TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Arnold Milihhin 194015IAAB

Saladustehoidla käideldavuse tõstmine Riigi Infosüsteemi Ameti näitel

Bakalaureusetöö

Juhendaja: Siim Vene
MSc, Tallinna Tehnikaülikool

Kaasjuhendaja: Priit Parmakson
MSc, Tallinna Tehnikaülikool

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: Arnold Milihhin

16.05.2022

Annotatsioon

Käesoleva bakalaureusetöö raames antakse ülevaade olemasolevast tugiteenusest, selle tööst, mille käigus kirjeldatakse ühtlasi praeguse lahenduse puudusi. Töö eesmärgiks on tagada kõrgkäideldavus läbi nuripunktide maandamise saladustehoidla rakenduse kontekstis.

Töö kolmandas peatükis kirjeldatakse tööle seatud nõudeid ja piiranguid.

Neljandas peatükis viiakse läbi võrdlus ning analüüs, mille käigus selgitatakse välja sobilik tagasüsteem ning parim võimalik arhitektuuriline lahendus nuripunktide maandamiseks. Analüüsi käigus antakse hinnang igale kandidaadile mitmete kriteeriumite põhjal ning lõppjärelduse põhjal luuakse arhitektuurikavand ning tõenduskontseptsioon.

Sellele järgneb töö teostamine, kus kaardistatakse vajaminevad komponendid ning juurutatakse lahendus võttes arvesse tööle seatud nõudeid ja piiranguid.

Kuuendas peatükis teostatakse testeksperiment valideerimaks autori loodud lahenduse vastavust lähtetingimustele.

Töö viimases peatükis antakse ülevaade tulemustest ning võetakse kokku, kas juurutatud lahendus vastab nõuetele ja lahendab püstitatud probleemi. Tuuakse välja ka edasiarenduse ideed ning kliendi tagasiside.

Lisades on kättesaadav töö käigus loodud tehniline paigaldusjuhend ning tõenduskontseptsioon.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 9 peatükki, 15 joonist, 2 tabelit, 1 koodinäide.

Abstract

Improving Secret Management Service Availability on the Example of Information System Authority

The aim of the current thesis is to improve availability of secret management service through the ellimination of single point failures from the end-to-end chain. The necessity of the paper derives from risk of high available loss of services that depend on mission-critical underlying components such as secret management service.

First methodology emphasis is to conduct extensive research of highly available persistent backend storage through comparative analysis which is based on a qualitative and quantitative methodology. The criteria evaluation is summarized with decision matrix and requirements met backend is chosen.

During designing the feasible and highly available arhitecture, the proof of concept is being implemented, which is followed by a try-and-test phase.

Use case based test cases helps to cover the entire system and to determine meet complience of the requirements. In the last chapter the author of this paper sums up whether the implemented solution meets the criterias and solves the problem.

The thesis is in Estonian and contains 41 pages of text, 9 chapters, 15 figures, 2 tables, 1 code example.

Lühendite ja mõistete sõnastik

ACL	Ingl.k *Access-Control List*, pääsuloend.
API	Ingl.k *Application Programming Interface*, rakendusliides.
CAP	Ingl.k *Consistency, Availabilty, Partition Tolerance*, akronüüm, mille tähtedele vastavad sõnad: terviklus, käideldavus, jaotustaluvus.
DCS	Ingl. k *Distributed Consensus Store; hajutatud konsensuse hoidla* – hajutatud sõlmedevahelise konsensuse tagamiseks vajalike konfiguratsioonide ja metaandmete hoidla.
Eesrakendus	Ingl.k *Front-end*, kasutajale nähtav liides, mis võimaldab suhelda tagasüsteemiga.
Elutukse	Ingl.k *Health Check*, komponendi või teenuse töövalmiduse välja selgitamiseks tehtav kontrollpäring.
End-to-end	Otspunkt kliendi eesrakenduse otspunktist kuni tagasüsteemini kogu teekonna algusest lõpuni läbivalt.
Kasutaja	Ingl. k *User*, kasutaja on süsteemi teenus(t)e kasutaja. Kasutaja võib olla nii inimene kui ka masin (teine süsteem). Süsteemil on tavaliselt mitmeid kasutajaid; need võivad süsteemi kasutada üheaegselt.
Klaster	Ingl.k *Cluster*, arvutite või teenuste kogumi moodustuv kobar, mis töötavad tihedalt koos võimaldades neid vaadelda ühe tervikuna.
Mastaabitavus	Ingl.k *Scalability*, süsteemi omadus tulla ühtviisi toime väga suurte või väga väikeste koormustega, ilma et ta tõhusus– ja kvaliteedinäitajad oluliselt muutuksid; tavaliselt hõlmab ka laiendatavust.
NSPoF	Ingl.k *No Single Points of Failure*, süsteem või komponentidest koosnev komplekt, kus puuduvad nurpunktid.
Nuripunkt	Ingl.k *Single Point of Failure*, süsteemielement või komponent süsteemis, mille tõrge halvab kogu süsteemi.
PKI	Ingl.k *Public Key Infrastructure*; avaliku võtme taristu – IT vahendite, inimeste, poliitikate ja protseduuride süsteem avalike võtmete sidumiseks kasutajate identiteetidega, tavaliselt digitaalsertifikaatide abil.
Round-robin	Koormusejaotamise algoritm, milles tööd jagatakse kobara liikmetele vaheldumisi, nn karuselli põhimõttel.
S2S	Ingl.k *System-to-System*, kasutajatest sõltumata, virtuaalserverite ja rakenduste või süsteemide vahelised.

Service Discovery	Teenustpakkuva komponendi kobara liikme dünaamiline ülesleidmine teenuse tarbija või süsteemi orkestraatori poolt.
Split-brain	Olukord, kus mitmest eksemplarist koosnev süsteem ei suuda veaolukorras otsustada klastri õiget ning terviklikku seisukorda võides põhjustada andmekadu või anomaaliaid.
SSL offload	Sissetuleva, TLS-iga (SSL-iga) turvatud päringu edasisuunamine ilma TLS-ita või sisemise TLS seansiga.
Sticky session	Tehniline võte, millega ühelt kliendilt saabuvad päringud suunatakse samasse kobara liikmesse.
Switchover	Sama, mis tõrkesiire aga käsitsi.
Tagasüsteem	Ingl.k *Back-end*, kasutajale nähtamatu süsteemi tagakomponent, mille ülesandeks on toetada eesteenuseid.
Tõrkesiire	Ingl k. *Failover*; *avarii-ümberlülitus* – käideldavust suurendav automatiseeritud või käsiprotsess elastsete IT-teenuste ümberlülituseks.
UI	Ingl.k *User Interface*, kasutaja ja arvutiprogrammi vaheline ühenduslüli.
Võrkteenus	Ingl.k *Service Mesh*, võrkstruktuurina töötavate, dubleeritud sõlmede või komponentide kogumina osutatud teenus.
WAF/IDS	Ingl.k *Web Application Firewall/Web Intrusion Detection System*, meetmed sissetungi tuvastamiseks ja maandamiseks.

Sisukord

Jooniste loetelu

Tabelite loetelu

Koodinäidete loetelu

1 Sissejuhatus

Digiteenused kujundavad suure osa meie tänapäeva elust ning ühiskonnal ja kodanikel ootused toimivuse suhtes kõrged. Riigi Infosüsteemi Ameti (edaspidi RIA) strateegia on tagada 24/7 turvaliselt toimiv digiriik [1], mille käideldavus sõltub missioonikriitilistest aluskomponentidest nagu on seda saladustehoidla.

Käesoleva bakalaureusetöö raames aitab autor välja selgitada otstarbekaim lahendus kõrgkäideldavuse tagamiseks nimetatud tugiteenuse kontekstis. Sobiva lahenduse väljaselgitamiseks viiakse läbi tehniline analüüs, mille käigus hinnatakse sobivad tehnoloogiaid mitmete kriteeriumite põhjal ning võetakse kokku hinnangut peegeldava otsustusmaatriksiga.

Tehnilistele nõuetele ja kriteeriumitele vastav lahend aga juurutatakse ning viiakse läbi testeksperiment valideerimaks, et lahend lahendab probleemi, vastab ootustele ja nõuetele. Teostamise käigus loodud paigaldusjuhendiga saab tutvuda lisades ning tõenduskontseptsiooniga teostamise lõpufaasis.

Käesolevas töös viidi ka läbi intervjuu RIA arhitektidega, saamaks vajaliku teavet sisendiks analüüsile. Kuvatõmmistel ja paigaldusjuhendis olevate virtuaalserverite nimed ning aadressid on asendatud kujutlike vastetega ning esitatud pseudonüümidega, mis ei vasta organisatsioonis olevaga.

Viimases peatükis sõnastatakse töö tulemused, esitatakse edasiarenduse ettepanekud ning võetakse kokku tööle antud tagasiside.

2 Probleemi püstitus ja eesmärk

Käesolevas peatükis kirjeldatakse teema aktuaalsust ja tausta. Tuuakse välja lahendatav probleem ning kirjeldatakse probleemi uurimiseks ja lahendamiseks valitud metoodikaid. Viimaks tuuakse välja töö eesmärgid.

2.1 Aktuaalsus

Ühiskonna avalikkusel ja riigi kodanikel on väga kõrged ootused riigi digiteenuste käideldavusele. Digiteenused töötavad aga tugi- ja äriteenuste kooslusel, kus terviku käideldavus sõltub aluskomponentide käideldavusest. Sellest tulenevalt on kõrgkäideldavus aktuaalne kõigis tugiteenustes.

Antud töö võib olla ajendiks avaliku sektori ettevõtetele ja organisatsioonidele tugiteenuste kvaliteedi parandamiseks ning abistada selle protsessi läbiviimisel.

2.2 Taust ja hetkeolukord

Riigi Infosüsteemi Amet viib ellu agiilset strateegiat erinevatel tasanditel – alates DevOps praktikate rakendamisest osakondades või protsessides kuni agiilse mõtteviisi lähtumiseni organisatsiooni tegevustes.

Agiilsuse printsiipe rakendavad eelkõige tootemeeskonnad, mille valguses püütakse pakkuda nii äriklientidele kui ka riigi kodanikele lisandväärtust, arendades kvaliteetseid e–teenuseid järgides DevOps praktikaid – jõuda võimalikult kiiresti muudatustega toodangukeskkonda.

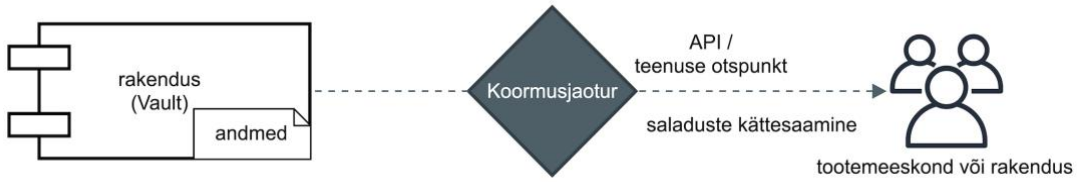
RIA tugiteenuste platvorm soodustab ja hõlbustab meeskondadel iseorganiseeruda ning optimeerida töö voogu rakenduse elutsükli vältel. Tulenevalt tootemeeskondade pidevast suurenemisest ning arendajate tarnetsüklite sageduse kasvust, muutuvad *end-to-end* tugiteenuste sõlmede kättesaadavus järjest ärikriitilisemaks.

Organisatsiooni saladusi hoiustatakse väljaspool teenuste seadistusi ning rakendusi keskses saladustehoidlas, mis on üks tugiteenuste platvormi olevatest teenustest, mille kasutajad on nii tootemeeskonnad, haldurid kui ka erinevad teenuste rakendused.

RIA kasutab vabavaralist identiteedipõhise turvapoliitikal põhinevat saladuste ja krüptimise haldussüsteemi Hashicorp Vault, mis pakub järgnevat funktsionaalsust:

- Saladuste genereerimine
- Rotatsioon
- Sertifikaatide haldamine
- Krüpteerimine
- Kehtivate reeglistike auditeerimine

Täna töötab terviklahendus ühel ainsal virtuaalserveril, kus saladused on talletatud failisüsteemi (joonis 1). Teenuse või teenustpakkuva virtuaalserveri katki minemisel võivad kaduma minna andmed, mis võib tingida vajaduse andmehoidla varukoopiast taastada, kuid terviksüsteemi funktsioonide taastamine niisugustel juhtudel võib nõuda palju aega. Samuti tuleb arvestada riskiga, et kaduma võivad minna pärast viimast varundamist tehtud muudatused.



Joonis 1. Arhitektuuriskeem „As is“.

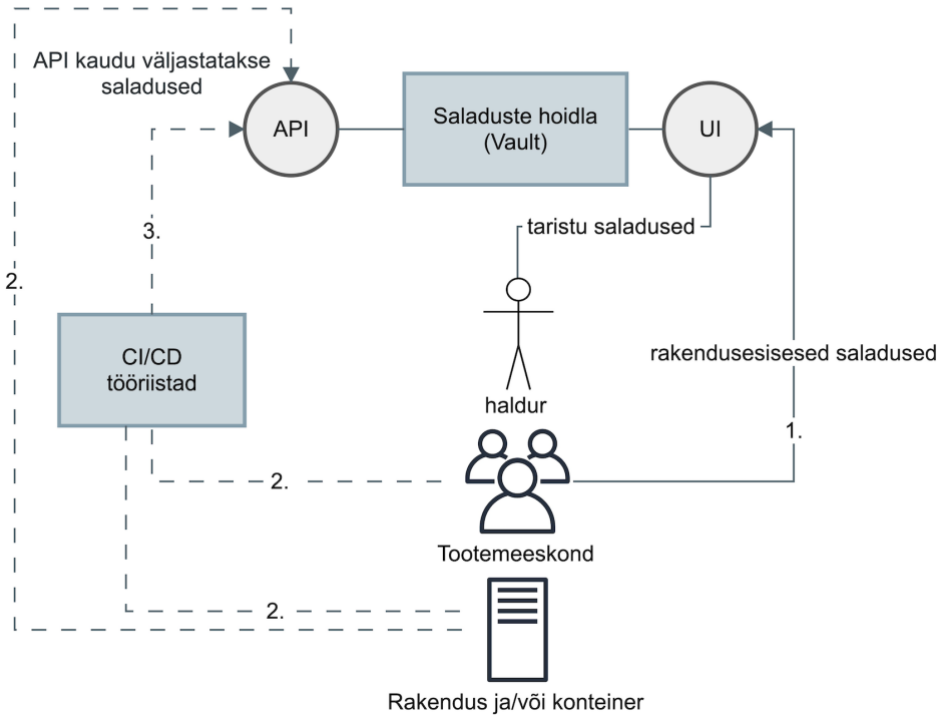
Saladuseks peetakse mistahes sõnekomplekti, mis võimaldab juurdepääsu erinevatele teenustele, rakendustele ning süsteemidele:

- API võtmed
- SSH võtmed
- S2S paroolid
- Sertifikaadid
- Krüptovõtmed
- Ühekordsed paroolid
- Kasutajate paroolid

Saladustehoidla kontekstis on organisatsioon kasutusele võtnud keskserveri arhitektuuri järgnevatel põhjustel:

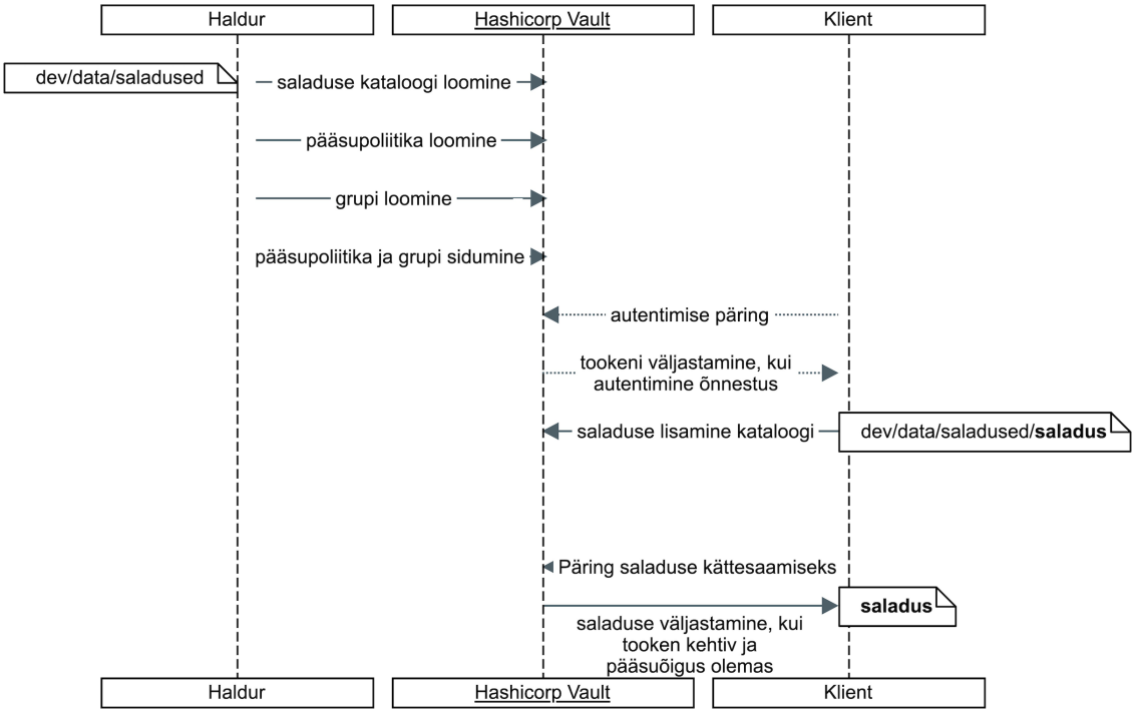
- Jagatud saladus
- Ühekordsed saladused
- Auditeerimine

Jagatud saladuse all mõeldakse seda, et ühte saladust võib kasutada n-kasutajat. Ühekordne saladus täidab mingit eesmärk, kuid saladuse sisust ei pruugi kasutaja teada, sest see genereeritakse pimesi evitusprotsessi hetkel, näiteks robot kasutaja poolt. Auditeerimine annab ülevaate kõikidest autenditud tegevustest, mis omakorda võimaldab tagasiulatavat jälitatavust.



