Esko Lähdesmäki

SOVELLUSKEHITYS KONTTIYMPÄ-RISTÖSSÄ

Opinnäytetyö

Tradenomi (AMK)

Tietojenkäsittely

2020

Kaakkois-Suomen ammattikorkeakoulu

Tekijä/Tekijät	Tutkintonimike	Aika
Esko Lähdesmäki	Tradenomi (AMK)	Toukokuu 2020

Opinnäytetyön nimi

Sovelluskehitys konttiympäristössä

32 sivua

Toimeksiantaja

Metatavu Oy

Ohjaaja

Janne Turunen

Tiivistelmä

Opinnäytetyön aiheena oli kehittää toimeksiantajan sovelluskehitystyöhön ratkaisumallinnus. Tavoitteena oli tutkia ja kehittää prosessi, jonka avulla sovelluskehittäjät pystyvät työskentelemään paikallisessa kehitysympäristössä ja käyttöjärjestelmästä riippumatta. Parantamalla kehitystyön osa-alueita, edistetään työmukavuutta ja työskentelyn sujuvuutta.

Virtuaaliset palvelinratkaisut mahdollistavat kustannustehokkaan ylläpidon. Toimeksiantaja hyödyntää omassa tuotannossaan Amazonin pilvipalvelua. Pilvipalvelussa hyödynnetään avoimeen lähdekoodiin perustuvaa Kubernetesta. Sen avulla ylläpidetään sivustoja, jotka rakentuvat konttien avulla. Automaattisen skaalauksen ansiosta sivusto voi mittakaavaltaan olla mitä vain.

Kontti on perinteistä virtuaalikonetta kevyempi ratkaisu, joka mahdollistaa nykyaikaisen mikropalveluarkkitehtuurin toteuttamisen. Tämä toi haasteita työn tavoitteena olleen kehitysympäristön luomiseen. DevOPS-periaatteen mukaan kehitysympäristön tulisi olla mahdollisimman automatisoitu ja mahdollistaa sujuva työskentely.

Työssä esitellään kaksi tapaa luoda kehitysympäristö konttiteknologiaa hyödyntäen. Toinen on docker-composea hyödyntäen luotu ympäristö ja toinen on Minikuben avulla luotu paikallinen Kubernetes. Paikallisen Kubernetes-ympäristön kehitysympäristön luomiseen käytetään siihen tarkoitettua ohjelmaa, nimeltään Tilt. Ohjelmien avulla luotiin Wordpress-kehitysympäristö. Sivustolle luotiin teema, johon tehdyt muutokset tulisivat olla näkyvissä, ilman konttien uudelleen rakentamista. Työhön valitut ohjelmistot tarjosivat hyvän vaihtoehdon toimeksiantajan sovelluskehitysympäristöksi.

Kasvavan yrityksen täytyy pystyä standardoimaan prosesseja, jotta projektien uudet työntekijät pääsevät työskentelemään mahdollisimman omatoimisesti. Paikallinen Kubernetes ei aina ole paras tapa toteuttaa sovelluskehitysympäristöä. Tärkeintä on yhdessä päättää, kuinka prosessista saadaan mahdollisimman jouheva ja yksiselitteinen sovelluskehittäjälle. Kubernetes yhdessä Tilt-ohjelman kanssa antoi hyvän mahdollisuuden toteuttaa halutunlainen sovelluskehitysympäristö.

Asiasanat

ohjelmistokehitys, virtuaaliympäristö, konttiteknologia, Kubernetes, DevOPS

South-Eastern Finland University of Applied Sciences

Author (authors)	Degree	Time
Esko Lähdesmäki	Bachelor of Business Administration	May 2020

Thesis title

Software development in container environment — 32 pages

Commissioned by

Metatavu Oy

Supervisor

Janne Turunen

Abstract

The objective of the thesis was to research and develop a process to help developers to work in a local development environment regardless of their operating system. Environments are built by using containers and run by Kubernetes. Nowadays many companies are using virtualized environments in production. Development process and workflow should be automated as much as possible according to DevOPS thinking.

Container technology means light virtualization when compared to traditional virtual computers. This thesis presented two different styles on how to create a development environment with containers. The first development environment was made by using the docker-compose tool to create a Wordpress website with a custom theme. The second environment was built in local Kubernetes with the same website and a file sync provided by the Tilt development tool.

The study showed that running all development environments on Kubernetes was not wise. This was mainly because Kubernetes took lot of resources from a computer and could make the process unnecessarily complicated. A growing company needs standardised processes, so it's important to agree together on a common way for creating development environments. This thesis was a helpful overview of development environments with containers and it also demonstrated the capabilities of Kubernetes and Tilt.

Keywords

software development, virtual environment, container technology, Kubernetes, DevOPS

SISÄLLYS

1 JOHDANTO

Opinnäytetyön aiheena oli kehittää Metatavu Oy:lle nykyisten ja tulevien projektien sovelluskehitystyöhön ratkaisumallinnus. Opinnäytetyön tutkimuksellinen osuus on yrityksen sisäistä kehitystyötä, jota on jatkossa tarkoitus hyödyntää projekteissa. Tavoitteena oli tutkia ja kehittää prosessi, jonka avulla sovelluskehittäjät pystyvät työskentelemään paikallisessa kehitysympäristössä ja käyttöjärjestelmästä riippumatta.

Työn toimeksiantaja, Metatavu Oy, keskittyy liiketoiminnassaan sovelluskehitykseen. Se tuottaa tietoteknisiä ratkaisuja ja palveluita. Metatavu Oy tekee websovellusten lisäksi mm. sulautettuja järjestelmiä, järjestelmien ylläpitoa ja konsultaatiota. Yrityksen verkkoratkaisujen tuotanto on Amazonin pilvipalvelussa, Kubernetes-palvelimella. Opinnäytetyön sovelluskehitysympäristöksi haluttiin mahdollisimman monipuolinen alusta ja siksi käytännön toteutuksen kehitysalustana toimii paikallinen Kubernetes-ympäristö.

Opinnäytetyön luvussa 2 esitellään tarkemmin teoriaa kehitysympäristöjen taustalta. Teoriaa kuvataan aluksi laajoina ja isoina ilmiöinä, joista edetään yksittäisiin teknologioihin ja ohjelmistoihin. Luvussa 3 esitellään niiden ohjelmien asentamista, joita hyödynnetään opinnäytetyön sovelluskehitysympäristön luomisessa. Lisäksi havainnollistetaan ohjelmien perustoimintoja. Teorian ja ohjelmistojen jälkeen opinnäytetyössä siirrytään käytännön toteutukseen, jossa luodaan sovelluskehitysympäristö konttien avulla sekä paikallisella Kubernetes-palvelimella.

Luvun 3 loppuosassa kuvataan vielä vaihe vaiheelta, kuinka Kubernetes-sovelluskehitysympäristö luodaan. Luvussa 4 esitellään tulokset ja pohditaan työhön liittyviä näkökulmia työn tekijän ja toimeksiantajan -näkökulmasta. Luvussa 5 on päätäntö, jossa kerrotaan työn tekemisestä ja raportin laatimisesta.

2 VIRTUAALISET- JA KEHITYSYMPÄRISTÖT

Luvussa käsitellään virtuaaliympäristöjen ominaisuuksia ja niiden mahdollisuuksia. Teknisen näkökulman lisäksi tarkastellaan sitä, miksi sovelluskehitysympäristö on oleellinen osa DevOPSin luomaa tehokasta työskentely-ympäristöä. Lopuksi paneudutaan opinnäytetyön kannalta tärkeimpiin ohjelmistoihin.

Tässä opinnäytetyössä sovelluskehitysympäristö on ohjelmistokokonaisuus. Kokonaisuutta voidaan käyttää kehitystyöhön, testaukseen sekä osittain tuotannossa. Tyypillisesti työskennellessään sovelluskehittäjä kirjoittaa koodia, tarvittaessa kääntää sen ja lopuksi suorittaa sen. Tämän jälkeen kehittäjä tutkii muutoksien vaikutukset, mahdolliset ongelmat ja siirtyy takaisin kirjoittamaan koodia. Työskentelyn kannalta on olennaista, että kaikki kehitystyön osa-alueet toimivat mahdollisimman sujuvasti. (Boxell 2019.)

2.1 DevOps

DevOps-ajattelun ydin on automaatiossa. Tavoitteena on, että ihmisten tekemää toistuvaa työtä siirretään koneen tehtäväksi. DevOps käsittää vaatimustenhallinnan, kehitysympäristöt, jatkuvan julkaisun mallin, testaukset, virtualisoinnin, monitoroinnin ja rajapinnat. (Devops s.a.)

DevOps korostaa, että organisaation ja tietotekniikan parempi suorituskyky on mahdollista saavuttaa ryhmän yhteisillä ponnisteluilla. DevOps -periaatteiden ansiosta tietotekniset työt toteutuvat nopeammin, pienemmillä virheillä ja nopeammalla virheiden korjauksella. (Brown ym. 2016.)

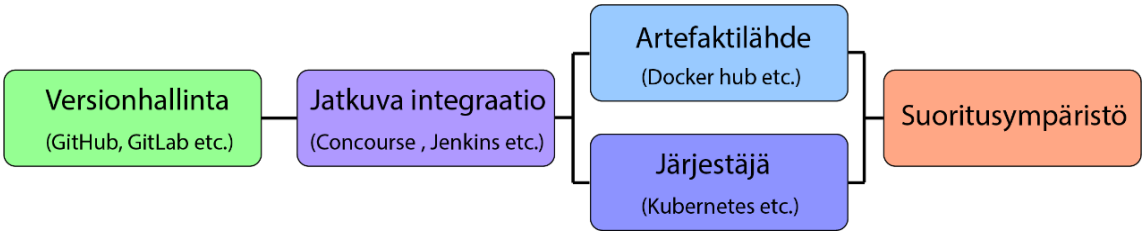
Kehitysympäristöjen luominen saattaa kestää kauan. Kehitysympäristön luomiseen käytetty aika on pois varsinaisesta kehitystyöstä ja siten kuluttavat kehittäjän aikaa ja energiaa. Ympäristö on kuitenkin välttämätön kehitystyön suorittamisen kannalta. (Efftinge 2019.)

