OULUN AMMATTIKORKEAKOULU

Kasperi Klaavo

WEB-SOVELLUKSEN KEHITTÄMINEN MICROSOFT AZURE -YHTEENSOPIVAKSI

WEB-SOVELLUKSEN KEHITTÄMINEN MICROSOFT AZURE -YHTEENSOPIVAKSI

Kasperi Klaavo
Opinnäytetyö
Kevät 2018
Tietotekniikan koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistokehitys

Tekijä: Kasperi Klaavo
Opinnäytetyön nimi: Web-sovelluksen kehittäminen Microsoft Azure -yhteensopivaksi
Työn ohjaaja: Jukka Jauhiainen
Työn valmistumislukukausi ja -vuosi: Kevät 2018 Sivumäärä: 57

Tämä työ pohjautuu yrityksen tarpeeseen julkaista web-sovellus pilvipalveluun perinteisen palvelimen sijaan. Työn tavoitteena on tutustua Microsoft Azure -pilvipalvelukokonaisuuteen ja selvittää alustan erojen vaikutusta sovellukseen. Selvitystyön lisäksi tavoitteena on kehittää sovellusta niiltä osin, että se toimisi hajautetussa palvelinjärjestelmässä.

Aineistona käytettiin pääosin Microsoftin verkkosivuilta löytyvää dokumentaatiota ja yleisissä asioissa muita luotettavia sivustoja. Työ sisälsi paljon myös käytännön tekemistä ja työn tuloksesta oivaltamista.

Tavoitteet täyttyivät ja sovellus täyttää nyt hajautetulla palvelinalustalla toimimisen vaatimukset. Kehitysversio saatiin julkaistua pilvipalveluun. Työn aikana opittua tietoa voidaan hyödyntää työn jälkeen samankaltaisissa projekteissa työskennellessä.

Asiasanat: Microsoft Azure, ASP.NET, pilvipalvelut, web-sovellukset

ABSTRACT

Oulu University of Applied Sciences
Information Technology, Software Engineering

Author: Kasperi Klaavo
Title of thesis: Web application development and migration to Microsoft Azure
Supervisor: Jukka Jauhiainen
Term and year when the thesis was submitted: Spring 2018 Number of pages: 57

This thesis is based on the need of the company to deploy web application to cloud hosting services instead of dedicated web server. The goal of thesis is to study and get familiar with Microsoft Azure cloud services and investigate the impact of the platform on the application. In addition to investigation, the goal is to develop the application to fill requirements to function in a distributed web-server system.

I mainly used documentation on Microsoft website as a study material and for more general cases I used other trustworthy websites. Thesis also included lots of working in practice and learning from output of work.

The goals were met, and application now fills the requirements of functioning in a distributed server system, and development version of application was deployed to cloud hosting service. The information learned during thesis is going to be useful for later similar projects.

Keywords: Microsoft Azure, ASP.NET, cloud computing, web applications

SISÄLLYS

LYHENTEET

.NET	.NET Framework, Microsoftin kehittämä monipuolinen ohjelmistokehys
AAD	Azure Active Directory, Windowsin AD:tä muistuttava pilvipalvelu
AD	Active Directory, Windows-toimialueen käyttäjätietokanta ja hakemistopalvelu
C#	C Sharp, alun perin Microsoftin kehittämä korkean tason ohjelmointikieli
CLR	Common Language Runtime, .NETin hallittu ajoympäristö
DTU	Database Transaction Unit, Azuren hinnoitteluun käytetty datansiirtoyksikkö
FTP	File Transfer Protocol, sovelluskerroksen protokolla tiedostojen siirtoon
HTTP	Hypertext Transfer Protocol, sovelluskerroksen protokolla hypertekstin siirtoon
HTTPS	HTTP Secure, TLS/SSL:llä salattu HTTP
IaaS	Infrastructure as a Service, infrastruktuuri palveluna -palvelumalli
IIS	Internet Information Services, Windowsin mukana tuleva web-palvelinohjelmisto
IP	Internet Protocol, verkkokerroksen tietoliikenneprotokolla
JSON	JavaScript Object Notation, standardoitu tiedostomuoto tiedonvälitykseen
NoSQL	Käsite, joka kuvaa SQL:stä poikkeavaa tietokantamallia. Tulee sanoista Not only SQL
Paas	Platform as a Service, alusta palveluna -palvelumalli
RDBMS	Relational Database Management System, yleiskäsite relaatiomallisten tietokantojen hallintajärjestelmälle
REST	Representational State Transfel, HTTP-protokollaan perustuva arkkitehtuurimalli
SaaS	Software as a Service, ohjelmisto palveluna -palvelumalli
SMB	Server Message Block, sovelluskerroksen protokolla tiedostojen siirtoon, suunniteltu myös verkkolaitteiden jakamiseen
SMTP	Simple Mail Transfer Protocol, sovelluskerroksen protokolla sähköpostiliikenteeseen
SQL	Structured Query Language, standardoitu tietokantakyselykieli
TLS/SSL	Transport Layer Security, aiemmin Secure Sockets Layer, salausprotokolla
T-SQL	Transact-SQL, Microsoftin ja Sybasen kehittämä SQL-laajennus
URL	Uniform resource locator, viittaus web-resurssiin, esimerkiksi web-sivuun
VM	Virtual Machine, yleiskäsite virtuaalitietokoneelle

1 JOHDANTO

Tämä opinnäytetyö pohjautuu yrityksen tarpeeseen siirtää pitkään kehityksessä ollut web-sovellus dedikoidusta eli täysin omasta fyysisestä palvelimesta pilvipalveluun. Koska käytössä on pääosin Microsoftin tekniikkaa, on luontevinta valita Microsoft Azure.

Työn voi jakaa karkeasti neljään osaan. Aluksi tutustutaan Microsoft Azure -pilvipalvelun eri ratkaisuihin ja perehdytään tarkemmin niistä oleellisimpiin. Sen jälkeen selvitetään pitkään kehityksessä olleen sovelluksen Azure-yhteensopivuus ja mahdolliset kehitystarpeet sekä kehitetään sovellusta niiltä osin, että se toimisi Azuren palveluiden päällä. Lopuksi julkaistaan testiversio sovelluksesta Azuren pilvipalveluun.

Selvitettäessä sovelluksen soveltuvuutta Azuren pilvipalveluihin on tarpeellista perehtyä myös yleisesti palvelumalleihin ja eri hosting- eli isännöintipalveluiden eroavaisuuksiin. Koska dedikoitu palvelin eroaa huomattavasti pilvipalveluista, eroavaisuuksien yksityiskohdat on hyvä tiedostaa sovellusta kehittäessä. Perusperiaatteet eivät kuitenkaan eroa eri palveluntarjoajien välillä. Palveluntarjoajien vertailu on rajattu pois opinnäytetyön sisällöstä.

Microsoft Azuren pilvipalvelut eivät rajoitu vain web-sovellusten hostaamiseen, vaan koko yrityksen infrastruktuuri on mahdollista rakentaa Azureen. Lisäksi Azuresta löytyy erittäin laajasti erilaisia älykkäitä pilvilaskentapalveluita ja rajapintoja, esimerkiksi koneoppimiseen. Keskityn tässä työssä kuitenkin web-sovelluksiin ja niihin liittyviin välttämättömiin palveluihin, kuten tietokanta-, tiedostontallennus- ja analytiikkapalveluihin.

2 KÄYTETYT TYÖKALUT JA TEKNOLOGIAT

Tässä osiossa käyn läpi oleellisimmat työkalut ja teknologiat, joita käytin projektin aikana. Yrityksen jatkokehitettävä sovellus on kehitetty C#:lla (1) ja ASP.NET Web Forms -tekniikalla (3), joka on osa .NET Framework -ohjelmistokehystä (2). Kehitysympäristönä oli luonnollisesti Visual Studio (5) tarvittavine lisäosineen. Versionhallintaan käytin Git-tekniikkaa (6) ja GitLab-palvelua (8). Palvelinpuolella projektiin vaikutti Windows Server -palvelinkäyttöjärjestelmä (9) ja Internet Information Services -web-palvelinohjelmisto (10), sillä sovellus on alun perin kehitetty toimivaksi vain tällä alustalla. Tietokantoihin käytin Microsoft SQL Server -palvelinta (15) ja T-SQL-kyselykieltä (20).

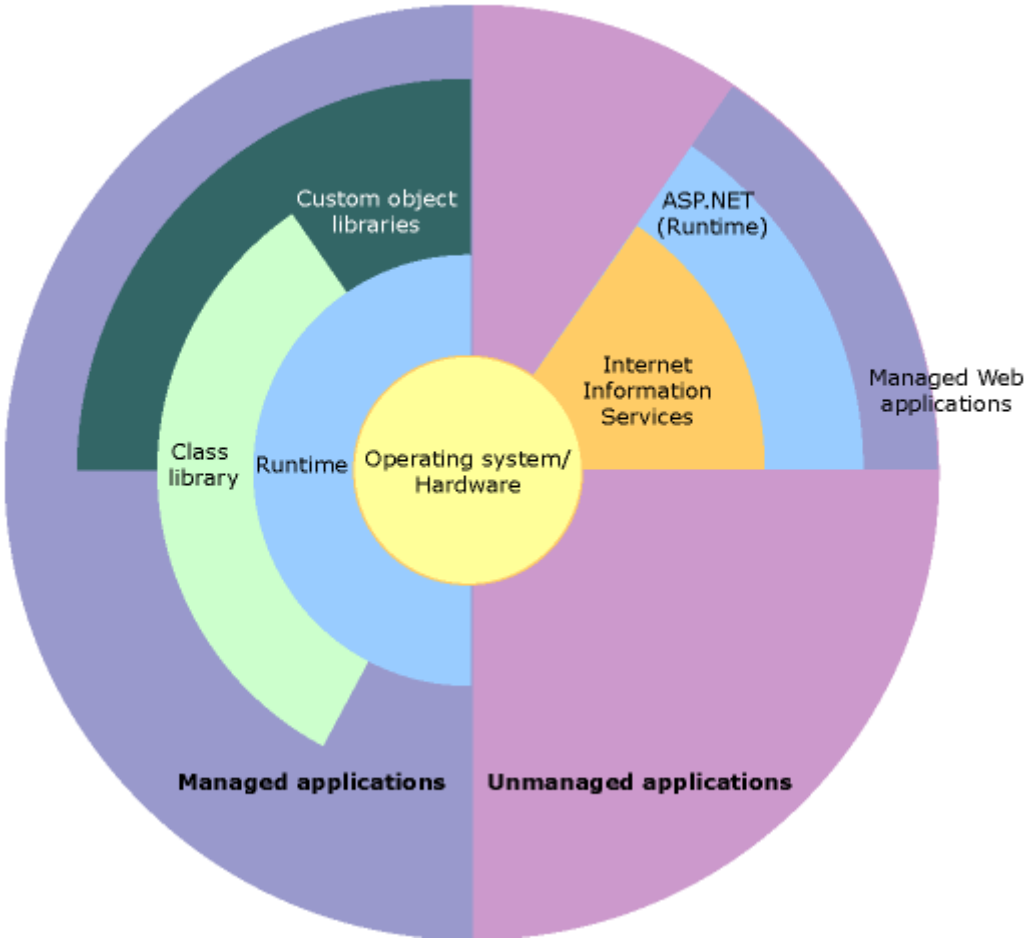
2.1 C#-ohjelmointikieli

C# (lausutaan C sharp) on olio-orientoitunut ja vahvasti sekä staattisesti tyypitetty ohjelmointikieli. Vahvasti tyypitetyissä ohjelmointikielissä kaikki muuttujat hyväksyvät vain etukäteen määriteltyä sisältöä. Staattisesti tyypitetyissä ohjelmointikielissä muuttujan tyyppi on määritelty ennen käännöstä, eikä se voi muuttua ajon aikana. C# on alun perin Microsoftin .NET-ohjelmistokehystä varten kehittämä kieli, mutta myöhemmin virallisesti standardoitu. C# muistuttaa C++- ja Java-ohjelmointikieliä sisältäen yhtä aikaa esimerkiksi tehokkaita muistinkäsittelyominaisuuksia, mutta myös hallitun ajoympäristön ominaisuuksia. Hallitun ajoympäristön ansiosta C# tukee korkean tason ominaisuuksia, kuten virheen käsittely (exception handling) ja roskien keruu (garbage collection). (1.)

2.2 .NET Framework -ohjelmistokehys

.NET Framework on Microsoftin kehittämä monipuolinen ohjelmistokehys, joka tukee kymmeniä eri ohjelmointikieliä, joista C# on käytetyin. Kuvassa 1 nähdään .NET Frameworkin arkkitehtuuri visualisoituna. .Net Framework koostuu laajasta luokkakirjastokokoelmasta ja hallitusta ajoympäristöstä Common language runtimesta (CLR). CLR mahdollistaa esimerkiksi muistinhallinnan, säiehallinnan sekä turvallisen tyypityksen. .NET Frameworkin luokkakirjasto sisältää yleisimmät olio-ohjelmoinnissa tarvittavat tyypit perustoimintoineen, esimerkiksi merkkijonon käsittelyn, data kokoelmat, tietokanta yhteydet ja tiedoston käsittelyn. Luokkakirjaston avulla voidaan kehittää useita erilaisia sovelluksia, kuten komentorivi, Windows Forms, Windows Presentation Foundation

(WPF), ASP.NET, Windows-palvelut, Windows Communication Foundation (WCF) ja Windows Workflow Foundation (WF). (1; 2.)



KUVA 1. Microsoftin sivuilla oleva .NET Frameworkin arkkitehtuurikuva (2).

2.2.1 ASP.NET-ohjelmistokehys

.NET Framework sisältää web-ohjelmistokehyksen ASP.NET, jolla kehitetään palvelimella suoritettavia web-sovelluksia. ASP.NET korvasi aiemmin kehitetyn ASP-teknologian ja toimii CLR:n päällä. ASP.NET sisältää ohjelmointimallin, kattavan ohjelmistoinfrastruktuurin ja tarvittavat palvelut web-sovellusten kehitykseen. ASP.NET toimii HTTP-protokollan päällä käyttäen standardimetodeja ja -käytäntöjä selaimen ja palvelimen väliseen kommunikointiin. ASP.NET sisältää useita erilaisia laajennettavia ja uudelleenkäytettäviä komponentteja interaktiivisten ja datapohjaisten web-sovellusten kehitykseen. ASP.NETin yleisimmät osat ovat ASP.NET Web Forms, ASP.NET MVC, ASP.NET Web Pages, ASP.NET Single Page Applications ja ASP.NET Web API. (2.)

2.2.2 ASP.NET Web Forms

ASP.NET Web Forms -tekniikka mahdollistaa dynaamisten web-sivujen kehityksen käyttäen HTML:ää, asiakaspuolen skriptejä ja palvelinpuolen koodia. Sivut käännetään ja suoritetaan palvelimella ja selaimeen palautuu dynaamisesti luotu HTML-sivu. Sivuilla voidaan käyttää uudelleenkäytettäviä valmiita tai mukautettuja komponentteja. Microsoft Visual Studiosta löytyy graafiset työkalut Web Forms -sivujen ja -komponenttien kehitykseen. Komponenttien ominaisuuksia tai metodeja voi nopeasti muuttaa ja testata. Web Forms -luokkakirjastoissa ja -projektipohjissa on valmiina monia olennaisia asioita, kuten URL-reititys, käyttäjän autorisointi, lokalisointi ja virheen käsittely. (3.)

2.3 Visual Studio -kehitysympäristö

Microsoft Visual Studio on kattava kehitysympäristö, jossa on koodin kirjoittamisen ja projektirakenteen selaamisen lisäksi kattavat debuggaus-, kääntö- ja käyttöönottotyökalut. Siinä on valmiita projektipohjia eri .NET-projekteihin esimerkiksi konsoli-, Windows Forms-, Windows service- ja ASP.NET-sovelluksille. Visual Studiosta löytyy myös työkalut raahaa ja pudota -menetelmällä (drag and drop) graafisten käyttöliittymien suunnitteluun. (4; 5.)

Visual Studio sisältää useita integrointeja versionhallintatyökaluihin, mm. Git ja Team Foundation, sekä monia tehokkuutta lisääviä työkaluja, mm. IntelliSense ja CodeLens. Visual Studiossa on myös Azure-integrointi, jonka avulla voi selata Azuren resursseja tai julkaista projektin Azureen. Ohjelman suorituksen analysointia helpottaa Application Insights -integrointi, joka näyttää monimuotoista analytiikkaa ohjelmasta. Visual Studioon voi ohjelmoida lisäosia, joita laajan suosion vuoksi on runsaasti jaossa sekä ilmaisina että kaupallisina. (4; 5.)

2.4 Git-versionhallintajärjestelmä

Git on erittäin suosittu hajautettu versionhallintajärjestelmä. Git on alun perin Linus Torvaldsin kehittämä, nykyään aktiivisesti ylläpidetty avoimen lähdekoodin projekti. Hajautuksen ansiosta tietovaraston kopio on aina myös itsessään täydellinen tietovarasto historioineen. Gitillä on erittäin tehokas datan tallennustapa, ja se on turvallinen käyttää historiarakenteen ansiosta. (6; 7.)

2.5 GitLab

GitLab on kaupallinen verkkopalvelu, joka hostaa ja ylläpitää Git-tietovarastopalvelimia. Git-tietovarastopaikan lisäksi GitLab tarjoaa web- ja mobiilikäyttöliittymän palveluun sekä työkaluja, kuten virheseuranta, käyttäjäaktiivisuudenseuranta ja wikisivut. (8.)

2.6 Windows Server -palvelinkäyttöjärjestelmä

Windows Server on yleisnimitys Microsoftin kehittämistä palvelinkäyttöjärjestelmistä. Windows Serverit ovat työasemakäytössäkin olevien Windowsien palvelinversioita, jotka on optimoitu verkkoliikenteelle, internetpalveluiden hostaamiseen ja taustaprosessien suoritukseen. Windows serverin versiot sisältävät hyödyllisiä palvelinohjelmistoja, kuten Active Directory (AD) ja Internet Information Services (IIS). (9.)

2.7 Internet Information Services -web-palvelinohjelmisto

Internet Information Services (IIS) on Microsoftin kehittämä web-palvelinohjelmisto. IIS tukee yleisimpiä verkkoprotokollia kuten HTTP/HTTPS, FTP ja STMP. IIS on versiosta 7.0 lähtien ollut täysin modulaarinen eli jokainen toiminto on itsenäinen komponentti. Modulaarisuus lisää tietoturvaa, koska käyttämättömät komponentit voidaan poistaa käytöstä ja haavoittuvuuspinta pienenee. Modulaarisuuden avulla palvelimet voidaan optimoida suorittamaan yhtä osaa suuremmasta arkkitehtuurista, mikä on resurssien kannalta tehokkaampaa. IIS-komponentteja voi myös lisätä tai korvata omilla mukautetuilla komponenteilla, esimerkiksi autentikointi-, monitorointi- tai vaikka kuormantasausmoduuleilla. (10; 11; 12; 13.)

