jamk.fi

Kubernetes-klusterin automatisoitu käyttöönotto AWS-pilvipalvelussa

Toni Ahola

Opinnäytetyö
Joulukuu 2020
Tieto- ja viestintätekniikka
Insinööri (AMK), tieto- ja viestintätekniikan tutkinto-ohjelma
Tietoverkkotekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Ahola Toni	Opinnäytetyö, AMK	Joulukuu 2020
	Sivumäärä 89	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Kubernetes-klusterin automatisoitu käyttöönotto AWS-pilvipalvelussa

Tutkinto-ohjelma

Tieto- ja viestintätekniikka

Työn ohjaaja(t)

Ari Rantala, Esa Salmikangas

Toimeksiantaja(t)

Valtion tieto- ja viestintätekniikkakeskus Valtori

Tiivistelmä

Toimeksiantajana toimi Valtion tieto- ja viestintätekniikkakeskus Valtori, joka tuottaa toimialariippumattomat ICT-palvelut valtionhallinnolle. Tavoitteena oli luoda Infrastructure as a Code -työkaluin automaattisesti pystytettävä, skaalautuva korkean saatavuuden konttiorkestrointijärjestelmä, pilvipalveluun. Teknologiavertailuiden jälkeen päätettiin pystyttää hallittu EKS Kubernetes-klusteri AWS-pilvipalveluun. Infrastruktuuri koodattiin käyttäen Terraform Infrastructure as a Code -työkalua.

Klusteriin vaadittavia komponentteja varten tehtiin Terraform -moduulit, jotka sisältävät resurssikokonaisuuksia ja ovat käytettävissä moniin ympäristöihin. Klusteria varten luotiin virtuaaliverkko, jonka aliverkot sijaitsevat kolmella eri saatavuusvyöhykkeellä. Kubernetes-klusterin hallintataso luodaan monennetusti kolmelle eri saatavuusvyöhykkeelle. Klusterin työläisnoodit luodaan autoskaalausryhmään, joka skaalaa instanssien määrää kolmella eri saatavuusvyöhykkeellä. Klusteriin sisällytetään kontrollerit, jotka mahdollistavat klusterin autoskaaluksen, DNS-tietuiden luonnin, monitoroinnin, sekä sovellusten paljastamisen klusterin ulkopuolelle.

Klusterin toimivuutta, skaalautuvuutta ja saatavuutta testattiin luomalla testisovellus, joka paljastettiin klusterin ulkopuolelle. Sovellusta kuormitettiin ja tutkittiin klusterin skaalautuvuutta. Klusterin saatavuutta testattiin poistamalla työläisnoodeja kuormitetusta klusterista.

Avainsanat (asiasanat)

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Ahola Toni	Bachelor's thesis	December 2020
		Language of publication: Finnish
	Number of pages: 89	Permission for web publication: x

Title of publication

Automated deployment of the Kubernetes cluster in the AWS cloud service

Degree programme

Information and Communication Technology

Supervisor(s)

Ari Rantala, Esa Salmikangas

Assigned by

Government ICT-Center Valtori

Abstract

The client was Valtori, the State Information and Communication Technology Center, which provides industry-independent ICT services to the state administration. The goal was to create an automatically scalable, high-availability container orchestration system using Infrastructure as a Code tools for the cloud service. After the technology comparisons, it was decided to set up a managed EKS Kubernetes cluster in the AWS cloud service. The infrastructure was coded using the Terraform Infrastructure as a Code tool.

Terraform modules were created for the components required for the cluster, which contain resource collections and are available for multiple environments. A virtual network was created for the cluster, with subnets located in three different availability zones. The management level of the Kubernetes cluster is multiplied for three different access zones. Cluster worker nodes are created in an autoscaling group that scales the number of nodes in across three different availability zones. The cluster includes controllers that enable cluster autoscaling, DNS record creation, monitoring, and application discovery outside the cluster.

The functionality, scalability, and availability of the cluster were tested by creating a test application that was revealed outside the cluster. The application was load tested and the scalability of the cluster was analyzed. Cluster availability was tested by removing worker nodes from the loaded cluster.

Keywords/tags (subjects)

Miscellaneous

Sisältö

Kuviot

Taulukot

Lyhenteet

ALB	Aplication Load Balancer
API	Application Programming Interface
ASG	Autoscaling Group
AWS	Amazon Web Services
CI/CD	Continuous Integration / Continuous Delivery
CNCF	Cloud Native Computing Foundation
HPA	Horizontal Pod Autoscaler
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
DNS	Domain Name System
DHCP	Dynamic Host Configuration Protocol
EC2	Elastic Compute Cloud
ECR	Elastic Container Registry
EBS	Elastic Block Storage
EKS	Elastic Kubernetes Service
IaaS	Infrastructure as a Service
IAM	Identity and Access Management
IaC	Infrastructure as a Code
IP	Internet Protocol
NAT	Network Address Translation
OIDC	Open ID Connect
PaaS	Platform as a Service
PV	Persistent Volume

PVC	Persistent Volume Claim
SaaS	Software as a Service
TLS	Transport Layer Security
VPC	Virtual Private Cloud
VPA	Vertical Pod Autoscaler

1 Johdanto

1.1 Valtion tieto- ja viestintäkeskus Valtori

Opinnäytetyön toimeksiantaja on Valtion tieto- ja viestintätekniikkakeskus Valtori, joka tuottaa toimialariippumattomat ICT-palvelut valtionhallinnolle. Opinnäytetyön toimeksianto syntyi tarpeesta vastata Valtorin asiakasorganisaatioiden haluun hyödyntää pilvipalveluja konttipohjaisten sovellusten ajoalustana.

Valtion tieto- ja viestintätekniikkakeskus Valtori on valtiovarainministeriön hallinnonalalla toimiva virasto, joka perustettiin vuonna 2014 nojaten lakiin valtion yhteisten tieto- ja viestintäteknisten palvelujen järjestämisestä. Valtori tuottaa toimialariippumattomat ICT-palvelut kaikille valtionhallinnon organisaatioille. (Laki valtion yhteisten tieto- ja viestintäteknisten palvelujen järjestämisestä 1226/2013)

Toimialariippumattomat ICT-palvelut ovat palveluita, joiden tuottaminen tai järjesteleminen ei vaadi toimikohtaista osaamista ja ne perustuvat yleisesti käytettyihin laite- ja ohjelmistoteknologioihin. Valtori tuottaa asiakasorganisaatioilleen mm. viestintätekniset palvelut, päätelaitepalvelut, kapasiteetti- ja konesalipalvelut, sekä tietoliikennepalvelut. (Valtorin tuottamat palvelut). Valtori tuottaa asiakkailleen kapasiteettipalveluita konesaleista.

Valtori tarjoaa pilvipalvelukapasiteettia asiakasorganisaatioilleen Amazon AWS ja Microsoft Azure pilvipalvelualustoilta yhdessä kumppaniorganisaatioidensa kanssa.

Valtorilla on toimipisteitä yli 30 paikkakunnalla ympäri maata ja Valtorissa työskentelee noin 1100 henkilöä. Valtorin suurin toimipiste sijaitsee Helsingissä ja sen päätoimipaikka on Jyväskylässä.

1.2 Toimeksiannon taustat

1.2.1 Asiakasorganisaatioiden tavoitteet

Valtorin asiakasorganisaatiot tavoittelevat sovelluksilleen korkeaa saatavuutta, pienempiä kustannuksia ja nopeampia käyttöönottoja. Perinteisissä konesaliympäristöistä edellä mainittuja tavoitteita on hankala saavuttaa, joten asiakasorganisaatioiden kiinnostus pilvipalveluiden käyttöönottoon on kasvanut.

1.2.2 Konesalista tarjotun palvelun ongelmat

Suurin osa Valtorin asiakasorganisaatioiden sovelluksista tarjotaan konesaliympäristöistä. Konesalista tarjotun kapasiteetin ongelmana on, että se ei ole dynaamisesti skaalautuvaa, jolloin suurimman osan ajasta palvelut toimivat ylimitoitetulla resurssilla, jolloin maksetaan mahdollisesti moninkertaisia summia kapasiteetista sovelluksen todellisen kapasiteettitarpeeseen nähden.

Myös konesaliympäristön muutos- ja ylläpitotyöt ovat kalliita ja aikaa vieviä. Levytilan kasvattaminen, käyttöjärjestelmäpäivitykset ja verkkokonfiguraatioiden ja muut konesaliympäristön ylläpitotehtävät vaativat todella paljon työtunteja. Sovellus saattaa sijaita monimutkaisten verkkosegmenttien takana, jolloin palomuuriavausten tekeminen tiketin luomisesta alkaen saattaa kestää yli viikon. Pelkkä uusien ympäristöjen infrastruktuurin luominen vaatii paljon aikaa ja monta palvelupyyntöä käyttöpalvelutarjoajalle.

1.2.3 Pilvipalvelusta ratkaisu

Pilvipalvelut mahdollistavat sovelluksen vaatiman kapasiteetin dynaamisen skaalaamisen, jolloin resurssien määrää tai kokoa voidaan kasvattaa tai pienentää tarpeen mukaan. Pilvipalveluissa resurssien määrä voidaan asettaa skaalautumaan myös automaattisesti, jolloin esimerkiksi virtuaalikoneiden määriä voidaan säädellä niiden kuormitusasteen mukaan.

Pilvipalvelussa fyysiset verkkolaitteet ja fyysiset palvelimet eivät aiheuta ylläpitokustannuksia, koska pilvipalveluntarjoaja vastaa niistä ja asiakkaalle näkyvät vain virtuaaliverkot ja virtuaalikoneet. Pilvipalvelussa resurssit ovat otettavissa käyttöön heti,

jolloin tietokantojen, virtuaalikoneiden ja muiden resurssien luomiseen ei tarvita yhteydenottoa käyttöpalveluntarjoajalle ja viikkojen odottelua. Pilvipalveluun voidaan luoda monimutkaisia ja isoja kokonaisuuksia minuuteissa, käyttämällä Infrastructure as Code -työkaluja (IaC), joka luo deklaratiivisessa koodissa määritellyt resurssit käyttäen pilvipalveluntarjoajan ohjelmointirajapintoja.

1.3 Toimeksianto

Opinnäytetyön tavoitteena on suunnitella ja toteuttaa automaattisesti pilvipalveluun pystytettävä ja skaalautuva konttiorkestrointiympäristö Valtion tieto- ja viestintätekniikkakeskus Valtorille. Ympäristöön tarvittavat pilviresurssit luodaan automatisoidusti Infrastructure as a Code -työkaluja käyttäen, jolloin se on helposti käyttöönotettavissa muihin tarkoituksiin.

Ympäristön pystytyksen tulee olla mahdollisimman pitkälle automatisoitua, jotta kontitettujen sovellusten käyttöönotosta tulee mahdollisimman helppoa. Konttiorkestrointiympäristössä toimivien sovellusten tulee olla mahdollista skaalautua käytön mukaan.

Ympäristöön pystytetään myös testisovellus, jonka avulla testataan ympäristön skaalautuvuutta ja saatavuutta.

2 Opinnäytetyön teemat

2.1 Pilvipalvelut

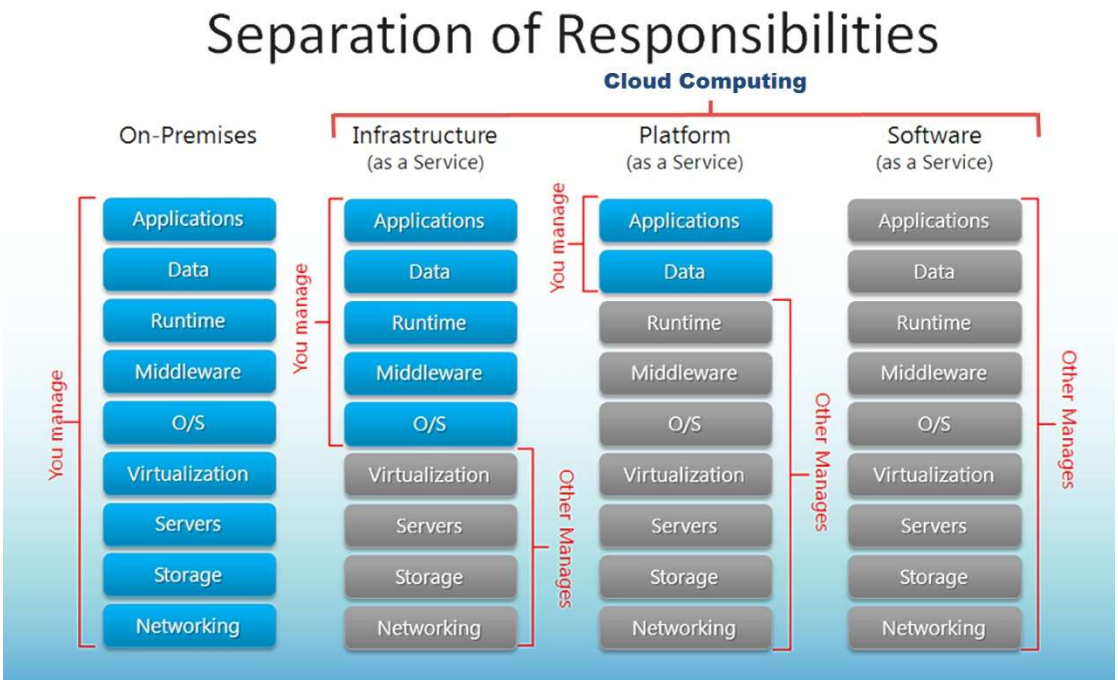
2.1.1 Mitä on pilvilaskenta

Pilvilaskenta (cloud computing) on tietoteknisten palveluiden hajauttamista ja ulkoistusta. Pilvilaskennassa asiakkaat eivät omista itse oman järjestelmänsä infrastruktuuria vaan maksavat käytön mukaan pilvipalveluntarjoajalle sovelluksista, prosessointitehosta, tallennustilasta verkkoliikenteestä. Infrastruktuuria ylläpidetään pilvipalveluntarjoajan palvelimilta ja se on saavutettavissa ja hallittavissa julkisen internetin kautta. Pilvilaskentaa hyödyntämällä vältytään infrastruktuurin hankkimisen etukä-

teismaksuilta, sekä huolto ja ylläpitotöiltä. Pilvipalveluita voidaan skaalata käyttötarpeen mukaan, jolloin sovellusten kustannusta ja suorituskykyä voidaan optimoida helposti. (Ranger 2018)

2.1.2 Pilvipalvelumallit

Pilvipalvelut voidaan jakaa yleensä kolmeen eri malliin. Infrastructure as a Service (IaaS), Platform as a Service (PaaS) ja Software as a Service (SaaS). Jokaisessa mallissa on erilaiset ylläpidolliset vastuut pilvipalveluntarjoajan ja asiakkaan välillä, joten on tärkeää ymmärtää mallit, jotta voidaan valita sopivin vaihtoehto. Julkisen pilven jaetun vastuun malli esitetty kuviossa 1.



Kuvio 1. Julkisen pilven jaetun vastuun malli

IaaS

IaaS tarjoaa virtualisoitua infrastruktuuria kuten virtuaalikoneet, virtuaaliverkot, käyttöjärjestelmät ja tallennustilan. Virtuaalikoneet ovat hallittavissa selaimen tai API:en kautta. IaaS tarjoaa perinteistä konesalia vastaavat teknologiat ja kyvykkyydet ilman fyysistä huolto- ja ylläpitotarvetta. IaaS on joustava pilvipalvelumalli, joka mahdollistaa infrastruktuurin käyttöönoton automaation. IaaS resurssit ovat skaalautuvia ja käytön mukaan laskutettavia. Resursseja voidaan ottaa käyttöön tai poistaa käytöstä tarpeen mukaan.

IaaS mallissa asiakkaat ovat ylläpidollisessa vastuussa mm. sovelluksista, ohjelmistoista, käyttöjärjestelmistä, middlewaresta ja sovellusten datasta. Pilvipalveluntarjoajan vastuulla on ylläpitää virtualisointikerros, fyysiset palvelimet, fyysiset verkot ja tallennustila.

PaaS

PaaS malli tarjoaa alustan kehittäjille luoda ja ajaa sovellustaan. Alusta on käytettävissä internetin kautta. Pilvipalveluntarjoaja huolehtii käyttöjärjestelmästä ja middlewaresta sallien kehittäjien keskittyä sovelluksen. Asiakkaan vastuulla on itse sovellus ja sen käyttämä data.

SaaS

SaaS on pilvipalvelun malli, jossa pilvipalvelun tarjoaja toimittaa kolmannen osapuolen hallitsemia ohjelmistoja asiakkaan käyttöön julkisen internetin yli. Suuri osa SaaS-palveluista toimii käyttäjän selaimessa. SaaS malli ei vaadi asiakkaalta mitään ylläpidollista työtä, vaan palvelu on täysin palveluntarjoajan ylläpidettävä. SaaS-palveluiden huonoja puolia voivat olla esim. kontrollin puute, huono mukautettavuus, vendor lock-in ja puutteelliset ominaisuudet. (Rasa & Watts 2019)

2.2 Infrastruktuuri koodina

Infrastructure as Code (IaC) on toimintamalli, jossa IT-infrastruktuuri otetaan käyttöön ohjelmallisesti määrittelemällä luotavat resurssit tiedostoihin, jolla IaC-työkalu tekee API-kutsuja esimerkiksi pilvipalvelualustalle. IaC mahdollistaa infrastruktuurin provisioimisen ja ylläpidon ilman manuualisia prosesseja. IaC:tä voidaan hyödyntää monella IT-alan osa-alueella, mutta se on erityisen hyödyllistä pilvipalveluiden ja IaaS-palveluiden käyttöönotossa ja DevOps-toimintamallin tukemisessa. DevOps vaatii ketteriä prosesseja ja automaatiota, jonka helposti käyttöönotettava IT-infrastruktuuri mahdollistaa. (Merron 2018)

Ympäristöjen manuaalinen luominen mahdollistaa useiden pienien, mutta mahdollisesti merkittävien konfiguraatiovirheiden tapahtumisen. IaC:n tapauksessa kun testiympäristö luodaan samasta koodista kuin kehitysympäristö, voidaan olla varmoja, että ympäristöt ovat identtisiä ja toimivat samalla tavalla.

Useat avoimen lähdekoodin IaC-työkalut, kuten Ansible ja Terraform mahdollistavat infrastruktuurin koodaamisen useille pilvipalvelualuistoille. Pilvipalveluntarjoajat ovat kehittäneet myös omia suljetun lähdekoodin IaC-työkaluja, jotka ovat alustariippuvaisia. Esimerkkeinä alustariippuvaisista IaC-työkaluista ovat Azure ARM ja AWS Cloudformation.

2.3 Konttiorkestrointi

Konttiorkestroinnilla automatisoidaan konttien käyttöönotto, ylläpito ja verkkokonfiguraatiot. Konttiorkestrointia voidaan käyttää kaikissa ympäristöissä missä on käytössä kontteja. Konttiorkestrointi helpottaa saman sovelluksen käyttöönottoa useissa ympäristöissä. Konttien hallitseminen konttiorkestroinnilla tukee DevOps-mallia sillä se on helposti liitettävissä CI/CD-putkeen (Continuous Integration / Continuous Delivery). Konttiorkestroinnilla voidaan automatisoida ja hallita useita tehtäviä kuten:

- Konttien käyttöönotto
- Konttien konfiguraationhallinta
- Resurssien kuten muistin, prosessorin ja tallennustilan allokoiminen
- Konttien replikointi ja skaalaus
- Kuormantasaus ja reititys
- Konttien terveyden monitorointi

Nykyään on saatavilla monia konttiorkestrointityökaluja kuten Apache Mesos, Docker Swarm ja Kubernetes.

3 Käytetyt teknologiat

3.1 Teknologiavalinnat

Toimeksianto oli luoda automatisoidusti pilvipalvelualustalle pystytettävä uudelleenkäytettävä konttiorkestrointijärjestelmä. Vertaillaan pilvialusta-, IaC- ja konttiorkestrointiteknologioita, joista valitaan soveltuvimmat toimeksiannon vaatimusten täyttämiseksi.

3.1.1 Pilvipalvelualusta

Valtori tarjoaa asiakkailleen käytettäväksi Amazon Web Services (AWS)- ja Microsoft Azure -pilvipalvelualustoita. Molemmat alustat tarjoavat konttiorkestrointiin samankaltaisia PaaS-ratkaisuja. AWS tarjoaa Elastic Container Service ja Elastic Kubernetes Service -konttiorkestrointipalveluita, joiden hallintataso on AWS:n hallitsema ja ylläpitämä. Azure tarjoaa vastaavina palveluina Azure Kubernetes Serviceä (AKS) ja Openshift Container Platformia. Serverless konttialustoina AWS tarjoaa Fargatea ja Azure Container Instancea.

Molemmilla AWS:llä ja Azurella on useita konesaleja EU/ETA-alueella. IaC-tyokaluksi AWS tarjoaa Cloudformationia ja Resource Manageria. Molemmille on myös tarjolla kolmannen osapuolen IaC-työkaluja, kuten Terraform Chef, Ansible ja Puppet.

Taulukko 1. Pilvialustojen vertailu

	AWS	**Azure**
Konttiorkestrointi	EKS, ECS ,Fargate	AKS, ACI, OpenShift
IaC	Cloudformation, kolmannet osapuolet	ARM, kolmannet osapuolet
Konesali EU/ETA-alueella	Kyllä	Kyllä
Markkinaosuus	45,0% (Public Cloud Market report 2020)	17,9% (Public Cloud Market report 2020)
Julkaistu	2006	2010
Virtuaalikoneen hinta (yleiskäyttöinen 2CPU, 8GB muistia, purskekyvykäs)	t3.large $0.1512/h (RHEL) (4.10.2020) eu-west-1	B2MS $0.156/h (RHEL) (4.10.2020) West Europe

Pilvipalvelualustaksi valikoitui AWS, koska se on pilvipalvelualustoista vanhin ja käytetyin, jolle on eniten saatavilla alustaa tuntevia kehittäjiä ja asiantuntijoita. Euroopassa AWS-tarjoaa myös hieman halvemmat virtuaalikoneiden tuntihinnat.

3.1.2 Konttiorkestrointi

AWS tarjoaa konttiorkestrointiin ECS- ja EKS-konttiorkestrointijärjestelmiä. Kolmas vaihtoehto on ylläpitää omaa konttiorkestrointijärjestelmän hallintatasoa virtuaalikoneissa.

ECS- ja EKS-palvelut tarjoavat AWS:n ylläpitämän korkeasti saatavilla olevan hallintatason. Jos haluttaisiin ylläpitää Kubernetes-hallintatasoa itse, tulisi virtuaalikoneille asentaa hallintatason komponentit itse. Myös hallintatason virtuaalikoneiden käyttöjärjestelmäpäivitykset ja Kubernetes-komponenttien versiopäivitykset olisivat käyttäjän ylläpitovastuulla. AWS-tarjoaa valmiit virtuaalikonekuvat EKS- ja ECS-palvelun työläisnoodeille, joissa kontteja ajetaan. Käyttäjät ovat silti vastuussa siitä, että käytössä on ajantasaiset virtuaalikonekuvat. Itse ylläpidetyssä klusterissa virtuaalikoneille tulee asentaa työläisnoodien vaatimat komponentit itse ja huolehtia käyttöjärjestelmän ja ohjelmistojen ajantasaisuudesta. ECS:n hallintaso on ilmainen, kun taas EKS:n hallintataso maksaa 0.10$ tunnilta. Itse ylläpidetyn Kubernetes-klusterin hallintataso maksaa virtuaalikoneiden tuntihinnan ja ylläpitotyöt.

EKS:n ja itse yllä pidetyn Kubernetesin käyttöönotetut sovellukset ovat siirrettävissä muilla pilvipalvelualustoilla tai konesaleissa toimiviin Kubernetes-klustereihin. ECS on AWS:lle spesifi palvelu ja näin ollen sen päälle käyttöönotetut sovellukset toimivat suoraan vain AWS:ssä. ECS on helpoin käyttöönotettava vaihtoehdoista, koska se sisältää valmiina useita integraatiota AWS-palveluihin, joita muissa ratkaisuissa joudutaan ottamaan ja konfiguroimaan erikseen käyttöön. Kubernetes vaihtoehdoissa on mahdollista isoloida erillisiä ympäristöjä klusterin sisällä eri nimiavaruuksiin, mikä ECS:ssä ei ole mahdolista. Taulukossa 2 on vertailtu konttiorkestrointijärjestelmien ominaisuuksia.

Taulukko 2. Konttiorkestrointijärjestelmien vertailu

	ECS	**EKS**	**Itse ylläpidetty Kubernetes**
AWS:n ylläpitämä hallintataso	Kyllä	Kyllä	Ei
Valmiit virtuaalikonekuvat työläisnoodeille	Kyllä	Kyllä	Ei
Sisäänrakennettu monitorointi	Kyllä	Kyllä	Ei
Hallintatason hinnoittelu	Ilmainen	0.10$/h	Virtuaalikoneiden hinta + ylläpitokulut
Tuetut alustat	Vain AWS	Alustariippumaton	Alustariippumaton
Integraatiot muihin AWS palveluihin	Valmiit Lokien siirto, kuormantasaajien hallinta, Domain nimien hallinta	Ei, mutta otettavissa käyttöön AWS:n ja avoimen lähdekoodin konteilla	Ei, mutta otettavissa käyttöön AWS:n ja avoimen lähdekoodin konteilla
Nimiavaruudet	Ei	Kyllä	Kyllä

Kontteja / virtuaalikone	120	750 Podia (jotka voivat sisältää useita kontteja)	750 Podia (jotka voivat sisältää useita kontteja)

Itse ylläpidetty ja asennettu Kubernetes-klusterin hallintataso ei valikoitunut ratkaisuksi, koska hallintatason vaatimat virtuaalikoneet ja niiden ylläpidon aiheuttamat kustannukset ja työmäärä ovat paljon suurempia muihin vaihtoehtoihin nähden. Lisäksi klusterin hallintason konfigurointi ja sen päivittäminen monimutkaistaa klusterin pystytyksen automatisointia. ECS:n etu on sen ilmainen hallintataso, mutta EKS:n 72 dollarin kuukausihinta on pieni kun otetaan huomioon miten suuria kuluja klusterissa ajettavat sovellukset aiheuttavat. ECS on helpompi pystyttää ja operoida yksinkertaisuutensa ja integraatioidensa ansiosta. EKS on kuitenkin monikäyttöisempi ja sen kanssa ei törmätä rajoituksiin yhtä helposti sovellusten kasvaessa ja monimutkaistuessa. Käyttöön voidaan ottaa avoimen lähdekoodin tai kaupallisia Kubernetesille spesifejä kontrollereita tai työkaluja, jotka eivät ole ECS:n kanssa yhteensopivia. Lisäksi Kubernetes on alustariippumaton avoimen lähdekoodin ratkaisu, jonka päällä voidaan ajaa sovelluksia muilla pilvipalvelualustoilla ja perinteisissä konesaleissa. Konttiorkestrointialustaksi valikoitui EKS.

3.1.3 Infrastructure as a Code -työkalu

Toimeksiannon vaatimusten mukaisesti infrastruktuuri luodaan pilvipalveluun koodina, joten vertaillaan Infrastructure as a Code -työkaluja, jotta valitaan parhaiten kriteerit täyttävä työkalu.

Vaihtoehtoina on AWS:n Cloudformation ja avoimen lähdekoodin vaihtoehdot Terraform ja Ansible. Kaikki vaihtoehdot luovat infrastruktuuria idempotentisti, joten vain lisätyt muutokset infrastruktuurikoodiin suoritetaan. Cloudformation on AWS:n infrastruktuuriin integroitu palvelu, jolle infrastruktuurikoodi annetaan YAML- tai JSON-muodossa. Ansible ja Terraform vaativat ohjelman asennusta työasemalle tai CI-järjestelmään. Ansiblen infrastruktuurikoodi on YAML- tai Jinja2-muotoista, kun taas Terraform käyttää omaa HCL-kieltään. Terraformiin ja Ansibleen on saatavilla useita lisäosia ja moduuleja, joilla voidaan luoda Infrastruktuuria muille pilvialustoille, hallita virtuaalikoneiden konfiguraatiota kuten Kubernetes-resursseja. Ansible on proseduraalinen työkalu, jolle määritellään askeleet millä saavutetaan haluttu tila.

Cloudformation ja Terraform ovat deklaratiivisia työkaluja, jolle määritellään jokin haluttu tila ja työkalu itse keksii tavan toteuttaa se.

Taulukko 3. IaC-työkalujen vertailu

	Ansible	**Cloudformation**	**Terraform**
Alustat	Alustariippumaton	AWS	Alustariippumaton
Mistä ajetaan	Työasema / CI	Integroitu AWS:n	Työasema / CI
Indempotenttius	Indempotentti	Indempotentti	Indempotentti
Konfiguraatiokieli	YAML/Jinja2	YAML/JSON	HCL
Deklaratiivinen / Proseduraalinen	Proseduraalinen	Deklaratiivinen	Deklaratiivinen
Moduulit, Pluginit , providerit	Useita	Ei	Useita

Ansible ei valikoitunut IaC-työkaluksi, koska proseduraalisella koodilla ei voi täysin kuvata infrastruktuurin tilaa. Proseduraalisessa koodissa askelten järjestyksellä on merkitystä ja koodi saattaa kasvaa monimutkaiseksi ajan kuluessa. Myös proseduraalisen koodin uudelleenkäytetävyys on rajoitettua, koska infrastruktuurin tila vaihtelee jatkuvasti eivätkä vanhat stepit enää päde infrastruktuurin uuteen tilaan.

