AWS CloudFormation -malline toiminnanohjausjärjestelmän luomiseen

Petri Kauhtio

Haaga-Helia ammattikorkeakoulu
Tradenomin tutkinto
Amk-opinnäytetyö
2023

Tiivistelmä

Tekijä(t)
Petri Kauhtio

Tutkinto
Tradenomin tutkinto

Raportin/Opinnäytetyön nimi
AWS CloudFormation -malline toiminnanohjausjärjestelmän luomiseen

Sivu- ja liitesivumäärä
43 + 5

Tämä opinnäytetyö selvittää, kuinka AWS CloudFormation YAML-malline luodaan ja käytetään Odoo-toiminnanohjausjärjestelmän asentamiseen Ubuntu-palvelimelle. Työn tausta ja tavoite perustuvat tarpeeseen yksinkertaistaa ja automatisoida Odoo-toiminnanohjausjärjestelmän ja Ubuntu-palvelimen asennusta. Työn tietoperustassa keskitytään AWS CloudFormationiin, AWS EC2:een, AWS Elastic Load Balancingiin, Odoo-toiminnanohjausjärjestelmään ja Ubuntu-palvelimeen, sekä YAML-mallineen rakenteeseen, infrastruktuuriin koodina ja muuttumattoman infrastruktuurin periaatteisiin.

YAML-mallineen luominen oli iteratiivinen prosessi, joka suoritettiin tekstieditorissa. Jokainen mallineen rivi luotiin ja optimoitiin sitä mukaa, kun uusia teknologioita sisäistettiin ja ymmärrettiin paremmin. Tässä prosessissa AWS:n virallista dokumentaatiota käytettiin keskeisenä apuvälineenä ja tietolähteenä.

Lopputuloksena opinnäytetyöstä luotiin toimiva AWS CloudFormation YAML-malline Odoo-toiminnanohjausjärjestelmän asentamiseksi Ubuntu-palvelimelle. Tämä malline on testattu ja todettu toimivaksi ja sen avulla asiakasdemojen luomisprosessi voidaan nopeuttaa merkittävästi. Työ osoittaa, että infrastruktuurin hallinta koodin kautta tarjoaa tehokkaan ja joustavan tavan automatisoida ja ylläpitää IT-infrastruktuureja.

Asiasanat
Automatisointi, YAML, Pilvipalvelut, Amazon Web Services, Odoo

Sisällys

1 Johdanto ... 1
1.1 Kohteen kuvaus ja rajaus ... 1
1.2 Kohderyhmä ... 2
1.3 Keskeiset käsitteet ... 2
2 Amazon Web Services ... 4
2.1 AWS CloudFormation ... 4
2.1.1 Mallineen tiedostomuoto ... 5
2.1.2 Infrastruktuuri koodina ... 6
2.1.3 Muuttumaton infrastruktuuri ... 7
2.2 AWS EC2 ... 8
2.3 AWS Elastic Load Balancing ... 9
2.4 YAML-mallineen rakenne ... 10
3 Toiminnanohjausjärjestelmä ... 13
3.1 Odoo-toiminnanohjausjärjestelmä ... 13
3.2 Odoon ominaisuudet ja sovellukset ... 13
3.3 Odoon laitteistovaatimukset ... 14
4 Ubuntu-palvelimen valinta ja konfigurointi ... 16
4.1 Ubuntu-palvelimen konfigurointi AWS CloudFormationin avulla ... 16
4.2 Tietoturva- ja suorituskykyvaatimukset ... 16
5 Mallineen toteutus ... 18
5.1 Tuotoksen tuottamisen kuvaus ... 18
5.2 Mallineen testaus ... 21
5.2.1 Testien tulokset ja niiden analysointi ... 22
5.3 Lopullisen tuotoksen esittely ... 23
5.3.1 Mallineen komponentit ja resurssit ... 25
5.4 Mallineen käyttö ... 34
6 Arviointi ja pohdinta ... 38
6.1 Opinnäytetyön tavoitteiden saavuttaminen ... 38
6.2 Keskeiset löydökset ja niiden merkitys ... 38
6.3 Mahdolliset jatkotutkimusaiheet ... 38
Lähteet ... 40
Liitteet ... 44
AWS CloudFormation YAML -malline toiminnanohjausjärjestelmän luomiseen ... 44

odoo.yaml 44

1 Johdanto

Nykypäivän teknologisesti kehittyneessä ja nopeatempoisessa liiketoimintaympäristössä yritykset vaativat tehokkaita sekä joustavia ratkaisuja niiden monimuotoisiin hallintatarpeisiin. Yksi näistä ratkaisuista on Odoo, joka on avoimen lähdekoodin paketti täynnä mahdollisuuksia liiketoiminnan hallitsemiseen. Odoo tarjoaa käyttäjille ilmaisia, sekä maksullisia sovelluksia, joilla on omat toimintonsa, kuten projektinhallinta, varastonhallinta, kirjanpito sekä asiakkuuksien hallinta. (Odoo, s.a.) Odoo on saavuttanut suosiota tehokkuutta parantavien yritysten keskuudessa sen modulaarisen arkkitehtuurin ansiosta, joka tarkoittaa järjestelmän rakentamista erillisistä yhteensopivista moduuleista. (Solvve 2022)

IT-infrastruktuurin hallinta ja käyttöönotto saattaa kuitenkin olla monimutkaista ja aikaa vievää tehtävää. Amazon Web Services (AWS) CloudFormation-palvelu kuitenkin sisältää kustannustehokkaan, skaalautuvan sekä tehokkaan ratkaisun tehtävään vaadittavien infrastruktuuriresurssien käyttöönottoon sekä hallintaan. Kehittäjät voivat toimittaa ja määrittää AWS-resursseja hyödyntämällä AWS CloudFormation YAML-mallineita. Tämä tapa parhaillaan säästää useita työtunteja työntekijän päivästä, lisäten tuottavuutta ja vähentäen inhimillisiä virheitä. (Amazon Web Services s.a. a)

Tämä opinnäytetyö pyrkii esittämään kattavan yleiskuvan siitä, miten AWS CloudFormation-palvelua ja YAML-mallinetta voi hyödyntää nopeasti ja vaivattomasti Odoo toiminnanohjausjärjestelmän käyttöönottoon Ubuntu-palvelimella.

1.1 Kohteen kuvaus ja rajaus

Tämän opinnäytetyön aiheena on AWS CloudFormation-palvelun avulla toteutetun YAML-mallineen luominen. Valitsin aiheen koska olen käyttänyt mallineita aiemmin, ja työpaikallani on tarvetta YAML-mallineen hyödyntämiseen.

Tämä työ keskittyy erityisesti Ubuntu-käyttöjärjestelmällä ja Odoo-toiminnanohjausjärjestelmällä varustettuun palvelimen luomiseen AWS CloudFormationin avulla. Rajaus on tehty siten että työn ulkopuolelle jäävät toiminnanohjausjärjestelmän yksityiskohtaisen käytön tai sen tarjoamien toiminnallisuuksien yksityiskohtainen esittely.

Opinnäytetyön tavoitteena on luoda toimiva ja tehokas YAML-malline AWS CloudFormationin avulla. Työ sisältää sekä teoreettista että käytännön tietoa AWS CloudFormationista ja YAML-mallintamisesta.

Työn tietoperustassa (luvut 2–4) käyn läpi pilvipalvelut, AWS CloudFormationin ja YAML-mallineen suunnittelun, sekä tutkin Ubuntu-käyttöjärjestelmän ja Odoo-toiminnanohjausjärjestelmän valintaa työhön. Lisäksi käsittelen Ubuntu-palvelimen konfigurointia AWS CloudFormationin avulla sekä tietoturva- ja suorituskykyvaatimuksia. Työn empiirinen osa (Luku 5) sisältää YAML-mallineen luomisen esittelyn. Lopullinen malline löytyy liitteenä.

1.2 Kohderyhmä

Tällä hetkellä kohderyhmänä toimii yritys, jossa työskentelen. Meidän tiimimme pääasiallisena tarkoituksena on käyttää tätä mallinetta Odoo-asiakasdemojen nopeampaan luontiin ja erilaisiin tarpeisiin, kuten asiakastapaamisiin ja testiympäristöihin. Mallineen avulla voi tehokkaasti havainnollistaa Odoo-toiminnanohjausjärjestelmän ominaisuuksia ja toiminnallisuuksia, testata uusia ideoita ja suorittaa suorituskykytestejä.

Vaikka mallineen käyttäjät ovat rajattu tällä hetkellä pieneen ryhmään on mahdollista, että myöhemmin yrityksessämme muut tiimit tai henkilöt voivat hyötyä mallineen käytöstä. Siksi on tärkeää dokumentoida mallineen käyttö ja ylläpito selkeästi, jotta se on helposti ymmärrettävissä ja sovellettavissa tarvittaessa.

Yrityksemme ulkopuolelle kuuluvia kohderyhmiä ovat esim. muut Odoo-toimittajat, ohjelmistokehittäjät, IT-konsultit ja järjestelmäarkkitehdit, jotka ovat kiinnostuneita käyttämään AWS CloudFormation-mallinetta Odoo-toiminnanohjausjärjestelmän asennukseen ja ylläpitoon. He voivat hyödyntää tätä mallinetta omien asiakkaidensa tarpeisiin, kuten esimerkiksi tarjoamalla heille Odoo-pohjaisia ratkaisuja.

1.3 Keskeiset käsitteet

Automatisointi Prosessien, järjestelmien tai tehtävien suorittaminen ilman tai vähäisellä ihmisen toiminnalla.

AWS Amazon Web Services on kattava ja turvallinen pilvipalvelualusta, joka tarjoaa laskentatehoa, tietokantapalveluita, tiedonsäilytystä ja monia muita palveluita, jotka auttavat yrityksiä skaalautumaan ja kasvamaan.

CloudFormation Infrastruktuurin hallintatyökalu, joka mahdollistaa resurssien määrittämisen ja käyttöönoton automatisoidun ja johdonmukaisen prosessin kautta.

EC2	Virtuaalisten palvelinten, tai niin sanottujen virtuaalikoneiden (VM), tarjoaminen ja hallinta pilvipalvelussa.
ERP	ERP/Toiminnanohjausjärjestelmä on kattava ohjelmistotyökalu, joka yhdistää eri toimintojen hallinnan ja parantaa liiketoiminnan tehokkuutta.
Odoo	Avoimen lähdekoodin toiminnanohjausjärjestelmä, joka kattaa monia yrityksen toimintoja, kuten CRM, myynti, projektinhallinta, varastonhallinta ja kirjanpito.
Pilvipalvelu	Tietotekniikan palveluiden tarjoaminen etäyhteydellä asiakkaille. Resurssit, kuten laskenta, tallennus ja sovellukset jaetaan dynaamisesti käyttäjien kesken.
Ubuntu	Laajalti suosittu ja käytetty Linux-pohjainen käyttöjärjestelmä. Ubuntu on tunnettu sen joustavuudesta ja luotettavuudesta, mikä tekee siitä ihanteellisen valinnan sekä palvelin- että työasemaympäristöihin.
YAML	Yksinkertainen ja ihmiselle luettava tiedostomuoto, joka on suunniteltu määrittämään tietorakenteita, kuten konfiguraatioita, datan serialisointia ja mallineita.

2 Amazon Web Services

Amazon Web Services (AWS) on maailman johtava pilvipalveluntarjoaja, joka tarjoaa kattavan ja monipuolisen valikoiman pilvipohjaisia palveluita. AWS:n palveluihin kuuluvat muun muassa laskentapalvelut, tietokantapalvelut, analytiikka, koneoppiminen, tietoturva, tallennus ja sisällönjakelu, ja ne tukevat monia erilaisia sovelluksia ja liiketoimintaskenaarioita. AWS:n infrastruktuuri kattaa useita maantieteellisiä alueita ja kattaa satoja palvelimia ympäri maailmaa, tarjoten korkean suorituskyvyn, luotettavuuden ja skaalautuvuuden. (Amazon Web Services s.a. b)

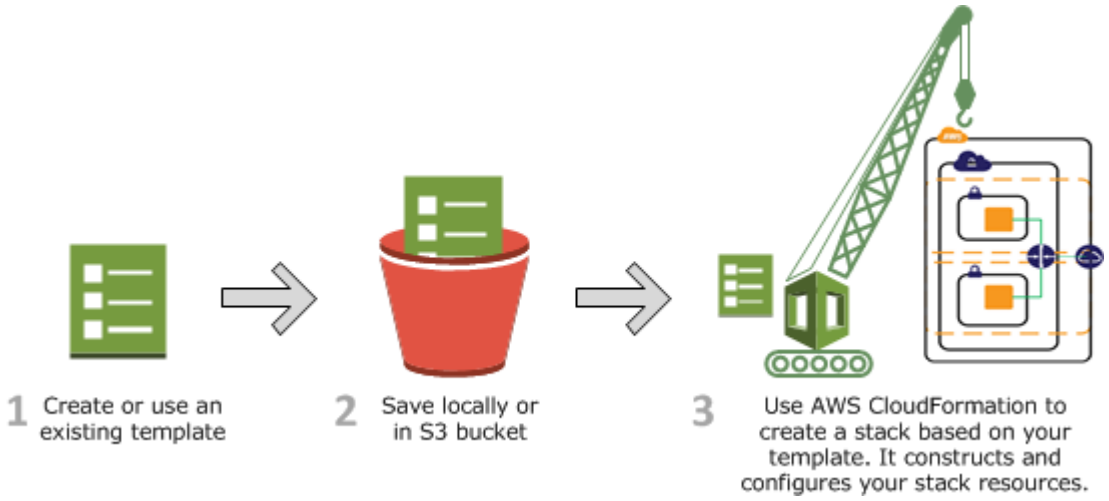
Opinnäytetyöhön AWS valikoitui useista syistä. Ensinnäkin AWS:n tarjoamat palvelut ovat alan johtavia niiden suorituskyvyn, luotettavuuden ja turvallisuuden suhteen. Toiseksi AWS tarjoaa laajan valikoiman työkaluja ja palveluita, jotka helpottavat ja nopeuttavat infrastruktuurin asettamista, hallintaa ja skaalausta. Näitä työkaluja ovat esimerkiksi AWS CloudFormation, joka on keskeinen osa tätä opinnäytetyötä. Kolmanneksi AWS tarjoaa erinomaisen tukiverkoston ja kattavan dokumentaation, mikä helpottaa sen käyttöä ja oppimista. Lopuksi AWS tarjoaa Ubuntu-pohjaisia virtuaalipalvelimia vuokrattavaksi, joka on olennainen osa opinnäytetyönäni. (Zdnet 2022)

2.1 AWS CloudFormation

AWS CloudFormation on Amazonin tarjoama palvelu, joka auttaa käyttäjiä hallitsemaan AWS-resursseja. CloudFormation mahdollistaa infrastruktuurin mallintamisen koodina, mikä tarkoittaa, että AWS-resurssit, kuten virtuaalikoneet, tietokannat ja verkkokomponentit voidaan luoda, päivittää ja poistaa automatisoidusti. CloudFormationissa käyttäjät voivat luoda ja hallita AWS-resurssien kokoelmaa, jota kutsutaan pinoiksi (stacks). Nämä pinot määritellään mallineissa (templates), jotka voidaan kirjoittaa joko JSON- tai YAML-formaatissa. (Amazon Web Services s.a. a)

Kuten kuvassa 1 on kuvattuna nämä mallinetiedostot määrittelevät AWS-resurssit ja niiden konfiguroinnit. Kun malline lähetetään AWS CloudFormationille, palvelu luo ja konfiguroi määriteltyjä resursseja automaattisesti ja järjestelmällisesti mallineen perusteella. CloudFormationin käyttöönotto on tehokasta, koska se automatisoi resurssien luomisen ja hallinnan, mikä vähentää manuaalisen työn tarvetta. Koska resurssit määritellään koodilla, on mahdollista seurata infrastruktuurin muutoksia versionhallintajärjestelmän kautta, mikä parantaa järjestelmän hallittavuutta ja toistettavuutta. (Amazon Web Services s.a. a)

Opinnäytetyössäni käytän AWS CloudFormationia luomaan YAML-mallineella Ubuntu-palvelimen, jossa on Odoo-toiminnanohjausjärjestelmä.



