Haaga-Helia
ammattikorkeakoulu Oy

Kubernetes ja AWS EKS

Heikki Ma

Opinnäytetyö
Tietojenkäsittely
2018

Tiivistelmä

Haaga-Helia
ammattikorkeakoulu Oy

Tekijä(t)

Heikki Ma

Koulutusohjelma

Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi	**Sivu- ja liitesivumäärä**
Kubernetes ja AWS EKS	28 + 1

Opinnäytetyö oli rajattu tutkimaan Kubernetes konttiorkestrointityökalua ja Amazon Web Servicen (AWS) uutta Kubernetes -palvelua, Amazon Elastic Container Service for Kubernetes (AWS EKS). Tutkimus koostuu teoria- ja käytännönosuuksista.

Kubernetes on Googlen aloittama projekti. Google julkaisi projektin maailmalle avoimena lähdekoodina. Kubernetes on konttiorkestrointityökalu, jonka tehtävänä on hallita useita kontteja keskitetysti. Yksi kontti koostuu yleensä yhdestä mikropalvelusta. Mikropalvelut ovat jaettuja kokonaisuuksia sovelluksesta.

Kubernetes -klusteri koostuu yhdestä tai useammasta mestarikoneesta, jotka toimivat klusterin ohjaamona. Ohjaamosta järjestelmänvalvoja kykenee hallitsemaan koko klusterin toimintaa ohjaamon rajapinnan kautta. Ohjaamon lisäksi klusterissa tulee olla vähintään kaksi (2) kappaletta kätyrikoneita, joihin kontit rakennetaan. Ohjaamo valvoo automaattisesti klusterin tilaa ja korjaa ongelmatilanteissa itse itsensä.

AWS:n julkaisema Kubernetes -palvelu on nimeltään EKS. EKS -palvelu tarjoaa käyttäjälleen korkea käytettävän ohjaamon. Käyttäjä joutuu itse luomaan kätyrit ja yhdistämään nämä palvelun klusteriin. EKS käyttää AWS:n omia resursseja hyödykseen tunnistautumisessa ja verkkoyhteyksissä.

Käytännön osuudessa luodaan harjoitusympäristö käyttäen EKS -palvelua. Kaikki materiaalit, joita käytettiin harjoitusympäristön luonnissa ovat GitHub säilytyspaikassa.

Kubernetes soveltuu käytettäväksi useamman projektin infrastruktuurina. Kubernetes ei sovellu tilallisten palvelujen kuten tietokantojen ajoon. Kubernetes on tehokas työkalu, jonka suurena etuna on järjestelmän siirtäminen eri ympäristöjen välillä. Samanlainen Kubernetes -ympäristö voidaan luoda omassa konesalissa, pilvessä tai hybridipilvessä. Kubernetes ei sido järjestelmää ollenkaan ympäristöön.

EKS on ominaisuuksiltaan huono vaihtoehto kilpailijoihinsa nähden. EKS -palvelua en suosittele käytettäväksi, ellei ole tarkoitus siirtää olemassa olevaa Kubernetes järjestelmää AWS ympäristöön.

Asiasanat

Konttiorkestrointityökalu, Kubernetes, Infrastruktuuri, Mikropalvelu, Amazon Web Services, Pilvipalvelu

Sisällys

1 Johdanto

Konttisovelluksien suosio on kasvanut ohjelmistoalalla huomattavasti. Sovellukset pilkotaan pienemmiksi mikropalveluiksi. Mikropalveluista rakennetaan kontti, joita ajetaan eristyksessä toisistaan. Konttien hallintaan on kehitetty erilaisia konttiorkestrointityökaluja, joista tällä hetkellä suosituin on Kubernetes. Pilvipalveluntarjoajat kuten Amazon Web Services, Google Cloud ja Microsoft Azure ovat kehittäneet oman Kubernetes palvelunsa työkalun suuren suosion vuoksi.

Opinnäytetyön aiheena on Kubernetes ja AWS ympäristössä uusi EKS -palvelu. Työn tavoitteena on luoda hyvä yleisymmärrys Kubernetes työkalusta ja tämän toiminnasta. Opinnäytetyön toimeksiantajana toimii Webscale Oy. Webscale Oy on konsultointiyritys, joka on erikoistunut ohjelmistotuotantoon ja AWS konsultointiin. Toimeksiantajaa kiinnostaa Kubernetes -työkalun lisäksi AWS -pilvipalvelun uusi Kubernetes -palvelu EKS. Toimeksiantaja haluaisi tietää mahdollisista hyödyistä ja haitoista, joita EKS -palvelu toisi, mikäli tämän palvelun ottaisi käyttöön projekteissa. Opinnäytetyön aiheeksi valikoitui Kubernetes, koska haluan omakohtaisesti tutkia tätä konttiorkestrointityökalua ja oppia uuden tavan rakentaa infrastruktuuria konttisovelluksille.

Opinnäytetyö jakautuu teoria- ja käytännön osuuksiin. Teoriaosuudessa käsitellään Kubernetes konttiorkestrointityökalun yleisestä taustasta ja työkalun tärkeimpien komponenttien tehtäviä. Tämän lisäksi tutustutaan EKS -palveluun yleisesti. Käytännönosuudessa tavoitteena on luoda toimiva harjoitusympäristö, jossa hyödynnetään EKS -palvelua. Harjoitusympäristön pääaiheena on ympäristön infrastruktuurin rakentaminen, eikä sinänsä ohjelmistotuotanto. Harjoitusympäristössä käytetään olemassa olevaa sovellusta. Käytännön osuuden tavoitteena on saada kokemusta EKS -palvelusta, jotta voidaan kertoa asiantuntevasti palvelun hyvistä ja huonoista ominaisuuksista. Harjoitusympäristön kaikki materiaalit julkaistaan GitHub säilytyspaikassa. Opinnäytetyön lopussa pohditaan työstä saatuja tuloksia.

Opinnäytetyössä hyödynnetään erilaisia lähteitä. Lähteinä käytetään olemassa olevia tutkimuksia, kirjoja, virallisia dokumentaatioita ja artikkeleita. Opinnäytetyössä pyritään käyttämään monipuolisesti mahdollisimman tuoreita lähteitä.

1.1 Lyhenteet, termit ja käännökset

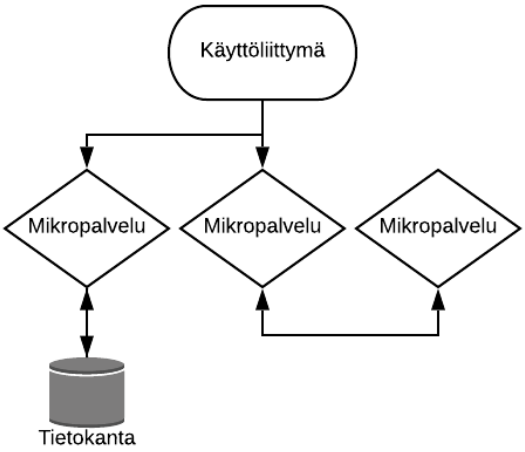
AKS	Azure Kubernetes Service
ASG	Auto Scaling Group, AWS ympäristössä automaatti skaalaaja
AWS ECS	Amazon Elastic Container Service
AWS	Amazon Web Services. Amazonin pilvipalvelu
CNCF	Cloud Native Computing -säätiö
Docker-levykuva	Docker image. Levykuva, josta kontti rakentuu
EC2	Pilvipalvelin AWS ympäristössä
EKS	Elastic Container Service for Kubernetes
GKE	Google Kubernetes Engine
HA -klusteri	Korkea käytettävyyden klusteri
Isäntäkone	Host. Fyysinen tai virtuaalinen palvelin
K8s	Kubernetes
Kapseli	Pod
Klusteri	Cluster. Joukko palvelimia, jotka muodostavat yhdessä järjestelmän
Konttiorkestrointityökalu	Container orchestration tool
Korkean käytettävyys	Highly Available. HA
kuormantasaaja	Loadbalancer
Solmu	Node. Fyysinen- tai virtuaalinen kone, joka kykenee lähettämään ja vastaanottamaan tietoa.
Varanto	Pool. Abstraktinen käsite resurssiryhmästä
VPC	Virtual private cloud. Oma pilviverkko

2 Kubernetes

Tässä luvussa käsittelemme Kubernetesta, tämän arkkitehtuuria ja hyötyjä. Kubernetes tunnetaan myös lyhenteellä K8s. Koko tutkimuksessa käytetään kyseistä lyhennettä, kun viitataan Kubernetekseen.

2.1 Kuberneteksen tausta

Docker julkaistiin avoimena lähdekoodina vuonna 2013. Siitä lähtien konttiteknologia on ollut suuressa suosiossa sovelluskehityksessä. Docker tarjoaa työkalut sovelluksen ja tämän riippuvuuksien paketointiin Docker-levykuvaksi. Hakkaraisen (2018) mukaan sovellus tulee kuitenkin pilkkoa pienempiin mikropalveluihin ennen paketointia. Mikropalvelut ovat jaettuja kokonaisuuksia sovelluksesta. Jokaisella mikropalvelulla on oma tehtävänsä ja nämä kommunikoivat keskenään tarvittaessa (kuva 1). Jokaisen mikropalvelun tulee olla itsenäisesti hallittavissa ja muutettavissa. Paketoidusta mikropalvelusta voidaan käynnistää kontti missä tahansa käyttöjärjestelmässä, kunhan tämä tukee Docker teknologiaa. Kontit ovat hyvin kevyitä ja yhdessä isäntäkoneessa voidaan ajaa samanaikaisesti useita kontteja. Teknologian etuna on sovellusten yksinkertainen pakkaaminen ja monistaminen erilaisiin ympäristöihin. Tämä nopeuttaa sovelluksen julkaisemista tuotantoon, ja tuotannossa tämän skaalaamista (Docker, s.a.).



Kuva 1. Mikropalvelut (mukailen Nadareishvili, Mitra, McLarty, & Amundsen, 2016)

Google julkaisi K8s projektin avoimena lähdekoodina vuonna 2014. Google ja Linux -säätiö perustivat yhdessä Cloud Native Computing -säätiön, jolle Google lahjoitti K8s:n hallittavakseen vuonna 2015 (Lardinois, 2015). K8s on konttiorkestrointityökalu, jonka tarkoituksena on tarjota alusta ja työkalut konttien keskitettyyn hallintaan. Tämän avulla voidaan hallita useista isäntäkoneista muodostunutta klusteria, sovelluksien välisiä verkkoyhteyk-

siä ja näiden skaalaamista nopeasti ja helposti. Muita samantyylisiä alustoja ovat esimerkiksi AWS ECS ja Docker Swarm. K8s:ia käytetään useimmiten Dockerin konttien kanssa. K8s kuitenkin tukee myös muita konttijärjestelmiä, jotka täyttävät Open Container Initiative (OCI) standardit (Yegulalp, 2019; Kubernetes, 2018a).

AWS ECS on Amazonin pilvipalvelualustalla toimiva konttiorkestrointityökalu. AWS ECS hyödyntää AWS ympäristössä jo olemassa olevia resursseja, esimerkiksi pilvi-instanssit EC2 ja näiden automaatti skaalaustyökalua. AWS ECS ajaa kontteja pilvi-instanssin sisällä ja tarjoaa työkalut konttien hallintaan AWS konsolin tai AWS rajapinnan avulla. AWS ECS skaalaa kontteja tarvittaessa ylöspäin/alaspäin automaattisesti. Automaatti skaalaus noudattaa skaalaamisessa ennalta määrättyjä sääntöjä, jotka määrittelevät vähimmäis- ja enimmäismäärän konteille. Mikäli resurssit loppuvat pilvi-instanssilta voidaan instanssien määrää skaalata ylös- tai alaspäin. AWS ECS on saatavilla vain AWS ympäristössä (Sarkar & Shah. 2018).

Docker Swarm on Dockerin kehittämä orkestrointityökalu Docker konteille. Docker Swarm ryhmittää konttien isäntäkoneet yhteen varantoon ja tarjoaa käyttäjälle työkalut hallita varantoa yhtenäisesti (Ravindra, 2018; Paraiso, Challita, Al-Dhuraibi & Merle 2016).