Joonis 2. Voodiagramm – Saladustehoidla kasutamise voog.

Saladus luuakse vastavalt nõuetele, mis lisatakse kasutaja poolt eesrakenduse veebiliidese kaudu (joonis 2). Igal meeskonnal oma vastav projekti kataloogi, millega tagatakse andmete konfidentsiaalsus [2]. Saladuste kättesaamiseks kasutatakse Rakenduse HTTP API otspunkti, mis võtab vastu kliendi päringu ning tagastab kliendile vastu saladuse [3]. API operatsioonide eelduseks on kliendi identsustõend (*identity token*) olemasolu, mis tagab pääsuõiguse vastava projektikataloogi teabele [3] (joonis 3).



Joonis 3. Järgnevusskeem – Saladustehoidla kasutamine.

2.3 Probleemi püstitus

Täna kasutusel olevas saladustehoidla teenust pakkuval rakendusel on käideldavuskao risk, millest tingituna võib süsteemi avarii korral saladuste kättesaadavus volitatud subjektidele olla piiratud ning IT süsteemide pideva juurutamise ahel häiritud.

Kuna konkreetses tugiteenuses hoiustatakse ärikriitilise tähtsusega andmeid ning kuna andmete olulisim omadus on käideldavus [4], on seejuures oluline tagada tugiteenuse kättesaadavus saladustehoidla kasutaja vaatepunktist.

Kõrgkäideldav saladustehoidla on kriitiliselt vajalik ka stsenaariumis, kus isegi üle andmekeskuste paigaldatud teenused võivad langeda erakorralise olukorra ohvriks, mis langetab rivist välja ja tingib vajaduse uuesti evitada. Evitamise käigus aga kasutatakse saladustehoidlat.

2.4 Eesmärk

RIA soov on saavutada olukord, kus käsitletav tugiteenus on hallatud kui kõrgkäideldav teenus, tingimusel, et sellest teenusest sõltuvate infosüsteemide käideldavus oleks tagatud [5].

Käesoleva lõputöö eesmärgid on:

- Võrrelda rakenduse tagasüsteeme.
- Luua NSPoF lahendus.

Võimalike lahenduste välja töötamisel võetakse aluseks tööle esitatud nõuded. RIA kontekstis on eeldus, et võrgu– ja virtualiseerimiskihtide tasemel on käideldavus juba tagatud. On oluline, et tulemus ei tõstaks halduskulusid.

2.5 Metoodika

Käesolevas peatükis antakse ülevaade kasutatavatest metoodikatest, mis aitavad autoril viia lahendus vastavusse lähtetingimustega.

Valitud metoodikateks on:

- Võrdlev analüüs
- Intervjuu
- Tõenduskontseptsioon
- Eksperiment
- Testimine

Võrdlev analüüs põhineb kvalitatiivsel ja kvantitatiivsel metoodikal. Autor on välja tulnud kriteeriumitega, mis võetakse aluseks analüüsi teostamisel. Esmalt uuritakse ja kogutakse andmeid selle kohta, kui palju mingi omadus uuritaval objektil esineb ning seejärel antakse numbriline arvamus arvulisel skaalal 0, 0,5 või 1 punkt iga omaduse kohta. Lisaks on igale kriteeriumile omistatud prioriteet ehk kordaja, mis korrutatakse omaduse hinnanguga. Prioriteetsem kriteeriumitulemus korrutatakse kordaja kolmega ning vähem prioriteetsed kahanevas järjekorras.

Analüüsi käigus viiakse läbi intervjuu avatud ning struktureerimata vormis RIA arhitektidega, kellelt saadakse vajaliku teavet sisendiks analüüsile. Võrdleva analüüsi tulemus võetakse kokku otsustusmaatriksiga, kus summeeritud hinnangute põhjal selgub nõuetele vastav ning RIA konteksti sobilikem kandidaat.

Selle järgneb arhitektuurilise kavandi koostamine, läbivaatus ning kavandi tutvustamine RIA võtmeisikutele. Koostatud arhitektuurilise kavandi ja tehtud otsused võeti aluseks tõendikontseptsiooni loomisel. Viimaks juurutas autor lahenduse.

Testimisel kasutatakse kasutuslugude-põhist (*use case based testing*) lähenemist, mis peegeldavad süsteemi erinevaid avarii-stsenaariumeid. Testilood on vahend süstemaatilise ja korratava testimise saavutamiseks [6]. Testilood loovad väärtust haldurile, kes saab läbida süsteemi voo testilugude põhjal, et veenduda süsteemi toimivuses ning vajadusel neid tulevikus taaskasutada. Testilugude ja testiformaat on autori enda valitud.

Sisend testide loomisel on lahendatavat probleemi kirjeldav paigaldusjuhis ehk dokumentatsioon ning tõenduskontseptsioon ehk arhitektuurimudel.

3 Lähtetingimused

Käesolevas peatükis tuuakse välja teostatavale lahendusele esitatud tehnoloogilised nõuded. Lisaks esitatud nõuetele tuuakse välja ka lisapiirangud, mis kehtivad antud projekti raames.

3.1 Funktsionaalsed nõuded

1. Tagasüsteemi ühilduvus rakendusega.
2. Komponentidevaheline autentimine.
3. Komponentidevaheline krüpteeritud kommunikatsioon.
4. Andmebaasi ja rakenduse eraldamine ja jagamine eraldi paigaldavaks liideskomponendiks.
5. Paigaldatavus hajusalt, mitmesse andmekeskusesse.

3.2 Piirangud

1. Kõrgkäideldav lahendus ei tõsta halduskulusid.
2. Kõrgkäideldav lahendus ei vähenda praeguse rakenduse kasutatavust ärimeeskondade vaates.

4 Tehniline analüüs

Käesoleva peatüki eesmärk on viia läbi analüüs leidmaks otstarbekaim lahendus probleemi lahendamiseks arvestades arendusele esitatud nõudeid ning piiranguid.

Hashicorp Vault toetab üle kahekümne erineva tagasüsteemi ühildamist, millest pooled toetavad kõrgkäideldavuse tagamist klasterlahendusel [2].

Käesolevas peatükis võrreldakse ning leitakse sobiv andmehoidla tüüp ning seejärel võrreldakse kolme järgnevalt nimetatud andmehoidlat:

- Consul
- PostgreSQL
- Etcd

Valik on tingitud sellest, et andmehoidlad on isemajutatavad, hästi dokumenteeritud ning avatud lähtekoodiga. PostgreSQL ning Etcd on laialt tuntud, mistõttu autori arvates on nende haldamiseks vajalikud teadmised ning kompetentsid olemas, ning ka organisatsiooni potentsiaalsetel süsteemihaldajatel võrreldes teiste vähem tuntud tagasüsteemidega. Teadaolevalt on Etcd ning PostgreSQL kokkupuude vähemalt kolmel meeskonnal ning kasutusel mitmel teenusel.

Consuli osas puudub organisatsioonil teadmine ja kompetents, kuid et tegemist on HashiCorp tootega, on võimalik lahendus lihtsasti ühilduv saladustehoidlaga ning arvestades asjaolu, et tegemist on kaasaegse mikroteenustele orienteeritud arhitektuuriga, võib lahendus olla sobilik.

Valitud tagasüsteeme võrreldakse järgmiste omaduste alusel:

- Halduskeerukus
- Turvalisus
- Süsteemi omadused
- Mastaabitavus
- Kõrgkäideldavus

Nimetatud omaduste põhjal koostatakse lõppjäreldus, mis aitab valida välja sobiva lahenduse juurutamiseks.

Käesolevas töös ei käsitleta varundust ega andmetaastet.

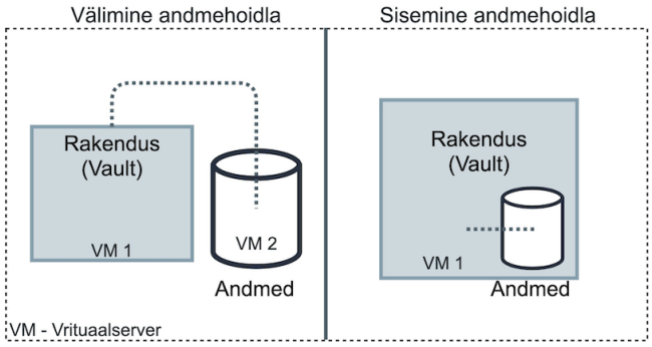
4.1 Tagasüsteemi tüübid

Käesolevas peatükis võrreldakse kahte andmehoidla tüüpi, mis ühildub saladustehoidla rakendusega.

Vault toetab kahte järgnevalt nimetatud andmehoidla tüüpi [2]:

- Sisemine andmehoidla (integrated storage)
- Väline andmehoidla (external storage)

Nimetatud andmehoidla tüüpide peamine erinevus on see, et andmeid talletatakse erinevates virtuaalserverites [2] (joonis 4). Kui sisemise andmehoidla puhul hoitakse andmeid Vault teenust osutavas virtuaalserveri failisüsteemis, siis välise andmehoidla puhul virtuaalserveriga eraldatud andmehoidlas [2]. Välise andmehoidla valikul tuleks arvestada lisaressursi eraldamisega ning võrgusõlme lisandumisel lisahüppega (*hop*) marsruutimisel [2].



Joonis 4. Sisemine ja välimine andmehoidla.

Vaatamata sellele, et HashiCorp soovitab [2] kasutada sisemist andmehoidlat (*integrated storage*), ei ole antud tehnoloogia valik sobiv RIA konteksti arvestades 3.1 nõuet 4. eraldada ja jagada andmebaas ja rakendus kaheks eraldi paigaldavaks liideskomponendiks.

Selliselt monoliitarhitektuuri lõhkumine pakub paremat turvalisust, käideldavust, mastaabitavust ning taaskasutatavust [7]. Mikroteenuste arhitektuuri järgimisega hoitakse ära ebavajalike kihtide seosed, sest probleemi korral võib olla keeruline tuvastada allikas.

Näiteks komponendi mälu puudusest (*OOM killer*) võib tuleneva teenuse katkestuse juurprobleemi tuvastamine olla keeruline ja ajakulukas, kui konkreetses virtuaalserveris jooksevad mitmed teenused ning potentsiaalseid intsidendi põhjustajaid võib olla mitu. Samuti olukorras, kus ründaja on saavutanud kontrolli mingi komponendi üle, on tal võimalik tegutseda ainult konkreetse komponendi või konteineri piires. Selline isoleeritus hoiab ära komponentide kaudu ligipääsetavuse teistele keskkondadele eeldusel, et järgitud on turvapoliitika ja turbenõudeid.

Sellest järeldame, et RIA kontekstis on sobivam võtta kasutusele väline andmehoidla (*external storage*). Samuti ei ole seatud RIA poolt ressursikasutuse piiranguid, mis võimaldab andmehoidlat paigaldada eraldiseisvate komponentidena.

4.2 Tagasüsteemid

Consul on võrkteenus (*service mesh*) haldusliides ning tööriist taristu haldusel, mis liidab ühtse halduse alla ühe süsteemi erinevad osad [8]. Tegemist on võimeka tööriistaga, mis lisaks võtmeväärtuste andmete talletamisele võimaldab ka võrgu segmenteerimist ning marsruutimist agentide vahel, mis omakorda hõlbustab tõrgete tuvastamist [2], [8].

Consul on platvormi agnostiline, mis teeb ta ühilduvaks paljude erinevate platvormidega, näiteks Kubernetes ning Hashicorp Nomad [8].

Klastri moodustavad agendid, mille ülesandeks on komponentide vahelise suhtluse vahendamine ja moodustamine. Võrgustikuavastamise (*service discovery*) funktsioon tuvastab paigaldatud agendid ning lisab need omakorda automaatselt kogumisse (Gossip pool), mis omakorda moodustab teenuste võrgustiku (*service mesh*) [9]. Consuli oskuslik kasutamine aitab ära hoida manuaalsed konfiguratsioonitööd, hõlbustab ja lihtsustab tõrgete tuvastamist [9].

Etcd on avatud lähtekoodiga, kergekaaluline, võtmeväärtuste ja konfiguratsioonide andmehoidla, mis pakub teenuste leidmise funktsionaalsust (*service discovery*), et klastri liikmed saaksid üksteist avastada ning mis aitab ka koordineerida klastris hajutatud komponentide vahelist ajastatud tööd [10].

Tegemist on paljude teiste projektide põhikomponendina, näiteks *de facto* standardsüsteem teada-tuntud konteinerite orkestreerimise haldusplatvormi Kubernetese primaarse andmehoidlana [10].

PostgreSQL on avatud lähtekoodiga relatsiooniline andmebaasihaldur objektorienteeritud kallakuga, mis olnud üle 30 aasta arenduses ning nii tehniliselt kui ka funktsionaalselt tõsine konkurent ka parimatele kommertsanaloogidele [11].

Tegemist on Postgres andmebaasi vabavaralise edasiarendusega, mille algus sai Berkeleys ülikoolist ning millel on taga suur globaalne kogukond arendajaid ja ettevõtteid [11].

Andmebaaside arengu ajaloolises kontekstis oli PostgreSQL üks esimesi andmebaase, mis pakkus MVCC-d (*multiversion concurrency control*) vastukaaluks rea- või tabelilukustusele, mis omakorda parandas oluliselt andmebaaside ühiskasutuse süsteemi ning tõstis selle efektiivsust [12].

Kõrgkäideldavuse juurutamiseks on PostgreSQL välja töötanud terviklahenduse Patroni klastrite juurutamise ja hoolduse kohandamiseks ning automatiseerimiseks. Kitsamas mõistes on Patroni komplekt Pythoni skripte ning teeke, millega juhitakse klastriga seotud protsesse. Tegemist on üheotstarbelise lahendusega ja välja töötatud ainult PostgreSQL jaoks. Patroni on pälvinud tugeva maine oma töökindla arhitektuuri, terviklikkuse ning robustse funktsionaalsusepagasiga – klasterlahendused on tõrkekindlad, mastabeeritavad nii suure hulga andmete kui ka suure hulga samaaegsete kasutajate arvu poolest [13].

4.3 Tagasüsteemide võrdlus

Käesolevas peatükis võrreldakse valituks osutunud tagasüsteemi kandidaate.

1975. aastal avaldati esmakordselt kahe kindrali probleem, kirjutises „*Some Constraints and Trade-offs in the Design of Network*“ [14], mis kirjeldab hajussüsteemide väljakutset, et osapoolte omavahelise konsensuse ja usaldusväärsuse saavutamine on keeruline olukorras, kus süsteemi osapoolte arv suureneb.

