HAMK
Hämeen ammatti-
korkeakoulu

INFRAKOODIN HYÖDYNTÄMINEN PILVIPALVELUYMPÄRISTÖJEN LUOMISESSA JA MUUTOKSISSA

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus

Syksy 2023

Toni Pärnänen

HAMK
Hämeen ammatti-
korkeakoulu

Tietojenkäsittelyn koulutus Tiivistelmä
Tekijä Toni Pärnänen Vuosi 2023
Työn nimi Infrakoodin hyödyntäminen pilvipalveluympäristöjen luomisessa ja muutoksissa
Ohjaaja Esa Huiskonen

Opinnäytetyön tavoitteena oli perehdyttää lukija pilvipalveluiden ja Infrastructure as Coden (IaC) perusteisiin ja opastaa lukijaa tekemään ensimmäinen pilvi-infrastruktuuri pilvialustalle IaC-työkalun avulla. Opinnäytetyössä myös opastetaan tarvittavien työkalujen asennus ja käyttöönotto. Tavoitteena oli myös tarkastella, mitä hyötyjä ja mitä mahdollisia ongelmia IaC:n käytössä on.

Opinnäytetyö on pääosin toiminnallinen ja sen tietopohja koostuu pilvipalveluiden ja Infrastructure as Coden ominaisuuksista ja käsitteiden avaamisesta. Selitettyjen kokonaisuuksien pohjalta ja niiden esittelyn jälkeen opinnäytetyössä esitellään, kuinka ensimmäinen IaC-projekti saadaan vietyä pilvipalvelualustalle. Tämä auttaa lukijaa konkretian kautta ymmärtämään paremmin kokonaisuutta, josta opinnäytetyö kertoo.

Johtopäätöksenä voidaan todeta, että vaikka Infrastructure as Codea käytettäessä pilvipalveluympäristöjen luonti ja ylläpito nopeutuu ja helpottuu, on myös tilanteita ja infrakokonaisuuksia, joissa sitä ei välttämättä ole järkevää hyödyntää. IaC-palveluita käyttää kuitenkin yhä useampi yritys ja onkin yritys- ja tapauskohtaista kannattaako IaC-palveluita hyödyntää ja miten laajalla skaalalla.

Avainsanat Pilvipalvelut, IaC, Azure, Terraform
Sivut 29 sivua ja liitteitä 1 sivu

HAMK
Hämeen ammatti-
korkeakoulu

Degree Programme in Business Information Technology — Abstract
Author Toni Pärnänen — Year 2023
Subject Utilizing infracode in the creation and changes of cloud service environments
Supervisors Esa Huiskonen

The aim of the thesis was to introduce the reader to the basics of cloud services and Infrastructure as Code (IaC) and to guide the reader to create the first cloud infrastructure on a cloud platform using an IaC tool. The thesis also guides the installation and deployment of the necessary tools. The aim was also to examine the benefits and possible problems of using IaC.

The thesis is mainly functional and its knowledge base consists of the features and concepts of cloud services and Infrastructure as Code. Based on the explained entities and their presentation, the thesis presents how the first IaC project can be transferred to the cloud service platform. This helps the reader to better understand this topic through concrete examples.

The conclusion is that although Infrastructure as Code speeds up and simplifies the creation and maintenance of cloud service environments, there are also situations and infrastructure entities where it may not make sense to use it. However, more and more companies use IaC services and it is company- and case-specific whether it is worthwhile to use IaC services and on what scale.

Keywords Cloud services, IaC, Azure, Terraform
Pages 29 pages and appendices 1 page

Sanasto

Pilvi, cloud	tarkoittaa verkon välityksellä tarjottavaa dataa, ohjelmistoa tai muuta ratkaisua
Azure	Microsoftin pilvipalvelualusta
Terraform	Hashicorpin ylläpitämä pilviresurssien hallinnointityökalu
IaC	Infrastructure as a Code. Pilvi-infrastruktuurin luominen koodia kirjoittamalla

Sisällys

Kuvat ja ohjelmakoodit

Liitteet

1 Johdanto

Pilvipalvelut ovat jo pitkälti yli vuosikymmenen olleet saatavilla yrityksille, ja tulevaisuudessa yhä useampi yritys suunnittelee siirtävänsä palveluitaan pilveen. Pilvipalveluiden avulla ohjelmiston skaalautuvuus ja hallinta on helpompaa kuin yrityksen omassa konesalissa, eli on-premises-ympäristössä. Pilvipalveluiden kehittyessä myös oheis- ja ylläpitopalvelut ovat kehittyneet ja nykyään on mahdollista avata ja muokata pilvi-infrastruktuuria kirjoittamalla koodia erilaisissa siihen tarkoitukseen suunnitelluissa ympäristöissä. Erityisesti toisteisessa palveluiden avauksessa infrakoodin käyttö on merkittävä etu ja säästää työtä ja aikaa. Tässä opinnäytetyössä tutkin kuinka Infrastructure as a Code-työkalu Terraform otetaan käyttöön ja saadaan toimimaan Microsoftin Azure -pilvialustan kanssa, mitä hyötyjä sen käytöstä on ja mitkä ovat sen rajoitteet ja riskit. Valitsin Terraformin, koska se on käytetyin deklaratiivinen IaC-alusta tällä hetkellä.

Tavoitteena on käydä läpi, kuinka IaC, tässä tapauksessa Terraform, auttaa pilvi-infrastruktuurin luomisessa, muokkauksessa ja ylläpidossa. Tarkoitus on myös kuvata Terraformin ja muiden tarvittavien ohjelmien käyttöönotto ja opastaa Terraformilla toteutettu Azure-palveluiden avaaminen. Vaikka IaC-järjestelmiä on olemassa useita, käyn tässä työssä läpi asioita vain suosituimman palvelun ja Microsoftin Azuren kautta Windows-käyttöjärjestelmässä

Opinnäytetyössä käsittelen seuraavia tutkimuskysymyksiä:
1. Miten Terraform otetaan käyttöön?
2. Millaisissa tilanteissa IaC:a kannattaa käyttää?
3. Mitä haasteita IaC:n käytössä voi olla?

2 Pilvipalvelut

Pilvipalvelu ja pilviteknologia tarkoittavat verkossa olevia it-palveluita. Pilvessä oleva palvelu tai tiedosto sijaitsee palveluntarjoajan palvelimella, eikä asiakkaan tai organisaation omalla palvelimella (on-premises). Julkipilveä käyttämällä vuokrataan kapasiteettia tai resursseja palveluntarjoajalta, esimerkiksi Microsoftin Azuresta. Pilvessä tarjolla olevia resursseja ovat esimerkiksi tekoäly, tallennuspalvelut, palvelimet ja analytiikkaratkaisut ja tietokannat. Myös verkon yli käytettäviä sovelluksia on valtava määrä. (Vento, 2021)

2.1 Pilvi-infrastruktuuri

Pilvi-infrastruktuuri tarkoittaa kaikkia niitä resursseja, joita pilvipalvelussa hyödynnetään, kuten tallennustilaa ja palvelimia. Kun infrastruktuuri on suunniteltu hyvin, se mahdollistaa pilvipalveluiden kehityksen ja oikean toiminnan. Kaikki pilvikehittäminen on infrapohjaista, eikä pilvessä voi luoda uusia kyvykkyyksiä, jos infra ei ole kunnossa. (Liimatta, 2021)

2.1.1 Yksityinen pilvi

Yksityinen pilvipalvelu on tarkoitettu vain yhdelle organisaatiolle ja se sijaitsee yksityisessä verkossa. Rahoitusalan toimijat ovat julkisen sektorin lisäksi esimerkkejä yksityisen pilven käytöstä. (Liimatta, 2021). Kuitenkin nyt Microsoftin laajentaessa Suomeen (Microsoft, 2022a), OP ja Säästöpankki aikovat siirtää palveluitaan Azureen (Microsoft, 2023b; Säästöpankki, 2023). Yksityisen pilven ylläpito voidaan ulkoistaa tai hoitaa itse. Virtuaalinen yksityinen pilvipalvelu on eriytetty ympäristö, joka voidaan yhdistää yrityskohtaiseen yksityiseen verkkoon.

2.1.2 Julkinen pilvi

Julkisessa pilvessä palveluita käytetään verkon yli ja näiden palveluiden hallinnoija on pilvipalvelun toimittaja, esimerkiksi Microsoftin Azure. Muita suuria alustoja ovat Google Cloud Platforms (GCP) ja Amazon Web Services (AWS). Julkisesta pilvestä on saatavilla ratkaisuja kaiken kokoisten yritysten käyttöön maailmanlaajuisesti. (Liimatta, 2021)

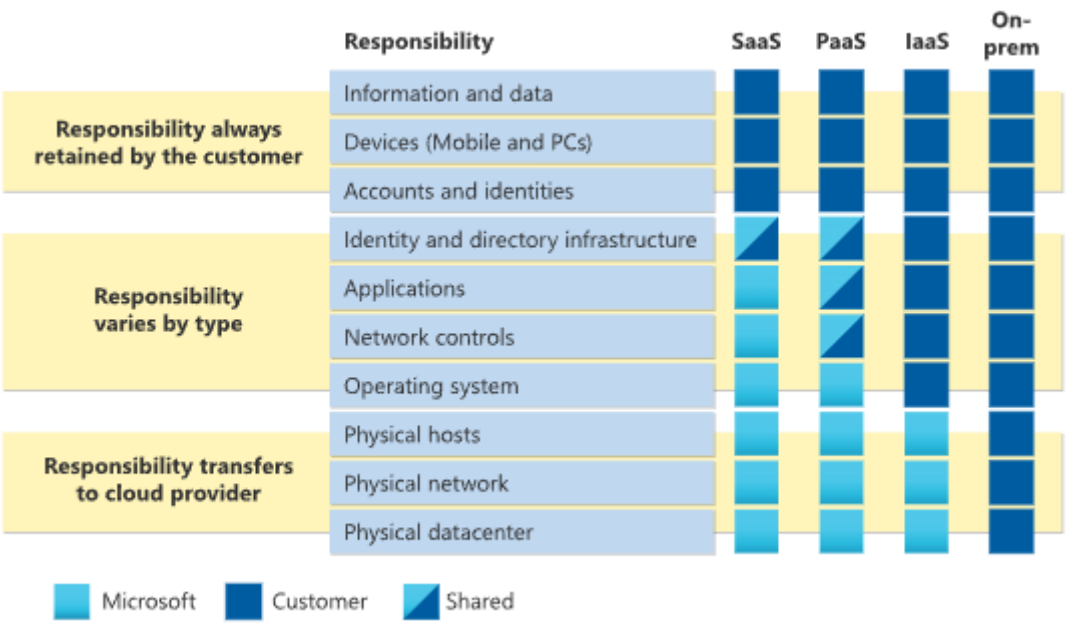
2.1.3 Hybridipilvi

Hybridipilvellä tarkoitetaan toteutusta, jossa käytetään sekä julkista että yksityistä pilveä kuitenkin erillisinä toteutuksina. Toteutusta käytetään usein silloin kun järjestelmässä on tietoa, jota ei saa siirtää ulkopuoliselle palvelimelle. Hybridiä voidaan käyttää myös siirtymätilanteissa, kun palveluita siirretään pilveen pala kerrallaan. Jatkossa myös eri pilvipalveluiden yhdistelmät yleistyvät, kun halutaan käyttää parhaat osat eri julkipilvistä. (Floor, 2023)

