jamk

Elli-sovelluksen AWS-pilvimigraatio

Marianne Äikäs

Opinnäytetyö, AMK
Toukokuu 2025
Tietojenkäsittelyn tutkinto-ohjelma

jamk | Jyväskylän ammattikorkeakoulu
University of Applied Sciences

jamk

Kuvailulehti

Äikäs, Marianne

Elli-sovelluksen AWS-pilvimigraatio

Jyväskylä: Jyväskylän ammattikorkeakoulu. **Toukokuu 2025**, 53 sivua.

Tietojenkäsittelyn tutkinto-ohjelma. Opinnäytetyö AMK.

Julkaisun kieli: suomi

Julkaisulupa avoimessa verkossa: kyllä

Tiivistelmä

Nykypäivänä oppilaitosten ja yritysten yhteistyömahdollisuudet ovat tärkeä osa työelämän ensiaskeleita ja usein avaintekijöitä henkilökohtaisen ammattitaidon aidossa kehittämisessä. Elli-sovellus on Jyväskylän ammattikorkeakoulussa toteutetun Ticorporate-opintojakson lopullinen sovellustuotos, jonka tarkoitus on hallita osapuolien yhteistyötoimintaa. Elli-sovelluksen toimeksianto tuli ulkoisesti Elisa Oyj:ltä, joka on Jyväskylän ammattikorkeakoulun yhteistyökumppani. Opinnäytetyön ensisijainen kehittämistavoite oli Elli-sovelluksen onnistunut migraatio Jyväskylän ammattikorkeakoulun pilviympäristöön siten, että sovellus on valmis käyttöönottoon Jyväskylän ammattikorkeakoulun sekä Elisa Oyj:n henkilökunnalle. Toissijaisena tavoitteena oli lisäksi Elli-sovelluksen DevOps-toimintamalliin liittyvien kehitystoimenpiteiden prosessien kehittäminen, sekä ohjeoppaan tuottaminen, jonka tarkoitus on toimia informatiivisena tiedon lähteenä esimerkiksi avustamaan Jyväskylän ammattikorkeakoulua Elli-sovelluksen ja sen infrastruktuurin ylläpidossa, hallinnassa ja monistamisessa.

Toteutus on sisäinen kehitystyö Jyväskylän ammattikorkeakoululle ja sen ollessa sovelluskehitystoimintaa, sen toteutustyypiksi valikoitui tutkimuksellinen kehittämistyö.

Kehittämistehtävän tuloksena Elli-sovelluksen migraatio suoritettiin onnistuneesti ja Elli-sovellus on käytettävissä Jyväskylän ammattikorkeakoulun AWS-pilviympäristössä. Migraatioprosessin toteutuksen yhteydessä Elli-sovelluksen DevOps-toimintamalliin liittyviä kehitystoimenpiteiden prosesseja edistettiin automatisoidummiksi. Sovelluksen migraation lisäksi toteutettiin ohjeopas, jossa käsitellään pystytettävän AWS-ympäristön ominaisuuksia ja kuinka Elastic Beanstalk-sovelluksen julkaisu automatisoidaan Github Actions-alustan avulla.

Elli-sovelluksessa ja siihen liittyvissä osissa on kiistämätön potentiaali suurelle määrälle erilaisia mahdollisuuksia teknologian ja liiketalouden aloilla, mutta myös oppimisympäristöissä.

Avainsanat (asiasanat)

AWS, Amazon Web Services, pilvipalvelut, käyttöönotto, DevOps, CI/CD

Muut tiedot (salassa pidettävät liitteet)

Opinnäytetyö ei sisällä salassa pidettäviä liitteitä. Liitteenä ohjeopas, 23 sivua.

jamk

Description

Äikäs, Marianne

AWS Cloud Migration of the Elli Application

Jyväskylä: JAMK University of Applied Sciences, May 2025, 53 pages.

Degree Programme in Business information technology. Bachelor´s thesis.

Permission for open access publication: Yes

Language of publication: Finnish

Abstract

In today's world, collaboration between educational institutions and businesses plays a crucial role in the first steps of working life and is often a key factor in the genuine development of professional competence. The Elli application is the result of the Ticorporate course at JAMK University of Applied Sciences and it is designed to manage collaborative activities between the two parties. The development of the Elli application was commissioned externally by Elisa Corporation, a partner of JAMK.

Primary development objective of this thesis was successful migration of the Elli application to JAMK's cloud environment, ensuring the application is ready to use for the staff of JAMK and Elisa Corporation. A secondary goal was to enhance the DevOps processes related to the Elli application and to produce a user guide intended to serve as an informative resource for JAMK to handle maintenance, administration and replication of the Elli application and its infrastructure.

This thesis was an internal development effort for JAMK. Given its focus on software development, the chosen implementation method was research-based development.

As a result of the development project, the migration of the Elli application was successfully completed, and the application is now available within JAMK's AWS cloud environment. During the migration process, DevOps processes related to the Elli application were improved. A user guide was produced that covers the characteristics of the AWS cloud environment and automation of the Elastic Beanstalk application deployment using Github Actions platform.

Keywords/tags (subjects)

AWS, Amazon Web Services, cloud services, cloud computing, deployment, DevOps, CI/CD

Miscellaneous (Confidential information)

No confidential information. Includes appendix of guide, 23 pages.

Sisältö

Kuviot

1 Johdanto

Nykypäivänä oppilaitosten ja yritysten yhteistyömahdollisuudet ovat tärkeä osa työelämän ensiaskeleita ja usein avaintekijöitä henkilökohtaisen ammattitaidon aidossa kehittämisessä. Yhteistyöprosessien vaiheiden tukeminen ja kehittäminen on tärkeää tulevaisuuden osaajien ammattitaidon kannalta. Käyttöönoton lisäksi Elli-sovelluksella on suuri potentiaali jatkokehitykselle perinteisen sovelluskehityksen sekä liiketoiminnan osa-alueilla. Sovelluksen jatkuvuus mahdollistaa uusia oppimismahdollisuuksia Jyväskylän ammattikorkeakoulussa, sekä edistää työelämälähtöisyyttä kehittämällä Jyväskylän ammattikorkeakoulun opintojen osia. Elli-sovelluksen ajoympäristö on suunniteltu pilviteknologioiden ympärille, jolloin sovelluksen toiminta ja sen kehitys käsittelee kestävän kehityksen ympäristövaikutuksia. Kokonaisuutena nämä mainitut ajankohtaisuuden, ammatillisen edistyksen ja oppimisen aihealueet liittyvät vahvasti eettisyyteen ja ottavat kantaa kestävän kehityksen eri näkökulmiin.

Elli-sovellus on Jyväskylän ammattikorkeakoulussa toteutetun Ticorporate-opintojakson lopullinen tuotos. Elli-sovelluksen toimeksianto tuli ulkoisesti Elisa Oyj:ltä, joka on Jyväskylän ammattikorkeakoulun yhteistyökumppani. Toimeksiantona oli kehittää yhteistyösovellus Jyväskylän ammattikorkeakoulun ja Elisan välille. Sovelluksen tarkoituksena on edistää ja parantaa yhteistöiden hallintaa ja läpinäkyvyyttä. Elli-sovelluksen kehitys Ticorporate-opintojaksolla ei sisältänyt Jyväskylän ammattikorkeakoulun pilviympäristöön migraatiota, vaikka tarve sovelluksen tuomille hyödyille on todellinen ja ajankohtainen. Vastaavanlaista sovellusta ei ole aikaisemmin toteutettu Jyväskylän ammattikorkeakoulussa ja yhteistöiden hallinta helpottuisi ja selkeytyisi sovelluksen julkaisun ja käyttöönoton myötä.

Opinnäytetyö oli sisäinen kehitystyö Jyväskylän ammattikorkeakoululle ja aiheena oli Elli-sovelluksen migraatio Jyväskylän ammattikorkeakoulun pilviympäristöön. Ensisijainen kehittämistavoite oli Elli-sovelluksen onnistunut migraatio Jyväskylän ammattikorkeakoulun pilviympäristöön siten, että sovellus on valmis käyttöönottoon Jyväskylän ammattikorkeakoulun sekä Elisa Oyj:n henkilökunnalle. Tämä kehittämistavoite keskittyi vahvasti pilviteknologioiden aihealueeseen, jonka vuoksi sovelluskehityksen aiheen rajauksen ulkopuolelle jäi esimerkiksi käyttöliittymäpuolen kehitys (engl. *Frontend*) ja palvelinpuolen kehitys (engl. *Backend*), vaikka niidenkin osalta laajempi jatkokehitys olisi tarpeellista.

Kehittämistavoitteeseen kuului toissijaisena tavoitteena myös Elli-sovelluksen DevOps-toimintamalliin liittyvien kehitystoimenpiteiden prosessien kehittäminen, sekä tuottaa ohjeopas, jonka tarkoitus on toimia informatiivisena tiedon lähteenä esimerkiksi avustamaan Jyväskylän ammattikorkeakoulua Elli-sovelluksen ja sen infrastruktuurin ylläpidossa, hallinnassa ja monistamisessa. Tämä kyseinen ohjeopas on myös suunnattu tuleville Ticorporate-opintojakson toteutuksille, jotta monipuolisia oppimismahdollisuuksia voidaan edesauttaa ja opintojen osia kehittää eteenpäin.

2 Tutkimusasetelma

2.1 Toimeksiantaja

Opinnäytetyö on sisäinen kehitystyö Jyväskylän ammattikorkeakoululle. Jyväskylän ammattikorkeakoulun tavoitteet ovat keskeisesti kumppaneiden ja yhteistyöverkostojen kanssa sidoksissa, kehittäen ja vahvistaen oppimista, osaamista ja kilpailukykyä. Työelämän vaatimuksiin perustuva korkeakouluopetus on yksi Jyväskylän ammattikorkeakoulun perustehtävistä. Valmistuvia opiskelijoita on vuosittain yli 1500 ja heitä on 8 eri ammattialalta, sekä työllistymisprosentti vuosi valmistumisen jälkeen on 80 %. Eri elämäntilanteisiin sopivia vaihtoehtoja on tarjolla yli 40 erilaista tutkintoa 7 eri alalla. (Jamk n.d.)

Ticorporate on Jyväskylän ammattikorkeakoulun tietojenkäsittelyn tutkinto-ohjelmaan kuuluva laaja opintokokonaisuus, joka koostuu kahdesta osasta: Ticorporate Demo Lab 1 ja Ticorporate Demo Lab 2. Ensimmäinen osa keskittyy vahvasti siihen, kuinka uudet konseptit ja tuoteideat luodaan ja sulavasti jatkuu toiseen osaan, jossa työskennellään osana tiimiä noudattaen ketterän kehityksen viitekehystä. Kurssin lopputuloksena tähdätään valmiiseen tuotedemototeutukseen pohjautuen luotuihin konsepteihin, jotka voivat olla joko pelejä tai ohjelmistoja. (FF Ticorporate Demo… n.d.)

Tiimien roolit muodostuvat erilaisista projektinhallintarooleista ja näiden lisäksi opiskelijoille jaetaan myös ohjelmistokehittäjäroolit. (FF Ticorporate Demo… n.d.) Ticorporate-kurssilla yleisesti kehitetään tuotteita opiskelijoiden omien konseptien pohjalta, mutta mahdollisuus on myös osallistua ulkoiseen toimeksiantoon. Elisa Oyj:ltä saatu toimeksianto kuvasi tarvetta sovellukselle, jolla pystyttäisiin hallitsemaan Jyväskylän ammattikorkeakoulun ja Elisa Oyj:n välistä yhteistyötoimintaa.

2.2 Kehitystyön tavoitteet ja rajaus

Tavoitteet

Kehittämistehtävän ensisijainen tavoite on onnistunut sovelluksen migraatio Jyväskylän ammattikorkeakoulun pilviympäristöön. Sovelluksen kehityksessä on käytetty Amazon Web Services-pilviympäristöä ja siihen kuuluvia resursseja. Jyväskylän ammattikorkeakoulun ympäristö on myös Amazon Web Services-pilviympäristössä, mutta onnistunutta migraatiota varten tarvitsee tehdä valmisteluja ja muutoksia. Kehittämistyön lopputuloksena syntyy käyttövalmis tuotos, jota ajetaan Jyväskylän ammattikorkeakoulun pilviympäristössä. Työ selvittää ja sisältää migraatiota varten tarvittavat toimenpiteet ja muutokset, jotka toteutetaan. Kehittämistehtävään kuuluu lisäksi myös DevOps-toimintamallin kehitystoimenpiteisiin liittyvät kehitettävät osat sekä ohjeopas, jonka tarkoitus on toimia informatiivisena tiedon lähteenä esimerkiksi avustamaan Jyväskylän ammattikorkeakoulua Elli-sovelluksen ja sen infrastruktuurin ylläpidossa, hallinnassa ja monistamisessa. Tämä kyseinen ohjeopas on myös suunnattu tuleville Ticorporate-opintojakson toteutuksille, jotta monipuolisia oppimismahdollisuuksia voidaan edesauttaa ja opintojen osia kehittää eteenpäin. Kehittämistehtävän tavoitteet ovat ajankohtaisia ja ottavat kantaa kestävän kehityksen aatteisiin monilta eri näkökulmilta: Ticorporate-opintojaksolla on tulevaisuudessakin toteutuksia ja siellä harjoitetaan ketterän kehityksen viitekehystä, kehittämistyön tavoitteiden täyttyminen tukee kestävän kehityksen osia esimerkiksi tasa-arvoisen oppimisen osalta, sekä pilviympäristöjen hyödyntäminen noudattaen parhaita käytänteitä edesauttaa luonnon varojen kuluttamisen minimoimista.

Elli-sovelluksen toimeksianto ja kehitystarve oli annettu selkeästi tarpeeseen perustuen. Jyväskylän ammattikorkeakoulu ja Elisa Oyj ovat yhteistyökumppaneita ja yhteistyömahdollisuuksia on jo toimeenpantu ja toteutettu, mutta niiden hallinta on ollut hankalaa ja aikaa vievää kummankin osapuolien kohdalla. Selkeää yhteistä hallitsevaa yhteyskanavaa ei olla määritetty, vaan yhteistöiden hallinta, seuranta ja tiedotus on ollut tiettyjen henkilöiden vastuulla. Kehittämistyön tavoite on suorittaa sovelluksen migraatio onnistuneesti Jyväskylän ammattikorkeakoulun pilviympäristöön, jotta sovelluksen käyttöönotto voidaan aloittaa. Sovelluksen käyttöönoton hyödyt vastaavat suoraan kehittämistehtävän tarpeeseen, johon liittyen yhteistöiden hallinta oli todettu hankalaksi. Sovellus edistää yhteistöiden hallintaa, seurantaa ja tiedotusta parantaen niiden tehokkuutta ja läpinäkyvyyttä. Yhteistöiden hallinnan, seurannan ja tiedotuksen parantaminen on suoraan yhtey-

dessä siihen, että yhteistöitä voidaan järjestää ja niiden kautta edistää oppilaiden ammatillista pätevyyttä, sekä valmiuksia työelämään. Kehittämistehtävän tuotoksena syntyvää ohjeopasta voidaan hyödyntää esimerkiksi Jyväskylän ammattikorkeakoulun Ticorporate-opintojakson kehittämiseen ja osana oppimisen edistämistä.

Aiheen rajaus

Aiheen rajaus on aloitettu siitä näkökulmasta, että sovellukselle on todettu aito tarve. Jo sovelluksen varhaisimmissa kehitysvaiheissa käyttäjätestauksien kautta opittiin, että huolimatta sovelluksen ulkoisista tai toiminnallisista puutteista, sen käytöllä olisi ollut jo suuri potentiaali tuoda arvoa yhteistöiden osalta. Tämä näkökulma on ratkaiseva tekijä siinä, että Elli-sovelluksen kehitys valikoitui kehittämistyön aiheeksi kokonaisuudessaan. Myös henkilökohtainen kiinnostus yleisesti pilvipalveluita, infrastruktuuria ja julkaisuputkia kohtaan on vaikuttanut aiherajaukseen. Henkilökohtainen kiinnostus varmistaa sen, että projektin parissa työskentely säilyy mielenkiintoisena, joka edesauttaa kokonaisvaltaisesti edistymistä ja laatua. Ticorporate-opintojaksolla kehitystiimiin kuului useita jäseniä, joilla jokaisella oli omat vastuualueensa. Jo kehitysvaiheessa oma roolini oli pilviasiantuntija ja olin vastuussa pilvi-infrastruktuurin rakentamisesta ja julkaisun osista. Tämä luonnollisesti muodostaa aiherajauksen kohdennuksen kehitysroolini mukaisiin tehtäviin ja rajaa pois esimerkiksi käyttöliittymäpuolen kehityksen (engl. *Frontend*) ja palvelinpuolen kehityksen (engl. *Backend*), vaikka niidenkin osalta laajempi jatkokehitys olisi tarpeellista.

Kehitystyön tavoitteiden saavuttaminen edistää Elli-sovelluksen yleistilaa ja jatkokehitystoimenpiteiden tarve voi olla hyödyllinen tuleville opiskelijoille tai muille kehittäjille. Tilaisuus oppia sovelluskehitystä, teknologioita ja projektikäytänteitä Elli-sovelluksen parissa mahdollistaa tilaisuuksia ja työelämäedellytyksiä opiskelijoille tai muille kehittäjille. Tavoitteiden täyttymisen seurauksena syntyvä ohjeopas on arvokasta informaatiota Elli-sovelluksen jatkokehityksen kannalta, sekä se tuo lisäarvoa esimerkiksi Ticorporate-opintojaksolle tulevaisuudessa. Nykypäivänä oppilaitosten ja yritysten yhteistyömahdollisuudet ovat tärkeä osa työelämän ensiaskeleita ja usein avaintekijöitä ammattitaidon aidossa kehittämisessä.

Migraatio pilviympäristöissä on ajankohtainen aihealue, sillä yhä useammat yritykset suunnittelevat sivustojen tai jopa ympäristöjen siirtoja joko perinteisistä konesaleista pilviympäristöihin tai toisesta pilviympäristöstä toiseen. Migraatioprosessiin kuuluu olennaisesti se, että monistettavuus

on mahdollisimman helppoa ja sujuvaa, joten kyseisen aiheen käsitteleminen tuo hyötyjä tiedon ja läpinäkyvyyden osalta tietotekniikan alalle. Liiketoimintanäkökulmasta ajateltuna Elli-sovelluksen edistäminen itsenäiseksi tuotteeksi voisi lisätä työpaikkoja Suomeen, edistää talouskasvua ja lisätä innovaatioajattelua. Tuotteistaminen edellyttäisi yrityksen perustamista tai tuoteidean myymistä, joka mahdollistaa työpaikkojen lisääntymisen. Kestävän kehityksen kannalta pilviympäristöön siirtyminen ja IaC-menetelmät edistävät ekologista tietotekniikan käyttöä, sillä mallipohjat nopeuttavat resurssien käyttöönottoa, jolloin turhaa kulutusta pystytään välttämään esimerkiksi sähkö- ja vesiresurssien osalta.

2.3 Tutkimusmenetelmä ja tiedonhaku

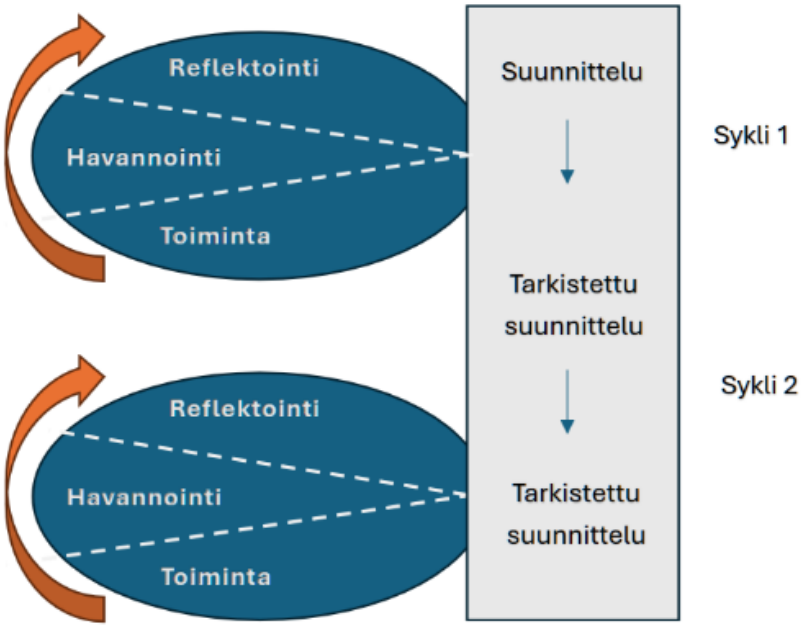
Tutkimusmenetelmän valinta websovelluskehityksessä

Opinnäytetyön tyyppi on tutkimuksellinen kehittämistyö, sillä tavoitteena on ratkaista käytännön ongelma, joka on sovelluksen migraatio. Kehitystyö on sovelluskehitystä ja sen lopputulos on tuote, sekä tutkimusote vahvasti painottuu kehityksen osiin. Bister (2019, 42–44) kertoo kirjassaan kriteereitä tutkimukselliseen kehittämistoimintaan liittyen ja selventää, että jos toimintatutkimuksella saavutetaan korjauksia, parannuksia tai tehostumista sekä tavoitteet ovat tietyn ongelman ratkaisuun johtavia, kyse on kehittämistoiminnasta.

Bister (2019) kuvaa tietojenkäsittelyn opinnäytetyötä käsittelevässä kirjassaan erilaisia kehittämistyön menetelmiä ja toimenpiteitä, joita ovat muun muassa: toimintojen muuttaminen, poistaminen tai niiden korvaaminen uusilla. Esimerkiksi lopputulokseen kuuluu osana dokumentaation kehittäminen, jonka Bister ehdottaa yhtenä osana kehittämistyötä kirjassaan. (Bister 2019, 31.) Koska opinnäytetyön aihe ja tavoite on suorittaa web-sovelluksen migraatio, siihen kuuluu olennaisesti kaikki nämä kehittämisen menetelmät ja toimenpiteet.

Bisterin (2019) mukaan ennen varsinaista kehitystyötä suoritetaan kartoittava tiedon keruuprosessi, jossa selvitetään kohteen olemus ja luonne. Tutkijan, eli tässä tapauksessa kehitystyöhön osallistuvalla on vastuu perehtyä toimintaympäristöön ja ilmiöön erityisen tarkasti, jotta ymmärrys kohteesta ja siihen liittyvistä mahdollisuuksista on syvällinen. Kokonaisuuden relevanttien osien valitseminen on kehitystyön tekijän harkittavissa. (Bister 2019, 44.)

Kehittämistyö usein on syklistä toimintaa, eli eteneminen tapahtuu useiden kehittämiskierroksien kautta. Tätä toimintatapaa voidaan kutsua termillä iteraatio, sillä se on lähestymistä kierros kierrokselta. Yleissuunnitelma on joustava, sillä prosessimallissa korostetaan jatkuvaa reflektointia. (Bister 2019, 44.) Kehittämistyössä tavoitteisiin aiotaan päästä noudattamalla syklistä prosessimallia, jonka aikana etenemistä vahvistaa toiminnan havainnointi ja reflektointi. Prosessi voi sisältää useitakin syklejä ja muovautua sen mukaan, kuinka työ etenee. Koko prosessia kuitenkin ohjaa määritellyn kehittämistehtävän täyttäminen.



Kuvio 1. Toimintatutkimuksen syklinen prosessi (Bister 2019, 45, muokattu; Alun perin lähteessä Linturi 2003)

Tiedonhaun kuvaus

Opinnäytetyön tietoperustan kirjoittamiseen ja kokoamiseen on kiinnitetty erityistä huomiota siten, että käytettyjä lähteitä ja niiden alkuperää on alusta asti arvioitu ja analysoitu kriittisesti. Tiedon aihealue on otettu huomioon samalla, kun sen julkaisuajankohtaa on tarkisteltu. Tiedon ajantasaisuus on otettu huomioon lähteitä valittaessa ja käytettäessä vanhempaa kirjallisuutta, on tiedoista poimittu sellaisia seikkoja, jotka eivät ole ajan kuluessa muuttuneet. Erityisen tarkkoja

lähteiden valinnassa on oltu sen osalta, onko sillä kaupallista lähtöasetelmaa tai onko sillä puolueellista kantaa. Tietoperustan tieto perustuu erilaisten asiantuntijoiden kirjoituksiin, joilla on selkeästi tietoa, taitoa ja kokemusta aihealueeseen liittyen. Suurin lähdeaineisto on peräisin AWS-dokumentaatioista, joiden kirjoittajat ovat AWS:n ammattilaisia, sekä nämä dokumentaatiot ovat kaikista luotettavin lähde koskien kyseisiä palveluita. AWS-dokumentaatioita päivitetään ahkerasti ja niiden muutoslokit ovat nähtävillä jokaisessa dokumentissa. Kokonaisuudessaan lähteet ovat kategorioiltaan pääasiassa olleet: tutkimuskirjallisuus (e-julkaisut), opas (käyttöopas, e-opas) ja yleinen tietokirjallisuus (e-kirjat, tietokirjat). Lähteiden haku on ensisijaisesti toteutettu tietokannoista ja hakupalveluista, kuten Janet Finna ja Finna-palvelut, mutta Googlen hakukonetta on myös hyödynnetty. Oma opittu ammattitaito on suuressa roolissa lähteiden valinnassa, sillä laaja ammattitaito, -tieto ja kokemus auttavat ymmärtämään tiedon oikeellisuuden ja ajankohtaisuuden. Aineistot oli valittu niillä perusteilla, että niiden sisällöt tukevat tavoitteiden saavuttamista ja tarvittavat pohjatiedot onnistumisen edellytyksille oli kerättävissä niiden sisällöistä.

3 Pilvityökalut websovelluskehityksessä

3.1 Pilvipalveluiden yleiskatsaus

Pilviympäristöt voidaan ajatella jakautuvan kahteen osaan, jotka ovat julkinen pilviympäristö ja yksityinen pilviympäristö. Yleinen käyttötapa yrityksillä on hybridipilviympäristö, jossa yksityiset resurssit kuten datakeskukset (engl. *Data center*) ja yksityinen pilvi (engl. *Private cloud*) ovat samaan aikaan käytössä julkisten pilviresurssien (engl. *Public cloud*) kanssa. On olemassa myös monipilviympäristö (engl. *Hybrid cloud environment*), jossa organisaatiot hyödyntävät samanaikaisesti useiden eri julkisten pilvipalveluiden tarjontaa. (Hurwitz & Kirsch 2020, 9.) Pilviympäristöjen eri muodot tarjoavat mahdollisuuden valita eri tasoja ohjauksen, joustavuuden ja hallinnan osalta (Overview of Amazon Web Services 2024, 4).

Pilvipalveluiden keskeinen toimintaperiaate, jossa resursseja toimitetaan verkon yli perustuu tarpeen mukaan saatavilla olevaan malliin, jota kutsutaan termillä ”*on-demand*” (Overview of Amazon Web Services 2024, 2). Tämä tarkoittaa sitä, että palveluita voidaan ottaa käyttöön omien tarpeiden ja vaatimuksien mukaisesti, eikä turhia resurssikuluja synny. ”*Pay-as-you-go*”-hinnoittelumallilla tarkoitetaan sitä, että vain käytettyjen resurssien ja palveluiden kulut kuuluvat

laskutettuun hintaan. Kyseinen hinnoittelumalli on keskeinen osa pilvipalveluiden toimintaa ja liittyy vahvasti siihen, kuinka pilvipalvelut ovat joustavia ja skaalautuvia. Joustavuus ja skaalautuvuus näkyy käytön hinnoissa, mutta ne liittyvät myös siihen, kuinka käytettävät resurssit muuntautuvat käyttötarkoituksien mukaan esimerkiksi tehojen tai koon osalta. Pilvipalveluiden tarjontaan kuuluu muun muassa: laskentateho (engl. *Computing*), tietokannat, tallennusratkaisut, sovellukset ja muut IT-resurssit. Nopeus on myös avainasemassa pilvipalveluiden hyödyntämisessä, sillä niiden käyttöönotto ei vaadi etukäteen laitteistojen fyysistä hankkimista ja hallintaa, vaan käyttöönotto voidaan tehdä melkein välittömästi verkon yli. (Overview of Amazon Web Services 2024, 1–3.)

Kolme suosituinta julkisen pilvipalveluiden tarjoajaa ovat Amazon Web Services, Microsoft Azure ja Google Cloud. Amazon Web Services kattaa noin 30 % markkinaosuudesta, Microsoft Azure 24 % ja Google Cloudin markkinaosuus on huomattavasti vähäisempi noin 11 % osuudellaan. Muitakin julkisen pilvipalveluiden tarjoajia on markkinoilla, mutta niiden osuus on hyvin pieni. Näihin pienempiin palveluntarjoajiin kuuluu muun muassa: Alibaba Cloud, Oracle Cloud, Salesforce, IBM Cloud ja Tencent Cloud. (Posey 2024.) Vuonna 2022 IT-yhtiö Solita oli teettänyt selvityksen, jossa tutkittiin suomalaisten yritysten pilvipalveluiden käyttöä. Selvityksestä ilmenee, että suomalaisten suuryritysten suosiossa on Microsoftin Azure-palvelu. Selvityksessä mukana olleista suomalaisista suuryrityksistä 82 % käyttää Microsoft Azurea, 34 % Amazon Web Services-palvelua, sekä 27 % Google Cloud-alustaa. Yritykset eivät kuitenkaan kaikki tukeudu vain yhden palvelun varaan, vaan 43 % tutkimuksessa mukana olleista suomalaisista suuryrityksistä hyödyntää monipilviympäristöä. Eurostatin tilastojen perusteella Euroopan kärkimaiksi pilvisiirtymässä ovat osoittautuneet Suomi ja Ruotsi. (Microsoftin pilvi on… 2022.)

