Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Aleksi Telkkä

Kubernetes-asennuksen automatisointi ja integrointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

12.4.2020

Tekijä Otsikko Sivumäärä Aika	Aleksi Telkkä Kubernetes-asennuksen automatisointi ja integrointi 26 12.4.2020
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Pelisovellukset
Ohjaaja	Yliopettaja Kari Aaltonen

Insinöörityön tarkoituksena oli perehtyä konttiteknologiaan, joka mahdollistaa tehokkaan infrastruktuurin käytön ja sovellusten hallinnan. Tavoitteena oli myös luoda järjestelmä, jossa konttien edut voidaan tutkia ja asennus automatisoida. Toimeksiantajana oli suomalainen graafisen alan ohjelmistoyritys.

Konttien hallintajärjestelmäksi valittiin Kubernetes-järjestelmä. Insinöörityössä luotiin ohjelmisto, jonka avulla Kubernetes-asennus ja konttien luonti voidaan suorittaa automaattisesti, integroituna toimeksiantajan omaan järjestelmään.

Aluksi tutustuttiin virtualisointiin ja konttiteknologioihin sekä Kuberneteksen dokumentaatioon, minkä jälkeen siirryttiin itse toteutukseen. Kubernetes-järjestelmän asennusohjelma luotiin Shell-skripteinä, jotka integroitiin toimeksiantajan hallintajärjestelmään käyttäen C++ -ohjelmointikieltä.

Kubernetes osoittautui monimutkaiseksi teknologiaksi, ja sen käyttöön liittyy paljon asioita, mitkä täytyy ymmärtää, jotta sitä voi hyödyntää. Sen avulla on kuitenkin mahdollista muuttaa perinteisiä käsin ylläpidettäviä IT-ympäristöjä ja saavuttaa merkittäviä etuja.

Insinöörityön lopputuloksena syntyi asennusohjelma, joka automatisoi Kubernetes-järjestelmän asentamisen ja tarjoaa toimeksiantajalle hyvän pohjan jatkokehitystä varten.

Avainsanat	kontti, Kubernetes, klusteri

Author Title Number of Pages Date	Aleksi Telkkä Automating the installation and integrating Kubernetes cluster 26 12.4.2020
Degree	Bachelor of Engineering
Degree Programme	Information and Communications Technology
Professional Major	Game Applications
Instructor	Kari Aaltonen, Principal Lecturer

The goal of this thesis was to get familiarized with container technologies and to create a system which the employer could use to find the best working solution for them. Containers enable efficient usage of server infrastructure and application management. The employer was a Finnish software company in graphic industry.

Kubernetes was chosen for the container orchestration system. In this thesis a program was created, that would automate the installation of Kubernetes and setting up containers. The program was integrated into the employer's own system.

The first steps in this project were getting familiarized with virtualization and container technologies and the documentation of Kubernetes. Next, ShellScripts were made to handle the installation of Kubernetes and said ShellScripts were used to create a program in C++ which was integrated into the employer's own system.

Kubernetes proved to be a complicated system and its usage includes a lot of things one needs to understand in order to make the most of it. With Kubernetes it is possible to change traditional virtualization-based server infrastructure into a more efficient one.

As a result of this thesis, a program which automated the installation of Kubernetes integrated into their own system was created. It creates a solid foundation for the employer to further build on it.

Keywords	Container, Kubernetes, Cluster

Sisällys

Lyhenteet ja termit

1 Johdanto 1

2 Ohjelmistoyritys Anygraaf 2

3 Teknologiat tehokkaaseen infrastruktuurin käyttöön ja sovellusten hallintaan 3

3.1 Palvelinten virtualisointi 4

3.2 Sovelluskontit 5

4 Konttien luonti, suorittaminen ja hallinta 7

4.1 Docker ja konttien luonti 7

4.2 Kubernetes-järjestelmä konttien suorittamiseen ja hallintaan 8

5 Kubernetes-järjestelmän asentaminen 10

5.1 Yleistä asentamisesta 10

5.2 Asentamista varten luodut skriptit 11

5.3 Yrityskohtaisten palveluiden asentaminen 17

6 Integrointi yrityksen mediasisällön suunnittelu- ja hallintajärjestelmään 18

6.1 Neo-järjestelmän toiminnallinen yleiskuvaus 18

6.2 Neo-järjestelmän arkkitehtuuri 20

6.3 Kubernetes-asennusohjelmiston integrointi Neo-järjestelmään 22

7 Havainnot ja jatkosuunnitelmat 24

8 Yhteenveto 25

Lähteet 27

Lyhenteet ja termit

SQL	Structured Query Language. IBM:n kehittämä standardoitu kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja.
PostgreSQL	Avoimena lähdekoodina jaettava tietokannan hallintajärjestelmä.
SaaS	Software as a Service. Ohjelmisto hankitaan palveluna perinteisen lisenssipohjaisen tavan ja ohjelmiston asentamisen sijaan. Asiakaskohtaisia tuotantoympäristöjä ei ole, vaan sama tuotantoympäristö palvelee useaa asiakasta.
CISF	Common Internet File System. CIFS on uusi nimi SMB-verkkotiedostojärjestelmälle. SMB-lyhenne tulee sanoista Server Message Block. CIFS-nimen käyttöönoton yhteydessä Microsoft lisäsi SMB:hen joitakin ominaisuuksia. Nykyisin SMB ja CIFS ovat käytännössä synonyymejä.
Samba	SMB- ja CIFS-toiminnallisuutta Linuxissa tarjoaa Samba-ohjelmistopaketti.
REST	Representational State Transfer. Arkkitehtuurimalli rajapintojen luomiseen.
API	Application Programming Interface. Ohjelmointirajapinta, jonka kautta eri ohjelmat voivat keskustella toistensa kanssa.
CLI	Command-line interface. Komentoriviltä käytettävä sovellus.
JSON	JavaScript Object Notation. Avoimen standardin tiedostomuoto tiedonvälitykseen.
YAML	YAML Ain't Markup Language. Kehityksen alkuvaiheessa nimen sanottiin tulevan sanoista ”Yet Another Markup Language”. YAML:ä käytetään usein konfiguraatiotiedostoissa, mutta sitä voi käyttää myös muunlaiseen tietojen tallennukseen.
gRPC	gRPC Remote Procedure Calls. Avoimen lähdekoodin RPC-järjestelmän kehitti alun perin Google vuonna 2015. Yleisimmät käyttökohteet ovat

palveluitten yhdistäminen mikropalveluarkkitehtuurin tapaan ja mobiililaitteiden, selainkäytön ja taustapalveluitten yhdistäminen.

Apache — Apache HTTP Server on avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma. Apachen HTTP-palvelin on Apache Software Foundationin tunnetuin tuote, ja se on saatavilla Unixille (BSD:t, Linux), Windowsille ja useille muille käyttöjärjestelmille, ja lisäksi se on integroitu OS X -käyttöjärjestelmään.

Hypervisor — Hypervisor (myös virtual machine monitor, VMM) on virtuaalikoneen suorittamiseen käytetty ohjelmistokerros. Kerros suorittaa ohjelmia kuten tietokone, mukaan lukien sen käyttöjärjestelmä. Hypervisorin alaisuudessa toimiva käyttöjärjestelmä voi suorittaa tavallisia ohjelmia normaalisti.

1 Johdanto

Insinöörityön toimeksiantajana on suomalainen graafisen alan ohjelmistoyritys Anygraaf Oy, jonka tämänhetkinen palveluratkaisu koostuu useasta tehokkaasta palvelinkoneesta ja niihin toteutetuista useammasta kymmenenestä virtuaalikoneesta, jotka taas hoitavat eri asiakkaiden palvelut. Näiden virtuaalikoneitten luominen on helppoa mallinepohjaisesti (template-pohjaisesti), mutta niiden heikkous on se, että kun virtuaalikoneissa on aina käyttöjärjestelmän pakolliset taustaprosessit, nettopalvelun tuottamia prosesseja on vain muutama suhteessa lukuisiin taustaprosesseihin virtuaalikonetta kohti. Anygraaf uskoo, että konttiteknologialla saavutetaan parempi teho samasta laitteistosta. Insinöörityössä luodun järjestelmän avulla on tarkoituksena päästä testaamaan konttiteknologian avulla toteutetun ratkaisun tehokkuutta ja helpottaa itse perusjärjestelmän asennusta ja konttien luontia siten, että asennuksen pystyy suorittamaan ”kuka tahansa”.

Aluksi tässä insinöörityöraportissa kerrotaan taustaa työn toimeksiantajasta ja kuvataan toimeksianto tarkemmin. Seuraavassa luvussa esitellään yleiskatsauksena mahdollistavat teknologiat tehokkaaseen infrastruktuurin käyttöön ja sovellusten hallintaan. Sen jälkeen luvussa 4 siirrytään kuvaamaan, miten kontteja luodaan, suoritetaan ja hallinnoidaan.

Luku 5 keskittyy Anygraafin valinnan mukaisesti nimenomaan Kubernetes-järjestelmän asentamiseen. Asentamista varten luotiin skriptit, ja ne esitellään siinä järjestyksessä, missä ne tulee suorittaa asentamisen yhteydessä.

Seuraavaksi luvussa 6 kerrotaan, miten luodut asennusskriptit integroitiin osaksi Anygraafin Neo-järjestelmää. Kokonaisuuden ymmärtämiseksi esitellään ensin Neo-järjestelmän toiminnallisuudet ja tekninen arkkitehtuuri, ja sen jälkeen keskitytään kuvaamaan integroinnin toteutus.

Lopuksi esitetään työn aikana ja lopuksi tehdyt havainnot, erityisesti toimeksiantajan tavoitteiden saavuttamisen näkökulmasta, ja kerrotaan, miten toimeksiantaja aikoo jatkaa tehdyn työn hyödyntämistä.

2 Ohjelmistoyritys Anygraaf

Insinöörityön toimeksiantaja Anygraaf Oy on vuonna 1996 perustettu suomalainen graafisen alan ohjelmistoyritys, jonka tuoteportfolio kattaa kaikki keskeisimmät komponentit sisältöjen suunnittelusta julkaisuun. Työntekijöitä Anygraafissa on noin 50, ja noin 80 henkilöä työskentelee päivittäin Anygraaf konsernissa jälleenmyyjien palveluksessa. Anygraafin tuotteiden asennuksia on 18 eri maassa, ja niitä hyödyntää yli 400 asiakasorganisaatiota ja yli 15 000 loppukäyttäjää. [1.]

