TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Maido Paalmäe 206637IAAB

Ettevõtte IT-sisetaristu arendus tarkvarateenust pakkuva ettevõtte näitel

Bakalaureusetöö

Juhendajad: Kaido Kikkas PhD
Karl Hendrik Leppmets BSc

Tallinn 2023

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: Maido Paalmäe

15.05.2023

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on leida sobivaim lahendus tarkvarateenust osutava ettevõtte IT-siseteenuste sisemajutamiseks ning luua sidusarendust võimaldav keskkond ettevõttes arendatavale sisehaldusportaalile.

Töö käigus analüüsib autor kuute levinuimat automatiseeritud konteinerihalduse lahendust tuues välja sobivaimad ja ebasobivaimad aspektid pidades silmas sisemajutatud taristu eeldusi ja võimalusi. Praktilises osas paigaldab autor valitud virtuaalse taristu lahenduse rakendades taristu koodina põhimõtteid ning kasutades selleks Ansible'it. Lisaks juurutab autor paigaldatud keskkonnas sidusarenduse konveieri automatiseerimaks sisehaldusportaali koodi kompileerimist, testimist ja paigaldamist.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 8 peatükki, 13 joonist, 1 tabelit.

Abstract

Development of Internal IT Infrastructure Based on a SaaS Software Company

The aim of this thesis is to find the most suitable solution for self-hosting the internal IT services of a SaaS software company and creating development environment with automated continuous deployment for the internal management portal being developed internally.

The author analyses the six most common container orchestration solutions, highlighting the most suitable and unsuitable aspects, taking into consideration the specific requirements of a self-hosted infrastructure. In the practical part of the thesis, the author installs the chosen suitable virtual infrastructure solution applying the infrastructure as a code principles using Ansible. In addition, the author implements a continuous integration and continuous deployment pipeline to automate the compiling, testing and deployment processes of the internal management portal.

The thesis is in Estonian and contains 25 pages of text, 8 chapters, 13 figures, 1 table.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface*, rakendusprogrammiliides
CI/CD	*Coninuous Integration / Continuous Delivery*, pidev integreerimine ja pidev tarnimine (sidusarendus)
CLI	*Command Line Interface*, käsurida
CNCF	*Cloud Native Computing Foundation*, pilvepõhiste rakenduste sihtasutus
DHCP	*Dynamic Host Configuration Protocol*, hosti dünaamilise konfigureerimise protokoll
DNS	*Domain Name System*, domeeninimede süsteem
GB	*Gigabyte*, gigabait
IaaC	*Infrastructure as a Code*, taristu koodina
IP aadress	*Internet Protocol Address*, interneti protokolli aadress
IPI	*Installer Provisioned Infrastructure*, paigaldusprogrammi loodud taristu
ISP	*Internet Service Provider*, interneti teenusepakkuja
NFS	*Network File System*, võrgufailisüsteem
OKD	*Origin Community Distribution of Kubernetes*, algne kogukonna Kubernetese distributsioon
PV	*Persistent Volume*, püsisalvestuspind
PVC	*Persistent Volume Claim*, püsisalvestuspinna taotlus
RKE	*Rancher Kubernetes Engine*, Rancheri Kubernetese mootor
SaaS	*Software as a Service*, tarkvara teenusena
UI	*User Interface*, kasutajaliides
USB	*Universal Serial Bus*, universaalne jadasiin
vCPU	*Virtual Central Processing Unit*, virtuaalne protsessor
VPN	*Virtual Private Network*, virtuaalne privaatvõrk
YAML	*YAML Ain't Markup Language,* YAML pole märgistuskeel (inimloetav andmete jadastuse keel)

Sisukord

1 Sissejuhatus 10
2 Taust 12
2.1 Hetkeolukord 12
2.2 Lähtetingimused 13
2.3 Eeldused 13
3 Metoodika 16
3.1 Tegevusuuringu ülevaade 16
3.2 Etapid 16
3.2.1 Planeerimine 16
3.2.2 Tegutsemine 17
3.2.3 Vaatlemine 17
3.2.4 Analüüsimine 17
4 Analüüs 18
4.1 Levinud lahendused 18
4.2 Lahenduste võrdlus 19
4.3 Sobivaim lahendus 21
5 Taristu paigaldamine 23
5.1 Eeltingimused 23
5.2 Paigaldamine 23
5.3 Seadistamine 26
5.3.1 Püsisalvestuspind (PV) 26
5.3.2 Konteineritõmmiste register 27
5.3.3 Sisehaldusportaali paigaldamine 28
5.3.4 CI/CD konveier 29
6 Tulemused 31
7 Järeldused ja edasised tegevused 33
8 Kokkuvõte 34
Kasutatud kirjandus 35

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks 41
Lisa 2 – Kontainerihalduse lahenduste populaarsuse analüüs 42
Lisa 3 – OpenShifti paigaldamise Ansible mänguplaan 43
Lisa 4 – Helm pakett 50
Lisa 5 – GitLabi CI/CD konveieri fail .gitlab-ci.yaml 54

Jooniste loetelu

Joonis 1. Andmekeskuse lihtsustatud ülevaade. 13
Joonis 2. Sidusarenduse konveieri etapid [18]. 14
Joonis 3. Tegevusuuringu etapid [22]. 16
Joonis 4. OpenShifti ülesehitus [64]. 21
Joonis 5. OpenShifti paigaldamise ülevaade [24]. 24
Joonis 6. OpenShift paigaldusprotsessi lõppinfo. 25
Joonis 7. OpenShift klaster vSpheres. 25
Joonis 8: Graafiline OpenShifti veebiliides. 26
Joonis 9. Püsisalvestuspinna lisamine OpenShift klastrisse. 27
Joonis 10. Konteineritõmmiste registri konfiguratsioon. 27
Joonis 11. Konteineritõmmiste registri staatuse kontroll. 28
Joonis 12. Konteinergruppide uuendamine. 30
Joonis 13. Sisehaldusportaali paigaldus. 31

Tabelite loetelu

Tabel 1. Klastrisõlmede spetsifikatsioonid [70]. 24

1 Sissejuhatus

Samal ajal kui on käimas erinevate pilvetehnoloogiate võidukäik ja üha rohkem ettevõtteid kolivad oma IT-siseteenuste haldamist üle pilvetehnoloogiatele [1], jääb järjest vähemaks vastavate teenuste sisemajutamist. Pilvetehnoloogiate kasutuselevõtt on kordades väiksema kuluga kui sisemajutatud [2] - omades praktiliselt nullilähedast alguskulu -, kuid pikemas perspektiivis on ettevõttele siiski soodsam servereid ja teenuseid sisemajutada [3]. Tähtsat rolli mängib ka pilvetaristu ja sisemajutatud taristu turvaerinevused, milles oma olemuselt on pilvetehnoloogiad limiteeritud riistvara kontrolli tõttu mõningaste märkimisväärsete turvariskidega [4].

Käesoleva töö eesmärgiks on mõista IT-sisetaristu nõudeid, väljakutseid ja eripärasid analüüsides erinevaid virtuaalse taristu lahendusi ning leides nende seast sobiva nõuetele vastava süsteemi, mille paigaldamisega süvendaks autor erialaseid teadmisi ja oskusi. Samuti otsib autor käesolevas töös vastust järgnevatele küsimustele:

1. Milline lahendus sobib tarkvarateenust osutava ettevõtte virtuaalse sisemajutatud sisetaristu loomiseks?

2. Mille poolest erinevad eksisteerivad lahendused ning millised neist on lihtsamini hallatavad ja paigaldatavad arvestades ettevõtte limiteeritud ressursse?

Käesolevas töös rakendab autor tegevusuuringu põhietappe, mille käigus uuritakse ja võrreldakse sobivaid virtuaalse taristu lahendusi. Seejärel paigaldatakse ja seadistatakse neist kõige sobivam, tagades arendusjärgus oleva sisehaldusportaali kõrgkäideldav majutuse ja sidusarenduse keskkond.

Ülejäänud töö on üles ehitatud järgnevalt. 2. peatükis on kirjeldatud hetkeolukorda ja lähtetingimusi, millele vastavalt otsib autor ettevõttele parimat võimalikku lahendust. 3. peatükis toob autor välja detailse metoodika kirjelduse, kuidas käesolevas töös esitatud probleemile läheneti ja kuidas lahenduseni jõuti. Peatükis 4 analüüsib autor enamlevinumaid virtuaalse taristu lahendusi ning valib nende seast välja ettevõttele sobivaima. 5. peatükk kirjeldab valitud keskkonna ja lahenduse paigaldamist ning

seadistamist. Peatükis 7 analüüsib autor paigaldatud lahenduse sobivust ja kasutajamugavust ning administreerimise lihtsust ettevõtte siseteenuste majutamiseks pidades silmas teises peatükis esitatud nõudeid ja eeldusi.

2 Taust

Käesolev peatükk annab ülevaate näidisettevõttes valitsevast IT-sisetaristu hetkeolukorrast, hetkeolukorra probleemidest ja puudujääkidest. Samuti tuuakse välja ootused ja eesmärgid valmivale virtuaalsele taristule.

2.1 Hetkeolukord

Ettevõttes, mille näitel käesolev töö on kirjutatud, on siiani kasutanud oma siseteenuste (nagu arendatav sisehaldusportaal, DHCP, DNS, VPN jms) majutamiseks Intel NUC [5] miniarvuteid. Kuigi need miniarvutid on seni täitnud oma ülesandeid, esineb praeguse lahenduse juures mitmeid probleeme.

Miniarvutid, mida kasutatakse pakuvad piiratud jõudlust ja ressursse, mis omakorda võivad mõjutada teenuste töökindlust. Intel NUC miniarvutid on küll võimekad, kuid need on loodud kodukasutajate vajadusi silmas pidades ning lisaks sellele on miniarvutite üheks tunnuseks nende komponentide kompaktsus ja asendamatus [6]. Näiteks on Intel NUC miniarvutil ainult üks integreeritud võrgukaart, mis teeb kõrgkäideldava võrguühenduse loomise keeruliseks. See tähendab, et kasutada tuleks lisaks välist USB võrguadapterit, mis on pigem hädapärane paikamine kui lahendus.

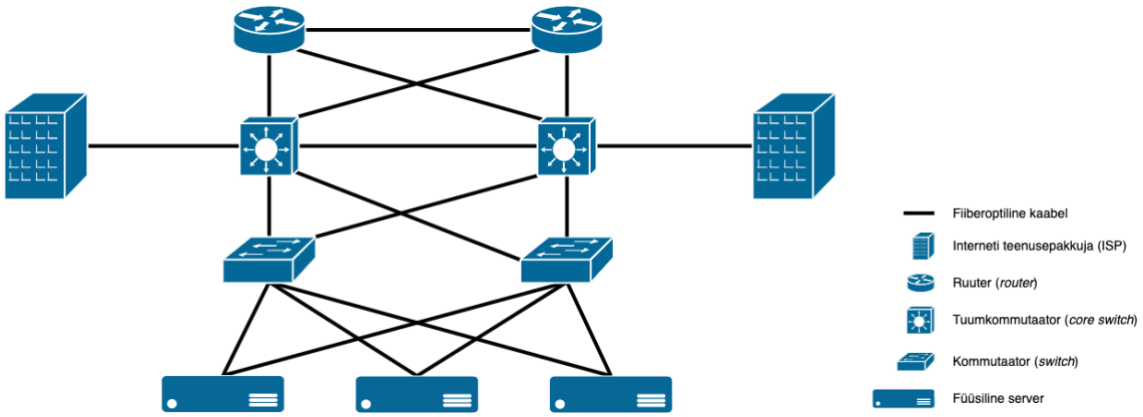
Praegune olukord ei paku lisaks ka riistvara veakindlust, omades mitmeid nõrkasid lülisid, mis tähendab, et ühe süsteemi komponendi rikke tõttu võib kogu teenus olla kättesaamatu, põhjustades töökatkestuse tervele kontorile ja mõjudes negatiivselt produktiivsusele.

