Avoimen lähdekoodin Kubernetes-alustat

Kustannustehokkaat Kubernetes-alustavaihtoehdot yritystoiminnassa

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Hämeenlinnan korkeakoulukeskus, tietojenkäsittely

Syksy, 2020

Joni Komulainen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

TIIVISTELMÄ

Tietojenkäsittely
Hämeenlinnan korkeakoulukeskus

Tekijä Joni Komulainen **Vuosi** 2020

Työn nimi Avoimen lähdekoodin Kubernetes-alustat

Työn ohjaaja/t Tero Keso, Lasse Seppänen

TIIVISTELMÄ

Opinnäytetyön tavoitteena oli löytää avoimen lähdekoodin Kubernetes-alustavaihtoehtoja yritykselle, joka haluaa ottaa Kuberneteksen käyttöön testaus- tai tuotantoympäristössään mahdollisimman pienillä kustannuksilla. Samalla tutkimuksessa selvitetään, onko olemassa yhtä alustaa, joka sopii valtaosan yritysten tarpeisiin. Teoriaosuudessa käytiin lyhyesti ja ytimekkäästi läpi mitä ovat virtualisointi, Docker, monoliittinen sovellusarkkitehtuuri, mikropalveluarkkitehtuuri, julkiset pilvipalvelut ja Kubernetes.

Käytännön osuudessa vertailtiin yhdeksää avoimen lähdekoodin Kubernetes-alustaa. Näistä yhdeksästä alustasta kolme valittiin tarkempaan testaukseen. Valitut alustat olivat Minikube, Rancher ja Kubesphere, joista kahdesta jälkimmäisestä paljastui kaksi eri asennustapaa. Molemmat asennustavat otettiin huomioon testauksissa.

Alustat pyrittiin asentamaan virtuaaliseen ympäristöön, mutta tutkimuksissa selvinneistä syistä yksi alustoista päädyttiin asentamaan kannettaville tietokoneille. Asennuksen jälkeen jokaisella alustalla otettiin käyttöön Wordpress MySQL-tietokannalla ja Rocket.Chat. Opinnäytetyössä keskitytään alustojen eroavaisuuksiin, jotka kävivät ilmi itse alustojen, että testauspalveluiden asennusten aikana, eikä ohjeistusta asennuksiin anneta.

Tutkimuksen tulokset kasattiin yhteen ja testatut alustat pisteytettiin kokemuksien perusteella. Selkeästi parasta alustaa ei kyetty selvittämään, sillä alustojen sopivuus riippuu sitä käyttävästä yrityksestä.

Avainsanat Docker, Kubernetes, Kubernetes-alusta, mikropalveluarkkitehtuuri, avoin lähdekoodi

Sivut 35 sivua, joista liitteitä 2 sivua

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

Degree Programme in Business Information Technology
Visamäki university Centre

Author Joni Komulainen **Year** 2020

Subject Open source Kubernetes platforms

Supervisors Tero Keso, Lasse Seppänen

ABSTRACT

The main objective of this thesis was to find open source Kubernetes platform alternatives for a company that wants to implement Kubernetes in their testing or production environment with as low expenses as possible. At the same time the aim was to find out whether there is one platform that suits the needs of the majority of companies. The theory part of this thesis covered briefly what virtualization Docker, monolithic application architecture, microservice architecture, public cloud and Kubernetes are.

In the practical part, nine open source Kubernetes platforms were compared. From those nine platforms, three were selected for further testing. The selected three platforms were Minikube, Rancher and Kubesphere, latter two of which had two different install methods. Both installation methods were considered in the testing.

The intention was to install the platforms inside virtual environments, but for the reasons that emerged in the studies, one of the platforms was installed on laptops. After installation of the platforms, Wordpress using MySQL and Rocket.Chat were deployed on each platform. The main focus of the thesis is in the differences of the platforms that were detected during the installations and testing, and no installation instructions are provided.

Finally, the results of the study were gathered, and the tested platforms were scored based on the experiences. The best platform could not be determined because the suitability of the platforms depends on the company using the platform.

Keywords Docker, Kubernetes, Kubernetes platform, microservice architecture, open source

Pages 35 pages including appendices 2 pages

SISÄLLYS

1 JOHDANTO

Ohjelmien ja järjestelmien kasvaessa suuriksi kokonaisuuksiksi, niiden servereiltä vaatima laskentateho kasvaa merkittävästi. Tämä voi johtaa olemassa olevien servereiden päivittämiseen ja uusiin laitehankintoihin vanhan laitteiston tehon loppuessa kesken. Mikäli muutoksia ei tehdä, ohjelma tai järjestelmä käy ennen pitkää hitaaksi tai pahimmassa tapauksessa siitä voi tulla epävakaa ja täysin käyttökelvoton. Laitteiston päivitys ja uudet hankinnat kuitenkin tuovat mukanaan suuret kustannukset, jotka eivät rajoitu pelkästään itse laitteistoon, vaan varoja tarvitaan myös niiden käyttöönottoon ja ylläpitoon. Sekä käyttöönotto että ylläpito ovat aikaa vieviä prosesseja, jolloin olemassa olevat henkilöstöresurssit voivat myös loppua kesken.

Voidakseen käyttää olemassa olevan laskentatehon mahdollisimman tehokkaasti, yrityksen tulee purkaa laskentatehoa käyttävät ohjelmat pienempiin, erillisiin yksikköihin. Erilliset yksiköt kyetään jakamaan tasaisemmin olemassa oleville laitteille niin, että jokainen osa saa tarvitsemansa määrän laskentatehoa ja servereiden tehoista mahdollisimman pieni osa jää käyttämättä. Toimimalla näin, yrityksen on mahdollista säästää resursseihin, niiden ylläpitoon ja huoltoon kuluvia kustannuksia.

Näiden erillisten yksiköiden manuaalinen hallinta on kuitenkin hyvin työlästä. Kubernetes on kehitetty automatisoimaan näiden yksittäisten palojen hallintaa. Tässä opinnäytetyössä tutustutaan Kuberneteksen eri alustoihin sekä vertaillaan niitä toisiinsa. Tavoitteena on löytää avoimen lähdekoodin vaihtoehtoja alustaksi yritykselle, joka haluaa aloittaa Kuberneteksen käytön testaus- ja kehitysympäristöissään tehostaen näin resurssiensa käyttöä, muttei kykene sijoittamaan siihen merkittävää summaa rahaa. Varsinaiset tutkimuskysymykset ovat seuraavat:

- Mitä vaihtoehtoja on yrityksellä, joka haluaa ottaa käyttöön Kuberneteksen avoimen lähdekoodin alustalla edullisesti omassa liiketoiminnassaan, kehityksessään tai tuotantoympäristössään?
- Onko olemassa yksi ainoa alusta, joka soveltuu usean erilaisen yrityksen tarpeisiin?

Opinnäytetyön vertailevaan tapaustutkimukseen valitaan yhdeksän suosituinta avoimen lähdekoodin alustaa perustuen niiden saamiin tähtiin GitHub-palvelussa. GitHubin tähdet eivät ole välttämättä tarkin mittaustapa alustojen suosiolle, mutta ne kertovat käyttäjien arvostuksesta ja kiinnostuksesta projekteja kohtaan. Näiden yhdeksän alustan perusominaisuuksia, jotka alustojen julkaisijat ovat julkistaneet, vertaillaan ensin toisiinsa taulukon avulla. Yhdeksästä alustasta kuusi pudotetaan pois perustuen alustojen ominaisuuksiin ja jäljelle jäävät kolme alustaa otetaan seuraavaan testausvaiheeseen, joka toteutetaan käytännössä alustojen

todellisten erojen selvittämiseksi, sillä pelkät numerot ja ominaisuuslistaukset eivät aina kykene kertomaan koko totuutta.

Jokainen näistä kolmesta alustasta otetaan käyttöön erillisillä vCommanderin Ubuntu 18.04 virtuaalikoneilla, joiden tekniset ominaisuudet rajoitetaan toisiaan vastaaviksi. Alustojen asennusta kokeillaan myös oppilaitoksen verkon ulkopuolisille koneille.

Ensin järjestelmissä tullaan ajamaan Wordpress-kehitysympäristöä ja tämän jälkeen jokaiseen järjestelmään tullaan asentamaan Rocket.Chat-ohjelmisto. Käytännön testejä tehdessä pyritään löytämään eroavaisuuksia alustoista, jotka eivät tule esille ilman käytännön toteutusta.

2 VIRTUALISOINTI

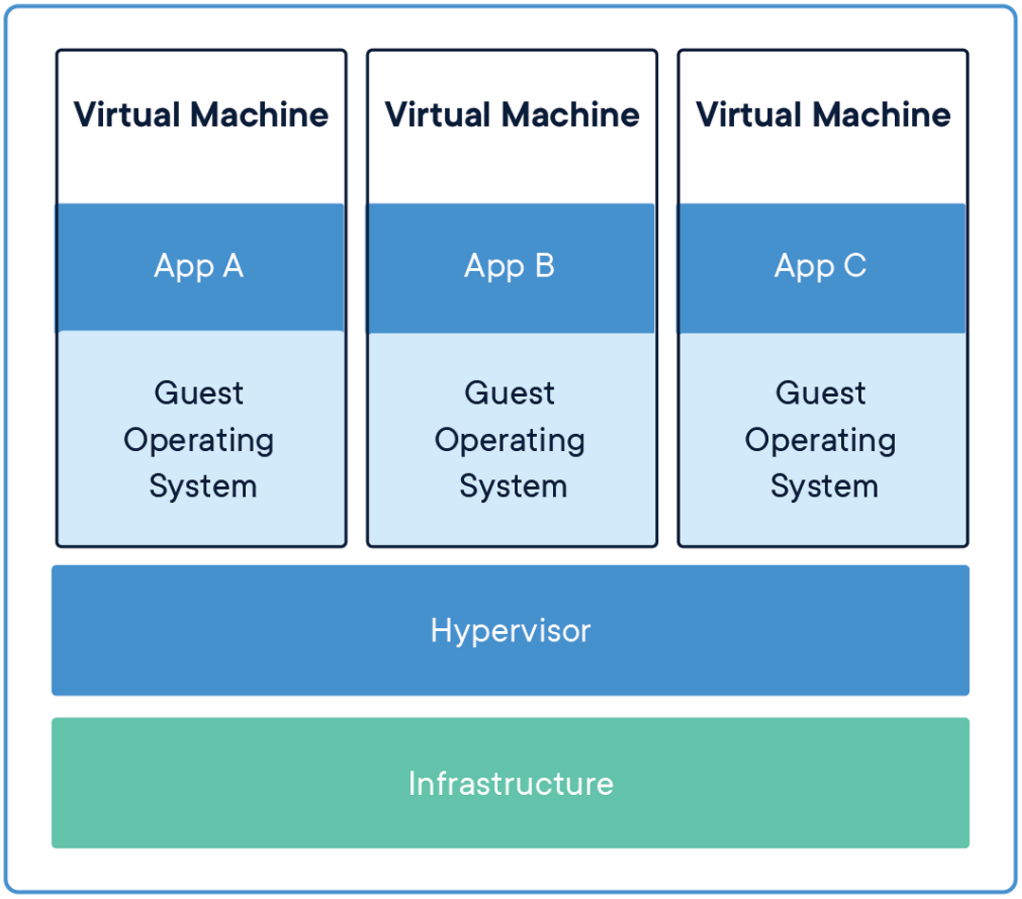
Virtualisointi on 2000-luvun alkupuolella yleistynyt tekniikka, jonka avulla olemassa olevat laskentaresurssit kyetään jakamaan useampaan osaan ja käyttämään ne tällä tavoin tehokkaammin ja säästämään rahaa olemassa olevien resurssien riittäessä suuremmalle määrälle tehtäviä. Virtualisoinnissa olemassa oleva tietokone jaetaan useampaan virtuaaliseen tietokoneeseen, joilla kaikilla on omat virtuaaliset komponenttinsa. (RedHat, n.d.c)

2.1 Virtuaaliset koneet

Koneen resurssien jakaminen virtuaalisiin koneisiin hoidetaan ohjelmalla, joka hallitsee koneen fyysisiä resursseja ja luo niistä virtuaalisia komponentteja, kuten esimerkiksi prosessoreita tai kovalevyjä. Tätä ohjelmaa kutsutaan hypervisoriksi. Hypervisor voi olla tyypin yksi hypervisor, joka asennetaan suoraan ”raakaraudalle” (bare metal) ilman alla olevaa käyttöjärjestelmää ja toimii itse ikään kuin fyysisen koneen käyttöjärjestelmänä. Tyypin kaksi hypervisor muistuttaa enemmän täysin tavallista tietokoneella käytettävää ohjelmaa ja se asennetaan alla olevan käyttöjärjestelmän päälle. (Rouse, n.d.b; Opensource.com, n.d.)

