jamk

Valmiin palvelukokonaisuuden monistaminen AWS CloudFormation mallipohjia käyttäen

Ilmari Metsävainio

Opinnäytetyö, AMK
Marraskuu 2024
Tietojenkäsittelyn tutkinto-ohjelma

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Ilmari Metsävainio

Valmiin palvelukokonaisuuden monistaminen AWS CloudFormation mallipohjia käyttäen

Jyväskylä: Jyväskylän ammattikorkeakoulu. Lokakuu 2024, 38 sivua

Tietojenkäsittelyn tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Joskus tuotekehityksessä on tarpeellista laajentaa jo käytössä olevia palveluita siten, että eri asiakkaiden tiedot ovat eri Amazon Web Service (AWS) käyttäjätileillä. Eri asiakkaiden tietojen jakamiseksi käyttäjätileille on tarpeellista pystyttää monia kopioita tarvittavista palveluista AWS käyttäjätileille. Näiden kopioiden luomisen vaatiman työn helpottamiseksi käytetään Amazon Web Services CloudFormation -palvelua. Tässä opinnäytetyössä tarkastellaan Amazon Web Services CloudFormation -palvelun toimintaa.

Opinnäytetyön toimeksiantajana toimi CareUS OY, joka kehittää tuotetta, joka auttaa fysioterapeutteja toimimaan heidän potilaidensa kanssa. Työn tarkoituksena oli kehittää AWS CloudFormation -palvelua hyödyntäen mallipohja, jota käyttämällä CareUS OY voi pystyttää tarvitsemansa palvelut uusilla AWS käyttäjätileillä. Lisäksi tehtävänä oli tutkia, miten AWS CloudFormation -mallipohjat toimivat, miten niitä luodaan, ja miten AWS CloudFormation -designer toimii tässä käyttötarkoituksessa. Opinnäytetyö toteutettiin tutkimuksellisena kehitystyönä.

Opinnäytetyön tuloksena syntyi AWS CloudFormation mallipohja, jota käyttämällä CareUS OY voi pystyttää haluamansa palvelut uusiin AWS-käyttäjätileihin. Työn kautta myös syntyi ymmärrys siitä, miten AWS CloudFormation -mallipohjia voidaan luoda ja muokata tarpeen tullen. Työ vastasi myös tarpeiseen saada tietoa siihen, miten AWS CloudFormation -designeria käytetään.

Työn ansiosta toimeksiantaja voi nyt luoda mallipohjalla kopioita palveluista sekä muokata mallipohjaa. Koska opinnäytetyön tarkoitus oli ratkaista juuri CareUs OY:n ongelma, sen tulokset eivät todennäköisesti yleisty täysin esteettömästi muihin ympäristöihin. Kuitenkin työn tuloksia voi silti käyttää apuna samankaltaista työtä tehdessä.

Avainsanat (asiasanat)

AWS, AWS CloudFormation, Amazon Web Services, tutkimuksellinen kehitystyö, CloudFormation Designer, IAC, CloudFormation template, CloudFormation mallipohja

Muut tiedot (salassa pidettävät liitteet)

Opinnäytetyö ei sisällä salassa pidettävää tietoa.

jamk

Description

Metsävainio Ilmari

Cloning of a ready service package using AWS CloudFormation templates

Jyväskylä: JAMK University of Applied Sciences, October 2024, 38 pages

Degree Programme in Web & Data. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

During product development, it is sometimes necessary to scale services so that different customers have their data in completely different AWS accounts. This requires setting up multiple copies of the same services on multiple AWS accounts. This is made easier by using AWS CloudFormation. This study explored the use of AWS CloudFormation for this purpose.

The thesis was commissioned by CareUS Oy in the spring of 2023. The company is developing a product to help physiotherapists work with their patients. The aim of the thesis was to develop AWS CloudFormation templates that, when used, would create copies of services already used by the company on its current single AWS account, and then use these templates to set up these services on other accounts. It also aimed to gain a better understanding of how AWS CloudFormation is used. The thesis was conducted as development research.

As a result of the thesis an AWS CloudFormation template was created that set up the correct services. Also, as a result knowledge of how AWS CloudFormation works and how it is used to create templates was gained.

The commissioner can now create copies of their services and modify the template as need arises. As the thesis was specifically designed to solve the problem of CareUs OY, it is unlikely that the results can be fully generalised to other environments. However, the results of the work can still be used as a help when doing similar work.

Keywords/tags (subjects)

AWS, AWS CloudFormation, Amazon Web Services, Development research, CloudFormation Designer, IAC, CloudFormation template

Miscellaneous (Confidential information)

Thesis contains no confidential information.

Sisältö

Kuviot

Lyhenteet ja termit

- IaC = Infrastructure as Code (suomeksi infrastruktuuri koodina).
- AWS = Amazon Web Services
- AWS CloudFormation template = AWS ,mallipohja
- User pool = käyttäjäryhmä
- Identity pool = identiteettiryhmä
- Policy = Käytänne (esim. inline policy = inline -käytänne, managed policy = hallittu käytänne)

1 Johdanto

Nykyään yritykset käyttävät pilvipalveluita yhä enemmän paikallisten palvelimien sijasta, mutta vaikka pilvi-infrastruktuurin käyttöönotto ja ylläpito on helpompaa ja nopeampaa kuin perinteisten paikallisen infrastruktuurin käyttöönotto ja ylläpito, sen manuaalinen hallinnointi on silti monimutkaista. Haasteita syntyy esimerkiksi siitä, että eri ohjelmien versiot muuttuvat, niihin tehdään muutoksia, ja niitä on käytössä monessa eri paikassa ja palvelussa. Jotta niistä kaikista pysytään perillä, tulisi niistä pitää ajantasaista dokumentaatiota, tai palveluiden hallinta menee helposti sekaisin. Jos palvelua muutetaan, kuten esimerkiksi tapahtuu otettaessa käyttöön uusi palvelin nousseen käyttäjämäärän takia, joudutaan päivittämään dokumentaatiota. Jollei tätä tehdä, ei enää helposti tiedetä, missä tilassa mikäkin palvelu on sillä hetkellä. Tämän ongelman välttämiseksi IaC:tä eli infrastruktuuria koodina käyttävien yritysten määrä kasvaa. IaC-teknologian avulla pysytään aina tietoisena siitä, missä tilassa pystytetyt palvelut ovat, kuinka paljon ja kuinka tehokkaita palvelimia on käytössä, mitä tietokantainstansseja on olemassa ja niin edelleen. Myös muutosten tekeminen on helpompaa, kun kaikki infrastruktuurin tiedot löytyvät samasta paikasta ja muutos tehdään vain muuttamalla koodia. Koodin sijainti versionhallintajärjestelmässä myös mahdollistaa koodin muutosten näkemisen versiohistoriassa ja tarvittaessa mahdollistaa vanhempiin versioihin palaamisen, jos uudessa versiossa esiintyy jokin vika. IaC:n hyvä tunteminen on tärkeää yrityksille, koska näyttää siltä, että IaC:n käytön kasvu jatkuu tulevaisuudessa, sillä sen hyödyt ovat niin suuret. Esimerkiksi Precedence Researchin, Kanada/Intia -pohjaisen markkinatutkimusta tekevän yhtiön Shivani Zotingin (2024) ennustaa raportissaan, että IaC:n vuoden 2024 maailmanlaajuinen markkina-arvo 1,06 miljardia dollaria tulee seuraavan kymmenen vuoden aikana, vuoteen 2034 mennessä, nousemaan 9,40 miljardiin dollariin (Infrastructure as Code Market Size, Share and Trends 2024 to 2034. 2024).

2 Lähtökohdat

2.1 Toimeksiantaja ja tavoite

Työn toimeksiantajana toimi vuonna 2020 (Kaupparekisterissä 2021) perustettu CareUS Oy. Yhtiö kehittää laitetta, jonka avulla fysioterapeutti voi seurata potilaan tekemiä kuntoutusliikkeitä etänä sekä hallita eri kuntoutusohjelmia. CareUS Oy on vielä pieni yritys, jolla on ainoastaan muutama työntekijä, mutta he ovat löytäneet jo potentiaalisia asiakkaita ja työskentelevät heidän kanssaan kehittäessään heidän laitetta ja sovellusta.

CareUs Oy oli kehittänyt toimivan prototyypin laitteensa käyttämästä sovelluksesta, mutta tämän opinnäytetyön alkaessa heillä oli ainoastaan yksi AWS-tili, jolle oli pystytetty sovelluksen käyttämät palvelut. CareUS Oy:n tavoitteena oli monistaa helposti heidän valmistamansa palvelukokonaisuutta niin, että asiakkaiden tiedot eivät näkyisi muille asiakkaille. Tämän mahdollistamiseksi työssä kehitettiin AWS CloudFormation -mallipohjia, joilla uuden AWS-tilin luodessa voidaan helposti pystyttää kaikki tarvittavat palvelut mahdollisimman vähällä manuaalisella työllä.

Työn tavoitteiden saavuttamiseksi otettiin kaksi tutkimuskysymystä:

- Miten valmista AWS-palvelukokonaisuutta voidaan laajentaa AWS CloudFormationilla?
- Miten AWS CloudFormationin IaC (Infrastructure as Code) mallipohjat toimivat?

IaC:n käyttöä varten on olemassa monia palveluita, jotka käsittelevät infrastruktuuria hallinnoivaa koodia. Työssä keskityttiin erityisesti Amazon Web Services:in kehittämään AWS CloudFormationiin, koska CareUs Oy käyttää omassa sovelluksessaan AWS-palveluita. CloudFormation on IaC-palvelu, joka on osa Amazonin Amazon Web Services -palveluja. Sen avulla voidaan helposti hallinnoida eri AWS-palveluja.

2.2 Menetelmät

Tämä opinnäytetyö on tutkiva kehitystyö, koska siinä pyritään löytämään ratkaisu tiettyyn työnantajan käytännön ongelmaan. Timo Bister kirjoittaa (2019) kirjassaan ”Tietojenkäsittelyn

opinnäytetyö: viittoja ja karttoja tutkimisen ja kehittämisen teille” kehitystyön taustoituksesta: ”Varsinaisen tutkimuskohteen kehittämiseen liittyvää työtä edeltää tutkimuskohteen olemusta ja luonnetta kartoittava tiedon keruu. Tutkijan tulee perehtyä toimintaympäristöön ja ilmiöön mahdollisimman tarkasti, jotta syntyy syvällinen ymmärrys kohteesta ja mahdollisuuksista vaikuttaa asioiden tilaan.” (Bister 2019, 44.) Tässä työssä tätä ohjetta on noudatettu siten, että ennen työn toteutuksen kuvausta ja sen toteutuksen pohtimista kappaleissa 6, 7, ja 8, kappaleissa 4 ja 5 käydään läpi teoriaosassa CloudFormationin toimintaa, eri AWS palveluja, sekä IaC-periaatteita.

Työssä käsitellään AWS CloudFormation:ia sekä muita AWS-pohjaisia palveluita. AWS sisältää satoja sivuja dokumentaatiota, joten tiedon kerääminen eri palveluiden toiminnasta ei ole vaikeaa. Koska työssä tehtiin mallipohjia, joilla voitiin monistaa jo olemassa olevia palveluita, piti tietää myös yksityiskohtaisesti kaikki monistettavien palveluiden asetukset, jotta mallipohjat vastasivat täydellisesti valmista palvelua.

Työssä käytettiin AWS CloudFormation:in sisältämää editoria, jolla voi luoda CloudFormation mallipohjia. Mallipohjia voisi luoda myös millä tahansa muulla koodieditorilla, mutta tähän työhön oli valittu CloudFormation editori. Valinta perustui siihen, että CloudFormation editori sisältää monia tapoja helpottaa mallipohjien luontia ja sallii esimerkiksi lukuisia aloituspohjia eri palveluiden mallipohjille, sekä linkkejä suoraan eri palveluiden CloudFormation -ohjeisiin. Editorilla voi myös helposti testata mallipohjaa, jolloin AWS yrittää pystyttää mallipohjan mukaan palveluita. Mahdollisesta epäonnistumisesta saa tällöin selville, missä on ongelma ja vastaavasti onnistuneesti pystytettyä palvelua voi tarkastella ja tutkia, onko se halutunlainen ja vastaa tarkoitustaan.

Koska työssä monistettiin jo olemassa olevia palveluita, oikeanlainen mallipohja saatiin selville tutkimalla alkuperäisiä palveluita, ja kopioimalla niistä muistiinpanoja, jotka sisältävät: nimet, asetukset, ja muut muuttujat. Muistiinpanojen pohjalta tehtiin tämän jälkeen editorissa lopulliset mallipohjat. Työssä mallipohjat tehtiin uudella erillisellä AWS-tilillä ja alkuperäiset palvelut sijaitsivat eri AWS-tilillä, ettei mallipohjan testaaminen voinut vahingossakaan vaikuttaa jo käytössä oleviin palveluihin.

