Valerio Tatananni

Pilvipalvelujen ja DevOpsin hyödyntäminen Web-kehityksessä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

16.4.2025

Tiivistelmä

Tekijä:	Valerio Tatananni
Otsikko:	Pilvipalvelujen ja DevOpsin hyödyntäminen Web-kehityksessä
Sivumäärä:	35 sivua
Aika:	16.4.2025
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Monimuoto/Verkkototeutus
Ohjaajat:	Osaamisaluejohtaja Janne Salonen

Tässä opinnäytetyössä tutkittiin pilvipalvelujen ja DevOpsin hyödyntämistä web-kehityksen näkökulmasta, keskittyen erityisesti Microsoft Azure ja Azure DevOps -alustoihin. Työn tavoitteena oli rakentaa täysin automatisoitu CI/CD-putki, joka kattaa koko ohjelmiston aina rakentamisesta testaukseen ja julkaisuun. Julkaisuputki toteuttiin Azure DevOpsin avulla, ja siinä julkaistiin tuotantoon Api Currency Converter -sovellus, joka on tehty Reactilla ja Java Spring Bootilla. Työn lopputuloksena oli toimiva CI/CD-prosessi, joka mahdollistaa sovelluksen automaattisen päivittämisen ja julkaisemisen Microsoft Azuren pilvipalveluun.

Alkuun työssä tutustuttiin siihen, mitä pilvipalvelut ja DevOps ovat ja millaista ohjelmistokehitys oli ennen niiden yleistymistä. Näihin aiheisiin syvennyttiin ja selvitettiin pilvipalvelujen ja DevOpsin syntymiseen johtaneet tekijät yleisellä tasolla. Lisäksi esiteltiin niiden tuomia hyötyjä ohjelmistokehityksen kannalta. Tämän jälkeen tutkittiin sekä Azure DevOpsin että Microsoft Azuren App Servicen ja Static Web Appsin käyttöönottoa.

Lopuksi rakennettiin CI/CD-julkaisuputki niin frontendille kuin backendille ja testattiin sen toimivuutta. Tähän vaiheeseen sisältyy myös pohdintaa siitä, miten Azure DevOpsin käyttöliittymää voi hyödyntää julkaisuputken ajon tulosten perusteella ja miten Microsoft Azurea ja Azure DevOpsia voisi vielä enemmän hyödyntää projektissa.

Avainsanat: Pilvipalvelut, DevOps, Microsoft Azure, Azure DevOps

---

Abstract

Author:	Valerio Tatananni
Title:	Utilizing Cloud Services and DevOps in Web Development
Number of Pages:	35 pages
Date:	16 April 2025
Degree:	Bachelor of Engineering
Degree Programme:	Information and communication technology
Professional Major:	Blended/Online studies
Supervisors:	Janne Salonen, Director of school

This thesis explores the utilization of cloud services and DevOps in the context of web development, with a particular focus on the Microsoft Azure and Azure DevOps platforms. The primary goal was to build a fully automated CI/CD pipeline that covers the entire software lifecycle from building and testing to deployment. The deployment process was implemented using Azure DevOps and used to release the Api Currency Converter apllication built with React and Java Spring Boot. The end result was a functional CI/CD process that enables automatic updates and deployment of the application to Microsoft Azure’s cloud services.

The thesis begins by examining what cloud services and DevOps are, and how software development was conducted before their rise in popularity. It delves into the key factors that led to their emergence and highlights the benefits they bring to software development. Following this, the study explores the setup and use of Azure DevOps, Microsoft Azure App Service and Static Web Apps.

Finally, a CI/CD pipeline was built for both the frontend and backend, and its functionality was tested. This phase also includes reflections on how Azure DevOps' interface can be used to analyze the pipelines run results, as well as suggestions for further enhancements in the project using Microsoft Azure and Azure DevOps.

Keywords: Cloud services, DevOps, Microsoft Azure, Azure DevOps

Sisällys

Lyhenteet

SaaS: Software as a Service. Pilvipalvelu, jossa sovellusta käytetään internetissä.

PaaS: Platform as a Service. Pilvipalvelu, mikä antaa alustan sovelluskehitykselle.

IaaS: Infrastructure as a Service. Pilvipalvelu, missä vuokrataan verkkolaitteita ja infrastruktuuria.

CI/CD: Continuous integration, continuous delivery. DevOps menetelmä, millä automatisoidaan sovelluskehityksen ja it-toimintojen prosesseja.

IDE: Integrated development environment. Ohjelmointiympäristö, eli sovellus millä ohjelmoidaan.

REST: Representational state transfer. Arkkitehtuurityyli ohjelmistorajapintojen toteuttamiseen.

1 Johdanto

Nykyaikaisessa web-kehityksessä on useita toisiinsa kytkeytyviä vaiheita, joita usein toteuttavat eri alojen asiantuntijat. Yksi tehokas tapa tehostaa kehitysprosessia on käyttää pilvipalveluita. Toinen merkittävä keino on DevOpsin hyödyntäminen, jonka avulla ohjelmistokehitystiimit voivat julkaista tuotantokelpoista koodia huomattavasti nopeammin ja luotettavammin. Yhdistämällä pilvipalvelut ja DevOps ohjelmistokehityksestä tulee sujuvaa, joustavaa ja helposti skaalautuvaa.

Tässä opinnäytetyössä tutkitaan, miten pilvipalveluita ja DevOpsia voidaan hyödyntää web-kehityksessä. Pilvipalvelualustaksi valittiin Microsoft Azure ja DevOps-työkaluksi Azure DevOps, sillä nämä palvelut on helppo ottaa käyttöön ja niiden välinen integraatio toimii saumattomasti. Azure DevOpsilla rakennettiin kaksi automatisoitua pipelinea, joiden avulla API Currency Converter -niminen web-sovellus julkaistaan tuotantoon Microsoft Azuren App Service- ja Static Web Apps -palveluihin.

Työn alussa selvitetään, mitä pilvipalvelut ja DevOps tarkoittavat, minkä jälkeen siirrytään pilvipalveluiden ja DevOps-työkalujen käyttöönottoon. Sen jälkeen esitellään käytännön esimerkkeihin sitä, miten Azure DevOpsia ja Microsoft Azurea on hyödynnetty projektissa. Lopuksi arvioidaan ratkaisun hyötyjä web-kehityksen näkökulmasta.

2 Pilvipalvelut

Tässä kappaleessa perehdytään siihen, mitä pilvipalvelut ovat ja miksi niitä käytetään nykyaikaisessa ohjelmistokehityksessä. Tarkastelun kohteena ovat pilvipalveluiden hyödyt, ominaispiirteet sekä taustat, jotka ovat johtaneet niiden kehittämiseen ja yleistymiseen.

Ymmärtääksemme pilvipalveluiden merkityksen, on ensin tarkasteltava, millainen ohjelmistokehityksen ja IT-infrastruktuurin toimintaympäristö oli ennen pilvipalveluita.

2.1 Ohjelmistokehitys ennen pilvipalveluita

Ennen pilvipalveluiden yleistymistä yritysten oli tyypillisesti hankittava ja ylläpidettävä oma palvelininfrastruktuuri. Tämä tarkoitti fyysisten konesalien perustamista, palvelimien hankintaa ja asennusta, sekä niiden ylläpitoa. Tällainen toimintamalli vaati merkittäviä investointeja etukäteen – ei ainoastaan laitteiston, vaan myös osaavan henkilöstön, tilojen ja sähkönkulutuksen osalta.

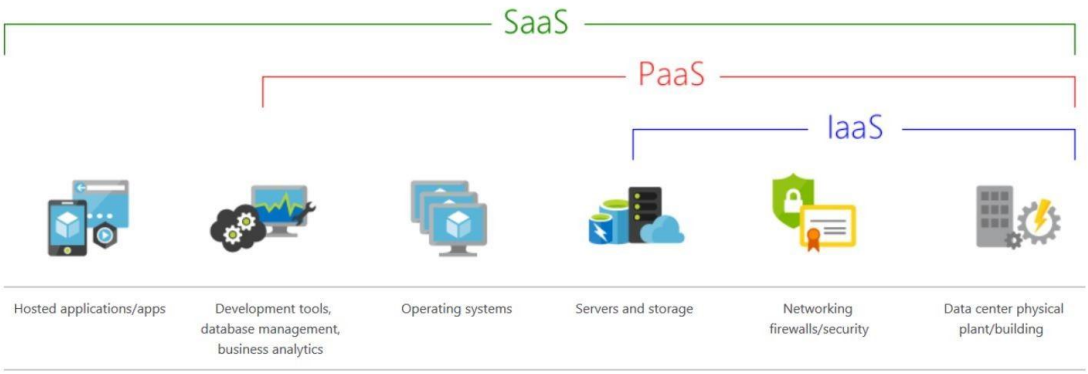
Lisäksi, mikäli yrityksen palvelu tarvitsi enemmän suorituskykyä, laskentatehon kasvattaminen tarkoitti uuden laitteiston hankintaa. Tämä vei aikaa ja toi mukanaan lisää kustannuksia [1].

2.2 Mitä ovat pilvipalvelut?

Pilvipalvelut ovat sovellus- ja infrastruktuuriresursseja, joita käytetään internetin kautta. Palveluntarjoajat tekevät sopimuksia tilaajien kanssa näiden resurssien käytöstä, mikä mahdollistaa asiakkaille tehokkaiden tietojenkäsittelyresurssien käytön ilman tarvetta ostaa tai ylläpitää laitteistoja ja ohjelmistoja. Toisin sanoen käyttäjä vuokraa resursseja palveluntarjoajalta ja maksaa siitä käytön mukaan. Pilvipalveluja käyttäessään organisaatiot pystyvät ulkoistamaan infrastruktuuriin hallinnoinnin ja keskittyä sen sijaan pelkästään sen käyttöön. Palveluntarjoaja tukee erilaisia toimintoja ja työkaluja, jotka auttavat yrityksen toiminnan ylläpitämistä, kuten analysointityökaluja, tallennustilaa, datahallintaa ja tietoturvaa. Pilvipalveluiden avulla organisaatiot voivat vuokrata tarvitsemansa laskentaresurssit ilman suuria alkuinvestointeja, skaalata toimintaa joustavasti lisäämällä tai vähentämällä kapasiteettia tarpeen mukaan sekä kehittää ja julkaista sovelluksia nopeammin ja luotettavammin hyödyntämällä valmiita palveluita ja automaatioita [2].

2.3 SaaS, PaaS ja IaaS

Pilvipalveluista puhuttaessa tulee väistämättä vastaan seuraavat termit - SaaS (Software as a Service), PaaS (Platform as a Service) ja IaaS (Infrastructure as a Service). Ne ovat kolme pääkategoriaa pilvipalveluista, jotka tunnetaan myös palvelutasoina. Selvennyksenä "as a service" tarkoittaa toimintoa, joka tyypillisesti liittyy informaatiotekniikkaan, tietotekniikkaan, tietojenkäsittelyyn tai telekommunikaatioon, mikä tarjotaan käyttäjille palveluna verkon yli [3]. Tällöin käyttäjän ei tarvitse asentaa tai ylläpitää palvelua omilla laitteillaan, vaan sitä voi käyttää suoraan palveluntarjoajan hallinnoimassa ympäristössä. Kuva 1 havainnollistaa palvelutasot.



