OULUN AMMATTIKORKEAKOULU

Sami Mappes

PALVELIMETTOMAN ARKKITEHTUURIN HYÖDYNTÄMINEN VALVONTATYÖKALUN UUDISTAMISESSA PILVIYMPÄRISTÖÖN

PALVELIMETTOMAN ARKKITEHTUURIN HYÖDYNTÄMINEN VALVONTATYÖKAULUN UUDISTAMISESSA PILVIYMPÄRISTÖÖN

Sami Mappes
Opinnäytetyö
Kevät 2024
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijä(t): Sami Mappes
Opinnäytetyön nimi: Palvelimettoman arkkitehtuurin hyödyntäminen valvontatyökalun uudistamisessa pilviympäristöön
Työn ohjaaja(t): Raili Simanainen
Työn valmistumislukukausi ja -vuosi: Kevät 2024 Sivumäärä: 23

Opinnäytetyön tavoitteena oli tutkia palvelimetonta arkkitehtuuria käsitteenä ja Google Cloudin palvelimettomia palveluita. Opinnäytetyön toimeksiantona uudistettiin monitorointityökalu pilviympäristöön hyödyntäen palvelimettomia ratkaisuja. Ratkaisussa pyrittiin pilvinatiiviin ratkaisuun, joka oli helposti ylläpidettävä ja automatisoitu kaikilta osin.

Työn alussa käsiteltiin palvelimettoman arkkitehtuurin mallia ja esiteltiin osa-alueet, joihin malli voidaan jakaa. Teoriaosuudessa tutkittiin palvelimettoman arkkitehtuurin hyötyjä ja haittoja, sekä tutustuttiin tarkemmin Google Cloudin tarjoamiin palvelimettomiin palveluihin. Teoriaosuuden lopussa vertailtiin muiden pilvitarjoajien, Azuren ja AWS:n vastaavia palvelimettomia palveluita. Tarkemman vertailun kohteeksi valikoitiin kaikkien kolmen pilvitarjoajan palvelimettomat funktiot-palvelut.

Toteutusvaiheessa kuvattiin työkalun infrastruktuurin arkkitehtuuri ja infrastruktuurin toteutus, sekä työkalun reverse proxy -toiminnallisuuden yksityiskohdat. Toteutusvaiheesta esiteltiin automaatiot, joita hyödynnettiin infrastruktuurin ja työkalun julkaisemissa. Seuraavassa vaiheessa tutustuttiin työkalun testaukseen ja käyttöönottoon. Opinnäytetyön lopussa tarkasteltiin uudistamisen tuloksia ja pohdittiin toteutuksen eri osa-alueita ja miten niissä suoriuduttiin. Samalla etsittiin parannusehdotuksia tulevia vastaavia projekteja varten.

Asiasanat: palvelimeton arkkitehtuuri, reverse proxy, automaatio, monitorointi, Google Cloud

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author(s): Sami Mappes
Title of thesis: Leveraging serverless architecture to modernise a monitoring tool for the cloud
Supervisor(s): Raili Simanainen
Term and year when the thesis was submitted: Spring 2024
Number of pages: 23

The goal of the thesis was to explore the concept of serverless architecture and Google Cloud's serverless services. The project related to the thesis was to redesign and implement a monitoring tool for a cloud environment using serverless solutions. The solution aimed to be cloud native, easy to maintain and automated in every aspect.

At the beginning of the work, the concept of serverless architecture was explored and divided into two main sub-areas. The theoretical part explored the advantages and disadvantages of serverless architecture, and digged deeper into the serverless services offered by Google Cloud. At the end of the theory section, a comparison was made with similar serverless services from other cloud providers. For a more detailed comparison, the serverless FaaS services of all three major cloud providers were selected.

The implementation phase described the infrastructure architecture and implementation of the reverse proxy tool. The implementation phase presented the infrastructure code and reverse proxy tool release pipelines. The next step was the testing of the tool in new environment and getting ready for production with the implementation. At the end of the thesis, the results of the redesign project were discussed and it was considered what went well and what could still be improved.

Keywords: serverless architecture, reverse proxy, automation, monitoring, Google Cloud

SISÄLLYS

SANASTO

CentOS – Avoimen lähdekoodin Linux-jakelu. (Red Hat 2024.)

Cloud Armor – Google Cloudin palvelu, joka suojaa sovelluksia ja verkkosivustoja automaattisesti hyökkäyksiltä, kuten DDoS-hyökkäyksiltä (Distributed Denial of Service). Se tarjoaa myös erilaisia sääntöjä verkkoliikenteen suodatukseen. (Google 2024d.)

DNS – DNS eli Domain Name System kääntää ihmisten luettavissa olevat verkko-osoitteet (esimerkiksi www.google.fi) koneiden ymmärrettäviksi IP-osoitteiksi (esimerkiksi 216.58.210.163). (Amazon Web Services 2024a.)

GitHub – Pilvipohjainen palvelu koodin tallentamiselle, jakamiselle ja työskentelylle. (GitHub 2024b.)

GitHub Actions – GitHubin jatkuvan integroinnin ja toimituksen (CI/CD) palvelu. (GitHub 2024a.)

Load Balancer – Kuormantasain, joka jakaa saapuvan verkkoliikenteen useiden palvelinten välillä. Kuormantasaimet voivat toimia eri tasoilla, kuten sovellus- tai verkkokerrostasolla, ja ne tarjoavat ominaisuuksia, kuten terveystarkastukset palvelimille, istunnon ylläpitoa ja SSL-terminointia. (F5 2024.)

NAT – NAT eli Network Address Translation on palvelu, jolla yksityisessä verkossa olevat palvelut voivat käyttää julkista internetiä. NAT kääntää sisäverkon yksityiset IP-osoitteet julkisiksi IP-osoitteiksi ennen liikenteen viemistä ulkoverkkoon. (Cisco Systems 2024a.)