Lehtialalla on käynnissä murros. Yhä useammat pienet lehdet ovat päätyneet siihen, että niillä ei ole tarvittavaa osaamista oman IT-infrastruktuurinsa hallitsemiseen, eivätkä ne pidä liiketaloudellisesti kannattavana hankkia omaa osaamista siihen. Lehdet haluavat ostaa koko tietojärjestelmänsä palveluna (ns. SaaS-palvelu). Lehtien tietojärjestelmät koostuvat useimmiten toimitusjärjestelmästä, ilmoitusten myynti-, hallinta- ja markkinointijärjestelmästä, tilausten myynti- ja hallintajärjestelmästä sekä sähköisestä julkaisujärjestelmästä.

Anygraafin tämänhetkinen palvelinratkaisu koostuu 12 tehokkaasta Supermikron palvelinkoneesta. Niissä suoritetaan Xenserver-klusteria, johon on toteutettu useampi kymmenen virtuaalikoneita, jotka taas hoitavat eri asiakkaiden palvelut. Teknisesti palvelut koostuvat lähinnä SQL (Structured Query Language) -tietokantapalvelusta (tarkemmin PostgresSQL), CIFS-tiedostopalvelusta (Common Internet Filesystem, useimmiten SAMBA) ja www-palvelusta.

Tarkoituksena insinöörityössä oli perehtyä konttiteknologiaan ja luoda järjestelmä, jossa konttiteknologian edut voidaan tutkia ja asennus automatisoida. Kontteja on mahdollista suorittaa joko Docker Swarm- tai alun perin Googlen kehittämässä Kubernetes-järjestelmässä. Anygraaf uskoo, että näistä Kubernetes tarjoaa enemmän vaihtoehtoja ammattimaiseen käyttöön, vaikka se onkin monimutkaisen asennettavan ja ylläpidettävän maineessa.

Insinöörityössä luotiin järjestelmä, jonka avulla tehdään Kuberneteksen asennus ja konttien luonti helposti ja automaattisesti, integroituna Anygraafin Neo-ohjelmistoon.

3 Teknologiat tehokkaaseen infrastruktuurin käyttöön ja sovellusten hallintaan

Palveluntarjoajilla on tarve pitää eri sovelluksia erillään. Syitä tähän tarpeeseen ovat muun muassa tietoturva ja sovelluksien yhteensopivuus. Mikäli jostakin sovelluksesta löytyisi haavoittuvuus, olisi riski, että kyseisen tietoturva-aukon kautta hyökkääjä pääsisi palveluntarjoajan muihinkin järjestelmiin. Jotkin sovellukset tarvitsevat eri version samasta kirjastosta, mutta usean version pitäminen samasta kirjastosta yhdellä tietokoneella ei ole aina mahdollista. Tämä ajatusmalli on vahvasti esillä myös toisaalla tietotekniikassa, esimerkiksi olio-ohjelmoinnissa. Mahdolliset virheet on helpompi paikantaa ja korjata eikä koko järjestelmä ole vaarassa kaatua, kun se on jaettu pieniin osiin, jotka muodostavat isomman kokonaisuuden.

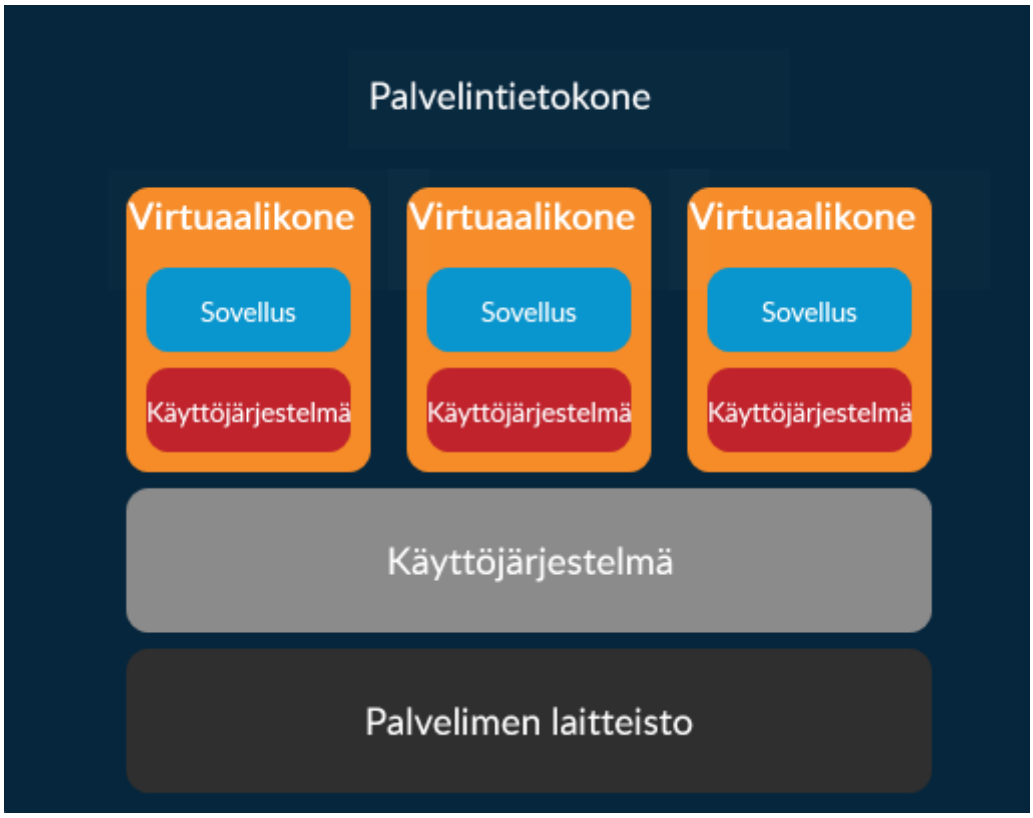
Palveluntarjoajat etsivät myös keinoja käyttää infrastruktuuriaan mahdollisimman tehokkaasti. Aikaisemmin sovellusten erillään pitäminen on vaatinut palveluntarjoajan sijoittamaan useampaan palvelimeen, kuten kuvassa 1 esitetään. Tämä vaatii enemmän rahaa ja tilaa eikä tietokoneen kaikkia resursseja pystytä aina hyödyntämään täysipainoisesti. Palvelintietokoneet vaativat usein oman huoneen ilmastointijärjestelmineen palvelinkoneiden tuottaman melun ja lämmön vuoksi. Aikaa tuhlaantuu myös usean palvelinkoneen huoltoon, ylläpitoon ja asennuksiin. [2.]

Kuva 1. Kaksi erillistä palvelinta tarvitaan, kun sovellukset vaativat eri käyttöjärjestelmät.

3.1 Palvelinten virtualisointi

Tilaa saadaan säästettyä ja resursseja paremmin hyötykäytettyä luomalla virtuaalikoneita. Tämä tarkoittaa käytännössä sitä, että yhdessä palvelimessa on kaksi tai useampia itsenäisiä tietokoneita. Virtualisoinnissa virtuaalikoneet saavat niille määritellyn määrän palvelinkoneen resursseista. Jokainen virtuaalikone käyttäytyy kuin mikä tahansa muu itsenäinen tietokone, eli sillä voi suorittaa samoja asioita samalla tavalla kuin tavallisella fyysisellä tietokoneella. Aivan kuten tavallinen fyysinen tietokone, virtuaalinen tietokone vaatii myös käyttöjärjestelmän, tosin virtuaalisen tietokoneen käyttöjärjestelmä voi olla eri kuin fyysisen tietokoneen, jolla virtuaalikone on.

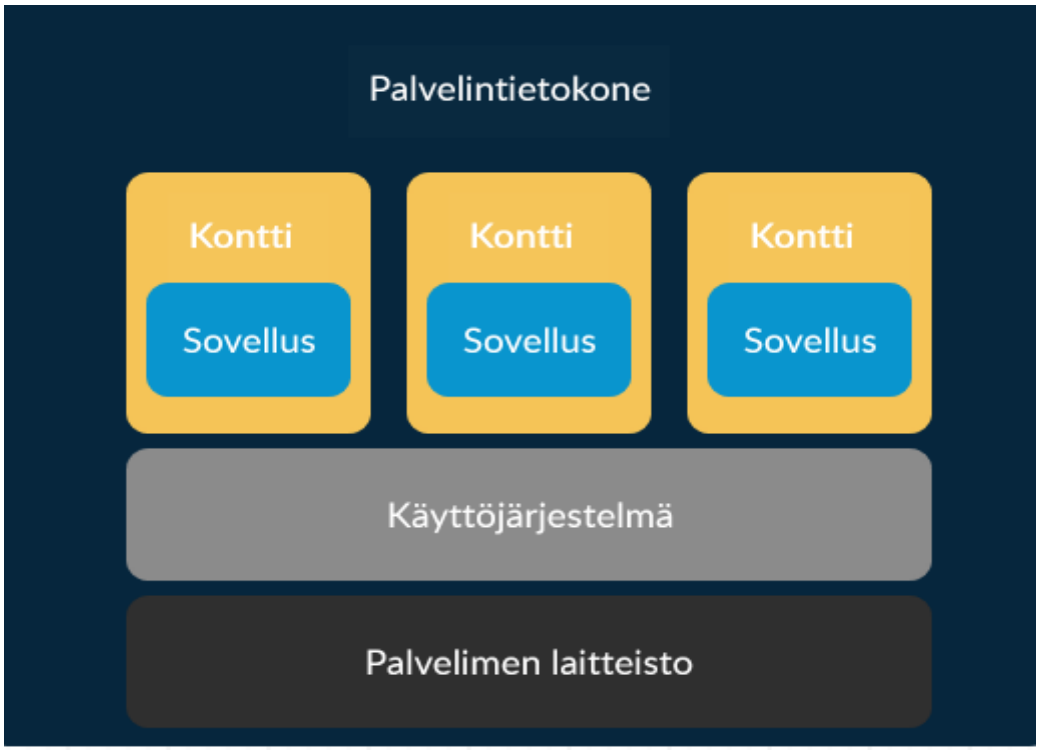
Palveluntarjoajan näkökulmasta virtuaaliset koneet lisäävät kustannustehokkuutta, mutta jättävät vielä parantamisen varaa. Virtuaalikoneen tarve käyttöjärjestelmään aiheuttaa suuren määrän hukkaresursseja tietokoneessa. Käyttöjärjestelmät vaativat usein suuren määrän muistia ja prosessointiaikaa. Kuten kuvasta 2 näkyy, jokaisella yksittäisellä virtuaalikoneella on oma käyttöjärjestelmänsä fyysisen tietokoneen käyttöjärjestelmän päällä. Tämä tarkoittaa, että fyysisen koneen ja virtuaalisten koneiden luomassa kokonaisuudessa on useaan otteeseen samat taustaprosessit tuhlaamassa resursseja. [3.]



