KARELIA-AMMATTIKORKEAKOULU
Tradenomi, tietojenkäsittely

Roni Pietarinen

WWW-EDUSTAPALVELINSOVELLUS HAPROXY

Opinnäytetyö
Joulukuu 2019

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Joulukuu 2019
Tradenomi, tietojenkäsittely

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (vaihde)

Tekijä
Roni Pietarinen

Nimeke
WWW-edustapalvelinsovellus HAProxy

Toimeksiantaja
Solenovo Oy

Tiivistelmä

Opinnäytetyön tavoitteena on luoda edustapalvelin kuormantasauksella hyödyntäen avoimen lähdekoodin ohjelmistoa HAProxyä. Edustapalvelimen ensisijaisena päämääränä on toimia välityspalvelimena, eli palvella nk. asiakaspalvelimia suodattamalla sille saapuvia pyyntöjä. Edustapalvelin vastaanottaa käyttäjän selaimelta saapuvat HTTP-pyynnöt, välittää ne oikealle asiakaspalvelimelle ja palauttaa käyttäjälle asiakaspalvelimen lähettämän vastauksen.

Projektissa asennettiin erilliset Ubuntu-käyttöjärjestelmällä toimivat palvelimet toimimaan edustapalvelimena, asennettiin tarvittavat sovellukset ja konfiguroitiin HAProxy toimeksiantajan kriteerien mukaiseksi. Lisäksi ylläpito ja tietoturva voidaan hoitaa keskitetysti edustapalvelimen kautta.

Tuloksena saatiin tuotantokäyttöön viety edustapalvelin kuormantasauksella, jonka avulla kaikki asiakaspalvelimille tuleva liikenne suodattuu edustapalvelimen kautta.

Kieli	Sivuja 16
suomi	Liitteet 0

Asiasanat

edustapalvelin, haproxy, kuormantasaus, välityspalvelin

Karelia AMMATTIKORKEAKOULU

THESIS
December 2019
Business Information Technology

Tikkarinne 9
80200 JOENSUU
+358 13 260 600 (switchboard)

Author
Roni Pietarinen

Title
WWW-frontend application HAProxy

Commissioned by
Solenovo Oy

Abstract

The purpose of this thesis is to create a front-end server with load balancing by using open source application HAProxy. The primary intent of the front-end server is to act as a proxy server, meaning to filter incoming requests to client servers. The front-end server receives incoming requests from end-user's browser, passes them on to the right client server and the client server has to return a correct answer back to the client.

Ubuntu stand-alone servers were installed for the sole purpose of this project to work as front-end servers and HAProxy was configured as the commissioner requested. By centralizing the information security to few servers, maintenance is eased and the services will be available through load balancing.

As a result, front-end servers with load balancing were installed to production usage and all the incoming traffic to the client servers is filtered by the front-end servers.

Language

Finnish

Pages 16
Appendices 0

Keywords

Front-end server, haproxy, loadbalancing, proxy

Sisältö

1 Johdanto

Solenovo Oy on vuonna 1996 perustettu ohjelmistoyritys, jonka päätuotteena on tuottaa räätälöityä teknologiaa, erityisesti koulutussektorille ja julkishallinnolle. Tuotteina ovat selainpohjaiset oppilaitosten toiminnanohjauksen ja -kehittämisen ratkaisut sekä projektien johtamisen ja -hallinnan ratkaisut. Solenovon juuret ovat Joensuussa, mutta yritys palvelee asiakkaitaan valtakunnallisesti.