Cloudformationin ja Terraformin välillä IaC-työkaluksi valikoitui Terraform. Terraformin etuina ovat useat lisäosat, moduulit ja mahdollisuus ajaa esimerkiksi shell-skriptejä infrastruktuurikoodin yhteydessä. Lisäksi Terraform-koodi on uudelleenkäytettävämpää, koska se mahdollistaa moduulien luomisen suuremmista resurssikokonaisuuksista, mikä ei ole mahdollista Cloudformationissa.

3.2 Amazon Web Services

Amazon Web Services eli AWS on Amazonin vuonna 2006 perustama pilvipalvelualusta, joka tarjoaa asiakkailleen käytön mukaan laskutettavaa pilvikapasiteettia. AWS on pilvipalvelualustojen markkinajohtaja ja sen suurimmat kilpailijat ovat Microsoft Azure ja Google Cloud Platform. AWS:ssä on käytettävissä yli 160 IaaS-, SaaS- ja PaaS-palvelua sisältäen palvelimet, tietokannat, verkot, kehitystyökalut ja Internet of Things -työkalut.

3.2.1 AWS:n globaali infrastruktuuri

Suurin osa AWS:n palveluista on sidottu johonkin maantieteelliseen alueeseen. AWS:n palveluja ylläpidetään useissa eri regionissa maailmanlaajuisesti. Regionit ovat erillisiä maantieteellisiä alueita, jotka muodostuvat useista eristetyistä vyöhykkeistä, joita kutsutaan saatavuusvyöhykkeiksi (engl. Availability Zone). Jokainen region on täysin eristetty muista regioneista, jolloin AWS:n infrastruktuuri on vikasietoinen ja stabiili. Saatavuusvyöhykkeet ovat yhden tai useamman konesalin eristetty kokonaisuus, jotka mahdollistavat korkeasti saatavien sovelluksien luomisen yhdelle regionille. Monentamalla resurssit usealle saatavuusvyöhykkeelle, voidaan varmistaa palveluiden saatavuus yhden saatavuusvyöhykkeen häiriöityessä. (Using regions and availability zones, N.d)

AWS:llä on 18 regionia ympäri maailmaa, joista viisi Euroopassa. Suomea lähin region sijaitsee Tukholmassa ja muut Euroopan regionit sijaitsevat Frankfurtissa, Dublinissa, Lontoossa ja Pariisissa. Kaikilla regioneilla ei ole saatavilla samoja palveluja ja Euroopan regioneista ensimmäisenä uudet ominaisuudet saa EU-West-1 (Irlanti), koska se on Euroopan regioneista ensimmäinen. (Using regions and availability zones, N.d)

3.2.2 Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud eli EC2 on AWS:n palvelu, joka tarjoaa skaalautuvaa virtuaalikonekapasiteettia. Virtuaalikoneista eli instansseista laskutetaan käytön mukaan, joten etukäteishankintoja ei tarvitse tehdä. Instanssi luodaan Amazon Machine Imagesta (AMI), joka sisältää käyttöjärjestelmän ja mahdolliset lisäohjelmistot. Instanssien koot ja ominaisuudet vaihtelevat instanssityypin mukaan. Instanssityyppejä on saatavilla kymmenillä erilaisilla prosessori-, muisti-, tallennustila- ja verkkokonfiguraatioilla. Hallintayhteydet instansseihin toteutetaan SSH-yhteydellä, jolloin AWS säilöö julkisen avaimen ja käyttäjä privaatin avaimen. EC2-instanssien tallennustilana voidaan käyttää väliaikaisia Instance Store -voluumeja, jotka säilövät dataa vain instanssin ollessa käynnissä, tai persistenttejä Elastic Block Store (EBS) -voluumeja. EC2-instansseihin on liitettynä Security Group, johon voi luoda palomuurisääntöjä IP-osoitteiden, porttien ja protokollien perusteella. EC2-resurssit, kuten EC2–instanssit ja EBS–voluumit voidaan sijoittaa useille saatavuusvyöhykkeille ja regioneille saatavuuden varmistamiseksi. (EC2 User Guide, N.d)

3.2.3 Amazon Virtual Private Cloud

Amazon Virtual Private Cloud (VPC) on virtuaaliverkko, joka toimii EC2 resurssien verkkoalustana. VPC muistuttaa perinteistä konesaliverkkoa, mutta hyödyntää AWS:n skaalautuvaa infrastruktuuria.

VPC:hen määritellään yksityinen IP-osoiteavaruus, joka voidaan lohkoa useiksi aliverkoiksi. VPC tukee myös IPv6-osoiteavaruuksia. Aliverkkoja voidaan reitittää Internet Gatewayn kautta julkiseen internettiin, jolloin resurssit saavat julkisen IP-osoitteen. Aliverkkoja voidaan reitittää internettiin myös NAT Gatewayn (Network Address Translation) kautta, jolloin resurssit eivät ole saavutettavissa internetistä. VPC:n sisäiset aliverkot voidaan sijoittaa eri saatavuusvyöhykkeille. VPC:itä voidaan yhdistää reitittämällä toimimaan yhtenäisenä verkkona. VPC:t voidaan reitittää yhteen käyttämällä VPN-ratkaisua tai AWS Transit Gatewayta. Myös oman konesalin verkon reitittäminen VPC:hen on mahdollista useilla AWS:n tarjoamilla ratkaisuilla. (VPC User Guide, N.d)

3.2.4 AWS Elastic Load Balancing

Elastic Load Balancing jakaa tulevaa sovellus tai verkkoliikennettä useiden kohteiden kuten EC2 Instanssien, konttien ja IP-osoitteiden välillä. Kuormantasaajat voidaan laittaa tasaamaan kuormaa eri saatavuusvyöhykkeillä olevien resurssien kesken, jolloin parannetaan sovelluksen saatavuutta.

Application Load Balancer (ALB) toimii asiakkaiden yhteyspisteenä. ALB reitittää liikennettä sovelluskerroksella (HTTP/HTTPS). Listener kuuntelee asiakkailta tulevia pyyntöjä käyttäen konfiguroituja protokollia ja portteja. Listenerille määritetyt säännöt määrittävät miten pyynnöt reititetään kohteille. Target Groupit reitittävät pyyntöjä kohderesursseille kuten EC2–instansseille perustuen konfiguroituun porttin ja protokollaan. Target Groupeille voidaan konfiguroida health checkit, jotka monitoroivat rekisteröityjen kohteiden saatavuutta, jolloin ALB ohjaa pyyntöjä vain terveille kohteille. (Application Load Balancing User Guide N.d)

Network Load Balancer käyttää ALB:n tapaan listenerejä ja target grouppeja, mutta se reitittää liikennettä OSI-mallin siirtokerroksella. TCP- ja UDP-liikenteelle NLB valitsee kohteen perustuen lähde IP-osoitteeseen, lähdeporttiin ja kohde IP-osoitteeseen ja kohdeporttiin. (Network Load Balancing User Guide, N.d)

3.2.5 AWS Autoscaling

AWS EC2-instanssit skaalataan autoskaalausryhmillä (engl. Auto Scaling Group). Autoskaalausryhmä on ryhmä EC2–instansseja, joita kohdellaan loogisena ryhmänä automaattisen skaalauksen ja sen hallinnan tarkoituksiin. Autoskaalausryhmän koko riippuu konfiguraatioon määritetystä ihannemäärästä. Autoskaalausryhmässä on aluksi ihannemäärän verran EC2–instansseja ja se lähettää health checkejä instansseille. Mikäli autoskaalausryhmässä ei ole ihannemäärän verran terveitä instansseja, niin ASG luo tai tuhoaa uusia instansseja ihannemäärän säilyttämiseksi. Autoskaalausryhmään voidaan määritellä käytäntöjä, joilla ryhmän kokoa pienennetään tai suurennetaan ehtojen mukaisiksi. Ihannemäärän lisäksi ryhmälle määritetään minimi- ja maksimiarvo, joiden välillä ryhmän suuruuttaa skaalataan. (AWS Autoscaling Group User Guide, N.d)

Autoskaalausryhmän kokoa voidaan skaalata myös manuaalisesti. Autoskaalausryhmän skaalaustapahtumat voidaan ajoittaa tietylle ajanhetkelle mikä voi hyödyllistä tapauksissa, joissa tiedetään kuorman kasvavan tiettyinä ajanhetkinä. Skaalaus voidaan toteuttaa myös perustuen instansseista saataviin metriikoihin. Esimerkiksi jos instanssien muistinkäyttö tai prosessorin kuormitus kasvaa yli tietyn pisteen niin autoskaalausryhmään lisätään instansseja. (Scaling the Size of your Auto Scaling Group, N.d)

3.2.6 AWS Route53

Route 53 on AWS:n korkeasti saatavissa oleva Domain Name System (DNS) -palvelu. Se on ainoa AWS:n palvelu, jolle on luvattu 100% palvelutaso. Route 53:n kolme päätoiminnallisuutta ovat domainien rekisteröinti, DNS –reititys ja health checkit. (Route 53 Developer Guide. N.d)

Kun verkkotunnus rekisteröidään Route 53:ssa, Route 53:sta tulee verkkotunnuksen DNS-palvelu. Route 53 luo Hosted Zonen ja osoittaa Hosted Zonelle neljä nimipalvelinta (How Domain Registration Works, N.d). DNS-reititys tehdään luomalla tietueita Hosted Zoneen. Route 53:n tukemat tietuetyypit ovat A, AAAA, CAA, CNAME, MX, NAPTR, NS, PTR, SOA, SPF, SRV ja TXT (Resource Record Types, N.d.).

Route 53 Health Checkit monitoroivat resurssien kuten verkkopalvelimien terveyttä. Health Check konfiguroidaan monitoroimaan kohteen IP–osoitetta tai verkkotunnusta. Health Check käyttää joko HTTP-, HTTPS- tai TCP-pyyntöjä monitoroinnin toteuttamiseen. Health Checkille määritellään intervalli, jolla kutsuja lähetetään ja raja-arvo montako epäonnistunutta kutsua tarvitaan, että palvelua voidaan pitää epäterveenä. Lisäksi Route 53 voidaan liittää Cloudwatchiin, jolloin hälytykset palvelun tilan muutoksesta voidaan ohjata eteenpäin. (Route 53 Health Checks Developer Guide, N.d.)

3.2.7 AWS CloudWatch

Cloudwatch monitoroi AWS-resursseja ja AWS:ssä toimivia sovelluksia reaaliaikaisesti. Cloudwatch kerää ja seuraa metriikoita, joita voidaan mitata resursseista ja sovelluksista. Cloudwatchissa on mahdollista luoda hälytyksiä metriikoille, jos ne rikkovat määriteltyjä raja-arvoja. (CloudWatch User Guide, N.d.)

CloudWatch Logs -palvelulla on mahdollista kerätä ja monitoroida lokia EC2-instansseista ja muista AWS:n resursseista. EC2–instansseille voidaan asentaa Cloudwatch–agentti, jolle voidaan määrittää lokitiedostot, jotka kerätään Cloudwatchiin. Lokivirroista on mahdollistaa suodattaa tapahtumia, jolloin lokista pystyy luomaan omatekoisen Cloudwatch-metriikan. Tämä mahdollistaa sovellustason valvonnan toteuttamisen Cloudwatchia käyttäen. (Cloudwatch Logs User Guide, N.d.)

3.2.8 AWS Identity and Access Management

AWS Identity and Access Management on AWS:n identiteetti- ja pääsynhallintapalvelu. IAM mahdollistaa usean eri käyttäjän luomisen AWS-tilille. IAM mahdollistaa roolipohjaisen pääsynhallinnan, jolloin käyttäjille voidaan luoda eri ryhmiä, joille on annettu tietynlaiset oikeudet tietynlaisiin resursseihin. AWS:n IAM on mahdollista federoida käyttämään esim. Microsoft Active Directorya. (IAM User Guide, N.d.)

Käyttäjien ja ryhmien lisäksi voidaan luoda IAM–rooleja, joita voidaan lisätä käyttäjille, mutta myös AWS-resursseille. Esimerkiksi jollekin EC2-instanssille voidaan liittää rooli, joka antaa sille oikeuden käyttää kontti-imagea Amazonin ECR-konttirekisteristä. (IAM Roles User Guide, N.d.)

3.3 Terraform

Terraform on Hashicorpin kehittämä työkalu infrastruktuurin luomiseen, muuttamiseen ja versiointiin. Terraformilla voi hallita tunnettujen palvelutarjoajien kuten AWS:n, Azuren ja Kubernetesin resursseja tai kustomoituja omia infrastruktuuriratkaisuja. Terraform luo infrastruktuuria, joka on kuvattu konfiguraatiotiedostoissa. Terraform generoi suunnitelman luotavista resursseista ja tallentaa infrastruktuurin tilan. Kun konfiguraatio muuttuu Terraform vertaa konfiguraatiota infrastruktuurin tilaan ja luo inkrementaalisen suunnitelman tapahtuvista muutoksista. Terraformilla voi hallita matalan tason infrastruktuurikomponentteja, kuten virtuaalikoneita, tallennustilaa ja verkkokonfiguraatiota, mutta myös korkean tason komponentteja kuten DNS–konfiguraatioita ja SaaS-palveluita. (Introduction to Terraform, N.d.)

3.3.1 Terraformin käyttäminen AWS:ssä

Terraformilla voi hallita AWS–resursseja provider-lisäosan avulla. Providerit muuttavat konfiguraatiotiedostot API-interaktioksi ja luovat näkyvyyden resursseihin. Terraform tarvitsee AWS:n komentorivi-krendentiaalit saadakseen oikeudet hallita infrastruktuuria AWS-tilillä. Kredentiaalit voidaan konfiguroida suoraan providerin konfiguraatioon, ympäristömuuttujiin, AWS:n komentorivityökaluun, tai kredentiaalivarastoon. (Terraform AWS Provider Documentation, N.d.)

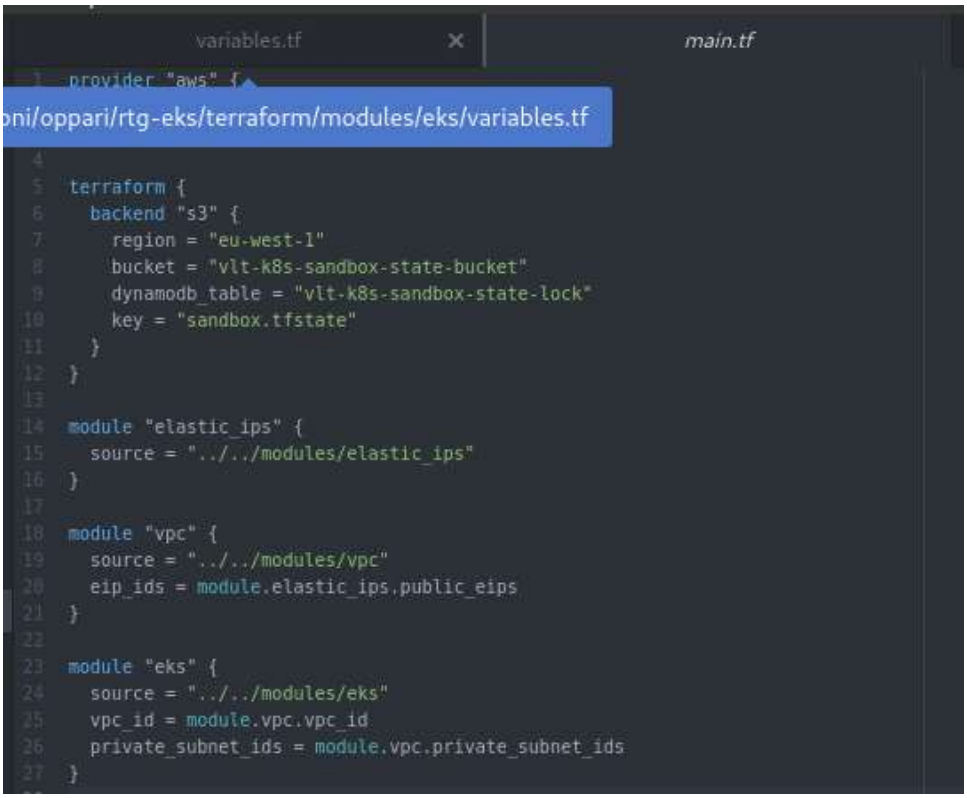
Oletuksena Terraform tallentaa infrastruktuurin tilan tiedostoon paikalliselle työasemalle. Monen kehittäjän ympäristössä paikalliset tilatiedostot tekevät kehityksestä vaikeaa, koska kehittäjien pitää huolehtia, että heillä on Terraformia ajaessaan ympäristön viimeisin tila ja ettei kukaan muu käytä Terraformia samaan aikaan. Remote State ominaisuudella Terraform voi tallentaa ympäristön tilan AWS:n S3–palveluun. S3 Remote State tukee tilan lukitusta ja eheyden tarkistusta DynamoDB:n avulla, jolloin Terraformin samanaikainen ajaminen ei ole mahdollista. Esimerkki S3 Remote State konfiguraatiosta on esitetty kuviossa 2.

```
provider "aws" {
  region = "eu-west-1"
}

resource "aws_s3_bucket" "terraform_state_bucket" {
  bucket = "vlt-k8s-sandbox-state-bucket"

  versioning {
    enabled = true
  }

  lifecycle {
    prevent_destroy = true
  }

}

resource "aws_dynamodb_table" "terraform_state_lock" {
  name = "vlt-k8s-sandbox-state-lock"
  hash_key = "LockID"
  read_capacity = 20
  write_capacity = 20

  attribute {
    name = "LockID"
    type = "S"
  }

}
```

Kuvio 2. S3-Bucketilla ja DynamoDB-taululla toteutettavan Remote Staten Terraform-konfiguraatio

Terraform-konfiguraatiot tehdään Terraformin omalla konfiguraatio kielellä tf-päätteisiin tiedostoihin. Terraform-kielen päätoiminnallisuus on resurssien kuvaaminen, ja kielen muut ominaisuudet tekevät konfiguraatiosta joustavampaa ja helppokäyttöisempää. Resursseja on mahdollista koota moduuleiksi, jotka luovat suurempia resurssikokonaisuuksia. Resurssi luo yhden infrastruktuuriobjektin, kun taas moduulilla voi luoda kokoelman objekteja ja tarvittavat riippuvuudet niiden välillä, korkeamman tason järjestelmän luomiseksi. Kuviossa 3 on esitetty esimerkki VPC–moduulin konfiguraatiosta, joka luo VPC:n riippuvuuksineen. Kuviossa 4 on esimerkki moduulin käytöstä Terraformin konfiguraatiotiedostossa. (Terraform Configuration Language, N.d.)

Kuvio 3. VPC-moduuli, joka luo tarvittavat resurssit VPC:n toimintaa varten



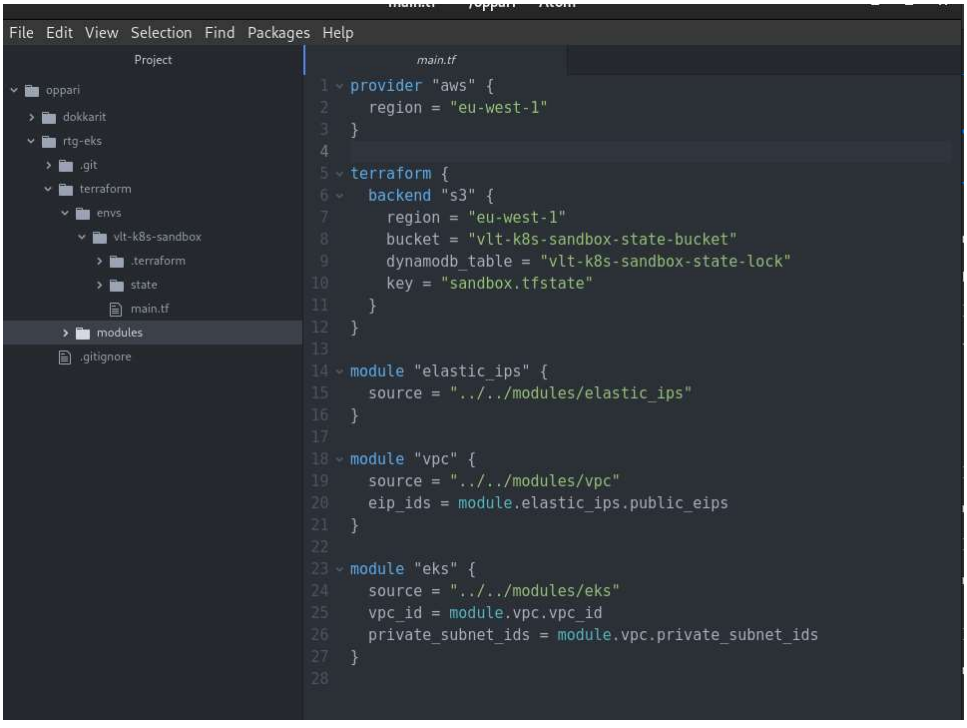
Kuvio 4. Esimerkki moduulien käyttämisestä Terraform konfiguraatiossa

Terraformin syntaksi koostuu muutamasta elementistä. *Block type* kertoo minkälainen objekti on kyseessä. *Block label* määrittää yleensä resurssin tarkemman tyypin kuten *aws_vpc* ja resurssin loogisen nimen. Terraform-objekteilla on yleensä argumentteja, joilla määritellään arvo jollekin muuttujalle. Esimerkki Terraform-objektista Kuviossa 5. (Terraform Configuration Language, N.d.)

```
resource "aws_vpc_dhcp_options_association" "this" { # block type "block_label" "block_label"
  vpc_id = aws_vpc.this.id # argument
  dhcp_options_id = aws_vpc_dhcp_options.this.id # argument
}
```

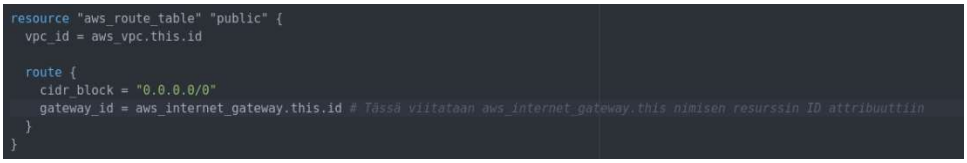
Kuvio 5. Terraform-objekti elementteineen

Terraform konfiguraatiotiedostot ovat tf-päätteisiä tiedostoja, joiden tulee käyttää UTF-8-enkoodausta ja niiden suositellaan käyttävän Unix-tyylisiä LF-rivipäätteitä. Terraform rakentaa juurimoduulin nykyisen hakemiston konfiguraatiotiedostoista ja muissa hakemistoissa sijaitseviin moduuleihin voidaan viitata konfiguraatiotiedostoissa. Yksinkertaisimmillaan juurimoduuli voi olla vain yksi tf-tiedosto. Esimerkki Terraformin juurimoduulista on esitetty kuviossa 6.



Kuvio 6. Terraform-juurimoduuli

Terraform mahdollistaa dynaamisen viittaamisen toisen Terraform-objektin attribuutteihin, mikä on erityisen hyödyllistä, sillä resurssien välillä on yleensä riippuvuuksia ja näitä arvoja ei välttämättä pystytä kovakoodaamaan. Esimerkki dynaamisesta viittauksesta kuviossa 7.

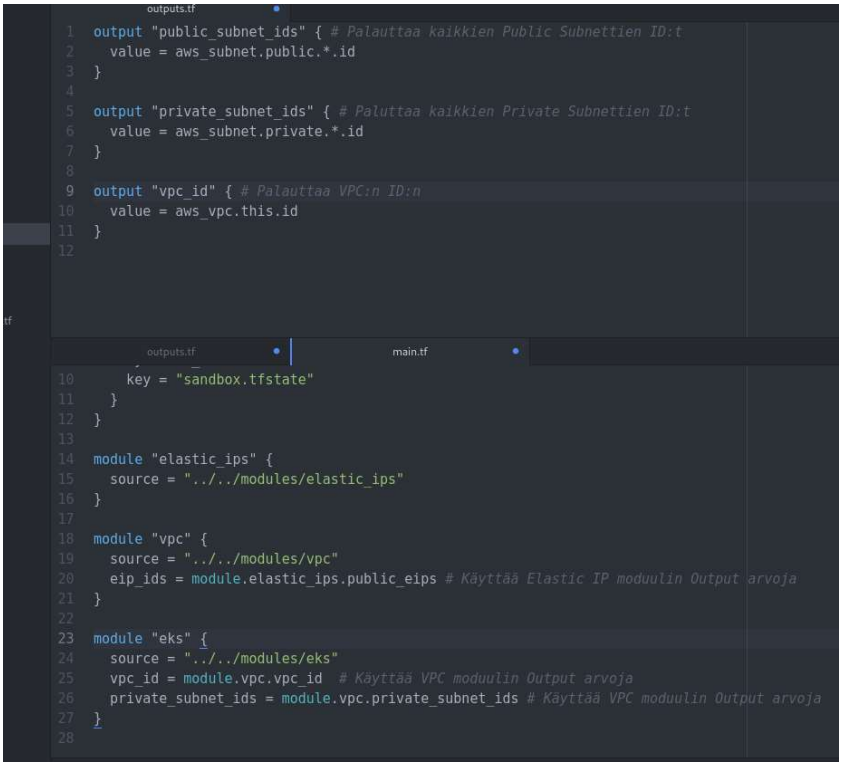


Kuvio 7. Dynaaminen viittaus Terraform-objektin attribuuttiin.

Input Variablet ovat muuttujia, joilla voidaan parametrisoida moduulin ominaisuuksia muuttamatta sen omaa lähdekoodia. Muuttujalle luodaan oma objekti, jossa kerrotaan muuttuja tyyppi, esimerkiksi merkkijono, ja tähän muuttujaan voidaan viitata jonkin argumentin arvossa. Esimerkki muuttujan luomisesta ja siihen viittaamisesta esitetty kuviossa 8. (Terraform Input Variables Documentation, N.d.)

Kuvio 8. Terraform-muuttuja ja viittaus muuttujaan

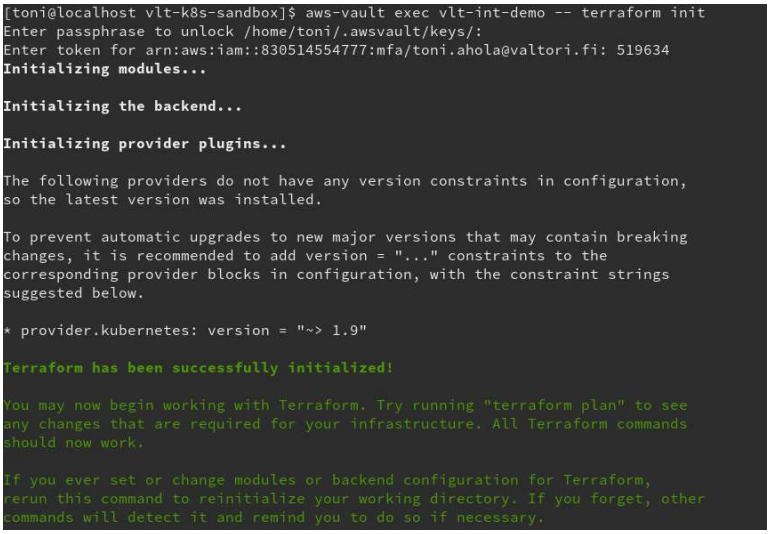
Output Valuet ovat arvoja, joita Terraform-moduulit palauttavat. Output Valuet ovat hyödyllisiä, kun lapsimoduuleista halutaan tuoda arvoja muiden lapsimoduulien käyttöön. Esimerkiksi VPC–moduulista voidaan tuoda aliverkkojen tunnisteet EKS-moduulin käyttöön, jotta EKS-moduuli osaa luoda klusterin VPC-moduulissa luoduille aliverkoihin. Esimerkki Output Valueiden luomisesta ja niiden käytöstä kuviossa 9. (Terraform Output Values Documentation, N.d.)