2.8 SQL-kyselykieli

SQL (Stuctured Query Language) on standardoitu kyselykieli tietokantojen käsittelyyn relaatiotietokannan hallintajärjestelmissä (RDBMS). Vaikka SQL on standardoitu, on sen eri versioissa paljonkin eroavaisuuksia, sillä SQL-palvelimien kehittäjät tarjoavat omia versioita tai laajennoksia SQL-kieleen. (14.)

2.8.1 Microsoft SQL Server -tietokantapalvelin

Microsoft SQL Server on yleisnimitys Microsoftin kehittämille SQL-palvelinohjelmistoille. Microsoft SQL Server on asiakas-palvelin-arkkitehtuurin järjestelmä, eli tietokanta on palvelimella ja sitä käytetään asiakasohjelmalla samasta tai eri sijainnista. (15.)

Yhdessä palvelimessa voi olla useampi SQL Server -instanssi (instance), joista yksi on vakio instanssi (default instance). Useamman instanssin palvelimessa muita kuin vakio instanssia kutsutaan nimetyiksi instansseiksi (named instance). Asiakasohjelma voi ottaa yhteyttä suoraan tiettyyn instanssiin. (16.)

Microsoft SQL Server sisältää useita eri komponentteja, kuten SQL Server Agent, SQL Server browser, SQL Server Integration Services (SSIS), SQL Server Analysis Services (SSAS) ja SQL Server Reporting Services (SSRS). SQL Server Agent on Windows-palvelu, joka suorittaa ajastettuja ylläpitotehtäviä SQL-palvelimelle, kuten tietokantojen varmuuskopiointi. Kellonajan lisäksi tehtäviä voi määritellä tehtäväksi tietyn tapahtuman tapahtuessa, esimerkiksi jonkin resurssin käytön ylittäessä raja-arvon. SQL Browser Service on Windows-palvelu, joka tarjoaa tietoa järjestelmään asennetun SQL-palvelimen resursseista ja instansseista. (15; 16; 17; 18.)

2.8.2 SQL Server Management Studio

SQL Server Management Studio (SSMS) on integroitu ympäristö SQL-palvelimien hallintaan. SSMS:llä voi ottaa yhteyden SQL-palvelimeen, selata tietokantoja, tehdä kyselyitä ja konfiguroida palvelimen tietokantoja. SSMS toimii perinteisen SQL Serverin kanssa, mutta myös Azure SQL Databasen ja Azure SQL Data Warehousen kanssa. SSMS sisältää monipuolisen skriptieditorin, sekä useita skriptin generointityökaluja. (19.)

2.8.3 T-SQL-kyselykieli

T-SQL (Transact-SQL) on SQL kielen laajennos. T-SQL sisältää standardin SQL:n lisäksi proseduraalisen ohjelmoinnin ominaisuuksia kuten muuttujat, funktiot ja ehdolliset lausekkeet. T-SQL sisältää paljon sisäänrakennettuja funktioita esimerkiksi merkkijonon tai päivämäärän käsittelyyn. (20.)

3 HOSTING- JA PILVIPALVELUT

Hosting-ratkaisun valinta vaikuttaa esimerkiksi web-sovelluksen tietoturvaan ja suorituskykyyn, ja se on otettava huomioon myös kehitysvaiheessa. Hosting-palveluyritykset tarjoavat monia erilaisia palveluja konesalipaikoista pilvipalveluihin, ja palveluntarjoajat määrittelevät palvelut joskus eri tavalla. Yleisimmät vaihtoehdot web-sovellusten hostaamiseen ovat dedikoitu palvelin, virtuaalipalvelin ja erilaiset pilvipalvelut. (21; 22; 23.)

3.1 Dedikoitu palvelin

Dedikoitu palvelin (dedicated server) on fyysinen palvelin, joten koneen kaikki resurssit ovat käytettävissä rajoittamatta. Yleensä dedikoitujen palvelimien vuokraamisessa voi myös vapaammin valita käyttöjärjestelmän sekä muun ohjelmiston. Usein palvelimen ylläpidosta huolehtii vuokraaja, mutta monesti palveluntarjoajat tarjoavat erilaisia valvontapalveluita. Virtualisointiohjelmistojen ja tietoturvan kehittyessä dedikoitujen palvelimien hyödyt pienenevät virtuaalipalvelimiin verrattuna. (23; 24.)

3.2 Virtuaalipalvelin

Virtuaalipalvelin (Virtual Private Server, VPS) on fyysisen palvelimen sisään asennettu virtuaalinen palvelin, joten niitä voi olla useita samassa fyysisessä koneessa. Virtuaalipalvelimen resurssit voivat olla täysin jaettuja, osittain jaettuja tai taattuja (fixed). Virtuaalipalvelimien resurssien määrää on helpompi lisätä ja vähentää verrattuna dedikoituihin palvelimiin. Virtuaalipalvelimen käyttö ei välttämättä eroa dedikoidun palvelimen käytöstä juurikaan, riippuen palveluntarjoajasta. Karkeasti yleistettynä virtuaalipalvelinpalvelut ovat enemmän rajoitettuja esimerkiksi ohjelmiston kannalta, mutta vaativat myös vähemmän ylläpitoa vuokraajalta. (23; 24; 25; 26.)

3.3 Pilvipalvelut

Pilvipalvelut ovat kustannustehokkaita vaihtoehtoja virtuaalipalvelimille web-sovellusten kehitykseen ja hostaamiseen. Pilvipalvelu on käsitteenä laaja, ja tarkoittaa monenlaista kapasiteetti- tai ohjelmistopalveluita, joita tarjotaan nimenomaan palveluina eikä niitä siis tarvitse asentaa tai ostaa

kokonaan. Yleensä hinnoittelu ja laskutus on käyttöön pohjautuvaa, minkä ansiosta palveluihin siirtyminen on helppoa. (29; 30.)

Pilvilaskenta tarkoittaa hajautettua palvelintietokoneiden ryhmää, jotka jakavat resursseja ja toimivat yhtenä verkkona muodostaen alustan suorituskykyiselle, skaalautuvalle ja kustannustehokkaalle palvelulle. Pilvilaskennassa on monia hyötyjä verrattuna perinteisiin palvelimiin. Matalan kynnyksen käyttöönotto, laitteita ja ohjelmistoja ei tarvitse ostaa ja useimpia ominaisuuksia voi kokeilla vaivatta ja tarpeen kasvaessa resursseja voidaan lisätä. Hajautuksen ansiosta laiteviat eivät pysäytä käyttöä, kun palvelu toimii aina usean fyysisen laitteen voimin. Suurimmilla palveluntarjoajilla on maailmanlaajuisesti palvelinsaleja ja automaattiset kuormantasaajat palveluissa, joten niiden avulla voi kehittää erittäin korkean saatavuuden (availability) sekä toimintavarmuuden palveluita suhteellisen kustannustehokkaasti. (26; 27; 28; 29; 30.)

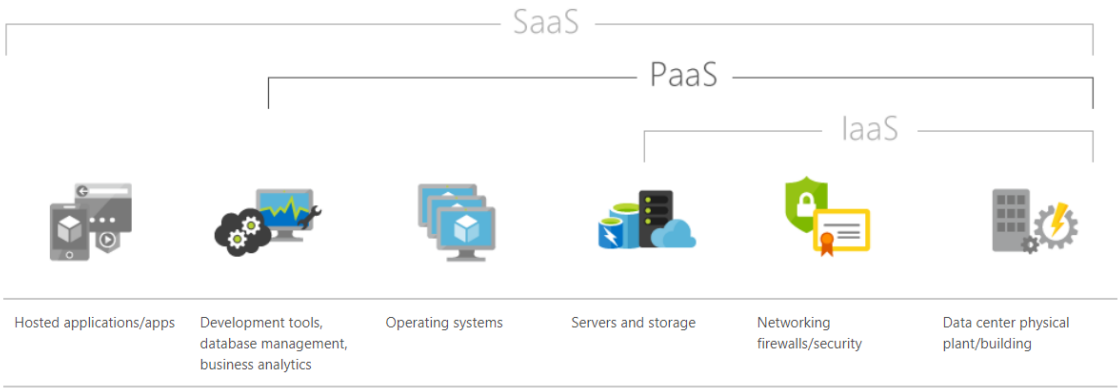
3.4 Palvelumallit

Usein pilvipalveluista puhutaan palvelumallien mukaan termeillä IaaS (Infrastructure as a Service), PaaS (Platform as a Service) ja SaaS (Software as a Service). Kuvassa 2 näkyy Azuren palvelumallit. (27; 31.)

IaaS eli infrastruktuuri palveluna -palveluissa palveluntarjoaja tarjoaa kokonaisvaltaisen infrastruktuurin käyttäjille. IaaS-palveluita käytetään yleensä web-käyttöliittymän, komentorivin tai REST-rajapintojen kautta. Palveluissa käyttäjä voi esimerkiksi luoda kokonaisia virtuaalikoneita ja tietokantoja tai vaikka käyttäjätietojenhallinta palveluita. (27; 31; 32.)

PaaS eli sovellusalusta palveluna on vielä kattavampi kokonaisuus, missä palveluntarjoaja huolehtii myös käyttöjärjestelmistä ja tarjoaa mahdollisesti työkaluja sovelluskehitykseen, käyttöönottoon, testaukseen ja analytiikkaan. PaaS-palvelut mahdollistavat esimerkiksi sen, että käyttäjä voi valmiilla työkaluilla julkaista oman web-sovelluksensa muutamassa sekunnissa ilman että tarvitsisi huolehtia palvelimien konfiguroinneista tai päivityksistä. PaaS-palvelut voivat sisältää myös perinteistä web-sovellusta pienempiä kokonaisuuksia, kuten pieniä rajapintakomponentteja tai ajastettuja toiminnallisuuksia, jotka auttavat esimerkiksi nopeassa prototyyppauksessa. (27; 31; 32.)

SaaS eli ohjelmisto palveluna on erittäin yleinen palvelumalli, josta esimerkkinä sähköpostipalvelut, toimisto-ohjelmistot tai tiedostonjakopalvelut. Yrityskäytössä SaaS-palveluita ovat esimerkiksi asiakkuudenhallintaohjelmistot (CRM) tai toiminnanohjausjärjestelmät (ERP). (27; 31; 32.)



KUVA 2. Microsoftin sivuilla oleva kuva eri palvelumalleista (31).

4 MICROSOFT AZURE

Microsoft Azure on Microsoftin julkinen pilvipalvelu. Azuresta löytyy sekä IaaS- että PaaS-mallin palveluita. Azure tarjoaa perinteisiä virtuaalitietokoneita ja tietokantapalvelimia, mutta myös modernimpia palveluita esimerkiksi koneoppimiseen ja datamassojen analysointiin. Hinnoittelu on käyttöön pohjautuvaa, joten eri palveluita on helppo kokeilla sen suuremmin investoimatta. (33; 34; 35.)

Valmiiden palveluiden lisäksi Azuressa on kehitysalustoja, esimerkiksi pienen yksinkertaisen ajastetun toiminnon ohjelmoimiseen ja suorittamiseen. Azure ei ole rajoittunut tekniikoiltaan vain omiin Microsoftin kehittämiin vaan joukosta löytyy myös paljon avoimen lähdekoodin tekniikkaa ja palveluita voi hyödyntää .NET-kehittäjien lisäksi esimerkiksi Java- tai node.js-kehittäjätkin. Ohjelmien käyttöönotto on mutkatonta ja nopeaa, etenkin Microsoftin omilla työkaluilla. (33; 34; 35.)

Azuresta löytyy ratkaisu käyttäjätietojen hallintaan: Azure Active Directory (AAD), joka muistuttaa hyvin paljon Windows-palvelimista löytyvää Active Directoryä. Identiteettien ja pääsynhallintaan voi käyttää Azure Active Directory B2C- ja Azure Active Directory B2B -ratkaisuja, joiden avulla voidaan toteuttaa esimerkiksi kertakirjautumista (single sign on) ja kaksivaiheista tunnistautumista (multi-factor authentication) tukeva kirjautumispalvelu omaan mobiili- tai web-sovellukseen. (33; 34; 35.)

Azure tarjoaa useita eri arkkitehtuurin tietokantapalveluita sekä tallennustilapalveluita erittäin laajalla saatavuudella ja joustavilla hinnoittelumalleilla. Azure tarjoaa myös hybridi-pilvipalveluita eli osittain sisäiseen ympäristöön asennettavia palveluita, kokonaisen infran tai vaikka pelkästään datan varmuuskopiointiin. Myös suomesta löytyy Azuren kumppaneita, joilta saa Azuren palveluita hostattuna suomessa. Azuren palvelut ovet käytettävissä web-portaalista, PowerShellillä, Azure CLI:llä tai Visual Studiolla. (33; 34; 35.)

4.1 Resource Group

Azuren Resource Groupit eli resurssiryhmät helpottavat resurssien hallintaa, seurantaa ja käyttöönottoa, sillä kaikkia osia ei tarvitse hallita erikseen, vaan esimerkiksi tietyn projektin resurssit

voidaan lisätä samaan resurssiryhmään. Yksi resurssi siis tarkoittaa esimerkiksi virtuaalikonetta, web-sovellusta, tietokantaa tai virtuaalista verkkoa. Resursseja voidaan lisätä ja poistaa resurssiryhmistä milloin vain, ja resurssi voi kuulua vain yhteen resurssiryhmään kerralla. Resurssiryhmien perusteella resursseja voidaan seurata, sekä niillä määritellään resurssien laskutusta ja pääsynhallintaa. Azure Resource Manager (ARM) on tekniikka, joka huolehtii resurssiryhmistä kaiken taustalla. ARM:a ei yleensä erikseen tarvitse käyttää vaan resurssiryhmät voidaan luoda samalla kun resursseja luodaan. Tarpeen vaatiessa ARM-toimintoihin pääsee käsiksi PowerShellin kautta. (36; 37.)

4.2 App Service

Azure App Service mahdollistaa web-sovelluksen ylläpidon ilman, että tarvitsee huolehtia infrastruktuurista eli se on ns. alusta palveluna (PaaS). App Service -resurssityyppejä on Web Appin lisäksi esimerkiksi Api App, Function App ja Mobile App. Niistä kukin sisältää eri toiminnallisuuksia käyttötarkoitukseen liittyen. App Servicen resurssit ovat skaalattavia ja niihin saadaan lisättyä esimerkiksi kuormituksen tasaus (load balancer). Käyttöönotto voidaan automatisoida esimerkiksi suoraan GitHubista tai Visual Studio Team Servicestä. Lisäksi tuotanto-, demo- ja kehitysympäristöt voidaan pitää vaivatta erillään Deployment Slots -toiminnallisuuden ansiosta. Ohjelma voidaan julkaista App Serviceen automaattisen käyttöönoton lisäksi manuaalisesti FTP:llä tai Visual Studion julkaisutoiminnolla. App Service -resursseihin voidaan lisätä automaattinen sisällön tai konfiguraation varmuuskopiointi. App Servicet lisätään aina yhteen App Service Planiin, joka määrittää sovellukselle ympäristön. (38; 39.)

4.2.1 Web App

Web App on yksi App Service -resurssityypeistä, joka on tarkoitettu yleisesti web-sovelluksille. Web App -resurssi voi toimia Windows- tai Linux-pohjaisena. Alustalle voi viedä .NET-, .NET Core-, Java-, Node.js-, PHP-, Python- ja Ruby-sovelluksia, riippuen alustasta. Web Appin konfigurointi on samalla yksinkertaista ja helppoa, silti säilyttäen erittäin laajat mahdollisuudet mukautettuun konfigurointiin. Azuren web-portaalista voi konfiguroida samoja palvelimen asetuksia, joita perinteisesti konfiguroitaisiin IIS:n käyttöliittymästä. Web App alusta tarjoaa myös valmiin autentikoinnin Azure Active Directoryn (Azure AD), Microsoft-tilin, Google-tilin, Facebook-tilin tai Twitter-tilin kautta. Web

App -resurssiin voi portaalista lisätä myös domain-nimiä, SSL-sertifikaatteja, IP-suojauksia, virtuaalilähiverkkoyhteyksiä ja monia muita web-sovelluksiin liittyviä asetuksia. (40.)

4.2.2 Deployment Slots

Azuren Deployment Slots -tekniikka mahdollistaa sen, että yksi App Service -resurssi riittää, vaikka sovelluksesta tarvitaan eri ympäristöjä, esimerkiksi kehitysversio, demoversio ja tuotantoversio. Kun App Servicestä otetaan Deployment Slot -toiminnallisuus käyttöön, Azure luo itseasiassa sovelluksesta täyden kopion omalla domain-nimellä, mutta käyttäen eri sisältöä tai konfigurointeja. Eri Deployment Slot -tiloja voi vaihtaa nopeasti, jolloin palvelun ei tarvitse olla suljettuna päivityksen takia. Sovelluksen muutokset voidaan esimerkiksi julkaista demoympäristöön ja testata tuotantoa vastaavassa ympäristössä. Sen jälkeen voidaan julkaista muutokset vaihtamalla Deployment Slot -tilat keskenään. Jos päivitys menee pieleen, voidaan vaihto perua ja käynnistää viimeisin toimiva konfiguraatio. (41.)

4.3 App Service Plan

App Service Planilla määritellään tavallaan App Servicelle ympäristö eli fyysiset resurssit ja samalla myös laskutus. App Service voi kuulua vain yhteen App Service Planiin, jossa taas voi olla yksi tai useita App Service -resursseja. App Serviceen määritellään fyysinen sijainti (esimerkiksi Pohjois-Eurooppa), virtuaalikoneiden määrä, virtuaalikoneiden kokoluokka (small, medium tai large) ja hinnoittelutaso. Virtuaalikoneen kokoluokka määrittelee prosessoriytimien määrän ja muistikapasiteetin. App Service Plan määrittelee App Service -resurssin ulosnäkyvät IP-osoitteet. (42.)

Hinnoittelutasoja on Free, Shared, Basic, Standard, Premium, PremiumV2, Isolated, Consumpiton. Free- ja Shared-tasot käyttävät jaettua palvelinta ja näin ollen saavat tietyn prosessoriajan käyttöön, resurssit voivat olla siis jaettuja jopa eri Azure-käyttäjien kesken. Free- ja Shared-tasoja suositellaan vain testaus- ja kehityskäyttöön. (42.)