Kasutusel olevad ja majutatud teenused ning taristu ei oma lisaks ka avariijärgset taastevõimekust (*disaster recovery*), mis võib tekitada olukorra, kus ettevõtte kontoris kaob ühendus välismaailmaga või on see väga tundlik tarkvarariketele ja kergesti haavatav ohusubjektide poolt. Selline süsteemi haavatavus ei ole aga ettevõtte ärieesmärkide ja infoturbestandarditega vastavuses.

2.2 Lähtetingimused

Uue virtuaalse taristu loomiseks on ettevõttes soetatud kolm füüsilist serverit *HP ProLiant DL360 Gen10 Plus* [7] ja millele on installeeritud VMware vSphere [8] virtualiseerimise tarkvara. VMware vSphere on üks tuntumaid ja suurima turuosaga virtualiseerimise tehnoloogiatest [9].

Füüsilised serverid asuvad andmekeskuses, mille võrgul on fiiberoptiline otseühendus ettevõtte kontoriga ning on seega juurdepääsetav ettevõtte sisevõrgust. Joonis 1 annab lihtsustatud ülevaate andmekeskuses olevast füüsilisest seadistusest.



Joonis 1. Andmekeskuse lihtsustatud ülevaade.

Joonis 1 kujutatud füüsiline seadistus omab dubleeritud võrgulülisid ja seadmeid, suurendades sellega füüsilise taristu veakindlust, tagades erinevate võrguseadmete riistvaraprobleemide puhul internetiühenduse ja serverite töö. Kolm füüsilist serverit omakorda annavad veel võimaluse virtuaalsed teenused ja taristu dubleerida, tagades sellega ka tarkvarateenuste veakindluse ja kõrgkäideldavuse.

Lisaks on ettevõttes arendatava sisehaldusportaali lähtekoodi versioonihalduseks kasutusel GitLabi [10], mis pakub tarkvaraarenduse elutsükli terviklahendust [11].

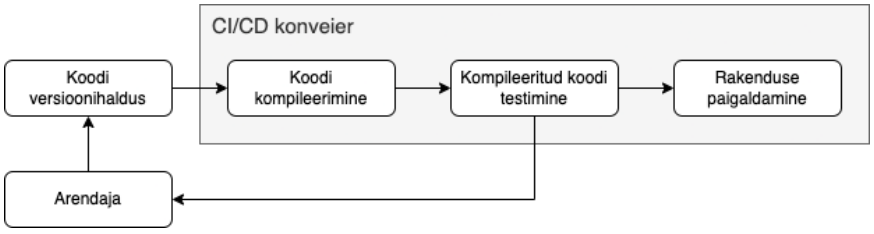
2.3 Eeldused

Sisehaldusportaali, mida ettevõttes arendatakse ja millele on autori loodav taristu peamiselt mõeldud, arendatakse mikroteenuste arhitektuurile põhinevalt ja sellest tulenevalt vajab see mikroteenuseid toetavat virtuaalset taristut. Konteineritest on saanud üks populaarsemaid viise mikroteenuste paigaldamiseks ja Docker [12] on selleks üks

enimkasutatud vahendeid [13], [14]. Ühe tööjaama piires on konteinerite teineteisega suhtlemist, tänu Dockeri konteineritele loodavale võrgule, lihtne korraldada kuid kõrgkäideldavuse ja veakindluse tagamiseks konteinerid mitmele erinevale tööjaamale paigutades tekib kiiresti probleem, mis mastaabi suurenedes muudab konteinerite haldamise väga keeruliseks [15]. Konteinerite isoleeritusest [16] ja hallatavust lahendab automatiseeritud konteinerihaldus (*container orchestration*), mis vähendab rakenduste arendamise keerukust ja lihtsustab konteinerite vastupidavust, nende turvalisust ja haldamist [17].

Lisaks automatiseeritud konteinerihaldusele on oluline rakendada ka sidusarenduse (edaspidi CI/CD) üldlevinud ja tunnustatud põhimõtteid. Sidusarenduse konveier (*CI/CD pipeline*) muutub järjest olulisemaks ja vajalikumaks peamiselt just mikroteenuste puhul kuna üha suurenev kogus mikroteenuseid lisab järjest rohkem nende haldamise keerukust [18].

Ilma CI/CD lahenduseta on vaja rakenduse koodi kompileerida, testida ja juurutada käsitsi protsessina, mis teeb selle väga aeganõudvaks ja ressursse raiskavaks tegevuseks [11]. Lisaks võimaldab CI/CD rakendustesse kiiresti uut funktsionaalsust lisada, vältides liigsuuri muudatusi, mis võivad oma keerukuse tõttu rakendusi kasutuskõlbmatuks muuta [19]. Lihtsustatud ülevaate sidusarenduse konveieri sammudest annab Joonis 2. Kuna ettevõttes on juba kasutusel GitLab versioonihaldustarkvara, mis pakub lisaks versioonihaldusele ka sidusarenduse konveieri võimalusi [11], siis on eelduseks, et sidusarenduse konveier on arendatud GitLabi võimalusi kasutades.



Joonis 2. Sidusarenduse konveieri etapid [18].

Valmivalt virtuaalse taristu lahenduselt eeldab ettevõte ka lihtsustatud hallatavust, vähendades keerukust ja sellele vajaminevaid ressursse kus vähegi võimalik, mis on

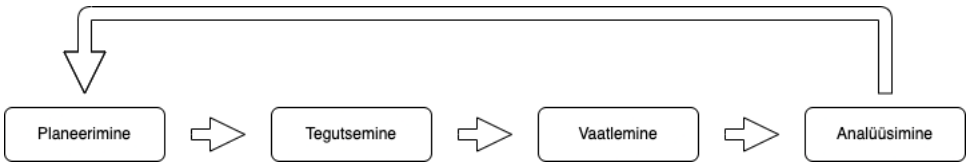
oluline peamiselt väikese meeskonna tõttu. Taristu haldamine peaks olema detailselt ettevõttesiseselt dokumenteeritud ja lihtsasti omandatav, vähendades selle haldamise õppimiseks kulutatavat aega.

3 Metoodika

Selles peatükis annab autor ülevaate tegevusuuringust ning selle metoodika rakendamisest käesoleva töö koostamisel.

3.1 Tegevusuuringu ülevaade

Käesoleva töö metoodikana valis autor tegevusuuringu, mis eristub teistest uurimisviisidest praktilise rakendatavuse poolest [20]. Tegevusuuringu alguseks peetakse 1946. aastal K. Lewini poolt avaldatud tööd „*Action Research and Minority Problems*“ [21], hiljem on seda metoodikat põhjalikult täpsustanud ja selgitanud veel mitmed teadlased [22]. Tegevusuuring on oma olemuselt iteratiivne, hõlmates endas planeerimise, tegutsemise, vaatlemise ja analüüsimise etappe ja milles iga analüüsi etapi järel on võimalik luua uus tegevuskava alustades sellega uut uuringu tsüklit [20], [22]. Tegevusuuringu etappidest annab ülevaate Joonis 3.



Joonis 3. Tegevusuuringu etapid [22].

3.2 Etapid

Järgnev peatükk toob välja ülevaate etappidest, millele autor töö käigus keskendub.

3.2.1 Planeerimine

Planeerimise etapis kaardistab autor loodava taristu vajadused ja hetkeolukorra, pidades silmas ettevõtte poolt esitatud nõudeid ja füüsilise taristu võimalusi ja iseärasusi. Samuti võtab autor planeerimise etapis vaatluse alla kuus enamlevinumat ja tuntumat automatiseeritud konteinerihalduse lahendust, analüüsides nende sobivust ettevõtte nõuetega, ja teeb valiku, millise lahendusega liikuda edasi tegutsemise etappi. Kuna

lahenduste üks-ühele võrdlemine on keeruline [23], analüüsib autor lahendusi ettevõtte vajaduste seisukohast.

Vaatluse all oleva kuue automatiseeritud konteinerihalduse tööriista (vt 4.1.1) valiku tegi autor kümne erineva allika kõige enam mainitud ja käsitletud lahenduste põhjal. Selleks kasutatud allikate nimekirja ja analüüsi kokkuvõtva tabeli leiab lisast 2. Valikust ja allikates mainimise nimistust on välja jäetud pilveteenuse pakkujate automatiseeritud konteinerihalduse lahenduste distributsioonid.

3.2.2 Tegutsemine

Planeerimise etapile järgneb tegevusuuringu tegutsemise etapp [20], mille käigus keskendub autor valituks osutunud lahenduse paigaldusele, seadistamisele ja testimisele.

Paigaldamise etapp hõlmab endas tarkvara allalaadimist, paigaldamist, konfiguratsioonifailide seadistamist ja vajalike ressursside nagu virtuaalne riistvara ja võrguühenduse määramist. Paigaldamise etapi käigus järgib autor ametlikku paigaldusjuhendit [24] ja juhendeid raamatust „*Operating OpenShift“* [25] ning dokumenteerib kogu protsessi ettevõttesiseses dokumentatsioonis.

3.2.3 Vaatlemine

Vaatlemise etapis testib autor paigaldatud lahendust ning vaatleb selle toimimist CI/CD andmekonveieris (*CI/CD pipeline*). Selle etapi eesmärk on hinnata lahenduse toimimist, vastupidavust ja sobivust ettevõtte vajadustega. Testimise käigus tuvastatud probleemide lahendamist ja optimeerimist analüüsib autor järgnevas tegevusuuringu etapis.

3.2.4 Analüüsimine

Analüüsimise etapp keskendub tulemuste hindamisele ning aitab tuvastada paigaldatud lahenduse tugevusi ja nõrkusi tuues välja võimalikke lahendusi ja soovitusi lahenduse optimeerimiseks. Analüüsimise etapp lõpeb järelduste ja edasiste tegevuste väljatöötamisega, põhinedes analüüsi tulemustel ja soovitustel.

4 Analüüs

Järgnevas peatükis uurib autor levinumaid konteinerihalduse automatiseerimise lahendusi ning toob välja mõned peamised erinevused, mis on olulised paigaldatava lahenduse valikul. Eksisteerivad automatiseeritud konteinerihalduse platvormid on arenenud osalt vägagi erinevates suundades, mis teeb nende üks-ühele võrdluse keeruliseks [23], mistõttu ei ole väljatoodud võrdlus lõplik vaid lähtub autori seisukohast ja ettevõtte vajadustest.

4.1 Levinud lahendused

Kuus levinuimat automatiseeritud konteinerihalduse lahendust on Kubernetes [26], Nomad [27], OpenShift [28], Mesos [29], Docker Swarm [30] ja Rancher [31]. Valik on teostatud kümne erineva allika enim käsitletud automatiseeritud konteinerihalduse lahenduse põhjal. Allikad, mille põhjal see valik on tehtud, leiab lisast 2.

Kubernetes on Google poolt 2014. aastal loodud [32], [33] ja aastast 2016 *Cloud Native Computing Foundation* (edaspidi CNCF) hallatav automatiseeritud konteinerihalduse lahendus [33]. Vastavalt Datadog tehtud uuringule, milles vaadeldi rohkem kui poolteist miljardit konteinerit, on Kubernetes kõige populaarsem automatiseeritud konteinerite haldamise lahendus [34].

Nomad on HashiCorpi poolt 2015. aastal loodud avatud lähtekoodiga konteinerihalduse lahendus [35], mis lisaks konteineritele toetab ka mitmete pärandvara rakenduste automatiseeritud haldamist [36]. Tuntud ettevõtetest kasutab Nomadi oma taristus Cloudflare [37].

OpenShift on Red Hati poolt 2011. aastal loodud automatiseeritud konteinerihalduse tööriist, mis alates 2014. aastast kasutab oma põhjana Kubernetese konteinerihalduse mootorit, mis võimaldab automatiseeritud konteinerihaldust üle mitme tööjaama [38]. Red Hat OpenShift baseerub avatud lähtekoodiga projektil OKD (endise nimega OpenShift Origin [39]) [40], mis erineb OpenShiftist peamiselt ärikasutajatele suunatud lisateenuste nagu tehnilise toe, turvareaktsiooni meeskonnad, valideeritud kolmandate

osapoolte jms poolest [41]. Käesolevas töös käsitleb autor OpenShifti ja OKD sünonüümsetena, keskendudes selle vabavaralisele versioonile.