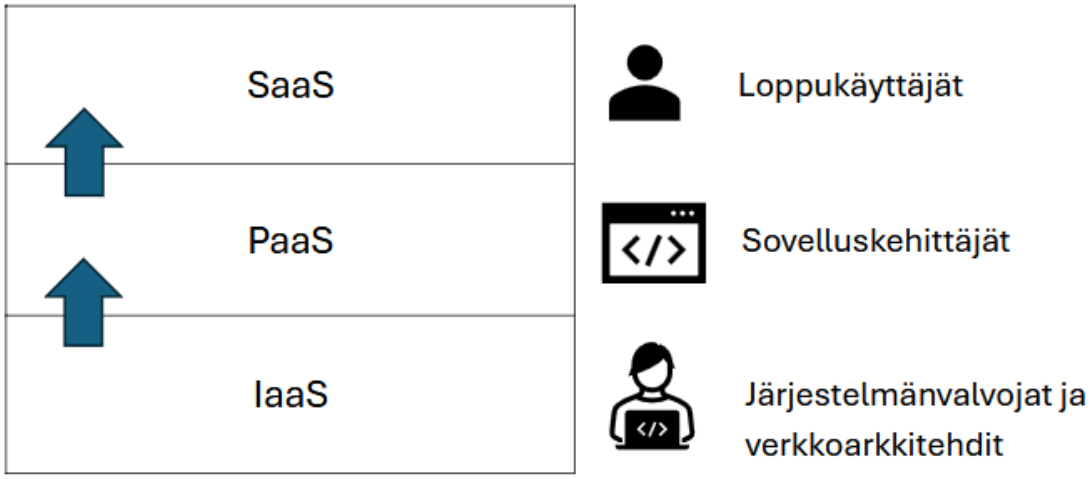
Pilvipalvelumallit

Pilvipalvelut voidaan jakaa kolmeen eri palvelumalliin, joiden tarkoitus on edesauttaa IT-infrastruktuurin hallintaa samalla parantaen kustannustehokkuutta. Mallit ovat nimeltään SaaS, eli ”*Software as a Service*”, PaaS eli ”*Platform as a Service*” ja IaaS, eli ”*Infrastucture as a Service*”. Kun yritykset panostavat oikeanlaisten palvelumallien valintaan, saadaan pilvipalveluteknologioista täysi hyöty ja siitä seuraa innovaatiotoiminnan kiihtyminen ja markkina-aseman vahvistuminen. (Pilvipalveluiden palvelumallit: PaaS… 2024.) Palvelumalleja voidaan käyttää yksittäin tai samanaikaisesti useampia vaihtoehtoja, sillä ne voivat tukevat toinen toisiansa (Cloud Computing Technology 2023, 33).

Jos ajatellaan palvelumallit pinona, niin SaaS-malli on pinon ylimmäisenä. Nimensä mukaisesti palvelun idea on se, että ohjelmisto on saatavilla pilviympäristössä ja sitä hallinnoidaan sieltä käsin. Keskeinen ajatus on se, että käyttäjä ei asenna omalle koneelleen ohjelmistoa, vaan käyttää sitä esimerkiksi verkkoselaimen kautta. Pilvipalveluiden tarjoaja on vastuussa kaikesta ylläpidosta, päivityksistä ja tietoturvasta liittyen ohjelmiston toimintaan. Hyvä esimerkki SaaS-palvelusta on esimerkiksi sähköpostipalvelut tai AWS-ympäristössä AWS WorkSpaces-niminen palvelu. AWS WorkSpaces-palvelu on virtuaalityöpöytäpalvelu, jonka avulla sovellukset ja tiedostot ovat saatavilla mistä tahansa. (Pilvipalveluiden palvelumallit: PaaS… 2024.)

PaaS-malli mahdollistaa infrastruktuurista irtautumisen kehitystyössä edistäen kehittäjien innovointia ja keskittymistä koodin kirjoittamiseen. Kehittäjien ei tarvitse hallinnoida ja huolehtia palvelinympäristöistä ja he voivat vaivattomammin rakentaa, testata ja ajaa sovelluksia. PaaS-palvelumalli ratkaisuna sopii käyttöön silloin, kun kokonaisuudet ovat selkeästi rajattuja, esimerkiksi web-käyttöliittymät ja niiden taustalla olevat järjestelmät. PaaS-malli on suunniteltu sovelluskehityksen helpottamiseksi ja tarjoaa esimerkiksi kehitystyökaluja, tietokantoja ja palvelinympäristöjä. AWS-ympäristössä PaaS-palvelusta hyvä esimerkki on AWS Elastic Beanstalk. AWS Elastic Beanstalk on tarkoitettu sovellusten käyttöönottoon ja hallintaan. (Pilvipalveluiden palvelumallit: PaaS… 2024.)

IaaS-pilvipalvelumallissa IT-infrastruktuuri on vuokrattavissa tarpeiden mukaisesti. Näihin infrastruktuurin osiin kuuluu muun muassa virtuaalikoneet, tallennustilaratkaisut ja verkkoresurssit. Pääperiaate perustuu joustavuuteen ja skaalautuvuuteen ja siihen, että vain käytetyistä resursseista maksetaan. Esimerkiksi Amazon Web Services kokonaisuutenaan voidaan luokitella IaaS-pilvipalvelumalliin. Kaikista palvelumallien vaihtoehdoista IaaS-malli tarjoaa eniten vapautta ja joustavuutta, mutta tuo myös vastuuta ylläpidosta mukanaan. IaaS-malli soveltuu käytettäväksi silloin, kun projekti on monimutkainen tai kokonaisuus ei vaadi muutoksien osalta erityisen nopeaa reagointia. (Pilvipalveluiden palvelumallit: PaaS… 2024.)



Kuvio 2. Palvelumallit ja niiden käyttäjät kuvattuna (Cloud Computing Technology 2023, 34, muokattu)

3.2 Amazon Web Services-pilviympäristö

Amazon aloitti vuonna 2006 tarjoamaan yrityksille IT-infrastruktuuripalveluita verkkopalveluina, nykyään näitä palveluita kutsutaan yleisesti englanninkielisellä termillä ”*cloud computing*”, tarkoittaen pilvipalveluita. Amazon Web Services (AWS) tarjoaa nykypäivänä matalakustanteisen pilvi-infrastruktuurialustan, joka on luotettava ja skaalautuva. Palveluita käyttävät sadat tuhannet yritykset kattaen maailmanlaajuisesti 190 maata. (Overview of Amazon Web Services 2024, 1.) Nishimura (2022) toteaa kirjansa ”*AWS for non-engineers*” esittelykappaleessa että yritykset, kuten esimerkiksi hyvin tunnetut Airbnb, Adobe, Disney, Comcast ja McDonald’s käyttävät AWS-palveluita verkkosivujensa isännöintiin tai IT-infrastruktuurin hallintaan. AWS pilvipalveluiden tarjoajana on siis keskeisessä osassa ihmisten arkea internetmateriaalien kuluttamisessa. (Nishimura 2022.)

Well-Architected Framework-arkkitehtuurikehys

Well-Architected Framework-termillä tarkoitetaan hyvin optimoitua arkkitehtuurikehystä, eli parhaita käytänteitä pilvi-infrastruktuurin rakentamiseksi. Amazon Web Services on määrittänyt kuusi pilaria, joihin parhaiden käytänteiden toimet perustuvat, sillä niitä noudattamalla voidaan saavuttaa kaikista turvallisin, vikasietoisin, tehokkain ja suorituskykyisin pilvi-infrastruktuuri. Myös turvallisuus, luotettavuus ja kustannustehokkuus on otettu pilarissa huomioon. (AWS Well-Architected

Framework 2024;Nishimura 2022.) Nishimura (2022) kiteyttää pilareiden keskeisen tarkoituksen tekstissään siten, että pilarin parhaita käytänteitä noudattamalla voidaan saavuttaa vakaa ja kustannustehokas IT-ympäristö, jonka ansiosta enemmän aikaa voidaan sijoittaa palveluiden ja tuotteiden kehittämiseen. Hän on tiivistänyt pilareiden kuvaukset seuraavasti:

- ”*Operational Excellence*” eli toiminnallisuus: Järjestelmien käyttö, valvonta, kehittäminen.
- ”*Security*” eli turvallisuus: Järjestelmien ja informaation suojaaminen.
- ”*Reliability*” eli luotettavuus: Toimintahäiriöiden torjuminen ja niistä palautuminen.
- ”*Performance efficiency*” eli suorituskyvyn tehokkuus: Laskentaresurssien tehokas käyttö.
- ”*Cost optimization*” eli kustannustehokkuus: Ylimääräisten kulujen välttäminen.
- ”*Sustainability*” eli kestävyys: Pilvityöskentelyn ympäristövaikutusten minimointi. (Nishimura 2022.)

Hyvin optimoitu arkkitehtuurikehys auttaa ymmärtämään päätöksien hyvät ja huonot puolet systeemejä rakentaessa Amazon Web Services-palvelussa ja se on tarkoitettu käytettäväksi teknologian rooleissa työskenteleville henkilöille. Näitä rooleja voivat olla esimerkiksi: teknologiajohtajat, arkkitehdit, kehittäjät tai operaatiotiimin kuuluvat henkilöt. Arkkitehtuurikehyksen käyttö edistää myös arkkitehtonisia parhaita käytänteitä pilviympäristöjä suunnitellessa ja hallinnoidessa, sillä jatkuva arviointi auttaa tunnistamaan kehityksen kohteita. AWS-verkkosivuilla on todettu, että he uskovat hyvien käytänteiden mukaisesti toteutettujen järjestelmien edesauttavan yritysten menestystä. He myös selventävät, että AWS-palveluiden kehittyessä ja heidän oppiessa lisää yhteistyökumppaneiltaan, parhaiden käytänteiden määritelmää jatkuvasti kehitetään. (AWS Well-Architected Framework 2024.)

3.3 DevOps-toimintamalli

Pihlajamäki (n.d.) tiivistää kirjoittamassaan artikkelissa seuraavasti: *”DevOps on joukko käytäntöjä ja kulttuurifilosofiaa, joita on mahdollista toteuttaa työkaluilla ja tekniikoilla, jotka automatisoivat ja integroivat ohjelmistokehityksen ja ylläpitotiimien välistä viestintää ja yhteistyötä.”*

DevOps-toimintamalli muodostuu sanoista kehitys (engl. *Development*) ja toiminta (engl. *Operations*) eli nimensä mukaisesti se yhdistää prosesseja jatkuviksi perinteisen ohjelmistokehitysmallin

sijaan. Perinteisessä ohjelmistokehitysmallissa oli tapana pitää käyttöönotto- ja muut tukitoiminnat erillään. DevOps-toimintamalli ei siis varsinaisesti ole mitään tietynlaista teknologiaa, työkaluja tai ohjelmointikieliä, vaan paljon laajempi monimuotoinen yhdistelmä ihmisiä, työkaluja ja kulttuuria. Ohjelmistokehittäjien ja ylläpitäjien välinen yhteistyön edistäminen on tärkeässä asemassa. DevOps-toimintamallin tarkoitus on lisätä nopeutta ja laatua ohjelmistoversioiden kehittämisen elinkaaren aikana. Myös luotettavuus lisääntyy, kun prosesseja automatisoidaan ja nopeutetaan. Tärkeitä käsitteitä, joita DevOps-toiminta käsittelee, ovat: jatkuva integraatio ja jatkuva toimitus (engl. *CI/CD, Continuous Integration and Continuous Delivery*), automaatio ja yhteistyö. DevOps-kulttuurilla tarkoitetaan yhteistyön ja viestinnän omaksumista uutena työtapana, joka suuntaa ihmiset, prosessit ja työkalut yhtenäisempään asiakaslähtöisyyteen. (Pihlajamäki n.d.) Hurwitz ja Kirsch (2020) toteavat, että DevOps-toimintamallin perusperiaate perustuu siihen, että kehitys ja julkaiseminen, jotka ovat sovelluskehityksen kaksi isointa vaihetta, eivät ole erillään. He selventävät, että niiden yhdistäminen on edellytys sille, että CI/CD on mahdollisimman tehokasta. (Hurwitz & Kirsch 2020, 156.)

Jatkuva integraatio

Jatkuva integraatio (engl. *Continuous integration*) on sovelluskehitykseen kuuluva käytäntö, jossa kehittäjät säännöllisesti yhdistävät koodimuutoksensa yhteiseen repositorioon ja automatisoidut koonnit ja testit ajetaan. Jatkuva integraatio auttaa saavuttamaan ohjelmiston laadun parantamisen sekä löytämään ja korjaamaan ohjelmistovirheitä nopeammin. Myös ohjelmistopäivitysten validointi onnistuu nopeammin, kun jatkuvaa integraatiota sovelletaan. (What is DevOps? n.d.)

Jatkuva toimitus

Jatkuva toimitus (engl. *Continuous delivery*) on myös sovelluskehitykseen kuuluva käytäntö, joka liittyy vahvasti suoraan jatkuvaan integraatioon. Se on seuraava askel, jossa tehdyt koodimuutokset julkaistaan testi- tai tuotantoympäristöön. Automaatiolla koodimuutoksiin suoritetaan koonti (engl. *Build*) ja testaus, sekä valmistellaan muutokset julkaisuvalmiiksi tuotantoympäristöön. (What is DevOps? n.d.)

Infrastruktuuri koodina

Infrastruktuuri koodina (engl. *Infrastructure as a Code*) kuuluu olennaisesti osaksi DevOps-toimintamallia ja se tarkoittaa IT-infrastruktuurin tilan ja konfiguraation mallintamista ja hallintaa ohjelmakoodina (IaC – Infrastruktuurin hallinta Funidatalla 2024). Kun infrastruktuuri on määritelty koodina, sen hallintaa voidaan käsitellä hyvin samaan tapaan kuin ohjelmistokoodia käsiteltäisiin. Infrastruktuuri ja palvelimet voidaan helposti ja nopeasti pystyttää uudelleen, ja ne ovat nopeammin päivitettävissä ja monistettavissa. Koodin käyttämisen etuja on esimerkiksi se, että konfiguraatioiden muuttaminen on standardisoitua ja hyvin toistettavissa. Tämä tarkoittaa sitä, että ohjelmistokehittäjien ja järjestelmänvalvojien ei tarvitse käyttää aikaansa manuaaliseen konfiguraatioon koskien käyttöjärjestelmiä tai järjestelmä- ja palvelinohjelmistoja. (What is DevOps? n.d.)

Testausautomaatio

Testiautomaatio on toimintatapa, jossa ohjelmistokehityksen tuote tarkistetaan ja validoidaan automaattisesti. Testiautomaation tarkoituksena on varmistaa, että ennalta määritellyt laatuvaatimukset täyttyvät eri vaiheissa. Ohjelmistokehityksessä testikohteina ovat yleensä koodityyli, toiminnallisuus ja käyttäjäkokemus. Ennen testiautomaatiotyökalujen luontia eri testikäytänteet olivat manuaalinen prosessi, joka oli hidasta, kallista ja todella virhealtista. Tänä päivänä testiautomaatio on osa parhaita käytänteitä ohjelmistokehityksessä, koska ongelmien aikainen havaitseminen tuotantoketjussa on kustannustehokasta ja nopeampaa. DevOps-toimintamallin parhaisiin käytänteisiin kuuluu se, että automaattisia testejä ajetaan mahdollisimman ajoissa ja useasti CI/CD-julkaisuputkessa. Eri menetelmiä testaukseen ovat esimerkiksi: yksikkötestaus (engl. *Unit testing*), integraatiotestaus (engl. *Intregration testing*) ja e2e-testaus eli end-to-end-testaus. (Hristov n.d.)

4 Elli-sovelluksen infrastruktuuri

4.1 Elli-sovelluksessa käytetyt työkalut ja niiden käyttötarkoitus

Amazon Elastic Beanstalk -palvelu

Amazon Elastic Beanstalk on AWS:n tarjoama PaaS-palvelu, joka julkaistiin vuonna 2011 (Wadia 2018, 54). Elastic Beanstalk-palvelu tarjoaa alustan, jolla pystyy kehittämään, julkaisemaan ja hallitsemaan sovelluksia. Se pitää huolen koko sovelluksen julkaisuprosessista, muun muassa EC2-instanssien kapasiteetin, automaattisen skaalauksen ja kuormanjakamisen osalta. Elastic Beanstalk-palvelulla sovelluksen lataamiseen pystytään käyttämään useita eri vaihtoehtoja, kuten AWS Management Console-portaalia tai jopa Visual Studio Code-tekstieditoriohjelmaa. Itse Elastic Beanstalk palveluna on ilmainen käyttää, mutta sen avulla käyttöönotetut AWS-resurssit maksavat. (Wadia 2018, 54.) Elastic Beanstalk-palvelu hyödyntää AWS:n ydinpalveluita, kuten Elastic Compute Cloud (EC2), Elastic Container Service (ECS), Auto Scaling ja Elastic Load Balancing (ELB) (Why AWS Elastic Beanstalk? n.d.). Elastic Beanstalk-palelua käytetään yleisesti esimerkiksi websovelluksiin, sisällönhallintajärjestelmiin (CMS) tai API-backend-ratkaisuihin (Dalbhanjan 2015).

Elastic Beanstalk-palvelun ollessa PaaS-palvelu, se automaattisesti pystyttää ja käyttää tarvitsemiansa AWS-resursseja, jotka tarvitaan sovelluksen ajamiseen. Yksi näistä resursseista on EC2-instanssit, eli ”*Elastic Compute Cloud*”-virtuaalitietokoneet. EC2 itsessään on IaaS-palvelumallia soveltava alusta, jonka itsenäinen käyttö vaatii laajempaa suunnittelua ja konfigurointia suuremmassa infrastruktuurisuunnitelmassa. Elastic Beanstalk-palvelu pystyy hallinnoimaan EC2-palvelun resursseja ja voidaan ajatella sen valmistelevan nämä instanssit käyttövalmiiksi sovelluksen asennusta varten. Näiden hallinnoitujen instanssien ominaisuuksia pystyy kuitenkin itse konfiguroimaan tarvittaessa, eli Elastic Beanstalk-palvelu ei kuitenkaan estä itse muuttamasta instanssien ominaisuuksia, jos niin on tarve tehdä. (Introduction to AWS… 2023.)

AWS Elastic Beanstalk- palvelua konfiguroitaessa ensimmäistä kertaa asetetaan seuraavat palvelut käyttöön: VPC (engl. *Virtual Private Cloud*), aliverkot (engl. *Subnet*) ja Security Group-asetukset. VPC-alueella on käytössään joko julkiset aliverkot tai yksityiset aliverkot. Aliverkot ovat IP-osoitteiden alueita VPC-alueen sisällä. Julkinen aliverkko käyttää Internet gateway-komponenttia VPC-alueen ja Internetin väliseen kommunikaatioon ja julkisen aliverkon instansseilla on julkinen IP-osoite. Internet gateway-komponentti on osa VPC-alueen ominaisuuksia. Yksityinen aliverkko

käyttää NAT-palvelua (engl. *Network Access Translation*), jolloin aliverkon instansseilla ei ole julkista IP-osoitetta, mutta ne konfiguroidaan tietoturvallisesti kommunikoimaan resurssien kanssa. NAT-palvelu on Availability Zone-kohtainen asetus infrastruktuurissa. Availability Zone-alue, eli saatavuusalue on VPC-alueen sisällä oleva eriytetty alue, jonka sisällä olevat resurssit sijaitsevat eri maantieteellisissä alueissa. Näiden voidaan ajatella olevan konesaleja, joihin palvelut eriytetään ja tällä saavutetaan parempi vikasietoisuus. (Amazon Virtual Private… 2024, 1,2,8,12,222;AWS Elastic Beanstalk 2025, 1378.)

Oletusarvoisissa Elastic Beanstalk-palvelun konfiguraatioissa Security Group-asetukset määritellään instanssitasolla sallimaan sisään tuleva liikenne portissa 80. Tämä tarkoittaa sitä, että kuormanjakaja ohjaa http-liikenteen sovellukselle. Oletuksena mikään muu portti ei ole asetettu olemaan auki. Oletusasetuksiin kuuluu myös muun muassa S3-bucketin luominen, CloudWatch-hälytykset, CloudFormation-pinot ja Domain-nimi sovellukselle. (AWS Elastic Beanstalk 2025, 9.)

Amazon S3 -palvelu

Amazon Simple Storage Service, eli lyhennettynä S3 on objektien tallennuspalvelu. Data tallennetaan objekteina bucketteihin, jotka ovat säiliöitä objekteille. Objektit ovat yksiköitä tai olioita, jotka koostuvat objektidatasta ja metadatasta. Metadata kuvailee objektia ja on joukko nimiarvopareja, kuten esimerkiksi viimeisin muokkauspäivä tai sisällön tyyppi. Amazon S3-palvelua käytetään erilaisten datatyyppien suojaamiseen ja tallentamiseen, kuten datajärvet, verkkosivustot, mobiilisovellukset, varmuuskopiot, palautukset, arkistointi, yrityssovellukset, IoT-laitteet ja big data-analytiikka. S3-palvelulla on mahdollista hallita dataa, kuten optimoida ja organisoida dataa, sekä määrittää pääsyoikeuksia eri tarpeiden mukaisesti. (What is amazon S3? n.d.)

Amazon CloudFormation -palvelu

CloudFormation on palvelu, joka auttaa mallintamaan ja määrittämään AWS-resursseja. Sen käytön hyödyt perustuvat mallipohjien käyttöön siten, että resurssien hallintaan ei tarvitse käyttää turhaa aikaa. (What is AWS CloudFormation? n.d.) CloudFormation-palvelu on tarkoitettu järjestelmänvalvojille, verkkoarkkitehdeille tai muille IT-henkilöstön jäsenille. Mallipohjien käyttäminen mahdollistaa versionhallinnan ja nopean toistettavuuden hyödyntämisen infrastruktuurin pystyttämiseksi. CloudFormation-palvelua suositellaan, jos infrastruktuurin eri osa-alueita on tarve hallita

yksityiskohtaisesti. (Overview of Deployment Options on AWS n.d.) Mallipohjat ovat YAML- tai JSON-tiedostoja, joita CloudFormation-palvelu käyttää suunnitelmapohjina resurssien rakentamiseen. Mallipohjassa voidaan kuvata esimerkiksi EC2-instanssin ominaisuuksia, kuten instanssityyppi, AMI ID, block device mappings-asetukset tai instanssin avain-arvoparin nimi. (How CloudFormation works n.d.)

Amazon IAM -palvelu

Amazon Identity and Access Management eli IAM, on AWS-resurssien pääsyoikeuksien hallintaan tarkoitettu palvelu. IAM-palvelulla voidaan hallita pääsyoikeuksia palveluihin ja resursseihin. Hallittavia kohteita voivat olla käyttäjät, ryhmät ja roolit. Näiden kohteiden oikeuksia, eli käyttöoikeuskäytäntöjä (engl. *Policy*) voidaan hallita pääsyn sallimiseksi tai estämiseksi. Käyttäjillä voidaan tarkoittaa ihmistä tai sovellusta, eli IAM-palvelun avulla resurssien ja palveluiden välistä toimintaa voidaan myös hallita. Ryhmällä tarkoitetaan sitä, kun käyttäjät kuuluvat ryhmään ja perivät ryhmän käyttöoikeuskäytännöt sen sijaan, että heille asetettaisiin yksittäin käyttöoikeuskäytänteitä. Käyttäjä voi kuulua samanaikaisesti useaan eri ryhmään. Roolit ovat tarkoitettu väliaikaisen pääsyoikeuden sallimiseksi esimerkiksi ulkoiselle henkilölle, joka tarvitsee pääsyn AWS-resursseihin, palveluihin, IAM-käyttäjiin tai sovelluksiin. IAM-käyttöoikeuskäytännöt ovat JSON-dokumentteja, jotka koostuvat yhdestä tai useammasta lausekkeesta. (Adetunji 2022.)

AWS Certificate Manager -palvelu

AWS Certificate Manager, eli lyhennettynä ACM, on SSL/TLS X.509-sertifikaattien ja avaimien luomiseen, säilyttämiseen ja uusimiseen tarkoitettu palvelu. X.509 on ISO-standardi kryptografisille salausvarmanteille. Nämä sertifikaatit ja avaimet ovat tarkoitettu suojaamaan verkkosivuja ja sovelluksia. ACM-palvelulla voidaan joissakin tapauksissa ottaa sertifikaatti suoraan käyttöön sen kautta tai vaihtoehtoisesti ladata ACM-hallinnointijärjestelmään kolmannen osapuolen sertifikaatti. Palvelussa olevat sertifikaatit ovat aluekohtaisia (engl. *Region*). ACM-sertifikaatit ovat X.509 SSL/TSL-sertifikaatteja, jotka sitovat verkkosivun identiteetin ja organisaatiotiedot julkiseen avaimeen. (AWS Certificate Manager 2024.)

ACM-palvelua voidaan käyttää yhdessä Elastic Beanstalk-palvelun ja sen kuormanjakaja-asetusten kanssa. Elastic Beanstalk-palvelun asetuksissa voidaan muokata kuormanjakajan asetuksia, jotka

käyttävät ACM-palvelua SSL-sertifikaatin hakemiseen ja todentamiseen. Konfiguroimalla ja lisäämällä SSL-sertifikaatin, mahdollistetaan turvallinen https-yhteys sovelluksen liikenteeseen (AWS Certificate Manager 2024).

MongoDB Atlas -palvelu

MongoDB on nykyaikaiseen sovelluskehitykseen suunniteltu yleiskäyttöinen dokumenttitietokantaratkaisu, jonka arkkitehtuuri soveltaa skaalautumista järjestelmän kuormituksen mukaisesti. MongoDB soveltuu erityisen hyvin pilviympäristöihin. Dokumenttimallinen tietokantaratkaisu tarjoaa joustavan tavan työskennellä eri lähteistä tulevasta datasta, vaikka se olisi monimutkaista ja nopeasti muuttuvaa. MongoDB muuntaa dokumentit JSON- tai BSON-tiedostomuotoihin nopeampia sisäisiä yhteyksiä ja datatyyppien laajempaa tukea varten, mutta kehittäjänäkökulmasta voidaan puhua MongoDB:n olevan JSON-tietokanta. (MongoDB Basics n.d.)

MongoDB Atlas on MongoDB:n luoma pilvipohjainen avoimen lähdekoodin (engl. *Open source*) tietokantapalvelu, joka on täysin hallinnoitu. Atlas on saatavilla eri pilvipalvelutarjoajien ympäristöihin, kuten Google Cloud Platform, Azure ja Amazon Web Services. Skaalautuvuus, saatavuus ja huipputasoinen automaatio ovat MongoDB Atlas-palvelun hyödyllisiä ominaisuuksia sen lisäksi, että se hyödyntää MongoDB-ekosysteemin integraatioita, ajureita ja työkaluja. (Duggal 2025.)

Github -palvelu

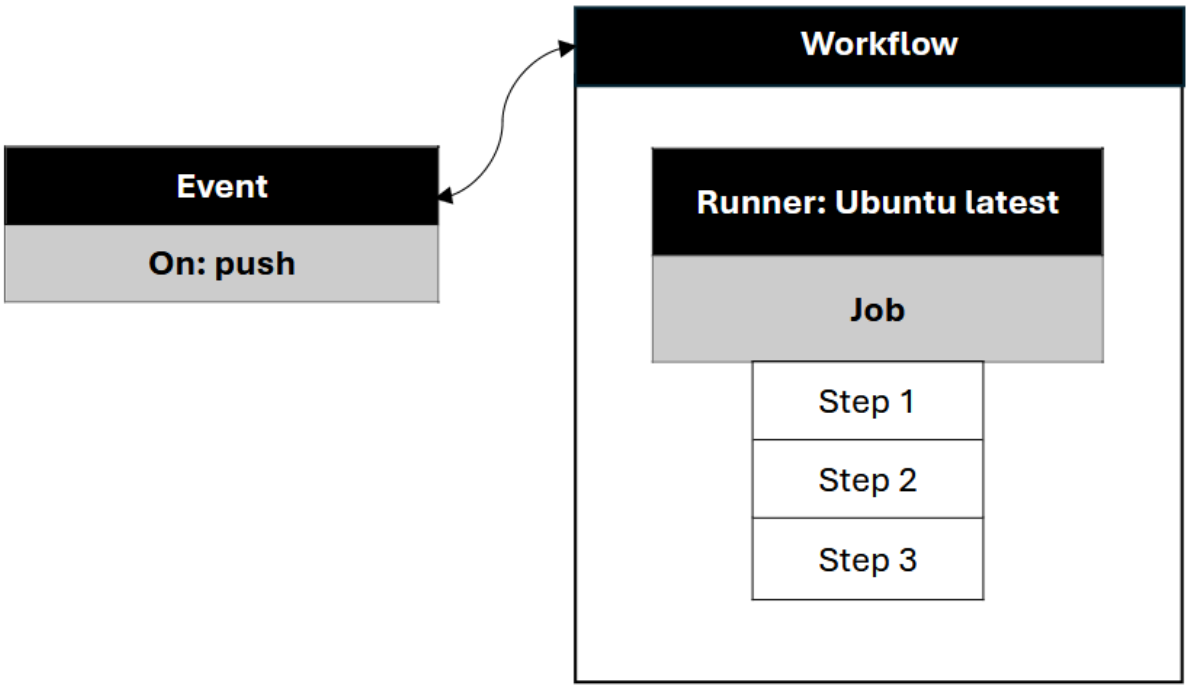
Githubin voidaan ajatella kuuluvan pilvipalveluiden kategoriaan. Se on verkkosivusto, jonne kehittäjät voivat tallentaa koodia ja hallita sitä esimerkiksi koodin seurannan ja muutoksien osalta. Githubin toiminta perustuu vahvasti Gitiin ja versionhallintaan. Git on Linus Torvaldsin vuonna 2005 kehittämä avoimen lähdekoodin versionhallintajärjestelmä, joka noudattaa hajautettua toimintaperiaatetta. Se tarkoittaa sitä, että koodikanta ja sen koko historia ovat saatavilla kaikille osallistuville kehittäjille, jolloin branching- ja merge-toiminnot ovat helpompia suorittaa. Versionhallinta perustuu näihin toimintoihin ja tarkoittaa yksinkertaistettuna sitä, että kun koodikannasta kopioidaan osa koodia, siihen voidaan tehdä turvallisesti muutoksia vaikuttamatta muun projektin osiin. Tämä on branching-toiminto. Kun muutokset ovat valmiita, voidaan koodi lopuksi yhdistää takaisin

päälähdekoodin osaksi ja tätä kutsutaan merge-toiminnoksi. Kaikki tehdyt toimenpiteet ja muutokset jäävät seurantaan ylös ja niitä voidaan perua tarpeen mukaan. (What is Github? A Beginner's Introduction to Github 2023.)

