OULUN SEUDUN
AMMATTIKORKEAKOULU

Mikko Keränen

WEB-PALVELUN SKAALAUS ISOILLE KÄYTTÄJÄMÄÄRILLE

WEB-PALVELUN SKAALAUS ISOILLE KÄYTTÄJÄMÄÄRILLE

Mikko Keränen
Opinnäytetyö
Kevät 2013
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikka koulutusohjelma, ohjelmistotekniikka

Tekijä: Mikko Keränen
Opinnäytetyön nimi: Web-palvelun skaalaus isoille käyttäjämäärille
Työn ohjaaja: Kari Laitinen
Työn valmistumislukukausi ja -vuosi: kevät 2013 Sivumäärä: 33 + 1 liite

---

Työ tehtiin WhileOnTheMove Oy:lle, joka tarjoaa mm. varhaiskasvatukselle mobiilia NFC-sovellusta nimeltä Daisy. Sovellus on tarkoitettu lasten ja työntekijöiden paikallaolon seurantaan päiväkodeissa. Asiakkaiden käyttäjämäärät saattavat vaihdella 700:n ja 100 000:n välillä, joten tässä työssä selvitettiin, miten saadaan sovelluksesta niin skaalautuva, että se pystyy palvelemaan isojakin asiakkaita.

Työssä selvitettiin, mitä tarvittavan palvelinarkkitehtuurin rakentaminen vaatii, jos kaikki tehdään itse. Toisena vaihtoehtona selvitettiin pilvipalvelun käyttöä. Lopussa testataan, miten sovellusta voisi ajaa Amazon pilvipalvelussa.

Pienelle yritykselle paras tapa toteuttaa vaadittava ympäristö on selkeästi pilvipalvelun käyttö, koska silloin aloituskustannukset ovat pienet ja ylläpitoresursseja tarvitaan vähemmän.

Asiasanat: Daisy, pilvipalvelu, web-palvelu, palvelinarkkitehtuuri, mobiilisovellukset

ALKULAUSE

Kiitokset WhileOnTheMove Oy:n Kari Kivistölle ja Seppo Salowille opinnäytetyön aiheesta ja sen tekemiseen saaduista resursseista. Kiitokset myös työn ohjaajalle Kari Laitiselle. Työssä ei ole voitu mennä kovin tarkkoihin Daisy-järjestelmän yksityiskohtiin, koska ne ovat salassa pidettäviä asioita.

6.5.2013

Mikko Keränen

SISÄLLYS

SANASTO

API	Ohjelmointirajapinta, joka tarjoaa rajapinnan, jolla jotakin palvelua voi ohjata ohjelmallisesti.
DNS	Nimipalvelujärjestelmä, jolla tietty merkkijonomuotoinen osoite muunnetaan IP-osoitteeksi.
GET	HTTP-protokollan metodi, jolla välitetään esimerkiksi web-lomakkeen tietoja palvelimelle.
HTTP	Internetissä yleisesti käytetty sovellustason protokolla, jota käytetään tietojen siirtoon.
Index (tietokanta)	Indeksointi on SQL-tietokannoissa tapa optimoida hakujen nopeutta.
NFC	Near Field Communication on tekniikka, jossa tietoa siirretään langattomasti hyvin lyhyen matkan päästä.
TCP-portti	Tarkoittaa tietoliikenteessä porttia, josta tiettyä palvelua tarjotaan. TCP-porttien ansiosta voi yhdessä IP-osoitteessa olla useampi palvelu, joille jokaiselle on määritetty oma TCP-portti.
URL	Uniform Resource Identifier on tapa kertoa tietyn tiedon olinpaikka. Esimerkiksi web-sivujen osoitteet ovat URL-muodossa.

1 JOHDANTO

Daisy on varhaiskasvatuksen alalle suunnattu ohjelmisto, jota voidaan käyttää esimerkiksi päiväkodeissa lasten ja hoitajien läsnäolon seurantaan. Järjestelmään kuuluu NFC-lukulaitteella varustettussa mobiilipäätelaiteessa ajettava asiakassovellus ja palvelimella ajettava palvelinsovellus. Palvelinsovellus on keskitetty tietopankki, josta asiakassovellukset hakevat ja lähettävät tietoa eri asioista, kuten lasten läsnäolosta, viesteistä jne. Palvelinsovellus on toteutettu ASP.NET-teknologialla, joka tarkoittaa, että sovellusta suoritetaan web-palvelun alaisuudessa.

Asiakassovellukset keskustelevat palvelinsovellukselle käyttäen HTTP-protokollaa. Voidaan siis ajatella, että jokainen käytössä oleva mobiilipäätelaite on kuin internetissä surffaava henkilö, joka hakee yhtä www-sivua hyvin tiheällä aikavälillä eli painelee selaimen hae uudelleen -painiketta. Tämä aiheuttaa omat haasteensa palvelimessa ajettavalle web-palvelulle, jonka pitää pystyä vastaamaan kyselyihin mahdollisimman pienellä viiveellä.

Tämän työn tarkoituksena oli selvittää, miten Daisy-järjestelmään kuuluva palvelinalusta tulisi rakentaa siten, että sitä pystytään skaalaamaan isommille käyttäjämäärille, kuten 700-100 000 käyttäjälle. Haluttiin tutkia, millainen järjestelmäarkkitehtuuri tarvitaan. Lisäksi selvitettiin, voidaanko käyttää valmiita pilvipalveluita kuten Azure tai Amazon.

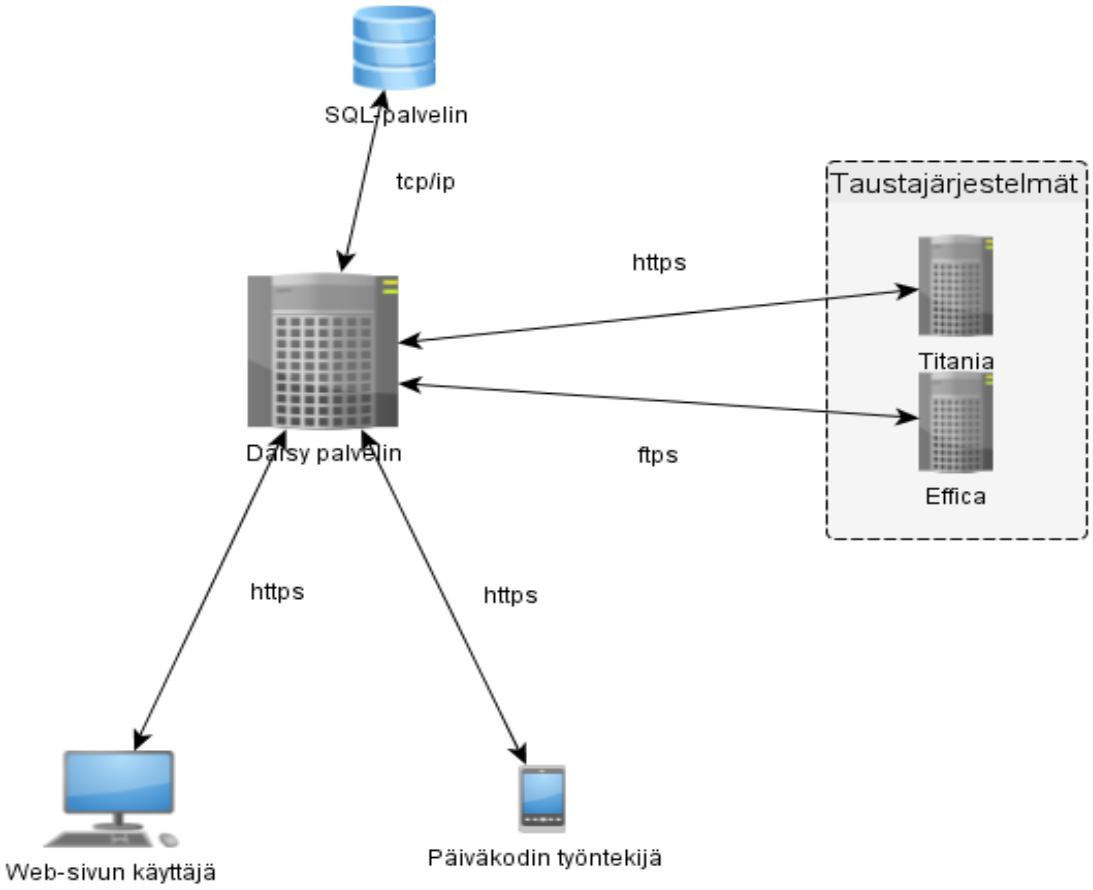
2 DAISY-JÄRJESTELMÄ

Daisy on WhileOnTheMove Oy:n kehittämä varhaiskasvatukselle suunnattu tuote. Tuote koostuu useasta eri osa-alueesta: mobiililaitteissa käytettävistä asiakassovelluksista, päiväkodin tai koulun henkilökunnalle sekä vanhemmille tarkoitetuista selainsovelluksista.

Daisy-mobiilisovellus koostuu mobiililaitteissa ajettavasta asiakassovelluksesta ja web-palvelimella ajettavasta palvelinsovelluksesta. Kaikki ajantasaiset tiedot sijaitsevat palvelimella, josta asiakassovellukset kyselevät niitä tarvittaessa ja lähettävät päivittyneitä tietoja. Sovellusten välinen kommunikointi tapahtuu HTTP-kyselyillä. Tämän takia web-palvelimen ja web-palvelimen käyttämän tietokannan suorituskyky on ratkaisevassa asemassa toiminnan kannalta: jos palvelin ei pysty vastaamaan kyselyihin nopeasti, alkaa asiakassovellus hidastella ja näyttää vanhentunutta tietoa.

