Opinnäytetyö (AMK)

Elektroniikan koulutusohjelma

Tietoliikennejärjestelmät

2017

Jesse Juuti

WWW-SOVELLUKSEN KEHITYS AMAZON AWS SERVERLESS – ARKKITEHTUURILLA

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Elektroniikan koulutusohjelma | Tietoliikennejärjestelmät

2017 | 34

Jesse Juuti

WWW-SOVELLUKSEN KEHITYS AMAZON AWS SERVERLESS -ARKKITEHTUURILLA

Tämän opinnäytetyön tavoitteena oli luoda järjestelmä, joka suodattaa sovelluskaatumisista saatavien lokitiedostojen oleelliset osat ja kirjaa tiedot tietokantaan. Tämän lisäksi tavoitteena oli luoda www-käyttöliittymä, jonka avulla tietokantaan on mahdollista tehdä hakuja ja etsiä tietoja. Työssä hyödynnettiin Amazon Web Services -pilvipalveluita järjestelmän eri osa-alueiden toteuttamiseen. Työ on Nuviz Oy:n toimeksiantama, ja sen tarkoitus on auttaa sovelluskehitystyön virheiden korjausta.

Amazon S3 -palveluun ladatut sovelluskaatumisista johtuvat lokitiedostot suodatetaan AWS Lambda -funktion avulla, joka hoitaa myös oleellisten tietojen kirjaamisen Amazon DynamoDB -tietokantaan. Tietojen kirjaamisen jälkeen sovelluskaatumisten tiedot ovat luettavissa www-käyttöliittymän avulla yrityksen verkossa.

Lopputuloksena saatiin aikaiseksi selkeä ja yksinkertainen järjestelmä helpolla käyttöliittymällä, jota on mahdollista edelleen kehittää tarpeiden mukaan.

ASIASANAT:

Amazon AWS, Serverless, Lambda, DynamoDB, S3, pilvipalvelut, pilvilaskenta, virtualisointi

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Electronics degree programme | Telecommunication Systems

2017 | 34

Jesse Juuti

DEVELOPING A WWW APPLICATION USING AMAZON AWS SERVERLESS ARCHITECTURE

The objective of this thesis was to create a system which parses relevant parts from application crash logs and writes this information into a database. In addition, another objective was to create a web interface for querying and searching data from the database. Amazon Web Services cloud services were used to achieve the objectives. This thesis was commissioned by Nuviz Ltd to help application developers to search and fix flaws in code.

Crash logs uploaded to Amazon S3 service were parsed with AWS Lambda function which also handles writing information into the Amazon DynamoDB database. After writing the information into database, the information about application crashes was easily accessible via the web interface in the corporate network.

As a result, the commissioner of this thesis obtained a clear and simple system with a user-friendly user interface with the possibility to further develop it if and when needed.

KEYWORDS:

Amazon AWS, Serverless, Lambda, DynamoDB, S3, cloud services, cloud computing, virtualization

SISÄLTÖ

KUVAT

LYHENTEET JA SANASTO

API	Application Programming Interface. Ohjelmointirajapinta, joka tarjoaa käyttäjälle pääsyn tiettyihin toiminnallisuuksiin.
AWS	Amazon Web Services. Amazonin tarjoama pilvipalvelukokonaisuus.
Backend	Ohjelmiston taustalla toimivat osat.
Bootstrap	Twitterin työntekijöiden kehittämä www-ohjelmistokehys HTML, CSS ja Javascript ohjelmointikielille.
CSS	Cascading Style Sheets. Tyylisivukieli, jota käytetään muun muassa HTML-verkkosivujen esittämiseen.
DynamoDB	Amazon DynamoDB. NoSQL-tietokantapilvipalvelu.
Frontend	Ohjelmiston osat, jotka ovat käyttäjärajapinassa.
HTML	Hypertext Markup Language. Kieli, jota käytetään verkkosivujen luomisessa.
JavaScript	Netscape Communications Corporationin kehittämä ohjelmointikieli, jota käytetään dynaamisten verkkosivujen luomisessa.
Lambda	AWS Lambda. Pilvipalvelu yksittäisten funktioiden suorittamiseen.
Node.js	Javascript ajoympäristö, jolla suoritetaan Javascript-koodia palvelimen puolella.
PHP	Hypertext Preprocessor. Palvelinpuolen ohjelmointikieli, jota käytetään dynaamisten verkkosivujen luomisessa.
SDK	Software development kit. Ohjelmistokehitystyökalu.
Serverless	Pilvipalveluiden arkkitehtuurilaji, jossa suoritetaan funktiota.
S3	Amazon Simple Storage Service (Amazon S3), pilvitallennus-palvelu.
Virtuaalikone	Ohjelmallisesti toteutettu tietokone.
Virtualisointi	Fyysisen palvelimen tai tietokoneen lohkominen useampaan toisistaan erilliseen virtuaaliseen palvelimeen tai tietokoneeseen eli virtuaalikoneeseen.

1 JOHDANTO

Pilvipalveluiden käyttö on kasvanut huomattavasti viime vuosien aikana. Tuoteperheet kasvavat jatkuvasti uusilla tuotteilla monissa eri käyttötarkoituksissa. Amazon on yksi keskeisimmistä pilvipalveluidentarjoajista Amazon Web Services -pilvipalvelullaan ja Nuviz Oy on valinnut sen palvelut tukemaan omaa sovelluskehitystään. NUVIZ kehittää moottoripyöräilijöille HUD-laitetta, jonka avulla motoristi saa helposti käyttöönsä muun muassa navigointipalvelut, musiikin kuuntelun sekä puhelut. Laitteen toiminta perustuu Android-käyttöjärjestelmään sekä mobiilisovellukseen, jonka avulla laitetta voi konfiguroida. Pilvipalveluiden keskeisin houkutin on niiden skaalautuvuus pienistä käyttäjämääristä suuriin massoihin ympäri maailman. Lisäksi hinnoittelu on hyvinkin kohtuullinen, sillä palveluiden hinta määräytyy niiden käytön mukaan. Käyttöperusteinen hinnoittelu mahdollistaa sen, että palvelun ostajan ei tarvitse maksaa siltä ajalta kun palvelulla ei ole käyttöä. Tämän lisäksi säästytään suurilta palvelininvestoinneilta.

Amazon AWS -palveluiden hyödyntämistä sovelluskehityksessä on aiemmin tutkinut Tero Toomas vuonna 2016 julkaistussa opinnäytetyössään, jossa aihetta on tutkittu olemassa olevaan sovellukseen integroimisen näkökulmasta [1]. Lisäksi aiheeseen liittyy mikropalvelut ja kontit, joiden hyötyjä ja parhaita käytäntöjä on tutkinut Miina Koskinen vuonna 2016 julkaistussa opinnäytetyössään [2].

NUVIZilla on tarve seurata kehitteillä olevan Android-mobiilisovelluksen sekä Androidiin perustuvan HUD-laitteen kaatumisia, jotta tällaisten kaatumisten aiheuttaneet virheet koodissa saadaan korjattua mahdollisesti jo ennen kuin lopullinen sovellus on loppukäyttäjällä. Sovelluksen kaatuessa virhetilanteesta tallentuu lokitiedosto, joka lähetetään internet yhteyden välityksellä Amazonin S3 -tallennusalustaan. Lokitiedostojen seurannan helpottamiseksi ja tulkitsemiseksi tarvittiin kuitenkin selkeää ja helppokäyttöistä työkalua.

Opinnäytetyönä luotiin järjestelmä, joka suodattaa hyödylliset tiedot lokitiedostoista ja tallentaa ne tietokantaan. Lisäksi kehitettiin tietokannan lukua varten selkeä ja helppokäyttöinen käyttöliittymä. Järjestelmän sekä käyttöliittymän kehittämiseen on käytössä yrityksen Amazon Web Services -käyttäjätilin palvelut sekä yrityksen omat paikalliset palvelimet ja tietokoneet.

Aluksi työssä käsitellään pilvipalveluita, niiden ominaisuuksia sekä historiaa ja kehittymistä. Kolmannessa luvussa käsitellään tässä työssä käytettyjen Amazon AWS:n palveluita sekä niiden ominaisuuksia ja neljännessä luvussa itse järjestelmän toteuttamista.

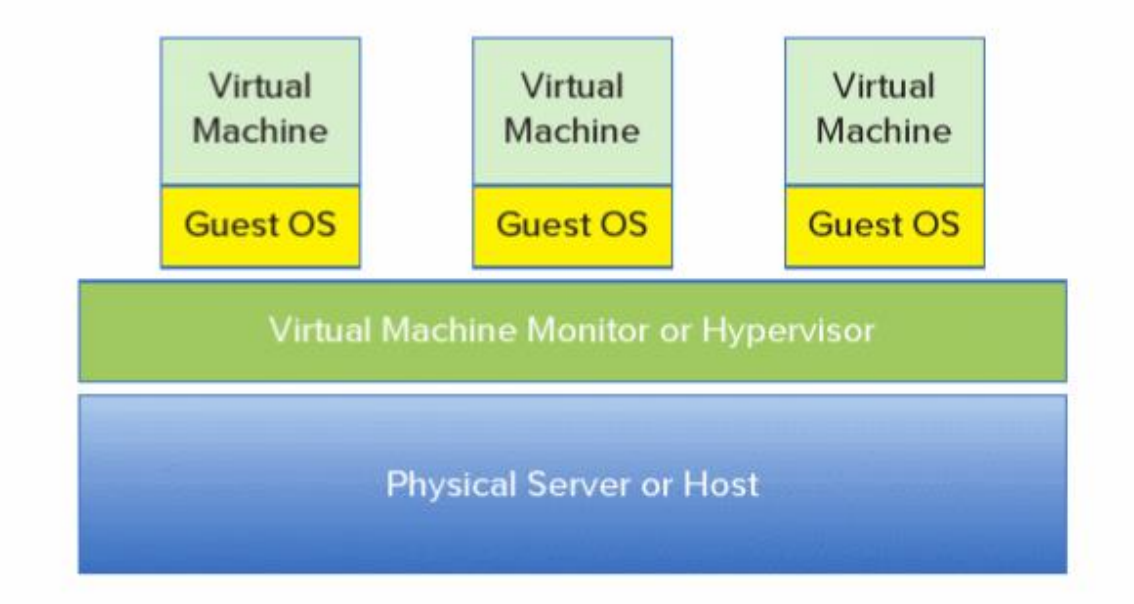
2 PILVIPALVELUT

Pilvipalvelut mielletään usein vain pilvitallennusalustaksi, kuten Google Drive tai Microsoft OneDrive. Todellisuudessa pilvipalvelut käsittävät yhä laajenevan joukon erilaisia palveluita, jonka yksi osa on pilvitallennus. Nykypäivänä monet käyttävät tietämättään pilvipalveluita osana esimerkiksi älypuhelimen käyttöä tai suoratoistopalveluita [3]. Tästä yksi esimerkki on monille tuttu suoratoistopalveluja tarjoava Netflix [4].