Github Actions on Githubissa saatavilla oleva alusta, jolla voidaan automatisoida koonnin, testauksen ja julkaisun osat kuten esimerkiksi uusien muutosten testaaminen, julkaisujen käyttöönotto tai koodianalyysien suorittaminen ohjelmistovirheiden löytämiseksi. Github tarjoaa virtuaalikoneita muun muassa Linux-, Windows- ja macOS-alustoilla, mutta on myös mahdollista rakentaa itse isännöity ympäristö esimerkiksi omaan konesaliin tai pilvi-infrastruktuuriin. Github Actions-alustan kuvataan olevan enemmän ja menevän pidemmälle kuin DevOps-toimintamalliajattelu, sillä se mahdollistaa tapahtumien välisten interaktioiden laukaista toimenpiteitä. (Foster n.d.;Understanding GitHub Actions n.d.)

Foster (n.d.) kuvailee Github Actions-alustan koostuvan viidestä ydinkomponentista, jotka ovat: Workflow-prosessit, Event-tapahtumat, Job-vaiheet, Actions-toiminnot ja Runner-komponentit. Workflow-prosessi on konfiguroitavissa oleva automatisoitu prosessi, jonka sisällä on job-vaiheita. Job-termillä kutsutaan joukkoa vaiheita, joita suoritetaan virtuaalikoneen toimesta (engl. *Virtual machine runner*) tai kontin (engl. *Container*) sisällä. Oletusasetus job-vaiheiden ajamiseen on samanaikainen ajo, mutta ne voidaan myös asettaa riippuvaiseksi toisesta job-vaiheesta. Tämä tarkoittaa sitä, että ajettava job-vaihe odottaa sen job-vaiheen valmistumista, mistä se on asetettu olemaan riippuvainen. Job-vaiheeseen voidaan asettaa useita eri toimenpiteitä (engl. *Steps*), jotka voivat esimerkiksi ajaa koodiskriptejä tai uudelleen käytettäviä lisäosatoimintoja (eng. Action) workflow-prosessin yksinkertaistamiseksi. Job-vaiheita ajetaan järjestyksessä ja ne ovat riippuvaisia toisistaan, sekä toimenpiteiden tiedot ovat saatavilla toimenpiteiden välillä. Esimerkkinä tästä voidaan kuvata vaihetta, joka kokoaa sovelluksen ja sitä seuraa toinen vaihe, joka testaa juuri tuon kootun sovelluksen. (Understanding GitHub Actions n.d.)

Workflow-prosessin toiminta perustuu repositorioon asetettuun YAML-tiedostoon ja se voidaan laukaista ajettavaksi esimerkiksi repositoriotapahtumalla, manuaalisesti tai asetetussa aikataulussa. Yhdessä repositoriossa voi olla useita workflow-tiedostoja, jotka jokainen voivat suorittaa erilaisia toimintoja tai jopa laukaista toisia workflow-tiedostoja. Workflow-prosessin laukaisevaa aktiviteettia kutsutaan termillä ”*Event*”. (Understanding GitHub Actions n.d.)

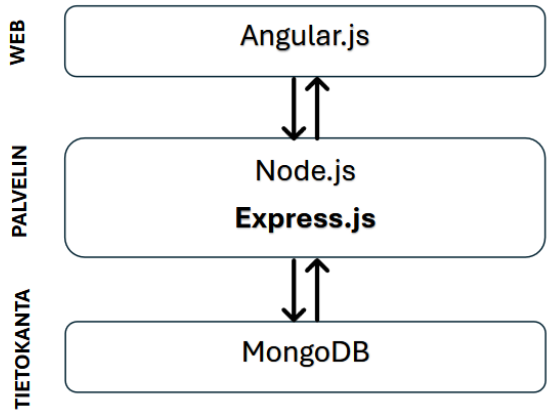


Kuvio 3. Github Actions-komponentit kuvitettuna (Foster n.d., muokattu)

Github Actions-alusta voi käyttää **Github Secrets**-ominaisuutta, joka on tarkoitettu arkaluontoisen datan säilyttämiselle, kuten esimerkiksi API-avaimille, pääsyavaimille (engl. *Access token*) tai salasanoille. Secret-muuttujat ovat kryptattuja ympäristömuuttujia (engl. *Variable*), joita hyödynnetään Github Actions-alustan workflow-prosessien kanssa eri tasoilla, kuten organisaatio-, repositorio- tai ympäristötasolla. Organisaatiotason secrets-muuttujat voidaan ajatella olevan ylimmän tason secret-muuttujia, sillä samaa muuttujaa voidaan jakaa ja käyttää useiden eri repositorioiden kanssa. Tällaisen jaetun muuttujan muokkaaminen vaikuttaa kaikkiin niihin repositorioihin, joissa se on käytössä. Secrets-muuttujan käyttäminen Actions-alustalla tapahtuu kutsumalla secret-muuttujaa workflow-tiedostossa ja esimerkiksi organisaatio- ja repositoriotason secret-muuttujat luetaan workflow-tiedoston alussa, kun taas ympäristötason secret-muuttujat luetaan silloin kun job-vaihe viittaa ympäristöön. Secret-muuttuja on näkyvissä vain workflow-ajolle, eli käyttäjät, Githubin käyttöliittymä tai toiset Actions-toiminnot eivät pääse arkaluontoiseen tietoon käsiksi missään työn vaiheissa. (About secrets n.d.; What is a GitHub Actions Secret? n.d.)

4.2 Infrastruktuurin kuvaus

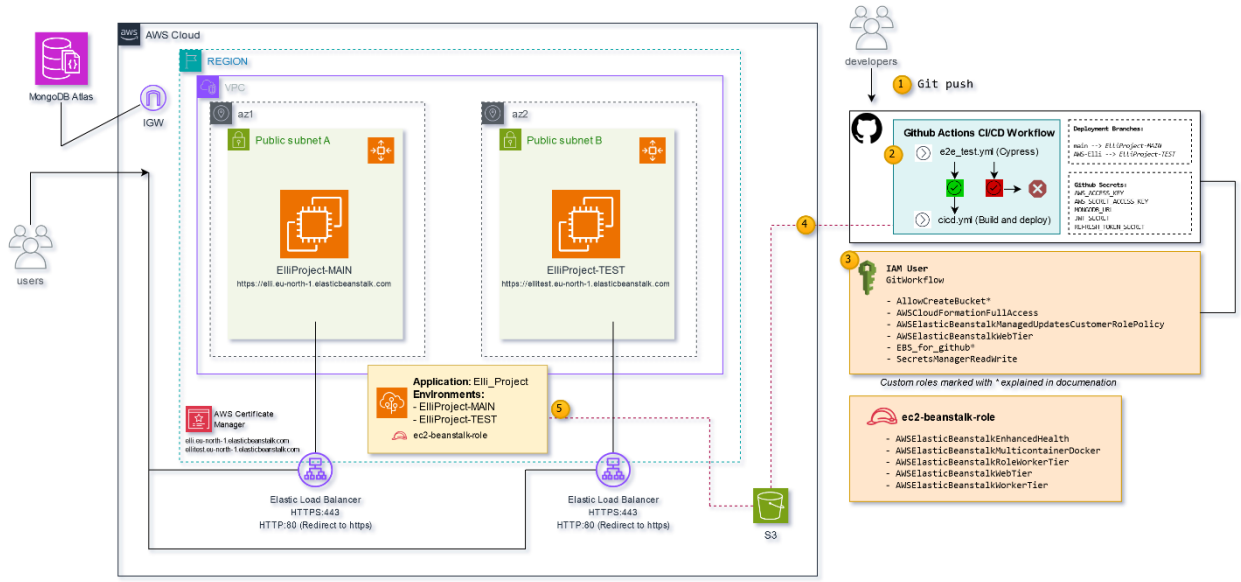
Sovelluksen kehityksessä on käytetty MEAN stack-nimistä JavaScript-ohjelmointikieleen painottuvaa teknologiakokoelmaa, eli MEAN-ohjelmistopakkaa. MEAN-sanan kirjaimet muodostuvat sen käytetyistä teknologioista, jotka ovat: MongoDB, Express, AngularJS ja NodeJS. (What Is the MEAN Stack? n.d.) Elli-sovellus on toteutettu Angular-ohjelmistokehyksellä (engl. *Framework*) ja se ottaa yhteyden sovelluksen backendiin, joka suorittaa tarvittavat kyselyt käytettyyn tietokantaan.



Kuvio 4. MEAN-ohjelmistopino Elli-sovelluksessa (What Is the MEAN Stack? n.d., muokattu)

Alla olevassa infrastruktuurikaaviossa (kts. Kuvio 5) on kuvattu koko infrastruktuurin rakenne sisältäen käyttäjien tapahtumat, sekä kehittäjien tapahtumat. Infrastruktuuriin kuuluu olennaisesti AWS-pilviympäristö hyödyntäen Elastic Beanstalk-palvelua, ulkoinen MongoDB-tietokanta ja Github työkaluineen. Kehittämisprosessi on asetettu sellaiseksi, että se käyttää ympäristömuuttujia siten, että samaa sovelluskoodia voidaan kehittää paikallisesti sekä pilviympäristöön. Tämä tarkoittaa sitä, että kun Githubin kautta tehtävään julkaisutiedostoon on asetettu node_modules-ympäristömuuttuja, se saa arvokseen ”*production*” tai ”*development*”. ”*Production*”-ympäristömuuttujan sisältävässä koodissa esimerkiksi URL-osoite on määritelty pilviympäristöä vastaavaksi, kun taas ”*development*”-ympäristömuuttujaan on asetettu paikallinen osoite. Tässä Elli-sovelluksen toteutuksessa voidaan ajatella olevan kolme eri ympäristöä, joista kaksi ajetaan pilviympäristössä ja yksi on paikallinen ympäristöversio. Koodin laatuun liittyen ilmeni kuitenkin puutteita, esimerkiksi pilviympäristöjen URL-osoitteet vaihdettiin manuaalisesti ohjelmakoodin eri tiedostoihin. Tämä voi aiheuttaa virheitä ja hidastaa julkaisun toimenpiteitä.

Pilviympäristössä ajettavat kaksi versiota vastaavat kehitysympäristöä ja tuotantoympäristöä. Kummatkin ympäristöversiot (engl. *Environment*) ovat saman Elastic Beanstalk-sovelluskonfiguraation (engl. *Application*) hallinnan alla omina ympäristöinään.

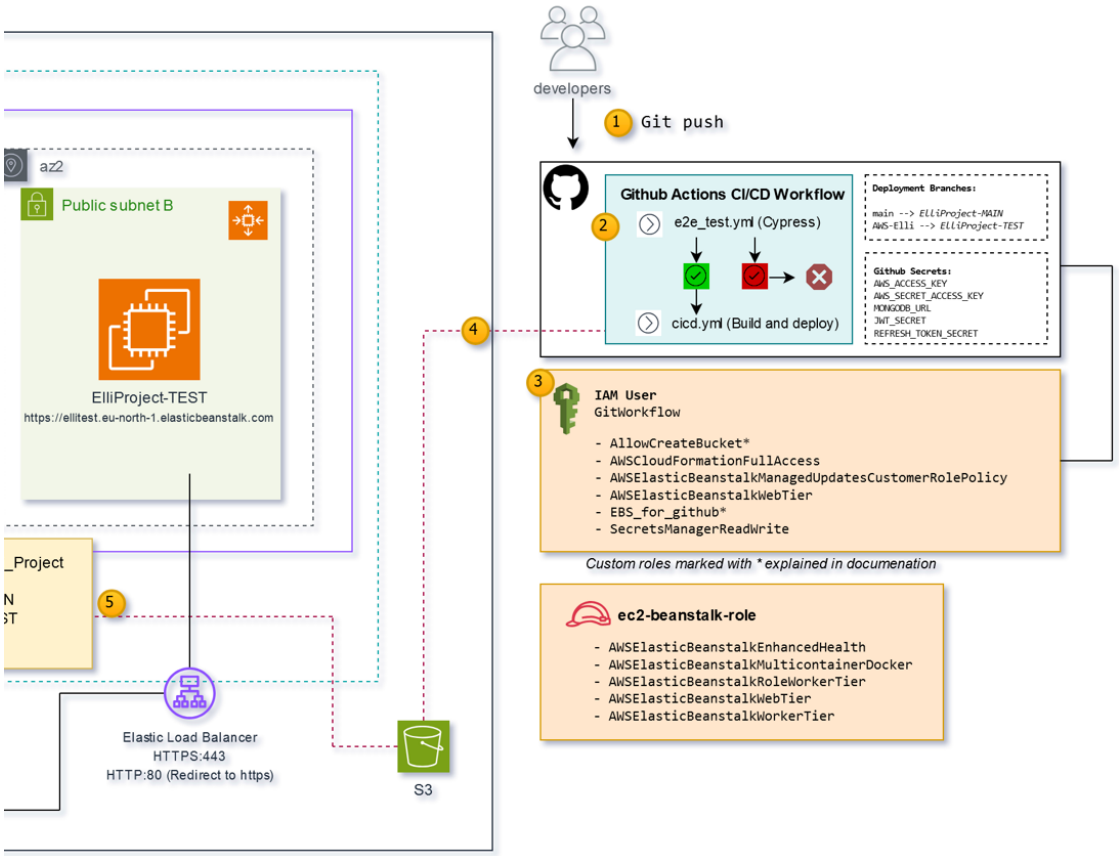


Kuvio 5. Elli-sovelluksen infrastruktuurirakenne kokonaisuudessaan AWS-ympäristössä

Elli-sovelluksen ohjelmakoodia säilytetään ja hallitaan Github-repositoriossa. Kaikilla sovelluksen kehittäjillä on pääsy repositorioon ja se sijaitsee Jyväskylän ammattikorkeakoulun omistaman organisaatioryhmän alla, jota käytetään Ticorporate-opintojakson töissä. Kuvio 6:ssä on eritelty kehittäjien osuus, jonka toiminnot laukaistaan Git-toiminnoilla:

1. Kehittäjä tekee push-toiminnon haarassa, joka laukaisee Githubin Actions-toimenpiteet.
2. Github Actions-alusta suorittaa e2e_test.yml-tiedoston, joka sisältää end-to-end-testauksen sovelluksen koodiin. Jos testit eivät mene läpi, workflow-prosessin suorittaminen lopetetaan. Jos testit menevät läpi, workflow-prosessi siirtyy ajamaan cicd.yml-tiedostoa, joka suorittaa sovelluksen valmistelut Elastic Beanstalk-palvelua varten.
3. Cicd.yml-tiedostossa on määritelty IAM-rooli, joka tarkastetaan ajamisprosessissa. Tällä IAM-roolilla on oikeus tehdä tarvittavat toimenpiteet AWS-ympäristössä.
4. Cicd-yml-tiedoston lopuksi julkaisupaketti lähetetään S3-bucketiin. Se on zip-tiedosto, joka sisältää tarvittavat tiedostot Elastic Beanstalk-palvelua varten.

5. Elastic Beanstalk-palvelu käyttää ladattua zip-tiedostoa sovelluksen pystyttämiseen.



Kuvio 6. Kehittäjätoiminnot esitettynä infrastruktuurikaaviossa

Elli-sovelluksen testaus ja julkaisu on toteutettu Github-repositorioon automatisoituun julkaisuputkeen. Julkaisuputki hyödyntää Github Actions-alustan ominaisuuksia ja sille on konfiguroitu kaksi yml-tiedostoa, jotka erottavat toisistaan testaustoimenpiteet ja sovelluksen julkaisun pilviympäristöön. Nämä workflow-prosessin tiedostot ovat riippuvaisia toisistaan ja julkaisutiedostoon siirrytään vain, jos testaustoimenpiteet ovat suoritettu onnistuneesti. Ympäristön suunnittelun idea perustuu siihen, että kehitystoimenpiteet tehtäisiin aina ensin kehitysympäristöön, jonka jälkeen ne voidaan julkaista tuotantoympäristöön. Tuotantoympäristön tarkoitus olisi olla aina viimeisin toimiva versio ja tästä syystä kehitysympäristö on eriytetty rakenteellisesti sen kanssa. kehitysympäristöön voidaan suorittaa testiajoja ja päivityksiä, ilman että ne vaikuttavat tuotantoympäristön toimintaan. Kummatkin ympäristöt tästä huolimatta käyttävät samaa tietokantaa, vaikka olisi asianmukaisempaa eriyttää nämä resurssit.

Elli-sovelluksen tärkeimpiä työkaluja sovelluksen toiminnan kannalta ovat Angular-sovelluskehys ja sen kanssa toimiva MongoDB Atlas-tietokanta. Julkaisuprosessi hyödyntää sovelluksen rakennetta, mutta sen tärkeimmät osat ovat Github ja Github Actions-alusta. Näihin olennaisesti liittyy Github Secrets-muuttujien hyödyntäminen workflow-prosessin tiedostoissa, sekä erilaisten valmiiden Actions-komponenttien sisällyttäminen. Lopputuloksen kannalta AWS-resurssit ja niiden konfiguroiminen on erittäin tärkeä osa, sillä ilman niitä sovellus ei tule ajettavaksi pilviympäristössä.

5 Toteutus

5.1 Aineiston keruu

Opinnäytetyön tutkimuksellisuuteen liittyvät aineistot olivat pääasiassa Amazon Web Services-sivuston omia ohjedokumentaatioita palveluihin ja resursseihin liittyen. Näitä aineistoja tutkittiin ja havainnoitiin keräämisen vaiheissa, ja ne olivat olennaisena tukena kehitystyön etenemisessä. Kehitystyön aiheen ollessa migraatio nimenomaan AWS-pilviympäristöön, kaikista luotettavin lähde on heidän omat dokumentaationsa. Dokumentaatiot ovat aineistoiltaan laajoja, joten aineistoa on saatavilla kattavasti. Myös Ticorporate-opintokokonaisuuden aikana luodut dokumentaatiot ja tiedostot Elli-sovellukseen liittyen olivat osana aineistoa, sillä jo olemassa olevia ohjelmakoodeja ja tiedostoja oli tarkoitus muokata ja parantaa.

Aineistot olivat olennainen osa kehitystyön etenemisessä ja ne vahvasti edesauttoivat kehitystyölle asetetun tavoitteen saavuttamisessa. Tavoitteen täyttämisessä tuli ottaa huomioon parhaat käytänteet tietotekniikkaan liittyen, esimerkiksi tietoturva ja kustannustehokkuus. Aineistojen analyysit tukivat näiden tavoitteiden saavuttamista.

5.2 Toteutuksen suunnittelu

Toimeksiannon tehtävä oli alusta alkaen selkeä, mutta siihen kuuluvia pienempiä osia piti rajata pois työn lopputuotoksesta. Suunnittelu oli aloitettu siitä näkökulmasta, että työn lopullinen tuotos olisi käytettävissä oleva sovellus Jyväskylän ammattikorkeakoulun pilviympäristössä. Suunnittelun alkuvaiheisiin kuului kokousluontoiset tapaamiset toimeksiantajan yhteyshenkilön kanssa, joissa keskusteltiin esimerkiksi teknisistä vaatimuksista ja pääsyoikeuksista käytettyihin resurssei-

hin liittyen. Yhteyshenkilön kautta toteutettiin pääsy Jyväskylän ammattikorkeakoulun Github-organisaatioon, jonne Elli-sovelluksen repositorio ja julkaisuputki luotiin. Myös AWS-resursseihin järjestettiin tili, jolla oli oikeudet luoda ja käyttää tarvittavia resursseja ja palveluita. Nämä tilit ja pääsyoikeudet olivat erittäin tärkeässä osassa suunnittelua ja kehitystyön aloittamisen kannalta välttämättömiä toimenpiteitä.

Toteutuksen suunnittelun olennainen osa oli tutkia ja analysoida AWS-dokumentaatioita liittyen tilien väliseen migraatioon. Kun tietoa oli analysoitu riittävästi, voitiin suorittaa testiajoja ympäristöön ja havainnoida sen käyttäytymisen perusteella, millaisia toimenpiteitä tarvitsee seuraavaksi tehdä.

Projektinhallinnan osalta sovellettiin toimintatutkimuksen syklistä prosessia (kts. Kuvio 1) ja spiraalimainen eteneminen tapahtui testiajoilla ja toiminnan vaiheiden dokumentoinnilla. Versionhallinta ohjelmakoodille tapahtui Github-repositorion avulla, jonka ansiosta koodimuutosten välillä oli helppo liikkua tarvittaessa. Elli-sovelluksen eri ympäristöillä on eriytetyt haarat Github-repositoriossa ja näihin yhdistetään vain toimivaksi havaittuja muutoksia. Muutoksia tehdessä paikallisessa kehitysympäristössä tarvittavasta haarasta tehtiin kopio, jotta kehityksen vaiheet pysyivät läpinäkyvinä ja hallittuina. Tätä kyseistä toimintamallia harjoitettiin jo Ticorporate-opintojaksolla Github-repositorion hallintaan Elli-sovelluksessa.

Ympäristön migraatio Jyväskylän ammattikorkeakoulun AWS-tilille vaati paljon erilaisia ympäristön valmisteluja, jotta migraatio voitiin suorittaa. Näiden vaiheiden voitiin ajatella kuuluvan suunnitteluvaiheeseen, sillä ne olivat välttämättömiä toimenpiteitä toiminnan testausajojen kannalta. Ensimmäinen päämäärä oli valmistella ympäristöt työskentelyä varten, jonka jälkeen voitiin suorittaa manuaalinen sovelluksen konfiguraatio kohdetilille. Tämä ensimmäinen manuaalinen konfiguraatio toimi testinä ja pohjana kaikelle sitä seuraavalle toiminnalle. Sen tarkoitus oli varmistaa, että sovellusmigraatio on mahdollista suorittaa, sekä kerätä informaatiota mahdollisista komplikaatioista ja muutoksia tarvitsevista osista.

5.3 Tietokantamuutokset

Alkuperäinen Elli-sovelluksen toteutus toimi yhden tietokannan kautta, eli tuotanto- ja kehitysympäristöt käyttivät kummatkin samaa tietokantaa. Tietokantana oli käytössä MongoDB Atlas, joka

mahdollistaisi saman tietokannan käyttämisen Elli-sovelluksen uudessa versiossa, mutta oli asianmukaisempaa luoda tietokannasta uusi versio. Tämä simuloi paremmin oikeaa työelämää, jossa suoritettaisiin sovelluksen migraatio toiseen kohteeseen, koska uuden ympäristön oli tarkoitus olla eriytetty vanhasta ympäristöstä ja toimia omana kokonaisuutenaan. Tämä mahdollisti sen, että vanha ympäristö pystyy toimimaan itsenäisesti vielä migraation suorittamisen jälkeen. Oikean työelämän käytänteitä tuki myös se, että tietokannan rakennetta ja sitä, kuinka sovellus kyseistä tietokantaa käytti, parannettiin toimivuudeltaan. Tarkoitus oli eriyttää ympäristöjen tietokannat siten, että tuotanto- ja kehitysympäristöt käyttävät eri tietokantoja. Eriyttäminen edesauttaa vikasietoisuutta ja tietoturvaa. Tietokantojen ollessa omina kokonaisuuksinaan, tietoihin pääsy on erilaista ja sovelluksen tietojen käyttäminen on selkeämpää. Kehitysympäristö pystyttiin tietokantojen eriyttämisellä irrottamaan paremmin tuotantoympäristöstä, eikä kehitysaikaiset toimenpiteet vaikuta tuotantoympäristön toimintaan. Tietokantojen eriyttäminen mukailee AWS:n parhaita käytänteitä esimerkiksi turvallisuus- ja luotettavuuspilarien osalta, eli tietojen suojaaminen ja toimintahäiriöiden ehkäiseminen ja niistä palautuminen on otettu huomioon sovelluksen toiminnassa.

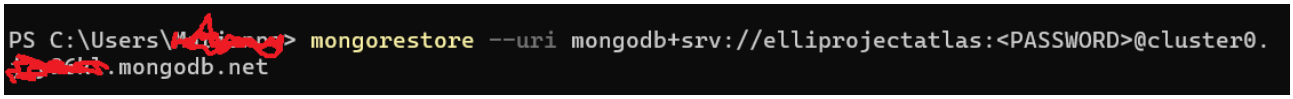
Vanhalta Atlas-tililtä pystyttiin tekemään tietokannasta kopio mongodump-komennolla, joka ajettiin tietokoneen komentokehotteessa. Komentoa varten tarvittiin MongoDB Atlas-osoite, käyttäjätunnus ja salasana. Syöttämällä komennon komentokehotteeseen, se latasi tietokannasta koneluettavan version siihen hakemistoon, jossa komento ajettiin. Tietokannassa on kokoelmia (engl. *Collection*) ja nämä kokoelmat sisältävät dokumentteja (engl. *Documents*), jotka ovat JSON-muotoisia erilaisia tietoja. MongoDB Atlas-palvelussa nämä kantakokonaisuudet sijaitsevat klusterilla (engl. *Cluster*).

Kuvio 7. Mongodump-komento tietokantakopion luomista varten

MongoDB Atlas-ympäristöön luotiin uusi käyttäjätili, jonne tietokannan kopio tuotiin. Elli-sovelluksen kehityksessä käytettiin tilin Free Tier-ominaisuuksia, jotka mahdollistivat vain yhden klusterin käytön. Tästä syystä tuotanto- ja kehitysympäristöjen tietokannat sijaitsevat samalla klusterilla,

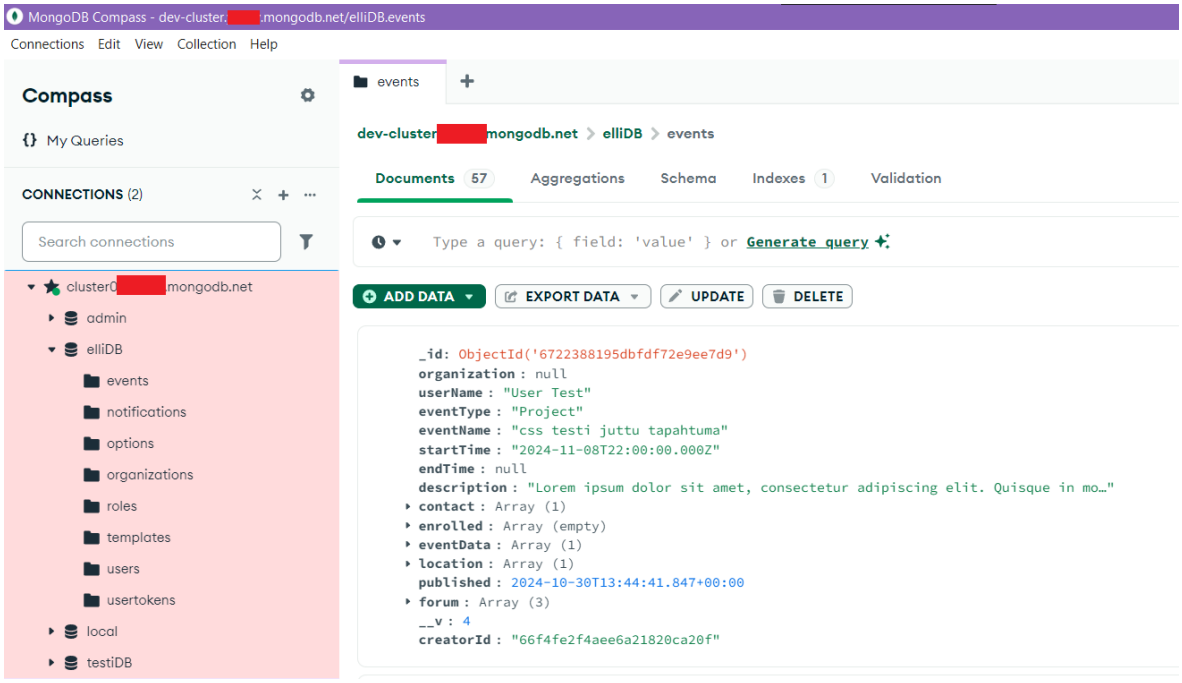
mutta ovat erillisinä tietokantoina, joilla on omat erilliset kokoelmansa. Ihanteellisin toteutustapa olisi sijoittaa tietokannat erillisille klustereille, jotta eriyttämisen kaikki edut olisivat käytössä täydellisesti.

