jamk.fi

DevOps-käytäntöjen hyödyntäminen Microsoft Azure -pilvialustalla

Joni Montonen

Opinnäytetyö
Toukokuu 2020
Tekniikan ala
Insinööri (AMK), tieto- ja viestintätekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Montonen, Joni	Opinnäytetyö, AMK	Toukokuu 2020
	Sivumäärä 38	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
DevOps-käytäntöjen hyödyntäminen Microsoft Azure -pilvialustalla

Tutkinto-ohjelma
Tieto- ja viestintätekniikka

Työn ohjaaja(t)
Niemi, Kari

Toimeksiantaja(t)
Tridea Oy

Tiivistelmä

Digitalisaation seurauksena ihmisten kulutuskäyttäytyminen ja siten myös odotukset yrityksiä kohtaan ovat muuttuneet. Digitaalisia palveluita tuottavien yritysten on vastattava kysyntään entistä nopeammin ja tehokkaammin. DevOps-toimintamalli pyrkii nopeuttamaan ohjelmiston kehitykseen, testaukseen ja julkaisuun liittyviä toimintoja automatisoinnin avulla. DevOps-toimintamallilla yritykset voivat yleensä tehostaa toimintaansa ja saavuttaa kilpailuedun markkinoilla.

Opinnäytetyössä tutkittiin DevOps-toimintamallia, siihen kuuluvia menetelmiä ja Microsoft Azuren DevOps-palveluita ja työkaluja. Lisäksi esitettiin konkreettinen ratkaisu niiden hyödyntämisestä Microsoft Azuren pilvialustalla. Toimeksiantajana työssä toimi asiakaskokemuksen kehittämiseen, johtamiseen ja mittaamiseen erikoistunut yritys Tridea Oy.

Opinnäytetyön tuloksena oli Azure DevOps-projektina luotu web-sovellus, joka hyödyntää DevOps-toimintamallia Azure DevOps-palvelujoukon avulla. Myös sovelluksen julkaisu tapahtui Microsoft Azuren alustalle. Sovelluksen käyttöliittymän rakentamiseen käytettiin toimeksiantajalle tuttua React.js kirjastoa. Työn tuloksella osoitettiin joidenkin DevOps-toimintamallin menetelmien, React-sovelluskehityksen ja Microsoft Azure -pilvialustan yhteensopivuus.

Azure DevOps-projektin avulla voidaan onnistuneesti hyödyntää osaa teoriaosuudessa esiin tulleita menetelmiä ja käytänteitä. Opinnäytetyö tarjoaa sekä teorian että käytännön osalta hyvän pohjan DevOps-menetelmien käytöstä ohjelmiston kehityksessä ja julkaisussa. DevOps-toimintamallia voidaan projektin pohjalta laajentaa yrityksen tarpeen mukaan.

Avainsanat (asiasanat)
DevOps, Microsoft Azure, jatkuva integraatio, jatkuva julkaisu

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Montonen, Joni	Bachelor's thesis	May 2020
		Language of publication: Finnish
	Number of pages: 38	Permission for web publication: x

Title of publication

Utilization of DevOps practices on Microsoft Azure cloud platform

Degree programme

Information and Communication Technology

Supervisor(s)

Niemi, Kari

Assigned by

Tridea Oy

Abstract

As a result of digitalisation, people's consumption behavior and thus also their expectations towards companies have changed. Companies providing digital services need to respond to demand more quickly and efficiently. The DevOps operating model aims to speed up software development, testing and publishing through automation. With the DevOps operating model, companies can usually increase their efficiency and gain a competitive advantage in the market.

The thesis examined the DevOps operating model, its methods and Microsoft Azure DevOps services and tools. In addition, a concrete solution was presented for utilizing them on the Microsoft Azure cloud platform. The client was Tridea Oy, a company specializing in the development, management and measurement of customer experience.

The result of the thesis was a web application created in the Azure DevOps project, which utilizes the DevOps operating model with the help of the Azure DevOps service set. The application was also released on the Microsoft Azure platform. The React.js library familiar to the client was used to build the application's user interface. The result of the work demonstrated the compatibility of some methods of the DevOps operating model, React application development and Microsoft Azure cloud platform.

The Azure DevOps project can be used to successfully utilize some of the methods and practices identified in the theory section. The thesis provides a good basis for the use of DevOps methods in software development and publishing, both in theory and practice. Based on the project, the DevOps operating model can be expanded according to the company's needs.

Keywords/tags (subjects)

DevOps, Microsoft Azure, continuous integration, continuous release

Miscellaneous (Confidential information)

Sisältö

Kuviot

1 Työn lähtökohdat

1.1 Tausta ja toimeksiantaja

DevOps-toimintamalli on tällä vuosikymmenellä noussut yleiseksi puheenaiheeksi ohjelmistokehittäjien piirissä. Aihetta käsitellään eri julkaisuissa, sosiaalisessa mediassa ja blogeissa. DevOps-termiä on alettu käyttää jopa työnhaun yhteydessä, vaikka sen merkitykset voivat olla varsin moninaiset. Usein DevOps yhdistetään ketterään kehitykseen, jonka tavoitteena on tuottaa lisäarvoa mahdollisimman nopeasti pieninä paloina (Kylmäkoski 2018). Amazonin ja Netflixin kaltaisten suurten ohjelmistofirmojen menestystä selitetäänkin usein DevOps-toimintamallin hyödyntämisellä (Null n.d.).

Digitalisaation seurauksena ihmisten kulutuskäyttäytyminen ja siten myös odotukset yrityksiä kohtaan ovat muuttuneet. Kun markkinat vaativat nopeutta, tehokkuutta ja ketteryyttä, on digitaalisessa maailmassa toimivien yritysten myös muututtava. DevOps on kokeilevan kehityksen toimintamalli, joka perustuu mm. järkevien työkaluvalintojen hyödyntämiseen. Sen tarkoitus on helpottaa sovellusten jatkuvaa versiointia ja siten myös uusien, markkinoiden vaatimuksiin vastaavien ominaisuuksien luomista. DevOps onkin usein keino, jolla yritykset voivat muuttaa ja parantaa toimintaansa. (DevOps on ketteryyden uusi määritelmä 2017.)

Toimeksiantajana opinnäytetyölle toimi Tridea Oy. Tridea on asiakaskokemuksen kehittämiseen, johtamiseen ja mittaamiseen erikoistunut yritys. Tridean yhtenä palveluna on web-pohjainen, *BisLenz -johdon raportointitila*, joka yhdistää olennaisimmat mittarit asiakaskokemuksen kokonaiskuvan luomiseksi (Mitä teemme n.d.). Palvelua kehitetään jatkuvasti, joten työskentelytavat ovat siten myös tärkeässä roolissa.

1.2 Tavoitteet ja rajaukset

Opinnäytetyön tavoitteena oli toimittaa toimeksiantajalle selvitys DevOps-toimintamallista ja sen merkityksestä web-sovelluskehityksessä ja digitaalisia palveluita tar-

joavassa liiketoiminnassa. Tarkasteltiin myös, mistä DevOps-toimintamallin käytännöistä ja toimintatavoista voisi olla hyötyä toimeksiantajalle nyt tai tulevaisuudessa. Tavoitteena oli myös selvittää, onko käytännöissä jotain ongelmallista tai haitallista toimeksiantajan näkökulmasta. Lisäksi selvitettiin, miten DevOps-käytäntöjä voidaan toteuttaa, kun sovellus julkaistaan Microsoft Azure -pilvialustalle, ja millaisia työkaluja Microsoft Azure tarjoaa tähän käyttötarkoitukseen.

Koska DevOps käsitettä ei ole virallisesti ja tarkkaan määritelty, opinnäytetyö lähestyi käsitettä, toimeksiantajan tarpeiden mukaan, pilvipalveluiden tarjoamien menetelmien ja työkalujen kautta. Työssä käsiteltiin siis vain sellaisia menetelmiä, joita suurimmat pilvipalveluita ja DevOps-työkaluja tarjoavat yritykset pitävät DevOpsiin kuuluvina menetelminä. Tällaisia yrityksiä ovat mm. Microsoft ja Amazon. Työssä tuotiin esiin myös Microsoft Azuren palveluita nimenomaan web-sovelluksen julkaisun ja DevOpsin näkökulmasta. Microsoft Azuren lukuisat muut palvelut rajattiin työn ulkopuolelle.

1.3 Tutkimusmenetelmä

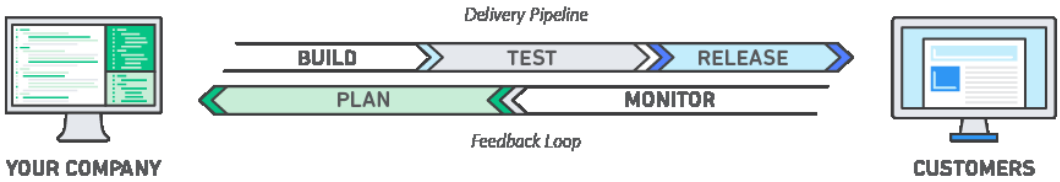
Opinnäytetyön tutkimusmenetelmänä käytettiin tutkimuksellista kehittämishanketta. Kyseessä oli siis ns. toiminnallinen opinnäytetyö, jossa tehtiin projektina käytännön toteutus testisovelluksen julkaisusta. Työllä pyrittiin lisäämään ohjelmistokehitysprosessin automatisointia toimeksiantajalle uusilla DevOps-menetelmillä. Käytännön toteutuksen tehtävänä oli tutkia DevOps-toimintamallin hyödyntämistä Microsoft Azure -pilvialustalla. Tutkimuksessa käytettiin mm. Azure DevOps -palvelujoukkoa, jonka avulla eri menetelmiä otettiin käyttöön testisovelluksen kehityksessä ja julkaisussa. Testisovelluksella pyrittiin lisäksi tutkimaan BisLenz -palvelussa käytetyn React-sovelluskehityksen ja Microsoft Azure -pilvialustan yhteensopivuutta.

2 DevOps

2.1 DevOpsin historia ja määritelmä

DevOps on ohjelmistokehityksessä käytetty termi, joka tulee sanoista *Development* (kehitys) ja *Operations* (tuotannon ylläpito/järjestelmänhallinta). Yleisesti DevOpsin katsotaan syntyneen vuonna 2009, jolloin Gentissä, Belgiassa, pidettiin ensimmäinen *Devopsdays*-nimeä kantava konferenssi. Tämä oli seurausta tarpeelle kehittää ketteriä ohjelmistokehitysmenetelmiä siten, että kehitystiimit (Dev) ja infrastruktuuria ylläpitävät tiimit (Ops) toimisivat paremmin yhteistyössä. Devopsdays-konferenssin inspiraationa toimi aiemmin samana vuonna kahden Flickr yhtiön työntekijän pitämä esitelmä ”10+ Deploys per Day: Dev and Ops Cooperation at Flickr”. Esitelmässä tuotiin esille menetelmiä, joita nykyisin pidetään DevOps-menetelminä. (Mezak 2018.)

DevOps-toimintamallin perusajatus on lisätä yrityksen kykyä tarjota sovelluksia, palveluita ja myös kehittää nykyisiä tuotteita nopeammin, kuin perinteisiä ohjelmistokehitysmalleja käyttävät yritykset kykenevät. Nopeudella palvellaan asiakasta ja saavutetaan kilpailuetu markkinoilla. Kuvio 1 esittää DevOps-toimintamallia palvelun toimittajan ja asiakkaan välillä. (What is DevOps? N.d.a.)

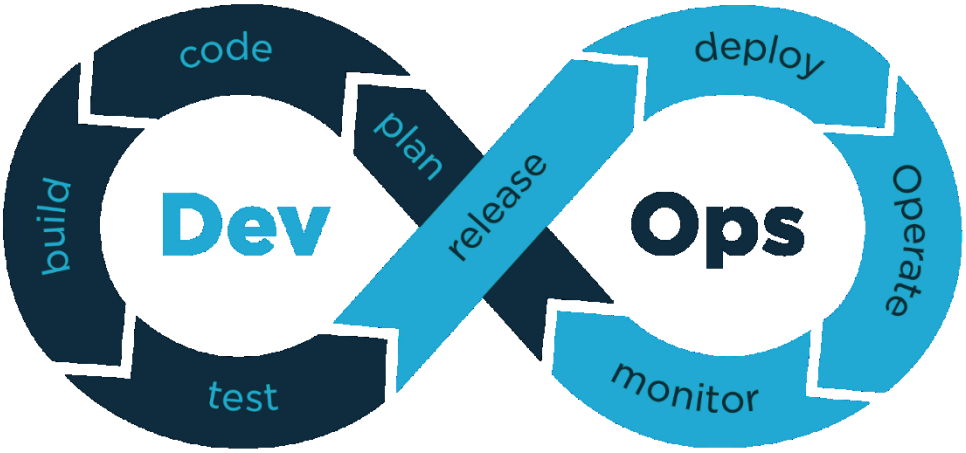


Kuvio 1. DevOps-toimintamalli (What is DevOps? N.d.a.)