Kehittäjillä saattaa olla työkoneessaan käytössä erilaisia toimintaympäristöjä kuten käyttöjärjestelmät tai tietokoneiden tekniset ominaisuudet. Projektin siirtäminen tällaisessa tilanteessa kehittäjältä toiselle aiheuttaa riskin kehitystyön jatkuvuudelle. Näin voi käydä, jos ensimmäinen kehittäjä sairastuu tai jostain muusta syystä ei voi jatkaa työtään. (Devops s.a.)

DevOps ajattelussa ympäristöt tulisi olla asennettavissa keskitetysti. Vaatimuksena on minimoida manuaalista konfigurointia. Ympäristön tulee olla yhteydessä versionhallintaan sekä automaattiseen testaukseen. Automaation ansiosta tehdyn työn osalta saadaan varmistettua kirjoitetun koodin laatu sekä ennalta sovittujen vaatimusten täyttäminen. (Devops s.a.)

Yksi DevOpsin kulmakivistä on jatkuva integraatio (continuous integration), jossa lähdekoodi on myös helposti kaikkien sovelluskehittäjien saatavilla. Järjestelmä yhdistää kehittäjien muutokset ja ilmoittaa mikäli niissä on ristiriitoja. Muutosten jälkeen versionhallinnassa voidaan toteuttaa julkaisu (release), joka rakentuu automaattisesti tuotannossa. (Cois 2015.)

Kuvassa 1 on kuvattu nykyaikainen tuotantoympäristö, joka tekee automaattisesti välivaiheet. Kun koodista luodaan uusi julkaisu, luodaan kokonaan uusi palvelin. Kun uusi palvelin on rakentunut oikein, ohjataan liikenne siihen ja suljetaan vanha palvelin. Samalla automaatio seuraa ja ilmoittaa tarvittaessa ilmenneistä ongelmista. (Cois 2015.)



Kuva 1. Esimerkki jatkuvan integraation tuotantoputkesta (mukaillen Grinten 2019)

DevOps liittyy myös pilviteknologioihin. Tilaajayrityksen, yksi Metatavun perustajista, teknologiajohtaja Antti Leppä (2020) kertoo keskustelussa, että yrityksillä oli aiemmin käytössään omia palvelimia. Niiden ylläpitäminen on kuitenkin ollut kallista ja aikaa vievää, joten tyypillisesti palvelintila on nykyisin vuokrattu konesaleista.

Tilaajayrityksellä on käytössään Amazon-pilvipalvelu, joka tarjoaa pilvipalvelimiaan kustannustehokkaaseen hintaan. Hinta koostuu käytetyn datan, liikenteen ja kuormituksen määrästä, lopullinen hinta kuitenkin selviää vasta kuukauden lopussa, kun lasku saapuu. Näihin eri tekijöihin vaikuttaa sivustojen

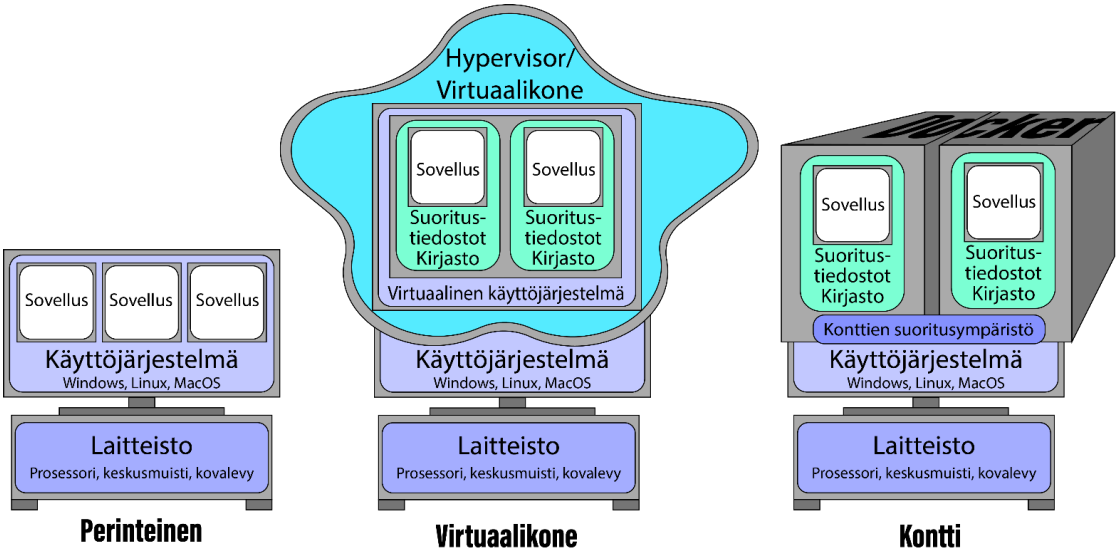
käyttöaste, sillä taustalla on automaattisesti skaalautuva järjestelmä, Kubernetes. Kubernetes-järjestelmällä on oikeus lisätä ja vähentää resursseja, tietyissä raameissa. Esimerkiksi tuotantokapasiteetin satakertaistaminen onnistuu minuuteissa. Tämän ansiosta kysyntään vastaaminen onnistuu aina ja palvelun laatu saadaan pidettyä korkeana. (Leppä 2020.)

2.2 Virtualisointi ja konttitekniikka

Virtuaalisella käyttöjärjestelmällä ei ole omaa fyysistä laitteistoa, mutta sen avulla voidaan tuottaa samoja ja samankaltaisia palveluita, kuin fyysisellä tietokoneella. Tuotannon kustannustehokkuus yleensä paranee siirryttäessä virtualisoituihin ratkaisuihin (Rathod ym. 2014, 9). Saman toteaa myös Miska Kaipiainen Kotilaisen (2017) kirjoittamassa artikkelissa. Virtuaaliympäristön pystyttäminen yhä uudelleen ja uudelleen on nopeaa ja helppoa. Mikäli ympäristöstä haluaa eroon, sen sammuttaminen ja lopettaminen on aivan yhtä nopeaa kuin käynnistäminen. Palvelin voi sijaita miltei missä päin maailmaa tahansa tai jopa useassa paikassa yhtä aikaa. (Heino 2010, 59–61.)

Suoritusympäristö voi olla perinteinen tietokone, virtuaalikone tai konttiteknologiaan (container) perustuva virtuaalikone. Kuvassa 2 näytetään perinteinen, fyysinen tietokone. Siinä laitteiston avulla suoritetaan käyttöjärjestelmää, joka toimii suoritusympäristönä ohjelmille ja sovelluksille. Toisena on virtuaalikone, jossa käynnistetään käyttöjärjestelmä perinteisen tietokoneen käyttöjärjestelmän sisällä ja siten yhdellä perinteisellä tietokoneella, voidaan suorittaa useita käyttöjärjestelmiä yhtä aikaa.

Kolmantena on konttiteknologia (container), joka tarkoittaa normaalia keveämpää virtuaalikonetta. Konttien virtualisointi tapahtuu käyttöjärjestelmätasolla, eikä siinä avata sisäkkäin useaa käyttöjärjestelmää. Useista lähteistä, kuten Kotilaisen (2017) artikkelissa tai Boxellin (2017) videoesityksessä vahvistetaan, että yleisimmäksi konttistandardiksi on muodostunut Docker. Dockerkontteja voi ajaa useita yhtä aikaa ja niiden yhteyksiä voi liittää toisiinsa tai ne voivat toimia erillään toisistaan. (Cito ym. 2017.)



Kuva 2. Erilaisia suoritusympäristöjä

Konttien ansiosta pieniä palveluja kannattaa suorittaa virtuaaliympäristöissä, sillä kontin ajaminen ei käytä niin paljoa resursseja, kuin kokonaisen virtuaalikoneen ajaminen käyttöjärjestelmineen. Tämä on osaltaan vauhdittanut mikropalveluiden yleistymistä. Artikkelissa kontteja sanotaan jopa merkittävämmäksi mullistukseksi tietotekniikalle ja tietojärjestelmille, kuin itse virtualisointia. (Kotilainen 2017.)

2.3 Docker

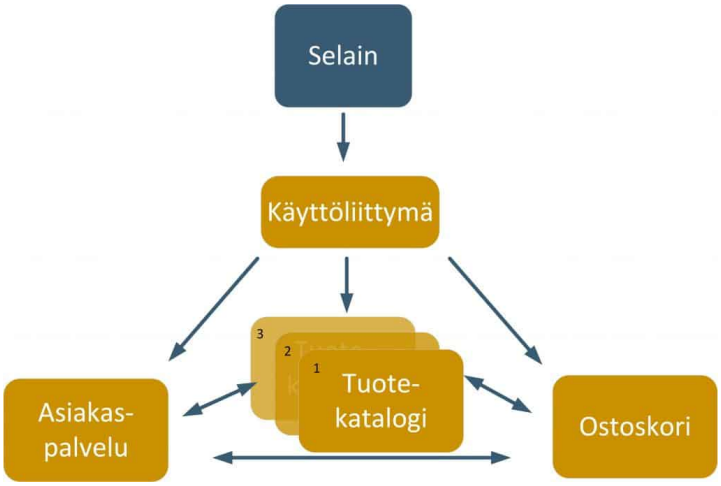
Kuvaus konttiteknologian eduista perinteiseen virtuaaliseen tietokoneeseen verrattuna löytyy Docker-dokumentaatiosta. Mikäli yhden gigatavun kokoinen Docker-kontin levykuva (container image) käynnistetään perinteisessä virtuaalikoneessa, tämä vie 1 gigatavun verran tilaa. Mikäli näitä koneita käynnistettäisiin 1 000 kappaletta, ne veisivät yli 1 000 gigatavua kovalevytilaa. Dockerissa suoritettuna, 1 000 yllä kuvattua konetta veisivät vain vähän yli 1 gigatavun verran kovalevytilaa yhteensä, sillä Docker käyttää kerrostettua tiedostonhallintaa (AuFS). (Cochrane 2013.)

Dockerin käynnistyminen vie murto-osan siitä ajasta mitä kuluu perinteisen virtuaalikoneen käynnistymiseen. Kun kontti rakennetaan (build), sille voidaan antaa ympäristömuuttujien arvoja. Konttien ominaispiirre on, että kun se rakennetaan ja käynnistetään, sen sisällä on aina sama data. (Cochrane 2013.)

