TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Stefan Artur Adov 194023IAAB

Veebirakenduse katkematu uuendamine Kubernetesega

Bakalaureusetöö

Juhendaja: Edmund Laugasson
MSc

Tallinn 2022

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: Stefan Artur Adov

13.05.2022

Annotatsioon

Käesoleva bakalaureusetöö eesmärgiks on luua kõrgkäideldav majutuskeskkond veebirakendusele, mis võimaldab antud rakenduse katkematut uuendamist kasutades Kubernetest ja toetavat vabavaralist tarkvara. Valmistatud lahendust rakendatakse ettevõttes ka edaspidi näidisena sarnaste lahenduste loomiseks.

Lõputöö käigus antud rakendus konteineriseeritakse, tagamaks selle sobivust Kubernetesega, luuakse Kubernetes kobara kõrgkäideldav keskkond ning majutatakse antud rakendus kobarasse. Viiakse läbi katsetused lahenduse peal erinevate konfiguratsioonidega, et leida lähtetingimustele parim vastav lahendus.

Lahenduse väljatöötamise käigus pööratakse tähelepanu käideldavusele, kasutajakogemusele ning rakenduse arendajale saadav kasu. Peale sobivuse lähtetingimustele kaalutakse ka potentsiaalsete edasistele tegevustele, näiteks pideva integratsiooni lahenduste osas. Lõputöö tulemuseks on taristu, mis võimaldab antud veebirakendust kasutaja vaatest katkematult uuendada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 23 leheküljel, 9 peatükki ja 16 joonist.

Abstract

Seamless Web Application Upgrade Deployments with Kubernetes

The aim of the current thesis is to develop a highly available infrastructure for a web application, which allows the application to be seamlessly updated using Kubernetes and supporting free software. The developed solution will also serve another purpose as an example for similar future projects in the company.

During the development phase of the thesis the application will be containerized, allowing it to be hosted on the Kubernetes platform, the Kubernetes highly available cluster will be deployed, and the aforementioned application will be hosted on the cluster. Tests will be run on the application hosted on the cluster to assess its quality as a solution.

The focus of the solution will be on availability, user experience and benefits for the application development process. Besides checking the solution quality based on predefined requirements, the solution will also be assessed for futureproofing and how well it leaves room to grow, for example with continuous integration solutions. Result of the thesis is a working infrastructure that allows for seamless web application upgrades and an already deployed web application.

The thesis is in Estonian and contains 23 pages of text, 9 chapters and 16 figures.

Lühendite, mõistete ja terminite sõnastik

Dockerfile	Dockerfile kirjeldab ära juhised süsteemile, kuidas konteinerifaili valmistada. Alustades sellest, missugust alustõmmist kasutada ning lõpetades sellega, et missugune käsk käivitada.
K8s	Numeronüüm või lühend sõnast Kubernetes.
CNCF	Cloud Native Computing Foundation on Linux Foundation projekt, mis haldab ja majutab palju tänapäevaseid tulevikutehnoloogiaid, näiteks Kubernetes.
JRE	Java Runtime Environment ehk Java käivituskeskkond on paigaldatav tarkvara, mis võimaldab Java programmikoodi käivitamist.
AWS	Amazon Web Services on üks suurimaid pilveteenuse pakkujaid.
YAML	Yet Another Markup Language on inimloetav andmete serialiseerimiseks kasutatav keel.
IaC	Infrastructure as Code võimaldab kirjeldada taristu olekut inimloetavates failides, mida tihti jagatakse versioonihalduses.
TTÜ	Tallinna Tehnikaülikool
Vabavara	Avatud lähtekoodiga tarkvara, mis võimaldab lõppkasutajal seda piiranguteta muuta, uurida, kopeerida, kasutada ja levitada.
Horisontaalne skaleerimine	Käideldavuse või võimekuse tõstmine objektide dubleerimise meetodil.
Kaun (ingl *Pod*)	Kubernetse objekt, mis kirjeldab ühte konteineriseeritud rakendust, mis käivitub ja täidab teatud eesmärke. Kaunasid ümbritseva keskkonna kirjeldamiseks on kasutatud paigalduse (ingl Deployment) objektid. Nendes objektides enamasti kirjeldatakse ära tõmmise asukoht registris, ketaste haakumine konteineritele ja kasutusel olevad ressursid. Paigalduse kasutamise puhul on kaunade nimed genereeritud suvaliselt, lisades nime lõppu ainulaadse sõne.
Saladus (ingl *Secret*)	Kubernetese objekt, mis sisaldab endas tundliku sisu, näiteks andmebaasiga ühendamise jaoks vajalikud parameetrid ja salasõnad. Saladusi kinnitatakse kaunade külge väliste ketastena, mis võimaldab valmistada kaunades kasutatavaid konteineri tõmmiseid selliselt, et nendes ei ole tundliku informatsiooniga faile.

Teenus (ingl *Service*)	Objekt, mis kirjeldab seda, kuidas kobaravälised võrguseadmed saavad ligi kobaras pakutavatele teenustele, millest saab mõelda ka kui koormusjaoturit, mis pakub kaunadele staatilise võrguaadressi. Selleks, et kasutada teenustele ligi pääsemiseks näiteks domeeninime alusel ruutimist, saab kasutada ka veel teenuse ees Sissepääsu (ingl Ingress), mis võimaldab ka sealhulgas kasutada krüpteeritud liiklust andmeedastuseks.

Sisukord

Jooniste loetelu

1 Sissejuhatus

Konteineriseerimine on leidnud aina rohkemates ettevõtetes omale rolli kui järgmine samm virtualiseerimisele. Ennustatavalt 2022. aasta lõpuks on enam kui 75% globaalsetes ettevõtetes kasutusel konteineriseeritud teenused, kasv 2020. aasta 30% pealt[4] . Konteineriseerimisel on palju kasulikke külgi, kuid nende modulaarsus ning kergekaalulisus võrreldes virtuaalmasinatega võimaldab nende lihtsa skaleerimise ning tööle panemise paljudes keskkondades. Ühine platvorm nii arendaja arvutis kui ka lõplikus toote arvutis annab parema kontrolli lõpptoote üle, vähendades arendusprotsessi keerukust.

Tänapäeva vestlustes käivad käsikäes konteinerid ja mikroteenused. Mikroteenused on enamasti väikesed rakendused, mis töötavad üle HTTP protokolli ja teenindavad mingisugust väikest eesmärki. Traditsiooniliselt on veebirakendused ehitatud otse üksikute virtuaalmasinate peale, mis ei ole sobilik keskkond mikroteenustele. Selline majutus oleks ülemäära kallis, kuna teenus ise on väike, ning ka kohmakas, kuna mikroteenuseid on vaja horisontaalselt skaleerida.

Lõputöö raames konteineriseeritakse veebirakendus, mis koosneb taga- ja eesrakendusest, luuakse Kubernetes kobar ning majutatakse veebirakendus konteineriseeritud kujul kobarasse. Kuberneteses sees majutatud teenuse vastupidavust kontrollitakse uuenduse ajal nõudega, et veebirakendus vastab pidevalt uuenduse käigus etteaimataval moel.

Lõputöö käigus analüüsitakse virtuaalmasinas rakenduse uuenduse, Kubernetes kobara sees veereva uuenduse ning Kubernetes kobara sees „roheline/sinine” uuenduse rakendamise mõjusid kasutajakogemusele ning veebirakenduse kättesaadavusele. Lõputöö eesmärgiks on veebirakenduse uuendamine nii, et kasutaja ei ole häiritud ning veebirakenduse töö säilib uuendamise protsessi käigus.

