Tuomas Karjalainen

ETL-työkalun pilvimigraatio Microsoft Azure SQL -ympäristöön

Insinööri (AMK)

Tieto- ja viestintätekniikka

Kevät 2023

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Karjalainen Tuomas

Työn nimi: ETL-työkalun pilvimigraatio Microsoft Azure SQL -ympäristöön

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: Microsoft Azure, ETL, pilvimigraatio, pilvipalvelut, SQL, tietovarastointi

Opinnäytetyön tavoitteena oli tutkia ETL-prosessissa eri lähdedatojen täsmäytykseen ja vertailuun käytettävän työkalun toimivuutta Microsoft Azuren pilvialustalla. Kyseinen työkalu on jo tuotantokäytössä asiakkaan on-premises -ympäristössä, mutta tavoitteena oli selvittää tarvittavat vaiheet, jotta sovellus saataisiin toimimaan tarvittaessa myös pilvialustalla esimerkiksi asiakkaan infrastruktuuristrategian muuttuessa.

Työkalu sisältää myös frontend-puolen eli web-käyttöliittymän, josta työkalua pääasiassa käytetään, mutta viitekehyksen rajaamiseksi tässä työssä käsiteltiin vain sovelluksen backend-puolta, joka muun muassa sisältää eri lähteistä tulevien datasettien täsmäytys- ja vertailulogiikan. Työkalu ei siis pelkästään ole vain ETL-työkalu, vaan se toimii myös data-analyysin työkaluna. Datan laatu, tarkkuus ja yhdenmukaisuus ovat tärkeää viranomaisraportoinnin kannalta, mitä esimerkiksi pankki- ja finanssialalla toimivat yritykset ovat velvollisia tuottamaan.

Työssä kartoitettiin mahdolliset yhteensopivuusongelmat sekä potentiaalien refaktoroinnin tarve ja sen laajuus. Työn teoreettinen viitekehys muodostui ETL-prosessista, pilvipalveluista, niiden vertailusta ja käyttöönotosta. Lisäksi teoriaosuuksiin on tuotu nostoja pankki- ja finanssialan näkökulmista. Opinnäytetyön loppupuolella käytiin tässä käyttötapauksessa suoritetut prosessin vaiheet läpi sekä niiden tulokset. Viimeisenä käsiteltiin läpi vielä työstä saatua palautetta sekä omaa pohdintaa.

Abstract

Author: Karjalainen Tuomas

Title of the Publication: Migrating ETL Tool to Microsoft Azure SQL Cloud Platform

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: Microsoft Azure, SQL, ETL, Cloud computing, Migration, Data warehousing

In this bachelor's thesis, the steps taken to setup and configure reconciliation application in Microsoft Azure are defined. The application is usually part of the ETL process, and it is a solution for comparing differences in two datasets. Moreover, the application is also a data analytical tool for finding mismatching patterns. The identification of source data is critical to ensure data quality and financial regulatory reporting.

The application was originally designed to be used in on-premises environments, but currently, it is topical to investigate its functionality in a cloud environment, which in this case is Azure. The application consists mainly of SQL-based reconciliation logic (backend) and Python Django-based web-UI (frontend). To limit the scope of this thesis, only backend compatibility in the cloud platform is studied.

In addition, the objective of this thesis is also to detect potential compatibility issues, potential needs for database refactoring, and choose the appropriate cloud service for the application. Finally, the project provides documentation for internal use, which can be assistance in future customer projects.

The theoretical section discusses the reasons for choosing Azure as well as the comparison of the cloud services offered, and the selection of the service used in this project. Furthermore, the principles of the ETL process and the steps of cloud migration are discussed in the theoretical part. Perspectives from banking and the financial field have also been included in the theoretical part. The thesis concludes with a summary of the findings and an evaluation of the results.

Sisällys

Termistö

Backend	Paikallisella palvelimella tai pilvipalvelussa suoritettava ohjelmakoodi, joka käsittelee tietokantakyselyitä, logiikkaa ja hallitsee tietojen siirtämistä käyttöliittymän ja tietokannan välillä eli tarjoaa käyttöliittymän takana tapahtuvat palvelut.
ETL	(*Extract – Transform – Load)* Tietojen jalostusketju operatiivista järjestelmistä tai muista tietolähteistä tietovarastoon.
Microsoft Azure	Microsoftin kehittämä pilvipalvelualusta, joka tarjoaa myös erilaisia valmiita komponentteja esimerkiksi koneoppimiseen sekä suurten datamassojen (*engl*. *Big Data*) analysointiin.
Migraatio	Tietotekniikassa prosessi, jossa dataa siirretään järjestelmästä toiseen tai kokonainen tietojärjestelmä siirretään toiseen ympäristöön.
On-premises	Paikallisesti toimiva ratkaisu, jonka infrastruktuuri on rakennettu esimerkiksi organisaation yksityisen datakeskuksen päälle.
PaaS	(*Platform as a Service*) Yksi yleisimmistä pilvipalvelumalleista, jossa palveluna tarjotaan alustaa, missä pyöritetään sovelluksia, tietokantaa tai käännetään koodia.
Python	Korkean tason ohjelmointikieli, joka osaa proseduraalisen ohjelmointityylin lisäksi käyttää olioita ja sisältää kehittyneet kirjastot useilla eri sovellusalueilla.
Refaktorointi	Tietokoneohjelman lähdekoodin muuttaminen siten, että sen toiminnallisuus säilyy rakenteen muuttuessa. Taustalla yleensä on laadun parantaminen, optimointi tai laajennus tapauksiin, joita ei alun perin ole otettu huomioon.
T-SQL	Structured Query Language eli SQL on pitkälle standardoitu relaatiotietokantojen kysely- ja määrittelykieli, jonka rakenteet muistuttavat etäisesti englannin kieltä. ”T” tulee sanasta Transact, joka viittaa Microsoftin ja Sybasen omaan SQL-kielen laajennukseen.

1 Johdanto

Tämän insinöörityön toimeksiantajana on ALM Partners Oy. Se on suomalainen vuonna 2011 perustettu finanssialalla toimiva yritys, jonka toimipisteet sijaitsevat Helsingissä, Tukholmassa ja Jyväskylässä. Sen päätehtäviin kuuluvat viranomaisraportointi, riskien- ja pääomien hallinta sekä finanssialan asiantuntijapalvelut. Näiden lisäksi ALM Partnersin tehtäviin kuuluvat myös teknillinen osaaminen, kuten yksilöllisesti suunniteltujen palveluiden ja innovatiivisten teknisten ratkaisujen toteuttaminen rahoitusalan asiakkaille, kuten useimmille suomalaisille pankeille. [1.]

Teknilliseen alueeseen sisältyy muun muassa data-analytiikka ja raportoinnin työkalut, tietokantaratkaisut, datan laadunhallinta sekä järjestelmien muutos- ja migraatiohankkeet [1]. Tämä insinöörityö keskittyy teknilliseen osa-alueeseen: ETL-prosessiin ja tietokantojen migraatioprosessiin, missä yleensä ETL-prosessin yhteydessä toimiva sovellus siirretään pilviympäristöön. Työn tarkoituksena on myös tuottaa dokumentaatio ALM Partnersin sisäiseen käyttöön, kuinka prosessi toteutetaan ja minkälaisia havaintoja sen aikana saatiin. Dokumentaatiossa käsitellään prosessin toteuttamiseen tarvittavat vaiheet sekä mahdolliset yhteensopivuusongelmat ja refaktoroinnin tarpeet.

Monet ALM Partnersin asiakkaat käyttävät Microsoft Azurea pilvipalveluiden tarjoajanaan. Myös osa ALM Partnersin tuotteista on jo toiminnassa Azuren pilviympäristössä. Pilvimigraatio on aikaisempien vuosien lisäksi yhä edelleen ajankohtainen aihe monilla muillakin yrityksillä ja toimialoilla [2]. Pilviympäristöön siirtymisellä on yleensä tarkoituksena hakea sen tunnetuimpia etuja, kuten palveluiden nopeampaa käyttöönottoa, helposti ja nopeasti skaalautuvaa alustaa sekä kustannussäästöjä luopumalla omista fyysisistä laitteista [3].

Työn tavoitteena on siirtää ALM Partnersin kehittämä ja jo pidempään tuotanto- ja asiakaskäytössä ollut sovellus toimimaan myös Azuressa. Kaksi edellisessä kappaleessa mainittua syytä jo riittävät Microsoft Azuren valintaan tämän työn pilviympäristöksi, eikä muita pilvipalveluiden tarjoajia tulla tässä työssä sen tarkemmin käsittelemään.

Tällä hetkellä sovellus pyörii perinteisemmällä ns. on-premises -ympäristössä, johon se on aikanaan myös suunniteltu. Kyseessä on lähdedatojen täsmäytystyökalu, joka on osana ETL-prosessia ja jonka tarkoituksena on selvittää eri lähdeaineistojen eroavaisuuksia. Työkalun avulla täsmennetään kahdesta eri lähteestä tulevat tiedot eli toisin sanoen nämä kaksi datasettiä yhdenmukais-

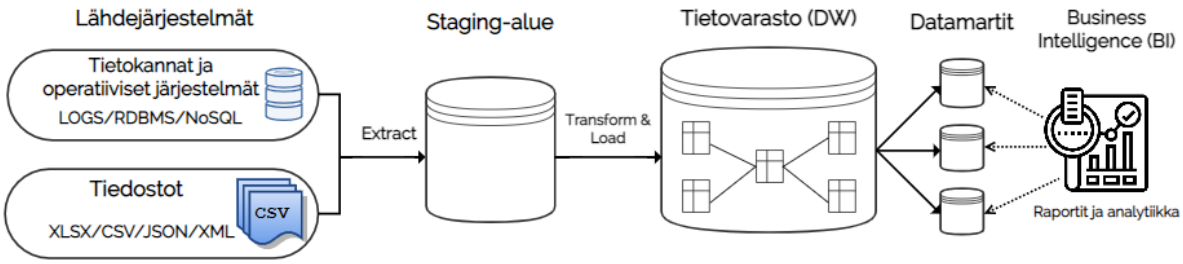
tetaan. Työkalu ei siis pelkästään ole täsmäytystyökalu, vaan se toimii myös data-analyysin tukena. Sen tavoitteena on parantaa viranomaisraportoinnin laatua takaamalla lähdetietojen oikeellisuus ja yhdenmukaisuus. Raportoinnin tarkkuus ja ajantasaisuus ovat kriittisiä, sillä esimerkiksi finanssialalla toimivat yritykset ovat usein tiukasti säänneltyjä, ja niiden on noudatettava tiukkoja raportointivaatimuksia [4].

Työkalua käytetään web-käyttöliittymän kautta, jonka taustalla (backend) pyörii pääasiassa SQL-kielellä toimiva (lähdedatojen) täsmäytyslogiikka. Riippuen siitä, kuinka suoraan nykyisen täsmäytyslogiikan SQL-koodin saa sovitettua Azuren ympäristöön, työn tutkimuksen ja kehityksen viitekehys keskittyisi pääasiassa pilvikäytön prosessin suunnittelemiseen ja implementointiin Azure SQL:n pilviympäristöön. Työ mahdollisesti tulee sisältämään myös jonkin verran backend-koodin refaktorointia, sillä hypoteesina on, että nykyisen täsmäytyslogiikan backend sitä vaatii. Tämä johtuu esimerkiksi useista eri proseduureista ja muista tietokantaobjekteista, joita täsmäytyslogiikka sisältää. Sen lisäksi tietyt tietokantaobjekteissa käytetyt skeemat, eivätkä myöskään tietokantojen väliset kyselyt, toimi samalla tavalla Azuressa. Lisäksi datan lukeminen ja lataaminen ETL-putkessa vaatii muutoksia suhteessa tämänhetkiseen on-premises -ratkaisuun. Ominaisuuseroja on-premise SQL Serverin ja Azure SQL:n välillä siis on ja nämä tulisi selvittää. Lopputuloksena saadaan käyttöönottoprosessin suunnitelma ja tarvittavat vaiheet selville. Lisäksi tiedetään, minkälainen prosessi on, mitä se vaatii ja saadaanko pilvipalveluiden oletettuja etuja hyödynnettyä tässä käyttötapauksessa. Näihin kysymyksiin vastaamalla saadaan kyseiseen työkaluun liittyviä tulevia asiakasprojekteja helpommiksi ja suoraviivaisemmiksi, sillä ennakkotietoja on jo tällöin olemassa.

Työkalua voidaan käyttää esimerkiksi tietovarastossa olevien tietojen ja kirjanpidosta tulevien tietojen vertailuun tai kahdesta eri lähteestä tulevien tietojen täsmäytykseen. Se kuinka sovellus käsittelee eroavaisuuksia aineistojen välillä, määritellään sovellustietokannan tauluihin. Kun käyttäjä kirjautuu web-käyttöliittymään, hänelle aukeaa ensimmäisenä kaikkien suorituksien tulokset, joita hän voi eri tavoin suodattaa. Käyttäjä näkee esimerkiksi, millä lähdetiedoilla täsmäytyksiä on ajettu ja kuinka monta ratkaisematonta eroavaisuutta niissä havaittiin. Käyttäjällä on mahdollisuus tutkia eroavaisuuksia ja ratkaista ne valitsemalla oikea arvo (Lähde A tai Lähde B) tai valita arvo manuaalisesti, minkä jälkeen kommenttikenttään lisätään kommentti. Kun kaikki eroavaisuudet ovat ratkaistu, suorituksen voi merkitä hyväksytyksi. Myös uuden täsmäytyksen suorittaminen voidaan käynnistää käyttöliittymän kautta, jossa käyttäjä valitsee, minkä vertailun hän haluaa suorittaa ja mitä datapäiviä käytetään. Sovellus ei kuitenkaan vaadi frontend-puolta toimiakseen.

2 ETL-prosessi

ETL-prosessi on yksi isompi vaihe dataputkea (engl. *data pipeline*), mikä tarkoittaa datan matkaa raakadatasta esikäsittelyyn, varastointiin ja edelleen analysoitavaksi sekä hyödynnettäväksi (Kuva 1) [5]. Dataputkeen kuuluu joukko erilaisia toimintoja ja menetelmiä, joiden tarkoituksen on tarjota organisoitu, johdonmukainen ja mahdollisimman automaattinen dataflow, jotta tietoja voidaan jatkossa hyödyntää [6]. Sen suunnitteluun voi vaikuttaa, minkälaisesta datasta on kysymys ja mikä sen volyymi on. Esimerkiksi Big Datan käsittely vaatii erilaisia toimenpiteitä, joihin on omat työkalut tarjolla [6]. Lisäksi prosessin suunnitteluun vaikuttaa se, mitä datasta halutaan kysyä tai mitä sillä halutaan tehdä. Prosessin suunnittelussa otetaan myös huomioon, halutaanko datojen tulevan esimerkiksi yöllisinä eräajoina (engl. *batch processing*) vai jatkuvana latauksena (engl. *stream processing*) [7, s. 29–30].



Kuva 1. Esimerkkikuvaus ETL-prosessista.

ETL tulee sanoista Extract, Transform ja Load, mikä tarkoittaa tietojen automaattista poimintaa, muokkausta ja latausta. Yksinkertaistettuna tässä prosessissa tiedot ladataan jostain lähteestä esimerkiksi operatiivisesta järjestelmästä, rajapinnasta tai siirtotiedostosta. Tämän jälkeen dataa muokataan tietovarastokannan (engl. *Data Warehouse*) muotoon ja lopuksi kirjoitetaan eli ladataan jollekin tietovaraston kannalla sijaitsevaan tauluun, joita yleensä on lukuisia. Se miten monimutkainen datan muokkausvaihe on, riippuu lähdedatasta ja lopputuotteen käyttötarkoituksesta. Muokkausvaiheessa on kuitenkin monia yleisiä perusosatehtäviä, kuten erilaisten datavirheiden tarkistukset, tietotyyppimuunnokset, tietojen historiointi, rivilukumäärätarkistukset, erilaiset loogiset arvopäättelyt ja raja-arvotarkistukset. Lisäksi on sovittu, kuinka puutteellisia tietoja käsitellään. [7, s. 48, 56.]

2.1 Datalähteet

Kuten jo mainittiin, datojen lähteitä voi olla useita erilaisia. Lähteet voivat olla peräsin organisaation omista järjestelmistä tai sen ulkopuolelta. Esimerkkinä omista datalähteistä voidaan pitää organisaation omia perusjärjestelmiä, kuten tietokantoja, ohjelmistoja, ERP- eli toiminnanohjausjärjestelmiä (engl. *Enterprice Resource Planning*), CRM- eli asiakashallintajärjestelmiä (engl. *Customer Relationship Management*) tai web-sivustojen/ohjelmistojen tapahtumia, kuten klikkauksia, kävijämääriä ja latauksia. [7, s. 4, 18.]

Ulkoiset datalähteet voivat olla peräisin muilta tahoilta ja organisaatioilta. Nämä tiedot voivat olla esimerkiksi rahoitukseen, valuuttoihin, säätietoihin, sijanteihin tai väestö- ja kuntatietoihin liittyvää informaatiota. Ulkoisten lähteiden tiedot voidaan hakea esimerkiksi rajapinnoista (engl. *Application Programming Interface*). Tilastokeskus, Ilmatieteenlaitos, Suomen Pankki ja Posti voivat olla esimerkkejä ulkoisista järjestelmistä. Näillä tiedoilla on tarkoituksena yhdistämällä rikastaa olemassa olevia tietoja niin kyselyihin kuin raportteihin. [7, s. 18.]

Pankki- ja finanssialalla tallennettavat tiedot liittyvät asiakastietoihin, tileihin, lainoihin, pankkitapahtumiin, luottokorttitapahtumiin ja tiliotteisiin. Lisäksi tietoja kerätään talletuksista, myynneistä, rahoitusinstrumenteista, osakkeista ja joukkovelkakirjoista [8, s. 2]. Tietoja on siis laajasti, joten tarvitaan laadukas, johdonmukainen ja dokumentoitu ETL-prosessi tietojen säilyttämistä ja hyödyntämistä varten.

2.2 Stage-kerros

Tavallisesti datat luetaan lähteensä niin sanotulle lastausalueelle (engl. *staging*). Tätä aluetta voidaan kutsua myös ”työtilaksi”, jossa datat odottavat varsinaista käsittelyä ja tietovarastoon vientiä [7]. Staging-alueen roolia voidaan pitää hyvin teknisenä, millä voidaan hallita myös kompleksisuutta ja vähentää tietovarastoon kohdistuvaa ”painetta” [9]. Staging-alueelle ei loppukäyttäjällä tule olla pääsyä [7, s. 25, 49].

Tässä vaiheessa datat voidaan haluta pitää vielä ”raakana” eli alkuperäismuodossa, jolloin edes tietotyyppimuunnoksia ei tehdä. Tarkoituksena on saada kaikki tiedot tallennettua ilman virheitä latausvaiheessa, jotta tiedot saadaan ns. SQL:n piiriin eli tietokantahallintajärjestelmään. Tämä toteutetaan yleensä lataamalla kaikki tiedot vakiopituisina merkkijonoina. [7, s. 25.] Staging-alu-

eella voidaan säilyttää vain viimeisimpiä tietoja, mikäli lataukset tehdään erissä. Tässä tapauksessa staging-kantaan suoritettaisiin esimerkiksi SQL:n `TRUNCATE`-käsky, jolloin kanta tyhjennetään aina ennen uuden erän sisäänajoa [10]. Tyhjennyksen ansiosta lataukset voidaan aloittaa aina alusta ilman vaaraa, että sama data tulisi ajettua moneen kertaan samoihin kohdetauluihin.

