetese põhjal. [1] Kubernetes on avatud lähtekoodiga konteinerite orkestreerimise platvorm, mille eesmärk on automatiseerida konteineriseeritud rakenduste paigaldust, skaleeritavust ja haldust. Konteiner kujutab endast kergekaalulist, portatiivset, järjekindlat keskkonda, mis kapseldab rakendust koos selle sõltuvustega, mis omakorda lihtsustab selle paigaldamist ja jooksutamist üle erinevate arvutuskeskkondade.

Traditsiooniliselt rakendused arendatakse monoliitse arhitektuuri järgi. [2] Monoliitse arhitektuurimudeli tulemiks on mahukas tarkvara üksus, mille komponendid ja moodulid on tihedalt sidestatud ja käivitatavad ühe protsessina. Kuigi monoliitne arhitektuur on pika aja jooksul olnud levinuimaks ja efektiivseks mustriks paljude rakenduste arenduseks, sellega kaasnevad olulised puudused, mis on seotud skaleeritavuse, hallatavuse ja uute tehnoloogiate kasutusele võtmisega.

Käesolevas lõputöös lahendatakse probleemi, mille kohaselt organisatsioon kavatseb võtta kasutusele Kubernetese mikroteenuste arhitektuuril baseeruvate projektide jaoks. Kubernetese kasutusele võtmine erinevates organisatsioonides ei toimu ühtemoodi, sh ka tehnilise poole pealt. Paljud organisatsioonid kasutavad Kubernetese keskkonna püstitamiseks arvutuspilvede natiivteenuseid, [3] vähendades selliselt halduskulusid ja suurendades süsteemi stabiilsust. Selline lähenemine aga on erinevatel põhjustel mõnede ettevõtete puhul vastuvõetamatu, nt sisepoliitika või spetsiifiliste nõuete tõttu. Lõputöös figureeriva organisatsiooni puhul on nõutud infosüsteemide käitamine organisatsiooni oma IT-taristul ja vabavaraliste tehnoloogiate kasutamine, mis välistab avalike arvutuspilvede teenuste kasutamist.

Lõputöö eesmärgiks on arendada ja testida Kubernetesel põhineva taristu taaskasutatava haldusmooduli eesmärgiga tagada organisatsioonile lahenduse funktsioneeriva Kubernetese kobara ja selle sõltuvuste püstitamiseks.

1.1 Ülevaade probleemist

Kaasaegsete tehnoloogiate arendamine toob kaasa oluliselt suure vajaduse erinevate võrgu kaudu ligipääsetavate teenuste, nn pilveteenuste, järele. Kasvava kasutamise pärast traditsioonilised monoliitsed rakendused enam ei pruugi tagada piisaval määral käideldavust, skaleeritavust ja tõrkekindlust. Igasugu planeerimata süsteemi seisak kujuneb kulukaks asjaoluks, kuna kahjustab teenuseid pakkuva organisatsiooni reputatsiooni ja mõjutab SLA-st kinnipidamist. Lisaks süsteemi rikked tekitavad olulist halduskoormust süsteemiadministraatorile, põhjustades märgatavat psühholoogilist ja intellektuaalset pinget.

Eelmainitud murede lahendamiseks oli loodud mikroteenuste kontseptsioon. [4] Mikroteenused on tarkvaraarenduse arhitektuuriline lähenemisviis, kus süsteemi komponendid kujutavad endast väikseid, iseseisvaid teenuseid, millest igaüks täidab konkreetset äriülesannet. Selline süsteemi ülesehitus võimaldab käsitleda komponente eraldi nii arendus- kui ka käitusprotsessi poolelt, sh skaleerimine, rikkeotsing, hooldus, uuendamine jms.

Väljakutseks mikroteenuste arhitektuuri puhul on teenuste majutamine. Ühe süsteemi piires võib olla suur kogus mikroteenuseid ning kasutatav majutusviis peab võimaldama neid lihtsamini hallata ja evitada. Kõige levinumaks lahenduseks on kaua olnud Linuxi konteinerid. Konteiner kujutab endast isoleeritud keskkonda, mis sisaldab rakendust ja selle sõltuvusi. Enamasti konteinerite elutsükli haldamiseks kasutatakse orkestreerimisplatvorme. Tänapäeval *de facto* standardiks konteinerite orkestreerimises on Kubernetes. [5]

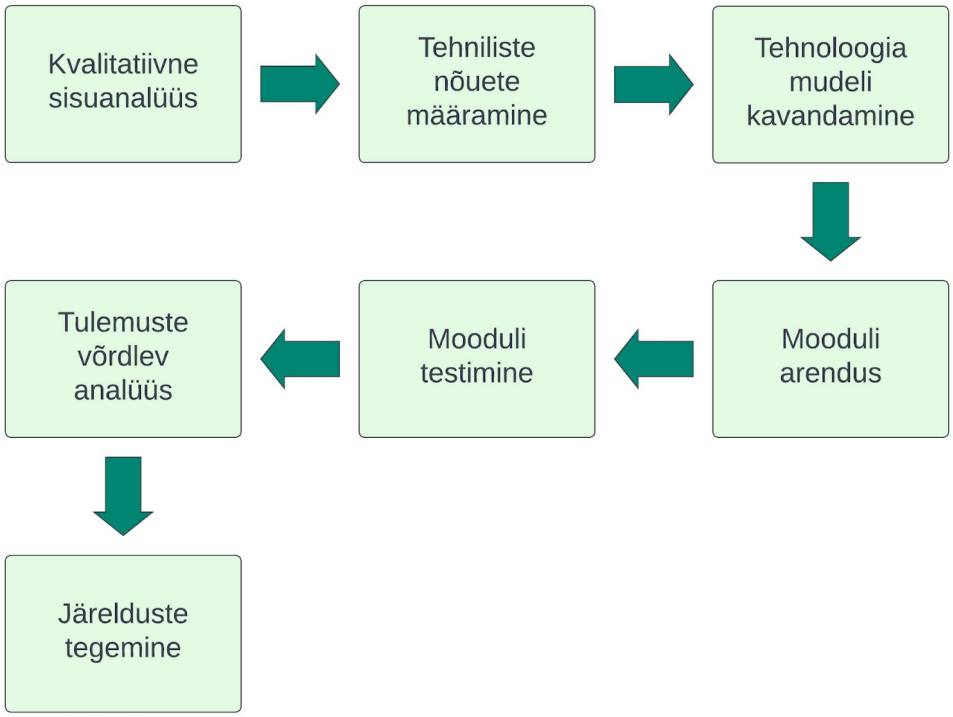
Kuigi Kubernetes on võimas platvorm, selle kasutusele võtmine ja töötava kobara (ingl *cluster*) püstitamine pole lihtne protsess ning nõuab konkreetset kava. Võimalusi eeltoodud probleemi lahendamiseks uurib autor käesolevas lõputöös. Lõpptulemuseks peab saama taaskasutatav ning laiendatav moodul, millega süsteemiadministraator oleks võimeline püstitada organisatsioonile töötava Kubernetese kobara koos selle vajaliku virtuaalse IT-taristu ja kobarasiseste taristuteenustega.

1.2 Metoodika

Käesolev lõputöö on kavandatud kvalitatiivse sisuanalüüsina, kus kasutatakse induktiivset lähenemist. [6] Kõigepealt uurimisprobleemi lahendamiseks kogutakse andmeid, mille põhjal valitakse tehnoloogiad ja tööriistad, mis kõige paremini rahuldavad püstitatud nõudeid.

Meetodi valik on põhjendatud uuritava probleemi iseloomust, mille kohaselt lihtsustatakse süsteemiadministraatori tööd, mis on ettenähtud mitte otsese ärilise kasu tekitamiseks, vaid halduskulude kokkuhoidmiseks. Uurimisteema pole uudne ning lahenduse väljatöötamise saavutab autor läbi funktsionaalsete ja mittefunktsionaalsete nõuete ning parimate praktikate määramise.

Andmete kogumist teostatakse kasutades erinevaid kirjandusallikaid, sh tehnoloogiate ja tööriistade tehniline dokumentatsioon, teadusartiklid, riist- ja tarkvara nõrkuste ja haavatavuste veebipõhised kogumid ning arvutiteaduse raamatud. Kogutud andmeid rakendatakse mooduli funktsionaalsete ja mittefunktsionaalsete nõuete määramiseks ning mooduli kavandamiseks. Uurimuse käik on esitatud joonisel 1.



Joonis 1. Uurimuse käik.

2 Analüüs ja arhitektuur

Analüüsi ja arhitektuuri kirjeldamise faasis lähtuti järgmistest organisatsiooni poolt esitatud nõuetest:

1. Moodul toetab OpenNebula hüperviisorit, kuna sellel põhineb organisatsiooni enda käitav privaatne arvutuspilv. Samuti peab arvestama võimaliku hüperviisori vahetusega, mille juures mooduli koodibaasis peaks muutuma ainult virtuaalse arvutusressursi eest vastutav komponent;
2. Mooduli kehtivuspiirkond algab hüperviisori tasemest, st virtuaalse IT-taristu ja sellest järgmised tasemed täies ulatuses langevad mooduli halduse alla, v.a rakenduse- ja andmebaasisisesed operatsioonid, ehk kasutajate ja nende õiguste haldamine, tabelite loomine jms;
3. Moodul on edasiarendatav, st on võimalik selle koodibaasi efektiivselt kasvatada ja kohandada;
4. Moodulis kasutatavad komponendid on vabavaralised organisatsiooni sisepoliitika tõttu. Proprietaarse tarkvara kasutamine moodulis on võimalik alles kriitiliste komponentide puhul, kui nendele puudub sobiv vabavaraline alternatiiv.

2.1 Nõuete kirjeldus

Mooduli nõuete määramiseks rakendatakse lõputöös raamistikku FURPS. [7] Raamistiku näol tegemist on akronüümiga, mis määrab süsteemile esitatavaid nõudeid järgmistes kategooriates:

- F – *Functionality* (funktsionaalsus) – süsteem sisaldab asjakohaseid funktsioone vastavalt kasutaja vajadustele;
- U – *Usability* (kasutatavus) – süsteem on kasutajasõbralik ja lihtne;
- R – *Reliability* (usaldusväärsus) – süsteem on stabiilne ja töökindel;

- P – *Performance* (jõudlus) – süsteem on piisavalt kiire ja reageerib kasutaja päringutele sobivalt;
- S – *Supportability* (toetatavus) – süsteem on hõlpsasti hooldatav, laiendatav ja integreeritav teiste süsteemidega.

Arhitektuuri nõuete kirjeldusel FURPS raamistiku ei rakendata. Selle asemel kirjeldatakse üldised tehnilised nõuded lähtuvalt organisatsiooni ja Kubernetese spetsiifikast tulenevatest vajadustest.

2.1.1 Nõuded moodulile

Funktsionaalsuse nõuded:

1. Moodul püstitab, kohendab ja hävitab virtuaalse IT-taristu komponente OpenNebula pilvekeskkonnas, sellele paigaldatavat Kubernetese kobarat ja kobarateenuseid.
2. Moodul haldab mitut omavahel mittekattuvat keskkonda. Keskkondade erinevused seisnevad alles nende parameetrite poolelt, mis on spetsiifilised konkreetsele keskkonnale.
3. Moodul võimaldab sisestada tabelil 1 toodud konfiguratsiooni parameetreid.

Tabel 1. Mooduli konfiguratsiooni parameetrid.

Parameeter	**Kirjeldus**
Projekt	Projekti, mille raames moodulit kasutatakse, nimi või lühinimi.
Keskkonnd	Keskkonna, mis kajastab mooduli poolt püstitava taristu eesmärki, lühinimi. Näide: *dev* (e *development*, arenduskeskkond).
Domeen	Peamine DNS-domeen, mille all püstitava keskkonna teenused tehakse kättesaadavaks. Näide: example.com.
Arvutus-ressurssid	Virtuaaltaristu arvutusvõimekuse mahud, so serverite operatiivmälu, protsessor, kettamaht.

Kasutatavuse nõuded:

1. Mooduli kasutamine on intuitiivselt selge ja lihtsasti arusaadav.
2. Moodul pakub kasutajale selget ja arusaadavat tagasisidet oma töö käigus.

3. Moodul pakub selget dokumentatsiooni ja juhiseid selle kasutamiseks ning konfigureerimiseks.
4. Mooduli käivitamine nõuab minimaalset konfiguratsiooni ja sisendit kasutaja (so süsteemiadministraatori) poolt.

Usaldusväärsuse nõuded:

1. Moodul töötab stabiilselt ja töökindlalt.
2. Mooduli konfiguratsioonid on deklaratiivsed.
3. Moodul töötab turvaliselt, kui turvanõuete rikkumine pole tingitud kasutajapoolsest mooduli kasutamisest.
4. Moodul ei rakenda muudatusi, kui süsteemi olek juba vastab kirjeldatud konfiguratsioonile.
5. Moodul haldab püstitava infosüsteemi olekufaile ja võimaldab nende salvestust nii lokaalselt, kui ka kaugelt.

Jõudluse nõuded:

1. Moodul töötab kiirelt ja tõhusalt.
2. Mooduli käivitusaeg ja rakendamise kiirus on optimaalne ooteaja poolest.
3. Moodul lihtsasti kohaneb erinevate hallatava ressursimahtudega.