2.2 Pilvipalveluiden luokittelu

Pilvipalveluiden luokittelun yhteydessä voidaan puhua hankintamalleista. Eri hankintamallit määrittelevät vastuunjaon asiakkaan ja palveluntarjoajan välillä. Pilvipalvelut jaetaan kolmeen ryhmään vastuunjaon perusteella. Niitä ovat IaaS, PaaS ja SaaS. (Floor, 2023) Kuvassa 1 näkyy hankintamallien vastuunjako palveluntarjoajan ja asiakkaan kesken.

Kuva 1 Vastuunjako (Microsoft, 2022b)



2.2.1 IaaS

Iaas eli Infrastructure as a Service on verrattavissa yrityksen omassa konesalissa ajettavaan palvelinratkaisuun, mutta IaaS:ssa lisäkapasiteettia saa hankittua yksinkertaisesti ja palvelimien ylläpidosta vastaa pilvipalvelun tarjoaja. (Floor, 2023)

IaaS siis tarjoaa netin kautta pääsyn resursseihin, kuten palvelimiin, verkkoihin ja tallennusjärjestelmiin. IaaS:ssa tarvitsee maksaa vain käytössä olevista resursseista, joten oman konesalin rakentamiskustannuksilta on mahdollista näin välttyä. Oman konesalin resurssit täytyy myös mitoittaa suurimpien käyttöpiikkien mukaan, joten turhaa kapasiteettia on mahdollista välttää pilvipalvelua käyttämällä. (IBM, 2023a)

IaaS:n käyttäjää laskutetaan vain käyttämistään resursseista ja näin vältytään käyttämättömänä seisovan oman konesalin resurssien kustannuksilta. IaaS:n valtteja ovat myös nopeus ja saatavuus. Tarvittavat resurssit ovat heti käytössä ja saatavilla eri alueiden myötä ympäri maailman. Esimerkiksi eurooppalaiset loppukäyttäjät voidaan ohjata pilvipalvelun tarjoajan Euroopassa sijaitsevaan pilveen tai yhdysvaltalaiset käyttäjät siellä sijaitsevaan pilveen. (IBM, 2023c)

IaaS:ssa asiakkaalla on eniten vastuuta itsellään verrattaessa PaaS:iin tai SaaS:iin. Kymmenisen vuotta sitten IaaS oli suosituin hankintamalli, mutta nykyään muiden mallien käyttö kasvaa IaaS:a nopeammin. (IBM, 2023a)

2.2.2 PaaS

PaaS, eli Platform as a Servicessä palveluntarjoajalta voi ostaa palveluna myytäviä alustakomponentteja. Sovelluskehitys alustakomponenttien päälle on nopeaa ja helppoa, koska palveluntarjoaja vastaa esimerkiksi palvelun päivityksistä. Laskentakapasiteetin lisääminen käyttökuorman kasvaessa hoituu alustan puolesta. PaaS-palveluita käytettäessä ollaan riippuvaisia alustan toimittajasta. Kustannustehokkuus PaaS-palveluissa on parempi, kuin vastaavasti itse toteutetussa kokonaisuudessa. (Floor, 2023)

PaaS sisältää kaikki yleisimmät palvelut, joita kehittäjät tarvitsevat, sisältäen mm. infrastruktuurin ja ohjelmistot. PaaS:ssa pilvialustan tarjoajan vastuulla on enemmän asioita kuin edellä mainitussa IaaS:ssa. Siinä palveluntarjoajan vastuulla on palvelimet, verkot, tallennustila, tietokannat, sekä käyttöjärjestelmä. Monesti PaaS:ssa on käytössä ”pay as you

go” hinnoittelu, joten asiakkaat maksavat vain sillä hetkellä tarvitsemistaan palveluista. Näin esimerkiksi testaus on halvempaa kun voidaan käyttää testaukseen vain hetkellisesti resursseja ja testauksen ollessa ohi, resurssit voidaan sulkea. (IBM, 2023d)

PaaS:n avulla projektit saadaan nopeammin kehitykseen ja markkinoille asiakkaiden saataville. Yritys säästää aikaa, kun ohjelmistot ja laitteet voidaan avata suoraan pilvestä käyttöön heti oman konesalin rakentamisen ja ohjelmistojen asennuksen sijaan. PaaS:ssa voi myös helposti kokeilla erilaisia muutoksia pienillä kustannuksilla. Käyttöjärjestelmää tai muuta ohjelmistoa voi vaihtaa ja kokeilla ilman, että tarvitsee esimerkiksi ostaa uutta infraa omaan konesaliin. Kokonaisuus on helppoa skaalata vastaamaan sen hetkisisä tarpeita. Jos tulee korkeita käyttöpiikkejä, voidaan PaaS-alusta ohjelmoida vastaamaan näitä nousuja automaattisesti ja vähentämään resursseja sitten kun käyttö pienenee. PaaS mahdollistaa myös kehittäjien pääsyn työkaluihin internetin yli mistä vain. Skaalautuvuuden lisäksi kustannussäästöjä voi tulla esimerkiksi lisensointikustannusten laskemisen myötä. (IBM, 2023d)

2.2.3 SaaS

SaaS, eli Software as a Service tarkoittaa internetin kautta palveluna myytävää sovellusta. Palveluntarjoaja hoitaa kaiken infrastruktuurin ja ohjelmiston käyttöön, ylläpitoon ja hallintaan liittyvästä. Ohjelmistopäivitykset, korjauspaketit ja tietoturvapäivitykset sekä käyttökuorman jako ja varmuuskopiointi ovat asioita, joita käyttäjän ei tarvitse pohtia, vaan ne hoitaa palveluntarjoaja. (IBM, 2023b)

SaaS on nykyään suosituin julkisen pilven hankintamalli. Sitä käytetään sekä pienissä, että suurissa yrityksissä helppouden takia. Esimerkiksi Microsoft 365, eli Microsoftin toimistosovelluspaketti on nykyään SaaS-palvelu, joka toimii pilvessä omalle koneelle asennuksen sijaan. (IBM, 2023b)

SaaS:n avulla palvelut ovat heti käyttäjän käytettävissä, eikä asennuksia jokaisen käyttäjän koneelle tarvita. Palvelun päivitykset asennetaan automaattisesti, eikä niitä tarvitse erikseen asentaa jokaiselle käyttäjälle, joten kaikki uusimmat ominaisuudet ovat kaikilla käyttäjillä heti käytössä. Myös SaaS:n etuihin voidaan laskea skaalautuvuus, koska lisäkapasiteettia voi ostaa tai sitä vähentää päivittämällä palvelutasoa. Skaalautuvuuden lisäksi kustannussäästöjä voidaan saada siitä, että IT-henkilöstön ei tarvitse päivittää jokaista konetta erikseen sekä omaa infrastruktuuria ei tarvitse rakentaa omaan konesaliin.

2.3 Pilvipalveluiden etuja

Pilvipalvelut ovat skaalautuvia, eli tarvitsemansa resurssit voi sopeuttaa omiin tarpeisiin, eikä asiakkaan tarvitse huolehtia konesalien riittävän suuresta kapasiteetista tai ylläpidosta. Pilvipalvelussa voi käyttää vain juuri sillä hetkellä tarvitsemiaan resursseja. Tästä tulee kustannussäästöjä, koska konesalissa ei ole käyttämätöntä ylikapasiteettia, vaan pelkästään tarvittava määrä. Oman konesalin mitoitus täytyy tehdä huippukuormien mukaan, joten käyttämätöntä kapasiteettia jää konesaliin helposti. Esimerkiksi jos tarvitaan lisää prosessoritehoa, sitä voi lisätä helposti esimerkiksi pilvipalvelun portaalin kautta parilla klikkauksella. Tai jos verkkoliikennettä ei olekaan niin paljon kuin ennen, pilvestä voi vähentää verkkoresursseja. Skaalautuvuudella voidaan siis taata, että resursseja on aina sopiva määrä käytössä. Jos nettisivujen käyttäjämäärä kasvaa odotetusti tai yllättäen, pilvipalvelussa saadaan lisäresursseja käyttöön, jotta palvelu kestää kasvavia käyttäjämääriä. Resurssien mukautus voidaan myös automatisoida, joten asiakkaan ei tarvitse itse valvoa ja huolehtia, että sopivat resurssit ovat käytössä. (Eskola, 2023). (Wallenius, 2019)

Pilvipalvelussa voi kokeilla palvelun toimivuutta helposti. Jos palvelu toimii kuten pitää, voi resursseja lisätä tarvittavan määrän tai jos palvelu ei toimikaan vielä odotetulla tavalla, voidaan resurssit lopettaa ja palvelu sulkea, jolloin siitä ei tule enää kustannuksia. Näin palvelua voi riskittä testata, eikä esimerkiksi omaan konesaliin jää tällöin käyttämättömiä resursseja. (Floor, 2023)

