jamk.fi

PaaS-alustojen vertailu palveluntar-jontaan

Tuomas Suvela

Opinnäytetyö
Huhtikuu 2017
Tekniikan ja liikenteen ala
Insinööri (AMK), Tietotekniikan tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Suvela, Tuomas	Opinnäytetyö, AMK	Kuukausi 2017
	Sivumäärä 61	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
PaaS-alustojen vertailu palveluntarjontaan

Tutkinto-ohjelma
Tietotekniikan tutkinto-ohjelma

Työn ohjaaja(t)
Rantonen Mika, Häkkinen Antti

Toimeksiantaja(t)
Telia Finland Oyj

Tiivistelmä

Opinnäytetyön toimeksiantajana oli Telia Finland. Se on suomalainen tietoliikenneoperaattori, joka tarjoaa kiinteän ja mobiilin verkon tietoliikenneratkaisuja sekä suunnittelee myös IaaS-pilvipalvelua. Yritysten tarpeiden muuttuessa Telialla oli tullut tarve alkaa suunnitella myös PaaS-alustan tarjoamista asiakkaille.

Työn tavoitteena oli vertailla eri PaaS-alustoja pilvipalveluntarjoajan näkökulmasta sekä tutkia yleensä konttiteknologian tuotantovalmiutta ja laskutusratkaisuja sekä eri pilvipalvelumallien käyttäjäkohderyhmiä. Alustan vaatimuksina olivat multitenanttisuus, integroitavuus OpenStackiin sekä luotettava kehitysyhteisö. Opinnäytetyö toteutettiin vertailemalla teoriatiedon ja tutkimusten perusteella OpenShiftiä, Rancheria, Cloud Foundryä ja OpenStack Magnumia. Tutkimuksen tuloksena OpenStack Magnum valikoitui Telian käyttöön sopivimmaksi.
Laskutusratkaisujen tutkiminen toteutettiin vertailemalla muiden pilvipalveluntarjoajien ratkaisuja. Tuloksena oli kolme vaihtoehtoa: itse tehty laskutusjärjestelmä, valmis laskutusjärjestelmä JBilling tai laskutus alla olevan IaaSin mukaan. Telian käyttöön soveltuisi parhaiten laskutus IaaSin mukaan.
Konttiteknologian tuotantovalmiuden tutkiminen toteutettiin vertailemalla muita tutkimuksia. Tuloksena oli, että tuotantokäyttöön liittyviin haasteisiin on kehitetty ratkaisuja, joita yritykset hyödyntävät kasvavassa määrin.

Telia voi hyödyntää opinnäytetyön tutkimuksia valitessaan PaaS-alustaa ja siihen sopivaa laskutusratkaisua.

Avainsanat (asiasanat)
Docker, Kontti, Kubernetes, PaaS, Pilvipalvelu

Muut tiedot
-

jamk.fi

Description

Author(s)	Type of publication	Date
Suvela, Tuomas	Bachelor's thesis	Month Year
		Language of publication:
	Number of pages	Permission for web publication: x

Title of publication

Comparison of PaaS platforms for providing cloud services

Degree programme

Information Technology

Supervisor(s)

Rantonen Mika, Häkkinen Antti

Assigned by

Telia Finland Oyj

Abstract

The bachelor's thesis was assigned by Telia Finland. Telia Finland is a Finnish Internet service provider offering fixed and mobile broadband services and also planning infrastructure as a service (IaaS). As the needs of the companies have been changing, Telia needed to begin the planning of a platform as a service (PaaS) for their customers.

The aim of the thesis was to compare four different PaaS platforms from the cloud service provider's perspective and examine the maturity and billing solutions of container technology as well as the target groups of different cloud service models. The requirements for the platform were multitenancy, integration to OpenStack and a reliable development community. Based on the theory and surveys, the research was carried out by comparing OpenStack Magnum, OpenShift, Cloud Foundry and Rancher. As a result, OpenStack Magnum was selected as the most suitable platform for Telia.
The research of the billing solutions was carried out by learning from the example of other cloud service providers. As a result, there were three options: a self-made billing system, ready-made billing system JBilling, or to charge in accordance with the underlying IaaS. As a result, the most suitable option for Telia was to charge in accordance with the underlying IaaS.
The research of the maturity of the container technology was carried out by examining other surveys. The results revealed that the solutions have been developed for the challenges of the production use of the container technology.

Telia can utilize this research when choosing a platform and a billing system for the PaaS -project.

Keywords/tags (subjectshttp://vesa.lib.helsinki.fi/)

Cloud service, Container, Docker, Kubernetes, PaaS

Miscellaneous

-

Sisältö

Kuviot

Taulukot

Lyhenteet

API	Application Programming Interface
AMQP	Advanced Messaging Queue Protocol
AWS	Amazon Web Services
BBS	Bulletin Board System
CaaS	Containers as a Service
CI/CD	Continuous integration & continuous deployment
CNCF	Cloud Native Computing Foundation
CNI	Container Network Interface
COE	Container Orchestration Engine
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EAP	Enterprise Application Platform
EC2	Elastic Cloud Computing
ECS	EC2 Container Service
HTTPS	Hypertext Transfer Protocol Secure
IaaS	Infrastructure as a Service
IDS	Intrusion Detection System
LBaaS	Load Balancer as a Service
LDAP	Lightweight Directory Access Protocol
lmctfy	Let Me Contain That For You
NIST	National Institute of Standards and Technology
OCI	Open Container Initiative
OVS	Open vSwitch
PaaS	Platform as a Service
QoS	Quality of Service
RBAC	Role-based access control
REST	Representational State Transfer
RHEL	Red Hat Enterprise Linux
SDN	Software defined network
SaaS	Software as a Service
SSH	Secure Shell
vCPU	Virtual Central Processing Unit
VPN	Virtual Private Network
VXLAN	Virtual eXtensive Local Area Network
YAML	YAML Ain't Markup Language

1 Työn lähtökohdat

1.1 Tausta

Yritykset ovat siirtäneet palveluitansa pilveen, koska oman palvelininfrastruktuurin ylläpitäminen on kallista ja työlästä. Tähän on reagoitu Telialla alkamalla suunnitella Telia Cloud pilvipalvelua. Se tulee pohjautumaan OpenStackiin ja se tuotetaan Telian omista Suomessa sijaitsevista konesaleista. (Vie liiketoimintasi kehitys pilveen 2017.)

OpenStack on malliltaan Infrastructure as a Service (IaaS) ja se vastaa useimpien yritysten tarpeita tällä hetkellä. Toisaalta yhä useampi yritys, erityisesti ohjelmistoyritykset, ovat siirtymässä ketterän ohjelmistokehtyksen malliin, joka muuttaa heidän tarpeitaan. Myöskin sovelluksien arkkitehtuuri on muuttumassa monoliittisista sovelluksista mikropalveluihin, joka usein vaatii sovelluksen alustan täydellisen uudistuksen. Tällaisiin tarpeisiin sopii paremmin ylemmän hierarkiatason alusta, joka mahdollistaa nopeamman sovelluksien käyttöönoton ilman niiden vaatimien alustojen pystyttämistä ja ylläpitämistä. Tällainen pilvipalvelumalli on Platform as a Service (PaaS).

1.2 Vaatimukset ja tavoitteet

Opinnäytetyön aihe oli neljän eri PaaS-alustan vertailu, joita olivat OpenShift, Cloud Foundry, Rancher ja OpenStack Magnum. Toimeksiantaja vaatimukset alustalle olivat:

- Luotettava kehitysyhteisö ja tuki
- Multitenanttisuus eli käyttäjien eristettävyys toisistaan
- Integroitavuus OpenStackiin

Lisäksi työn aiheena oli seuraavien asioiden tutkiminen pilvipalveluntarjontaan ja tuotantokäyttöön liittyen:

- Eri pilvipalvelumallien käyttäjäkohderyhmät
- Konttiteknologian tuotantovalmius
- PaaS-pilvipalvelumallin laskutusratkaisut

Opinnäytetyön tavoitteena oli selvittää paras PaaS-alusta pilvipalveluntarjoajan näkökulmasta ja tälle sopiva laskutusratkaisu. Lisäksi tavoitteena oli saada käsitys konttiteknologian tuotantovalmiudesta.

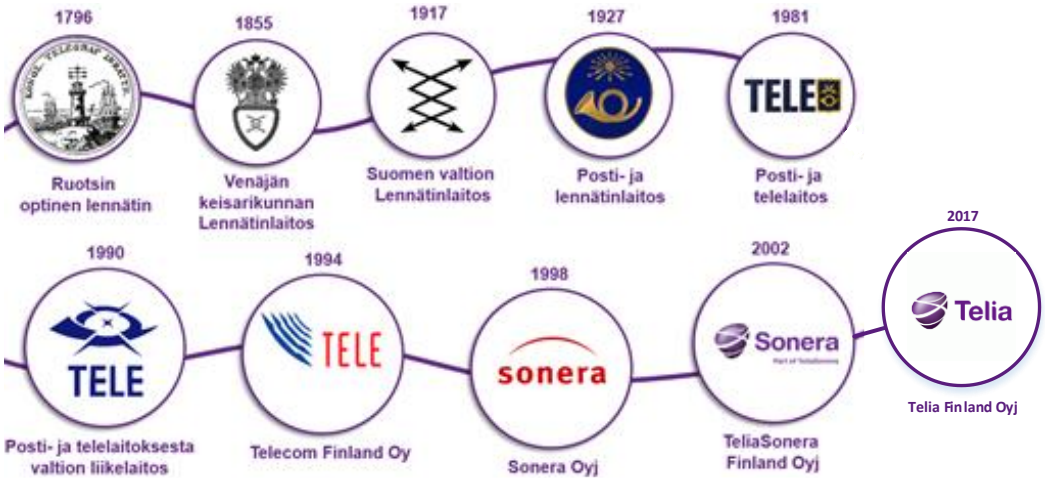
1.3 Tutkimusmenetelmä

Opinnäytetyö on rakenteeltaan kvalitatiivinen eli laadullinen tutkimus, jossa vertaillaan PaaS-alustoja pilvipalveluntarjoajan näkökulmasta ja tutkitaan niille eri laskutusratkaisuja sekä tutkitaan yleisesti konttiteknologian tuotantovalmiutta ja eri pilvipalvelumallien käyttäjäkohderyhmiä. Teoriatietoa analysoidaan Telia Cloud -projektin tarpeiden lähtökohdista. Tutkimuksen päämääränä on valita paras alusta Telia Cloud -projektia varten. Alustan käyttöönottoa ei käsitellä opinnäytetyössä projektin pitkäkestoisuuden vuoksi.

1.4 Toimeksiantaja

Toimeksiantajana oli Telia Finland, joka on Telia Companyn tytäryhtiö ja Suomen maaorganisaatio. Sen juuret ulottuvat jopa 1700-luvulle asti. Vuonna 2002 Sonera fuusioitui Telian kanssa, jolloin siitä tuli Pohjoismaiden suurin operaattori TeliaSonera (Komissio hyväksyy Telian ja Soneran fuusion tietyin ehdoin 2002). TeliaSonera muutti nimensä Teliaksi 23.3.2017 (Sonera ja Tele Finland ovat nyt Telia – Mitä se tarkoittaa? n.d).

Kuviossa 1 näkyy yhtiön historia Ruotsin optisesta lennättimestä aina nykyiseen Telia Finlandiin asti.



Kuvio 1. Telian historia (alkup. kuvio ks. Kekäläinen 2015)

Telia Finland tarjoaa sekä kiinteän että mobiilin verkon tietoliikennepalveluja yrityksille ja kuluttajille. Sen liikevaihto vuonna 2015 oli n. 1,3 mrd €, ja se työllistää n. 3000 henkilöä (Kekäläinen 2015).

Telia Finland on alkanut suunnitella myös IaaS-pilvipalvelun sekä konesalipalveluiden tarjoamista. Se rakentaa Helsingin Pitäjänmäkeen uutta konesalia, jonka on määrä valmistua 2017 loppuvuodesta. Valmistuessaan se tulee olemaan Suomen suurin avoin konesali, eli kuka tahansa voi ostaa sieltä palveluita. (Konesalien kuningas nousee Pitäjänmäkeen n.d).

2 Pilvipalvelu

2.1 Yleistä

Pilvi käsitteenä tietotekniikassa kuvaa monien tekniikoiden ja palveluiden yhdessä muodostamaa järjestelmää, jonka sisältö on piilotettu käyttäjältä ja sitä tarjotaan yhtenä isona palveluna. Tässä tapauksessa se voi tarkoittaa esim. resurssien käyttöönottoa tai käyttöjärjestelmän asentamista.

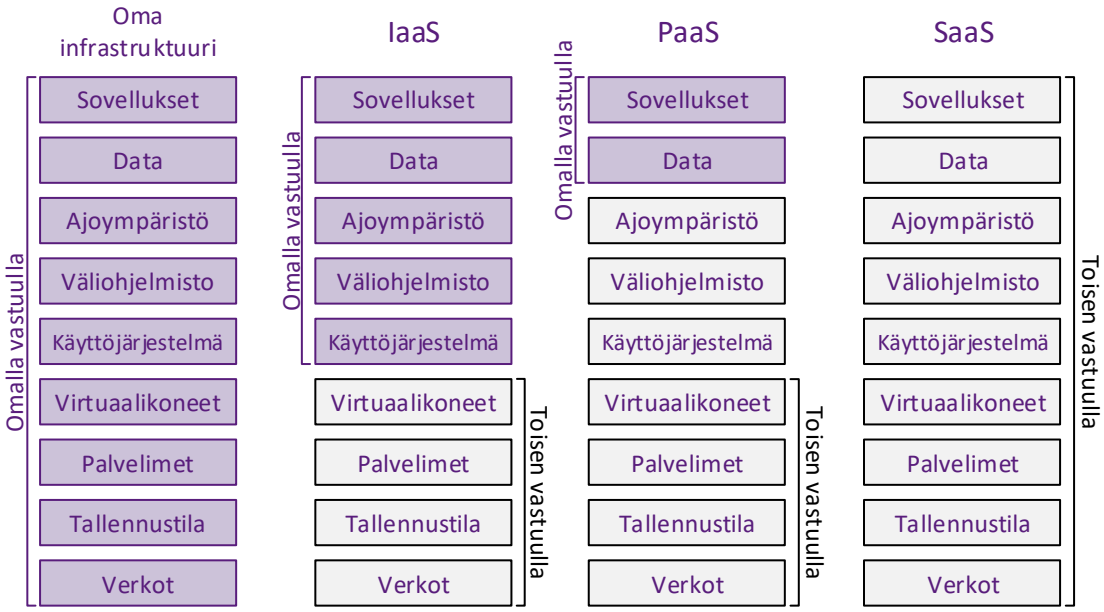
Pilvipalvelun määrittely on hankalaa, koska sen käyttötarkoitukset muuttuvat jatkuvasti. NIST (National Institute of Standards and Technology) on tehnyt muodollisen määrityksen pilvipalvelulle, mutta sitäkin on jouduttu muuttamaan ajan saatossa. Samat perusperiaatteet siinä ovat silti pysyneet:

- Pitkälle automatisoitu itsepalvelu, eli käyttäjä voi pyytää käyttöönsä lisää resursseja (esim. verkotusta, palvelimia, tallennustilaa tai ohjelmia) ilman pyynnön manuaalista käsittelyä
- Palveluiden on oltava saatavilla internetin ylitse ja toimittava hitaammallakin yhteydellä. Niiden käyttö ei pitäisi vaatia client-ohjelmistoa tai ainakin sen on oltava kevyt ja toimittava useimmilla päätelaitteilla.
- Virtualisoitu resurssipooli, joka skaalautuu nopeasti käytön mukaan. Eli palvelimet ovat jo olemassa, mutta ne käynnistyvät automaattisesti kuorman kasvaessa.
- Palveluita on pystyttävä valvomaan eri mittareilla (esim. tallennustila, prosessointi, kaistankäyttö). Tämä mahdollistaa myös veloituksen käyttäjältä kulutuksen mukaan.

(Rountree & Castrillo 2014.)

2.2 Pilvipalvelumallit

NIST on myös määritellyt palvelumallit, jotka erottelevat palvelut kolmelle eri tasolle. Mallit määrittävät, mitä palvelua tarjotaan, mutta samalla myös kenen vastuulla mikin osa on. Kuviossa 2 on havainnollistettu vastuualueet eri malleissa.



Kuvio 2. Palvelumallien vastuut (alkup. kuvio ks. Greiner 2014)

SaaS (Software as a Service) tarkoittaa mallia, jossa käyttäjälle tarjotaan pääsy palveluntarjoajan itsensä ylläpitämään palveluun (esim. sähköpostin web-käyttöliittymä). Käyttäjällä ei ole pääsyä taustalla olevaan palvelimeen, käyttöjärjestelmään tai ohjelman lähdekoodiin. (Mell & Grance 2011.)

PaaS (Platform as a Service) on alusta, jolle käyttäjä voi itse asentaa ohjelmia ja hallita niitä, mutta käyttäjällä ei silti ole pääsyä käyttöjärjestelmään, jonka päällä ohjelmia ajetaan (Mell & Grance 2011).

IaaS (Infrastructure as a Service) on näistä kolmesta alin taso. Tässä palvelumallissa käyttäjän on mahdollista provisioida esim. prosessointitehoa (yleensä suoritinytimiä tai -aikaa), tallennustilaa ja verkkoja omaan ympäristöönsä. Käyttäjä voi siis asentaa oman käyttöjärjestelmän ja ajaa sen päällä mitä tahansa. (Mell & Grance 2011.)

2.3 Infrastruktuurimallit

Palvelumallien lisäksi pilvipalvelut on eroteltu neljään kategoriaan niiden infrastruktuurin perusteella: yksityinen pilvi, yhteisöpilvi, julkinen pilvi ja hybridipilvi.

Yksityinen pilvi on sellainen, jonka koko infrastruktuuri (palvelimet, tallennustila, verkot jne.) on käytössä ainoastaan yhdellä organisaatiolla. Palveluntarjoaja voi olla itse organisaatio tai kolmas osapuoli. (Mell & Grance 2011.)

Yhteisöpilvi on muuten samalainen, mutta organisaation sijasta sitä käyttää yhteisö, johon kuuluu useita tahoja (Mell & Grance 2011).

Julkinen pilvi on nimensä mukaisesti julkiseen käyttöön tarkoitettu, eikä sen käyttäjäkuntaa ole rajattu mitenkään. Sama resurssipooli on jaettu kaikkien käyttäjien kesken. (Mell & Grance 2011.)

Hybridipilvi on yksityisen ja julkisen pilven yhdistelmä, jossa sovelluksien kuormaa voidaan ohjata näiden välillä (kuormantasaus) (Mell & Grance 2011).

3 OpenStack

3.1 Yleistä

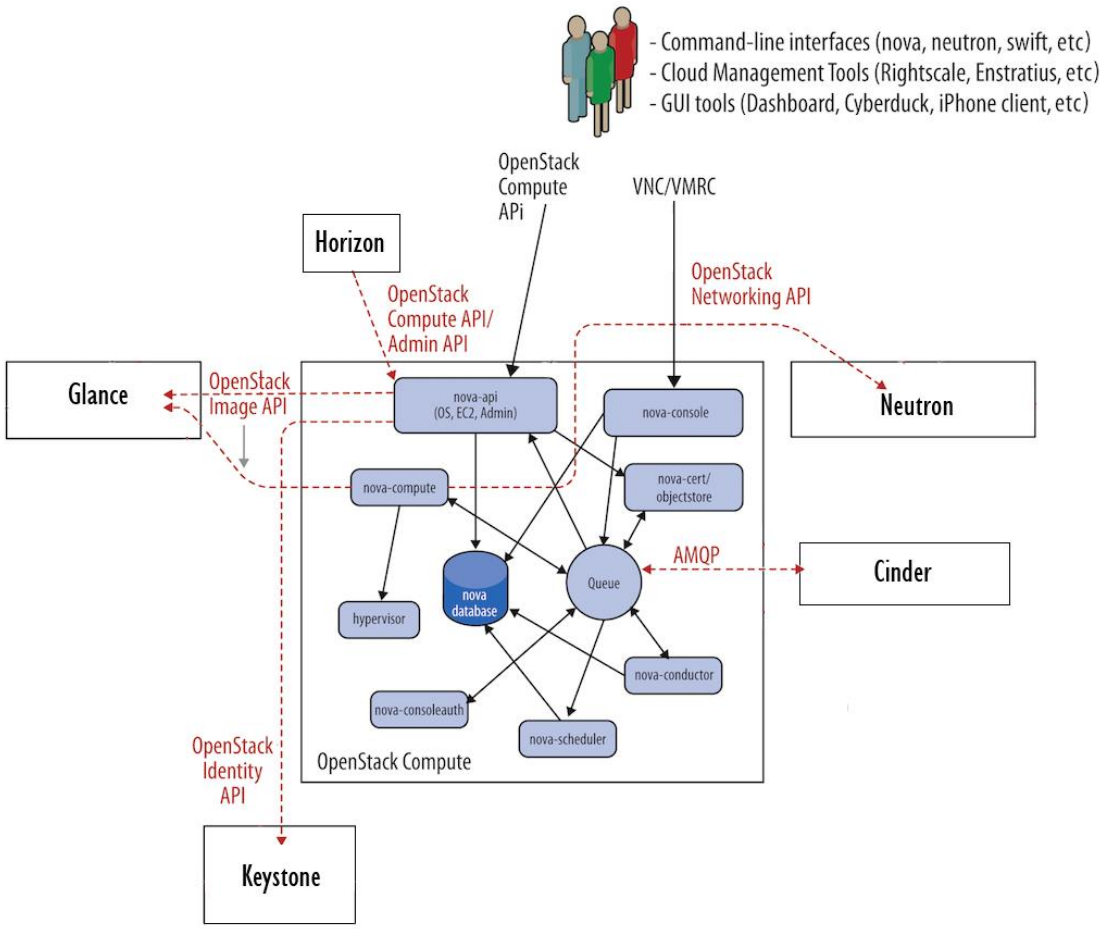
OpenStack on avoimeen lähdekoodiin perustuva IaaS-alusta, joka lähti liikkeelle Rackspacen ja NASAn yhteistyöprojektista. Se on kokoelma erilaisia komponentteja, jotka yhdessä muodostavat tehokkaan ja laajennettavan ympäristön. Ydinkomponentit, joista OpenStack koostuu, ovat Nova, Glance, Swift, Keystone, Neutron, Cinder ja Horizon. (Smith. n.d.)