2 Rakenduse kirjeldus ja hetkeseisu infosüsteem

2.1 Rakenduse kirjeldus

Töös käideldav veebirakendus koosneb kolmest osast – tagarakendusest, mis on kirjutatud Java programmeerimiskeeles, ning kasutab Spring Boot raamistikku, eesrakendusest, mis on kirjutatud Javascript programmeerimiskeeles, ning kasutab Angular raamistikku, ning ühest PostgreSQL andmebaasist.

Tagarakendus kasutab programmikoodi käivitamiseks vabavaralist AdoptOpenJDK[1] Java virtuaalmasinat. Lisades ka juurde teadmise, et tagarakendus on üpriski mahukas, siis selle käivitamine võtab keskmiselt pool minutit, ning sellel perioodil näitab veebirakendus lõppkasutajale veateadet.

2.2 Hetkeseisu infosüsteem

Veebirakenduse teenindamiseks on kasutusel kolm virtuaalmasinat, iga rakenduse komponent on eraldi virtuaalmasinal. Andmebaasile on provisioneeritud 4 virtuaalset protsessorituuma ja 12GiB muutmälu, tagarakendusele 4 virtuaalset protsessorituuma ja 16GiB muutmälu ning eesrakendusele 2 virtuaalset protsessorituuma ja 2GiB muutmälu.

Kõik masinad praeguse seisuga töötavad vabavaralise Arch Linux operatsioonisüsteemi baasil, mis raskendab süsteemide haldamise ja uuendamise protsessi, kuna on kasutusel rolling release tüüpi uuendused, ning Arch Linux ametlikult ei toeta poolikuid uuendusi[25] . Sellises operatsioonisüsteemis on paigaldatud paketid omavahel tugevalt seotud, ning nõuab kõikide üheaegsed uuendamist. See on vastuolus tagarakenduse raamistiku poolt seotud piirangutega kasutada vanemaid versioone Java virtuaalmasinast[2] , ning operatsioonisüsteemi uuendamine on sedavõrd rohkem raskendatud.

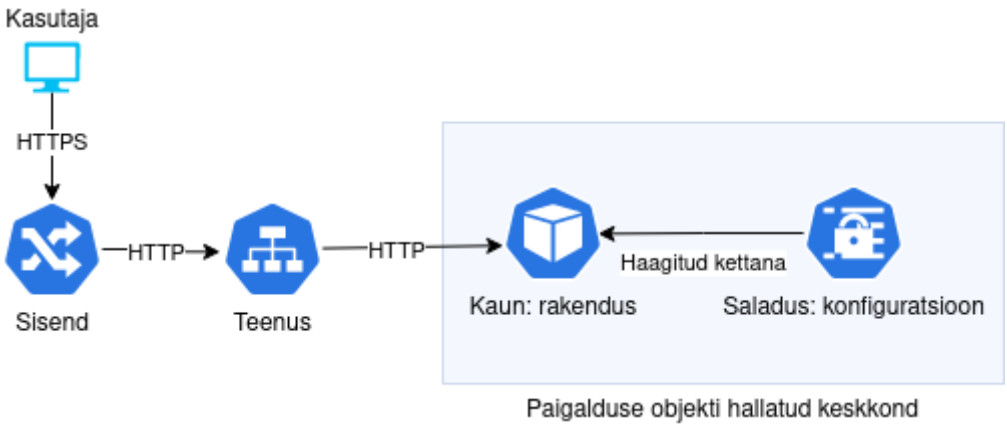
3 Kubernetese ülevaade

Kubernetes on vabavaraline süsteem, mis võimaldab hallata konteineritel töötavaid teenuseid ja ülesandeid. Välja töötatud Google'i poolt kasutades üle 15 aasta jooksul kogunenud erialaseid teadmisi ja kogemusi[3] , tänaseks on saanud selle ja sarnaste tehnoloogiate teenustena pakkumisest tööstus, mille väärtus ulatub mitmesaja miljoni euroni[4] .

Kubernetese süsteemis on olemas sõlmed (ingl *node*), mille töö on konteineriseeritud rakenduste tööle panemine, ning juhtimistasand (ingl *control plane*), mis haldab kõiki sõlmi ja nende poolt teenindatavaid kaunasid (ingl *pod*). Kõik komponendid on kujundatud selliselt, et neid oleks lihtne horisontaalselt skaleerida.

Kobaraga suhtlemiseks on kasutusel *kubectl[5]* käsk. Selle käsuga on võimalik anda otseseid käske, tuntud kui ka imperatiivsed käsud, mis teevad otse käsus kirjeldatud tegevused kobaras. Selline käsitlus on soovitatav kasutada ainult arenduskeskkondades, tootekeskkondades on soovitatav rakendada imperatiivset objekti konfiguratsiooni või deklaratiivset objekti konfiguratsiooni[6] . Lõputöö käigus rakendan ma imperatiivset objekti konfiguratsiooni, mis tähendab üksikute failide kirjutamist YAML kujul. Nende failide sisu kirjeldab ära IaC kujul seda, missugust kobara ja selles majutatud objektide olekut ma antud hetkel tahan.

Selle lõputöö käigus juurutan Kubernetes kobara kolme juhtimistasandi ning kolme sõlmega. Kõik kolm sõlme panevad tööle veebirakenduse samaaegselt, mis võimaldab kasutajale esitada veebirakendust normaalkujul ühe sõlme peal samal ajal, kui teised sõlmed tegelevad uuendusega.



Joonis 1. Päringu näide paigalduse objekti kasutamise puhul.

Joonis 1 kirjeldab kuidas kliendi päring liigub läbi Kubernetese objektide, kui kasutusel on kobaraga paigaldusega kaasa tuleva sisendi objekt. Sisend tegeleb antud juhul ka kliendi ja kobara vahelise turvalise liikluse haldamisega. Selle objekti peamisteks tööülesanneteks antud juhul on liikluse dekürpteerimine ja edastamine läbi teenuse kauna suunas. Kaunast tuleb tagasi vastus

4 Metoodika

Lõputöö tulemus on saavutatud eksperimendi tulemusel. Valmistatakse kolm erinevat veebirakenduse majutamise keskkonda, mida võrreldakse üksteise vastu. Kubernetes valiti arendatava süsteemi aluseks kuna tegemist on vabavaralise tarkvaraga, mis võimaldab ühendada pilveteenused ja riistvaralised sõlmed, ning pakub rohkelt paindlikkust süsteemi seadistuse osas ning tugevat tuge horisontaalsele skaleerimisele. Kubernetese rakendamine nõuab rakendustelt konteineriseerimist, mis on ka lõputöö osa. Konteineriseerimine peab olema läbi viidud sedasi, et rakendusele ei ole tehtud kahju funktsionaalsusele, käideldavusele või jõudlusele.

Võrdlemise tulemuste kogumiseks oli kasutusel skript (joonis 2), mis tegi 2000 päringut teenuse pihta. Pärast esimest 150 päringut saadeti kobarale käsk käivitada uuenduse protsess. Vastavalt vajadusele eraldati välja ja asendati parameetrid, kuna kaks erinevat keskkonda oli vaja läbi katsetada. Andmed väljundati komaga eraldatud faili, mis võimaldas nende töötlemist vabavaralise kontoritarkvaraga LibreOffice Calc'i.

```
#!/bin/bash

FILENAME=http_status_deployment.csv
URL=https://test.rakendus.firma.ee
# URL=http://test.rakendus.firma.ee:30764
for i in {1..2000};
do
    if [[ $i -eq "150" ]]
    then
        # kaf VirtualService.yaml
        kaf Deployment.yaml
    fi
content=$(curl -s -X GET --head -m 0.5
https://test.rakendus.firma.ee/backend/getListJson
--insecure)
    echo -n "$i;$?;$(date +"%Y-%m-%d %T.%3N");" >>
$FILENAME;
    echo $content | grep -Eo "[0-9]\.[0-9]\.[0-9]"
| tee -a $FILENAME;
    echo "" >> $FILENAME
done
```

Joonis 2. Andmete kogumiseks kasutatud kesta skript.

