Haaga-Helia
ammattikorkeakoulu Oy

Web-sovellus MERN-pinoa hyödyntäen ja Serverless-palvelujen kustannustehokkuusanalyysi

Erik Ilonen

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2021

Haaga-Helia
ammattikorkeakoulu Oy

Tiivistelmä

Tekijä(t)
Erik Ilonen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi
Web-sovellus MERN-pinoa hyödyntäen ja serverless-palvelujen kustannuste-hokkuusanalyysi

Sivu- ja liitesivu-määrä
45 + 2

Pilvipalvelujen suosio on kokenut räjähdysmäisen kasvun maailmanlaajuisesti ja niiden käyttömäärät kasvavat yhä. Niiden käytöstä koituvien kustannuksen selvittämisen on todettu olevan yleisesti haastavaa. Sen takia tämän työn aiheeksi on valikoitunut kyseisten palvelujen käyttökustannusten selvitys.
Pilvipalvelujen hinnoitteluissa käytetään usein termejä kuten ”maksat vain käytöstä”, muistin allokointi ja laskentatehon suuruus. Kyseisten käsitteiden ymmärtäminen on tärkeässä osassa pilvi-palveluiden arkkitehtuurien hallinnassa.
PaaS-, FaaS-, BaaS- ja Serverless-hintoja myös analysoidaan ja niitä verrataan käytännössä saatuihin tuloksiin.

Työssä esitellään ensin työn tutkimuksen taustat sekä keskeisimmät käsitteet. Tämän jälkeen työssä esitellään tutkimukseen keskeisimmin liittyvä teoria, joka sisältää
dokumenttiaineiston internetsivuilta saaduista hintatiedoista ja muuta tietoperustaa. Tätä dataa (ai-neisto) verrataan käytännössä saatuun dataan hinnoista (konstruktio), mikä tarkoittaa sitä, että sitä varten on toiminnallisesti tuotettu MERN-pinolla web-sovellus, jota käytetään hinnan muodostumisen apuvälineenä ja palvelujen räätälöinnissä. Tutkimus toteutetaan laadullisena tutkimuksena, jonka tarkoitus on tarkastella hinnankehitystä syvällisemmin
Tietoperusta sisältää myös toiminnallisen osuuden, jonka tarkoituksena on antaa lukijalle selvempi käsitys kustannustehokkuuden analysointiin käytetystä ratkaisusta.
Dokumenttiaineiston pohjalta vastataan kysymyksiin mitä, miksi ja miten.

Tutkimuksen empriirinen osuus koostuu useista laadullisista analyyseistä.
Aineiston raakadataa on käsitelty matemaattisesti, jotta se olisi vertailukelpoista tuloksen kanssa.
Tulos on, että pilvipalvelujen käyttöönotto tuo pk-yritykselle selkeitä säästöjä, se parantaa laatua ja mahdollistaa hyvin skaalautuvia ratkaisuja. Jo pelkästään laitteen kuten palvelimen sähkön kulutuk-sesta koituvat kustannukset ovat suuremmat kuin serverless palvelujen hinnat. Tämä merkitsee sitä, että pk-yrityksen palvelujen ulkoistaminen Amazoniin, Microsoftiin tai Googleen on kannattavaa.
Yleisesti kustannustehokkuutta ajatellen serverless-pilvipalvelut ovat edullinen muoto korvikkeena täysimittaiselle infralle. Kulut ovat paljon pienemmät kuin kiinteiden laitteiden ylläpidosta koituvat kustannukset. Vaikka hintojen seuranta oli vaikeaa, saatiin aikaiseksi yksinkertaistettu taulukko hin-noille tässä tutkimuksessa.

Asiasanat
MERN, Nodejs, Serverless, Kustannustehokkuus, FaaS, PaaS

Sisällys

1. Johdanto

Maailmalla on viime vuosina räjähdysmäisesti yleistyneet monet pilvipalvelut, joita on yhä enemmissä määrin saatavissa, niin pk-yrityksille, suuremmille yrityksille kuin myös yksittäisille henkilöille. Koska palveluja on paljon tarjolla ja kilpailuakin on melkoisesti, näiden sisältämien palvelujen hinnoittelun vertailu on haastavaa, varsinkin jos palvelujen hankkijana toimii pienempi yritys, yksittäinen käyttäjä tai ryhmä ilman suurempien yrityksien sisältämiä tietohallinnollisia arkkitehtuurin kykyjä tai voimavaroja. Useimmat palvelut niin sanotusti räätälöidään käytöstä perustuviin kustannuksiin ja tämän selvittäminen voi olla haastavaa pienien tai keskisuurien tarpeiden määrittelyssä. Useimmilla palvelun tarjoajilla on usein suurehko lista palveluiden käyttökustannuksista ja niiden ymmärtäminen saatikka vertaileminen voi olla haastavaa monelle palvelun käyttöönottajalle.

Koska palveluja on paljon erilaisille käyttötapauksille, joita ei mitenkään kerkeä kaikkia selvittämään ja vertailemaan tässä projektissa, pitääkin selvittää testimielessä erilaisten palvelujen hinnat rajattuun kohteeseen. Käytännössä tehtiin MERN-pinoa (kts. Kohta 2. Käsitteet) hyödyntävän yksinkertaisen web-sovelluksen, jonka käyttöönottoon liittyviä kustannuksia analysoitiin erilaisissa käyttötilanteissa ja eri palveluntarjoajien välillä. Huomioon otettiin vain kolme suosituinta palveluntarjoajaa, joka oli riittävä otanta kustannustehokkuutta määriteltäessä. Kyseessä oli tällöin toiminnallinen tutkimus. Koska hinnoittelu on käytöstä koituvaa ja pienellä web-sovelluksella ei voi saavuttaa suuria käytöstä koituvia kustannuksia, joutui laskemaan myös kustannusarvioita pohjautuen ilmoitettuihin hintoihin. MERN-pino toteutettiin Docker konttia käyttäen ja upotettiin kaikille pilvipalvelu alustoille. Tietokantana käytettiin MongoDB Atlasta.

Taustatyönä oli perehdytty MERN-pinon arkkitehtuuriin ja oli tehty tili Microsoftin Azureen. Tarvittiin tilit lisäksi Googlen ja Amazonin palveluihin, joista koitui kustannuksia tähän projektiin. Palvelujen käyttö aiheutti kustannuksia noin 20€ virtuaalirahayksikköä.

Tässä projektissa ei vertailla hintoja varsinaisesti ilmaisten- ja opiskelijapalveluiden pohjalta. Tavoitteena oli kuitenkin selkeästi analysoitu kustannustehokkuus eri palvelujen välillä erilaisissa käyttötilanteissa rajaten PaaS-, FaaS- ja BaaS-palvelujen mukaan. Tarkastelun kohteeksi otettiin huomioon myös skaalautuminen ja elastisuus palvelittomien palvelujen yhteydessä.
Hyötynä oli kuitenkin saada kattava perusymmärrys pilvipalvelujen tuomista palvelujen hyödyistä kustannuksia arvioidessa.

Projekti mahdollistaa perusymmärryksen MERN-pinosta, Dockeroinnista ja pientä DevOps-puolen asiaa ja näiden toteuttamisesta. Antaa kattavan kuvan pilvipalvelujen mahdollisuuksista yhdessä kustannustehokkuutta kriittisesti analysoiden, jotta saatu kokonaiskuva mahdollisesti helpottaisi palvelujen valinnassa tietynlaisissa käyttäjätilanteissa. Työ antaa myös kattavan kokonaiskuvan

pilvipalvelujen mahdollisuuksista kustannustehokkuutta kriittisesti analysoimalla, joka mahdollisesti helpottaa palvelujen valinnassa tietynlaisissa käyttäjätilanteissa.

Kuitenkin siten, että tavoitteena oli kustannustehokkuuden analysoiminen pienehkön yksittäisen kuluttajan tarpeisiin tai vaikka pienen yrityksen tarpeisiin, hieman huomioiden skaalautumista ja elastisuutta. Tuotoksena MERN-pinolla tuotettu websovellus, jota käytettiin kokonaan tai osissa erilaisissa käyttötilanteissa pilvipalveluja räätälöidessä ja konfiguroinnissa. Palvelualustojen hintatarkasteluun valittiin Serverless, Google Cloud, Amazon AWS ja Microsoft Azure.

Kaikista työvaiheista on perusteellinen kuvaus: ohjelmistokomponenteista, datasta ja käyttöönotosta on dokumentointia ja tästä saatujen kustannusarvioiden tulos on vertailukelpoista dataa, myös kaaviot ja tutkimustulokset ovat selkeästi kuvattuna.

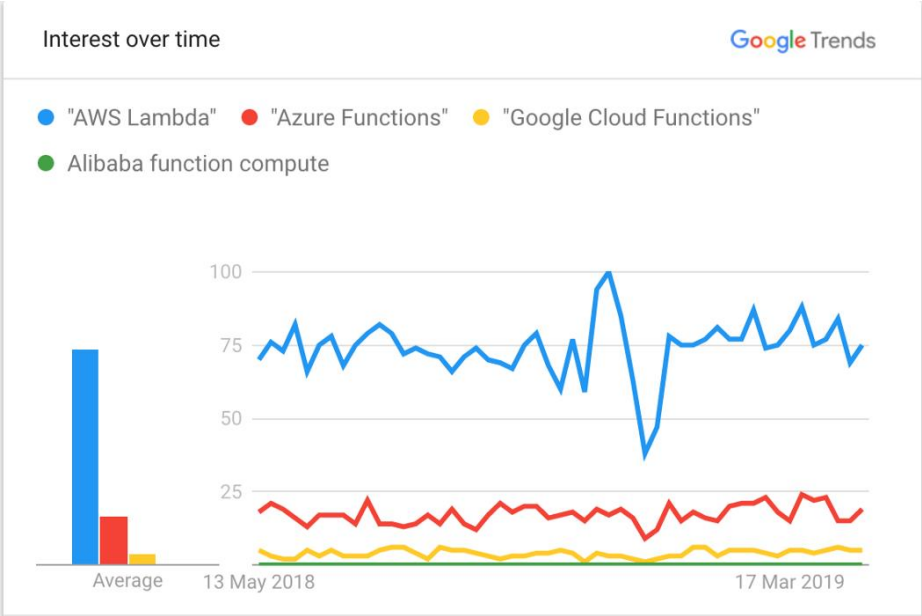
Koska hyvän nykyaikaisen ohjelmistonkehittäjän työkaluarsenaaliin kuuluu kyky hallita kokonaista ohjelmistopinoa parhaassa tapauksessa. Lisäksi yleistyneet pilvipalvelut mahdollistavat pilkottujen kokonaisuuksien hallintaa. Tavoitteena oli oppia näitä hallitsemaan, ainakin perusteellisella tasolla ja koska hinnasto palveluissa on iso osuus, niin sen helppo selvitystyö ja arviointi on kriittisen tärkeää kilpailullisien kykyjen vahvistamisessa. Tavoitteena oli siis oppia full-stack kehittämistä kuten palvelimen ja ulkoasun hallintaa, NoSQL-tietokannan hallintaa, Dockerointia ja kontittamista. Lisäksi oppia tuntemaan konfigurointia ja DevOps-puolen toimintaa eri palvelualustoilla. Myös palvelujen hintojen muodostuminen ja tarkka selvittäminen oli tavoitteena. Kustannusanalyysia tehdessä piti siis kerätä dokumenttiaineisto internetistä (palvelun tarjoajat) luoda oma toimiva web-sovellus ja vertailla näiden hintoja keskenään kriittisesti. Dataa oli paljon ja se on selkeytetty erilaisin kuvaajin ja kaavioin. Kulujen data koostui siis kahdesta lajista vCPU/h laskentaa vaativat kulut ja GB/h muistin allokoinnista vaativat kulut. Näiden käsitteiden ympärillä pyörii pääpiirteittäin koko kulujen määrittäminen. Hinnat laskettiin siten, että kaikista listan hinnoista laskettiin keskimääräinen arvio ja se kerrottiin summana tässä projektissa tuotteen hinnaksi. Kuvat ovat pääsääntöisesti englanniksi, koska sillä kielellä sai enemmän aineistoa.

Koska palveluja on paljon erilaisille käyttötapauksille, joita ei mitenkään kerkeä kaikkia selvittämään ja vertailemaan tässä projektissa, oli tällöin tarve selvittää testimielessä erilaisten palvelujen hinnat rajattuun kohteeseen. Käytännössä tehtiin MERN-pinoa hyödyntäen yksinkertaisen web-sovelluksen, jonka käyttöönottoon liittyviä kustannuksia analysoitiin erilaisissa käyttötilanteissa ja eri palveluntarjoajien välillä. Huomioon otettiin vain kolme suosituinta palveluntarjoajaa (listattu aiemmin), joka oli riittävä vertailun kohde kustannustehokkuutta määriteltäessä. Ei otettu huomioon siis kaikkia palveluntarjoajan käyttöönottokustannuksia, vaan vain ne, joita käytettiin web-sovelluksen yhteydessä ja Serverless-palvelujen hinnat. Hintadataa kertyi muun muassa seuraavista palveluista: Amazon AWS Microsoft Azure ja Google Cloud Platform.

Markkinoilla on ollut jo pitkään suositut palvelut kuten AWS Lambda, Azure Functions ja Google Cloud Functions, ja muut palvelut, joista lisää alla olevassa kuvassa 1. Huomataan, että jokaisella palvelun tarjoajalla on lukuisia serverless palveluja tarjolla.

Kuva 1. Serverless market offerings (Medium 2019)

Lisäksi Google Trends hakukoneesta selviää, että AWS Lambda on ollut niistä suosituin jo pidemmän aikaa. Kuvasta 2. nähdään, että palvelun suosio oli noin kolminkertainen 2018 vuonna hinnan, saatavuuden, moninaisuuden ja palvelun laadun vuoksi.



Kuva 2. Intrest over time. (Medium 2019, Google Trends)

Kuva 2. mukaisesti AWS Lambdan suosio johtui siitä, että Amazon avasi FaaS-palvelunsa kaksi vuotta aiemmin kuin muut palvelun tarjoajat ja Amazonin AWS-pilvipalvelun markkinakoko on paljon suurempi muihin palvelun tarjoajiin verrattuna, joka mahdollistaa suuremmat asiakaskoot. (Eismann, S et al. 2020, 5.).

Alla oleva kuva 3. selittää palvelujen eroja. Kuvasta huomaa, että hinnat ovat lähes samat AWS Lambdalla (hinnasto määräytyy muistin allokoinnista) ja Azure Functionsilla (hinnasto määräytyy muistin käytöstä) Googlen palvelu on selvästi halvin (erikseen laskutetaan muistista ja suorittimesta).