Liitteessä 1 on havainnollistettu koko OpenStackin arkkitehtuuri ja se selitetään auki seuraavissa kappaleissa.

3.2 Nova

Nova on koko OpenStackin tärkein komponentti. Se vastaa virtuaalikoneiden pystyttämisestä ja eri resurssien kuten verkkojen ja levyjen liittämisestä niihin. Verkotuksen

se pyytää Neutronilta Networking APIn avulla ja levyt Cinderiltä AMQP:llä (Advanced Message Queuing Protocol). (Smith. n.d.)



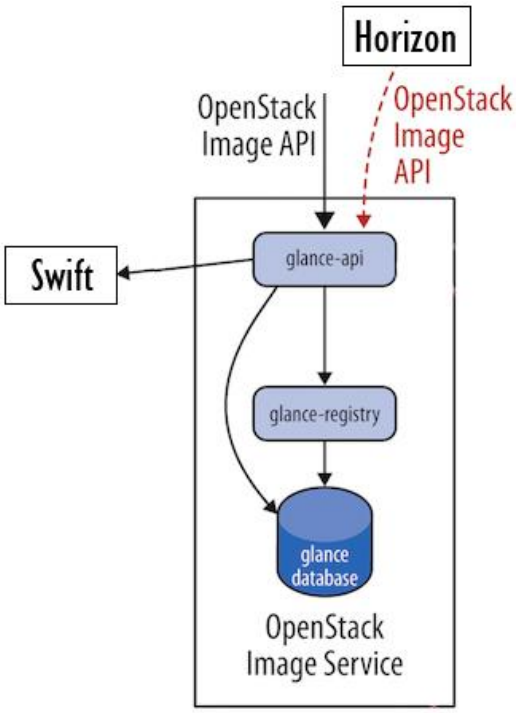
Kuvio 3. Nova komponentit (alkup. kuvio ks. Architecture 2017.)

Novan prosessit kuviosta 3

- nova-api: APIn (Application Programming Interface) kautta käyttäjät hallitsevat Novaa. API terminä tarkoittaa tapaa, jolla ohjelma pyytää jotain tai vastaa toiselle ohjelmalle.
- nova-compute: taustaprosessi, joka luo tai poistaa virtuaalikoneita kommunikoimalla hypervisoreiden APIen kanssa.
- AMQP: tarkoitettu komponenttien väliseen kommunikointiin samoin kuin APIt. Nova-api pyytää AMQP:n kautta virtuaalikoneille levyvolyymejä Cinderiltä.
- Networking API: Nova pyytää verkotuksen virtuaalikoneelle Neutronilta Networking APIn avulla.
- nova-scheduler: käsittelee virtuaalikoneiden pyyntöjä ja määrittää millä isäntäkoneella niitä voidaan ajaa.
- nova database: tallennetaan tilatiedot: esim. onko virtuaalikone käytössä ja mitä verkkoja on käytössä. (Smith. n.d.)

3.3 Glance

Glance on levykuvavarasto, johon voidaan varastoida virtuaalikoneiden templaatteja ja varmuuskopioita. Glance ainoastaan hallinnoi levykuvia ja tarjoaa niitä uusiin virtuaalikoneisiin niitä käyttöönotettaessa. Itse data säilötään OpenStack Swiftiin tai Cinderiin. Glancea hallitaan APIn avulla.



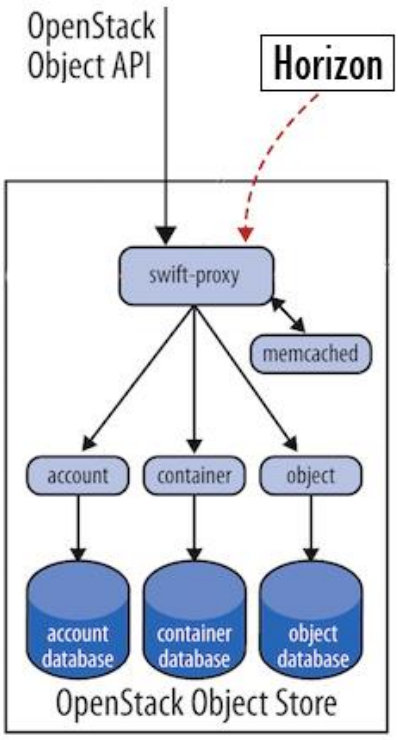
Kuvio 4. Glancen arkkitehtuuri (alkup. kuvio ks. Architecture 2017.)

Kuviossa 4 näkyvät Glancen prosessit ja niiden vastuualueet:

- glance-api: vastaa käyttäjien ja muiden komponenttien API-pyyntöihin
- glance-registry: tallentaa, muokkaa ja hakee meta-tietoja levykuvista
- glance database: meta-tietojen tallennuspaikka
- Swift: levykuvien tallennuspaikka. (Smith. n.d.)

3.4 Swift

Swift on OpenStackin tallennustilapalvelu. Se pystyy käsittelemään suurta määrää dataa hajautettuna useille eri tallennuslaitteille. Tämä takaa datan redundanttisuuden, eli levyn hajotessa data replikoidaan joltain muulta isäntäkoneelta. Sen tallennustilaa voidaan kasvattaa yksinkertaisesti levyjä lisäämällä, jotka se osaa liittää sen hetkiseen levyklusteriin. Swiftiä hallitaan APIn avulla. (Wang 2015.)



Kuvio 5. Swift arkkitehtuuri (alkup. kuvio ks. Architecture 2017.)

Kuviossa 5 on havainnollistettu Swiftin arkkitehtuuri. Swift-proxy käsittelee käyttäjien ja muiden palveluiden API-pyynnöt. Object-palvelin säilöö tiedostot. Objektin polku määräytyy sen nimen tiivisteestä (hash) ja sitä koskeneen toimenpiteen aikaleimasta. Container-palvelin pitää yllä listaa objekteista. Objektit jaotellaan säiliöihin (voi verrata kansioihin). Account-palvelin puolestaan pitää yllä listaa säiliöistä ja käyttäjien oikeuksista niihin. (Wang 2015.)

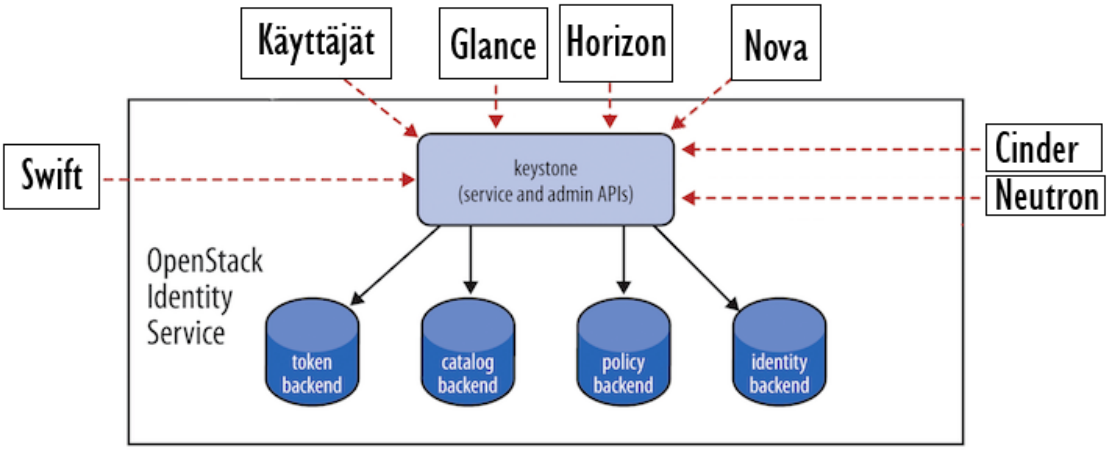
3.5 Keystone

Keystone on OpenStackin todennus- ja valtuutuspalvelu, joka tarjoaa keskitetyn käyttäjähakemiston, jossa heidän oikeutensa kuhunkin OpenStack palveluun määritetään. Keystone on myös mahdollista liittää ulkoisiin palveluihin, kuten LDAPiin, oAuth, SAML ja OpenID. Keystonen toimintoja ovat

- Käyttäjien todennus ja sen myötä tokenin tarjoaminen käyttäjälle. Tokenin avulla käyttäjä pystyy todentamaan itsensä OpenStackin eri palveluille
- Käyttäjien valtuutus RBAC-menetelmällä (Role-based access control), jossa oikeudet määritellään rooliperustaisesti, eikä suoraan käyttäjille
- Luettelon tarjoaminen käytössä olevista OpenStackin palveluista (Smith. n.d).

Keystonen RBAC-menetelmä koostuu projekteista, ryhmistä ja käyttäjistä. Projektit ovat ”tenantteja”, jotka ovat esim. samassa julkisessa pilvessä toimivia eri organisaatioita. Ryhmät ovat projektin sisäisiä ryhmiä käyttöoikeuksien määrittelyä varten. Nämä voivat olla esim. organisaation eri osastoja, joihin käyttäjät kuuluvat. Käyttäjät pääsevät tokenin avulla käsiksi vain heille tarkoitettuihin palveluihin. Näin ollen palveluiden ei tarvitse tarkistaa käyttäjien tunnuksia jatkuvasti, vaan pelkästään tokenin tarkistaminen riittää. (Smith. n.d.)

Kuviossa 6 on havainnollistettu Keystonen arkkitehtuuri ja komponentit joiden kanssa se kommunikoi.

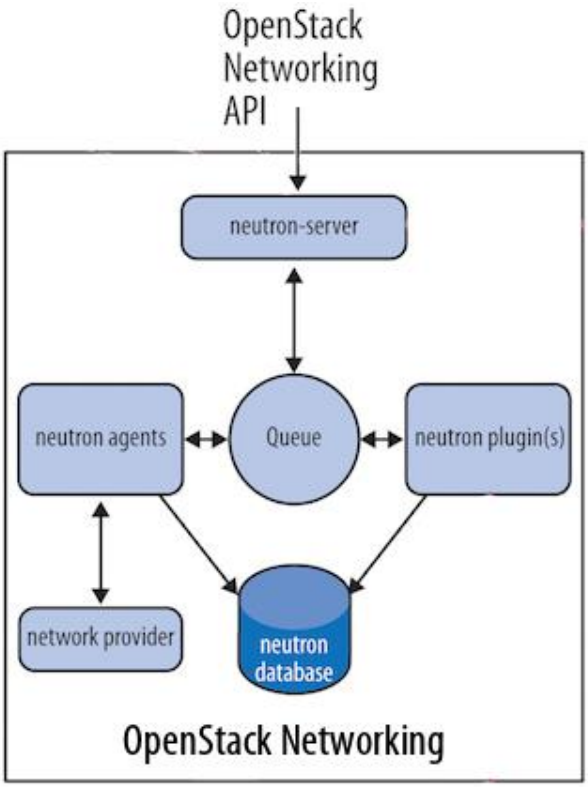


Kuvio 6 Keystone arkkitehtuuri (alkup. kuvio ks. Architecture 2017.)

3.6 Neutron

Neutron on OpenStackin SDN-palvelu (Software Defined Network), eli se vastaa virtuaalikoneiden verkoista. Sen avulla käyttäjät tai muut OpenStackin palvelut voivat luoda verkkoja ja liittää niihin virtuaalikoneita. Neutronilla on mahdollista tehdä perinteisten verkkojen lisäksi myös edistyneempiä toimia, kuten L2-in-L3 tunneleita ja QoSia (Quality of Service). Neutronin avulla voidaan rakentaa myös muita verkkopalveluita kuten kuormantasaus (LBaaS), VPN-palvelu (VPNaas) tai palomuuri (FWaas). (Neutron 2014.)

Kuviossa 7 on havainnollistettu Neutronin arkkitehtuuri.



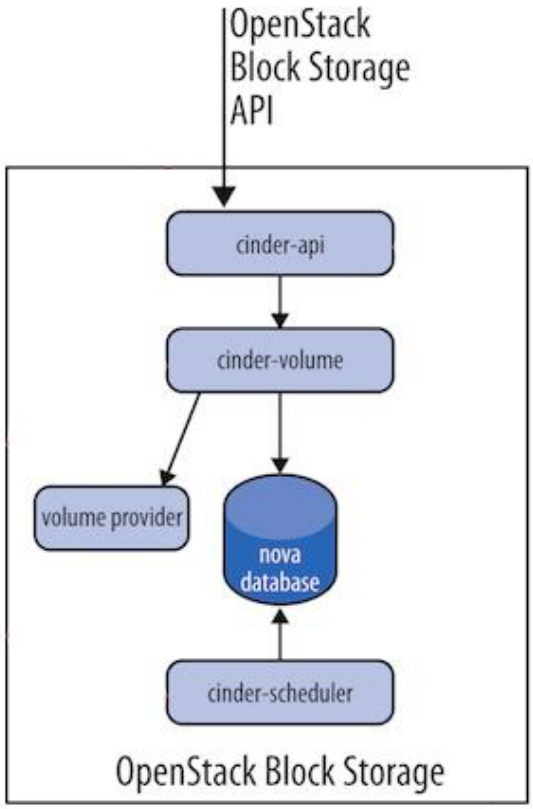
Kuvio 7 OpenStack Neutron arkkitehtuuri (alkup. kuvio ks. Architecture 2017.)

3.7 Cinder

Cinder on OpenStackin lohkotallennuspalvelu. Se vastaa virtuaalikoneiden tallennustilasta yhdistämällä niihin levyvolyymeitä, joihin voidaan sitten alustaa virtuaalikoneissa haluttu tiedostojärjestelmä. Cinderin keskeisimpiä ominaisuuksia ovat

- Volyymien luominen, kloonaaminen, poistaminen sekä niiden liittäminen virtuaalikoneisiin
- Snapshotien muodostaminen, joka tarkoittaa järjestelmän sen hetkistä tilaa tallennettuna levykuvaksi
- Volyymien statistiikkojen tarkastelu. (Smith. n.d.)

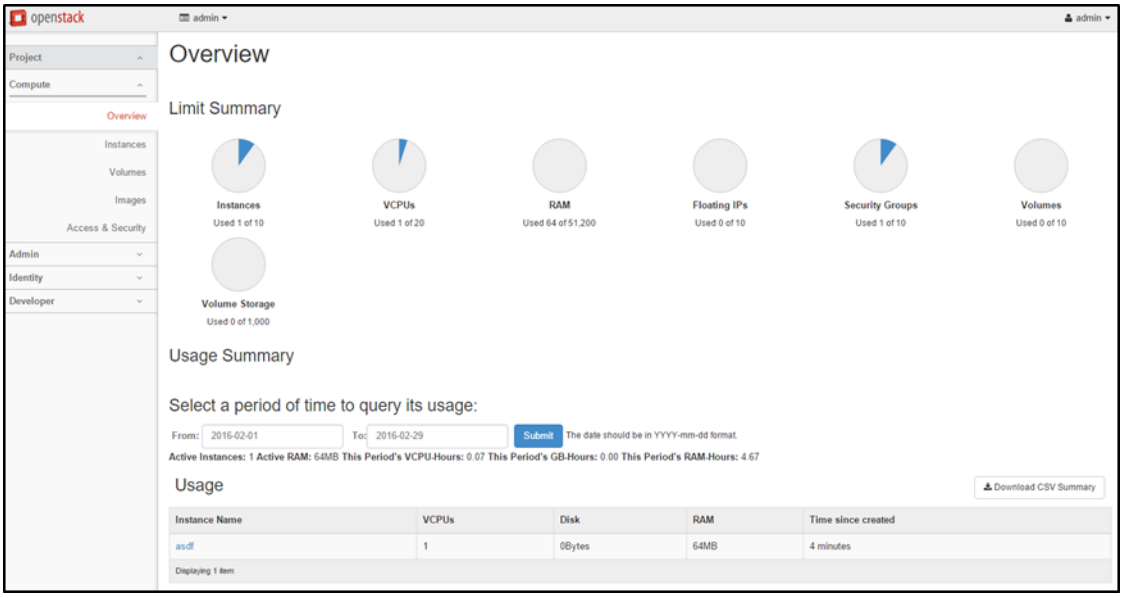
Kuviossa 8 on havainnollistettu Cinderin arkkitehtuuri.



Kuvio 8 Cinderin arkkitehtuuri (alkup. kuvio ks. Architecture 2017.)

3.8 Horizon

Horizon on OpenStackin hallintaportaali, jonka kautta on mahdollista hallita kaikkia OpenStackin komponentteja. Horizon on tilaton web-sovellus, eli se lähettää API-pyyntöjä OpenStackin eri komponenteille ja näyttää niiden vastaukset käyttäjälle. Sillä on mahdollista suorittaa kaikki perustoimet OpenStackissa kuten luoda ja poistaa virtuaalikoneita, verkkoja ja levykuvia. (Smith. n.d.)



Kuvio 9. Horizon hallintaportaali

Horizonin navigointi näyttää sisäänkirjautuneelle käyttäjälle vain sen käyttöoikeuksien sallimat paneelit. Kuviossa 9 on Horizonin käyttöliittymä, jossa on näkyvissä kaikki paneelit, koska sinne on kirjauduttu sisään admin-tunnuksilla. (Smith. n.d.)

3.9 Heat

Heat ei ole OpenStackin ydinkomponentti, joten se ei näy koko OpenStackin arkkitehtuuria havainnollistavassa kuviossa (ks. Liite 1.). Heat on OpenStackin orkestrointipalvelu. Se mahdollistaa infrastruktuurin eri resurssien (verkko, virtuaalikoneet, levyvolyymit jne.) luomisen ja konfiguroinnin automatisoidusti. Heat konfiguroidaan yleensä YAML-templaatin (YAML Ain't Markup Language) avulla, jossa kuvaillaan sovelluksen tarvitsemat resurssit. Heat pystyttää resurssit OpenStackin eri API-komentojen avulla. Heatin pystyttämiä resurssisettejä kutsutaan stackeiksi. (Villarreal 2017.)

YAML-templaatti koostuu seuraavista asioista:

- Versio
- Kuvaus
- Resurssit
- Parametrit
- Tuloste. (Villarreal 2017.)

Esimerkki yksinkertaisesta templaatista:

```
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute in-
stance

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: my_key
      image: ubuntu-trusty-x86_64
      flavor: m1.small
```

Yllä olevassa templaatissa on ainoastaan vähimmäismäärittelyt: versio, kuvaus ja resurssit (Heat Orchestration Template (HOT) specification n.d).

Version avulla Heat tunnistaa mitä rakennetta templaatti noudattaa ja mitä ominaisuuksia tuetaan. Kuvaus on vain käyttäjää itseään varten. (Heat Orchestration Template (HOT) specification n.d.)

Luotavia resursseja on ainoastaan yksi m1.small-kokoinen virtuaalikone, jossa on ajossa ubuntu-levykuva ja sille on luotu avainpari my_key (Heat Orchestration Template (HOT) specification n.d).

Parametrit ovat muuttujia, jotka odottavat syötettä templaatin suorittajalta. Tyypillisiä parametreja ovat käyttäjätunnukset, salasanat, levykuvat jne. (Heat Orchestration Template (HOT) specification n.d).

Tuloste palauttaa templaatin suorittajalle luotujen resurssien IP- tai URL-osoitteet (Heat Orchestration Template (HOT) specification n.d).