1.1 Projektin tavoite ja kulku

Opinnäytetyön tavoitteena on luoda edustapalvelin kuormantasauksella, hyödyntäen avoimen lähdekoodin ohjelmistoa, Haproxyä. Edustapalvelimen ensisijaisena tarkoituksena on toimia välityspalvelimena, eli palvella nk. asiakaspalvelimia suodattamalla sille saapuvia pyyntöjä. Saman prosessin aikana jopa satojen asiakaspalvelimien ylläpidosta tulee helpompaa sekä turvallisempaa. Edustapalvelimen ansiosta itse asiakaspalvelimen IP-osoite pysyy salattuna ja käyttäjä kykenee näkemään ainoastaan edustapalvelimen IP-osoitteen, joka vaikeuttaa mahdollisia hakkerointiyrityksiä. Myös ylläpito on helpompaa, sillä esimerkiksi jatkuvan tietoturvan parantaminen voidaan hoitaa pääasiassa tekemällä muutokset ainoastaan edustapalvelimelle. SSL-sertifikaatin (HTTPS) jako onnistuu kaikille asiakaspalvelimille/sovelluksille tätä reittiä pitkin sen sijaan, että se olisi jokaisella asiakaspalvelimella itsenäisesti. Vaikka edustapalvelin tarjoaa ylimääräisen kerroksen tietoturvaa, se ei kykene itsenäisesti torjumaan esimerkiksi kohdistettuja palvelunestohyökkäyksiä. (Luotonen 1994.)

Nykyään selainpohjaisilla verkkosivujen tai -sovelluksien on palveltava valtavia määriä pyyntöjä käyttäjiltä tai asiakaspalvelimilta ja sovelluksen on palautettava oikea vastaus nopeasti sekä luotettavasti. Vastauksena voi olla kuvia, videoita, tekstiä tai muuta dataa. Kuormantasaaja toimii “liikennepoliisina”, tässä tapauksessa edustapalvelimen kautta. Se on asiakaspalvelimien etupuolella ja reitittää sille saapuvat pyynnöt siten, että asiakaspalvelimet kykenevät vastaamaan kyselyihin mahdollisimman nopeasti ja tehokkaasti. Mikäli yksittäinen palvelin olisi käyttämättömänä esimerkiksi ylläpidollisista syistä tai vahingon seurauksena, osaa kuormantasaaja ohjata liikenteen jäljelle jääneille palvelimille. (Luotonen, 1994.)

Tarkoituksena on siis luoda edustapalvelin, joka vastaanottaa käyttäjän selaimelta saapuvat HTTP-pyynnöt, välittää ne oikealle asiakaspalvelimelle ja palauttaa käyttäjälle asiakaspalvelimen lähettämä vastaus. Välitysohjelman ohella tarkoituksena on myös luoda palvelin, jonka avulla kaikkien asiakaspalvelimien tietoturvaa voidaan ehostaa keskitetysti, helpottaa niiden ylläpitoa sekä mahdollistaa palvelun saatavuus kuormantasauksen kautta.

1.2 Projektin tutkimusmenetelmät

Opinnäytetyö on toteutettu pääasiassa Solenovon toimistotiloissa Joensuussa. Työn tekeminen on aloitettu tutustumalla kuormantasaukseen ja välityspalvelimiin omina käsitteinään. Projektissa oli tutustuttava, mitä välityspalvelimena sekä kuormantasaajana sovellus Haproxy pitää sisällään ja millä tavalla se kykenee palvelemaan yrityksen käyttötarpeita. Yrityksen toiminta oli jo entuudestaan tuttua viiden kuukauden harjoittelujakson sekä melkein vuoden mittaisen työjakson kautta. Erityisesti yrityksen tietohallinto- sekä teknologiajohtajan vuosien kokemuksesta oli erityistä apua projektin toteuttamisessa.

2 Yritysesittely

Yritys on perustettu vuonna 1996 Joensuussa ja toiminta kattaa nykypäivänä kaksi toimipaikkaa Suomessa, Joensuussa ja Helsingissä. Yritys palvelee asiakkaitaan valtakunnallisesti sekä maailmanlaajuisesti, osa Solenovon tämänhetkisestä kasvusta tapahtuu kansainvälisillä markkinoilla. Yritys työllistää nelisenkymmentä asiantuntijaa. Intohimo yrittämistä kohtaan on laittanut yrityksen alunperin liikkeelle. (Tanskanen 2019.)