Kuvio 9. Terraform output valuen määritys siihen viittaaminen

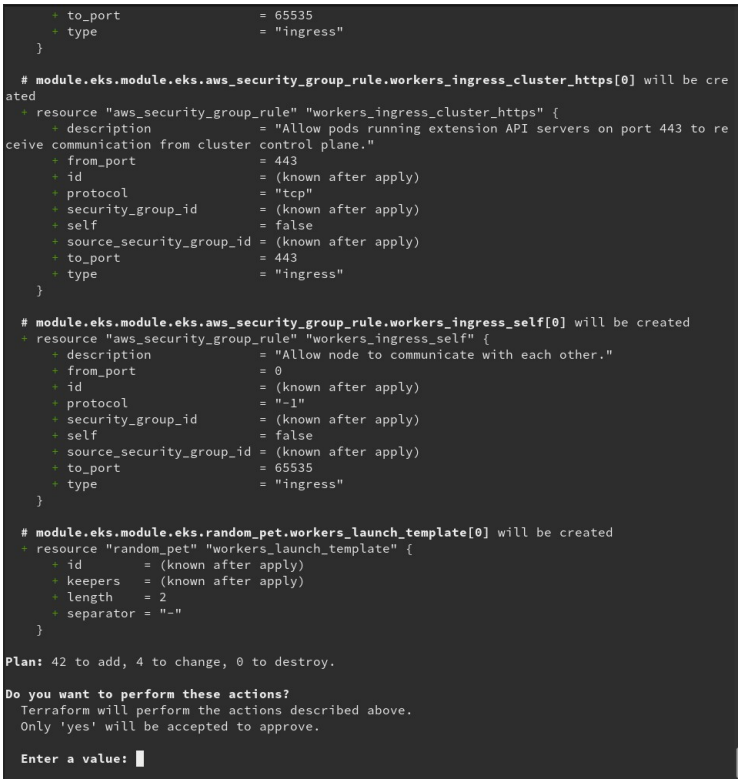
3.3.2 Terraform-komentorivityökalun käyttö

Terraform on asennettavissa binääripakettina Windows-, Linux- ja OSX-käyttöjärjestelmille. Kun Terraform on asennettu, uusi työhakemisto alustetaan *terraform init* -komennolla. Alustaessa luodaan paikallinen tilatiedosto, ellei konfiguraatiossa ole erikseen määritelty esim. Remote Statea. *Init* myös asentaa konfiguraatiossa määritellyt provider-lisäosat ja lapsimoduulien lähdekoodit. Optiolla *upgrade* Terraform päivittää lisäosien versiot ja hakee lapsimoduulien uusimman lähdekoodin. Esimerkki *terraform init* -komennosta esitetty kuviossa 10. (Terraform Command Reference, N.d.)



Kuvio 10. Terraform Init -komennon komentorivituloste

Kun konfiguraatio muuttuu Terraformin tekemiä muutoksia voi katselmoida *terraform plan* -komennolla. Muutokset infrastruktuuriin tehdään *terraform apply* -komennolla. Terraformin koodi on deklaratiivista ja mikäli konfiguraatiosta poistetaan objekteja niin Terraform tuhoaa konfiguraatiosta poistuneet resurssit. Koko Terraform juurimoduulin infrastruktuurin voi tuhota komennolla *terraform destroy*. Esimerkki *terraform apply* -komennosta on esitetty kuviossa 11. (Terraform Command Reference, N.d.)

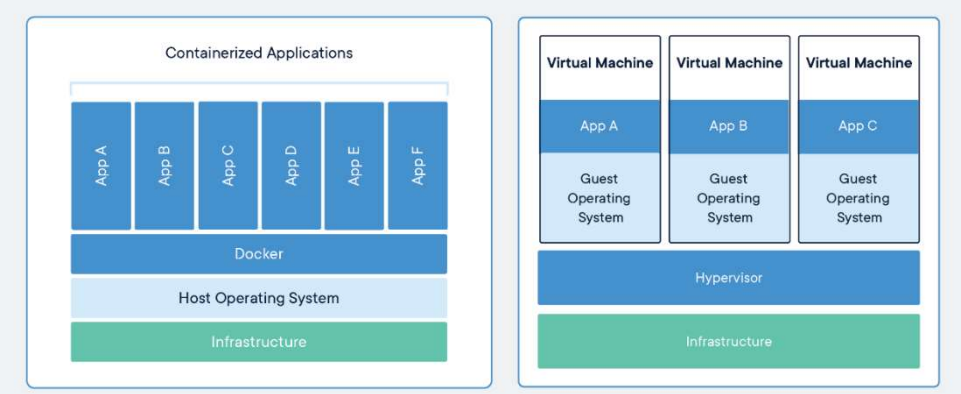


Kuvio 11. Terraform apply -komennon komentorivituloste

3.4 Docker

Docker-konttiteknologia julkaistiin vuonna 2013 avoimen lähdekoodin Docker Enginenä. Kontti on standardoitu ohjelmiston yksikkö, joka pakkaa koodin ja sen riippuvuudet, jotta sovellukset voivat toimia nopeasti ja luotettavasti ajoympäristöstä riippumatta. Docker Imageen on pakattu kaikki sovelluksen ajoon tarvittavat riippuvuudet kuten itse koodi, käyttöjärjestelmä, työkalut, kirjastot ja asetukset. Kun Docker Imageita ajetaan Docker Enginellä, niistä tulee Docker-kontteja. Docker Engineä voi ajaa Linux- ja Windows-ympäristössä, mutta kontit toimivat niissä täysin samalla tavalla. Kontit eristävät sovelluksen ajoympäristöstä ja varmistavat että sovellus toimii yhdenmukaisesti jokaisessa ympäristössä. (What is a Container, N.d.)

Konteilla ja virtuaalikoneilla on yhteneväisyyksiä resurssien eristämisessä ja allokoimisessa, mutta ne toimivat eri tavalla, koska kontit virtualisoivat käyttöjärjestelmää laitteiston sijaan. Useita kontteja voidaan ajaa samalla koneella, ja ne jakavat saman kernelin muiden konttien kanssa, jokainen ajaen omia eristettyjä prosessejaan. Kontit ovat kevyempiä virtuaalikoneisiin verrattuna, koska virtuaalikoneet jakavat ainoastaan fyysisen laitteiston, mutta jokainen virtuaalikone tarvitsee täyden käyttöjärjestelmän, kun taas kontit jakavat kernelin. Kuva konttien ja virtuaalikoneiden erosta esitetty kuviossa 12. (Wong 2016)

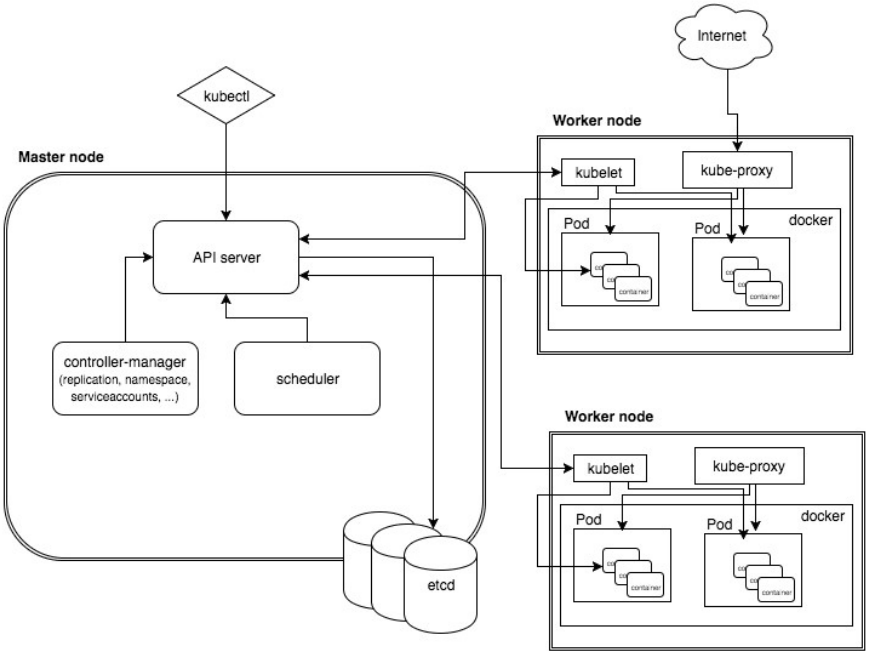


Kuvio 12. Konttien ja virtuaalikoneiden isolaatiot (What is a Container, n.d.)

3.5 Kubernetes

Kubernetes on avoimen lähdekoodin konttiorkestrointialusta, joka mahdollistaa kontitettujen sovellusten hallitsemisen ja automaation deklaratiivisen konfiguraation avulla. Kubernetes oli alun perin Googlen vuonna 2014 aloittama avoimen lähdekoodin projekti, mutta on nykyään Cloud Native Computing Foundationin (CNCF) ylläpitämä. Kubernetes mahdollistaa useita tärkeitä ominaisuuksia konttien hallintaan, kuten konttien ryhmittelyn, service discoveryn, kuormantasauksen, tallennustilan orkestraation, automaattiset päivitykset, palautukset, resurssien käyttörajoitukset, itsestään palautuvuuden sekä salaisuuksien ja konfiguraation hallinnan. (What is Kubernetes, N.d.)

Kubernetes toimii klusterina. Klusteri on kokoelma palvelimia, joita kutsutaan noodeiksi. Noodeilla ajetaan kontteja, joita Kubernetes hallitsee. Klusterilla on vähintään yksi isäntänoodi ja yksi työläisnoodi. Työläisnoodeilla ajetaan podeja, jotka ovat yhden tai useamman kontin kokonaisuuksia. Isäntä hallitsee työläisnoodeja ja klusterin podeja. Usealla isäntänoodilla voidaan saavuttaa korkeampi vikasietoisuus, mutta vain yksi noodi voi olla aktiivinen isäntä. Kubernetes-klusterin arkkitehtuuri esitetty kuviossa 13. (Kubernetes Components, N.d.)



Kuvio 13. Kubernetes-klusterin arkkitehtuuri (Introduction to Kubernetes architecture 2016)

3.5.1 Isäntänoodi

Isäntänoodi komponentteineen, muodostaa klusterin ohjaustason. Komponentit tekevät päätöksiä koko klusterin tasolla ja reagoivat klusterin tapahtumiin. Isäntänoodin neljä komponenttia ovat apiserver, etcd, scheduler ja controller-manager. (Kubernetes Components, N.d.)

Apiserver on ohjaustason komponentti, joka paljastaa Kubernetes API:n klusterin ulkopuolelle. Apiserver toimii ohjaustason käyttöliittymänä ja sitä komennetaan kubectl-komentorivityökalulla. Scheduler on ohjaustason komponentti, joka tarkkailee uusia podeja ja jakelee niitä työläisnoodeille isännöitäväksi. Scheduler tekee valinnat yksittäisten ja kollektiivisten resurssivaatimusten, laitteisto- tai ohjelmistorajoitteiden, riippuvuuksien, datan sijainnin ja kuormien välisten häiriöiden ja määriaikojen perusteella. Etcd on korkeasti saatavissa oleva avainarvo varasto, johon kaikki data klusterista tallennetaan. (Kubernetes Components, N.d.)

Controller-manager on yksi binääri, josta ajetaan Kubernetesin kontrollereja erillisinä prosesseinaan. Yksi kontrolleri tarkkailee ainakin yhtä Kubernetesin resurssityyppiä. Kontrollerit vertailevat objektien nykyistä tilaa konfiguraatiossa määritettyyn ihannetilaan ja muuttavat nykyistä tilaa lähemmäksi ihannetilaa. Noodi-kontrolleri tarkkailee työläisnoodien terveyttä ja ilmoittaa jos noodi vikaantuu. Replikaatio-kontrolleri huolehtii, että podista on tietty määrä replikoita ajossa. Endpoint-kontrolleri huolehtii, että podit ovat saavutettavissa klusterin sisällä muille palveluille. Palvelutili- ja tunnuskontrollerit luovat oletustunnukset ja API-avaimet uusille nimiavaruuksille. (Kubernetes Controllers Overview, N.d.)

3.5.2 Työläisnoodi

Työläisnoodit ovat Kubernetes-klusterin koneita, joilla kontteja isännöidään. Työläisnoodit voivat olla fyysisiä palvelimia tai virtuaalikoneita. Jokaisella työläisnoodilla on ajossa kubelet, kube-proxy ja Docker. Kubelet huolehtii, että noodilla ajettavat kontit ovat ajossa podien sisällä. Kubelet tarkistaa, että podin konfiguraatiossa määritellyt kontit ovat käynnissä ja terveitä. Kube-proxy on verkko-proxy, joka säätelee verkkosääntöjä noodeilla mahdollistaen yhteydet podeihin klusterin sisä- ja ulkopuolelta.

Kube-proxy käyttää käyttöjärjestelmän paketinsuodatustasoa, kuten Linux-käyttöjärjestelmien Iptables-palomuuria. Kontteja ajetaan Kubernetesissa yleensä Dockerilla, mutta Kubernetes tukee myös vaihtoehtoisia konttiteknologioita. (Kubernetes Components, N.d.).

3.5.3 Kubernetes-objektit

Kubernetes-objektit ovat persistenttejä itsenäisiä kokonaisuuksia, joita Kubernetes käyttää kuvaamaan klusterin tilaa. Niillä voidaan kuvata mitkä kontitetut sovellukset ovat käynnissä ja millä podilla, mitä resursseja näille aplikaatioille on saatavilla ja miten sovellukset käyttäytyvät, kuten uudelleenkäynnistymiskäytänteet, päivitykset ja vikasietoisuus. Objekteja luodaan, muokataan ja tuhotaan Kubernetes API:n kautta. Kubernetes API:a hallitaan kubectl-komentorivityökalulla ja objektit luodaan YAML-muotoisissa tiedostoissa. (Kubernetes Objects, N.d.)

Jokaisella Kubernetesin objektilla on *spec-* ja *status*-kentät, jotka hallitsevat objektin konfiguraatiota. Objektia kuvatessa *spec*-kenttään määritellään objektin haluttu tila ja sen ominaisuudet. *Status* sen sijaan on objektin todellinen tila, jota Kubernetes päivittää. Kubernetesin hallintataso aktiivisesti säätelee objektin todellista tilaa vastaamaan *spec*-kentässä ilmoitettua haluttua tilaa. Esimerkiksi objektin *spec*-kentässä, voidaan ilmoittaa podin replikoiden määräksi 3, jolloin podin vikaantuessa käynnistetään automaattisesti uusi podi, jolloin haluttu tila säilytetään. (Understanding Kubernetes Objects, N.d.)

Kubernetes-objekti luodaan määrittämällä objektille *spec*-kenttä, johon määritellään haluttu tila ja objektin yleiset tiedot. Kubernetes API vastaanottaa objektin tiedot JSON-muotoisena API-kutsuna. Kubectl-komentorivityökalulla objektit kuvataan YAML-muodossa, jonka kubectl konvertoi JSON-muotoon Kubernetes API:a varten. Objekti kenttineen kuvattu kuviossa 14. Kubernetes-objektit luodaan komennolla *kubectl apply* ja antamalla komennon argumentiksi konfiguraatiotiedosto. Objektin luominen esitetty kuviossa 15. (Working with Kubernetes Objects, N.d.)

nginx-deployment.yaml

```yaml
apiVersion: apps/v1 # Kubernetes API:n versio jolla objekti luodaan
kind: Deployment # Objektin tyyppi
metadata: # Tietoa joka helpottaa objektin tunnistamisessa
  name: nginx-deployment
spec: # Objektin haluttu tila
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # Kuinka monta replikaa podista on käynnissä
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9 # Ajettavan kontin image
        ports:
        - containerPort: 80 # Portti josta kontti saavutetaan
```

Kuvio 14. Deployment-objekti

```
toni@localhost:~/oppari/rtg-eks/kube
[toni@localhost kube]$ kubectl apply -f nginx-deployment.yaml
deployment.apps/nginx-deployment created
[toni@localhost kube]$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
alb-ingress-controller-858f5b6659-68q2l   1/1     Running   0          14m
external-dns-56f8dfd6b4-qshvz             1/1     Running   0          14m
nginx-deployment-6dd86d77d-d8lpz          1/1     Running   0          8s
nginx-deployment-6dd86d77d-rhbnl          1/1     Running   0          8s
[toni@localhost kube]$
```

Kuvio 15. Kuvion 13 Deployment-objektin luominen kubectl apply -komennolla

3.5.4 Pod ja ReplicaSet

Podi on pienin Kubernetesiin luotava objekti. Podi on yhden tai useamman kontin joukko, jotka jakavat tallennustilan, verkkokonfiguraation ja määritelmän miten kontteja ajetaan. Podin kontteja ajetaan jaetussa kontekstissa eli joukossa Linux-nimiavaruuksia, CGroupeja ja muita eristysmenetelmiä. Podin kontit jakavat IP-osoitteen ja porttiavaruuden, joten ne löytävät toisensa localhostista, kun taas eri podeissa sijaitsevat kontit kommunikoivat podien IP-osoitteiden avulla. Podit ovat suunniteltu suhteellisen lyhytikäisiksi ja tilattomien sovellusten ajamiseen. Podit luodaan uniikilla tunnisteella jollekin noodille. Podi pysyy samalla noodilla koko sen elinkaaren ajan, kunnes se poistetaan. Kun podi on poistunut sitä ei voida ottaa käyttöön uudelleen vaan se korvataan uudella identtisellä podilla ja uudella tunnisteella. (Pod Documentation, N.d.)

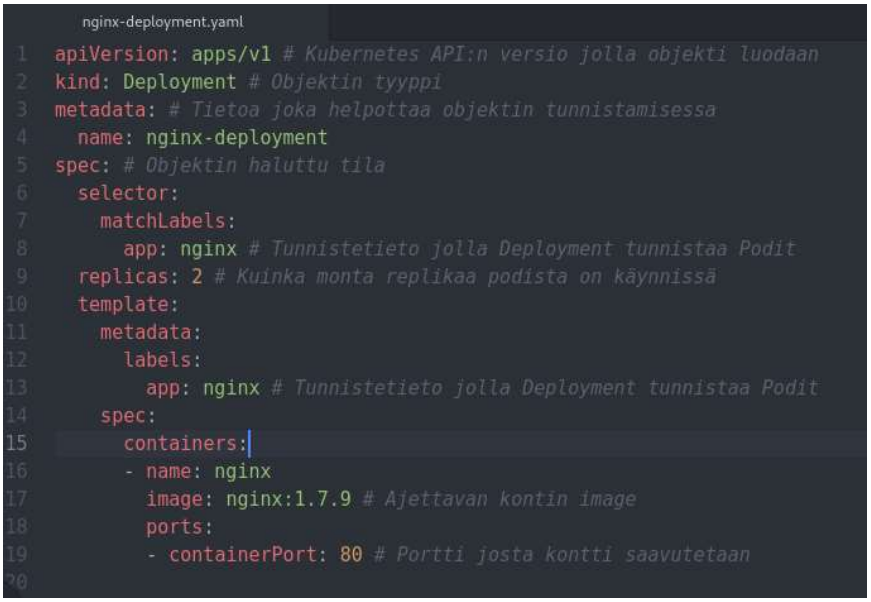
Podien pääasiallinen käyttötarkoitus on tukea tilattomia samapaikkaisia, yhteisesti ohjattavia apuohjelmia. Podit ovat suunniteltu lyhytikäisiksi ja ne eivät selviydy noodien häiriöistä tai resurssien puutteesta. Podeja ei yleensä tule luoda itsenäisenä vaan jonkin muun kontrollerin kuten Deploymentin kautta. Kontrollerit mahdollistavat sovellukselle mm. vikasietoisuutta klusterin tasolla replikoimalla podeja usealle noodille. StatefulSet-kontrollerilla voidaan tukea tilallisia podeja. (Pod Documentation, N.d.)

ReplicaSet ylläpitää tiettyä määrää podin kopioita ja sitä käytetään usein takaamaan, että tietty määrä identtisiä podeja on saatavilla. ReplicaSetiin määritellään *selector*, jolla tunnistetaan podit, ylläpidettävien podin replikoiden määrä ja replikoitavan podin konfiguraatio. ReplicaSetejä kuten podeja ei kannata luoda itsenäisenä vaan käyttää niiden hallintaan korkeamman tason kontrollereja kuten Deploymentia, joka mahdollistaa mm. podien deklaratiivisen päivityksen. (Replicaset Documentation, N.d.)

3.5.5 Deployment

Deploymentilla voidaan toteuttaa deklaratiivisia päivityksiä podeille ja ReplicaSeteille. Deploymentissa määritellään podin tai ReplicaSetin haluttu tila. Deployment kontrolleri säätelee todellista tilaa vastaamaan konfiguraatiossa määritettyä haluttua tilaa. Deploymentilla on mahdollista esim. päivittää podi tai replicaset käyttämään uutta konttiversiota deklaratiivisesti. Deploymentit mahdollistavat myös palauttamisen edelliseen Deploymentin versioon mikäli nykyinen ei ole vakaa. Deploymentit kuten muutkin Kubernetes-objektit kuvataan YAML-tiedostossa, joka välitetään Kubernetes API:lle kubectl-komentorivityökalulla. (Deployment Documentation, N.d.)

Deployment-objektin konfiguraatiossa määritellään podit sekä niissä ajettavat kontit. Deployment-konfiguraatiossa voidaan määrittää podin replikoiden lukumäärä ja päivitysstrategiat. Esimerkki Deployment-konfiguraatiosta esitetty kuviossa 16.



Kuvio 16. Deploymentin konfiguraatio

3.5.6 Storage

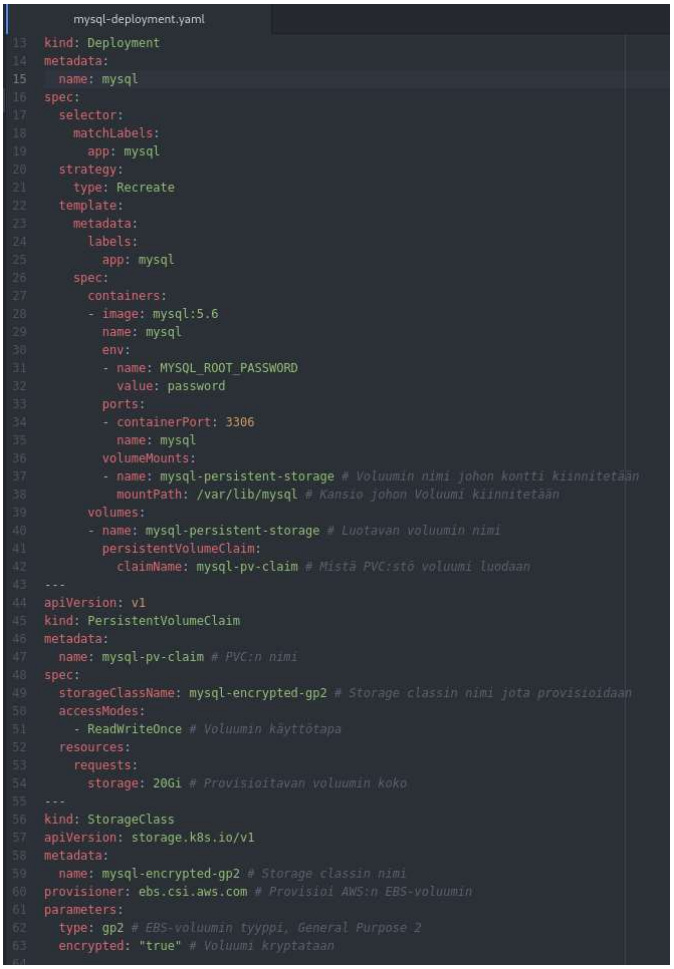
Konttien kirjoittamat levytiedostot ovat väliaikaisia, joka aiheuttaa ongelmia joidenkin sovellusten ajamisessa. Kubernetes tarjoaa ratkaisuja tallennustilan hallitsemiseen podien yli. Podille voidaan määrittää voluumi podin *spec*-kentässä. Voluumi on pohjimmiltaan vain hakemisto, joka on podin konttien saavutettavissa. Voluumin tyyppi määrittää millaisella levyllä hakemisto on. Voluumin ja Podin elinkaari on sama, eli voluumi poistetaan kun podi poistetaan. Kubernetes tukee useita eri voluumityyppejä kuten ISCSI, NFS, AWS EBS ja Azure Disk. Podit voivat olla kiinnitettynä useisiin eri voluumehin samanaikaisesti. (Volume Documentation, N.d.)

Kubernetes tukee EBS-voluumin kiinnittämistä podiin. Kun podi poistetaan, EBS-levyn sisältö säilyy. EBS-levyt voidaan esipopuloida datalla, joka voidaan antaa podien käytettäväksi. EBS-voluumien kiinnittämiseen liittyy tiettyjä rajoituksia, kuten työläisnoodien tulee sijaita samalla saatavuusvyöhykkeellä EBS-voluumin kanssa. EBS-voluumiin voi kiinnittää vain yhden noodin. (Kubernetes Volume Documentation, N.d.)

Kubernetesin Persistent Volumet tarjoavat API:n pysyvien voluumien luomiseen. API-resurssit Persistent Volume (PV) ja Persistent Volume Claim (PVC) abstraktoivat miten tallennustila tarjotaan ja miten sitä käytetään. PV on tallennustilaa klusterissa,

joka provisioidaan itse tai dynaamisesti Storage Classin avulla. PV:t ovat samankaltaisia kuin voluumit, mutta niillä on itsenäinen elinkaari voluumia käyttävistä podeista. PVC:llä otetaan podeille tallennustilaa käyttöön PV:stä. PVC:ssä kerrotaan tarvittava tallennustilan määrä ja käyttötapa (luku/kirjoitus tai vain luku) Esimerkki PV:n konfiguraatiosta esitetty kuviossa 16. (Persistent Volume Documentation, N.d.)

Storage Classit ovat Kubernetes-resursseja, joilla voidaan kuvailla eri tallennustiloja luokkina. Luokittelu voidaan tehdä esim. palvelunlaadun tai varmuuskopiointikäytänteiden perusteella. Storage Classilla määritellään minkä tyyppinen voluumi provisoidaan ja miten niiden elinkaaria hallitaan. Esimerkki Storage Classin konfiguraatiosta esitetty kuviossa 17. (Kubernetes Storage Class Documentation, N.d.)