K8s on hyvin tehokas konttiorkestrointityökalu, joka on suunniteltu hallitsemaan ja ohjaamaan useita kontteja samanaikaisesti. K8s järjestelmä toimii Mestari - kätyri (master - minion) konseptilla. Mestari – kätyri järjestelmä koostuu yleensä yhdestä mestarista ja monesta kätyristä. Mestarin tehtävänä on hallita järjestelmää ja jakaa käskyjä kätyreille. Kätyrien tehtävä on ajaa sovelluksia konteissa.

Guptan (2017) mukaan K8s suosio johtuu siitä, että sillä on yksi maailman isoimmista avoimen lähdekoodin yhteisöistä. Yhteisö on ollut aktiivisesti mukana kehittämässä K8s ohjelmaa GitHubissa. Gupta kertoo myös, että CNCF -säätiöllä on suuri rooli K8s:n suosioon. CNCF kuuluu Linux -säätiöön (Linux Foundation), jota tukevat monet suuret yritykset, kuten pilvipalveluntarjoajat Google, Microsoft ja AWS. Tämän lisäksi CNCF:llä on omia tukijoita, esimerkiksi Oracle ja SAP.

Moni pilvipalvelun tarjoaja on ottanut K8s:n tuotteekseen, esimerkiksi Microsoft Azuren AKS, Google Cloud Platformin GKE ja Amazon Web Servicesin EKS. Pilvipalveluntarjoajat voivat hyödyntää K8s avointa lähdekoodia ja luoda lisäominaisuuksia palvelulleen. Tämä parantaa palvelun käyttökokemusta ja mahdollistaa K8s palvelun liittämisen tarjoajan omaan ekosysteemiin. Tämän lisäksi palveluntarjoajat voivat hyödyntää olemassa olevia työkaluja, joita K8s yhteisö on luonut ja jakanut.

2.2 Kuberneteksen arkkitehtuuri

K8s koostuu klusterista. Klusteri muodostuu monesta koneesta (fyysisestä tai virtuaalisesta), joissa ajetaan K8s ohjelmaa. Koneita voidaan liittää jälkikäteen järjestelmään helposti, mikäli järjestelmän resurssit eivät riitä. Klusterin sisällä on yksi tai useampia mestarisolmuja ja useita kätyrisolmuja (Kubernetes 2019a). Tässä kappaleessa käymme läpi muutamaa isompaa komponenttia K8s:n klusterissa. Näiden lisäksi K8s sisältää pienempiä, mutta silti tärkeitä komponentteja.

2.2.1 Master node

Mestarisolmu toimii klusterin ohjaamona. Ohjaamossa päätetään klusterin asioista, kuten esimerkiksi ajastetuista tehtävistä ja klusterin muutoksiin liittyvistä asioista. Mestarisolmut koostuvat useista eri mestarikomponenteista. Näitä komponentteja ovat kube-apiserver, kube-scheduler, kube-controller-manager ja etcd. Komponentit voidaan jakaa usealle koneelle, mutta yksinkertaisuuden vuoksi komponentit ovat yleensä samassa koneessa (pois lukien etcd, tästä lisää seuraavassa kappaleessa) (Kubernetes 2019a; Arundel & Domingus, 2019).

Kube-apiserver on klusterin ulkopuolelle näkyvä rajapinta. Rajapintaan tehdään kutsuja ja lähetetään haluttuja muutoksia klusteriin. Kätyrisolmut käyttävät myös kyseistä rajapintaa kommunikoidessaan mestarisolmun kanssa. *Etcd* on klusterin oma tietokanta. Klusterin rajapinta, Kube-apiserver, tallentaa datan avain-arvo pareina tietokantaan. Tietokantaan tallennetaan muun muassa klusterin nykyinen tila ja haluttu tila. K8s -klusteri käyttää etcd tietokantaa totuuden lähteenä. On tärkeää luoda varmuuskopioita etcd -tietokannasta. K8s dokumentaatio suosittelee luomaan etcd -komponentille oman klusterin. Etcd -klusteri voi olla K8s -klusterin sisällä tai ulkopuolella (kuva 2). *Kube-scheduler* tarkkailee, jos uusia kapseleita (komponentti, joka pitää sisällään kontteja) luodaan ja ohjaa nämä oikeisiin kätyrisolmuihin. Kube-scheduler valitsee kapseleille sopivia kätyrisolmuja ja osaa ottaa huomioon näiden resurssi- ja muut vaatimukset.

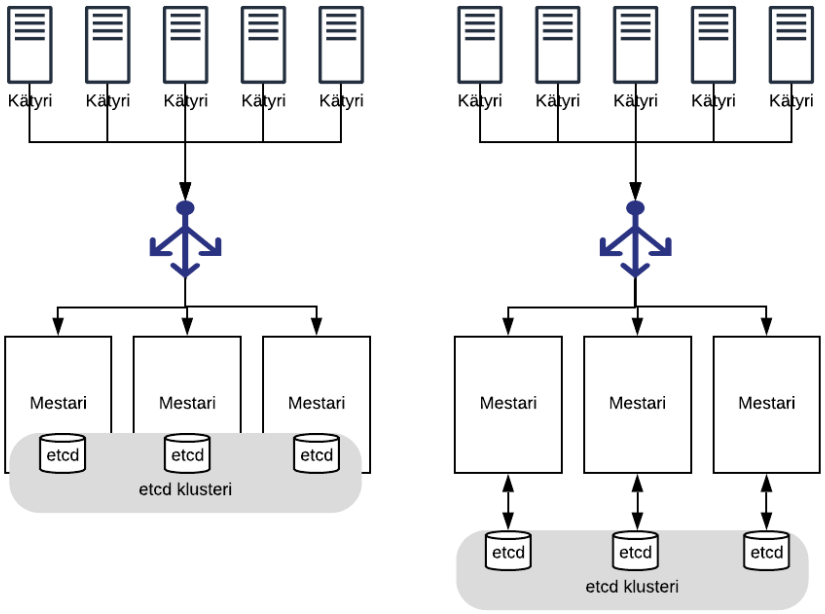
Kube-controller-manager komponentti hallinnoi ja tarkkailee klusterin tilaa. Tämä hakee nykyisen ja halutun tilan etcd -tietokannasta kube-apiserverin kautta, ja muuttaa klusteria haluttuun suuntaan, mikäli se on erilainen. Kube-controller-manager käyttää klusterin tarkkailuun erilaisia valvojia. Jokaisella valvojalla on oma tehtävänsä ja tarkkailukohteensa.

Valvojia ovat esimerkiksi:

- Node Controller
    - Tarkkailee kätyrisolmuja. Jos kätyrisolmu kuolee, pyrkii valvoja nostamaan uuden tilalle.
- Replication Controller
    - Tarkkailee kapseleiden lukumäärä solmujen sisällä. Jos kapselien määrä on eri kuin halutussa tilassa, valvoja tiputtaa tai nostaa kapseleita kätyrisolmuun.

Valvojia on näiden lisäksi monia, joko K8s virallisia tai K8s yhteisön luomia, ja jokaisella valvojalla on oma tehtävänsä (Baier 2017; Kublr, 2017; Kubernetes, 2019a).

K8s -klusterilla on yleensä yksi mestari, joka ohjaa ja käskee muita koneita klusterissa (Yegulalp, S, 2019). Mestareita voi kuitenkin olla klusterissa monta, kun rakennetaan korkean käytettävyyden (lyhenne HA) klusteria. HA -klusterissa yksi kone kerrallaan toimii mestarina. Mestari ajaa ajastettuja tehtäviä ja ohjaa muita koneita klusterissa. HA:n ideana on ylläpitää sovelluksen hallintaa mestari koneella, myös silloin kun yksi mestareista kaatuu esimerkiksi laitevian vuoksi. Kätyrisolmut ottavat mestariin yhteyttä kuormantasaajan kautta (kuva 2). Kuormantasaaja ohjaa liikenteen mestarille, jonka tämä tunnistaa olevan pystyssä (Kubernetes 2019b; Arundel & Domingus, 2019).



Kuva 2. Vasemmalla sisänen etcd -klusteri. Oikealla ulkoinen etcd -klusteri (mukailen Kubernetes 2019a).

2.2.2 Pod

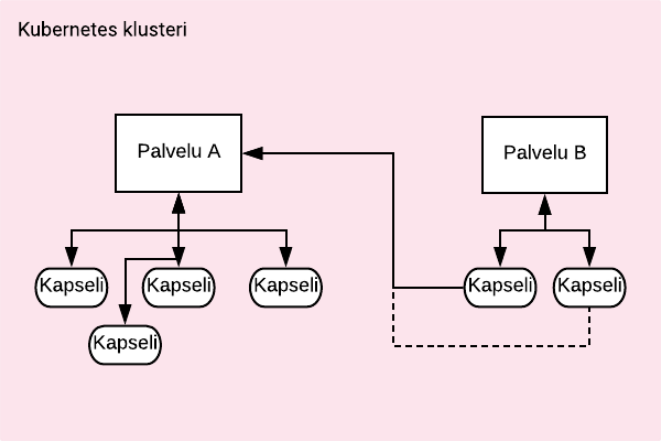
Pod eli kapseli muodostuu usein miten yhdestä kontista. On kuitenkin mahdollista luoda kapseli useammasta hyvin tiiviiksi kytketystä kontista, jotka ovat hyvin riippuvaisia toisistaan. Kapseli sijoitetaan aina kätyrisolmuun. Tämän sisällä olevat kontit jakavat saman IP osoitteen ja muistin. Jokaisella kapselilla on oma elämänkaarensa. Elämänkaaren eri vaiheita ovat:

- Tulossa oleva (Pending)
    - K8s -klusteri on hyväksynyt kapselin, mutta kapselin kontti ei ole vielä luotu.
- Käynnissä (Running)
    - Kapseli on ohjattu kätyrisolmulle ja kontti on luotu kapselin sisälle.
- Onnistunut (Succeeding)
    - Kaikki kontit kapselin sisällä on poistettu onnistuneesti.
- Epäonnistunut (Failed)
    - Kaikki kontit kapselin sisällä on poistettu, mutta yksi tai useampi kontti on palauttanut virhekoodin poistaessa.
- Tuntematon (Unknown)
    - Tuntemattomasta syystä kapselin tilaa ei tiedetä.
- Valmis (Completed)
    - Kapseli on suorittanut tehtävänsä.

Kun kapseli poistetaan tai se kuolee pois, sitä ei nosteta enää takaisin ylös. K8s järjestelmä luo uuden tilalle käyttäen mestarisolmun valvojaa, jos on tarve uudelle kapselille (Kublr, 2017; Kubernetes, 2019c).

