Santeri Nihti

SOVELLUSKONTTIEN HALLINTA

Tietojenkäsittelyn koulutusohjelma

2018

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SOVELLUSKONTTIEN HALLINTA
Nihti, Santeri
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Helmikuu 2018
Ohjaaja: Grönholm, Jukka
Sivumäärä: 54
Liitteitä:

Asiasanat: ohjelmointiympäristö, sovelluskehittimet, sovellukset

---

Tässä opinnäytetyössä käsiteltiin sovelluskonttien hallinnoimista. Aiheesta tuotiin ensimmäisenä esille mitä sovelluskontit ovat yleisesti. Siinä todettiin, että sovelluskontit ovat nykyaikainen ratkaisu sovelluskehitysympäristöissä ja auttavat sovelluksien kehityksessä.

Tämän jälkeen tarkasteltiin sovelluskonttien historiaa ja todettiin, että sovelluskontit ovat olleet olemassa melkein 20 vuotta. Sovelluskonttien historian jälkeen paneuduttiin Docker sovelluskonttialustaan. Dockeria tarkasteltiin monesta eri näkökulmasta ja todettiin, että se on erittäin monipuolinen sovelluskonttialusta. Todettiin myös, että Docker toimii monessa eri käyttöjärjestelmässä rautatasolla ja monessa eri pilvipalvelujen tarjoamissa palveluissa.

Dockerin jälkeen tutkittiin sovelluskonttien hallintaohjelmia. Näitä hallintaohjelmia tarkasteltiin kolmelta eri valmistajalta. Ohjelmia tutkittiin niiden ominaisuuksien osalta ja miten ne soveltuisivat yritysympäristöihin. Ohjelmia tutkittiin myös siltä kannalta, miten helppoa niiden käyttö on ja kuinka käyttäjäystävällisiä ne ovat. Sovelluskonttihallintaohjelmien jälkeen tarkasteltiin RedHat-yhtiön julkaisemaa OpenShift -alustaa, joka yhdistää Dockerin ja Kuberneteksen. OpenShiftistä todettiin, että se on varteenotettava ratkaisu, joka yhdistää kahden suosituimman alustan ja hallintaohjelman ominaisuuksia. RedHat on lisännyt siihen myös omia ominaisuuksia. Openshift tarkastelun jälkeen päädyttiin tarkastelemaan Windowsin ja Dockerin suhdetta. Todettiin, että Windows tuli mukaan sovelluskontteihin vasta Windows 10 ja Windows Server 2016 -käyttöjärjestelmissä. Todettiin myös, että Windows on hyvin mukana Docker ja Kubernetes- toiminnassa, vaikka ovat juuri julkaisseet tuen näille omille alustoilleen ja ohjelmilleen.

Tämän jälkeen demonstroitiin WordPress -internetsivujen luonti työkalun asentamiseksi Kubernetes klusteri Azuren pilvipalvelussa. Tämä todettiin helpoksi asentaa ja dokumentaatio sopi hyvin asennukseen. Asennuksessa paneuduttiin myös käytettäviin komentoihin ja selitettiin mitä komennot tekivät. Asennus onnistui hyvin eikä suurempia ongelmia tullut. Asennuksen ympäristö todettiin myös hyväksi käytettävyydeltään ja joustavuudeltaan.

Application container management

Nihti, Santeri
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Data Processing
February 2018
Supervisor: Grönholm, Jukka
Number of pages:54
Appendices:

Keywords: application containers, developing, managing programs

---

In this thesis we exanimated managing application containers. Thesis subject was first examined what containers are in general. There we noticed that application containers are modern solution in developing applications environments. Application containers are also great help in developing new applications.

After this we viewed the history of application containers and stated that application containers were created almost 20 years ago. After history of application containers, we delved into Docker. We examined Docker from different point of views and came to conclusion that it is very variable platform. We also stated that Docker works great with other Operating Systems and on cloud services.

After Docker we examined managing programs for application containers. We viewed three application container managing programs from three different manufacturers. Managing programs were researched what properties they got and how would they fit in corporation environment. Programs were also investigated how users friendly they are. After container managing programs we looked to Red Hats OpenShift platform. We noticed that OpenShift is worthy solution that combines two great properties. OpenShift also added their own properties to the platform. After OpenShift we examined the relationship between Docker and Windows. There we noticed that Docker was introduced in Windows 10 and Windows server 2016. We also stated that Docker and Kubernetes are very well part of Windows Operating Systems although they are just released to Windows.

After that we delved into installing WordPress into Kubernetes cluster in Azure cloud service. The installation process was easy and documentation from installing WordPress to Kubernetes were also easy to follow. We delved into also to the commands what we used to install WordPress to Kubernetes. The install was very successful and there were no big problems on the process. Usability of the installing environment were great, and the flexibility was also very good.

SISÄLLYS

1 JOHDANTO

Sovelluskontit saivat alkunsa 2000-luvun alussa Linux-käyttöjärjestelmissä. Vuosien myötä uusia teknologioita kehitettiin sovelluskontteihin ja Linux-käyttöjärjestelmiin. Vuoden 2013 jälkeen sovelluskonttien käyttö alkoi kasvaa räjähdysmäisesti, kun Docker julkaistiin. Docker kuuluu tällä hetkellä suosituimpiin sovelluskonttiratkaisuihin. Dockerin suosio on kasvanut koko ajan ja yhä isompi osa yrityksistä on siirtymässä jonkinlaiseen ympäristöratkaisuun, jossa Docker on osallisena.

Sovelluskontit auttavat nykyajan IT-yrityksiä siirtymään kevyempään ympäristöön ja uudempaan tekniikkaan. Lisäksi ne auttavat yrityksiä pääsemään eroon vanhoista virtuaalipalvelinratkaisuista. Sovelluskontit tuovat ketteryyttä ja nopeutta ympäristöön, kun kaikki toimivat yhdenmukaisessa ympäristössä. Ne tukevat myös kaikkia käyttöjärjestelmiä ja pilvipalveluja, joten käyttäjille ei tarvitse luoda omia virtuaalikoneita ja eri käyttöjärjestelmien eriäväisyyksiä ei tule sovelluskonttien kanssa.

Tässä työssä tarkastellaan yleisesti, mitä sovelluskontit sisältävät ja tämän jälkeen niiden kehitystä ja historiaa. Sen jälkeen otetaan sovelluskonttialustoista Docker tarkempaan käsittelyyn. Dockeria tarkastellaan sen takia, että se voidaan luetella suosituimpiin sovelluskonttialustoihin sen monipuolisuuden ja soveltuvuuden takia. Opinnäytetyössä käsitellään Dockeriin liittyviä käsitteitä sekä käyttöjärjestelmiä ja pilvipalveluita, joilla Dockeria voidaan suorittaa. Dockerin jälkeen paneudutaan erilaisiin sovelluskonttien hallintaohjelmiin. Tarkastellaan Googlen toteuttamaa Kubernetes-hallintaohjelmaa, Dockerin omaa Swarm-ohjelmaa ja Mesos & Marathon-hallintasovelluksia. Näiden jälkeen toteutetaan demoasennus sovelluskontista. Demonstraatiossa käytetään Microsoft Azure-sovelluskonttipalvelua, johon asennetaan Kubernetes-sovelluskontin hallintaohjelma. Azureen asennetaan myös WordPress-ohjelma, joka on avoimen lähdekoodin internetsivujen luontityökalu.

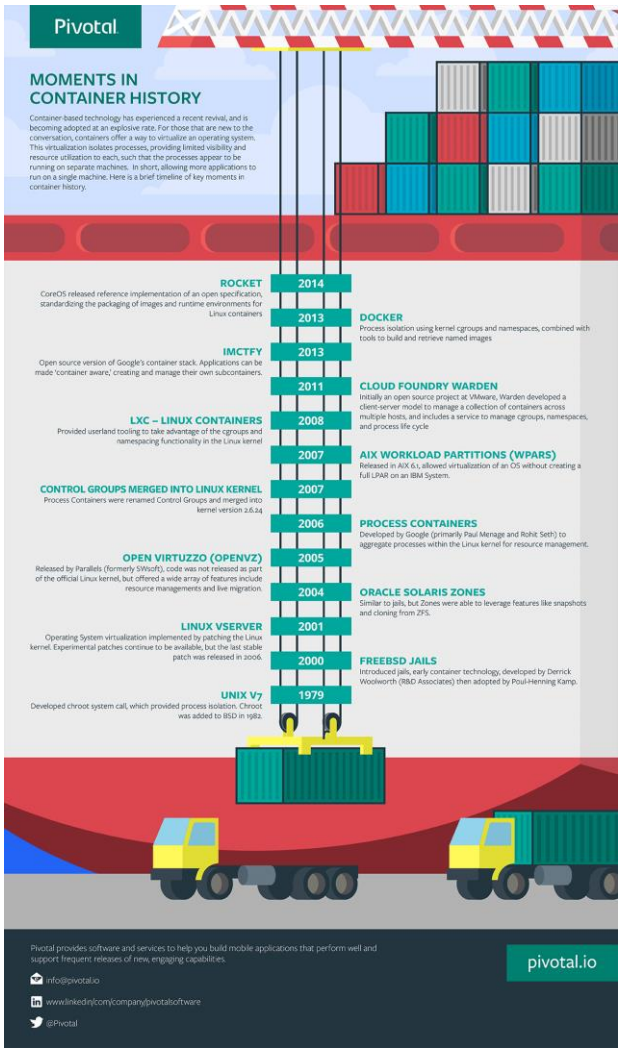
2 SOVELLUSKONTIT YLEISESTI

Sovelluskontit ovat ratkaisu silloin, kun halutaan saada ohjelma toimimaan turvallisesti eri järjestelmissä. Ohjelma voidaan siirtää esimerkiksi ohjelmoijan tietokoneelta testiympäristöihin, jotka voivat olla erilaisia ympäristöjä. Konttien avulla ohjelma voidaan ajaa eri alustoilla riippuen alustasta. Kontti sisältää ohjelmalle kaiken tarvittavan, jotta se voidaan ajaa testausympäristöstä tuotantoympäristöön. (https://www.cio.com/article/2924995/software/what-are-containers-and-why-do-you-need-them.html)

Sovelluskonttien pienuudesta johtuen niitä voidaan laittaa monta samalle järjestelmälle ja sovelluskontit keskustelevat suoraan järjestelmän kanssa ilman välikäsiä. Sovelluskonttien käyttö mahdollistaa myös mikropalvelun käytön konteissa. Mikropalvelut jakavat sovelluskontin pienempiin osiin, jotka keskustelevat keskenään. Tämä sovelluksen jako pienempiin osiin mahdollistaa ohjelmistokehittäjien paremman työnjaon. Ohjelmistokehittäjät voivat työstää ohjelman eri osia samaan aikaan, kunhan he eivät tee ohjelmaan mitään isompia muutoksia. Kaikkien sovelluskonttien hallintaan tarvitaan myös erillinen ohjelma. Yksi hyvä esimerkki on Kubernetes, jonka Google kehitti ja jolla saadaan sovelluskontit eri alustoille. Ohjelmalla voi myös laittaa sovelluskontteja varotoimenpiteenä varastoon ja käyttää näitä kontteja silloin, kun on tarvetta. Moni nykyinen sovellus ei toimi sovelluskonteilla uuden teknologian takia, mikä estää yrityksiä ottamasta sovelluskontteja käyttöön. (https://techcrunch.com/2016/10/16/wtf-is-a-container/)

Microsoft julkaisi syksyllä 2017 Windows-palvelimille ison päivityksen, joka mahdollistaa sovelluskonttien suorittamisen niillä. Ennen tätä isoa päivitystä Windowsin Nano Server oli tarkoitettu sovelluskonttien suorittamiseen. Tämän päivityksen vuoksi Nano Serverin rooli muissa palvelintehtävissä vanhentui. Windowsin Server Core on tämän päivityksen jälkeen Microsoftin suosima valinta sovelluskonttien levykuvien ylläpitämiseen. (https://redmondmag.com/articles/2017/09/14/windows-server-2016-version-1709.aspx)

3 SOVELLUSKONTTIEN HISTORIA



Kuva1

(https://uberflip.cdntwrk.com/files/aHViPTYzOTc1JmNtZD1pdGVtZWRpdG9yaW1hZ2UmZmlsZW5hbWU9aXRlbWVkaXRvcmltYWdlXzU4NDIxZjg5ODViNDMucG5nJnZlcnNpb249MDAwMCZzaWc9Y2UwMTMwZDkxODVmMzJiOWE3ZDZjMjU1YWFjY2U0NmU%253D)

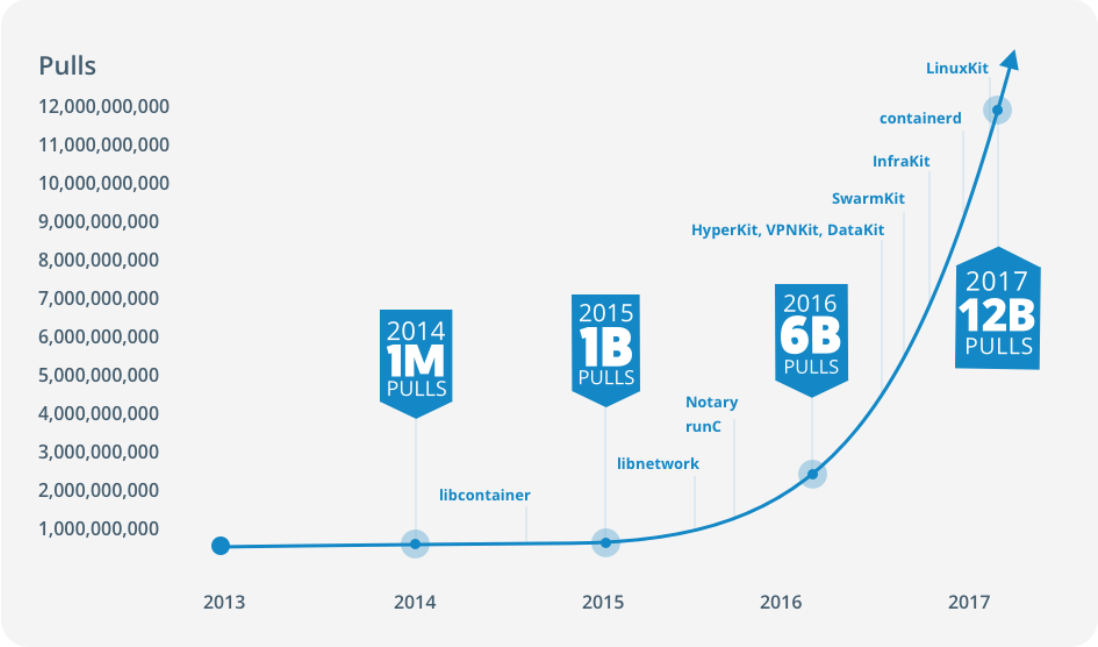
Sovelluskontit saivat alkunsa 2000-luvun alussa ja ilmestyivät Linux-käyttöjärjestelmiin ensimmäisenä. Sovelluskonteissa käytetty teknologia esiintyi ensimmäisen kerran unix-pohjaisessa FreeBSD-käyttöjärjestelmässä. FreeBSD-järjestelmä käytti ”Jail” tekniikkaa, luoden sovelluskontin ja virtuaalipalvelimen välimuodon. Jailit eli vankilat suunniteltiin turvallisiksi ympäristöiksi, joita pystyttiin jakamaan monelle käyttäjälle organisaation sisällä tai ulkopuolella. Vankilat ovat tiiviitä ympäristöjä, jotka luodaan FreeBSD -järjestelmään jakamalla käyttöjärjestelmä pienempiin osiin. Prosessit luovat muokatun tilan, josta on pääsy tiedostojärjestelmiin, verkkoratkaisuihin ja käyttäjät ovat virtualisoitu. Käyttäjät eivät pääse pois tästä vankilasta, mutta käyttäjäkokemus on samanlainen, kuin oikeassa käyttöjärjestelmässä. Käyttäjien ollessa näissä vankiloissa he eivät pääse käsiksi koko käyttöjärjestelmään. Tällä tavalla voidaan suojata käyttöjärjestelmä, jos ulkopuoliset käyttäjät käyttävät järjestelmää. (https://www.redhat.com/en/topics/containers/whats-a-linux-container)