Basic-, Standard- ja Premium-tasot takaavat sovelluksille oman virtuaalikoneen, eli vain samassa App Service Planissa olevat ohjelmat käyttävät App Service Planin virtuaalikonetta ja sen resursseja. Basic-, Standard- ja Premium-tasot toimivat samalla tavalla, paitsi että niissä on eri määrä resursseja tai ominaisuuksia, esimerkiksi virtuaalikoneita, kovalevytilaa tai Deployment Slot -tiloja.

Kalliimmissa luokissa on tiheämmät varmuuskopioinnit ja Traffic Manager -palvelu auttamaan laajempaa saatavuutta. PremiumV2-taso tarjoaa nopeammat prosessorit, SSD-levyt perinteisten kovalevyjen sijaan, tuplasti nopeammat muistiväylät sekä mahdollisuuden vielä suuremiin virtuaalikonemääriin. Basic-, Standard- ja Premium-tasoista on myös eri kokoluokan versiot, joissa on erimäärä prosessoriytimiä ja muistikapasiteettia. (42.)

Isolated-taso määritellään suoritettavaksi erikseen luodun Azure VM -virtuaalikoneen sisälle, jolloin saavutetaan erittäin laajat skaalausmahdollisuudet. Consumption-taso määrittelee sovelluksen laskutuksen käytön mukaan. Consumption-taso on mahdollinen vain Function App -resursseissa. App Service Planin hinnoittelutasoa voi vaihtaa lennosta ominaisuuksien tai resurssien tarpeen mukaan. Esimerkiksi projektin kehitys voidaan aloittaa tekemällä proof-of-concept Free-tasolla, nostaa Shared-tasoon kun tarvitaan mukautettu domain-nimi esittelyyn, nostaa Basic-tasoon kun tarvitaan SSL-sertifikaatti, nostaa Standard-tasoon kun tarvitaan Deployment Slot -tilat erikseen kehitys-, demo- ja tuotantoympäristöön. Lisäksi, jos tarvitaan tehoa lisää, voidaan nostaa hinnoittelutasoa vaikka vain väliaikaisesti. (42.)

4.4 Tietokannat

Azuresta löytyy useita eri tietokantapalveluita. Esimerkiksi Azure SQL Database, Azure Cosmos DB, Azure SQL Data Warehouse, Azure Redis Cache ja Azure Table Storage. (43; 44.)

4.4.1 Azure SQL Database

Azure SQL Database on Microsoft SQL Serverin kaltainen tietokantapalvelu. Kumpaakin voi hallinnoida samalla tavalla esimerkiksi SQL Server Management Studiolla. Azure SQL Database eroaa kumminkin muutamilta osin olemalla rajoitetumpi, esimerkiksi sallituissa T-SQL-lausekkeissa. (45.)

Hinnoittelu on joustavampaa. Lisenssin ostamisen sijaan Azure SQL Database -tietokantojen käytöstä maksetaan tallennuskapasiteetin ja DTU-tason (Database Transaction Unit) mukaan. Käytöstä laskutetaan yksittäisten tietokantojen mukaan tai useasta tietokannasta Elastic Pools -tekniikan avulla. Azure tarjoaa monipuolisia älykkäitä toimintoja ja työkaluja tietokantojen seurantaan ja

optimointiin. Automaattinen optimointi seuraa tietokantakyselyitä ja tunnistaa resurssiahneet toiminnot, jonka jälkeen, riippuen asetuksista, palvelu tekee automaattisesti tarvittavat muutostoimenpiteet tai ehdottaa niitä käyttäjälle. Automaattinen optimointitoimenpide voi olla esimerkiksi indeksin lisäys tietokantaan, jos tekoäly tunnistaa tietyllä sarakkeella tehtävän paljon hakuja. Lisäksi älykkäät toiminnot varoittavat epäilyttävistä kyselyistä tietokannassa ja näin parantavat entisestään tietoturvaa. Tietokantojen hinnoitteluluokkia voi vaihtaa samalla tavalla kuin App Service Planissa lennosta pysäyttämättä palvelua. (45.)

4.4.2 Azure Redis Cache

Azure Redis Cache on tiedontallennuspalvelu, joka pohjautuu suosittuun avoimen lähdekoodin Redis-palvelimeen. Redis-palvelimen nimi tulee sanoista Remote Dictionary Server, ja nimensä mukaisesti Redis on palvelin avain-arvo-parien tallennukseen. Redis tukee yksinkertaisen datan lisäksi myös rakenteellistakin dataa, kuten järjestettyjä listoja. Se tallentaa datan muistiin ja tästä syystä on erittäin nopea verrattuna esimerkiksi perinteisiin SQL-palvelimiin. Redis-palvelimet on suunniteltu toimimaan hajautetusti ilman erityisiä synkronointeja tai yhteyksien lukkiutumista. Hajautettu järjestelmä luonnollisesti kasvattaa suorituskykyä entisestään sekä takaa laajemman saatavuuden. (44; 46; 47.)

4.4.3 Azure Table Storage

Azure Table Storage on NoSQL avain-arvo-pari pohjainen tietokanta, johon voi tallentaa joustavarakenteista dataa (semi-structured data). Table Storage soveltuu sekä minimaalisen että laajan saatavuuden sovelluksen joustavaan datan tallennukseen, eikä vaadi juurikaan konfigurointia verrattuna perinteisiin tietokantoihin. Azure tarjoaa valmiit kirjastot useimmille Azuren tukemille kielille ja alustoille. Kirjastot muuntavat valmiin .NET-luokan JSON-serialisoimalla dokumenttimuotoon ja tallentaa sen tietokantaan ilman, että rakennetta tarvitsee etukäteen kuvata. (48.)

4.5 Tallennustila

Azuresta löytyy useita eri vaihtoehtoja tiedostojen tallennukseen. Esimerkiksi Disk Storage -palvelu, joka tarjoaa hallittuja HDD- tai SSD-levyjä, ja erittäin skaalautuva Blob Storage -palvelu, jonka

laskutus on käyttöön pohjautuvaa. Lisäksi Azuresta löytyy kustannustehokas Azure Archive Storage, Azure Files tiedostonjakoon SMB-protokollalla (Server Message Block) ja Queue Storage viestipohjaiseen ohjelmien väliseen kommunikointiin. (49.)

Blob Storage on erittäin skaalautuva rakenteettoman datan säilytykseen tarkoitettu palvelu. Data voidaan määrittää hot-, cool- tai archive-tarpeellisuustasolle. Taso määrittää datan saatavuuden tason ja käytön hinnan. Dataa voidaan hakea REST-arkkitehtuurin mukaisesti laajalla saatavuudella. Blob Storage soveltuu median tai minkä tahansa datan tallennukseen ja suoratoistoon. Blob Storagessa ei tarvitse ostaa etukäteen tiettyä varastotilaa vaan käytöstä laskutetaan datan määrän, datan käsittelyoperaatioiden tyypin ja määrän sekä datan tarpeellisuustason mukaan. (50.)

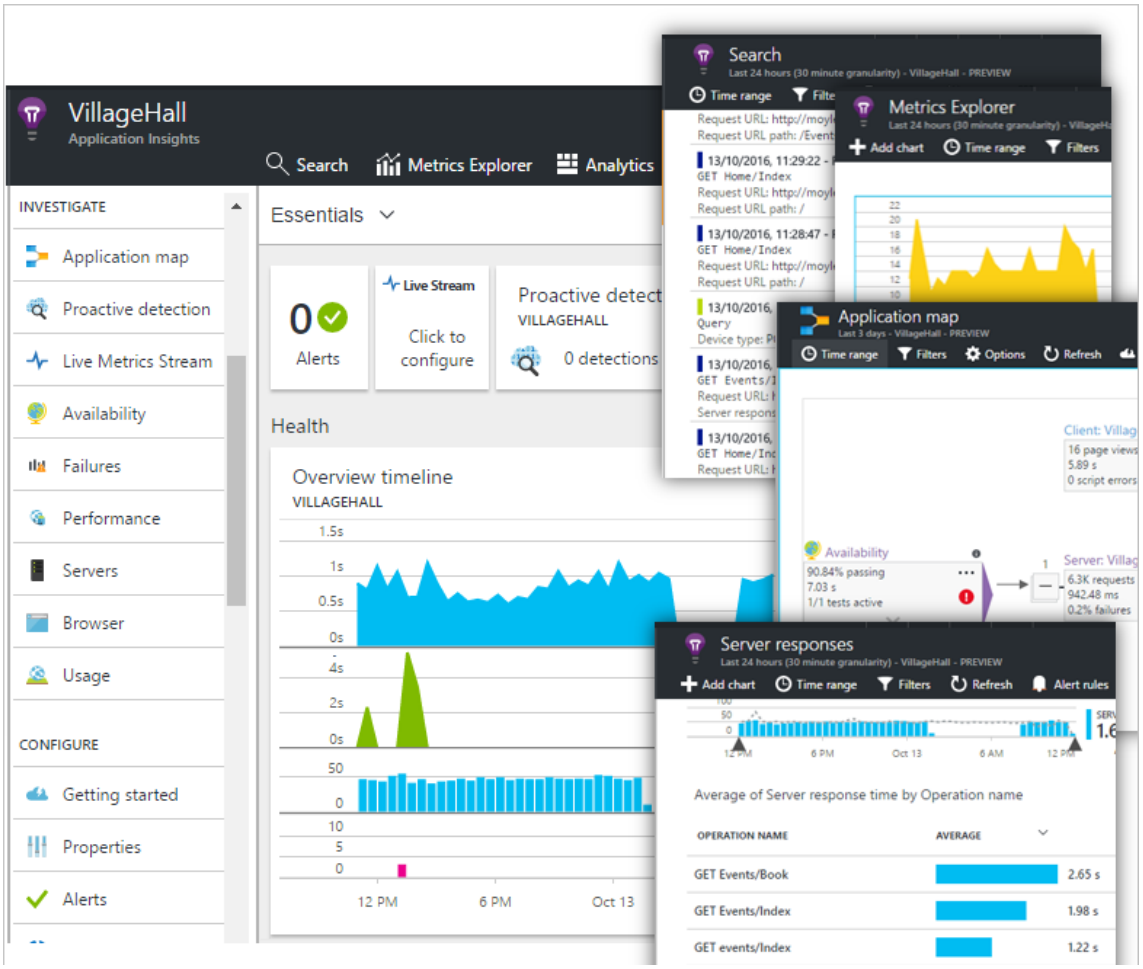
4.6 Application Insights

Application Insights (AI) on web-sovellusten suorituskyvynhallintapalvelu. Sillä voi tunnistaa ja analysoida ongelmia, joita korjaamalla voi optimoida sovelluksen suorituskykyä. Manuaalisen datan analysoinnin lisäksi palvelu kykenee tunnistamaan poikkeuksellisia tapahtumia sovellusten käytössä ja raportoimaan niistä automaattisesti esimerkiksi sähköpostilla. (51; 52.)

Application Insights kerää monipuolista dataa web-sovelluksen käytöstä ja rakentaa selkeitä ja havainnollisia yhteenvetoja liittyen web-sovelluksen prosessorin, verkkoliikenteen, levyn ja muiden resurssien käyttöön (kuva 3). Näin voidaan tunnistaa pullonkaulat ja virheelliset tai resurssiahneet komponentit ja kehittää entistä suorituskykyisempiä sovelluksia. Lisäksi Application Insights kerää tietoa virheellisistä, poikkeuksia sisältävistä tai suorituskykyrajojen ylittävistä HTTP-kutsuista. Näin virheet tunnistetaan mahdollisimman ajoissa ja tarkan suoritusdatan avulla virhe on helpompi toistaa ja korjata. (51; 52.)

Application Insightsilla voi myös manuaalisesti seurata web-sovelluksen käyttöä esimerkiksi uuden featuren julkaisun jälkeen, kun tarvitaan tietoa, miten käyttäjä tarkalleen käyttää ohjelmaa. Application Insights on integroitu Visual Studioon, joten dataa voidaan selata myös suoraan Visual Studiosta kirjautumatta erikseen Azuren web-portaaliin. Lisäksi koodieditoriin saa hyödyllisiä työkaluja ja integrointeja. Esimerkiksi jos tuotantopalvelimella tulee poikkeustilanne koodiin ja ohjelma kes-

keytyy, Visual Studion tekstieditorissa kyseisen keskeytyneen metodin kohdalle tulee ilmoitus. Ilmoituksen avaamalla nähdään, montako kertaa, milloin ja miten koodi on keskeytynyt tähän kohtaan. (51; 52.)



KUVA 3. Microsoftin sivuilla oleva esittelykuva Application Insightista (51).

5 SOVELLUKSEN KEHITYS JA JULKAISU AZUREEN

Tämän työn tavoitteena oli julkaista yrityksen sovellus Microsoft Azure -pilvipalveluun. Ennen varsinaista kehitystyötä piti perehtyä tarkemmin Azuren palveluihin ja selvittää, mitä niistä voitaisi hyödyntää tähän projektiin. Azure App Services -palvelu osoittautui ilmiselväksi alustaksi sovellukselle, sillä se on Azuren pääasiallinen alusta web-sovelluksille.

Sen jälkeen, kun olin perehtynyt yleisesti pilvipalveluiden ja hajautettujen web-palvelimien toimintaan, tutustuin erityisesti Azure App Services -palveluun ja selvitin, mitä sovelluksesta pitää muuttaa, jotta sen voisi julkaista palveluun. Tässä osassa käyn läpi keskeisimmät ominaisuudet, jotka vaativat muutoksia sovelluksessa, ja toiminnot, jotka tulee vastaan sovellusta julkaistaessa.

5.1 Web Site -projektin migraatio Web Application -projektiksi

Yrityksen sovellus on kehitetty ASP.NET Web Forms -tekniikalla Web Site -projektina Visual Studiolla. Web Site -projekti eroaa monellakin tavalla Web Application -projektista. Huomattavimpana erona Web Application -projekti käännetään yhdeksi binääriksi, kun taas Web Site -projekti mahdollistaa lähdekoodin dynaamisen kääntämisen palvelimella ennen ohjelman suoritusta. Web Site -projekteissa julkaistaan yleensä sivujen kooditiedostot luettavassa muodossa. Web Application -projektissa on selkeämpi projektirakenne, sillä tiedostot, jotka kuuluvat projektiin, valitaan erikseen. Web Application -projektirakenne mielestäni auttaa huomattavasti Azureen julkaisussa Visual Studion julkaisutoiminnon takia. Migraatioon löytyi valmis ohje Microsoftilta, joka menee tiivistettynä seuraavasti:

1. Ennen migraatiota avaa ja tarkista, että Web Site -projekti toimii, suorittamalla se kerran.
2. Luo uusi Web Application -projekti tyhjällä pohjalla.
3. Lisää tarvittavat viittaukset ulkoisiin kirjastoihin.
4. Kopioi kaikki tiedostot Web Site -projektista Web Application -projektiin, ja lisää ne erikseen projektiin include in project -toiminnolla.
5. Konvertoi projektin tiedostot convert to web application -toiminnolla. Toiminto muuttaa komponenttien kooditiedostot osittaisiksi luokiksi (partial class) ja generoi .designer.cs-tiedostot sekä muuttaa codeFile-attribuutit codeBehind-atribuuteiksi.

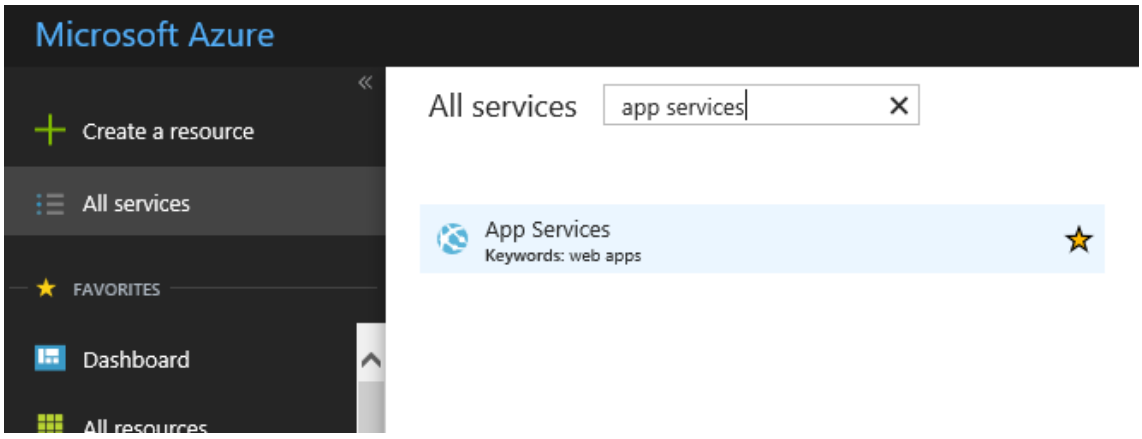
6. Käännä ja aja projekti, tarkista että projekti käynnistyy oikein ja korjaa mahdolliset koodigeneraattorin virheet.
7. Lisää nimiavaruudet (namespace) kooditiedostoihin surround-with-työkalulla. (53; 54.)

5.2 Julkaisu Azure App Service -palveluun

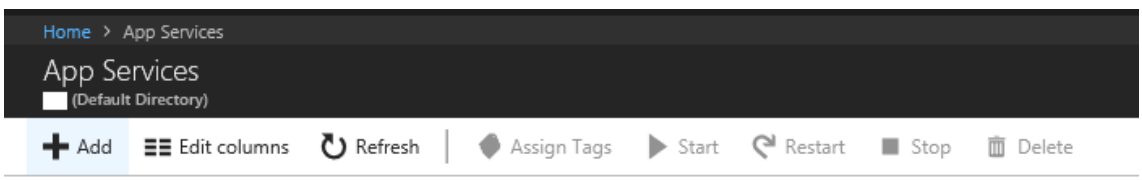
Web Application -projektimigraation jälkeen on hyvä kokeilla ennen muita ominaisuuksia julkaista sovellus Azure App Services -palveluun ja testata, että perustoiminnallisuus toimii. Mahdolliset ongelmatilanteet julkaisussa on helpompi selvittää ja korjata, kun on vähemmän vaihtoehtoja, jotka voivat mennä pieleen.

App Service -resurssin voi luoda suoraan Visual Studion graafisella julkaisutyökalulla tai sitten erikseen web-portaalista. Julkaisu on helpoin tehdä Visual Studion julkaisutoiminnolla.

Ensin käydään web-portaalin kautta luomaan uusi App Service -resurssi. Kaikki resurssit ja työkalut löytyvät Azuren web-portaalista All services -sivun hausta. Haetaan kuvan 4 mukaisesti termillä ’app services’. Uusia App Service -resursseja voidaan luoda App Services -sivulta kuvan 5 mukaisesta yläpaneelista.