Kuva 1: SaaS, PaaS ja IaaS

2.3.1 Ohjelmistot palveluna (*Software as a Service*, SaaS)

SaaS on ylin pilvipalvelumalli, jossa käyttäjä käyttää sovellusta suoraan internetin kautta, ilman tarvetta asentaa tai ylläpitää ohjelmistoa omalla laitteellaan. Kaikki tekninen hallinta, kuten päivitykset, tietoturva ja infrastruktuuri, on palveluntarjoajan vastuulla. Näistä syistä palvelu on saatavilla missä vaan millä tahansa laitteella, mutta kuitenkin suojattuna ulkopuolisilta. Palvelun käyttäminen vaatii ainoastaan sovelluksen tai web selaimen käyttöä ja sisäänkirjautumisen.

Tyypillisiä esimerkkejä SaaS-palveluista ovat:

- Sähköposti (esim. Gmail)
- Pilvitallennus (esim. Google Drive, Dropbox, Azure)
- Office-työkalut (esim. Microsoft 365)
- Sosiaalinen media (esim. Facebook, LinkedIn)

2.3.2 Sovellusalusta palveluna (*Platform as a Service*, PaaS)

PaaS on pilvipalvelumallien keskimmäinen kerros, joka vaatii vähemmän käyttäjähallintaa eikä tarjoa suoraa pääsyä käyttöjärjestelmään. Se tarjoaa yrityksille digitaalisen alustan, jonka avulla ne voivat kehittää, testata ja ottaa käyttöön omia sovelluksia ja palveluita ilman tarvetta hallita itse palvelimia, ohjelmistoympäristöjä tai tietoturvatoimia. Tämä palvelumalli on erityisen suosittu ohjelmistokehityksessä, koska se vähentää tarvetta hallita matalamman tason resursseja ja mahdollistaa nopeamman kehityksen ja käyttöönoton. Yrityksille on monesti nopeampaa, halvempaa ja yksinkertaisempaa rakentaa tuotteita ja palveluita valitsemalla PaaS sen skaalautuvuuden ja resurssivapauden takia.

Esimerkiksi App Services, Azure Functions ja Static Web Apps ovat Microsoft Azuren PaaS-palveluita.

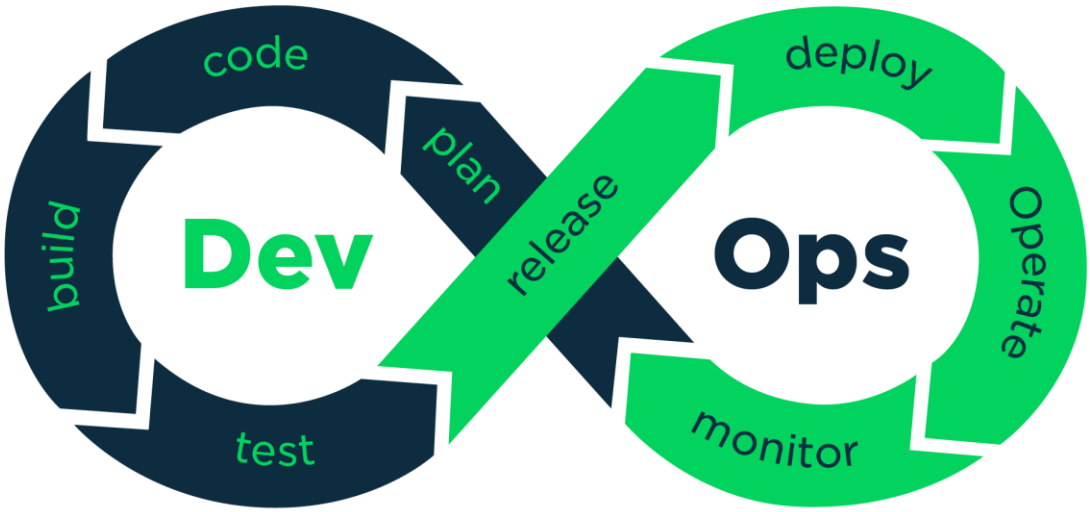
2.3.3 Infrastruktuuri palveluna (*Infrastructure as a Service*, IaaS)

IaaS tarjoaa kaikkein matalimman tason pilvipalvelun, jossa palveluntarjoaja antaa asiakkaille pääsyn virtuaalisiin palvelimiin, tallennustilaan ja verkkoresursseihin. Käyttäjä hallitsee itse käyttöjärjestelmiä, sovelluksia ja tietoturvaa. Sen sijaan, että yritys ostaisi verkkolaitteet ja IT infrastruktuurin, vuokrataan ne suoraan palveluntarjoajalta. IaaS on hyvä valinta organisaatioille, jotka haluavat enemmän kontrollia ja joustavuutta ilman tarvetta ylläpitää omaa fyysistä laitteistoa. Perinteisesti nämä tietojenkäsittelyn osa-alueet olisivat organisaation oman IT-osaston vastuulla omassa konesalissa, mutta IaaS:n avulla organisaatioilla on mahdollisuus saada ne käyttöönsä suoraan pilvipalveluntarjoajalta.

Esimerkkejä Azuressa voi ottaa käyttöön virtuaalikoneita IaaS palveluna, joihin käyttäjät voivat itse asentaa käyttöjärjestelmän jne. Tämän kaltaisissa palveluissa käyttäjät voivat itse hallita resursseja esimerkiksi selainkäyttöliittymän kautta, luoda uusia palvelimia, säätää kapasiteettia ja määrittää verkkoasetuksia [3].

3 DevOps

Yksi opinnäytetyön keskeisistä aihealueista on DevOps. Termi tulee sanoista *Development* (kehitys) ja *Operations* (ylläpito). Se on ihmisten, prosessien ja teknologioiden liitto, jonka tavoite on tuottaa jatkuvaa arvoa käyttäjille. Tiimeille DevOps mahdollistaa eri roolien kuten kehityksen, IT-operaatioiden, laadunvarmistuksen ja tietoturvan koordinoinnin yhteensovittamisen parempien ja luotettavimpien tuotteiden aikaan saamiseksi. Ottamalla käyttöön DevOps käytännöt ja työkalut tiimit saavat kyvyn vastata käyttäjien tarpeisiin paremmin, nopeammin, luotettavammin ja tehokkaammin [4]. DevOps koostuu kuvan 2 mukaisesti monesta eri vaiheesta, jotka avataan tarkemmin seuraavaksi.



Kuva 2: DevOpsin elinkaari

3.1 Suunnitteluvaihe

Suunnitteluvaiheessa tiimit ideoivat, määrittelevät ja kuvailevat projektinsa vaatimukset. Suunnittelu tehdään yksityiskohtaisesti, ja siinä vastataan muun muassa siihen, millaisia ominaisuuksia ja kyvykkyyksiä siltä odotetaan. Kehitys jaetaan yksittäisiin tehtäviin, jotka jaetaan tiimin jäsenten kesken. Edistystä seurataan hyödyntämällä esimerkiksi Scrumia tai jotain muuta ketterän kehityksen menetelmää.

3.2 Kehitysvaihe

Kehitysvaihe pitää sisällään kaikki sovelluskehityksen osa-alueet kuten ohjelmakoodin kirjoittamisen, testaamisen, katselmoinnin ja koodin integroinnin tiimin jäsenten toimesta sekä koodin rakentamisen käyttövalmiiksi artefakteiksi, jotka voidaan ottaa käyttöön eri ympäristöissä. DevOps tiimit hyödyntävät erilaisia työkaluja kuten automaatiota ja versionhallintaa, ja etenevät kehitystyössä pienin iteraatioin jatkuvan integraation ja jatkuvan julkaisun avulla (CI/CD, Continuous integration and Continuous delivery).

3.3 Julkaisuvaihe

Julkaisuvaiheessa sovellus otetaan käyttöön tuotantoympäristöön johdonmukaisesti ja luotettavasti pitäen sisällään täysin hallinnoidun perusinfrastruktuurin käyttöönoton ja konfiguroinnin, joka muodostaa sovellukselle tarvittavat ympäristöt. DevOps-tiimit määrittävät julkaisuprosessin selkeillä vaiheilla, joihin voi sisältyä manuaalisia hyväksyntöjä esimerkiksi laadunvarmistuksen tai tietoturvan näkökulmasta. Julkaisuprosessissa hyödynnetään automaatiota, joka siirtää sovelluksen vaiheittain eteenpäin kehitysympäristöstä kohti tuotantoa, kunnes se on asiakkaan saatavilla. Tämä tekee prosesseista skaalautuvia, toistettavia ja hallittuja. Näin DevOps tiimit pystyvät julkaisemaan tuotteitaan usein, helposti ja luotettavasti ilman huolta tai riskiä virheistä.

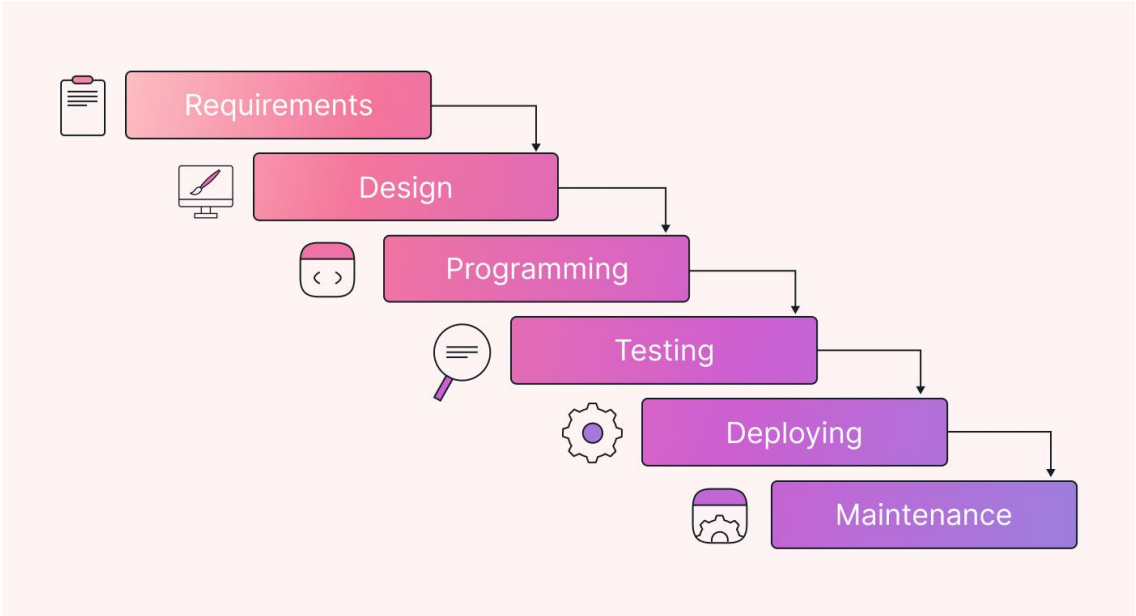
3.4 Käyttövaihe

Käyttövaiheessa keskitytään sovellusten ylläpitoon, valvontaan ja vianmääritykseen tuotantoympäristöissä. DevOps-käytäntöjä hyödyntävät tiimit pyrkivät varmistamaan järjestelmien luotettavuuden ja saatavuuden sekä minimoimaan mahdolliset käyttökatkot. Ongelmat pyritään havaitsemaan ennen kuin ne vaikuttavat käyttäjiin ja niihin pyritään reagoimaan nopeasti vahinkojen minimoimiseksi. Tällaisen toimintavarmuuden ylläpitäminen edellyttää kattavaa telemetriaa, tehokkaita hälytysjärjestelmiä ja täyttä näkyvyyttä sekä sovelluksiin että taustalla toimivaan infrastruktuuriin.