Vuonna 2001 toteutus tästä eristetystä alustasta löysi tiensä Linux-käyttöjärjestelmiin. Jacques Gelinas oli yksi näistä henkilöistä, jotka toteuttivat tämän tekniikan Linux-käyttöjärjestelmille. Sen jälkeen, kun perusta tälle tekniikalle oli luotu Linux-järjestelmissä, tekniikkaan luotiin lisää palasia, joista nykyiset sovelluskontit koostuvat. (https://www.redhat.com/en/topics/containers/whats-a-linux-container)

Eri tekniikoita alettiin nopeasti yhdistelemään, jotta eristetystä toteutuksesta saataisiin todellinen. Kontrolliryhmät (cgroups) on Linuxin kernelissä oleva ominaisuus, joka säätelee ja rajoittaa resurssien käyttöä prosesseissa ja prosessiryhmissä. Käyttäjien muistitila mahdollistaa käyttäjille ja ryhmille eri oikeudet konttien sisällä kuin niiden ulkopuolella. Tämä loi lisäkerroksen turvallisuutta konttien ja käyttöjärjestelmän ympäristöön. Linux Container Project (LXC) lisäsi työkaluja, malleja, kirjastoja ja kielipaketteja, jotka lisäsivät käyttäjäkokemusta konttien käytössä. (Kuva 1)
(https://www.redhat.com/en/topics/containers/whats-a-linux-container)

4 DOCKER

Docker julkaistiin vuonna 2013 avoimen lähdekoodin projektina. Docker luottaa Linux kernelin ominaisuuksiin kuten nimitiloihin ja kontrolliryhmiin, jotta resurssit voidaan eristää ja ohjelma voidaan pakata kaikkine tarvikkeineen. Melkein heti ohjelmistokehittäjät huomasivat miten Dockerin uusi lähestyminen selvittäisi heidän isoimmat murheensa. Vuonna 2013, kuukausi julkaisun jälkeen, 10000 ohjelmistokehittäjää kokeili Dockeria. Vuoden sisällä isot yhtiöt kuten Red Hat ja Amazon lisäsivät julkisen tuen Dockerille. Kesäkuussa 2014 Docker ilmoitti Docker Engine 1.0 -version julkaisusta. Docker Engineä oli tuolloin ladattu 2.75 miljoonaa kertaa ja toukokuussa 2015 latauksia oli yli 100 miljoonaa. (Kuva 2)
(http://searchservervirtualization.techtarget.com/feature/A-brief-history-of-Docker-Containers-overnight-success)



(Kuva 2)
(https://cdn-images-1.medium.com/max/2000/0*g3sV5OiLtYMfEnzJ.)

4.1 Docker-käsitteet

Dockerin arkkitehtuurin käyttöönottoon ja hallitsemiseen liittyy monia eri käsitteitä. Käsitteiden hallitseminen auttaa ymmärtämään Dockerin arkkitehtuuria. Docker-käsitteet eivät muutu, vaikka niissä oleva ympäristö muuttuisi tai ne olisivat eri käyttöjärjestelmässä. Sovelluskonttien tekniikka ei ylety vain Dockeriin ja on paljon enemmän kuin pelkkä Docker. (Kuva 3)
(https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/)

4.1.1 Docker arkkitehtuuri

Container image on paketti, joka sisältää kaiken tarvittavan tiedon, jotta voidaan luoda sovelluskontti. Container image sisältää myös käyttöönoton ja konfiguraation, jotta sovelluskontti voidaan käynnistää. Useimmiten container image koostuu monesta tavallisesta imagesta. Nämä tavalliset imaget ovat päällekkäin toistensa päällä ja näin muodostavat kontin tiedostojärjestelmän.

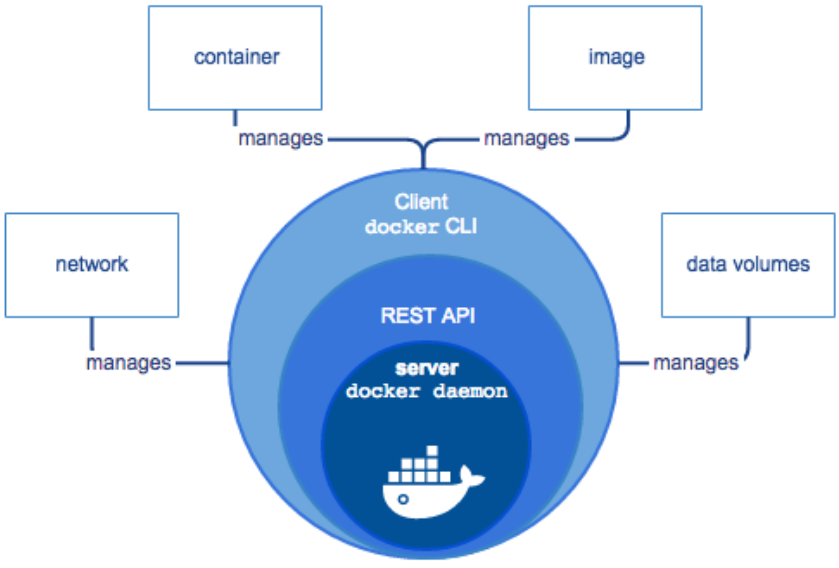
Container on instanssi Dockerin imagesta. Container edustaa käyttöä yhdelle sovellukselle, prosessille tai palvelulle. Container koostuu Docker-imagen sisällöstä, ympäristöstä, jossa container suoritetaan ja standardeista ohjeista koskien containeria. Ympäristöä skaalattaessa isommaksi luodaan instansseja containerin samasta imagesta. Voidaan myös tehdä erätyönä monta containeria samasta imagesta antaen eri parametrejä instansseille.

Repository on kokoelma Docker imageja, joissa on merkki tai leima, joka kertoo imagen versionumeron. Osa repositoreista koostuu monesta eri variaatiosta tietystä imagesta. Osa imageista voi olla isompia ja sisältävät esimerkkinä ohjelmiston kehitystyökalut. Jotkut imageista voivat olla myös kevyempiä ja sisältävät vain tiedostot imagen tiedoista. Yksi repository voi sisältää eri alustaratkaisuja, kuten Windows -magen ja Linux-imagen.

Registry on rekisteripalvelu, joka mahdollistaa pääsyn repositoryyn. Oletusrekiseri julkisille imageille on Docker Hub, jonka Docker omistaa.. Rekisteri sisältää usein repositoreja useilta eri ryhmiltä. Yrityksillä on usein yksityiset rekisterit levykuville, joita he voivat säilyttää ja hallinnoida.

Compose on komentorivityökalu ja YAML -tiedosto metadatan kanssa määrittelyyn ja multicontainer sovellusten ajamiseen. Voidaan määritellä sovellus, joka perustuu moneen levykuvaan joko yhdellä tai monella YAML -tiedostolla, joka voi ylikirjoittaa arvoja riippuen ympäristöstä. Erottuvuuden luomisen jälkeen voidaan julkaista kokonainen multicontainer-sovellus käyttämällä yhtä komentoa, joka luo containerin per levykuva Docker-isännälle.

Cluster on kokoelma avoimia Docker-isäntiä niin kuin ne olisivat yksi virtuaalinen Docker-isäntä, jotta sovellus voi skaalata itsensä moneen instanssiin. Sovellus voi levitä moneen isäntään clusterin sisällä. Docker-clustereita voi luoda Docker Swarm-ohjelmalla, Mesosphere OS-ohjelmalla sekä Kubernetes-ohjelmalla, jonka Google alkujaan loi.
(https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/container-docker-introduction/docker-terminology)



(Kuva 3)
(https://docs.docker.com/engine/images/engine-components-flow.png)

4.2 DOCKER VERSIOT

Docker tarjoaa käyttäjilleen kahta eri versiota Dockerista. Nämä ovat Docker Community Edition (CE) ja Docker Enterprise Edition (EE). (Kuva 4)
(https://www.docker.com/get-docker#/more_resources)

Platform	Docker CE x86_64	Docker CE ARM	Docker CE ARM64	Docker CE IBM Z (s390x)	Docker EE x86_64	Docker EE IBM Z (s390x)
CentOS	✓				✓	
Debian	✓	✓	✓			
Fedora	✓					
Microsoft Windows Server 2016					✓	
Oracle Linux					✓	
Red Hat Enterprise Linux					✓	✓
SUSE Linux Enterprise Server					✓	✓
Ubuntu	✓	✓	✓	✓	✓	✓

(Kuva 4)

4.2.1 Docker Community Edition

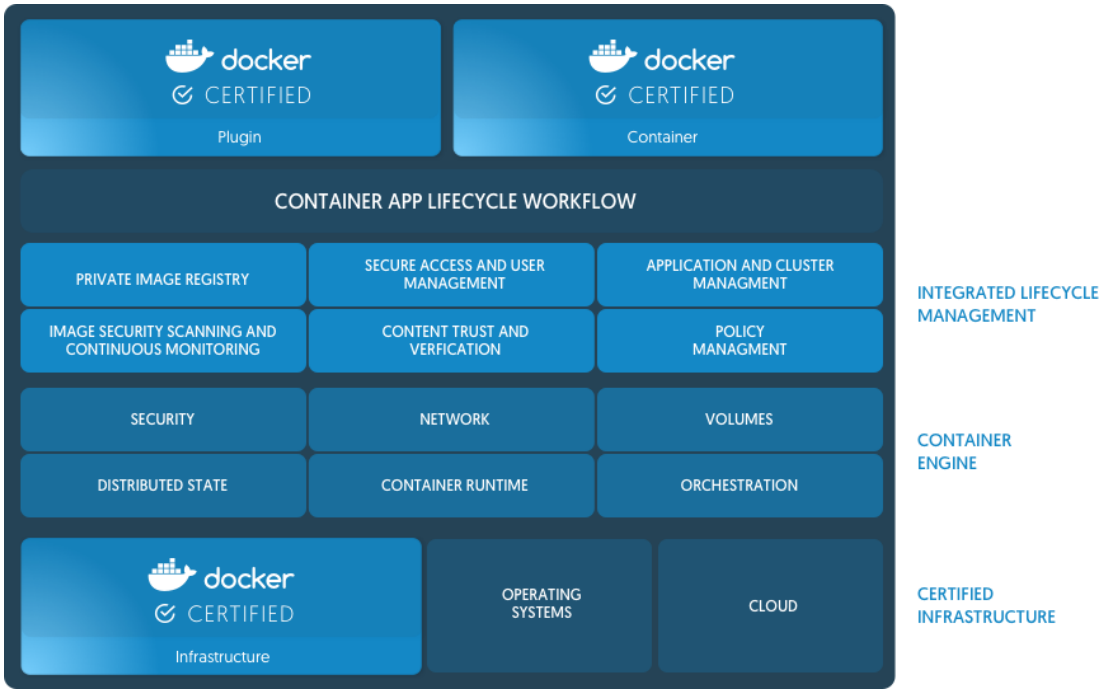
Docker Community Edition on ilmainen versio Dockerista, joka on suunnattu ohjelmistokehittäjille ja pienille ryhmille, jotka aloittelevat Dockerin käyttöä. Docker Community Edition on saatavilla monelle suositulle infrastruktuurialustalle, kuten tietokoneille, pilvipalveluihin ja vapaan lähdekoodin käyttöjärjestelmiin. Community Edition tarjoaa asennusohjelman nopeaa ja yksinkertaista asennusta varten, jotta voidaan aloittaa sovelluksen kehittäminen mahdollisimman pian. Community Edition on integroitu ja optimoitu kehittäjän infrastruktuuriin, jotta voidaan ylläpitää natiivin ohjelmiston kokemusta samalla, kun aloitetaan Dockerin käyttö. Voidaan rakentaa omia kontteja, jakaa kontti ryhmän kanssa ja automatisoida ohjelmistokehityksen kulku. (Kuva 5)
(https://www.docker.com/community-edition)

Docker Community-version ominaisuudet:

- Viimeisin Docker-versio integroiduilla työkaluilla sovelluskontin rakentamiseen, testaamiseen ja sovelluksien ajamiseen.
- Maksuton sovelluksen ylläpito viimeisimpään Docker versioon.
- Kuukausittainen Edge-version julkaisu ja joka neljännesvuosi julkaistaan vakaa versio Dockerista.
- Loputtomasti julkisia repositoreja ja yksi yksityinen repository.
- Automaattinen rakennus palveluna
- Levykuvien skannaaminen ja kokoaikainen haavoittuvuuksien etsintä palveluna.

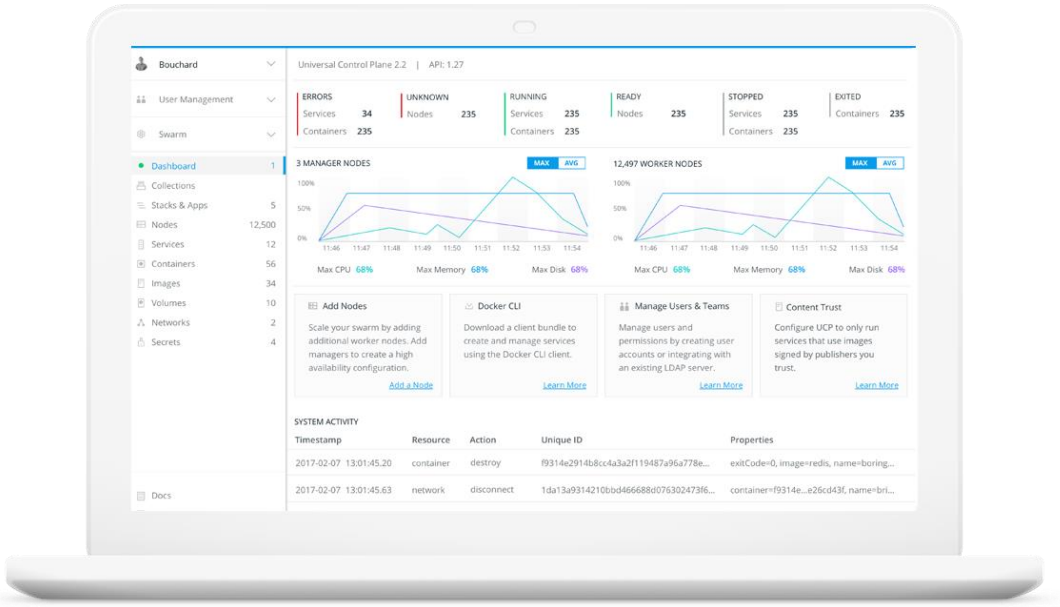
4.2.2 Docker Enterprise Edition

Docker Enterprise Editionia voisi kuvailla Container as a Service -palveluna. Se on palvelu, jossa yritykset tarjoavat valmiita sovelluskontteja pilvipalveluissa. Docker Enterprise Editionin avulla yritys voi modernisoida sovelluksiaan, infrastruktuuria ja toiminnan mallia tuomalla esiin olemassa olevia IT-ratkaisuja ja samalla integroimalla uutta teknologiaa.
(https://www.docker.com/enterprise-edition)



(Kuva 5)
(https://www.docker.com/sites/default/files/Container-App-Lifecycle-4.png)

Docker Enterprise Edition tarjoaa integroituja, testattuja ja sertifioituja alustoja, jotta sovellukset toimivat Linux, Windows ja pilvipalvelualustoilla. Enterprise Edition on tiukasti integroitu alusta, joka tarjoaa natiivin, helpon ja optimoidun Docker-ympäristön. Containerit ja liitännäiset ovat eksklusiivisia Docker Enterprise Editionille yhteistyökykyisellä tuella Dockerilta ja sertifioiduilta teknologiapartnereilta. (Kuva 6)
(https://www.docker.com/enterprise-edition)

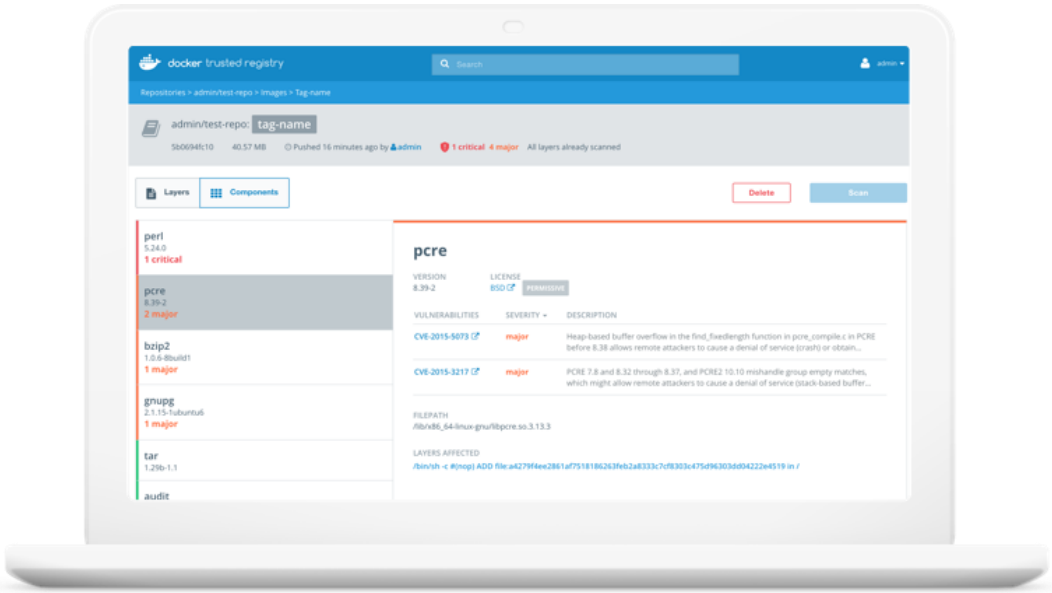


(Kuva 6)
(https://www.docker.com/sites/default/files/Enterprise_management_EE.png)

Dockerin datakeskus on osa Docker Enterprise Editionia, joka tarjoaa integroidun kontin hallinnoimisen ja turvallisuutta ohjelman kehityksestä tuotantoon. Heti valmiit yritysvalmiudet kuten multiarkkitehtuurien järjestäminen, turvallinen sovellusketju ja infrastruktuurin riippumattomuus antavat IT-ryhmille mahdollisuuden hallinnoida ja turvata sovelluskontit ilman, että heidän pitää puuttua sovelluskehittäjien työhön.