Uudelle käyttäjätilille voitiin tuoda tietokannan kopio käyttämällä mongorestore-komentoa. Samalla tavalla kuin mongodump-komento, se ajettiin tietokoneen komentokehotteessa ja siihen tarvitaan MongoDB Atlas-osoite, käyttäjätunnus ja salasana. Tämän komennon ajamiseen käytettiin uuden käyttäjätunnuksen ja tietokannan tietoja.



Kuvio 8. Mongorestore-komento tietokantakopion tuomiseksi uudelle tilille

Nämä komennot ajamalla onnistuneesti siirrettiin tietokanta vanhalta tililtä uudelle tilille. Tietokanta oli kuitenkin edelleen samanlainen, kuin alkuperäisessä versiossa. Eriytettyjen tietokantojen luomiseksi tarvittiin kaksi eri kantaa ja tässä vaiheessa sen luominen oli ajankohtaista. Toisen tietokannan luominen oli vain kopio jo olemassa olevasta tietokannasta kyseisen saman klusterin sisällä. Toistaiseksi kummankin ympäristön tietokannat voivat sisältää samoja tietoja, koska ne eivät ole vielä käytössä. Käyttöönoton jälkeen tietokantojen tiedot tulevat kehitystoimenpiteiden jälkeen olemaan erilaisia. Tuotantoympäristölle määriteltiin tietokanta nimellä elliDB ja kehitysympäristölle määriteltiin nimeksi testiDB.



Kuvio 9. Tietokannat MongoDB Compass-ohjelmassa

Uuden tietokannan tiedot liittyivät olennaisesti Elli-sovelluksen ohjelmakoodiin ja tehtyjen muutoksien vuoksi ohjelmakoodia tuli muuttaa jonkin verran, jotta eriytetyt tietokantaympäristöt voitiin ottaa käyttöön. Koska toteutuksessa käytettiin yhtä klusteria, jossa tuotanto- ja kehitysympäristöjen tietokannat sijaitsivat, MongoDB Atlas-osoitteita ei tarvittu useampia. Sovellus käyttää .env-tiedostoa ympäristömuuttujien käsittelemiseen, joten kyseiseen tiedostoon tuli päivittää tietokannan uuden tiedot. Tämä tiedosto sijaitsee paikallisessa kehityksessä sovelluksen Server-hakemiston juuressa ja pilviympäristöä käyttäessä ne tallennetaan Github Secrets-muuttujiksi. Github-repositoriota käyttäessä .env-tiedostoa ei käsitellä, koska sen syöttäminen repositorioon sellaisenaan olisi tietoturvauhka tiedoston sisällön näkyessä salaamattomana. Github Secrets-ominaisuuksia käyttämällä .env-tiedoston muuttujat ovat salattuja, eikä niitä paljasteta missään vaiheessa julkaisuprosessia. Elli-sovelluksen julkaisuputki osaa käyttää Github Secrets-muuttujia .env-tiedoston luomiseksi myöhemmissä julkaisun vaiheissa, tehden siitä turvallista.

Sovelluksen tietokantayhteydestä vastuussa olevaan ohjelmakoodiin tehtiin muutoksia ympäristömuuttujan käsittelyyn, jotta eriytetyt tietokannat voitiin ottaa käyttöön. NODE_ENV-muuttujan avulla voidaan asettaa sovellus eri tiloihin, joiden mukaan ohjelmakoodi käyttää eri arvoja. Muutoksien ansiosta tietokantayhteyden ottaminen Elli-sovelluksessa tapahtuu NODE_ENV-muuttujan

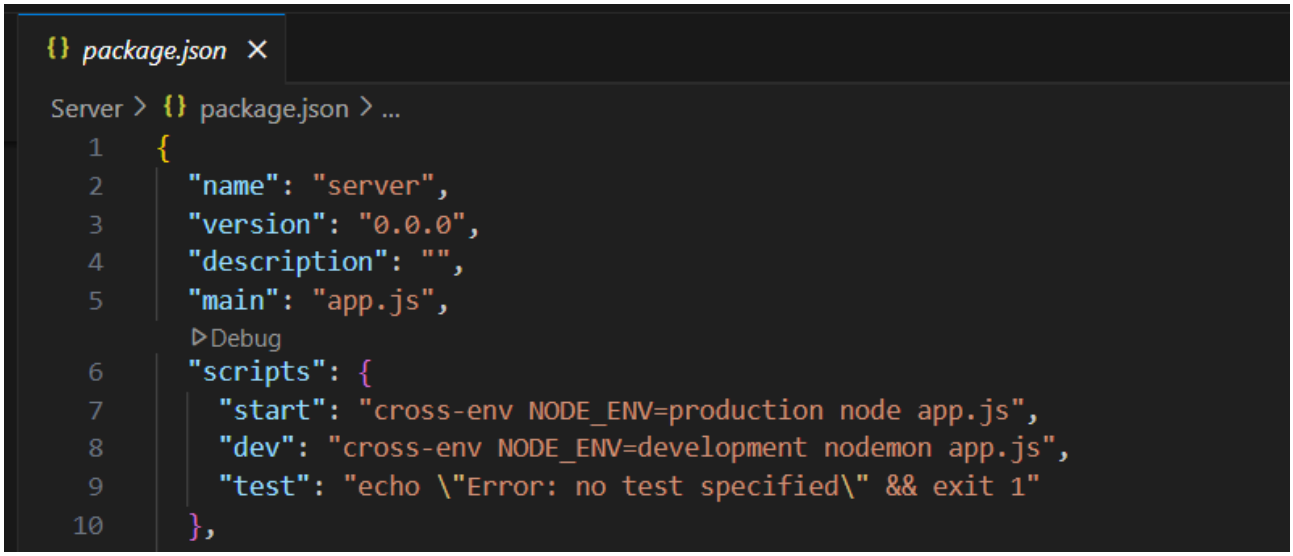
avulla. Muuttujia on kaksi: production ja development. Production-muuttujaa käyttämällä otetaan yhteys elliDB-tietokantaan ja development-muuttujaa käyttämällä sovellus saa yhteyden testiDB-tietokantaan.

Server > JS dbconnection.js > ...

```
1  const mongoose = require('mongoose');
2  if (process.env.NODE_ENV !== 'production') {
3    require('dotenv').config();
4  }
5
6  /************************YHTEYS KANTAAN*****************************/
7
8  // Define the database name based on the environment
9  let dbName;
10 if (process.env.NODE_ENV === 'production') {
11   dbName = 'elliDB';
12 } else {
13   dbName = 'testiDB';
14 }
15
16 mongoose
17   .connect(process.env.MONGODB_URL, {
18     dbName: dbName,
19   })
20   .then(() => {
21     console.log(`Connected to MongoDB database: ${dbName}`);
22   })
23   .catch((err) => {
24     console.error('MongoDB connection error:', err);
25   });
26
```

Kuvio 10. Ohjelmakoodin muutokset dbconnection.js-tiedostossa tietokantojen eriyttämiseksi ympäristöille

Ympäristömuuttujia voidaan kutsua useissa eri paikoissa ohjelmakoodissa ja niiden käynnistystoimenpiteet ovat määritelty sovelluksen palvelinpuolen package.json-tiedoston scripts-osiossa.



Kuvio 11. Ympäristömuuttujien määrittelyt Server-hakemiston package.json-tiedoston scripts-osiossa

Käynnistystoimenpiteillä tarkoitetaan sitä, että eri komentoja antamalla sovellus käynnistyy eri tavoilla. Antamalla komennon ”*npm start*” sovelluksen Server-hakemistossa se käynnistyy tuotantotilassa ja saa production-ympäristömuuttujaan liitettyjä ominaisuuksia. Kehitysympäristön development-ympäristömuuttujaa voidaan kutsua komennolla ”*npm run dev*”.

5.4 Github-repositorio

Ticorporate-opintojakson aikana luotu ja käytetty repositorio oli sellaisenaan kopioitavissa uuteen repositorioon Jyväskylän ammattikorkeakoulun organisaation puolelle. Repositorion kopioimista Githubissa kutsutaan englanninkielisellä termillä ”*fork*”. Tämä toiminto mahdollistaa repositorion kopioinnin uuteen repositorioon niin, että uudessa tehdyt toimenpiteet eivät vaikuta alkuperäiseen tuotokseen. Kun repositorio oli forkattu uuteen Git-ympäristöön, siitä voitiin poistaa kaikki ylimääräiset haarat, joita ei tarvita migraatioprojektissa. Projektinhallinnan kannalta suunnitelma oli se, että alkuperäiseen main-haaraan ei tehdä muutoksia, vaan se jää repositorioon eri nimellä ”pohjahaaraksi”. Tämän haaran lisäksi haaroja tulee olemaan kaksi: production ja development. Production-haaran tarkoitus on toimia tuotantoympäristön julkaisussa, kun taas development-haaran toiminta hallinnoi kehitysympäristöä.

Uuden Git-ympäristön etärepositorion kloonaaminen paikallisesti kehitettäväksi tapahtui fork-toimenpiteen ja ylimääräisten haarojen poistamisen jälkeen. Paikallisessa kehitysympäristössä forkatun repositorion main-haarasta tehtiin development-niminen haarakopio. Paikalliseen haaraan tuli tässä vaiheessa lisätä tai päivittää aikaisemmassa kappaleessa mainittu .env-tiedosto, sekä lisätä nämä kyseisen tiedoston sisältämät ympäristömuuttujat myös Github repositorion Secrets-osuuteen, jotta Github Actions-alusta ja julkaisuputken workflow-tiedostot toimivat pilviympäristöön julkaisemisessa.

5.5 AWS-ympäristön pystyttäminen

Elastic Beanstalk-ympäristön konfiguraatio

AWS Elastic Beanstalk-ympäristön manuaalinen pystyttäminen edellytti eri AWS-resurssien valmistelemista. Esimerkiksi IAM-palvelussa luotiin Elastic Beanstalk-palvelua varten rooli, jonka ansiosta se pystyy luomaan ja hallinnoimaan tarvittavia resursseja. Myös Github Actions-alustaa ja sen prosesseja varten oli luotiin käyttäjä, jolla on tarvittavat oikeudet suorittaa AWS-toimintoja julkaisuputkeen liittyen. Tämän luodun käyttäjän pääsyavaintiedot (engl. *Access key*) välitettiin Github-repositoriolle käyttämällä Github Secrets-ominaisuutta.

Seuraavaksi Elastic Beanstalk-ympäristön toimintaa varten konfiguroitiin VPC-palvelun asetuksia. Jyväskylän ammattikorkeakoulun AWS-ympäristössä ei ollut oletus-VPC-aluetta, jolloin sellaisen luominen oli tarpeellista. Oletus-VPC-alue luo automaattisesti kolme aliverkkoa, kaksi saatavuusaluetta ja internet gateway-komponentin. Elli-sovelluksessa käytetään kahta aliverkkoa, joten ylimääräinen aliverkko poistettiin.

Näiden resurssien luomisen jälkeen pystyttiin siirtymään Elastic Beanstalk-ympäristön konfiguraatioasetuksiin. Näissä asetuksissa valittiin ominaisuuksia pystytettävälle ympäristölle, esimerkiksi liittyen nimeämiseen, palvelujen käyttöoikeuksiin, instanssien ominaisuuksiin, korkean saatavuuden ominaisuuksiin ja sovelluksen teknisen toiminnan ominaisuuksiin. Konfiguraatioasetusten lopussa oli mahdollista valita Elastic Beanstalk-palvelun oma mallipohjasovellus, joka pystytetään ensimmäisessä ajossa.

Ohjelma- ja julkaisuputkitiedostojen muutokset

Aikaisemmassa kappaleessa mainittujen toimenpiteiden jälkeen ympäristö oli konfiguroitu yhteensopivaksi Elli-sovellukselle. Elastic Beanstalk-palvelu oli konfiguroitu niin, että se pystyttää oman mallisovelluksensa ympäristöön ja se tässä vaiheessa se korvattiin Elli-sovelluksen ohjelmatiedostoilla. Ympäristö oli valmis vastaanottamaan sovelluksen, kun Github-repositorion development-haara laukaisee julkaisuputken toimenpiteet. Näihin julkaisuputken tietoihin oli kuitenkin tarve tehdä manuaalisesti muutoksia ennen julkaisuputken ajamista, jotta ne vastaavat uutta konfiguroitua Elastic Beanstalk-ympäristöä. Näitä muutoksia olivat seuraavat toimenpiteet:

- Elli-sovelluksen ohjelmakoodissa sijaitsevat kaksi eri tiedostoa, joihin vaihdettiin konfiguraatiossa määritelty uusi URL-osoite, eli domain-nimi.
- e2e-test.yml-tiedostoon vaihdettiin procfile-tiedoston tekstiin kehitysympäristön ympäristömuuttuja, sekä palvelimen käynnistyskomento, joka käynnistää palvelimen kehitystilassa.
- cicd.yml-julkaisuputkitiedostoon korvattiin haaran nimi, repositorion polku, ympäristön nimi ja ympäristömuuttujan tieto.

Client > src > app > shared > environments > TS urls.ts > ...

```
1   // the production URL
2   const productionUrl = 'https://ellidevelopment.eu-north-1.elasticbeanstalk.com';
3
4   // the development URL
5   const developmentUrl = 'http://localhost:8080';
6
7   // Get the current URL's origin | this URL's shows me the origin of the application on the browser
8   const currentUrl = window.location.origin;
9
10  // Check whether we're in production mode by checking if the current URL matches the production URL.
11  // If it does, we're in production mode. If it doesn't, we're in development mode.
12  const production = currentUrl === productionUrl;
13
14  // Determine the base URL based on whether we're in production mode
15  // If we're in production mode, use the production URL. If we're in development mode, use the development URL
16  const baseUrl = production ? productionUrl : developmentUrl;
17
18  // The URLs for use in the application
19  export const urls = {
20    production: production,
21    authServiceUrl: `${baseUrl}/auth/`,
22    userServiceUrl: `${baseUrl}/user/`,
23    eventServiceUrl: `${baseUrl}/event/`,
24    optionsServiceUrl: `${baseUrl}/option/`,
25    notifyServiceUrl: `${baseUrl}/notify/`,
26  };
```

Kuvio 12. URL-osoitteen vaihtaminen Client-hakemiston urls.ts-tiedostossa

Kun julkaisuputken workflow-tiedostot olivat päivitetty vastaamaan uuden ympäristön tietoja, ne voitiin laukaista ajettavaksi push-komennolla paikallisessa repositoriossa. Julkaisuputki ensin ajoi Elli-sovelluksen Githubin virtuaalikoneessa samaan tapaan, kuin kehittäjä ajaisi sitä paikallisessa kehityksessä omalla tietokoneellaan. Sovellukselle suoritettiin end-to-end-testit ja kun ne läpäistiin onnistuneesti, julkaisuputki siirtyi julkaisuosuuteen.

Elastic Beanstalk-ympäristön pystytyksen jälkeen sille luotiin ja konfiguroitiin SSL-sertifikaatti. Tämä SSL-sertifikaatti liitettiin ympäristön kuormanjakajan asetuksiin, jotta sovellus pystyy käyttämään salattua https-yhteyttä.

Tuotantoympäristön luominen kehitysympäristön rinnalle

Elastic Beanstalk-palvelussa kehitysympäristön konfiguraatio voitiin tallentaa ja käyttää kyseistä tallennettua konfiguraatiota uuden ympäristön pystyttämiseen. Konfiguraatio tallentuu S3-bucketiin YAML-muotoisena mallipohjatiedostona, josta Elastic Beanstalk-palvelu pystyy käyttämään sitä. Kun tallennetusta konfiguraatiosta lähdetään laukaisemaan uutta ympäristöä, Elastic Beanstalk-palvelu aloittaa saman tyyppisen konfiguraatiovalinnan, kuin uutta ympäristöä luodessa. Tallennettu konfiguraatio pyytää syöttämään joitakin tietoja itse, kun ympäristöä pystytetään. Nämä tiedot ovat esimerkiksi ympäristön nimi- ja domaintiedot, roolivalinnat, VPC-alue ja instanssikohtaiset asetukset. Valintojen itse syöttäminen tuo tallennettujen konfiguraatioiden käyttämiseen joustavuutta ja mahdollistaa ympäristön mukautumisen muutoksien varalta. Toteutusta tehdessä todettiin, että tallennettujen konfiguraatioiden hyödyntäminen mahdollistaa uuden ympäristön pystyttämisen vain silloin, kun Elastic Beanstalk-palvelussa on jo olemassa yksi ympäristö. Se soveltuu siis juuri ympäristön luomiseen toisen rinnalle, mutta ei uuden ensimmäisen ympäristön luomiseen.

Kun tuotantoympäristö pystytettiin käyttäen tallennettua konfiguraatiota, sille voitiin tehdä samat muutostoimenpiteet ohjelma- ja julkaisuputkitiedostoihin, kuin kehitysympäristön pystyttämisessä. Ennen toimenpiteitä repositorion paikallisessa haarassa development-haarasta tehtiin kopio, joka nimettiin production-nimiseksi. Ohjelma- ja julkaisuputkeen tehtävät muutokset suoritettiin tuotantoympäristöä vastaavilla tiedoilla.

Havaitut muutokset ja komplikaatiot ympäristön pystyttämisessä

Elli-sovelluksen manuaalinen konfiguraatio AWS-ympäristöön toi ilmi erilaisia toimenpiteitä, jotka vaativat mukauttamista onnistuneen migraation suorittamiseksi. Nämä toimenpiteet tulivat ilmi suorittamalla useita eri konfiguraatiotestejä ja analysoimalla niiden tuloksia ja komplikaatioita.

Ensimmäinen eroavaisuus alkuperäiseen Elli-sovelluksen toimintaan ilmeni jo alussa, kun tietokannat uudistettiin eriytetyiksi ympäristöjen perusteella. Tämä uudistus vaati sovelluksen ohjelmakoodin mukauttamisen ja nämä mukautukset vaativat julkaisuputken muuttamisen niin, että se käsittelee testi- ja julkaisutoimenpiteissä ympäristöä vastaavaa tietokantaa.

AWS-ympäristöissä ilmeni pieniä eroavaisuuksia, jotka vaikuttivat ympäristön pystyttämiseen. Ti-corporate-opintojakson aikana käytössä olleeseen AWS-tiliin ei ollut tarvetta erikseen luoda oletus-VPC-aluetta, mutta Jyväskylän ammattikorkeakoulun tiliä käyttäessä sen luominen oli tärkeä askel. Sovelluksen ajaminen omassa VPC-alueessaan on tietoturvallisesti viisas ajatus ja ilman oman alueen luomista Elastic Beanstalk-palvelun automaattiset osat eivät toimineet oikein, johtaen ympäristöjen virheelliseen pystytykseen. Ympäristöä konfiguroitaessa Elastic Beanstalk-palvelun VPC-osuudessa todetaan, että jos VPC-aluetta ei määritellä, luodaan ympäristö ja sen osat oletus-VPC-alueelle. Jos oletus-VPC-aluetta ei ole, konfiguraatiossa tapahtuu virhe.

Elastic Beanstalk-ympäristöä konfiguroidessa ensimmäistä kertaa roolien valinta aiheutti ongelmia, sillä sen asetusvalikko ei ollut tarpeeksi selkeästi ilmaistu. Elli-sovelluksen dokumentaatioissa kerrottiin palveluroolista, joka asetetaan konfiguraatiossa ympäristön käytettäväksi. Tätä roolia ei kuitenkaan ole olemassa, ennen kuin Elastic Beanstalk-palvelun annetaan luoda se ensimmäisellä konfiguraatiokerralla. Tämän jälkeen palvelurooli on valittavissa uusien ympäristöjen luomiseen, mutta on siis tärkeää ymmärtää luoda kyseinen rooli ensimmäisen ympäristön kohdalla. Tämän roolin puuttuminen tai väärin konfiguroiminen aiheuttaa sen, että Elastic Beanstalk-palvelu ei pysty toimimaan oikein.

Elli-sovelluksen ympäristö oli suunniteltu niin, että tarpeen mukaan se voidaan asettaa korkeasti saavutettavaksi. Infrastruktuurikaaviossa oli suunniteltu toiminta niin, että tuotanto- ja kehitysympäristöt ovat eriytettyjä omille saatavuusalueilleen. Näiden saatavuusalueiden sisällä on aliverkko,

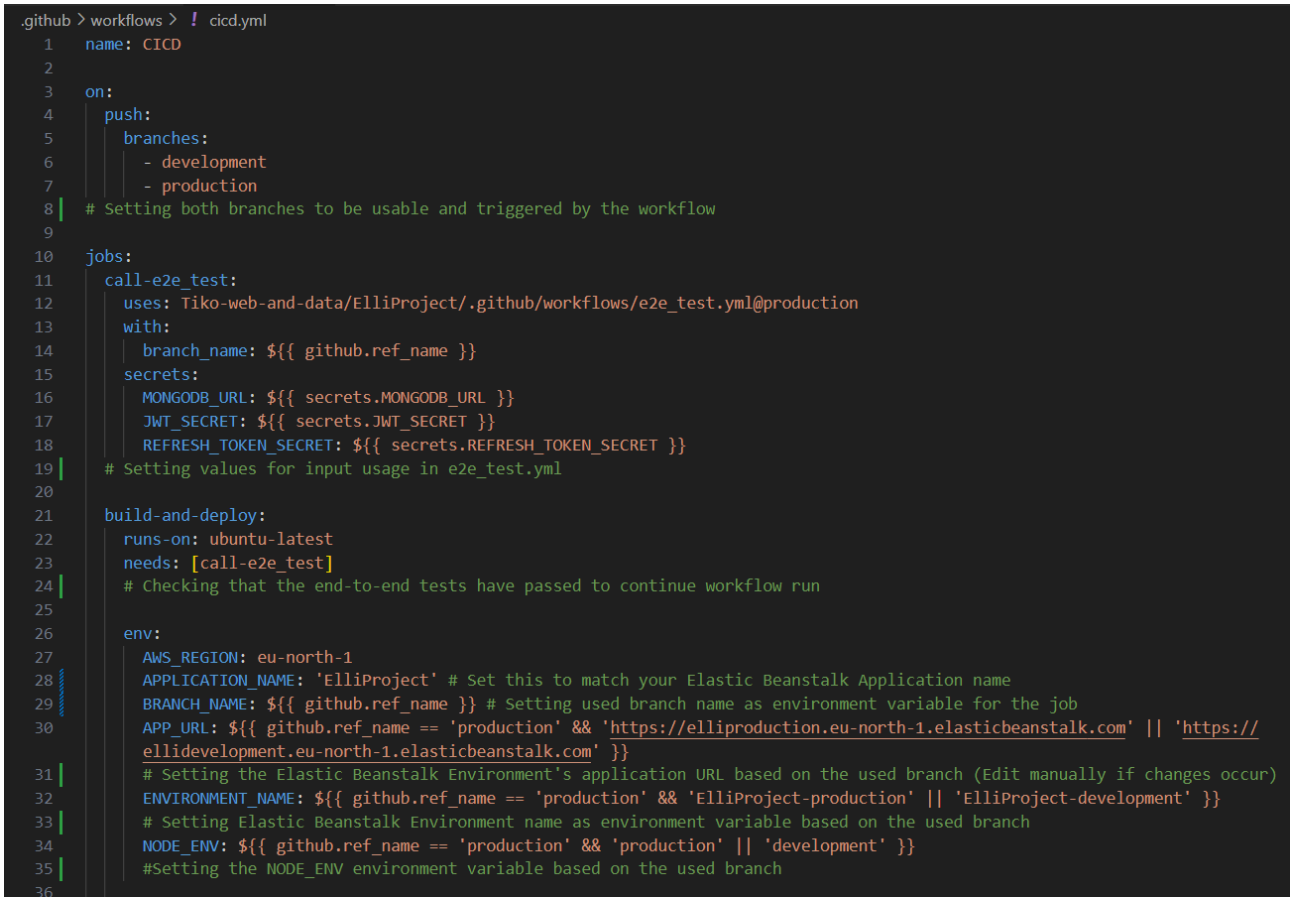
jonne instanssit pystytetään Elastic Beanstalk-palvelun toimesta. Tästä syystä ympäristön konfiguraatiovalinnassa valittiin räätälöity ympäristö, joka mahdollistaa kuormanjakajien asettamisen toimivaksi. Kuormanjakajien saatavuusalueiden asetukset vaativat konfiguraation aikana sen, että vähintään kaksi aliverkkoa on valittu. Tämä kyseinen valinta oli ilmaistu konfiguraatiokonsolissa erittäin epäselvästi, johtaen virheisiin, kun valinnaksi asetettiin vain yksi aliverkko.

5.6 Julkaisuputken kehittäminen

Julkaisuputkitiedostojen kehitys automatisoidummaksi

Elli-sovellukseen oli luotu Github Actions-alustalla toimiva julkaisuputki Ticorporate-opintojakson aikana ja silloin huomattiin, että sitä olisi syytä kehittää. Julkaisuputken tiedostot sijaitsivat omissa haaroissaan ja olivat erillisiä omia tiedostojaan, vaikka ne tekivät samat toimenpiteet julkaistavalle sovellusversiolle. Myös ympäristöjen URL-osoitteet olivat vaihdettava manuaalisesti sovelluksen ohjelmakoodin tiedostoista, riippuen oliko käsiteltävä haara ja julkaisu tuotanto- vai kehitysympäristö. Git-käytänteiden vuoksi, jossa ohjelmakoodia ensin kehitettiin omassa haarassaan ja sen valmistuttua yhdistettiin testaushaararasta kehityshaaraan ja lopuksi tuotantohaaraan, seurasi joskus merge-toiminnon yhteydessä koodisekaannuksia. Nämä ongelmat usein havaittiin ajoissa julkaisuputken testausosion ansiosta, mutta tällaisia ongelmia ei tulisi tapahtua alun perinkään.