FaaS Comparison

	AWS Lambda	Microsoft Azure	Google Cloud	Alibaba Cloud
Free-tier	1M REQUESTS *per month* 400,000 GB-SECONDS *of compute time per month.*	1M REQUESTS *per month* 400,000 GB-SECONDS *of compute time per month.*	2M REQUESTS *per month* 400,000 GB-SECONDS *of compute time per month.*	1M REQUESTS *per month* 400,000 GB-SECONDS *of compute time per month.*
Pricing	*$0.00001667 FOR EVERY GB-SECOND USED THEREAFTER* *Price changes as per memory allocation*	*$0.000016 FOR EVERY GB-SECOND USED THEREAFTER* *Price changes as per memory usage*	*$0.0000004 FOR EVERY GB-SECOND USED THEREAFTER* *Separate billing for memory and CPU usage*	*$0.00001668 FOR EVERY GB-SECOND USED THEREAFTER* *Separate billing for memory and CPU usage*
Languages supported	Java (OpenJDK 8) Node.js .NET languages (C#, Visual Basic and F#) Python	Java (Azul Zulu Enterprise) Node.js .NET languages (C#, Visual Basic and F#) Python PHP Bash, PowerShell	Node.js Python Go	Java (OpenJDK 8) Node.js C# Python PHP
Execution time and concurrency	1000 executions concurrently with 15 min maxium duration Concurrency by account or individual function.	Unlimited executions with 5 min maxium duration Concurrency by function.	1000 executions concurrently with 1 min maxium duration Concurrency by account or individual function.	100 executions concurrently with 10 min maxium duration Concurrency by account or individual function.
Triggers	Various AWS products (e.g., AWS Kinesis, AWS S3, AWS Dynamo, HTTP requests)	Web API triggering, scheduled invocation, and trigger types performed with other Microsoft services, Azure Event Hub and Azure Storage.	Various Google products (e.g., Google PubSub, Google Storage, Firebase, HTTP requests)	OSS event triggers, HTTP triggers, Timers, CDN event triggers

Pablo Iorio

Kuva 3. FaaS Comparison (Medium 2019).

2. Käsitteet

PAAS

Akronyymi sanoista "Platform As A Service" on pilvessä toimiva palvelu, joka mahdollistaa yleensä jonkin ohjelman pyörittämisen tuotannossa minimaalisin konfiguroinnin. Tarjoaja yleensä tarjoaa verkon, palvelimet, tallennustilan, käyttöjärjestelmän, ajoympäristöt, kuten esimerkiksi Java runtime, .NET runtime, integraatio, tietokanta.
Hyödyt: mahdollistaa korkeatasoisen ohjelmoinnin minimaalisilla vaativuuksilla, joten hallinnointi ja ylläpito on helpompaa.
Haitat: korkea käyttökustannus suuremmissa skaaloissa, vähäinen kontrolli ja mahdolliset ongelmat reitityksessä. (Wikipedia s.a.).

SAAS

Akronyymi sanoista "Software As A Service" Pilvipalvelu, jossa käyttäjän ohjelmisto pysyy ilman tarvetta erilliselle kiintolevylle ja joka on helposti saatavilla.
Esimerkiksi tietokannanhallintaohjelmisto voi olla eräänlainen SaaS palvelu. (Wikipedia s.a.).

IAAS

Akronyymi sanoista "Infrastructure As A Service", joka mahdollistaa virtuaalisen laitteiston käsittelyn, jota voi todella monipuolisesti hallinnoida. Käyttäjä joutuu hallinnoimaan palvelimiaan itse. Tällaisia palveluja tarjoaa esimerkiksi Oracle VirtualBox VM- tai WMware-palvelu tai kontit. Mahdollistaa levykuvien käytön. IaaS on kaikki verkkohallinnoinnin jälkeen tapahtuva hallinnointi, kuten esim. käyttöjärjestelmien hallinnointi ja sen ohjelmiston hallinnointi.
Hyödyt: Kustannustehokkuus ja infrastrukturaaliset säästöt, monipuolisesti joustava, keskeytymätön palvelu.
Haitat: Tietoturvapuutokset, liiallinen riippuvuus ja käyttäjän yksityisyys. (Wikipedia s.a.).

BaaS

Akronyymi sanoista "Backend As A Service" vähemmin käytetty termi (nykyisemmin MBaaS), joka kuvaa lähinnä tietokantapalveluja niin mobiili- kuin webohjelmiston yhteydessä ja siihen liittyviä työkalusettejä. Näitä käytetään usein jonkin IDEn, SDKn tai APIn kautta. Esimerkiksi Google Firebase on sellainen palvelu.
Hyödyt: resurssitehokkuus, ajansäästö, palvelittomuus ja tehokas autentikointi.
Haitat: vähäinen jousto ja pienentynyt kontrolli. (Wikipedia s.a.).

FAAS

Akronyymi sanoista "Function As A Service" on palveluympäristö, jolla käyttäjä voi kehittää, ajaa ja hallinnoida ohjelmiston funktioita ilman tuotantoon viemisen vaikeuksia ja ilman infrastruktuurin ylläpidosta aiheutuvia vaivoja. Liittyy myös mikropalveluihin. Amazonin AWS Lambda oli ensimmäinen tarjoaja. Useimmat FaaS palvelut ovat myös Serverless palveluja. (Wikipedia s.a.).

SERVERLESS COMPUTING

Palvelimetonta käyttömäärään pohjautuvaa palvelun jakamista, ts. palvelimet ja niiden hallinta ovat ulkoistettu palvelun tarjoajalle. Näitä palveluja tarjoavat esimerkiksi Kubernetes, Google App Engine, Amazonin AWS Lambda, Microsoftin Azure tai Cloudflare.
Serverless tietokantapalveluja ovat esim. Firebase, Aurora, MariaDB, PostgreSQL tai Microsoft SQL Server. Serverless palvelujen yhteydessä on kehittynyt fraasi "pay as you go", joka kuvaa että laskutus tapahtuu allokoidun muistin ja ajan käytöstä.
Hyödyt: Yksinkertaisempaa hallintaa ja ero ohjelmistonkehittäjän ja järjestelmäasiantuntijan välillä väljenee. Skaalautuvuuden säädöt helpompia ja vähäisempiä, jota kutsutaan elastiseksi tästä syystä, koska siirtymävaihe prototyypistä maailmanlaajuiseen käyttöön on helpompaa. Eikä tarvitse huolehtia monisäieisyydestä.
Haitat: mahdolliset puutteet tietosuojassa ja yksityisyys. (Wikipedia s.a.).

LEVYKUVA

Eng. Disk-image on tiedosto, johon on tallennettu koko sisältö massamuistista, yleensä varmuuskopiointia varten. Esimerkiksi käyttöjärjestelmä voi olla levykuvana tallennettu asetuksineen. (Wikipedia s.a.).

REST

Akronyymi sanoista "Representational state transfer", joka on eräänlainen standardi, SOAPia hyödyntävä, pohja web-pohjaisen tiedon esittämiselle URIn avulla. Toimii HTTProtokallan yli pyynnöillä GET, PUT, POST ym. (Wikipedia s.a.).

FULL-STACK JA MERN

Ohjelmistopino, joka kuvaa usein tietynlaista järjestelmäarkkitehtuuria tai sen hallintaa, kuten esim. Se voi olla esimerkiksi MERN-pino, joka koostuu kehyksistä ja komponenteista kuten:

MongoDB, Express.js, React.js ja Node.js ja paljon muusta. Täyteen pinoon kuuluu osata sekä front- ja back-end osaamista, eli osata hallita HTML puolta (MCV mallin View-aluetta) ja palvelimen puolta (MVC mallin Model ja Controller aluetta). Täysi pino on sama asia kuin full-stack. (MongoDB s.a.).

MONGODB

Open source monialustainen NoSql tietokantajärjestelmä. Data on dokumenttipohjainen, eli data tallennetaan dokumentteina eikä esim. riveinä ja sarakkeina ja data tallennetaan Json muodossa. Korkeasti skaalautuva datan tallennuksen muodon vuoksi. (MondoDB s.a).

EXPRESS JS

Modulaarinen ja kevyt kehys Node JS:lle backendin pystyttämiseksi. Express on kuin väylä palvelimen ja webpyyntöjen välillä. (MongoDB s.a.).

REACT JS

Avoimen lähdekoodin JavaScript kirjasto frontend puolen hommille saadakseen yksisivuisia sovelluksia. Facebooking tuottama. React-sana tulee siitä kun se on reaktiivinen palautteessaan. (MongoDB s.a.).

NODE JS

Avoimen lähdekoodin JavaScript pohjainen ajoympäristö, joka on suunniteltu ajamaan koodia palvelimen puolella. Node käyttää NPM paketinhallintaa. (MongoDB s.a.).

PILVIPALVELU

Eng. Cloud Computing on laaja kokonaisuus tietoteknisiä palveluja tarjoavaa toimitusta (laskentateho, tallennus, sovelluksia). Yksinkertaisesti datan käsittelyä varten vuokrattavia palveluja (ohjelmistot, laitteet, datakeskukset tai arkkitehtuurit). Käsitteestä käytetään useita erilaisia lyhenteitä, kuten esimerkiksi PaaS. (Wikipedia s.a.).

KONTITUS TAI SÄILIÖINTI

Eng. container on virtuaalinen alusta, jossa voidaan säilöä ohjelmistoa tai käyttöjärjestelmiä luotettavuuden ja tehokkuuden parantamiseksi ympäristöstä toiseen, josta juontuu sana ’kontti’. Kontti tai

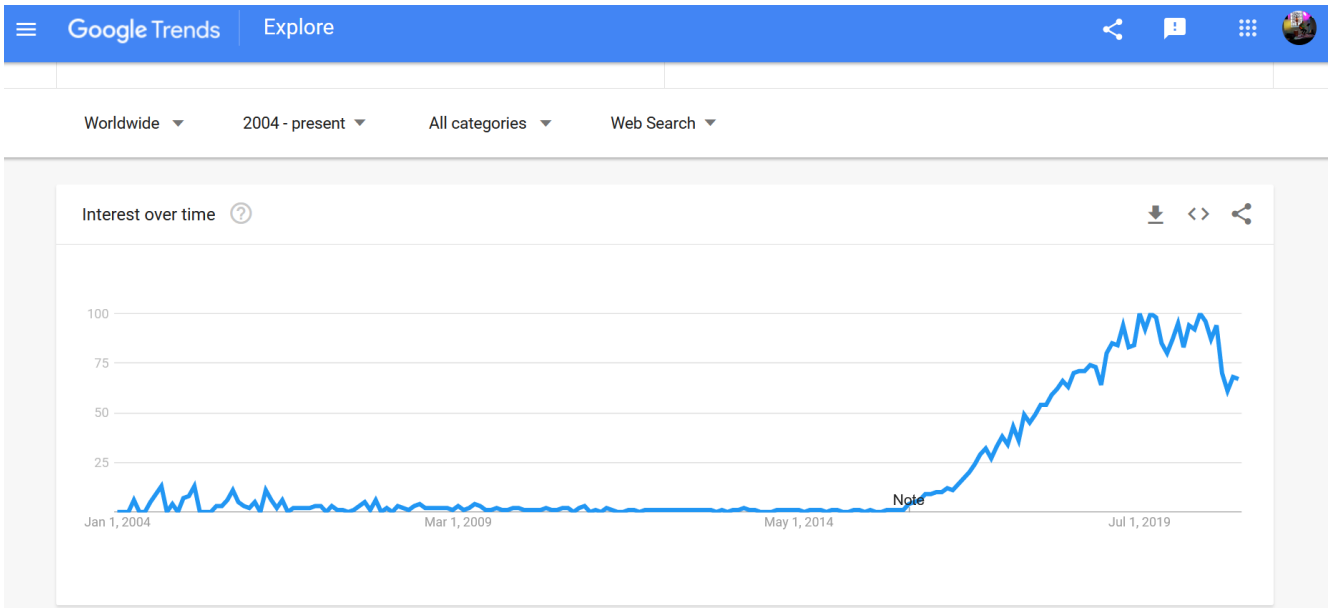
säiliö on yksikkö kokonaisuudelle ohjelmistoa tai ohjelmistoja. Niistä yleensä luodaan levykuva ja ne ovat helppo asentaa toisessa ympäristössä. Esimerkiksi Docker ja Solaris ovat sellaisia palveluja. Virtuaalikoneen ja kontin pääero on siinä, että kontit mahdollistavat tavan virtualisoida käyttöjärjestelmän, jotta moni tapahtuma voi ajaa yhtä käyttöjärjestelmä instanssia, kun taas virtuaalikoneet ajavat laitteen ajureilla useita virtuaalisia käyttöjärjestelmä instansseja. Ts. virtualisointi mahdollistaa abstraktion ja emuloinnin, kun taas kontit mahdollistavat abstraktion ilman emulointia, joka tekee siitä kevyen ympäristön.
Hyödyt: Standardisoitu, kevyt ympäristö ja suojattu. (Docker s.a.).

MIDDLEWARE

Väliohjelmistoa joka toimii ohjelmalogiikan ja käyttöjärjestelmän välissä. Esimerkiksi: brokers, laiteajurit tai JDBC väylä. (Wikipedia s.a.).

3. Teoriaa

Koska ajankohtaisuuden selvittely on tärkeää tällä alalla, on tässä tutkimuksessa haluttu selvittää tietotarpeen ajankohtaisuutta käyttäen apuna mm. Google Trends palvelua.
Alla oleva kuva 4. esittää serverless-palvelun Google -hakumääriä vuosina 2004–2020. Kuvaajasta ilmenee selvästi, että hakumäärät ovat moninkertaistuneet kyseisellä aikavälillä. Varsinaisia käyttäjämääriä ei ole vielä tilastoitu.



Kuva 4. Serverless (Google Trends 2021).

3.1 Serverless ja hinnat

Serverless- ja pilvipalvelujen suosio on viimevuosina kasvanut suosiotaan melkoisesti, varmaankin siksi koska yhä useampi kokee, että oma palvelinympäristö ja sen ylläpitämisestä koituu kalliita kuluja organisaatioille ja tarjolla on keskihintaisia ratkaisuja, tosin data on silloin välikäsien alaisuudessa. Leitner et al. puhuivat työssään *”A mixed-method empirical study of Function-as-a-service software development in industial practice”*, että laskenut operaatiovaiva, laskeneet kulut ja nopeampi kehitystyö suosivat serverless-palveluja maailmalla. Syy ei ole pelkästään kulut, vaan myös hyvä ja tehokas skaalautuminen sekä vähentyneet monitoroinnin taakat. (Eismann, S et al. 2020, 3). Nämä ovatkin hyviä syitä serverless-palvelujen suosioon.

Kun puhutaan palvelimien käyttökustannuksista käytetään termiä ”Total cost of ownership” (TCO). Artikkeli myös mainitsee, että keskimäärin yhden palvelun vuosiylläpitokustannukset ovat $ 731,94 puhumatta investointikustannuksista. Henkilöstön palkka on samaa palvelimien hallinnalle kuin devOps

henkilöstölle (Sherweb s.a.). Tämä ei sisällä palvelimia ylläpitävän henkilöstön kuluja. On huomioitavaa, että näin palvelimien käyttökustannukset ovat jo korkeampia pelkästään sähkönkulutuksen takia kuin useat serverless-palvelujen kulut.