Kuvio 17. Storage Classin ja Persistent Volumen konfigurointi MySQL-Deploymentin podien käytettäväksi

3.5.7 Service

Servicellä paljastetaan podit klusterin sisäiselle liikenteelle. Service on abstraktio, joka määrittelee loogisen ryhmän podeja ja käytänteet, miten ne saavutetaan. Service objekti kohdistaa liikenteen podille *selectorin* avulla. Kubernetes antaa Servicelle IP-osoitteen, jota Servicen proxyt käyttävät. Servicessä määritetään portti mistä Service kuuntelee ja portti mihin liikenne ohjataan podien konteille. Esimerkki Servicen konfiguraatiosta esitetty kuviossa 18. (Service Documentation, N.d.)

mysql-deployment.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql # Servicen nimi
spec:
  ports:
  - port: 3306 # Portti jota sovellus käyttää,
               # koska targetPortia ei ole määritelty
               # liikenne ohjataan porttiin 3306 myös kontin sisällä
  selector:
    app: mysql # Service liitetään podeihin tällä selectorilla
  clusterIP: None
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql # Service liitetään podeihin tällä selectorilla
  strategy:
    type: Recreate
```

Kuvio 18. Servicen konfiguraatio

3.5.8 Ingress

Ingress on API-objekti, jolla hallinnoidaan klusterin ulkopuolelta tulevaa liikennettä klusterissa ajettaviin sovelluksiin kuormantasaajaa käyttäen. Ingress ohjaa sisääntulevan liikenteen kuormantasaajalta Serviceille, jotka ohjaavat liikenteen podeille. Ingress paljastaa HTTP/HTTPS-reitit klusterin sovelluksiin. Reititys määritellään sääntöinä Ingress-resurssiin. Ingressiin voidaan konfiguroida ulkoisesti saavutettavissa olevat verkkotunnukset, kuormantasaajan tyyppi, TLS-varmenne (Transport Layer Security) ja nimipohjainen virtuaalinen isännöinti. Ingress tarvitsee toimiakseen Ingress Conrollerin, joka yleensä toteutetaan kuormantasaajalla. Esimerkki Ingress-konfiguraatiosta esitetty kuviossa 19. (Kubernetes Ingress Documentation, N.d.)

Kuvio 19. Kubernetes Ingressin konfiguraatio

3.6 AWS Elastic Kubernetes Service

3.6.1 EKS-hallintataso

Amazon Elastic Kubernetes Service (EKS) on AWS:n hallittu palvelu, joka mahdollistaa Kubernetes-klusterin pystytyksen helposti AWS–pilvipalveluun. EKS sisältää Kubernetesin hallintatason replikoituna usealle saatavuusvyöhykkeelle saatavuuden takaamiseksi. EKS:n hallintataso sisältää valmiiksi provisioidut isäntänoodit komponentteineen, joiden ylläpitäminen on AWS:n vastuulla. EKS ajaa avoimen lähdekoodin Kubernetes-ohjelmistoa, joten kaikki itse ylläpidetyissä klustereissa toimivat sovellukset ovat yhteensopivia EKS:n kanssa. (EKS User Guide, N.d.)

3.6.2 EKS-työläisnoodit

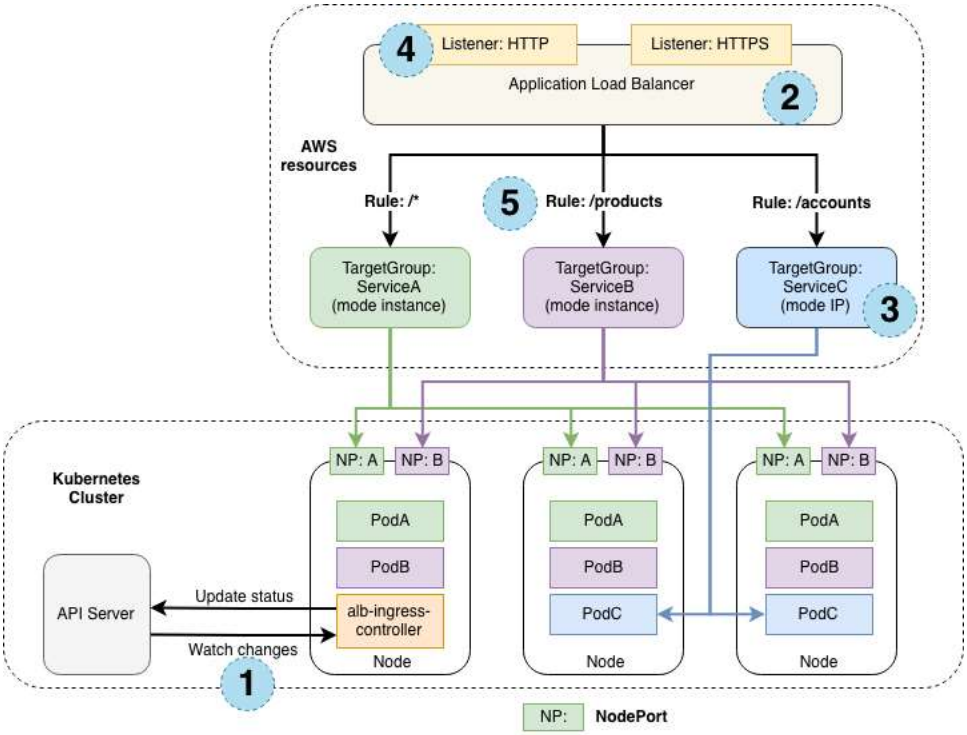
Työläisnoodit ovat EC2–instansseja, joihin Amazon tarjoaa valmiita levykuvia provisiointia varten. Toisin kuin isäntänoodit, työläisnoodien ylläpito ja päivittäminen on käyttäjän vastuulla. AWS julkaisee päivitettyjä levykuvia työläisnoodeille säännöllisin väliajoin ja noodit tulisi päivittää uusiin versioihin säännöllisesti tietoturvapäivitysten saamiseksi. (EKS Worker Node User Guide, N.d.)

3.6.3 Klusterin sisäiset kontrollerit

AWS ALB Ingress Controller

Kubernetes-klusteri tarvitsee Ingress Controllerin, jotta kontit ovat saavutettavissa klusterin ulkopuolelta. Koska klusteri on tehty AWS-pilvipalvelualustalle, käytetään AWS ALB Ingress Controlleria. AWS ALB Ingress Controller tarkkailee Kubernetes-

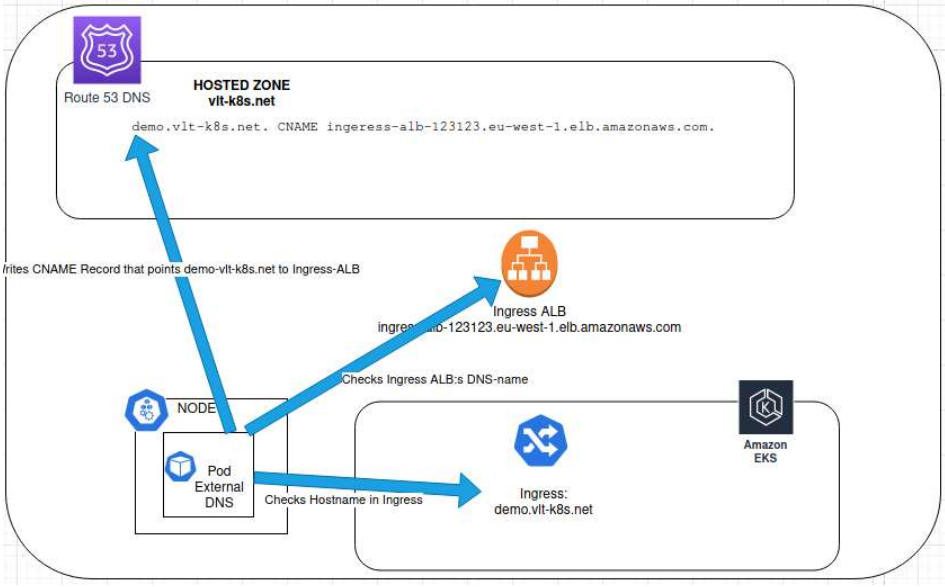
klusterin Ingress-tapahtumia ja kun se löytää Ingress-resurssin se luo ALB–kuormantasaajan. Ingress Controller luo TargetGroupit Ingress-resurssissa määritellyille backendeille, sekä Listenerit Ingress-resurssissa määritellyille porteille. Ingress Controller luo myös kuormantasaajalle reitityssäännöt, jotta liikenne ohjautuu polkua vastaavalle Podille. Ingress Controller otetaan käyttöön Deploymentina. AWS ALB Ingress Contorollerin toiminta esitetty kuviossa 20.



Kuvio 20. AWS ALB Ingress Controllerin toiminta

External DNS

External DNS luo saavutettaville sovelluksille julkisen Domain-nimen. External DNS katsoo sovelluksen Ingress-säännössä määritellyn domain nimen ja luo CNAME-tietueen, joka osoittaa määritellyn verkkotunnuksen Ingress–kuormantasaajan nimeen. External DNS otetaan käyttöön Deploymentina. External DNS:n toiminta esitetty kuviossa 21.



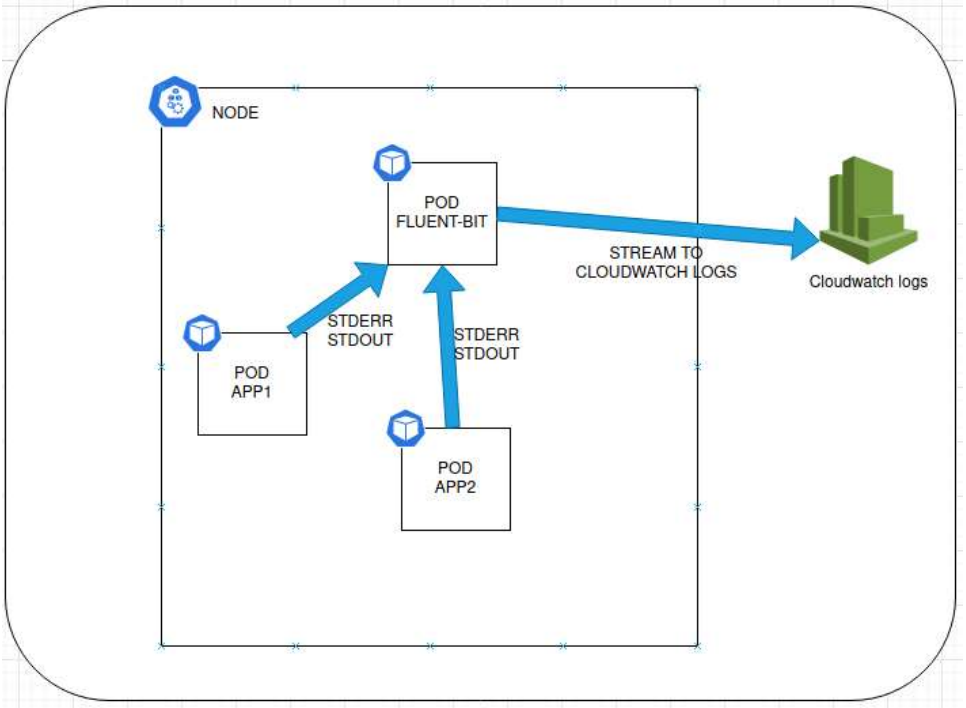
Kuvio 21. External DNS:n toiminta

Cloudwatch Agent

Cloudwatch Agent vie podien ja noodien metriikat kuten muistin- ja prosessorikäytön Cloudwatch-monitorointipalveluun. Yksittäisten noodien ja podien metriikoiden lisäksi saadaan myös koko klusterin laajuista metriikkaa, kuten epäterveiden podien ja noodien määrä. Cloudwatch Agent otetaan käyttöön DaemonSetinä, jolloin yhtä Cloudwatch Agent -podin replikaa ajetaan jokaisella klusterin noodilla.

Fluent-bit

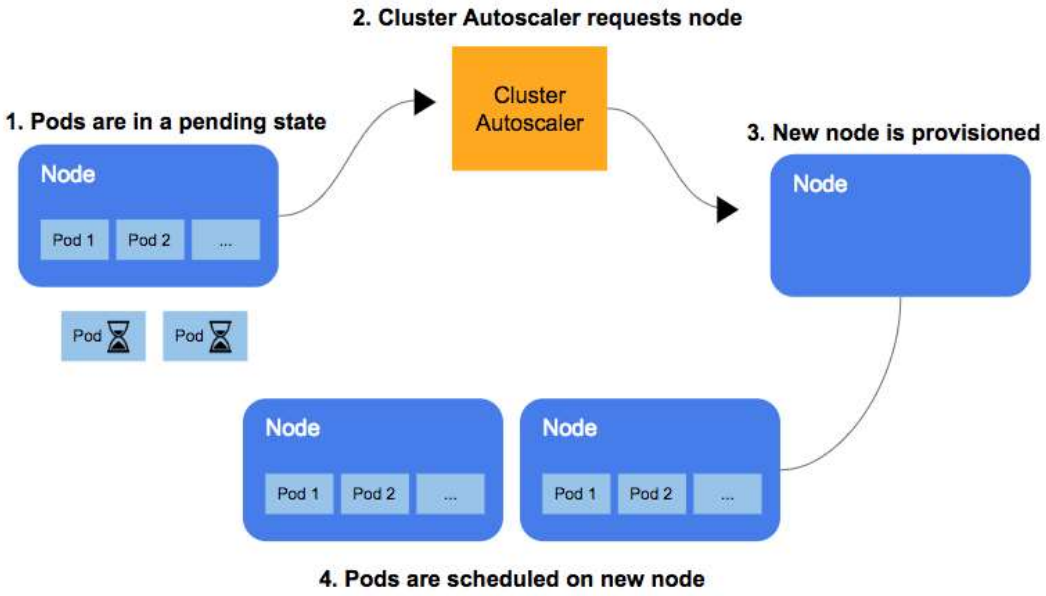
Fluent-bit on avoimen lähdekoodin sovellus, joka kerää, prosessoi ja eteenpäin lähettää lokia. Fluent-bit lukee podien stdout- ja stderr-virtoja, joista se kerää logit ja lähettää ne eteenpäin Cloudwatch Logs –palveluun. Cloudwatch Logsissa voi prosessoida lokia, hallita sen elämänkaarta ja luoda siihen pohjautuvia valvontoja. Fluent-bit otetaan käyttöön DaemonSetinä, jolloin yhtä podin replikaa, ajetaan jokaisella klusterin noodilla. Fluent-bitin toiminta esitetty kuviossa 22.



Kuvio 22. Fluent-bitin toiminta Kubernetes noodilla

Cluster Autoscaler

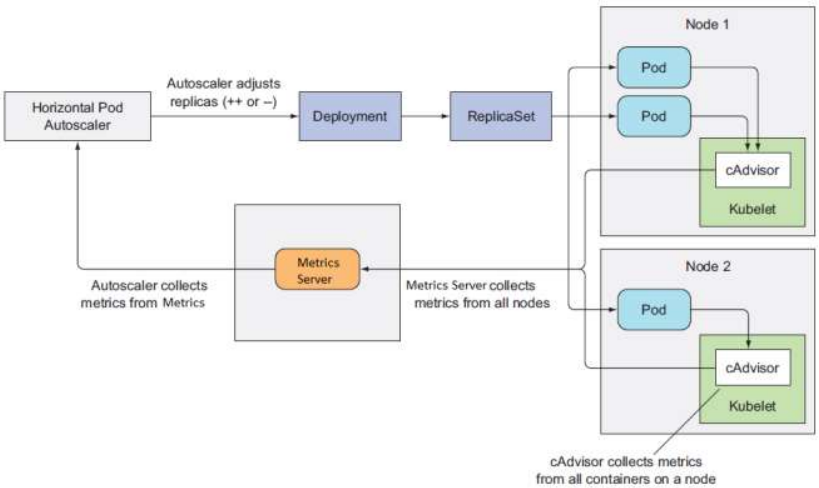
Cluster Autoscaler poistaa ja lisää noodeja klusteriin podien resurssivaatimusten perusteella. Cluster Autoscaler lisää autoskaalausryhmään noodin kun joku podeista menee *pending* tilaan, koska sen CPU- tai muistivarauksia ei pystytä täyttämään. *Pending* tilassa olevat podit otetaan käyttöön uudella noodilla. Noodien lisäysalgoritmi esitetty kuviossa 23. Cluster Autoscaler poistaa noodeja klusterista, kun podien resurssivaraukset alittavat määritellyn raja-arvon (oletuksena 50% resurssista), lukuunottamatta poikkeustapauksia, jotka estävät noodin poiston (esim. noodille persistoidaan tilaa). Cluster Autoscaler otetaan käyttöön Deploymentina. (Tripathy 2019)



Kuvio 23. Cluster Autoscalerin noodin lisäys -algoritmi (Tripathy 2019)

Metrics Server

Metrics Server kerää metriikkaa podien kubeletien kuormittumisesta ja julkaisee ne MetricsAPI:n kautta Apiserverille, joka mahdollistaa Horizontal Pod Autoscalerin (HPA) ja Vertical Pod Autoscalerin (VPA) käytön. Metrics Server otetaan käyttöön Kubernetes Deploymentina. Metrics Server mahdollistaa autoskaalauksen vain prosessori- ja muistinkäytön perusteella. HPA:n toiminta Metrics Serverin avulla esitetty kuviossa 24.



Kuvio 24. Metrics Serverin ja HPA:n toiminta Kubernetes-klusterissa (Metrics Server Github project)

4 Tekninen toteutus

4.1 Työkalujen asennus ja konfigurointi

Ohjelmalliseen infrastruktuurin luomiseen vaaditaan AWS:n komentorivityökalu AWS-Cli. AWS-Cli vaatii API-kutsujen lähettämiseen AWS-käyttäjän pääsyavaimet, jotka voidaan konfiguroida suoraan AWS-Clin konfiguraatiotiedostoon. Pääsyavaimien säilyttäminen salaamattomana konfiguraatiotiedostossa on kuitenkin huono idea, joten käytetään avoimen lähdekoodin aws-vaultia, joka kryptaa avaimet tiedostoon tai salasanamanageriin. AWS-Clin konfiguraatiotiedostoon on konfiguroitu profiili käyttäjänhallintatilille federation, jossa IAM-käyttäjä ja pääsyavaimet sijaitsevat. Demo-profiili on tilille mihin infrastruktuuri pystytetään. Demo-tilille ei ole pääsyavaimia, eikä käyttäjätunnusta vaan federaatiotilin käyttäjätunnus ottaa IAM-roolin käyttöönsä tällä tilillä. AWS-Clin konfiguraatio esitetty kuviossa 25.

Kuvio 25. AWSCli-konfiguraatiotiedosto

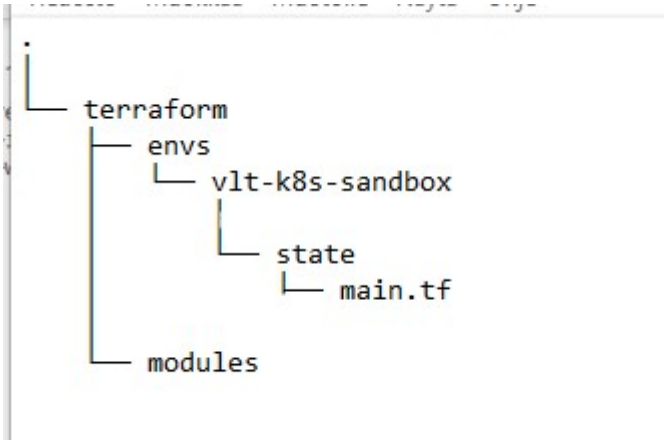
Kun aws-vaultille on annettu pääsyavaimet federationiin, voidaan suorittaa AWSCli-komentoja tilille demo. Komentoihin tarvitaan ”aws-vault exec profiilin-nimi -- ” etuliite pääsyavainten saamiseksi. Esimerkki AWSCli-komennosta aws-vaultin kanssa esitetty kuviossa 26.

Kuvio 26. AWSCli-komento aws-vaultia käyttäen

4.2 Infrastruktuurin luominen

4.2.1 Terraform-tilatiedoston asettaminen S3-tallennuspalveluun

Terraform-konfiguraatio luodaan Github-versionhallintaan. Github-repositorion juureen luodaan *terraform* niminen hakemisto, johon luodaan kaksi alihakemistoa *terraform/modules* ja *terraform/envs*. *Envs*-hakemisto sisältää ympäristökohtaiset infrastruktuurikonfiguraatiot ja *modules*-hakemisto sisältää yleiskäyttöiset moduulit, jotka voidaan ottaa käyttöön useassa ympäristössä. Hakemistorakenne esitetty kuviossa 27.



Kuvio 27. Terraform -koodin hakemistorakenne

Ensimmäisenä ympäristölle täytyy luoda Terraform-tilatiedosto S3-palveluun. Ympäristön hakemistoon luodaan *state*-alihakemisto, jonka *main.tf*-tiedostossa luodaan Terraform Remote Stateen vaadittava infrastruktuuri. Konfiguraatiossa asetetaan juurimoduulille AWS-provider, joka mahdollistaa Terraformin tehdä API-interaktioita AWS-tilille. Tiedostossa luodaan myös S3-bucket, johon Terraformin tilatiedosto tallennetaan ja DynamoDB-taulu, joka lukitsee tilatiedoston Terraform-ajon ajaksi, ettei monta käyttäjää voi ajaa samaa Terraform-juurimoduulia samanaikaisesti. Remote State -konfiguraatio esitetty kuviossa 28.

terraform > envs > vlt-k8s-sandbox > state > main.tf

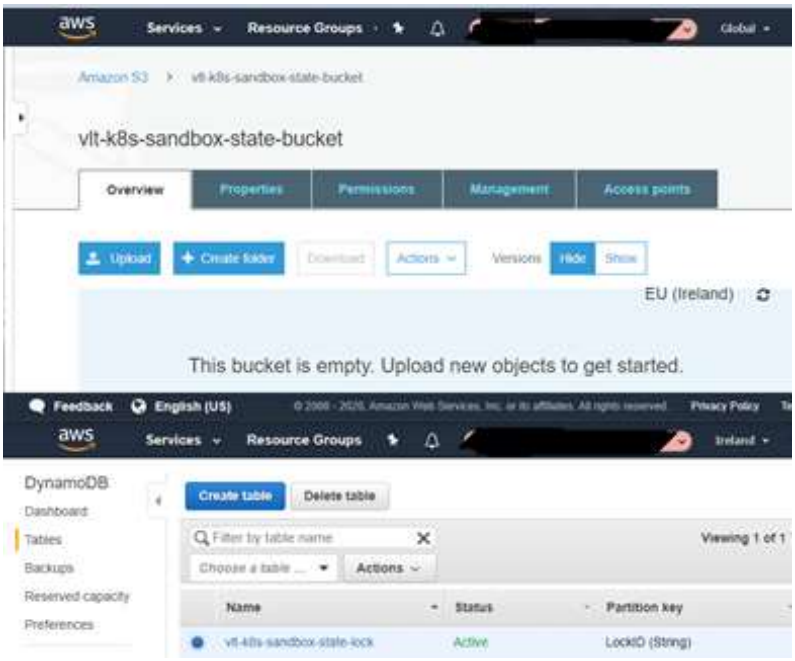
```
provider "aws" {
  region = "eu-west-1"
}

resource "aws_s3_bucket" "terraform_state_bucket" {
  bucket = "vlt-k8s-sandbox-state-bucket"

  versioning {
    enabled = true
  }

  lifecycle {
    prevent_destroy = true
  }
}

resource "aws_dynamodb_table" "terraform_state_lock" {
  name = "vlt-k8s-sandbox-state-lock"
  hash_key = "LockID"
  read_capacity = 20
  write_capacity = 20

  attribute {
    name = "LockID"
    type = "S"
  }

}
```

Kuvio 28. Terraform S3 Remote State ja DynamoDB-lukko

Terraform ajetaan *state*-hakemistosta komennolla *terraform apply* ja se tarvitsee pääsyavaimet AWS-tilille, jotka saadaan komennolla *”aws-vault exec vlt-int-demo –”.* Tilatiedoston luonti esitetty kuviossa 29. S3-bucket ja DynamoDB-taulu esitetty kuviossa 30.

```
  # aws_s3_bucket.terraform_state_bucket will be created
  + resource "aws_s3_bucket" "terraform_state_bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = "private"
      + arn                         = (known after apply)
      + bucket                      = "vlt-k8s-sandbox-state-bucket"
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + versioning {
          + enabled    = true
          + mfa_delete = false
        }
    }

Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_bucket.terraform_state_bucket: Creating...
aws_dynamodb_table.terraform_state_lock: Creating...
aws_s3_bucket.terraform_state_bucket: Creation complete after 5s [id=vlt-k8s-sandbox-state-bucket]
aws_dynamodb_table.terraform_state_lock: Creation complete after 9s [id=vlt-k8s-sandbox-state-lock]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

Kuvio 29. Terraform-tuloste S3-bucketin ja DynamoDB-taulun luomisesta



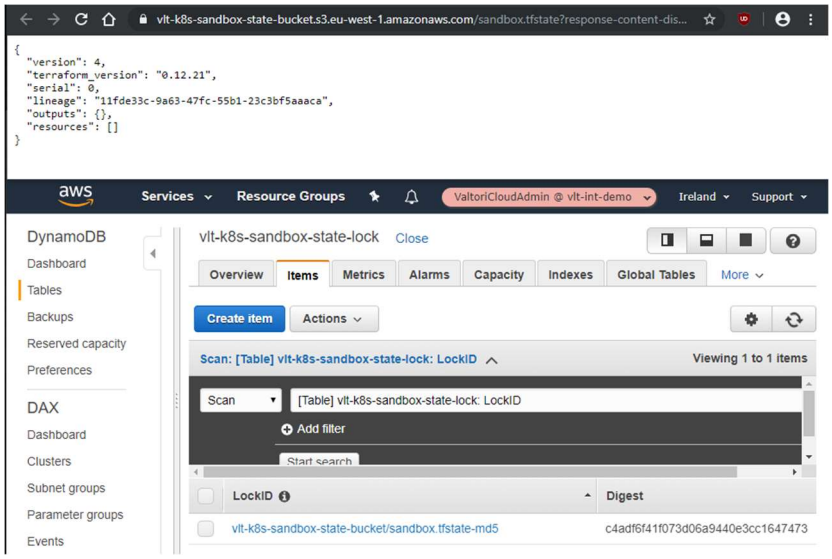
Kuvio 30. Tilatiedosto ja lukko AWS-käyttöliittymässä

Kun S3-bucket ja DynamoDB-taulu on luotu, voidaan alustaa ympäristön juurimoduuli hakemistossa vlt-k8s-sandbox. Providerin lisäksi juurimoduulille konfiguroidaan S3-bucket tilatiedoston tallentamista varten ja DynamoDB-taulu lukkoa varten. Juurimoduulin tilatiedoston konfiguraatio esitetty kuviossa 31. Kun ajetaan komento *terraform apply*, Terraform ei luo yhtään resurssia, mutta nähdään, että Terraformin tilatiedosto on S3-bucketissa ja DynamoDB-tauluun on ilmestynyt lukko. Lukko ja tilatiedosto esitetty kuviossa 32.

terraform > envs > vlt-k8s-sandbox > main.tf

```
1   provider "aws" {
2     region = "eu-west-1"
3   }
4
5   terraform {
6     backend "s3" {
7       region = "eu-west-1"
8       bucket = "vlt-k8s-sandbox-state-bucket"
9       dynamodb_table = "vlt-k8s-sandbox-state-lock"
10      key = "sandbox.tfstate"
11    }
12  }
```

Kuvio 31. Terraform-juurimoduulin tilatiedoston konfiguraatio

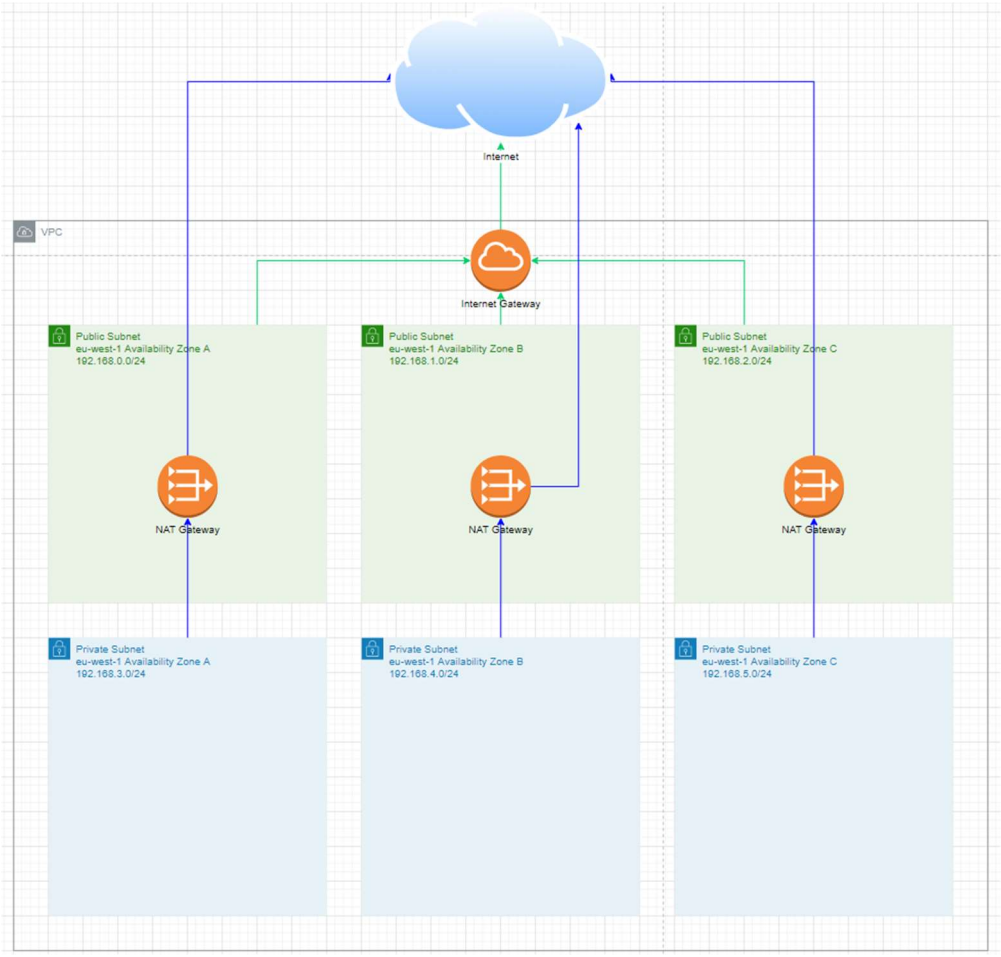


Kuvio 32. Tilatiedosto ja lukko AWS-käyttöliittymässä

Nyt kun vlt-k8s-sandbox ympäristön tilatiedosto on asetettu S3-buckettiin, voidaan ruveta luomaan itse infrastruktuuria.

4.2.2 VPC

Ensimmäisenä infrastruktuurimoduulina kannattaa luoda virtuaaliverkko VPC. VPC kannattaa luoda ensin, koska osa muusta infrastruktuurista tarvitsee valmiin VPC:n toimiakseen. VPC:stä tehdään korkeasti saatava luomalla kolme julkista ja kolme yksityistä aliverkkoa kolmelle eri saatavuusvyöhykkeelle. VPC:hen luodaan Internet Gateway, jonka kautta julkiset aliverkot reititetään internettiin. Julkisissa aliverkoissa olevat resurssit saavat myös julkisen IP-osoitteen, jolloin ne ovat myös saavutettavissa internetistä. Julkisiin aliverkkoihin luodaan NAT Gatewayt, joiden kautta yksityiset aliverkot saavat reitin internettiin. Yksityiset aliverkot eivät saa julkista IP-osoitetta eivätkä ole saavutettavissa internetistä. Kuva VPC:n arkkitehtuurista esitetty kuviossa 33.



Kuvio 33. VPC:n arkkitehtuuri

NAT Gatewayt tarvitsevat staattiset IP-osoitteet, joita varten tehdään moduuli, joka luo kolme staattista IP-osoitetta. Moduulit luodaan omiin alihakemistoihin *modules*-hakemistoon. Oletuksena AWS resurssien IP-osoite vaihtelee dynaamisesti, mutta staattinen ja pysyvä IP-osoite saadaan asettamalla resurssille Elastic IP. Elastic IP -moduuli luo kolme Elastic IP:tä ja palauttaa niiden tunnisteet output arvona, joihin VPC-moduulissa voidaan viitata. Elastic IP -moduulin konfiguraatio esitetty kuviossa 34.

rtg-eks > terraform > modules > elastic_ips > main.tf

```
variable "env_name" {
  description = "Name of the environme
  default = None
}

resource "aws_eip" "public1" {
  vpc = true
  tags = {
    "Name" = var.env_name
  }
}

resource "aws_eip" "public2" {
  vpc = true
  tags = {
    "Name" = var.env_name
  }
}

resource "aws_eip" "public3" {
  vpc = true
  tags = {
    "Name" = var.env_name
  }
}
```

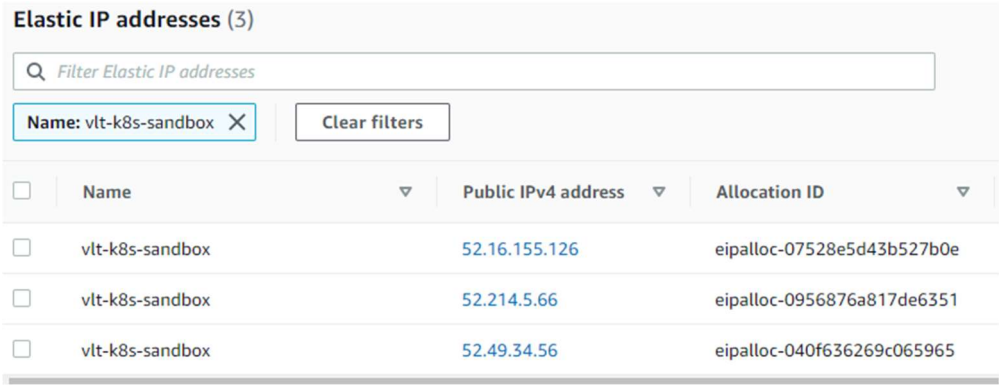
rtg-eks > terraform > modules > elastic_ips > outputs.tf

```
output "public_eips" {
  value = [aws_eip.public1.id, aws_eip.public2.id, aws_eip.public3.id]
}
```

Kuvio 34. Elastic IP -moduulin Terraform konfiguraatio

Moduuli otetaan käyttöön ympäristön vlt-k8s-sandbox Terraform-konfiguraatiossa, viittaamalla moduulin hakemistoon. Moduulin muuttujalle *env_name* annetaan arvoksi ympäristön nimi, joka leimataan moduulin luomiin resursseihin. Moduuli lisätty ympäristön juurimoduuliin kuviossa 35. Elastic IP-osoiteet AWS käyttöliittymässä esitetty kuviossa 36.