2.2.3 Service

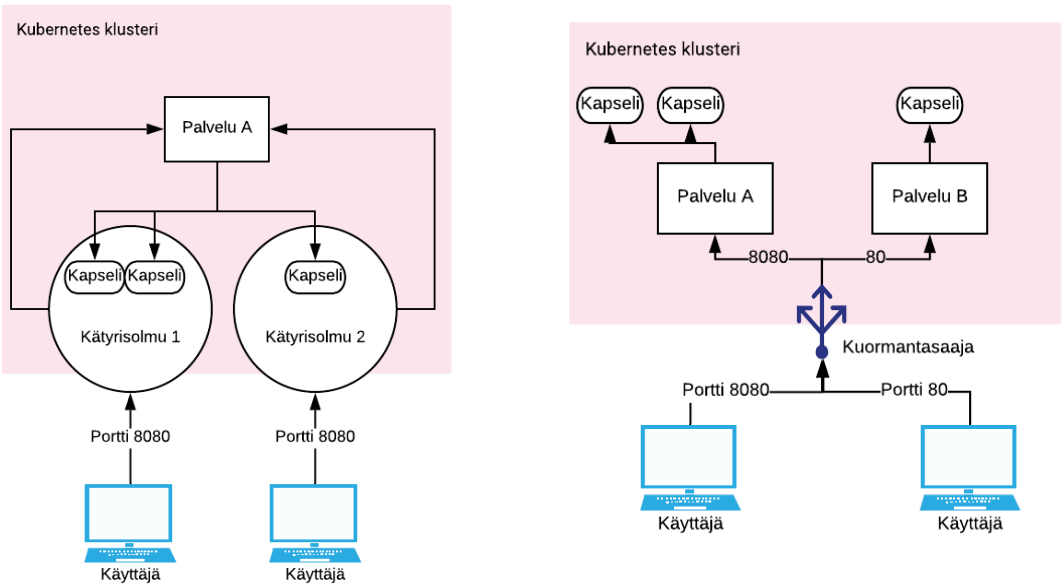
Aikaisemmin kerrottiin kapseleiden elämänkaaresta. Kapselit nousevat ylös ja kuolevat pois. Tämä luo uuden ongelman. Kapseleiden sisällä olevalle sovellukselle pitäisi ohjata liikennettä, mutta kun nämä muuttuvat dynaamisesti, ei sovelluksella ole pysyvää osoitetta. K8s -järjestelmässä on palvelu-objekti, jonka tehtävänä on ratkaista kyseinen ongelma. Palvelu-objekti on abstraktinen käsite, joka pitää sisällään tiedon yhden sovelluksen olemassa olevista kapseleista ja siitä, miten näihin saa yhteyden. Palvelu-objekteja tulee luoda jokaiselle sovellukselle oma, mikäli haluaa näihin yhteyttä. Palvelu-objektin oletus tyyppi on ClusterIP. Muita palvelu-objektin tyyppejä on NodePort ja LoadBalancer. *ClusterIP* -tyyppinen palvelu-objekti saa itselleen sisäisen IP-osoitteen. IP-osoitetta voidaan kutsua vain klusterin sisältä (kuva 3). Palvelu-objektissa on sisäänrakennettu kuormantasaaja, joka lähettää yhteyden vain saatavilla oleville kapseleille



Kuva 3. Palvelu-objekti ClusterIP

NodePort -tyyppinen palvelu-objekti näkyy K8s -klusterin ulkopuolelle. Palvelu-objekti avaa jokaisen klusterin kätyrisolmusta saman portin (kuva 4). Kätyrisolmu ohjaa portista tulevan liikenteen ClusterIP objektille (ClusterIP objekti, luodaan automaattisesti NodePort palveluobjektin kanssa).

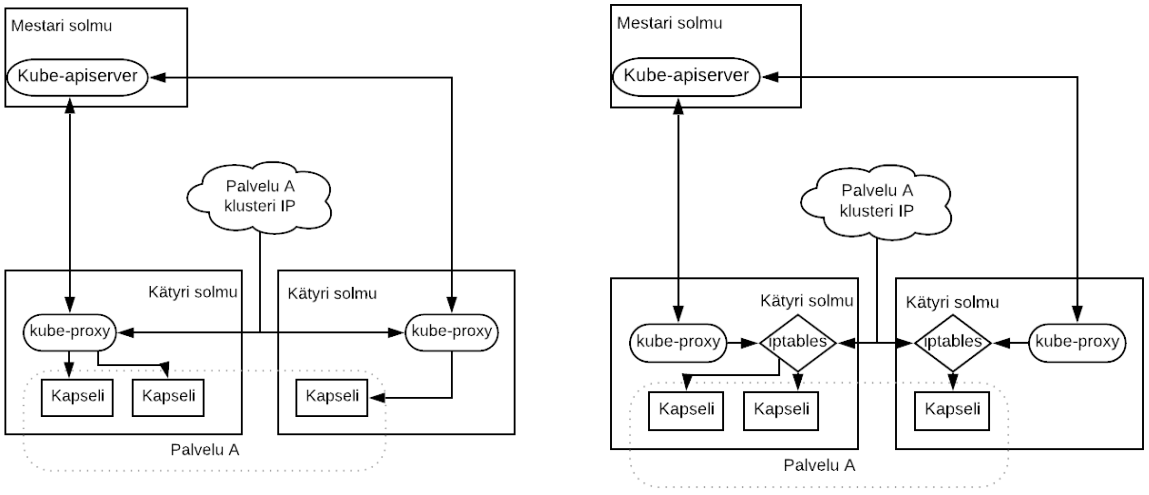
Palvelu-objekti tyyppiä *LoadBalancer* on käytössä vain pilvipalveluntarjoajilla, joilla on K8s -yhteensopiva kuormantasaaja. Kuormantasaajalle määritellään portti, jonka liikenne ohjataan eteenpäin palvelulle (kuva 4). Kuormantasaaja ohjaa kaiken tyyppiset liikenteet palvelulle, esimerkiksi HTTP, TCP ja UDP. Kuormantasaaja ei tue salauksen purkamista. Samalla kuormantasaajalla voi olla useampi portti määritelty eri palveluille. (Kubernetes, 2018c; Kubernetes, 2018d; Yegulalp, 2019; Sandeep, 2018).



Kuva 4. Vasemalla palvelu-objekti NodePort. Oikealla palvelu-objekti LoadBalancer

2.2.4 Minion node

Kätyrisolmut ovat koneita, joissa sijaitsevat sovelluksen kapselit. Kätyrisolmut koostuvat kubelet, kube-proxy ja container runtime -komponenteista. *Kubelet* -komponentin tehtävänä on valvoa solmun sisällä olevia kapseleita. Tämä varmistaa, että kapselit ovat terveitä ja täyttävät niille annetut vaatimukset. Kubelet saa kapselien vaatimukset mestarisolmulta ja raportoi mestarisolmulle, mikäli kapseli ei täytä vaatimuksia. *Kube-proxy* -komponentin tehtävä on ohjata liikenne oikealle kapselille. K8s versiossa 1.0 kube-proxy:llä on vain yksi välitystila, ”*userspace*”. Tässä tilassa komponentin tehtävä on ohjata palvelu-objectista tullut kutsu tälle tarkoitetulle kapselille (kuva 5). Kube-proxy tarkkailee uusia ja poistettavia palvelu-objekteja. Jos uusi palvelu-objekti luodaan. Komponentti avaa sattumavaraisen portin kätyrisolmussa tälle palvelulle. Kun palvelu-objektista tulee kutsu komponentin avattuun porttiin, ohjaa tämä yhteyden oikealle kapselille. Kube-proxy saa tiedon kenelle yhteys kuuluu etcd -tietokannasta kommunikoimalla mestari rajapinnan kanssa. Mikäli kapseleita on useampi, määräytyy yhteys round robin -säännön mukaisesti (kiertovuorottelu). K8s version 1.1 mukana tuli uusi välitystila kube-proxy:lle nimeltään ”iptables”. Tässä tilassa kube-proxy tarkkailee mestarisolmua. Jos mestari luo tai poistaa uuden palvelu-objektin, kube-proxy muuttaa oman koneensa IP taulua (iptables, palomuuri Linux-kerneleissä. Iptablesissa voi luoda filtteröinti sääntöjä IP-paketeille ja NAT sääntöjä) (kuva 5). Tällä tavoin kube-proxy ohjaa liikenteen oikealle kapselille.



Kuva 5. Vasemmalla: välitys tila: userspace; Oikealla: välitystila: ip-tables (mukailen Kubernetes, 2018c; Openstack, 2018)

Container runtime -komponentin tehtävänä on ajaa kontteja. K8s tukee useita eri runtime ohjelmia kuten Docker, containerd, cri-o, rktlet sekä muita ohjelmia, joissa on K8s CRI (Container Runtime Interface) tuki (Kubernetes, 2019a).

2.3 Kuberneteksen hyödyt

Hightower, Burns & Beda (2017) kertoivat kirjassaan neljä yleisintä hyötyä mitä K8s:n käyttäjät hakevat ottaessaan käyttöön K8s alustan.

- Nopeus (Velocity)
- Skaalautuvuus (Scaling)
- Infrastruktuurin abstrahointi (Abstracting your infrastructure)
- Tehokkuus (Efficency)

2.3.1 Nopeus

Nykyään loppukäyttäjät vaativat sivustoilta ja sovelluksilta jatkuvaa saatavuutta. Katkokset sovelluksessa tuovat huonon kuvan. Sovelluksen pakkaaminen kontteihin ja näiden hallitseminen K8s ympäristössä nopeuttaa uusien päivityksien viemistä tuotantoon ilman, että sovellukseen tulisi huoltokatkoksia. Nopeuden mahdollistaa kontteihin pakattujen sovelluksien muuttumaton rakenne, K8s konfiguraatioiden deklaratiivinen muotoilu ja K8s:n automaattinen valvonta.

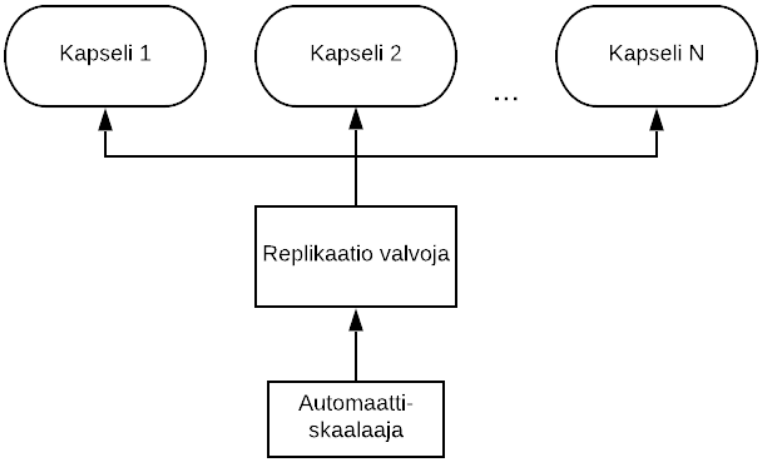
Sovelluksen *muuttumaton* rakenne perustuu siihen, miten kontteja rakennetaan. Kontit luodaan konttilevykuvasta. Kun sovellusta päivitetään ja luodaan uusia ominaisuuksia, luodaan lopputuotteesta uusi levykuva. Uudesta levykuvasta käynnistetään uusi kontti. K8s odottaa, että uusi kontti on käynnissä ja terve ennen, kuin aloittaa sammuttamaan vanhaa konttia. Tämän avulla palveluun ei muodostu katkoksia. Jo luotuja levykuvia ei ole tarkoitus muuttaa, sen sijaan jokaisesta päivityksestä tehdään uusi levykuva. Tämän etuna on muun muassa siinä, kun vanha versio sovelluksesta pitäisi palauttaa. Uusi kontti pitää vain käynnistää vanhalla levykuvalla. (Rize, 2017; Arundel & Domingus, 2019).

K8s konfiguraatiotiedostot kirjoitetaan YAML tai JSON tiedostoina. Tiedostoihin ei kirjoiteta komentoja mitä K8s tulisi tehdä. Sen sijaan konfiguraatio tiedostoon kuvaillaan deklaratiivisesti, millainen ympäristö halutaan. K8s lukee tiedoston ja päättää miten toimitaan, jotta päästään haluttuun tilaan. Esimerkiksi jos halutaan kaksi kopiota samasta kapselista, kirjoitetaan konfiguraatiotiedostoon `replicas: 2`. Konfiguraation kirjoittajan ei tarvitse kirjoittaessa tietää montako kopiota on jo olemassa, jotta pääsisi haluttuun tilaan. Vastaavasti jos samaan lopputulokseen halutaan päästä käyttämällä komentoja imperatiivisesti. Tulisi komennon antajan tietää montako kopiota on jo olemassa sillä hetkellä, jotta tiedetään tuleeko ympäristöstä poistaa vai lisätä kapseleita saavuttaakseen halutun tilan (Ali, 2018).

K8s valvoo jatkuvasti klusteria ja korjaa itse itseään. K8s kykenee tunnistamaan automaattisesti muutoksia ympäristössä ja pystyy toimimaan heti korjatakseen muutoksen. Hyvänä esimerkkinä tästä toimii replikaatiovalvoja. Kyseinen valvoja tarkistaa tietyin väliajoin ympäristössä käynnissä olevien konttien lukumäärään. Mikäli valvoja huomaa konteja olevan liikaa tai liian vähän. Käynnistää tämä uuden työn, jonka tehtävänä on saada ympäristön konttien lukumäärä samaksi kuin halutussa tilassa (Sayfan, 2017).

