Tampereen yliopisto

Veikko Nieminen

KONTTIEN HALLINTAJÄRJESTELMÄT HAJAUTETUSSA YMPÄRISTÖSSÄ

Informaatioteknologian ja viestinnän tiedekunta
Kandidaatintyö
Marraskuu 2021

TIIVISTELMÄ

Veikko Nieminen: Konttien hallintajärjestelmät hajautetussa ympäristössä
Kandidaatintyö
Tampereen yliopisto
Tietotekniikka
Marraskuu 2021

Virtualisointi on yleistynyt palvelinympäristöissä viimeisten kahden vuosikymmenen aikana. Aikaisemmin virtuaalikoneet olivat suuressa suosiossa, mutta viime aikoina konttiteknologioiden käyttö on yleistynyt. Tässä työssä esitellään kolme eri konttienhallintajärjestelmää sekä vertaillaan niiden käyttöä. Käsiteltävät järjestelmät ovat Docker compose, Docker swarm ja Kubernetes. Työn tavoitteena on esitellä yleisimmät virtualisointitekniikat palvelinkontekstissa sekä tutustuttaa lukija eri hallintajärjestelmiin ja niiden ominaisuuksiin.

Työ jakaantuu kahteen osaan. Johdannon jälkeen tarkastellaan virtualisoinnin historiaa palvelinkontekstissa, konttiteknologioita sekä tarkastellaan konttiteknologian edustajana Dockeria. Tämän jälkeen esitellään kolme eri konttienhallintajärjestelmää. Työn lopussa esitellään kunkin järjestelmän käyttöönottoa ja käyttöä.

Lähteiden perusteella havaittiin, että Kubernetes on monipuolisin vertailluista järjestelmistä. Sillä voidaan tehdä hyvinkin monimutkaisia konfiguraatioita. Toisaalta Kuberneteksen käyttäminen vaatii huomattavasti enemmän perehdytystä kuin muut vertailtavana olevat kohteet. Se on yleisesti käytetty monista konteista koostuvien ohjelmistojen ajoympäristö. Docker swarm on tarkoitettu nimenomaan vain useiden Docker ohjelmistojen yhdistämiseen. Sen avulla voidaan käynnistää kontteja useissa laitteissa, mutta se ei tue useista konteista koostuvia ohjelmistoja. Docker compose sen sijaan on helppokäyttöinen ja nopea oppia. Sen avulla voidaan hyvin yksinkertaisesti käynnistää monista konteista koostuvia ohjelmistoja, mutta se ei tue usealla eri laitteella ajamista. Hallintajärjestelmän valinta riippuu hyvin paljon käyttötarpeesta ja tottumuksesta. Eri järjestelmät ovat selvästi tarkoitettu eri käyttökohteisiin.

Avainsanat: konttiteknologiat, Docker, Kubernetes, virtuaalikone

SISÄLLYSLUETTELO

1 JOHDANTO

Perinteisesti palvelimilla on ajettu ohjelmistoja ”one server one application” -ajatusmallin mukaisesti, eli yksi ohjelma yhdellä laitteella. Näin ajettavat ohjelmat ja ohjelmistot ovat toisistaan eristettyjä eivätkä ne häiritse toistensa suoritusta. Useimmiten ohjelmisto ei kuitenkaan käytä palvelimen resursseista kuin osan. Näin ollen koneen arvokasta laskentatehoa menee hukkaan.

Virtualisoinnin avulla palvelimen resurssit voidaan jakaa useamman, erillisen ohjelmiston käyttöön. Yleinen virtualisointitapa on ollut virtuaalikoneiden käyttö, mutta viime vuosikymmenen aikana konttiteknologiat ovat kasvattaneet suosiotaan. Konttiteknologiassa on myös monia muita hyviä lisäominaisuuksia.

Yksittäisten konttien hallinta on yksinkertaista, mutta varsinkin monimutkaisissa kymmenistä konteista koostuvissa järjestelmissä hallinta muuttuu hyvin hankalaksi. Tätä ongelmaa varten on kehitetty monia eri konttienhallintajärjestelmiä, kuten Kubernetes.

Tässä työssä pyritään vastaamaan kysymykseen: Mitä ovat konttienhallintajärjestelmät ja millaista niiden käyttöönotto on hajautetussa ympäristössä. Työssä käydään läpi virtuaalikoneiden perusteet, kerrotaan mitä kontit ovat ja miten ne eroavat virtuaalikoneista. Konttiteknologian esimerkkiohjelmana käytetään Docker-ohjelmistoa. Lisäksi käydään läpi kolme eri konttienhallintajärjestelmää sekä käydään läpi kunkin ohjelmiston käyttöönoton ja käytön helppous.

Luvussa 2 esitellään virtualisointi, virtuaalikoneet, konttiteknologia sekä Docker. Luvussa 3 esitellään kolme eri konttienhallintajärjestelmää. Luvussa 4 esitellään järjestelmien käyttöä. Luku 5 on yhteenveto.