Kuvio 35. Elastic IP -moduulin käyttöönotto ympäristön juurimoduulissa



Kuvio 36. Elastic IP:t AWS:n käyttöliittymässä

Kun moduuli Elastic IP -osoitteille on tehty, voidaan kirjoittaa moduuli itse VPC:lle. Moduulissa luodaan itse VPC, jolle määritellään IP-osoiteavaruus muuttujana. VPC:hen luodaan Internet Gateway ja asetaan VPC:n (Dynamic Host Configuration Protocol) DHCP-asetukset. VPC, Internet Gateway ja DHCP-asetusten konfiguraatio esitetty kuviossa 37.

rtg-eks > terraform > modules > vpc > main.tf

```
data "aws_region" "current" {}

data "aws_availability_zones" "available" {}

resource "aws_vpc" "this" {
  cidr_block = var.cidr_block
  enable_dns_hostnames = true

  tags = {
    "kubernetes.io/cluster/EKS-Cluster" = "shared"
    "Name" = var.env_name
  }
}

resource "aws_vpc_dhcp_options" "this" {
  domain_name = "${data.aws_region.current.name}.compute.internal"

  domain_name_servers = [
    "AmazonProvidedDNS",
  ]
}

resource "aws_vpc_dhcp_options_association" "this" {
  vpc_id = aws_vpc.this.id
  dhcp_options_id = aws_vpc_dhcp_options.this.id
}

resource "aws_internet_gateway" "this" {
  vpc_id = aws_vpc.this.id
  tags = {
    "Name" = var.env_name
  }
```

Kuvio 37. VPC:n, Internet Gatewayn ja DHCP-asetusten Terraform-konfiguraatio

VPC:hen luodaan kolme julkista aliverkkoa eri saatavuusvyöhykkeille. Aliverkoille luodaan reititystaulu, jossa määritellään reitti internettiin Internet Gatewayn kautta. Julkisiin aliverkkoihin luodaan myös NAT Gatewayt, joiden kautta yksityiset aliverkot saavat reitin internettiin. Julkisten aliverkkojen konfiguraatio esitetty kuviossa 38. Yksityisten aliverkkojen konfiguraatio esitetty kuviossa 39.

rtg-eks > terraform > modules > vpc > main.tf

```
34  resource "aws_subnet" "public" {
35    count = length(var.public_subnet_cidr_blocks)
36    vpc_id = aws_vpc.this.id
37    cidr_block = var.public_subnet_cidr_blocks[count.index]
38    availability_zone = data.aws_availability_zones.available.names[count.index]
39    map_public_ip_on_launch = true
40
41    tags = {
42      "kubernetes.io/cluster/EKS-Cluster" = "shared"
43      "kubernetes.io/role/elb" = 1
44      "Name" = var.env_name
45    }
46  }
47
48  resource "aws_route_table" "public" {
49    vpc_id = aws_vpc.this.id
50
51    route {
52      cidr_block = "0.0.0.0/0"
53      gateway_id = aws_internet_gateway.this.id
54    }
55  }
56
57  resource "aws_route_table_association" "public" {
58    count = length(aws_subnet.public)
59    subnet_id = aws_subnet.public[count.index].id
60    route_table_id = aws_route_table.public.id
61  }
62
63  resource "aws_nat_gateway" "public" {
64    count = length(var.public_subnet_cidr_blocks)
65    subnet_id = aws_subnet.public[count.index].id
66    allocation_id = var.eip_ids[count.index]
67
68    depends_on = [aws_internet_gateway.this]
69    tags = {
70      "Name" = var.env_name
71    }
72  }
```

Kuvio 38. Julkisten aliverkkojen Terraform-konfiguraatio

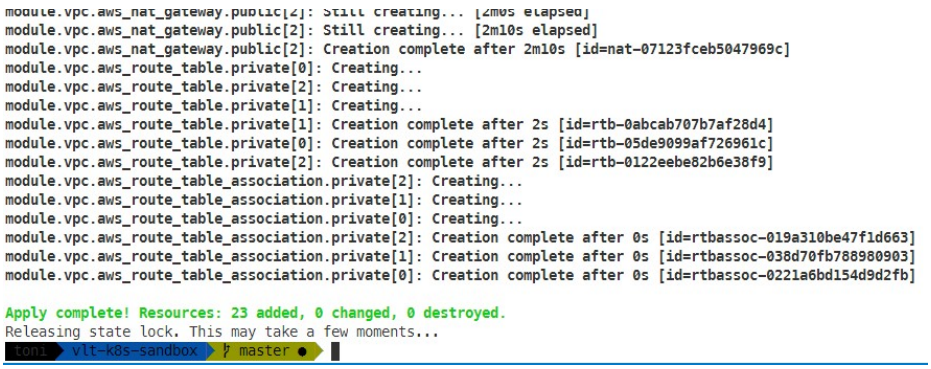
rtg-eks > terraform > modules > vpc > main.tf

```
73
74  resource "aws_subnet" "private" {
75    count = length(var.private_subnet_cidr_blocks)
76    vpc_id = aws_vpc.this.id
77    cidr_block = var.private_subnet_cidr_blocks[count.index]
78    availability_zone = data.aws_availability_zones.available.names[count.index]
79
80    tags = {
81      "kubernetes.io/cluster/EKS-Cluster" = "shared"
82      "kubernetes.io/role/internal-elb" = 1
83      "Name" = var.env_name
84    }
85  }
86
87  resource "aws_route_table" "private" {
88    count = length(aws_subnet.private)
89    vpc_id = aws_vpc.this.id
90
91    route {
92      cidr_block = "0.0.0.0/0"
93      nat_gateway_id = aws_nat_gateway.public[count.index].id
94    }
95  }
96
97  resource "aws_route_table_association" "private" {
98      count = length(aws_route_table.private)
99      subnet_id = aws_subnet.private[count.index].id
100     route_table_id = aws_route_table.private[count.index].id
101 }
```

Kuvio 39. Yksityisten aliverkkojen Terraform-konfiguraatio

VPC-moduuli otetaan käyttöön samalla tavalla ympäristön juurimoduulissa, kuin Elastic IP -moduuli. VPC-moduulin luomat NAT Gatewayt tarvitsevat Elastic IP:t käyttöönsä, joten annetaan moduulille muuttujana Elastic IP -moduulin palauttamat Elastic IP -osoitteiden tunnisteet. VPC-moduulin käyttöönotto esitetty kuviossa 40. Terraform-ajon tulos esitetty kuviossa 41.

Kuvio 40. VPC-moduulin käyttöönotto ympäristön juurimoduulissa



Kuvio 41. VPC-moduulin luonnin aiheuttama Terraform-tuloste

Kun Terraformilla on luotu VPC-moduulin resurssit, tarkistetaan AWS:n käyttöliittymästä, että resurssit varmasti luotiin. NAT Gatewayiden näkymä esitetty kuviossa 42. Aliverkkojen näkymä esitetty kuviossa 43.

Kuvio 42. NAT Gatewayt AWS:n käyttöliittymässä

search : vlt-k8s-sandbox Add filter

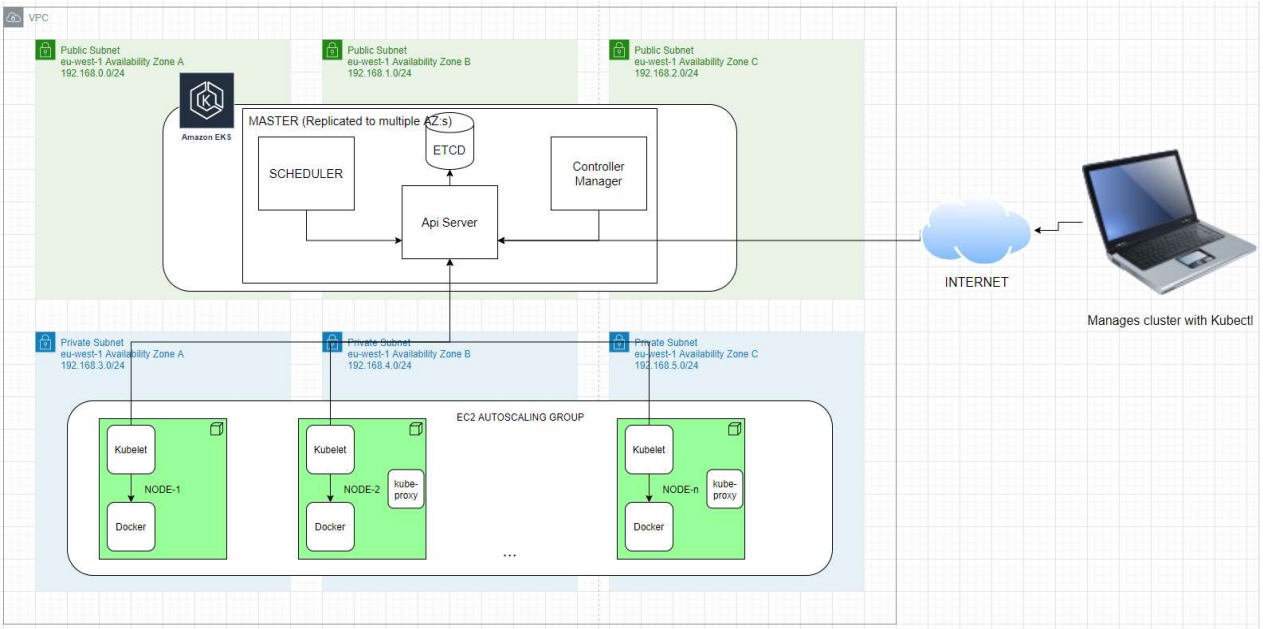
Name	Subnet ID	State	VPC	IPv4 CIDR	Ava	IPv6	Availability Zone	Availability Zone ID
vlt-k8s-sandbox-Private	subnet-079824757fb97291f	available	vpc-0398...	192.168.4.0/24	251	-	eu-west-1b	euw1-az1
vlt-k8s-sandbox-Private	subnet-07b7ea85d71b78ba2	available	vpc-0398...	192.168.3.0/24	251	-	eu-west-1a	euw1-az3
vlt-k8s-sandbox	subnet-0809c44aa50272ac1	available	vpc-0398...	192.168.1.0/24	250	-	eu-west-1b	euw1-az1
vlt-k8s-sandbox	subnet-08278d329251573b9	available	vpc-0398...	192.168.2.0/24	250	-	eu-west-1c	euw1-az2
vlt-k8s-sandbox	subnet-0b25995907e8b13ed	available	vpc-0398...	192.168.0.0/24	250	-	eu-west-1a	euw1-az3
vlt-k8s-sandbox-Private	subnet-0f9626f6e5a5b6a48	available	vpc-0398...	192.168.5.0/24	251	-	eu-west-1c	euw1-az2

Kuvio 43. Aliverkot AWS:n käyttöliittymässä

4.3 EKS

4.3.1 EKS-klusterin arkkitehtuuri

Kun ympäristölle on luotu VPC, voidaan luoda EKS-moduuli. EKS-moduulissa luodaan EKS-klusteri, joka sisältää Kubernetes-klusterin ohjaustason eli isäntänoodit. Työläisnoodeille tehdään EC2-autoskaalausryhmä, joka levittyy kolmella eri saatavuusvyöhykkeillä oleviin yksityisiin aliverkkoihin. Klusterin ohjaustaso luodaan julkisiin aliverkkoihin. EKS-klusterin arkkitehtuuri esitetty kuviossa 44.



Kuvio 44. EKS-klusterin arkkitehtuuri

4.3.2 EKS-ohjaustason ja työläisnoodien konfiguraatio

EKS-moduuli kirjoitetaan omaan alihakemistoonsa muiden moduulien tapaan. Moduuliin tulee useita komponentteja, joten on selvempää luoda jokaiselle komponentille oma tiedostonsa. EKS-klusterille luodaan IAM-rooli, jolle kiinnitetään IAM-

käytänteitä. Roolille luodaan käytänne, jossa EKS-klusterin annetaan ottaa rooli käyttöönsä. Roolille kiinnitetään myös AWS:n tekemät valmiit oletuskäytänteet AmazonEKSClusterPolicy ja AmazonEKSServicePolicy. AmazonEKSClusterPolicy sallii klusterin hallita noodien ja kuormantasaajien konfiguraatiota. AmazonEKSServicePolicy sallii klusterin mm. luoda noodeille verkkorajapintoja ja lähettää ohjaustason lokit AWS Cloudwatchiin. Itse EKS-klusterin Terraform-resurssissa määritellään klusterille nimi, aliverkot joihin klusteri luodaan, IAM-rooli ja klusterin versio. Klusteri luodaan VPC-moduulin luomiin julkisiin aliverkkoihin. IAM-rooliksi annetaan aiemmin samassa konfiguraatiotiedostossa luotu rooli, jolle on kiinnitetty sopivat käytänteet. Klusterin versio on EKS 1.16, joka käyttää Kubernetes-versiota 1.16. Klusterille luodaan myös Open ID Connect (OIDC) -provider, joka mahdollistaa IAM-roolien antamisen klusterissa ajettaville konteille. EKS-klusterin Terraform-konfiguraatio esitetty kuviossa 45.

terraform > modules > cluster > cluster.tf

```
data "aws_iam_policy_document" "cluster_assume_role" {
  statement {
    actions = [
      "sts:AssumeRole"
    ]
    principals {
      type        = "Service"
      identifiers = ["eks.amazonaws.com"]
    }
  }
}

resource "aws_eks_cluster" "cluster" {
  name     = var.cluster_name
  role_arn = aws_iam_role.cluster_role.arn
  vpc_config {
    subnet_ids = var.cluster_subnets
  }
  depends_on = [
    aws_iam_role_policy_attachment.eks-cluster-policy,
    aws_iam_role_policy_attachment.eks-service-policy,
  ]
  version = var.cluster_version
}

resource "aws_iam_role" "cluster_role" {
  name               = "eks-cluster-role"
  assume_role_policy = data.aws_iam_policy_document.cluster_assume_role.json
}

resource "aws_iam_role_policy_attachment" "eks-cluster-policy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSClusterPolicy"
  role       = aws_iam_role.cluster_role.name
}

resource "aws_iam_role_policy_attachment" "eks-service-policy" {
  policy_arn = "arn:aws:iam::aws:policy/AmazonEKSServicePolicy"
  role       = aws_iam_role.cluster_role.name
}

resource "aws_iam_openid_connect_provider" "cluster" {
  client_id_list  = ["sts.amazonaws.com"]
  thumbprint_list = [var.eks_oidc_root_ca_thumbprint]
  url             = aws_eks_cluster.cluster.identity.0.oidc.0.issuer
}
```

Kuvio 45. EKS-klusterin hallintatason Terraform-konfiguraatio

Noodeja varten luodaan NodeGroup-resurssi, joka luo työläisnoodeille autoskaalausryhmän. Noodeille luodaan myös IAM-rooli, joka liitetään NodeGroupiin. Roolille luodaan käytänne, joka sallii EC2-instanssien ottaa roolin käyttöönsä. Roolille liitetään tarvittavat oletuskäytänteet. Käytänteet sallivat noodien muokata omaa IP-konfiguraatiotaan, luoda verkkorajapintoja podeja varten ja nähdä muiden resurssien verkkokonfiguraatiot. Noodeille liitetyt käytänteet sallivat myös metriikkojen ja lokien lähettämisen Cloudwatchiin sekä Docker-imagejen hakemisen ECR-konttirekisteristä. NodeGroup-resurssissa määritellään mihin klusteriin työläisnoodit liitetään. Rooliksi annetaan aiemmin konfiguraatiossa luotu rooli. Työläisnoodit luodaan VPC-moduulin luomiin yksityisiin aliverkkoihin. Noodit on määritelty olemaan instanssityypiltään t3.small, joten yhdellä noodilla on 2 prosessoriydintä ja 2 gigatavua muistia. AMI eli Amazon Machine Image on määritetty olemaan Amazonin tarjoama virtuaalikonekuva, jolle on konfiguroitu valmiiksi työläisnoodin komponentit. Noodien käyttöjärjestelmä on Amazon Linux 2, joka on Amazonin oma Red Hat Linuxiin pohjautuva Linux-jakelu. NodeGroupille on asetettu skaalautuvuuskonfiguraatiossa minimisuuruudeksi 1 noodi. Noodien ihannemääräksi on asetettu 3 ja maksimääräksi 6. Työläisnoodien konfiguraatio esitetty kuviossa 46.

```
resource "aws_eks_node_group" "node_group" {
  cluster_name    = var.cluster_name
  node_group_name = "${aws_eks_cluster.cluster.name}-node_group"
  node_role_arn   = aws_iam_role.node_role.arn
  subnet_ids      = var.worker_subnets
  instance_types  = var.instance_types
  ami_type = data.aws_ami.eks_worker.image_id
  disk_size       = 20
  scaling_config {
    desired_size = 3
    max_size     = 6
    min_size     = 1
  }
  depends_on = [
    aws_iam_role_policy_attachment.EKSWorkerNodeAttachment,
    aws_iam_role_policy_attachment.EKSCNIAttachment,
    aws_iam_role_policy_attachment.ECRReadOnlyAttachment,
  ]
}
```

Kuvio 46. Työläisnoodien Terraform-konfiguraatio

Noodien ja klusterin lisäksi klusterille on luotava Kubernetesin ConfigMap-objekti, jossa AWS-tilin IAM-rooli asetetaan klusterissa masters-ryhmään. ConfigMapissa

myös noodien IAM-rooli liitetään nodes- ja bootstrappers-ryhmiin. Kubernetes resurssien luomista varten on konfiguroitava Terraformissa Kubernetes-provider. Provider tarvitsee Kubernetes-klusterin Apiserverin osoitteen, klusterin CA-sertifikaatin ja tokenin, jolla omaksua kirjoitusoikeudet klusteriin. Arvot edellä mainittuihin argumentteihin saadaan viittaamalla EKS-klusterin attribuutteihin. Kubernetes-providerin konfiguraatio ja aws-auth ConfigMap esitetty kuviossa 47.

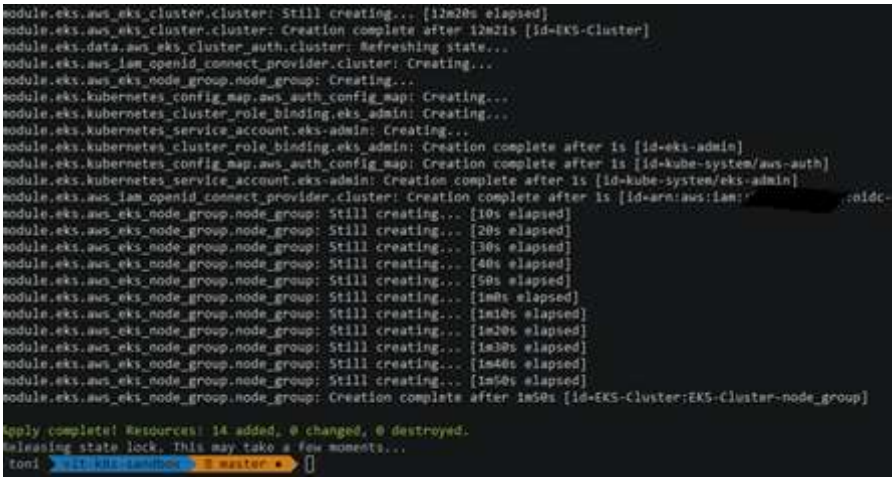
```
data "aws_eks_cluster_auth" "cluster" {
  name = aws_eks_cluster.cluster.name
}

provider "kubernetes" {
  host                   = aws_eks_cluster.cluster.endpoint
  cluster_ca_certificate = base64decode(aws_eks_cluster.cluster.certificate_authority.0.data)
  token                  = data.aws_eks_cluster_auth.cluster.token
  load_config_file       = false
}

resource "kubernetes_config_map" "aws_auth_config_map" {
  metadata {
    name      = "aws-auth"
    namespace = "kube-system"
  }
  data = {
    mapRoles = <<-YAML
      - rolearn: ${aws_iam_role.node_role.arn}
        username: system:node:{{EC2PrivateDNSName}}
        groups:
          - system:bootstrappers
          - system:nodes

      - rolearn: arn:aws:iam::${data.aws_caller_identity.current.account_id}:role/[redacted]
        username: [redacted]
        groups:
          - system:masters
    YAML
  }
}
```

Kuvio 47. Kubernetes-providerin ja aws-auth-config-mapin konfiguraatiot

Lisätään EKS-moduuli ympäristön juurimoduuliin ja asetetaan VPC-moduulin palauttamat aliverkkojen tunnisteet moduuliin muuttujina. EKS-moduulin lisääminen ympäristön juurimoduuliin esitetty kuviossa 48. Luodaan EKS-moduulin resurssit komennolla *terraform apply*. *Terraform apply* -komennon tuloste esitetty kuviossa 49.

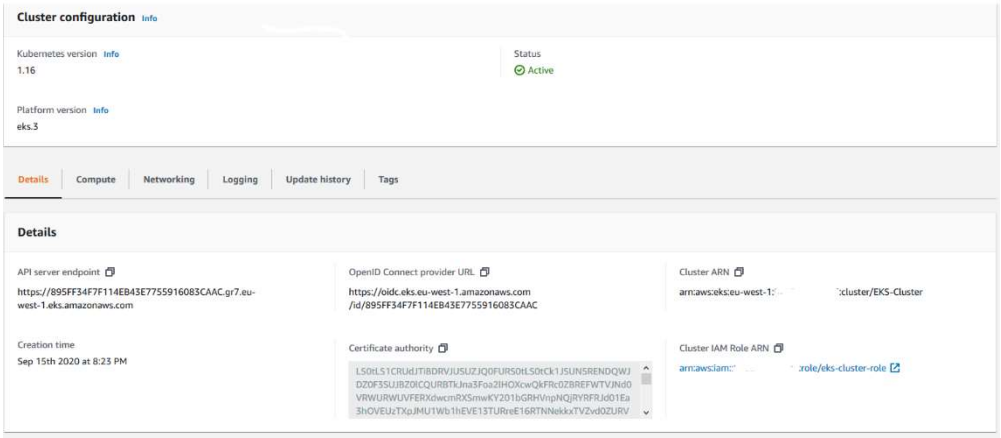
```
module "vpc" {
  source   = "../../modules/vpc"
  eip_ids  = module.elastic_ips.public_eips
  env_name = "vlt-k8s-sandbox"
}

module "eks" {
  source                = "../../modules/cluster"
  cluster_subnets       = module.vpc.public_subnet_ids
  worker_subnets        = module.vpc.private_subnet_ids
}
```

Kuvio 48. EKS-moduulin käyttöönotto ympäristön juurimoduulissa



Kuvio 49. EKS-moduulin luomisen aiheuttama Terraform tuloste

Terraformin ajamisen jälkeen tarkistetaan AWS:n käyttöliittymästä, että klusteri ja noodit luotiin. Kuviossa 50 on esitetty EKS-klusteri ja NodeGroup konfiguraatioineen AWS:n käyttöliittymässä.



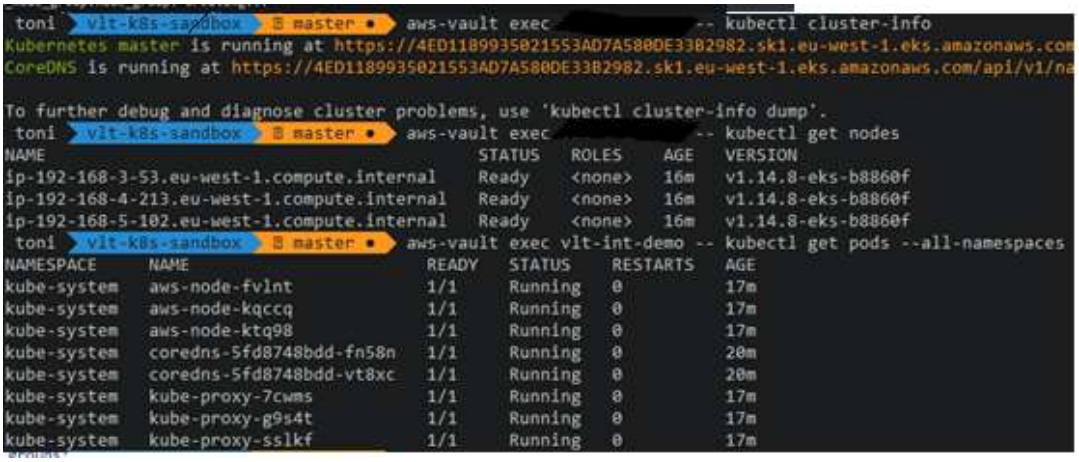
Kuvio 50. EKS-klusteri AWS:n käyttöliittymässä

Kun klusteri on luotu, voidaan klusterille ajaa komentoja kubectl-työkalulla. Kubectl konfiguroidaan käyttämään luotua klusteria *aws eks update-config* -komennolla (kts. kuvio 51).

Kuvio 51. Kubectl-työkalun konfigurointi käyttämään luotua klusteria

Varmistetaan että saadaan hallintayhteys klusteriin ajamalla komento *kubectl cluster-info,* joka palauttaa API-serverin ja CoreDNS:n endpointit. Komennolla *kubectl get*

nodes, nähdään että klusterissa on kolme noodia valmiina isännöimään podeja. Komennolla *kubectl get pods*, nähdään klusterin toiminnan kannalta välttämättömät oletuspodit kuten KubeProxy ja Core DNS toimintakunnossa. Komennot, joilla varmistetaan klusterin toimivuus esitetty kuviossa 52.



Kuvio 52. Klusterin toimivuuden varmistus

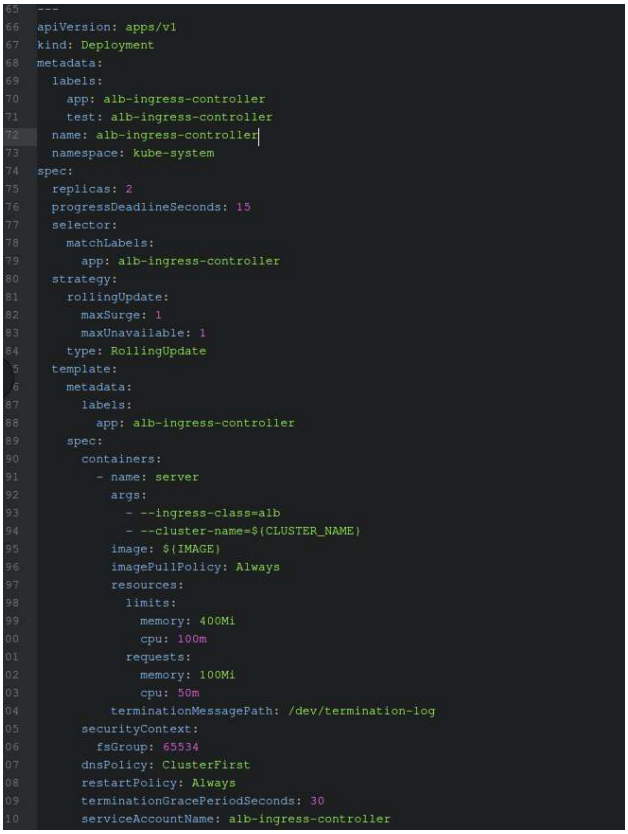
4.3.3 Kontrollerien käyttönotto

EKS-klusterin oletuskonfiguraatioon halutaan seuraavat kontrollerit:

- AWS ALB Ingress Controller
- External DNS
- Cloudwatch Agent
- Fluent-bit
- Cluster Autoscaler
- Metrics Server

Kaikki yllälistatut kontrollerit otetaan käyttöön Kubernetes-klusterissa ajettavina kontteina. YAML-muotoiset Kubernetes-konfiguraatiot on sijoitettu EKS-moduulin alle omaan hakemistoonsa.