On-Premises – On-Premises tai on-prem tarkoittaa tilaa, jossa yrityksen laitteistot ja ohjelmistot ovat sijoitettuna. Tämä tila voi olla esimerkiksi vuokrattu konesalista. (Insight 2024.)

Python – Oliosuuntautunut, korkean tason ohjelmointikieli. (Python Software Foundation 2024.)

Reverse proxy – Käänteinen välityspalvelin, joka ottaa vastaan verkkopyyntöjä ja välittää ne eteenpäin yhdelle tai useammalle taustapalvelimelle. (Cloudflare 2024a.)

SaaS – Software as a Service tarkoittaa ohjelmistoa, jota tarjotaan pilvipalveluna käyttäjille. (Microsoft 2024.)

Serverless VPC Access Connector – Huolehtii palvelimettoman ympäristön ja VPC-verkon välisestä liikenteestä. (Google 2024e.)

SLA – Palvelutasosopimus (SLA) määrittelee toimittajan asiakkaalle lupaaman palvelutason, joka sisältää muun muassa vasteajat ja käytettävyystavoitteet. Palvelutasosopimus kertoo myös toimenpiteistä, jos palvelutasoa ei saavuteta, esimerkiksi alennuksista tai lisätuesta. (Amazon Web Services 2024c.)

Terraform – HashiCorpin kehittämä infrastruktuurikoodin (Infrastructure as Code, IaC) työkalu. Terraform mahdollistaa infrastruktuurin automaattisen hallinnan monissa palveluntarjoajissa, kuten AWS:ssä, Azuressa ja Google Cloudissa. (HashiCorp 2024.)

TFLint – Avoimen lähdekoodin työkalu, jolla tehdään staattista koodianalyysia Terraform-koodille. TFLintillä voidaan tunnistaa koodin mahdolliset konfigurointiongelmat ja syntaksivirheet. (Roper, Jack 2023.)

YAML – Datan sarjallistamiskieli konfiguraatiotiedostojen luomiseen. (Sharma 2022.)

WAF – WAF (web application firewall) on tietoturvatyökalu, joka suojaa verkkosovelluksia yleisiltä uhkilta. (Cisco Systems 2024b.)

1 JOHDANTO

Palvelimeton arkkitehtuuri on noussut nopeasti suosituksi ohjelmistoalalla, koska kehittäjä voi keskittyä täysin sovelluksensa logiikkaan ilman huolta alustan ylläpidosta. Tämä kehitysmalli ei ainoastaan vähennä tarvetta hallita fyysisiä palvelimia vaan myös parantaa järjestelmien kustannustehokkuutta ja joustavuutta. Kun yritykset ja kehittäjät pyrkivät vastaamaan nopeasti muuttuviin markkinatarpeisiin ja skaalaamaan palveluitaan ketterästi, palvelimeton arkkitehtuuri tarjoaa ratkaisevia etuja.

Tämän opinnäytetyön tavoitteena oli selvittää palvelimettoman arkkitehtuurin käsitteitä ja sitä, miten se eroaa perinteisestä palvelinarkkitehtuurista. Osana opinnäytetyötä uudistettiin valvontatyökalun yhteydessä käytetty niin kutsuttu reverse proxy -palvelin, jota hyödynnetään olennaisena osana saatavuuden valvontaan liittyvän verkkoliikenteen ohjaamiseen sellaisissa tilanteissa, joissa valvottava sovellus ei ole tavoitettavissa julkisesta verkosta. Työkalu toteutettiin hyödyntäen Google Cloud -pilviympäristön palvelimettomia ratkaisuja. Työssä etsittiin ratkaisuja muun muassa siihen, mitä hyötyjä palvelimettomasta arkkitehtuurista konkreettisesti voidaan saada ja miten sitä voidaan hyödyntää sellaisissa tilanteissa, joissa pohditaan siirtymistä perinteisestä konesali-infrastruktuurista pilveen.

Työn tavoitteena oli pilvinatiivitoteutus, joka oli linjassa muiden pilviympäristöjen ylläpitotyössä käytössä olevien työkalujen kanssa. Tarkoituksena oli vähentää ylläpitotyön kuormaa tehostamalla automaatiota ja hyödyntämällä muun muassa automatisoitua infrastruktuurin julkaisuputkea. Lisäksi tarkoituksena oli pienentää kustannuksia automaattisesti käyttökuorman perusteella skaalautuvien pilvipalveluiden avulla. Kehitystyössä syntynyttä sisäistä dokumentaatiota ja koodipohjaa voidaan tulevaisuudessa käyttää vastaavien kehitystarpeiden suunnitteluun ja toteuttamiseen.

Tehtävän toimeksiantajana oli IT-alan yritys Solita Oy. Opinnäytetyö kohdistui Solitan Cloud Platforms -yksikköön. Solita tukee asiakkaitaan AWS:n, Azuren ja Google Cloudin pilviteknologioiden käyttöönotossa, sekä palvelujen jatkuvassa kehityksessä ja operoinnissa. Solita on kehittänyt ratkaisuja pilvipalvelujen hallintaan ja automatisointiin, ja yhtiön kehittämää Solita CloudBlox® -palvelukokonaisuutta hyödynnetään monitoimittajaympäristöissä kapasiteetin, kustannusten, vaatimustenmukaisuuden ja tietoturvan hallintaan sekä sovellusten operointiin.

Solitan pilviasiakkaisiin kuuluvat muun muassa Säästöpankki, Terveystalo, Tamro, Duodecim ja Ponsse.