2 VIRTUALISOINTI PALVELINYMPÄRISTÖSSÄ

Virtualisointi tarkoittaa tietotekniikassa jonkin fyysisen osan abstrahointia loogiseksi osaksi [1]. Tällöin fyysisen laitteen merkitys poistuu, joten laite voidaan korvata toisella ilman yhteensopivuusongelmia. Virtualisointi mahdollistaa fyysisten resurssien optimaalisen käytön [2].

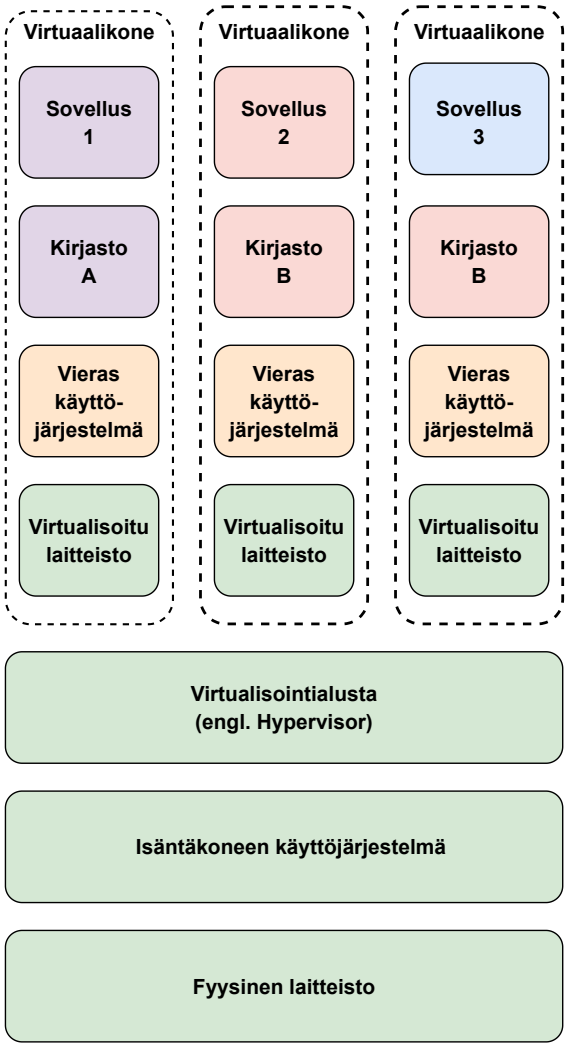
Virtualisointia käytetään monissa eri käyttökohteissa. Palvelinkontekstissa yleisimpiä näistä ovat järjestelmän, verkon, tallennustilan, sovelluksien ja työpöytien virtualisointi [3]. Tässä työssä keskitytään järjestelmän ja sovelluksien virtualisointiin.

2.1 Virtuaalikone

Popekin ja Goldbergin vuoden 1974 määritelmän mukaan virtuaalikone on ohjelmisto, joka voi virtualisoida kaikkia laitteiston resursseja, kuten prosessoria, muistia ja verkkokorttia [4]. Tämä määritelmä kuvaa myös nykyaikaisia virtuaalikoneita. Ensimmäiset virtuaalikoneet luotiin käyttöjärjestelmien kehittämistä varten [1]. Kehitettävän ohjelmiston testaaminen virtuaalikoneessa oli huomattavasti helpompaa ja nopeampaa kuin asentaminen erilliselle tietokoneelle. Ensimmäinen kaupallisesti saatavilla oleva virtualisointiohjelmisto x86 suoritinarkkitehtuurille julkaistiin vuonna 2001 [1].

Virtualisointialusta luo jokaiselle virtuaalikoneelle omat, vieraan käyttöjärjestelmän käyttämät virtuaaliset laitteet. Virtuaaliset laitteet käyttävät fyysisen laitteen ominaisuuksia toimiakseen. Ajettavan ohjelmiston näkökulmasta virtualisoitu ympäristö ei poikkea oikeasta ympäristöstä mitenkään. Yhdellä fyysisellä palvelinkoneella voidaan ajaa useaa virtuaalikonetta samaan aikaan, jolloin samat resurssit saadaan laajempaan käyttöön. Eri virtuaalikoneet ovat toisistaan täysin eristettyjä.

Kuvassa 2.1 on esitetty palvelin, jossa on käytössä kolme virtuaalikonetta. Jokaisessa koneessa pyörii oma ohjelma oman vieraan käyttöjärjestelmän päällä. Virtuaalikoneita ohjaa virtualisointialusta (engl. hypervisor), joka voidaan ajaa joko käyttöjärjestelmän päällä tai suoraan laitteistolla.



Kuva 2.1. *Kolme virtuaalikonetta yhdellä laitteella lähteiden [3, 5] mukaisesti*

Yksittäisen virtuaalikoneen resursseja voidaan muokata jopa käynnissä ollessaan, jolloin skaalautuvuus helpottuu. Yksityiskäytössä eristettyä virtuaalikonetta voidaan hyödyntää tietoturvatarkoituksessa. Virtuaalikoneessa voidaan ajaa mahdollisen tietoturvariskin aiheuttamia ohjelmia ilman muutoksia oikeaan käyttöjärjestelmään. Useimmissa virtualisointiohjelmistoissa voidaan ajaa eri arkkitehtuurin käyttöjärjestelmiä. Tämä mahdollistaa eri järjestelmien ajamisen samassa tietokoneessa, jolloin ei tarvita kolmea eri tietokonetta kolmelle eri järjestelmälle.