Pilvipalvelussa resurssien käyttöönotto onnistuu pienessä ajassa. Tarvittavat resurssit saa käyttöön välittömästi, eikä tarvita on-premises-ympäristössä tehtäviä hankintoja ja asennuksia, jotka veisivät aikaa jopa viikkoja. Käyttöönoton nopeus voi vaikuttaa jopa yrityksen menestykseen positiivisesti, jos ehditään toteuttaa palvelu nopeammin kilpailijoihin verrattuna. Verrattaessa pilvipalvelun konesalia ja perinteistä omaa fyysistä konesalia, käyttöönoton nopeus tuo siis suuria hyötyjä. (Wallenius, 2019)

Ylläpito on pilvipalvelussa helpompaa kuin omassa ympäristössä. Jos esimerkiksi tarvitaan tietokantoja, vaatii tietokannan ylläpito omassa konesalissa kuitenkin kaikkien muidenkin konesalin järjestelmien päivittämistä ja ylläpitoa. Pilvipalvelua käyttämällä voi avata vain tarvitsemansa tietokantapalvelun, eikä asiakkaan tarvitse huolehtia muusta ylläpidosta. Asiakkaan aika ja työtunnit ovat näin käytettävissä olennaisempaan. (Wallenius, 2019)

2.4 Azure-pilvipalvelualusta

Azure on Microsoftin julkinen pilvialusta, joka julkaistiin alun perin vuonna 2008. Sen markkinaosuus on tällä hetkellä sen verran suuri, että se pääsee suosituimpien pilvialustojen listalla toiselle sijalle heti Amazon Web Servicesin jälkeen. (Statista, 2023). Fortune 500-yrityksistä 95 prosenttia käyttää pilvialustanaan Azurea. (Microsoft, 2023c). Kaupallisesti Azure julkaistiin 2010. (Microsoft Techcommunity, 2022)

Azure tarjoaa valikoiman pilvipalveluita esimerkiksi laskennan, tallennuksen, analytiikan ja verkkojen osalta. Käyttäjät voivat valita näistä palveluista tai kehittää sovelluksia itse. Azurea käyttää monen eri alan yritykset, mukaan lukien esimerkiksi rahoitus- ja verkkokauppayritykset. Avoimen lähdekoodin ohjelmistojen ja Azuren yhteensopivuus auttaa laajentamaan Azuren käytettävyyttä. Azuressa on käytössä ”Pay As You Go”-laskutus, eli siinä maksetaan vain käytetyistä palveluista. (Bigelow, 2023)

3 IaC – Infrastructure as a Code

IaC, eli Infrastructure as a Code tarkoittaa pilvi-infrastruktuurin määrittelemistä koodilla. Infrakoodi sisältää projektissa käytettävät resurssit, eli esimerkiksi palvelimet, verkkoyhteydet ja käyttöoikeudet. Määrittelyt voidaan tallentaa GitHubiin tai muuhun versionhallintajärjestelmään, josta koodi on sitten kaikkien projektin osallisten saatavilla. IaC:ssa dokumentointi ja ympäristön nykytila on helposti saatavilla. Myös muutoksia voidaan helposti jäljittää. Testaaminen on myös helppoa, koska on helppo palata alkuperäiseen tilanteeseen, ellei testatut muutokset olekaan halutulla tavalla toimivia. IaC toimii ilman versionhallintaakin suoraan esim. Azuren kanssa mutta versionhallinnan mukana ollessa voidaan luoda projektille hyväksyntäprosessi ja nähdään kuka mitkäkin muutokset on tehnyt. IaC-kirjastoista löytyy valmiita pohjia moniin erilaisiin tarkoituksiin. Näitä pohjia voi käyttää suoraan tai muokata paremmin omiin käyttötarkoituksiin sopiviksi. Yritykset voivat myös itse rakentaa kirjastoja omiin projekteihin sopivista määrittelyistä. (Webscale, 2023)

Jos käytössä on versionhallinta, voidaan samaa infrakoodia käyttää sekä kehitys- että tuotantoympäristössä. Myös testaukseen kannattaa käyttää samaa koodia. Jos testaus tehdään onnistuneesti kerran, voidaan testausympäristö poistaa ja ajaa myöhemmin käyttöön samalla infrakoodilla parilla napin painalluksella. (Webscale, 2023)

Kun infraa ei enää tarvita, voidaan koko asetelma poistaa kerralla. IaC osaa poistaa kaiken oikeassa järjestyksessä ja ymmärtää resurssien välisten riippuvuuksien merkityksen. Tällöin pilveen ei jää resursseja, jotka manuaalisesti olisivat kenties jääneet poistamatta. Tästä tulee säästöjä kustannuksiin, koska pilveen ei jää käyttämättömiä, maksullisia resursseja (Webscale, 2023)

Infrastructure as a Code-työkaluja on olemassa niin pilvialustakohtaisia kuin useissa alustoissa toimivia. Suurimmilla pilvialustoilla on omat IaC-työkalunsa. Esimerkiksi Azuressa Bicep. Tunnetuin IaC-työkalu lienee kaikissa suurimmissa pilvialustoissa toimiva Hashicorpin hallinnoima Terraform. (Webscale, 2023)

3.1 Deklaratiivinen ja imperatiivinen IaC

On olemassa kahdenlaisia IaC-toteutustapoja. Joihinkin toteutuksiin sopii paremmin deklaratiivinen ja toisiin imperatiivinen tapa. Jotkin työkalut toimivat jommallakummalla tavalla, mutta useat IaC-työkalut tukevat molempia. Esimerkiksi infraan tehtävien muutosten

kannalta deklaratiivinen malli on helpompi käyttää, koska halutun lopputuloksen voi kirjoittaa koodiin ja työkalu osaa toteuttaa lisäykset, poistot ja muutokset itsekseen, eikä kehittäjän tarvitse pohtia resurssien välisiä riippuvuuksia, eikä miettiä uudelleen toteutusjärjestystä. (HPE, 2023)

3.1.1 Deklaratiivinen IaC

Deklaratiivisessa, eli toiminnallisessa lähestymistavassa työkalulle kuvaillaan haluttu lopputulos määrittelemättä kuitenkaan välivaiheita, kuinka päästä lopputulokseen. Työkalu osaa toteuttaa halutunlaisen lopputuloksen oikeassa järjestyksessä ja riippuvuuksista itse huolehtien. Tämän lähestymistavan avulla pystyt siis määrittämään haluamasi resurssit, mutta myös haluamasi konfiguraatiot. Kun muutoksia tehdään, IaC-ohjelmisto ohjelmisto valmistelee automaattisesti halutun infrastruktuurin ja osaa soveltaa muutoksia automaattisesti. (HPE, 2023)

3.1.2 Imperatiivinen IaC

Imperatiivisessa lähestymistavassa täytyy lopputuloksen sijaan määritellä, kuinka pilvi-infra tulee toteuttaa ja missä järjestyksessä, eli ohjeistaa IaC-ohjelmistoa askel askeleelta pääsemään haluttuun lopputulokseen. Komennot tulee suorittaa oikeassa järjestyksessä, askel kerrallaan. Jos infrastruktuuriin tarvitsee tehdä muutoksia, pitää imperatiivisessa lähestymistavassa ottaa huomioon paljon enemmän asioita kuin deklaratiivisessa toteutuksessa. (HPE, 2023)

4 IaC:n haasteet ja riskit

Vaikka IaC tuokin mukanaan paljon hyötyjä on sillä myös haasteensa. Infrakoodin kirjoittaminen voi olla suurissa projekteissa haastavaa, sillä on olemassa käytäntöjä ja standardejakin, joita on noudatettava infrakoodia tehtäessä ja tekijöiden voi olla hankala ymmärtää ja ylläpitää IaC-projekteja. Tätä ongelmaa helpottaa, jos yritys kouluttaa IaC-osaajiaan tarpeeksi. Jos osaamista on henkilöstöllä jo riittävästi, on IaC-projektin aloittaminenkin helpompaa. (Kimachia, 2022)

Suuremmissa projekteissa myös tarvitaan laaja IaC-työkalujen valikoima ja joskus saattaa olla niin, että tarvitaankin useita IaC-työkaluja käyttöön. IaC-työkaluihin päivittyy ominaisuuksia vaihtelevalla nopeudella, joten voi olla, että käytössä olevassa työkalussa ei olekaan vielä tarvittavia ominaisuuksia saatavilla. Siksi kannattaakin valita sellainen työkalu, jonka ominaisuudet päivittyvät nopeasti tai ovat keskittyneet tarvitsemiisi ominaisuuksiin, jolloin niiden päivitysnopeus voi olla vauhdikkaampi. (Kimachia, 2022)

IaC-projektiin voi joutua päivittämään tietoturva-asioita tai korjauksia manuaalisesti ja tällöin infrakoodi ei vastaakaan olemassa olevaa toteutusta suoraan. Nämä ristiriidat voivat johtaa virheisiin ja virheitä voi olla vaikea havaita. (Kimachia, 2022). Esimerkiksi Terraformin state-tiedoston ja manuaalisesti muokatun infran yhteensovittamisessa voi tulla esiin esimerkiksi seuraavanlaisia virheitä: Terraform voi tuhota resursseja tarpeettomasti tai luoda resursseja uudelleen. Terraformissa voi kuitenkin havaita ja korjata eroja state-tiedoston ja olemassa olevan, manuaalisesti muokatun infran, välillä, mutta helpommalla pääsee, jos toteuttaa kaikki Terraformilla toteutetun infran muutokset jatkossakin pelkästään Terraformilla. (Hashicorp, 2023). Manuaaliset muutokset voivat myös jäädä testausten tai turvatarkastusten ulkopuolelle ja muodostavat näin tietoturvariskin. (Feldsher, 2022)

Roolipohjaisen käyttöoikeuksien hallinnan (RBAC) hallinta voi olla haasteellista, koska infrakoodi tallennetaan monesti yhteiskäytössä olevaan hakemistoon (repository) esimerkiksi GitHubiin. Jos RBAC:n hallintaa ei ole toteutettu oikein, voi se olla tietoturvariski. (Kimachia, 2022)