Kun virtuaaliset resurssit on luotu hypervisorin avulla, niille voidaan asentaa käyttöjärjestelmä. Jokaisessa virtuaalikoneessa voi olla eri käyttöjärjestelmä tai eri versio samasta käyttöjärjestelmästä tarpeiden mukaan. Tämän jälkeen virtuaalikoneilla kyetään ajamaan ohjelmia ja ohjelmistoja samaan tapaan kuin fyysisilläkin koneilla. Virtualisointi mahdollistaa myös jokaisen ohjelman ajamisen omassa virtuaalikoneessa erillään muista virtuaalikoneista tai itse fyysisestä koneesta. Kuvassa 1 on selitettynä visuaalisesti virtualisoinnin toimintaperiaate. Vihreä *Infrastructure*-palikka voi siis koostua tyypin yksi hypervisorista asennettuna fyysiselle koneelle tai tyypin kaksi hypervisorista ja fyysisen koneen omasta käyttöjärjestelmästä. (RedHat, n.d.c; Rouse, n.d.b.; Opensource.com, n.d.)

Resurssien tehokkaamman hyödyntämisen lisäksi virtualisoinnin hyviä puolia ovat helpompi varmuuskopiointi fyysisiin koneisiin verrattuna. Virtuaaliset koneet on myös helpompi palauttaa tilaan ennen ongelmaa tai tapahtunutta virhettä. Tämä tekee virtuaalisista koneista hyviä testausympäristöjä. Virtuaaliset koneet ovat myös eristettyinä itse fyysisestä koneesta, joten virtuaalisten koneiden ongelmat eivät vaikuta itse fyysiseen koneeseen tai sen toimintaan. (Rouse, n.d.b; Opensource.com, n.d.)



Kuva 1 Virtual machines (Docker, n.d.)

2.2 Docker ja kontit

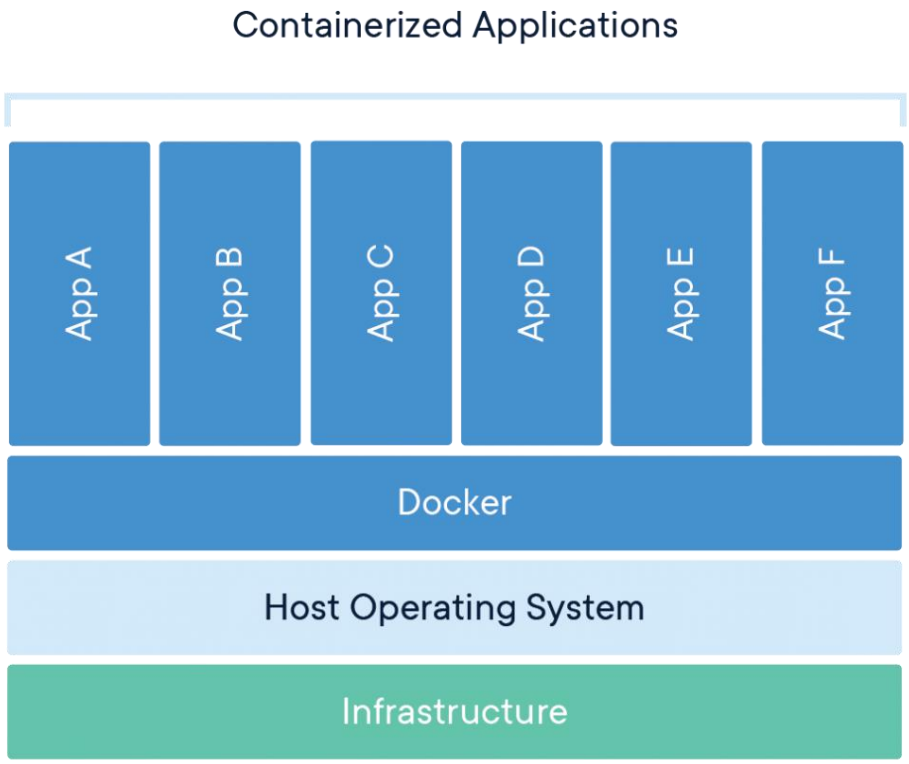
Docker on vuonna 2013 julkaistu työkalu, jonka avulla sovelluksia voidaan pakata pienten ja kevyiden virtuaalisten ympäristöjen, eli konttien sisään. Kontit sisältävät itse sovelluksen lisäksi kaikki sovelluksen tarvitsemat osat ja riippuvuudet, eikä konttien ajamiseen siis tarvita muuta kuin Docker. Tämä mahdollistaa konttien ajamisen jokaisessa ympäristössä, jolle Docker saadaan asennettua. (Matthias, Kane, 2015, s. 1, 9)

Virtualisoinnissa fyysisen tietokoneen resurssit jaetaan virtuaalisille tietokoneille, jotka ovat eristettyinä toisistaan. Tämä mahdollistaa olemassa olevien resurssien tehokkaamman käytön. (Opensource.com, n.d.; RedHat, n.d.a)

Toisin kuin virtualisoinnissa, kontit eivät virtualisoi koko konetta ja sen resursseja kuten käyttömuistia ja prosessoria, vaan kontit virtualisoivat pelkän käyttöjärjestelmän ja sen osat ajettavaa ohjelmaa ja sen riippuvuuksia varten. Virtuaalikoneiden tapaan kontit toimivat eristettyinä ympäristöinä, mutta ovat nopeampia käynnistää ja kevyempiä ajaa pelkän käyttöjärjestelmän virtualisoinnin seurauksena. Konttien siirtely onnistuu myös helpommin laitteelta ja alustalta toiselle konttien sisältäessä kaiken

sovelluksen tarvitseman, eikä itse alusta vaikuta konttien toimintaan, oli kyseessä sitten Windows, Linux, pilvi tai raakarauta. (Docker, n.d.)

Kuvassa 2 Dockerin toimintaperiaate on visualisoitu. Vihreä *Infrastructure*-palikka tarkoittaa tässä tapauksessa laitteistoa, joka suorittaa laskentaa, eli esimerkiksi serveri. Tämän päälle on asennettu käyttöjärjestelmä ja itse Docker, jonka päälle kontit rakennetaan.



Kuva 2 Containers (Docker n.d.)

2.3 **Kubernetes**

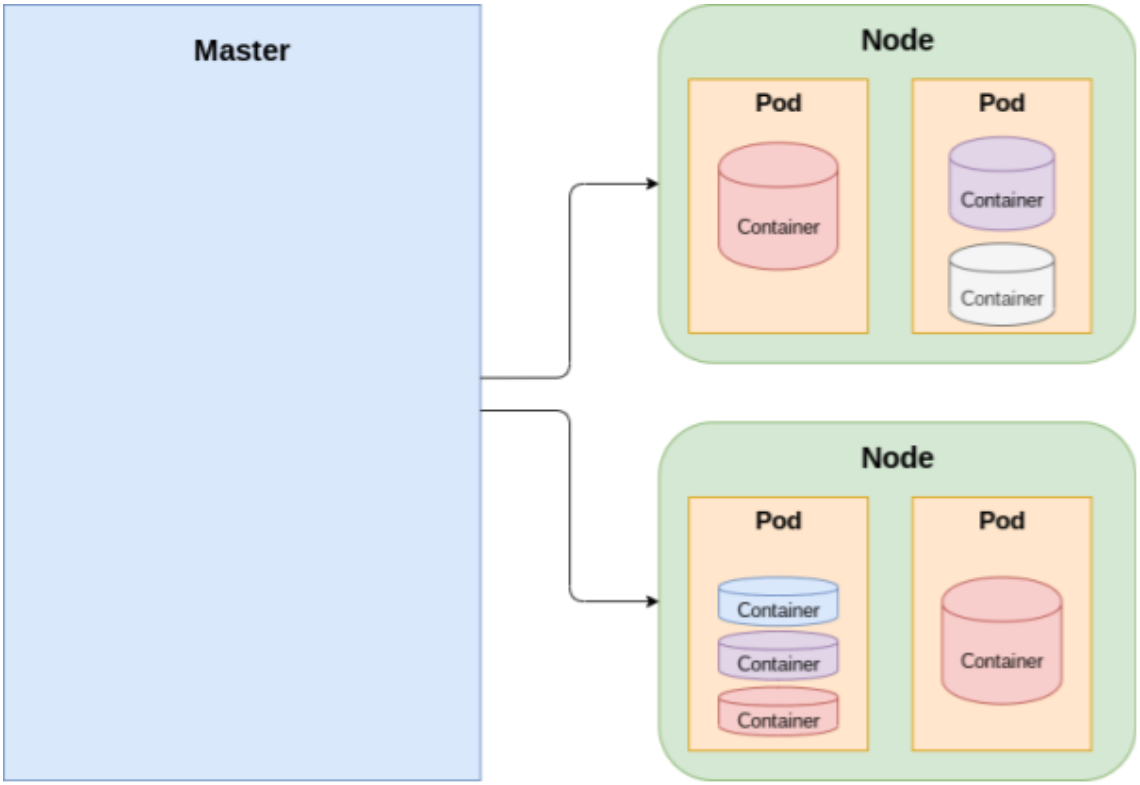
Kubernetes on Googlen kehittämä Borg-hallintakonsoliin perustuva Linux-kontteihin purettujen mikropalveluiden hallintaohjelma, jonka lähdekoodin Google julkaisi vuonna 2014 vapaasti kaikkien saataville. Kubernetes “orkestroi” kontteja, eli se tietää mitkä kontit toimivat yhdessä ja tarvitsevat toisiaan, ja se pitää huolen siitä, että nämä kontit ovat käynnissä ja toiminnassa saman aikaisesti. Kuberneteksen avulla voidaan myös automatisoida järjestelmän tarkkailua, jolloin Kubernetes havaitsee mahdollisesti alhaalla olevat ja vastaamattomat kontit ja käynnistää ja sulkee niitä automaattisesti pitäen järjestelmän toimivana. Se pystyy myös jakamaan

käytettävissä olevaa laskentatehoa automaattisesti sen mukaan, mitkä kontit sitä tarvitsevat eniten. (RedHat, n.d.b; Kubernetes, n.d.b)

Kubenreteksen arkkitehtuuri koostuu yksinkertaistetusti masterista tai mastereista, nodesta tai nodeista, podeista ja konteista (Kuva 3). Kontit ovat kuin pieniä ja kevyitä virtuaalikoneita, jotka virtualisoivat pelkästään käyttöjärjestelmän kokonaisen koneen sijaan muodostaen pieniä ympäristöjä yksittäisille ohjelmille. Kontit sisältävät ohjelman lisäksi kaikki sen vaatimat riippuvuudet, jotka mahdollistavat ohjelman toiminnan. Podit sisältävät yhden tai joissain tapauksissa useamman kontin.