Toetatavuse nõuded:

1. Moodul võimaldab versioonihaldust.
2. Mooduli struktuur on loogiline ja intuitiivselt selge.
3. Moodul on laiendatav ja muudetav.
4. Moodul on kirjutatud ja kommenteeritud inglise keeles.
5. Moodul on käivitatav Ubuntu 22.04 LTS operatsioonisüsteemil.
6. Moodulis kasutatud komponendid ja tehnoloogiad on vabavaralised.
7. Moodulis kasutatud tööriistad on tootjaneutraalsed.
8. Moodul kasutab ainult ‘taristu kui teenuse’ (ingl *infrastructure as a servic*e, IaaS) taseme funktsionaalsust virtuaalressursside püstitamiseks.

2.1.2 Nõuded arhitektuurile

Mooduli rakendatavale arhitektuurile kehtivad järgmised nõuded:

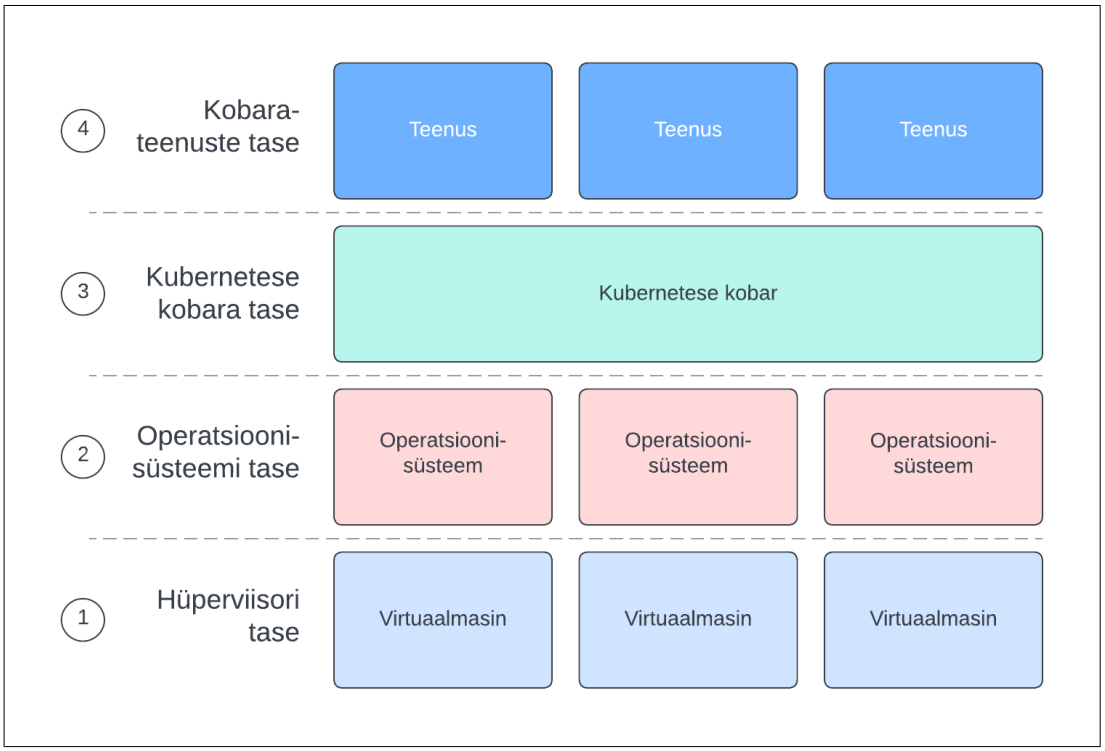
1. Virtuaalne IT-taristu on paigaldatav OpenNebula pilvekeskkonda.
2. Kubernetese kobar on kõrgelt käideldav ja skaleeritav nii juht- kui andmetasandi poolest.
3. Kuna Kubernetes kobar on hajussüsteem, millele on võimalik ligi pääseda mitme hosti kaudu, ligipääs sellele toimub läbi eraldiseisva kobaravälise koormusjaoturi. Koormusjaotur on kõrgkäideldav, et välistada nõrga lüli probleemi.
4. Ligipääs Kubernetese kobara siseteenustele ja rakendustele on turvatud protokolliga HTTPS.
5. Kubernetese kobaras on rakendatud sisese nimelahendus ja teenusetuvastus.
6. Kubernetese kobaras on rakendatud dünaamiline salvestusruumihaldus, mille kohaselt kobarasisestele rakendustele tarnitakse salvestusuumi vajadusel ilma lisakonfigureerimiseta.
7. Taristu sisaldab komponente Kubernetese kobara, virtuaalse IT-taristu ja selle komponentide jälgimiseks.

2.2 Arhitektuur

Mooduli arhitektuuri koostamisel arvestatakse vajadusega hallata neli infosüsteemi taset:

1. Hüperviisor – käsitletakse virtuaalarvutuspilve ressursse.
2. Operatsioonisüsteem – hallatakse esimesel tasemel loodud virtuaalmasinate operatsioonisüsteemi konfiguratsiooni.
3. Kubernetese kobar – hõlmab Kubernetese kobara komponentide konfiguratsiooni.
4. Kobarateenused – hallatakse kobara taristuteenuseid, mis pakuvad abifunktsionaalsust kobara haldamiseks ja jälgimiseks ning konteineriseeritud rakenduste käivitamiseks.

Tasemete jaotus on esitatud joonisel 2.



Joonis 2. Mooduli hallatavad süsteemi tasemed

2.2.1 Virtuaalne taristu

Virtuaalse taristu arhitektuuri koostamisel lähtuti järgmistest vajadustest:

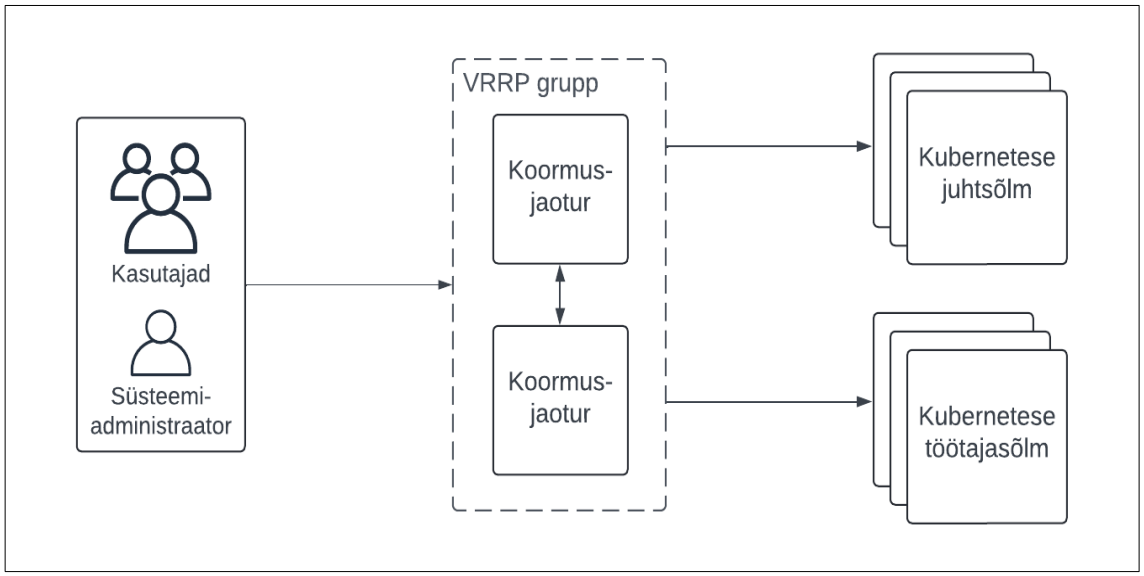
1. Süsteem peab võimaldama ligipääsu nii tavakasutajale, kui ka süsteemiadministraatorile;
2. Kubernetese kobara minimaalse kõrgkäideldavuse saavutamiseks virtuaalmasinate arv nii juhtsõlmede kui töötajasõlmede grupis peab olema 3 (vastavalt algoritmile Raft [8]).

Seoses nõuega 2 vajalikuks osutub kobara eelseisva koormusjaoturi paigaldamine. Koormusjaoturi peamine ülesanne on isiku ja Kubernetese kobara suhtluse vahendamine eesmärgiga tagada stabiilne kanal kobarale ligipääsemiseks. Koormusjaotur rakendab tagaserverite (ingl *backend servers*) tervisekontrolli mehhanisme, tänu millele garanteeritakse kasutaja päringu suunamist töökorras hostile. Koormusjaoturi kõrgkäideldavuse saavutamiseks rakendatakse vastavatel hostidel protokolli VRRP, mille kohaselt serverid töötavad aktiivses-passiivses

konfiguratsioonis ja jagavad virtuaalset IP-aadressi. [9] Nimetatud aadressi kasutatakse Kubernetese kobara ligipääsupunktina.

Kubernetese juhtsõlmed koos moodustavad juhttasandi (ingl *control plane*), mis haldab kobarat ja selle hostitud töökoormusi. [10] Töötajasõlmed omakorda moodustavad keskkonda kasutajarakenduste ja -teenuste hostimiseks ning kasulike andmete hoidmiseks. [11] Koormusjaotur on seadistatud vastavalt pordinumbrile suunama kasutaja päringut kas juhtsõlmele kobara haldusligipääsu saamiseks või töötajasõlmele rakendustele juurdepääsemiseks. [12]

Virtuaalse taristu arhitektuuridiagramm on esitatud joonisel 3.



Joonis 3. Virtuaalse taristu arhitektuur.

2.2.2 Kobarateenused

Kubernetese kobarasse evitatavad teenused on kriitilised korrektseks rakenduste töötamiseks ja avalikustamiseks.

Vaikimisi Kubernetese rakendused pole ligipääsetavad kasutajatele ja süsteemiadministraatoritele väljastpoolt kobara võrku. [13] Rakenduste avalikustamiseks kasutatakse antud projektis Kubernetese sissevoolu kontrollerit (ingl *ingress controller*) [14] , millise näol tegemist on kobarasisese veebiserveriga.

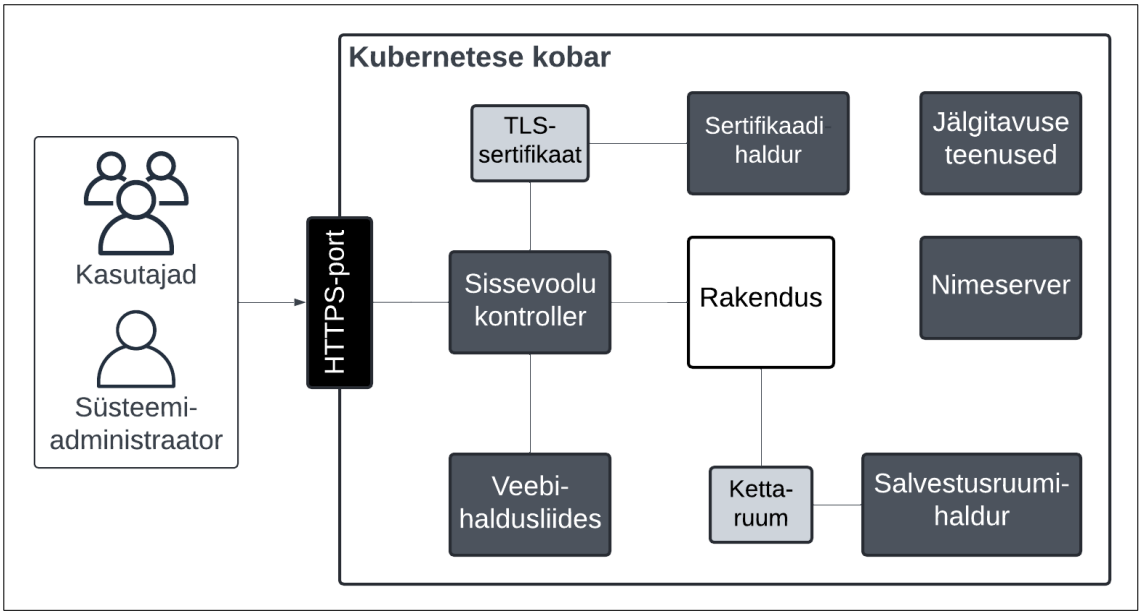
Kontrolleri ja kasutaja vahelise suhtluse turvamiseks vajalike TLS-sertifikaatide [15] väljaandmiseks kobarasse samuti paigaldatakse sertifikaaduhaldurit. [16]

Paljud Kubernetese distributsioonid vaikimisi pakuvad alles staatilise salvestusruumi tarnimise funktsionaalsust rakenduste jaoks. Kuna arendatava mooduli eesmärk on halduse lihtsustamine, arhitektuur näeb ette dünaamilise salvestusruumi tarnimise [17] lähenemist, mille kohaselt kettaruumi eraldamine rakendustele toimub kobaras automaatselt vastavalt vajadusele. Antud funktsionaalsust rakendab kobarasisene salvestusruumihaldur.

Nimeserver pakub kobarasisese nimelahenduse teenust. [18] Arvestades Kubernetese keskkonna dünaamilisusega, antud teenuse evitus osutub kohustuslikuks antud arhitektuuris.

Veebihaldusliidese ja jälgitavusevirna (ingl *observability stack*) näol tagatakse vahendeid Kubernetese kobara jälgimiseks. Kui veebihaldusliides on peamiselt ettenähtud kobara veebipõhiseks halduseks, jälgitavusevirn kujutab endast tööriistade komplekti, mille abil kogutakse kobara ja selle teenuste telemeetriaandmeid (ingl *telemetry data*). [19]

Kobarateenuste arhitektuur on toodud joonisel 4.



Joonis 4. Kobarateenuste arhitektuur.