2000-luvulta alkaen palvelimilla suoritettavia ohjelmia on ajettu virtuaalikoneissa. Ohjelmien siirtäminen virtuaalikoneisiin vähensi laitteiston määrän tarvetta neljännekseen; joissain tapauksissa jopa kahdeksan serverin ohjelmistot voitiin yhdistää yhteen laitteeseen. Laitteiston määrän vähentäminen vaikutti suoraan palveluiden ylläpitokustannuksiin muun muassa uusien laitteiden hankinnan, sähkön ja jäähdytyksen kustannuksien laskemisena. [1]

Virtuaalikoneiden hallintajärjestelmissä on myös useimmiten mahdollista tehdä virtuaalikoneiden tilannekuvia (engl. snapshot). Tilannekuva sisältää koneen tilan kyseisellä het-

kellä. Virtuaalikoneen voi palauttaa tiettyyn tilannekuvaan, jolloin järjestelmän suoritusta voidaan jatkaa kyseisestä tilasta tai hetkestä. Virtuaalikone on oma kokonaisuutensa, joka voidaan useimmiten siirtää helposti toiselle fyysiselle laitteelle. Näitä kokonaisuuksia voidaan myös replikoida ja ajaa rinnakkain tarpeen vaatiessa. Virtuaalikoneet eivät sovi raskaaseen käyttöön, sillä virtualisointi ei ole yhtä tehokasta kuin suora laitteistolla ajaminen. Toisaalta taas virtuaalikoneita on mahdollista pakata liikaa yhdelle koneelle.

Vuonna 2008 finanssialan yritykselle Accountants Inc. tuli ajankohtaiseksi palvelinlaitteiston uusiminen. Yrityksellä oli palvelimia käytössä sisäisten tietoteknisten operaatioiden takia ”one server one application” -metodin mukaisesti. Virtualisoinnin avulla fyysisten serverien määrä saatiin laskettua viidestäkymmenestä kymmeneen. Palvelinkustannuksien lisäksi säästöä syntyi vähentyneistä sähkön ja jäähdytyksen kulutuksista. Virtualisoinnin avulla myös yhtiön tuotantoympäristön vikasietoisuus parani. [6]

2.2 Kontti

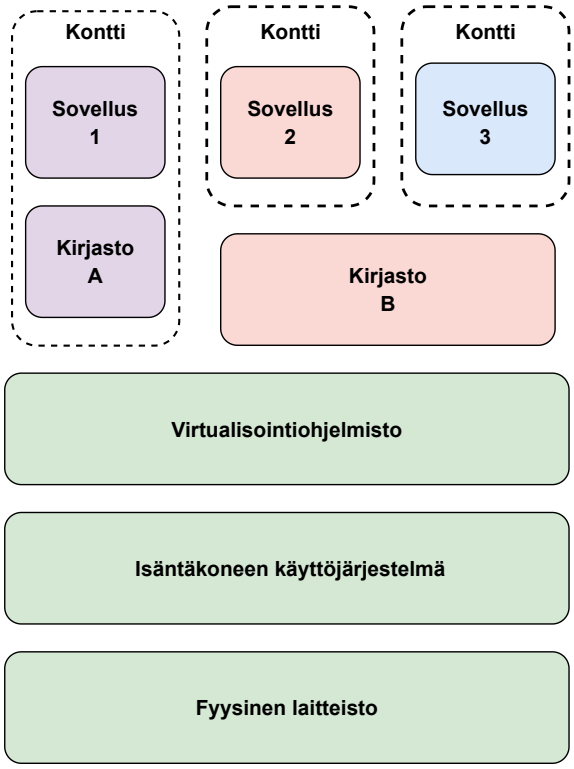
Konttiteknologia (engl. containerization) on viimeaikoina yleistynyt virtualisointitekniikka, jolla voidaan ajaa ohjelmistoja. Ohjelmistot pakataan kontteihin, jotka sisältävät ohjelman lisäksi kaikki sen tarvitsemat muut ohjelmat, kirjastot ja asetustiedostot. Kontittaminen on universaali tapa pakata ohjelmisto yhdeksi kokonaisuudeksi, jolloin ajoympäristö tai käytettävissä oleva laitteisto ei vaikuta ohjelman ajamiseen. Näin kontteja voidaan siirtää eri laitteista toisiin ilman riippuvuusongelmia. [7]

Virtuaalikoneeseen vertailtuna kontti ei tarvitse omaa käyttöjärjestelmää eikä virtualisoituja laitteita, joka tekee konteista huomattavasti virtuaalikoneita kevyempiä ajettavia. Konttiteknologia ei tosin mahdollista eri arkkitehtuurille luotujen ohjelmistojen ajamista, sillä konteissa ei ole omaa käyttöjärjestelmää.