Kuva 1. CloudFormation-pinon luominen mallineella (Amazon Web Services s.a. c)

2.1.1 Mallineen tiedostomuoto

AWS CloudFormation tukee kahta tiedostomuotoa mallineiden luomiseen: JSON ja YAML. Nämä tiedostomuodot määrittävät AWS-resurssit ja niiden konfiguroinnit. Kumpikin muoto on joustava ja sopii hyvin infrastruktuurin mallinnukseen koodin muodossa, mutta niissä on myös joitakin merkittäviä eroja. (Amazon Web Services s.a. d)

JSON (JavaScript Object Notation) on tiedostomuoto, jota hyödynnetään usein tietojen serialisoinnissa ja se on yhteensopiva monien ohjelmointikielten kanssa. JSON-muoto perustuu avain/arvo-pareihin, jotka muodostavat objekteja. JSON-muodossa ei kuitenkaan ole kommenttien tukemista, joka saattaa tehdä koodin ymmärtämisestä haastavaa, ja sen syntaksi vaatii kaikkien objektien ja taulukoiden sulkemisen sulkumerkkeihin, mikä voi tehdä siitä hieman hankalan lukea ja kirjoittaa. (Json s.a.)

YAML (YAML Ain't Markup Language) on toinen tiedostomuoto, jota käytetään paljon tietojen serialisoinnissa. YAML on suunniteltu olemaan helposti luettava ihmiselle ja se tukee kommentteja, mikä tekee koodin ymmärtämisestä ja ylläpitämisestä helpompaa. Lisäksi YAML käyttää sisennystä tietojen rakenteen kuvaamiseen, mikä voi tehdä siitä selkeämmän ja luettavamman kuin JSON-muoto. (Red Hat 2023a)

Opinnäytetyössäni valitsin käyttää YAML-muotoa AWS CloudFormation -mallineiden luomiseen useista syistä. YAML-muodon selkeys ja helppo luettavuus tekee siitä erinomaisen valinnan infrastruktuurin mallintamiseen koodin muodossa. Koska mallineet voivat olla melko monimutkaisia, on tärkeää, että ne ovat mahdollisimman helppoja ymmärtää ja ylläpitää.

YAML myös tukee kommentteja, mikä on erittäin hyödyllinen ominaisuus koodin selittämiseen ja dokumentoimiseen. Kommenttien avulla voin selittää mitä tietty osa mallinetta tekee, tai jättää muistiinpanoja itselleni tai muille kehittäjille tulevaa varten.

YAML-muodon sisennyksen ansiosta on myös helppoa hahmottaa tiedoston rakenne. Tämä tekee mallineiden kirjoittamisesta ja ymmärtämisestä helpompaa varsinkin, jos mallineet kasvavat suuriksi ja monimutkaisiksi.

Minulla on lisäksi aiempaa kokemusta YAML:n käyttämisestä. Olen käyttänyt sitä aikaisemmissa projekteissa, jonka takia olen varmempi tämän käyttämisestä.

2.1.2 Infrastruktuuri koodina

Infrastruktuuri koodina (Infrastructure as Code (IaC)) on moderni IT-alan lähestymistapa, joka korostaa infrastruktuurin hallintaa automatisoitujen prosessien kautta sen sijaan, että se tehtäisiin manuaalisesti. IaC:ssä infrastruktuurin määrittely, konfigurointi ja hallinta tapahtuvat koodin muodossa. Tämä tarkoittaa, että järjestelmän asetukset mukaan lukien palvelimet, verkkolaitteet ja tallennusratkaisut määritellään ja hallinnoidaan kooditiedostoissa. (Amazon Web Services s.a. e)

IaC on tärkeä konsepti, koska se mahdollistaa järjestelmän toistettavuuden, automatisoinnin ja versionhallinnan. Toistettavuus tarkoittaa sitä, että samaa infrastruktuuria voidaan luoda uudelleen ja uudelleen identtisesti koodin kautta. Tämä vähentää inhimillisten virheiden riskiä ja tekee ympäristöjen, kuten testi-, kehitys- ja tuotantoympäristöjen, luomisesta nopeaa ja tehokasta. (Sourcefuse 2023)

Automatisointi on toinen keskeinen IaC:n ominaisuus. Se mahdollistaa infrastruktuurin luomisen, päivittämisen ja poistamisen ilman manuaalista osallistumista. Tämä nopeuttaa merkittävästi toimitusketjua ja vapauttaa IT-tiimin aikaa muihin tehtäviin. (Amazon Web Services s.a. e)

Lisäksi versionhallinta mahdollistaa muutosten seurannan ja palauttamisen. Jos järjestelmässä ilmenee ongelma, voi IT-tiimi palata aiempaan toimivaan versioon nopeasti. Tämä parantaa järjestelmän saatavuutta ja vähentää ongelmien aiheuttamia kustannuksia. (Red Hat 2023b)

IaC:n soveltaminen AWS CloudFormationissa oli luontevaa opinnäytetyön kontekstissa, koska se mahdollistaa AWS-resurssien hallinnan ja konfiguroinnin koodin kautta. AWS CloudFormationin avulla voidaan määrittää ja käyttää AWS-resursseja koodipohjaisesti, mikä tekee infrastruktuurin hallinnasta johdonmukaista, toistettavaa ja tehokasta. (Amazon Web Services s.a. e)

AWS CloudFormation tarjoaa eri formaatteja, kuten JSON ja YAML tiedostojen kirjoittamiseen. Nämä tiedostot, jotka ovat CloudFormation-mallineita, määrittelevät AWS-resurssit ja niiden konfiguroinnit. (Amazon Web Services s.a. d)

Opinnäytetyössäni luon YAML-mallineen, joka hyödyntää AWS CloudFormationia Odoo-toimintajärjestelmällä varustetun Ubuntu-palvelimen luomiseen.

2.1.3 Muuttumaton infrastruktuuri

Muuttumaton infrastruktuuri (Immutable infrastructure) on lähestymistapa, jossa infrastruktuurin resursseja, kuten palvelimia tai tietokantoja, ei päivitetä tai muuteta sen jälkeen, kun ne on alun perin luotu. Sen sijaan, kun infrastruktuuriin tarvitaan muutoksia, esimerkiksi päivitysten tai konfiguraatiomuutosten vuoksi, luodaan uusi infrastruktuurin versio ja vanha infrastruktuuri korvataan kokonaan uudella. Tämä lähestymistapa eroaa perinteisestä infrastruktuurin hallinnasta, jossa olemassa olevia resursseja muokataan ja päivitetään tarpeen mukaan. (TechTarget 2022)

Muuttumattoman infrastruktuurin perusajatuksena on, että infrastruktuurin resurssit ovat johdonmukaisia ja ennustettavia, koska niitä ei muuteta sen jälkeen, kun ne on alun perin määritelty. Tämä vähentää inhimillisten virheiden riskiä, parantaa järjestelmän luotettavuutta ja helpottaa ongelmien havainnointia ja korjaamista, koska infrastruktuurin tila on aina tiedossa. (HashiCorp 2018)

Muuttumaton infrastruktuuri ilmenee opinnäytetyössäni AWS CloudFormationin ja Infrastructure as Code (IaC) -lähestymistavan kautta. CloudFormationin avulla voin määritellä infrastruktuurin resurssit koodissa ja luoda uuden infrastruktuurin version joka kerta, kun tarvitaan muutoksia. Esimerkiksi, kun tarvitsen päivityksen Ubuntu-palvelimeeni en päivitä olemassa olevaa palvelinta, vaan luon uuden CloudFormation-mallin, joka määrittelee uuden palvelimen version. Tämän jälkeen CloudFormation luo uuden palvelimen, joka otetaan käyttöön ja vanha palvelin voidaan poistaa käytöstä.

Odoo-toiminnanohjausjärjestelmän tapauksessa sen tietokanta säilytetään tavallisesti paikallisesti palvelimella. Tässä lähestymistavassa tulee kuitenkin ongelma: jos Odoo-tietokanta säilytetään paikallisesti, se menetetään aina, kun luodaan uusi palvelimen versio. Tämä ei ole hyväksyttävää, sillä Odoon tietojen säilyvyys on tärkeää. Ratkaisuna tähän ongelmaan voi käyttää Amazon Web Services RDS:ää (Relational Database Service).

AWS RDS on pilvipohjainen relaatiotietokantapalvelu, jossa voi luoda, ylläpitää sekä hallita tietokantoja pilvessä. Käyttämällä RDS:ää, Odoon tietokannan voi pitää erillään palvelimesta ja siirtää suoraan uudelle palvelimelle palvelimen päivityksen yhteydessä. Tämä varmistaa, että Odoon käyttäjien tietoja ei menetetä, vaikka infrastruktuuri muuttuu. (Amazon Web Services s.a. f)

Huomautan kuitenkin, että tässä opinnäytetyössä olen rajannut ulkopuolelle RDS-tietokannan luomisen ja Odoon liittämisen tähän AWS RDS -tietokantaan. Mallinne tällä hetkellä sen sijaan luo paikallisen tietokannan Ubuntu-palvelimelle, mutta tavoitteena on myöhemmin luoda malline, jossa tietokanta saadaan liitettyä automaattisesti RDS-tietokantaan.

Muuttumaton infrastruktuuri parantaa järjestelmän luotettavuutta, koska infrastruktuurin tila on aina tiedossa ja ennustettavissa. Se myös nopeuttaa päivitysprosessia, koska uuden infrastruktuurin version luominen ja vanhan poistaminen voidaan automatisoida. Lisäksi se mahdollistaa infrastruktuurin versionhallinnan, koska jokainen infrastruktuurin versio on määritelty koodissa ja tallennettu versionhallintajärjestelmään. (Digital Ocean 2017)

Käyttämällä muuttumatonta infrastruktuuria opinnäytetyössäni voin varmistaa, että infrastruktuurini on aina johdonmukainen ja toistettavissa. Kohdatessani ongelmia tai tarvitessani muuttaa jotain, voin aina luoda uuden version infrastruktuurista sen sijaan, että yrittäisin korjata tai muuttaa olemassa olevia resursseja.

2.2 AWS EC2

AWS Elastic Compute Cloud (EC2) on virtuaalinen palvelinkeskus, joka tarjoaa skaalautuvaa laskentakapasiteettia pilvessä. Se on osa Amazon Web Services -pilvipalvelukokonaisuutta ja tarjoaa käyttäjille mahdollisuuden vuokrata virtuaalikoneita, joita voidaan käyttää monenlaisissa sovelluksissa. AWS EC2 tarjoaa nopean ja joustavan tavan skaalata kapasiteettia tarpeen mukaan, mikä tekee siitä erinomaisen valinnan sekä pienille että suurille yrityksille, jotka tarvitsevat joustavaa infrastruktuuria. (Amazon Web Services s.a. g)

AWS EC2 tarjoaa monia eri virtuaalikoneiden tyyppejä, jotka on optimoitu eri käyttötarkoituksiin, kuten laskentaan, muistiin, tallennukseen tai verkko-optimoituun käyttöön. Käyttäjät voivat myös valita käyttöjärjestelmän ja sovelluspaketin virtuaalikoneilleen, kuten tässä opinnäytetyössä, jossa käytän Ubuntu-pohjaista virtuaalikonetta. (Amazon Web Services s.a g)

AWS EC2-palveluun kuuluu myös monia muita ominaisuuksia, kuten automaattinen skaalaus, joka mahdollistaa virtuaalikoneiden automaattisen lisäämisen tai poistamisen kapasiteetin tarpeen mukaan. Tämä auttaa käyttäjiä hallitsemaan kustannuksia ja takaa, että palvelimet pystyvät vastaamaan kysynnän muutoksiin. (Amazon Web Services s.a. h)

EC2 tarjoaa myös tiiviin integraation muihin AWS-palveluihin, kuten AWS CloudFormationiin, joka on keskeinen osa tämän opinnäytetyön infrastruktuuria. CloudFormationin avulla voimme määritellä ja hallita EC2-resursseja koodin avulla, mikä tekee palvelinten käyttöönotosta, konfiguroinnista ja hallinnasta nopeaa ja tehokasta. (Jenna Pederson 2021)

Opinnäytetyössäni käytän AWS EC2-palvelua luomaan Ubuntu-pohjaisen virtuaalipalvelimen, joka toimii Odoo-toiminnanohjausjärjestelmän alustana. Tämä palvelin määritellään ja luodaan AWS CloudFormationin avulla.

2.3 AWS Elastic Load Balancing

AWS Elastic Load Balancing (ELB) on keskeinen osa Amazon Web Servicesin (AWS) tarjoamaa infrastruktuuria. Se auttaa jakamaan verkkoliikennettä useiden AWS-resurssien, kuten EC2-palvelimien välillä. ELB:n avulla voi varmistaa, että sovellukset toimivat sujuvasti ja luotettavasti, jopa suuren käyttäjämäärän ja liikenteen alla. (Amazon Web Services s.a. i)

Tässä opinnäytetyössä keskityn AWS:n Elastic Load Balancerin (ELB) käyttöön osana Odoo-toiminnanohjausjärjestelmän käyttöönottoa AWS CloudFormationin ja YAML-mallineen avulla. Käytännössä tämä tarkoittaa, että luon infrastruktuurimallinetta, joka määrittää tarvittavat AWS-resurssit ja niiden konfiguraatiot. Tässä mallineessa hyödynnän erityisesti ELB:n tyyppiä nimeltään Application Load Balancer (ALB), jota kutsutaan yrityksemme sisällä nimellä "netitysALB".

NetitysALB on erityisesti HTTP- ja HTTPS-liikenteen tasapainotukseen suunniteltu kuormantasain. Se toimii sovelluskerroksella (kerros 7 OSI-mallissa) ja tarjoaa monia erinomaisia ominaisuuksia, kuten sisällön perusteella reitityksen, WebSocket-tuen ja HTTP/2-tuen. (Amazon Web Services s.a. j)

NetitysALB:n avulla voin reitittää pyynnöt tiettyihin palvelimiin perustuen pyynnön sisältöön, kuten URL-osoitteeseen. NetitysALB tukee myös automaattista skaalautumista. Kun liikenne kasvaa, ELB lisää automaattisesti kapasiteettiaan vastaamaan kysyntää, jolloin sovelluksen sujuva toiminta säilyy. Vastaavasti, kun liikenne vähenee, ELB vähentää automaattisesti kapasiteettiaan, mikä säästää kustannuksia. (Amazon Web Services s.a. j)

Tässä opinnäytetyössä netitysALB:n käyttö tarjoaa monia etuja. Se auttaa varmistamaan, että Odoo-toiminnanohjausjärjestelmä on aina saatavilla ja toimii sujuvasti, jopa suuren käyttäjämäärän alla. Lisäksi netitysALB jakaa liikenteen tasaisesti useiden palvelimien välillä, parantaa sovelluksen suorituskykyä ja vähentää yksittäisen palvelimen kuormitusta. (Amazon Web Services s.a. j)

NetitysALB:n käyttöönotto tapahtuu osana CloudFormationin YAML-mallineen luomista. CloudFormationin YAML-mallinetta käytetään määrittämään ja luomaan tarvittavat AWS-resurssit, kuten EC2-instanssit, turvallisuusryhmät ja netitysALB. Tämä malline tarjoaa selkeän ja järjestelmällisen tavan hallita ja automatisoida infrastruktuurin luonti AWS:ssä.

NetitysALB:n konfiguroinnin osalta määritellään esimerkiksi, minkä tyyppistä liikennettä tasapainotetaan (esim. HTTP tai HTTPS), mitkä EC2-instanssit kuuluvat sen tasapainotusryhmään ja millä säännöillä liikennettä reititetään.

Tämän YAML-mallineen luon ja päivitän iteratiivisesti opinnäytetyön aikana. Aluksi luon yksinkertaisen mallineen, joka määrittää perusinfrastruktuurin. Tämän jälkeen mallinetta kehitetään lisäämällä uusia resursseja ja ominaisuuksia, kuten netitysALB ja mukauttamalla olemassa olevien resurssien konfiguraatiota tarpeiden mukaan.