Tätä hyödynnetään esimerkiksi testauksessa, kun halutaan, että tietokanta on oikeassa tilassa aina kun testi ajetaan. Tämä saavutetaan siten, että luodaan tietokannalle kontin halutusta tilasta (snapshot) uusi levykuva. Näin tietokantoja voidaan käynnistää juuri niin paljon kuin tarvitaan ja kaikki niistä on samassa tilassa. (Cochrane 2013.)

Docker on konttien suoritusympäristö. Dockerin suosiosta kertoo se, että tähän mennessä Docker Hubissa on 6 miljoonaa levykuvaa ja 5 miljoonaa käyttäjää. Docker Hub on sen kehittäjän ylläpitämä varasto (repository), johon käyttäjät voivat lisätä omia Docker levykuviaan. Docker Hubista löytyy useita tunnettuja yrityksiä, jotka päivittävät ja ylläpitävät aktiivisesti omia tuotteitaan, kuten Oracle, IBM ja Cisco Systems. (Docker 2020b.)

Kontit ovat olleet merkittävä osa mikropalveluiden yleistymisessä. Yhden monoliittisen palvelun sijaan palvelut voidaan ripotella yksittäisiin kontteihin. Kuvassa 3 nähdään, miten mikropalveluista koostuvassa järjestelmässä voidaan lisätä resursseja yhdelle osa-alueelle sen sijaan, että lisättäisiin koko järjestelmän resursseja. (Wallenius s.a.)



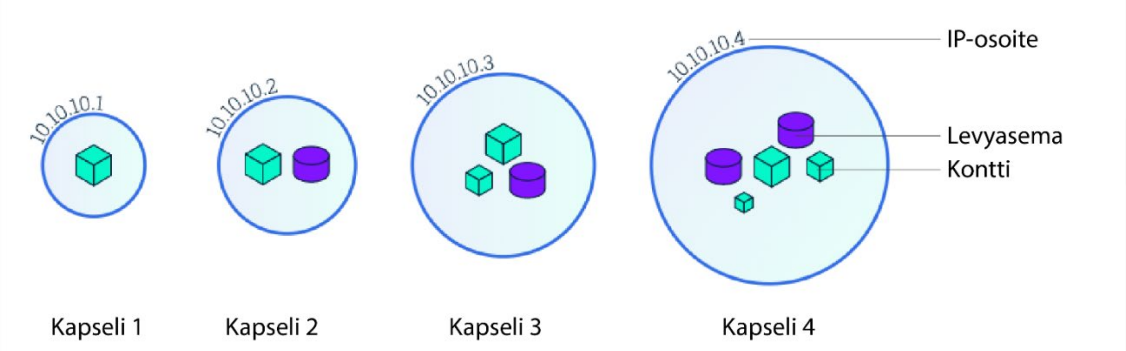
Kuva 3. Mikäli tuotekatalogi osa-alue vaatii lisää resursseja, voidaan lisätä vain sen suorituskykyä (Wallenius s.a.)

Docker-kontteja voi suorittaa paikallisesti tai verkkopalvelussa. Useiden konttien ylläpitämiseen on olemassa ratkaisuja, jotka keräävät määritellyt kontit yhteen ja muodostaa niistä kokonaisuuden. Yksi käytetyimpiä on Kubernetes. (Docker 2020a.)

2.4 Kubernetes

Kubernetes on kokonainen webkokonaisuuksien ylläpitoratkaisu, josta löytyy kaikki toiminnot sivuston ylläpitämiseksi. Ratkaisua voi käyttää mittakaavaltaan minkä kokoisissa sivustoissa tahansa (Leppä 2020). Uuden palvelimen tekeminen vie vain muutamia minuutteja aikaa. Kubernetes kokonaisuus koostuu pienistä palasista ja sen avulla ylläpidetään konttikokonaisuuksia ja järjestelmiä. Kubernetes ohjaa liikennettä ja huolehtii resurssien riittävyydestä. (Kubernetes documentation 2020a.)

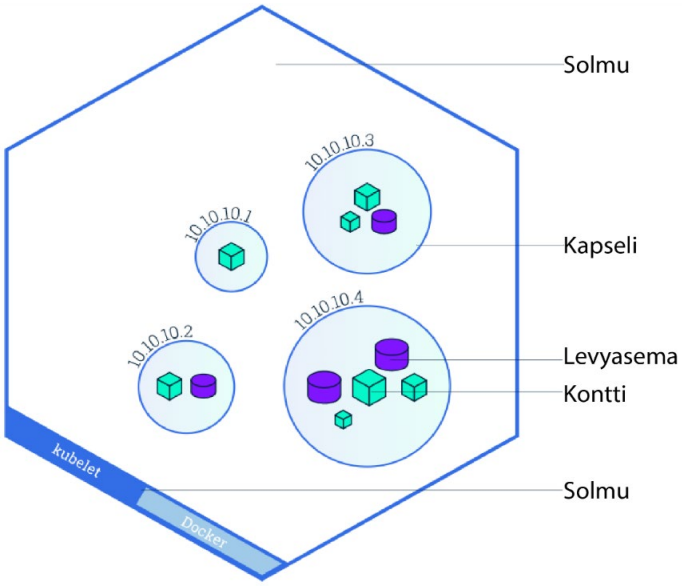
Kuvassa 4 kapseli (pod) sisältää kontin tai useita kontteja. Kapselissa voi olla oma levyasema, mutta se voi käyttää myös kapselin ulkopuolista ratkaisua. Kapselit on suunniteltu kertakäyttöisiksi. Niitä tuhoutuu ja rakentuu, joko tasaisin väliajoin tai aina kun muutoksia tehdään. Kun uudet kapselit ovat rakentuneet, Kubernetes ohjaa liikenteen niihin ja tuhoaa vanhat kapselit. Näin sivuston päivitys voidaan toteuttaa, ilman että sen käyttö häiriintyy. Jokaisella kapselilla on oma IP-osoite Kuberneteksen sisällä, joiden liikennettä Kubernetes hallitsee. (Kubernetes documentation 2019a.)



Kuva 4. Kapseleiden sisältö vaihtelee, riippuen mitä palveluita on käynnistetty (mukaillen Kubernetes documentation 2019b)

Laitteisto Kubernetes-järjestelmässä voidaan jakaa karkeasti yksittäisiin solmuihin (node) ja solmuista koostuvaan parveen (cluster). Kubernetes-järjestelmässä yhtä solmua voidaan ajatella yksittäisenä tietokoneena. Suoranaisesti kyse ei ole virtuaalikoneesta, mutta selkeyden vuoksi, sitä voidaan ajatella prosessoritehona ja keskusmuistimääränä. Yhdelle solmulle varataan tietty

määrä laskentatehoa ja muistikapasiteettia. Kuvassa 5 ilmenee tarkemmin yksittäisen solmun rakenne ja sisältö. (Kubernetes documentation 2020d.)



Kuva 5. Yksittäinen Kubernetes solmu, eli ”node” (mukaillen Kubernetes documentation 2019b)

Seuraava taso on parvi. Parvi koostuu solmuista, mutta sen sijaan että Kubernetes ajaisi jokaista solmua erikseen, on parvessa yksi laitteisto, joka määräytyy solmujen määrän mukaan. Parvessa on laskentatehoa ja muistia tarpeeksi jokaisen solmun käyttöön. Solmuja voidaan lisätä ja poistaa, mikä muuttaa parven laskentatehon ja muistin kapasiteettia. (Sanche 2018.)

Koska solmuja luodaan ja poistetaan eri yhteyksissä, niihin ei kannata tallentaa sellaista tietoa, joka pitäisi säilyttää. Tämän vuoksi täytyy olla pysyviä asemia (persistent volume), joiden dataa käytetään ohjelmien suorittamisen yhteydessä. Pysyvä asema liitetään parveen, eikä sitä yhdistetä mihinkään tiettyyn solmuun. Parvessa olevat solmut pystyvät kuitenkin hyödyntämään sen dataa. (Sanche 2018; Kubernetes documentation 2020f.)

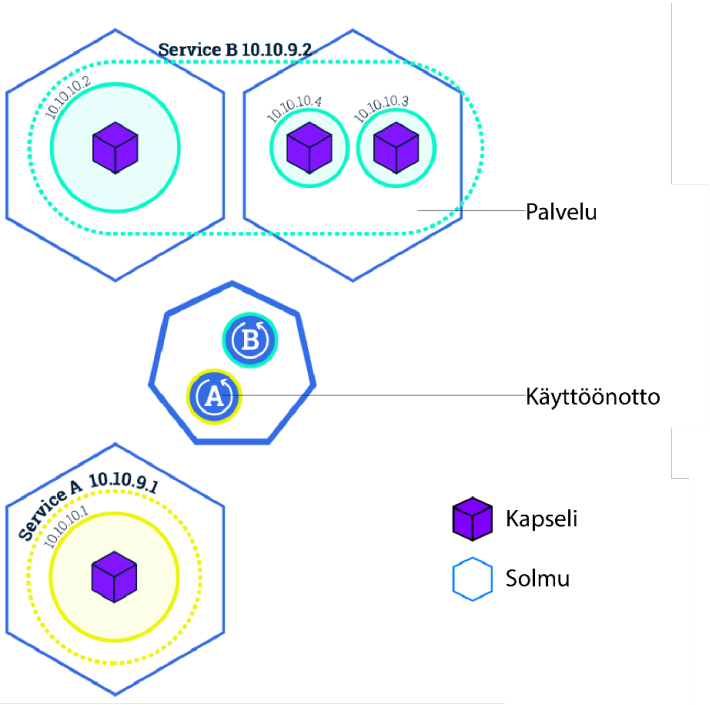
Esimerkiksi tilanteessa, jossa halutaan säilyttää tallennetut tiedostot, mutta päivittää vain yhtä konttikokonaisuutta, onnistuu se luomalla uusi kapseli, joka sisältää uuden version halutusta kontista. Koska kapseleita voidaan luoda tarvittaessa lisää, antaa tämä mahdollisuuden skaalaukselle. Mikäli palvelun kysyntä nousee ja se tarvitsee lisää kapasiteettia, voidaan sitä luoda tekemällä

lisää kapseleita tai solmuja, eli ikään kuin ottaa kopioita palvelusta, jolloin sillä on enemmän resursseja käytössään. (Kubernetes documentation 2019a.)

