SAVONIA
ammattikorkeakoulu

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

KUBERNETES-YMPÄRISTÖN PYSTY-TYS VIRTUAALIKONEKLUSTERIIN

Opinnäytetyö

TEKIJÄ Harri Pirskanen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Harri Pirskanen	
Työn nimi KUBERNETES-YMPÄRISTÖN PYSTYTYS VIRTUAALIKONEKLUSTERIIN	
Päiväys 17.2.2024	Sivumäärä/Liitteet 24
Toimeksiantaja/Yhteistyökumppani(t) Savonia-ammattikorkeakoulu oy	

Kubernetes-ympäristö on container- eli konttipohjainen ympäristö, jolla hallitaan mikropalveluita. Nykyaikana osa pilvipalveluista tuotetaan konttipohjaisina. Konttiympäristöjen etuja ovat palvelujen helpompi hallittavuus ja käytettävyys.

Tavoitteena opinnäytetyössä oli selvittää, kuinka kubernetes-ympäristö pystytetään virtuaalikoneklusteriin ja kuinka docker-ympäristön kontit muutetaan kuberneteksen ymmärtämään muotoon. Selvitettiin myös salaisuuksien hallinnan toimintaa kubernetes-ympäristössä ja konttikuvarekisterin toimintaa. Tutkittiin docker- ja kubernetes-ympäristöjen toimintaa ja tutustuttiin samalla konttipohjaisiin ympäristöihin.

Työssä suunniteltiin kehitysympäristö, jossa käytettiin virtuaalikoneita, johon asennukset suoritettiin. Selvitettiin dokumentaation perusteella, kuinka tarvittavat toimenpiteet suoritetaan. Selvitysten jälkeen suoritettiin asennukset kehitysympäristöön. Asennusten aikana kohdattiin ongelmia, jotka täytyy ottaa huomioon, kun tehdään lopullisia asennuksia. Docker-konttien muuttaminen kuberneteksen ymmärtämään muotoon onnistui.

Työvaiheista ja ongelmakohdista tehtiin dokumentaatio, jota toimeksiantaja voi käyttää omien asennusten tekemiseen tulevaisuudessa. Selviää että työvaiheiden etukäteen suunnittelu ja testaus on tärkeää, jotta saadaan toimiva klusteri.

Avainsanat
Docker, kubernetes,

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author(s) Harri Pirskanen	
Title of Thesis Setting Up a Kubernetes Environment to the Virtual Machine Cluster	
Date 17 February 2024	Pages/Appendices 24
Client Organisation /Partners Savonia University of Applied Sciences	

A Kubernetes environment is a container-based environment that manages microservices. Today, some cloud services are provided on a containerized basis. The advantages of containerized environments are easier manageability and availability of services.

The aim of the thesis was to find out how to set up a Kubernetes environment to the virtual machine cluster and how to convert the containers of the Docker environment into a format that the Kubernetes can understand. The operation of secret management in the Kubernetes environment and the operation of the container-registry were also investigated. The Docker and Kubernetes environments were explored, while also learning about container-based environments.

The work involved designing a development environment using virtual machines to perform the installations. The documentation was used to find out how to carry out the necessary measures. After clarifications, installations were carried out in the development environment. Problems were encountered during the installation, which must to be taken into account when making the final installations. The conversion of Docker containers to a format that Kubernetes understands was successful.

A documentation of the work steps and problem areas was made, which can be used by the client for their own installations in the future. It became obvious that planning and testing the steps in advance is important to get a functioning cluster.

Keywords
Docker, kubernetes,

SISÄLTÖ

KUVALUETTELO

1 JOHDANTO

Opinnäytetyön aiheena oli Kubernetes-ympäristön pystytys virtuaalikoneklusteriin. Tämän projektin tavoitteena on luoda toimiva Kubernetes-ympäristö sovellusten käyttöönoton hallintaa varten ja tutkia kuinka olemassa olevan Docker-ympäristön containereita eli kontteja saataisiin siirrettyä Kubernetes-ympäristöön. Toimeksiantajana toimi Savonia-ammattikorkeakoulu oy ja kontaktihenkilönä Jussi Nivamo.

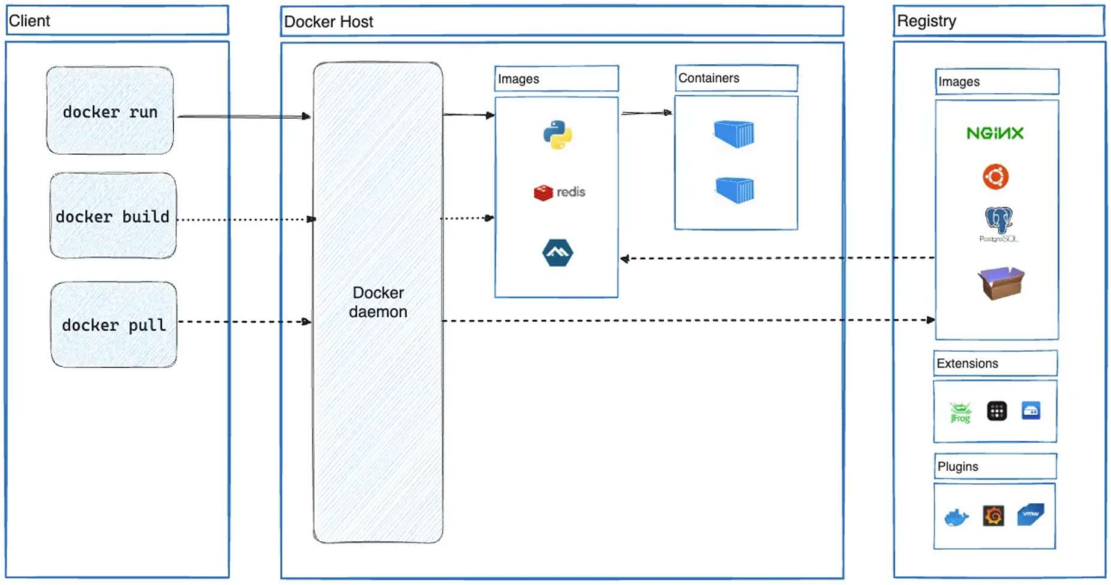
2 TEORIAOSA

2.1 Docker

Docker on avoin alusta sovellusten kehitykselle, toimitukselle ja käyttämiselle. Dockerin avulla voidaan käsitellä sovelluksia erillään infrastruktuurista, jotta voidaan nopeuttaa sovelluskehitystä vähentämällä viivettä koodin kirjoituksen ja sen käyttämisellä tuotannossa. Docker-alustalla saadaan tehtyä yhtenäinen tuotantoalusta pilveen niin että konttiin sisällytetään kaikki mitä sovelluksen käyttämiseen tuotannossa tarvitaan. (Docker, 2024)

2.1.1 Docker image

Docker image eli levykuva on vain-luku malli, jossa on ohjeet docker kontin luontiin. Docker levykuva voi olla esimerkiksi ubuntu-linux levykuvaan perustuva mutta lisäosien kanssa. Esimerkiksi Apache web-serveri ja muita sovelluksia, joita voidaan tarvita sovelluksen käyttöön, sisällytetään mukaan levykuvaan. Näin saadaan luotua yksilöllisiä ja kevyitä alustoja sovelluksien käyttöön. (Docker, 2024)



Kuva 1. Docker arkkitehtuuri

2.1.2 Docker kontti

Docker kontti on ajettava instanssi levykuvasta, joka voidaan luoda, käynnistää, pysäyttää, siirtää tai poistaa käyttämällä Docker API:a tai CLI:tä. Kontti voidaan yhdistää yhteen tai useampaan verkkoon ja siihen voidaan liittää siihen tallennustilaa tai luoda uuden imagen perustuen sen hetkiseen tilaan. Perusasetuksilla kontti on eristetty toisista konteista ja isäntäkoneesta. Kun kontti käynnistetään levykuvasta, voidaan käynnistäessä antaa lisäparametreja ja kun kontti pysäytetään häviävät kaikki tiedot, jotka eivät ole tallennettuna erilliseen tallennustilaan. (Docker, 2024)