2 PALVELIMETON ARKKITEHTUURI

Palvelimeton arkkitehtuuri tarkoittaa mallia, jossa keskitytään sovelluskoodin suorittamiseen ilman, että tarvitsee hallinnoida palvelimia. Se ei kuitenkaan tarkoita sitä, että palvelimia ei itsessään käytettäisi hyödyksi, vaan sitä, että palvelun hallintaan ja ylläpitoon liittyvät toimenpiteet ovat pilvipalvelun tarjoajan vastuulla. Merkittävänä erona perinteisten palvelinten käyttöön on se, että palvelimetonta palvelua käytettäessä maksetaan vain palvelun todellisesta käytöstä. Perinteiset palvelimet tuottavat kuluja myös silloin, kun niitä ei käytetä. (Smart 2022, 1.)

Käsitteenä palvelimeton palvelu voidaan jakaa kahteen malliin: Backend as a Service (BaaS) ja Functions as a service (FaaS). Opinnäytetyöhön liittyvässä toteutuksessa hyödynnettiin kumpaakin mallia.

Backend as a Service on pilvipalvelumalli, jossa kehittäjät voivat ulkoistaa web- tai mobiilisovellusten taustajärjestelmät keskittyen vain käyttöliittymän kehittämiseen. BaaS-tarjoajat toimittavat valmiita ratkaisuja palvelinpään toimintoihin, kuten käyttäjien autentikointiin, tietokantahallintaan, etäpäivityksiin ja push-ilmoituksiin sekä pilvitallennukseen. (Cloudflare 2024b.)

Function as a Service mahdollistaa taustakoodin suorittamisen ilman palvelimia tai pitkäikäisiä palvelinsovelluksia, minkä vuoksi se eroaa muista arkkitehtuureista kuten PaaSista ja konteista. FaaSissa koodi ladataan palveluntarjoajan ympäristöön (esimerkiksi Google Cloud Functionsiin), joka hoitaa kaiken tarvittavan, kuten resurssien varauksen ja prosessien hallinnan. Palvelu skaalautuu automaattisesti käyttötarpeen mukaan eikä kehittäjien tarvitse huolehtia alustan ylläpidosta tai konfiguroinnista. Funktionaalinen arkkitehtuuri sallii tapahtumavetoisen suorituksen ja tukee monia ohjelmointikieliä. (Roberts 2018.)

Taulukkoon 1 on listattu, mitä hyötyjä ja haittoja palvelimettomista ratkaisuista voidaan saada.

TAULUKKO 1. Palvelimettoman ratkaisun hyödyt ja haitat

Hyödyt	Haitat
Skaalautuvuus	Potentiaaliset suorituskykyongelmat
Kustannustehokkuus	Viiveet kylmäkäynnistyksessä
Ylläpitovapaus	Rajoitetumpi hallinta
Kehityksen nopeuttaminen	Konfigurointimahdollisuudet
	Toimittajaloukku

2.1 Google Cloudin palvelimettomat palvelut

Google Cloud tarjoaa erilaisia palvelimettomien palvelujen vaihtoehtoja, joissa käyttäjän ei tarvitse itse hallita taustapalvelimia, kuten Eventarc, Workflows, Cloud SQL, Cloud Spanner ja Cloud Scheduler. Seuraavaksi tarkastellaan palvelut, jotka koskettavat tätä opinnäytetyötä, eli Cloud Run, Cloud Functions ja Firestore.

Cloud Run on hallinnoitu palvelu, jonka avulla voidaan käyttää kontteja suoraan Googlen skaalautuvan infrastruktuurin päällä. Cloud Run voi automaattisesti skaalautua palvelun liikenteen ja prosessorin käytön mukaan. Cloud Runissa on kaksi hinnoittelumallia, pyyntöpohjainen ja instanssipohjainen. Pyyntöpohjainen tarkoittaa sitä, että jos palveluun ei tule pyyntöjä, silloin prosessoria ei varata palvelulle eikä kuluja synny. Kuluja tulee tällöin kutsupohjaisesti. Instanssipohjaisessa mallissa prosessoria on aina varattuna palvelulle ja kuluja syntyy instanssin elinkaaren ajalta. Cloud Run tukee kaikilla ohjelmointikielillä rakennettuja sovelluksia, kunhan sovelluksesta voi rakentaa kontti-imagen. (Google 2024a.)

Cloud Functions on palvelimeton suoritusympäristö, jolla voidaan tehdä yksinkertaisia, yksitarkoituksellisia funktioita, jotka laukaistaan pilvi-infrastruktuurista tulevien tapahtumien

perusteella. Cloud Functionsin koodi suoritetaan hallinnoidussa ympäristössä ilman, että käyttäjän tarvitsee huolehtia infrastruktuurin ylläpidosta. Cloud Functions tukee useita ohjelmointikieliä ja tarjoaa koodin ajoon kaksi eri versiota: alkuperäisen version ja uudemman, Cloud Runiin ja Eventarciin perustuvan version, joka tarjoaa laajemmat konfiguraatiomahdollisuudet. (Google 2024b.)

Firestore on Firebasen ja Google Cloudin joustava ja skaalautuva tietokanta mobiili-, web- ja palvelinsovelluksille. Firestorella voidaan synkronoida tietoja reaaliajassa sovellusten välillä. Palvelu integroituu saumattomasti muihin Google Cloud -tuotteisiin. Firestore tukee useita ohjelmointikieliä ja mahdollistaa datan tallennuksen dokumenteissa, jotka on järjestetty kokoelmiin. Tämä rakenne tukee monimutkaisia datatyyppejä ja mahdollistaa tehokkaat kyselyt, mukaan lukien datan suodattamisen, lajittelun ja rajaamisen. Firestore suojaa datan pääsyn Firebase Authenticationin ja Firestore Security Rulesin avulla. (Google 2024c.)

