Tampereen yliopisto

Jerry Selin

KONTITUSTEKNIIKKA LINUX-YMPÄRISTÖSSÄ

Tekniikan ja luonnontieteiden tiedekunta
Kandidaatintyö
Tammikuu 2020

TIIVISTELMÄ

Jerry Selin: Kontitustekniikka Linux-ympäristössä
Kandidaatintyö
Tampereen yliopisto
Teknisten tieteiden kandidaatin tutkinto-ohjelma
Tammikuu 2020

Konttitekniikka on osa virtualisaatiota. Sen käyttö on yleistynyt palvelimilla, mutta muitakin käyttökohteita on olemassa. Teknologia kasvatti suosiotaan merkittävästi Linux-ympäristön myötä. Konttien avulla saadaan monissa käyttökohteissa virtuaalikoneen hyödyt kevyemmällä laitteiden kuormituksella. Konttiteknologian avulla sovellusten käytännöllinen siirto eri käyttöjärjestelmien kesken helpottuu sekä palvelimilla suoritettavat prosessit saadaan eristettyä toisistaan.

Tässä työssä tarkastellaan konttien sekä konttienhallintaohjelmien ominaisuuksia ja käyttökohteita. Konttien eristys perustuu käyttöjärjestelmän ytimen ominaisuuksiin. Tarkastelussa todettiin eri käyttötarkoituksen ohjelmistokonttien käyttävän ytimen ominaisuuksia eri tavoin.

Tutkimalla konttien käyttöä, totesimme myös, että konttien määrän kasvaessa manuaaliset työvaiheet kasvavat. Tätä ongelmaa varten tutkimme konttienhallintaohjelmia, sekä niiden rakennetta ja käyttötarkoitusta. Konttienhallintaohjelmien yhteneväisyydeksi löysimme ominaisuuden, jonka avulla hajonneen kontin tilalle käynnistetään automaattisesti uusi kontti. Myös hajonneen palvelimen kontit jaetaan automaattisesti toimiville palvelimille. Konttienhallintaohjelmien hyödyllisimmäksi käyttökohteeksi löysimme usean palvelimen infrastruktuurin.

Avainsanat: kontitustekniikka, Linux-kontit

SISÄLLYSLUETTELO

LYHENTEET JA MERKINNÄT

API	Application programming interface, ohjelmointirajapinta
CLI	Command-line interface, komentorivikäyttöliittymä
gRPC	Remote Procedure Calls, tietoliikenneprotokolla
HPC	High-Performance Computing, suurteholaskenta
MAC	Mandatory Access Control, pääsynvalvontaohjelma
JSON	JavaScript Object Notation, tiedostoformaatti
LXC	Linux Containers, konttiohjelmisto
REST	Representational State Transfer, arkkitehtuurimalli rajapintojen toteuttamiseen
TSL	Transport Layer Security, salausprotokolla
YAML	YAML Ain't Markup Language, tiedostoformaatti

1. JOHDANTO

Ohjelmistokehityksen haasteisiin kuuluu palvelinresurssien optimointi ja ohjelmistojen käytännöllinen julkaiseminen. Aiemmin palvelimilla olevia sovelluksen eristäviä virtuaalikoneita voidaan korvata konteilla. Huhtikuussa 2018 jopa 20 prosenttia valvontaohjelmisto Datadogia käyttävistä palvelinkoneista käytti myös konttiohjelmisto Dockeria [1]. Konttiteknologia ei kuitenkaan rajoitu palvelinsovelluksen eristämiseen, vaan konttiin on myös mahdollista eristää esimerkiksi sovelluksen osa tai vaikka kokonainen virtuaalikone. Konttien sisältöjen lisäksi myös konttien toteutustavat eroavat toisistaan.

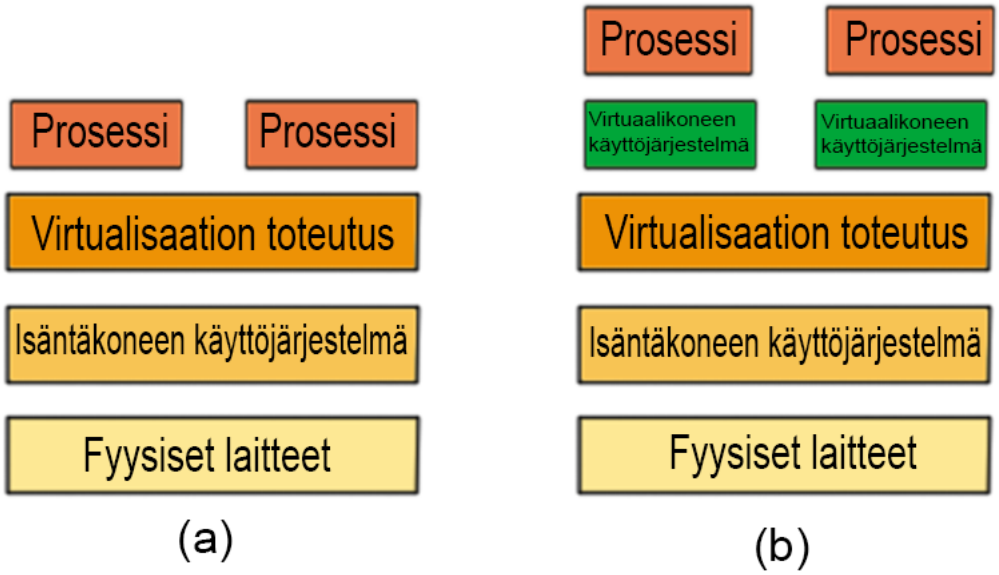
Työn tavoitteena on tutkia ohjelmistokonttien toimintaa sekä erilaisten konttien käyttötarkoituksia esimerkkikonttien avulla. Ohjelmistokontit ovat olleet olemassa Unix-käyttöjärjestelmissä pitkään. Myös Windows 10:n mukana on tullut ominaisuuksia, jotka mahdollistavat konttiteknologian. Tässä työssä kuitenkin keskitytään tutkimaan konttiteknologiaa Linux-ympäristössä. Aluksi työssä perehdytään Linux-ympäristön ominaisuuksiin, jotka mahdollistavat konttien luomisen. Tämän jälkeen tarkastellaan erilaisia kontteja, sekä niiden käyttämiä tekniikoita. Lopuksi vertaillaan konttienhallintaohjelmia toiminnan ja käyttötarkoituksen mukaan.

2. KONTTITEKNOLOGIA

Palvelimilla on perinteisesti ajettu ohjelmia yhdellä tai useammalla virtuaalikoneella, jotta palvelinohjelmille voitaisiin jakaa resursseja sekä ne voitaisiin eristää toisistaan. Konttien käyttö palvelimilla on yleistynyt viime aikoina. Palvelimilla suoritettavilla konteilla pyritään optimoimaan palvelimen resursseja entistä paremmin.

Konttiteknologia mahdollistaa käyttöjärjestelmän, sovelluksen tai sovelluksen osan suorittamisen eristetyssä ympäristössä. Ohjelma voidaan laittaa tarvittavine kirjastoineen konttiin, minkä jälkeen se pystytään suorittamaan eri Linux-versiolla ilman ongelmia [2]. Pieni sovellus yhdessä virtuaalikoneen kanssa voi viedä gigatavuja tilaa, kun taas kontitettuna sama sovellus vie vain kilotavuja tilaa [3].

Kontit ja virtuaalikoneet ovat osa virtualisaatiota, ja niillä on palvelinkäytössä usein sama käyttökohde. Konttien ja virtuaalikoneiden erona on, että konttien suorittama virtualisaatio tapahtuu käyttöjärjestelmän tasolla jakaen isäntäkoneen ytimen, kun taas virtuaalikoneiden virtualisaatio tapahtuu fyysisten laitteiden tasolla, jolloin virtuaalikone vaatii oman ytimen [4]. Ytimen jakamista on havainnollistettu kuvassa 1, jossa vasemmalla (a) on kuvailtu konttia ja oikealla (b) virtuaalikonetta. Vihreä laatikko tarkoittaa virtuaalikoneen käyttöjärjestelmää, johon sisältyy myös käyttöjärjestelmän ydin. Ytimen jakamisen vuoksi kontit kuluttavat vähemmän resursseja kuin virtuaalikoneet. Kontit voidaan helposti ajatella virtuaalikoneet korvaavaksi teknologiaksi, vaikka todellisuudessa ne toimivat hyvin yhdessäkin. Linuxilla luotuja kontteja ei ole mahdollista suorittaa muilla käyttöjärjestelmillä, kuitenkin muilla käyttöjärjestelmillä on mahdollista suorittaa kontteja Linux-virtuaalikoneen avulla [5].



Kuva 1. *Kontin ja virtuaalikoneen ero.*

2.1 Kontit yleisesti

Konttien avulla voidaan rajata kontin sisällä suoritettavan prosessin resursseja sekä eristää se muista prosesseista. Kontilla on oma IP-osoite sekä tiedostojärjestelmä, jonka sisällä olevat prosessit pystyvät suorittamaan komentoja kontin sisäisenä pääkäyttäjänä [6]. Myöhemmin toteamme, että riippuen sovellettavasta kontista, kontin sisällä olevalla käyttäjällä ei aina ole pääkäyttäjän oikeuksia.

Yleensä kontteja varten luodaan kuvatiedosto, joka sisältää tarvittavat tiedot ja rajoitteet kontin luomista varten [7]. Yhdestä kuvatiedostosta voi siis luoda useita samanlaisia kontteja, joita voi siirtää kehitysympäristöstä tuotantoympäristöön. Joillain ohjelmilla voi luoda erilaisia kontteja samasta kuvatiedostosta käyttämällä eri parametreja.

