Juuso Rusanen

PILVIORKESTROINNIN TYÖKALUT

Opinnäytetyö

Tietojenkäsittely

Liiketalouden ammattikorkeakoulu

2024

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä	Juuso Rusanen
Työn nimi	Pilviorkestroinnin työkalut
Toimeksiantaja	Metatavu Oy
Vuosi	2024
Sivut	45 sivua
Työn ohjaaja	Turunen Janne

TIIVISTELMÄ

Tämän opinnäytetyön tarkoituksena oli etsiä ja tutkia olemassa olevia pilviorkestroinnin työkaluja pilviresurssien orkestrointiin, määrittelyyn ja hallintaan. Tavoitteena oli löytää vaatimuksiin verrattuna sopivin tai sopivimmat pilviorkestroinnin työkalut ja vertailla niitä keskenään.

Työkalujen toivottuihin vaatimuksiin kuului pilviresurssien määrittelyn ja hallinnan lisäksi pääsy konfiguraatioihin, metriikoihin, hälytyksiin ja käyttö- sekä komentoliittymään. Lisäksi toivottiin, että työkalut olisivat yhteensopivia AWS:n, Azuren, Google Cloudin, Githubin ja Kuberneteksen kanssa. Tuettuja ominaisuuksia ja alustoja tutkittiin työkalujen kirjallisesta dokumentaatiosta.

Vertailuun valittiin viisi eri pilviorkestroinnin työkalua: Apache Brooklyn, Cloudify, OpenStack Heat, Terraform ja OpenTofu. Työkalujen vertailussa noudatettiin laadullisen tutkimuksen perusteita, ja työkaluissa toivottujen ominaisuuksien sisältymistä tutkittiin työkalujen dokumentaatioiden avulla. Vertailussa havainnollistettiin ominaisuuksien sisältymistä ja alustojen kanssa yhteensopivuutta taulukkojen avulla sekä työkalujen toimintaa käytännön kokeilulla.

Vertailussa pärjäsivät parhaiten Cloudify, OpenTofu ja Terraform, mutta kokeiluun valittiin vain OpenTofu ja Terraform, sillä Cloudify oli maksullinen käyttää. Kokeilussa asetettiin kummallekin työkalulle sama kokeilutapaus, jonka tavoitteena oli ottaa käyttöön Apache-verkkopalvelin AWS-alustalla skripteillä muokattuna. Kummallakin työkalulla päästiin hyväksyttyyn lopputulokseen. Kokeilun tuloksista todettiin, että Terraform ja OpenTofu olivat teknisesti hyvin samanlaisia ja että kumpikin sopivat yhtä hyvin pilviorkestroinnin ratkaisuun.

OpenTofun ja Terraformin olennaisimpana erona oli niiden erilainen lisensointi. OpenTofulla on avoimen lähdekoodin lisenssin, kun taas Terraformilla ei. Täten avoimen lähdekoodin lisensointi herätti enemmän kiinnostusta OpenTofua kohtaan. Opinnäytetyön tuloksista huolimatta OpenTofua ei otettu vielä käyttöön, mutta tulokset loivat pohjaa toimeksiantajan pilviorkestrointisuunnitelmille.

Asiasanat: pilvipalvelu, resurssi, käyttöönotto, pilviorkestrointi, pilviautomatisaatio

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Business Administration
Author	Juuso Rusanen
Thesis title	Cloud orchestration tools
Commissioned by	Metatavu Oy
Time	2024
Pages	45 pages
Supervisor	Turunen Janne

ABSTRACT

The objective of the thesis was to research existing cloud orchestration tools for provisioning, defining, and managing cloud resources. The goal was to find the most suitable cloud orchestration tools according to the requirements and compare them with each other.

The requirements for the tools included access to configurations, metrics, alerts, GUI and CLI. Additionally, it was hoped that the tools would be compatible with AWS, Azure, Google Cloud, Github, and Kubernetes. Supported features and platforms were examined with the tools' documentation.

Five cloud orchestration tools were selected for comparison: Apache Brooklyn, Cloudify, OpenStack Heat, Terraform, and OpenTofu. The comparison followed the principles of qualitative research, and the inclusion of required features in the tools was investigated through their documentation. The comparison used tables to illustrate the inclusion of features and compatibility with platforms, as well as the functionality of the tools through practical experimentation.

Cloudify, OpenTofu and Terraform performed best in the comparison, but only OpenTofu and Terraform were selected for experimentation because Cloudify wasn’t free. The test case for both tools in the experiment was to deploy an Apache web server on AWS in which both tools succeeded. The results indicated that Terraform and OpenTofu were technically similar and that both were equally suitable.

However, OpenTofu had an open-source license, whereas Terraform did not. Therefore, the open-source licensing sparked more interest in OpenTofu. Despite the results of the thesis, OpenTofu has not yet been adopted, but it laid the groundwork for the thesis commissioner’s cloud orchestration plans.

Keywords: cloud service, resource, deployment, cloud orchestration, cloud automation

SISÄLLYS

1 JOHDANTO 5
2 PILVIPALVELUT JA PILVIORKESTROINTI 6
2.1 Pilvipalvelu ja pilvipalvelutyypit 6
2.2 Pilvipalvelun käyttöönottomallien tyypit 8
2.3 Kontti 10
2.4 Konttiorkestrointi 11
2.5 Pilviorkestrointi 12
2.6 Pilviautomatisaatio 14
3 PILVIORKESTROINNIN TYÖKALUJA 15
3.1 Apache Brooklyn 15
3.2 Clodify 17
3.3 OpenStack Heat 18
3.4 Terraform 19
3.5 OpenTofu 21
4 PILVIORKESTROINNIN TYÖKALUJEN VERTAILU 22
4.1 Tutkimusmenetelmät 22
4.2 Työkalujen tukemat ominaisuudet 24
4.3 Työkalujen yhteensopivuus eri alustoihin 25
4.4 Työkalujen kokeilu 26
4.4.1 Kokeilutapaus 26
4.4.2 Terraformin kokeilu 27
4.4.3 OpenTofun kokeilu 34
4.5 Vertailun yhteenveto 38
5 PÄÄTÄNTÖ 39
LÄHTEET 41

1 JOHDANTO

Opinnäytetyön aiheena on pilviorkestroinnin työkalut. Aiheidea syntyi työharjoittelun ensimmäisenä päivänä, kun minulle esitettiin ongelma vailla ratkaisua: onko olemassa pilviorkestroinnin työkaluja, joilla voitaisiin orkestroida pilviresursseja eri pilvipalveluihin ja hallita siihen liittyviä palasia yhden sovelluksen tai käyttöliittymän kautta?

Opinnäytetyön toimeksiantaja Metatavu Oy on sovelluskehitykseen erikoistunut yhtiö, joka tarjoaa räätälöityjä ohjelmistoja ja digitaalisten palveluiden muotoilua. Näihin sisältyy muun muassa verkko- ja mobiilisovellukset sekä sulautetut järjestelmät. Tämän opinnäytetyön tuotos voisi auttaa heitä valitsemaan sopivan pilviorkestroinnin työkaluston. Koska pilviratkaisut ovat yksi keskeisiä Metatavun palveluita, opinnäytetyön tulokset voisivat tehostaa niiden toteutusta ja siten edesauttaa yhtiön kilpailukykyä markkinoilla.

Aihe on tutkimuspainotteinen, vaikka opintoni ovat tähän asti olleet enimmäkseen käytännönläheisiä. Aiheena se on kuitenkin hyvin relevantti sovelluskehityksen alalla, sillä pilvipalveluiden kasvavan tarpeen seurauksena sovellusten käyttöönottaminen vaatii yhä enemmän manuaalista työtä, jota voisi keventää automatisoimalla, orkestroimalla ja yhdistämällä useiden palveluntarjoajien palveluita yhteen käyttöliittymään.

Tutkimusongelmana on löytää sopiva pilviorkestroinnin työkalusto toimeksiantajalle. Tavoitteena olisi löytää yhtiön tarpeisiin soveltuva pilviorkestroinnin työkalusto, joka orkestroi sovelluksien tai resurssien käyttöönoton pilvipalveluihin ja tarjoaa pääsy konfiguraatioihin, metriikoihin, hälytyksiin ja käyttöliittymään. Lisäksi työkalustoon mielellään sisältyisi graafinen käyttöliittymä ja integraatio Visual Studio Codeen. Tärkeintä työkalustossa on yhteensopivuus AWS-, Google Cloud-, Amazon ECR-, Github- ja Kubernetes-pilvipalveluihin.

Sovelluksen käyttöönotto pilvipalveluun vaatii riippuvuuksien täyttämisen. Näitä riippuvuuksia työkalusto pyytäisi käyttäjältä, mutta työkalusto myös täyttäisi loput ohjelmallisesti täytettävät riippuvuudet ilman käyttäjän interventiota aina kun mahdollista.

Ratkaisun tulee pystyä määrittelemään ja ottamaan käyttöön sovelluksia tai niiden tukemiseen vaadittavia pilviresursseja. Käyttäjän syötteestä voidaan määritellä käyttöönoton nimi tai tunnus, pilvialustat, joita käytetään, tunnukset pilvipalveluihin ja muut resurssit tukemaan käyttöönottoa. Ratkaisun tulisi myös suorittaa tarvittavat pyynnöt pilvipalveluihin, jotta riippuvuuksien täyttyminen voisi tapahtua oikeassa järjestyksessä.

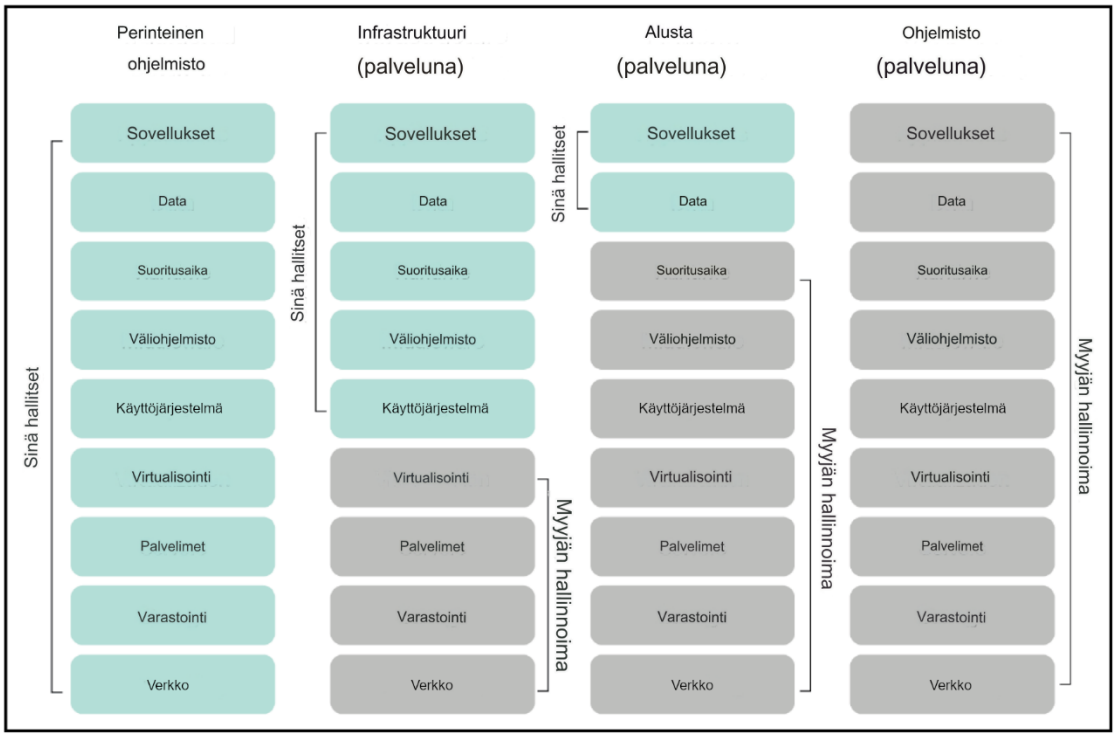
Tässä opinnäytetyössä käydään ensin läpi, mitä pilviorkestrointi ja pilvipalvelu tarkoittaa, ja avataan myös muita aiheeseen liittyviä termejä, kuten pilvipalvelutyypit, pilviautomatisaatio, kontti ja konttiorkestrointi. Sen jälkeen luetellaan ja esitellään erilaisia pilviorkestroinnin työkaluja, minkä jälkeen niitä vertaillaan ja kokeillaan.

2 PILVIPALVELUT JA PILVIORKESTROINTI

2.1 Pilvipalvelu ja pilvipalvelutyypit

Pilvipalvelu on palvelu, joka on saatavilla internetin välityksellä käyttäjälle pilvipalveluntarjoajan kautta, sen sijaan, että paikanpäällinen pilvilaskentapalvelu tarjoaisi käyttäjälle palvelua itse (Beal 2011). Pilvipalvelut ovat suunniteltu tarjoamaan käyttäjille mahdollisimman vaivattoman pääsyn sovelluksiin, resursseihin ja palveluihin. Pilvipalveluita voidaan skaalata käyttäjän tarpeen vaatiessa. Pilvipalveluita ovat esimerkiksi datan varastointi-, varmuuskopiointi-, sähköposti- ja verkossa toimivat toimisto-ohjelmistopalvelut. (Murugesan 2023, 65.)

Pilvipalvelut voidaan jakaa karkeasti kolmeen eri tyyppiin. Ensimmäinen on ohjelmisto palveluna, eli SaaS (englanniksi Software as a Service), toinen on alusta palveluna, eli PaaS (englanniksi Platform as a Service), ja kolmas on infrastruktuuri palveluna, eli IaaS (Infrastructure as a Service). (Murugesan 2023, 70.) Kuvasta 1 voidaan nähdä kyseisen kolmen pilvipalvelutyypin erot taulukossa. Perinteisellä ohjelmistolla tarkoitetaan sitä, että kaikki ohjelmiston osat tuotetaan itse (kuva 1).



Kuva 1. Pilvipalveluiden erot lueteltuina taulukkoon (mukaillen Murugesan 2023)

SaaS, eli ohjelmisto palveluna, on ohjelmiston jakelutapa, jossa käyttövalmiit sovellukset ovat käyttäjille saatavilla internetin välityksellä (Murugesan 2023, 70). Myös Hirwayn (2018, 7) e-kirjassa kerrotaan, että käyttäjät voivat muodostaa sovelluksiin yhteyden verkon välityksellä, ja lisää, että useimmat SaaS-sovellukset ovat suoraan käytettävissä verkkoselaimella ilman, että käyttäjän tarvitsee suorittaa suuria latauksia tai asennuksia. SaaS-ohjelmistojen käyttäjien täytyy tilata SaaS-tarjoajan palvelu käyttääkseen niitä. Tilaamisen hinta voi lyhyellä käytöllä olla matalampi kuin ohjelmiston ostaminen, asentaminen ja huoltaminen. SaaS-ohjelmistojen käyttö mahdollistaa organisaation työvoiman liikkeellä pysyminen kaikkialla, missä internetyhteys on saatavilla. (Hirway 2018, 8.)

PaaS, eli alusta palveluna, on tietokonelaskenta-alustan tarjoava palvelu, joka on tarkoitettu pääsääntöisesti sovelluskehittäjien tarpeita varten. Se tarjoaa sovelluskehittäjille sovelluskehitysalustan, johon käyttöönottaa sovelluksia ilman, että palvelinraudan hankintaa, huoltamista tai säännöstelyä tarvitsee tehdä itse. Tällaiset alustat voivat myös tarjota tietokantaintegraatioita ja varastointitilaa. (Murugesan 2023, 67.) PaaS-mallin olennaisena etuna on se, että kaikki tällaisella alustalla luodut sovellukset perivät kaikki saman pilvialus-

tan ominaisuudet. Sovelluskehitys, testaaminen ja käyttöönottaminen on paljon nopeampaa PaaS-alustalla, sillä se vaatii vähemmän ohjelmointia ja helpottaa sovellusten siirtämistä hybridipilveen. (Hirway 2018, 8.)

IaaS, eli infrastruktuuri palveluna, on palvelu, joka tarjoaa käyttäjillensä palvelinrautaa, datan varastointitilaa, palvelimia ja verkkokomponentteja. Tällainen palveluntarjoaja huoltaa ja ylläpitää omistamaansa infrastruktuuria itse. (Murugesan 2023, 65.) Tällaista pilvipalvelumallia käyttäessä voidaan luoda virtuaalinen datakeskus pilveen, jolla on samanlaiset resurssikyvyt kuin perinteisellä datakeskuksella (Hirway 2018, 8).