3 Tehnoloogiline mudel

Tehnoloogilise mudeli koostamisel valitakse tehnoloogiad kolmes kategoorias:

1. Haldustööriistad – tööriistad, mis rakendavad mooduli taristu püstitamise ja haldamise töövoogu;
2. Kobarateenused – lahendused kobarateenuste funktsionaalsuse teostamiseks;
3. Taristu keskkonna tarkvara.

3.1 Haldustööriistad

Moodul koosneb mitmest komponendist, millest igaüks on kutsutud tagama konkreetse taristu taseme halduse (vt Joonis 2). Nimetatud komponentide näol tegemist on haldustööriistadega, milliseid kasutatakse koos soovitud süsteemi oleku saavutamiseks.

3.1.1 Taristuhaldus

Moodulis rakendatava taristuhalduse töö põhineb ‘taristu kui koodi’ (ingl *infrastructure as code*, IaC) [20] kontseptsioonil. Nimetatud lähenemise kohaselt taristu ressursse hallatakse programmselt, kasutades deklaratiivseid konfiguratsioonifaile, tänu millele saavutatakse parema koodi loetavuse, haldusprotsessi korduvuse ning taristu skaleeritavuse, automatiseerimise ja taastamise võimekuse. IaC tööriistad reeglina integreeruvad paljude populaarseimate pilveteenuste platvormidega, sh AWS, GCP, Azure.

Tänapäeval ‘taristu kui kood’ on üsna populaarne suund IT-taristu arendamises, seoses millega valik tööriistu on samuti lai. [21] Tihtipeale ‘taristu kui koodi’ tööriistade hulka mõistetakse konfiguratsioonihalduse tööriistu, nagu Ansible, Puppet, Progress Chef, Salt jms. Kuigi nimetatud lahendused mõneti pakuvad ‘taristu kui koodi’ funktsionaalsust, need on peamiselt ettenähtud konfiguratsioonihalduseks [22] ning nende kasutamine taristu haldamiseks pole asjakohane.

Suurtel pilveplatvormidel eksisteerivad natiivsed IaC teenused, millised haldavad konkreetselt oma pilveteenuste pakkuja ressursse, nt AWS CloudFormation, Azure Resource Manager või Google Cloud Deployment Manager. Kirjeldatud nõuete tõttu sellelaadsete lahenduste kasutamine on välistatud.

Tänapäeval tuntuimaks ‘taristu kui koodi’ tööriistaks mitteametlikult on tunnustatud [23] Terraform. [24] Terraform võimaldab kirjeldada soovitud IT-taristu oleku deklaratiivsete konfiguratsioonifailide kaudu, milleks kasutatakse domeenispetsiifilist keelt (ingl *domain-specific language*, DSL) nimega HCL (HashiCorp Configuration Language). Terraformi kood on moduleeritav, [25] tagades selliselt komponentide taaskasutatavuse ja konfigureerimise keerukuse haldamise. Konfiguratsiooni teostus toimub Terraformis planeerimise ja rakendamise (ingl *plan* ja *apply*) funktsioonide kaudu. [26] Süsteemi oleku jälgimiseks kasutab Terraform olekufaile, milliseid on võimalik salvestada nii lokaalselt kui kaugelt. [27] Terraform integreerub paljude pilveteenuste pakkujate, platvormide ja tööriistadega, sh OpenNebula ja Kubernetes.

Peamiseks alternatiiviks Terraformile tunnustatakse tänapäeval Pulumit. [28] Pulumi võimaldab kirjeldada taristu üldotstarbelistes programmeerimiskeeltes (ingl *general-purpose language*, GPL) nagu Java, Python, Go, C# jms. [29] Toetatavate keelte valik positsioneerib Pulumit kui arendajasõbralikku tööriista, kuid programmeerimise kogemus taristuarendajal on enamasti puudulik või ebapiisav, mille tõttu õppimiskeerukus osutub tema jaoks üsna kõrgeks. Samuti Pulumi on võrdlemisi uus tööriist, seoses millega ei pruugi integreeruda paljude muude lahedustega ega pakkuda piisavat kogukonna toetust.

Eelkirjeldatust lähenevalt, valib autor taristu haldustööriistaks Terraformi. Terraform on küps ‘taristu kui koodi’ tööriist, millel on suur kogukond ja lai valik integratsioone erinevate pilveplatvormide, tehnoloogiate ja tööriistadega. Samuti Terraform kasutab konfiguratsioonifailide kirjutamiseks domeenispetsiifilist keelt. Kuigi viimased nõuavad taristuarendajalt uue keele õppimist, nende õppimiskeerukus pole kõrge tänu kitsemale ülesannete fookusele. [30] Terraform on kättesaadav käivitatava kahendfailina ning sobib kasutamiseks süsteemiadministraatori tööjaamas. Lisaks omab autor eelnevat kogemust nimetatud tööriistaga.

3.1.2 Konfiguratsioonihaldus

Populaarseimateks konfiguratsioonihalduse [31] tööriistadeks osutuvad eelnevalt mainitud Ansible, [32] Puppet, [33] Progress Chef [34] ja Salt. [35] Kuigi võimekuse kohaselt nimetatud lahendused on üsna sarnased, arvesse tasub võtta järgmised aspektid:

- Agendipõhine või agendita arhitektuur – agendipõhine tööriist nõuab agenttarkvara paigaldamist sihtserverile. Agenttarkvara pärib konfiguratsioonimuudatusi keskserverilt ja rakendab neid lokaalselt. [36]

- Tõuke- või tõmbesüsteem – tõukemeetodi (ingl *push method*) puhul konfiguratsiooni muudatuste tarnimine on initsieeritud keskserveri poolt, samas kui tõmbemeetodi (ingl *pull method*) puhul muudatusi pärivad sihtserverid ise, kasutades selleks tavaliselt agenttarkvara. [37]

Chef ja Puppet kasutavad agendipõhist arhitektuuri, mis kõrgendab süsteemi algkonfigureerimise keerukust. Salt omakorda võimaldab nii agendipõhist kui ka agendivaba arhitektuuri. Ansible järgib agendita arhitektuuri.

Konfiguratsioonimuudatuste rakendamise meetodiks Chefi ja Puppeti korral on tõmbesüsteem, millega saavutatakse parema konfiguratsioonioleku järjepidevuse tänu perioodilisele muudatuste pärimisele sihtsüsteemide poolt. Ansible kasutab peamiselt tõukemeetodit, kuid on võimeline pakkuda tõmbepõhist lähenemist. [38] Salti puhul tõmbemeetod on standardne ja põhineb agendiga arhitektuuril ning tõukemeetod samuti võimaldab agendivaba lähenemist. [39] Tõukemeetodi eeliseks on reaalajas tagasiside sihtmasinate toimimise kohta ja täpsem kontroll haldusoperatsioonide üle, kuna muudatusi juhitakse viivitamatult pärast konfigureerimise käsu andmist. Tõukemeetodil konfigureerimist rakendavad Ansible ja Salt kasutades protokolli SSH.

Chefi ja Puppeti kasutamisega kaasneb samuti konfigureerimiskeele õppimiskeerukus. Kui Puppeti puhul kasutuseks on omandvaraline domeenispetsiifiline keel, mis põhineb programmeerimiskeelel Ruby, siis Chefi konfiguratsioonikeeleks on Ruby ise, seoses millega õppimiskeerukus süsteemiadministraatori jaoks osutub päris kõrgeks. [40] Ansible ja Salt toetavad konfiguratsioonide kirjutamist märgistuskeeles YAML, mille õppimiskeerukus on madal tänu inimloetavale formaadile ja lihtsale süntaksile. [41]

Vältimaks liigset konfigureerimise ja õppimise keerukust, otsustas autor kasutada konfiguratsioonihalduseks Ansiblet. Mooduli kavandamisel arvestatakse vajadusega rakendada konfiguratsioonihaldust viivitamatult peale virtuaaltaristu püstitamist ning Ansible agendivaba lähenemine ja tõukemeetod võimaldavad seda teostada hõlpsasti ilma liigse tööriista konfigureerimiseta. Võrreldes alternatiiviga, nimelt Salti näol, Ansible pakub laiemat valikut mooduleid integratsiooniks muude tehnoloogiatega. [42] Samuti Ansible on küps tööriist ja seda toetab oluliselt suurem kogukond.

3.1.3 Kubernetese kobarahaldus

Vastavalt CNCF-i majandatud sertifitseeritud Kubernetese tarkvara vastavuse (ingl *Certified Kubernetes Software Conformance*) programmile, [43] lõputöö kirjutamise hetkel eksisteerib üle 90 sertifitseeritud Kubernetese toodet. Kuna käesoleva lõputöö puhul Kubernetese kobarat paigaldatakse privaatsele pilvekeskkonnale, vaadeldavateks osutuvad paigaldustööriistade (ingl *installer*) kategooria tooted.

Enamusel lahendustest esinevad olulised takistused kasutamiseks toodangukeskkondades, sh ebapiisav küpsusaste ja orienteeritus arendusele. Ebapiisava küpsusaste näol tööriist on alles tootmise algfaasis ning selle kogukond ei pruugi tagada toetust määral, mis julgustaks selle kasutamist toodangus. Arendusorienteerituse puhul tööriist on kavandatud viisil, mis eeldab selle rakendamist arenduskeskkondades ega taga toodanguks kriitilist turvalisuse ja skaleeritavuse võimekust. Sellisteks osutuvad nt Minikube, KWOK ja kind. [44]

OpenNebula pakub enda Kubernetese distributsiooni nimega OneKE (OpenNebula Kubernetes Engine). [45] Kuigi antud lahendus on kiiresti rakendatav ja tagab parema ühilduvuse virtuaaltaristu ja Kubernetese kobara vahel, oluliseks puuduseks on selle natiivsus konkreetsele pilveplatvormile, mille kohaselt esineb tootjalukustus. [46]

Tuntuimaks lahenduseks Kubernetese kobara püstitamiseks on Kubespray. [47] Kubespray kujutab endast kogumit Ansible mänguraamatuid (ingl *playbooks*), mis pakuvad funktsionaalsust kõrgkäideldava Kubernetese kobara elutsükli haldamiseks erinevates keskkondades. Tööriist toetab enamust Linuxi distributsioone ja on komponeeritav. Kuigi antud lahendus sobib käesolevale stsenaariumile, sellel esineb puudus Ansible kui alustehnoloogia näol. Kubespray on kavandatud olema

mitmeplatvormiline lahendus ning Ansible koodibaas selle kohaselt kasvab oluliselt suureks. Lisaks Ansible järgib imperatiivset programmeerimisparadigmat, st Kubespray kasutamine ei võimalda Kubernetese kobarahaldust deklaratiivsel viisil. [48] Koos nimetatud aspektid kujutavad märgatavat raskust kobara halduse ja skaleerimise osas.

Võttes arvesse eelkirjeldatud asjaolud, Kubernetese kobara haldustööriistaks valib autor RKE (Rancher Kubernetes Engine). [49] RKE pakub Kubernetese kobara haldust deklaratiivsel ja automatiseeritud viisil. [50] RKE on neutraalne operatsioonisüsteemi ja platvormi suhtes, kuna selle komponente paigaldatakse Dockeri konteineritena. Lisaks RKE pakub natiivset integratsiooni Terraformiga läbi pistikprogrammi. [51] Tänu suurele kogukonnale, RKE on varustatud ulatusliku dokumentatsiooniga.

3.1.4 Kobarateenuste haldus

Tänu Kubernetese ressursside deklaratiivsele olemusele, kobarasse evitatavad rakendused lõpptulemusena kujunevad kogumiks YAML-vormingus kirjutatud manifeste (ingl *manifest*). [52]

Tuntud tööriistaks Kubernetese rakenduste evitamiseks on Kustomize, [53] mis kuulub ametlikku Kubernetese projektiportfelli. Kustomize võimaldab luua kihilised konfiguratsioonid, võimaldades nende komponeerimist ja seega pakkudes suuremat paindlikkust ja kontrolli konfiguratsioonide üle. Kustomize'i konfiguratsioonid on deklaratiivsed ning muudatusi rakendatakse paikadena.

Helm [54] on populaarne Kubernetese pakendihalduse tarkvara, mis kasutab mallipõhist lähenemist, kus rakenduse konfiguratsioon on määratud taaskasutatavate Kubernetese manifestimallide kaudu, milliste kogumit nimetatakse pakendiks (ingl *chart*). Helmi eeliseks on suur hulk olemasolevaid pakendeid. Puuduseks on mallide kirjutamise keerukus, mis on tingitud Go kui mallide kirjutamiskeele kasutamisest. [55]

Kuigi Helm ja Kustomize on mõlemad populaarsed lahendused Kubernetese rakenduste halduseks, Helm osutub paremaks valikuks tänu suurele arvule olemasolevaid pakendeid kolmanda osapoole rakenduste evitamiseks. [56] Lisaks, erinevalt Kustomize'ist, Helm on integreeritav Terraformiga läbi ametliku pistikprogrammi. [57]

3.2 Kobarateenused

3.2.1 Nimeserver

Kubernetese kobar on dünaamiline keskkond, milles teenused ja nende identiteet võivad uueneda üsna tihti, tekitades vajadust kobarasisese nimelahenduse järele. Kobarasisesed DNS-serverid võimaldavad domeeninimede haldust automatiseeritud viisil, kuna pärivad andmeid teenuste nimedest ja nendele vastavatest IP-aadressidest otse Kubernetese API-serverilt. Antud projektis kasutatakse Kubernetese standardset nimeserveri lahendust CoreDNS. [58] [59]

3.2.2 Veebihaldusliides

Kuigi Kubernetese kobara haldust teostatakse peamiselt käsurea kaudu, võimalus hallata kobarat graafilisel viisil on samuti hea praktika. Standardse lahendusena kasutatakse Kubernetese kobara visualiseerimiseks antud projektis Kubernetes Dashboardi. [60] [61] Tegemist on veebipõhise kasutajaliidesega, mis pakub graafilist keskkonda Kubernetese kobara haldamiseks, jälgimiseks ja tõrgete otsimiseks. Dashboardi liides on lihtne ja intuitiivselt selge.