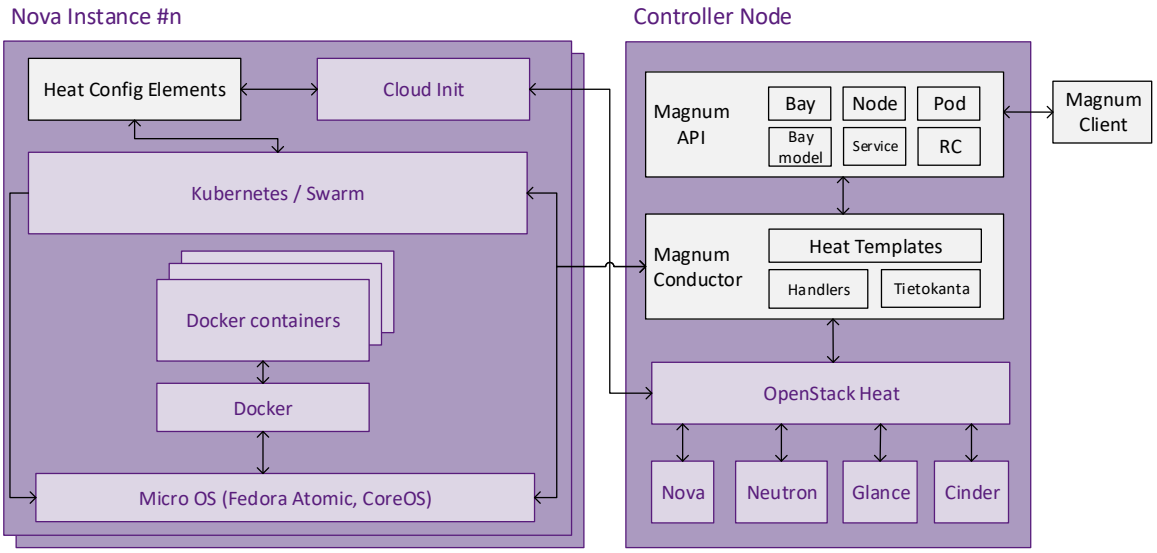
Heatilla on mahdollista autoskaalata ympäristöä sen kuorman mukaan. Tähän tarvitaan OpenStack Ceilometeriä, joka monitoroi ympäristöä ja liipaisee hälytyksen määritetyn raja-arvon ylityksestä. Heat voidaan määrittää seuraamaan tätä hälytystä ja luomaan/poistamaan resursseja, riippuen onko kuorma kasvava vai vähenevä. (Berendt 2015.)

3.10 Magnum

Magnum ei myöskään ole OpenStackin ydinkomponentti, joten sekään ei näy koko OpenStackin arkkitehtuuria havainnollistavassa kuviossa (ks. Liite 1.). Magnum on yksi vertailukohteista muiden PaaS-alustojen rinnalla.

Magnum tuo COEn (Container Orchestration Engine) OpenStackiin. Sen avulla voidaan luoda klusteri käyttäjän OpenStack-ympäristöön ja ajaa siellä kontteja. COEtä ja konttiteknologiaa yleisesti käsitellään tarkemmin kappaleessa 4. Magnum tukee kolmea eri COEtä: Kubernetesiä, Docker Swarmia ja Mesosia. Magnumia voidaan hallita suoraan APIlla tai OpenStackin hallintaportaalin (Horizon) kautta. Luotua klusteria hallitaan sen omalla natiivilla komentorivirajapinnalla. (Magnum n.d.)

Kuviossa 10 näkyy Magnumin arkkitehtuuri.



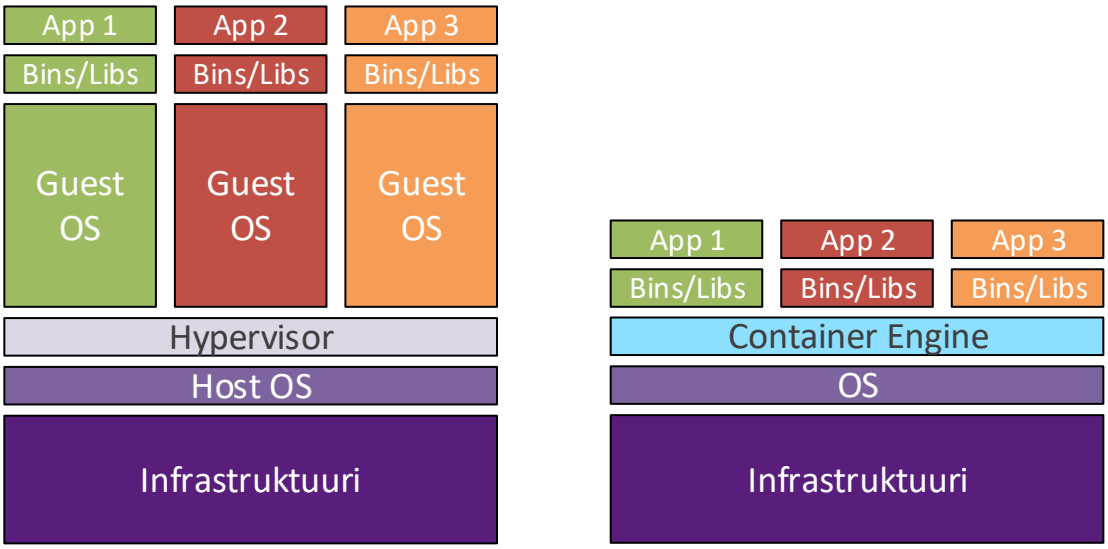
Kuvio 10. Magnum arkkitehtuuri (alkup. kuvio ks. Magnum n.d)

Bay tarkoittaa Magnumilla luotua klusteria, jossa voidaan ajaa kontteja. Baymodel on templaatti klusteria varten, jossa määritellään COE, avainpari ja nova-instansseilla ajettava levykuva. Magnumia hallitaan Magnum Client -asiakasohjelmalla, joka välittää komennot Magnum APIlle ja siitä edelleen Magnum Conductorille. Conductorilla on erilaisia Heat templaatteja, joilla voidaan luoda käyttäjän Baymodelin mukainen klusteri. OpenStack Heat luo kyseisen klusterin eri OpenStack-komponenttien avulla ja konfiguroi valitun COEn valmiiksi käytettäväksi. (Dake 2015.)

4 Konttiteknologia

4.1 Docker

Docker on käyttöjärjestelmätason virtualisointiohjelma, joka perustuu kontteihin. Sen pääasiallinen tehtävä on sovellusten paketointi ja niiden ajaminen eristettynä kontissa. Kontit ovat käyttöjärjestelmän prosesseja ja ne jakavat saman käyttöjärjestelmän ytimen, mutta käyttäjälle ne näyttävät erillisiltä virtuaalikoneilta. Perinteisessä, hypervisor-pohjaisessa virtualisoinnissa, kukin ohjelma tarvitsi oman virtuaalikoneen, jossa ajettiin omaa käyttöjärjestelmää. Kontti-pohjaisessa virtualisoinnissa ohjelmat eivät tarvitse erillistä käyttöjärjestelmää, vaan niissä ajetaan ainoastaan ohjelman vaatimia kirjastoja ja alla on kaikille yhteinen käyttöjärjestelmän ydin (kernel) (Ks. kuvio 11). (What is Linux container? n.d.)



Kuvio 11. Virtualisointi verrattuna konttiteknologiaan (alkup. kuvio ks. Wang 2016)

Ohjelmien ajaminen konteissa on paljon tehokkaampaa kuin erillisellä virtuaalikoneella. Kontti rakennetaan levykuvasta, joka sisältää itse ohjelman lisäksi myös sen kaikki riippuvuudet. Näin ollen ne ovat täysin siirreltäviä esim. testiympäristöstä tuotantoon. (What is Linux container? n.d)

Ensimmäinen konttiohjelma kehitettiin jo vuonna 2000. Se oli FreeBSD:n luomat vankilat (jails), jotka mahdollisti chroot-toiminto. Chroot muuttaa ohjelman juurihakemiston (root) osoittamaan omaan, eristettyyn hakemistoon. Näin ollen kyseinen ohjelma ei pääse itse käyttöjärjestelmät juureen. (Riondato n.d)

Konttiteknologiaan on sittemmin kehitetty monia parannuksia. Cgroups, joka rajoittaa konttien oikeuksia ytimen eri resursseihin ja muihin prosesseihin, ja näin ollen mahdollistaa prosessien eristämisen niin, etteivät ne näe toisiaan (namespace). Systemd-nspawn on chrootista edistyneempi versio. Se luo ohjelmalle virtuaalisen juurihakemiston lisäksi myös virtuaalisen tiedostojärjestelmän, prosessipuut, prosessien väliset kommunikointiväylät. Nämä toimet paransivat konttien eristyskykyä. (Petazzoni 2015.)

Sittemmin on syntynyt projekteja kuten Oracle Solariksen alueet (zones) ja Googlen lmctfy (Let me contain that for you), mutta ne eivät koskaan nousseet suureen suosioon ja näin ollen niiden kehitys lopetettiin. Vuonna 2013 julkaistiin Docker, joka nousi suureen suosioon ja siitä tuli konttien ”de facto”. (Tsidulko 2016.)

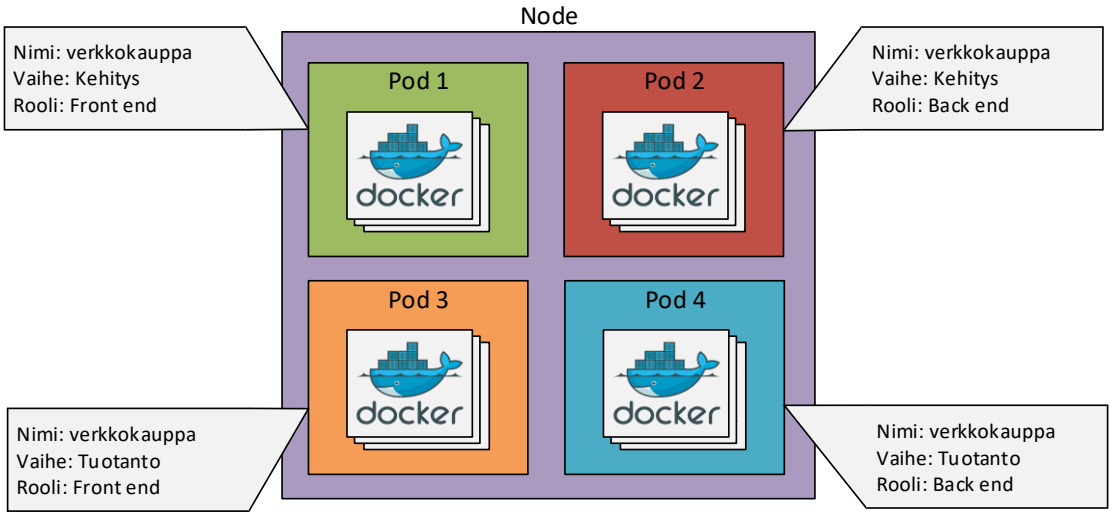
Ohjelman kontittamisen lisäksi, Docker tarjoaa niille verkon ja tallennustilan. Mikropalveluarkkitehtuurissa yksi sovellus muodostuu useista pienistä palveluista ja Dockerin avulla tämä on mahdollista toteuttaa. (Docker for the Virtualization Admin n.d.)

4.2 Kubernetes

Kubernetes on alun perin Googlen kehittämä orkestrointityökalu sovelluskonttien hallintaan palvelinklusterissa. Se helpottaa konttien suurien määrien hallintaa. Se osaa vastata automaattisesti sovelluksien tarpeisiin: skaalata niitä, replikoida toiselle nodelle tai yrittää uudelleenkäynnistää. Kubernetes yksinkertaistaa infrastruktuuritason: se abstrahoi palvelimet yhtenäiseksi resurssipooliksi, jossa kontteja voidaan ajaa millä tahansa palvelimella. Kubernetes käyttää konttiklusterissa ”isäntä-orja” -rakennetta, jossa isäntä, eli master vuorontaa kontteja orjille, eli nodeille ajettavaksi. Kubernetes ei ota kantaa, ovatko klusterin palvelimet fyysisiä vai virtuaalisia. (Braun 2016.)

Sovelluskontit erotellaan omiin ryhmiinsä (Pods). Samaan podiin kuuluvat kontit toimivat osana samaa sovellusta, esimerkiksi web-sovellus, jota Nginx ajaa omassa kontissaan ja Git synkronoi sen repositorya. Kontit jakavat saman tallennustilan, IP-osoitteen ja porttiavaruuden. Myöskin nimiavaruus on yhteinen konteilla, eli ne löytävät toisensa ”localhost”-nimellä ja voivat kommunikoida sen kautta. Kommunikointi onnistuu myös yhteisen muistiavaruuden kautta. (A Technical Overview of Kubernetes (CoreOS Fest 2015) 2015.)

Kuviossa 12 on havainnollistettu noden, podin ja konttien tasot toisiinsa nähden.



Kuvio 12. Kubernetesin node ja sen sisällä ajettavat podit (alkup. kuvio ks. Nadeeshani 2016)

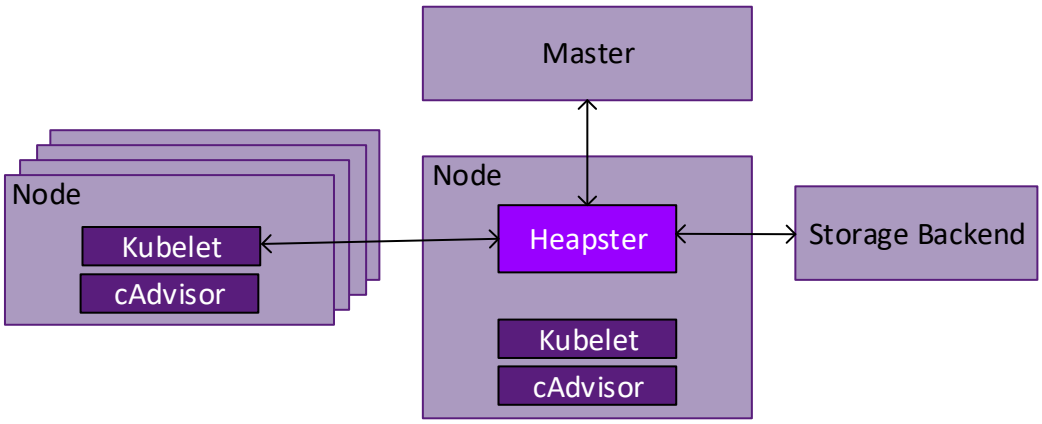
Labelit ovat avain-arvopareja, joita voidaan liittää objekteihin, kuten podeihin. Niiden avulla voidaan tunnistaa objektin rooli tai ominaisuus. Labeleiden avulla voidaan järjestellä tai valita objekteja, joille suoritetaan jokin toimenpide. Kuviossa 12 on merkitty labeleiden avulla sovelluksen nimi, elinkaaren vaihe ja rooli. (A Technical Overview of Kubernetes (CoreOS Fest 2015) 2015.)

ReplicationController vastaa siitä, että podien kopioita on aina oikea määrä ajossa. Se monistaa podeja, jos niitä on liian vähän, tai tappaa muutaman, jos niitä on liian paljon, eli skaalaa podeja sille annettujen määrittelyjen mukaan. ReplicationController on suunniteltu myös helpottamaan sovelluksien jatkuvaa päivittämistä siten, että se päivittää podit yksi kerrallaan uudempaan versioon. Tämä mahdollistaa sovelluksen palvelevan käyttäjiä ilman katkosta. (Replication Controller n.d)

Palvelua ajavaan podiin (back-end) tarvitsee usein ottaa yhteyttä etupään podeista (front-end), joten sille on annettava kiinteä IP-osoite tai nimi. Palvelupod merkataan service-labelillä, ja sille annetaan kiinteä ip-osoite (clusterIP). Kubernetes käyttää tähän nodella sijaitsevaa kube-proxy-komponenttia, joka asentaa palvelupodiin iptablesin, kuuntelee clusterIP:seen kohdistuvaa liikennettä ja ohjaa sen kyseisessä podissa ajettavalle palvelulle. Jos samalla service-labelillä merkittyjä podeja on useampi, liikennettä ohjataan kullekin satunnaisesti kuorman tasaamiseksi. (Services n.d)

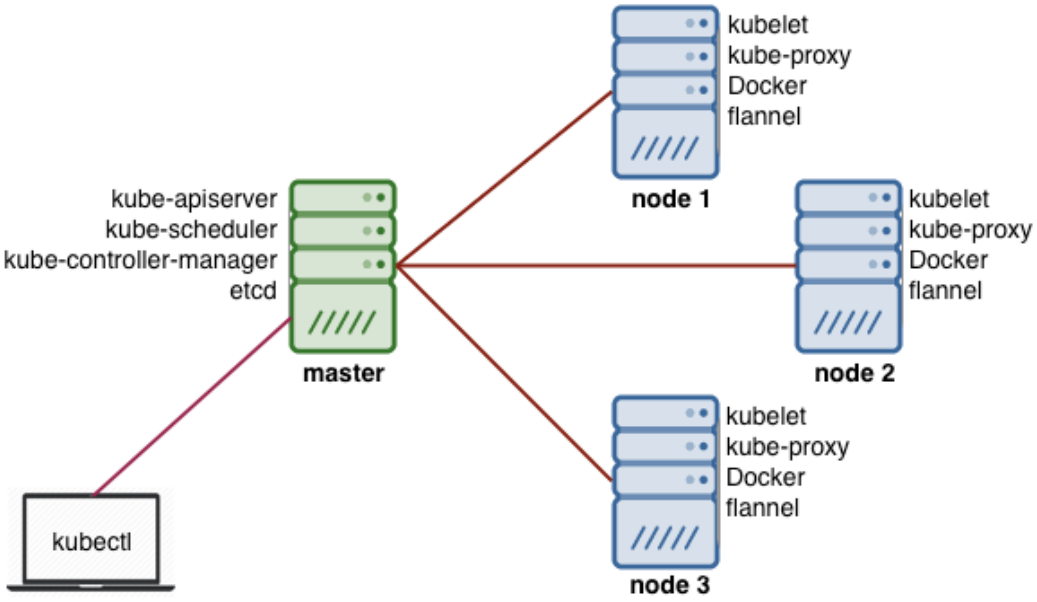
Kubelet on ajossa jokaisella nodella ja se pitää huolen, että kontit ovat PodSpecin määritysten mukaiset. PodSpec on YAML- tai JSON-objekti, jossa määritellään mm. kontin nimi, kontissa ajettava sovellus ja sovelluksen versio. PodSpecit eivät ole käyttäjien luomia vaan ne generoituvat pääasiallisesti Kubernetesin API-palvelimen kautta. (Francia 2016b.)

Osana kubeletiin kuuluu myös cAdvisor, joka kerää nodesta ja siellä ajossa olevista konteista statistiikkaa CPUn, muistin, tiedostojärjestelmän ja verkon käytöstä. Kubelet välittää kerätyt statistiikat eteenpäin Heapsterille, joka ryhmittelee datan podien ja labeleiden mukaan. Kerätty data ohjataan edelleen monitorointityökaluille, jotka visualisoivat ja varastoi datan. Esim. InfluxDB ja Grafana ovat suosittuja työkaluja monitorointiin. Kuviossa 13 on resurssienvalvonnan rakenne, jossa vasemmalla olevat nodet kuvaavat käyttäjien podeja ajavia nodeja, joista statistiikat lähetetään Heapsterille. Kuviossa näkyy myös, että Heapsteriä ajetaan omassa podissaan josta se välittää kerätyn datan klusterin masterille ja monitorointityökaluille (Storage Backend). (Kannan & Marmol 2015.)



Kuvio 13. Kubernetes resurssienvalvonnan rakenne (alkup. kuvio ks. Kannan & Marmol 2015)

Kuviossa 14 on koko Kubernetes-klusterin rakenne. Kuvion alla käydään läpi tarkemmin loput komponentit joita ei vielä ole käsitelty.

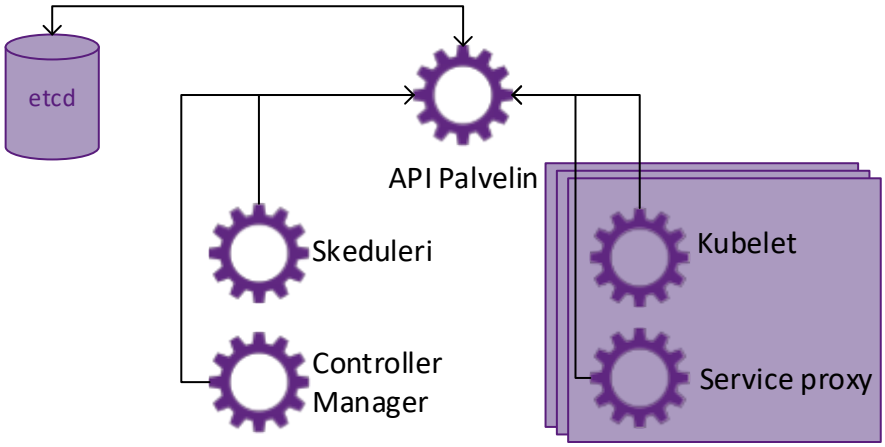


Kuvio 14. Kubernetes-klusterin rakenne (Braun 2016)

Kubectl on komentorivirajapinta Kubernetes-klusteriin. Sen avulla voidaan hallita ja tarkastella niin klusteria kuin myös siellä ajettavia podeja. (Kubectl Overview n.d.)

Etcd on avoimeen lähdekoodiin pohjautuva hajautettu avain-arvo-varasto (distributed key-value store), jota Kubernetesin API-palvelin käyttää tilatietojen säilyttämiseen (Mizerany 2014).

API-palvelin on keskeinen osa Kubernetesiä. Se käsittelee kaikki REST-pyynnöt, jotka koskevat jotenkin klusterin tilaa. REST-pyyntöjen lisäksi API-palvelinta voidaan hallita kubectl:llä. Kuviossa 15 on havainnollistettu API-palvelimen kommunikointi eri komponenttien kanssa.