Apache Mesos on 2009. aastal Berkeley Ülikooli doktorantide loodud hajussüsteemide lahendus, mis koos Marathoniga moodustavad automatiseeritud konteinerihalduse platvormi [42], [43]. Mesost kasutab tuntud ettevõtetest näiteks Twitter, Netflix ja Airbnb [42]. Edaspidiselt peab autor Mesose all silmas Mesose ja Marathoni kombinatsiooni.

Docker Swarmi algus ulatub aastasse 2014 kommunikatsiooniprotokolliga Beam, mille Docker novembris samal aastal nimetas ümber Swarmiks [44]. 2014. aastal avaldatud Docker Swarmil aga ilmnesid tsentraliseeritud süsteemi skaleerimise probleemid, mistõttu läbis see põhjaliku ümber disainimise ja 2016. aastal avaldati see nimega Docker Swarm Mode [44]. Kuigi Docker Swarm ja Docker Swarm Mode on küll olemuselt erinevad [45], peab autor edaspidiselt käesolevas töös Docker Swarmiga silmas Docker Swarm Mode'i. Docker Swarm Mode on Dockeri mootorisse sisse ehitatud ning on hõlpsasti juhitav Dockeri CLI kaudu [30].

Rancher sai alguse aastal 2014 ning oli loodud mõttega, et kõik teenused peaksid olema avatud lähtekoodiga ja vabatahtlike arendatud [46]. 2020. aastal omandas SUSE [47] Rancheri, et lisada automatiseeritud konteinerihaldus oma tooteportfelli [48]. Rancher ei ole ise automatiseeritud konteinerihalduse lahendus, pigem aga Kubernetese klastrite haldamise vahend, mis muuhulgas pakub võimalust ka Kubernetese klaster paigaldada [49], [50].

4.2 Lahenduste võrdlus

Kubernetese plussiks on suur aktiivne kogukond [17], mis teeb abi saamise hõlpsamaks ja kättesaadavamaks. Lisaks on Kubernetesel vastavalt Truyen *et al*. tehtud uuringule, kõige suurem kogus omadusi ja funktsioone võrreldes teiste levinud automatiseeritud konteinerihalduse lahendustega [23].

Nomad paistab silma oma paigalduslihtsusega, sest see koosneb ainult ühest iseseisvast kahendfailist, mis kombineerib endas ressursside haldamise ja planeerimise ühe kompaktse süsteemina [27]. Aidates seeläbi vähendada suurte süsteemide keerukust ja suurendades kasutajate efektiivsust ja väledust [51]. Vaatamata Nomadi lihtsusele on

selle toetajaskond väiksem kui Kubernetesel, mistõttu on kõikvõimalike abimaterjalide kogus väiksem ja selle arendus väga tihedalt seotud HashiCorpiga [52]. Samuti on Nomadile HashiCorpi andmetel 30 välist lisatööriista, millega Nomadi funktsionaalsust tõsta [53], võrreldes enamaga kui 250 Kubernetesel [54].

OpenShift on paigaldamiseks üks lihtsamaid ja parima turvalähenemisega lahendusi [17]. Lisaks kuna OpenShift on Kubernetese najal arendatud, toetab see Kubernetese võimalusi ja platvormi olles lihtsam omades sisseehitatud veebipõhist UI'd, mis Kubernetese puhul tuleks eraldiseisvalt paigaldada [55]. Samuti on OpenShifti eeliseks sisseehitatud süsteem lähtekoodist konteineritõmmiste loomiseks, see aga vähendab süsteemi keerukust [56].

Mesos on loodud suuri andmekeskusi silmas pidades ja on võimeline edukalt koordineerima kuni 50 000 sõlme (*node*) [57] võrreldes Kubernetese 5000-ga või Nomadi 10 000'ga [52]. Mesos on kui „andmekeskuse operatsioonisüsteem", mis suure miinusena ilma lisatööriistade ja konfiguratsioonita ei toeta automaatset ressursinäitajatele (nt protsessori ja mälu utilisatsioon) vastavat skaleerimist [42].

Docker Swarmi eeliseks on aga selle ametlik tugi Dockeri konteineritele sh on see otseselt integreeritud Dockeri API'ga [58] ja võrreldes Kubernetesega vajab vähem lisakomponente klastri moodustamiseks [59] ning on võrreldes Kubernetesega palju kiirema hosti tõrkesiirdega (*failover*) [60]. Docker Swarmi negatiivseteks külgedeks on limiteeritud kohandamise võimalused ja laiendused ning väiksem automatiseerimise võimekus võrreldes Kubernetesga [61].

Rancheri eesmärgiks on lihtsustada Kubernetese klastrite haldamist. Rancher pakub võimalust hallata erinevaid, nii sisemajutatud kui ka pilveteenusena pakutavaid, Kubernetese klastreid. Lisaks võimaldab Rancher ka RKE'le (*Rancher Kubernetes Engine*) põhinevat klastrit luua. [48]

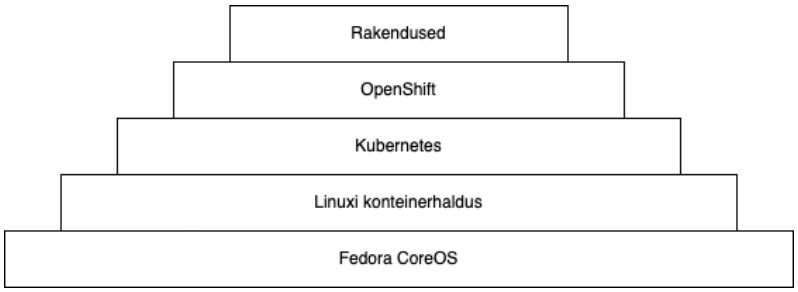
Rancheri miinusteks on aga paljuski erinevate lisade puhul välistest ja kolmandatest osapooltest sõltumine, näiteks nagu konteinerite võrgu haldamise pistikprogrammid [46] lisades sellega süsteemile keerukust.

4.3 Sobivaim lahendus

Pidades silmas Kubernetese võimalusterohkust [23], populaarsust [13] ja OpenShifti paigaldamise lihtsust ja lisa turvafunktsionaalsust [17] ning seda, et OpenShifti näol on tegemist Kubernetese distributsiooniga [41], mis sisaldab Kubernetese platvormi ja võimalusi [55], leiab autor, et kõige sobivam automatiseeritud konteinerihalduse lahendus ettevõtte virtuaalse sisetaristu loomiseks on OpenShift/OKD.

OpenShift kasutab sõlmede operatsioonisüsteemina Fedora CoreOS'i [24], mis on automaatselt uuendatav minimaalne konteinerite käitamiseks mõeldud operatsioonisüsteem [62]. Fedora CoreOS'i suureks eeliseks teiste operatsioonisüsteemide ees on selle minimaalsus, konteinerite käitamisele spetsialiseerumine ja muutmatus – see tähendab, et operatsioonisüsteemi uuendatakse ja käitatakse samasugustel põhimõtetel nagu konteinereidki neid mallidest luues ja vanu kustutades. See teeb omakorda operatsioonisüsteemi uuendused automaatseks vähendades haldamisele kuluvaid ressursse ja aega. [63]

Kuigi Rancher pakub sarnaselt OpenShiftile täisteenust, on Rancheri miinuseks (mõnes teises kontekstis on see aga plussiks) OpenShifti ees selle võimalus olla installeeritud pea igale Linuxi hostile, millele on võimalik paigaldada Docker. See, aga omakorda vajab lisaressursse Rancherit majutavate sõlmede tarkvarauuenduste haldamiseks [48], mis pidades silmas haldamise lihtsuse nõuet, teeb OpenShifti ettevõttele atraktiivsemaks.



Joonis 4. OpenShifti ülesehitus [64].

Kombinatsioon Kubernetesest ja OpenShiftist on sobiv lahendus limiteeritud ressursside puhul lihtsustades Kubernetese hallatavust ning lisades funktsionaalsust, mille lisamine tavapärasele Kubernetese seadistusele võib osutuda kordades keerulisemaks protsessiks. Joonis 4 annab ülevaate kuidas OpenShift toetub Kubernetesele. OpenShiftil on lisaks

sisseehitatud graafiline veebikasutajaliides, mis võimaldab lihtsustatult klastrit hallata ning saada visuaalset ülevaadet ressursside kasutusest ja klastri toimimisest.

OpenShift ühildub ka GitLabiga, luues tervikliku lahenduse alustades rakenduse lähtekoodist ja lõpetades rakendusele juurutamisega [11].

5 Taristu paigaldamine

Käesolev peatükk annab ülevaate OpenShift paigaldamisest ning selle protsessi automatiseerimisest. Paigaldamisel ja seadistamisel järgib autor ametlikku OpenShift/OKD paigaldusjuhendit [24] ning juhendeid raamatust „*Operating OpenShift*“ [25]. GitLabi sidusarenduse konveieri seadistamisel järgib autor juhendeid raamatust „*Automating DevOps with GitLab CI/CD Pipelines*“ [11] ja GitLabi ametlikku dokumentatsiooni [65].

5.1 Eeltingimused

Autori eesmärgiks on OpenShifti paigaldus automatiseerida kasutades IaaC (taristu koodina) lähenemist, mille eesmärgiks on vähendada eksimuste võimalust manuaalsetes protsessides. IaaC on protsess, milles taristu komponendid ja seadistused paigaldatakse ja hallatakse kasutades korduvkasutatavaid skripte [66]. Autor kasutab taristu paigalduse automatiseerimiseks Ansible’it [67], mis on üks populaarsemaid ja enimkasutatud IaaC tööriistasid [68].

Üks eeltingimus enne OpenShifti paigaldamist on DNS kirjete loomine API ja rakenduste väravaks (*ingress*) [69]. Oluline on need DNS kirjed seadistada enne paigaldusprotsessi kuna paigaldusprotsess pöördub paigaldatavate virtuaalmasinate poole DNS kirjete kaudu mitte IP aadresside.

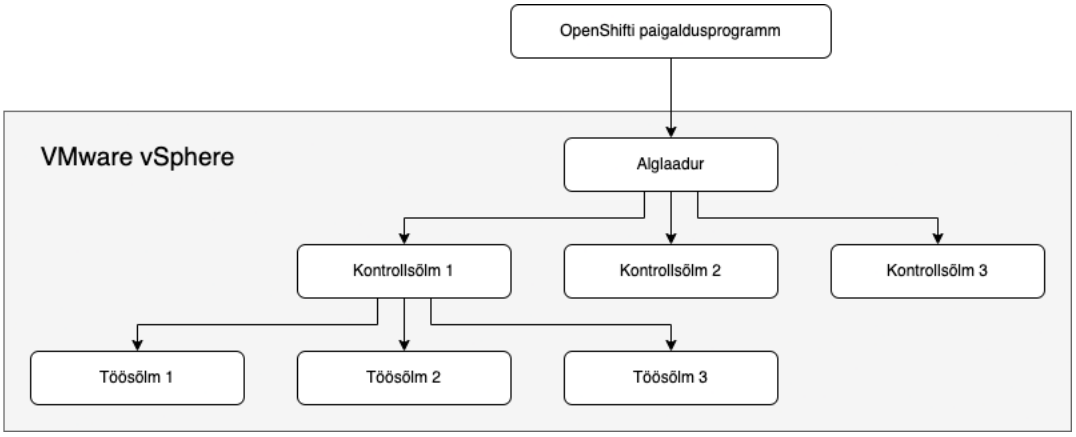
Eeltingimuseks on ka püsiva salvestuspinna (*persistent volume*) loomine OpenShifti sisseehitatud privaatse konteineritõmmiste registeri (*private image registry*) tarbeks mahuga vähemalt 100 GB [69]. Selleks on käesoleva töö skoobi väliselt autor loonud jagatava NFS serveri, mille saab lisada klastrile selle seadistamise käigus.