Pilvipalvelu käsitteenä on kuitenkin hieman epäselvä, sillä yleisesti hyväksyttyä määritelmää ei ole. Pilvipalveluilla tarkoitetaan kuitenkin tietotekniikkaresurssien tarjoamista internet yhteyden välityksellä käyttäjälle. Pilvipalveluissa ei ole oleellista se, missä tarjottavat resurssit sijaitsevat. Käyttäjän ei myöskään tarvitse huolehtia resurssien ylläpidosta. Pilvipalveluille on tyypillistä myös niiden itsepalvelullisuus eli käyttäjä voi itse tarvitessaan aktivoida palveluita tai poistaa niitä käytöstä ilman palveluntarjoajaan yhteyden ottamista. [5]

2.1 Historia ja kehittyminen

Pilvipalvelut mahdollistava tekijä on virtualisointi, joka taas mahdollistaa useamman virtuaalikoneen ajamisen yhdessä fyysisessä tietokoneessa. Virtualisoinnilla tarkoitetaan mallia, jossa fyysisessä koneessa ajetaan yhtä tai useampaa ohjelmallisesti toteutettua tietokonetta eli virtuaalikonetta, joka matkii fyysisen koneen ominaisuuksia. Kuvassa 1 on mallinnettu isäntäpalvelinta, jonka sisällä toimii kolme virtuaalikonetta ja niiden käyttöjärjestelmät. Hypervisor on ohjelma, joka toteuttaa virtuaalikoneiden pyytämiä palveluita. [3]



Kuva 1. Mallinnuskuva kolmesta virtuaalikoneesta yhdessä fyysisessä palvelimessa [3].

Ennen virtualisointia vallitsi malli, jossa yhdessä palvelimessa toimi ainoastaan yksi sovellus. Tämä aiheutti sen, että datakeskukset alkoivat olla ongelmissa kasvavien datamäärien takia, jolloin tarvittiin yhä isompi määrä fyysisiä palvelimia, josta seurasi väistämättä tilanpuutetta. Toinen merkittävä ongelma oli se, että palvelinrikon tai luonnonkatastrofin sattuessa palvelimet saattoivat olla pitkään pois käytöstä. Lisäksi käyttöaste jäi verrattain pieneksi, koska yksittäiset palvelimet olivat allokoituina yksittäisiin sovelluksiin ja palvelimien tehot kasvoivat työmäärän pysyessä lähes samana. Tämä tarkoittaa sitä, että palvelimet saattoivat olla joutilaana, mutta silti varattuina käyttöä varten. [3]

Virtualisointi tarjosi moniin ongelmiin ratkaisut. Se mahdollisti monen yksittäisen fyysisen palvelimen muuttamisen virtuaalikoneiksi, joka nosti yksittäisen fyysisen palvelimen käyttöastetta. Tämä luonnollisesti myös vähensi tarvittavien fyysisten palvelimien määrää, josta seurasi myös palvelinkeskuksen kulujen ja hiilijalanjäljen pienentymistä. Kaiken lisäksi virtuaalikone on mahdollista siirtää toiseen isäntäkoneeseen palvelinrikon tai muun katastrofin sattuessa, jolloin virtuaalikone on mahdollisimman paljon käytettävissä. [3]

Virtualisointi on siis pilvipalveluiden edellytys, joka muutti palvelinkeskusten toimintaa merkittävästi. Palvelinkeskukset muuttuivat automatisoiduiksi ja skaalautuviksi resurssipankeiksi. Muutos antoi tilaa palveluiden kehittämiselle, kun aikaa ei enää

kulunut niin paljon palveluiden ylläpitämiseen. Tämän muutoksen myötä aikanaan syntyi käsite virtuaalisesta palvelinkeskuksesta, joka oli pilvessä ja oli kaikkea, mitä fyysiset palvelinkeskuksetkin olivat olleet. [3]

Tässä kohtaa on syytä kuitenkin todeta, että pilvipalvelut eivät ole yhtä kuin virtualisointi vaikkakin sillä on tärkeä rooli pilvipalveluiden mahdollistavana tekijänä. Virtualisoinnin toteuttaminen palvelinkeskuksessa ei siis tee palvelinkeskuksesta vielä pilvipalveluntarjoajaa. [5]

Nykypäivän pilvipalveluihin liittyy vahvasti myös kontit. Kontit ovat ohjelmistollisesti toteutettuja ohjelmien suoritus ympäristöjä ilman käyttöjärjestelmää. Vaikka kontit eivät sisälläkään käyttöjärjestelmää ne silti vaativat sellaisen toimiakseen. [6] Kontit ovat osa nykypäivän Serverless-arkkitehtuuria eli mallia, jossa palvelimet ja virtuaaliset palvelimet ovat yhä enemmän näkymättömissä käyttäjälle. Serverless-arkkitehtuurissa suoritetaan funktioita eristetyissä ympäristöissä eli konteissa kokonaisten sovelluksien sijasta. [7]

2.2 Pilvipalvelumallit

Pilvipalvelut jaetaan tavallisesti kolmen perustyypin mukaan: Platform as a Service, Infrastructure as a Service ja Software as a Service eli lyhennettynä PaaS, IaaS ja SaaS. PaaS-tyypin pilvipalvelussa asiakkaalle lohkotaan palveluita virtuaalisesta palvelinympäristöstä, kun taas IaaS-tyypin pilvipalvelussa palveluntarjoajalla on yksi tai useampi virtuaalinen palvelinsali, josta lohkotaan asiakkaalle osioita. SaaS-tyypin pilvipalvelussa asiakas saa käyttöönsä pelkän sovelluksen internet yhteyden välityksellä. [8]

Perustyyppien lisäksi puhutaan myös FaaS eli Function as a Service -palvelumallista, jolla viitataan Serverless-arkkitehtuuriin, jossa suoritetaan sovelluksien sijasta funktioita. [7] Näiden neljän lisäksi on olemassa myös käsitteet Everything as a Service eli EaaS ja X as a Service eli XaaS, joilla viitataan yleisesti kaikkeen palvelullistamiseen. [5]

2.3 Hyödyt ja riskit

Pilvipalveluiden edut ja niiden tuomat mahdollisuudet vaihtelevat tietysti sen mukaan mitä eri palveluita valjastetaan käyttöön. Pilvipalvelut yleisesti ottaen tarjoavat kuitenkin ennen kaikkea vapautta resurssirajoitteista. Vapaus resurssirajoitteista johtuu sitä, että

palveluntarjoajan resurssit ovat niin suuret, että ne saattavat vaikuttaa käyttäjästä jopa rajattomilta. [5]

Lisäksi pilvipalvelut tarjoavat myös vapautta ylläpidollisista tehtävistä, sillä ne on ulkoistettu palveluntarjoajalle. Tästä syystä myös palvelut ovat aina ajantasalla, sillä palveluntarjoajalla on paine pitää tarjoamansa palvelut ajantasalla kilpaillakseen muiden palveluntarjoajien kanssa. [5]

Osa tai kaikki laskentatehollisesti vaativista tehtävistä voidaan suorittaa pilvessä, jolloin tehtävät suoritetaan nopeammin, koska käytössä on suurempi määrä laskentatehoa. Käyttöön perustuva laskutusmalli mahdollistaa sen, että resursseja on tarvittaessa käytössä lähes rajattomasti ja kun käyttöä resursseille ei ole, ei tarvitse myöskään maksaa. [5]

Haittapuolena yksi selkeimmistä on omassa hallinnassa olevien resurssien siirtyminen palveluntarjoajan hallintaan. Tämä osaltaan kasvattaa riskejä, koska resurssien täysi hallinta menetetään. Pilvipalveluihin liittyviä riskejä ovat muun muassa: tietojen päätyminen ulkopuolisille, ulkopuolisen pääsy palveluihin, pilvipalvelun toimimattomuus tai pilvessä olevien tietojen katoaminen lopullisesti. [5]

Pilvipalveluihin liittyy myös riski lukittautumisesta, joka tarkoittaa sitä, että palveluntarjoajan valittuaan ja sen palvelut käyttöönotettuaan palveluiden käytön siirtäminen toisen palveluntarjoajan vastaaviin palveluihin voi olla haastavaa tai jopa osin mahdotonta. Tästä syystä palveluntarjoajan vaihtamiseen liittyvät kustannukset nousevat niin suuriksi ettei sitä kannata lopulta tehdä. Lukittautumisen riskiä voi myös olla hyvin vaikea ennustaa etukäteen. [5]