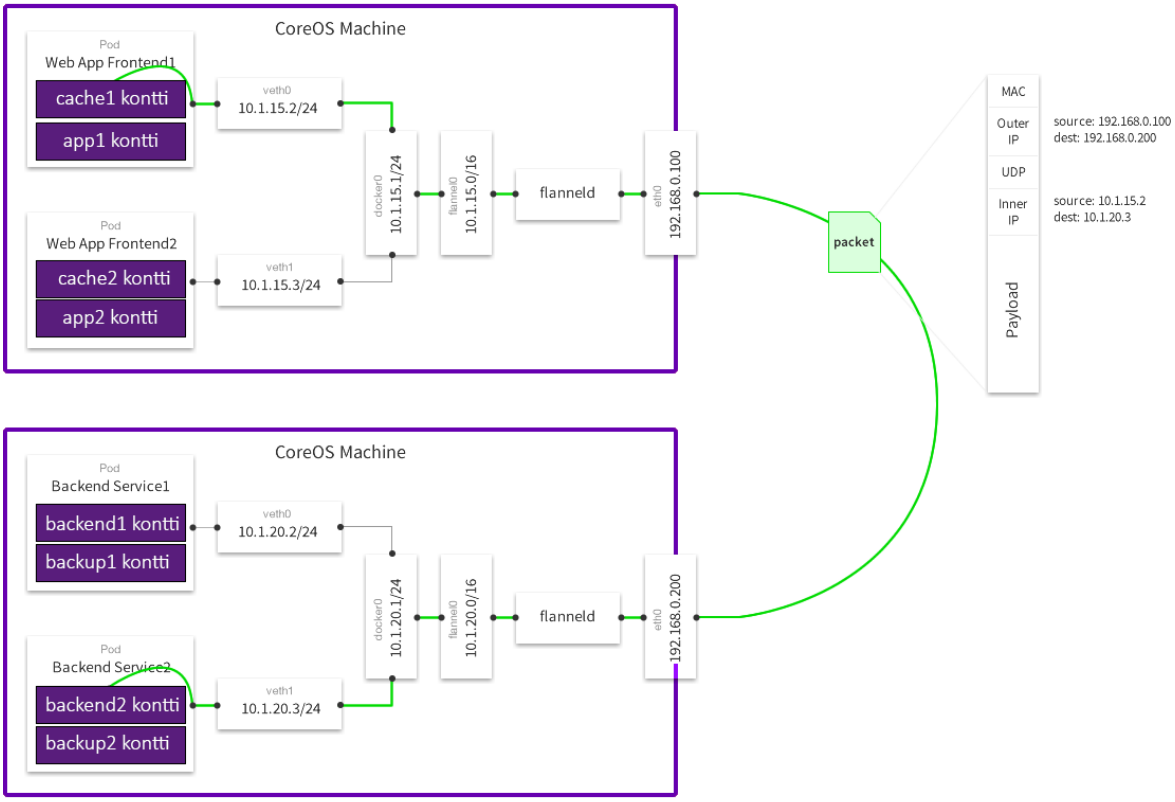


Kuvio 15. Kubernetes API-palvelin (Alkup. kuvio ks. Singh 2015)

Kube-scheduler vuorontaa kontteja nodeille niiden sääntöjen, tavoiteaikataulujen sekä palvelunlaatu- ja sijaintimääritysten mukaan (Braun 2016).

Controller manager on ydintaustaprosessi, joka yrittää jatkuvasti pitää objektit halutussa tilassa. Esim. edellä mainittu ReplicationController käyttää Controller manageria podien määrien säätelemiseen. (Francia 2016a.)

Kuviossa 16 on havainnollistettu nodejen SDN-palvelun **Flannelin** toimintaperiaate. Se on kolmannen osapuolen SDN-palvelu, jolla on mahdollista tehdä eri podien välisiä verkkoja (ks. kuvio 16). Vaihtoehtoisia SDN-palveluita ovat esim. Project Calico tai OpenVSwitch.



Kuvio 16. Flannel verkkotopologia (alkup. kuvio ks. Denham 2017)

Kubernetes ei tarjoa verkkoratkaisua, eli kolmannen osapuolen palveluita on pakko käyttää, jos halutaan vähänkään edistyneempiä podien välisiä verkkoja. Kubernetes tarjoaa kuitenkin oman ratkaisunsa palveluiden saatavuuteen, eli mahdollistaa niille kiinteän IP-osoitteen tai nimen. Myöskin konttien välinen kommunikointi on mahdollista localhost-nimen kautta (käyty läpi aiemmin tässä kappaleessa). (Denham 2017.)

5 Ansible

Ansible on avoimeen lähdekoodiin perustuva työkalu automatisointiin, infrastruktuurin orkestrointiin ja konfiguraation hallintaan. Se käyttää SSH-yhteyttä laitteiden hallintaan, joten vastapäähän ei tarvitse asentaa erillistä agenttia. SSH-yhteyden autentikointiin käytetään oletuksena SSH-avaimia, mutta tuki löytyy myös esim. salasana-autentikoinnille ja Kerberosille. (How Ansible Works n.d.)

Ansible lukee hallittavien laitteiden IP-osoitteet tai nimet polusta `/etc/ansible/hosts`. Siellä ne voidaan luokitella eri ryhmiin, jotta niitä on helpompi automatisoida. Ansible koostuu moduleista, jotka ovat valmiita komentoja ja skriptejä, joihin vain lisätään halutut argumentit. Käyttäjät voivat myös itse kirjoittaa moduleita.

Yksittäisiä ja helppoja tehtäviä varten Ansiblessa on ad-hoc-komento. Esimerkiksi komento `ansible all -m yum -a ”name=apache2 state=installed”` yrittää asentaa kaikille laitteille yum-moduulia käyttämällä apache2-palvelun. Tämä luonnollisesti onnistuu vain laitteilla, joissa on Yum paketinhallintatyökalu.

Ansiblen todellinen etu piilee kuitenkin playbookeissa. Ne ovat YAML-tiedostoja, joihin listataan tehtävät, jotka Ansiblen tulee suorittaa. Alla esimerkki playbookista, joka asentaa viimeisimmän Apachen version ”webservers”-ryhmän laitteille, konfiguroi sen kuuntelemaan porttia 80 ja rajoittamaan sessioiden lukumäärä 200:aan. Se käyttää konfiguraatiopohjana httpd.j2-tiedostoa, johon se korvaa muuttujat ”http_port” ja ”max_clients”.

```
---
- hosts: webservers
  vars:
    http_port: 80
    max_clients: 200
  remote_user: root
  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd state=latest
  - name: write the apache config file
    template: src=/srv/httpd.j2 dest=/etc/httpd.conf
    notify:
    - restart apache
  - name: ensure apache is running (and enable it at boot)
    service: name=httpd state=started enabled=yes
  handlers:
    - name: restart apache
      service: name=httpd state=restarted
```

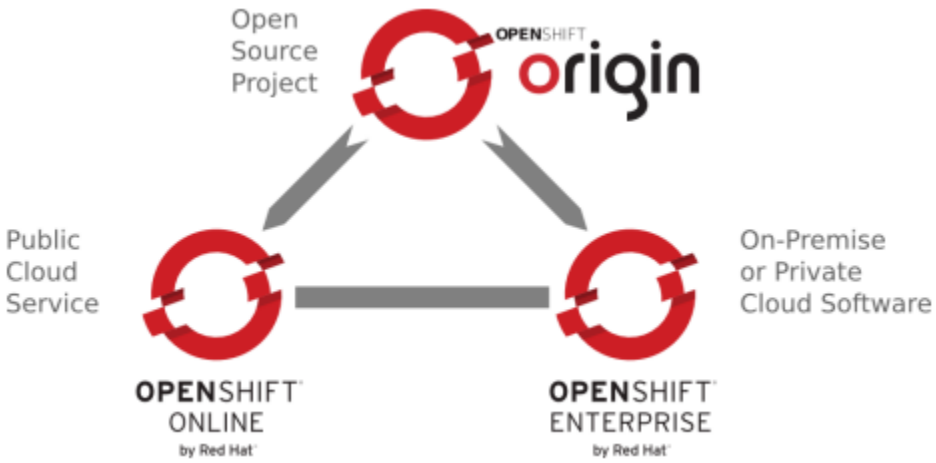
Playbookeilla voidaan keskitetysti hallinnoida konfiguraatioita ja niiden käyttöönottoa. Sen lisäksi niillä on myös mahdollista esim. päivittää palvelin ja samalla poistaa se siksi aikaa valvonnasta ja kuormantasaajalta. Näin ollen kuorma saadaan pois palvelimelta päivityksen ajaksi ja käyttökokemus ei kärsi. (Coca 2017.)

6 PaaS-alustat

6.1 Openshift

6.1.1 Yleistä

OpenShift on RedHatin kehittämä PaaS-alusta. Siitä on saatavilla kolme eri versiota: OpenShift Origin, OpenShift Online ja OpenShift Enterprise. OpenShift Origin on ilmainen, avoimeen lähdekoodiin perustuva ylemmän tason versio, johon Online ja Enterprise pohjautuvat (Ks. kuvio 17). (Pousty & Miller 2014.)



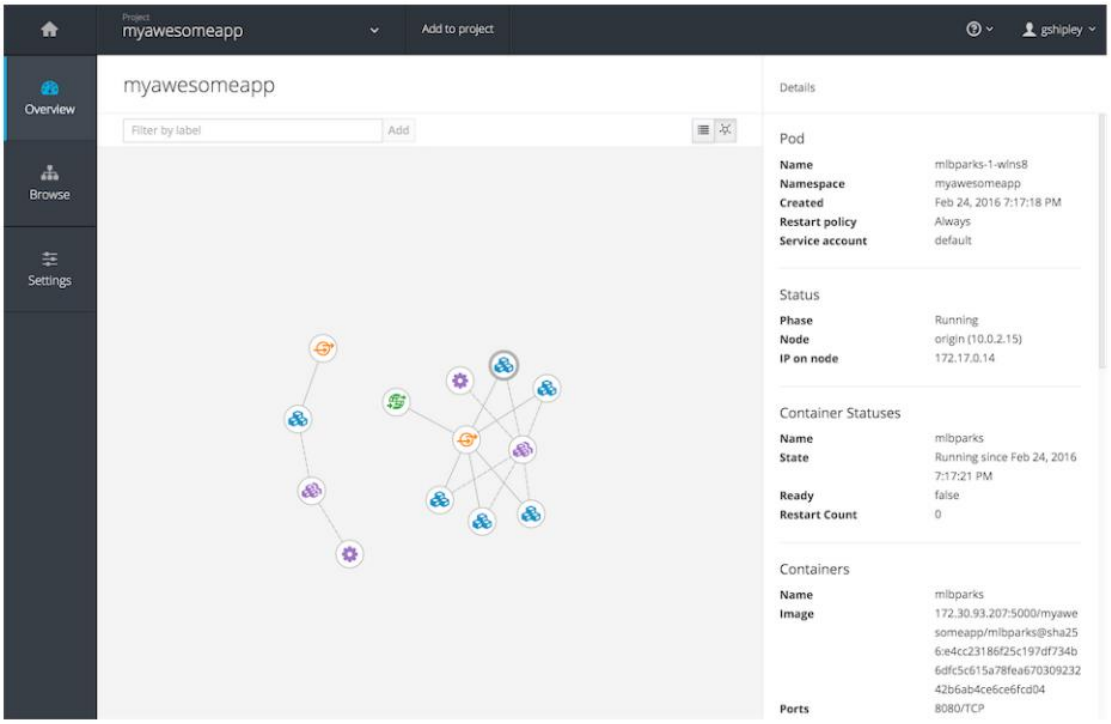
Kuvio 17. OpenShift projektit (Pousty & Miller 2014)

OpenShift Online on muuten sama kuin Origin, mutta se myydään palveluna, jota ajetaan Amazon AWS:ssä ja siihen ajetaan uusi ohjelmisto aina kolmen viikon välein. OpenShift Enterprise on nimensä mukaisesti yrityksille suunnattu versio. Siinä on pidetty tärkeämpänä asiana vakautta kuin uusia ominaisuuksia. Tästä syystä siitä voi puuttua jotain mitä Originissa jo olisikin. Myöskin uusia versioita julkaistaan paljon harvemmin; keskimäärin neljä kertaa vuodessa. OpenShift Enterprise on täysin valmis levykuva, joka on koottu RHEL:n (Red Hat Enterprise Linux) päälle. OpenShift Enterpriseen kuuluu myös Red Hatin täysi tuki. (Pousty & Miller 2014.)

Kaikki kolme versiota perustuvat kuitenkin samaan tekniikkaan: Docker-kontteihin ja niiden orkestrointiin Kubernetesin avulla. OpenShiftin lisäominaisuuksia ovat seuraavat:

- Web-käyttöliittymä
- Komentorivityökalu
- S2I-työkalu (Source-to-Image)
- Jatkuva integraatio ja käyttöönotto
- JBoss EAP -levykuva (Enterprise Application Platform)
- Keskitetty lokien hallinta ja suorituskykymittaus. (Shipley & Dumpleton 2016.)

Web-käyttöliittymän kautta voidaan hallita koko ympäristöä, mm. luoda projekteja, muuttaa skaalausasetuksia ja päivittää sovelluksia. Kuviossa 18 näkyy, miten web-käyttöliittymällä voidaan myös näyttää graafisesti sovelluksen rakenne (Shipley & Dumpleton 2016).



Kuvio 18. OpenShift web-käyttöliittymä (Shipley & Dumpleton 2016)

Komentorivityökalu on yksittäinen suoritettava binääritiedosto, joka tukee kaikkia yleisimpiä käyttöjärjestelmiä (Windows, OSX ja Linux). Sillä voidaan tehdä kaikki samat toimet mitä web-käyttöliittymässäkin. Sen etu on, että yhdestä paikkaa voidaan hallita myös muiden komponenttien natiivityökaluja, kuten Dockerin ja Kubernetesin mukana tulevia. (Shipley & Dumpleton 2016.)

S2I-työkalu helpottaa sovelluksen rakentamista suoraan koodista. Tämän ansiosta OpenShift pystyy rakentaa Docker-kontin pelkän sovelluksen Githubin URL:n perus-

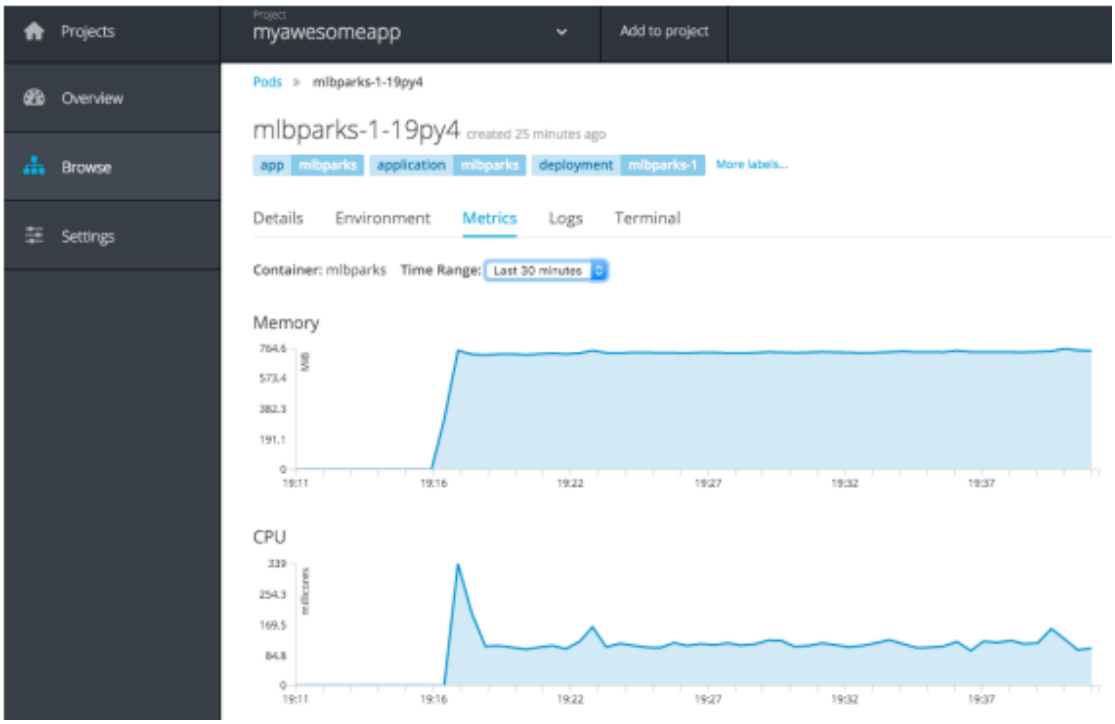
teella. OpenShiftissä on valmiina pohjalevykuvia yleisimmille ohjelmointikielille ja niiden rakennusprosesseille. Se yhdistää sovelluksen lähdekoodin ja valitun levykuvan, ja muodostaa näistä Docker-kontin. (Shipley & Dumpleton 2016.)

OpenShiftiin on integroitu Jenkinsiin pohjautuva CI/CD-palvelu (Continuous Integration & continuous delivery), jolla käyttäjät voivat hallita sovelluksiensa automaattista testaamista ja käyttöönottoa. OpenShiftiin on myös mahdollista liittää käyttäjän oma CI/CD-palvelu jos sellainen löytyy jo entuudestaan. (Continuous Integration with Jenkins n.d.)

RedHat tarjoaa Jboss EAP-alustaa Java-sovelluksien kehittämistä ja tuotantokäyttöä varten. Kohderyhmänä on web- ja mobiilisovellukset. Jboss on sovelluspalvelin, joka helpottaa sovelluksen kommunikointia APIen avulla esim. tietokantaan. Red Hat on tehnyt Jboss EAPista kontti-yhteensopivan levykuvan, jolla sitä voidaan ajaa OpenShiftin päällä. (Red Hat JBoss Enterprise Application Platform for OpenShift n.d.)

OpenShift kerää lokia ajoympäristöstä (runtime), sovelluksien rakentamisista ja käyttöönotoista. Nämä ovat käytettävissä niin web-käyttöliittymällä kuin komentorivityökalullakin. (Shipley & Dumpleton 2016.)

Kuviossa 19 näkyy web-käyttöliittymässä ”myawesomeapp”-sovelluksen muistin ja prosessorin statistiikka.

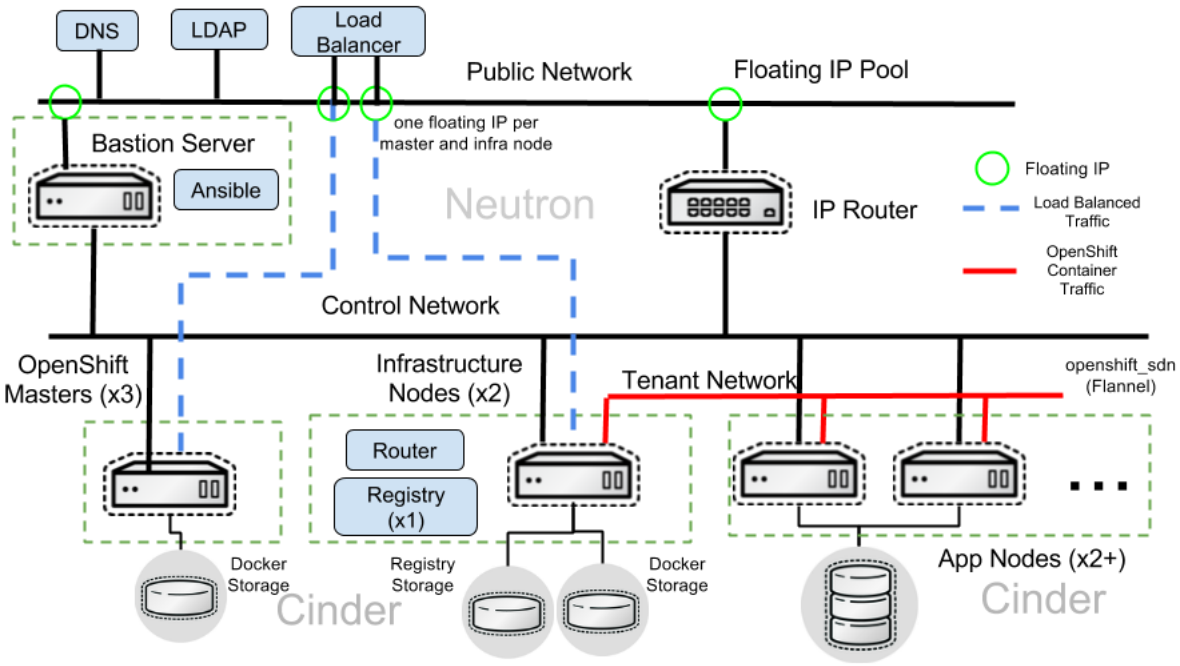


Kuvio 19. OpenShift muisti- ja prosessorigraafit (Shipley & Dumpleton 2016)

6.1.2 Arkkitehtuuri

Tässä kappaleessa käydään läpi OpenShiftin esimerkkiarkkitehtuuria, jossa se on asennettu OpenStackin päälle.

Esimerkkiarkkitehtuurissa (ks. kuvio 20) on isäntäkoneita yhteensä 8: 1 Bastion-palvelin, 3 masteria, 2 infra-nodea (OpenShift Router ja Local Registry) ja 2 app-nodea. Bastion-palvelimella hallitaan isäntäkoneiden ja palveluiden konfiguraatioita Ansible playbookien avulla. Bastion-palvelimella, master- ja infra-nodeilla on oltava julkinen IP-osoite, jotta Bastion-palvelin saa niihin yhteyden SSH:lla. Masterit ja infra-nodet tarvitsevat julkisen IP-osoitteen, koska niihin otetaan yhteyttä ulkoisesta kuormantasaajasta. App-nodet eivät tarvitse julkisia IP-osoitteita, koska ne liikennöivät OpenShift Routerin kautta. Masterit ja nodet ovat Kubernetes-klusterin komponentteja (selitetty kappaleessa 5.2). (Lamourine 2016.)