2.2 Kubernetes

2.2.1 Yleiskatsaus

Kubernetes-projekti on Googlen 2014 julkaisema avoimen lähdekoodin sovellus. Nimi Kubernetes tulee Kreikan kielen sanasta, joka tarkoittaa ohjaajaa. Yleisesti käytetty lyhennys K8s tulee, kun otetaan sanan ensimmäinen kirjain ja viimeinen kirjain ja välissä on kahdeksan kirjainta. (Kubernetes.io, 2023)

Perinteisessä käyttöönotossa organisaatiot käyttävät sovelluksia fyysisillä servereillä eikä ole selvää tapaa erottaa sovelluksien ja resurssien rajoja fyysisillä servereillä, ja tästä tulee resurssien jako ongelmia. Esimerkiksi jos serverillä pyörii monta sovellusta yhtä aikaa voi olla yksi sovellus, joka vie suuren osan resursseista hidastaa se muita sovelluksia. Yksi ratkaisu tälle on, että ajetaan sovelluksia eri servereillä mutta tämä ei skaalaudu ja ylimääräisiä resursseja saattaa jäädä käyttämättä ja tämä tulee kalliiksi, kun organisaatiossa joudutaan ylläpitämään useita fyysisiä servereitä. (Kubernetes.io, 2023)

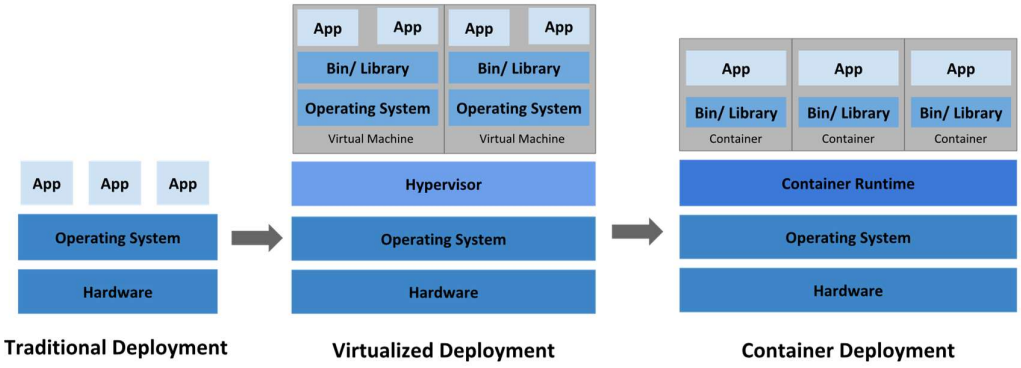
Virtuaalisen käyttöönoton aikakaudella pyöritetään useita virtuaalikoneita yhdellä fyysisellä serverillä. Virtualisoinnin avulla saadaan sovellukset eristettyä omille virtuaalikoneilleen ja näin saadaan yksi kerros turvallisuutta, kun sovelluksen tietoihin ei päästä vapaasti käsiksi toisista sovelluksista. Sovellusten pyöritys virtuaalikoneilla myös mahdollistaa paremman fyysisen serverin resurssien käytön ja skaalautumisen, kun sovellukset ovat helpompia päivittää ja lisätä. Virtuaalikoneiden avulla voidaan esittää joukko fyysisiä resursseja kertakäyttöisten virtuaalikoneiden klusterina. Tässä jokainen virtuaalikone on kokonainen virtuaalinen tietokone, jossa pyörii kaikki komponentit sisältäen oman käyttöjärjestelmänsä virtualisoidun laitteiston päällä. (Kubernetes.io, 2023)

Konttikäyttöönotossa kontit ovat samankaltaisia kuin virtuaalikoneet, mutta niissä on väljemmät eristysominaisuudet, jotta käyttöjärjestelmä voidaan jakaa sovellusten kesken. Kuten virtuaalikoneilla, myös konteilla on oma tiedostojärjestelmä, osuus suorittimesta, muistista ja prosessitilasta ym. Mutta koska ne ovat irrallaan fyysisestä infrastruktuurista ovat ne siirrettävissä eri pilvipalveluihin ja käyttöjärjestelmäjakeluihin. (Kubernetes.io, 2023)

Konteista on tullut suosittuja useiden hyötyjen kautta. Sovellusten luominen ja käyttöönotto on ketterämpää kuin virtuaalikone-kuvien käyttö. Koska konttikuvat ovat muuttumattomia mahdollistaa se luotettavan ja tiheän konttikuvan rakentamisen ja nopeat ja tehokkaat palautukset helpottaen kehitystä, integraatiota ja käyttöönottoa. Luomalla sovelluskonteista kuvat rakennus-/julkaisuajankohtana eikä käyttöönottoaikana saadaan sovellukset erotettua infrastruktuurista. (Kubernetes.io, 2023)

Kun ympäristö pysyy yhdenmukaisena kehitystyössä, testauksessa ja tuotannossa on toiminta taattu niin kannettavassa tietokoneessa kuin pilvipalvelussa. Kun pilvi- ja käyttöjärjestelmäjakelut toimivat missä tahansa käyttöjärjestelmässä, toimitilassa tai pilvipalvelussa ovat ne helposti siirrettäviä. Sovelluskeskeisellä hallinnalla saadaan nostettua abstraktiotasoa käyttöjärjestelmän ajamisesta virtuaalilaitteistossa sovelluksen ajamiseen käyttöjärjestelmässä käyttäen loogisia resursseja. (Kubernetes.io, 2023)

Koska mikropalvelut ovat hajautettuja ja joustavia voidaan sovellukset pilkkoa pienempiin itsenäisiin osiin. Voidaan hallita sovelluksien pienempiä osia dynaamisesti ja helpottaa käyttöönottoa. Kyseessä ei ole mikropalveluiden avulla yksi monoliittinen pino, joka pyörii yhdellä isolla yksikäyttöisellä koneella. Resurssit ovat eristettyinä ja sovelluksen suorituskyky on ennustettavissa paremmin. Resurssit ovat käytettävissä korkeammalla tehokkuudella ja tiheydellä. (Kubernetes.io, 2023)



Kuva 2. Kubernetes konttikäyttöönotto verrattuna perinteiseen ja virtualisoituun

2.2.2 Kubernetes käyttötarkoitukset

Kontit ovat hyvä tapa paketoida ja ajaa sovelluksia. Tuotantoympäristössä tarvitaan kontteja, jotka ajavat sovelluksia ja varmistaa ettei tapahdu käyttökatkoksia. Esimerkiksi jos yksi kontti kaatuu, täytyy toinen käynnistää ja tämä on helpompaa, kun sen hoitaa järjestelmä automaatiolla. Tätä varten on kehitetty kubernetes kehys, joka vastaa sovellusten skaalauksesta ja vikasietoisuudesta. (Kubernetes.io, 2023)

Palvelun löytäminen ja kuorman tasapainottaminen helpottuu, kun kubernetes kontit voidaan paljastaa DNS-nimen tai IP-osoitteen avulla. Jos konttiin kohdistuva liikenne on suurta, Kubernetes pystyy tasaamaan kuormaa ja jakamaan verkkoliikennettä niin, että käyttö on vakaata. Automatisoiduilla käyttöönotoilla ja palautuksilla voidaan muuttaa konttien tilaa haluttuun tilaan hallitusti. Esimerkiksi voidaan automatisoida uusien konttien luonti ja vanhojen poisto niin että aiemmin käytössä olleet resurssit jaetaan uusille konteille. Kubernetes osaa myös käynnistää vikaantuneet kontit uudelleen, korvata tai tappaa kontteja, jotka eivät vastaa käyttäjän määrittelemiä terveysvaatimuksia. (Kubernetes.io, 2023)