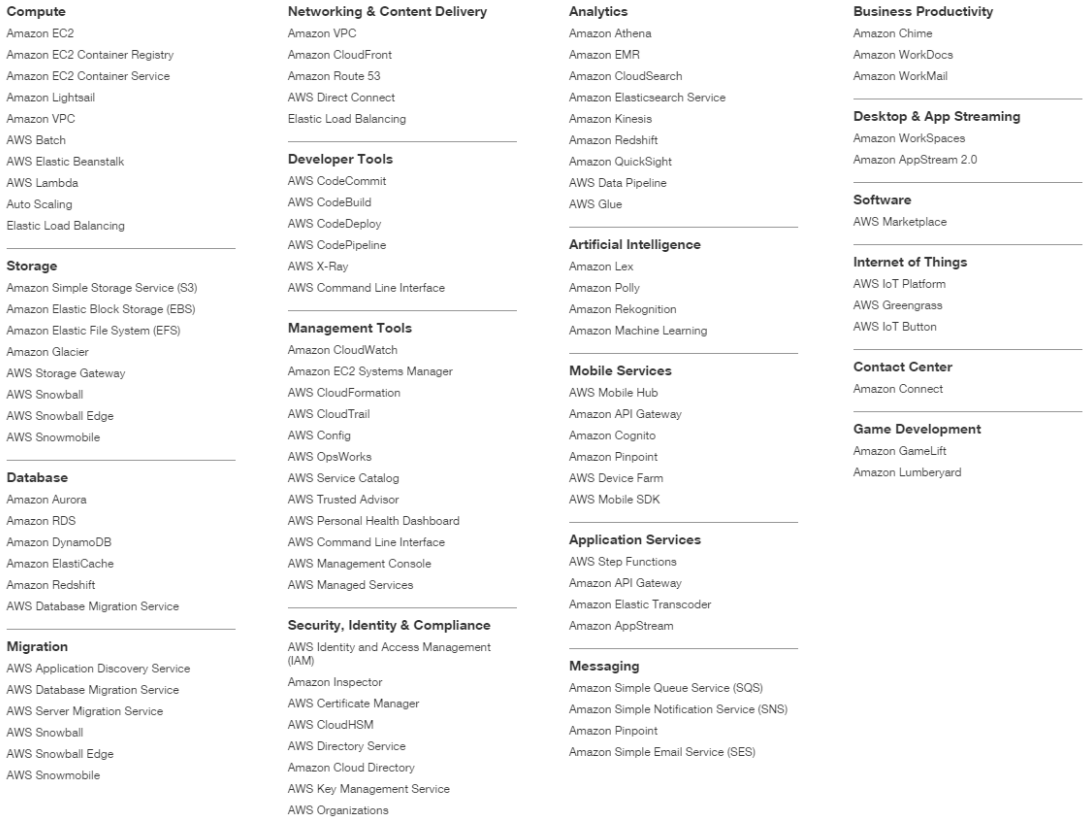
2.4 Palveluntarjoajat

Pilvipalveluntarjoajia on alalla nykyään monia, mutta tunnetuimmat niistä ovat Amazon, Microsoft ja Google. Kaikki kolme ovat toistensa suoria kilpailijoita alalla ja siten tarjoavatkin hyvin samankaltaisia palveluita. Tämän kappaleen tarkoitus on lyhyesti esitellä edellä mainitut toimijat alalla.

2.4.1 Amazon

Amazon on aloittanut toimintansa verkossa toimivana kirjakauppana vuonna 1994. Amazon on sittemmin laajentanut toimintaansa kaikeinlaisten tavaroiden verkkokauppaamiseen. Amazon ei siis ole ohjelmistotalo toisin kuin esimerkiksi Google tai Microsoft, jotka myös kilpailevat nykyään pilvipalveluntarjoajina. Amazonin vahvuus on kuitenkin siinä, että sillä on vankka kokemus internetissä tapahtuvasta kaupankäynnistä. [8]

Amazon Web Services on käynnistynyt jo vuonna 2002, jolloin se aloitti tarjoamalla sovelluksia palveluna jakamalla internetissä tapahtuvasta kaupankäynnistä saatua dataa. Suunta kohti varsinaisia pilvipalveluita otettiin kuitenkin vasta vuonna 2006, kun Elastic Compute Cloud eli EC2 virtuaalipalvelinpalvelu aloitettiin. [8] Kuvasta 2 voidaan nähdä suurin osa AWS:n tarjoamista palveluista.



Kuva 2. Osa Amazon Web Services palvelutarjonnasta [9].

Kuvasta 2 nähdään myös kuinka paljon nykyään on saatavilla erilaisia pilvipalveluita hyvin monesta eri kategoriasta. AWS tarjoaa ilmaista kokeilua, jotta asiakkaan on helppo

tutkia palveluiden soveltumista omaan käyttöön. Kuvassa 3 on listattuna eri alueita, joissa palvelinkeskukset sijaitsevat.

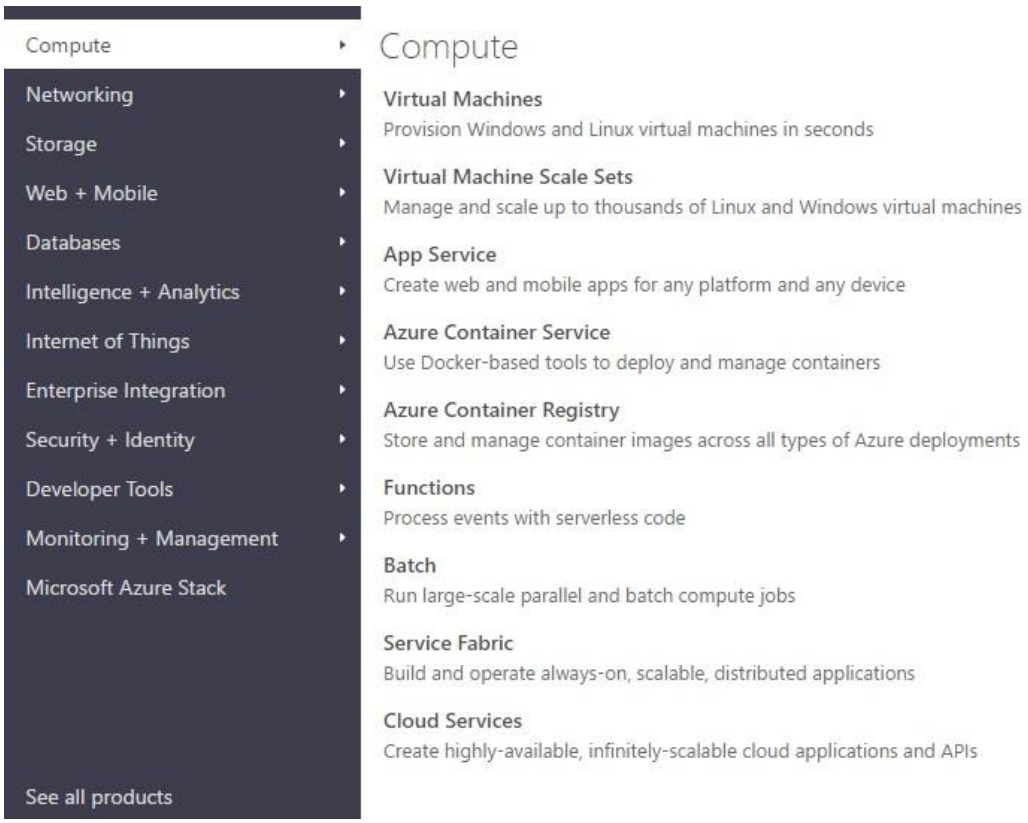
Kuva 3. AWS:llä palveluille tarjolla olevat palvelinkeskusalueet [10].

AWS:llä on palvelinkeskusalueita ympäri maailman, joista käyttäjän tulee valita vähintään yksi mihin hän haluaa palvelunsa sijoittaa. Tällä tavoin voidaan vaikuttaa muun muassa palveluiden käyttämiseen liittyviin viiveisiin. Yhdellä palvelinkeskusalueella on käytössään useita palvelinkeskuksia, joissa palvelut voivat sijaita. Kuvassa 3 sulkujen sisällä ilmoitetut luvut ovat alueella olevien palvelinkeskuksien määriä. [10] Palveluiden tarjonta vaihtelee hieman eri alueittain [11]. AWS:llä on työn kirjoittamishetkellä Skyhigh Networks:n mukaan 37,1 % markkinaosuus, joka on suurin markkinaosuus alalla [12]. Amazon Web Services -pilvipalveluista kerrotaan lisää luvussa kolme.

2.4.2 Microsoft

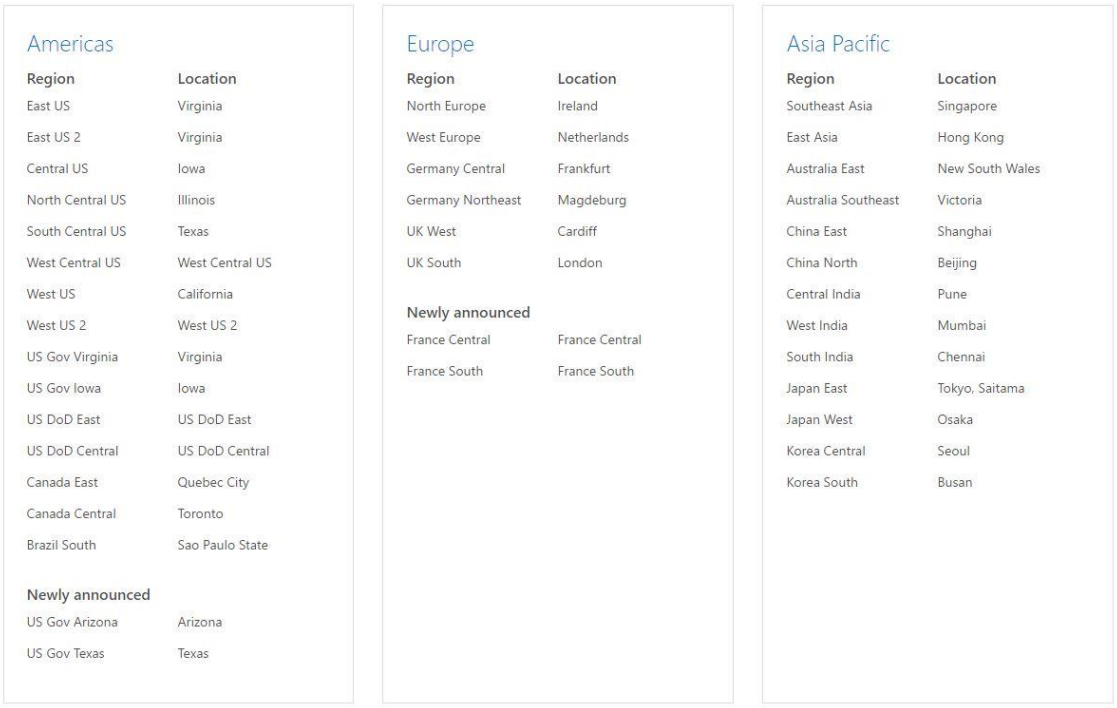
Microsoftin historia ulottuu 1970-luvulle, jolloin henkilökohtaiset tietokoneet alkoivat yleistyä. Microsoft on kehittänyt Windows käyttöjärjestelmän sekä Office toimisto-ohjelmistopaketin, joilla se on saavuttanut itselleen vankan aseman markkinoilla. [8]