Nykyaikaiset kontit koostuvat kuvista (engl. image), joita käytetään kontin lähtökohtana. Kuva on muuttumaton olio, jossa määritellään kontin sisältö. Yksinkertaisissa palveluissa kuva on hyvin kevyt, jolloin konttien käynnistäminen, luonti ja hallinta ylipäänsä on hyvin nopeaa ja ketterää.

Konttiteknologioiden avulla voidaan ajoympäristö lisätä versionhallintaan mukaan. Pelkän koodin versionhalllinta ei takaa, että ohjelma toimii ajoaikana samalla tavalla, sillä se voi olla riippuvainen ajoympäristöstä sekä ulkoisista riippuvuuksista. Kun ajoympäristö on määritelty koodissa, voidaan palata takaisin helposti vanhoihin versioihin versionhallinnan avulla. Konttien käyttämisen avulla ohjelmistojen ylläpitäjät voivat keskittyä yksinkertaisesti konttien ylläpitämiseen ja käynnistämiseen eikä ympäristöjen asentamiseen ja asetuksien määrittelemiseen. [8]

Konttiteknologia ei sovi kaikenlaisille ohjelmistoille. Isot ja monipuoliset yhden prosessin ohjelmistot eivät ole otollisia kontittamiselle. Teknologia on ideaali palvelukeskeistä arkkitehtuuria noudattaville ohjelmistoille. Palvelukeskeiset ohjelmistot voidaan jakaa palveluittain eri kontteihin, jolloin jokaista eri palvelua voidaan skaalata tarpeen mukaan. [8]



Kuva 2.2. *Kontteja yhdellä laitteella lähteiden [3, 5] mukaisesti*

Kuvassa 2.2 on esitetty tyypillinen järjestelmä, jossa on kontteja ajossa. Huomioitavina asioina kuvaan 2.1 verrattuna on vieraiden käyttöjärjestelmien sekä virtualisoitujen laitteiden puuttuminen sekä kirjaston B jakaminen kahden kontin kesken.

Open containers initiative on vuonna 2015 perustettu projekti, jonka tavoitteena on kehittää avoimet määritelmät ohjelmistojen ajossa käytettäville konteille sekä niiden työkaluille. Projekti on tuottanut kolme eri standardia: kontin ajoympäristön määritelmän, kontin kuvan määritelmän sekä kuvien jakamisen määritelmän. [9]

2.3 Konttiteknologiat

Konttiteknologiat ovat työkaluja konttien luomiseen, ajamiseen ja hallinnointiin. Konttiteknologioita on useita, kuten LXC, rkt ja podman. Jotkin ohjelmat keskittyvät konttien ajoon ja toisen niiden hallinnointiin. Tässä työssä keskitytään vain yleisimmin käytössä olevaan Docker ohjelmistoon.

Docker on ohjelmisto, jolla voidaan luoda, siirtää ja ajaa kontteja [7, 10]. Sen ensimmäinen avoimen lähdekoodin versio julkaistiin vuonna 2013. Alunperin ohjelmien ajaminen toteutettiin LXC Enginen avulla. Myöhemmässä versiossa tämä kuitenkin korvattiin omalla ajo-ohjelmalla. Docker ohjelma koostuu sen ytimestä (Docker Daemon) sekä sitä hallitsevista työkaluista. Hallinta tapahtuu client-server -mallin mukaisesti rajapintaa hyödyntäen. Yleisin keino hallita Dockeria on sen komentorivityökalu. Toiseen laitteeseen asennettua Dockeria voidaan myös hallita internetin ylitse. Dockerin komentoriviohjelma

tarjoaa tähän suoraan asetuksia, jolloin Dockerin komentoja voidaan ajaa normaalisti, mutta toteutus tapahtuu toisella laitteella. [7]

Dockerin kuvat muodostetaan Dockerfile nimisen tiedoston pohjalta. Tiedostossa määritellään eri komentojen avulla kuville kerroksia siten, että jokainen komento on oma kerroksensa. Muutokset kerroksen sisältöön, kuten lähdekoodimuutos tai kirjaston version muuttaminen vaikuttaa vain kyseiseen tasoon. Kuvaa päivittäessä vain tämä taso ja sen yläpuolella olevat tasot pitää päivittää. Alapuolella olevat tasot pysyvät muuttumattomina ja ne voidaan ladata välimuistista. Tämä toiminnallisuus nopeuttaa kuvien luontia huomattavasti. Moderneissa, usean vaiheen kuvissa voidaan määrittää myös erillinen koontiympäristö erikseen lopullisesta kuvasta. Näin tuotettava lopullinen kuva voi olla huomattavasti kevyempi, sillä siihen voidaan sisällyttää vain ohjelman ajettava versio sekä sen tarjoamiseen tarvittavat ohjelmat. [10]

Kuvien hallintaan ja jakamiseen voidaan käyttää rekistereitä, joista yleisin ja oletuksena käytössä oleva rekisteri on Docker Hub. Rekisterissä on monien eri jakajien yleisesti käytettyjä ohjelmistoja valmiina kuvina. Rekistereitä voidaan käyttää kuvien jakamiseen eri kohderyhmille. Docker Hub tarjoaa maksullisia lisenssejä yritysten käyttöön, jolloin sen kautta voidaan jakaa kuvia yksityisesti oman tiimin käyttöön. Docker tukee myös omien, itse ylläpidettyjen rekisterien käyttöä.