Kuvio 20. ”OpenShift on OpenStack” arkkitehtuuri (Lamourine 2016)

Kaikilla isäntäkoneilla on kaksi verkkorajapintaa. Toinen on hallintaverkkoon ja toinen sovellusverkkoon. Hallintaverkossa kulkee OpenShiftin eri palvelujen välinen liikenne. Bastion-palvelimen, mastereiden ja infra-nodejen julkiset IP-osoitteet ovat reititetty tähän verkkoon. Sovellusverkossa kulkee podien välinen liikenne ja se kulkee OpenStack Neutronin sisällä. (Lamourine 2017.)

OpenShift tarjoaa sovellusverkkoa varten kahta eri SDN-palvelua, OpenShift SDN, joka pohjautuu OVSään (Open vSwitch) tai Flannel. Molemmat näistä muodostavat verkon vxlan-tunneleiden avulla. OpenShift SDN on kustomoitu OVS ja se eristää podien liikenteen ja verkot toisistaan, eli se on multitenanttinen. Toisaalta sitä käytettäessä, myöskin nodejen sisäinen liikenne reitittyy Neutronin kautta, jolloin se paketoidaan kahteen kertaan ja yhteys hidastuu. Flannelissa puolestaan on ”host-gateway”-tila, jossa suoraan nodeilla voidaan reitittää paketti oikealle podille. Näin ollen liikenteen ei tarvitse mennä Neutronin kautta nodejen sisäisessä kommunikoinnissa. Flannelin huono puoli taas on, että se käyttää samaa aliverkkoa kaikille podeille, eli liikennettä ei eristetä toisiltaan. (Lamourine 2017.)

Mastereilla ja nodeilla on kullakin oma Cinder-blokki Dockerin imageja, kontteja ja local registryä varten. Local registry on Dockerin paikallinen Docker-sovellusten tallen-

nuspaikka, josta käyttäjät voivat nopeasti ladata sovelluksensa tai lähettää sovelluksiaan sinne. Cinder-blokeista voidaan myös tehdä volyymejä, joita on mahdollista liittää käyttäjien podeihin pysyvää tallennusta varten. (Lamourine 2017.)

Tässä esimerkkiarkkitehtuurissa käytetään erillistä LDAP-palvelua käyttäjien tunnistamiseen ja todentamiseen. OpenShift kuitenkin tukee useita eri autentikointitapoja, kuten Github OAuth, Google OAuth, OpenID, HTPasswd ja RequestHeader (Configuring Authentication n.d).

OpenShiftissä on kolme eri näkymää: ylläpito, kehittäjät ja käyttäjät (ks. taulukko 1).

Taulukko 1 OpenShift näkymät (Lamourine 2017)

Rooli	**Tehtävä**	**Nimi (esimerkki)**	**Isäntäkone**
Ylläpito	Infrastruktuurin monitorointi ja hallinta	bastion.openshift.sonera.com	Bastion-palvelin
Kehittäjät	Sovelluksien luominen ja hallinta	devs.openshift.sonera.com	OpenShift Master
Käyttäjät	Sovelluksien käyttäminen	*.apps.openshift.sonera.com	OpenShift Router

Näkymillä tarkoitetaan, mitä kautta kunkin roolin edustaja ottaa yhteyttä OpenShiftiin. Erillinen DNS-palvelu kääntää nämä nimet IP-osoitteiksi, jotka kuuluvat kunkin osa-alueen reitittimelle tai suoraan palvelulle. Esimerkiksi OpenShift Router välittää tietylle sovellukselle kuuluvan liikenteen edelleen oikealle palvelulle, eli se toimii reverse proxynä. Lisäksi palvelukin voi vielä olla ajossa useammalla podilla, eli viimeisen päätöksen, mille podille liikenne ohjataan, tekee kube-proxy. Jos mastereita on useita, korkean saatavuuden takaamiseksi, tarpeeseen tulee myös erillinen kuormantasaaja, jotta kehittäjät voivat saavuttaa kaikki masterit yhden IP:n tai nimen kautta. Jokaisella isäntäkoneella tulee myös olla nimi, ja koska niitä on voitava luoda dynaamisesti, niille on tarjottava IP-osoite DHCP:llä ja nimi dynaamisella DNS:llä, esim. nsupdatella. (Lamourine 2017.)

6.2 Cloud Foundry

6.2.1 Yleistä

Cloud Foundry on PaaS-alusta, mutta sen kehittäjät kutsuvat sitä pilvinatiiviksi alustaksi. Pilvinatiivilla tarkoitetaan, että se abstrahoi taustalla olevan järjestelmän selkeämmin kuin PaaS. Pilvinatiivi alusta pyrkii keskittämään kehittäjien ajan sovelluksen kehittämiseen, eikä esim. konttitekniikoiden pystyttämiseen ja ylläpitämiseen. (Duncan 2016.)

Cloud Foundry on avoimeen lähdekoodiin pohjautuva, Apache2-lisensoitu ja sitä kehittää moniorganisaatioinen yhdistys Cloud Foundry Foundation. Cloud Foundryn kehittäjille tarkoitettu asennuspaketti on ladattavissa ilmaiseksi BOSH-työkalusetin kautta. Se voidaan asentaa mille tahansa alustalle ja se tukee useita eri sovelluskehyksiä (framework) ja ohjelmointikieliä. (Duncan 2016.)

6.2.2 Ominaisuudet

Cloud Foundryn merkittävimpiä ominaisuuksia ovat:

- Elastic Runtime
- Sovelluksen automaattinen pystytys ja skaalaus
- Lokitus ja monitorointi
- Multitenanttisuus

Cloud Foundryssä on uusi Elastic Runtime -konttiympäristö. Se on kirjoitettu uudelleen GO-ohjelmointikielellä ja siihen on kehitetty uusi COE - Diego, joka tukee Dockeria. (Williams 2014.)

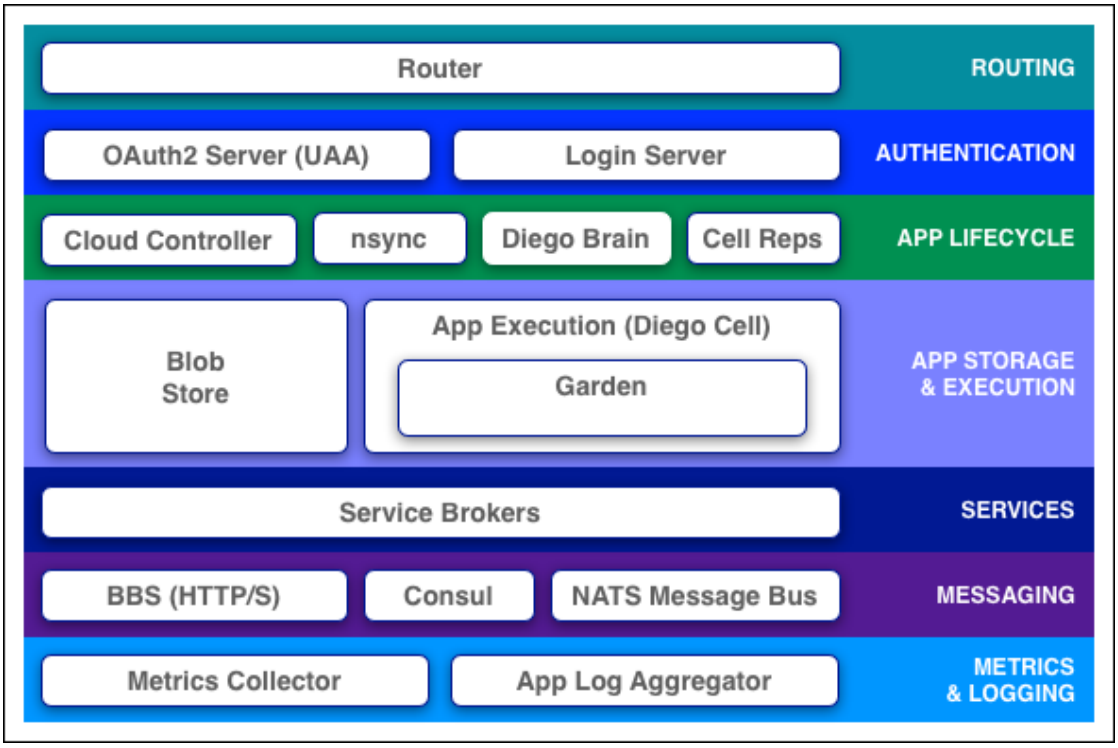
Käyttäjät voivat pystyttää sovelluksiaan antamalla kyseisen sovelluksen lähdekoodin `cf push` -komennolla Cloud Foundrylle ja se hoitaa loput. Cloud Foundry päättelee sovelluksen ohjelmointikielen ja sovelluskehyksen, asentaa ajoympäristön sekä kirjastot, jolloin sovellus on valmis suoritettavaksi. Jos Cloud Foundry ei tunnista ohjelmointikieltä, käyttäjä voi itse määrittää mitä rakennuspakettia (buildpack) kyseiselle sovellukselle pitäisi käyttää. Sovelluksia käynnistetään käyttäjän valitsema määrä ja liikennettä ohjataan tasaisesti näille kaikille. Käyttäjä voi muuttaa skaalausta myös sovelluksen pystytyksen jälkeen. (Tarnoff 2016b.)

Cloud Foundry kerää lokia sovelluksista ja välittää ne käyttäjälle. Käyttäjä taas voi ohjata ne edelleen levylle tallennettavaksi. Cloud Foundry myöskin monitoroi sovelluksien resurssienkäyttöä, jonka avulla käyttäjä voi seurata sovelluksiensa kuormaa. (Tarnoff 2016a.)

Cloud Foundry on multitenanttinen alusta, eli käyttäjien ympäristöt ovat eristetty toisistaan. Tämä on toteutettu RBAC:lla. Organisaatio (org) on vastaava kuin OpenStackissa projekti eli ”tenantti”. Sen alle kuuluu tilat (spaces), joiden avulla sovelluksien ja palveluiden käyttöoikeudet määritellään käyttäjille. Cloud Foundryn ylläpitäjät voivat jäädyttää organisaation, jos sitä käytetään väärin tai esim. laskut ovat maksamatta. (Xu 2017.)

6.2.3 Arkkitehtuuri

Cloud Foundry koostuu kuvion 21 mukaisista komponenteista, jotka käydään läpi kuvion alla.



Kuvio 21. Cloud Foundryn komponentit (Hufnagel 2016a)

Reititin (Router) reitittää liikenteen joko Cloud Controllerille tai Diego Cellissä ajettavalle sovellukselle. Se myös kyselee määräajoin Diego BBS:ltä (Bulletin Board System), missä solussa (Diego cell) tai kontissa mikin sovellus on sillä hetkellä ajossa ja tämän perusteella päivittää reititystaulunsa. (Hufnagel 2016a.)

OAuth2-palvelin ja kirjautumispalvelin vastaavat yhdessä käyttäjien ja sovellusten todentamisesta. OAuth2 jakaa tokeneita sovelluksille. (Hufnagel 2016a.)

Cloud Controller vastaa sovellusten pystyttämisestä. Se välittää pyynnön Diego Brain:lle, joka käynnistää sovelluksen Diego Cell:ssä. Diego Cell on virtuaalikoneissa ajossa oleva palvelu, joka suorittaa käyttäjien sovelluksia konteissa. Nsync, BBS ja Cell Rep ovat sovelluksen skaalaamiseen liittyviä komponentteja. Ne pitävät huolta, että sovelluksia on oikea määrä ajossa, sekä monitoroivat niitä. (Hufnagel 2016a.)

Blobstore on pakettivarasto, johon säilötään suuret binääritiedostot, jotka eivät mahdu Githubiin. Sellaisia ovat esim. sovelluksien koodipaketit, rakentamispaketit ja ”dropletit”, suoritettavat sovellukset. (Hufnagel 2016a.)

Garden on osa Diego Celliä ja sen päällä ajetaan konteissa käyttäjien sovelluksia (Hufnagel 2016).

Sovellukset vaativat usein toimiakseen eri taustapalveluita. Service Broker vastaa näiden taustapalveluiden tarjoamisesta sovelluksille. (Hufnagel 2016a.)

Cloud Foundryn komponentit kommunikoivat keskenään HTTP:llä ja HTTPS:llä. Data tallennetaan kahteen paikkaan: BBSään ja Consuliin. BBS säilöö tiheään päivitettävää ja kertakäyttöistä dataa, kuten cellien ja sovellusten tilatietoja, suorittamattomia tehtäviä ja heartbeat-viestejä. Consul puolestaan säilöö pidempi-ikäistä dataa, kuten komponenttien IP-osoitteita ja ”jaettuja lukkoja”, jotka estävät useamman samanaikaisen kirjoittamisen tietokantaan. (Hufnagel 2016a.)

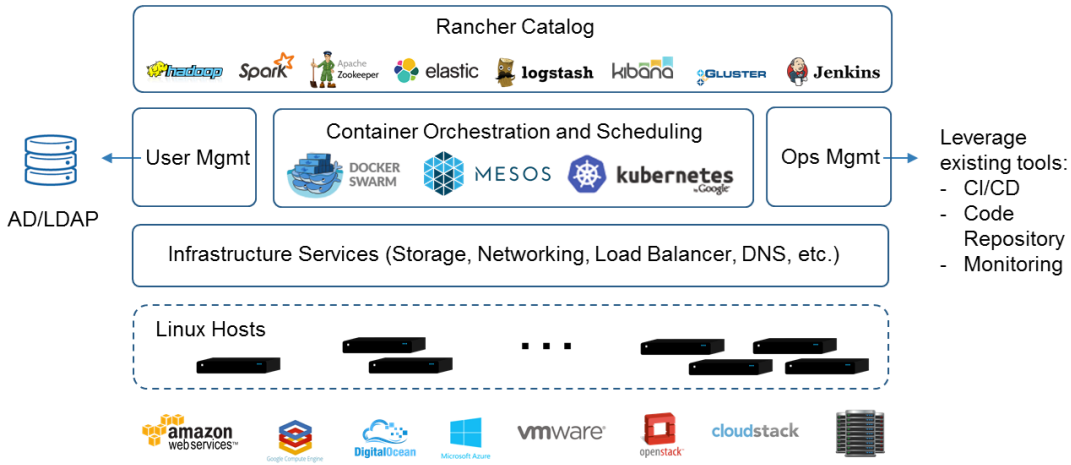
NATS oli Diegon aiemmassa versiossa ensisijainen viestiväylä, mutta nykyään sen sijaan suositaan suoraa kommunikointia HTTP:llä ja HTTPS:llä. NATS on kuitenkin säilytetty joitain kommunikointeja varten. (Hufnagel 2016a.)

Metrics Collector kerää komponenttien statistiikkaa ja sen avulla on mahdollista seurata ympäristön suorituskykyä. App Log Aggregator kerää sovellusten lokia käyttäjiä varten. (Hufnagel 2016a.)

6.3 Rancher

6.3.1 Yleistä

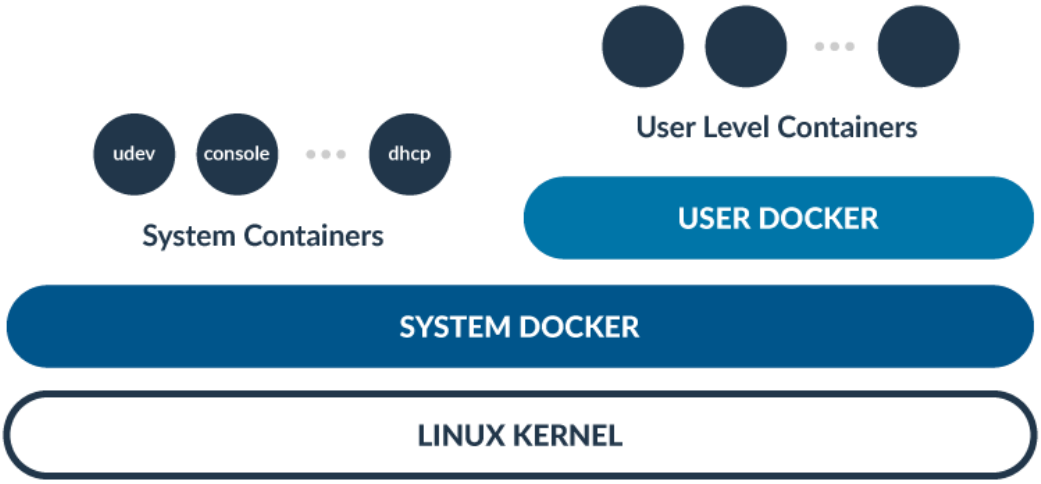
Rancher on avoimen lähdekoodin PaaS-alusta. Se on helppo ja nopea pystyttää. Rancher voidaan asentaa mille tahansa Linux-jakelulle, eikä se välitä onko alla fyysinen palvelin vai virtuaalipalvelin. Rancher koostuu neljästä vaihtoehtoisesta COE:stä, infrapalveluista ja sovelluskatalogista (ks. Kuvio 22). (Denise 2017b.)



Kuvio 22. Rancherin rakenne (Denise 2017b.)

Rancher ajaa palvelimilla Dockeria, jossa pyörii käyttäjien kontit. Lisäksi myös infrapalvelut ovat kontitettu. Näitä ovat seuraavat: verkkopalvelut, tallennustila, kuormantasaaja ja nimipalvelut. Tämä takaa edellä mainitun yhteensopivuuden minkä tahansa Linuxin kanssa. (Denise 2017b.)

Rancher on myös mahdollista asentaa ohjelmapaketin sijasta käyttöjärjestelmänä: RancherOS. Se on minimaalinen käyttöjärjestelmä, jossa kaikki järjestelmäpalvelutkin ajetaan konteissa (ks. kuvio 23). Tällaisia ovat mm. udev, console ja dhcp. RancherOS on kooltaan pieni, koska se ei sisällä mitään ylimääräisiä kirjastoja tai palveluita. Käyttäjät sisällyttävät kaikki heille tarpeelliset kirjastot joka tapauksessa omiin konteihinsa. RancherOS pitää myös huolen, että sen Docker-versio on aina ajan tasalla. Lisäksi se käynnistyy nopeasti (n. 5-10s), joten se suoriutuu hyvin autoskaalauksesta. (Denise 2017b.)



Kuvio 23. RancherOS kontit (Ahmad 2016.)

Rancherissa käyttäjät valjastavat resursseja käyttöönsä heidän pystyttämiin ympäristöihin. Ympäristö koostuu isäntäkoneista ja valituista infrapalveluista, jotka määritetään ympäristötemplaatissa. Käyttäjillä voi olla useita erillisiä ympäristöjä, kuten esim. testaus ja tuotanto. (Denise 2017a.)

6.3.2 Infrastruktuuripalvelut

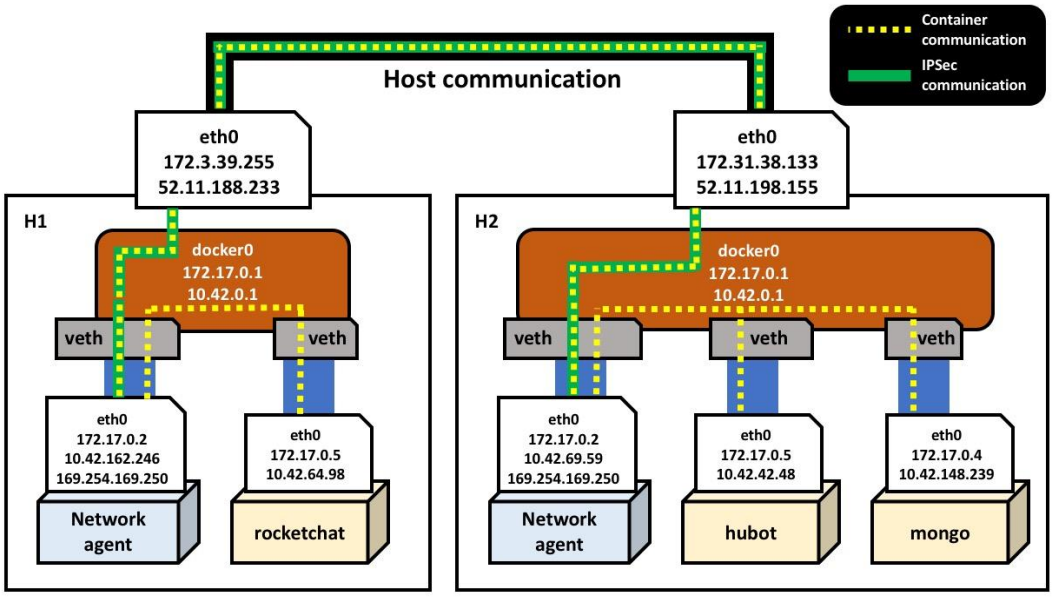
Verkko

Yksi tärkeimmistä infrapalveluista on verkko. Rancher käyttää CNItä (Container Network Interface) verkkojen implementointiin konteille. CNI koostuu määrityksistä ja kirjastoista, joiden avulla voidaan konfiguroida konttien verkkorajapintoja. CNI keskittyy ainoastaan yhdistämään ja poistamaan resursseja konteilta. (Williams 2017.)