Elli-sovelluksen migraatioprojektin alussa sen käyttämää tietokantaratkaisua paranneltiin. Ohjelmakoodin vaadittavat muutokset tietokantojen eriyttämistä varten suoritettiin ensimmäisen testausajon toteutukseen. Tietokantojen käsittely perustuu NODE_ENV-ympäristömuuttujan käsittelyyn, joten ideana oli lisätä julkaisuputkitiedostoihin tämä sama logiikka. Github-repositorion haarat ovat tästä syystä nimetty termeillä ”*production*” ja ”*development*”, koska näitä termejä käyttämällä haaran nimeä pystyi helposti hyödyntämään ympäristömuuttujissa. Ympäristömuuttujiksi lisättiin julkaisuputkelle haaran nimi, sovelluksen URL-osoite, ympäristön nimi, sekä NODE_ENV-ympäristömuuttuja. Haaran nimen ympäristömuuttujan arvo vaihtuu käytettävän haaran nimen perusteella ja sovelluksen URL-osoitteen, ympäristön nimen sekä NODE_ENV-ympäristömuuttujan arvo perustuu tämän kyseisen haaran ympäristömuuttujan arvoon. Kehityksen ansiosta julkaisuputkitiedostoja ei tarvitse enää manuaalisesti muuttaa ympäristöjen tietoja vastaamaan, kun käsitellään olemassa olevia ympäristöjä.



```
.github > workflows > ! cicd.yml
name: CICD

on:
  push:
    branches:
      - development
      - production
# Setting both branches to be usable and triggered by the workflow

jobs:
  call-e2e_test:
    uses: Tiko-web-and-data/ElliProject/.github/workflows/e2e_test.yml@production
    with:
      branch_name: ${{ github.ref_name }}
    secrets:
      MONGODB_URL: ${{ secrets.MONGODB_URL }}
      JWT_SECRET: ${{ secrets.JWT_SECRET }}
      REFRESH_TOKEN_SECRET: ${{ secrets.REFRESH_TOKEN_SECRET }}
  # Setting values for input usage in e2e_test.yml

  build-and-deploy:
    runs-on: ubuntu-latest
    needs: [call-e2e_test]
    # Checking that the end-to-end tests have passed to continue workflow run

    env:
      AWS_REGION: eu-north-1
      APPLICATION_NAME: 'ElliProject' # Set this to match your Elastic Beanstalk Application name
      BRANCH_NAME: ${{ github.ref_name }} # Setting used branch name as environment variable for the job
      APP_URL: ${{ github.ref_name == 'production' && 'https://elliproduction.eu-north-1.elasticbeanstalk.com' || 'https://
      ellidevelopment.eu-north-1.elasticbeanstalk.com' }}
      # Setting the Elastic Beanstalk Environment's application URL based on the used branch (Edit manually if changes occur)
      ENVIRONMENT_NAME: ${{ github.ref_name == 'production' && 'ElliProject-production' || 'ElliProject-development' }}
      # Setting Elastic Beanstalk Environment name as environment variable based on the used branch
      NODE_ENV: ${{ github.ref_name == 'production' && 'production' || 'development' }}
      #Setting the NODE_ENV environment variable based on the used branch
```

Kuvio 13. Haaran perusteella toimivat ympäristömuuttujat julkaisuputkitiedostossa

NODE_ENV-ympäristömuuttujan tarkoitus pysyi samana, eli se on vastuussa palvelinpuolen toiminnasta ja tietokantayhteyden valitsemisesta, mutta sen käyttöä automatisoitiin. Ympäristön nimen ympäristömuuttuja (ENVIRONMENT_NAME) varmistaa sen, että ympäristön julkaisu Elastic Beanstalk-palveluun tapahtuu tarkoituksenmukaiseen pilviympäristöön. APP_URL-ympäristömuuttujalla käsitellään sovelluksen URL-osoitetta. Aikaisemmin nämä osoitteet vaihdettiin manuaalisesti sovelluksen ohjelmakoodiin useaan tiedostoon julkaisun yhteydessä, mutta kehityksen ansiosta julkaisuputki päivittää nämä tiedot ohjelmakoodiin automaattisesti sovelluksen julkaisun valmistelujen yhteydessä. URL-osoitteen muuttuessa, jatkossa se tarvitsee päivittää vain julkaisuputken tiedoston tietoihin yhteen paikkaan. Kuvio 14 havainnollistaa muutokset, joiden alkuperäinen tila esitettiin aikaisemmassa luvussa (kts. Kuvio 12).

```
57
58      # Create Procfile in project's root directory: Uses the NODE_ENV variable from branch name
59      # This is needed for web deployments
60      - name: Create Procfile in project's root directory
61        run: |
62          echo "web: NODE_ENV=$NODE_ENV node Server/app.js" > Procfile
63
64          echo "Procfile contents:"
65          cat Procfile
66          cp Server/package.json .
67
68      # Adding the EB Environment URL for Server/app.js file (CORS)
69      - name: Update Allowed Origins in app.js (Server)
70        run: |
71          echo "Replacing applicationURLplaceholder with: $APP_URL"
72          sed -i "s|'applicationURLplaceholder'|'$APP_URL'|g" Server/app.js
73          awk '/allowedOrigins = \[/,/];/' Server/app.js
74
75      - name: Install Angular CLI and Dependencies for Client
76        working-directory: Client
77        run: npm install -g @angular/cli@latest && npm install @angular/cli@latest --save-dev && npm install --include=dev
78
79      # Adding the EB Environment URL for Client directorys urls.ts file (baseUrl)
80      - name: Update URL in code urls.ts (Client)
81        run: |
82          sed -i "s|const productionUrl = '.*'|const productionUrl = '${APP_URL}'|g" Client/src/app/shared/environments/urls.ts
83          grep "const productionUrl" Client/src/app/shared/environments/urls.ts
84
```

Kuvio 14. Ympäristömuuttujien hyödyntäminen julkaisuputkessa

Haasteet julkaisuputken automaatiossa

Ratkaisuja julkaisuputken automaatioon on mahdollista toteuttaa monella eri tavalla. Migraatio-projektin alussa alettiin tietokantojen eriyttämistä ratkaisemaan NODE_ENV-ympäristömuuttujan avulla, joten tästä syystä muutkin automaation osat päätettiin kehittää noudattaen samaa logiikkaa. Kehitystoimenpiteiden syklit aloitettiin ensin kehitysympäristön tiedostoilla ja kun kehitetyt kohteet todettiin toimiviksi, niitä alettiin soveltamaan tuotantoympäristön tiedostoihin. Tässä vaiheessa ilmeni ongelmia liittyen NODE_ENV-ympäristömuuttujan käyttöön tuotantoympäristössä. Julkaisuputken testiosuudessa tämä ympäristömuuttuja asetetaan koko job-vaiheen käytettäväksi, jolloin job-vaiheen kaikki step-toimenpiteet käyttävät sitä. Kun workflow-prosessi eteni käyttöliittymäpuolen riippuvuuksien asennusvaiheeseen, production-ympäristömuuttujan käyttö aiheutti sen, että kehitysaikaiset riippuvuudet eivät asentuneet ja prosessit keskeytyivät. Angular-viitekehykseen kuuluva cli-työkalu kuuluu kehitysaikaisiin riippuvuuksiin ja kun se ei asennu, Angular-viitekehys ei pysty ajamaan komentoja. Ongelma korjattiin asettamalla julkaisuputken tiedostossa näihin kahteen step-toimenpiteeseen erikseen ympäristömuuttujan arvoksi ”*development*”.

Cloudformation-mallipohjat

AWS-ympäristön ja Elli-sovelluksen pystyttämiseen olisi mahdollista luoda Cloudformation-mallipohjatiedostot, joiden avulla ympäristön ja sovelluksen pystyttäminen tai monistaminen olisi nopeaa ja helppoa. Cloudformation-mallipohjissa määriteltäisiin samoja palveluita ja resursseja, joita toteutuksen vaiheissa käsiteltiin, vähentäen manuaalisen työn määrää. Opinnäytetyön aikaresurssien ja aiheen rajauksen vuoksi toteutettaviksi mallipohjan osiksi muodostui lopulta VPC-alueen ja tarvittavien IAM-resurssien luominen.

Elli-sovelluksen rakenteeseen ja julkaisutoimenpiteiden toimintatapaan liittyen oli kuitenkin sellaisia osia, jotka osoittautuivat hankalaksi toteuttaa mallipohjaa hyödyntämällä. Elastic Beanstalk-ympäristön pystyttäminen uudelle AWS-tilille mallipohjan kautta vaatisi sen, että ympäristö pystytetään ensin mallisovelluksella toimivaksi, jonka jälkeen olisi pakollista suorittaa manuaalista työtä esimerkiksi seuraavien toimenpiteiden osalta:

- IAM-käyttäjälle pääsyavaimen luominen ja sen vieminen Github Secrets-osioon
- .env-tiedoston ympäristömuuttujien vieminen Github Secrets-osioon
- EC2 avain-arvoparin luominen ja liittäminen ympäristön käytettäväksi
- SSL-sertifikaatin luominen, lataaminen ja vieminen ACM-palveluun
- SSL-sertifikaatin konfigurointi kuormanjakajalle ja http-uudelleenreititys
- Repositorion lataaminen paikallista kehitystä varten
- Julkaisuputkitiedoston päivittäminen ympäristökohtaisten tietojen osalta
- .env-tiedoston lisääminen paikalliseen repositorioon

Kaikkiin näihin toimenpiteisiin voitaisiin todennäköisesti löytää ratkaisu, esimerkiksi luomalla tietokoneella ajettavia script-tiedostoja tai hyödyntämällä Github Actions-alustaa ja erilaisia workflow-prosesseja. Näiden toimenpiteiden aikaan saaminen vaatisi enemmän aikaresursseja ja ne eivät olleet osa aiherajausta. Tästä syystä kyseinen aihealue sopii hyvin jatkokehityksen osuuteen. Toteutuneet mallipohjan osat VPC-alueen ja IAM-resurssien osalta kehittivät pystytysprosessia eteenpäin alkuperäiseen prosessiin verrattuna, vaikka ne eivät tee siitä täysin automatisoitua.

6 Tulokset

Opinnäytetyön kehittämistehtävän tavoitteena oli ensisijaisesti Elli-sovelluksen onnistunut migraatio Jyväskylän ammattikorkeakoulun pilviympäristöön. Toissijaisena tavoitteena oli kehittää DevOps-toimintamalliin liittyviä kehitystoimenpiteiden prosesseja, sekä tuottaa ohjeopas migraatioprosessin toteuttamiselle. Kehittämistehtävän tuloksena Elli-sovelluksen migraatio suoritettiin onnistuneesti ja Elli-sovellus on käytettävissä Jyväskylän ammattikorkeakoulun AWS-pilviympäristössä (kts. Liite 1, sivu 3). Migraatioprosessin toteutuksen yhteydessä Elli-sovelluksen DevOps-toimintamalliin liittyviä kehitystoimenpiteiden prosesseja edistettiin automatisoidummiksi ja mukailemaan paremmin AWS:n parhaita käytänteitä (kts. Kuvio 13). Sovelluksen migraation lisäksi toteutettiin ohjeopas, jossa käsitellään pystytettävän AWS-ympäristön ominaisuuksia ja kuinka Elastic Beanstalk-sovelluksen julkaisu automatisoidaan Github Actions-alustan avulla (kts. Liite 1).

Elli-sovellus Jyväskylän ammattikorkeakoulun pilviympäristössä

Elli-sovellus on tarkoitettu käyttöön Jyväskylän ammattikorkeakoulun ja Elisa Oyj:n henkilökunnalle, jotka järjestävät yhteistyötoimintaa yritysten ja oppilaitosten välillä. Sovelluksen onnistunut migraatio Jyväskylän ammattikorkeakoulun pilviympäristöön mahdollistaa sen käyttöönoton yhteistöiden hallintaan, seurantaan ja tiedotukseen. Yhteistyömahdollisuuksien järjestäminen yhteistyökumppani Elisa Oyj:n kanssa edistää oppilaiden valmiuksia työelämään, sekä vahvistaa ammatillista pätevyyttä. Myös Jyväskylän ammattikorkeakoulu ja Elisa Oyj hyötyvät yhteistyötoiminnan hallinnan, seurannan ja tiedotuksen kehityksestä, sillä yhteistyötoiminta useasti liittyy esimerkiksi erilaisiin toimeksiantoihin, joilla voidaan tuottaa arvoa yrityksille.

Elli-sovelluksen julkaisuputken automatisointi

Sovelluksen julkaisuputki oli luotu Ticorporate-opintojakson aikana ja sille oli todettu kehittämistarve jo ennen migraatioprosessin toimeksiantoa. Migraatioprosessin kehitysvaiheiden tuloksena julkaisuputken osia onnistuttiin kehittämään enemmän automatisoidummaksi ja mukailemaan paremmin AWS:n parhaita käytänteitä. Kuvio 13 ja Kuvio 14 osoittavat kuinka ympäristömuuttujien avulla julkaisutoimenpiteet onnistuttiin edistämään niin, että manuaalinen työ on vähentynyt ohjelmakoodin muutoksien osalta. Muutoksien kohteena olleet julkaisuputkitiedostot (e2e_test.yml, cicd.yml) ovat katseltavissa Elli-sovelluksen Github-repositoriossa ja ne ovat kommentoitu hyvien

käytänteiden mukaisesti. Kommentointikäytänne mahdollistaa sen, että tiedostojen sisällöt ymmärretään jatkokehitystä suorittavien osalta nopeammin ja helpommin. Julkaisuputkitiedostojen kehitetyt versiot otettiin käyttöön toteutettuun ohjeoppaaseen. Julkaisuputken automatisointi edistää Elli-sovelluksen DevOps-toimintamalliin kuuluvia kehitystoimenpiteiden prosesseja, sillä DevOps-toimintamallin tarkoitus, kuten tietoperustassa kerrottiin, on lisätä nopeutta ja laatua ohjelmistoversioiden kehittämisen elinkaaren aikana.

Cloudformation-mallipohjien avulla Elli-sovelluksen ympäristön pystyttäminen on nopeampaa, kuin ilman niitä. Mallipohjien avulla pystytään tällä hetkellä luomaan VPC-alue AWS-ympäristöön, sekä Elastic Beanstalk-palvelun toiminnan osalta tärkeät IAM-resurssit. Näitä mallipohjia on mahdollista kehittää entistä paremmiksi automaation osalta, sekä mallipohjien resursseihin on mahdollista lisätä lisää osia, kuten esimerkiksi Elastic Beanstalk -ympäristön luominen. Mallipohjien täydellinen toteutus migraatioprosessissa kärsi aikaresurssien puuteen vuoksi, mutta niillä saavutetaan pieni hyöty siitä huolimatta. Ne ovat myös hyvä rakennuspohja jatkokehityksen kannalta. Toteutetut mallipohjat ovat sijoitettu erilliseen julkiseen Github-repositorioon, josta ne ohjeistetaan lataamaan ohjeoppaassa (Liite 1, sivu 4). Cloudformation-mallipohjien hyödyntäminen edistää AWS Well-Architected Framework-viitekehyksen pilarien noudattamista esimerkiksi siten, että ympäristön nopeampi pystyttäminen kuluttaa vähemmän resursseja, sekä vikatilanteista toipuminen on mahdollisesti nopeampaa.

Ohjeopas: Elastic Beanstalk-sovelluksen pystyttäminen hyödyntäen Github Actions-alustaa

Toteutetun ohjeoppaan tarkoitus on toimia informatiivisena tiedon lähteenä avustamaan Jyväskylän ammattikorkeakoulua Elli-sovelluksen ja sen infrastruktuurin ylläpidossa, hallinnassa ja monistamisessa. Ohjeopas on myös suunnattu tuleville Ticorporate-opintojakson toteutuksille käytettäväksi oppimateriaalin tavoin, jotta monipuolisia oppimismahdollisuuksia voidaan edesauttaa ja opintojen osia kehittää eteenpäin.

Ohjeoppaassa käsitellään yksityiskohtaisesti AWS-ympäristön ja Elastic Beanstalk-palvelun valmistelu Elli-sovellukselle tai sen kaltaiselle sovellukselle. Ohjeopas on selostettu siihen tapaan, että sen avulla on mahdollista soveltaa toimintavaiheet myös eri sovellukselle. Ohjeen soveltaminen muuhun, kuin Elli-sovellukseen vaatii suorittajaltaan kuitenkin jonkin verran tietotekniikan ammattitaitoa.

Ohjeoppaan rakenne alkaa johdanto-osasta, jossa kerrotaan sen tarkoitus, keskeiset palvelut ja tarvittavat työkalut. MEAN-ohjelmistopinon osien tarkoitus ja yhteydet kehitettyihin tai käytettyihin osiin on kuvattu kuvion avulla. Pilvi-infrastruktuurin koko rakenne on havainnollistettu kaaviossa ja sen avulla pilviteknologioihin perehtynyt yksilö pystyy ymmärtämään koko ohjeoppaassa kerrotut asiat.

Seuraava osa aloittaa AWS-ympäristön valmistelujen ohjeistuksen. Näihin vaiheisiin kuuluu:

- Cloudformation-mallipohjien hyödyntäminen VPC-alueen, sekä tarvittavien IAM-resurssien luomiseksi
- Elastic Beanstalk -ympäristön luominen
- SSL-sertifikaatin luominen ja käyttöönotto Elastic Beanstalk-ympäristölle
- Tallennettujen konfiguraatioiden käyttö ympäristön pystyttämiseksi

Viimeisessä osassa siirrytään Github-repositorion valmisteluun ja sovelluksen julkaisun toimenpiteisiin. Näitä vaiheita ovat:

- Github-repositorion käyttöönotto eli kopioiminen tai kloonaaminen
- .env -tiedoston käyttöönotto paikallisesti ja etärepositoriossa
- IAM-käyttäjän pääsyavaimien käyttöönotto
- Julkaisuputkitiedostojen muokkaaminen
- Julkaisutoimenpiteiden laukaiseminen

Ohjeoppaan kaikki vaiheet ovat olennaisia osia pilviympäristön konfiguraation oppimisessa. Erityisesti Ticorporate-opintojakson tulevat toteutukset voivat hyötyä sen käyttämisestä, sillä jokaiseen sen kehitystiimiin kuuluu pilviasiantuntijan rooli. Opintojaksolla on tärkeää päästä kokemaan pilviympäristöjen konkreettinen konfigurointiprosessi oikeaa työelämää vastaavassa työskentely-ympäristössä. Ohjeoppaan sisällyttäminen opintojakson materiaaleihin madaltaa kynnystä oppia ja toteuttaa erilaisia menetelmiä pilviympäristöihin.

7 Pohdinta

Elli-sovelluksen migraatio Jyväskylän ammattikorkeakoulun pilviympäristöön suoritettiin onnistuneesti ja Elli-sovellus voidaan ottaa käyttöön. Kehittämisprosessi kokonaisuudessaan eteni alusta

loppuun niin kuin oli suunniteltu. Ennen migraatioprojektin aloittamista tiedossa oli, että kehittämisprosessi tulee sisältämään erilaisia hidasteita ja haasteita, mutta näiden ilmaantuminen kuuluu ohjelmistokehityksen vaiheisiin lähes poikkeuksetta. Opinnäytetyön aiheen rajauksessa oli määritelty tarkasti, että sen osat käsittelevät pilviympäristöä ja siihen liittyviä julkaisun osia. Tästä syystä Elli-sovellus on toiminnallisuuksiltaan siinä tilassa, mihin se toteutettiin Ticorporate-opintojaksolla. Vaikka Elli-sovellus on valmis käyttöönottoon, sovelluksessa itsessään on vielä toiminnallisia puutteita, jotka voivat vaikuttaa sen täyden potentiaalin ja hyödyn saavuttamiseen. Käyttöönotto ja Elli-sovelluksen hyödyntäminen jää loppujen lopuksi Jyväskylän ammattikorkeakoulun päätettäväksi. Myös toteutettu ohjeopas ja sillä saavutettavat hyödyt riippuvat siitä, kuinka sitä tullaan käyttämään tulevaisuudessa.

Johtopäätökset

Opinnäytetyön hyödyt suuntautuvat vahvasti oppimisen ja työelämätaitojen kehittämisen osa-alueisiin. Hyötyjä on kuvattu opinnäytetyön aikana useista eri näkökulmista ja ne koskettavat yrityksien saamaa hyötyä, mutta myös oppilaiden mahdollisuuksien lisääntymistä ammattitaidon kehittämisessä. Opinnäytetyöllä ei saavuteta aivan suoraa hyötyä tuloksilla, sillä Elli-sovelluksen ja migraatioprosessin tuloksena syntyneen ohjeoppaan käyttöönoton ja hyödyntämisen vastuu on niiden käyttäjillä. Se, mitä tuloksilla tullaan tekemään tulevaisuudessa, määrittää tuloksien todellisen hyödyn. Opinnäytetyön tuloksien saavuttaminen kuitenkin on edistänyt sitä, että tuloksilla on tulevaisuudessa mahdollisuus edistää oppimisen ja työelämätaitojen osa-alueita.

Jatkokehitys

Jatkokehitys Elli-sovellukselle ja siihen liittyville osille on erittäin tärkeässä asemassa opinnäytetyön tuloksien osalta. Jatkokehitysmahdollisuudet esimerkiksi Ticorporate-opintojakson tulevaisuuden toteutuksilla mahdollistavat ammatillisen oppimisen edistymisen. Elli-sovellus ja sen toiminnallisuudet ovat erittäin hyvä kohde jatkokehitykselle, sillä sovellusta kehittämällä paremmaksi sen kehittäjät saavat oppimismahdollisuuksia, mutta myös sovelluksen käyttäjäosapuolet hyötyvät sen paremmasta toiminnallisuudesta. Nämä toimenpiteet kuuluisivat perinteiseen ohjelmistokehityksen osiin Frontend-kehityksen ja Backend-kehityksen piirissä, mutta myös pilviteknologioiden ja automatisaation kehitys on tarpeellista Elli-sovelluksessa. Julkaisuputki, mallipohjat ja testausautomaatio ovat kaikki sellaisia osia, joiden kehittäminen hyödyttäisi opiskelijoita oppimisen kannalta.

Elli-sovelluksessa on myös suuri potentiaali liiketaloudelliselle kehitykselle, sillä sen toimintaa on mahdollista laajentaa useampien yritysten väliseksi. Tuotteistaminen ja sen suunnittelu esimerkiksi erilaisten pilvipalvelumallien osalta avaa useita erilaisia mahdollisuuksia.

Elli-sovelluksessa ja siihen liittyvissä osissa on kiistämätön potentiaali suurelle määrälle erilaisia mahdollisuuksia teknologian ja liiketalouden aloilla, mutta myös oppimisympäristöissä.

Eettisyys, luotettavuus ja kestävä kehitys

Elli-sovelluksen toteutuksen osissa hyödynnettiin tukiälynä tekoälypalveluita, kuten ChatGPT ja Microsoft Copilot. Tekoälyä käytettiin ainoastaan ohjelmakoodiin liittyvissä ongelmissa ja avustamaan sellaisissa ohjelmakoodin osien luomisessa, joihin ei ollut kertynyt aikaisempaa ammatillista osaamista. Tekoälylle oli esitetty kysymyksiä liittyen erilaisiin toteutustapoihin liittyen ympäristömuuttujiin ja pyydetty tekoälyä selventämään esimerkiksi Angular-ohjelmistokehyksen toimintaperiaatteita. Nämä esitetyt kysymykset auttoivat etenemään suunnitteluvaiheessa ja tekemään päätöksiä toimintatapojen valitsemiseksi. Jotkin toimintatavat valikoituvat myös kustannusresurssisyistä, joiden seurauksena esimerkiksi AWS:n määrittelemiä parhaita käytänteitä ei pystytty noudattamaan täysin.

Kaikissa opinnäytetyön vaiheissa on kiinnitetty huomiota hyvän tieteellisen käytännön (HTK) noudattamiseen. Tutkimuseettisen neuvottelukunnan (TENK) HTK-ohjeessa on todettu hyvän tieteellisen käytännön perusperiaatteiden olevan muun muassa luotettavuus ja rehellisyys (Tutkimuseettisen neuvottelukunnan julkaisuja 2/2023, 11). Työn lähteiden kriittinen analysointi ja niiden monipuolisuus, sekä käytettyjen lähteiden asianmukainen merkintä kuuluvat näihin käytäntöihin ja niin on toimittu.

Luvussa 2, jossa käsiteltiin tutkimusasetelmaa, käsiteltiin erilaisia ajatuksia liittyen eettisyyteen ja kestävään kehitykseen. Kokonaisuutena nämä ajankohtaisuuden, ammatillisen edistyksen ja oppimisen aihealueet liittyvät vahvasti eettisyyteen ja ottavat kantaa kestävän kehityksen näkökulmiin. Yhdistyneet Kansakunnat ovat määritelleet tavoitteita kestävälle kehitykselle, joiden tarkoitus on turvata nykyisille ja tuleville sukupolville hyvän elämisen mahdollisuudet. Yhdistyneet kansakunnat ovat yhteistyöjärjestö, joka suojelee maailmaa uusilta sodilta. Yhdistyneisiin Kansakuntiin kuuluu

193 jäsenvaltiota, jotka sopivat yhteisistä pelisäännöistä. Agenda 2030, eli kestävän kehityksen tavoiteohjelma koostuu 17 tavoitteesta ja niillä on yhteensä 169 alatavoitetta. (Tietoa YK:sta n.d.;Kestävä kehitys n.d.)

Tavoite 4 on määritelty otsikolla ”Hyvä koulutus” ja pitää sisällään ajatuksia siitä, että kaikilla tulisi olla avoin, tasa-arvoinen ja laadukas koulutus sekä elinikäiset oppimismahdollisuudet. Hyvän koulutuksen tavoitteen alatavoite 4.4 painottaa sitä, että vuoteen 2030 mennessä lisättäisiin merkittävästi nuorten ja aikuisten määrää, joilla on työllistymiseen, säällisiin työpaikkoihin ja yrittäjyyteen tarvittavat taidot, kuten tekninen ja ammatillinen osaaminen. (Hyvä koulutus n.d.) Elli -sovelluksen avulla yhteistöiden mahdollistaminen edistää tavoitteen ideaa sillä, että koulutuksen aikana on mahdollista saada erityisiä oppimismahdollisuuksia ja niitä pystytään peilaamaan oikeaan työelämään, edistäen työelämävalmiuksia.

Tavoite 8 otsikollaan ”Ihmisarvoista työtä ja talouskasvua” on kuvailtu seuraavasti: ”Edistää kaikkia koskevaa kestävää talouskasvua, täyttä ja tuottavaa työllisyyttä sekä säällisiä työpaikkoja”. Tämä tukee edellistä tavoitetta vahvasti, mutta sen alatavoite 8.6 osoittaa, että olisi syytä vähentää merkittävästi nuorten määrää, jotka eivät käy töissä tai opiskele (Ihmisarvoista työtä ja… n.d.). Samalla tavalla kuin toinenkin tavoite, Elli -sovelluksen avulla toteutuneet yhteistyöt edesauttavat opintojen jälkeisessä työllistymisessä. Opiskelija pääsee vuorovaikuttamaan yritysten kanssa jo opiskelunsa aikana, mahdollistaen harjoittelu- tai työpaikan saamisen helpottumisen.

Suomen YK-liiton sivuilla on kerrottu tavoitteet 7 ”Edullista ja puhdasta energiaa” ja 9 ”Kestävää teollisuutta, innovaatiota ja infrastruktuureja” (Kestävä kehitys n.d.). Nämä mukailevat Elli -sovelluksen eettisyyttä sitä kautta, että se hyödyntää pilviympäristöjä. Käytettäessä AWS:n parhaiden käytänteiden viitekehystä ja erityisesti ottaen huomioon siihen kuuluvan pilarin ”Kestävyys” (engl. *Sustainability*), erityisesti alatavoitteet 7.3 ja 9.4 täyttyvät. Kyseiset tavoitteet ovat Suomen YK-liiton sivuilla kuvailtu seuraavasti: ”Tuplata vuoteen 2030 mennessä energiatehokkuuden maailmanlaajuinen parantumisvauhti.” ja ” Uudistaa vuoteen 2030 mennessä infrastruktuuria ja jälkiasennusaloja kestävän kehityksen mukaisiksi, tehostaa resurssien käyttöä ja lisätä puhtaiden sekä ympäristöystävällisten teknologioiden ja tuotantoprosessien käyttöönottoa jokaisen maan omien valmiuksien mukaisesti.”. (Edullista ja puhdasta… n.d.;Kestävää teollisuutta, innovaatiota… n.d.)

Kestävyys -pilarin pohjimmainen ajatus ja tarkoitus on se, että ympäristövaikutukset pilvityöskentelyyn liittyen minimoidaan.

Pilvipalvelut ovat energiatehokkaampia, kuin perinteiset konesalit ja niillä on pienempi hiilijalanjälki. Tämä on sen ansiota, että pilvipalveluiden ylläpidossa panostetaan tehokkaisiin virta- ja jäähdytysteknologioihin, palvelimet ovat energiatehokkaita ja niiden hyötykäyttö on maksimoitu. Pilvipalveluiden käyttö vähentää tarvetta fyysisen raudan hankkimiselle ja pilvipalveluiden ydinperiaate perustuu jakamiselle. (Sustainability Pillar – AWS… 2021, 3.)

Lähteet

About secrets. N.d. Github Secrets-dokumentaatio Github Docs-sivustolla. Viitattu 2.4.2025. https://docs.github.com/en/actions/security-for-github-actions/security-guides/about-secrets.

Adetunji, D. 2022. AWS Identity and Access Management (IAM) – Explained With an Analogy. Verkkoartikkeli. Julkaistu 16.11.2022. Viitattu 13.3.2025. https://www.freecode-camp.org/news/aws-iam-explained/.

Amazon Virtual Private Cloud. 2024. AWS-dokumentaatiot. Julkaistu 27.3.2011. Päivitetty 9.12.2024. Viitattu 31.3.2025. https://docs.aws.amazon.com/pdfs/vpc/latest/userguide/vpc-ug.pdf.

AWS Certificate Manager. 2024. AWS-dokumentaatiot. Päivitetty 11.7.2024. Viitattu 31.3.2025. https://docs.aws.amazon.com/pdfs/acm/latest/userguide/acm-ug.pdf.

AWS Elastic Beanstalk. 2025. AWS-dokumentaatiot. Julkaistu 30.4.2024. Päivitetty 27.2.2025. Viitattu 31.3.2025. https://docs.aws.amazon.com/pdfs/elasticbeanstalk/latest/dg/awseb-dg.pdf

AWS Well-Architected Framework. 2024. AWS-dokumentaatiot. Julkaistu 6.11.2024. Viitattu 11.3.2025. https://docs.aws.amazon.com/wellarchitected/latest/framework/welcome.html.

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

Cloud Computing Technology. 2023. Singapore: Springer Nature Singapore. Viitattu 17.3.2025. https://finna.fi, SpringerLink Open Access eBooks.

Dalbhanjan, P. 2015. Overview of Deployment Options on AWS. AWS-dokumentaatiot. Viitattu 13.3.2025. https://d0.awsstatic.com/whitepapers/overview-of-deployment-options-on-aws.pdf.

Duggal, N. 2025. Learn What is MongoDB Atlas Database. Verkkoartikkeli. Julkaistu 26.1.2025. Viitattu 14.3.2025. https://www.simplilearn.com/tutorials/mongodb-tutorial/what-is-mangodb-atlas-database

Edullista ja puhdasta energiaa. N.d. Suomen YK-liiton verkkosivusto. Viitattu 18.3.2025. http://www.ykliitto.fi/edullista-ja-puhdasta-energiaa.

Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all. N.d. Yhdistyneiden kansakuntien kestävän kehityksen tavoitteet. Viitattu 4.3.2025. https://sdgs.un.org/goals/goal4#targets_and_indicators.

FF Ticorporate Demo Lab 2 (20 op). N.d. Jyväskylän ammattikorkeakoulun opetussuunnitelma. Viitattu 20.3.2025. https://opetussuunnitelmat.peppi.jamk.fi/48/fi/59/5158/777/0/40034?lang=fi.

Foster, G. N.d. Getting started with GitHub Actions. Opaskirjoitus graphite.dev-sivustolla. Viitattu 2.4.2025. https://graphite.dev/guides/github-actions.