Microsoft Azure on julkistettu loppuvuodesta 2008, mutta vasta myöhemmin 2010 vuoden alkupuolella otettu laajemmin käyttöön. [8] Kuvassa 4 nähdään osa Azuren palvelutarjonnasta.



Kuva 4. Osa Microsoft Azuren palvelutarjonnasta [13].

Azure tarjoaa suoria vastineita AWS:n palveluille ja myös Azure tarjoaa ilmaista kokeilujaksoa palveluihin tutustumista varten. Esimerkiksi suora vastine AWS Lambda -palvelulle on Azure Functions. Azurella on työn kirjoittamishetkellä Skyhigh Networks:n mukaan 28,4 % markkinaosuus ja jää siten toiseksi verrattuna AWS:ään [12]. Kuvasta 5 nähdään Azurella palveluille tarjotut alueet.



Americas

Region	Location
East US	Virginia
East US 2	Virginia
Central US	Iowa
North Central US	Illinois
South Central US	Texas
West Central US	West Central US
West US	California
West US 2	West US 2
US Gov Virginia	Virginia
US Gov Iowa	Iowa
US DoD East	US DoD East
US DoD Central	US DoD Central
Canada East	Quebec City
Canada Central	Toronto
Brazil South	Sao Paulo State

Newly announced

US Gov Arizona	Arizona
US Gov Texas	Texas

Europe

Region	Location
North Europe	Ireland
West Europe	Netherlands
Germany Central	Frankfurt
Germany Northeast	Magdeburg
UK West	Cardiff
UK South	London

Newly announced

France Central	France Central
France South	France South

Asia Pacific

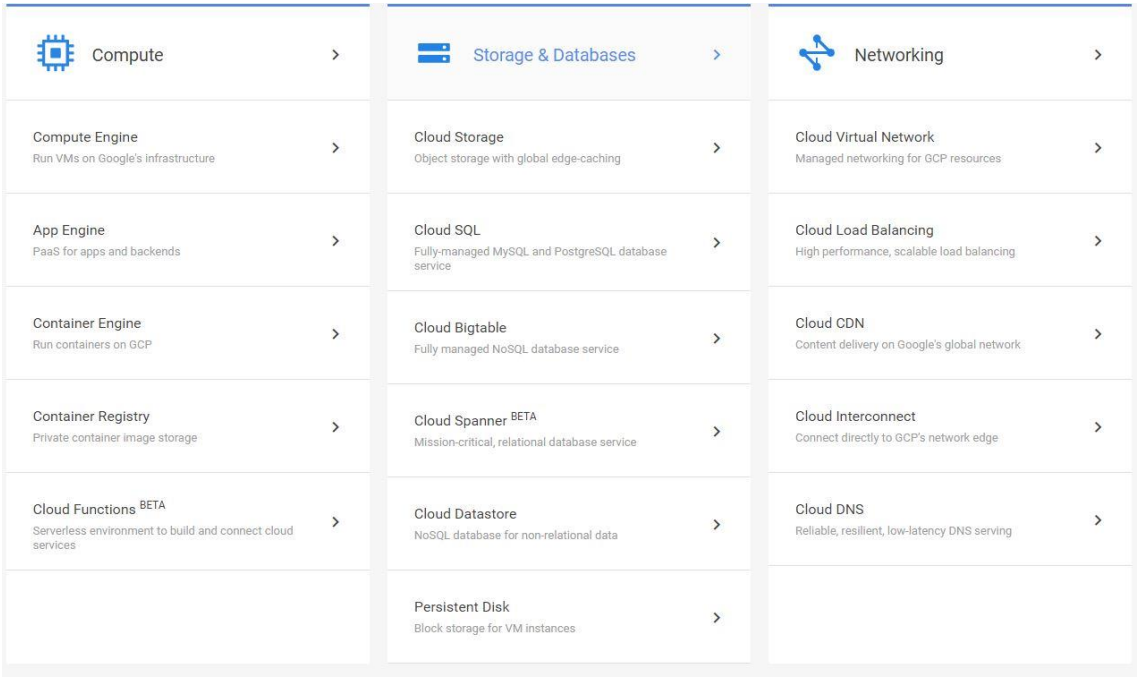
Region	Location
Southeast Asia	Singapore
East Asia	Hong Kong
Australia East	New South Wales
Australia Southeast	Victoria
China East	Shanghai
China North	Beijing
Central India	Pune
West India	Mumbai
South India	Chennai
Japan East	Tokyo, Saitama
Japan West	Osaka
Korea Central	Seoul
Korea South	Busan

Kuva 5. Azurella palveluille tarjolla olevat palvelinkeskusalueet [14].

Tarjolla olevia alueita on työn kirjoittamishetkellä huomattavasti enemmän kuin AWS:llä, mutta AWS:llä on käytössään 42 palvelinkeskusta jakautuen eri alueille. Microsoft Azure ei ole sen enempää eritellyt palvelinkeskuksien määrää.

2.4.3 Google

Google on tunnettu ennen kaikkea hakukonepalvelustaan, mutta tämän lisäksi myös Android-käyttöjärjestelmästään, YouTube videopalvelustaan sekä itse ajavien autojen pilotoinneista USA:ssa. [8] Tämän kaiken lisäksi Google tarjoaa myös pilvipalveluitaan Google Cloud Platform -palvelun kautta, joka on aloittanut toimintansa vuoden 2011 lopulla. Google on kuitenkin tarjonnut Google App Engine PaaS-mallin pilvipalveluaan jo vuodesta 2008. [15] Kuvassa 6 on osa Google Cloud Platformin palvelutarjonnasta.



Kuva 6. Osa Google Cloud Platformin palvelutarjonnasta [16].

Kuvasta 6 nähdään, että palvelutarjonta noudattaa samaa kaavaa kilpalijoidensa kanssa kuten aiemminkin on todettu. Kilpailijoidensa tavoin myös Google Cloud Platform tarjoaa mahdollisuutta kokeilla palveluitaan ilmaiseksi. Google Cloud Platformilla on työn kirjoittamishetkellä Skyhigh Networks:n mukaan 16,5 % markkinaosuus ja on siten kolmannella sijalla markkinaosuuksista [12].

Kuva 7. Google Cloud Platformilla palveluille tarjolla olevat palvelinkeskusalueet [17].

Kuvasta 7 nähdään, että Google Cloud Platform ilmoittaa tarjolla olevat alueet sekä palvelinkeskuksien määrät samaan tapaan kuin AWS. Kuvasta 6 nähdään myös, että Google Cloud Platformin Cloud Functions -palvelu on vasta beta-kehitysvaiheessa eli se on kilpailijoitaan jäljessä Serverless-arkkitehtuuria toteuttavan palvelun osalta.

3 AMAZON WEB SERVICES

Amazon on aloittanut virallisesti pilvipalveluiden tarjoamisen vuonna 2006 ja oli siten ensimmäinen suuri toimija alalla Amazon Web Services palvelullaan. [5] Sen keskiössä on samana vuonna aloitettu Elastic Compute Cloud (EC2), joka on virtuaalisten palvelimien pilvipalvelu. [8] Tämä työ keskittyy kuitenkin muiden AWS-palveluiden ympärille, joten tästä syystä EC2-palvelua ei käsitellä tässä työssä sen enempää.

3.1 Palvelut

Tässä osiossa on tarkoitus käydä läpi tämän työn kannalta tärkeimpiä palveluita ja hieman myös niiden hinnoittelua perustavanlaatuisesti. Työn kannalta tärkeimpiä palvelut ovat yksinkertaisesti siksi, että niitä on käytetty tämän työn toteutuksessa. Palveluita on tarjolla lukemattomia eikä kaikkien läpi käyminen palvele millään tavalla tämän työn toteutusta.