2.2 Muiden pilvipalvelualustojen palvelimettomat palvelut

Muutkin kuin Google tarjoavat palvelimettomia palveluita. Tämä vertailu tarkastelee kolmen johtavan pilvipalvelutarjoajan, Google Cloudin, AWS ja Azuren palvelimettomia ratkaisuja. Vertailusta saa tukea päätöksentekoon, kun tiedetään mitä vaatimuksia pilvipalvelualustan tulee täyttää. Pääasiallisesti kaikkien pilvipalvelualustojen palvelut ovat samankaltaisia, mutta eroja löytyy esimerkiksi funktiot-palveluiden tuetuissa kielissä. Taulukossa 2 on kuvattu yleisimmät palvelimettomat palvelut Google Cloudista, AWS:stä ja Azuresta. Taulukossa 3 on vertailtu funktiot-palveluiden muutamia merkittäviä ominaisuuksia eri pilvipalvelualustojen välillä.

TAULUKKO 2. Yleisimmät palvelimettomat ratkaisut pilvipalvelualustoilla

Google	AWS	Azure
Cloud Run	AWS Fargate	Azure Container Apps
Cloud Functions	AWS Lambda	Azure Functions
Firestore	AWS Amplify	Azure Cosmos DB

TAULUKKO 3. Funktiot-palveluiden ominaisuuksien vertailu pilvipalvelutarjoajien välillä

Ominaisuus	**Google Cloud Functions**	**AWS Lambda**	**Azure Functions**
Kielituki	Node.js, Python, Go, Java, Ruby, PHP, .NET	Java, Go, PowerShell, Node.js, C#, Python ja Ruby	Python, .NET, JavaScript, Java, PowerShell, Azure CLI
Automaattinen skaalaus	Kyllä	Kyllä	Kyllä
Laskutus	Ajoaika, kutsujen lukumäärä ja resurssien varaus	Ajoaika ja kutsujen lukumäärä ja resurssien varaus	Ajoaika, kutsujen lukumäärä tai premium-vaihtoehto, jossa maksetaan resurssien varauksesta

AWS Lambda on tietojenkäsittelypalvelu, joka mahdollistaa koodin suorittamisen ilman palvelimien hallintaa. Lambda huolehtii kapasiteetin varauksen, automaattisen skaalauksen ja lokituksen. Palvelulle toimitetaan koodi Lambda-funktioina, jotka suoritetaan tarpeen mukaan ja skaalautuvat automaattisesti. Palvelun kustannukset syntyvät vain käytetystä ajasta. Lambda sopii erityisesti sovelluksiin, jotka vaativat nopeaa skaalautumista ylös ja alas, kuten tiedostojen ja datavirtojen käsittelyyn, verkkosovelluksiin, IoT-taustajärjestelmiin ja mobiilitaustajärjestelmiin. Lambda tarjoaa useita ominaisuuksia, kuten ympäristömuuttujat, versiohallinnan, konttikuvat ja yksityisverkon tuen. (Amazon Web Services 2024a.)

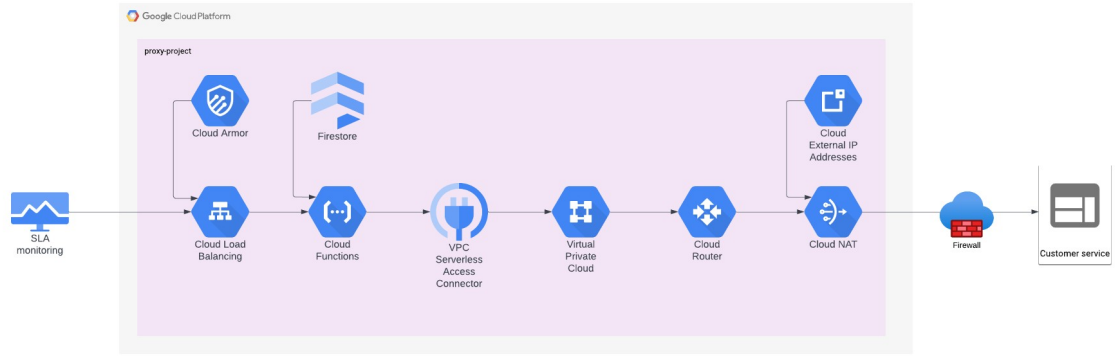
Azure Functions on Microsoft Azuren pilvipalvelu, joka mahdollistaa keskittymisen vain koodiin ja sen suorittaviin tapahtumiin. Azure Functions poistaa pitkäaikaisten laskentaresurssien varauksen ja hallinnan tarpeen, laskuttaen vain käytetyn suoritusajan perusteella. Käyttäjät voivat luoda,

konfiguroida ja ladata koodia sekä määrittää laukaisijat, jotka suorittavat koodin. Funktiot voivat käsitellä tietoja, ajastettuja tehtäviä ja muita tapahtumia eri lähteistä. Azure Functions tukee monia ohjelmointikieliä ja integraatioita eri Azuren palveluihin, kuten Azure Cosmos DB:hen ja Azure Service Busiin. (Kirvan, Paul 2023.)

3 TYÖKALUN UUDISTAMINEN PALVELIMETTOMALLA ARKKITEHTUURILLA

Opinnäytetyöhön liittyvässä toimeksiannossa oli tehtävänä toteuttaa räätälöity reverse proxy -palvelu osaksi sovellusten valvontakokonaisuutta. Tehtävän toteutukseen hyödynnettiin Google Cloudin palveluita. Muut pilvipalvelutarjoajat, kuten Azure tai AWS olisivat sopineet myös työn vaatimuksien täyttämiseen. Esimerkiksi reverse proxy -toiminnallisuuden toteutukseen käytetty Python-kieli on tuettu kaikissa mainituissa kolmen pilvipalvelutarjoajan funktiot-palveluissa. Google Cloudiin päädyttiin johtuen osittain siitä, että projektissa olleilla henkilöillä oli aiempia hyviä kokemuksia kyseisen pilvipalvelualustan käytöstä.