5.2 Paigaldamine

Openshifti paigaldamiseks on mitmeid erinevaid võimalusi, mille hulgas on nii interaktiivseid kui ka täismanuaalseid võimalusi [24] olenevalt platvormist, millele

OpenShift paigaldatakse. OpenShifti paigaldustarkvara toetab vSphere platvormil interaktiivset IPI (*Installer Provisioned Infrastructure*) paigaldust, mis loob automaatselt kõik klastriks vajalikud virtuaalmasinad [24]. Joonis 5 annab ülevaate OpenShifti IPI virtuaalmasinate loomisest vSphere keskkonnas.

Olenemata OpenShift paigaldusprogrammi interaktiivsusest on eesmärgiks kogu paigaldusprotsess automatiseerida kasutades selleks Ansible'it. Selleks loob autor Ansible mänguplaani (*playbook*) vastavalt eelmainitud paigaldusjuhenditele. Mänguplaan ja selle Ansible kood on välja toodud lisas 3. Ansible mänguplaan on korduvkasutatav ja võimaldab kiiret avariitaastet tagades igal korral samasuguse tulemuse.



Joonis 5. OpenShifti paigaldamise ülevaade [24].

OpenShifti paigaldusprogramm loob kolm kontrollsõlme ja kolm töösõlme, mille spetsifikatsioonid on märgitud Tabel 1. Loodavate kontrollsõlmede ja töösõlmede arvu on võimalik muuta Ansible mänguraamatu (vt lisa 3) põhifailis ***okd_install.yml*** muutujate (*vars*) sektsioonis. OpenShifti paigaldusprogramm loob esmalt ajutise alglaadurvirtuaalmasina, mis loob kontrollsõlmed, nendest üks omakorda seab üles kõik töösõlmed (vt Joonis 5).

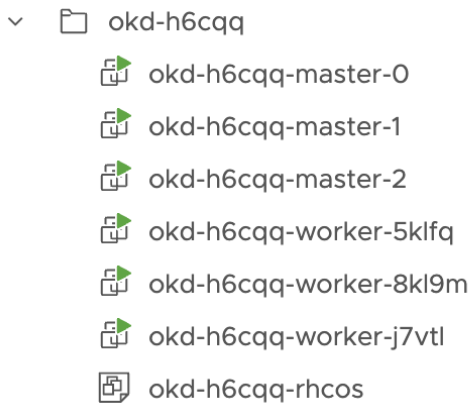
Tabel 1. Klastrisõlmede spetsifikatsioonid [70].

	Protsessor	**Mälu**	**Salvestuspind**
Kontrollsõlm	4 vCPU	16 GB	120 GB
Töösõlm	2 vCPU	8 GB	120 GB

Paigaldusprotsessi lõpus väljastab Ansible vastavalt mänguplaanile paigaldusprogrammi logifaili viimased 13 rida (ridade arvu valis autor selliselt, et see sisaldaks kogu olulist informatsiooni) klastrisse sisselogimiseks vajaliku infoga ja samuti kokkuvõtvalt klastri paigaldamise protsess kestvuse (vt Joonis 6).

Joonis 6. OpenShift paigaldusprotsessi lõppinfo.

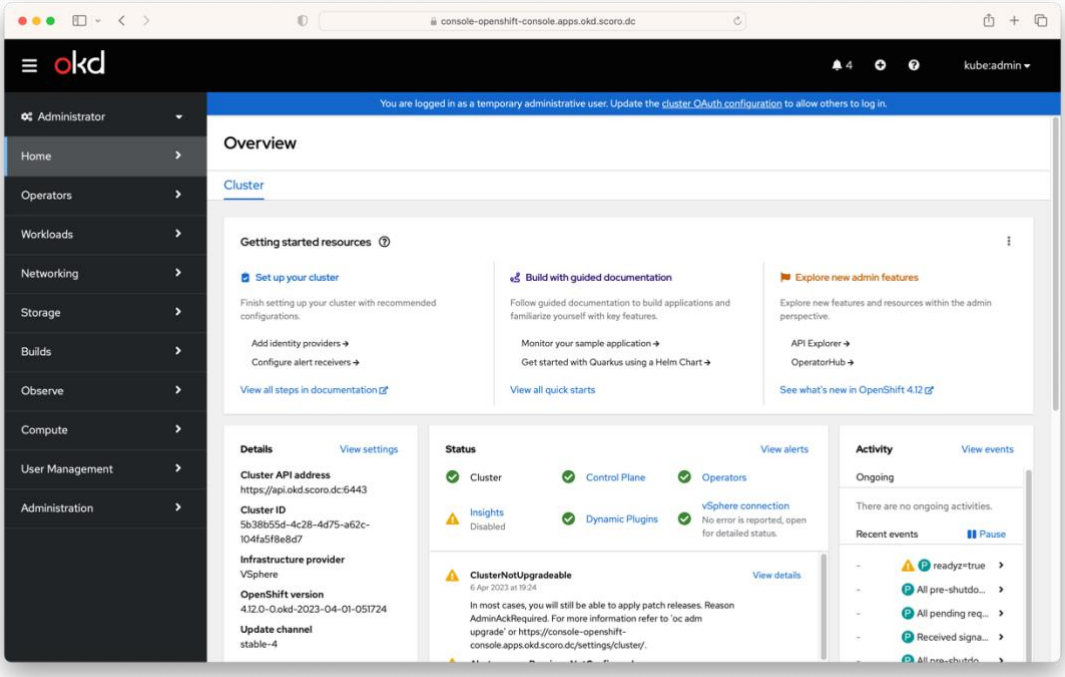
Paigaldusprotsessi lõppedes on VMware vSphere keskkonnas OpenShifti klaster kuue virtuaalmasinaga ja ühe malliga, millest kolm on kontrollmasinad ja kolm on töömasinad, ning mall on vajadusel hilisemaks töömasinate sätestamiseks (vt Joonis 7). Lisaks on paigaldusprotsessi lõpuks eemaldatud alglaadur, millest klastri loomine alguse sai (vt Joonis 5).



Joonis 7. OpenShift klaster vSpheres.

Lisaks on ka automaatselt paigaldatud OpenShift klastri haldamise hõlbustamiseks veebiliides, millele pääseb ligi väljastatud kasutajanime ja ajutise parooliga paigaldusprotsessi lõpus näidatud aadressil (vt Joonis 6). Graafiline veebiliides on kujutatud Joonis 8. Graafiline kasutajaliides on üks kahest OpenShift klastri haldamise liidesest, mis pakub eraldi vaadet ja võimalusi administraatorile ja arendajale [71].

Veebiliides võimaldab saada kõige muu seas visuaalset informatsiooni rakenduste, nende konteinerite ja nende uuenduste ja ka klastri virtuaalmasinate uuendustest [72], mis teeb nii rakenduste kui ka klastri haldamise lihtsamaks vähendades käsurea käskude õppimise vajadust [64]. Üheks veebiliidese klastri haldamist hõlbustavaks teguriks on otse brauserist objekte kirjeldavate YAML failide lisamise ja muutmise võimalus.

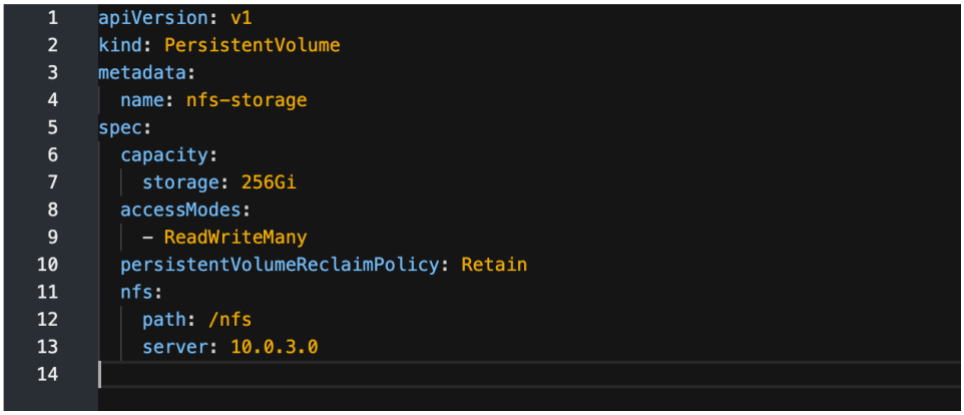


Joonis 8: Graafiline OpenShifti veebiliides.

5.3 Seadistamine

5.3.1 Püsisalvestuspind

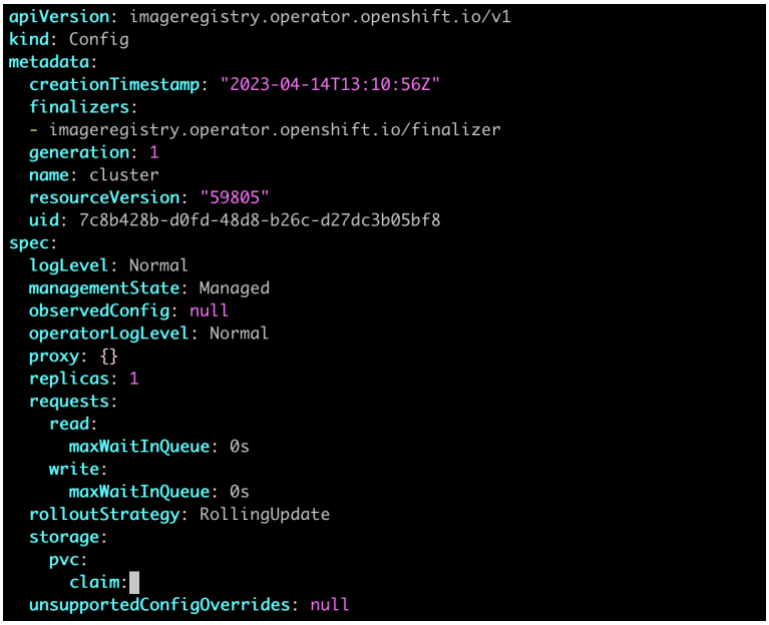
Ühe esimese seadistusammuna tuleks värskelt loodud klastrile lisada püsisalvestuspind (edaspidi PV), mis teeb võimalikuks kasutada OpenShifti sisseehitatud privaatset konteineritõmmiste registrit, kuna VMware vSphere on üks OpenShifti toetatud platvormidest, mis ei paku automaatselt jagatavat objektsalvestuspinda [69]. Autor on eelnevalt seadistanud salvestuspinnaks NFS serveri, mille peab lisama klastrisse kui püsisalvestuspinna, seda on kujutatud Joonis 9. Privaatne konteineritõmmiste register võimaldab kasutada OpenShifti koodist rakenduse kompileerimise funktsionaalsust ja loob ka salvestuspinna konteineritõmmiste salvestamiseks.



Joonis 9. Püsisalvestuspinna lisamine OpenShift klastrisse.

5.3.2 Konteineritõmmiste register

Lisaks PV klastrile lisamisele on vaja vastava registri operaatori konfiguratsioonis muuta *managementState Removed* oleksust *Managed* olekusse [69]. Seda on võimalik teha nii käsurealt kui ka veebiliidesest, selle muudatus on kujutatud Joonis 10. Peale registri oleku muutmist tuleb lisada konfiguratsioonivälja püsisalvestuspinna taotlus (*persistent volume claim*). Jättes *claim* välja tühjaks luuakse püsisalvestuspinna taotlus (edaspidi PVC) automaatselt, samas aga on ka võimalik PVC eelnevalt käsitsi luua ja selle nimetuse panna *claim* väljale.



Joonis 10. Konteineritõmmiste registri konfiguratsioon.

Seejärel kui register on lisatud ja sellele PVC määratud, on kasulik kontrollida registri staatust. Staatuse kontrolli ja selle tulemust on kujutatud Joonis 11.

```
maido@mp:~$ oc get clusteroperator image-registry
NAME             VERSION                         AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
image-registry   4.12.0-0.okd-2023-04-01-051724   True        False         False      29m
maido@mp:~$
```

Joonis 11. Konteineritõmmiste registri staatuse kontroll.