Rautosen (2013) mukaan DevOps on tietynlainen jatkumo ketterille ohjelmistokehitysmenetelmille, joihin on lisätty automatisoitu järjestelmänhallinta. Siinä missä ketterät menetelmät ja iteratiivinen ohjelmistokehitys paransivat asiakkaan ja toimittajan välistä viestintää, DevOps vastaa jatkuvan integraation ja automatisoidun tes-

tauksen tarpeisiin ja nopeuttaa sovellusten päivittämistä, testausta ja julkaisua. Pilvipalvelut ja virtualisointi puolestaan mahdollistavat automatisoinnin ja muuttuviin vaatimuksiin vastaamisen infrastruktuuritasolla.

Palvelinten mekaanisen ylläpidon väheneminen ja infrastruktuurinhallinnan muuttuminen ovat vähentäneet perinteistä järjestelmäasiantuntijan työtä. DevOps-menetelmien ja toimintatapojen avulla pyritään myös parantamaan sovelluskehittäjien ja järjestelmäasiantuntijoiden välistä viestintää ja yhteistyötä, jolloin infrastruktuurin hallinnasta tulee osa sovelluskehitystä. DevOps lisää sovelluskehittäjien sekä sosiaalisia että teknisiä osaamisvaatimuksia, ja järjestelmäasiantuntijoiden on puolestaan omaksuttava uusia toimintatapoja sovelluskehittäjiltä. (Rautonen 2013.). Kuviossa 2 on kuvattu ohjelmistokehityksen työvaiheita ja em. työtiimien rooleja.



Kuvio 2. DevOps roolit ja työvaiheet (Oteyowo 2018.)

2.2 DevOps-käytännöt

2.2.1 Jatkuva integraatio ja jatkuva toimitus

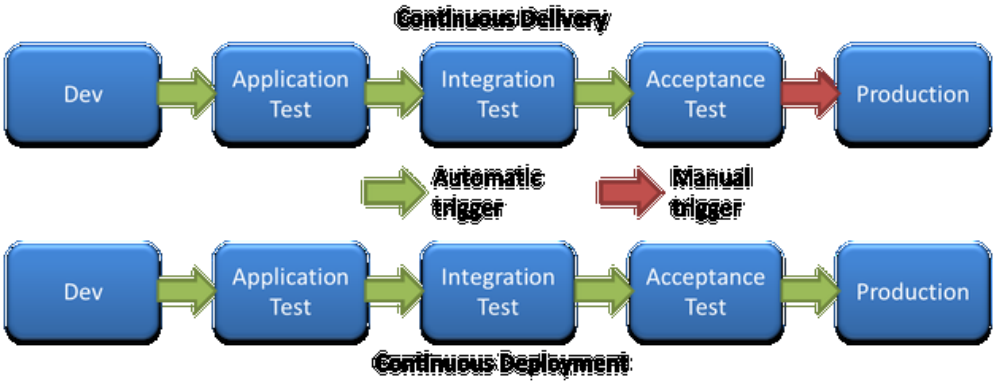
Jatkuvalla integraatiolla (CI, Continuous Integration) tarkoitetaan DevOps-toimintamallin vaihetta, jossa koodin kokoaminen ja testaus tehdään automaattisesti, kun ohjelmistonkehittäjä tekee muutoksia versionhallintajärjestelmään (esim. Git). Jatkuva

integraatio on siis julkaisuprosessin alkuvaiheen käytäntö, jossa automatisoinnin lisäksi keskeinen periaate on integroida muutokset usein. (What is Continuous Integration? N.d.)

Guckenheimerin (2017) mukaan jatkuva integraatio on noussut parhaaksi käytännöksi, koska ohjelmistokehittäjät työskentelevät usein erillään työryhmästä ja heidän on integroitava muutoksensa muiden työryhmäläisten koodikantaan. Jos integroimista ei tehdä usein ja lyhyin väliajoin, voi syntyä monia yhdistämisristiriitoja. Jatkuva integraatio vaatii kehitysryhmän koodin yhdistämistä jatkuvasti jaettuun versionhallintaan ongelmien välttämiseksi. Tällaisia ongelmia voivat olla esimerkiksi vaikeasti korjattavat bugit, päällekkäiset työt tai toisistaan poikkeavat koodistrategiat.

Jatkuvan toimituksen (CD, Continuous Delivery) käytännössä jatkuvalla integroinnilla tehdyt koodimuutokset valmistellaan julkaistavaksi tuotantoon. Se on siis ikään kuin laajennettu versio jatkuvasta integroinnista. Valmisteluun kuuluu automatisoituja kääntö- ja testausvaiheita, jotka suoritetaan testaus- ja/tai tuotantoympäristöissä. Testaukset voivat sisältää esimerkiksi käyttöliittymä-, integraatio- tai ohjelmointirajapinnan luotettavuustestauksen. Testausvaiheet auttavat kehittäjiä ennaltaehkäisemään ongelmia tuotannossa ja validoimaan päivitykset perusteellisemmin. Kun jatkuvan toimituksen käytäntöjä toteutetaan oikein, on kehittäjillä aina käytettävissään standardoidun testiprosessin läpikäynyt, julkaisuvalmis artefakti (Artifact). Jatkuvassa toimituksessa julkaisuvalmiit artefaktit hyväksytään tuotantoon manuaalisesti. (What is Continuous Delivery? N.d.)

DevOps-käytäntöihin kuuluu myös ns. jatkuva julkaisu (Continuous Deployment), joka sisältää samat työvaiheet kuin jatkuva toimitus, mutta erona on se, että myös tuotantoon julkaisu on automatisoitu. Jatkuvan toimituksen ja jatkuvan julkaisun työvaiheet on kuvattu kuviossa 3.



Kuvio 3. Jatkuvan toimituksen ja jatkuvan julkaisun ero (Hering 2014.)

2.2.2 Mikropalvelut

Mikropalveluilla tarkoitetaan tietynlaista ohjelmiston suunnittelussa käytettyä arkkitehtuuria, jossa sovellus koostuu useista, omina prosesseinaan toimivista, tiettyyn tarkoitukseen rajoitetuista palveluista. Mikropalvelut toteuttavat yhdessä sovellukselta halutut toiminnot viestimällä toisilleen, tyypillisesti http-pohjaisen ohjelmointirajapinnan (API) tai muun kevyen mekanismin kautta. Mikropalvelut voidaan toisistaan riippumatta toteuttaa eri ohjelmointikielillä tai ohjelmistokehyksillä, kunhan toteutus noudattaa rajapintamäärittelyä. Tämä antaa sovelluksen kehittäjälle vapauden valita mieleisensä tai tarkoitukseen parhaiten sopivan ohjelmointikielen. Lisäksi tällaisten pienten kokonaisuuksien tehtävänasettelu tekee sovelluskehittäjälle työstä selkeämpää ja helpommin hallittavaa. (Hämäläinen 2016; What is DevOps? N.d.)

Mikropalvelut sopivat erinomaisesti jatkuvaan toimitukseen mm. niiden helpon testauksen vuoksi. Esimerkiksi uudelle käyttöliittymäelementille voidaan tehdä vertailevaa a/b-testausta ohjaamalla käyttäjiksi ensiksi vaikka kymmenen prosenttia kaikista käyttäjistä ja päättää vasta tulosten perusteella, halutaanko uudistus ottaa käyttöön täyspainoisesti. Myös palvelinten suorituskyvyn pullonkauloja on helpompi välttää mikropalveluilla. Kuormantasaus helpottuu, kun vain pahimmaksi pullonkaulaksi muodostunut toiminto voidaan monistaa. Perinteiset monoliittiset sovellukset täytyy sen sijaan monistaa kokonaan ja ohjata uudet käyttäjät vähiten kuormitetulle palvelimelle kuormantasaimen avulla. (Hämäläinen 2016.)

2.2.3 Infrastruktuuri koodina

Infrastruktuuri koodina tai toisin sanoen ohjelmoitava infrastruktuuri tarkoittaa, että infrastruktuurin manuaalisen määrittämisen sijasta voidaan kirjoittaa skriptejä sen tekemistä varten. Käytännössä IT-infrastruktuuria voidaan siis hallinnoida konfigurointitiedostojen avulla. Koska se on vain tekstiä, sitä on helppo muokata, kopioida ja jakaa. Konfigurointitiedostot voidaankin sisällyttää versionhallintaan kuten mikä tahansa muu lähdekooditiedosto. Konfiguraation/kokoonpanon manuaalinen hallinta on kuulunut perinteisesti järjestelmäasiantuntijoiden tehtäviin, mutta ohjelmoitava infrastruktuuri, joka on ikään kuin automatisoitua konfiguraation/kokoonpanon hallintaa, kuuluu yleensä sovelluskehittäjälle. Näin kehittäjä voi sovelluskoodin lisäksi kirjoittaa myös infrastruktuurin, jolla koodi toimii. (Riley 2014; Schultz 2019.)

Schultzin (2019) mukaan ohjelmoitava infrastruktuuri säästää aikaa ja rahaa, koska fyysisen infrastruktuurin ylläpidosta ei tarvitse maksaa palkkaa useille ammattilaisille, suunnittelijoista laitteistojen huoltoteknikoihin. Lisäksi infrastruktuurin toteuttaminen koodina on nopeampaa, johdonmukaisempaa ja tehokkaampaa, koska manuaaliset, virheille alttiit työvaiheet jäävät minimiin. Kun konfiguroinnit ovat tiedostoissa, niitä voidaan testata ja siten palvelimien virheitä ja epäjohdonmukaisuuksia voidaan tarkistaa. Korjatut konfiguroinnit on myös helppo kopioida ja ottaa käyttöön tuotantoympäristössä. (Schultz 2019.)

2.2.4 Jatkuva monitorointi

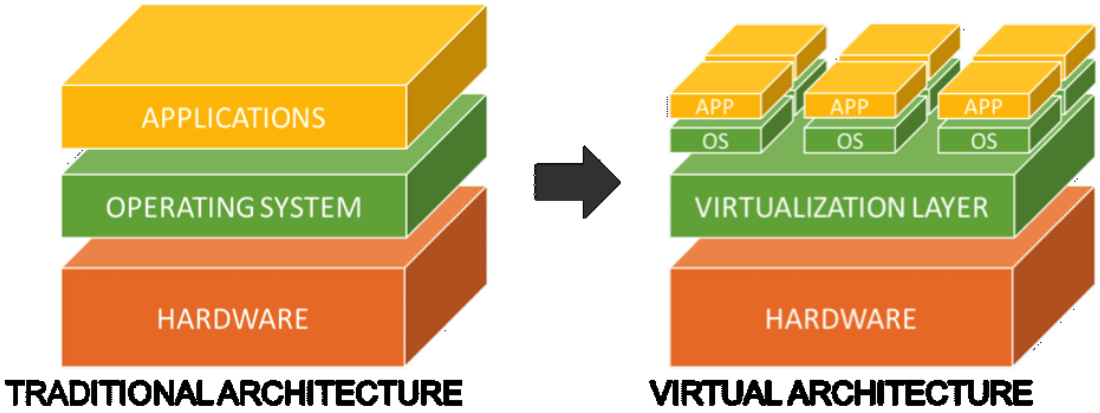
Sovellusten ja infrastruktuurin suorituskyky vaikuttaa loppukäyttäjän kokemukseen. Jatkuvalla monitoroinnilla koko sovelluspinon suorituskykyyn ja terveyteen saadaan täydellinen ja reaaliaikainen näkyvyys sovelluksen taustalla olevasta infrastruktuurista ohjelmiston komponentteihin asti. Tämä näkyvyys koostuu telemetrian ja metatietojen keräämisestä sekä järjestelmän ylläpitäjän huomiota vaativien hälytysten asettamisesta ennalta määritettyihin olosuhteisiin. Telemetria käsittää tapahtumien tiedot ja lokit, jotka on kerätty järjestelmän eri osista ja tallennettu analysoitavaksi. Monitoroinnin avulla saaduilla tiedoilla seurataan muutosten ja päivityksien vaikutuksia käyttäjiin. Tietoa kerätään myös ongelmien ja muutosten perussyiden selvittämiseen. (What is DevOps? N.d.a; What is DevOps? N.d.b.)

Jatkuvasta monitoroinnista tulee yhä tärkeämpää, kun sovellusten ja infrastruktuurien päivitystaajuudet kasvavat ja palvelujen on oltava käytettävissä ympäri vuorokauden. Hälytysten luominen tai näiden tietojen reaaliaikainen analysointi auttaa organisaatioita myös aktiivisemmin seuraamaan palveluitaan. Suorituskykyiset DevOps-tiimit varmistavat, että ne asettavat toimivia, merkityksellisiä hälytyksiä ja keräävät monipuolista telemetriaa, jotta he saavat oikean käsityksen palvelun tilasta. Nämä oivallukset auttavat ryhmää lievittämään ongelmia reaaliajassa ja näkemään, kuinka sovellusta voi parantaa tulevissa kehityssykleissä. (What is DevOps? N.d.a; What is DevOps? N.d.b.)