3.2.3 Sissevoolu kontroller

Sissevoolu kontroller (Joonis 4) käsitleb sissetulevate HTTP-päringute suunamist rakendustele vastavalt HTTP-päringus sisalduvale teavele, st domeeninimele ja teekonnale. Tuntuimaks sissevoolu kontrolleriks [62] on Ingress NGINX Controller, [63] mis põhineb populaarsel veebiserveril NGINX ning mis on hallatud Kubernetese arendusmeeskonna poolt. Kuna veebiserveerimise osas ei nõuta edasijõudnud funktsionaalsust ning nimetatud lahendus on kõige populaarsem oma kategoorias, osutub Ingress NGINX Controller mõistlikuks valikuks käesoleva uurimuse raames.

3.2.4 Sertifikaadihaldur

TLS-sertifikaatide elutsükli haldamiseks projektis kasutatakse cert-manageri [64] kui standardset Kubernetese sertifikaadihalduse lahendust. cert-manager automatiseerib TLS-sertifikaatide hankimist, värskendamist ja uuendamist, selliselt vähendades süsteemiadministraatori töökoormust.

3.2.5 Salvestusruumihaldur

Kubernetes võimaldab dünaamilist salvestusruumi loomist, mille puhul rakendusele eraldatakse salvestusruumi vastavalt vajadusele. See võimaldab süsteemiadministraatoril hõlpsalt ja tõhusalt hallata kobara salvestusressursse ja vajadusel neid vabastada.

Sobivaimaks lahenduseks salsvestusruumihalduse osas osutub Longhorn. [65] Longhorn on hajusplokksalvestuse süsteem, mis pakub tõhusat ja usaldusväärset andmekaitset, sh salvestusruumi replikatsioon ja sünkroniseerimine plokksalvestusruumi tasemel.

Kuigi Longhornile eksisteerivad alternatiivid OpenEBS'i ja RookFS'i näol, [66] nimetatud lahenduste puhul ainsa eelisena esineb laiem andmesalvestuse tüüpide valik. Selle kohaselt Longhorn osutub kergekaalulisemaks platvormiks, mis ei rakenda liigset funktsionaalsust ja täidab oma peamist funktsiooni.

3.2.6 Jälgitavuse teenused

Jälgitavus (ingl *observability*) on tähtis süsteemi omadus, mis hõlmab erinevate süsteemikomponentide järjepidevat ja põhjalikku jälgimist eesmärgiga tagada nende tõrgeteta toimimist, jõudluse optimeerimist ning võimalike vigade tuvastamist ja lahendamist. Süsteemi jälgitavuse alla kuuluvad järgmised protsessid: [67]

- Seire (ingl *monitoring*) ehk mõõdikute jälgimine. Mõõdikud on kvantitatiivsed näidud, mis kirjeldavad süsteemi või rakenduse seisundit, nt protsessori- või mälukasutus, võrguliiklus, päringute arv, latentsusaeg. Mõõdikute jälgimine võimaldab süsteemiadministraatoril jälgida ressursikasutust ja jõudlust ning tuvastada võimalikke kõrvalekaldeid või probleeme süsteemis.

- Logimine (ingl *logging*). Logid sisaldavad üksikasjalikku teavet süsteemi tegevuste, sündmuste ja vigade kohta. Logide jälgimine võimaldab vaadelda süsteemi käitumist reaalajas, tuvastada probleeme ja teostada diagnostikat.

- Jälitamine (ingl *tracing*). Jäljed võimaldavad jälgida üksikute päringute teekonda, mis on eriti kasulik hajutatud infosüsteemide ja mikroteenuste

arhitektuuri korral, kui päring läbib mitmeid teenuseid. Jälitamise abil on lihtne mõista päringute voog ning teenusepõhise latentsuse ja vigade põhjused.

Nimetatud protsesside käigus genereeritud telemeetriaandmeid kasutatakse süsteemi oleku hindamiseks, vigade tuvastamiseks ning süsteemi käitumise analüüsimiseks ja prognoosimiseks. Tavaliselt telemeetriaandmeid esitatakse graafilisel kujul, et võimaldada neid paremini mõista, kasutades analüüsi- ja visualiseerimise tarkvara.

Süsteemi jälgitavuse tagamiseks antud projektis kasutatakse tööriistade komplekti, mis koosneb Grafana ökosüsteemi [68] kuuluvatest lahendustest:

- Prometheus [69] mõõdikute kogumiseks ja salvestamiseks,
- Grafana Loki koos Promtailiga [70] logide kogumiseks ja indekseerimiseks,
- Grafana Tempo [71] jälgede hoidmiseks,
- OpenTelemetry Collector [72] jälgede kogumiseks ja eksportimiseks,
- Grafana [73] telemeetriaandmete visualiseerimiseks.

Grafana virna komponendid on omavahel tihedalt integreeritavad ja võimaldavad luua terviklikku ja paindlikku lahendust süsteemi ja rakenduste jälgimiseks. Ökosüsteemi komponendid on hõlpsasti paigaldatavad ja hallatavad Kubernetese keskkondades. [74] Grafana virna kasutamist samuti toetab fakt, et sellel puuduvad samaväärsed vabavaralised alternatiivid. [75]

3.3 Virtuaalse taristu tarkvara

3.3.1 Serverite operatsioonisüsteem

Operatsioonisüsteem on kriitiline arvutisüsteemi osa, mis võimaldab tarkvara ja riistvara sujuvat koostööd ning pakub süsteemiadministraatorile liidest serveriga suhtlemiseks. Käesoleva projekti raames võetakse arvesse tavalisi operatsioonisüsteemi omadusi, sh turvalisus, jõudlus, stabiilsus, usaldusväärsus, toetus ja hallatavus. Tähtsaks aspektiks on samuti organisatsioonipoolne oskusteave, mis oluliselt mõjutab meeskondade võimet valitud platvormi haldamiseks.

Tänapäeval eksisteerib suur kogus konteinerorienteeritud operatsioonisüsteeme [76] nagu Talos Linux, Flatcar Container Linux, Fedora CoreOS, Bottlerocket, RancherOS jms. Sellised süsteemid järgivad muutumatuse (ingl *immutability*) paradigmat, [77] mille kohaselt süsteemi tuum on kirjutuskaitstud. Muutumatu Linuxi distributsiooni näol tegemist on tavapärasest turvalisema, usaldusväärsema, stabiilsema, hallatavama ja enamasti kergekaalulisema platvormiga. Kuigi sellise lahenduse kasutamine oleks asjakohane käesolevas projektis, antud süsteemid pole toetatud OpenNebula hüperviisori poolt, st nendele puuduvad vastavad kettatõmmised ja kontekstualiseerimispakendid (ingl *contextualization package*). [78] Lisaks oluliseks puuduseks on organisatsiooni meeskonna ebapiisav teadmine sellelaadsetest operatsioonisüsteemidest, mis on tingitud nimetatud kontseptsiooni suhtelisest uudsusest.

Tavaliseks valikuks serveri operatsioonisüsteemiks osutuvad üldotstarbelised Linuxi distributsioonid nagu Ubuntu Server, Debian, CentOS, RHEL, Fedora, AlmaLinux, Alpine Linux, Oracle Linux, [79] milliste eeliseks on küpsus ja tuttavlikkus süsteemiadministraatorile ja taristuinsenerile. Sobilikumaks variandiks nimetatud lahendustest osutub populaarne distributsioon Ubuntu Server, [80] mis pakub pikaajalist korporatiivset tuge ja suurt kogukonda ning mille näol tegemist on organisatsioonile tuttava lahendusega. Kuigi Ubuntu on peamiselt tuntud selle kasutajasõbralikkuse ja kasutusmugavuse eest, selle serverivariant osutub üsna usaldusväärseks ja turvaliseks alternatiiviks teistele serveridistributsioonidele. Samuti tasub võtta arvesse Ubuntu Serveri skaleerimisvõimet ja jõudlust, [81] mis esineb olulise eelisena Kubernetese töökoormuste käitamise osas.

3.3.2 Konteinerimootor

Konteinerimootori näol tegemist on platvormiga, mis hõlbustab konteinerite käivitamist ja haldamist. [82] Antud projekti raames konteinereid kasutatakse mitte ainult Kubernetese rakenduste, vaid samuti RKE komponentide käivitamiseks.

RKE komponente disaini kohaselt paigaldatakse Dockeri konteineritena. Kubernetese konteinerite jooksutamiseks arendatavas moodulis on kasutatud CRI-O. [83] CRI-O on kergekaaluline konteinerimootor, mis on kavandatud ja optimeeritud spetsiaalselt Kubernetese jaoks. CRI-O väljalasketsüklid on üksüheses vastavuses Kubernetese

väikeversioonide omadega, tänu millele on tagatud kõrge ühilduvus viimasega ja stabiilne keskkond konteinerite käivitamiseks.

3.3.3 Koormusjaotur

Virtuaalse taristu arhitektuuri kohaselt ligipääs Kubernetese kobarale peab toimuma kobaravälise koormusjaoturi kaudu. Võttes arvesse fakti, et Kubernetese kobara sissevoolu kontrolleri näol juba kasutatakse Ingress NGINX Controllerit (3.2.3), mõistlikuks lahenduseks koormusjaoturi püstitamiseks osutub NGINX. [84] Sama tootja poolt arendatud tööriistade kasutamine võimaldab kitsendada tehnoloogiavirna tehes seda lihtsamaks haldamiseks ja rakendamiseks. [85]

Kõrgkäideldavuse saavutamiseks paigaldatakse koormusjaotushostidele tarkvara Keepalived, [86] mis rakendab VRRP protokolli ja tagab jagatud virtuaalse IP-aadressi. Antud lahendus on tuttav autorile ning tänu hõlpsale konfiguratsioonile ei tekita vajadust alternatiivide otsimise järele.

4 Mooduli arendamine ja testimine

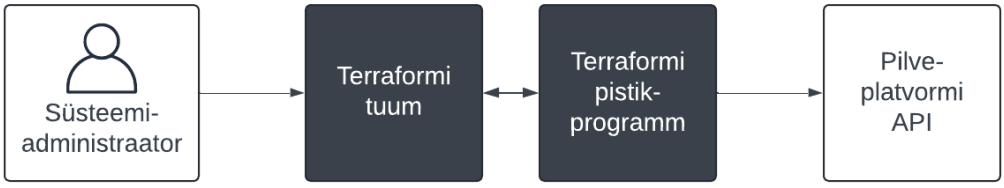
Tööjaama operatsioonisüsteemina kasutatakse Ubuntu 22.04 LTS, [87] mis on peamiselt tingitud lõputöö autori isiklikust eelistusest. Objektiivse hinnangu kohaselt on antud platvorm sobilik tänu selle populaarsusele, pikaajalisele toele, tugevale kogukonnale, stabiilsusele ja kasutajasõbralikkusele.

Mooduli koodibasi arendamiseks eelistas autor integreeritud arenduskeskkonda (ingl *integrated development environment*, IDE) IntelliJ IDEA Ultimate Edition, [88] mille litsents on tagatud organisatsiooni poolt. IntelliJ IDEA pakub laia valikut pistikprogramme, mis toetavad erinevate tehnoloogiate ja tööriistade konfiguratsiooni kirjutamist, sh käesolevas projektis kasutatavad Terraform, Ansible, Helm, Kubernetes, NGINX jpt. Valikut toetab samuti programmi kasutajasõbralikkus ja kasutusmugavus.

4.1 Terraformi moodul

Arendatava lahenduse näol tegemist on Terraformi mooduliga. [89] Terraformi moodul kujutab endast kogumit Terraformi konfiguratsioonifaile, mis sisaldavad Terraformi seadistust ja püstitatava taristu ressursside kirjeldust. Käesolev moodul on kavandatud alammoodulina (ingl *child module*), st on ettenähtud kasutamiseks teises, nn juur-, moodulis (ingl *root module*).

Terraformi tööprintsiip põhineb erinevate platvormide rakendusliidese (ingl *application programming interface*, API) kasutamisel ressursside haldamiseks. Terraform rakendab pistikprogramme erinevate platvormidega integreerumiseks. [90] Tänu laiale valikule pistikprogramme tööriist on võimeline mitte ainult luua virtuaaltaristu komponente, vaid samuti hallata lokaalseid faile, genereerida juhuväärtuseid ja krüptovõtmeid, jooksutada teiste tööriistade programme jpm. Nimetatud võimekus aitab täielikumalt automatiseerida taristu püstitamise töövoogu ja vähendada sisendit süsteemiadministraatori poolt. Terraformi pistikprogrammi arhitektuur on esitatud joonisel 4.