5.3.3 Sisehaldusportaali paigaldamine

Järgmise seadistussammuna loob autor klastrisse projektid kolme erineva paigalduse jaoks vastavalt versioonihalduses loodud töö-, proovi- ja arendusharule (*production*, *staging*, *development*). Need hoiavad koos kõiki erinevad sisehaldusportaali komponente vastavalt lähtekoodi harule ja lubavad neil komponentidel omavahel suhelda ja andmeid vahetada. OpenShifti projekt on oma olemuselt samasugune nagu Kubernetese nimeruum (*namespace*), millel on lisaks veel annotatsioonid [25]. Luues OpenShifti projekti luuakse ka automaatselt Kubernetese nimeruum [64].

Loodud nimeruumi saladuste hoidlasse lisab autor kaks saladust (*secret*): ühe, mis sisaldab GitLabi pääsumandaati (*access credentials*), ja teise, mis sisaldab sisahldusportaali toimimiseks vajalikke keskkonnamuutujaid (*environment variables*) nagu näiteks andmebaasi kasutajanimed ja paroolid.

Rakenduse paigaldamiseks OpenShifti ja Kubernetese keskkonnas tuleb rakenduse konteinerid paigaldada (*deployment*) ja seejärel need ka välismaailmale kättesaadavaks teha (*service* ja *ingress/route*). Selleks on kasutusel objektid, mis defineerivad detailid, kuidas miski töötama peaks. [73]

Kuigi Kubernetese objekte on võimalik hallata nii imperatiivselt (käsurealt) kui ka deklaratiivselt (konfiguratsioonifailide kaudu), on mõlemal puhul olemas kindlad kasutusjuhud. Näiteks on imperatiivne objektide haldamine kiiremini õpitav, kuid ebasoovitatav kasutada väljaspool arendusjärgus olevaid projekte, samuti ei ole imperatiivsel haldusmeetodil muutuste ajalugu. [74]

Deklaratiivselt rakenduste paigaldamiseks OpenShifti ja Kubernetese keskkonda on vaja mitmeid konfiguratsioonifaile, mille haldamine isegi ühe lihtsa rakenduse piires võib olla

keerukas, rakenduse suurenedes kasvab see keerukus eksponentsiaalselt. Seda keerukust aitab vähendada Helm [75], mis on paketihaldur (*package manager*) Kubernetesele. Helm on Kuberneteses ametlikult toetatud ja on samuti CNCF'i arendatud nagu Kubernetes isegi. [76]

Helm paketid (*Helm charts*) [77] on kogum kõikidest konfiguratsioonifailidest, mida on vaja rakenduse Kubernetese (k.a OpenShift) klastrisse paigaldamiseks ning koosneb ***Chart.yaml***, ***values.yaml*** failidest, mis vastavalt on metaandmete ja vaikimisi konfiguratsiooniväärtused ja muutujad paketile, ja ***templates/*** kataloogist, mis sisaldab Kubernetese objektide konfiguratsioonifaile [78]. Lisas 4 on välja toodud autori loodud Helm paketi failid sisehaldusportaali ühele komponendile koos paketi ülesehitusega.

Helm võimaldab lihtsasti ühe käsuga paigaldada, uuendada ja eemaldada rakendusi [79], mis koosnevad mitmetest konfiguratsioonifailidest abstraheerides sellega Kubernetese klastri kontrollsõlmedega suhtlemist [78].

Lisas 4 esitatud Helm paketi failid on salvestatud GitLabis samasse repositooriumisse, kus asub sisehaldusportaali vastava komponendi lähtekood ja ***Dockerfile*** konteineritõmmise loomiseks.

5.3.4 CI/CD konveier

Kuigi Helm võimaldab lihtsasti rakenduste klastrisse paigaldamist, ei ole manuaalselt paigalduskäsu kasutamine parim lahendus. Selle automatiseerimiseks seadistab autor igas GitLabi repositooriumis sidusarenduse konveieri, mille ülesandeks on rakendust kontrollida, see pakendada ning OpenShifti klastrisse paigaldada [11].

Lisas 5 on esitatud GitLabi CI/CD konveieri fail, mis defineerib kolm konveieri etappi rakenduse Dockeri konteineritõmmise loomiseks, selle testimiseks ning eduka testi puhul ka OpenShifti klastrisse paigaldamiseks. Sidusarenduse konveieri etapid on kujutatud Joonis 2.

Vastavad sidusarenduse konveieri failid on iga sisehaldusportaali komponendi repositooriumis ning käivituvad automaatselt kui vastavas koodiharus (*branch*) toimub muutus (koodi lisamine, muutmine jms). Kuna erinevate rakenduse komponentide puhul on konteineritõmmise valmistamise protsess suuremas osas samasugune, siis protsessi lühendamiseks ning kiirendamiseks ja dubleeriva koodi vähendamiseks valmistas autor

ühisosa jaoks eraldi konteineritõmmise, mis on iga konveieri käivitamiseks valmis ja kasutatav ning salvestatud konteineitõmmiste registrisse. Samamoodi valmistas autor ka konteineritõmmise rakenduse paigaldamiseks, kuna iga käivituse korral uuesti vajaliku tarkvara (nt OpenShifti klientprogramm) installeerimine oleks koodi liigne dubleerimine ja ka ressursside ebaotstarbekas kasutamine.

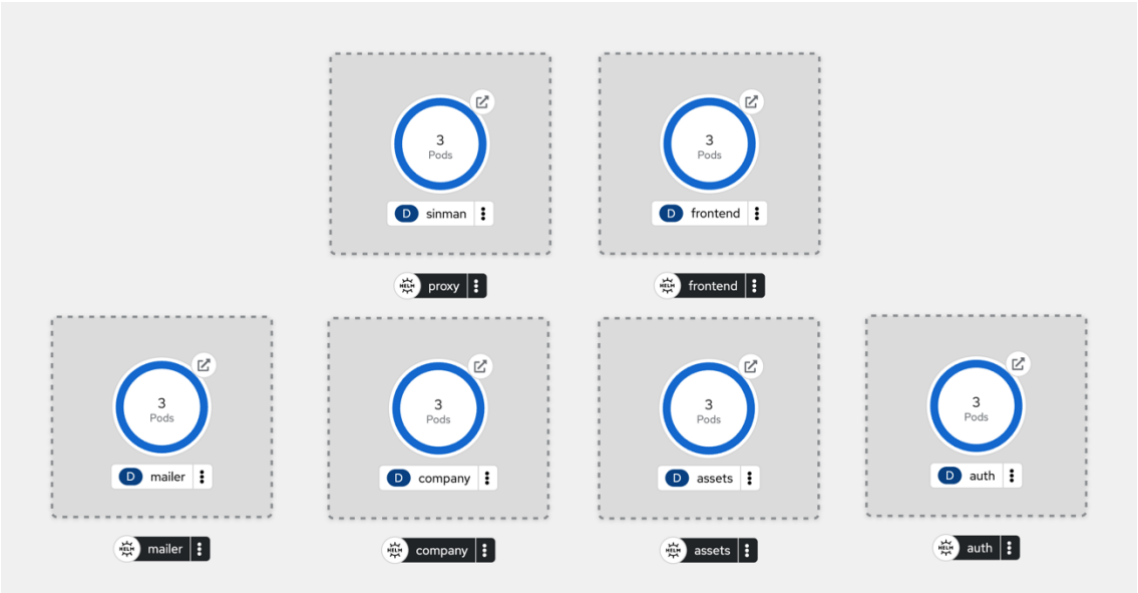
Loodud sidusarenduse konveier valmistab vastavalt repositooriumile rakendusest uue konteineritõmmise, mis salvestatakse GitLabi konteineritõmmiste registrisse. Järgnev testimise etapp võtab äsja valmistatud konteineritõmmise ja installeerib sellesse rakenduse testimiseks vajaliku tarkvara ning käivitab arendajate poolt loodud testid. Testide eduka läbimise tulemusena käivitub rakenduse paigaldamise etapp, mis ühendub OpenShifti klastrisse ning paigaldab või uuendab Helm paketihalduri abil rakenduse või selle komponendi. Selle tulemusena omakorda vahetab OpenShift vanad konteinergrupid (*pods*) välja uute vastu järkjärguliselt ilma teenuse või rakenduse katkestuseta [80]. Konteinergruppide järkjärgulist uuendamist on kujutatud Joonis 12, millel vasakpoolne kujutab vana koodiga konteinergruppe ja parempoolne uuenenud koodiga konteinergruppe.

Joonis 12. Konteinergruppide uuendamine.

6 Tulemused

Tulemusena on valminud OpenShiftil põhinev kolmest kontrollsõlmest ja kolmest töösõlmest koosnev klaster, mis on loodav ja paigaldatav IaaC põhimõtteid silmas pidades Ansible mänguraamatu abil ning mis seetõttu on automatiseeritud vähendades võimalikke eksimusi, mis võivad tekkida käsitsi paigalduse käigus. Samuti hoiab automatiseeritud paigaldusprotsess kokku ettevõtte ressursse nii ajalises kui ka rahalises väärtuses. Lisaks on OpenShifti klaster integreeritud ettevõttes kasutusel oleva versioonihaldustarkvaraga GitLab. Autori seadistatud integratsioon automatiseerib sisehaldusportaali arendusprotsessi uuendades rakendust järkjärguliselt, ilma katkestuseta kasutajale, automaatselt iga koodi uuendamise või muutmisega arendajate poolt.

Rakenduse paigaldus on seadistatud hoidma igast komponendist kasutusel kolme konteinerigruppi, mille OpenShift automaatselt jagab ära klastri sõlmede vahel tagades sellega rakenduse veakindluse (vt Joonis 13). Tänu OpenShifti toetumine Kubernetesele töötab rakenduse liikluse koormusjaotus konteinergruppide vahel samadel põhimõtetel nagu Kuberneteseski.



Joonis 13. Sisehaldusportaali paigaldus.

GitLab alustab automaatset testimise ja paigaldusprotsessi koheselt kui vastavasse eeldefineeritud koodiharusse on tehtud täiendus või muutus. Paigaldusprotsess võtab hetkel aega kaks kuni neli minutit, olenevalt rakenduse komponendi keerukusest ja kasutatavatest lisadest. Hetkel eksisteerivas seadistuses on olemas kolm eraldi rakenduse klastrit – üks arendamiseks (*development*), milles luuakse uued rakenduse konteinerigrupid kui GitLabis lisatakse kood arenduse harusse; teine mis on kasutusse mineva koodi testimiseks (*staging*) ning kolmas, mis on valmis versioonihalduse põhiharu ja sisaldab kasutajatele ligipääsetava rakenduse koodi (*master*).

Lisaks toimivale virtuaalsele taristule dokumenteeris autor kogu protsessi ettevõttesiseselt võimaldades vajadusel saada põhjaliku ülevaate paigaldusprotsessist ja samuti selle toimimisest. See võimaldab samuti kaastöötajatel teha vajalikke muudatusi ja arendada vajadusel taristut edasi.

7 Järeldused ja edasised tegevused

Kuigi OpensShiftil on Kubernetese ees mitmeid eeliseid, sealhulgas lihtsustatud paigaldusprotsess ja klastri haldamine, on OpenShifti dokumentatsioon kohati keerulisem ja tehnilisem. Seda ilmselt suuresti sellepärast, et Kubernetesel on suurem toetajaskond erinevatel oskustasemetel ning seetõttu on saadaval ka hulgaliselt juhendeid erinevatele oskustasemetele. Tänu sellele on Kubernetese puhul abi leidmine kiirem ja lihtsam kui OpenShiftil. Samas on OpenShifti tugevuseks just Kubernetese moontori kasutamine, mis tähendab, et saadaval olevad Kubernetese abimaterjalid on rakendatavad ka OpenShifti puhul.

Autori arvates on OpenShift siiski parim võimalik lahendus käesolevas töös püstitatud küsimustele „Milline lahendus sobib tarkvarateenust osutava ettevõtte virtuaalse sisemajutatud sisetaristu loomiseks?“ vastamiseks. OpenShift pakub ettevõttesisese haldusportaali arendamiseks ja majutamiseks mugavamat, lihtsasti hallatavat keskkonda ja on terviklahendus arendustsükli juurutamiseks.