Tietojen tallennuksessa on riskinsä IaC-ympäristössä. Jos avaimia tai muita sensitiivisiä tietoja tallennetaan tarkoituksella esimerkiksi väliaikaisesti testausta varten suoraan koodiin, muodostavat ne mahdollisesti tietoturvariskin jäädessään vahingossa tuotantoympäristön koodiin. Vaikka lähdekoodin käyttöoikeuksia hallitaan ja ne olisivat kunnossa, on IaC-koodi

silti riskialtis paikka tällaisten tietojen tallentamiseen. Sensitiiviset tiedot tulisikin tallentaa aina salaisuuksien hallintaan, esimerkiksi Azuressa Key Vaultiin. (Feldsher, 2022)

Liian laajat käyttöoikeudet ovat myös mahdollisia. Joskus kehityksen alkuvaiheessa kehittäjille annetaan pääkäyttäjän oikeudet, koska se on nopeampaa kuin luoda sellaiset roolit, jotka riittäisivät työskentelyyn. Pääkäyttäjän oikeuksilla myös voidaan varmistaa, että kehittäjällä on kaikki tarpeelliset oikeudet heti, eikä oikeuksia tarvitse lisätä tai muokata myöhemmin. Tämä on kuitenkin helposti vältettävissä, kun luodaan oikeudet vain tarpeellisiin toimintoihin, eikä lähdetä oikomaan esimerkiksi ajansäästön vuoksi. Kannattaa myös luoda automaattinen tai manuaalinen käyttämättömien roolien poistoprosessi. Ylimääräisten roolien poisto pienentää osaltaan tietoturvariskiä. (Feldsher, 2022).

Valmiiden pohjien käytössä kannattaa olla tarkkana ja tarkistaa, etteivät ne sisällä mitään ylimääräistä kuten suojaamattomia SSH-portteja. Tämäkin riski on helposti vältettävissä käytettäessä vain yrityksen omia tai muista luotettavista lähteistä hankittuja pohjia. (Feldsher, 2022)

Resurssien tagit paitsi helpottavat resurssien luojien tai omistajien jäljittämistä ongelmatilanteissa, ne myös vähentävät sellaisten resurssien löytämistä, jotka eivät enää ole käytössä tai muuten ovat irrallisia projektista. (Feldsher, 2022)

Kaikki IaC-resurssit kannattaa luoda niin, että ne menevät pipelinen läpi. Pipelineen voidaan luoda automaatioita, jotka löytävät mahdollisia ongelmia, kuten kovakoodattuja sensitiivisiä tietoja. Tietoturvan kannalta tätä kannattaa käyttää, vaikka luodut muutokset olisivat pieniäkin. Tällöin myös lokit ovat ajan tasalla ja paluu tarvittaessa edelliseen tilanteeseen on helpompaa. (Feldsher, 2022)

5 Terraform

Hashicorpin Terraform on avoimen lähdekoodin infrastructure as code-työkalu, jolla voi rakentaa, muuttaa ja versioida pilvi-, mutta myös on-prem-resursseja tehokkaasti ja turvallisesti. Terraformilla voi määrittää pilvi-infrastruktuurin ja resurssit helposti luettaviin määritystiedostoihin, jotka ovat uudelleenkäytettävissä ja jaettavissa. Terraform soveltuu sekä pienempien resurssien, kuten tallennusresurssien luontiin, mutta myös monimutkaisempiin, kuten SaaS-palveluiden hallintaan ja konfigurointiin, jos SaaS-palvelun tarjoaja tukee Terraformia. (Terraform, 2023a)

5.1 Terraformin toiminta

Terraformilla luodaan määritystiedostot, joihin määritellään haluttu infrastruktuurin tai palvelun lopputulos. Terraform toimii deklaratiivisesti, eli se itse toteuttaa halutun infran annetun lopputuloksen perusteella. Terraformille ei siis tarvitse määritellä esimerkiksi haluttua palveluiden avausjärjestystä, vaan se pystyy päättelemään lopputuloksesta, missä järjestyksessä ja resurssit tulee avata ja mitä ominaisuuksia niillä tulee olla, jotta lopputulos toimii. Terraform tallentaa itselleen järjestelmän nykytilan, joten resursseja lisätessä, muutettaessa tai poistettaessa, se pystyy päättelemään, mitä sen tulee jättää voimaan, mitä poistaa ja mitä muokata. (Terraform, 2023a)

Terraformin ydintyökalu koostuu kolmesta osasta: Wire, plan ja apply.
Write-vaiheessa määritellään Terraformille projektin resurssit, eli kirjoitetaan Terraform-koodi. Esimerkiksi virtuaalikone, verkot ja kuormantasaaja.(Terraform, 2023a)

Plan-vaiheessa Terraform tekee toteutussuunnitelman kirjoittamasi koodin perusteella ja kertoo millainen toteutettavan infran lopputulos on ja mitä se tulee lisäämään, muokkaamaan tai poistamaan toteutuksesta, perustuen antamiisi tietoihin ja olemassa olevaan infraan. (Terraform, 2023a)

Apply-vaiheessa Terraform tekee muutokset pilvialustalle, esimerkiksi Azureen. Hyväksynnän jälkeen Terraform tekee ehdottamansa toiminnot oikeassa järjestyksessä, muuttamatta mahdollisia toimivia resurssiriippuvuuksia. Esimerkiksi jos muutat virtuaalikoneen ominaisuuksia ja lisäät virtuaalikoneen, se luo virtuaalikoneen uudelleen, ennen kaikkien koneiden ominaisuuksien muutosta. (Terraform, 2023a)

5.2 Terraformin ominaisuudet

Monien palveluntarjoajien palveluita voi hallita Terraformilla. Esimerkiksi Azureen löytyy valmiina rajapinnat Terraform registrystä, joiden välityksellä saa Terraformin toimimaan Azuren kanssa. Myös muut suuret ja pienemmätkin palveluntarjoajat löytyvät valmiina samasta paikasta. Tarvittaessa myös omien rajapintojen rakentaminen on mahdollista. (Terraform, 2023a)

Eri palveluntarjoajien käyttäminen pilvi-infrassa varmistaa esimerkiksi paremman saatavuuden palveluun, jos jollain pilvialustalla onkin katkoksia palvelussaan. Kuitenkin kaikilla pilvialustoilla on omanlaisensa käyttöliittymät ja työkalut. Terraformin työnkulku on käytettävissä useille palveluntarjoajille, joten kaikkien palveluntarjoajien hallinnoinnin voi hoitaa Terraformilla. (Terraform, 2023b)

Muutosten seuranta on Terraformissa tehty helpoksi. Terraform luo toteutus- tai muutossuunnitelman koodisi perusteella ja hyväksyttää sen ennen muutosten ajamista infraan. Se myös luo senhetkisestä toteutuksesta tiedoston (state-tiedosto), johon se vertaa toteutettavaa muutosta. Tämä tiedosto kertoo aina infrastruktuurin rakenteen, edellyttäen, että kaikki muutokset tehdään Terraformia käyttäen. (Terraform, 2023a)

Terraform osaa tehdä muutokset automaattisesti oikeassa järjestyksessä. Terraform toimii deklaratiivisesti, eli sille voi kuvailla halutun lopputuloksen ja Terraform osaa päätellä miten ja missä järjestyksessä kyseinen lopputulos saadaan aikaan. Eli ei ole tarvetta syöttää tietoa toteutusjärjestyksessä, eikä antaa tietoa keskinäisistä riippuvuuksista. (Terraform, 2023a)

Terraformia käyttämällä konfiguraatiot voidaan yhdenmukaistaa. Terraformissa voi hyödyntää uudelleenkäytettäviä komponentteja, eli moduuleita. Terraform registrystä löytyy valmiita julkisia moduuleja, jotka ovat käytettävissä tai muokattavissa tarpeen mukaan. Moduuleita voi kirjoittaa myös itse, jos valmiista tarjonnasta ei löydy sopivia. (Terraform, 2023a)

Projektin osalliset voivat helposti tehdä yhteistyötä. Koska määrittelyt on kirjoitettu tiedostoon, voi tiedoston liittää versionhallintajärjestelmään, esimerkiksi GitHubiin, joten se on tiimin nähtävillä ja muokattavissa kaikkien käyttöoikeutettujen kesken. Tämä helpottaa kokonaiskuvan ylläpitämistä projektissa ja kaikki osalliset näkevät infrastruktuurin tilanteen, muutokset ja muutosten tekijät. (Terraform, 2023a)