How CloudFormation works. N.d. AWS-verkkosivusto. Viitattu 13.3.2025. https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cloudformation-overview.html.

Hristov, A. N.d. How automated testing enables DevOps. Verkkoaineisto. Viitattu 14.3.2025. https://www.atlassian.com/devops/devops-tools/test-automation.

Hurwitz, J. & Kirsch, D. 2020. Cloud Computing for Dummies. Toinen, uudistettu painos. Ho-boken, New Jersey: John Wiley & Sons. Viitattu 6.3.2025. https://janet.finna.fi, Ebook Central Academic Complete International Edition.

Hyvä koulutus. N.d. Suomen YK-liiton verkkosivusto. Viitattu 18.3.2025. http://www.yk-liitto.fi/hyva-koulutus.

Ihmisarvoista työtä ja talouskasvua. N.d. Suomen YK-liiton verkkosivusto. Viitattu 18.3.2025. http://www.ykliitto.fi/ihmisarvoista-tyota-ja-talouskasvua.

Introduction to AWS Elastic Beanstalk. 2023. Artikkeli geeksforgeeks-verkkosivustolla. Viitattu 26.3.2025. https://www.geeksforgeeks.org/introduction-to-aws-elastic-beanstalk/.

Jamk. N.d. Jyväskylän ammattikorkeakoulun verkkosivusto. Viitattu 20.3.2025. https://www.jamk.fi/fi/jamk.

Kestävä kehitys. N.d. Suomen YK-liiton verkkosivusto. Viitattu 18.3.2025. http://www.yk-liitto.fi/kestava-kehitys.

Kestävää teollisuutta, innovaatiota ja infrastruktuureja. N.d. Suomen YK-liiton verkkosivusto. Viitattu 18.3.2025. http://www.ykliitto.fi/kestavaa-teollisuutta-innovaatioita-ja-infrastruktuureja.

Microsoftin pilvi on Suomessa suosituin. 2022. Uutinen. Julkaistu 9.2.2022. Viitattu 9.3.2025. https://www.uusiteknologia.fi/2022/02/09/microsoftin-pilvi-on-suomessa-suosituin/.

MongoDB Basics. N.d. Verkkosivusto. Viitattu 14.3.2025. https://www.mongodb.com/resources/products/fundamentals/basics.

Nishimura, H. 2022. AWS for non-engineers. Shelter Island, New York: Manning Publications. Viitattu 11.3.2025. https://janet.finna.fi, Skillsoft Books ITPro (Skillport Platform).

Overview of Amazon Web Services. 2024. AWS-dokumentaatiot. Julkaistu 27.8.2024. Viitattu 28.2.2025. https://docs.aws.amazon.com/pdfs/whitepapers/latest/aws-overview/aws-overview.pdf.

Pihlajamäki, J. N.d. Mitä on DevOps?. Verkkoartikkeli. Viitattu 6.3.2025. https://otaverkko.fi/mita-on-devops/.

Pilvipalveluiden palvelumallit: PaaS, IaaS, BaaS ja Saas. 2024. Blogi-artikkeli. Julkaistu 12.6.2024. Viitattu 17.3.2025. https://www.hurja.fi/blogi/pilvipalveluiden-palvelumallit-paas-iaas-baas-ja-saas/.

Posey, B. 2024. Top public cloud service providers of 2025: How they compare. Verkkoartikkeli. Julkaistu 30.12.2024. Viitattu 9.3.2025. https://www.techtarget.com/searchcloudcomputing/tip/Top-public-cloud-providers-A-brief-comparison.

Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all. N.d. Yhdistyneiden kansakuntien kestävän kehityksen tavoitteet. Viitattu 4.3.2025. https://sdgs.un.org/goals/goal8#targets_and_indicators.

Sustainability Pillar- AWS Well-Architected Framework. 2021. AWS-dokumentaatiot. Julkaistu 2.12.2021. Päivitetty 6.11.2024. Viitattu 21.3.2025. https://docs.aws.amazon.com/pdfs/wellarchitected/latest/sustainability-pillar/wellarchitected-sustainability-pillar.pdf.

Tietoa YK:sta. N.d. Suomen YK-Liiton verkkosivusto. Viitattu 18.3.2025. http://www.ykliitto.fi/tietoa-yksta.

Tutkimuseettisen neuvottelukunnan julkaisuja 2/2023. Hyvä tieteellinen käytäntö ja sen loukkausepäilyjen käsitteleminen Suomessa. Julkistettu 15.3.2023. Viitattu 4.5.2025. https://tenk.fi/sites/default/files/2023-03/HTK-ohje_2023.pdf

Understanding GitHub Actions. N.d. Ohjedokumentaatio GitHub Docs-sivustolla. Viitattu 1.4.2025. https://docs.github.com/en/actions/about-github-actions/understanding-github-actions/.

Wadia, Y. 2018. AWS Administration- the Definitive Guide: Design, Build, and Manage Your Infrastructure on Amazon Web Services. Toinen painos. Birmingham, Mumbai: Packt Publishing. Viitattu 7.3.2025. https://janet.finna.fi, Ebook Central Academic Complete International Edition.

What is a GitHub Actions Secret? N.d. Ohjeistus Pulumi-verkkosivustolla. Viitattu 2.4.2025. https://www.pulumi.com/what-is/what-is-a-github-action-secret/.

What is Amazon S3? N.d. AWS-verkkosivusto. Viitattu 13.3.2025. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.

What is AWS CloudFormation? N.d. AWS-verkkosivusto. Viitattu 13.3.2025. https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html.

What is DevOps? N.d. Amazon Web Services-verkkosivusto. Viitattu 6.3.2025. https://aws.amazon.com/devops/what-is-devops/?nc1=f_cc.

What is Github? A Beginner’s Introduction to Github. 2023. Verkkoartikkeli. Julkaistu 17.11.2023. Viitattu 14.3.2025. https://kinsta.com/knowledgebase/what-is-github/.

What Is the MEAN Stack? N.d. MongoDB-verkkosivusto. Viitattu 25.3.2025. https://www.mongodb.com/resources/languages/mean-stack.

Why AWS Elastic Beanstalk? N.d. AWS-verkkosivusto. Viitattu 11.3.2025. https://aws.amazon.com/elasticbeanstalk/details/.

Liitteet

Liite 1. Ohjeopas: Elastic Beanstalk-sovelluksen pystyttäminen hyödyntäen Github Actions-alustaa

Ohjeopas: Elastic Beanstalk-sovelluksen pystyttäminen hyödyntäen Github Actions-alustaa

Sisällysluettelo

(jatkuu)

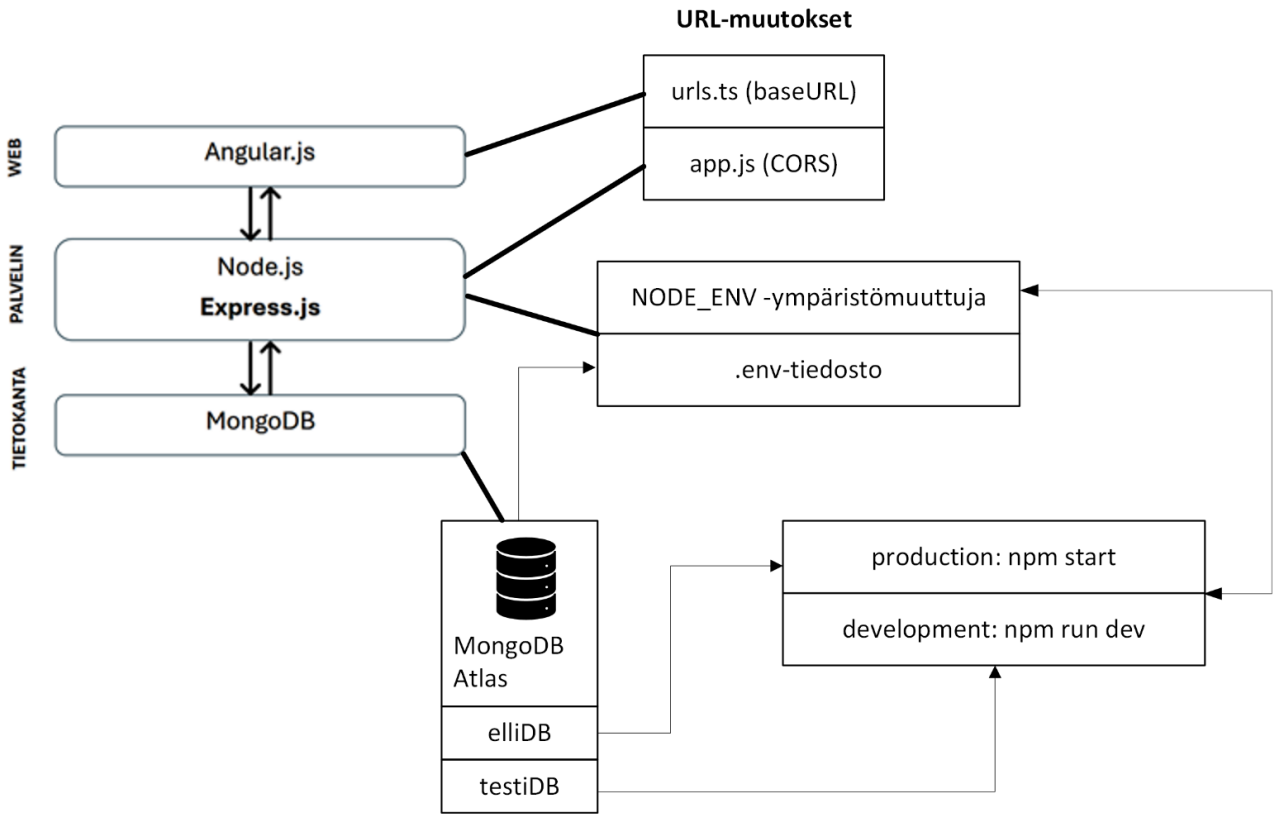
Johdanto

Tämä ohjeopas käsittelee sovelluksen pystyttämistä AWS Elastic Beanstalk-palveluun. Ohjeopas on tehty osana Elli-sovelluksen AWS-pilvimigraatioprosessia ja käsittelee ympäristöjen ja sovelluksen pystytyksen kyseisen sovelluksen näkökulmasta. Jotkin vaiheet voivat vaatia mukauttamista, jos sovelluksen ohjelmakoodin rakenne eroaa Elli-sovelluksen rakenteesta. Elli-sovellus ajetaan AWS Elastic Beanstalk -ympäristössä ja sen ohjelmakoodia säilytetään ja hallitaan Github-repositoriosta. Github Actions-alustan avulla Elli-sovelluksen julkaisutoimenpiteet ovat automatisoitu workflow-tiedostoissa.

Elli-sovelluksen keskeisiä käytettyjä alustoja ovat: AWS-pilviympäristö, Github (sisältäen Github Actions ja Github Secrets) ja MongoDB Atlas.

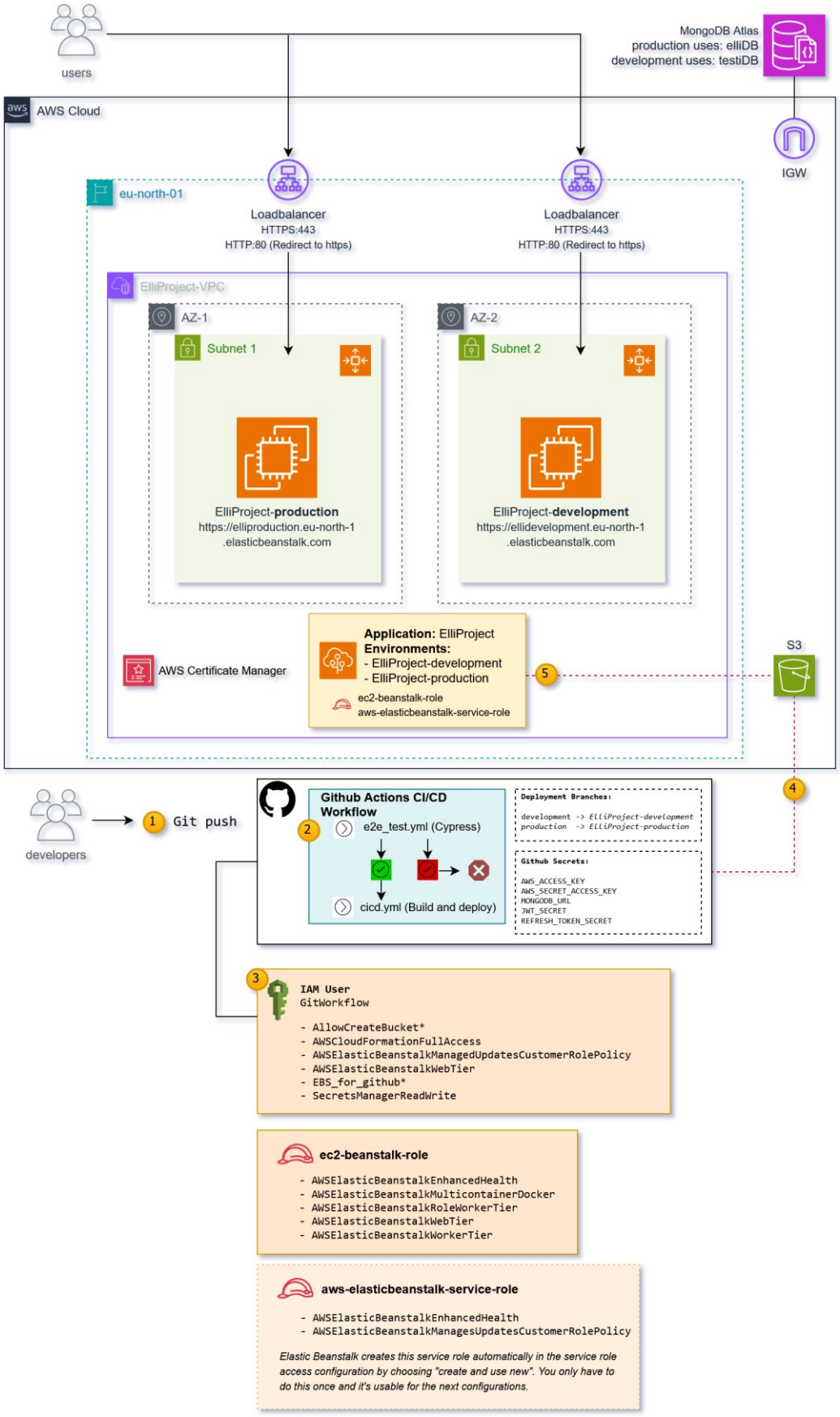
Ohjeoppaan seuraamiseen tarvitset:

- AWS-tilin, jolla on asianmukaiset oikeudet luoda ja käyttää käsiteltyjä AWS-resursseja
- Github-tunnukset ja/tai pääsyn Github Organisaatioon
- Git-työkalujen asennus tietokoneellesi
- Visual Studio Coden tai vastaavan ohjelman
- Node.js -sovellus, jota säilytetään Github-repositoriossa



Opinnäytetyön Kuvio 4: MEAN-stack-ohjelmistopino ja sen osat, joita on käsitelty julkaisuputken toiminnassa (What Is the MEAN Stack? n.d., muokattu)

(jatkuu)



Opinnäytetyön Liite 1: Elli-sovelluksen pilvi-infrastruktuurikaavio

(jatkuu)

AWS-ympäristön valmistelu

Opinnäytetyön tavoitteisiin kuului CI/CD-toimintamalliin liittyvien kehitysprosessien parantaminen, jonka seurauksena luotiin Cloudformation-mallipohjat ympäristön pystyttämisen helpottamiseksi. Nämä kyseiset mallipohjat pystyttävät VPC-alueen, sekä tarvittavat IAM-resurssit Elastic Beanstalk-palvelun käyttämiseksi. Ladataan tiedostot ja käytetään niitä AWS Cloudformation-palvelussa.

HUOM! *VPC-alueen ja sen osien nimet ovat kovakoodattuina tiedostossa vastaamaan Elli-sovelluksen infrastruktuurisuunnitelmaa. Jos haluat eri nimiset osat, voit vaihtaa osien nimet Tag-osion Name-arvosta. Huomioi myös eu-north (stockholm) saatavuusalue.*

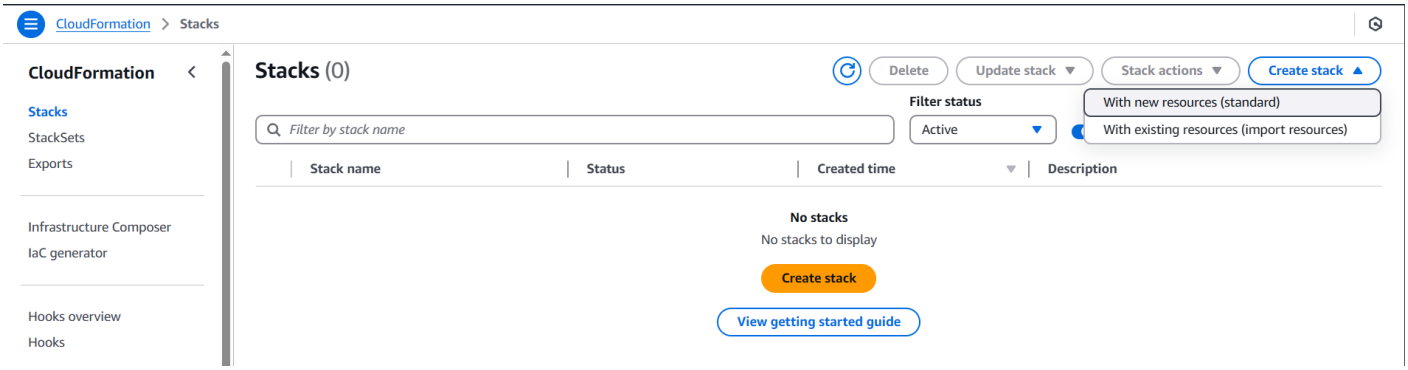
Lataa VPC-tiedosto tästä: Github repositoriolinkki

Lataa IAM-tiedosto tästä: Github repositoriolinkki

VPC-alueen, käyttäjän ja roolien luominen Cloudformation-mallipohjilla

Avaa Cloudformation -palvelu.

Kannattaa palveluita käyttäessä aina tarkistaa oikeasta yläkulmasta, millä alueella on työskentelemässä. Elli-sovelluksen haluamme Stockholm-alueelle.



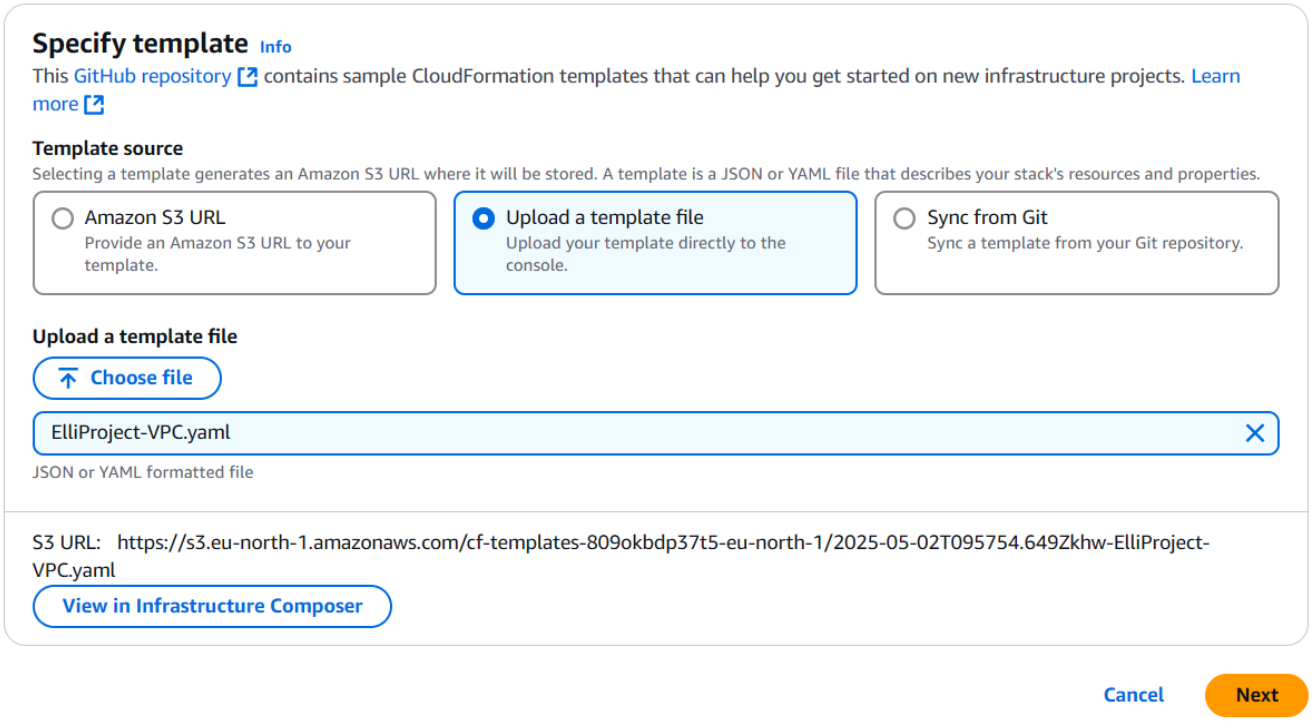
Create stack → With new resources (standard)

(jatkuu)

Cloudformation -palvelussa tehdään kaksi erillistä stackia juuri ladatuilla tiedostoilla, aloitetaan VPC-tiedostolla.

Valitaan ensin ”Choose an existing template” ja ”Upload a template file”.

”Choose file” -painikkeesta valitse juuri ladattu VPC.yaml -tiedosto.



Syötä stackille jokin sopiva nimi, jotta tunnistat sen myöhemmin. Siirry next-painikkeella seuraavaan vaiheeseen.

Voit valita Stack failure options-valikosta valinnan *”Delete all newly created resources”*. Tämä valinta poistaa kaikki tällä stackilla luodut resurssit, jos stack poistetaan. Siirry next-painikkeella seuraavaan vaiheeseen ja paina submit-painiketta.

Voit suorittaa VPC-tiedoston stackin luomisen jälkeen samat toimenpiteet IAM-tiedostolle: **Luo uusi stack.**

Stackit pystyttävät mallipohjissa määritellyt resurssit ja sen etenemistä voi seurata stackin tiedoista.

Elastic Beanstalk -ympäristön valmistelu

Ensimmäistä ympäristöä luodessa se konfiguroidaan käsin konsolin avulla. Luomme ensin kehitysympäristön, josta teemme saved configurations -tallennuksen. Tätä tallennusta pystymme käyttämään tuotantoympäristön luomiseen myöhemmin.

Jos muutoksista ei mainita tässä ohjeessa, ne voidaan jättää muokkaamatta ja säilyttää pohjan mukaisina tietoina/valintoina.

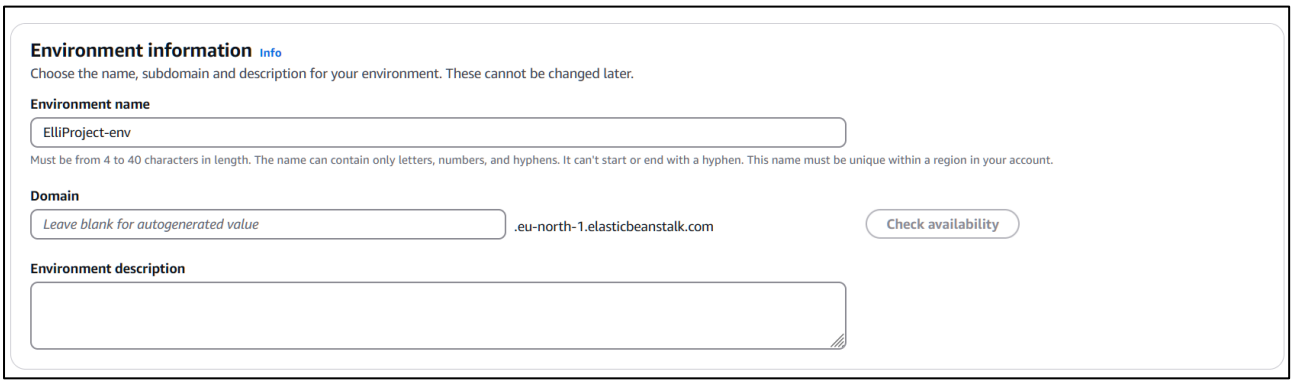
(jatkuu)

Avaa Elastic Beanstalk-palvelu → ”Create application” *(Jos ympäristöjä on jo olemassa niin: Environments → Create environment)*

Syötetään tiedot Step 1 -vaiheeseen Configure environment:

Application name: ElliProject
Environment name: ElliProject-development
Domain: ellidevelopment
Environment description: Development environment for Application Elli.

Application viittaa sovellusnimeen Elastic Beanstalk -palvelussa, jonka alla voi olla monta ympäristöä.

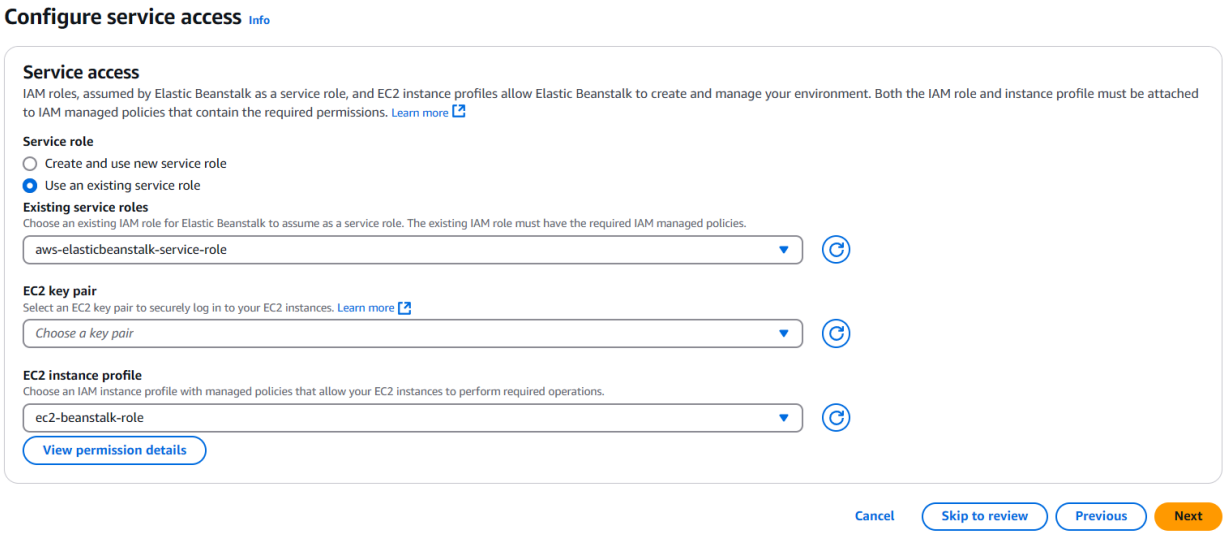


Platform type: Managed platform
Platform: Node.js
Application code: Sample application
Presets: Custom configuration

Siirry next-painikkeella seuraavaan vaiheeseen.

Step 2: Configure service access:

Koska loimme mallipohjan avulla tarvittavat roolit, Service role-valinnan pitäisi olla automaattisesti ”*Use an existing service role*”.



(jatkuu)

IAM role: aws-elasticbeanstalk-service-role
EC2 key pair: jätetään tyhjäksi
EC2 instance profile: ec2-beanstalk-role

Siirry next-painikkeella seuraavaan vaiheeseen.

Step 3: Set up networking, database and tags

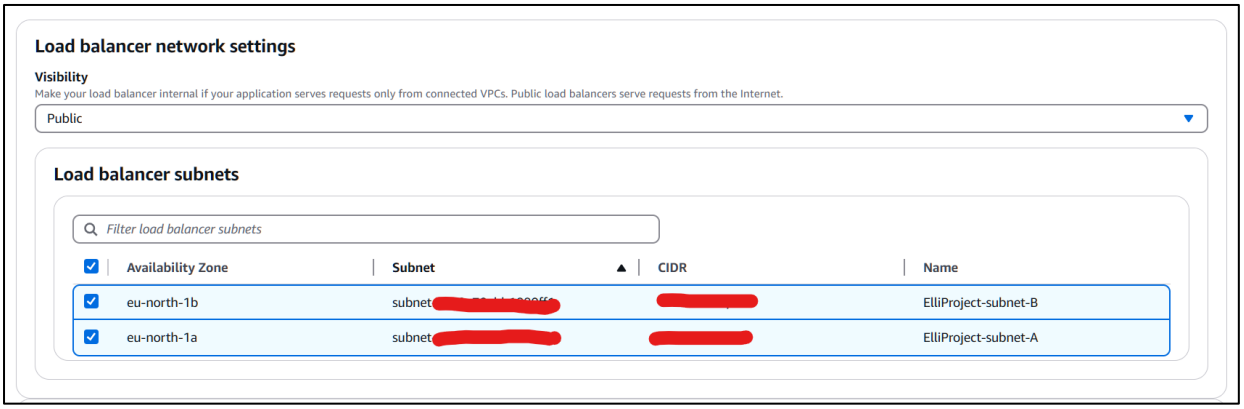
VPC-valikosta valitaan ympäristöä varten luotu VPC-alue, sekä instansseille valitaan aliverkko, jolle ne voidaan luoda.