Edasiste tegevustena on võimalik paigaldatud taristut optimeerida muutes selle protsesse kiiremaks ning turvalisemaks. Näiteks oleks ühe edasise tegevusena võimalik uurida paremat lahendust saladuste hoiustamiseks ning automaatseks muutmiseks kui seda on OpenShifti sisseehitatud saladuste hoidla. Samuti oleks võimalik täiendada autori loodud taristut täielikult automatiseerides, eemaldades praegused manuaalsed tegevused nagu näiteks saladuste lisamine ja projekti loomine klastrisse.

8 Kokkuvõte

Käesolev bakalaureusetöö uurib tarkvarateenust pakkuva ettevõtte näitel IT-sisetaristu võimalikke lahendusi pidades silmas vajadust loodud virtuaalset taristut sisemajutada. Töös keskendub autor kuuele enamlevinud automatiseeritud konteinerihalduse lahendusele analüüsides nende erinevusi ja sobivust ettevõtte vajadustele.

Töö tulemusena leidis autor, et võimalikest analüüsitud lahendustest on sobivaim ettevõtte virtuaalse sisetaristu loomiseks Red Hat OpenShifti ülesvoolu avatud lähtekoodiga ja vabavaraline OKD, mida Red Hat arendab koos vabatahtliku toetajaskonnaga. OKD paistab silma, võrreldes teiste automatiseeritud konteinerihalduse tööriistadega, vaikimisi turvalahenduste ja Kubernetesga integreerituse ning lihtsa kasutuselevõtu poolest.

Autor kirjeldab OKD paigaldusprotsessi, seadistust ja ettevalmistust ettevõtte siseselt arendatava sisehaldusportaali jaoks pidades silmas sidusarenduse põhimõtteid ja levinud standardeid. Lisaks kirjeldab autor sidusarenduse konveieri seadistust ja sisehaldusportaali paigalduse automatiseerimist.

Töö tulemused pakuvad asjakohast teavet skaleeruva virtuaalse sisemajutatud taristu loomiseks ning annab praktilisi soovitusi OpenShift (OKD) paigaldamiseks, mis aitab kiiresti ja tõhusalt ette valmistada sobiv sidusarenduse ja siseteenuste majutuskeskkond.

Kasutatud kirjandus

[1] „Pandemic has accelerated cloud transformation for nearly half of organizations“, *Security Magazine*, 19. oktoober 2020. https://www.securitymagazine.com/articles/93654-pandemic-has-accelerated-cloud-transformation-for-nearly-half-of-organizations (vaadatud 24. märts 2023).

[2] F. Nata, „Software-as-a-Service (SaaS) VS In-House Software?“, *Ritase*, 18. mai 2021. https://medium.com/ritase/software-as-a-service-saas-vs-in-house-software-80adbdf63dcc (vaadatud 24. märts 2023).

[3] P. Haefele, „EC2 is 380% more expensive than internal cluster“, 20. oktoober 2012. http://deepvalue.net/ec2-is-380-more-expensive-than-internal-cluster/ (vaadatud 24. märts 2023).

[4] D. Molnar ja S. E. Schechter, „Self Hosting vs. Cloud Hosting: Accounting for the Security Impact of Hosting in the Cloud.“, esitatud WEIS, 2010, lk 1–18.

[5] „Intel® NUC Products“, *Intel*. https://www.intel.com/content/www/us/en/products/details/nuc.html (vaadatud 24. aprill 2023).

[6] M. Assumma, „What can you do with an Intel NUC?“, *Newegg Insider*, 28. veebruar 2019. https://www.newegg.com/insider/what-can-you-do-with-an-intel-nuc/ (vaadatud 24. märts 2023).

[7] „HPE ProLiant DL360 Gen10 Plus server“, *PSNow*. https://www.hpe.com/psnow/doc/a50002559enw (vaadatud 7. aprill 2023).

[8] „VMware vSphere | Enterprise Workload Platform“, *VMware*. https://www.vmware.com/products/vsphere.html (vaadatud 7. aprill 2023).

[9] „Top virtualization technologies market share 2022“, *Statista*, 21. november 2022. https://www.statista.com/statistics/1252355/top-virtualization-technologies-by-domain/ (vaadatud 24. aprill 2023).

[10] „The DevSecOps Platform“. https://about.gitlab.com/ (vaadatud 16. aprill 2023).

[11] C. Cowell, N. Lotz, ja C. Timberlake, *Automating DevOps with GitLab CI/CD Pipelines*. Packt Publishing, 2023.

[12] „Docker: Accelerated, Containerized Application Development“, *Docker*, 10. mai 2022. https://www.docker.com/ (vaadatud 7. aprill 2023).

[13] „Stack Overflow Developer Survey 2022“, *Stack Overflow*, 2022. https://survey.stackoverflow.co/2022/?utm_source=social-share&utm_medium=social&utm_campaign=dev-survey-2022 (vaadatud 25. märts 2023).

[14] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe, ja F. Khendek, „Microservice Based Architecture: Towards High-Availability for Stateful Applications with Kubernetes“, *2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)*, juuli 2019, lk 176–185. doi: 10.1109/QRS.2019.00034.

[15] A. Khan, „Key Characteristics of a Container Orchestration Platform to Enable a Modern Application“, *IEEE Cloud Computing*, kd 4, nr 5, lk 42–48, sept 2017, doi: 10.1109/MCC.2017.4250933.

[16] L. A. Vayghan, M. A. Saied, M. Toeroe, ja F. Khendek, „Kubernetes as an Availability Manager for Microservice Applications“. arXiv, 15. jaanuar 2019. doi: 10.48550/arXiv.1901.04946.
[17] A. Malviya ja R. K. Dwivedi, „A Comparative Analysis of Container Orchestration Tools in Cloud Computing“, *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, märts 2022, lk 698–703. doi: 10.23919/INDIACom54597.2022.9763171.
[18] C. Singh, N. S. Gaba, M. Kaur, ja B. Kaur, „Comparison of Different CI/CD Tools Integrated with Cloud Platform“, *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, jaan 2019, lk 7–12. doi: 10.1109/CONFLUENCE.2019.8776985.
[19] J. Fairbanks, A. Tharigonda, ja N. U. Eisty, „Analyzing the Effects of CI/CD on Open Source Repositories in GitHub and GitLab“. arXiv, 28. märts 2023. Vaadatud: 16. aprill 2023. [Online]. Available at: http://arxiv.org/abs/2303.16393
[20] E. Löfström, „Tegevusuuringu käsiraamat“, *Tallinn: Archimedes*, 2011.
[21] R. O'Brien, „An Overview of the Methodological Approach of Action Research“, 2008. Vaadatud: 25. märts 2023. [Online]. Available at: https://www.semanticscholar.org/paper/An-Overview-of-the-Methodological-Approach-of-O%27Brien/d9664350e9792dbae3fe4842a9459ed1ecce3cab
[22] P. J. M. Costello, *Effective Action Research: Developing Reflective Thinking and Practice*. London, UNITED KINGDOM: Bloomsbury Publishing Plc, 2011. Vaadatud: 1. aprill 2023. [Online]. Available at: http://ebookcentral.proquest.com/lib/tuee/detail.action?docID=655497
[23] E. Truyen, D. Van Landuyt, D. Preuveneers, B. Lagaisse, ja W. Joosen, „A Comprehensive Feature Comparison Study of Open-Source Container Orchestration Frameworks“, *Applied Sciences*, kd 9, nr 5, Art. nr 5, jaan 2019, doi: 10.3390/app9050931.
[24] „Installation overview | Installing | OKD 4“, *OKD.io*. https://docs.okd.io/latest/installing/index.html (vaadatud 2. aprill 2023).
[25] Manuel Dewald ja Rick Rackow, *Operating OpenShift*. O'Reilly Media, Inc, 2022.
[26] „Production-Grade Container Orchestration“, *Kubernetes*. https://kubernetes.io/ (vaadatud 7. aprill 2023).
[27] „Nomad“. HashiCorp. Vaadatud: 25. märts 2023. [Online]. Available at: https://github.com/hashicorp/nomad
[28] „Red Hat OpenShift enterprise Kubernetes container platform“. https://www.redhat.com/en/technologies/cloud-computing/openshift (vaadatud 7. aprill 2023).
[29] „Apache Mesos“, *Apache Mesos*. https://mesos.apache.org/ (vaadatud 7. aprill 2023).
[30] „Swarm mode overview“, *Docker Documentation*, 24. märts 2023. https://docs.docker.com/engine/swarm/ (vaadatud 26. märts 2023).
[31] „Rancher by SUSE“, *Rancher Labs*. http://www.rancher.com (vaadatud 7. mai 2023).
[32] D. Bernstein, „Containers and Cloud: From LXC to Docker to Kubernetes“, *IEEE Cloud Computing*, kd 1, nr 3, lk 81–84, sept 2014, doi: 10.1109/MCC.2014.51.
[33] A. Sharma, „The History of Kubernetes“, *DEV Community*, 31. juuli 2022. https://dev.to/iarchitsharma/the-history-of-kubernetes-4kkd (vaadatud 25. märts 2023).

[34] „9 insights on real world container use“, *Datadog*, november 2022. https://www.datadoghq.com/container-report/ (vaadatud 25. märts 2023).
[35] B. Pariseau, „HashiCorp Nomad vs. Kubernetes matchup intensifies with 0.11“, *TechTarget*. https://www.techtarget.com/searchitoperations/news/252480628/HashiCorp-Nomad-vs-Kubernetes-matchup-intensifies-with-011 (vaadatud 25. märts 2023).
[36] „Non-Containerized Orchestration“, *Nomad by HashiCorp*. https://www.nomadproject.io/use-cases/non-containerized-application-orchestration (vaadatud 26. märts 2023).
[37] T. Lefebvre, „How we use HashiCorp Nomad“, *The Cloudflare Blog*, 5. juuni 2020. http://blog.cloudflare.com/how-we-use-hashicorp-nomad/ (vaadatud 26. märts 2023).
[38] J. Fernandes, „Why Red Hat Chose Kubernetes for OpenShift“, *Red Hat Hybrid Cloud*, 7. november 2016. https://content.cloud.redhat.com/blog/red-hat-chose-kubernetes-openshift (vaadatud 25. märts 2023).
[39] D. Mueller, „OKD: Renaming of OpenShift Origin with 3.10 Release“, *Red Hat Hybrid Cloud*, 3. august 2018. https://content.cloud.redhat.com/blog/okd310release (vaadatud 26. märts 2023).
[40] „About OKD“, *OKD.io*. https://www.okd.io/about/ (vaadatud 26. märts 2023).
[41] „Red Hat OpenShift vs. OKD“, *Red Hat*, 16. november 2022. https://www.redhat.com/en/topics/containers/red-hat-openshift-okd (vaadatud 25. märts 2023).
[42] A. Sharif, „Kubernetes vs. Mesos“, *crowdstrike.com*, 21. detsember 2022. https://www.crowdstrike.com/cybersecurity-101/observability/kubernetes-vs-mesos/ (vaadatud 25. märts 2023).
[43] „Marathon: A container orchestration platform for Mesos and DC/OS“. https://mesosphere.github.io/marathon/ (vaadatud 25. märts 2023).
[44] F. Soppelsa ja C. Kaewkasi, *Native Docker Clustering with Swarm: Deploy, Configure, and Run Clusters of Docker Containers with Swarm*. Birmingham, UNITED KINGDOM: Packt Publishing, Limited, 2016. Vaadatud: 2. aprill 2023. [Online]. Available at: http://ebookcentral.proquest.com/lib/tuee/detail.action?docID=4773124
[45] „Understanding Difference between Docker Swarm(Classic), Swarm Mode & SwarmKit“, *Dockerlabs*. https://dockerlabs.collabnix.com/intermediate/swarm/difference-between-docker-swarm-vs-swarm-mode-vs-swarmkit.html (vaadatud 2. aprill 2023).
[46] M. Mattox, *Rancher Deep Dive*. Packt Publishing, 2022.
[47] „SUSE - Open Source Solutions for Enterprise Servers & Cloud | SUSE“. https://www.suse.com/ (vaadatud 7. mai 2023).
[48] „Rancher vs. Openshift: The Guide“, *Densify*. https://www.densify.com/openshift-tutorial/rancher-vs-openshift/ (vaadatud 7. mai 2023).
[49] B. Reselman, „Kubernetes vs. Rancher: The differences all devs should know“, *TheServerSide.com*, 23. aprill 2021. https://www.theserverside.com/answer/Kubernetes-vs-Rancher-The-differences-all-devs-should-know (vaadatud 26. märts 2023).
[50] L. Dillon, „Rancher vs. OpenShift: Platform and Feature Comparison“, *OpenLogic by Perforce*, 11. märts 2021. https://www.openlogic.com/blog/rancher-vs-openshift (vaadatud 7. mai 2023).