Avoimet käyttöliittymät mahdollistavat helpon integraation valmiina oleviin järjestelmiin ja joustavuuden tukea kaiken pituisia prosesseja. Enterprise Edition tarjoaa integroidun hallinnoimisen kaikkien sovelluksien resursseihin yhdestä ylläpitäjän näkymästä. Monen haltijan järjestelmä RBAC ja LDAP/AD integraatiolla. Docker

tarjoaa integroidun turvallisen käyttöliittymän, joka tarjoaa vahvemman perusturvallisuuden sillä joustavuudella, että voit vaihtaa konfiguraatioita ja standardisoida käyttöliittymiä.
(Kuva 7)



(Kuva 7)
(https://www.docker.com/sites/default/files/sessc_0_1.png)

Docker Enterprise Edition sisältää kolme tasoa. Perustaso sisältää Docker alustan sertifioiduille infrastruktuureille Docker inc tuella ja sertifioiduilla konteilla ja lisäosilla. Tavallinen taso lisää kehittyneen levykuvan ja kontin hallinnoimisen. LDAP ja AD käyttäjien integrointi tulee tavallisen tason mukana. Edistynyt taso tarjoaa Dockerin turvallisuusskannauksen ja kokoaikaisen riskien monitoroimisen.
(https://docs.docker.com/enterprise/#docker-ee-feature-tiers)

Docker Enterprise Edition julkaistaan vuoden joka neljännes. Jokaista julkaistua Docker Enterprise Editionia tuetaan ja ylläpidetään vuosi julkaisusta eteenpäin. Enterprise Edition saa myös kriittisiä korjauksia ja turvallisuuspäivityksiä. (https://docs.docker.com/enterprise/#docker-ee-feature-tiers)

5 DOCKER ERI ALUSTOILLA

Docker on saatavilla nykypäivänä melkein jokaiselle käyttöjärjestelmälle. Docker voidaan asentaa Windows-ympäristöön (työpöytä ja palvelin) sekä Linux-ympäristöön, jossa se voidaan asentaa joko työpöytä- tai palvelinalustalle. Docker voidaan asentaa myös OS X - käyttöjärjestelmälle, joten jokainen ohjelmistonkehittäjä on otettu huomioon.
(https://docs.docker.com/install/)

5.1 Docker for Windows

Dockeria voidaan käyttää Windowsin työpöytä- ja palvelinjulkaisuilla. Työpöytäversiona Dockerista suositellaan Windows 10 Pro, Enterprise tai Education versioita. Vanhemmille Windows-versioille on myös ratkaisu, jolla voidaan käyttää Dockeria: Docker Toolbox, joka käyttää Oraclen Virtual box-ohjelmaa Hyper-V ratkaisun sijasta.

Asentuakseen ilman Oracle Virtual box-ratkaisua Docker tarvitsee Hyper-V ominaisuuden ja virtualisointi pitää olla sallittu järjestelmässä. Docker for Windows versiolla voidaan myös asentaa linux container image ja täten toteuttaa linux containerien kehittäminen ja hallinta Windows-asemalta.

Dockerin ohjaaminen Windows-versioilla tapahtuu PowerShellin tai Docker komentokehotteen avulla. Dockerin asetusten määrittäminen Windows-ympäristöissä on helppoa, sillä asetusten vaihtaminen on tehty hyvin käyttäjäystävälliseksi. (https://docs.docker.com/docker-for-windows/)

5.2 Docker Linux

Docker on saatavilla monelle Linux-jakelulle eivätkä ne eroa paljoa toisistaan. Dockerin voi asentaa Linuxille muutamalla eri tavalla. Tämä poikkeaa Windows-versiosta sillä, että Windows versiossa on vain yksi asennusohjelma. Useimmat käyttäjät asentavat Dockerin Linuxiin aloittamalla asennuksen Dockerin repositoreista. Dockerin asentaminen repositorien kautta on erittäin helppoa, sillä aluksi asennetaan repository Dockerille ja sen jälkeen asennetaan itse Docker järjestelmään.

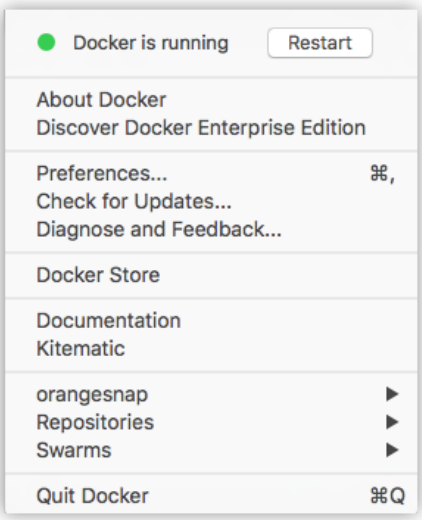
Asennuksen aloittaminen repositoreista on yleistä asennuksen ja päivitysten helppouden vuoksi. Jotkut Dockerin käyttäjät haluavat asentaa Dockerin DEB-paketin kautta. DEB-paketti on verrattavissa Windowsin exe-pakettiin. Yleensä DEB-paketin kautta asentavat henkilöt haluavat hallinnoida päivityksiä kokonaan manuaalisesti. Päivitettäessä Dockeria pitää aina ladata uusi DEB-paketti Dockerin sivuilta riippuen verisosta, johon Docker halutaan päivittää. Tätä tapaa suositellaan järjestelmiin, jotka eivät ole yhteydessä internetiin.
(https://docs.docker.com/install/linux/docker-ce/ubuntu/#extra-steps-for-aufs)

Docker voidaan myös asentaa Linux-järjestelmiin automaattisten skriptien kautta. Näiden skriptien asennusta ei suositella tuotantoympäristöön. Testaus- ja kehitysympäristöön convience-skriptien käyttö sopii hyvin, mikäli käyttäjät ovat ammattitaitoisia ja kiinnostuneita aiheesta.
(https://docs.docker.com/install/linux/docker-ce/ubuntu/#set-up-the-repository)

5.3 Docker Mac OS X

Docker on saatavilla Mac-tietokoneille kahdella tavalla. Ensimmäisenä on Windows-ympäristöstäkin tuttu Docker Toolbox. Dockerin käyttö tämän Toolboxin kautta onnistuu ihan normaalisti ja toimii samalla tavalla, kuin Windows-ympäristössäkin. Mac-ympäristössä käytetään myös todella pientä Linux-jakelua, joka on nimeltään boot2docker. Tällä jakelulla saadaan Docker toimimaan Mac- ympäristössä nopeasti. (https://docs.docker.com/docker-for-mac/docker-toolbox/#the-docker-toolbox-environment)

Docker For Mac on hieman erilainen kuin Docker Toolbox, joka toimii OSX -käyttöjärjestelmässä. Docker for Mac -versiossa käytetään HyperKit-virtualisointia toisin kuin Oraclen Virtual Boxissa. HyperKit on OS X järjestelmiin tarkoitettu kevyt virtualisointiratkaisu, joka on rakennettu macOS 10.10 ja sitä myöhäisempiin versioihin. Docker Toolboxilla luodut containerit ja imaget eivät vaikuta Docker for Mac asennukseen. Sen sijaan Docker for Mac kysyy asennuksen jälkeen, kopioidaanko Docker Toolboxilla tehdyt containerit ja imaget Docker for Mac ympäristöön. Vanhat containerit ja imaget eivät häviä. Dockerin valikot OS X järjestelmässä ovat pitkälti samanlaisia kuin Windows -ympäristössä. Valikot ovat käyttäjäystävällisiä eikä käyttäjän tarvitse käyttää terminaalia, vaikka macOS pohjautuu UNIX-järjestelmään. (Kuva 8)
(https://docs.docker.com/docker-for-mac/#preferences)

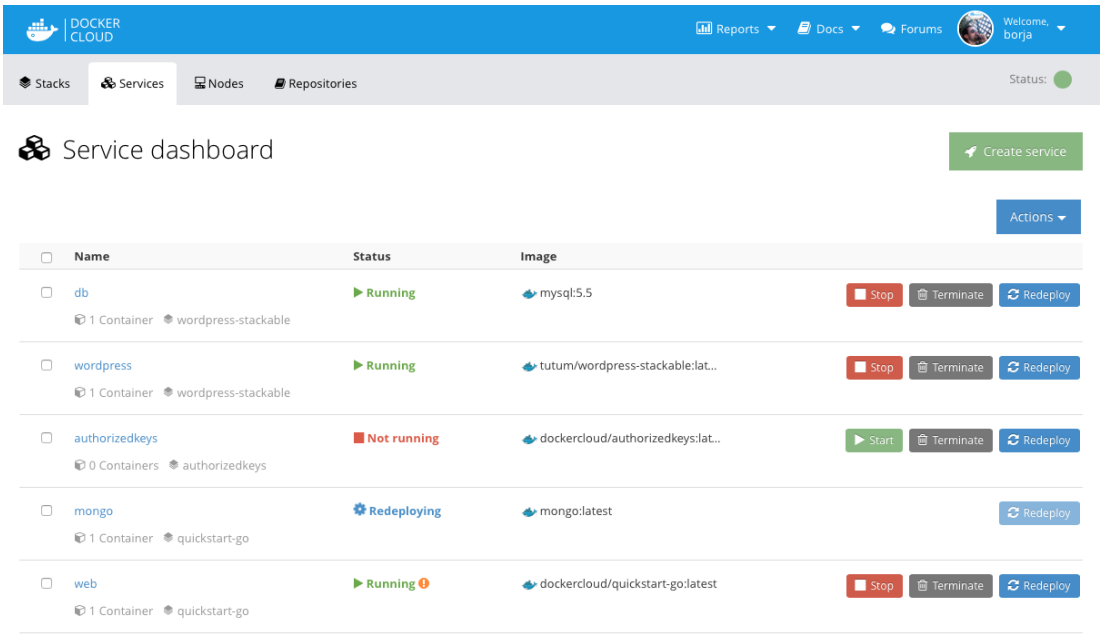


(Kuva 8) (https://docs.docker.com/docker-for-mac/images/menu-prefs-selected.png)

Docker for Mac tarjoaa mahdollisuuden monen arkkitehtuurin tukeen. Tämän mahdollisuuden vuoksi voit ajaa containereita, jotka ovat eri Linux -arkkitehtuurilla toteutettuja. Tämä monen arkkitehtuurin käyttäminen Dockerin Mac-versiossa onnistuu ilman mitään erillistä konfiguraatiota.
(https://docs.docker.com/docker-for-mac/multi-arch/)

5.4 Docker pilvipalveluissa

Docker tukee monia eri pilvipalveluiden tarjoajia, kuten Google Cloud Platform, Amazon Web Services ja Microsoft Azure. Pilvipalvelussa Docker säästää rahaa laskemalla tarvittavan sovelluksen käyttämiseen käytettäviä resursseja. Tämä pienentää palvelimista aiheutuvia kustannuksia ja vähentää henkilökuntatarvetta. Yritys voi pitää ryhmät pienenä ja tiiviinä. Dockerin hallinnoiminen yhdessä pilvipalvelussa on helppoa, koska se tukee monia eri alustoja, käyttöjärjestelmiä ja ympäristöjä. Sovelluksen luominen, testaaminen ja tuotantoympäristö voi olla saman pilvipalvelun tarjoajan alaisuudessa. Säästöä tulee myös, koska ei tarvitse omistaa omaa palvelinsalia ja ylläpitää sitä.
(https://apiumhub.com/tech-blog-barcelona/top-benefits-using-docker/)
Dockerilla on myös oma Docker Cloud -palvelu, jossa voidaan hallinnoida tilejä ja Docker containeneita ja muuta Dockeriin liittyvää. Docker Cloudin avulla voidaan esimerkiksi hoitaa Docker Swarmia pilvipalvelusta käsin. Voidaan aloittaa uusi Docker Swarm, rekisteröidä jo olemassa oleva tai tarjota Docker Swarmia pilvipalvelun tarjoajalle. Tämä Dockerin Swarm toiminto on vasta betavaiheessa, joten sen käytössä voi olla vielä ongelmia, mutta sitä voidaan käyttää. Infrastruktuuria voidaan hallinnoida Docker Cloudin kautta helpolla käyttöliittymällä. (Kuva 9)
(https://docs.docker.com/docker-cloud/)



(Kuva 9)

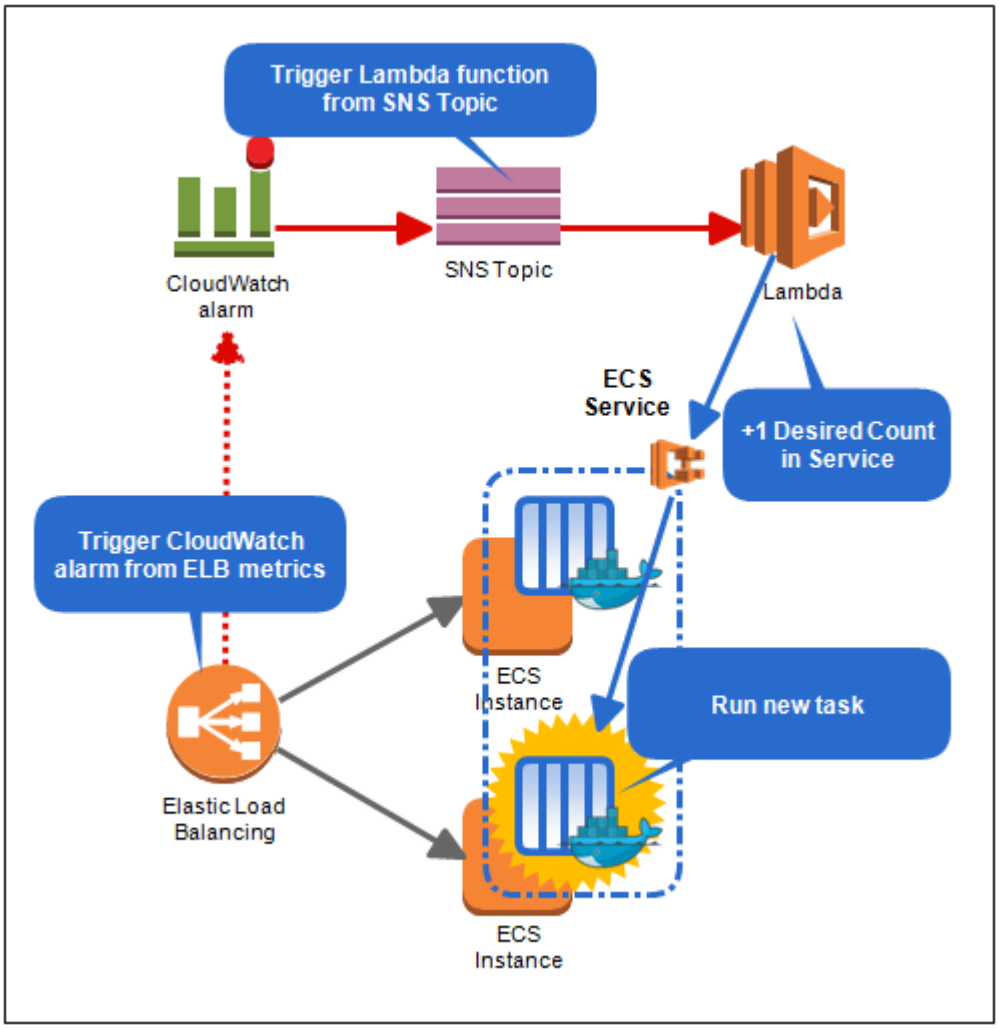
(https://blog.docker.com/wp-content/uploads/docker_cloud_dashboard.png)