Kubernetes sisältää myös automaattisen bin-pakkauksen, jossa kubernetesille annetaan solmupaketti, jota se käyttää konttitehtävien suorittamiseen. Kun kerrotaan kubernetesille kuinka paljon resursseja kukin kontti tarvitsee se osaa jakaa kontit nodeille eli solmuille niin, että resurssit jakautuvat tasaisesti. Sovelluksen vaakasuora skaalaus onnistuu yksinkertaisella komennolla, käyttöliittymän avulla tai automaattisesti suorittimen käytön perusteella. (Kubernetes.io, 2023)

Kubernetes voi käyttää monia eri tallennusjärjestelmiä, kuten paikallinen tallennusjärjestelmä tai julkinen pilvitallennustila. Arkaluontoisten tietojen, kuten salasanojen tai SSH-avaimien tallennuksen

ja hallinnan käyttöönotto onnistuu rakentamatta konttikuvia uudelleen ja paljastamatta salaisuuksia pinon konfiguraatiossa. Koska kubernetes on suunniteltu laajennettavuutta varten, voidaan kubernetes-klusteriin lisätä ominaisuuksia muuttamatta lähdekoodia. (Kubernetes.io, 2023)

2.2.3 Mitä kubernetes ei ole

Kubernetes ei ole perinteinen, kaiken kattava PaaS-järjestelmä. Platform as a service (PaaS) on kehitys- ja käyttöönottoympäristö pilvipalvelussa, jossa on resursseja, joiden avulla voit toimittaa kaikkea yksinkertaisista pilvipohjaisista sovelluksista kehittyneisiin, pilvipohjaisiin yrityssovelluksiin. Asiakas ostaa tarvitsemansa resurssit pilvipalvelulta ja käyttää niitä internet-yhteyden kautta. (Kubernetes.io, 2023, Microsoft Azure, 2024)

Koska kubernetes operoi konttitasolla eikä laitteistotasolla, se tarjoaa joitakin yleisesti sovellettavia ominaisuuksia, jotka ovat yhteisiä PaaS-järjestelmän tarjoamille. Käyttöönotto, skaalaus ja kuorman tasaus, joiden avulla käyttäjät voivat integroida loki-, seuranta- ja hälytysratkaisunsa. Kubernetes ei kuitenkaan ole monoliittinen ja nämä kaikki perustoteutukset ovat valinnaisia. Kubernetes tarjoaa rakennuspalikat kehittäjäalustojen rakennukseen mutta säilyttäen käyttäjän valinnan ja joustavuuden. (Kubernetes.io, 2023)

Kubernetesin tavoitteena on tukea monenlaisia työtehtäviä, kuten tilattomat, tilalliset ja tietojenkäsittelyyn liittyvät työkuormat. Sovellukset, jotka voivat toimia kontissa toimivat myös kubernetesissa eikä se rajoita tuettuja sovellustyyppejä. Kubernetes ei tarjoa sovellustason palveluita, kuten väliohjelmistoja, tietojenkäsittelykehyksiä (esimerkiksi Spark), tietokantoja (esimerkiksi mySQL), välimuisteja eikä klusterientallennusjärjestelmiä (esimerkiksi Ceph) sisäänrakennettuina palveluina. Tällaisia komponentteja voidaan käyttää kubernetesissa tai sovellukset voivat käyttää niitä siirrettävien mekanismien, kuten Open Service Brokerin kautta. (Kubernetes.io, 2023)

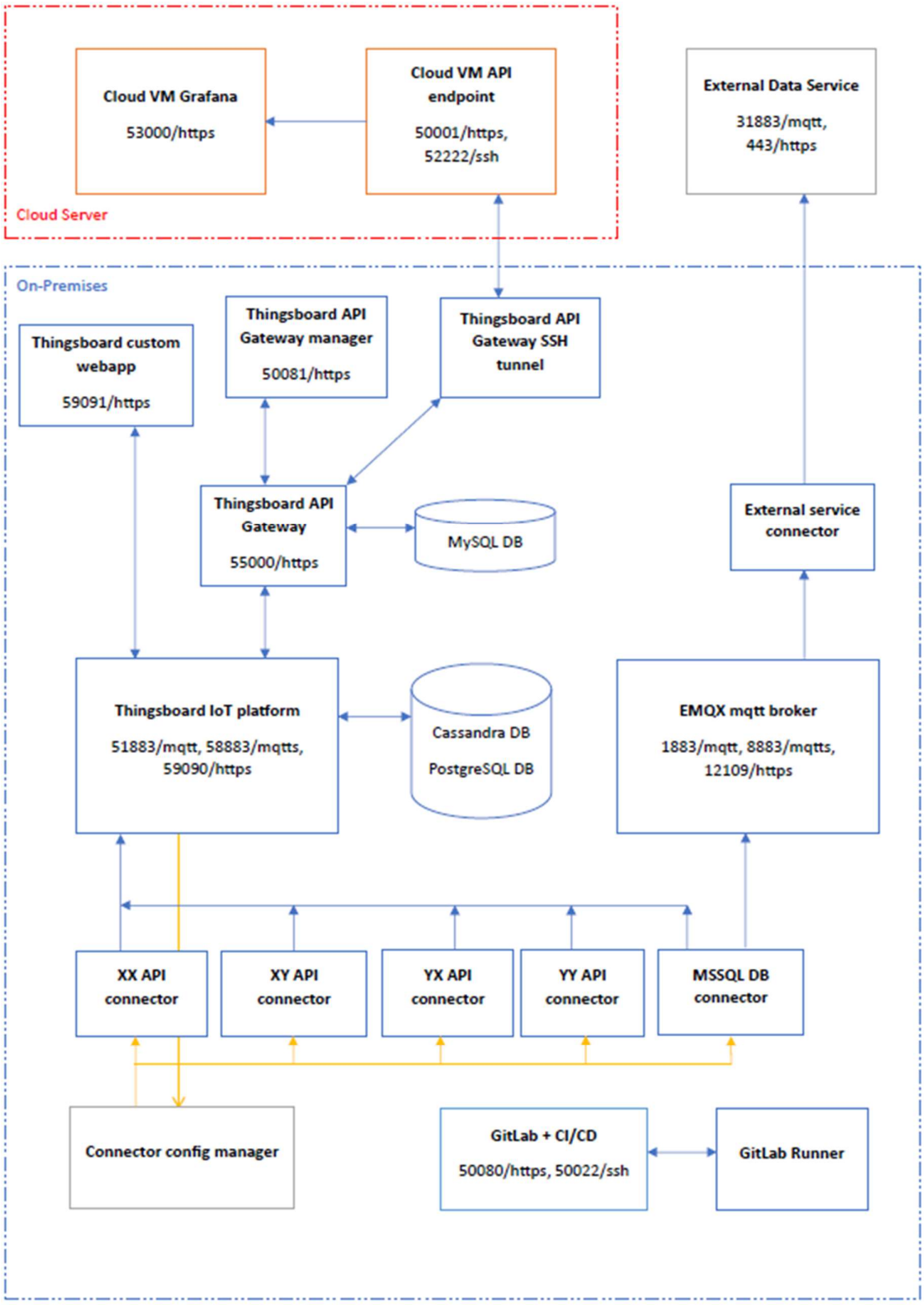
Kubernetes ei sanele kirjaus-, seuranta- tai hälytysratkaisuja. Se tarjoaa joitakin integraatioita konseptin todisteeksi sekä mekanismeja metriikoiden keräämiseksi ja viemiseksi. Kubernetes API on deklaratiivinen eikä se tarjoa eikä edellytä tiettyä konfigurointikieltä tai -järjestelmää (esimerkiksi Jsonnet). Ei myöskään tarjota tai oteta käyttöön kattavia koneen konfigurointi-, ylläpito-, hallinta- tai itsekorjautumisjärjestelmiä. (Kubernetes.io, 2023)