Nodet ovat “työtä tekeviä” virtuaalisia tai fyysisiä servereitä. Ne sisältävät podeja ja ajavat niiden sisältämiä kontteja. Kaikki konttien suorittama laskenta tapahtuu nodeissa.

Master ei yleensä tee varsinaista työtä, eli ohjelmien ajamista kuten nodet, vaan se hallitsee nodeja. Se jakaa työtehtävät nodejen välille ja ohjeistaa niitä. Master on siis ikään kuin työnjohto.



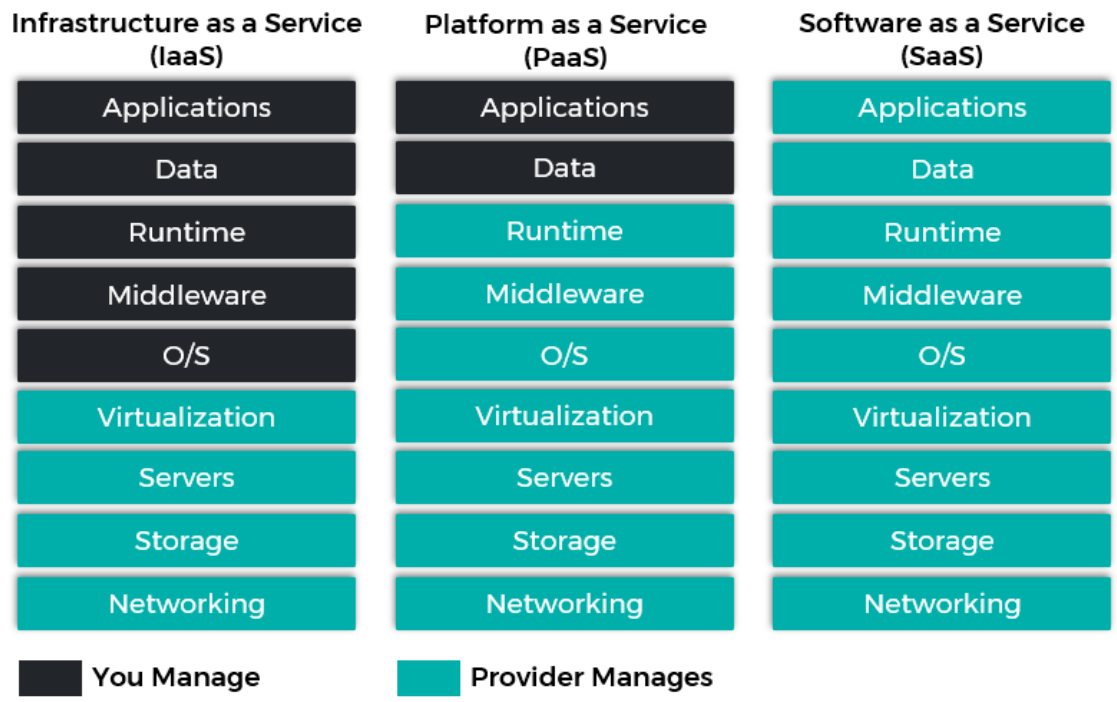
Kuva 3 Yksinkertaistettu havainnollistus Kuberneteksen rakenteesta

3 JULKISET PILVIPALVELUT JA NIIDEN MALLIT

Mikäli yritys ei halua käyttää omaa laitteistoa Kuberneteksen ja oman ohjelman tai ohjelmistonsa pyörittämiseen, ovat myös julkiset pilvipalvelut varteenotettava vaihtoehto. Julkiset pilvipalvelut tarjoavat käyttäjilleen laskentatehoa, tallennustilaa ja valmiita sovelluksia, joista maksetaan yleisimmin käytön mukaan. Sen sijaan että laskentateho, tallennustila tai sovellukset olisivat omalla laitteistolla omissa tiloissa, pilvipalveluissa nämä saadaan ulkoiselta palveluntarjoajalta internetin välityksellä omilla laitteilla käytettäväksi. (Ranger, 2018; Citrix, n.d.)

Yleisimmät pilvipalvelumallit (Kuva 4) ovat IaaS (infrasrtucture as a Service), PaaS (Platform as a Service) ja SaaS (Software as a Service). IaaS-mallissa palveluntarjoaja tarjoaa käyttöön laskentatehoa, tallennustilaa ja hypervisorin ja kaikki muu on itse käyttäjän vastuulla käyttöjärjestelmästä koneilla ajettaviin ohjelmiin. IaaS-malli on näin ollen työläämpi ylläpitää, mutta se antaa myös vapaammat kädet käyttäjälleen. PaaS on malli, jossa palvelun ostaja saa käyttöönsä IaaS-mallin komponenttien lisäksi myös käyttöjärjestelmän ja muut ohjelmistotyökalut esimerkiksi ohjelmistojen kehitykseen. Palveluntarjoaja huolehtii tällöin myös käyttöjärjestelmän toimivuudesta, joka vähentää palvelun käyttäjän työmäärää, mutta vähentää samalla myös käyttäjän valtaa, koska käyttöjärjestelmä ei ole enää täysin käyttäjän hallittavissa. SaaS-mallissa palveluntarjoaja tarjoaa käyttäjälle suoraan valmiin ohjelmiston, eikä käyttäjän tarvitse huolehtia juurikaan mistään itse. Malli on myös kaikista vähiten käyttäjän itsensä hallittavissa. (Hou, n.d.; Rouse, n.d.c.; Citrix, n.d.)

Kolme suurinta ja tunnetuinta pilvipalveluntarjoajaa ovat Amazon Web Services (AWS), Microsoft Azure sekä Google Cloud (anuupadhyay, n.d.; OneClick, 2019; Dignan, 2020). Kyseisiltä palveluntarjoajilta löytyy sekä IaaS- että PaaS-ratkaisuja Kubernetesta varten. Kubernetes kyetään asentamaan julkista pilveä käyttäessä siis sekä IaaS- että Paas-mallin mukaisiin pilvipalveluihin.



Kuva 4 IaaS, PaaS, SaaS (Williamson, 2020)

4 SOVELLUSARKKITEHTUURI

Sovelluskehittäjät käyttävät työssään sovelluksen kehittämistä helpottavia malleja ja tekniikoita (Crypterium, 2018). Nämä tekniikat määrittävät sovelluksen toimintaa ja sitä, kuinka sovellus toimii yhdessä toisten sovellusten tai esimerkiksi tietokantojen kanssa. Näiden tekniikoiden ja mallien yhdistelmää kutsutaan sovellusarkkitehtuuriksi. (Rouse, n.d.a)

Sovellusarkkitehtuureja on useita erilaisia, joista mikropalveluarkkitehtuuri sekä monoliittinen arkkitehtuuri ovat keskeisimpiä tämän opinnäytetyön kannalta.

4.1 Monoliittinen arkkitehtuuri

Sovellusten, ohjelmien ja järjestelmien arkkitehtuurista puhuessa, voidaan puhua monoliittisesta arkkitehtuurista. Ns. “perinteinen sovellus” on arkkitehtuuriltaan monoliittinen. Monoliittisessa sovelluksessa on kolme suurta pääosaa: tietokanta, joka sisältää kaiken tiedon, käyttäjälle näkyvä käyttöliittymä eli front-end sekä kaikista toiminnoista huolehtiva koodi, eli back-end. Nämä kolme suurta osaa koostavat yhden kokonaisen monoliittisen sovelluksen. (Mulesoft, n.d.)

Monoliittiset sovellukset ovat usein helppoja kehittää ja koko sovelluksen testaus on helppoa, johtuen siitä, että kaikki ohjelman rakennuksessa käytettävä koodi on yhdessä palassa ja monesti jaettu myös selkeästi front-endin ja back-endin koodiin. Tämä saa sovelluksen rakenteesta yksinkertaisen ja sitä kautta myös omalta osaltaan helpommin ymmärrettävän. (Anastasia D., 2019)

Monoliittinen arkkitehtuuri voi muodostua ongelmaksi sovelluksen koon kasvaessa. Jos yhtä osaa sovelluksesta halutaan muuttaa tai siihen halutaan lisätä uusi ominaisuus, nämä muutokset yhteen osaan sovellusta vaativat koko sovelluksen uudelleenrakentamisen yhtenäiseksi paketiksi. Myös yksittäinen toimimaton osa voi kaataa koko sovelluksen, jolloin virheen korjaus vaatii jälleen koko sovelluksen uudelleenrakentamisen. Myös koodista tulee vaikeampaa lukea ja hallita, koska sitä on enemmän. (Gnatyk, 2018)

Sovelluksen koon kasvaessa sovellus muuttuu koko ajan raskaammaksi ajaa, jolloin käynnistysaika kasvaa ja sovellusta on skaalattava, jotta se saisi yhä tarvitsemansa laskentatehon. Skaalaus voidaan toteuttaa ylöspäin, eli vertikaalisesti, jolloin lisätään sovellusta ajavan serverin laskentatehoa. Monoliittisen sovelluksen vertikaalisen skaalauksen ongelmana on se, ettei sovellusta saada jakamaan laskentatehoa järkevästi niin, että eniten tehoa vaativat osat sovellusta saavat enemmän tehoa kuin osat, jotka pärjäävät pienemmällä määrällä. (Belamaric, Adkins, Robinson, Giersch & Makogon, 2016, s. 103; Nortal, 2017)

Toinen skaalausmahdollisuus on sivuille levittämällä, eli horisontaalisesti, jolloin sovellus laitetaan pyörimään useilla eri servereillä samanaikaisesti. Monoliittista arkkitehtuuria käytettäessä sovellusta ei kuitenkaan kykene täysin purkamaan paloihin pyörimään useille eri servereille ja näin ollen horisontaalisesti skaalatessa useille eri servereille, serverit vaativat tarpeettoman paljon laskentatehoa, sillä varsinaista lisätehoa tarvitsevien osien mukana servereille siirretään myös muut sovelluksen osat. (Belamaric, Adkins, Robinson, Giersch & Makogon, 2016, s. 103; Nortal, 2017)

4.2 **Mikropalveluarkkitehtuuri**

Mikropalveluarkkitehtuurissa yksi suuri sovelluskokonaisuus on purettu pienempiin itsenäisesti toimiviin komponentteihin, jotka yhdistetään toisiinsa yhdeksi yhtenäiseksi sovellukseksi käyttäen sovellusrajapintoja (Anastasia D., 2019). Näin esimerkiksi käyttäjän sisäänkirjautuminen ja sisällön näyttäminen palvelussa voivat olla toisistaan erillisissä paloissa, mutta loppukäyttäjä ei huomaa eroa monoliittiseen järjestelmään tai sovellukseen.