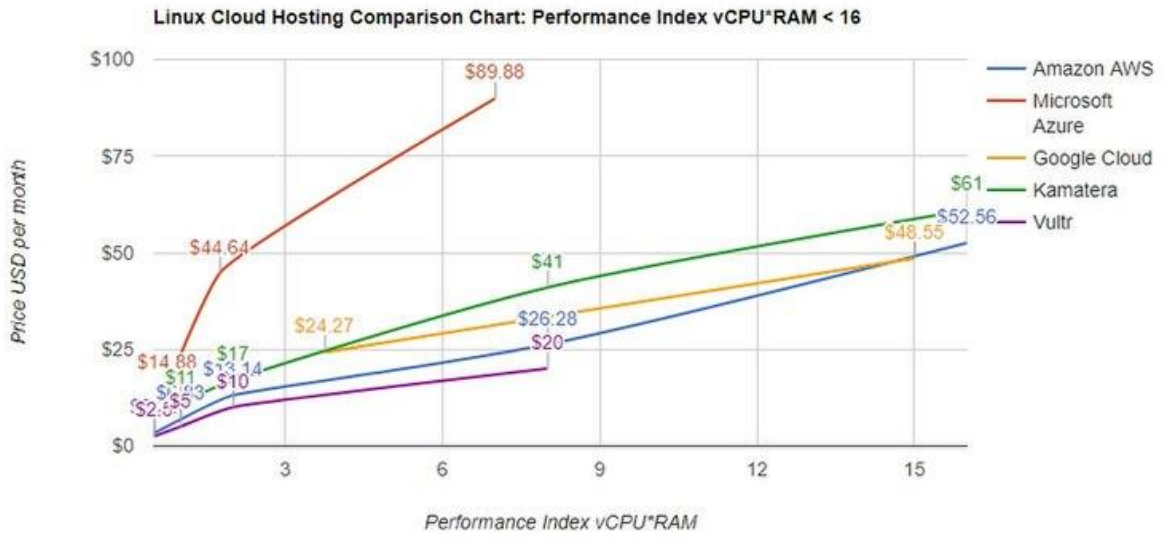
PaaS-palvelun ja serverless-palvelun välillä on eroja, vaikka jotkut serverless-palvelut ovatkin PaaS kategorian alla. Erona on kuitenkin se, että serverless-palvelut ovat skaalautuvuudeltaan enemmän automatisoituja. PaaS työkalut mahdollistavan enemmän itsenäistä hallintaa tuotantoon viemisessä (Zdnet 2019). Vaikka PaaS palveluissa asiakas vastaa ohjelmiston ja datan käsittelystä itse myös serverless-palveluissa näistä vastataan itsenäisesti, erona kuitenkin se, että serverless-palveluissa skaalautuminen on automatisoidumpaa. *Mutta serverless-palvelut useimmiten ovat FaaS-palveluja, kuten AWS Lambda, Azure Functions ja Google Cloud Functions-palvelut*. (Curtis, B s.a.)

Alla olevassa kuvassa 5. näkyy kuinka hinnastot vaihtelevat karkeasti arvioituna erilaisten serverless-palvelujen tuottajilla, joista Amazonin ja Googlen ovat selkeästi halvimpia laskentatehon (vCPU) ja RAM/GB kulutuksen kannalta laskettuna. Tästä voidaan jo ennakoiden päätellä mitä mahdollisesti kustannustehokkuusanalyysi on suhteutettuna. Tämä kuva selventää hintasuhdetta pienemmissä käyttötapauksissa. (Aiheesta vielä lisää alempana.)

Mitä ovat serverless palvelut?

> *” Serverless-palveluja ovat muun muassa Amazon lambda, Azure functions ja Google Cloud Functions. Nämä ovat koodia ja funktioita, jotka eivät sijaitse millään tietyllä palvelimella tai levynkulmalla, vaan funktiot saavat ”valita” mitä servereitä käyttävät, niille määritetyissä rajoissa. Palveluntarjoaja hoitaa liikenteen jakamisen vapaille palvelimille, joten kehittäjän ei tästä tarvitse murehtia.”* (Solita – Rita Lintilä 2017)

Käytännössä fraasi ”**maksat vain siitä mistä käytät**” on yleistä serverless-palveluissa. (Solita – Riita Lintilä 2017).



Kuva 5. Pilvipalvelujen hinnat (Wikipedia s.a. CC BY 2.0).

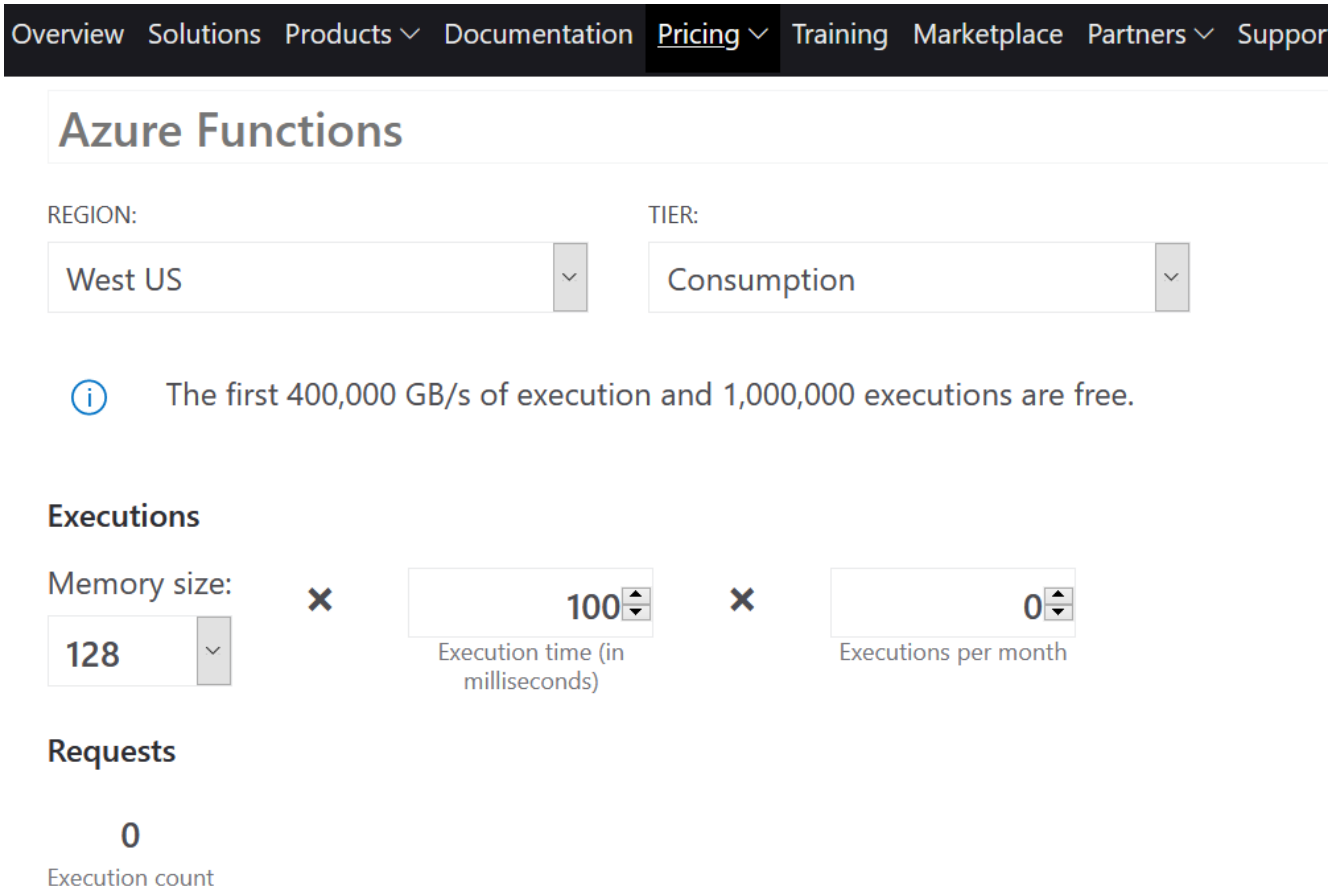
Kuvan 5. mukaan Microsoft Azuren palvelut ovat selkeästi kalliimpia saman laskentatehon (vCPU) ja muistin (RAM) käytön mukaan ja Amazonin ja Googlen välillä pieniä eroja. Tämä valtava ero voi johtua virheellisestä tai subjektiivisesta tulkintatavasta.

Seuraavassa kuvassa 6. on tarkennettuna eroteltuna Amazonin, Googlen ja Microsoftin palveluista muodostuvat hinnat karkeasti arvioituna (hinnat ovat dollaria GB-sekunnissa).

Kuva 6. FaaS PRICING COMPARISON (Altexsoft 2018).

Kuten kuvasta 6. voidaan päätellä, ovat ilmaiset käyttäjätilit lähes identtiset kaikilla muilla paitsi Googlella, joka on tuplana pyynnöissä (2 miljoonaa pyyntöä kuukaudessa ja 400 000 GB-sekuntia laskentakapasiteettia). Mikä tarkoittaa sitä, että saa kuukaudessa tehdä vaikka esim. 2 miljoonaa API pyyntöä ja tai laskentasuorituksia yhteensä 400 tuhatta Gigatavusekunttia kuukaudessa. Asiaa on parempi lähestyä laskemalla laskentatehon kulut sekä datan säilytykseen ja muistiin liittyvät kulut. Kuvasta voi päätellä, että kaiken tämän (Free tier) yli menevästä määrästä laskutetaan $ 0.000016 per GB-s (Azure), mikä on hieman hankalaa hahmottaa yleissilmäyksellä. Lisäksi Azure lupaa halvempaa hintaa Windows käyttöjärjestelmien käyttäjille.

Alla kuva 7. Azuren Functions palvelulle laskimesta, joka on ylivoimaisesti yksinkertaisin palvelun hinnan laskemiseen tarkoitetuista laskimista
(vrt. Googlen ja Amazonin) laskea hintaa palvelulle.



Kuva 7. Azure Calculator (Azure 2021).

3.2 Muut pilvipalvelut

Koska käytännön MERN-webbisovellus sisälsi palvelinpuolen (eng. Backend) NodeJS ja ExpressJS, käyttöliittymän (eng. Frontend) ReactJS ja erillisen MongoDB Atlas tietokannan, en voinut verrata pelkkien serverless-palvelujen kustannuksia tässä projektissa tästä syystä. Kaikkien näiden komponettien tarkoituksena oli viedä ne tuotantoon AWS-, Azure- ja GCP-pilveen. Projektissa on tehty docker levykuvat webbisovelluksesta käyttäen docker-compose työkalua. Lisäksi täytyi erikseen tilata laskentaa vaativa palvelu (IaaS) ja joku palvelu, jonne pystyi upottaa staattisen käyttöliittymän (PaaS). Laskentaa varten valittiin Amazonin EC2, ECR, ECS/FARGATE ja S3 palvelut. Azuren vastaavat ovat Container Registry, Container Instance ja App Services. Googlen vastaavat palvelut olivat Container Registry, App Engine ja Cloud Functions, Cloud Run. Kaikki palvelut skaalautuvat helposti suurempiin organisaatioihin tai käyttötarpeisiin. Amazonin, Azuren ja Googlen palvelut skaalautuvat käyttäen dynamic-autoscale menetelmää, mikä aiheuttaa sen, että ne ovat helposti laajennettavissa (Amazon s.a., Microsoft s.a., Google s.a.). Näiden useiden palvelujen hintaa jouduttiin analysoimaan projektin edetessä. Koska sivujen sisältämät hintojen listat ovat niin moninaisia ja pitkiä. Näistä on laskettu

karkeat arviot ja keskiarvot. Käsitteenä oli lähinnä vCPU/h (eng. virtual centrilized processing unit per hour of usage) ja GiB/h (gibibytes per hour). Myös NoSQL hintoja on tässä huomioitu kuten AWS DynamoDB, Azure CosmosDB ja GCP Firestore.

Vaikka serverless-palvelujen hinnat ovat halvempia se ei kuitenkaan takaa halvempia hintoja joka käyttötapauksessa. Joissakin skaalautumisen kokoluokissa se saattaa olla jopa kalliimpaa käyttää palveluja verrattuna omiin palvelimiin. (Techbeacon s.a.).
Kun pitää ohjelmiston ja businesslogiikan pienenä voi tehdä merkittäviä säästöjä sillä, että siirtyy serverless-palveluarkkitehtuuriin. (Techbeacon s.a.).
Tässä työssä huomioitiinkin yksittäisen kuluttajan ja pienen yrityksen tarpeiden kokoluokat, joten tuloksen pitäisi olla selkeää säästöä havainnoivaa.

3.2.1 Amazon AWS hinnat

- Amazonin EC2 (laskenta) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 1,122 per GiB/h muistin allokointia (Amazon s.a.).
- Amazonin ECR (konttirekisteri) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,07 per GB tallennustilaa (Amazon s.a.).
- Amazonin ECS/FARGATE (konttien käsittely) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,045 per vCPU/h ja $ 0,005 per GB tallennustilaa (Amazon s.a.).
- Amazonin S3 (tallennus) pavelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,023 GB tallennustilaa (Amazon s.a.).
- Lisäksi tarvittavan Route53 DN-liitoksen tietokantaan tekevän palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,30 per kysely (Amazon s.a.).
- Amazonin Lambda palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,20 per miljoona pyyntöä ja $ 0,06 GiB/h laskentaa tunnissa (Amazon s.a.).
- Amazonin DynamoDB (tietokanta) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 1,25 per miljoona kirjausta, $ 0,25 per miljoona kirjausta ja $ 0,25 per GB tallennustilaa. Kaikki tämä on laskettu kuukaudessa olevaksi kuluksi (Amazon s.a.).

3.2.2 Azure hinnat

- Azuren Container Registry (konttirekisteri) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,087 per GB tallennustilaa (Microsoft s.a.).
- Azuren Container Instance (kontin käyttö) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,041 per vCPU/h ja $ 0,004 per GiB/h (Microsoft s.a.).

- Azuren App Services (web-sivun ylläpito) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,20 per vCPU ja GiB/h (Microsoft s.a.).
- Azuren Azure Functions (FaaS) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,058 GiB/h ja $ 0,20 per miljoona pyyntöä (Microsoft s.a.).
- Azuren CosmosDB (tietokanta) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,012 per 100/s kirjausta (per tunti). Lisäksi $ 0,25 per GB tallennustilaa (Microsoft s.a.).

3.2.3 Google Cloud hinnat

- Google Cloudin Container Registry (konttirekisteri) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,026 per GB tallennustilaa (Google s.a.).
- Google Cloudin Cloud Storage (tallennus) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,02 per GB tallennustilaa (Google s.a.).
- Google Cloudin Cloud Run (kontin käyttö) pavelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,104 per vCPU/h ja 0,011 GiB/h (Google s.a.).
- Google Cloudin App Engine (web-sivun ylläpito) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,14 per tunti per instanssi (Google s.a.).
- Google Cloudin Cloud Functions (FaaS) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,40 yli 2 miljoonan menevät pyynnöt ja $ 0,045 per GiB/h (Google s.a.).
- Google Cloudin Firestore (tietokanta) palvelun keskimääräiseksi hinnaksi on laskettu pk-yritykselle $ 0,052 per 100 000 dokumentin luku, kirjoitusta ja poistoa ja $ 0,108 per GB/kk tallennustilaa (Google s.a.).