Lisäksi käyttäjät voivat perustaa yksityisen datakeskuksen pilveen ja luoda yhteyden siihen omilla paikanpäällisillä datakeskuksilla. Infrastruktuuri palveluna voi tarjota ohjelmistokehittäjille myös ohjelmistorajapintoja, joiden avulla kehittäjillä on pääsy pilviresursseihin ohjelmallisesti. Käyttäjien ei tarvitse investoida laitteistoon ja pystyvät ottamaan pilviresurssit hyödykseen ja voivat siten maksaa palvelusta vain käytön mukaan. (Hirway 2018, 8.)

2.2 Pilvipalvelun käyttöönottomallien tyypit

Yksityispilvi on yhden käyttäjän tai organisaation pilviympäristö, joka on sen käyttäjän yksinomaisessa käytössä ja hallinnassa (Kimachia 2023). Se on organisaation sisäiseen käyttöön luotu pilviympäristö (Hirway 2018, 10). Yksityispilvi voi olla monella tapaa hallittu ja isännöity. Resursseja voidaan käyttää esimerkiksi organisaation omista palvelimista tai kolmannen osapuolen infrastruktuurista (Kimachia 2023).

Yksityispilvellä on monia samankaltaisuuksia, kun sitä verrataan julkiseen pilveen, johon sisältyy muun muassa resurssien allastaminen, elastisuus ja itsehuolto, toimitettuna standardisoiduilla käytännöillä, mutta paremmilla hallinta- ja kustomointimahdollisuuksilla (Hirway 2018, 10). Organisaatioissa yksityispilvi valitaan julkisen pilven sijaan yleensä tietosuojan ja noudatuksien helpottamiseksi (Maguire 2017).

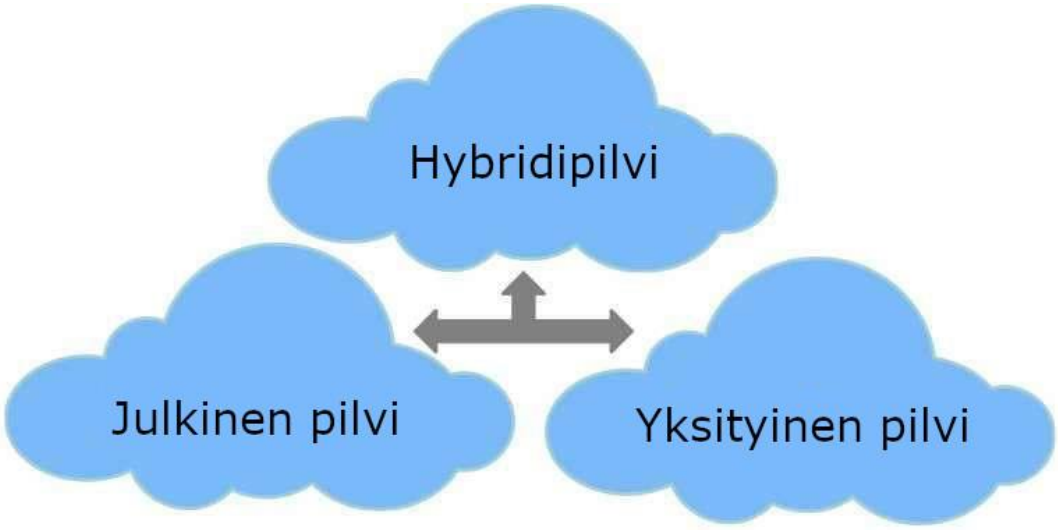
Yksityispilven etuihin kuuluu se, että asiakas voi ostaa itselleen sopivat laitteistot ja ohjelmistot ja voi muokata näitä tarpeisiinsa sopiviksi esimerkiksi liitännäisten avulla (Hewlett Packard Enterprise s.a.). Yksityispilven kykyjä voidaan myös laajentaa julkisen pilven tarjoamilla pilvipalveluilla. Esimerkiksi sovelluksen vaatiessa enemmän resursseja voidaan niitä hankkia julkisesta pilvestä lisää ilman, että sisäistä infrastruktuuria tarvitsisi laajentaa. (Citrix s.a.)

Monipilvi tai monipilvisyys tarkoittaa useamman pilvipalveluntarjoajan palveluiden käyttöä samanaikaisesti yhden sovelluksen tukemista varten. Julkinen pilvipalvelu on palvelu, jota useampi asiakas käyttää keskenään. (Cloudflare s.a.) Monipilviympäristössä, pilviympäristöjä on mahdollista pitää julkisina, yksityisinä tai niiden yhdistelmänä. Monipilviympäristön etuna on vapaus käyttää useita pilvipalveluja, jotka parhaiten sopivat sovelluksen tueksi. (Google s.a.)

Monipilveä käytetään usein sovelluksen ominaisuuksien, kuten laskennan ja datan käsittelyn jakamiseen eri ympäristöihin, jotka parhaiten tukevat näiden toimintaa. Resurssien käyttö voi olla tehokkaampaa ja näin vähentää kuluja. Mahdollisuuksia on monia. Esimerkiksi sovelluksen käyttöliittymä voi olla erillisesti käyttöönotettuna toisessa pilvipalvelussa kuin tietokanta. Myös data-analytiikka ja tietokanta sekä tuotanto- ja kehitysversiot sovelluksista voivat olla erillään toisissa palveluissa. (Oracle s.a.)

Julkinen pilvi on pilvipalvelu, joka on saatavilla kenelle tahansa internetin välityksellä ja yksityinen pilvi on yksityinen pilviympäristö, joka on käytettävissä vain sen palveluympäristön omistajille tai toisille heidän valikoiduille entiteeteilleensä, kuten asiakkailleen tai liikekumppaneilleen. Hybridipilvi toisaalta tarjoaa kyvyn integroida ja yhdistää julkisen ja yksityisen pilven palveluihin samanaikaisesti luoden virtuaalisen tietokonelaskenta ympäristön, joka on sujuva yhdistelmä paikallista fyysistä infrastruktuuria ja virtualisoitua infrastruktuuria, joka voi sijaita paikan päällä tai jossain muualla. (Hurwitz 2012, 25.)

Hirwayn (2018, 18) e-kirjassa myös todetaan, että laajimmin hyväksytty määritelmä hybridipilvelle on se, että hybridipilvi on yksityisestä ja julkisesta pilvestä koostuva pilviympäristö. Kuvassa 2 esitellään hybridipilvi julkisen ja yksityisen pilven muodostamana kokonaisuutena.



Kuva 2. Hybridipilvi yhdistää julkisen ja yksityisen pilven (mukaillen Shaik 2018)

Hybridipilven kolmantena komponenttina voi Neenanin (s.a.) mukaan olla myös jokin PaaS- tai SaaS-tarjoajan isännöimä palvelu, joka voisi olla sovellus tai datavaranto. Yritykset tarvitsevat tällaisia työkuormia, mutta päättävät olla isännöimästä niitä itse (Neenan s.a).

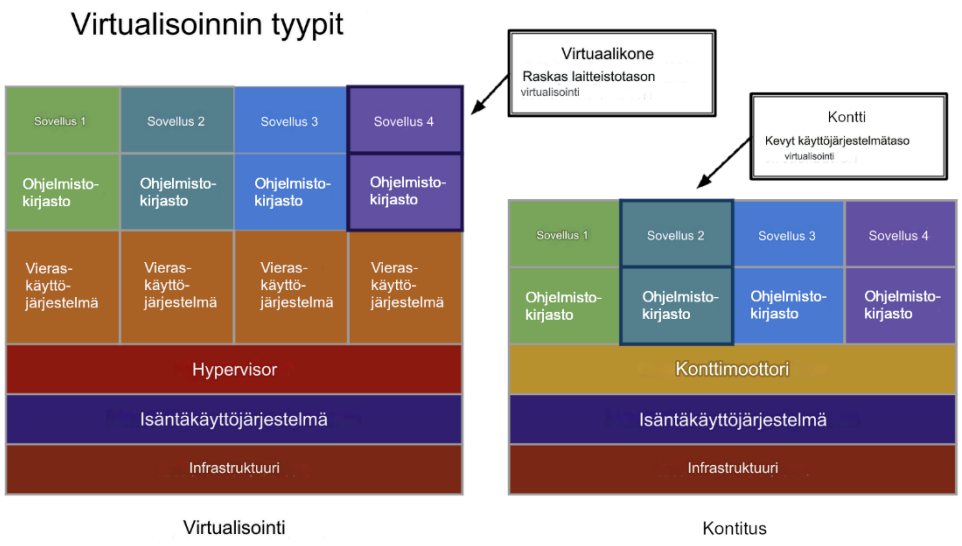
2.3 Kontti

Kontit ovat suoritettavia ohjelmiston yksiköitä, joissa ohjelmistokoodit ovat pakattu sen kirjastojen ja riippuvuuksien kanssa sellaisella tavalla, jotta sen voisi suorittaa millä tahansa alustalla tai laitteella (IBM s.a). Kontti emuloi käyttöjärjestelmää ja se on eristetty tietokonelaskennan ympäristö, jossa ohjelmistot säilötään ja käytetään (Ingalls 2021).

Ongelma, jonka kontit ratkaisevat, on se, kuinka suorittaa ohjelmistoja luotettavasti, kun niitä siirretään laskentaympäristöstä toiseen, kuten sovelluskehittäjän kannettavasta tietokoneesta testausympäristöön tai datakeskuksen palvelimesta yksityisen- tai julkisen pilven virtuaalikoneeseen (Rubens 2017). Virtualisoinnin myötä sovellusten liikutettavuudesta ja joustavuudesta on tullut välttämättömyys alalla (Ingalls 2021).

Mitä virtualisointiin tulee, kaksi virtualisoinnin ohjelmistokehystä on noussut esiin: virtuaalikoneet ja kontit. Kumpikaan ei ole toisiaan poissulkevia, vaan

kummankin tarkoitus on helpottaa fyysisen laitteen sisältöjen siirtämistä toiselle laitteelle. (Ingalls 2021.) Merkittävin ero virtuaalikoneiden ja konttien välillä on virtuaalikoneen riippuvuus hypervisor-ohjelmistokerroksesta, jolla virtualisoidaan tietokoneen laitteistoa (IBM s.a). Lisäksi konteissa ei tarvita vieraskäyttöjärjestelmiä eri ohjelmistojen suorittamista varten, toisin kuin virtualisoinnissa (kuva 3).



Kuva 3. Virtualisoinnin kaksi tyyppiä: virtualisointi ja kontitus (mukaillen Ingalls 2021)

Kontin käytöllä on useita etuja virtuaalikoneen käyttöön verrattuna. Merkittävin niistä on kontin pienempi koko, koska kontissa oleva sovellus voi jakaa samaa käyttöjärjestelmää useamman toisen kontissa olevan sovelluksen kanssa. Toisinaan yksittäisen virtuaalikoneen ympäristö pitää sisällään kokonaista käyttöjärjestelmää yhtä sovellusta kohti. (Rubens 2017.) Laitteiston virtualisoinnin sijaan kontit virtualisoivat vain käyttöjärjestelmän sisältäen vain sovelluksen itse ja sen tarvitsemat sovelluskirjastot ja riippuvuudet (IBM s.a).

2.4 Konttiorkestrointi

Kontitettujen sovellusten hallinta voi olla monimutkaista, kun niihin usein käytetään useampia kontteja, jotka täytyy konfiguroida ja ottaa käyttöön erillisinä entiteetteinä (Bhuyan 2023). Tätä ongelmaa ratkaisee konttiorkestrointi, jota Bhuyan (2023) artikkelissaan kuvailee kontitettujen sovellusten automaattiseksi käyttöönotoksi, skaalaamiseksi, ja hallinnaksi. Se auttaa sovellusten

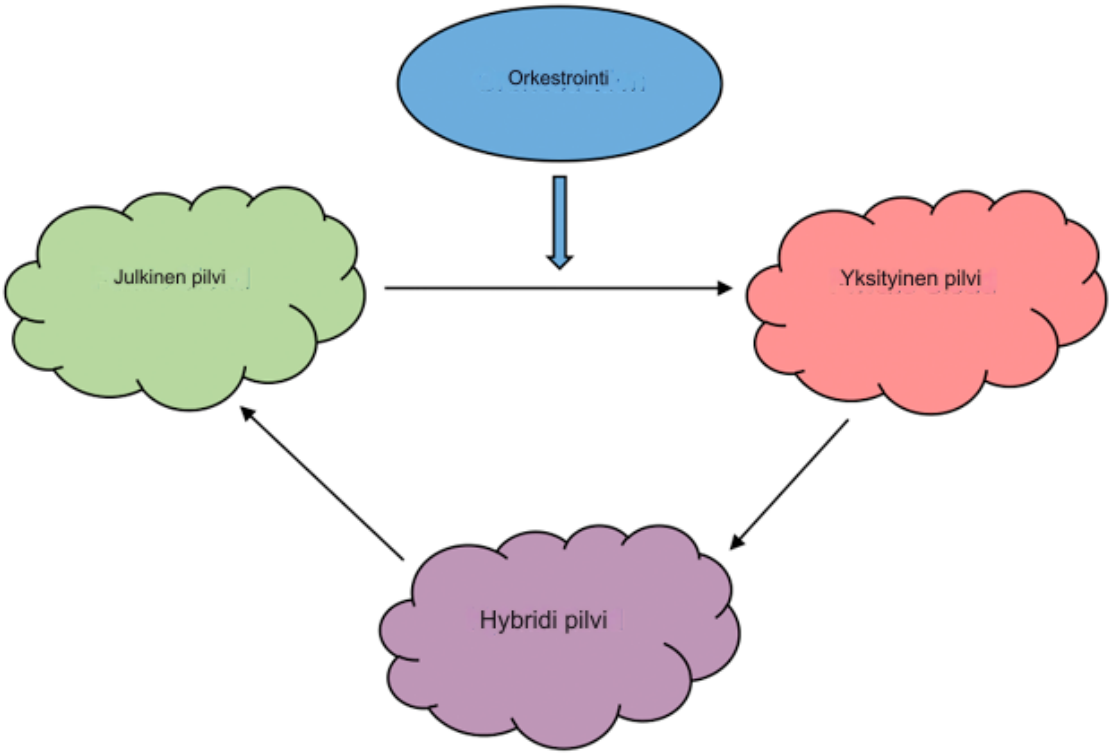
käyttöönottoa useisiin erilaisiin ympäristöihin ilman keskeytyksiä tai tarvetta suunnitella sovellusta uudelleen (Ihuman 2021).

Toisinaan Liun (2020) mukaan konttiorkestroinnilla referoidaan työkaluihin ja alustoihin, jotka ovat yksittäisten konttien määrittelemien työkuormien automatisointia, hallintaa ja ajoittamista. Konttiorkestroinnin työkalut tarjoavat alustan tehtävien automatisoinnille, mahdollistaen ison mittakaavan konttien hallinnan kevyemmin (Bhuyan 2023). Kubernetes on tällä hetkellä yksi suosituimmista konttiorkestroinnin työkaluista (Liu 2020).

Bhuyan (2023) listaa artikkelissaan konttiorkestroinnin etuja, joita on muun muassa sovelluksen parempi skaalautuminen ja siirrettävyys. Konttiorkestroinnin työkalut tarjoavat ominaisuuksia automaattiseen skaalaamiseen ja kuormituksen tasapainottamiseen, mikä tekee sovellusten käynnistämisestä ja sammuttamisesta helpompaa, kun tarve muuttuu. Nämä työkalut mahdollistavat myös sovelluksien helpomman siirtämisen erilaisten infrastruktuuripalveluiden välillä. (Bhuyan 2023.)

2.5 Pilviorkestrointi

Pilviorkestrointi on prosessien, resurssien, tietokonejärjestelmien ja palveluiden automatisointia ja koordinointia pilviympäristössä (Maksimov 2023). Tällaisten palveluiden hallintaa voidaan automatisoida julkisessa-, yksityisessä- ja hybridipilvessä, kuten kuvassa 4 illustroidaan. Pilviorkestrointi on palveluiden päästä päähän automatisoitua käyttöönottoa pilviympäristössä ja tietokonejärjestelmien, väliohjelmistojen ja palveluiden automatisoitua järjestämistä, organisointia ja hallinnointia. (Venugopal 2016.) Sillä voidaan automatisoida sovelluksia, työkuormia, tukevia resursseja ja infrastruktuuria yhdessä tai useammassa pilviympäristössä (Redhat 2023). Pilviorkestrointi varmistaa useiden automatisoitujen tehtävien ja resurssien saumattoman integraation (Maksimov 2023).