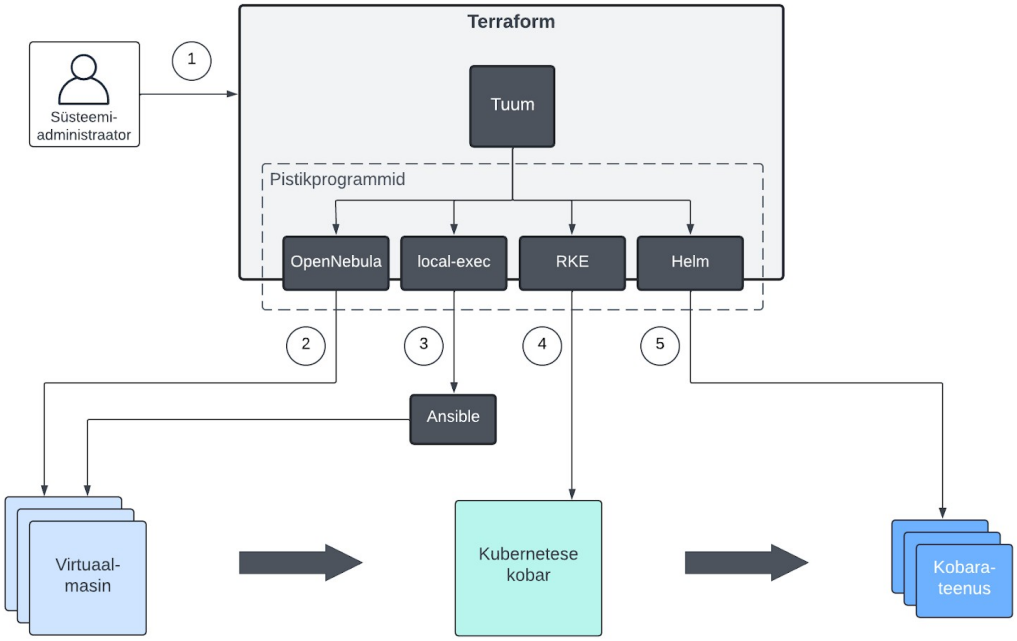
Joonis 5. Terraformi pistikprogrammi arhitektuur.

Oluliseks eeliseks Terraformi puhul on sõltumatus kataloogi struktuurist, st tööriist käsitleb mooduli juurkataloogis asuvaid konfiguratsioonifaile ühtse dokumendina. [91] Terraformi koodiplokkide eraldamine failidesse on tähtis alles koodi loetavuse ja kirjutamise mugavuse kohaselt ning ei mõjuta mooduli käitumist. Arendatav moodul oli koostatud vastavalt parimatele praktikatele. [92] Mooduli kataloogi struktuur on esitatud tabelil 2.

Tabel 2. Mooduli kataloogi struktuur.

Nimi	**Kirjeldus**
ansible/	Ansible mänguraamatu kataloog
helm/	Helmi pakendite kataloog
templates/	Terraformi poolt kasutatavate mallifailide kataloog
main.tf	Terraformi ressursside konfiguratsiooni sisaldav fail
outputs.tf	Terraformi väljundväärtuste konfiguratsiooni sisaldav fail
README.md	Mooduli dokumentatsioonifail
variables.tf	Terraformi sisendväärtuste konfiguratsiooni sisaldav fail
versions.tf	Terraformi ja selle pakkujate versiooni konfiguratsiooni sisaldav fail

Mooduli töövoog kujutab endast automatiseeritud protsessi, mis teostab taristu püstitamist vastavalt süsteemiadministraatori poolt tagatud parameetritele. Mooduli töövoog on toodud joonisel 5.



Joonis 6. Evituse töövoog.

Esitatud töövoog hõlmab järgmiseid samme:

1. Terraformi manuaalne käivitamine. Süsteemiadministraator algatab töövoogu käsurea, mille näol tegemist on Terraformi tuumaga (ingl *core*), abil. Tuum omakorda suhtleb pistikprogrammidega, milliste kaudu toimub ressursside haldamine.

2. Virtuaalmasinate loomine. Terraform rakendab OpenNebula pistikprogrammi, [93] mis omakorda teostab OpenNebula rakendusliidese kutseid virtuaalmasinate püstitamiseks pilvekeskkonnas.

3. Operatsioonisüsteemide konfigureerimine. Terraform kutsub Ansible konfiguratsioonihalduse protseduuri rakendamiseks eelmises sammus loodud

virtuaalmasinatel, kasutades pistikprogrammi local-exec. [94] Nimetatud pistikprogramm teostab käskude täitmist lokaalselt.

4. Kubernetese kobara juurutamine. Terraform püstitab Kubernetese kobara eelnevalt loodud ja konfigureeritud virtuaalmasinatel pistikprogrammi RKE abil.

5. Kobarateenuste evitamine. Terraform rakendab Helmi pistikprogrammi evitamaks teenuseid Kubernetese kobarasse.

Püsilink mooduli koodibaasile on esitatud lisas 2.

4.2 Ansible mänguraamat

Ansible on agendivaba süsteem, mis kasutab konfiguratsioonihalduse protseduuride täitmiseks turvaühenduse protokolli SSH. Protokolli seadistamise protsess nii kesk- kui sihtserveritel on automatiseeritud Terraformi poolt, seega täiendavaid tegevusi süsteemiadministraatori poolt pole nõutud.

Antud projekti raames Ansible on kutsutud täitma järgmiseid ülesandeid:

- Operatsioonisüsteemi tuuma komponentide uuendamine;
- Kõrgkäideldava koormusjaoturi paigaldamine ja konfigureerimine;
- Saalemälu (ingl *swap memory*) keelamine Kubernetese sõlmedel;
- Konteinerimootorite paigaldamine ja seadistamine;
- Loogiliste köidete (ingl *logical volumes*) haldamine.

Ansible koodibaas on kavandatud ühtse mänguraamatuna, milles vastavalt gruppikuuluvusele rakendatakse rolle (ingl *roles*) igale serverile. Rollid kujutavad endast taaskasutatavaid komplekte tegumitest (ingl *tasks*), mis rakendavad serveritel konkreetset konfiguratsiooniloogikat. Moodulis kasutatud rollide ja nendele vastavate gruppide nimekiri on toodud tabelil 3.

Tabel 3. Ansible mänguraamatu rollid.

Nimi	**Kirjeldus**	**Grupp**
crio	CRI-O paigaldamine ja seadistamine	Kubernetese sõlmed
docker	Dockeri paigaldamine ja seadistamine	Kubernetese sõlmed
init	Hostide algne uuendamine ja seadistamine	Kõik
keepalived	Keepalivedi paigaldamine ja seadistamine	Koormusjaoturid
lvm	Longhorni andmekataloogi loogiliste köidete seadistamine	Kubernetese töötajasõlmed
nginx	NGINX'i paigaldamine ja seadistamine	Koormusjaoturid
post	Hostide lõppseadistamine	Kõik
swap	Saalemälu seadistamine	Kubernetese sõlmed

4.3 RKE kobar

RKE kujutab endast Kubernetese distributsiooni, mis on terviklikult paigaldatav Dockeri konteineritena. Kubernetese kobara püstitamiseks, st komponentide paigaldamiseks ja konfigureerimiseks tulevastel sõlmedel, RKE kasutab protokolli SSH.

Teostuse kohaselt tegemist on Terraformi ressurssiga, [95] mille argumendid moodustavad RKE konfiguratsiooni, sh Kubernetese versioon, kobara nimi, komponentide suvandid, sõlmede seadistus jms. Sõlmede konfiguratsiooni kohaselt on tarvis määrata SSH-ühenduse parameetreid. Kobara konfiguratsiooni on võimalik Terraformi puhul koostada kas eraldi YAML-vormingus konfiguratsiooni, ressursi parameetrite või nimetatud meetodite kombinatsiooniga. Moodulis on kasutatud teine meetod, kuna see on natiivsem Terraformile ning võimaldab saavutada ulatuslikumat kontrolli üle kobarakonfiguratsiooni.

RKE kobara püstitamisele järgneb ligipääsufaili (nn kubeconfig) [96] genereerimine, mis sisaldab andmeid kobarale ligipääsemiseks, sh kobara otspunkt ja administraatorkasutaja TLS-sertifikaadid. Loodud faili kasutatakse edaspidi kobara administreerimiseks.

4.4 Helmi pakendid

Kobarateenuseid paigaldatakse kasutades kolmanda osapoole Helmi pakendeid. Helmi pakendiga evitatava rakenduse konfiguratsiooni on võimalik kohandada kasutades väärtusi (ingl *values*), mis sisalduvad väärtustefailis (ingl *values file*). [97] Reeglina kolmanda osapoole pakendid pakuvad kõrget kohandamise taset seoses orienteeritusega suurele kasutajakonnale. Evitatud Helmi pakendi olemit nimetatakse väljalaskeks (ingl *release*). Terraformi koodi kohaselt kirjeldatakse igat väljalaset deklaratiivselt ühe ressursiga.

Kuna mõned kobarateenused sõltuvad teiste poolt pakutavast funktsionaalsusest, ressursside evitus toimub konkreetses järjekorras. Nimelt, Kubernetes Dashboard, Longhorn ja Grafana Stack on varustatud veebipõhise kasutajaliidesega ning veebi serveerimise ja sertifikaadihalduse osas sõltuvad Ingress NGINX Controllerist ja cert-managerist. Grafana Stack on jälgitavuse tööriistade kogum, mis hõlmab ajalooliste telemeetriaandmete püsivsalvestust teostavaid teenuseid, so Prometheus, Grafana Loki ja Tempo, milliste jaoks on nõutud salvestusruumi tarnimine Longhorni poolt.

Nimekiri paigaldatavatest Helmi pakenditest ja nende vahelistest sõltuvustest on toodud tabelil 4.

Tabel 4. Kobarateenuste Helmi pakendid.

Nimi	**Kirjeldus**	**Sõltuvused**
coredns	CoreDNS, nimeserver	
cert-manager	cert-manager, sertifikaadihaldur	
ingress-nginx	NGINX Ingress Controller, veebiserver	
kubernetes-dashboard	Kubernetes Dashboard, veebipõhine kasutajaliides	cert-manager, ingress-nginx
longhorn	Longhorn, salvestusruumihaldur	cert-manager, ingress-nginx
grafana-stack	Grafana Stack, jälgitavuse tööriistade komplekt	cert-manager, ingress-nginx, longhorn

4.5 Testimine

Arendatava lahenduse testimine on kavandatud manuaalse läbivtestimisena (ingl *end-to-end testing*, E2E). [98] Nimetatud lähenemine eeldab testimist kujul, kus simuleeritakse tegeliku kasutaja kogemust eesmärgiga valideerida terve süsteemi funktsionaalsust ning tuvastada võimalikke vigu süsteemi töös. Lisaks valitud testimise meetod võimaldab kasutaja seisukohalt hinnata tehnoloogiavirna integratsiooni edukust.

Terraformi mooduli läbivtestimine on oluline praktika taristu muudatuste töökindluse ja ootuspärase toimimise tagamiseks, kuna sellega saavutatakse süsteemi üldise stabiilsuse ja käideldavuse. Terraformi mooduli läbivtestimine võimaldab:

- Valideerida kogu taristu komplekti alates ressursside loomisest kuni nende hävitamiseni, kindlustades selliselt Terraformi koodi ootuspärast töötamist;
- Suurendab usaldust mooduli poolt rakendatavate muudatuste vastu, tagades, et nad ei põhjusta ootamatut süsteemi käitumist ega riku olemasolevat funktsionaalsust;
- Tuvastada süsteemi vigu ja defekte varasemalt, vähendades nende parandamisega kaasnevad tulevased kulud.

Testimise protsess oli teostatud järgmiste sammude näol:

1. Nõuete, arhitektuuri ja tehnoloogilise mudeli ülevaatamine testimismallide koostamiseks;
2. Testimismallide koostamine mooduli töö hindamiseks ja taristu valideerimiseks;
3. Testimiskeskkonna ülesseadmine;
4. Testide läbiviimine ja tulemuste kirjapanemine;
5. Tulemuste analüüs.

Mooduli testitud funktsionaalsuse nimekiri on esitatub lisas 3 tabelil 5.

5 Hinnang

5.1 Saavutatud tulemus

Arendatud lahendust saab hinnata realiseeritud funktsionaalsuse ja saavutatud kvaliteedi põhjal. Terraformi mooduli kui lõpliku lahenduse näol saavutati organisatsiooni poolt püstitatud ülesannet, mis hõlmab töövalmis Kubernetese kobara ja selle sõltuvuste püstitamist. Kuigi käesoleva uurimuse raames ei teostatud vastuvõtutestimist (ingl *user acceptance testing*), arendatud lahenduse näol tegemist on hõlpsasti kasutatava tööriista mooduliga, mille algsele konfigureerimisele ja edaspidisele kasutamisele kuuluv aeg ja töökoormus on minimaalne.

Lisas 4 tabelil 6 on esitatud saavutatud tulemuste ja püstitatud nõuete võrdlev analüüs.

5.2 Aktuaalsus

Väljatöötatud moodulis on kasutatud kaasaegne lähenemine IT-taristu haldamise juures. Nimelt, arendatud moodul rakendab automatiseeritud töövoogu, kasutades selleks deklaratiivset haldusmeetodid, mille kohaselt süsteemiadministraatorilt abstraheeritakse keerulist konfiguratsiooniloogikat. Antud asjaolu on eriti oluline kontekstis, kus süsteemiadministraator pole tuttav moodulis kasutatud tööriistadega, kuna see hõlbustab kasutajakogemust ja aitab paremini mõista süsteemi töövoogu. Moodul teostab taristu püstitamist ühe täitmisega.