Vaikka staging-latauksissa ei tyypillisesti ole vielä kompleksia logiikkaa, latausten yhteydessä kuitenkin tehdään ainakin lähdedatan validointeja koskien esimerkiksi aikaleimoja [7, s. 49]. Mikäli esimerkiksi lähdedata on väärälle päivämäärälle tai se puuttuu kokonaan, latauksen tulisi tässä vaiheessa keskeytyä tai jollain muulla tavalla ilmoittaa lähdedatan virheellisyydestä. Tässä voitaisiin hyödyntää esimerkiksi tietokantasähköposteja tai lokitusta, jolloin virhe ja sen ajankohta jäävät talteen [7, s. 61]. Tyypillisesti myös erilaiset ETL- ja integraatiotyökalut ilmoittavat latauksissa havaituista virheistä. Näin päästään helpommin virheeseen johtaneen juurisyyn jäljille.

Joissain tapauksissa voi olla liiketoiminnallisia tarpeita säilyttää eli historioida staging-alueen tiedot tyhjentämisen sijaan. Tässä tapauksessa aluetta kutsutaan persistoivaksi stageksi (engl. *Persistent Staging Area*) [7]. Toinen vaihtoehto lähdetietojen historiointiin voisi olla myös erillinen tietokanta staging- ja tietovarastokerroksen välissä. Tämän kerroksen (tietokannan) funktiona olisi siis pääasiassa historioida datat, jotta staging-kerros voidaan edelleen aina tyhjentää säilyttäen vain viimeisimmät tiedot. Tähän uuteen välikerrokseen ei siis suoritettaisi SQL:n TRUNCATE-/DELETE-käskyjä vain ainoastaan pieniä muutoksia, kuten tietotyyppimuutokset ja puuttuvien tietojen käsittelyt.

2.3 Tietovarasto

Kun tiedot ovat muokattu raportointi- ja kyselykäyttöön sopivaan muotoon, ne ladataan varsinaiseen tietovarastoon (engl. *Data Warehouse*), joka on aikaisempia isompi kokonaisuus. Tietovarasto on suunniteltu tietojen helppoa ja nopeaa hakua eli kyselyä varten. Määritelmä ”tietovarasto” on saanut alkunsa vuonna 1988 IBM System Journal -lehdessä Barry Devlinin ja Paul Murphyn kirjoittamasta artikkelista [7, s. 11]. William H. Inmonia pidetään Yhdysvalloissa yleisesti tietovarastointikonseptin isänä, sillä hän on kirjoittanut lukuisia kirjoja aiheesta, joista muun muassa ensimmäinen on ”*Building the Data Warehouse*” vuodelta 1990. [7, s. 11.]

Tietovarastossa tiedot ovat jaettu omiin asiakohtaisiin tietokantatauluihin esimerkiksi fakta-dimensio-tyylisesti. Tämä on yksinkertaisesti esitetty Kuvassa 1, joka kuvastaa taulujen linkittyvän toisiinsa tietojen yhdistelemiseksi, missä keskimmäinen taulu kuvaa numeerisia arvoja sisältävää

faktataulua. Siihen yhdistyy kategorisia muuttujia sisältäviä dimensiotauluja. Dimensiotaulut antavat kontekstin faktalle, jolloin pysytään ryhmittelemään ja rajaamaan tietoa liiketoiminnalle järkevillä tavoilla. Ne vastaavat yleensä esimerkiksi kysymyksiin ”kuka, mitä, missä, milloin” eli sisältävät kuvailevaa dataa jostakin entiteetistä. Teknisesti tietojen yhdistely tapahtuu pääasiassa tauluihin määriteltyjen avainkenttien avulla (engl. *Primary/foreign key*) [11]. Koska avainten avulla voidaan rivejä yksilöidä, voidaan niiden avulla myös varmistaa, ettei tauluun ladata samoja tietoja useampaan otteeseen. Tietovarastoinnin mallinnus on itsessään todella laaja aihe, joten sen sisällyttäminen tästä yksityiskohtaisemmin tämän työn viitekehyksessä ei ole välttämätöntä.

2.4 Datamartit

Datamartti (*Data Mart suom. paikallisvarasto*) on erillinen kohde tiedon varastoinnille ja on yleensä varsinaista tietovarastoa pienempi. Ne ovat suunniteltu eri käyttötarkoituksille sekä osastoille ja ovat yleensä myös suppeampia. Niiden toiminnot ovat laadittu tiettyjä kyselyitä, toimintoja, käyttäjäryhmiä ja raportointitarpeita varten. Datamartti on siis osajoukko tietovaraston datasta, jossa datat ovat jo valmiiksi puhdistettu, muokattu ja yhdistelty liiketoiminnan tarpeisiin. [7, s. 24.]

Datamartit tarjoavat tiivistetyn tiedon, jonka avulla käyttäjäryhmät voivat tehdä nopeasti perusteltuja päätöksiä. Datamarttien avulla tietoihin on helpompi ja nopeampi päästä käsiksi suhteessa tietovarastoon, joka on yleensä raskaampi ja jossa tiedot ovat enemmän hajallaan. Täten esimerkiksi analyytikkojen ei tarvitse etsiä koko tietovarastoa tuottaakseen raportteja tai visualisointeja, sillä datamarttien tulisi sisältää jo valmiiksi halutut tiedot. [12.]

Koska datamartit ovat yksittäisiä tietokantoja ja sisältävät aihekohtaisia tietoja, niille voidaan laatia erilaiset käyttöoikeudet. Tämä auttaa tietojen hallittavuudessa, sillä organisaatio voi määrittää henkilöt kenellä on näihin tietoihin oikeuksia ja vielä erikseen määritellä oikeuksien tasot, kuten luku-, kirjotus-, suoritus- tai omistusoikeudet. [12.]

Erillisiä datamartteja on melko helppoa luoda, joten kunnollisen tietovarastostrategian puutteessa on vaarana ajautua tilanteeseen, jossa on useampia datamartteja ja osa niistä on täysin redundantteja. Niiden ylläpito voi kuitenkin olla työlästä, mikä taas aiheuttaa ylimääräistä työtä ja kuluerää tietokantakehityksen ja -ylläpidon puolella. [13.] Voi myös olla haastavaa kysellä esimerkiksi asiakastietoja datamarteista, jotka ovat toteutettu toisistaan poikkeavasti. Lisäksi samat tiedot voivat olla useaan otteisiin eri datamarteissa, jolloin on tehty ylimääräistä työtä. [7, s. 26.]

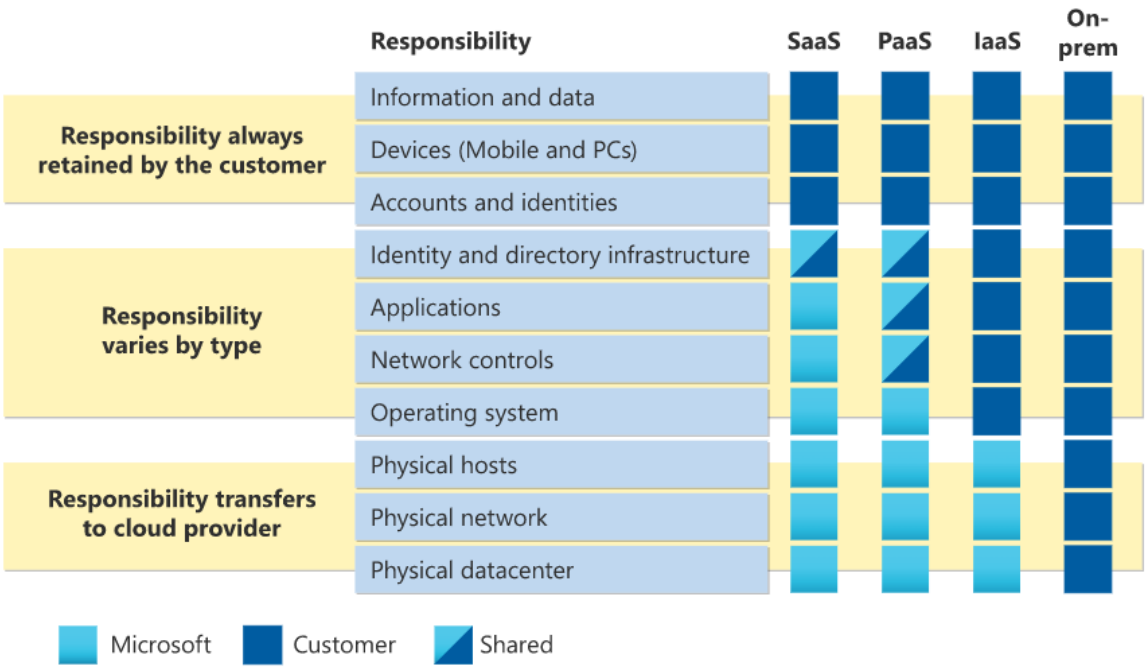
3 Pilvipalvelut

On-premises eli paikallisia datakeskuksia käyttämällä täytyy itse vastata ja hallita koko infrastruktuuria mukaan lukien sen laitteisto, käyttöjärjestelmät, ohjelmistot, verkko- ja palvelinasetukset (ml. kaapelit), palomuurit, tallennustila ja tietoturva. Sen lisäksi, että koko arkkitehtuuri täytyy määritellä ja pystyttää itse, on sitä vielä jatkuvasti ylläpidettävä koko sen elinkaaren ajan. Joskus kuitenkin voi olla liiketoiminnallisia tai lakisääteisiä perusteita ylläpitää yksityistä järjestelmää ja hallita sitä ilman ulkoisia osapuolia. Toisaalta tämä antaa täydet vapaudet valita laitteisto ja ohjelmisto itsenäisesti. [14, s. 17.] Näihin liittyviin seikkoihin palataan vielä tämän luvun lopussa.

Pilvipalveluilla (engl. *Cloud Services*) tarkoitetaan yleensä julkisessa ”pilvessä” toimivia palveluita. Tämä tarkoittaa Internetin yli tarjottavaa kapasiteetti- tai ohjelmistopalveluita, jotka palveluiden käyttäjät saavat käyttöönsä palveluna ilman oman infrastruktuurin hankkimista. [15.] Tämä siis tarkoittaa toimintamallia, joka mahdollistaa yleisesti saatavilla olevan, kätevän ja tarpeiden mukaisen pääsyn vapaasti konfiguroitaviin tietotekniikkaresursseihin, joita voidaan hankkia ja ottaa käyttöön nopeasti vähäisellä hallinnointityöllä tai vuorovaikutuksella palveluntarjoajan kanssa [16, s. 3].

Pilvipalvelut ovat modernimpi tapa pystyttää ja hallita infrastruktuuria. Julkista pilvipalvelua käyttämällä palvelun tarjoaja on täysin vastuussa palveluiden takana olevasta laitteistosta, eikä loppukäyttäjän tarvitse niistä huolehtia, koska hänellä ei niihin ole pääsyä eikä todennäköisesti edes tietoa, missä ne fyysisesti sijaitsevat. [14, s. 17.]

Pilvipalvelut ovat jaettu useampaan eri luokkaan ja näitä nimitetään eri akronyymein (muotoa XaaS). Kuvan 2 mukaisesti pääluokkina ja tunnetuimpina palveluina voidaan kuitenkin pitää SaaS (*Software as a Service*), PaaS (*Platform as a Service*) ja IaaS (*Infrastructure as a Service*). [17, s. 8.] Nämä kolme edellä mainittua luokkaa kattavat suurimman osan palveluista, mutta niiden rinnalle on kehityksen myötä noussut vielä muitakin luokkia kuten FaaS (*Functions as a Service*) ja DBaaS (*Database as a Service*) [18]. Kuva 2 myös ilmaisee, kuinka vastuut jaetaan eri palveluiden välillä suhteessa on-premiseen.



Kuva 2. Pilvipalveluiden pääluokat ja niiden vastuunjako [19].

Tämän lisäksi pilvipalveluiden käyttöönottomalleja on vielä erilaisia, kuten julkinen pilvi (engl. *Public Cloud*), yksityinen (engl. *Private/internal Cloud*) ja hybridi-/yhdistelmäpilvi (engl. *Hybrid Cloud*). Yksinkertaistettuna nämä tyypit määräytyvät sen mukaan, kuka sen omistaa ja keillä siihen on käyttöoikeus. Hinta ja tietoturva ovat siis avainsanoja käyttöönottomallia määriteltäessä. Määrittelyyn myös vaikuttaa, missä palvelinlaitteet fyysisesti sijaitsevat ja kuka niitä ylläpitää.

Yksityisessä käyttöönottomallissa pilvi-infrastruktuuri on varattu yhden organisaation käyttöön. Infrastruktuuri voi olla joko organisaation omistama, hallinnoima ja käyttämä tai kolmannen osapuolen. Infrastruktuuri voi fyysisesti sijaita organisaation tiloissa, muttei se kuitenkaan ole välttämätöntä. [15.] Yksityisen pilvimallin etuina ovat joustavuus, korkeampi valvonta ja yksityisyys sekä paremmat mahdollisuudet skaalautuvuuteen suhteessa paikalliseen on-premiseen. Joustavuudella tässä tapauksessa voidaan tarkoittaa organisaation mahdollisuutta mukauttaa ympäristöä vastaamaan omia liiketoiminnan tarpeita. [20.]

Julkisessa mallissa infrastruktuuri on aina palveluntarjoajan hallussa. Se on kuitenkin varattu julkisesti avoimeen käyttöön. Näin ollen käyttäjillä ei ole tarvetta eikä vastuuta palvelinlaitteiston ylläpidosta eikä huoltamisesta, mikä on yksi julkisen mallin eduista mahdollisten kustannusetujen lisäksi. Julkinen pilvi-infra voi olla esimerkiksi jonkin yrityksen tai hallinnollisen organisaation omistama ja hallinnoima. [15.]

Hybridimallinen pilvi-infrastruktuuri koostuu kahdesta tai useammasta erillisestä käyttöönottomallista. Ne ovat kuitenkin yhdistetty toisiinsa standardoitujen tai omaan teknologiaan perustuvien ratkaisujen avulla, jotka mahdollistavat datan ja sovellusten siirrettävyyden esimerkiksi kuormituksen tasapainottamisen vuoksi pilvien välillä. Ratkaisu voi siis tarjota joustavuutta erikoistilanteisiin. [15.]

Tunnetuimmat palveluntarjoajat ovat Amazon Web Services (AWS), Google Cloud Platform ja Microsoft Azure, joiden tuotteet ovat myös pitkälle automatisoituja [21][22]. Nämä ovat esimerkkejä myös julkisista pilvipalveluiden tarjoajista. Eri yrityksien tarjoamia pilvipalveluita kuitenkin yhdistää palveluiden nopea käyttöönotto, skaalautuvuus, ajantasaisuus, korkea saatavuus, tietoturva, hinnoittelu ja palveluiden hyvät hallintavälineet [23].

Pilvipalveluita hyödyntämällä pyritään säästämään paikallisten konesalien perustus- ja ylläpitokulut. Lisäksi pilvipalvelut noudattavat yleisesti ns. ”*pay-as-you-go*” -maksumetodia eli maksetaan vain käytöstä, joten myös hinnoittelun on mahdollista olla joustavaa. Joustavuudella tarkoitetaan myös mahdollisuutta soveltaa hinnoittelumallia ja tarvittavien tallennus- ja laskentaresurssien määrää. Tämän myötä pilvipalveluiden todelliset kulut ovat pyritty tekemään näkyviksi ja ennustettaviksi. Pilviympäristöissä kustannukset muodostuvat aina seuraavista tekijöistä: laskentatehosta, verkkoasetuksista, levytilasta ja muistista. Kustannuksiin eniten vaikuttaa laskentakapasiteetti.

Pilvipalveluita käyttämällä ei välttämättä tarvitse huolehtia samalla tavalla tietoturvasta eikä myöskään ylläpidosta ja päivityksistä, sillä ne ovat automaattisia. Esimerkiksi Microsoft sijoittaa miljardin (US dollaria) joka vuosi Azuren tietoturvan ylläpitoon ja kehittämiseen suojatakseen asiakkaidensa datoja kyberhyökkäyksiltä vuoden 2023 tiedon mukaan [24]. Tietoturvaa tukee myös erilaiset sertifioinnit, jotka voivat olla maailmanlaajuisia tai alueellisia, kuten GDPR (*General Data Protection Regulation*), joka on EU-alueen yleinen tietosuoja-asetus [23].

Julkiset pilvipalvelut perustuvat hajautettuihin ympäristöihin eli tuotteiden loppukäyttäjillä ei ole mahdollisuutta nähdä tai hallita palveluiden takana tapahtuvaa toiminnallisuutta, kuten laitteistoa (engl. *hardware*). Palveluiden tarjoajien datakeskukset sijaitsevat ympäri maailman ja esimerkiksi Azurella vuonna 2023 yli kuudella kymmenellä eri alueella [23]. Tämä tarjoaa palveluille joustavuutta, korkeaa saatavuutta sekä vähentää asiakkaan ja palvelun välistä viivettä. Pilviresurssia luodessa on mahdollisuus valita palvelun sijainti ja yleensä kannattaa valita se itselleen lähin, joka esimerkiksi Suomen tapauksessa tällä hetkellä olisi Hollanti (West Europe). Tässä kuitenkin kannattaa myös ottaa huomioon, mistä päin maailmaa valtaosa palvelun loppukäyttäjistä sijaitsee.

Kuten tässä vaiheessa on jo tiedossa, pilvipalveluiden käyttö ehdottomasti vaatii Internet-yhteyden. Yhteys datakeskuksiin, joissa pilvipalvelut pyörivät, muodostetaan siis Internetin yli. Yhteys halutaan tietenkin muodostaa tietoturvasta tinkimättä, joten yhteyden turvaamiseksi on mahdollista käyttää VPN-yhteyttä (*Virtual Private Network*). Azure tarjoaa myös ExpressRoute-nimellä kutsuttua yhteyttä, joka on yksityinen suora VPN-yhteys Azuren datakeskuksiin. [14, s. 129.]

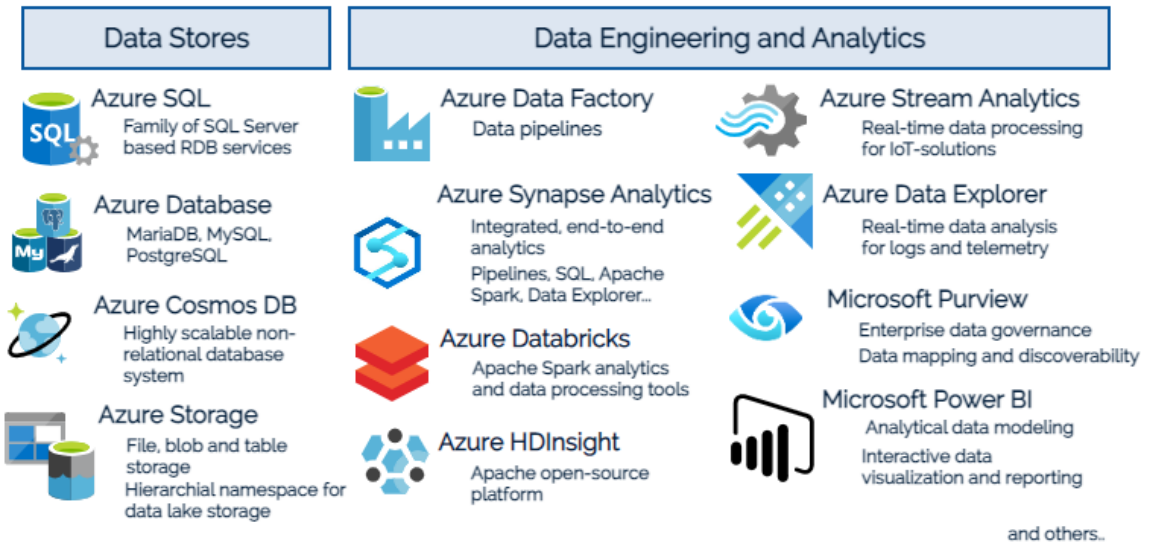
Pankki- ja finanssialalla resurssien ulkoistaminen pilvipalveluihin voi olla haastavampaa, sillä erilaiset säänteleyt tulee ottaa huomioon. Esimerkiksi ESMA:n (Euroopan arvopaperimarkkinaviranomainen) ja EBA:n (Euroopan pankkiviranomainen) ohjeistuksien mukaan ulkoistavan yhtiön tulee ilmoittaa viranomaisille merkittävistä toiminnoista ja tehdä kirjallinen sopimus palveluntarjoajan kanssa. Lisäksi yhtiön tulee olla huolellinen ETA-alueen ulkopuolisissa sopimuksissa ja sisällyttää jatkuvuussuunnitelma sekä selkeä exit-strategia, mikäli palveluiden tarjonta keskeytyy tai heikentyy hyväksymättömälle tasolle. Myös ajantasainen rekisteri tulee pitää ulkoistamisesta, mukaan lukien esimerkiksi pilvipalvelun malli, alkamispäivämäärä ja tietojenkäsittelyn sijainti. [16.][25.]