Kuva 4. Pilviorkestrointi voi automatisoida usean pilvityypin palveluiden hallintaa (mukaillen Venugopal 2016)

Pilviorkestroinnin työkalut ovat alustoja, jotka mahdollistavat useamman pilvipalvelun käyttämisen yhdeltä alustalta (Walker 2023). Nämä työkalut auttavat tietotekniikan organisaatioita vähentämään toistavaa ja manuaalista työtä, standardoimaan sovellusten käyttöönottoprosesseja ja operaatioita sekä vauhdittaa sovellusten julkaisua (Cisco s.a). Pilviympäristöt vaativat nykyään automatisointia toimiakseen, sillä niihin liittyvien kompleksien tehtävien määrä on niin suuri, että näiden manuaalinen suorittaminen olisi käytännössä hyvin epätehokasta. Näihin tehtäviin sisältyy muun muassa palvelinten käyttöönotto, verkon hallinta, tallennuskapasiteetin hallinta, virtuaalikoneiden luominen ja sovellusten käyttöönotto. (Redhat 2023.)

Pilviorkestroinnin ydinetuihin kuuluu monitorointi, hälytykset, raportit odottamattomista tilanteista, yksinkertaisemmat datan integraatiot, automaattinen sääntöjen noudattaminen, automaattinen riippuvuuksien hallinta ja työnkuormien suoraviivaistaminen. Pilviorkestrointi eliminoi myös manuaaliseen työhön liittyviä virheitä. (Vmware s.a.)

Pilviorkestroinnin implementointiin on kaksi eri suuntaa: imperatiivinen, eli dynaaminen riippuvuuksien hallinta, tai deklaratiivinen, eli valmiiksi määritetyt

riippuvuudet. Pilviorkestroinnin implementaatiot vaihtelevat myös paljon mitä tulee erilaisten pilvipalveluiden yhteensopivuuteen. Jotkut implementaatiot tarjoavat kohdistetumpaa tukea tiettyihin alustoihin tai pilvipalveluihin tuoden mukanaan enemmän valmiiksi koottuja ratkaisuja ja vaihtoehtoja kustomointiin. (Cisco s.a.)

Maksimov (2023) suosittelee blogipostauksessaan käytettäväksi IaC, eli infrastruktuuri koodina (englanniksi Infrastructure as Code) -mallia resurssien määrittelyyn ja hallintaan sekä monitorointityökalujen käyttöä pilvi-infrastruktuurin tarkkailuun ja optimointiin. Hän myös kehottaa resurssien hallintaan modulaarista ja uudelleen käytettävää lähestymistapaa.

2.6 Pilviautomatisaatio

Pilviautomatisaatio on pilviresurssien ja pilvi-infrastruktuurin osien käyttämistä ilman ihmisen väliintuloa. Automatisaatiolla tässä tapauksessa tarkoitetaan sitä, että yksittäinen, suoraviivainen tehtävä saadaan tapahtumaan automaattisesti. Tällainen tehtävä voisi esimerkiksi olla virtuaalikoneen resurssien säännöstely tai Kubernetes-kontin konfigurointi. (Maronde 2022.) Pilviautomatisaatio eroaa pilviorkestroinnista siten, että pilviautomatisaatio keskittyy yksittäisten tehtävien suorittamiseen, kun pilviorkestrointi yhdistää useampia automatisoituja tehtäviä (Redhat 2023).

Pilviautomatisaatio ratkaisee suurten resurssimäärien hallintaongelman ja vähentää siihen liittyviä virheitä. Sen lisäksi se helpottaa resurssien testaamista pilviratkaisun kehitysvaiheessa ja käyttöönottoa sen kehitysvaiheessa sekä vähentää yleisesti paljon manuaalista työtä. Pilviautomatisaatio yleensä sisältyy pilvipalveluissa integroituina työkaluina tai sisältyy julkisen pilvipalveluntarjoajan tarjoamien ominaisuuksien sviittiin. (Perry 2019.)

Pilviympäristöt sisältävät lukuisia erityyppisiä palveluita, joita voidaan jatkuvasti skaalata tai sammuttaa. Nämä ovat resursseja, joita voidaan hallita pilviautomatisaatiolla, jotta pilviympäristöstä voidaan saada eniten irti ja kulut voidaan minimoida. (Hempstead 2021.)

3 PILVIORKESTROINNIN TYÖKALUJA

3.1 Apache Brooklyn

Apache Brooklyn on avoimen lähdekoodin ohjelmistokehys sovellusten mallintamiseen, monitorointiin ja hallintaan autonomisten suunnitelmien (englanniksi autonomic blueprints) avulla. Se on työkalu, jolla voidaan hallita säännöstelyä ja sovelluksien käyttöönottoa, monitoroida sovelluksien terveyttä ja metriikoita sekä hallita sovelluksia kompleksien käytäntöjen mukaisesti. Brooklyn ymmärtää eri komponenttien välisiä riippuvuuksia, kuten esimerkiksi sen, että uuden verkkopalvelimen lisääminen vaatii kuormantasaajan (englanniksi load balancer) uudelleenkonfigurointia. (The theory behind Brooklyn s.a.)

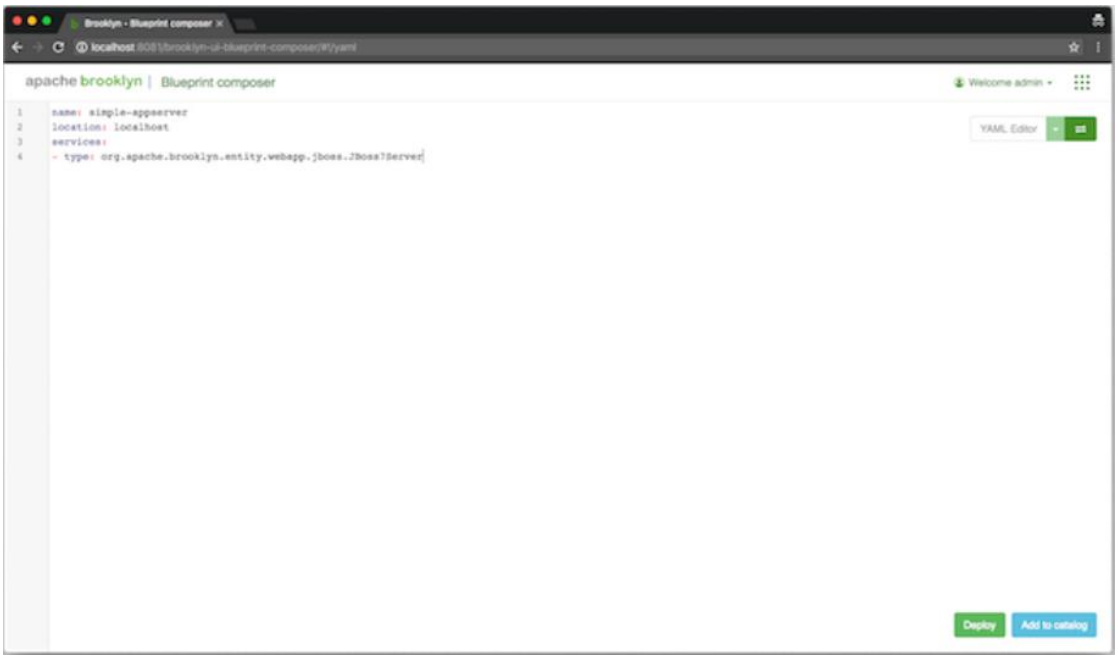
Brooklyn tarjoaa REST-ohjelmointirajapinnan ja graafisen käyttöliittymän sekä sallii autonomisten suunnitelmien kohtelun olevan olennainen osa sovellusta. Tämän työkalun avulla nämä käytännöt tulevat olemaan modulaarisia komponentteja, joita voidaan uudelleen käyttää ja lisätä suunnitelmiin. Brooklynin avulla voidaan muovata sovellukselle täysipino (englanniksi full stack); pilvi- ja ei-pilvikohteisiin käyttöönotto, monitorointityökaluja hyödyntämällä sovelluksen hyvinvointi- ja suorituskykymetriikoiden kerääminen, ja tapahtumiin, kuten epäonnistuvaan solmuun (englanniksi node) vastaaminen sekä kapasiteetin lisääminen että vähentäminen tarpeen vaatiessa. (The theory behind Brooklyn s.a.)

Brooklyn suunnitelma määrittelee sovelluksen käyttämällä deklaratiivista YAML-syntaksia tukien JVM-liitännäisiä (englanniksi plugin). Perussuunnitelma voi sisältää yhden prosessin, kuten verkkosovelluspalvelimen suorittaen WAR-tiedoston tai SQL-tietokannan ja sen liittyvän DDL-skriptin. Kompleksisemmat suunnitelmat voivat kattaa prosessien yhdistelmiä useamman koneen tai palvelun välillä. Myös isompi rykelmäsovellus (englanniksi clustered application) toimimassa useammalla alueella voidaan määritellä. (The theory behind Brooklyn s.a.)

Brooklyn ottaa suunnitelman ja voi ottaa sen käyttöön yhdessä tai useammassa eri pilvessä, yksityispilvessä tai jollain mulla alustalla. Työkalu dynaa-

misesti konfiguroi ja yhdistää kaikki sovelluksen eri komponentit, jolloin kuormantasaajat tietävät missä verkkopalvelimet ovat ja verkkopalvelimet tietävät missä tietokannat ovat. (The theory behind Brooklyn s.a.)

Brooklyn voi kerätä avainmetriikoita sovelluksen hyvinvointiin liittyen, esimerkiksi lähettämällä pyynnön sille ja mittaamalla viiveen tai asentamalla monitorointityökaluja ja käyttämällä niitä palvelimen pyyntöjonon pituuden mittaamiseen. Näitä metriikoita voidaan syöttää käytäntöihin, jotka automaattisesti suorittavat toimintoja, kuten solmujen uudelleen käynnistyksen sen epäonnistuessa, tai skaalata sovellusta uudestaan käyttäjäkysynnän ylittäen tarjonnan. (The theory behind Brooklyn s.a.) Moni näistä työkaluista on myös käyttöliittymän kautta käytettävissä, esimerkiksi YAML-suunnitelman käyttöönotto onnistuu käyttöliittymän avulla (Kuva 5).



Kuva 5. YAML-suunnitelman käyttöönotto Brooklyn:in käyttöliittymässä (Apache Brooklyn Creating YAML Blueprints s.a)

Brooklyn-sijainti on ympäristö, jolla Brooklyn voi ottaa sovelluksia käyttöön. Tällaisia sijainteja voi esimerkiksi olla AWS tai IBM Softlayer (Apache Brooklyn Locations s.a). Brooklynissa voi määritellä myös Kubernetes-sijainnin (Apache Brooklyn Kubernetes-luokkamääritelmä s.a). Brooklynissä on dokumentaation (Apache Brooklyn Deploying Blueprints s.a) mukaan myös tuki Azure- ja Google-sijainnille.

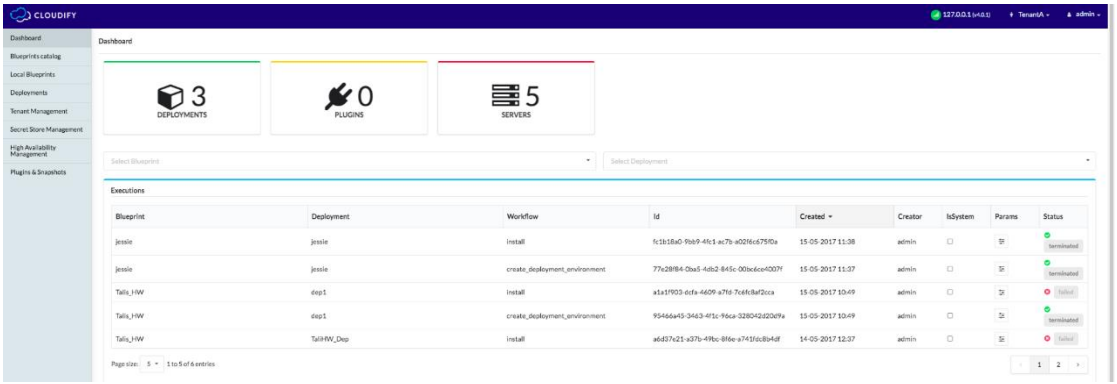
3.2 Clodify

Cloudify on monipilvi ja reunaorkestrointialusta (englanniksi edge orchestration platform), joka mahdollistaa organisaatioiden siirtymisen julkiseen pilveen ja pilvinatiiviin arkkitehtuuriin (englanniksi cloud-native infrastructure) mahdollistamalla olemassa olevan infrastruktuurin automatisoinnin pilvinatiivin ja hajautettujen reunaresurssien (englanniksi distributed edge resources) rinnalla. Cloudify antaa myös käyttäjille mahdollisuuden hallita toisia orkestrointi- ja automaatioverkkotunnuksia osana yhteistä CI/CD-kanavaa (englanniksi CI/CD pipeline). (What is Cloudify s.a.)

Cloudifyn avainominaisuuksiin kuuluu palvelukokoonpano (englanniksi service composition), joka mahdollistaa komposiittipalvelun mallintamisen, sisältäen komponentteja useasta Cloudifyn palvelusta ja muista orkestrointiverkkotunnuksista. Se käsittelee palveluiden välisien suhteiden mallintamista, peräkkäisiä työnkulkuja, jaettuja resursseja ja hajautettua elämänkaaren hallintaa (englanniksi distributed life-cycle management). (What is Cloudify s.a.)

Cloudify tukee Kubernetes-konttiorkestrointityökalua ja useita Kubernetes-rykelmiä, kuten OpenShiftiä, GKE:tä, EKS:ää, AKS:ää ja KubeSprayta. Rykelmien perustamisen ja konfiguroinnin voi automatisoida sisäänrakennetuilla suunnitelmilla. Clodifyilla on myös integraatiot esimerkiksi seuraaviin infrastruktuuriorkestrointialustoihin: AWS Cloudformation, Azure ARM, Ansible ja Terraform. Sen lisäksi sisäänrakennettu CI/CD-integraatio CI/CD-alustoille, kuten Jenkinsille, tarjoaa yhden kanavan kaikkien orkestrointialustojen hallintaan. (What is Cloudify s.a.) Google Cloud tuki löytyy myös laajennuksen myötä (Cloudify Google Cloud Plugin s.a), niin kuin myös Kubernetekselle (Cloudify Kubernetes Plugin s.a).

Cloudifyihin sisältyy graafinen käyttöliittymä, jonka ominaisuudet ovat saatavilla myös REST-rajapinnan ja komentolinjan kautta (What is Cloudify s.a.), ja VSCode-integraatio virallisena liitännäisenä (Terramel 2023). Kuvassa 6 on Cloudify-käyttöliittymä, josta voidaan nähdä eri komponenttien katalogi ja käyttöönotettujen suunnitelmien tila.



Kuva 6. Cloudify-käyttöliittymän yleisnäkymä (Sela 2017)

Cloudify käyttää IaC:hen (Infrastructure as Code) perustuvaa mallintamista, jossa käyttäjät määrittelevät halutun järjestelmän tilan, sen sijaan, kuinka kyseiseen tilaan päästään. Cloudify automaattisesti generoi asennuksen, poistamisen, parantamisen ja skaalaustyönkulun. Cloudify myös sallii käyttäjien määritellä omia, mukautettuja työnkulkuja. (What is Cloudify s.a.)

3.3 OpenStack Heat

OpenStack Heat on palvelu, jolla voi orkestroida komposiittisovelluksia käyttämällä deklaratiivista kaavaformaattia OpenStack-native REST-ohjelmointirajapinnan kautta. Heat tarjoaa kaavapohjaisen orkestroinnin pilvisovellusten määrittelemiseksi suorittaen sopivia OpenStack API-kutsuja generoidakseen pilvisovelluksia. (Welcome to the Heat documentation s.a.)

Heat-kaavio kuvaa infrastruktuurin pilvisovellukselle tekstitiedostona, jotka ovat ihmiselle luettavia ja kirjoitettavia, ja niitä voi hallita version hallintatyökaluilla. Kaaviot täsmentävät eri resurssien väliset suhteet, mikä mahdollistaa Heatin tehdä kutsuja OpenStack-rajapintoihin luodakseen kaiken infrastruktuurin oikeassa järjestyksessä sovelluksen käynnistämiseksi. (Welcome to the Heat documentation s.a.)