2.3.2 Skaalautuvuus

Kun sovellus siirtyy tuotantoon, tulee sovelluksen pystyä skaalautumaan nopeasti ja huomaamattomasti sovelluksen käyttäjille. Suorituskyvyn pullonkaulana voi olla esimerkiksi yhden mikropalvelun kontti, joka käsittelee tapahtumia hitaasti. Konttien lukumäärää voidaan K8s:sa lisätä muuttamalla kapselien määrää konfiguraatiotiedossa. Tämän voi kuitenkin myös tehdä automaattisesti K8s:sa automaatti skaalauksella. Skaalaaja kykenee muuttamaan kapselien lukumäärä automaattisesti annettujen sääntöjen puitteissa. Automaatti skaalaaja ei itse luo tai poista kopioita kapseleista. Skaalaaja kommunikoi replikaatiovalvojan kanssa, joka sitten luo tai poistaa kapseleita (kuva 6). Näin toimittaessa kapseleiden skaalaus ei luo konflikteja replikaatiovalvojan kanssa. Uudet kapselit käynnistetään vierekkäin jakaen sovelluksen kuormaa. Automaattiskaalaajalle voi määrittää metriikkaa, jonka perusteella tämä skaalaa kapseleita ylös tai alaspäin. Metriikka voi olla esimerkiksi prosessorin käyttöprosentti tai joku sovelluksen tarjoama oma arvo.



Kuva 6. Automaattiskaalaaja (mukailen Sayfan, 2017)

Kapseleita ei voi skaalata loputtomiin ylöspäin. Jokainen kapseli sijoitetaan kätyrisolmulle ja tämä käyttää solmun resursseja. Kun resurssit loppuvat klusterista, uudet kapselit jäävät odottamaan resurssien vapautumista olemassa olevilta kapseleilta tai uusien solmujen liittämistä klusteriin. Koneiden lisääminen olemassa olevaan K8s -klusteriin ei häiritse jo

klusterissa olevia sovelluksia. Uudet palvelimet rekisteröivät itsensä mestarisolmuun automaattisesti, mikäli K8s käynnistetään koneissa komennolla, josta löytyy `--register-node` lippu. (Kubernetes, 2018b).

Moni pilvipalveluntarjoaja kuten Google Cloud, Amazon Web Services ja Microsoft Azure tarjoavat hyvät työkalut K8s -klusterin skaalaamiseen. Edellä mainituilla palveluntarjoajilla on mahdollista skaalata K8s -klusteria automaattisesti. Klusterin automaattinen skaalautuminen tapahtuu, kun palvelu tunnistaa mestarisolmun yrittävän luoda uuden kapselin, mutta kätyrisolmujen resurssit eivät riitä tai eivät vastaa kapselin vaatimuksia. Tällöin palvelu liittää uuden solmun K8s -klusteriin ennalta määrättyjen sääntöjen mukaisesti. Vastaavasti palvelut osaavat skaalata klusteria myös alaspäin (Arundel & Domingus, 2019).

Tutkimuksessa tulemme vielä tutkimaan AWS:n tarjoamaa Amazon Elastic Container Service for Kubernetes (EKS) palvelua syvemmin.

2.3.3 Infrastruktuurin abstrahointi

K8s ei ole riippuvainen ympärillä olevasta ympäristöstä. Tämä on nähty yhtenä K8s:n suurimmista hyödyistä. K8s:n voi pystyttää omaan konesaliin, pilviympäristöön tai hyödyntää kumpaakin hybridipilvessä. Tämä ei siis lukitse käyttäjäänsä yhteen tiettyyn ympäristöön tai palveluntarjoajan alustaan (vendor lock-in). Käyttäjä voi siirtää koko K8s -järjestelmänsä helposti omasta konesalistaan esimerkiksi pilvipalveluntarjoajalle, käyttäen melkein samoja konfiguraatiotiedostoja (Hightower & ym. 2017; Jayanandana, 2018)

2.3.4 Tehokkuus

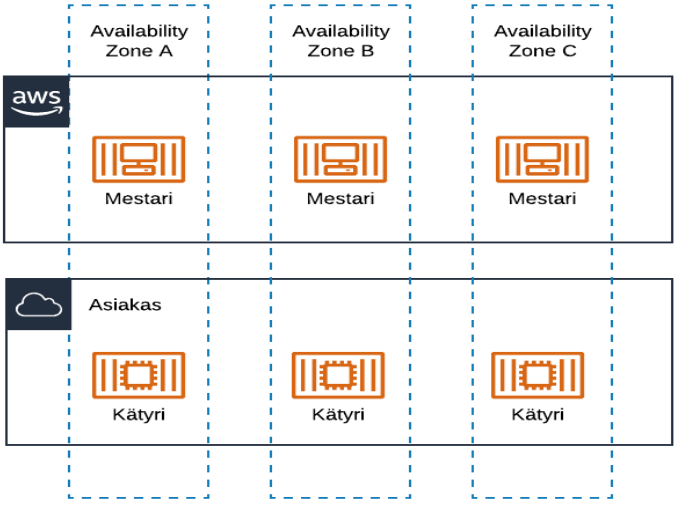
Konttiteknologian ansiosta K8s:n käyttäminen on tehokasta. K8s osaa automaattisesti sijoittaa kapseleita koneisiin, joihin nämä mahtuvat. Tämän ansiosta jokaisen koneen resursseja käytetään tehokkaasti hyödyksi ja koneita ei tarvita ympäristössä niin paljon. Näin voidaan välttää tarpeettomien laitteiden hankinta omassa konesalissa ja säästää rahaa ylimääräisten resurssien maksamisesta pilvipalveluissa (Hightower & ym. 2017; Sanders, 2018).

3 Amazon Elastic Container Service for Kubernetes

Amazonin tarjoama palvelu K8s:ta on nimeltään Amazon Elastic Container Service for Kubernetes. Tutkimuksessa käytetään lyhennettä EKS viitattaessa Amazonin Kubernetes palveluun.

3.1 AWS EKS tausta

AWS avasi EKS -palvelun asiakkailleen 2018 kesäkuussa. EKS -palvelussa järjestelmän vastuu jakautuu palveluntarjoajan AWS:n ja asiakkaan kanssa (kuva 7). AWS takaa pilvipalveluntarjoajana laitteiston, ohjelmiston, verkkoyhteyksien toimivuuden ja fyysisen turvauksen konesaleissa. Edellä mainittujen lisäksi AWS huolehtii EKS -palvelussa K8s -klusterin ohjaamon konfiguraatiosta ja toimivuudesta sekä etcd -tietokannan saatavuudesta. Mestarisolmujen ja tietokannan saatavuuden takaamiseksi AWS ajaa useita mestareita usealla eri saatavuusalueella (availability zones, fyysesti eri paikoissa olevia konesaleja). Toiminta jatkuu tällä tavoin normaalisti, vaikka yksi saatavuusalueista putoaisi pois verkosta esimerkiksi sähkökatkoksen vuoksi. AWS tarkkailee myös jatkuvasti mestarisolmuja ja korvaa automaattisesti vialliset solmut uusilla. Asiakkaan vastuulla ovat K8s -klusterin muut konfiguraatiot, esimerkiksi kätyrisolmujen konfiguraatio, sekä ohjaamon yhteys asiakkaan VPC -verkkoon. Asiakkaan vastuulla on myös kätyrisolmujen pystyttäminen, käyttöjärjestelmien valitseminen ja päivittäminen (Amazon Web Services, 2018a).



Kuva 7. EKS jaettu vastuu (mukailen Amazon Web Services, 2018a)

3.2 EKS:n toiminta AWS ympäristössä

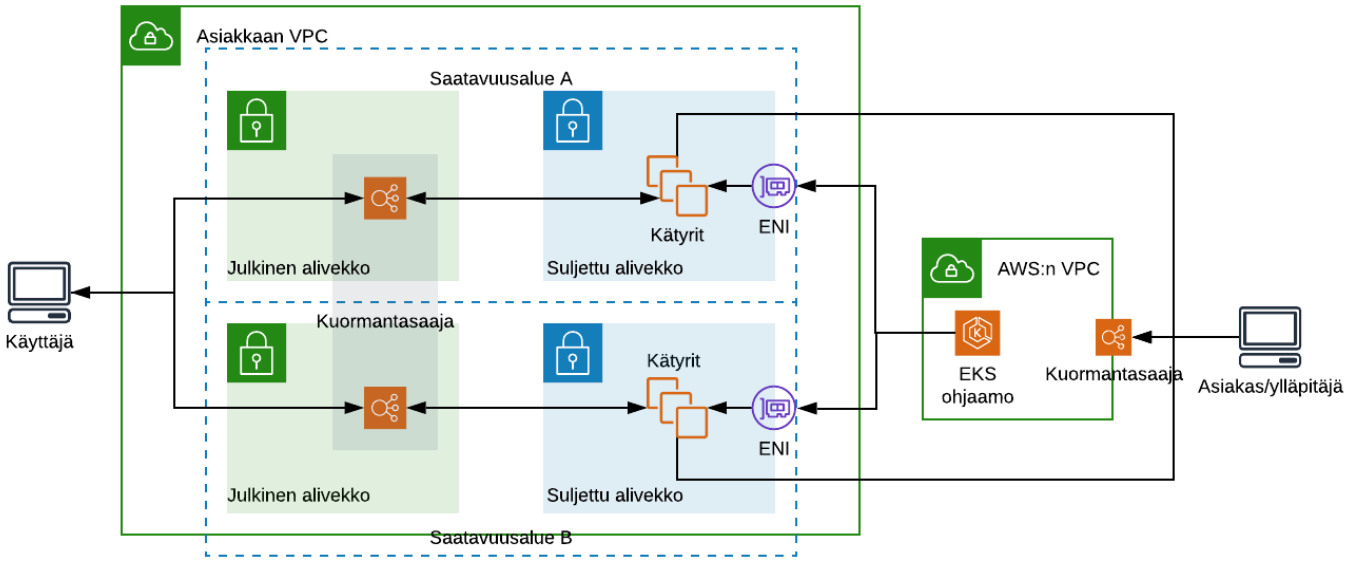
EKS palvelu toimii muiden AWS:n olemassa olevien palveluiden kanssa. Nämä palvelut tukevat ja parantavat EKS palvelun skaalautumista ja tietoturvaa. AWS -palvelut, jotka toimivat suoraan EKS -palvelun kanssa ovat: Elastic Load Balancing, IAM, VPC ja PrivateLink.

Elastic Load Balancing (ELB) on AWS:n tarjoama kuormantasaajapalvelu. EKS tukee kahden tyyppistä ELB -kuormantasaajaa, Network Load Balancer (NLB) ja Classic Load Balancer (CLB). Oletuskuormantasaajatyyppi on CLB. NLB jakaa kuormaa hyödyntäen TCP tai TLS protokollaa. Kuormantasaajilla on oma IP-osoite, joka näkyy ulkoverkkoon. NLB:n ohjauksessa välittyy alkuperäisen kutsun tekijän IP osoite. Osoitetta voidaan hyödyntää tällöin myös sovelluksessa. Protokollatason ohjauksen ansiosta NLB on erittäin nopea ja tehokas. Tämän kuormantasaajamallin yhteensopivuus k8s:een on vielä alpha-vaiheessa, eikä sitä suositella vielä k8s tuotantoklustereihin. CLB toimii samalla tavalla kuin palvelu-objekti jonka tyyppi on *LoadBalancer* (otsikko 2.2.3). (Bala 2017; Kubernetes, 2018c)

EKS palvelu luo käyttäjän puolesta resursseja, joita hyödynnetään K8s -klusterissa. EKS tarvitsee oikeudet näiden resurssien luontiin. *IAM* on AWS:n identiteetin- ja pääsynhallinta -palvelu. Palvelun ideana on vastata kysymykseen: Kenellä on oikeus tehdä mitä ja missä? IAM palvelussa määritellään rooleja, käyttäjiä ja ryhmiä. Näihin objekteihin liitetään policy, jossa määritellään oikeuksista. AWS kertoo dokumentaatiossaan vähimmäisoikeudet EKS palvelulle (Amazon Web Services, s.a. (a)).