Kuviossa 1 on kuvattu pilvi-infrastruktuuri kokonaisuudessaan ja se, miten SLA-valvontajärjestelmistä lähtevät kutsut päätyvät pilviympäristön läpi valvottavan sovelluksen rajapintaan. Uudistetun työkalun ylläpito ja hallinta automatisoitiin kaikilta mahdollisilta osin GitHub Actionsia hyödyntäen.



KUVIO 1. Toteutuksen pilviarkkitehtuurikaavio

3.1 Uudistamisen taustaa

Uudistamisen lähtökohtana oli korvata perinteisessä konesalissa oleva työkalu pilvinatiivilla ratkaisulla, joka olisi mahdollisimman huoltovapaa ja automaattisesti skaalautuva. Työkalun toimintaperiaate on hyvin uniikki ja sen takia ei pystytty tunnistamaan valmista palvelua, jota olisi voitu hyödyntää. Tästä syystä työkalu päätettiin toteuttaa itse. Työ toimi samalla arviointina ja

suunnannäyttäjänä sille, voiko tällä tapaa ylipäätänsä tehdä ja mikä on hyvä suunnittelumalli tämän kaltaiselle projektille.

Yksi merkittävä lisäsyy uudistamiselle on vanha käytöstä poistuva reverse proxy -palvelin, joka sijaitsee on-prem-palvelinympäristössä CentOS 7 Linux-palvelimella. Koska CentOS 7 Linux-jakelu lähestyy elinkaarensa loppua, havaittiin, että tässä tilanteessa oli otollinen hetki siirtyä pois on-prem-palvelimelta.

SLA-valvontatyökalut ovat yksi osa Solitan Cloud Platforms -yksikön jatkuviin palveluihin liittyvää valvontojen palvelukokonaisuutta. Reverse proxy -palvelimen rooli on ohjata SLA-valvonnoista tulevat sivustojen saatavuuden valvontaan liittyvät pyynnöt kohdepalvelimelle, joka ei ole julkisesti saatavilla internetissä. Kohdepalvelin voi olla esimerkiksi asiakkaan sovelluksen sivusto tai rajapinta, jonka ei haluta olevan tavoitettavissa julkisesti. Reverse proxy -palvelu on olennainen osa saatavuuden valvontaa, jota Solitan pilviyksikkö tarjoaa asiakkailleen.

3.2 Uudistamisprosessi

Työkalun uudistamisprosessi alkoi infrastruktuurin rakentamisesta. Uuden palvelun pilvi-infrastruktuuri toteutettiin Terraformia hyödyntäen. Terraform-koodi jaoteltiin kahden ajoympäristön eli testi- ja tuotantoympäristön koodeihin sekä näiden ympäristöjen hyödyntämiin yhteisiin moduuleihin. Testiympäristön käyttötarkoitus oli toteutuksen koodimuutosten varmentaminen sekä monitorointien testaaminen ennen tuotantokäyttöön siirtymistä. Reverse proxy -toiminnallisuuden käyttämät asiakasympäristöjen ohjaussäännöt rakennettiin tuotantoympäristön koodin alaisuuteen.

Ajoympäristöjen infrastruktuurikoodi jaoteltiin neljään eri kokonaisuuteen: Cloud Armor WAF-säännöt, GCP-projektien bootstrap eli ympäristöjen alustaminen, työkalun infrastruktuuri (verkko, Load Balancer ja Cloud Functions) sekä valvontojen ohjaussäännöt (Firestore-dokumentit). Reverse proxy -palvelimen rajapinnalle varattiin julkinen IP-osoite ja liikenne Cloud Functionsissa olevalle reverse proxy -palvelulle ohjattiin Cloud Load Balancerilla. Load Balancerin eteen rakennettiin Cloud Armor -säännöstö, jossa sallittiin liikenneyhteydet palveluun vain luotetuista verkkolähteistä eli tässä tapauksessa SLA-valvontatyökalun julkisista IP-osoitteista ja ylläpitoon ja testaukseen käytetyistä verkkolähteistä.

Työkalun varsinainen toiminnallisuus koodattiin Pythonilla, joka otettiin käyttöön Cloud Functions -palvelussa. Työkalun hyödyntämät ohjaussäännöt tallennettiin Firestore-tietokantaan, josta Cloud Functions pystyi lukemaan oikean ohjaussäännön http-kutsussa olevan polun perusteella. Työkalun koodi toimii siten, että ensin alustetaan Firebase Adminin tunnukset Firestore-tietokantaan ja käytetään Firestorea määrittämään pyyntöjen käsittely eri poluille. Koodi tarkistaa Firestoresta kutsun polun perusteella oikean konfiguraation, kun palvelu vastaanottaa pyynnön. Jos konfiguraatio löytyy tietokannasta ja on ehtojen mukainen, pyyntö ohjataan eteenpäin määritellylle kohdepalvelimelle. Pyyntöön voidaan tarvittaessa lisätä mukautettu otsake. Tämän jälkeen saatu vastaus välitetään takaisin SLA-valvontajärjestelmään. Työkalu tukee myös mahdollista saatavuuden testauksen ajoittamista cron-ajastuksella.

Kutsut reverse proxy -palvelusta eteenpäin ohjattiin kohti valvottua sivustoa tai palvelua käyttäen Serverless VPC Access Connectoria, jonka tehtävä oli ohjata liikenne ulospäin Cloud Functionsista Google Cloudin verkosta Cloud Routeria ja Cloud NATia hyödyntäen. Valvotun palvelun kohdepäässä täytyi olla lisättynä palomuurisääntö, jossa sallittiin liikenne reverse proxy -palvelun julkiselle NAT-osoitteelle, jotta kutsu meni läpi.