Ohjelmistot, joiden ajaminen uudella laitteella vaatii tarkkaa ajoympäristön määrittelyä muuttuvat helposti siirrettäviksi Dockerin avulla. Kun kuva on kerran määritelty, se voidaan yhden komennon avulla ajaa millä tahansa saman arkkitehtuurin laitteella, jossa on Docker asennettuna [11]. Näiden metodien käyttö voi merkittävästi vähentää viivettä ohjelman luomisen ja tuotannossa ajamisen välillä [10].

3 KONTTIEN HALLINTAJÄRJESTELMÄT

Yksittäisten konttien hallinta on helppoa komentorivillä, mutta hallinta muuttuu huomattavasti monimutkaisemmaksi, jos konttien määrä kasvaa. Vaikka konttiin on pakattu kaikki sen tarvitsema, pitää kontille silti määritellä sen ajamiseen tarvittavat ominaisuudet, kuten esimerkiksi avattavat portit, liitettävät tallennustilat ja ympäristömuuttujat.

Ongelman ratkaisuksi on kehitetty monia eri keinoja hallita kontteja. Hallintajärjestelmät helpottavat konttien käynnistämistä määrittäen asetukset valmiiksi. Jotkin järjestelmät ovat kokonaisia systeemeitä, joiden avulla voidaan luoda automaattisesti skaalautuvia ja vikasietoisia kokonaisuuksia ylläpidettynä monilla eri laitteilla. Kirjan ”Kubernetes: Up and Running” mukaan konttien ja konttienhallintajärjestelmien käytön syyt voidaan perustella neljällä hyödyllä: nopeus, skaalattavuus, infrastruktuurin abstrahoiminen ja tehokkuus. [12]

3.1 Docker Compose

Docker Compose on komentorivityökalu, jonka avulla voidaan määritellä ja hallita useammista konteista koostuvia kokonaisuuksia. Se käyttää asetustiedostoa, jossa määritellään palveluita (engl. service), jotka sisältävät tiedon kontin kuvasta sekä kaikki sen ajamiseen vaadittavat asetukset. Tiedostossa voidaan myös määritellä palveluille verkkoja tai tallennustiloja. Työkalun avulla konttien ajoasetukset voidaan tallettaa asetustiedostoon, jolloin ne saadaan mukaan versionhallintaan, helpottaen kokonaisuuden siirtämistä ja replikointia eri laitteille. Se on alun perin suunniteltu kehitysympäristöjen nopeaan käynnistämiseen. [7]

Docker Compose ei itsessään ole oma kokonaisuutensa, vaan se on työkalu, joka hallitsee kontteja käyttäen Docker -ohjelman rajapintaa. Näin ollen se helpottaa Dockerin käyttöä useiden konttien hallinnassa. Suurin etu Docker Composen käyttämisessä on sen helppokäyttöisyys. Asetustiedoston luomisen jälkeen yhdellä komennolla voidaan hallita monista palveluista koostuvia kokonaisuuksia. Esimerkkinä ohjelma, jossa on webpalvelin sekä sen käyttämä tietokanta. Docker Compose tarjoaa myös monimutkaisempia ominaisuuksia, kuten palveluiden skaalaamisen. Siinä ei kuitenkaan ole sisäänrakennettua kuormituksen tasaamista. Yleinen käyttökohde Docker Composelle on jakaa palvelun asetustiedosto, jolloin toinen käyttäjä voi yhden komennon avulla käynnistää täydellisen kopion ajoympäristöstä. Docker Composea voidaan käyttää myös tuotannon ajoissa, mutta tosin vain yhdellä laitteella. Se ei sisällä usean laitteen tukea. [7, 13]

3.2 Docker Swarm Mode

Docker Swarm Mode on Docker ohjelman mukana tuleva työkalu, jonka avulla voidaan yhdistää useampi Docker ohjelma yhdeksi kokonaisuudeksi. Swarm Mode lisättiin Dockeriin versiossa 1.12, joka julkaistiin elokuussa 2016. Sen toteutus pohjautuu ennen erillisenä olleeseen Swarmkit työkaluun. Huomioitavaa on, että ennen Dockerin version 1.12 julkaisua Dockeriin oli mahdollista asentaa saman kaltainen työkalu nimeltä Docker Swarm. Tämä ohjelma on kuitenkin vanhentunut ja käytöstä poistunut. Swarm Modeen verrattuna ohjelma on varsinkin toteutukseltaan hyvin erilainen. [14]