5.4.1 Docker Amazon Web Services

Amazonin pilvipalveluun tarkoitettu Docker tarjoaa käyttäjälle natiivin käyttökokemuksen Dockeriin ilman tarvittavia API:ja. Amazonin Docker -versio esilataa kaikki tarvittavat ja suositellut infrastruktuurit. Tämä mahdollistaa sen, että käyttäjien ei tarvitse käyttää omia instanssejaan, turvallisuusryhmiään sekä kuormantasauksiaan silloin, kun he käyttävät Amazonin pilvipalveluun tarkoitettua Dockeria. (Kuva 10)

Amazonin Docker -versioon käytetään muokattua Linux -jakelua, joka on tarkoitettu toimimaan Amazonin ympäristössä. Linux -jakelun tarkoitus on tarjota käyttäjälle paras mahdollinen Dockerin käyttökokemus. Linux -jakelu on muokattu siten, että kaikki sen osat olisivat parhaita Dockerin suorittamiseen. Amazonin pilvipalveluun saa asennettua sekä Community Editionin että Enterprise Editionin. Community Editionista on saatavilla tutut Stable -julkaisu ja Edge -julkaisu, jotka noudattavat samoja määritelmiä kuin Windows -ympäristössä. Amazonin pilvipalveluun on myös saatavilla testikanava, jossa käyttäjät voivat testata uusia julkaisuja ennen kuin ne julkaistaan.
(https://docs.docker.com/docker-for-aws/why/)

Dockeri voidaan ottaa käyttöön kahdella tapaa Amazonin pilvipalvelussa: jo olemassa olevaan virtuaaliseen yksityiseen pilveen tai luomalla oma virtuaalinen yksityispilvi Dockerin avulla. Dockeri suosittelee käyttöönottoa luomalla oma virtuaalinen pilvi, koska tällöin ympäristö voidaan optimoida Dockerin käyttöön eikä olemassa olevaa ympäristöä tarvitse mennä muokkaamaan, mikä voi olla aikaa vievää ja hankalaa.
(https://docs.docker.com/docker-for-aws/#deployment-options)



(Kuva 10)

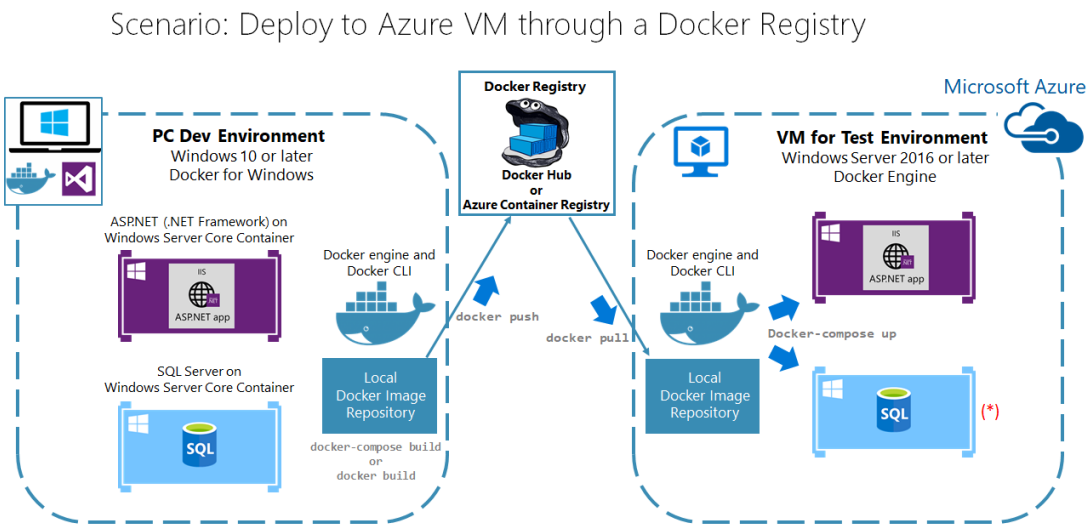
https://s3.amazonaws.com/chrisb/CW-ECS-diagram.png

5.4.2 Docker Azure

Docker tarjoaa Microsoftin Azure-pilvipalveluun samanlaisia toimintoja sekä saman käyttökokemuksen helppouden, joita se tarjoaa Amazonin pilvipalveluun. Mitään ylimääräisiä API:ja ei tarvita Azuren pilvipalvelussa.
(https://docs.docker.com/docker-for-azure/)

Sekä Community Edition ja Enterprise Edition ovat saatavilla tähän pilvipalveluun. Docker tarjoaa myös Community Editioniin molemmat versiot Dockerista. Stable Channel, joka on vakaa, joka neljäs kuukausi päivitettävä versio Dockerista ja Edge Channeliin päivitetään uusimmat päivitykset joka kuukausi.
(https://docs.docker.com/docker-for-azure/#docker-community-edition-ce-for-azure)

Dockerin asennus Azure -ympäristöön eroaa aika paljon Amazonin asennuksesta. Azure -ympäristössä käyttäjän tarvitsee syöttää komentoja enemmän ja asennus voi tuntua monimutkaisemmalta. Docker tarvitsee käyttöoikeuden Azure -tiliin järjestelmävalvojan valtuuksilla. Docker tarvitsee myös SSH-avaimen, jota käyttäjä voi käyttää asennuksen jälkeen hallinnoidakseen Dockeria. Docker voidaan asentaa Azure -ympäristöön kahdella tavalla: joko käyttämällä selainpohjaista Azure-portaalia tai käyttämällä Azuren komentokehotekäyttöliittymää. Molemmat näistä asennustavoista käyttävät samoja konfiguraatioasetuksia. (Kuva 11)
(https://docs.docker.com/docker-for-azure/#configuration)



(Kuva 11)

https://user-images.githubusercontent.com/1712635/30402804-d62632a2-9893-11e7-817a-f9f616cdf380.png

5.4.3 Docker IBM Cloud

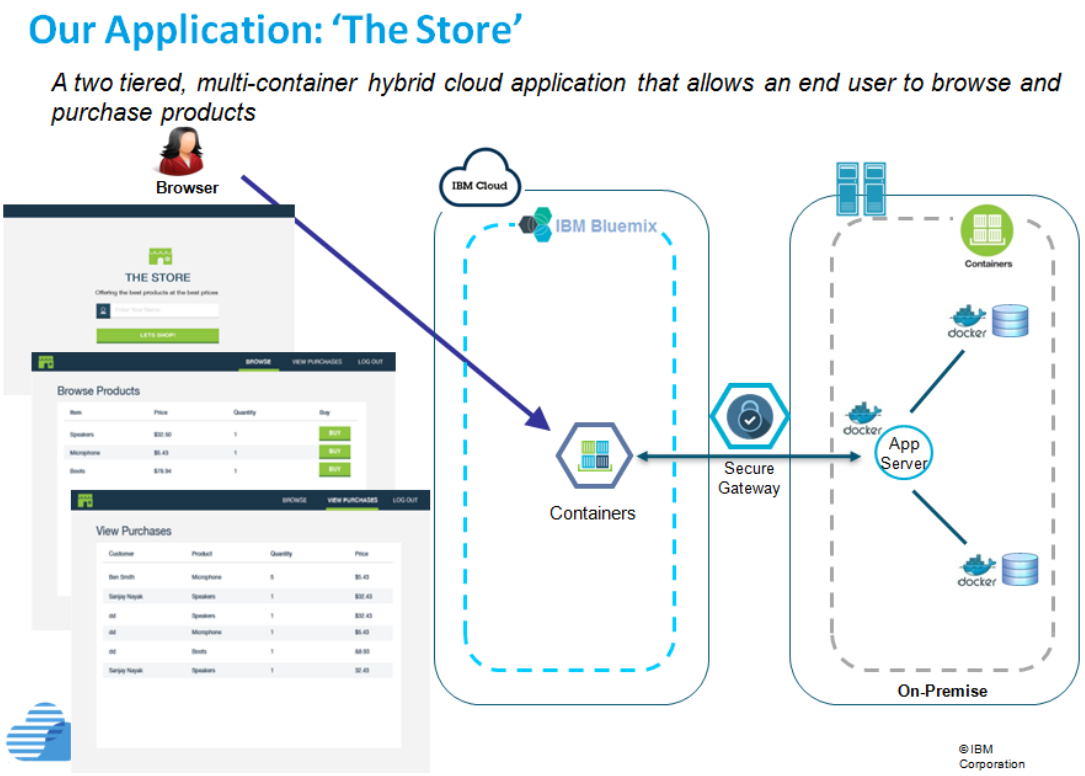
Docker IBM-pilvipalvelu on vielä betavaiheessa (2016), eikä sitä ole vielä kokonaan julkaistu. Docker tarjoaa vain Enterprise Editionia IBM:n pilvipalveluun. IBM -versioon tarjotaan melko samanlaiset palvelut kuin Amazonin ja Azuren palveluihin. Tietysti nämä palvelut on optimoitu IBM-pilvipalveluun tarjoten parhaan mahdollisen käyttäjäkokemuksen. Docker käyttää IBM:n pilvipalvelussa swarm clustereita. (Kuva 12)
(https://docs.docker.com/docker-for-ibm-cloud/why/#native-to-docker)

Dockerin asennus IBM-pilvipalveluun vaatii enemmän tekemistä kuin Amazonin tai Azuren palveluihin. Tämä johtuu siitä, että Docker on vielä betavaiheessa. Docker tarvitsee käyttöoikeudet IBM-tileihin, joita voi joutua linkittämään Dockeriin ja ehkä päivittämään. Käyttäjän täytyy tarjota Dockerille tietyt resurssit pilvipalvelussa. Docker tarvitsee tiedostotilaa ja tämä tiedosto tulisi olla ”block” -muodossa eli kiintolevy on lohkottu osiin. Käyttäjän tarvitsee myös varustautua kuorman tasaukseen, SSH avaimiin, aliverkon IP-osoitteisiin, virtuaalisen serverin laitteisiin ja virtuaalisiin lähiverkkkoihin. (Kuva 13)
(https://docs.docker.com/docker-for-ibm-cloud/#docker-enterprise-edition-ee-for-ibm-cloud)

Permissions set	Description	Required permissions
Devices	Connect to and configure your VSI, load balancers, and firewalls.	• View hardware detail • View virtual server details • Hardware firewall • Software firewall manage • Manage load balancers • Manage device monitoring • Reboot server and view IPMI system information • Issue OS Reloads and initial rescue kernel[1] • Manage port control
Network	Provision, connect, and expose IP addresses.	• Add compute with public network port[2] • View bandwidth statistics • Add IP addresses • Manage email delivery service • Manage network VLAN spanning[1] • Manage security groups[2]
Services	Provision and manage services such as CDN, DNS records, SSH keys, NFS storage volumes.	• View CDN bandwidth statistics • Vulnerability scanning • Manage CDN account[1] • Manage CDN file transfers[1] • View licenses • Manage DNS, reverse DNS, and WHOIS • Antivirus/spyware • Host IDS • Manage SSH keys[1] • Manage storage[1] • View Certificates (SSL)[1] • Manage Certificates (SSL)[1]
Account	General settings to provision or remove services and instances.	• View account summary • Manage notification subscribers • Add/upgrade cloud instances[1] • Cancel server[1] • Cancel services[1] • Add server[1] • Add/upgrade services[1]

(Kuva 12)

https://docs.docker.com/docker-for-ibm-cloud/faqs/#what-ibm-cloud-infrastructure-permissions-do-i-need



(Kuva 13)
https://www.ibm.com/blogs/bluemix/wp-content/uploads/2015/06/StoreApplication.png

6 SOVELLUSKONTTIEN HALLINTAOHJELMAT

Sovelluskonttien hallintaan on saatavilla monta ohjelmaa, joiden tarve riippuu käyttäjästä. Suosituimmat sovelluskonttien hallintaohjelmat ovat Googlen kehittämä Kubernes, Mesos & Marathon ja Dockerin oma hallintajärjestelmä Swarm. Näitä hallintajärjestelmiä voidaan kutsua myös klusterointiohjelmiksi, koska monien konttien yhteistä ryhmää kutsutaan klusteriksi. (Tivi 11/2017)

Sovelluskonttien yhdeksi ongelmaksi voi sanoa pysyväismuistin riittävyyden niiden käynnistyessä ja sammuessa. Käyttäjän tarvitsee luoda aina käynnistyvälle sovelluskontille riittävä pysyväismuisti ja tiedot tallennetaan palvelimelle, josta sovelluskontti käynnistyy uudestaan. Ympäristön ylläpitäjän pitää huolehtia siitä, että sovelluskontit käynnistyvät samalta palvelimelta, jossa niiden data on tallennettu. (Tivi 11/2017)

Sovelluskontin hallintaohjelman valintaa tulisi miettiä siltä kannalta mikä on yrityksen palvelimien lukumäärä. Hallintaohjelmilla voidaan ohjata tuhansia palvelimia ja yrityksen kannattaa tutkia eri hallintaohjelmia ja paneutua niiden käyttöön ja etsiä heidän ympäristöönsä sopivin hallintaohjelma. (Tivi 11/2017)

6.1 Kubernetes

Kubernetes on Googlen kehittämä projekti, jonka he ulkoistivat vuonna 2014. Kubernetes on avoimen lähdekoodin alusta kontitettujen sovellusten ja palveluiden hallinnoimiseen. Kubernetes on noussut nopeasti yhdeksi käytetyimmäksi hallintaohjelmaksi ja sen ekosysteemi kasvaa valtavalla vauhdilla.

Kuberneteksessa on monia ominaisuuksia, joista esimerkkeinä konttien alusta, mikropalveluiden alusta ja kannettava pilvipalveluiden alusta. Kubernetes tarjoaa myös kontteihin keskitetyn hallintaympäristön. Kubernetes järjestää palvelimien laskennan, verkkotyöskentelyn, tallennuspaikan infrastruktuurin ja kuorman tasauksen.