Komponentide hajutamine on kõrgkäideldavuse eelduseks [15] ja on oluline vaadata otsa mehhanismidele, mis tagavad klastri sõlmedevahelise konsensuse ja usaldusväärsuse

ning turvalisuse. Lähtudes kindralite probleemist, teostatakse käesolevas töös võrdlus ka omaduste põhjal, mis tagavad turvalise, usaldusväärse ning konsistentse klasterarhitektuuri.

Kriteeriumite kandidaatideks on:

- Consul
- Etcd
- PostgreSQL (Patroni)

Iga alampeatüki kokkuvõttes antakse hinnang punktiskaalal 0, 0,5 või 1, mis peegeldab, kui palju vastab andmehoidla kriteeriumile. Iga võrdluse tulemus kajastatakse antud peatüki lõpus lõppjärelduse tabelis.

4.3.1 Halduskeerukus

Kõik nimetatud andmehoidlad on sobilikud Vaulti saladuste talletamiseks, kuid tähelepanu tuleks pöörata nimetatud tehnoloogiate poolt pakutavate lisaväärtustele ning mis ressurss on vajalik nende implementeerimiseks täiendustegevuste ja integratsioonide näol. Käesolevas peatükis vaadeldakse, mis halduskoormus kaasneb tehnoloogia juurutamisel, näiteks üleseadmise ja haldamise keerukus, mis tuleneb tehnilise dokumentatsiooni põhjal tehtavast hinnangust. Samuti tuleks autori hinnangul andmehoidla valikul tähelepanu ka pöörata avaliku võtme taristu (PKI) haldusele, sest selle seadistamine võib olla keeruline ning manuaalselt tehes võib osutuda ajaliselt mahukaks ja keerukaks protsessiks. Sertifikaatide halduse teema on RIA kontekstis aktuaalne, sest täna moodustatakse ja väljastatakse sertifikaate nii teenustele kui kasutajatele käsitsi.

Consul erineb võrdlemisi teistest andmehoidlatest selle poolest, et pakub lisaks võtmeväärtuste hoiustamisele mikroteenustele orienteeritud lisafunktsionaalsusega platvormi. Autor leiab, et Consul lahendab mitmed mikroteenuste ja hajutatud arhitektuuriga seonduvaid probleeme, mis võivad esile kerkida teenuste konfiguratsioonide sidususe keerukusega hilisemates staadiumites. Näiteks nõuab dünaamiliselt muutuv topoloogia nagu võrkteenuse (*service mesh*) mudel teenuste hajutatust ja horisontaalset mastabeerimist, vähendab Consuli sarnaste tööriistade implementeerimine staatilisi konfiguratsioone, näiteks teenuste elutuksed (*health check*).

Consuli seire ning tõrketuvastusmehhanismid pakuvad lisaväärtust, mis vajavad küll eraldi seadistamist, kuid vähendavad oluliselt koormust hilisemates staadiumites, kus haldur ei pea eraldi seiresüsteeme juurutama. Consul paistab silma oma platvormi agnostilisusega, mis vähendab oluliselt täiendustegevusi, näiteks pilvetehnoloogiatega integreerimisel. Ametlik dokumentatsioon kergesti loetav ja arusaadav.

Etcd on kergekaaluline ning teiste andmehoidlatega võrreldes on selle evitamine ja haldus märkimisväärselt lihtsam. Ka mitmed arendajad on arvanud: „*One of the best-designed tools precisely because of this simplicity*" [16], [17]. Ametlik dokumentatsioon on kergesti loetav ja mõistetav.

Patroni on võimekas tööriist, mis toob kaasa endaga hulga tööriistu klastri protsesside juhtimiseks, mis lihtsustavad kõrgkäideldava klastri juurutamist ning haldust, kuid ametlik dokumentatsioon on liiga detailirohke, millest tulenevalt on teabe leidmine ebamugav ja ajamahukas, ning näiteks dokumentatsioonis puudub teave klastri mastaabitavuse kohta.

PKI halduse lihtsustamise osas suudab Etcd ja Consul võrreledes Patroniga ise sertifikaate genereerida ja allkirjastada klastritööks [18]. HashiCorp võimaldab Consulit ja Vaulti põimida selliselt, et sertifikaatide ja krüptomaterjali roteerimine on teostatav dünaamiliselt Vault saladuste mootori kaudu [19]. Patroni klastri puhul tuleks vajalik krüptomaterjal ning sertifikaadid genereerida ise klastriväliselt [20].

Kokkuvõttes lihtsustavad Etcd ja Consul osaliselt sertifikaadimajandust, kus võrreldes Patronil tuleb avaliku võtme taristu rakendamiseks vajalik genereerida Patroni- väliselt, mis lisab halduskoormust. Autori hinnangul on Consul väga perspektiivikas *end-to-end* teenuse avastamise funktsiooni sisaldav klasterlahendus (*service discovery*), kuid antud töö ja halduskeerukuse kontekstis on liiga palju arhitektuuriliselt liikuvaid osi, mis vajavad eraldi halduskompetentsi ning aega seadistamiseks ning haldamiseks. Patroni dokumentatsioonis on olulise teabe puudus klastri mastaabitavuse ning geograafilise hajutamise osas. Patroni koosneb mitmetest komponentidest ning tööriistadest, mis vajavad lisaoskusi ja teadmisi klastri haldamiseks, mis teeb ta Etcd'ga võrreldes keerukamaks. Kuna RIAl on kompetents Patroni ja Etcd haldusel, on antud töö kontekstis need sobivamad kui Consul.

Autori hinnangul on Etcd aga sobivaim, sest hoiab juurutamist minimalistlikumana, dokumentatsioonist leiab vajaliku teavet, on selgesti ja lihtsasti mõistetav ning lisaks võimaldab Etcd osaliselt PKI halduse lihtsustamist. Autori hinnangul on Consuli ning Patroni juurutamine kuni kõrge raskusastmega ning Etcd kuni keskmise raskusastmega.

Hinnang:

- Consul: 0,5
- Patroni: 0,5
- Etcd: 1

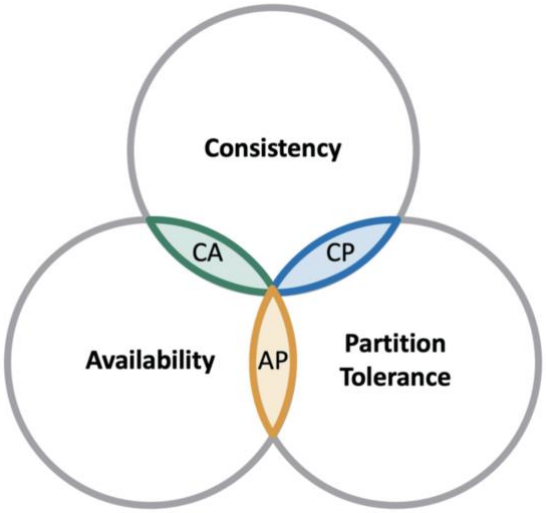
Tuleb arvestada, et teatud ulatuses on halduskoormus vältimatu nagu mainib ka Frederick P. Brooks oma kirjutises „*No Silver Bullet*“ – Juurutamisel tuleb arvesse võtta lisaks olemuslikule keerukusele (*essential complexity*) juhuslik keerukus (*accidential complexity*), teisisõnu, arvestada õpikõverat ning tööga kaasnevaid komplekssusi [21].

4.3.2 Süsteemsed omadused

Käesolevas peatükis vaadeldakse, milliseid süsteemseid omadusi pakuvad valitud andmehoidlad.

Süsteemsete omaduste all mõistetakse terviklikkust, käideldavust ja jaotavustaluvust. Arusaam omadustest on oluline, et paremini mõista hajutatud andmebaase, kuna saladused on kriitilise tähtsusega ning nimetatud omadused mõjutavad kliendi ja andmebaasi vahelist andmevahetust.

Hajusa andmebaasi korral tagada korraga kolme süsteemse nõude garantiid: terviklikkus, käideldavus ja jaotuvustaluvus on võimatu, ütles tuntud CAP teoreemi sõnastanud Eric Brewer [22]. Teoreemi seisukohalt saab olla samaaegselt tagatud maksimaalselt kaks omadust kolmest (joonis 5).



Joonis 5. Venni diagramm – CAP teoreem [23].

Consistency – Konsistensus või terviklikkus, kus kliendile loetavad andmete seis on alati viimane. Olukorras, kus mistahes sõlmes andmed muutuvad, kajastatakse ja ühtlustatakse see muudatus ka teistes sõlmedes [22].

Availability – Käideldavus tähendab seda, et pöördumine mistahes sõlme poole peab õnnestuma. Süsteem peab olema alati kättesaadav ükskõik, mis tõrke korral. Pöörduja peab saama igal ajahetkel andmeid kas lugeda või kirjutada [22].

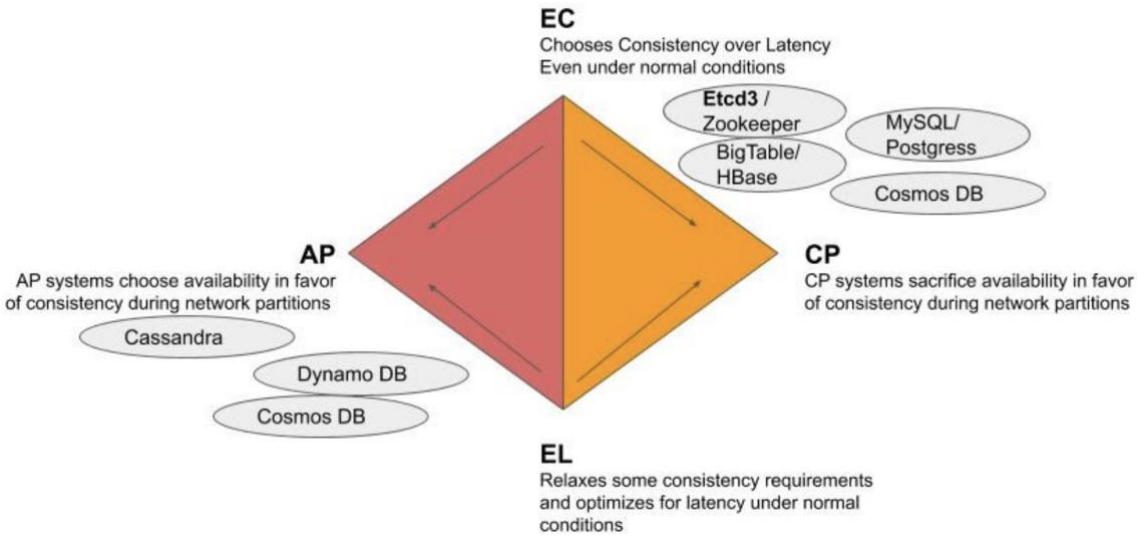
Parition Tolerance – Andmete jaotavustaluvus tähendab seda, et võrgukatkestuse tulemusena ei teki andmetes vigu, isegi kui sõlmedevaheline ühendus puudub [22].

Arvutivõrke ei saa alati pidada tõrkekindlaks, seega hajutatud süsteemide juurutamisel ei ole võimalik loobuda jaotavustaluvuse omadusest, sest olukorras, kus klastri komponentide vaheline kommunikatsioon läheb katki, võib see põhjustada andmerikkeid [22]. Sellest järeldame, et CAP teoreemi kohaselt võivad hajutatud süsteemide kavandamisel olla järgnevad süsteemiomadused:

1. AP (käideldavus + jaotavustaluvus)
2. CP (terviklikkus + jaotavustaluvus)

Teisisõnu, kas saada päringu vastusena kirje, mille seis ei pruugi olla õige, või saada vastuseks õige kirje, kuid kättesaadavuse hinnaga ehk viivitusega.

Aastaid hiljem tuli Daniel J. Abadi välja teesi täiendusega (joonis 6), mille kohaselt pakub teoreem hajussüsteemi omaduste täielikumat kirjeldamist ning mis ütleb, et kõrgkäideldavuse tagamine nõuab andmete replikatsiooni, mis omakorda kompromissi konsistentsuse ja järjepidevuse vahel [24].



Joonis 6. PACELC teoreem [25].

RIA kontekstis on oluline, et saladuste väljastamisel oleks saladuse seis alati viimane. Viimane seis tähendab, et juhtserver ei väljasta enne kasutajale vastust, kui on veendunud, et kõikide sõlmede vahel on seis sama. Selline ühtsuse tagamine vähendab läbilaskevõimet kirjutamisel, sest ühtse seisu valideerimine replikate vahel võtab aega. Praktikas võib see välja näha nii, et saladus väljastatakse kasutajale viivitusega. Samas on autori hinnangul pideva juurutamisel mõistlikum pigem oodata saladuse väljastamise taga kuni paar minutit, kui et kasutajale väljastatakse vale seis, mis omakord võib evitusprotsessi katki teha. Selle tulemusena jõuame järeldusele, et saladustehoidla tagasüsteem peaks olema tugeva konsistentsusega (*strong consistency*) ehk PACELC teesi kohaselt disainivalik (CP/EC).

Kõik nimetatud andmehoidlad toetavad tugeva konsistentsuse mudeli seadistust [26], [27], [28]. Tugev konsistentsus väljendub peamiselt selles, et enne kasutajale vastuse väljastamist veendutakse replikatevahelises andmete ühtsuses. Sama on näiteks Etcd näitel kasutaja tehtud kirjutamise või uuendamise päringud alati lineariseeritud, isoleeritud ja järjestatud [29], mille tulemusena liiguvad andmed alati kindlas järjekorras,

et ära hoida andmeoperatsioonide samaaegne kasutamine ehk andmete parallelism, mis omakorda võib põhjustada palju probleeme [30].

Kokkuvõttes on sünkroonimine hajusate süsteemide koostöötamise põhiraskus. Ei ole ühtegi hõbekuulilahendust, mis kõrvaldaks sünkroonimisprobleemi kõigi kasutusjuhtude puhul. Iga omadus lahendab probleemi erineval viisil või vähendab selle mõju teatud tingimustes. Saladuste replitseerimisel tuleks eelistada sünkroonset asünkroonsele, et vältida andmete terviklikkuse ning kao riski, sest andmete ühtsuse tagamine ei ole ajakriitiline.

Kõik valitud andmehoidlad on sobilikud, kui andmeid replitseeritakse sünkroonses režiimis ning käideldakse tugeva konsistentsuse mudeli järgi.

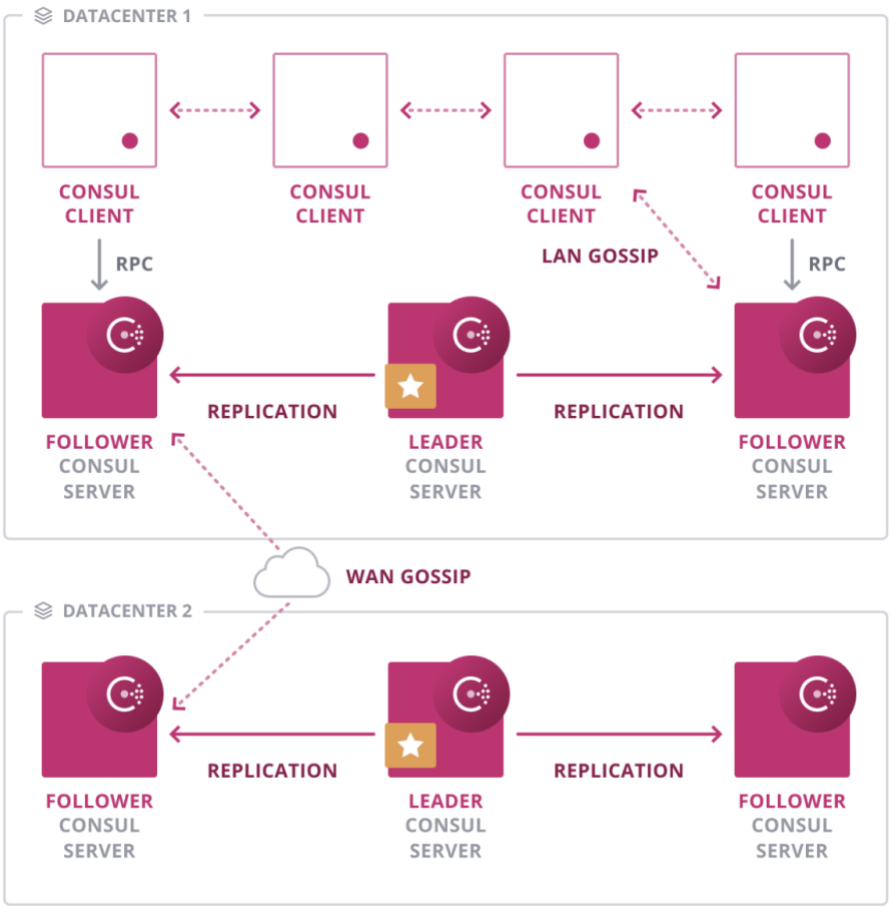
Hinnang:

- Consul: 1
- Patroni: 1
- Etcd: 1

4.3.3 Mastaabitavus

Käesolevas peatükis käsitletakse võimalusi ning piire klastri mastaabitavuse osas, juhul näiteks kui kliendil on soov jaotada klastrikomponente üle mitme andmekeskuse tagamaks paremat tõrkekindlust.