[51] „HashiCorp Nomad“, *Traefik Labs*, 5. august 2022. https://traefik.io/glossary/hashicorp-nomad-101/ (vaadatud 26. märts 2023).
[52] A. Gamela, „Nomad vs. Kubernetes: container orchestration tools compared“, *Imaginary Cloud*, 11. november 2021. https://www.imaginarycloud.com/blog/nomad-vs-kubernetes/ (vaadatud 26. märts 2023).
[53] „Nomad Tools“, *HashiCorp Developer*. https://developer.hashicorp.com/nomad/tools (vaadatud 26. märts 2023).
[54] „A Curated List of Kubernetes Tools“, *Kubetools*. https://collabnix.github.io/kubetools/ (vaadatud 26. märts 2023).
[55] „OpenShift vs. Kubernetes: What’s the Difference?“, *IBM Cloud Education*, 16. september 2021. https://www.ibm.com/cloud/blog/openshift-vs-kubernetes (vaadatud 26. märts 2023).
[56] A. Lossent, A. R. Peon, ja A. Wagner, „PaaS for web applications with OpenShift Origin“, *J. Phys.: Conf. Ser.*, kd 898, nr 8, lk 082037, okt 2017, doi: 10.1088/1742-6596/898/8/082037.
[57] B. Hindman *et al.*, „Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center“, *NSDI*, kd 11, lk 22–22, jaan 2011.
[58] M. Moravcik ja M. Kontsek, „Overview of Docker container orchestration tools“, *2020 18th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, nov 2020, lk 475–480. doi: 10.1109/ICETA51985.2020.9379236.
[59] Y. Pan, I. Chen, F. Brasileiro, G. Jayaputera, ja R. Sinnott, „A Performance Comparison of Cloud-Based Container Orchestration Tools“, *2019 IEEE International Conference on Big Knowledge (ICBK)*, nov 2019, lk 191–198. doi: 10.1109/ICBK.2019.00033.
[60] I. M. A. Jawarneh *et al.*, „Container Orchestration Engines: A Thorough Functional and Performance Comparison“, *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, mai 2019, lk 1–6. doi: 10.1109/ICC.2019.8762053.
[61] C. Rosen, „Docker Swarm vs. Kubernetes: A Comparison“, *IBM Blog*, 7. juuni 2022. https://www.ibm.com/blog/docker-swarm-vs-kubernetes-a-comparison/ (vaadatud 2. aprill 2023).
[62] „Fedora CoreOS Documentation“, *Fedora Docs*. https://docs.fedoraproject.org/en-US/fedora-coreos/ (vaadatud 7. aprill 2023).
[63] B. Gilbert, „Introducing Fedora CoreOS“, *Fedora Magazine*, 24. juuli 2019. https://fedoramagazine.org/introducing-fedora-coreos/ (vaadatud 7. aprill 2023).
[64] B. Reselman, „A developer’s guide to using OpenShift with Kubernetes“, *Red Hat Developer*, 11. jaanuar 2023. https://developers.redhat.com/articles/2023/01/11/developers-guide-using-openshift-kubernetes (vaadatud 2. aprill 2023).
[65] „GitLab Documentation“, *GitLab*. https://docs.gitlab.com/ (vaadatud 16. aprill 2023).
[66] I. Kumara *et al.*, „The do’s and don’ts of infrastructure code: A systematic gray literature review“, *Information and Software Technology*, kd 137, lk 106593, sept 2021, doi: 10.1016/j.infsof.2021.106593.
[67] „Ansible is Simple IT Automation“, *Ansible*. https://www.ansible.com (vaadatud 7. aprill 2023).
[68] Mi. Guerriero, M. Garriga, D. A. Tamburri, ja F. Palomba, „Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry“, *2019 IEEE*

International Conference on Software Maintenance and Evolution (ICSME), sept 2019, lk 580–589. doi: 10.1109/ICSME.2019.00092.
[69] „Installing a cluster on vSphere - Installing on vSphere“, *OKD.io*. https://docs.okd.io/latest/installing/installing_vsphere/installing-vsphere-installer-provisioned.html#installing-vsphere-installer-provisioned (vaadatud 14. aprill 2023).
[70] „vSphere IPI Deployment - OKD.io“. https://www.okd.io/guides/vsphere-ipi/ (vaadatud 7. aprill 2023).
[71] M. Abushamleh, „OpenShift 101: Web console and CLI“, *IBM Developer*, 8. september 2020. https://developer.ibm.com/blogs/openshift-101-web-console-and-cli/developer.ibm.com/blogs/openshift-101-web-console-and-cli (vaadatud 7. aprill 2023).
[72] „Chapter 1. Web Console Overview OpenShift Container Platform 4.10 | Red Hat Customer Portal“. https://access.redhat.com/documentation/enus/openshift_container_platform/4.10/html/web_console/web-console-overview (vaadatud 7. aprill 2023).
[73] B. Krebs, „Kubernetes Tutorial - Step by Step Guide to Basic Kubernetes Concepts“, *Auth0 - Blog*, 23. aprill 2019. https://auth0.com/blog/kubernetes-tutorial-step-by-step-introduction-to-basic-concepts/ (vaadatud 23. aprill 2023).
[74] „Kubernetes Object Management“, *Kubernetes*. https://kubernetes.io/docs/concepts/overview/working-with-objects/object-management/ (vaadatud 23. aprill 2023).
[75] „Helm“. https://helm.sh/ (vaadatud 23. aprill 2023).
[76] B. Boucheron, „An Introduction to Helm, the Package Manager for Kubernetes | DigitalOcean“, *DigitalOcean*, 6. august 2018. https://www.digitalocean.com/community/tutorials/an-introduction-to-helm-the-package-manager-for-kubernetes (vaadatud 23. aprill 2023).
[77] „Helm Charts“, *Helm*. https://helm.sh/docs/topics/charts/ (vaadatud 23. aprill 2023).
[78] A. Zerouali, R. Opdebeeck, ja C. De Roover, „Replication package for the Helm charts empirical study“. Zenodo, 19. jaanuar 2023. doi: 10.5281/ZENODO.7552697.
[79] „Using Helm“. https://helm.sh/docs/intro/using_helm/ (vaadatud 23. aprill 2023).
[80] „Using deployment strategies“, *OKD.io*. https://docs.okd.io/latest/applications/deployments/deployment-strategies.html (vaadatud 14. aprill 2023).
[81] S. Hoque, M. S. De Brito, A. Willner, O. Keil, ja T. Magedanz, „Towards Container Orchestration in Fog Computing Infrastructures“, *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, juuli 2017, lk 294–299. doi: 10.1109/COMPSAC.2017.248.
[82] R. Buyya, M. A. Rodriguez, A. N. Toosi, ja J. Park, „Cost-Efficient Orchestration of Containers in Clouds: A Vision, Architectural Elements, and Future Directions“, *J. Phys.: Conf. Ser.*, kd 1108, nr 1, lk 012001, nov 2018, doi: 10.1088/1742-6596/1108/1/012001.
[83] B. Wilson, „16 Best Container Orchestration Tools And Services In 2023“, *DevopsCube*, 5. jaanuar 2022. https://devopscube.com/docker-container-clustering-tools/ (vaadatud 25. märts 2023).

[84] „15 Essential Container Orchestration Tools For 2023“, 26. august 2022. https://www.cloudzero.com/blog/container-orchestration-tools (vaadatud 7. mai 2023).
[85] A. Ali, „14 Container Orchestration Tools for DevOps“, *Geekflare*, 31. jaanuar 2023. https://geekflare.com/container-orchestration-software/ (vaadatud 8. mai 2023).

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

Mina, Maido Paalmäe

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Ettevõtte IT-sisetaristu arendus tarkvarateenust pakkuva ettevõtte näitel“, mille juhendajad on Kaido Kikkas ja Karl Hendrik Leppmets
   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.

15.05.2023

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 – Kontainerihalduse lahenduste populaarsuse analüüs

Allika pealkiri	**Kubernetes**	**Nomad**	**OpenShift**	**Mesos**	**Docker Swarm**	**Rancher**	**Portainer**	**Cloudify**
Overview of Docker container orchestration tools [58]	x		x	x	x			
Key Characteristics of a Container Orchestration Platform to Enable a Modern Application [15]	x	x		x	x	x		
Container Orchestration Engines: A Thorough Functional and Performance Comparison [60]	x			x	x	x		
A Comprehensive Feature Comparison Study of Open-Source Container Orchestration Frameworks [23]	x			x	x			
A Comparative Analysis of Container Orchestration Tools in Cloud Computing [17]	x		x	x	x			
Towards Container Orchestration in Fog Computing Infrastructures [81]	x			x	x			
Cost-Efficient Orchestration of Containers in Clouds: A Vision, Architectural Elements, and Future Directions [82]	x	x		x	x			
16 Best Container Orchestration Tools and Services [83]	x	x	x	x	x	x		
15 Essential Container Orchestration Tools For 2023 [84]	x	x	x	x	x	x	x	
14 Container Orchestration Tools for DevOps [85]	x	x	x	x	x	x		X
Kokku	**10**	**5**	**5**	**10**	**10**	**5**	**1**	**1**

Lisa 3 – OpenShifti paigaldamise Ansible mänguplaan

Ansible mänguplaani failid ja kataloogipuu.

```
/
|-- roles/
|   |-- vsphere/
|   |   |-- tasks/
|   |   |   |-- main.yml
|   |   |   |-- dhcp_vm.yml
|   |   |   |-- remove_vm.yml
|   |   |-- vars
|   |   |   |-- main.yml
|   |-- okd/
|   |   |-- files/
|   |   |   |-- ssh_key
|   |   |-- tasks/
|   |   |   |-- main.yml
|   |   |-- templates/
|   |   |   |-- install-config.yaml.j2
|   |   |-- vars/
|   |   |   |-- main.yml
|-- okd_install.yml
```

Ansible mänguplaani põhifail ***okd_install.yml,*** mis sätestab üldised klastri omadused sealhulgas klastri sõlmede arvud ja sisevõrgu IP aadressid.

```
---
- name: OKD Install
  hosts: localhost
  gather_facts: true
  vars:
    # Temporary VM details:
    datacentre: MCF-01
    cluster: TLN-01
    memory: 4096     # In MB
    disk_size: 32    # In GB.
    disk_type: thin
    datastore: H1-data
    cpus: 2
    vm_name: okd-install-temp
    vm_template: ubuntu2204
    vm_network: VSS Servers
    vm_folder: /okd
```

```
    # Network details:
    network_setup_type: static # Possible values: static or dhcp

    # Change the following only when selecting static IP setup:
    vm_ip: 10.0.3.198
    vm_netmask: 255.255.252.0

    # OKD Cluster details:
    admin_public_ssh_key: ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIDDUrbC+xgRqkeFom/Q1RPnpKSC+nR8YpA3SxrkdYw8c
    okd_api: 10.10.17.10
    okd_ingress: 10.10.17.11
    base_domain: scoro.dc
    cluster_name: okd
    workers: 3
    masters: 3

  roles:
    - vsphere

  tasks:
    - name: Install OKD
      ansible.builtin.include_role:
        name: okd
        apply:
          delegate_to: "{{ newvm.instance.ipv4 }}"

    - name: Include VM removal task from vSphere role
      ansible.builtin.include_role:
        name: vsphere
        tasks_from: remove_vm.yml
```