Kubernetes rakennettiin alustaksi, jotta sen ekosysteemi ja siihen saatavat komponentit voisivat laajentua. Tämä helpottaa ohjelmien lähettämistä, skaalautumista ja hallinnoimista. Sitä voisi kuvitella PaaS -alustaksi, vaikka se ei sitä ole. Kubernetes toimii sovelluskonttien tasolla sen sijaan, että se toimisi laitteistotasolla. Vaikka Kubernetes ei toimi laitteistotasolla se tarjoaa silti skaalautumisen, kuorman tasauksen, lokitiedot ja monitoroimisen. Se ei kuitenkaan ole monoliittinen ja nämä ratkaisut ovat valinnaisia. Kubernetes tarjoaa yrityksille rakennuspalikat heidän ympäristöönsä säästäen käyttäjien valintoja sekä joustavuuden siellä missä sitä tarvitaan.
https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Kubernetekseen on saatavilla monia eri lisäosia ja komponentteja, jotka helpottavat ympäristön hallinnointia ja monitorointia. Isoin komponentti sen komponenteista on Master components, joka muodostaa Kuberneteksen ohjauspaneelin. Master components hoitaa Kuberneteksen globaalit valinnat klusterissa kuten esimerkiksi klustereiden aikataulutuksen. Se myös havaitsee klusterin tapahtumia ja vastaa niihin. Master components voidaan ajaa millä koneella tahansa klusterissa. Asioiden yksinkertaistamiseksi voidaan Kubernetekseen tehdä skriptejä, jotka käynnistävät komponentit samalle laitteelle ja suositeltavaa olisi, ettei käyttäjien sovelluskontteja ajettaisi tällä laitteella. Master components koostuu kaikista komponenteista, joita käyttäjä on ottanut käyttöön. Isoimmassa osassa ovat controller-komponentit, jotka ohjaavat Kuberneteksen toimia. Kuberneteksessa on kahdenlaisia controller-komponentteja: kube-controller-manager ja cloud-controller-manager.

Loogisesti jokainen ohjain on eri prosessi, mutta monimutkaisuuden vähentämiseksi kaikki on yhdistetty yhteen binääriin ja ovat yksi ja sama prosessi. Kontrolleri sisältää Nodejen kontrollerin, monistuskontrollerin, päätepisteiden kontrollerin sekä käyttäjien ja tokenien kontrollerin. Node -kontrolleri on vastuussa siitä, jos nodeja kaatuu. Monistuskontrolleri ylläpitää saman määrän podeja ja monistuskontrolleriobjekteja. Päätepistekontrolleri täyttää päätepisteiden objektit. Käyttäjä & tokenkontrolleri luo oletuskäyttäjiä ja API:lle oikeustokeneja uusille nimitiloille.

Pilvipalvelujen kontrolleriohjain voi suorittaa kontrolleja, jotka keskustelevat pilvipalvelujen tarjoajien kanssa. Pilvipalvelujen kontrolleri pitää poistaa, jos ympäristössä aiotaan käyttää kube -kontrolleriohjainta. Pilvipalvelujen kontrollerien ohjaimessa on eriäväisyyksiä kuberneteksen oman kontrollerin kanssa, jota ei käytetä pilvipalveluissa. Pilvipalvelujen kontrollerihallintaohjain sisältää myös nodeohjaimen, reittiohjaimen, palveluohjaimen ja volumekontrollerin. Nodeohjain antaa pilvipalvelulle päätännän, jos node poistetaan tai jos se ei vastaa. Reittikontrolleri asettaa reitit pilvipalvelun infrastruktuuriin. Palveluohjain on tarkoitettu pilvipalvelujen kuormantasaajien luomiseen, päivittämiseen ja poistamiseen. Volumekontrolleri liittyy levytilaan. Kontrollerin avulla luodaan ja liitetään alustamalla tallennustilan kanssa. (https://kubernetes.io/docs/concepts/overview/components/)

6.1.1 Kuberneteksen arkkitehtuuri

Kuberneteksen arkkitehtuuri koostuu monista osista, joista isoimpana ja tärkeimpänä voidaan pitää nodea. Nodea voidaan kuvailla Kuberneteksessa virtuaalikoneeksi tai fyysiseksi serveriksi, jossa kubernetes on. Jokainen node sisältää tarvittavat palvelut, jotta podeja voidaan ajaa ja tätä hallinnoi masterkomponentit. Podit ovat ryhmä sovelluskontteja esim. Docker, jotka jakavat levytilan ja lähiverkon. Palvelut nodessa sisältävät Dockerin, kubeletin ja kube-proxyn. Noden status kertoo nodesta neljä asiaa: osoitteet, tilanne, suorituskyky/kapasiteetti ja lopuksi infotiedot.

Noden osoitteet sisältävät isäntänimen, ulkopuolisen IP-osoitteen ja sisäverkon IP-osoitteen. Noden tilanne kertoo mikä on kaikkien päällä olevien nodejen tilanne. Oheisessa kuvassa on näytetty nodejen tilannetiedot, mitä ne voivat sisältää.

Node Condition	Description
`OutOfDisk`	`True` if there is insufficient free space on the node for adding new pods, otherwise `False`
`Ready`	`True` if the node is healthy and ready to accept pods, `False` if the node is not healthy and is not accepting pods, and `Unknown` if the node controller has not heard from the node in the last 40 seconds
`MemoryPressure`	`True` if pressure exists on the node memory – that is, if the node memory is low; otherwise `False`
`DiskPressure`	`True` if pressure exists on the disk size – that is, if the disk capacity is low; otherwise `False`
`NetworkUnavailable`	`True` if the network for the node is not correctly configured, otherwise `False`
`ConfigOK`	`True` if the kubelet is correctly configured, otherwise `False`

(Kuva 14)

Nodejen kunto on esitetty JSON -objektina. Kapasiteettistatus kertoo avoinna olevat resurssit nodessa: prosessorin ja keskusmuistin kapasiteetin sekä nodeen ajoitettavien podejen maksimimäärän. Noden tilannetiedoista näkee esimerkiksi Kernelin version, Kuberneteksen version, käytössä olevan Docker-version ja noden käyttöjärjestelmän. (Kuva 14)

Nodekontrolleri ohjaa monia eri näkökulmia nodesta. Se ylläpitää listaa nodeista, jotka ovat ajan tasalla pilvipalvelujen järjestelmien kanssa. Nodekontrolleri myös ylläpitää nodejen tilannetta koko ajan sekä on vastuussa noden statuksen vaihdosta ja sen päivittämisestä. Sen pitää vaihtaa status määrittelemättömäksi, jos node on tavoittamattomissa. Nodekontrolleri tarkastelee ensimmäiseksi kaikkia nodeja klusterissa, ennen kuin tekee päätöksen podin häädöstä.
(https://kubernetes.io/docs/concepts/architecture/nodes/)

Kaikki kommunikaatioreitit klusterista masterille kulkevat API-palvelimen kautta. Tavallisessa jakelussa API-palvelin kuuntelee etäyhteyksiä turvatun https-portin (portti nro 443) takaa. Nodeille pitäisi suoda julkiset juurivarmenteet, jotta ne voivat turvallisesti yhdistyä API- palvelimeen asiakasohjelman tunnuksilla. Podit, jotka tahtovat yhdistyä API -palvelimeen voivat tehdä tämän vaikuttamalla palvelukäyttäjään, jotta Kubernetes voi antaa julkisen juurisertifikaatin ja oikean ylläpitotokenin podiin, kun se on asennettu. Masterkomponentit keskustelevat klusterin API -palvelimen kanssa turvattoman portin kautta, jota ei ole suojattu. Tämä portti on yleensä avoin vain paikallisen isännän käyttöliittymälle masterlaitteessa, jotta masterkomponentit, jotka ovat samassa laitteessa voivat keskustella klusterin API -palvelimen kanssa.

Kuberneteksessa on kaksi keskeistä keskustelureittiä API -palvelimelta klusteriin. Ensimmäinen reitti näistä kahdesta on API -palvelimen ja kubeletin välinen prosessi, joka toimii jokaisessa nodessa klusterissa. Toinen näistä reiteistä on API -palvelimelta mihin tahansa nodeen, podiin tai palveluun API -palvelimen proxyn lävitse. Yhteyksiä API -palvelimelta kubelettiin käytetään lokien hakemiseen podeista, liittyäkseen jo käynnissä oleviin podeihin ja tarjoten kubeletin portin edelleenlähetys ominaisuutta.
(https://kubernetes.io/docs/concepts/architecture/master-node-communication/)

6.1.2 Service, Load balancing, networking, Storage

Service on Kuberneteksessa REST-objekti. Kuten kaikki REST-objektit POST-metodilla voidaan luoda API palvelimelle uusi instanssi. Jokainen node kubernetes-sen klusterissa käyttää proxya. Proxy on vastuussa siitä, että se antaa jonkun näköisen virtuaalisen ip-osoitteen servicelle. Kuberneteksen proxylla on ainakin kolme ominaisuutta: nimitila, ip-osoitteiden taulu ja ipvs tila. Nimitilassa Kuberneteksen proxy avaa satunnaisen portin jokaiselle servicelle paikalliseen nodeen. Kaikki yhteydet tähän proxyn porttiin kulkevat servicen backendin podin kautta. IP-osoitteiden tauluominaisuudessa jokaista servicea kohtaan asennetaan ip-taulun säännöt, jotka tallentavat liikettä servicen klusteri IP-osoitteeseen, joka on virtuaalinen. IP-osoitteiden taulu on nopeampi ja luotettavampi kuin nimitilan proxy. IPVS eli IP virtual server ohjaa liikennettä paljon nopeammin kuin ip-taulut. IPVS tarjoaa enemmän kuormantasausalgoritmeja kuin edellä mainitut menetelmät.
(https://kubernetes.io/docs/concepts/services-networking/service/#proxy-mode-ipvs)

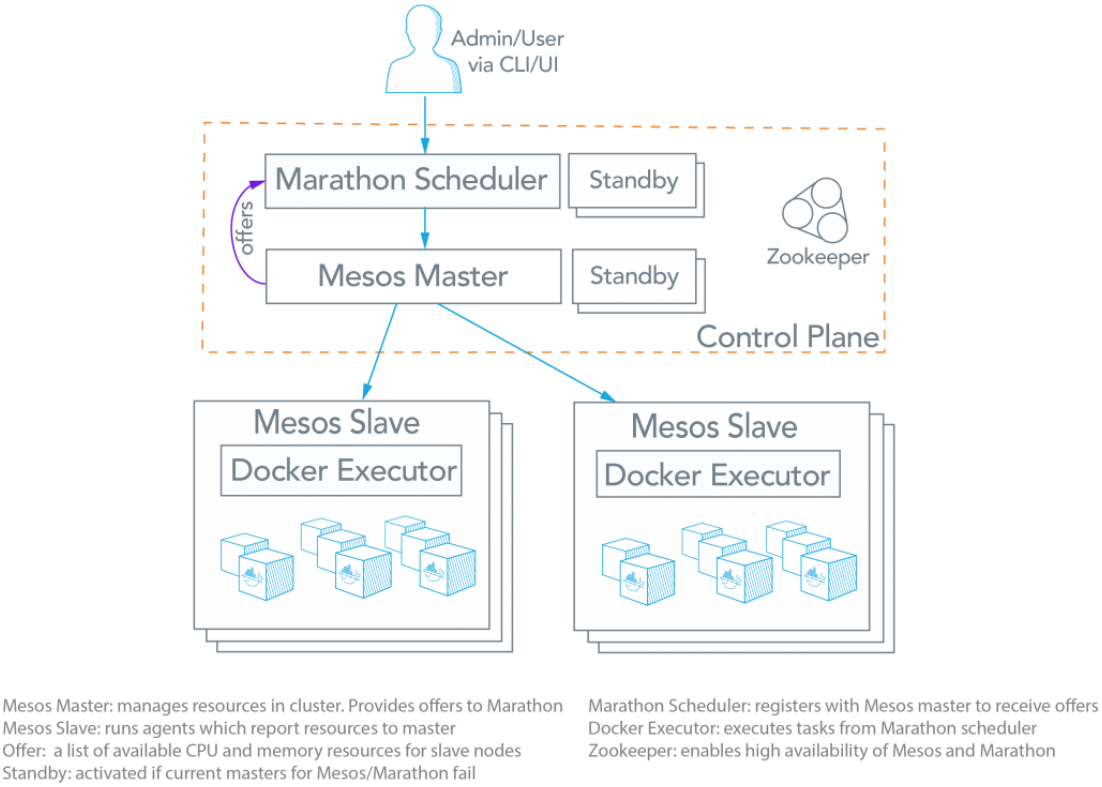
Levyllä olevat tiedostot ovat hetkellisiä epätriviaaleille sovelluksille. Silloin kun sovelluskontti ei toimi enää, kubelet käynnistää sen uudestaan, mutta kaikki tiedostot katoavat samalla. Podeja käytettäessä tiedostoja jaetaan näiden konttien välillä. Kuberneteksen loogisten levyjen abstraktointi hoitaa molemmat nämä ongelmat. Kuberneteksen loogisilla levyillä on täsmällinen elinikä toisin kuin Dockerin levyillä. Dockerin levykuvat kestävät yhtä pitkään kuin podit. Tämän takia looginen levy elää kauemmin, kuin podi ja data on säilöttynä yli kontin, kunnes kontti käynnistyy uudelleen. Podin tuhouduttua siihen laitetut levyt tuhoutuvat myös. Kubernetes tukee monia eri loogisia levyjä ja podi voi käyttää näistä joka ikistä samanaikaisesti. Kubernetes tukee monia eri levymuotoja pilvipalvelujen omista loogisista levyistä perinteisiin iscsi ja nfs muotoihin.
(https://kubernetes.io/docs/concepts/storage/volumes/)

6.2 MESOS & MARATHON

Mesos on Apachen kehittämä yleinen systeemin kernel. Mesos rakennettiin noudattamaan samoja periaatteita kuin Linuxin kernel. Mesos kuitenkin toimii hieman eri tavalla. Mesos toimii jokaisessa laitteessa ja sillä voidaan toteuttaa sovelluksia API:en avulla resurssienhallinnasta aikataulutukseen ympäri koko tietokeskuksien ja pilvipalvelu ratkaisujen. Mesos myös eristää cpu:n, keskusmuistin, tallennustilan ja muut laskentaresurssit muista järjestelmistä. Tämä tekee järjestelmästä vikasietoisen ja elastisen, joka voidaan helposti rakentaa ja toimii tehokkaasti.

Mesos toimii Linux, Windows ja MacOS -käyttöjärjestelmissä. Mesos tarvitsee Linux kernelistä 3.10 version, jotta prosessit voidaan eristää. Mac OS X -käyttöjärjestelmät tarvitsevat Xcode -ohjelman, jotta Mesos voidaan asentaa järjestelmään. Mesos pystyy skaalautumaan 10000 palvelimeen ja siinä on rakennettu käyttäjävalikko, jotta nähdään clusterin tilanne.
(http://mesos.apache.org/)

Marathon on konttien automatisoitu, koordinoitu ja hallinnoitu alusta Apachen Mesokselle. Marathonissa on monia ominaisuuksia, kuten korkea saatavuus, joka tarjoaa jopa 100%:n ylläpitovalmiuden. Marathon tarjoaa myös monen kontin samanaikaista käyttöä, kuten Mesoksen ja Dockerin oman konttialustan samaan aikaan. Marathonissa on myös käyttöliittymä hallinnointia varten ja se tarjoaa myös kuormantasausta nimipalvelun ja proxyn kautta. Mesoksen oma nimipalvelin on hyödyllinen, kun sovelluksia suoritetaan monen eri käyttöliittymän lävitse. Kuormantasausta voidaan hoitaa myös HAproxyn kautta. HAproxy on Marathon-ib:n mukana oleva ohjelma, joka on docker sovellus. Marathon-ib tukee edistynyttä SSL -tekniikkaa, kiinteitä yhteyksiä ja virtuaaliseen isäntään perustavaa kuorman tasausta. Näin voit osoittaa Marathon sovelluksille virtuaalisen isännän, joka vähentää kuormantasausta.
(https://mesosphere.github.io/marathon/docs/service-discovery-load-balancing.html)



(Kuva 15)

(https://platform9.com/wp-content/uploads/2017/07/mesos-architecture-1024x731.png)

Ylemmässä kuvassa näkyy, miten Mesoksen ja Marathonin yhteistyö toimii. Mesos Master node sallii resurssien käytön erilaisilla käyttöliittymillä. Marathon Scheduler ottaa vastaan tietoa Mesos masterilta vapaana olevista prosesseista ja keskusmuistista. Mesos Slave raportoi Mesos Masterille vapaana olevat resurssit. Docker Executor ottaa vastaan tehtäviä Marathon Schedulerilta ja julkaisee sovelluskontteja muille palvelimille. (Kuva 15)