Kubernetes ei myöskään ole pelkkä koordinointijärjestelmä. Itse asiassa se poistaa järjestelyn tarpeen. Järjestelyn tekninen määritelmä on määritellyn työnkulun suorittaminen: ensin tehdään A, sitten B, sitten C. Kubernetes sen sijaan koostuu joukosta riippumattomia, kokoonpantavissa olevia ohjausprosesseja, jotka ohjaavat nykytilaa jatkuvasti kohti haluttua tilaa. Sillä ei pitäisi olla merkitystä, miten pääset A:sta C:hen. Keskitettyä ohjausta ei myöskään tarvita. Tuloksena on helpommin käytettävä järjestelmä, joka on tehokkaampi, vankempi, joustavampi ja laajennettavissa. (Kubernetes.io, 2023)

3 TOTEUTUS

3.1 Alkutilanne

Nykyinen järjestelmä sisälsi dockerilla toteutetun järjestelmän, joka sisälsi web-servereitä, tietokantoja ja reitityksiä (Kuva 3). Työssä selvitettiin, kuinka nykyisen järjestelmän muuttaminen kubernetes-järjestelmään toteutettaisiin. Yksinkertaistettuna tarvittiin kontit web-serverille, tietokantaserverille ja näiden reitit ykset. Myös tietoturva, yksityinen konttikuvarekisteri ja salaisuuksien hallinta olivat tutkittavien asioiden listalla.

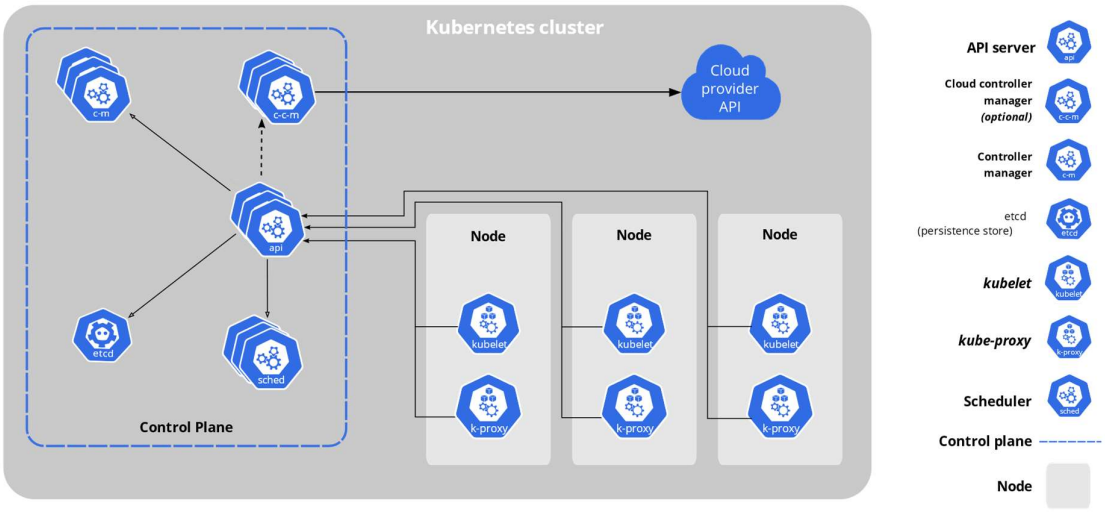


Kuva 3. Docker-ympäristö

3.2 Työ

Selvitettiin, kuinka docker-ympäristö siirretään kubernetes-ympäristöön ja kuinka kubernetes asennus suoritetaan. Docker-ympäristön ”.yml” tiedostojen muutto kuberneteksen ymmärtämään muotoon, jotta saataisiin käytettyä jo valmiita konfiguraatiotiedostoja mahdollisimman paljon hyväksi.

Kuberneteksen reititys on erilainen kuin docker-ympäristössä ja tapahtuu erillisellä reititys kapselilla. Selvitettiin konttien muuttaminen palveluiksi jotta saadaan käyttöön kapseleiden replikointi kuormanhallinnalla. Salaisuuksien hallinta eli kuinka pieni määrä arkaluonteisia tietoja, kuten salasana, tunniste tai avain voitaisiin sisällyttää kapseliin tai konttiin.



Kuva 4. Kubernetes klusteri

3.2.1 Klusteri

Kehitysympäristön kubernetes-klusteri toteutettiin virtuaalikoneilla, joihin asennettiin kaikkiin Ubuntu-linux-käyttöjärjestelmä. Virtuaalikoneet asetettiin omaan virtuaaliseen lähiverkkoonsa. Yhdestä koneesta tuli klusterin master-solmu ja muista worker-solmuja, joilla itse kubernetes-kontit pyörivät. Kaikkiin virtuaalikoneisiin asennettiin SSH-yhteys helpon etäkäytön takaamiseksi. Kaikkiin koneisiin myös asennettiin kubernetes komponentit kubectl ja kubeadm.

Kun asennukset olivat onnistuneesti suoritettu, voitiin kubernetes-klusterin koneet yhdistää toisiinsa ajamalla master-solmusta komento ”sudo kubeadm init --pod-network-cidr=192.168.0.0/16 –pod-network-cidr=10.10.0.0/16”. Suoritettuaan komennon antaa master-solmu join-komennon, joka ajamalla worker-solmuissa saadaan yhdistettyä worker-solmut kubernetes-klusteriin.

```
ubu1@ubu1:~$ kubectl get nodes
NAME   STATUS   ROLES           AGE   VERSION
ubu1   Ready    control-plane   17d   v1.28.4
ubu2   Ready    <none>          17d   v1.28.4
```

Kuva 5. Kubernetes-klusteri näkyy master-solmussa

Kun ajettiin ”join”-komento worker-solmussa, voitiin tarkistaa master-solmusta onnistuiko yhdistys oikein komennolla ”kubectl get nodes” (Kuva 5.).

3.2.2 Reititys Calicolla

Reititykseen kubernetes tarvitsee verkkokäytäntöjen tarjoajan ja tähän valittiin Tigera Calico. Calico on verkko- ja tietoturvaratkaisu, jonka avulla kubernetes-työkuormat ja muut kuin kubernetes/legacy-työkuormat voivat kommunikoida saumattomasti ja turvallisesti. Calicon avulla hoidetaan Kubernetes-klusterin solmujen reititys. Kubernetes-verkkomallissa jokaiselle kapselille annetaan oma IP-osoite ja kontit jakavat kapselin IP-osoitteen ja voivat näin keskustella toistensa kanssa. Kapselit voivat keskustella toistensa kanssa klusterissa käyttämällä kapselien IP-osoitteita ilman NAT:ia. Kapselien kommunikointia rajoittaa verkkokäytäntöjen tarjoaja kuten calico. (Tigera.io)

calico.yaml

```yaml
---
# Source: calico/templates/calico-kube-controllers.yaml
# This manifest creates a Pod Disruption Budget for Controller to allow K8s Cluster Autoscaler to evict

apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: calico-kube-controllers
  namespace: kube-system
  labels:
    k8s-app: calico-kube-controllers
spec:
  maxUnavailable: 1
  selector:
    matchLabels:
      k8s-app: calico-kube-controllers
---
# Source: calico/templates/calico-kube-controllers.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: calico-kube-controllers
  namespace: kube-system
---
# Source: calico/templates/calico-node.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: calico-node
  namespace: kube-system
---
# Source: calico/templates/calico-node.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: calico-cni-plugin
  namespace: kube-system
---
```