Metrics Server, Cluster Autoscaler, External DNS ja AWS ALB Ingress Controller otetaan käyttöön Deploymenteina. Fluent-bit ja Cloudwatch Agent otetaan käyttöön DaemonSetienä, jolla varmistetaan, että jokaisella noodilla on ajossa yksi podin replika. Esimerkkinä AWS ALB:n Deployment-konfiguraatio esitetty Kuviossa 53. Esimerkki Fluent-bitin DaemonSet-konfiguraatiosta esitetty kuviossa 54.



```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: alb-ingress-controller
    test: alb-ingress-controller
  name: alb-ingress-controller
  namespace: kube-system
spec:
  replicas: 2
  progressDeadlineSeconds: 15
  selector:
    matchLabels:
      app: alb-ingress-controller
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: alb-ingress-controller
    spec:
      containers:
        - name: server
          args:
            - --ingress-class=alb
            - --cluster-name=$(CLUSTER_NAME)
          image: $(IMAGE)
          imagePullPolicy: Always
          resources:
            limits:
              memory: 400Mi
              cpu: 100m
            requests:
              memory: 100Mi
              cpu: 50m
          terminationMessagePath: /dev/termination-log
      securityContext:
        fsGroup: 65534
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      serviceAccountName: alb-ingress-controller
```

Kuvio 53. ALB Ingress Controllerin Deployment-konfiguraatio

```
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentbit
  namespace: kube-system
  labels:
    app.kubernetes.io/name: fluentbit
spec:
  selector:
    matchLabels:
      name: fluentbit
  template:
    metadata:
      labels:
        name: fluentbit
    spec:
      serviceAccountName: fluent-bit
      priorityClassName: system-node-critical
      containers:
      - name: aws-for-fluent-bit
        image: $(IMAGE)
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
        - name: fluent-bit-config
          mountPath: /fluent-bit/etc/
        - name: mnt
          mountPath: /mnt
          readOnly: true
        resources:
          limits:
            memory: 500Mi
          requests:
            cpu: 500m
            memory: 100Mi
      securityContext:
        fsGroup: 65534
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
      - name: fluent-bit-config
        configMap:
          name: fluent-bit-config
```

Kuvio 54. Fluent-bitin DaemonSet-konfiguraatio

Kaikille kontrollereille on tehty klusterin sisäiset roolit, jotka ovat liitetty kontrollereiden palvelutileihin, joita kukin kontrolleri käyttävät klusterin sisällä. Esimerkkinä Fluent-bitin palvelutili ja rooli esitetty kuviossa 55.

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: fluent-bit
  namespace: kube-system
  annotations:
    eks.amazonaws.com/role-arn: ${SERVICE_ACCOUNT_IAM_ROLE_ARN}
  labels:
    app: fluent-bit
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: pod-log-reader
rules:
- apiGroups: [""]
  resources:
  - namespaces
  - pods
  verbs: ["get", "list", "watch"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: pod-log-crb
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: pod-log-reader
subjects:
- kind: ServiceAccount
  name: fluent-bit
  namespace: kube-system
```

Kuvio 55. Fluent-bitin ServiceAccount ja ClusterRole

Jokaisen kontrollerin Kubernetes konfiguraatio on sisällytetty EKS-moduulin Terraform-konfiguraatioon käyttämällä null resourcea. Null resource ajaa kubectl-komentona kontrollerin konfiguraatiotiedoston Kubernetes klusteriin. Null resourcelle määritellään liipaisimet, joiden muuttuessa Terraform päivittää kontrollerin ajamalla kubectl-komennon uudestaan. Arvot, jotka laukaisevat päivityksen ovat konfiguraatiotiedoston sha256-tarkastussumman muuttuminen ja uusi versio kontrollerin kontista. Null resource konfiguroi kubectl:n käyttämään EKS-Klusterin endpointtia Terraform-resursseista saaduilla attribuuteilla, jotka syötetään Kubernetesin Config-objektiin muuttujina. Kontrollerien käyttämät kontti-imaget on määritelty muuttujina EKS-moduulin variables.tf tiedostossa (ks kuvio 56). Esimerkkinä External DNS:n null resource esitetty kuviossa 57. Klusterin endpointin konfiguroiminen Terraform null resourcea varten esitetty kuviossa 58.

```
resource "null_resource" "kube_external_dns" {
  triggers = {
    deployment     = sha256(file("${path.module}/files/${local.external_dns_config}"))
    kubeconfig     = sha256(local.kubeconfig)
    image          = var.external_dns_image
    sa_arn         = aws_iam_role.external_dns_sa.arn
    domain_filters = yamlencode(formatlist("--domain-filter=%s", var.route_53_domains))
  }

  provisioner "local-exec" {
    command     = <<-EOS
      envsubst < ${local.external_dns_config} | kubectl apply --kubeconfig <(echo '${base64encode(local.kubeconfig)}' | base64 --decode) -f -
    EOS
    interpreter = ["/bin/bash", "-c"]
    working_dir = "${path.module}/files"
    environment = {
      SERVICE_ACCOUNT_IAM_ROLE_ARN = aws_iam_role.external_dns_sa.arn
      IMAGE                        = var.external_dns_image
      CLUSTER_NAME                 = var.cluster_name
      DOMAIN_FILTERS               = indent(10, join("\n", formatlist("- --domain-filter=%s", var.route_53_domains)))
    }
  }
}
```

Kuvio 56. External DNS:n null resource

kubeconfig.tpl

```
variable "kubeconfig_name" {
  default = ""
}
variable "kubeconfig_aws_authenticator_command" {
  description = "Command to use to fetch AWS EKS credentials."
  type        = string
  default     = "aws-iam-authenticator"
}

variable "kubeconfig_aws_authenticator_command_args" {
  description = "Default arguments passed to the authenticator comm
  type        = list(string)
  default     = []
}

variable "kubeconfig_aws_authenticator_additional_args" {
  description = "Any additional arguments to pass to the authentica
  type        = list(string)
  default     = []
}

variable "kubeconfig_aws_authenticator_env_variables" {
  description = "Environment variables that should be used when exe
  type        = map(string)
  default     = {}
}

locals {
  kubeconfig_name = var.kubeconfig_name == "" ? "eks_${var.cluster_na

  kubeconfig = true ? templatefile("${path.module}/templates/kubeconfi
    kubeconfig_name                  = local.kubeconfig_name
    endpoint                         = aws_eks_cluster.cluster.end
    cluster_auth_base64              = aws_eks_cluster.cluster.cer
    aws_authenticator_command        = var.kubeconfig_aws_authentic
    aws_authenticator_command_args   = length(var.kubeconfig_aws_au
    aws_authenticator_additional_args = var.kubeconfig_aws_authentic
    aws_authenticator_env_variables  = var.kubeconfig_aws_authentic
  }) : ""
}
```

kubeconfig.tpl

```
apiVersion: v1
preferences: {}
kind: Config

clusters:
- cluster:
    server: ${endpoint}
    certificate-authority-data: ${cluster_auth_base64}
  name: ${kubeconfig_name}

contexts:
- context:
    cluster: ${kubeconfig_name}
    user: ${kubeconfig_name}
  name: ${kubeconfig_name}

current-context: ${kubeconfig_name}

users:
- name: ${kubeconfig_name}
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      command: ${aws_authenticator_command}
      args:
%{~ for i in aws_authenticator_command_args }
        - "${i}"
%{~ endfor ~}
%{ for i in aws_authenticator_additional_args }
        - ${i}
%{~ endfor ~}
%{ if length(aws_authenticator_env_variables) > 0 }
      env:
  %{~ for k, v in aws_authenticator_env_variables ~}
        - name: ${k}
          value: ${v}
  %{~ endfor ~}
%{ endif }
```

Kuvio 57. Null resourcejen konfiguroiminen käyttämään EKS-klusterin endpointtia

```
variable "aws_alb_ingress_controller_image" {
  type        = string
  description = "ALB Ingress Contorller Image from Dockerhub"
  default     = "amazon/aws-alb-ingress-controller:v1.1.9"
}

variable "external_dns_image" {
  type        = string
  description = "External DNS Docker Image"
  default     = "registry.opensource.zalan.do/teapot/external-dns:v0.7.3"
}

variable "cloudwatch_agent_image" {
  type        = string
  description = "Cloudwatch Agent Docker Image from Dockerhub"
  default     = "amazon/cloudwatch-agent:1.246396.0"
}

variable "fluent_bit_image" {
  type        = string
  description = "Fluentd docker image"
  default     = "amazon/aws-for-fluent-bit:2.6.1"
}

variable "cluster_autoscaler_image" {
  type = string
  description = "cluster_autoscaler_image"
  default = "eu.gcr.io/k8s-artifacts-prod/autoscaling/cluster-autoscaler:v1.16.5"
}

variable "metric_server_image" {
  type = string
  default = "k8s.gcr.io/metrics-server-amd64:v0.3.6"
}
```

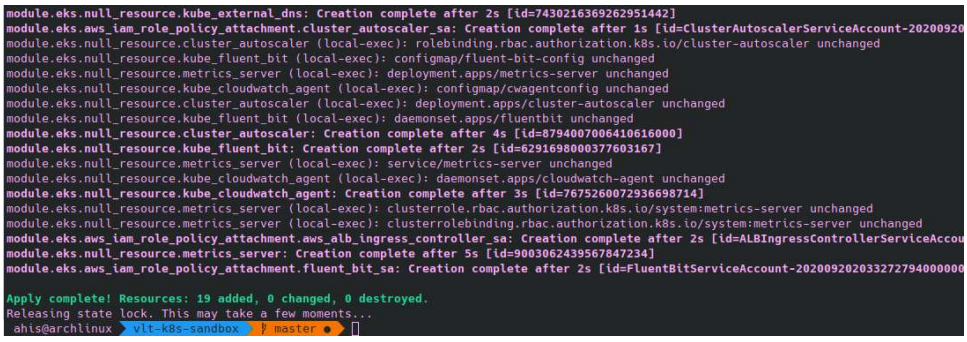
Kuvio 58. Kontrollerien kontti-imaget Terraform-muuttujina

Kontrollerit tarvitsevat vielä AWS IAM–roolin, jotta ne voivat suorittaa tehtäviään klusterin ulkopuolella. Cluster Autoscaler tarvitsee oikeuden autoskaalausryhmän ihannekoon muuttamiseen, External DNS nimipalvelutietueiden hallintaan, Ingress Controller kuormantasaajien hallitsemiseen, Cloudwatch Agent metriikoiden julkaisemiseen Cloudwatchiin ja Fluent-bit lokivirtojen kirjoittamiseen Cloudwatchiin. Kontrollereille luodaan AWS IAM-roolit tarvittavin käytänteineen, joihin kontrollereiden palvelutileille annetaan omaksumisoikeudet. Esimerkkinä Cluster Autoscalerin IAM–roolin konfiguraatio esitetty kuviossa 59.

```
resource "aws_iam_policy" "ClusterAutoscaler" {
  name   = "ClusterAutoscaler"
  policy = data.aws_iam_policy_document.ClusterAutoscaler.json
}
resource "aws_iam_role" "cluster_autoscaler_sa" {
  name               = "ClusterAutoscalerServiceAccount"
  assume_role_policy = data.aws_iam_policy_document.cluster_autoscaler_sa.json
}
resource "aws_iam_role_policy_attachment" "cluster_autoscaler_sa" {
  role       = aws_iam_role.cluster_autoscaler_sa.name
  policy_arn = aws_iam_policy.ClusterAutoscaler.arn
}
resource "aws_iam_role_policy_attachment" "cluster_autoscaler" {
  role = aws_iam_role.node_role.name
  policy_arn = aws_iam_policy.ClusterAutoscaler.arn
}
data "aws_iam_policy_document" "cluster_autoscaler_sa" {
  version = "2012-10-17"
  statement {
    effect = "Allow"
    principals {
      identifiers = [aws_iam_openid_connect_provider.cluster.arn]
      type        = "Federated"
    }
    actions = ["sts:AssumeRoleWithWebIdentity"]
    condition {
      test     = "StringEquals"
      values   = ["system:serviceaccount:kube-system:cluster-autoscaler"]
      variable = "${split("//", aws_eks_cluster.cluster.identity[0].oidc.0.issuer)[1]}:sub"
    }
  }
}

data "aws_iam_policy_document" "ClusterAutoscaler" {
  statement {

    actions = [
      "autoscaling:DescribeAutoScalingGroups",
      "autoscaling:DescribeAutoScalingInstances",
      "autoscaling:DescribeLaunchConfigurations",
      "autoscaling:DescribeTags",
      "autoscaling:SetDesiredCapacity",
      "autoscaling:TerminateInstanceInAutoScalingGroup",
      "ec2:DescribeLaunchTemplateVersions"
    ]

    resources = [
      "*",
    ]
  }
}
```

Kuvio 59. Cluster Autoscalerin AWS IAM -rooli

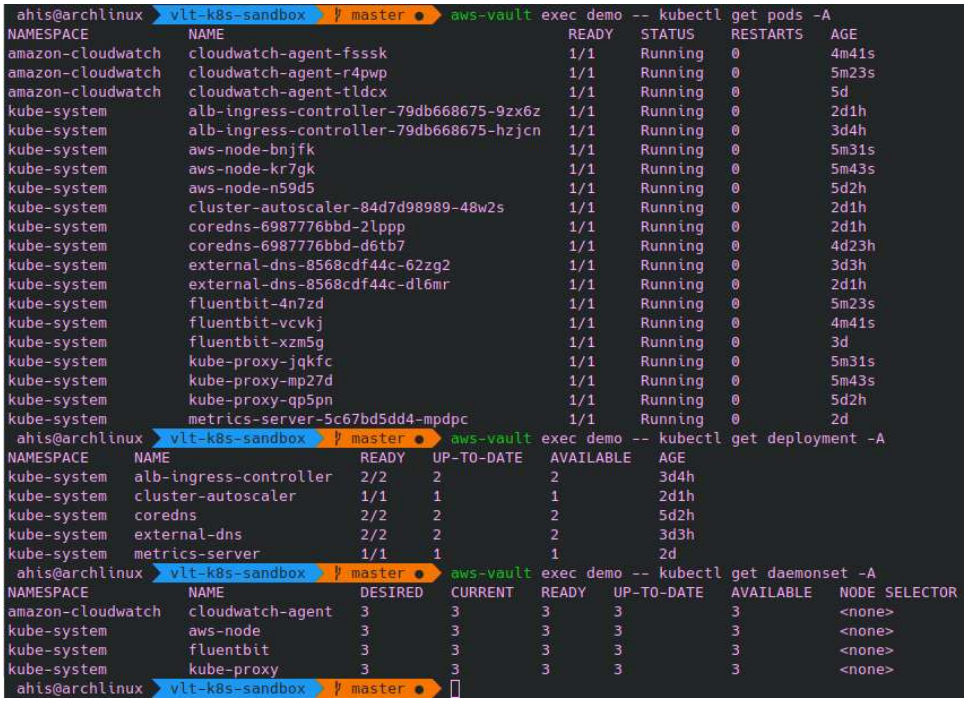
Kontrollereiden Kubernetes- ja Terraform-konfiguraatiot ovat paikallaan ja kontrollerit otetaan käyttöön ajamalla *kubectl apply* -komento ympäristön juurimoduulin hakemistossa. Kontrollerien luomisen aiheuttama Terraform–tuloste kuviossa 61.



```
module.eks.null_resource.kube_external_dns: Creation complete after 2s [id=7430216369262951442]
module.eks.aws_iam_role_policy_attachment.cluster_autoscaler_sa: Creation complete after 1s [id=ClusterAutoscalerServiceAccount-20200920
module.eks.null_resource.cluster_autoscaler (local-exec): rolebinding.rbac.authorization.k8s.io/cluster-autoscaler unchanged
module.eks.null_resource.kube_fluent_bit (local-exec): configmap/fluent-bit-config unchanged
module.eks.null_resource.metrics_server (local-exec): deployment.apps/metrics-server unchanged
module.eks.null_resource.kube_cloudwatch_agent (local-exec): configmap/cwagentconfig unchanged
module.eks.null_resource.cluster_autoscaler (local-exec): deployment.apps/cluster-autoscaler unchanged
module.eks.null_resource.kube_fluent_bit (local-exec): daemonset.apps/fluentbit unchanged
module.eks.null_resource.cluster_autoscaler: Creation complete after 4s [id=8794007006410616000]
module.eks.null_resource.kube_fluent_bit: Creation complete after 2s [id=6291698000377603167]
module.eks.null_resource.metrics_server (local-exec): service/metrics-server unchanged
module.eks.null_resource.kube_cloudwatch_agent (local-exec): daemonset.apps/cloudwatch-agent unchanged
module.eks.null_resource.kube_cloudwatch_agent: Creation complete after 3s [id=7675260072936698714]
module.eks.null_resource.metrics_server (local-exec): clusterrole.rbac.authorization.k8s.io/system:metrics-server unchanged
module.eks.null_resource.metrics_server (local-exec): clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server unchanged
module.eks.aws_iam_role_policy_attachment.aws_alb_ingress_controller_sa: Creation complete after 2s [id=ALBIngressControllerServiceAccou
module.eks.null_resource.metrics_server: Creation complete after 5s [id=9003062439567847234]
module.eks.aws_iam_role_policy_attachment.fluent_bit_sa: Creation complete after 2s [id=FluentBitServiceAccount-2020092020332727940000000

Apply complete! Resources: 19 added, 0 changed, 0 destroyed.
Releasing state lock. This may take a few moments...
 ahis@archlinux  vlt-k8s-sandbox  master ●
```

Kuvio 60. Kontrollerien luomisen aiheuttama Terraform-tuloste

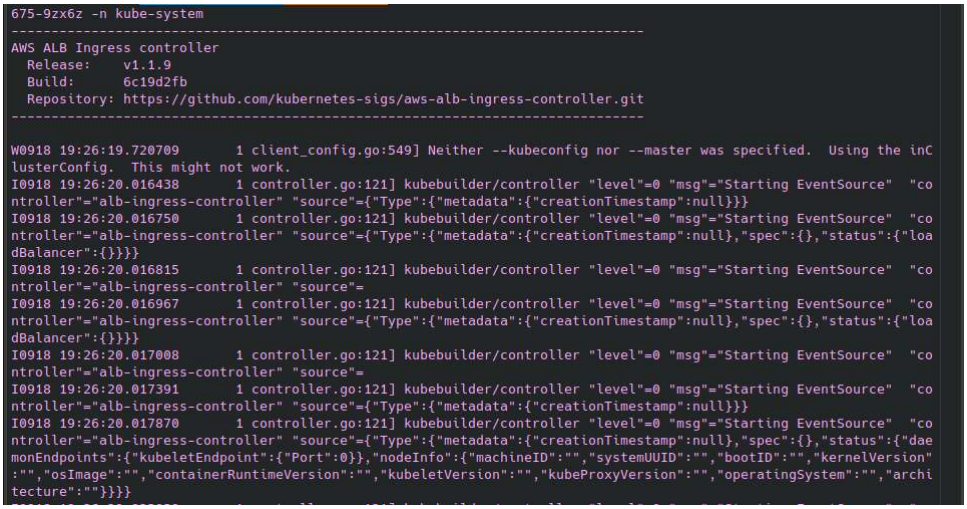
Kun Terraform on luonut resurssit, tarkastetaan klusterista, että Podit ovat running-tilassa. Klusterissa olevat Podit, Deploymentit ja DaemonSetit esitetty kuviossa 62.



```
 ahis@archlinux  vlt-k8s-sandbox  master ●  aws-vault exec demo -- kubectl get pods -A
NAMESPACE           NAME                                       READY   STATUS    RESTARTS   AGE
amazon-cloudwatch   cloudwatch-agent-fsssk                     1/1     Running   0          4m41s
amazon-cloudwatch   cloudwatch-agent-r4pwp                     1/1     Running   0          5m23s
amazon-cloudwatch   cloudwatch-agent-tldcx                     1/1     Running   0          5d
kube-system         alb-ingress-controller-79db668675-9zx6z    1/1     Running   0          2d1h
kube-system         alb-ingress-controller-79db668675-hzjcn    1/1     Running   0          3d4h
kube-system         aws-node-bnjfk                             1/1     Running   0          5m31s
kube-system         aws-node-kr7gk                             1/1     Running   0          5m43s
kube-system         aws-node-n59d5                             1/1     Running   0          5d2h
kube-system         cluster-autoscaler-84d7d98989-48w2s        1/1     Running   0          2d1h
kube-system         coredns-6987776bbd-2lppp                   1/1     Running   0          2d1h
kube-system         coredns-6987776bbd-d6tb7                   1/1     Running   0          4d23h
kube-system         external-dns-8568cdf44c-62zg2              1/1     Running   0          3d3h
kube-system         external-dns-8568cdf44c-dl6mr              1/1     Running   0          2d1h
kube-system         fluentbit-4n7zd                            1/1     Running   0          5m23s
kube-system         fluentbit-vcvkj                            1/1     Running   0          4m41s
kube-system         fluentbit-xzm5g                            1/1     Running   0          3d
kube-system         kube-proxy-jqkfc                           1/1     Running   0          5m31s
kube-system         kube-proxy-mp27d                           1/1     Running   0          5m43s
kube-system         kube-proxy-qp5pn                           1/1     Running   0          5d2h
kube-system         metrics-server-5c67bd5dd4-mpdpc            1/1     Running   0          2d
 ahis@archlinux  vlt-k8s-sandbox  master ●  aws-vault exec demo -- kubectl get deployment -A
NAMESPACE     NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
kube-system   alb-ingress-controller   2/2     2            2           3d4h
kube-system   cluster-autoscaler       1/1     1            1           2d1h
kube-system   coredns                  2/2     2            2           5d2h
kube-system   external-dns             2/2     2            2           3d3h
kube-system   metrics-server           1/1     1            1           2d
 ahis@archlinux  vlt-k8s-sandbox  master ●  aws-vault exec demo -- kubectl get daemonset -A
NAMESPACE           NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
amazon-cloudwatch   cloudwatch-agent  3         3         3       3            3           <none>
kube-system         aws-node          3         3         3       3            3           <none>
kube-system         fluentbit         3         3         3       3            3           <none>
kube-system         kube-proxy        3         3         3       3            3           <none>
 ahis@archlinux  vlt-k8s-sandbox  master ●
```

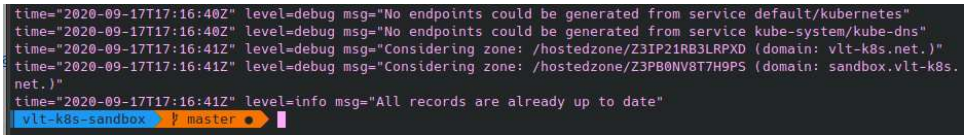
Kuvio 61. Klusterin podit, deploymentit ja daemonsetit

Podien toimivuus tarkastetaan katsomalla *kubectl logs* -komennolla podien lokeja. Ingress Controllerin lokissa huomataan, että podi lokittaa tyhjiä JSON-objekteja. Tämä johtuu siitä, ettei klusterissa ole vielä yhtään Ingressiä eikä näin ollen hallittavaa kuormantasaajaa (ks. Kuvio 63). External DNS -podin lokeista huomataan, että podi tarkkailee nimipalvelutietueita ja toteaa, että ne ovat ajan tasalla, koska hallittavia tietueita ei ole (ks. Kuvio 64).



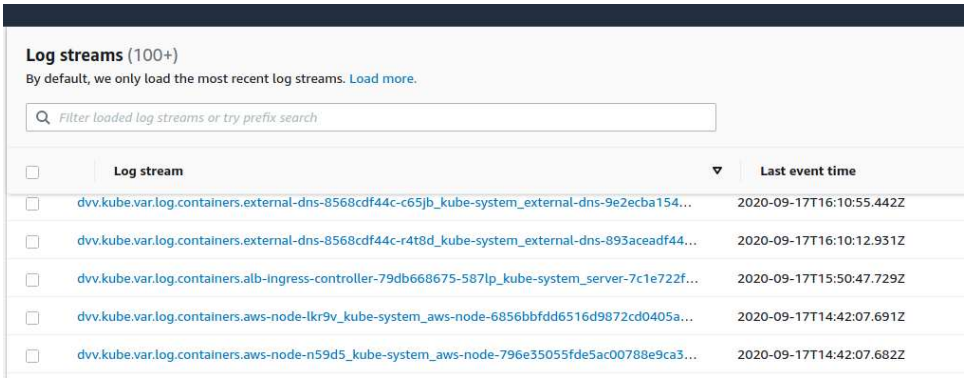
```
675-9zx6z -n kube-system
--------------------------------------------------------------------------------
AWS ALB Ingress controller
  Release:    v1.1.9
  Build:      6c19d2fb
  Repository: https://github.com/kubernetes-sigs/aws-alb-ingress-controller.git
--------------------------------------------------------------------------------

W0918 19:26:19.720709       1 client_config.go:549] Neither --kubeconfig nor --master was specified.  Using the inC
lusterConfig.  This might not work.
I0918 19:26:20.016438       1 controller.go:121] kubebuilder/controller "level"=0 "msg"="Starting EventSource"  "co
ntroller"="alb-ingress-controller" "source"={"Type":{"metadata":{"creationTimestamp":null}}}
I0918 19:26:20.016750       1 controller.go:121] kubebuilder/controller "level"=0 "msg"="Starting EventSource"  "co
ntroller"="alb-ingress-controller" "source"={"Type":{"metadata":{"creationTimestamp":null},"spec":{},"status":{"loa
dBalancer":{}}}}
I0918 19:26:20.016815       1 controller.go:121] kubebuilder/controller "level"=0 "msg"="Starting EventSource"  "co
ntroller"="alb-ingress-controller" "source"=
I0918 19:26:20.016967       1 controller.go:121] kubebuilder/controller "level"=0 "msg"="Starting EventSource"  "co
ntroller"="alb-ingress-controller" "source"={"Type":{"metadata":{"creationTimestamp":null},"spec":{},"status":{"loa
dBalancer":{}}}}
I0918 19:26:20.017008       1 controller.go:121] kubebuilder/controller "level"=0 "msg"="Starting EventSource"  "co
ntroller"="alb-ingress-controller" "source"=
I0918 19:26:20.017391       1 controller.go:121] kubebuilder/controller "level"=0 "msg"="Starting EventSource"  "co
ntroller"="alb-ingress-controller" "source"={"Type":{"metadata":{"creationTimestamp":null}}}
I0918 19:26:20.017870       1 controller.go:121] kubebuilder/controller "level"=0 "msg"="Starting EventSource"  "co
ntroller"="alb-ingress-controller" "source"={"Type":{"metadata":{"creationTimestamp":null},"spec":{},"status":{"dae
monEndpoints":{"kubeletEndpoint":{"Port":0}},"nodeInfo":{"machineID":"","systemUUID":"","bootID":"","kernelVersion"
:"","osImage":"","containerRuntimeVersion":"","kubeletVersion":"","kubeProxyVersion":"","operatingSystem":"","archi
tecture":""}}}}
```

Kuvio 62. ALB Ingress Controllerin loki

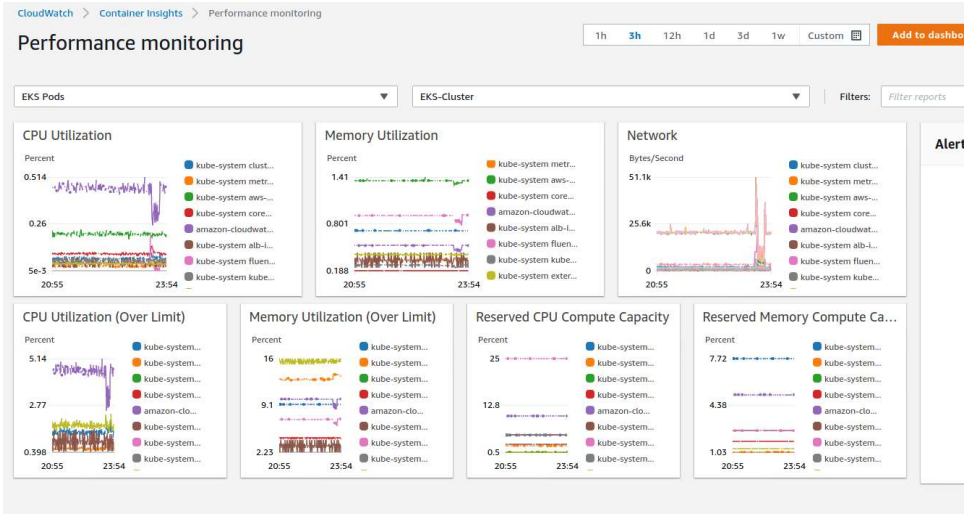


Kuvio 63. External DNS:n loki

Fluent-bitin toimivuus tarkastetaan katsomalla Cloudwatch Logsista ilmestyykö klusterin podeista lokivirtoja. Podien lokivirrat Cloudwatch Logsissa esitetty kuviossa 65. Cloudwatch Agentin toimivuus voidaan tarkastaa katsomalla Cloudwatchista ilmestyykö klusterista metriikkaa Cloudwatchiin. EKS-klusterin podien metriikkaa Cloudwatchissa esitetty kuviossa 66.



Kuvio 64. Lokivirrat Cloudwatch Logsin käyttöliittymässä



Kuvio 65. Klusterin podien metriikat Cloudwatch-valvontanäytöllä

Cluster Autoscalerin lokista nähdään, että se tarkkaillee klusterin kuormaa ja on jo poistanut tarpeettomia noodeja (ks Kuvio 67). Metrics Serverin lokista huomataan,

että se on saatavilla portissa 443, mutta ei ole tuottanut vielä muita lokimerkintöjä, koska Horizontal Pod Autoscalereita ei ole vielä käytössä (ks. Kuvio 68).