4 Microsoft Azure

Kuten jo tähän mennessä on mainittu, tässä työssä palveluntarjoajaksi on valittu Microsoftin Azure, joten keskitytään ainoastaan sen tarjoamiin ratkaisuihin. Microsoft Azure on saanut alkunsa lokakuussa vuonna 2008 Microsoftin sisäisestä projektista, jota nimettiin koodinimellä ”*Project Red Dog*”. Palvelu nimettiin aluksi Windows Azureksi, mutta myöhemmin vuonna 2014 muutettiin se muotoon Microsoft Azure. Azure suunniteltiin tarjoamaan organisaatioille ja erityisesti kehittäjille alustaa ilman lisäkoodausta, jolloin kehittäjillä jäisi enemmän aikaa luovempaan työhön esimerkiksi tuotteen ominaisuuksien suunnittelun ja kehittämisen parissa. Tavoitteena oli luoda pilvipohjainen käyttöjärjestelmä, joka mahdollistaisi sovellusten suorittamisen Internetin yli Microsoftin omassa datakeskuksessa. Ensimmäisiin palveluihin kuului mm. mahdollisuus ASP.NET-pohjaisten web-sovellusten ja -rajapintojen suorittamiselle. Hieman myöhemmin käyttöön tulivat myös SQL Azure -niminen pilvipohjainen relaatiotietokantapalvelu ja sen lisäksi tuki muillekin ohjelmointikielille, kuten Javalle ja PHP:lle. Pian näiden jälkeen tuli myös palvelu, joka mahdollisti virtuaalitietokoneiden käytön pilviympäristössä. Azure lanseerattiin virallisesti ja maailmanlaajuisesti vuoden 2010 alussa, jolloin sillä alkoi olla jo enemmän palveluita tarjolla. [26.]

Tänä päivänä Microsoft Azure on skaalautuva ja luotettava julkinen pilvipalveluiden tarjoaja, joka tarjoaa lukuisia eri palveluita kaikilla palvelumalleilla. Azure tarjoaa yli 200 erilaista tuotetta ja pilvipalvelua erinäisiin tarkoituksiin kuten IoT-menetelmiin (*Internet of Things*), Big Dataan ja koneoppimiseen. [26.]

Koska tämä työ keskittyy tietokanta- ja ETL-arkkitehtuurin suunnittelemiseen ja toteuttamiseen Azuren palveluilla, käydään seuraavaksi läpi, minkälaisia tietokantaratkaisuja Azure tarjoaa. Azuren laajaa tuotevalikoimaa on hankala mahduttaa yhteen kuvaan, joten Kuvaan 3 on listattu tämän työn viitekehyksen puolestakin oleellisempia tietokanta-, ETL- ja analytiikkatyökaluja.



Kuva 3. Azuren työkaluja datan hallintaan pilviympäristössä [23].

Seuraavassa kuvassa (Kuva 4) on puolestaan esitetty, mihin edellä mainittuja työkaluja voisi soveltaa pilvialustalla toimivassa dataputkessa. Kuten Kuvasta nähdään, voidaan Azuren työkaluilla rakentaa ketterästi automaattisia ja skaalautuvia dataputkia, ilman tarvetta toteuttaa prosessi täysin ohjelmoimalla perinteisempään tyyliin ohjelmointikielillä, kuten esimerkiksi Pythonilla, SQL:llä tai Scalalla.

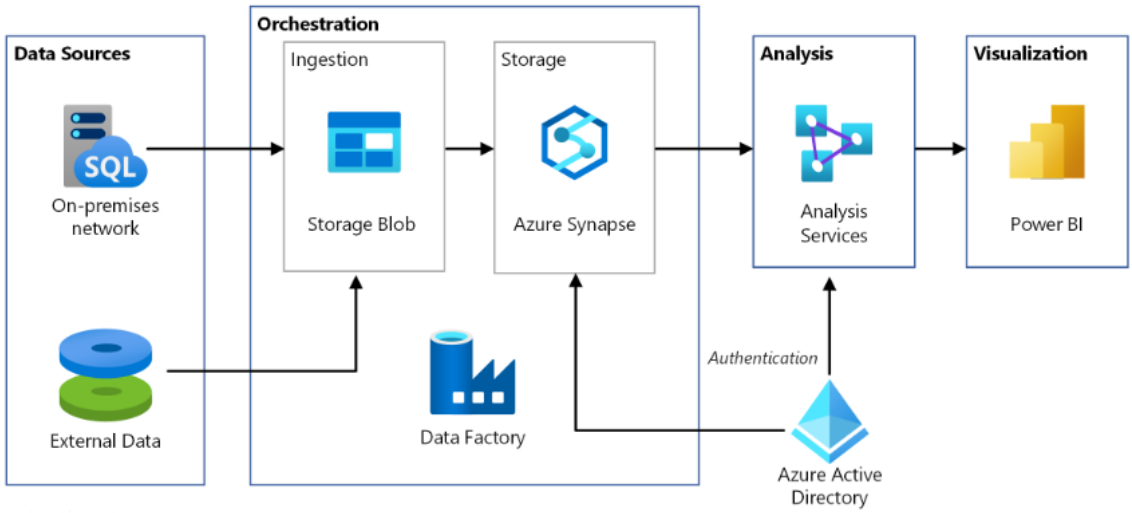
Viimeisimpänä mainittu toteutustapa on luonnollisesti myös työläämpää. Sen lisäksi ongelmana voi myös olla se, että ohjelmoijat toteuttavat koodia hyvin eri tavalla suhteessa toisiinsa. Tämän vuoksi ohjelmat voivat olla henkilöriippuvaisia, joten muilla kuin koodin alkuperäisellä toteuttajalla voi olla hankaluuksia tulkita koodin toiminnallisuutta, mikä voi tehdä sen ylläpidosta ja kehityksestä hitaampaa ja haastavampaa. [7, s. 53.]

Kuvassa 4 näkyvä esimerkkiarkkitehtuuri kattaa inkrementaalisen ELT-prosessin datan lähteestä aina visualisointiin ja raportointiin saakka. ELT on vaihtoehtoinen ETL-prosessin kaltainen tietojen integraatiomenetelmä, mutta siinä muokkaus- ja latausvaiheet tehdään käänteisesti verraten ETL-prosessiin. Menetelmää käytetään lähinnä käsiteltäessä strukturoimatonta dataa, jossa datat varastoidaan lähteestä suoraan tietoaltaaseen (engl. *Data Lake*), joka soveltuu hyvin massiiviselle määrälle strukturoimatonta dataa. Dataa ei siis muokata ennen varastointia, vaan vasta myöhemmin tarpeen vaatiessa. [27.]

Inkrementaalisessa ETL-/ELT-prosessissa halutaan ladata vain muuttuneet tiedot eli jotka poikkeavat edellisestä latauksesta. Tällä vältetään turhaa tietokantaan kohdistuvaa kuormitusta ja liikennettä. [28.] Inkrementaalista prosessia voidaan käyttää esimerkiksi SCD-taulujen (*Slowly*

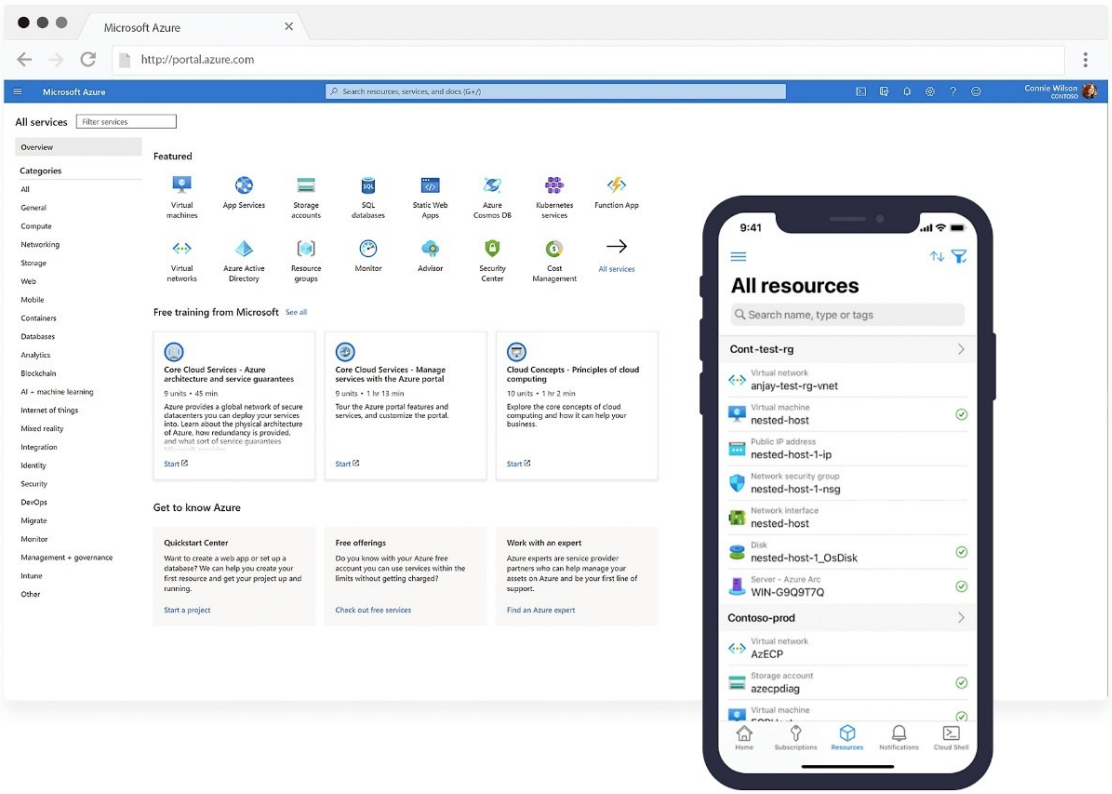
Changing Dimension) delta-latauksessa, jossa dimensiotaulun tiedot vaihtuvat hitaasti. Tällaisia tietoja ovat esimerkiksi asiakas-/osapuolitiedot, kuten tilinumerot, nimet ja osoitteet.

Kuvassa näkyvä dataputki on toteutettu Azuren Data Factory (ADF) -data-alustalla, jossa tämänkin työn dataputki tullaan toteuttamaan. ADF on datan siirtoon, transformaatioon ja dataputkiin kuten ETL-/ELT-prosesseihin tarkoitettu integraatiotyökalu. Sillä voidaan ohjata ja automatisoida tietojen siirtämistä ja muuntamista paikasta toiseen. ADF:n prosessit kuvataan JSON-tiedostoilla, joten prosessit ovat tarvittaessa JSON:lla kehitettävissä sekä myös siirrettävissä versionhallintaan. Monissa tapauksissa ADF:n putkia kehitetään kuitenkin graafisella käyttöliittymällä.



Kuva 4. ELT-putki toteutettuna Azuren työkaluilla [28].

Microsoft Azuren palveluita voidaan ottaa käyttöön kolmella eri tavalla. Helpoin ja intuitiivisin keino on käyttää Microsoft Azure -portaalia (Kuva 5), joka on web-käyttöliittymä. Sen kautta saadaan helposti luotua ja määriteltyä erilaisia palveluita, kuten virtuaalikoneita ja tietokantoja. [14, s. 20.] Tarvittavat välivaiheet ovat näkyvissä ja niitä on hankala ohittaa tahattomasti. Sivusto myös pakottaa täyttämään tietyt kohdat resurssia luodessa. Lisäksi määrittelyt validoidaan ennen käyttöönottoa. Nopea validointivaihe ilmoittaa puutteellisista määrittelyistä.



Kuva 5. Microsoft Azure -portaalia voidaan hallita myös mobiilisovelluksen kautta [29].

Graafisen käyttöliittymän lisäksi on saatavilla myös Azuren monikäyttöinen komentorivityökalu (Azure CLI), jolla myös voidaan muodostaa yhteys Azuren palveluihin. Sen avulla voidaan yhtä lailla luoda Azure-resursseja ja suorittaa pääkäyttäjäkomentoja. Yleensä tehtävät, jotka ovat suoritettavissa komentoriviltä, on mahdollista myös automatisoida. Azure CLI:n avulla voidaan siis automatisoida tehtäviä, kuten palveluiden pystyttämisiä ja dataputkien suorittamista. Azure CLI on asennettavissa Windows-, Linux- ja macOS-käyttöjärjestelmiin. Syntaksi CLI:ssä noudattaa yksinkertaista kaavaa:

```
reference name – command – parameter – parameter value
```

Azurea on myös mahdollista hallinnoida PowerShellistä ilman Azure CLI -komentorivityökalua. Tässä tapauksessa komennot hieman eroavat CLI-komentoihin. Molemmissa vaihtoehdoissa on kuitenkin yhtäläisyyksiä ja tämän vuoksi valinta näiden väliltä voi riippua henkilökohtaisista mieltymyksistä. Molemmat vaihtoehdot ovat suunniteltu Azuren hallintaan skriptien ja komentorivin kautta. [30.] [14, s. 219–220.]

Azure tarjoaa useita relaatio ja ei-relaatiotietokantapalveluita. Relaatiomalli on yksi keino säilyttää dataa ja se on yleisimmin käytetty datamalli. Relaatiolla tarkoitetaan taulua, joten relaatiomalleissa käytetään tauluja ja niiden välisien suhteiden kokoelmaa tietojen esityksessä. Jokainen taulu sisältää vaihtelevan määrän sarakkeita, ja jokaisen sarakkeen tulisi sisältää jokin arvo. Sarakkeet tulee nimetä yksilöllisesti ja varsinkin isoissa kokonaisuuksissa nimeämiseen on yleensä määritelty omia sääntöjä. Tämän myötä jo sarakkeen nimestä on pääteltävissä, minkälaista tietoa sen arvot sisältävät. Relaatiotietokanta on siis tietokanta, joka käyttää relaatiomallia tietojen säilyttämisessä. [8, s. 9.] Relaatiotietokanta on monesti ainut järkevä vaihtoehto sen sopiessa täydellisesti strukturoidulle datalle. Relaatiomalli antaa myös mahdollisuudet erilaisille tietojen mallintamisrakenteille, kuten Ralph Kimballin tähti- tai Bill Inmonin lumihiutalemallille. Myös tässä työssä käytetään relaatiotietokantoja.

4.1 Azure SQL Database

Azure SQL -tietokanta on yksi mahdollinen ratkaisu tietojen säilyttämiselle Azuren pilviympäristössä. Tämä on yksi ensimmäisistä Azuren tarjoamista palveluista, joista jo aikaisemmin mainittiinkin. Azure SQL -tietokanta tarjoaa relaatiotietokantaa palveluna ja tämä voidaan luokitella joko PaaS- tai DBaaS-palveluksi. [14, s. 146.]

Azure SQL -tietokanta tarjoaa monia houkuttelevia ominaisuuksia, jotka voivat olla motiivina perinteisten tietokantojen pilvimigraatiolle. Se kykenee tarjoamaan pilvipalveluiden tunnetuimmat edut eli skaalautuvuuden sekä joustavan hinnoittelun. Kustannuksiin voi esimerkiksi vaikuttaa valitsemalla jokin kolmesta tasosta: Basic, Standard tai Premium. Näillä tasoilla suorituskykyä ilmaistaan yksiköllä DTU, joka tulee sanoista *Database Transaction Unit* tarkoittaen suoritustehoyksikköä. Se on synteettinen mitta, jolla voidaan nopeasti vertailla eri tietokantatasojen suhteellista suorituskykyä. [14, s. 147.] Edullisimmillaan Azure SQL -tietokannan saa käyttöön noin neljän euron kuukausihintaan. Resursseja voidaan tarpeen vaatiessa kuitenkin nopeasti lisätä tai vähentää eli skaalata palveluportaalin kautta.

Koska Azure SQL -tietokantaa tarjotaan PaaS -muodossa, se tarjoaa jatkuvasti viimeisintä vakaata versiota SQL Serveristä automaattisesti. Ylläpito ja tietoturva ovat myös automaattisia. Täten voidaan luopua tietokantapalvelimien fyysisistä hallintavastuista.

Uuden Azure SQL -tietokannan luominen käy nopeasti, ja se voidaan toteuttaa kaikilla edellisessä luvussa mainituilla menetelmillä. Ennen tietokannan tai minkään muunkaan Azure-resurssin luomista, täytyy kuitenkin olla pakolliset ennakkovaatimukset kunnossa. Niihin kuuluvat Microsoft-käyttäjätili, Azure-tilaus ja -resurssiryhmä. Esimerkiksi Azuren palveluportaalina kautta luodessa tulee määritellä vähintään tietokannan nimi, hinnoittelutaso, tietokannan kollaatio, palvelin, resurssiryhmä ja tilaus. Kun pakolliset määrittelyt ovat tehtynä, tietokanta on parin minuutin kuluessa käytettävissä.

Azure SQL -tietokanta sopii hyvin esimerkiksi uusiin projekteihin, moderneihin ja pilvinatiiveihin applikaatioihin sekä myös olemassa oleviin applikaatioihin, joissa on esimerkiksi tärkeää käyttää viimeisimpiä SQL Serverin ominaisuuksia. Azure SQL -tietokanta ei kuitenkaan tue kaikkia ominaisuuksia, kuten tietokantojen välisiä kyselyitä, linkattuja palvelimia (engl. *Linked servers*), SQL Agent Jobeja, järjestelmäobjektien kollaatioita, tietokantasähköposteja, palvelinvälittäjää (engl. *Service broker*) eikä myöskään kaikkea T-SQL syntaksia, kuten esimerkiksi `USE <database>` -komentoa. Tietokantojen vaihtamiseksi on siis muodostettava uusi yhteys. [14, s. 172.]

Azure SQL -tietokantoja käyttämällä kustannukset koostuvat per tietokanta, joten se ei myöskään sovi kovin hyvin tilanteisiin, joissa sovellus sisältää kymmeniä eri tietokantoja. Esimerkiksi jos sovellus käyttää kymmentä itsenäistä tietokantaa, voidaan ehkä puhua pilvinatiivista sovelluksesta, mutta tällä palvelumallilla se ei automaattisesti ole ainakaan kustannustehokkain. Tässä tapauksessa voisi olla parempi vaihtoehto hyödyntää ns. tietokantapoolia, jossa kaikki tietokannat jakavat samat pilviresurssit eli suorittimen, muistin, levytilan ja oikeastaan koko infrastruktuurin. Tämän tyyppiseen käyttötapaukseen Azuren Elastic Pool voisi tarjota optimaalisemman ratkaisun.