Oppilaitoksiin ja opiskeluun liittyvä toiminta on ollut yrityksen perustajalle ja entiselle toimitusjohtajalle Kimmo Tanskaselle elämässä aina tärkeää, joten siirtymä yrittäjyyteen ja yrityksen oppilaitossektoria palvelevat tuotteet tuntuivat selkeältä etenemissuunnalta. Yrityksen historia alkaa yliopistojen kanssa tehdystä yhteistyöstä toiminnanohjausjärjestelmillä. Joensuun yliopisto oli yrityksen historian alussa hyvin edistyksellinen oppilaitos työaikojen kohdentamisen SoleTM-järjestelmän kehittämisessä. Lisäksi tutkimustietojärjestelmä SoleCRIS sekä projektinhallintajärjestelmä SolePRO olivat tuotteet, jotka toimivat keihäänkärkinä

yrityksen ensimmäisinä askeleina menestystä kohti. Samaiset tuotteet ovat olleet jälkeenpäin myös osana julkishallinnon tarjoamaa. (Tanskanen 2019.)

Yliopistokenttä on ollut aina selkeästi esillä yrityksen asiakkaina. Kun korkeakoulujen toiminta alkoi siirtymään ammattikorkeakouluihin kohti, tuntui siirtymä muihinkin oppilaitosten toiminnanohjausjärjestelmiin luontevalta. Karelia-ammattikorkeakoulun rooli opetuksen suunnittelujärjestelmän SoleOPSin suunnittelussa ja kehityksessä on ollut merkittävä. Ammatillisen koulutuksen toiminnanohjausjärjestelmä StudentaPlus tuli mukaan viimeisimpänä jäsenenä Solenovon tuotevalikoimaan. (Tanskanen 2019.)

Solenovon tavoitteena aina sen perustamisesta saakka on ollut olla mukana asiakkaiden liiketoiminnan ytimessä ja sitä tukemassa. Solenovolla on loppukäyttäjiä maailmanlaajuisesti erityisesti opiskelijavaihtojärjestelmän SoleMOVEn ansiosta. Yrityksen kasvun kannalta painopiste on tällä hetkellä kansainvälistymisessä ensisijaisesti EU-/ETA-alueella ja tavoitteena on laajentaa sekä työstää yhteyksiä myös Aasian suuntaan. (Tanskanen 2019.)

Seuraavia tuoteaihioita Solenovolla pohditaan mm. siltä pohjalta, että opinto-ohjauksen merkitys kasvaa jatkuvasti, mutta opinto-ohjaajia on liian vähän. Erilaisia opintopolkuja on tuhansia ja dataa valtavasti. Tekoälyllä ja koneoppimisen vaikutuksilla olisi mahdollista saada suuria, helpottavia askeleita oppijoille, mutta kuinka hyvin kyetään tuottamaan lisäarvoa sekä -hyötyä olemassa olevan datan pohjalta? Mitä datasta saadaan irti, miten laadukasta data on ja mitä liiketoimintaprosesseja voi mallintaa taustalle? Substanssiosaaminen on Solenovolle tärkeää, jotta saadaan oikeita näkökulmia sekä ymmärrystä ja näkemystä asiakkaan liiketoimintaan. (Tanskanen 2019.)

3 HAProxyn konfigurointi

Tärkein tiedosto Haproxyn konfiguroinnin kannalta on oletuksena /etc/haproxy/ -hakemistossa sijaitseva haproxy.cfg-tiedosto. Konfiguraatiotiedoston rakenne on seuraava:

```
global
# global settings here
```

defaults
defaults here
frontend
frontends that accept requests from clients
backend
servers that fulfill the requests

Kyseiseen konfiguraatiotiedostoon merkitään avainsanat “global”, “defaults”, “frontend” sekä “backend”. Nämä jäsentävät tiedoston helppolukuiseksi ja mahdollistavat tiettyjen parametrien sekä argumenttien käytön. (Lavoie 2018.)