Kuva 6. Calico.yaml esimerkki

Calicon asennus kehitysympäristöön toteutettiin yksinkertaisesti vain lataamalla ”calico.yaml”-tiedosto komennolla ”curl https://raw.gitusercontent.com/projectcalico/calico/v3.25.0/manifest/calico.yaml -O” (Kuva 5). Kun ”calico.yaml”-tiedosto oli ladattu voitiin komennolla ”kubectl apply -f calico.yaml” käynnistää calico-kapseli. Tämän jälkeen solmut näkyivät ”Ready”-tilassa ja voitiin käynnistää ensimmäiset kapselit. (Tigera.io, 2024)

3.2.3 Kapselin käynnistys ja kompose

Ensimmäiseksi kapseliksi valittiin nginx-webserveri, jonka docker-compose-tiedosto oli helposti saatavilla. ”Docker-compose.yml” tiedosto pitää muuttaa kuberneteksen ymmärtämään muotoon ja se hoidettiin kompose-työkalulla. Kun kompose oli asennettu, voitiin ajaa komento ” kompose convert”, jolloin kompose yrittää muuttaa ”Docker-compose.yml” tiedoston sisällön kuberneteksen tukemaan muotoon. (Kompose.io, 2024)

Kun kompose saatiin ajettua onnistuneesti (kuva 7) saatiin kuberneteksen tukema ” Nginx-deployment.yaml”-tiedosto, joka voitiin sitten komennolla ”kubectl apply -f nginx-service.yaml,nginx-deployment.yaml” ottaa käyttöön kubernetesklusterissa. Näin saatiin yksi nginx-kapseli, jonka jakama testisivu näkyi virtuaalikoneen selaimella ”10.110.247.247:8000”-osoitteessa(kuva 9). Osoite saatiin komennolla ”kubectl describe svc nginx”, joka kertoo nginx-palvelun tarkemmat tiedot kuten IP-osoitteen, josta palvelu löytyy. (Kompose.io, 2024)

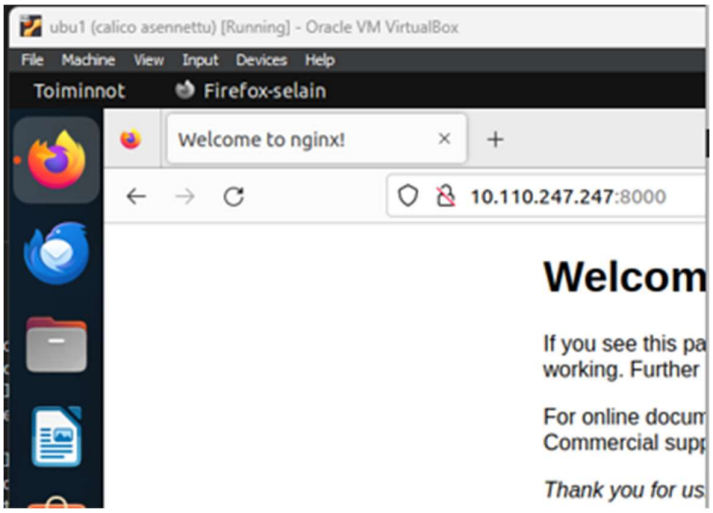
compose > docker-compose.yml

```yaml
version: "3"

services:
  nginx:
    image: nginx
    container_name: nginx
    restart: unless-stopped
    ports:
      - 8000:80

```

Kuva 7.Nginx-serverin docker-compose.yml

```
ubu1@ubu1:~$ cd compose/
ubu1@ubu1:~/compose$ kompose convert
WARN Restart policy 'unless-stopped' in service nginx is not supported, convert it to 'always'
INFO Kubernetes file "nginx-service.yaml" created
INFO Kubernetes file "nginx-deployment.yaml" created
ubu1@ubu1:~/compose$
```

Kuva 8. Tulosteet kun kompose on ajettu onnistuneesti



Kuva 9. Nginx oletussivu virtuaalikoneen selaimessa

```
ubu1@ubu1:~$ kubectl describe svc nginx
Name:              nginx
Namespace:         default
Labels:            io.kompose.service=nginx
Annotations:       kompose.cmd: kompose convert
                   kompose.version: 1.26.0 (40646f47)
Selector:          io.kompose.service=nginx
Type:              ClusterIP
IP Family Policy:  SingleStack
IP Families:       IPv4
IP:                10.110.247.247
IPs:               10.110.247.247
Port:              8000  8000/TCP
TargetPort:        80/TCP
Endpoints:         <none>
Session Affinity:  None
```