Nykyaikaiset pilviratkaisut mahdollistavat lähes rajattoman laskentatehon. Laskentatehoa voidaan ostaa hyvin yksityiskohtaisilla vaatimuksilla. On tavallista, että kapseleita on lähtökohtaisesti useampia, jonka ansiosta voidaan jakaa kuormaa ja parannetaan järjestelmän virheensietokykyä. (Sanche 2018.)

Kapselin koko määrittää tarpeellisen resurssin määrän, joten kannattaa välttää liian suuria kapseleita. Kun kapseleita tehdään, resursseja varataan aina sama määrä kapselia kohti. Mikäli kapselissa on ylimääräistä dataa, moninkertaistuu turhan resurssin määrän ja sitä myötä turhia kuluja. (Sanche 2018.)

Käyttöönotto (deployment) tarkoittaa tasoa, joka ylläpitää kapseleita. Kuva 6 esittää käyttöönottoa, jossa on kaksi palvelua, kolme solmua ja neljä kapselia. Palvelu B sisältää kaksi solmua ja kolme kapselia, palvelu A sisältää yhden solmu ja yhden kapselin. Käyttöönotossa määritellään käynnissä olevien kapseleiden lukumäärä. Käyttöönotto ylläpitää kapseleiden määrän haluttuna automaattisesti. Mikäli jokin kapseleista sammuu, käyttöönotto luo sinne uuden kapselin sammuneen tilalle. (Kubernetes documentation 2020b.)



Kuva 6. Kubernetes, ”master”-solmu keskellä (mukaillen Kubernetes documentation 2020g)

Sisääntulo (ingress) käsittelee ulkopuolelta tulevia yhteyksiä. Lähtökohtaisesti Kubernetes estää ulkopuolelta tulevan liikenteen ja eristää kapselit toisistaan. Usein toivomuksena on kuitenkin käyttää kapselin sisällä toimivaa palvelua. Tälle täytyy luoda oma erillinen kanava, jonka kautta saadaan luotua yhteys kapselin sisälle. Nämä yhteydet hoitaa sisääntulo. (Kubernetes documentation 2020c.)

2.5 Kehitysympäristön työkaluja

Opinnäytetyön toteutus osassa projektien sovelluskehitystyön ratkaisumallinnuksen tekemistä varten on käytetty useita eri ohjelmistoja, joista tärkeimmät ohjelmat on kuvailtu seuraavaksi. Ohjelmille saattaa olla muita vastineita ja vaihtoehtoja. Olen mahdollisuuksien mukaan pyrkinyt käsittelemään myös ohjelmien vastineita.

Kubectl-ohjelmalla on sama ohjelmistokehittäjä kuin Kuberneteksella. Siinä komennot annetaan komentoriville suoraan tai ne voi määritellä ohjelman suoritettavaksi. Kubectl on Kubernetes komentorivityökalu. Sen avulla tarkastellaan käynnissä olevia toimintoja, kuten kapseleita, nimiavaruuksia (namespace), palveluita (service) ja julkaisuja (deploy). (Kubernetes documentation 2020e.)

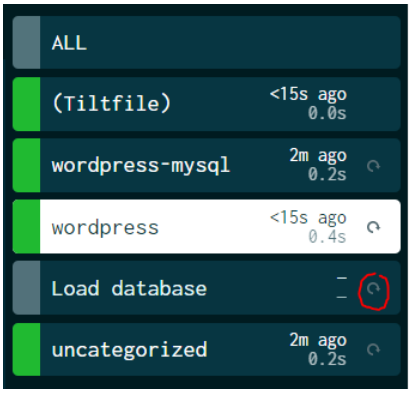
Tarkastelun lisäksi Kubectl mahdollistaa komentojen suorittamisen Kuberneteksessa käynnissä oleviin kontteihin. Opinnäytetyön käytännön osassa käytän bash-skriptiä kontin sisälle, jonne kopioidaan sql-tiedosto ja viedään sen sisältö tietokantaan. Skripti suoritetaan konttiin käyttämällä Kubectl sh -komentoa. (Kubernetes documentation 2020e.)

Docker-compose on työkalu, jonka avulla Docker kontteja pystyy käynnistämään useita kerrallaan. Opinnäytetyössä kuvataan miten docker-composen avulla saadaan kehitysympäristö käynnistettyä. Tämä ratkaisu toimii yhden abstraktiotason alempana verrattuna Kubernetekseen, sillä kontit eivät ole Kubernetes klusterin sisällä. Ratkaisu on otettu työhön koska tällä hetkellä kehitysympäristöjen luomiseen käytetään usein docker-compose -ohjelmaa. Lisäksi mekanismit ja määritystiedostot ovat samankaltaisia kuin käytettäessä kehitysympäristötyökaluja Kubernetes-ympäristössä.

Docker-composessa määritellään docker-compose.yaml-tiedostoon halutut paikalliset Dockerimage-tiedostot tai Docker Hub:sta käytettävät Dockerimaget. Samalla määritellään konttien väliset muuttujat sekä ympäristömuuttujat. Tarvittaessa voidaan kertoa mihin portteihin liikenne halutaan ohjata. Porttiohjauksen ansiosta sivusto saadaan näkymään selaimessa ja palvelut keskustelemaan keskenään. Docker-compose ei juurikaan sisällä graafista toimintaympäristöä, jonka vuoksi virheenkorjaus voi olla hankalaa. Mikäli konttien rakentaminen onnistuu ongelmitta, ohjelma rakentaa kokonaisuuden ja ohjaa verkkoliikenteen.

Kompose on työkalu, jolla voidaan kääntää Docker-compose.yaml-tiedostot Kubernetes manifest-määritystiedostoiksi. Se kääntää docker-compose.yaml -tiedoston sisällön tarvittaessa useammaksi määritystiedostoksi, jonka tavoitteena on luoda sama kokonaisuus Kubernetes solmun sisälle. Useimmiten Kompose tekee hyvää työtä, mutta tiedostoihin voi joutua tekemään muutoksia, jotta lopullinen tavoite saavutetaan. Komposen avulla tiedostot voidaan käynnistää komennolla ”kompose up”, eli kompose sisältää itsessään käynnistämisen ja ajamisen. Sitä voidaan käyttää Kubernetes kehitysympäristötyökaluna, mutta tiedon esitys on siinä melko rajallinen eikä se tarjoa juurikaan tietoa ajon aikana.

Tilt-ohjelma toimii kehitysympäristön pystyttäjänä ja ajoympäristönä. Se päivittää muutetut tiedostot Kubernetes-järjestelmän sisälle ja päivittää näkymää kehittäjälle. Tilt on komentoriviohjelma, jolle määritellään sen tehtävät tiltfile-tiedostoon. Tiltin jaoteltu näkymä on kuvassa 7. Näkymään voidaan luoda myös mukautettuja resursseja.



Kuva 7. Tilt esittää rakennetun kokonaisuuden välilehtinä, punaisella ympyröitynä mukautetun skriptin suorituspainike

Tilt voidaan määritellä käyttämään docker-compose tiedostoa ja suorittamaan kontteja ilman Kubernetes pakettia. Näin tiltfile voidaan määritellä yksilöllisesti jokaiseen projektiin vastaamaan sen vaatimuksia ja ominaisuuksia. Tiltin ympäristö käynnistetään aina samalla tavalla. Komento ”tilt up” suorittaa tiltfileen määritellyt toimenpiteet. Näin sovelluskehittäjän prosessi säilyy samanlaisena.

3 KEHITYSYMPÄRISTÖT KÄYTÄNNÖSSÄ

Luvussa kerrotaan projektien sovelluskehitystyön ratkaisumallinnuksen toteutuksesta. Aluksi perustellaan testiympäristön ominaisuuksien valinta. Opinnäytetyössä esitellään kaksi eri tapaa luoda sama ympäristö. Ensin kuvataan docker-composen avulla luotu sovelluskehitysympäristö ja sen jälkeen toteutetaan se paikallisessa Kubernetes ympäristössä. Lopuksi esitellään vaihe vaiheelta Kubernetes kehitysympäristön luominen.

3.1 Järjestelmän kuvaus ja vaatimukset

Kehitysympäristö luodaan websivustolle. Tässä tapauksessa sivuston pohjana toimii Wordpress, joka vaatii toimiakseen tietokannan ja php-tuen. Websivu pohjautuu toimeksiantajan aikaisempaan projektiin. Sivustolle luodaan kehitysympäristö, jota muokkaamalla vastaava kehitysympäristö voidaan luoda myös muihin projekteihin. Wordpressille on erillinen teema, johon tehtävät muokkaukset näkyvät selaimessa, ilman erillistä palvelimen tai konttien uudelleenkäynnistämistä.

Kehitysympäristö koostuu useasta eri osasta. Osat latautuvat automaattisesti konttien rakentumisen yhteydessä. Dockerin avulla voidaan hyödyntää muiden toteuttamia valmiita kontteja. Tietokannasta ja Wordpress-paketista löytyy valmiit kontit Docker Hub-palvelusta. Windowsissa käytetään ohjelmistoa nimeltä Docker for Windows. Ohjelmassa tulee mukana kaikki tarvittava konttien ajamiseen ja se on helposti asennettava asennuspaketti. Linuxissa asentaminen tehdään usean vaiheen kautta, ohjeet löytyvät kirjoitushetkellä Dockerin omasta dokumentaatiosta.

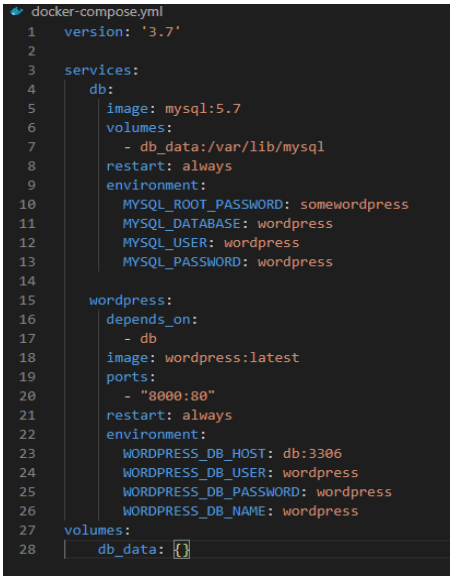
Docker rakentaa kontin asetuksissa viitatun levykuvan mukaisesti. Tyypillisesti viittaus on muotoa “wordpress:latest”, jossa kerrotaan mitä levykuvaa halutaan käyttää ja kaksoispisteen jälkeen määritellään sen versio. Kun kokonaisuutta rakennetaan, Docker lataa ja käynnistää kontin.