3.5 DevOpsin historia

DevOpsin juuret ulottuvat 2000-luvun alkuun, jolloin ohjelmistokehityksessä alkoi näkyä tarve ketterämmille toimintamalleille. Ennen tätä yleisesti käytetty malli oli niin kutsuttu vesiputousmalli, joka havainnollistetaan kuvassa 3. Siinä kehitystyö eteni lineaarisesti vaiheesta toiseen, ja tuotteen julkaisu saattoi viedä kuukausia tai jopa vuosia. Vesiputousmallissa projekti jaetaan ennalta määritettyihin vaiheisiin, joista siirrytään seuraavaan vasta aiemman vaiheen valmistuttua, sillä vaiheet ovat riippuvaisia toisistaan. Tämän lisäksi ohjelmistokehittäjät ja IT-osasto eivät juurikaan tehneet yhteistyötä keskenään, vaan tekivät omia tehtäviään toisistaan riippumatta. Lyhyesti kuvailtuna vesiputousmallin vaiheet ovat seuraavat:

- Vaatimusten määrittely
- Suunnittelu
- Ohjelmointi
- Testaus
- Käyttöönotto
- Ylläpito



Kuva 3: Vesiputousmalli

Vaikka mallissa on hyviä puolia, kuten helppokäyttöisyys ja hallinta, on se kuitenkin varsin tehoton ja riskialtis menetelmä. Tämän lisäksi asiakkaan vaatimukset saattavat muuttua kesken kehityksen, jolloin suunnitelmaa on usein liian myöhäistä muuttaa. Pahimmassa tapauksessa projekti on aloitettava alusta tällaisessa tapauksessa. Tämänkaltaiset ongelmat olivat yleisesti tiedossa jo 1990-luvun lopussa, ja sen korvaavaksi menetelmäksi kehittyi niin sanotut ketterän kehityksen menetelmät, joissa vaiheet jaetaan lyhyempiin jaksoihin, joita kutsutaan sprinteiksi. Tavoite oli puuttua vesiputousmallin ongelmiin, jotta ohjelmakoodia voidaan julkaista nopeammin ja luotettavammin ja jotta asiakastyytyväisyys paranisi. Kuitenkin vielä ketterän kehityksen yleistyttyä koettiin, että ohjelmistokehittäjät ja it-osasto eivät tehneet tarpeeksi yhteistyötä. Sen takia vuosina 2007–2008 Patrick Debois – IT konsultti, piti tapaamisen ketterästä kehityksestä Agile Conferencessä, minne liittyi Andrew Shafer. DevOps sai alkunsa tästä, ja vuonna 2010 Continuous Delivery niminen kirja julkaistiin, jossa kerrotaan DevOpsin pääperiaatteet. Viimeistään tässä vaiheessa DevOps on noussut yleiseen tietoon ja saavuttanut suosiota, ja sitä on kehitetty jatkuvasti eteenpäin aina siitä lähtien. DevOps on vaikuttanut ohjelmistokehitykseen muun muassa

- Nopeuttamalla kehitystä suunnitelmasta tuotantoon
- Parantamalla yhteistyötä kehittäjien, testaajien ja IT-osastojen välillä
- Parantamalla ohjelmistokoodin laatua automaattisen testaamisen ja julkaisun myötä
- Lisäämällä tehokkuutta poistamalla manuaalisia vaiheita ja suoraviivaistamalla prosesseja
- Parantamalla asiakastyytyväisyyttä

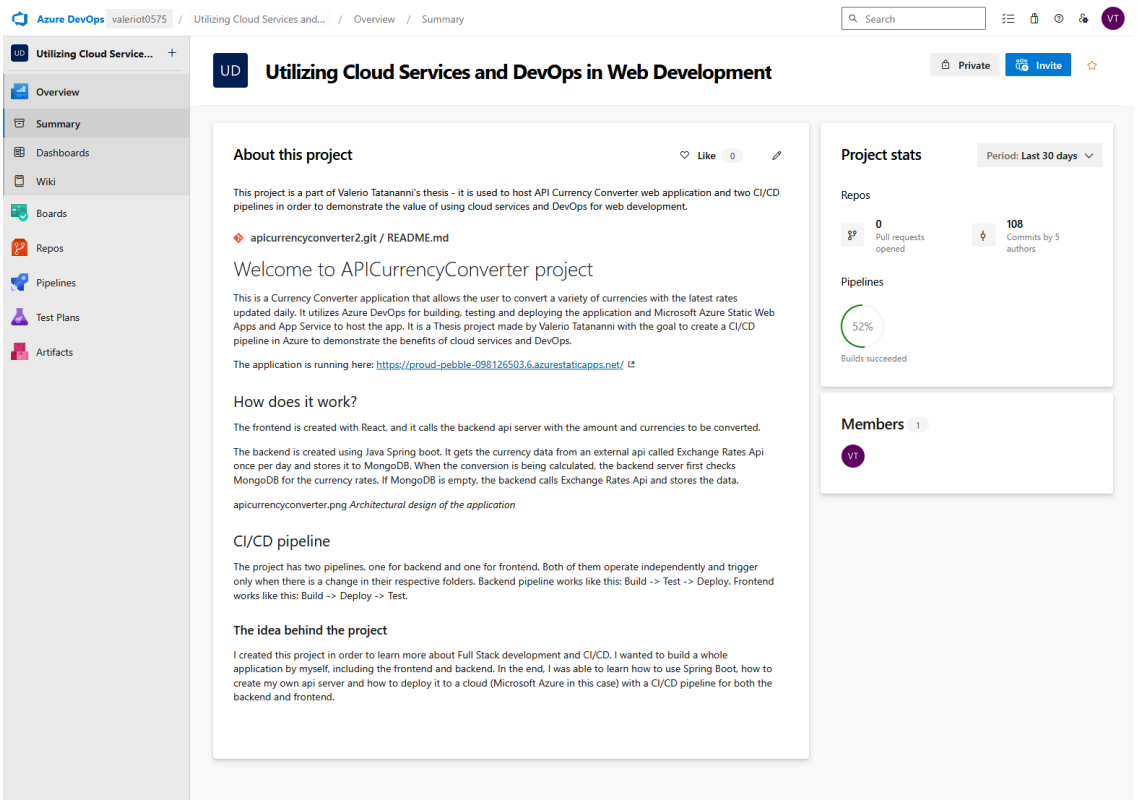
Näin ollen DevOpsin merkitystä ohjelmistokehityksen kannalta ei voi korostaa tarpeeksi [5]. Seuraavassa kappaleessa perehdytään tarkemmin Azuren DevOps palveluun.

4 Azure DevOps

Azure DevOps on Microsoftin tarjoama kattava projektinhallintapalvelu, joka tukee koko ohjelmistokehitysprosessia. Se tuo yhteen kehittäjät, projektipäälliköt ja muut tiimin jäsenet ja sen avulla organisaatiot voivat luoda ja parantaa tuotteita nopeammin kuin perinteisillä ohjelmistokehitysmenetelmillä. Azure DevOpsia voi käyttää joko pilvessä tai omassa ympäristössä hyödyntäen Azure DevOps Serveriä. Azure DevOps tarjoaa integroituja ominaisuuksia, joita voi käyttää verkkoselaimen tai IDE:n (integrated development environment) kautta. Organisaatiot voivat käyttää kaikkia Azure DevOpsiin sisältyviä palveluita tai vain niitä, joita kokevat tarvitsevansa työnkulkujen tehostamiseksi [6].

Azure DevOpsin keskeiset palvelut ovat seuraavat ja ne on saatavilla käyttöliittymän vasemmassa laidassa kuvan 4 mukaisesti [7]:

- Dashboards – Kustomisoitavissa oleva yleiskatsaus projektista.
- Wiki – Dokumentaation kirjoittamiseen ja jakamiseen tarkoitettu työkalu.
- Boards – Tehtävien ja työjonojen hallinta (Scrum, Kanban jne.).
- Repos – Git-pohjainen versionhallinta Azure DevOpsin sisällä.
- Pipelines – Automatisoidut build-, testaus- ja deploy-prosessit.
- Test Plans – Testauksen hallinta ja seuranta.
- Artifacts – Binääritiedostojen ja muiden artefaktien hallinta ja jakelu.



Kuva 4: Azure DevOpsin perusnäkymä

Tässä opinnäytetyössä käytetään Azure DevOpsin palveluista Repos ja Pipelines ominaisuuksia versionhallintaa ja CI/CD pipelinen luomista varten. Seuraavissa kappaleissa esitellään yksityiskohtaisemmin, kuinka Azure DevOps otetaan käyttöön, kuinka pipeline-prosessit rakennettiin ja miten ne toimivat käytännössä.

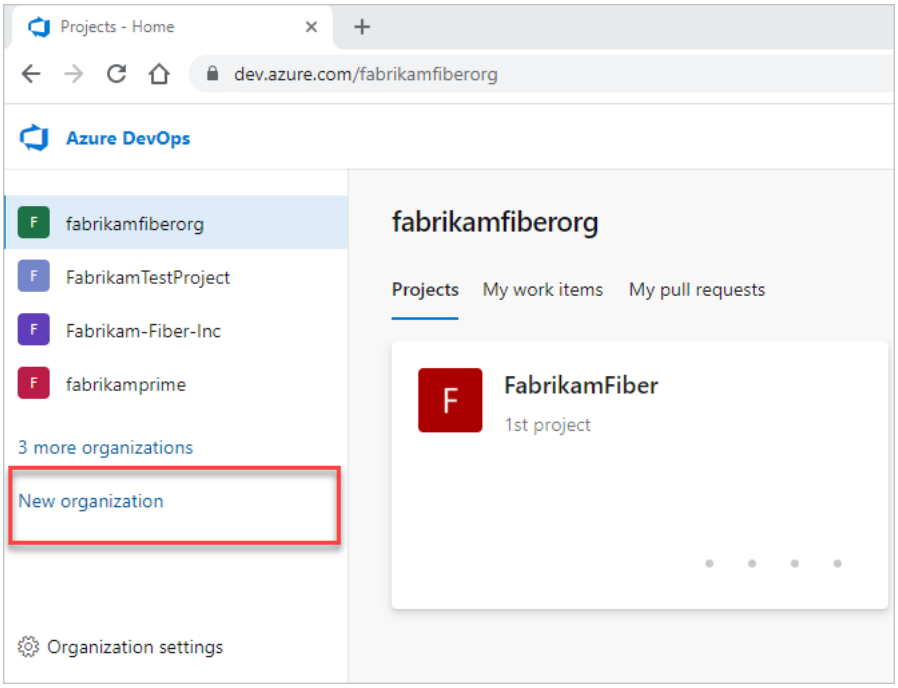
4.1 Azure DevOpsin käyttöönotto

Tässä luvussa käydään läpi Azure DevOpsin käyttöönottoon liittyvät vaiheet. Ensin luodaan organisaatio ja projekti, jonka jälkeen valitaan käytettävä repositorio.