Consul võimaldab geograafilist liiasust (*geo redundancy*) ehk nii andmeid dubleerida kui ka hajutada komponente erinevate andmekeskuste vahel, mis võivad paikneda erinevates geolokatsioonides [9]. Eraldatud andmekeskuste ühendamiseks on võimalik kasutada nii Gossip protokolli laivõrgu kogumit (*WAN Gossip pool*) (joonis 7), kuhu on integreeritud ka funktsionaalsus tõrketuvastuseks ning võime tulla toime võrgukatkestustega, kui ka Consuli ühendavat lüüsi [31]. Agente on soovituslik juurutada ühe andmekeskuse kohta maksimaalselt 5000 [32]. Agendikogum võimaldab teha andmekeskuste üleseid (*cross-datacenter requests*) päringuid, kus iga päring edastatakse töötlemiseks juhtserverile, vaatamata, mis sõlme poole pöördumine toimub [9].



Joonis 7. Evitusskeem – Consul, andmekeskuste kokkuühendamine [33].

Etcd võimaldab samuti andmekeskustevahelist mastaabitavust, kuid mitmete andmekeskuste klastriliikmete vahelise konsensuse saavutamine võib tingida latentsusaja suurenemise, mis omakord võib põhjustada ka sagedasi klastri juhtserveri valimisi või elutuksete ajalõppe (*timeout*). Samuti tingib andmekeskuste vaheliste andmete replitseerimine võrgu läbilaskevõimet [34].

Patroni puhul on mastaabitavus mitme andmekeskuse vahel oluliselt keerulisem, sest VIP haldur (*virtual IP manager*) nõuab, et komponendid paikneksid ühes võrgu segmendis. Sellisel juhul on võimalik võrgusegmendi venitamise meetodil (*VLAN stretching*) üle mitme andmekeskuse [13], mida praktikas ei soovitata mitmel põhjusel. Teine võimalus on kasutada koormusjaotur HaProxy'it, et jaotada ühendusi üle interneti laiali, kuid sellisel juhul tuleb ka seda klasterdada ning kuidas teha Vaultile selgeks klastri sõlmede IP-aadressid. Üks võimalus on kasutada *keepalived* teenust, et luua Virtuaalne IP-aadress.

Kokkuvõttes sobivad antud mastaabitavuse kontekstis kõik andmehoidla valikud, sest horisontaalseks mastabeerimiseks piisab kolmest kuni viiest komponendist käideldavuse tagamiseks. Geograafiliselt on aga kõige mastabeeritavamad Consul ning Etcd, sest mastaabitavuseks ei pea kaasama koormusjaoture ega muid komponente. Kui klient soovib horisontaalselt mastabeerida rohkem kui viis virtuaalserverit, tuleks tehnoloogia valikutele uuesti otsa vaadata lähtuvalt jõudluse aspektist, sest vastavalt sõlmede arvu kasvule võivad tekkida jõudluse ning sünkroonimise probleemid, kuna virtuaalserverid nõuavad rohkem aega konsensuse saavutamiseks ning andmete sünkroonimiseks, et tagada ühtne seis. Samuti ei pruugi näiteks geograafiliselt hajutatud serverid vastata teenindatavate kasutajate töötlusjõudluse vajadustele ,vaatamata sellele, et serveril võib olla hea töökindlus. Jõudluse probleemid väljenduvad kasutajate ooteaja pikkuse latentsuses ning teenindavate protseduuride täitmise ajas ehk kasutajad peavad ootama vastuseid oma toimingutele liiga kaua ja teenindavad protseduurid võivad jääda toppama [35].

Hinnang:

- Consul: 1
- Patroni: 0
- Etcd: 1

4.3.4 Turvalisus

Käesolevas peatükis vaadeldakse, mis moel valitud andmehoidlad tagavad konfidentsiaalsuse, terviklikkuse ning osapoolte autentsuse. RIA ootus on, et komponentide vaheline *end-to-end* liiklus oleks krüpteeritud ning kahepoolselt autenditud, et saavutada nullusaldus (*zero trust*) mudel.

Consuli peamiseks liideseks on RESTful HTTP API, mis vastab CRUD-toimingutele, näiteks GET, PUT ja DELETE. Kasutamaks API, CLI või kasutajaliidest, tuleks tagada juurdepääs ressurssidele juurdepääsuloendi (*access-list control*) põhise poliitikaga [36]. Näiteks defineerida halduri poolt reegel või roll ning siduda see identsustõendiga (*identity token*), mida teenus või masin saab hiljem ligipääsu taotlemiseks kasutada. Consuli klastri liikmetevaheline kommunikatsioon teabeedastamiseks toimib vaikimisi üle HTTP protokolli. Nii klient-server teenuste, kui ka serveritevaheliste arhitektuuride puhul on võimalik osapoolte autentsuse kontrolliks tagada TLS-ühendus, mille eelduseks on ühe

ja sama sertifitseerimisasutuse (CA) poolt väljastatud ja allkirjastatud sertifikaat, mis paigaldatakse erisõlmedele [37]. Teenustevahelise (*service-to-service*) identiteedipõhist pääsu reguleerimist on võimalik tagada OSI mudeli kihtide 4 ja 7 tasemel. Transpordikihis toimub näiteks teenustevaheline TLS-ühendus üle TCP protokolli ning rakenduskihis, kes ning mis CRUD-päringut konkreetse otspunkti pihta teha saab [38]. Teenustevahelise autoriseerimiseks saab korraga kasutada ühte eelnimetatud kihi autoriseerimisvalikut korraga, kus päringu õnnestumise eelduseks on vahemälus talletatud pääsupoliitika konkreetse otspunkti suhtes [38], [39]. Iga ühenduse puhul viiakse läbi päringu kontroll vastuvõtmiseks või tagasilükkamiseks [38]. Sissetulevate päringute puhul ei tohiks vaikeseaded olla vaikimisi defineeritud, kui parameeter (*allow*), mis lubab läbi kõik sissetulevad päringud [38]. Saladushoidla kontekstis on oluline, et Vault kommunikatsioon Consuliga toimuks üle krüpteeritud sidekanali ning pääsuõigus oleks tagatud identsustõendi (*identity token*) alusel [40].

Nii klient-server ühendamisel kui ka Etcd klastri moodustamisel autenditakse ning krüpteeritakse suhtlus kliendi sertifikaatide puhul tavapärase TLS-ühendusena [18]. Kasutajate pääsu nende tööalastele funktsioonidele reguleeritakse läbi rollipõhise pääsupoliitika [41].

Patronil puudub rollipõhine pääsupoliitika, kuid on võimalik seadistada autentimine CRUD-päringute tasemel API otspunkti pihta. Vaikimisi saavad kõik teha kõiki päringuid, kuid *Basic-auth*'i sisselülitamisel saavad ainult kasutaja ja parooliga autenditud kasutajad teha kirjutamisõigusega päringuid ning autentimata kasutajad ainult lugeda. [42]

Kokkuvõttes tagavad kõik nimetatud andmehoidlad komponentide vahelise kommunikatsiooni usaldusväärse ning kaitstud kanali kaudu, mis teeb ründajale kanali sisu loetamatuks ning ligipääsmatuks. Tähelepanu tuleks pöörata API otspunktide turbele, mis võivad olla siseohuks (*insider threat*), mille kaudu on ründajal võimalik pääseda ligi rakendusele või klastrile. [43].
Mitmekasutajasüsteemide ja rollipõhiste pääsude puhul tuleks lähtuda *least privilege* printsiibist, kus kasutajale, virtuaalserverile, protsessile või teenusele antakse minimaalsed õigused, mida ta vajab oma toiminguteks. Näiteks Etcd puhul juurkasutaja rolli väljastamine ainult kasutajale, mis tegeleb klastri haldamisega. [43].

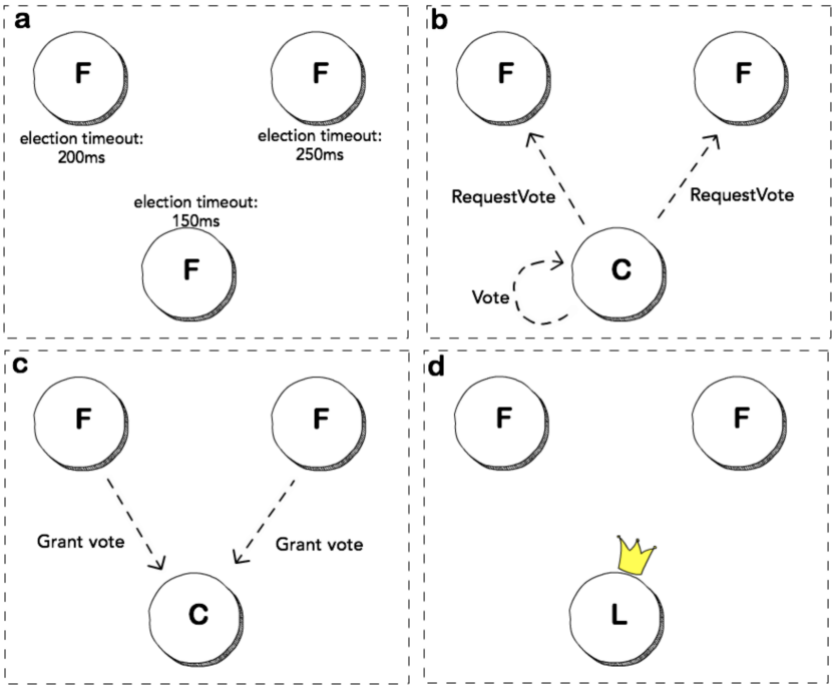
Hinnang:

- Consul: 1
- Patroni: 1
- Etcd: 1

4.3.5 Käideldavus

Kuna käideldavuse tagamiseks tuleb lisaks andmehoidlale klasterdada ka süsteemi teised komponendid, mis on omavahel seoses ja tagavad rakenduse töö, [1] vaadeldakse käesolevas peatükis andmehoidlate päringute edastamise ja koormusjaotamise võimalusi ning arhitektuuri disaini. Lähtepunktiks on see, et mida rohkem ühendusi või komponente, seda keerulisem süsteem tervikuna. Süsteemi disaini kavandamisel on lihtsuse hoidmine üks parimaid viise, kuidas parandada süsteemi turvalisust – optimaalne disain lihtsustab arusaama süsteemi üle ja hõlbustab hallatavust [43].

Kõrgkäideldavuse tagamiseks on vajalik komponentide paigaldamine $n \geq 2$ eksemplari [15]. Tõrkekindluse ja kvoorumihäälte saavutamiseks on vaja soovitatavalt vähemalt kolme-liikmelist klastrit, mis tolereerib tõrke korral ühe sõlme eemaldamist klastergrupist [34]. Klasterarhitektuuri Raft algoritmi kohaselt on üks klastri liikmetest alati juhtrollis (*leader*) ja teised liikmed järgija (*follower*) või kandidaatrollis (*candidate*) tänu millele hoitakse ära potentsiaalne *split-brain* stsenaarium [44].

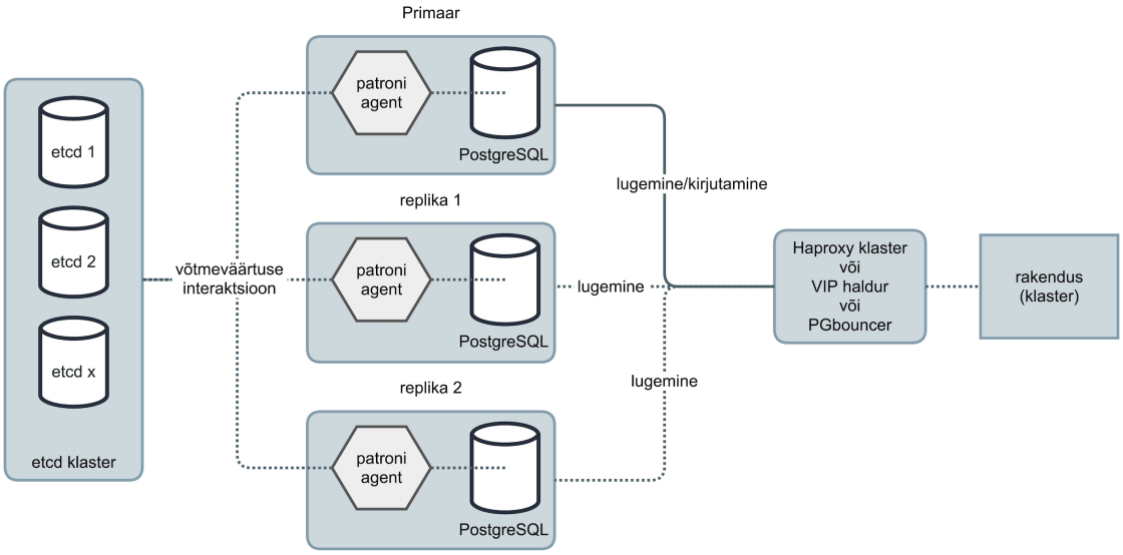


Joonis 8. Raft algoritm – Juhtrolli määramine [45].

Juhtrolli määramise protsess algab klastrisõlmede initsialiseerimisest eri aegadel, mil hetkest hakkavad sõlmed ootama juhtrollis oleva sõlme märguannet (joonis 8). Esimese sõlme märguande ajalõpp (*election timeout)* tingib kandidaadi rolli omistamise ning juhtrolli hääletuse algatamise, kus algataja sõlm hääletab esmalt enda poolt ning seejärel palub ka enda poolt hääletada teistel klastri liikmetel. Tõrkekindla klastri aluseks on pidev komponentidevaheline kommunikatsioon [45].

Päringute suunamiseks on Patroni puhul võimalik kasutada nii virtuaalset IP aadresside haldurit, PgBouncerit kui ka koormusjaoturit HaProxy [46] (joonis 9). PgBounceri kasutamise kasu tuleneb sellest, et andmebaasi kliendid teevad suurel hulgal ühendumisi PgBounceri vastu, kuid PgBouncer teeb ühendumisi andmebaasi enda külge suhteliselt vähe ning klientide jaoks kasutatakse olemasolevaid ühendusi [47]. Virtuaalne IP-aadresside haldur (VIP manager) tekitab *floating* ehk näilise IP-aadressi, mis seotakse juhtserveriga, et kliendi pöördumisel oleksid päringud suunatud alati talle [48]. HaProxy võimaldab jälgida lisaks päringute edastamisele jälgida sõlmede seisu tehes HTTP/2 päringuid teatud intervalli tagant API-i otspunktide pihta, mis väljastavad informatsiooni selle kohta, kes on juhtserver ning mis on nende olek. HaProxy ülesandeks on leida

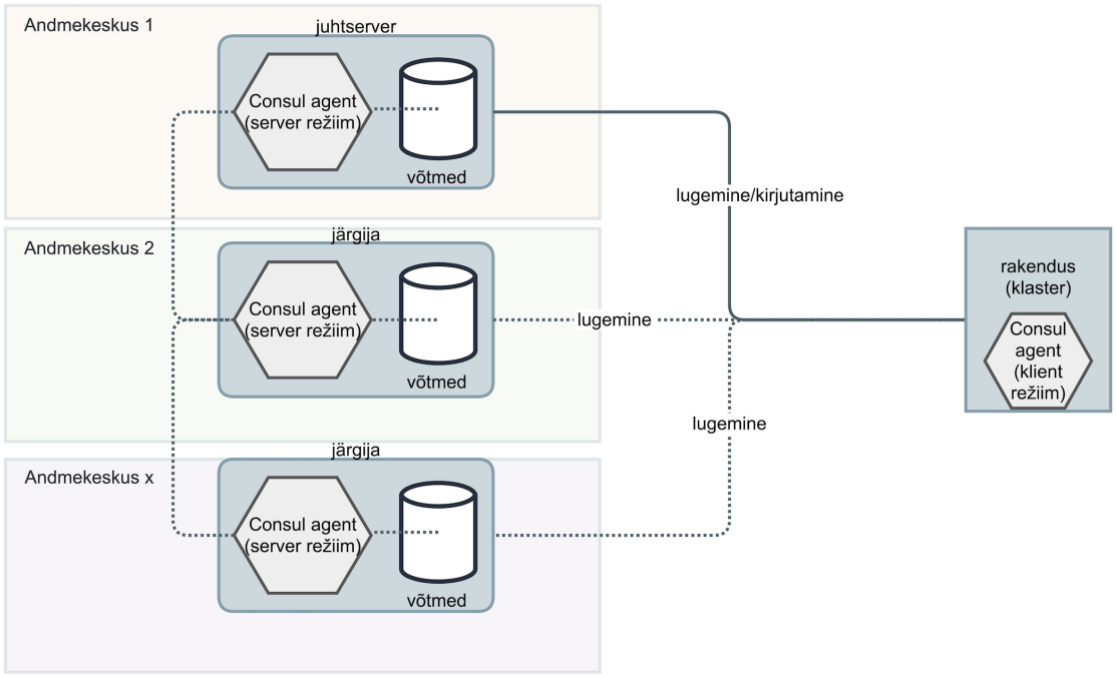
Patroni klastrist üles parasjagu juhtserver ja siis ühendada rakendus ja andmebaas TCP kanali kaudu kokku.