6 Terraformin ja muiden työkalujen asennus

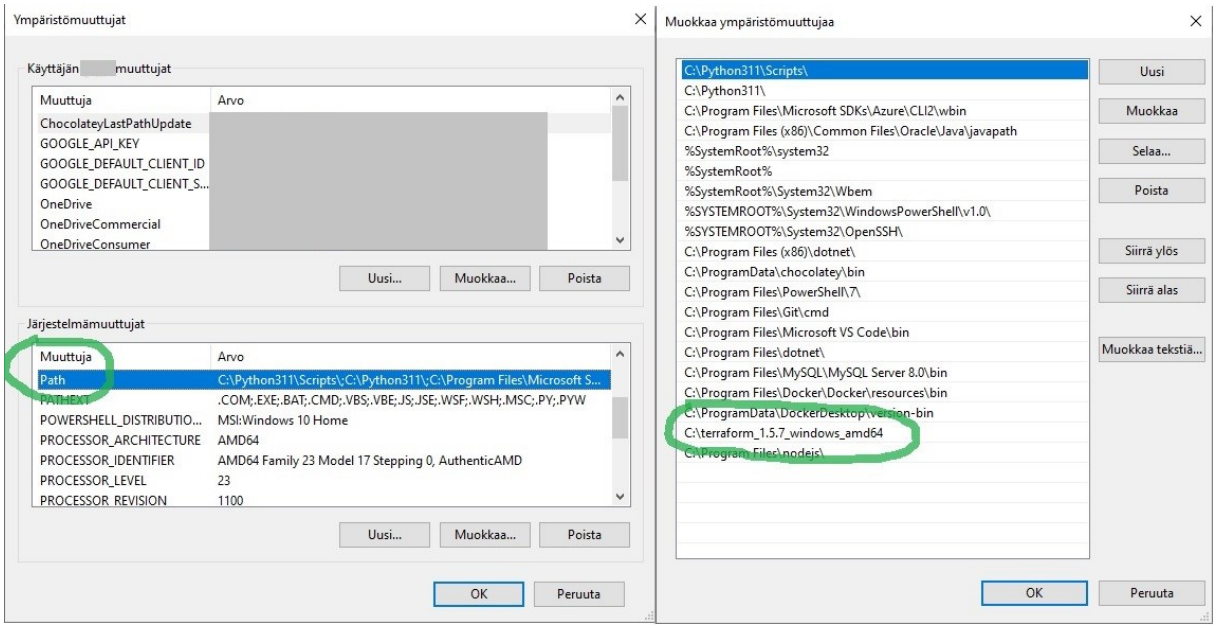
Tässä esimerkissä tavoitellaan lopputulosta, jossa Azureen avataan Windows Server 2019 Datacenter-virtuaalikone, sekä sen käyttöön olennaisesti liittyvät palvelut. Ennen kuin päästään kirjoittamaan Terraform-koodia, pitää asentaa useita ohjelmistoja taustalle ja myös muokata käyttöjärjestelmän ympäristömuuttujia. Terraform-koodia voidaan kirjoittaa lähes millä tahansa koodieditorilla. Terraformin voi asentaa useilla eri tavoilla ja asennus eroaa hieman käytetyn käyttöjärjestelmän, Windowsin, OS X:n tai Linuxin mukaan. Asennukseen voi käyttää Chocolatey-paketinhallintaohjelmistoa tai Terraformin voi asentaa manuaalisesti.

6.1 Terraform

Ohje soveltuu erityisesti tietokoneelle, joka toimii 64-bittisessä ympäristössä ja sisältää Windows 10-käyttöjärjestelmän.

Terraformin asennus aloitetaan lataamalla ensin asennustiedosto Hashicorpin nettisivuilta. Valitaan käyttöjärjestelmää ja tietokoneen muita ominaisuuksia vastaava asennustiedosto annetuista vaihtoehdoista. Ladattu paketti puretaan Terraform-kansioon esimerkiksi C-juureen. Terraform.exe-tiedostoa tuplaklikataan, jotta asennus lähtee käyntiin. Mustan ruutu vilahtaa näytöllä. Tämän enempää ei tässä vaiheessa tarvitse havaita. Kopioidaan Terraform-kansion polku leikepöydälle. Haetaan Windowsin haulla ”muokkaa järjestelmän ympäristömuuttujia”. Valitaan ”Ympäristömuuttujat”. Järjestelmämuuttujista valitaan ”Path” ja painetaan ”Muokkaa”. ”Uusi”-näppäimellä lisätään Terraformin polku leikepöydältä listan viimeiseksi, hyväksytään OK-näppäimellä ja tallennetaan sen jälkeen painamalla OK-näppäintä. Asennuksen onnistuminen voidaan tarkistaa komentokehotteessa komennolla ”terraform -version”. Jos vastaukseksi saadaan Terraformin versionumero, on asennus onnistunut.

Kuva 2 Ympäristömuuttujien muokkaus Windowsissa



6.2 Visual Studio Code

Visual Studio Code on yleisesti käytössä oleva koodieditori, joka on saatavilla yleisimmille käyttöjärjestelmille. VS Code tukee suoraan monia suosituimpia ohjelmointikieliä ja on laajennettavissa muillekin kielille sekä ohjelmistokomponenttikirjastoille. Esimerkiksi C#, Java, Go, .NET. (Microsoft Visual Studio Code, 2023)

Ladataan VS Code asennustiedosto (.exe) code.visualstudio.com-sivustolta. Käynnistetään asennus avaamalla ladattu tiedosto. Asennuksen alussa valitaan asennusvaiheen kieli. Tämän jälkeen hyväksytään ehdot ja valitaan asennuskansio. Tämän jälkeen voidaan valita esimerkiksi, halutaanko luoda pikakuvake työpöydälle. Muut oletuksena olevat rastit voidaan jättää päälle. Painetaan ”Install”.

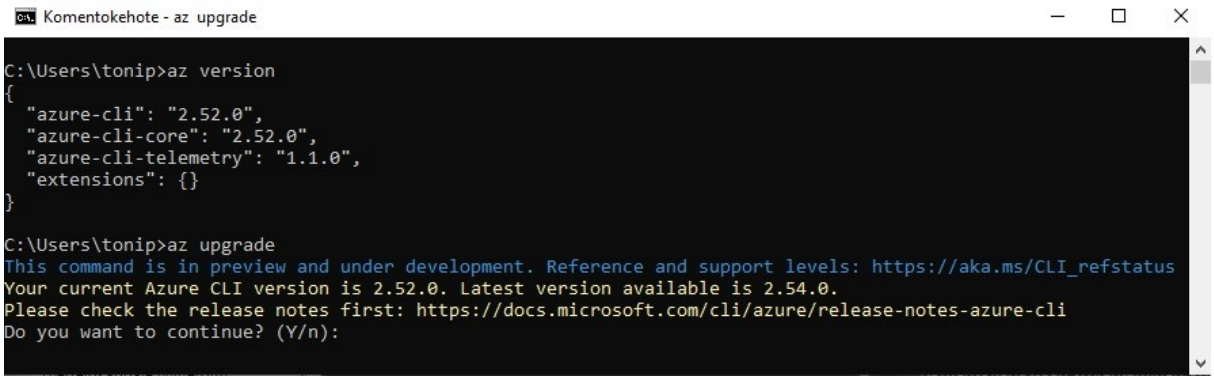
6.3 Azure CLI

Azure Command-Line Interface (CLI) on työkalu, jota käytetään komentoriviltä ja se on käyttöympäristöjen välinen. Tätä tarvitaan Terraformin käytön yhteydessä muodostamaan Azureen yhteys, sekä tällä voidaan kirjoittaa komentoja Azure-resursseille. Se sallii komentojen tai skriptien kirjottamisen komentorivin kautta. Azure CLI:n voi asentaa

paikallisesti tietokoneelle tai sitä voi käyttää myös selaimen kautta. Azure CLI:a voi käyttää myös koodieditorin, esimerkiksi VS Coden terminaalin kautta. (Microsoft, 2023a)

Azure CLI-asennustiedosto ladataan learn.microsoft.com-sivustolta. Valitaan tietokoneen käyttöjärjestelmää vastaava tiedosto. Asennus käynnistetään klikkaamalla ladattua tiedostoa. Ehtojen hyväksymisen jälkeen painetaan ”Install”. Nyt käytössä on Azure CLI. Kirjoittamalla komentokehotteeseen ”az version” -komento, nähdään versio ja voidaan varmistua, että Azure CLI on onnistuneesti asennettu. Azure CLI:n voi päivittää uusimpaan versioon komennolla ”az upgrade”.

Kuva 3 Azure CLI version tarkistus ja päivitys komentokehotteessa



6.4 Kirjautuminen Azureen

Jotta resursseja voidaan luoda Azureen, täytyy käyttäjällä olla Azure-tunnukset, jotka voi luoda portal.azure.com-sivuston kautta . Kun tunnukset on olemassa, sisälle voi kirjautua seuraavasti: komentokehotteeseen kirjoitetaan ”az login”. Selaimeen aukeaa selainikkuna, jossa voi kirjautua Azureen sisälle.

7 Terraform-koodi

Terraformin asentamisen lisäksi tarvitaan erilaisia laajennuksia. Jos haluaa käyttää Azurea tai muuta pilvipalvelualustaa, pitää ottaa käyttöön laajennus (plugin) tätä varten. Jotta kokonaisuus saadaan toimimaan, asennetaan nämä laajennukset alustusvaiheessa (komento ”terraform init”). Pilvialustan pluginit sisältävät omat konfiguraationsa, resurssityyppinsä ja datalähteensä. Terraform registrystä löytyy kaikki tietyn palveluntarjoajan tiedot ja ohjeet. (Ninawe, 2021)

7.1 Terraform-koodin osiot esimerkki-infrastruktuurissa

Jotta koodin voi kirjoittaa Visual Studio Codessa, luodaan uusi kansio ja luodaan tähän kansioon main.tf ja variables.tf nimiset tiedostot. Tiedostojen sisältö on selitetty alla.

Heti Terraform-koodin main.tf-tiedoston alussa on asetusosio, jossa määritellään Terraformin vaadittu versio, sekä käytössä oleva provider, eli tässä tapauksessa Azure. Tarvittavat tiedot löytyy Terraform registrystä, registry.terraform.io/providers/hashicorp/azurerm/latest-osoitteesta. Yläkulmasta painetaan ”Use provider”-kohtaa, niin saadaan valmiina kyseisen pilvialustan käyttöön tarvittava koodi, jonka voi kopioida koodieditoriin.

Kuva 4 Terraform asetukset ja Azuren konfigurointi Terraform-koodilla

```
# Terraform asetukset
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = ">= 3.75.0" # Valinnainen mutta suositellaan
tuotantoympäristössä
    }
  }
}
# Konfiguroidaan Microsoft Azure Provider käyttöön
provider "azurerm" {
  features {}
}
```

Seuraavaksi luodaan resurssiryhmä, johon resurssit avataan. "azurerm_resource_group" määrittelee uuden Azure-resurssin. ”rg” on resurssiryhmälle annettu tunnus, johon voidaan viitata muissa blokeissa viittaamalla siihen nimellä ”azurerm_resource_group.rg”. Name ja location on toteutettu muuttujilla, jotka on tallennettu omaan variables-tiedostoonsa. Var.rgname viittaa variables-tiedoston saman nimiseen muuttujaan. Samalla tavalla var.rglocation viittaa omaan muuttujaansa. Azure-portaalista luotu resurssi löytyy sillä nimellä, mitä name-kohtaan on kirjoitettu. Tässä tapauksessa nimi poimitaan variables.tf-tiedostosta, joten resurssiryhmän nimeksi tulee ”project1-rg”. (kts. variables.tf-tiedoston sisältö alempaa).