5 Lahenduse väljatöötamine

5.1 Lähtetingimused

Loodud lahendus peab võimaldama kliendil kasutada veebirakendust samal hetkel, kui rakendus on läbimas uuendamise protsessi. Välja töötatud lahendus peab rakendama vabavaralist tarkvara, tagamaks parema turvalisuse läbi läbipaistvuse. Vabavaralise tarkvara kasutamine võimaldab ka vajadusel edaspidiselt muuta tarkvara lähtekoodi, et täita ettevõtte erijuhtusid paremini.

Kasutatud lahendus peaks kasutama piisavalt tuntud tarkvara, et võimaldada jätkusuutlik pikaajaline haldus, võttes arvesse tööturul pakutava tööjõuga. Kui lahenduse välja töötamiseks oleks kasutusel patenteeritud, tarkvara, näiteks IBM või Oracle poolt pakutav pilv, selle piiratud kättesaadavus õppimise eesmärgil võib suurel määral piirata võimalike haldurite hulka ning seetõttu ka tõsta kulusid.

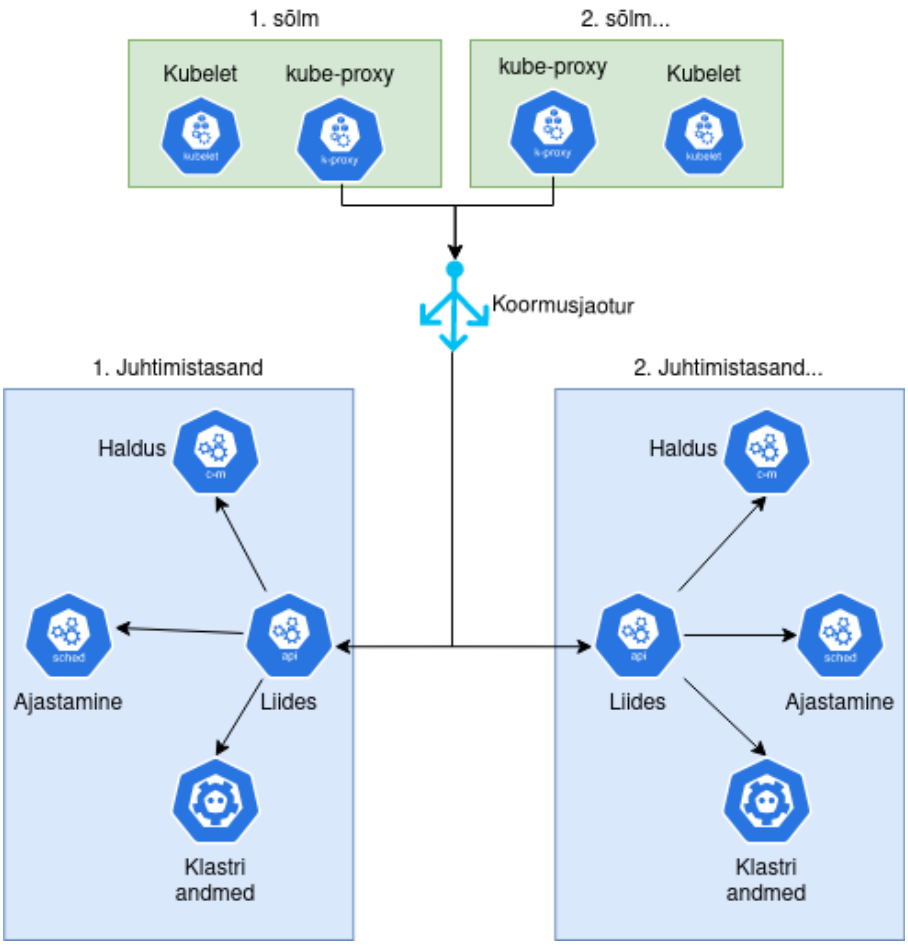
Selleks, et tagada täielik kindlus katkematuse osas, on nõutud tõrgete puudumine antud katsetuste käigus, sest tõrgete puudumine uuenduste käigus võimaldab ettevõttel kasutajale nähtamatul kujul esitada uut versiooni või rakendada turvauuendusi. Peale tõrgete puudumise on arvestatud ka inimtegevusest tulenevate vigadega. Kui rakenduses on programmaatiline viga, mis jäi arenduse protsessis olevate katsetuste käigus märkamata, siis on vajalik võimalus võtta tehtud muudatused kiiresti ja kasutajale märkamatult tagasi.

Kokkuvõtvalt, rakendus peaks vastama järgmistele tingimustele:

- klient peab saama kasutada rakendust uuendamise käigus;
- välja töötatud taristu lahendus kasutab vabavaralist tarkvara;
- taristu jaoks kasutatud tarkvara peab olema tuntud, tagamaks võimaluse värvata tulevikus inimesi süsteemi haldama;

- uuendus peab võimaldama ka samamoodi nähtamatult võimaldama rakenduse uuenduse tagasi võtmist.

5.2 Kobara ülevaade



Joonis 3. Virtuaalmasinate rollid.

Planeeritud Kubernetese ökosüsteem koosneb kuuest virtuaalmasinast – kolm juhtimistasandit ja kolm sõlme (joonis 3). Eksperimendi jaoks on vaja vähemalt kahte sõlme, kuna veebirakendusest peab olema vähemalt kaks koopiat, et neid saaks kordamööda uuendada. Tootekeskkonnas on soovituslik kasutada vähemalt kolme juhtimistasandit, et tagada infosüsteemide kõrgkäideldavus.

5.3 Infosüsteemi juurutamine

Kobara juurutamiseks kasutasin *kubeadm* tööriista, mis võimaldab juurutada Kubernetese kvaliteeditestidele vastava kobara[7] . Juurutamine toimus Ubuntu distributsiooni 20.04 versiooni peal, tagamaks pikaajalise uuendustega seotud toe. Kubernetese sõlmi ning juhtimistasandeid saab ohutult tühjendada[26] , mis võimaldab alloleva operatsioonisüsteemi uuendamise ilma katkestuseta. Iga sõlm oli paigaldatud eraldi virtuaalmasinasse, mis on hoiustatud serveripargis, millel on sõlmi nii Tartus kui Tallinnas. Otsus paigaldada kobar virtualiseerimiskihi peale ja mitte otse *bare-metal* tugineb sellele, et virtuaalmasinaid on lihtsam taastada hetktõmmiste (ingl *Snapshot*) abil, lihtsustatud kloonimine ning lihtsalt olemasoleva taristu peale toetumiseks, mis võimaldas meil teha vähem tööd.

Hetkeseisuga on rakenduse kood majutatud *GitLab* versioonikontrolli süsteemis, millest ei oleks kobarale suuremat kasu. Selleks, et rakendust tööle panna Kubernetese kobaras, on vaja selle konteineri tõmmis registris. Üks võimalus oleks kasutada *Docker*i enda poolt arendatavat *Docker Registry* tarkvara[8] , kuid me otsustasime kasutada CNCF kraadiga[9] projekti *Harbor*, mis võimaldab peale sama funktsionaalsuse ka teha majutatud tõmmistele teha turbe ja haavatavuse analüüsi[10] . Registri aktiivne kasutuselevõtt tähendab ka omakorda protsessi muudatust ettevõttes. Praeguse seisuga tähendab see seda, et enne uuenduse paigaldust on vaja administraatoritel teha lisasamm, kus versioonihalduses olev kood paigaldatakse konteinerina registrisse.