3.1 Yleiset asetukset (global)

Ensimmäisenä HAProxyyn konfiguroidaan “global”-sektio. Sektio aloitetaan määrittelemällä avainsana “global” omana rivinään. Kaikki parametrit ja asetukset kyseisen avainsanan alapuolella määrittävät, mitä asetuksia konfiguraatiossa käytetään minimissään koko järjestelmän tasolla.

Projektissa on käytetty seuraavia asetuksia:
log /dev/log local0
chroot /var/lib/haproxy
pidfile /var/run/haproxy.pid
maxconn 55000
user haproxy
group haproxy
stats socket /var/lib/haproxy/stats

“log”-asetus varmistaa, että kaikki varoitukset sekä virheet lokitetaan erilliseen lokitiedostoon. Tällä varmistetaan, että esimerkiksi HAProxyn käynnistyessä tai liikenteen liikkuessa ylläpitäjät kykenevät etsimään mahdolliset ongelmatilanteet ja ratkomaan ne lokitiedostojen avulla. Määrittämällä parametriksi /dev/log/ HAProxy tietää luoda lokitiedostot kyseiseen hakemistoon ja local0 määrittää, että lokia pidetään yllä kernelitasolla UUCP-protokollaa hyödyntäen. (Mhehdbi 2019.)

“chroot”-asetus vaihtaa nykyisen hakemiston sille asetettuun parametriin, eli /var/lib/haproxy/ ennen käyttöoikeuksien pudottamista. Tällä nostetaan tietoturvan tasoa, mikäli jotakin tuntematonta tietoturva-aukkoa koitettaisiin hyödyntää. Vaihtamalla siis hakemistoa ja pudottamalla käyttöoikeudet tehdään hyökkääjän tavoitteesta mahdollisimman vaikeaa. (Lavoie 2018.)

“pidfile”-asetus kirjoittaa kaikkien daemonien PID-numerot parametrissa sijaitsevaan tiedostoon. Tällä helpotetaan esimerkiksi HAProxyn prosessin tappamista, sillä se kyetään tappamaan ainoastaan lukemalla kyseisessä tiedostossa sijaitseva PID. (Tarreau 2019.)

“maxconn”-asetus määrittää, kuinka monta yhteyttä HAProxy hyväksyy. Sen ainoa käyttötarkoitus on estää muistin loppuminen kuormantasaajalta. Lukuun 55000 on päädytty kertomalla RAM-muistin gigabittien lukumäärä luvulla 5000, jota suositellaan HAProxyn omassa dokumentaatiossa: (https://www.haproxy.com/documentation/hapee/1-8r1/onepage/intro/#3.5, Sizing)

“user” ja “group”-rivit kertovat HAProxylle, kuinka käyttöoikeudet pudotetaan HAProxyn ollessa käynnissä tietoturvan kannalta (chroot). Yksityisten TLS-avainten on oltava luettavissa ainoastaan roottina, joten käyttöoikeuksien pudottamisesta ei ole haittaa HAProxyn toimivuuden tai luotettavuuden kannalta. Mikäli “user” tai “group” ei määritetä, jatkaa HAProxy toimintaansa roottina ja tuntemattomien uhkien saapuessa heillä olisi valmiina täydet oikeudet vahingontekoa varten. (Lavoie 2018.)

“stats socket” -rivi mahdollistaa Runtime API:n käyttämisen, eli ylläpitäjät voivat dynaamisesti ottaa pois käytöstä esimerkiksi palvelimia HAProxystä, muokata kuormantasauksen priorisointia tai ylläpitäjät voivat ohjata kaiken liikenteen yksittäisen edustapalvelimen kautta. Tämä siis helpottaa ylläpitoa sekä nostaa tietoturvatasoa, sillä yksittäisen palvelimen saastuessa se voidaan ottaa pois kokonaan edustapalvelimelta, muiden palvelimien tietämättä. (Mhedhbi 2017.)

Yläpuolella olevista asetuksista on jätetty pois cipher suitet ja niitä käsitellään luvussa 3.2 (Cipher suites).