Kuva 2. Palvelintietokone ja virtuaalikoneet.

3.2 Sovelluskontit

Konttien avulla päästään eroon virtuaalikoneiden tarpeesta omaan käyttöjärjestelmään, koska tällöin riittää, että vain fyysisessä tietokoneessa on käyttöjärjestelmä asennettuna. Kontteja hyödyntämällä sovellukset pysyvät ohjelmistotasolla erillään, eikä niitä käyttämällä tarvita useampaa käyttöjärjestelmää, jolloin sovellusten suorittamisen kannalta ei synny turhia taustaprosesseja. Kontissa suoritetaan siis ainoastaan niitä prosesseja, mitkä ovat välttämättömiä sovelluksen suorittamiseen. Tämä säästää huomattavan määrän tietokoneen rajallisia resursseja. Kuvassa 3 näkyy esimerkki, jossa yksi käyttöjärjestelmä palvelee kolmea konttia. [4.]



Kuva 3. Palvelintietokone ja kontit.

Konttien toiminta perustuu kahteen eri toimintoon, jotka ovat Linux-käyttöjärjestelmän ytimessä: nimiavaruudet (namespaces) ja cgroups (control groups). Nimiavaruuksien avulla jokaisella kontilla on oma nimiavaruutensa, jonka sisällä suoritettavilla prosesseilla ei ole oikeuksia toimia niiden oman nimiavaruuden ulkopuolella. Cgroups-ominaisuuden avulla voidaan asettaa rajoituksia prosessien resurssien käyttöön, kuten muistin tai prosessorin käyttöön. Cgroups kykenee myös pysäyttämään, lopettamaan ja käynnistämään sovelluksia. Näiden oikeuksien vuoksi cgroups estää muistivuotoja, mikä puolestaan estää sen, että kaikki palvelimella olevat prosessit kaatuisivat tai hidastuisivat. Cgroups on alun perin Googlen kehittämä ja lisätty Linuxin ytimeen vuonna 2008. [5.]

Toisin kuin virtuaalikoneissa, kontteja käyttäessä eristämiseen ei tarvita ns. hypervisor-virtualisointia. Hypervisor on virtuaalikoneen suorittamiseen käytetty ohjelmistokerros, eli virtuaalikoneissa koko laitteistoa emuloidaan ja sen sisällä suoritetaan toista täyttä käyttöjärjestelmää. Tämän vuoksi konteilla sovelluksien eristäminen on huomattavasti kevyempää tietokoneelle. Konttien eristämisen mukana tuleva ajattelutapa tukee ohjelmistoarkkitehtuuria, jossa sovellusohjelma rakennetaan erillisistä pienistä

palveluista. Mikropalveluksi kutsuttua ohjelmistoarkkitehtuuria sovelletaan eniten jatkuvan integraation ja jatkuvan toimituksen kehitysmallien yhteydessä. Vaikka kontit ja mikropalveluajattelutapa säästävät tietokoneen resursseja, kannattaa silti ohjelmistoarkkitehtuurissa miettiä, mitkä palvelut kannattaa laittaa eri kontteihin ja mitkä ei. [6.]

4 Konttien luonti, suorittaminen ja hallinta

4.1 Docker ja konttien luonti

Konttien luontiin ja hallintaan on monia työkaluja. Niistä yleisin ja tässäkin työssä käytetty työkalu on Docker. Se koostuu useasta peruskomponentista: Docker Engine, Docker Image, Docker File, Docker Container ja Registry. Docker Engine on sananmukaisesti moottori, jonka avulla voidaan rakentaa kontteja ja suorittaa niitä. Docker File-tiedosto kertoo, kuinka Docker Image rakennetaan. Docker Image on tiedostokokonaisuus, joka sisältää sovelluksen sekä sen vaatimat ohjelmistot ja käyttöjärjestelmän. Image tekee sovelluksista todella helposti siirrettäviä. Sen avulla voidaan sovellus siirtää nopeasti ja helposti esimerkiksi kannettavalta tietokoneelta pilveen pienen tiedostokoon vuoksi. Samoin sovelluksen suorittaminen onnistuu uudessa paikassa helposti ja samalla tavalla kuin kannettavallakin, kunhan Docker on asennettuna. Docker Container on kontti, joka syntyy Docker Imagen perusteella. Registry on Docker Imagejen säilytyspaikka. Säilytyspaikkoja on monia, joista osa on julkisia, kuten Dockerin kehittäjien virallinen kirjasto, jonka sisältöä he itse kuratoivat. Registry voi olla myös yksityinen. Tässä työssä käytetyt Docker Imaget vedettiin Anygraafin omasta säilytyspaikasta. Näiden komponenttien lisäksi tarvitaan vielä konttien hallinnointiin työkalu, mikäli kontteja halutaan suorittaa useammalla palvelimella. Docker on itse kehittänyt tähän helppokäyttöisen Docker Swarm-hallinnointityökalun. Tässä työssä hallinnointiin käytetään kuitenkin Kubernetesta.

Docker nopeuttaa sovelluskehitystä ja konttiteknologian takia vähentää IT-infrastruktuurin kuluja. Docker-kontit käynnistyvät myös virtuaalikoneita huomattavasti nopeammin, sillä niiden ei tarvitse käynnistää käyttöjärjestelmää. Docker Imaget tekevät

sovelluksista erittäin helposti siirrettäviä, mikä tekee testaamisen erilaisissa ympäristöissä helpoksi. [7.]

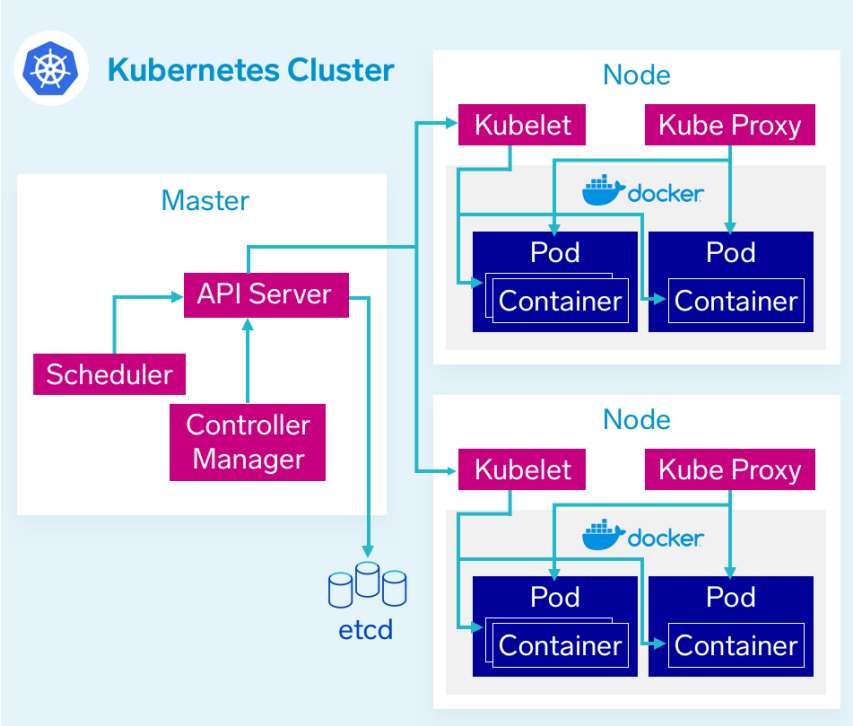
4.2 Kubernetes-järjestelmä konttien suorittamiseen ja hallintaan

Kubernetes on alun perin Googlen kehittämä, mutta perustuu nykyään avoimeen lähdekoodiin ja on Cloud Native Computing Foundationin ylläpitämä järjestelmä konttien ajamiseen ja hallintaan. Kubernetes on yhteensopiva monien konttityökalujen kanssa, kuten Dockerin. Se ei siis itse luo kontteja, vaan on konttien hallinnointityökalu. Järjestelmän avulla voidaan hallita, mikä ryhmä kontteja suoritetaan milläkin Kubernetes-klusteriin liitetyllä palvelimella. Järjestelmän avulla voidaan myös seurata sitä, toimivatko kontit halutulla tavalla, ja se yhdistää yhdeksi laskentaresurssiksi kaikki palvelimet, jotka on siihen klusteriin liitetty. Järjestelmä optimoi ja automatisoi tämän yhden suuren laskentaresurssin käyttöä, eikä tässä tapauksessa tarvita ihmistä päättämään, millä palvelimella mikäkin kontti suoritetaan. [8.]

Kubernetestä tarvitaan, kun halutaan käynnistää tuotantoympäristöön sovellus, joka koostuu useasta kontista. Se tietää, mitkä ovat sovellukselle vaadittavia kontteja, ja osaa käynnistää ne, ja se hoitaa konttien sijoittamisen niille palvelimille, jotka on kyseiseen klusteriin liitetty. Järjestelmä pitää kirjaa, mitä kontteja suoritetaan milläkin alustapalvelimella. Vikaantuneet kontit voidaan asettaa käynnistymään uudelleen automaattisesti. Järjestelmä osaa myös skaalata sovelluksen automaattisesti. Jos kuormitus lisääntyy ja sovellus tarvitsee lisää kapasiteettia, se käynnistää uusia kontteja ja huolehtii niiden välisen kuorman jakamisesta. Sovelluksia, jotka koostuvat useammasta kontista, voidaan myös helposti päivittää. Päivityksessä järjestelmä osaa tehdä muutokset kontteihin, ja mikäli päivitykset eivät tuota haluttua tulosta, voidaan aikaisempi kontti ottaa käyttöön. Järjestelmä huolehtii sovelluksen julkaisemisesta jakamalla konteille niiden vaatimat IP-osoitteet. [9.]