2.2 Konttien toiminta Linux-ympäristössä

Kontit Linux-ympäristössä ovat muusta järjestelmästä eristettyjä, yhden tai useamman, prosessin sarjoja [8]. Käsite ohjelmistokontti on kuitenkin häilyvä, joten konteissa voidaan käyttää useita eri tekniikoita tarvittavan eristyksen aikaansaamiseksi. Linux-ytimen nimiavaruuksien sekä Control Groups (CGroups) -ominaisuuksien yhdistämistä voidaan kuvailla termillä kontti [9]. Open Container Initiative (OCI) projektin tavoitteena on luoda konttiteknologialle standardeja, jotta kontteja voisi käyttää helpommin eri

ohjelmilla. OCI-projekti [10] sisältää tällä hetkellä kolme määritelmää. Määritelmät liittyvät kontin suorittamiseen, kuvatiedostoihin sekä kuvatiedostojen jakamiseen [10].

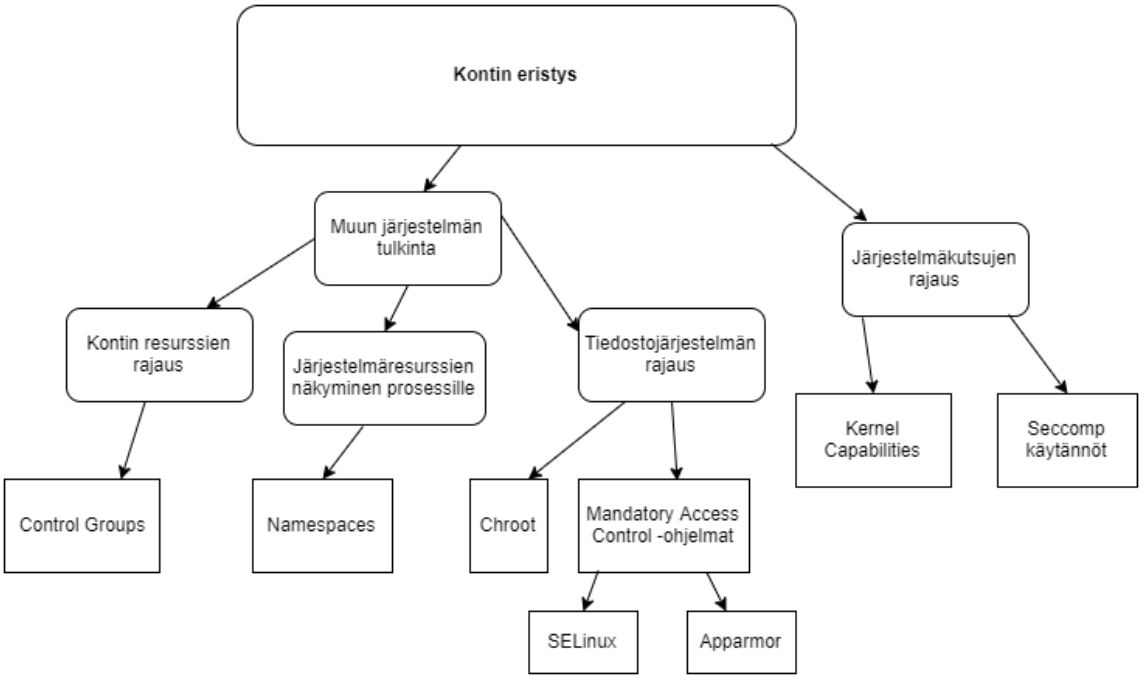
Kontit käyttävät Linuxin nimiavaruuksia luodakseen suoritettaville prosesseille omanlaisen kuvan järjestelmästä. Nimiavaruuksien avulla voidaan eristää kontti isäntäkoneesta monella tavalla, kuten rajaamalla pääkäyttäjän oikeudet toimimaan vain kontin sisällä. Nimiavaruuksien avulla voidaan myös estää ulkopuolisen prosessien näkyminen kontin sisälle sekä tehdä kontille oma verkkolaite ja IP-osoite. [5] Nimiavaruudet luovat globaalista järjestelmäresurssista ilmentymän, joka on näkyvillä vain nimiavaruuden sisällä. Resurssin muutokset eivät näy nimiavaruuden ulkopuolelle. Nimiavaruuksien tyyppejä ovat prosessien tunnisteet, käyttäjien ja ryhmien tunnisteet, prosessien väliset kommunikaatiot, verkot, tiedostojärjestelmien liittämiset sekä control groupit. [11] Nimiavaruuksien avulla voidaan näyttää yksi prosessi useassa eri prosessin tunnisteen nimiavaruudessa. Yksittäisellä prosessilla on yhtä monta prosessin tunnistetta, kuin nimiavaruuksia, johon prosessi kuuluu. [12] Käyttäjän nimiavaruutta lukuun ottamatta nimiavaruuksien luominen vaatii yleensä järjestelmänvalvojan oikeuksia [11].

Linux-ytimen control groups -ominaisuutta käytetään konttiteknologiassa eristämään, priorisoimaan sekä kirjaamaan resurssien käyttöä [6]. Eri resursseja varten on olemassa eri tyyppisiä ryhmiä. Esimerkiksi ryhmä voi vastata muistin käytön jakamisesta. Kontrolliryhmät ovat hierarkkisesti järjesteltyjä, joten periytetyissä kontrolliryhmissä pätee samat rajoitukset. Kontrolliryhmän sisällä voi olla useita eri prosesseja, minkä avulla voidaan rajata koko kontin käyttämiä järjestelmäresursseja. [13]

Kontit rajaavat usein sallittavia järjestelmäkutsuja luodakseen turvallisemman suoritusympäristön prosessille. Linux Capabilities tarkoittaa ytimen ominaisuuksiin liittyviin toimintoihin myönnettäviä käyttöoikeuksia. Järjestelmäkutsujen rajaamiseen käytetään myös Seccomp-käytäntöjä. Seccomp on ytimen ominaisuus, jonka avulla voidaan rajata järjestelmäkutsuja Capabilities-ominaisuutta tarkemmin [14].

Tyypillisesti konteissa rajataan kontin sisälle näkyvä tiedostojärjestelmä. Rajaamiseen on mahdollista käyttää useita eri tapoja. Pitkään Linuxin ominaisuutena on ollut Chroot-komento, jonka avulla ohjelma näkee valitun kansion tiedostojärjestelmän juurena [15]. Useimmiten konttien sisälle luodaan kuitenkin tiedostojärjestelmän eri osista yhdistelty kokonaisuus pelkän rajaamisen sijaan. Konttiteknologiassa paljon käytetyt Apparmor- ja Selinux-ohjelmat ovat Mandatory Access Control -järjestelmiä, joiden avulla voidaan evätä pääsy internetporttiin, tiedostopolkuun tai tiedostoon [16].

Kuvassa 2 on havainnollistettu tässä työssä käsiteltäviä kontin eristämiseen käytettäviä keinoja. Kuva ei sisällä kaikkia mahdollisia tapoja eristää konttia. Kuvassa on kuitenkin havainnollistettu yleisimpien eristystapojen toimintaa.



Kuva 2. *Kontin eristys.*

3. ERILAISIA KONTTIRATKAISUJA

Konttien sisällöt vaihtelevat yksittäisestä prosessista virtuaalikoneeseen, minkä takia konttitekniikoiden toteutuksissa on paljon eroja. Seuraavaksi tutkitaan konttien toimintaa sekä mahdollisia käyttötarkoituksia. Vertailuun on otettu mukaan konttitekniikoita, joiden käyttökohteet eroavat paljon toisistaan. Tällä tavoin saadaan selvitettyä konttien monipuolisia käyttötarkoituksia.

3.1 LXC

LXC (Linux-containers) on ensimmäinen konttiteknologia, joka on saanut paljon suosiota [17]. Ohjelma on avoimeen lähdekoodiin perustuva, ja sen ensimmäinen versio julkaistiin vuonna 2008 [18]. LXC-kontit perustuvat control groups -ominaisuuteen, joka kehitettiin samana vuonna kuin LXC julkaistiin. LXC-kontit ovat tehty käyttöjärjestelmän kontittamista varten, minkä vuoksi yksittäisen kontin sisällä voidaan suorittaa useita eri prosesseja samanaikaisesi [19].

LXC-konttien toiminnallisuus perustuu ytimen nimiavaruuksiin, CGroupsiin, Chroot-komentoihin, Apparmor- ja SELinux-profiileihin, Seccomp-käytäntöihin sekä Kernel Capabilitiesiin [20]. Chroot-komennon avulla ohjelma ei löydä tiedostoja, jotka sijaitsevat ylempänä tiedostojärjestelmässä.

Alkuperäiseen julkaisuun verrattuna kontteihin on lisätty monia turvallisuutta parantavia ominaisuuksia. Kontteja voidaan luoda sekä suorittaa tavallisen käyttäjän tai pääkäyttäjän oikeuksilla. LXC:n internetsivujen [21] mukaan kontit ilman pääkäyttäjän oikeuksia otettiin käyttöön vuonna 2014. Pääkäyttäjän oikeuksilla luotuja kontteja ei pidetä turvallisina, eikä niistä havaittuihin karkaamisiin reagoida nopeilla korjauksilla [21]. Tavallisen käyttäjän luomista konteista karkaamisen seurauksena vaarallinen ohjelma ei myöskään pääse pääkäyttäjän resursseihin.