4.2 Azure SQL Elastic Pool

Elastic Pool on kokoelma tietokannoista, jossa tietokannat jakavat keskenään samat resurssit. Tämä tarjoaa kustannustehokkaan vaihtoehdon tilanteeseen, jossa täytyy hallita useita tietokantoja, joilla on toisistaan poikkeavat käyttötavat. Elastic Poolin tietokannat toimivat yhdellä palvelimella ja jakavat keskenään määritellyn määrän resursseja. [31.]

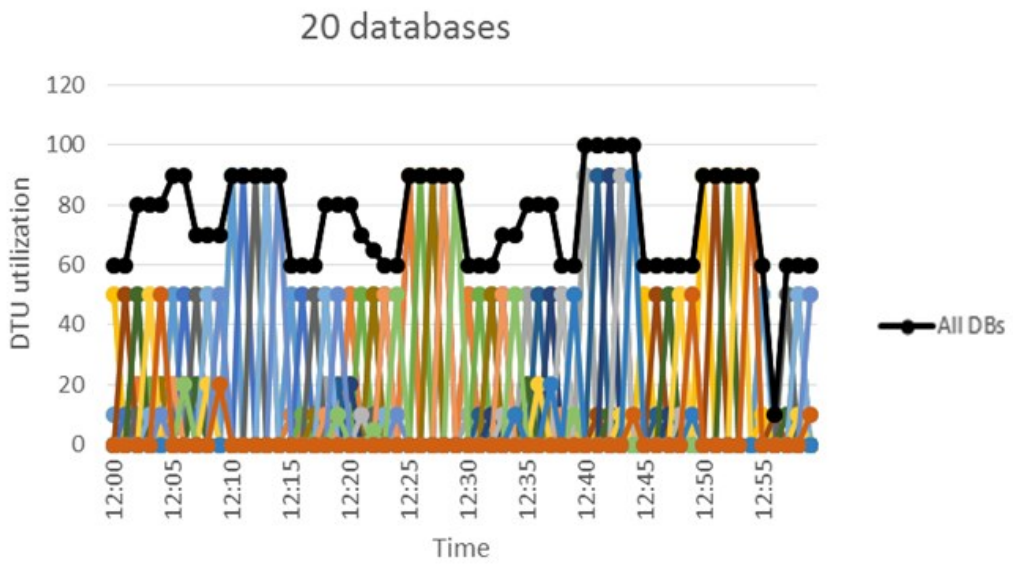
Tässä ratkaisussa täytyy huomioida, ettei veloitus ole tietokantakohtaista, vaan kustannukset koostuvat jokaisesta tunnista, jolloin Elastic Pool on ollut käynnissä suurimman eDTU- tai vCore-kapasiteetin mukaan. Elastic Poolin eDTU:n yksikköhinta on 1,5 kertaa suurempi, kuin yksittäisen

tietokannan DTU-hinta. Säästöä on kuitenkin mahdollista kerryttää sitä enemmän, mitä enemmän pooliin lisätään tietokantoja, sillä kannat jakavat resurssit keskenään. vCore-ostomallissa yksikköhinta on sama yksittäisen tietokantojen kanssa. [31.]

On tavallista, että tietokannoilla on tiettyjä ajanhetkiä, jolloin niihin kohdistuu enemmän kyselyitä. Käydään seuraavaksi esimerkin kautta läpi, kuinka Elastic Poolia voisi hyödyntää. Kuva 6 esittää edellä mainittua tilannetta neljällä eri tietokannalla.

Kuva 6. Tietokantojen aktiivisuus ajan funktiona [31].

Kuvasta voidaan todeta, että tietokantojen käyttö poikkeaa toisistaan ja jokaisella on omat ajanhetket, jolloin kantaan kohdistuu enemmän liikennettä. Kuva 7 puolestaan havainnollistaa vastaavaa tilannetta, mutta kuvaajassa on yhteensä kaksikymmentä eri tietokantaa.



Kuva 7. Mustalla kuvaajalla havainnollisettaan tietokantojen summatasoista käyttöä [31].

Kuvassa näkyvässä esimerkkitapauksessa DTU-taso ei ylitä missään vaiheessa sadan DTU:n rajaa. Elastic Poolin kannalta vastaavaa tilannetta voidaan pitää ihanteellisena, sillä maksimikäytön ja keskimääräisen käytön välillä on suurempi varianssi eri tietokantojen välillä. Näiden tietokantojen maksimikuormitus tapahtuu siis eri ajankohtina ja täten nämä kaksikymmentä eri tietokantaa voivat jakaa niille määritetyn 100 eDTU:ta tuona esimerkin ajanjaksona. Koska DTU:t voidaan jakaa useiden tietokantojen kanssa kesken, niiden kokonaismäärää tarvitaan vähemmän. [31.]

4.3 Azure SQL Database Managed Instance

Azuren SQL -perheeseen kuuluu myös Managed Instance -niminen vaihtoehto, joka mahdollistaa useiden tietokantojen pystytyksen saman instanssin alle, tietokannan automaattisen varmuuskopioinnin sekä ajankohtaisen palautuksen. Se on älykäs, skaalautuva ja tarjoaa myös erittäin korkean yhteensopivuuden SQL Serverin kanssa. Tämä on suositeltava vaihtoehto silloin, kun palvelulta halutaan täyttä hallittavuutta ilman virtuaalikoneiden tai niiden käyttöjärjestelmien hallintaa. Managed Instance tarjoaa korkean hallittavuuden, yhteensopivuuden ja kattavamman määrän muita ominaisuuksia. Täten Managed Instance sopii hyvin esimerkiksi tilanteisiin, joissa halutaan siirtää paikallinen tai itserakennettu IaaS-sovellus pilviympäristöön poistaen samalla hallintakulut. Tämän lisäksi haluttaisiin käyttää kaikkia samoja ominaisuuksia, jotka olivat myös paikallisessa SQL Serverissä. Täytyy kuitenkin ottaa huomioon, että joidenkin SQL Managed Instancen ominaisuuksien aiheuttamia ylimääräisiä kuluja ei voida poistaa.

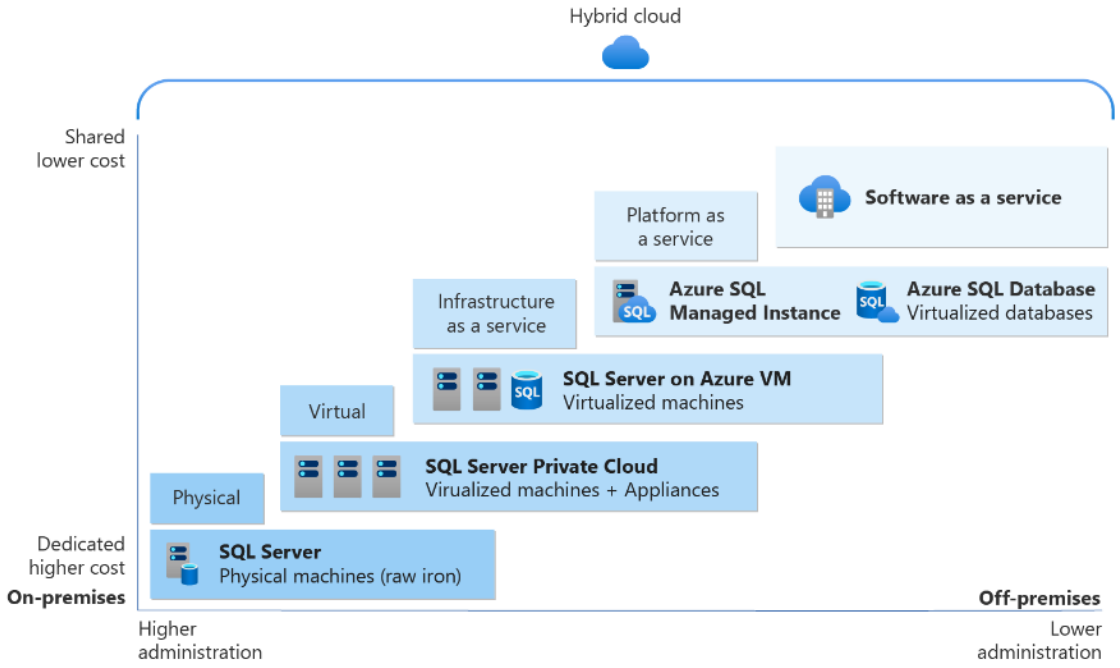
Managed Instance tarjoaa luotettavan tietoturvan ja joukon edistyneitä suojausominaisuuksia tietojen salaamista varten. Esimerkkeinä näihin, Managed Instance seuraa tietokannan tapahtumia ja kirjoittaa ne Azuren tallennustilille sijoitettuun lokitiedostoon. Se myös mahdollistaa kaiken liikenteen suojauksen enkryptaamalla tiedot myös kyselyiden käsittelyn aikana. Managed Instanceen on myös sisäänrakennettu lisätietoturvakerros, joka havaitsee epätavalliset ja mahdollisesti haitalliset yritykset päästä tietokantoihin käsiksi. Epäilyttävistä toiminnoista ja mahdollisista haavoittuvuuksista tulee myös hälytys. Managed Instance tukee myös Azure Active Directory ja Windowsin monivaiheista tunnistautumista. Tämä myös mahdollistaa migraatioprosessin tekemättä muutoksia sovelluksen todennuspinoihin. [32.] [33.]

4.4 Azure SQL -tietokanta virtuaalikoneessa

Kaikki sovellusratkaisut eivät kuitenkaan sovi Azuren PaaS-malliin vaatimalla lähes täydellisen infrastruktuurin hallinnan. Tällaisia tilanteita varten onkin mahdollista luoda Azuren SQL-tietokanta myös Azuren virtuaalikoneeseen, jolloin palvelumalli voidaan kategorisoida IaaS-malliksi. Azuren virtuaalikoneet tukevat Windows Server- tai Linux-virtuaalikoneiden käyttöönottoa Microsoftin datakeskuksessa. Näissä voidaan hallita täysin virtuaalikoneiden asetuksia. Vastuu palvelinohjelmistojen asennuksesta, määrittelystä ja ylläpidosta jäävät kuitenkin käyttäjälle. [14, s. 78.]

On kolme tärkeää kustannustekijää, jotka täytyy huomioida tässä ratkaisussa. Kustannukset koostuvat itse Azuren virtuaalikoneesta, jossa veloitus on minuuttikäyttömallilla. Kustannukset kertyvät siis jokaisesta minuutista, kun virtuaalikone on ollut päällä. Se on kuitenkin mahdollista pysäyttää, jolloin myös kustannuksien kertyminen keskeytyy. Palvelukustannukset sisältävät myös Windows Serverin -käyttöjärjestelmämaksun ja täten Linux-pohjaiset instanssit ovat halvempia, kun niiden käyttöjärjestelmästä ei peritä lisenssimaksua. Azuren virtuaalikoneeseen asennettavien lisäohjelmistojen kustannukset ja asianmukaiset lisenssit ovat myös käyttäjän vastuulla. Täten, kun SQL Server -instanssi luodaan Azuren virtuaalikoneen päälle, sisällytetään siihen myös SQL Serverin lisenssikustannukset. Tässä kannattaa huomioida, jos organisaatiolla on jo olemassa olevat SQL Server -lisenssit, nämä voidaan sisällyttää myös Azuren virtuaalikoneessa olevaan SQL Serveriin, eikä erillistä lisenssiä enää tarvita. Kolmas kustannustekijä on vielä Azuren tallennuskustannukset. [14, s. 78, 169.]

Yhteenvetona SQL Server Azuren virtuaalikoneessa voi olla hyvä ratkaisu esimerkiksi uusille tai olemassa oleville sovelluksille, jotka vaativat korkeatasoista hallintaa ja mukauttamista IaaS-ympäristössä sekä myös lähes täydellisen yhteensopivuuden on-premise SQL Serverin kanssa. Näiden lisäksi, jos vielä haluttaisiin luopua paikallisten laitteistojen ylläpidosta ja sen tuottamista kuluista. [14, s. 172.] Virtuaalikoneratkaisu ei ehkä ole suositeltava vaihtoehto ilman erityistä syytä, mutta sovelluksen pilvimigraatio voisi olla relevantti käyttötapaus, sillä paikallisen SQL Serverin migraatio Azuren virtuaalikoneeseen ei juurikaan eroa tietokantojen migraatiosta paikallisesta palvelimien välillä. Migraatio Azuren virtuaalikoneeseen voidaan suorittaa minimimäärällä muutoksia alkuperäiseen sovellukseen. Yhteenvedon lopuksi Kuvassa 8 visualisoidaan kaikki edellä mainitut ratkaisut kuvaajaan.



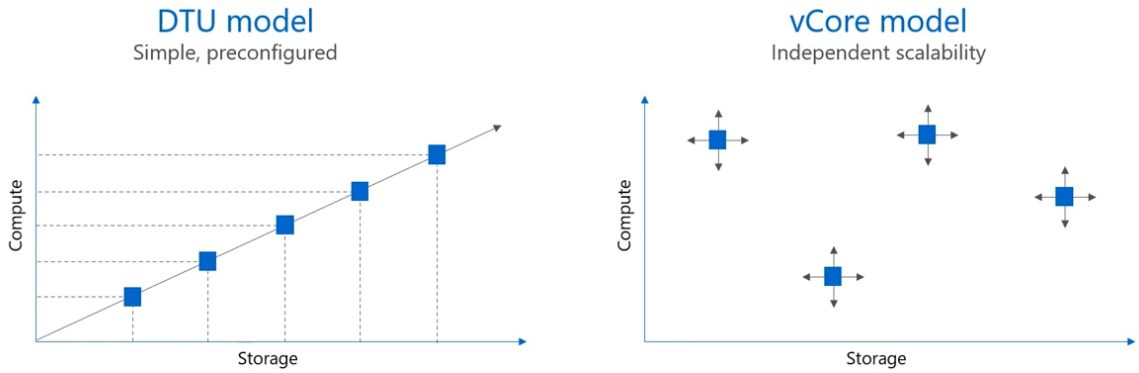
Kuva 8. Tietokantapalveluiden vertailu kustannuksien ja vastuiden suhteen [34].

4.5 Hinnoittelumallit

Palveluiden hinnoitteluista on jo tähän mennessä ollut mainintoja, mutta käydään niitä vielä hieman tarkemmin läpi. On olemassa erityyppisiä tilauksia, joilla saadaan Azuren palveluita käyttöön. Aivan aluksi tarvitaan kuitenkin Microsoft-tili, joka voi olla yksityinen eli itseluotu henkilökohtaiseen käyttöön tai työ-/koulutili, jonka järjestelmänvalvoja on myöntänyt akateemiseen- tai yri-

tyskäyttöön. Microsoft tarjoaa myös ilmaisen kokeilukäyttöön tarkoitetun tilauksen, jossa palveluita on mahdollista kokeilla aina kahteensataan dollariin asti kuukauden ajan. Kokeilujakson päätyttyä palvelut poistetaan, eivätkä ne myöskään enää toimi ilman tilauksen jatkamista. Pay-as-you-go -tilauksessa maksetaan käytetyistä resursseista liittämällä tilaukseen maksukortti. Organisaatiot voivat myös hyväksyä laskutustavan. Toinen yleinen sopimus on yrityssopimukset (*Enterprise agreements*), joissa sitoudutaan käyttämään tietty määrä Azuren palveluita seuraavan vuoden ajan. Kyseinen summa maksetaan etukäteen. Mikäli sitouduttu määrä ylitetään, maksu lisäkäytöstä voidaan suorittaa kvartaaleittain tai vuosittain. [14, s. 32–33.] Näiden tilauksien lisäksi on muitakin malleja, joista voi lukea lisää esimerkiksi Microsoftin verkkosivuilta.

Azuren tietokantapalveluiden hinnoitteluun puolestaan vaikuttaa hinnoittelumalli, joita on kahdenlaista. Kuva 9 havainnollistaa hinnoittelumallien eroa kuvaajien kautta. Tähän mennessä on jo mainittu DTU-pohjaisesta mallista, jolla tarkoitetaan tietokantatapahtumayksikköä. Tämä on hieman yksinkertaistetumpi hinnoittelumalli, koska se tarjoaa valmiiksi tasapainotettuja, niin sanotusti niputettuja laskenta- ja tallennusresurssipaketteja. DTU siis keskittyy tietokannassa tapahtuvien työkuormien ja liikenteen määrään. [35.]



Kuva 9. Hinnoittelumallien laskentateho tallennustilan funktiona [36].

Toinen vaihtoehto on vCore-malli (virtuaalinen prosessoriydin), joka antaa mahdollisuuden skaalata resursseja (Kuvassa 9 ulottuvuuksia) itsenäisesti ja valita vaihtoehto, joka vastaa parhaiten mahdollisesti olemassa olevan järjestelmän resursseja. Tällöin pystytään tarkemmin määrittelemään ytimien lukumäärä, muisti ja tallennustila. vCore-malli myös mahdollistaa esimerkiksi palvelimettoman (*serverless*) Azure SQL -tietokannan kustannuksien säästämiseksi. Palvelimeton tietokanta keskeytyy automaattisesti epäaktiivisina aikoina ja jatkaa automaattisesti tietokantojen käyttöä, kun toiminta palautuu. [36.]

Kahden eri hinnoittelumallin taustalla on Microsoftin pyrkimys tarjota asiakkailleen joustavuutta vastaamalla erilaisiin asiakastarpeisiin ja käyttötapauksiin. Esimerkiksi uusissa sovelluksissa ja ratkaisuissa ei välttämättä ole vielä olemassa historiatietoa sovelluksen teknillisten vaatimuksien suhteen. Asiakkaat voivat ajatella resurssien tarpeita yksinkertaistamalla ajatusta lineaarisesti skaalautuvalla resurssien määrällä (DTU-malli). Niitä ei tarvitse alkuun osata määritellä sen yksityiskohtaisemmin.

Vaikka pilvipalveluista puhuttaessa tuodaan lähes aina esille kustannustehokkuus sekä niiden läpinäkyvyys ja ennustettavuus, ei pilvipalvelut riippuen sovelluksesta ole automaattisesti aina halvempi ratkaisu. Esimerkiksi mikäli kaikki palvelimet vain siirretään pilveen eli ryhdytään käyttämään virtuaalipalvelimia, voi jatkossa koitua ongelmia, mikäli uuden ympäristön ylläpitoa jatketaan vanhaan tapaan. Julkiseen pilviympäristöön siirtymisessä täytyy siis myös ajattelutapaa muuttaa. Huomiota tulee kiinnittää esimerkiksi ylläpitotottumuksiin ja palvelimien optimointiin. [22.]

5 Pilvimigraatio

Pilvimigraatio on prosessi, jossa paikallisia sovelluksia, tietoja tai muita järjestelmiä ryhdytään siirtämään pilviympäristöön. Migraatio voi tarjota lukuisia uusia mahdollisuuksia esimerkiksi skaalautuvuuden, joustavuuden tai kustannussäästöjen ansiosta. Jopa pienempikin startup-yritys voi rakentaa suuryrityksen IT-infrastruktuurin hyödyntämällä pilvipalveluita [22]. Pilvi myös tarjoaa pääsyn kehittyneisiin sovelluksiin ja teknologioihin [3].

Syitä pilvimigraatiolle on monenlaisia. Jokaiselle sovellukselle tulee jossain vaiheessa elinkaaren aikana hetki, jolloin sitä on aika päivittää syystä tai toisesta. Se voi olla vaatimuksien uudelleenmäärittelyä, uusien ominaisuuksien implementointia, käyttöliittymän uudelleensuunnittelua, optimointia, modernisointia tai jotain muuta. [14, s. 240.]