Heat integroi muita OpenStack-komponentteja ja kaavio sallii OpenStack-resurssityyppien luomisen. Se päämääräisesti hallinnoi infrastruktuuria, mutta sen kaaviot integroivat hyvin ohjelmistokonfiguraation hallintatyökaluja, kuten Puppettia ja Ansiblea. Heatin ominaisuuksia voidaan laajentaa asennettavilla liitännäisillä. (Welcome to the Heat documentation s.a.)

Heatiin on olemassa graafinen käyttöliittymä Heat Dashboard-liitännäisen muodossa (Welcome to Heat Dashboard s.a). Heat tukee myös metriikoita ja hälytyksiä OpenStack-resurssityyppien kautta (OpenStack Resource Types s.a). Heat käyttää dokumentaation (OpenStack Heat Architecture s.a) mukaan AWS CloudFormation -kaavaformaattia vain OpenStackiä varten. Tosin Kubernetes on myös tuettu (Kubernetes OpenStach Heat s.a).

3.4 Terraform

Terraform on IaC-työkalu, joka mahdollistaa pilvi- ja paikan päällisten resurssien määrittelyn ihmisluettavilla konfiguraatiotiedostoilla, joita voidaan versioida, uudelleen käyttää ja jakaa. Infrastruktuurin säännöstely ja hallinta voidaan hoitaa johdonmukaisella työnkululla kautta infrastruktuurin elinkaaren. Terraform pystyy hallitsemaan matalan tason komponentteja, kuten laskennan, tallennuksen ja verkon resursseja, yhtä hyvin kuin korkeamman tason komponentteja, kuten DNS-merkintöjä ja SaaS-toimintoja. (What is Terraform s.a.)

Terraform luo ja hallitsee resursseja pilvialustoilla ja muissa palveluissa oman ohjelmointirajapintansa kautta. Terraform-tarjoajat mahdollistavat Terraformin toimimaan lähes millä tahansa alustalla tai palvelulla rajapintansa avulla. Terraform-tarjoajat ovat HashiCorpin ja Terraform-yhteisön kirjoittamia resurssityyppejä ja palveluita, jotka ovat vapaasti saatavilla Terraform Registry -palvelusta. Näihin resursseihin kuuluu esimerkiksi AWS, Azure, Google, Cloud Platform, Kubernetes, Helm, Github, Splunk ja DataDog. (What is Terraform s.a.)

Terraformin työnkulku koostuu kolmesta eri vaiheesta: kirjoitus-, suunnittelu- ja käyttövaihe. Kirjoitusvaiheessa määritellään resurssit, jotka voivat olla useampia pilvipalvelun tarjoajia tai palveluita. Suunnitteluvaiheessa Terraform luo suoritussuunnitelman kuvaillen infrastruktuuria, jonka se luo, päivittää tai tuhoaa riippuen olemassa olevasta infrastruktuurista ja konfiguraatiosta. Käyttövaiheessa Terraform toteuttaa ehdotetut operaatiot oikeassa järjestyksessä ottaen huomioon resurssiriippuvuudet. (What is Terraform s.a.)

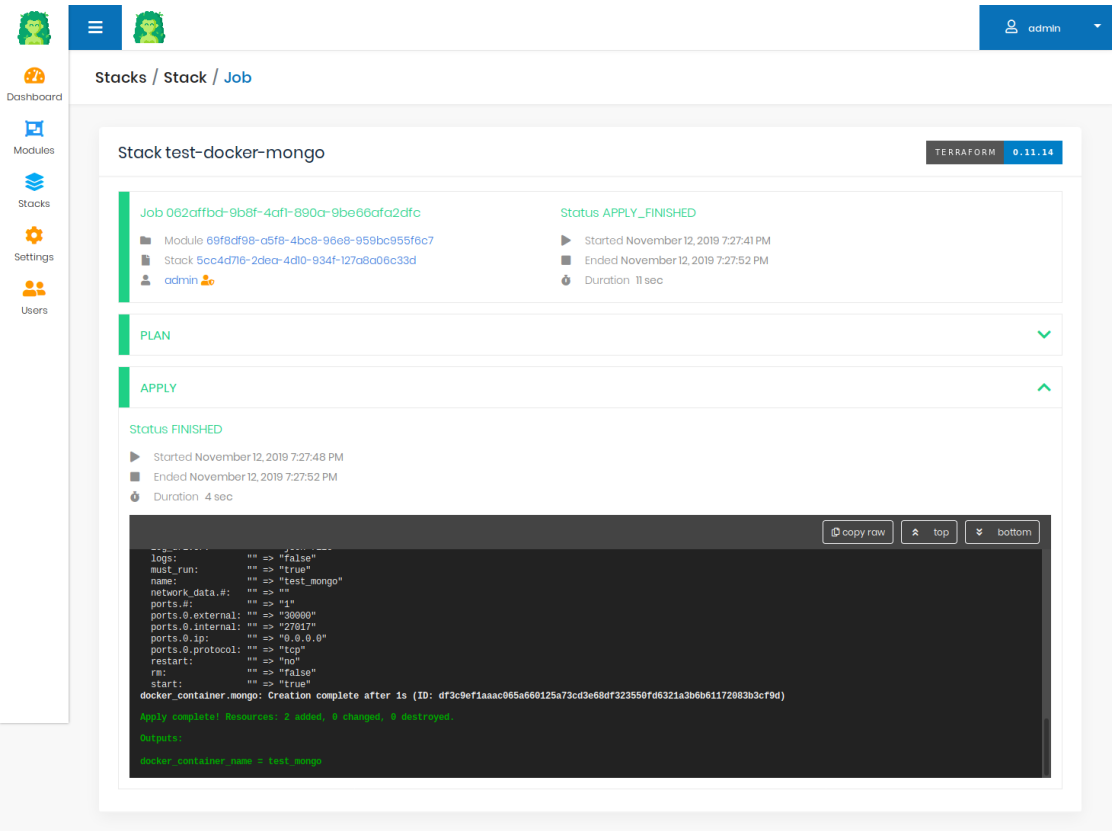
Terraform luo suunnitelman, jonka voi havainnoida listana komentoliittymästä (kuva 7), ja pyytää hyväksymistä ennen muutoksien tekemistä infrastruktuuriin. Se myös pitää kirjaa infrastruktuurista tilatiedostossa (englanniksi state file), joka toimii ympäristön totuuden lähteenä. Terraform käyttää tilatiedostoa määritelläkseen muutokset, jotka tehdä infrastruktuuriin, jotta ne vastaisivat konfiguraatiota. (What is Terraform s.a.)

```
Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_vpc.main will be created
  + resource "aws_vpc" "main" {
      + arn                              = (known after apply)
      + assign_generated_ipv6_cidr_block = false
      + cidr_block                       = "10.0.0.0/16"
      + default_network_acl_id           = (known after apply)
      + default_route_table_id           = (known after apply)
      + default_security_group_id        = (known after apply)
      + dhcp_options_id                  = (known after apply)
      + enable_classiclink               = (known after apply)
      + enable_classiclink_dns_support   = (known after apply)
      + enable_dns_hostnames             = (known after apply)
      + enable_dns_support               = true
      + id                               = (known after apply)
      + instance_tenancy                 = "default"
      + ipv6_association_id              = (known after apply)
      + ipv6_cidr_block                  = (known after apply)
      + main_route_table_id              = (known after apply)
      + owner_id                         = (known after apply)
      + tags                             = {
          + "Name" = "main"
        }
      + tags_all                         = {
          + "Name" = "main"
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.
```

Kuva 7. Terraform-suunnitelma listaa tehtävät muutokset infrastruktuuriin (Clements 2021)

Terraform konfiguraatiotiedostot ovat deklaratiivisia tarkoittaen sitä, että ne kuvaavat infrastruktuurin lopputulosta. Se rakentaa resurssikaavion, joka määrittelee resurssiriippuvuuksia ja luo tai muokkaa ei-riippuvaisia resursseja rinnakkain. Terraform tukee uudelleen käytettäviä konfiguraatiokomponentteja, moduuleja, jotka määrittelevät konfiguroitavia infrastruktuurin kokoelmia. Moduuleja on julkisesti saatavilla Terraform Registrystä tai sitten niitä voi kirjoittaa itse. (What is Terraform s.a.)

Terraformiin on saatavilla kolmannen osapuolen, vapaan lähdekoodin käyttöliittymä Gaia (Gaia documentation s.a.), sekä monia muita visuaalisia työkaluja, kuten Terraformin sisäänrakennettu graph-komento, Blast Radius, Terraform Visual ja Inframap (Machupalli 2021). Kuvassa 8 on työnäkymä Gaia-käyttöliittymästä, josta voidaan tarkastella työnkulkuja ja kirjauksia.



Kuva 8. Gaia-käyttöliittymän työnäkymä (Gaia documentation s.a)

Lisäksi VSCodeen on olemassa HashiCorpin ylläpitämä laajennus (Terraform tools s.a.) sekä Terraformissa on ominaisuudet metriikoihin (Terraform metrics s.a.) ja hälytyksiin (Terraform AWS alarm resource s.a).

3.5 OpenTofu

OpenTofu on Terraformista vastaikään haaroittunut (englanniksi forked), Linux Foundationin ylläpitämä, avoimen lähdekoodin IaC-työkalu. Teknisesti OpenTofu versio 1.6 on hyvin samanlainen ominaisuuksiltaan verrattuna Terraformin versioon 1.6, tosin tulevaisuudessa samankaltaisuudet vähenevät projektien haarojen erkaantuessa. OpenTofu haaroitettiin Terraformista reaktiona HashiCorpin tekemästä muutoksesta Terraformin lisenssiin. OpenTofu on Terraformin version 1.5.x kanssa täysin yhteensopiva, eikä migraatio toisesta

alustasta toiseen vaadi koodimuutoksia toimimisen varmistamiseksi kyseistä versiota käyttäessä. (OpenTofu s.a.)

OpenTofun manifestissa (OpenTofu Manifesto s.a.) ilmaistaan huoli siitä, että Terraformin uusi BUSL (Business Source License)-lisenssi ja siihen liitetyt lisäkäyttöoikeudet ovat tehneet Terraform-riippuvaisesta infrastruktuurista potentiaalisen oikeusriskin kaikille Terraformin käyttäjille. OpenTofu (OpenTofu Manifesto s.a) perustelee, että lisenssin vaihtuminen ilman ilmoitusta tai julistusta etukäteen tuo epävarmuutta Terraformin kehityksen tulevaisuudelle ja että Terraformin uudet käyttöoikeudet ovat liian paljon tulkinnan varaisia.

Harnischin artikkelin (Harnisch 2023) mukaan BUSL eroaa avoimen lähdekoodin lisenssistä ja on välimaasto avoimenlähdekoodin ja loppukäyttäjän lisenssille. Toisin kuin avoimenlähdekoodin lisenssit, BUSL kieltää lisensoidun koodin käyttämistä tuotannossa ilman erillistä hyväksyntää lisenssin laatijalta. BUSL kuitenkin sallii lähdekoodien tarkkailun ja testaamisen testiympäristöissä. (Harnisch 2023.)

4 PILVIORKESTROINNIN TYÖKALUJEN VERTAILU

4.1 Tutkimusmenetelmät

Kvalitatiivisessa eli laadullisessa tutkimuksessa lähtökohtana on moninaisen todellisen elämän kuvaaminen. Kohdejoukko valitaan harkinnanvaraisesti, eikä satunnaisotoksena ja tutkija luottaa omiin havaintoihinsa enemmän kuin mittausvälineillä hankittavaan tietoon. Tapauksia käsitellään ainutlaatuisina ja aineistoa tulkitaan sen mukaisesti ja kohteita tutkitaan mahdollisimman kokonaisvaltaisesti luonnollisessa tilanteessa. Lisäksi laadullisessa tutkimuksessa ei haeta yleistävyyttä populaatioon, vaan pikemminkin käsitteellistä ja teoreettista pitävyyttä. (Bister 2019, 33.)

Laadullinen tutkimus on nimitys monenlaisten tutkimustapojen joukolle. Menetelmät, joissa tutkittavien näkökulmat ja ääni pääsevät esille ovat laadullisessa tutkimuksessa suosittuja tiedonhankinnan menetelmiä. Näihin kuuluu esimerkiksi teemahaastattelu, osallistuva havainnointi, ryhmähaastattelu ja erilaisten dokumenttien ja tekstien analyysit. (Bister 2019, 33.) Myös Bhandarin (Bhan-

dari 2020) artikkelin mukaan laadullisessa tutkimuksessa kerätään ja analysoidaan ei-numeerista aineistoa, joka voi esimerkiksi olla tekstiä, videota, kuvaa tai ääntä.

Tämä tutkimus omaa laadullisen tutkimuksen piirteitä esimerkiksi sen takia, koska aineisto kerätään työkalujen dokumentaatioista tekstin muodossa ja niistä tehdyillä havainnoilla valittu pilviorkestroinnin työkalut voidaan vertailla ja analysoida. Tutkimukseen mukaan otetut pilviorkestroinnin työkalut ovat harkinnanvaraisesti valittu kohdejoukko, ja niistä jokainen työkalu on ainutlaatuinen kohde, jonka piirteisiin perehdytään.

Vertailevassa tutkimuksessa hahmotetaan valittujen tapauksien tai sosiaalisten yksiköiden välisiä yhtäläisyyksiä ja eroja. Vertailun kohteena voivat olla erilaiset tapaukset, jotka on todettu jollain tavoin yhteismitallisiksi ja sen vuoksi vertailukelpoisiksi. (Vertaileva tutkimus 2015.) Tämän tutkimuksen tutkimusmenetelmä on vertailu, sillä vertailuun valittujen pilviorkestroinnin työkalut on tehty saman ongelman ratkaisemiseksi ja niillä on samankaltaisia ominaisuuksia, jolloin ne ovat yhteismitallisia ja siten vertailukelpoisia.

Vertailun avuksi on valmistettu taulukkoja, joista voidaan havaita työkalujen eroavaisuudet työkalun ominaisuuksien puuttumisella tai sisältymisellä. Vertailun myötä voidaan analysoida aiemmin esiteltyjä pilviorkestroinnin työkaluja tarkemmin. Näitä työkaluja on otettu vertailuun mukaan viisi erilaista: Apache Brooklyn, Cloudify, OpenStack Heat, OpenTofu ja Terraform.

Toimeksiantaja on toivonut pilviorkestroinnin työkaluista graafisen käyttöliittymän ja VSCode-integraation sisältymistä sekä pääsyä metriikoihin ja hälytyksiin. Ominaisuudet, joita ei ole erikseen toivottu, mutta sisällytetty silti, ovat yksityis-, hybridi- ja monipilvi- sekä kontitustuki. Toimeksiantaja on myös toivonut yhteensopivuutta tiettyihin pilvipalveluihin ja alustoihin. Näitä ovat esimerkiksi AWS, Azure, Google Cloud, Github ja Kubernetes.

4.2 Työkalujen tukemat ominaisuudet

Ominaisuuksien vertailun avuksi on tehty taulukko, josta voidaan nähdä, mitä ominaisuutta kukin työkalu tukee. Ominaisuudet, joihin pilviorkestroinnin työkaluja verrataan, ovat toimeksiantajan toivomia ominaisuuksia työkaluissa. Lisäksi ominaisuuksien vertailuun on otettu huomioon kyvykkyys kontitukseen ja tuki kolmeen eri pilvipalvelun käyttöönottomalliin: yksityis-, hybridi- ja monipilveen. (Taulukko 1.)

Taulukko 1. Vertailussa olevat pilviorkestroinnin työkalut ja niiden tukemat ominaisuudet

	Apache Brooklyn	Cloudify	OpenStack Heat	OpenTofu	Terraform
Yksityispilvi	X	X	X	X	X
Hybridipilvi	X	X	X	X	X
Monipilvi	X	X	X	X	X
Kontitus	X	X	X	X	X
Metriikat	X	X	X	X	X
Hälytykset	X	X	X	X	X
Graafinen käyttöliittymä	X	X	X	-	X
VSCode-integraatio	-	X	-	X	X

Brooklyn tukee yksityispilveä, hybridipilveä, monipilveä ja kontitusta. Lisäksi työkalussa on pääsy metriikoihin ja hälytyksiin sekä siihen sisältyy graafinen käyttöliittymä. Ainut puute on VSCode-integraatiossa, sillä kyseiseen koodieditoriin ei ole laajennusta Brooklynille. Myös Cloudify tukee yksityis-, hybridi- ja monipilveä sekä kontitusta. Lisäksi se tukee hälytyksiä ja siihen sisältyy graafinen käyttöliittymä sekä VSCode-integraatio. OpenStack Heat tukee myös yksityis-, hybridi- ja monipilveä, kontitusta, metriikoita, hälytyksiä ja siihen on saatavilla graafinen käyttöliittymä. Tosin VSCode-integraatio puuttuu.