5.4 Rakenduse konteineriseerimine

Rakenduse konteineriseerimise eesmärk on tagada rakenduse töö konteineris sees samal kujul, nagu ta praegu on virtuaalmasina peal. Konteineri ehitamise (ingl *Build*) käigus paigaldatakse rakenduse tööks vajalikud teenused (näiteks Java käivituskeskkond või NGINX veebiserver) ning kopeeritakse ka failid (näiteks tagarakenduse .jar-fail või eesrakenduse staatilised .js- või .html-failid). Kobara puhul enne rakenduse käivitamist haagitakse neile külge ka konfiguratsioonifailid ja muud parameetrid. Selliste parameetrite ja tundlike andmete majutamine konteineris ehitamise ajal sees võimaldab potentsiaalsetel küberkurjategijatel skaneerida konteinerite tõmmiseid ning nendest varastada näiteks AWS keskkonna paroole[11] .

Tagarakendus on ehitatud Java programmikeeles, mis omakorda tähendab ka seda, et rakenduse käivitamine on siiani toimunud kasutades Java virtuaalmasinat (joonis 5). Java virtuaalmasin võimaldab arendajal kirjutada koodi selliselt, et nad ei peaks muretsema milline süsteem on kasutusel, ainuke nõue on kasutada toetatud versiooni Java käivituse keskkonnast (*JRE*). See omadus suuresti lihtsustas rakenduse konteineriseerimise protsessi (joonis 4), kuid näiteks rakenduse konfiguratsioonifail ja kasutatavad sertifikaadid on majutatud väljaspool Java virtuaalmasinat, ning need tuleb külge haakida Kubernetes abiga.

```
FROM adoptopenjdk:8-jre-hotspot

WORKDIR /code

COPY ./app.jar ./app.jar

CMD java -Dfile.encoding=UTF-8 -
Djavax.net.ssl.trustStore=/code/certs.keystore -
Djavax.net.ssl.trustStorePassword=AComplexPassword
-jar /code/app.jar
```

Joonis 4. Tagarakenduse Dockerfile

Joonis 5. Konteineriseeritud tagarakenduse paiknemine virtuaalmasinas.

Eesrakendus on serveri vaatest staatiliste failide edastamine kasutades NGINX veebiserverit. Käigu pealt sisu genereerimine, andmete sisestamine ja muutmine käib kõik tagarakenduse kaudu. Eesrakendus küll hõlmab endas ka interaktiivset JavaScript

alusel ehitatud sisu, kuid selle töötlemine toimub kõik kliendi arvutist, mistõttu see ei sõltu antud olukorras konteinerist (joonis 6). Veebiserveri rakendus käivitub konteineris, mis on omakorda isoleeritud keskkonnas virtuaalmasinas (joonis 7).

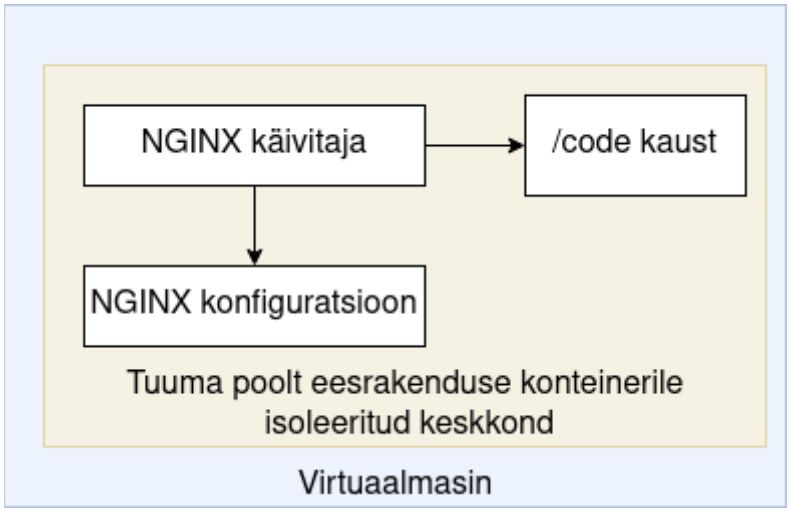
```
FROM nginx:1.20.2

WORKDIR /code

COPY ./default.conf /etc/nginx/conf.d/default.conf

COPY ./html /code
```

Joonis 6. Eesrakenduse Dockerfile

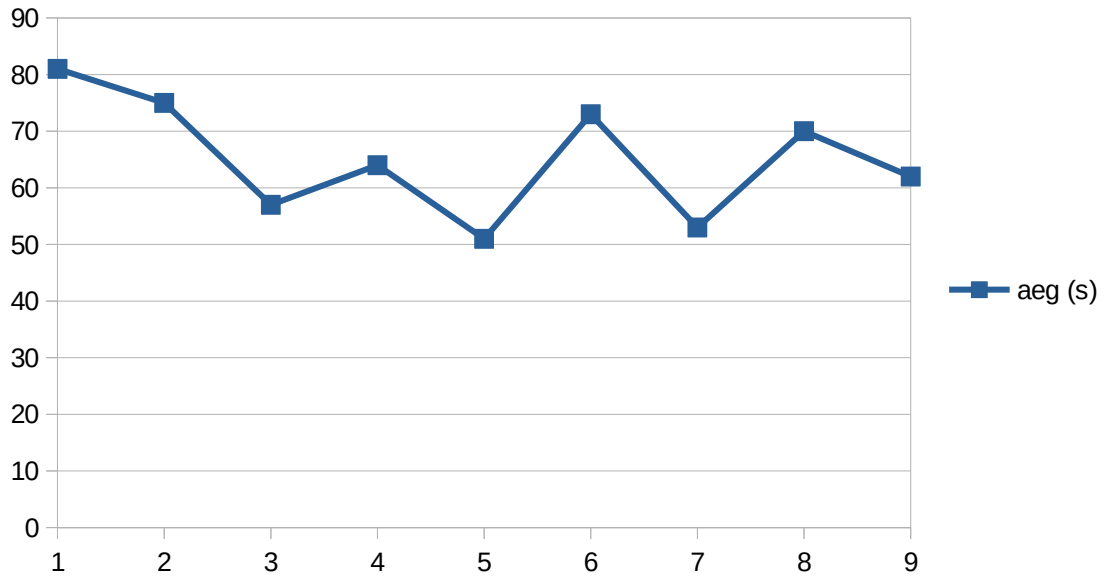


Joonis 7. Konteineriseeritud eesrakenduse keskkonna paiknemine virtuaalmasinas.

6 Uuendamise protsesside võrdlus

6.1 Hetkel kasutusel protseduur

Praeguse infosüsteemiga rakenduse uuendamise protsess väljendub kasutajatele tavaliselt minuti pikkust katkestust, mis hõlmab endas kõiki veebirakenduse teenuseid. Enamjaolt katkestus tuleb sellest, et tagarakendusest ei saa töötada kaks koopiat ühe masina peal, mistõttu kõik tegevused toimuvad erinevatel aegadel. Enne kui on võimalik uut versiooni hakata käivitama, on vaja eelmine versioon täielikult sulgeda, mis tähendab juba kasutaja vaatest katkestuse algust. Tegemist on Java virtuaalmasina rakendusega, ning nagu tavaliselt on kombeks virtuaalmasinatel, võtab selle käivitamine pisut aega, enamustel juhtudel alla ühe minuti (joonis 8).



Joonis 8. Katkestuse pikkus sekundites viimase üheksa uuenduse puhul.