3.2 Cipher suites

Cipher suitet kuuluvat yleisiin asetuksiin (global), eli ne määritellään ko. avainsanan jälkeen. “Cipher suite” tarkoittaa protokollaa, jota käytetään avainten vaihtamiseen, datan eheyteen sekä salaamiseen kahden eri laitteen tai verkon välillä. HAProxyn konfiguroinnissa voidaan hyödyntää erinäisiä cipher suiteja, joilla määritetään, mitä protokollia halutaan käyttää tiedon välittämisessä. Tällä tavalla varmistutaan siitä, että data kulkee ylipäätään eri verkkoprotokollien välillä tietoturvallisesti. (Lavoie 2018.)

Edustapalvelimen HAProxyn konfigurointitiedostoon on oletusasetuksista poiketen lisätty tune.ssl.default-dh-param 2048. Kyseisellä parametrilla määritetään Diffie-Hellmanin avaimen koko, kun käytetään DHE-protokollaa vaihtaessa avaimia. Oletuskoko avaimille on 1024 bittiä. Kyseiselle konfiguraatiolle määritettiin avaimen kooksi 2048 bittiä, jotta sillä kyetään vaihtamaan esimerkiksi väliaikaiset 2048 RSA-avaimet ja tätä reittiä pitkin saadaan joustavuutta tinkimättä tietoturvasta. Bittikoon kaksinkertaistamisesta on kuitenkin haittapuolena se, että se aiheuttaa ylimääräistä CPU-kuormaa. Tämän kuorma ei kuitenkaan ole ongelma, sillä yksittäisellä Solenovon tuotteella on maksimissaan muutamia tuhansia käyttäjiä. (Haproxy. https://www.haproxy.com/documentation/aloha/9-0/traffic-management/lb-layer7/tls/)

HAProxyn konfiguraatiotiedostossa mukana tulevat cipher suitet ovat riittävät muutoin. Solenovon tuotteet ovat räätälöityjä eri asiakkaille, joten osa protokollista on sallittava esimerkiksi rajapintojen toimivuuden varmistamiseksi. Lisäksi oletusasetukset ovat tietoturvan kannalta todella laajat, joten haluttu tietoturvan taso saavutetaan niillä. Projektiin asetetut parametrit ja asetukset ovat kappaleen alapuolella. Oletusasetuksen parametrien oikeellisuus on varmistettu OpenSSL:n dokumentaatiosta. (OpenSSL 2018. https://www.openssl.org/docs/man1.0.2/man1/ciphers.html)

ssl-default-bind-ciphers ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-

SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-SHA:AES128-GCM-SHA256:AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:AES:CAMELLIA:DES-CBC3-SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:!aECDH:!EDH-DSS-DES-CBC3-SHA:!EDH-RSA-DES-CBC3-SHA:!KRB5-DES-CBC3-SHA

3.3 Frontend-määritykset

Kun HAProxy asetetaan reverse proxyn -rooliin backend-palvelinten edustalle, “frontend”-osio määrittää IP-osoitteet ja portit, joihin käyttäjät yhdistyvät. Konfiguraatioon voidaan lisätä frontendejä täysin tarpeen mukaan paljastamaan useat eri verkkosivut Internetiin. Jokaista “frontend” avainsanaa seuraa nimike, kuten www.mysite.com erottaakseen sen muista.

Punnitaan seuraavaa esimerkkiä ja katsotaan, mitä rivit tarkoittavat:

```
frontend www.mysite.com
    bind 10.0.0.3:80
    bind 10.0.0.3:443 ssl crt /etc/ssl/certs/mysite.pem
    http-request redirect scheme https unless { ssl_fc }
    use_backend api_servers if { path_beg /api/ }
    default_backend web_servers
```

“Bind” asettaa kuuntelijan (listener) asetettuun IP-osoitteeseen ja porttiin. “SSL” ja “CRT” argumentit opastavat HAProxyä SSL/TLS päätteiden hallintaa varten, sen sijaan että verkkopalvelimet hoitaisivat sen.