4.1.1 Organisaation luonti

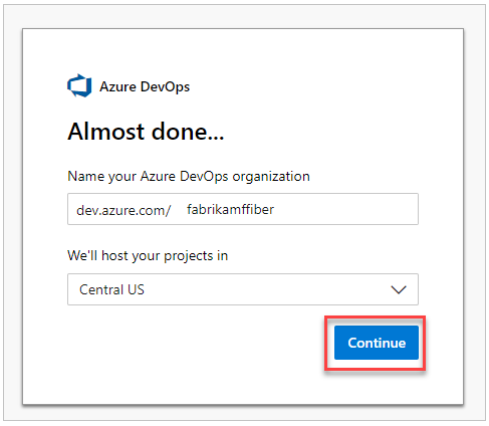
Ensimmäinen vaihe Azure DevOpsin käyttöönotossa on organisaation luominen. Organisaatio toimii pääasiallisena hallintayksikkönä, jonka alle kaikki projektit sijoittuvat [8].

- Kirjaudu sisään Azure DevOpsiin käyttämällä olemassaolevaa Microsoft tai Github tiliä
- Valitse *New organization* kuvan 5 mukaisesti



Kuva 5: Organisaation luonti

- Anna organisaatiolle nimi, valitse sen isännöintisijainti ja paina *Continue* kuvan 6 mukaisesti

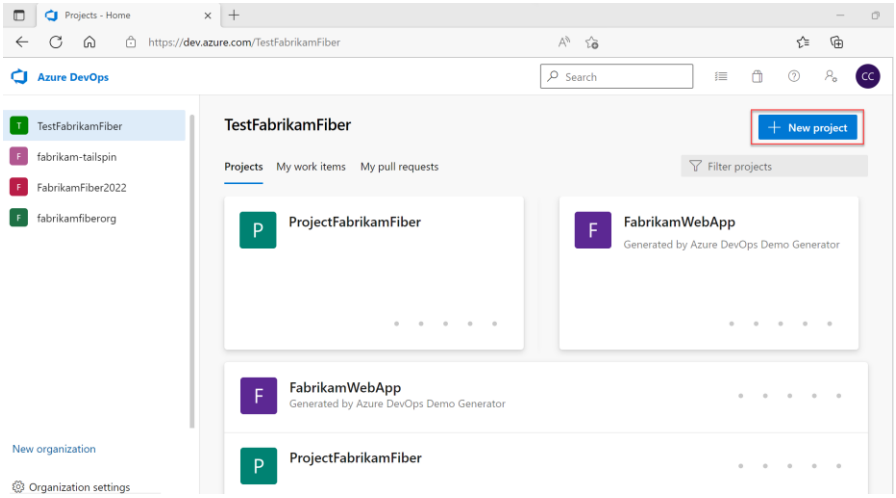


Kuva 6: Organisaation nimeäminen

4.1.2 Projektin luonti

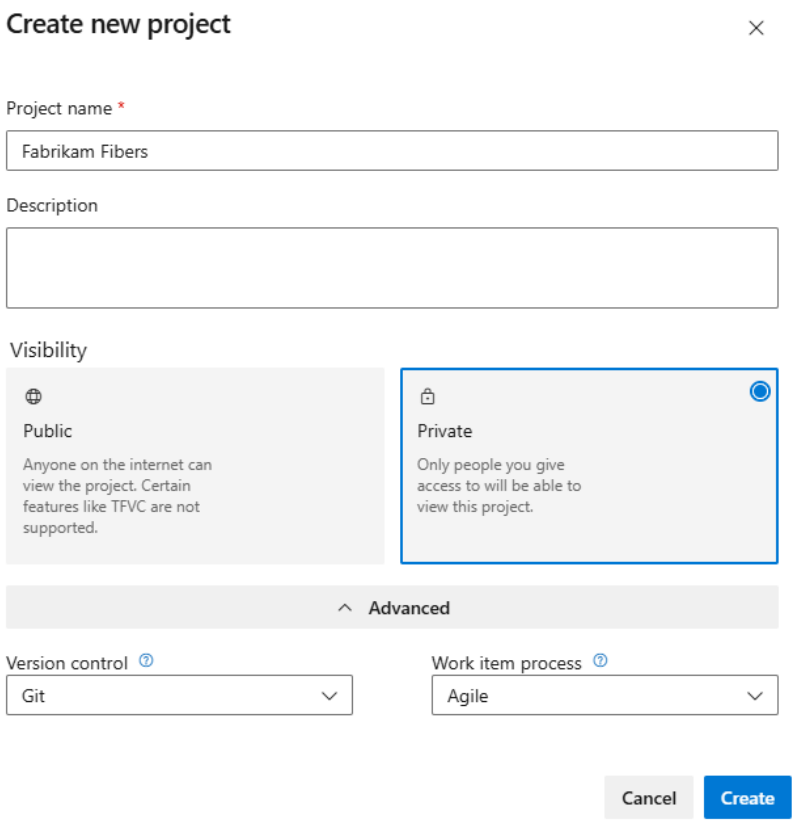
Organisaation luomisen jälkeen luodaan projekti, jonka sisälle kaikki kehitystyöhön liittyvät toiminnot keskitetään [9].

- Kirjaudu sisään organisaatioon
- Valitse *New project* oikeasta laidasta kuvan 7 mukaisesti



Kuva 7: Uuden projektin luonti

- Täytä lomakkeen tiedot (projektin nimi, kuvaus, näkyvyys ja versionhallinta) kuvan 8 mukaisesti



Kuva 8: Uuden projektin luonti 2

- Paina *Create*

4.1.3 Repositorion valinta

Azure DevOps tukee sekä omia että ulkoisia repositorioita, kuten GitHubia. Tässä projektissa päätettiin kuitenkin käyttää Azure DevOpsin omaa repositoriota, sillä se tarjoaa paremman integraation DevOps-palveluiden kanssa. Etuna on helpompi julkaisuputken luonti sekä mahdollisuus hyödyntää Azure Boardsia repositorion kanssa liittämällä taskeja suoraan repositoriossa oleviin tiedostoihin.

5 Microsoft Azure

Azure on vuonna 2010 julkaistu Microsoftin tarjoama pilvipalvelu. Se tarjoaa yrityksille maailmanlaajuisen verkon datakeskuksia, joita hallitsee ja ylläpitää Microsoft. Palvelun tavoite on tarjota asiakkaille laajasti erilaisia työkaluja aina infrastruktuurista tekoälyyn ja sovelluspalveluihin [10]. Tässä opinnäytetyössä Azuren palveluista käytetään App Serviceä ja Static Web Appsia.

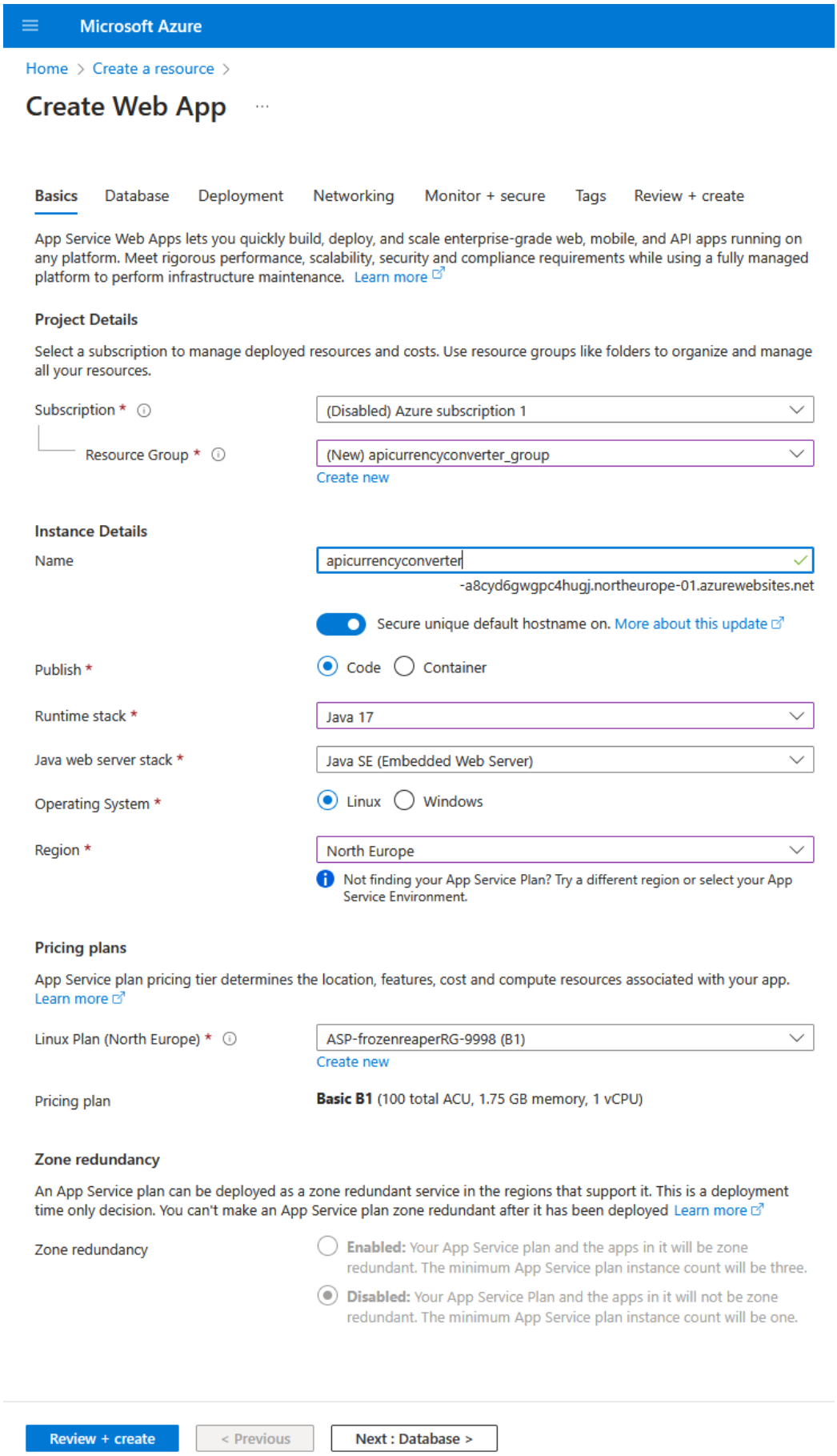
5.1 App Servicen luonti

Projektin backend-sovellusta varten tuli luoda palvelu, johon se voidaan julkaista. Tätä varten käytettiin Azure App Serviceä, joka on HTTP-pohjainen pilvipalvelu web-sovellusten, REST-rajapintojen ja mobiilisovellusten backendien isännöintiin. Azure App Service tukee useita ohjelmointikieliä, kuten .NET, .NET Core, Java, Node.js, Python ja PHP [11]. Koska tämän projektin backend on toteutettu Java Spring Bootilla, valittiin palvelun luonnin yhteydessä käyttöön Java 17 Runtime Stack.