Työkalun julkaisuputki toteutettiin GitHub Actions workflow'eilla. GitHub Actions koostuu workflow'eista, joita voidaan käynnistää eri tavoin, kuten automaattisesti tietyillä ehdoilla tai manuaalisesti. Yksi workflow voi sisältää yhden tai useampia tehtävän, joita voidaan suorittaa peräkkäin tai rinnakkain. Tehtävät ajetaan omassa virtuaaliympäristössä ja jokainen tehtävä voi sisältää yhden tai useamman vaiheen. Vaiheet suorittavat määritettyjä toimintoja tai skriptejä. Workflow'it määritetään YAML-muotoisina tiedostoina GitHub-repositoryyn. (GitHub, 2024.)

Työkalun infrastruktuurille tehtiin ympäristökohtaiset GitHub Actions workflow'it, joilla Terraform infrastruktuurikoodi voidaan ajaa sisään GCP-projekteihin. Kullekin ympäristölle on kolme workflow'ia: WAF-säännöt, infrastruktuuri ja ohjaussäännöt. Tämä mahdollisti sen, että toteutuksen eri osia voitiin muokata ja julkaista riippumatta toisistaan. Esimerkiksi uusia ohjaussääntöjä voitiin lisätä koskematta infrastruktuuritoteutuksen koodiin.

4 TYÖKALUN TESTAUS JA KÄYTTÖÖNOTTO

Työkalun toiminnallisuutta testattiin käyttämällä ennalta sovittua asiakastapausta tai jotakin muuta valvottavaa palvelua tai rajapintaa. Uuden toteutuksen Terraform-koodiin lisättiin valvottavan kohteen ohjaussääntö. Valvottavan kohteen palomuuriin sallittiin liikenne pilviympäristön julkisesta NAT-osoitteesta ja SLA-valvontatyökaluun määritettiin testitapauksen saatavuusvalvonnalle Load Balancerin julkinen DNS-osoite ja ohjaussäännössä käyttämä polku. Kun konfiguraatiot oli tehty, voitiin SLA-valvontajärjestelmästä tarkistaa, että valvontajärjestelmä sai yhteyden rajapintaan pilvessä ja valvonta palautti OK-vastauksen. Testauksessa varmistettiin myös, että valvonta hälytti, jos valvottava sovellus tai palvelu oli alhaalla.

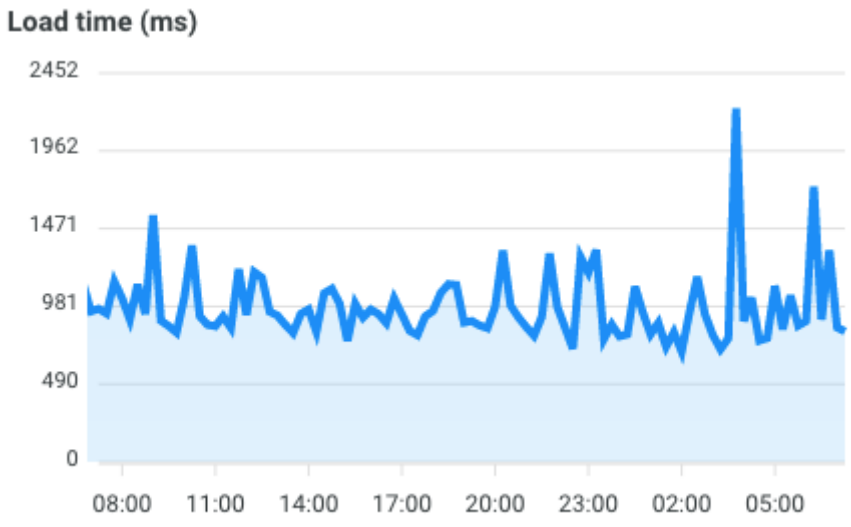
Uuden palvelun käyttöönotto oli tämän opinnäytetyön tekemisen aikaan vielä kesken, mutta käyttöönottoa varten on laadittu suunnitelma. Uudet valvontatapaukset hyödyntävät jo uutta reverse proxy -toiminnallisuutta, mutta vanhat valvonnat tullaan siirtämään vaiheittain asiakas kerrallaan käyttäen samaa prosessia kuin testausvaiheessa. Käyttöönoton lopuksi vanhat asiakaskonfiguraatiot poistetaan on-premises-palvelimelta.

Selkeä ja ennalta testattu suunnitelma vähentää riskejä käyttöönotossa. Reverse proxy -palvelu on olennainen osa valvontaputkea ja ilman sitä valvonnat eivät toimi. Pitkät käyttökatkot valvontaputkessa voivat aiheuttaa asiakkaille merkittävää haittaa, koska he eivät silloin saa palvelua, jota heille on luvattu. On elintärkeää, että saatavuudenvalvonta toimii katkotta, jotta päivystäjät ja ylläpitotiimi voivat reagoida sovellusten käyttökatkoihin ja korjata mahdolliset ongelmat.

5 TYÖN TULOKSET

Palvelun käyttöönoton jälkeen havaittiin, että pilvikapasiteetin kustannukset ovat pysyneet maltillisina – yhteensä noin 130 euroa kuukaudessa. Nämä kulut ovat noin 100 euroa suuremmat vanhaan toteutukseen verrattuna. Tämän perusteella voidaan todeta, että pelkät kapasiteettikulut ovat kalliimmat pilviympäristössä. Uuteen palveluun ei kuitenkaan ole tarvinnut tehdä huoltotöitä. Uusien ohjaussääntöjen lisääminen on ollut vaivatonta, koska sääntöjen lisääminen on automatisoitu alusta loppuun asti Terraformin ja GitHub Actionsin avulla.