Mooduli arendamisel alustööriistana kasutati Terraformi, mis on tunnustatud mitteametliku standardina taristu haldamisel. Terraformi konfiguratsioonikeele inimloetavus ja sõltumatus kataloogi struktuurist võimaldab koostada keerulist koodibaasi, säilitades selle juures koodi hallatavust ja skaleeritavust. Vajadusel Terraformi on võimalik integreerida teiste organisatsiooni poolt kasutatud tööriistade ja teenustega.

Oluliseks osaks käesolevas projektis on Kubernetes kui võtmetehnoloogia. Kubernetes on populaarseim konteinerite orkestreerimise platvorm, mida kasutavad tuhanded ettevõted üle maailma. Iseenesest Kubernetese kasutuselevõtmine kujuneb raskeks protsessiks, mis on seotud lahenduste valimise ja konfigureerimisega. Olukorda samuti süvendavad organisatsiooni esitatud nõuded, milliste kohaselt Kubernetese kobara juurutamine peab toimuma privaatses pilvekeskkonnas. RKE kasutamine lahendab seda probleemi, võimaldades rakendada sama konfiguratsiooni sõltumata operatsioonisüsteemist või pilvekeskkonnast, mis soodustab tulevikus teise hüperviisori kasutuselevõtmist.

Mooduli rakendatud tehnoloogiavirna kuuluvad alles populaarsed vabavaralised lahendused, mis on toetatud tugevate kogukondade poolt ning kindlasti ei kaota aktuaalsust lähimatel aastatel. Mooduli rakendamine ja kasutamine ei too kaasa lisakulusid, mille kohaselt vastab organisatsiooni poliitikale ja käesoleva projekti nõuetele.

5.3 Edasiarendus

Mooduli näol tegemist pole lõpliku lahendusega, vaid prototüübiga, mida oleks võimalik kiirelt rakendada organisatsiooni projektides. Ajapiirangute tõttu käesoleva lõputöö raames jõuti arendada alles baaslahendust, arvestades selle juures vajadusega edaspidi seda täiendada funktsionaalsuse laiendamiseks ja usaldusväärsuse tugevdamiseks. Mooduli edasiarendamiseks näeb autor järgmiseid võimalusi:

- Testimise raamistiku väljatöötamine. Esiteks, Terraformi mooduli kooditasemelise testimise ja valideerimise rakendamine tagamaks mooduli siseste väärtuste korrapärasust. Teiseks, läbivtestimise protsessi automatiseerimine eesmärgiga vähendada testimise ajakulu ja suurendada tulemuste järjekindlust. Lõpuks, vastuvõtutestimise läbiviimine, mille tulemusena selguksid mooduli kvaliteedi kitsaskohad ning kasutajapoolsed soovid mooduli funktsionaalsuse täiendamiseks.

- Deklaratiivse konfiguratsioonihalduse kasutuselevõtmine. Kuigi Ansible on enamasti stabiilne ja usaldusväärne tööriist, selle näol tegemist on imperatiivse haldusprotsessiga. Kaasaegsemad tööriistad, nagu Nix, [99] võimaldavad

teostada konfiguratsioonihaldust deklaratiivselt ja atoomiliselt, mispuhul on tagatud vigaste muudatuste tagasipööre.

- Kubernetese kobara täiendamine haldust lihtsustavate vahenditega. Kubernetese kobara näol on esitatud terve kogum protsesse, mida on tarvis käsitleda kobara ja selle taristu korrektse töö tagamiseks. Näiteks, võtta kasutusele Kured (Kubernetes Reboot Daemon) [100] Kubernetese sõlmede järkjärgulise rebuutimise täitmiseks, Eraser [101] kasutamata või aegunud konteineritõmmiste kustutamiseks, Descheduler [102] kobara töökoormuste ühtlaseks hajutamiseks sõlmede vahel jms.

- Taristu turvalisuse tugevdamine. Turvalisus on kriitiline süsteemi omadus, mis nõuab turvaanalüüsi ja sellele järgneva suuremahulise tehnilise töö elluviimist, mis osutub liiga nõudlikuks kvalifikatsiooni ja ajakulu suhtes ning mille teostamine ei kuulu töö käsitlusalasse. Üheks vaadeldavaks probleemiks nt moodustub kobarateenuste vahelise turvaühenduse puudumine, mille kohaselt ohusobjektidele avaneb võimalus kobarasiseseks nuuskimiseks.

6 Kokkuvõte

Lõputöö eesmärk oli arendada ja testida haldusmoodulit, mis tagab organisatsioonile viisi töövalmis Kubernetese kobara ja selle sõltuvuste püstitamiseks. Püstitatud ülesande täitmiseks kirjutati ‘taristu kui koodi’ kontseptsiooni järgiva mooduli, mis püstitab ja seadistab vajalikud virtuaaltaristu ressursid OpenNebula pilvekeskkonnas ning paigaldab nimetatud virtuaalkeskkonda Kubernetese kobara koos kobarasiseste taristu abiteenustega.

Funktsionaalsete ja mittefunktsionaalsete nõuete kirjeldamiseks kasutati lõputöös FURPS raamistikku. Määratud nõuete alusel koostati arhitektuur ja kavandati tehnoloogiline mudel, mille kohaselt olid valitud haldustööriistad ja taristu komponentide tarkvara. Peamiseks haldustööriistaks osutus Terraform. Terraform on ‘taristu kui koodi’ tööriist, mis võimaldab kirjeldada taristu ressursse deklaratiivsete konfiguratsioonidega ning mille põhiselt projekti arendatud lahendus oli kavandatud Terraformi moodulina, mis rakendab automatiseeritud püstitamise ja haldamise töövoogu, st virtuaaltaristu loomine, serverite operatsioonisüsteemi konfigureerimine, Kubernetese kobara juurutamine ning kobarasiseste taristuteenuste evitamine.

Valminud moodul kujunes taaskasutatavaks ühikuks, mida on võimalik vajadusel rakendada organisatsiooni erinevates projektides töövalmis Kubernetese kobara juurutamiseks minimaalse koormusega. Kuna arendatud lahendus saavutab ülesseatud eesmärki ning rahuldab püstitatud nõudeid, projekti võib arvata edukalt täidetuks.

Kasutatud kirjandus

[1] Diamanti, “2024 Kubernetes Adoption Survey,” 2024 [Võrgumaterjal]. Kättesaadav: https://diamanti.com/2024-kubernetes-adoption-survey/. [Kasutatud 12.03.2024]

[2] Wikipedia, “Monolithic application,” [Võrgumaterjal]. Kättesaadav: https://en.wikipedia.org/wiki/Monolithic_application. [Kasutatud 12.03.2024]

[3] Akamai, “What Is Managed Kubernetes?” [Võrgumaterjal]. Kättesaadav: https://www.akamai.com/glossary/what-is-managed-kubernetes. [Kasutatud 12.03.2024]

[4] Wikipedia, “Microservices,” [Võrgumaterjal]. Kättesaadav: https://en.wikipedia.org/wiki/Microservices. [Kasutatud 12.03.2024]

[5] Kong Inc., “Kubernetes: The De Facto Standard of Infrastructure: How Kubernetes Is Modernizing the Microservices Architecture,” 2023 [Võrgumaterjal]. Kättesaadav: https://assets.prd.mktg.konghq.com/files/2023/12/6577a23c-kubernetes-the-de-facto-standard-of-infrastructure.pdf. [Kasutatud: 12.03.2024]

[6] Kalmus, V. (2015). Kvalitatiivne sisuanalüüs. V. Kalmus, A. Masso ja M. Linno (toim), *Sotsiaalse analüüsi meetodite ja metodoloogia õpibaas*. https://samm.ut.ee/kvalitatiivne-sisuanalyys/

[7] Wikipedia, “FURPS,” [Võrgumaterjal]. Kättesaadav: https://en.wikipedia.org/wiki/FURPS. [Kasutatud 15.03.2024]

[8] KubeMQ Docs, “Clustering and HA,” [Võrgumaterjal]. Kättesaadav: https://docs.kubemq.io/learn/cluster-scale#cluster-size. [Kasutatud 18.03.2024]

[9] Wikipedia, “Virtual Router Redundancy Protocol,” [Võrgumaterjal]. Kättesaadav: https://en.wikipedia.org/wiki/Virtual_Router_Redundancy_Protocol. [Kasutatud 18.03.2024]

[10] ARMO Ltd, “Kubernetes Control Plane,” [Võrgumaterjal]. Kättesaadav: https://www.armosec.io/glossary/kubernetes-control-plane/. [Kasutatud 18.03.2024]

[11] The Kubernetes Authors, “Nodes,” Kubernetes Documentation, [Võrgumaterjal]. Kättesaadav: https://kubernetes.io/docs/concepts/architecture/nodes/. [Kasutatud 18.03.2024]

[12] GeeksforGeeks, “What is a TCP load balancer?” [Võrgumaterjal]. Kättesaadav: https://www.geeksforgeeks.org/what-is-a-tcp-load-balancer/. [Kasutatud 18.03.2024]

[13] Scaleway, “Exposing Kubernetes services to the internet,” [Võrgumaterjal]. Kättesaadav: https://www.scaleway.com/en/docs/containers/kubernetes/reference-content/exposing-services/. [Kasutatud 18.03.2024]

[14] F5, Inc., “What Is an Ingress Controller?” [Võrgumaterjal]. Kättesaadav: https://www.nginx.com/resources/glossary/kubernetes-ingress-controller/. [Kasutatud 18.03.2024]

[15] Cloudflare, “What is Transport Layer Security (TLS)?” [Võrgumaterjal]. Kättesaadav: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/. [Kasutatud 18.03.2024]

[16] J. Schmidt, “Automating Certificate Management in a Kubernetes Environment,” [Võrgumaterjal]. Kättesaadav: https://www.nginx.com/blog/automating-certificate-management-in-a-kubernetes-environment/. [Kasutatud 18.03.2024]

[17] The Kubernetes Authors, “Dynamic Volume Provisioning,” Kubernetes Documentation, [Võrgumaterjal]. Kättesaadav: https://kubernetes.io/docs/concepts/storage/dynamic-provisioning/. [Kasutatud 18.03.2024]

[18] Extio Technology, “Understanding Kubernetes DNS: A Key Component for Seamless Service Discovery,” [Võrgumaterjal]. Kättesaadav: https://medium.com/@extio/understanding-kubernetes-dns-a-key-component-for-seamless-service-discovery-f0688f140025. [Kasutatud 18.03.2024]

[19] E. Bennett, “What Is Telemetry Data?” [Võrgumaterjal]. Kättesaadav: https://logit.io/blog/post/what-is-telemetry-data/#telemetry-data-defined. [Kasutatud 18.03.2024]

[20] Amazon, “What is Infrastructure as Code?” [Võrgumaterjal]. Kättesaadav: https://aws.amazon.com/what-is/iac/. [Kasutatud 18.03.2024]

[21] F. Pialoux, “Best Infrastructure as Code Tools (IaC): The Top 15 for 2024,” [Võrgumaterjal]. Kättesaadav: https://bluelight.co/blog/best-infrastructure-as-code-tools. [Kasutatud 18.03.2024]

[22] HashiCorp, “Terraform vs. Chef, Puppet, etc.,” [Võrgumaterjal]. Kättesaadav: https://developer.hashicorp.com/terraform/intro/vs/chef-puppet. [Kasutatud 20.03.2024]

[23] A. Critelly, “Components of a Modern Terraform Stack,” [Võrgumaterjal]. Kättesaadav: https://itnext.io/components-of-a-modern-terraform-stack-19f8797e8c4c. [Kasutatud 20.03.2024]

[24] HashiCorp (2024). *Terraform* [Tarkvara]. GitHub. https://github.com/hashicorp/terraform

[25] HashiCorp, “Modules Overview,” Terraform Documentation, [Võrgumaterjal]. Kättesaadav: https://developer.hashicorp.com/terraform/language/v1.5.x/modules. [Kasutatud 25.03.2024]

[26] HashiCorp, “How does Terraform work?” Terraform Documentation, [Võrgumaterjal]. Kättesaadav: https://developer.hashicorp.com/terraform/intro/v1.5.x#how-does-terraform-work. [Kasutatud: 20.03.2024]

[27] HashiCorp, “Backend Configuration,” Terraform Documentation, [Võrgumaterjal]. Kättesaadav: https://developer.hashicorp.com/terraform/language/v1.5.x/settings/backends/configuration#available-backends. [Kasutatud: 20.03.2024]

[28] Pulumi (2024). *Pulumi* [Tarkvara]. GitHub. https://github.com/pulumi/pulumi

[29] Pulumi, “Pulumi languages & SDKs,” [Võrgumaterjal]. Kättesaadav: https://www.pulumi.com/docs/languages-sdks/. [Kasutatud 23.03.2024]

[30] JetBrains s.r.o., “Domain-Specific Languages,” [Võrgumaterjal]. Kättesaadav: https://www.jetbrains.com/mps/concepts/domain-specific-languages/. [Kasutatud 24.03.2024]

[31] Red Hat, “What is configuration management?” [Võrgumaterjal]. Kättesaadav: https://www.redhat.com/en/topics/automation/what-is-configuration-management. [Kasutatud 15.03.2024]

[32] Ansible (2024). *Ansible* [Tarkvara]. GitHub. https://github.com/ansible/ansible

[33] Puppet (2024). *Puppet* [Tarkvara]. GitHub. https://github.com/puppetlabs/puppet

[34] Progress Chef (2024). *Chef* [Tarkvara]. GitHub. https://github.com/chef/chef

[35] Salt Project (2024). *Salt* [Tarkvara]. GitHub. https://github.com/saltstack/salt