Terraform tukee kaikkia taulukossa mainittuja ominaisuuksia. Toisaalta OpenTofuun sisältyy kaikki paitsi graafinen käyttöliittymä. Vaikka OpenTofu (OpenTofu s.a.) on sivustonsa mukaan teknisesti lähes samanlainen Terraformiin verrattuna, Terraformin kolmannen osapuolen käyttöliittymät eivät välttämättä ole yhteensopivia OpenTofun kanssa.

4.3 Työkalujen yhteensopivuus eri alustoihin

Alustojen vertailua varten on tehty taulukko, jossa verrataan pilviorkestroinnin työkalujen yhteensopivuutta eri alustoihin. Toimeksiantaja on toivonut työkaluista tukea AWS:sään, Azureen, Google Cloudiin, Github Actions -alustaan ja Kubernetekseen. (Taulukko 2.) Github Actions -alusta on sivustojensa (Undestanding Github Actions s.a.) mukaan CI/CD-alusta, jolla voidaan automatisoida ohjelmiston rakentaminen, testaaminen ja käyttöönoton työnkulku.

Taulukko 2. Vertailutaulukko pilviorkestroinnin työkalujen yhteensopivuudesta eri alustoihin

	Apache Brooklyn	Cloudify	OpenStack Heat	OpenTofu	Terraform
AWS	X	X	-	X	X
Azure	X	X	-	X	X
Google Cloud	X	X	-	X	X
Github Actions	-	X	-	X	X
Kubernetes	X	X	X	X	X

Apache Brooklyn tukee alustoista AWS:ää, Microsoft Azurea, Google Cloudia ja Kubernetes:ta. Tosin implementaatio Githhub-toiminnoille puuttuu. Myös Cloudify tukee AWS:ää, Microsoft Azurea, Google Cloudia ja Kubernetesia, mutta myös Githubia. Github-toimintojen lisäksi Cloudify tukee muitakin CI/CD integraatioita, kuten Jenkinsiä (Cloudify Jenkins Plugin s.a) ja CircleCI:tä (Cloudify CircleCI s.a). OpenStack Heat tukee taulukosta vain Kubernetesta, eikä OpenStack Heatin tuesta muihin alustoihin ole mainintoja.

Terraform tukee AWS:ää, Microsoft Azurea, Google Cloudia, Githubia ja Kubernetesta. Github-toimintojen lisäksi Terraformissa on myös Github OAuth -

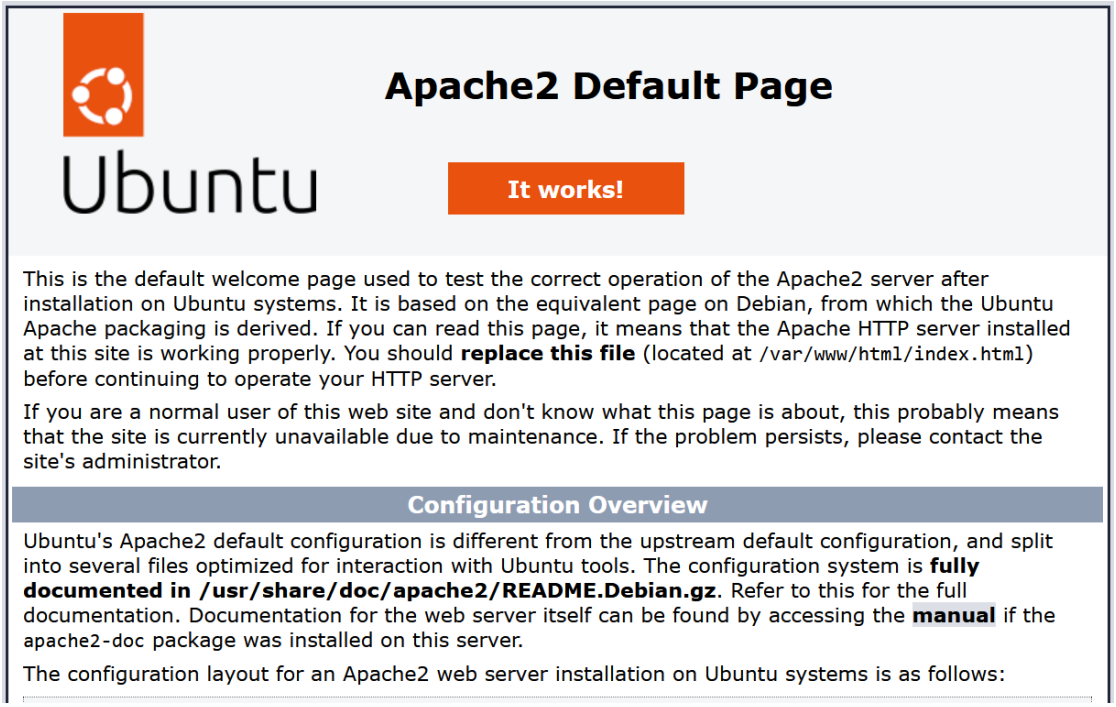
integraatio (Terraform Github OAuth integration s.a). OpenTofu tukee taulukosta kaikkea, mitä myös Terraform tukee, eli AWS:ää, Microsoft Azurea, Google Cloudia, Githubia ja Kubernetesia. OpenTofulle löytyy Github Marketplacen valikoimasta oma Github-toiminto (OpenTofu Github Action s.a).

4.4 Työkalujen kokeilu

4.4.1 Kokeilutapaus

Kokeiluun valitaan kaksi eri pilviorkestroinnin työkalua: Terraform ja OpenTofu, sillä ne ovat pärjänneet vertailussa parhaiten ja ovat ilmaisia testata ja käyttää. Kumpikin työkaluista täyttivät toivotut vaatimukset hyvin kattavasti ja ovat täten parhaita kandidaatteja pilviorkestroinnin ratkaisuun. Nämä kaksi työkalua kokeillaan kokeilutapauksen määrittelemien rajojen sisällä. Kokeilun tuloksena saadaan lisätietoa työkalun kyvyistä, soveltuvuudesta ja käytettävyydestä.

Kokeilun tapauksena on verkkopalvelimen konfiguroiminen, asentaminen ja käynnistäminen julkiseen pilveen. Kokeilutapauksessa tullaan käyttämään Amazonin AWS-pilvialustan ilmaiskokeilua, joka tarjoaa mahdollisuuden käynnistää Ubuntu-virtuaalikoneen AWS-EC2-pilvilaskentainstanssiin. Virtuaalikoneeseen asennetaan Apache2-verkkopalvelinohjelmisto, jolla voidaan isännöidä yksinkertaista HTML-sivua. Kyseinen HTML-sivu tulee olemaan muokattu Apache2-palvelimen oletussivu. Palvelimen oletussivun ulkonäkö voidaan nähdä kuvasta 9.

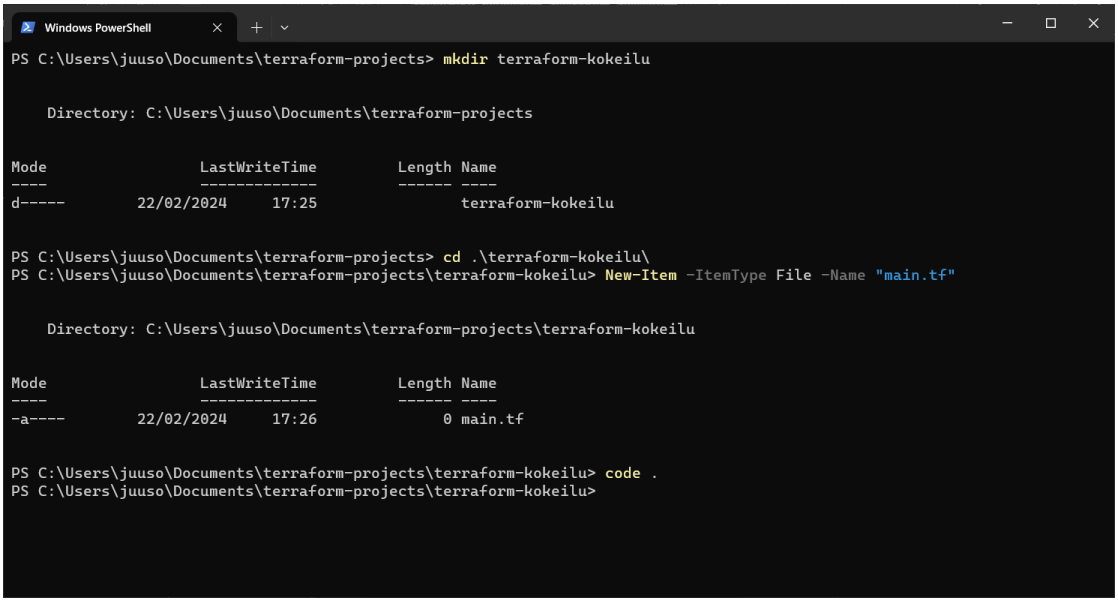


Kuva 9. Ubuntu-virtuaalikoneessa toimiva Apache2-verkkopalvelimen oletussivu

Apache2-verkkopalvelimen oletussivun muokkaaminen tapahtuu kokeiltavan pilviorkestroinnin työkalulla ja tuloksena odotetaan oletussivun sisältävän muokatun otsikon ja työkalun logon Ubuntu-logon sijaan. Työkalun kokeilun hyväksyttäväksi lopputulokseksi kelpaa esimerkiksi seuraava skenaario: Terraformilla luodun verkkopalvelimen sivun otsikossa lukee ”Verkkopalvelin Terraformilla”, esimerkiksi siinä missä kuvassa 9 lukee ”Apache2 Default Page”, ja Ubuntun logon sijalla olisi Terraformin logo. Kyseisen näkymän tulisi avautua, kun EC2-instanssin osoite avataan selaimella.

4.4.2 Terraformin kokeilu

Ensin luodaan projektikansio nimeltä ”terraform-kokeilu” ja luodaan tämän kansion sisään uusi Terraform-tiedosto nimeltä ”main.tf”, johon kirjoitetaan tarvittava infrastruktuuri koodina. Lopuksi voidaan avata projektikansio VSCode-tekstieditorilla, jotta koodit voidaan kirjoittaa tiedostoon ”main.tf”. Tämän prosessin voin havainnoida kuvasta 10.



Kuva 10. Projektikansion ja main.tf-tiedoston luominen komentolinjalla

Sitten muokataan main.tf-tiedostoa ja lisätään AWS-tarjoaja. Tarjoajalle määritellään alueeksi ”eu-north-1” ja salassa pidettävät AWS-avaimet identifikaatiota varten tuodaan erillisistä tiedostoista riveillä 3 ja 4. Lisäksi määritellään VPC (lyhenne Virtual Private Network) rivillä 7. (Kuva 11.)

Kuva 11. AWS-tarjoaja sisältäen AWS-avaimet ja VPC:n määrittely

Seuraavaksi määritellään yhdyskäytävä rivillä 14 ja annetaan sille yllä luodun VPC:n tunniste parametrina. Annetaan sen nimeksi ”gateway-1”. (Kuva 12.)

```
14  resource "aws_internet_gateway" "gateway-1" {
15    vpc_id = aws_vpc.vpc-1.id
16
17    tags = {
18      Name = "gateway-1"
19    }
20  }
```

Kuva 12. Yhdyskäytävän määrittely

Yhdyskäytävän jälkeen määritellään reittitaulukko. Reittitaulukko tarvitsee aiemmin luodun VPC:n tunnisteen parametrina, joten lisätään se rivillä 23. Määritellään IPV4-reitti rivillä 26 ja IPV6-reitti rivillä 31. (Kuva 13.)

```
22  resource "aws_route_table" "route-table-1" {
23    vpc_id = aws_vpc.vpc-1.id
24
25    route {
26      cidr_block = "0.0.0.0/0"
27      gateway_id = aws_internet_gateway.gateway-1.id
28    }
29
30    route {
31      ipv6_cidr_block        = "::/0"
32      gateway_id = aws_internet_gateway.gateway-1.id
33    }
34
35    tags = {
36      Name = "route-table-1"
37    }
38  }
```

Kuva 13. Reittitaulukon määrittely

Sitten määritellään aliverkko osoitteeseen 10.0.1.0/24 rivillä 42 ja annetaan sille VPC:n tunniste rivillä 41. Muodostetaan aliverkon ja reittitaulukon välille assosiaatiotaulukko rivillä 49. (Kuva 14.)

```
40  resource "aws_subnet" "subnet-1" {
41    vpc_id     = aws_vpc.vpc-1.id
42    cidr_block = "10.0.1.0/24"
43
44    tags = {
45      Name = "subnet-1"
46    }
47  }
48
49  resource "aws_route_table_association" "association-1" {
50    subnet_id      = aws_subnet.subnet-1.id
51    route_table_id = aws_route_table.route-table-1.id
52  }
```

Kuva 14. Aliverkon ja assosiaatiotaulukon määrittely

Seuraavaksi luodaan AWS-turvallisuusryhmä nimellä ”allow_tls” rivillä 54. Sitten määritellään turvallisuusryhmään uusi sisääntulosääntö rivillä 64, jossa hyväksytään kaikki tuleva liikenne portista 443 TCP-protokollalla. (Kuva 15.)

```
54  resource "aws_security_group" "allow_tls" {
55    name        = "allow_tls"
56    description = "Allow TLS inbound traffic and all outbound traffic"
57    vpc_id      = aws_vpc.vpc-1.id
58
59    tags = {
60      Name = "allow_tls"
61    }
62  }
63
64  resource "aws_vpc_security_group_ingress_rule" "allow_tls_ipv4" {
65    security_group_id = aws_security_group.allow_tls.id
66    cidr_ipv4         = "0.0.0.0/0"
67    from_port         = 443
68    ip_protocol       = "tcp"
69    to_port           = 443
70  }
```

Kuva 15. AWS-turvallisuusryhmän ja ensimmäisen sisääntulosäännön määrittely

Sitten määritellään lisää sisääntulosääntöjä resursseina, jotta virtuaalikoneeseen voidaan tarvittaessa yhdistää SSH:n avulla. Resurssien määrittelemät säännöt ovat muuten identtiset, lukuun ottamatta avattuja portteja. (Kuva 16.)

```
72  resource "aws_vpc_security_group_ingress_rule" "HTTP" {
73    security_group_id = aws_security_group.allow_tls.id
74    cidr_ipv4         = "0.0.0.0/0"
75    from_port         = 80
76    ip_protocol       = "tcp"
77    to_port           = 80
78  }
79
80  resource "aws_vpc_security_group_ingress_rule" "SSH" {
81    security_group_id = aws_security_group.allow_tls.id
82    cidr_ipv4         = "0.0.0.0/0"
83    from_port         = 22
84    ip_protocol       = "tcp"
85    to_port           = 22
86  }
```

Kuva 16. Lisää sisääntulosääntöjä määritelty resursseina

Sisääntulosääntöjen lisäksi täytyy myös määritellä ulostulosäännöt riveillä 88 ja 94. Sen jälkeen luodaan verkkoliittymä rivillä 100, johon tulee aliverkon tunniste, yksityinen IP-osoite ja aiemmin määritelty turvallisuusryhmän tunniste. (Kuva 17.)

```
88  resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv4" {
89    security_group_id = aws_security_group.allow_tls.id
90    cidr_ipv4         = "0.0.0.0/0"
91    ip_protocol       = "-1"
92  }
93
94  resource "aws_vpc_security_group_egress_rule" "allow_all_traffic_ipv6" {
95    security_group_id = aws_security_group.allow_tls.id
96    cidr_ipv6         = "::/0"
97    ip_protocol       = "-1"
98  }
99
100 resource "aws_network_interface" "network_interface-1" {
101   subnet_id       = aws_subnet.subnet-1.id
102   private_ips     = ["10.0.1.50"]
103   security_groups = [aws_security_group.allow_tls.id]
104 }
```