Kubernetes-järjestelmän klusteri koostuu yhdestä sitä hallinnoivasta palvelimesta (master node) ja useammasta suorittavasta palvelimesta (worker node), kuten kuvassa 4 on esitetty. Master on klusterin aivot. Se päättää, mitä konttia milläkin palvelimella suoritetaan, eli se jakaa työkuorman suorittaville palvelimille. Suorittava palvelin voi olla joko fyysinen tai virtuaalinen tietokone. Master jakautuu neljään eri komponenttiin: API

Server, Scheduler, Controller Manager ja Etcd. API Server on rajapinta Masterin ja työkuormaa suorittavien palvelimien välillä. Scheduler näyttää tilanteen ja tarpeen mukaan sovelluksille (pod) paikat eri palvelimien välillä. Controller Manager vahtii työkuormaa suorittavia palvelimia ja käynnistää kaatuneita tai muuten viallisia komponentteja uudestaan. Etcd on klusteria hallinnoivan palvelimen tietokanta, jossa on Kubernetes-klusterin konfiguraatiotiedot tallennettuina. [10.]



Kuva 4. Kubernetes-klusteri. [11.]

Työkuormaa suorittavassa palvelimessa on myös useita komponentteja. Kubelet-komponentin kautta työkuormaa suorittava palvelin keskustelee klusteria hallinnoivan palvelimen kanssa ja kubelet varmistaa, että pod-oliot ovat konttien sisällä. Kube-proxyn kautta työkuormaa suorittava palvelin pystyy keskustelemaan klusterin ulkopuolisen maailman kanssa. Se siis julkaisee sovellukset maailmalle. Container Runtime on kontteja suorittava komponentti, kuten Docker. Sen tehtävänä on suorittaa kontteja kyseisellä palvelimella. [10.]

Komentoja Kubernetekselle annetaan kubectl-nimisen CLI-komentorivityökalun kautta. Komennot tulevat klusterin API-palvelimelle JSON-formaatissa. Kuberneteksessä on

kolme oleellista oliota, jotka ovat pod, service ja deployment. Oliot kuvataan joko YAML- tai JSON-tekstitiedostoina, jotka sisältävät muun muassa olion nimen sekä oliolle halutun tilan. Samoin kuin Linux-käyttöjärjestelmässä, kubectl-työkalun komennoille voi antaa tarkentavia lisäehtoja, kuten esimerkkikoodissa 1 on näytetty. [12; 13.]

```
Kubectl -n kubernetes-dashboard get service kubernetes-dashboard
```

Esimerkkikoodi 1. -n-merkinnän avulla voidaan hakea tietystä nimiavaruudesta.

Alin käsite Kubernetes-klusterissa on pod-olio. Sen sisällä ovat suoritettavat kontit. Kontteja voi tarvittaessa olla enemmän kuin yksi yhden podin sisällä. Saman podin sisällä suoritettavat kontit käyttävät samaa nimiavaruutta, eli niillä on yhteinen muisti. Pod-oliolla on useampi tila, jossa olio voi olla. Olion tilaa voidaan tarkastella get pod- ja vielä perusteellisemmin describe pod -komennon kautta tai selaimen kautta Kubernetes-komentopaneelilla. Pod-oliot saavat syntyessään satunnaisen IP-osoitteen, joten niiden kanssa kommunikoimiseen tai kontissa suoritettavan sovelluksen käyttöä varten on service-olio. Se pitää kirjaa seuraamistaan pod-olioista ja niiden IP-osoitteista. Service-olion IP-osoite pysyy aina samana, joten sitä kautta on helpointa päästä pod-olioiden sisältöön kiinni. Deployment-olio sisältää useamman identtisen kopion halutuista pod-oliosta ja automaattisesti käynnistää uudelleen kaatuneet palvelut. Deployment-olion avulla pod-olion sisällä olevat palvelut pysyvät koko ajan käytettävissä. [9; 13.]

5 Kubernetes-järjestelmän asentaminen

5.1 Yleistä asentamisesta

Kubernetes-järjestelmän asentaminen ei ole triviaali tehtävä, vaikka useitakin ohjeita löytyy. Jatkuvan ja voimakkaan kehityksen takia osa ohjeista ja dokumentaatioista laahaa perässä. Virallista dokumentaatiota tulee seurata tarkkaan, sillä samoin kuin itse järjestelmää, sen dokumentaatiotakin päivitetään usein. Järjestelmän asentaminen ja käyttäminen vaativat käyttäjältä vähintään perustason Linux-osaamista. Tämä asennus käsittelee Kubernetes 1.17 -version asennusta.

Ennen kuin itse asennukseen päästään, tulee varmistaa, että laite, jolle asennus tehdään, täyttää laitevaatimukset. Jokaisella laitteella, jolle Kubernetes halutaan asentaa, tulisi olla vähintään 2 GB muistia, kaksi tai useampia ytimiä sekä Kuberneteksen kanssa yhteensopiva Linux-käyttöjärjestelmä. Laitteilla tulee olla myös pääsy verkkoon. Konttien suorittamiseen tarvitaan myös ohjelmisto, joka on Kuberneteksen kanssa yhteensopiva, esimerkiksi Docker. Konttien suorittamiseen käytettävän ohjelmiston version ja Kuberneteksen version yhteensopivuudessa pitää olla tarkka: vaikka ohjelmistot ovat yhteensopivat, ohjelmistojen eri versioissa saattaa olla ongelmia yhteensopivuuden kanssa. Klusterin voi tilata valmiina myös toisilta palveluntarjoajilta, mikäli oma palvelininfrastruktuuri puuttuu. [14.]

5.2 Asentamista varten luodut skriptit

Asentamista varten luotiin viisi erillistä Shell-skriptiä, jotka on lueteltu taulukossa 1. Skiptejä käytettiin hyödyksi, kun Kubernetes-asennus intergroitiin Anygraafin Neo-järjestelmään (luku 6).

Taulukko 1. Kubernetes-asennukseen tehdyt skiptit asennusjärjestyksessä.

Skripti	**Toiminnallisuus**
Kubeprereq.sh	Asettaa asennuksen esivaatimukset oikein
Kubeinstall.sh	Asentaa Kuberneteksen
Kubedashboard.sh	Asentaa Kuberneteksen komentopaneelin (lisäosa)
Kubeag.sh	Asentaa Anygraafin palvelut Kubernetekseen
Kubereset.sh	Nollaa Kuberneteksen

Asennuksen alustus

Itse asentamisessa ensimmäisenä tehtävänä on konfiguroida iptables-asetukset oikein ja kytkeä Linux-käyttöjärjestelmän Swap-ominaisuus pysyvästi pois päältä, kuten

esimerkkikoodissa 2 näytetään. Swap liittyy Linuxin tapaan käsitellä tilanteita, kun muisti loppuu, mutta Kubernetes ei tue tämän ominaisuuden toimintaa. [15; 14.]

```
sudo swapoff -a
sudo sed -i 's/\/swap/#\/swap/' /etc/fstab
cat <<EOF > /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sysctl -system
```

Esimerkkikoodi 2. Linux-asennuksien konfigurointi.

Seuraavaksi tulee asentaa konttien suorittamiseen tarkoitettu ohjelmisto, tässä tapauksessa Docker, jonka asennus on esitetty esimerkkikoodissa 3. Ennen Dockerin asentamista on etsittävä Kubernetes-klusterin kanssa yhteensopiva versio Dockerista. Yksinkertaisuuden ja järjestelmän toimivuuden vuoksi Dockerin cgroup-ajuriksi asetetaan systemd. Vaihtoehtona olisi konfiguroida klusteri hyödyntämään useampaa cgroup-ajuria. Useamman cgroup-ajurin käyttö tarkoittaa, että samoista resursseista on useita eri näkymiä, jotka mahdollisesti eroavat toisistaan, sillä cgroup-ajurit näkevät samat resurssit itsenäisesti. Tämä saattaa joskus johtaa ongelmiin. [16.]

```
## Install Docker CE.
sudo apt-get update && sudo apt-get install -y \
  containerd.io=1.2.10-3 \
  docker-ce=5:19.03.4~3-0~ubuntu-$(lsb_release -cs) \
  docker-ce-cli=5:19.03.4~3-0~ubuntu-$(lsb_release -cs)
## Setup daemon.
cat << EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
sudo mkdir -p /etc/systemd/system/docker.service.d
## Restart docker.
sudo systemctl daemon-reload
sudo systemctl restart docker
sudo systemctl enable docker
sudo apt-mark hold containerd docker-ce docker-ce-cli
sudo reboot
```

Esimerkkikoodi 3. Dockerin asennus.

Vielä ennen itse Kuberneteksen asentamista tulee tietokone käynnistää uudelleen ja tarkistaa, että Docker asennus onnistui ja että Docker on päällä tietokoneen käynnistyttyä.

Asennus

Seuraavaksi on vuorossa itse Kuberneteksen asentaminen. Sitä varten tulee asentaa kolme eri komponenttia, jotka luovat Kuberneteksen: kubeadm, kubectl ja kubelet. Koska nämä kolme komponenttia ovat välttämättömiä Kuberneteksen toimivuuden kannalta, niitten päivittämisessä tulee olla erittäin tarkka, sillä eri versiot eivät välttämättä ole yhteen sopivia toistensa kanssa. Jotta tietokoneen muiden osien päivittäminen ei rikkoisi klusterin toimivuutta, kannattaa Dockerin ja Kuberneteksen toiminnan kannalta tärkeimmiltä komponenteilta estää päivittyminen. Tämän voi tehdä Linuxin apt-mark hold -komennolla. Näin estetään merkittyjen komponenttien päivittyminen, kun muu tietokone päivitetään. Välttämättömien osien päivittämisen kanssa tulee olla erittäin huolellinen, jotta klusterin toiminta jatkuu eikä kaadu toistensa kanssa epäyhteensopiviin osiin. Esimerkkikoodissa 4 on nähtävillä osa asennusskriptista, jossa näkyy keskeisin osa, eli itse Kuberneteksen asennus.

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key
add -
cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update -y
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Esimerkkikoodi 4. Leike Kuberneteksen asennusskriptista.