2.3 Aineiston keräys ja tietoperustan tiedonhaku

Opinnäytetyön tiedonkeruu lähti Amazon Web Servicesea koskevan tiedon etsimisestä. Amazon Web Services sisältää massiivisen määrän dokumentaatiota, joten tiedon keruu ei ollut vaikeaa. Jokaisesta AWS-palvelusta on olemassa sivuttain sekä pinnallista, että yksityiskohtaista tietoa. Amazon on tuottanut myös faktuaalisia julkaisuja heidän palveluistaan, kuten ”Introduction to DevOps on AWS - AWS Whitepaper” jossa kerrotaan DevOpsista ja Infrastruktuurista koodina AWS-alustalla. (Mansoor Ym. 2014)

Taustaksi ja tehtävän hahmottamiseksi tarvittiin tietoa myös yleisemmin infrastruktuurista koodina, ja tietoa tästä lähdettiin etsimään janet.finna.fi sivulta eli Jyväskylän Ammattikorkeakoulun Verkkokirjastosta. Sieltä taustatietoa löytyi hyvin esimerkiksi kirjasta ”Introduction to Infrastructure as Code: A Brief Guide to the Future of DevOps” (Sneh, Thakurta, 2022), joka löytyi hakemalla ”Infrastructure as code” -haulla.

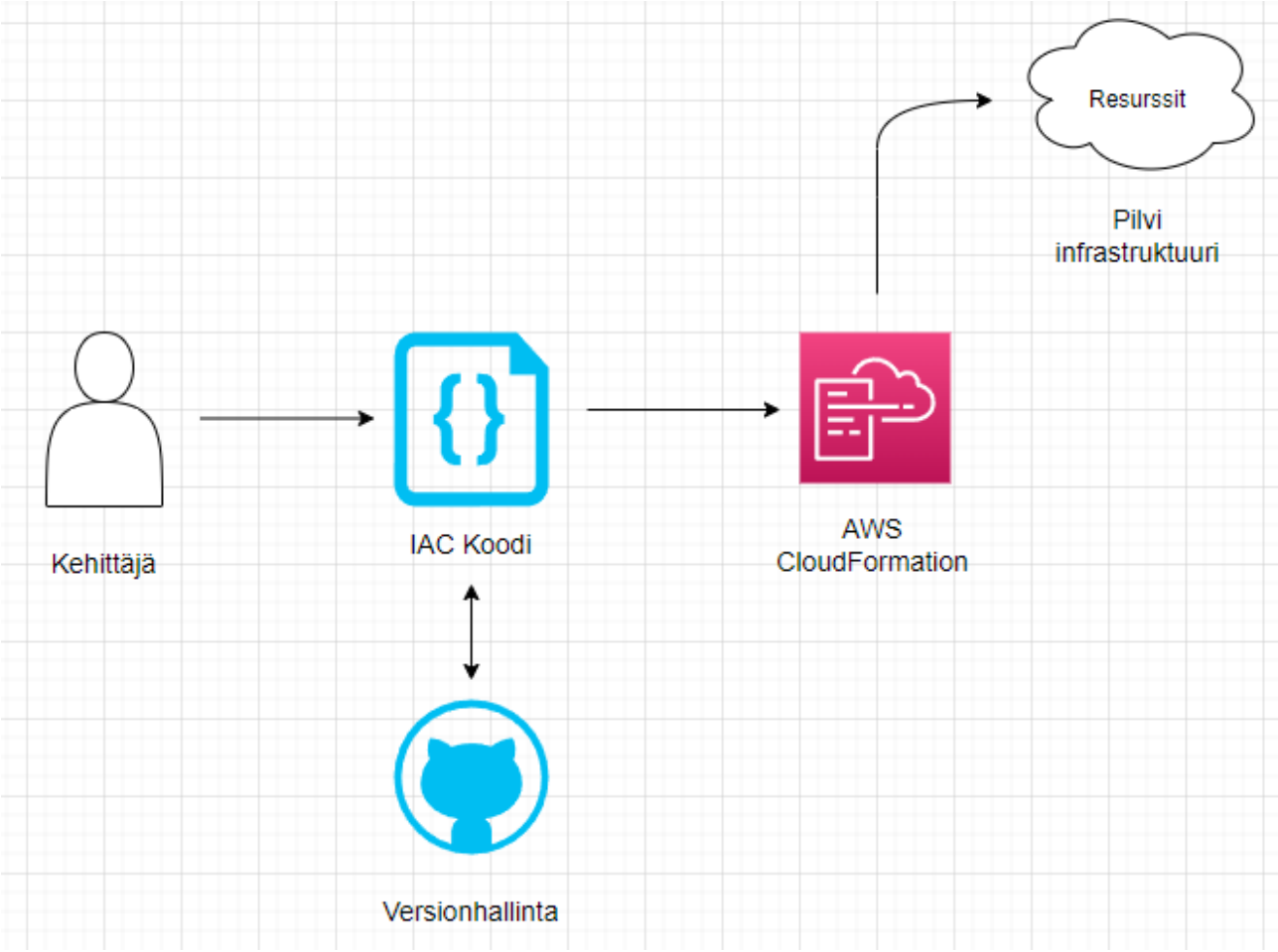
Kehitystyön prosessin hahmottamiseen löytyi tietoa, kun opinnäytetyön ohjaaja kertoi Timo Bisterin (2019) kirjasta ”Tietojenkäsittelyn opinnäytetyö: viittoja ja karttoja tutkimisen ja kehittämisen teille”.

3 Infrastruktuuri koodina

3.1 IaC:n tarkoitus

Infrastruktuuri koodina (jatkossa tekstissä käytän lyhennettä ”IaC”, joka tulee englannista ”Infrastructure as Code”) on tapa hallita ja ottaa käyttöön tarvittava infrastruktuuri palveluille koodia käyttämällä manuaalisten prosessien sijaan. Siinä käytetään tiedostoja, jotka sisältävät mallipohjan, joka kertoo infrastruktuurinhallintaohjelmalle, mitä palveluita ja resursseja otetaan käyttöön. Mallipohjia voi rakentaa joko itse tai käyttää valmista mallipohjaa, jos sellainen löytyy. (Mansoor Ym. 2014, 10)

3.2 IaC osana DevOpsia



Kuva 1. Malli siitä, miten IaC toimii osana DevOpsia AWS CloudFormationia käyttämällä.

DevOps tarkoitaa yhdistelmää käytänteistä ja tekniikoista, joiden tavoitteena on tehdä softan kehittämistyöstä nopeampaa ja tehokkaampaa. DevOps nimi tulee yhdistelmästä sanoista ”software development” ja ”IT operations”. Automaatio koodin testauksessa sekä automaatio uuden koodin pystytyksessä ovat keskeisiä osia DevOpsissa (Sneh, Thakurta, 2022, luku 1).

Kymmenen vuotta sitten, ohjelman backend:in pystytys ja huolto oli monimutkainen ja aikaa vievä prosessi, kun kaikki paikalliset laitteet piti virittää oikein itse, mutta nykyään IaC tekee eri backend -palveluiden pystyttämisestä ja hallinnoimisesta nopeaa ja helppoa. Backend-kehittäjien ei enää tarvitse manuaalisesti pystyttää palveluita, vaan he voivat kirjoittaa koodia, kuten DevOps-tiimi.

Kun koodi on kirjoitettu, käytössä oleva automaatio-ohjelma, kuten Terraform, Kubernetes, Ansible, tai AWS CloudFormation pystyttää kaikki tarvittavat palvelut halutuilla asetuksilla (Sneh, Thakurta, 2022, luku 1).

Yksinkertaisesti sanottuna frontend:in osat kehitetään ja pystytetään helposti DevOpsia käyttäen ja backend -palvelut IaC:n avulla. Kun kehitettävän palvelun kaikki osat toimivat tällä tavalla, sujuu ohjelmiston kehitys nopeasti ja uusien ominaisuuksien testaaminen ja lisääminen sujuu nopeasti. (Sneh, Thakurta, 2022, luku 1).

Käytettäessä IaC:tä tällä tavalla, kuuluu IaC -koodi tallentaa versionhallintajärjestelmään. Tallentamalla IaC-koodi versionhallintajärjestelmään vältetään vaikeuksia, koska jos koodissa ilmenee ongelma se saattaa aiheuttaa ongelmia koko palveluun. Ongelmatilanteessa on tarpeellista palauttaa koodi aikaisempaan, toimivaksi todettuun tilaan.

3.3 IaC ilman DevOpsia

Infrastruktuuria koodina voi käyttää myös yksinkertaisemminkin. Ei ole pakko pystyttää systeemiä, joka automaattisesti päivittää tai pystyttää uusia palveluita, kun tapahtuu muutoksia. AWS CloudFormation sallii myös täysin manuaalisten mallipohjien käytön. Tässä työssä käytetään juuri tätä tapaa eli manuaalisten mallipohjien käyttöä, sillä toimeksiantajalla ei ole käytössä järjestelmää, joka tekisi automaattisesti muutoksia. Manuaalisten mallipohjien käyttöön liittyy kuitenkin monia huonoja puolia. Esimerkiksi, jos mallipohjaa käyttää viidellä eri AWS-tilillä ja sitten mallipohjaan tehdään muutoksia, pitää manuaalisesti mennä tililtä tilille käyttämässä päivitettyä mallipohjaa, että kaikki tilit olisivat ajan tasalla. Automatisoidussa ympäristössä nämä päivitykset olisivat automaattisia ja vaatisivat vain murto-osan manuaalisesta työstä.

3.4 IaC:n hyödyt

Sneh ja Thakurta (2022, luku 1” Benefits Adapting Infrastructure as Code”) kertovat IaC:n hyödyistä. He kuvaavat, miten IaC nopeuttaa suuresti eri palveluiden pystyttämistä sekä skaalausta. IaC myös luonnostaan kodifioi ja jättää tallenteen menneistä versioista ja helpottaa vanhan infrastruktuurin päivittämistä.

IaC pitää huolen siitä, että joka kerta kun koodilla pystytetään ympäristö, se on samanlainen. Tämä estää ns. ”Drifting Configurations”- ongelman, jossa kehitys, testaus, ja käyttöympäristö ajan kuluessa eroavat toisistaan, kun niihin tehdään muutoksia ja päivityksiä epätasaisesti (Sneh, Thakurta, 2022, luku 1” Benefits Adapting Infrastructure as Code”).

IaC nopeuttaa monia eri ohjelmistokehityksen osia. Se nopeuttaa infrastruktuurin käyttöönottoa sekä takaa, että ympäristö on täsmälleen sama jokaisella alustalla. Testaajat voivat pystyttää testausympäristöjä, jotka ovat täsmälleen samanlaisia kuin käyttöympäristö (Sneh, Thakurta, 2022, luku 1” Benefits Adapting Infrastructure as Code”).

Monista syistä infrastruktuurin pystytys jätetään usein pienelle määrälle kokeneita insinöörejä tai IT-työntekijöitä. Jos infrastruktuuria koskevan tieto ja ymmärrys on yrityksessä vain muutamalla IT-asiantuntijalla, jo yhden tiimin jäsenen lähtiessä pois yhtiöstä lähtee hänen mukanaan myös tärkeää kokemusta ja tietoa käytössä olevasta infrastruktuurista, ja tiimin työ vaikeutuu väliaikaisesti. IaC mahdollistaa ja takaa, että kaikki tieto käytössä olevasta infrastruktuurista on saatavilla helposti luettavasta koodista, joka helpottaa uusien kehittäjien perehdytystä (Sneh, Thakurta, 2022, luku 1” Benefits Adapting Infrastructure as Code”).

IaC myös auttaa bisneksiä saamaan kaiken irti IaaS (Infrastructure as a service) -palveluista nopeuttamalla ja helpottamalla infrastruktuurin käyttöönottoa ja skaalausta (Sneh, Thakurta, 2022, luku 1”Benefits Adapting Infrastructure as Code”).