Kasutusel on automatiseerimistarkvara Jenkins. Jenkins on juhtiv vabavaraline tarkvara, mis võimaldab automatiseerida erinevaid tarkvara ehitamise, katsetamise ja paigaldamisega seotud tegevusi[12] . Hetkel kasutusel olev skript võtab

programmikoodi versioonikontrollist, paigaldab selle veebiserverile peale ning taaskäivitab veebiserveris vajalikud teenused muudatuse rakendumiseks.

6.2 Kubernetesega „veerev uuendus”

Kui rakendus on üles sätestatud selliselt, et kobar teenindab sellest vähemalt kahte koopiat, siis on võimalik kasutada Kubernetese veerevat uuendust (ingl *rolling update*). Vaikimisi veerev uuendus asendab rakendust teenindatavad kaunad ühe kaupa uuemate vastu, ning suunab kõik sissetulevad päringud ainult valmis olekuga kaunade pihta. Kui uued kaunad on valmis päringuid vastu võtma, siis tasakaalujaotur saadab liiklust ka nende suunas. Selle tõttu on olemas vahepealne faas, kus kasutajatele teenindatakse korraga nii vana kui ka uut rakendust.

Uuenduse algfaasis alustatakse uue kauna käivitamist. Kuna uue versiooni kaun ei ole valmis, siis kasutaja vaatest ei ole rakendus muutunud (joonis 9). Sel hetkel, kui uus kaun on valmis olekuga, võetakse üks vana kaun maha, ning kasutaja võib sattuda edaspidi kas uue või vana lehekülje pihta. Pärast seda, kui kõik vanad kaunad on asendatud, on kasutaja vaatest ainult saadaval uus versioon rakendusest.

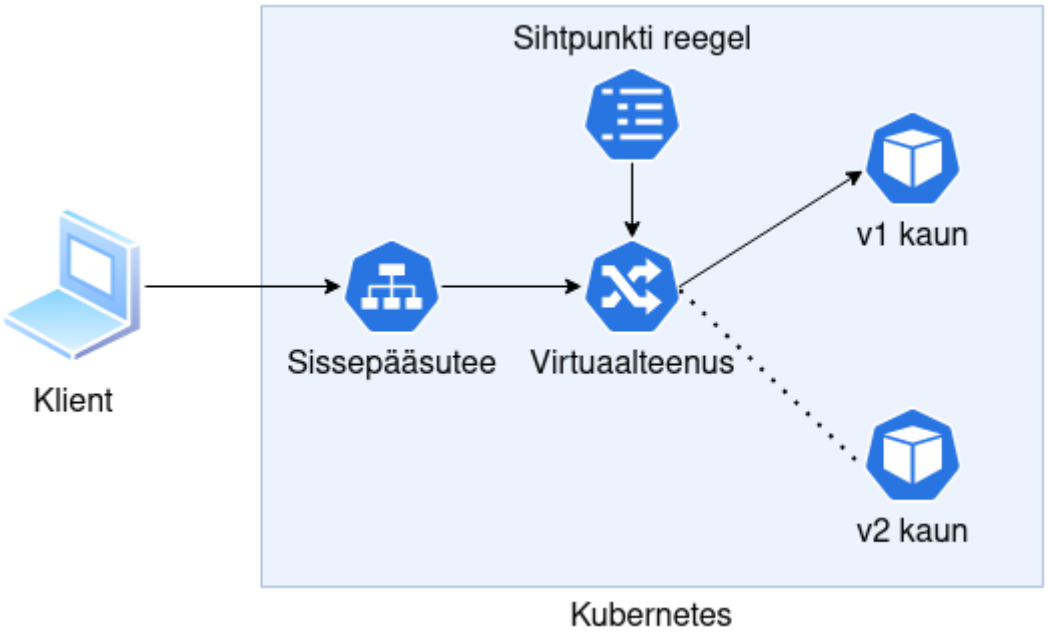
Kubernetes vaikimisi toetab ainult sedasorti katkematu uuendamise protsessi, kuid teenuseräsi (ingl *Service Mesh*) kasutades on võimalik saavutada palju kontroll oma rakenduse liikluse üle. Joonise pealt saab välja lugeda, et veereval uuendusel on tegemist ringmängu tüüpi tasakaalujaotusega, kus erinevate sõlmede poole pöördutakse kordamööda.

6.3 Kanaari paigaldused (roheline-sinine, punane-must)

Istio[13] võimaldab rakendada Kuberneteses veebirakenduste uuendamise protsessis kanaari paigaldusi (ingl *Canary Deployments*)[14] . Kanaari paigaldused võimaldavad esitada veebirakendusele tehtud uuendusi väikesele kasutajahulgale, piirates võimaliku kahju ning kasutades reaalseid kasutajaid kui testijaid. Kuigi Kubernetes võimaldab[15] sellist funktsionaalsust piiratud kujul ilma muudatusteta, siiski on vaja Istio kasutusele võtta et olla meetodi rakendamisel efektiivne. Istio ei ole ainus võimalik teenuseräsi

valik, on ka olemas muid variante, näiteks HashiCorp Consul[31] ja Linkerd[30] , kuid Istio valiti toetudes töötajate eelnevatele kogemustele, selle paindlikkusele, arvestades et tulevikus oleks vaja välja töötada hulga erilahendusi.

Lõputöö esimestes peatükkides seletasime ära ideed kaunadest, saladustest ja muudest Kubernetese ressursi objektidest. Kubernetes võimaldab ka kasutajatel ning kobara arendust toetatavatel arendajatel teha kohandatud ressursi kirjeldusi (ingl *Custom Resource Defintinion*). Kohandatud ressurssi kirjeldused võimaldavad pakkuda kobarale lisafunktsionaalsust sedavõrd mugavalt, nagu need olevat kobarasse juba sisseehitatud.



Joonis 9. Istio liiklusega seotud Kubernetese objektide suhted.

Istio puhul on lõputöö käigus vajalikud kolm ressursi kirjeldust: virtuaalne teenus (ingl *Virtual Service*), sissepääsutee (ingl *Gateway*) ja sihtpunkti reegel (ingl *Destination Rule*) (joonis 10). Virtuaalne teenus kirjeldab sissetuleva liikluse ruutingu parameetrid, enamasti missuguse rakenduse versiooni pihta läheb ja missuguse rõhuga on erinevad sihtpunktid[16] . Sissepääsutee töötab nagu teenus, lubades väljastpoolt tuleval liiklusele ligipääsu kobaras sees olevate teenustele[17] . Sihtpunkti reegel võimaldab ruutingujärgseid konfiguratsioone, lõputöö käigus kasutame seda ressursi et kirjeldada kuidas eristada ühte versiooni teisest.

Nimetused roheline-sinine ja punane-must viitavad sellele, et kaks sarnast, kuid siiski erinevat, keskkonda töötavad paralleelselt. Tasakaalujaoturi abil on võimalik suunata liiklust vastavalt vajadusele kas uuele (rohelisele) keskkonnale või vanale (sinisele) keskkonnale. Istio võimaldab sellise praktika rakendamiseks täpset kontrolli sissetuleva liikluse üle. Mõlemale keskkonnale saab anda kaalu, mis määrab ära selle, kui palju liiklusest läheb ühes või teises suunas. Võttes näiteks sinine keskkond on kaaluga 1 ja roheline keskkond on kaaluga 99, see tähendab seda, et iga 100 päringu peale saabuvad 10 päringut sinises keskkonnas ja ülejäänud 990 jõuavad kõik rohelisse keskkonda (joonis 11).

```
- route:
  - destination:
      port:
        number: 80
      host: rakendus
      subset: v1
    weight: 99
  - destination:
      port:
        number: 80
      host: rakendus
      subset: v2
    weight: 1
```

Joonis 10. Väljavõte Istio virtuaalse teenuse objekti kirjeldusest, kus 1% sissetulevast liiklusest suunatakse uuele versioonile.