Web-palvelimella ajetaan myös DaisyNet- ja DaisyManager-sovelluksia, jotka ovat palveluun kuuluvia web-sivustoja. Lisäksi palvelimen kanssa kommunikoivat erilaiset taustajärjestelmät, joista voidaan tuoda lasten ja henkilökunnan tietoja. Taustajärjestelmät eivät merkittävästi palvelinta kuormita, koska niiden kanssa kommunikointi tapahtuu muutaman kerran päivässä. Yleinen järjestelmän arkkitehtuuri on esitetty kuvassa 1.

Palvelinsovellus ja web-sivustot on toteutettu Microsoftin kehittämällä ASP.NET-teknologialla. Web-palvelinsovelluksena toimii Microsoft IIS 7.x ja tietokanta on myös Microsoftin tekniikkaa eli Microsoft SQL Server 2008. Asiakassovellus on toteuttu käyttäen Trolltechin alun perin kehittämää Qt-teknologiaa.



KUVA 1. Daisy-arkkitehtuurikuvaus

3 TIETOKANNAN JA WEB-PALVELIMEN PULLONKAULAT

Daisy-asiakassovelluksen ja palvelimen välinen kommunikointi perustuu HTTP-kyselyihin. Kyselyt tapahtuvat siten, että asiakassovellus ottaa yhteyttä tiettyyn URL-osoitteeseen. Kysely sisältää myös erilaisia GET-parametreja, joilla määritetään, mitä tietoja palvelimelta halutaan. Palvelin tekee kyselyn ja sen parametrien perusteella tarvittavat SQL-kyselyt kantaan sekä tekee muut tarvittavat muokkaustoimenpiteet tiedoille, jotka se palauttaa sovitun muotoisena tekstinä asiakassovellukselle. Näitä kyselyitä asiakassovellus tekee ennalta määritellyllä tiheydellä. Joitakin kyselyitä tehdään jopa minuutin välein, kyselyn tyypin mukaan. Tätä voisi verrata palvelimen näkökulmasta siihen, että on joku web-sivun selailija, joka painaa selaimen päivitys-nappia minuutin välein. Lähes jokaiseen kyselyyn vastaaminen vaatii myös, että tietokannasta haetaan tietoa, ja tämä aiheuttaa tietenkin viivettä kyselyyn vastaamiseen. Jokaiseen kyselyyn pitäisi pystyä vastaamaan muutamassa sekunnissa, ettei käyttäjän käyttökokemus kärsi. Tähän on tehty erilaisia ratkaisuja asiakassovelluksen päässä siten, että kaikki, mitä vain pystytään, tehdään taustalla häiritsemättä käyttöliittymää.

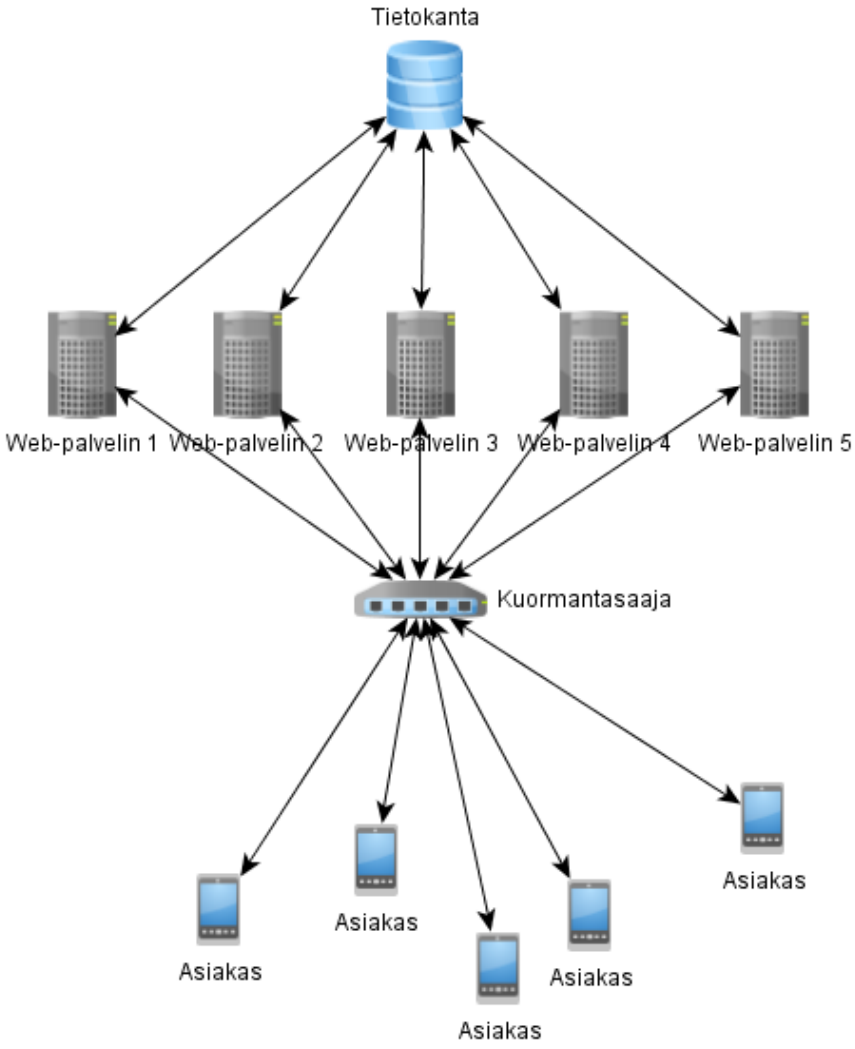
Käytäntö on osoittanut seuraavaa: jos kaikki tietokantakyselyt tehdään silloin, kun asiakassovellus niitä kyselee, viivettä vastauksiin alkaa tulla jo siinä vaiheessa, kun asiakassovelluksia on käytössä muutamia satoja. Viiveen kasvaessa tarpeeksi tapahtuu jossain vaiheessa ns. lumipalloefekti, eli kun edellisen kyselyn käsittely on kesken seuraavan kyselyn tullessa, kysely jää jonoon ja näin pikkuhiljaa jonossa olevien kyselyjen määrä vain kasvaa. Web-palvelin pystyy käsittelemään useampaa kyselyä kerralla, mutta tässäkin on rajansa.

Tietokannan nopeutta on optimoitu lisäämällä tauluihin puuttuvia indeksejä. Lisäksi olennaisimpiin kyselyihin tehdään vastauksia valmiiksi, jolloin niihin pystytään palauttamaan vastaus murto-osasekunneissa. Vaikuttaisikin siltä, että isoin pullonkaula on web-palvelimen kyky käsitellä riittävästi kyselyjä rinnakkain. (Using Missing Indexes Information to Write CREATE INDEX Statements. 2009.)

Pelkkien kyselyjen määrään antaa tuntumaa tilanne, jossa yksi mobiilipääte tekee perustilannekyselyitä minuutin välein. Otetaan vaikka 300 mobiilipäätettä, jotka kaikki tekevät kyselyitä samalla tahdilla. Pelkästään minuutin aikana tulee palvelimelle näitä tilannekyselyitä n. 300 kpl. Tämän lisäksi voi tulla muita kyselyitä, kuten henkilökunnan työvuorojen päivityksiä, leimauksia, viestien hakuja jne. 300 mobiilipäätettä riittää noin 60 päiväkodille. Näin ollen web-palvelinta pitää pystyä jollakin tapaa skaalaamaan tehokkaammaksi siten, että se pystyy palvelemaan useampaa tuhatta mobiilipäätettä. Tähän on olemassa erilaisia vaihtoehtoja, kuten useamman web-palvelimen ottaminen käyttöön ja kuormituksen jakaminen niiden kesken tai jonkin palveluntarjoajan pilvipalvelun käyttäminen. Näitä vaihtoehtoja käsitellään tarkemmin myöhemmin.

4 SKAALAUTUVA WEB-PALVELU

Web-palvelun, niin kuin minkä tahansa muunkin palvelun skaalaus suuremmille käyttäjämäärille perustuu seuraaviin asioihin. Otetaan käyttöön useampi palvelin, joiden kesken kuormitus jaetaan jollakin kuormantasaustekniikalla. Tällaista usealla palvelimella toteutettua palvelua kutsutaan myös palvelinklusteriksi (Klusteri (tietotekniikka). 2013). Klusterissa yksittäistä jäsentä, tässä tapauksessa palvelinta, kutsutaan noodiksi. Palvelimet voivat olla joko virtuaalisia tai fyysisiä palvelimia. Kuvassa 2 näkyy yleinen arkkitehtuurikuvaus web-palvelusta, jossa on useampi web-palvelin ja yksi yhteinen tietokantapalvelin.



KUVA 2. Skaalautuva web-palvelu kuormantasauksella

4.1 Virtuaalipalvelin

Virtuaalipalvelimet ovat toisen käyttöjärjestelmän sisällä ajettavia käyttöjärjestelmiä. Virtuaalipalvelin on kuin mikä tahansa fyysinen palvelin. Sillä on käytössään kaikki samat resurssit kuin fyysisellä palvelimellakin. Resurssit vain ovat oikeasti jaettu virtualisointialustan resursseista, eli samaa fyysistä prosessoria, muistia ja kovalevyä käyttääkin useampi käyttöjärjestelmä. Virtualisoinnissa on se etu, että fyysisten palvelimien resurssit saadaan tehokkaammin käyttöön. Jos jollekin virtuaalipalvelimelle jaettu resurssi ei ole sillä itsellään käytössä, virtualisointialusta antaa sen jollekin toiselle sitä enemmän tarvitsevalle käyttöön. Myös virtuaalipalvelimen resurssien alkaessa loppua virtuaalialusta pystyy antamaan sille niitä lisää. Virtuaalipalvelimen skaalautuvuus on siis ihan omaa luokkaansa verrattuna fyysiseen palvelimeen. Lisäksi toimintavarmuuden pitäisi olla parempi, koska yleensä virtualisointialustat on toteutettu useammalla fyysisellä palvelimella. Jos joku näistä palvelimista rikkoontuu, siinä ajettavat virtuaalipalvelimet siirretään automaattisesta ehjälle palvelimelle. (What is virtualization?)