Docker Swarm Mode on osa Dockeria, joten erillistä asentamista ei tarvita. Swarm Moden parhaita puolia on juurikin työkalun helppokäyttöisyys. Uuden parven (engl. Swarm) saa luotua yhdellä komennolla ja parveen liittyminen toisella laitteella tapahtuu myös yhdellä komennolla. Parvi koostuu kahdesta tai useammasta laitteesta, joista jokainen on rooliltaan joko manageri (engl. manager) tai työntekijä (engl. worker). Swarm Moden ajettavat ohjelmat määritellään palveluilla (engl. service). Palvelulle määritellään ajamiseen tarvittavat parametrit, kuten avoimet portit, yhdistettävät tallennustilat ja kopioiden määrä. Parven aktiivinen manageri päättää palvelun konttien jakamisesta eri laitteille. [15]

Swarm Mode luo kaikkien palvelun konttien päälle yhteisen verkon, jolloin kaikki parveen tulevat yhteydet reititetään määriteltyjen sääntöjen mukaisesti. Oletuksellisesti tämä verkko jakaa tulevat pyynnöt tasaisesti eri parven konttien kesken. Tätä kutsutaan kuormituksen tasaamiseksi (engl. load balancing). [15]

3.3 Kubernetes

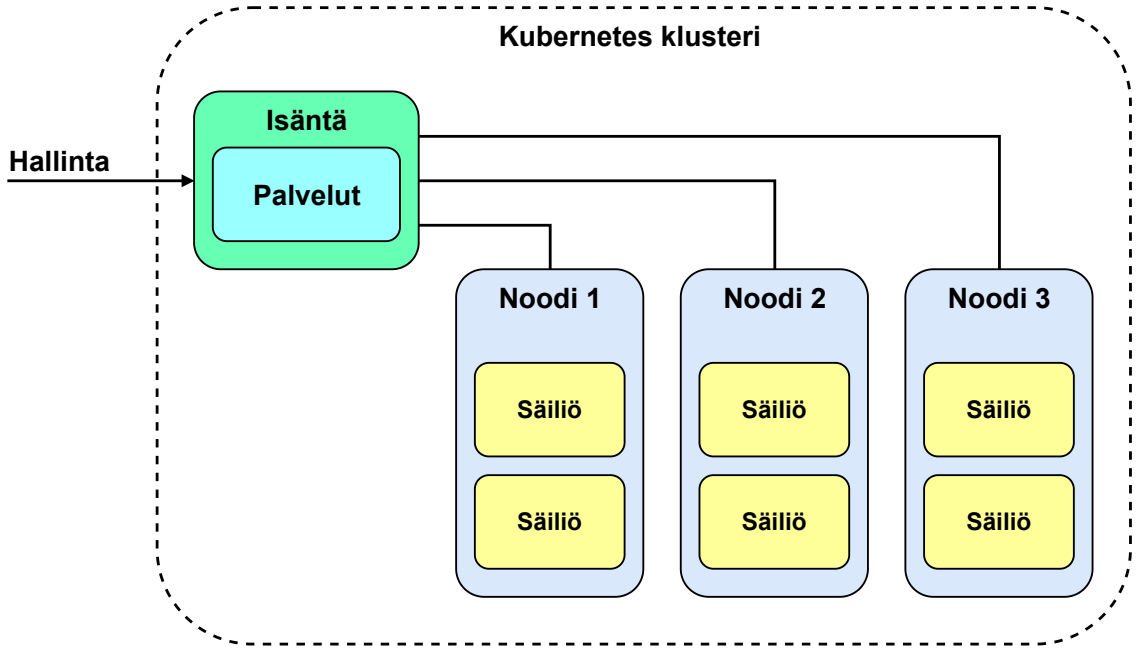
Kubernetes on avoimen lähdekoodin konttien orkestrointijärjestelmä [16]. Konttien yleistymisen myötä myös niiden määrät kasvoivat hurjasti. Googlen sisällä aloitettiin kehittää systeemiä isojen konttimäärien automaattiseen hallintaan. Kubernetes perustuu Googlen sisäiseen käyttöön luotuun Borg ohjelmaan, jonka juuret juontavat 2000-luvun alkuun. Vuonna 2014 julkaistussa Googlen tiedotteessa mainitaan kaikkien Googlen palveluiden ajettavan Linux konteissa. Dockerin julkaisun jälkeen 2013 Google otti Borgin parhaat ominaisuudet ja julkaisi sen avoimella lähdekoodilla. Version 1.0 julkaisun yhteydessä Google muodosti Cloud Native Computing Foundationin (CNCF) yhteistyössä Linux Foundationin ja monen muun yrityksen kanssa. CNCF on nykyisin vastuussa Kuberneteksen ylläpidosta. [17]

Kubernetes on yleisin tuotantokäytössä oleva konttien hallintajärjestelmä. Sen ominaisuuksiin kuuluu

- Palveluiden automaattinen löytyminen ja automaattinen kuormituksen tasaus
- Tallennustilojen hallinnointi
- Hallittu rullaava päivitys. Palvelut voidaan päivittää yksitellen siten, että palvelu on koko ajan saatavilla.
- Automaattinen konttien sijoittaminen eri noodeille
- Sammuneiden palveluiden korvaaminen uusilla
- Tuotannossa käytettävien salaisuuksien ja asetusten hallinnointi