2.3 DevOps-toimintamallia tukevat teknologiat

2.3.1 Virtualisointi

Kuten aiemmin luvussa 2.1 todettiin, virtualisoinnilla voidaan vastata muuttuviin vaatimuksiin ohjelmistokehityksessä infrastruktuuritasolla. Virtualisoinnissa yhden tietokoneen laitteistoa kuten prosessoria, muistia ja tallennustilaa, voidaan jakaa useisiin virtuaalisiin tietokoneisiin ohjelmiston avulla, kuten kuviossa 4 on havainnollistettu. Jokainen virtuaalikone käyttää osaa taustalla olevaa tietokonelaitteistoa, mutta sillä on oma käyttöjärjestelmä ja se käyttäytyy kuin täysin itsenäinen tietokone. Virtualisointi on vakiokäytäntö nykyajan yritysten IT-arkkitehtuurissa. Virtualisoinnin avulla voidaan hyödyntää fyysinen tietokonelaitteisto tehokkaasti, ja se mahdollistaa myös organisaatioiden laitteistoinvestoinneille suuremman tuoton. (Virtualization 2019.)

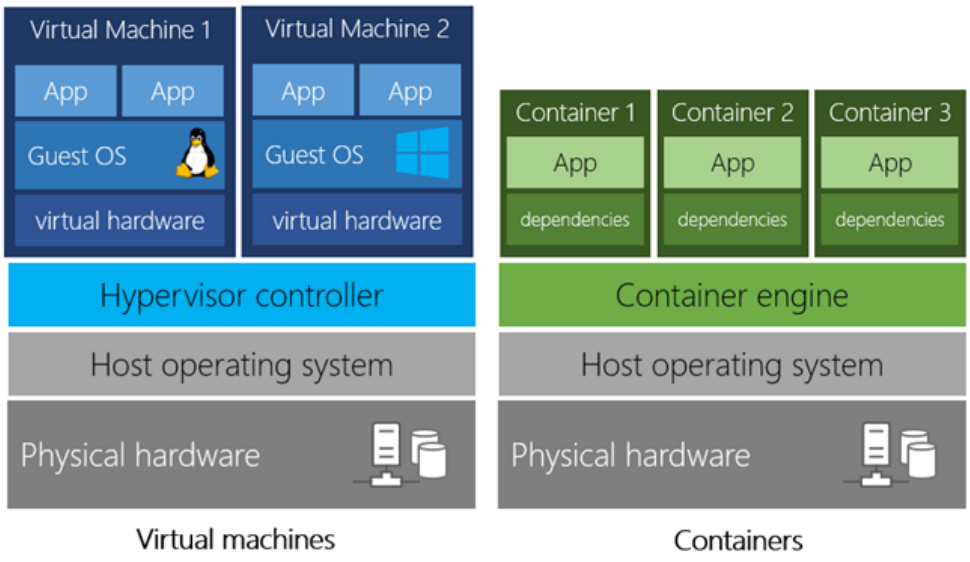


Kuvio 4. Virtualisointi (11 Points to Consider When Virtualizing Security 2018.)

Virtualisoinnilla on merkittävä rooli DevOps-toimintamallin toteuttamisessa, kun tavoitteena on lisätä ohjelmistokehitysprosessin nopeutta ja laatua. Virtualisoinnin avulla voi samanaikaisesti kehittää ja testata simuloiduissa ympäristöissä, jotka vastaavat kuluttajien käytettävissä olevaa tuotantoympäristöä. Tämä mahdollistaa kehityksen tapahtuvan reaaliaikaisen testauksen rinnalla, jolloin muutokset vaikuttavat koko järjestelmään. Tämä tarkkuustaso testauksessa vähentää huomattavasti käyttöönottoaikoja ja parantaa vakautta.

2.3.2 Kontitus

Kontitus (Containerization) on suosittu työkalu DevOps-tiimeissä. Sen avulla virtualisointi viedään askeleen pidemmälle. Kontitus hyödyntää taustalla olevia käyttöjärjestelmäytimiä sovellusten ajamiseen kontissa (Container), joka sisältää digitaalisen kokoonpanon koko sovelluksen vaatimasta ympäristöstä. (Watts n.d.) Kontit tarjoavat siis yhdenmukaisen, eristetyn suoritusympäristön sovelluksille. Ne poikkeavat virtuaalikoneista siten, että ne eivät edellytä omaa käyttöjärjestelmää palvelimen käyttöjärjestelmän lisäksi. Sen sijaan sovellus ja kaikki sen riippuvuudet pakataan "konttiin" ja ajonaikaista ympäristöä käytetään sovelluksen suorittamiseen. (What is cloud computing? N.d.). Kuviossa 5 on esitetty, miten kontitettujen sovellusten arkkitehtuuri eroaa virtuaalikoneelle asennetun sovelluksen arkkitehtuurista.



Kuvio 5. Konttien ja virtuaalikoneiden arkkitehtuurit (What is cloud computing? N.d.)

Rudnäsin (2018) mukaan Docker-ohjelmaa pidetään kontituksen merkittävimpänä luojana. Dockeriin on luotu ominaisuuksia, joiden avulla voidaan sammuneen tai kaatuneen kontin tilalle käynnistää täysin samanlainen kontti niin, ettei ohjelmien käytössä huomaa katkoa. Nopean käynnistämisen, kloonauksen ja sammuttamisen lisäksi kontituksella ja Dockerilla on myös muita etuja, joihin kuuluvat mm. liikuteltavuus ja skaalautuvuus. Kontteja voidaan siirrellä ja jakaa muille käyttäjille, ja ne toimivat käyttöjärjestelmästä, esiasennetuista ohjelmista ja työkaluista riippumatta, kunhan Docker-ohjelma on asennettu. Skaalautuvuuden voi toteuttaa esim. Dockeriin saatavalla Swarm-lisäosalla, joka mahdollistaa kuormantasauksen. Docker Swarm kloonaa ja sammuttaa kontteja tarpeen vaatiessa, joten käyttökokemus pysyy sulavana, eikä ylimääräisiä resursseja käytetä turhaan.

2.3.3 Pilvilaskenta ja pilvipalvelut

Pilvipalveluiden tarjoajat kuten Microsoft Azure, Amazon Web Services ja Google Cloud Platform tarjoavat internetin yli virtuaalisia resursseja kuten pilvilaskentaa (Cloud computing) palveluna. Pilvipalvelut ovat dynaamisesti skaalautuvia ja käytön mukaan laskutettavia ja niiden etuna on se, ettei palvelun ostajan tarvitse itse käyttää ajallisia tai taloudellisia resursseja palvelinkoneiden ylläpitoon. Tähän opinnäytetyöhön on pilvipalvelualustaksi valittu Microsoft Azure, josta kerrotaan enemmän luvussa 3.

Pilvipalveluiden tavoitteena on tehdä kaikenkokoisten yritysten liiketoiminnasta helpompaa ja tehokkaampaa. Jokainen yritys on ainutlaatuinen ja pilvipalveluntarjoajat tarjoavat laajan valikoiman palveluita erilaisten tarpeiden täyttämiseksi. Pilvilaskenta on kustannustehokasta ja joustavaa, joten kaikenkokoiset yritykset voivat hyötyä siitä. (What is cloud computing? N.d.) Kulutukseen pohjautuvan hinnoittelumallin ansiosta kuluttaja voi maksaa vain niistä resursseista, joita tarvitsee. Tämä mahdollistaa myös paremman kustannusennusteen. Koska yksittäisten resurssien ja palveluiden hinnat ovat saatavilla, voidaan odotetun käytön perusteella ennustaa myös kustannukset. Lisäksi pilvipalvelut ovat yleensä halvempia, luotettavampia ja joustavampia kuin paikalliset palvelimet, koska niiden kanssa laitteiden huollosta, varkauksista tai vaurioista johtuvia seisokkeja ei ole ollenkaan (Introduction to cloud computing n.d.).

2.4 DevOps-toimintamallin edut ja haitat

Vuodesta 2014 alkaen julkaistut ”Accelerate: State of DevOps” -raportit osoittavat suoran yhteyden organisaation suorituskyvyn ja ohjelmistojen toimituksen suorituskyvyn välillä. Raportit perustuvat kyselyihin, joihin kuuden vuoden aikana on vastannut yli 31 000 ammattilaista maailmanlaajuisesti. Raporteissa ala jaetaan alhaisen, keskitason, korkean ja eliittitasoisen suorituskyvyn toimijoihin, neljän avainmittarin avulla: läpimenoaika, käyttöönoton taajuus, keskimääräinen viasta palautumisaika (Mean Time To Restore) ja muutosten epäonnistumisprosentti. (Four key metrics n.d.; Forsgren, Frazelle, Humble & Smith 2019.)

Viimeisin raportti “Accelerate: State of DevOps 2019” osoittaa, että DevOpsin edistämät ohjelmiston nopeus, vakaus ja saatavuus edistävät organisaation suorituskykyä kannattavuuden, tuottavuuden ja asiakastyytyväisyyden osalta. Korkean suorituskyvyn toimijat yltävät organisaation suoritustavoitteisiin, tai jopa ylittävät ne, kaksi kertaa todennäköisemmin. Yksi havainnoista oli, että pilviteknologian hyödyntäminen edistää korkeaa suorituskykyä ja on yksi avaintekijä eliittitason toimijoilla. Lisäksi aikaisempien raporttien havainnot siitä, että vakauden optimointi nopeudesta tinkimättä on mahdollista, vahvistuivat. Esimerkiksi muutoskomitean (CAB, Change Advisory Board) kaltaiset raskaat prosessit muutosten hyväksymiseksi vaikuttavat nopeuteen ja vakauteen negatiivisesti. (Forsgren, Frazelle, Humble & Smith 2019.)

Vaikka DevOps on ns. päivän sana ja varmasti monelle organisaatiolle loistava tapa optimoida ohjelmistojen suunnittelua ja toimitusta, ei sitä voida kuitenkaan pitää täydellisenä innovaationa. DevOps-mallin mukaan jokaisella ohjelmistoa toimittavan ryhmän jäsenellä on tiukemmin määritelty rooli. Tästä voi olla haittaa organisaatioille, jos heidän on palkattava ja johdettava enemmän henkilöstöä sen sijaan, että sijoittaisi pienempään määrään ns. *full stack* -kehittäjiä. Yksi DevOpsin tärkeimmistä eduista on, että se edistää jatkuvaa ohjelmistotoimitusta, ja siten uusia ominaisuuksia voidaan ottaa käyttöön nopeammin. Siinä piilee kuitenkin riski, että toimintoja lisätään vain siksi, että voidaan eikä siksi, että niitä tarvitaan. Toimintojen lisääminen ei aina ole paras vaihtoehto. Lisäksi DevOpsin hyödyntäminen vaatii useimmiten or-

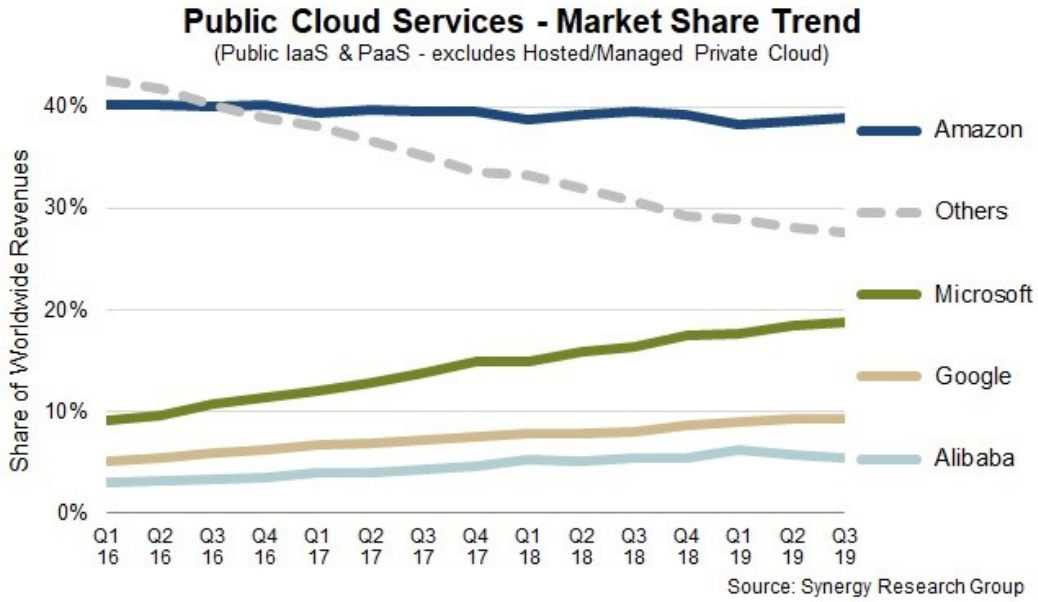
ganisaatiolta infrastruktuurin muutoksia, ainakin osittain. Esimerkiksi sovelluksia voidaan joutua siirtämään virtuaalikoneilta kontteihin. Infrastruktuurin muutokset voivat olla monimutkaisia ja kalliitakin ponnistuksia, ja jos niitä ei tehdä oikein, ei muistakaan DevOps-käytännöistä saada välttämättä niistä odotettua hyötyä. (Tozzi 2016.)