Kuva 10. Nginx-palvelun tiedot

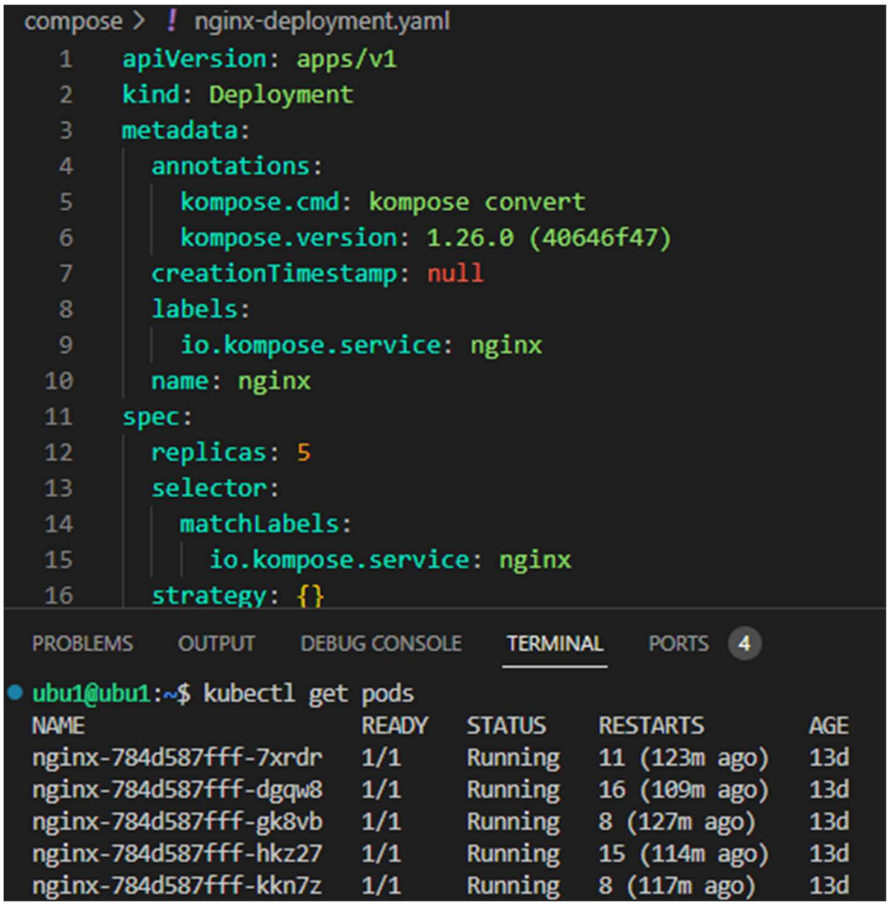
compose > ! nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    kompose.cmd: kompose convert
    kompose.version: 1.26.0 (40646f47)
  creationTimestamp: null
  labels:
    io.kompose.service: nginx
  name: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      io.kompose.service: nginx
  strategy: {}
  template:
    metadata:
      annotations:
        kompose.cmd: kompose convert
        kompose.version: 1.26.0 (40646f47)
      creationTimestamp: null
      labels:
        io.kompose.service: nginx
    spec:
      containers:
        - image: nginx
          name: nginx
          ports:
            - containerPort: 80
          resources: {}
      restartPolicy: Always
status: {}
```

Kuva 11. Nginx-deployment.yaml -tiedosto komposen luomana

3.2.4 Kuormantasaus

Kuormantasaus kubernetes klusterissa voidaan hoitaa muuttamalla esimerkki.yaml-tiedostoa niin että kohtaan ”replicas” lisätään esimerkiksi numero 5. Nyt ajamalla tiedoston uudestaan komennolla ”kubectl apply -f nginx-service.yaml,nginx-deployment.yaml” käynnistyykin 5 kapselia samasta palvelusta, joihin kaikkiin johtaa sama IP-osoite, jota sitten hallitaan kuormanhallinnalla siten että jokaiselle kapselille jaetaan yhteyksiä. Kuormanhallinta pitää myös huolen, että jos joku kapseli kaatuu, käynnistetään se uudestaan (kuva 10).



Kuva 12. Viisi nginx-kapselia käynnissä

3.2.5 Tallennustila

Kuberneteksen tallennustilan käytöillä erilaisia tyyppejä. Kapselit voivat käyttää erilaisia tallennustila tyyppejä yhtä aikaa. Ephemeral volume eli lyhytaikainen tallennustila on olemassa vain kapselin elinaikana ja Persistent Volume eli pysyvä tallennustila on olemassa kapselin elinajasta riippumatta. (Kubernetes.io, 2023)

Kehitysympäristön palvelulle täytyi luoda pysyvä tallennustila. Pysyvää tallennustilaa pystyttiin käyttämään nginx-palvelun kapselien jakaman internet sivun jakamiseen niin että itse sivu sijaitsi pysyvässä tallennustilassa niin että kaikki kapselit käyttivät samaa sivua. Näin esimerkiksi pystytään päivittämään kaikkien kapseleiden jakama sivu kerralla vain päivittämällä tallennustilassa olevat tiedostot. (Kubernetes.io, 2023)

Resursseja PersistentVolume ja PersistentVolumeClaim käytetään varaamaan tallennustilaa käyttäjille ja pääkäyttäjille. Kun kapseli käyttää pysyvää tallennustilaa on se suojattu poistamiselta niin pitkään kuin se on käytössä, vaikka poistaja olisi pääkäyttäjä. (Kubernetes.io, 2023)

Asennus aloitettiin luomalla kansio ja index.html tiedosto worker-säikeelle klusterissa, jonne haluttiin luoda pysyvä tallennustila. Worker-säikeeseen otettiin SSH-yhteys ja luotiin komennolla ” sudo mkdir /mnt/data” tarvittava kansio, joka voi olla mikä vaan mitä järjestelmässä haluttiin käyttää. Yksinkertainen index.html luotiin kansioon komennolla ” sudo sh -c "echo 'Hello from Kubernetes storage' >

/mnt/data/index.html" ” , joka lisää tekstin ”Hello from Kubernetes storage” ja samalla luo tiedoston index.html kansioon. Tämän jälkeen testattiin, että tiedosto on luotu oikein komennolla ” cat /mnt/data/index.html”, jonka palautus on index.html-tiedoston sisältö eli ” Hello from Kubernetes storage”. (Kubernetes.io, 2023)

Kehitysympäristössä luotiin pysyvä tallennustila isäntäpolkuun, mutta tuotanto käytössä pääkäyttäjä loisi verkkoresurssin, kuten Google Compute Engine pysyvän levyn, NFS jaon, tai Amazon Elastic Block Store levyn. Kuvan 12 konfiguraatio tiedostossa nähdään, että tallennustila on klusterin säikeessä osoitteessa ”/mnt/data”. Konfiguraatiossa myös annetaan maksimi kapasiteetti tallennustilalle joka tässä tapauksessa oli 5Gi ja ”accessModes”-kohdassa määritellään ”ReadWriteOnce” eli se voi olla kiinnitettynä kerrallaan yhteen säikeeseen. ”StorageClassName” määrittelee luokan tallennustilalle, jota voidaan käyttää PersistentVolumeClaim-pyyntöjen sitomiseen tähän pysyvään tallennustilaan. (Kubernetes.io, 2023)

pods > storage > ! pv-volume.yaml

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: task-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

Kuva 13. Pysyvän tallennustilan pv-volume.yaml

Komennolla ” kubectl apply -f pv-volume.yaml” käynnistetään pysyvän tallennustilan kapseli ja komennolla ” kubectl get pv task-pv-volume” voidaan tarkastaa kapselin tila. Aluksi ”STATUS”-osiossa tilana oli Available. (Kubernetes.io, 2023)

```
ubu1@ubu1:~$ kubectl get pv task-pv-volume
NAME             CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                   STORAGECLASS
task-pv-volume   5Gi        RWO            Retain           Bound    default/task-pv-claim   manual
```

Kuva 14. Pysyvän tallennustilan tarkistus komento

PersistentVolumeClaim kapselin luominen tapahtui käynnistämällä ”pv-claim.yaml”-kapseli komennolla ” kubectl apply -f pv-claim.yaml”. Kun PersistentVolumeClaim-kapseli käynnistetään kuberneteksen ohjaustaso etsii sopivaa PersistentVolume-kapselia, jossa on määritelty sama ”StorageClass” ja sitoo sen sitten siihen tallennustilaan. Komennolla ”kubectl get pv task-pv-volume” nähdään onko

komento onnistunut kun tallennustilan ”STATUS”-osio on muuttunut muotoon ”Bound”. (Kubernetes.io, 2023)

```
ubu1@ubu1:~$ kubectl get pvc task-pv-claim
NAME            STATUS   VOLUME           CAPACITY   ACCESS MODES   STORAGECLASS
task-pv-claim   Bound    task-pv-volume   5Gi        RWO            manual
```

Kuva 15. Pysyvän tallennustilan varauksen tarkistus