Konttien avulla voidaan helposti vaihtaa esimerkiksi versiota projektien välillä. Docker Hubista löytyy julkaisutiedot kaikista sinne ladatuista levykuvista. Ympäristö on helppo jakaa, sillä suurimmat tiedostopaketit eivät ole paikallisesti tietokoneella vaan paketit ladataan ennen rakentamista. Levykuvat latautuvat tietokoneen paikalliseen rekisteriin. Docker käyttää lähtökohtaisesti paikallista rekisteriä, jolloin levykuvaa ei aina tarvitse ladata uudestaan.

3.2 Kehitysympäristön luominen Docker-compose ohjelmalla

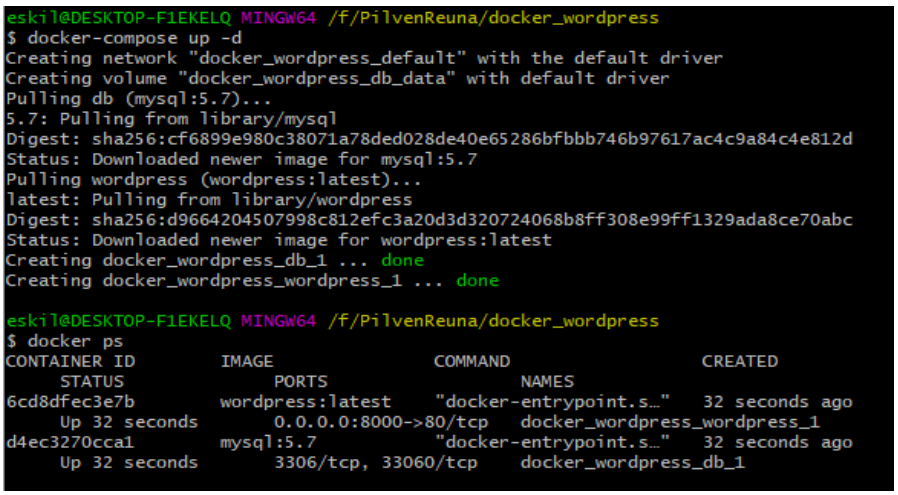
Docker-compose on samalta tekijältä kuin Docker-paketti ja se sisältyy Docker for Windows-kokonaisuuteen. Linuxilla docker-compose täytyy asentaa erikseen, varsinaisen Dockerin asennuksen jälkeen. Docker-composessa voi määritellä usean kontin ja ajaa niitä samaan aikaan. Konteille voidaan määritellä myös parametrejä. Esimerkkinä tietokannan käyttäjätunnus ja salasana, joiden ansiosta kontit pystyvät vaihtamaan tietoja keskenään. Docker-compose -työkalussa tarvittavat tiedot määritellään YAML-tiedostoon. Sovellus suorittaa komennolla ”*docker-compose up*” ja lopettaa suorittamisen komennolla ”docker-compose down”.

Kehitysympäristön luomisen pohjana hyödynnetään kuvassa 8 näkyvää, docker-compose dokumentaatiosta haettua esimerkkiä. Tiedostossa ”version” kertoo docker-compose-työkalusta käytettävän versionumeron. Palveluina tässä esimerkissä on Wordpress ja MySQL-tietokanta. Palveluille tehdään määrittelyt ja asetetaan muuttujien arvot halutuiksi. Kun ympäristömuuttujat vastaavat toisiaan Wordpress saa yhteyden tietokantaan.



Kuva 8. Docker-compose.yaml -tiedosto, Visual Studio Code editorissa

Komennolla ”docker-compose up” käynnistetään yaml -tiedostossa määritellyt kontit. Kuvassa 9 käytetään parametria ”-d”, joka viittaa sanaan ”detach”. Konenolla kontit asetetaan toimimaan taustalla ja sen ansiosta voidaan jatkaa saman komentorivin käyttöä. Kuvassa 9 nähdään kuinka järjestelmä lataa mysql:stä version 5.7 ja Wordpressistä viimeisimmän. Sen jälkeen se rakentaa ne yllä kuvattujen määritysten mukaisesti.



Kuva 9. Docker Composen käynnistys, alapuolella ”ps” -komennolla nähdään käynnissä olevat kontit

Mikäli kontin sisään rakennetun sivuston koodiin tehdään muutoksia, ne eivät tallennu mihinkään. Kun kontin ajaminen lopetetaan ja se korvataan uudella, muutokset katoavat. Kehitystyössä kehittäjällä tulee olla tietokoneellaan erilli-

nen sisältö paikallisessa kansiossa, joka yhdistetään konttiin rakennusvaiheessa. Kun kaikki Docker-kontit pakotetaan kiinni, tuhoutuu samalla myös kontin kautta rakennettu tietokanta. Tämän vuoksi massakomentoja täytyy ajaa harkiten. Kriittisissä vaiheissa kannattaa ottaa tietokannasta paikallinen kopio (sql dump).

Wordpress-ympäristö on valmis otettavaksi käyttöön ja se toimii tietokannan kanssa yhdessä. Komennolla ”*docker-compose down*” sammutetaan kontit- ja verkko pois käytöstä. Tällöin tietokanta säilyy. Lisäämällä parametrin ”--volumes", eli ”*docker-compose down --volumes*” komento hävittää yllämainittujen lisäksi myös tietokannan.

Testiympäristö vaatii vielä teemakansion yhdistämisen. Tällä hetkellä Wordpress toimii vain omalla teemallaan. Docker-compose.yaml -määrittelytiedostoon lisätään kaksi kuvassa 10 näkyvää riviä. Siinä määritellään yhdistettävät paikallisen koneen- ja kontin tiedostosijainnit. Kuvassa 10 nähdään ensin määriteltävän paikallisen kansion sijainti. Erotin merkkinä toimii kaksoispiste. Kaksoispisteen perässä on kontin sisältämän teemakansion sijainti.

Kuva 10. Lisäämällä teema kansio Wordpressin vastaavaan, yhdistetään teema Wordpress kontin sisälle

Docker-compose on kevyt tapa luoda sovelluskehitysympäristö konttien avulla. Seuraavaksi luomme kehitysympäristön paikallisessa Kubernetes-ympäristössä.

3.3 Paikallinen Kubernetes-ympäristö

Opinnäytetyössä käytetään Minikube-ohjelmaa paikallisen Kuberneteksen luomiseen. Minikuben on luonut sama organisaatio, joka kehittää ja ylläpitää Kubernetes-pakettia. Kirjoitushetkellä Minikuben pystyy asentamaan yleisimmille käyttöjärjestelmille.

Docker for Windows -kokonaisuudesta löytyy myös oma Kubernetes. Sitä käytettäessä kohdattiin kuitenkin vikatilanteita. Käytössä ohjelma vei kaiken keskusmuistin, joka teki työskentelystä tahmeaa. Kannattaa ottaa huomioon Minikuben asennuksessa, että ohjelmat saattavat aiheuttaa keskenään konfliktin. Kyseessä on eri yritysten kehittämät tuotteet paikallisen Kuberneteksen pystytykseen.

Minikube on joustava ja se tarjoaa kattavan käyttöjärjestelmätuen. Sen pystyy asentamaan Windowsille, MacOS:lle ja Linuxille. Useiden virtualisointivaihtoehtojen suhteen se tarjoaa hyvän vaihtoehdon Docker for Windowsille. Lisäksi on olemassa muita ohjelmia, jotka lähestyvät samaa asiaa hieman eri näkökulmalta. Linux-käyttöjärjestelmälle on saatavilla microk8s, jonka tuore versio tarjoaa mahdollisuuden käyttää Kubernetesta ilman virtuaalikonetta.

Toimiakseen Minikube tarvitsee virtuaalikoneohjelmiston. Se vaatii VirtualBox-ohjelmiston tai Windowsin oman Hyper-V:n. Hyper-V:n saattaa joutua aktivoimaan erikseen käyttöjärjestelmän asetuksista. Muilla käyttöjärjestelmillä vaaditaan joko VirtualBox tai vastaava ohjelmisto. Minikuben asennuksessa Windows-käyttöjärjestelmään käytettiin Chocolatey -ohjelmaa, jota suositeltiin asennussivustolla. Kuberneteksen lisäksi täytyy asentaa Kubernetes komentorivityökalu kubectl.

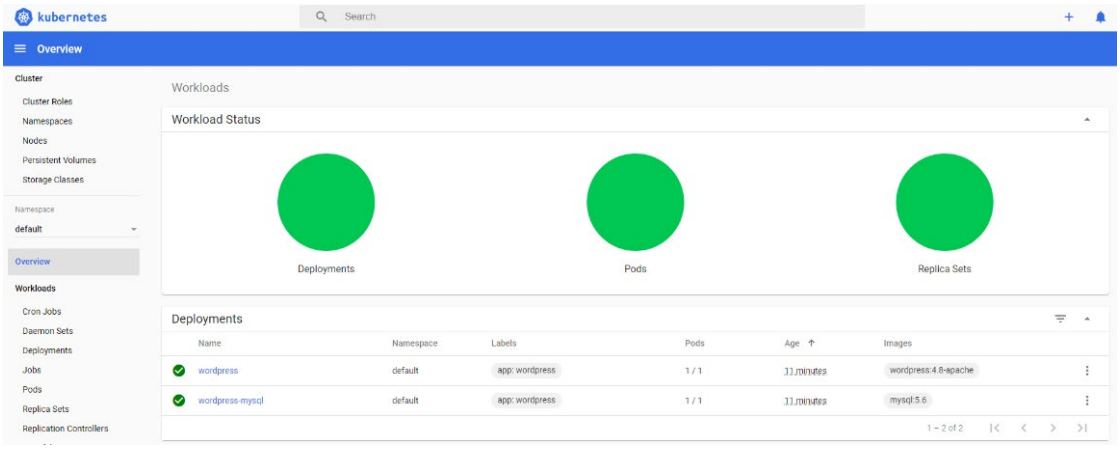
Ohjelma käynnistetään komennolla “minikube start”. Kuvassa 11 nähdään Minikuben käynnistäminen. Komentoriviohjelmana on käytetty Powershelliä, jota täytyy suorittaa järjestelmänvalvojan oikeuksilla. Docker for Windows -paketin Kubernetes ei vaadi toimiakseen järjestelmänvalvojan oikeuksia komentorivissä.

```
* minikube v1.9.2 on Microsoft Windows 10 Pro 10.0.19041 Build 19041
* Using the hyperv driver based on existing profile
* Starting control plane node m01 in cluster minikube
* Updating the running hyperv "minikube" VM ...
* Preparing Kubernetes v1.18.0 on Docker 19.03.8 ...
E0414 13:57:03.863018   16172 kubeadm.go:331] Overriding stale ClientConfig host
 https://172.20.36.17:8443 with https://172.26.42.40:8443
* Enabling addons: dashboard, default-storageclass, storage-provisioner
* Done! kubectl is now configured to use "minikube"
```

Kuva 11. Minikube käynnistyi ongelmitta

Kehitysympäristömme tarvitsee konttien suoritusympäristön, tässä työssä se on Docker. Muita vaihtoehtoja voisi esimerkiksi olla CRI-O tai containerd. Mikään näistä tekniikoista ei ole sidottu toisiinsa. Dockerin dokumentaatio on kattava ja sille löytyy paljon esimerkkejä keskustelupalstoilta.