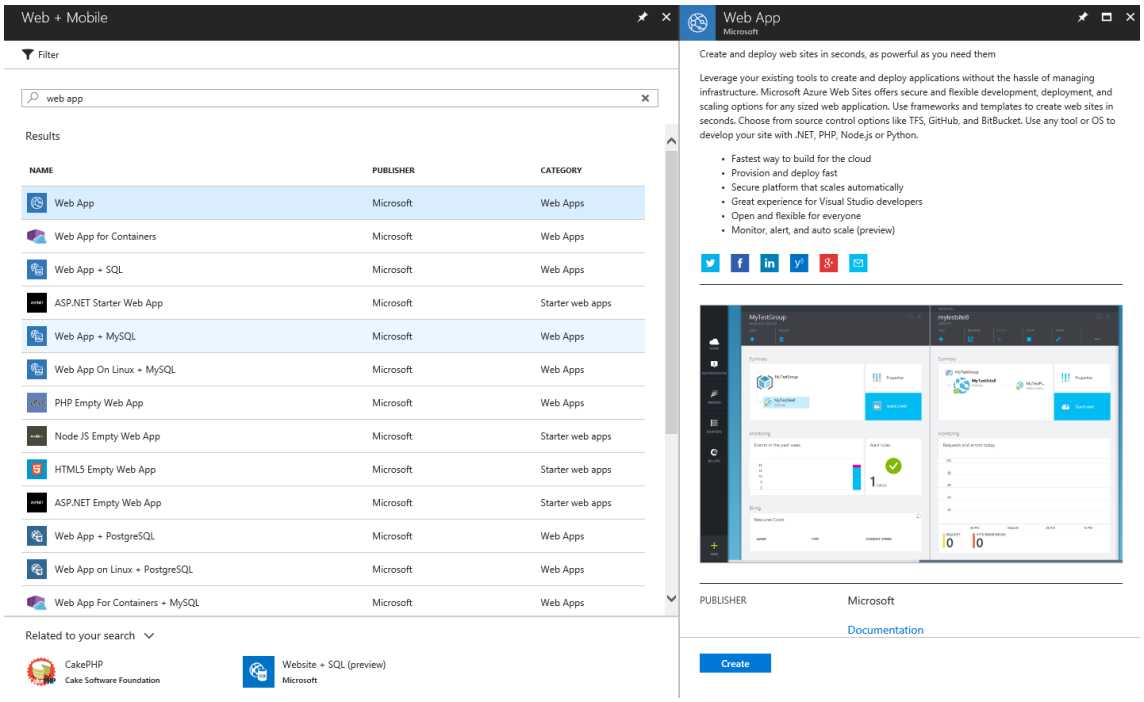


KUVA 4. Azuren web-portaalin haku.



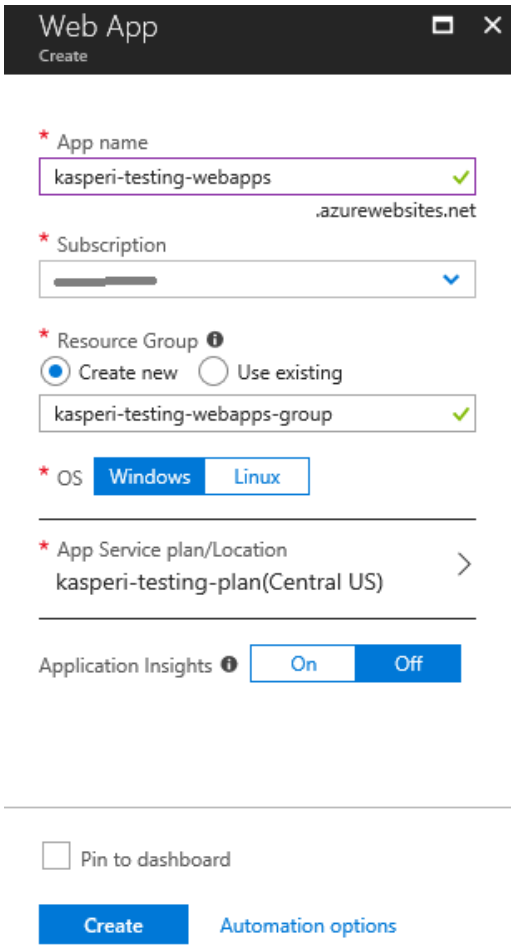
KUVA 5. Azuren web-portaalin tyypillinen yläpaneeli.

Sen jälkeen aukeaa valikko josta valitaan, minkä tyyppinen App Service -resurssi halutaan luoda. Valmiita pohjia on runsaasti. Valitaan Web App -malli pohjaksi. (Kuva 6.)



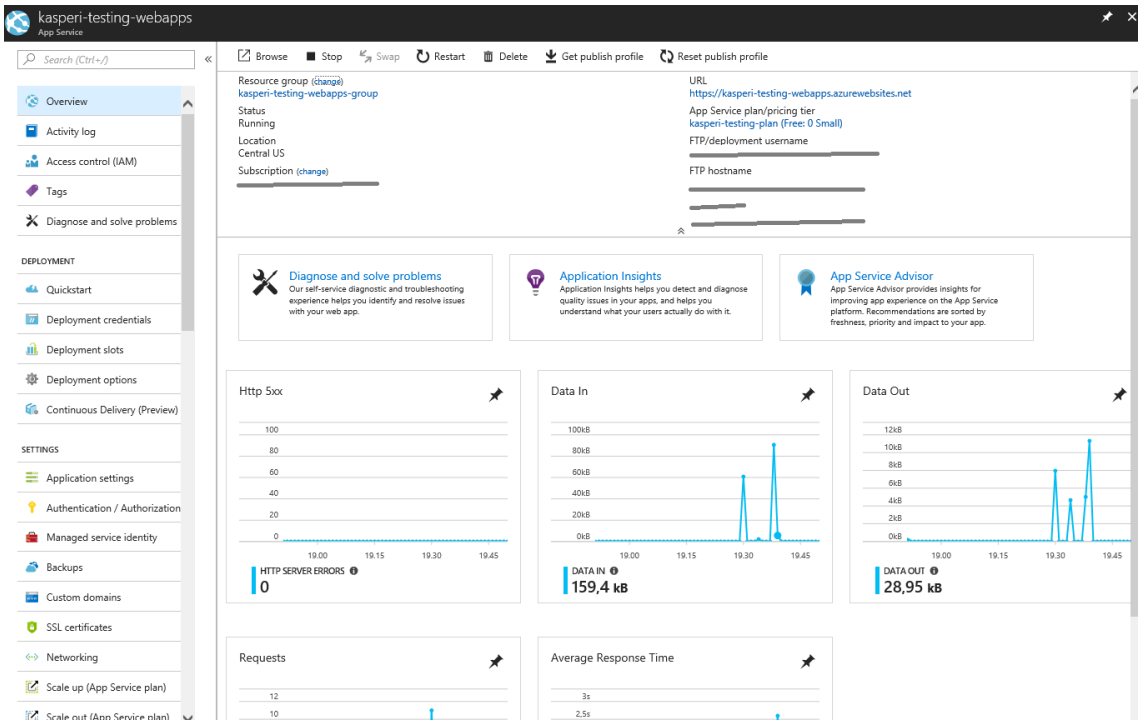
KUVA 6. App Service -pohjien listaus.

Web App -resurssille asetetaan arvot kuten esimerkiksi mikä nimi, mihin tilaukseen yhdistetään, mihin Resource Groupiin yhdistetään, Windows- vai Linux-pohjainen alusta tai mihin App Service Planiin yhdistetään. Klikkaamalla Resource Group tai App Service Plan -valintaa, avautuu uusi valikko, josta voi myös luoda kokonaan uusia resursseja. Tässä vaiheessa kannattaa luoda sovellukselle oma App Service Plan, ja koko projektille oma Resource Group, johon lisätään myöhemmin myös muita resursseja. (Kuva 7.)



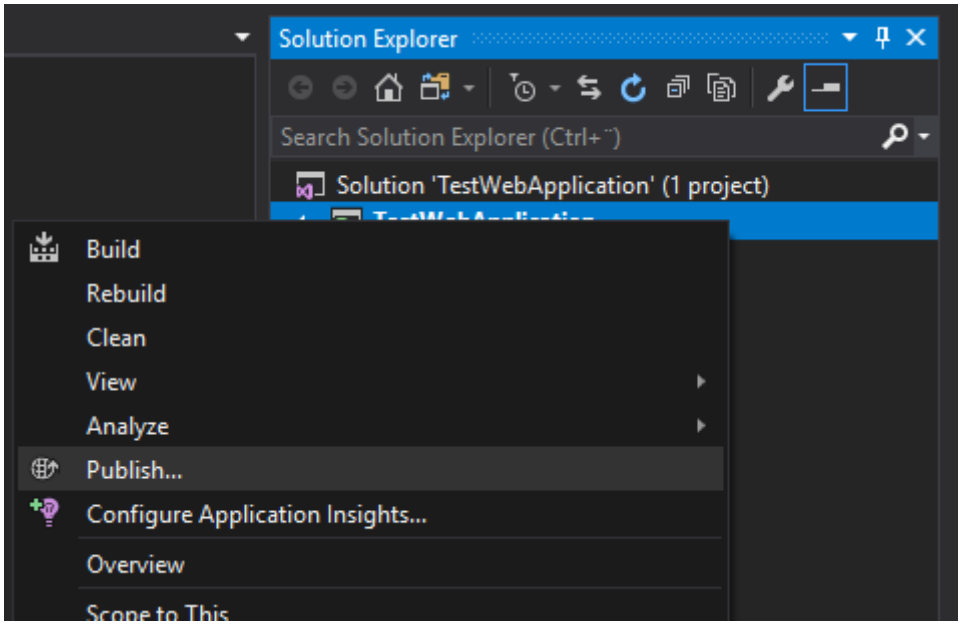
KUVA 7. Web App -resurssin luonti.

Nyt Web App -tyyppinen App Service -resurssi on luotu, ja sen löytää App Services -valikosta, josta klikkaamalla avautuu yleiskatsaus-sivu. Sivulta näkee äsken valitut arvot, esimerkiksi Resource Group ja App Service Plan. Lisäksi sivulla on sovelluksen URL-osoite sekä julkaisuun FTP-yhteystiedot. Sivulla on myös graafeja sovelluksen käytöstä, esimerkiksi 500 HTTP-statuksen vastauksien määrä graafina. (Kuva 8.)



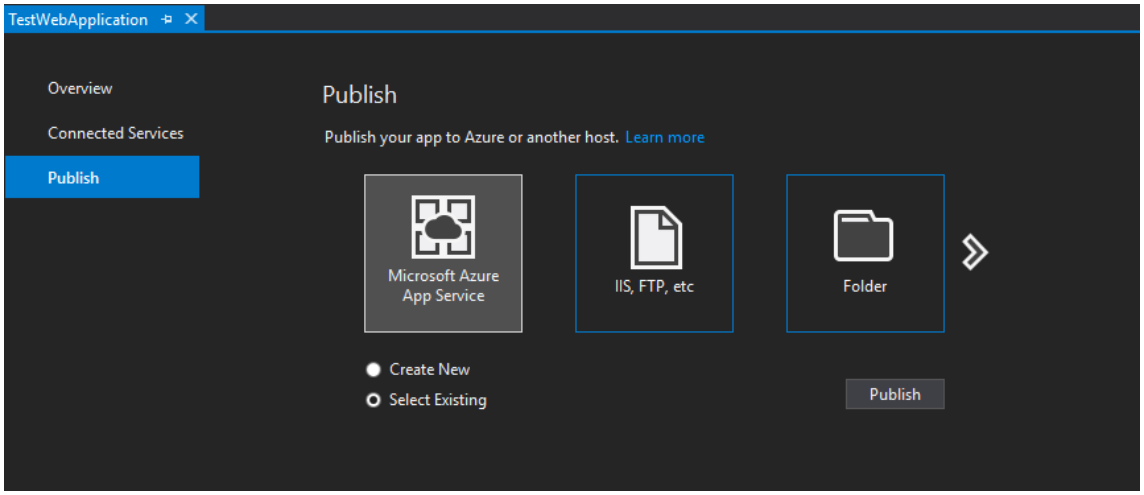
KUVA 8. App Service -resurssin yleiskatsaus-sivu.

Nyt kun App Service -resurssi on luotu, se näkyy myös Visual Studiossa, jos on kirjauduttu Microsoft-tunnuksilla. Solution Explorerista valitaan oikea projekti, josta pudotusvalikosta avataan julkaisutoiminto. (Kuva 9.)



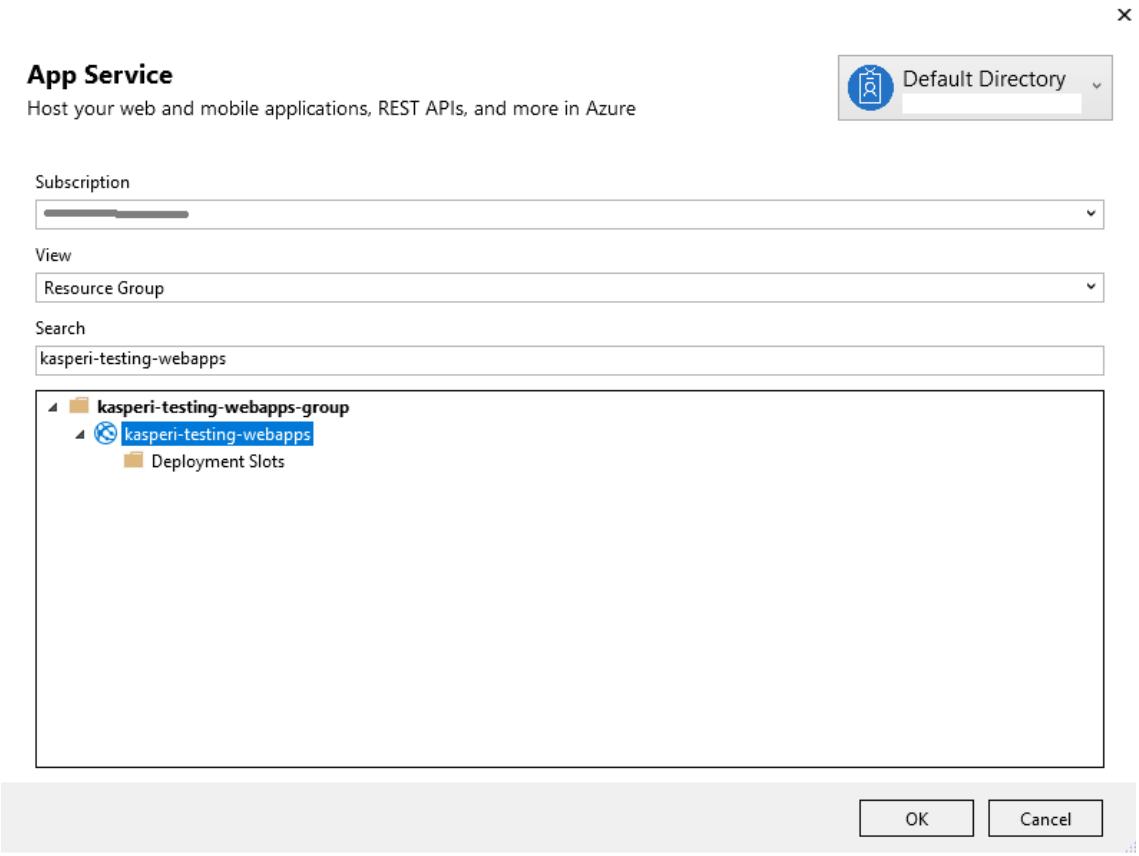
KUVA 9. Projektin toiminnot pudotusvalikossa.

Julkaisutoiminnon näkymästä valitaan olemassa oleva App Service -resurssi. Tässä vaiheessa olisi voitu myös luoda uusi App Service -resurssi käymättä Azuren web-portaalissa. (Kuva 10.)



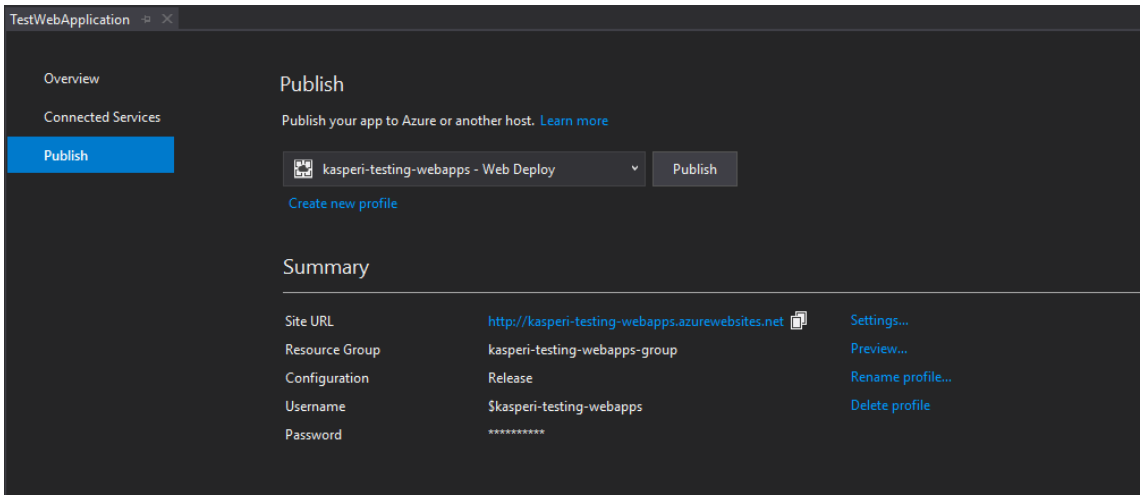
KUVA 10. Julkaisutoiminto.

Ikkunasta valitaan, millä tunnuksilla Azureen otetaan yhteys ja millä tilauksella olevat App Service -resurssit listataan. Sen jälkeen voi haulla hakea resurssin ja valita suoraan myös oikean Deployment Slot -tilan, jos sellainen on luotu. (Kuva 11.)