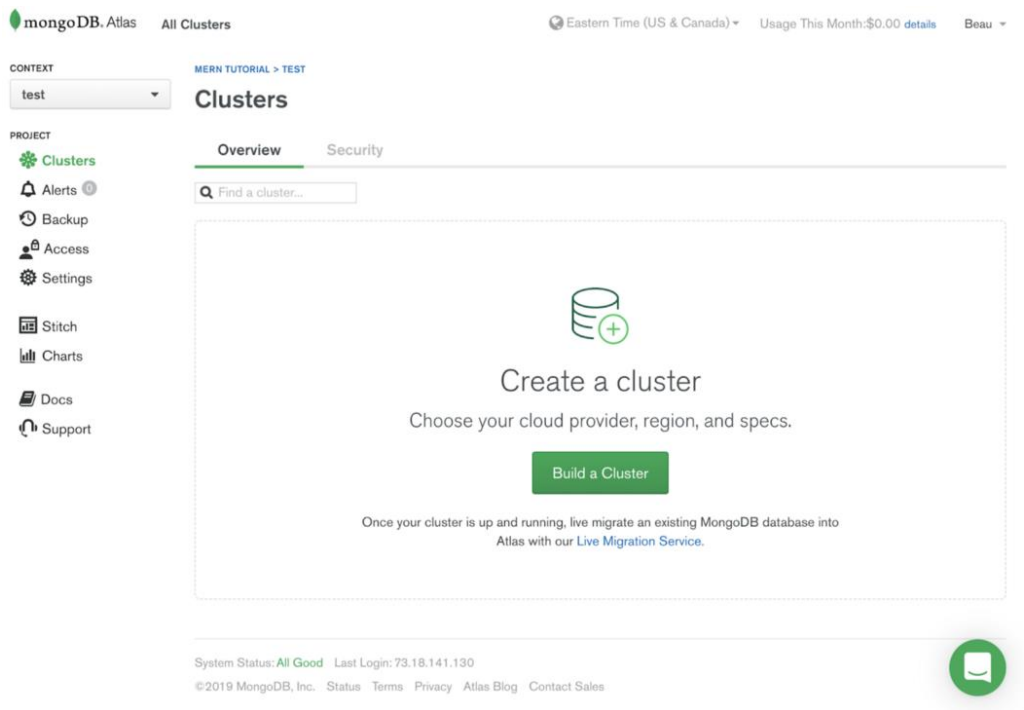
4. MERN Pino ja webbisovellus

Tässä kappaleessa selitetään ja avataan, että mikä MERN-pino on. Lisäksi on luotu esimerkkiverkkosovellus, jota on käytetty kappaleessa kustannustehokkuusanalyysi ja tulokset, kustannustehokkuutta määriteltäessä. Ks. kohta 2 käsitteet ennen tätä vaihetta. Tässä luvussa suuri kiitos Medium.com sivustolle ja käyttäjälle Beau Carnes, jotka mahdollistivat helpon yleiskuvauksen MERN pinoon. (Medium 2019).

MERN pinon komponentit ovat: MongoDB, joka tässä tapauksessa on MongoDB Atlas, Express JS, React ja Node. Lisäksi tietokannan ja sovelluksen välinen kommunikaatio hoidetaan skeema pohjaisella Mongoosella.

4.1 Tietokanta

Aluksi asennettiin MongoDB tietokannan käyttäjätiedot mongodb.com sivulla, josta saa ohjeet toimenpiteiden tekemiseksi. Googlen tunnuksia käytettiin kirjautumiseen ja tilin tekemiseen. Erona relaatiotietokantaan ja dokumenttipohjaiseen MongoDB kantaan on lähinnä yksinkertaisuus käsitellä dataa, MongoDB ei monimutkaisia SQL komentojakaan tarvitse. Dokumentti on JSON dataa sisältävä objekti. Valitse uusi projekti ja anna sille nimi. Sieltä valitse luo tietokanta (Cluster) ja anna sille nimi ja hakemiston nimi. Jonka jälkeen hinnaston säätely ym. On valittu ”free tier”, joka riitti testiprojektiin hyvin.



Kuva 8. Kuvankaappaus (Mongodb 2021).

Sitten valittu ”connect to cluster”. Tuli lisätä listaan IP-osoite, jotta sai yhteyden sovelluksesta tietokantaan (whitelist), lisäksi käyttäjätunnus ja salasana. Sen jälkeen yhdistä applikaatioon. Ja kopioitu osoitteen tietokantaan (Tämä tallennettiin myöhempää tarvetta varten).

Pilvipalvelujen tarjoajilla on myös omat tietokannat kuten: Amazon AWS Aurora, RDS, DynamoDB ja Redshift (Amazon s.a.), Azure SQL Database, Cosmos DB ja useita muita (Microsoft s.a.), Google Cloud SQL, Firestore ja Bigtable (Google s.a.). Palveluja on runsaasti erilaisia, joissa on myös paljon joustoa ja skaalautumisen mahdollisuuksia. Koska projektissa käytettiin NoSQL tietokantaa, otetaan huomioon vain sellaiset tietokannat kustannuksia laskettaessa.

4.2 Node

Asennettiin node ajoympäristö virtuaaliselle Ubuntu 20.14 distrolle
nodejs.org ohjeiden mukaisesti.

Asennettiin Express.js ympäristö. Express on kevyt ja nopea kehys noden palvelinympäristölle. Lisäksi tarvittiin CORS työkalu. Cors on akronyymi sanoille Cross-origin-resource-sharing, jonka avulla voi ohittaa saman lähteen käytännöt ja saada ulkoisia yhteyksiä. Asennettiin myös Mongoosen, josta oli puhetta aiemmin. Sekä dotenvin, joka lataa muuttujia .env tiedostosta ja muuttaa process.env. Toisin sanoen se yksinkertaistaa koodaamista.

Lisättiin Server API Endpointit

Tehty kansio ’routes’ ja sinne tiedostot exercises.js ja users.js.
Lisättiin tiedostoon reitit users.js ja exercises.js, jotka määrittelevät backendin reititykset käyttäen React Routea. Palvelinta oli testattu Postmanilla GET-pyynnöllä http://localhost:5000/users/add osoitteeseen.

```
{
 "username": "Matti"
}
```

Vastauksena tuli paluuviesti: ’user added’.

Muutaman tapahtuman lisäys oli tehty Postmanilla osoitteeseen http://localhost:5000/exercises/add ja body kenttään seuraavaa:

{
"username": "Matti",
"description": "Juoksua",
"duration": 60,
"date": "2019-04-29T21:19:15.187Z"
}

{

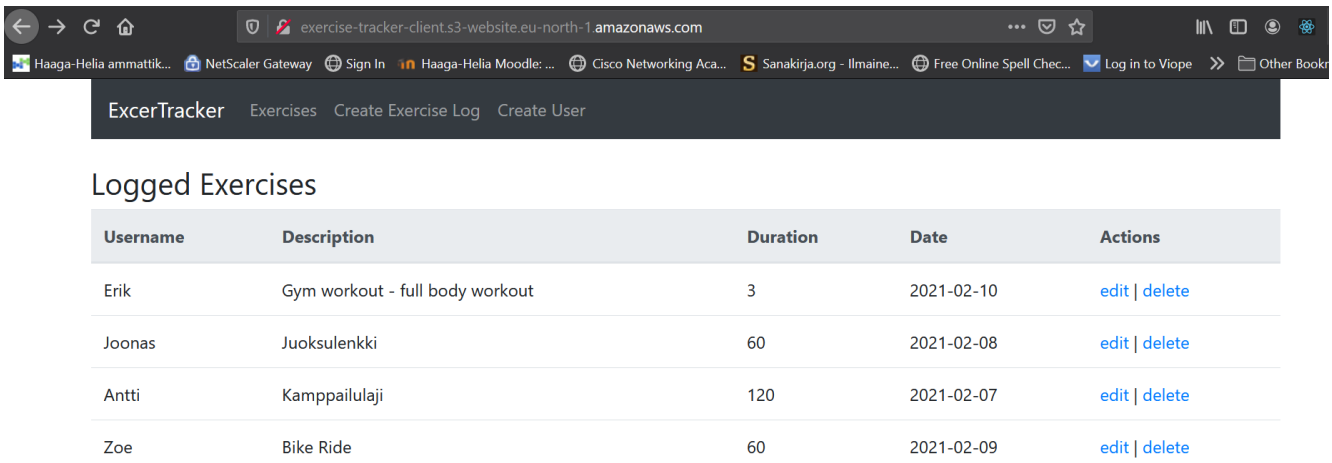
"username": "Matti",
"description": "Pyöräily",
"duration": 20,
"date": "2019-04-30T21:19:15.187Z"
}

Lisäksi oli tehty sovellukseen perus CRUD toiminnallisuutta. Asia on esitetty alla olevassa kuvassa 9.

Kuva 9. Kuvankaappuas (Javascript tiedosto).

Sitten oli tehty käyttöliittymä, johon oli lisätty Bootstrappia. Lisätty datepicker komponentit. Front-End ja Back-End yhdistäminen tapahtui käyttämällä axios työkalua. Palvelimen käynnistämiseen käytin nodemonia.

Lopputuloksena on yksinkertainen ja toimiva ohjelma käyttäjistä ja heidän tekemistään harjoituksista (MongoDB kannassa), jossa on tyylikästä bootstrappia hyödyntävää React.js toiminnallisuutta. Tämän kokonaisuuden tarkoitus oli se, että sitä hyödynnettiin käytännön varmentamisen johdosta kustannus-analyysia laskettaessa.



Kuva 10. Kuvankaappaus (AWS S3).

4.3 MERN-Pinon kontittaminen

Työssä jouduttiin luopua ratkaisusta käyttää AWS Elastic computea (EC2) alustana MERN-pinolle, koska lopputulos ei ollut toimiva MongoDB atlaksen konfigurointien vuoksi ja oli liian monimutkaista devopsaamista. Lisäksi olisi joutunut konfiguroida Azuren ja Google Cloudin kanssa sen omin ehdoin. Asia ratkaistiin siirtymällä suosittuun kontitukseen (kts. Kappale 2. Käsitteet) ja sen käyttämistä yhdessä docker-composea (monikonttityökalu) käyttäen erikseen backendille ja frontendille.

Aluksi tuli asennettua docker ympäristön virtualisoidulle Ubuntu 20.14 versiolle. Jonka jälkeen tuli refaktoroida githubista ladatun projektin. Muutoksena tuli erikseen kansio serverille ja clientille.
Tehtiin Dockerfilet palvelimelle ja käyttöliittymälle.

Sitten näiden yhdistämiseksi tarvittiin docker-composea, joka yhdistää useita kontteja.
Tässä käytettiin konfigurointiin docker-compose.yml tiedostoa. Sisältäen paljon konfiguraatioita.
Konfigurointiin käytettiin tässä työssä docker-compose.yml tiedostoa. Tiedoston sisältämä koodi on esitetty liitteenä Liite 1. Docker compose .yml tiedosto.

Nyt portissa 3000 on toimiva frontti ja portissa 5000 on backend palvelinympäristö. MongoDB on normaalisti portissa 27017 mutta tässä tapauksessa on MongoDB Atlas, jonka yhteys toimii pelkästään .env tiedostolla. (Medium 2020).

5. Tutkimuksen toteutus

Tämän työn tutkimusmenetelmä on laadittu laadullisena toteutuksena, joka merkitsee sitä, että piti viedä rakentama MERN-pino hajautetusti Amazonin AWS, Azuren Portaaliin ja Google Cloudiin ja siitä saatua dataa (konstruktio) analysoitiin ja verrattiin lähtökohtaisesti olemassa oleviin hintatietoihin (dokumenttiaineisto). Tällä vastataan kysymyksiin mitä, miksi, miten sekä tätä tietoa verrataan keskenään.

Tässä oli tavoitteena saada mahdollisimman monipuolinen ja kattava tutkimusdata, joka on vertailukelpoinen ja helposti ymmärrettävässä muodossa. Yrityksen ilmoittamat hinnat ja sen pohjalta käytännön kustannusarvio. Tämän arvion tekemistä vaikeutti se, että hinnat eivät olleet datalle vertailukelpoisia kuten vCPU ja GB-muistin osalta. Aineistoa jouduttiin käsittelemään vertailukelpoisuuden saavuttamiseksi eri palveluntarjoajien välillä.

5.1 Amazon AWS analyysi

Vaikuttaa siltä, että Amazonin AWS palvelu on hyvin monipuolinen ja kattava. Se sisältää niin paljon konfiguraatioita, joka tekee tästä palvelusta todella monipuolisen ja kattavan. Tässä tehtävässä vaadittiin melkoisia ponnistuksia konfiguroinnissa saadakseen palvelut toimimaan. Lisäksi huomasin, että palveluista kertyi melkoisia kustannuksia jo alkuvaiheessa, joita tehty Free Tier tili ei kattanut.

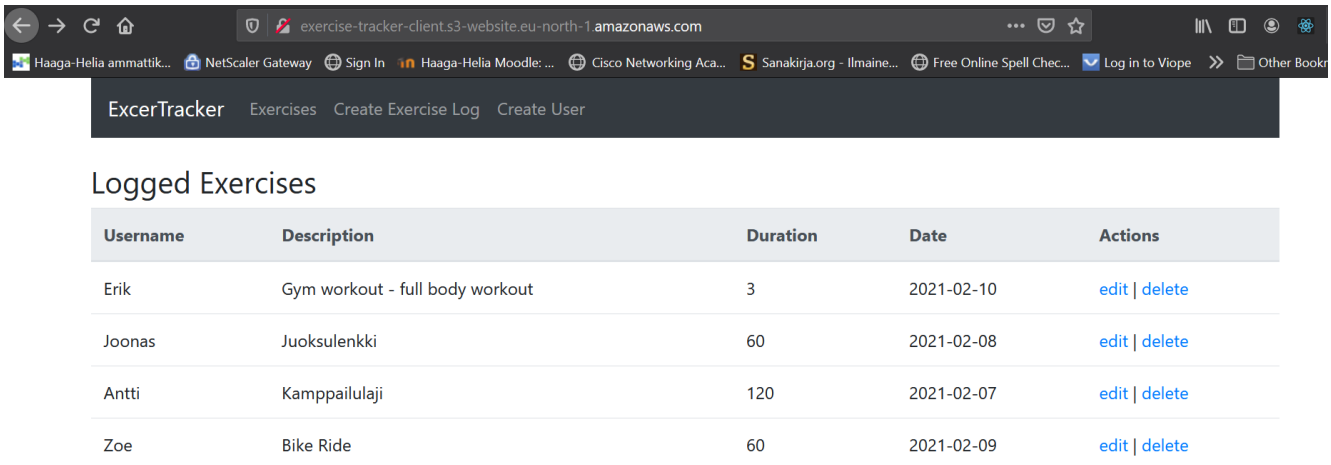
Ensin asennettiin docker ympäristö MERN pinolle ja docker-compose työkalu, jolla saatiin palvelu vietyä vaikeiden ja aikaa vievien konfiguroinnin jälkeen pilveen (ECR), josta joutui erottelemaan backendin (server) frontista (client). Client meni AWS S3 palveluun staattiseksi webbisivuksi, josta ei aiheudu kuluja alle 5 GB datan määrälle ja 20 000 GET pynnölle / kk ensimmäisen vuoden ajan. Server meni AWS ECS/EC2 konttipalveluun, käyttäen yhteyksien helpottamiseksi MongoDB tietokantaan Fargate palvelua. Myös palvelu AWS Route 53 tuli mukaan yhdistämisen johdosta ja DNS käyttöönotosta.