```
I0918 19:26:22.131707       1 scale_down.go:1162] All pods removed from ip-192-168-5-160.eu-west-1.compute.internal
I0918 19:26:22.314745       1 auto_scaling_groups.go:279] Terminating EC2 instance: i-0bfb9573a98404a09
I0918 19:26:22.314764       1 aws_manager.go:291] Some ASG instances might have been deleted, forcing ASG list refresh
I0918 19:26:22.382197       1 auto_scaling_groups.go:354] Regenerating instance to ASG map for ASGs: [eks-b6ba496c-2896-7ead-9133-fbed9678a237]
I0918 19:26:22.449830       1 aws_manager.go:263] Refreshed ASG list, next refresh after 2020-09-18 19:27:22.449821918 +0000 UTC m=+732.934167145
I0918 19:26:22.449990       1 event.go:255] Event(v1.ObjectReference{Kind:"Node", Namespace:"", Name:"ip-192-168-5-160.eu-west-1.compute.internal", UID:"6fe5382e-4e2f-4fb
9-84f7-e047e5fa9267", APIVersion:"v1", ResourceVersion:"719376", FieldPath:""}): type: 'Normal' reason: 'ScaleDown' node removed by cluster autoscaler
I0918 19:26:23.221433       1 reflector.go:383] k8s.io/client-go/informers/factory.go:134: Watch close - *v1.StatefulSet total 0 items received
I0918 19:26:27.097009       1 static_autoscaler.go:192] Starting main loop
I0918 19:26:27.098090       1 utils.go:590] No pod using affinity / antiaffinity found in cluster, disabling affinity predicate for this loop
I0918 19:26:27.098105       1 filter_out_schedulable.go:65] Filtering out schedulables
I0918 19:26:27.098147       1 filter_out_schedulable.go:130] 0 other pods marked as unschedulable can be scheduled.
I0918 19:26:27.098173       1 filter_out_schedulable.go:130] 0 other pods marked as unschedulable can be scheduled.
I0918 19:26:27.098191       1 filter_out_schedulable.go:90] No schedulable pods
I0918 19:26:27.098206       1 static_autoscaler.go:334] No unschedulable pods
I0918 19:26:27.098217       1 static_autoscaler.go:381] Calculating unneeded nodes
I0918 19:26:27.098234       1 pre_filtering_processor.go:66] Skipping ip-192-168-5-160.eu-west-1.compute.internal - node group min size reached
I0918 19:26:27.098242       1 pre_filtering_processor.go:66] Skipping ip-192-168-3-92.eu-west-1.compute.internal - node group min size reached
I0918 19:26:27.098250       1 pre_filtering_processor.go:66] Skipping ip-192-168-4-44.eu-west-1.compute.internal - node group min size reached
```

Kuvio 66. Cluster Autoscalerin loki

```
ahis@archlinux  vlt-k8s-sandbox  ? master ●  aws-vault exec demo -- kubectl logs metrics-server-5c67bd5dd4-mpdpc -n kube-system
I0918 20:00:58.916972       1 serving.go:312] Generated self-signed cert (apiserver.local.config/certificates/apiserver.crt, apiserver.local.config/certificates/apiserver.key)
I0918 20:01:03.822021       1 secure_serving.go:116] Serving securely on [::]:443
```

Kuvio 67. Metrics Serverin loki

4.4 Testaus

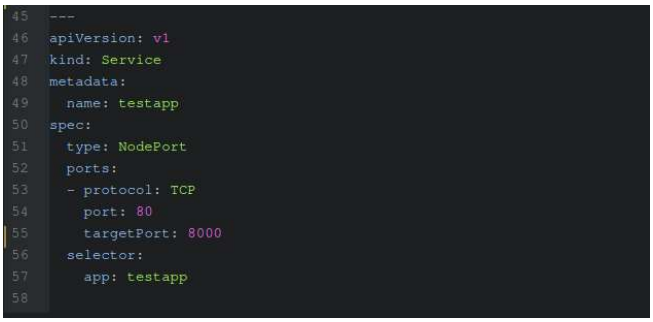
4.4.1 Testisovelluksen käyttöönotto

Testataan klusterin ja sen komponenttien toimivuutta, laittamalla klusteriin ajoon Web-palvelin, joka tulostaa sitä isännöivän noodin IP:n ja podin nimen. Sovellukselle luodaan TLS-sertifikaatti, joka asennetaan klusterin Ingress-kuormantasaajaan, jonka kautta klusterissa ajettaviin kontteihin otetaan yhteys. Sovellukselle luodaan verkko-tunnus *testapp.vlt-k8s.net*, jonka External DNS -kontrolleri laittaa osoittamaan CNAME-tietueella Ingress-kuormantasaajaan.

Deployment ja Service

Sovellukselle luodaan Deployment, jossa määritellään, että podista ajetaan kolmea replikaa ja Elastic Container Registry -konttirekisteriin (ECR) viedyn sovelluskontin image. Kontit paljastetaan klusterin sisäiselle liikenteelle, luomalla Service, joka ohjaa Servicen porttiin 80 tulevan liikenteen konttien porttiin 8000. Testisovelluksen Deploymentin konfiguraatio esitetty kuviossa 68. Testisovelluksen Servicen konfiguraatio esitetty kuviossa 69.

Kuvio 68. Testisovelluksen Deployment



Kuvio 69. Testisovelluksen Service

Sovelluksen paljastaminen klusterin ulkopuolelle Ingressillä

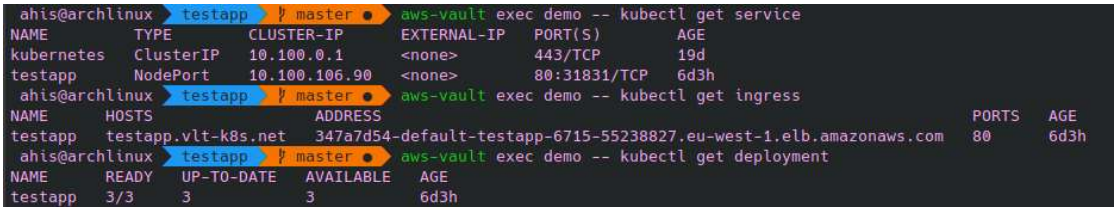
Testisovellus paljastetaan klusterin ulkopuolelle Ingressillä, joka luo julkiseen aliverkkoon kuormantasaajan. Ingress luo kuormantasaajaan listenerin porteille 80 ja 443, johon se liittää kohteeksi sovellukselle luodun Servicen. Kuormantasaaja reitittää liikenteen oikealle Servicelle polun ja hostnamen perusteella.

Ingressin konfiguraatiossa määritellään portit, mille kuormantasaaja luo listenerit. Ingressiin konfiguroidaan myös SSL-uudelleenohjaus. Ingressille määritellään säännönä domain nimi, jonka External DNS -kontrolleri käy osoittamassa CNAME-tietueella kuormantasaajaan. Ingress Controller myös asentaa tälle domain-nimelle luodun TLS-sertifikaatin kuormantasaajaan. Säännössä luodaan polku SSL-uudelleenohajukselle ja toinen polku, joka reitittää liikenteen testisovellukselle luodun Servicen porttiin 80. Ingress-konfiguraatio esitetty kuviossa 70.

Kuvio 70. Testisovelluksen Ingress

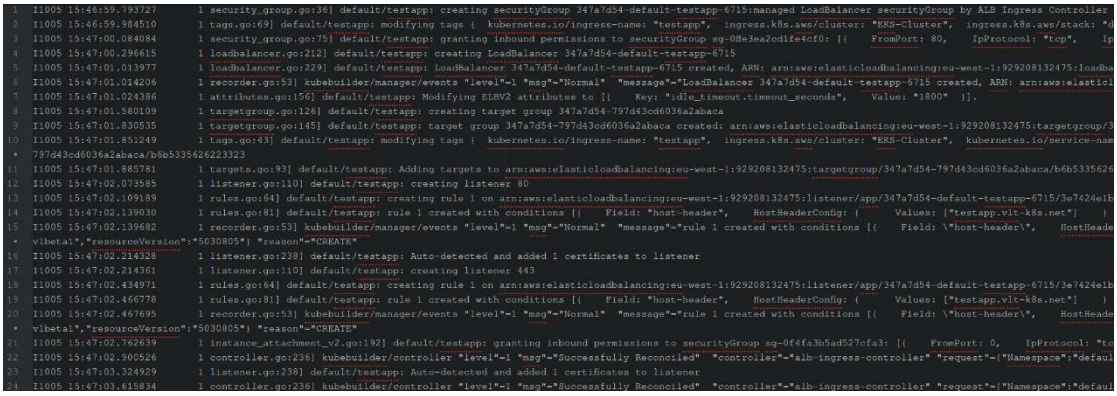
Sovelluksen toiminnan varmistaminen

Kun Ingress, Deployment ja Service ovat otettu käyttöön tarkastetaan, että klusteriin luotiin halutut resurssit ja että ne ovat toiminnassa. Kuviossa 71 nähdään ajossa olevat Podit, Service ja Ingress.



Kuvio 71. Testisovelluksen deployment, service ja ingress toiminnassa

Ingress Controllerin lokeista havaitaan, että se on luonut kuormatasaajan ja konfiguroinut sen määritetyn mukaiseksi. Lokeista näkee myös, että Ingress Controller on havainnut TLS-sertifikaatin ja asentanut sen kuormantasaajaan. Ingress Controllerin lokit esitetty kuviossa 72.

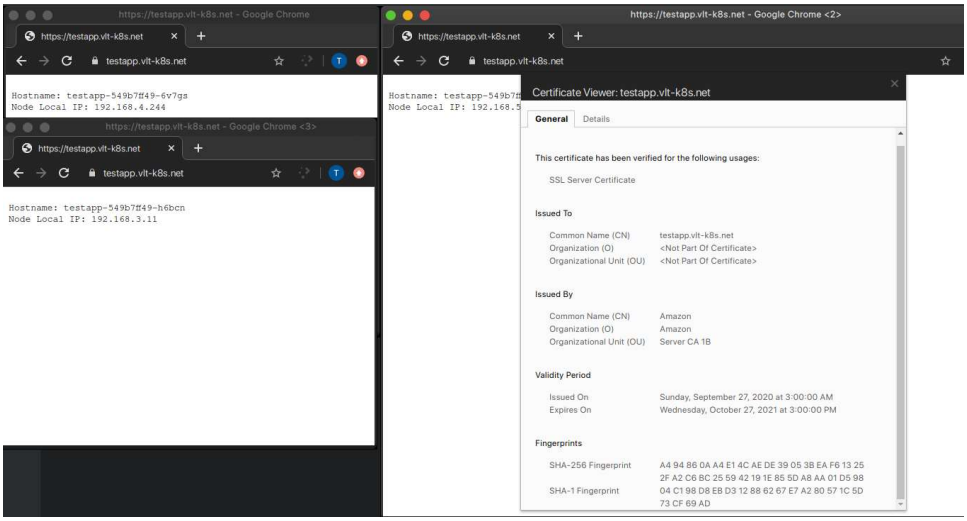


Kuvio 72. Ingress kontrollerin lokitapahtumat

External DNS -kontrollerin lokeista huomataan, että se on luonut CNAME-tietueen osoittamaan kuormantasaajaan. External DNS -kontrollerin loki esitetty kuviossa 73.

Kuvio 73. External DNS -kontrollerin lokitaphtumat

Selaimesta nähdään, että sivusto *testapp.vlt-k8s.net* palvelee kolmelta eri podilta ja noodilta, ja että sillä on Amazonin myöntämä TLS-varmenne. Sovellus selaimessa ja TLS-varmenteen tiedot esitetty kuviossa 74.



Kuvio 74. Testisovellus selaimessa ja TLS-varmenteen tiedot

4.4.2 Klusterin skaalautuvuuden testaus

Testataan klusterin skaalautuvuutta generoimalla HTTP-pyyntöjä testisovellukselle Locust-työkalulla. Klusterin Terraform-konfiguraatiossa klusterin minimikooksi on määritetty 3 noodia ja maksimikooksi 6 noodia. Noodit ovat instanssityypiltään

t3.small, joissa on 2 virtuaalista prosessoriydintä ja 2 gigatavua muistia. Cluster Autoscaler säätelee noodien määrää kolmen ja kuuden noodin välillä, perustuen konttien resurssivarauksiin. Jos uudelle podille ei ole riittävästi resursseja, se menee odotustilaan ja Cluster Autoscaler luo uuden noodin, jolle podi sijoitetaan. Cluster Autoscaler poistaa noodin, jos podien resurssivaraukset pystytään täyttämään pienemmällä noodimäärällä.

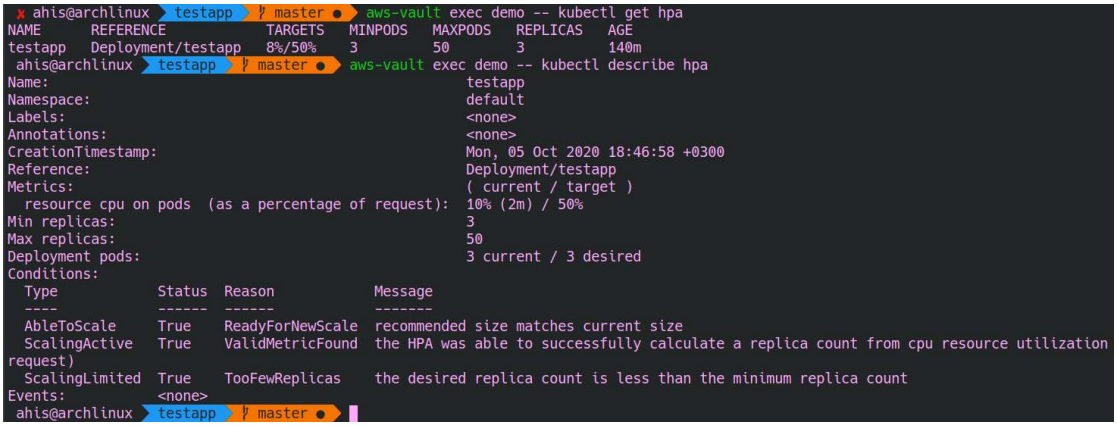
Horizontal Pod Autoscalerin konfigurointi

Cluster Autoscalerin toiminnan mahdollistamiseksi tarvitaan Horizontal Pod Autoscaler, joka kasvattaa tai laskee podien määrää jos tietty kuormitusaste ylittyy. Testisovelluksen Deploymentille luodaan Horizontal Pod Autoscaler, joka skaalaa podien määrää kolmen ja viidenkymmenen podin välillä. Konfiguraatiossa on määritelty tavoite CPU-käyttöaste 50%, jossa HPA pyrkii pitämään podien käyttöasteen keskiarvon. Jos keskiarvoinen käyttöaste ylittää rajan, HPA lisää podeja kunnes käyttöaste on alle 50%. Jos käyttöaste alittaa rajan, HPA poistaa podeja kunnes käyttöaste on 50%. Horizontal Pod Autoscalerin konfiguraatio esitetty kuviossa 75.

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: testapp
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: testapp
  minReplicas: 3
  maxReplicas: 50
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
```

Kuvio 75. Horizontal Pod Autoscaler testisovellukselle

Kun HPA on otettu käyttöön, huomataan, että podien replikoita on ajossa kolme ja maksimimäärä on viisikymmentä (ks. Kuvio 76). HPA lukee onnistuneesti metrics-serveriltä saadut metriikat ja laskee tarvittavaa podien määrää. HPA ei vähennä podien määrää alle kolmen, koska se on määritetty miniarvoksi HPA:n konfiguraatiossa.



Kuvio 76. Horizontal Pod Autoscaler toiminnassa

Klusterin kuormittaminen HTTP-pyynnöillä

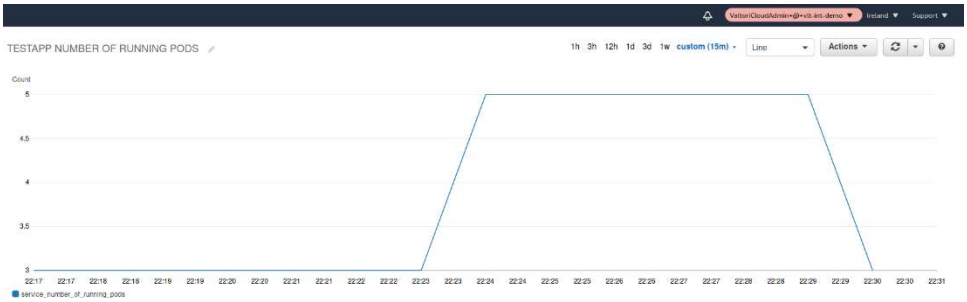
Klusterille generoidaan käyttäjiä Locust-työkalulla, joka asennetaan työasemalle Pythonin Pip-pakettimanagerilla. Python tiedostoon määritellään tehtäviä, mitä käyttäjät suorittavat. Tässä tapauksessa tehtävä on yksinkertainen HTTP GET -pyyntö testapp.vlt-k8s.net hostille. Locust avaa localhostin porttiin 8089 selaimessa toimivan käyttöliittymän, jossa määritellään, kuinka montaa käyttäjää simuloidaan ja kuinka monta käyttäjää testiin lisätään sekunnissa. Taulukossa 4 on tilastoitu skaalautuvuustestin tulokset.

Taulukko 4. Skaalautuvuustestin tulokset

Käyttäjät	Lisää käyttäjiä / s	Maks. Noodit	Maks. Podit	Pyyntöä / s	Epäonnistuneet pyynnöt	Keskiarvo podin CPU kuorma (%)	Keskiarvo vastausaika (ms)	Maks. vastausaika (ms)
100	1	3	5	35	0	50	67	594
100	10	3	5	35	0	50	65	338
100	50	3	5	35	0	50	66	424
300	1	4	15	105	0	50	66	2057
300	10	4	16	105	0	50	65	490
300	50	4	15	105	0	50	65	480
600	1	6	29	210	0	50	67	1075
600	10	6	30	210	0	50	67	946
600	50	6	29	210	0	50	65	1376
1000	50	6	49	350	0	67	65	2156

Testejä simuloitiin 100:lla, 300:lla, 600:lla ja 1000:lla käyttäjällä. Jokainen testi aloitettiin alkutilanteesta, jossa klusterissa oli kolmella noodilla ajossa kolme podia. Sadalla käyttäjällä klusterin noodien määrä ei kasvanut ja podien määrä kasvoi kolmesta viiteen. Jokaisella testauskerralla alussa vastausajat pyyntöihin olivat pidempiä, koska autoskaalaus ei ehdi luoda uusia podeja heti kuorman noustessa vaan ja olemassa olevat podit kuormittuneina palvelevat hitaammin. Kun autoskaalaus luo lisää podeja, vastausajat pienenevät. Kuviossa 77 esitetyssä Locust -testissä statistiikkaa testistä, jossa simuloidaan 100 käyttäjää ja käyttäjiä lisätään testiin 1 sekunnissa. Kuvion vastausaikojen käyrästä nähdään, että vastausajat ovat suuremmat testin alkaessa. Keltainen käyrä symboloi 95 persentiilin vastausaikaa, josta huomataan, että se on ensimmäisen minuutin ajan n. 250 – 300 millisekuntia, jonka jälkeen se laskee alle 100 millisekuntiin. Vihreästä käyrästä nähtävä mediaani vastausaika on ensimmäisen 10 sekunnin jälkeen noin 60 ms. Podien määrän Cloudwatch-metriikka esitetty kuviossa 78.

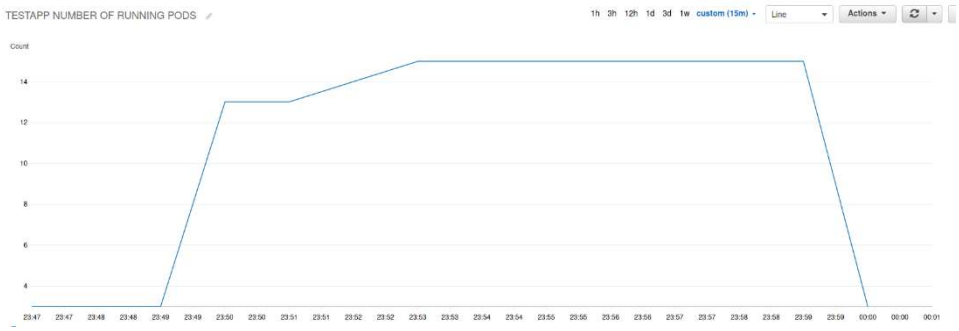
Kuvio 77. Testausmetriikka kun simuloidaan 100 käyttäjää lisäämällä 1 käyttäjä sekunnissa



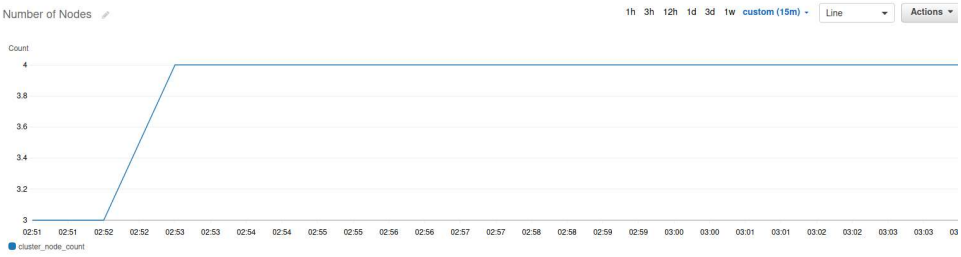
Kuvio 78. Podien skaalausmetriikka Cloudwatchissa 100 käyttäjän kuormalla

Kolmensadan käyttäjän testeissä noodien määrä kasvoi kolmesta neljään. Podien määrä kasvoi kolmesta viiteentoista. Virhevastauksia ei tullut missään testissä. Vastausajat olivat alussa suurempia, mutta laskivat kun autoskaalaus lisäsi noodeja ja podeja. Kuvion 79 vastausaika-statistiikasta nähdään, että käyttäjien lisäysvaiheessa vastausaikojen 95 persentiili on n. 300-250 millisekunnissa, mutta kun käyttäjämäärä tasaantuu ja autoskaalaus lisää noodin (ks. kuvio 80) ja podeja (ks. Kuvio 81) niin se laskee alle sataan millisekuntiin.

Kuvio 79. Testausmetriikka 300 käyttäjän kuormalla kun testiin lisätään 10 käyttäjää sekunnissa

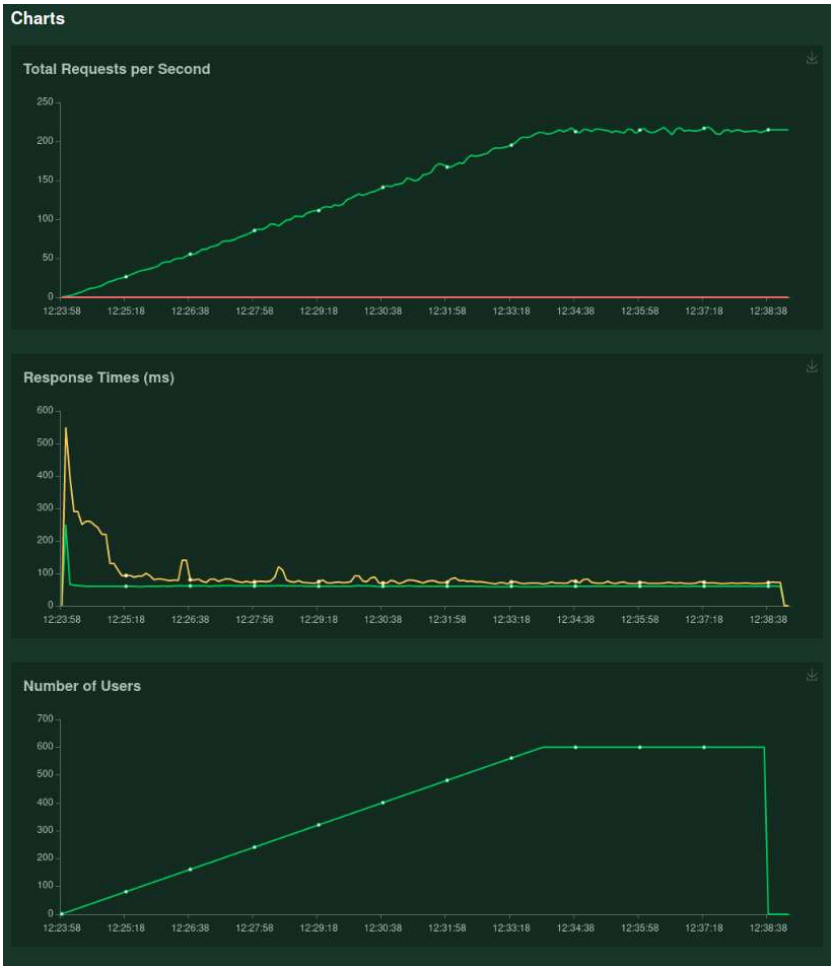


Kuvio 80. Podien skaalausmetriikka 300 käyttäjän kuormalla

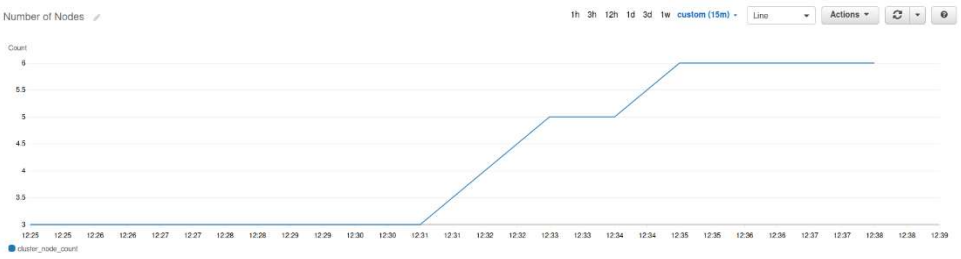


Kuvio 81. Noodien skaalausmetriikka 300 käyttäjän kuormalla

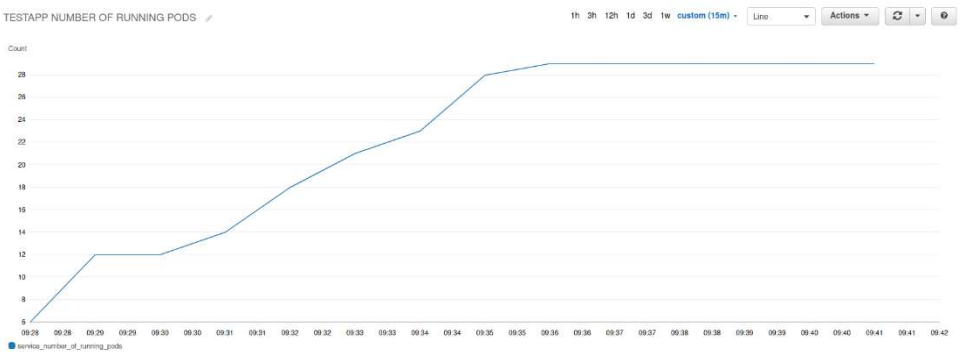
Kuudensadan käyttäjän testissä noodien määrä nousi kolmesta kuuteen (ks. Kuvio 83) ja podien määrä kolmesta 29:ään (ks. Kuvio 84). Testien alussa mediaani vastausaika ja 95 persentiilin vastaustaika olivat suuremmat, mutta ne laskivat nopeasti autoskaalauksen lisätessä nodeja ja podeja klusteriin (ks. Kuvio 82).



Kuvio 82. Testausmetriikka 600 käyttäjällä kun testiin lisätään 1 käyttäjä sekunnissa



Kuvio 83. Noodien skaalausmetriikka 600 käyttäjän kuormalla



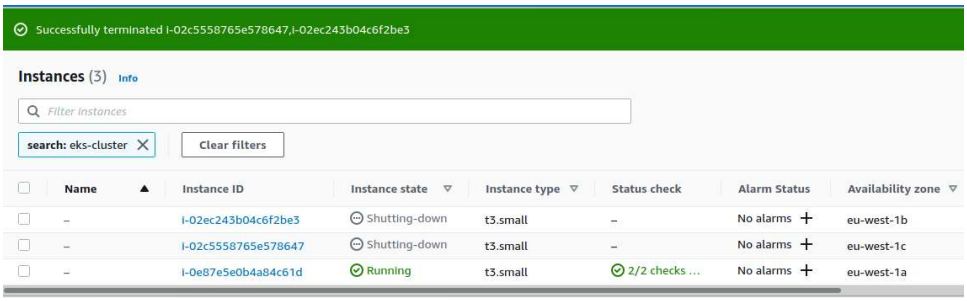
Kuvio 84.Podien skaalausmetriikka 600 käyttäjän kuormalla

4.4.3 Klusterin saatavuuden testaus

Klusterin saatavuutta testataan poistamalla noodeja klusterista. Klusterin virtuaaliverkko jakautuu kolmelle eri saatavuusvyöhykkeelle, joihin klusterin hallintataso ja Ingress-kuormantasaaja on monennettu. Klusterin noodit jakautuvat tasaisesti kolmelle eri saatavuusvyöhykkeelle. Saatavuusvyöhykkeiden häiriöitä simuloidaan poistamalla saatavuusvyöhykkeellä sijaitsevat noodit kuormitetusta klusterista. Noodin poisto klusterista esitetty kuviossa 85. Taulukossa 5 on esitetty saatavuustestien tulokset.