Koska sovellus koostuu useista itsenäisistä komponenteista, on näitä komponentteja helppo jakaa eri servereille ja näin sovellusta saadaan skaalattua horisontaalisesti monoliitteja tehokkaammin. Myös vertikaalinen skaalaus helpottuu, koska sovellus on pienemmissä paloissa ja laskentatehoa voidaan jakaa näin enemmän palalle, joka sitä tarvitsee. Tämä pienentää myös omalta osaltaan kustannuksia, koska serverit eivät tarvitse “turhaa laskentatehoa” kokonaisen sovelluksen pyörittämiseen. (Nortal, 2017)

Mikropalveluiden päivittäminen ja ylläpito voi olla myös helpompaa, koska kehittäjät voivat ottaa yksittäisiä paloja sovelluksesta ja korjata, kehittää tai ylläpitää niitä yksittäisinä pieninä osina. Tällöin koko sovellusta ei tarvitse rakentaa uusiksi jokaisen muutoksen yhteydessä, vaan pelkästään yksi pieni osa. Näiden pienien osien testaus on myös helpompaa, kuin monoliittisella arkkitehtuurilla rakennetuissa sovelluksissa. (Anastasia D., 2019)

Koska sovellus on purettu useisiin pieniin osiin, sen testaus yhtenä suurena kokonaisuutena on yleensä kuitenkin vaikeampaa verrattuna monoliitteihin. Huonosti rakennettuna mikropalvelut voivat olla myös tietoturvaltaan huonompia, koska kaikki osat sidotaan toisiinsa sovellusrajapinnoilla, jotka tuovat mukanaan lisähaavoittuvuuksia. (Anastasia D., 2019)

Mikropalveluiden kehitys on myös osittain vaikeampaa, koska kaikki koodi ei ole selkeästi yhdessä paikassa ja kaikki eri osat tulee saada toimimaan toistensa kanssa yhdessä. Tämä vaatii huolellisempaa suunnittelua ja tiimiä, jolla on tietotaitoa mikropalveluarkkitehtuurilla rakentamisesta. Erilliset palaset kasvattavat myös tarvetta automaatiolle. (Anastasia D., 2019)

5 TUTKIMUSMETODI

Opinnäytetyön tutkimusmetodina käytetään tapaustutkimusta, jossa kohdetta tutkitaan sille ominaisessa ympäristössä. Tapaustutkimuksissa käytössä on usein määrällistä ja laadullista aineistoa, sekä monia erilaisia lähteitä. Yksittäinen tapaus tai tapaukset ovat keskeinen asia tapaustutkimukselle ja niiden keskeisyys on erottava tekijä tapaustutkimuksen ja muiden tutkimuksien välillä. (KvantiMOTV, n.d.; Eriksson & Koistinen, 2014)

Tapaustutkimusta on suosittu sosiaalitieteissä molemmin puolin 1950-lukua. Kun klassinen tapaustutkimus alkoi yleistyä yhteiskuntatieteissä, seurasi keskustelua tutkimuksen tulkinnasta ja pinnalle nousi myös se, ettei tutkimustuloksien avulla kyetä tekemään tilastollisia yleistyksiä. Keskustelua edisti survey-tutkimusten nouseva suosio ja tapaustutkimuksia alettiin suosia lähinnä esitutkimuksen teossa. (Eriksson & Koistinen, 2014)

Myöhemmin survey-tutkimuksia on alettu tarkastella kriittisemmin ja tapaustutkimusten suosio on noussut. Tapaustutkimusten on nähty soveltuvana monimutkaisten ja muuttuvien kokonaisuuksien tutkimiseen ja sen vahvuuksina on pidetty kykyä tuottaa monipuolisia ja yhteen kietoutuvia rakenne- ja toimijasuhteita analysoivia tuloksia. Nykyään tapaustutkimuksesta on tullut oma erillinen tutkimuksellinen lähestymistapansa, eikä sitä käytetä enää pelkästään esitutkimuksiin tai survey-tutkimusten täydentämiseen. (Eriksson & Koistinen, 2014)

Vaikeaa tapaustutkimuksen määrittelystä ja hahmottamisesta tekee sen monimuotoisuus tieteenfilosofisten oletusten suhteen. Tapaustutkimuksen hedelmällisyys johtuukin juuri siitä, että tutkija kykenee valitsemaan positivistisen tai postpositivistisen kehyksen omalle tutkimukselleen. (KvantiMOTV, n.d.; Eriksson & Koistinen, 2014)

Vertaileva tapaustutkimus vaatii käytännön toteutuksen todellisuuden ja tutkimuskysymysten kannalta olennaisten vastausten löytämiseksi. Jokaisessa vaiheessa itse Kubernetes-alustan asennuksesta lähtien alustoista pyritään etsimään eroja esimerkiksi helppouden sekä suorituskyvyn välillä sekä muita huomioita alustoihin liittyen. Nämä asiat pyritään myös taulukoimaan ja pisteyttämään.

Tutkimuksessa käytettävät ohjelmat eivät kuvaa muita ohjelmia, niiden toimivuutta tai vaatimuksia, joten tulokset voivat vaihdella käytettävien ohjelmien mukaan. Kyseiset ohjelmat ovat kuitenkin jotain, mitä yritys voi haluta käyttää. Samalla nämä ohjelmat ovat myös helposti saatavilla tutkimusta varten ja ovat jo ennestään tuttuja tutkimuksen tekijälle.

Myöskään tutkimuksessa käytettävä testiympäristö ei välttämättä ole yrityksen testaus- tai tuotantoympäristöä vastaava ja käytettävissä olevat resurssit voivat poiketa hyvinkin suuresti yrityksestä riippuen. Kyseistä

testiympäristöä käytettiin, koska myös yritykset voivat haluta jakaa omat laskentaresurssinsa virtuaalisiin koneisiin ja kyseinen ympäristö oli käytettävissä tutkimuksen aikana.

Opinnäytetyössä tutkittavat alustat kyetään jakamaan asennusohjelmiin sekä distribuutioihin. Kuberneteksen asennusohjelmiksi luokitellaan ohjelmat, jotka lataavat ja asentavat “puhtaan” Kuberneteksen sen viimeisimmistä asennustiedostoista. Distribuutiot ovat Kubernetekseen pohjautuvia ohjelmia, jotka voidaan asentaa pilvipalveluun, virtuaaliselle koneelle, tai raakaraudalle (bare metal) ja jotka sisältävät korjaustiedostot alkuperäisestä lähteestä (upstream codebase). (Cloud Native Computing Foundation, n.d.; kdopen, 2015)

Tutkittaviksi alustoiksi valittiin yhdeksän Cloud Native Computing Foundationin (CNCF) sertifioimaa, eniten GitHub-tähtiä saanutta avoimen lähdekoodin alustaa. GitHub-tähdet helpottavat käyttäjiä löytämään kiinnostavat projektit uudestaan. Tähdillä osoitetaan myös arvostusta projektien ylläpitäjiä kohtaan ja monet GitHubin hakutuloslajitteluista käyttävät myös tähtiä projektien järjestämiseen. (GitHub Help, n.d.)

Opinnäytetyön tutkimuksen aikaan 4.2.2020 nämä alustat olivat asennusohjelmiksi luokiteltavat Minikube (17 303 tähteä), Kops (10 778 tähteä), Cubespray (7 986 tähteä) ja Kind (4 439 tähteä) sekä distribuutioiksi tai “distroiksi” luokiteltavat Rancher (13 608 tähteä), K3s (11 434 tähteä), OpenShift (7 194 tähteä), Kubesphere (2 719 tähteä) ja MicroK8s (2 686 tähteä). Alustoista Minikube sekä Kind ovat ainoat, jotka ovat suunniteltu pelkästään paikalliseen käyttöön.

Tutkimuksen ensimmäisellä kierroksella valittuja alustoja analysoidaan niistä löytyvien aineistojen perusteella. Alustoista etsitään tietoja niiden julkaisijoiden verkkosivuilta sekä GitHubista. Pääasiallisena tarkkailun kohteena ovat alustojen ominaisuudet ja toiminnot, kuten tukeeko alusta useita mastereita sekä nodeja, sisältääkö alusta oman hallintanäkymän (dashboard), onko alustaan sisäänrakennettuna DNS ja mitkä ovat alustan järjestelmävaatimukset. Nämä tullaan listaamaan jokaisen alustan kohdalta taulukkoon, josta on helposti nähtävissä alustojen eroavuudet. Tämän vertailun pohjalta pyritään löytämään 2-4 alustaa, jotka etenevät tutkimuksen toiselle kierrokselle.

Tutkimuksen toiselle kierrokselle valituista alustoista jokainen otetaan käyttöön omassa virtuaalikoneessaan ja jokaiselle masterille annetaan käyttöön yksi oma node. Ensimmäisenä jokaiseen järjestelmään asennetaan WordPress-työympäristö. Wordpress on avoimen lähdekoodin sisällönhallintajärjestelmä, joka mahdollistaa verkkosivujen luonnin ja ylläpidon (Kinsta, n.d.).

Itse Kubernetes-alustojen asennuksen jälkeen toiselle kierrokselle päässeille alustoille asennetaan Rocket.Chat. Rocket.Chat on avoimen

lähdekoodin ohjelmisto, joka on tarkoitettu ryhmän sisäiseen kommunikointiin. Sen tavoitteena on olla korvaava alusta esimerkiksi sähköpostille ja muille keskustelualustoille. (Rocket.Chat, n.d.)

Toisen kierroksen alustoja vertaillaan keskenään ja niille tullaan antamaan pisteet asteikolla 0-5 kategorioista *alustan asennuksen helppous*, *Wordpress-ympäristön asennus*, *Rocket.Chat-ympäristön asennus* ja *toimivuus oppilaitoksen vCommander-ympäristössä*. Yksittäisen alustan kokonaispistemäärä koostuu näistä osioista saaduista pisteistä ja on siis maksimissaan 20 pistettä. Pisteyttäessä huomioidaan helppokäyttöisyys sekä toimivuus sekä itse alustojen, että niille asennettavien ohjelmistojen osalta.

6 TUTKIMUKSEN ENSIMMÄINEN KIERROS

Tutkimuksen ensimmäisen kierroksen alustat ominaisuuksineen taulukoitiin vertailun ja toisen kierroksen alustojen valinnan helpottamiseksi. Kyseinen taulukko löytyy opinnäytetyön lopusta liitteenä (Liite 1).

Yhdeksää eniten GitHub-tähtiä saanutta alustaa vertaillessa huomattiin, että osa vaihtoehdoista tukee useita nodeja ja myös useita mastereita, mutta esimerkiksi Minikubessa kumpikaan edellä mainituista ei ole mahdollista. Tästä huolimatta se oli kuitenkin suosituin avoimen lähdekoodin alusta perustuen GitHubissa annettuihin tähtiin opinnäytetyön tutkimusaikana.

Oli myös nopeasti selvää, että RedHatin tarjoama Openshift oli vertailtavista ainoa maksullinen. Openshift sisältää 60 päivää kestävän ilmaisen kokeilun, jonka jälkeen käyttö muuttuu maksulliseksi. Koska tavoitteena oli löytää edulliset vaihtoehdot yritykselle, oli selvää, että alustoista ainoa maksullinen pudotettaisiin heti pois pelistä.

Alustojen julkaisijoita tutkittaessa havaittiin, että yhdeltä julkaisijalta on tarjolla useita eri vaihtoehtoja Kuberneteksen käyttöön. Vertailuun haluttiin ottaa vain yksi alusta per julkaisija kattavamman kuvan saamiseksi, joten vähemmän tähtiä saaneet alustat samoilta julkaisijoilta jätettiin pois. Näin ollen pois pudotettiin Rancherin valmistama K3s, Kuberneteksen julkaisema Kops, sekä Kubernetes SIGs julkaisijan alla oleva Kind. Alustoista viimeisenä mainittu on kehittäjän sanoin vielä toistaiseksi “work in progress”, jonka vuoksi alusta olisi joka tapauksessa pudotettu pois, sillä ei voida olla varmoja onko alusta tarpeeksi vakaa yrityksen kehitys- sekä testaustoimintaan. Jäljelle tässä vaiheessa vertailua jäivät siis Minikube, Rancher, Kubespray, Kubesphere sekä MicroK8s.

Pelkän vertailutaulukon (Liite 1) perusteella Minikube vaikutti hyvin suppealta verrattuna muihin vaihtoehtoihin, sillä se ei tue useita nodeja tai mastereita, eikä sen käyttö onnistu paikallisen koneen lisäksi yhdessäkään pilvipalvelussa. Näistä huolimatta Minikube on kuitenkin suosituin avoimen lähdekoodin alusta saaden lähes 4000 tähteä enemmän GitHubissa kuin toiseksi suosituin Rancher. Tästä syystä Minikube haluttiin ottaa mukaan myös tutkimuksen toiselle kierrokselle voidaksemme selvittää, onko Minikubessa jotain ominaisuuksia, jotka eivät näy vertailutaulussa tai tule selkeästi esille ilman, että alustaa testataan käytännössä. Minikube jäi vertailun ainoaksi asennusohjelmaksi.