4 Amazon Web Services

4.1 Amazon Web Services

Amazon Web Services (AWS) on vuonna 2006 perustettu Amazon.com:in omistama pilvipalvelualusta, joka sisältää monia eri etätietojen käsittelyresurssien palveluja. Näitä palveluja on esimerkiksi DynamoDB NoSQL-tietokannoille, S3 tiedostojen säilyttämistä varten, EC2, jolla voi vuokrata skaalautuvia virtuaalisia tietokoneita omaan käyttöön, ja Cognito, jolla voi hallinnoida kirjautumisia omaan palveluun. AWS sisältää myös monia valvontapalveluita, joilla voidaan seurata

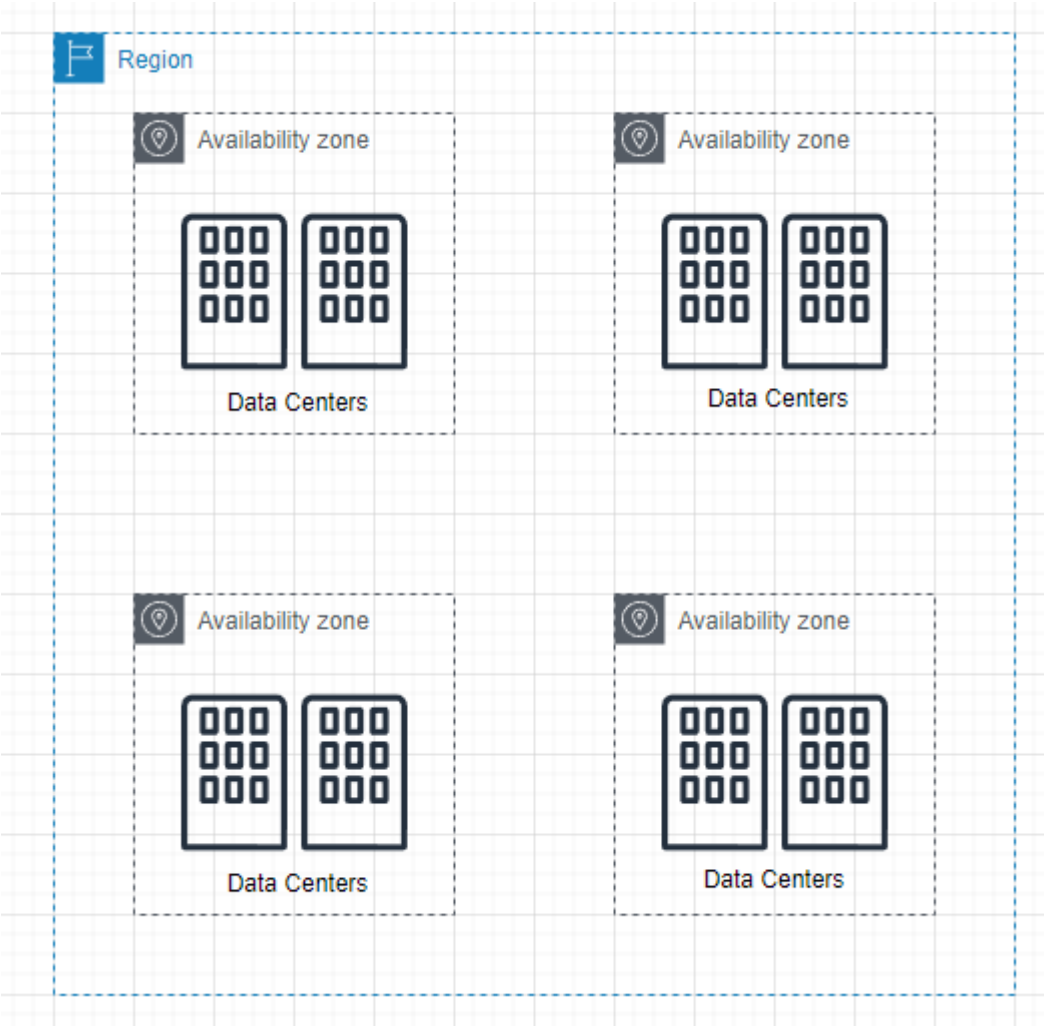
käytettävien palveluiden tilaa ja käyttöä, sekä automaattisesti tehdä toimia, jos jokin kriteeri täyttyy. Esimerkkinä uusien EC2 instanssien käynnistäminen, jos jo käytössä olevat alkavat olla täynnä.

Statista.com:in mukaan mukaan vuoden 2022 loppupuolella AWS:n osuus oli 34 % pilviarkkitehtuurimarkkinasta. Tämä 34 %:n osuus tekee Amazon Web Services:tä suurimman pilviarkkitehtuurin tuottajan. Toiselle sijalle jää Microsoftin Azure 21 %:n osuudella ja kolmanneksi Googlen Google Cloud 11 %:n osuudella (Statista.com, 2022).

4.2 AWS DynamoDB

Amazon DynamoDB on AWS:än sisältämä NoSQL tietokantapalvelu, joka sisältää ominaisuuksia, joilla käyttäjä voi hallinnoida datan turvallisuutta ja eheyttä, sekä skaalata tietokannan käyttöä tarpeen mukaan. DynamoDB on tietokantapalvelu, joka on tarkoitettu datan kirjoittamisen ja lukemisen suurelle määrälle. Data on tallennettuna SSD:ille eli puolijohdelevyille, jotka ovat nopeampia kuin perinteiset HDD levyt eli kovalevyt (docs.aws.amazon.com/amazondynamodb).

DynamoDB levittää datan automaattisesti eri palvelimiin ja saatavuusalueisiin sen pystytys regioonan sisällä (katso kuva 2) niin, että data on aina saatavilla ja palvelu voi aina toimia korkeankin käytön aikana, ja vaikka joissain saatavuusalueissa olisi ongelmia, jotka estäisivät palvelun käytön niissä. Tämän takia palvelu on ns. korkean saatavuuden palvelu, joka tarkoittaa sitä, että palvelu on käytettävissä melkein aina ja tarkoittamattomien katkosten minimointia varten on tehty töitä.



Kuva 2. Amazon Web Services palvelimien jakautuminen regiooniin, jotka sisältävät saatavuusalueita tietokeskuksineen.

DynamoDB sisältää kaksi eri datan varmuuskopiointitoiminnallisuutta. ”On-demand” -varmuuskopiot sekä ”point-in-time recovery”. ”On demand” -varmuuskopiointi on käyttäjän hallinnoima varmuuskopiointitoiminnallisuus, jolla käyttäjä voi luoda tauluista varmuuskopioita, joita säilytetään pidemmällä aikavälillä asiakkaan tarpeisiin. ”Point-in-time” -varmuuskopiointi on ominaisuus, jolla taulun voi palauttaa tilaan, jossa se oli minä tahansa hetkenä viimeisen 35 päivän aikana. Tällä ominaisuudella voidaan helposti korjata vahingollisia taulujen muokkauksia, jotka ovat liian suuria manuaaliselle korjaamiselle (docs.aws.amazon.com/amazondynamodb).

Ilman erillisiä toimenpiteitä DynamoDB taulut ovat saatavilla ainoastaan AWS-regioonassa, jossa ne luotiin. Kuitenkin, jos tauluja tarvitsee monessa regioonassa, voi tehdä ”Global tablen” eli maailmanlaajuisen taulun. Nämä taulut synkronoituvat automaattisesti haluttuihin regiooniin, jolloin

niiden data on saatavilla kaikissa halutuissa regioonissa (docs.aws.amazon.com/amazondynamodb).

4.3 AWS Cognito

Cognito on AWS-palvelu, jonka avulla hallinnoidaan sovellusten käyttäjä tilejä. Cognitossa luodaan käyttäjäryhmiä, johon jonkin tietyn palvelun käyttäjätilit luodaan. Sisäänkirjautuminen hoidetaan ottamalla yhteyttä tiettyyn käyttäjäryhmään, ja jos oikealla tunnisteilla varustettu tili löytyy käyttäjäryhmästä, niin Cognito lähettää Access Tokenin, jolla voi sen jälkeen tunnistautua Cogniton APIn kautta (AWS.Amazon.com/cognito, 2023).

Käyttäjäryhmä sisältää listan luoduista käyttäjistä ja yhdellä AWS-tilillä voi olla monta käyttäjäryhmää. Kun käyttäjä kirjautuu sisään Cognito-tunnukselleen, pitää sisäänkirjautumiseen käytettävän sovelluksen ottaa yhteyttä oikeaan käyttäjäryhmään. Jos sovellus yrittää kirjata käyttäjän, jonka käyttäjätunnus sijaitsee käyttäjäryhmässä ’ryhmä’1 ottamalla yhteyttä käyttäjäryhmään ’ryhmä2’ kirjautuminen epäonnistuu, vaikka molemmat käyttäjäryhmät sijaitsevat samalla AWS-tilillä (AWS.Amazon.com/cognito, 2023).

AWS Cognito sisältää myös niin kutsuttuja Identity Pooleja. Toisin kuin Käyttäjäryhmät, jotka hallitsevat käyttäjien rekisteröinnin ja kirjautumisen, Identity Poolit hallitsevat käyttäjien AWS IAM -rooleja (katso kappale 5.3). Roolit asetetaan erikseen rekisteröidyille käyttäjille ja rekisteröimättömille käyttäjille, eli vierailijakäyttäjille. Identity poolin käyttämiseen ei ole pakko käyttää Cognito käyttäjäryhmän käyttäjää, sillä on mahdollista käyttää myös kolmannen osapuolen Identity provideria, kuten googlen tai Facebookin sisäänkirjautumispalvelua (AWS Cognito identity pools, 2023).

4.4 AWS IAM

IAM (Identity and Access Management) on AWS-palvelu, jonka kautta hallitaan käyttöoikeuksia AWS-palveluihin. Lupien hallinnointi tapahtuu käyttämällä rooleja, käyttäjiä, sekä ryhmiä, joita kutsutaan yleisesti IAM-identiteeteiksi. Identiteettien lupia hallinnoidaan käytänteillä.

Policyt eli käytänteet sisältävät tiettyjä lupia, jotka määrittelevät, mitä käytänteen sisältämä identiteetti saa tehdä, kuten hakea tai kirjoittaa tietoa tietokannan taulusta. Mahdollisia käytänteitä on

monenlaisia: identiteettipohjaisia käytänteitä, resurssipohjaisia käytänteitä, lupa- rajoja, organisaatioiden palvelunhallinta käytänteitä, pääsynhallintalistoja, ja sessiokäytänteitä. Näistä kaikista yleisin on identiteettipohjainen käytänne.

Identiteettipohjaisia käytänteitä on kahdenlaisia: ns. inline-käytänteet ja hallitut käytänteet. Inline-käytänteet luodaan suoraan johonkin identiteettiin, ja ne ovat olemassa ainoastaan niin kauan kuin se identiteetti, johon se luotiin on olemassa. Inline-käytänteet ovat hyödyllisiä esimerkiksi tilanteissa, joissa luodaan identiteetti ja halutaan antaa sille tietty uniikki käytänne, jota ei ole muilla. Koska samaa käytännettä ei käytetä tämän yhden tapauksen ulkopuolella, on inline-käytänne helppo ja yksinkertainen keino antaa lupa identiteetille (AWS User Guide: Policies and Permissions, 2023).

Hallitut käytänteet ovat itsenäisiä. Ne eivät ole automaattisesti yhdistettynä mihinkään yhteen identiteettiin, vaan niitä voi lisätä kuinka moneen tahansa identiteettiin. Hallitut roolit eivät myöskään katoa, jos niitä sisältävä identiteetti poistetaan. AWS sisältää satoja AWS:än hallitsemia käytänteitä, joita voi lisätä tarvittuihin käyttäjiin, rooleihin, tai ryhmiin. Käyttäjät voivat luoda myös omia hallittuja käytänteitä, jos olemassa olevat käytänteet eivät tee juuri sitä, mitä halutaan. Itse luotuja käytänteitä voi muokata, ja jos niihin liitettyjä lupia poistetaan tai lisätään, kaikki sitä käytännettä käyttävät identiteetit muuttuvat automaattisesti (AWS User Guide: Policies and Permissions, 2023).

Resurssipohjaiset käytänteet ovat inline-käytänteitä, joita kiinnitetään tiettyyn resurssiin identiteetin sijasta. Resurssipohjainen käytänne kertoo, mitkä ”principal” saa käyttää sitä ja miten. ”Principal” on joko: AWS käyttäjä/root käyttäjä, IAM-rooli, roolisessio, IAM-käyttäjä, federoitu käyttäjäsessio, AWS-palvelu, tai kaikki edellä mainitut (AWS User Guide: AWS JSON policy elements: Principal, 2023). Hallittuja resurssipohjaisia käytänteitä ei ole. Resurssipohjaiset käytänteet toimivat myös AWS-käyttäjien välillä, jolloin oikeuksia voi antaa joko eri AWS käyttäjälle tai sillä sijaitsevalla IAM identiteetille. Näissä tapauksissa pitää oikeus antaa myös identiteettipohjaisella käytänteellä. Jos resurssi ja ”principal” sijaitsevat samalla AWS käyttäjällä, jo pelkästään resurssipohjainen käytänne on tarpeeksi. Resurssipohjaisia käytänteitä käytetään esimerkiksi S3 buketeissa hallinnoimaan, kenellä on oikeus lisätä, poistaa, muokata, tai muunnella niiden sisältöä (AWS User Guide: Policies and Permissions, 2023).

IAM Permission boundary eli luparaja on edistynyt ominaisuus, jolla rajoitetaan, kuinka monta lupaa identiteettipohjainen käytänne voi antaa IAM identiteetille. Luparaja ei vaikuta resurssipohjaisiin käytänteisiin (AWS User Guide: Policies and Permissions, 2023).

Access Control List eli pääsynhallintalista on resurssipohjaisten käytänteiden kaltainen palvelukäytänne, joka hallinnoi käyttäjän ulkopuolisten lupia käyttää resurssia. Se on ainoa käytännetyyppi, joka ei käytä JSON-tiedostomuotoa. Pääsynhallintalistat ovat vanhentunut käytännetyyppi ja AWS suosittelee, että se jätetään pois päältä. (AWS User Guide: Access control list (ACL) overview, 2023.)