Esimerkiksi Elli -sovelluksen infrastruktuurissa on suunniteltu, että tuotanto- ja kehitysympäristöt sijaitsevat eri aliverkoissa, omilla saatavuusalueillaan. Kehitysympäristö, eli development, sijoitetaan tästä syystä ElliProject-subnet-B -aliverkkoon.

VPC: ElliProject-VPC (Jos muokkasit yml-tiedostoa nimien osalta, niin valitse sen niminen)
Public IP addess: Activated
Instance subnets: eu-north-1b (Valitse aliverkko, jonne instanssit pystytetään)

Siirry next-painikkeella seuraavaan vaiheeseen.

Step 4: Configure instance traffic and scaling



Load balancereille kaikki aliverkot valitaan. ElasticLoadBalancingV2 vaatii vähintään 2 aliverkkoa toimiakseen.

Root volume type: General Purpose 3(SSD)
Environment type: Load balanced
Load balancer network settings: Public, eu-north-1a, eu-north-1b (Kummatkin aliverkot valitaan)

Siirry next-painikkeella seuraavaan vaiheeseen.

(jatkuu)

Step 5: Configure updates, monitoring and logging

Health reporting: Basic
Managed platform updates: Activated pois päältä

Siirry next-painikkeella seuraavaan vaiheeseen. Tarkista valinnat ja valitse ”submit”.

SSL-sertifikaatin luominen ja käyttöönotto

Elli -sovelluksen toteutuksessa on käytetty ilmaista Let’s Encrypt-tahon SSL-sertifikaattia. Tämä käytetty SSL-sertifikaatti on voimassa 90 päivää, jonka jälkeen se pitää itse uusia. Sertifikaatin luominen onnistuu toiminnassa olevan EC2-instanssin kautta, joten ympäristön pystyttämisen jälkeen se on mahdollista luoda itse. Instanssiin otetaan SSH-yhteys, jonka jälkeen sen avulla voidaan asentaa certbot -työkalu sertifikaatin luomista ja käyttöönottoa varten. Certbot -työkalu luo sertifikaatin tiedostot instanssin tallennustilaan, josta ne ovat mahdollista scp-yhteydellä ladata omalle tietokoneelle. Omalta tietokoneelta nämä tiedostot seuraavaksi siirretään AWS Certificate Manager -palveluun, josta Elastic Beanstalk -palvelun on mahdollista käyttää sitä. Sertifikaatin uusiminen tapahtuu samalla tavalla, kuin sen luominenkin.

Elastic Beanstalk -ympäristön tulee olla konfiguroituna load balanced -tilaan.

Sertifikaatti luodaan toimivan instanssin kautta. Instansseilla tulee olla SSH-yhteyden ottaminen sallittuna, keypair -avain luotuna ja ladattuna omalle tietokoneelle.

1. **Luodaan EC2 Key pair -avain:**
   Avaa EC2 -palvelu → Vasemmalla valikkolistassa otsikon ”Network & Security” alla on ”Key Pairs” → Create keypair

   Nimeä avain, käytä mieluiten jotain helposti kirjoitettavaa nimeä, sillä tähän viitataan myöhemmin. Valitse ”private key file format” -valinnaksi .pem, jotta se toimii komentokehotteessa.

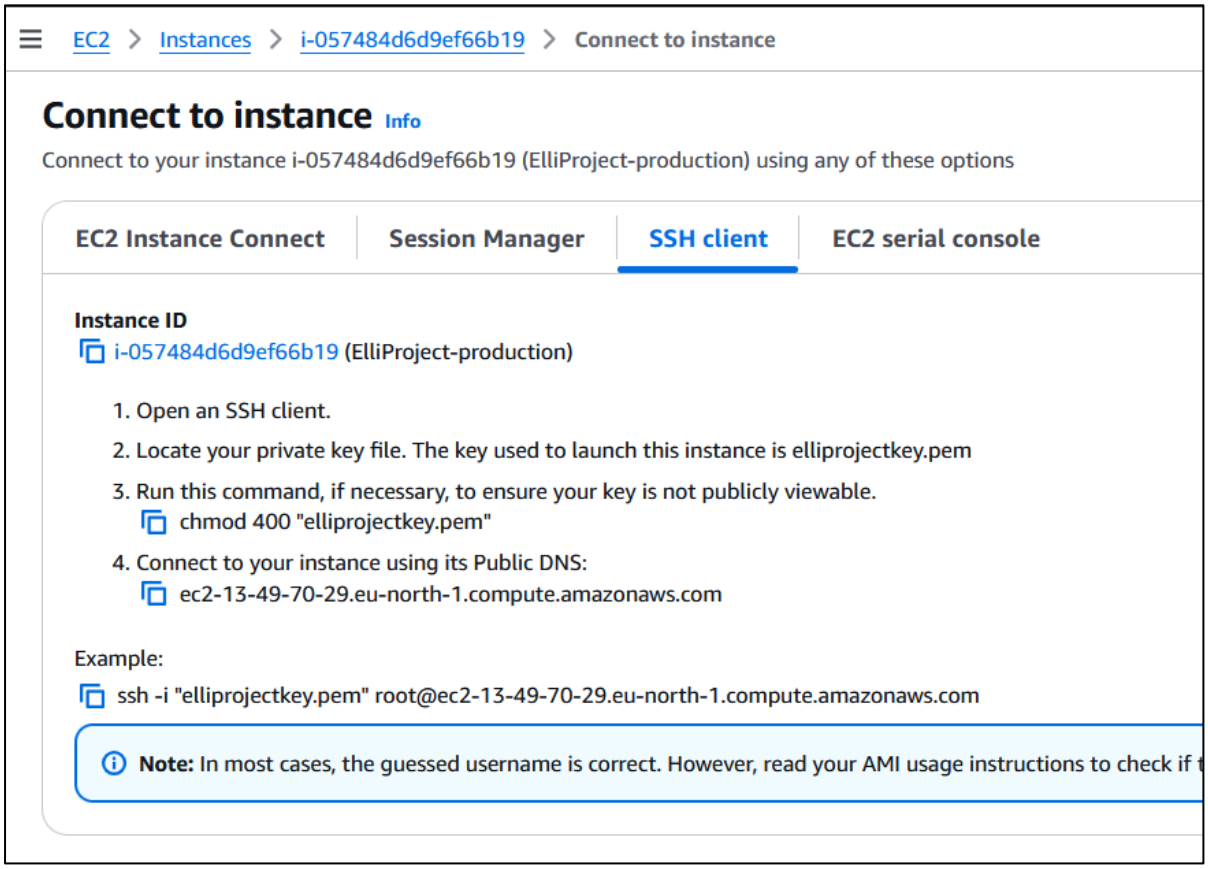
   Lisätään avain käytettäväksi Elastic Beanstalk-ympäristölle:
   Avaa Elastic Beanstalk-palvelu → Environments → Valitse ympäristö (klikkaa nimestä) → Vasemmalla valikkolistassa on ”Configuration” → Service access (klikkaa edit-painiketta) → Valitse EC2 key pair -kohdalle pudotusvalikosta luomasi avain → Valitse ”apply”

   Elastic Beanstalk päivittää ympäristön tämän jälkeen, instanssien uudelleen pystyttämisessä voi mennä jonkin aikaa.

(jatkuu)

2. **Otetaan yhteys instanssiin SSH-yhteydellä:**
   Avaa EC2-palvelu → Vasemmalla valikkolistassa ”Instances” → Valitse instanssi listalta klikkaamalla sen ID:tä → Ylävalikossa on valinta ”Connect” → SSH Client

   HUOM! *Instanssin tulee olla edellisen vaiheen jälkeen ”Status check: 3/3 checks passed” -tilassa.*



   Kopioi ”Example” -kohdan komento, mutta muutetaan käyttäjä rootista ec2-useriksi:
   Esimerkiksi:
   ssh -i "elli_key.pem" ec2-user@ec2-13-48-133-134.eu-north-1.compute.amazonaws.com

   Avaa komentokehote järjestelmänvalvojana ja siirry siihen hakemistoon, johon olet tallentanut .pem avain-tiedoston. Syötä ssh-komento komentokehotteeseen ja vastaa ”yes” ilmaantuvaan kysymykseen.

3. **Certbot -työkalun asennus:**
   Syötä seuraavat komennot yksitellen instanssin komentokehotteeseen:

   sudo yum install -y certbot
   sudo systemctl stop nginx

   sudo certbot certonly --standalone -d <oma domain URL> -d <oma domain URL>

(jatkuu)

Esimerkki yllä olevasta komennosta:
sudo certbot certonly --standalone -d ellidevelopment.eu-north-1.elasticbeanstalk.com -d www.ellidevelopment.eu-north-1.elasticbeanstalk.com

Vastaa kysymyksiin, joita asennus antaa seuraavaksi. Odota kärsivällisesti, kunnes kehotteeseen ilmenee success-viesti. Tämän jälkeen syötä vielä:

sudo systemctl start nginx

4. **Sertifikaattien lataaminen omalle tietokoneelle:**
Ladataan sertifikaatit omalle tietokoneelle instanssin tallennustilasta.

Komentojen ajamista varten annetaan ec2-user -käyttäjälle oikeudet kansioon, jossa sertifikaatit sijaitsevat.

Syötä komentokehotteeseen:

sudo chmod -R 755 /etc/letsencrypt

Avaa uusi komentokehoteikkuna järjestelmänvalvojana ja siirry sillä siihen kansioon, jossa .pem avain-tiedosto on ladattuna. Ladataan seuraavaksi certbot-työkalun luomat kolme tiedostoa.

Syötä kehotteeseen ensimmäinen komento:

scp -i <pem-tiedosto> <ec2-user@instanssin osoite>: /etc/letsencrypt/live/<oma domain URL>/fullchain.pem <latauskansion polku omalle koneelle>

Esimerkki komennosta:

scp -i elli_key.pem ec2-user@ec2-13-48-133-134.eu-north-1.compute.amazonaws.com:/etc/letsencrypt/live/ellidevelopment.eu-north-1.elasticbeanstalk.com/fullchain.pem C:\Users\Marianne\Downloads

Komennon rakenne on siis .pem-avain + käyttäjänimi@instanssiosoite + : + ladattavan tiedoston sijainti + latauskohteen sijainti

Yllä oleva komento lataa **fullchain.pem** -tiedoston. Ladataan vielä kaksi muuta tiedostoa, jotka kuuluvat sertifikaattiin. Tähän voidaan käyttää samaa komentoa, kuin yllä, mutta vaihdetaan ladattavan tiedoston nimi polun lopusta. Syötä komennot:

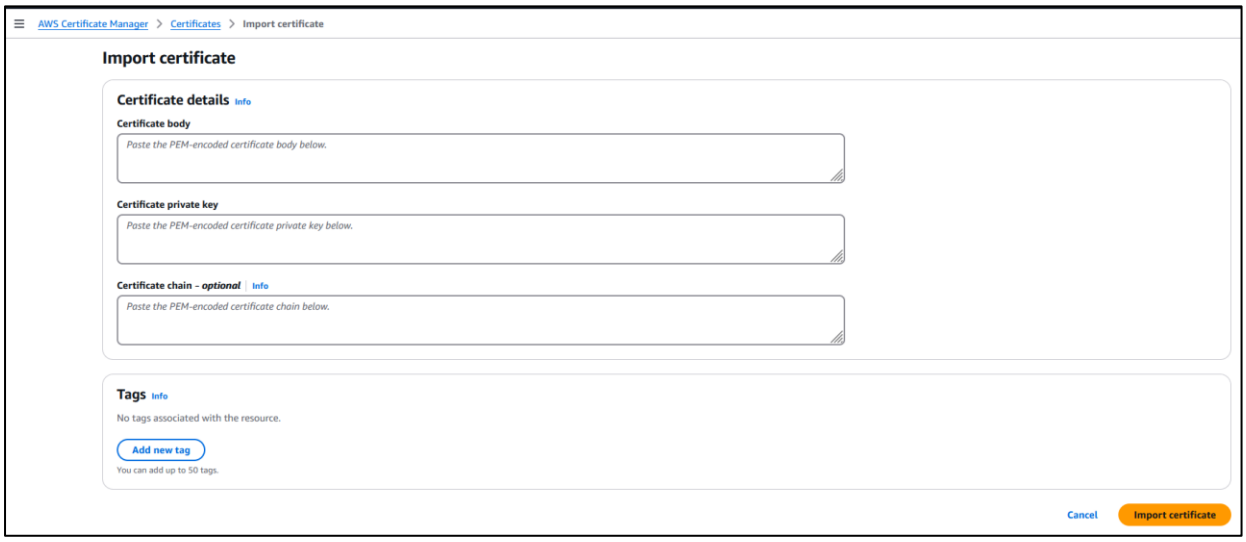
*scp -i <pem-tiedosto> <ec2-user@instanssin osoite>: /etc/letsencrypt/live/<oma domain URL>/**privkey.pem** <latauskansion polku omalle koneelle>*

*scp -i <pem-tiedosto> <ec2-user@instanssin osoite>: /etc/letsencrypt/live/<oma domain URL>/**chain.pem** <latauskansion polku omalle koneelle>*

(jatkuu)

- Kannattaa huomioida Windows-tyyliset kenoviivat latauspolussa, jos työskentelet Windows-koneella.
- Instanssin osoitteen löytää AWS EC2 -palvelusta instanssin SSH connect -osiosta, jos se on hukassa. Se on sama osoite, jota käytettiin SSH-yhteyden ottamiseen (muista ec2-user!)
- Voit avata ladatut tiedostot esimerkiksi Visual Studio Codella.

5. **Sertifikaatin vieminen AWS Certificate Manager -palveluun**
   Avaa Certificate Manager-palvelu → Vasemmalla valikkolistassa on ”Import Certificate”



   Syötä tiedostojen sisällöt kenttiin. Sisällytä mukaan -----BEGIN CERTIFICATE----- and -----END CERTIFICATE-----.

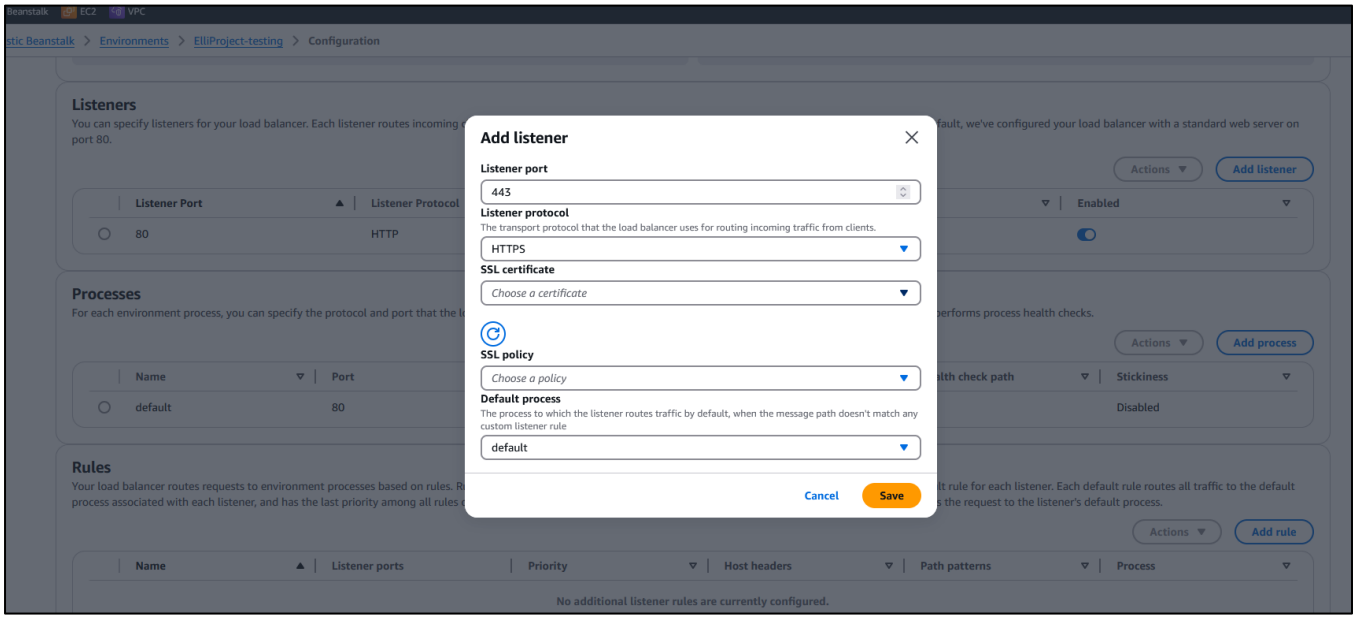
   Certificate body: fullchain.pem (vain ylempi osa)
   Private Key: privkey.pem -tiedoston sisältö
   Certificate chain: chain. pem -tiedoston sisältö

   Tallenna sertifikaatti “Import certificate” -painikkeella.

6. **Sertifikaatin lisääminen Elastic Beanstalk -ympäristöön**
   Avaa Elastic Beanstalk-palvelu → Environments → Valitse ympäristö (klikkaa nimestä) → Vasemmalla valikkolistassa on “Configuration” → Instance traffic and scaling (klikkaa edit-painiketta)

   Lisätään https-portti kuuntelijaksi kuormanjakajalle. Vieritä alas, kunnes löydät kohdan ”Listeners” → Klikkaa ”Add listener”

(jatkuu)



Listener port: 443
Listener protocol: HTTPS
SSL Certificate: Valitse listasta sertifikaatti, jonka lisäsit ACM-palveluun.

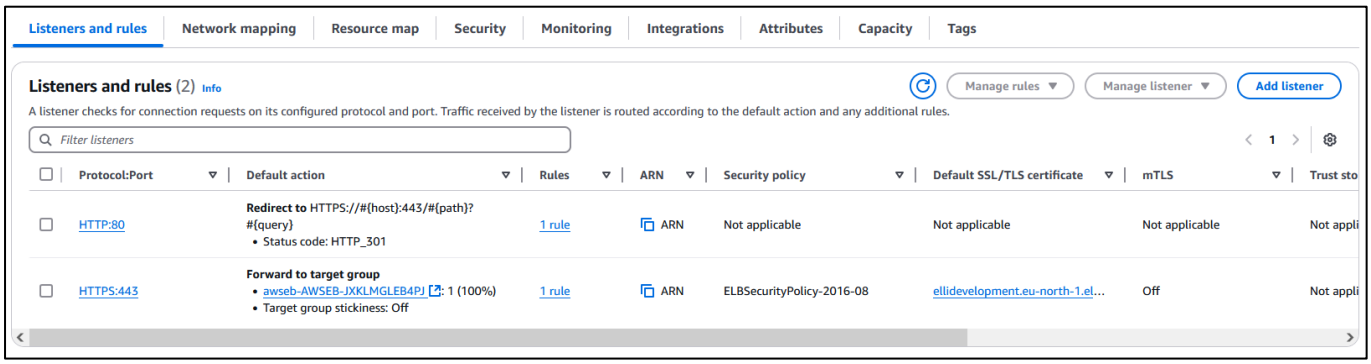
Valitse ”Save”, jonka jälkeen voit valita sivun alhaalta ”Apply”.

7. **Reititysten asetukset**
   Avaa EC2-palvelu → Vasemmalla valikkolistassa on ”Load Balancers” → Valitse kuormanjakaja (klikkaa nimestä)

   Elastic Beanstalk on luonut kuormanjakajan automaattisesti, valitse listasta oikean ympäristön kuormanjakaja. Jos sinulla on vain yksi ympäristö, listassa on vain yksi kuormanjakaja.

   ”Listeners and rules” -kohdassa pitäisi näkyä kaksi sääntöä: HTTPS:443 ja HTTP:80.

   HTTPS:443 pitäisi olla asetettu ”Forward to target group” -tilaan, sitä ei tarvitse muokata.



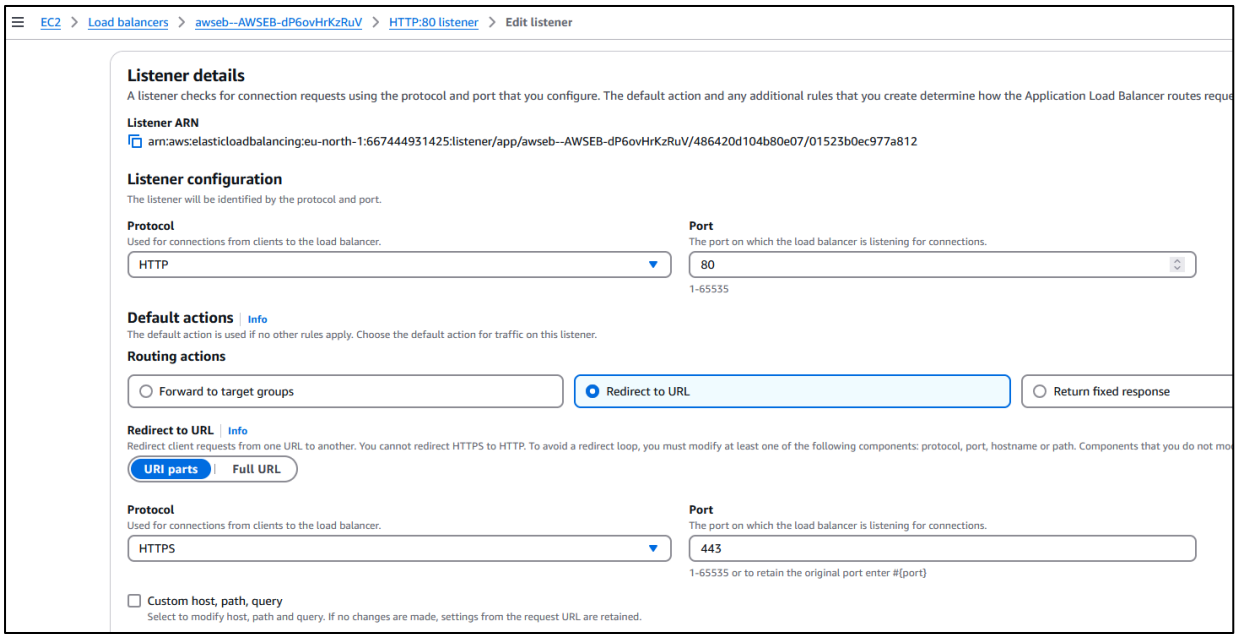
Valitse HTTP:80 klikkaamalla sen nimestä (Protocol:Port) → Oikealla ylhäällä on ”Actions” → ”Edit listener”

(jatkuu)

Edit Listener: Listener details (ei muutoksia)
Protocol: HTTP
Port: 80

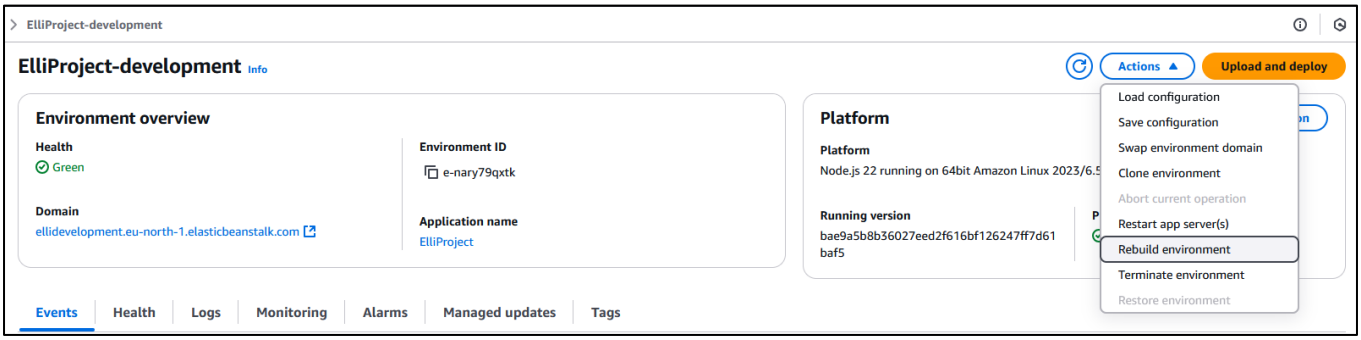
Default Actions-valikon alla:
Routing actions: Redirect to URL (HTTPS ja PORT 443)

Tallenna muutokset “Save changes” -painikeella.



Tämä toimenpide reitittää uudelleen porttiin 80 (http) kulkevan liikenteen ohjaten sen porttiin 443 (https). Yhteys on siis salattu, eikä http-yhteyttä pysty ottamaan sovellukseen selaimella.

Joissakin tapauksissa Elastic Beanstalk -ympäristö pitää pystyttää uudelleen porttisääntöjen muokkaamisen jälkeen. Jos ympäristössä ilmenee ongelmia:
Avaa Elastic Beanstalk-palvelu → Environments → Valitse ympäristö (klikkaa nimestä) → Actions → Rebuild Environment



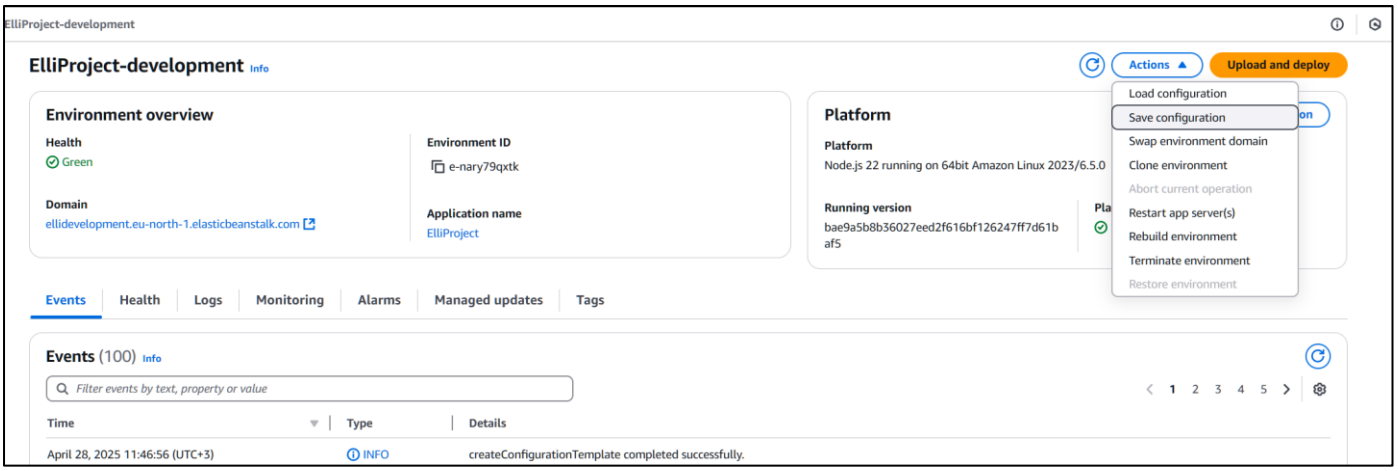
(jatkuu)

Ympäristön pystyttäminen saved configurations -valinnan kautta

Olemassa olevan ympäristön voi tallentaa saved configurations -pohjaksi ympäristön asetuksista:

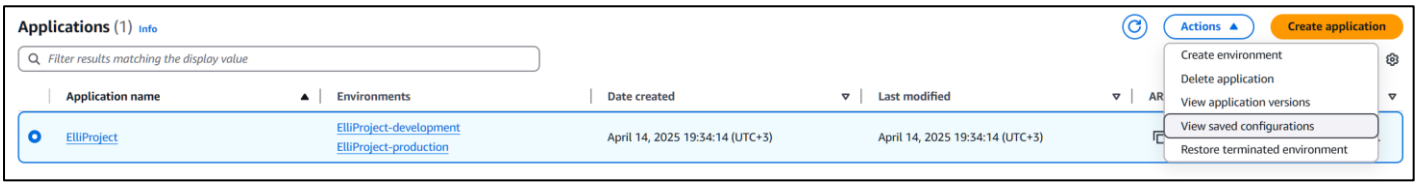
Avaa Elastic Beanstalk -palvelu → Environments → Valitse ympäristö (klikkaa nimestä) → Actions → Save configuration

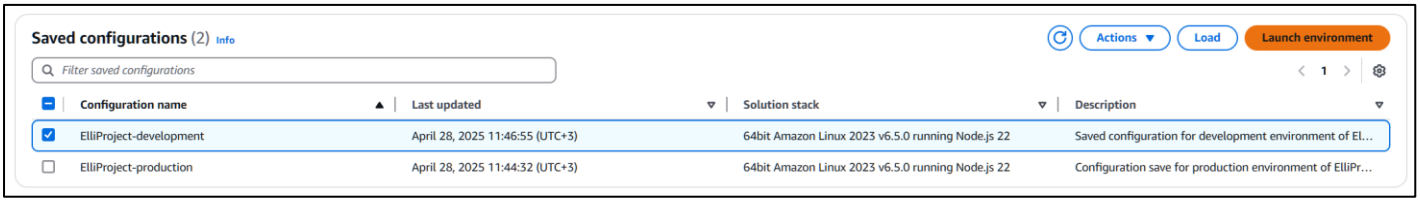
Anna tallennukselle osuva nimi ja sille sopiva kuvaus, tallenna valitsemalla “Save”.