Toiseksi alustaksi valittiin distribuutio Rancher, joka on vaihtoehdoista toiseksi suosituin ja kaikkien kolmen suurimman pilvialustan tukema siltä varalta, jos yritys haluaa joskus tulevaisuudessa siirtyä paikallisilta koneilta pilveen. Rancher tarjoaa myös korkean saavutettavuuden (high availability), eli mahdollisuuden useisiin mastereihin sekä nodeihin, jota

Minikube ei tarjoa. Tämän lisäksi Rancher tarjoaa myös käyttäjätukea maksua vastaan, joka voi olla hyödyksi ongelmatilanteessa, josta yritys haluaa vain nopeasti eroon.

Kolmanneksi alustaksi tutkimuksen toiselle kierrokselle haluttiin ottaa Kubesphere. Se on Rancherin tapaan distribuutio ja tarjoaa samat ominaisuudet kuin Rancher lukuun ottamatta maksullista käyttäjätukea. Kubespherea mainostetaan myös helppokäyttöisenä tuotantotason orkestrointijärjestelmänä. Mielenkiinnon kohteena olivat olisiko alusta todellisuudessa helppokäyttöisempi kuin kilpailijansa ja kuinka erilainen vajaat 3000 tähteä saanut alusta on verrattuna kahteen muuhun alustaan, joilla molemmilla tähtiä on yli 10 000, vai huomaako alustan pienempää suosiota millään tapaa.

7 TUTKIMUKSEN TOINEN KIERROS

Paikallisena ympäristönä testien aikaan toimivat Ubuntu server 18.04 -virtuaalikoneet, jotka haettiin HAMKin vCommander-palvelusta. Kullekin virtuaaliselle koneelle annettiin sama määrä prosessoriytimiä, tallennustilaa sekä työmuistia varmistaaksemme, etteivät kyseiset muuttujat pääse vaikuttamaan alustojen käyttöön tai asennukseen.

Tämän opinnäytetyön testeissä käyttömuistia jokaiselle koneelle annettiin 5GB, tallennustilaksi liitettiin 100GB kovalevyt sekä jokainen kone sai käyttöönsä kaksi prosessoriydintä. Koska virtuaalikoneet ovat osa HAMKin opetusympäristöä, koneet eivät ole täydellisesti käyttäjän hallittavissa ja tämä voi osaltaan vääristää tutkimuksesta saatavia tuloksia ja vaikeuttaa hallintaa ja/tai asennusprosesseja.

Alustojen testaukseen on myös varattu kaksi HAMKin verkon ulkopuolista tietokonetta, jotka molemmat on varustettu kahdeksan ytimen prosessoreilla, 32GB käyttömuistilla ja 250GB tallennustilalla. Koneisiin asennetaan myös virtuaalisilla koneilla käyttöjärjestelmänä toimiva Ubuntu server 18.04.

Ensimmäisenä kun alustoja alettiin asentamaan testausympäristöön, huomatiin aiemmin löydettyjen minimivaatimusten Kubespheren tapauksessa olevan vähimmäisvaatimuksia “all-in-one -paketin” ajamiseen, joka on ensisijainen versio, jonka asennusohjeeseen etusivun “Get Kubesphere” -painike vie. All-in-one -paketissa ei ole useita nodeja sekä mastereita vaan vain pelkästään yksittäinen kone, joka on jaettu sekä masteriin että nodeen. Dokumentaation takaa löytyy myös ohje, jonka avulla näillä samoilla vähimmäisvaatimuksilla kyetään myös ajamaan useammasta koneesta koostuvaa klusteria, joka sisältää useita nodeja sekä mastereita, muttei ole täysi versio Kubespherestä. Nämä asennukset tapahtuvat ns. suoraan raudan päällä olevaan käyttöjärjestelmään.

Jos halutaan ottaa käyttöön “täysi ympäristö” useilla nodeilla ja mastereilla, tulee käytössä olevissa koneissa olla aiemmin mainitun neljän gigan suuruisen RAM-muistin sijaan yhteensä vähintään 16 gigaa RAM-muistia. Myös prosessoriydinten lukumäärä tulee minimissään nelinkertaistaa kahdesta ytimestä yhteensä kahdeksaan ytimeen. Täysi versio sisältää lisäkomponentteja kuten esimerkiksi oman sovelluskaupan, joita ei ole oletuksena aktivoitu usean noden ja masterin versiossa, mutta muuten nämä kaksi versiota vastaavat toisiaan. Ohje tähän täyteen versioon löytyy dokumentaatiosta kohdan “vapaavalintaisten komponenttien asennus” alta.

Myös Rancherin minimivaatimukset, jotka on aiemmin listattu taulukkoon, ovat vähimmäisvaatimuksia versiolle, joka ei ole “täysi Rancher”. Vähimmäisvaatimuksilla kyetään ajamaan Rancher-klusteria pelkästään Dockerin avulla. Rancherin master käynnistetään tällöin omaan erilliseen Docker-

konttiinsa, johon on mahdollista kuitenkin liittää nodeja. Nodejen lisäys tapahtuu ajamalla kaikissa nodeissa Rancher Desktopin generoima Docker-komento. Näin ollen myös nodet toimivat Docker-konteissa asennettaessa Rancher minimivaatimuksilla, toisin kuin Kubespheren tapauksessa. Mikäli Rancherista halutaan ajaa täyttä versiota, jolloin masterit ja nodet eivät enää ole Docker-konttien sisällä, on käyttömuisti tuplattava kahdeksaan gigaan ja prosessoriytimiä tulee olla käytettävissä kaksi kappaletta aiemman yhden sijaan.

Minikube on testattavista alustoista ainoa, jonka resurssivaatimuksista ei paljastunut uutta tietoa. Minikube on myös näistä kolmesta ainoa alusta, josta on olemassa vain yksi asennusversio. Variaatiota kuitenkin luovat asennuksessa esiin tulevat erilaiset ajurimahdollisuudet.

Muuttuneiden tietojen pohjalta tehtiin uusi vertailutaulukko näiden kolmen alustan välillä uusien tietojen visualisoimiseksi (Liite 2).

7.1 **Alustojen asennus**

Asennuksista suoraviivaisin on Rancherin Docker-versio, joka vaatii vain Docker-asennuksen sekä yhden komennon. Tämä asennustapa tuotti kuitenkin ongelmia oppilaitoksen verkkoon sidotussa vCommander -ympäristössä. Rancher-klusterin sai vCommander-ympäristön virtuaalikoneilla pystyyn onnistuneesti, mutta järjestelmä saattoi rikkoutua täysin itsestään parin tunnin kuluessa käynnistyksestä. Omalta osaltaan ongelmia lisäsi vCommander-ympäristössä olleiden virtuaalikoneiden IP-osoitteiden muutokset. Lopulta Rancher päädyttiin asentamaan vain oppilaitoksen verkon ulkopuolisille fyysisille koneille ja testaus toteutettiin käyttäen niitä.

Rancherin täyden version asennus oli puolestaan kaikista testatuista alustoista ja asennusmuodoista selkeästi haastavin. Ennen itse Rancherin asennusta pohjalle oli asennettava Docker ja kokonainen Kubernetes-klusteri, joiden lisäksi tarvittiin myös kubectl sekä Helm. Koko asennuksen dokumentaatiota joutui lukemaan huomattavasti tarkemmin verrattuna kilpailijoihin ja osasta asioita oli vain nopea maininta muun tekstin seassa. Esimerkiksi vaadittavien SSH-avainten luonti koneiden yhdistämistä varten jäi ensimmäisellä asennuskerralla kokonaan huomaamatta ja tämä hidasti asennusprosessia. Vielä parin asennuskerrankin jälkeen dokumentaatio on osittain hieman hämmentävä ja asennukseen uppoaa huomattavasti enemmän aikaa muihin alustoihin verrattuna. Docker-versiosta poiketen täyttä versiota ei kyetty asentamaan ollenkaan vCommander-ympäristöön.

Mainittujen asioiden lisäksi Rancherin asennuksessa ilmeni jokaisella asennuskerralla kaksi samaa virheilmoitusta, joista eroon pääseminen oli helppoa, mutta omalla tavallaan hämmentävää. Toinen virheilmoituksista

ilmoitti epäonnistuneesta komponentin käynnistämisestä Rancheria asentaessa:
FATA[0239] Failed to get job complete status for job
rke-network-plugin-deploy-job in namespace kube-system

Korjaus ongelmaan oli kirjaimellisesti saman komennon ajaminen uudestaan tekemättä yhtäkään muutosta komentoon taikka asennuksessa käytettäviin tiedostoihin, kuten GitHub-käyttäjä iahmad-khan kommentissaan kertoi (iahmad-khan, 2019). Toinen virheilmoituksista ilmestyi, kun Rancher-klusteri saatiin pystyyn ja koski serverin URL-osoitetta:
Waiting for server-url setting to be set

Tämä korjattiin klikkaamalla Rancherin web-käyttöliittymästä klusterin editointipainiketta ja tämän jälkeen klikkaamalla “Tallenna muutokset” -painiketta koskematta yhteenkään asetukseen GitHub-käyttäjän derWeihnachtsmann ohjeistuksella (derWeihnachtsmann, 2019).

Käytössä olleessa fyysisten koneiden testausympäristössä Rancherin web-käyttöliittymän näkeminen ensimmäisellä kerralla vaati graafisen käyttöliittymän asentamista masterina toimivalle koneelle ja käyttöliittymään navigoimista kyseisen koneen selaimella. Ensimmäisen kirjautumisen jälkeen käyttöliittymä saatiin näkyviin myös muilla saman verkon laitteilla. Tämä todennäköisesti johtui siitä, että testeissä domain-nimenä jouduttiin käyttämään localhostia.

Kubespheren kaksi eri asennusvaihtoehtoa, all-in-one ja multi node erosivat toisistaan vain hyvin vähän. Kumpikaan asennustavoista ei vaatinut muita ohjelmia tai komponentteja pohjalle. All-in-one -asennuksessa ladattiin kaikki Kubespheren asennustiedostot yhtenä pakettina ja purettiin ne masterina toimivalle koneelle vain yhtä komentoa käyttäen. Tämän jälkeen ajettiin paketin sisältämä install.sh -skripti ja valittiin annetuista vaihtoehdoista all-in-one, jota seurasi pelkkää odotusta skriptin ladatessa ja asentaessa kaiken tarvittavan koneelle.

Kubespheren multi node -asennustavan ainoana erona oli hosts.ini -tiedoston muokkaus ennen install.sh -skriptin ajoa. Tiedosto sisälsi kaksi eri tapaa yhdistää käytettävät koneet toisiinsa, joista toinen oli tarkoitettu sudo-oikeuksilla varustettuja käyttäjiä varten ja toista käytettiin root-käyttäjien kanssa. Tiedostoon muokattiin käytössä olevien koneiden IP-osoitteet, salasanat ja käyttäjätunnukset toiseen mainituista tavoista. vCommander-ympäristö vaati non-root -käyttäjille tarkoitettua asennustapaa, kun taas oppilaitoksenn verkon ulkopuolisilla fyysisillä koneilla root-käyttäjille suunnattu tapa oli toimiva. Jälkimmäinen tapa vaati myös nodena toimivan koneen root-käyttäjän salasanan vaihtoa sekä SSH:n konfiguraatiotiedoston muokkausta niin, että root-kirjautuminen etäyhteydellä sallittiin. Näiden muutosten jälkeen install.sh -skripti kyettiin ajamaan ja asennusvaihtoehdoista valittiin “multi node”, jonka jälkeen skripti latasi ja asensi jälleen kaiken tarvittavan automaattisesti koneelle.