“Http-request redirect” -asetus vastaa loppukäyttäjälle, että käyttäjän on koetettava eri URL-osoitetta. Esimerkissä käyttäjät jotka koettavat päästä verkkosivuille käyttäen suojaamatonta HTTP:tä uudelleenohjataan verkkosivujen HTTPS-versioon.

“Use_backend” valitsee backendien altaasta yksittäisen palvelimen, joka vastaa saapuviin pyyntöihin, mikäli asetettu ehto on tosi. Sitä seuraa ACL-lausunto, kuten “if

path_beg /api/”. Tämä mahdollistaa HAProxyn valitsemaan jonkin tietyn backendin perustuen esimerkiksi tiettyihin kriteereihin, kuten sisältääkö polku “/api/”.

“Default_backend” antaa nimen backendille, mikäli “use_backend” -sääntö ei lähetä sitä minnekään muualle ensin. Mikäli pyyntöä ei reititetä “use_backend” tai “default_backend” kautta, HAProxy palauttaa “503 Service Unavailable” -virheen. (Lavoie 2018.)

3.4 Backend-määritykset

Backendien sektio määrittää joukon palvelimia, jotka ovat kuormantasauksen piirissä ja vastaavat saapuviin kutsuihin. Jokaiseen backendiin on määriteltävä erillinen nimike, kuten vaikkapa “web_servers”. Yleisesti ottaen sektio on suoraviivainen ja peruskäytössä asetuksia ei tarvita paljoa

Puretaan allaoleva esimerkki käsitteiksi:

```
backend web_servers
    balance roundrobin
    cookie SERVERUSED insert indirect nocache
    option httpchk HEAD /
    default-server check maxconn 20
    server server1 10.0.1.3:80 cookie server1
    server server2 10.0.1.4:80 cookie server2
```

“Balance”-asetus kontrolloi, kuinka HAProxy valitsee palvelimen vastaamaan saapuvaan kutsuun, mikäli mikään muu metodi ei ohita tätä valintaa. Pysyvä metodi voi lähettää saman kutsun samalle palvelimelle perustuen evästeisiin. Argumentti “roundrobin” aloittaa listan läpikäynnin ylhäältä alaspäin ja valitsee seuraavan mahdollisen palvelimen vastaamaan kutsuun. Vaihtoehtoisesti “roundrobinin” tilalla voisi olla “leastconn”, jossa HAProxy valitsee palvelimen, jolla on vähiten aktiivisia sessioita. “Roundrobiniin” päädyttiin, sillä Solenovon tarpeessa eivät ole SAAS-palvelut ja tietyn palvelimen on vastattava tiettyyn kutsuun.

“Cookie” mahdollistaa eväste-pohjaisuuden. Asetus kertoo HAProxylle, että eväste nimeltä “SERVERUSED” lähetetään käyttäjälle ja se assosioidaan palvelimen nimen kanssa, joka lähetti alkuperäisen vastauksen. Tämä aiheuttaa käyttäjän ja palvelimen välisen yhteyden koko session ajaksi.

Oletuksena HAProxy lähettää tason 4 (TCP) terveydentilaan liittyviä tarkistuksia. “Option httpchk” aiheuttaa HAProxyn lähettämään tason 7 (HTTP) terveydentilan tarkistuksia oletuksen sijasta. Palvelimille, jotka eivät vastaa, ei lähetetä uusia kutsuja. TCP-tarkistuksen onnistuessa backendin IP-osoitteeseen sekä porttiin mitään ei odoteta saapuvan takaisin. HTTP-tarkistuksen yhteydessä HAProxy jää odottamaan saapuvaa, onnistunutta HTTP-kutsua. Tällä tavalla mahdollistamme vikasietoisuutta ja yrityksen asiantuntijat kykenevät reagoimaan nopeammin vikatilanteisiin.

“Default-server”-asetus konfiguroi oletusasetukset mille tahansa tai kaikille palvelimille, kuten esimerkiksi terveydentilan tarkistukset tai maksimiyhteysmäärät. Tämä tekee konfiguraatiotiedostosta helppolukuisempaa sekä helpomman muokattavaksi. Vaihtoehtoisesti nämä asetukset voitaisiin määritellä palvelinkohtaisesti.