[36] G. S. Ajith, “Beginner Fundamentals: Push & Pull Configuration Management Tools,” [Võrgumaterjal]. Kättesaadav: https://gayatrisajith.medium.com/beginner-fundamentals-push-pull-configuration-management-tools-85eff1b41447. [Kasutatud 25.03.2024]

[37] A. Fashakin, “Agent vs Agentless Configuration Management,” [Võrgumaterjal]. Kättesaadav: https://attuneops.io/agent-vs-agentless-configuration-management/. [Kasutatud 25.03.2024]

[38] Ansible, “ansible-pull,” Ansible Documentation, [Võrgumaterjal]. Kättesaadav: https://docs.ansible.com/ansible/2.9/cli/ansible-pull.html. [Kasutatud 25.03.2024]

[39] VMware, “Salt SSH,” Salt Documentation, [Võrgumaterjal]. Kättesaadav: https://docs.saltproject.io/en/latest/topics/ssh/index.html#salt-ssh. [Kasutatud 25.03.2024]

[40] N. Hallett, “Chef vs. Puppet vs. Ansible vs. SaltStack – configuration management tools compared,” [Võrgumaterjal]. Kättesaadav: https://www.justaftermidnight247.com/insights/chef-vs-puppet-vs-ansible-vs-saltstack-configuration-management-tools-compared/. [Kasutatud 25.03.2024]

[41] Red Hat, “What is YAML?” [Võrgumaterjal]. Kättesaadav: https://www.redhat.com/en/topics/automation/what-is-yaml. [Kasutatud 25.03.2024]

[42] phoenixNAP, “SaltStack vs. Ansible: Comprehensive Comparison,” [Võrgumaterjal]. Kättesaadav: https://phoenixnap.com/kb/saltstack-vs-ansible#ftoc-heading-7. [Kasutatud 25.03.2024]

[43] Cloud Native Computing Foundation, “Certified Kubernetes Software Conformance,” [Võrgumaterjal]. Kättesaadav: https://www.cncf.io/training/certification/software-conformance/. [Kasutatud 27.03.2024]

[44] S. Bostandoust, “Kubernetes Installation Methods: The Complete Guide Update 2022,” [Võrgumaterjal]. Kättesaadav: https://kubedemy.io/kubernetes-installation-methods-the-complete-guide-update-2022. [Kasutatud 27.03.2024]

[45] OpenNebula, “OpenNebula Kubernetes Engine (OneKE),” OpenNebula Documentation [Võrgumaterjal]. Kättesaadav: https://docs.opennebula.io/6.8/marketplace/appliances/oneke.html. [Kasutatud 27.03.2024]

[46] Cloudflare, “What does ‘vendor lock-in’ mean?” [Võrgumaterjal]. Kättesaadav: https://www.cloudflare.com/learning/cloud/what-is-vendor-lock-in/. [Kasutatud 27.03.2024]

[47] Kubespray, “Deploy a Production Ready Kubernetes Cluster,” [Võrgumaterjal]. Kättesaadav: https://kubespray.io/#/. [Kasutatud 27.03.2024]

[48] B. Ruiz, “Declarative vs imperative,” [Võrgumaterjal]. Kättesaadav: https://dev.to/ruizb/declarative-vs-imperative-4a7l. [Kasutatud 27.03.2024]

[49] SUSE Rancher (2024). *Rancher Kubernetes Engine (RKE)* [Tarkvara]. GitHub. https://github.com/rancher/rke

[50] SUSE Rancher, “Overview of RKE,” RKE Documentation, [Võrgumaterjal]. Kättesaadav: https://rke.docs.rancher.com/. [Kasutatud 27.03.2024]

[51] SUSE Rancher (2024). *Terraform Provider for RKE* [Tarkvara]. GitHub. https://github.com/rancher/terraform-provider-rke

[52] The Kubernetes Authors, “Objects In Kubernetes,” Kubernetes Documentation, [Võrgumaterjal]. Kättesaadav: https://kubernetes.io/docs/concepts/overview/working-with-objects/. [Kasutatud 30.03.2024]

[53] Kubernetes SIGs (2024). *Kustomize* [Tarkvara]. GitHub. https://github.com/kubernetes-sigs/kustomize

[54] The Helm Project (2024). *Helm* [Tarkvara]. GitHub. https://github.com/helm/helm

[55] D. Odazie, “Kustomize vs. Helm – How to Use & Comparison,” [Võrgumaterjal]. Kättesaadav: https://spacelift.io/blog/kustomize-vs-helm#comparing-helm-and-kustomize-what-are-the-differences. [Kasutatud 30.03.2024]

[56] S. Kapare, “Helm vs. Kustomize in Kubernetes,” [Võrgumaterjal]. Kättesaadav: https://medium.com/@sushantkapare1717/helm-vs-kustomize-in-kubernetes-cc063bbb4b0e. [Kasutatud 30.03.2024]

[57] HashiCorp (2024). *Helm Provider for Terraform* [Tarkvara]. GitHub. https://github.com/hashicorp/terraform-provider-helm

[58] CoreDNS (2024). *CoreDNS* [Tarkvara]. GitHub. https://github.com/coredns/coredns

[59] The Kubernetes Authors, “Configure DNS for a Cluster,” Kubernetes Documentation, [Võrgumaterjal]. Kättesaadav: https://kubernetes.io/docs/tasks/access-application-cluster/configure-dns-cluster/. [Kasutatud 30.03.2024]

[60] Kubernetes (2024). *Kubernetes Dashboard* [Tarkvara]. GitHub. https://github.com/kubernetes/dashboard

[61] The Kubernetes Authors, “Deploy and Access the Kubernetes Dashboard,” Kubernetes Documentation, [Võrgumaterjal]. Kättesaadav: https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/. [Kasutatud 30.03.2024]

[62] S. Hussain, “Kubernetes Ingress Controller Examples with Best Option,” [Võrgumaterjal]. Kättesaadav: https://k21academy.com/docker-kubernetes/kubernetes-ingress-controllers/#nginx. [Kasutatud 30.03.2024]

[63] Kubernetes (2024). *Ingress NGINX Controller* [Tarkvara]. GitHub. https://github.com/kubernetes/ingress-nginx

[64] cert-manager (2024). *cert-manager* [Tarkvara]. GitHub. https://github.com/cert-manager/cert-manager

[65] Longhorn.io (2024). *Longhorn* [Tarkvara]. GitHub. https://github.com/longhorn/longhorn

[66] R. Kumar, “Compare RookCeph Vs Longhorn vs OpenEBS,” [Võrgumaterjal]. Kättesaadav: https://www.devopsschool.com/blog/compare-rookceph-vs-longhorn-vs-openebs/. [Kasutatud 30.04.2024]

[67] A. Sharif, “The Three Pillars of Observability: Logs, Metrics, and Traces,” [Võrgumaterjal]. Kättesaadav:

https://www.crowdstrike.com/cybersecurity-101/observability/three-pillars-of-observability/. [Kasutatud 01.04.2024]

[68] Grafana Labs, “The Grafana Stack,” [Võrgumaterjal]. Kättesaadav: https://grafana.com/about/grafana-stack/. [Kasutatud 01.04.2024]

[69] Prometheus (2024). *Prometheus* [Tarkvara]. GitHub. https://github.com/prometheus/prometheus

[70] Grafana Labs (2024). *Loki: Like Prometheus, but for logs* [Tarkvara]. GitHub. https://github.com/grafana/loki

[71] Grafana Labs (2024). *Tempo* [Tarkvara]. GitHub. https://github.com/grafana/tempo

[72] OpenTelemetry - CNCF (2024). *OpenTelemetry Collector* [Tarkvara]. GitHub. https://github.com/open-telemetry/opentelemetry-collector

[73] Grafana Labs (2024). *Grafana* [Tarkvara]. GitHub. https://github.com/grafana/grafana

[74] Benny’s Bytes, “Deploying the Grafana Stack using Helm Charts,” [Võrgumaterjal]. Kättesaadav: https://bennysbytes.dev/docs/o11y-cluster/deploy-grafana-stack/. [Kasutatud 01.04.2024]

[75] G2.com, “Top 10 Grafana Labs Alternatives & Competitors,” [Võrgumaterjal]. Kättesaadav: https://www.g2.com/products/grafana-labs/competitors/alternatives. [Kasutatud 01.04.2024]

[76] N. Vermande, “Looking for a k3OS Alternative? Choosing a Container OS for Edge K8s,” [Võrgumaterjal]. Kättesaadav: https://thenewstack.io/looking-for-a-k3os-alternative-choosing-a-container-os-for-edge-k8s/. [Kasutatud 05.04.2024]

[77] S. Vaughan-Nichols, “What is immutable Linux? Here’s why you’d run an immutable Linux distro,” [Võrgumaterjal]. Kättesaadav: https://www.zdnet.com/article/what-is-immutable-linux-heres-why-youd-run-an-immutable-linux-distro/. [Kasutatud 05.04.2024]

[78] OpenNebula (2024). *OpenNebula Linux VM Contextualization* [Tarkvara]. GitHub. https://github.com/OpenNebula/addon-context-linux

[79] S. Rajhi, “Choosing the right Linux Operating System for you Kubernetes cluster,” [Võrgumaterjal]. Kättesaadav: https://medium.com/@seifeddinerajhi/choosing-the-right-linux-operating-system-for-your-kubernetes-cluster-80a0c97d3619. [Kasutatud 05.04.2024]

[80] Canonical Ltd., “Scale out with Ubuntu Server,” [Võrgumaterjal]. Kättesaadav: https://ubuntu.com/server [Kasutatud 05.04.2024]

[81] MangoHost, “Ubuntu vs. Other Operating Systems: Better choice for a server,” [Võrgumaterjal]. Kättesaadav: https://mangohost.net/blog/ubuntu-vs-other-operating-systems-better-choice-for-a-server/. [Kasutatud 05.04.2024]

[82] Aqua Security Software Ltd., “Container Engines: How The Work and Top 7 Options,” [Võrgumaterjal]. Kättesaadav: https://www.aquasec.com/cloud-native-academy/container-platforms/container-engines/. [Kasutatud 05.04.2024]

[83] CRI-O (2024). *CRI-O - OCI-based implementation of Kubernetes Container Runtime Interface* [Tarkvara]. GitHub. https://github.com/cri-o/cri-o

[84] F5. Inc., “NGINX,” [Võrgumaterjal]. Kättesaadav: https://www.nginx.com/ [Kasutatud 05.04.2024]

[85] T. Tyler, “6 Advantages of Working with a Single Vendor,” [Võrgumaterjal]. Kättesaadav: https://www.verizonconnect.com/resources/article/6-advantages-of-working-with-a-single-vendor/. [Kasutatud 05.04.2024]

[86] A. Cassen (2024). *keepalived* [Tarkvara]. GitHub. https://github.com/acassen/keepalived

[87] Canonical Ltd., “Ubuntu for Desktops,” [Võrgumaterjal]. Kättesaadav: https://ubuntu.com/desktop [Kasutatud 10.04.2024]

[88] JetBrains s.r.o, “IntelliJ IDEA,” [Võrgumaterjal]. Kättesaadav: https://www.jetbrains.com/idea/ [Kasutatud 10.04.2024]

[89] HashiCorp, “Modules,” Terraform Documentation [Võrgumaterjal]. Kättesaadav: https://developer.hashicorp.com/terraform/language/v1.5.x/modules. [Kasutatud 10.04.2024]

[90] HashiCorp, “Providers,” Terraform Documentation, [Võrgumaterjal]. Kättesaadav: https://developer.hashicorp.com/terraform/language/v1.5.x/providers. [Kasutatud 10.04.2024]

[91] HashiCorp, “Files and Directories,” Terraform Documentation, [Võrgumaterjal]. Kättesaadav: https://developer.hashicorp.com/terraform/language/v1.5.x/files. [Kasutatud 10.04.2024]

[92] A. Babenko, “Code Structure,” Terraform Best Practices, [Võrgumaterjal]. Kättesaadav: https://www.terraform-best-practices.com/code-structure [Kasutatud 10.04.2024]

[93] OpenNebula (2024). *Terraform Provider for OpenNebula* [Tarkvara]. GitHub. https://github.com/OpenNebula/terraform-provider-opennebula

[94] HashiCorp, “local-exec Provisioner,” Terraform Documentation, [Võrgumaterjal]. Kättesaadav: https://developer.hashicorp.com/terraform/language/v1.5.x/resources/provisioners/local-exec. [Kasutatud 10.04.2024]

[95] Rancher, “rke_cluster,” Terraform Registry, [Võrgumaterjal]. Kättesaadav: https://registry.terraform.io/providers/rancher/rke/latest/docs/resources/cluster. [Kasutatud 10.04.2024]

[96] The Kubernetes Authors, “Organizing Cluster Access Using kubeconfig Files,” Kubernetes Documentation, [Võrgumaterjal]. Kättesaadav: https://kubernetes.io/docs/concepts/configuration/organize-cluster-access-kubeconfig/. [Kasutatud 10.04.2024]

[97] Helm Authors, “Values Files,” Helm Documentation, [Võrgumaterjal]. Kättesaadav: https://helm.sh/docs/chart_template_guide/values_files/. [Kasutatud 15.04.2024]

[98] Katalon, Inc., “What is End-to-End Testing? Definition, Tools, Best Practices,” [Võrgumaterjal]. Kättesaadav: https://katalon.com/resources-center/blog/end-to-end-e2e-testing. [Kasutatud 15.04.2024]

[99] Nix/Nixpkgs/NixOS (2024). *Nix* [Tarkvara]. GitHub. https://github.com/NixOS/nix

[100] Kured Project (2024). *kured - Kubernetes Reboot Daemon* [Tarkvara]. GitHub. https://github.com/kubereboot/kured

[101] Eraser (2024). *Eraser: Cleaning up Images from Kubernetes Nodes* [Tarkvara]. GitHub. https://github.com/eraser-dev/eraser

[102] Kubernetes SIGs (2024). *Descheduler for Kubernetes* [Tarkvara]. GitHub. https://github.com/kubernetes-sigs/descheduler

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Roman Vilu

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Kubernetesel põhineva taristu haldusmooduli arendamine ja testimine organisatsiooni sisemiseks kasutamiseks", mille juhendaja on Siim Vene

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.