CNIn ansiosta Rancherissa voidaan käyttää useita erilaisia verkkoajureita. Oletusympäristötemplaatissa on valittu käyttöön IPSec-verkkoajuri, jolla voidaan luoda yksinkertaisia ja tietoturvallisia verkkoja konteille. Oletusverkko, johon kontti luodaan, on managed-verkko. Siinä kaikki kontit pystyvät oletuksena kommunikoimaan keskenään. Monet Rancherin ominaisuudet, kuten esim. kuormantasaajat ja dns-palvelu vaativat managed-verkon toimiakseen. Muita verkkoja ovat host, bridge, container ja none. Host-verkko on pelkästään isäntäkoneen laajuinen, eli siihen kuuluvat kontit on oltava samalla isäntäkoneella kommunikoidakseen. Bridge-verkko on sama mikä Dockerissa oletuksena, eli se muodostaa 172.17.0.0/16-aliverkon, joka sillataan isäntäkoneen johonkin verkkorajapintaan. Container-verkko yhdistää verkkorajapinnan

toisen kontin verkkorajapintaan, jolloin niillä on sama IP-osoite. None puolestaan tarkoittaa, ettei kontille aseteta verkkorajapintaa ollenkaan, jolloin se on täysin eristetty muista. (Denise 2017h.)

Kuviossa 24 on esimerkki verkkoratkaisusta Rancherissa. Siinä kolme palvelua rocketchat, hubot ja mongo muodostavat yhdessä sovelluksen. Kaikki näistä kuuluvat samaan aliverkkoon 10.42.0.0/16. Hubot ja mongo pystyvät kommunikoimaan suoraan keskenään, koska ne sijaitsevat samalla isäntäkoneella. Rocketchatin kanssa kommunikoidakseen niiden tarvitsee käyttää Network agenttia, joka muodostaa IPSec-tunnelin toiselle isäntäkoneelle.



Kuvio 24. Rancher verkkoratkaisu (A Day in the Life of a Packet Inside Rancher 2016).

Rancherissa verkon sääntöjä voidaan hallita Network Policy Managerilla tai APIen avulla. Network Policy Manager on tällä hetkellä yhteensopiva ainoastaan Cattlen kanssa. Cattle on yksi Rancherin vaihtoehtoisista COEistä. Sääntöjä voidaan tehdä labeleiden ja stackien mukaan. Esim. sallitaan kommunikointi backend- ja frontend-labelit omaavien konttien välillä. Stack puolestaan on Cattlen ominaisuus, jolla saman sovelluksen eri konteista voidaan muodostaa ryhmä. (Denise 2017i.)

Tallennus

Rancherissa tallennustila voidaan liittää muutamalla eri tavalla kontteihin. Stack-kohtainen levyvolyymi tarkoittaa, että se on jaettu samaan stackiin kuuluvien konttien

kanssa. Tämä on oletuskäytäntö. Konttikohtainen levyvolyymi tarkoittaa, että kyseisellä kontilla on oma levyvolyyminsä, jota se ei jaa muiden kanssa. Ympäristökohtainen levyvolyymi on puolestaan jaettu kaikkien kyseiseen ympäristöön kuuluvien konttien kesken. (Denise 2017n.)

Tietoturva

Rancher pitää yllä lokia APIen käytöstä: kuka käytti, mitä kautta ja milloin, sekä klusterin tapahtumista: esim. uuden instanssin luonti konttien skaalauksen vuoksi (Denise 2017a).

Jos palvelun tarvitsee käyttää APIa, niin se tarvitsee todentaa käyttämällä service account API -avaimia. Nämä luodaan konteille ympäristömuuttujina (ks. taulukko 2). (Denise 2017l.)

Taulukko 2. Rancher service account API -avaimet (Denise 2017l.)

Ympäristömuuttuja	**Arvo**
CATTLE_URL	Rancher-palvelimen URL
CATTLE_ACCESS_KEY	Palvelun ympäristön avainparin julkinen osa
CATTLE_SECRET_KEY	Avaimen yksityinen osa

6.3.3 Cattle

Yleistä

Cattle on Rancherin natiivi-COE ja se pohjautuu Docker Swarmiin. Siinä missä Kubernetes ryhmittelee kontit podeihin, Cattle ryhmittelee ne stackeihin. Käyttäjät voivat luoda kontteja käsin, mutta Cattle tarjoaa myös valmiin sovelluskatalogin, josta voidaan asentaa sovelluksia helposti pelkällä napin painalluksella. Käyttäjät voivat tehdä myös omia, yksityisiä palvelukatalogeja. (Lareo 2016.)

Stackit

Samaan stackiin kuuluvat kontit toimivat osana samaa sovellusta. Stackeja voidaan luoda orkestroidusti Rancher Composen avulla. `Rancher-compose.yml` on tiedosto, jossa määritellään, miten kontit luodaan ja liitetään toisiinsa. Konttien määrää voidaan skaalata ja niiden linkkejä kuormantasaajaan tai toisiinsa voidaan muuttaa

myös luomisen jälkeen. Koska kontit ovat lyhytikäisiä, niitä ei ole tarkoituskaan pystyä muokkaamaan määräänsä enempää ajossa, vaan sen sijaan kontti tuhotaan ja tilalle luodaan uusi kontti uudella konfiguraatiolla. Tämä onnistuu Cattlessa helpoiten kloonaamalla vanha kontti ja tekemällä siihen halutut muutokset ennen sen käynnistämistä. (Denise 2017m.)

HA (High Availability)

Palveluiden korkeasta saatavuudesta vastaa HAproxy. Se monitoroi kontteja ja pitää huolta, että niitä on ajossa haluttu määrä. HAproxy voidaan konfiguroida toimimaan vikatilanteissa kolmella eri tavalla:

1. Ei tehdä mitään
2. Tuhotaan vikaantunut kontti ja luodaan uusi tilalle
3. Tuhotaan vikaantuneet kontit ja luodaan tilalle uudet vasta kun ajossa olevia on jäljellä enää X määrä. (Denise 2017e.)

Labelit

Samoin kuin Kubernetes, myös Rancher Cattle käyttää labeleita. Niillä voidaan esim. merkitä palveluita. Labeleilla merkittyjen palveluiden vuorontamiseen voidaan tehdä erilaisia sääntöjä. Esim. 2 samaa labelia omaavaa konttia ei voi olla ajossa samalla isäntäkoneella. (Denise 2017k.)

Myös kuormantasausta voidaan tehdä niiden perusteella (Denise 2017g). Myös isäntäkoneita voidaan merkitä labeleilla, jolloin käyttäjä voi päättää, millä isäntäkoneella kyseistä konttia ajetaan (Denise 2017d). Cattlessa on myös mahdollista linkittää palveluita toisiinsa ja jakaa niiden levyvolyymit tai verkkorajapinta. Tällaista linkkiä kutsutaan sidekickiksi. (Denise 2017k.)

Kuormantasaus

Kuormantasauksesta vastaa myöskin HAproxy. Se käyttää oletuksena round-robin-algoritmiä, mutta siihen voidaan myös konfiguroida oma algoritmi. HAproxy voidaan asettaa tasaamaan kuormaa yksittäisille konteille tai kaikille tietyn labelin omaaville konteille. (Denise 2017g.)

DNS

Cattlessa on sisäinen DNS-palvelu, joka kääntää kaikkien palveluiden nimet kyseisessä ympäristössä. Palvelut pystyvät kommunikoimaan toisillensa nimillä, jos ne ovat samassa stackissa, mutta toiseen stackiin kommunikoitaessa täytyy tarkentaa pelkän palvelun nimen lisäksi myös stackin nimi seuraavalla tavalla: `<palvelun_nimi>.<stackin_nimi>`. (Denise 2017f.)

DNS-nimien julkistaminen onnistuu Cattlella Route53-palvelun avulla. Route53 on AWS:n DNS-palvelu, jolla voidaan luoda yhden verkkotunnuksen alle useita aliverkkotunnuksia. Tällaista verkkotunnusta AWS kutsuu hosted zoneiksi. Cattle päivittää Route53-palveluun julkaistuja nimiä automaattisesti palvelujen metadatatietojen avulla. Metadata Service on Rancherin infrapalvelu, joka mahdollistaa erilaisten tietojen hakemisen kontin sisältä, kuten esim. palvelun nimen, IP-osoitteen ja portin. (Denise 2017c.)

Webhook

Webhookeilla voidaan liipaista toimintoja Rancherissa URL:n kautta. Esim. ulkoinen monitorointijärjestelmä voi webhookin avulla skaalata palveluja niiden kuorman mukaan. (Denise 2017o.)

7 Pilvipalveluntarjonta ja tuotantokäyttö

7.1 Käyttäjäkohderyhmät

Ohjelmistoyrityksillä oli ennen omat palvelimet, joissa ajettiin omia sovelluksia. Toimeksiantajan havaintojen mukaan nykyään yritykset ovat siirtymässä ketterämpään kehitysmalliin, mikropalveluihin ja konttiteknologiaan, jolloin siirrytään omista palvelimista vuokralle julkiseen pilveen.

Yrityksen tulee selvittää, millainen pilvipalvelumalli sovelluksille halutaan. Jos yritys haluaa itse rakentaa sovellusalustan ja ylläpitää sitä, IaaS on paras tähän tarkoitukseen. Suurten ja monimutkaisten ympäristöjen kohdalla sovellusalustan ylläpitäminen on kuitenkin raskasta ja sitoo paljon työntekijöitä. Hyvää tässä on kuitenkin se, ettei käyttäjä ole sidoksissa yhteen palveluntarjoajaan. Käyttäjä voi vuokrata IaaS-resursseja mistä tahansa ja valjastaa ne sovellusalustansa käyttöön. (Childers 2017)

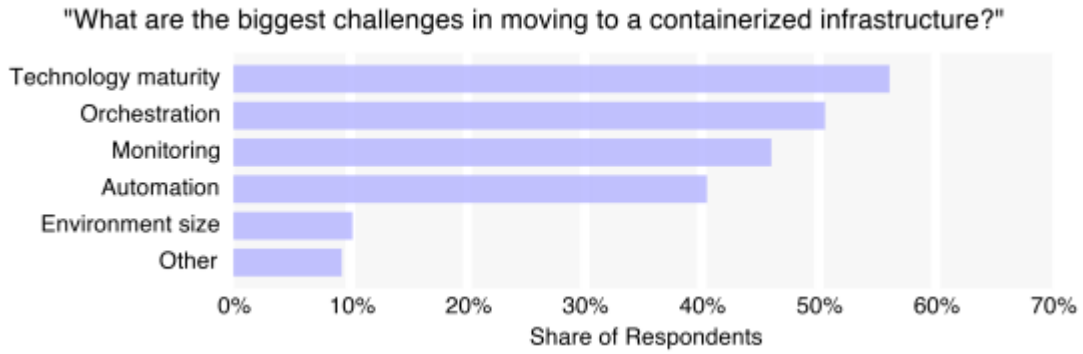
Jos yritys haluaa keskittyä ainoastaan koodin tuottamiseen ja ulkoistaa sen vaatiman sovellusalustan täysin, on paras vaihtoehto PaaS. Siinä palveluntarjoaja vastaa koko sovellusalustasta, joka nykyään tarkoittaa käytännössä Dockeria ja siihen liittyviä palveluita. Huono puoli tässä on, että yritys on sidoksissa yhteen palveluntarjoajaan. (Dougherty 2015.)

Jos yritys haluaa rakentaa itse kontit ja niihin liittyvät palvelut haluamatta kuitenkaan pystyttää tai hallita koko alustaa, on paras vaihtoehto CaaS (Williams 2016). Se on Rightscalen tuoreen tutkimuksen mukaan (Rightscale 2017) suosituin Dockerin käyttömuoto tällä hetkellä. Tässä mallissa palveluntarjoaja ylläpitää alla olevaa konttiklusteria, joka vastaa esimerkiksi suurten ympäristöjen vaatimasta alustan skaalauksesta. Tällainen on esimerkiksi AWS EC2 Container Service (ECS) (Amazon EC2 Container Service n.d). CaaS ei kuitenkaan ole virallinen pilvipalvelumalli sen uutuuden takia. Näin ollen sillä ei myöskään ole yhteistä määritelmää (Williams 2016).

7.2 Tuotantovalmius

Konttiteknologian käyttöönotto yrityksissä lähti räjähdysmäiseen kasvuun Dockerin julkistuksen jälkeen. Syynä oli sen parempi hyötysuhde verrattuna virtuaalikoneisiin ja siirrettävyys. Käytännössä tämä vaikutti yrityksissä virtualisoinnista aiheutuneiden lisenssimaksujen laskuun ja siirrettävyys puolestaan helpotti sovellusten kehittämistä. (Tsidulko 2016.)

O'Reilly ja Ruxit tekivät yhdessä tutkimuksen vuoden 2015 DockerConissa, jossa selvitettiin konttiteknologian käyttökohteita, siihen liittyvien eri työkalujen käyttöönottoja ja konttiteknologian tuomia haasteita. Tutkimuksesta selvisi, että 40% vastanneista käytti Dockeria tuotannossa siinä missä 86% käytti sitä kehittämiseen. Vähäisen tuotantokäytön syyksi ilmeni useita eri haasteita: teknologian kypsyys, orkestrointi, monitorointi, automatisointi, ympäristön suuruus sekä eri osapuolten vakuuttaminen sen hyödyistä (ks. kuvio 25). (Greber 2015.)



Kuvio 25. Konttiteknologian haasteet -kysely (Greber 2015)

Teknologian kypsyydellä viitattiin siihen, että eri työkaluja ja projekteja syntyi jatkuvasti, jolloin niiden kaikkien seuraaminen kävi hankalaksi. Lisäksi monista työkaluista ja projekteista puuttui avainominaisuuksia tai ne olivat vielä betavaiheessa. (Greber 2015.)

Vastanneiden mukaan toiseksi suurin haaste Dockerin tuotantokäytölle oli orkestrointi, jolla tarkoitetaan useiden konttien muodostaman ketjun yhdistämistä ja hallintaa. Orkestrointiin tarkoitettuja työkaluja ovat mm. Kubernetes, Mesos, Docker Swarm ja Docker Compose. Tutkimuksen mukaan näistä suosituin oli Docker Swarm, jota käytti tai aikoi käyttää 44% vastanneista, siinä missä vastaava luku oli Kubernetesillä ja Docker Swarmilla, molemmilla, 22%. Tutkimuksen aikaan Kubernetes ja Docker Swarm olivat vielä betaversiossa. (Greber 2015.)

Monitorointi oli myös yksi konttiteknologian tuotantokäytön haasteista. Perinteiset monitorointijärjestelmät ovat tarkoitettu yksittäisille palvelimille ja ne tarkkailevat lokitiedostoja levyltä. Tällaiset monitorointijärjestelmät eivät skaalaudu hyvin konteihin, joita on usein paljon. Konteilla ei myöskään välttämättä ole pysyvää tallennustilaa, joten levyllä olevat lokit häviävät, jos kontti sammutetaan. Konteille sopii paremmin keskitetty lokijärjestelmä. Tällaisia ovat esim. cAdvisor, Sensu ja Prometheus. Konttien lokit ovat myös mahdollista streamata ulkoiseen palveluun, jossa niitä voidaan keskitetysti monitoroida. Tällaisia palveluita ovat esim. New Relic, Scout ja DataDog. (Greber 2015.)

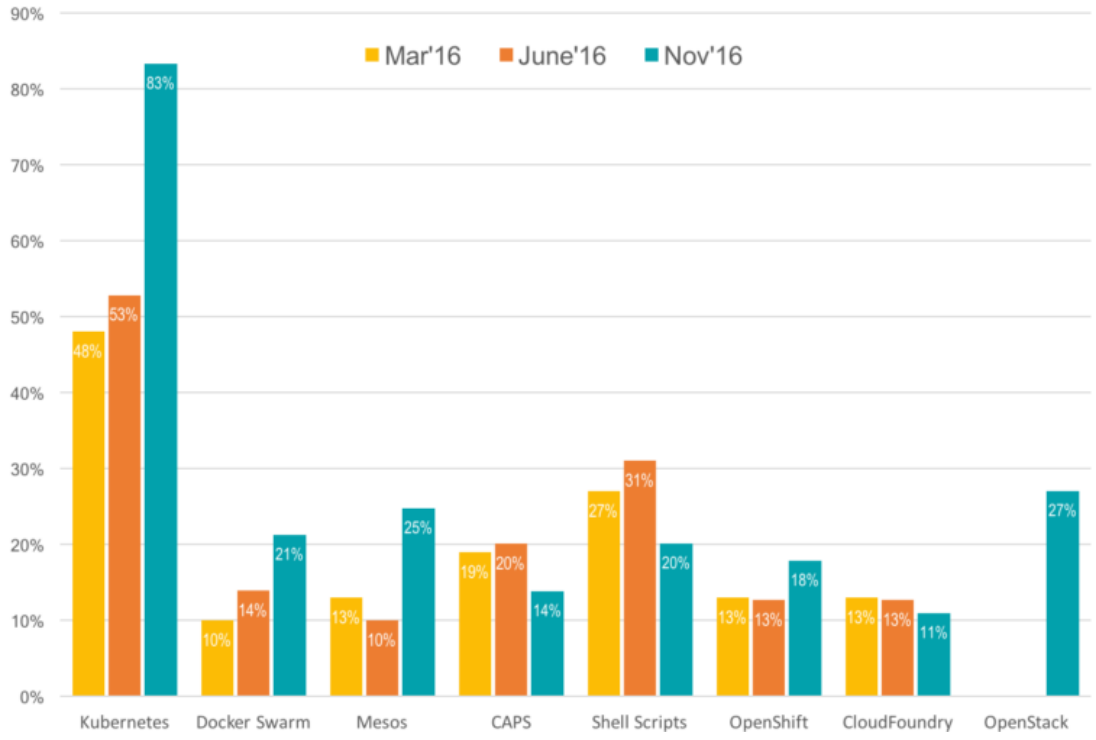
Automatisointi koettiin myös konttiteknologian tuotantokäytön haasteena. Kontteja on käsiteltävä lyhytikäisinä, eli jos kontissa ajettava sovellus pitää päivittää, vanha

kontti poistetaan ja uuden version sisältävä kontti asennetaan tilalle. Tämä prosessi on mahdollista automatisoida CI/CD -työkalulla, Jenkinsillä. (Greber 2015.)

Pitää ottaa kuitenkin huomioon, että tutkimus tehtiin 2015 kesäkuussa, ja sen jälkeen moni asia on muuttunut. Tutkimuksen jälkeen on perustettu yhdistykset Open Container Initiative (OCI) ja Cloud Native Computing Foundation (CNCF), joihin kuuluu monia alan yrityksiä, jotka yhdessä muodostavat standardeja ja määrittelyjä konttiteknologian eri työkaluille. Standardeilla ja määrittelyillä tähdätään eri työkalujen yhtenäistämiseen ja alan hektisyyden vakauttamiseen (Greber 2015). Esimerkiksi OCI teki määrittelyn konttien ajoympäristölle, josta syntyi runC. RunC:tä voidaan käyttää pohjana uusien konttialustojen kehittämiseen ja muokkaamiseen yhteensopivaksi muille käyttöjärjestelmille (Estes 2016).

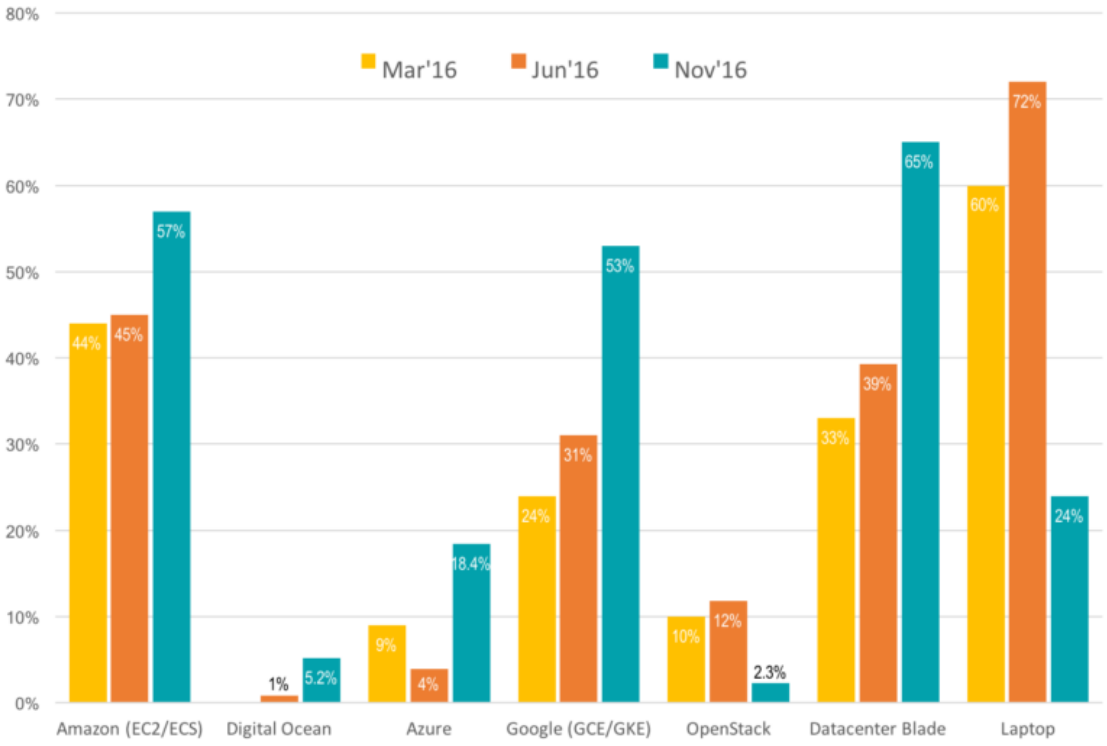
CNCF teki tutkimuksen CloudNativeConin ja KubeConin yhteistapahtumassa marraskuussa 2016. Siinä pyrittiin selvittämään konttiympäristöjen ja konttiteknologian tuotantokäytön kehittymistä.