Latenssiajat ovat olleet jonkin verran suurempia verrattuna vanhaan toteutukseen, mutta tällä ei ole suurta merkitystä valvonnan toiminnallisuudelle. Latenssiaikojen ero selittyy osaksi sillä, miten Cloud Functionsin kylmäkäynnistys toimii. Funktion suoritusympäristö alustetaan joskus tyhjästä, koska funktiot ovat tilattomia (Google 2024f). Kun uusi reverse proxy -palvelu vastaanottaa kutsun, sillä kestää jonkin verran aikaa käynnistyä nollatilanteesta. Kuviossa 2 näkyy uuden toteutuksen kutsujen latenssiaikoja SLA-valvontajärjestelmästä, ja kuviossa 3 on vanhan toteutuksen latenssiaikoja. Latenssiaikoja voisi pienentää sillä, että funktion asetuksiin määrittää vähimmäisinstanssien määrän.



KUVIO 2. Esimerkki latenssiajoista uudella toteutuksella

KUVIO 3. Esimerkki latenssiajoista vanhalla toteutuksella

Uutta työkalua ei opinnäytetyön tekemisen aikaan ollut vielä täysin otettu käyttöön kaikissa asiakkuuksissa, joten kaikkia mahdollisia hyötyjä ja haittoja ei ole tunnistettu. Työkalun käyttö ja kuormitus tulee tulevaisuudessa kasvamaan, kun vanhan ympäristön ohjaussäännöt siirretään uuden työkalun alaisuuteen. Voidaan myös odottaa, että palvelun pilvikapasiteetin kustannukset tulevat kasvamaan jonkin verran.

Työkalun toteutukseen käytettyä infrastruktuurikoodia on mahdollista hyödyntää uudelleen tulevissa uudistamisprojekteissa. Tämä nopeuttaa uusien toteutuksien aloittamista merkittävästi. Toteutusta voidaan käyttää esimerkkinä projekteissa, joissa hyödynnetään muitakin pilvipalvelutarjoajia, kuten Azurea ja AWS:ää.

Uusi työkalu on saanut hyvää palautetta Solitan pilviyksikön ylläpitotiimiltä. Mitään suurempia ongelmia ei ole havaittu työkalun käytössä ja uusien valvontojen ohjaussääntöjen lisääminen on ollut vaivatonta. Tämän työkalun uudistaminen on johtanut erään toisen pienemmän sisäisen työkalun pilvisiirtymän Google Cloud -ympäristöön. Tässäkin tapauksessa palvelimettomat palvelut olivat merkittävässä roolissa.

6 POHDINTA

Ennen työn aloittamista oli selkeästi tiedossa, että työkalun uudistaminen halutaan tehdä hyödyntäen Google Cloudin palveluita. Missään vaiheessa ei pohdittu muiden pilvitarjoajien käyttöä, kuten AWS:ää. Jos uudistamisen kohteena oleva palvelu olisi ollut hieman laajempi, olisi vertailu ollut hyödyllistä tehdä. Yksi hyvä vertailukohde olisi ollut muiden pilvipalvelualustojen funktiot-palvelut ja miten niiden ominaisuudet eroavat toisistaan, sekä miten ne integroituvat muihin pilvipalveluihin, jota työkalun toteutus vaatii. Vertailua olisi ollut hyvä tehdä myös infrastruktuurin kapasiteetin kustannusarvioista pilvipalvelualustojen välillä.

Infrastruktuurikoodin toteuttamiseen ei kulunut kauan aikaa, koska pystyttiin hyödyntämään olemassa olevia koodiesimerkkejä. Infrastruktuuri oli kokonaisuudessaan yksinkertainen ja eri komponentteja ei ollut paljon. Koodin testausta helpotti TFLint-työkalu, joka tarkisti Terraform-koodin automaattisesti mahdollisista syntaksi- ja konfiguraatiovirheistä.

Vaikein vaihe työssä oli Python-koodin toiminnallisuuden varmistaminen, vaikka koodirivien määrä oli aika vähäinen. Minulla ei ollut juurikaan Python-kokemusta ennen opinnäytetyötä, joten jouduin nojaamaan aika paljon internetin koodiesimerkkeihin sekä työkavereihini koodin kirjoitusvaiheessa. Onneksi Python on helppolukuinen kieli. Pohja työkalun koodille luotiin tekoälyä hyödyntäen. Tekoälyn tuottamaa koodia ei pystynyt sellaisenaan käyttämään, koska se sisälsi jonkin verran koodivirheitä. Havaitsin kuitenkin, että tekoälyn avulla voidaan nopeuttaa koodin tuottamista ja koodiongelmien ratkaisemista. Ongelmatilanteiden selvittämistä hidasti osittain se, että koodi täytyi joka kerta julkaista Cloud Functionsiin julkaisuputken kautta. Tämä viivytti hieman työn valmistumista. Kehittäjän työasemalla oleva paikallinen kehitysympäristö olisi nopeuttanut koodin ongelmien ratkaisemisessa ja ylipäätänsä koodin testaamisessa.

Työkalun testaus ja käyttöönotto sujui hyvin, vaikka käyttöönotto ei ollut opinnäytetyön toteutuksen aikaan laajamittaista. Käyttöönotto tarkoitti käytännössä muutamien asiakkuuksien uusien valvontojen tekemistä uutta työkalua hyödyntäen. Muut asiakkuudet ja niiden olemassa valvonnat käyttivät vielä vanhaa reverse proxy -toiminnallisuutta, mutta näiden siirtoa varten laadittiin selkeä käyttöönottosuunnitelma.

Toteutuksen tekninen dokumentaatio oli hyvällä tasolla ja sitä laadittiin jo ennen toteutuksen valmistumista. Dokumentaatio on yhtä merkittävässä osassa itse teknisen toteutuksen kanssa. Dokumentaatio on tärkeää tiedonjaon kannalta ja se auttaa muita ihmisiä ymmärtämään, mitä

työkalun odotetaan tekevän, miten sitä käytetään ja miten siihen tehdään muutoksia. Tämän työkalun dokumentaatiossa on muun muassa kerrottu, että miten uusia valvontojen ohjaussääntöjä lisätään koodiin.