Joonis 9. Evitusskeem – Patroni.

Koormusjaoturist läbinud päringud võtab vastu Patroni, mis koordineerib juhtserveri initsialiseerimist ning *Patronictl* on tööriist tõrkesiirdeks (*failover*), initsialiseerimiseks, taaskäivitamiseks ja uuesti laadimiseks. Patroni iga virtuaalserver on eraldiseisev ning kasutab iseenda kõhus olevat andmehoidlat PostgreSQL. Tagasüsteemiks on võtmeväärtuste hoiustamise andmehoidla (*distributed consensus store*), näiteks Etcd, kus hoiustatakse klastri konfiguratsioone ja olekuid.

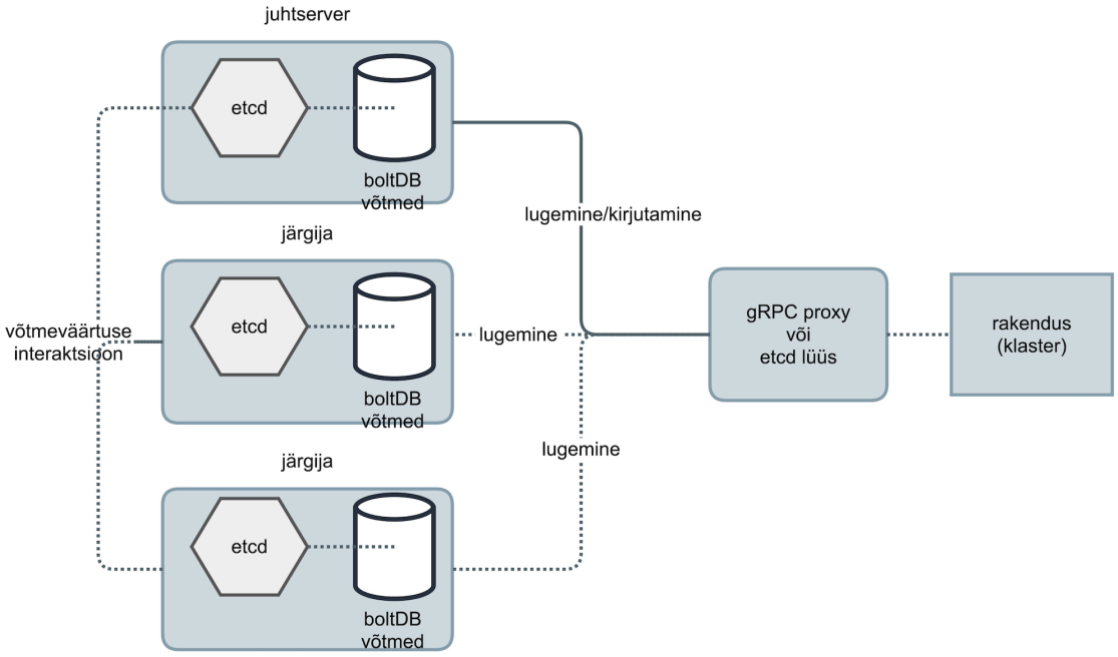
Consuli agendi võrgustiku moodustavad serveritel töötav agent peab töötama igas sõlmes, sest on peamine komponentide kommunikatsioonivahend, mis haldab liikmestaatust, registreerib teenuseid, teostab kontrolle ning vastab päringutele. Agendid töötavad kas klient- või serverrežiimis, mis on omavahel liidestuvad (joonis 10).



Joonis 10. Evitusskeem – Consul.

Agentidega varustatud serverid on üks osa Raft algoritmi komplektist, mis töötavad eesmärgil jõuda konsensusele valimaks juhtserver, mis vastutab kõikide päringute ja transaktsioonide töötlemise eest [9]. Igal sõlmel on oma andmehoidla, kus hoiustatakse klastri konfiguratsioone, olekuid ning andmeid.

Etcd puhul on võimalik kasutada päringute suunamiseks gRPC puhverserverit (*gRPC Proxy*) või lüüsi (*gateway*) (joonis 11). Lüüs on TCP puhverserver, mis ei tee muud, kui edastab päringud klastrile *round-robin* meetodil [49], [50].



Joonis 11. Evitusskeem – Etcd.

gRPC pöördpuhverserveri kasutamine aitab vähendada Etcd klastri tuumiku töötlemiskoormust. Puhverserveri käivitumisel valitakse juhuslik otspunkt, mis hakkab teenindama kõiki päringuid. gRPC puhverserver erineb lüüsist ka selle poolest, et otspunkti tõrke korral teostatakse ümberlülitamine teisele otspunktile käsitsi (*switchover*), et lõppkasutaja vaates ei esineks tõrkeid [49], [50].

Kokkuvõttes on kõige sobilikumad Etcd ning Consul, sest vajavad vähem komponentide paigaldamist, mis lisab arhitektuuri keerukust.

Kõikide nimetatud klastri juhtloogika on üles ehitatud Raft algoritmile, mis aitab tagada üksmeele klastri liikmete vahel valimaks juhtserverit. On selge, et kaheliikmelises grupis on võimatu jõuda konsensusele, millest järeldub, et kvoorumisüsteem ja hääletusprotokolli kehtestamine on üks peamiseid põhjuseid, miks klastrid on üldjoontes vähemalt kolme-sõlmelised. *Split-brain* vältimiseks hoiustavad kõik nimetatud andmehoidlad klastrikonfiguratsioone ja olekuid võtmeväärtuste hoidlas (*DCS*).

Autori hinnangul on koormusjaotamine valikuline, sest Rafti kohaselt edastatakse pärinugud klastri juhtserverile, vaatamata, mis sõlme poole pöördumine toimub. Küll annab koormusjaotur võimaluse edastada päringud otse juhtserverile, mis omakorda aitab

vähendada klastri koormust, et klastri ootel olevad sõlmed ei peaks lisa võrguliiklust tekitama päringute edastamiseks.

Hinnang:

- Consul: 1
- Patroni: 0
- Etcd: 0,5

4.4 Metoodika lõppjäreldus

Käesolevas peatükis teostatatud võrdluste alusel loodi otsustusmaatriks, kus igale andmehoidlale anti numbriline arvamus arvulisel skaalal 0, 0,5 või 1 punkt iga omaduse kohta. Prioriteetsem kriteeriumitulemus korrutatakse kordaja kolmega ning vähem prioriteetsed kahanevas järjekorras. Näiteks halduskoormus ning turvalisus on antud töö kontekstis kõige olulisemad kriteeriumid, mille suhtes ei ole võimalik organisatsioonil järelandmisi teha, sest ressurss on piiratud ning turve esmatähtis. Mastaabitavuse osas on võimalik leida kompromiss, sest antud töö kontekstis ei ole vajalik suuremahulist mastaabsust.

Näiteks HashiCorp Consul halduskeerukuse (tabel 1) hinnanguväärtus saadakse järgnevalt: 3 * 0,5 + 2 * 1 + 1 * 1 + 3 * 1 + 2 * 1 = **9,5**

Kvantitatiivsed väärtused summeeritakse ning selle põhjal valitakse välja töö teostamiseks ja probleemi lahendamiseks sobilikum andmehoidla.

Tabel 1. Otsustusmaatriks

Tagasüsteem / Kriteerium	**HashiCorp Consul**	**PostgreSQL (Patroni)**	**Etcd**
Halduskeerukus (3)	0,5	0,5	1
Süsteemsed omadused (2)	1	1	1
Mastaabitavus (1)	1	0	1
Turvalisus (3)	1	1	1
Kõrgkäideldavus (2)	1	0	0,5
Kokku	**9,5**	**6,5**	**10**

Kokkuvõtteks on kõigil kolmel tagasüsteemil on võimekus töötada võtmeväärtuste andmehoidlana. Otsustustabeli (tabel 1) tulemusena järeldame, et antud töö konteksti on Etcd võrreldes teostega kõige sobilikum.

Peamine eelis võrreldes teiste andmehoidlatega on see, et Etcd nõuab minimaalset halduskeerukust, kuna RIA-l on olemasolev kogemuspagas olemas. Halduskoormust vähendab ka sõlmedevaheline isegenereeritud sertifikaatide funktsioon, kus klient ei pea mõtlema eraldi sertifikaatide loomise peale, mis teatud juhtudel võib olla manuaalselt aeganõudev ja tülikas.

Klastri komponentidevaheline kommunikatsioon võimaldab autenditud ja turvalist ühendust ning konsistentne süsteemiomadus tagab andmete terviklikkuse, mis on saladuste puhul kriitiline olukorras, kus evitusprotsessis osalejaid on paralleelselt mitu. Kõrgkäideldava klastri omadused sõlmedevahelise konsensuse tagamiseks on sobilikud, kus ohverdatakse aeg andmete ühtsusele ning mis hoiavad ära *split-brain* stsenaariumi. Etcd-l on võimekus horisontaalselt skaleerida ning vajadusel ka tagada andmekeskusteülest suhtlust.

Autori hinnangul on Etcd, oskusteave kasulik ka seepärast, et on paljude teenuste tagasüsteemiks, nagu näiteks Patroni, Terraform ja Kubernetes, mis on ka RIA-s kasutusel.

5 Teostamine

Neljanda peatüki analüüsi tulemusel osutus valituks Etcd andmehoidla. Käesolevas peatükis antakse ülevaade kõrgkäideldava lahenduse juurutamisest, esitatakse tõenduskontspetsioon, luuakse paigaldusjuhend ning esitatakse kliendile lõplik toodanguvalmis evitusskeem.

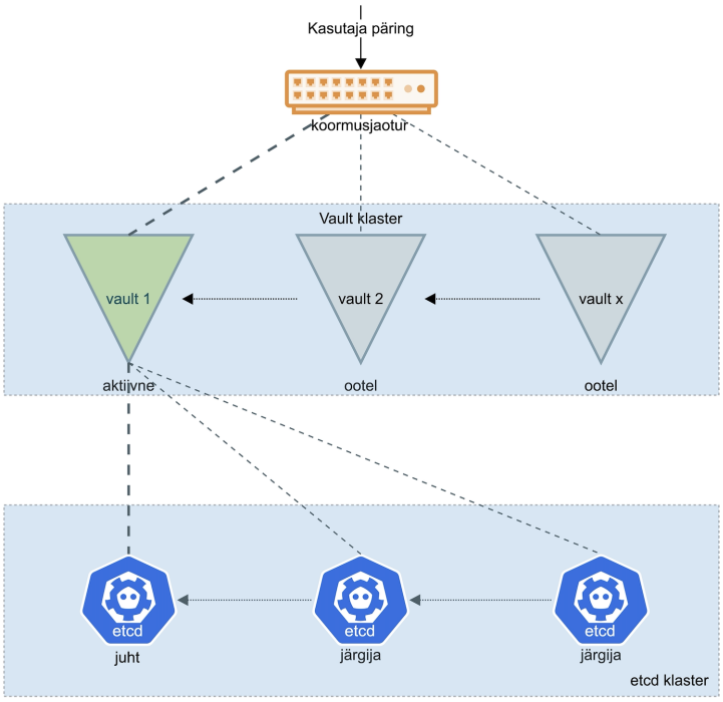
Masinate paigalduseks kasutati HashiCorp Vagrant platvormi, mis on sisuliselt tööriist virtuaalmasinate haldamiseks käsurealiidese abil. Vagrant kasutab taustal VirtualBox'i. Ruby programmeerimiskeeles autori kirjutatud Vagrantfile (lisa 3) hõlbustab tööks vajalike virtuaalserverite taasloomist ning konfigureerimist.

5.1 Komponentide kaardistamine

Top-to-bottom tehnika, aitab vaadelda kogu *end-to-end* ahelat alates tugiteenuse otspunktist kuni andmehoidlani ja kaardistada kõik vajaminevad komponendid.

1. Koormusjaotur
2. Vault Server
3. Etcd

Kliendi päring edastatakse Vault serverile, mis omakorda edastab selle tagasüsteemile (joonis 12).



Joonis 12. Evitusskeem – Lähtekontseptsioon „To be“.

Kõik tükid peaksid töötama kõrgkäideldavus režiimis, et pakkuda kas peremeesmasina peale kolimist, koormuse jaotamist või mastaabitavust, sest vastasel juhul on süsteemis nuripunkt. Käesoleva paigaldise puhul on koormusjaoture vähemalt kahes eksemplaris ning Vault ja Etcd virtuaalservereid vähemalt kolmes eksemplaris.

Esmane lähtekontseptsioon ei näe ette eraldi puhverserveri või koormusjaoturi implementeerimist Vault serveri ja andmehoidla vahele, kuna rakendus oskab pöörduda ise tagasüsteemi poole, pidades arvet nende IP-aadressite üle – see hoiab see ära lisatüki juurutamise, mis võib ka omakorda osututada nuripunktiks.

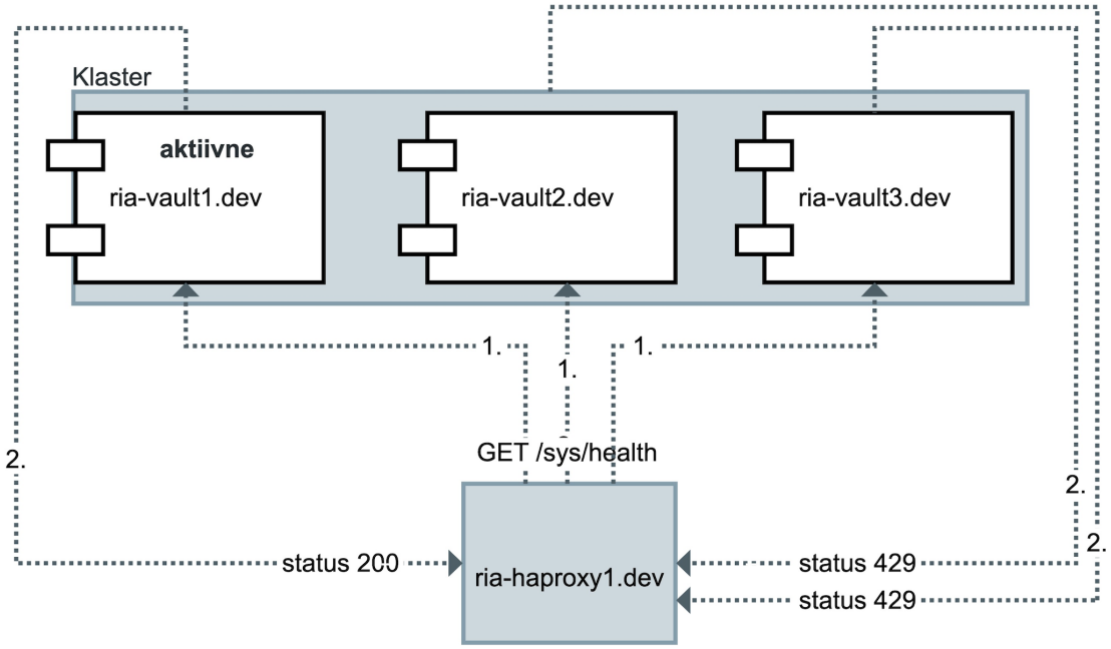
Lõppkasutaja otspunktis peaks paiknema koormusjaotur, mis vahendab rakenduskihi ja serveritesse edastatavad päringud, kuna klient ei pea teadma kõikide virtuaalserverite IP aadresse.