Minikuben asennus sijoittui helppoudeltaan muiden alustojen ja asennustapojen väliin. Asennus vaati pohjalle Dockerin sekä kubectl-asennuksen, jonka jälkeen ladattiin Minikuben asennustiedosto, annettiin tiedostolle suoritusoikeudet ja siirrettiin tiedosto oikeaan sijaintiin koneella. Tämän jälkeen tiedosto ajetaan ja se asentaa Minikuben koneelle. Asennuskomennon yhteydessä tulee määrittää ajuri Minikubea varten, jonka tutkiminen vei hetken aikaa. Lopulta päätin tässä testauksessa asentaa Minikuben ilman ajuria. Ensimmäisellä kerralla vCommander-ympäristön ulkopuolelle asennettaessa Minikube asentui kuitenkin käyttäen Docker-ajuria tarvitun conntrack-työkalun puuttuessa kyseiseltä koneelta. Tämän asentamalla Minikuben sai myös käyttöön ilman Docker-ajuria.

Kun alustat yksin olivat pystyssä fyysisillä koneilla, ne olivat hyvin lähellä toisiaan resurssien käytöltään, eikä merkittäviä eroja ollut. Ainoa suuri ero oli se, että Kubesphere käyttää oletuksena myös master-konetta podien pyörittämisessä ja näin ollen kahdella koneella podien maksimimäärä oli 220 kappaletta verrattuna muiden alustojen oletuksena olevaan 110 podin maksimimäärään.

7.2 **Wordpress- ja Rocket.Chat-palveluiden asennus**

Wordpress ja sen käyttämä MySQL asennettiin jokaiselle alustalle käyttäen Kuberneteksen virallisia ohjeistuksia (Kubernetes, n.d.a). Kyseisessä ohjeessa käytettiin Minikubea alustana, joten Minkikuben kanssa työskennellessä kyseistä ohjetta ei tarvinnut soveltaa millään tapaa ja sekä Wordpressin että MySQL:n nosto sujui ongelmitta. Kyseinen asennus toteutettiin kokonaisuudessaan terminaalista käsin koskematta Minikuben graafiseen web-käyttöliittymään.

Kubespheren kohdalla Kuberneteksen ohjeistusta jouduttiin soveltamaan vasta julkaistessa Wordpressin kirjautumissivua näkyville verkkoon. Wordpress-palvelun NodePort otettiin käyttöön Kubespheren web-käyttöliittymän puolelta, jolloin Wordpress saatiin näkyviin verkon kaikille koneille ja tämän toteuttaminen kävi varsin helposti. Prosessi oli sama Kubespheren molemmilla asennustavoilla.

Rancherin täyden version kanssa Wordpressin ja MySQL:n asennus vastasi Kubespherellä toteutettua asennusta. Myös Rancherin tapauksessa NodePort tuli ottaa käyttöön ja toteutus tapahtui myös web-käyttöliittymän kautta. Rancherin kohdalla kyseisen asetuksen löytäminen web-käyttöliittymän puolelta oli kuitenkin itselle hieman hankalampaa ja vaati enemmän aikaa etsimiselle.

Rancherin Docker-versiolla Wordpressiä ja MySQL:ää ei saatu opinnäytetyön aikana toimimaan. Ensimmäiset ongelmat koskivat Kubernetes-ohjeistuksessa luotavia yaml-tiedostoja ja niiden sijainteja (Kubernetes, n.d.a). Kyseisillä tiedostoilla ei saatu luotua palveluita Rancheriin ollenkaan

riippumatta siitä olivatko tiedostot sisällä Rancherin kontissa vai eivät. Palvelut saatiin kyllä tämän jälkeen luotua Rancherin web-käyttöliittymän kautta pelkän käyttöliittymän tutkimisen ja kokeilun tuloksena, mutta ongelmaksi nousivat pysyvät tallennustilat (persistent volumes) ja niiden oikeudet, joita ei kyetty korjaamaan opinnäytetyön aikana. Ilman kyseisten ongelmien korjausta Wordpress ja MySQL eivät suostuneet käynnistymään oikein.

Rocket.Chat-palvelun asennuksessa käytettiin Rocket.Chatin itsensä julkaisemaa Helm-ohjeistusta (Rocket.Chat, n.d.a). Kyseistä asennuskomentoa jouduttiin kuitenkin muokkaamaan hieman asennuksen onnistumiseksi alustasta riippumatta Rocket.Chatin sivuilta löytyvän asennuskomennon ollessa ilmeisesti hieman vanhentunut: name-attribuutti siirrettiin komennon lopusta kaikkien attribuuttien etupuolelle ja attribuutteihin lisättiin kohta volumePermissions.enabled=true kuten GitHub-käyttäjä marcosbc on kommentissaan ehdottanut (marcosbc, 2019).

Helm-asennustapa vaati Helmin asennuksen alustoille, joille sitä ei oltu alustan asennuksen yhteydessä asennettu. Wordpressin tapaan myöskään Rocket.Chat ei suostunut asentumaan Rancherin Docker-versioon. Ongelmana olivat Rocket.Chatin puutteelliset oikeudet, jolloin se ei kyennyt asentamaan omia palveluitaan. Kyseisiä oikeusongelmia ei kyetty korjaamaan opinnäytetyön aikana.

Yllätykseksi vastaavia ongelmia oli myös Minikuben kanssa vCommander-testiympäristössä eikä Rocket.Chattia saatu toimimaan. Kun Minikube myöhemmin asennettiin oppilaitoksenn verkon ulkopuoliselle koneelle, Rocker.Chat saatiin kuitenkin nostettua onnistuneesti ja se saatiin toimimaan myös Minikubella ilman ongelmia ja muita toimenpiteitä.

Kubespheren ja Rancherin täyden version kohdalla Helm-komento ei luonut automaattisesti pysyviä tallennustiloja, joten ne jouduttiin luomaan manuaalisesti jälkikäteen. Tämän jälkeen ongelmaksi koituivat riittämättömät oikeudet, jotka olivat ongelmana myös Rancherin Docker-version sekä vCommander-ympäristöön asennetun Minikuben kanssa. Ongelmat kuitenkin onnistuttiin korjaamaan Kubespheren ja Rancherin täyden version kohdalla volumePermissions-attribuutilla komennossa sekä muokkaamalla pysyvien tallennustilojen sijaintikansion oikeuksia, jota GitHub-käyttäjä jmpolvera oli käyttänyt vastaavan ongelman ratkaisuun (jmpolvera, 2020).

7.3 **Toimivuus oppilaitoksen vCommander-ympäristössä**

Asennus oppilaitoksen tarjoamaan ja hallitsemaan vCommander-ympäristöön vaikutti jokaiseen alustaan. Tämä johtui todennäköisesti siitä, että koneet eivät olleet koskaan täydessä hallinnassa, vaan oppilaitoksen palomuurisäännöt ja muut säädöt vaikuttivat koneisiin.

Myös koneiden IP-osoitteet vaihtelivat rajusti ja vaikeuttivat työskentelyä. Rancherin Docker-version kohdalla IP-osoitteiden vaihtelua ei kyetty toimivasti korjaamaan, joka johti edellisten konttien tuhoamiseen jokaisella kerralla, kun toinen Rancherin käytössä olleista koneista vaihtoi IP-osoitetta.

Sama IP-osoitteen vaihtuminen vaivasi myös Kubesphere-klusteria, mutta ongelma oli helppo korjata. Vaihtamalla klusterin konfiguraatiotiedostoon oikeat IP-osoitteet ja suorittamalla skriptin, klusteri saatiin palautettua toimivaksi, eikä yksikään klusterissa olleista podeista tuhoutunut, vaan jatkoi korjauksen jälkeen normaalia toimintaansa. Myös podeissa oleviin palveluihin tehdyt muutokset säilyivät, eikä tietoja menetetty.

Minikube oli testatuista alustoista ainoa, johon IP-osoitteen muutokset eivät vaikuttaneet. Tämä johtuu siitä, että Minikubella luotava klusteri kykenee sisältämään vain yhden koneen, joten masterien ja nodejen välinen liikennöinti ei pääse häiriintymään IP-osoitteiden muutoksista.

Muut ongelmat liittyivät klustereissa pyörivien järjestelmien paljastamiseen muille saman verkon koneille. Oppilaitokseun vCommander-testiympäristössä palveluita ei kyetty avaamaan isäntäkoneen ulkopuolelle muille koneille, vaikka koneet olisivat saman verkon sisällä. Kubespheren ja Minikuben kohdalla tämä kyettiin kiertämään käyttämällä SSH-tunneleita lukuun ottamatta Minikuben web-käyttöliittymää, joka saatiin näkyviin vain isäntäkoneen selaimen kautta. Koska Rancher ei ollut tarpeeksi vakaa vCommander-ympäristössä palveluiden asentamiseksi, opinnäytetyön aikana ei kyetty selvittämään olisiko myös Rancher toiminut kyseisessä ympäristössä kahden muun alustan tapaan ja vaatinut SSH-tunneleiden käyttöä.

Fyysisille oppilaitoksen verkon ulkopuolisille koneille asennettaessa SSH-tunneleita ei enää vaadittu, vaan kaikki palvelut olivat näkyvissä myös muille verkon koneille. Minikuben kohdalla myös web-käyttöliittymä saatiin näkyviin isäntäkoneen ulkopuolelle Computing for Geeks -sivuston asennusohjeistuksien avulla, joissa web-käyttöliittymälle asetetaan pysyvä portti NodePort ja luodaan admin-käyttäjä käyttöliittymää varten (Mutai, 2020a; Mutai, 2020b).

8 PISTEYTYS JA TULOKSET

Alustat pisteytettynä ja taulukkoon sijoitettuna löytyvät luvun lopusta (Kuva 5). Alustojen sekä testipalveluiden asennukset sekä niiden toiminta vCommander-ympäristössä pisteytettiin taulukkoon asteikolla 0-5, jossa numero 0 tarkoittaa, ettei asennus onnistunut tai toiminut vCommander-ympäristössä ja numero 5 tarkoittaa helppoa ja vaivatonta asennusta, sekä toimintaa suhteutettuna oletuksiin. Tämä tarkoittaa Wordpress-asennuksen yhteydessä sitä, että otetaan huomioon se fakta, että kyseinen ohjeistus oli kirjoitettu Minikubelle, joten Ḿinikubelle itselleen kirjoitettu komento ei toimi muissa ympäristöissä. Rocket.Chatin kohdalla huomioitiin virallisen ohjeistuksen komennon mahdollinen vanhentuminen.

Aiemmassa kappaleessa selitettyjen seikkojen perusteella alustojen asennuksen helppous pisteytettiin niin, että asennuksien helppous järjestyksessä helpoimmasta vaikeimpaan on seuraava: Rancher (Docker), Kubesphere (all-in-one), Minikube, Kubesphere (multi node), Rancher (täysi). Kubespheren multi node -versio jäi Minikuben taakse vain siitä syystä, että sen asennus vaati konfiguraatiotiedoston muokkausta sekä ensimmäisellä kerralla kokeilua kumpi tiedoston tarjoamista tavoista on oikea. Minikube puolestaan jäi Kubespheren all-in-one -asennuksen taakse ensiasennuksella hämmentäneen ajurivalinnan vuoksi.

Wordpress-ympäristön asennus jokaiselle alustalle oli yhtä helppo ja kivuton huomioiden ohjeen olevan Minikubea varten kirjoitettu. Rancherin Docker-versio ei saanut pisteitä, koska Wordpress-asennus ei onnistunut missään vaiheessa.

Rocket.Chat-ympäristön asennuksesta vain Minikube sai täydet pisteet, sillä Rancherin täyden version sekä Kubespheren molempien versioiden kohdalla Rocket.Chatin tarvitsemia pysyviä tallennustiloja ei luotu Helm-komennon seurauksena, vaan ne jouduttiin luomaan manuaalisesti jälkikäteen. Rancherin Docker-versio jäi jälleen ilman pisteitä.