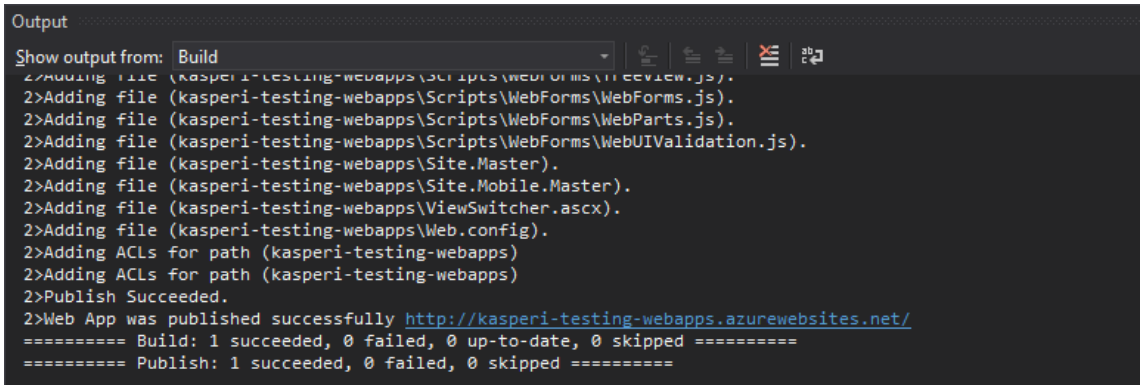
KUVA 11. App Service -resurssin valinta kohteeksi.

Julkaisutoiminto julkaisee sovelluksen Azureen ja tallentaa myös julkaisukonfiguraation tiedostoon. Seuraavalla kerralla avaamalla julkaisutoiminto Visual Studio lataa aikaisemman julkaisukonfiguraation pohjaksi ja näyttää esimerkiksi, mihin URL-osoitteeseen sovellus on menossa julkaistavaksi. Lisäasetuksia julkaisuun löytyy settings-linkin takaa. Kääntökonfiguraatio on myös vaihdettavissa. Julkaisua ennen tarkistetaan, että valittu App Service -resurssi on oikea ja että kääntökonfiguraatio on oikea. (Kuva 12.)



KUVA 12. Julkaisutoiminto esiasetuksilla.

Julkaistaessa voi seurata Output-ikkunaa. Siitä huomaa, jos jotain poikkeuksellista tapahtuu tai jos kääntö tai julkaisu ei onnistu ollenkaan. (Kuva 13.)



KUVA 13. Output-tuloste.

Julkaisun jälkeen käydään selaimella testaamassa, onnistuiko julkaisu, ja aloitetaan mahdollisten virheiden selvittely. Ensimmäisellä kerralla tässä vaiheessa törmäsin ongelmiin, jotka lopulta olivat puuttuvia kirjastoja tai vääriä konfiguraatioita liittyen resursseihin. Ongelmat korjaantuvat lisäämällä puuttuvat tiedostot projektiin manuaalisesti tai NuGet-paketinhallintatyökalulla.

5.3 Istunnon tallennus

HTTP on tilaton protokolla eli web-palvelin käsittelee jokaisen pyynnön erillisenä itsenäisenä pyyntönä tietämättä, mitä edellisessä pyynnössä tapahtui. ASP.NETin istuntotila (ASP.NET session

state) mahdollistaa käyttäjätietojen tallentamisen web-sovelluksen sivupyyntöjen välillä, eli se luo tietyn ajan kestävän istunnon. (55.)

ASP.NETin istuntotilan tallennuksen oletusmenetelmä toimii mutkattomasti dedikoidussa palvelimessa, kun istunto voidaan tallentaa paikallisesti. Kun siirrytään hajautettuihin pilvipalveluihin, oletusmenetelmä ei toimikaan suoraan, jolloin istunto pitää konfiguroida tallentumaan keskitetysti.

5.3.1 ASP.NETin istuntotila

ASP.NETin istuntotila on oletuksena InProc-tilassa, jossa istunto tallennetaan palvelimen muistiin. Toinen vaihtoehto nopeaan session tallennukseen on StateServer-tila, jossa istunto tallennetaan ASP.NET State Service -Windows-palveluun. Tämän etu on, että istunnot säilyvät, vaikka web-sovellus käynnistettäisiin uudelleen. ASP.NET State Service voidaan avata käytettäväksi verkosta, jolloin mahdollistetaan hajautettu web-palvelinympäristö. Siinä on kumminkin huonot puolensa, kriittisimpänä huono tietoturva salaamattoman liikenteen takia. InProc- ja StateServer-tilat ovat suhteellisen nopeita, koska kummassakin data tallennetaan muistiin, joskin StateServer-tila voi olla hitaampi käytettäessä eri sijainnista. (55; 56; 57.)

Istunnon voi tallentaa myös SQLServer-tilassa, nimensä mukaisesti SQL Server -palvelimelle. Tämä mahdollistaa StateServer-tilan tavalla istunnon säilymisen sovelluksen uudelleenkäynnistyksen yli sekä käytön hajautetuissa ympäristöissä. SQLServer-tila on kumminkin hitaampi kuin StateServer-tila, koska data tallennetaan levylle. SQLServer-tila on huomattavasti tietoturvallisempi, koska SQL-yhteys voidaan salata. (55; 56; 57.)

Lisäksi istunto voidaan tallentaa mukautetusti Custom-tilassa oman implementaation kautta. System.Web.SessionState-kirjastosta löytyy pohjaluokka SessionStateStoreProviderBase, jonka perimällä voidaan toteuttaa istunnon tallennus. Valmiita mukautettuja implementaatioita löytyy myös runsaasti, esimerkiksi Azuren Redis Cachea tai Azure Table Storagea hyödyntäviä toteutuksia. (56; 58.)

Kun web-sovellus asennetaan Azuren App Services -palveluun, sovellus toimii hajautetussa palvelinympäristössä, eli sovellusta suorittava palvelin voi vaihtua lennosta. Tämän vuoksi muistiin

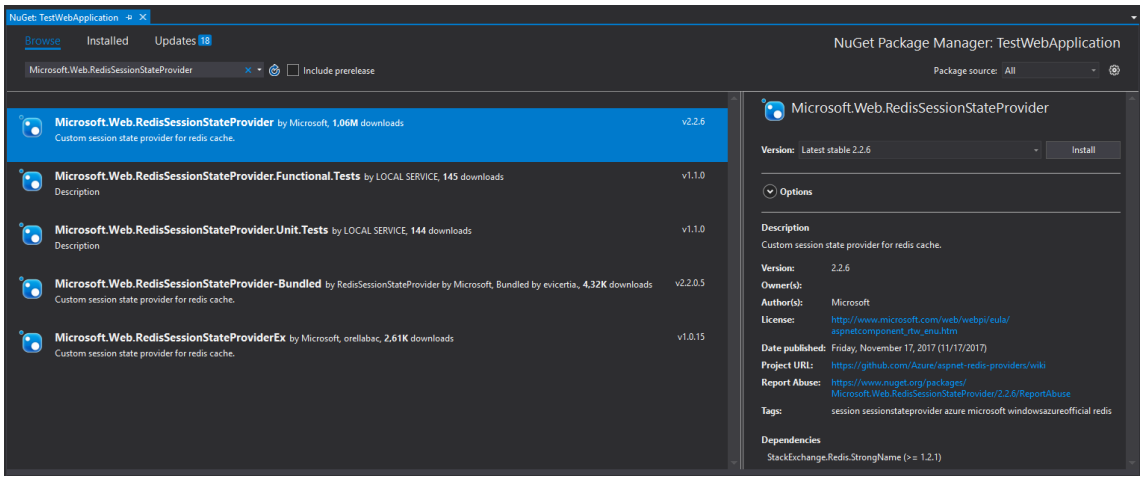
tallentaminen ei ole mahdollista, eli istuntotilan InProc-tila ei ole mahdollinen. StateServer-tilaa en katsonut tarpeeksi tietoturvalliseksi vaihtoehdoksi liikenteen salaamattomuuden takia.

Jäljelle jää joko SQLServer- tai Custom-tila. Koska halusin pitää kokonaisuuden Azuressa, SQL-Server-tilassa käyttäisin Azure SQL Database -palvelua. Custom-tilassa taas olisi järkevää käyttää Azuren palveluiden pohjalle toteutettua valmista kirjastoa.

Lopulta vertailin seuraavia kolmea vaihtoehtoa, Azure SQL Database, Azure Redis Cache ja Azure Table Storage. Redis Cache nousi esiin nopeimpana, mutta toisaalta Azure Table Storage olisi huomattavasti halvin käyttää. Näistä päädyin lopulta Azure Redis Cache -pohjaiseen toteutukseen. (59.)

5.3.2 Azure Redis Cache -palvelun konfigurointi

Avataan NuGet-paketinhallintatyökalu Solution Explorerista oikean projektin kohdalta pudotusvalikosta. Kuvan 14 mukaisesti haetaan termillä 'Microsoft.Web.RedisSessionStateProvider', valitaan oikea hakutulos ja oikealle ilmestyy paketin tiedot. Asennetaan paketti install-painikkeesta, ja hyväksytään paketin käytössä olevat lisenssit. (59.)



KUVA 14. NuGet-paketinhallinnan haku.

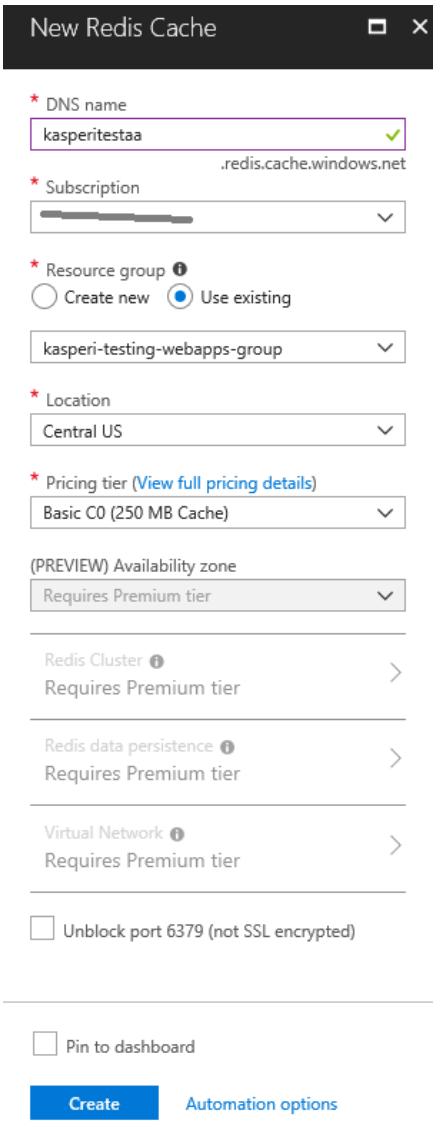
NuGet-paketti sisältää tarvittavat koodit ja viittaukset sekä lisää automaattisesti web.config-konfiguraatiotiedostoon istuntotilan konfiguroinnin juuri asennettuun kirjastoon. host- ja accessKey-arvot ovat aluksi tyhjiä, ja niihin lisätään myöhemmin Azure Redis Cache -palvelun osoite ja pääsyavain. Lisäksi eri konfiguraatioihin, esimerkiksi kehitys- ja tuotantoympäristöön, voidaan asettaa

eri Redis Cache -instanssit muuntamalla host- ja accessKey-arvoja konfiguraatiomuunnoksissa. Pitää myös muistaa poistaa vanha istuntotila-konfigurointi. Tässä tapauksessa aiemmin oli käytetty paikallisella koneella olevaa ASP.NET State Service -Windows-palvelua. (Kuva 15.)

```
24    <sessionState mode="Custom" customProvider="MySessionStateStore">
25      <providers>
26        <add name="MySessionStateStore" type="Microsoft.Web.Redis.RedisSessionStateProvider" host="" accessKey="" ssl="true" />
27      </providers>
28    </sessionState>
```

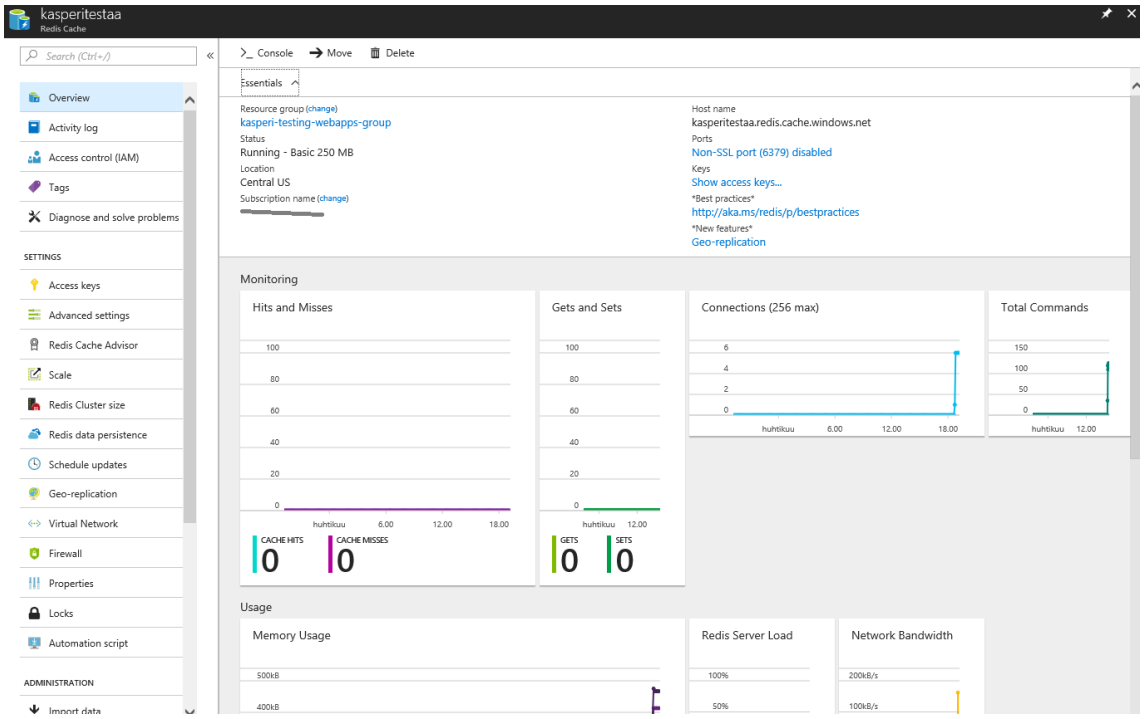
KUVA 15. sessionState-elementti konfiguraatiotiedostossa.

Seuraavaksi mennään Azuren web-portaaliin, haetaan hausta termillä ’redis caches’, ja avataan Redis Caches -sivu. Sivulta luodaan uusi Redis Cache -resurssi, ja avautuu kuvan 16 mukainen sivu. Asetetaan nimi, tilaus, Resource Group, alue ja hinnoittelutaso. Resource Groupiksi laitetaan sama kuin App Service -resurssillekin, jotta voidaan ajatella nämä samana ratkaisuna ja seurata näitä portaalista jälkeenpäin yhtenä ryhmänä. Alueeksi laitetaan myös sama, koska se vaikuttaa tiedonsiirtonopeuteen. Hinnoittelutasoksi laitetaan ensin kaikista halvin, ja sitä voidaan nostaa tarvittaessa myöhemmin. (60.)



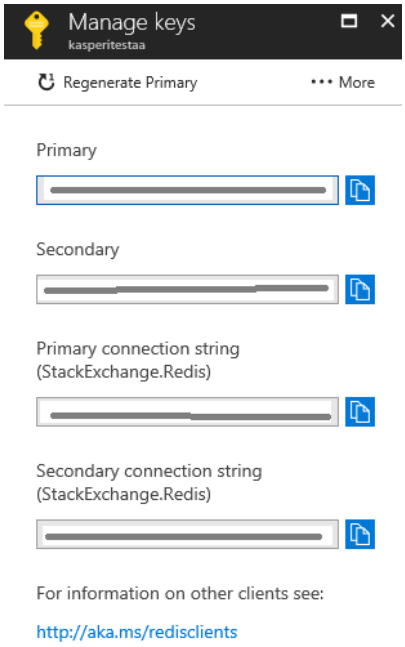
KUVA 16. Redis Cache -resurssin luonti.

Kun Redis Cache -resurssi on luotu, voidaan navigoida sen yleisnäkymä-sivulle. Kopioidaan Host name -osoite ja asetetaan se sovelluksen istuntotilan konfigurointiin host-arvoksi. Vielä tarvitaan pääsyavain, joka löytyy painamalla show access keys -linkkiä yleisnäkymä-sivulla. Avautuu näkymä, missä Redis Cachen pääsyavaimet voidaan generoida ja kopioida. (Kuva 17.)



KUVA 17. Redis Cache -resurssin yleiskatsaus-sivu.

Kopioidaan Primary-avain, ja liitetään se sovelluksen istuntotilan konfigurointiin accessKey-arvoksi. Tämän jälkeen Redis Cache pitäisi toimia istuntotilan säilönä yksinkertaisimmalla mahdollisella konfiguraatiolla. (Kuva 18.)



KUVA 18. Redis Cachen pääsyavainten hallintasivu.

Kokeillaan, toimiiko istunnon tallennus Redis Cache -palveluun. Lisätään kuvan 19 mukainen tekstin tallennus ja luku istunnosta eri sivunlatauksiin. Toteutus toimi odotetulla tavalla eikä vaadi koodimuutoksia olemassa olevaan sovellukseen. Istunnon tallennus voidaan konfiguroida myös niin, että paikallisesti käytetään edelleen InProc-tilaa, mutta julkaisukonfiguraatioon lisätään muutoskonfiguraatio käyttämään sen sijaan Redis Cache -palvelua.

```
protected void Page_Load(object sender, EventArgs e)
{
    Session["just-testing-this-feature"] = "hello redis cache";
}
```

```
protected void Page_Load(object sender, EventArgs e)
{
    TestLiteral.Text = Session["just-testing-this-feature"] as string ?? "failed to get value from session";
}
```

KUVA 19. Istuntoon tallentamisen testauskoodi.

5.4 Tiedostojen tallennus

Yrityksen sovelluksessa ylläpidetään geneeristä sisältöä eli sivuja, tekstejä, linkkejä tai kuvia ja muita tiedostoja. Suurin osa sisällöstä voidaan tallentaa SQL-tietokantaan, paitsi tiedostot, joista tallennetaan vain metatiedot SQL-tietokantaan ja itse tiedosto on nykyisessä toteutuksessa paikallisella levyllä.