Palvelun olisi saanut upotettua AWS Lambdaan ja serverin EC2 myös, mutta tästä olisi koitunut enemmän kuluja kuin client AWS S3 buckettiin, johon se upotettiin. Valitsin halvemman muodon, koska esimerkkisovellus on pieni kooltaan.

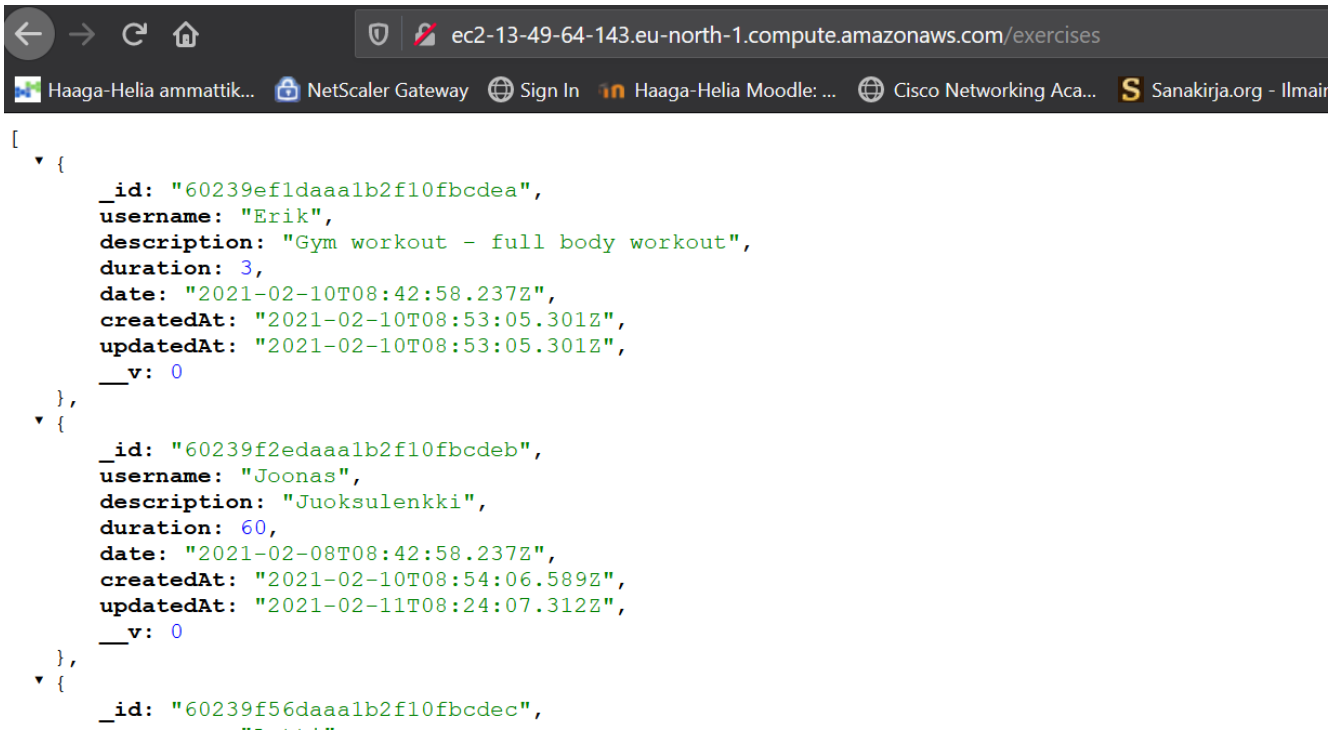
Seuraavat komennot helpottivat melkoisesti Nodessa ja AWS CLI toimintaa:

$ npm run build
$ npm run deploy
Ja package.json tiedostoon piti lisätä:
"deploy": "aws s3 sync build/ s3://exercise-tracker-client --acl public-read"
käyttäen aws-access-id ja aws-secret salausavaimia.
(Medium 2018).

Alla vielä kuvankaappaukset pilvessä pyörivästä MERN-webbisovelluksesta.



Kuva 11. Kuvankaappaus (Amazon AWS S3).

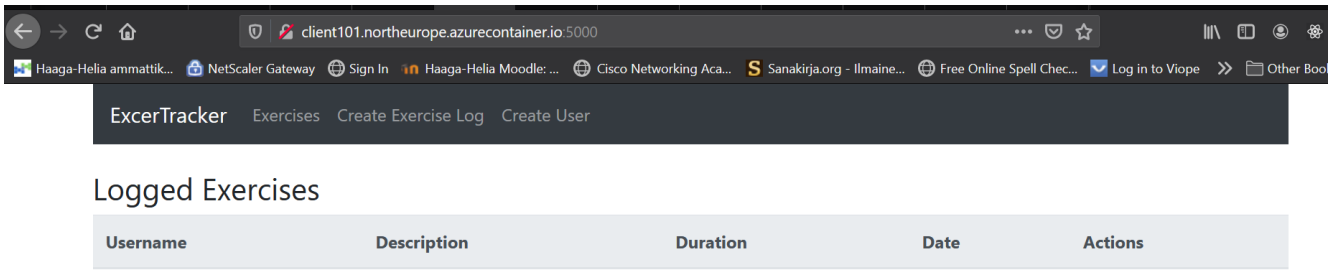


Kuva 12. Kuvankaappaus (Amazon AWS EC2).

5.2 Microsoft Azure analyysi

Azureen oli tehty uusi tili sillä koulun tiliä ei voinut käyttää, koska kaikki palvelut eivät sillä olleet käytössä. Azuren asennukseen tarvittiin käyttöliittymän, joka asennettiin Ubuntu 20.14 distrollen. Docker imageja varten joutui rekisteröimään Azure Container Registry (ACR) , jota varten myös RegistryGroup ja jonne vietiin docker imaget. Imagen viemiseen pilveen käytettiin Docker CLI ohjelmistoa salauksineen.
Sen jälkeen piti luoda Container Instance, johon liitin tekemäni serverin ja clientin ARC:stä. (Microsoft 2020. Tutorial).

Jostakin syystä serveriä ei saanut toimintakuntoiseksi. Se johtuu luultavasti puutteellisesta konfiguroinnista. Tuntien yrityksien jälkeen piti luopua sen pilveen viemisessä. Client lähti toimimaan nyt ilman tietokantaa.



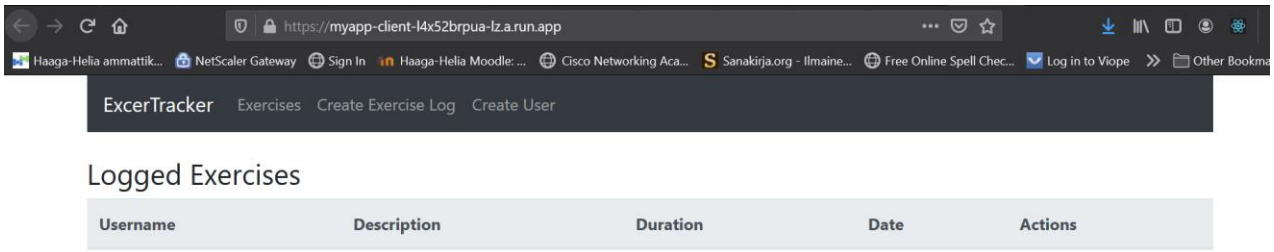
Kuva 13. Kuvankaappaus (Azure Container Instance).

Azuren palvelujen hallinta ja tuotantoon vieminen vaikutti muuten melko yksinkertaiselta. Tässä projektissa käyttämät palvelut olivat: container registry ja container instance (kaksi instanssia).

Tässä olisi voinut hyödyntää Azure App Serviceä, johon olisi saanut sekä serverin että clientin. Tai Clientin Azure Functions palveluun ja serverin App Serviceen, mutta näistä olisi koitunut lisää kuluja. Azurella on myös VM-palvelu, jolla saa kokonaisen infrastruktuurin upotettua.

5.3 Google Cloud analyysi

Googlen palvelujen konfigurointi oli melkoisen työlästä aloittelijalle. Piti lähteä asentamaan CLI ohjelmiston, konfiguroimaan salaukset ja avaimet. Kaikki palvelut vaativat hienosäätöä. Docker-konteilla oli kirjasto Google Cloudin Container registy palvelun käyttöönottoon, eikä sitä oikeastaan muuta kautta onnistunut juurikaan tekemään. Vaativaa I AM ehtojen säätelyä. Palvelut, jotka tuli otettua käyttöön olivat muun muassa: Cloud Storage – Bucket, Container Registry, Cloud run. Serveriä ei saanut toimimaan pitkien säätöjen jälkeenkään. Mutta senkin asennus olisi liitoksissa Container Registry palveluun. Alla oleva kuva 14., jossa client on viety pilveen.



Kuva 14. Kuvankaappaus (Google Cloud Console).

Googlella oli lukuisia palveluja mitä olisi voinut hyödyntää kuten esimerkiksi App Engine ja Cloud Functions, jonne olisi voinut viedä MERN-pinon. Näistäkin olisi tullut lisää kuluja ja siksi se tehtiin halvimmalla mahdollisella tavalla.

5.4 Tutkimusmenetelmä

Tutkimuksessa käytettiin useita laadullisia menetelmiä. Aineiston jäsentämisessä käytettiin useita menetelmiä. Aineistoa kerättiin useista julkisista internet-sivuista ja sitä hyödynnettiin kutakin asiaryhmää varten laadullisesta näkökulmasta. Esimerkiksi hintadata tuli suoraan palvelun tarjoajien nettisivulta raakadatana, jota piti käsitellä saadakseen siitä helposti luettavaa ja selkeytettyä ulkoasua. Data piti myös matemaattisesti muuttaa samansuuruiseksi, jotta se olisi vertailukelpoista. Hakukonetta käytettiin jokaisen selvitystyön pohjalla. Tutkimusmenetelmässä huomioitiin myös hypoteettiset käyttäjätilanteet, joita ei käytännössä toteutettu kuten esimerkiksi: AWS Lambdan, Azure App Service ja Azure Functions, GCP Functions ja App Engine, joita myös hinnaston kannalta oli otettu huomioon. Tutkimuksessa myös spekuloidaan pk-yrityksen pilvipalvelujen tarpeita ja hintoja serverless-palvelujen osalta. Empiriassa käytiin läpi tulokset syvällisemmin ja niitä verrattiin aikaisemmin teoriassa kerättyyn aineistoon.

Tutkimuksen rakenne on sellainen, että aluksi luvussa käsitteet kuvattiin tutkimuksessa käytettyjä käsitteitä, jonka jälkeen taustaa luvussa kuvattiin tietoperustaa, jonka pohjalle rakennettiin toiminnallinen käytännön osuus, ja jota vertaillaan analyyttisesti kustannuksia laskettaessa. Tärkein kriteeri oli kuitenkin varmistaa laadukas lopputulos ja selkeä ymmärrettävyys tuotteiden hinnan ja toteutuksen välillä.

5.5 Tutkimuksen tulokset

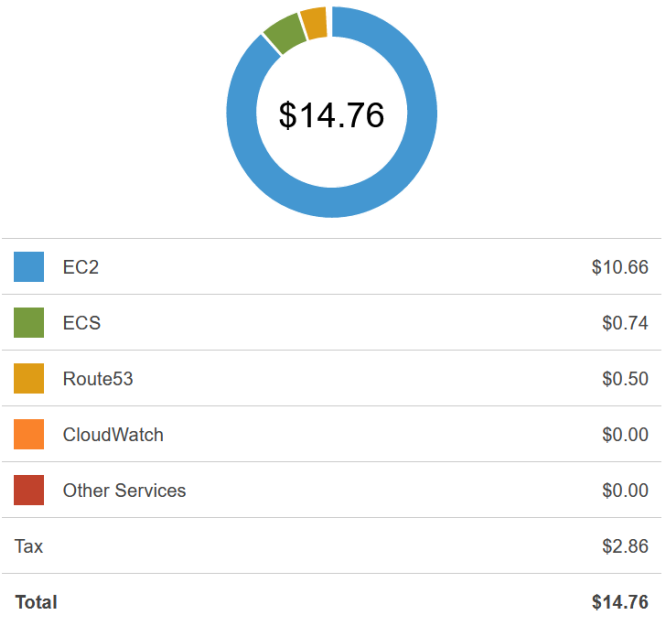
Tutkimuksen tulokset olivat havaintodataa toteutuksen pohjalta, joka vastaa täysin odotuksia tietoperustaan. Datasta ei saatu aikaiseksi mitään uutta merkillistä. Tulokset koostuivat Amazonin AWS, Microsoftin Azuren ja Google Cloudin palvelujen käytöstä aiheutuneista hinnoista. Tulokset ovat käsitelty laskettavaan ja verrannolliseen muotoon, jotta siitä saatiin vertailukelpoista. Varsinaiset tulokset ovat seuraavassa kappaleessa.

6. Kustannustehokkuusanalyysi ja tulokset

Kustannuksia selvitettäessä oli otettu huomioon myös omia kustannuksia, joita koitui pilveen tuotantoon viemisestä, muttei niitä otettu huomioon pk-yritysten kustannusanalyysia laskettaessa. Kaiken kaikkiaan tulokset vastasivat odotusta ja luvussa 3. Teoriaa käytyihin hintoihin, poikkeuksena AWS EC2 palvelu, joka yllätti kuluissa kahden vuorokauden käytöstä. Kulut jakautuivat siten että halvin serverless palvelu oli AWS Lambda ja halvin laskentaa vaativa palvelu GCP (Google Cloud Platform) Cloud Run $ 0,104/h maksava palvelu.

6.1 Amazon AWS tulokset

AWS palveluista koitui MERN-pinolle melkoisia kuluja jo kahden 48 tunnin hostaus käyttöajalle. Alla kuva niistä. AWS-palvelut skaalautuvat melkoisen hyvin ja niihin on lukuisia lisäominaisuuksia. EC2-palvelu oli niistä hintavin. Kahden tunnin hinnaksi koitui: $ 0,615 senttiä (14.76 / 48). ECS palvelulle koitui hintaa $ 0,74, mikä vastaa odotuksia, koska sen palvelun hinta oli $ 0,045 per vCPU/h ja $ 0,005 per GB tallennustilaa. Lisäksi Route53 dns-yhteyden hinnaksi tuli 0,5 dollaria. Kaikkiaan AWS hinnat ovat odotuksen mukaiset.