Apparmor, Selinux, Seccomp ja Capabilities ovat välttämättömiä pääkäyttäjän oikeuksilla luodussa kontissa, mutta ominaisuuksia käytetään myös muissa konteissa [21]. Ilman käyttöoikeuksien myöntämistä tai eväämistä kontilla olisi oikeudet kaikkiin järjestelmäkutsuihin tai ei mihinkään riippuen siitä, onko kontti luotu pääkäyttäjän oikeuksilla. Kontti voidaan esimerkiksi luoda ilman pääkäyttäjän oikeuksia, mutta sallia CAP_WAKE_ALARM-ominaisuus. Kontista on nyt mahdollista asettaa ajastimia käynnistämään laitteen haluttuun aikaan.

LXC:n käyttökohteet ovat usein samat kuin virtuaalikoneilla. Palvelinkäytössä tietokanta on usein eristettynä eri virtuaalikoneeseen kuin palvelinsovellukset. Näin palvelusovelluksien altistuminen kyberhyökkäyksille ei altista tietokantaa hyökkäykselle. Konttien tuomana etuna voidaan palvelimella suorittaa useampaa palvelusovellusta kuin aikaisemmin. Konttien siirrettävyyden ansiosta niitä on helppo varmuuskopioida sekä palauttaa vanha sovellusversio takaisin palvelimelle. LCX-kontit eivät eristä käyttöjärjestelmän ydintä, minkä ansiosta konttien sisällä olevia tiedostoja voi hallinnoida helposti kontin ulkopuolelta [14]. Kontteja voidaan käyttää esimerkiksi internethotelleiden palvelimilla, jotta jokaista asiakasta varten ei tarvitsisi omaa fyysistä palvelinta [19].

3.2 Docker

Vuonna 2018 käytössä olevista konteista 83 prosenttia oli Docker-kontteja, kun taas vuonna 2017 prosenttimäärä oli 99 [23]. Dockeria on siis voinut pitää melkein synonyyminä konttiteknologialle. Aikaisemmin suuren suosion saavuttaneet LXC-kontit keräsivät vuoden 2018 raportissa vain yhden prosentin käyttöasteen [23].

Docker-kontit ovat suunniteltu yhtä prosessia varten, kuitenkaan luomatta uutta konttia, mikäli palvelupyyntö luo uuden prosessin [24]. Yksittäisellä kontilla ei siis yleensä ole tarkoitus pystyä korvaamaan virtuaalikonetta. Palvelinkäytössä Docker-kontteja tarvitaankin usein suurempi määrä kuin LXC-kontteja.

Aluksi Docker oli yhtenäinen ohjelmisto, joka sisälsi kaiken tarvittavan konttien suoritukseen. Nykyään se koostuu useista erillisistä komponenteista ja ohjelmista. Osa komponenteista, kuten konttien suorittamisesta vastaava containerd, ovat avoimen lähdekoodin sovelluksia [25]. Containerd perustuu runc-työkalun hyödyntämiseen konttien luomisessa, minkä lisäksi se sisältää kuvatiedoston hallinnan sekä ohjelmointirajapinnan. Työkalun käyttötarkoituksena on konttien luominen ja suorittaminen OCI-määrittelyn mukaisesti. Työkalun käytön seurauksena Docker-konttien käyttö puolestaan helpottuu muilla ohjelmilla. Runc-työkalun avulla käytetään ytimen ominaisuuksia kuten CGroupsia, Apparmor- ja SELinux-profiileja sekä Seccomp-käytäntöjä. [26]

Docker-konttien eristys on aikaansaatu namespaces-, control groups- sekä capabilities-ominaisuuksien avulla. Docker mahdollistaa myös muiden ohjelmien käytön, jotta konttien eristystä voitaisiin lisätä. Ohjelmia ei integroida Dockeriin, vaan Docker on tehty niin, että se toimii yhdessä ulkopuolisten turvaohjelmien kanssa. Konttien turvallisuutta voi lisätä esimerkiksi Mandatory Access Control -ohjelmilla tai tiedostojärjestelmän

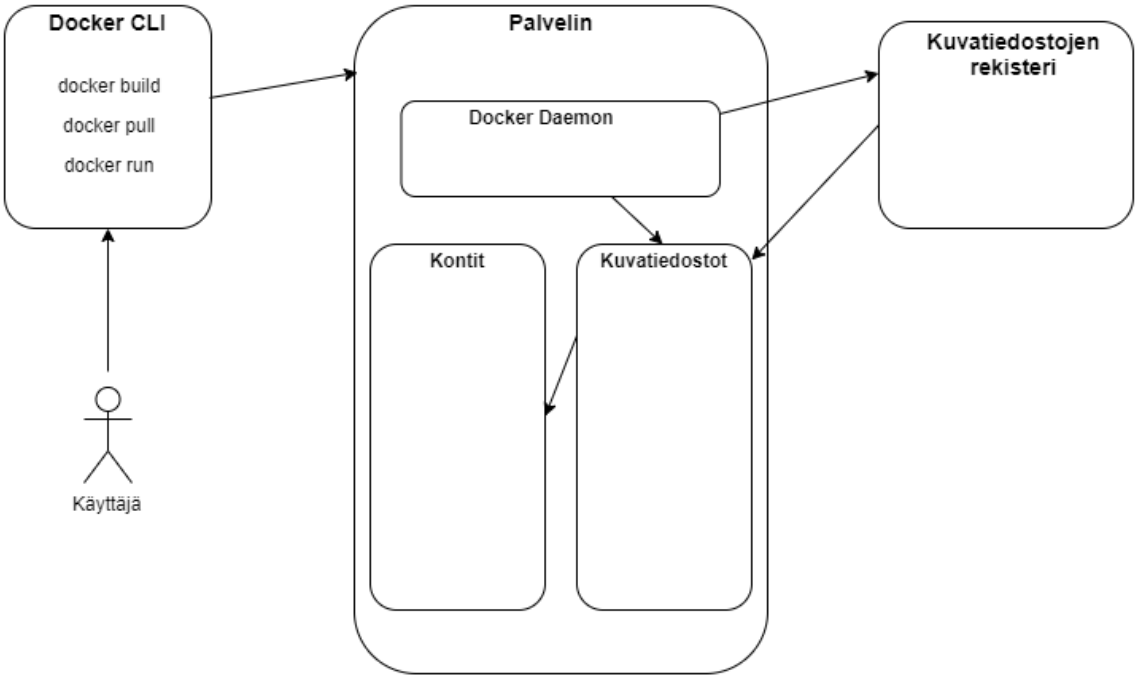
muutoksia tarkkailevalla ohjelmalla. [27] Runc-työkalu mahdollistaa konttien luomisen ilman pääkäyttäjän oikeuksia. Tällöin käytetään user-nimiavaruutta, jonka käyttö ei vaadi pääkäyttäjän oikeuksia. [28] Kontit ilman pääkäyttäjän oikeuksia vähentävät konttien suorittamisesta vastaavan taustaohjelma Docker daemonin sekä konttien haavoittuvuuksia. Konteissa ei kuitenkaan voi käyttää control groupeja, AppArmoria, kontin tilojen tallennusta, nettiyhteyksien kerrostamista tai SCTP-protokollan (Stream Control Transmission Protocol) tiedonsiirtoon käytettävien porttien paljastamista. [29]

Containerd sisältää oman nimiavaruuksien toteutuksen, jota käytetään eri tarkoitukseen kuin Linux-ytimen nimiavaruuksia. Containerdin nimiavaruuksia voidaan käyttää jakamaan kontteihin liittyviä käsitteitä eri käyttäjille. Tällä tavoin kahden eri käyttäjän on mahdollista suorittaa saman nimisiä, mutta toisistaan eroavia kontteja tai kuvatiedostoja. Ominaisuuden avulla käyttäjän poistama kuvatiedosto ei poistu muilta käyttäjiltä. [30]

Docker toimii asiakkaan ja palvelimen periaatteella REST (Representational State Transfer) API:n (Application programming interface) välityksellä, jolloin palvelimella toimiva Docker daemon -ohjelma hoitaa kuvatiedostojen sekä konttien rakentamisen ja hallinnoinnin [31]. Docker daemonia käytetään pääsääntöisesti komentoriviltä Docker CLI:n (Command-line interface) API:n avulla, mutta Docker daemonia voi käyttää myös muilla Dockerin tai kolmansien osapuolien työkaluilla [32]. Daemon ja komentorivi voivat olla asennettuina samalle tai eri palvelimelle. Tällä tavoin Docker soveltuu usean palvelimen infrastruktuuriin. Kuvassa 3 on vasemmalla Docker CLI ja keskellä palvelinperiaatteella toimiva Docker daemon.

Docker-ohjelmistoon kuuluu Docker hub, joka on konttien luomiseen käytettävien kuvatiedostojen yksityinen tai julkinen rekisteri. Docker hubin avulla käyttäjän ei tarvitse luoda kuvatiedostoja, vaan kuvatiedostot voi ladata valmiista malleista. Kuvatiedostojen jakopaikkoja on myös kolmansien osapuolien tekeminä. Kuvassa 3 kuvatiedostojen rekisteri on merkitty oikeaan yläkulmaan.

Kuva 3 havainnollistaa Dockerin komponenttien työnjakoa konttien luomisessa. Käyttäjä kutsuu komentorivikäyttöliittymästä Docker daemonia, joka komennon perusteella hakee kuvatiedoston yksityisestä tai julkisesta kuvarekisteristä. Kuvatiedoston haun jälkeen Docker daemon huolehtii konttien suorittamisesta haettujen kuvatiedostojen perusteella.