Docker-kontit voidaan toteuttaa käyttämällä JSON-erottuvuuksia. Nämä erottuvuudet määrittelevät säilytyspaikan, resurssit, montaako instanssia käytetään ja toteutuksen. Konttien skaalaus ylöspäin voidaan tehdä Marathonin -käyttöliittymän kautta ja Marathonin järjestäjä vie nämä oikealle solmulle riippuen kriteereistä. Marathon tarkkailee jatkuvasti instanssien määrää Dockerin konteista. Yhden kontin epäonnistuessa Marathon järjestää tämän kontin toiselle solmulle.
(https://platform9.com/blog/kubernetes-vs-mesos-marathon/)

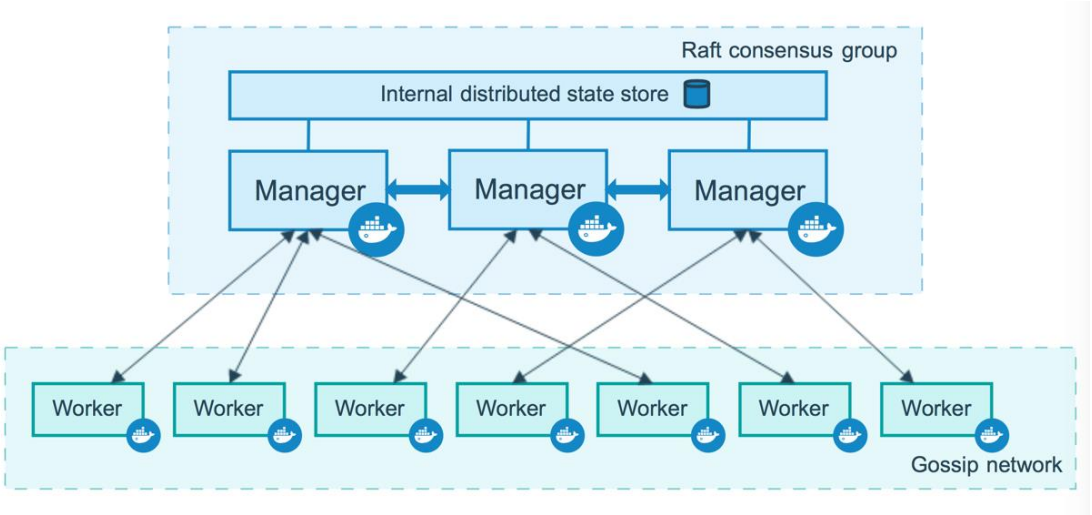
Marathon ja Mesos käyttävät molemmat konttien terveystarkastuksia. Molemmat käyttävät http, https ja tcp protokollia näiden toteuttamiseen. Marathonin terveystarkastuksissa on monia rajoitteita verrattuna Mesoksen terveystarkastuksiin. Terveystarkastus luo lisäliikennettä verkkoon, jos tehtävä ja järjestelijä ovat eri solmussa. Marathonin hoitaa montaa tehtävää samaan aikaan, mikä voi johtaa järjestelijässä suorituskykyhaittoihin. Pienissä klustereissa ja ympäristöissä nämä Marathon -tason terveystarkastukset voivat toimia, mutta isossa ratkaisussa suositellaan käytettäväksi Mesos -tason terveystarkastuksia. Mesos -tason terveystarkastukset suoritetaan paikallisesti ja ne suoritetaan niin lähellä tehtävää kuin mahdollista, joten verkon ongelmat eivät vaikuta niihin. Terveystarkastusprosessi jakaa resurssit tehtävän resurssien kanssa, joten lisäresurssit pitää ottaa huomioon ympäristössä.
(https://mesosphere.github.io/marathon/docs/health-checks.html)

Mesos ja Marathon käyttävät myös podeja, joita Kuberneteskin tukee. Podeissa jaetaan tallennustila, verkko ja muut resurssit tunnetusti yhdeksi ryhmäksi. Marathonissa podit voidaan tehdä REST API:n kautta käyttöliittymän sijaan. Podi on myös erityinen Mesos tehtäväryhmä ja tehtävät tai sovelluskontit ovat osa tätä ryhmää. Marathonin podit tukevat vain Mesoksen kontitustekniikkaa, joka tukee monia levykuvia, esimerkiksi Dockeria. Mesoksen kontitustekniikka yksinkertaistaa verkon käyttämisen ja kontit keskustelevat keskenään virtuaalisen lähiverkon tai yksityisen verkon kautta.
(https://mesosphere.github.io/marathon/docs/pods.html)

7 DOCKER SWARM

Docker Swarm on Dockerin oma sovelluskonttien hallintaohjelma. Docker Swarm koostuu useasta Docker -isännästä, jotka ovat joukkomoodissa ja toimivat hallinnoijana tai työläisinä. Docker-isäntä voi päättää kumman roolin hän haluaa vai haluaako olla molemmat. ”Workerit” eli työläiset hoitavat Docker-palveluita. Palvelu on tehtävä, joka suoritetaan Docker nodessa. Docker Swarm palveluiden etu verrattuna yksittäiseen sovelluskonttiin on mahdollisuus muuttaa tallennustilaa ja verkkoa, johon palvelu on liittynyt ilman manuaalista palvelun uudelleenkäynnistämistä. Sovelluksen levittäminen Docker Swarmissa tapahtuu sovelluksen toimittamisella hallintanodelle. Tämän jälkeen hallintanode lähettää tarvittavat resurssit eli tehtävät työntekijänodelle. Hallintanode myös suorittaa järjestämisen ja klusterin hallinnan, jotta saadaan tarvittava tila Swarmiin. Docker nodella tarkoitetaan virtuaalista tai fyysistä palvelinta.
(https://docs.docker.com/engine/swarm/key-concepts/#nodes)

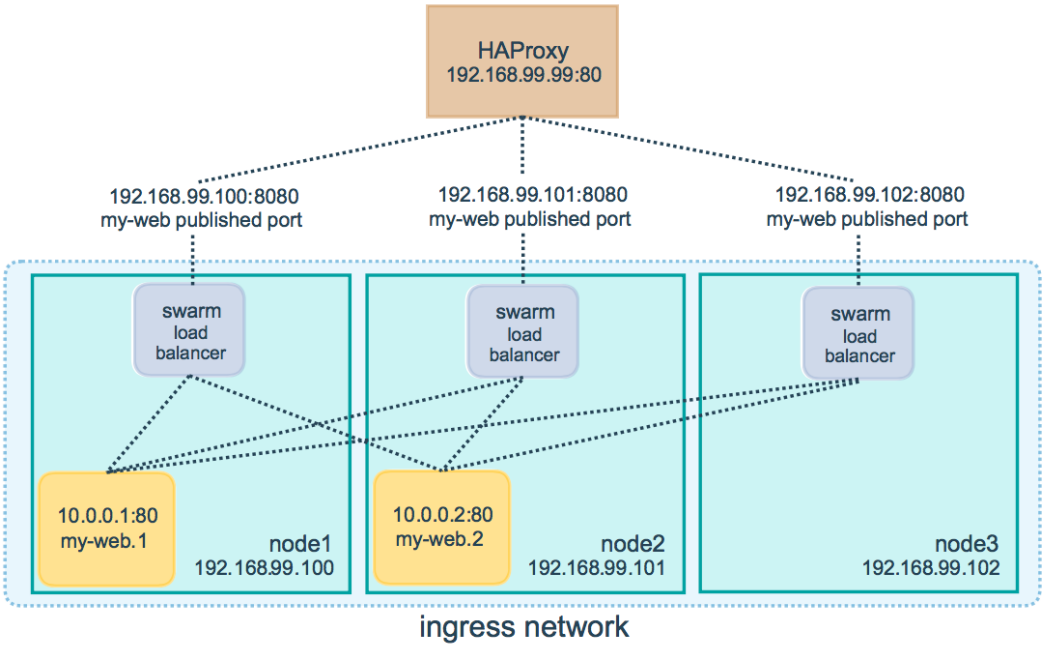
Docker Swarm käyttää omaa komentokehoteriviä hallinnoimiseen, joten erillistä graafista käyttöliittymää ei ole suunniteltu. Mitään ylimääräisiä sovelluksia ei tarvita Docker Swarmin luomiseen tai hallinnoimiseen. Docker helpottaa käyttäjää, sillä Docker Engine luo molemmat nodet mitä Docker Swarmissa on. Tämä tarkoittaa, että kokonaisen Swarmin voi luoda yhdestä levykuvasta. Voit myös Docker Swarmissa valita kuinka monta tehtävää palveluissa on päällä samaan aikaan. Ympäristön laajentuessa tai pienentyessä Docker osaa lisätä tai poistaa tehtäviä, jotta haluttu tehtävä on päällä. Tällöin vältetään resurssien turha käyttäminen ja käytössä olevien palvelimien ylikuormittaminen.
(https://docs.docker.com/engine/swarm/#feature-highlights)



(Kuva 16)
(https://docs.docker.com/engine/swarm/images/swarm-diagram.png)

Voit luoda Docker -palveluille päällysverkon, jonka avulla Docker-manageri antaa osoitteet sovelluskonteille päällysverkossa silloin kun sovellus asennetaan tai päivitetään. Docker Swarmin hallintanodet antavat jokaiselle palvelulle uniikin nimipalvelunimen ja auttavat tasaamaan kuormaa päällä olevissa konteissa. Nimipalvelimen kautta voidaan kysyä jokaista sovelluskonttia, joka on päällä Swarmissa. (Kuva 16)
(https://docs.docker.com/engine/swarm/#whats-next)

Docker tekee helpoksi porttien julkistamisen palveluille, jotta ulkopuoliset resurssit ovat saatavilla Swarmin ulkopuolella. Kaikki Dockerissa olevat nodet ovat osallisia reititysverkkoon. Reititysverkko sallii jokaiselle nodelle yhteyden julkistettuihin portteihin palveluille, jotka ovat käynnissä Swarmissa. Tämä toimii vaikka yksikään tehtävä ei olisi käynnissä nodessa. Reititysverkko ohjaa kaikki sisään tulevat pyynnöt julkistettuihin portteihin, joissa on aktiivisia sovelluskontteja. Reititysverkkoon voi myös konfiguroida ulkoisen kuormantasaajan. Ulkoista kuormantasaajaa voi käyttää reititysverkon kanssa tai ilman. Kuormantasaajan voi laittaa tasaamaan kuormaa palvelussa, joka on tietyssä portissa. Portti, jossa tämä palvelu on käynnissä pitää olla auki kuormantasaajalle ja nodeille Swarmissa. (Kuva 17)
(https://docs.docker.com/engine/swarm/ingress/)

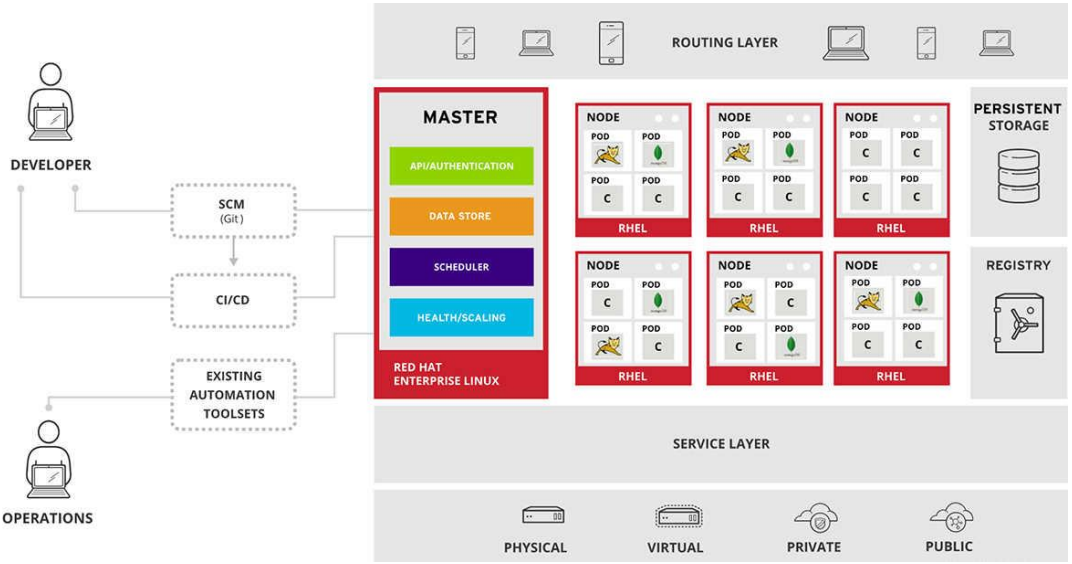


(Kuva 17)
(https://docs.docker.com/engine/swarm/images/ingress-lb.png)

8 OPENSHIFT

RedHat -yhtiö on kehittänyt alustaratkaisun Dockerin ja Kuberneteksen saman aikaiseen käyttöön. RedHat tarjoaa myös oman sovelluksen ohjelmointirajapinnan, jotta voidaan hallinnoida näitä palveluja. OpenShiftin tarkoituksena on auttaa kehittämään, ottamaan käyttöön ja hallinnoimaan sovelluskonttien sovelluksia. RedHat kuvailee OpenShiftiä alustapalveluratkaisuna, joka tunnetaan myös Platform as a service -palveluna. (Kuva 18)
(https://docs.openshift.com/container-platform/3.7/getting_started/index.html)

8.1 Openshift arkkitehtuuri



(Kuva 18)

https://hostadvice.com/wp-content/uploads/2017/05/rhelos.jpg

OpenShift käyttää podeja Kuberneteksen avulla. Maksimissaan podeja voi olla yhdellä OpenShift isäntäpalvelimella 110. OpenShift käyttää podien uudelleen käynnistämisessä kolmea arvoa. Always -arvo yrittää käynnistää podia 10, 20 ja 40 sekunnin välein, kunnes podi käynnistyy uudelleen. OnFailure -arvo sen sijaan tekee saman, kuin Always -arvo, mutta lopettaa yrittämisen viiden minuutin jälkeen. Never -arvo ei yritä käynnistää podia uudelleen. Podi poistuu tämän tapahtuman jälkeen. OpenShift käyttää myös Init kontteja Kuberneksen kanssa. Init kontti käynnistyy ennen podin sovelluskontin käynnistymistä. Se voi jakaa levytilaa, suorittaa verkossa tehtäviä operaatioita ja suorittaa laskentaa ennen kuin sovelluskontti käynnistyy. Init kontti voi estää tai viivästyttää sovelluskontin käynnistämistä ennen kuin joku tietty edellytys on saavutettu. Niitä voidaan käyttää vain yhtä kerrallaan ja ensimmäisen Init kontin on pitänyt poistua ennen kuin seuraava voi käynnistyä.
(https://docs.openshift.com/container-platform/3.7/architecture/core_concepts/pods_and_services.html)

OpenShift käyttää Kuberneteksen persistent volumea eli pysyvää loogista levyä. Tämä auttaa hallitsemaan tallennustilaa klustereissa. Sovelluksien kehittäjät voivat täten pyytää loogisten levyjen resursseja ilman, että heidän tarvitsee tietää tallennustilan infrastruktuurista mitään. Nämä pysyvät loogiset levyt ovat tiettyjä projektissa ja ne luodaan klusteriin, jotta sitä voidaan käyttää jokaisesta projektista. OpenShift tukee tiettyjä loogisten levyjen muotoja. Listassa on pilvipalvelujen omia levymuotoja ja perinteisiä levymuotoja. OpenShift käyttää myös levyissä Access moodia, jossa tietty loogisen levyn muoto voi tehdä tiettyjä asioita. Access moodi sisältää kolme eri tilaa tietyille levyille. ReadWriteOnce -tilassa levy voidaan liittää yhteen nodeen, jossa tähän levyyn voidaan kirjoittaa tietoa ja lukea myös tietoa. ReadOnlyMany -tilassa levy voidaan liittää moneen eri nodeen, joista sen sisältämää dataa voidaan vain lukea. ReadWriteMany -tilassa levy voidaan liittää moneen eri nodeen ja siihen voidaan kirjoittaa dataa, jota voidaan myös lukea. Esimerkiksi NFS -levy tukee näitä kolmea eri tilaa.
(https://docs.openshift.com/container-platform/3.7/architecture/additional_concepts/storage.html)