Roolit ovat tapa antaa väliaikaisia lupia käyttäjille, ja sovelluksille. Rooleilla ei ole omia salasanoja eikä käyttäjänimiä, vaan rooli luodaan IAM:issa, ja se voidaan antaa tarvittavalle käyttäjälle, kun sitä tarvitaan. Roolit olevat olemassa käyttäjien ja sovellusten ulkopuolella, joten vaikka roolin sisältävä asia poistetaan, se ei vaikuta roolin olemassaoloon. Roolit ovat tapa antaa väliaikaisia lupia niitä tarvitseville ilman, että tarvitsee luoda uuden IAM käyttäjän. (AWS User Guide: IAM roles, 2023.)

IAM sisältää myös ryhmiä, jotka antavat kaikille ryhmän käyttäjille tietyt luvat. AWS käyttäjäopas antaa esimerkkinä ’järjestelmänvalvojat’ ryhmän, jonka jäsenet saavat järjestelmänvalvojaoikeudet AWS-palveluihin. Ryhmät voivat sisältää ainoastaan käyttäjiä. (AWS User Guide: IAM user groups, 2023.)

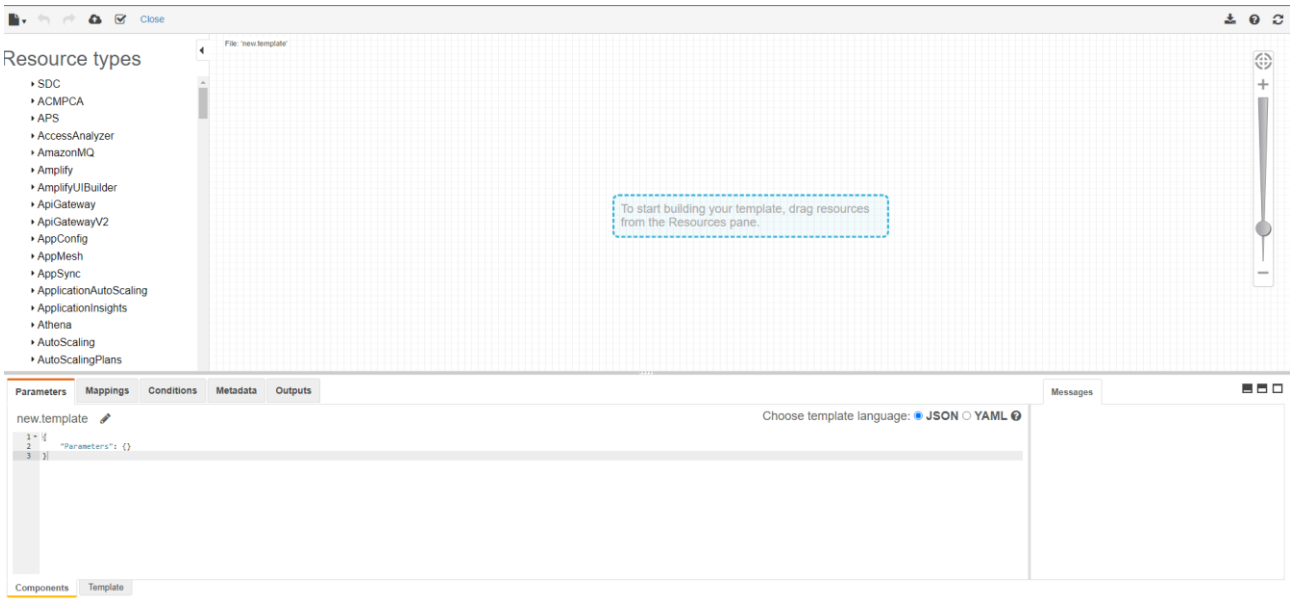
4.5 AWS CloudFormation

AWS CloudFormation on AWS IaC -palvelu, jonka avulla voi hallinnoida AWS-palveluita ja tiettyjä kolmannen osapuolen palveluita käyttäen koodia infrastruktuurin mallintamiseen. Malleja voi tehdä millä vain koodausohjelmalla, esimerkiksi ohjelmilla VSCode tai notepad++. Mallipohjia voi tehdä JSON/YAML-tietoformaateilla sekä TypeScript-, Python-, Java-, ja .NET-ohjelmointikielillä. CloudFormation mallipohjia voi kehittää myös visuaalisesti CloudFormation Designerilla, mikä helpottaa eri osien liitoksen ymmärtämistä sekä yleistä kykyä hahmoitta, mitä on luonut. CloudFormation Designer tukee ainoastaan YAML ja JSON formaatteja (AWS.Amazon.com/cloudformation, 2023).

CloudFormation kutsuu malleja ”stackeiksi” eli pinoiksi. Pino sisältää ohjeet siitä, mitä CloudFormationin kuulu tehdä, että kaikki sen sisältämät palvelut saadaan pystytetyksi. Kun pino pystytetään CloudFormationissa, se nähdään CloudFormationin ”Stacks” -välilehdellä, josta näkee pinon statuksen ja sen hallintavaihtoehdot. Pinoja voi aktivoida, muokata, ja poistaa riippuen siitä, mitä ne sisältävät. Joitain AWS-palveluita ei voi muokata, vaan niiden päivittämistä varten pitää koko pino poistaa ja pystyttää uudestaan päivitetyllä koodilla.

Kun luodaan monia eri palveluita samaan aikaan, esimerkiksi Cognito-käyttäjäryhmä ja siihen käyttäjä, on käyttäjä riippuvainen käyttäjäryhmästä ollakseen olemassa, sillä käyttäjä ei voi olla olemassa ilman ryhmää. CloudFormation luo kaikki pinossa luotavat osat automaattisesti niin, että ensin luodaan osat, joista muut osat saattavat olla riippuvaisia. Näin CloudFormation estää virheet, jotka syntyvät, kun yritetään luoda riippuvainen osa palvelua ilman emopalvelua.

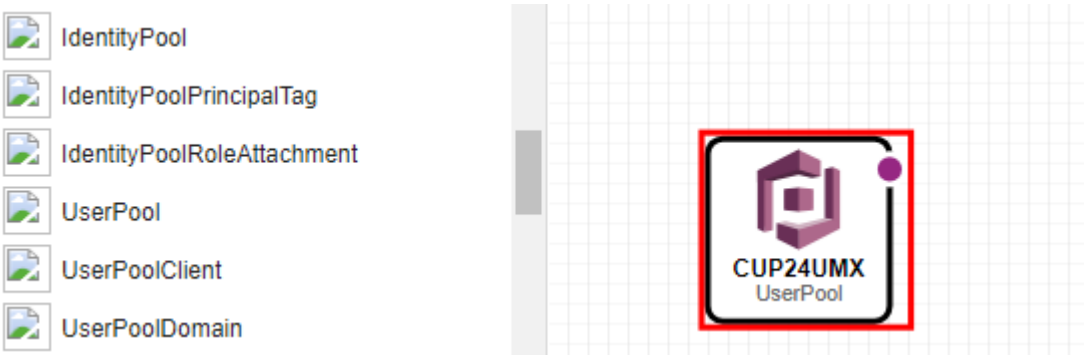
4.6 CloudFormation Designer ja miten sitä käytetään



Kuva 3. CloudFormation Designer aloitusnäkymä.

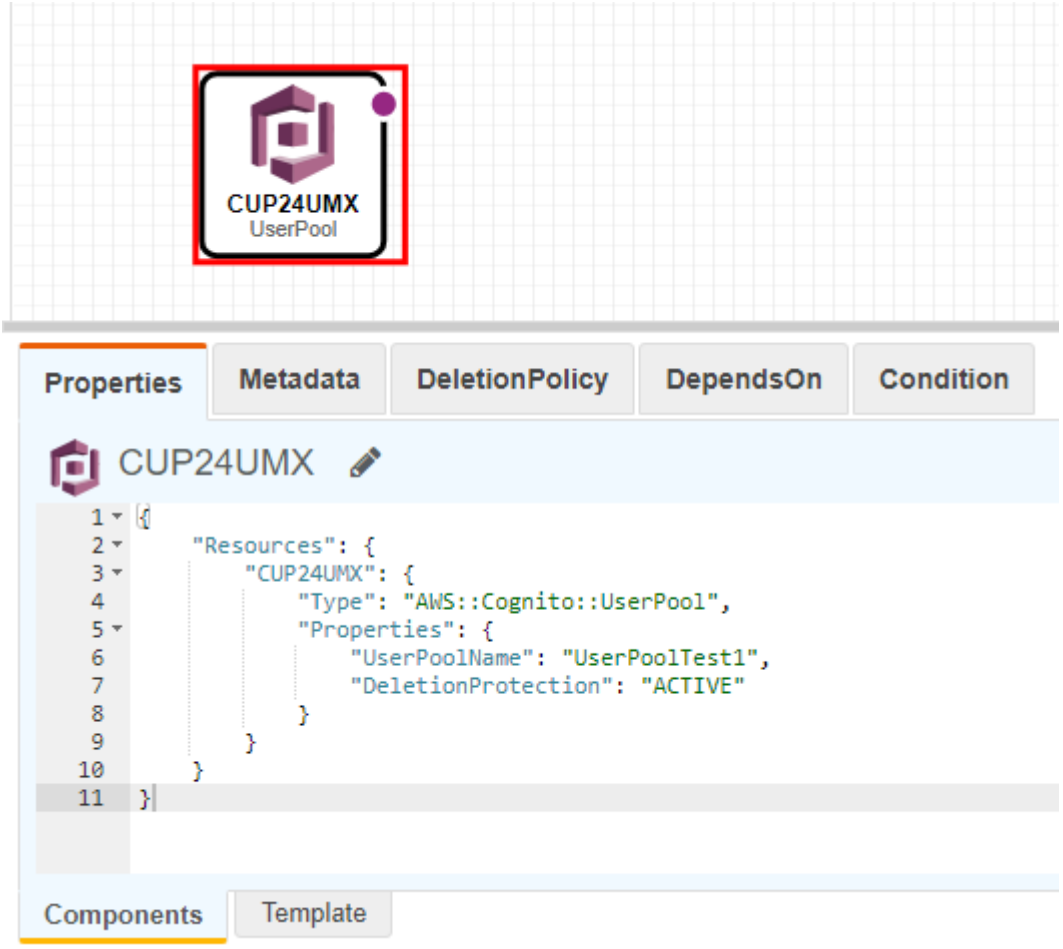
Kuvassa kolme näkyy CloudFormation Designerin aloitusnäkymä. Yläpalkissa on yleiset hallintavaihtoehdot tiedoston tallentamiseen, lataamiseen, testaamiseen, sekä nappi ohjesivulle.

Vasemmassa reunassa on lista kaikista resurssityypeistä, josta valitaan haluttu resurssi pystytettäväksi. Ruudun alapuolella on ikkuna, joka näyttää resurssien tiedot, sekä IaC-koodin. Koodin voi näyttää joko JSON- tai YAML-muodossa.



Kuva 4. Resurssi, joka on valittu valikosta ja lisätty mallipohjaan.

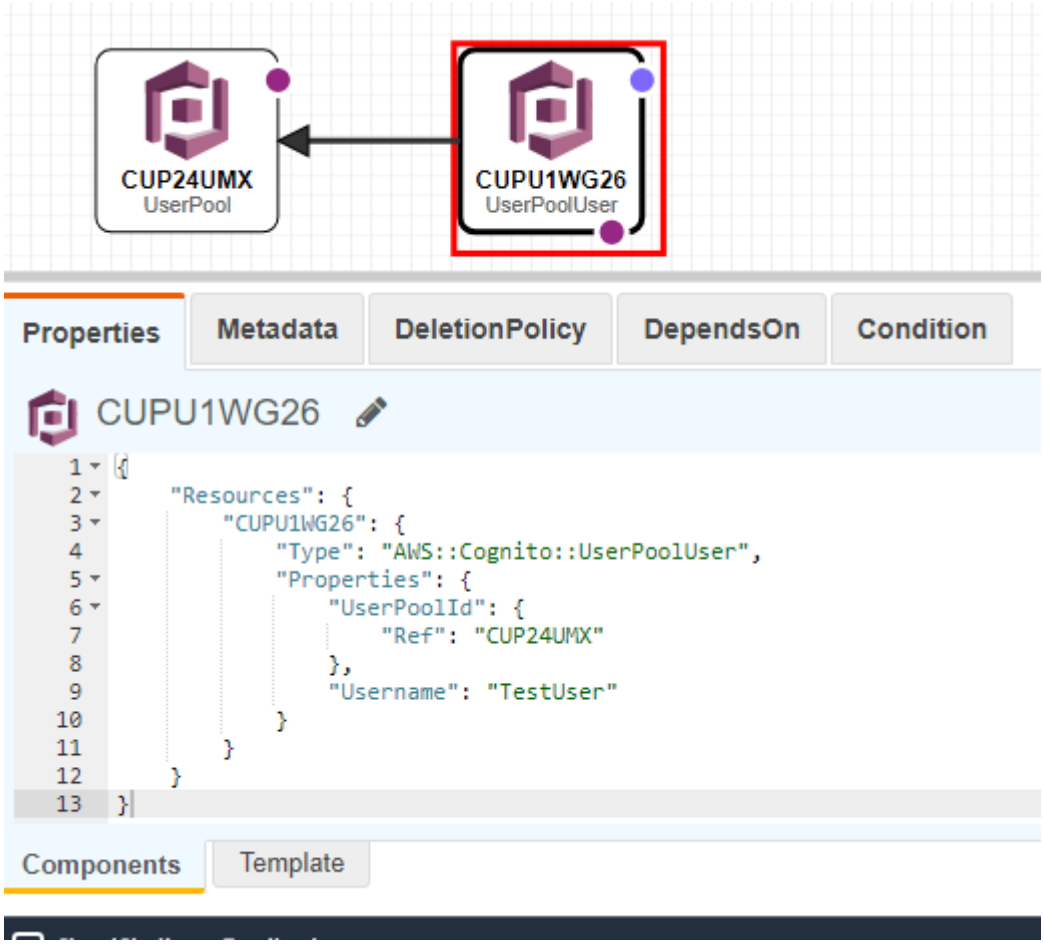
Kuvassa 4 on valittu UserPool tyyppinen resurssi, se näytetään neliönä, jossa resurssin ikoni, designerin sisäinen nimi (normaalin kokoinen teksti) sekä resurssin tyyppi (pienempi harmaa teksti).