Kuva 15. Kuvankaappaus (Amazon AWS 2021).

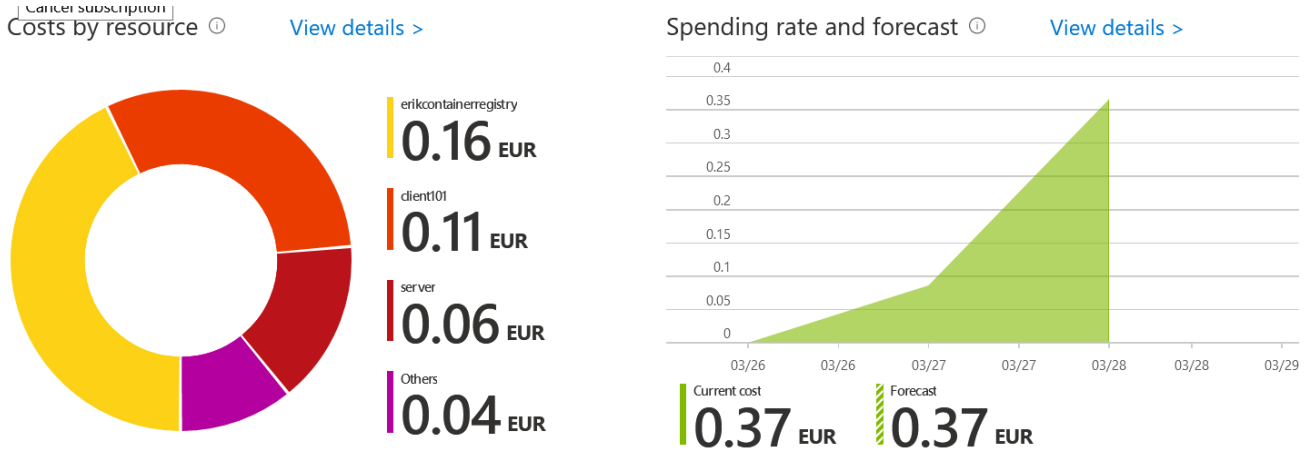
Hypoteettisesti AWS Lambdan (client) ja EC2 (server) olisi voinut toteuttaa tässä projektissa, jolloin niiden kustannukset olisivat olleet kahden tunnin hostauksella $ 1.442, joka on selvästi hintavampi

kuin $ 0,615 nykyisille palveluille. Vaikka AWS Lambda on halpa vaihtoehto, laskentaa vaativa EC2 palvelimelle koitui tässäkin ratkaisussa kalliiksi käyttää.

Esimerkiksi noin miljoona pyyntöä kuukaudessa tekevä pk-yritys keskimääräisin varustein maksaa AWS pilvessä AWS Lambdan osalta $ 0,20 per miljoona kyselyä (miljoona pyyntöä kuukaudessa) on silloin $ 2,40 pyynnöt vuodessa ja $ 0,06 GiB/h hinnalla on $ 525,26 vuodessa. Eli kaikkiaan pk-yrityksen vuosikulut olisivat silloin $ 527,66 vuodessa. Mikä on silti vähemmän kuin aikaisemmin mainittu $ 731,94 dollaria palvelimen sähkönkulutuksen vuosiylläpitokustannus. Tässä selvästi tekisi pk-yritys miljoonalla GET-pyynnöllä kuukaudessa säästöä. Mikäli tietokannaksi pk-yritys olisi valinnut AWS DynamoDB miljoonalla tietokannan kirjauksella kuukaudessa ja 50 GB tallennustilalla, josta vähennetään ensimäiset 25 GB, tällöin kuukausihinnaksi olisi muodostunut $ 9,375 mikä myös vähemmän kuin oman palvelimen pyörittäminen. Hinta on ilmaisen hinnan ylittävän osuuden jälkeinen hinta.

6.2 Microsoft Azure tulokset

Azuren palveluista koitui kustannuksia epäilyttävän vähän, vaikka palvelut olivat käytössä kaksi tuntia. Ottaen huomioon, että backend ei ollut päällä virheen vuoksi. Tästä syystä Microsoft Azuren palvelujen hinnastot käytännön osalta eivät ole täysin vertailukelpoisia Amazonin AWSn kanssa tällä hetkellä käytännön osalta ja projektin kannalta. Kuitenkin Konttirekisterin hinnaksi koitui 0,16 euroa ja kahdelle kontille hinnaksi 0,11 eur ja 0,06 euroa. Hinta on $ 0,087 per GB tallennustilaa ja kontti instanssille $ 0,041 per vCPU/h ja $ 0,004 per GiB/h. Karkeasti laskettuna konttipalvelujen hinta olisi pitänyt olla 0,528 euroa vaikka tuossa se on yhteensä 0,33 euroa per kaksi tuntia.



Kuva 16. Kuvankaappaus (Azure Portal).

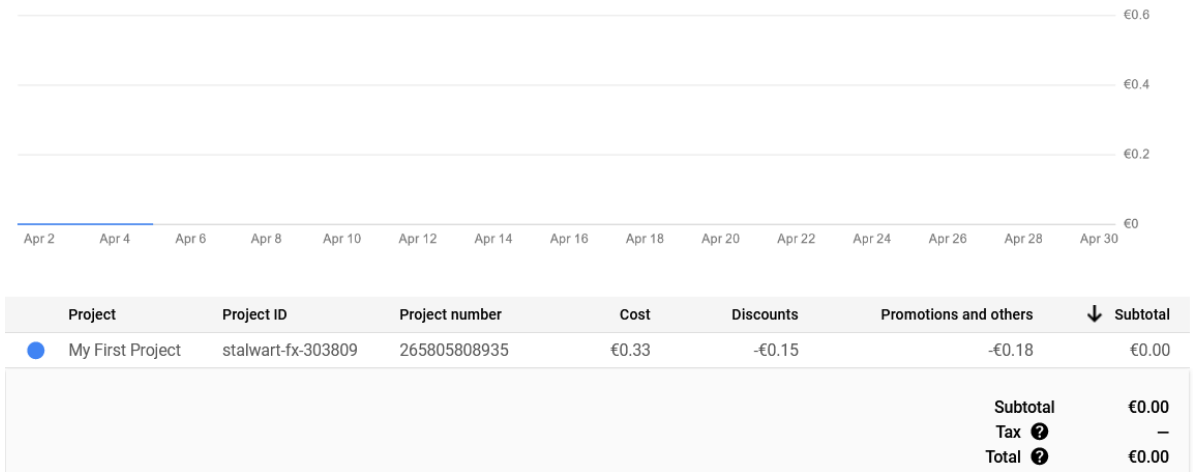
Kuvasta 16. huomataan. että konttipalvelun hinta (€ 0,16) on huomattavasti AWS palvelua halvempi ($ 0,74). Googlen vastaava palvelu on $ 0,052, mikä tekee sen halvimmaksi kaikista. Tässä laskutettiin toimimattomasta palvelimesta, mikä merkitsee sitä, että vaikka palvelu ei toimisi niin sen laskennasta peritään maksu. Koska kone ei osaa erottaa toimivan ja toimimattoman väliltä silloin kun se vaatii laskentaa käyttöönotossa.

Azuren hypoteettinen ratkaisu olisi voinut sisältää Azuren App Servicen tai Azure Functions palvelut, joiden hinnaksi olisi koitunut kahden tunnin hostauksella: $ 0,40 (App Service) tai $ 0,316 (Azure Functions). Hypoteettiset hinnat olisivat olleet kalliimmat kuin nykyinen ratkaisu konttipalveluja käyttäen.

Esimerkiksi noin miljoona pyyntöä kuukaudessa tekevä pk-yritys keskimääräisin varustein maksaa Azure pilvessä Azure Functions osalta $ 0,20 per miljoona kyselyä (miljoona pyyntöä kuukaudessa) on silloin $ 2,40 pyynnöt vuodessa ja $ 0,058 GiB/h hinnalla on $ 508,08 vuodessa. Eli kaikkiaan pk-yrityksen vuosikulut olisi silloin $ 512,48 vuodessa. Mikä on huomattavasti vähemmän kuin aikaisemmin mainittu $ 731,94 dollaria palvelimen sähkönkulutuksen vuosiylläpitokustannus. Tässä selvästi tekisi pk-yritys miljoonalla GET-pyynnöllä kuukaudessa säästöä. Mikäli pk-yritys olisi käyttänyt lisäksi tietokannassa Azuren CosmosDB palvelua, hinnaksi olisi tällöin keskimäärin miljoonalla tietokantakirjauksella kuukaudessa tai 100RU/s ja 50 GB tallennustilalla muodostunut $ 8,64 kuukaudessa ja $ 12,5 kuukaudessa, joka olisi silloin $ 21,14 kuukaudessa. Tämä on paljon edullisempaa kuin oman tietokantapalvelimen pyörittäminen. Hinta on ilmaisen hinnan ylittävän osuuden jälkeinen hinta.

6.3 Google Cloud tulokset

Google Cloud pavelut tulivat hyvin edullisiksi, joista koitui kahden tunnin hostauksesta $ 0,33 senttiä hintaa ja nekin vähennettiin alennuksessa. Googlella on selvästi näistä edullisimmat palvelut. Googlen laskelman ulkoasu oli kaikista heikoin. GCP (Google Cloud Platform) palveluja oli käytössä Container Registry, jonka hinta on $ 0,026 per GB tallennustilaa, Cloud Storage $ 0,02 per GB tallennustilaa ja Cloud Run palvelu, jonka hinta on $ 0,104 per vCPU/h ja 0,011 GiB/h. Karkeasti laskettuna kahden tunnin käytöstä koituisi hinnaksi yhteensä 0,3, mikä vastaa suurin piirtein palvelun käytännössä saatua hintaa.



Project	Project ID	Project number	Cost	Discounts	Promotions and others	Subtotal
My First Project	stalwart-fx-303809	265805808935	€0.33	-€0.15	-€0.18	€0.00

Subtotal	€0.00
Tax	–
Total	€0.00

Kuva 17. Kuvankaappaus (Google Cloud - Console – Billing).

Google Cloud palvelujen hypoteettiset palvelut olisivat voineet olla esimerkiksi: App Engine tai Cloud Functions, joiden hinnat olisivat voineet olla kahden tunnin hostauksella luokkaa: $ 0.28 (App Engine) tai $ 0,49 (Cloud Functions). Kuten nähdään App Enginen hinnasta, että tätä palvelua käyttäen olisi voinut säästää useita kymmeniä senttejä kahden tunnin käytöllä. Cloud Functionsin käyttö olisi koitunut kalliimmaksi kuin nykyinen ratkaisu.

Esimerkiksi noin miljoona pyyntöä kuukaudessa tekevä pk-yritys keskimääräisin varustein maksaa Google Cloud pilvessä Google Cloud Functions osalta $ 0,40 per 2 miljoona kyselyä (miljoona pyyntöä kuukaudessa) on silloin $ 2,40 pyynnöt vuodessa ja $ 0,045 GiB/h hinnalla on $ 394,2 vuodessa. Eli kaikkiaan pk-yrityksen vuosikulut olisivat silloin $ 396,6 vuodessa. Mikä on merkittävästi vähemmän kuin aikaisemmin mainittu $ 731,94 dollaria palvelimen sähkönkulutuksen vuosiylläpitokustannus. Pk-yritys tekisi miljoonalla GET-pyynnöllä kuukaudessa säästöä. Halvimmat vuosikustannukset kaikista palvelun tarjoajista tämän laskun mukaan. Mutta todellisia hintoja mitatessa konfigurointi määrää lopullisen hinnan ja siinä saa optimoida melkoisen tarkkaan halutun tuloksen saavuttamiseksi. Tästä voidaan päätellä, että keskisuuren pk-yrityksen kaupallisen webbisivun hostaaminen miljoonalla GET-pyynnöllä kuukaudessa maksaisi kaikkien näiden palveluiden tarjoajien mukaan vähemmän, kuin pelkän palvelimen sähkön kulutus. Mikä on ihmeellistä tätä asiaa teoreettisesti katsoen. Kulut ovat 396,5 – 525,26 dollarin välillä, vaikka keskimääräinen Yhdysvaltojen vuosisähkönkulutuksen hinta palvelimelle on 731,94 dollaria kuukaudessa. Säästöä on Googlen osalta jopa yli 300 dollaria vuodessa puhumattakaan niiden hallinnollisia kustannuksia. Mikäli tietokannaksi olisi valittu GCP Firestore, niin kulut miljoonalle kirjaukselle kuukaudessa ja 50 GB tallennustilan hinnaksi olisi muodostunut silloin $ 10,34 kuukaudessa (0,052 * 95 + 50 * 0,108). Hinta on ilmaisen hinnan ylittävän osuuden jälkeinen hinta.

6.4 Kustannustehokkuusanalyysi

Kokonaisuudessaan palvelujen hinnat olivat melko vaihtelevia ja eivätkä ole täysin vertailukelpoisia, koska Azuren ja Googlen osalta palvelinpuolen toiminnot eivät tulleet huomioiduksi. Kallein oli selvästi Amazonin AWS hintaan $ 0,615 senttiä kahden tunnin hostauksella. Toiseksi hintavin oli Azuren palvelut hinnalla € 0,37 ($ 0,43) senttiä kahden tunnin hostauksella. Kolmanneksi Google Cloud palvelut hinnalla € 0,33 ($ 0,39) senttiä kahden tunnin hostauksella. Hintarakenteesta voi päätellä, että nämä tulokset vastaavat alussa esiteltyyn laskelmaan, vaikka yhden palvelimen kustannukset puuttuivat Azuren ja Google Cloudin käytännön kokeilusta.

Alla olevassa kuvassa 18. on aiemmin mainittujen palvelujen hintoja suoraan palvelun tarjoajien sivuilta, joita voi verrata käytännössä muodostuneisiin kustannuksiin. Hinnat muodostuvat lasketuista keskiarvoista tunnin käyttöajalle huomioiden vCPU, GiB/h, GB-tallennustilaa ja pyyntöjen määrä.

Amazon AWS Services	Average price for SME = *	**EC2**	**ECS/Fargate**	**ECR**	**S3**	**Route 53**	**Total**	**Ilmaiset: Free Tier**
$ Price / GB Storage	*		0,005	0,070	0,023			Lisätietoa kenttä ilmaisille palveluille
$ Price / vCPU / Hour	*		0,045					S3 storage: < 5 GB + < 20 000 -
$ Price / Host or Query	*					0,300		GET pyyntöä / kk / vuosi
$ Price / GiB Memory / Hour	*	1,122						EC2: 750 tuntia käyttöä t2.micro
Total		**1,122**	**0,049**	**0,070**	**0,023**	**0,300**	**1,565**	ECR: 500 MB / kk / vuosi
Microsoft Azure Services	Average price for SME = *	**Container registry**	**Container Instance**	**Resource Groups**	**App Service**		**Total**	
$ Price / GB Storage	*	0,087		N/A				App Service - < 1 GB free
$ Price / Hour	*			N/A	0,200			Azure free $ 200 credits for 30 days
$ Price / vCPU / Hour	*		0,041	N/A				
$ Price / Host or Query	*			N/A				
$ Price / GiB Memory / Hour	*		0,004					
Total		**0,087**	**0,045**	**0,000**	**0,200**		**0,668**	
Google Cloud Services	Average price for SME = *	Container registry	Cloud Storage	Cloud run			**Total**	
$ Price / GB Storage / Month	*	0,026	0,020					Google Cloud – free $ 300 / 90 days credit
$ Price / Hour	*							
$ Price / vCPU / Hour	*			0,104				
$ Price / Host or Query	*							
$ Price / GiB Memory / Hour	*			0,011				
Total		**0,026**	**0,020**	**0,114**	**0,000**		**0,160**	

Kuva 18. Kuvankaappaus taulukko (AWS, Azure, Google Cloud).