App Service luotiin seuraavasti, mikä on nähtävissä kuvassa 9:

- Siirrytään Microsoft Azuren etusivulle ja valitaan "Create a resource"
- Hakukentän avulla etsitään Web App -palvelu
- Valitaan käytettävä Azure Subscription ja joko luodaan tai valitaan olemassa oleva Resource Group
- Nimetään sovellus
- Valitaan Runtime Stackiksi Java 17 yhteensopivuuden vuoksi
- Määritellään maantieteellinen sijainti, jossa sovellus isännöidään
- Valitaan tarvittava laskentateho.

Tämän jälkeen App Service on valmis käytettäväksi ja siihen voidaan julkaista sovellus [12].


Microsoft Azure
Home > Create a resource >
Create Web App
Basics Database Deployment Networking Monitor + secure Tags Review + create
App Service Web Apps lets you quickly build, deploy, and scale enterprise-grade web, mobile, and API apps running on any platform. Meet rigorous performance, scalability, security and compliance requirements while using a fully managed platform to perform infrastructure maintenance. Learn more
Project Details
Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription * (Disabled) Azure subscription 1
Resource Group * (New) apicurrencyconverter_group
Create new
Instance Details
Name apicurrencyconverter
-a8cyd6gwgpc4hugj.northeurope-01.azurewebsites.net
Secure unique default hostname on. More about this update
Publish * Code Container
Runtime stack * Java 17
Java web server stack * Java SE (Embedded Web Server)
Operating System * Linux Windows
Region * North Europe
Not finding your App Service Plan? Try a different region or select your App Service Environment.
Pricing plans
App Service plan pricing tier determines the location, features, cost and compute resources associated with your app.
Learn more
Linux Plan (North Europe) * ASP-frozenreaperRG-9998 (B1)
Create new
Pricing plan Basic B1 (100 total ACU, 1.75 GB memory, 1 vCPU)
Zone redundancy
An App Service plan can be deployed as a zone redundant service in the regions that support it. This is a deployment time only decision. You can't make an App Service plan zone redundant after it has been deployed Learn more
Zone redundancy
Enabled: Your App Service plan and the apps in it will be zone redundant. The minimum App Service plan instance count will be three.
Disabled: Your App Service Plan and the apps in it will not be zone redundant. The minimum App Service plan instance count will be one.
Review + create
< Previous
Next : Database >

Kuva 9: Web Appin luonti

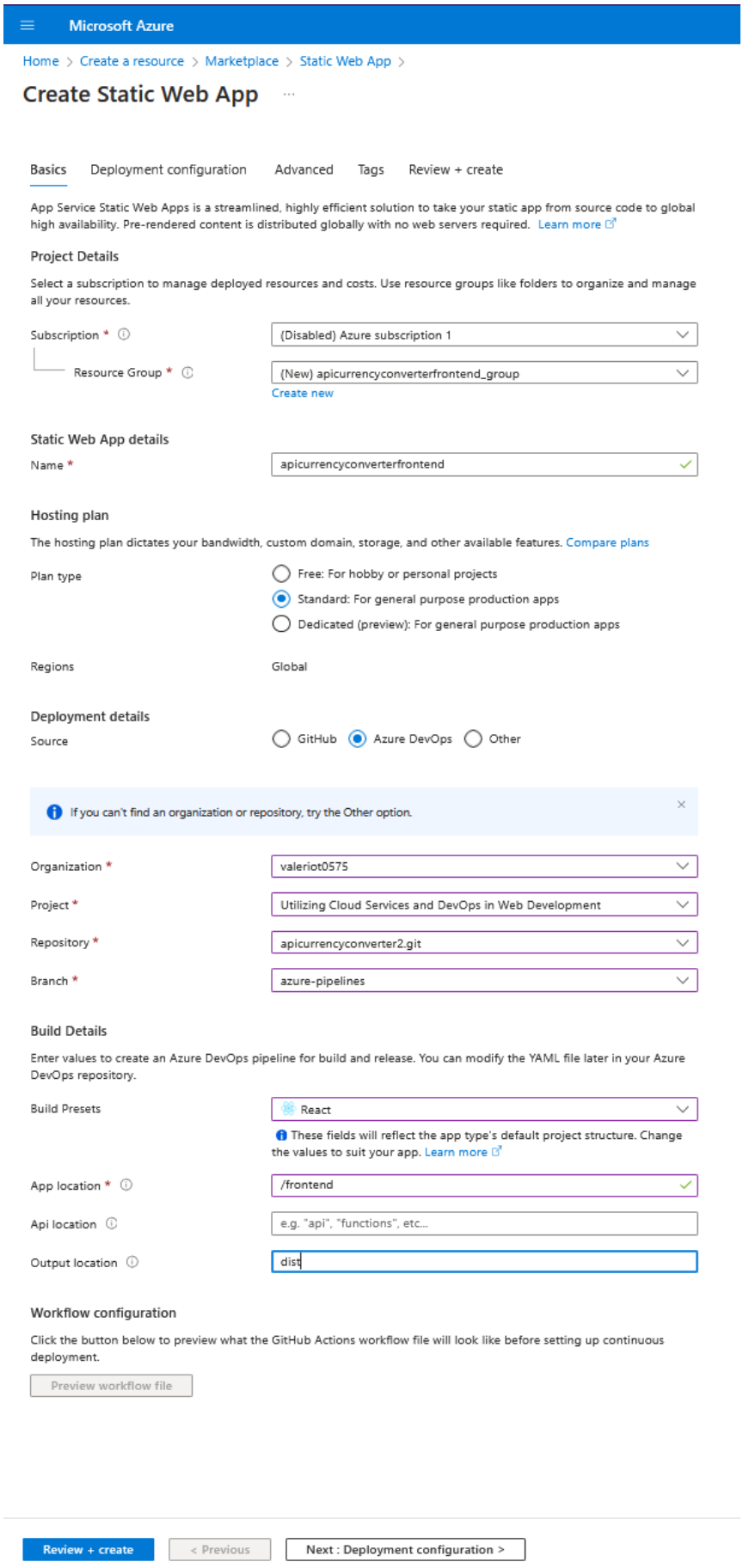
5.2 Static Web Appin luonti

Frontend-sovellusta varten luotiin Azure Static Web App, joka soveltuu erityisesti staattisten web-sivustojen, kuten React-sovellusten, isännöintiin. Luontiprosessi muistuttaa pitkälti App Servicen käyttöönottoa.

Toimenpiteet Static Web Appin luomiseksi, mikä näkyy kuvassa 10:

- Valitaan tilaus (Subscription) ja Resource Group
- Annetaan nimi palvelulle
- Valitaan isännöintisuunnitelma (Free tai Standard)
- Valitaan lähdekoodin sijainti. Vaihtoehtoina ovat esimerkiksi GitHub tai Azure DevOps. Tässä projektissa käytettiin Azure DevOpsia, joten se valittiin lähteeksi
- Valitaan oikea organisaatio, projekti, repositorio ja branch Azure DevOpsista.

Tämän jälkeen staattinen web-sovellus on luotu ja valmis käyttöä varten [13].


Microsoft Azure
Home > Create a resource > Marketplace > Static Web App >
Create Static Web App
Basics
Deployment configuration
Advanced
Tags
Review + create
App Service Static Web Apps is a streamlined, highly efficient solution to take your static app from source code to global high availability. Pre-rendered content is distributed globally with no web servers required. Learn more
Project Details
Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.
Subscription *
(Disabled) Azure subscription 1
Resource Group *
(New) apicurrencyconverterfrontend_group
Create new
Static Web App details
Name *
apicurrencyconverterfrontend
Hosting plan
The hosting plan dictates your bandwidth, custom domain, storage, and other available features. Compare plans
Plan type
Free: For hobby or personal projects
Standard: For general purpose production apps
Dedicated (preview): For general purpose production apps
Regions
Global
Deployment details
Source
GitHub
Azure DevOps
Other
If you can't find an organization or repository, try the Other option.
Organization *
valeriot0575
Project *
Utilizing Cloud Services and DevOps in Web Development
Repository *
apicurrencyconverter2.git
Branch *
azure-pipelines
Build Details
Enter values to create an Azure DevOps pipeline for build and release. You can modify the YAML file later in your Azure DevOps repository.
Build Presets
React
These fields will reflect the app type's default project structure. Change the values to suit your app. Learn more
App location *
/frontend
Api location
e.g. "api", "functions", etc...
Output location
dist
Workflow configuration
Click the button below to preview what the GitHub Actions workflow file will look like before setting up continuous deployment.
Preview workflow file
Review + create
< Previous
Next : Deployment configuration >

Kuva 10: Static Web Appin luonti

Tässä vaiheessa kaikki projektissa tarvittavat palvelut on luotu.

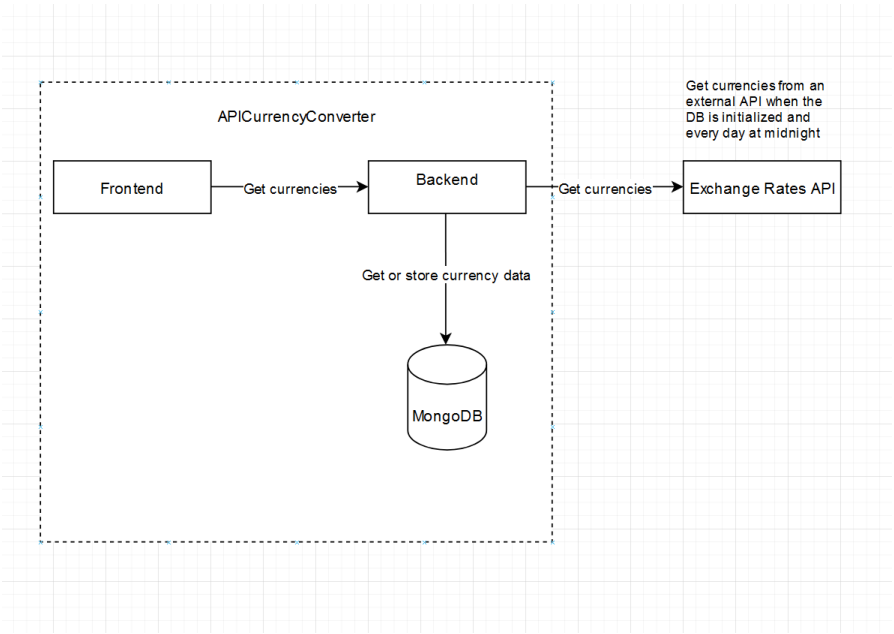
5.3 Azure DevOpsin ja Microsoft Azuren yhdistäminen

Vaikka Azure DevOpsiin on luotu projekti ja Microsoft Azureen tarvittavat palvelut, ne eivät oletuksena ole yhteydessä toisiinsa. Julkaisuprosessin onnistumiseksi on tärkeää luoda yhteys Azure DevOpsin ja Microsoft Azuren välille.Tämä toteutetaan luomalla Service Connection Azure DevOpsin projektin asetuksista. Yhteyden luonnin yhteydessä valitaan oikea Azure Subscription, jonka jälkeen DevOps saa tarvittavat käyttöoikeudet resurssien hallintaan [14]. Yhteyden luonnin jälkeen Service Connectionille muodostuu yksilöllinen ID, joka määritellään backendin pipelinen muuttujana. Tämä mahdollistaa backend-sovelluksen julkaisun App Serviceen. Static Web Appin tapauksessa julkaisu ei tapahdu Service Connectionin kautta, vaan siihen käytetään erillistä deployment tokenia, joka määritellään frontendin pipelinessä. Token toimii todennustapana ja mahdollistaa turvallisen ja hallitun käyttöönoton.