Kuva 5 Resurssiryhmä Terraform-koodilla

```
# Luodaan resursseille resurssiryhmä
resource   "azurerm_resource_group" "rg"   {
   name   =   var.rgname
   location   =   var.rglocation
 }
```

Variables.tf-tiedostossa rgname ja rglocation voivat sisältää esimerkiksi muuttujan tyypin, oletusnimen tai oletussijainnin, eli fyysisen Azure-palvelimen sijainnin. Kun muuttujat on talletettu omaan tiedostoonsa, tietoja ei tarvitse kertoa main.tf-tiedostossa vaan tiedot tulevat suoraan variables-tiedostosta muuttujaa kutsumalla. Variables-tiedostoon tulee tässä esimerkissä vain nämä kaksi muuttujaa.

Kuva 6 Resurssiryhmä ja sijainti muuttujina Terraform-koodilla

```
/*Tässä tiedostossa on muuttujat, joihin main-tiedostossa viitataan*/
variable "rgname" {
  type    = string
  default = "project1-rg"
}
variable "rglocation" {
  type    = string
  default = "northeurope"
}
```

”Azurerm_virtual_network”-osiossa määritetään virtuaaliverkko, joka nimetään ”vnet1”:ksi ja sille annetaan osoitealue ”10.0.0.0/16”. Tämä tarkoittaa, että se sisältää kaikki IP-osoitteet

välillä 10.0.0.0-10.0.255.255. Resurssiryhmän nimi- kohdassa viitataan ”rg”-nimiseen resurssiryhmään ”.name”-ominaisuuden kautta. Eli resurssi liitetään tähän kyseiseen resurssiryhmään.

Kuva 7 Virtuaaliverkko Terraform-koodilla

```
# Luodaan virtuaaliverkko
resource   "azurerm_virtual_network" "vnet"   {
   name   =   "vnet1"
   address_space   =   [ "10.0.0.0/16" ]
   location   =   "northeurope"
   resource_group_name   =   azurerm_resource_group.rg.name
 }
```

Ympäristö tarvitsee myös aliverkon, joka määritellään seuraavassa osiossa. Nimeäminen ja yhdistäminen oikeaan resurssiryhmään tehdään samoin kuin edellä. Aliverkko liitetään ”vnet”-nimiseen verkkoon ”.name”-ominaisuuden kautta. Aliverkolle annetaan IP-osoitealue ”10.0.1.0/24”, joten se sisältää kaikki IP-osoitteet välillä 10.0.1.0-10.0.1.255.

Kuva 8 Aliverkko Terraform-koodilla

```
# Luodaan aliverkko
 resource "azurerm_subnet" "frontendsubnet" {
  name                 = "frontendSubnet"
  resource_group_name  = azurerm_resource_group.rg.name
  virtual_network_name = azurerm_virtual_network.vnet.name
  address_prefixes     = ["10.0.1.0/24"]
}
```

Virtuaalikoneeseen voidaan luoda pääsy julkisen ip-osoitteen kautta. ”azurerm_public_ip” määrittelee uuden julkisen ip:n resurssin. Allocation method: dynamic tarkoittaa, että ip-osoite on dynaaminen, eli Azure valitsee automaattisesti vapaan ip-osoitteen. SKU on lyhenne stock keeping unitista, eli se tarkoittaa palvelutasoa. Basic on sisällöltään laajempi kuin Standard-vaihtoehto.

Kuva 9 IP-osoite Terraform-koodilla

```
# Luodaan julkinen ip-osoite
 resource   "azurerm_public_ip" "publicip"   {
   name   =   "pip1"
```

```
  location    =    "northeurope"
  resource_group_name    =    azurerm_resource_group.rg.name
  allocation_method    =    "Dynamic"
  sku    =    "Basic"
}
```

”Azurerm_network_interface” tarvitaan hallinnoimaan verkkoympäristöä. Kaikki ip-configurationin alla olevat tiedot ovat pakollisia toiminnan kannalta. Tämä osio määrittää kuinka ip-osoitteita käytetään. ”Subnet_id” määrittää mihin aliverkkoon verkkoympäristö liitetään. ”azurerm_subnet.frontendsubnet.id” viittaa aiemmin määritettyyn aliverkkoon. ”public_ip_address_id = azurerm_public_ip.publicip.id” kertoo, että verkkoliittymä liitetään aiemmin määritettyyn julkiseen IP-osoitteeseen.

Kuva 10 Verkkoympäristö Terraform-koodilla

```
# Luodaan Network Interface
resource    "azurerm_network_interface" "nic"    {
  name    =    "nic1"
  location    =    "northeurope"
  resource_group_name    =    azurerm_resource_group.rg.name

  ip_configuration    {
    name    =    "ipconfig1"
    subnet_id    =    azurerm_subnet.frontendsubnet.id
    private_ip_address_allocation    =    "Dynamic"
    public_ip_address_id    =    azurerm_public_ip.publicip.id
  }
}
```

Varsinainen virtuaalikone luodaan seuraavassa blokissa. Perustietojen, kuten nimen ja alueen lisäksi pitää kertoa, millainen kone luodaan, käyttäjätunnus ja salasana kirjautumiseen, koneen tarkemmat tiedot sekä tallennustilan tiedot. Käyttäjätunnus ja salasana kannattaa tuotannossa tallentaa Azure Key Vaultiin, joka on suojattu salaisten tietojen säilytyspaikka. Näin kirjautumistiedot eivät ole suoraan koodista luettavissa. Tässä esimerkissä ne on talletettu koodiin, jotta on helpompaa kokeilla virtuaalikoneen luonnin jälkeen koneen toimintaa näillä tunnuksilla. Nimeäminen, sijainti, resurssiryhmä ja verkkoympäristön valinta toteutetaan samalla logiikalla, kuin aiemmissa osioissa on tehty. ”Size=Standard_B1s” kertoo palvelimen koon. Virtuaalikoneen suorituskyky ja hinta

perustuvat tähän valintaan. Käyttötarkoitukseen sopivia vaihtoehtoja voi etsiä Azure-portaalista tai Azuren dokumentaatiosta. Vaihtoehtojen tutkimiseen voi käyttää myös PowerShelliä tai Azure CLI:tä.

”Source_image_reference” -kohdassa määritellään käyttöjärjestelmä, joka virtuaalikoneelle asennetaan. Tässä käytössä on Windows Serverin vuoden 2019 Datacenter-versio. Myös palvelutaso (SKU) ja versio määritellään tässä osassa.

“Os_disk”-kohdassa määritetään käyttöjärjestelmän tallennustilan asetukset. Vaihtoehtoja ”storage_account_type” -kohtaan löytää Azure-portaalista, Azuren dokumentaatiosta tai PowerShelliä tai Azure CLI:tä käyttämällä.

Kuva 11 Virtuaalikone Terraform-koodilla

```
 # Luodaan virtuaalikone. Älä koskaan tallenna salasanaa oikeasti tähän
tiedostoon.
 resource   "azurerm_windows_virtual_machine" "vm-win"   {
   name                  =   "vm1"
   location              =   "northeurope"
   resource_group_name   =   azurerm_resource_group.rg.name
   network_interface_ids =   [ azurerm_network_interface.nic.id ]
   size                  =   "Standard_B1s"
   admin_username        =   "adminuser"
   admin_password        =   "Oppari2023"

   source_image_reference   {
     publisher   =   "MicrosoftWindowsServer"
     offer       =   "WindowsServer"
     sku         =   "2019-Datacenter"
     version     =   "latest"
   }

   os_disk   {
     caching             =   "ReadWrite"
     storage_account_type   =   "Standard_LRS"
   }
 }
```

Jotta virtuaalikoneeseen voidaan muodostaa RDP-yhteys, luodaan Azureen ”Network Security Group”, ja liitetään se verkkoympäristöön. ”Security_rule”-kohdassa määritellään sääntö, joka sallii yhteyden 3389-portin kautta. Protokollana käytetään TCP:tä, säännön

prioriteetti on 100 ja tämä määrittää sääntöjen suoritusjärjestyksen. Prioriteetti voi olla välillä 100-4096 ja mitä alhaisempi luku on, sitä aiemmin se suoritetaan. Address prefix-kohdissa oleva “*” tarkoittaa, että kaikki osoitteet ovat sallittuja. Näihin kohtiin voidaan määritellä tarvittaessa lähde- ja kohdeosoitteet tarkemmin.