4.2 Kuormantasaus

Kuormantasaus on tekniikka, jolla palveluun kohdistuvat verkkokyselyt jaetaan tasaisesti useamman palveluresurssin kesken. Yksi palveluresurssi voi olla esimerkiksi web-palvelin.

Yksi kuormantasaustyyppi on round-robin DNS (Dynamic Name Service). Tämä on yksikertaisin kuormantasaustekniikka. Tekniikka perustuu siihen, että kun asiakas kysyy mistä osoitteesta www.palvelu.fi löytyy, nimipalvelu tietää useamman IP-osoitteen, mistä sama palvelu löytyy, ja palauttaa eri asiakkaille eri osoitteen eli kierrättää osoitteita. Näin siis teoriassa jokainen kysely jakautuu tasaisesti eri palvelimien kesken. Round-robin-tekniikka ei vaadi välttämättä erillistä fyysistä laitetta. Tässä kuorman tasauksessa on se ongelma, että tekniikassa ei ole minkäänlaista älyä. Jos esimerkiksi joku palvelun palvelin kaatuu, ei tekniikka osaa poistaa sitä palvelusta, vaan kyselyjä saatetaan ohjata kaatuneelle palvelimelle. Lisäksi moni asiakassovellus tallentaa itselleen välimuistiin

kerran saamansa IP-osoitteen ja tämä aiheuttaa sen, että tästä eteenpäin asiakas kuormittaa vain yhtä palvelinta. (Tarreau 2003.)

Älykkäämpi kuormantasaustekniikka on toteutettu joko erillisellä verkkolaitteella tai sovelluspohjaisesti käyttämällä palvelinta, jossa on kuormantasausohjelmisto. Tällaisissa tekniikoissa on yleensä heartbeat-ominaisuus eli kuormantasaaja kyselee palvelimilta niiden tilasta. Jos joku palvelin ei vastaa määritetyn ajan kuluessa, palvelin merkitään vialliseksi, eikä sille enää välitetä kyselyitä. Tällainen tekniikka jakaa kuorman tasaisemmin kuin round-robin-DNS-tekniikka, koska asiakkaille näkyy vain yksi yhteysosoite. (Tarreau 2003.)

Microsoftin Windows Server -käyttöjärjestelmissä on jo pitkään ollut mahdollista toteuttaa kuormantasaus ohjelmallisesti. Esimerkiksi Windows Server 2008 -versiossa kyseinen ominaisuus on nimeltään Network Load Balancing tai lyhyemmin NLB. Ominaisuus pitää kuitenkin ottaa erikseen käyttöön ja konfiguroida.

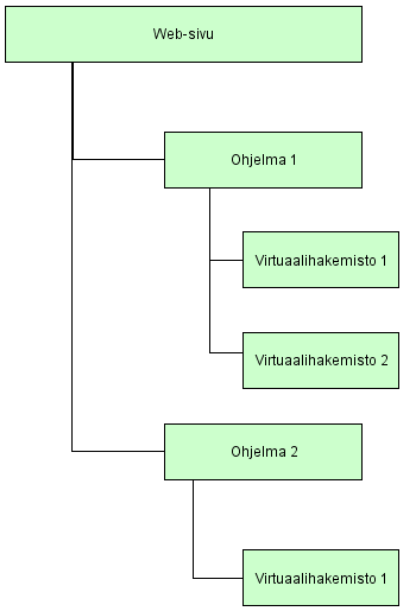
NLB-palvelun käyttöönotto tapahtuu seuraavasti. Asennetaan tarvittava määrä palvelimia identtisellä konfiguraatiolla. Yhteen palvelimeen tai nodeen asennetaan NLB-palvelu, minkä jälkeen tämä node toimii kuormantasaajana. Jokaiselle nodelle määritetään yksilöllinen IP-osoite ja yhteinen ns. palveluosoite. Palveluosoite on osoite, jonka kautta asiakkaat kommunikoivat palvelun kanssa. Jokainen node liitetään NLB-palveluun. (Bomb 2008.)

NLB-palvelulle voidaan määritellä erilaisia affiniteettejä eli määrityksiä, millä periaatteella kuormantasaus tehdään. Voidaan esimerkiksi määrittää, että 60 prosenttia kuormasta menee nodelle A ja loput nodelle B tai että asiakkaan pyynnöt menevät samalla nodelle mihin asiakkaan ensimmäinenkin pyyntö. Muita affiniteettimäärityksiä ovat Single, VPN ja IPSec, Ei affiniteettiä ja Class C. Näitä ei ole tarpeen käsitellä tässä tarkemmin. (Bomb 2008.)

4.3 Web-palvelu palvelinklusterissa

Koska Daisy-järjestelmä käyttää Microsoft-teknologiaa, tässä työssä käsitellään web-palvelun toteutusta ainoastaan Microsoftin teknologialla toteutettuna. Microsoftin web-palvelu on nimeltään Internet Information Services eli lyhyemmin IIS. IIS:ssä web-sivusto koostuu yhdestä tai useammasta web-ohjelmasta. Oh-

jelma koostuu yhdestä tai useammasta virtuaalihakemistosta, ja virtuaalihakemisto viittaa fyysiseen sijaintiin kovalevyllä. Kuvassa 3 on esitetty hierarkia yleisesti. (Templin 2007.)



KUVA 3. Web-sivuston eri elementtien hierarkia

Web-sivujen ohjelmia ajetaan yhdessä tai useammassa ohjelmistopoolissa (ApplicationPool). Ohjelmistopoolit ovat toisistaan eristettyjä ”hiekkalaatikoita” eli ohjelmat eri ohjelmistopooleista eivät pääse käsiksi toistensa tietoihin. Jokaiselle ohjelmistopoolille voidaan erikseen määritellä montako työläisprosessia (worker-process) sillä on käytössä. Työläisprosessi on prosessi, joka hoitaa asiakkailta tulevien pyyntöjen käsittelyt. Ohjelmistopoolia voidaan muutenkin hallita erikseen, esimerkiksi ohjelmistopooli voidaan käynnistää uudelleen, jolloin ainoastaan siinä ajettaviin ohjelmiin tulee käyttökatkos. (Templin 2007.)

Web-palvelun klusteroinnissa pitää suunnitella millä tavalla kuorma jaetaan eri nodejen välillä. Tähän on kaksi tapaa: ohjelman tyyppiin perustuva malli (application load distribution) ja jaettu malli (aggregate load distribution). Ohjelman tyyppiin perustuvassa mallissa palvelinklusterin eri nodeille on määritetty eri tehtävät. Tyypillisesti tällaisessa mallissa dynaamista sisältöä tarjottavat websivut ovat eri nodeilla ja staattista sisältöä sisältävät sivut toisilla. Jaetussa mal-

lissa kaikki nodet hoitavat kaikkia tehtäviä. (Hirschman – Baldwin – Leverette – Johnson 2007.)

Lisäksi pitää miettiä millaisiin ohjelmistopooleihin web-sivustot jaetaan. Tähänkin on käytännössä kaksi eri mallia, eristetty tai skaalautuva malli. Eristetyssä mallissa jokaiselle web-sivustolle tehdään oma ohjelmistopooli, jolloin tietoturva on parempi, koska jokainen web-sivusto ajetaan omassa prosessissaan eikä mitään resursseja jaeta muiden web-sivustojen kesken. Näin web-sivuilla ei ole mahdollista päästä toistensa tietoihin käsiksi. Tässä mallissa muistivaatimukset ovat isommat, koska jokainen pooli varaa erikseen muistia. Varsinkin tuhansia sivuja isännöitäessä muistivaatimukset saattavat tulla vastaan. Skaalautuvassa mallissa skaalautuvuus on parempi, koska muistivaatimukset eivät ole samaa luokkaa kuin eristetyssä mallissa. Tietoturva ei ole niin hyvä kuin eristetyssä mallissa, koska web-sivustot suoritetaan saman prosessin alla, jolloin sivustoilla on teoreettinen mahdollisuus päästä käsiksi toistensa tietoihin. (Hirschman ym. 2007.)

5 PILVIPALVELUT

Web-palvelun voi tehdä myös käyttämällä jonkin palveluntarjoajan valmiita pilvipalveluja. Pilvipalveluissa palveluntarjoajalla on valmis palvelinjärjestelmä, josta ostetaan resursseja, kuten erilaisia palveluja, prosessoriaikaa, levytilaa, verkkoliikennettä jne. Yleensä pilvipalveluntarjoajalla on oma pilvipalvelualusta, jossa kaikki palvelut pyörivät. Tässä palvelumallissa on kätevää se, että alkuinvestoinnit ovat yleensä huomattavasti pienemmät, kuin oman palvelinjärjestelmän rakentamisessa, koska hinnoittelu perustuu käytettyihin resursseihin ja monesti myös tarjotaan alkuun ilmainen kokeilujakso, johon sisältyy rajattu määrä resursseja ja palveluita. Lisäksi palveluntarjoaja huolehtii palvelimien ylläpidosta, joten siitäkään ei tarvitse asiakkaiden itse huolehtia.