6 Esimerkkisovellus hyödyntäen Azure DevOpsia ja automatisoitua CI/CD-putkea

Opinnäytetyön esimerkkiprojektina toimii itse toteuttamani API Currency Converter -niminen web-sovellus. Kyseessä on valuutanvaihtosovellus, jossa on erillinen frontend ja backend. Backend on toteutettu käyttäen Java Spring Bootia, ja se vastaa valuutanvaihtologiikasta sekä yhteyksistä ulkoisiin palveluihin ja tietokantaan. Sovellus hakee päivittäin Exchange Rates API -rajapinnasta uusimmat valuuttakurssit, jotka tallennetaan MongoDB-tietokantaan. Backend tarjoaa kaksi REST-rajapintaa (endpointia): toinen palauttaa saatavilla olevat valuutat sekä vaihtoarvot, ja toinen suorittaa varsinaisen valuutanvaihdon. Frontend on toteutettu Reactilla, ja se tarjoaa käyttöliittymän valuuttojen vaihtamiseen. Frontend ei sisällä omaa toimintalogiikkaa, vaan se kommunikoi suoraan backendin kanssa hakien sieltä tarvittavat tiedot, kuten valuuttojen nimet ja

viimeisimmät vaihtoarvot. Toisin sanoen frontend ainoastaan esittää backendin palauttamat vastaukset. Valitsin tämän sovelluksen opinnäytetyöhöni, koska sitä pystyy käyttämään esimerkkinä siitä, miten DevOpsin automatisointia ja pilvipalveluita voidaan hyödyntää web-kehityksessä. Sovellus noudattaa monia tyypillisen web-sovelluksen rakenteita ja käytäntöjä ja sisältää kaikki olennaiset osa-alueet, joita tarvitaan kokonaisvaltaisen CI/CD-putken rakentamiseen. Kuvassa 11 on sovelluksen rakennekaavio.



Kuva 11: Api Currency Converterin kaavio

6.1 Julkaisuputket

Projektissa luotiin frontendille ja backendille omat julkaisuputket (pipeline), joissa sovellus rakennetaan, testataan ja julkaistaan. Tässä luvussa esitellään julkaisuputket ja kerrotaan mitä niissä tapahtuu ja miksi. Julkaisuputket on toteutettu Azure DevOpsin avulla käyttäen YAML-muotoisia määritystiedostoja, joihin kuvataan vaiheittain buildaus-, testaus- ja julkaisuprosessit. Azure DevOps tarjoaa mahdollisuuden luoda julkaisuputki lähes täysin automaattisesti valmiiden mallien ja käyttöliittymän avulla, mutta yhtä lailla käyttäjä voi rakentaa koko putken käsin kirjoittamalla YAML-tiedostoon. Julkaisuputki käynnistyy, kun sille määritellyt ehdot täyttyvät (esimerkiksi muutos tietyssä kansiossa tai

haarassa), minkä jälkeen suoritetaan yksi tai useampi job eli tehtäväkokonaisuus. Jokainen job sisältää sarjan task-vaiheita, kuten:

- Build – lähdekoodin kääntäminen ja sovelluksen kokoaminen
- Test – automaattisten yksikkö- tai integraatiotestien ajaminen
- Deploy – valmiin sovelluksen julkaiseminen

Tämä rakenne tekee julkaisuputkista modulaarisia, uudelleenkäytettäviä ja selkeitä. Jokainen vaihe voidaan myös erikseen konfiguroida ehdolliseksi tai riippuvaiseksi aiemmista vaiheista, mikä mahdollistaa tarkasti hallitun julkaisuputken.

6.2 CI/CD:n merkitys

Continuous Integration (CI) ja Continuous Delivery (CD) on nykyaikaisen ohjelmistokehityksen yksi keskeinen toimintatapa. Se mahdollistaa nopean ja turvallisen ohjelmiston kehityksen ja julkaisun automaation avulla. CI/CD:n avulla voidaan:

- Automaattisesti testata koodi aina muutoksen yhteydessä, mikä parantaa koodin laatua ja estää virheiden päätymistä tuotantoon.
- Rakentaa ja julkaista sovellus automaattisesti ilman manuaalisia vaiheita, mikä nopeuttaa kehityssyklejä.
- Parantaa yhteistyötä tiimissä, koska muutokset integroidaan jatkuvasti yhteiseen koodipohjaan.
- Vähentää inhimillisiä virheitä, kun sovelluksen rakennus- ja julkaisuprosessit tehdään automaattisesti.

CI/CD tukee DevOpsin filosofiaa, jossa kehitys ja operointi ovat saumattomasti yhteydessä ja ohjelmisto voidaan julkaista milloin tahansa, luotettavasti ja hallitusti.

6.3 Frontendin julkaisuputki

Frontendin julkaisuputki on luotu automaattiisesti Microsoft Azuren toimesta, kun Static Web Apps luotiin. Tämä julkaisuputki mahdollistaa React-sovelluksen rakentamisen ja julkaisemisen Azure Static Web App -palveluun käyttämällä

Azure-portaalista saatavaa julkaisutunnusta (deployment token). Peruskonfiguraation lisäksi julkaisuputkeen on lisätty kehitystyötä tukevia ominaisuuksia: muutosten tarkkailu pelkästään frontend-kansiosta, testien suorittaminen Playwrightilla ja testiraporttien julkaisu Azure DevOpsiin.

6.3.1 Laukaisu ja ehdot

Julkaisuputki käynnistyy aina, kun azure-pipelines-haaraan tehdään muutoksia, mutta vain jos muutos koskee frontend -kansiota. Tämä varmistaa, että frontendin julkaisuputki aktivoituu vain tarpeen vaatiessa. Esimerkkikoodissa 1 näkyy, miten laukaisimet määritellään YAML-tiedostoon.

```
trigger:
  branches:
    include:
      - azure-pipelines
  paths:
    include:
      - frontend/*
```

Esimerkkikoodi 1. Frontend putken trigger eli laukaisija

6.3.2 Rakennus- ja julkaisuvaihe

Ensimmäisessä vaiheessa julkaisuputki rakentaa React-sovelluksen ja julkaisee sen Azure Static Web App -palveluun. Tätä varten käytetään AzureStaticWebApp@0-taskia, joka hyödyntää ympäristömuuttujana tallennettua julkaisutunnusta. Määritellyt parametrit kertovat, missä frontendin lähdekoodi sijaitsee (frontend kansio) ja mistä rakennettu sisältö löytyy (dist kansio). Nämä on esitetty esimerkkikoodissa 2.

```
  - job: build_and_deploy_job
    displayName: Build and Deploy Job
    condition: or(eq(variables['Build.Reason'], 'Manual'),or(eq(varia-
bles['Build.Reason'], 'PullRequest'),eq(variables['Build.Rea-
son'], 'IndividualCI')))
    pool:
      vmImage: ubuntu-latest
    variables:
      - group: Azure-Static-Web-Apps-proud-pebble-098126503-variable-
group
    steps:
      - checkout: self
        submodules: true
      - task: AzureStaticWebApp@0
        inputs:
          azure_static_web_apps_api_token: $(AZ-
URE_STATIC_WEB_APPS_API_TOKEN_PROUD_PEBBLE_098126503)
          app_location: "/frontend" # App source code path
          api_location: "" # Api source code path - optional
          output_location: "dist" # Built app content directory
```

Esimerkkikoodi 2. Frontend putken rakennus ja julkaisu

Tämän vaiheen onnistunut suoritus on edellytys seuraavalle vaiheelle eli testaukselle.

6.3.3 Playwright-testivaihe

Toisessa vaiheessa suoritetaan frontendin testit Playwrightilla. Tämä vaihe käynnistyy vain, jos rakennus ja julkaisu onnistuivat. Esimerkkikoodissa 3 näkyy, miten Playwright testit ajetaan. Testaus toteutetaan seuraavasti:

- Asennetaan Node.js versiolla 18, jota tarvitaan Playwrightin ja riippuvuuksien ajamiseen.
- Ajetaan npm ci, joka asentaa projektin riippuvuudet package-lock.json -tiedoston mukaisesti.
- npx playwright install --with-deps varmistaa, että tarvittavat selaimet ja riippuvuudet löytyvät testejä varten.
- npx playwright test suorittaa kaikki määritellyt testit ja tuottaa JUnit-muotoisen tulostiedoston sekä HTML-raportin.
- Testien tulokset julkaistaan Azure DevOpsin testitulosten näkymään JUnit-muodossa. Lisäksi HTML-muotoinen raportti Playwrightin omasta testiraportista julkaistaan pipeline-artifaktina, jota voi tarkastella erikseen.

```
- job: run_playwright_tests
    displayName: Run Playwright Tests
    dependsOn: build_and_deploy_job
    condition: succeeded()
    pool:
      vmImage: ubuntu-latest
    steps:
      - task: UseNode@1
        inputs:
          version: "18"
        displayName: Install Node.js
      - script: npm ci
        workingDirectory: frontend
        displayName: Install Dependencies
      - script: npx playwright install --with-deps
        workingDirectory: frontend
        displayName: Install Playwright Browsers
      - script: npx playwright test
        workingDirectory: frontend
        displayName: Run Playwright Tests
        env:
          CI: "true"
      - task: PublishTestResults@2
        displayName: Publish JUnit Test Results
        inputs:
          searchFolder: "frontend/test-results"
          testResultsFormat: "JUnit"
          testResultsFiles: "e2e-junit-results.xml"
          mergeTestResults: true
          failTaskOnFailedTests: true
          testRunTitle: "Playwright End-To-End Tests"
        condition: succeededOrFailed()
      - task: PublishPipelineArtifact@1
        inputs:
          targetPath: frontend/playwright-report
          artifact: playwright-report
          publishLocation: pipeline
        displayName: Publish Playwright HTML Report
        condition: succeededOrFailed()
```

Esimerkkikoodi 3. Frontend putken Playwright testit

6.4 Backendin julkaisuputki

Backendin julkaisuputki on luotu suoraan Azure DevOpsin käyttöliittymästä, ja sen ensisijaisena tarkoituksena on rakentaa ja julkaista Java Spring Boot -sovellus Azure App Serviceen. Julkaisuputkeen on lisätty muutosten tarkkailu backend kansiosta, testien automaattinen suoritus ja koodikattavuuden raportointi.