Tässä kohtaa voi klusterin käynnistää kubeadm-työkalun kautta. Kubeadm init -käsky käynnistää Kuberneteksen. Käynnistymisessä voi kulua useitakin minuutteja. Mikäli itse asennus on suoritettu oikein, pitäisi ruudulle ilmestyä tulostus, joka ilmoittaa klusterin toimivan ja kehottaa vielä suorittamaan pari komentoa.

Kuvassa 5 näkyy onnistuneen asennuksen tulostus.

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a Pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  /docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node
as root:

  kubeadm join <control-plane-host>:<control-plane-port> --token <token> --discovery-token-ca-cert-hash sha256:<hash>
```

Kuva 5. Tulostus onnistuneesta Kubernetes-asennuksesta.

Jotta pod-oliot kykenevät keskustelemaan toistensa kanssa, pitää vielä asentaa CNI (Container Network Interface) -lisäosa. Niitä on useita, eri ominaisuuksilla varustettuina. Kun tarpeet kattava CNI-lisäosa on asennettu, voi klusteriin liittää toisia tietokoneita, virtuaalisia tai fyysisiä. Liitettävillä työkuormaa suorittavilla tietokoneilla on samat vaatimukset kuin klusteria hallinnoivalla tietokoneella. Liittyminen klusteriin suoritetaan liitettäviltä tietokoneilta käyttäen kubeadm join -komentoa.

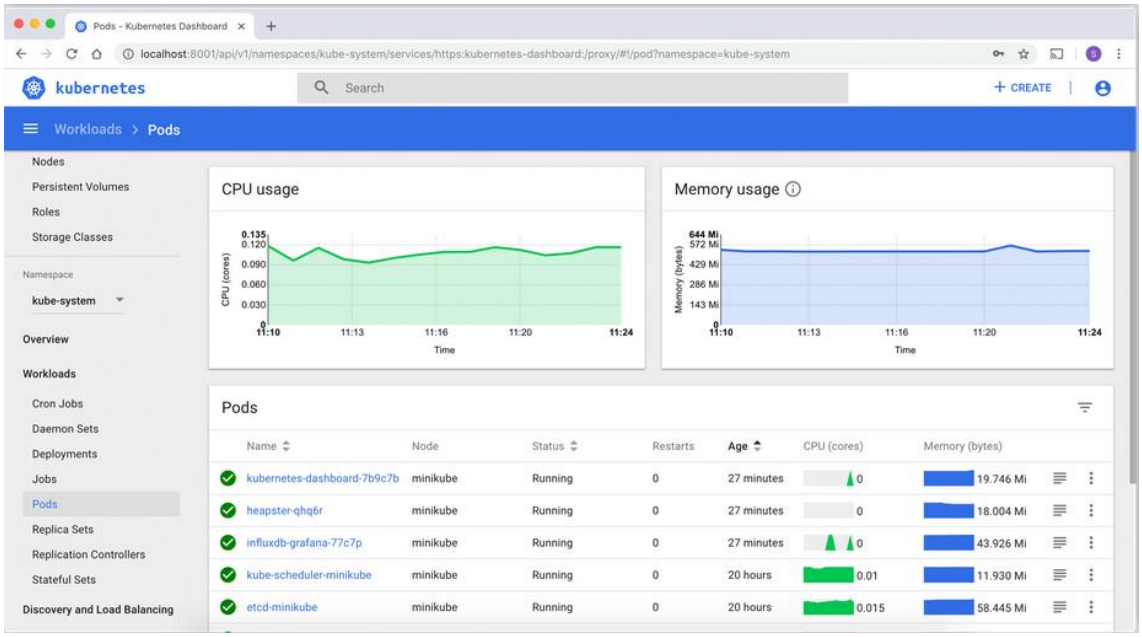
Mikäli suoritettavana on vain hallinnoiva palvelin (master node) ilman työkuormaa suorittavia palvelimia tai halutaan muuten suorittaa palveluja myös klusteria hallinnoivalla tietokoneella, klusterin Scheduler-komponentti ei turvallisuussyistä pysty määräämään pod-olioita suorittamaan tehtäviään. Tässä vaiheessa lisätyt pod-oliot jäävät pysyvästi pending-tilaan. Se on tila, jossa pod-olio rakennetaan, ja se jää odottamaan Scheduler-komponentilta käskyä, millä palvelimella pod-olion sisällä olevaa tai olevia sovelluksia ryhdytään suorittamaan. Jotta Scheduler saa käskettyä pod-olioita itse klusteria hallinnoivalle tietokoneelle, pitää hallinnoiva palvelin merkitä puhtaaksi (untainted), kuten esimerkkikoodissa 5 näytetään. [17.]

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

Esimerkkikoodi 5. Klusteria hallinnoivan palvelimen merkitseminen puhtaaksi.

Komentopaneelin asennus

CNI-lisäosien lisäksi Kubernetekseen voi hankkia useita muitakin lisäosia, omien tarpeiden mukaan. Tässä tapauksessa haluttiin käyttäjäystävällinen keino valvoa järjestelmää. Tätä varten asennettiin komentopaneeli-lisäosa (Kubernetes Dashboard), joka näkyy kuvassa 6. Tämä lisäosa on vielä beeta-vaiheessa, mutta auttaa jo huomattavasti käyttäjiä havaitsemaan erilaisia ongelmatilanteita.



Kuva 6. Kubernetes-komentopaneeli selaimessa [18].

Koska palvelintietokoneissa graafinen käyttöjärjestelmä on yleensä resurssien tuhlaamista, komentopaneelista ei suoraan asennettuna ole apua. Komentopaneeli avaa ikkunan selaimeen, ja tätä varten tulisi olla graafinen käyttöjärjestelmä. Klusterin komentopaneeliin ei myöskään pääse käsiksi muualta kuin hallinnoivalta palvelimelta. Tämä voidaan kiertää muuttamalla lisäosan asetuksia. Sen jälkeen komentopaneeliin pääsee käsiksi toisella tietokoneella avaamalla selaimen ja yhdistymällä Kubernetes-Dashboard-palvelun IP-osoitteeseen. Komentopaneelin avaaminen vaatii vielä kirjautumisen joko järjestelmän pääkäyttäjän tunnuksilla (admin-käyttäjä) tai turvallisuusvaltuudella (token). [19.]

Vaihtoehtoinen tapa valvoa Kubernetes-klusteria on kubectl-työkalun kautta. Kubectl-työkalun komentoja on esitetty esimerkkikoodissa 6. Kubectl get -kommennon avulla voidaan hakea mikä tahansa pod, service tai deployment. Kubectl logs -komennon avulla näkee halutun objektin lokimerkinnät. Kubectl describe -komennolla saadaan tarkempaa tietoa podista, servicestä tai deploymentista. Kubectl describe -komento on erittäin hyödyllinen virhetilanteita selvitettäessä.

```
kubectl get pod <pod-olion nimi>
kubectl logs pod <pod-olion nimi>
kubectl describe service <service-olion nimi>
```

Esimerkkikoodi 6. Yleisiä komentoja.

Kuvassa 7 näkyy näyttötulostus Kubectl get pods --all-namespaces -komennosta, joka näyttää kontit kaikista nimiavaruuksista.

```
ubuntu@kube:~$ kubectl get pods --all-namespaces
NAMESPACE              NAME                                        READY   STATUS    RESTARTS   AGE
default                postgres-547855c4-qwhg5                     1/1     Running   0          29d
default                pvkuva-5dcfb974d8-whzgc                     1/1     Running   0          36d
default                samba-56c655db44-8tr7f                      1/1     Running   0          17d
kube-system            coredns-6955765f44-kbppk                    1/1     Running   0          36d
kube-system            coredns-6955765f44-pwnw5                    1/1     Running   0          36d
kube-system            etcd-kube                                   1/1     Running   0          36d
kube-system            kube-apiserver-kube                         1/1     Running   0          36d
kube-system            kube-controller-manager-kube                1/1     Running   0          36d
kube-system            kube-flannel-ds-amd64-l2swt                 1/1     Running   0          36d
kube-system            kube-proxy-p4vbr                            1/1     Running   0          36d
kube-system            kube-scheduler-kube                         1/1     Running   0          36d
kubernetes-dashboard   dashboard-metrics-scraper-76585494d8-ngrz2  1/1     Running   0          36d
kubernetes-dashboard   kubernetes-dashboard-5996555fd8-6g8rn       1/1     Running   0          36d
```

Kuva 7. Kubectl get pods --all-namespaces -komennon tulostus.

Kuberneteksen nollaaminen

Kubereset.sh skriptiä käytettiin järjestelmän palauttamiseen aikaisempaan tilaan tai kokonaan nollaamaan muutokset. Uusien järjestelmien asentamisessa tulee olla huolellinen, jotta lopulliseen asennukseen ei jäisi virheitä tai muuta ylimääräistä. Jos huomattiin virheitä, nollattiin järjestelmä. Näissä tilanteissa hyödynnettiin jo luotuja asennusskriptejä, joiden avulla päästiin takaisin nopeasti ja helposti tilaan, jossa järjestelmässä ei ollut virheitä. Järjestelmän nollaamisskriptin avulla voitiin myös asennusskriptien toiminta tarkastaa.

5.3 Yrityskohtaisten palveluiden asentaminen

Jotta Anygraafin omia palveluita saatiin suoritetuksi Kubernetes-klusterissa, piti asentaa lisäosa CIFS- ja SMB-tiedostonjakoa varten. Lisäosan oikeuksia Linux-käyttöjärjestelmässä piti myös muuttaa. Lisäosan asennus ja oikeuksien muuttaminen on näytetty esimerkkikoodissa 7.