Pilvipalveluissa sovellusta ajetaan yhdessä tai useammassa palvelininstanssissa eli virtuaalikoneessa. Yleensä hinnoittelu perustuu siihen, montako instanssia on käytössä ja kauanko mikäkin instanssi on ajossa. Suorittavien instanssien määrää, kuten muitakin tarvittavia resursseja, voidaan pienentää ja kasvattaa käyttäjämäärien kasvaessa hyvin helposti, monesti pelkällä napin painalluksella tai jopa ohjelmallisesti pilvipalvelun omaa API-rajapintaa käyttäen.

5.1 Amazon-pilvipalvelu

Yksi vaihtoehto palvelun toteuttamiseksi on käyttää Amazonin pilvipalvelualustaa eli Amazon Web Serviceä. Amazon tarjoaa useita erilaisia tuotteita, joita yhdistelemällä rakentuu erilaisia tarpeita vastaava kokonaisuus. Tuotteet on jaettu seuraaviin ryhmiin: laskenta, sisällön toimitus, tietokanta, käyttöönotto ja hallinta, sovelluspalvelut, verkko, maksut ja laskutus, levytila ja web-liikenne. Jokaisen tuoteryhmän alla on useita eri vaihtoehtoja. Liitteestä 1 löytyy Amazonin esimerkkiarkkitehtuurikuvaus web-palvelun toteuttamisesta. Toteutus on Amazonin mukaan hyvin skaalautuva. (Amazon products & services. 2013.)

Amazon-pilvipalvelualusta tarjoaa sovelluskehittäjille API-rajapinnan, jolla pyritään helpottamaan palvelun käyttöä. Rajapinnan käyttöön on tarjolla SDK:t useille suosituille kehitysympäristöille, kuten Android, iOS, Java, .NET, Node.js,

Python, PHP ja Ruby. Rajapinnan kautta pääsee hallitsemaan pilvipalvelualustan eri palveluita ohjelmallisesti.

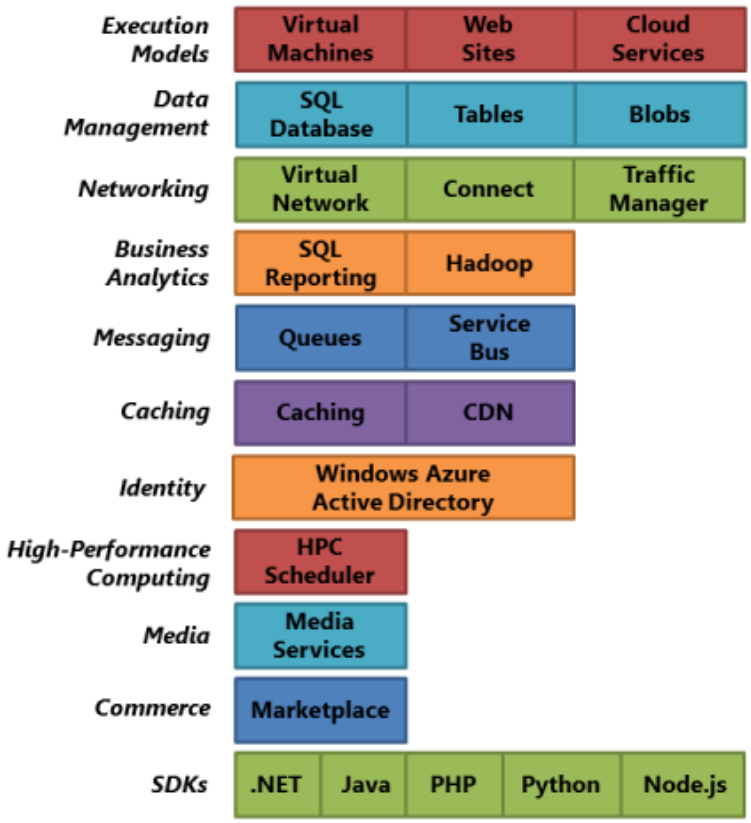
Pilvipalvelun käyttöönottoon on ainakin kaksi tapaa. Se voidaan tehdä suunnittelemalla ja kokoamalla itse palveluarkkitehtuuri eri palvelukomponenteista. Vaihtoehtoisesti voi käyttää Elastic Beanstalk -palvelua, joka hoitaa tarvittavien palveluiden käyttöönoton. Amazon tukee ohjelmistoja, jotka ovat tehty jollakin seuraavista teknologioista:

- .NET
- Java
- PHP
- Python
- Ruby.

Ohjelmistoja voidaan ajaa joko Windows- tai Linux-käyttöjärjestelmällä.

5.2 Microsoft Azure-pilvipalvelu

Microsoft tarjoaa myös oman pilvipalvelualustan nimeltään Windows Azure. Kuten Amazon myös Microsoftin palvelu jakautuu useaan eri palvelukomponenttiin. Palvelukomponenttien ryhmittely on esitetty kuvassa 4. Tarvittava palvelukokonaisuus muodostuu yhdestä tai useammasta palvelukomponentista. (Chappell 2013b.)



KUVA 4. Windows Azure -pilvipalvelun eri palvelukomponentit (Chappell 2013b)

Asiakas siis valitsee tarpeidensa mukaan kuvassa esitetyistä komponenteista tarvitsemansa palvelut. Microsoft tarjoaa kolme eri vaihtoehtoa millä mallilla sovelluksia ajetaan. Nämä on esitetty kuvan 4 ylimmässä Execution Models -ryhmässä.

Virtual Machines -malli on samanlainen kuin monella muullakin palveluntarjoajalle eli virtuaalikone, jota ajetaan Microsoftin konesalissa. Tässä mallissa palvelimen ylläpito jää asiakkaalle, ja tämä antaa samalla vapaat kädet tehdä mitä haluaa virtuaalipalvelimen sisällä. Myös palvelimen käyttöjärjestelmä on valittavista erilaisista Windows-versioista Linuxin eri versioihin. Vaihtoehtoisesti voi käyttää myös omaa virtuaalikonekuvaa. (Chappell 2013a.)

Web Sites -malli on tarkoitettu, kuten nimikin kertoo, tilanteisiin, joissa tarve on vain ajaa web-sivustoa palvelussa. Tämä malli on huomattavasti rajoitetumpi verrattuna virtuaalikonemalliin. Tässä ei esimerkiksi ole itsellä hallintaa itse alustaan ja käytettävissä on ainoastaan ne kirjastot jotka alustassa ovat valmii-

na. Web-sivustoja ajetaan IIS-web-palvelussa ja tietokantana voi käyttää joko ClearDB- tai MySql-tietokantaa. Tuettuja ohjelmointiteknologioita ovat .NET, Php ja Node.js. Käytettävissä on myös erilaisia julkaisujärjestelmiä, kuten WordPress, Joomla ja Drupal. (Chappell 2013a.)

Cloud Services -malli on kaikista joustavin. Tässä mallissa voidaan ajaa muunkin tyyppistä sovellusta kuin web-sovellusta, mutta toisin kuin virtuaalikonemallissa, itse palvelinalustan ylläpidosta huolehtii Windows Azure. Sovellusta suorittavien instanssien käyttöjärjestelmä on kuitenkin rajoitettu Windows Serveriin. Laskutus perustuu tässäkin, kuten muissakin malleissa, ajossa olevien instanssien ajoaikaan. (Chappell 2013a.)

5.3 Muut pilvipalvelut

Muitakin pilvipalveluita on olemassa, esimerkiksi Googlen App Engine. Googlen palvelussa sovelluksia ajetaan Java-virtuaalikoneen alaisuudessa, joten kaikki Java-teknologiaan perustuvat ohjelmointikielet ovat tuettuja. Muita tuettuja kieliä ovat Python ja Go, joille löytyy omat ajoympäristönsä. Google ei kuitenkaan tue ASP.NET-teknologiaa, joten Daisyä ei palvelussa voi ajaa. Palvelussa ei ole tarjolla samanlaista virtuaalikonepalvelua kuin Azuressa ja Amazonissa eli palvelussa ei voi ajaa omiin tarpeisiin räätälöityä virtuaalipalvelinta. (What Is Google App Engine? 2013.)

Pilvipalvelun kaltaisia konesalipalveluja tarjoavia yrityksiä on useitakin, mutta yleensä nämä ovat sellaisia, että ne tarjoavat ainoastaan palvelintilaa, jolloin kaikesta muusta asiakas vastaa itse. Tarjoajilla ei ole samalla tavalla valmiita kokonaisratkaisuja kuormatasaukseen ja palvelinklusterin tekemiseen kuin Amazonilla, Microsoftilla tai Googlella.