Monet organisaatiot pitävät määräajoin myös laitteiston päivitysjakson, jolloin voi olla syytä pohtia, uusiako paikalliset laitteistot vai olisiko aika ulkoistaa laitteistoa pilveen. Pilvimigraatioiden taustalla voi siis olla infrastruktuuristrategian muuttuminen esimerkiksi ylläpitokustannuksien vuoksi. Organisaatio on voinut myös tulla tilanteeseen, jossa sen paikallinen datakeskus on saavuttanut fyysisen kapasiteetin rajan, jolloin tilaa laajentamiselle ei yksinkertaisesti riitä. Laajentaminen tällöin olisi liian kallista ilman, että esimerkiksi laitteiston jäähdytys ja virransyöttö kärsivät. Vastaavassa tilanteessa voisi olla mahdollista hyödyntää jo aikaisemmin mainittua hybridipilviratkaisua, jolloin yritys voisi osittain luopua fyysisten konesalien toiminnasta ja ylläpidosta tai ainakin niiden laajentamisesta [14, s. 240]. Tällöin osa resursseista pyörisi omassa konesalissa ja osa siirrettäisiin pilveen.

Pilvimigraatio ei siis välttämättä ole joko/tai -valinta infrastruktuurien välillä, vaan migraatio voi antaa yritykselle mahdollisuuden hyödyntää esimerkiksi julkista pilvi-infrastruktuuria laajentaakseen sekä kompensoidakseen paikallisia resurssejaan [37, s. 17]. Tällaisissa tilanteissa voidaan alkaa suunnittelemaan, kuinka siirtää palvelut mahdollisimman tietoturvallisesti ja kustannustehokkaasti pilveen.

Esimerkkinä myös pandemia (COVID-19) ja sitä myöhemmin syttynyt sota Ukrainassa ovat osakseen vaikuttaneet organisaatioiden suunnitelmiin IT-infrastruktuurin kehittämisestä. Huomattiin, että perinteinen IT-infrastruktuuri voi estää liiketoimintaa joillain aloilla. Mikäli kuvitellaan skenaario, jossa miljoonat ihmiset lisäävät radikaalisti erilaisten palveluiden kulutusta, voi tällöin asiakkaille suunnatut järjestelmät, kuten verkkosivustot ylikuormittua nopeasti. Palveluiden takana

tapahtuvat toiminnallisuudet, kuten tietokannat ja sovelluspalvelimet kuormittuvat. Tämä osoitti, ettei paikallinen infrastruktuuri ole aina ensisijaisesti suunniteltu joustavuutta ajatellen. Näiden lisäksi vielä myös yritysten toimintatavat ovat muuttuneet ja etätyöstä on tullut monille uusi normaali**.** [35.] Tutkimuksen mukaan pandemian ja sodan myötä on yleisimmin panostettu kyberturvallisuuteen etenkin julkisella sektorilla. Tutkimustuloksen mukaan yleisin ajatus on myös hyödyntää yhä paremmin sekä konesaleja, että pilviympäristöjä yhdessä. Osa myös varautuu siihen, että Suomi olisi irti verkosta. Tutkimus on Elisa Oyj:n toimeksi antama ja aineisto kerättiin 23.5.-7.6.2022 välisenä aikana. Tutkimuksen kohderyhmänä oli suurten yritysten ja organisaatioiden ICT-päättäjät Suomessa ja tutkimukseen osallistui 209 ICT-päättäjää, jotka edustivat 196 eri yritystä ja julkiorganisaatioita. [38.]

Pilvimigraatioiden kasvavassa trendissä yksi olennaisimmista asioista on tietenkin kyberturvallisuus. Kun arkaluontoisempia ja kriittisempiä tietoja tai sovelluksia siirretään pilveen, tarve kyberturvallisuudesta huolehtimiseen ja asiaankuuluvien säädösten noudattamiseen kasvaa entistä tärkeämmäksi. Organisaatioiden tulisi ottaa käyttöön kattavia tietoturvastrategioita varmistaakseen tietojensa ja sovelluksiensa suojaamisen pilviympäristössä. [3, s. 79.]

5.1 Migraation vaiheet

Perinteisemmän on-premises -tietojärjestelmän tai sovelluksen siirtäminen pilveen voi tapahtua hyvin eri tavoin riippuen siirrettävästä järjestelmästä, sen laajuudesta sekä tulevasta palveluntarjoajasta. Se voi olla monimutkainen prosessi vaatien huolellista suunnittelua ja toteutusta onnistuneen lopputuloksen varmistamiseksi. Tärkeimpiä haasteita prosessissa ovat yhteensopivuus, tiedonsiirto, tietoturva, vaatimustenmukaisuus, kustannuksien hallinta, suorituskyky sekä henkilöstö ja niiden taidot. Haasteita on mahdollisuus kuitenkin lievittää tehokkailla strategioilla ja suunnitelmalla. Mahdolliset pilvimigraation tuottamat riskit on myös syytä tunnistaa, jotta niihin voidaan ennalta varautua. [3, s. 75.]

Prosessi yleensä noudattaa ja sisältää Kuvan 10 mukaisia vaiheita. Prosessin jakaminen vaiheisiin auttaa hallittavuudessa ja mahdollistaa paremman keskittymisen kuhunkin vaiheeseen. Lisäksi se myös edesauttaa selkeämpään aikataulutukseen. Jokaisesta vaiheesta syntyy dokumentaatiota, mikä taas auttaa prosessin raportoinnissa. [3, s. 75.]

Kuva 10. Karkeasti määritellyt migraatioprosessin vaiheet [3].

Prosessi lähtee liikkeelle sen jälkeen, kun tarve migraatiolle on tunnistettu. Pilvistrategia pitäisi aina lähteä liiketoiminnan tarpeista keskustellen tavoitteista ja uusien teknologioiden mahdollisuuksista [22]. Organisaatiota kiinnostaa myös migraation kustannukset, joten alkuvaiheisiin kuuluu myös niiden arviointia.

5.1.1 Migraatioanalyysi

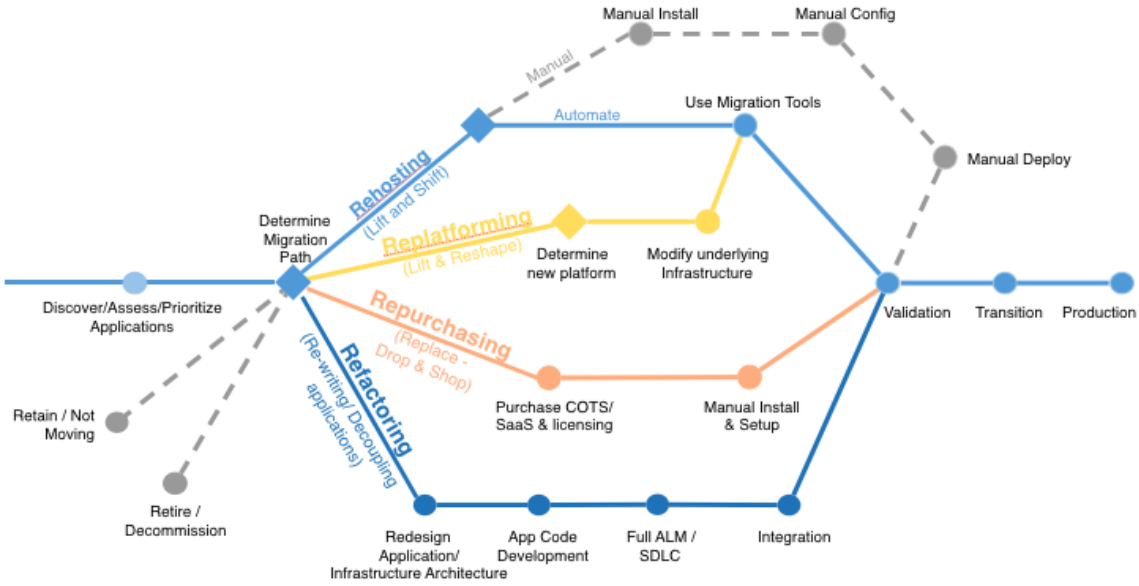
Kun tarve migraatiolle on tunnistettu ja prosessia ryhdytään edistämään, sen ensimmäisiin vaiheisiin kuuluu nykyisen ympäristön ja sovelluksen kokonaisvaltainen arviointi. Selvitetään nykytila ja tunnistetaan infrastruktuurin ja sovelluksien valmius. Samalla voidaan estimoida migraatioprosessin laajuus. Täytyy tunnistaa mm. palvelimet, tietokantojen ja sovelluksen objektit sekä rajapintakuvaukset. Lisäksi, kuinka data liikkuu ja missä sitä säilytetään. Jo näihin kysymyksiin vastaaminen voi auttaa valitsemaan sopivan menetelmän. Arvioinnin tarkoituksena on tutkia migraation toteutettavuutta ja tunnistaa mahdolliset esteet, haasteet ja riskitekijät. Tässä vaiheessa olisi hyvä piirtää sovelluksen alkuperäinen arkkitehtuuri sen hahmottamisen helpottamiseksi. On myös tärkeää ottaa huomioon sovelluksen kriittisyys ja monimutkaisuus, jotta migraatio osataan priorisoida oikein. [3, s. 79.]

Analyysin/arvioinnin suorittamiseen voidaan hyödyntää myös esimerkiksi Microsoftin tarjoamia migraatiotyökaluja, kuten DMA:ta (*Data Migration Assistant*). Se on työpöytäsovellus, johon voidaan määritellä siirrettävä tietokanta ja sen tuleva sijainti, joka voi olla esimerkiksi Azure SQL -tietokanta. Tämän jälkeen suoritetaan arviointi ja työkalu kertoo mahdolliset yhteensopivuusongelmat, jotka se havaitsi siirrettävästä tietokannasta. Tämä on nopea ja suoraviivainen vaihe ja sen avulla saadaan heti kartoitettua joitain mahdollisia prosessin hidastavia tekijöitä.

5.1.2 Strategian suunnittelu

Kun nykyinen infrastruktuuri ja sovellus on kuvattu ja niiden valmius arvioitu, voidaan aloittaa tarkempi migraatioprosessin suunnittelu. Oletettavasti tässä vaiheessa pilvipalveluntarjoaja on

myös jo tiedossa. Sen lisäksi, kun nykyisen sovelluksen ja infrastruktuurin rakenne on palautettu mieleen, on tässä vaiheessa ehkä jo helpompi suunnitella, kuinka varsinainen migraatio teknisesti toteutetaan. Kuvassa 11 on esitetty kuusi erilaista polkua on-premises -järjestelmästä pilven puolelle. Tämä kuva voi auttaa hahmottamaan valintaa ja sen sisältämiä välivaiheita. Valittu menetelmä ei välttämättä tule sisältämään kaikkia kuvaan merkittyjä välivaiheita, mutta kuva voi kuitenkin toimia suuntaa antavana. Sen lisäksi polkuja voidaan jaotella yhä useampaan eri luokkaan.



Kuva 11. AWS:n 6R-strategia pilvimigraatiolle [39].

Kuvassa vasemmalla alhaalla kaksi skenaariota on merkitty harmaalla. Näissä on todettu, ettei varsinaiseen prosessiin ryhdytä. Ensimmäinen näistä skenaarioista on ”**Retain**”, joka tarkoittaa, ettei järjestelmää tulla siirtämään pilviympäristöön, vaan se jatkaa toimintaansa vanhaan malliin. Syynä päätökseen voi olla esimerkiksi migraatiosta saavutettavan potentiaalisen hyödyn vähäpätöisyys.

Retire on toinen skenaario, jossa sovellusta ei siirrettä pilveen. Syynä voi olla, että järjestelmä on vanhentunut (engl. *legacy*). Tämä tarkoittaa, ettei se vastaa enää nykyajan vaatimuksiin ja sen migraatio tulisi olemaan liian monimutkainen, hankala ja kallis, jolloin myös hyötysuhde jää pieneksi. Järjestelmä on siis päätynyt sellaisenaan elinkaarensa päähän ja sen tarvetta tulevaisuudessa voidaan myös kyseenalaistaa.

Jäljelle jää neljä vaihtoehtoa, joista jokaisella suoritetaan pilvimigraatio jollain tavalla. Käydään nämä läpi järjestyksessä Kuvan mukaan ylhäältä alas, joten aloitetaan **rehostista**. Tämä tunnetaan

myös termillä ”*lift & shift*” ja suomeksi menetelmää voidaan kutsua järjestelmän uudelleenisännöintinä. Tämä menetelmä vaatii kaikista vähiten muutoksia alkuperäiseen järjestelmään eli sovellus pyritään siirtämään sellaisenaan. Usein tämä menetelmä toimii osana isompaa muutosta. Järjestelyä helpottaa, jos organisaation omassa järjestelmässä olevat lisenssit on mahdollista siirtää käytettäväksi myös pilviympäristössä. Tässä menetelmässä sovellus siirretään IaaS-ympäristöön ja sovellus pysyy ”ehjänä”. Rehost-menetelmä voidaan tehdä manuaalisesti tai sitä voidaan helpottaa esimerkiksi käyttämällä joitain migraatioon tarkoitetuista työkaluista, joita esimerkiksi Microsoft tarjoaa Azureen siirtyessä.

Vaikka lift & shift on menetelmänä näistä yksinkertaisin, vaatii se kuitenkin erilaista osaamista sekä asioiden huomioonottamista. Näitä voivat olla käytettävyystavoitteiden määrittäminen, pilvipalveluiden palvelutasosopimuksien (SLA) ymmärtäminen, kustannuksien hallinta, käyttöoikeuksien hallinta, backupit ja niin edelleen. Ylläpitoa ei tule myöskään ajatella enää vanhalla tavalla, vaan on tärkeä jatkaa muutosta, jotta saavutettaisiin suurempia hyötyjä. Esimerkiksi monoliittisia sovellusarkkitehtuureja voitaisiin alkaa pilkkomaan mikropalveluiksi tai hyödyntämään serverless-ominaisuuksia. [37, s. 31.]

Replatform vastaa eniten edellä mainittua ”*lift & shift*” -menetelmää. Tässä menetelmässä täytyy sovellusta kuitenkin jossain määrin muokata, jotta se voisi hyödyntää pilvialustalle tyypillisiä ominaisuuksia. Sovellusta voi myös joutua muokkaamaan parantaakseen sen käytettävyyttä ja toimivuutta pilvialustalla. Esimerkiksi sovelluksen koodia osittain refaktoroimalla tai sovelluksen rakennetta hieman muuttamalla. [37, s. 32.]

Repurchasing: Mikäli sovellukseen tarvitaan niin laajasti refaktorointia, että se koituu liian kalliiksi ja aikaa vieväksi, voi kolmannen osapuolen pilvipohjainen SaaS-sovellus korvata vanhan siirrettävän sovelluksen, jos sopivia sellainen on tarjolla. Mikäli kyseessä on kuitenkin yksilöllinen ja johonkin tiettyyn käyttötapaukseen erikoistunut sovellus, joudutaan alkuperäistä sovellusta refaktoroimaan, jotta se saadaan yhteensopivaksi pilvialustalle. [40.] Tässä joudutaan hylkäämään koodipohjaa osittain korvatakseen se uudella. Tämä on yleensä aikaa vievä prosessi, joten siihen ryhdytään silloin, kun nykyiset ratkaisut eivät täytä liiketoiminnan tarpeita.

Refactoring on polku, jossa päätetään refaktoroida omaa sovellusta laajemmin ennen pilveen siirtymistä. Tässä tapauksessa tullaan siirtymään PaaS-malliin. Laajempaan refaktorointiin voidaan ajatella kuuluvaksi sovelluksen ja/tai tietokantojen arkkitehtuurin uudelleen ajattelua, suunnittelua ja kehitystä, jotta siitä saataisiin pilvinatiivimpi versio. Tämä voi johtua liiketoiminnallisesta tarpeesta, jossa sovellukselta vaaditaan tehokkaampaa suorituskykyä ja skaalautuvuutta.

Toisaalta refaktoroinnin tarve voi myös johtua siitä, ettei sovellusta alun perin suunniteltu pilvipohjaiseksi, jolloin kaikkia ominaisuuksia ei välttämättä pystytty huomioimaan. Refaktorointi voi olla esimerkiksi monoliittisemman sovelluksen muuttaminen mikropalveluarkkitehtuuriin tai siirtyminen palvelittomaan arkkitehtuuriin. [39.] Sovelluksen ydinarkkitehtuuri kuitenkin säilyy ennallaan.

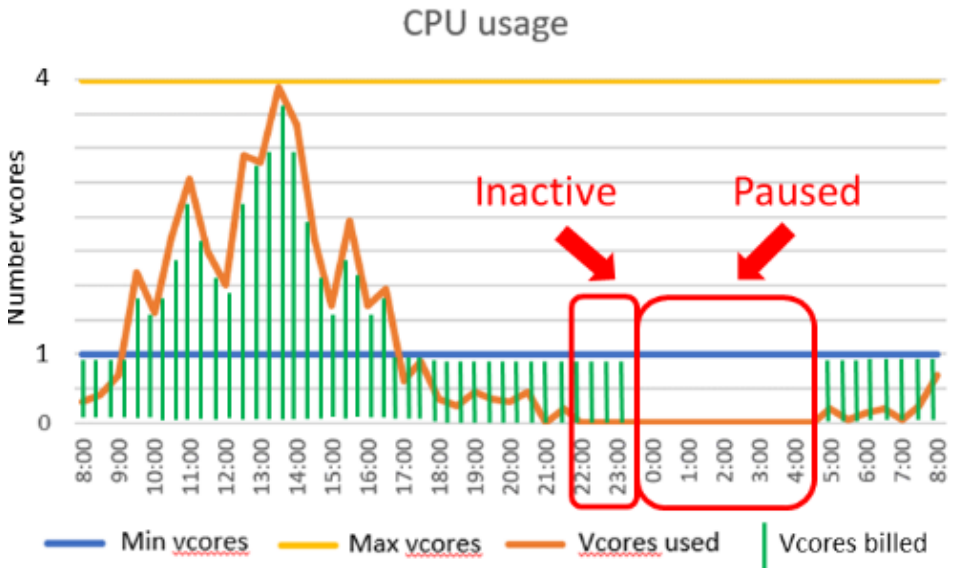
5.2 Migraation suorittaminen valituilla menetelmillä

Voidaan siis todeta, että menetelmän valinta on useampien tekijöiden summa. Kuitenkin tämän työn viitekehyksen puolesta tärkeämpänä voidaan pitää tietoa ja arviointia sovelluksen backend-puolen yhteensopivuudesta sekä toimivuudesta Azuren pilviympäristössä. Yhteensopivuus oli yksi tyypillisistä haasteista pilvimigraatiossa, mitkä mainittiin luvun 4.1 ensimmäisessä kappaleessa.

Mikäli yhteensopivuus tuottaa kohtuuttoman paljon haasteita, saadaan tällöin selville sovelluksen osa-alueet, jotka niitä aiheuttavat. Tämän jälkeen saadaan kartoitettua mahdollisen refaktoroinnin laajuus, jota kautta voidaan taas tarkastella aikaisemmin mainittuja migraatiostrategioita uudestaan. Myös toteaminen alkuperäisen sovelluksen toimimattomuudesta pilviympäristössä olisi tutkimustulos jo itsessään.

Tässä työssä palvelumalliksi on valittu yksittäinen Azuren SQL-tietokanta. Järjestelmä siirrettään siis PaaS-alustalle ja se implementoidaan testiympäristössä. Azure SQL -tietokantaa käsittelevässä luvussa mainittiin sen ominaisuuksia, sille tyypillisiä käyttötapauksia sekä myös kaksi eri hinnoitteluvaihtoehtoa. Mainittiin myös, että Azure SQL -tietokanta voidaan toteuttaa myös palvelimettomana. Tässä mallissa etuna oli se, että tietokanta ”nukahtaa” silloin, kun se ei ole ollut tarpeeksi pitkään aktiivisessa käytössä, eikä tällöin kerry myöskään kustannuksia. Tietokanta taas herää,

kun sen käyttö aktivoituu (Kuva 12). Automaattista nukahtamisaikaa on mahdollista säätää portaalin kautta kymmenen minuutin tarkkuudella tai sen voi ottaa kokonaan pois käytöstä.