5.1.1 Koormusjaotur

Kliendil on küll võimalik pöörduda aktiivse Vault sõlme poole otsepöördusena, kuid eeldusel, et peetakse arvet kõikide klastrisõlmede aadressite üle [51], mis praktikas on konarlik ja ebamugav. Koormusjaoturi implementeerimine võimaldab aga lihtsustada

kliendipoolset pöördumist selliselt, et teeb selle arvepidamise ise ning tekitab kliendile ühe näilise virtuaalaadressi.

Vault API otspunkt võimaldab väljastada kõiki vajalikke olekuid, läbi mille saab koormusjaotur teha kindlaks, kes on parasjagu juhtserveri rollis ja teha vastava kliendi päringu ümbersuunamise (joonis 13), näiteks *status 200* väljendab aktiivse sõlme olekut ning *status 429* viitab ootel oleva sõlmele [52]. Sellest tulenevalt ei ole tarvilik kasutada koormusjaotamise tehnikaid *round-robin* ega *sticky-session*.



Joonis 13. Elutukse diagramm.

Infoturbe seisukohalt saab koormusjaotur tagada ka olukorra, kus rakenduse suunas pöördumised on logitud ja sinna on rakendatud ka keskse L7 taseme WAF/IDS kaitsemehhanismid.

Praktilise teostamise käigus võeti kasutusele koormusjaotur HaProxy. Terviklahendus eeldab ka selle klasterdamist, sest vastasel juhul on *end-to-end* ahelas nuripunkt, kuna RIA kasutab täna aga teist koormusjaotamise tehnoloogiat, leidis autor, et koormusjaoturi klastrilahenduse tagamine ning selle dokumenteerimine on lisakeerukus, mis ei paku kliendile antud töö kontekstis väärtust.

5.1.2 Vault

Vault klaster koosneb kolmest virtuaalserveri eksemplarist ning kõrgkäideldav lahendus töötab *active-passive* režiimis [51].

Klastri sõlmedevaheline kommunikatsioon toimub läbi tagasüsteemi, seega otsesuhtlust ei toimu, välja arvatud päringute suunamisel. Tagasüsteemiga suhtleb aktiivses klastriolekus sõlm, teised mitteaktiivset on ootel ning vajadusel suunavad päringud juhtserverile. Päringuid töötleb, käitleb ning peab arvet *Secret Engine*.

5.2 Turvalisus

Vastavalt EITS-i (Eesti Infoturbestandard) CON.1.M1 [53] järgi võeti aluseks NIST (Riiklike standardite ja tehnoloogia instituut) ning Cybernetica AS krüptograafiliste algoritmide uuringu – eelistatud TLS puhul versioon 1.3, mis pakub ulatuslikumat krüpteeringut, paremat latentsi ning kus on eemaldatud on ka ebaturvalised algoritmid [54]. Mööndustega võib kasutada versioon 1.2 olukorras, kui kasutusele võetud šifrikomplektid, mis on heaks kiidetud ja turvalised. Digitaalsignatuur algoritmiks soovitatud elliptkõver (ECDSA), mille arvutused on kiired ning mis samalajal muudab algoritmide realiseerimise lihtsaks elliptkõvera keeruturvalisuse omadus [54], [55].

Igale virtuaalserverile loodi unikaalne sertifikaat ning loomisel kasutati 384-bitist elliptkõverat ning sertifikaatide parameetrite puhul välditi koondsertifikaate (*wildcard certificate*), mis hoiab ära sertifikaadi võltsimise.

5.2.1 Minimaalõiguste printsiip

Klastri juurkasutaja loomisel lähtuti sellest, et parool oleks tugevalt räsitud, et ründajal ei oleks võimalik arvutada parooli ei jõuründega ega vikerkaare tabelite või paroolisõnastike abil. Süsteemis olevaid sertifikaate saab lugeda ainult failiõigusena omistatud teenusekasutaja ning kasutaja ligipääs on kitsendatud vaid konkreetse teenuse jaoks, st samat juurdepääsu ei ristkasutata eri teenuste juures.

5.2.2 Autentimine ja autoriseerimine

Analüüsi käigus toodi välja Etcd klastri võimekus luua ning signeerida sertifikaate klastrisiseselt, kuid see ei ole päris vastav tänase RIA sertifikaatide majandamise poliitikaga – kõikide teenuste sertifikaadid peavad olema väljastatud ühe usaldatud

sertifitseerimisasutuse poolt. Sellest tulenevalt loodi sertifikaadid klastriväliselt käsitsi autori tööarvutis, sertifikaatide allkirjastamise, verifitseerimise ja komplekteerimise tööriista CFSSL (CloudFlare's PKI/TLS) abil. Et tagada maksimaalne turvalisus, võeti jõumeetodil kasutusele edastusohje protokoll versioon 1.3, mis defineeriti ära iga teenuse konfiguratsioonifailis.

Etcd klastri töötamisega kaasneb kahte sorti liiklust ja mõlemat turvatakse TLS protokolli abil (koodinäide 1):

- ETCD_PEER_CLIENT_CERT_AUTH – klastrisse kuuluvate sõlmede omavaheline liiklus (port 2380).
- ETCD_PEER_CLIENT_CERT_AUTH – kliendi ja klastrivaheline liiklus (port 2379).

```
{
        cat << EOF > /etc/default/etcd
        ETCD_NAME="${HOSTNAME}"
        ETCD_INITIAL_CLUSTER="ria-etcd1=https://${ETCD_HOST_1}:2380,ria-
        etcd2=https://${ETCD_HOST_2}:2380,ria-
        etcd3=https://${ETCD_HOST_3}:2380"
        ETCD_INITIAL_CLUSTER_STATE="new"
        ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-1"
        ETCD_INITIAL_ADVERTISE_PEER_URLS="https://${ETCD_HOSTNAME}:2380"
        ETCD_DATA_DIR="/var/lib/etcd"
        ETCD_LISTEN_PEER_URLS="https://${ETCD_HOST_IP}:2380"
        ETCD_LISTEN_CLIENT_URLS="https://${ETCD_HOST_IP}:2379,http://127
        .0.0.1:2379"
        ETCD_ADVERTISE_CLIENT_URLS="https://${ETCD_HOSTNAME}:2379"
        ETCD_CLIENT_CERT_AUTH="false"
        ETCD_TRUSTED_CA_FILE="/etc/ssl/localcerts/ca.pem"
        ETCD_CERT_FILE="/etc/ssl/localcerts/${HOSTNAME}.pem"
        ETCD_KEY_FILE="/etc/ssl/localcerts/${HOSTNAME}-key.pem"
        ETCD_PEER_CLIENT_CERT_AUTH="true"
        ETCD_PEER_TRUSTED_CA_FILE="/etc/ssl/localcerts/ca.pem"
        ETCD_PEER_CERT_FILE="/etc/ssl/localcerts/${HOSTNAME}.pem"
        ETCD_PEER_KEY_FILE="/etc/ssl/localcerts/${HOSTNAME}-key.pem"
        ETCD_PEER_CERT_ALLOWED_CN="ria-etcd.dev"
        EOF
}
```

Koodinäide 1. Etcd klastrikonfiguratsioon.

Klient liikluse pöördumisel toimub:

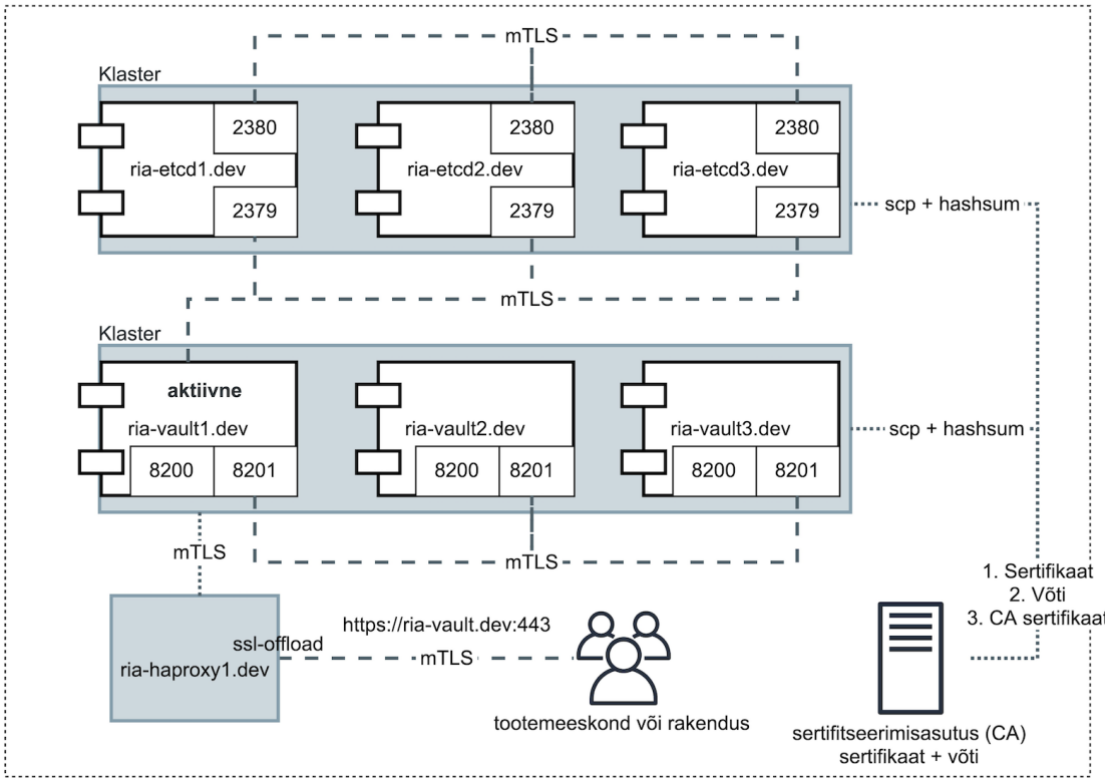
- Autentimine – Kliendi sertifikaadi esitamisel võrreldakse sertifikaadi CN väärtust Etcd sisemiselt kirjeldatud kasutajanimega.
- Autoriseerimine – Etcd sisemiselt kirjeldatud kasutajale on omistatud roll (grupp) ja omakorda on grupiga seotud õigused teha üht või teistsuguseid toiminguid andmebaasis.

Peer liikluse pöördumisel toimub:

- Autentimine – Klientsertifikaadi CN väärtuse kasutamine klienti autentsuse kontrolliks on lisandunud alatest Etcd v3.3. *Peer*-i pöördumisel kontrollitakse SAN väärtust ning ühtlasi peavad kõikide sõlmede CN väärtus ühtima. Eelnevates versioonides oli võimalik kasutada erinevaid abistavaid tehnikaid, mis põhinevad nimelahenduste pöördteisendustel, sertifikaatides IP-aadresside kasutamisel jms.
- Autoriseerimine – *Peer*'ide vaheline suhtlemise juures ei ole ette nähtud ACL kasutamist.

Koormusjaoturiga töötamisel kaasneb kahte sorti liiklust:

- *Frontend* – kehtib sissetulevatele ühendustele.
- *Backend* – kehtib rakendusele suunatud ühendustele.Ühendust kliendist koormusejaoturini tehakse krüpteeritult sertifikaatide abil.



Joonis 14. Autentimise diagramm.

Koormusjaoturist teenusemasinani teostatakse koormusevabastamine (*SSL offload*) (joonis 14), mille tulemusena vähendatakse koormust ning ühtlasi on koormusjaotur prii krüpteerimisest.

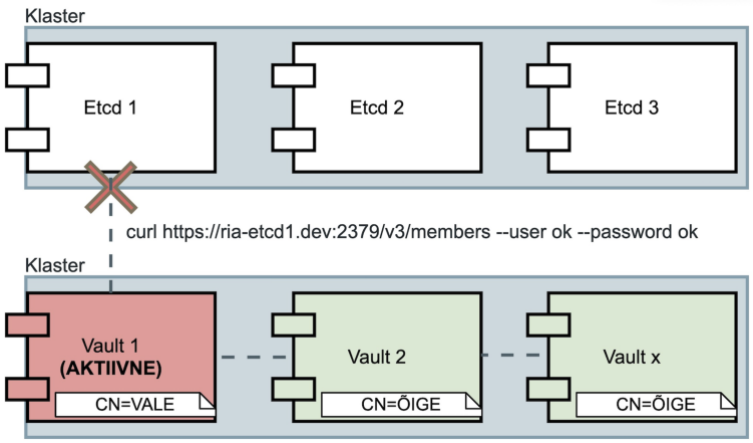
6 Testimine

Käesolevas peatükis antakse ülevaade läbiviidud kasutusloo-põhise (*use case based*) testimise tulemustest. Testimine on vajalik selleks, et saada paremat ülevaadet tarkvara kvaliteedi kohta.

Valitud testimiseliigid olid järgnevad:

- Turvatestimine
- Süsteemtestimine
- Funktsionaalne testimine

Turvatestimise tulemuse ootus oli, et väljatöötatud kontseptsioon vastab infoturbe nõuetele – komponentide vaheline ühendus on krüpteeritud, autenditud ja autoriseeritud ainult volitatud kasutajatele (joonis 15).



Joonis 15. Turvatestimine – Sertifikaadi võltsimine.

Süsteemtestimise eesmärk oli tuvastada, kas kõrgkäideldavus on rakendatud ehk rakendus- ja andmekihis olevate lubatud arvu sõlmede riknemisel jääb teenus tööle ning klastri juhtrolli vahetuse protsess on ootuspärane vastavalt Raft algoritmile. Soov oli ka, et klastrid suudavad toime tulla avarii korral iseseisvalt ehk kasutaja sekkumiseta ning ei vaja selleks eraldi käsitsi ümberlülitamist (*switchover*).

Funktsionaalse testimise osas oli ootus, et kasutaja seisukohast avarii korral teenus ega evitusprotsess ei katkeks, st kättesaadavad tegevused ning ligipääsetavad funktsioonid toimiksid ootuspäraselt nagu saladuste lisamine saladustehoidlasse.

Testitavateks objektideks olid Etcd ja Vault klaster ning kasutaja UI otspunkt.

Tulemust hinnati nii lõppkasutaja vaatest, kui ka süsteemisiseselt võrguliikluste jälgimise ning klastritarkvara kooseisu kuuluvate tööriistade abil.

6.1 Testilood

Sisendiks testide loomisele on lahendatavat probleemi ja tarkvara olemust ja käitumist kirjeldav paigaldusjuhend (lisa 2) ehk dokumentatsioon, mille põhjal koostati testiideed ja testilood (tabel 2).

Tabel 2. Vault sõlmedevahelise side katkestus – Testilugu 1.