6 DAISY-JÄRJESTELMÄN KOEKÄYTTÖ PILVIPALVELUSSA

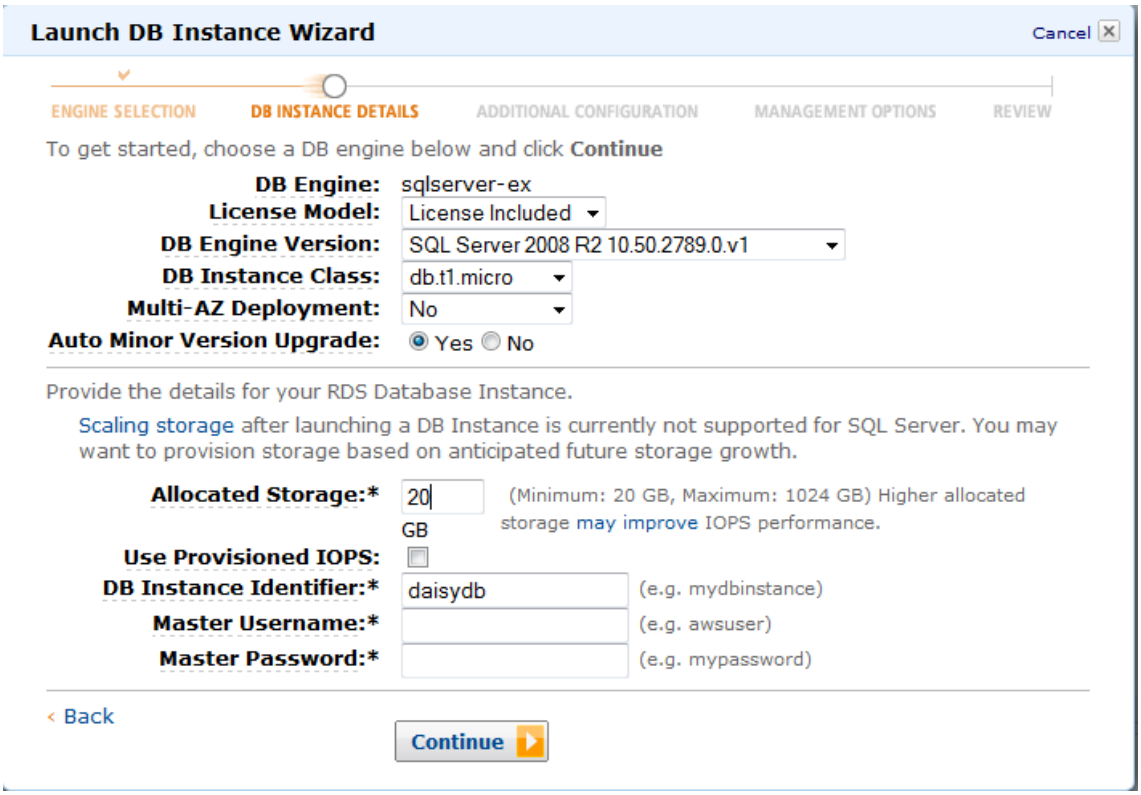
Skaalautuvuuden testaaminen pilvipalvelussa vaatii vähiten alkuinvestointeja verrattuna oman palvelinfarmin pystyttämiseen. Joustavuutta tällä ratkaisulla saadaan vähintäänkin yhtä paljon. Testialustaksi valittiin Amazonin Web Services-pilvipalvelualusta. Amazonkin tarjoaa ilmaisen kokeilujakson (Free Usage Tier), johon kuuluu tietty määrä palvelininstansseja, prosessointiaikaa, tietoliikennettä ja oikeus käyttää tärkeimpiä palveluja, kuten Amazon RDS:ää eli tietokantapalvelua.

Daisyn asennus Amazoniin aloitettiin rekisteröitymällä palveluun. Rekisteröinnissä saatuja tunnuksia suositellaan käytettäväksi ainoastaan palvelun hallinnan ylätason tehtäviin, kuten uusien palveluiden ostamiseen, IAM-käyttäjätunnusten (AWS Identity & Access Management) hallinnointiin jne. Kehitystyötä varten Amazon suosittelee tehtäväksi erilliset pienemmillä käyttöoikeuksilla varustetut tunnukset. Rekisteröitymisen jälkeen luotiin siis IAM-tunnukset, joille annettiin tehokäyttäjän oikeudet. IAM-tunnuksien luonnin yhteydessä kannattaa ottaa talteen Access Key, koska sitä tarvitaan myöhemmin kun palveluun julkaistaan ohjelmistoa Visual Studion kautta.

Asennuksissa noudatettiin aika pitkälle Amazonin Developer Guidesta löytyvää Getting started .NET -ohjetta (Get Started – Elastic beanstalk 2010). Ohjeessa käytetään AWS Toolkitiä, jossa tulee mukana AWS SDK ja työkalut Visual Studioon. Seuraavaksi asennettiin AWS Toolkit. Asennus suoritettiin oletusasetuksilla.

Asennuksen jälkeen varattiin tarvittavat palvelut ja resurssit. Suurimman osan tästä hoitaa Elastic Beanstalk -palvelu, jonka tarkoituksena on helpottaa sovelluksien siirtoa pilvipalveluun. Beanstalk tekee sovellukselle ympäristön, joka sisältää mm. ajoympäristön (Environment), kuormantasaajan ja palomuuriasetukset. Beanstalkilla tehtyjä ympäristöjä kutsutaan Beanstalk-sovelluksiksi. Jokaiselle sovellukselle luodaan oma ympäristö, jolloin skaalautuvuus ja virhesietoisuus pysyy hyvänä. Beanstalk tekee hyvät pohjat, joita pääsee jälkikäteen muokkaamaan haluamikseen.

Kuitenkin ennen sovelluksen luomista Elastic Beanstalkilla tehtiin tietokantapalvelu valmiiksi, jotta sen voi sitten liittää sovellukseen sen luontivaiheessa. Tietokantapalvelu eli Amazon RDS -instanssi luotiin web-konsolin kautta, toisin kuin Amazonin ohjeessa sanotaan, koska jostain syystä instanssin luonti ei onnistunut Visual Studion kautta. Palvelun luonnissa käynnistyy avustava velho, jossa valitaan instanssit asetukset. Aluksi valitaan millä moottorilla kantaa ajetaan. Tähän valittiin Microsoft SQL Server Express Edition, koska Daisy käyttää Microsoftin SQL-kantaa. Express Edition on tuotantokäyttöön liian köykäinen, mutta tarkoitus on tässä vaiheessa pysytellä ilmaisen kokeilujakson tarjoamissa palveluissa ja Express Edition kuuluu niihin. Seuraavaksi valittiin SQL:n versio. Valittavana on kaksi eri versioita: SQL Server 2012 ja SQL Server 2008 R2. Koska vanhakin toimii 2008-version päällä, valittiin myös tähän sama versio. Kuvasta 5 näkyvät muut valinnat: ajettavan instanssin luokka eli se millaisessa virtuaalikoneessa tietokanta pyörii, kuinka paljon tietokannalle varataan alussa tilaa, tietokantainstanssin nimi sekä pääkäyttäjän tunnus ja salasana tietokantaan.



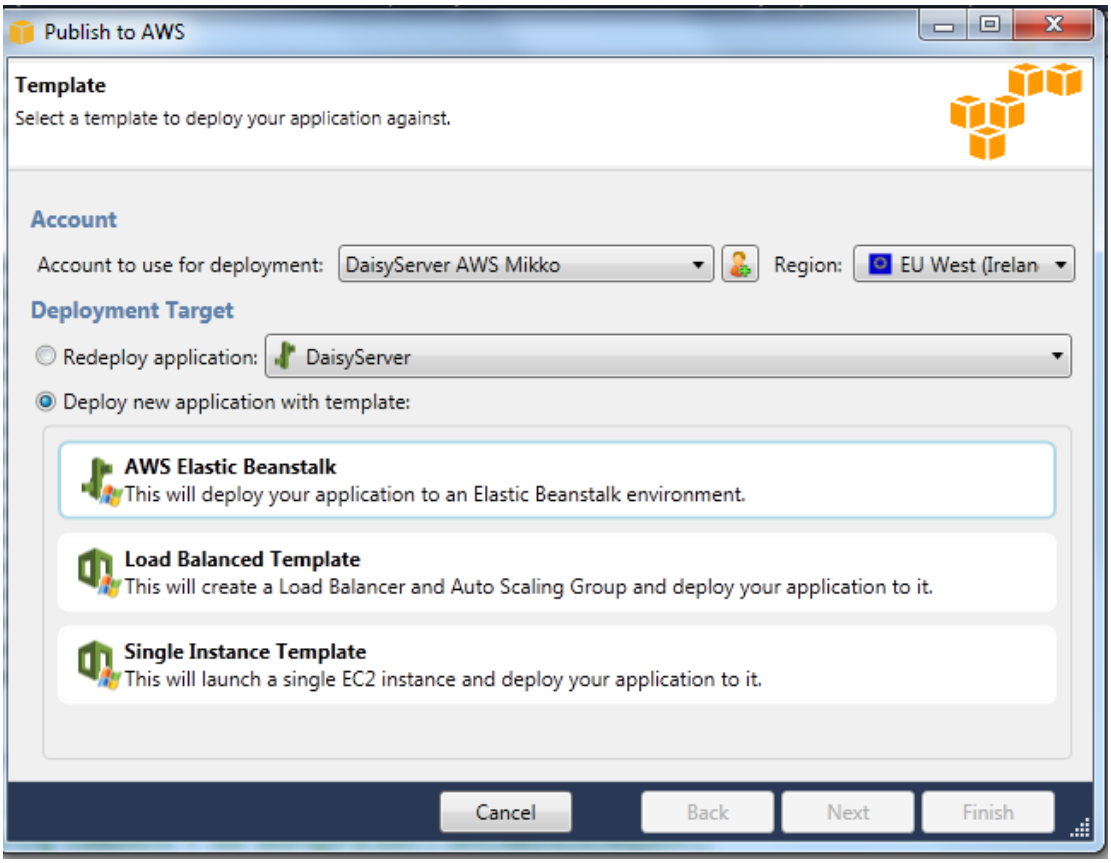
KUVA 5. RDS-instanssin ominaisuuksien valinta

Lisäksi velhossa määritellään mihin tietoturvaryhmään instanssi kuuluu eli mistä osoitteista palveluun pääsee ulkomaailmasta. Tätä varten luotiin uusi ryhmä, jossa sallitaan tietokantaan suora pääsy sovelluksen kehitystyössä käytettäviltä koneilta. Lisäksi voi muokata seuraavia asioita, jotka kuitenkin jätettiin oletusarvoihin:

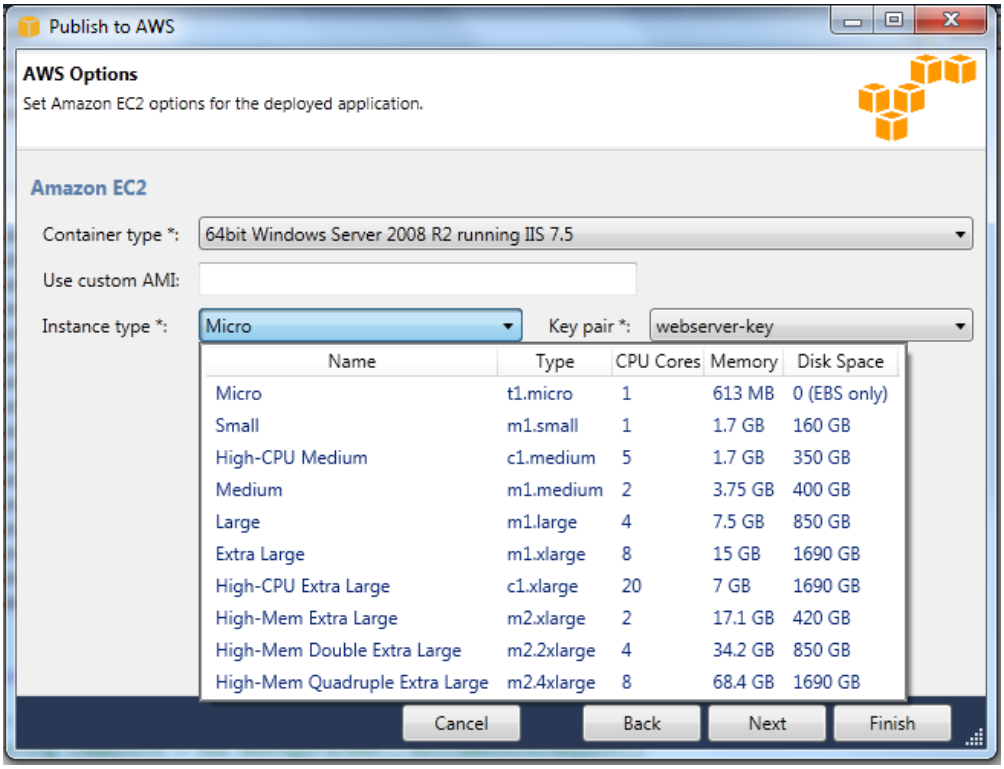
- tietokantapalvelun TCP-portti
- varmuuskopioinnin asetukset, kuten kuinka tiheään niitä otetaan ja milloin
- huoltoikkuna eli milloin tietokantaan tehdään huoltotoimenpiteitä.

Seuraavaksi olikin vuorossa itse tietokannan luonti palveluun. Tämä tehtiin käyttäen Visual Studiota ja SQL Management Studiota. Visual Studiolla luotiin skripti olemassa olevasta tietokannasta käyttäen Publish to provider -toimintoa Server Explorerista. Toiminto luo skriptin, joka tekee tietokannan. Taulujen luontiin tehtiin toinen skripti käyttäen SQL Management Studion Generate scripts -toimintoa. Kun skriptit olivat valmiina, otettiin yhteys edellä tehtyyn Amazonin RDS-instanssiin SQL Management Studiolla ja luotiin tietokanta käyttäen skriptejä.

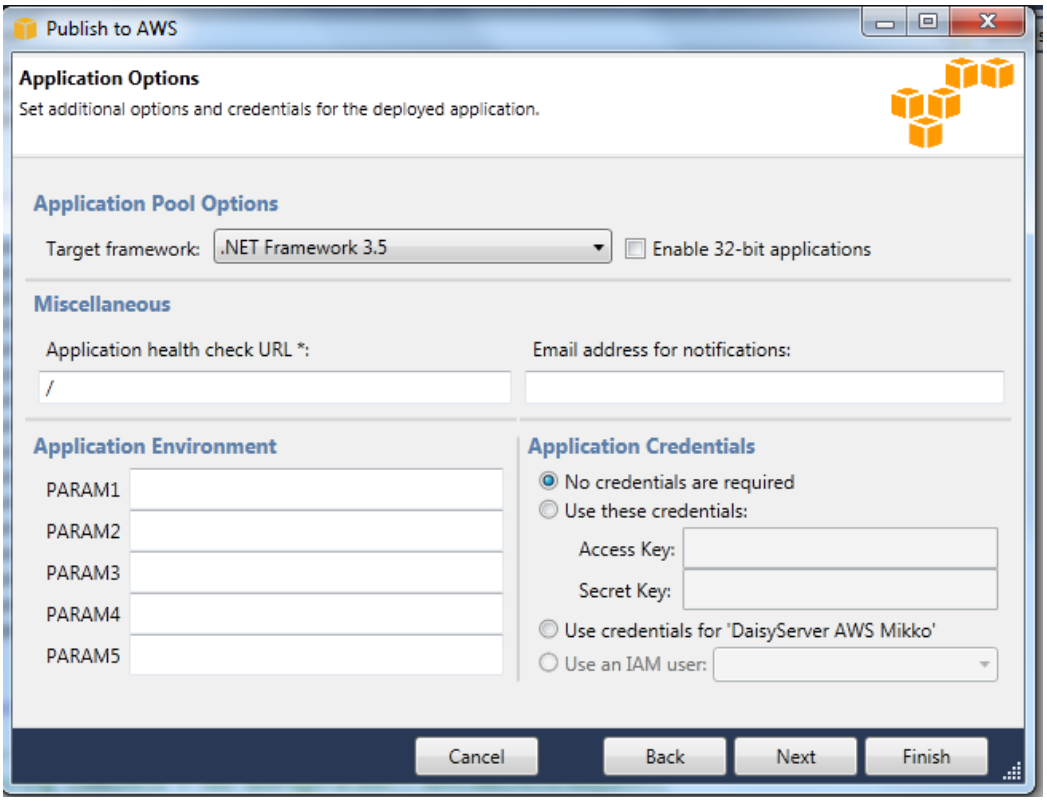
Tietokantapalvelun käyttöönoton jälkeen olikin vuorossa itse sovelluksen julkaisu pilveen. Ennen sovelluksen siirtoa palveluun sovelluksen asetustiedosto muutettiin siten, että sovellus käyttää tietokantana Amazonissa olevaa tietokantaa paikallisen tietokannan sijaan. Sovelluksen julkaisu aloitettiin samaan tapaan kuin sovelluksen julkaisu mihin tahansa web-palvelimelle. Nyt käytettiin vain Publish to AWS -toimintoa normaalin Publish…-toiminnon sijaan. Toiminto tuli AWS Toolkitin asennuksen myötä Visual Studioon. Toiminnolla luodaan Elastic Beanstalk -sovellus. Sovelluksen luonti on hyvin suoraviivainen. Sovelluksen luontivelholla määritetään aluksi millä Amazon sovelluspohjalla sovellus luodaan. Tähän valittiin Elastic Beanstalk. Seuraavaksi valittiin versio, jolla ASP.NET-sovellusta ajetaan, instanssien tyyppi eli kuinka tehokkaita käynnistettävät virtuaalikoneet ovat, ajettava käyttöjärjestelmä, käytetäänkö koneiden luontiin jotain valmista levykuvaa ja tietokantaryhmä eli RDS DB Security Group. Kuvista 6–9 näkyy tarkemmin mitä muita valintoja voidaan tehdä.