3.1.1 AWS Lambda

AWS Lambda on FaaS-palvelumallin pilvipalvelu eli se on niin kutsuttua Serverless-arkkitehtuuria toteuttava palvelu. AWS Lambda -palveluun siis lisätään haluttu funktio, jota sitten suoritetaan vasteena eri palveluiden tapahtumille. Palvelu hallinnoi automaattisesti tarvittavia laskentaresursseja, joten käyttäjän ei tarvitse murehtia niistä. AWS Lambdan funktioita voidaan hyödyntää yhdessä muiden AWS-palveluiden kanssa ja toteuttaa näiden välillä erilaisia kustomoituja toimintoja. Lisäksi AWS Lambda -funktioilla voidaan toteuttaa omia backend-palveluita, kuten esimerkiksi uuden käyttäjän luominen ja siihen liittyvät oheistapahtumat. [18]

AWS Lambda -funktion suoritus voidaan laukaista erilaisilla tapahtumilla, kuten S3-palveluun (Amazon Simple Storage Service) ladatulla tiedostolla tai muutoksella DynamoDB-tietokannassa. Lambda-funktion luomisen yhteydessä sille määritellään muun muassa funktion nimi, suorituksen laukaiseva tekijä, ajoympäristö (Java, Node.JS tai Python) sekä resurssivaatimukset. Lisäksi luomisen yhteydessä on mahdollista valita joitakin valmiita esimerkkikoodeja funktion pohjaksi. [18]

AWS Lambdan hinnoittelu perustuu funktioiden ajoaikaan sekä muistinkäyttöön. AWS Lambdan hinnoittelussa jokaisesta käytetystä gigatavusekunnista käyttäjä maksaa $0,00001667. Käyttäjällä on käytössään joka kuukausi 400 000 ilmaista gigatavusekunttia. Tämän lisäksi käyttäjä maksaa jokaisesta miljoonasta funktion suorittamisesta $0,20, kun käyttäjä on ylittänyt joka kuukauden ensimmäisen ilmaisen miljoonan suorituskerran määrän. Tällä tavoin Lambda funktiolle, jolle on allokoitu 512 megatavua muistia, ja jota on suoritettu kolme miljoonaa kertaa siten, että joka suoritus on kestänyt yhden sekuntin verran, hintaa koituu kokonaisuudessaan $18,74 per kuukausi. [19] 512 megatavua muistia riittänee tosin jo melko ison funktion suorittamiseen, ja tässä työssäkin toteutun funktion muistin tarve on murto-osa tuosta määrästä. Kolme miljoonaa suorituskertaa tarkoittaisi tämän työn kannalta ajateltuna kolmen miljoonan kaatumislokin käsittelyä ja kirjausta tietokantaan.

3.1.2 Amazon DynamoDB

Amazon DynamoDB on NoSQL-tietokantapilvipalvelu, jonka avainominaisuudet ovat skaalautuvuus, suorituskyky sekä joustavuus. Joustavuus tarkoittaa sitä, että DynamoDB-tietokantaan voidaan tallentaa tietoa dokumenttimuodossa, kuten esimerkiksi JSON-dokumenttina. Tällä tavoin koodissa ei tarvitse käsitellä dataa sen takia, että se sopisi tietokantaan. Tämän lisäksi DynamoDB-tietokantaan voidaan tallentaa tietoa avain-arvo mallilla eli jokaisella tietokannan rivillä on pääavain ja mahdollisesti lajitteluavain, joiden avulla jokainen tietokannan rivi voidaan tunnistaa. Pääavaimen tai pääavain ja laijitteluavain yhdistelmän on oltava uniikkeja, jotta rivit voidaan eritellä ja tunnistaa. Toisin sanoen samalla pääavaimen arvolla ei voi olla kahta riviä. [20]

Jokaisella rivillä voi myös olla eri määrä avaimia eli DynamoDB-tietokanta on siis skeematon. DynamoDB-tietokantoja kutsutaan tauluiksi. DynamoDB-taulun tiedot replikoituvat kolmen palvelinkeskuksen välillä käyttäjän valitsemalla alueella, jolla taataan palvelun luotettavuus ja datan säilyvyys. [20]

Amazon DynamoDB -palvelun hinnoittelu perustuu taulun kirjoitus- ja lukutarpeisiin, tallennetun datan määrään sekä valinnaiseen ominaisuuteen nimeltä DynamoDB Streams. Hinnat vaihtelevat hieman eri alueittain. Yksinkertaistettuna esimerkkinä jos käyttäjä tarvitsee taululleen miljoona kirjoitus- ja lukukertaa vuorokaudessa se tekee 11,6 kirjoitusta tai lukua vuorokauden sekuntia kohden jos yhden rivin koko on alle 1

kilotavu ja dataa kertyy yhteensä 1 gigatavun verran. Tällöin käyttäjä tarvitsee 12 kirjoitus- ja lukuyksikköä, joista ensimmäiset 25 kirjoitus- ja lukuyksikköä ovat ilmaiset. Us-east-1 alueella 12 kirjoitusyksikköä maksaa päivässä $0,1872 ja saman verran lukuyksiköitä maksaa $0,0374. Tallennuskulut ovat ensimmäisen ilmaisen 25 gigatavun jälkeen $0,25 gigatavulta. Valinnaisella DynamoDB Streams -ominaisuudella voidaan seurata taulussa tapahtuvia muutoksia ja maksu koostuu näiden tietojen lukemisesta lukuyksiköiden mukaan. Ilman DynamoDB Streams -ominaisuutta edellä mainituista kuluista koostuu kuukaudessa $6,99 mikäli käyttäjä on käyttänyt jo ilmaiset lukuyksikkönsä sekä tallennusmääränsä. [21]

3.1.3 Amazon Simple Storage Service (Amazon S3)

Amazon Simple Storage Service eli Amazon S3 on pilvitallennusalusta, johon tiedostoja tallennetaan resursseihin nimeltä ”bucket”. Yksi tiedosto voi olla maksimissaan 5 teratavun kokoinen eikä tiedostojen määrällä ole rajaa. S3 Bucketin käyttöoikeuksia on mahdollista säätää AWS Identity Access Management (IAM) avulla, josta kerrotaan myöhemmin tässä luvussa. Amazon S3 toimii yhteen monien eri AWS-palveluiden kanssa, joista yksi esimerkki on AWS Lambda. DynamoDB:n tavoin myös S3-tiedostot replikoituu käyttäjän valitseman alueen kolmen eri palvelinkeskuksen välillä. [22]

Amazon S3 -palvelun hinnoittelu perustuu tallennetun datan määrään sekä siihen kuinka usein tiedostoille tarvitsee tehdä tiettyjä operaatioita eli käytännössä siihen kuinka usein käyttäjän tarvitsee päästä käsiksi tiedostoihin eri toimintojen kautta. Hinnoitteluun vaikuttaa myös käyttäjän valitsema alue, mihin tiedostot säilötään. Lisäksi Amazon S3 tukee tiedostojen tallentamista eri luokkiin, joiden hinnoittelussa sekä ominaisuuksissa on eroja. Esimerkiksi käyttäjä voi valita, että tiedosto tallennetaan standardi-luokasta arkistointi-luokkaan, jolloin säilöntähinnat laskevat huomattavasti, mutta samalla tiedostoon käsiksi pääsy hankaloituu ja tulee kalliimmaksi. [23]

Seuraavaksi käsitellään standardi-luokassa säilöttyjen tiedostojen hintoja. Ensimmäiset 50 teratavua dataa maksaa kuukaudessa $0,023 gigatavulta, seuraavat 450 teratavua kuukaudessa $0,022 gigatavulta ja 500 teratavua ylittävältä osuudelta kuukaudessa maksetaan $0,021 gigatavulta. Tämän lisäksi Amazon S3 hinnoittelee pyynnöt, jotka on jaettu kolmeen ryhmään. Ensimmäiseen ryhmään kuuluu palveluun lataus, kopiointi, postitus ja listaus. Toiseen ryhmään kuuluu palvelusta lataus ja kaikki muut operaatiot. Kolmanteen ryhmään kuuluu poistaminen, joka on käyttäjälle maksuton toiminto.

Ensimmäisen ryhmän toiminnot maksavat $0,005 jokaista 1 000 pyyntöä kohti ja toisen ryhmän toiminnot maksavat $0,004 jokaista 10 000 pyyntöä kohti. Tietojen siirto saman alueen sisällä toiseen palveluun on maksutonta niin kuin myös tietojen siirto S3-palveluun saman alueen sisällä. Hinnoitteluun liittyy myös joitakin erikoistapauksia ja lisäpalveluita, joihin ei tämän työn osalta perehdytä. [23]

3.1.4 Amazon CloudWatch

Amazon CloudWatch eli vapaasti suomennettuna ”pilvivahti” on palvelu, jonka avulla voidaan monitoroida eri palveluiden resurssien käyttöä sekä toimintaa. CloudWatchin avulla voidaan asettaa hälytyksiä mikäli tietty palvelu kuluttaa esimerkiksi liikaa resursseja tai vastapainoisesti mikäli Amazon EC2 -instanssi eli virtuaalipalvelin ei ole käytössä se voidaan CloudWatchin kautta asettaa automaattisesti sammumaan. [24] CloudWatch näyttää myös esimerkiksi Lambda-funktioiden lokeja, mikäli koodissa on lokitus komennot käytössä. Tällä tavoin voidaan seurata Lambda-funktion toimintaa ja varmistaa, että se toimii oikein kaikissa tilanteissa. [25]

Amazon CloudWatch -palvelu on ilmainen, kun käyttö on vähäistä ja lisäpalveluiden tarve pientä. Ilmaiset rajat sisältävät perustavanlaatuista metriikkaa ja monet palvelut onnistuvat toimimaan rajojen sisällä, jolloin palvelulle ei koidu hintaa. Tästä syystä hinnoittelua ei sen tarkemmin käydä läpi. [26]

3.1.5 AWS Identity and Access Management (IAM)

AWS Identity Access Management on palvelu, jolla voidaan kontroloida eri palveluiden sekä käyttäjien käyttöoikeutta tiettyyn osiin palveluita ja resursseja. IAM tulee vastaan takuuvarmasti käytettäessä AWS-palveluita ja etenkin kun eri palveluita käytetään yhdessä. IAM:n avulla voidaan asettaa esimerkiksi Lambda-funktiolle käyttöoikeus kirjoittaa tiettyyn DynamoDB-tauluun, mutta samalla rajata poistamiskäyttöoikeus kokonaan pois. [27]