6.4.1 Laukaisu ja ehdot

Pipeline käynnistyy aina, kun azure-pipelines haaraan tehdään muutoksia ainoastaan muutoksen koskiessa backend kansiota. Tämä rajaus estää frontend-puolen muutoksia turhaan käynnistämästä backendin julkaisuputkea. Esimerkkikoodissa 4 näkyy, miten laukaisimet määritellään YAML-tiedostoon. Se ei poikkea frontendin julkaisuputken määrityksistä.

```
trigger:
  branches:
    include:
      - azure-pipelines
  paths:
    include:
      - backend/* trigger:
```

Esimerkkikoodi 4. Backend putken trigger eli laukaisija

Lisäksi määritellään tarvittavat muuttujat esimerkkikoodin 5 mukaisesti, kuten käytettävä Azure-tilaus, web-sovelluksen nimi, ympäristön nimi sekä käytettävä ”build-agent” (ubuntu-latest).

```
variables:

  # Azure Resource Manager connection created during pipeline creation
  azureSubscription: '2eff4369-f879-4f83-912b-cb18202f1533'

  # Web app name
  webAppName: 'apicurrencyconverterbackend'

  # Environment name
  environmentName: 'apicurrencyconverterbackend'

  # Agent VM image name
  vmImageName: 'ubuntu-latest'
```

Esimerkkikoodi 5. Backend putken muuttujat

6.4.2 Rakennusvaihe

Build-vaiheen tavoitteena on kääntää projekti Mavenilla, suorittaa yksikkötestit ja tuottaa JUnit-tulosraportit, generoida koodikattavuusraportti Jacocolla,

julkaista koodikattavuusraportti Azure DevOpsiin sekä siirtää käännetty JAR-paketti artefaktiksi deploy-vaihetta varten. Tämä on määritelty esimerkkikoodissa 6. Tärkeimmät osavaiheet buildissa:

- **Maven Package**: Ajaa clean package -komennon, joka kääntää projektin ja ajaa testit
- **Coverage Reportin luonti**: verify-komento ajetaan Jacoco-pluginin avulla, joka tuottaa kattavuusraportin
- **Koodikattavuusraportin julkaisu**: Jacocon XML-raportti julkaistaan Azure DevOpsin analyysinäkymään
- **Artefaktin luonti:** Valmis JAR-tiedosto siirretään artifact staging directoryyn ja julkaistaan pipeline-artefaktiksi, jota käytetään myöhemmin deploy-vaiheessa.

```
stages:
- stage: Build
  displayName: Build stage
  jobs:
  - job: MavenPackageAndPublishArtifacts
    displayName: Maven Package and Publish Artifacts
    pool:
      vmImage: $(vmImageName)

    steps:
    - task: Maven@4
      displayName: 'Maven Package'
      inputs:
        mavenPomFile: 'backend/pom.xml'
        mavenOptions: '-Xmx3072m'
        javaHomeOption: 'JDKVersion'
        jdkVersionOption: '1.17'
        jdkArchitectureOption: 'x64'
        publishJUnitResults: true
        testResultsFiles: 'backend**/surefire-reports/TEST-*.xml'
        goals: 'clean package'
    - task: Maven@4
      displayName: 'Maven - Generate Coverage Report'
      inputs:
        mavenPomFile: 'backend/pom.xml'
        goals: 'verify'
    - task: PublishCodeCoverageResults@2
      displayName: 'Publish Code Coverage Results'
      inputs:
        summaryFileLocation: 'backend/target/site/jacoco/jacoco.xml'

    - task: CopyFiles@2
      displayName: 'Copy Files to artifact staging directory'
      inputs:
        SourceFolder: '$(System.DefaultWorkingDirectory)/backend'
        Contents: '**/target/*.?(war|jar)'
        TargetFolder: $(Build.ArtifactStagingDirectory)

    - upload: $(Build.ArtifactStagingDirectory)
      artifact: drop
```

Esimerkkikoodi 6. Backend putken rakennusvaihe

6.4.3 Julkaisuvaihe

Julkaisuvaihe suoritetaan vain, jos rakentaminen onnistuu. Tässä vaiheessa julkaistaan aiemmin tuotettu artefakti Azure App Serviceen Linux-ympäristössä. Julkaisu toteutetaan AzureWebApp@1-taskin avulla, ja käytössä on muuttujissa määritellyt Azure-resurssit kuten azureSubscription ja appName. Esimerkkikoodissa 7 näkyy YAML-tiedoston julkaisuvaihe.

```
- stage: Deploy
  displayName: Deploy stage
  dependsOn: Build
  condition: succeeded()
  jobs:
  - deployment: DeployLinuxWebApp
    displayName: Deploy Linux Web App
    environment: $(environmentName)
    pool:
      vmImage: $(vmImageName)
    strategy:
      runOnce:
        deploy:
          steps:
          - task: AzureWebApp@1
            displayName: 'Azure Web App Deploy: apicurrencyconverter-
backend'
            inputs:
              azureSubscription: $(azureSubscription)
              appType: webAppLinux
              appName: $(webAppName)
              package: '$(Pipeline.Workspace)/drop/**/tar-
get/*.?(war|jar)'
```

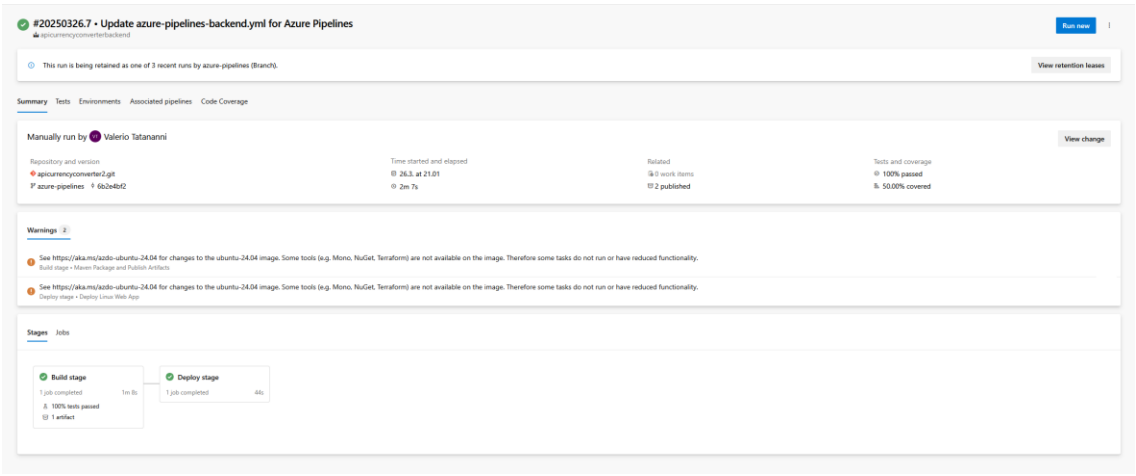
Esimerkkikoodi 7. Backend putken julkaisuvaihe

Julkaisuvaiheen avulla varmistetaan, että valmis backend-sovellus siirtyy automaattisesti Azureen ilman manuaalista väliintuloa heti rakentamisen jälkeen.

6.5 Julkaisuputkien testaus

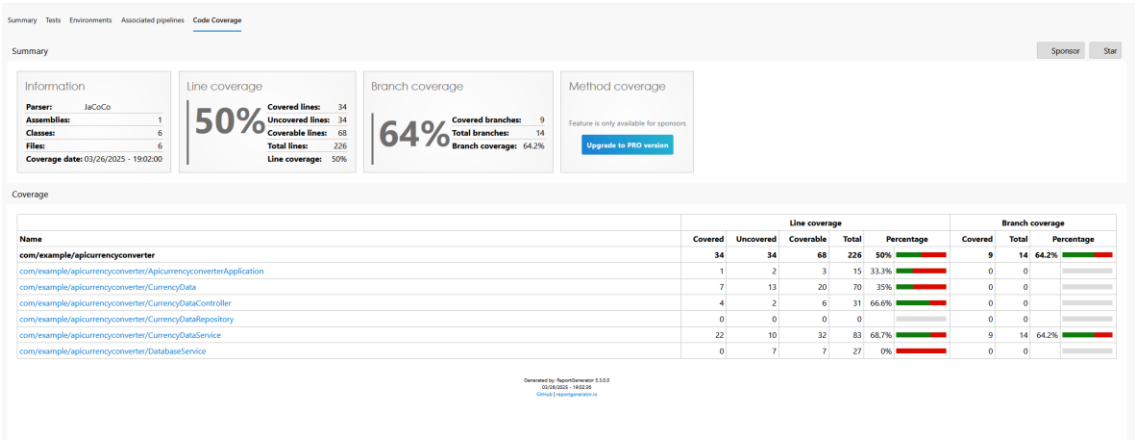
Lopuksi varmistetaan, että rakennetut julkaisuputket toimivat odotetulla tavalla. Ensimmäisenä testattiin backendin putki, sillä frontend ei voi toimia ilman taustajärjestelmää. Azure DevOpsissa julkaisuputki voidaan käynnistää suoraan painamalla "Run pipeline" -painiketta, mutta koska molempiin julkaisuputkiin on asetettu oman kansion tarkkailu käynnistyvät ne myös tekemällä muutoksia kansioiden sisällä.

Kuten kuvasta 12 ja 13 nähdään, toimii backendin julkaisuputki odotetusti. Koko prosessi kestää 2 minuuttia ja 7 sekuntia, kaikki testit suoritetaan onnistuneesti, ja testikattavuusraportti generoituu oikein. Sovellus julkaistaan Microsoft Azuren App Serviceen ja toimii suunnitellusti.



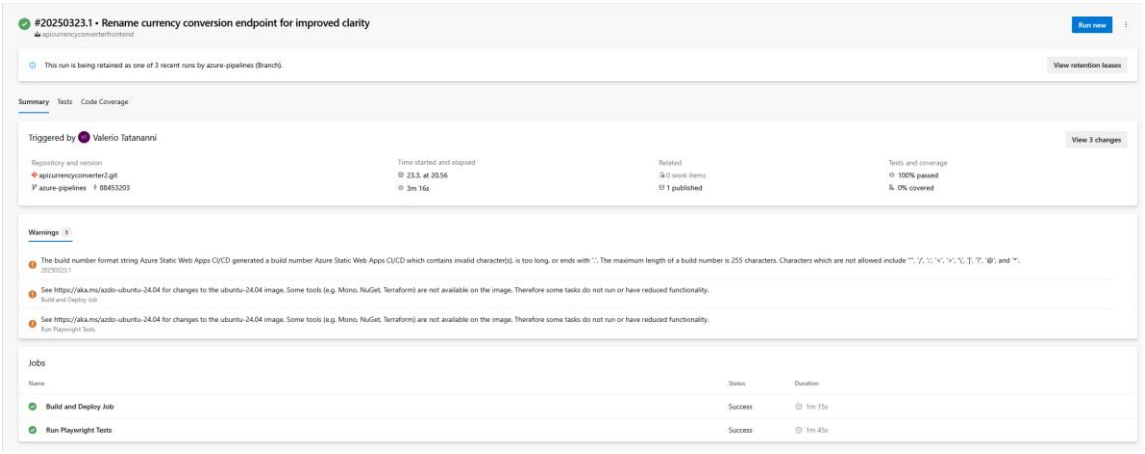
Kuva 12: Backendin julkaisu

Julkaisusivun näkymää voi vaihtaa klikkaamalla välilehtiä tai julkaisuputken eri vaiheita.