2.4 YAML-mallineen rakenne

Tässä opinnäytetyön osiossa kuvailen mistä YAML-mallineen rakenne koostuu AWS CloudFormation-ympäristössä. AWS CloudFormation-mallineen struktuuri on monimutkainen, ja se koostuu useista osista kuten kuvattu kuvassa 2. Jokaisella niistä on omat roolinsa ja tarkoituksensa.

Malline alkaa yleensä "Format Version" (AWSTemplateFormatVersion) -osalla. Tämä osio määrittelee CloudFormation-mallineen version, joka auttaa AWS:ää ymmärtämään mallineen sisällön oikein. Tällä hetkellä ainoa tuettu versio on "2010-09-09". (Amazon Web Services s.a. k)

Toinen osio on "Description". Tämä osio antaa yleiskatsauksen mallineen toiminnasta ja tavoitteista. Se auttaa ymmärtämään, mitä malline tekee ja mihin sitä käytetään. (Amazon Web Services s.a. l)

Seuraava osio on "Parameters". Tässä osiossa määritellään arvoparit, joihin voidaan viitata myöhemmin mallineessa. Ne tarjoavat tavan määrittää tietoja, jotka ovat erilaisia joka kerta, kun mallinetta käytetään. Esimerkkinä parametreillä voidaan määrittää mikä Odoon versio asentuu asettamalla parametriin tietyn numeron, kuten 15. Tällöin malline asentaisi Odoo 15-version. (Amazon Web Services s.a. m)

"Mappings"-osio on seuraava. Mappings-osiossa voidaan määritellä esimerkiksi eri arvoja eri alueille, tai muita arvopareja, jotka voivat vaihdella riippuen esimerkiksi ympäristöstä tai konfiguroinnista. (Amazon Web Services s.a. n)

"Conditions"-osio sisältää logiikan, jonka perusteella tietyt resurssit luodaan tai määritetään eri tavalla eri tilanteissa. Esimerkiksi, saatan haluta luoda erilaisia resursseja tuotanto- ja testiympäristöissä. (Amazon Web Services s.a. o)

"Transform"-osio on erityinen osio, joka mahdollistaa AWS Serverless Application Model (SAM) tai muiden makrojen käytön, jotka voivat muuttaa mallineen ennen kuin se lähetetään AWS:lle. (Amazon Web Services s.a. p)

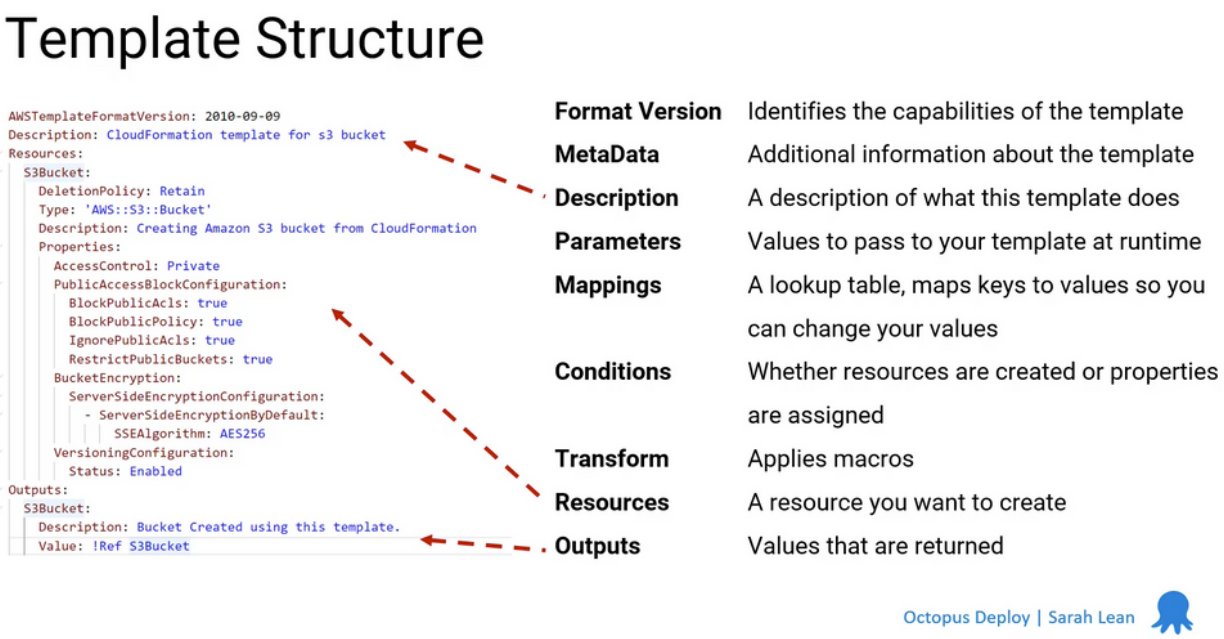
Mallineen sydän on "Resources"-osio, joka sisältää AWS-resurssit, jotka luodaan mallineen käyttämisen yhteydessä. Tässä osiossa määritellään AWS-resurssit, kuten Amazon EC2 -instanssit. "Metadata"-osio on osa tätä "Resources"-osiota ja sisältää yleensä tietoja, jotka auttavat ymmärtämään resurssien sisältöä. Se voi sisältää esimerkiksi kuvaustietoja, dokumentaatiota tai kuvauskuvan mallineesta. "Init"-osio, joka on osa "Metadataa", mahdollistaa sovellusten, kuten Odoo, tiedostojen ja muiden resurssien asentamisen EC2-instansseihin. (Amazon Web Services s.a. q)

Toinen osio, "MetaData", on valinnainen ja on osa ”Resources”-osiota. MetaData sisältää yleensä tietoja, jotka auttavat ymmärtämään resurssien sisältöä. MetaData sisältää myös mallineelle tärkeän osan ”Init” ja se mahdollistaa sovellusten, kuten Odoo, tiedostojen ja muiden resurssien asentamisen EC2-instansseihin. Odoo-toiminnanohjausjärjestelmän asentaminen ja siihen tarvittavien pakettien asentaminen asennus tapahtuu tässä mallineen osiossa. (Amazon Web Services s.a. r)

Vielä yksi mainitsemisen arvoinen resurssi on ”WaitCondition”. Sen tarkoitus on tarjota mekanismi, jolla AWS CloudFormation voi viivästyttää stackin luomista tai päivitystä, kunnes tietyt ehdot täyttyvät. Se on hyödyllinen resurssien riippuvuustilanteissa, joissa tietyn tapahtuman on tapahduttava ennen kuin AWS CloudFormation voi jatkaa prosessia. (Amazon Web Services s.a. u)

Esimerkiksi, WaitConditionia voitaisiin käyttää varmistamaan, että Odoo on asennettu ja käynnistynyt onnistuneesti ennen kuin stackin luominen tai päivitys päättyy. Huolimatta WaitConditionin potentiaalisesta hyödystä, en koe sen olevan välttämätön mallineessani.

Kaikki nämä osiot toimivat yhdessä määrittämään AWS CloudFormation-mallineen, joka ohjaa AWS-resurssien luomista ja konfigurointia. Lopuksi haluan huomauttaa, että mallineessani en käytä "Mappings" ja "Transform" osia, koska mallineeni ei tarvitse niitä.



Kuva 2. YAML-mallineen rakenne AWS CloudFormation -käytössä (Techielass 2022)

3 Toiminnanohjausjärjestelmä

Toiminnanohjausjärjestelmä (Enterprise Resource Planning, ERP) on ohjelmistojärjestelmä, joka on suunniteltu auttamaan organisaatioita hallitsemaan, integroimaan ja automatisoimaan niiden perusliiketoimintaprosesseja. Nämä prosessit voivat kattaa useita eri osa-alueita kuten myynnin, markkinoinnin, hankinnan, varastonhallinnan, tuotannon, logistiikan, taloushallinnon, henkilöstöhallinnon ja asiakkuudenhallinnan. (Oracle s.a.)

Toiminnanohjausjärjestelmän tavoitteena on yhdistää eri liiketoimintaprosessit yhdeksi kokonaisvaltaiseksi järjestelmäksi, joka tarjoaa yhteisen datalähteen ja järjestelmäalustan organisaation sisällä. Tämä mahdollistaa tiedon yhtenäisen ja reaaliaikaisen jakamisen eri osastojen ja toimintojen välillä, mikä puolestaan parantaa organisaation sisäistä koordinaatiota, päätöksentekoa ja toiminnan tehokkuutta. Tämä tekee toiminnanohjausjärjestelmistä kriittisen osan nykyaikaisissa organisaatioissa, sillä ne tukevat liiketoimintaprosessien hallintaa ja yhtenäistävät tiedonkäsittelyä. (Oracle s.a.)

3.1 Odoo-toiminnanohjausjärjestelmä

Odoo on toiminnanohjausjärjestelmä, jonka kehitys alkoi vuonna 2005 belgialaisen yrityksen TinyERP:nä. Tuolloin sen perustajat Fabien Pinckaers ja hänen veljensä Gilles huomasivat markkinoiden puutteen täydellisestä toiminnanohjausjärjestelmästä. Tämän takia he päättivät kehittää oman järjestelmän vastaamaan heidän yrityksensä tarpeita. (Odoo s.a.)

Odoo on avoimen lähdekoodin järjestelmä, joka tarjoaa ratkaisuja kaikenkokoisille yrityksille myynnin, varaston, hankinnan, tuotannon, kirjanpidon ja henkilöstöhallinnon osa-alueille. Se on täysin modulaarinen, joten käyttäjät voivat valita vain ne ominaisuudet, joita he tarvitsevat. Tämän takia se on hyvin skaalautuva järjestelmä. (Synconics 2023)

Odoo on kehitetty Python-ohjelmointikielellä, ja sen arkkitehtuuri on modulaarinen. Tämä tekee siitä erittäin joustavan ja helpon muokata. Tämä on yksi syistä, miksi Odoo on avoimen lähdekoodin järjestelmänä. Kaiken kaikkiaan Odoo on monipuolinen ja skaalautuva toiminnanohjausjärjestelmä, joka on suunniteltu vastaamaan kaikkia yrityksen tarpeita. (Captivea s.a.)

3.2 Odoon ominaisuudet ja sovellukset

Odoon ominaisuudet ja sovellukset tarjoavat valikoiman työkaluja yrityksen eri toimintojen hallintaan. Asiakkuudenhallinta on yksi Odoon tärkeimmistä sovelluksista. Se auttaa yrityksiä hallitsemaan asiakassuhteitaan paremmin tarjoamalla työkaluja, kuten automatisoituja sähköpostikampanjoita, myyntisuppilon seurantaa ja asiakkaiden käyttäytymisen tilastoja. Näiden avulla yritykset

voivat rakentaa tarkempia asiakasprofiileja ja kehittää tehokkaampia myyntistrategioita. (Odoo s.a. b)

Odoon varastonhallintaratkaisu on toinen keskeinen sovellus, joka kattaa tuotteiden vastaanottamisen, varastoinnin ja toimituksen. Järjestelmä mahdollistaa varastojen tarkan seurannan, automaattisen tilauksen ja toimitusten hallinnan. Tämä auttaa vähentämään virheitä ja parantamaan toimitusketjun tehokkuutta. (Odoo s.a. c)

Taloushallinnon puolella Odoo sisältää laajan valikoiman rahoitus- ja kirjanpitotoimintoja. Esimerkiksi laskutuksen, palkanlaskennan, raportoinnin ja budjetoinnin hoitaminen on helppoa käyttöliittymän avulla, joka mahdollistaa nopean ja tarkan kirjanpidon. Reaaliaikaisen seurannan avulla yritykset voivat tehdä tietoon perustuvia päätöksiä talouden osalta. (Odoo s.a. d)

Henkilöstöhallinnon saralla Odoo tarjoaa työkaluja ja sovelluksia henkilöstöresurssien hallintaan ja työntekijöiden taitojen kehittämiseen. Rekrytointi, työajanseuranta, palkanlaskenta ja koulutus ovat kaikki osa Odoon henkilöstöhallinnon ratkaisua. Tämän ansiosta yritykset voivat parantaa työntekijöidensä tyytyväisyyttä ja sitoutumista. (Odoo s.a. e)

Lopuksi Odoon projektinhallintasovellus on tärkeä työkalu, joka auttaa yrityksiä hallitsemaan projektejaan tehokkaasti. Se tarjoaa työkaluja projektien suunnitteluun, seurantaan ja raportointiin sekä resurssien hallintaan. Yritykset voivat siten saavuttaa projektien tavoitteet aikataulun ja budjetin puitteissa. (Odoo s.a. f)

3.3 Odoon laitteistovaatimukset

Tässä osiossa käsittelen Odoon laitteistovaatimuksia käytettäessä aiemmin opinnäytetyössä esiteltyä AWS EC2:ta. Käsiteltävänä on nimenomaan sellainen ympäristö, jota käytetään asiakasdemojen ja testiympäristöjen toteuttamiseen. Laitteiston on pystyttävä takaamaan Odoon sujuva toiminta näissä olosuhteissa, jonka takia laitteiston ei tarvitse olla erityisen tehokas.

Valitsen tähän tarkoitukseen AWS EC2:n t3.small -instanssin, joka tarjoaa 2 virtuaalista prosessoria ja 2 GB RAM-muistia. AWS EC2 -instanssit on jaettu useisiin tehotasoihin, jotka on suunniteltu eri käyttötarkoituksiin. Kuten kuvassa 3 esitellään t-tyypin instanssit, kuten t3.small, ovat yleiskäyttöisiä ja tarjoavat tasapainoisen suorituskyvyn erilaisille työkuormille. T-tyypin instanssit on suunniteltu sovelluksille, jotka hyödyntävät ajoittain suurta CPU-tehoa, mutta suurimman osan ajasta ne käyttävät vähemmän CPU-resursseja. (Amazon Web Services 2018a)

Tämä on sopiva valinta Odoon kaltaiselle järjestelmälle, joka ei välttämättä vaadi jatkuvaa korkeaa suorituskykyä, mutta saattaa tarvita lisätehoa hetkittäin.

T3.small -instanssin 2 GB RAM-muisti riittää hyvin Odoon toimintaan, sillä se ei ole erityisen muistintehoinen sovellus, varsinkin kun se on tarkoitettu asiakasdemoihin ja testikäyttöön. (Ventor Tech 2021)

Name	vCPUs	Baseline Performance / vCPU	Memory	Price / Hour (Linux)	Price / Hour (Windows)
t3.nano	2	5%	0.5 GiB	$0.0052	$0.0098
t3.micro	2	10%	1 GiB	$0.0104	$0.0196
t3.small	2	20%	2 GiB	$0.0209	$0.0393
t3.medium	2	20%	4 GiB	$0.0418	$0.0602
t3.large	2	30%	8 GiB	$0.0835	$0.1111
t3.xlarge	4	40%	16 GiB	$0.1670	$0.2406
t3.2xlarge	8	40%	32 GiB	$0.3341	$0.4813

Kuva 1. EC2 instanssien t3-tyypin tehotasot (Amazon Web Services 2018a)

Odoo-toiminnanohjausjärjestelmän tallennustilavaatimukset ovat myös tärkeitä huomioon otettavia tekijöitä. Koska Odoo voi vaatia merkittävää tallennustilaa riippuen sen käyttötarkoituksesta kannattaa varata vähintään 10 GB:n tallennustilaa, joka tarjoaa riittävän kapasiteetin tietojen säilyttämiseen ja järjestelmän sujuvan toiminnan varmistamiseen.

On tärkeää muistaa, että AWS EC2 -instanssien valinnassa ja laitteistovaatimusten määrittämisessä tulee ottaa huomioon järjestelmän tarpeet ja käyttöolosuhteet. Tässä tapauksessa valittu t3.small -instanssi on sopiva ratkaisu, mutta suuremman kuormituksen tai erilaisten käyttötapauksien kohdalla voi olla tarpeen arvioida eri instanssityyppejä ja niiden tarjoamia resursseja.