Vastaajien käyttämistä konttien hallintajärjestelmistä kysyttäessä tuloksista kävi ilmi, että Kubernetesin käyttöaste oli kasvanut 83%:iin, kun vielä kesäkuussa vastaava luku oli 48%. Tuloksista selvisi myös, että suosio käyttäjien omien työkalujen (Shell skriptit, Chef, Ansible, Puppet, Salt) käyttämisestä konttien hallinnassa oli siirtynyt valmiisiin järjestelmiin kuten esim. Kubernetes, Docker Swarm ja Mesos (ks. kuvio 26).



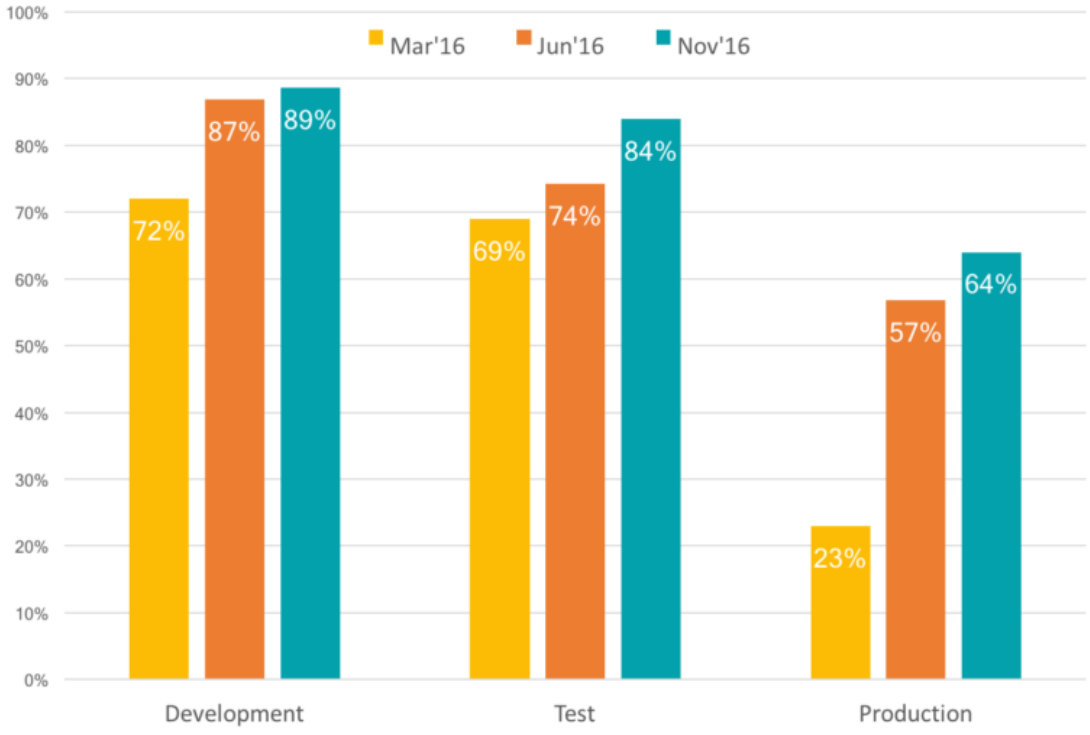
Kuvio 26. Konttien hallintajärjestelmien suosio (Preston 2017.)

Vastaajien konttiympäristöjen sijainnista kysyttäessä tuloksista kävi ilmi, että suosio konttien ajamisesta omilla kannettavilla oli siirtynyt pilvipalveluntarjoajille (Amazon, Digital Ocean, Azure ja Google) ja vastanneiden omiin konesalipalvelimiin (ks. kuvio 27). Tämä tarkoittaa, että kontteja ei enää ajeta pelkästään testimielessä omilla työasemilla vaan ne ovat siirtymässä konesaleihin tuotantokäyttöön.



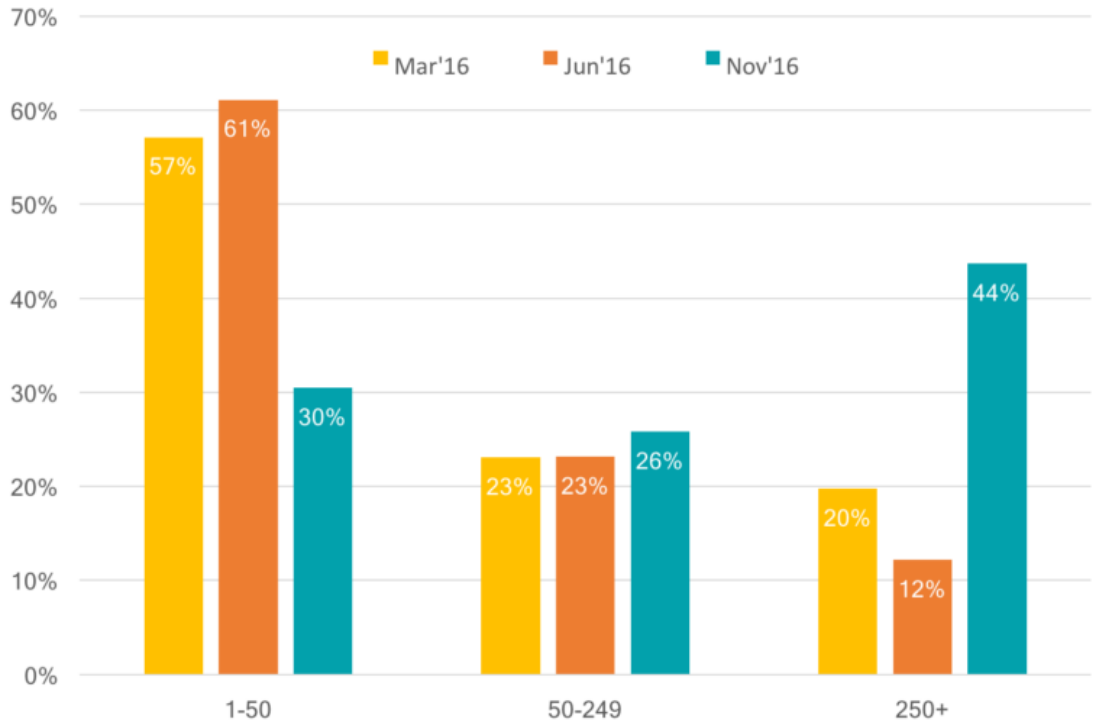
Kuvio 27. Konttiympäristöjen sijainti (Preston 2017.)

Kubernetesin käytöstä kysyttäessä mielenkiintoisin havainto tuloksissa oli, että maaliskuun jälkeen sen tuotantokäyttö on lähes kolminkertaistunut (ks. kuvio 28).



Kuvio 28. Kubernetesin kehitys-, testi- ja tuotantokäyttö (Preston 2017.)

Viimeisenä kysyttiin vastanneiden konttiympäristöjen kokoja. Tutkimuksen tuloksista kävi ilmi, että pienten ympäristöjen (1-50 konttia) määrä on vähentynyt ja suurten (yli 250 konttia) määrä kasvanut (ks. kuvio 29). Tämä kertoo siitä, että tutkimukseen vastanneet ovat uskaltaneet viedä enemmän palveluita konttiympäristöihin ja alkaneet rakentaa suurempia ympäristöjä.



Kuvio 29. Konttiympäristöjen koot konttien lukumäärän mukaan (Preston 2017.)

7.3 Laskutusratkaisut

Yksi toimeksiannon osa-alueista oli laskutusratkaisujen selvittäminen konttiteknologiaa hyödyntävässä PaaS-mallissa. Ongelma on, että PaaS-mallissa palvelimet ovat abstrahoitu pois, joten käyttäjälle allokoidun palvelinresurssin mukaan laskuttaminen ei ole vaihtoehto. Laskutusratkaisuihin otettiin mallia muilta pilvipalveluntarjoajilta.

Heroku laskuttaa konttien suorituskyvyn, lukumäärän ja suoritusajan mukaan sekunnin tarkkuudella. Käyttäjä voi valita kuuden eri suorituskyvyn omaavan kontin väliltä. (Simple, flexiblepricing n.d.)

AWS EC2 on verrattavissa OpenStackiin, eli se on IaaS, mutta Amazon tarjoaa sille lisäpalveluna ECSää, joka valjastaa käyttäjän EC2-resurssit konttiklusteriksi. ECS palvelu on itsessään ilmainen, mutta sen käyttämistä EC2-resursseista laskutetaan. (Amazon EC2 Container Service n.d.)

Jelastic puolestaan laskuttaa sen itse määrittämän yksikön, cloudletin perusteella. Yksi cloudlet vastaa 400Mhz CPU aikaa ja 128Mb muistia. Laskutus tapahtuu cloudletien lukumäärän ja suoritusajan mukaan tunnin tarkkuudella (Jelastic Usage-Based Pricing n.d). Taustalla oleva laskutusjärjestelmä on JBilling (Jelastic Billing Overview n.d).

8 Alustojen vertailu

8.1 OpenShift

OpenShiftin kehittäjä, Red Hat, on tunnettu laadukkaista avoimen lähdekoodin yrityskäyttöön suunnatuista tuotteista. Sen tuotteita käyttää yli 90% Fortune 500 yrityksistä, jotka ovat Yhdysvaltojen 500 suurinta yritystä (Tried. Tested. Trusted. n.d). OpenShift Enterprise versioon kuuluu Red Hatin täysi tuki (Pousty & Miller 2014).

OpenShiftin multitenanttisuus on toteutettu RBAC-menetelmällä, eli kullakin organisaatiolla on oma työtilansa, jonka sisällä voidaan määritellä käyttäjien roolit. Konttien eristyksestä OpenShift ei kerro tarkemmin, mutta niiden verkot se pystyy eristämään OVS:n avulla (Lamourine 2017).

OpenShift on integroitavissa OpenStackiin niin, että se skaalaa automaattisesti alustalle resursseja (nova-instansseja, verkkoja, levyvolyymejä) kuorman mukaan. OpenShift käyttää tähän OpenStack Heatia ja Ceilometeriä (Lamourine 2017). Red Hat tarjoaa kattavan ohjeen OpenShiftin asentamiseen nimenomaan OpenStackin päälle. OpenShift ei kuitenkaan hyödynnä moniakaan OpenStackin komponentteja (esim. Neutron, Glance ja Keystone) parhaalla mahdollisella tavalla, vaan se tuo mukanaan omansa (esim. OpenShift router ja Local Registry), jotka lisäävät kompleksisuutta.

OpenShift on PaaS-alusta, joten se on kohdennettu käyttäjille, jotka haluavat keskittyä ainoastaan koodin tuottamiseen, eikä alustan ylläpitämiseen. Kesäkuussa 2015 OpenShiftin kolmas versio julkaistiin, jossa sen vanha ydin oli vaihtunut Kubernetesiin (Barret 2015). Sen mukana tuli komentorivityökalu, jolla myös käyttäjät pystyivät hallitsemaan Kubernetesiä ja Dockeria, joten tämän puolesta OpenShiftiä voidaan pitää myös jonkin asteisena CaaS-alustana. Referenssiasiakkaista ainoastaan T-Systems tarjosi OpenShiftiä julkisena pilvenä, josta voidaan päätellä, että se on suunniteltu enemmänkin yksityiseksi pilveksi (Our customers n.d). Erityisesti OpenShift sopii Jboss EAPin käyttäjille, koska muissa PaaS-alustoissa ei ole tukea tälle.

8.2 CloudFoundry

Cloud Foundryn takana on yhdistys, Cloud Foundry Foundation. Siihen kuuluu 70 IT-alan suuryritystä, kuten Google, Cisco, HP, IBM ja monia muita (Cloud Foundry Members n.d). Cloud Foundryllä on vertailukohteista vakuuttavin kehitysyhteisö.

Cloud Foundryn multitenanttisuus on toteutettu RBAC-menetelmällä. Se koostuu organisaatioista (orgs), tiloista (spaces) ja käyttäjistä. Organisaatiot ovat ”tenantteja” ja tilat ovat sovelluksien ja palveluiden käyttöoikeuksien määrittämistä varten. Tilat voidaan erotella esim. tuotannoksi ja kehitykseksi. Konttien eristystapa on samanlainen kuin Rancherissa, eli käyttäjän niin halutessa sen kontit voidaan eristää omalle isäntäkoneelle. (Hufnagel 2017.)

Cloud Foundry tarjoaa yksinkertaiset ohjeet sen asentamiseen OpenStackin päälle. Se asennetaan BOSHilla, joka provisioi määritetyt resurssit OpenStackin APIen avulla. Cloud Foundry ei tue autoskaalausta infrastruktuuri- eikä sovellustasolla, mutta sen kaupallisessa versiossa, Pivotalissa, on tuki sovelluksien autoskaalaamiselle (Hufnagel 2016b). Cloud Foundry tuo mukanaan omat komponenttinsa, mutta sen blobstore-komponentti voidaan tuottaa OpenStack Swiftillä (Tarnoff 2016c).

Cloud Foundry on PaaS-alusta, eli se on kohdennettu kehittäjille jotka haluavat keskittyä koodin tuottamiseen. Sitä käytetään sekä julkisena, että yksityisenä pilvenä. Cloud Foundry Foundation järjestää ohjelmaa Certified Platforms, jossa yhdistys auttaa yrityksiä tarjoamaan Cloud Foundryä julkisena pilvenä yhtenäisillä standardeilla.

Tässä ohjelmassa ovat mukana mm. Huawei, IBM, Pivotal, SAP ja Swisscom (Cloud Foundry Memebers n.d.)

8.3 Rancher

Rancher Labs on vuonna 2014 perustettu startup-yritys, jonka takana ovat entiset Cloud.comin (nykyään Citrix) kehittäjät. Vuosina 2015 ja 2016 Rancher Labs on kerännyt sijoituksia yhteensä 30 miljoonaa dollaria. (Rancher Labs n.d.)

Rancher Labsin kehittämä konttialusta Rancher esiteltiin ensimmäistä kertaa vuonna 2014, betaversio julkaistiin 2015 ja virallinen julkaisu oli vuonna 2016. Virallisen julkaisun aikaan betatestaajia oli yli 2500 ja latauskertoja yli miljoona. (Liang 2016.)

Rancherin multitenanttisuus on toteutettu RBAC:lla ympäristötasolla, eli Rancherin ympäristöt ovat organisaatiokohtaisia (ks. kappale 6.3.1). Eri organisaatioiden kontteja ajetaan oletuksena samalla isäntäkoneella, mutta Rancher pystyy myös eristämään ne eri isäntäkoneille, jos näin halutaan. Käytännössä tämä kuitenkin tarkoittaa, että kullekin eristyksen haluavalle organisaatiolle olisi varattava vähintään yksi oma isäntäkone klusterista. (Westoby 2017.)

Rancher ei tarjoa kovin hyviä työkaluja sen integroimiseksi OpenStackiin. Tällä hetkellä se tarjoaa ainoastaan valmista RancherOS-levykuvaa OpenStack Glanceen (Shepherd 2016). Esimerkiksi alustan autoskaalautuvuus pitäisi ohjelmoida itse.

Rancher on CaaS-alusta, joten se on kohdennettu käyttäjille, jotka haluavat itse rakentaa ja hallita kontteja, mutta eivät halua kompleksista ja raskasta alustaa. Rancherin etu on sen helppo ja nopea käyttöönotto ja hallinta. Sen referenssiasiakkaat käyttävät sitä sekä yksityisenä, että julkisena pilvenä. Pilvipalveluntarjoajia referenssiasiakkaista ovat Addteq, EVRY ja EZops (Find a Rancher Partner n.d.)

8.4 Magnum

Magnum on yksi OpenStackin Big Tent -projekteista, eli se on hyväksytty viralliseksi projektiksi OpenStackin teknisen komitean toimesta. (OpenStack Project Teams n.d.)

Magnumin multitenanttisuudesta vastaa OpenStack Keystone (ks. kappale 3.5). Magnum on myöskin täysin eristetty muista käyttäjistä, koska konttiympäristöä ajetaan käyttäjän itse vuokraamilla nova-instansseilla.

OpenStack Magnum tarvitsee toimiakseen OpenStack Nova, Neutron, Glance ja Cinder projektit. Magnum on kevyin vaihtoehto, koska se ei ole kokonainen alusta niin kuin muut vertailukohteista. Sovellustason autoskaalauksesta vastaa käyttäjän valitsema COE, joka voi olla esim. Kubernetes. Magnumissa ei ole valmista ratkaisua resurssien autoskaalaamiselle, mutta tällaista ollaan parhaillaan suunnittelemassa siihen (Exploring Magnum and Senlin Integration for Autoscaling Containers 2015).

OpenStack Magnum on CaaS-palvelu, joten se sopii käyttäjille, jotka haluavat rakentaa ja hallita itse kontteja haluamatta kuitenkaan ylläpitää koko alustaa. Magnum on suunniteltu nimenomaan pilvipalveluntarjoajille, jotka tarjoavat OpenStackia, mutta haluavat lisäpalveluna tarjota myös konttialustaa (Magnum n.d).

9 Johtopäätökset

Toimeksiantona oli PaaS-alustojen vertailu Telialle pilvipalveluntarjontaan, sillä se haluaa vastata konttiteknologiaa käyttävien yritysten tarpeisiin. Tutkimusten edetessä huomattiin, että PaaS ei olekaan niin yksiselitteinen termi, ja että toimeksiannossa sillä tarkoitettiin pikemminkin konttialustojen vertailua. PaaS-alustojen vertailussa oli mukana myös OpenStack Magnum ja Rancher, jotka ovat tarkemmin ottaen CaaS-alustoja.

Kullakin alustalla on omat käyttötarkoituksensa, mutta Telian käyttöön sopivimpia alustoja olivat Cloud Foundry ja OpenStack Magnum. Ne olivat suunniteltu käytettäväksi yksityisen pilven lisäksi myös julkisena pilvenä.

Cloud Foundryssä on kattavammat ominaisuudet kuin Magnumissa, mutta sen pystyttäminen ja ylläpito ovat haastavampaa. Tällä hetkellä suurin osa yrityksistä haluaa CaaS-tyyppisen ratkaisun (Rightscale 2017.), jolloin Cloud Foundryn ominaisuudet voivat olla niille tarpeettomia. Toisaalta Cloud Foundryllä on vertailukohteiden vakuuttavin kehitysyhteisö, joka koostuu lukuisista alan suuryrityksistä. Tämä takaa varman ja jatkuvan kehityksen.

Tämän tutkimuksen tietojen perusteella paras konttialusta Telian käyttöön olisi OpenStack Magnum. Se vastaa tällä hetkellä parhaiten sekä käyttäjien että Telian omia tarpeita. Myöskin Magnumin investointikustannukset ovat pienet, koska se liitettäisiin osaksi jo olemassa olevaa OpenStack-alustaa.

Toimeksiannon yhtenä osa-alueena oli laskutusratkaisut, joita tutkittiin vertailemalla muiden pilvipalveluntarjoajien ratkaisuja. Malleja oli kolme:

- Konttien suoritusajan mukaan laskuttaminen (tämä vaatisi palvelun, jolla seurataan konttien suoritusaikaa)
- Valmis laskutusjärjestelmä JBilling
- Alla olevan infrastruktuurin mukaan laskuttaminen, esim. Magnumissa käyttäjän nova-instanssien mukaan.

Telian käyttöön sopivin laskutusratkaisu olisi nova-instanssien mukaan laskuttaminen, eli sama malli kuin tälläkin hetkellä Telia Cloudissa. OpenStack Magnum voitaisiin tarjota ilmaisena lisäpalveluna tai kiinteällä kuukausimaksulla.

Myös konttiteknologian tuotantovalmiuden tutkiminen oli yksi toimeksiannon osa-alue. Kappaleessa 7.2 käsitellyiden tutkimusten mukaan moniin konttiteknologian tuotantokäytön haasteisiin (teknologian kypsyys, orkestrointi, monitorointi ja automatisointi) on kehitetty ratkaisuja ja yritykset hyödyntävät niitä kasvavassa määrin.

10 Pohdinta

Opinnäytetyön tavoitteena oli selvittää paras PaaS-alusta Telian käyttöön ja tälle sopiva laskutusratkaisu. Lisäksi tavoitteena oli saada käsitys konttiteknologian tuotantovalmiudesta. Tutkimuksen edetessä huomattiin, että aihealueesta ei oltu tehty riittävän laajaa taustatutkimusta, jolloin suuri osa ajasta kului uuden tiedon omaksumiseen tutkimuksen ohessa. Esimerkiksi ennen vertailun toteuttamista, täytyi ymmärtää nykypäivän pilvipalvelumallien erot ja niiden kohderyhmät.