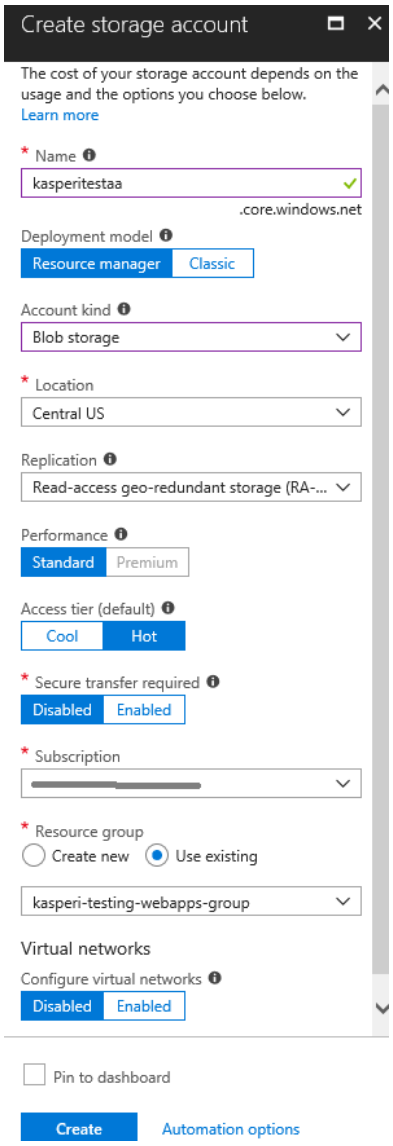
5.4.1 Tiedostojen tallennus Azure App Service -alustalla

Nyt kun järjestelmä asennetaan toimimaan Azure App Services -palvelussa, ei ohjelmalla ole pääsyä paikallisiin tiedostoihin. Tiedostosijainniksi halusin jonkin palvelun Azuresta, jotta alkuperäinen ajatus pitää koko kokonaisuus Azuressa toteutuisi. Päädyin Blob Storage -palveluun, koska se on suunniteltu juuri tämänkaltaista tarvetta varten. Blob Storage -palvelussa on valmiit rajapinnat tiedostojen tallennukseen, hakuun ja jopa streamaukseen. (61.)

5.4.2 Azure Blob Storage -palvelun konfigurointi

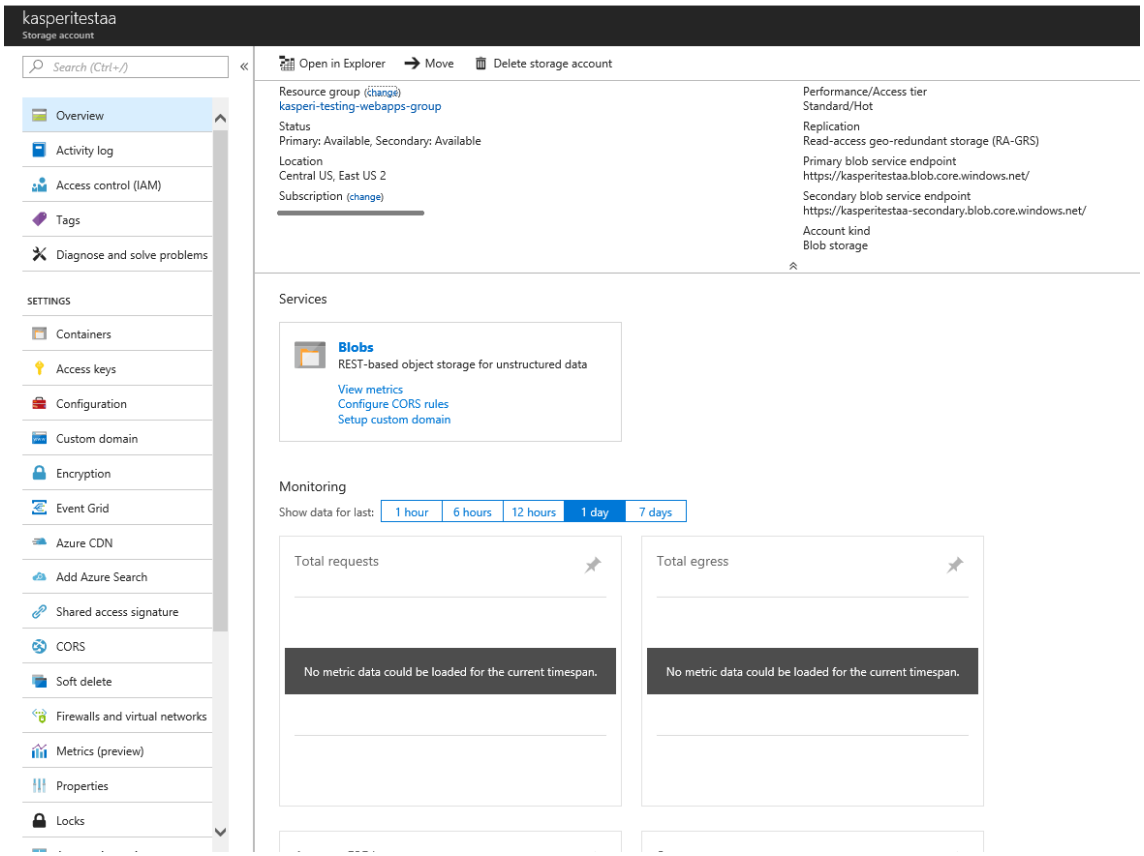
Haetaan Azuren web-portaalista termillä ’storage accounts’ ja avataan sivu. Luodaan uusi Storage Account ja päästään aiemmista kohdista tutun näköiselle kuvan 20 mukaiselle sivulle. Lisätään

nimi Storage Account -resurssille ja valitaan Account kind -arvoksi Blob storage. Asetetaan sijainniksi ja Resource Group -arvoksi sama kuin App Service -resurssillakin. Asetetaan Access tier -arvoksi hot, koska varastossa aiotaan pitää tiheästi haettavia kuvatiedostoja. (61.)

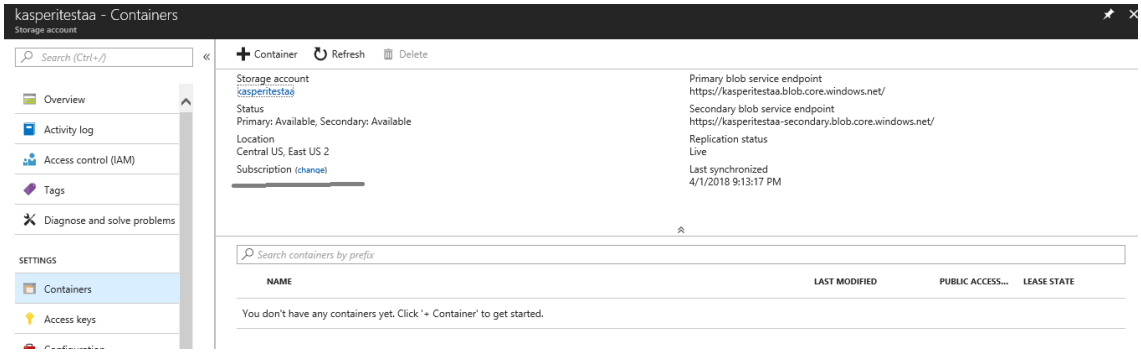


KUVA 20. Storage Account -resurssin luonti.

Kun Storage Account on luotu, voidaan navigoida sen kuvan 21 mukaiselle yleiskatsaus-sivulle. Vasemmalta valikosta siirrytään kuvan 22 mukaiselle Containers-sivulle. Containers-sivulla luodaan uusi Container yläpalkista.

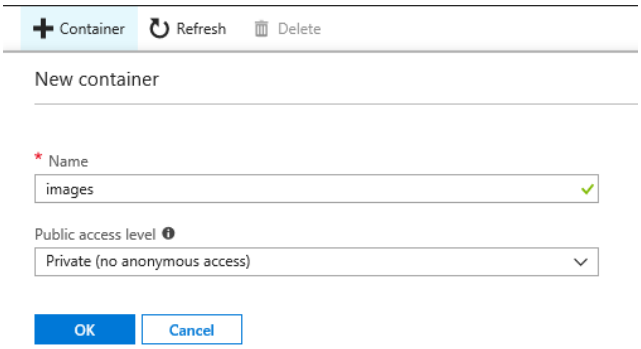


KUVA 21. Storage Account -resurssin yleiskatsaus-sivu.



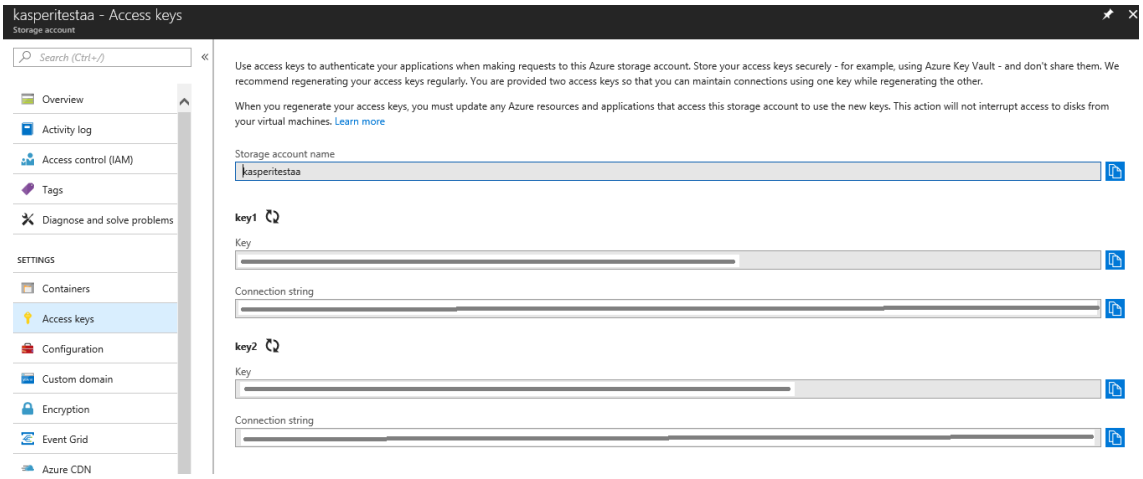
KUVA 22. Storage Account -resurssin Containers-sivu.

Asetetaan kuvan 23 mukaisesti Containerille nimi ja valitaan pääsytasoksi Private, koska käsittelemme tallennettavaa dataa aina taustajärjestelmistä. Jos data pitäisi olla asiakas-skriptillä haettavissa web-sivulle, pitäisi pääsytasoksi asettaa Blob. Blob-pääsytasolla lukuoikeus olisi kaikille avoin.



KUVA 23. Uuden Containerin luonti.

Siirrytään Access keys -sivulle. Kopioidaan Connection string eli valmis yhteyskonfiguraatio kohdasta key1 (kuva 24). Liitetään yhteyskonfiguraatio sovelluksen appSettings-elementtiin yhdeksi asetukseksi (kuva 25).



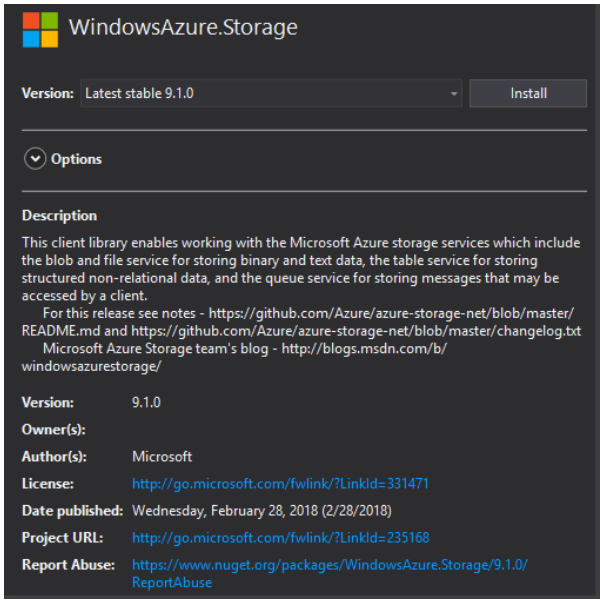
KUVA 24. Storage Account -resurssin pääsyavain-sivu.

```
12  <appSettings>
13    <add key="StorageAccountConnectionString" value="***************" />
14  </appSettings>
```

KUVA 25. appSettings-elementti konfiguraatiotiedostossa.

Asennetaan NuGet-paketti Microsoft.WindowsAzure.ConfigurationManager (kuva 26). Paketti sisältää kirjaston Storage Account -yhteyskonfiguraation parsimiseen. Lisäksi asennetaan WindowsAzure.Storage-niminen NuGet-paketti, koska se sisältää oleellisia luokkia Azure Storage Accountin käyttöön (kuva 27).

KUVA 26. Microsoft.WindowsAzure.ConfigurationManager-paketti.

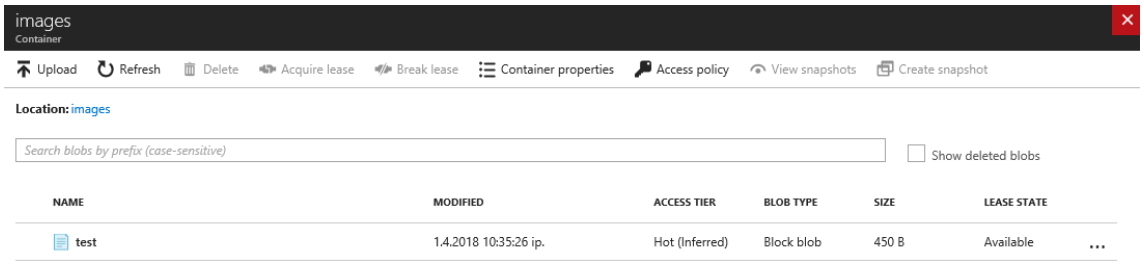


KUVA 27. WindowsAzure.Storage-paketti.

Kokeillaan lähettää kuvatiedosto portaalissa luotuun Containeriin. Ensin parsitaan yhteyskonfiguraatio ja luodaan rajapintaolio. Rajapintaoliolla luodaan nimen perusteella viittaus Containeriin, josta luodaan nimen perusteella viittaus Blobiin. Lopulta ladataan paikallinen tiedosto palvelimelle. Kun koodi suoritetaan web-sovelluksessa, voidaan portaalista tarkistaa, että kaikki sujui odotetusti ja Containeriin ilmestyi sisältöä. (61.) (Kuvat 28 ja 29.)

```
20          //parsitaan yhteyskonfiguraatiosta tiedot yhteyden luomiseen
21          var storageAccount = CloudStorageAccount.Parse(
22              CloudConfigurationManager.GetSetting("StorageAccountConnectionString"));
23
24          //luo blob rajapintaolio
25          var cloudBlobClient = storageAccount.CreateCloudBlobClient();
26
27          //luodaan viittaus Containeriin joka on nimetty 'images'
28          var cloudBlobContainer = cloudBlobClient.GetContainerReference("images");
29
30          //luodaan viittaus Block Blobiin joka on nimetty 'test'
31          var cloudBlockBlob = cloudBlobContainer.GetBlockBlobReference("test");
32
33          //uploadataan testikuva blobiin
34          cloudBlockBlob.UploadFromFile(@"C:\Users\kasperi\Downloads\test.png");
```

KUVA 28. Esimerkkikoodi tiedoston lataamisesta Blob Storage -palveluun.



KUVA 29. Containerin sisältölistaus.

Containerista voidaan hakea Blob-listaus, jolloin saadaan lista IListBlobItem-rajapinnan toteuttavia olioita. Testataan listauksen hakemista tulostamalla rivit sivulle (kuva 30). Haetaan Containerin riveistä Block Blobit, ja ladataan Blob paikalliseen tiedostoon (kuva 31).

```
37          //haetaan Blob listaus Containerista
38          var items = cloudBlobContainer.ListBlobs();
39
40          //testataan printtaamalla tuloksista jotain sivulle
41          TestListView.DataSource = items.Select(i => new {Url = i.Uri.ToString()});
42          TestListView.DataBind();
```

KUVA 30. Esimerkkikoodi Containerin sisällön listauksen hakemiseen.

```
43          //otetaan Containerin listauksesta vain blobit suodattamalla tyypin mukaan, testataan latausta listan ensimmäisellä blobilla
44          var cloudBlockBlobs = items.Where(i => i.GetType() == typeof(CloudBlockBlob)).Select(b => b as CloudBlockBlob).ToList();
45          if (!cloudBlockBlobs.Any()) return;
46          var cloudBlockBlob = cloudBlockBlobs.First();
47          cloudBlockBlob.DownloadToFile(@"C:\Users\kasperi\Downloads\test_downloaded_from_blob_storage.png", FileMode.Create);
```

KUVA 31. Esimerkkikoodi Blobin lataamisesta.

Nyt kun Blob Storage -palvelun käyttö on testattu ja todettu toimivaksi, pitää kehitettävän sovelluksen tiedostojenkäsittelykirjasto muokata. Kirjastosta pitää tiedostojen tallennus ja luku muuttaa käyttämään Blob Storage -kirjastoa ylläolevan koodin pohjalta.

5.5 Tietokanta

Yrityksen sovelluksessa käytetään datan tallennukseen Microsoft SQL Server -palvelimen tietokantaa. Tietokantapalvelin on samalla koneella, jolloin ei ole ollut tarvetta yhteyden salaukselle. Azuresta löytyvä Azure SQL Database -tietokantapalvelu on onneksi niin lähelle Microsoft SQL Serverin kaltainen, että sovellukseen vaadittavat muutokset ovat minimaaliset. Azure SQL Database -palvelu on kumminkin rajoitetumpaa esimerkiksi sallittujen T-SQL-lauseiden osalta, joten nämä rajoitukset piti selvittää. Tässä tapauksessa rajoituksista ei ollut vaikutusta sovelluksen tietokannan käyttöön, joten Azure SQL Database -palvelu soveltui täysin käytettäväksi. (62.)

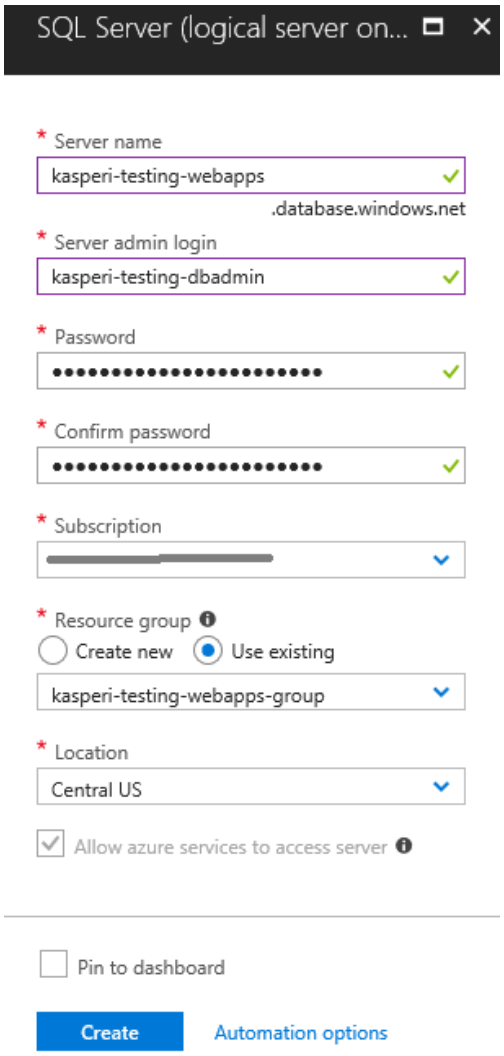
5.5.1 SQL-yhteyden salaus

Yhteys SQL-tietokantaan täytyy salata riittävällä tavalla, kun liikutaan julkisessa verkossa hajautettujen palvelimien väleillä. Tietokannan luomisen jälkeen Azure generoi SQL-yhteyden konfiguroinnin, jonka voi suoraan kopioida omaan konfiguraatiotiedostoon. Tärkeintä yhteyden konfiguroinnissa on, että yhteys määritellään salattavaksi Encrypt-arvolla (’Encrypt=True’) ja että palvelinsertifikaattiin ei luoteta TrustServerSertificate-arvolla (’TrustServerSertificate=False’). Azure SQL Database ei edes hyväksy suojaamattomia yhteyksiä, joten vahinkoa ei pääse käymään. (63.)