Kuva 17. Ulostulosääntöjen ja verkkoliittymän määrittely

Konfiguraatiotiedoston loppuun määritellään vielä AWS EIP -resurssi rivillä 106 ja AWS Instance -resurssi rivillä 113. AWS Instance sallii virtuaalikoneen määrittelyn AMI-tunnisteella rivillä 114. Rivillä 123 syötetään skripti erillisestä tiedostosta. (Kuva 18.)

```
106  resource "aws_eip" "eip-1" {
107    domain                    = "vpc"
108    network_interface         = aws_network_interface.network_interface-1.id
109    associate_with_private_ip = "10.0.1.50"
110    depends_on = [ aws_internet_gateway.gateway-1 ]
111  }
112
113  resource "aws_instance" "web-server-instance" {
114    ami = "ami-0014ce3e52359afbd"
115    instance_type = "t3.micro"
116    key_name = "main-key"
117
118    network_interface {
119      device_index = 0
120      network_interface_id = aws_network_interface.network_interface-1.id
121    }
122
123    user_data = file("${path.module}/script.sh")
124    tags = {
125      Name = "web-server"
126    }
127  }
```

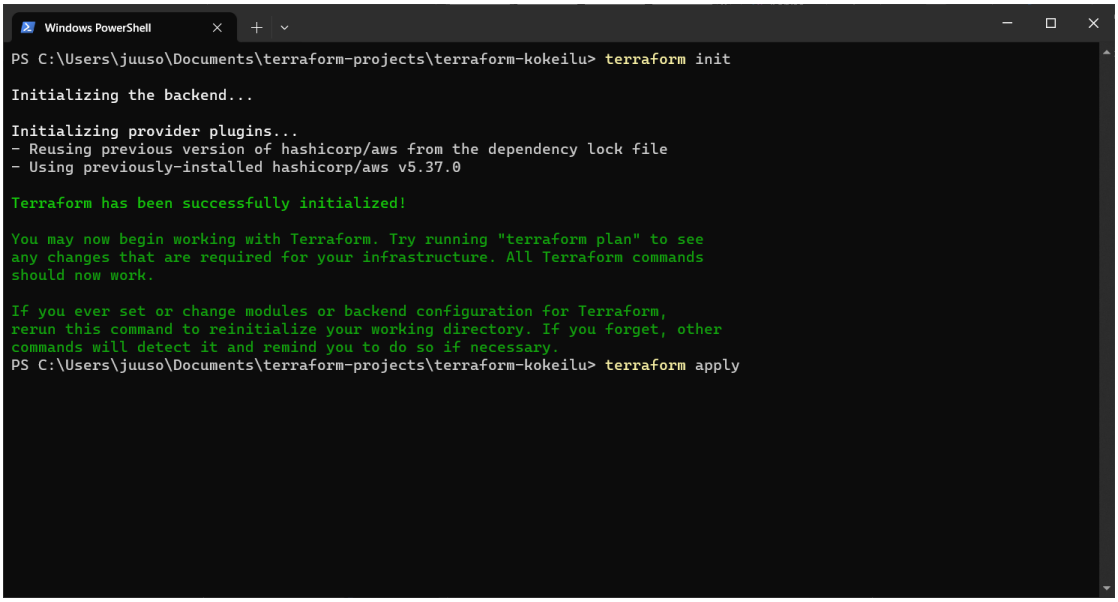
Kuva 18. EIP-resurssin ja AWS-instanssin määrittely

Kuvassa 19 on virtuaalikoneella suoritettava skripti, joka suoritetaan heti virtuaalikoneen käynnistyttyä. Skriptin avulla virtuaalikone ensin päivitetään rivillä 2 ja virtuaalikoneeseen asennetaan apache2-verkkopalvelinohjelmisto, minkä jälkeen se käynnistetään prosessina rivillä 4. Riveillä 6 ja 7 muokataan käynnistetyn verkkopalvelimen oletussivua. (Kuva 19.)

```
1  #!/bin/bash
2  sudo apt update -y
3  sudo apt install apache2 -y
4  sudo systemctl start apache2
5  sudo systemctl enable apache2
6  sudo sed -i 's/Apache2 Default Page/Verkkopalvelin Terraformilla/' /var/www/
   html/index.html
7  sudo sed -i 's/"icons\/ubuntu-logo.png"/"https:\/\/www.datocms-assets.com\/
   2885\/1620155116-brandhcterraformverticalcolor.svg"/' /var/www/html/index.html
```

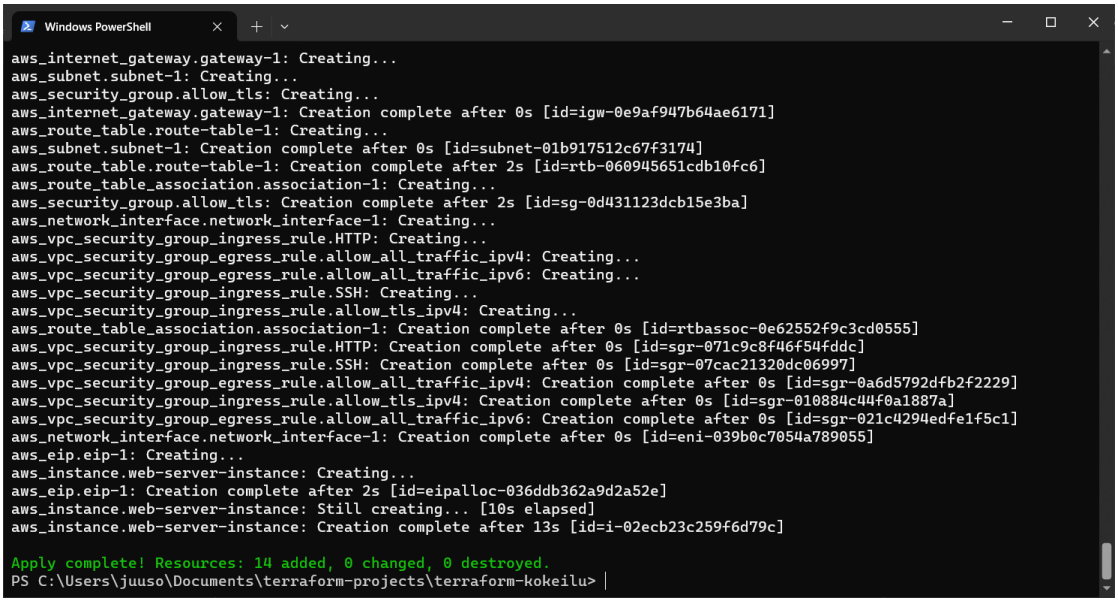
Kuva 19. Virtuaalikoneella ajettava skripti, joka asentaa verkkopalvelinohjelmiston ja päivittää sen oletussivun sisältöä

Lopuksi suoritetaan kaksi komentoa, jotta määritellyt resurssit voidaan ottaa käyttöön AWS-alustalla. Terraform-konfiguraatio initialisoidaan komentoliittymästä ”terraform init”-komennolla, minkä jälkeen käyttö suoritetaan komennolla ”terraform apply”. Kuvasta 20 voidaan nähdä initialisoinnin tuloste ja kuvasta 21 käyttämisen tuloste.



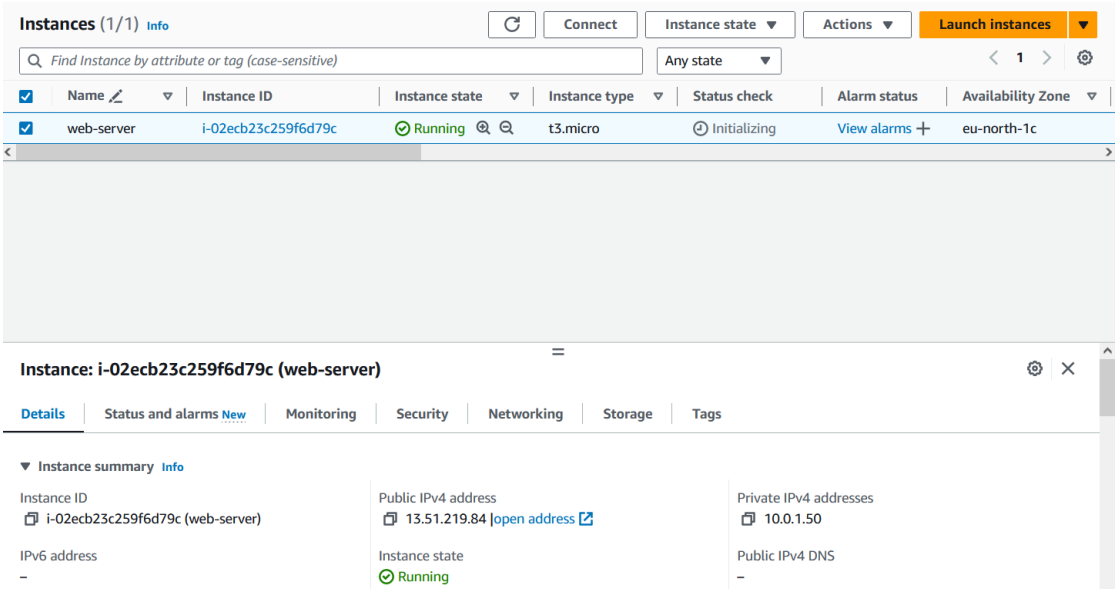
Kuva 20. Terraform-komentojen suorittaminen projektin hakemistossa

Kuvassa 21 käyttökomento onnistui ja tulosteen mukaan 14 uutta resurssia on lisätty. Nyt resurssit ovat otettu käyttöön AWS-alustalle.



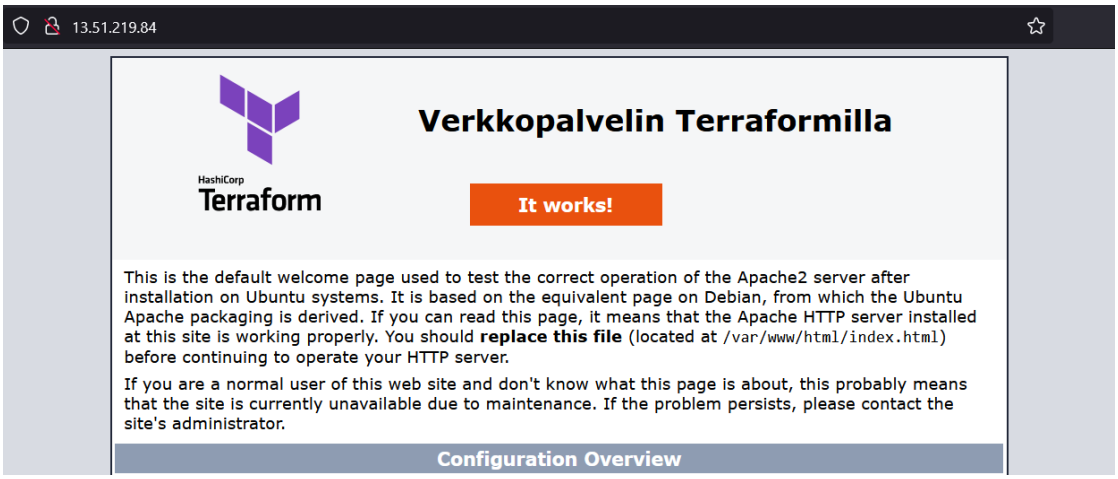
Kuva 21. Apply-komennon onnistunut suoritus näyttää mm. lisättyjen resurssien määrän

AWS-alustan käyttöliittymästä voidaan tarkistaa luotujen instanssien tilan. Juuri luotu instanssi tagilla ”web-server” on käyttöliittymän mukaan käynnissä osoitteessa ”http://13.51.219.84”. (Kuva 22.)



Kuva 22. AWS-alustan Instances-näkymästä voidaan nähdä luodut instanssit ja niiden tiedot

Firefox-selaimella yhdistettäessä osoitteeseen ”http://13.51.219.84” verkkopalvelimen isännöimä sivu avautuu (Kuva 23). Kuvasta 23 voidaan myös havainnoida, että kuvan 19 skripti tekee tarvittavat muutokset oikein.



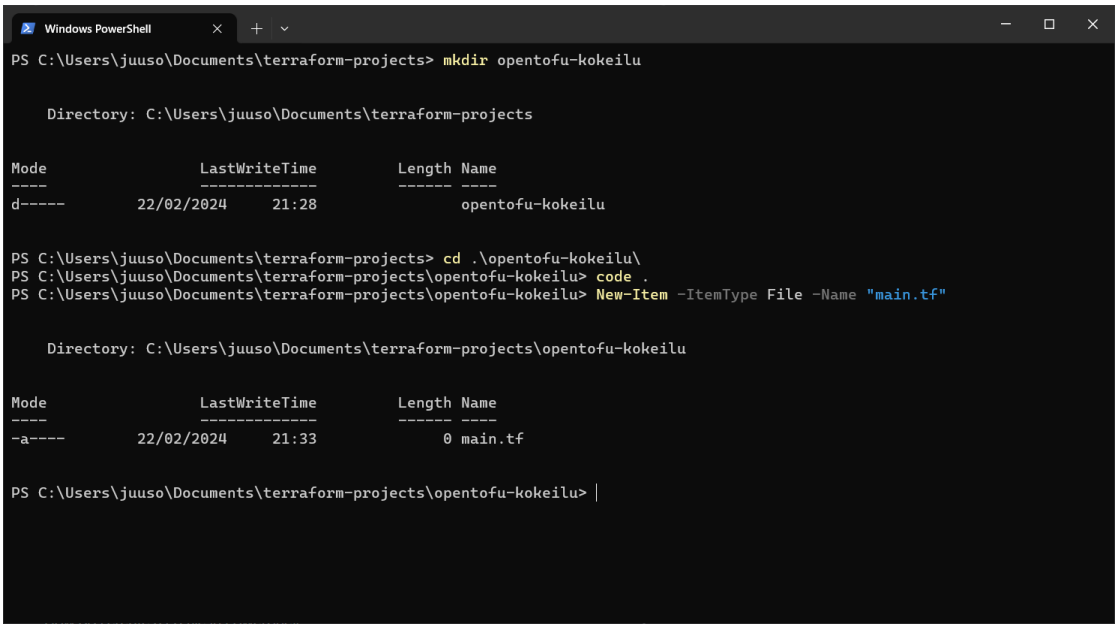
Kuva 23. Verkkopalvelimen sivu avattuna Firefox-verkkoselaimella hakupalkissa näkyvällä IPv4-osoitteella HTTP-yhteydellä

Kokeilu Terraformilla onnistui ja se täytti kokeilutapauksen vaatimukset. Tuloksena verkkopalvelin isännöi sivua, jonka näkymänä on muokattu Apache2-palvelimen oletussivu. Sivusta on muuttunut otsikko ja kuva on vaihtunut Ubuntu-logosta Terraformin logoon, kuten kuvasta 23 voi nähdä.

4.4.3 OpenTofun kokeilu

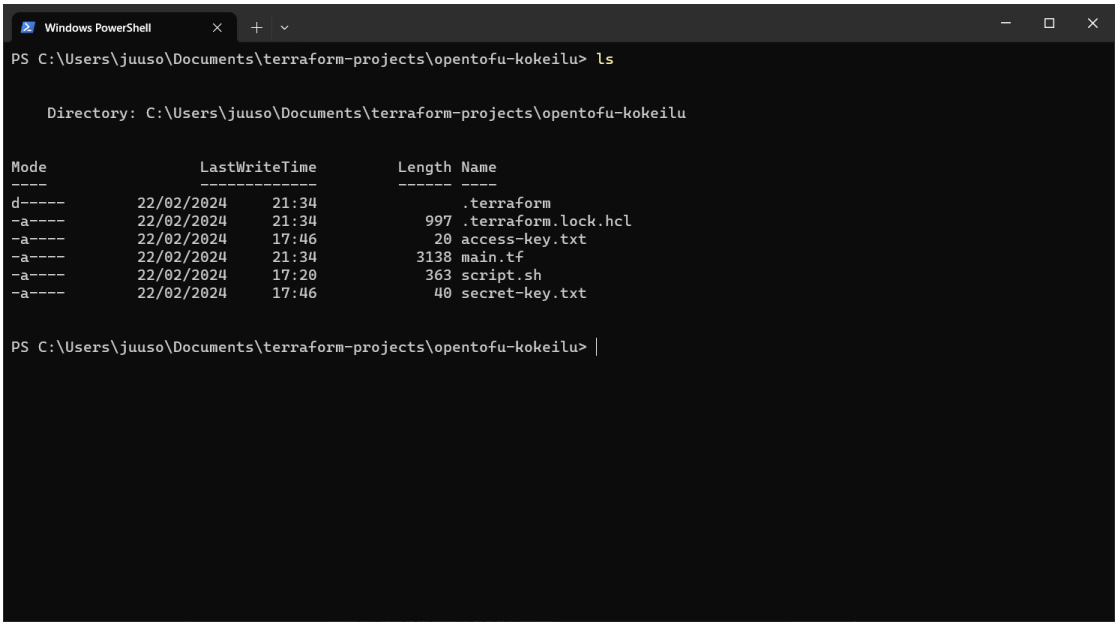
OpenTofu on teknisesti hyvin samanlainen verrattuna Terraformiin, joten aiempia koodeja voidaan hyödyntää uudelleen OpenTofun kokeiluun. Ensin

luodaan uusi projektihakemisto ja luodaan sen sisälle tiedosto ”main.tf”. Sitten avataan projekti VSCodella tiedostojen muokkaamista varten. (Kuva 24.)