PaaS-alustojen vertailu toteutettiin teoriatiedon ja kohderyhmiä koskeneiden tutkimusten perusteella. Teoriatieto perustui pitkälti yritysten omiin dokumentaatioihin ja tuote-esittelyihin.

Tulokset osoittivat, että PaaS-alustat ovat murrosvaiheessa uuden CaaS-palvelumallin takia. Lisäksi huomattiin, että PaaS-alustat ovat komplekseja pystyttää

ja ylläpitää, ja näin ollen sitovat paljon työntekijöitä. Työssä oli alun perin tarkoitus tutkia alustoja myös käytännössä, mutta tutkimuksen edetessä huomattiin aihealueen laajenevan liikaa, joten tämä osuus jätettiin pois. Aiheeseen sopiva jatkotutkimus voisi olla OpenStack Magnumin tai CloudFoundryn käyttöönotto ja alustan autoskaalauksen toteuttaminen.

Telia voi hyödyntää opinnäytetyön tutkimuksia valitessaan PaaS-alustaa ja siihen sopivaa laskutusratkaisua.

Lähteet

A Day in the Life of a Packet Inside Rancher. 2016. Rancher blogikirjoitus. Viitattu 28.2.2017. http://rancher.com/day-life-packet-inside-rancher/

A Technical Overview of Kubernetes (CoreOS Fest 2015). 2015. Video CoreOS:n Youtube-kanavalla. Kertojana Brendan Burns. Viitattu 10.2.2017. https://www.youtube.com/watch?v=WwBdNXt6wO4

Ahmad, E. 2016. Rancher Github. Viitattu 22.2.2017. https://github.com/rancher/os/blob/master/docs/rancheros.png

Amazon EC2 Container Service. N.d. AWS:n tuotesivut. Viitattu 13.4.2017. https://aws.amazon.com/ecs/

Architecture. 2017. OpenStack dokumentaatio. Viitattu 15.2.2017. https://docs.OpenStack.org/ops-guide/architecture.html

Barret, M. 2015. OpenShiftin artikkeli. Viitattu 17.4.2017. https://blog.openshift.com/openshift-enterprise-3-evolving-paas-future/

Berendt, C. 2015. Simple auto scaling environment with Heat. Superuserin artikkeli. Viitattu 17.3.2017. http://superuser.OpenStack.org/articles/simple-auto-scaling-environment-with-heat/

Braun, S. 2016. Introduction to Kubernetes. Mirantiksen blogikirjoitus. Viitattu 9.12.2016. https://www.mirantis.com/blog/introduction-to-kubernetes/

Childers, C. 2017. Of Pies and Platforms: Platform-as-a-Service vs. Containers-as-a-Service. Viitattu 12.4.2017. https://thenewstack.io/pies-platforms-paas-vs-caas/

Cloud Foundry Certified Platforms. N.d. Cloud Foundryn infosivut. Viitattu 11.4.2017. https://www.cloudfoundry.org/certified-platforms/

Cloud Foundry Members. N.d. Cloud Foundryn infosivut. Viitattu 11.4.2017. https://www.cloudfoundry.org/members/

Configuring Authentication. N.d. OpenShift dokumentaatio. Viitattu 20.2.2017. https://docs.openshift.com/enterprise/3.0/admin_guide/configuring_authentication.html

Continuous Integration with Jenkins. N.d. OpneShift dokumentaatio. Viitattu 23.3.2017. https://developers.openshift.com/managing-your-applications/continuous-integration.html

Coca, B. 2017. Intro to Playbooks. Ansiblen dokumentaatio. Viitattu 15.2.2017. http://docs.ansible.com/ansible/playbooks_intro.html

Dake, S. 2015. DEVNET 1157 - Meet Magnum, OpenStack[1]s New Containers-as-a-Service Project. Video Cisco DevNetin Youtube-kanavalla. Viitattu 21.3.2017. https://www.youtube.com/watch?v=BM6nFH7G8Vc

Denham, T. 2017. Flannel. CoreOS Github. Viitattu 14.2.2017. https://github.com/coreos/flannel

Denise. 2016. Overview of RancherOS. Rancher dokumentaatio. Viitattu 22.2.2017. https://docs.rancher.com/os/

Denise. 2017a. Audit Logging. Rancher dokumentaatio. Viitattu 9.3.2017. https://docs.rancher.com/rancher/v1.5/en/rancher-services/audit-log/

Denise. 2017b. Environments. Rancher dokumentaatio. Viitattu 9.3.2017. https://docs.rancher.com/rancher/v1.5/en/environments/

Denise. 2017c. External DNS Service. Rancher dokumentaatio. Viitattu 9.3.2017. http://docs.rancher.com/rancher/v1.5/en/cattle/external-dns-service/

Denise. 2017d. Getting Started with Hosts. Rancher dokumentaatio. Viitattu 9.3.2017. http://docs.rancher.com/rancher/v1.5/en/hosts/

Denise. 2017e. Health Checks. Rancher dokumentaatio. Viitattu 9.3.2017. http://docs.rancher.com/rancher/v1.5/en/cattle/health-checks/

Denise. 2017f. Internal DNS Service in Cattle Environments. Rancher dokumentaatio. Viitattu 9.3.2017. http://docs.rancher.com/rancher/v1.5/en/cattle/internal-dns-service/

Denise. 2017g. Load Balancers. Rancher dokumentaatio. Viitattu 9.3.2017. http://docs.rancher.com/rancher/v1.5/en/cattle/adding-load-balancers/

Denise. 2017h. Overview of Rancher. Rancher dokumentaatio. Viitattu 22.2.2017. http://docs.rancher.com/rancher/v1.4/en/

Denise. 2017i. Networking. Rancher dokumentaatio. Viitattu 28.2.2017. http://docs.rancher.com/rancher/v1.4/en/rancher-services/networking/

Denise. 2017j. Network Policy. Rancher dokumentaatio. Viitattu 28.2.2017. http://docs.rancher.com/rancher/v1.4/en/rancher-services/network-policy/

Denise. 2017k. Services. Rancher dokumentaatio. Viitattu 9.3.2017. http://docs.rancher.com/rancher/v1.5/en/cattle/adding-services/

Denise. 2017l. Service Accounts. Rancher dokumentaatio. Viitattu 9.3.2017. https://docs.rancher.com/rancher/v1.5/en/rancher-services/service-accounts/

Denise. 2017m. Stacks. Rancher dokumentaatio. Viitattu 9.3.2017. http://docs.rancher.com/rancher/v1.5/en/cattle/stacks/

Denise. 2017n. Storage Service. Rancher dokumentaatio. Viitattu 2.3.2017. http://docs.rancher.com/rancher/v1.3/en/rancher-services/storage-service/

Denise. 2017o. Webhooks. Rancher dokumentaatio. Viitattu 9.3.2017. http://docs.rancher.com/rancher/v1.5/en/cattle/webhook-service/

Docker for the Virtualization Admin. 2016. Docker E-kirja. Viitattu 7.12.2016. https://goto.docker.com/rs/929-FJL-178/images/Docker-for-Virtualization-Admin-eBook.pdf?mkt_tok=eyJpIjoiTlRaa09ERXhNelE1TVRabCIsInQiOiJNdHVcL1N5a3hqdlp1ekUzR3hhQ0xtUW5UUEIzYVd5aXR4XC9BSEZpZVVJT05KZndGWWY4ZURpTzJ0bmdsK2U2VkFBaUxrNTBFMmRVN1g1WmNkS0w3RmhvMHAwa3RhcTNWY3UxWXNxRXJLQTF3PSJ9

Dougherty, P. 2015. IaaS vs. PaaS vs. CaaS — Which Cloud Architecture is Right For You? Part 2. ContainerShipin artikkeli. Viitattu 13.4.2017. https://blog.containership.io/iaas-vs-paas-vs-caas-which-cloud-architecture-is-right-for-you-part-2-a72623d7d001

Duncan, Winn. 2016. O'Reillyn E-Kirja. Viitattu 13.12.2016. https://www.safaribooksonline.com/library/view/cloud-foundry-the/9781491932421/

Estes, P. 2016. runC: The little container engine that could. Opensourcen artikkeli. Viitattu 11.4.2017. https://opensource.com/life/16/8/runc-little-container-engine-could

Exploring Magnum and Senlin Integration for Autoscaling Containers. 2015. OpenStack Foundationin Youtube-kanava. Viitattu 22.4.2017. https://www.youtube.com/watch?v=54Ysv4d8qsw

Find a Rancher Partner. N.d. Rancherin infosivut. Viitattu 11.4.2017. http://rancher.com/partners-index

Francia, S. 2016a. Kube-controller-manager. Kubernetes dokumentaatio. Viitattu 10.2.2017. https://kubernetes.io/docs/admin/kube-controller-manager/

Francia, S. 2016b. Kubelet. Kubernetes dokumentaatio. Viitattu 10.2.2017. https://kubernetes.io/docs/admin/kubelet/

Greber, A. 2015. State of Containers and the Docker Ecosystem 2015. O'Reillyn E-kirja. Viitattu 11.4.2017. http://www.oreilly.com/webops-perf/free/files/containers-and-docker-ecosystem-2015.pdf

Greiner, R. 2014. Windows Azure IaaS vs. PaaS vs. SaaS. Greinerin blogikirjoitus. Viitattu 28.11.2016. http://robertgreiner.com/2014/03/windows-azure-iaas-paas-saas-overview/

Heat Orchestration Template (HOT) specification. N.d. OpenStack dokumentaatio. Viitattu 17.3.2017. https://docs.OpenStack.org/developer/heat/template_guide/hot_spec.html

How Ansible Works. N.d. Ansiblen kotisivut. Viitattu 15.2.2017. https://www.ansible.com/how-ansible-works

Hufnagel, M. 2016a. Cloud Foundry Components. Cloud Foundryn dokumentaatio. Viitattu 13.12.2016. https://docs.cloudfoundry.org/concepts/architecture/

Hufnagel, M. 2016b. Managing Scheduled Scaling in the App Autoscaling Service. Pivotalin dokumentaatio. Viitattu 17.4.2017. https://docs.pivotal.io/pivotalcf/1-8/appsman-services/autoscaler/scheduled-scaling.html

Hufnagel, M. 2017. Understanding Cloud Foundry Security. Cloud Foundryn dokumenaatio. Viitattu 17.4.2017. https://docs.cloudfoundry.org/concepts/security.html

Jelastic Billing Overview. N.d. Jelastic tuotedokumentaatio. Viitattu 21.3.2017. http://ops-docs.jelastic.com/billing-system-overview

Jelastic Usage-Based Pricing. N.d. Jelasticin tuotesivut. Viitattu 16.4.2017. https://docs.jelastic.com/pricing-model

Kannan, V & Marmol, V. 2015. Resource Usage Monitoring in Kubernetes. Kubernetesin blogi. Viitattu 10.2.2017. http://blog.kubernetes.io/2015/05/resource-usage-monitoring-kubernetes.html

Kekäläinen J. 2015. Tervetuloa taloon kesätyöntekijät 2015. TeliaSoneran intranet. Vain sisäiseen käyttöön. Viitattu 28.11.2016.

Komissio hyväksyy Telian ja Soneran fuusion tietyin ehdoin. 2002. Euroopan unionin lehdistötiedote. IP/02/1032. Viitattu 28.11.2016. http://europa.eu/rapid/press-release_IP-02-1032_fi.pdf

Konesalien kuningas nousee Pitäjänmäkeen. N.d. Telian tuotedokumentaatio. Viitattu 8.4.2017. https://www.telia.fi/yrityksille/tuotteet/tietoliikenne/telia-data-center/datakeskus-pitajanmakeen

Kubectl Overview. N.d. Kubernetes dokumentaatio. Viitattu 10.2.2017. https://kubernetes.io/docs/user-guide/kubectl-overview/

Liang, S. 2016. Announcing Rancher 1.0 GA. Rancherin artikkeli. Viitattu 17.4.2017. http://rancher.com/announcing-rancher-1-0-ga/

Linux* Containers Streamline Virtualization and Complement Hypervisor-Based Virtual Machines. N.d. White paper. Viitattu 5.12.2016. http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/linux-containers-hypervisor-based-vms-paper.pdf

Lamourine, M. 2016. OpenShift on OpenStack. OpenShift Github. Viitattu 20.2.2017. https://github.com/redhat-OpenStack/openshift-on-OpenStack/blob/master/README.adoc

Lareo David A. 2016. Rancher blogikirjoitus. Viitattu 10.3.2017. http://rancher.com/cattle-swarm-kubernetes-side-side/

Lamourine, M., Cook, R. & Collier, S. 2017. Deploying Red Hat OpenShift Container Platform 3 on Red Hat OpenStack. Red Hatin opas. Viitattu 17.2.2017. https://access.redhat.com/sites/default/files/attachments/ocp-on-osp-1.4.pdf

Magnum. N.d. OpenStack dokumentaatio. Viitattu 7.4.2017. https://wiki.openstack.org/wiki/Magnum

Mell, P. & Grance, T. 2011. The NIST Definition of Cloud Computing. NIST:n suositus. Viitattu 28.11.2016. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf

Merket, K. 2015. Containers & Kubernetes. Googlen SlideShare-esitys. Viitattu 10.2.2017. http://www.slideshare.net/kitmerker/devnexus-2015-kubernetes-container-engine

Mizerany, B. 2014. About Etcd, the Distributed Key-Value Store Used For Kubernetes, Google's Cluster Container Manager. Thenewstackin artikkeli. Viitattu 9.12.2016. http://thenewstack.io/about-etcd-the-distributed-key-value-store-used-for-kubernetes-googles-cluster-container-manager/

Nadeeshani, P. 2016. What is Kubernetes. Nadeeshanin Blogi. Viitattu 6.2.2017. http://nshani.blogspot.fi/2016/02/getting-started-with-kubernetes.html

Networking architecture. 2017. OpenStackin wiki-sivusto. Viitattu 11.1.2017. http://docs.OpenStack.org/security-guide/networking/architecture.html

Neutron. 2014. OpenStackin wiki-sivusto. Viitattu 9.1.2017. https://wiki.OpenStack.org/wiki/Neutron

OpenStack Project Teams. N.d. OpenStackin dokumentaatio. Viitattu 22.4.2017. https://governance.openstack.org/tc/reference/projects/

Orenstein, D. 2000. Application Programming Interface. Computerworldin artikkeli. Viitattu 14.4.2017.

Our Customers. N.d. OpenShift infosivut. Viitattu 11.4.2017. https://www.openshift.com/container-platform/customers.html

Pepple, K. 2011. Deploying OpenStack. Safari verkkokirjasto. O'Reilly E-kirja. Viitattu 11.1.2017. https://www.safaribooksonline.com/library/view/deploying-OpenStack/9781449311223/ch03.html

Petazzoni, J. 2015. Anatomy of a Container: Namespaces, cgroups & Some Filesystem Magic – LinuxCon. Slideshare-esitys. Viitattu 17.4.2017. https://www.slideshare.net/jpetazzo/anatomy-of-a-container-namespaces-cgroups-some-filesystem-magic-linuxcon

Plans & Pricing. N.d. OpenShift Onlinen tuotesivut. Viitattu 11.4.2017. https://www.openshift.com/pricing/index.html

Pousty, S. & Miller, K. 2014. Getting Started with OpenShift. OpenShift E-kirja. Viitattu 11.12.2016. http://bit.ly/openshift-ebook

Preston, B. 2017. Container Management Trends: Kubernetes moves out of testing and into production. CNCF:n kysely. Viitattu 15.4.2017. https://www.cncf.io/blog/2017/01/17/container-management-trends-kubernetes-moves-testing-production/

Pricing. N.d. AWS-sivusto. Viitattu 17.3.2017. https://aws.amazon.com/lambda/pricing/

Professional Support. N.d. Rancherin tukisivut. Viitattu 11.4.2017. http://rancher.com/support

Rancher Labs. N.d. Crunchbasen artikkeli. Viitattu 21.4.2017. https://www.crunchbase.com/organization/rancher-labs

Red Hat JBoss Enterprise Application Platform for OpenShift. N.d. Red Hat tuotedokumentaatio. Viitattu 18.3.2017. https://access.redhat.com/documentation/en-us/red_hat_jboss_middleware_for_openshift/3/html/red_hat_jboss_enterprise_application_platform_for_openshift/

Replication Controller. N.d. Kubernetes dokumentaatio. Viitattu 10.2.2017. https://kubernetes.io/docs/user-guide/replication-controller/

Rightscale. 2017. STATE OF THE CLOUD REPORT. Rightscalen tutkimus. Viitattu 13.4.2017. http://links.rightscale.com/Y07oZ0GtB001T0038GN0vZ6

Riondato, M. N.d. Chapter 14. Jails. FreeBSD:n dokumentaatio. Viitattu 17.4.2017. https://www.freebsd.org/doc/handbook/jails.html

Rountree, D & Castrillo, I. 2014. The Basics of Cloud Computing: Understanding the Fundamentals of Cloud Computing in Theory and Practice. E-kirja. Syngress Publishing. Luku 1 Introduction to the Cloud. Viitattu 30.11.2016. http://library.books24x7.com.ezproxy.jamk.fi:2048/assetviewer.aspx?bookid=56576&chunkid=999775036&noteMenuToggle=0&hitSectionMenuToggle=0&leftMenuState=1

Services. N.d. Kubernetes dokumentaatio. Viitattu 10.2.2017. https://kubernetes.io/docs/user-guide/services/

Shepherd, D. 2016. Rancherin Github. Viitattu 17.4.2017. https://docs.rancher.com/os/running-rancheros/cloud/openstack/

Simple, flexible pricing. N.d. Herokun tuotesivut. Viitattu 16.4.2017. https://www.heroku.com/pricing

Singh, S. 2015. Cluster management with Kubernetes. Googlen Powerpoint-esitys. Viitattu 13.2.2017. http://www.slideshare.net/SatnamSingh67/2015-0605-cluster-management-with-kubernetes

Smith, S. OpenStack Essentials. Linux Academyn verkkokurssi. Viitattu 23.12.2016. https://linuxacademy.com/cp/courses/lesson/course/128/lesson/1/module/18

Sonera ja Tele Finland ovat nyt Telia – Mitä se tarkoittaa?. N.d. Telian tuotedokumentaatio. Viitattu 8.4.2017. https://www.telia.fi/asiakastuki/tietoa-muutoksesta/

Tarnoff, B. 2016a. Cloud Foundry Logging. Cloud Foundryn dokumentaatio. Viitattu 15.3.2017. https://docs.cloudfoundry.org/running/managing-cf/logging.html

Tarnoff, B. 2016b. How Applications Are Staged. Cloud Foundryn dokumentaatio. Viitattu 15.3.2017. https://docs.cloudfoundry.org/concepts/how-applications-are-staged.html#stage-buildpack

Tarnoff, B. 2016c. Using OpenStack Swift as a Cloud Foundry Blobstore. Cloud Foundryn dokumentaatio. Viitattu 17.4.2017. https://docs.cloudfoundry.org/deploying/openstack/using_swift_blobstore.html

Tsidulko, J. 2016. Get On Board: Docker's Channel Maturity Unlocks The Container Tech Opportunity. CRN:n artikkeli. Viitattu 13.4.2017. http://www.crn.com/news/applications-os/300083111/get-on-board-dockers-channel-maturity-unlocks-the-container-tech-opportunity.htm

Vie liiketoimintasi kehitys pilveen. 2017. Telian tuotesivut. Viitattu 3.4.2017. https://www.telia.fi/yrityksille/tuotteet/tietoliikenne/telia-data-center/pilvipalvelu

Villarreal, J. 2017. A quick introduction to OpenStack Heat. Superuserin artikkeli. Viitattu 17.3.2017. http://superuser.OpenStack.org/articles/quick-intro-OpenStack-heat/

Wang, C. 2015. OpenStack - Swift Dev Box - SAIO on Ubuntu 14.04 via VirtualBox. Wang blogikirjoitus. Viitattu 18.3.2017. http://chianingwang.blogspot.fi/2015/01/OpenStack-swift-dev-box-saio-on-ubuntu.html

Wang, C. 2016. Containers 101: Linux containers and Docker explained. InfoWorldin artikkeli. Viitattu 18.3.2017. http://www.infoworld.com/article/3072929/linux/containers-101-linux-containers-and-docker-explained.html

What is Linux container?. N.d. Red Hatin kotisivut. Viitattu 5.12.2016. https://www.redhat.com/en/containers/whats-a-linux-container

Williams, A. 2014. Docker on Diego, Cloud Foundry's New Elastic Runtime. TheNewStackin artikkeli. https://thenewstack.io/docker-on-diego-cloud-foundrys-new-elastic-runtime/

Williams, A. 2016. THE EMERGING CONTAINERS AS A SERVICE MARKETPLACE. Thenewstackin artikkeli. Viitattu 12.4.2017. https://thenewstack.io/emerging-containers-service-marketplace/

Williams, D. 2017. CNI – the Container Network Interface. CNI Github. Viitattu 28.2.2017. https://github.com/containernetworking/cni

Xu, M. 2017. Orgs, Spaces, Roles, and Permissions. Cloud Foundry dokumentaatio. Viitattu 15.3.2017. https://docs.cloudfoundry.org/concepts/roles.html

Liitteet

Liite 1. OpenStack arkkitehtuuri (Architecture 2017.)