Toimivuudesta oppilaitoksen tarjoamassa vCommander-ympäristössä yksikään alustoista ei saanut täysiä pisteitä. Kubespheren asennuksien kohdalla pisteitä vietiin palveluiden näyttämiseksi tarvittujen SSH-tunneleiden vuoksi, Minikuben kohdalla pisteitä vietiin myös samasta syystä ja sen lisäksi vCommander-ympäristössä toimimattomasta Rocket.Chat-asennuksesta. Rancherin täysi versio jäi ilman pisteitä, ja Docker-versiolle annettiin yksi piste, koska se saatiin edes hetkellisesti pystyyn vCommanderin virtuaalikoneille toisin kuin täysi versio.

Suurimman pistemäärän vertailluista alustoista kokosi Kubespheren all-in-one asennusvaihtoehto. Rancherin kokonaispisteet jäivät testeissä vertailukohteita alhaisemmiksi.

	Minikube	Rancher (Docker)	Rancher (täysi)	Kubesphere (all-in-one)	Kubesphere (multinode)
Alustan asennuksen helppous	3	5	1	4	2
Wordpress -ympäristön asennus	5	0	5	5	5
Rocket.Chat -ympäristön asennus	5	0	4	4	4
Toimivuus koulun vCommander -ympäristössä	3	1	0	4	4
Yhteispisteet:	16	6	10	17	15

Kuva 5 Alustat pisteytettynä

9 JOHTOPÄÄTÖKSET

Tehtyjen testien perusteella voidaan sanoa, että Minikube soveltuu hyvin asennettavaksi henkilökohtaiselle koneelle omia projekteja ja harjoittelua varten. Se on kevyt, eikä vie paljoa resursseja koneelta, joten sen asennus ihan tavalliselle kannettavalle onnistuu hyvin. Minikubelle löytyy paljon ohjemateriaalia ja suuri osa verkosta löytyvästä opetusmateriaalista, joka lähtee aivan perusteista, on suunnattu Minikubelle. Alustan GitHubissa saama huomio perustunee näihin seikkoihin.

Yritykselle Minikube ei ehkä kuitenkaan ole sopiva vaihtoehto, sillä Minikube kykenee käyttämään vain yhtä konetta, eikä näin ollen mahdollisuuksia useisiin mastereihin tai nodeihin ole. Tämä tarkoittaa myös hyvin rajallisia laskentaresursseja, joten raskaiden järjestelmien tai suurien kappalemäärien pyörittäminen on hankalaa ja tiettyyn pisteeseen päästessä mahdotonta. Minikube voisi kuitenkin soveltua joidenkin pienien ja yksittäisien osien testailuun, joita vain yhden henkilön tarvitsee työstää kerrallaan.

Testailun aikana havaittiin myös, että Minikuben web-käyttöliittymä ei ole niin hyödyllinen, kuin kilpailevien alustojen. Käyttöliittymästä on mahdollista tarkkailla resursseja, mutta ei juurikaan muuta. Kilpailevien alustojen web-käyttöliittymillä kykeni tekemään tehtäviä, kuten pysyvien tallennustilojen luontia, joka Minikubella oli toteutettava terminaalin kautta.

Rancher oli tutkituista alustoista toiseksi suosituin GitHubissa. Toisin kuin Minikube, Rancher kykenee käyttämään useiden koneiden laskentatehoa, joten sen avulla on mahdollista pyörittää huomattavasti raskaampia järjestelmiä ja suurempia kappalemääriä podeja. Se soveltuu niin omaan, kuin yrityksenkin käyttöön. Rancher tukee myös asennusta pilveen, joten jos yritys haluaa käyttää oman raudan sijaan pilvipalveluntarjoajien laskentatehoa, myöskin se on täysin mahdollista.

Minikuben tapaan myös Rancherille löytyy laajasti ohjemateriaalia sekä foorumikeskusteluita. Näiden lisäksi Rancher tarjoaa myös ainoana tutkituista alustoista maksullista tukea asiakkailleen, josta voi olla hyötyä mahdollisten hankalampien ongelmatilanteiden kohdalla.

Testauksen aikana Rancherin versiot eivät kuitenkaan suostuneet toimimaan oppilaitoksen tarjoamassa vCommander-ympäristössä. Sen perusteella voinee tehdä oletuksen, että Rancher ei välttämättä ole toimin vaihtoehto yritykselle, jonka palomuurisäännöt ovat hyvin tiukat ja jossa halutaan pitää Kubernetes omilla servereillä pilvipalvelutarjoajien sijaan.

Rancher on myös vaatinut testatuista alustoista huomattavasti eniten työtä ja aikaa, eikä Rancherin dokumentaatio ole helpoimmasta päästä ymmärtää. Tämä voi osaltaan tuoda haasteita yritykselle, jolla ei ole

entuudestaan kokemusta Kuberneteksesta tai työntekijöitä, joille Kubernetes ja Rancher ovat tuttuja.

Huomattavasti vähemmän GitHub-tähtiä saanut Kubesphere pärjäsi hyvin testeissä suositumpia Minikubea ja Rancheria vastaan, josta voinee päätellä, etteivät tähdet välttämättä kerro suoraan ominaisuuksista, vaan osa tähdistä voi tulla pelkästään tunnettavuudesta. Rancherin tapaan Kubesphere soveltuu niin yrityksen, kuin yksityishenkilönkin käyttöön ja asennus on mahdollista myös pilveen, joskin vain yhtä alustaa tuetaan virallisesti.

Kubespheren asennus ja ylläpito on helpompaa verrattuna Rancheriin, joka on hyvin vastaava ominaisuuksiltaan. Alustan web-käyttöliittymä on selkeä ja helppokäyttöinen sekä soveltuu muuhunkin kuin pelkkään resurssien tarkkailuun. Näistä syistä Kubesphere nousi tutkitusta kolmikosta omaksi henkilökohtaiseksi suosikiksi.

Kubespheren pienemmästä suosiosta kuitenkin kertovat hyvin vähäiset ohjeistukset kyseiselle alustalle, jonka seurauksena mahdollisesti käytettäviä ohjeistuksia voi joutua hieman soveltamaan, sillä kyseisiä ohjeita ei välttämättä löydä Kubespherelle. Tämän lisäksi suuri osa foorumikeskusteluista on käyty kiinaksi, joten ongelmatilanteiden ratkaisu voi olla haastavaa verrattuna esimerkiksi Rancheriin. Nämä tekijät vaikeuttavat alustan käyttöä yrityksessä, joka ei ole ollut aiemmin tekemisissä Kuberneteksen kanssa, mutta alustan käyttäjäystävällisyys tuo omalta osaltaan helppoutta käyttöön.

Omaksi suosikiksi opinnäytetyön aikana nousi Kubesphere sen käyttäjäystävällisen lähestymistavan vuoksi. On kuitenkin todennäköisempää, että yritys kallistuu enemmän Rancherin puolelle paremmin saatavissa olevien ohjeistuksien sekä maksullisen käyttäjätuen olemassaolon myötä. Opinnäytetyön tekovaiheessa käynnissä oleva USA:n ja Kiinan välinen kauppasota sekä huolet tietojen vuotamisesta Kiinaan voivat vaikuttaa myös Kubespheren suosioon kiinalaisten yritysten ja sovellusten ollessa kiistanalainen aihe ympäri maailman (Rydman, 2020; Lyons, 2020; Ruvic, 2020).

Yrityksellä on siis hyvin monia vaihtoehtoja tilanteessa, jossa halutaan ottaa Kubernetes käyttöön liiketoiminnassa, tuotannossa sekä testauksessa ja tässä opinnäytetyössä esille tuodut alustat ovat vain pintaraapaisu kaikista mahdollisuuksista.

On täysin mahdotonta antaa yksiselitteinen vastaus siitä, mikä alusta soveltuu parhaiten yrityksen käyttöön. Alustan valintaan vaikuttavat yrityksen tarpeet, resurssit sekä preferenssit, eikä ole mahdollista antaa myöskään suoraa vastausta siihen, minkä tapaisille yrityksille juuri tässä opinnäytetyössä käsitellyt alustat sopivat. Jokaisessa alustassa on omat hyvät ja huonot puolensa, joiden painon yritys joutuu itse määrittelemään omiin tarpeisiinsa sopivasti.

10 YHTEENVETO

Opinnäytetyön tutkimuskysymyksiin vastaaminen osoittautui hyvin hankalaksi. Ensimmäinen kysymyksistä koski sitä, mitä vaihtoehtoja on olemassa ja toinen soveltuuko yksi alusta useammalle yritykselle. Vaihtoehtoja on olemassa hyvin paljon ja niiden kaikkien käyminen läpi yhdessä opinnäytetyössä on täysin mahdotonta.

Toisen tutkimuskysymyksen vastaus on samanaikaisesti sekä kyllä että ei. Yksi alusta voi soveltua useammalle yritykselle, mutta ei ole varsinaista ”kaavaa”, jonka mukaan jäisi vain yksi vaihtoehto yhdelle yritystyypille. Sama alusta voi soveltua hyvinkin erilaisille yrityksille, mutta voi olla myös, että toinen yritys haluaa ja tarvitsee jotain täysin muuta, vaikka pintapuolin yritykset olisivat hyvinkin samanlaisia.

Opinnäytetyön alussa Kubernetes oli vain käsite, joka oli tullut vastaan Dockerin yhteydessä, mutta siitä, mikä Kubernetes oikeasti on ja mitä se tekee, ei ollut mitään tietoa. Opinnäytetyön teko opetti Kuberneteksen perusteet ja mitä Kubernetes mahdollistaa. Samalla tietoutta kertyi myös hieman enemmän erilaisista sovellusarkkitehtuureista ja niiden hyvistä sekä huonoista puolista.

Tässä opinnäytetyössä käytiin läpi yhdeksän alustaa hyvin pintapuolisesti ja kolme niistä vähän syvemmin. Laajemman kuvan saamiseksi vastaavia testejä voisi suorittaa myös muilla alustoilla. Saatuja tuloksia on mahdollista hyödyntää pohjana hankittaessa sopivaa Kubernetes-alustaa yritykselle, eteenkin jos yrityksellä on ollut mielenkiintoa jotain opinnäytetyössä käsiteltyä alustaa kohtaan.

Tutkimusta voisi viedä eteenpäin testaamalla niitä järjestelmiä ja ohjelmistoja, joihin yritys tulisi alustaa tulevaisuudessa käyttämään. Alustat olisi hyvä myös asentaa yrityksen käyttämään tai sitä vastaavaan ympäristöön. Tämä antaisi parhaan mahdollisen kuvan juuri kyseiselle yritykselle siitä, kuinka alustat toimisivat heidän tarpeisiinsa ja käyttäytyisivät heidän ympäristössään.