3 Microsoft Azure

3.1 Yleistä

Microsoft Azure on alusta pilvipohjaisille tietojenkäsittelypalveluille, joita Microsoft tarjoaa yrityksille, kehittäjille ja kaikille, jotka haluavat rakentaa sovelluksen tai johtaa yritystä internetissä ilman omaa palvelinlaitteistoa tai -ohjelmistoa. Azurea voidaan mm. käyttää alustana sekä virtuaalipalvelimille (Infrastructure as a Service) että kehittäjille tarkoitettuna kehitysalustana (Platform as a Service). Erilaisia palveluita on paljon ja niiden kirjo laajenee jatkuvasti. Azure tarjoaa työkaluja DevOpsin lisäksi mm. liiketoiminta-analyysi- ja IoT-ratkaisuihin. Azure tukee avoimen lähdekoodin tekniikoita, joten lähes mikä tahansa sovellus toimii alustalla, työkaluista ja tekniikoista ja käyttöjärjestelmästä riippumatta. (What is Azure? N.d.)

Julkisen IaaS- ja PaaS-pilvipalveluiden maailmanlaajuiset kokonaiskulut olivat vuoden 2019 kolmannella vuosineljänneksellä 20 miljardia dollaria, mikä on yli 80% pilvipalveluiden kokonaismäärästä. Loput markkinat koostuvat yksityisistä pilvipalveluista, joissa IBM on markkinajohtaja. (Amazon, Microsoft, Google and Alibaba Strengthen their Grip on the Public Cloud Market 2019.). Kuten Kuviosta 6 selviää, Microsoftin markkinaosuus julkisten pilvipalveluiden tarjoajana on vielä selvästi Amazonia pienempi, mutta kasvu on ollut kilpailijoita suurempaa.



Kuvio 6. Julkisten pilvipalveluntarjoajien markkinaosuudet (Amazon, Microsoft, Google and Alibaba Strengthen their Grip on the Public Cloud Market 2019.)

3.2 Azure App Service

Azure App Service on HTTP-pohjainen palvelu verkkosovellusten, REST-sovellusliittymien ja mobiilisovellusten ylläpitämiseen. Sovellukset toimivat ja skaalautuvat sekä Windows- että Linux-pohjaisissa ympäristöissä. ”Azure App Service” -sovelluspalvelussa voi myös hyödyntää DevOps-ominaisuuksia, kuten jatkuvaa integraatiota esimerkiksi Azure DevOps, GitHub ja Docker Hub lähteistä. (App Service overview 2020.)

Sovelluspalvelun käyttämistä Azure-laskentaresursseista maksetaan sovelluspalvelusuunnitelman (App Service Plan) avulla. Yksi tai useampi sovellus voidaan määrittää toimimaan samoilla laskentaresursseilla, samassa sovelluspalvelusuunnitelmassa. Sovelluspalvelusuunnitelma määrittelee seuraavat resurssit web-sovelluksen ajamista varten:

- Alue (esim. Pohjois-Eurooppa)
- Virtuaalikoneiden määrä
- Virtuaalikoneiden koko/suorituskyky (pieni, keskikokoinen, suuri)
- Hinnoittelutaso (Free, Shared, Basic, Standard, Premium, PremiumV2, Isolated)

Sovelluspalvelusuunnitelman hinnoittelutaso määrittelee, mitä sovelluspalvelun ominaisuuksia saadaan käyttöön ja kuinka paljon suunnitelmasta maksetaan. Hinnoittelutasoilla on seuraavat luokat, joita voidaan vaihtaa tarpeen mukaan milloin tahansa:

Jaettu laskenta: Free ja Shared-tasot ajavat sovelluksen samassa virtuaalikoneessa, kuin muut App Service -sovellukset, mukaan lukien muiden asiakkaiden sovellukset. Nämä tasot jakavat CPU-kiintiöt jokaiselle sovellukselle, joka käyttää jaettuja resursseja, eikä resursseja voida skaalata.

Omistettu laskenta: Basic-, Standard-, Premium- ja PremiumV2-tasot ajavat sovelluksia omistetuissa virtuaalikoneissa. Vain samassa sovelluspalvelusuunnitelmassa olevilla sovelluksilla on samat laskentaresurssit. Mitä korkeampi taso, sitä enemmän virtuaalikoneita on käytettävissä skaalattavaksi.

Eristetty: Isolated-taso ajaa omistettujen virtuaalikoneiden sovelluksia yksityisessä Azure-verkossa (Azure Virtual Network). Se tarjoaa verkon eristyksen sovellusten laskennan eristämisen lisäksi ja tarjoaa myös suurimman mahdollisuuden skaalata. (Azure App Service plan overview 2017.)

3.3 Azure DevOps

3.3.1 Yleistä

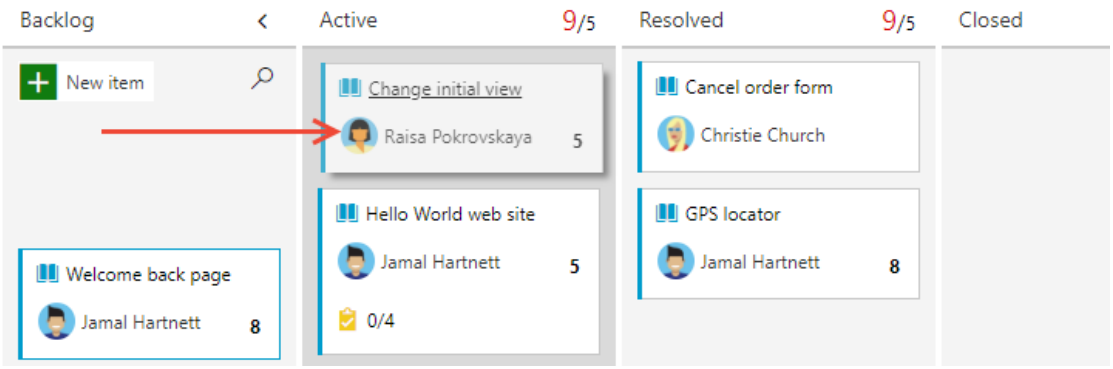
Azure DevOps tarjoaa kehittäjille lukuisia palveluita helpottaakseen DevOps-toimintamallin toteutusta. Palvelut tukevat kehitystiimejä työn suunnittelussa, sekä sovellusten rakentamisessa ja käyttöönotossa. Kehittäjät voivat työskennellä pilvessä Azure DevOps Services -palvelun avulla tai paikallisesti Azure DevOps Server -palvelun avulla. (What is Azure DevOps? 2019.)

Azure DevOps tarjoaa integroituja ominaisuuksia, joihin pääsee verkkoselaimen tai IDE-asiakasohjelman kautta. Azure DevOpsista voidaan käyttää yhtä tai useampaa seuraavista palveluista yrityksen tarpeiden perusteella:

- Azure Repos
- Azure Pipelines
- Azure Boards
- Azure Test Plans
- Azure Artifacts

3.3.2 Azure Boards

Azure Boards -verkkopalvelu on tarkoitettu ohjelmistoprojektien hallintaan. Se tarjoaa tuen mm. Scrumin ja Kanbanin kaltaisille projektinhallintamenetelmille, joilla voi seurata eri työkohteiden etenemistä taululla. Työkohteita voi määrittää työryhmän eri jäsenille, merkitä ja siirrellä taululla drag-and-drop -tyylillä. Työkohteissa on myös keskusteluosio, jossa voi mainita(@mentions) tai poimia jäsennumerolla(#ID), työryhmän jäseniä mukaan keskusteluun. Azure Boardsissa voi asettaa myös ilmoitukset aktiiviseksi, jolloin uuden työkohteen luomisesta tai muuttamisesta annetaan ilmoitus työryhmän jäsenille. Kuviossa 7 on esimerkkinäkymä Azure Boardsin Scrum-projektinhallintataulusta. (What is Azure Boards? 2020.)



Kuvio 7. Scrum-projektinhallintataulu (What is Azure Boards? 2020.)

3.3.3 Azure Pipelines

Azure Pipelines on pilvipalvelu, jonka avulla koodiprojektin voi luoda, testata automaattisesti ja asettaa muiden käyttäjien saataville. Azure Pipelines yhdistää jatkuvan integroinnin (CI) ja jatkuvan toimituksen (CD), joiden avulla voi testata ja rakentaa koodia jatkuvasti ja johdonmukaisesti, sekä lähettää sen haluttuun kohteeseen. (What is Azure Pipelines? 2019.)

Azure Pipelines -palvelussa voi käyttää useita sovellustyyppejä, kuten Java, JavaScript, Node.js, Python, .NET, C ++, Go, PHP ja XCode. Palvelun käyttö kuitenkin edellyttää, että lähdekoodi on tallennettu versionhallintajärjestelmään. Azure Pipelinesiin integroituvia versionhallintajärjestelmiä ovat GitHub, GitHub Enterprise, Azure Repos Git & TFVC, Bitbucket Cloud ja Subversion. Julkaisukohteet voivat puolestaan olla konttirekistereitä, virtuaalikoneita, Azure-palveluita tai mitä tahansa paikallisia tai pilvikohteita. (What is Azure Pipelines? 2019.)

3.3.4 Azure Repos

Azure Repos on joukko versionhallintatyökaluja, jotka ovat tarkoitettu koodin hallinnan avuksi. Versionhallintajärjestelmät ovat ohjelmistoja, joiden avulla voi seurata ajan myötä koodiin tehtyjä muutoksia. Kun koodin muutokset tallennetaan versionhallintajärjestelmään se ottaa tilannekuvan tiedostoista. Tallennetut tilannekuvat voidaan tarvittaessa myöhemmin palauttaa. Versiohallinnan avulla voi tallentaa työn ja koordinoida koodimuutokset koko työryhmän sisällä. Azure Repos tarjoaa kahdentyyppistä versionhallintaa, hajautettua Git- ja keskitettyä Team Foundation (TFVC)-versionhallintaa. (What is Azure Repos? N.d.)

Git on nykyään yleisimmin käytetty versionhallintajärjestelmä, josta on nopeasti tulossa versionhallinnan standardi. Git on hajautettu versionhallintajärjestelmä, mikä tarkoittaa, että paikallinen koodikopio on täysin toimiva versio ohjelmiston koodista. Nämä paikalliset koodikopiot tekevät hajautetusta versionhallinnasta helpon tavan työskennellä offline-tilassa tai etäyhteyden kautta. Koodimuutokset tehdään paikallisesti ja hakemistot synkronoidaan sitten palvelimella olevan version kanssa. Azure

Repos tukee myös Team Foundation Version Control (TFVC) -sovellusta. TFVC on keskitetty versionhallintajärjestelmä, jossa tyypillisesti työryhmän jäsenillä on vain yksi versio jokaisesta tiedostosta omalla koneellaan. Historiallista tietoa ylläpidetään vain palvelimella. (What is Azure Repos? N.d.)

3.3.5 Azure Test Plans

Azure Test Plans -palvelu tarjoaa kehittäjille helppokäyttöisen selainpohjaisen testinhallintaratkaisun, jossa on tarvittavat ominaisuudet suunnitellulle manuaaliselle testaukselle, käyttäjän hyväksymistestaukselle, tutkivalle testaukselle ja palautteen keräämiselle sidosryhmiltä. Palvelussa voi luoda, hallita ja valvoa erilaisia testejä ja kerätä diagnostiikkaa koko kehitysprosessin elinkaarelta. Palvelulla pystytään esimerkiksi luoda ja seurata jatkuvan integraation testien tulosta tai tehdä käyttäjätestejä tietyistä järjestelmän osista. Käyttäjätestin tekijät voidaan määrittää ja käyttäjät voivat seurata laatua Kanban-taulun korteista, joihin testatessa syntyneet virheet liitetään automaattisesti. Testien tuloksista voi myös luoda erilaisia graafeja laadun seuraamisen helpottamiseksi. (Exploratory and manual testing scenarios and capabilities 2019.)

3.3.6 Azure Artifacts

Azure Artifacts -palvelu on paketinhallintaratkaisu, jonka avulla voi sekä luoda että jakaa Maven-, npm- ja NuGet-pakettisyötteitä julkisista ja yksityisistä lähteistä. Palvelun avulla voidaan lisätä täysin integroitu paketinhallinta jatkuvan integraation tai jatkuvan toimituksen (CI/CD) menetelmiin. Azure Artifacts -palvelussa voi käyttää myös omia ohjelmistokomponentteja, jotka voi siirtää pakettiin ja jakaa oman tiimin sisällä. Tällöin samaa ohjelmiston osaa voidaan käyttää eri projekteissa eikä sitä tarvitse kirjoittaa jokaiseen projektiin erikseen. (What is Azure Artifacts? 2018.)

4 Case: BisLenz

4.1 Yleistä

Tridea Oy:n kehittämä ”BisLenz -johdon raportointitila” on web-palvelu, joka muodostaa käyttäjälleen kokonaiskuvaa asiakkaidensa asiakaskokemuksesta. Palvelussa tuodaan esille olennaisimmat mittarit, joilla asiakaskokemusta voidaan tarkkailla useasta näkökulmasta. (BisLenz n.d.)