Kun Minikube on käynnistetty, sen toimintaa voi testata käynnistämällä sen hallintapaneelin. Hallintapaneeli aukeaa komennolla “minikube dashboard”. Kuvassa 12 näkyvä graafinen hallintapaneeli avautuu selaimessa. Hallintapaneelin avulla voi tarkastella käynnissä olevia kapseleita ja niiden statusta. Kubernetes osaa suorittaa palveluita ja käyttöönottoja omissa nimiavaruuksissaan (namespace), joten siinä voi olla päällä useita kokonaisuuksia. Hallintapaneelin vuoksi Minikube käynnistää Kubernetes-klusteriin oman kapselin, palvelun ja käyttöönoton. Hallintapaneelin osat sulkeutuvat automaattisesti komennolla CTRL + C.



Kuva 12. Minikuben hallintapaneeli

Kubernetes ei sisällä omaa sovelluskehitysympäristöä vaan se tarvitsee avukseen erillisen työkalun. Työkalun avulla voidaan yhdistää paikallisia tiedostoja Kubernetes-ympäristöön. Työkaluja on esimerkiksi Kompose, Tilt.dev, Draft ja Skaffold. Kompose on näistä yksinkertaisin ja sen avulla voidaan luoda Docker-compose.yaml -tiedostoista Kubernetes määrittelytiedostot. Tilt.dev, Draft ja Skaffold ovat kattavampia kokonaisuuksia.

3.4 Paikallisen Kubernetes-kehitysympäristön työkalut ja rakenne

Kehitysympäristön luomisessa suurilla toimijoilla kuten Microsoftilla (Draft) ja Googlella (Skaffold) on omat työkalut. Tähän opinnäytetyöhön valikoitu työkaluksi Windmill Engineeringin Tilt. Kaikki nämä työkalut ovat pohjimmiltaan Kubernetes ympäristössä tapahtuvaan kehitystyöhön tarkoitettuja työkaluja. Tilt tarjoaa selkeyttävää visuaalista sisältöä ja sillä on aktiivinen kehitystiimi. Microsoftin Draft-ohjelmisto ei ole saanut uutta päivitystä yli vuoteen, joten sitä ei edes harkittu vaihtoehdoksi. Skaffold tarjoaa pitkän kehitysajan, vakaan pohjan ja taustaltaan suuren yrityksen. Tiltin suhteen Skaffoldin perustoiminnoissa ei kauheasti ole eroa. Todettakoon mikäli Tilt-ohjelman kehitys jostain syystä lakkaisi, tiltfile-määritystiedosto on suhteellisen helppo muuttaa Skaffold-määritykseksi.

Tilt on komentoriviohjelma ja sille toteutetaan oma määritystiedosto. Tiedostoihin määritellään mitä käynnistetään ja millaisia resursseja luodaan. Kuvassa 13 tilfile-tiedostoon määritellään kehitysympäristön tarvitsemat tiedot. Tilt käynnistetään samassa kansiossa komennolla ‘tilt up’. Tilt ohjelma alkaa käymään läpi määritystiedoston sisältöä. Mikäli käytössä on Minikube, täytyy komento suorittaa järjestelmänvalvojan oikeuksilla avatussa komentorivissä.

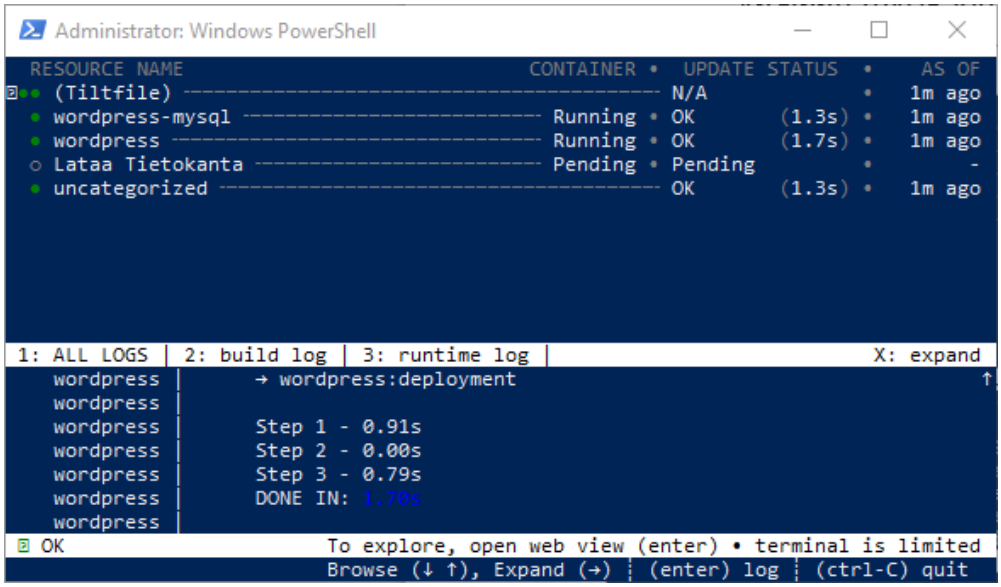
Tiltfile

```
1  theme='wordpress-musicfairytales-theme'
2
3  k8s_yaml(['mysql-deployment.yaml', 'wordpress-deployment.yaml'])
4
5  docker_build('wordpress-local-image', '.', dockerfile='./Dockerfile',
6      live_update=[
7          sync('./wordpress-musicfairytales-theme/',
8          '/var/www/html/wp-content/themes/wordpress-musicfairytales-theme/')
9          ])
10
11 k8s_resource('wordpress-mysql', port_forwards=3306)
12 k8s_resource('wordpress', resource_deps=['wordpress-mysql'], port_forwards=8080)
13
14 local_resource('Lataa Tietokanta', resource_deps=['wordpress'],
15     cmd='mysqlPod=`kubectl get pods -l tier=mysql -o=jsonpath=\'{.items[0].metadata.name}\'` &&
16     kubectl exec -it "$mysqlPod" -n default -- mysql -u root -ppassword wordpress < ./devDump.sql',
17     auto_init=False, trigger_mode=TRIGGER_MODE_MANUAL)
```

Kuva 13. Tiltfile-tiedoston sisältö

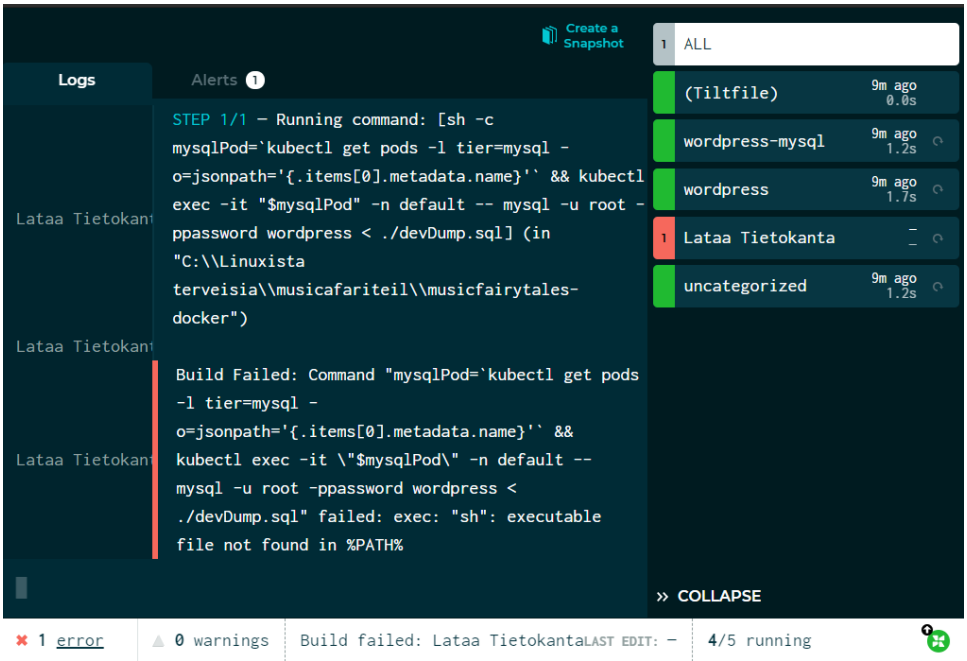
Tilt tarjoaa graafisen käyttöliittymän myös komentorivillä, joka on näkyvissä kuvassa 14. Tilt-komentorivi sovellus ottaa vastaan myös Vim-komentoja. Windows-käyttöjärjestelmässä Tilt ei aina toimi odotetulla tavalla. Välillä sovel-

lus ei toteuta rakennusvaihetta vaan vaatii käyttäjältä tiltfilen uudelleen tallennuksen. Tiltfilen tallentaminen herättää tiedostoissa tapahtuvia muutoksia seuraavan mekanismin (watcher). Kuvassa 14 Tilt on onnistuneesti rakentanut ja toimittanut kontit. Tässä työssä kehitysympäristö luodaan kahdesta dockerimage-tiedostosta. Wordpressin Dockerimage voisi olla sama tiedosto mitä käytetään tuotannossa.