13.05.2024

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Püsilink mooduli koodibaasile

https://github.com/romanvilu/terraform-module-one-rke

Lisa 3 – Testitud funktsionaalsuste nimekiri

Tabel 5. Mooduli testitud funktsionaalsus.

Funktsionaalsus	**Vastuvõtukriteeriumid**
Kasutajanime genereerimine	Kasutajanimi on genereeritud ilma vigadeta.
Parooli genereerimine	Parool on genereeritud ilma vigadeta.
Võtmepaari genereerimine	Võtmepaar on genereeritud ilma vigadeta.
Kasutajanime ja parooli sisaldava lokaalse faili loomine	▪ Lokaalne fail on loodud ilma vigadeta; ▪ Fail sisaldab genereeritud kasutajanime ja parooli.
Privaatvõtme sisaldava lokaalse faili loomine	▪ Lokaalne fail on loodud ilma vigadeta; ▪ Fail sisaldad privaatvõtit genereeritud võtmepaarist.
Kubernetese juhtsõlmede virtuaalmasinate loomine	▪ Virtuaalmasinad on loodud ilma vigadeta; ▪ Virtuaalmasinate nimed OpenNebula veebiliideses vastavad formaadile ja sisaldavad projekti ja keskkonna nimesid; ▪ Virtuaalmasinate arv vastab konfigureeritud väärtusele, sh kui see on vaikeväärtus; ▪ Virtuaalmasinatel on seadistatud kasutaja enne genereeritud nime ja võtmepaariga. Antud andmetega on võimalik virtuaalmasinasse sisse logida; ▪ Virtuaalmasinate vahel on olemas aktiivne võrguühendus; ▪ Virtuaalmasinate arvutusressursside mahud vastavad konfigureeritud väärtustele, sh kui see on vaikeväärtus.
Kubernetese töötajasõlmede virtuaalmasinate loomine	
Koormusjaoturite virtuaalmasinate loomine	
Ansible loendifaili (ingl *inventory file*) loomine	▪ Fail on loodud ilma vigadeta;

Funktsionaalsus	**Vastuvõtukriteeriumid**
	▪ Fail on koostatud korrektses Ansible loendifaili formaadis; ▪ Fail sisaldab enne loodud virtuaalmasinate vastavaid gruppide ja hostide konfiguratsioone, sh virtuaalmasina nimed ja IP-aadresid, ning koormusjaoturi virtuaalset IP-aadressi.
Ansible mänguraamatu täitmine	▪ Mänguraamat on täidetud ilma vigadeta; ▪ Ansible pääseb ligi hostidele kasutades enne genereeritud kasutajanime ja privaatvõtit; ▪ Mänguraamat teostab hostide konfiguratsiooni vastavalt genereeritud loendifailile ja kirjeldatud tegumitele; ▪ Mänguraamatu korduvtäitmine toimub hostide kettaruumi suuruste, loendifaili või mänguraamatu muutmisel.
RKE kobara paigaldamine	▪ RKE kobar on loodud ilma vigadeta; ▪ RKE kobara Kubernetese versioon vastab konfiguratsioonis määratud versioonile; ▪ RKE kobara komponentide seadistus vastab konfiguratsioonis määratud parameetritele, sh kobarasisesed IP-aadresside vahemikud; ▪ RKE kobarasse on lisatud enne loodud virtuaalmasinad Kuberetese sõlmede gruppidest; ▪ Mänguraamatu korduvtäitmine toimub hostide kettaruumi suuruste, loendifaili või mänguraamatu muutmisel.
Kobara ligipääsukonfiguratsiooni	▪ Fail on loodud ilma vigadeta;

Funktsionaalsus	**Vastuvõtukriteeriumid**
(kubeconfig) faili loomine	▪ Failiga on võimalik saavutada administratiivset ligipääsu RKE kobarale; ▪ Fail pääseb ligi RKE kobarale kasutades koormusjaoturi virtuaalset IP-aadressi.
CoreDNS'i evitamine	▪ Helmi pakend on evitatud ilma vigadeta; ▪ Kobara rakendused kasutavad nimeserverina CoreDNS'i teenust; ▪ Kobara rakendused on võimelised lahendada teineteise kobarasisesed domeeninimed.
cert-manageri evitamine	▪ Helmi pakend on evitatud ilma vigadeta; ▪ Avalikustatud kobara rakendustele ligi pääsemine on turvaline (kasutab HTTPS protokolli) ning töötab ilma vigadeta.
NGINX Ingress Controlleri evitamine	▪ Helmi pakend on evitatud ilma vigadeta; ▪ Avalikustatud kobara rakendustele on võimalik pääseda ligi üle veebi.
Kubernetes Dashboardi evitamine	▪ Helmi pakend on evitatud ilma vigadeta; ▪ Dashboardile on võimalik pääseda ligi üle veebi kasutades HTTPS protokolli; ▪ Dashboardi veebiliidesse on võimalik sisse logida kobarasiseselt loodud tookeniga; ▪ Dashboardi veebiliideses on võimalik jälgida ja halata Kubernetese kobara ressurrse.
Longhorni evitamine	▪ Helmi pakend on evitatud ilma vigadeta; ▪ Longhornile on võimalik pääseda ligi üle veebi

Funktsionaalsus	**Vastuvõtukriteeriumid**
	kasutades HTTPS protokolli; ▪ Longhorni veebiliidesse on võimalik sisse logida; ▪ Longhorni veebiliideses on võimalik jälgida ja hallata rakenduste salvesturuumi köiteid; ▪ Longhorn vajadusel varustab rakendusi salvestusruumiga nende käivitumisel.
Grafana Stacki evitamine	▪ Helmi pakend on evitatud ilma vigadeta; ▪ Grafanale on võimalik pääseda ligi üle veebi kasutades HTTPS protokolli; ▪ Grafana veebiliidesse on võimalik sisse logida; ▪ Grafana veebiliideses on võimalik visualiseerida ajaloolisi telemeetriaandmeid (so mõõdikud, logid ja olemasolul jäljed) kasutades vaikenäidulaudu.

Lisa 4 – Tulemuste võrdlev analüüs

Tabel 6. Tulemuste võrdlev analüüs.

Nõue kategooria	**Nõue kirjeldus**	**Saavutatud tulemus**
Funktsionaalsus	Moodul püstitab, kohendab ja hävitab virtuaalse IT-taristu komponente OpenNebula hüperviisoris, sellele paigaldatud Kubernetese kobarat ja kobarateenuseid.	Terraform võimaldab vastavalt etteantud konfiguratsioonile taristut püstitada, kohendada ja hävitada.
	Moodul haldab mitut omavahel mittekattuvat keskkonda. Keskkondade erinevused seisnevad alles nende parameetrite poolelt, mis on spetsiifilised konkreetsele keskkonnale.	Terraform võimaldab keskkonnahaldust sisseehitatud funktsionaalsuse näol. Moodulis keskkonnahalduseks on samuti rakendatud lisaparameeter.
	Moodul võimaldab sisestada järgmiseid konfiguratsiooni parameetreid: ▪ Projekt – projekti, mille raames moodulit kasutatakse, nimi või lühinimi; ▪ Keskkonnd – keskkonna, mis kajastab mooduli poolt püstitava taristu eesmärki, lühinimi. Näide: *dev* (e *development*, arenduskeskkond); ▪ Domeen – peamine DNS-domeen, mille all püstitava keskkonna teenused tehakse kättesaadavaks. Näide: example.com;	Terraformi moodul võimaldab konfigureeritavate parameetrite sisestamist, sh nõutud väärtused.

Nõue kategooria	Nõue kirjeldus	Saavutatud tulemus
	▪ Arvutusressursid – virtuaaltaristu arvutusvõimekuse mahud, so serverite operatiivmälu, protsessor, kettamaht.	
Kasutatavus	Mooduli kasutamine on intuitiivselt selge ja lihtsasti arusaadav.	Terraform on kavandatud olla lihtne õppimiseks ja kasutamiseks. Selle töövoog on selgelt arusaadav.
	Moodul pakub kasutajale selget ja arusaadavat tagasisidet oma töö käigus.	Terraformi käsurida pakub kasutajale arusaadavat perioodilist väljundit.
	Moodul pakub selget dokumentatsiooni ja juhiseid selle kasutamiseks ning konfigureerimiseks.	Arendatud moodul pakub dokumentatsiooni selle rakendamiseks ja kasutamiseks. Dokumentatsioon samuti kirjeldab mooduli sisend- ja väljundväärtuseid ja loodavaid ressursse.
	Moodul nõuab minimaalset konfiguratsiooni ja sisendit kasutaja poolt.	Arendatud moodul sisaldab alles mitut kohustuslikku parameetrit, nt projekti ja keskkonna nimed. Moodulis on esitatud samuti valikulised vaikeväärtustega parameetrid.
Usaldusväärsus	Moodul töötab stabiilselt ja töökindlalt.	Õige parameetrite konfigureerimise puhul on garanteeritud mooduli stabiilne ja kindel töö. Võimalikud vead on enamasti tingitud pilvepakkujate defektidest.
	Mooduli konfiguratsioonid on deklaratiivsed.	Mooduli konfiguratsioonid on enamasti deklaratiivsed. Erandiks on Ansible

Nõue kategooria	Nõue kirjeldus	Saavutatud tulemus
		mänguraamat.
	Moodul pöörab tagasi muudatusi nende ebaõnnestunud rakendamisel.	Terraform ja Ansible ei võimalda ebaõnnestunud muudatuste tagasipööramist. Helmi puhul on võimalik teostada tagasipööre eelmisele versioonile.
	Moodul töötab turvaliselt, kui turvanõuete rikkumine pole tingitud kasutajapoolsest mooduli kasutamisest.	Vastavalt CVE andmebaasile, moodulis kasutatavad tööriistad ja tehnoloogiad ei oma parandamatuid kriitilisi turvanõrkusi.
	Moodul ei rakenda muudatusi, kui süsteemi olek juba vastab kirjeldatud konfiguratsioonile.	Terraformi kasutab deklaratiivset lähenemist ning tagab idempotentsuse konfiguratsioonide rakendamisel. Ansible ja Helmi idempotentsus arendatud moodulis on samuti tagatud Terraformi funktsionaalsuse abil.
	Moodul haldab püstitava infosüsteemi olekufaile ja võimaldab nende salvestust nii lokaalselt, kui ka kaugelt.	Terraform kasutab olekufaile süsteemi oleku salvestamiseks. Vaikimisi olekufailid hoitakse lokaalselt, kuid Terraform pakub mitmeid variante nende kaugsalvestamiseks.
Jõudlus	Moodul töötab kiirelt ja tõhusalt.	Terraform töötab kiirelt ja tõhusalt, mis on tingitud programmeerimiskeele Go jõudlusest.
	Mooduli täitmisaeg ja rakendamise kiirus on optimaalne ooteaja poolest.	Moodul ei sisalda liigseid ressursse ja kasutamata väärtusi. Mooduli täitmisaeg

Nõue kategooria	Nõue kirjeldus	Saavutatud tulemus
		vaikeväärtustega on ~30 minutit.
	Moodul lihtsasti kohaneb erinevate hallatava ressursi mahtudega.	Ressursside arvu on võimalik vajadusel hõlpsasti suurendada, mis mõjutab täitmisaega, kuid mitte otseselt mooduli täitmise jõudlust.
Toetatavus	Moodul võimaldab versioonihaldust.	Terraformi konfiguratsioone on võimalik salvestada koodihoidlas. Arendatud moodul on samuti avalikustatud GitHubi koodihoidlas.
	Mooduli struktuur on loogiline ja intuitiivselt selge.	Mooduli kataloogi struktuur on koostatud vastavalt parimatele praktikatele. Moodul järgib lihtsat ja arusaadav struktuuri.
	Moodul on laiendatav ja muudetav.	Moodul on koostatud vastavalt parimatele praktikatele. Moodulis ei esine laiendamist või muutmist takistavaid konfiguratsioone või sõltuvusi.
	Moodul on kirjutatud ja kommenteeritud inglise keeles.	Moodul on täies ulatuses kirjutatud ja dokumenteeritud inglise keeles.
	Moodul on käivitatav Ubuntu 22.04 LTS operatsioonisüsteemil.	Moodul on kirjutatud ja testitud arenduskeskkonnas, mis põhineb Ubuntu 22.04 LTS operatsioonisüsteemil.
	Moodul kasutatud komponendid ja tehnoloogiad on vabavaralised.	Moodulis kasutatud tehnoloogiad ja tööriistad on vabavaralised ja tasuta.

Nõue kategooria	Nõue kirjeldus	Saavutatud tulemus
	Moodul kasutab ainult ’taristu kui teenuse’ taseme funktsionaalsust virtuaalressursside püstitamiseks.	Moodul kasutab ainult virtuaalarvutuse baasressursse ega rakenda OpenNebula spetsiifilisi teenuseid, millistel pole alternatiivi teiste hüperviisorite puhul.