BisLenz -johdon raportointitila koostuu teknisesti kahdesta pääsovelluksesta ja tietokannasta. Taustalla toimiva ns. palvelinpuolen (back-end) sovellus päivittää tietokantaa ja tarjoaa rajapinnan, josta ns. selainpuolen (front-end) sovellus voi hakea tarvitsemansa datan, joka käyttäjälle esitetään. Nykyisellään taustalla toimiva sovellus on toteutettu Node.js+Express tekniikoilla ja käyttäjälle näkyvä osuus on joukko Wordpress-julkaisujärjestelmän päällä toimivia pienempiä sovelluksia ja sivuja, jotka on toteutettu mm. React.js:lla ja visualisointityökaluilla kuten Google Data Studio. Selainpuolen toteutuksessa Wordpressistä ollaan luopumassa ja käyttöliittymää korvataan puhtaasti React-pohjaisella ratkaisulla.

Opinnäytetyön käytännön osuudessa otettiin huomioon toimeksiantajan tarve React-sovelluksen kehityksestä ja julkaisusta. Tavoitteena oli julkaista React.js kirjastolla toteutettu testisovellus Microsoft Azure -pilvialustalle. Sovelluksen kehityksessä ja julkaisussa hyödynnettiin DevOps -toimintamallia. Tarkoituksena oli esitellä toimeksiantajalle Microsoft Azure -pilvialustaa ja DevOps -toimintamallia sekä niiden yhdistämistä käytännössä.

4.2 Suunnittelu

4.2.1 Vaatimusten määrittely

Ensimmäisenä vaiheena projektin suunnittelussa oli vaatimusten määrittely, jossa selvitettiin tiettyjä reunaehtoja, joilla tutkimusta saadaan rajattua haluttuihin kohtei-

siin. Rajauksella pyrittiin löytämään toimeksiantajalle hyödyllisintä tietoa React-sovelluksen julkaisusta Microsoft Azure -pilvialustalle. Vaatimukset keskittyivät DevOps-toimintamallin käyttämiseen ja BisLenz -palvelun siirtämisen soveltuvuuteen Microsoft Azure -pilvialustalle. Tuloksena vaatimusten määrittelyssä syntyi seuraavia ehtoja:

- Microsoft Azure -pilvialustalle tulee julkaista testisovellus
- Sovellus tulee toteuttaa toimeksiantajalle tutulla React.js kirjastolla
- Sovelluksen tulee sisältää komponentteja BisLenz -palvelusta, joilla todetaan tiedonkulku tietokannasta
- Sovelluksen kehityksessä ja julkaisussa tulee hyödyntää DevOps-toimintamallia vähintään jatkuvan integroinnin ja julkaisun osalta
- DevOps-toimintamallin toteuttamiseen tulee käyttää Azuren palveluita

4.2.2 Ratkaisut

Vaatimusten perusteella seuraavaksi etsittiin käytännön toteutukselle sopivat ratkaisut. Sovelluksen julkaisu päätettiin tehdä nimenomaan verkkosovellusten ylläpitoon tarkoitetun, Azure sovelluspalvelun (Azure App Service) avulla. Sovelluspalvelun käyttö edellyttää myös sovelluspalvelusuunnitelman (Azure App Service plan) käyttöönottoa ja määrittelyä. Tähän toteutukseen sovelluspalvelun hinnoittelutasoksi valittiin Free (ilmainen). Ilmaisessa hinnoittelutasossa sovellus julkaistaan Azuren virtuaalikoneelle, jonka käyttämät laskentaresurssit on jaettu myös muille samaa hinnoittelutasoa käyttävien käyttäjien kanssa (Azure App Service plan overview 2017).

Sovelluksen kehityksessä ja julkaisussa käytettin DevOps-toimintamallia ja sen toteutukseen käytettiin Azure DevOps -palvelujoukkoa. Azure DevOps -palveluiden perustason lisenssi on ilmainen ensimmäisille viidelle käyttäjälle, mutta se ei sisällä Azure Test Plans -palvelua (Pricing for Azure DevOps N.d). Tällä lisenssillä voitiin toteuttaa vaaditut jatkuvan integroinnin ja jatkuvan julkaisun menetelmät Azure Pipelines -palvelua hyödyntäen. Projektin versionhallinta toteutettiin myös Azure DevOps -palvelujoukkoon kuuluvan, Git-versionhallintajärjestelmää hyödyntävän Azure Repos -palvelun avulla.

DevOps-toimintamallin laajempaa käyttöä varten sovelluksen jatkuvaan monitorointiin päätettiin ottaa käyttöön palvelu myös Azure DevOps -palvelujoukon ulkopuolelta. Tähän tarkoitukseen valittiin Application Insights, joka on maksullisen Azure Monitor -palvelun yksi ominaisuus. Application Insights on kehittäjille ja DevOps-ammattilaisille tarkoitettu sovelluksen suorituskyvyn hallinta (APM) -palvelu.

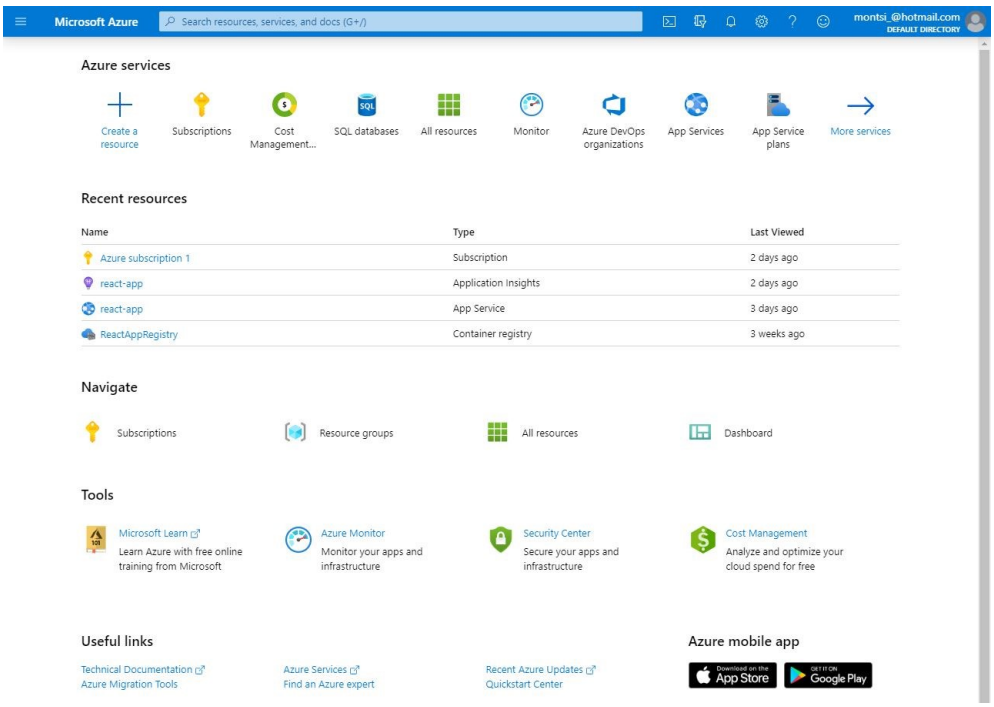
Testisovellukseksi luotiin React-sovellus, joka jäljittelee BisLenz -palvelun markkinointiosion käyttöliittymää. Sovellukseen sisällytettiin kaksi BisLenz -palvelun komponenttia, jotka hakevat markkinointidataa BisLenzin tietokannasta ja esittävät sen käyttäjälle visuaalisesti. Koko palvelua ei siis pyritty siirtämään Azuren pilvialustalle vaan testisovelluksella oli tarkoitus luoda pohjaa DevOps-toimintamallin hyödyntämiselle BisLenz -palvelun uutta, Reactilla kehitettävää käyttöliittymää varten.

4.3 Toteutus

4.3.1 Ympäristön käyttöönotto

Sovelluksen julkaisu Microsoft Azure -pilvialustalle edellyttää Azure-tilin luomista. Tilin luomiseen voi käyttää myös jo olemassa olevia Microsoft-tilejä. Azure-tilin luominen on ilmaista ja uusi käyttäjä saa käyttää lukuisia palveluita ilmaiseksi 12:n kuukauden ajan. Suurin osa Azuren palveluista on käytön mukaan laskutettavia.

Sovelluksen julkaisua varten Azuren portaalissa tuli ottaa käyttöön uusi sovelluspalvelusuunnitelma (App Service plan) ja luoda uusi sovelluspalvelu (App Service). Microsoft Azure -portaalin yleisnäkymä on esitetty Kuviossa 8. Portaalin kautta hallitaan kirjautuneen käyttäjän Azure -palveluita ja resursseja.



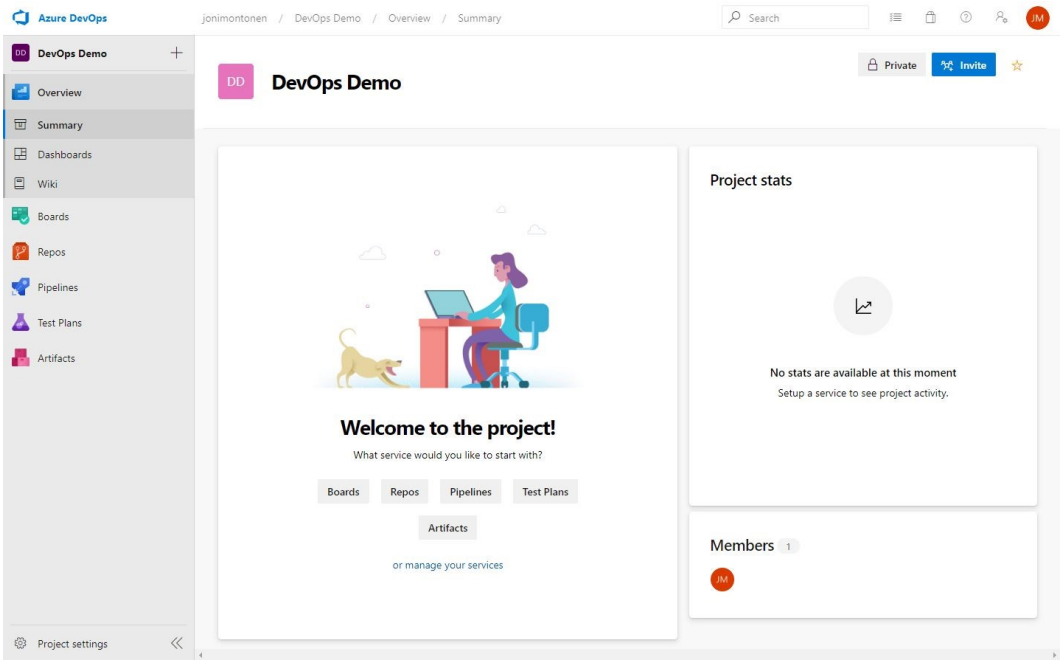
Kuvio 8. Azure portaalin yleisnäkymä

Kun uusi sovelluspalvelusuunnitelma otetaan käyttöön, tulee sille nimetä resurssiryhmä, jonka voi myös luoda tässä vaiheessa. Lisäksi valitaan nimi, käyttöjärjestelmä, alue ja hinnoittelutaso. Uudelle sovelluspalvelulle valitaan puolestaan resurssiryhmän, nimen ja alueen lisäksi suorituksenaikainen pino (Runtime Stack), joksi tässä tapauksessa valittiin uusin Node-versio React-sovellusta silmällä pitäen. Lisäksi sovelluspalvelulle valittiin aikaisemmin luotu sovelluspalvelusuunnitelma ja ”Monitoring” välilehdeltä kytkettiin Application Insights -palvelu käyttöön. Kun em. palvelut ovat otettu käyttöön, niistä syntyy Azure portaaliin resurssit, joita voidaan tarkastella portaalin kautta.

4.3.2 Azure DevOps projektin luominen

Azure DevOps projektin luominen aloitettiin luomalla uusi organisaatio Azuren portaalissa. Organisaatio luodaan hyväksymällä Microsoft Azure palvelun ehdot, tietosuojalauseke ja käytänteet, nimeämällä organisaatio ja valitsemalla maanosa, jonka palvelimille projekti isännöidään.

Seuraavaksi Azure DevOps -portaalissa luotiin uusi projekti, joka käyttää Git versionhallintaa. Azure DevOps projektin yleisnäkymä on esitetty Kuviossa 9. Projektin versionhallinnan käyttöönottoon on monta tapaa ja vaihtoehdot löytyvät ”Repos” välilehden kautta. Tässä projektissa luotiin portaalin kautta ensin README.md tiedosto, johon voidaan lisätä yleistä tietoa sovelluksesta ja sen versionhallinnasta. Projektin koodia muokattiin paikallisesti kloonaamalla repositorio haluttuun hakemistoon omalle tietokoneelle. Tämän jälkeen projektiin lisättiin valmis malli React.js -sovelluksesta Node-moduulin ”create-react-app” avulla.