“Server”-asetus on backendin sydän. Sen ensimmäinen argumentti on nimi, jota seuraa IP-osoite ja portti backend-palvelimelle. Erillinen nimipalvelimen nimi voidaan spesifioida IP-osoitteen sijasta. Kyseisessä tapauksessa se ratkaistaan käynnistyksen yhteydessä tai sitä voidaan hallita erillisellä “resolvers”-argumentilla, jota päivitetään HAProxyn ollessa käynnissä. Mikäli nimipalvelintietue sisältää SRV-tietueen, portti ja painoarvo täytetään SRV-tietueen yhteydessä. Mikäli porttia ei määritellä erikseen, HAProxy hyödyntää samaa porttia kuin loppukäyttäjä.

Backendissä voi olla loputtomasti palvelimia, jotka hyödyntävät “server”-asetusta. Jokainen erillinen asiakaspalvelin lisätään omana rivinään backendiin, jolloin edustapalvelin ohjaa loppukäyttäjän oikealle verkkopalvelimelle. (Lavoie 2018.)

4 Testaus

Testauksella on monia etuja ja yksi suurimmista sekä tärkeimmistä eduista on kustannustehokkuus. Pitkällä aikavälillä testaaminen voi säästää rahaa projektin

budjetista. Mikäli mahdolliset bugit ja virhetilanteet todetaan aikaisessa vaiheessa, ne voidaan korjata pois mahdollisimman nopeasti ennen tuotantokäyttöä. Jos testaus jätetään tekemättä, menettävät molemmat sekä asiakas, että oma yritys rahaa. Asiakas joutuu käyttämään aikaa vikatikettien tekemiseen oikeiden töiden sijasta ja omassa yrityksessä aikaa kuluu selvittelytyöhön sekä ongelmanratkaisuun, jonka olisi voinut hoitaa jo alussa. Tällä samalla syyllä voidaan siis vaikuttaa myös tuotteen laatuun sekä asiakastyytyväisyyteen. (Testdevlab 2018. https://www.testdevlab.com/blog/2018/07/importance-of-software-testing/)

Testaus oli oleellinen osa projektia ennen tuotantokäyttöön siirtymistä edellämainituista syistä sekä tietoturvallisuuden kannalta. Testaukseen saadut kriteerit yritykseltä olivat kaikkien HAProxyn kautta välittyvien palveluiden toimivuus ulko- sekä sisäverkosta testattuna. Sisäverkon laitteet reitittyvät eri tavalla palveluihin johtuen sisäverkon IP-osoitteista, jotka on sallittu eri tavalla yrityksen palomuurissa. Ulkoverkosta käsin testattuna saadaan sama tulos, minkä loppukäyttäjä saa. Valtaosa palveluiden liikenteestä on loppukäyttäjien tuottamaa, eikä sisäverkon kautta reitittyvää.

HAProxyn toimivuutta testattiin avaamalla HAProxyn lokitiedosto /var/log/haproxy.log seurantaan. Sovellukset puolestaan pyörivät Apache Tomcatin kautta, joka toimii sovelluksen ns. moottorina. Samaan aikaan avattiin siis uusi SSH-yhteys asiakaspalvelimelle ja otettiin myös Apache Tomcatin Catalinan lokitiedosto seurantaan. Catalinaan tulostuvat kaikki sovelluksen virhetilanteet.