4 Ubuntu-palvelimen valinta ja konfigurointi

Ubuntu on Linux-pohjainen käyttöjärjestelmä, joka on tunnettu luotettavuudestaan, helppokäyttöisyydestään ja joustavuudestaan. Yksi Ubuntun keskeisistä eduista on sen avoimen lähdekoodin luonne. Avoimen lähdekoodin ansiosta käyttäjät voivat tarkastella, muokata ja jakaa koodia vapaasti, mikä edistää innovointia ja jatkuvaa kehitystä. Tämä tarkoittaa, että Ubuntu hyötyy laajasta kehittäjäyhteisöstä, joka tarjoaa tukea, päivityksiä ja parannuksia järjestelmälle. (Hostinger 2023)

Ubuntu-palvelin tarjoaa erinomaisen suorituskyvyn ja vakauden komentorivipohjaisena versiona. Se on suunniteltu toimimaan tehokkaasti ja luotettavasti erilaisissa palvelinympäristöissä ja sen optimointi resurssien käytölle takaa, että palvelin toimii sujuvasti jopa raskaiden kuormitusten aikana. (Make use of 2022)

Ubuntu-palvelimen skaalautuvuus on erinomainen etu etenkin pilvipohjaisten palveluiden kanssa. Se integroituu vaivattomasti pilvipalveluihin, kuten AWS:ään mahdollistaen palvelimen laajentamisen kapasiteetin tarpeen mukaan. Tämä tekee siitä kustannustehokkaan ratkaisun, joka voi kasvaa ja kehittyä yrityksen tarpeiden mukaisesti. (Ubuntu 2023)

4.1 Ubuntu-palvelimen konfigurointi AWS CloudFormationin avulla

Ubuntu-palvelimen asentaminen AWS CloudFormationin avulla on järkevää, sillä se yhdistää parhaat puolet sekä Ubuntu-käyttöjärjestelmästä että Amazon Web Services -pilvipalvelusta. CloudFormation-mallineiden ansiosta voin automaattisesti luoda ja asettaa Ubuntu-palvelimen käyttöön, mikä tekee prosessista nopeamman ja vähentää manuaalista työtä. (Ubuntu s.a.)

Samaan aikaan voin määritellä verkkoyhteyksiä ja turvallisuusasetuksia, kuten VPC:t, jotka ovat eristettyjä virtuaalisia yksityisiä pilviympäristöjä, ja palomuurisääntöjä, jotka auttavat suojaamaan ubuntu-palvelimiani verkossa. (Amazon Web Services s.a. a)

4.2 Tietoturva- ja suorituskykyvaatimukset

Tietoturva- ja suorituskykyvaatimukset ovat tärkeässä asemassa luotaessa AWS CloudFormation YAML-mallinetta Ubuntu-palvelimelle, jossa on asennettuna Odoo-toiminnanohjausjärjestelmä. On tärkeää ottaa nämä vaatimukset huomioon, jotta voin varmistaa, että järjestelmä on suojattu.

Tietoturvan osalta on tärkeää määritellä tarkat palomuurisäännöt, jotka sallivat vain tarvittavan liikenteen pääsyn palvelimelle. Tämä tarkoittaa esimerkiksi pääsyn sallimista ainoastaan tiettyjen IP-osoitteiden tai IP-alueiden kautta. Tämä saavutetaan lisäämällä YAML-mallineeseen erilaisia turvaryhmiä (Security Groups).

Suorituskyvyn varmistamiseksi Odoo-toiminnanohjausjärjestelmän sujuvaan ja nopeaan toimintaan minun on varmistettava, että infrastruktuuri tarjoaa riittävän suorituskyvyn. Tämä tarkoittaa sopivan EC2-instanssin valitsemista, joka tarjoaa riittävän suorituskyvyn ja resurssit Odoo-toiminnanohjausjärjestelmän pyörittämiseen. Tämä kohta on määritelty aikaisemmassa opinnäytetyön kappaleessa, jossa valittiin EC2-instanssin tasoksi t3.small. Tietokannan kohdalla on tärkeää valita sellainen RDS-instanssi, joka täyttää Odoo-järjestelmän tietokantavaatimukset parhaiten. Tämä sisältää PostgreSQL:n käytön tietokantana sekä instanssin, joka tarjoaa optimaalisen suorituskyvyn ja kustannustehokkuuden yhdistelmän.

Tämä opinnäytetyö ei kuitenkaan käsittele tietokannan liittämistä RDS-instanssin PostgreSQL-tietokantaan vaan käytän työssäni palvelimeen paikallisesti asennettavaa PostgresSQL-tietokantaa.

5 Mallineen toteutus

Tämän opinnäytetyön tavoitteena on luoda toimiva AWS CloudFormation-malline Odoo-toiminnanohjausjärjestelmän asentamiseen Ubuntu-palvelimelle. Tämä projekti syntyi tarpeesta nopeuttaa ja tehostaa Odoo-asiakasdemojen luomista ja hallintaa yrityksessämme. Projektissa käytän perustyökaluina AWS CloudFormationia, YAML-mallinnusta ja Ubuntun Vim-tekstieditoria. Projektin toteuttaminen edellyttää yrityksemme tilauspohjaista AWS-palvelua sekä pääsyä asianmukaisiin teknisiin dokumentaatioihin ja aikaa sisäistää tarvittavat tekniikat.

Kohderyhmänä on ensisijaisesti yritys, jossa työskentelen ja sen tiimi, jolla on tarve käyttää tätä mallinetta erilaisiin tarkoituksiin, kuten asiakastapaamisiin, testiympäristöihin ja suorituskykytestaukseen. Mallineen avulla on mahdollista havainnollistaa Odoo-toiminnanohjausjärjestelmän ominaisuuksia, testata uusia ideoita ja suorittaa suorituskykytestejä. Lisäksi mallinetta voi tulevaisuudessa käyttää myös muut yrityksen tiimit tai henkilöt. Tämän lisäksi yrityksemme ulkopuolella olevat kohderyhmät, kuten muut Odoo-toimittajat, ohjelmistokehittäjät, IT-konsultit ja järjestelmäarkkitehdit, voivat hyödyntää tätä mallinetta omiin tarpeisiinsa. Rajoittavia tekijöitä ovat teknologian haasteet. Teknologian haasteet liittyvät erityisesti YAML-mallinuksen ja AWS CloudFormationin käytön erityispiirteisiin ja haasteisiin.

Mallineen toteutukselle asetan laadullisia kriteerejä, joiden avulla voin arvioida työn onnistumista ja sen tuomaa lisäarvoa Odoo-toiminnanohjausjärjestelmän käyttöönotossa. Tärkeimpänä kriteerinä ja tavoitteena on luoda toimiva YAML-malline, joka luo Ubuntu-palvelimen ja asentaa siihen Odoo-toiminnanohjausjärjestelmän automaattisesti. Tämä tarkoittaa sitä, että mallineen tulee sisältää kaikki tarvittavat konfiguraatiot ja riippuvuudet, jotta Odoo voi käynnistyä ja toimia odotetusti. Mallineen kriteerinä on myös nopeuttaa ja helpottaa Odoo-asiakasdemojen luomista ja hallintaa yrityksessämme. Mikäli mallineen avulla voi nopeasti ja vaivattomasti luoda toimivia Odoo-ympäristöjä se täyttää tämän kriteerin. Laadullisten kriteerien toteutumisen käyn läpi kappaleessa 5.3 Lopullisen tuotoksen esittely.

5.1 Tuotoksen tuottamisen kuvaus

Tuotoksen tuottamisen prosessi käynnistyi syvällisellä tutkimuksella AWS:n tarjoamista palveluista, sekä AWS CloudFormationin esimerkillisten YAML-mallineiden opiskelulla. Tämä prosessi sisälsi perusteellisen oppimisjakson AWS:n palveluista, kuten EC2, yrityksemme nimeämä netitysALB (Application Load Balancer) ja CloudFormation.

Opinnäytetyön kehittämistyön menetelmiä oli kolme, jotka yhdessä muodostivat lähestymistavan AWS CloudFormation YAML-mallineen luomiseen. Ensimmäinen näistä menetelmistä oli AWS-

dokumentaation tutkiminen ja hyödyntäminen. Tämä antoi minulle tarvittavan ymmärryksen AWS CloudFormation-palvelusta, mikä oli ratkaisevaa YAML-mallineen tehokkaalle suunnittelulle ja toteutukselle. Dokumentaation avulla sain selkeän käsityksen AWS CloudFormationin käytännöistä ja suositelluista menetelmistä, jotka auttoivat minua rakentamaan toimivan ja tehokkaan mallineen.

Toinen kehittämistyön menetelmä liittyi suunnitteluun ja mallineen kirjoittamiseen. Tässä vaiheessa Ubuntun Vim-tekstieditori osoittautui erittäin hyödylliseksi työkaluksi. Vim mahdollisti tarkan ja tehokkaan tavan kirjoittaa ja muokata YAML-koodia, joka määritti mallineen rakenteen. Mallineen luomisen työkaluksi valitsin Ubuntu-käyttöjärjestelmän Vim-työkalun.

Testaus ja arviointi olivat seuraavat menetelmät kehittämistyössäni. Kun malline oli kehitetty suoritin testauksen AWS CloudFormation-ympäristössä. Tämä varmisti, että malline toimi odotetusti ja kykeni luomaan Ubuntu-palvelimen ja asentamaan siihen Odoo-toiminnanohjausjärjestelmän onnistuneesti.

Aloitin YAML-mallineen luomisen perustason AWS-resursseilla, kuten EC2-instanssilla ja siihen liittyvillä parametreilla, kuvastettuna kuvassa 4. Tämä alkuperäinen mallinne muodosti perustan, jota kehitin iteratiivisesti lisäämällä uusia resursseja ja ominaisuuksia, kuten netitysALB:n, Route 53 DNS-konfiguraatiot ja tietoturva-asetukset.

```yaml
AWSTemplateFormatVersion: "2010-09-09"
Description: 'AWS Odoo '
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the bastion host
    Type: AWS::EC2::KeyPair::KeyName
    Default: efaws
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
Resources:
  InstanceName:
    Type: AWS::EC2::Instance
    Properties:
      SubnetId: !Ref Subnet
      InstanceType: t3.small
      KeyName: !Ref KeyName
      ImageId: ami-0f93e856d36a101f8
      BlockDeviceMappings:
        - DeviceName: "/dev/xvda"
          Ebs:
            VolumeSize: '10'
            Encrypted: 'true'
```

Kuva 2. YAML-mallineen ensimmäinen versio

Koko tämän kehitysprosessin ajan jokaisen uuden resurssin, tai ominaisuuden lisäämisen yhteydessä testasin mallineen AWS-ympäristössä. Kuvassa 5 esitetään yksi esimerkki

virheilmoituksesta, joka ilmeni yrittäessäni lähettää CloudFormationiin YAML-mallinetta, joka oli formatoitu virheellisesti.

```
20:36:04 pkauhtio@DESKTOP-2PD1D4D ~$ aws cloudformation create-stack --template-body file://oppari.yaml --parameters ParameterKey=ScheduleVal
ue,ParameterValue=manual ParameterKey=DNS,ParameterValue=oppariodoo1 ParameterKey=OdooVersion,ParameterValue=15 ParameterKey=PriorityNumber,P
arameterValue=2100 --stack-name oppariodoo1

An error occurred (ValidationError) when calling the CreateStack operation: Template format error: YAML not well-formed. (line 15, column 13)

20:36:10 pkauhtio@DESKTOP-2PD1D4D ~$
```

Kuva 3. Esimerkki virheellisen YAML-mallinteen lähettämisestä

Tällaiset virheet opettivat minua ymmärtämään, kuinka tärkeää on tarkistaa mallineen syntaksi ja rakenne ennen sen käyttöönottoa, mihin käytin AWS:n omaa validointityökalua kuvattuna kuvassa 6.

```
20:46:18 pkauhtio@DESKTOP-2PD1D4D ~$ aws cloudformation validate-template --template-body file://oppari.yaml

An error occurred (ValidationError) when calling the ValidateTemplate operation: Template format error: YAML not well-formed. (line 15, colum
n 13)
```

Kuva 4. Mallineen virhetarkastuskomento

Kuvassa 7 näkyy YAML-mallineen kehittyneempi versio, mutta se ei ole vielä lopullinen versio. Tämä malline kuvastaa kuitenkin merkittävää edistystä alkuperäiseen mallineeseen verrattuna.

```yaml
AWSTemplateFormatVersio: "2010-09-09"
Description: 'AWS Odoo '
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the bastion host
    Type: AWS::EC2::KeyPair::KeyName
    Default: aws-avain
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  Subnet:
    Description: Subnet to place the EC2 in
    Type: String
    Default: subnet-002454646233fhdfhbb97c5a02e
  ALBListener:
    Type: String
    Default: arn:aws:elasticloadbalancing:eu-west-1:5747457252153:listener/app/NetitysALB/3e36fdgdfgd343feb68ed/7c4642124a68bvc0b
Resources:
  EC2TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 30
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 15
      HealthyThresholdCount: 5
      Matcher:
        HttpCode: '200'
      Port: 8069
      Protocol: HTTP
      TargetGroupAttributes:
        - Key: deregistration_delay.timeout_seconds
          Value: '20'
      Targets:
        - Id: !Ref InstanceName
          Port: 8069
      UnhealthyThresholdCount: 3
      VpcId: vpc-01abcgvhfgd256
  InstanceName:
    Type: AWS::EC2::Instance
    Properties:
      SecurityGroupIds:
        - !Ref AllowNetitysJumper
        - !Ref AllowOdoo
        - !Ref AllowNetitys
      SubnetId: !Ref Subnet
      InstanceType: t3.small
      KeyName: !Ref KeyName
      ImageId: ami-3535466f93e856dfghdgh1f8
      BlockDeviceMappings:
        - DeviceName: "/dev/xvda
```

Kuva 7. Kehittyneempi YAML-malline

Tästä eteenpäin jatkoin kehittämistä iteratiivisesti, kunnes sain mallineen suorittamaan haluamani tuloksen. Tilanteessa, jossa jäin jumiin jatkoin aiheeseen liittyvän teknologian opiskelemista, kunnes sain työn etenemään.

5.2 Mallineen testaus

Testausprosessin tavoitteena oli varmistaa, että AWS CloudFormation-malline toimii oikein Odoo-toiminnanohjausjärjestelmän asentamisessa ja konfiguroinnissa sekä Ubuntu-palvelimen asennuksessa. Mallineen testaaminen sisälsi erilaisia vaiheita, joiden avulla pystyin varmistamaan mallineen toimivuutta.

Aloitin tarkistamalla YAML-mallineen syntaksin ja korjasin mahdolliset virheet ennen testaamista. Seuraavaksi ajoin AWS CloudFormation-mallineen ja tarkistin, että Odoo-toiminnanohjausjärjestelmä ja Ubuntu-palvelin asentuivat oikein.

Lisäksi varmistin, että kaikki määritellyt resurssit tuli luotua oikein. Sitten testasin Odoon suorituskykyä ja varmistin, että se on riittävä käyttäjäkokemuksen ja järjestelmän tehokkuuden kannalta. Tämän jälkeen arvioin, oliko EC2-instanssin taso sopiva Odoon vaatimuksiin nähden, ja tulin siihen päätökseen, että kappaleessa 3.3 Odoon laitteistovaatimukset valittu tehotaso oli toimiva.

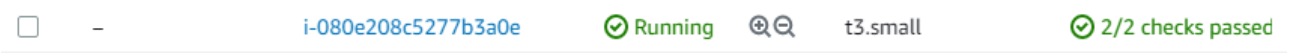
Seuraavaksi keskityin Odoon tietokanta- ja verkkoyhteyksien testaamiseen ja varmistin, että ne toimivat oikein. Sen jälkeen siirryn Odoon käyttäjähallinnan ja tietoturvan testaamiseen, jotta pystyin varmistaa, että asennusprosessi saavutti kaikki tarvittavat konfiguraatioasetukset ja integraatiot. Tarkastin myös, että levytila oli salattu Ubuntu-palvelimella.

Mikäli testausvaiheessa ilmeni ongelmia, korjasin ne ja toistin testit varmistaakseni, että malline toimii oikein. Testauksen jälkeen analysoin tulokset ja tein tarvittavia muutoksia mallineeseen parantaakseni sen toimivuutta ja suorituskykyä.

5.2.1 Testien tulokset ja niiden analysointi

Suoritin useita testejä varmistaakseni, että malline toimi oikein ja täytti laadulliset kriteerit mainittuna kappaleessa 5 Mallineen toteutus. Tässä yhteydessä käsittelen kolmea erityistä ongelmaa, jotka kohtasin testien aikana.