Kuva 12. Palvelimettoman tietokannan toiminta [41].

Serverless-tasolla laskentaresurssit skaalautuvat automaattisesti ja laskenta-/hinnoittelumallina on jo aikaisemmin esille tullut vCore. Kustannukset lasketaan käytettyjen vCorejen määrästä per sekunti eli voidaan sanoa, että malli noudattaa myös aikaisemmin esille tullutta ”*pay-as-you-go*” -käsitettä. Nämä ominaisuudet ovat perusteluna valinnalle, sillä ne soveltuvat parhaiten siirrettävän tietokannan toimintamalliin. Täsmäytystyökalua käytetään vain säännöllisin väliajoin, kuten päivittäin tai kuukausittain, joten tietokannan ei tarvitse olla jatkuvassa valmiustilassa. Toinen hyvä seikka on, että Azure SQL -tietokanta omaa myös listan ei-tuettuja ominaisuuksia. Joten mikäli sovelluksen tietokanta saadaan suhteellisen vähäisellä vaivalla toimimaan tässä palvelussa, saadaan se tarvittaessa toimimaan myös muissa palvelumalleissa.

Tässä migraatiossa ei siirretä tietokantaa objekteineen ja datoineen suoranaisesti ns. ”*lift & shift*” -menetelmällä, vaan haluttu tietokantarakenne pystytetään pilveen ”puhtaana”. Menetelmä voidaan luokitella osittain Replatform- ja Refactoring-migraatiomalleihin. Tähän on olemassa sopiva työkalu nimeltään Ahjo, jolla tietokanta saadaan populoitua halutun malliseksi. Ahjo on Python-pohjainen ja avoimeen lähdekoodiin perustuva tietokantojen kehittämiseen ja hallintaan tarkoitettu työkalu. Ahjo hyödyntää myös tunnettuja Python-kirjastoja, kuten SQLAlchemya ja Alembicia. Tietokantojen hallinnan suurimpia haasteita ovatkin usein objektien versioimattomuus ja useat toisistaan poikkeavat ympäristöt. Ahjo auttaa myös tähän haasteeseen tukemalla tietokan-

nan sisältämien skeemojen ja objektien versiointia, jotta esimerkiksi aina olisi mahdollisuus pystyttää haluttu tietokantarakenne Ahjon skripteillä. Ahjo on ALM Partnersin kehittämä maksuton sovellus, joka on ladattavista Python Package Indexistä (PyPI). Se tekee yhdessä versionhallinnan (Git) kanssa kehityksestä myös läpinäkyvää ja seurattavaa. [42.] Prosessissa tullaan etenemään Kuvassa 13 esitettyjen vaiheiden mukaisesti.

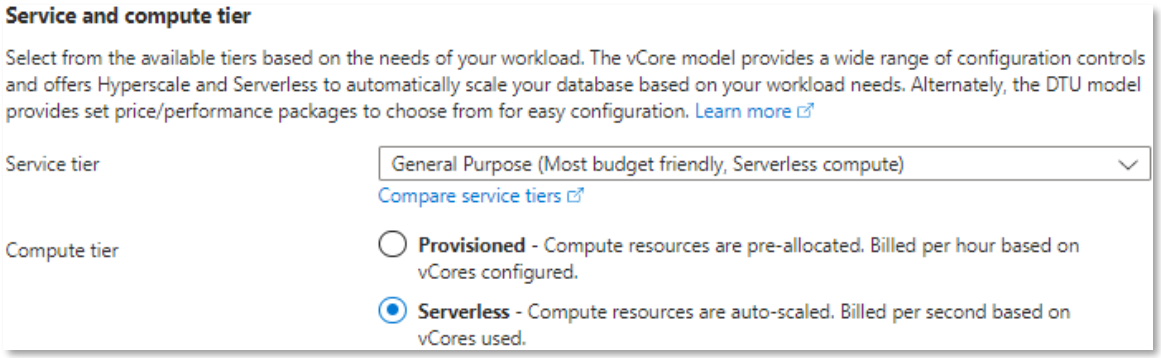
Kuva 13. Vaiheet täsmäytyssovelluksen pilvimigraatioon.

Azure-tietokanta tullaan pystyttämään Azure CLI -komennolla, jotta siihen voidaan erikseen määritellä tietokantakohtainen kollaatio. Ensimmäisellä kerralla tietokanta pystytettiin Azuren portaalin kautta, mutta myöhemmin populointivaiheessa SQL-skripteissä ilmeni ongelmia mm. taulujen ja kenttien luomisessa ja tarkemmin ottaen niiden nimeämisessä. Pian ongelmaksi selvisi palvelin- ja tietokantakollaatioiden välinen eroavaisuus. Joten luontivaiheessa täytyy määritellä erikseen, ettei käytössä tule olemaan palvelimen CI-muotoinen (Case-insensitive) oletuskollaatio, vaan tietokannan katalogikollaatio annetaan erikseen ja se pakotetaan CS-muotoon (Case-sensitive).

Tietokantakollaatiolla tarkoitetaan sääntöjoukkoa, joka määrittelee tietokannan, taulun ja sarakkeiden vertailun ja niiden järjestäytymisen. Yksinkertaisesti kuvattuna kollaatio vaikuttaa siihen, kuinka SQL Server käsittelee merkkijonoja, jolloin virheellinen kollaatio voi johtaa odottamattomiin tuloksiin. [43.] Tietokanta voidaan luoda Azure CLI:n `az sql db create` -komennon avulla, johon kollaatio voidaan parametrisoida:

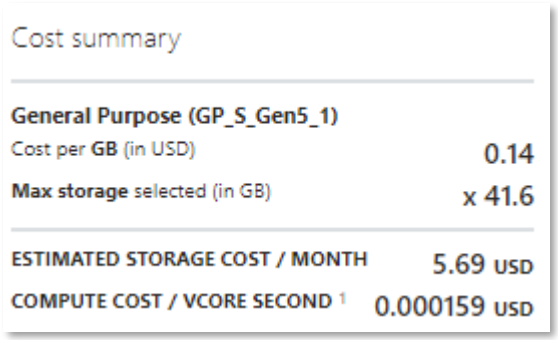
```
--catalog-collation "DATABASE_DEFAULT" --collation "Latin1_General_CS_AS"
```

Toisaalta tietokannan pystyttäminen yhdellä komennolla PowerShellissä on myös nopeampaa. Yllä olevasta komennosta on esitetty vain pätkä, jota aiemmin on määritelty myös muita parametrejä, kuten tilaus, resurssiryhmä, SQL Server -palvelin ja tietokannan nimi. Tämän jälkeen on kuitenkin kirjauduttava vielä Azure-portaalin puolelle tarkistamaan hinnoittelumalli (Kuva 14), sillä serverless-laskentamalli ei kyseisellä komennolla tule oletuksena.



Kuva 14. Serverless-laskentamallin valitseminen Azure-portaalista.

Kun haluttu hinnoittelumalli on valittu palveluportaalin kautta, tulee tietokannan kustannusarvio näkyviin Kuvan 15 mukaisesti. Kun muutokset ovat hyväksytty, tietokannan uudelleenskaalaus kestää muutaman minuutin, minkä jälkeen se on taas valmiina käytettäväksi.



Kuva 15. Kustannusarvio palveluportaalin näkymästä.

Seuraavaksi juuri luotu tietokanta populoidaan Ahjon Python-pohjaisilla skripteillä. Azure SQL -tietokantaan julkiyhteydet ovat oletuksena estetty. Palveluportaalin kautta voi kuitenkin määritellä palomuurisäännöt, joissa listataan mahdolliset yhteydet.

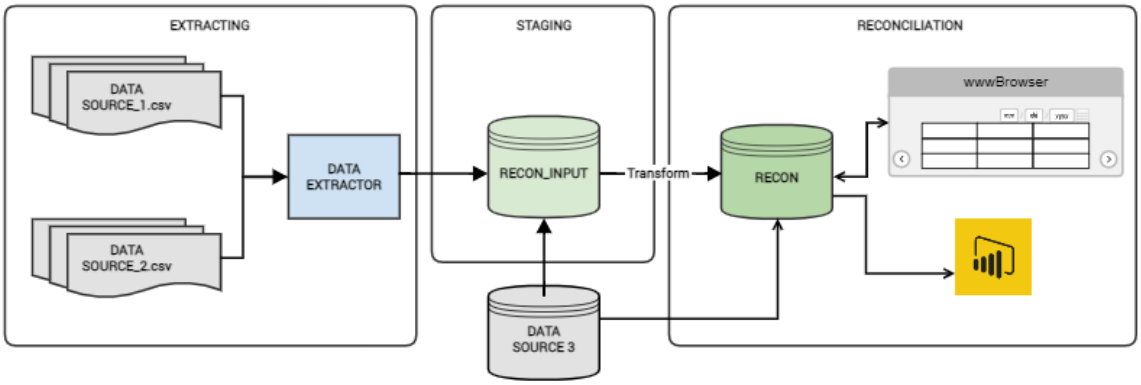
Tietokantaan otetaan yhteys Azuren virtuaalikoneella, jossa populointiskriptit suoritetaan. Virtuaalikone on etukäteen luotu Azure-resurssi ja se on ainoa määritelty tapa ottaa yhteys kyseiseen tietokantaan, sillä vastaavasti toimittaisiin myös asiakasympäristössä. Virtuaalikoneeseen saadaan yhteys esimerkiksi Azuren portaalin kautta. Yhteys voidaan ottaa esimerkiksi Azure Bastionin kautta, joka mahdollistaa turvallisen etäyhteyden. Tällöin virtuaalikoneen näkymä aukeaa selainikkunaan. Virtuaalikoneeseen kirjaudutaan omilla AAD-tunnuksilla (Azure Active Directory) käyttäen monivaiheista tunnistautumista (engl. *Multi-Factor Authentication, MFA*).

Populoinnin mukana haluttavat objektit eli mm. proseduurit, näkymät ja taulut sijaitsevat versionhallinnassa. Versionhallinnassa sijaitseva repository kloonataan Git-versionhallintatyökalua käyttäen virtuaalikoneeseen, jossa Ahjon skriptin tullaan suorittamaan. Tämän jälkeen siirrytään kloonattuun hakemistoon, jossa lähdekoodit sijaitsevat. Kansiossa sijaitsevaan Ahjon config-tiedostoon määritellään tarvittavat tiedot, kuten populoitavan tietokannan nimi, palvelin ja portti. Tämän jälkeen komentokehotteessa suoritetaan seuraava Ahjon komento: `ahjo deploy config_local.jsonc`, jossa JSONC-tiedosto on kyseinen konfigurointitiedosto. Tämän jälkeen ollaan vaiheessa, jossa uusi tietokanta sijaitsee Azuren pilvialustalla ja se on populoitu versioiduilla täsmäytystyökalun objekteilla.

Jotta täsmäytyksiä voidaan suorittaa Azuren pilvialustalla, täytyy sitä varten tehdä vielä erikseen Azureen liittyviä konfigurointeja. Tätä varten versionhallintaan luotiin erillinen ”recon_almbank” -niminen repository, joka sisältää mm. ”ADF_CONTROL_TABLE” -nimisen tietokantataulun. Tämä kohdetietokantaan tuleva taulu sisältää ”mappaukset” eli se määrittelee, mitä tietoja ladataan lähdetietokannasta ja mihin tauluihin ne siirretään kohdetietokannassa. Näin ADF-putki tietää, mitä tietoja täsmäytykseen halutaan hakea.

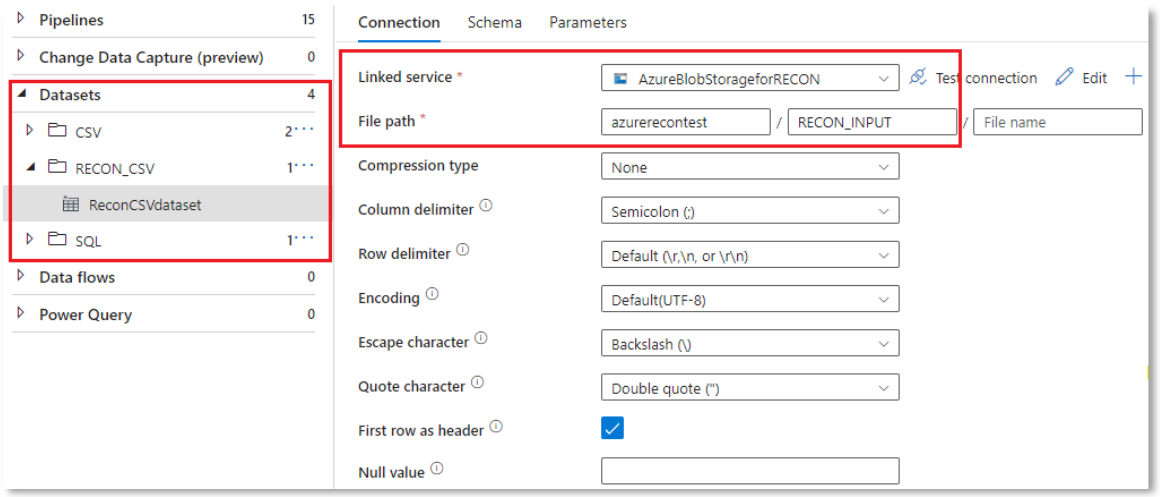
Lisäksi täsmäytystä varten tulee määritellä kolme erillistä kontrollitaulua, joiden perusteella lähdetietojen eroavaisuuksia tutkitaan. Näihin määritellään esimerkiksi täsmäytyksen syötteeksi tulevat taulut ja niiden solut tai rivit, joiden tietoja halutaan vertailla keskenään. Myös säännöt, millä tasolla eroavaisuuksia tutkitaan. Tämän ”recon_almbank” Git-repositoryn mukana tulee myös versiointiin liittyviä objekteja ja refaktoroitu versio täsmäytysprosessin käynnistävästä alkuperäisestä `ui.spSTART_RUN` -nimisestä ”Master”-proseduurista. Jotta myös nämä objektit saadaan kohdetietokantaan, on kloonatussa hakemistossa jälleen suoritettava sama `ahjo deploy` -komento, jolloin tietokanta jälleen täydentyy tarvittavilla objekteilla.

Nyt uuteen tietokantaan on tuotu tietokantaobjekteja kahdesta eri Git-repositorysta. Seuraavaksi täytyy rakentaa datan liikutteluun tarvittava putki alkuperäisestä sovelluksesta poiketen Azuren Data Factoryssa (ADF). Täsmäytykseen tulevat tiedot voivat Kuvan 16 mukaisesti tulla csv-muotoisista lähdetiedostoista tai toisesta tietokannasta. Alkuperäisessä täsmäytystyökalussa käytetään toista ETL-sovellusta csv-tiedostojen siirtämiseen täsmäytystietokannan kohdetauluihin. Lisäksi alkuperäisessä arkkitehtuurissa on myös ”RECON_INPUT”-kanta, johon csv-tiedostot ja mahdollisesti myös tietokantalähteestä tulevat tiedot aluksi ladataan, mikäli ne tarvitsevat muokkausta ennen täsmäytysprosessia.



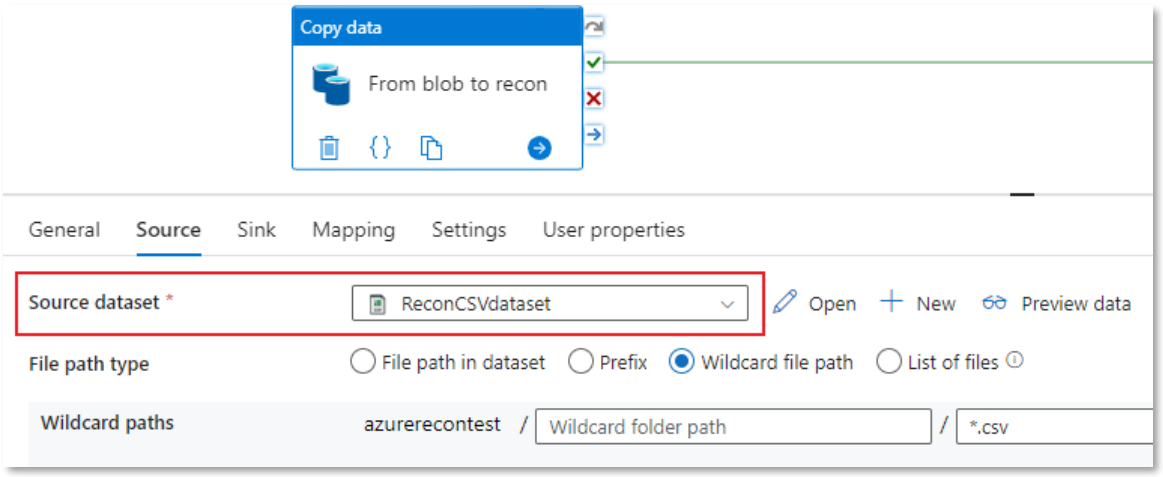
Kuva 16. Datan kulku alkuperäisessä sovelluksessa.

Pilvimigraation myötä voidaan luopua kyseisestä lataussovelluksesta. Sen sijaan hyödynnetään ADF:n ominaisuuksia csv-tietojen siirtämiseen tietokantaan. Tätä varten Azureen täytyy luoda Blob Storage -tallennusresurssi, jossa lähdeaineistoina käytettäviä csv-tiedostoja säilytetään. Blob on akronyymi ja tulee sanoista ”binary large object”. Blobeissa voidaan säilöä miltei mitä tahansa tiedostoja aivan kuten omalla paikallisellakin tietokoneella. [14, s. 101.] Azure Data Factoryssa luodaan ”Dataset”, johon tämä Azure Blob Storage -tallennusresurssi linkitetään (Kuvan 17).



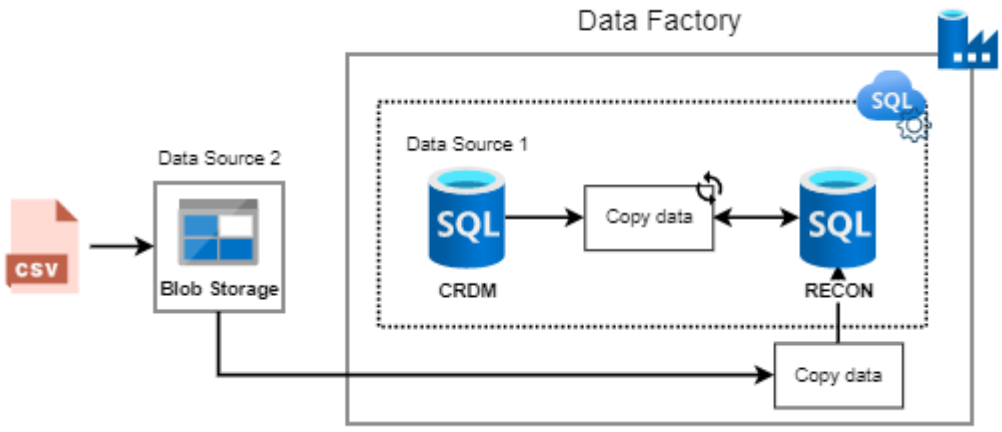
Kuva 17. Blob Storagesta luotu Dataset, jota käytetään Copy data -aktiviteetin datalähteenä.

Nyt Blob Storage on linkitetty Data Factoryssa sijaitsevaan ADF-datasettiin. Tämän jälkeen Copy data -aktiviteetissä kyseistä datasettiä voidaan käyttää tietolähteenä (Kuva 18).