Kuva 24. Projektihakemiston ja main.tf-tiedoston luominen

OpenTofun pitäisi pystyä lukemaan Terraform-tiedostoja, joten kopioidaan edellisen projektin tiedostot uuteen projektiin ja kopioidaan sen lisäksi myös edellisen projektin koodit main.tf-tiedostoon. Kuvasta 25 voidaan nähdä projektin hakemiston sisällöt.



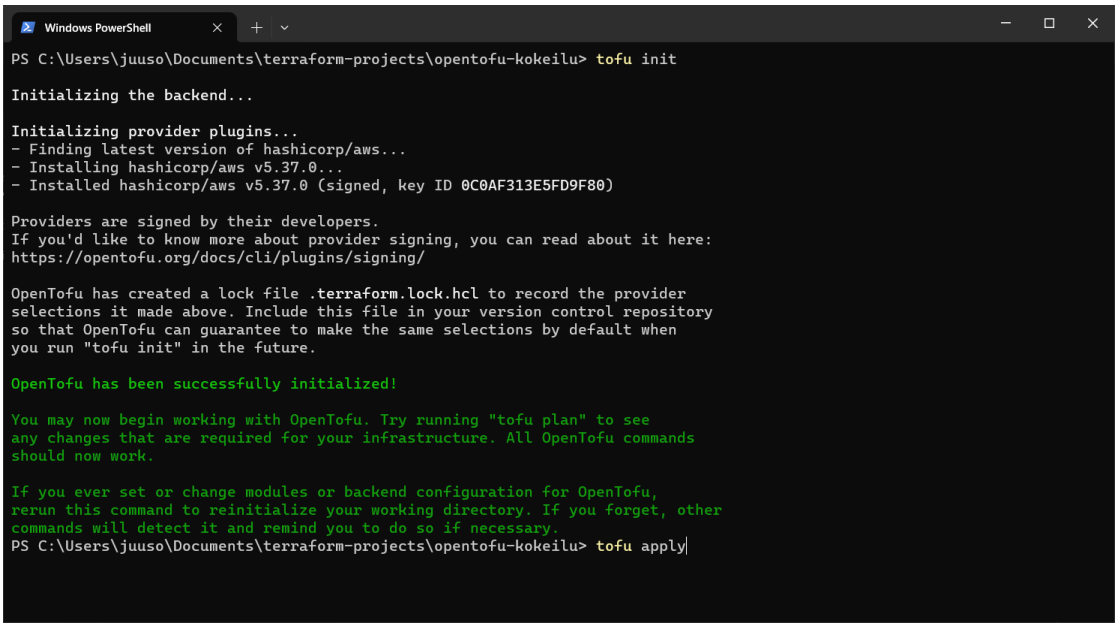
Kuva 25. OpenTofu-kokeiluprojektin hakemiston sisältö

Jotta OpenTofun luoma instanssi erottuu Terraformin instanssista, tehdään pieni muutos AWS-instanssiin ja vaihdetaan siitä tagin nimi rivillä 125. Muut main.tf-tiedoston koodit pysyvät samana kuin edellisessä projektissa. (Kuva 26.)

```
113  resource "aws_instance" "web-server-instance" {
114    ami = "ami-0014ce3e52359afbd"
115    instance_type = "t3.micro"
116    key_name = "main-key"
117
118    network_interface {
119      device_index = 0
120      network_interface_id = aws_network_interface.network_interface-1.id
121    }
122
123    user_data = file("${path.module}/script.sh")
124    tags = {
125      Name = "web-server2"
126    }
127  }
```

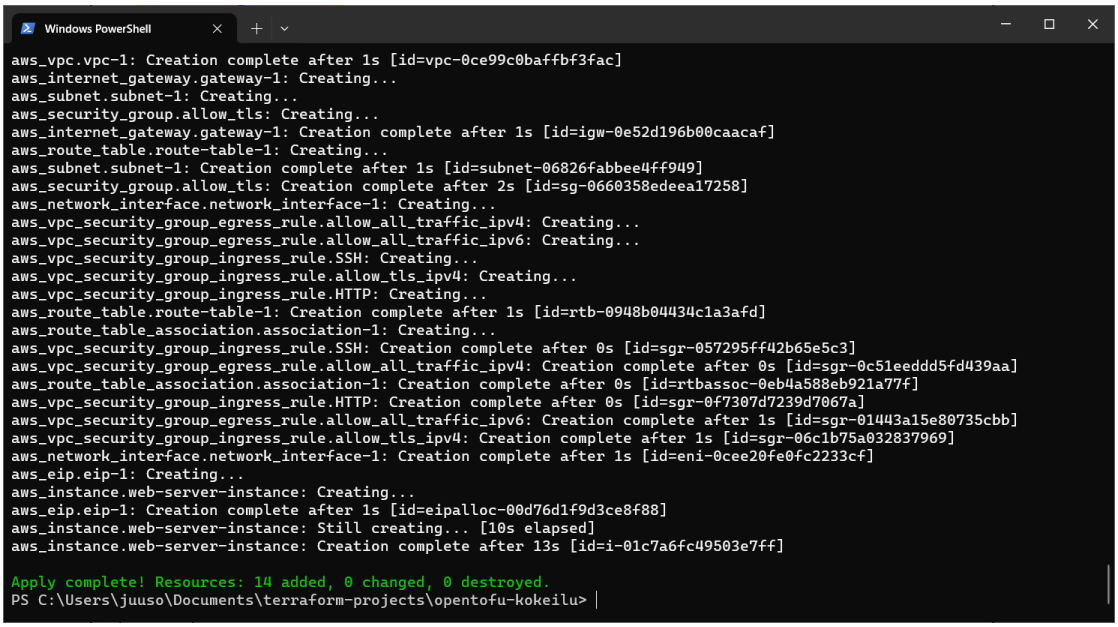
Kuva 26. Koodi muutos main.tf-tiedostoon

Projektin initialisointia varten suoritetaan komento ”tofu init”. Viimeisenä komentona suoritetaan käyttökomento ”tofu apply” samalla tavalla kuin Terraformia käyttäen. (Kuva 27.)



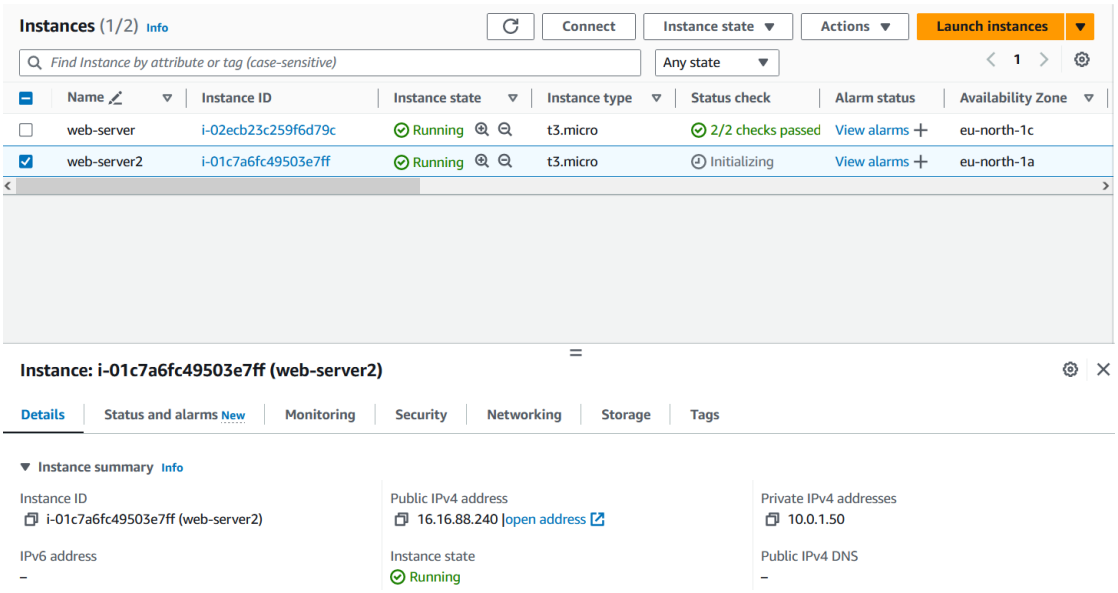
Kuva 27. OpenTofu initialisointi- ja käyttökomento

Kuvassa 28 OpenTofun käyttökomennon tulosteesta voidaan nähdä, että 14 resurssia on lisätty onnistuneesti. Resurssien pitäisi nyt olla AWS-alustalla, kuten koodeissa on määritelty.



Kuva 28. OpenTofun käyttökomennon tuloste

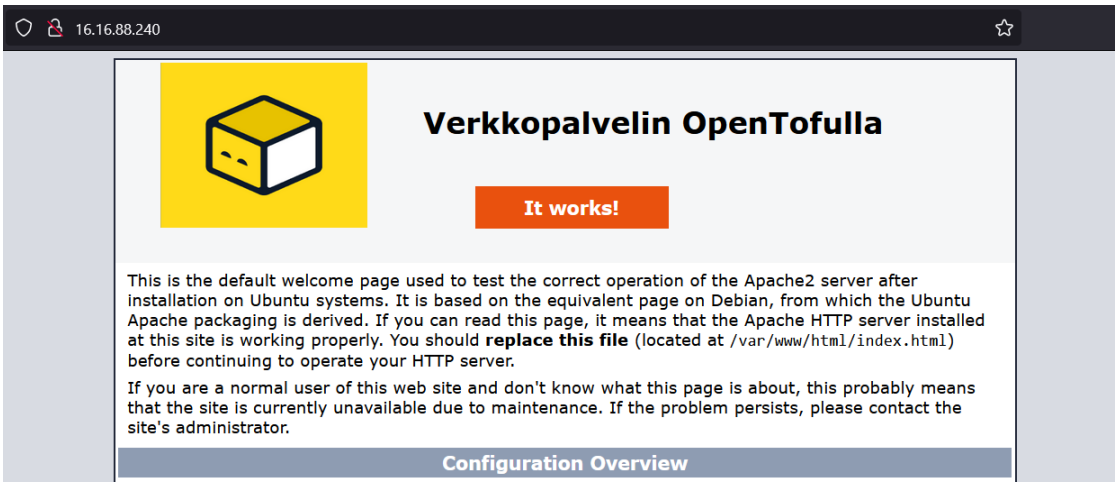
AWS-alustan käyttöliittymästä voidaan tarkistaa luodun instanssin tila (Kuva 29). Kuvasta 29 voidaan havaita, että uusi instanssi tagilla ”web-server2” on käynnistetty ja se on saanut osoitteen ”16.16.88.240”.



Kuva 29. AWS-instanssien näkymään on ilmestynyt odotetusti uusi instanssi

Instanssille sijoitettu IPv4-osoite voidaan nyt avata verkkoselaimella, kun osoitteen kirjoittaa hakukenttään muodossa ”http://16.16.88.240”. Kuvasta 30

voidaan nähdä käynnistyneen verkkopalvelimen sivu avattuna Firefox-selaimessa.



Kuva 30. OpenTofulla luodun verkkopalvelimen sivun näkymä

Pienillä muutoksilla koodeihin saatiin OpenTofun kokeilusta samanlainen tulos. OpenTofu loi uuden AWS EC2 -instanssin Ubuntu-virtuaalikoneella ja Apache2-palvelimen oletussivu on onnistuneesti muokattu näyttämään uuden otsikon ja OpenTofun logon Ubuntun logon sijaan, kuten kuvasta 30 voidaan nähdä.

4.5 Vertailun yhteenveto

Pilviorkestroinnin työkalujen vertailussa olivat mukana Apache Brooklyn, Cloudify, OpenStack Heat, Terraform ja OpenTofu, joista Terraform ja OpenTofu otettiin käytännön kokeiluun. Terraform ja OpenTofu valittiin vertailun jatkoon, eli kokeilun osioon, sillä ne kattavat toimeksiantajan vaatimukset parhaiten ja sen lisäksi ovat ilmaisia käyttää. Cloudify on taulukon 1 ja 2 mukaan kattavin näistä työkaluista, mutta se on maksullinen, minkä takia sitä ei otettu kokeiluun.

Kokeilun tarkoituksena ei ollut todistaa työkalujen tuettujen ominaisuuksien toimivuutta tai pilvialustojen yhteensopivuutta, vaan tutustua työkaluihin syvemmin ja kerätä niistä lisää tietoa. Kun työkalujen käyttäminen ja toiminta on tarkemmin selvillä, voidaan ideoida niiden ympärille uusia osia täydentämään pilviorkestroinnin ratkaisua. Tällaisia osia voisivat olla toiset ohjelmistot tai räätälöidyt liitännäiset.

Terraformin ja OpenTofun kokeilussa kummallakin työkalulla oli sama kokeilutapaus, jonka haluttuna lopputuloksena oli verkkopalvelimen käyttöönotto AWS-alustalla. Tämän halutun lopputuloksen pystyi saavuttamaan kummallakin työkalulla, miltei samoilla koodeilla, mikä vahvistaa OpenTofun (OpenTofu s.a.) omia väitteitä siitä, että OpenTofu on Terraformiin verrattuna yhä hyvin samanlainen toimivuudeltaan.

Toisaalta eroja näiden kahden työkalun välillä on hankala löytää tämän kokeilun pohjalta, mikä ei auta sopivimman työkalun löytämisessä näiden kahden väliltä, jolloin on ehkä tehokkaampaa hyödyntää dokumenttimateriaaleja erojen löytämiseksi jatkossa. Myös kokeilun rajoitettu laajuus vaikutti löytyneiden eroavaisuuksien niukkuuteen.

Voidaan toistaiseksi todeta, että kumpikin työkalu soveltuu yhtä hyvin pilviorkestroinnin ratkaisuun niiden toiminnallisuuksien perusteella. Tosin Terraformin ja OpenTofun olennaisena erona ovat niiden erilaiset lisenssit. Terraform on lisensoitu BUSL-lisenssillä ja OpenTofu on lisensoitu WPL-lisenssillä, jota pidetään aitona avoimen lähdekoodin lisenssinä, toisin kuin BUSL-lisenssiä. Tällä tiedolla on todennäköisesti suurin painoarvo, kun näistä kahdesta työkalusta valitaan yksi tuotantoa varten.

5 PÄÄTÄNTÖ

Ennen opinnäytetyön aloittamista pilviorkestrointi ei ollut entuudestaan tuttu aihe, puhumattakaan pilviorkestroinnin työkaluista. Työn aloitus oli työn hankalin vaihe. Jotta tietäisin, minkälaisia työkaluja minun tulisi etsiä, minun täytyi perehtyä siihen, mitä pilviorkestroinnin saavuttaminen tarkalleen vaatii. Laadukasta kirjallisuutta pilviorkestroinnista oli erityisen hankalaa löytää. En löytänyt aiheesta yhtään kirjaa, joten viitattu sisältö pilviorkestrointiin riippuvat enimmäkseen artikkeleista ja blogipostauksista.

Pilvipalvelut olivat jo jokseenkin tuttu aihe tietojenkäsittelyn koulutuksen ansiosta. Opintojen toisena vuonna sovelluskehityksen opintojaksoilla hyödynnettiin Heroku-pilvialustaa ReactJS-verkkosovellusten käyttöönottoon ja tutustut-

tiin myös Microsoft Azure -alustan toimintaan. Tämä antoi hyvän pohjan perehtyä pilviteknologioihin syvemmin ja kartoittamaan tarkemmin, minkälaista infrastruktuuria verkkosovellukset vaativat.