Kuva 5. UserPool resurssi ja sen tiedot.

Kuvassa 5 näkyy UserPool ja pari sen ominaisuutta. Ne ovat JSON-muodossa. Valittuna on tällä hetkellä ”Properties” eli ominaisuudet-välilehti, jolloin resurssista näytetään ainoastaan ”properties”-sisältö, joka helpottaa ominaisuuksien hallintaa. Muita välilehtiä ovat metadata, joka sisältää resurssin id-koodin, DeletionPolicy, jossa määritetään, mitä tehdään resurssille, jos sen sisältävä pino poistetaan, ja viimeiseksi condition. Conditionilla voi määritellä ehtoja, joilla voi esimerkiksi määritellä, miten resurssi luodaan ainoastaan silloin, kun jokin ehto, kuten muun resurssin käyttö on ylittää annetun rajan.

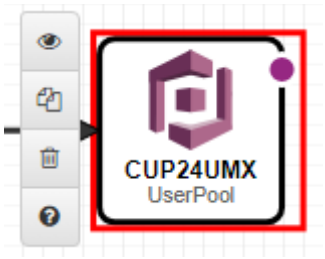
Kuvassa 5 on valittu components-välilehti, joka pilkkoo koko templaatin osiin sen muokkaamisen helpottamiseksi. Componets-napin vieressä on myös Template-nappi, joka näyttää koko templaatin.



Kuva 6. Desinger näyttää, kuinka käyttäjä on riippuvainen UserPool resurssista nuolen avulla.

Joillakin resursseilla on riippuvuuksia. Esimerkkinä kuvassa 6 on UserPoolUser-resurssi, joka tarvitsee UserPool-resurssin, johon se kuuluu, että sen voi luoda. Kun riippuvaista resurssia luodaan, sen ominaisuudet sisältävät muuttujan, jonka arvo tulee olla emoresurssin designerin sisäinen nimi. Kuvassa UserPoolUser ”CUPU1WG26” on riippuvainen UserPool resurssista ”CUP24UMX” ja se näkyy sen ”properties” tiedon alla kohdassa ”UserPoolid”. Siinä puolestaan on viittaus tarvittavaan UserPool resurssiin. Kun toimiva UserPoolId lisätään resurssiin, CloudFormation-designer piirtää automaattisesti nuolen, joka näyttää, kuinka resurssi on riippuvainen toisesta. Riippuvaisuuden voi myös lisätä vetämällä nuolen resurssilaatikon reunoilla sijaitsevista värikkäistä palloista. Kaikissa resursseissa on violetti pallo, joka lisää geneerisen riippuvuuden. Geneerisen riippuvuuden näkee resurssin DependsOn-välilehdellä. Jos resurssilla on ominaisuus, jonka arvon kuuluu olla jokin muu resurssi, näkyy resurssilaatikon reunalla sininen pallo. Kun hiiren siirtää sinisen pallon päälle, designer kertoo, minkä ominaisuuden riippuvuutta pallo edustaa, ja jos siitä vetää nuolen

oikeanlaiseen resurssiin, designer automaattisesti täyttää ominaisuuden arvon valitun resurssin nimellä.



Kuva 7. Pieni valikko resurssilaatikon vieressä.

Kuvassa 7 näkyy valikko, jonka voi avata painamalla resurssilaatikkoa hiiren oikealla napilla. Valikossa silmänappi avaa resurssin tiedot (katso kuva 5), kahden tiedoston sisältämä nappi monistaa resurssilaatikon, roskakori poistaa resurssin, ja kysymysmerkki avaa resurssin ohjesivun uudessa välilehdessä. Ohjesivulta löytyy kaikki resurssin ominaisuudet: (1) tieto siitä, miten ne kuuluvat täyttää, (2) tietoa siitä, mitä resurssi palauttaa, jos siihen viitataan sen ulkopuolelta ”ref” -komennolla, ja (3) mitä palautetaan, jos sen tietoja haetaan ”getAtt”-komennolla.

5 Kehittämistyön toteutus

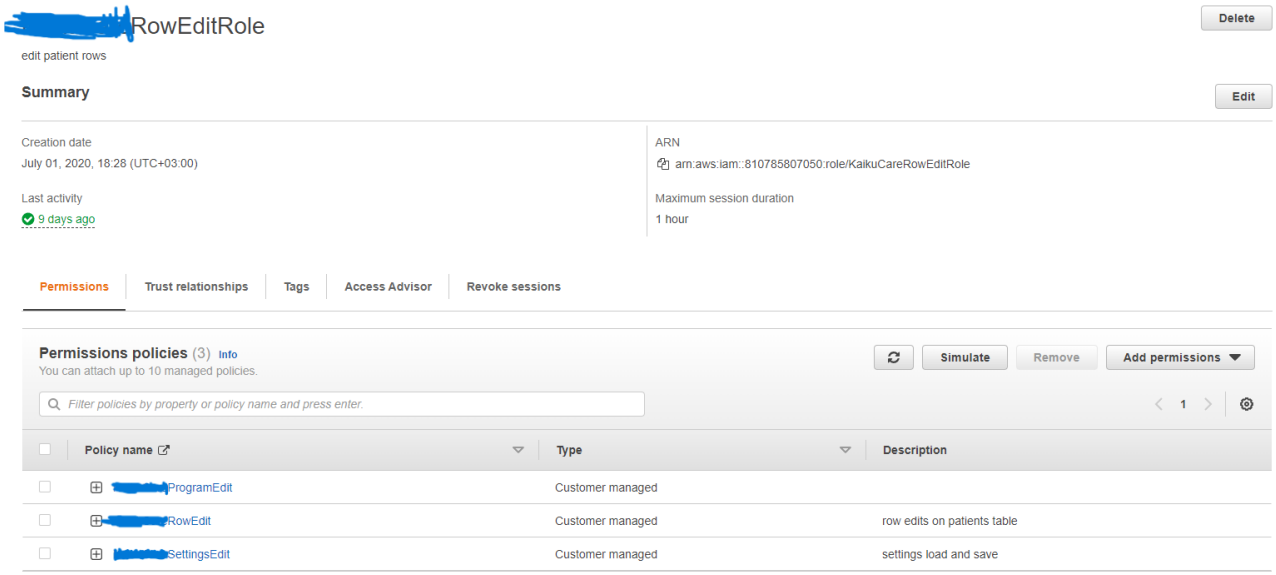
5.1 Työn alkutila

Sovelluksessa, jota työn toimeksiantaja CareUs OY kehitti, tietokantaan tallennettiin asiakkaiden potilaiden tiedot, potilaiden harjoitusrutiinit, sekä harjoituksien suorituksia koskevat tiedot. Kaikki asiakastiedot olivat samassa tietokannassa opinnäytetyön alkuvaiheessa, mutta CareUS halusi, että jokaisella asiakkaalla olisi oma tietokanta, jossa olisi vain kyseisen asiakkaan potilaiden tiedot. Työn alussa CareUs OY:llä on yksi AWS-tili, jolla sijaitsee heidän pystyttämänsä sovellus. Tilillä on käytössä: Identity Access Management (IAM), Cognito, sekä DynamoDB.

CareUs:än sovelluksessa alkutilassa kaikki käyttäjät olivat yhdessä käyttäjäryhmässä. Tämän vuoksi oli mahdollista, että teoriassa henkilö yhtiöstä 1 voisi kirjautua yhtiön 2 käyttäjälle omalta laitteeltaan, jos he tietäisivät käyttäjänimen ja salasanan.

Työn tarkoitus oli luoda AWS CloudFormation -mallipohja, jolla pystyy kerralla luomaan kaikki sovelluksen tarvitsemat palvelut yhdellä mallipohjan käyttökerralla.

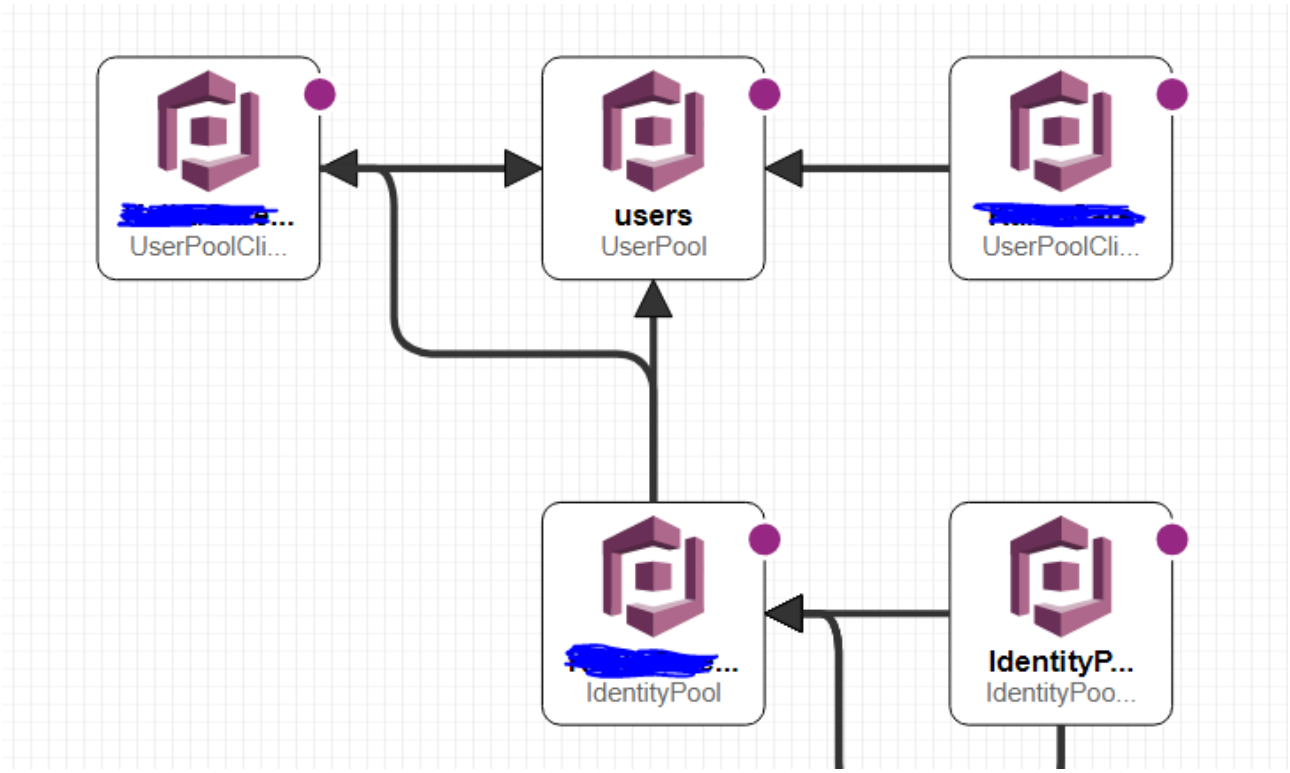
5.2 Työn aloitus



Kuva 8. Esimerkki kuvakaappaus jossa näkyy IAM roolin sisältämät käytänteet. (Sovelluksen nimi piilotettu)

Työn alussa CareUS loi minulle IAM-käyttäjän heidän AWS-käyttäjätilillensä, joka antoi minun nähdä kaikkien luotujen sovellusten asetukset. Työn teko alkoi ottamalla kuvakaappauksia kaikista luoduista palveluista ja niiden asetuksista. Kun kaikki tarvittavat kuvakaappaukset oli otettu, niistä saatujen tietojen pohjalta alettiin luomaan CloudFormation -mallipohjaa.