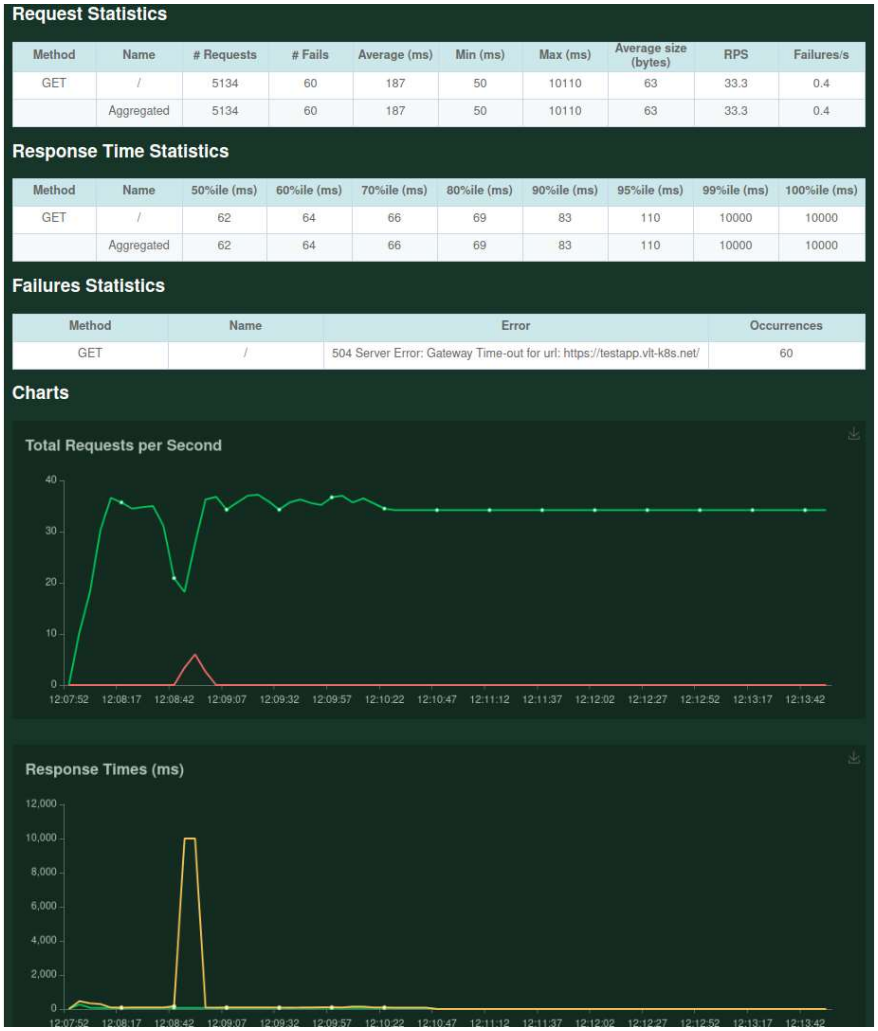
Taulukko 5. Saatavuustestien tulokset

Käyttäjät	**RPS**	**Noodit**	**Poistetut Noodit**	**Virheet**	**Häiriön kesto (s)**
100	35	3	1	60	20
100	35	3	2	270	60
600	210	6	1	1056	45
600	210	6	2	1375	50
600	210	6	4	1901	60



Kuvio 85. Klusterin noodien tuhoaminen

Ensin klusteria kuormitettiin sadan käyttäjän kuormalla, joka generoi HTTP-pyyntöjä noin 35 sekunnissa. Klusterin podit jakautuvat kolmelle noodille. Yhden noodin poistaminen aiheuttaa sovellukseen noin 20 sekunnin pituisen häiriön, jonka aikana maksimissaan 5% lähetetyistä pyynnöistä sekunnissa epäonnistuu (ks. Kuvio 86). Kun klusterista poistetaan kaksi noodia kolmesta, syntyy 60 sekunnin pituinen häiriö, jonka aikana alle puolet sekunnin aikana lähetetyistä pyynnöistä epäonnistuu (ks. kuvio 87).



Request Statistics

Method	Name	# Requests	# Fails	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	RPS	Failures/s
GET	/	5134	60	187	50	10110	63	33.3	0.4
	Aggregated	5134	60	187	50	10110	63	33.3	0.4

Response Time Statistics

Method	Name	50%ile (ms)	60%ile (ms)	70%ile (ms)	80%ile (ms)	90%ile (ms)	95%ile (ms)	99%ile (ms)	100%ile (ms)
GET	/	62	64	66	69	83	110	10000	10000
	Aggregated	62	64	66	69	83	110	10000	10000

Failures Statistics

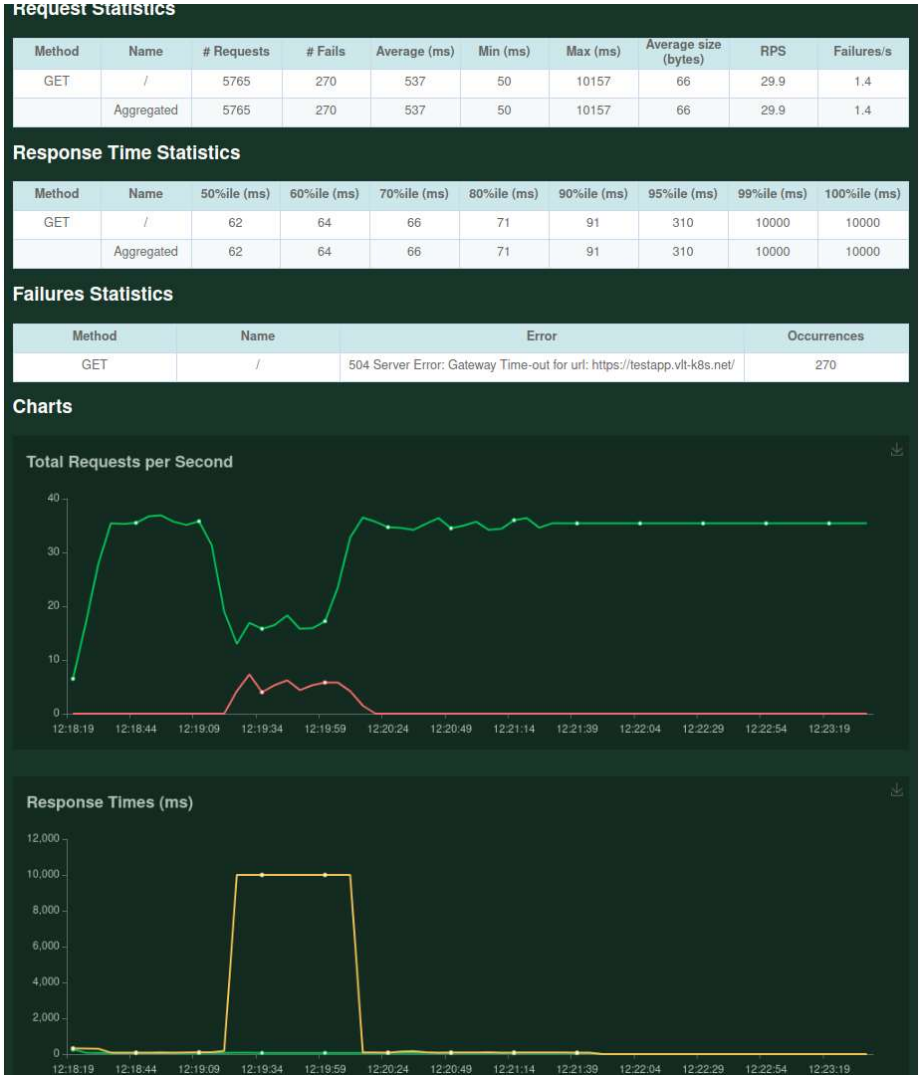
Method	Name	Error	Occurrences
GET	/	504 Server Error: Gateway Time-out for url: https://testapp.vlt-k8s.net/	60

Charts

Total Requests per Second

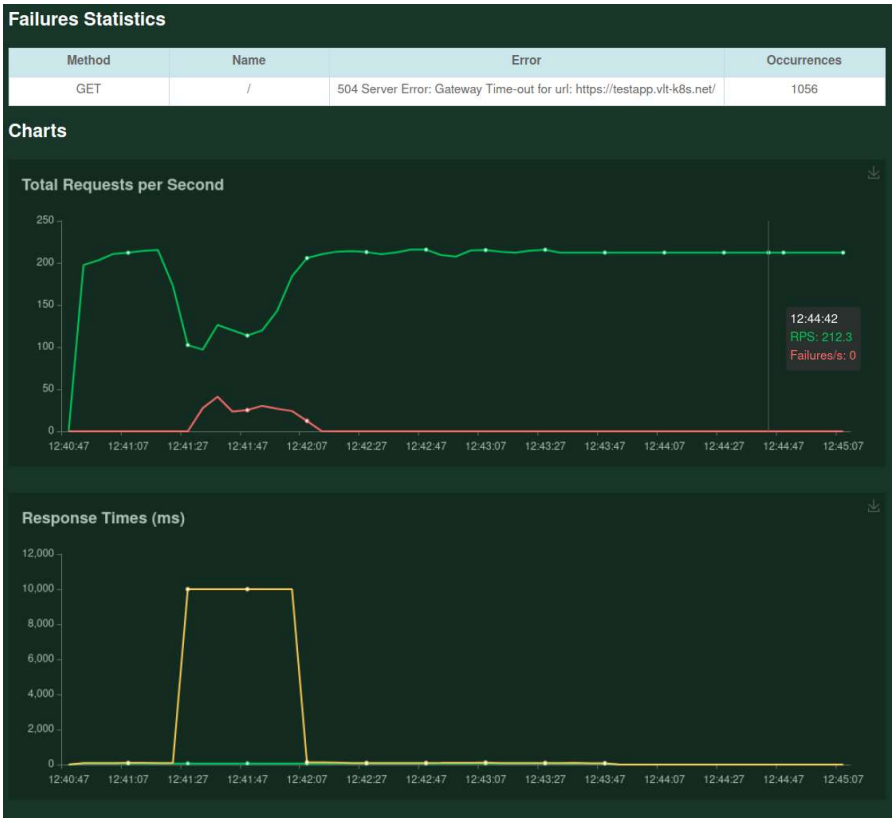
Response Times (ms)

Kuvio 86. Testausmetriikka kun käyttäjiä 100 ja 1 noodi tuhotaan

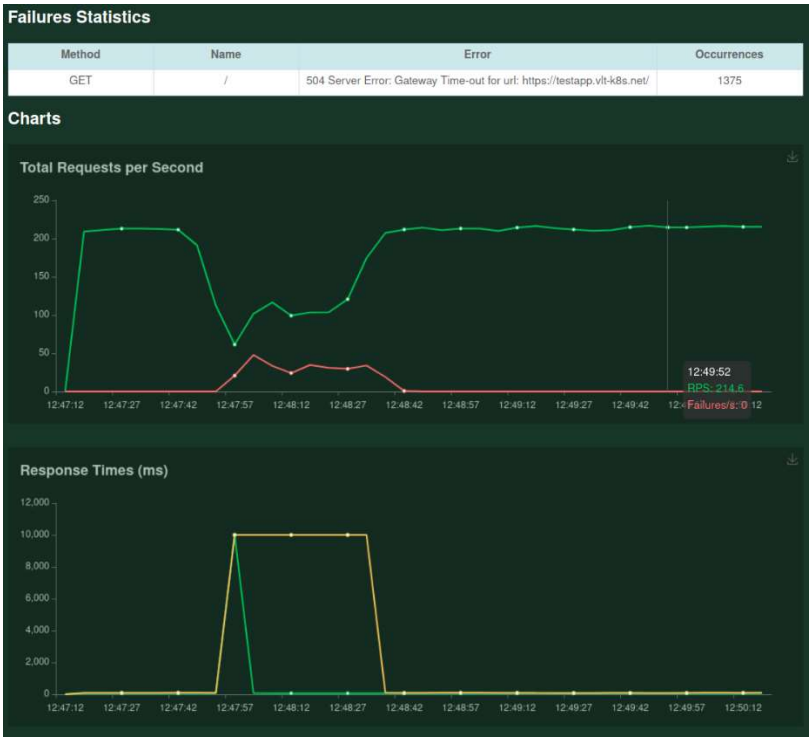


Kuvio 87. Testausmetriikka kun käyttäjiä 100 ja 2 noodia tuhotaan

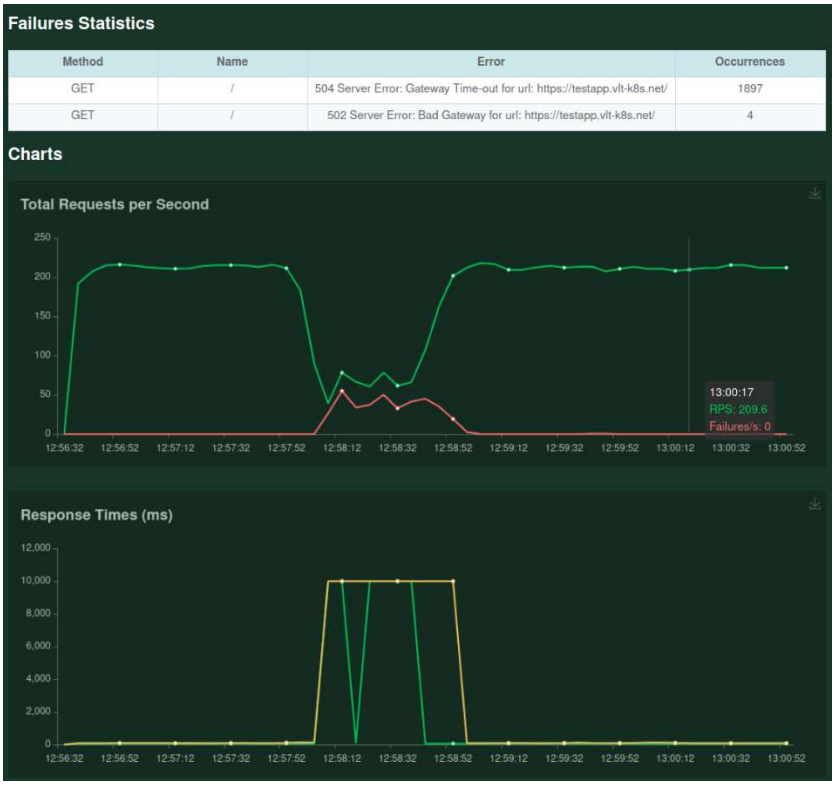
600:n käyttäjän kuormalla klusteri skaalautuu kuuden noodin kokoiseksi. Noodit jakautuvat tasaisesti saatavuusvyöhykkeille eu-west-1a, eu-west-1b ja eu-west-1c. Kun klusterista poistetaan yksi noodi, aiheutuu 45 sekunnin pituinen häiriö, jolloin neljäsosa sovellukselle lähetetyistä pyynnöistä epäonnistuu (ks. Kuvio 88). Kun klusterista poistetaan kaksi noodia eli simuloidaan yhden saatavuusvyöhykkeen häiriötä, aiheutuu 50 sekunnin pituinen häiriö jolloin, noin kolmasosa pyynnöistä epäonnistuu (ks. Kuvio 89). Kun klusterista poistetaan 4 noodia eli simuloidaan kahden saatavuusvyöhykkeen häiriötä, syntyy 60 sekunnin pituinen häiriö, jolloin noin 60% sovellukselle lähetetyistä pyynnöistä epäonnistuu (ks. Kuvio 90).



Kuvio 88. Testausmetriikka kun käyttäjiä 600 ja 1 noodi tuhotaan



Kuvio 89. Testausmetriikka kun käyttäjiä 600 ja 2 noodia tuhotaan



Kuvio 90. Testausmetriikka kun käyttäjiä 600 ja 4 noodia tuhotaan

5 Pohdinta

Tavoitteena oli toteuttaa Infrastructure as a Code -työkaluin pystytettävä skaalautuva korkean saatavuuden konttiorkestrointijärjestelmä pilvipalveluun. Tarkoituksena oli luoda konttiorkestrointijärjestelmä infrastruktuurikoodina, joka on helposti uudelleenkäytettävissä moneen ympäristöön. Tuotoksena syntyi Terraform-työkalulla koodattuja infrastruktuurimoduuleja, joilla voidaan luoda AWS-pilvipalveluun EKS-klusteri. Moduulit ovat helposti uudelleenkäytettävissä viittaamalla Terraformin juurimoduulissa, moduulien sijaintiin versionhallinnassa ja syöttämällä moduuleille halutut parametrit. Moduuleista voidaan julkaista uusia versioita, jolloin ympäristöön haluttu moduulin versio voidaan lukita viittauksessa moduulin versionhallintasijaintiin.

Infrastruktuurin luominen onnistuu viittaamalla moduuleihin ympäristön Terraform-juurimoduulissa. Kun moduuleihin on viitattu Terraform -konfiguraatiossa, ympäristö pystytetään yhdellä *terraform apply* -komennolla. Infrastruktuurin luomisessa kestää noin 20 minuuttia. Virtuaaliverkon luomisessa kestää noin kaksi minuuttia, klusterin hallintatason luomisessa noin 16 minuuttia ja työläisnoodien autoskaalausryhmän

luomisessa noin neljä minuuttia. Klusteriin pystytettävien kontrollerien luomisessa kestää noin 20 sekuntia. Sovelluksille saa vaivatta käyttöön TLS-varmenteen ja julkisen verkkotunnuksen klusterin sisältämien kontrollerien ansiosta.

Itse EKS -konttiorkestrointijärjestelmän Kubernetes-hallintataso on helposti käyttöönotettavissa, eikä vaadi ylläpitotyötä. Vaikka AWS tarjoaakin työläisnoodeille valmiit virtuaalikonekuvat, vaatii klusterikokonaisuus silti ylläpitotöitä. Kun klusterin hallintason Kubernetes-versiota päivitetään, pitää klusterin työläisnooditkin päivittää hallintatasoa vastaavaan versioon. Lisäksi myös klusterin sisältämiä kontrollerien versiota pitää päivittää vastaamaan klusteriversioita. Työläisnoodien ja kontrollerien päivittäminen uusiin versioihin aiheuttamatta häiriöitä klusterissa ajettaviin sovelluksiin vaatii manuaalisia työvaiheita tai automaatioskriptien luomista.

Klusterin autoskaalaus toimii hyvin ja se skaalaa klusteriin sovellukselle podeja ja noodeja usealle saatavuusvyöhykkeelle. Autoskaalaus ylöspäin toimii hyvin ja virheettömästi, kun sovelluksen käyttö kasvaa tasaisesti. Jos tiedetään, että sovelluksen käyttö tulee olemaan tietyllä ajanhetkellä monikertainen, nykyisen resurssimäärään verrattuna, kannattaa klusterin resurssimäärää skaalata manuaalisesti tai ajastetusti, koska klusterin autoskaalauksella kestää työläisnoodien käyttöönotossa. Kuorman kasvaessa rajusti, sovelluksen vastausajat voivat pidentyä ja virhevastaukset lisääntyä, sillä aikaa kun noodien määrää lisätään.

Pystytettävä klusteri on korkeasti saatavissa ja kaikki klusterin komponentit on monennettu kolmella eri saatavuusvyöhykkeillä sijaitseviin aliverkkoihin. Klusterin ohjaustaso, työläisnoodit ja Ingress-kuormantasaaja on kaikki monennettu kolmelle eri saatavuusvyöhykkeelle. Jos saatavuusvyöhykkeellä on häiriö, epäterveillä työläisnoodeilla ajossa olevat podit antavat hetkellisesti virhevastauksia, mutta klusteri ohjaa pyynnöt ohjautumaan terveille podeille ja pystyttää uusia noodeja terveille saatavuusvyöhykkeille.

Klusterin pystyttämisen automatisoinnin hankalin vaihe oli sisällyttää kontrollerien Kubernetes-konfiguraatiot osaksi EKS-moduulia. Aluksi tähän käytettiin Terraformin Kubernetes-provideria, mutta provider ei tukenut kaikkia tarvittavia Kubernetesin objektityyppejä, joten päädyttiin ajamaan Kubernetes-konfiguraatiot klusteriin Terraformiin sisällytettynä skripteinä, jotka ajavat kubectl-komentoja. Skriptillä ajettavien

Kubernetes-konfiguraatioiden huono puoli on, että kontrollerit eivät poistu klusterista, jos komennon sisältävä Terraform-resurssi poistuu. Tämä ei sinänsä ole ongelma, koska kontrollereja ei tulisikaan poistaa ja niiden deklaratiivinen päivitys onnistuu silti skriptien avulla. Toinen hankaluus Terraform-konfiguraatioissa on Cluster Autoscalerin ja Terraformissa luodun autoskaalausryhmän ristiriita. Kun Cluster Autoscaler on päällä, se säätelee autoskaalausryhmän ihannekokoa, jolloin Terraformissa määritelty autoskaalausryhmän ihannekoko aiheuttaa muutoksen infrastruktuurissa. Tämä muutos voidaan kuitenkin ohittaa Terraform-ajossa.

Lähteet

Davidson, N. 2018. Kubernetes Ingress with AWS ALB Ingress Controller. AWS Open Source Blog 20.11.2018. Viitattu 16.11.2019. https://aws.amazon.com/blogs/opensource/kubernetes-ingress-aws-alb-ingress-controller/

Merron, D. 2018. IaC: an Introduction DevOps blog -blogikirjoitus BMC sivustolla 17.12.2018 https://www.bmc.com/blogs/infrastructure-as-code/

Ranger, S. 2018 What is cloud computing? ZDNet 13.12.2018. Viitattu 7.10.2020. https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-from-public-and-private-cloud-to-software-as-a/

Rasa, M. & Watts, S. SaaS vs PaaS vs IaaS. Multi-Cloud blog -blogikirjoitus BMC sivustolla 15.6.2019. Viitattu 7.10.2020. https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/

Tripathy, A. 2019. Understanding Kubernetes Cluster Autoscaling. Artikkeli Medium.com sivustolla 31.1.2019. Viitattu 7.10.2020. https://medium.com/kubecost/understanding-kubernetes-cluster-autoscaling-675099a1db92

Wong, W. G. 2016, What's the Difference between Containers and Virtual Machines? Electronic Design 15.7.2016. Viitattu 16.11.2019. https://www.electronicdesign.com/dev-tools/what-s-difference-between-containers-and-virtual-machines

Application Load Balancing User Guide. N.d. Sovelluskuormantasauksen käyttäjäohjeistus AWS:n dokumentaatiosivustolla. Viitattu 16.11.2019. https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

CloudWatch Logs User guide. N.d. Cloudwatch Logien käyttäjäopas AWS:n dokumentaatiosivustolla. Viitattu 16.11.2019. https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html

CloudWatch User Guide. N.d. Cloudwatchin käyttäjäopas AWS:n dokumentaatiosivustolla. Viitattu 16.11.2019. https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html

Deployment Documentation. N.d. Deploymentin dokumentaatio Kubernetesin sivuilla. Viitattu 16.11.2019. https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

EC2 User Guide. N.d. EC2 -palveluiden käyttäjäohjeistus AWS:n dokumentaatiosivustolla. Viitattu 16.11.2019. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

EKS User Guide. N.d. EKS:n käyttäjäopas AWS:n sivuilla. Viitattu 16.11.2019. https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html

EKS Worker Node User Guide. N.d. EKS Worker Nodejen käyttäjäopas AWS:n sivuilla. Viitattu 16.11.2019.
https://docs.aws.amazon.com/eks/latest/userguide/worker.html

How Domain Registration Works. N.d. Ohjeistus Domainien rekisteröintiin AWS:n Route 53 -palvelun kehittäjäoppaassa. Viitattu 16.11.2019.
https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/welcome-domain-registration.html

IAM Roles User Guide. Ohejistus IAM-roolien käyttöön IAM käyttäjäoppaassa. Viitattu 16.11.2019.
https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

IAM User Guide. N.d. Identiteetinhallinnan käyttäjäopas AWS:N dokumentaatiosivustolla. Viitattu 16.11.2019.
https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html

Ingress Controller Documentation. N.d. Ingress Controllerien dokumentaatio Kubernetesin sivuilla. Viitattu 16.11.2019.
https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/

Ingress Documentation. N.d. Ingressien dokumentaatio Kubernetesin sivuilla. Viitattu 16.11.2019. https://kubernetes.io/docs/concepts/services-networking/ingress/

Input Variables Documentation. N.d. Dokumentaatio muuttujien käyttöön Terraform konfiguraatiossa Terraformin dokumentaatiosivustolla. Viitattu 16.11.2019.
https://www.terraform.io/docs/configuration/variables.html

Introduction to Terraform. N.d. Esittely Terraformista tuotteen kotisivuilla. Viitattu 16.11.2019. https://www.terraform.io/intro/index.html

Kubernetes Components. N.d. Katsaus Kubernetes komponentteihin Kubernetesin dokumentaatiossa. Viitattu 16.11.2019.
https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Kubernetes Controllers Overview. N.d. Kubernetes Controllerin esittely Kuberntesin dokumentaatiossa. Viitattu 16.11.2019.
https://kubernetes.io/docs/concepts/architecture/controller/

Kubernetes Objects. N.d. Kubernetes objektit selitetty Kubernetesin dokumentaatiossa. Viitattu 16.11.2019.
https://kubernetes.io/docs/concepts/overview/working-with-objects/kubernetes-objects/

L 1226/2013 Laki valtion yhteisten tieto- ja viestintäteknisten palvelujen järjestämisestä. 2013. Viitattu 16.11.2020.
https://www.finlex.fi/fi/laki/ajantasa/2013/20131226

Metrics Server Github Project. N.d. Metrics Serverin Github -projekti. Viitattu 7.10.2020. https://github.com/kubernetes-sigs/metrics-server

Network Load Balancing User Guide. N.d. Verkkokuormantasauksen käyttäjäohejistus AWS:n dokumentaatiosivustolla. Viitattu 16.11.2019.
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

Output Values Documentation. N.d. Dokumentaatio Output Arvojen käytöstä Terraform moduuleissa Terraformin dokumentaatiosivustolla. Viitattu 16.11.2019. https://www.terraform.io/docs/configuration/outputs.html

Persistent Volume Documentation. N.d. Persistent Volumejen dokumentaatio Kubernetesin sivuilla. Viitattu 16.11.2019 https://kubernetes.io/docs/concepts/storage/persistent-volumes/

Pod Documentation. N.d. Podien dokumentaatiosivu Kubernetesin sivuilla. Viitattu 16.11.2019. https://kubernetes.io/docs/concepts/workloads/pods/pod/

Public Cloud Market report. 2020. Julkisen pilven markkinaraportti Gartnerin sivuilla 10.9.2020. Viitattu 7.10.2020. https://www.gartner.com/en/newsroom/press-releases/2020-08-10-gartner-says-worldwide-iaas-public-cloud-services-market-grew-37-point-3-percent-in-2019

ReplicaSet Documentation. N.d. ReplicaSetien dokumentaatio Kubernetesin sivuilla. Viitattu 16.11.2019. https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

Resource Record Types. N.d. Ohjeistus DNS-tietuetyyppien käyttöön AWS:n Route 53 -palvelun kehittäjäopppaassa. Viittattu 16.11.2019. https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/ResourceRecordTypes.html

Route53 Developer Guide. N.d. Route 53 -palvelun kehittäjäopas AWS:n Dokumentaatiosivulla. Viitattu 16.11.2019. https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html

Route 53 Health Checks Developer Guide. N.d. Route 53 Health Checkin kehittäjäopas AWS:n dokumentaatiosivustolla. Viitattu 16.11.2019. https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/ResourceRecordTypes.html

Service Documentation. N.d. Servicejen dokumentaatio Kubernetesin sivuilla. Viitattu 16.11.2019. https://kubernetes.io/docs/concepts/services-networking/service/

Storage Class Documentation. N.d. Storage Classien dokumentaatio Kubernetesin sivuilla. Viitattu 16.11.2019. https://kubernetes.io/docs/concepts/storage/storage-classes/

Terraform AWS Provider Documentation. N.d. Dokumentaatio AWS Providerista Terraformin dokumentaatiosivulla. Viitattu 16.11.2019. https://www.terraform.io/docs/providers/aws/index.html

Terraform Command Reference. N.d. Terraform komentorivityökalun komentoviite Terraformin dokumentaatiosivuilla. Viitattu 16.11.2019. https://www.terraform.io/docs/commands/init.html

Terraform Configuration Language. N.d. Terraform kielen dokumentaatio Terraformin dokumentaatiosivustolla. Viitattu 16.11.2019. https://www.terraform.io/docs/configuration/index.html

Terraform Upgrade Guide. N.d. Terraform työkalun päivitysopas tuotteen kotisivuilla. Viitattu 16.11.2019. https://www.terraform.io/upgrade-guides/0-12.html

Using Regions and Availability Zones. N.d. Ohjesivu AWS:n Dokumentaatiosivustolla. Viitattu 16.11.2019. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html

Valtorin tuottamat palvelut. N.d. Tietoa Valtorin tuottamista palveluista Valtorin kotisivuilla. Viitattu 16.11.2020. https://valtori.fi/palvelut

Volume Documentation. N.d. Volumejen dokumentaatio Kubernetesin sivuilla. Viitattu 16.11.2019. https://kubernetes.io/docs/concepts/storage/volumes/

VPC User Guide. N.d. VPC -palvelun käyttäjäohjeistus AWS:n dokumentaatiosivustolla. Viitattu 16.11.2019. https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html

What is Kubernetes? N.d. Esittely Kubernetesiin projektin kotisivuilla. Viitattu 16.11.2019. https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/