Kubernetes koostuu useista eri komponenteista, joten ohjelman toimintaa voidaan mukauttaa helposti vaihtamalla komponentteja. Kun ohjelmisto on muokattu toimimaan Kuberneteksessä, sen skaalaaminen on hyvin yksinkertaista ja se voidaan jopa automatisoida. [16, 18]

Kuberneteksen klusteri koostuu isännästä (enlg. master) ja yhdestä tai useammasta noodista (engl. node). Noodit sisältävät yhden tai useampia säiliöitä (engl. pod). Säiliö on kokoelma kontteja ja tallennustiloja. Hyvä nyrkkisääntö säiliöiden koon määrittämiseen on että säiliön sisällöllä on jokin yhteinen resurssi. Esimerkiksi yhden verkkopalvelun webserveri, sen käyttämä tietokanta ja tietokannan käyttämä tallennustila. Klustereita voi olla useampi ja yhdessä klusterissa voi olla myös useampi isäntä. [18]



Kuva 3.1. *Yksinkertainen Kubernetes klusteri lähteen [18] mukaisesti*

Kuvassa 3.1 on esitetty esimerkiksi yksinkertainen Kubernetes klusteri, joka sisältää kolme eri noodia. Esimerkissä eri noodit on sijoitettu eri laitteille. Noodien sisällä on säiliöitä, jotka voivat olla kopioita toisistaan tai täysin erillisiä. Klusterin hallinta tapahtuu kokonaan isännän tarjoaman rajapinnan kautta. [18]

4 HALLINTAJÄRJESTELMIEN KÄYTTÖ

Tässä kappaleessa esitellään edellisessä kappaleessa käsiteltyjen hallintajärjestelmien käyttöä ja niiden käyttöönottoa. Tarkoituksena on vertailla järjestelmien helppoutta ja käytön oppimisen nopeutta.

Vertailua huomattavasti hankaloittaa järjestelmien erilaisuus. Jokainen vertailtava järjestelmä on kehitetty eri tarkoitukseen, joten niitä on mahdotonta vertailla täysin.

4.1 Docker Compose

Docker Composen käyttöönotto on helppoa. Se pitää kuitenkin asentaa erillisenä Dockerista, mikä on yksinkertainen prosessi. Asennuksen jälkeen ohjelma on täysin käyttövalmis. Docker Composen käytto on yksinkertaista. Määrittelytiedostojen tekeminen vaatii oman opiskelunsa, mutta toisaalta valmiita määrittelytiedostoja on saatavilla laajasti. Itse ohjelman käyttö koostuu pääosin kahdesta eri komennosta. [7]

Ohjelma on parhaimmillaan joko testiympäristöjen pystyttämisessä tai yhden laitteen tuotantoympäristössä. Docker Compose ei tarjoa Dockeriin verrattuna uusia ominaisuuksia, vaan yksinkertaistaa Dockerin käyttöä tehden siitä käyttäjäystävällisemmän.

4.2 Docker Swarm Mode

Docker Swarmiin tutustuessa havaittiin sen käyttämisen olevan joukon helpointa. Kun Docker oli asennettu, Swarmin käyttöönotto onnistui yhdellä komennolla. Uuden parven luominen oli helppo toteuttaa. Ohjelma antoi parvea luodessa koodin, joka toimi parveen liittymisen salasanana. Toisilla laitteilla voidaan yhtä yksinkertaisesti liittyä parveen.

Palveluiden luominen ja ajaminen on nopeaa ja kätevää. Palvelut määritellään kuvan ja sen ajamiseen vaikuttavien parametrien avulla, kuten kopioiden määrä, resurssien rajoitteet ja avoimet portit. Docker swarm luo oletuksellisesti julkaistujen ohjelmien päälle jokaisen laitteen kattavan verkon joka tarjoaa myös automaattisen kuormituksen tasaamiseksi. Käyttötarkoitukseltaan Docker Swarm on huomattavasti lähempänä Kubernetestä kuin Docker Compose, mutta ominaisuuksiltaan se jää huomattavasti Kuberneteksestä jälkeen. [15]

4.3 Kubernetes

Kuberneteksen asentaminen on esitellyistä järjestelmistä selvästi hankalin. Oman paikallisen testiympäristön pystyttämiseen on tehty avustavia työkaluja, kuten Minikube. Kuberneteksen testiympäristön voi myös pystyttää Dockerin päälle. Tuotantovalmiin ympäristön asentaminen on huomattavasti hankalampaa. [17]

Kuberneteksen käyttäminen on joukon hankalinta. Pelkästään yhden palvelun pystyttämistä varten tulee sisäistää iso joukko termejä ja käytäntöjä. Toisaalta nämä opittuaan skaalattavan systeemin luominen on melkein yhtä helppoa. Kubernetes sopii parhaiten isojen, mahdollisesti skaalausta tarvitsevien ohjelmistojen ajamiseen. Se sopii myös palvelukeskeistä arkkitehtuuria noudattavien ohjelmistojen ajamiseen. [17]

5 YHTEENVETO

Työn aikana havaittiin Kuberneteksen olevan selvästi suosituin konttien hallintajärjestelmä. Kuberneteksessä on laajasti eri ominaisuuksia, joten sillä voi tehdä hyvinkin monimutkaisia konfiguraatioita. Työn muihin vertailukohtiin verrattuna Kubernetes on huomattavasti hankalampi oppia ja ottaa käyttöön. Ohjelman käytön aloitusta varten on tehty monia eri lisäohjelmia, mutta myös näitä hyödyntäen järjestelmän käyttöönotto on muita hankalampaa. Kubernetes sopii valituista ohjelmista parhaiten laajaan tuotantokäyttöön.