Ensimmäinen ongelma liittyi AWS EC2:n "Name"-tagiin. Testien aikana huomasin, että malline ei asettanut "Name"-tagia luomalleen EC2-instanssille, minkä vuoksi instanssilla ei ollut nimeä kuten esitetty kuvassa 8. Tämä saattoi vaikeuttaa instanssin tunnistamista ja hallintaa AWS-ympäristössä. Korjasin tämän ongelman lisäämällä tarvittavan komennon mallineeseen, joka määrittelee "Name"-tagin ja asettaa sille arvon. Näin ollen EC2-instanssilla on nyt selkeä nimi, mikä helpottaa sen hallintaa ja tunnistamista, jonka voi havaita kuvassa 9.



Kuva 8. EC2 instanssi ilman Name Tagia määriteltynä

Kuva 9. EC2 instanssi määriteltynä Name Tagin kanssa

Testausprosessin aikana kohtasin myös useita ongelmia liittyen YAML-mallineen formatointiin. YAML-kieli on erittäin tarkka syntaksiltaan ja sisennyksiltään, mikä tarkoittaa, että pienetkin virheet voivat aiheuttaa ongelmia mallineen toiminnassa. Käytännössä jouduin korjaamaan useita formatointivirheitä mallineen eri osissa.

Koska formatointiongelmia oli paljon ja ne olivat pääasiassa teknisiä yksityiskohtia, en käy niitä tässä yhteydessä läpi yksitellen. Kuitenkin on tärkeää mainita, että nämä ongelmat olivat merkittävä osa testausprosessia, ja niiden korjaaminen auttoi minua saamaan mallineen toimimaan oikein.

Testausprosessin aikana huomasin erään tärkeän tietoturvaan liittyvän seikan, joka oli jäänyt huomioimatta mallineen toteutuksessa. Kun malline loi Ubuntu-palvelimen sen levytila ei ollut salattu. Tämä on merkittävä tietoturvariski, sillä tietojen salaamaton tallentaminen voi mahdollistaa niiden luvattoman käytön ja vuotamisen.

Mielenkiintoisesti huomasin, että YAML-malline suostui salaamaan levyn, kun sitä käytettiin Amazon Linux 2-käyttöjärjestelmän luomiseen. Tästä huolimatta malline ei suostunut kryptaamaan levyä, kun sitä käytettiin Ubuntu-palvelimen luomiseen. Tämä rajoitus voi johtua erosta Amazon Linux 2:n ja Ubuntun välillä ja siitä, miten ne käsittelevät levytilan salaamista, mutta tarkkaa syytä en saanut selville.

Amazon Machine Image (AMI) on pakattu kokoelma, joka sisältää tiedot tarvittavasta käyttöjärjestelmästä, palvelinohjelmistoista, sekä niiden asetuksista, joita tarvitaan EC2-instanssin luomiseen. Yksinkertaisesti sanottuna AMI toimii pohjana, josta EC2-instanssi luodaan ja käynnistetään. (Amazon Web Services s.a. r)

Tietoturvan parantamiseksi päätin lisätä mallineeseen kohdan ImageId, joka määrittelee itse luotuni Amazon Machine Image (AMI) -kuvan, jossa on salattu levytila. Tämä tarkoittaa, että kun malline luo uuden EC2-instanssin Ubuntulla, se käyttää tätä salattua AMI:a, jolloin myös uuden instanssin levytila on salattu. Tämä parantaa merkittävästi palvelimen tietoturvaa ja suojaa tallennettuja tietoja luvattomalta käytöltä.

Muutoksen jälkeen suoritin testit uudelleen ja varmistin, että uusi EC2-instanssi luotiin käyttäen salattua AMI:a, ja että sen levytila oli salattu. Tämä muutos auttoi parantamaan järjestelmän tietoturvaa, ja vastasi paremmin tietosuojaa koskeviin vaatimuksiin.

5.3 Lopullisen tuotoksen esittely

Tämän opinnäytetyön lopputuloksena sain luotua toimivan YAML-mallineen, joka mahdollistaa Odoo-toiminnanohjausjärjestelmän automaattisen käyttöönoton Ubuntu-palvelimella AWS CloudFormationin avulla. Lopullinen YAML-malline on esitelty tämän opinnäytetyön liitteissä.

YAML-malline on kehitetty iteratiivisesti, ja jokainen kehitysaskel on testattu AWS-ympäristössä. Tämä on varmistanut, että jokainen lisätty resurssi tai ominaisuus toimii oikein ja on yhteensopiva

muiden resurssien kanssa. Malline on kehittynyt useiden kehitys- ja testausvaiheiden kautta, joista alkuperäinen ja kehittyneempi versio esiteltiin aiemmin.

Lopullinen YAML-malline sisältää kaikki tarvittavat resurssit ja asetukset Odoo-toiminnanohjausjärjestelmän automaattiseen käyttöönottoon. Tämä sisältää EC2-instanssin, netitysALB:n, Route 53 DNS-konfiguraatiot ja tietoturva-asetukset. YAML-malline on vastaus yrityksemme tarpeeseen nopeuttaa ja tehostaa Odoo-asiakasdemojen luomista ja hallintaa. Se tarjoaa yksinkertaisen ja automatisoidun tavan käyttöönottoon, joka säästää aikaa ja resursseja. Lopullinen malline helposti ymmärrettävä ja käytettävä, minkä vuoksi se ei ole arvokas resurssi vain meidän yrityksellemme, vaan myös muille tahoille, kuten muille Odoo-toimittajille, ohjelmistokehittäjille ja IT-konsulteille.

Malline on luotu ja testattu huolellisesti, ja se on osoittautunut toimivaksi ja tehokkaaksi työkaluksi Ubuntu-palvelimen luomisessa ja Odoo-toiminnanohjausjärjestelmän asennuksessa. Kaikki tarvittavat konfiguraatiot ja riippuvuudet ovat olleet mukana, ja Odoo on käynnistynyt ja toiminut odotetusti. Tämä osoittaa, että malline on täyttänyt tärkeimmän kriteerin ja tavoitteen: se on toimiva ja luotettava työkalu, joka mahdollistaa Odoo-toiminnanohjausjärjestelmän nopean käyttöönoton.

Mallineen avulla voi myös nopeasti ja vaivattomasti luoda toimivia Odoo-ympäristöjä asiakasdemoja varten. Tämä on ollut merkittävä käytettävyyden parannus, joka on helpottanut Odoo-ympäristöjen hallintaa ja käyttöä. Tämä osoittaa, että malline on täyttänyt tämän kriteerin ja onnistunut tehtävässään. Näiden havaintojen perusteella voin todeta, että YAML-malline on onnistunut täyttämään asettamani laadulliset kriteerit ja tavoitteet, jotka määrittelin kappaleessa 5 Mallineen toteutus.

Mallinetta voi helposti mukauttaa ja laajentaa tulevaisuudessa. Tämä voi sisältää uusien resurssien, tai ominaisuuksien lisäämisen, tai olemassa olevien resurssien muokkaamisen. Malline tarjoaa myös mahdollisuuden ottaa nopeasti käyttöön useita Odoo-instansseja samanaikaisesti. Tämä ominaisuus voi olla erittäin hyödyllinen yritykselle, joka tarvitsee useita erilaisia Odoo-ympäristöjä eri tarkoituksiin, kuten kehitykseen, testaukseen ja tuotantoon.

Yhteenvetona voin todeta, että lopullinen YAML-malline on laadukas työkalu, joka täyttää asetetut vaatimukset ja tavoitteet. Se on toimiva, tehokas, joustava ja helposti ymmärrettävä ja käytettävä. Se on arvokas resurssi yrityksellemme, mutta se voi myös olla hyödyllinen muille tahoille, jotka ovat kiinnostuneita Odoo-toiminnanohjausjärjestelmän ja AWS CloudFormationin käytöstä. Opinnäytetyössä liitteenä oleva malline antaa lukijalle mahdollisuuden tutkia sitä tarkemmin, ja hyödyntää sitä omassa työssään.

5.3.1 Mallineen komponentit ja resurssit

Tässä opinnäytetyön osiossa käyn läpi mitä kukin mallineen komponentti tekee, ja kuinka ne yhdessä muodostavat AWS-infrastruktuurin. AWS CloudFormation-mallineet alkavat usein kahdella komponentilla: AWSTemplateFormatVersion ja Description kuten esitettynä kuvassa 10. Nämä komponentit esiteltiin myös teoriassa, jossa käsiteltiin mallineiden rakennetta.

AWSTemplateFormatVersion-komponentti määrittää mallineen AWS CloudFormatio -formaatin version. Tällä hetkellä on olemassa vain yksi versio, joka on "2010-09-09". Tämä tieto on tarpeen, koska tulevaisuudessa AWS saattaa julkaista uusia versioita CloudFormation-formaatista, jolloin tämä kenttä mahdollistaa vanhempien mallineiden yhteensopivuuden. Tämä komponentti on valinnainen, mutta sen sisällyttäminen varmistaa, että mallinne on yhteensopiva tietyn CloudFormation-palvelun version kanssa.

Description-komponentti on myös valinnainen. Se tarjoaa mahdollisuuden lisätä lyhyen kuvauksen mallineesta. Kuvaus auttaa hahmottamaan, mitä malline tekee, tai mitä resursseja se luo, tai hallitsee. Esimerkiksi tässä tapauksessa Description: 'AWS Odoo' viittaa siihen, että malline liittyy Odoo-toiminnanohjausjärjestelmään AWS-ympäristössä. Kuvaus auttaa ylläpitäjiä ja kehittäjiä ymmärtämään mallineen tarkoituksen ilman, että heidän tarvitsee tutkia sen yksityiskohtaisia määritelmiä.

```
AWSTemplateFormatVersion: "2010-09-09"
Description: 'AWS Odoo'
```

Kuva 10. AWSTemplateFormatVersion-, ja Description -komponentit mallineen alussa

Parameters-komponentti tarjoaa mahdollisuuden syöttää dynaamisia arvoja mallineen käyttöönoton yhteydessä. Tämä ominaisuus mahdollistaa mallineen mukauttamisen erilaisiin käyttötarkoituksiin ja konfiguraatioihin, ja se tekee mallineesta joustavan työkalun resurssien hallinnassa. Mallineessa asetetaan kaksitoista eri parametriä, kuten kuvattuna kuvassa 11.

Mallineessa esiintyvät parametrit on suunniteltu tiettyä tarkoitusta varten, ja ne vastaavat tietyntyyppisiä arvoja. Esimerkiksi "KeyName"-parametri on suunniteltu vastaanottamaan EC2-avainparin nimi, joka mahdollistaa SSH-yhteyden ”jumpperiin”. Parametrin tyyppi, AWS::EC2::KeyPair::KeyName, tarkoittaa, että se odottaa EC2-avainparin nimeä. Tämä parametri on pakollinen turvallisen ja suojatun SSH-yhteyden luomiseksi palvelimelle.

"Subnet"-parametri määrittelee, mihin aliverkkoon EC2-instanssi sijoitetaan. Tämä on tärkeää, koska se määrittää, missä virtuaaliverkon osassa instanssi toimii, mikä voi vaikuttaa sen saatavuuteen, suorituskykyyn ja turvallisuuteen.

"HostedZone"-parametri on tärkeä AWS:n Route53-palvelun kannalta, koska se määrittelee, mihin isännöintivyöhykkeeseen instanssi sijoitetaan. Route53 on Amazonin skaalautuva ja erittäin saatavilla oleva verkkotunnusjärjestelmä (DNS) palvelu, mutta opinnäytetyö ei sisällä sitä tarkemmin, joten en mene sen yksityiskohtiin.

"OdooVersion"-parametri määrittää, minkä version Odoo-toiminnanohjausjärjestelmästä malline asentaa.

"PriorityNumber"-parametri on AWS CloudFormationissa käytetty parametri, jolla määritellään Application Load Balancerin kuuntelijasääntöjen prioriteetti. Tämä numero voi vaihdella välillä 1-50000, ja jokaisella kuuntelijasäännöllä on oltava oma ainutlaatuinen prioriteettinsa.

"ScheduleValue" puolestaan määrittelee aikataulun, jonka mukaan tietyt toiminnot tapahtuvat. Esimerkiksi, voi määrittää tietyn ajan päivässä, jolloin resursseja päivitetään tai luodaan. Tämä voi olla hyödyllistä esimerkiksi, jos haluaa minimoida haitalliset vaikutukset liiketoiminnan kriittisiin tunteihin, kuten luomalla tai päivittämällä resursseja vain liiketoiminnan ulkopuolisina aikoina. Tässä mallineessa kuitenkin tämän parametrin arvoon on asetettu ”manual”, joka tarkoittaa, että resurssien luominen tai päivitys on manuaalista, eikä tapahdu automaattisesti.

```
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the bastion host
    Type: AWS::EC2::KeyPair::KeyName
    Default: efaws
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  Subnet:
    Description: Subnet to place the EC" in
    Type: String
    Default: subnet-0022680bb97c5a02e
  ALBListener:
    Type: String
    Default: arn:aws:elasticloadbalancing:eu-west-1:4436366357893:listener/app/NetitysALB/3e36436363fdhb68ed/7c478124346368
  AllowNetitysJumper: #Aseta Security Group
    Type: String
    Default: sg-0egfjfjn85ghfjfghjb8fc0c55
  AllowOdoo: #Aseta Security Group
    Type: String
    Default: sg-0a59jhfgmk35e17009ee
  AllowNetitys: #Aseta Security Group
    Type: String
    Default: sg-029dfhk6454ae0e
  HostedZone: #Määrittelee mihin hosted zoneen instanssi asetetaan
    Type: String
    Default: Z26IFDGSDGSG52216
  VPC:
    Type: String
    Default: vpc-01ab3adsgfd6348f0
  DNS:
    Type: String #Määrittelee palvelimen nimen rivi 70. Nimi asetetaan cloudformation create-stack komennon parametrissä.
  OdooVersion:
    Type: String #Määrittelee Odoon version esim. 13, 14. Numero asetetaan cloudformation create-stack komennon parametrissä.
  PriorityNumber:
    Type: String
  ScheduleValue:
    Type: String
```

Kuva 11. Mallineen Parameters-komponentti

Resources-komponentti on AWS CloudFormation-mallineen keskeinen elementti. Tämä osio sisältää tiedot kaikista AWS-resursseista, joita mallineen avulla luodaan, tai hallitaan. Resurssit on kuvattu erikseen, ja kullakin on omat määritelmänsä.

Tässä osiossa tarkastelen mallineen Resources-komponenttia, joka koostuu eri alikomponenteista, mukaan lukien EC2TargetGroup, ListenerRule1, InstanceName, OdooPublicDNS ja OdooNetitysDNS. Jokainen näistä alikomponenteista edustaa yksittäistä resurssia, joka luodaan, tai hallitaan AWS-pilviympäristössä. Seuraavaksi tarkastelen jokaista Resources-osion alikomponenttia yksityiskohtaisemmin, ja selitän niiden toimintaa ja roolia AWS-infrastruktuurin luomisessa.

EC2TargetGroup-alikomponentti on osa Resources-komponenttia mallineessani. Tämä alikomponentti hallinnoi tulevan verkkoliikenteen reititystä EC2-instanssille. Tämä komponentti on osa opinnäytetyön tietoperustassa selitettyä AWS:n Elastic Load Balancing (ELB)-palvelua ja se määrittää tarkemmin, kuinka ELB ohjaa liikennettä EC2-instanseille.

Kuten kuvassa 12 näytetään EC2TargetGroupin konfiguraatio määritellään sen Properties-osiossa. Yksi merkittävä konfiguraatio on terveystarkastukset (Health Checks), jotka ovat automaattisia testejä, joita ELB suorittaa EC2-instanseille varmistaakseen, että ne ovat toiminnassa ja kykenevät

vastaanottamaan ja käsittelemään liikennettä. Jos EC2-instanssi ei läpäise terveystarkastusta, ELB ei ohjaa liikennettä sille.