Selleks, et Istios kanaari paigaldused töötaksid, on vaja ära kirjeldada kuidas versioonide määramine toimub. Kuberneteses saab kasutada silte, et kirjeldada objektide kuuluvust. Tihti on kasutusel „app" nimeline silt, mis võimaldab leida rakendust olenemata sellest, missugune ainulaadne genereeritud kauna nimi tal on. Kanaari paigalduste rakendamiseks on vaja lisada paigalduse objektile juurde „version" silt (joonis 12), mis lisab selle sama sildi ka tema poolt tekitatud kaunadele. Järgnevalt on vaja seostada „version" silt sihtpunkti reegli objekti abil ka Istiole mõistetavaks, et süsteem ostaks ruutingut teha vastavalt soovile (joonis 13). Istio poolt lisatud uute objektide tutvustamine omakorda lisab kolmanda potentsiaalse protsessi administraatori vaatest (joonis 14).

```
spec:
  host: rakendus
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
```

Joonis 11. Väljavõte Istio sihtpunkti reegli objekti kirjeldusest, kus on kirjeldatud ära versioonide määramiseks kasutatud märksõna „version”.

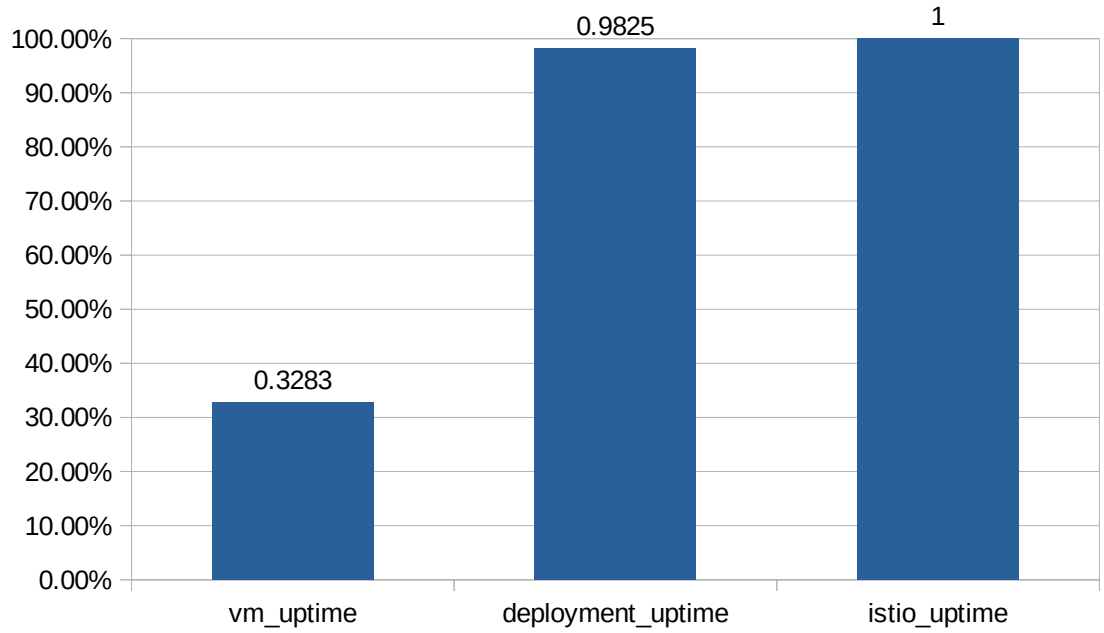
```
metadata:
  name: rakendus-v2
  labels:
    app: rakendus
    version: v2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rakendus
      version: v2
  template:
    metadata:
      labels:
        app: rakendus
        version: v2
```

Joonis 12. Väljavõte rakenduse paigalduse objekti kirjeldusest, kus kaunadele ja rakenduse grupile pannakse külge „version” sildid (ingl *label*).

Joonis 13. Kanaari uuenduse protseduur võrreldes hetkelise protseduuriga ja veereva uuenduse protseduuriga.

7 Tulemuste analüüs

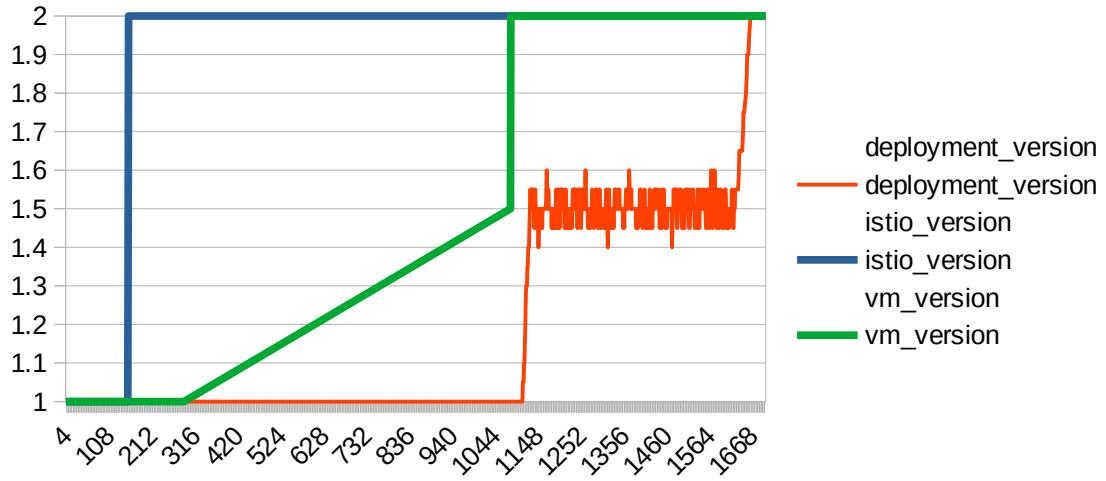
Lõputöö alguses püstitatud ülesande täidavad mõlemad nii Istioga kui ka Kubernetese paigalduse objektiga kirjeldatud versioonid rakendusest. Paigalduse objekti kasutamisest tingitud viivitus on tingitud rakenduse äsjasest käivitusest, kus päringule vastamine võtab oodatust kauem aega. Teenuse käideldavus oli kasutaja vaatest märgatavalt mõjutatud virtuaalmasinas majutatud rakendusega, kus terve rakendus oli kättesaamatu. Istio pakkus selles katsetuses kõige paremaid tulemusi, kuna teenuse nii uus kui ka vana versioon olid tükk aega juba töötanud paralleelselt ning vahemälu oli rohkem rakendatud, luues võimaluse kiirematele päringu vastustele.



Joonis 14. Rakenduse käideldavus uuenduse käigus virtuaalmasina infosüsteemis (vm_uptime), Kubernetese paigalduse objektina (deployment_uptime) ja Istio virtuaalse teenusena (istio_uptime).

Rakenduse versioonide vahetuse vaates oli taaskord kõige kiirema mõjuga Istio. Seda võimaldas taaskord see, et rakenduse mõlemad versioonid olid paralleelselt tööl ning ainuke vajalik muudatus kobara vaatest oli liikluse ümber lülitamine teise kauna pihta. Kubernetese paigalduse objekti kasutamisega on näha see, et tükk aega kulub esimese

käivitamise peale, pärast seda on näha kuidas esitatakse kasutajatele mõlemaid versioone korraga, ning lõpus esitatakse kasutajatele ainult uuem versioon. Algse virtuaalmasinal põhjal loodud rakenduse peal on tulemus etteaimatav – enne katkestust on teenindatud vana versioon ning pärast katkestust on teenindatud ainult uus verisoon.