```
pods > storage > ! pv-claim.yaml
1   apiVersion: v1
2   kind: PersistentVolumeClaim
3   metadata:
4     name: task-pv-claim
5   spec:
6     storageClassName: manual
7     accessModes:
8       - ReadWriteOnce
9     resources:
10      requests:
11        storage: 1Gi
```

Kuva 16. Pysyvän tallennustilan pv-claim.yaml

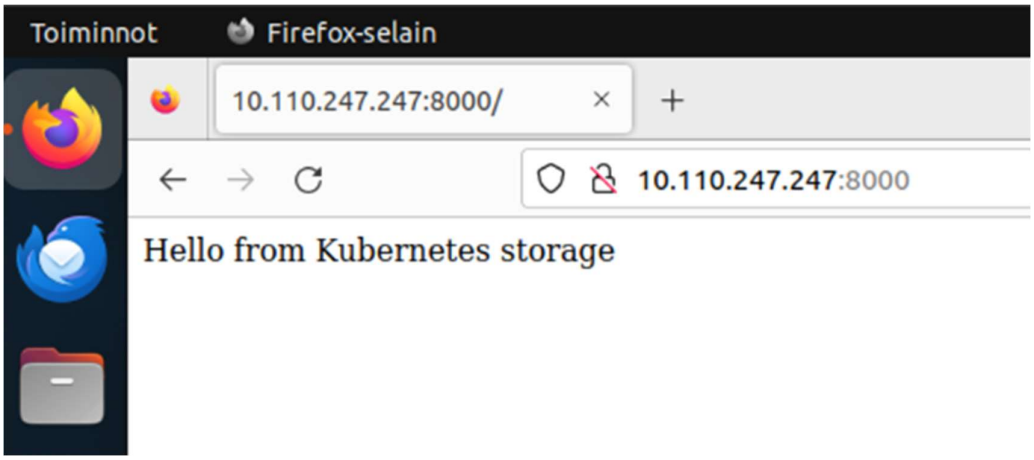
Lisäämällä pysyvän tallennustilan tiedot ”nginx-deployment.yaml”-tiedostoon saadaan nginx-palvelu käyttämään aiemmin luotua pysyvää tallennustilaa(Kuva 16). Kun tiedostoa oli muokattu komennolla ”kubectl apply -f nginx-service.yaml,nginx-deployment.yaml” käynnistettiin kapselit uudestaan. Varmistettiin toiminta menemällä virtuaalikoneen selaimella osoitteeseen ”10.110.247.247:8000” oli käytössä aikaisemmin luotu ”index.html”-sivu. Sivu voidaan päivittää vain muokkaamalla ”index.html”-tiedostoa pysyvässä tallennustilassa. (Kubernetes.io, 2023)

Jos käytössä on kapseleita, joilla on tarvetta kirjoittaa tietoa tallennustilaan, voidaan kapseleille luoda ryhmä ID (GID), joka antaa kirjoitusluvan kapseleille, joilla on oikea GID. Väärät tai puuttuvat GID:t aiheuttavat virheen. Lisäämällä GID:n pysyvän tallennustilan kapselin konfiguraatio tiedostoon GID periytyy tallennustilaa käyttäville kapseleille automaattisesti. (Kubernetes.io, 2023)

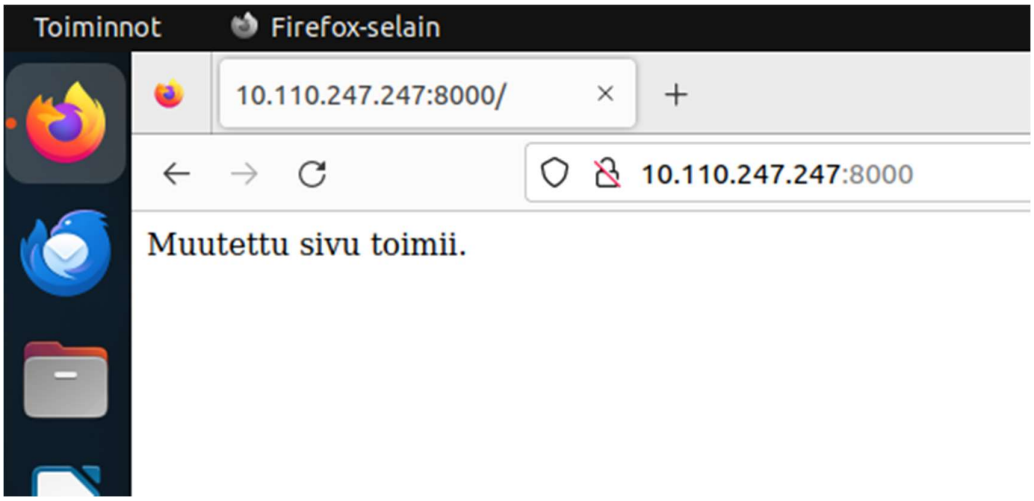
compose > ! nginx-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    kompose.cmd: kompose convert
    kompose.version: 1.26.0 (40646f47)
  creationTimestamp: null
  labels:
    io.kompose.service: nginx
  name: nginx
spec:
  replicas: 5
  selector:
    matchLabels:
      io.kompose.service: nginx
  strategy: {}
  template:
    metadata:
      annotations:
        kompose.cmd: kompose convert
        kompose.version: 1.26.0 (40646f47)
      creationTimestamp: null
      labels:
        io.kompose.service: nginx
    spec:
      volumes:
        - name: task-pv-storage
          persistentVolumeClaim:
            claimName: task-pv-claim
      containers:
        - image: nginx
          name: nginx
          ports:
            - containerPort: 80
          volumeMounts:
            - mountPath: "/usr/share/nginx/html"
              name: task-pv-storage
          resources: {}
      restartPolicy: Always
status: {}
```

Kuva 17. Nginx konfiguraatio tiedosto pysyvällä tallennustilalla



Kuva 18. Pysyvän tallennustilan toiminnan testaus nginx-kapselissa



Kuva 19. Päivitetty sivu

3.2.6 Salaisuuksien hallinta

Salaisuudet kuberneteksessa ovat objekteja, joihin tallennetaan arkaluontoisia asioita kuten salasanoja, OAuth-tunnisteita ja SSH-avaimia. Salaisuuksilla hallitaan kuinka arkaluontoista tietoa käytetään ja vähennetään vahinko altistumisen riskiä. Salaiset arvot koodataan base64-merkkijonoiksi, ja ne tallennetaan oletusarvoisesti salaamattomina, mutta ne voidaan määrittää leposalattaviksi. (Kubernetes.io, 2023)

Kapselissa voidaan viitata salaisuuteen eri tavoin kuten tallennustilan kiinnityksessä tai ympäristömuuttujassa. Salaisuudet ovat suunniteltu luottamuksellisille tiedoille ja ei-luottamuksellisille tiedoille on ConfigMap. ConfigMap on konfiguraatioasetusten sanakirja, joka koostuu merkkijonojen avain-arvopareista, jotka asetetaan konteille. Konfiguraatio tallennetaan YAML-tiedostoon, joka käynnistetään kapselina ja luetaan sitä käyttävää kapselia käynnistettäessä joko tallennustilasta tai suoraan muuttujana kapselin YAML-tiedoston kautta. (Mathewpalmer.net, 2024)

Käyttäjä, joka voi luoda kapselin, joka käyttää salaisuutta, voi myös nähdä kyseisen salaisuuden arvon. Vaikka klusterikäytännöt eivät sallisi käyttäjän lukea salaisuutta suoraan, sama käyttäjä voi saada oikeuden käyttää kapselia, joka paljastaa salaisuuden. Suosituksiin kuuluu, että käytettäisiin lyhytikäisiä salaisuuksia ja otettaisiin käyttöön tarkastussäännöt, jotka varoittavat tietyistä tapahtumista, kuten useiden salaisuuksien samanaikaisesta lukemisesta yhden käyttäjän toimesta. (Kubernetes.io, 2023)

Salaisuuksia voidaan käyttää esimerkiksi konttikuvan lataamiseksi yksityisestä konttikuva rekisteristä tai arkistosta (repository). Yksityisiä konttikuva rekistereitä on monia kuten docker hub. Docker hub:n käyttöön tarvitaan docker-tunnus. Docker-tunnuksia käyttämällä saadaan auktorisointitunniste, jota käytetään kubernetes-salaisuuden luomiseen. Kun salaisuus on luotu, voidaan luoda kapseli, joka käyttää salaisuutta konttikuvan lataamiseen rekisteristä. (Kubernetes.io, 2023)