TrustServerSertificate-arvo siis määrittelee sen, tarkistetaanko palvelimen sertifikaatti vai käytetäänkö sitä sokeasti vain salaukseen. Julkisissa SQL-palveluissa pitää aina asettaa ’TrustServerSertificate=False’, kun taas jos käyttäisi omaa SQL-palvelinta omalla dedikoidulla palvelimella ja itse generoidulla sertifikaatilla, arvo tulisi olla ’TrustServerSertificate=True’, jotta yhteys toimisi. (63.)

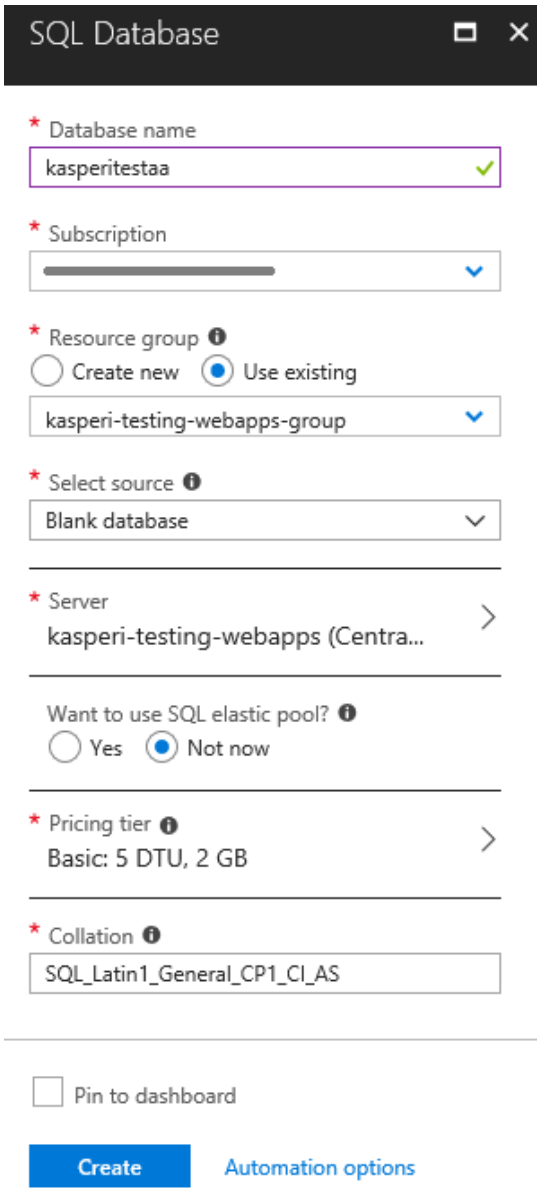
5.5.2 Azure SQL Database -palvelun konfigurointi

Azuren web-portaalista haetaan termillä ’sql servers’, ja mennään SQL Servers -sivulle. Luodaan uusi SQL Server -resurssi. Asetetaan nimi, pääkäyttäjän tunnus, pääkäyttäjän salasana ja yhdistetään tilaukseen. Lisätään Resource Group ja sijainti, joihin laitetaan samat, mitä App Service -resurssiinkin. Toiminto siis luo virtuaalisen SQL-palvelimen. (Kuva 32.)



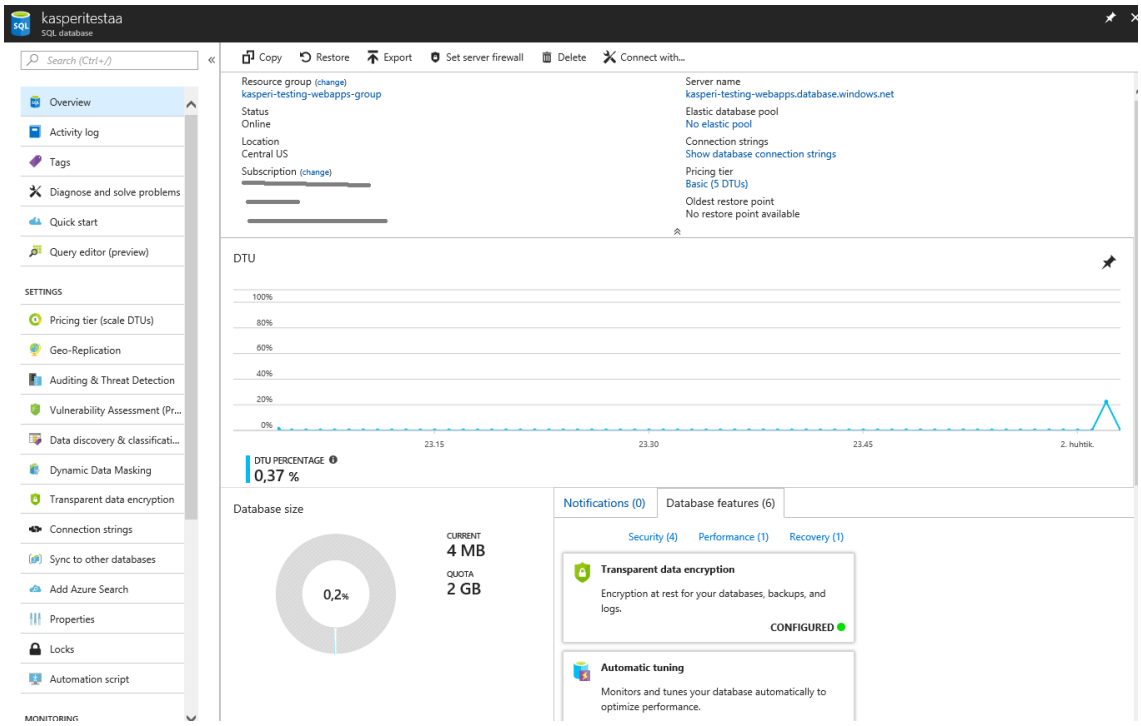
KUVA 32. SQL Server -resurssin luonti.

Sen jälkeen haetaan termillä 'sql databases', ja mennään SQL Databases -sivulle. Luodaan SQL Database -resurssi. Asetetaan nimi, tilaus, Resource Group ja hintaluokka. Palvelimeksi valitaan äskettäin luotu SQL Server -resurssi. Toiminto luo tietokannan virtuaaliseen SQL-palvelimeen. (Kuva 33.)



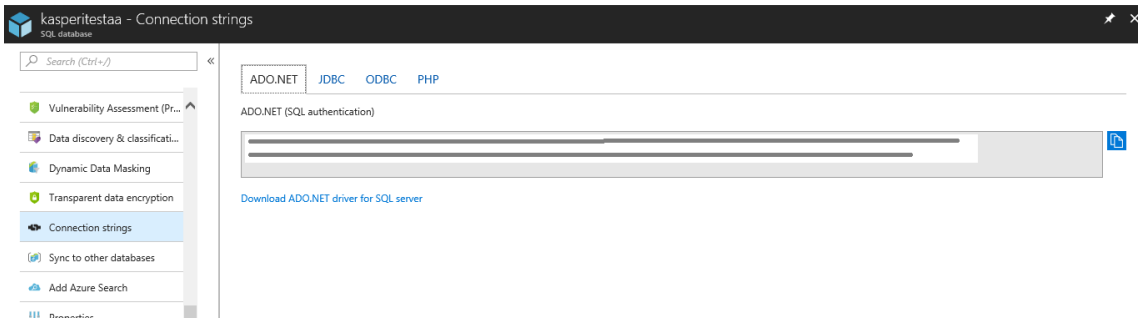
KUVA 33. SQL Database -resurssin luonti.

Kun tietokanta on luotu, voidaan navigoida sen yleiskatsaus-sivulle, jossa Azuren web-portaalille tuttuun tapaan näytetään perustiedot sekä erilaisia kuvaajia tietokannasta (kuva 34).



KUVA 34. SQL Database -resurssin yleiskatsaus-sivu.

Yleiskatsaus-sivulta Connection strings -linkistä päästään näkymään, mistä saadaan valmiiksi generoitu yhteyskonfigurointi. Tämä konfigurointi pitää asettaa sovelluksen konfiguraatiotiedostoon entisen konfiguraation tilalle. Käytännössä siinä siis muuttuu vain palvelimen osoite, tunnus ja salasana sekä yhteydensalausarvot. (Kuva 35.)



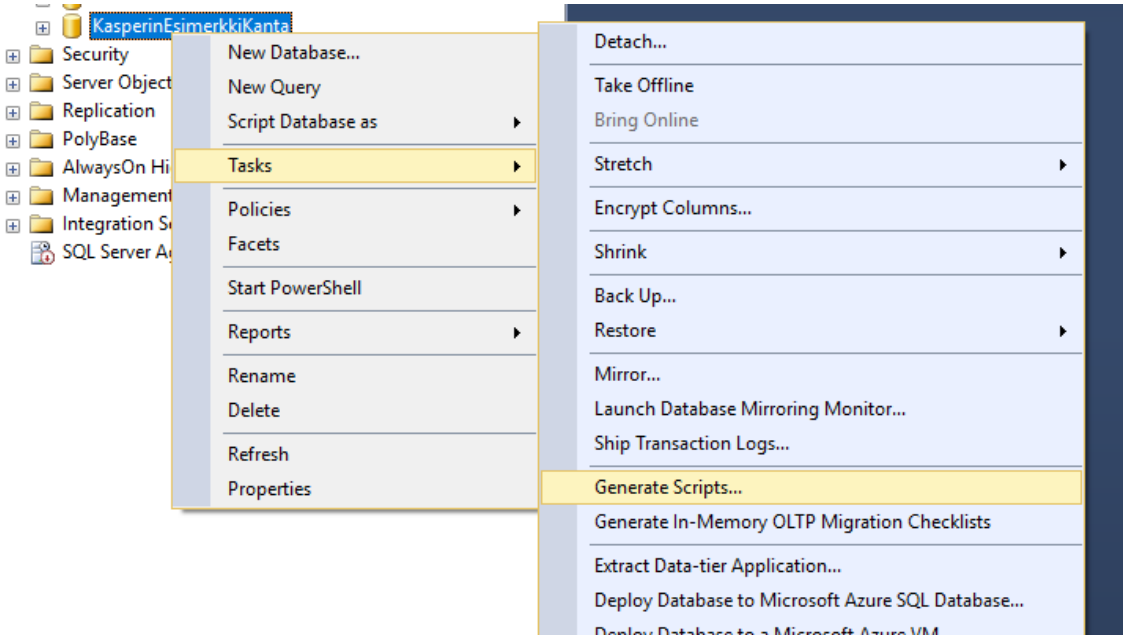
KUVA 35. SQL Database -resurssin Connection strings-sivu.

Aikaisemmassa kohdassa kopioitu yhteyskonfigurointi asetetaan connectionStrings-elementtien sisälle aikaisempien yhteyskonfiguraatioiden tilalle. Tunnus ja salasana pitää erikseen kopioida, sillä ne eivät ole valmiina generoidussa konfiguraatiossa. Jokaiselle eri tietokantayhteydelle pitää olla oma rivinsä konfiguraatiossa. (Kuva 36.)

```
12  <connectionStrings>
13      <add name="KasperiTestaa" connectionString="***************" providerName="System.Data.EntityClient" />
14  </connectionStrings>
```

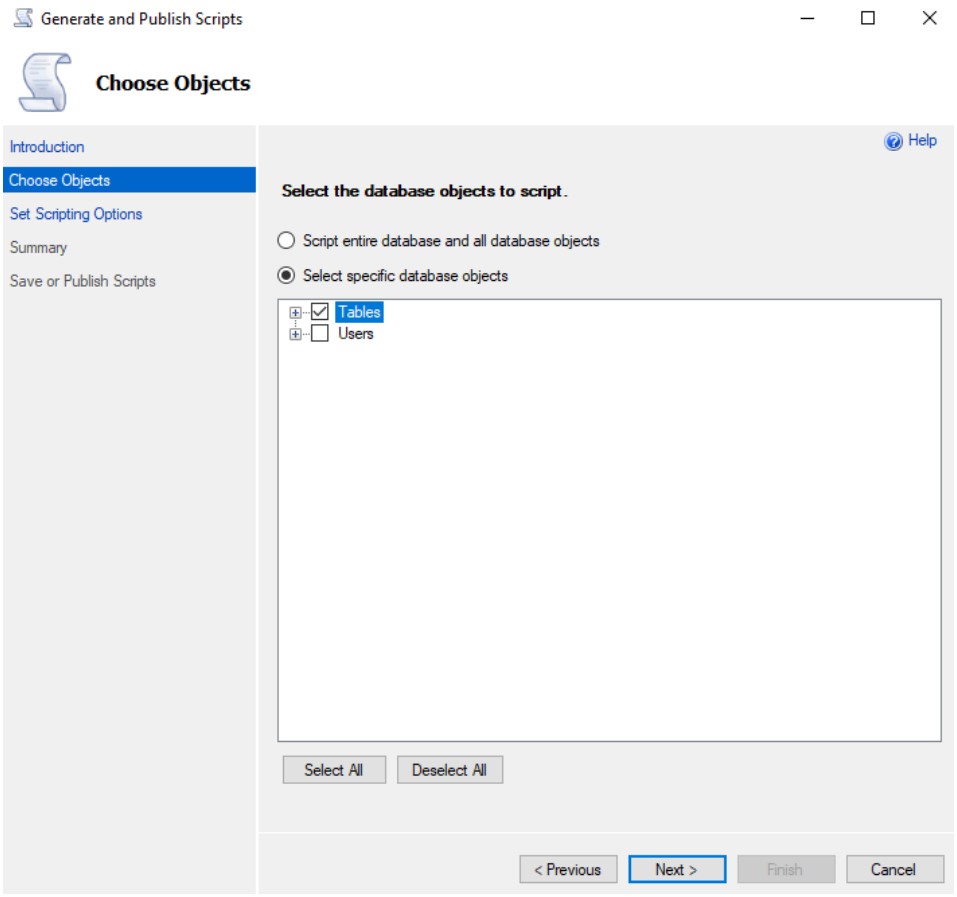
KUVA 36. connectionStrings-elementti konfiguraatiotiedostossa.

Nyt kun tietokanta on luotu ja valmis käytettäväksi, pitää vielä kopioida olemassa oleva data vanhalta SQL-palvelimelta. Helpoiten tämä hoituu generoimalla skripti SQL Server Management Studiolla. Otetaan siis yhteys SQL-palvelimeen Management Studiolla, ja valitaan oikean tietokannan kohdalta pudotusvalikosta Tasks ja Generate scripts -valinnat. (Kuva 37.)



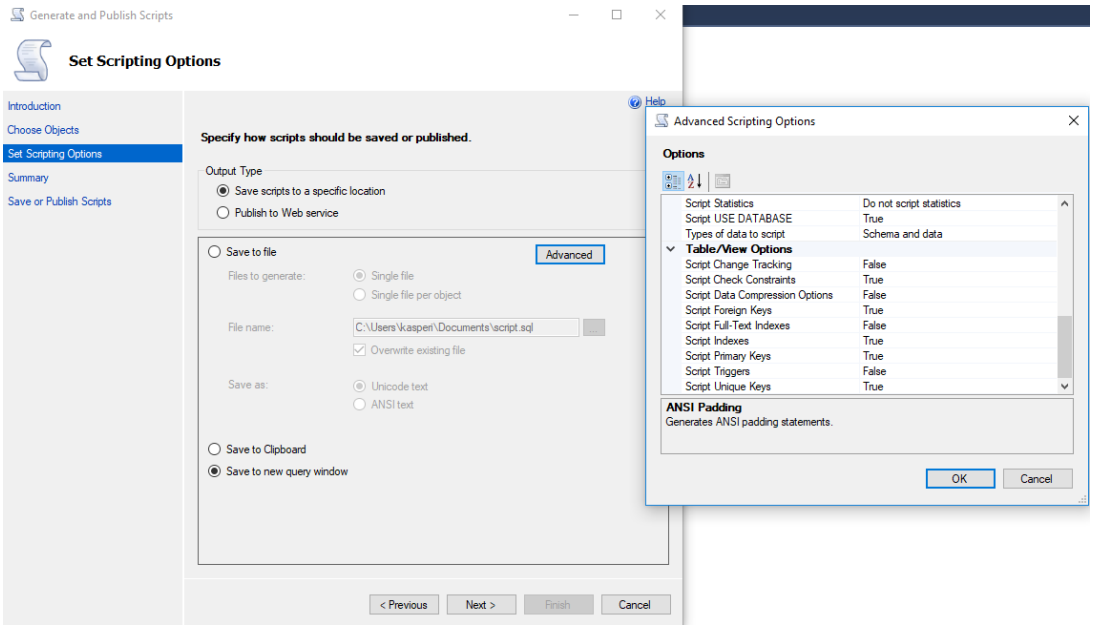
KUVA 37. Skriptiengenerointitoiminto pudotusvalikosta SQL Server Management Studiossa.

Valitaan kaikki tietokannan taulut kopioitavaksi. Käyttäjiä taas ei kannata kopioida, koska Azure SQL Database -palvelussa käyttäjät eivät toimi täysin samalla tavalla kuin perinteisessä SQL Serverissä. (Kuva 38.)



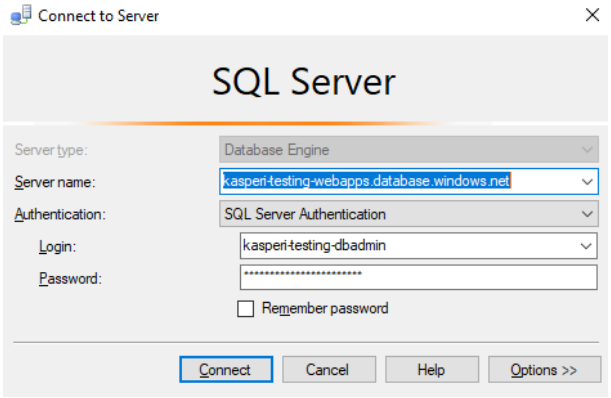
KUVA 38. Skriptingenerointitoiminnon sisällön valinta.