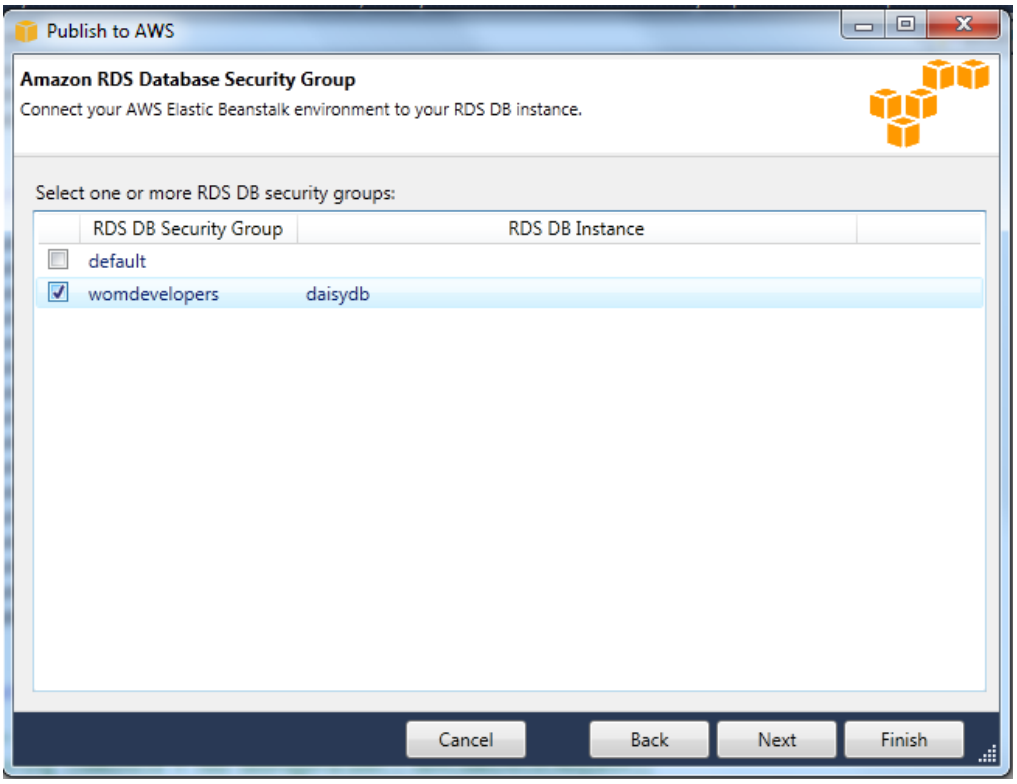
KUVA 6. Amazon-sovelluspohjan valinta



KUVA 7. EC2-Instanssien asetukset

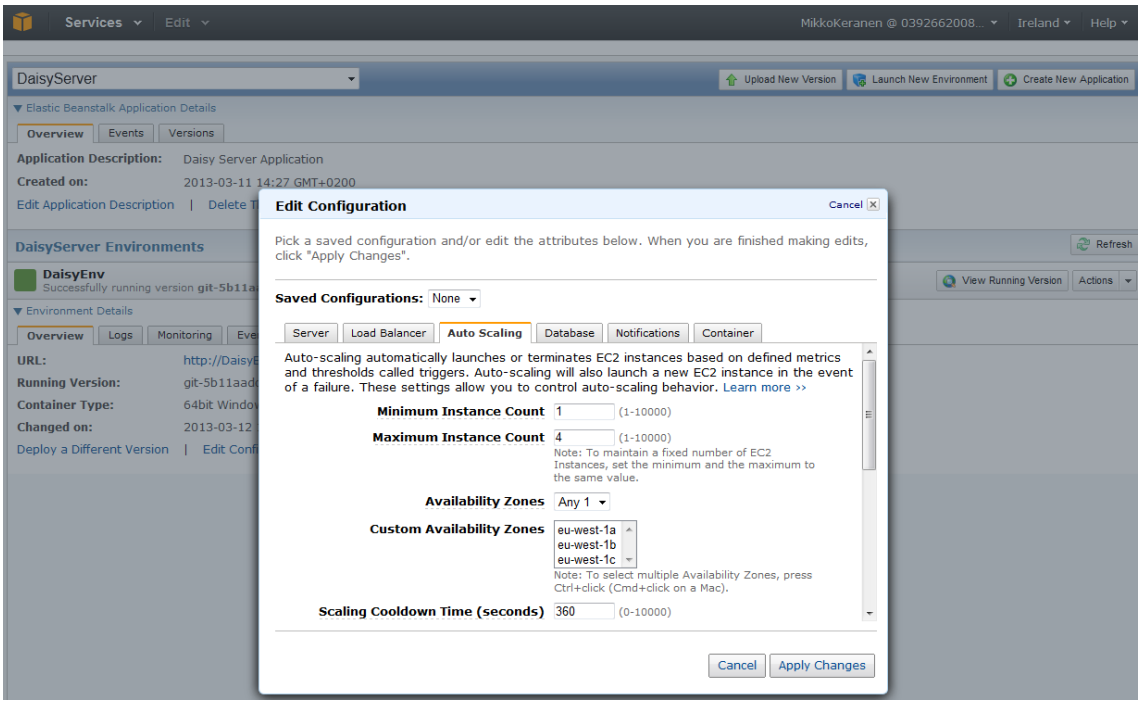


KUVA 8. Sovelluksen asetukset



KUVA 9. Tietokantaryhmän valinta

Sovelluksen julkaisun jälkeen Daisy-sovellus toimii pilvipalvelussa ja sille voidaan antaa tarpeen mukaan lisää resursseja käytettäväksi muokkaamalla ajoympäristön ominaisuuksia. Kuvassa 6 näkyy ajoympäristön muokkausikkuna, jossa voidaan säätää ympäristön skaalautuvuutta eli voidaan mm. määrittää kuinka monta instanssia pitää vähintään olla ajossa ja kuinka paljon maksimissaan käynnistetään. Instanssien lisäystä ja vähentämistä hallitaan erilaisilla liipaisimilla. Esimerkiksi jos prosessorikuorma ylittää määritetyn rajan, käynnistetään uusi instanssi. Kuvassa näkyy myös muita välilehtiä kuten Server, Load Balancer, Database, Notification ja Container, joista päästää muokkaamaan ympäristön eri ominaisuuksia. Server-välilehdeltä pystytään muokkaamaan yksittäisen instanssin ominaisuuksia, kuten kuinka paljon sillä on prosessoritehoa ja muistia käytettävissä. Load Balancer -välilehdeltä muokataan kuormatasauksen ominaisuuksia: mihin osoitteeseen ympäristö tekee HTTP-kutsuja tarkistaakseen instanssien tilan, millaisella aikavälillä tarkistuksia tehdään ja milloin instanssin tila tulkitaan virheelliseksi.



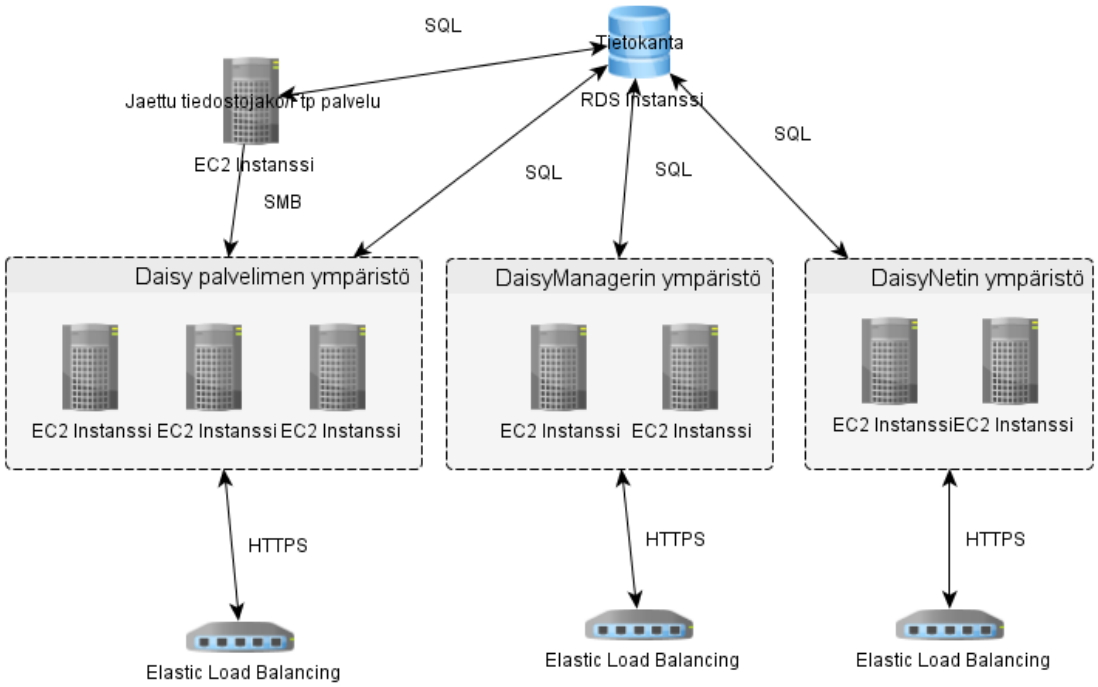
KUVA 10. Ajoympäristön konfiguraation muokkaus

Sovelluksen julkaisun jälkeen oli vielä vuorossa instanssin tarkempi asetuksien muokkaus, kuten FTP-palvelun asennus ja sovelluksen tarvitsemien hakemisto-

rakenteiden tekeminen. Kun asetukset oli saatu määritettyä, tehtiin instanssista AMI-image eli levykuva. Tämä levykuva asetettiin ajoympäristön levykuvaksi, joka tekee sen, että kun palvelu luo uusia instansseja, tehdään ne käyttäen muokattua levykuvaa, jolloin siihen tehdyt asetukset ovat aina myös uusissa instansseissa. Muuten kaikki tehdyt asetukset häviävät, kun instanssi vaihdetaan toiseen automaattiskaalauksen toimesta.

Levykuvan luomisen jälkeen vuorossa oli testidatan ajaminen tietokantaan ja sovelluksen testaus. Testidatan lataaminen kantaan jo hyydytti käytössä olleen mikrokokoisen instanssin täysin. Testidata ladattiin käyttäen Effica-rajapintaa, ja tämäkin näytti olevan jo raskas prosessi mikroluokan instanssille. Tämä ei sinänsä yllätä, koska mikroluokan instanssilla on ainoastaan 613 megatavua muistia ja niiden prosessoriteho on hyvin rajoitettu. Ainoastaan lyhyisiin piikkeihin saadaan enemmän tehoa. Tuotantokäyttöön kannattaakin määritellä järeämpi instanssityyppi.

Tuotantokäyttöä ajatellen tarvitaan kuvan 11 kaltainen ympäristö. Kuvassa on FTP-palvelulle ja yhteisille tiedostoille varattu oma instanssi. Tässä instanssissa generoidaan välimuistia ja otetaan vastaan taustajärjestelmistä tiedostoja. Daisy-palvelin lukee täältä välimuistitiedostot. Lisäksi kaikille sovelluksille on olemassa yhteinen tietokantainstanssi. Jokaiselle sovellukselle on oma ajoympäristö, joihin kuormantasaus ohjaa pyynnöt. Ajoympäristöissä on tarvittava määrä instansseja käynnissä. Sen mukaan, mikä on sen hetkinen kuormitus asiakkailta, pilvipalvelualusta huolehtii, että käynnissä on tarpeeksi instansseja. Jos jossakin instanssissa huomataan vikaa, kuormantasauspalvelu ottaa sen pois käytöstä.



KUVA 11. Arkkitehtuurikuva Daisystä Amazon-pilvipalvelussa

7 POHDINTA

7.1 Työn tulokset

Työssä oli tarkoitus selvittää, millaisilla ratkaisuilla saataisiin Daisy-palvelimesta niin skaalautuva, että se pystyy palvelemaan suuriakin määriä Daisy-käyttäjiä.