Terveystarkastukset suoritetaan tässä tapauksessa 30 sekunnin välein (HealthCheckIntervalSeconds) ja ne käyttävät HTTP-protokollaa (HealthCheckProtocol). Terveystarkastuksen aikakatkaisu on asetettu 15 sekuntiin (HealthCheckTimeoutSeconds), ja jos viisi peräkkäistä tarkastusta (HealthyThresholdCount) on onnistunutta niin kohde katsotaan terveeksi. Terveystarkastus odottaa HTTP 200 -vastauksen (Matcher: HttpCode), mikä tarkoittaa, että instanssin on pystyttävä palauttamaan onnistunut HTTP-vastaus.

Verkkoliikenne ohjataan porttiin 8069 (Port), joka on määritetty sekä EC2TargetGroupin konfiguraatiossa, että TargetGroupin kohteissa (Targets). Tämä on sen takia, että mallineessa asennettavan Odoo-toiminnanohjausjärjestelmän oletusportti verkkoliikenteelle on 8069.

```
Resources:
  EC2TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 30
      HealthCheckProtocol: HTTP
      HealthCheckTimeoutSeconds: 15
      HealthyThresholdCount: 5
      Matcher:
        HttpCode: '200'
      Port: 8069
      Protocol: HTTP
      TargetGroupAttributes:
        - Key: deregistration_delay.timeout_seconds
          Value: '20'
      Targets:
        - Id: !Ref InstanceName
          Port: 8069
      UnhealthyThresholdCount: 3
      VpcId: vpc-01ab3af124ea8f8f0
      Tags:
        - Key: Name
          Value: EC2TargetGroup
        - Key: Port
          Value: 8069
```

Kuva 12. Mallineen EC2TargetGroup-komponentti

ListenerRule 1-alikomponentti liittyy myös AWS:n Elastic Load Balancing (ELB)-palveluun ja tarkemmin Application Load Balancerin (ALB) kuuntelijasääntöihin (Listener Rules). Kuuntelijasäännöt määrittävät, kuinka ALB reagoi saapuviin yhteydenottoihin ja kuinka ne ohjataan eteenpäin.

ListenerRule1 ohjaa saapuvan liikenteen palvelimelle, joka on määritelty EC2TargetGroupissa. Tämä tarkoittaa, että jonkun yrittäessä ottaa yhteyttä mallineessa luotuun toiminnanohjausjärjestelmään ALB näkee tämän, tarkistaa ListenerRule1:n säännöt ja ohjaa yhteyden oikealle palvelimelle. Tämä on erittäin tärkeää, koska se varmistaa, että käyttäjät pääsevät oikeaan paikkaan, ja että liikenne jakautuu tehokkaasti, jos käytössä on useita palvelimia. Lisäksi tämän säännön avulla voi määritellä, miten järjestelmä reagoi erilaisiin yhteydenottopyyntöihin. Lyhyesti sanottuna, ListenerRule1 on kuin liikenteenohjaaja Odoolle. Se ohjaa saapuvan liikenteen oikeaan paikkaan ja varmistaa, että kaikki sujuu hyvin. Tämä alikomponentti on esitelty kuvassa 13.

```
ListenerRule1:
  Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
  Properties:
    Actions:
      - Type: forward
        TargetGroupArn: !Ref EC2TargetGroup
    Conditions:
      - Field: host-header
        HostHeaderConfig:
          Values:
            - !Sub ${DNS}.netitys.fi
    ListenerArn: !Ref ALBListener
    Priority: !Ref PriorityNumber
```

Kuva 13. ListenerRule1-alikomponentti

InstanceName-alikomponentti on seuraava osio Resources-komponenttia, ja tämä osio sisältää myös omia alikomponenttejansa, jotka ohjaavat mallineen EC2-instanssin luontia, asennusta ja konfigurointia.

InstanceName-resurssi on tyypiltään AWS::EC2::Instance, joka tarkoittaa, että se määrittelee yksittäisen virtuaalisen palvelimen AWS:n EC2-palvelussa. Kuten kuvassa 14 näkyy mallineen MetaData-osiossa on määritelty AWS::CloudFormation::Init-asetukset, jotka antavat yksityiskohtaisia ohjeita instanssin asennuksesta ja konfiguroinnista. Nämä asetukset sisältävät erilaisia komentoja, jotka suoritetaan tietyssä järjestyksessä instanssin luonnin yhteydessä.

```
Metadata:
  AWS::CloudFormation::Init:
    configSets:
      OnCreation: #Määrittelee missä järjestyksessä eri vaiheet asennuksesta tapahtuvat.
        - hup
        - preinstall
      OnUpdate:
        - configs
        - aptupdate
        - install
    hup:
      files:
        /etc/cfn/cfn-hup.conf:
          content: !Sub |
            [main]
            stack=${AWS::StackId}
            region=${AWS::Region}
            interval=1
          mode: '000400'
          owner: root
          group: root
        /etc/cfn/hooks.d/cfn-auto-reloader.conf:
          content: !Sub |
            [cfn-auto-reloader-hook]
            triggers=post.update
            path=Resources.InstanceName.Metadata.AWS::CloudFormation::Init
            action=/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource InstanceName --configsets OnUpdate --region ${AWS::Region}
            runas=root
          mode: "000400"
          owner: "root"
          group: "root"
      services:
        sysvinit:
          cfn-hup:
            enabled: true
            ensureRunning: true
            files:
            - '/etc/cfn/cfn-hup.conf'
            - '/etc/cfn/hooks.d/cfn-auto-reloader.conf'
```

Kuva 14. Mallineen AWS::CloudFormation::Init-osio

OnCreation- ja OnUpdate-konfiguraatiosarjojen alla näkyy joukko eri vaiheita, kuten "hup", "preinstall", "configs", "aptupdate" ja "install". Nämä vaiheet suoritetaan instanssin luonnin yhteydessä.

Kuvassa 15 näkyy, kuinka "preinstall"-vaiheessa suoritetaan joukko komentoja, jotka hakevat ja asentavat tarvittavia ohjelmistoja ja paketteja, kuten Odoon avaintiedoston, jolla tarkistetaan ladatun Odoon eheys, ja tarvittavia moduuleja toiminnanohjausjärjestelmään. Nämä tarvittavat moduulit mahdollistavat verkkolaskutuksen Odoo-toiminnanohjausjärjestelmässä, joka ei normaalisti olisi mahdollista. Moduulit asennetaan, jotta näiden toiminnallisuus voidaan esitellä mallineen luomissa asiakasdemoissa. Tässä vaiheessa myös html2text-ohjelma asennetaan, jota vaaditaan tiettyjen tarvittavien moduulien toimintaan.

```
preinstall:
  commands:
    01_get_key:
      command: wget -O - https://nightly.odoo.com/odoo.key | apt-key add -
    02_get_maventa_module:
      command: wget https://packages.s3.eu-west-1.amazonaws.com/moduulit.15.03.tar.gz
    03_unpack_module2:
      command: tar -xzf moduulit.15.03.tar.gz -C /
    04_wkhtmltopdf:
      command: wget https://github.com/wkhtmltopdf/packaging/releases/download/0.12.6-1/wkhtmltox_0.12.6-1.focal_amd64.deb
    05_wkhtmltopdf_install:
      command: apt install -y ./wkhtmltox_0.12.6-1.focal_amd64.deb
    06_install_html2text:
      command: apt install -y html2text
  packages:
    apt:
      postgresql: []
      python3-boto3: []
      python3-requests: []
      html2text: []
```

Kuva 15. Mallineen preinstall-osio

Samoin "install"-vaiheessa suoritetaan sarja komentoja, kuten kuvattuna kuvassa 16. Nämä komennot asentavat Odoo-ohjelmiston, asettavat instanssin hostnamen, ja käynnistävät PostgreSQL- ja Odoo-palvelut.

```
install:
  packages:
    apt:
      odoo: []
  commands:
    01_set_hostname:
      command: !Sub hostname ${DNS}.netitys
    02_update_hostname_file:
      command: !Sub echo ${DNS}.netitys > /etc/hostname
    03_enable_postgres:
      command: systemctl enable postgresql
    04_start_postgres:
      command: systemctl start postgresql
    05_enable_odoo:
      command: systemctl enable odoo
    06_start_odoo:
      command: systemctl start odoo
```

Kuva 16. Mallineen install-osio

Kuvan 17 Properties-osiossa määritellään instanssin perusasetukset, kuten sen turvallisuusryhmät, aliverkko, instanssityyppi, avainpari, tagit ja käytetty AMI-kuva (Amazon Machine Image).

```yaml
Properties:
  SecurityGroupIds:
    - !Ref AllowNetitysJumper
    - !Ref AllowOdoo
    - !Ref AllowNetitys
  SubnetId: !Ref Subnet
  InstanceType: t3.small
  KeyName: !Ref KeyName
  Tags:
    - Key: "Schedule"
      Value: !Ref ScheduleValue
  ImageId: ami-45664d3dfsfs6a1
  BlockDeviceMappings:
    - DeviceName: "/dev/xvda"
      Ebs:
        VolumeSize: '10'
        Encrypted: 'true'
```

Kuva 17. InstanceName-komponentin Properties-osuus

UserData-osiossa määritellään skripti, joka suoritetaan EC2-instanssin käynnistymisen yhteydessä. Tämä toimii käynnistyskomentona jokaiselle uudelle EC2-instanssille, joka luodaan tämän mallineen perusteella.

Skripti aloittaa päivittämällä instanssin käyttöjärjestelmän käyttämällä komentoja 'apt-get -qqy update' ja 'apt-get -qqy dist-upgrade'. Tämän jälkeen se poistaa tarpeettomat paketit, ja puhdistaa paketinhallinnan välimuistin komennolla 'apt-get -qqy autoremove' ja 'apt-get -qqy autoclean'.

Seuraavaksi skripti asentaa tarvittavia Python-paketteja, kuten 'python3-pip', 'python3-setuptools' ja 'python3-testresources'. Tämän jälkeen se asentaa AWS CloudFormation Bootstrap -paketin komennolla ”pip3 install https://netitys-packages.s3.eu-west-1.amazonaws.com/aws-cfn-bootstrap-py3-latest.tar.gz”.

Tämän jälkeen skripti luo '/opt/aws/bin' -hakemiston ja luo siihen linkit moniin CloudFormationin tiedostoihin, kuten 'cfn-hup', 'cfn-init', 'cfn-signal'. Nämä työkalut auttavat hallitsemaan ja seuraamaan CloudFormation-instanssin tilaa ja toimintoja.

Skripti linkittää myös CloudFormation HUP-palvelun 'init.d' -hakemistoon ja tekee siitä suoritettavan. Tämä palvelu pitää huolta, että CloudFormation-instanssi päivittyy automaattisesti, jos mallineen pinossa tehdään muutoksia.

Sitten skripti suorittaa 'cfn-init'-komentoja, jotka suorittavat CloudFormationin määrittelemät configSetit, nimeltään 'OnCreation' ja 'OnUpdate'. Nämä vaiheet on määritetty aikaisemmin CloudFormation-mallineessa. Lopuksi 'cfn-signal'-komento lähettää signaalin CloudFormationiin kertoen, että instanssi on käynnistynyt onnistuneesti, kuten kuvassa 18 näkyy.

```
UserData:
  Fn::Base64: !Sub |
    #!/bin/bash -xe
    apt-get -qqy update
    apt-get -qqy dist-upgrade
    apt-get -qqy autoremove
    apt-get -qqy autoclean
    #          apt-get -qqy install curl wget
    apt-get -qqy install python3-pip python3-setuptools python3-testresources
    pip3 install https://packages.s3.eu-west-1.amazonaws.com/aws-cfn-bootstrap-py3-latest.tar.gz
    #          sleep 5
    mkdir -p /opt/aws/bin
    ln -s /usr/local/bin/cfn-hup /opt/aws/bin/cfn-hup
    ln -s /usr/local/bin/cfn-init /opt/aws/bin/cfn-init
    ln -s /usr/local/bin/cfn-signal /opt/aws/bin/cfn-signal
    ln -s /usr/local/bin/cfn-elect-cmd-leader /opt/aws/bin/cfn-elect-cmd-leader
    ln -s /usr/local/bin/cfn-get-metadata /opt/aws/bin/cfn-get-metadata
    ln -s /usr/local/bin/cfn-send-cmd-event /opt/aws/bin/cfn-send-cmd-event
    ln -s /usr/local/bin/cfn-send-cmd-result /opt/aws/bin/cfn-send-cmd-result
    #sleep 5
    ln -s /usr/local/init/ubuntu/cfn-hup /etc/init.d/cfn-hup
    chmod +x /usr/local/init/ubuntu/cfn-hup
    update-rc.d cfn-hup defaults
    echo set bell-style none >> /etc/inputrc
    echo set vb >> /etc/vim/vimrc
    /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource InstanceName --configsets OnCreation --region ${AWS::Region}
    /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource InstanceName --configsets OnUpdate --region ${AWS::Region}
    /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource InstanceName --region ${AWS::Region}
```

Kuva 18. UserData-osio

Mallineeni viimeisessä kohdissa on kuvassa 19 esitetyt OdooPublicDNS ja OdooNetitysDNS osiot, jotka määrittelevät Amazonin AWS Route 53 -nimipalvelun tietueet (RecordSetGroup). Nämä tietueet tarjoavat yhteyden Amazonin AWS-ympäristössä toimivaan Odoo-toiminnanohjausjärjestelmään. OdooPublicDNS asettaa joukon DNS-tietueita julkista verkkotunnusta varten. Tässä dokumentissa mainittu verkkotunnus "${DNS}.netitys.fi" on esimerkki, joka oikeassa käytössä korvataan todellisella DNS-nimellä. Käytännössä tämä tapahtuu create-stack-komennon yhteydessä annettavalla parametrilla. Tällöin esimerkkinä palvelimen verkko-osoite voisi olla "oppari.netitys.fi". Tämä DNS-tietue linkittää kyseisen verkkotunnuksen Amazonin Application Load Balanceriin (ALB), joka hallitsee liikennettä Odoo-toiminnanohjausjärjestelmän ja käyttäjien välillä.

OdooNetitysDNS määrittelee DNS-tietueen sisäistä liikennettä varten. Se yhdistää "${DNS}.netitys"-verkkotunnuksen EC2-instanssin yksityiseen IP-osoitteeseen. Tämä on ns. A-tietue, joka suorittaa tavallisen DNS-nimeämisen IP-osoitteeksi. Julkinen DNS-tietue on tärkeä käyttäjien pääsyn kannalta, kun taas sisäinen DNS-tietue mahdollistaa EC2-instanssin saavutettavuuden sisäisesti AWS-infrastruktuurissa.

```
OdooPublicDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneId: !Ref HostedZone
    RecordSets:
    - Name: !Sub ${DNS}.netitys.fi.
      Type: A
      AliasTarget:
        DNSName: NetitysALB-43602304.eu-west-1.elb.amazonaws.com
        HostedZoneId: 2323201SDGHSHSHQLNVNC
OdooNetitysDNS:
  Type: AWS::Route53::RecordSetGroup
  Properties:
    HostedZoneId: RHFBNGC8I3553ESAFGC
    RecordSets:
    - Name: !Sub ${DNS}.netitys.
      Type: A
      TTL: 300
      ResourceRecords:
        - !GetAtt InstanceName.PrivateIp
```

Kuva 19. OdooPublicDNS ja OdooNetitysDNS

Mallineen lopullisen version voi tarkistaa opinnäytetyön Liitteet-osiosta, mutta huomaa, että kaikki arkaluonteiset tiedot, kuten avaimet, salasanat tai tunnistetiedot, on piilotettu tai viitattu turvallisesti, eikä niitä ole paljastettu tässä mallineessa.