Jokaista sovellusta testattiin yleisimpien valmistajien selaimilla (Google Chrome, Mozilla Firefox, Internet Explorer sekä Microsoft Edge) ulko- sekä sisäverkosta. Kun lokitiedostot olivat seurannassa, kirjoitettiin sovellukselle määritetty URL-osoite selaimen hakukenttään ja varmistettiin, että käyttäjä saa oikean sivun näkyville ilman uudelleenohjauksia tai -reitityksiä. Ensimmäisenä oli varmistettava kirjautumisen toimivuus etukäteen saaduilla käyttäjätunnuksilla. Tämän jälkeen jokaista sovellusta eli asiakaspalvelinta testattiin avaamalla jokainen välilehti erikseen. Jokaisen välilehden omat toiminnallisuudet oli testattava, eli esimerkiksi erilliset raporttiajot tai tietokantayhteyksien toimivuus varmistumalla siitä, että dataa tulee näkyville oikein. Mikäli virhetilanteita olisi syntynyt missään vaiheessa, olisi joko Catalinan tai HAProxyn lokitiedostoon syntynyt erillistä virhelokia. Kun HAProxy esiasetukset saatiin konfiguroitua

oikein ja ensimmäiset front- sekä backendit lisättyä, ei mitään erillistä virhetilannetta projektin aikana syntynyt.

5 Pohdinta

Välityspalvelimen toteuttaminen sekä kuormantasaus terminä olivat täysin tuntemattomia minulle alussa. Käytin aikaa projektin alkaessa kyseisten asioiden tutustumiseen sekä HAProxyn dokumentaatioon. Asioiden selvitessä olin valmis poistumaan mukavuusalueeltani sekä aloittamaan projektin toteuttamisen. Projektin edetessä käsitykseni välityspalvelimista sekä kuormantasauksen tärkeydestä vahvistuivat ja nyt projektin päättyessä ymmärrän näiden kahden tärkeimmät merkitykset; tietoturva, ylläpito sekä palveluiden luotettavuus. Projekti on saatettu onnistuneesti tuotantokäyttöön ja toimeksiantaja on tyytyväinen lopputulokseen.

Projektia kyetään kehittämään myöhemmin esimerkiksi erikseen luoduilla sivuilla, jotka tulevat näkyviin HTTP-virhekoodien syntyessä. Virhesivua voidaan muokata tarpeen mukaan siten, että se ohjaa käyttäjää ottamaan yhteyttä yrityksen vikapalveluun ja kertomalla käyttäjälle selkeästi, että kyseessä on vikatilanne.

Lähteet

Tanskanen, haastattelu. 13.11.2019.

Luotonen, A. & Altis, K. 04.1994. World-Wide Web Proxies.Haettu osoitteesta http://courses.cs.vt.edu/~cs4244/spring.09/documents/Proxies.pdf 16.11.2019.

jjuglans. Install and configure HAProxy Load Balancer On Ubuntu 16.04. https://devops.ionos.com/tutorials/install-and-configure-haproxy-load-balancer-on-ubuntu-1604/ 11.11.2019.

Lavoie, C. 2018. The Four Essential Sections of an HAProxy Configuration. https://www.haproxy.com/blog/the-four-essential-sections-of-an-haproxy-configuration/ 11.11.2019.

HAProxy, Starter Guide:https://www.haproxy.com/documentation/hapee/1-8r1/onepage/intro/#3.5 11.11.2019.

Mhedhbi, M.. 2017. Dynamic Configuration with the HAProxy Runtime API. https://www.haproxy.com/blog/dynamic-configuration-haproxy-runtime-api/ 09.11.2019.

HAProxy Starter Guide: https://www.haproxy.com/documentation/hapee/1-8r1/onepage/intro/#3.5. 11.11.2019.

Tarreau, W. 2019. HAProxy Configuration Manual. https://cbonte.github.io/haproxy-dconv/1.7/configuration.html 12.11.2019.

Mhedhbi, M. 2019. Introduction to HAProxy Logging. https://www.haproxy.com/blog/introduction-to-haproxy-logging/ 13.11.2019

HAProxy, Configuring Transport Layer Security (TLS). https://www.haproxy.com/documentation/aloha/9-0/traffic-management/lb-layer7/tls/ 11.11.2019 & 12.11.2019.

OpenSSL, Ciphers. https://www.openssl.org/docs/man1.0.2/man1/ciphers.html 11.11.2019.

Testdevlab. 2018. 4 Reasons Why Software Testing Is Important https://www.testdevlab.com/blog/2018/07/importance-of-software-testing/ 29.11.2019.