```
##install additional plugin for mounting CIFS/SMB fileshares
sudo apt-get install -y jq cifs-utils util-linux coreutils
VOLUME_PLUGIN_DIR="/usr/libexec/kubernetes/kubelet-plugins/volume/exec"
sudo mkdir -p "$VOLUME_PLUGIN_DIR/fstab~cifs"
cd "$VOLUME_PLUGIN_DIR/fstab~cifs"
sudo curl -L -O https://raw.githubusercontent.com/fstab/cifs/master/cifs
sudo chmod 755 cifs
VOLUME_PLUGIN_DIR="/usr/libexec/kubernetes/kubelet-plugins/volume/exec"
sudo $VOLUME_PLUGIN_DIR/fstab~cifs/cifs init
```

Esimerkkikoodi 7. CIFS- ja SMB -lisäosan asennus.

Koska Anygraafilla on Docker ollut käytössä jo pitkään, palvelujen siirtämiseen Kubernetes-klusteriin käytettiin valmiita Docker Imageja. Klusteriin siirrettiin Anygraafin ylläpitämä asiakkaan verkkosivu, postgreSQL-tietokantapalvelin ja hajautettu levyjärjestemä SMB (Server Message Block).

Yksityisestä Docker-kirjastosta valmiiden Docker Imagejen vetämiseen tulee luoda Kubernetekselle Secret nimeltä regcred, jota käytetään tunnistautumiseen yksityisestä Docker-kirjastosta Docker Imageja vedettäessä. Kubernetes Secrets -ominaisuuden avulla käyttäjä voi tallentaa turvallisesti arkaluontoisia tietoja, kuten salasanoja. Yhden Kubernetes Secretin voi luoda esimerkkikoodilla 8. [20.]

```
kubectl create secret docker-registry regcred \
--docker-server=<Docker kirjasto> \
--docker-username=<käyttäjätunnus> \
--docker-password=<salasana> \
--docker-email=<sähköposti>
```

Esimerkkikoodi 8. Secret Docker-kirjastoa varten.

Jos sovellus vaatii arkaluontoista dataa toimiakseen, pitää sille luoda oma Secret. Pod-olio tarvitsee myös ConfigMap-tiedoston toimiakseen. Nämä voidaan luoda joko Kuberneteksen Kustomize-työkalun avulla tai luomalla suoraan tiedostoista käyttäen

kubectl create configmap- ja kubectl create secrets -komentoja, kuten esimerkkikoodissa 9 näytetään (Kustomize-työkalun komennot kommenttimerkkien sisällä). [21; 22.]

```
### You can use either the kustomization.yaml:
#kubectl apply -k .
### or the folowing commands:
kubectl create configmap test123 --from-file=test123/conf/
kubectl create secret generic certs --from-file=test123/certs/
```

Esimerkkikoodi 9. Pod-olion konfiguraatiotiedostojen luonti.

Tämän jälkeen pitää vielä käynnistää pod- ja service-oliot kubectl apply -komennolla, kuten esimerkkikoodissa 10 näytetään. Lopputuloksena on pod-olio, jonka sisällä on test123-palvelu.