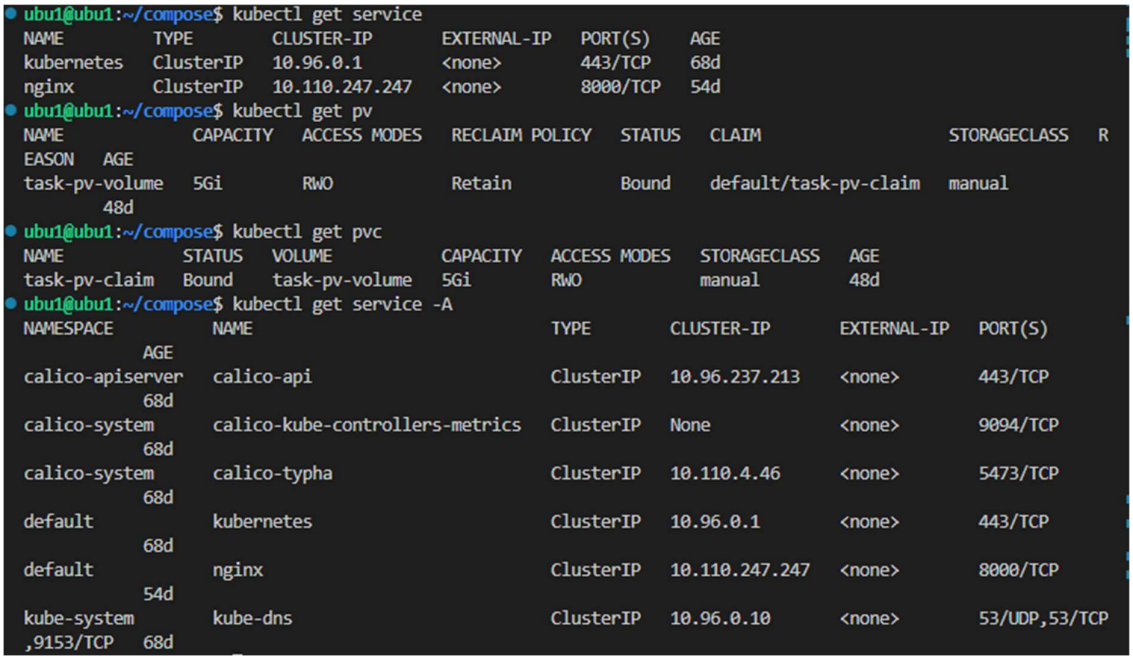
4 YHTEENVETO

Kubernetes-klusterin asennuksessa täytyy ottaa huomioon etukäteen millaista ympäristöä ollaan rakentamassa ja mitä tarvitaan. Kubernetes asennuksessa verkon asetuksien suunnittelu on tärkeää, että kaikki klusterin osat keskustelevat toistensa kanssa oikein. Jos ei haluta joidenkin kapseleiden keskustelevan joidenkin tiettyjen kapseleiden kanssa täytyy ottaa se huomioon reitityksen suunnittelussa. Kun on saatu kaikki klusterin solmut näkymään master-solmun näkymässä, voidaan asentaa reititys- ja kuormanhallintakapseli, joka yhdistää solmut ja asettaa ne ”Ready”-tilaan. Reititys-palvelulle ilmoitetaan mihin verkkoon luodaan käytettävät kapselit ja palvelut. Vaikka palvelut näkyvät yhdessä IP-osoitteessa ovat palvelun kapselit omissa IP-osoitteissaan.

Kun muunnetaan komposella docker konttien compose-tiedostoja on myös tärkeää ensin ottaa huomioon, tarvitaanko tallennustilaa. Luodaan tarvittavat tallennustilojen kapselit ennen komposen ajamista, jotta kompose tunnistaa tallennustilat ja osaa luoda YAML-tiedoston, johon tallennustila on sisällytetty oikein. Jos tehdään kehitystyötä voi olla helpompi ensin aloittaa karsitummalla ”.yaml”-tiedostolla ja laajentaa kun ensin ollaan saatu toimivat karsitummat asetukset.

Kuberneteksen käytön aikana tärkeitä komentoja:

- ”kubectl get nodes” näyttää säikeiden tilan
- ”kubectl get pods” näyttää kapseleiden tilan
- ”kubectl get deploy” näyttää palveluiden tilan
- ”kubectl delete deploy ’palvelun nimi’” sammuttaa palvelun
- ”kubectl delete pod ’kapselin nimi’” sammuttaa yksittäisen kapselin
- ”kubectl apply -f jokin.yaml” käynnistää YAML-tiedoston kapselin tai palvelun
- ”kubectl get pv” näyttää pysyvät tallennustilat
- ”kubectl get pvc” näyttää pysyvien tallennustilojen varaukset
- ”kubectl delete pv ’tallennustilan nimi’” sammuttaa pysyvän tallennustilan kapselin
- ”kubectl delete pvc ’tallennustilan varauksen nimi’” sammuttaa pysyvän tallennustilan varauksen kapselin



Kuva 20. Esimerkki tietojen näyttämisestä master-säikeessä

5 POHDINTA

Opinnäytetyön tavoitteena oli tutustua kubernetes-ympäristöön ja samalla selvittää mahdollisuutta siirtää jo valmiiden docker-konttien muuttamista muotoon, joita voidaan käyttää kuberneteksessa. Tavoitteena oli myös toteuttaa dokumentaatiota kuberneteksen asennuksesta, jota voitaisiin käyttää, jos päätetään muuttaa docker-ympäristö kubernetes-ympäristöksi. Koska kubernetes on hyvin dokumentoitu, suurin osa lähteistä on kuberneteksen omista dokumentaatioista. Kuberneteksen dokumentaatiosta löytyy jokaisesta osa-alueesta tarkat dokumentaatiot ja työtä tehdessä joutuikin käyttämään paljon aikaa tärkeän tiedon siivilöimiseen epäolennaisen tiedon joukosta. Uutena asiana myös dockerin toimintaa täytyi tutkia ja selvittää mitä tarvitsee tietää, jotta muutos kubernetekseen on mahdollinen.

Virtuaalikoneiden ja Linux-ympäristön käyttö oli entisestään tuttua mutta samalla työtä tehdessä oppi hyviä tapoja virtuaalikoneiden käytöstä ja kuinka asiat voi tehdä helposti. Tästä hyvä esimerkki on SSH-yhteyden käyttäminen Visual Studio Code ohjelmiston kautta niin että saadaan helposti käytettyä virtuaalikoneita käynnistämättä niitä kokonaan omiin ikkunoihinsa. Samalla saadaan käyttöön Visual Studio Code ohjelmiston ominaisuuksia ohjelmointityökaluna, kun käsitellään YAML-tiedostoja. Voitiin samalla simuloida tilannetta, että koneet, joita halutaan käyttää ovat esimerkiksi jossain pilvipalvelussa tai erillisellä serverillä jossain eri verkossa. Myös Linux-ympäristön komentokehote tuli tässä entistä tutummaksi.

Kubernetes ja docker ovat todella laajoja aiheita, joihin voi uppoutua todella syvälle. Opinnäytetyötä tehdessä täytyikin olla tarkkana, ettei vajoa liian syvälle dokumentaatioon, kun asioita selvitti. Työtä tehdessä kuitenkin luulen, että opin asiasta todella paljon.

LÄHTEET

Docker. Docker docs. https://docs.docker.com/ Viitattu 9.1.2024

Kompose.io. Kompose – Convert your Docker Compose file to Kubernetes or OpenShift. https://kompose.io/ Viitattu 30.1.2024

Kubernetes.io. Kubernetes Documentation | Kubernetes. https://kubernetes.io/docs/home/ Viitattu 9.1.2024

Kubernetes.io. Install and Set Up kubectl on Linux | Kubernetes. https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/ Viitattu 1.9.2024

Kubernetes.io. Persistent Volumes | Kubernetes. https://kubernetes.io/docs/concepts/storage/persistent-volumes/ Viitattu 30.1.2024

Kubernetes.io. Good practices for Kubernetes Secrets | Kubernetes. https://kubernetes.io/docs/concepts/security/secrets-good-practices/ Viitattu 2.2.2024

Mathewpalmer.net. Ultimate Guide to ConfigMaps in Kubernetes - Kubernetes Book. https://matthewpalmer.net/kubernetes-app-developer/articles/ultimate-configmap-guide-kubernetes.html Viitattu 5.2.2024

Microsoft Azure. What is PaaS? Platform as a Service | Microsoft Azure. Retrieved from https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-paas Viitattu 31.1.2024

Tigera.io. About Calico | Calico Documentation. Retrieved from https://docs.tigera.io/calico/latest/about/ Viitattu 28.1.2024