LÄHTEET

Anastasia D. (2019). Best Architecture for an MVP: Monolith, SOA, Microservices, or Serverless? Haettu 24.1.2020 osoitteesta https://rubygarage.org/blog/monolith-soa-microservices-serverless

anuupadhyay. (n.d.). Top 5 Cloud Platform Service Providers in 2020. Päivitetty 8.8.2020. Haettu 25.10.2020 osoitteesta https://www.geeksforgeeks.org/top-5-cloud-platform-service-providers-in-2020/#:~:text=Top%205%20Cloud%20Platform%20Service%20Providers%20in%202020,benefit%20of%20cloud%20services.%20...%20More%20items...%20

Belamaric, J., Adkins, S., Robinson, J. E., Giersch, V. & Makogon, D. (2016). OpenStack Cloud Application Development. Haettu 24.1.2020 osoitteesta https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/detail.action?docID=4094475

Citrix. (n.d.). What is public cloud? Haettu 25.10.2020 osoitteesta https://www.citrix.com/fi-fi/glossary/what-is-public-cloud.html

Cloud Native Computing Foundation. (n.d.). Kubernetes Distributions & Platforms. Google Sheets -dokumentti. Haettu 22.1.2020 osoitteesta https://docs.google.com/spreadsheets/d/1LxSqBzjOxfGx3cmtZ4EbB_BGCxT_wlxW_xgHVVa23es/edit#gid=815022624

Crypterium. (2018). Application Architecture explained in a way your Mom will understand. Haettu 24.1.2020 osoitteesta https://hackernoon.com/application-architecture-explained-in-a-way-your-mom-will-understand-d9ce71c12903

derWeihnachtsmann (4.6.2019). Sometimes local cluster in HA setup takes about 10 mts to get to "Active" state. [Avoin ongelma GitHubissa]. Haettu 17.4.2020 osoitteesta https://github.com/rancher/rancher/issues/16213#issuecomment-498647912

Dignan, L. (1.10.2020). Top cloud providers in 2020: AWS, Microsoft Azure, and Google Cloud, hybrid, SaaS players. Haettu 25.10.2020 osoitteesta https://www.zdnet.com/article/the-top-cloud-providers-of-2020-aws-microsoft-azure-google-cloud-hybrid-saas/

Docker. (n.d.). What is a Container? Haettu 26.2.2020 osoitteesta https://www.docker.com/resources/what-container

Eriksson, P. & Koistinen, K. (2014). Monenlainen tapaustutkimus. Haettu 16.10.2020 osoitteesta https://helda.helsinki.fi/handle/10138/153032

GitHub Help. (n.d.). About stars. Haettu 26.2.2020 osoitteesta https://help.github.com/en/enterprise/2.13/user/articles/about-stars

Gnatyk, R. (2018). Microservices vs Monolith: which architecture is the best choice for your business? Haettu 23.1.2020 osoitteesta https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/#

Hou, T. (n.d.). IaaS vs PaaS vs SaaS Enter the Ecommerce Vernacular: What You Need to Know, Examples & More. Haettu 25.10.2020 osoitteesta https://www.bigcommerce.com/blog/saas-vs-paas-vs-iaas/#executive-summary-summing-up-saas-vs-paas-vs-iaas

iahmad-khan (10.8.2019). Re: Failed to get job complete status for job rke-network-plugin-deploy-job [Avoin ongelma GitHubissa]. Haettu 19.4.2020 osoitteesta https://github.com/rancher/rancher/issues/19713#issuecomment-520178290

jmpolvera (16.1.2020). mkdir: cannot create directory '/bitnami/mariadb/data': Permission denied [Suljettu ongelma GitHubissa]. Haettu 19.4.2020 osoitteesta https://github.com/bitnami/bitnami-docker-mongodb/issues/178#issuecomment-575276789

kdopen (9.7.2015). Re: What does ‘upstream’ mean? [Kysymys keskustelufoorumilla]. Haettu 26.2.2020 osoitteesta https://opensource.stackexchange.com/a/995

Kinsta. (n.d.) What Is WordPress? Explained for Beginers. Haettu 13.2.2020 osoitteesta https://kinsta.com/knowledgebase/what-is-wordpress/

Kubernetes. (n.d.a). Example: Deploying WordPress and MySQL with Persistent Volumes. Päivitetty 11.1.2020. Haettu 8.4.2020 osoitteesta https://kubernetes.io/docs/tutorials/stateful-application/mysql-wordpress-persistent-volume/

Kubernetes. (n.d.b). What is Kubernetes? Haettu 13.2.2020 osoitteesta https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

KvantiMOTV. (n.d.). Tutkimusasetelma. Päivitetty 21.12.2009. Haettu 14.9.2020 osoitteesta https://www.fsd.tuni.fi/menetelmaopetus/tutkimus/asetelma.html

Lyons, K. (23.10.2020). Judge again blocks Trump administration push to ban WeChat in the US. Haettu 27.10.2020 osoitteesta https://www.theverge.com/2020/10/23/21531154/judge-denies-trump-administration-ban-wechat-tencent-china

marcosbc (28.10.2019). mkdir: cannot create directory '/bitnami/mariadb/data': Permission denied [Suljettu ongelma GitHubissa]. Haettu 8.4.2020 osoitteesta https://github.com/bitnami/bitnami-docker-mongodb/issues/178#issuecomment-546837135

Matthias, K. & Kane, S. P. (2015). Docker Up & Running: Shipping Reliable Containers in Production. Haettu 24.2.2020 osoitteesta https://books.google.fi/books?id=IDvcCQAAQBAJ&lpg=PP1&dq=what%20is%20docker&lr&hl=fi&pg=PP1#v=onepage&q=what%20is%20docker&f=false

Mulesoft. (n.d.). Microservices vs Monolithic Architecture. Haettu 24.1.2020 osoitteesta https://www.mulesoft.com/resources/api/microservices-vs-monolithic

Mutai, J. (2020a). How To Create Admin User to Access Kubernetes Dashboard. Haettu 26.4.2020 osoitteesta https://computingforgeeks.com/create-admin-user-to-access-kubernetes-dashboard/

Mutai, J. (2020b). How To Install Kubernetes Dashboard with NodePort. Haettu 26.4.2020 osoitteesta https://computingforgeeks.com/how-to-install-kubernetes-dashboard-with-nodeport/

Nortal. (2017). Are Monolithic Software Applications Doomed for Extinction? Haettu 24.1.2020 osoitteesta https://nortal.com/fi/blog/are-monolithic-software-applications-doomed-for-extinction/

OneClick. (19.11.2019). Kubernetes – everything you need to know. Päivitetty 2.6.2020. Haettu 25.10.2020 osoitteesta https://oneclick-cloud.com/en/blog/trends-en/kubernetes/

Opensource.com. (n.d.). What is virtualization? Haettu 13.2.2020 osoitteesta https://opensource.com/resources/virtualization

Ranger, S. (13.12.2018). What is cloud computing? Everything you need to know about the cloud explained. Haettu 25.10.2020 osoitteesta https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/

Red Hat 1. (n.d.a). Understanding virtualization. Haettu 13.2.2020 osoitteesta https://www.redhat.com/en/topics/virtualization?intcmp=7016000000127cYAAQ

RedHat. (n.d.b). What is Kubernetes? Haettu 27.2.2020 osoitteesta https://www.redhat.com/en/topics/containers/what-is-kubernetes

RedHat. (n.d.c). What is virtualization? Haettu 27.9.2020 osoitteesta https://www.redhat.com/en/topics/virtualization/what-is-virtualization

Rocket.Chat. (n.d.a). Installing Rocket.Chat Chart on Kubernetes using Helm. Haettu 25.4.2020 osoitteesta https://rocket.chat/docs/installation/helm-chart/

Rocket.Chat. (n.d.b). Rocket.Chat. Haettu 26.2.2020 osoitteesta https://rocket.chat/

Rouse, M. (n.d.a). [A]pplication architecture. Päivitetty 08/2019. Haettu 24.1.2020 osoitteesta https://searchapparchitecture.techtarget.com/definition/application-architecture

Rouse, M. (n.d.b). [V]irtualization. Päivitetty 10/2019. Haettu 17.9.2020 osoitteesta https://searchservervirtualization.techtarget.com/definition/virtualization

Rouse, M. (n.d.c). What is public cloud? Everything you need to know. Päivitetty 08/2020. Haettu 25.10.2020 osoitteesta https://searchcloudcomputing.techtarget.com/definition/public-cloud

Ruvic, D. (20.10.2020). Sweden bans Huawei, ZTE from upcoming 5G networks. Haettu 27.10.2020 osoitteesta https://www.cnbc.com/2020/10/20/sweden-bans-huawei-zte-gear-from-5g-spectrum-auction.html#close

Rydman, A. (19.5.2020). Jääkö EU Kiinan ja USA:n jalkoihin? ”Syytä olla huolissaan”. Haettu 27.10.2020 osoitteesta https://www.verkkouutiset.fi/jaako-eu-kiinan-ja-usan-jalkoihin-syyta-olla-huolissaan/

Williamson, A. (6.5.2020). IaaS, PaaS, SaaS. Haettu 28.10.2020 osoitteesta https://www.inap.com/blog/iaas-paas-saas-differences/

Liite 1

Yhdeksän Kubernetes-alustan vertailutaulukko

	Minikube	Rancher	K3s	Kops	Kubespray	Openshift	Kind	Kubesphere	MicroK8s
Hinta	Ilmainen	Ilmainen	Ilmainen	Ilmainen	Ilmainen	Maksullinen	Ilmainen	Ilmainen	Ilmainen
Lähdekoodi	Avoin	Avoin	Avoin	Avoin	Avoin	Avoin	Avoin	Avoin	Avoin
Ominaisuudet									
Multi node	N	Y	Y	Y	Y	Y	Y	Y	Beta
Multi master	N	Y	Y	Y	Y	Y	Y	Y	N
Dashboard	Y	Y	N		Y	Y		Y	Y
DNS	Y	Y	Y	Y	Y	Y		Y	Y
Automaattiset päivitykset						Y			Y
Käyttäjätuki	Kommuuni	Y	Y	Kommuuni	Kommuuni	Y	Kommuuni	Kommuuni	Y
Alustatyyppi	installer	distro	distro	installer	installer	distro	installer	distro	distro
Status	Stable	Stable	Stable	Stable	Stable	Stable	Work in progress	Stable	Stable
Julkaisija	Kubernetes/minicube	Rancher	Rancher	Kubernetes/Kubernetes Kops Project	Kubernetes SIGs	RedHat	Kubernetes SIGs	Kubesphere	Canonical
Järjestelmävaatimukset									
RAM (min)	5,0 GB	4,0 GB	0,5 GB		1,5 GB	16,0 GB	6,0 GB	4,0 GB	4,0 GB
Tallennustila (min)			0,2 GB			42,0 GB		100,0 GB	20,0 GB
Käyttöjärjestelmä	Linux, MacOS, Windows	Ubuntu 16.04, Red Hat Enterprise Linux 7.5, RancherOS 1.4	Linux	Linux, MacOS	Container Linux by CoreOS, Ubuntu 16.04/18.04, CentOS/RHEL 7, Fedora 28, Fedora/CentOS Atomic, Oracle Linux 7, openSUSE Leap 42.3/Tumbleweed, Debian Buster/Jessie/Stretch/W heezy	Linux, MacOS, Windows	Linux, MacOS, Windows		Ubuntu 16.04/18.04 (Linux with snapd support)
Pilvialustat	-	AWS, Google Cloud, Azure		AWS (official), Google Cloud (beta)	AWS, Google Cloud, Azure	AWS, Google Cloud, Azure	-	Google Cloud	AWS, Google Cloud, Azure

Y = Kyllä
N = Ei

Liite 2

Kolmen Kubernetes-alustan vertailutaulukko lisätyillä tiedoilla

Y	= Kyllä	
N	= Ei	

	Minikube	Rancher	Kubesphere
Hinta	Ilmainen	Ilmainen	Ilmainen
Lähdekoodi	Avoin	Avoin	Avoin
Ominaisuudet			
Multi node	N	Y	Y
Multi master	N	Y	Y
Dashboard	Y	Y	Y
DNS	Y	Y	Y
Automaattiset päivitykset			
Käyttäjätuki	Kommuuni	Y	Kommuuni
Alustatyyppi	installer	distro	distro
Status	Stable	Stable	Stable
Julkaisija	Kubernetes/minicube	Rancher	Kubesphere
Järjestelmävaatimukset			
RAM all-in-one (min)	5,0 GB	4,0 GB	4,0 GB
RAM Multi node (min)	-	4,0 GB	4,0 GB
RAM Täysi asennus (min)	-	8,0 GB	16,0 GB
Tallennustila (min)			100,0 GB
CPU-ydinten lkm all-in-one		1	2
CPU-ydinten lkm multi node		1	2
CPU-ydinten lkm Täysi		2	8