Kuva 3. *Dockerin rakenne.*

Yleensä hajonneen kontin tilalle käynnistetään uusi kontti. Vaikka kontin tekemät muutokset tiedostojärjestelmään säilyvät, uudelleen käynnistäessä muisti ja käynnissä olevat prosessit eivät säily. Dockerin avulla kontin tilan voi tallentaa. Tallennetusta tilasta voi palauttaa kontin tai luoda uuden kontin. Tällä tavoin kontin muisti ja prosessit eivät katoa. Kontin käynnistäminen on myös nopeampaa suorittaa tallennetusta tilasta kuin luomalla kokonaan uusi kontti. Tilojen tallennuksessa ja lataamisessa käytetään CRIU-työkalua. [33]

Docker on sopivin konttiratkaisu useimpiin haasteisiin. Konttiteknologian suosion nostanut ohjelma sisältää monia kehittyneitä ominaisuuksia. Dockerin hyödyt ovat parhaimmillaan usean kehittäjän projekteissa, jolloin konttiteknologialla saadaan helpotusta sovelluksen siirtämiseen kehitysympäristöstä tuotantoympäristöön. Docker-konteista koostuvan laajan ohjelmistokokonaisuuden hallinta helpottuu Dockerin avulla. Hajonneen tai haavoittuvaisen kontin palauttaminen aiempaan versioon onnistuu nopeasti.

3.3 Singularity

Singularity on tarkoitettu tieteelliseen käyttöön helpottamaan sovelluksen siirtämistä suurteholaskentaan. Merkittävä ero palvelimien ja suurteholaskennan käytössä on, että suurteholaskennassa tietokoneella on useita käyttäjiä, eikä käyttäjillä ole ylläpitäjän

oikeuksia. Singularityn kehittäjien [34] mukaan Docker ei vastannut tarpeeksi hyvin suurtehotietokoneiden vaatimuksia. Dockerin luomat kontit ovat luotu pääkäyttäjän omistaman Docker daemonin jälkeläisinä. Tämän ansiosta tavallisen käyttäjän on periaatteessa mahdollista saada itselleen pääkäyttäjän oikeudet. Pääkäyttäjän oikeuksien saaminen on suuri turvallisuusriski monen käyttäjän kesken jaetussa ympäristössä. [34] Singularity suorittaa tavallisesti palvelut etualalla, mutta palveluita on mahdollista suorittaa myös daemonin tapaisesti taustalla [35].

Singularityn käyttötarkoitus ja rakenne eroaa paljon palvelinkäyttöön kehitetyistä konteista. Sen kuvatiedostojen pätevyys ja alkuperäisyys on helposti varmistettavissa. Singularityn tarkoituksena on olla yhteensopiva kaikkien suurteholaskentaan tarkotettujen tietokoneiden kanssa. Ohjelman voi jopa asentaa laitteelle, joka ei tue käyttäjän nimiavaruuksia. Tavallisessa palvelinkäytössä ohjelmistokehittäjä käyttää konttiteknologiaa siirtääkseen luotettavaa ja mahdollisesti itse kehittämää ohjelmistoa. Singularityn turvallisuusmalli perustuu tilanteeseen, jossa epäluotettava käyttäjä suorittaa epäluotettavia kontteja. Singularityn on siis luotava turvallinen suoritusympäristö myös tätä käyttötapausta varten. [36]

Singularity-kontin sisällä käyttäjällä on samat oikeudet kuin kontin ulkopuolella, kuitenkin kaikkien resurssien käyttäminen ei vaadi pääkäyttäjän oikeuksia [15]. Toisin kuin LCX ja Docker, Singularity ei alunperin käyttänyt ollenkaan CGroups ominaisuutta [37]. Singularityn versioon 3.0 on lisätty mahdollisuus käyttää control groups -ominaisuutta, mutta sitä ei käytetä vakiona konteissa [38]. Aikaisemmin vertailluissa konttitekniikoissa control groups -ominaisuutta on käytetty rajoittamaan kontin käyttämiä resursseja, kuten muistin käyttöä tai prosessorin käyttöaikaa. Kyseiset konttitekniikat olivat tarkoitettu konttien eristämiseen siirrettävyyden lisäksi. Koska Singularityn avulla ei ole tarkoitus eristää, vaan pystyä siirtämään sovellus mahdollisimman helposti, siinä on käytetty namespaces-ominaisuutta eri tavoin kuin LXC- tai Docker-konteissa [39]. Singularity käyttää normaalisti vain mount-nimiavaruutta, jonka avulla voidaan eristää tiedostosijainteja. Käyttäjä voi kuitenkin eristää konttia enemmän käyttämällä muitakin nimiavaruuksia. [38]

Docker kuvatiedostoja sekä Docker Hub -kuvatiedostojen jakopaikkaa voi käyttää Singularity-ohjelmistolla ilman Docker enginen lataamista [34]. Singularity pystyy muuntamaan Docker-kuvatiedostoja Singularity-kuvatiedostoiksi. Dockerin tukemiseen on johtanut mm. kehittäjien kiinnostus ohjelmaa kohtaan sekä tutkijoiden näkemä vaiva jo olemassa olevien Docker-kuvatiedostojen rakentamisessa. [40]

Singularity sopii parhaiten ongelmiin, joissa vaaditaan sovelluksen helppoa siirrettävyyttä. Yleiseen palvelinkäyttöön Singularity ei sovellu hyvin, koska kyseisen ohjelmistokontin tarkoituksena ei ole eristää prosesseja. Helmikuussa 2019 julkaistun päivityksen myötä Singularity-kontit ovat täysin OCI-projektin standardien kanssa yhteensopivia [41]. Singularitysta on eniten hyötyä sovelluksen siirtämisessä suurteholaskentaan. Ohjelmasta voi olla hyötyä myös tavallisten tietokoneiden välisessä konttien siirtämisessä varsinkin OCI-yhteensopivuuden sekä Docker kuvatiedostojen tukemisen ansiosta.

3.4 Snap

Snap on Canonical Ltd:n kehittämä pakettiformaatti, joka on tarkoitettu työpöytä- ja IoT-sovelluksien jakamiseen. Snap-pakettien avulla voidaan rajata IoT-laitteen nettiyhteyksiä sekä I/O porttien käyttöä [42]. Pakettiformaatin lisäksi ohjelmistokokonaisuus koostuu saman nimisestä komentorivikäyttöliittymästä, snapd-taustaohjelmasta, snapcraft-viitekehyksestä sekä Snap Store -sovelluskaupasta. Ohjelmia voi paketoida Linuxilla Snap-pakettiformaattiin, minkä jälkeen niitä voi jakaa ja hallinnoida samannimisellä pakettienhallintajärjestelmällä [43]. Pakettien luomiseen käytetään snapcraft viitekehystä [44]. Ohjelma toimii Dockerin tavoin asiakkaan ja palvelimen periaatteella, jolloin snapd toimii palvelimella suoritettavana taustaohjelmana, kun taas snap-pakettienhallintajärjestelmä toimii asiakkaana. Molemmat ohjelmat ovat yleensä asennettuina samalle tietokoneelle. Snap-pakettiformaatin sovelluksia voi ladata Snap Storesta, mutta myös muualta ladattujen sovelluksien asennus on mahdollista. Snap-paketteja voidaan pitää kontteina, sillä ne suoritetaan omassa hiekkalaatikossaan eristettynä muilta ohjelmilta. Pakettiformaattiin sisällytetään sovelluksen lisäksi tarvittavat kirjastot sekä eristämiseen tarvittavat tiedot [42]. Kirjastojen sisällyttäminen mahdollistaa ohjelmien helpon asennuksen jopa yhdellä komennolla.

Snap-pakettien eristykseen käytetään AppArmor-profiileja, tiedostojärjestelmän rajaamista, seccomp-käytäntöjä, devpts-virtuaalista tiedostojärjestelmää, yksityistä väliaikaisten tiedostojen kansiota sekä cgroupsin device-ominaisuutta [45]. Vähäisen cgroupsien käytöstä seuraa, ettei esimerkiksi sovellukselle annettavaa muistia tai prosessorin käyttöä rajoiteta. Snapcraftin dokumentaation [45] mukaan Seccomp-käytäntöjen avulla listataan sallitut järjestelmäkutsut. AppArmor-profiilit rajaavat Capabilities-ominaisuuden avulla ytimen toimintojen käyttöoikeuksia. AppArmor huolehtii myös tiedostojärjestelmien yhdistämisestä, tiedostoihin pääsystä, ohjelmien suorittamisesta, prosessien valvonnasta ja ohjauksesta ptrace-järjestelmäkutsujen

avulla, prosessien välisestä kommunikaatiosta, signaaleista sekä korkean tason verkkotoiminnasta. [45]

Snap-paketeista saatavat hyödyt ovat osittain samat kuin muussa konttiteknologiassa. Kirjastoista riippumattomat, sekä eristetyt sovellukset helpottavat niiden käyttöä ja vähentävät sovelluksista aiheutuvia ristiriitoja. Paketit voi asettaa päivittymään automaattisesti, minkä lisäksi käyttäjä voi valita haluaako esimerkiksi sovelluksen vakaan- vai betaversiojulkaisun [46]. Snap-paketin tekeminen onnistuu määrittelemällä snapcraft.yaml tiedosto sekä oheistiedostot, kuten kuvakkeet. Tämän jälkeen paketin voi luoda komentoikkunan avulla suorittamalla snapcraft komennon yaml-tiedoston sijainnissa. [47] Sovelluskehittäjä hyötyy snap-paketeista niiden laajan soveltuvuuden ansiosta. Snap paketteja tukee useimmat tunnetut Linux-jakelut kuten Ubuntu, Linux Mint, Debian sekä Fedora [48].