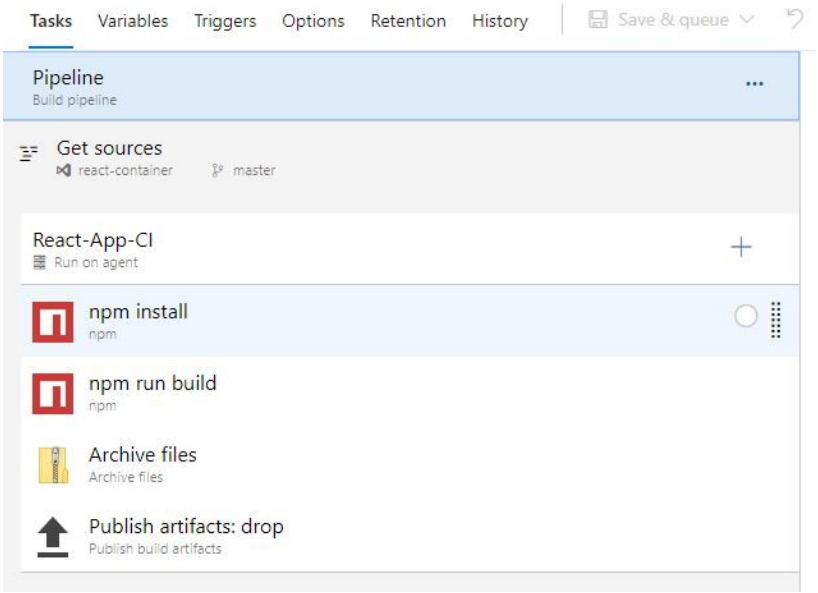


Kuvio 9. Azure DevOps projektin yleisnäkymä

4.3.3 Sovelluksen jatkuvat integroinnit ja julkaisut

Projektin jatkuvan integroinnin ja jatkuvan julkaisun vaiheet määriteltiin Azure DevOps -portaalissa ”Pipelines” välilehden kautta. Eri vaiheiden määritykset aloitettiin lisäämällä uusi liukuhihnoitus (New Pipeline). Azure tarjoaa määritykseen myös vanhan käyttöliittymän käyttämistä, mikä sisältää ainakin toistaiseksi hieman enemmän vaihtoehtoja ja tarkempaa määrittelyä. Valittavissa on myös erilaisia määrittelyn malleja tietyntyyppisille sovelluksille.

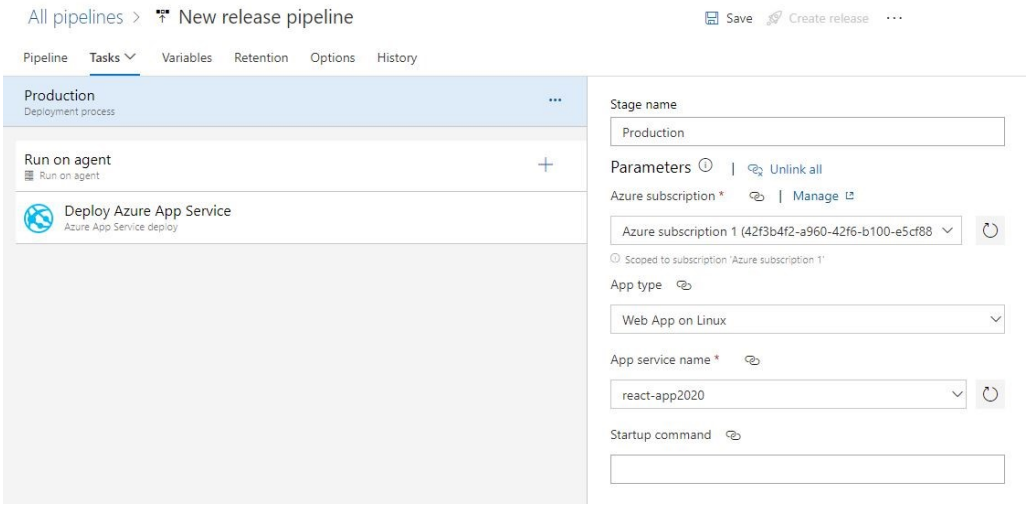
Tähän sovellukseen valittiin lähteeksi Azure Repos Git ja malliksi Node.js With Grunt, mutta mikä tahansa Node.js pohjainen malli sopii hyvin React-sovelluksille ja ylimääräisiä liukuhihnoituksen työvaiheita voi poistaa helposti portaalissa. Tässä tapauksessa Grunt työvaihe koodin kokoamiseen poistettiin käytöstä ja kokoaminen toteutettiin react-scriptin avulla uudessa työvaiheessa. ”Triggers” välilehdeltä otettiin jatkuva integraatio käyttöön (Enable continuous integration). Oletuksena liukuhihnoitus koskee kaikkia valitun repositorion master-haaraan tehtyjä muutoksia, mutta esimerkiksi tarkemman hakemiston määrittäminen onnistuu myös. Kun version muutos tapahtuu, liukuhihnoitus tekee Kuviossa 10 esitetyt jatkuvan integroinnin vaiheet, jotka sisältävät sovelluksen riippuvuuksien asentamisen, koontiversion luomisen ja pakkaamisen julkaisukelpoiseksi artefaktiksi (Artifact).



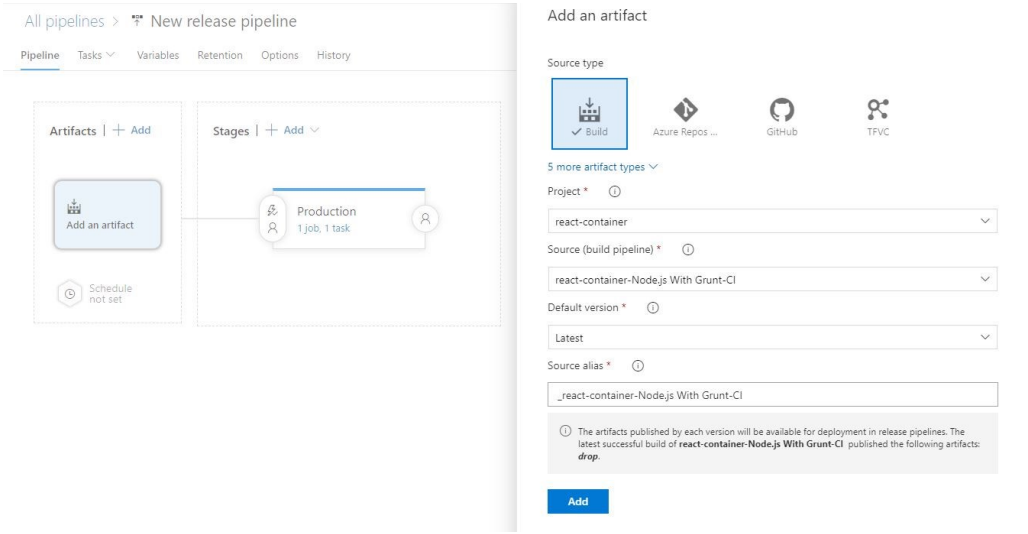
Kuvio 10. Jatkuvan integraation vaiheet

Jatkuva julkaisu määriteltiin Azure DevOps -portaalissa ”Pipelines” välilehden ”Releases” kohdasta, missä lisättiin uusi liukuhihnoitus, joka jatkuu jatkuvasta integroinnista julkaisuun. Myös julkaisuun löytyy valmiita malleja, joista tälle sovellukselle valittiin ”Azure App Service Deployment”. Tämän mallin määrityksiin tulee ”Tasks” välilehdeltä tarkentaa, mitä Azure-tiliä (Azure subscription) ja sovelluspalvelua (App ser-

vice name) julkaisussa käytetään, kuten Kuviossa 11 on esitetty. ”Pipeline” välilehdeltä julkaistavaksi artefaktiksi valittiin jatkuvan integraation työvaiheissa luotu artefakti (Ks. Kuvio 12). Seuraavaksi jatkuva toimitus kytkettiin käyttöön (Continuous deployment trigger) artefaktin päälle ilmestyneen salamakuvioisen painikkeen kautta.



Kuvio 11. Parametrit jatkuvaan toimitukseen

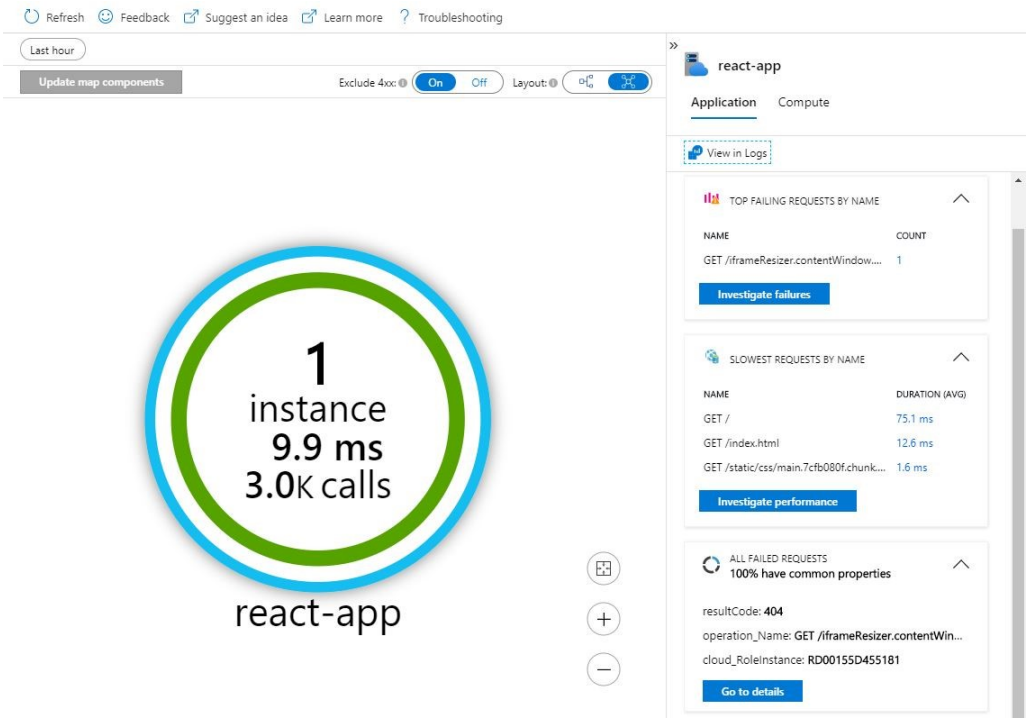


Kuvio 12. Julkaistavan artefaktin valinta

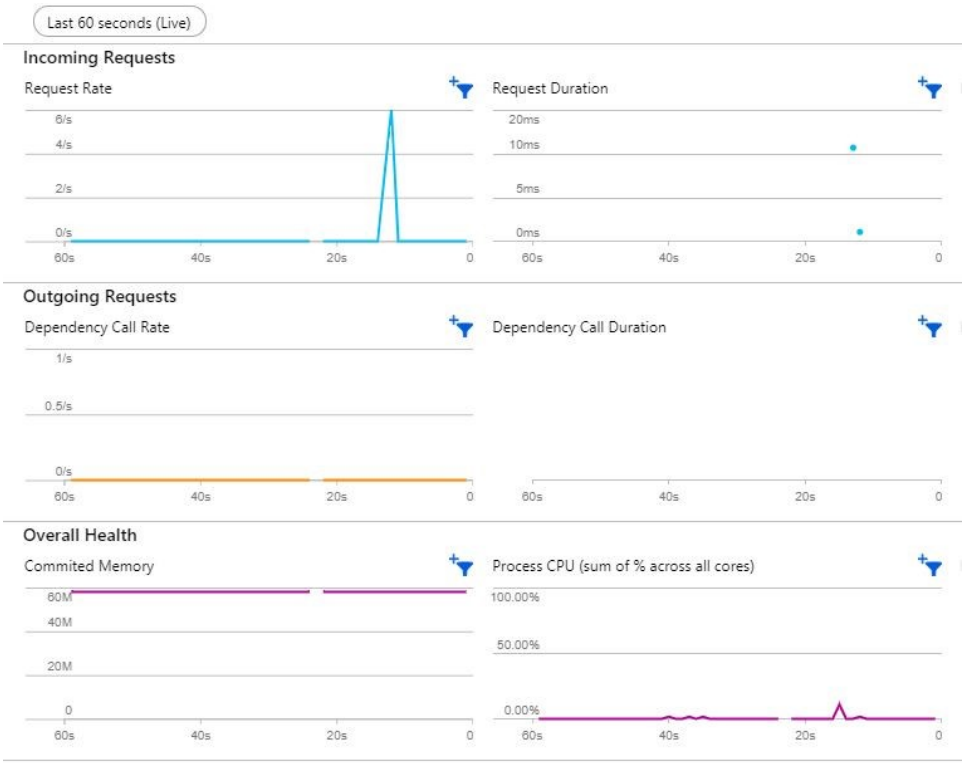
4.3.4 Sovelluksen jatkuva monitorointi

React-sovelluksen jatkuva monitorointi Application Insights -palvelulla edellyttää Liitteessä 1 esitetyn skriptin sisällyttämistä html-tiedoston <head> -osioon. Palvelun voi vaihtoehtoisesti ottaa käyttöön myös NPM-asennusohjelman avulla, jolloin skripti asennetaan riippuvuudeksi sovellukseen. Tähän toteutukseen valittiin ensimmäinen vaihtoehto.