VPC on virtuaaliverkko johon käyttäjä voi luoda AWS resurssejaan. VPC:n sisälle luodaan aliverkkoja. Aliverkot määritellään saatavuusalueille ja ne voivat olla joko julkisia tai suljettuja verkkoja. Julkiseen aliverkkoon voidaan olla yhteydessä VPC -verkon ulkopuolelta. Suljettuun aliverkkoon ei voida olla yhteydessä verkon ulkopuolelta, mutta suljetusta aliverkosta voidaan tarvittaessa ottaa yhteys VPC verkon ulkopuolelle. EKS vaatii klusterin luonnin yhteydessä vähintään kaksi aliverkkoa eri saatavuusalueilta. AWS suosittelee kuormantasaajan sijoittamista julkiseen aliverkkoon ja kätyrisolmujen sijoittamista suljetuille aliverkoille (kuva 8). AWS luo EKS -klusterin luonnin yhteydessä asiakkaan aliverkkoihin Elastic Network Interface (ENI) resurssin, mikäli klusterin oikeudet tämän sallivat. Klusterin ohjaamo käyttää ENI -resurssia kommunikoidakseen kätyreiden kanssa. Klusterin ylläpitäjä ja kätyrisolmut ottavat yhteyttä klusterin ohjaamon API palvelimelle kuormantasaajan kautta. Ohjaamon API palvelin voidaan muuttaa myös yksityiseksi. Tällöin yhteydet API palvelimeen menevät ENI resurssin kautta. Tämä vaatii ylläpitäjää luomaan

VPC:n sisälle hyppypalvelimen, jonka kautta ylläpitäjä voi ottaa yhteyttä klusterin ohjaamoon. Hyppypalvelin on virtuaalikone, johon pääsee ulkoverkosta ja jolla on pääsy sisäverkkoon.



Kuva 8. AWS:n suosittelema verkkoinfrastruktuuri EKS palvelulle (Amazon Web Services, s.a (b))

3.3 Kilpailijat

Monet pilvipalveluntarjoajat ovat ottaneet K8s tarjottavaksi palveluksi. AWS:n kilpailijoita K8s pilvipalveluntarjoajana ovat esimerkiksi Googlen GKE, Microsoftin AKS ja DigitalOceanin Kubernetes. Acreman, S. (s.a.) julkaisi artikkelissaan taulukon (liite 1). Taulukossa vertaillaan pilvipalveluntarjoajien K8s palveluja.

GKE on Googlen täysin hallittu K8s palvelu. Asiakas määrittelee, kuinka monta kätyrisolmua klusteriin tulee ja Google luo tämän perusteella koko klusterin. Google huolehtii, että mestarisolmuissa ja kätyrisolmuissa on uusimmat tietoturvapäivitykset. Palvelu tunnistaa vialliset solmut ja korvaa solmut uusilla. Klusterin saatavuudessa on kaksi tilaa: *zonal* ja *regional*. Zonal -tilassa klusterilla on vain yksi mestarisolmu. Regional tilassa mestarisolmuja on useampi ja nämä on hajautettu eri saatavuusalueille. Googlella on K8s:ta eniten kokemusta ja Arundel, J & Domingus, J (2019) mukaan tämä näkyy heidän palvelussaan. Heidän mielestään Google on paras K8s palveluntarjoaja, joka on myös tehnyt K8s käyttöönoton nopeaksi, edulliseksi ja yksinkertaiseksi heidän tarjoamien työkalujen ansiosta.

AKS tarjoaa GKE tapaan täysin hallitun K8s palvelun. Microsoft Azure ylläpitää ja hoitaa mestarisolmujen ja kätyrisolmujen tietoturvapäivitykset. AKS, GKE ja DigitalOcean Kubernetes palveluissa ei veloiteta mestarisolmujen pystyttämisestä eikä ylläpidosta. Palveluissa veloitetaan muista maksullisista resursseista käytön mukaan, esimerkiksi kätyrisolmujen pilvipalvelimet. AKS palvelussa mestarisolmuja on vain yksi jokaista klusteria kohden (Selvan, 2018; Microsoft Azure, 2019).

DigitalOcean julkaisi oman K8s palvelun (DOK8s, epävirallinen lyhenne) joulukuussa 2018. Haider, H (2019) mukaan DOK8s on yksinkertainen ja kustannustehokas. DOK8s hallinnoi klusterin ohjaamoa ja tarkkailee klusterin kätyrisolmuja. Klusterille luodaan yksi tai useampi kätyrisolmuvaranto (minion node pool). Samanlaisia ominaisuuksia löytyy myös GKE ja EKS palveluissa. Kaikki kätyrisolmut varannon sisällä ovat identtisiä. DOK8s poistaa tarvittaessa vialliset solmut varannosta ja korvaa nämä uudella. DOK8s palvelussa koko klusteri pystytetään yhdelle saatavuusalueelle. Tämä huonontaa klusterin vikasietoisuutta, esimerkiksi paikallinen sähkökatkos kaataa koko sovelluksen. DOK8s klustereissa mestarisolmuja on vain yksi kappale (DigitalOcean, 2018).

4 Harjoitusympäristö

Tässä otsikossa käsitellään sovellusta, joka hyöyntää AWS EKS palvelua. Harjoituksessa käytämme olemassa olevaa sovellusta nimeltä strm/helloworld-http (https://hub.docker.com/r/strm/helloworld-http/). Sovellus on yksinkertainen verkkosivu, joka palauttaa kontin nimen, jossa sovellusta pidetään ajossa.

4.1 Suunnitelma

Harjoituksen pääpainona on sovelluksen arkkitehtuuri. Sovellusta ajetaan konteissa. Kontteja ajetaan useita vierekkäin eristettyinä toisistaan ja näitä hallitaan AWS EKS palvelun avulla. Sovellukselle luodaan oma VPC -verkko. Virtuaaliverkko pitää sisällään kaksi julkista aliverkkoa ja kaksi suljettua aliverkkoa. Aliverkot sijaitsevat kahdella eri saatavuusalueella. Julkisissa aliverkoissa on kuormantasaaja. Suljetuissa aliverkoissa on kummassakin oma pilvipalvelin. Sovelluksen verkkorakenne on samanlainen kuin kuvassa 8. Suljetun aliverkon pilvipalvelimet yhdistetään AWS EKS palvelun K8s -klusteriin.

4.2 Toteutus

Tässä kappaleessa käsitellään harjoitusympäristön toteutusta. Kaikki toteutuksessa käytetyt koodit ja mallitiedostot löytyvät GitHub säilytyspaikasta (https://github.com/heikkima/heikki-thesis-2019). Kaikki AWS resurssit luodaan AWS CloudFormation -palvelulla. CloudFormation on palvelu, joka luo AWS resurssit automaattisesti. Palvelulle lähetetään YAML tai JSON -mallitiedosto. Tiedostossa tulee kuvailla minkälaisia AWS resursseja halutaan ja miten nämä ovat mahdollisesti konfiguroitu. Tämä mahdollistaa ympäristöjen monistamisen automaattisesti ja nopeasti. Mallitiedostot voidaan myös säilyttää versiohallintajärjestelmässä, kuten GitHub säilytyspaikassa. CloudFormation mallitiedostojen tulee noudattaa AWS:n omaa dokumentaatiota (Amazon Web Services, s.a (c)).

AWS CloudFormation -palvelun kanssa käytetään harjoituksessa Sceptre -työkalua. Sceptre helpottaa CloudFormation -mallien hallintaa ja tarjoaa hyvät työkalut ympäristöjen luomiseen nopeasti. Muita ohjelmia, joita hyödynnetään harjoitusympäristössä, ovat AWS CLI ja kubectl. AWS CLI tarjoaa komentoriviltä kommunikoinnin AWS ympäristöön. Kubectl on komentorivityökalu, jonka avulla kommunikoidaan K8s -klusterin ohjaamon kanssa. Kaikki K8s konfiguraatiotiedostot on YAML muodossa harjoituksessa. Kaikki konfiguraatiotiedostot lähetetään ohjaamoon kubectl komennolla `kubectl apply -f <tiedoston nimi>`.

4.2.1 Virtuaaliverkko

Harjoitusympäristölle luodaan VPC -verkko. Verkko pitää sisällään kaksi julkista aliverkkoa ja kaksi suljettua aliverkkoa. VPC -verkon luomista varten luodaan CloudFormation mallitiedosto. Mallitiedosto aloitetaan ilmoittamalla mitä AWS formaatti versiota käytetään (Kuva 9). Harjoituksessa käytetään uusinta versiota, eli 2010-09-09. `Parameters:` -otsikon alle kuvaillaan minkä nimisiä muuttujia mallitiedostossa käytetään. Kyseisessä mallitiedostossa käytetään muuttujia `Project` ja `CidrBlock`. `Project` muuttujan arvoksi tulee projektin nimi. `CidrBlock` muuttujaa käytetään mallitiedoston ensimmäisessä resurssissa `Vpc`, joka on tyyppiä `AWS::EC2::VPC`. Yksi `AWS::EC2::VPC` -tyyppisten resurssien vaadituista arvoista on `CidrBlock`. Tämän arvo määrittelee koko VPC -verkon CIDR-notaation ja samalla verkon IP avaruuden. Esimerkiksi arvo `10.20.0.0/16` rajaisi VPC-verkon IP osoitteet `10.20.0.0 – 10.20.255.255` sisälle (Amazon Web Services, s.a (d)).

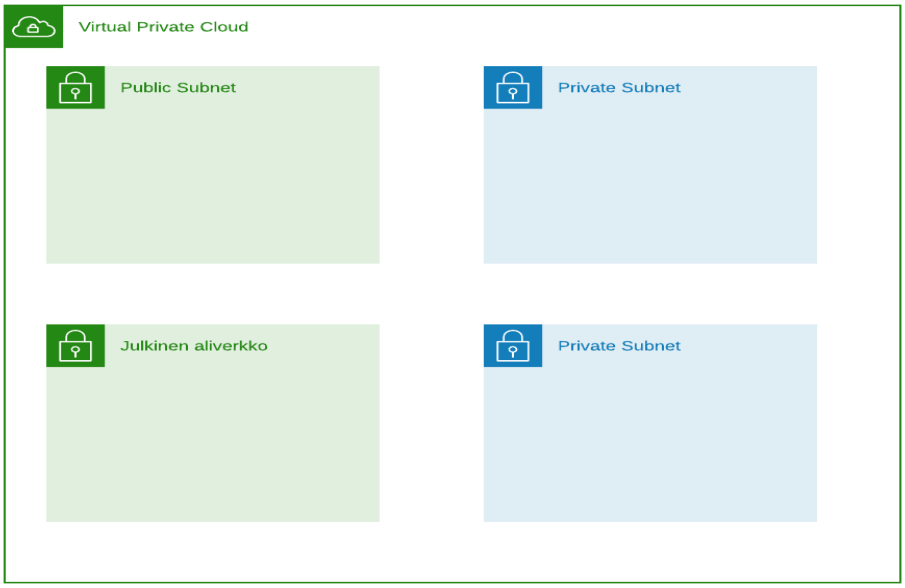
```yaml
AWSTemplateFormatVersion: 2010-09-09
Description: VPC for EKS sample

Parameters:
  Project:
    Description: Project name
    Type: String
  CidrBlock:
    Description: CIDR block for VPC
    Type: String

Resources:
  Vpc:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref CidrBlock
      EnableDnsHostnames: true
      EnableDnsSupport: true
      Tags:
        - Key: Name
          Value: !Sub ${Project}-VPC
```

Kuva 9. VPC mallitiedoston alku

VPC mallitiedostossa luodaan myös muita tarvittavia resursseja VPC -verkolle. Muita resursseja ovat esimerkiksi kaksi (2) kappaletta julkisia aliverkkoa ja kaksi (2) kappaletta suljettuja aliverkkoa (Kuva 10). Tämä kaikki toimii harjoituksen perustuksena.