4. KONTTIEN HALLINTA

Kontit saattavat kuulostaa erittäin käytännölliseltä ja turhia työvaiheita poistavalta teknologialta. Kuitenkin isommissa ohjelmistoissa saattaa olla todella paljon kontteja. Konttien resurssien asettaminen sekä manuaalinen siirtäminen ovat työläitä prosesseja useiden konttien järjestelmässä. Konttienhallintaohjelmat pyrkivät automatisoimaan useille konteille suoritettavia toimenpiteitä. Konttienhallintaohjelman avulla voidaan esimerkiksi säätää kontin tarvitsemia resursseja perustuen sen tarpeeseen, sekä palauttaa ehjä kontti rikkinäisen tilalle.

Konttien hallinnan vaatimuksiin vaikuttaa palvelimien määrä. Konttienhallintaohjelmien suurimpiin haasteisiin kuuluu tietojen siirtäminen kontin mukana, jos kontti suoritetaankin eri palvelimella [49]. Konttienhallinnalle onkin tyypillistä, että ohjelmisto valitsee kontille parhaiten soveltuvan palvelimen. Konttienhallintaohjelmalla pyritään usein myös parempaan palvelinkapasiteetin hyödyntämiseen.

4.1 Docker Swarm

Docker Swarm on Dockerin mukana tuleva konttienhallintaohjelma. Ohjelma mukailee Dockerin komentoja. Erona on kuitenkin, että yhden palvelimen sijaan kontteja suoritetaan usealla eri palvelimella. Docker Swarmin käyttöönotto onnistuu helposti, koska se on integroituna Docker-kontitusohjelmaan ja käyttäminen tapahtuu Dockerin tavoin Docker CLI:n avulla. Docker Swarm on tehokas, mutta sisältää rajallisen määrän ominaisuuksia verrattuna laajempiin konttienhallintaohjelmiin. Konttienhallintaohjelma on suunniteltu toimimaan korkealla suorituskyvyllä sekä toimimaan useiden tuhansien palvelimien ryhmissä. Ohjelmasta on tehty turvallinen, vaikka käyttäjä ei muuttaisi mitään asetuksia. Ilman käyttäjän tekemiä toimenpiteitä, kaikki tietokoneiden välinen liikenne käyttää molemminpuolista TLS-protokollaa (Transport Layer Security) [50]. Protokollaa käytetään todentamaan osapuolet, toisen koneen valtuuksien tarkastukseen sekä tietoliikenteen salaukseen [51].

Ohjelmassa jaetaan tietokoneet isäntä- ja työskenteleviksi solmuiksi, minkä lisäksi yksi isäntäsolmu on muita korkeammassa asemassa. Palvelimien jaottelu tehdään automaattisesti, mutta sitä on mahdollista muuttaa. Työskentelevillä solmuilla suoritetaan kaikki ohjelman kontit. Niiden vastuulla on myös väärälle kontille saapuvien pyyntöjen reititys sekä oman tilan ilmoittaminen isäntäsolmuille. Isäntäsolmujen

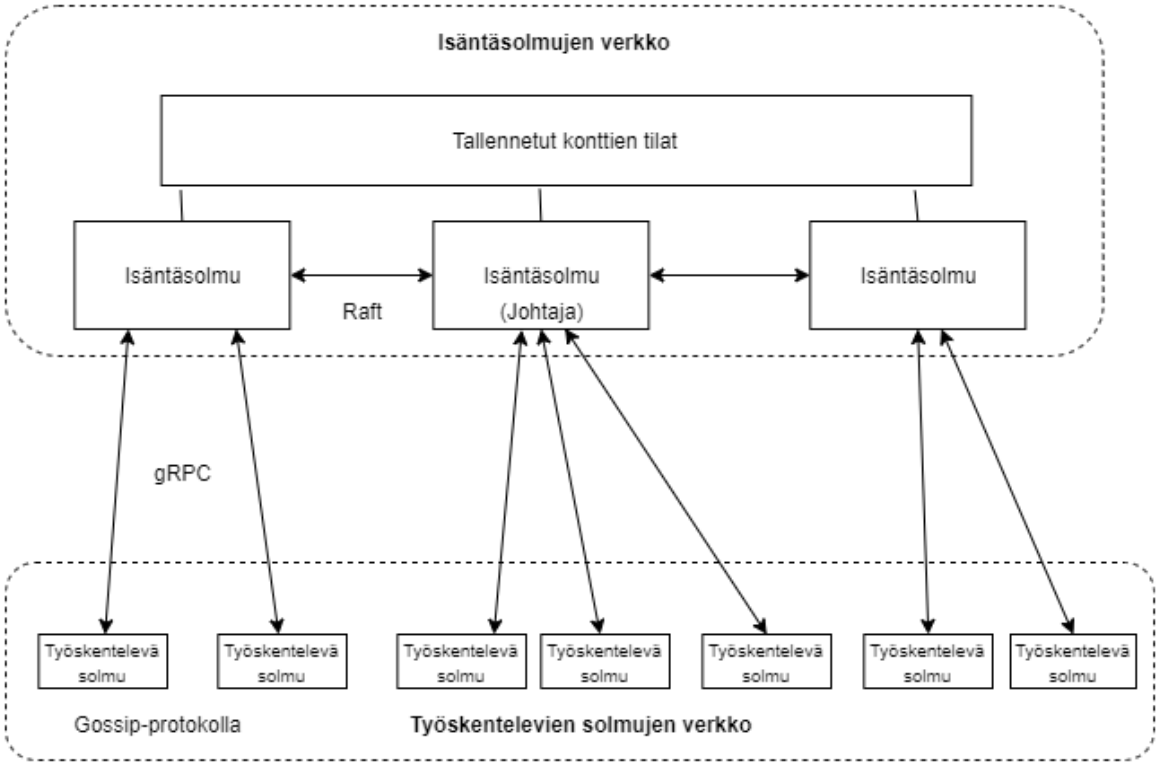
vastuulla on organisoida konttien luominen ja ylläpito. Isäntäsolmut tarkkailevat, mikäli työskentelevä solmu tai kontti lakkaa toimimasta, minkä seurauksena ne järjestävät kontteja eri solmuille tai käynnistävät uusia. Palvelin voi olla sekä isäntäsolmu, että työskentelevä solmu. Kuitenkin palvelimien jaottelu joko isäntäsolmuksi tai työskenteleväksi solmuksi vähentää haavoittuvuutta, mikäli konttia suorittavalle palvelimelle hyökätään. [50]

Isäntäsolmujen väliseen tiedonsiirtoon käytetään Raft-algoritmia. Raft-algoritmin mukaan yksi isäntäsolmuista valitaan johtajaksi. Johtaja ilmoittaa muille isäntäsolmuille tietovaraston muutoksista. Tämän jälkeen muut isäntäsolmut muuttavat niiden paikallista tietovarastoaan. Tällä tavoin Johtavan solmun hajottua se voidaan helposti korvata eri solmulla, koska kaikkien solmujen paikalliset tiedot ovat yhteneviä. Toteutus on myös suorituskykyinen, koska kaikki isäntäsolmut pystyvät lähettämään nopeasti paikallisesta tietovarastosta tiedon konttien tiloista. [50] Isäntäsolmujen vikasietoisuutta rajoittaa Raft-protokolla, jonka mukaan isäntäsolmut pystyvät toimimaan, jos yli puolet niistä on käytössä.

Työskentelevien solmujen ja isäntäsolmujen välinen kommunikaatio tapahtuu internetin välityksellä gRPC-protokollan avulla. Protokolla sisältää sisäisen versioinnin, joka mahdollistaa kommunikoinnin, vaikka palvelimilla suoritetaan eri versiota Dockerista. Isäntäsolmu lähettää työskentelevälle solmulle suoritettavia tehtäviä, kun taas työskentelevä solmu lähettää tehtävien tiloja sekä vahvistuksen, että kontit ovat toimintakunnossa. [50]

Työskentelevien solmujen välinen kommunikaatio tapahtuu vertaisverkossa gossip-protokollan mukaisesti. Docker Swarmissa työskentelevät solmut ilmoittavat toisilleen, mikäli ne aloittavat uuden kontin suorittamisen. Tiedotus tapahtuu ennalta valitulle vakiomäärälle solmuja. Kun solmu saa tiedon toiselta solmulta, se lähettää tiedon eteenpäin vakiomäärälle satunnaisia muita solmuja. [50]

Kuvassa 4 on esitetty Docker Swarmin rakenne. Ylhäällä on esitetty Isäntäsolmut, jotka vastaanottavat käyttäjältä saadut komennot. Isäntäsolmujen välinen kommunikaatio tapahtuu johtajan kautta. Isäntäsolmujen ja työskentelevien solmujen välisessä kommunikaatiossa isäntäsolmu ja sille annetut työskentelevät solmut ovat vuorovaikutuksessa. Työskentelevien solmujen välistä vuorovaikutusta ei ole merkitty nuolilla kuvaan, koska se on gossip-protokollan mukaista satunnaisille solmuille lähetettyä tietoa.