Käyttöoikeuksien antaminen palveluille tapahtuu luomalla uusi IAM-rooli tai käyttämällä useita valmiiksi generoituja rooleja. Tämän lisäksi roolille luodaan menettelytapa tai useita sellaisia, joissa määritellään roolin antamat oikeudet tiettyihin palveluihin. IAM:n avulla voidaan luoda myös uusia käyttäjiä tai käyttäjäryhmiä sekä asettaa näille

käyttöoikeuksia tiettyihin osiin AWS-palveluita. Tällä tavoin esimerkiksi yrityksellä voi olla AWS-tili, johon on luotu käyttäjiä eri työntekijöille rajatuin oikeuksin. [27]

3.2 NUVIZin syyt AWS:n valintaan

NUVIZ valitsi pilvipalvelut, koska se tarvitsee palveluita, joiden avulla on toteutettu mobiilisovelluksien taustatoiminnot sekä käyttäjätietojen säilytys. Palveluiden tulee olla käytettävissä aina ja koko maailman ympäri. Lisäksi palveluilta tarvitaan skaalautuvuutta, kun käyttäjämäärät kasvavat. Tällaisten palveluiden toteuttaminen itse olisi vaatinut niin suuria investointeja palvelininfrastruktuurin osalta, että se ei olisi ollut järkevää eikä startup-yritykselle edes mahdollista. [28]

NUVIZ valitsi AWS:n siksi, että NUVIZin tekemän kustannusarvioin mukaan se oli edullisin ja palveluiden kustannuskehitys oli lineaarisin. NUVIZ selvitti myös eri palveluiden teknisiä valmiuksia ja selvitys osoitti, että AWS on pisimmällä pilvipalveluiden kehityksessä, kuten esimerkiksi Serverless-arkkitehtuurissa. Edelläkävijyys Serverless-arkkitehtuurissa oli merkittävä tekijä, sillä NUVIZilla on halu toteuttaa palveluita mikropalveluiden avulla, joka mahdollistaa muun muassa palveluiden rinnakkaisen kehityksen toisistaan riippumatta. AWS myös tuki monia erilaisia teknologioita, kun taas esimerkiksi Azure oli rajoittunut Microsoftin tarjoamiin teknologioihin. [29]

NUVIZin mobiilisovellukset hyödyntävät muun muassa AWS Cognito -käyttäjän tunnistautumispalvelua, joka mahdollistaa käyttäjien kirjautumisen ja käyttäjätietojen tallentamisen. Lisäksi DynamoDB-tietokantoihin kerätään monenlaista tietoa niin käyttäjistä kuin laitteistakin. Näiden palveluiden keskiössä on kuitenkin AWS Lambda -palvelu, jonka avulla dataa käsitellään ja välitetään palveluiden välillä monien eri Lambda-funktioiden kautta.

NUVIZilla on haluttu käyttää mikropalveluarkkitehtuuria eli mallia, joka käytännössä tarkoittaa yhden kokonaisuuden pilkkomista pieniin helpommin hallittaviin osiin eli esimerkiksi Lambda-funktiohin. Tällä tavoin saadaan skaalautuva eheä kokonaisuus, jota on helppo hallinnoida, sillä palvelut ovat erillisiä. Palvelun pilkkominen mikropalveluihin auttaa sovelluskehitystä, sillä riski koko sovelluksen rikkomisesta minimoituu, mikä on taas merkittävä riski perinteistä monoliittista mallia hyödynnettäessä. Mikropalveluita hyödyntämällä sovelluskehitystä voidaan tehdä

pienemmissä osissa, jotka ovat erillisiä toisistaan, mutta toimivat silti yhteen. Yhden osan ollessa rikki muut toimivat normaalisti, ja tällä tavoin ongelma saadaan rajattua helposti. [30]

4 WWW-SOVELLUKSEN KEHITTÄMINEN

Tässä työssä www-sovelluksen kehittämisen voidaan ajatella jakautuvan selkeästi kahteen osaan, jotka ovat toisistaan hyvin irrallisia, mutta kummastakaan ei yksinään olisi juuri hyötyä. Nämä osat ovat frontend ja backend, jotka ovat termeinä tuttuja etenkin sovelluskehittäjille, sillä kehitystyö on usein jaettu näiden perusteella. Backend puolessa on DynamoDB-tietokanta sekä tietokantaan kirjoittamisen hoitava Lambda-funktio ja funktion käynnistävä tiedostontallennusalusta eli S3. Frontend puolta on itse www-sovellus.

Www-sovelluksen kehittäminen oli järkevää aloittaa backend puolen palveluiden konfiguroinnista ja ohjelmoinnista, sillä konfigurointi tietoja tarvittiin yhä www-sovelluksen kehittämisessä, oikeiden palveluiden ja niiden osien kutsumisessa.

4.1 Amazon S3

Amazon S3 -palvelu ei tämän työn osalta vaatinut suurempaa konfigurointia, sillä lokitiedostojen kerääminen oli jo toteutettu. Työn toteuttamisen kannalta on kuitenkin syytä käydä läpi hieman sitä, miten lokitiedostot S3:een oikein ilmestyvät.

NUVIZ:n kehittämä Android-mobiilisovellus sekä Androidiin perustuva HUD-laite lataavat S3 bucket -resurssiin omat lokitiedostonsa sovelluskaatumisista omiin sijainteihinsa. Lokitiedostot ovat ”.dmp”-päätteisiä tiedostoja ja ne ladataan seuraavanlaisten tiedostopolkujen päähän.

Mobiilisovelluksen lokitiedostojen tiedostopolku on:

S3Bucket/NuvizApp/SoftwareVersion/nuviz_app_crash_datetime.dmp

mikäli kyseessä on HUD-laitteen lokitiedosto, tiedostopolku on seuraavanlainen:

S3Bucket/NuvizDevice/FirmwareVersion/SoftwareVersion/nuviz_crash_deviceid_datetime.dmp

”S3Bucket” viittaa S3 bucket -resurssin juureen. HUD-laitteen lokitiedoston nimestä saadaan aikaleiman lisäksi laitetunnistetieto, jolla lokitiedosto voidaan yksilöidä tiettyyn

laitteeseen. Tämän lisäksi muita tiedostopolusta saatavia tietoja käytetään hyväksi tietojen suodattamisessa sekä kirjaamisessa tietokantaan.

4.2 AWS Lambda

AWS Lambda -palvelu on tämän järjestelmän keskeisin osa, sillä se hoitaa lokitiedoston datan suodattamisen sekä tietokantaan kirjoittamisen. AWS Lambda -palvelussa tehtiin uusi funktio, jolle määritettiin suorituksen laukaisevaksi tekijäksi S3-buckettiin ladattu uusi ”.dmp”-päätteinen tiedosto. Lambda-funktion ajoympäristöksi valittiin Node.js ja siten ohjelmointikieli on Javascript. Node.js valittiin ajoympäristöksi pilvessä siitä syystä, että yrityksen kaikki muutkin Lambda-funktiot oli toteutettu sen avulla, jolloin pärjätään yhdellä ohjelmointikielellä usean sijaan. Kuvassa 8 on esimerkki Lambda-funktiosta.

```
//dependencies
var AWS = require('aws-sdk');
var s3 = new AWS.S3();
var docClient = new AWS.DynamoDB.DocumentClient();

//handler
exports.handler = function(event, context, callback) {
  console.log("Event: " + JSON.stringify(event));
  console.log("Context: " + JSON.stringify(context));
  var srcBucket = event.Records[0].s3.bucket.name;
  var srcKey = decodeURIComponent(event.Records[0].s3.object.key);
  var params = {Bucket: srcBucket, Key: srcKey};
  var tableName = "ExampleDatabase";
  s3.getObject(params, function(err, data) {
    if (err) console.log(err, err.stack);
    else {
      var content = data.Body.toString();
      var filename = srcKey.substring(srcKey.lastIndexOf("/") + 1, srcKey.lastIndexOf(".dmp"));
      var parsedcontent = content.substring(0, content.lastIndexOf("\n"));
      var timetamp = new Date().toISOString().replace(/T/, ' ').replace(/\..+/, '');
      var dbparams = {
                TableName: tableName,
                Item: {
                    "FileName": filename,
                    "ParsedContent": parsedcontent,
                    "TimeStamp": timestamp,
                }
      };
      docClient.put(dbparams, function(err, data) {
          if (err) {
              console.error("Unable to add items to database" + err);
          } else {
              console.log("Added items to database");
          }
      });
    }
  });
};
```

Kuva 8. Esimerkki Lambda-funktiosta.

Kuvan 8 Lambda-funktio lukee S3-palveluun ladatusta tiedostosta tiedostonimen, osan tiedoston sisällöstä sekä generoi aikaleiman. Nämä tiedot lisätään määriteltyyn DynamoDB-tietokantaan funktion lopussa.

4.3 Amazon DynamoDB

DynamoDB-tauluun tallennetaan AWS Lambda -funktion suodattamat hyödylliset tiedot lokitiedostoista. Taulun luominen ja konfigurointi on yksinkertaista. Taulua luodessa annettiin tietokannalle pääavain sekä lajitteluavain. Avaimien tulee olla sellaisia, että ne yksilöivät jokaisen tietokannan rivin eli näiden kahden avaimen yhdistelmä on silloin uniikki. Tämän lisäksi näiden kahden avaimen täytyy olla sellaisia, että ne tunnetaan, jotta tietokantaan voidaan tehdä hakuja. Siten pääavaimeksi valittiin sovellustyyppi ja lajitteluavaimeksi aikaleima. Avaimille valittiin tyypiksi merkkijono. Muita avaimen tyyppejä ovat binääri sekä luku. Taulun luomisen yhteydessä taululle voidaan määrittää myös kirjoitus- ja lukuyksiköiden määrä, joiden arvo oletuksena on kummassakin 5. Arvoa voidaan myös muuttaa jälkeenpäin, joten alkuun valittiin oletusarvot. Loput taulun avaimet määrittyvät sitä mukaa kun tauluun kirjoitetaan tietoja Lambda -funktion avulla.