Opinnäytetyötä tehdessä opin paljon pilviorkestroinnin ja pilviautomatisaation lisäksi pilvipalveluista, pilvipalveluiden käyttöönottomalleista ja niitä hyödyttävistä teknologioista, kuten konteista ja konttiorkestroinnista. Sen lisäksi opin tutkimusmenetelmien ja -strategioiden teoriasta ja hyödyntämisestä. Opinnäytetyön alussa esittelin käsitteitä, jotka olivat tarpeellista pohjaa pilviorkestroinnin ymmärtämiseksi, minkä jälkeen työ siirtyi vahvasti pilvi-infrastruktuurin hallintatyökaluihin. Esittelin näitä työkaluja viisi kappaletta, vertailin niitä ja kokeilin niistä kahta parhaiten vertailussa pärjännyttä: Terraformia ja OpenTofua.

Vertailun ja kokeilun myötä tultiin siihen tulokseen, että Terraform ja OpenTofu toimivat samalla tavalla ja sopivat täten yhtä hyvin tuotantoon pilviorkestroinnin ratkaisuksi. Ainut huomattava ero näiden kahden välillä oli kuitenkin lisenssi: Terraformilla on BUSL-lisenssi, kun taas OpenTofulla on MPL2-lisenssi. Esittelin vertailun tuloksia toimeksiantajalle ja he kiinnostuivat OpenTofusta. He eivät vielä tule ottamaan työkalua käyttöön, mutta aikovat tutkia mahdollisuuksia työkalun kanssa tarkemmin. Sain kuulla, että työni tuloksista tähän on hyvä pohja.

Opinnäytetyö oli projektina suurempi onnistuminen kuin osasin odottaa ja työ eteni enimmäkseen tasaista tahtia. Työkalujen kokeilun sisällyttämistä opinnäytetyöhön ei pidetty pakollisena, vaan se oli pikemminkin asia, jonka tekisin mahdollisuuksien mukaan. Kokeilun puuttuminen ei kuitenkaan olisi haitannut tässä tutkimuspainotteisessa työssä, mutta sillä oli positiivisempi vaikutus kuin osasin odottaa. Loppujen lopuksi olen tyytyväinen tulokseen ja jos tekisin työn uudestaan, niin ehkä lisäisin kokeiluun konttien ja konttiorkestroinnin hyödyntämistä.

LÄHTEET

Apache Brooklyn Creating YAML Blueprints. s.a. Apache Brooklyn. WWW-dokumentti. Saatavissa: https://brooklyn.apache.org/v/1.0.0/blueprints/creating-yaml.html [viitattu 17.1.2024].

Apache Brooklyn Deploying Blueprints. s.a. Apache Brooklyn. WWW-dokumentti. Saatavissa: https://brooklyn.apache.org/v/latest/locations/#named-locations [viitattu 17.1.2024].

Apache Brooklyn Kubernetes-luokkamääritelmä. s.a. Apache Brooklyn. WWW-dokumentti. Saatavissa: https://brooklyn.apache.org/v/latest/misc/javadoc/org/apache/brooklyn/container/location/kubernetes/KubernetesLocation.html [viitattu 17.1.2024].

Apache Brooklyn Locations. s.a. Apache Brooklyn. WWW-dokumentti. Saatavissa: https://brooklyn.apache.org/v/0.12.0/locations/ [viitattu 17.1.2024].

Beal, V. 2011. Cloud Services. Webopedia. Artikkeli. Päivitetty 6.9.2021. Saatavissa: https://www.webopedia.com/definitions/cloud-services/ [viitattu 11.10.2023].

Bhandari, P. 2020. What Is Qualitative Research? | Methods & Examples. Artikkeli. Päivitetty: 22.6.2023. Saatavissa: https://www.scribbr.com/methodology/qualitative-research/ [viitattu 7.2.2024].

Bhuyan, A. 2023. Introduction to Container Orchestration. DZone. Artikkeli. Päivitetty 22.3.2023. Saatavissa: https://dzone.com/articles/introduction-to-container-orchestration-1 [viitattu 9.11.2023].

Bister, T. 2019. Tietojenkäsittelyn opinnäytetyö: Viittoja ja karttoja tutkimisen ja kehittämisen teille. Jyväskylä: Jyväskylän ammattikorkeakoulu. E-kirja. Saatavissa: https://www-booky-fi.ezproxy.xamk.fi/lainaa/1274 [viitattu 6.2.2023].

Clements, K. 2021. Web Age Solutions. WWW-dokumentti. Päivitetty: 21.10.2021. Saatavissa: https://www.webagesolutions.com/blog/how-to-set-up-an-aws-vpc-with-terraform [viitattu 10.3.2024].

Cloudify CircleCI. s.a. Cloudify. WWW-dokumentti. Saatavissa: https://docs.cloudify.co/6.3.0/working_with/integration/circleci/ [viitattu 17.1.2024].

Cloudify Google Cloud Plugin. s.a. Cloudify. WWW-dokumentti. Saatavissa: https://docs.cloudify.co/latest/working_with/official_plugins/infrastructure/gcp/ [viitattu 17.1.2024].

Cloudify Jenkins Plugin. s.a. Cloudify. WWW-dokumentti. Saatavissa: https://docs.cloudify.co/6.3.0/working_with/integration/jenkins-plugin/ [viitattu 17.1.2024].

Cloudify Kubernetes Plugin. s.a. Cloudify. WWW-dokumentti. Saatavissa: https://docs.cloudify.co/latest/working_with/official_plugins/orchestration/kubernetes/ [viitattu 17.1.2024].

Cisco. s.a. What is cloud orchestration? Artikkeli. Saatavissa: https://www.cisco.com/c/en/us/solutions/cloud/what-is-cloud-orchestration.html [viitattu 11.10.2023].

Citrix. s.a. What is private cloud? WWW-dokumentti. Saatavissa: https://www.citrix.com/solutions/app-delivery-and-security/what-is-private-cloud.html [viitattu: 29.10.2023].

Glass, E. What is the Public Cloud? WWW-dokumentti. Saatavissa: https://www.digitalocean.com/community/tutorials/what-is-the-public-cloud [viitattu 5.11.2023].

Cloudflare. s.a. What is multi-cloud | Multi-cloud definition? WWW-dokumentti. Saatavissa: https://www.cloudflare.com/learning/cloud/what-is-multicloud/ [viitattu 14.10.2023].

Gaia documentation. Gaia s.a. Gaia. WWW-dokumentti. Saatavissa: https://docs.gaia-app.io/ [viitattu 11.1.2024].

Harnisch, M. 2023. Business Source License (BSL 1.1): Requirements, Provisions, and History. Fossa. Artikkeli. Päivitetty: 22.8.2023. Saatavissa: https://fossa.com/blog/business-source-license-requirements-provisions-history/ [viitattu 11.1.2024].

Hempstead, Y. 2021. Cloud automation: what it is, use cases and benefits. Contino. Artikkeli. Päivitetty 20.1.2021. Saatavissa: https://www.contino.io/insights/cloud-automation-versus-cloud-orchestration [viitattu 26.10.2023].

Hewlett Packard Enterprise. s.a. What is private cloud? WWW-dokumentti. Saatavissa: https://www.hpe.com/us/en/what-is/private-cloud.html [viitattu 29.10.2023].

Hirway, M. 2018. Hybrid Cloud for Developers : Develop and Deploy Cost-Effective Applications on the AWS and OpenStack Platforms with Ease. Birmingham: Packt Publishing, Limited. E-kirja. Saatavissa: https://ebookcentral.proquest.com/lib/xamk-ebooks/reader.action?docID=5371684 [viitattu: 29.10.2023].

Hurwitz, JS, Kaufman, M, Halper, F, Kirsch, D, & Hurwitz. 2012. Hybrid Cloud for Dummies. Somerset: John Wiley & Sons, Incorporated. E-kirja. Saatavissa: https://ebookcentral.proquest.com/lib/xamk-ebooks/reader.action?docID=818070 [viitattu 3.11.2023].

IBM. s.a. What are containers? WWW-dokumentti. Saatavissa: https://www.ibm.com/topics/containers [viitattu 27.10.2023].

Ihuman, A. 2021. Introducing Container Orchestration with Kubernetes. Me-

dium. WWW-dokumentti. Päivitetty 11.9.2021. Saatavissa: https://medium.com/@Anita-ihuman/introducing-container-orchestration-kubernetes-7fa90e7a5573 [viitattu 9.11.2023].

Ingalls, S. 2007. Container. Webopedia. Artikkeli. Päivitetty 14.06.2021. Saatavissa: https://www.webopedia.com/definitions/container/ [viitattu 23.11.2023].

Kimachia, K. 2023. What is private cloud? Definition, how it works and more. TechRepublic. Artikkeli. Päivitetty 19.7.2023. Saatavissa: https://www.techrepublic.com/article/private-cloud/ [viitattu 29.10.2023].

Kubernetes OpenStack Heat. s.a. Kubernetes. WWW-dokumentti. Saatavissa: http://erictune.github.io/docs/getting-started-guides/openstack-heat/ [viitattu 17.1.2024].

Liu, J. 2020. A beginner's guide to Kubernetes container orchestration. Opensource. Artikkeli. Päivitetty 5.6.2020. Saatavissa: https://opensource.com/article/20/6/container-orchestration [viitattu 9.11.2023].

Machupalli, V. 2021. Tools to visualize your Terraform plan. Dev. Blogi. Päivitetty: 22.4.2021. Saatavissa: https://dev.to/vidyasagarmsc/tools-to-visualize-your-terraform-plan-5g3 [viitattu 11.1.2024].

Maguire, J. 2017. What is a private cloud? The complete guide to private cloud. Datamation. Artikkeli. Päivitetty 1.8.2017. Saatavissa: https://www.datamation.com/cloud/what-is-private-cloud/ [viitattu 29.10.2023].

Maronde, D. 2022. What you need to know about cloud orchestration and automation tools. Redwood. Artikkeli. Päivitetty 15.8.2022. Saatavissa: https://www.redwood.com/article/cloud-orchestration-and-automation-tools/ [viitattu 26.10.2023].

Maksimov, A. 2023. Cloud Orchestration Platforms: A Comprehensive Guide to Tools and Technologies. Hands on Cloud. Blogi. Päivitetty 16.4.2023. Saatavissa: https://hands-on.cloud/cloud-orchestration-platforms [viitattu 8.11.2023].

Murugesan, S & Bojanova, I. 2016. Encyclopedia of Cloud Computing. West Sussex: John Wiley & Sons, Incorporated. E-kirja. Saatavissa: https://ebookcentral.proquest.com/lib/xamk-ebooks/reader.action?docID=4526670 [viitattu 11.10.2023].

Neenan, S & Bigelow, SJ. s.a. What is hybrid cloud? The ultimate guide. WWW-dokumentti. Päivitetty: heinäkuu 2023. Saatavissa: https://www.techtarget.com/searchcloudcomputing/definition/hybrid-cloud [viitattu 3.11.2023].

OpenStack Resource Types. OpenStack s.a. OpenStack. WWW-dokumentti. Saatavissa: https://docs.openstack.org/heat/latest/template_guide/openstack.html [viitattu 10.1.2024].

OpenTofu s.a. OpenTofu. WWW-dokumentti. Saatavissa: https://opentofu.org/ [viitattu 11.1.2024].

OpenTofu Github Action. Github. WWW-dokumentti. Saatavissa: https://github.com/marketplace/actions/opentofu-setup-tofu [viitattu 17.1.2024].

Oracle. s.a. What is multicloud? WWW-dokumentti. Saatavissa: https://www.oracle.com/cloud/multicloud/what-is-multicloud/ [viitattu 14.10.2023].

Perry, Y. 2019. Cloud automation: why, where and how. Netapp. Artikkeli. Päivitetty 31.12.2019. Saatavissa: https://bluexp.netapp.com/blog/cloud-automation-why-where-and-how-cvo-blg [viitattu 26.10.2023].

Redhat. 2023. What is cloud orchestration? Artikkeli. Päivitetty 20.02.2023. Saatavissa: https://www.redhat.com/en/topics/automation/what-is-cloud-orchestration [viitattu 11.10.2023].

Rubens, P. 2017. What are containers and why do you need them? CIO. Artikkeli. Päivitetty 27.6.2017. Saatavissa: https://www.cio.com/article/247005/what-are-containers-and-why-do-you-need-them.html [viitattu 27.10.2023].

Sela, T. 2017. The New Cloudify UI – Features and Tutorial. Clodify. Blogi. Päivitetty: 16.4.2017. Saatavissa: https://cloudify.co/blog/new-cloudify-ui-features-tutorial/ [viitattu 17.1.2024].

Shaik, S. 2018. The importance of Hybrid Cloud. Accurate Pedia. WWW-dokumentti. Saatavissa: https://accuratepedia.blogspot.com/2018/07/importance-of-hybrid-cloud-defination-advantages.html [viitattu 3.11.2023].

Terraform AWS alarm resource. HashiCorp s.a. HashiCorp. WWW-dokumentti. Saatavissa: https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/cloudwatch_metric_alarm [viitattu 11.1.2024].

Terraform Github OAuth integration. HashiCorp s.a. HashiCorp. WWW-dokumentti. Saatavissa: https://developer.hashicorp.com/terraform/tutorials/automation/github-oauth [viitattu 17.1.2024].

Terraform metrics. HashiCorp s.a. HashiCorp. WWW-dokumentti. Saatavissa: https://developer.hashicorp.com/terraform/cloud-docs/agents/metrics [viitattu 11.1.2024].

Terraform tools. HashiCorp s.a. HashiCorp. WWW-dokumentti. Saatavissa: https://developer.hashicorp.com/terraform/docs/terraform-tools [viitattu 11.1.2024].

Terramel, A. 2023. Cloudify VS Code Extension. Cloudify. Blogi. Päivitetty: 26.6.2023. Saatavissa: https://cloudify.co/blog/cloudifys-vs-code-extension/ [viitattu 10.1.2024].

The OpenTofu Manifesto. OpenTofu s.a. OpenTofu. WWW-dokumentti. Saatavissa: https://opentofu.org/manifesto [viitattu 10.1.2024].

The theory behind Brooklyn. Apache s.a. Apache Brooklyn. WWW-dokumentti. Saatavissa: https://brooklyn.apache.org/learnmore/theory.html [viitattu 9.1.2024].

Undestanding Github Actions. s.a. Github. WWW-dokumentti. Saatavissa: https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions [viitattu: 21.1.2024].

Vertaileva tutkimus. 2015. Lähdesmäki, T., Hurme, P., Koskimaa, R., Mikkola, L. & Himberg, T. WWW-dokumentti. Jyväskylän yliopisto, humanistinen tiedekunta. Päivitetty: 23.4.2015. Saatavissa: https://koppa.jyu.fi/avoimet/hum/menetelmapolkuja/menetelmapolku/tutkimusstrategiat/vertaileva-tutkimus [viitattu 30.1.2024].

Venugopal, S. 2016. Cloud orchestration technologies: Explore your options. IBM Developer. Artikkeli. Päivitetty 11.06.2016. Saatavissa: https://developer.ibm.com/articles/cl-cloud-orchestration-technologies-trs [viitattu 11.10.2023].

Vmware. s.a. What is cloud orchestration? WWW-dokumentti. Saatavissa: https://www.redhat.com/en/topics/automation/what-is-cloud-orchestration [viitattu 11.10.2023].

Walker, J. 2023. 10 Most useful cloud orchestration tools & platforms. Spacelift. WWW-dokumentti. Päivitetty 5.10.2023. Saatavissa: https://spacelift.io/blog/cloud-orchestration-tools [viitattu 26.10.2023].

Welcome to Heat Dashboard! OpenStack s.a. OpenStack. WWW-dokumentti. Saatavissa: https://docs.openstack.org/heat-dashboard/latest/ [viitattu 10.1.2024].

Welcome to the Heat documentation! OpenStack s.a. OpenStack. WWW-dokumentti. Saatavissa: https://docs.openstack.org/heat/latest/ [viitattu 10.1.2024].

What is Cloudify? Cloudify s.a. Cloudify Platform Ltd. WWW-dokumentti. Saatavissa: https://docs.cloudify.co/latest/about/what-is-cloudify/ [viitattu 10.1.2024].

What is Terraform? HashiCorp s.a. HashiCorp. WWW-dokumentti. Saatavissa: https://developer.hashicorp.com/terraform/intro [viitattu 11.1.2024].