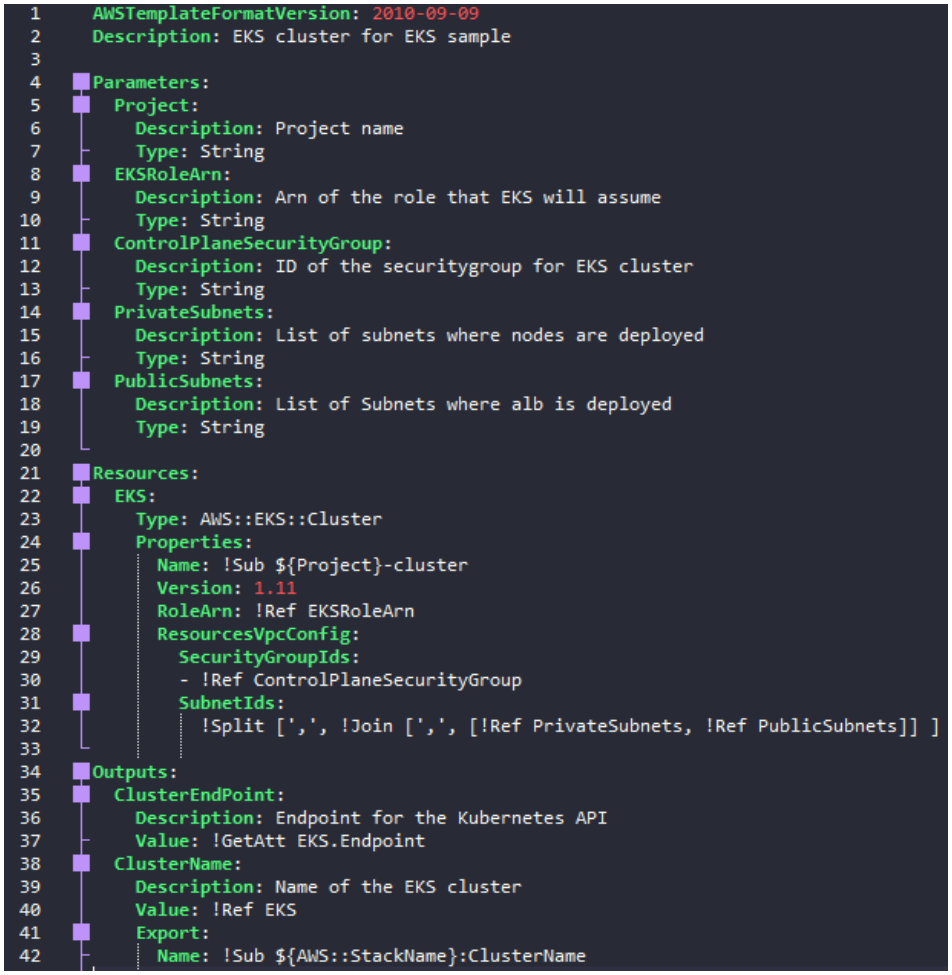


Kuva 10. Luotu VPC

4.2.2 EKS -klusteri

VPC -verkon luotua voidaan siirtyä sovelluksen K8s ohjaamon rakentamiseen. Ohjaamolle tulee luoda IAM rooli, joka määrittelee EKS -klusterin oikeudet. AWS dokumentaation mukaan vähimmäisoikeudet EKS -klusterin toiminnalle on kerätty heidän luomalle policy -listoille `AmazonEKSClusterPolicy` ja `AmazonEKSServicePolicy` (Amazon Web Services, s.a. (a)).

Luotu rooli syötetään parametrina EKS mallitiedostossa (Kuva 11). EKS resurssi omaksuu annetun roolin, joka annetaan `RoleArn` -arvoksi. EKS pystyy tämän jälkeen tekemään vain toimintoja, joita roolin policy -listoille on määritelty. EKS resurssille on tärkeä antaa `SubnetIds` arvoiksi suljettujen ja julkisten aliverkkojen ID tunnisteet. Tällä tavoin EKS-palvelu tietää mihin aliverkkoon kuormantasaajat ja kätyrisolmut luodaan. Harjoituksessa ei valittu uusinta tuettua K8s 1.12 versiota koska tämän tuki tuli äskettäin vasta EKS -palvelulle. AWS ei myöskään ole vielä omaa dokumentaatiotaan päivittänyt tämän mukaiseksi.



```yaml
AWSTemplateFormatVersion: 2010-09-09
Description: EKS cluster for EKS sample

Parameters:
  Project:
    Description: Project name
    Type: String
  EKSRoleArn:
    Description: Arn of the role that EKS will assume
    Type: String
  ControlPlaneSecurityGroup:
    Description: ID of the securitygroup for EKS cluster
    Type: String
  PrivateSubnets:
    Description: List of subnets where nodes are deployed
    Type: String
  PublicSubnets:
    Description: List of Subnets where alb is deployed
    Type: String

Resources:
  EKS:
    Type: AWS::EKS::Cluster
    Properties:
      Name: !Sub ${Project}-cluster
      Version: 1.11
      RoleArn: !Ref EKSRoleArn
      ResourcesVpcConfig:
        SecurityGroupIds:
        - !Ref ControlPlaneSecurityGroup
        SubnetIds:
          !Split [',', !Join [',', [!Ref PrivateSubnets, !Ref PublicSubnets]] ]

Outputs:
  ClusterEndPoint:
    Description: Endpoint for the Kubernetes API
    Value: !GetAtt EKS.Endpoint
  ClusterName:
    Description: Name of the EKS cluster
    Value: !Ref EKS
    Export:
      Name: !Sub ${AWS::StackName}:ClusterName
```

Kuva 11. EKS mallitiedosto

EKS mallitiedoston resurssien luotua on harjoitusympäristöön luotu K8s ohjaamo. Mallitiedostoon on määritelty ulostuloksi K8s -klusterin nimi. Klusterin nimi on tärkeä ottaa talteen, koska tätä tarvitaan silloin kun yhdistetään kätyrisolmut mestariin ja kun ylläpitäjä muodostaa ensimmäisen kerran yhteyden ohjaamoon.

EKS -klusterin ohjaamo tulee konfiguroida sallimaan yhteys järjestelmän valvojalta. EKS-klusteri voidaan konfiguroida nopeasti AWS CLI:n tarjoamalla komennolla:

```
aws eks --region <region> update-kubeconfig --name <cluster_name>
```

`<region>` tulee korvata klusterin AWS aluetunnuksella, esimerkiksi eu-west-1. `<cluster_name>` korvataan klusterin nimellä. Mikäli komento palauttaa virheviestin tulee järjestelmänvalvojan tarkistaa omat IAM-oikeudet ja varmistaa että oikeudet riittävät toimintoon. Järjestelmänvalvoja voi tarkistaa yhteyden toimivuuden EKS -klusterin ohjaamoon kubectl -komennolla. Komento palauttaa tietoa klusterista, mikäli yhteys toimii.

```
kubectl get svc
```

4.2.3 Kätyrisolmut

Kätyreitä luodessa tulee pitää mielessä tietoturva. Vaikka kätyrit luodaan VPC -verkon suljettuun aliverkkoon, on hyvä käytäntö luoda securitygroup -resurssi. Tällä resurssilla voidaan määritellä, mistä voidaan ottaa yhteyttä kätyrisolmuihin ja mihin portteihin yhteydenotto sallitaan. Kuvassa 12 luodaan kaksi resurssia. Molemmat resurssit lisäävät säännön securitygroup -resurssin sisälle. `AWS::EC2::SecurityGroupIngress` -tyyppinen resurssi luo säännön tulevasta liikenteestä. Vastaavasti `AWS::EC2::SecurityGroupEgress` luo säännön ulos lähtevästä liikenteestä. Mikäli lähtevä tai tuleva yhteys ei ole sääntöjen mukainen, tiputetaan tämä pois ja yhteys ei koskaan pääse perille asti. `GroupId` määrittelee resursseissa mitä securitygroup -resurssia sääntö koskee. Resursseissa tulee määritellä, mistä porteista liikenne sallitaan tai mihin porttiin yhteys voi mennä määränpäässä. Harjoitusympäristössä sallitaan yhteys kätyreistä ohjaamoon porteista 1025 – 65535 ja 443. AWS dokumentaatio suosittelee käyttämään näitä portteja (Amazon Web Services, s.a (e)). Securitygroup -resurssiin ei tarvitse määritellä sääntöä järjestelmänvalvojan pääsystä ohjaamoon.

```
NodeSecurityGroupFromControlPlaneIngress:
  Type: AWS::EC2::SecurityGroupIngress
  DependsOn: NodeSecurityGroup
  Properties:
    Description: Allow worker Kubelets and pods to receive communication from the cluster control plane
    GroupId: !Ref NodeSecurityGroup
    SourceSecurityGroupId: !Ref ControlPlaneSG
    IpProtocol: tcp
    FromPort: 1025
    ToPort: 65535

ControlPlaneEgressToNodeSecurityGroup:
  Type: AWS::EC2::SecurityGroupEgress
  DependsOn: NodeSecurityGroup
  Properties:
    Description: Allow the cluster control plane to communicate with worker Kubelet and pods
    GroupId: !Ref ControlPlaneSG
    DestinationSecurityGroupId: !Ref NodeSecurityGroup
    IpProtocol: tcp
    FromPort: 1025
    ToPort: 65535
```

Kuva 12. SecurityGroup sääntöjen määrittely mallitiedostossa

Kätyreitä voidaan luoda EKS -klusteriin käsin, mutta tämä ei ole suositeltavaa tuotantokäytössä. Tämä lisää riskiä inhimillisille virheille ja vaikeuttaa klusterin skaalautumista nopeasti. AWS ympäristössä on mahdollista ryhmittää pilvipalvelininstanssit yhdeksi varannoksi Auto Scaling Group:illa (lyh. ASG). ASG:ssa voidaan määrittää instansseihin liittyviä konfiguraatioita, kuten minkälaisia instansseja halutaan, kuinka monta pilvipalvelininstanssia voi olla samanaikaisesti päällä ja kuinka monta voi olla vähimmillään. Tämän lisäksi ASG -konfiguraatioon voidaan lisätä komentosarja, joka ajetaan aina kun instanssi luodaan. Jokainen ASG:n luoma instanssi on käynnistyessä samanlaisia. Jokainen instanssi saa saman IAM -roolin, joka määrittelee mitä oikeuksia instansseilla on.

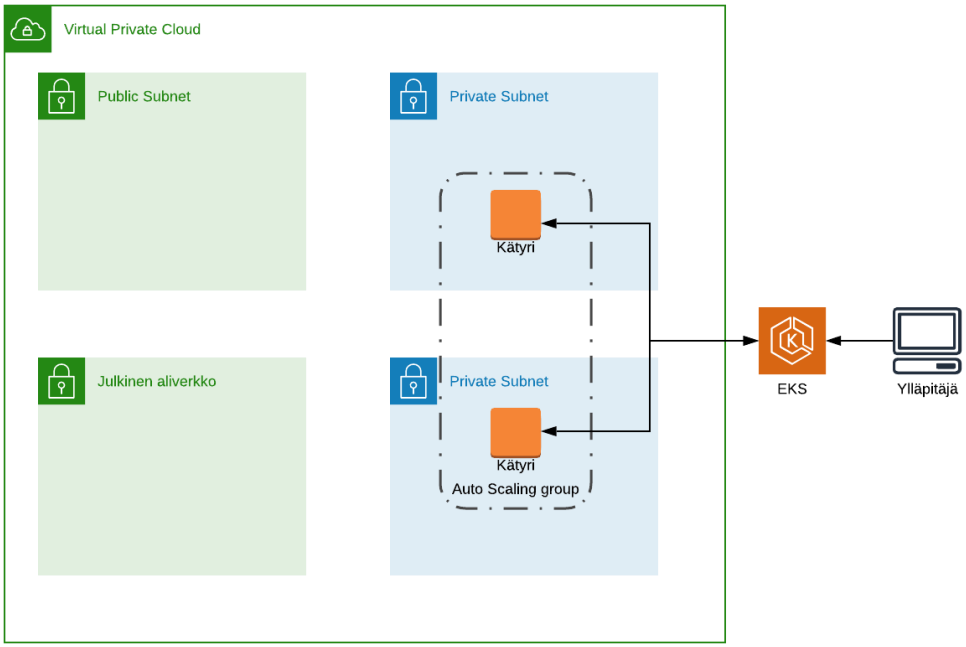
Harjoituksessa ASG käynnistää kaksi (2) instanssia heti ja ajaa kummassakin komentosarjan. Komentosarja käynnistää kubelet -komponentin, joka yrittää jatkuvasti ottaa yhteyttä klusterin ohjaamoon. Yhteydenotto ei onnistu ennen kuin ohjaamolle ilmoitetaan mitä IAM -roolia instanssit käyttävät. Tiedon ilmoittaminen ohjaamolle tapahtuu k8s konfiguraatiotiedostolla (Kuva 13). Tiedostossa tulee vain korvata `rolearn` arvoksi instanssien IAM -rooli tunnus (Amazon Web Services, s.a (f)).