Vsphere rolli põhifail ***roles/vsphere/tasks/main.yml***, mis vastavalt eelmises failis valitud ajutise virtuaalmasina võrgu seadistuse valikule suunab edasi vastavale failile.

```
---
- name: Create a VM with DHCP network setup
  ansible.builtin.include_tasks:
    file: dhcp_vm.yml
  when: network_setup_type == "dhcp"

- name: Create a VM with DHCP network setup
  ansible.builtin.include_tasks:
    file: static_ip_vm.yml
  when: network_setup_type == "static"
```

Vsphere rolli virtuaalmasina loomise fail ***roles/vsphere/tasks/dhcp_vm.yml***, mis loob vSphere keskkonnas ajutise virtuaalmasina OpenShifti paigaldamiseks.

```
---
- name: Install required software to be able to communicate with VMware
  ansible.builtin.pip:
    name:
      - pyvmomi

- name: Provision VM from Template
  community.vmware.vmware_guest:
    hostname: "{{ vsphere_host }}"
    username: "{{ vsphere_user }}"
    password: "{{ vsphere_pass_secret }}"
    validate_certs: false
    datacenter: "{{ datacentre }}"
    folder: "{{ vm_folder }}"
    name: "{{ vm_name }}"
    template: "{{ vm_template }}"
    hardware:
      memory_mb: "{{ memory }}"
      num_cpus: "{{ cpus }}"
    disk:
      - size_gb: "{{ disk_size }}"
        type: "{{ disk_type }}"
        datastore: "{{ datastore }}"

    networks:
      - name: "{{ vm_network }}"
        type: dhcp
        start_connected: true
    state: poweredon
    wait_for_ip_address: true
  register: newvm
```

OKD paigalduse rolli põhifail ***roles/okd/tasks/main.yml***.

```
---
- name: Update apt cache if older than 24h
  ansible.builtin.apt:
    update_cache: true
    cache_valid_time: 86400
  become: true

- name: Install unzip to extract zip file
  ansible.builtin.apt:
    name: unzip
    state: present
  become: true

- name: Download OKD installer
  ansible.builtin.get_url:
    url: "{{ okd_installer }}"
    dest: "/home/ansible/okd_installer.tar.gz"
    checksum: "{{ okd_installer_checksum }}"
    mode: "0644"
  become_user: ansible

- name: Unpack OKD installer file
  ansible.builtin.unarchive:
    remote_src: true
    src: "/home/ansible/okd_installer.tar.gz"
    dest: "/home/ansible"
  become_user: ansible

- name: Generate SSH key for the installation
  community.crypto.openssh_keypair:
    path: /home/ansible/.ssh/temp_ssh_key
    type: ed25519
  register: temp_ssh_key

- name: Download vCenter root CA certificates
  ansible.builtin.get_url:
    url: "https://{{ vsphere_host }}/certs/download.zip"
    dest: "/home/ansible/certs.zip"
    mode: "0644"
    validate_certs: false

- name: Extract certificates from the downloaded Zip file
  ansible.builtin.unarchive:
    remote_src: true
    src: "/home/ansible/certs.zip"
    dest: "/usr/local/share/ca-certificates"
  become: true
```

```
- name: Update CA certificates
  ansible.builtin.command: update-ca-certificates
  become: true
  changed_when: true

- name: Make a folder for installation configuration files
  ansible.builtin.file:
    path: /home/ansible/install
    state: directory
    owner: ansible
    group: ansible
    mode: "0700"

- name: Copy installation configuration file
  ansible.builtin.template:
    src: install-config.yaml.j2
    dest: /home/ansible/install/install-config.yaml
    owner: ansible
    group: ansible
    mode: "0600"
  no_log: true

- name: Running OKD installer
  ansible.builtin.command: /home/ansible/openshift-install --dir=install
create cluster
  changed_when: true
  async: 3600
  poll: 0

- name: Wait until the install finishes (can take up to an hour)
  ansible.builtin.wait_for:
    path: /home/ansible/install/.openshift_install.log
    search_regex: Install complete!
    sleep: 30
    timeout: 3600

- name: Read last 13 lines from the installer log file
  ansible.builtin.command: tail -n 13
/home/ansible/install/.openshift_install.log
  changed_when: true
  register: logfile

- name: Display OKD installer final info
  ansible.builtin.debug:
    var: "{{ logfile.stdout_lines }}"
```

OpenShift klastri seadistusfaili mall ***roles/okd/templites/install-config.yaml.j2***, mis kasutades muutujaid mängukava põhifailist ***okd_install.yml***.

```
additionalTrustBundlePolicy: Proxyonly
apiVersion: v1
baseDomain: {{ base_domain }}
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform: {}
  replicas: {{ workers }}
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform: {}
  replicas: {{ masters }}
metadata:
  creationTimestamp: null
  name: {{ cluster_name }}
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 10.0.0.0/16
  networkType: OVNKubernetes
  serviceNetwork:
  - 172.30.0.0/16
platform:
  vsphere:
    apiVIPs:
    - {{ okd_api }}
    cluster: {{ cluster }}
    datacenter: {{ datacentre }}
    defaultDatastore: {{ datastore }}
    ingressVIPs:
    - {{ okd_ingress }}
    network: {{ vm_network }}
    password: {{ vsphere_pass_secret }}
    username: {{ vsphere_user }}
    vCenter: {{ vsphere_host }}
publish: External
pullSecret: '{"auths":{"fake":{"auth":"aWQ6cGFzcwo="}}}'
sshKey: |
  {{ admin_public_ssh_key }}
  {{ temp_ssh_key.public_key }}
```

Ajutise virtuaalmasina eemaldamise fail **/roles/vsphere/tasks/remove_vm.yml**, mis peale OpenShift klastri loomist kustutab paigaldamiseks loodud ajutise masina.

```
---
- name: Remove "{{ vm_name }}"
  community.vmware.vmware_guest:
    hostname: "{{ vsphere_host }}"
    username: "{{ vsphere_user }}"
    password: "{{ vsphere_pass_secret }}"
    validate_certs: false
    name: "{{ vm_name }}"
    state: absent
    force: true
```

Lisa 4 – Helm pakett

```
helm-chart/
|-- templates/
|   |-- deployment.yaml
|   |-- route.yaml
|   |-- service.yaml
|-- values-environments/
|   |-- values-dev.yaml
|   |-- values-staging.yaml
|-- Chart.yaml
|-- values.yaml
```

Chart.yaml:

```
apiVersion: v2
name: SinMan Auth
description: Helm chart for SinMan auth Service
version: 0.0.1
```

values.yaml:

```
app: auth
namespace: sinman
hostDomain: .apps.okd.scoro.dc
replicaCount: 3
containerPort: 80
servicePort: 80
image:
  repository: git.repository.address/scoro-internal/inman/auth
  tag: latest
secrets:
  pullSecret: mp-gitlab-image-pull
  envSecret: mp-sinman-environment
```

templates/deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ .Values.app }}
  namespace: {{ .Values.namespace }}
spec:
  replicas: {{ .Values.replicaCount }}
```

```
selector:
  matchLabels:
    app: {{ .Values.app }}
template:
  metadata:
    labels:
      app: {{ .Values.app }}
  spec:
    containers:
      - name: {{ .Values.app }}
        image: "{{ .Values.image.repository }}:{{ .Values.image.tag }}"
        ports:
          - containerPort: {{ .Values.containerPort }}
            protocol: TCP
        env:
          - name: DATABASE_URL
            valueFrom:
              secretKeyRef:
                name: {{ .Values.secrets.envSecret }}
                key: DATABASE_URL
          - name: REDIS_IP
            valueFrom:
              secretKeyRef:
                name: {{ .Values.secrets.envSecret }}
                key: REDIS_IP
          - name: REDIS_PASS
            valueFrom:
              secretKeyRef:
                name: {{ .Values.secrets.envSecret }}
                key: REDIS_PASS
    imagePullSecrets:
      - name: {{ .Values.secrets.pullSecret }}
    restartPolicy: Always
    terminationGracePeriodSeconds: 30
    serviceAccountName: "{{ .Values.namespace }}-sa"
    serviceAccount: "{{ .Values.namespace }}-sa"
strategy:
  type: RollingUpdate
  rollingUpdate:
    maxUnavailable: 25%
    maxSurge: 25%
revisionHistoryLimit: 10
progressDeadlineSeconds: 600
```

templates/route.yaml:

```
kind: Route
apiVersion: route.openshift.io/v1
metadata:
  name: {{ .Values.app }}
  namespace: {{ .Values.namespace }}
  labels:
    app: {{ .Values.app }}
spec:
  host: "{{ .Values.app }}{{ .Values.hostDomain }}"
  to:
    kind: Service
    name: {{ .Values.app }}
    weight: 100
  port:
    targetPort: "{{ .Values.servicePort }}-tcp"
  tls:
    termination: edge
    insecureEdgeTerminationPolicy: Redirect
status:
  ingress:
  - host: "{{ .Values.app }}{{ .Values.hostDomain }}"
    routerCanonicalHostname: router-default{{ .Values.hostDomain }}
    routerName: default
    wildcardPolicy: None
```

templates/service.yaml:

```
kind: Service
apiVersion: v1
metadata:
  name: {{ .Values.app }}
  namespace: {{ .Values.namespace }}
  labels:
    app: {{ .Values.app }}
spec:
  ports:
    - name: "{{ .Values.servicePort }}-tcp"
      protocol: TCP
      port: {{ .Values.servicePort }}
      targetPort: {{ .Values.containerPort }}
  selector:
    app: {{ .Values.app }}
```

values-environments/values-dev.yaml:

```
app: auth-dev
namespace: sinman-dev
```

values-environments/values-staging.yaml:

```
app: auth-staging
namespace: sinman-staging
```

Lisa 5 – GitLabi CI/CD konveieri fail .gitlab-ci.yaml

```
stages:
  - build
  - test
  - deploy

build:
  stage: build
  image: docker:stable
  variables:
    DOCKER_DRIVER: overlay2
    DOCKER_TLS_CERTDIR: ""
  only:
    - master
    - staging
    - development
  before_script:
    - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN $CI_REGISTRY
  services:
    - name: docker:dind
      command: ["--experimental"]
  script:
    - docker build --build-arg DOCKER_REGISTRY=$CI_REGISTRY --squash -t
$CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA -f docker/Dockerfile .
    - docker push $CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA

test:
  stage: test
  image: $CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA
  script:
    - service apache2 start
    - curl localhost | grep "No API token provided"
    - composer require --dev symfony/test-pack
    - php bin/phpunit
  needs:
    - build

deploy to production:
  stage: deploy
  image: git.repository.address/scoro-internal/inman/docker/sinman-deploy-
base:latest
  tags:
    - openshift
```

```
  only:
    - master
  script:
    - oc login $OKD_API_ENDPOINT --token=$OKD_API_TOKEN --insecure-skip-tls-
verify=true
    - helm upgrade --install --namespace sinman --set
image.repository=$CI_REGISTRY_IMAGE --set image.tag=$CI_COMMIT_SHORT_SHA
$CI_PROJECT_NAME ./helm-chart
  needs:
    - test

deploy to staging:
  stage: deploy
  image: git.repository.address/scoro-internal/inman/docker/sinman-deploy-
base:latest
  tags:
    - openshift
  only:
    - staging
  script:
    - oc login $OKD_API_ENDPOINT --token=$OKD_API_TOKEN --insecure-skip-tls-
verify=true
    - helm upgrade --install --namespace sinman-staging --values ./helm-
chart/values-environments/values-staging.yaml --set
image.repository=$CI_REGISTRY_IMAGE --set image.tag=$CI_COMMIT_SHORT_SHA
$CI_PROJECT_NAME ./helm-chart
  needs:
    - test

deploy to development:
  stage: deploy
  image: git.repository.address/scoro-internal/inman/docker/sinman-deploy-
base:latest
  tags:
    - openshift
  only:
    - development
  script:
    - oc login $OKD_API_ENDPOINT --token=$OKD_API_TOKEN --insecure-skip-tls-
verify=true
    - helm upgrade --install --namespace sinman-dev --values ./helm-
chart/values-environments/values-dev.yaml --set
image.repository=$CI_REGISTRY_IMAGE --set image.tag=$CI_COMMIT_SHORT_SHA
$CI_PROJECT_NAME ./helm-chart
  needs:
    - test
```