<table>
<tr><td>Testilugu 1</td><td colspan="2">Sõlmedevahelise side katkestus (Vault)</td></tr>
<tr><td>Testiloo eesmärk</td><td colspan="2">Teeme kindlaks, kas juhtrollis sõlm annab oma juhtrolli teisele sõlmele tulenevalt sõlmedevahelisest sidekatkestusest, kus sõlmed ei näe enam üksteist võrgu tasemel.</td></tr>
<tr><td>Eeltingimused</td><td colspan="2">Lähtepunktiks paigaldusjuhise põhjal initsialiseeritud klaster</td></tr>
<tr><td>Oodatav tulemus</td><td colspan="2">Kuigi sõlm on aktiivne, ei ole tal võrgu teiste klastri liikmetele, peaks ta ennast välja lülitama / juhtrollist loobuma.</td></tr>
<tr><td>Testitsükli sammud</td><td colspan="2"></td></tr>
<tr><td colspan="3">1. Keelame ära nii sissetuleva, kui väljamineva liikluse (kõik paketid) kasutades paketifiltrit iptables (ria-vault1.dev)
2. Valideerime, et ria-vault1.dev ei näe teisi klastri liikmeid.</td></tr>
<tr><td>Tegelik tulemus (täidetakse testimisel)</td><td colspan="2">Side kaotamisel, siiski juhiroll säilib.</td></tr>
<tr><td>Järeldus</td><td colspan="2">Klastriliikmed ei vaheta omavahel teavet klastrisõlmede osas – andmed on talletatud Etcd andmebaasis. Suheldakse ainult päringute edastamiseks juhtserverile olukorras, kus klient pöördub otsepöördusena stand by sõlme poole.</td></tr>
</table>

6.2 Testitulemused ja hinnang

Turvatestimise tulemusena jõuti järeldusele, et nii klastrite- kui ka sõlmedevaheline liiklus on krüpteeritud ning erinevate sertifikaadi parameetrite nagu *Common Name* ja *Issuer* võltsimine ei taga ligipääsu klastrile. Samuti tehti kindlaks, et API otspunktid on

turvaliselt kaitstud, st et pöörduval kliendil peavad lisaks aktsepteeritavatele sertifikaatidele olema andmehoidla suhtes vastavad volitused ehk õigused.

Süsteemitestimise tulemusena järeldatakse, et autori juurutatud lahendus tagab kõrgkäideldavuse soovitud tulemusel – nuripunkte kummagis klastris ei esine. Läbitud testilood peegeldavad erinevaid klastriavarii stsenaariumeid, mis näitavad klastri iseseisvust tulla toime erinevate tõrgete korral ilma halduri sekkumiseta. Teenusetööks vajab Vault klaster ainult ühte sõlme ning Etcd vähemalt kahte. Samuti võib järeldada, et mõlema klastri puhul hoiab Raft algoritm ära *split-brain* stsenaariumi olukorras, kus sõlm lõigatakse ootamatult klastrist välja, mis omakorda tingib juhirolli oleku loobumise teisele sõlmele.

Funktsionaalse testimise tulemusena ei tingi Etcd juhirolli vahetus teenuse katkestust, andmete käideldavusekadu ega tõrkeid kliendi vaatest. Vault klastri juhirolli vahetus tingis aga teenuse katkestuse kolmeks sekundiks. Klastri käivitamine, kus lähtepunktiks on töötav operatsioonisüsteem, kuid teenus tõrke tõttu tervikuna katkenud, võtab aega Etcd puhul kolm sekundit ning Vaulti puhul kaks sekundit. Kestust mõõdeti teenuse käivitamise hetkest, kuni kliendi vaatest oli teenus kättesaadav. Klastri käivitamise aeg annab indikatsiooni nii haldurile kui ka automaatikale, millal on sõlm valmis teenindama. Näiteks regulaarsete hooldustööde korral, kus viiakse läbi muudatusi sõlme kaupa on vajalik automaatika ehitamisel teada ootuspärane käivitusaeg ehk kuna teenus muutub saadavaks.

Mõõdetud tulemusi mõjutab ka koormusjaoturi komponent, sest juhtrolli vahetus tingib uue kontrollpäringu tegemist ning sellest tulenevalt võib kliendi päringu ümbersuunamine tingida viivituse.

7 Tulemus

Töö alguses püstitati kaks eesmärki: luua NSPoF lahendus ning analüüsida rakendusega ühilduvaid tagasüsteeme.

Esmalt viidi läbi võrdlev analüüs kolme tagasüsteemi kandidaadi suhtes, mille tulemusena osutus parimaks Etcd, mis vastas kriteeriumitele ning tehnilistele nõuetele. Informatsiooni hankimisel intervjueeriti ka RIA arhitekte. Juurutamise etapile eelnes arhitektuuri kavandamine ning tõenduskontseptsiooni välja töötamine, sest tagasüsteemi implementeerimine tingis kõrgkäideldavuse tagamist rakenduskihis. Viimaks leidis autor, et oluline on ka juurutatud lahenduse testimine, mis aitas veelgi täpsemini kontrollida töö vastavust lähtetingimustele. Töö tulemusena teostati 15 erineva testi, mis katavad lähtetingimustes nõutud funktsionaalsust.

Infosüsteemi komponendid on paigaldatud kõrgkäideldavas režiimis ning järgivad tööle seatud nõudeid. Paigaldusjuhised on kirjeldatud korrektselt ning osapooltele üheselt mõistetavalt. Paigaldusjuhendi põhjal on võimalik teostada taaskorratav süsteemipaigaldus, mille toimivust on testinud nii autor kui ka klient ise.

Kastutusloo-põhistest testitulemustest võib järeldada, et kummagi klastri sõlmede juhirolli vahetus ei tingi evitusprotsessi katkemist, kui evitusprotsessis kasutataval tööriistal on seadistatud mõnesekundiline ajalõpp (*timeout*). Kokku valmis umbes 15 testitulemust, mis katavad töö lähtetingimustes nimetatud funktsionaalsust.

Lõpptulemusena loodi ka toodangukeskkonna evitusskeem ehk tõenduskontseptsioon (lisa 2).

Enne toodangukeskkonda evitamist oleks soovitatav luua taasteplaan, seadistada varundus, logimine, logiedastus, monitooring ning seire.

7.1 Kliendi tagasiside

Käesolevas peatükis kajastatakse kahe spetsialisti tagasisidet tööle. DevOps meeskonnajuhi kommentaar oli oluline, sest tegemist on DevOps platvormi tugiteenusega. Tehnilisest aspektist vaadatuna, jättis tagasiside arhitekt, kes puutub kõrgkäideldavate lahenduste kokku igapäevases töös.

„*Töös püstitatud eesmärk oli kõrgkäideldavuse tõstmine läbi nuripunktide maandamise. Oluline küsimus oli Vaulti backendi valik, mis arvestaks RIA huve ja resursse seda kasutusele võtta. Autor jõudis analüüsi käigus järelduseni, et backendiks sobib etcd. Kinnitan RIA poolt, et pakutud tehnoloogia sobib meie plaanidesse ja tehnoloogiaportfelli. Vaadates autori loodud paigaldus- ja seadistusjuhendit, olen positiivselt meelestatud selle üle, et autor on ka pööranud tähelepanu paigalduste ja seadistuste testimisele mitte ainult rakenduse, vaid ka masina ja võrgu tasemel. Minu silmis tõstab see lahenduse usaldusväärsust ning aitab edendada testiportfelli arendamist. Mina jään huviga ootama pakutud lahenduse evitamist.*“
– Siim Haas, Devops meeskonnajuht, Riigi Infosüsteemi Amet

„Paigaldusjuhend 'Kõrgkäideldav saladustehoidla Vault - backend etcd' on terviklik, muuhulgas kirjeldatakse klastri komponendid, võrk, operatsioonisüsteem, rakendustarkvara ja testitakse lahenduse töökorrasust. Esitatud andmekiht ja rakenduskiht moodustavad kumbki omaette ja samuti komplektina kõrgkäideldava tulemuse. Kuna võrguliikluse krüptimine toimub rakenduste tasemel, siis ühtlasi on lahendatud ka komponentide ja komponentide õlgade vastastikuline autentimine. Süsteemi edasine haldamine oleks lihtsam kui kõik kasutatud tarkvara tuleks Ubuntu repost, samas on täiesti mõistetav spetsiifilise etcd versiooni kasutamine lähtudes turvalisuse aspektist; vault tarkvara ei publitseerita aga üldse Ubuntu repos. Töö väga oluline osa on erinevate rikete tingimustes laheduse käitumise analüüs. Kokkuvõttes, esitatud lahendusega saab pakkuda kõrge töökindlusega saladustehoidla teenust.“
– Arhitekt, Riigi Infosüsteemi Amet

7.2 Lahenduse laiendamine ja ettepanekud

Täna luuakse, komplekteeritakse ja väljastatakse RIA-s sertifikaate käsitööna, mis tingib pika ooteaja sertifikaatide järele. Ka käesoleva töö käigus loodi mitmeid sertifikaate klastri sõlmedevaheliseks turvaliseks kommunikatsiooniks. Autori hinnangul võiks organisatsioon kasutada Vault rakendusse sisseehitatud PKI lahendust, mis pakub dünaamilist sertifikaatide väljastamist, signeerimist ning komplekteerimist, mille tulemusel väheneks meeskondade ooteaeg.

Vaulti üheks miinuseks võib välja tuua selle, et paikamise protsess näeb ette binaarfailide asendamise, mille eelduseks on seisatatud teenuseprotsess, mis tingib teenuse katkestuse aja. Käesoleva diplomitöö raames väljatöötatud klasterlahendus võimaldab aga kindla protseduuri järgi paigata, minimeerides teenuse teenusekatkestuse aega [56]. Antud protseduuri autor töös ei käsitlenud kuid valideerimine võiks tagada kõrgkäideldavuse mitte ainult klastri riknemiste, vaid ka haldustegevuste ajal.

Kõrgkäideldavuse aspekti puudutab ka Vaulti sihipärane kasutamine - rakenduste, veebiserverite jm konfiguratsioone ei ole hea talletada, sest vastasel juhul on süsteemis nuripunktiks ka evitamisel konfiguratsioonide kättesaamine. Samuti arvestades asjaolu, et saladused on kriitilise missioonitähtsusega, võiks teostada turvatestimine valideerimaks, et saladused on tõepoolest turvatud nõuetekohaselt.

Kuna Kubernetes on RIA üks arendusjärgus olevaid projekte, oleks järgmiseks sammuks ka võimalik migreerida Vault kubernetese keskkonda. Vaulti ja Kubernetese ühisosaks on tagasüsteem Etcd, sest mõlemad tehnoloogiad talletavad seal konfiguratsioone ja klastriolekuid, ja nii vähendaks konsolideerimine osaliselt arhitektuuri keerukust. Käesolevas töös valminud klaster on asjakohane, sest Kubernetese konfiguratsioonid ja olekud on kriitilise tähtsusega. Lisaks pole Kubernetesel oma saladustehoidla teenust, võimaldaks Vault ka saladusi dünaamiliselt väljastada.

Viimaks soovitab autor vahetada välja Etcd Consuli vastu, kui klient peaks tulevikus aktsepteerima lisa halduskeerukust, kui tekib vajadus suurema mastaabitavuse järele või suurema mastaabitavusega geograafiliselt hajutada. Consul pakub ka paremat ülevaadet ja visualisatsiooni klastris olevate sõlmede tervise kohta.

8 Kokkuvõte

Käesoleva bakalaureuse töö alguses anti ülevaade organisatsioonist ning autori lahendatava probleemi aktuaalsusest ning vajalikkusest.

Kasutuseloleva saladustehoidla tugiteenuse probleem seisnes kõrgkäideldavuskao riskis, mis seadis ohtu sellest sõltuvate infosüsteemide käideldavuse. Autori eesmärk oli tõsta saladustehoidla kõrgkäideldavust nuripunktide maandamise kaudu, arvestades nõudeid ning piiranguid.

Tööle seatud lähtetingimustele vastavust kontrolliti kolme erineva metoodika abil.

Kõrgkäideldavuse tagamiseks valitud tagasüsteem Etcd on osutus otstarbekaimaks valikuks halduskeerukuse minimaalsena hoidmisel, kus samal ajal on esindatud vajalikud funktsionaalsused kõrgkäideldavuse tagamiseks.

Töö täidab oma eesmärki - tulemusena valmis kõrgkäideldav saladustehoidla, milles ei esine nuripunkte. Nii rakendus- kui ka andmekihi klasterlahendused on turvalised ning süsteemsed omadused tagavad andmete terviklikkuse üheaegsel mitmekasutaja kasutamisjuhul. Meeskondade vaatest ei vähenda antud töö kasutatavust ning teenuse minimaalne katkestus aeg ei tingi pideva juurutamise ja evitamise ahela katkemist. Töö lõpuks andsid RIA esindajad ka autorile tagasisidet.

Viimaks olemasolevate võimaluste täiustamiseks on autor tulnud välja mitmete ettepanekutega, näiteks automatiseerida PKI protsess.

Lõputöö võib aidata mitmeid avaliku sektori organisatsioone ning ettevõtteid, kes soovivad parandada tugiteenuste kvaliteetikäideldavuse tõstmise kaudu.

Kasutatud kirjandus

[1] Riigi Infosüsteemi Amet, „Riigi Infosüsteemi Ameti strateegia 2021–2025,“ [Võrgumaterjal]. Available: https://www.ria.ee/sites/default/files/content-editors/RIA/ria-strateegia-2021-2025.pdf. [Kasutatud 5 Aprill 2022].

[2] HashiCorp Corporation, „Storage,“ [Võrgumaterjal]. Available: https://www.vaultproject.io/docs/configuration/storage. [Kasutatud 23 Märts 2022].

[3] HashiCorp Corporation, „HTTP API,“ [Võrgumaterjal]. Available: https://www.vaultproject.io/api-docs. [Kasutatud 21 Märts 2022].

[4] V. Praust, „Küberturbe põhimõisted ja turbemudelid, turvaülesande lahendamine,“ 1 Veebruar 2018. [Võrgumaterjal]. Available: https://enos.itcollege.ee/~valdo/kyberturve/loeng_01.pptx. [Kasutatud 24 Märts 2022].

[5] R. Peaarhitekt, Interviewee, *Teemal kõrgkäideldavus.* [Intervjuu]. 15 Veebruar 2022.

[6] The International Software Testing Qualifications Board , „International Software Testing Qualifications Board,“ 2018. [Võrgumaterjal]. Available: https://istqb-main-web-prod.s3.amazonaws.com/media/documents/ISTQB-CTFL_Syllabus_2018_v3.1.1.pdf. [Kasutatud 12 Mai 2022].

[7] K. Z. W. Gos, „The Comparison of Microservice and Monolithic Architecture,“ %1 *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH),*, 2020.

[8] HashiCorp Corporation, „What is Consul?,“ [Võrgumaterjal]. Available: https://learn.hashicorp.com/tutorials/consul/get-started#what-is-consul. [Kasutatud 08 Aprill 2022].

[9] HashiCorp Corporation, „Consul Arhitecture,“ [Võrgumaterjal]. Available: https://www.consul.io/docs/architecture. [Kasutatud 13 Aprill 2022].

[10] IBM Cloud Education, „What is etcd?,“ Detsember 2019. [Võrgumaterjal]. Available: https://www.ibm.com/cloud/learn/etcd. [Kasutatud 2022 Aprill 15].

[11] The PostgreSQL Global Development Group, „About PostgreSQL,“ [Võrgumaterjal]. Available: https://www.postgresql.org/about/. [Kasutatud 15 Aprill 2022].

[12] S. Barol, „PostgreSQL,“ TalTech IT College, 2020. [Võrgumaterjal]. Available: https://wiki.itcollege.ee/index.php/PostgreSQL. [Kasutatud 03 Aprill 2022].

[13] R. Arhitekt, Interviewee, *Intervjuu teemal Patroni.* [Intervjuu]. 23 Märts 2022.

[14] E. A. Akkoyunlu, K. Ekanadham ja R. V. Huber, „Some constraints and tradeoffs in the design of network communications,“ %1 *Proceedings of the fifth ACM symposium on Operating systems principles*, 1975.

[15] P. Somasekaram, R. Calinescu ja R. Buyya, „High-availability clusters: A taxonomy, survey, and future directions,“ *The Journal of Systems & Software,* 29 Detsember 2022.

[16] A. Bonventre, „Go Advent Day 6 - Service Discovery with etcd,“ 2013. [Võrgumaterjal]. Available: https://blog.gopheracademy.com/advent-2013/day-06-service-discovery-with-etcd/. [Kasutatud 28 Märts 2022].

[17] Bizety LLC, „Service Discovery – Consul vs ZooKeeper vs etcd,“ 17 Jaanuar 2019. [Võrgumaterjal]. Available: https://www.bizety.com/2019/01/17/service-discovery-consul-vs-zookeeper-vs-etcd/. [Kasutatud 28 Märts 2022].

[18] Etcd, „Transport security model,“ 19 August 2021. [Võrgumaterjal]. Available: https://etcd.io/docs/v3.4/op-guide/security/. [Kasutatud 21 Aprill 2022].

[19] HashiCorp Corporation, „Generate mTLS Certificates for Consul with Vault,“ [Võrgumaterjal]. Available: https://learn.hashicorp.com/tutorials/consul/vault-pki-consul-secure-tls. [Kasutatud 02 Aprill 2022].