5.3 Cognito



Kuva 9. Cogniton käyttäjäryhmä patients (CloudFormationissa nimetty users) ja sen toiminnallisuuteen liittyvät osat.

Cognitossa on luotuna käyttäjäryhmä ”users” (kuva 9) ja identiteettiryhmä. Kun käyttäjäryhmästä luodaan CloudFormation mallipohja, sen ominaisuuksiin lisätään tavat, joilla siihen voidaan kirjautua sekä luoda uusia käyttäjiä, sekä tapa, jolla halutaan tehdä mahdolliseksi unohtuneen salasanan muutos.

```
{
    "Resources": {
        "users": {
            "Type": "AWS::Cognito::UserPool",
            "Properties": {
                "AccountRecoverySetting": {
                    "RecoveryMechanisms": [
                        {
                            "Name": "verified_email",
                            "Priority": 1
                        },
                        {
                            "Name": "verified_phone_number",
                            "Priority": 2
                        }
                    ]
                },
                "AdminCreateUserConfig": {
                    "AllowAdminCreateUserOnly": "False",
                    "UnusedAccountValidityDays": 7
                },
                "AutoVerifiedAttributes": [
                    "email"
                ],
                "DeletionProtection": "INACTIVE",
                "EmailConfiguration": {
                    "EmailSendingAccount": "COGNITO_DEFAULT"
                },
                "UsernameConfiguration": {
                    "CaseSensitive": "False"
                },
                "UserPoolName": "patients",
                "Schema": [
                    {
                        "Mutable": "False",
                        "Name": "email",
                        "Required": "True"
                    }
                ],
                "SmsConfiguration": {
                    "SnsCallerArn": "arn:aws:iam:[redacted]:role/SNSRole",
                    "SnsRegion": "eu-north-1"
                },
                "VerificationMessageTemplate": {
                    "DefaultEmailOption": "CONFIRM_WITH_LINK",
                    "SmsMessage": "Your password is {####}"
                }
            }
        }
    }
}
```

Kuva 10. Käyttäjäryhmän asetukset.

Kuvassa 10 näkyy käyttäjäryhmän asetukset. AccountRecoverySetting:n alla on valittu, että luodun käyttäjän salasanan voi vaihtaa kahdella tavalla, joko vahvistetulla sähköpostilla, tai vahvistetulla puhelinnumerolla. Asetuksissa on myös valittu, että muut kuin adminit voivat luoda käyttäjiä itsel-leen ja että adminien muille luomat käyttäjät poistetaan 7 päivän jälkeen, jos niitä ei ole käytetty kertaakaan. Muita asetuksia ovat: (1) vahvistussähköpostien lähettäjän osoite, joka on default

tässä tapauksessa, (2) käyttäjäryhmän nimi, joka on ”users”, sekä (3) sähköposti- ja tekstiviestien lähetys ja sisältöasetuksia.

Kuva 11. Käyttäjäryhmä clientin asetukset.

Käyttäjäryhmään on myös yhdistettynä kaksi userpool clienttia. Nämä ovat kuvassa 9 ylhäällä vasemmalla ja oikealla olevat palikat, joiden nimet ovat peitetty. Ne yhdistävät käyttäjäryhmän CareUs tekemään sovellukseen. Clientit viittaavat käyttäjäryhmään, johon ne kuuluvat kohdassa UserPoolId käyttämällä ”Ref” -toimintoa. Ref -toiminto palauuttaa yleensä nimen perusteella haetun resurssin ID:n. Toinen client vaatii secretin, joka on clientin salasana. Salasanan pitää olla mukana kaikissa sovelluksen clientille lähettämissä API-kutsuissa, jotta ne toimisivat. Client hallinnoi kirjautuessa sovelluksen käyttäjälle lähetettäviä tokeneita. Ne kertovat sovellukselle, kuka käyttäjä on, ja mitä oikeuksia hänellä on. Tokeneille annetaan clientin asetuksissa ajastukset, jotka määräävät, kuinka kauan ne ovat käyttökelpoisia niiden luonnista. Clientin asetuksissa voi myös valita,

mitä kautta clienttiin voi kirjautua. CareUsin sovellus hyväksyy ainoastaan Cognito-käyttäjäryhmän kautta kirjautumisen. Read ja write attributes -asetus määrää, mitä tietoja itsestään käyttäjät voi nähdä ja muokata. Kuvassa 11 ne ovat piilotettuja, koska listojen pituuden takia ne eivät mahtuneet kuvaan. Sovelluksessa on kuitenkin valittu, että käyttäjillä on lupa nähdä ja muokata kaikkia mahdollisia tietoja itsestään.

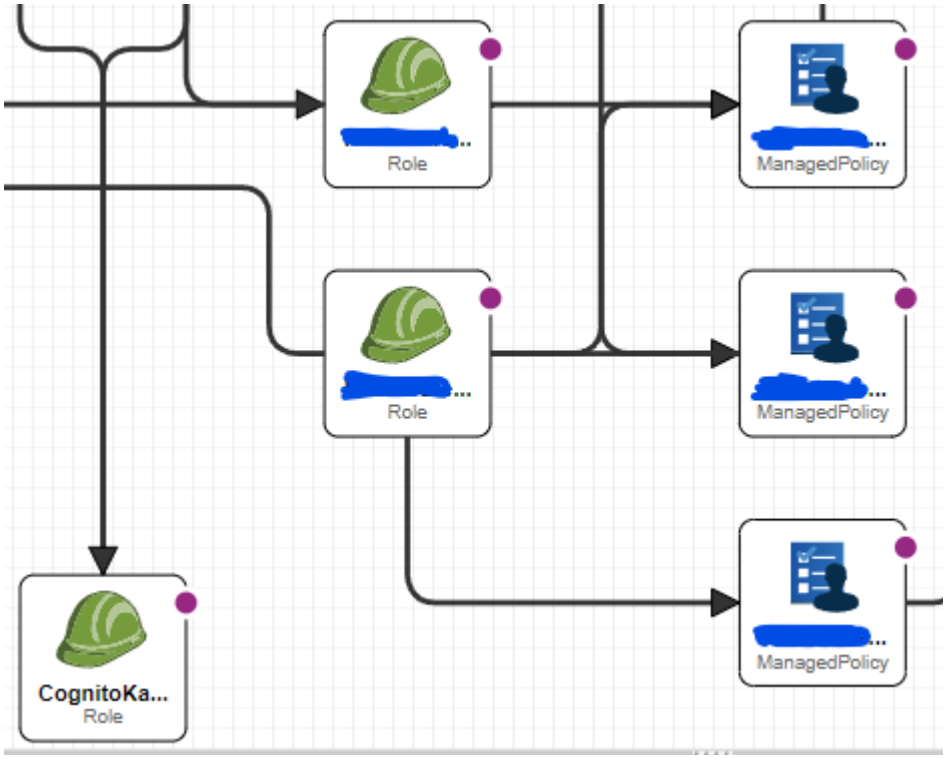
Kuva 12. Identiteettiryhmän asetukset.

Kuvassa 12 näkyy identiteettiryhmän asetukset. Asetuksissa on valittu, että sisään ei voi kirjautua ilman autentikaatiota sekä, että ei käytetä klassista autentikaatiotapaa. Valittuna on myös, mistä identiteettiryhmä saa kirjautumistiedot. Tässä tapauksessa se saa ne ’users’ -käyttäjäryhmästä clientin, jolla ei ole salasanaa, kautta. ProviderName arvo saadaan käyttämällä ”Fm::GetAtt” -funktiota, joka hakee tietystä CloudFormation -resurssista, tässä tapauksessa ”users”-käyttäjäryhmästä, ”ProviderName” arvon.

Kuva 13. IdentityPoolRoles asetukset.

IdentityPoolRoles on lisäosa identiteettiryhmälle, joka määrää, minkä roolin sisään kirjautunut käyttäjä saa. Kuvassa 9 se on alhaalla ja oikealla. Kuvan 13 asetuksissa on valittu, että autentikoidut eli sisään kirjautuneet käyttäjät saavat roolin RowEditRole. Se antaa käyttäjän editoida DynamoDB:n timedays -taulua, joka sisältää potilaan harjoitusten tuloksia. Asetuksissa on valittu myös, että ei sisään kirjautuneet eli vieras-käyttäjät saavat UnauthRole -roolin, joka ei anna mitään lupia. Tosin itse identiteettiryhmän asetuksissa on myös valittu, että autentikoimattomia käyttäjiä ei sallita, joten tämä osa jää käyttämättömäksi sovelluksen normaalissa toiminnassa.

5.4 IAM



Kuva 14. CloudFormationin IAM osa. Sovelluksen nimi peitetty.

Sovelluksen toiminnan mahdollistamiseksi luodaan kolme roolia ja kolme hallittua käytännettä. Roolien nimet ovat UnauthRole, johon ei ole yhdistetty käytänteitä. Sen käyttäjä ei saa siis mitään lupia. Toinen rooli on nimeltään RowEdit ja se antaa luvan editoida DynamoDB:n ”timedays” taulua. Kolmas rooli RowEditRoleRole puolestaan antaa luvan muokata kolmea käytössä olevaa DynamoDB taulua (”Settings”, ”Programs”, ja ”timedays”).

Kuva 15. Roolin RowEditRoleRole asetukset.

Roolien asetuksissa valitaan, mitkä käytänteet ovat voimassa roolin käyttäjälle. UnauthRole ja RowEdit roolit annetaan suoraan aikaisemmin kohdassa 6.3 käsitellyn Identiteettiryhmän kautta. Kumpikaan näistä rooleista ei anna lupaa editoida ”Settings” tai ”Programs” DynamoDB -tauluja, joita varten tarvitaan kolmas rooli ”RowEditRoleRole”. Tätä roolia ei anneta Identiteettiryhmästä, vaan se annetaan kuvassa 15 ”Condition”-kohdassa näkyvällä ehdolla. CareUs:in sovellus oli tämän työn tekovaiheessa vielä prototyyppi. Siksi ”RowEditRoleRole” saamisehto on vain, että on kirjautunut onnistuneesti sisään sovelluksen ainoaan identiteettiryhmään. Kun ehto on näin, saavat kaikki sisään kirjautujat molemmat ”RowEdit”- Ja ”RowEditRoleRole” -roolit, jolloin he kaikki voivat editoida kaikkia kolmea käytössä olevaa taulua. Roolin sisältämät käytänteet valitaan asetusten kohdassa ”ManagedPolicyArns”, jonka arvoksi laitetaan lista kaikista roolin sisältämistä hallituista käytänteistä. Roolien asetuksissa voisi myös olla inline-käytänteitä, jolloin rooliin ei linkitetä niiden

ulkoista hallittua käytännettä, vaan niihin lisätään suoraan käytänne dokumentti, kuten kuvan 16 käytänteessä näkyy kohdan ”PolicyDocument” alla.

Kuva 16. RowEditPolicy käytänteen asetukset.

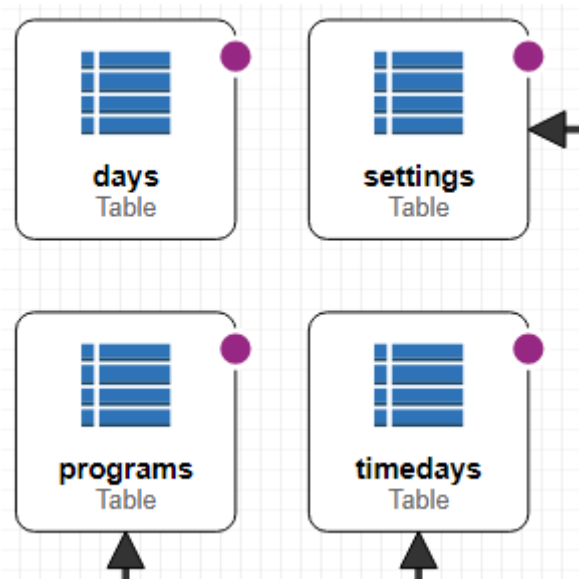
Kuvassa 16 näkyy hallittu käytänne nimeltä ProgramEdit. Hallittujen käytänteiden asetuksissa niille annetaan nimi, kuvaus, sekä mahdolliset viittaukset siihen, millä rooleilla, ryhmillä, tai käyttäjillä käytänne on käytössä, ja itse käytänteen dokumentti. Se kertoo, mitä käytänne antaa sen omistajan tehdä. Kuvassa näkyy kohdan ”PolicyDocument” alla ”Statement”-kohdassa, mitä käytänne tekee. ”Effect”: ”Allow” kertoo, että käytänne antaa luvan tehdä jotain. ”Action” -kohta kertoo, mitä toimintoja lupa koskee. Tässä käytänteessä luvat ovat dynamoDB -taulun lisäys-, haku-, poisto, ja muokkaustoiminnot. Viimeiseksi ”Resource” kertoo, mitä resurssia lupa koskee. Tämä käytänne koskee DynamoDb:n ”programs”-taulua se valitaan antamalla ”Resource” -kohdalle programs-taulun ARN (application resource name) eli applikaation resurssi nimi, joka on luodun resurssin uniikki osoite. Koska resurssia ei ole vielä luotu, eikä sillä ole vielä ARN:ia, viittaus tehdään CloudFormationin ”programs”-tauluun. Kun sitten mallipohjalla pystytetään tarvittavat resurssit, ”programs”-taulu luodaan ennen ”ProgramEdit”- käytännettä, jolloin käytännettä luodessa sille annetaan automaattisesti juuri luodun ”programs” -taulun ARN.

```
"Condition": {
    "ForAllValues:StringEquals": {
        "dynamodb:LeadingKeys": [
            "${cognito-identity.amazonaws.com:sub}"
        ]
    }
}
```

Kuva 17. SettingsEdit käytänteen ehto.