Kuva 18. Blob Storagesta luodun datasetin käyttö Copy data -aktiviteetin lähteenä.

Kun varsinainen täsmäytysprosessi aloitetaan, alkaa proseduurien automaattinen suoritusketju. Ketju alkaa `ui.spSTART_RUN` ”Master”-proseduurin kutsusta, joka kutsuu `ctrl.spLOAD_INPUT_Master` -proseduuria. Tämän jälkimmäisen proseduurin funktiona on hakea täsmäytykseen tulevat datat lähde- tai ”RECON_INPUT”-kannasta ja siirtää ne täsmäytystietokannan (RECON) kohdetauluihin, jotka tulee etukäteen luoda. Aikaisemmin mainittiin, etteivät Azure SQL -tietokannan ominaisuuksiin lukeudu tietokantojen väliset kyselyt ja linkatut palvelimet. Näin ollen, myös `ctrl.spLOAD_INPUT_Master` -proseduurin toiminnallisuudesta joudutaan luopumaan ja myös se tullaan korvaavaan Data Factoryn ominaisuuksilla. Kuvassa 19 on mm. esitetty, kuinka ADF:n datan kopiointi -aktiviteetillä tietoja siirretään eri lähteistä kohdekantaan.



Kuva 19. Datan kulku Azure-versiossa.

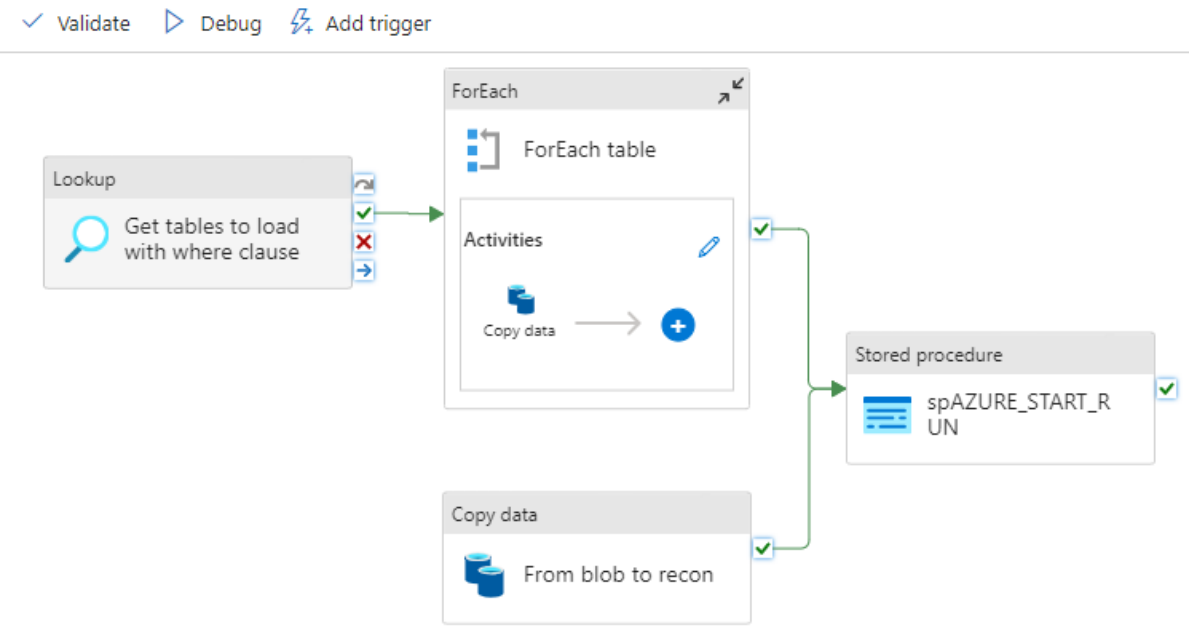
Putkessa käytetään toista Azure SQL -tietokantaa (CRDM), joka on luotu saman Azure SQL Server -instanssiin. Se ajaa ikään kuin saman asian kuin Kuvassa 16 esitetty ”RECON_INPUT”-kanta. Sieltä

halutut tiedot siirretään RECON-kantaan samaan aikaan, kun toisesta lähteestä (Data Source 2) siirretään csv-muotoiset tiedot myös RECON-kantaan. Tämä on parametrisoitu prosessi Data Factory -alustalla ja parametrit määritellään ennen prosessin käynnistämistä. Parametreihin lukeutuvat kohde- ja lähdetietokannat, datapäivämäärät sekä kohdetaulun nimi, johon csv-tiedoston datat siirtyvät. Siirron yhteydessä tehdään myös tietotyyppimuunnokset.

5.3 Toimivuuden testaaminen ja automatisointi

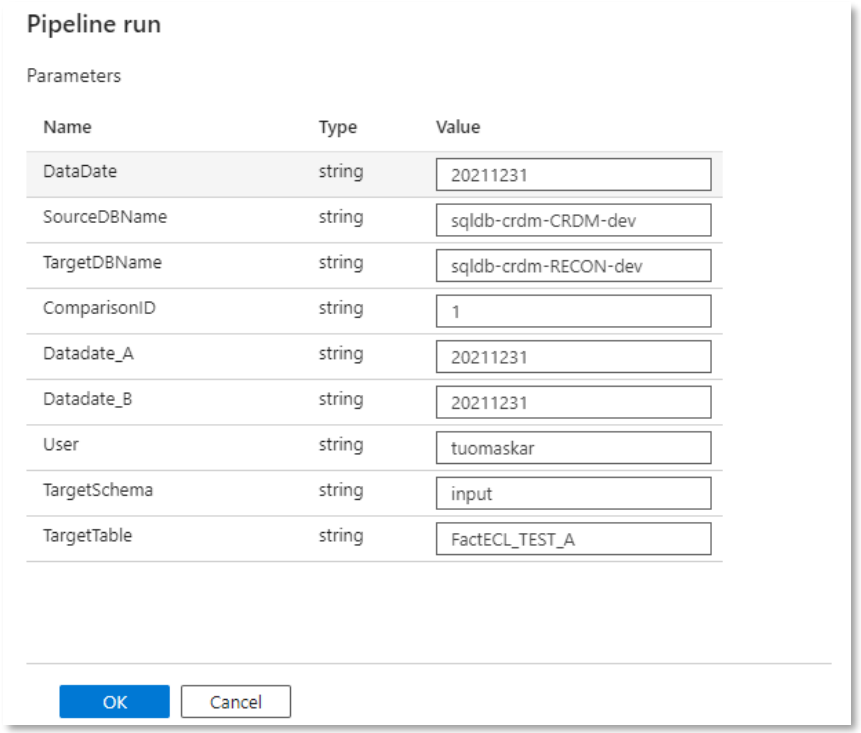
Jotta täsmäytyksen toimivuutta voidaan testata, luotiin sitä varten testiaineistot A ja B, jotka molemmat sisältävät saman rivi- ja sarakemäärän. Aineisto B luotiin Kuvassa 19 näkyvään CRDM-tietokantaan ja sen aineiston pohjalta luotiin csv-tiedosto (aineisto A), johon arvoja kuitenkin hieman muutettiin kertomalla niitä satunnaisilla kokonaisluvuilla. Csv-aineisto siirrettiin Azuressa sijaitsevaan Blob-tallennuskohteessa sijaitsevaan ”RECON_INPUT” -nimiseen kansioon.

Tässä vaiheessa on myös Azuren Data Factory -alustalle luotu parametrisoitu dataputki (Kuva 20), joka vastaa Kuvassa 19 esitettyä kaaviota. Putkessa on viisi eri aktiviteettia, jotka suoritetaan vihreiden nuolien osoittamassa järjestyksessä. Vihreä nuoli tarkoittaa seuraavan aktiviteetin aloittamista vasta, kun edellinen on onnistunut.



Kuva 20. ADF sisältää lukuisia eri aktiviteettejä, joita voi hyödyntää rakentaessa automatisoituja dataputkia. Aktiviteettiä klikkaamalla pääsee tarkastelemaan sen parametreja ja muita määrittelyjä.

Yksi keino suorittaa dataputki on vain painaa Kuvassa 20 näkyvää ”Debug”-painiketta portaalin kautta, jolloin aukeaa parametrien määrittelyikkuna Kuvan 21 mukaisesti. Dataputken käynnistäminen onnistuu tarpeen vaatiessa monella muullakin keinolla kuten esimerkiksi Azure CLI -komennolla: `az datafactory pipeline create-run` tai PowerShell-komennolla: `Invoke-AzDataFactoryV2Pipeline`. Tällöin dataputken parametrit voidaan määritellä esimerkiksi JSON-tiedostoon, johon suorituskomennossa viitataan parametrilla `-ParameterFile .\pipelineparametrit.json`. Dataputki on mahdollista käynnistää myös REST-API -rajapintakutsun kautta. Tarvittaessa käynnistys on mahdollista myös ajastaa sekunnin kymmenesosan tarkkuudelle ja ajastukselle voi myös määritellä sen toistuvuuden, kuten esimerkiksi, monenko minuutin, tunnin, päivän, viikon tai kuukauden välein prosessi automaattisesti suoritetaan.

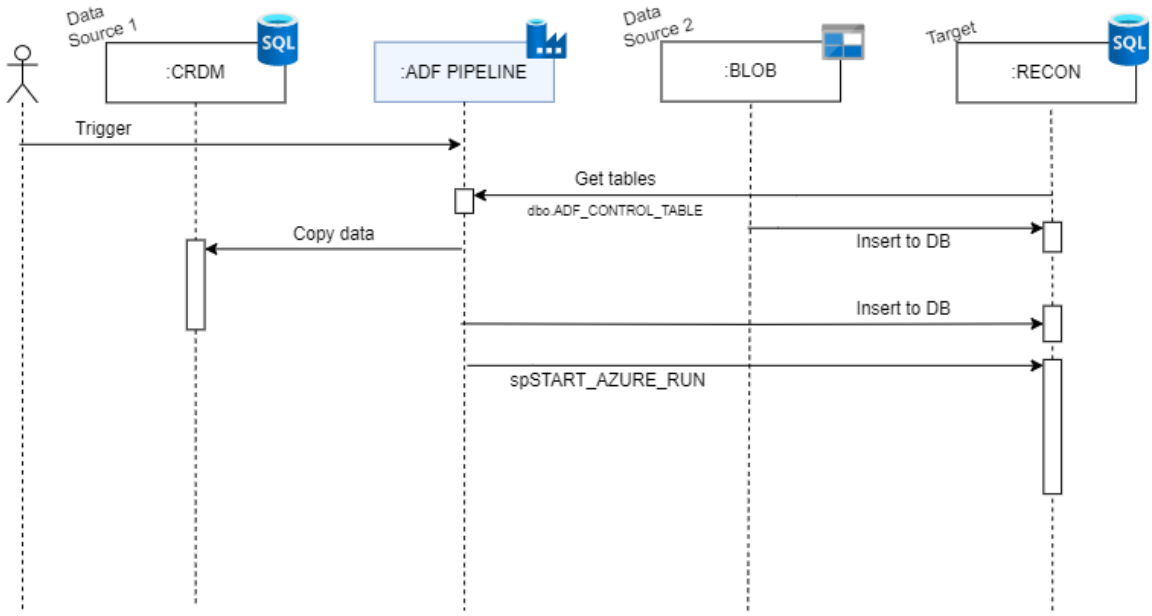


Kuva 21. Dataputken käynnistys manuaalisesti portaalin kautta.

Taulut, joihin tiedot CRDM-kannasta siirretään, tulee luoda kohde- eli RECON-kantaan etukäteen. Myös csv-tietoja varten on RECON-kantaan luotava taulu etukäteen. Näiden lisäksi on vielä täytettävä aikaisemmin mainitut kolme kontrollitaulua, jotka sisältävät täsmäytykseen liittyvät säännöt ja määrittelyt. Jotta manuaalista työtä saadaan vähemmäksi ja testaamista nopeutettua, luotiin tätä projektia varten tehtyyn ”recon_almbank” -repositoryyn SQL-skripti, joka luo RECON-kantaan tyhjät taulut testidatoille, täyttää täsmäytyksen kolme kontrollitaulua sekä lisää

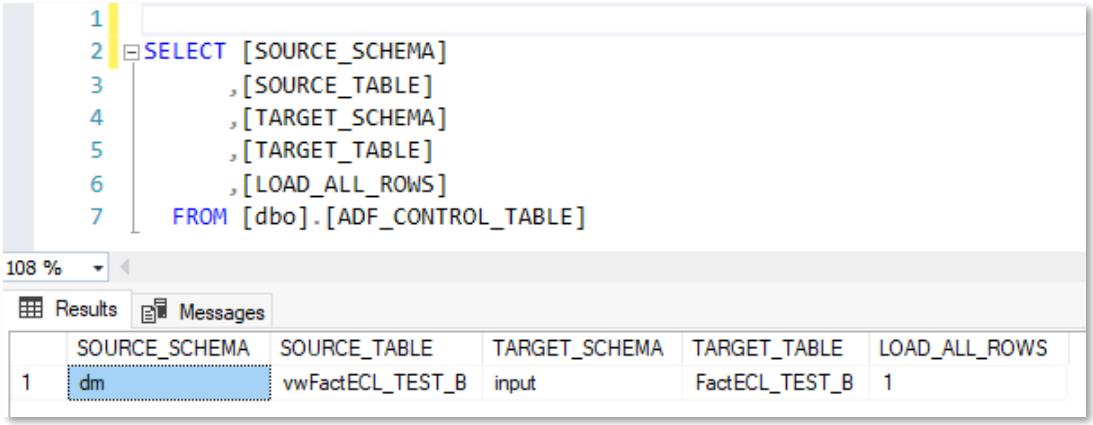
ADF_CONTROL_TABLE -tauluun yhden rivin. Nämä tiedot saadaan vietyä RECON-kantaan käyttäen Ahjon `ahjo data` -komentoa, joka etsii ja suorittaa kaikki SQL-tiedostot, jotka ovat pudotettu *./database/data/* -hakemiston alle.

Kuvassa 20 näkyvä dataputki luotiin aluksi portaalissa. Kun sen toimivuus todettiin, on se mahdollista muuttaa myös JSON-muotoon, jolloin se voidaan viedä esimerkiksi versionhallintaan. Tällöin kehitystä on helppo versioida. Näin tehtiin tässäkin testitapauksessa ja ADF-dataputken sisältämä JSON-tiedosto vietiin recon_almbank -repositoryyn, joka aikaisemmin kloonattiin Azuren virtuaalikoneeseen. JSON-muodossa oleva pipeline saadaan otettua käyttöön `az datafactory pipeline create` -komennolla, johon jälleen määritellään parametreiksi resurssiryhmä ja Azure Data Factory -instanssin nimi, jonne dataputki ilmestyy. Näiden lisäksi määritellään dataputken tuleva nimi ja JSON-tiedosto, johon se on tallennettu: `--pipeline "PL_RECON_INPUT.json"`. Komennon suoritettua putki ilmestyy lähes välittömästi Azure Data Factoryn puolelle ja se näkyy jälleen visuaalisena, kuten Kuvassa 20. Kun putki käynnistetään, tehtävät suorituvat Kuvan 22 sekvenssikaavion mukaisesti.



Kuva 22. ADF-dataputken aktiviteettien automaattinen suoritus.

Tehtävien suorittaminen tässä tapauksessa alkaa siis ihmisen aloitteesta. Ensimmäisenä tehtävänä suoriutuu Kuvassa 20 vasemmalla näkyvä Lookup-aktiviteetti, jossa haetaan parametrisoidulta kohdetietokannalta ADF_CONTROL_TABLE -taulussa sijaitsevat tiedot. Kuvassa 23 näkyy, että testausta varten kyseineen tauluun on lisätty vain yksi rivi, eli haluamme siirtää ”_B” -päätteisen tietokantanäkymän aineiston saman nimiseen kohdetietokannassa sijaitsevaan tauluun.



```
SELECT [SOURCE_SCHEMA]
      ,[SOURCE_TABLE]
      ,[TARGET_SCHEMA]
      ,[TARGET_TABLE]
      ,[LOAD_ALL_ROWS]
  FROM [dbo].[ADF_CONTROL_TABLE]
```

108 %

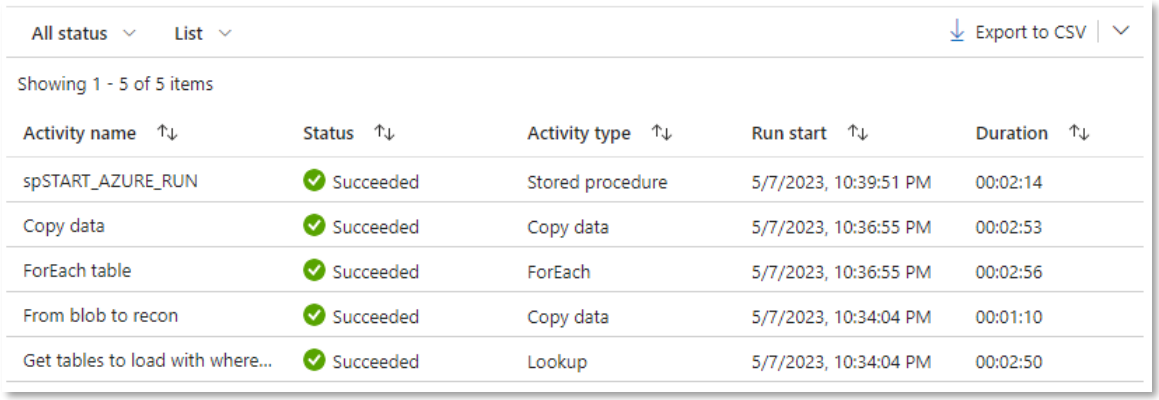
Results | Messages

	SOURCE_SCHEMA	SOURCE_TABLE	TARGET_SCHEMA	TARGET_TABLE	LOAD_ALL_ROWS
1	dm	vwFactECL_TEST_B	input	FactECL_TEST_B	1

Kuva 23. Komennolla `ahjo data` lisättiin kuvassa näkyvä rivi ADF_CONTROL_TABLE -tauluun.

Kun Lookup-aktiviteetti on hakenut ADF_CONTROL_TABLE -taulussa olevat tiedot, ForEach-aktiviteetti looppaa (engl. *loop*) sen sisällä olevaa Copy data -aktiviteettiä, joka etsii parametrisoidusta kohdekannassa sijaitsevassa ADF_CONTROL_TABLE -taulussa määritellyt objektit parametrisoidusta lähdetietokannasta ja siirtää niiden tiedot kohdetietokantaan. Samaan aikaan on käynnistynyt toinen Copy data -aktiviteetti, joka siirtää csv-tiedostojen datat myös samaan kohdetietokantaan. Kun molemmat Copy data -aktiviteetit ovat onnistuneesti suoriutuneet, Kuvassa 20 näkyvät vihreät nuolet osoittavat, että seuraavaksi kohdetietietokannassa käynnistetään `ui.spSTART_AZURE_RUN` -niminen proseduuri, joka käynnistää lähdedatojen täsmäytyksen.

Kun dataputki suoritetaan portaalin kautta, prosessin etenemistä pystytään sieltä hyvin seuraamaan. Mahdollisten virheidenkin selvittäminen on pyritty tekemään helpoksi, sillä Data Factory näyttää, missä aktiviteetissä virhe on sattunut ja mitkä olivat aktiviteetin sisään- ja ulostuloarvot. Dataputken suorittaminen onnistuneesti näyttää portaalista katsottuna Kuvan 24 mukaiselta.