Kuva 12 Network Security Group Terraform-koodilla

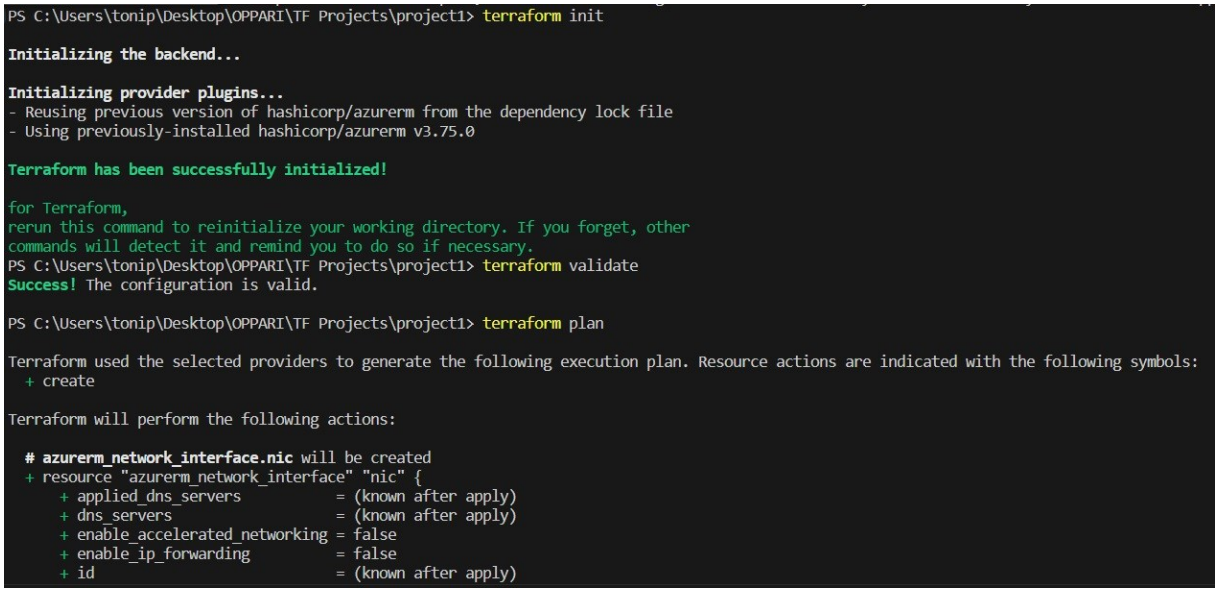
```
# Luodaan Network Security Group (NSG) ja verkkosäännöt
 resource "azurerm_network_security_group" "nsg-rdp" {
  name                = "allowrdpp"
  location            = azurerm_resource_group.rg.location
  resource_group_name = azurerm_resource_group.rg.name

  security_rule {
    name                       = "rdpport"
    priority                   = 100
    direction                  = "Inbound"
    access                     = "Allow"
    protocol                   = "Tcp"
    source_port_range          = "*"
    destination_port_range     = "3389"
    source_address_prefix      = "*"
    destination_address_prefix = "*"
  }

}
# Yhdistetään Network Interface ja NSG
resource "azurerm_network_interface_security_group_association" "association"
{
  network_interface_id      = azurerm_network_interface.nic.id
  network_security_group_id = azurerm_network_security_group.nsg-rdp.id
}
```

7.2 Resurssien avaaminen Azureen Terraform-koodin avulla

Nyt kun Terraform-koodi on valmis ja tallennettu, voidaan se muutamalla komennolla ajaa Azureen resursseiksi. VS Codessa avataan uusi terminaali valikosta kohdasta Terminal, New terminal. Tarkistetaan, että terminaalissa näkyvä tiedostopolku on se, jossa Terraform-projekti on. Terminaaliin kirjoitetaan komento ”terraform init”. Tämä komento valmistelee työhakemiston muita komentoja varten. Tämän jälkeen koodi kannattaa validoida, jotta mahdolliset virheet tulevat tässä vaiheessa esille ja ne on helppo korjata. Validointi tapahtuu kirjoittamalla terminaaliin komento ”terraform validate”. Kun validointi on onnistunut, Terraformin tulee luoda suunnitelma toteutuksesta koodiin perustuen. Komennolla ”terraform

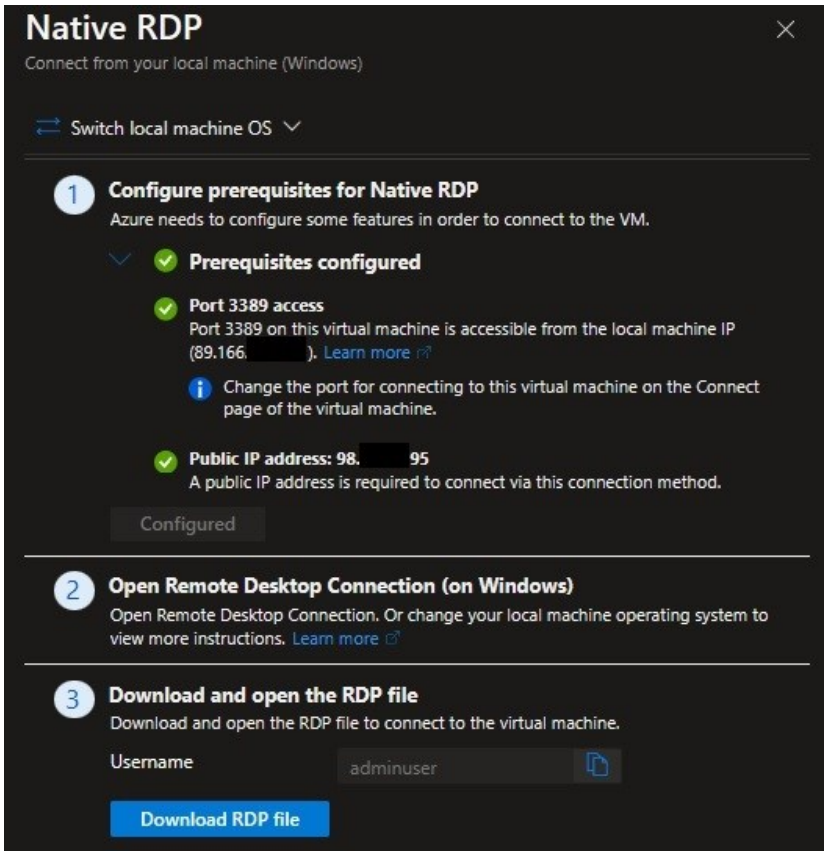
plan”, nähdään suunnitelma, miten resurssit Azureen luodaan. Suunnitelmassa nähdään lisättävät resurssit + -merkki edessään ja poistettavat on merkitty - -merkillä. Jos suunnitelma on halutun kaltainen, voidaan ajaa suunnitelma, eli avata resurssit Azureen, kirjoittamalla terminaaliin ”terraform apply”. Hyväksytään suunnitelma ”yes” -komennolla ja odotetaan, että saadaan onnistumisesta kuittaus. Tämän jälkeen resurssit ovat nähtävissä Azure-portaalissa.

Kuva 13 Terraform komentoja VS Codessa



Azure-portaalissa valitaan ”vm1”-niminen virtuaalikone. Valikosta valitaan ”Connect” ja avautuvalta sivulta ”Native RDP”-yhteys. Klikataan ”Download RDP file” ja tallennetaan tiedosto haluttuun kansioon. Kun tiedosto on ladattu, se avataan. Syötetään käyttäjätunnus ja salasana, jotka löytyvät tässä tapauksessa main.tf-tiedostosta. Edetään näytöllä näkyvien ohjeiden mukaan. Pian aukeaa Terraform-koodilla Azureen avattu virtuaalikone ja nähdään, että koodi on toiminut ja resurssit avautuneet oikein.

Kuva 14 RDP-tiedoston lataaminen Azure-portaalista



Terraformilla tehdyn projektin voi poistaa kirjoittamalla terminaaliin komennon ”terraform destroy”. Terraform näyttää tämän jälkeen suunnitelman, jossa on kerrottu kaikki poistettavat resurssit. Poistaminen hyväksytään kirjoittamalla ”yes”. Hetken kuluttua järjestelmä antaa ilmoituksen, jossa kerrotaan poistettujen resurssien määrä ja poiston onnistuminen. Käyttämättömät resurssit kannattaa poistaa heti kun niitä ei tarvitse, jotta niistä ei kerry kustannuksia.

8 Johtopäätökset ja pohdinta

Koska olin työssäni päässyt syventymään Azureen syvemmin ja Terraformiinkin hieman, ajattelin että haluan hyödyntää niitä tietoja ja taitoja opinnäytetyöni tekemisessä. Pilvipalvelut ovat yhä enemmän tulevaisuuttakin, joten halusin pitäytyä Azuressa ja opiskella sitä lisää opinnäytetyötä tehdessäni. Koska IaC kasvattaa myös suosiotaan, koin että voisin opastaa tulevia pilviammattilaisia alkuun Terraformin kanssa. Koen, että joku voi tämän työn avulla ymmärtää hieman enemmän IaC:n käyttötarkoituksesta. Terraform on vain yksi IaC-työkalu ja Microsoftillakin on oma deklaratiivinen IaC-työkalunsa, Bicep. Jos ymmärtää yhden työkalun perusperiaatteita, voi siitä olla hyötyä muidenkin vastaavien työkalujen käyttöönotossa ja päivittäisessä käytössä.

Ennen opinnäytetyön aloittamista ajattelin, että työssäni olevan Terraform-toteutuksen voisi toteuttaa jopa henkilö, jolla ei ole aiempaa tietoa Azuresta tai Terraformista. Nyt kuitenkin ajattelen, että Azuresta olisi hyvä olla jo jonkin verran perillä ennen kuin alkaa rakentamaan pilvi-infraa Terraformilla. Jos pilvipalveluiden perusteet ja Azure-tuntemus on kunnossa, uskon, että Terraform-osuuden voi toteuttaa ohjeistukseni perusteella ja päästä sitä kautta oppimaan mitä IaC käytännössä on.

Omassa nykyisessä työssä tulen jatkossa käyttämään IaC-työkaluja ja opinnäytetyötä kirjoittaessa tulikin hyvää kertausta sekä Azuren, että IaC:n perusteista, joten aiheenvalinta oli mielestäni onnistunut oman tulevaisuuteni kannalta. IaC on kuitenkin niin laaja kokonaisuus, että opittavaa varmasti riittää jatkossakin paljon.

9 Yhteenveto

Infrakoodin hyödyntäminen tuo monissa tilanteissa pilvipalveluympäristöjen luontiin huomattavia etuja ja se voi nopeuttaa ympäristön luomista merkittävästi. Erityisesti havaitsin opinnäytetyötä tehdessäni, että jos osaamista on jo valmiiksi ja kyse on mahdollisesti infrastruktuurista, joka voidaan joutua rakentamaan uudelleen tai sen osia voidaan hyödyntää myöhemmin edes osittain uudessa infraprojektissa työn nopeuttamiseksi tai helpottamiseksi, on IaC:n käyttö hyvin perusteltua. Internetistä löytyvät ohjeistukset ja keskustelut toistavat tätä samaa ajatusta ja hyvin monet ovatkin sitä mieltä, että IaC tulee kasvamaan lähivuosina yhä suuremmaksi osaksi pilvipalveluita.