OpenShift käyttää sovellusmääriteltyä verkostoitumista tarjotakseen yhdistetyn klusteri verkoston. Tämän avulla podit voivat keskustella keskenään koko OpenShift klusterissa. OpehShift tarjoaa kolme liitännäistä podiverkoston konfiguroimiseen. Ensimmäisenä liitännäisenä on ovs-subnet liitännäinen. Tämä on alkuperäinen liitännäinen ja tarjoaa verkon, jossa jokainen podi voi keskustella keskenään ja muiden palvelujen kanssa. Seuraava liitännäinen on ovs-multitenant, joka tarjoaa ryhmätason eristyksen podeille ja palveluille. Jokainen projekti saa tässä liitännäisessä oman virtuaalisen verkkotunnuksen. Tällä tavoin estetään projektien keskinäinen keskusteleminen verkossa. Kolmas liitännäinen on ovs-networkpolicy, joka antaa projektin johtajalle oikeuden konfiguroida omat verkon eristystavat käyttäen verkon menettelytapoja.
(https://docs.openshift.com/container-platform/3.7/architecture/networking/sdn.html)

9 MICROSOFT & DOCKER

Saadessaan kontitusteknologian Windows -käyttöjärjestelmille yhteistyönä Dockerin kanssa, Microsoft toi sovelluskonttien valitsemiseen uuden kilpailijan. Tämä tarkoitti sitä, että Windows 10 -käyttöjärjestelmä sai tuen Linux -alijärjestelmään ja Windows 2016 -palvelin sai Docker työkalut. Tämä vie Microsoftia eteenpäin pilvessä olevien ohjelmien kehityksessä ja tukee Microsoftin Azure -palvelun viemistä eteenpäin. Windows suosittelee uusien sovelluskonttien tekoon peruslevykuvia Nano -palvelimessa. Microsoft toi Nano -palvelimen Windows 2016 -palvelimen yhteydessä. Nano -palvelin on pilvipalveluihin keskittynyt sovellus, joka on pieni, nopea eikä sisällä käyttöliittymää.
(https://www.infoworld.com/article/3163257/application-development/what-you-need-to-know-about-docker-in-windows.html)

Windows Containers tukee kahta eri .NET -ratkaisua rakennettaessa palvelinpuolen Docker sovelluskonttisovelluksia. Nämä kaksi .NET -ratkaisua jakavat .NET -standardit työkalut ja voivat jakaa koodia näiden kahden kesken. Näiden kahden .NET -ratkaisun välillä on keskeisiä eroja riippuen siitä, mitä ratkaisua käyttäjä haluaa saavuttaa.
(https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/net-core-net-framework-containers/index)

Microsoft suosittelee .NET Core -ratkaisun käyttämistä Windows ja Linux sovelluskonteissa. NET Core ohjelmistokehystä voi myös käyttää, jos sovelluksien arkkitehtuuri käyttää mikropalveluja. Tätä ohjelmiston kehystä käytetään myös silloin, kun halutaan sovelluskonttien käynnistyvän nopeasti ja halutaan pieni jalanjälki per sovelluskontti. Tällöin saavutetaan se, että kontteja on paljon per laitteisto, jolloin voidaan laskea kuluja.

(https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/net-core-net-framework-containers/general-guidance)

NET Frameworkia tulisi käyttää Windows -sovelluskonteissa, jos sovellukset tukevat sitä ja ovat riippuvaisia Windows -ympäristöstä sekä silloin, jos sovelluksissa tarvitsee käyttää paljon Windows API:ja, jotka eivät tue .NET Corea. Microsoft suosittelee NET Frameworkin käyttöä, jos käyttäjät joutuvat käyttämään kolmansien osapuolien .NET kirjastoja jotka eivät ole saatavilla NET Corelle. (https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/net-core-net-framework-containers/general-guidance)

Microsoft suosii enemmän NET Core -ohjelmistokehystä sen monipuolisuuden sekä Linux -tuen takia. NET Core kehys ei kuitenkaan tue kaikkea, joten NET Framework toimii samoissa tilanteissa kuin NET Core. NET Framework toimii Windows -sovelluksien siirrossa sovelluskontteihin, johon NET Core -ohjelmistokehys ei pysty. NET Framerwork -kehyksen Windowsriippuvuus rajoittaa toimintaa ja käyttötapoja. (https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/net-core-net-framework-containers/container-framework-choice-factors)

10 WORDPRESS ASENNUS

Opinnäytetyössä käsitellään WordPress -internetsivujen työkalun asennusta Microsoftin Azure-pilvipalveluun. Asennuksessa käytetään Azure Container Service -palvelua, joka mahdollistaa sovelluskonttien suorittamisen Azuren -pilvipalvelussa. Tämän lisäksi Azuren pilvipalveluun asennetaan Kubernetes -sovelluskonttien hallintaohjelma. Nämä ovat asennuksen pääkomponentit Azuren pilvipalvelussa. Muita pienempiä liitännäisiä asennetaan myös ja niitä käydään läpi asennuksen edetessä.

Azuren pilvipalvelun valittiin siksi, että se on melko tunnettu. Azure tukee Dockeria omalla Azure Container Servicellään. Azuren Container Service tukee Windows ja Linux -käyttöjärjestelmiä. Azure Container Service tukee Kubernetesta, jota käytetään sovelluskontin hallinnoimiseen. Kuberneteksen valittiin asennukseen sen takia, että se on monipuolinen ja Kuberneteksen eri liitännäiset tekevät siitä käyttäjäystävällisen. Se myös tukee monia eri käyttöjärjestelmiä, pilvipalveluja ja monia eri alustoja. WordPressi valittiin siksi, että se on suosittu internetsivujen luonnissa ja monet internetsivut käyttävät WordPressiä. WordPress on myös helppokäyttöinen ja käyttäjäystävällinen. Komponentit asennukseen valittiin sen takia, että näillä komponenteilla voi esimerkiksi aloittaa oman blogin käytön ilman omia palvelimia ja saada palvelusta nykyaikaisen.

Asennus dokumentoidaan mahdollisimman tarkasti ja siinä pyritään näyttämään selkeästi asennuksessa käytetyt komennot. Myös koodinpätkät dokumentoidaan mahdollisimman selkeästi, esimerkiksi kerrotaan mitä kukin koodinpätkä tekee ja mihin sitä tarvitaan. Kuvia käytetään asennuksen hahmottamisessa ja selventämään ympäristöä. Tarvittavat tiedostot asennuksessa löytyvät Kuberneteksen dokumentaatioista.

10.1 Asennus

WordPress asennus aloitettiin rekisteröimällä Azuren Container Service ominaisuus käyttäjätiliin. Tämän rekisteröinnin jälkeen käyttäjätilillä voi aloittaa Kubernetes klusterien käyttämisen.

Azure CLI

```
az provider register -n Microsoft.ContainerService
```

(Kuva 19 komento Container Servicen rekisteröimiseen Azuren käyttäjätiliin.)

Rekisteröimisen jälkeen luotiin resurssiryhmä Azureen. Resurssiryhmä on ryhmä, jossa voi olla eri käyttäjiä, jotka hallinnoivat sen resurssiryhmän resursseja. Käyttäjille voidaan antaa eri arvoja resurssiryhmässä, jotka mahdollistavat eri tehtävät siinä. Käyttäjät voivat lisätä, poistaa ja päivittää siinä resurssiryhmässä olevia resursseja. Resurssiryhmän voi nimetä projektin tai ryhmän mukaan, joka vastaa projektista. Asennuksessa resurssiryhmän nimi on alla oleva ”myResourceGroup”.

Azure CLI

```
az group create --name myResourceGroup --location eastus
```

(Kuva 20 luodaan resurssiryhmä Azuren itäisen USA:n palvelimelle.)

Resurssiryhmän luomisen jälkeen luodaan itse Kubernetes klusteri. Kubernetes klusteri luotiin tässä demossa Azure Cloud Shell komentorivillä.

```
santeri@Azure:~$ az aks create --resource-group myResourceGroup --name mywpcluster --node-count 1 --generate-ssh-keys
```

(Kuva 21 komennolla luodaan Kubernetes klusteri. Komennolla luodaan ”myResourceGroup” Resurssiryhmään kubernetes klusteri, jonka nimi ”mywpcluster” ja samalla luodaan yksi node klusteriin ja tarvittavat ssh-avaimet, jotta saadaan yhteys klusteriin. Huomataan komennon jälkeen, että kubernetes klusterin asennus alkoi ja onnistui.

```
santeri@Azure:~$ az aks get-credentials --resource-group=myResourceGroup --name=mywpcluster
Merged "mywpcluster" as current context in /home/santeri/.kube/config
santeri@Azure:~$ kubectl get nodes
NAME                       STATUS    ROLES     AGE       VERSION
aks-nodepool1-26288741-0   Ready     agent     1h        v1.7.9
santeri@Azure:~$
```

(Kuva 22 klusterin asennuksen jälkeen konfiguroidaan kubectl toimimaan klusterissa, minkä jälkeen huomataan noden olevan päällä Kuberneteksen klusterissa.)

Kubernetes klusterin asennuksen jälkeen ja kubectl konfiguroimisen jälkeen asennetaan MySQL tietokantaohjelmisto kubernetes klusteriin. MySQL ja WordPress vaativat myös PersistentVolumeClaim- ja PersistentVolume-konfiguraatiot. Nämä konfiguraatiot mahdollistavat, että data ei tuhoudu vaikka podi, jossa MySQL ja WordPress ovat, tuhoutuisi tai lakkaisi toimimasta.

Ensimmäiseksi asennuksessa luodaan Secret MySQL salasanaan. Secret on objekti, joka pitää sisällään tärkeät tiedot esimerkiksi salasanat ja tärkeät avaimet.

```
santeri@Azure:~$ kubectl create secret generic mysql-pass --from-literal=password=qwerty123
secret "mysql-pass" created
santeri@Azure:~$
```

(Kuva 23)

Luodaan Secret ja annetaan sille salasana. Huomataan, että saatiin onnistuneesti luotua secret MySQL salasanalla. Tämän jälkeen voidaan vielä tarkistaa tuliko Secret todella luotua.

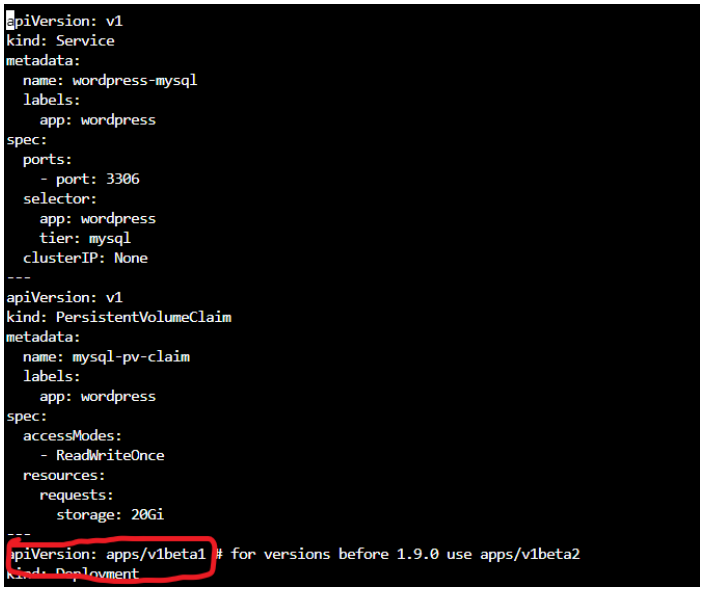
```
santeri@Azure:~$ kubectl get secrets
NAME                  TYPE                                  DATA      AGE
default-token-4t0dw   kubernetes.io/service-account-token   3         21d
mysql-pass            Opaque                                1         37s
santeri@Azure:~$
```

(Kuva 24 Nähdään että Secret on onnistuneesti luotu)

Seuraavaksi voidaan asentaa MySQL Kubernetes klusteriin. MySQL-sovelluskontti liittää PersistentVolumen paikalliseen kansioon. MySQL-sovelluskontin voi asentaa Azureen ainakin kahdella tavalla. Yksi mahdollisuus asentaa MySQL sovelluskontti on ladata MySQL yaml-tiedosto Azuren pilvipalveluun oman käyttäjän tiedostoihin. Tämä yaml-tiedosto pitää kuitenkin ensimmäiseksi ladata omalle työasemalle ja siitä

ladata se Azureen. Toinen tapa on ladata tiedosto Kuberneteksen käyttöliittymän kautta. Tämä voi olla vaikeampi tapa, koska Kuberneteksen käyttöliittymää ei saa avattua Azuren komentorivin kautta. MySQL-tietokantaohjelmistoa käytetään sen takia, että se on tuttu ja Windows alustat tukevat sitä myös.

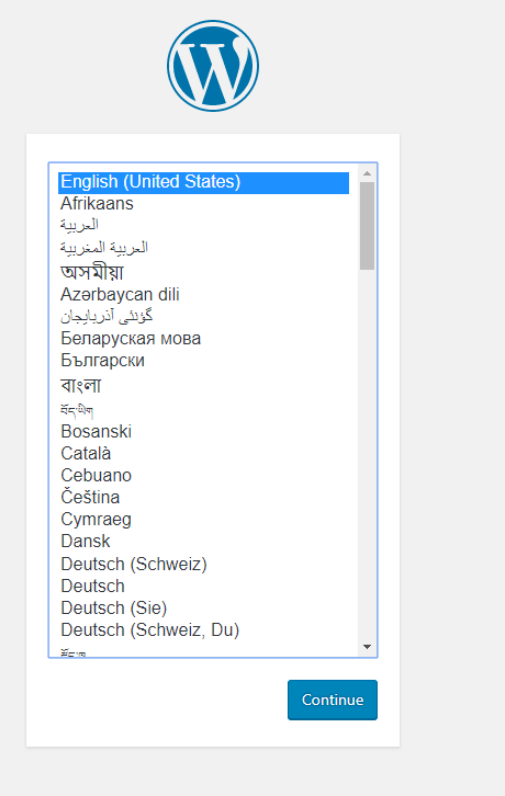
Tämän jälkeen voidaan asentaa MySQL sovelluskontti klusteriin. Sovelluskontin luonti onnistuu komennolla: kubectl create -f mysql-deployment.yaml. Komennon syöttämisen jälkeen huomataan, että sovelluskonttia ei voi luoda, koska API versio on eri tiedostossa kuin Kubernetes klusterissa. Nano mysql-deployment.yaml komennolla päästään muokkaamaan tiedostoa ja muuttamaan API -versio oikeaksi.



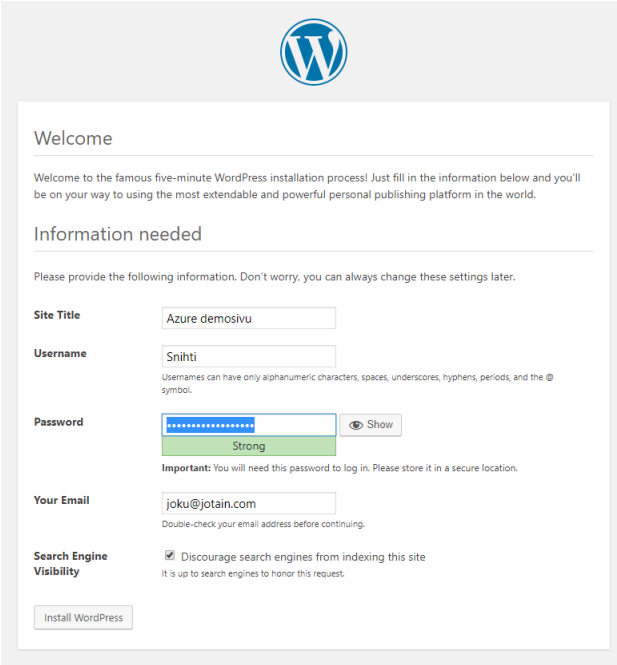
(Kuva 25 API version korjaus)

Muokataan API -versio oikeaksi kuvassa olevalla tavalla. Tämän jälkeen suoritetaan uudestaan MySQL sovelluskontin luomiskomento ja huomataan, että sovelluskontin luonti onnistui. MySQL sovelluskontin luonnin jälkeen voidaan asentaa WordPress sovelluskontti. WordPress sovelluskontti asennetaan ja sitä muokataan samalla tavalla kuin MySQL sovelluskonttia. Muokkauksen jälkeen voidaan asentaa WordPress sovelluskontti. WordPress sovelluskontin luonti tapahtuu samanlaisella komennolla kuin MySQL sovelluskontin luonti. Komennossa muuttuu vain käytettävän tiedoston nimi. WordPress sovelluskontin luontikomento on siis tässä asennuksessa: kubectl create -f wordpress-deployment.yaml. Tämän jälkeen huomataan teksti, jossa sanotaan WordPressin luonnin onnistuneen.