```
kubectl apply -f test123/fileshare-agdata-secret.yaml
kubectl apply -f test123/test123.yaml
```

Esimerkkikoodi 10. Kubectl apply -komento.

Tässä tapauksessa kyseessä on verkkosivu, johon saa yhteyden yhdistämällä sen IP-osoitteeseen, joka näkyy klusterin ulkopuolelle. IP-osoitteen voi tarkistaa kubectl get service -komennolla.

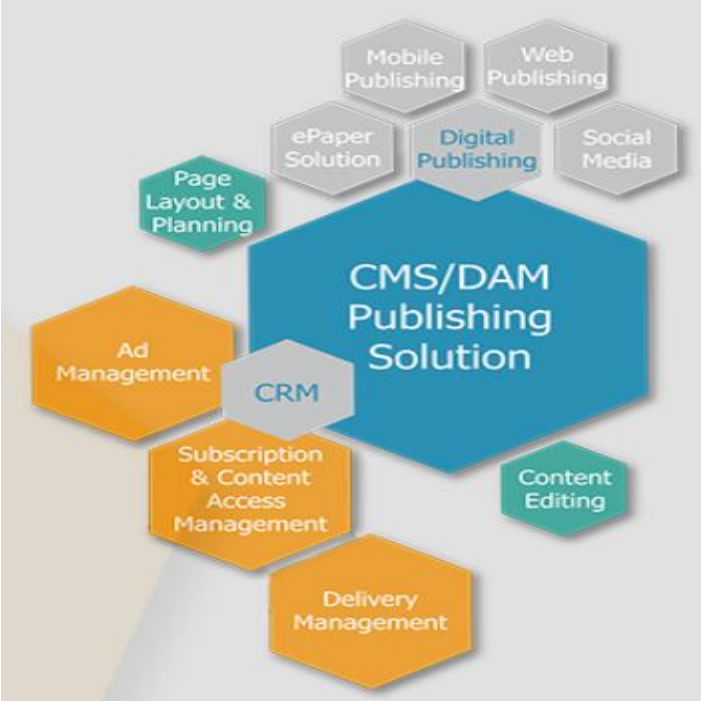
6 Integrointi yrityksen mediasisällön suunnittelu- ja hallintajärjestelmään

Kun luodut Kubernetes-järjestelmän asennusskriptit oli testattu ja todettu toimiviksi, ne integroitiin Anygraafin kehittämään Neo-järjestelmään. Luvuissa 6.1 ja 6.2 kerrotaan ensin yleisesti Neo-järjestelmän toiminnallisuudesta ja arkkitehtuurista, ja luvussa 6.3 syvennytään kuvaamaan integroinnin toteutus.

6.1 Neo-järjestelmän toiminnallinen yleiskuvaus

Neo on mediasisällön suunnittelu- ja hallintajärjestelmä. Se on työkalu toimituksellisen työn suunnitteluun ja toteutukseen, ja se on selkeästi laajempi ja monipuolisempi kuin pelkät toimitusjärjestelmät yleisesti.

Neossa on kehittyneet kommunikointimahdollisuudet, materiaalin tehokkaat ohjaamiskeinot eri julkaisukanaviin, erinomaiset työkalut sekä sisäiseen että ulkoiseen tiedonhakuun, luokittelumahdollisuudet ja paikkatiedot hyödyntämismahdollisuuksineen. Neon tärkeimmät osa-alueet on esitetty kuvassa 8.



Kuva 8. Anygraafin kehittämän Neo-järjestelmän eri osa-alueet.

Neo-järjestelmä tarjoaa mahdollisuuden mediasisältöjen tuottamiseen, muokkaamiseen ja jakeluun kokonaisvaltaisesti.

Mediasisältöjen suunnittelua ja tuottamista Neo-järjestelmä tukee mm. Teema – Aihe – Artikkeli -hierarkialla ja hyvillä muutoksenhallinta- ja ennakoivan taiton tukiominaisuuksilla. Neossa on myös monipuolinen ja automaattinen raportointiominaisuus.

Neon avulla kaikki sisältöelementit (artikkelit, kuvat ja videot) voidaan tuottaa samalla monikanavaeditorilla kaikkiin kanaviin, eikä erillistä sähköistä julkaisujärjestelmää tarvita. Materiaalista ei tarvitse luoda rinnakkaiskopioita, koska muutokset tulevat kaikkiin kanaviin, ja samat sisällöntuottajat pystyvät hallitsemaan kaikkia julkaisukanavia. Näin ollen online- ja mobiilijulkaiseminen voidaan tehdä rinnan

painettavan sisällön kanssa. Myös uusia julkaisukanavia on mahdollista lisätä tarvittaessa.

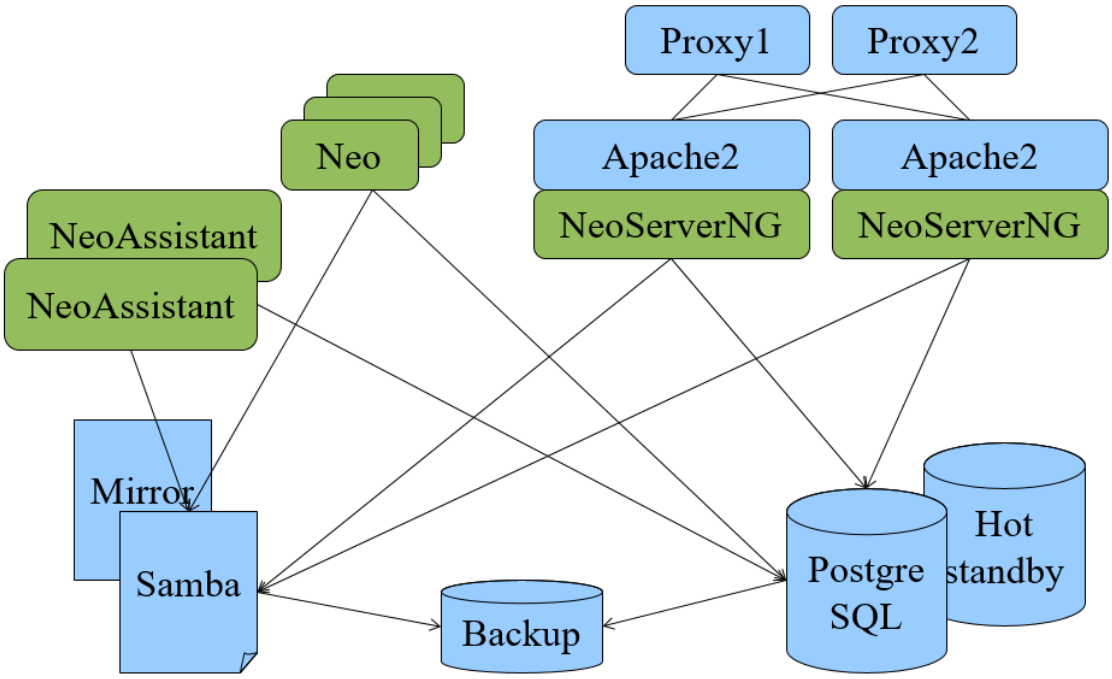
Neossa on myös henkilöresurssien hallintaa tukevia toiminnallisuuksia: budjetointi, resurssivaraukset ja -allokoinnit ja työvuorot.

Neo on suunniteltu joustavaksi ja skaalautuvaksi, jotta se pystyisi mahdollisimman hyvin vastaamaan erilaisiin julkaisutarpeisiin ja -ympäristöihin. Se soveltuu yhtä lailla pienempien paikallislehtien, isompien sanomalehtien kuin myös aikakauslehtien järjestelmäksi.

6.2 Neo-järjestelmän arkkitehtuuri

Kuva 9 kuvaa Neo-järjestelmän teknisen arkkitehtuurin, joka koostuu neljästä osasta:

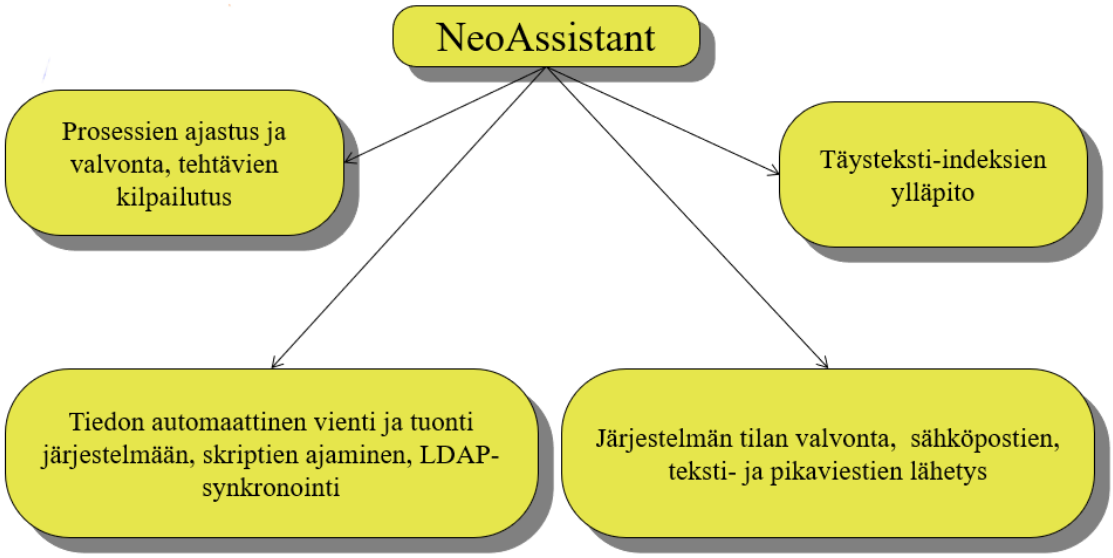
- Neo-moduulista
- NeoAssistant-moduulista
- NeoServerNG-moduulista
- infrapalveluista: tietokanta-, tiedosto- ja internetpalvelusta.



Kuva 9. Anygraafin kehittämän Neo-järjestelmän arkkitehtuuri.

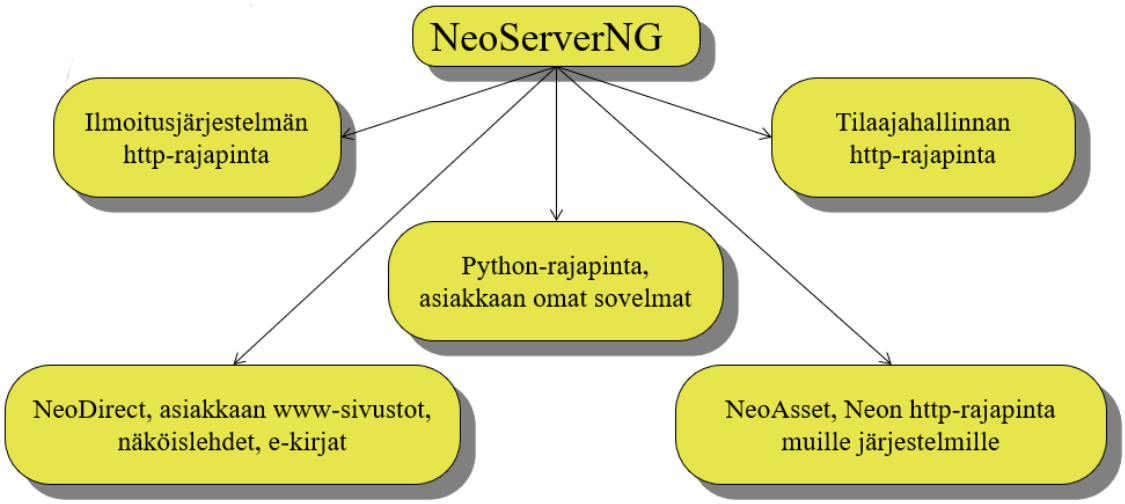
Neo on yksittäisen käyttäjän työpöytäohjelmisto. Sen toiminnallisuus kuvattiin luvussa 6.1.

NeoAssistant huolehtii kaikkien tarvittavien taustapalveluiden suorittamisesta. NeoAssistantin avulla voidaan halutut palveluprosessit ajastaa. Mikäli palvelu hajautetaan useampaan koneeseen, ne alkavat automaattisesti kilpailla samoista tehtävistä. Jos palvelu keskeytyy, käyttöjärjestelmä käynnistää sen uudelleen. Näiden ominaisuuksien avulla saavutetaan tilanne, jossa järjestelmän toimintakyky ei ole riippuvainen yhdestäkään yksittäisestä koneesta tai prosessista. NeoAssistantin toiminnallisuudet on esitetty kuvassa 10.



Kuva 10. NeoAssistantin toiminnallisuus.

NeoServerNG on Apachen laajennus, joka tuottaa Neon http-pohjaiset palvelut: selainkäytöt, näköislehdet, e-kirjat ja verkkosivustot. Selainkäytöllä tarkoitetaan tässä yhteydessä sitä, että Neon työpöytäohjelmisto korvataan selainliitynnällä järjestelmään. NeoServerNG:n toteutus on täysin REST-pohjainen, eli palvelimia voidaan suorittaa useampia rinnakkain ja saman istunnon peräkkäiset kutsut voidaan palvella eri palvelimilta. Tämän avulla voidaan toteuttaa sekä kuormanjako että vikasietoisuus. NeoServerNGn palvelut on esitetty kuvassa 11.



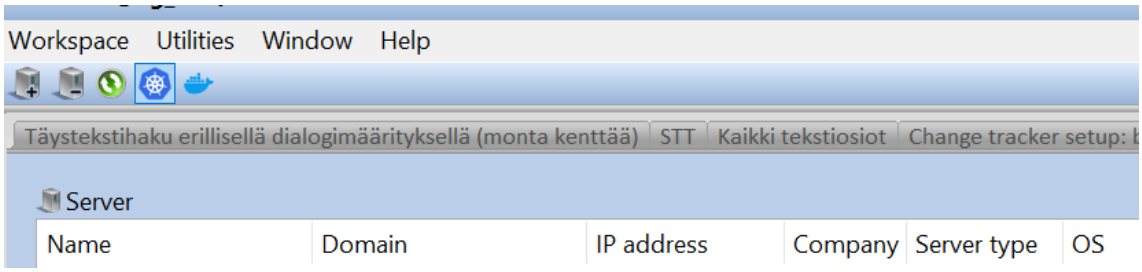
Kuva 11. NeoServerNG:n palvelut.

Neljäs Neo-järjestelmän osa, eli internetpalvelut, toteutuvat NeoServerNG + Apache -yhdistelmällä. Tiedostopalvelu on useimmiten joko Windows-tiedostopalvelin tai Linuxin päällä suoritettava Samba, joka toteuttaa saman protokollan. Tietokanta voi olla Oracle, SQL-Server tai nykyään useimmiten PostgreSQL.

6.3 Kubernetes-asennusohjelmiston integrointi Neo-järjestelmään

Asennusohjelmistoa (luodut skriptit) ei haluttu pitää erillisenä, vaan se integroitiin Neo-moduulissa olevaan infrastruktuurin hallintaosioon. Näin päästään siihen, että erillisiä ohjelmistomoduuleita on mahdollisimman vähän. On myös erittäin tärkeää, että Kubernetes-järjestelmän asennus on automatisoitu ja siten standardoitu, jotta voidaan olla täysin varmoja, että kaikki tarvittavat asennuksen vaiheet suoritetaan aina täsmälleen oikein ja oikeassa järjestyksessä. Infranstruktuurin hallintaosio (ja siten myös toteutettu Kuberneteksen asennusohjelmisto) tulee näkyviin ja käytettäväksi ainoastaan Neo-järjestelmän pääkäyttäjälle.

Kubernetes-asennuksen integroinnin ohella Neo-järjestelmään luotiin myös samalla Docker Swarmille automaattinen asennus. Näin Neo-järjestelmän avulla valitun konttien hallintajärjestelmän voi asentaa etänä nappia painamalla. Asennuspainikkeet osana Neo-käyttöliittymää näkyvät kuvassa 12.



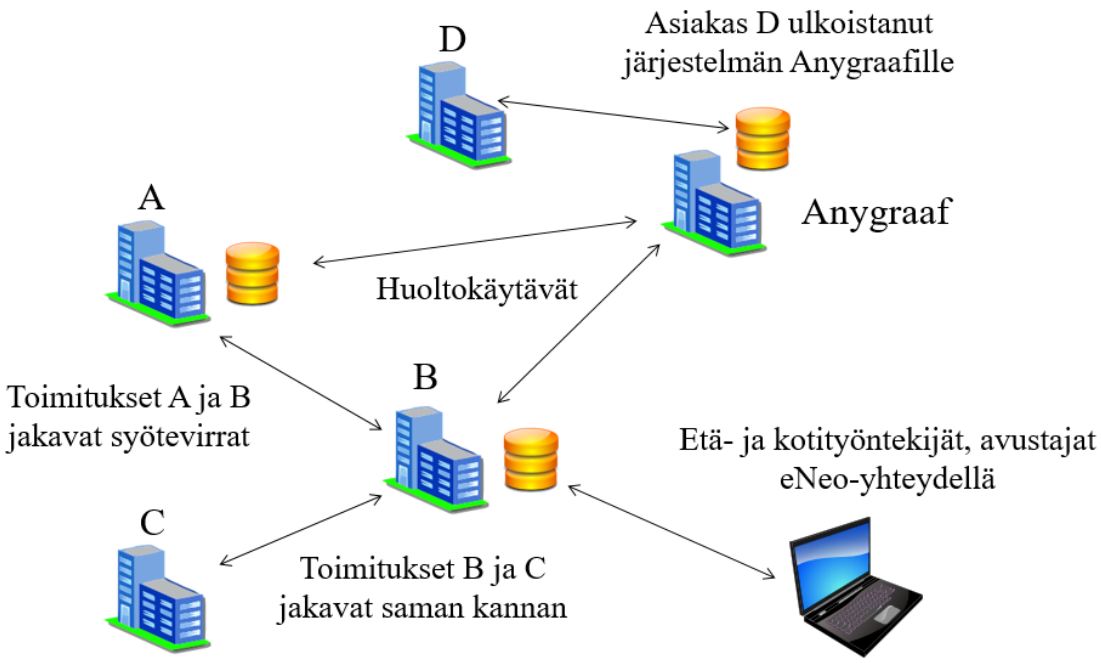
Kuva 12. Neo-käyttöliittymä ja sen Kubernetes- (valittu) ja Docker Swarm- (paneelin viimeinen painike) asennuspainikkeet.

Neo-järjestelmällä klusterin asentaminen suoritetaan SSH-yhteyden kautta. Jotta Neo-järjestelmä pääsee käsiksi tietokoneeseen, jolle klusteri halutaan asentaa, SSH-yhteyden lomassa luodaan sessio, jonka muuttujina ovat asennettavan tietokoneen IP-osoite, käyttäjätunnus sekä salasana. Sessioon avataan komentokanava. Komentokanava tulostaa komentojen tulosteen sekä virheet, jos sellaisia sattuu. Asennusskriptit on annettu tavallisina teksti-stringeinä Neo-järjestelmälle. Jotta asennuksen seuraaminen helpottuu ja mahdollisiin vikatilanteisiin on helpompi reagoida, Neo-järjestelmä lähettää asennettavalle tietokoneelle aina yhden komennon kerrallaan suoritettavaksi. Esimerkkikoodissa 11 luodaan yhteys asennettavaan koneeseen ja aloitetaan Kubernetes-komentojen antaminen.