Kuva 4. *Docker Swarmin rakenne*

Docker Swarm soveltuu hyvin yksinkertaiseen konttienhallintaan, mutta myös mittavankin kokoisen palvelininfrastruktuurin käyttäminen onnistuu sillä tehokkaasti. Docker Swarmin käyttö on nopeaa oppia erityisesti Dockeriin perehtyneille henkilöille tuttujen komentojen sekä mukana tulevan asennuksen avulla. Docker Swarm tarjoaa rajalliset, mutta tehokkaat ominaisuudet, minkä ansiosta se soveltuu hyvin tilanteisiin, joissa halutaan hallita useita kontteja sekä niille asetettavia resursseja. Konttien monimuotoisuuden ja määrän kasvaessa taas vaihtoehtoiset laajemmat konttienhallintaohjelmat voivat osoittautua hyödyllisemmiksi. Swarmin ominaisuuksiin ei kuulu muun muassa konttien automaattista skaalausta, eli kontit eivät käynnisty tai sammu automaattisesti käyttöasteen perusteella. Docker ei yritä korvata muita konttienhallintaohjelmia yksinkertaisella Docker Swarmilla, vaan on luonut esimerkiksi oman Dockeriin integroidun jakeluversion Kubernetes konttienhallintaohjelmasta [51].

4.2 Kubernetes

Kubernetes on alun perin Googlen kehittämä konttienhallintaprojekti, joka on luovutettu Cloud Native Computing Foundationille [52][53]. Kuberneteksen avulla kontteja voi suorittaa usealla palvelimella samanaikaisesti, mutta ohjelman avulla kontit käyttäytyvät kuin ne suoritettaisiin samalla palvelimella [54, s. 17]. Kuberneteksen käytöstä on siis enemmän hyötyä usean palvelimen ympäristössä. Myös yhden palvelimen

infrastruktuurissa Kuberneteksesta hyötyy esimerkiksi siten, että hajonneen kontin tilalle käynnistetään automaattisesti uusi kontti. Usean palvelimen ympäristössä Kubernetes osaa jakaa hajonneella palvelimella suoritetut kontit toimiville palvelimille [54, s. 21]. Kubernetes parantaa tällä tavoin konttien toimintavarmuutta.

Kuberneteksen pienimpänä perusyksikkönä on pod, joka sisältää yhden tai useamman kontin [54, s. 56]. Kubernetes ei siis käsittele yksittäisiä kontteja sellaisenaan. Podin sisältämät kontit suoritetaankin aina samalla palvelimella, minkä lisäksi niissä käytetään samaa Linuxin nimiavaruutta [54, s. 43]. Kontit kannattaa suorittaa omissa podeissaan, mikäli ne eivät käytä yhteisiä resursseja kuten jaettua tiedostojärjestelmää. Tällaisen jaottelun avulla Kubernetes pystyy jakamaan kontit tasaisesti palvelimien kesken, sekä kasvattamaan konteille annettavia resursseja paremmin. Podeja on havainnollistettu kuvassa 5 valkoisilla nelikulmioilla. Nelikulmioiden sisällä olevilla värikkäillä geometrisilla kuvioilla merkitään kontteja. Värikäs geometrinen kuvio ilman ympäröivää nelikulmiota tarkoittaa kontin luomiseen käytettävää kuvaa.

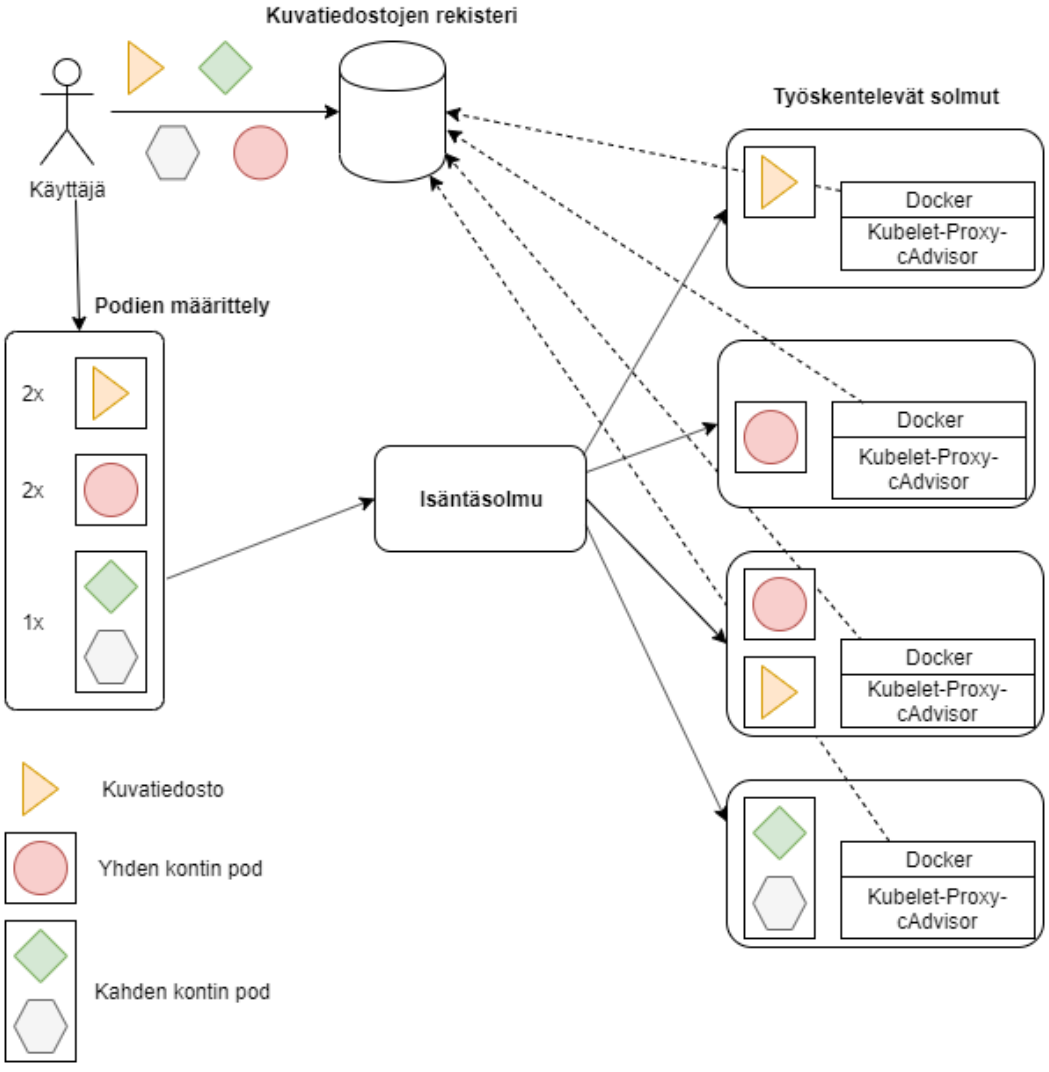
Podien luomiseen käytetään pääsääntöisesti YAML- tai JSON-tiedostoja, mutta podeja on mahdollista luoda myös komentoriviltä kubectl run -komennolla, jolloin kaikkia ominaisuuksia ei pysty valitsemaan [54, s. 61]. Alustustiedoston avulla voidaan määritellä mm. mihin palvelinryhmään podi sijoitetaan, esimerkiksi sijoituskohteeksi voidaan valita palvelin, jossa on käytössä SSD-levy [54, s. 22]. Kuvassa 5 on havainnollistettu podien määrittämistä käyttäjän alapuolella olevalla laatikolla. Kertoimet ennen kuvioita havainnollistavat, montako kopiota podeista halutaan luoda.

Kubernetes jakaa palvelimet isäntäsolmuiksi (master node) sekä työskenteleviksi solmuiksi (worker node). Työskentelevien solmujen sisällä suoritetaan yhtä tai useampaa podia. Isäntäsolmut vastaavat podien jakamisesta tasaisesti. Master- ja worker nodet kommunikoivat keskenään ohjelmointirajapinnan avulla. Isäntäsolmut suorittavat kolmea tärkeää ohjelmaa, joita ovat API Server, Etcd sekä Scheduler and Controller Manager. API Server vastaa työskentelevien solmujen ohjelmointirajapinnan kautta lähettämien kutsujen suorittamisesta. Etcd on hajautetuille järjestelmille suunniteltu avain-arvo-parien tallennuspaikka, joka valvoo palvelimien sekä niillä suoritettavien konttien tilaa ja vertaa sitä tunnettuihin järjestelmän tilaa kuvaaviin avain-arvo-pareihin. Etcd pyrkii ylläpitämään palvelimien tilan. Scheduler and Controller Manager -ohjelma ajoittaa podien sijoittamista työskenteleville solmuille. Ohjelma myös vastaa suoritettavien podien oikeasta määrästä.

Työskentelevässä solmussa suoritetaan yleensä kolmea tärkeää prosessia, jotka ovat Kubelet, Proxy ja cAdvisor. Kubelet on taustalla suoritettava prosessi, joka luo, tuhoaa

ja valvoo suoritettavia kontteja. Proxy on välityspalvelin, joka jakaa saapuvat pyynnöt podeille. cAdvisor kerää tietoa konttien resurssien käytöstä, sekä suorituskyvystä. [55, s.8-9, s. 22]

Kuvassa 5 isäntäsolmu on podien määrittelyn oikealla puolella. Isäntäsolmu valitsee podeille sopivat sijoituspaikat JSON- tai YAML-tiedoston perusteella. Loput podit isäntäsolmu sijoittaa niin, että resurssit saadaan hyödynnettyä parhaalla mahdollisella tavalla. Oikean alakulman palvelin voi olla tyhjä esimerkiksi silloin, jos muilla palvelimilla on enemmän resursseja käytettävissä ja isäntäsolmu antaa tehokkaammalle palvelimelle usean podin. Kuvassa näkyy myös, kuinka konttiohjelman vastuulla on hakea kuva kuvarekisteristä.