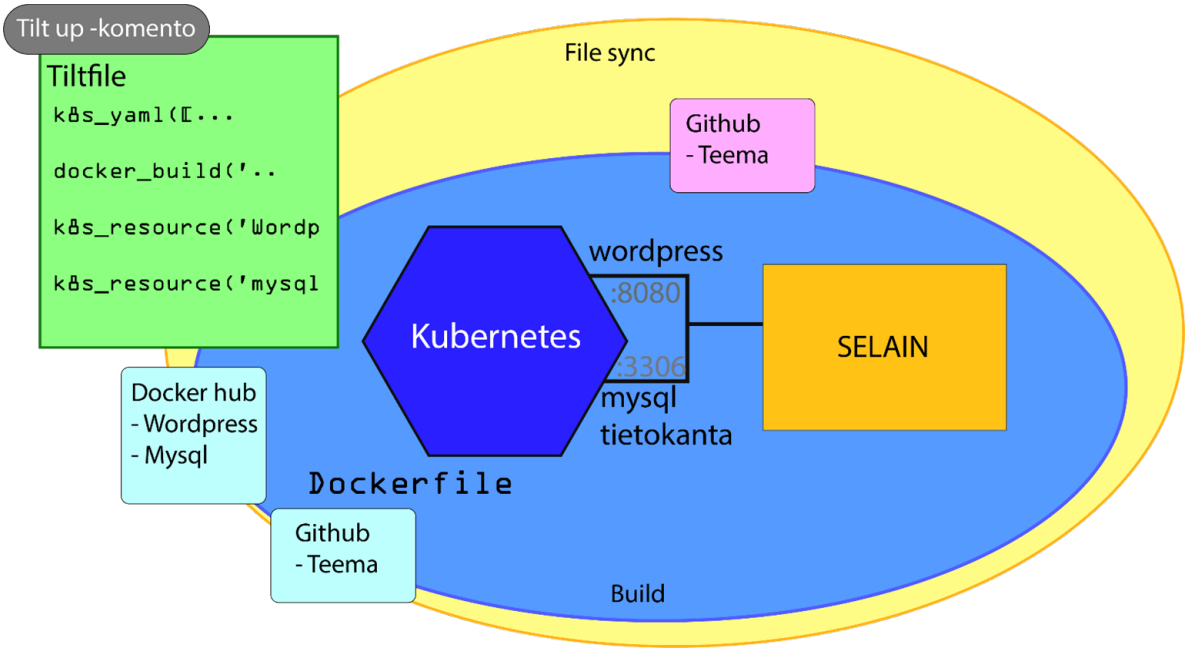
Kuva 14. Tilt komentorivikäyttöliittymä

Tilt näyttää reaaliajassa konttien rakentaminen etenemisen ja hälyttää virheistä. Kuvan 15 kohdassa “Lataa Tietokanta” näkyy virheilmoitus, koska Windows versiossa ei pysty ajamaan Bash skriptejä. Asia sai vahvistuksen Tilt.dev Slack -keskustelualustalla. Keskusteluni pohjalta GitHubiin kirjattiin issue, jonka ansiosta ongelma on nyt korjattu. Silti skriptiä ei voi käyttää sellaisenaan, vaan sen toimivuus Windows käyttöjärjestelmällä vaatisi muutoksia. Tilt on alpha-statuksella Windowsin suhteen ja sitä ei kehitetä kirjoitushetkellä aktiivisesti Windows-käyttöjärjestelmälle.



Kuva 15. Tilt web käyttöliittymä

Kuvassa 16 on graafisesti esitetty kehitysympäristön rakenne. Siinä kuvataan myös kehitettävän komponentin automaattinen tiedoston päivitys Kubernetek-sen sisälle. Järjestelmässä on käytännössä kaksi osuutta. Toisessa sivusto rakennetaan (build) ja toisessa jatkuvasti aktiivinen file sync -toiminto seuraa muutoksia tiedostoissa ja tarjoilee niitä Kuberneteksen sisälle.



Kuva 16. Kehitysympäristön rakenne

Toteutus mahdollistaa projektin pitämisen tuotannon kaltaisena. Wordpressistä tai Mysql-tietokannasta voi valita myös eri versioita. Tiltfile täytyy muokata jokaisessa projektissa vastaamaan projektin vaatimuksia. Periaate pysyy kuitenkin samana.

Tilt sisältää oman ongelmajakamiseen tarkoitetun ominaisuuden. Sen snapshot-ominaisuudella Tilt kopioi tilanteen ja lataa sen pilveen. Tiltin luoman linkin kautta pääsee tarkastelemaan vikatilannetta, joka esitellään kuin se olisi tapahtunut omalla koneella.

3.5 Sovelluskehitysympäristön luominen vaihe vaiheelta

Tässä luvussa esitellään vaihe vaiheelta, mitä ohjelmia ja komentoja käyttämällä kehitysympäristö luodaan. Tätä kirjoittaessa kaikki tiedostot ja ohjelmat ovat julkisesti saatavilla. Tämän ohjeen kautta voi vastaavan ympäristön toteuttaa myös omalle tietokoneelleen. Kirjoitushetkellä ohjeet ja tarvittavat tiedostot löytyvät kahdesta GitHub-säilöstä (Lähdesmäki 2020a; Lähdesmäki 2020b).

Kokonaisuuden luominen vaatii paikallisen Kubernetes ohjelmiston ja konttien ajoympäristön. Ohjelmina toimivat Minikube ja Docker. Muita ohjelmia ovat kubectl, MySQL Workbench ja git for windows. Kloonaa haluamaasi polkuun GitHub säilöt. Jälkimmäinen puretaan ensimmäisen säilön ”theme” -kansioon. Mikäli haluaa käyttää omaa teemaa, silloin täytyy muokata Dockerimagen rivillä 11 olevaa ”ENV theme” ympäristömuuttujaa.

Minikube käynnistetään komennolla “minikube start”. Ohjelma ilmoittaa, kun se on käynnistänyt kaikki toiminnot. Tilt täytyy avata siinä kansiossa missä tiltfile on. Komentorivityökalussa täytyy olla järjestelmänvalvojan oikeudet. Mikäli järjestelmä ei ala rakentamaan sivustoa, kannattaa aukaista tiltfile tekstieditorissa ja tallentaa se.

Tilt avaa selaimeen automaattisesti näkymän rakennuksen vaiheista. Kun Tilt on rakentanut ja julkaissut sivuston, klikataan wordpress-välilehdeltä “localhost:8080” linkkiä. Linkki ohjaa selaimen tuoreeseen Wordpress ympäristöön.

Toistaisesti sivu aukeaa Wordpress asennuksen kielivalintaan koska sillä ei ole tietokantaa.

Tietokannan asentaminen on automatisoitu tiltfilessä, mutta se toimii ainoastaan Linux-ympäristössä. Windows-käyttöjärjestelmällä täytyy käyttää erillistä ohjelmaa. Ohjelmalla MySQL Workbench avataan osoite 127.0.0.1:3306, joka viittaa paikalliseen mysql tietokantaan. Salasana tietokannalle on määritelty tiedostossa mysql-deployment.yaml. Oletuksena se on *password*. Kun ohjelma on yhdistänyt tietokantaan, valinnalla “Run SQL script” ja ladataan database.sql tiedoston sisältö ”wordpress” -kantaan.

Päivittämällä osoitteen “localhost:8080” näkee käynnissä olevan ympäristön. Tunnukset testiympäristöön ovat käyttäjätunnus: demouser ja salasana: password. Mikäli ”teema”-kansiossa sijaitsevaan esimerkkiteemaan tekee muutoksia, tulevat ne näkymään automaattisesti selaimessa. Lisäksi muutokset tallentuvat paikalliseen kansioon. Toteutuksessa versionhallinta on myös käytettävissä. Mikäli muuhun kuin teemaan tehtyjä muutoksia halutaan tallentaa, täytyy tietokannasta tallentaa kopio omalle koneelle.

Ympäristö suljetaan painamalla Tilt-komentorivissä näppäinyhdistelmää CTRL + C. Komennolla “tilt down” Tilt ajaa automaattisesti kubectl -käskyt. Se poistaa Kubernetes-ympäristöstä sivun, tietokannan ja levyt. Tämän jälkeen Kuberneteksen voi sulkea. Minikube suljetaan komennolla “minikube stop”.

4 TULOKSET JA JOHTOPÄÄTÖKSET

Näillä tuloksilla haluan herättää keskustelua ja tarjota yhtä ratkaisumallia. Tällä hetkellä projektien kehitysympäristöt on toteutettu monella eri tekniikalla. Tämä johtaa siihen, että työtunteja menee sen selvittämiseen, kuinka ympäristöt käynnistetään.

Projektien uudet työntekijät joutuvat tekemään selvitys- ja asennustyön. Käyttöjärjestelmän tai tietotekniikan erot aiheuttavat lisäongelmia ympäristöjen yhteensopivuudessa. Jos yritys pyrkii tehokkaaseen työskentelyyn, sillä täytyy olla vakiintuneita prosesseja.

Toteutukseni avulla yritys pystyy automatisoimaan kehitysympäristön luomista. Kehitysympäristö käynnistyy aina samalla komennolla, eikä käyttöjärjestelmällä tai tietotekniikalla ole juurikaan merkitystä. Parhaimmillaan saadaan yhtenäinen toimintatapa ja vähemmän kehitysympäristön parissa säätämistä.

Kubernetes ei sovi kaikkeen kehitykseen ja se vaatii tietokoneelta huomattavan paljon tehoja. Tietämys Docker-konteista, komentorivisovelluksista ja kokemusta Kubernetes-ympäristöistä auttaa ymmärtämään kokonaisuutta ja helpottaa häiriötilanteiden selvittämistä.

Aihe on tuore ja tietoa siihen liittyen tulee jatkuvasti lisää. Tiltin Windowsin alpha version myötä, Linux on kehityskäytössä käyttöjärjestelmänä parempi, silloin kun käytetään ratkaisumalliani. Uskon että tulevaisuudessa työkalut paranevat entisestään. Yhä suurempi osa yrityksistä suuntaa kohti mikropalveluita ja siten kysyntä Kubernetes-kehitysympäristöille kasvaa.