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: <node instance role arn> # Replace with node instance role arn
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
```

Kuva 13. EKS -klusterille kätyrisolmujen yhdistämistiedosto

Tässä kohtaa harjoitusympäristöön on luotu VPC -verkko, EKS -klusteri ja kaksi (2) kappaletta kätyrisolmuja (Kuva 14). Kätyrit ovat konfiguroitu kommunikoimaan EKS ohjaamon kanssa ja järjestelmänvalvojakin pystyy ottamaan yhteyttä ohjaamoon. Järjestelmänvalvoja voi tarkistaa yhdistyneiden ohjaamoiden tilan seuraavalla kubectl komennolla.

```
kubectl get nodes
```



Kuva 14. Harjoitusympäristön välikatsaus kaaviolla

4.2.4 Sovelluskontin rakentaminen

Sovelluksien vieminen K8s ympäristöön tapahtuu konfiguraatiotiedostolla (Kuva 15). Konfiguraatiotiedostoon tulee asettaa tyypiksi `Deployment`. Tämä kertoo ohjaamolle, että kyseessä on sovelluksen luonti tai päivitys. Ohjaamo osaa automaattisesti luoda tarvittavat komponentit sovellukselle, esimerkiksi kapselit. Koko konfiguraatiotiedostolle voi asettaa yleistietoja ylimpään `metadata` -kenttään. `spec.template` sisälle määritellään sovellukselle nimi, joka leimataan kaikkiin sovelluksen kapseleihin. `spec.selector` -kenttään määritellään vastaavasti minkä nimisiä kapseleita korvataan/muutetaan, mikäli näitä on jo olemassa. Kaikki klusterissa käynnissä olevien ohjelmien nimet ja kapselien lukumäärän saa selville seuraavalla kubectl komennolla.