Kuva 5. *Kuberneteksen toiminta.*

Kubernetes-ohjelma pystyy hallitsemaan useita eri kontteja. Lisäosana toimiva CRI-O:n avulla voidaan suorittaa OCI-yhteensopivia kontteja [56]. Kuberneteksen avulla voi

periaatteessa suorittaa mitä tahansa kontteja CRI (Container Runtime Interface) ohjelmointirajapinnan avulla, kunhan konteille on määritelty CRI:n mukaiset toiminnallisuudet [57].

Kuberneteksen tarkoitus on mahdollistaa saman käyttöliittymän avulla ohjelmistokokonaisuuksien käyttö fyysisellä palvelimella, virtuaalikoneella tai vaikkapa pilvipalvelussa. Kuberneteksen avulla ohjelmiston voi esimerkiksi jakaa osittain pilvipalveluun palvelinsalien lisäksi. Pilvipalvelua suosiville käyttäjille voi olla taloudellista hyötyä Kuberneteksen käytöstä. Julkisen pilven hinnoittelu saattaa perustua prosessorin käyttöaikaan, jolloin Kuberneteksen optimoinnilla voidaan aikaansaada säästöjä. [58] Tärkeä Kuberneteksen ominaisuus on palvelimien suorituskyvyn ylläpito kuormituksen muuttuessa. Konttien korvaamisen ja hajonneen palvelimen konttien jakamisen lisäksi Kubernetes osaa tasapainottaa konteille jaettavia resursseja sekä käynnistää tai sammuttaa kontteja niihin kohdistuvan kuormituksen perusteella. Kuberneteksesta on siis hyötyä lähes aina, kun palvelinkäytössä käytetään konttiteknologiaa. Kuberneteksesta on olemassa paljon eri jakeluversioita, joten ohjelmalla voidaan vastata monenlaisiin käyttötarkoituksiin. Alkuperäisestä avoimesta lähdekoodista on muokattu versioita myös kaupalliseen käyttöön.

4.3 Kontena Lens

Kontena Lens on Kontena, Inc:n tuottama graafinen käyttöliittymä ja lisäosa Kubernetekselle. Kontena sisältää monia Kuberneteksesta puuttuvia työkaluja konttienhallintaan. Ohjelmistoon voi esimerkiksi liittää autentikoinnin, jolloin käyttäjälle voidaan rajata oikeudet vain tiettyyn konttiryhmään. Ohjelmaa voi käyttää Windowsilla, Linuxilla, MacOS:llä tai selaimella. [59]

Kontena Lens tarjoaa helpon käyttöliittymän, sekä visualisoitua dataa tämänhetkisistä ja menneistä konttiryhmien tiloista [59]. Ohjelma sopii käyttäjille, jotka haluavat tavallista Kuberneteksen käyttöliittymää enemmän visualisoitua dataa. Historiatiedoista koottua dataa voidaan hyödyntää esimerkiksi konttien toiminnan arviointiin. Autentikoinnista voi olla myös hyötyä isoissa organisaatioissa. Jaottelemalla konttien käyttöoikeudet käyttäjäryhmien perusteella, voidaan vähentää konteille vahingossa tehtyjä toimenpiteitä.

5. YHTEENVETO

Konttiteknologia on hyödyllinen innovaatio ohjelmistotuotannossa. Siitä on kuitenkin hyötyä myös muilla sovellusalueilla. Työssä todettiin, kuinka konttiteknologiaa voitiin soveltaa myös suurteholaskentaan, esineiden internet -sovelluksiin sekä tavallisten työpöytäsovelluksien käytännölliseen jakamiseen ja turvalliseen suorittamiseen.

Konttien todettiin perustuvan käyttöjärjestelmän ytimen ominaisuuksiin. Konttien eristämiseen käytettyjä tapoja tutkiessa totesimme, että esimerkiksi tiedostojärjestelmän eristämiseen on useita mahdollisia tapoja. Yksittäisiä konttitekniikoita vertailemalla todettiin, että eri tarkoituksiin kehitetyt kontit käyttävät ytimen ominaisuuksia eri tavoilla.

Konttiteknologian suosio on kasvanut merkittävästi tällä vuosikymmenellä Dockerin julkaisun ansiosta. Konttitekniikoiden ja konttienhallintaohjelmien lisäksi myös konttien käyttötarkoitukset kehittyvät. Konttien käyttötarkoitukset eivät rajoitu pelkästään palvelimella käytettävän virtuaalikoneen korvaamiseen. Työssä tutkittiin Snap-työkalua, jonka avulla Linux-työpöytäsovellukset voidaan paketoida kontteihin. Muiden konttien tavoin sovelluksien mukaan paketoidaan tarvittavat kirjastot ja kontit pystytään suorittamaan monilla eri Linux-jakeluilla. Snap-paketoidut sovellukset suoritetaan kontitettuina, jolloin niiden suorittaminen on perinteistä työpöytäsovellusta turvallisempaa.

Työssä käsiteltiin myös konttienhallintaohjelmien toimintaa. Konttienhallintaohjelmien rakenne on tehty niin, että kontit voidaan jakaa helposti usealle eri palvelimelle. Konttienhallintaohjelmat osaavat myös reagoida palvelimien kaatumiseen käynnistämällä kontit uudestaan eri palvelimella. Konttienhallintaohjelmista saa suurimman hyödyn, kun palvelimien määrä on suuri. Konttienhallintaohjelmat voivat myös auttaa toimintavarmuudessa, vaikka käytössä olisi vain yksittäinen palvelin.

Työn tavoitteet saavutettiin, erilaisten konttitekniikoiden toimintaa sekä käyttötarkoitusta tutkittiin. Lisäksi konttienhallintaohjelmien rakennetta, ominaisuuksia ja käyttötarkoituksia analysoitiin. Työssä saatiin tärkeää tietoa Linux-pohjaisen konttiteknologian toiminnasta sekä sen mahdollistavista käyttöjärjestelmän ytimen ominaisuuksista.

LÄHTEET

[1] Datadog, 8 Surprising facts about real docker adoption, verkkosivu, päivitetty kesäkuu 2018. Saatavissa (19.10.2019): https://www.datadoghq.com/docker-adoption/

[2] J. Gomes, E. Bagnaschi, I. Campos, M. David, L. Alves, J. Martins, J. Pina, A. López-García, P. Orviz, Enabling rootless Linux Containers in multi-user environments: The udocker tool, Computer Physics Communications, vol. 232, Nov 2018, pp. 84–97. Saatavissa (19.10.2019): http://www.sciencedirect.com/science/article/pii/S0010465518302042

[3] A. Vartiainen, Konttiteknologia on merkittävimpiä tekniikan murroksia juuri nyt, Digiarjessa-blogi, julkaistu 10.3.2015. Saatavissa (19.10.2019): https://blog.digia.com/tech/2015/03/10/konttiteknologia-on-merkittavimpia-tekniikan-murroksia-juuri-nyt

[4] J. S. Hale, L. Li, C. N. Richardson and G. N. Wells, Containers for Portable, Productive, and Performant Scientific Computing, Computing in Science & Engineering, vol. 19, no. 6, 2017, pp. 40-50. Saatavissa (19.10.2019): https://ieeexplore.ieee.org/document/7933304

[5] D. Merkel, Docker: Lightweight Linux Containers for Consistent Development and Deployment, Linux Journal, päivitetty 19.5.2014. Saatavissa (19.10.2019): https://www.seltzer.com/margo/teaching/CS508.19/papers/merkel14.pdf

[6] P. Kasireddy, A Beginner-Friendly Introduction to Containers, VMs and Docker, freeCodeCamp, päivitetty (4.3.2019). Saatavissa (19.10.2019): https://www.freecodecamp.org/news/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b/

[7] OKD, Containers and Images, verkkosivu. Saatavissa (1.11.2019): https://docs.okd.io/latest/architecture/core_concepts/containers_and_images.html

[8] Red Hat, What’s a Linux container, verkkosivu, 2019. Saatavissa (19.10.2019): https://www.redhat.com/en/topics/containers/whats-a-linux-container

[9] J. Frazelle, Setting the Record Straight: containers vs. Zones vs. Jails vs. VMs, verkkosivu, julkaistu 28.3.2017. Saatavissa (19.10.2019): https://blog.jessfraz.com/post/containers-zones-jails-vms/

[10] S. J. Vaughan-Nichols, Open Container Initiative nails down container image distribution standard, ZDNet, verkkosivu, julkaistu 10.4.2018. saatavissa (31.12.2019): https://www.zdnet.com/article/open-container-initiative-nails-down-container-image-distribution-standard/

[11] M. Kerrisk, Namespaces, verkkosivu, päivitetty (8.2.2019). Saatavissa (23.11.2019): http://man7.org/linux/man-pages/man7/namespaces.7.html

[12] M. Ridwan, Separation Anxiety: A Tutorial for Isolating Your System with Linux Namespaces, Toptal, verkkosivu, 2010-2019. Saatavissa (23.11.2019):

https://www.toptal.com/linux/separation-anxiety-isolating-your-system-with-linux-namespaces

[13] G. Seltzer, Understanding cgroups, verkkosivu, julkaistu 23.11.2019. Saatavissa (23.11.2019): https://www.grant.pizza/blog/understanding-cgroups/