DynamoDB:n yksi hyvistä ominaisuuksista on se, että kaikkia taulun avaimia ei tarvitse tietää etukäteen vaan niitä voidaan lisätä tarvittaessa. Tämä on mahdollistanut sen, että taulua ei ole tarvinnut poistaa ja luoda uudestaan uuden avaimen takia. Tauluun on myös mahdollista lisätä lokaaleja toissijaisia avaimia, jotka ovat maksullisia lisätoimintoja. Lokaalien toissijaisten avainten avulla voidaan tehdä hakuja tuntematta määriteltyjä pää- ja lajitteluavainta. Lokaalin toissijaisen avaimen avulla voidaan tehdä haku esimerkiksi pääavaimen ja jonkun muun saman taulun avaimen perusteella.

4.4 Käyttöliittymän kehittäminen

Www-sovelluksen kehityksessä hyödynnettiin Amazonin tarjoamaa PHP ohjelmistonkehitystyökalua eli AWS-PHP-SDK:ta sekä Bootstrap-ohjelmakehystä. Amazonin PHP -ohjelmistonkehitystyökalu tarjoaa ohjelmointirajapinnat eri AWS-palveluiden hyödyntämiseen ja on tästä syystä avainasemassa www-sovelluksen kehittämisessä. Bootstrap-ohjelmakehyksellä toteutettiin ulkoasu, sillä siinä on valmiina tietynlaisia ulkoasuelementtejä, joiden avulla on helppo rakentaa yksinkertainen ulkoasu.

Valmiita elementtejä voi käyttää sellaisenaan tai muokkaamalla niitä omien tarpeiden mukaan. Tällä tavoin säästyy merkittävästi aikaa ulkoasun tekemisessä.

4.4.1 AWS-PHP-SDK

Amazon tarjoaa oman ohjelmistokehitystyökalun (SDK) monille eri ohjelmointikielille. Tällaisia ohjelmointikieliä ovat muun muassa: Java, C++, PHP ja Python. SDK:t eri ohjelmointikielille mahdollistavat eri alustojen käytön ja tukevat siten AWS-palveluiden käyttäjien tarpeita. [31] Amazonin tarjoama AWS-PHP-SDK on tämän www-sovelluksen toiminnan ydin, sillä se mahdollistaa AWS-palveluiden käytön. AWS-PHP-SDK on nimensä veroisesti PHP-ohjelmointikielelle tarkoitettu ohjelmistokehitystyökalu, jossa on ohjelmointirajapinnat (API) AWS-palveluille. APIen avulla on mahdollista kutsua tietyn palvelun tiettyjä toimintoja www-käyttöliittymästä. PHP on palvelinpäässä toimiva ohjelmointikieli, jonka avulla voidaan hyödyntää palvelimen resursseja ja toteuttaa siten esimerkiksi käyttäjän kirjautumisjärjestelmä tietokannan avulla. Tässä työssä PHP:n avulla on toteutettu erilaiset tietokantahaut sovelluskaatumisista. PHP:n käyttöön päädyttiin siksi, että se oli entuudestaan tuttua. Kuvassa 9 on esimerkkipätkä koodista, jolla toteutetaan DynamoDB-tietokantahaku.

```php
$sdk = new Aws\Sdk([
  'region' => 'eu-west-1',
  'version' => 'latest',
  'credentials' => $creds
]);
$dbclient = $sdk->createDynamoDb();
$response = $dbclient->query([
    'TableName' => 'nuviz-dump-db',
    'KeyConditionExpression' => 'UnitType = :v_UnitType and #DateTime BETWEEN :v_DateTimeS and :v_DateTimeE',
    'ExpressionAttributeNames' => [
      '#DateTime' => 'TimeStamp'
    ],
    'ExpressionAttributeValues' => [
      ':v_UnitType' => ['S' => $software ],
      ':v_DateTimeS' => ['S' => $datetime2],
      ':v_DateTimeE' => ['S' => $datetime]
    ],
    'ScanIndexForward' => false
]);
```

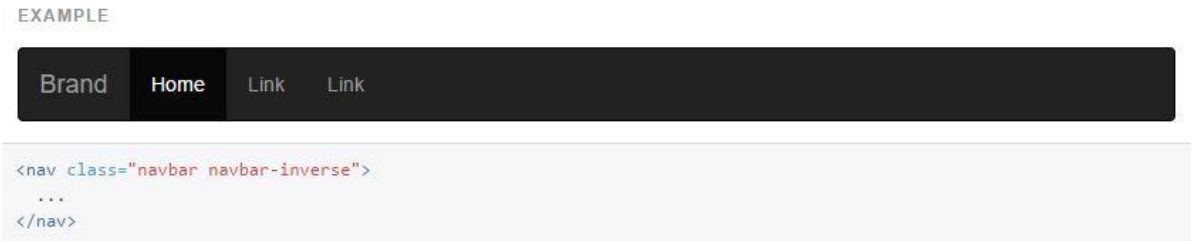
Kuva 9. Esimerkkikoodi PHP:lla toteutetusta DynamoDB-tietokantahausta.

Kuvan 9 esimerkkikoodi edellyttää AWS-PHP-SDK:n lisäämistä sekä muuttujien määräämistä. Tämän lisäksi DynamoDB-tietokantahaku edellyttää tunnuksien luomista AWS IAM -palvelun kautta. Toimiessaan kuvan 9 koodi tekee tietokantahaun

määritellystä taulusta seuraavilla ehdoilla: ”UnitType”-avaimen arvo vastaa muuttujan ”$software” arvoa sekä ”TimeStamp”-avaimen arvo on muuttujan ”$datetime” ja ”$datetime2” väliltä.

4.4.2 Bootstrap

Bootstrap on frontend www-ohjelmistokehys, jonka avulla voidaan tehdä responsiivisia mobiililaitteita tukevia nettisivuja. Bootstrap tarjoaa joukon valmiita ulkoasuelementtejä, joita on helppo ottaa käyttöön sekä kustomoida omien tarpeiden mukaan. [32] Se, miksi valittiin juuri tämä ohjelmistokehys monien muiden sijasta johtuu siitä, että tätä ohjelmistokehystä oli käytetty jo aiemmin. Mobiililaitteiden tuella ei niinkään ollut merkitystä tämän työn osalta, sillä www-sovellus on kohdennettu ensisijaisesti PC-päätteille. Kuvassa 10 ja 11 on esillä esimerkit Bootstrapin valmiista ulkoasuelementeistä.



Kuva 10. Esimerkki yhdestä Bootstrapin valmiista navigointikenttäelementistä [33].

Kuvassa 10 on esillä perustavanlaatuinen valmis navigointikenttäelementti. Tätäkin on mahdollista muokata lisäämällä omaa CSS-koodia, jolla Bootstrapin valmis CSS-koodi osittain korvataan. Kyseisen navigointikentän ja muut valmiit elementit saa käyttöön käyttämällä koodissa tiettyjä Bootstrapin dokumentoimia CSS-luokkia, kuten yllä olevassa esimerkissä on käytetty luokkia ”navbar” ja ”navbar-inverse”. Luokat löytyvät dokumentoituina Bootstrapin nettisivuilta.

Kuva 11. Esimerkki Bootstrapin valmiista pudotusvalikkonappielementeistä [33].

Myös kuvassa 11 oleva esimerkki pudotusvalikkonappielementeistä toimii samalla tavalla kuin kuvan 10 esimerkki. Näiden esimerkkien lisäksi Bootstrapillä on lukemattomia muita valmiita elementtejä, mitä voidaan tarvita erilaisia nettisivuja luodessa. Tämän lisäksi Bootstrap tarjoaa myös valmiita nettisivupohjia, joista on helppo päästä liikkeelle. Käyttämällä Bootstrappiä, on ollut mahdollista säästää aikaa huomattavasti ulkoasujen tekemisissä ja keskittyä enemmän toiminnalliseen puoleen. Valmiita elementtejä on jouduttu kuitenkin muokkaamaan melko paljon, jotta www-sovelluksen sisältö on saatu näytettyä järkevällä tavalla.

4.5 Testaus

Testaus on tärkeä osa sovelluksien kehitystä, jotta voidaan varmistaa sovelluksien toiminta jatkossakin vaikka koodiin tehtäisiinkin muutoksia. Testausta toteutettiin järjestelmän frontend- ja backend-päissä erikseen niiden eri kehitysvaiheissa sekä jatkokehityksen yhteydessä. Useimmat järjestelmään tehdyt muutokset olivat sellaisia, mitkä eivät vaatineet erikseen molempien frontend- ja backend-päiden testausta. Tästä syystä testaus on ollut suurimmaksi osaksi suhteellisen helppoa. Pieni vaikeus testaamisessa liittyy kuitenkin siihen, että kaikkia mahdollisia sovelluskaatumislokien formaatteja ei tunneta, jolloin Lambda-funktio saattaa tulkita lokeja väärin vaikka formaattien vaihtelu onkin melko pientä. Tämä onkin edellyttänyt useita korjauksia Lambda-funktioon uuden tyyppisten kaatumislokien ilmaantuessa. Työn

kirjoittamishetkellä Lambda-funktiosta on jo kehittynyt niin kattava, että virhetulkintojen määrä on melko marginaalinen.

4.5.1 Backend

Backend-testaus toteutettiin alkuun siten, että Lambda-funktiota testattiin työpöydällä esimerkkilokitiedostoilla Node.js ympäristössä. Kun funktio toimi riittävän hyvin työpöydällä se lisättiin AWS Lambda -palveluun omaksi funktiokseen. Lambda-funktion toimintaa pystyy myös auttavasti testaamaan AWS Lambda -palvelun omalla testaustoiminolla, jossa Lambda-funktiolle generoidaan tekaistu tapahtuma, joka laukaisee funktion suorituksen. Testausta jatkettiin lisäämällä manuaalisesti lokitiedostoja S3-palveluun ja seuraamalla DynamoDB-tietokantaa sekä AWS CloudWatch -palvelusta Lambda-funktion lokeja.