```
static void installKubernetes()
{
	CAgSShSession session;
	bool installedKey;
	std::string error;
	if (CAgSsh::InitSession(session, "<ip-osoite>", "<käyttäjätunnus>",
"<salasana>", CAgSsh::GetKeyPath(), false, installedKey, error))
	{
		CAgSShChannel channel2(session);
		if (channel2.OpenChannel(error) && CAgSsh::Sudo(channel2,
"<salasana>", error))
		{
			std::string out, err;
			if (CAgSsh::DoCmd(channel2, "sudo swapoff -a", out, err) ==
false) // 10s
				error = "sudo swapoff -a failed";
```

Esimerkkikoodi 11. Leike Neo-intregaatiosta: Kuberneteksen asennuksen alku.

Kuvassa 13 näkyvät keltaiset tietokantasymbolit kuvaavat kontitettuja tietokanta-, tiedosto- ja internetpalvelimia. Nämä ovat siis potentiaalisia sovelluskohteita Kubernetes-klustereille, jotka voidaan nyt kätevästi asentaa tehdyllä ohjelmistolla.



Kuva 13. Konfiguraatio- ja tiedonjakomahdollisuudet Anygraafin ratkaisussa.

7 Havainnot ja jatkosuunnitelmat

Tätä insinöörityötä tehtäessä Anygraafin näkemys vahvistui siitä, että konttiteknologian avulla pystytään merkittävästi parantamaan IT-infrastruktuurin käytön optimointia. Konttien ansiosta sovellukset tarvitsevat vähemmän resursseja, ja konttien suorituskyky on virtuaalipalvelimia parempi, koska virtuaalipalvelin tarvitsee täysiverisen käyttöjärjestelmän, mikä taas kuluttaa resursseja. Lisäksi konttien viemä levytila on vain joitakin prosentteja virtuaalikoneen levytilasta. Konttiteknologian avulla pystytään automatisoimaan ja nopeuttamaan sovellusten käyttöönottoa, päivitystä ja skaalaamista sekä säästämään kustannuksia, koska IT-infrastruktuurin optimoinnin lisäksi palvelinten ylläpitoon tarvitaan vähemmän ihmistyötä.

Kubernetes osoittautui myös Anygraafin aiemmin kokeilemaa Docker Swarmia paremmaksi ratkaisuksi konttien orkestrointiin. Vaikka Docker Swarm on yksinkertaisempi ja sen kanssa pääsee siten nopeasti alkuun, on Kubernetes monipuolisempi ja luonteva valinta palveluntarjoajayritysten tuotannolliseen käyttöön.

Lisäksi työtä tehdessä tuli selväksi, että Kubernetes-klusterin rakentaminen ei ole helppo tehtävä. Se vaatii isohkoa ajallista panostusta ja asiaan perehtymistä, ja ilmeisesti sen vuoksi moni yritys päätyykin suorittamaan Kubernetesia julkisessa pilvessä. Kaikki suurimmat pilvipalvelujen tarjoajat (Amazon AWS, Microsoft Azure ja Google Cloud Platform) tarjoavat Kubernetes-klustereita palveluna. Anygraafin palveluarkkitehtuurissa On Premises (omissa tiloissa itse hallinnoiden) -ympäristössä toimiva Kubernetes-klusteri kuitenkin tarvitaan. Toimivan Kubernetes-asennusohjelmiston aikaansaamiseksi käytiin lukuisia keskenään ristiriitaisia verkkodokumentteja ja asennusohjeita läpi, ja vasta useitten yritysten ja erehdysten kautta saavutettiin riittävä osaaminen toimivan asennusohjelmiston tekemiseksi.

Nyt rakennettu Kubernetes-asennusohjelmisto integroituna Anygraafin Neo-järjestelmään tarjoaa erinomaisen pohjan jatkokehittämiselle ja ratkaisun hyödyntämiselle. Seuraavaksi Anygraafissa aiotaan tehdä suunnitelmat siitä, millä tavoin nykyisestä mallista pystytään siirtymään kontitettuun palvelinympäristöön. On syytä myös tehdä lisää kuormitustestejä suurella kuormalla.

8 Yhteenveto

Insinöörityössä perehdyttiin konttiteknologiaan ja luotiin järjestelmä, jossa konttiteknologian edut voidaan tutkia ja asennus automatisoida. Käytetty teknologia oli Kubernetes, ja luotu Kubernetes-asennusohjelmisto integroitiin toimeksiantajan, Anygraaf Oy:n Neo-ohjelmistoon.

Kubernetes on monimutkainen teknologia, josta on tuotettu valtavasti eritasoista materiaalia. Sen käyttöön liittyy paljon asioita, jotka täytyy ymmärtää, jotta sitä voi hyödyntää. Sen avulla on kuitenkin mahdollista muuttaa perinteisiä käsin ylläpidettäviä IT-ympäristöjä ja saavuttaa merkittäviä etuja.

Anygraafissa oltiin tyytyväisiä tehtyyn työhön ja saavutettuihin tuloksiin. Suunnitelmat tuotteistuksen jatkamiseksi ovat selvillä, ja Anygraafissa harkitaan myöhemmin myös sitä, että toteutettu asennusohjelmisto liitettäisiin osaksi asiakkaan Neo-versiota. Tämä mahdollistaisi sen, että asiakkaat pääsisivät itse rakentamaan omia palvelinympäristöjään. Isoilla julkaisutaloilla tuntuu usein olevan tavoitteena se, että palvelimia pääsee itse hallitsemaan.

Lähteet

1 Anygraaf Oyn internetsivusto. Verkkoaineisto. Anygraag Oy. <https://neo-digital-publishing.anygraaf.net/neo?app=NeoDirect&com=6/96/a8ada834e9&lang> Luettu 1.4.2020.

2 What is virtualization? Verkkoaineisto. Redhat. <https://www.redhat.com/en/topics/virtualization/what-is-virtualization> Luettu 3.4.2020.

3 Virtualization. Verkkoaineisto. Vmware. <https://www.vmware.com/solutions/virtualization.html> Luettu 3.4.2020.

4 Containers. Verkkoaineisto. Google. <https://cloud.google.com/containers> Luettu 3.4.2020.

5 Menage, Paul. CGROUPS. Verkkoaineisto. <https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt> Luettu 3.4.2020.

6 What is a hypervisor? Verkkoaineisto. Redhat. <https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor> Luettu 5.4.2020.

7 Wallenius, Niklas. Mikä on Docker? Verkkoaineisto. Wallenius Consulting. <https://niklaswallenius.fi/digitalisaatio/mika-on-docker/> Luettu 5.4.2020.

8 What is Kubernetes?. Verkkoaineisto. Kubernetes. <https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/> Luettu 1.3.2020.

9 Wallenius, Niklas. Kubernetes. Verkkoaineisto. Wallenius Consulting. <https://niklaswallenius.fi/teknologiat/kubernetes/> Luettu 1.4.2020.

10 Kubernetes Components. Verkkoaineisto. Kubernetes. <https://kubernetes.io/docs/concepts/overview/components/> Luettu 4.3.2020.

11 Lane, Katie. 2019. Kubernetes Architecture and Components. Verkkoaineisto. Sumo Logic. <https://www.sumologic.com/blog/kubernetes-vs-docker/> Luettu 1.4.2020.

12 Vohra, Deepak. 2020. Using Basic Kubernetes Objects. Verkkoaineisto. Techwell. <https://www.techwell.com/techwell-insights/2020/03/using-basic-kubernetes-objects> Luettu 3.4.2020.

13 Deployment. Verkkoaineisto. Google. <https://cloud.google.com/kubernetes-engine/docs/concepts/deployment> Luettu 3.4.2020.

14 Installing kubeadm. Verkkoaineisto. Kubernetes. <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/> Luettu 1.3.2020.

15 Why disable swap on kubernetes. Verkkoaineisto. Serverfault. <https://serverfault.com/questions/881517/why-disable-swap-on-kubernetes> Luettu 8.4.2020.

16 Container runtimes. Verkkoaineisto. Kubernetes. <https://kubernetes.io/docs/setup/production-environment/container-runtimes/> Luettu 4.3.2020.

17 Creating a single control-plane cluster with kubeadm. Verkkoaineisto. Kubernetes. <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/> Luettu 1.3.2020.

18 Web UI (Dashboard). Verkkoaineisto. Kubernetes. <https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

19 How To Access Kubernetes Dashboard Externally. Verkkoaineisto. The Geek Diary. <https://www.thegeekdiary.com/how-to-access-kubernetes-dashboard-externally/> Luettu 8.3.2020.

20 Secrets. Verkkoaineisto. Kubernetes. <https://kubernetes.io/docs/concepts/configuration/secret/> Luettu 20.3.2020.

21 Understanding ConfigMaps and Pods. Verkkoaineisto. Kubernetes. <https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/#understanding-configmaps-and-pods> Luettu 20.3.2020.

22 Declaritive Management of Kubernetes Objects Using Kustomize. Verkkoaineisto. Kubernetes. <https://kubernetes.io/docs/tasks/manage-kubernetes-objects/kustomization/> Luettu 20.3.2020.