Joonis 15. Rakenduse versiooni trendi võrdlus Istio (istio_version), Kuberentese paigalduse objekti (replica_version) ja virtuaalmasina (vm_version) vahel.

Programmaatilise või muu põhjusega tingitud nõudest tagasipööret teha on olukord sama, kõige kiirema mõju kasutaja vaatest annab Istio, kuna nii roheline (uus) kui sinine (vana) keskkond töötavad paralleelselt, siis on võimalik kiire lülitamine liikluse ruutingu abil. Kubernetese paigalduse objekti puhul on tagasipööramine antud juhul kõige aeglasem, kuna protsessis on vahepealne aeg, kus teenindatakse nii vana kui ka uut versiooni paralleelselt klientidele.

Üks mõttekoht sedasorti süsteemi juurutamisel on ka ainulaadsed võimalused, mida pakub läbi sellise taristu toote arendajatele. Kubernetes pakub küll võimalust käsitsi paigaldada mitu versiooni samast rakendusest ning vastavalt vajadusele käsitsi teha ruutingut nende pihta, kuid see nõuab et vajaliku suhte jaoks on paigaldatud ka omajagu kaunasid. Näiteks kui on plaanis teha 10:3 suhe, mis tähendab 10 päringut läheb vana teenuse pihta ja kolm läheb uue teenuse pihta, on selle jaoks vaja paigaldada 13 erinevat kauna ning need käsitsi konfigureerida, samal ajal kui Istio võimaldab teha vajadusel 99:1 suhtega liiklust 2 kaunaga

Istio kasutamine kanaari paigalduseks kasutab omakorda, isegi kui ajutiselt, rohkem ressursse, mis ei saa reaalselt kasutaja vaatest kasutust. Kuna sellise funktsionaalsuse jaoks on vajalik, et teenused oleks koheselt kättesaadavad, siis nende alles hoidmiseks kulub ka järelikult kobara ressursse, enamjaolt protsessori ja muutmälu näol.

Kaaluga liikluse ruutimine ja paigalduse objektiga ehitatud rakenduse taristu võib olla kahjulik tingimusel, kus on muudatuse ajal käigus ka taustal toetava andmebaasi struktuuri muudatus, kus on vajalik et kõik või enamus kliente korraga lülituks ümber uuele süsteemile, kuid sedasorti probleemi saab välistada versioonitud rakendusliidesega või kasutades ajutiselt rakenduse koodi tasemel ühilduvuskihti.

Arvestades asjaoludega, et tänapäeval on veel laialt[20] maailmas kasutusel suurarvutid mis töötavad COBOL-i baasil, 1959. aastal välja töötatud programmeerimiskeel[19] , siis uuemate tehnoloogiate kasutuselevõtt võib ettevõtte vaatest olla riskantne ja kallis protsess. Konteineritehnoloogiate, Kubernetesega ning veel enam teenuseräsi Istio kasutuselevõtt võib näha tugevat vastupanu vanemarhitektitelt või vanemadministraatoritelt, kes on teinud sarnaseid protsesse aastaid, luues oma nišši jaoks parimad praktikad ning neid rangelt jälginud. Selline suhtumine võib ka seletada seda, et miks peaaegu pooled pangad tänapäeval siiani kasutavad COBOL-i peale ehitatud suurarvuteid oma kriitilisemates süsteemides[21] .

Sarnane probleem esineb ka mündi tagaküljel, kus idufirmadel pole raha ega ressursse et palgata tööjõudu, kes suudaks sellise pilve taristu neile valmis planeerida, ehitada ning seda ka järjepidevalt hallata. Sellise töö keerukus omakorda tõstab tööjõu hinda ning vähendab ka potentsiaalsete kandidaatide hulka. Usun, et TTÜ initsiatiiv omistada oma õppekavas mikroteenuste ja konteineriarhitektuuri aineid edendab sellise tööturu kasvu.

8 Edasised tegevused

Andmebaasiga seotud struktuuri muudatused on enamasti aeganõudvad ning kasutajakogemust häirivad. Enamjaolt kui teha andmebaasi tabelis muudatus, siis selle tabeli kõik kirjed on vaja uuesti üle kirjutada. Sellise probleemi vältimiseks võiks rakendada meetmeid, mis võimaldab teha muudatusi ilma andmebaasi lukke kasutamata. Percona pakub sellist lahendust nimega *pt-online-schema-change*[22] , kuid veebirakendus kasutab PostgreSQL andmebaasimootorit ning see lahendus töötab ainult MySQL andmebaasimootoriga. Hetkeseisuga Percona ei paku sarnast lahendust PostgreSQL andmebaasimootorile[23] .

Ettevõttes kasutusel oleva versioonihalduse GitLab on võimalik automatiseerimistarkvara Jenkins abil siduda konteineri registriga nii, et versioonihalduses tehtud koodimuudatused jõuaksid automaatselt ka kohe konteineri registrisse. Selline automaatika välistaks vajaduse käsitsi konteineri tõmmiste tegemiseks. Vajadusel on ka võimalik automaatika abil teha muudatusi ka kobaras selliselt, et uus tõmmis võetakse kasutusel test-keskkonnas.

Etcd on kobara andmekogum, millest sõltub kobara üldine tervis ja töö. Selleks, et tagada kobara maksimaalne efektiivsus, on vaja etcd teenus majutada eraldi masinas kiirele kettale[24] . Vaikimisi paigaldusega on kobara juhtimistasandid ehitatud kuhjatud (ingl Stacked) etcd kujul. Tulevikus oleks parema käideldavuse ja kiiruse tagamise jaoks vaja paigutada see teenus eraldi virtuaalmasinasse, mis ei sõltu ülejäänud juhtimistasandist.

Ühesuguse monitooringu rakendamise kõikidele teenustele võimaldaks ka anda lõppkasutajatele ettevõtte majutatud teenuste olekust ülevaadet leheküljel. Sedasorti ülevaade võimaldaks kliendil vea korral hinnata kas probleem on tema enda võrguühenduses või on teenus maas teenusepakkuja tasemel.

Tulevikus tuleks arvestada ka sellega, et kas oleks vajadus välja vahetada Ubuntu operatsioonisüsteem millegi sellise vastu, mis pakub paremat tuge ettevõtetele. Sellisteks võivad olla näiteks RHEL[27] , Rocky Linux[28] või SUSE[29] .

9 Kokkuvõte

Lõputöö eesmärgiks oli luua lahendus, kus veebirakenduse ees- ja tagarakenduse komponendid oleksid uuendamise protsessi ajal kättesaadavad.

Esmane samm selle suunas oli teha ülevaade praegusest rakenduse majutamise keskkonnast ning selle puudujäägid välja tuua. Kuna plaanis oli majutada antud teenus Kubernetese peal, siis töö käigus konteineriseeriti eesrakendus ja ka tagarakendus, mis võimaldaks seda majutada.

Kubernetese kobar sai paigaldatud kõrgkäideldavalt, mis tähendab et kasutusel oli vähemalt kolm juhtimistasandi sõlme, ning paigaldati ka teenuseräsi Istio, mis võimaldas palju rohkem paindlikkust ülesande täitmise osas.

Teenuse majutamistingimusi võrreldi üksteise vastu kolme põhilise parameetri osas – teenuse käideldavus uuendamise käigus, teenuse poolt pakutud versioon ning teenuse tagasipöörde efektiivsus.