Käytäntö on osoittanut, että yksi neljän prosessoriytimen palvelin pystyy palvelemaan yli 500:aa yhtäaikaista asiakassovellusta, kun tietokanta on eriytetty omalle palvelimelleen. Yhteen palvelimeen ei voi lisätä loputtomasti muistia ja prosessoriytimiä ja niitä ei kuitenkaan pystytä 100-prosenttisesti hyödyntämään, koska esimerkiksi prosessin siirtäminen toiselle prosessorille vie sekin tehoja.

Tehokkain ja skaalautuvin ratkaisu on kuorman hajauttaminen useammalle palvelimelle eli tehdä ns. palvelinfarmi. Palvelinfarmin voi rakentaa itse: huolehtimalla tarvittavien palvelimien hankinnasta joko fyysisinä tai virtuaalipalvelimina ja ottamalla vastuun kaikesta palvelimiin tehtävistä asetuksista, ylläpidosta, päivityksistä jne. Tässä ratkaisussa on se huono puoli, että alussa hankitaan joko liian vähän tai liikaa palvelimia, jolloin alkukustannukset tulevat aika isoiksi. Myös käyttöönotto on hitaampaa, koska jokainen palvelin pitää alusta lähtien asentaa. Kymmenien palvelimien ylläpito alkaa vaatia myös henkilöresursseja enemmän. Vaihtoehtona itse tehdylle on ostaa palvelintilaa pilvipalvelusta. Tällöin ei tarvitse huolehtia ylläpidosta tai hankinnoista itse, vaan kaiken hoitaa palveluntarjoaja. Laskutus perustuu käytettyihin resursseihin, jotka ovat pienellä käytöllä minimaaliset.

Tässä työssä testattiin, miten Daisyn asennus ja ajaminen Amazonin pilvipalvelussa onnistuu. Käytössä oli ainoastaan ilmaiseen kokeilujaksoon kuuluvat resurssit, mutta sovelluksen toimivuus pystyttiin niilläkin testaamaan ja tulokset olivat positiiviset. Tarvittavat määritykset pystyttiin tekemään web-instansseihin ja tietokanta istui tietokantainstanssiin. Sovelluksen julkaisu palvelussa oli helppoa ja se onnistui suoraan Visual Studio -kehitystyökalulla.

7.2 Jatkokehitysmahdollisuudet

Jatkossa kannattaisi miettiä, siirretäänkö kaikki asiakkaat pilvipalveluun. Tällaisia asiakkaita voisivat olla ainakin ne, joiden palvelimien ylläpidosta WhileOnTheMove vastaa. Näin saataisiin ylläpitoa helpotettua, kun kaikesta ei enää tarvitse huolehtia itse. Myös tehoja saa tarvittaessa palveluun lisää hetkessä. Palvelujen hinnoittelussa näkee selkeästi, että avoimeen lähdekoodiin perustuvien teknologioiden käyttö on halvempaa kuin esimerkiksi ASP.NET-teknologian. Nyt Daisya täytyy suorittaa Microsoftin käyttöjärjestelmällä ja tietokannalla, jolloin ajettavien instanssien tuntihinta on hieman kalliimpi kuin avoimeen koodiin perustuvien alustojen, kuten MySQL:n tai Linuxin hinta.

LÄHTEET

Amazon BeansTalk. 2013. Saatavissa: http://aws.amazon.com/elasticbeanstalk/. Hakupäivä 25.3.2013.

Amazon products & services. 2013. Saatavissa: http://aws.amazon.com/products/. Hakupäivä 25.3.2013.

Bomb, Adam 2008. Network Load Balancing (NLB) in Windows Server 2008. Saatavissa: http://technet.microsoft.com/en-us/video/network-load-balancing-nlb-in-windows-server-2008.aspx. Hakupäivä 12.2.2013.

Chappell, David 2013a. Windows Azure Execution Models. Saatavissa: http://www.windowsazure.com/en-us/develop/net/fundamentals/compute/. Hakupäivä 25.3.2013.

Chappell, David 2013b. Introducing Windows Azure. Saatavissa: http://www.windowsazure.com/en-us/develop/net/fundamentals/intro-to-windows-azure/. Hakupäivä 25.3.2013.

Get Started – Elastic beanstalk. 2010. Saatavissa: http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_NET.quickstart.html. Hakupäivä 25.2.2013.

Hirschman, Shaun – Baldwin, Mathew – Leverette,Tito – Johnson, Michael 2007. Mass-Hosting High-Availability Architectures. Saatavissa: http://msdn.microsoft.com/en-us/library/bb491120.aspx. Hakupäivä 5.2.2013.

Klusteri (tietotekniikka). 2013. Saatavissa: http://fi.wikipedia.org/wiki/Klusteri_%28tietotekniikka%29. Hakupäivä 12.2.2013.

Micro Instances. 2010. Saatavissa: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts_micro_instances.html. Hakupäivä 25.2.2013.

Tarreau, Willy 2003. Making applications scalable with Load Balancing. Saatavissa: http://1wt.eu/articles/2006_lb/index.html. Hakupäivä 4.5.2013.

Templin, Reagan 2007. Understanding Sites, Applications, and Virtual Directories on IIS 7. Saatavissa: http://www.iis.net/learn/get-started/planning-your-iis-architecture/understanding-sites-applications-and-virtual-directories-on-iis. Hakupäivä 12.2.2013.

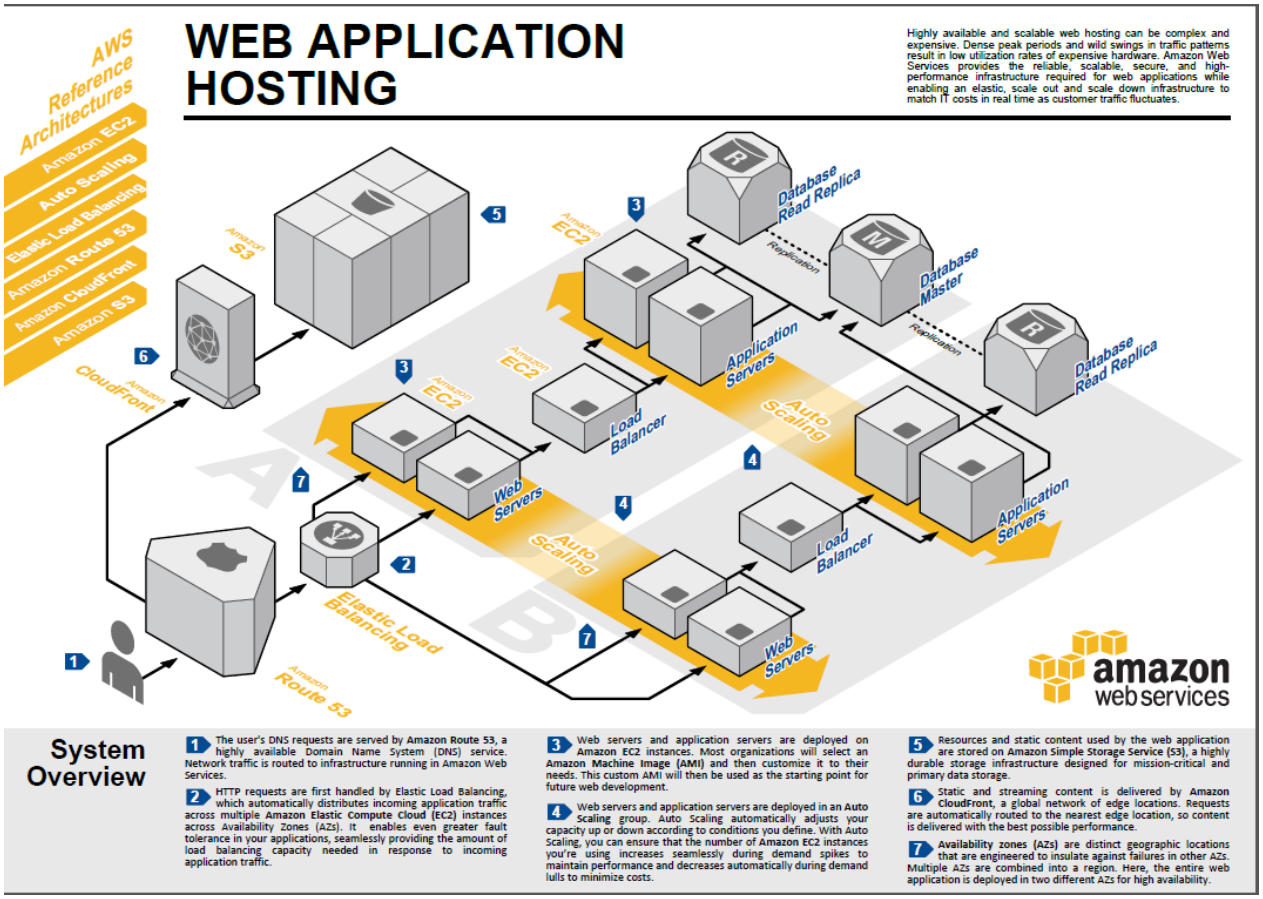
Using Missing Index Information to Write CREATE INDEX Statements. 2009. Saatavissa: http://msdn.microsoft.com/en-us/library/ms345405%28v=sql.105%29.aspx. Hakupäivä 12.2.2013.

What Is Google App Engine? 2013. Saatavissa: https://developers.google.com/appengine/docs/whatisgoogleappengine. Hakupäivä 25.4.2013.

What is virtualization?. Saatavissa: http://www.dell.com/Learn/fi/fi/fidhs1/virtualization-what-is-it?c=fi&l=fi&s=dhs. Hakupäivä 4.5.2013.

Web application hosting. Saatavissa: http://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_web_01.pdf. Hakupäivä 25.2.2013.

WEB-PALVELU AMAZON-PILVIPALVELUSSA



(Web application hosting)