Application Insights kerää tietoja sovelluksen suorituskyvystä ja käytöstä. Palvelu piirtää kartan sovelluksen yhteyksistä ja sivujen latausajoista, joita voidaan Azure -portaalin kautta tarkastella Application Insights -resurssin ”Application map” välilehdeltä (Ks. Kuvio 13). Palvelulla voidaan myös seurata reaaliaikaisesti mm. sivunlatauksia, AJAX-pyyntöjen määrää, yksityiskohtia ja virheitä tai istuntojen määrää. Näitä tietoja voidaan jakaa sivun, asiakaskäyttöjärjestelmän ja selainversion, maantieteellisen sijainnin ja muiden ulottuvuuksien perusteella. Kuviossa 14 on esimerkki Application Insights -resurssin ”Live Metrics” välilehdeltä.



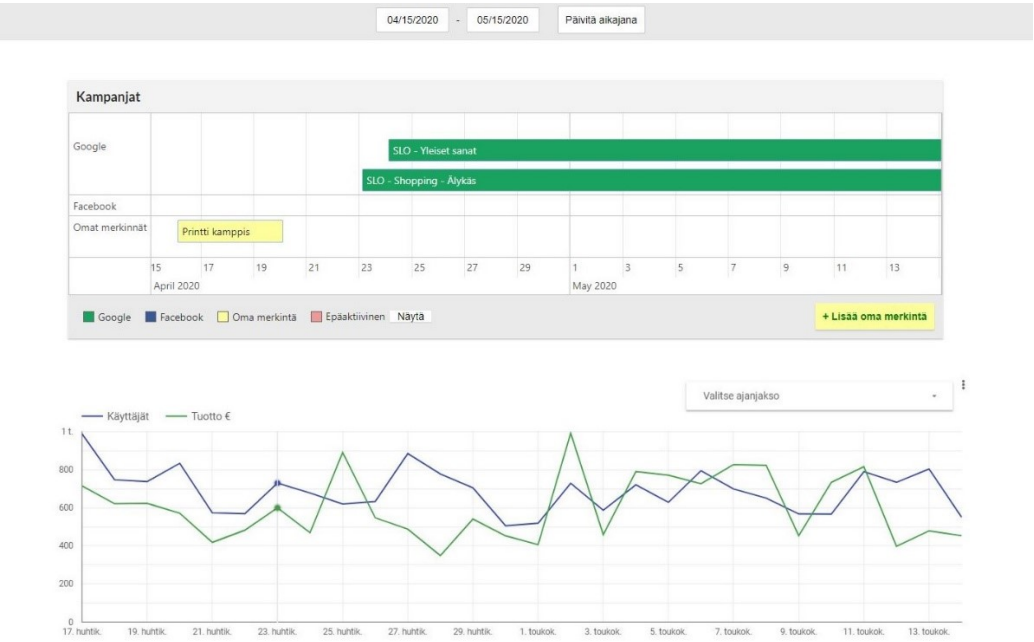
Kuvio 13. Application map



Kuvio 14. Live Metrics

4.3.5 Sovelluksen käyttöliittymä

Sovelluksen toteutuksessa sen käyttöliittymään lisättiin BisLenz -palvelussa käytettäviä visuaalisia komponentteja, joilla markkinointidataa esitetään käyttäjälle. Azuren sovelluspalveluun julkaistun testisovelluksen käyttöliittymä on esitetty Kuviossa 15. Testisovellus hyödyntää BisLenzin nykyistä palvelinpuolen sovellusta tiedon hakemiseksi. Markkinointidata haetaan HTTP-pyynnöillä palvelinsovellukselta, joka toimii mm. rajapintana BisLenzin tietokannalle. Esimerkiksi palvelimen palomuurimäärityksiä varten on hyvä tietää Azuren sovelluspalveluun julkaistun sovelluksen IP-osoite. Koska testisovellus julkaistiin ilmaisella hinnoittelutasolla, se jakaa verkkoinfrastruktuurin muiden sovellusten kanssa. Tämän vuoksi myös sen IP-osoitteet voivat vaihdella. Sovelluksen kaikkia mahdollisia IP-osoitteita voidaan tarkastella Azure-portaalista sovelluspalveluresurssin ”Properties” välilehdeltä.



Kuvio 15. Sovelluksen käyttöliittymä

Ensimmäinen komponentti on avoimen lähdekoodin vis.js-kirjastoon perustuva komponentti, jolla visualisoidaan käyttäjän sosiaalisen median mainoskampanjat ja itse lisätyt markkinointitapahtumat aikajanalle. Testisovelluksessa tämä komponentti näyttää vain yhden käyttäjän dataa, eikä dynaamista käyttäjänvalintaa ole toteutettu. Toisena komponenttina on samalle käyttäjälle luotu Google Data Studio-raportti, joka on upotettu sovellukseen <iframe> HTML-elementin avulla.

5 Tuloksen analysointi

Työn tuloksessa, eli Azure DevOps -projektina luodussa web-sovelluksessa, osoitettiin joidenkin DevOps-toimintamallin menetelmien, React-sovelluskehityksen ja Microsoft Azure -pilvialustan yhteensopivuus. Testisovelluksen julkaisussa ei törmätty teknisiin esteisiin, jotka voisivat olla haasteellisia tai ongelmallisia toimeksiantajalle.

Uuden alustan ja uuden toimintamallin omaksuminen ei kuitenkaan ole ihan yksinkertaista. Microsoft Azuren tarjoamien palveluiden määrä on valtava ja niistä sopivien valinta voi viedä aikaa. Alusta tarjoaa kuitenkin hyvät mahdollisuudet seurata kulurakennetta ja suorituskykyä, joten valintoja voidaan arvioida ja muuttaa myös niiden perusteella.

Toteutuksessa tutkimusongelmaa lähestyttiin Azure DevOps -palvelujoukon avulla. Azure DevOps -palveluilla on oma käyttöliittymä, joka helpottaa DevOps -toimintamallin käyttämistä ohjelmistoprojekteissa. Versionhallinta, työtehtävien määrittely ja tekijöiden nimeäminen, työvaiheiden automatisointi ja sovelluksen testaus onnistuu tämän käyttöliittymän kautta visuaalisesti ja loogisesti. Tarjolla on myös lukuisia valmiita malleja erilaisille ohjelmistoprojekteille.

Projektissa onnistuttiin lisäämään ohjelmistokehityksen automaatiota jatkuvan integraation ja jatkuvan julkaisun menetelmillä, jotka luvun 2.4 mukaan lisäävät organisaation suorituskykyä ja siten myös kilpailukykyä ja asiakastyytyväisyyttä. Lisäksi DevOps-toimintamallia laajennettiin myös Azure DevOps -palvelujoukon ulkopuolelle. Sovelluksen jatkuvaan monitorointiin otettiin käyttöön Application Insights -palvelu, jolla pystytään seuraamaan sovelluksen suorituskykyä. Projektilla tuotiin esille työvaiheita, joilla React-sovelluskehitykseen voidaan sisällyttää DevOps-menetelmiä käytännössä. Kaikkia DevOps-menetelmiä, kuten ohjelmiston testausta, ei tässä toteutuksessa kuitenkaan käytetty.

Kaiken kaikkiaan Microsoft Azure tarjoaa hyvät mahdollisuudet DevOps-toimintamallin käyttöön eikä aloittaminen vaadi suuria kuluja. Jos DevOpsia haluaa käyttää laajasti ja maksimoida turvallisuutta, joutuu käyttäjä kuitenkin panostamaan projektiin myös taloudellisesti. Esille nousi kuitenkin vaikeus arvioida Microsoft Azure -pilvialustan käytön hintaa, koska palvelut ovat käytön mukaan laskutettavia. Esimerkiksi sovelluksen skaalautuvuuden lisääminen ja Azure Test Plans -palvelun käyttöönotto olisivat vaatineet korkeampaa hinnoittelutasoa. Jos halutut palvelut ovat tiedossa ja niiden käyttö on melko tasaista, voidaan kuluja arvioida tarkemmin. Microsoft Azure tarjoaa mahdollisuuden seurata kuluja ja niiden ennustetta tilaajatietojen kautta.

6 Pohdinta

Opinnäytetyön tavoitteena oli selvittää toimeksiantajalle DevOps-toimintamallin periaatteita ja toimintatapoja, miten niitä voidaan käytännössä alkaa toteuttaa Microsoft Azuren -pilvialustalla ja mitä hyötyä uuden toimintamallin omaksumisesta voisi olla. Työn tuloksena julkaistiin web-sovellus Azure DevOps -palvelujoukkoa hyödyntäen. Pohdinta perustuu Montosen vuonna 2019 suoritetun työharjoittelujakson Tridea Oy:ssä aikana tekemiin havaintoihin toimintatavoista, joita vertailtiin luvussa 2 kuvailtuihin DevOps-toimintamallin menetelmiin sekä työn tuloksena syntyneen Azure DevOps-projektin työvaiheisiin.

Tridea Oy:llä on jo käytössään joitain ketteriä ohjelmistokehityksen menetelmiä, kuten hajautettu versionhallinta ja tehtävienhallintataulut, joita voidaan pitää myös osana DevOps-toimintamallia. DevOps-toimintamallissa on kuitenkin vielä paljon menetelmiä, joita hyödyntämällä voidaan kehittää ohjelmistokehitysprosessia ja siten pyrkiä palvelemaan asiakkaita paremmin. Etenkin jatkuvan toimituksen ja jatkuvan julkaisun menetelmillä voidaan automatisoida Tridea Oy:ssä jo nyt tehtäviä työvaiheita. DevOps-toimintamallia voidaan ottaa käyttöön myös laajemmin ohjelmistokehitysprosessin vaatimusten kasvaessa. DevOps-toimintamallia voidaan kehittää käsi kädessä Tridea Oy:n tarjoaman ohjelmiston kanssa.

Luvussa 2.4 esiteltyjen raporttien tuloksista kävi ilmi, että DevOpsin edistämällä ohjelmiston toimituksen suorituskyvyllä on suora yhteys organisaation suorituskykyyn. Ottamalla käyttöön DevOps-menetelmiä, voidaan edistää ohjelmiston nopeutta, vakautta ja saatavuutta, minkä on osoitettu parantavan organisaation kannattavuutta, tuottavuutta ja asiakastyytyväisyyttä. Organisaation suorituskyvyn kasvulla saavutetaan parempi kilpailukyky markkinoilla, mikä onkin koko DevOps-toimintamallin perusajatus.

Kuten luvussa 2.4 kerrotaan, DevOps-menetelmien käyttöönotossa täytyy kuitenkin ottaa huomioon, että uudet menetelmät ja roolit voivat johtaa organisaation tarpeeseen palkata ja johtaa enemmän henkilöstöä. Myös infrastruktuurin muutokset tulevat tarpeelliseksi. Tridea Oy:n nykyiset ohjelmistot ovat virtuaalikoneilla Microsoft

Azuren ulkopuolella ja koko ohjelmiston siirtäminen ja osittainenkin kontittaminen vaatii yritykseltä ponnistuksia eikä odotettua hyötyä välttämättä saada, ellei sitä tehdä oikein. Tridea Oy:n etuna muutoksessa voi kuitenkin pitää sitä, että ohjelmistoa on tehty vielä verrattain vähän aikaa eikä sen koko ole vielä kovin suuri. Pienehkö kehitystiimi ei myöskään ole toiminut useita vuosia vaan yhdellä tavalla eikä oletettavasti ole siten juuttunut rutiineihinsa.

Tridea Oy:llä ei ole käytössään paikallisia palvelinkoneita vaan se hyödyntää pilviteknologioita jo virtuaalikoneiden muodossa. Pilvipalveluita voi kuitenkin hyödyntää myös laajemmin ja esimerkiksi Microsoft Azuren palveluilla voidaan infrastruktuurista luoda dynaamisesti ja automaattisesti skaalautuvaa. Vaikka skaalautuvuutta ei todennettu työn käytännön osuudessa, luvussa 3.2 todetaan tämän olevan mahdollista Azuren sovelluspalvelualustalla (Azure App Service), jota käytettiin julkaisualustana myös käytännön osuudessa. Skaalautuvuus vaatii sovelluspalvelualustan korkeamman hinnoittelutason valitsemista, mikä voi vaikuttaa päätökseen kyseisen DevOps-menetelmän käyttöönotosta.

Kuten luvussa 2.3.3 kerrotaan, käytön mukaan laskutettava pilvilaskenta on kustannustehokasta ja joustavaa, joten siitä on hyötyä kaikenkokoisille yrityksille. Pilvipalveluiden tavoitteena onkin tehdä kaikenkokoisten yritysten liiketoiminnasta helpompaa ja tehokkaampaa. Tämän tiedon pohjalta sen tulisi soveltua myös Tridea Oy:lle. Koska yksittäisten resurssien ja palveluiden hinnat ovat saatavilla, voidaan odotetun käytön perusteella ennustaa myös tulevia kustannuksia. Käytön määrää voi kuitenkin olla vaikea arvioida etukäteen, joten tarkin tapa selvittää kustannuksia lienee kokeilujakso, jossa kaikki halutut palvelut ovat käytössä. Luvussa 2.4 esitellyistä tuloksista käy myös ilmi, että pilviteknologian hyödyntäminen edistää organisaation korkeaa suorituskykyä ja se on yksi avaintekijä eliittitason toimijoilla.