Docker Swarm Mode on tarkoitettu kahden tai useamman erillisen Docker ohjelmiston yhdistämiseen. Ohjelma on nopea ja helppo ottaa käyttöön, mutta se sisältää huomattavasti vähemmän ominaisuuksia kuin Kubernetes.

Docker Compose on yksinkertainen käyttää. Se on suunniteltu monista konteista koostuvien paikallisten testiympäristöjen ajamiseen. Sillä ei suoraan pysty hallitsemaan useampaa laitetta, joten se ei ole ideaali useamman laitteen tuotantoympäristössä. Yhden laitteen tuotantoympäristössä Docker Compose voi hyvinkin olla tarpeeksi hyvä ja monipuolinen.

Hallintajärjestelmää valittaessa tulee ottaa huomioon ohjelman käyttötarkoitus sekä sitä hallinnoivien tottumukset. Nämä esitellyt kolme järjestelmää ovat selvästi tarkoitettu eri käyttötarkoituksiin.

LÄHTEET

[1] Portnoy, M. *Virtualization essentials*. eng. Second edition. Indianapolis, Indiana: Sybex, 2016. ISBN: 1-119-26774-9.

[2] Aziz Shah, A., Piro, G., Alfredo Grieco, L. ja Boggia, G. A quantitative cross-comparison of container networking technologies for virtualized service infrastructures in local computing environments. eng. *Transactions on emerging telecommunications technologies* (2021). ISSN: 2161-3915.

[3] Pearce, M., Zeadally, S. ja Hunt, R. Virtualization: Issues, security threats, and solutions. eng. *ACM computing surveys* 45.2 (2013), 1–39. ISSN: 0360-0300.

[4] Popek, G. ja Goldberg, R. Formal requirements for virtualizable third generation architectures. eng. *Communications of the ACM* 17.7 (1974), 412–421. ISSN: 0001-0782.

[5] Mouat, A. *Using Docker*. eng. First edition. Sebastopol, CA: O'Reilly, 2015. ISBN: 1-4919-1575-7.

[6] VMware Platform Eliminates The 'One Server Per Application' Model For Accountants Inc., Improving Efficiencies and Flexibility. eng. *Business Wire* (2008).

[7] Jangla, K. *Accelerating Development Velocity Using Docker Docker Across Microservices*. eng. 1st ed. 2018. Berkeley, CA: Apress, 2018. ISBN: 1-4842-3936-9.

[8] *Google cloud Containers*. URL: `https://cloud.google.com/containers` (viitattu 13. 10. 2021).

[9] *About the Open Container Initiative - Open Container Initiative*. URL: `https://opencontainers.org/about/overview/` (viitattu 13. 10. 2021).

[10] *Docker overview Docker documentation*. URL: `https://docs.docker.com/get-started/overview/` (viitattu 17. 10. 2021).

[11] Kane, S. P. ja Matthias, K. *Docker: Shipping Reliable Containers in Production*. eng. Sebastopol: O'Reilly Media, Incorporated, 2018. ISBN: 1492036730.

[12] Burns, B., Beda, J. ja Hightower, K. *Kubernetes: Up and Running, 2nd Edition*. eng. 2. painos. O'Reilly Media, Inc, 2019. ISBN: 9781492046530.

[13] *Overview of Docker Compose*. URL: `https://docs.docker.com/compose/` (viitattu 23. 10. 2021).

[14] Farcic, V. *The DevOps 2.1 toolkit : Docker Swarm : building, testing, deploying, and monitoring services inside Docker Swarm clusters*. eng. 1st edition. Polymer Chemistry Series. Birmingham, [England: Packt Publishing, 2017. ISBN: 9781787280601.

[15] Vohra, D. Using Docker in Swarm Mode. eng. *Docker Management Design Patterns*. Berkeley, CA: Apress, 2017, 9–30. ISBN: 9781484229729.

[16] Burns, B., Beda, J. ja Hightower, K. *Kubernetes: Dive into the Future of Infrastructure*. eng. Sebastopol: O'Reilly Media, Incorporated, 2019. ISBN: 9781492046530.

[17] Rensin, D. *Kubernetes*. eng. 1st edition. O'Reilly Media, Inc., 2015. ISBN: 1-4920-4871-2.

[18] *Kubernetes Documentation*. URL: https://kubernetes.io/docs/home/ (viitattu 13. 10. 2021).