Saved configuration-pohjaa pystyy käyttämään uuden ympäristön pystyttämiseen sovelluksen (Application) valikosta:

Avaa Elastic Beanstalk-palvelu → Vasemmalla valikkolistassa “Applications” → Valitse sovellus valintaympyrällä → Actions → View saved configurations → Valitse luotu pohja → Launch environment



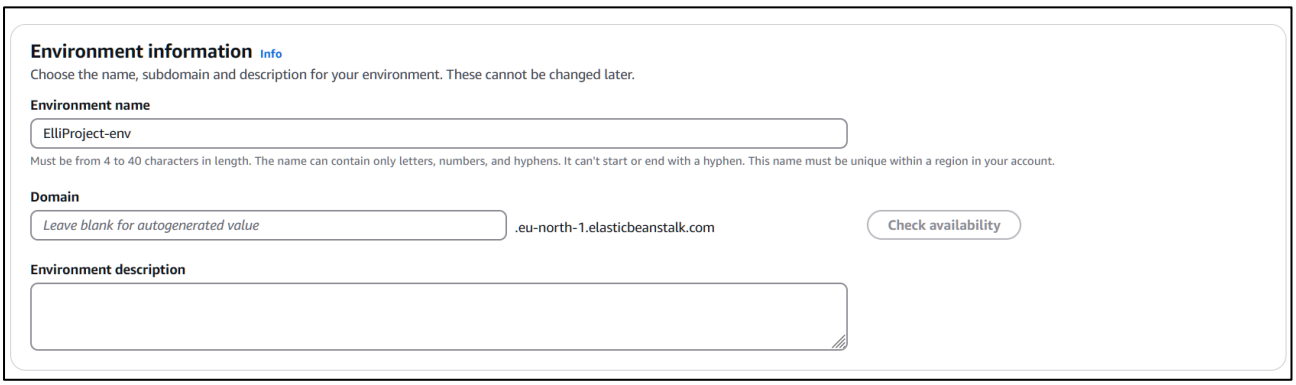


Elastic Beanstalk -palvelu käynnistää ympäristön pystyttämisen ohjatut vaiheet. Näihin vaiheisiin pitää pohjan käytöstä huolimatta syöttää itse joitakin tietoja, tässä ohjeessa käymme ne läpi. Jos muutoksista ei mainita tässä ohjeessa, ne voidaan jättää muokkaamatta ja säilyttää pohjan mukaisina tietoina/valintoina. Käytämme esimerkkisyötteinä tilannetta, jossa pystyttäisimme uutta production -nimistä tuotantoympäristöä.

(jatkuu)

Step 1: Configure environment

Ympäristökohtaiset tiedot tulee täyttää uudelle ympäristölle.



Environment name: Elliproject-production
Domain: elliproduction
Environment description: The production environment for Application Elli.

Siirry seuraavaan vaiheeseen “next” -painikkeella.

Step 2: Configure service access

EC2 key pair: Valitse tähän .pem -avain, jonka loit SSL-sertifikaatin luomisen vaiheessa

Roolien pitäisi olla automaattisesti valittuina (aws-elasticbeanstalk-service-role, ec2-beanstalk-role)

Siirry seuraavaan vaiheeseen “next” -painikkeella.

Step 3: Set up networking, database and tags

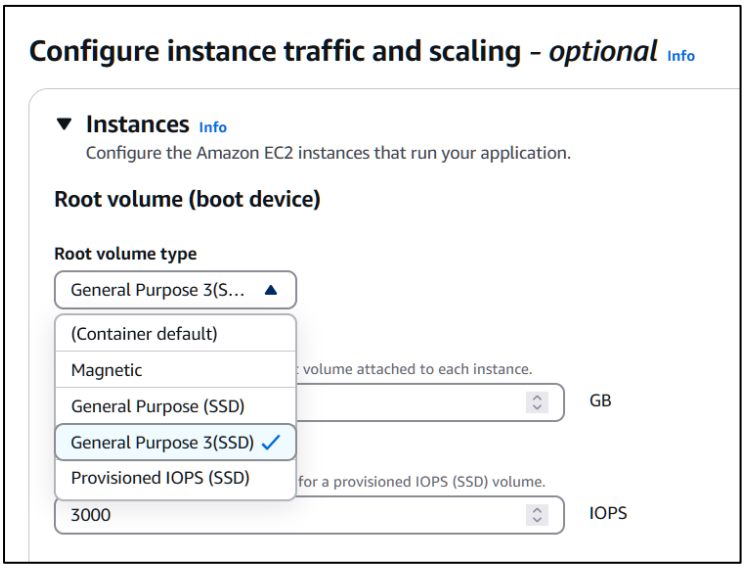
VPC: ElliProject-VPC (Jos muokkasit yml-tiedostoa nimien osalta, niin valitse sen niminen. Valitse kuitenkin sama kuin ensimmäisen ympäristön luomisessa.)
Public IP addess: Activated
Instance subnets: eu-north-1b (Valitse aliverkko, jonne instanssit pystytetään. Valitse eri aliverkko, kuin ensimmäisen ympäristön luomisessa, jos haluat ympäristöt eri alueille.)

Siirry seuraavaan vaiheeseen “next” -painikkeella.

(jatkuu)

Step 4: Configure instance traffic and scaling

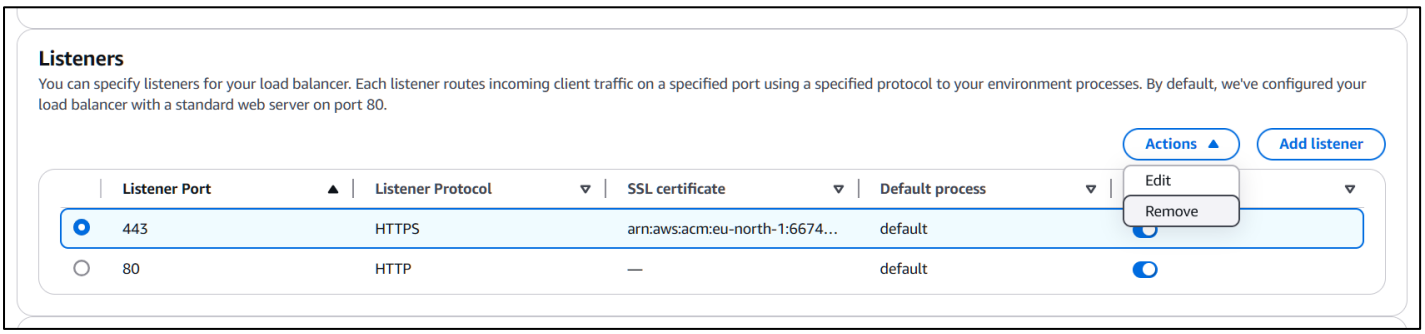
Instanssien ominaisuuksia ja skaalautuvuuden asetuksia muutetaan:



Root volume type: General Purpose 3 (SSD)
Load balancer network settings: eu-north-1b, eu-north-1a

Koska käytössä on jo olemassa olevan ympäristön tallennettu pohja, ”Listeners”-kohdassa on päällä portti 443 (https) ja siihen liitetty aikaisemman ympäristön SSL-sertifikaatti. Poistetaan kuuntelija ja sen sertifikaatti oikealta valikkopainikkeesta:

Listeners → Actions-painike → Remove



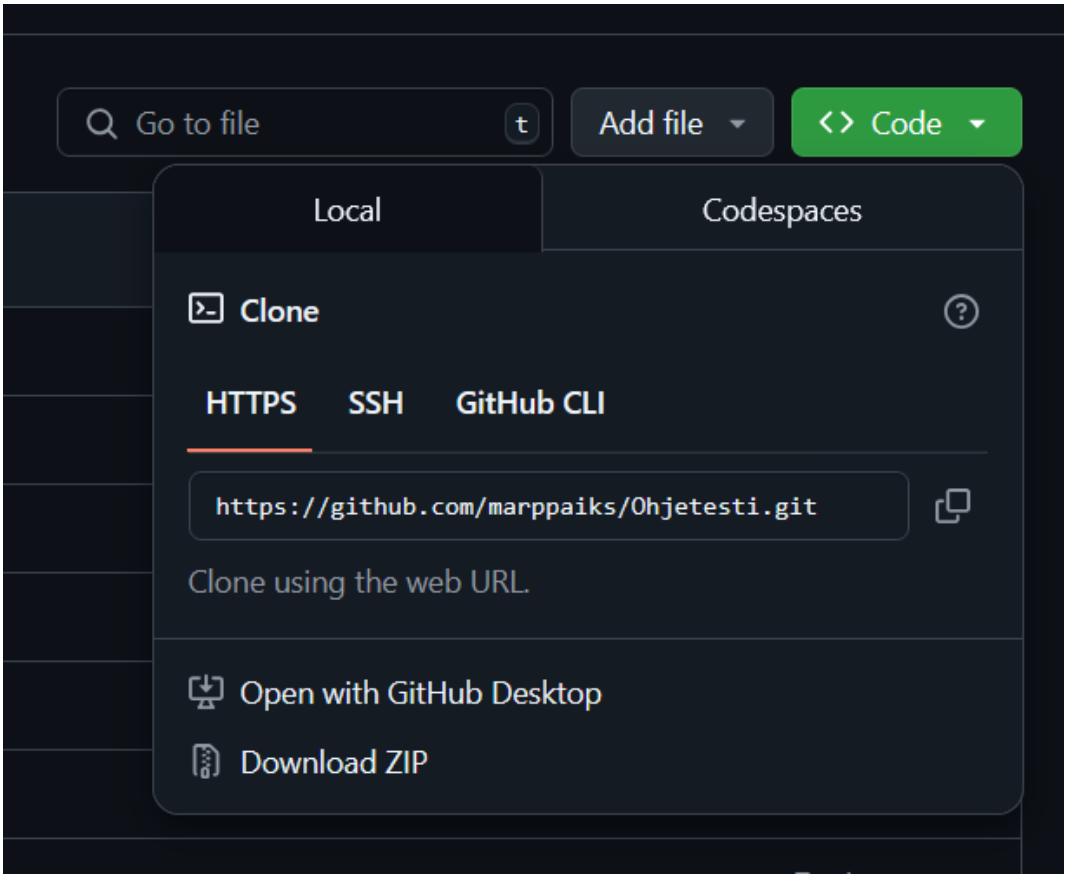
Ohjatun konfiguraation voi suorittaa loppuun ja odottaa, että Elastic Beanstalk -palvelu luo ympäristön. Kun ympäristö näyttää vihreää, sille voi luoda ja lisätä SSL-sertifikaatin (kts. Sivu 8).

(jatkuu)

Github-repositorion valmistelut

Elli-sovelluksen repositorio sijaitsee Jyväskylän ammattikorkeakoulun Github Organisaation alla. Repositorion kopioimiseksi ja sen käyttämiseen tarvitsee käyttöluvan.

Repositorion paikallista kehitysympäristöä varten on viisasta luoda oma kansio omalle tietokoneelle. Repositorion URL-osoitteen löytää repositorion omalta sivulta:



Luodussa kansiossa annetaan komentorivikomento, voit avata kansion esimerkiksi Visual Studio Codella ja syöttää komennon sen terminaali-ikkunaan:

`git clone <Repositorion URL-osoite>`

Siirrytään vielä repositorion kansioon erikseen komennolla:

`cd ElliProject` (tai muun sovelluksen tapauksessa repositorion nimi ElliProjectin sijaan)

.env -tiedosto paikallisesti ja etärepositoriossa:

.env -tiedostossa säilytetään ohjelmakoodin arkaluontoista dataa. Tästä syystä se on määritelty gitignore-tiedostoon, jotta sitä ei käsitellä etärepositoriossa. Tästä syystä .env -tiedosto ei tule tiedostojen mukana, kun kopioit repositorion.

(jatkuu)

Paikallista kehitystä varten .env -tiedosto pitää manuaalisesti itse lisätä Server-hakemiston juureen. Paikallista .env-tiedostoa ei tarvita, jos sovellusta ei ole tarkoitus ajaa paikallisessa ympäristössä. Elli-sovelluksessa .env -tiedosto sisältää:

- **MONGODB_URL** -muuttujan, jossa on MongoDB Atlas käyttäjätunnus, salasana ja tietokantaklusterin osoite.
- **JWT_SECRET** -muuttujan (JSON Web Token secret, sisäänkirjautumisen token)
- **REFRESH_TOKEN_SECRET** -muuttujan (liittyy kirjautumisen tokeniin)

Nämä samat .env -tiedoston muuttujat tulee lisätä etärepositoriolle saatavaksi Githubin asetuksissa.

Elli-sovelluksen automaattinen julkaisuputki käyttää Github Secrets -ominaisuutta käsitelläkseen .env -tiedoston ympäristömuuttujia. Tämä tekee näiden tietojen käyttämisestä tietoturvallista ohjelma- ja julkaisuputkitiedostoissa.

Avaa Github repositorio → Settings → Secrets and variables → Actions → New repository secret

Lisätään yllä kolme mainittua muuttujaa yksitellen: Name-kenttään MONGODB_URL, JWT_SECRET ja REFRES_TOKEN_SECRET. Secret-kenttään sitä vastaava tieto.

IAM Access Key

Jotta Elastic Beanstalk -palvelun ympäristö toimii Github Actions -alustan kanssa, aikaisemmin Cloudformation -mallipohjalla luodulle Gitworkflow -käyttäjän pääsyavaimet pitää myös lisätä Github Secrets -osioon.

Avaa AWS IAM-palvelu → Users → Gitworkflow → Security credentials → Access keys: Create access key

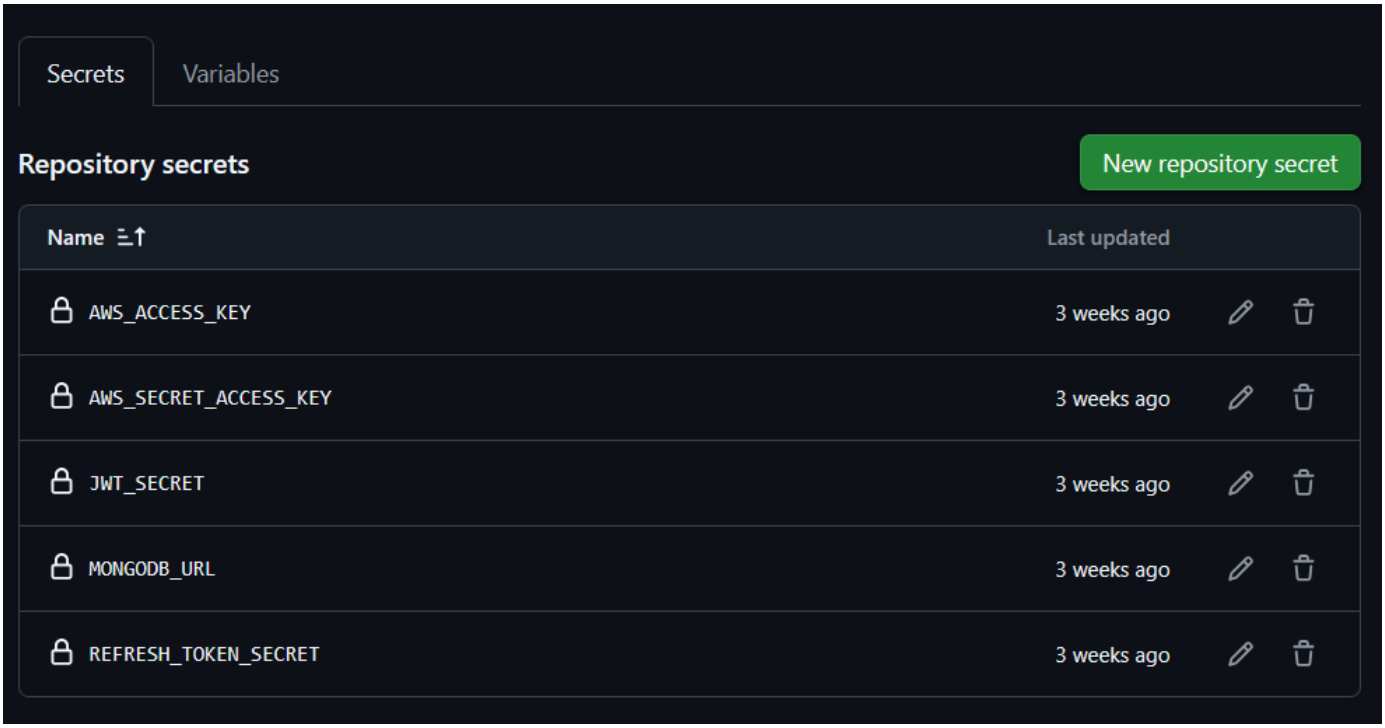
Use case: Third-party service

Set description tag: Github Actions workflow

Kopioi Access Key ja Secret Access key -tiedot. Näitä ei enää näe uudelleen ja jos ne hukkuvat, luo uusi pääsyavain ja poista vanha käytöstä.

Lisää kyseiset pääsyavaimen tiedot Github Secrets -osioon nimillä **AWS_ACCESS_KEY** ja **AWS_SECRET_ACCESS_KEY**. On tärkeää, että pääsyavaimet nimetään näillä nimillä, sillä niitä kutsutaan julkaisuputken tiedostoissa näillä nimillä. (Koskee myös .env -muuttujia.)

(jatkuu)



Julkaisuputkitiedoston muutokset ja julkaisu

Elli-sovelluksen julkaisuputkitiedostot sijaitsevat repositorion juuressa sijainnissa .github\workflows

Tiedostoja on kaksi:

- e2e_test.yml, vastuussa sovelluksen end-to-end -testeistä
- cicd.yml, vastuussa Elastic Beanstalk -palveluun julkaisusta

Kloonauksen jälkeen oletushaarana Elli-sovelluksessa on production -haara. Siirrytään development -haaraan. Avaa repositorio esimerkiksi Visual Studio Codella ja syötä sen terminaali-ikkunaan:

`git checkout development`

(Jos työskentelet muun sovelluksen parissa, syötä ensin `git branch development`, sitten `git checkout development`)

Muutamme julkaisuputkitiedostojen tietoja vastaamaan luodun Elastic Beanstalk -ympäristön tietoja.

Elli-sovelluksen cicd.yml -julkaisuputkitiedosto ottaa repositorion haaran nimen ympäristömuuttujaksi ja siksi on tärkeää käyttää haaroissa nimiä ”development” ja ”production”.

(jatkuu)

Muutetaan cicd.yml -tiedostossa:

- Job-vaiheen ensimmäisessä step-toimenpiteessä määritellään e2e_test.yml -tiedoston sijainti repositorion osoitteella. Repositorion osoite vaihdetaan, jos kyse ei ole Elli-sovelluksesta. *Esimerkiksi: Org_tai_käyttäjänimi/repositorionNimi/.github/workflows/tiedostonNimi@haaranNimi*

```
jobs:
  call-e2e_test:
    uses: Tiko-web-and-data/ElliProject/.github/workflows/e2e_test.yml@development
```

- Seuraavassa step-toimenpiteessä määritellään ympäristömuuttujia:

```
    env:
      AWS_REGION: eu-north-1
      APPLICATION_NAME: 'ElliProject'
      BRANCH_NAME: ${{ github.ref_name }}
      APP_URL: ${{ github.ref_name == 'production' && 'https://elliproduction.eu-north-1.elasticbeanstalk.com' || 'https://ellidevelopment.eu-north-1.elasticbeanstalk.com' }}
      ENVIRONMENT_NAME: ${{ github.ref_name == 'production' && 'ElliProject-production' || 'ElliProject-development' }}
      NODE_ENV: ${{ github.ref_name == 'production' && 'production' || 'development' }}
```

Merkityt kohdat ovat sellaisia, jotka ovat vaihdettava, jos käsittelet muuta kuin Elli-sovellusta. Elli-sovellusta varten tietoja ei tarvitse muuttaa, ellei kyseessä ole ympäristön uudelleen pystytys eri tiedoilla.

AWS_REGION: Alue, jonne Elastic Beanstalk -ympäristö on luotu
APPLICATION_NAME: Elastic Beanstalk -palvelussa annettu sovelluksen nimi
APP_URL: Production-ympäristön URL ensin, toisena development URL
ENVIRONMENT_NAME: Production-ympäristön nimi ensin, toisena development-ympäristön

cicd.yml -julkaisuputkitiedostossa APP_URL -muuttujalla korvataan ohjelmakooditiedostoissa tietoja (URL-osoitteita app.js -tiedostossa ja urls.ts -tiedostossa) ja nämä tiedostot voivat olla erilaisia omassa sovelluksessasi. Myös pakatun tiedoston nimi julkaisuputkitiedostossa on hyvä vaihtaa, jos et käsittele Elli-sovellusta.

(jatkuu)

Sovelluksen julkaisun laukaiseminen Elastic Beanstalk-ympäristöön

Muutoksien jälkeen Visual Studio Coden terminaali-ikkunassa voidaan syöttää komennot:

`git status`

Tällä komennolla voidaan katsoa tiedostot, joita on muutettu.

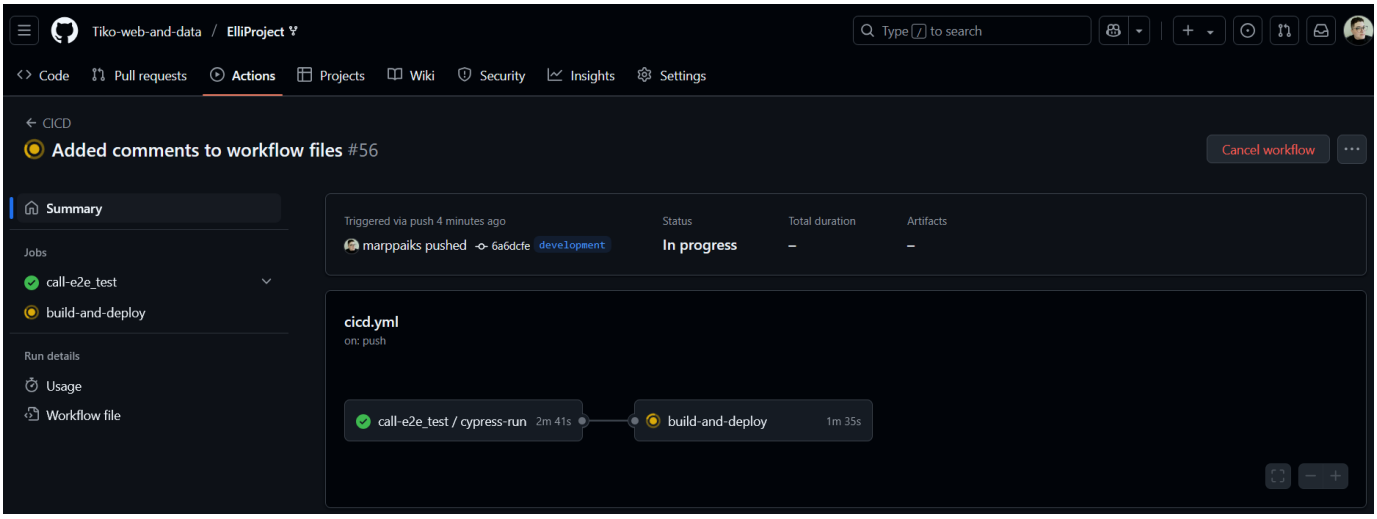
`git add .`

`git commit -m ”Write your commit message here”`

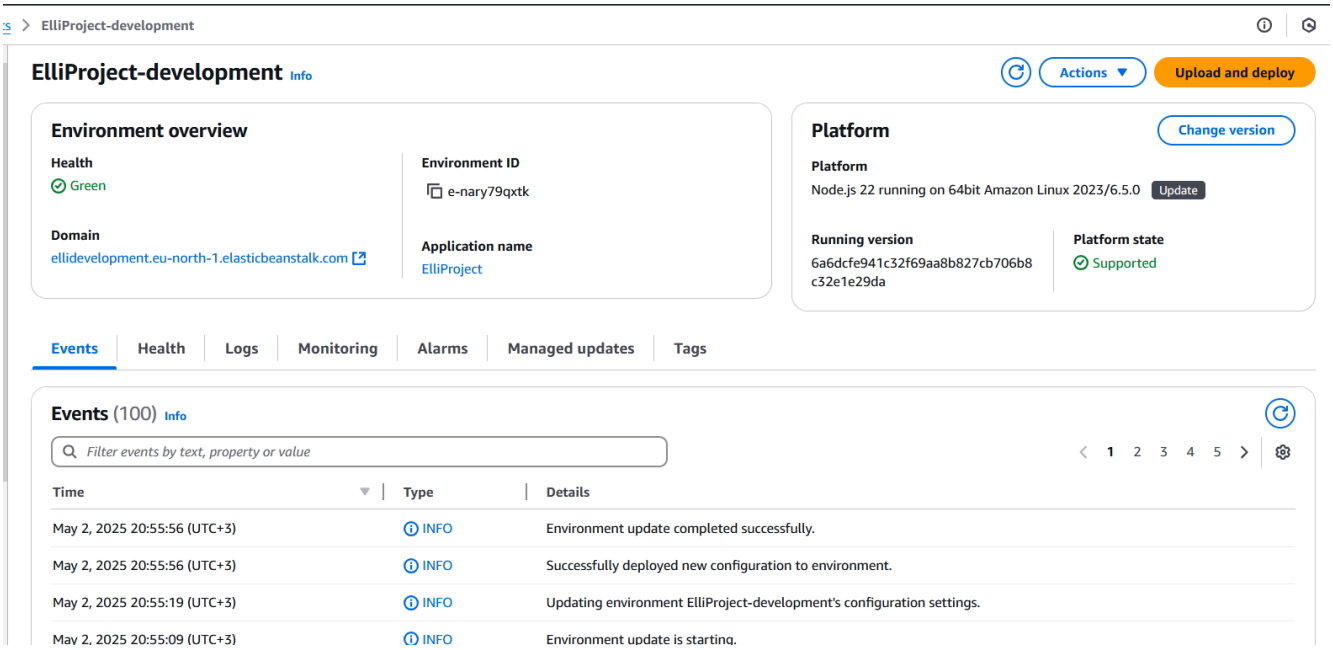
`git push`

(Jos luotu haara on aivan uusi push-komennon jälkeen git pyytää asettamaan `git push --set-upstream origin development` -komennon erikseen.)

Yllä teimme muutokset kehitysympäristöön, eli development -haaraan. Push-komento laukaisee julkaisuputkitoiminnot ja ne ovat seurattavissa Actions -välilehdeltä repositorion sivuilla:



(jatkuu)



Julkaisun edistymistä voi seurata Elastic Beanstalk -palvelusta.

Tuotantoympäristön julkaisemiseksi voit terminal-ikkunassa syöttää seuraavat komennot:

`git checkout production`

(Jos työskentelet muun sovelluksen parissa, syötä ensin `git branch production`, sitten `git checkout production`)

Kummatkin ympäristöt toimivat samanlaisten julkaisuputkitiedostojen kautta. Repositorion haaroilla on kuitenkin omat tiedostonsa, koska tulevaisuudessa ne voivat erota toisistaan. Jos loit production-haaran ensimmäistä kertaa, se on kopio development-haarasta ja sisältää juuri muutetun cicd.yml -tiedoston, eikä muutoksia tarvitse syöttää uudelleen. Elli-sovelluksen haaroissa voit joutua tekemään muutokset toiseen haaraan, koska production-haara on jo olemassa.

Jotta saamme annettua push-komennon, tehdään muutoksia cicd.yml -tiedostoon production -haarassa:

```
jobs:
  call-e2e_test:
    uses: Tiko-web-and-data/ElliProject/.github/workflows/e2e_test.yml@production
```

Vaihdetaan haaran nimi polun loppuun.

Muutoksien jälkeen Visual Studio Coden terminaali-ikkunassa voidaan syöttää komennot:

`git status`

Tällä komennolla voidaan katsoa tiedostot, joita on muutettu.

`git add`

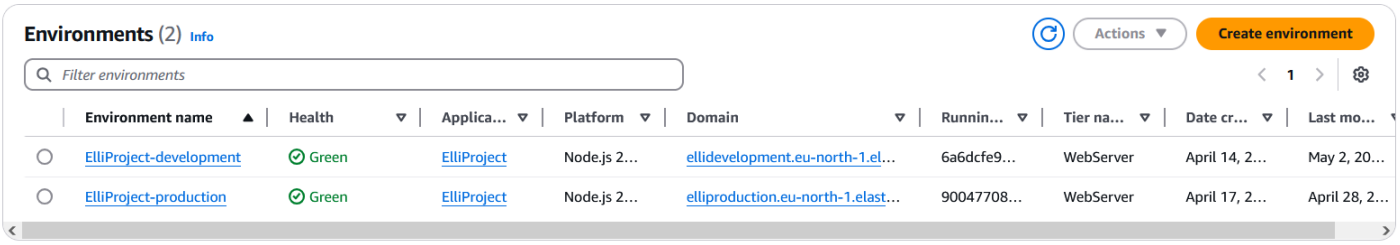
(jatkuu)

```
git commit -m ”Write your commit message here”
```

```
git push
```

Tämän jälkeen push-komento laukaisee tuotantoympäristön (production) julkaisun Elastic Beanstalk -palveluun.

Jos haara ei ollut aivan uusi kopio juuri tehdyistä muutoksista, muutokset pitää vielä tehdä production-haaraan samalla tavalla, kuin development-haarassa. Tämän jälkeen samaan tapaan ajetaan push-komento.



Kummatkin ympäristöt ovat vihreänä Elastic Beanstalk -palvelussa.