Kaikkien palvelujen yhteiset kustannukset koituivat siten, että Amazon AWS-palveluista koitui $ 1,565 kustannuksia. Microsoft Azuren kaikki kustannukset ilman palvelinpuolta koitui $ 0,668. Google Cloudin kaikki kustannukset ilman palvelinpuolta koitui $ 0,160. Mikäli AWS-palvelusta vähennetään myös palvelimen kustannukset en olisi silloin tarvinnut EC2 palvelu, jolloin kustannukset olisivat $ 0,443.

Konttien rekisteröinnin ja tallennuksen kulut olivat järjestettynä halvimmasta kalleimpaan järjestyksessä: GCS $ 0,026, AWS $ 0,07 ja Azure $ 0,087, jotka olivat hyvin lähellä toisiaan. Kontin ajamisen kulut olivat halvimmasta kalleimpaan seuraavasti: $ 0,045 (Azure), $ 0,049 (AWS) ja GCS $ 0,11, jossa Googlella oli kaksinkertaisesti korkeammat kulut.

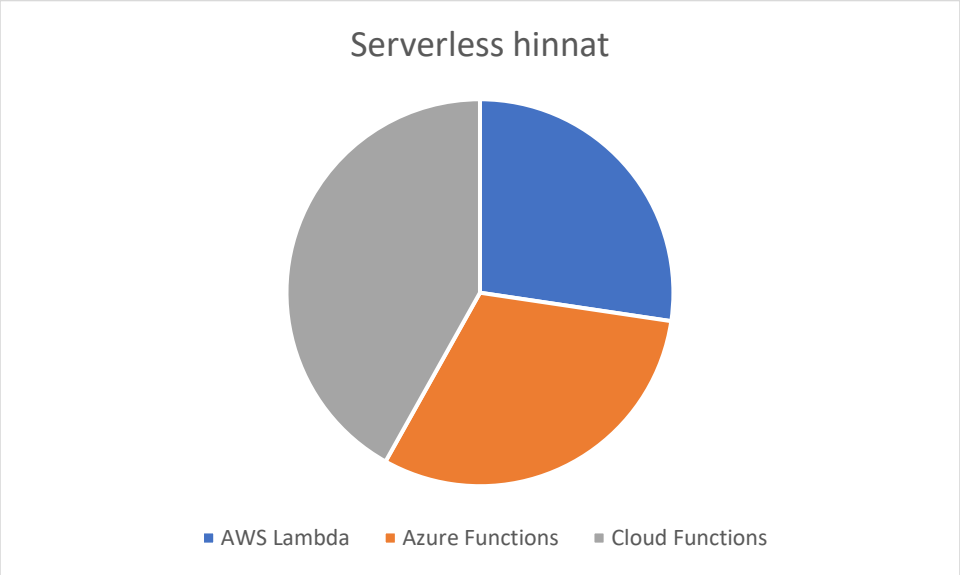
Serverless palvelujen hinnat ovat luokkaa: AWS Lambda $ 0,06 per GiB/h, Azure Functions $ 0,058 ja Google Cloud Functions $ 0,045 GiB/h. Joista Googlen palvelu on halvin, toiseksi Azuren ja sitten Amazonin. Satuin myös huomaamaan, että esittelemä kuva 5. Wikipedian laskelmista serverless palveluille eivät täysin pidä paikkaansa. Palvelujen hinnoissa kärkipaikan kalleudelle otti
Merkittävällä erolla muihin Azuren Functions, toiseksi AWS Lambda ja kolmanneksi Google Cloud Functions, mikä pitää täysin paikkaansa mutta suhteellisuus on kaukana toteutuksesta, sillä saadun tuloksen pohjalta laskettuna keskihajonta on luokkaa 0,0081, eli hinnat määräytyvät todella lähelle toisiaan. Kun taas kuvan perusteella keskihajonta on luokkaa 35.2 mikä tarkoittaa sitä, että vaihtelua on paljon hinnoissa. Kuvan 5. tuloksiin on varmasti vaikuttanut subjektiiviset mielipiteet.
Kuten aikaisemmin mainittu palvelimen ylläpitohinta on $ 731,94 jo pelkästään sähkön kulutuksen kannalta. Ylläpitohinta per tunti on karkeasti laskettuna $ 0,085. Hinta määräytyy jo pelkästään sähkönkulutuksen kannalta kalliimmaksi kuin monien serverless palvelujen tunnin käyttöhinta. Mikä on arveluttavaa, kun puhutaan suurista data keskuksista, joissa myös palvelimet käyttävät sähköä. Luultavasti käytetään palvelimia matalamman kustannustason maissa.

Hypoteettisena tuloksena (mikäli olisi ottanut palvelut käyttöön) koituneet kustannukset olisivat olleet muun muassa seuraavat:

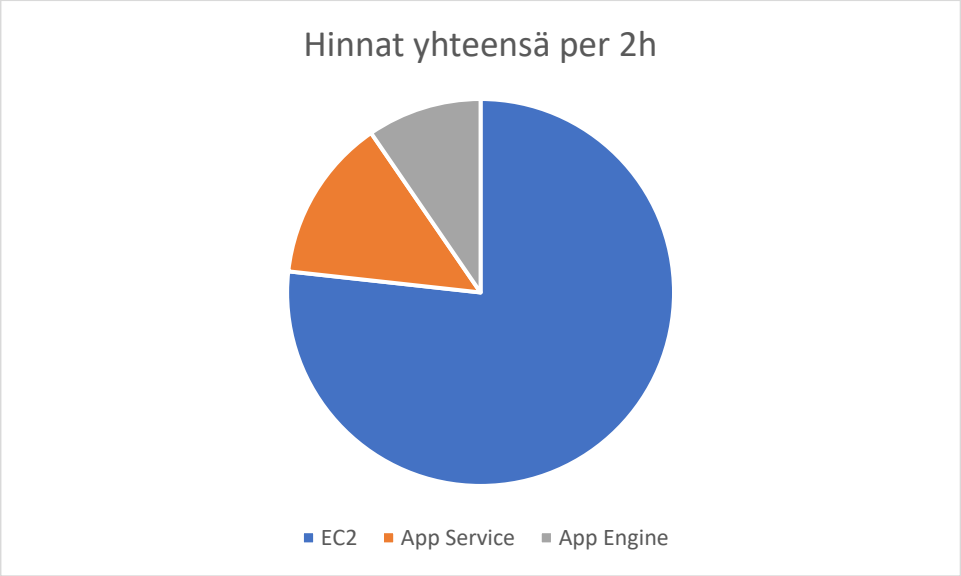
Taulukko 1. Hypoteettiset serverless-palvelut

Amazonin palveluilla:	EC2	AWS Lambda
yhteensä per 2h	2,244	0,320
Microsoftin palveluilla:	App Service	Azure Functions
yhteensä per 2h	0,4	0,360
Googlen palveluilla:	App Engine	Cloud Functions
yhteensä per 2h	0,28	0,490

Taulukon 1. hinnat ovat dollaria per kahden tunnin käyttöajalle.



Kuva 19. Kuvankaappaus ympyräkaavio (AWS, Azure, Google Cloud).



Kuva 20. Kuvankaappaus ympyräkaavio (AWS, Azure, Google Cloud).

Kuten kuvasta 20. nähdään, että laskentaa vaativat suoritukset ovat ylivoimaisesti kalleimmillaan Amazonilla (EC2) ja Googlen App Engine halvin. Serverless-palvelujen hinnat (Kuva 19.) ovat melko samansuuruisia kuitenkin siten, että halvin on AWS Lambda ja kallein on Google Functions. Nämä arviot olivat laskettu käyttäen aiemmin esiteltyä aineistoa kahden tunnin hostauksella ilmeneviksi hinnoiksi. Mikäli laskentaa haluaa ottaa käyttöön Amazonin AWS palvelu on kalleinta ja siinä kannattaa varautua korkeisiin kustannuksiin verrattuna kilpailijoihin. Amazon kompensoi sitä vuorostaan serverless-palvelujen hinnoissa, jotka ovat edullisimpia kaikilta kilpailijoilta. Laskennassa halvimmalla sijalla on Googlen App Engine, kun taas serverless-puolella Googlen palvelu on kallein. Tässä joutuukin valitsemaan myös mikä on tärkein ominaisuus palvelua valittaessa. Tässä työssä koitui helpoittavaksi

ominaisuudeksi se, että Amazonin palvelujen käyttöönoton monipuolisuus ja selkeät ohjeet olivat riittävät. Myös se auttoi, että koska asiakaskunta oli suurempi, oli myös ohjeiden laatijoita enemmän, mikä helpotti palvelujen käyttöönottoa huomattavasti tehokkaammin kuin kilpailijoilla.

Hypoteettisten palvelujen käyttöönotossa olisi säästänyt vain Googlen App Enginen tai AWS Lambdan käyttöönotoilla halpojen hintojen vuoksi. Muut palvelut olisivat koituneet kuluja nostattavaksi kriteeriksi. Etenkin tuon Amazonin laskentaa vaativan EC2 palvelun käytöstä koitui molemmissa tapauksissa korkeita kuluja verrattuna muihin kilpailijoihin. EC2 korkeat kulut käytännössä saattavat johtua siitä, että niissä oli käytetty laajempaa allokointia, joka on kalliimpi kuin pienemmän koon muistilla varustetut palvelut. Vaikka lasketut keskiarvot pohjautuvat keskiarvoon, olivat käytännön laskelmat kalliimpia kuin ne ovat. Ainakin hyvänä puolena Amazonin palveluissa on se, että niitä on melkoisen riittävästi mahdollista konfiguroida. Mitä taitavampi tässä on, sen halvempia hintoja on mahdollista tällä säästää itselleen. Kaiken kaikkiaan palvelut olivat hyvin kattavia kaikilla palvelun tarjoajilla ja niiden skaalautuvuus oli hyvin optimaalista ja riittoisaa, vaikka esimerkkisovellus oli siihen riittämätön käytännön kannalta. Palvelujen helppokäyttöisyydessä voiton otti AWS, sitten Azuren ja sitten Googlen. Googlen rajapintaa ohjeistukselle voisi kehittää, jotta asiakkaat voisivat paremmin palvelujen käyttöönottoon tarvittavia konfigurointia vähentää.
Hypoteettinen pk-yritysten NoSQL tietokantojen kuukausihinnaksi (miljoona tietokanta kirjausta/kk + 50 GB tallennustilaa/kk) muodostuivat halvimmasta kalleimpaan seuraavasti: Amazon AWS DynamoDB $ 9,375/kk, GCP Firebase $ 10,34/kk ja Azure CosmosDB $ 21,14/kk. Näiden laskemisessa virheiden mahdollisuus on suuri ja hinnat eivät ole vertailukelpoisesti esillä, vaan ne on pitänyt muuntaa laskennallisesti vertailukelpoisiksi.

Parhaimmaksi ratkaisuksi olisi koitunut MERN-pinolle Amazonin AWS Lambda ja EC2 palvelut, joilla olisi saanut edullisimman toimivan ratkaisun, koska näiden konfigurointi oli yksinkertaisinta ja sulavaa. ASW Lambda oli jopa halvempi kuin nykyinen konttipohjainen ratkaisu AWSässä, sillä se (Lambda) maksaa noin 0,32 dollaria per kahden tunnin hostaus vähäisillä kyselymäärillä. Azuren ja Google Cloudin osalta ei saatu toimimaan serveripuolen ohjelmalogiikkaa konfiguroinnin monimutkaisuuden vuoksi. Mutta Azuren App Service ja Azure Functions palveluilla olisi voinut päästä samoihin kuluihin nykyisen ratkaisun kanssa. Myös Google Cloudin App Enginen ja Cloud Functions palveluissa olisi voinut pyöriä samoissa kuluissa, ellei jopa halvemmalla, koska App Engine oli halvin kilpailijoiden laskentaa vaativissa tilanteissa. Kaikki palvelujen tarjoajat tarjosivat konteille myös Kubernetes palveluja, mutta ne olivat jätetty huomioimatta tässä projektissa, koska niiden kulut olivat perus konttipalveluja (Container Registry) kalliimpia ja eikä niihin ollut perehdytty tarpeeksi.

Oli palvelujen tarjoaja mikä tahansa näistä niin on mahdollista tehdä säästöä kuluissa valitsemalla palvelut oikein ja konfiguroida palvelut mahdollisimman pieniksi käyttöön tarvittavien resurssien pohjalta. Mitä vähemmän allokoi muistia (GiB), laskentaa (vCPU) ja tallennustilaa (GB) voi säästää sillä kuluissa huomattavia summia. Konfigurointi tuleekin tehdä huolellisesti ja turhia lisämoduuleja ei

kannata turhaan pyörittää, koska siitä aiheutuu lisäkuluja. Hyvänä muistisääntönä voisi pitää fraasia ”käytä vain tarpeeksi pientä palvelua”, joka auttaa räätälöidä sopivan kokoisia ratkaisuja. Myös kaikki pilvipalvelujen tarjoajat tarjoavat ilmaisia palveluja, joita kannattaa hyödyntää ja nekin alentavat kustannuksia. Niitä on listattu teoriaa kappaleessa sekä kuva 18. kohdassa ”Ilmaiset: Free tier”. Kaiken kaikkiaan palvelut ovat suhteellisen edullisia ottaen huomioon, että useissa tapauksissa ei tarvitse omaa palvelinta.

7. Pohdinnat ja johtopäätös

Kaiken kaikkiaan kuluja koitui serverless-palveluista odotetusti siten, että ne vastasivat odotusta. Projektissa käytännön osuudelta olisi teoriassa tullut halvemmaksi käyttää omaa erillistä palvelinta AWSn osalta. Luultavasti myös muiden palveluntarjoajien osalta. Koska web-sovellus oli sen verran pieni ja pyyntöjä koitui niin vähäisiä määriä, ettei se ollut kannattavaa palvelun kannalta katsottuna. Vaikka yleensä kannattavuuteen vaikuttavat skaalautuminen ja elastisuus. Huomattavasti kalleinta pilvipalveluissa on laskentaa vaativat toimenpiteet ja virtualisoidut palvelut (IaaS), vaikka niissä käyttäjä on vastuussa jo isosta osasta hallintaa. Halvimmaksi ratkaisuksi tuli upottaa staattinen sivu FaaS palvelulle ja erikseen hallinnoida palvelinta muualla. Tietokanta ulkoistettiin MongoDB Atlakseen ja sen yhteensopivuus tuotti ongelmia muissa palveluissa paitsi AWS, johon se integroitui kätevästi ja vaivatta Route 53 käyttäen. Docker kuvakkeiden käyttö oli selväsi helpointa koko pinon tuotantoon viemisessä. Muutoin konfigurointi olisi ollut paljon monimutkaisempaa ja haastavampaa, vaikka ei nytkään ilman haasteita edetty. Kaikin puolin MENR-pino uppoutui näille alusteille melko hyvin ja palvelut saatiin toimintakuntoisiksi tässä projektissa.