Joissakin tilanteissa etuja ei saavuteta ja voi olla, että IaC:n mukanaan tuomat haasteet voivat estää IaC:n järkevän käytön. Kuitenkin monissa haasteissa tarkalla suunnittelulla ja oikeanlaisella toteutuksella voidaan monet haasteet ylittää tai niitä ei ilmene lainkaan. Riittävällä osaamisella varustetut tekijät usein osaavat ratkaista monet haasteista jo etukäteen. Esimerkiksi salaisten tietojen jääminen koodiin voidaan kokea haasteena, mutta oikein toteutettuna tämäkään ei ole IaC:ssä kovin usein ongelma. Keskusteltuani pilviosaajien kanssa aiheesta, tulimme siihen lopputulokseen, että riskit ovat melko helposti hallittavissa ja tarvittavalla osaamisella haasteet ovat selätettävissä siinä määrin, että IaC:n käyttöä kannattaa harkita lähes aina, kun pilvi-infrastruktuuria lähdetään suunnittelemaan ja rakentamaan.

Tutkimuskysymyksiin löysin mielestäni tarpeeksi tietoa ja sekä IaC:n hyvien, että huonojen puolien arviointi auttaa itseäni ymmärtämään tätä kokonaisuutta laajemmin ja pystyn ottamaan oppimiani asioita työelämässä käytäntöön. Uskon, että ihmiset, jotka eivät ole IaC:hen tutustuneet aiemmin, pystyvät opinnäytetyöni avulla saamaan alustavan kuvan laajasta IaC-kokonaisuudesta ja kenties voivat syventää osaamistaan muista saatavilla olevista lähteistä. Terraformin käyttöönottoon löytyy paljon ohjeistusta Terraformin omasta ohjeistuksesta sekä myös harrastajien tekemiä ohjeistuksia löytyy esimerkiksi GitHubista ja Youtubesta runsaasti. Näitä soveltaen loin ohjeistuksen Terraformin käyttöönottoon ja ensimmäiseen projektiin.

IaC-ympäristö kehittyy pilvipalveluiden kanssa samaa tahtia ja jatkuvasti tulee uudenlaisia työkaluja sekä uusia ominaisuuksia olemassa oleviin palveluihin. Vaikka työni koskeekin pelkästään Azurea ja Terraformia, on sen sisältö hyödynnettävissä myös muiden pilvialustojen sekä deklaratiivisten IaC-alustojen kanssa.

Lähteet

Bigelow, S. J. (2023, marraskuuta 4). *What is Microsoft Azure and How Does It Work?* Techtarget, Cloud Computing. https://www.techtarget.com/searchcloudcomputing/definition/Windows-Azure

Eskola, J. (2023, tammikuuta 5). Mikä on Azure ja mitä hyötyä siitä on? *Vetonaula*. https://www.vetonaula.fi/mika-on-azure/

Feldsher, K. (2022, kesäkuuta 21). *Common Security Risks in Infrastructure as Code*. Coralogix. https://coralogix.com/blog/security-risks-infrastructure-as-code/

Floor, A. (2023, syyskuuta 12). *Opas pilvipalveluiden ostajalle*. Solita. https://hub.solita.fi/hubfs/Oppaat%20ja%20tiedostot/Solit_Pilvipalveluiden-ostajan-opas_FI.pdf

Hashicorp. (2023, lokakuuta 8). *Manage resource drift | Terraform | HashiCorp Developer*. Manage Resource Drift | Terraform | HashiCorp Developer. https://developer.hashicorp.com/terraform/tutorials/state/resource-drift

HPE. (2023, lokakuuta 17). *What is Infrastructure as Code (IaC)?* https://www.hpe.com/fi/en/what-is/infrastructure-as-code.html

IBM. (2023a, syyskuuta 28). *What is cloud computing? | IBM*. https://www.ibm.com/topics/cloud-computing

IBM. (2023b, syyskuuta 29). *SaaS – software-as-a-service | IBM*. https://www.ibm.com/topics/saas

IBM. (2023c, syyskuuta 29). *What is IaaS (Infrastructure-as-a-Service)? | IBM*. https://www.ibm.com/topics/iaas

IBM. (2023d, syyskuuta 29). *What is PaaS (Platform-as-a-Service)? | IBM*. https://www.ibm.com/topics/paas

Kimachia, K. (2022, kesäkuuta 6). *Benefits and Drawbacks of Infrastructure as Code (IaC)*. Enterprise Networking Planet. https://www.enterprisenetworkingplanet.com/data-center/infrastructure-as-code/

Liimatta, A. (2021, toukokuuta 21). *Pilvipalvelut: Tiedä tärkeimmät termit*. https://www.tietoevry.com/fi/blogi/2021/05/pilvipalvelut-tieda-tarkeimmat-termit/

Microsoft. (2022a, maaliskuuta 17). *Microsoft announces intent to build a new datacenter region in Finland, accelerating sustainable digital transformation and enabling large scale carbon-free district heating*. Microsoft News Centre Europe. https://news.microsoft.com/europe/2022/03/17/microsoft-announces-intent-to-build-a-new-datacenter-region-in-finland-accelerating-sustainable-digital-transformation-and-enabling-large-scale-carbon-free-district-heating/

Microsoft. (2022b, joulukuuta 5). *Shared responsibility in the cloud—Microsoft Azure*. https://learn.microsoft.com/en-us/azure/security/fundamentals/shared-responsibility

Microsoft. (2023a, elokuuta 2). *What is the Azure CLI?* https://learn.microsoft.com/en-us/cli/azure/what-is-azure-cli

Microsoft. (2023b, elokuuta 22). *OP Ryhmä aloittaa kattavan pilvisiirtymän ja hyödyntää Microsoftin Suomeen rakentuvia pilvipalveluja – Uutishuone*. https://news.microsoft.com/fi-fi/2023/08/22/op-ryhma-aloittaa-kattavan-pilvisiirtyman-ja-hyodyntaa-microsoftin-suomeen-rakentuvia-pilvipalveluja/

Microsoft. (2023c, syyskuuta 12). *What is Azure—Microsoft Cloud Services | Microsoft Azure*. https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-azure/

Microsoft Techcommunity. (2022, elokuuta 24). *The History of Microsoft Azure*. TECHCOMMUNITY.MICROSOFT.COM. https://techcommunity.microsoft.com/t5/educator-developer-blog/the-history-of-microsoft-azure/ba-p/3574204

Microsoft Visual Studio Code. (2023, syyskuuta 27). *Documentation for Visual Studio Code*. https://code.visualstudio.com/docs/?dv=win

Ninawe, S. (2021, toukokuuta 26). *Learn Basic Terraform Syntax in 20 minutes*. freeCodeCamp.Org. https://www.freecodecamp.org/news/terraform-syntax-for-beginners/

Statista. (2023, elokuuta 8). *Infographic: Amazon Maintains Lead in the Cloud Market*. Statista Daily Data. https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers

Säästöpankki. (2023, elokuuta 15). *Säästöpankkiryhmä investoi yli 100 miljoonaa euroa digitalisaatioon seuraavan viiden vuoden aikana—Säästöpankki*. https://www.saastopankki.fi/fi-fi/media/ajankohtaista/2023/08/saastopankkiryhma-investoi-yli-100-miljoonaa-euroa-digitalisaatioon-seuraavan-viiden-vuoden-aikana

Terraform. (2023a, syyskuuta 19). *What is Terraform | Terraform | HashiCorp Developer*. What Is Terraform | Terraform | HashiCorp Developer. https://developer.hashicorp.com/terraform/intro

Terraform. (2023b, lokakuuta 3). *Multi-cloud provisioning*. Terraform by HashiCorp. https://www.terraform.io/use-cases/multi-cloud-deployment

Vento, J. (2021, helmikuuta 4). *Mikä on pilvipalvelu?* Loihde Cloudon. https://loihdecloudon.com/mika-on-pilvipalvelu/

Wallenius, N. (2019, marraskuuta 23). Pilvipalvelut – 7 syytä miksi pilvestä on hyötyä liiketoiminnalle vuonna 2020. *Liiketoimintalähtöistä IT-konsultointia*. https://niklaswallenius.fi/pilvi-hyoty-liiketoiminta/

Webscale. (2023, elokuuta 30). *Kysy konsultilta: Mitä IaC tarkoittaa ja mitä hyötyä siitä on?* https://www.webscale.fi/08/2023/kysy-konsultilta-mita-iac-tarkoittaa-ja-mita-hyotya-siita-on

Liite 1: Aineistonhallintasuunnitelma

Aineiston tallennus ja säilytys

Opinnäytetyössä ei muodostu uutta aineistoa esimerkiksi haastatteluista tai kyselyistä. Aineisto kertyy olemassa olevista materiaaleista ja podcasteista, joista poimin asiantuntijoiden mielipiteitä ja näkemyksiä Terraformin hyödyistä ja puutteista, sekä siitä missä tilanteissa Terraform on hyödyllinen tai hyödyt eivät ole niin suuria, että Terraformia kannattaisi käyttää. Tästä muodostuu tekstimuotoista materiaalia, jota säilytän Hämeen Ammattikorkeakoulun opiskelijoilleen tarjoamassa OneDrive-pilvitallennuspaikassa. Varmuuskopiot ovat henkilökohtaisen tietokoneeni C-levyllä. Opinnäytetyöllä ei ole toimeksiantajaa.

Henkilötietojen ja arkaluonteisten tietojen käsittely

Opinnäytetyössäni syntyvä Terraform-koodi ei sisällä henkilötietoja, eikä arkaluonteista tietoa eikä myöskään salassapidettävää materiaalia. Terraform-koodi on kirjoitettu niin, että siinä ei ole mitään salattavaa ja sen on tarkoitus olla yleisluontoista esimerkein varustettua koodia, jotta muutkin Terraformista kiinnostuneet voivat sitä tarvittaessa käyttää.

Opinnäytetyöaineiston jatkokäyttö työn valmistumisen jälkeen

Opinnäytetyön data ja materiaali säilytetään vuoden ajan tietokoneeni kovalevyllä ja erillisellä muistikortilla. Tämän jälkeen materiaali hävitetään tietoturvallisesti. OneDrivestä materiaali poistuu/poistetaan kun valmistun.