Opinnäytetyön käytännön osuus toi esille vaiheita, joilla DevOps-toimintamallia voidaan ottaa käyttöön ja kuinka eri menetelmiä hyödynnetään Microsoft Azuren -pilvialustalla. DevOps-toimintamallin hyödyllisyyttä toimeksiantajalle ei kuitenkaan käytännön osuudessa pystytty toteamaan tarkasti. Hyödyllisyyden arvioiminen nojaa enimmäkseen opinnäytetyön teoriaosuudessa esitettyihin huomioihin. Käytännön

hyödyllisyyttä DevOps-toimintamallista ohjelmistokehityksessä voikin olla vaikea todeta yhden palvelun osalta. Uudet ominaisuudet ovat uusia vain kerran, joten niiden kehitystä ja julkaisua erilaisilla toimintamalleilla on vaikea vertailla. Käytännön osuudella kuitenkin todettiin, ettei DevOps-toimintamallin ja React-sovelluskehityksen yhdistämiselle ole teknisiä esteitä Microsoft Azure -alustalla.

Jatkokehitys

Luvussa 4 esitetyt ratkaisut tarjoavat hyvän pohjan myös jatkokehitykselle ja DevOps-toimintamallin laajemmalle käytölle. Kun projektin perustaminen ja toimintamallin peruskohdat on esitetty käytännössä, voidaan palveluita lisätä tarpeen mukaan. Azure DevOps tarjoaa luvussa 4 esiteltyjen palveluiden lisäksi myös muita ohjelmistokehityspalveluita, joita voidaan hyödyntää myös laajemmissa kokonaisuuksissa. Esimerkiksi Azure Boards -palvelu auttaa ketterässä ohjelmistokehityksessä käytettävien tehtävä- ja kehitysjonon hallintaa taulujen avulla. Azure Test Plans -palvelun avulla sovellukselle voidaan puolestaan luoda erilaisia testejä ja kerätä diagnostiikkaa koko kehitysprosessin elinkaarelta. Myös esitettyjä palveluita voidaan hyödyntää laajemmin. Esimerkiksi Azure Pipelines -palvelussa voidaan jatkuvan toimituksen vaiheessa ottaa käyttöön manuaalinen hyväksymismenettely, jolloin Azure DevOps organisaation valittu jäsen saa hyväksyttäväksi sähköposti-ilmoituksen uudesta versiosta. Tällöin uusi versio julkaistaan vasta hyväksynnän jälkeen.

Lähteet

11 Points to Consider When Virtualizing Security. 2018. Artikkeli Infosec-sivustolla. Viitattu 14.11.2019. https://resources.infosecinstitute.com/11-points-consider-virtualizing-security/#gref.

Amazon, Microsoft, Google and Alibaba Strengthen their Grip on the Public Cloud Market. 2019. Artikkeli Synergy Research Groupin verkkosivuilla. Viitattu 29.1.2020. https://www.srgresearch.com/articles/amazon-microsoft-google-and-alibaba-strengthen-their-grip-public-cloud-market.

App Service overview. 2020. Artikkeli Microsoft Azure -sivustolla. Viitattu 6.5.2020. https://docs.microsoft.com/en-us/azure/app-service/overview.

Azure App Service plan overview. 2017. Artikkeli Microsoft Azure -sivustolla. Viitattu 6.5.2020. https://docs.microsoft.com/en-us/azure/app-service/overview-hosting-plans.

BisLenz. N.d. BisLenz -johdon raportointitilan esittelyteksti bislenz.com -verkkosivulla. Viitattu 27.3.2020. https://bislenz.com/.

DevOps on ketteryyden uusi määritelmä. 2017. Artikkeli Elisan sivustolla. Viitattu 5.11.2019. https://hub.elisa.fi/devops-ketteryyden-uusi-maaritelma/.

Exploratory and manual testing scenarios and capabilities. 2019. Artikkeli Azure DevOpsin esittelysivulla. Viitattu 1.4.2020. https://docs.microsoft.com/fi-fi/azure/devops/test/overview?view=azure-devops.

Forsgren, N., Frazelle, J., Humble, J. & Smith, D. 2019. Accelerate State of DevOps 2019. Raportti. Viitattu 22.11.2019. https://services.google.com/fh/files/misc/state-of-devops-2019.pdf.

Four key metrics. N.d. Artikkeli thoughtworks.com-sivustolla. Viitattu 22.11.2019. https://www.thoughtworks.com/radar/techniques/four-key-metrics.

Guckenheimer, S. 2017. What is Continuous Integration? Artikkeli Microsoftin dokumentaatiosivustolla. Viitattu 7.11.2019. https://docs.microsoft.com/en-us/azure/devops/learn/what-is-continuous-integration.

Hering, M. 2014. Continuous Everything in DevOps…What is the difference between CI, CD,CD,…? Artikkeli notafactoryanymore.com-sivustolla. Viitattu 10.5.2020. https://notafactoryanymore.com/tag/continuous-delivery/.

Hämäläinen, P. 2016. Mikropalvelut nousivat hypen huipulle – mitä hyötyä niistä on? Artikkeli Tivin verkkolehdessä. Viitattu 5.11.2019. https://www.tivi.fi/uutiset/mikropalvelut-nousivat-hypen-huipulle-mita-hyotya-niista-on/2cd7b4f9-74fc-3e7b-830b-f2b4eaf49e42.

Introduction to cloud computing. N.d. Artikkeli Microsoft Azuren esittelysivulla. Viitattu 22.1.2020. https://azure.microsoft.com/en-us/overview/.

Kornilova, I. 2017. DevOps is a culture, not a role! Artikkeli medium.com sivustolla. Viitattu 5.11.2019. https://medium.com/@neonrocket/devops-is-a-culture-not-a-role-be1bed149b0.

Kylmäkoski, J. 2018. DevOps – kuinka varmistaa yrityksen arvontuotto ohjelmistokehityksessä. Artikkeli oppia.fi blogissa. Viitattu 19.11.2019. https://blog.oppia.fi/2018/08/10/devops-kuinka-varmistaa-yrityksen-arvontuotto-ohjelmistokehityksessa/

Mezak, S. 2018. The Origins of DevOps: What's in a Name? Artikkeli devops.com sivustolla. Viitattu 5.11.2019. https://devops.com/the-origins-of-devops-whats-in-a-name/.

Mitä teemme. N.d. Mitä teemme -osio Tridean verkkosivuilla. Viitattu 14.11.2019. https://tridea.fi/fi/mita-teemme/.

Null, C. N.d. 10 companies killing it at DevOps. Artikkeli techbeacon sivustolla. Viitattu 19.11.2019. https://techbeacon.com/devops/10-companies-killing-it-devops.

Oteyowo, T. 2018. DevOps in a Scaling Environment. Artikkeli medium.com-sivustolla. Viitattu 10.5.2020. https://medium.com/tech-tajawal/devops-in-a-scaling-environment-9d5416ecb928.

Pricing for Azure DevOps. N.d. DevOps palveluiden hinnoittelusivu Azure -sivustolla. Viitattu 10.5.2020. https://azure.microsoft.com/en-gb/pricing/details/devops/azure-devops-services/.

Rautonen, T. 2013. DevOps – Sovelluskehittäjän roolin evoluutio. Artikkeli Gofore sivustolla. Viitattu 5.11.2019. https://gofore.com/devops-sovelluskehittajan-roolin-evoluutio/.

Riley, C. 2014. Meet Infrastructure as Code. Artikkeli devops.com sivustolla. Viitattu 10.2.2020. https://devops.com/meet-infrastructure-code/.

Rudnäs, N. 2018. Ohjelmien kontitus ja sen hyödyt. Artikkeli SeAMK verkkolehdessä. Viitattu 15.11.2019. https://verkkolehti.seamk.fi/index.php/arkisto/2018/ohjelmien-kontitus-ja-sen-hyodyt/.

Schultz, C. 2019. What Is Infrastructure as Code? How It Works, Best Practices, Tutorials. Artikkeli stackify.com sivustolla. Viitattu 10.2.2020. https://stackify.com/what-is-infrastructure-as-code-how-it-works-best-practices-tutorials/.

Tozzi, C. 2016. Reasons Why DevOps May Not Work for You. Artikkeli devops.com sivustolla. Viitattu 21.11.2019. https://devops.com/reasons-devops-may-not-work-for-you/.

Virtualization. 2019. Artikkeli IBM-sivustolla. Viitattu 12.11.2019. https://www.ibm.com/cloud/learn/virtualization-a-complete-guide.

Watts, S. N.d. The Role of Virtualization in DevOps. Artikkeli BMC:n blogisivustolla. Viitattu 15.11.2019. https://www.bmc.com/blogs/devops-virtualization/.

What is Azure? N.d. Artikkeli Microsoft Azuren esittelysivulla. Viitattu 22.1.2020. https://azure.microsoft.com/en-us/overview/what-is-azure/.

What is Azure Artifacts? 2018. Artikkeli Azure DevOpsin esittelysivulla. Viitattu 1.4.2020. https://docs.microsoft.com/fi-fi/azure/devops/artifacts/overview?view=azure-devops.

What is Azure Boards? 2020. Artikkeli Azure DevOpsin esittelysivulla. Viitattu 19.2.2020. https://docs.microsoft.com/en-us/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops&tabs=scrum-process.

What is Azure DevOps? 2019. Artikkeli Azure DevOpsin esittelysivulla. Viitattu 6.5.2020. https://docs.microsoft.com/fi-fi/azure/devops/user-guide/what-is-azure-devops?view=azure-devops.

What is Azure Pipelines? 2019. Artikkeli Azure DevOpsin esittelysivulla. Viitattu 19.2.2020. https://docs.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops.

What is Azure Repos? N.d. Artikkeli Azure DevOpsin esittelysivulla. Viitattu 19.2.2020. https://docs.microsoft.com/en-us/azure/devops/repos/get-started/what-is-repos?view=azure-devops.

What is cloud computing? N.d. Artikkeli Microsoftin dokumentaatiosivustolla. Viitattu 12.11.2019. https://docs.microsoft.com/fi-fi/learn/modules/principles-cloud-computing/2-what-is-cloud-computing.

What is Continuous Delivery? N.d. Artikkeli Amazon Web Services -sivustolla. Viitattu 7.11.2019. https://aws.amazon.com/devops/continuous-delivery/.

What is Continuous Integration? N.d. Artikkeli Amazon Web Services -sivustolla. Viitattu 7.11.2019. https://aws.amazon.com/devops/continuous-integration/.

What is DevOps? N.d.a. Artikkeli Amazon Web Services -sivustolla. Viitattu 5.11.2019. https://aws.amazon.com/devops/what-is-devops/.

What is DevOps? N.d.b. Artikkeli Microsoft Azure -sivustolla. Viitattu 22.11.2019. https://azure.microsoft.com/en-us/overview/what-is-devops/

Liitteet

Liite 1. Azure Application Insights -skripti

```
<script type="text/javascript">
  var sdkInstance="appInsightsSDK";window[sdkInstance]="appInsights";
  var aiName=window[sdkInstance],
  aisdk=window[aiName]||function(n){var o={config:n,initialize:!0},t=document,e=window,i="script";
  setTimeout(function(){var e=t.createElement(i);
  e.src=n.url||"https://az416426.vo.msecnd.net/scripts/b/ai.2.min.js",
  t.getElementsByTagName(i)[0].parentNode.appendChild(e)});
  try{o.cookie=t.cookie}catch(e){}function a(n){o[n]=function()
  {var e=arguments;o.queue.push(function(){o[n].apply(o,e)})}}o.queue=[],o.version=2;
  for(var s=["Event","PageView","Exception","Trace","DependencyData","Metric","PageViewPerformance"];
  s.length;)a("track"+s.pop());var r="Track",c=r+"Page";a("start"+c),a("stop"+c);var u=r+"Event";
  if(a("start"+u),a("stop"+u),a("addTelemetryInitializer"),
  a("setAuthenticatedUserContext"),a("clearAuthenticatedUserContext"),
  a("flush"),o.SeverityLevel={Verbose:0,Information:1,Warning:2,Error:3,Critical:4},
  !(!0===n.disableExceptionTracking||n.extensionConfig&&n.extensionConfig.ApplicationInsightsAnalytics&&
  !0===n.extensionConfig.ApplicationInsightsAnalytics.disableExceptionTracking))
  {a("_"+(s="onerror"));var p=e[s];e[s]=function(e,n,t,i,a)
  {var r=p&&p(e,n,t,i,a);return!0!==r&&o["_"+s]({message:e,url:n,lineNumber:t,columnNumber:i,error:a}),r},
  n.autoExceptionInstrumented=!0}return o}(
    {
      instrumentationKey:"04ae0590-0a8e-442c-b763-6ffd71508b77"
    }
  );(window[aiName]=aisdk).queue&&0===aisdk.queue.length&&aisdk.trackPageView({});
</script>
```