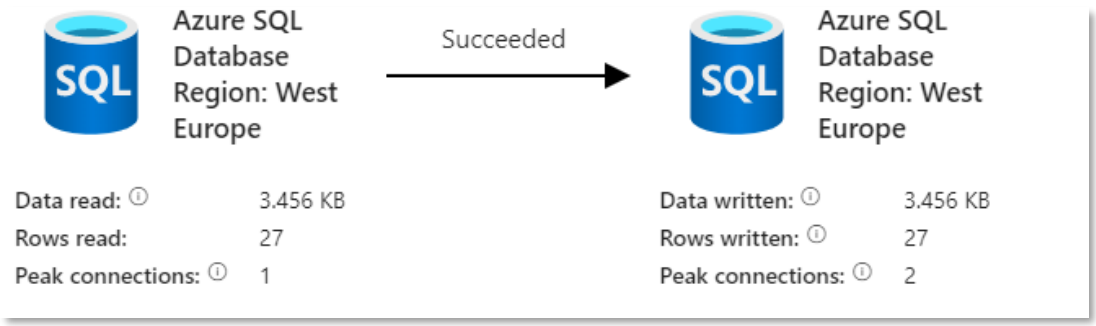
All status | List | Export to CSV

Showing 1 - 5 of 5 items

Activity name	Status	Activity type	Run start	Duration
spSTART_AZURE_RUN	Succeeded	Stored procedure	5/7/2023, 10:39:51 PM	00:02:14
Copy data	Succeeded	Copy data	5/7/2023, 10:36:55 PM	00:02:53
ForEach table	Succeeded	ForEach	5/7/2023, 10:36:55 PM	00:02:56
From blob to recon	Succeeded	Copy data	5/7/2023, 10:34:04 PM	00:01:10
Get tables to load with where...	Succeeded	Lookup	5/7/2023, 10:34:04 PM	00:02:50

Kuva 24. Näkymä, kun ADF-dataputki on onnistuneesti suoritettu.

Myös yksittäisen aktiviteetin tarkempia tietoja on mahdollista tarkastella painamalla aktiviteetin kohdalla olevaa ”Details”-painiketta. Tällöin aukeavat Kuvan 25- ja 26-mukaiset ikkunat, kun esimerkiksi Copy data -aktiviteetin lisätietoja putken suorittamisen jälkeen tarkastellaan.



Kuva 25. Copy data -aktiviteetin Details-ikkuna, joka näyttää tietoja suoritteesta.

Tämä oli Kuvassa 19 näkyvä vaihe, jossa lähdetietokannasta (CRDM) siirrettiin tiedot kohdetietokantaan (RECON). Vastaavasti Kuvassa 26 näkyy tiedot aktiviteetistä, jossa csv-aineisto siirrettiin kohdetietokantaan. Onnistuneen täsmäytyksen jälkeen Azuressa sijaitsevaan RECON-kantaan ilmestyy uudet taulut ja näkymät täsmäytyksestä, joista sen tuloksia ja loki-tietoja päästään tarkastelemaan. Näistä tiedot tulisivat näkyviin myös web-käyttöliittymään tai PowerBI-raportille.

Kuva 26. Vastaava Copy data -aktiviteetin Details-ikkuna.

Infrastruktuurin rakentaminen koodilla (*Infrastructure as Code*) muuttaa tapaa rakentaa ja ylläpitää IT-infrastruktuuria. Sen sijaan että ympäristöjä rakennetaan graafisten käyttöliittymien esimerkiksi Azuren portaalin kautta, kokonaisia ympäristöjä voidaan pystyttää ajamalla ympäristön kuvaustiedosto, kuten JSON-tiedosto tai ARM-template (*Azure Resource Manager*). Tämä nopeuttaa ja automatisoi infran pystyttämistä, mutta se myös muuttaa IT-infran ylläpitoa, sillä vikaantunutta ympäristöä ei välttämättä tarvitse korjata, vaan se voidaan asentaa kokonaan uudelleen [22]. Esimerkiksi palvelimien konfiguraatiot voidaan määritellä etukäteen niin sanotulle templa-

telle ja tarpeen vaatiessa voidaan uusi palvelin nopeasti pystyttää templateen määritellyillä tiedoilla. Tämä myös vähentää inhimillisiä virheitä esimerkiksi palvelinkonfiguraatiota määriteltäessä.

IT-infraa voidaan pystyttää koodilla esimerkiksi myös Terraformin avulla, joka on HashiCorp-nimisen ohjelmistoyrityksen tuote. Terraform on avoimeen lähdekoodiin perustuva ohjelmistotyökalu, joka on tähän tarkoitukseen suunniteltu. Se mahdollistaa infrastruktuurin automatisoinnin sekä minkä tahansa pilven, datakeskuksen tai palvelun provisioinnin ja hallinnan [44].

Infran pystyttäminen koodilla myös mahdollistaa sen versionhallinnoinnin ja tarvittaessa ympäristön pysymisen aina vakiona. Lisäksi tämä tarkoittaa myös sitä, että infraa pystytään versioimaan ja versioiden välisiä muutoksia tarkastelemaan. Tarvittaessa on myös mahdollista siirtyä aikaisempiin versioihin.

6 Tulokset ja pohdinta

Tuotantotapauksessa migraatio ei suinkaan jäisi vain sen varsinaiseen suorittamiseen. Tärkeänä osana jälkitöihin kuuluu sovelluksen validoiminen ja monitorointi, jotta se toimii odotetusti myös uudessa ympäristössään. Lisäksi sovelluksen suorituskykyä ja siihen tehtyjä muutoksia tulee testata ja seurata varmistamiseksi, että sovellus toimii jatkossa edelleen odotetulla tavalla. Sovellus on tärkeää testata myös mahdollisten tietoturva-aukkojen varalta. [3, s. 75.]

Prosessissa eniten haasteita tuotti uuden pilvessä sijaitsevan tietokannan populointi. Ongelmina olivat muun muassa aikaisemmin mainitut kollaatio-ongelmat sekä Ahjon ja muiden ohjelmistokirjastojen versiot. Versio-ongelmat saatiin selätettyä luomalla Azuren virtuaalikoneeseen virtuaaliympäristö, joka sisältää juuri oikeat versioinnit. Tämä ei kuitenkaan ole välttämättä aivan yhtä yksinkertainen prosessi esimerkiksi asiakasympäristön virtuaalikoneella, sillä niillä voi usein olla palomuuripääsyt tiukkoja, eikä esimerkiksi Python-pakettien lataaminen tai päivittäminen perinteisesti komentoriviltä onnistu, jos yhteys julkiseen Internetiin on evätty. Yksi mahdollinen keino ratkaisuksi asiakasympäristöissä on luomalla aluksi paikallisella koneella ZIP-kansio, joka sisältää tarvittavat ympäristömuuttujat. Seuraavaksi kansio siirrettäisiin asiakasympäristön virtuaalikoneelle, jossa luodaan uusi virtuaaliympäristö, johon puretun ZIP-kansion sisältö asennettaisiin.

Alussa mainittiin hypoteesina olevan backend-koodin refaktorointi, jonka mahdollinen laajuus ei ollut tiedossa. Tämä kuitenkin osoittautui teknisesti oletettua suoraviivaisemmaksi ja helpommaksi vaiheeksi. Ainoastaan joitain alkuperäisiä toiminnallisuuksia jouduttiin korvaavaan Azuren tarjoamilla ominaisuuksilla, mikä monessa tapauksessa onkin järkevää pilvialustalle siirryttäessä. Tämä on kuitenkin positiivinen havainto sovelluksen pilvivalmiuden kannalta.

Koska refaktorointiosuuteen tarvitsi käyttää odotettua vähemmän vaivaa, oli työssä mahdollista kiinnittää huomiota myös muihin seikkoihin, joita ei oltu alun perin määritelty, mutta ovat kuitenkin olennaisia työn viitekehyksen puolesta. Näitä olivat muun muassa virtuaaliympäristöt, ADF-dataputkien versiointi ja automatisointi.

Prosessin aikana sain paremman käsityksen täsmäytystyökalusta ja sen toiminnasta, sillä projektin alkuvaiheisiin kuului siihen perehtyminen. Projektin myötä pääsi tutustumaan ja kokeilemaan Azuren palveluita organisaation kehitysympäristössä.

Kaiken kaikkiaan projekti oli mielenkiintoinen, hyödyllinen ja antoisa tekijälleen, sillä se mahdollisti käytännön kautta tutustumisen kaikkiin edellä mainittuihin osa-alueisiin. Työn tarkoituksena

on kuitenkin olla hyödyksi tulevissa asiakasprojekteissa nyt tiedostaen, että täsmäytyssovelluksen backend-logiikka saadaan minimaalisilla muutoksilla toiminaan myös pilvialustalla.

Projektin jatkokehityskohteina voisi olla vastaavanlainen projekti myös työkalun frontend-puolelle. Sen lisäksi voitaisiin suunnitella, saataisiinko sovelluksesta kokonaisuutena pilvinatiivimpaa versiota ja kuinka pystyttää sovellus tuotantoympäristön pilvialustalle mahdollisimman suoraviivaisesti. Voitaisiin myös arvioida, kuinka kaukana tämä testiversio olisi tuotantoversiosta.

Esittelin tutkimuksen yrityksen sisäisessä projekti- ja sprinttikatselmoinnissa, joissa käydään yleisesti läpi myös muita projekteja, kehityksiä tai tutkimuksia. Toimeksiantajan puolelta on tullut positiivista palautetta dokumentoinnin monipuolisuudesta, projektin ripeästä edistymisestä ja myös tavoitteiden saavuttamisesta. Lisäksi työstä on ollut jo tähän mennessä apua kahdessa eri asiakasprojektissa, joten työ on myös siinä mielessä täyttänyt tavoitteensa.

Lähteet

[1] ALM Partners Oy. Tietoa yrityksestä. [Internet] 2023 [viitattu 16.3.2023] Saatavilla: https://almpartners.fi/

[2] Pilvipalveluiden käyttö. FiCom [Internet] 2022 [viitattu 15.3.2023] Saatavilla: https://ficom.fi/ict-ala/tietopankki/internetpalvelut/pilvipalvelut/pilvipalvelujen-kaytto/#pilvipalvelujen-kaytto-yrityksissa

[3] Vamsikrishna B. Applied Research in Artificial Intelligence and Cloud Computing. Optimizing IT Modernization through Cloud Migration: Strategies for a Secure, Efficient and Cost-Effective Transition. University of south Australia. 2022;5(1) [viitattu 11.5.2023] Saatavilla: https://researchberg.com/index.php/araic/article/view/97

[4] Raportointiohjeet. Suomen Pankki [Internet] [viitattu 19.3.2023] Saatavilla: https://www.suomenpankki.fi/fi/Tilastot/raportointiohjeet/

[5] What is a data pipeline? Snowflake [Internet] [viitattu 23.3.2023] Saatavilla: https://www.snowflake.com/guides/data-pipeline

[6] AltexSoft Inc. Software R&D engineering. Data Pipeline: Components, Types, and Use Cases. [Internet] 2022 [viitattu 19.3.2023] Saatavilla: https://www.altexsoft.com/blog/data-pipeline-components-and-types/

[7] Hovi A., Hervonen H. & Koistinen H. 2009. Tietovarastot ja Business Intelligence. Jyväskylä: WSOYpro (Sanoma Pro) Saatavilla: ISBN 978-951-0-34792-8

[8] Korth H., Silberschatz A., Silberschatz S. Database System Concepts (6th Edition). McGraw Hill; 2010;1(27) [E-kirja] Saatavilla: ISBN 978-0-07-352332-3

[9] Rick F. van der Lans. Data Virtualization for Business Intelligence Systems: Business Intelligence and Data Warehousing [Internet]. Elsevier Inc. 2012 (Sivut 27–57) Saatavilla: doi: https://doi.org/10.1016/B978-0-12-394425-2.00002-2

[10] Michiels H. Using a Persistent Staging Area: What, Why, and How [Internet]. 2017;2(18) [viitattu 24.3.2023] Saatavilla: https://www.hansmichiels.com/2017/02/18/using-a-persistent-staging-area-what-why-and-how/

[11] Kimball R., Ross M., The Data Warehouse Toolkit. The Definitive Guide to Dimensional Modeling (3rd Edition). Wiley; 2013 [E-kirja] Saatavilla: ISBN: 978-1-118-53080-1

[12] What Is a Data Mart? Amazon Web Services (AWS) [Internet] [viitattu 29.3.2023] Saatavilla: https://aws.amazon.com/what-is/data-mart/

[13] Ben Haddou M. Advantages and Disadvantages of a Data Mart. [Internet] 2022 [viitattu 29.3.2023] Saatavilla: http://mbenhaddou.com/2020/01/16/advantages-and-disadvantages-of-a-data-mart/

[14] Collier M., Shahan R. Microsoft Azure Essentials: Fundamentals of Azure. Microsoft Corporation. [E-kirja] 2015 Saatavilla: ISBN: 978-0-7356-9722-5

[15] NIST: Mell P., Grance T. The NIST Definition of Cloud Computing. U.S. Deparment of Commerce. 2011. Saatavilla: doi: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

[16] Suosituksia ulkoistamisesta pilvipalveluihin. EBA (European Banking Authory) [Internet] 2018;3(28) [viitattu 10.4.2023] Saatavilla: https://www.eba.europa.eu/sites/default/documents/files/documents/10180/2170125/de4c5d62-01f9-4441-9eb3-c8561eba7cc1/Recommendations%20on%20Cloud%20Outsourcing%20%28EBA-Rec-2017-03%29_FI.pdf?retry=1

[17] Fonseca N., Boutaba R. Cloud Services, Networking, and Management. Wiley [Internet] 2015;4(3) Saatavilla: doi: http://dx.doi.org/10.1002/9781119042655

[18] Sroczkowski P. Cloud: IaaS vs PaaS vs SaaS vs DaaS vs FaaS vs DBaaS. Brainhub [Internet] [viitattu 29.3.2023] Saatavilla: https://brainhub.eu/library/cloud-architecture-saas-faas-xaas

[19] Lanfear T., Hitchcock A.M., Berry D. Shared Responsibility in the Cloud. Microsoft Learn [Internet] 2022;6(12) [viitattu 29.3.2023] Saatavilla: https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility

[20] What are public, private and hybrid clouds? Microsoft [Internet] [viitattu 4.4.2023] Saatavilla: https://azure.microsoft.com/en-in/resources/cloud-computing-dictionary/what-are-private-public-hybrid-clouds/#overview

[21] Chand M. Top 10 Cloud Service Providers In 2023. C# Corner. [Internet] 2023;2(26) [viitattu 4.4.2023] Saatavilla: https://www.c-sharpcorner.com/article/top-10-cloud-service-providers/

[22] Wallenius N. Opas pilvistrategian laatimiseksi. Wallenius Consulting Oy; 2019 [viitattu 11.5.2023] Saatavilla: https://niklaswallenius.fi/opas-pilvistrategia/

[23] Hänninen T. Pilvipalveluiden käyttö ja mahdollisuudet. Azure pilvialustan esittely ja demo. Microsoft (Suomi). Tapahtuma 21.3.2023, Kajaanin ammattikorkeakoulu

[24] What is Azure? Microsoft [Internet] [viitattu 19.3.2023] Saatavilla: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/

[25] Ohjeet ulkoistamisesta pilvipalvelujen tarjoajille. ESMA (European Securities and Markets Authority) [Internet] 2021;5(10) [viitattu 10.4.2023] Saatavilla: https://www.finanssivalvonta.fi/globalassets/fi/saantely/maarayskokoelma/2021/04_2021/esma_cloud_guidelines_fi.pdf

[26] Abandy R. The History of Microsoft Azure. [Internet] 2022 [viitattu 25.3.2023] Saatavilla: https://techcommunity.microsoft.com/t5/educator-developer-blog/the-history-of-microsoft-azure/ba-p/3574204

[27] What is ELT (Extract, Load, Transform)? IBM [Internet] [viitattu 11.5.2023] Saatavilla: https://www.ibm.com/topics/elt

[28] Ekuan M. Automated enterprise BI. Microsoft Learn: Architectures. [Internet] 2022;7(28) [viitattu 10.4.2023] Saatavilla: https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/data/enterprise-bi-adf

[29] Kuva Microsoft Azuren palveluportaalista. Microsoft Azure [Internet] 2023 [viitattu 10.4.2023] Saatavilla: https://azure.microsoft.com/en-us/get-started/azure-portal ¨

[30] Bradish D. What is the Azure CLI? Microsoft Learn [Internet] 2021;9(21) [viitattu 10.4.2023] Saatavilla: https://learn.microsoft.com/en-us/cli/azure/what-is-azure-cli

[31] Shyamsundar A. Elastic pools help you manage and scale multiple databases in Azure SQL Database. Microsoft Learn [Internet] 2023;9(2) [viitattu 10.4.2023] Saatavilla: https://learn.microsoft.com/en-us/azure/azure-sql/database/elastic-pool-overview?view=azuresql

[32] Neugebauer N. What is Azure SQL Managed Instance? Microsoft Learn [Internet] 2023;6(2) [viitattu 10.4.2023] Saatavilla: https://learn.microsoft.com/en-us/azure/azure-sql/managed-instance/sql-managed-instance-paas-overview?view=azuresql

[33] Robles C. Migration overview: SQL Server to Azure SQL Managed Instance**.** Microsoft Learn [Internet] 2023;9(1) [viitattu 10.4.2023] Saatavilla: https://learn.microsoft.com/en-us/azure/azure-sql/migration-guides/managed-instance/sql-server-to-managed-instance-overview?view=azuresql

[34] Kuva. Microsoft Learn [Internet] 2022;4(11) [viitattu 11.4.2023] Saatavilla: https://learn.microsoft.com/en-us/azure/azure-sql/azure-sql-iaas-vs-paas-what-is-overview?view=azuresql

[35] Furman D. DTU-based purchasing model overview. Microsoft Learn [Internet] 2023;3(4) [viitattu 13.4.2023] Saatavilla: https://learn.microsoft.com/en-us/azure/azure-sql/database/service-tiers-dtu?view=azuresql

[36] Assaf W. Compare vCore and DTU-based purchasing models of Azure SQL Database. Microsoft Learn [Internet] 2023;3(3) [viitattu 13.4.2023] Saatavilla: https://learn.microsoft.com/en-us/azure/azure-sql/database/purchasing-models?view=azuresql

[37] Swoyer S. Migrating Applications to the Cloud. O'Reilly Media, Inc; 2021. [E-kirja] Saatavilla: ISBN: 9781098102791

[38] Tutkimus: Turvallisesti kohti joustavaa hybridipilveä. Elisa Oyj. [Internet] 2022 [viitattu 16.4.2023] Saatavilla: Tutkimus ladattu osoitteesta: https://yrityksille.elisa.fi/konesali-ja-kapasiteettipalvelut

[39] Orban S. 6 Strategies for Migrating Applications to the Cloud. AWS Amazon [Internet] 2016;11(1) [viitattu 17.4.2023] Saatavilla: https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/

[40] What is Cloud Migration? VMware [Internet] [viitattu 17.4.2023] Saatavilla: https://www.vmware.com/topics/glossary/content/cloud-migration.html

[41] Serverless compute tier for Azure SQL Database. Microsoft Learn [Internet] 2023;4(17) [viitattu 18.4.2023] Saatavilla: https://learn.microsoft.com/en-us/azure/azure-sql/database/serverless-tier-overview?view=azuresql&tabs=general-purpose

[42] Ahjo. Database deployment framework. Python Package Index (PyPI) [Internet] 2023;2(28) [viitattu 15.4.2023] Saatavilla: https://pypi.org/project/ahjo/

[43] Assaf W. Collation and Unicode support. Microsoft Learn [Internet] 2023;9(3) [viitattu 65.2023] Saatavilla: https://learn.microsoft.com/en-us/sql/relational-databases/collations/collation-and-unicode-support?view=sql-server-ver16

[44] Terraform-etusivu [Internet] [viitattu 19.3.2023] Saatavilla: https://www.terraform.io/