Kuva 13: Backend testikattavuusraportti

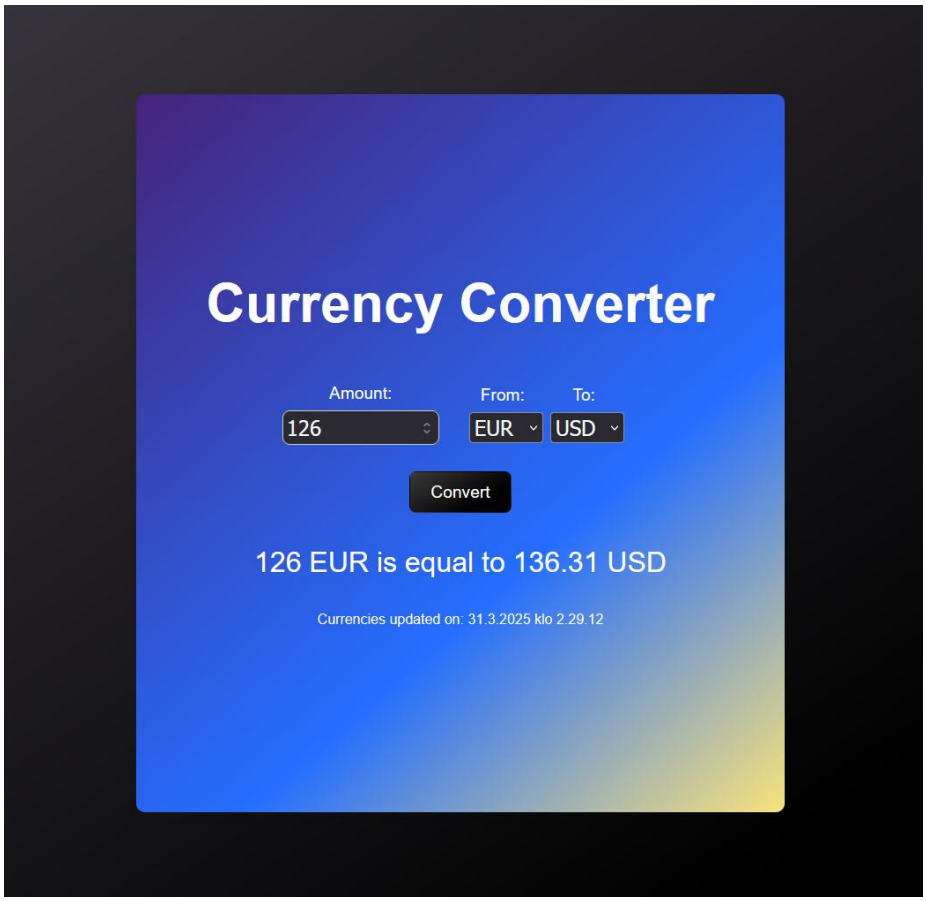
Frontendin julkaisu toimii sekin odotetusti, niin kuin kuvassa 14 näkyy. Julkaisuprosessi kestää 3 minuuttia ja 16 sekuntia, ja kaikki testit menevät läpi onnistuneesti. Frontend julkaistaan Azuren Static Web Appsiin ja on käyttövalmis.



Kuva 14: Frontend julkaisu

Julkaisuja on mahdollista tarkastella reaaliaikaisesti tai jälkikäteen Azure DevOpsin käyttöliittymän kautta. Näkymistä on myös mahdollista nähdä mahdolliset varoitukset tai virheet, mikä tekee siitä erinomaisen työkalun julkaisujen seurantaan ja analysointiin. Jokaisesta työstä (job) ja tehtävästä (task) on saatavilla yksityiskohtaista tietoa, kuten suoritusajat ja mahdolliset poikkeamat. Kuten

kuvasta 15 näkyy, sovellus toimii onnistuneen julkaisun jälkeen odotetusti ja on käytettävissä.



Kuva 15: Api Currency Converter

Julkaisuputkien testaamisen jälkeen on todettu, että ne toimivat odotetusti. Seuraavaksi hieman pohdintaa siitä, miten julkaisunäkymää hyödynnetään.

6.5.1 Julkaisunäkymän hyödyntäminen käytännössä

Azure DevOpsin julkaisunäkymä (run) mahdollistaa kokonaiskuvan jokaisesta julkaisuputken ajosta, mitä on mahdollista hyödyntää monin eri tavoin.

Esimerkiksi vianetsintä on tehty helpoksi, sillä käyttäjä pystyy näkemään

epäonnistuneet työt (job) tai tehtävät (task) nopealla vilkaisulla, koska ne on visualisoitu eri väreillä (vihreä onnistunut, punainen epäonnistunut). Tämän lisäksi on mahdollista tarkastella myös lokeja tarkempien virheilmoitusten löytämiseksi.

Myös suorituskykyä on mahdollista arvioida, sillä käyttöliittymästä näkee kuinka paljon aikaa, on mennyt eri vaiheisiin. Näiden tietojen avulla voidaan vertailla eri julkaisuputkiajojen suoritusaikoja ja havaita, mitkä vaiheet ovat hitaita tai voisivat hyötyä optimoinnista. Esimerkiksi testien rinnakkaistaminen tai välimuistin hyödyntäminen voi nopeuttaa suoritusta.

Testitulosten seuranta kertoo testien kokonaismäärän, läpäisyprosentin ja mahdolliset epäonnistuneet testit. Testikattavuusraportista nähdään, kuinka laajasti koodia on testattu.

Lisäksi näkymästä näkee mihin ympäristöön syntynyt artefakti on julkaistu, ja mikä muutos versionhallinnassa (commit) on käynnistänyt julkaisuputken.

7 Yhteenveto

Tässä opinnäytetyössä tarkasteltiin pilvipalveluiden ja DevOpsin hyödyntämistä web-kehityksessä. Työssä osoitettiin, miten tyypillinen web-sovellus voidaan julkaista pilveen Azure DevOpsin avulla. Tulokset osoittavat, että pilvipalveluiden ja DevOpsin käyttö tuo merkittäviä etuja kehitysprosessiin: se nopeuttaa työvaiheita, parantaa laatua ja vähentää manuaalista työtä.

Opinnäytetyön lopputuloksena syntyi toimiva CI/CD-putki sekä frontendille että backendille. Julkaisuputki aktivoituu vain sovelluksen siihen osaan tehdyistä muutoksista, ajaa testit automaattisesti, tuottaa testitulokset ja julkaisee sovelluksen Azuren pilvipalveluun. Näin kehittäjä voi keskittyä itse ohjelmointiin, sillä testaus ja julkaisu tapahtuvat automaattisesti versionhallintaan tehtyjen muutosten jälkeen.

Tämän kaltaisessa ratkaisussa säästyy huomattavasti aikaa, ja sovelluskoodi saadaan tuotantoon nopeammin ja luotettavammin – DevOpsin parhaiden käytäntöjen mukaisesti.

Projektia voisi kuitenkin vielä kehittää esimerkiksi lisäämällä erillisiä testiympäristöjä. Tällä hetkellä frontendin putki julkaisee sovelluksen suoraan tuotantoympäristöön, minkä jälkeen ajetaan Playwright-testit. Parempi ratkaisu voisi olla julkaista sovellus ensin testiympäristöön ja siirtyä tuotantoon vasta, jos kaikki testit menevät onnistuneesti läpi. Tämä parantaisi julkaisuvarmuutta ja mahdollistaisi virheiden havaitsemisen aikaisemmassa vaiheessa. Toinen parannusidea olisi ottaa käyttöön Microsoft Azuren analysointityökaluja, sillä niiden avulla pystyy tarkkailemaan pilvipalvelujen toimivuutta, statistiikkaa, kulutusta ja niin edelleen. Isommassa projektissa tällaisilla työkaluilla voisi edelleen tehostaa eri osa alueiden tehokkuutta.

Lähteet

1. Before Cloud: The Era of On-Premises Infrastructure! Verkkodokumentti. 2023. https://medium.com/@nishakeswani/before-cloud-the-era-of-on-premises-infrastructure-5e2108ad738d
   Viitattu: 12.4.2025

2. What are Cloud Services? Verkkodokumentti. 2025. https://www.hpe.com/emea_europe/en/what-is/cloud-services.html
   Viitattu 12.4.2025

3. Azure IaaS, PaaS, SaaS – what's the difference? Verkkodokumentti. 2021. https://www.linkedin.com/pulse/azure-iaas-paas-saas-whats-difference-roxana-beatrice-stoenescu-/
   Viitattu 12.4.2025

4. What is DevOps? Verkkodokumentti. 2025. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-devops
   Viitattu 13.4.2025

5. A Brief History of DevOps and Its Impact on Software Development. Verkkodokumentti. 2023. https://everythingdevops.dev/a-brief-history-of-devops-and-its-impact-on-software-development/
   Viitattu 13.4.2025

6. What is Azure DevOps? Verkkodokumentti. 2024. <https://learn.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?toc=%2Fazure%2Fdevops%2Fget-started%2Ftoc.json&view=azure-devops
   Viitattu 14.4.2025

7. Overview of services. Verkkodokumentti. 2025. https://learn.microsoft.com/en-us/azure/devops/user-

guide/services?toc=%2Fazure%2Fdevops%2Fget-started%2Ftoc.json&view=azure-devops
Viitattu 14.4.2025

8. Create an organization or project collection. Verkkodokumentti. 2025. https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/create-organization?toc=%2Fazure%2Fdevops%2Fget-started%2Ftoc.json&view=azure-devops
Viitattu 14.4.2025

9. Create a project in Azure DevOps. Verkkodokumentti. 2025. https://learn.microsoft.com/en-us/azure/devops/organizations/projects/create-project?view=azure-devops&tabs=browser
Viitattu 14.4.2025

10. What is Azure? Verkkodokumentti. 2025. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure
Viitattu 14.4.2025

11. App Service overview. Verkkodokumentti. 2025. https://learn.microsoft.com/en-us/azure/app-service/overview
Viitatttu 14.4.2025

12. Step-by-step Guide to Creating Web Apps and Using App Service Plans in Azure. Verkkodokumentti. 2023. https://medium.com/@marvin-conejo/step-by-step-guide-to-creating-web-apps-and-using-app-service-plans-in-azure-a9a1d597d24f
Viitattu 14.4.2025

13. Building Your First Static Web App on Azure: A Step-by-Step Guide. Verkkodokumentti. 2023. https://dev.to/jeptoo/building-your-first-static-web-app-on-azure-a-step-by-step-guide-35d3
Viitattu 14.4.2025

14. Connect to Azure with an Azure Resource Manager service connection. Verkkodokumentti. 2024. https://learn.microsoft.com/en-us/azure/devops/pipelines/library/connect-to-azure?view=azure-devops Viitattu 14.4.2025