5.4 Mallineen käyttö

Kun AWS CloudFormation käsittelee mallineen, se tekee useita toimintoja varmistaakseen sujuvan ja tehokkaan prosessin. Ensimmäiseksi se tarkistaa mallineen syntaksin ja validoi sen. Mikäli mallineessa on virheitä, prosessi pysähtyy tässä vaiheessa ja CloudFormation palauttaa virheilmoituksen, joka auttaa käyttäjää korjaamaan ongelman. (Amazon Web Services s.a. s)

Kun malline on oikein muotoiltu ja validi, CloudFormation aloittaa uuden pinon luomisen. Tällöin se alkaa luoda mallineessa määriteltyjä resursseja määritellyssä järjestyksessä. Prosessin sujuvuuden varmistamiseksi, jos jokin resurssien luonti epäonnistuu, CloudFormation lopettaa prosessin ja antaa käyttäjälle ilmoituksen, jossa selitetään, missä ja miksi virhe tapahtui. (Amazon Web Services s.a. t)

Onnistuneen prosessin seurauksena, kun kaikki resurssit on luotu onnistuneesti, CloudFormation merkitsee pinon luoduksi ("CREATE_COMPLETE"), kuten kuvassa 21 näkyy. Tässä vaiheessa

kaikki määritellyt resurssit, kuten EC2-instanssit, tietokannat ja kuormantasausjärjestelmä ovat käyttäjän käytettävissä.

AWS CloudFormation-palvelussa uuden pinon (stack) luomiseen käytetään create-stack-komentoa. Kuvassa 20 on esitetty käytetty komento.

```
19:31:41 pkauhtio@DESKTOP-2PD1D4D ~$ aws cloudformation create-stack --templ
ate-body file://odoo.yaml --parameters ParameterKey=ScheduleValue,ParameterV
alue=manual ParameterKey=DNS,ParameterValue=oppariodoo ParameterKey=OdooVers
ion,ParameterValue=15 ParameterKey=PriorityNumber,ParameterValue=2100 --stac
k-name oppariodoo
{
    "StackId": "arn:aws:cloudformation:eu-west-1:407225052153:stack/opparied
oo/f6629f60-f274-11ed-be84-020d07d89305"
}
19:32:48 pkauhtio@DESKTOP-2PD1D4D ~$
```

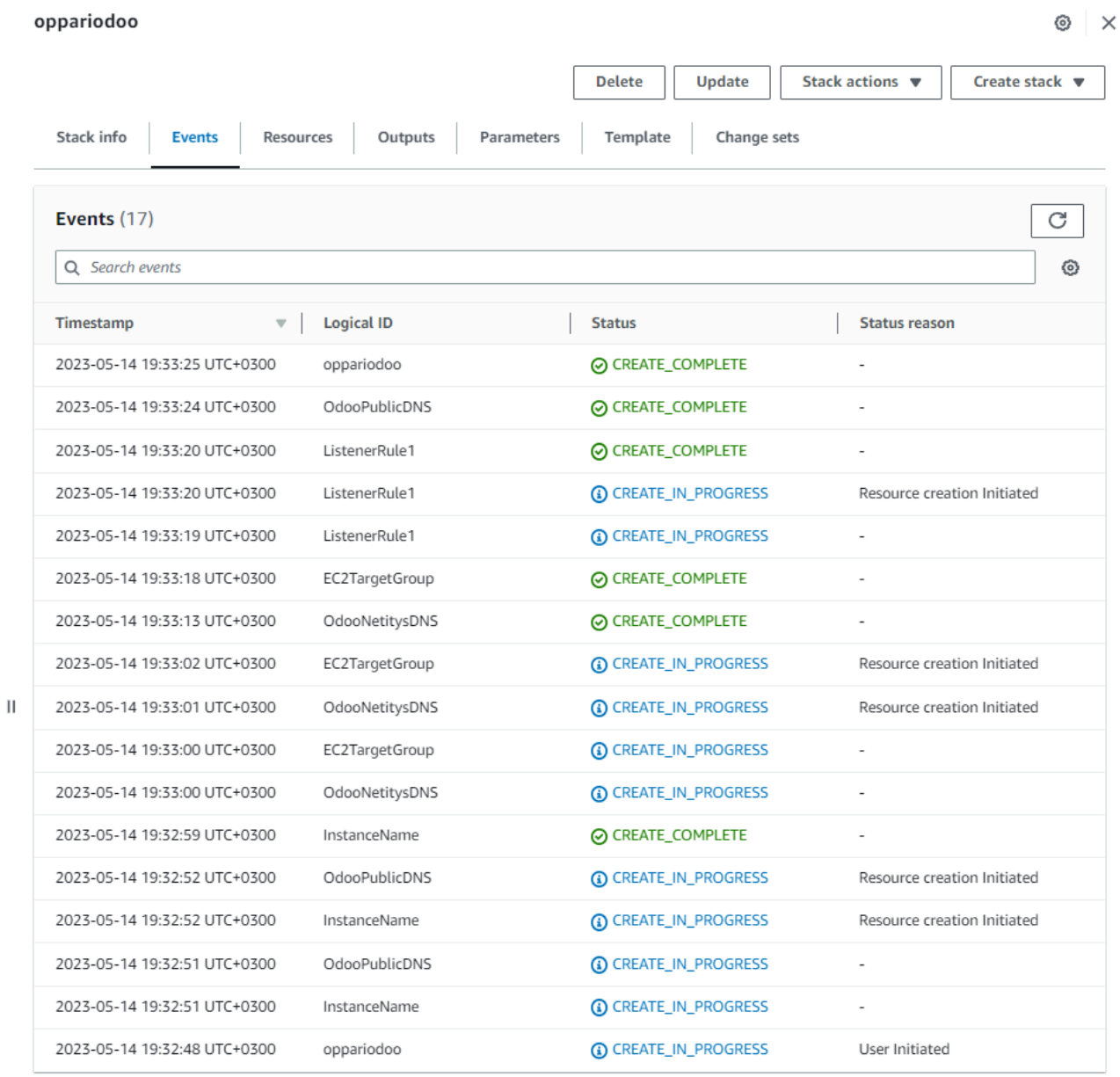
Kuva 20. Mallineen käyttö create-stack-komennolla

Käytetty komento suorittaa seuraavia toimintoja:

- aws cloudformation create-stack: Tämä on AWS CLI (Command Line Interface) -komento, joka luo uuden pinon AWS CloudFormation-palvelussa.
- --template-body file://oppari.yaml: Tämä argumentti osoittaa mallineen, jota käytän uuden pinon luomisessa. Tässä tapauksessa oletetaan, että oppari.yaml niminen tiedosto on käytettävissä samassa hakemistossa, missä komento ajetaan ja sisältää AWS CloudFormation-mallineen.
- --parameters: Tämä argumentti määrittelee parametrit, jotka lähetetään mallineeseen. Tässä tapauksessa määrittelen ScheduleValue, DNS, OdooVersion ja PriorityNumber -parametrit.
- --stack-name oppariodoo: Tämä argumentti määrittelee pinon nimen, joka luodaan AWS CloudFormation-palvelussa.

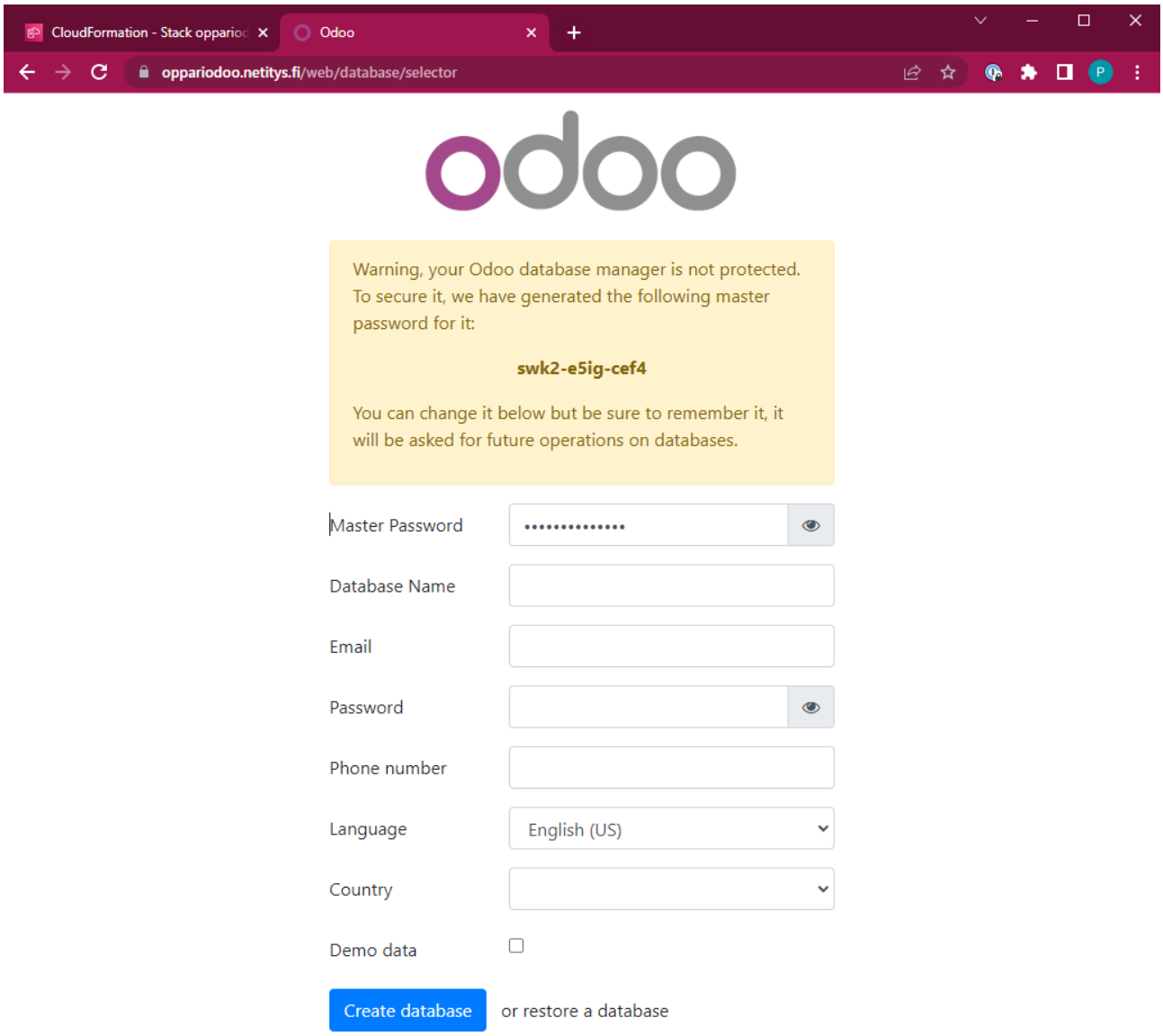
Kun tämä komento suoritetaan, AWS CloudFormation-palvelu lukee annetun mallineen, (tässä tapauksessa oppari.yaml) ja luo uuden pinon AWS-infrastruktuurin resursseilla, jotka ovat määritelty mallineessa.

Kuvakaappauksessa 10 on esitetty pinon "Events"-kohta AWS CloudFormation-palvelussa. Tämä näkymä tarjoaa tietoja pinon luontiprosessista, ja auttaa tarkastelemaan sen toimintaa. Se on erityisen hyödyllinen virheenkorjauksessa, ja pinon luonnin seurannassa.



Kuva 21. Pinon tarkastelu AWS CloudFormation-palvelussa

Kuvassa 22 on esillä toimiva Odoo-toiminnanohjausjärjestelmä. Tämä osoittaa, että malline toimii kuten suunnittelin. Olen onnistuneesti luonut toimivan Odoo-toiminnanohjausjärjestelmän käyttämällä mallinetta.



Kuva 22. Käyttövalmis Odoo-toiminnanohjausjärjestelmä

6 Arviointi ja pohdinta

Opinnäytetyöni aikana perehdyin AWS CloudFormationin käyttöön toiminnanohjausjärjestelmän luomisessa. Tämä pohdintaosio käsittelee opinnäytetyön tavoitteiden saavuttamista, keskeisiä löydöksiä ja niiden merkitystä, sekä mahdollisia jatkotutkimusaiheita.

6.1 Opinnäytetyön tavoitteiden saavuttaminen

Opinnäytetyön tavoitteet saavutettiin onnistuneesti. Sain luotua toimivan AWS CloudFormation -mallineen, joka luo ja konfiguroi Odoo-toiminnanohjausjärjestelmän ja Ubuntu-palvelimen. Tämä työ tarjosi mahdollisuuden perehtyä eri teknologioiden, kuten AWS EC2, Odoo ja YAML yhdistämiseen. Käytännön toteutuksessa pystyin hyödyntämään muuttumattoman infrastruktuurin periaatteita ja automaatioon liittyviä parhaita käytäntöjä.

6.2 Keskeiset löydökset ja niiden merkitys

Keskeiset löydökset opinnäytetyössä liittyvät AWS CloudFormationin ja YAML-mallineen mahdollisuuksiin toiminnanohjausjärjestelmän luomisessa. AWS CloudFormation ja YAML tarjosivat tehokkaan ja automatisoidun tavan luoda ja hallita resursseja.

Testaus osoitti, että malline toimii oikein, sekä laadulliset kriteerit ja tavoitteet toteutuivat, jotka määriteltiin kappaleessa 5 Mallineen toteutus. Laadullisiin kriteereihin kuului YAML-malline, joka kykenee luomaan automaattisesti toimivan Ubuntu-palvelimen Odoo-toiminnanohjausjärjestelmällä varustettuna. Tämän lisäksi mallineen kriteerinä oli myös nopeuttaa ja helpottaa Odoo-asiakasdemojen luomista ja hallintaa yrityksessämme.

Testausprosessin aikana ilmenneet ongelmat ratkaisin onnistuneesti, ja opin niistä tärkeitä oppeja, jotka voivat auttaa tulevaisuudessa samankaltaisten projektien toteuttamisessa.

6.3 Mahdolliset jatkotutkimusaiheet

Jatkotutkimusaiheita voisivat olla esimerkiksi AWS CloudFormationin soveltaminen muiden toiminnanohjausjärjestelmien käyttöönotossa, tai AWS:n muiden palveluiden hyödyntäminen Odoo-toiminnanohjausjärjestelmän tehokkuuden ja tietoturvan parantamiseksi. Lisäksi voitaisiin tutkia erilaisten palvelinarkkitehtuurien vaikutusta Odoon suorituskykyyn ja ylläpidettävyyteen.

Kaiken kaikkiaan tämä opinnäytetyö tarjosi mahdollisuuden syventää osaamistani AWS CloudFormationin ja YAML-mallineiden käytössä. Tämän prosessin aikana opin arvokkaita taitoja ja käytäntöjä, jotka auttavat minua tulevaisuudessa vastaavien projektien toteuttamisessa ja parantavat ammatillista osaamistani.

Tämän opinnäytetyön tulokset ja kokemukset antavat minulle ja muille alan ammattilaisille hyvän pohjan jatkaa AWS CloudFormationin hyödyntämistä erilaisten sovellusten ja järjestelmien käyttöönotossa ja ylläpidossa. Toivon, että tämä työ auttaa muita ymmärtämään AWS CloudFormationin ja YAML-mallineen potentiaalia, ja tarjoaa hyödyllisiä näkökulmia toiminnanohjausjärjestelmien käyttöönottoon liittyvissä projekteissa.

.

Lähteet

Amazon Web Services 2018. New T3 Instances – Burstable, Cost-Effective Performance. Luettavissa: https://aws.amazon.com/blogs/aws/new-t3-instances-burstable-cost-effective-performance/. Luettu: 15.5.2023.

Amazon Web Services s.a. a. AWS CloudFormation FAQs. Luettavissa: https://aws.amazon.com/cloudformation/faqs/. Luettu: 19.4.2023.

Amazon Web Services s.a. b. What is AWS. Luettavissa: https://aws.amazon.com/what-is-aws. Luettu: 12.5.2023.

Amazon Web Services s.a. c. How does AWS CloudFormation work? Luettavissa: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-whatis-howdoesit-work.html. Luettu: 12.5.2023.

Amazon Web Services s.a. d. AWS CloudFormation template formats. Luettavissa: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-formats.html. Luettu: 12.5.2023.

Amazon Web Services s.a. e. Infrastructure as code. Luettavissa: https://docs.aws.amazon.com/whitepapers/latest/introduction-devops-aws/infrastructure-as-code.html. Luettu 12.5.2023.

Amazon Web Services s.a. f. Amazon RDS. Luettavissa: https://aws.amazon.com/rds. Luettu 12.5.2023.

Amazon Web Services s.a. g. What is Amazon EC2? Luettavissa: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html. Luettu: 25.4.2023.