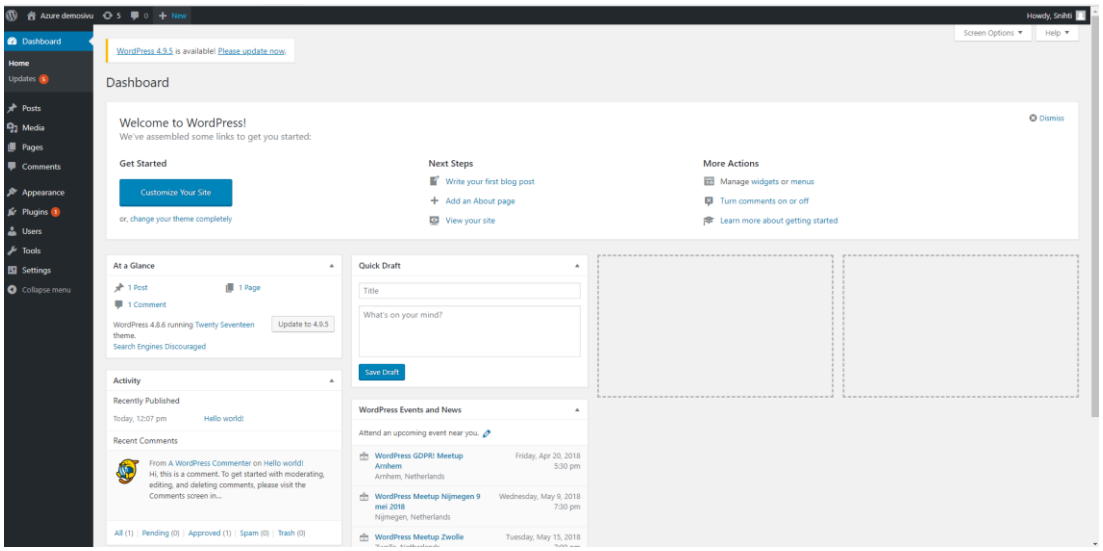
WordPressin asentaminen vaatii ensimmäiseksi palvelun aloittamista Kubernetes klusteriin ja WordPressiin. Palvelun aloittamisen jälkeen saadaan ulkoinen IP-osoite WordPress-sovelluskontille, mistä WordPress -sivu löytyy jatkossa sen asentamisen jälkeen. WordPressin asennussivulle pääsee syöttämällä ulkoisen IP-osoitteen ja portin numeron, johon ulkoinen liikenne menee. Portin numero on 80. WordPress käyttää tätä porttinumeroa, koska liikennettä ei ole vielä suojattu erilaisilla sertifikaateilla. Esimerkki WordPress asennus IP-osoitteesta: 1.2.3.4:80. Tämän jälkeen aukeaa WordPressin asennussivu. Asennus kannattaa tehdä loppuun asti tai poistaa koko asennus, koska ulkopuolinen henkilö voi syöttää saman IP-osoitteen ja asentaa WordPress-sivun omilla tiedoillaan. WordPressin asennussivun voi myös avata Kubernetes käyttöliittymästä Services kohdasta, missä näkyy External endpoint ja IP-osoite.



(Kuva 26 WordPress-asennuksen kielen valitseminen)



(Kuva 27 WordPress-asennuksen tiedot)



(Kuva 28 WordPress Dashboard, johon päästiin onnistuneen asennuksen jälkeen.)

Näin saatiin asennettua WordPress -sovelluskontti ja MySQL -sovelluskontti, jotta saadaan WordPress -internetsivujen luontityökalu toimimaan. Tätä asennusta voisi jatkokonfiguroida vielä eteenpäin, mutta sitä ei käsitellä tässä asennuksessa.

10.2 LOPUKSI

Sovelluskontit ovat yksi nykyaikaisista teknologian muodoista, jolla saadaan sovellukset toimimaan vähäisillä resursseilla. Sovelluskontit ovat kevyitä ja helpottavat sovelluksien kanssa työskentelyä, koska sovellus voidaan testata samalla työasemalla, jolla se on luotu. Mitään erillisiä ympäristöjä sovellusten testaamiselle ei tarvita. Tämä auttaa resurssien pienentämisessä ja niiden uudelleenjärjestämisessä. Sovelluskontit ovat hyvä muutos virtuaalipalvelimille syrjäyttämättä niitä, mutta tuovat erilaista teknologiaa niiden rinnalle.

Sovelluskontit ovat kehittymässä kovaa vauhtia eteenpäin ja auttavat varsinkin isoja yrityksiä pääsemään eroon isoista virtuaaliratkaisuista. Uskon, että tulevaisuudessa sovelluskontit ovat ratkaisu sovelluksien ympäristössä. Sovelluskonttien haittapuoli on se, että ne eivät välttämättä toimi vanhalla teknologialla. Teknologian kehittyessä ja uusien ratkaisujen tullessa sovelluskontit ovat niissä varmaan isossa osassa. Nykymaailmassa yhdessä klusterissa voi olla tuhansia eri nodeja pyörimässä samaan aikaan.

Tässä työssä käytiin läpi sovelluskontteja ja niiden sisältöä. Aihetta kävin omasta mielestäni hyvin lävitse ja aika monipuolisesti. En puuttunut pelkästään Dockeriin, vaan otin muitakin vaihtoehtoja ja paneuduin niihinkin enemmän. Tavoitteeni oli käydä kattavasti läpi Dockeria ja sen tekniikkaa sekä sovelluskonttien hallintaohjelmia sisältöineen ja tärkeimpine ominaisuuksineen. Tässä onnistuin mielestäni hyvin.

Työn lopussa tehtiin demoasennus Azuren pilvipalveluun. Demoasennuksessa asennettiin WordPress ja MySQL -sovelluskontit Azureen. Sovelluskonttien asennus onnistui hyvin Azuren palveluun. Asennus toteutettiin käyttämällä Azuren omaa komentoriviä, mikä on Linux-pohjainen. Azuren pilvipalvelu oli omasta mielestäni hyvä, koska heillä oli oma sovelluskonttipalvelu ja hyvät dokumentaatiot heidän palvelunsa toiminnasta. Huonoa Azuren omassa komentorivissä oli se, että se ei tue sovelluksien avaamista, vaan pitää asentaa Azuren komentorivi omalle työasemalle, jos haluaa suorittaa komentoja, jotka vaativat selaimen avaamista.

LÄHTEET

Apache Mesos Viitattu 20.2.2018 http://mesos.apache.org/

Bisson S 2017 What you need to know about Docker in Windows Viitattu 5.3.2018 https://www.infoworld.com/article/3163257/application-development/what-you-need-to-know-about-docker-in-windows.html

Docker Inc Get Docker CE for Ubuntu Viitattu 3.2.2018 https://docs.docker.com/install/linux/docker-ce/ubuntu/
Docker Inc Docker for Mac vs. Docker Toolbox Viitattu 6.2.2018 https://docs.docker.com/docker-for-mac/docker-toolbox/

Docker Inc Get started with Docker for Mac Viitattu 6.2.2018 https://docs.docker.com/docker-for-mac/#preferences

Docker Inc Leverage multi-CPU architecture support Viitattu 6.2.2018 https://docs.docker.com/docker-for-mac/multi-arch/

Docker Inc Product Viitattu 25.1.2018 https://www.docker.com/get-docker#/more_resources

Docker Inc Community Edition Viitattu 25.1.2108 https://www.docker.com/community-edition

Docker Inc Enterprise Edition Viitattu 25.1.2018 https://www.docker.com/enterprise-edition

Docker Inc About Docker EE Viitattu 3.2.2018 https://docs.docker.com/enterprise/

Docker Inc Install Docker Viitattu 3.2.2018 https://docs.docker.com/install/

Docker Inc Get started with Docker for Windows Viitattu 3.2.2018 https://docs.docker.com/docker-for-windows/

Docker Inc Welcome to the Docker Cloud docs Viitattu 8.2.2016 https://docs.docker.com/docker-cloud/

Docker Inc Why Docker for AWS Viitattu 10.2.2018 https://docs.docker.com/docker-for-aws/why/

Docker Inc Docker for AWS setup & prerequisites Viitattu 10.2.2018 https://docs.docker.com/docker-for-aws/

Docker Inc Docker for Azure setup & prerequisites Viitattu 10.2.2018 https://docs.docker.com/docker-for-azure/

Docker Inc Why Docker EE for IBM Cloud Viitattu 12.2.2018 https://docs.docker.com/docker-for-ibm-cloud/why/

Docker Inc Docker EE for IBM Cloud setup & prerequisites Viitattu 12.2.2018 https://docs.docker.com/docker-for-ibm-cloud/

Docker Inc Swarm mode key conceps Viitattu 24.2.2018 https://docs.docker.com/engine/swarm/key-concepts/

Docker Inc Swarm mode overview Viitattu 26.2.2018 https://docs.docker.com/engine/swarm/#feature-highlights

Docker Inc What's next Viitatty 26.2.2018 https://docs.docker.com/engine/swarm/#whats-next

Docker Inc Use swarm mode routing mesh Viitattu 27.2.2018 https://docs.docker.com/engine/swarm/ingress/

Kubernetes/The Linux Foundation What is Kubernetes Viitattu 15.2.2018 https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Kubernetes/The Linux Foundation Kubernetes Components Viitattu 15.2.2018 https://kubernetes.io/docs/concepts/overview/components/

Kubernetes/The Linux Foundation Nodes Viitattu 16.2.2018 https://kubernetes.io/docs/concepts/architecture/nodes/

Kubernetes/The Linux Foundation Master-Node communication Viitattu 16.2.2018 https://kubernetes.io/docs/concepts/architecture/master-node-communication/

Kubernetes/The Linux Foundation Services Viitattu 17.2.2018 https://kubernetes.io/docs/concepts/services-networking/service/#proxy-mode-ipvs

Kubernetes/The Linux Foundation Volumes Viitattu 17.2.2018 https://kubernetes.io/docs/concepts/storage/volumes/

Lardinois F 2016 wtf is container Viitattu 16.1.2018 https://techcrunch.com/2016/10/16/wtf-is-a-container/

Mackie K 2017 Windows Server 2016 version 1709 To Support Linux and Windows Containers Viitattu 16.1.2018 https://redmondmag.com/articles/2017/09/14/windows-server-2016-version-1709.aspx

Martin M 2015 A brief history of Docker Containers' overnight success Viitattu 16.1.2018 http://searchservervirtualization.techtarget.com/feature/A-brief-history-of-Docker-Containers-overnight-success

McCarty S 2018 A Practical Introduction to Container Terminology Viitattu 20.1.2018 https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/

Mesosphere Health Checks and Task Termination Viitattu 22.2.2018 https://mesosphere.github.io/marathon/docs/health-checks.html

Mesosphere Service Discovery & Load Balancing Viitattu 20.2.2018
https://mesosphere.github.io/marathon/docs/service-discovery-load-balancing.html

Mesosphere Pods Viitattu 24.2.2018
https://mesosphere.github.io/marathon/docs/pods.html

Microsoft Choosing Between .NET Core and .NET Framework for Docker Containers Viitattu 6.3.2018 https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/net-core-net-framework-containers/index

Microsoft Decision table: .NET Framework to use for Docker Viitattu 6.3.2018
https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/net-core-net-framework-containers/container-framework-choice-factors

Microsoft General guidance Viitattu 6.3.2018 https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/net-core-net-framework-containers/general-guidance

Microsoft 2017 Docker Terminology Viitattu 20.1.2018
https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/container-docker-introduction/docker-terminology

Novoseltseva E 2017 Top 10 Benefits you will get by using Docker Viitattu 8.2.2018
https://apiumhub.com/tech-blog-barcelona/top-benefits-using-docker/

Parthasarathy Akshai 2017 Kubernetes vs Mesos + Marathon Viitattu 22.2.2018
https://platform9.com/blog/kubernetes-vs-mesos-marathon/

RedHat What´s a Linux container Viitattu 16.1.2018
https://www.redhat.com/en/topics/containers/whats-a-linux-container

RedHat OpeShift Overview Viitattu 2.3.2018 https://docs.openshift.com/container-platform/3.7/getting_started/index.html

RedHat OpehShift Pods and Services Viitattu 2.3.2018
https://docs.openshift.com/container-platform/3.7/architecture/core_concepts/pods_and_services.html

RedHat OpenShift Persistent Storage Vittattu 3.3.2018
https://docs.openshift.com/container-platform/3.7/architecture/additional_concepts/storage.html

RedHat OpenShift OpenShift SDN Viitattu 3.3.2018
https://docs.openshift.com/container-platform/3.7/architecture/networking/sdn.html

Rubens P 2017 What are containers and why do you need them Viitattu 16.1.2018
https://www.cio.com/article/2924995/software/what-are-containers-and-why-do-you-need-them.html

Kuvalähteet

Kuva 1 Pivotal Viitattu 16.1.2018
https://uberflip.cdntwrk.com/files/aHViPTYzOTc1JmNtZD1pdGVtZWRpdG9yaW1hZ2UmZmlsZW5hbWU9aXRlbWVkaXRvcmltYWdlXzU4NDIxZjg5ODViNDMucG5nJnZlcnNpb249MDAwMCZzaWc9Y2UwMTMwZDkxODVmMzJiOWE3ZDZjMjU1YWFjY2U0NmU%253D

Kuva 2 Viitattu 20.1.2018 https://cdn-images-1.medium.com/max/2000/0*g3sV5OiLtYMfEnzJ

Kuva 3 Docker Inc. Viitattu 3.2.2018 https://docs.docker.com/engine/images/engine-components-flow.png)

Kuva 4 Docker Inc. Viitattu 8.2.2018

Kuva 5 Docker Inc Viitattu 8.2.2018

https://www.docker.com/sites/default/files/Container-App-Lifecycle-4.png

Kuva 6 Docker Inc Viitattu 10.2.2018
https://www.docker.com/sites/default/files/Enterprise_management_EE.png

Kuva 7 Docker Inc. Viitattu 10.2018
https://www.docker.com/sites/default/files/sessc_0_1.png

Kuva 8 Docker Inc Viitattu 12.2.2018 https://docs.docker.com/docker-for-mac/images/menu-prefs-selected.png

Kuva 9 Docker Inc Viitattu 12.2.2018 https://blog.docker.com/wp-content/uploads/docker_cloud_dashboard.png

Kuva 10 Amazon AWS Viitattu 10.2.2018 https://s3.amazonaws.com/chrisb/CW-ECS-diagram.png

Kuva 11 GitHub Viitattu 14.2.2018 https://user-images.githubusercontent.com/1712635/30402804-d62632a2-9893-11e7-817a-f9f616cdf380.png

Kuva 12 Docker Inc Viitattu 18.2.2018 https://docs.docker.com/docker-for-ibm-cloud/faqs/#what-ibm-cloud-infrastructure-permissions-do-i-need

Kuva 13 IBM Viitattu 18.2.2018 https://www.ibm.com/blogs/bluemix/wp-content/uploads/2015/06/StoreApplication.png

Kuva 14 Kubernetes Viitattu 22.2.2018

Kuva 15 Platform9 Viitattu 22.2018 https://platform9.com/wp-content/uploads/2017/07/mesos-architecture-1024x731.png

Kuva 16 Docker Inc Viitattu 25.2.2018
(https://docs.docker.com/engine/swarm/images/swarm-diagram.png

Kuva 17 Docker Inc 25.2.2018
https://docs.docker.com/engine/swarm/images/ingress-lb.png

Kuva 18 RedHat Viitattu 25.2.2018 https://hostadvice.com/wp-content/uploads/2017/05/rhelos.jpg

Kuvat 19-28 ovat omia otoksia.