```
kubectl get deploy
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sample-deployment
  labels:
    app: sample
spec:
  replicas: 2
  selector:
    matchLabels:
      app: sample
  template:
    metadata:
      labels:
        app: sample
    spec:
      containers:
      - name: sample
        image: strm/helloworld-http
        ports:
        - containerPort: 80
```

Kuva 15. Harjoitussovelluksen deployment konfiguraatio

4.2.5 Kuormantasaaja

Vaikka sovellus on luotu harjoitusympäristöön, ei voida siihen ottaa yhteyttä klusterin ulkopuolelta. Harjoitussovellus tarvitsee palvelu-objektin, jotta tähän voi ottaa yhteyttä klusterin ulkopuolelta. Kappaleessa 2.2.3 Service käydään läpi erilaisia palvelu-objektin tyyppejä. Yksi mainituista tyypeistä on LoadBalancer. Harjoitusympäristössä käytetään kyseistä tyyppiä. EKS luo automaattisesti LoadBalancer tyyppisen palvelu-objektin julkiselle aliverkolle. Mikäli EKS ei löydä yhtään julkista aliverkkoa klusterin verkosta, palauttaa tämä virheilmoituksen. Palvelu-objekti luodaan K8s konfiguraatiotiedostolla. Konfiguraatiotiedostoon tulee määritellä, minkä nimiselle sovellukselle palvelu-objektia luodaan, mitä protokollaa käytetään ja mikä portti avataan ulkoverkkoon.

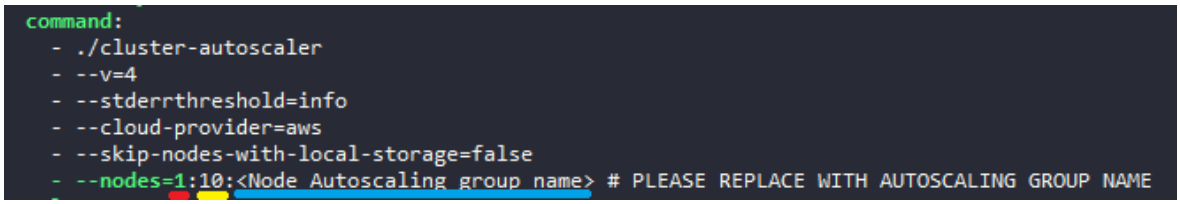
Harjoitusympäristössä sovellus on HTTP -sivusto, joten portti 80 avataan palvelu-objektissa ulkoverkkoon.

Kun sovelluksen kuormantasaaja on luotu, näkyy tämä AWS konsolissa. Konsolista löytää myös kuormantasaajan DNS osoitteen, jonka kautta pääsee ottamaan yhteyttä sovellukseen. Kun osoitetta haetaan selaimesta, avautuu sovelluksen sivusto (Kuva 16). Koska kuormantasaaja ohjaa liikenteen eri konteille, palautuu sivuston mukana eri konttien nimet, vaikka DNS -osoite on sama.

Kuva 16. Harjoitussovelluksen avaaminen selaimella

4.2.6 Klusterin skaalautuminen

AWS on luonut valmiin EKS konfiguraatiotiedoston, jolla voidaan skaalata klusterin kätyrisolmujen määrää (https://eksworkshop.com/scaling/deploy_ca.files/cluster_autoscaler.yml). Konfiguraatiotiedoston lopussa on `command` -kenttä, jossa määritellään parametreja klusterin automaatti skaalaamiselle (Kuva 17). Viimeinen rivi kyseisessä kentässä määrittelee automaatti skaalaamisen rajat ja mitä ASG resurssia käytetään. Järjestelmävalvojan ei tarvitse konfiguraatiotiedostossa muuttaa muita arvoja kuin `command` -kentän viimeisen parametrin arvot.



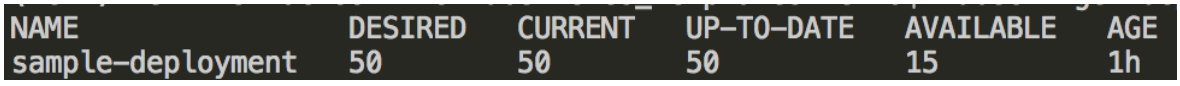
Kuva 17. Auto skaalaamisen parametrit

command -kentän viimeinen parametri selitettynä (Kuva 17):

- Punainen
  - Vähimmäislukumäärä instansseja yhtäaikaisesti ajossa
    - Tämä tulee olla sama tai suurempi kuin ASG -resurssissa määritelty
- Keltainen
  - Enimmäislukumäärä instansseja yhtäaikaisesti ajossa
    - Tämä tulee olla sama tai pienempi kuin ASG -resurssissa määritelty
- Sininen
  - ASG -resurssin nimi

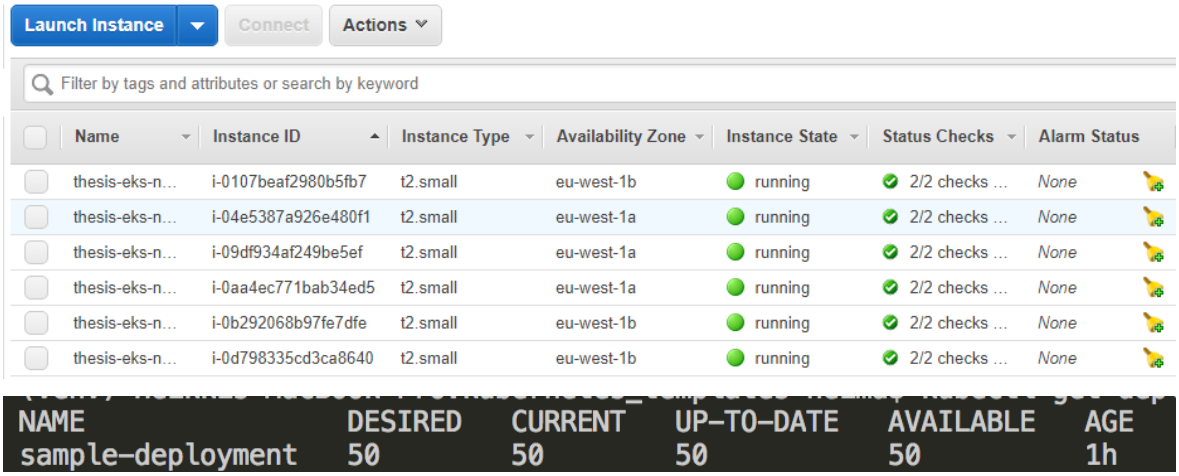
Harjoitusympäristöön luodaan yhden kopion sovellus nimeltä *cluster-autoscaler*. Sovellukselle annetaan konfiguraatiotiedostossa lupa käyttää ohjaamon rajapintaa hyödyksi, jotta tämä saa tarvittavat tiedot kätyrisolmuista ja muista sovelluksista klusterissa. Näiden tietojen perusteella sovellus skaalaa klusteria ylöspäin ja alaspäin automaattisesti hyödyntäen kätyrisolmujen ASG -resurssia. Cluster-autoscaler -sovellusta ajetaan normaalisti kätyrisolmuissa. Tämä tarkoittaa, sitä että kätyrisolmun IAM -rooleissa tulee olla oikeus hallinnoida ASG -resurssia.

Kun konfiguraatiotiedosto on lähetetty klusterin ohjaamolle, voidaan harjoitussovelluksen kopioiden määrää nostaa. Kopioiden määrän nostaminen tapahtuu kappaleessa 4.2.4 luodun deployment konfiguraatiotiedoston `replicas` arvoa muuttamalla. Pian kopioiden määrän nostamisen jälkeen voidaan huomata, että uusien kapseleiden luominen pysähtyy (Kuva 18). K8s -klusterin ohjaamo jää odottamaan vapautuvia resursseja tai uusia kätyrisolmuja, joihin jonossa olevat kapselit voidaan sijoittaa.



Kuva 18. Harjoitussovelluksen replikaatiojono

Cluster-autoscaler -sovellus huomaa, että klusterin ohjaamolle on muodostunut jono. Sovellus skaalaa automaattisesti ASG -resurssia ylöspäin ja luo samalla tarvittavan määrän uusia kätyrisolmuja, jotta jono saadaan purettua. Kuvasta 19 nähdään, että automaatti skaalaaja on luonut yhteensä neljä (4) instanssia lisää alkuperäisen kahden (2) instanssin avuksi. Kaikki viisikymmentä kapselia on luotu kätyrisolmuihin ja jokaiseen kapseliin on mahdollista ottaa yhteys kuormantasaajan kautta.



Kuva 19. Klusterin tila auto skaalaamisen jälkeen

Cluster-autoscaler -sovellus skaalaa klusteria automaattisesti myös alaspäin. Tämä tunnistaa kätyrisolmut, joissa ei ole kapseleita ajossa. Sovellus odottaa noin kymmenen (10) minuuttia ennen kuin sammuttaa instanssin.

5 Pohdinta

K8s ympäristö on tarkoitettu sovelluksille, jotka ovat tilattomia. Tilalliset sovellukset kuten tietokannat, eivät sovellu k8s ympäristöön. Tilalliset sovellukset menettävät muistissa olevat datat, kun sovelluksen kontti tuhotaan. K8s on todella hyvä konttiorkestrointityökalu projekteille, joissa on useita eri mikropalveluita. Tämä tarjoaa keskitetyn hallinnan jokaiseen palveluun järjestelmän sisällä. Sovelluksien päivittäminen ja skaalaaminen on yksinkertaista ja nopeaa. Järjestelmä kykenee ylläpitämään toiminnallisuutensa automaattisesti ja vikatilanteissa korjaa itse itsensä. Yksinkertaisen K8s -klusterin luominen pilvipalvelussa on helppoa AWS EKS -palvelun ja olemassa olevien konfiguraatiotiedostojen avulla. Klusterin muuttaminen erikoistilanteissa vastaamaan omia tarpeita ja vaatimuksia saattaa kuitenkin olla hankalaa. Monimutkaisen konfiguraationtiedoston luominen vaatii laajaa K8s konseptien ja käsitteiden tuntemusta.

AWS EKS on hyvä alku K8s -palvelulle. Tämä on kuitenkin ominaisuuksiltaan yksi heikoimmista palveluista, kun verrataan muihin pilvipalveluntarjoajien K8s -palveluihin (Liite 1). EKS -palvelun käyttäjälle jää kokonaan kätyrisolmujen ylläpitäminen ja K8s -klusteriin yhdistäminen. Tämä on hidasta ja aikaa vievää työtä, jota ei tarvitse huolehtia kilpailijoiden vastaavissa palveluissa. Googlen ja Microsoftin K8s -palveluissa klusterin ja kätyreiden luominen on täysin automatisoitu. Palvelut liittävät luodut kätyrisolmut automaattisesti klusteriin ja ylläpitävät kätyreitä ylläpitäjän puolesta. Samankaltaiseen lopputulokseen voitaisiin päästä, mikäli AWS tuo heidän ECS -konttihallintapalvelusta tutun Fargate -ominaisuuden EKS -palveluun. Fargate -ominaisuuden avulla ECS -palvelun käyttäjät kykenevät ajamaan konttisovelluksia ECS -palvelussa ilman että heidän tulee luoda konteille pilvipalvelimia. Fargate allokoi automaattisesti konteille pilvipalvelimet käyttäjän puolesta, eikä käyttäjän tarvitse tämän takia huolehtia pilvipalvelimista ollenkaan. Mikäli Fargate -ominaisuus saadaan toteutettua EKS -palveluun, parantaisi tämä palvellun kilpailuasemaa kilpailijoihinsa nähden huomattavasti.

Toistaiseksi en näe EKS -palvelussa suurta etua ECS -palveluun nähden. Mikäli olemassa oleva K8s -projekti halutaan migratoida AWS ympäristöön on EKS -palvelu siihen loistava ratkaisu. En näe muita syitä aloittaa projektia hyödyntäen EKS -palvelua. Olen samaa mieltä Arundelin ja Dominguksen (2019) kanssa. Mikäli halutaan aloittaa uusi projekti hyödyntäen K8s -konttiorkestrointityökalua, suosittelen käyttämään GKE -palvelua. Heidän palvelussaan on toistaiseksi eniten ominaisuuksia. K8s -klusterin luonti on erittäin nopeaa ja helppoa kun käytetään GKE -palvelua.

5.1 EKS vertailu

Kappaleessa tuodaan esille mielestäni EKS palvelun suurimmat vahvuudet ja suurimmat heikkoudet.

5.1.1 EKS vahvuudet

- EKS antaa paljon hallintaa kätyrisolmuihin
    - Voidaan luoda monimutkaisia ympäristöjä
- Mestarit ovat HA
    - Kannustaa jakamaan kätyrit usealle saatavuusalueelle
- Toimii hyvin muiden AWS resurssien kanssa
- Klusterissa on mahdollista olla erilaisia kätyrisolmuja

5.1.2 EKS heikkoudet

- AWS käyttäjille enintään 3 klusteria / tili
    - Mahdollista pyytää lisää AWS supportin kautta
- Mestari maksaa ($0.20 / 1h)
    - Muu infrastruktuuri klusterissa maksaa myös
- ”master as a service”
    - Palvelu tarjoaa vain klusterille ohjaamon
    - Kätyrisolmut tulee käyttäjän luoda itse ja liittää klusteriin
    - Kätyreiden tietoturvapäivitykset ja skaalautuvuus on myös käyttäjän vastuulla

Lähteet

Acreman, S. s.a. Kubernetes Cloud Services. Luettavissa: https://kube-dex.com/google-gke-vs-microsoft-aks-vs-amazon-eks/. Luettu 19.3.2019

Ali, S. 2018. Kubernetes Design and Development Explained. Luettavissa: https://the-newstack.io/Kubernetes-design-and-development-explained/. Luettu: 18.2.2019

Amazon Web Services, 2018a. Amazon EKS – Now Generally Available. Luettavissa: https://aws.amazon.com/blogs/aws/amazon-eks-now-generally-available/. Luettu 06.03.2019

Amazon Web Services, s.a (a). Amazon EKS Service IAM Role. Luettavissa: https://docs.aws.amazon.com/eks/latest/userguide/service_IAM_role.html. Luettu 13.03.2019

Amazon Web Services, s.a (b). Cluster VPC Considerations. Luettavissa: https://docs.aws.amazon.com/eks/latest/userguide/network_reqs.html. Luettu 18.03.2019

Amazon Web Services, s.a (c). AWS Resource and Property Types Reference. Luettavissa: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html. Luettu 22.04.2019

Amazon Web Services, s.a (d). AWS::EC2::VPC. Luettavissa: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-vpc.html. Luettu 29.04.2019

Amazon Web Services, s.a (e). Cluster Security Group Considerations. Luettavissa: https://docs.aws.amazon.com/eks/latest/userguide/sec-group-reqs.html. Luettu 01.05.2019

Amazon Web Services, s.a (f). Launching Amazon EKS Worker Nodes. Luettavissa: https://docs.aws.amazon.com/eks/latest/userguide/launch-workers.html. Luettu 02.05.0219

Arundel, J & Domingus, J. 2019. Cloud Native DevOps with Kubernetes. O'Reilly Media, Inc. Sebastopol

Baier, J. 2017. Getting Started with Kubernetes. Packt Publishing Ltd. Birmingham

Bala, E. 1.12.2017. Software Development Manager, Amazon Elastic Container Services for Kubernetes Deep Dive. Amazon Web Services. Seminaariesitys. Las Vegas. Nähtävissä: https://youtu.be/vrYLrx-a_Wg

DigitalOcean, 2018. Kubernetes Overview. Luettavissa: https://www.digitalocean.com/docs/Kubernetes/overview/. Luettu 20.3.2019

Docker, s.a. What is Container. Luettavissa: https://www.docker.com/resources/what-container. Luettu: 12.01.2019

Gupta, A. 2017. Why is Kubernetes so popular? Luettavissa: https://opensource.com/article/17/10/why-Kubernetes-so-popular. Luettu: 19.01.2019

Haider, H. 2019. DigitalOcean Managed Kubernetes: An Interview with the Experts. Luettavissa: https://www.replex.io/blog/digitalocean-managed-Kubernetes-an-interview-with-the-experts. Luettu 20.3.2019

Hakkarainen, P. 2018. Docker, Kubernetes, SUSE CaaS Platform – musiikkia kontilleni. Luettavissa: https://susesuomi.fi/ajankohtaista/uutiset/3/10/2018/docker-Kubernetes-suse-caas-platform-musiikkia-kontilleni/. Luettu: 12.01.2019

Hightower, K., Burns, B. & Beda, J. 2017. Kubernetes Up & Running. O'Reilly Media, Inc. Sebastopol

Jayanandana, N. 2018. Benefits of Kubernetes. Luettavissa: https://medium.com/platformer-blog/benefits-of-Kubernetes-e6d5de39bc48. Luettu: 22.2.2019

Kubernetes 2018a. What is Kubernetes? Luettavissa: https://Kubernetes.io/docs/concepts/overview/what-is-Kubernetes/. Luettu: 12.01.2019

Kubernetes, 2018b. Resizing a cluster. Luettavissa: https://Kubernetes.io/docs/tasks/administer-cluster/cluster-management/. Luettu 20.2.2019

Kubernetes, 2018c. Services. Luettavissa: https://Kubernetes.io/docs/concepts/services-networking/service/. Luettu 20.2.2019

Kubernetes, 2018d. Running Multiple Instances of Your App. Luettavissa: https://Kubernetes.io/docs/tutorials/Kubernetes-basics/scale/scale-intro/. Luettu 20.2.2019

Kubernetes, 2019a. Kubernetes Components. Luettavissa: https://Kubernetes.io/docs/concepts/overview/components/. Luettu: 9.2.2019

Kubernetes, 2019b. Options for Highly Available Topology. Luettavissa: https://Kubernetes.io/docs/setup/independent/ha-topology/#what-s-next. Luettu: 2.2.2019

Kubernetes, 2019c. Pod Lifecycle. Luettavissa: https://Kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/. Luettu: 20.2.2019

Kublr, 2017. Under the Hood: An Introduction to Kubernetes Acrhitecture. Luettavissa: https://kublr.com/blog/under-the-hood-an-introduction-to-Kubernetes-architecture/. Luettu: 9.2.2019

Lardinois, F. 2015. As Kubernetes Hits 1.0, Google Donates Technology To Newly Formed Cloud Native Computing Foundation. Luettavissa: https://techcrunch.com/2015/07/21/as-Kubernetes-hits-1-0-google-donates-technology-to-newly-formed-cloud-native-computing-foundation-with-ibm-intel-twitter-and-others/. Luettu: 19.01.2019

Microsoft Azure, 2019. Kubernetes core concepts for Azure Kubernetes Service (AKS). Luettavissa: https://docs.microsoft.com/en-us/azure/aks/concepts-clusters-workloads#cluster-master. Luettu 20.3.2019

Nadareishvili, I., Mitra, R., McLarty, M. & Amundsen, M. 2016. Microservice Architecture. O'Reilly Media, Inc. Sebastopol

Openstack, 2018. How to run a Kubernetes cluster in OpenStack. Luettavissa: https://superuser.openstack.org/articles/how-to-run-a-Kubernetes-cluster-in-openstack/. Luettu 19.3.2019

Paraiso, F., Challita, S., Al-Dhuraibi, Y. & Merle P. 2016. Model-Driven Management of Docker Containers. University of Lille & inria Lille. France

Ravindra, S. 2018. Kubernetes vs. Docker Swarm: What's the Difference? Luettavissa: https://thenewstack.io/Kubernetes-vs-docker-swarm-whats-the-difference/. Luettu 21.3.2019

Rize, L. 2017. Container Image Immutability and the Power of Metadata. Luettavissa: https://blog.codeship.com/container-image-immutability-power-metadata/. Luettu: 16.2.2019

Sandeep, D. 2018. Kubernetes NodePort vs LoadBalancer vs Ingress? When should I use what? Luettavissa: https://medium.com/google-cloud/Kubernetes-nodeport-vs-loadbalancer-vs-ingress-when-should-i-use-what-922f010849e0. Luettu 21.3.2019

Sanders, S. 2018. How Kubernetes improves IT's operational efficency. Luettavissa: https://jaxenter.com/Kubernetes-improves-efficiency-147699.html. Luettu: 23.2.2019
Sarkar, A & Shah, A. 2018. Learning AWS second Edition. Packt Publishing Ltd. Birmingham

Sayfan, G. 2017. Mastering Kubernetes. Packt Publishing Ltd. Birmingham
Selvan, T. 2018. GKE vs AKS vs EKS. Luettavissa: https://blog.hasura.io/gke-vs-aks-vs-eks-411f080640dc/. Luettu 20.3.2019

Yegulalp, S. 2019. What is Kubernetes? Container orchestration explained. Luettavissa: https://www.infoworld.com/article/3268073/Kubernetes/what-is-Kubernetes-container-orchestration-explained.html. Luettu: 31.01.2019

Liitteet

Liite 1. GKE, AKS, EKS vertailu taulu (Acreman, S. s.a.)

	Google GKE	Microsoft AKS	Amazon EKS
Year started	2014	2017	2018
Kubernetes Versions	1.10 (1.11 for whitelisted users)	1.11	1.10
Regions Supported	Worldwide	Almost Worldwide	North America and Ireland
Managed Control Plane	Yes	Yes	Yes
Control Plane HA	Multi AZ	No	Multi AZ
Cluster Create Time	3 minutes	20 mins	20 mins
Dynamic Admission Control	Yes	Yes	Yes
Multiple Node Pools	Yes	No	Yes
Managed Worker Nodes	Yes	Yes	No
Worker Node HA	Multi AZ	Multi AZ	Multi AZ
Worker Node GPU Support	Yes	Yes	Yes
Maximum pods per node	100	110	Limited by ENI
Maximum nodes per cluster	5000	100	500+
New worker start time	< 3 mins	< 15 mins	< 5 mins
Bare metal nodes	No	No	Yes
Kubernetes Upgrades	Automatic or On Demand	On Demand	No
Control Plane Costs	Free	Free	20 cents per hour per master
Control Plane Integrated Logging	Yes	Yes	No
Compliance	PCI DSS, ISO, SOC, HIPAA	PCI DSS, ISO, SOC, HIPAA	HIPAA
Auto Repair	Yes	No	No
Cross region load balancing	Yes	No	
Cross region networking	Globally flat network	Requires VPN	Requires VPN
Network Policy Support	Yes (Calico)	Yes (Kube Router)	Yes (Calico)
Aggregated API Support	Yes	Yes	Yes
Service Broker	Yes	Yes	Yes
Container as a Service integration	No	Virtual Kubelet with ACI	No
Dev UX	Good	Bad	OK