Työkalun uudistamisprosessi sujui mielestäni kaiken kaikkiaan hyvin. Työ eteni ennalta sovittua aikataulua nopeammin, mikä osittain selittyy sillä, että työn tekijöillä oli usean vuoden kokemus Google Cloudin palveluiden ja Terraformin käytöstä. Koko uudistamisprosessi toteutettiin noin muutamassa kuukaudessa. Vähemmän kokeneilla vastaavan työn tekeminen voi kestää pitempään. Työn tuloksista saatiin tärkeää tietoa siitä, miten palvelimetonta arkkitehtuuria voidaan hyödyntää vastaavanlaisten kokonaisuuksien uudistamisessa. Lisäksi tuloksista selviää, millaisia resursseja ja osaamista toteutukseen tarvitaan. Opinnäytetyö osoitti myös sen, että tällainen siirtymä perinteisestä konesalista pilveen voidaan toteuttaa määritettyjen tarpeiden perusteella. Toimeksiannosta saatiin uudelleen käytettävää infrastruktuurikoodia myöhempiä ratkaisuja varten.

Näen hyvänä asiana opinnäytetyön kannalta sen, että palvelimettomien ratkaisujen hyödyntäminen ei jäänyt pelkästään teoreettiseksi. Pystyin soveltamaan työelämän esimerkkiä, jolla on todellista merkitystä jokapäiväisessä pilvioperatiivisessa elämässä. Palvelimettomia palveluita voidaan hyödyntää jatkossa entistä enemmän vastaavanlaisissa uudistamisprojekteissa.

LÄHTEET

Amazon Web Services 2024a. What is DNS? Hakupäivä 13.5.2024. https://aws.amazon.com/route53/what-is-dns/.

Amazon Web Services 2024b. What is AWS Lambda? Hakupäivä 29.5.2024. https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

Amazon Web Services 2024c. What is SLA (Service Level Agreement)? Hakupäivä 4.6.2024. https://aws.amazon.com/what-is/service-level-agreement/.

Cisco Systems 2024a. What Is Network Address Translation (NAT)? Hakupäivä 13.5.2024. https://www.cisco.com/c/en/us/products/routers/network-address-translation.html.

Cisco Systems 2024b. What is a WAF? Hakupäivä 13.5.2024. https://www.cisco.com/site/us/en/learn/topics/security/what-is-web-application-firewall-waf.html.

Cloudflare 2024a. What is a reverse proxy? Hakupäivä 13.5.2024. https://www.cloudflare.com/learning/cdn/glossary/reverse-proxy/.

Cloudflare 2024b. What is BaaS? | Backend-as-a-Service vs. Serverless. Hakupäivä 24.4.2024. https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/.

F5 2024. What Is Load Balancing? Hakupäivä 13.5.2024. https://www.nginx.com/resources/glossary/load-balancing/.

GitHub 2024a. Understanding GitHub Actions. Hakupäivä 23.4.2024. https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions.

GitHub 2024b. About GitHub and Git. Hakupäivä 13.5.2024. https://docs.github.com/en/get-started/start-your-journey/about-github-and-git.

Google 2024a. What is Cloud Run. Hakupäivä 18.3.2024. https://cloud.google.com/run/docs/overview/what-is-cloud-run.

Google 2024b. Cloud Functions Overview. Hakupäivä 18.3.2024. https://cloud.google.com/functions/docs/concepts/overview.

Google 2024c. Firestore overview. Hakupäivä 11.4.2024. https://cloud.google.com/firestore/docs/overview.

Google 2024d. Product overview. Hakupäivä 13.5.2024. https://cloud.google.com/armor/docs/cloud-armor-overview.

Google 2024e. Serverless VPC Access. Hakupäivä 13.5.2024. https://cloud.google.com/vpc/docs/serverless-vpc-access.

Google 2024f. Configure minimum instances. Hakupäivä 13.5.2024. https://cloud.google.com/functions/docs/configuring/min-instances.

HashiCorp 2024. What is Terraform? Hakupäivä 13.5.2024. https://developer.hashicorp.com/terraform/intro.

Insight 2024. What is On-Premises? Hakupäivä 13.5.2024. https://www.insight.com/en_US/content-and-resources/glossary/o/on-premises.html.

Kirvan, Paul 2023. Microsoft Azure Functions. Hakupäivä 29.5.2024. https://www.techtarget.com/searchcloudcomputing/definition/Microsoft-Azure-Functions.

Microsoft 2024. What is Saas? Hakupäivä 27.5.2024. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-saas.

Morris, Kief 2024. Infrastructure as Code, 3rd Edition. Hakupäivä 5.2.2024. O'Reilly Online Learning: Academic/Public Library Edition.

Python Software Foundation 2024. What is Python? Executive Summary. Hakupäivä 13.5.2024. https://www.python.org/doc/essays/blurb/.

Red Hat 2024. What is CentOS? Hakupäivä 13.5.2024. https://www.redhat.com/en/topics/linux/what-is-centos.

Roberts, Mike 2018. Serverless Architectures. Hakupäivä 24.4.2024. https://martinfowler.com/articles/serverless.html.

Roper, Jack 2023. What is TFLint and How To Lint Your Terraform Code. Hakupäivä 30.5.2024. https://spacelift.io/blog/what-is-tflint.

Sharma, Alek 2022. What is YAML? A beginner's guide. Hakupäivä 13.5.2024. https://circleci.com/blog/what-is-yaml-a-beginner-s-guide/.

Smart, Thomas 2022. Serverless Beyond the Buzzword: A Strategic Approach to Modern Cloud Management. Hakupäivä 5.2.2024. O'Reilly Online Learning: Academic/Public Library Edition.