Yleisesti kustannustehokkuutta ajatellen serverless-pilvipalvelut ovat edullinen muoto korvikkeena täysimittaiselle infralle. Kulut ovat paljon pienemmät kuin kiinteiden laitteiden ylläpidosta koituvat kustannukset. Tosin samalla vaivalla hallinnoida pilvipalvelujen devops puolen konfigurointisia asioita olisi päässyt jopa helpommalla kiinteiden palvelimien hallinnassa. Opittiin myös se, että konfigurointia säätelemällä voi säästää kuluissa merkittäviä summia. Pilvipalvelujen hinnat olivat melko tasaisesti jakaantuneet eri palveluntarjoajien välillä paitsi Amazonin AWS laskennan EC2 palvelu, joka oli tavallista hintavampi. Keskihajonta serverless-hinnoille oli 0,0081, joka merkitsi sitä, että ne olivat aika homogenisiä vertailussa. Myös tietokantoja ajatellen kuten voitiin jo aiemmin havaita, tuli tietokantapalvelujen käyttäminen edullisemmaksi vaihtoehdoksi kuin oman tietokantapalvelimen käyttö.

Näiden palvelujen hintojen määräytymisen selvittäminen on erityisen haastavaa ja tarjolla olevat tiedot eivät ole vertailukelpoisia tai vaihteluväliä on runsaasti ja se vaikeuttaa selvitystyötä. Hinnat tulisikin laskea todellisissa käyttöajanjaksoissa eikä vain millisekunteina tai sekunteina, joka hämmentää hintoja arvioidessa tai vertaillessa. Siksi ne laskettiin vastaamaan käyttöä tunneissa, joka helpotti vertailussa. Havaittiin myös se, että pilvipalvelujen tarjoajat eivät tarjoa täysin vertailukelpoisia tuotteita. Näiden vertailussa täytyy syvällisemmin pohtia niiden arkkitehtuuria käytännössä ja yrittää vaivanoloisesti päätellä palveluista koituvia kustannuksia. Tässä olisi paljon kehittämisen tarvetta. Parempi ratkaisu olisi luoda palvelu, joka jäsentelee hintakehitysdatan ja luo siitä hintojen vertailulle kelpoisen vertailutaulukon, jossa luvut ovat homogenisoitu vastaamaan kokonaisuutta. Laskentaa ajatellen millisekunnin tarkkuudella vertailu ei välttämättä ole niin tärkeää kuluttajan kannalta katsottuna. Reaalimaailmassa palveluja käytetään pidempiä aikoja, jolloin hintaseuranta olisi parempi toteuttaa sitä suosivaksi. Tutkimuksessa huomattiin myös, että Googlen ilmoittamissa hinnoissa olisi kehitettävää, jotta hintaseuranta olisi helpompaa. Koska Google laskutti erikseen muistin allokoinnista ja

laskentatehosta, mitä muut palvelun tarjoajat eivät tehneet. Hinnoille myös voisi kehittää selkeän eron ilmaisien ja maksullisten palvelujen välille, jotta budjetointi olisi sulavampaa. Tietokantojen hinnoittelu oli hyvin työlästä selvittää ja tehdä datasta vertailukelpoista palvelujen välillä. Tässä olisi myös kehittämisen varaa.

Projekt olisi onnistunut helpommin, jos palvelimen asiat olisi rajannut kokonaan pois ja keskittynyt vain serverless-palveluihin. Tosin tästä tuli nyt kattavampi kuvaus kuin vain serverless-palvelujen osalta.

Kaiken kaikkiaan tuli opittua paljon tässä projektissa ja voi hyvin odotuksin suositella näitä kyseisiä palveluja muillekin. Voisi sanoa, että ”tosiaankin vain käytöstä maksetaan ja näillä palveluilla tekee selviä säästöjä”.

Lähteet

Altexsoft 2018. FaaS PRICING COMAPARISON. Luettavissa: https://www.altexsoft.com/blog/cloud/comparing-serverless-architecture-providers-aws-azure-google-ibm-and-other-faas-vendors/. Luettu: 21.02.2021.

Amazon s.a. Amazon EC2 On-Demand Pricing. Luettavissa: https://aws.amazon.com/ec2/pricing/on-demand/. Luettu: 07.04.2021.

Amazon s.a. Amazon EC2 Auto Scaling. Luettavissa: https://aws.amazon.com/ec2/autoscaling/. Luettu: 08.04.2021.

Amazon s.a. Amazon Elastic Container Registry pricing. Luettavissa: https://aws.amazon.com/ecr/pricing/. Luettu: 07.04.2021.

Amazon s.a. Amazon Elastic Container Service pricing. Luettavissa: https://aws.amazon.com/ecs/pricing/. Luettu 07.04.2021.

Amazon s.a. Amazon Route 53 pricing. Luettavissa: https://aws.amazon.com/route53/pricing/. Luettu: 07.04.2021.

Amazon s.a. Amazon S3 pricing. Luettavissa: https://aws.amazon.com/s3/pricing/. Luettu: 07.04.2021.

Amazon s.a. AWS Lambda pricing. Luettavissa: https://aws.amazon.com/lambda/pricing/. Luettu: 07.04.2021.

Amazon s.a. Databases on AWS. Luettavissa: https://aws.amazon.com/products/databases/. Luettu: 16.04.2021.

Amazon s.a. Pricing for On-Demand Capacity. Luettavissa: https://aws.amazon.com/dynamodb/pricing/on-demand/. Luettu 16.04.2021.

Amazon 2021. AWS calculator. Luettavissa: https://calculator.s3.amazonaws.com/index.html. Luettu: 21.02.2021.

Azure 2021. Azure Calculator. Luettavissa: https://azure.microsoft.com/en-us/pricing/calculator/. Luettu: 21.02.2021.

Curtis, B s.a. Top 4 Examples of Function as a Service (FaaS). YourTechDiet blogi. Luettavissa: https://www.yourtechdiet.com/blogs/function-as-a-service-faas-examples/. Luettu 09.04.2021.

Docker s.a. What is a Container? Luettavissa: https://www.docker.com/resources/what-container. Luettu: 11.04.2021.

Eismann et al. 2020. Serverless applications: why, when and how? Luettavissa: https://arxiv.org/pdf/2009.08173.pdf. Luettu 11.04.2021.

Google s.a. App Engine pricing. Luettavissa: https://cloud.google.com/appengine/pricing. Luettu: 07.04.2021.

Google s.a. Cloud Functions pricing. Luettavissa: https://cloud.google.com/functions/pricing. Luettu: 07.04.2021.

Google s.a. Cloud Run Pricing. Luettavissa: https://cloud.google.com/run/pricing. Luettu: 07.04.2021.

Google s.a. Cloud Strorage Pricing. Luettavissa: https://cloud.google.com/storage/pricing. Luettu: 07.04.2021.

Google s.a. Container Registry Pricing. Luettavissa: https://cloud.google.com/container-registry/pricing. Luettu: 07.04.2021.

Google s.a. Google Cloud databases. Luettavissa: https://cloud.google.com/products/databases. Luettu: 16.04.2021.

Google s.a. Load balancing and scaling.
Luettavissa: https://cloud.google.com/compute/docs/load-balancing-and-autoscaling. Luettu: 08.04.2021.

Google s.a. Pricing for Firestore. Luettavissa: https://cloud.google.com/firestore/pricing. Luettu 16.04.2021.

Google Trends 2021. Serverless. Luettavissa: https://trends.google.com/trends/explore?date=all&q=Serverless, Luettu: 20.01.2021.

Google 2021. Google Cloud Pricing Calculator. Luettavissa: https://cloud.google.com/products/calculator/?hl=ru. Luettu: 21.02.2021.

Microsoft s.a. App Service pricing. Luettavissa: https://azure.microsoft.com/en-us/pricing/details/app-service/windows/. Luettu: 07.04.2021.

Microsoft s.a. Azure autoscale. Luettavissa: https://azure.microsoft.com/en-us/features/autoscale/. Luettu: 08.04.2021.

Microsoft s.a. Azure Cosmos DB pricing. Luettavissa: https://azure.microsoft.com/en-us/pricing/details/cosmos-db/. Luettu: 16.04.2021.

Microsoft s.a. Azure Functions pricing. Luettavissa: https://azure.microsoft.com/en-us/pricing/details/functions/. Luettu: 07.04.2021.

Microsoft s.a. Container Instances pricing. Luettavissa: https://azure.microsoft.com/en-us/pricing/details/container-instances/. Luettu: 07.04.2021.

Microsoft s.a. Container Registry pricing. Luettavissa: https://azure.microsoft.com/en-us/pricing/details/container-registry/. Luettu: 07.04.2021.

Microsoft s.a. Types of Databases on Azure. Luettavissa: https://azure.microsoft.com/en-us/product-categories/databases/. Luettu: 16.04.2021.

Microsoft 2020. Tutorial: Deploy a multi-container group using Docker Compose Luettavissa: https://docs.microsoft.com/en-us/azure/container-instances/tutorial-docker-compose. Luettu 29.03.2021.

Medium s.a. Pilvipalvelut. Luettavissa: https://medium.com/@vanshvarshney_/what-is-iaas-vs-saas-vs-paas-and-xaas-whats-the-difference-examples-ceadeee146e6. Luettu 20.1.2021

Medium 2018. Deploy ReactJS App with S3 Static HostingLuettavissa: https://medium.com/serverlessguru/deploy-reactjs-app-with-s3-static-hosting-f640cb49d7e6. Luettu: 25.03.2021.

Medium 2019. FaaS Comparison. Luettavissa: https://pablo-iorio.medium.com/serverless-architectures-iii-market-offerings-aws-azure-google-and-alibaba-67c49d1a691d. Luettu: 19.02.2021

Medium 2019. Intrest over time. Luettavissa: https://pablo-iorio.medium.com/serverless-architectures-iii-market-offerings-aws-azure-google-and-alibaba-67c49d1a691d. Luettu: 19.02.2021

Medium 2019. Learn the MERN stack by building an exercise tracker — MERN Tutorial. Luettavissa: https://medium.com/@beaucarnes/learn-the-mern-stack-by-building-an-exercise-tracker-mern-tutorial-59c13c1237a1. Luettu 24.02.2021.

Medium 2019. Serverless martket offerings. Luettavissa: https://pablo-iorio.medium.com/serverless-architectures-iii-market-offerings-aws-azure-google-and-alibaba-67c49d1a691d. Luettu: 19.02.2021

Medium 2020. Dockerizing a MERN stack web application. Luettavissa: https://medium.com/mozilla-club-bbsr/dockerizing-a-mern-stack-web-application-ebf78babf136 Luettu: 19.03.2021

MongoDB s.a. MERN stack. Luettavissa: https://www.mongodb.com/mern-stack. Luettu: 11.04.2021.

Sherweb s.a. Cost of server ownership: on-premise vs. IaaS. Luettavissa: https://www.sherweb.com/blog/cloud-server/total-cost-of-ownership-of-servers-iaas-vs-on-premise/. Luettu: 08.04.2021.

Solita 2017. Serverless – mitä se tarkoittaa ja miksi siitä pitäisi kiinnostua? Luettavissa: https://www.solita.fi/blogit/serverless-mita-se-tarkoittaa-ja-miksi-siita-pitaisi-kiinnostua/. Luettu: 21.02.2021.

Techbeacon s.a. The economics of serverless computing: A real-world test. Luettavissa: https://tech-beacon.com/enterprise-it/economics-serverless-computing-real-world-test. Luettu: 08.04.2021.

Wikipedia s.a. Cloud computing. Luettavissa: https://en.wikipedia.org/wiki/Disk_image. Luettu: 11.04.2021.

Wikipedia s.a. Disk image. Luettavissa: https://en.wikipedia.org/wiki/Cloud_computing. Luettu: 11.04.2021.

Wikipedia s.a. Infrastructure as a service. Luettavissa: https://en.wikipedia.org/wiki/Infrastructure_as_a_service. Luettu: 11.04.2021.

Wikipedia s.a. Middleware. Luettavissa: https://en.wikipedia.org/wiki/Middleware. Luettu: 11.04.2021.

Wikipedia s.a. Mobile backend as a service. Luettavissa: https://en.wikipedia.org/wiki/Mobile_backend_as_a_service. Luettu: 11.04.2021.

Wikipedia s.a. Platform as a service. Luettavissa: https://en.wikipedia.org/wiki/Platform_as_a_service. Luettu: 11.04.2021.

Wikipedia s.a. Pilvipalvelujen hinnat. Luettavissa: https://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud. Luettu: 20.01.2021.

Wikipedia s.a. Representational state transfer. Luettavissa: https://en.wikipedia.org/wiki/Representational_state_transfer. Luettu: 11.04.2021.

Wikipedia s.a. Serverless computing. Luettavissa: https://en.wikipedia.org/wiki/Serverless_computing. Luettu: 11.04.2021.

Wikipedia s.a. Software as a service. Luettavissa: https://en.wikipedia.org/wiki/Software_as_a_service. Luettu: 11.04.2021.

Wikipedia s.a. Yleiset pilvipalvelut. Luettavissa: https://en.wikipedia.org/wiki/Cloud_computing#/media/File:Cloud_computing_layers.png. Luettu 20.1.2021.

Zdnet 2019. Serverless computing vs platform-as-a-service: Which is right for your business? Luettavissa: https://www.zdnet.com/article/serverless-computing-vs-platform-as-a-service-which-is-right-for-your-business/. Luettu: 09.04.2021.

Liitteet

Liite 1. Docker compose .yml tiedosto

```
version: '3.7'
services:
  server:
    build:
      context: ./server
      dockerfile: Dockerfile
    image: myapp-server
    container_name: myapp-node-server
    command: /usr/src/app/node_modules/.bin/nodemon server.js
    volumes:
          - ./server/:/usr/src/app
          - /usr/src/app/node_modules
    ports:
      - "5000:5000"
    depends_on:
      - mongo
    env_file: ./server/.env
    environment:
      - NODE_ENV=development
    networks:
      - app-network
  mongo:
    image: mongo
```

```
    volumes:
      - data-volume:/data/db
    ports:
      - "27017:27017"
    networks:
      - app-network
  client:
    build:
      context: ./client
      dockerfile: Dockerfile
    image: myapp-client
    container_name: myapp-react-client
    command: npm start
    volumes:
      - ./client/:/usr/app
      - /usr/app/node_modules
    depends_on:
      - server
    ports:
      - "3000:3000"
    networks:
      - app-network
networks:
    app-network:
        driver: bridge
volumes:  data-volume:
    node_modules:
    web-root:
      driver: local
```