Seuraavassa vaiheessa valitaan generoitu skripti tulostumaan uuteen kyselyikkunaan Save to new query window -valinnasta. Edistyneet asetukset on syytä käydä myös huolellisesti läpi. Tässä tapauksessa asetetaan toiminto generoimaan myös data pelkän skeeman lisäksi valitsemalla Types of data to script -arvoksi ’Schema and data’. Valitaan myös indeksit mukaan generoitavaan skriptiin Script indexes -arvolla. (Kuva 39.)



KUVA 39. Skriptingenerointitoiminnon asetukset.

Kun tietokannan koko sisältö on generoitu skriptiin, otetaan yhteys Azuressa olevaan tietokantaan ja suoritetaan skripti. Azure SQL Database -palvelun tietokantoihin voi ottaa samalla tavalla yhteyden SQL Server Management Studiolla. Palvelimen nimi saadaan portaalista yleiskatsaus-sivulta. Tunnukseksi ja salasanaksi asetetaan tietokantaa luodessa valitut arvot. Palvelimen pääkäyttäjän tunnuksen löytää myös web-portaalista Properties-sivulta. (Kuva 40.)



KUVA 40. SQL Server Management Studion yhteydenluonti-ikkuna.

Tarkistetaan että aiemmin luotu tietokanta näkyy nyt listauksessa. Avataan kyselyeditori pudotusvalikosta esimerkiksi tietokannan kohdalta. Liitetään kyselyeditoriin generoitu skripti ja muutetaan skriptistä aiemman tietokannan nimi USE-lauseesta uuteen tietokannan nimeen. Lopuksi suoritetaan skripti ja seurataan ulostuloa mahdollisten virheiden varalta. Jos skriptin suoritus onnistuu, tarkistetaan, että yhteys uuteen tietokantaan toimii suorittamalla sovellus. Jos yhteys tietokantaan saadaan ja dataa tulee sovellukseen normaalisti, voidaan todeta uusi tietokanta toimivaksi.

6 POHDINTA

Opinnäytetyön tarkoituksena oli saada yrityksen web-sovellus toimimaan Microsoft Azure -pilvipalvelussa. Azure ei ollut ennestään riittävän tuttu, joten tavoitteen täyttämiseksi piti ensin perehtyä Azuren palveluihin sekä miettiä, mitä niistä voisi hyödyntää ja miten. Myös yleisesti ottaen web-sovelluksen hostaamisen yksityiskohdista hajautetuissa ympäristöissä oli opittavaa.

Azuresta löytyi todella runsaasti eri palveluita, jotka kiihdyttävät monella tapaa ohjelmistokehitystä. Jo pelkästään kehitysympäristön pitäminen Azuren palveluissa vähentäisi testiympäristöjen julkaisuun kuluvaa aikaa ja näin tehostaisi projektien ajankäyttöä. Modernien arkkitehtuurien maailmanlaajuiseen saatavuuteen yltävät hajautetut palvelinkeskukset houkuttelevat ottamaan Azuren palveluja tuotantokäyttöön.

Käytin työn aikana paljon Azuren web-portaalia. Ajatustapa Azuren resursseista ja niiden hinnoitteluista tuli tutuksi. Opin App Services -alustan toiminnallisuuksien hyödyntämistä käytännössä ja julkaisu alustalle tuntuu nyt jopa helpommalta kuin perinteiseen palvelimeen.

Kävin läpi eri vaihtoehdot ASP.NET-sovelluksen istuntotilan tallentamiseen sekä yleisesti että Azuren palveluissa. Tämän käytännöllisen esimerkin kautta erot dedikoidun palvelimen ja hajautetun palvelinympäristön välillä on helpompi ymmärtää. Perehdyin kolmeen eri vaihtoehtoiseen palveluun istuntotilan tallentamiseen Azuressa: Azure SQL Database, Azure Redis Cache ja Azure Table Storage. Näistä valitsin Azure Redis Cache -palvelun suorituskyvyn takia. Opin Azure Redis Cache -palvelun hyödyntämisen sekä yleisesti ottaen Redis-palvelimen toimintalogiikan.

Tiedostojen tallentamiseen Azuressa valitsin Blob Storage -palvelun, koska se oli mielestäni ilmeisen selvä ratkaisu ilman varteenotettavia vaihtoehtoja. Blob Storage -palvelu on suunniteltu juurikin tämänkaltaiseen tarkoitukseen eli web-sovelluksen käyttäjän tiedostojen tallennukseen ja hakuun. Perehdyin saatavuus- ja pääsytasoihin, joilla määritellään, kuinka nopeasti ja laajalla saatavuudella sisältö on jaossa sekä kenellä on oikeus muokata ja hakea sisältöä. En ollut aiemmin tutustunut Blob Storage -palveluun, mutta työn aikana oivalsin sille montakin käyttötapaa. Monipuolisuutensa ansiosta tulen varmasti käyttämään palvelua tämän työn jälkeenkin muissa projekteissa.

Myös Azuren tietokantapalvelut tulivat tutuksi, erityisesti Azure SQL Database -palvelu. Onnekseni huomasin, että palvelun tietokannat ovat todellakin hyvin samankaltaisia kuin Microsoft SQL Server -palvelimen tietokannat. Kumpiakin voidaan hallinnoida samalla ohjelmalla SQL Server Management Studiolla, ja suurin osa kyselylauseistakin toimii samalla tavalla. Azure SQL Database -palvelun älykkäät toiminnot kuitenkin yllättivät minut täysin uskomattoman tehokkailla automaattisilla optimointitoiminnoilla. Tietokantapalveluun tutustuessa tuli perehdyttyä myös yleisesti SQL-tietokantayhteyksien salaamiseen, joka on välttämätön hajautetuissa palvelinympäristöissä.

Itse sovelluksen kehitystyö onnistui hyvin ja tavoitteet toteutuivat. Pääasiallisiin muutoksia vaativiin sovelluksen ominaisuuksiin saatiin useita ratkaisuvaihtoehtoja, joista parhaaksi osoittautuva toteutettiin.

Azuren palvelut kehittyvät varsin vauhdikkaasti ja jopa tämän työn aikana huomasin joitain muutoksia tapahtuneen. Voisi olettaa, että tässäkin sovelluksessa hyödynnettävät palvelut muuttuvat oleellisesti seuraavien vuosien aikana. Myös uusia vaihtoehtoisia palveluita voi tulla joko aikaisempien korvaavaksi tai lisäksi. Opiskelu ja selvitystyö tulee siis olemaan jatkuvaa tämänkin projektin jälkeen.

LÄHTEET

1. Microsoft. 2015. Introduction to the C# Language and the .NET Framework. Saatavissa: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework. Hakupäivä 24.3.2018.

2. Microsoft. 2017. Overview of the .NET Framework. Saatavissa: https://docs.microsoft.com/en-us/dotnet/framework/get-started/overview. Hakupäivä 24.3.2018.

3. Microsoft. 2014. What is Web Forms. Saatavissa: https://docs.microsoft.com/en-us/aspnet/web-forms/what-is-web-forms. Hakupäivä 24.3.2018.

4. Avery, James 2005. What Is Visual Studio. Saatavissa: http://archive.oreilly.com/pub/a/windows/2005/08/22/whatisVisualStudio.html. Hakupäivä 24.3.2018.

5. Microsoft. 2018. Visual Studio IDE overview. Saatavissa: https://docs.microsoft.com/en-us/visualstudio/ide/visual-studio-ide. Hakupäivä 24.3.2018.

6. Atlassian. 2018. What is Git. Saatavissa: https://www.atlassian.com/git/tutorials/what-is-git. Hakupäivä 24.3.2018.

7. Strumpflohner, Juri 2015. Git Explained: For Beginners. Saatavissa: https://juristr.com/blog/2013/04/git-explained/. Hakupäivä 24.3.2018.

8. GitLab. 2018. Features. Saatavissa: https://about.gitlab.com/features/. Hakupäivä 24.3.2018.

9. McCabe, John & Windows Server team 2016. Introducing Windows Server 2016. Saatavissa: https://aka.ms/WinServ16/StndPDF. Hakupäivä 24.3.2018.

10. Microsoft. 2018. IIS Overview. Saatavissa: https://www.iis.net/overview. Hakupäivä 24.3.2018.

11. PCWDLD.com. 2018. What is IIS? A Basic Tutorial of the Windows Web Server. Saatavissa: https://www.pcwdld.com/what-is-iis. Hakupäivä 24.3.2018.

12. Microsoft. 2007. Introduction to IIS Architectures. Saatavissa: https://docs.microsoft.com/en-us/iis/get-started/introduction-to-iis/introduction-to-iis-architecture. Hakupäivä 24.3.2018.

13. Microsoft. 2007. IIS Web Server Overview. Saatavissa: https://docs.microsoft.com/en-us/iis/get-started/introduction-to-iis/iis-web-server-overview. Hakupäivä 24.3.2018.

14. w3schools.com. 2018. Introduction to SQL. Saatavissa: https://www.w3schools.com/sql/sql_intro.asp. Hakupäivä 25.3.2018.

15. tutorialspoint. 2018. MS SQL Server – Overview. Saatavissa: https://www.tutorialspoint.com/ms_sql_server/ms_sql_server_overview.htm. Hakupäivä 25.3.2018.

16. Microsoft. 2017. Database Engine Instances (SQL Server). Saatavissa: https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/database-engine-instances-sql-server. Hakupäivä 25.3.2018.

17. Microsoft. 2017. SQL Server Agent. Saatavissa: https://docs.microsoft.com/en-us/sql/ssms/agent/sql-server-agent. Hakupäivä 25.3.2018.

18. Microsoft. 2018. SQL Server Browser Service. Saatavissa: https://technet.microsoft.com/en-us/library/ms181087(v=sql.105).aspx. Hakupäivä 25.3.2018.

19. Microsoft. 2017. SQL Server Management Studio (SSMS). Saatavissa: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms. Hakupäivä 25.3.2018.

20. tutorialspoint. 2018. T-SQL Tutorial. Saatavissa: https://www.tutorialspoint.com/t_sql/index.htm. Hakupäivä 25.3.2018.

21. Walldén, Jan 2018. Miksi myös sinun kannattaa siirtyä pilveen. Saatavissa: http://starcut.com/miksi-myos-sinun-kannattaa-siirtya-pilveen/. Hakupäivä 25.3.2018.

22. Go, Gregory 2017. Overview of the Major Types of Web Hosting Services for Your Online Business. Saatavissa: https://www.thebalance.com/types-of-web-hosting-services-2532072. Hakupäivä 25.3.2018.

23. Reese, Nick 2017. VPS vs Dedicated: What Web Hosting Does Your Site Need. Saatavissa: https://blogging.com/wordpress-hosting/vps-vs-dedicated/. Hakupäivä 25.3.2018.

24. Webhotellivertailu2. 2018. Mitä palvelintyyppi, eli virtuaalipalvelin tai dedikoitu palvelin tarkoittaa. Saatavissa: https://www.webhotellivertailu2.fi/mita-palvelintyyppi-eli-virtuaalipalvelin-tai-dedikoitu-palvelin-tarkoittaa/. Hakupäivä 17.3.2018.

25. Wood, Kevin 2016. What's The Difference Between VPS vs Dedicated Server Hosting. Saatavissa: https://www.hostgator.com/blog/difference-between-vps-vs-dedicated-server-hosting/. Hakupäivä 17.3.2018.

26. rackspace. 2018. What Are the Differences Between Cloud Computing vs. Virtual Private Servers (VPS). Saatavissa: https://www.rackspace.com/library/cloud-computing-vs-virtual-private-servers. Hakupäivä 17.3.2018.

27. Eronen, Heidi 2016. IaaS, PaaS, SaaS? Mikä pilvipalvelu sopii yrityksellesi. Saatavissa: https://blog.planeetta.net/iaas-paas-saas. Hakupäivä 17.3.2018.

28. Wood, Kevin 2016. Cloud vs Dedicated Hosting – Which Is Right For Me. Saatavissa: https://www.hostgator.com/blog/cloud-vs-dedicated-hosting/. Hakupäivä 18.3.2018.

29. interoute. 2018. WHAT IS CLOUD HOSTING. Saatavissa: https://www.interoute.com/what-cloud-hosting. Hakupäivä 18.3.2018.

30. Heikinmäki, Antti 2017. Mikä on pilvipalvelu ja mitä hyötyä siitä on minulle. Saatavissa: http://www.controla.fi/blogi/mika-on-pilvipalvelu-ja-mita-hyotya-siita-on-minulle. Hakupäivä 18.3.2018.

31. Microsoft. 2018. What is IaaS. Saatavilla: https://azure.microsoft.com/en-us/overview/what-is-iaas/. Hakupäivä 18.3.2018.

32. interoute. 2018. WHAT IS IAAS. Saatavilla: https://www.interoute.com/what-iaas. Hakupäivä 18.3.2018.

33. Bergius, Kimmo 2014. Mikä se Azure oikein on. Saatavilla: https://www.sulava.com/mika-se-azure-oikein/. Hakupäivä 28.3.2018.

34. Microsoft. 2018. Azure solutions. Saatavilla: https://azure.microsoft.com/en-gb/solutions/. Hakupäivä 28.3.2018.

35. Microsoft. 2018. What is Azure. Saatavilla: https://azure.microsoft.com/en-gb/overview/what-is-azure/. Hakupäivä 28.3.2018.

36. Microsoft. 2018. Azure Resource Manager overview. Saatavilla: https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview. Hakupäivä 29.3.2018.

37. Microsoft. 2017. What is an Azure resource group. Saatavilla: https://docs.microsoft.com/en-us/azure/architecture/cloud-adoption-guide/adoption-intro/resource-group-explainer. Hakupäivä 29.3.2018.

38. Microsoft. 2017. Basic web application. Saatavilla: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/app-service-web-app/basic-web-app. Hakupäivä 29.3.2018.

39. Finn, Aidan 2017. What Is the Azure App Service. Saatavilla: https://www.petri.com/azure-app-service. Hakupäivä 29.3.2018.

40. Microsoft. 2018. Web Apps. Saatavilla: https://azure.microsoft.com/en-us/services/app-service/web/. Hakupäivä 30.3.2018.

41. Microsoft. 2016. Set up staging environments in Azure App Service. Saatavilla: https://docs.microsoft.com/en-us/azure/app-service/web-sites-staged-publishing. Hakupäivä 30.3.2018.

42. Microsoft. 2017. Azure App Service plan overview. Saatavilla: https://docs.microsoft.com/en-us/azure/app-service/azure-web-sites-web-hosting-plans-in-depth-overview. Hakupäivä 30.3.2018.

43. Microsoft. 2018. Azure products. Saatavilla: https://azure.microsoft.com/en-us/services/. Hakupäivä 30.3.2018.

44. Microsoft. 2017. Azure Redis Cache FAQ. Saatavilla: https://docs.microsoft.com/en-us/azure/redis-cache/cache-faq. Hakupäivä 30.3.2018.

45. Microsoft. 2018. What is the Azure SQL Database service. Saatavilla: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-technical-overview. Hakupäivä 30.3.2018.

46. Microsoft. 2018. Azure Redis Cache. Saatavilla: https://azure.microsoft.com/en-us/services/cache/. Hakupäivä 1.4.2018.

47. Redis-kehittäjäyhteisö. 2018. Introduction to Redis. Saatavilla: https://redis.io/topics/introduction. Hakupäivä 1.4.2018.

48. Microsoft. 2018. Table Storage. Saatavilla: https://azure.microsoft.com/en-us/services/storage/tables/. Hakupäivä 30.3.2018.

49. Microsoft. 2018. Storage. Saatavilla: https://azure.microsoft.com/en-us/product-categories/storage/. Hakupäivä 30.3.2018.

50. Microsoft. 2018. Introduction to Blob storage. Saatavilla: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction. Hakupäivä 2.4.2018.

51. Microsoft. 2018. Application Insights. Saatavilla: https://azure.microsoft.com/en-us/services/application-insights/. Hakupäivä 30.3.2018.

52. Microsoft. 2017. Overview of Application Insights for DevOps. Saatavilla: https://docs.microsoft.com/en-us/azure/application-insights/app-insights-detect-triage-diagnose. Hakupäivä 30.3.2018

53. Microsoft. 2017. Walkthrough: Converting a Web Site Project to a Web Application Project in Visual Studio. Saatavilla: https://msdn.microsoft.com/en-us/library/aa983476.aspx. Hakupäivä 18.9.2017.

54. Microsoft. 2017. Web Application Projects versus Web Site Projects in Visual Studio. Saatavilla: https://msdn.microsoft.com/en-us/library/dd547590.aspx. Hakupäivä 18.9.2017.

55. Microsoft. 2018. ASP.NET Session State Overview. Saatavilla: https://msdn.microsoft.com/en-us/library/ms178581.aspx. Hakupäivä 31.3.2018.

56. Microsoft. 2018. Session-State Modes. Saatavilla: https://msdn.microsoft.com/en-us/library/ms178586.aspx. Hakupäivä 31.3.2018.

57. Ahuwanya, Sunny 2009. Peer to Peer ASP.NET State Server. Saatavilla: https://www.codeproject.com/Articles/41726/Peer-to-Peer-ASP-NET-State-Server. Hakupäivä 31.3.2018.

58. Microsoft. 2018. Implementing a Session-State Store Provider. Saatavilla: https://msdn.microsoft.com/en-us/library/ms178587.aspx. Hakupäivä 31.3.2018.

59. Microsoft. 2017. ASP.NET Session State Provider for Azure Redis Cache. Saatavilla: https://docs.microsoft.com/en-us/azure/redis-cache/cache-aspnet-session-state-provider. Hakupäivä 31.3.2018.

60. Microsoft. 2017. How to configure Azure Redis Cache. Saatavilla: https://docs.microsoft.com/en-us/azure/redis-cache/cache-configure#configure-redis-cache-settings. Hakupäivä 1.4.2018.

61. Microsoft. 2018. Upload image data in the cloud with Azure Storage. Saatavilla: https://docs.microsoft.com/en-us/azure/storage/blobs/storage-upload-process-images. Hakupäivä 31.3.2018.

62. Microsoft. 2018. Choose a cloud SQL Server option: Azure SQL (PaaS) Database or SQL Server on Azure VMs (IaaS). Saatavilla: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-paas-vs-sql-server-iaas. Hakupäivä 31.3.2018.

63. Microsoft. 2018. Securing your SQL Database. Saatavilla: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-security-overview. Hakupäivä 31.3.2018.