[14] Flockport, Discover LXC awesomeness!, verkkosivu, julkaistu 20.8.2014. Saatavissa (1.11.2019): https://archives.flockport.com/lxc-usecases/

[15] GNU Operating System, Chroot-invocation, verkkosivu. Saatavissa (1.11.2019): https://www.gnu.org/software/coreutils/manual/html_node/chroot-invocation.html#chroot-invocation

[16] R. Vieira, SELinux and AppArmor, verkkosivu. Saatavissa (1.11.2019): https://rodrigodelimavieira.com/selinux-and-apparmor-cju4vpgf80011f1s1bwkybmuv

[17] Á. Kovács, Comparison of different Linux containers, 2017 40th International Conference on Telecommunications and Signal Processing (TSP), Barcelona, 2017, pp. 47-51. Saatavissa (19.10.2019): https://ieeexplore.ieee.org/document/8075934

[18] LCX-kontit, Github, verkkosivu, 2019. Saatavissa (1.11.2019): https://github.com/lxc/lxc

[19] A. Cooke, Explore Docker vs. LXC through the lens of a three-tier app, TechTarget, verkkosivu, 2019, päivitetty (29.8.2019). Saatavissa (19.10.2019): https://searchitoperations.techtarget.com/tip/Explore-Docker-vs-LXC-through-the-lens-of-a-three-tier-app

[20] Linux containers, verkkosivu. Saatavissa (19.10.2019): https://linuxcontainers.org/lxc/

[21] Linux containers, security, verkkosivu. Saatavissa (1.11.2019): https://linuxcontainers.org/lxc/security/

[22] Red Hat, Chapter 8. Linux Capabilities and Seccomp, verkkosivu, 2019. Saatavissa (1.11.2019): https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html/container_security_guide/linux_capabilities_and_seccomp

[23] Eric Carter, 2018 Docker usage report., sysdig, verkkosivu, julkaistu 29.5.2018. Saatavissa (23.11.2019): https://sysdig.com/blog/2018-docker-usage-report/

[24] Docker, Best practices for writing Dockerfiles, verkkosivu, 2019. Saatavissa (23.11.2019): https://docs.docker.com/develop/develop-images/dockerfile_best-practices/

[25] M. Crosby, containerd Graduates Within the CNCF, Docker, verkkosivu, julkaistu 28.2.2019. Saatavissa (23.11.2019): https://www.docker.com/blog/containerd-graduates-within-the-cncf/

[26] I. Lewis, Container Runtimes Part 1: An Introduction to Container Runtimes, Ian Lewis, verkkosivu, julkaistu 6.12.2017. Saatavissa (23.11.2019): https://www.ianlewis.org/en/container-runtimes-part-1-introduction-container-r

[27] Docker, Docker security, verkkosivu, 2019. Saatavissa (23.11.2019): https://docs.docker.com/engine/security/security/

[28] Runc, Github, verkkosivu, 2019. Saatavissa (23.11.2019): https://github.com/opencontainers/runc

[29] Docker. Run the Docker daemon as a non-root user (Rootless mode), verkkosivu, 2019. Saatavissa (23.11.2019): https://docs.docker.com/engine/security/rootless/

[30] M. Crosby, containerd namespaces for Docker, Kubernetes, and beyond, Moby Blog, verkkosivu, julkaistu 7.11.2017. Saatavissa (23.11.2019): https://blog.mobyproject.org/containerd-namespaces-for-docker-kubernetes-and-beyond-d6c43f565084

[31] R. Radhakrishnan, Docker Architecture And Components, VMarena, verkkosivu, julkaistu 4.8.2018. Saatavissa (23.11.2019): https://vmarena.com/docker-architecture-and-components/

[32] N. Janetakis, Understanding How the Docker Daemon and Docker CLI Work Together, Nick Janetakis, verkkosivu, 2019, päivitetty 16.5.2017. Saatavissa (23.11.2019): https://nickjanetakis.com/blog/understanding-how-the-docker-daemon-and-docker-cli-work-together

[33] L. Jellema, First steps with Docker Checkpoint – to create and restore snapshots of running containers, Amis technology blog, verkkosivu, julkaistu 8.4.2018. Saatavissa (23.11.2019): https://technology.amis.nl/2018/04/08/first-steps-with-docker-checkpoint-to-create-and-restore-snapshots-of-running-containers/

[34] G. Kurtzer, V. Sochat, M. Bauer, Singularity: Scientific containers for mobility of compute, PLOS, julkaistu 11.5.2017. Saatavissa (23.11.2019): https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0177459

[35] Sylabs, Running Services, verkkosivu, 2017-2019. Saatavissa (23.11.2019): https://sylabs.io/guides/3.0/user-guide/running_services.html

[36] Singularity, About Singularity, verkkosivu, päivitetty 18.7.2018. Saatavissa (23.11.2019): https://singularity.lbl.gov/about

[37] C. Arango, R. Dernat, J. Sanabria, Performance evaluation of container-based virtualization for high performance computing environments, Universidad Industrial de Santander, julkaistu 16.7.2019. Saatavissa (23.11.2019): https://revistas.uis.edu.co/index.php/revistauisingenierias/article/view/10051

[38] Sylabs, Security in Singularity Containers, verkkosivu, 2017-2019. Saatavissa (23.11.2019): https://sylabs.io/guides/3.3/user-guide/security.html

[39] J. Layton, A Container for HPC, Linux New Media USA, verkkosivu, 2019. Saatavissa (23.11.2019): http://www.admin-magazine.com/HPC/Articles/Singularity-A-Container-for-HPC

[40] Singularity, Singularity and Docker, verkkosivu, päivitetty 18.7.2018. Saatavissa (23.11.2019): https://singularity.lbl.gov/docs-docker

[41] R. Brueckner, Singularity 3.1.0 brings in Full OCI Compliance, insideHPC, verkkosivu, julkaistu 27.2.2019. Saatavissa (23.11.2019): https://insidehpc.com/2019/02/singularity-3-1-0-brings-in-full-oci-compliance/

[42] Canonical Ltd, Getting Started, verkkosivu. Saatavissa (23.11.2019): https://docs.ubuntu.com/core/en/

[43] Linux.fi, Snap, verkkosivu, päivitetty 7.8.2019. Saatavissa (23.11.2019): https://www.linux.fi/wiki/Snap

[44] Canonical Ltd, Snap documentation, verkkosivu, 2019. Saatavissa (23.11.2019): https://snapcraft.io/docs

[45] Snapcraft, Security policy and sandboxing, verkkosivu, päivitetty 14.8.2019. Saatavissa (23.11.2019): https://forum.snapcraft.io/t/security-policy-and-sandboxing/554

[46] R. Feliciano, What Are Linux Snap Packages? Why Use Them?, Feliciano.Tech, verkkosivu, julkaistu 14.11.2018. Saatavissa (23.11.2019): https://www.feliciano.tech/blog/what-are-linux-snap-packages-why-use-them/

[47] Myriad-RF, Snap Packaging, verkkosivu. Saatavissa (23.11.2019): https://wiki.myriadrf.org/Snap_Packaging

[48] Canonical Ltd, Installing snapd, verkkosivu, 2019. Saatavissa (23.11.2019): https://snapcraft.io/docs/installing-snapd

[49] K. Falck, Kontit kuntoon, tivi, julkaistu 2.11.2017. Saatavissa (23.11.2019): https://www.tivi.fi/uutiset/kontit-kuntoon/d2beec7c-73fb-380b-86ce-ba111ee31361

[50] Docker, Docker Built-in Orchestration Ready for Production: Docker 1.12 Goes GA, verkkosivu, julkaistu 28.7.2016. Saatavissa (31.12.2019): https://www.docker.com/blog/docker-built-in-orchestration-ready-for-production-docker-1-12-goes-ga/

[51] V. Saraswat, Swarm Orchestration in Docker Enterprise Edition, Docker, verkkosivu, julkaistu 16.11.2017. Saatavissa (31.12.2019): https://www.docker.com/blog/swarm-orchestration-in-docker-enterprise-edition/

[52] Kubernetes, What is Kubernetes, verkkosivu, päivitetty 22.11.2019. Saatavissa (23.11.2019): https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

[53] Kubernetes, Github, verkkosivu, 2019. Saatavissa (23.11.2019): https://github.com/kubernetes/kubernetes

[54] M. Lukša, Kubernetes in Action, 1st ed. Manning Publications, 2018

[55] D. Rensin, Kubernetes Scheduling the Future at Cloud Scale, O'Reilly Media, First Edition, kesäkuu 2015. Saatavissa (28.12.2019): https://www.openshift.com/kubernetes-ebook/

[56] CRI-O, LIGHTWEIGHT CONTAINER RUNTIME FOR KUBERNETES, verkkosivu. Saatavissa (23.11.2019): https://cri-o.io/

[57] L. Liu, M. Brown, Containerd Brings More Container Runtime Options for Kubernetes, Kubernetes, verkkosivu, julkaistu 2.11.2017. Saatavissa (23.11.2019): https://kubernetes.io/blog/2017/11/containerd-container-runtime-options-kubernetes/

[58] DevelopIntelligence, What does Kubernetes actually do and why use it?, verkkosivu, 2013-2019. Saatavissa (23.11.2019): https://www.developintelligence.com/blog/2017/02/kubernetes-actually-use/

[59] Kontena, The Smart Dashboard for Kubernetes, verkkosivu, 2018. Saatavissa (23.11.2019): https://www.kontena.io/