Töö tulemus võimaldas käsitletud veebirakendus majutada tehtud kobarasse ning ka täita ka eeskuju rolli sarnastele rakendustele, mis tulevikus kobara peale majutatakse. Väljatöötatud kobarat kirjeldavad konfiguratsioonifailid, mida ei ole võimalik lõputöös välja tuua nende mahu tõttu, saab hästi rakendada ka tulevikus teiste rakenduste lahenduste väljatöötamisel. Algselt traditsioonilises majutatud veebirakendusest sai lõputöö käigus konteineriseeritud ja lihtsasti skaleeritav rakendus, mida majutati üha enam kasvavas Kuberenetese kobaras.

Lõputöö käigus käsitletud ja uuritud teenuseräsi Istio koha pealt omas autor hulgaliselt uusi teadmisi, mille abil võimaldab taristu poole pealt seda enam arendajaid toetada, et pakkuda eelnevalt võimatuna näivat funktsionaalsust, näiteks A/B katsetamist.

Kasutatud kirjandus

[1] Eclipse Foundation, “AdoptOpenJDK – Open source, prebuilt JDK binaries”. [Võrgumaterjal]. https://adoptopenjdk.net/ [Kasutatud 12 märts 2022].

[2] VMware Inc, “9. System Requirements”. [Võrgumaterjal]. https://docs.spring.io/spring-boot/docs/2.0.0.RELEASE/reference/html/getting-started-system-requirements.html [Kasutatud 12 märts 2022].

[3] The Linux Foundation, “What is Kubernetes?”. [Võrgumaterjal]. https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ [Kasutatud 12 märts 2022].

[4] Gartner, Inc., “Revenue Growth for Container Management Software”. [Võrgumaterjal]. https://www.gartner.com/en/newsroom/press-releases/2020-06-25-gartner-forecasts-strong-revenue-growth-for-global-co [Kasutatud 12 märts 2022]

[5] The Linux Foundation, “Install Tools | Kubernetes”. [Võrgumaterjal]. https://kubernetes.io/docs/tasks/tools/#kubectl [Kasutatud 15 aprill 2022]

[6] The Linux Foundation, “Kubernetes Object Management | Kubernetes”. [Võrgumaterjal]. https://kubernetes.io/docs/concepts/overview/working-with-objects/object-management/#management-techniques [Kasutatud 15 aprill 2022]

[7] The Linux Foundation, “Creating a cluster with kubeadm”. [Võrgumaterjal]. https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/ [Kasutatud 24 märts 2022]

[8] Docker Inc., “Docker Registry | Docker Documentation”. [Võrgumaterjal]. https://docs.docker.com/registry/ [Kasutatud 2 aprill 2022]

[9] The Linux Foundation, “Graduated and incubating projects | Cloud Native Computation Foundation”. [Võrgumaterjal]. https://www.cncf.io/projects/ [kasutatud 2 aprill 2022]

[10] The Linux Foundation, “Harbor”. [Võrgumaterjal]. https://goharbor.io/ [Kasutatud 2 aprill 2022]

[11] IOActive Inc., “Low-hanging Secrets in Docker Hub and a Tool to Catch Them All | Matías Sequeira | IOActive”. [Võrgumatejral]. https://ioactive.com/guest-blog-docker-hub-scanner-matias-sequeira/ [9 aprill 2022]

[12] The Linux Foundation, “Jenkins User Documentation”. [Võrgumaterjal]. https://www.jenkins.io/doc/ [Kasutatud 11 aprill 2022]

[13] Istio Authors, “Istio”. [Võrgumaterjal]. https://istio.io/latest/ [Kasutatud 12 aprill 2022]

[14] Istio Authors, “Istio / Canary Deployments using Istio”. https://istio.io/latest/blog/2017/0.1-canary/ [Kasutatud 12 aprill 2022]

[15] The Linux Foundation, “Managing Resources | Kubernetes”. [Võrgumaterjal]. https://kubernetes.io/docs/concepts/cluster-administration/manage-deployment/#canary-deployments [Kasutatud 12 aprill 2022]

[16] Istio Authors, “Istio / Virtual Service”. [Võrgumaterjal]. https://istio.io/latest/docs/reference/config/networking/virtual-service/ [Kasutatud 14 aprill 2022]

[17] Istio Authors, “Istio / Gateway”. [Võrgumaterjal]. https://istio.io/latest/docs/reference/config/networking/gateway/ [Kasutatud 14 aprill 2022]

[18] Istio Authors, “Istio / Destination Rule”. [Võrgumaterjal]. https://istio.io/latest/docs/reference/config/networking/destination-rule/ [Kasutatud 14 aprill 2022]

[19] Smithsonian Institute, “COBOL: Introduction | National Museum of American History”. [Võrgumaterjal]. https://americanhistory.si.edu/cobol/introduction [Kasutatud 16 aprill 2022]

[20] Micro Focus, “COBOL Market Shown to be Three Times Larger than Previously Estimated in New Independent Survey | Micro Focus”. [Võrgumaterjal]. https://www.microfocus.com/en-us/press-room/press-releases/2022/cobol-market-shown-to-be-three-times-larger-than-previously-estimated-in-new-independent-survey [Kasutatud 16 aprill 2022]

[21] The New Stack, “COBOL Is Everywhere, Who Will Maintain It? - The New Stack”. [Võrgumaterjal]. https://thenewstack.io/cobol-everywhere-will-maintain/ [Kasutatud 16 aprill 2022]

[22] Percona LLC, “pt-online-schema-change”. [Võrgumaterjal]. https://www.percona.com/doc/percona-toolkit/3.0/pt-online-schema-change.html [Kasutatud 16 aprill 2022]

[23] Percona LLC, “PT Online Schema Change for Postgres? - PostgreSQL / PostgreSQL General Discussions – Percona Community Forums”. [Võrgumaterjal]. https://forums.percona.com/t/pt-online-schema-change-for-postgres/10081/2 [Kasutatud 16 aprill 2022]

[24] The Linux Foundation, “Operating etcd clusters for Kubernetes | Kubernetes”. [Võrgumaterjal]. https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/ [Kasutatud 16 aprill 2022]

[25] Judd Vinet, Aaron Griffin ja Levente Polyák, “System maintenance”. [Võrgumaterjal]. https://wiki.archlinux.org/title/System_maintenance#Partial_upgrades_are_unsupported [Kasutatud 20 aprill 2022]

[26] The Linux Foundation, “Safely Drain a Node | Kubernetes”. [Võrgumaterjal]. https://kubernetes.io/docs/tasks/administer-cluster/safely-drain-node/ [Kasutatud 21 aprill 2022]

[27] Red Hat, Inc., “Red Hat – We make open source technologies for the enterprise”. [Võrgumaterjal]. https://www.redhat.com/en [Kasutatud 21 aprill 2022]

[28] Rocky Enterprise Software Foundation, “Rocky Linux”. [Võrgumaterjal]. https://rockylinux.org/ [Kasutatud 21 aprill 2022]

[29] SUSE, “SUSE – Open Source Solutions for Enterprise Servers & Cloud”. [Võrgumaterjal]. https://www.suse.com/ [Kasutatud 21 aprill 2022]

[30] Buoyant, Inc., “The world’s lightest, fastest service mesh. | Linkerd”. [Võrgumaterjal]. https://linkerd.io/ [Kasutatud 13 mai 2022]

[31] HashiCorp Inc., “Consul by HashiCorp”. [Võrgumaterjal]. https://www.consul.io/ [Kasutatud 13 mai 2022]

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

Mina, Stefan Artur Adov

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebirakenduse katkematu uuendamine Kubernetesega" mille juhendaja on Edmund Laugasson.

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.

13.05.2022

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.