”SettingsEdit” käytäntö on hieman erilainen kuin ”programs” tai ”timedays” taulujen hallinnoinnin mahdollistavat käytänteet. Se antaa luvan ainoastaan taulusta hakemiseen, ja siihen lisäämiseen. Sillä on myös ehto, jota kahdessa muusa käytänteessä ei ole. Kuvassa 17 näkyvä ehto antaa käyttäjän hakea ainoastaan oman käyttäjänsä asetuksia. Ehto toimii vertaamalla käyttäjän identiteetti ryhmän ID:tä (”cognito-identity.amazonaws.com:sub”) taulun rivien osioavaimeen (”dynamodb:LeadingKeys”), joka on niiden luojan ID. Jos ne ovat samat, ehto täyttyy.

5.5 Dynamo DB



Kuva 18. Sovelluksen DynamoDB -taulut CloudFormation designerissa.

Sovelluksessa on 4 DynamoDb-taulua, joista 3 on aktiivisesti käytössä. Kaikki 4 kuitenkin luodaan. Taulut ovat ”days”, ”settings”, ”programs”, ja ”timedays”. Ne sisältävät tietoa kuntoutusohjelmien sisällöstä, kuntoutusohjelmien toteutuksista, ja käyttäjistä.

```
{
    "Resources": {
        "settings": {
            "Type": "AWS::DynamoDB::Table",
            "Properties": {
                "AttributeDefinitions": [
                    {
                        "AttributeName": "id",
                        "AttributeType": "S"
                    }
                ],
                "KeySchema": [
                    {
                        "AttributeName": "id",
                        "KeyType": "HASH"
                    }
                ],
                "BillingMode": "PROVISIONED",
                "DeletionProtectionEnabled": "False",
                "PointInTimeRecoverySpecification": {
                    "PointInTimeRecoveryEnabled": "False"
                },
                "ProvisionedThroughput": {
                    "WriteCapacityUnits": 5,
                    "ReadCapacityUnits": 5
                },
                "TableClass": "STANDARD",
                "TableName": "settings"
            }
        }
    }
```

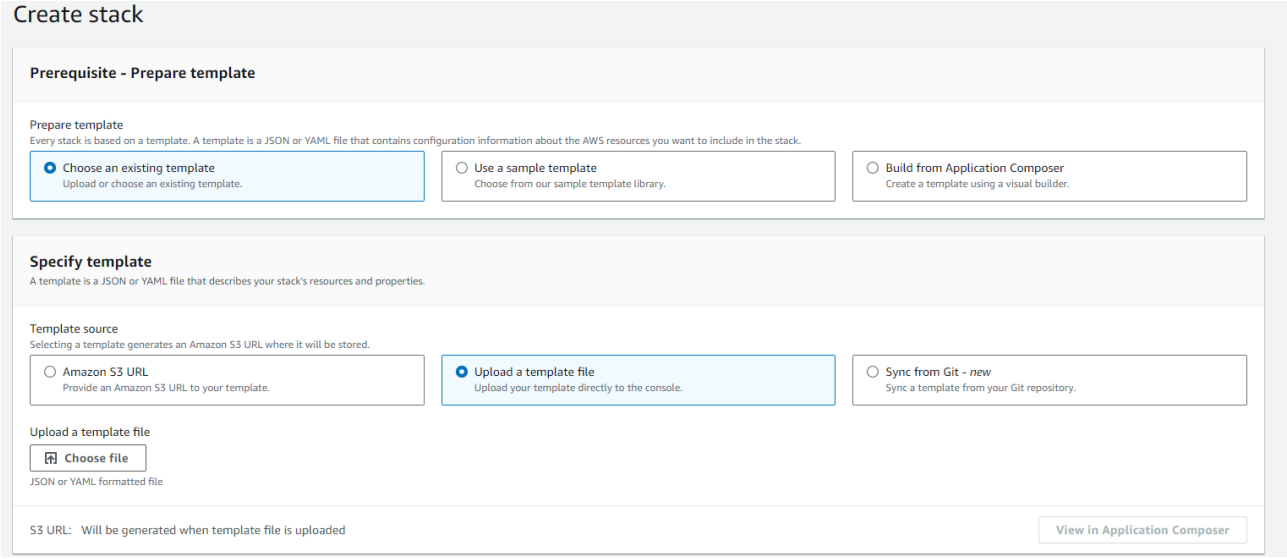
Kuva 19. Settings taulun asetukset

DynamoDB -taulut ovat tämän työn yksinkertaisimmat, sillä kaikilla niistä on käytännössä samat asetukset ja ainoastaan nimet ovat erilaiset. Tauluissa määritetään taulun pääavain kohdissa ”AttributeDefinitions” ja ”KeySchema”. Kun pääavaimia on vain yksi, kuten sovelluksen tauluissa ”KeySchema”- kohdassa annetaan vain yksi skeema elementti, joka sisältää attribuutin nimen ja tyypin. Kun on vain yksi pääavain, niin avaimen tyyppi on ”HASH”. Jos taululla olisi komposiittiavain tarvittaisiin toinen attribuutti, jonka tyyppi olisi ”RANGE”. Kuvassa 19 kohdassa ”AttributeDefitions” nähdään, että attribuutti ”id” on tyyppiä ”S” eli string, joka tarkoittaa, että sen arvo on merkkijono.

Muita asetuksia ovat ”BillingMode”, joka päättää, miten taulun käytöstä laskutetaan. Kuvassa 19 sen arvo on ”PROVISIONED” eli provisioitu, joka tarkoittaa sitä, että sille on valittu jokin tietty käyttömäärä, josta maksetaan. Se voisi olla myös ” PAY_PER_REQUEST”, jolloin taulun käytöstä maksettaisiin käytön perusteella, ja jolloin hinta vaihtelisi käyttömäärän perusteella. ”BilingMode”-

asetukseen kuuluu myös ”ProvisionedThroughput”- asetus, joka määrää, kuinka monta kirjoitus- ja lukupyyntöä palvelin voi enintään vastaanottaa sekunnissa.

5.6 Mallipohjan käyttö



Kuva 20. CloudFormationin uuden pinon pystytys.

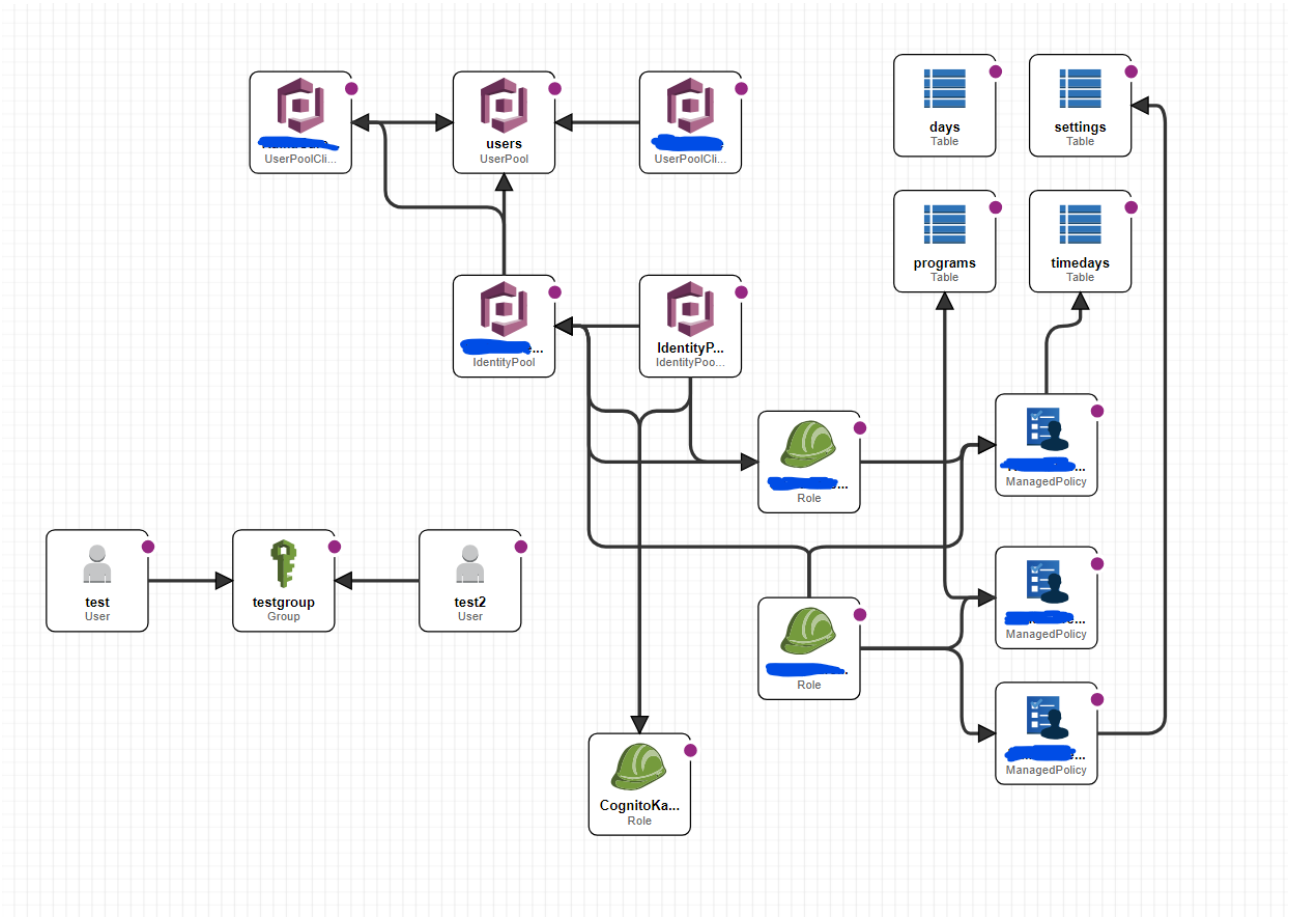
Kun mallipohja on luotu, sitä voidaan käyttää CloudFormationissa valitsemalla ”Create new stack”, jonka jälkeen valitaan mallipohja sekä sen lähde. Mallipohjan lähde voi olla AWS S3 bucket, joka on AWS-palvelu, johon voi tallentaa erilaisia tiedostoja, tai paikallinen tiedosto, joka ladataan AWS- palveluun sen käyttövaiheessa, tai GitHub repositorio. Kun mallipohja on valittu, painamalla ”next” nappia, sovellus pyytää käyttäjää nimeämään luodun pinon. Ja kun pino on nimetty ja ”next”-nappi painettu uudestaan, on vielä mahdollista antaa pinolle tageja. Tageilla voidaan lajitella AWS-palveluita sekä niiden osia. Lisäksi pinolle voi antaa IAM-roolin, jonka käyttäjillä on lupa tehdä toimintoja pinoon. Tässä vaiheessa valitaan myös, mitä tehdään, jos pinon pystytys epäonnistuu. Vaihtoehtoja ovat ”roll back”, jolloin kaikki jo tehdyt toiminnot perutaan, ja ”preserve”, jolloin jo onnistuneet resurssien pystytykset jätetään pystyyn. Näiden asetusten lisäksi on myös edistyneitä asetuksia, joita ei käydä tässä työssä läpi. Kun kaikki asetukset on määritelty kuten halutaan, seuraava sivu kokoaa kaikki pinon asetukset yhdelle sivulle. Siellä niitä voi vielä tarkistaa ja hyväksyä mahdolliset AWS-ilmoitukset pinon luomista resursseista, jonka jälkeen pino luodaan painamalla ”Submit”-nappia sivun oikeassa alakulmassa. Sen jälkeen palvelu suorittaa pinon pystytyksen.

6 Tulokset

6.1 CloudFormationin käyttö palvelukokonaisuuden monistamisessa

Ennen työn alkua keskustelin CareUS OY:n edustajan kanssa siitä, minkälainen tuote heillä oli sillä hetkellä olemassa sekä, miten he haluaisivat sitä laajentaa uusille asiakkaille. Työn alussa CareUS:illa oli yksi AWS-tili, jossa sijaitsi kaikki heidän palvelunsa. Keskustelussa heidän edustajansa kertoi, miten he halusivat, että luotaisiin AWS CloudFormation-mallipohja, jolla voitaisiin luoda sama palvelukokonaisuus käyttämällä sitä uudella tilillä. Keskustelussa tuli esille myös vaihtoehto, jossa kaikki palvelut sijaitsisivat samalla tilillä. Loppujen lopuksi päätimme kuitenkin, että monen tilin ratkaisu olisi parempi heidän käyttöönsä. Työn lopullinen tarkoitus oli siis luoda yksi mallipohja, joka pystyttää kaikki palvelut. Tätä luotua mallipohjaa voitaisiin käyttää monella tilillä pystyttämään täsmälleen samanlaiset palvelut niille kaikille.