Työssä olisi voitu tutkia enemmän erilaisten ympäristöjen pystyttämistä. Kubernetes antaa tekniikkana mahdollisuuden monenlaisten ympäristöjen ylläpitämiseen. Esimerkiksi api-serveri ja sen käyttäminen samassa Kubernetteksessa olisi ollut mielenkiintoista testata. Ympäristöjen käynnistäminen ja sulkeminen mahdollistaa tehokkaan testauksen, mutta sitä ei käsitelty tässä työssä lainkaan. Kubernetes oli aluksi niin abstrakti kokonaisuus, että työn tekemisessä meni paljon aikaa kokonaisuuden hahmottamiseen. Selvitystyössä on käytetty tuntikausia aikaa ”yrityksen ja erehdyksen” -kautta opiskeluun.

Työn ratkaisun ja yrityksen tulevaisuuden oleellisin näkökulma lienee se, ettei Kubernetes yksin ole paras kehitysympäristö kaikissa tilanteissa. Työssä esitellään docker-composella toteutettu samanlainen kehitysympäristö. Docker-compose toimii yhden abstraktiotason alempana. Tämä tarkoittaa parempaa suorituskykyä ja parempaa virheensietokykyä.

Tärkeämpää on jouheva prosessi. Tilt -kehitystyökalu mahdollistaa selkeän ja yksiselitteisen prosessin. Mielestäni yrityksellä tulisi olla yksi linja siitä, miten projektien kehitysympäristöt käynnistetään. Se voi olla esimerkiksi GitHub

readme-tiedostoon kuvattu polku, mikäli yksikään tekniikka ei ole riittävän kattava, kattamaan kaikkia tilanteita. Tämä on tärkein havaintoni, jonka olen tehnyt opinnäytetyötä tehdessä.

Tilt ja Kubernetes antavat yhdessä todella suuren liikkumavaran toteuttaa erilaisia kehitysympäristöjä. Kun kehitysympäristön vaatimiin osiin käyttää aikaa ja saa ne toimimaan jouhevasti, antaa se tulevaisuudessa takaisin paljon. Kun työntekijöillä on käytössä yksi tapa käynnistää ympäristö, nopeuttaa se uuden työntekijän perehdytystä, joka taas vapauttaa henkilöstöresursseja tuottavan työn tekemiseen.

Järjestelmän kehitysympäristö saattaa olla monimutkainen ja sen tekemiseen voi mennä aikaa. Siksi sovelluskehitysympäristön toteutukseen kannattaa varata aikaa. Väliaikaisten ratkaisujen tekemistä tulisi välttää.

5 PÄÄTÄNTÖ

Raportti on ollut kova puristus. Onneksi tätä kirjoittaessa päällä oleva pandemia ei ole vaikuttanut negatiivisesti prosessiin. Ainoa ero on seminaarin toteutus etänä. Aihe tuli minulta itseltäni, samalla kun seurasin harjoittelussa työntekijöiden tuskailua kehitysympäristöjen parissa. Toivottavasti pystyn tämän kokemuksen ansiosta tuomaan ilmi omaa tietämystäni ja näkemystäni aiheesta. Kyse on kuitenkin työntekijän mukavuudesta ja työn sujuvuudesta.

Kubernetes on hankala kokonaisuus otettavaksi haltuun. Internet tarjoaa paljon erilaisia selityksiä, artikkeleita, videoita ja kursseja. Olen vasta raapaissut ohjelman todellisen potentiaalin pintaa. Olen harmissani, kun työhön ei saatu monimutkaisempia järjestelmiä mukaan. Todennäköisesti se ei olisi ollut edes kauhean kaukana.

Aiheeseen perehtyminen ei ole mennyt hukkaan. Ilmeni että vasta viimeisen 5 vuoden aikana aiheesta on alettu puhumaan enemmän. Kyse on tekniikoista, joita käyttävät maailman suurimmat tietotekniikkayritykset.

Työn aihe muuttui pariin otteeseen tehdessä. Mielestäni se on osa opinnäytetyön teko prosessia. Olen tyytyväinen lopputulokseen ja opin valtavan paljon

uutta asiaa. En olisi voinut tehdä enempää sisältöä työhön. Olen myös tyytyväinen, että työ pysyi suunnitellussa aikataulussa.

Toivon, että yrityksessä opinnäytetyö herättää keskustelua. Yritykseen tulee uusia työntekijöitä tasaisesti lisää ja näkemykseni mukaan, selkeästä prosessista on hyötyä nimenomaan uudelle työntekijälle. Koulutukseni alkoi virtuaalikoneen asennuksen opettelulla. En olisi uskonut, että se myös päättyisi siihen.

LÄHTEET

Boxell, M. 2019. Container Native Development Tools Compared: Draft, Skaffold, and Tilt. Videoleike. Saatavissa: https://www.youtube.com/watch?v=Be1zgQmZFEY [viitattu 13.4.2020].

Brown, A., Forsgren, N., Humble, J., Kersten, N & Kim, G. 2016. State of DevOps Report. Puppet Labs, DORA. PDF-dokumetti. Saatavissa: https://services.google.com/fh/files/misc/state-of-devops-2016.pdf [viitattu 13.4.2020].

Cito, J., Schermann, G., Wittern, J., Leitner, P., Zumberi, S. & Gall, H. 2017. An Empirical Analysis of the Docker Container Ecosystem on GitHub. PeerJ Preprints. PDF-dokumentti. Saatavissa: https://peerj.com/preprints/2905.pdf [viitattu 13.4.2020].

Cochrane, K. 2013. ”How is Docker different from a virtual machine?”. Foorumi vastaus. Päivitetty 19.11.2019. Saatavissa: https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine [viitattu 23.3.2020].

Cois, C. 2015. Continuous Integration in DevOps. WWW-dokumentti. Saatavissa: https://insights.sei.cmu.edu/devops/2015/01/continuous-integration-in-devops-1.html [viitattu 28.4.2020].

Devops s.a. Eficode Oy. PDF-dokumentti. Saatavissa: https://www.eficode.com/learn/devops-guide [viitattu 10.3.2020].

Docker. 2020a. Build Kubernetes-ready applications on your desktop. WWW-dokumentti. Saatavissa: https://www.docker.com/products/kubernetes [viitattu 28.4.2020]

Docker. 2020b. Our Company. WWW-dokumentti. Saatavissa: https://www.docker.com/company [viitattu 21.4.2020].

Efftinge, S. 2019. Continous Dev Environments. Gitpod. Blogi. Saatavissa: https://www.gitpod.io/blog/continuous-dev-environment-in-devops/ [viitattu 23.3.2020].

Grinten, F. 2019. SecrectHub. WWW-dokumentti. Saatavissa: https://secrethub.io/blog/decouple-application-secrets-from-ci-cd-pipeline/ [viitattu 23.3.2020].

Heino, P. 2010. Pilvipalvelut. Helsinki: Talentum Media Oy.

Kotilainen, S. 2017. Koodi sujahtaa konttiin – sovellusten kehittäminen mullistuu. Tivi.fi. Artikkeli. Saatavissa: https://www.tivi.fi/uutiset/koodi-sujahtaa-konttiin-sovellusten-kehittaminen-mullistuu/7931ccac-1cd7-3c40-b338-be25469cf1dd [viitattu 13.4.2020].

Kubernetes documentation. 2019a. Pod-overview. The Linux Foundation: Kubernetes documentation. WWW-dokumentti. Saatavissa: https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/ [viitattu 10.3.2020].

Kubernetes documentation. 2019b. Viewing Pods and Nodes. The Linux Foundation: Kubernetes documentation. WWW-dokumentti. Saatavissa: https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/

Kubernetes documentation. 2020a. Concepts. The Linux Foundation: Kubernetes documentation. WWW-dokumentti. Saatavissa: https://kubernetes.io/docs/concepts/ [viitattu 10.3.2020].

Kubernetes documentation. 2020b. Deployment. The Linux Foundation: Kubernetes documentation. WWW-dokumentti. Saatavissa: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/ [viitattu 10.3.2020].

Kubernetes documentation. 2020c. Ingress. The Linux Foundation: Kubernetes documentation. WWW-dokumentti. Saatavissa: https://kubernetes.io/docs/concepts/services-networking/ingress/ [viitattu 10.3.2020].

Kubernetes documentation. 2020d. Nodes. The Linux Foundation: Kubernetes documentation. WWW-dokumentti. Saatavissa: https://kubernetes.io/docs/concepts/architecture/nodes/ [viitattu 10.3.2020].

Kubernetes documentation. 2020e. Overview of kubectl. The Linux Foundation: Kubernetes documentation. WWW-dokumentti. Saatavissa: https://kubernetes.io/docs/reference/kubectl/overview/ [viitattu 14.4.2020].

Kubernetes documentation. 2020f. Persistent volumes. The Linux Foundation: Kubernetes documentation. WWW-dokumentti. Saatavissa: https://kubernetes.io/docs/concepts/storage/persistent-volumes/ [viitattu 10.3.2020].

Kubernetes documentation. 2020g. WWW-dokumentti. Using a Service to Expose Your App. WWW-dokumentti. Saatavissa: https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/ [viitattu 12.5.2020].

Leppä, A. 2020. Teknologiajohtaja. Keskustelu 21.4.2020. Metatavu Oy. Mikkeli.

Lähdesmäki, E. 2020a. Local Wordpress in Kubernetes. GitHub-säilö. Saatavissa: https://github.com/Eskilmar/kubernetes-dev-env [viitattu 10.5.2020].

Lähdesmäki, E. 2020b. Example theme to Wordpress in Kubernetes. GitHub-säilö. Saatavissa: https://github.com/Eskilmar/welcome-to-kubernetes-theme/ [viitattu 10.5.2020].

Rathod, H., Townsend, J. 2014. Virtualization 2.0 For Dummies. Chichester: John Wiley & Sons, Ltd.

Sanche, D. 2018. Kubernetes 101: Pods, Nodes, Containers, and Clusters. Medium. WWW-dokumentti. Saatavissa: https://medium.com/google-cloud/kubernetes-101-pods-nodes-containers-and-clusters-c1509e409e16 [viitattu 10.3.2020].

Wallenius, N. ”Mitä mikropalvelut ovat?”. Blogi. Saatavissa: https://niklaswallenius.fi/teknologiat/mikropalvelut/attachment/docker-2/ [viitattu 02.05.2020].

KUVALUETTELO