[20] Patroni, „Security Considerations,“ [Võrgumaterjal]. Available: https://patroni.readthedocs.io/en/latest/security.html. [Kasutatud 04 Aprill 2022].

[21] J. Frederick P. Brooks, No Silver Bullet - Essence and Accident in Software Engineering, 1995.

[22] S. Gilbert ja N. A. Lynch, „Perspectives on the CAP Theorem,“ 2012. [Võrgumaterjal]. Available: https://groups.csail.mit.edu/tds/papers/Gilbert/Brewer2.pdf, https://doi.ieeecomputersociety.org/10.1109/MC.2011.389. [Kasutatud 14 Aprill 2022].

[23] Hazelcast, Inc., „What is the CAP Theorem?,“ [Võrgumaterjal]. Available: https://hazelcast.com/glossary/cap-theorem/. [Kasutatud 12 Aprill 2022].

[24] A. Ahmad, „System Design Interview Basics: CAP vs. PACELC,“ 4 Oktoober 2021. [Võrgumaterjal]. Available: https://medium.com/interviewnoodle/system-design-interview-basics-cap-vs-pacelc-cf7c5eebc313. [Kasutatud 08 Aprill 2022].

[25] A. Rana, „Understanding Etcd3,“ 2020. [Võrgumaterjal]. Available: https://medium.com/@ahadrana/understanding-etcd3-8784c4f61755. [Kasutatud 29 April 2022].

[26] Etcd, „Etcd versus other key-value stores,“ 19 November 2021. [Võrgumaterjal]. Available: https://etcd.io/docs/v3.4/learning/why/. [Kasutatud 18 Aprill 2022].

[27] PostgreSQL, „Replication modes,“ PostgreSQL, [Võrgumaterjal]. Available: https://patroni.readthedocs.io/en/latest/replication_modes.html. [Kasutatud 18 Aprill 2022].

[28] HashiCorp Corporation, „Consistency Modes,“ [Võrgumaterjal]. Available: https://www.consul.io/api-docs/features/consistency. [Kasutatud 18 Aprill 2022].

[29] Etcd, „KV API guarantees,“ 2021. [Võrgumaterjal]. Available: https://etcd.io/docs/v3.4/learning/api_guarantees/. [Kasutatud 18 Aprill 2022].

[30] PostgreSQL, „Chapter 27. High Availability, Load Balancing, and Replication,“ The PostgreSQL Global Development Group, [Võrgumaterjal]. Available: https://www.postgresql.org/docs/current/high-availability.html. [Kasutatud 18 Aprill 2022].

[31] HashiCorp Corporation, „Gossip Protocol,“ [Võrgumaterjal]. Available: https://www.consul.io/docs/architecture/gossip. [Kasutatud 19 Aprill 2022].

[32] HashiCorp Corporation, „Consul Reference Architecture,“ [Võrgumaterjal]. Available: https://learn.hashicorp.com/tutorials/consul/reference-architecture. [Kasutatud 19 Aprill 2022].

[33] HashiCorp Corporation, „Consul OSS Workshop,“ [Võrgumaterjal]. Available: https://hashicorp.github.io/field-workshops-consul/slides/multi-cloud/consul-oss/#46. [Kasutatud 20 Aprill 2022].

[34] Etcd, „Frequently Asked Questions (FAQ),“ 2021. [Võrgumaterjal]. Available: https://etcd.io/docs/v3.4/faq/. [Kasutatud 20 Aprill 2022].

[35] P. Raspel, „ANDMEBAASIDE TURVAMINE,“ [Võrgumaterjal]. Available: https://enos.itcollege.ee/~priit/2.%20Arhiiv/1.%20Andmebaaside%20administreerimine%20-%20%C3%95PPEMATERJALID/4.htm. [Kasutatud 20 Aprill 2022].

[36] HashiCorp Corporation, „ACL HTTP API,“ [Võrgumaterjal]. Available: https://www.consul.io/api-docs/acl. [Kasutatud 22 Märts 2022].

[37] HashiCorp Corporation, „Secure Consul Agent Communication with Encryption and Certificates,“ [Võrgumaterjal]. Available: https://learn.hashicorp.com/tutorials/consul/tls-encryption-openssl-secure. [Kasutatud 21 Aprill 2022].

[38] HashiCorp Corporation, „Intentions,“ [Võrgumaterjal]. Available: https://www.consul.io/docs/connect/intentions. [Kasutatud 20 Aprill 2022].

[39] HashiCorp Corporation, „Understand Consul Service Mesh,“ [Võrgumaterjal]. Available: https://learn.hashicorp.com/tutorials/consul/service-mesh. [Kasutatud 24 Märts 2022].

[40] HashiCorp Corporation, „Consul Storage Backend,“ [Võrgumaterjal]. Available: https://www.vaultproject.io/docs/configuration/storage/consul. [Kasutatud 22 Aprill 2022].

[41] Etcd, „Authentication Guides,“ 19 November 2021. [Võrgumaterjal]. Available: https://etcd.io/docs/v3.4/op-guide/authentication/. [Kasutatud 22 Aprill 2022].

[42] PostgreSQL, „https://patroni.readthedocs.io/en/latest/security.html,“ [Võrgumaterjal]. Available: Security Considerations. [Kasutatud 08 April 2022].

[43] H. Adkins, B. Beyer, P. Blankinship, P. Lewandowski, A. Opera ja A. Stubblefield, „Best Practices for Designing, Implementing and Maintaining Systems,“ %1 *Building Secure & Reliable Systems*, 2020, pp. 8-29.

[44] J. O. Diego Ongaro, „Raft Github,“ Stanford University, [Võrgumaterjal]. Available: https://raft.github.io/raft.pdf. [Kasutatud 16 Märts 2022].

[45] B. Storti, „Raft: Consensus made simple(r),“ Veebruar 2022. [Võrgumaterjal]. Available: https://www.alphasights.com/raft-consensus-made-simpler/. [Kasutatud 17 Märts 2022].

[46] M. Shaun, „Learning Oreilly,“ Veebruar 2017. [Võrgumaterjal]. Available: https://learning.oreilly.com/library/view/postgresql-high-availability/9781787125537/23194c14-43c9-4eb0-9a71-578b85c0a075.xhtml. [Kasutatud 24 Aprill 2022].

[47] Pgbouncer, „Features,“ Pgbouncer, 2022. [Võrgumaterjal]. Available: https://www.pgbouncer.org/features.html. [Kasutatud 19 Märts 2022].

[48] H.-J. Schönig, „POSTGRESQL CLUSTERING: VIP-MANAGER,“ 29 Oktoober 2020. [Võrgumaterjal]. Available: https://www.cybertec-postgresql.com/en/postgresql-clustering-vip-manager/. [Kasutatud 22 Aprill 2022].

[49] Etcd, „gRPC proxy,“ 19 August 2021. [Võrgumaterjal]. Available: https://etcd.io/docs/v3.4/op-guide/grpc_proxy/. [Kasutatud 17 Aprill 2022].

[50] Etcd, „Gateway,“ 14 Juuni 2021. [Võrgumaterjal]. Available: https://etcd.io/docs/v3.4/op-guide/gateway/. [Kasutatud 19 Aprill 2022].

[51] HashiCorp Corporation, „High Availability Mode (HA),“ HashiCorp Corporation, [Võrgumaterjal]. Available: https://www.vaultproject.io/docs/concepts/ha. [Kasutatud 23 Aprill 2022].

[52] HashiCorp Corporation, „Read Health Information,“ HashiCorp Corporation, [Võrgumaterjal]. Available: https://www.vaultproject.io/api-docs/system/health. [Kasutatud 25 Märts 2022].

[53] KPMG Baltics OÜ, Cybernetica AS, Riigi Infosüsteemi Amet, „CON.1: Krüptokontseptsioon,“ 14 märts 2022. [Võrgumaterjal]. Available: https://eits.ria.ee/et/versioon/2021/etalonturbe-kataloog/con-kontseptsioonid-ja-metoodikad/con1-krueptokontseptsioon/3-meetmed/#con1m1krueptovahendivalimisekordvastutavspetsialist3. [Kasutatud 25 Aprill 2022].

[54] Cybernetica AS, „Krüptoalgoritmid ning nende tugi teekides ja infosüsteemides,“ 2021. [Võrgumaterjal]. Available: https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/cryptoreport2021.pdf. [Kasutatud 28 Aprill].

[55] D. A. McKay ja K. A. Cooper, „Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations,“ August 2019. [Võrgumaterjal]. Available: https://doi.org/10.6028/NIST.SP.800-52r2. [Kasutatud 25 Aprill 2022].

[56] HashiCorp Corporation, „Upgrading Vault,“ [Võrgumaterjal]. Available: https://www.vaultproject.io/docs/upgrading. [Kasutatud 02 Mai 2022].

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

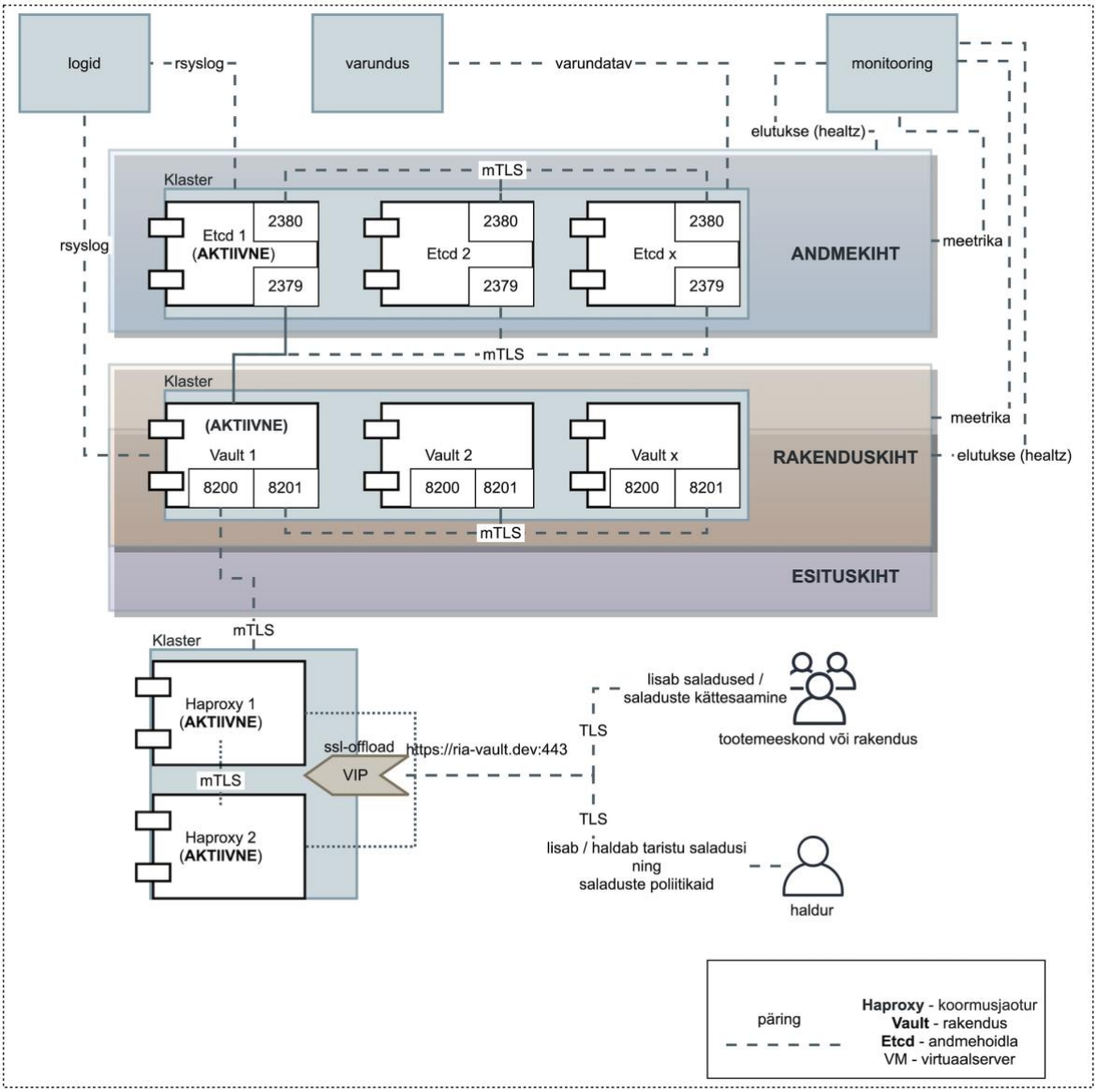
Mina, Arnold Milihhin

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose "Saladustehoidla käideldavuse tõstmine Riigi Infosüsteemi Ameti näitel", mille juhendaja on Siim Vene ning kaasjuhendaja Priit Parmakson.
   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.

17.05.2022

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.

Lisa 2 – Tõenduskontspetsioon



Lisa 3 – Vagrantfile

```
############
# -*- mode: ruby -*-
# vi: set ft=ruby :

VAGRANTFILE_API_VERSION = "2"
DOMAIN = "dev"
$disk_count = 1 # additional disks will be attached

Vagrant.configure(VAGRANTFILE_API_VERSION) do |config|
  # generate vault token - to use it you need to be logged in to vault first
from command line
  #user_vault_token = `vault token create -policy "vault" -ttl 10m -orphan -
field token`

  # default box
  box = "bento/ubuntu-20.04"

  # ansible galaxy_role_file
  #galaxy_role_file = "inventories/dev/requirements.yml"

  # Avaibale variables for servers array
  # - short_name (mandatory)
  # - hostname (mandatory)
  # - ip  (mandatory)
  # - port_host  (mandatory)
  # - group  (mandatory)
  # - box
  # - ram
  # - cpu
  # - disk (added extra disk to machine)
  # - ansible_limit
  # - ansible_playbook
  config.ssh.insert_key = false
  servers = [
    {
      :short_name => "haproxy",
      :hostname => "ria-haproxy1",
      :ip => "192.168.59.100",
      :port_host => "2200",
    },
    {
      :short_name => "etcd1",
      :hostname => "ria-etcd1",
      :ip => "192.168.59.101",
      :port_host => "2201",
      :disk => true,
    },
    {
```

```
      :short_name => "etcd2",
      :hostname => "ria-etcd2",
      :ip => "192.168.59.102",
      :port_host => "2202",
      :disk => true,
    },
    {
      :short_name => "etcd3",
      :hostname => "ria-etcd3",
      :ip => "192.168.59.103",
      :port_host => "2203",
      :disk => true,
    },
    {
      :short_name => "vault1",
      :hostname => "ria-vault1",
      :ip => "192.168.59.104",
      :port_host => "2204",
    },
    {
      :short_name => "vault2",
      :hostname => "ria-vault2",
      :ip => "192.168.59.105",
      :port_host => "2205",
    },
    {
      :short_name => "vault3",
      :hostname => "ria-vault3",
      :ip => "192.168.59.106",
      :port_host => "2206",
    },
  ]

  servers.each_with_index do |server, index|
    box_image = server[:box] ? server[:box] : box;
    config.vm.define server[:short_name] do |guest|
      guest.vm.box = box_image.to_s
      guest.vm.hostname = "#{server[:hostname]}.#{DOMAIN}"
      guest.vm.network :private_network, ip: server[:ip]
      guest.vm.network "forwarded_port", guest: 22, host: server[:port_host],
id: "ssh"
      guest.vm.provision "shell", inline: "if ! grep -q #{server[:ip]}
/etc/hosts; then echo #{server[:ip]} #{server[:hostname]}.#{DOMAIN}
#{server[:hostname]} >> /etc/hosts; fi"

      cpu = server[:cpu] ? server[:cpu] : 1;
      memory = server[:ram] ? server[:ram] : 1;

      guest.vm.provider "virtualbox" do |v|
        v.memory = 1024 * memory
        v.cpus = cpu
```

```
        v.gui = false

        if server[:disk] == true
        (1..$disk_count).each do |d|
        disk = "#{server[:hostname]}-disk#{d}.vdi"
        unless File.exist?(disk)
          v.customize ["createmedium", "--filename", disk, "--size", 2048]
          v.customize ["storageattach", :id, "--storagectl", "SATA
Controller", "--port", "#{d}", "--device", 0, "--type", "hdd", "--medium",
disk]
            end
          end
        end
      end
    end
  end
end
```

Lisa 4 – Paigaldusjuhend

Kõrgkäideldava saladustehoidla paigaldusjuhendi link.