Amazon Web Services s.a. h. What is Amazon EC2 Auto Scaling? Luettavissa: https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html. Luettu: 25.4.2023.

Amazon Web Services s.a. i. What is Elastic Load Balancing? Luettavissa: https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html. Luettu: 15.5.2023.

Amazon Web Services s.a. j. What is an Application Load Balancer? Luettavissa: https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html. Luettu: 15.5.2023.

Amazon Web Services s.a. k. Format version. Luettavissa: https://docs.aws.amazon.com/AWS-CloudFormation/latest/UserGuide/format-version-structure.html. Luettu: 15.5.2023.

Amazon Web Services s.a. l. Description. Luettavissa: https://docs.aws.amazon.com/AWSCloud-Formation/latest/UserGuide/template-description-structure.html. Luettu: 15.5.2023.

Amazon Web Services s.a. m. Parameters. Luettavissa: https://docs.aws.amazon.com/AWSCloud-Formation/latest/UserGuide/parameters-section-structure.html. Luettu: 15.5.2023.

Amazon Web Services s.a. n. Mappings. Luettavissa: https://docs.aws.amazon.com/AWSCloud-Formation/latest/UserGuide/mappings-section-structure.html. Luettu: 15.5.2023.

Amazon Web Services s.a. o. Conditions. Luettavissa: https://catalog.workshops.aws/cfn101/en-US/intermediate/templates/conditions. Luettu: 15.5.2023.

Amazon Web Services s.a. p. Transform. Luettavissa: https://docs.aws.amazon.com/AWSCloud-Formation/latest/UserGuide/transform-section-structure.html. Luettu: 15.5.2023.

Amazon Web Services s.a. q. Resources. Luettavissa: https://docs.aws.amazon.com/AWSCloud-Formation/latest/UserGuide/resources-section-structure.html. Luettu: 15.5.2023.

Amazon Web Services s.a. q. Metadata. Luettavissa: https://docs.aws.amazon.com/AWSCloud-Formation/latest/UserGuide/metadata-section-structure.html. Luettu: 15.5.2023.

Amazon Web Services s.a. r. Amazon Machine Images (AMI). Luettavissa: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html. Luettu: 25.4.2023.

Amazon Web Services s.a. s. Working with stacks. Luettavissa: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacks.html. Luettu: 25.4.2023.

Amazon Web Services s.a. t. Stack failure options. Luettavissa: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stack-failure-options.html. Luettu: 25.4.2023.

Amazon Web Services s.a. u. AWS::CloudFormation::WaitCondition. Luettavissa: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-waitcondition.html. Luettu: 25.4.2023.

Captivea s.a. Odoo and scalability. Luettavissa https://www.captivea.com/odoo/odoo-scalability. Luettu: 25.4.2023.

Digital Ocean 2017. What Is Immutable Infrastructure? Luettavissa: https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure. Luettu 12.5.2023.

HashiCorp 2018. What is Mutable vs. Immutable Infrastructure? Luettavissa: https://www.hashicorp.com/resources/what-is-mutable-vs-immutable-infrastructure. Luettu: 24.4.2023.

Hostinger 2023. What Is Ubuntu? A Quick Beginner's Guide. Luettavissa: https://www.hostinger.com/tutorials/what-is-ubuntu#:~:text=Unlike%20proprietary%20software%2C%20Ubuntu%20is,a%20license%20to%20use%20Ubuntu. Luettu: 23.4.2023.

Jenna Pederson 2021. PROVISIONING AN EC2 INSTANCE WITH CLOUDFORMATION (PART 1). Luettavissa: https://jennapederson.com/blog/2021/6/21/provisioning-an-ec2-instance-with-cloudformation-part-1/. Luettu: 21.4.2023.

Json s.a. Introducing JSON. Luettavissa: https://www.json.org/json-en.html. Luettu: 19.5.2023.

Make use of 2022. Ubuntu Desktop vs. Ubuntu Server: What's the Difference? Luettavissa: https://www.makeuseof.com/tag/difference-ubuntu-desktop-ubuntu-server/. Luettu: 23.4.2023.

Odoo s.a. a. The Odoo story. Luettavissa: https://www.odoo.com/blog/odoo-news-5/the-odoo-story-56. Luettu: 21.4.2023.

Odoo s.a. b. Running Your Own Service Business with Odoo - Everything You Need to Know. Luettavissa: https://www.odoo.com/blog/business-hacks-1/running-your-own-service-business-with-odoo-everything-you-need-to-know-620. Luettu 22.4.2023.

Odoo s.a. c. Maximize your warehouse efficiency. Luettavissa: https://www.odoo.com/app/inventory. Luettu: 22.4.2023.

Odoo s.a. d. The real customer centric CRM. Luettavissa: https://www.odoo.com/app/crm. Luettu: 22.4.2023.

Odoo s.a. e. Employee features. Luettavissa: https://www.odoo.com/app/employees-features. Luettu: 22.4.2023.

Odoo s.a. f. Agile Project Management. Luettavissa: https://www.odoo.com/app/project. Luettu: 23.4.2023.

Oracle s.a. What is ERP?. Luettavissa https://www.oracle.com/erp/what-is-erp/. Luettu: 22.4.2023.

Makeuseof 2022. Why You Should Prefer Ubuntu LTS Over Normal Releases. Luettavissa: https://www.makeuseof.com/why-use-ubuntu-lts-releases/. Luettu: 20.4.2023.

Red Hat 2023a. What is YAML? Luettavissa: https://www.redhat.com/en/topics/automation/what-is-yaml. Luettu: 15.5.2023.

Red Hat 2023b. What is Infrastructure as Code (IaC)?. Luettavissa: https://www.red-hat.com/en/topics/automation/what-is-infrastructure-as-code-iac. Luettu 12.5.2023.

Solvve 2023. What Makes Odoo Different from Other ERP Solutions? Luettavissa: https://solvve.com/blog/what-makes-odoo-different-from-other-erp-solutions/. Luettu: 18.4.2023.

Sourcefuse 2023. Infrastructure-as-Code on AWS. Luettavissa: https://www.sourcefuse.com/port-folio/infrastructure-as-code-aws. Luettu 12.5.2023.

Synconics 2023. What is Odoo ERP / Open ERP Software? Luettavissa: https://www.syn-conics.com/what-is-odoo-features/. Luettu: 20.5.2023.

Techielass 2022. AWS CloudFormation template explained. Luettavissa https://www.techielass.com/aws-cloudformation-template-explained/. Luettu: 20.4.2023.

TechTarget 2022. What is immutable infrastructure? Luettavissa: https://www.tech-target.com/searchitoperations/definition/immutable-infrastructure#:~:text=Immutable%20infrastruc-ture%20is%20an%20approach,each%20time%20any%20change%20occurs. Luettu: 24.4.2023.

Ubuntu 2023. Scale out with Ubuntu Server. Luettavissa: https://ubuntu.com/server. Luettu: 24.4.2023.

Ubuntu s.a. Build your CloudFormation Templates with the latest Ubuntu AMI. Luettavissa: https://ubuntu.com/tutorials/build-your-cloudformation-templates-with-the-latest-ubuntu-ami#1-overview. Luettu: 24.4.2023./

Ventor Tech 2021. Odoo hardware requirements. Luettavissa: https://ventor.tech/odoo/odoo-hard-ware-requirements/. Luettu: 15.5.2023.

Zdnet 2022. inux: AWS now offers Ubuntu virtual desktops for developers and engineers. Luetta-vissa https://www.zdnet.com/article/linux-aws-now-offers-ubuntu-virtual-desktops-for-developers-and-engineers/. Luettu: 15.5.2023.

Liitteet

AWS CloudFormation YAML -malline toiminnanohjausjärjestelmän luomiseen

odoo.yaml

```
AWSTemplateFormatVersion: "2010-09-09"
Description: 'AWS Odoo'
Parameters:
  KeyName:
    Description: Name of an existing EC2 KeyPair to enable SSH access to the
bastion host
    Type: AWS::EC2::KeyPair::KeyName
    Default: efaws
    ConstraintDescription: must be the name of an existing EC2 KeyPair.
  Subnet:
    Description: Subnet to place the EC" in
    Type: String
    Default: subnet-0022543764bb97c5a02e
  ALBListener:
    Type: String
    Default: arn:aws:elasticloadbalancing:eu-west-1:403535325052153:lis-
tener/app/NetitysALB/343dfhb68ed/7c478124a68ed80b
  AllowNetitysJumper: #Aseta Security Group
    Type: String
    Default: sg-dfhdfhd7321c55
  AllowOdoo: #Aseta Security Group
    Type: String
    Default: sg-1fdgn49w5rfdsgd
  AllowNetitys: #Aseta Security Group
    Type: String
    Default: sg-35890384fcgcbbd
  HostedZone: #Määrittelee mihin hosted zoneen instanssi asetetaan
    Type: String
    Default: FFDHD23342GGDFNJ
  VPC:
    Type: String
    Default: vpc-e5r6gh343ghnf
  DNS:
    Type: String #Määrittelee palvelimen nimen rivi 70. Nimi asetetaan
cloudformation create-stack komennon parametrissä.
  OdooVersion:
    Type: String #Määrittelee Odoon version esim. 13, 14. Numero asetetaan
cloudformation create-stack komennon parametrissä.
  PriorityNumber:
    Type: String
  ScheduleValue:
    Type: String
Resources:
  EC2TargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      HealthCheckIntervalSeconds: 30
      HealthCheckProtocol: HTTP
```

```
      HealthCheckTimeoutSeconds: 15
      HealthyThresholdCount: 5
      Matcher:
        HttpCode: '200'
      Port: 8069
      Protocol: HTTP
      TargetGroupAttributes:
        - Key: deregistration_delay.timeout_seconds
          Value: '20'
      Targets:
        - Id: !Ref InstanceName
          Port: 8069
      UnhealthyThresholdCount: 3
      VpcId: vpc-3456fbdkjipj345
      Tags:
        - Key: Name
          Value: EC2TargetGroup
        - Key: Port
          Value: 8069
  ListenerRule1:
    Type: 'AWS::ElasticLoadBalancingV2::ListenerRule'
    Properties:
      Actions:
        - Type: forward
          TargetGroupArn: !Ref EC2TargetGroup
      Conditions:
        - Field: host-header
          HostHeaderConfig:
            Values:
              - !Sub ${DNS}.netitys.fi
      ListenerArn: !Ref ALBListener
      Priority: !Ref PriorityNumber
  InstanceName: #Määrittelee instanssin nimen
    Type: AWS::EC2::Instance
    Metadata:
      AWS::CloudFormation::Init:
        configSets:
          OnCreation: #Määrittelee missä järjestyksessä eri vaiheet asen-
nuksesta tapahtuvat.
            - hup
            - preinstall
          OnUpdate:
            - configs
            - aptupdate
            - install
        hup:
          files:
            /etc/cfn/cfn-hup.conf:
              content: !Sub |
                [main]
                stack=${AWS::StackId}
                region=${AWS::Region}
                interval=1
              mode: '000400'
              owner: root
              group: root
            /etc/cfn/hooks.d/cfn-auto-reloader.conf:
```

```
              content: !Sub |
                [cfn-auto-reloader-hook]
                triggers=post.update
                path=Resources.InstanceName.Metadata.AWS::CloudFor-
mation::Init
                action=/opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --
resource InstanceName --configsets OnUpdate --region ${AWS::Region}
                runas=root
              mode: "000400"
              owner: "root"
              group: "root"
          services:
            sysvinit:
              cfn-hup:
                enabled: true
                ensureRunning: true
                files:
                - '/etc/cfn/cfn-hup.conf'
                - '/etc/cfn/hooks.d/cfn-auto-reloader.conf'
        preinstall:
          commands:
            01_get_key:
              command: wget -O - https://nightly.odoo.com/odoo.key | apt-key
add -
            02_get_maventa_module:
              command: wget https://netitys-packages.s3.eu-west-1.amazo-
naws.com/moduulit.15.03.tar.gz
            03_unpack_module2:
              command: tar -xzf moduulit.15.03.tar.gz -C /
            04_wkhtmltopdf:
              command: wget https://github.com/wkhtmltopdf/packaging/re-
leases/download/0.12.6-1/wkhtmltox_0.12.6-1.focal_amd64.deb
            05_wkhtmltopdf_install:
              command: apt install -y ./wkhtmltox_0.12.6-1.focal_amd64.deb
            06_install_html2text:
              command: apt install -y html2text
          packages:
            apt:
              postgresql: []
              python3-boto3: []
              python3-requests: []
              html2text: []
        configs:
          files:
            /etc/apt/sources.list.d/odoo.list:
              content: !Sub |
                deb http://nightly.odoo.com/${OdooVersion}.0/nightly/deb/ ./
        aptupdate:
          commands:
            01_apt_update:
              command: apt update
        install:
          packages:
            apt:
              odoo: []
          commands:
            01_set_hostname:
```

```
            command: !Sub hostname ${DNS}.netitys
          02_update_hostname_file:
            command: !Sub echo ${DNS}.netitys > /etc/hostname
          03_enable_postgres:
            command: systemctl enable postgresql
          04_start_postgres:
            command: systemctl start postgresql
          05_enable_odoo:
            command: systemctl enable odoo
          06_start_odoo:
            command: systemctl start odoo
    Properties:
      SecurityGroupIds:
        - !Ref AllowNetitysJumper
        - !Ref AllowOdoo
        - !Ref AllowNetitys
      SubnetId: !Ref Subnet
      InstanceType: t3.small
      KeyName: !Ref KeyName
      Tags:
        - Key: "Schedule"
          Value: !Ref ScheduleValue
      ImageId: ami-023fdgwersdgm455
      BlockDeviceMappings:
        - DeviceName: "/dev/xvda"
          Ebs:
            VolumeSize: '10'
            Encrypted: 'true'
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash -xe
          apt-get -qqy update
          apt-get -qqy dist-upgrade
          apt-get -qqy autoremove
          apt-get -qqy autoclean
          #          apt-get -qqy install curl wget
          apt-get -qqy install python3-pip python3-setuptools python3-
testresources
          pip3 install https://netitys-packages.s3.eu-west-1.amazo-
naws.com/aws-cfn-bootstrap-py3-latest.tar.gz
          #          sleep 5
          mkdir -p /opt/aws/bin
          ln -s /usr/local/bin/cfn-hup /opt/aws/bin/cfn-hup
          ln -s /usr/local/bin/cfn-init /opt/aws/bin/cfn-init
          ln -s /usr/local/bin/cfn-signal /opt/aws/bin/cfn-signal
          ln -s /usr/local/bin/cfn-elect-cmd-leader /opt/aws/bin/cfn-elect-
cmd-leader
          ln -s /usr/local/bin/cfn-get-metadata /opt/aws/bin/cfn-get-
metadata
          ln -s /usr/local/bin/cfn-send-cmd-event /opt/aws/bin/cfn-send-cmd-
event
          ln -s /usr/local/bin/cfn-send-cmd-result /opt/aws/bin/cfn-send-
cmd-result
          #sleep 5
          ln -s /usr/local/init/ubuntu/cfn-hup /etc/init.d/cfn-hup
          chmod +x /usr/local/init/ubuntu/cfn-hup
          update-rc.d cfn-hup defaults
```

```
          echo set bell-style none >> /etc/inputrc
          echo set vb >> /etc/vim/vimrc
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource In-
stanceName --configsets OnCreation --region ${AWS::Region}
          /opt/aws/bin/cfn-init -v --stack ${AWS::StackName} --resource In-
stanceName --configsets OnUpdate --region ${AWS::Region}
          /opt/aws/bin/cfn-signal -e $? --stack ${AWS::StackName} --resource
InstanceName --region ${AWS::Region}
  OdooPublicDNS:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      HostedZoneId: !Ref HostedZone
      RecordSets:
      - Name: !Sub ${DNS}.netitys.fi.
        Type: A
        AliasTarget:
          DNSName: NetitysALB-4305890325.eu-west-1.elb.amazonaws.com
          HostedZoneId: GHDFGH38923DF2
  OdooNetitysDNS:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      HostedZoneId: DFGGD45282HG
      RecordSets:
      - Name: !Sub ${DNS}.netitys.
        Type: A
        TTL: 300
        ResourceRecords:
          - !GetAtt InstanceName.PrivateIp
```