6.2 Valmis mallipohja



Kuva 21. Valmis mallipohja näyttää tältä.

Työn tuloksena syntyi AWS CloudFormation-mallipohja, joka luo kopiot kaikista toimeksiantajan jo luomista AWS-palveluista. Niiden asetukset on kopioitu tarkasti jo olemassa olevista palveluista ja AWS CloudFormation pystyttää mallipohjaa käyttäen kaikki tarvittavat palvelut ilman virheitä.

Mallipohja-kuvassa (kuva 21) näkyy myös kaksi käyttäjää ja ryhmä, johon ne kuuluvat. Pohjan tekemisen jälkeen päätin, että ne eivät olleetkaan tarpeellisia tähän työhön, joten en käynyt niitä läpi aikaisemmissa kappaleissa. Ne kumminkin näkyvät omana kohtanaan valmiissa mallipohjassa, mutta ne eivät ole yhteydessä mihinkään muuhun luotuun palveluun.

Kun mallipohja palautettiin CareUs OY:lle, heiltä puuttui sillä hetkellä AWS:n käytöstä vastaava henkilö, jonka takia he eivät pystyneet itse testaamaan tai antamaan palautetta mallipohjasta. Palautuksen aikana kerroin myös, että jos mallipohjasta ilmenisi heille puutteita tai ongelmia, he ottaisivat yhteyttä, jolloin nämä puutteet tai mahdolliset virheet korjattaisiin. Palautus tapahtui 28.04.2023 eikä sen jälkeen ole tullut asiasta sen enempää viestiä.

6.3 Miten valmista AWS palvelukokonaisuutta voidaan monistaa AWS CloudFormationilla?

Valmiin palvelukokonaisuuden monistaminen monelle tilille CloudFormationille oli työn yksi päätavoitteista ja sen tekeminen onkin tullut jo esille. CloudFormationilla tehdyissä mallipohjissa on kaikki tieto siitä, mitä CloudFormation tarvitsee pystyttääkseen halutut palvelut. Kun mallipohja on tehty oikein, sitä voi käyttää millä vain AWS-tilillä pystyttämään täsmälleen saman palvelukokonaisuuden. Työssä luotua AWS-mallipohjaa voi siis käyttää monistamaan CareUS OY:n valmiin palvelukokonaisuuden uusille AWS-käyttäjille niin, että kaikilla käyttäjillä on täsmälleen samat palvelut samoilla asetuksilla. Silloin niiden luominen uusille asiakkaille on helpompaa ja nopeampaa kuin manuaalisesti palveluiden käynnistäminen ja muokkaaminen.

6.4 Miten AWS CloudFormationin IaC mallipohjat toimivat?

AWS CloudFormation mallipohjat ovat joko JSON- tai YAML-tiedostoja, jotka sisältävät tiedostomuodon mukaan tehdyn tiedon siitä, mitä CloudFormationin halutaan pystyttävän. Vaikka teoriassa mallipohjia voi kirjoittaa millä vain sovelluksella, jolla voidaan kirjoittaa koodia, sitä varten tehty työkalu AWS CloudFormation designer on helpompi tapa muokata ja luoda mallipohjia.

Tämä perustuu siihen, että se antaa helposti luettavassa muodossa tietoa siitä, mitä mallipohja sisältää, ja miten sen sisällön väliset yhteydet liittyvät toisiinsa. Lisäksi se erottelee jokaisen luodun palvelun asetukset toisistaan, jolloin niitä on helpompi lukea ja muokata. Designer kertoo myös, jos se näkee ongelmia mallipohjissa, ja se sisältää linkkejä kaikkien palveluiden dokumentaation.

AWS CloudFormation designerissä mallipohja luodaan vetämällä ja pudottamalla palveluita valikosta kankaalle, jolla sijaitsevat palvelut tallennetaan mallipohjaan. Designerin kankaalla näkyy helposti, mitä palveluja on luotu ja niitä klikkaamalla voi avata yksittäisten palveluiden tiedot tarkastelua ja muokkausta varten. Designer myös näyttää eri palveluiden vaatimat yhteydet toisiin palveluihin nuolina eri palvelupalikoiden välillä, jolloin eri palveluiden yhteyksiä on helppo lukea. Tämä on helpompaa kuin suoraa JSON- tai YAML-koodia lukeminen, jolloin ainoa näkyvä linkki palveluiden välillä on palveluin koodissa olevan asetuksen viittaus toiseen palveluun.

7 Pohdinta

7.1 Luotettavuus ja eettisyys

Työn aikana on noudatettu tutkimuseettisen neuvottelukunnan (TENK), suomalaisen tiedeyhteisön kanssa laatimaa hyvää tieteellistä käytäntöä eli HTK:ta, avoimuudella, kunnollisilla lähdeviittauksilla, sekä rehellisyydellä.

Vaikka CareUs Oy:n sovellus on terveydenhuoltokäyttöön tarkoitettu ja se tulee sisältämään potilastietoja, työn teon aikana ei kerätty tai käsitelty yksityisiä potilastietoja. Työssä käsiteltiin ainoastaan itse tietokantoja eikä niiden sisältöä. Työn kuvioissa on peitetty sovelluksen sekä laitteen nimi eikä niissä ole näkyvillä arkaluontoista tietoa. Työ ei vaatinut erillistä salassapitosopimusta.

Työssä luotua AWS CloudFormation mallipohjaa testattiin ainoastaan yksityisellä AWS-käyttäjätilillä, joka luotiin työtä varten. Työtä ei työnantajan puolesta päästy testaamaan, koska oma lupani heidän AWS-palveluihinsa oli ainoastaan lukemistason lupa eikä muokkauslupa. Heillä ei ollut työn tekohetkellä kokoaikaista AWS-vastaavaa, joka olisi voinut testata mallipohjan yhtiön puolesta. Sen takia työn lopullista toimivuutta ei päästy testaamaan sen tarkoitetussa käyttöympäristössä, joten sitä ei voi täysin taata.

7.2 Tulosten tarkastelu

7.3 Puutteet

Työn toteutuksen jälkeen opin, että olisi ollut parempiakin tapoja tehdä työ, ja oma tuotokseni ei ollut paras mahdollinen ja mahdollisesti jopa puutteellinen. Suurin puute työssä on ”deletion policyn” eli poistokäytänteen poisjääminen. Poistokäytänne hallinnoi sitä, mitä tehdään resurssille, kun CloudFormation-pino poistetaan (AWS user guide, DeletionPolicy. 2023).

Koska työssä luodussa mallipohjassa ei erikseen täsmennetä poistokäytännettä, se käyttää automattisesti ”Delete”-asetusta, jolloin pinon poistaessa kaikki sen pystyttämät resurssit poistetaan. Muita vaihtoehtoja ovet ”retain”, jolloin resurssi jätetään pystyyn sen palveluun, mutta se poistetaan CloudFormationista, tai ”RetainExceptOnCreate”, joka poistaa vain vastaluodut, ei vielä käytössä olevat resurssit ja jättää käytössä olevat resurssit koskematta. Tietyillä resursseilla, joita ei työssä käytetty, olisi myös mahdollista valita ”snapshot”, jolloin resurssi poistetaan, mutta siitä otetaan ”snapshot” eli varmuuskopio sen poistohetkellä olevasta tilasta (AWS user guide, DeletionPolicy 2023).

7.4 Mahdollinen jatkokehitys

CloudFormation ei ole ainoa IAC-palvelu ja on todennäköistä, että muilla palveluilla on eri vahvuuksia ja heikkouksia. CloudFormation on osa Amazon Web Services-palveluita, joten se on tehty toimimaan mahdollisimman hyvin sen kanssa. AWS ei kuitenkaan ole ainoa pilvipalvelutarjoaja. Microsoftin Azure-palvelu sisältää Azure Resource Management -palvelun, joka toimii kuin CloudFormation, mutta Azurelle (What is Azure Resource Manager, 2024).

On olemassa myös ns. kolmannen osapuolen palveluja, kuten Terraform. Terraform ei sisällä omia pilvipalveluita, joten se toimii jonkin muun pilvipalvelun, kuten Amazon Web Servicesin tai Azuren päällä, jolloin Terraform kertoo pilvipalvelulle, mitä palveluja pystytetään

Kehitystä voisi jatkaa tutkimalla, minkälaisia ominaisuuksia eri IAC-palveluilla on, miten ne ovat parempia tai huonompia, ja minkälaisiin käyttötarkoituksiin ne soveltuvat parhaiten. Toinen jatkokehityksen kohde voisi olla muiden pilvipalvelutarjoajien, kuten Google Cloud Platformin ja Microsoft Azuren tutkiminen siinä mielessä, miten ne soveltuvat käyttöön eri kolmannen osapuolen IAC-palveluiden kanssa. Esimerkiksi voisi selvittää, mitä eroja on Terraformin käytössä AWS:sän tai Azuren kanssa.

Työn toteutuksen jälkeen Amazon Web Services korvasi CloudFormation Designerin uudella Infrastructure Composer-nimisellä ohjelmalla, jossa on uusi käyttöliittymä, ja joka toimii hieman eri tavalla. Mahdollinen jatkokehityskohde on tämän uuden ohjelman erojen tutkiminen. Esimerkiksi voisi tutkia, miten sillä voi tai ei voi tehdä verrattuna CloudFormation Designeriin.

Lähteet

Access control list (ACL) overview. 2023. Viitattu 26.05.2023. https://docs.aws.amazon.com/AmazonS3/latest/userguide/acl-overview.html

Amazon Cognito identity pools (federated identities). 2023. Viitattu 30.09.2023. https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-identity.html

Amazon Cognito. Implement secure, frictionless customer identity and access management that scales. 2023. Viitattu 27.02.2023. https://aws.amazon.com/cognito/

Amazon DynamoDB. Fast, flexible NoSQL database service for single-digit millisecond performance at any scale. 2023. Viitattu 27.02.2023. https://aws.amazon.com/dynamodb/.

AWS JSON policy elements: Principal. 2023. Viitattu 26.05.2023. https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_principal.html

AWS CloudFormation. Speed up cloud provisioning with infrastructure as code. 2023. Viitattu 27.02.2023. https://aws.amazon.com/cloudformation/.

Bister, T. 2019. *Tietojenkäsittelyn opinnäytetyö: Viittoja ja karttoja tutkimisen ja kehittämisen teille*. Jyväskylä: Jyväskylän ammattikorkeakoulu.

DeletionPolicy attribute. 2024. Viitattu 25.04.2024. https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-deletionpolicy.html

Finder.Fi. CareUS Oy. Viitattu 30.03.2023. https://www.finder.fi/IT-konsultointi+IT-palvelut/CareUs+Oy/Jyv%C3%A4skyl%C3%A4/yhteystiedot/3554194

IAM roles. 2023. viitattu 23.08.2023. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html

IAM user groups. 2023. viitattu 24.08.2023. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_groups.html

Kananen, J. 2017. Laadullinen tutkimus pro graduna ja opinnäytetyönä. Jyväskylän ammattikorkeakoulu.

Mansoor, M., Zadgaonkar, A., Lamadrid, J., Ball, D., Malladi, R., Nargund, P., Zahniser, B., Olaoye, A., Kiswani, M. & McCann, T. Introduction to DevOps on AWS - AWS Whitepaper. 2014. Viitattu 28.02.2023. https://docs.aws.amazon.com/pdfs/whitepapers/latest/introduction-devops-aws/introduction-devops-aws.pdf#infrastructure-as-code

Pandya, S, ja Thakurta, RG. Introduction to Infrastructure as Code: A Brief Guide to the Future of DevOps. Apress L. P. 2022. Viitattu 05.05.2023. https://doi.org/10.1007/978-1-4842-8689-0.

Policies and permissions in IAM. 2023. Viitattu 26.05.2023. https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html

Statista.com. 2022. Amazon, Microsoft & Google Dominate Cloud Market. Viitattu 27.02.2023. https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/

What is Amazon DynamoDB? 2023. Viitattu 20.08.2023 https://docs.aws.amazon.com/amazon-dynamodb/latest/developerguide/Introduction.html

What is Azure Resource Manager? 2024. Viitattu 27.08.2024 https://learn.microsoft.com/en-us/azure/azure-resource-manager/management/overview

Zoting, S. 2024. Infrastructure as Code Market Size, Share and Trends 2024 to 2034. Precedence Research. September 2024. Viitattu 17.11.2024. https://www.precedenceresearch.com/infrastructure-as-code-market