Tämän lisäksi luotiin erikseen Lambda-funktio testausta varten, jonka avulla on mahdollista testata funktion uusia kehitettyjä ominaisuuksia rikkomatta tai häiritsemättä toimivaa funktiota ja järjestelmää. Testausfunktio määritettiin laukeamaan samalla tavalla kuin oikea Lambda-funktiokin, mutta vain jos ladattu tiedosto ladattiin alikansioon nimeltä ”test”.

4.5.2 Frontend

Frontend-testausta toteutettiin aina, kun www-sovellukseen luotiin uusia ominaisuuksia. Tämän jälkeen kun testaus oli valmis, uusi versio www-sovelluksesta voitiin ottaa käyttöön koko yrityksessä. Välillä virheitä löytyi vielä sen jälkeen kun uusi versio oli julkaistu, mutta pääosin testauksessa jo oli onnistuttu saamaan virheet kiinni ja tekemään tarvittavat korjaukset.

Frontend-testaus oli hyvin manuaalista ja vaati paljon aikaa kehitystyön ohella. Esimerkiksi uuden tietokantahaun suodatustoiminnon kehittäminen vaati usein kaikkien mahdollisten tietokantahakukombinaatioiden läpi käymisen siltä varalta, että joku toinen ominaisuus olisi hajonnut uutta kehitettäessä.

5 YHTEENVETO

Tässä työssä suunniteltiin ja toteutettiin Amazon AWS -pilvipalveluita hyödyntävä järjestelmä www-käyttöliittymällä. Keskeisessä osassa oli Serverless-arkkitehtuuria toteuttava AWS Lambda -palvelu. Järjestelmän tarkoitus oli kerätä S3-palveluun kertyviä Android-sovelluksien kaatumislokitiedostoja, jonka jälkeen suodattaa niistä oleelliset tiedot ja tallentaa ne DynamoDB-tietokantaan. Käyttöliittymällä oli tarkoitus pystyä tekemään hakuja tietokannan sisällöstä.

Työ toteutettiin kahdessa osassa. Ensiksi kehitettiin backend-osat ja viimeisenä frontend eli www-käyttöliittymä. Työn tekojärjestys oli perusteltua, sillä backend-osien konfigurointitietoja tarvittiin www-käyttöliittymän kehittämisessä. Www-sovelluksen backend-osiin kuuluu S3-bucket, DynamoDB-tietokanta sekä Lambda-funktio. Työn toteutuksessa olisi voitu hyödyntää Serverless-ohjelmistokehystä, jonka avulla järjestelmän uudelleen asentaminen olisi nopeampaa eikä vaatisi kovin suurta manuaalista työtä. Tämä kuitenkin sivuutettiin kiireen takia, sillä Serverless-ohjelmistokehyksen toimintaan perehtyminen olisi vaatinut ylimääräistä aikaa eikä sen käyttö ollut välttämätöntä.

Työssä onnistuttiin luomaan toimiva järjestelmä, ja sen avulla on pystytty seuraamaan erilaisten sovelluskaatumisten esiintymisiä ja syitä sekä tekemään vaadittavia korjauksia sovelluksien lähdekoodeihin. Www-käyttöliittymä on helppokäyttöinen ja antaa perustavanlaatuista tietoa sekä statistiikkaa esiintyneistä kaatumisista. Käyttöliittymän avulla voidaan muun muassa tehdä hakuja samantyyppisistä kaatumisista ja siten tulkita niiden esiintyvyyttä ja vakavuutta käyttäjäkokemuksen kannalta. Järjestelmän avulla on onnistuttu parantamaan merkittävästi ohjelmistojen laatua.

Tässä työssä toteutettu järjestelmä on opinnäytetyön kirjoittamishetkellä aktiivisessa käytössä toimeksiantajalla, ja sille on esitetty jatkokehitysideoita. Jatkokehitysideoita on esitetty muun muassa havainnollistavan grafiikan sekä kommentointimahdollisuuden puolesta. Tulevaisuudessa järjestelmää on tarkoitus edelleen kehittää ja lisätä siihen uusia ominaisuuksia tukemaan ja helpottamaan käyttöä. Lisäksi on tarkoitus siirtyä käyttämään Serverless-ohjelmistokehystä.

LÄHTEET

[1] Tero Toomas 2016. Amazon AWS –pilvipalveluiden integrointi olemassa olevaan sovellukseen. [www-dokumentti]. Opinnäytetyö. Metropolia Ammattikorkeakoulu. Saatavilla: http://theseus.fi/bitstream/handle/10024/121846/Tero_Toomas.pdf?sequence=1. (Luettu: 6.5.2017).

[2] Miina Koskinen 2016. Microservices and containers – Benefits and Best Practices. [www-dokumentti]. Opinnäytetyö. Turun Ammattikorkeakoulu. Saatavilla: http://theseus.fi/bitstream/handle/10024/116329/Koskinen_Miina.pdf?sequence=1. (Luettu: 6.5.2017).

[3] Matthew Portnoy. Virtualization Essentials. Wiley, 2012. [www-dokumentti]. E-kirja. Saatavilla: https://ebookcentral-proquest-com.ezproxy.turkuamk.fi/lib/turkuamk-ebooks/detail.action?docID=821849. (Luettu: 12.3.2017).

[4] Customer Success Stories. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/solutions/case-studies/all/. (Luettu: 28.2.2017).

[5] Immo Salo. Cloud Computing palvelut verkossa. Jyväskylä: WSOYpro Oy, 2010.

[6] Operating-system-level virtualization. Wikipedia. [www-dokumentti]. Saatavilla: https://en.wikipedia.org/wiki/Operating-system-level_virtualization. (Luettu 20.4.2017).

[7] Serverless Computing. Wikipedia. [www-dokumentti]. Saatavilla: https://en.wikipedia.org/wiki/Serverless_computing. (Luettu 20.4.2017).

[8] Petteri Heino. Pilvipalvelut. Helsinki: Talentum Media Oy, 2010.

[9] Amazon Web Services (AWS) – Cloud computing services. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com. (Luettu: 20.4.2017).

[10] Global Infrastructure. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/about-aws/global-infrastructure/. (Luettu: 7.5.2017).

[11] Region Table. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/. (Luettu: 20.4.2017)

[12] AWS vs Azure vs Google Cloud Market Share 2017. Skyhigh Networks. [www-dokumentti]. Saatavilla: https://www.skyhighnetworks.com/cloud-security-blog/microsoft-azure-closes-iaas-adoption-gap-with-amazon-aws/. (Luettu: 20.4.2017).

[13] Microsoft Azure: Cloud Computing Platform & Services. Microsoft. [www-dokumentti]. Saatavilla: https://azure.microsoft.com/en-us/. (Luettu: 20.4.2017).

[14] Azure Regions | Microsoft Azure. Microsoft. [www-dokumentti]. Saatavilla: https://azure.microsoft.com/en-in/regions/. (Luettu: 20.4.2017).

[15] Google Cloud Platform. Wikipedia. [www-dokumentti]. Saatavilla: https://en.wikipedia.org/wiki/Google_Cloud_Platform. (Luettu: 20.4.2017).

[16] Products & Services | Google Cloud Platform. Google. [www-dokumentti]. Saatavilla: https://cloud.google.com/products/. (Luettu: 20.4.2017).

[17] Global Locations – Regions & Zones | Google Cloud Platform. Google. [www-dokumentti]. Saatavilla: https://cloud.google.com/about/locations/#regions-tab. (Luettu: 1.5.2017).

[18] AWS Lambda | Product Details. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/lambda/details/. (Luettu: 22.4.2017).

[19] AWS Lamba | Pricing. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/lambda/pricing/. (Luettu: 22.4.2017).

[20] Amazon DynamoDB Product Details. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/dynamodb/details/. (Luettu: 22.4.2017).

[21] Amazon DynamoDB Pricing – Amazon Web Services (AWS). Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/dynamodb/pricing/. (Luettu: 23.4.2017).

[22] Object Storage Details – Amazon Simple Storage Service (S3). Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/s3/details/. (Luettu: 22.4.2017).

[23] Cloud Storage Pricing – Amazon Simple Storage Service (S3). Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/s3/pricing/. (Luettu: 22.4.2017).

[24] Amazon CloudWatch Product Details. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/cloudwatch/details/. (Luettu: 22.4.2017).

[25] Accessing Amazon CloudWatch Logs for AWS Lambda – AWS Lambda. Amazon Web Services. [www-dokumentti]. Saatavilla: http://docs.aws.amazon.com/lambda/latest/dg/monitoring-functions-logs.html. (Luettu: 22.4.2017).

[26] AWS | Amazon CloudWatch | Pricing. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/cloudwatch/pricing/. (Luettu: 1.5.2017).

[27] IAM Product Details – Amazon Web Services (AWS). Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/iam/details/. (Luettu: 22.4.2017).

[28] Keskustelut NUVIZin Tero Nordströmin (Head of Software) kanssa.

[29] Backend Technology/Platform Selection. Tero Nordström. Nuviz Oy. [powerpoint-esitys].

[30] Sam Newman. Building Microservices. Sebastopol: O'Reilly Media, Inc, 2015.

[31] SDKs and Programming Toolkits for AWS. Amazon Web Services. [www-dokumentti]. Saatavilla: https://aws.amazon.com/tools/. (Luettu: 6.5.2017).

[32] Bootstrap · The world's most popular mobile-first and responsive front-end framework. Bootstrap. [www-dokumentti]. Saatavilla: http://getbootstrap.com. (Luettu: 6.5.2017).

[33] Components · Bootstrap. Bootstrap. [www-dokumentti]. Saatavilla: http://getbootstrap.com/components/. (Luettu: 6.5.2017).