Haaga-Helia
ammattikorkeakoulu Oy

Sovelluksen käyttöönotto OpenShift-konttialustalle

Suvi Vappula

Opinnäytetyö
Ohjelmistotuotanto
2021

Tiivistelmä

Haaga-Helia
ammattikorkeakoulu Oy

Tekijä(t)
Suvi Vappula.

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Raportin/Opinnäytetyön nimi
Sovelluksen käyttöönotto OpenShift-konttialustalle

Sivu- ja liitesivumäärä
32 + 0

Tämän opinnäytetyön aiheena on sovelluksen käyttöönotto OpenShiftille ja OpenShift yleisesti. Tavoitteena on antaa tietoa OpenShiftin toiminnasta ja peruskäsitteistä uusille suomenkielisille OpenShiftin käyttäjille.

Aluksi käydään läpi mikä on OpenShift ja mitä ovat kontit. Työssä käydään myös läpi OpenShiftin historiaa ja miten Kubernetes- ja Docker-teknologioita on käytetty hyväksi OpenShiftin luomisessa. Kolmannessa kappaleessa on selitetty lyhyesti OpenShiftin arkkitehtuuria ja sen eri kerroksia. Arkkitehtuurin jälkeen siirrytään peruskäsitteisiin, joita pitää ymmärtää OpenShiftiä käytettäessä. Viimeiseksi teoriaosiossa käydään läpi eri tapoja asentaa sovelluksia.

Tutkimusosassa asennetaan OpenShiftille Drupal-sisällönhallintajärjestelmä ja tietokantana MariaDB. Tämä vastaa tutkimuskysymykseen: miten sovellus asennetaan OpenShiftille? Tutkimus on kvalitatiivinen, sillä se käsittelee vain yhden sovelluksen (ja tietokannan) asennusta yhdellä tavalla.

Päätelmät ja johtopäätökset -osiossa pohditaan työssä tehdyn asennuksen verrattavuutta oikeisiin projekteihin ja kuinka hyvin työssä vastataan tavoitteeseen antaa tietoa aloitteleville OpenShiftin käyttäjille.

Asiasanat
OpenShift, konttiteknologia, Kubernetes, Docker

Sisällys

Käsitteet

Backend	Taustalla toimiva ohjelma, joka käsittelee tietoa frontendistä esim. kirjautuminen ja lomakkeiden käsittely.
CaaS	Kontti palveluna. Tarjoaa käyttäjälle samat kyvykkyydet kuin infrastruktuuri palveluna, mutta myös konttimoottorit ja orkestrointikyvyn.
DNS	Nimipalvelu. Muuntaa verkkotunnukset IP-osoitteiksi ja päinvastoin.
Dockerfile	Tiedosto, joka sisältää tarvittavat käskyt levykuvan rakentamiseen
Frontend	Käyttäjälle näkyvä osa ohjelmasta, joka esittää backendistä tulevaa tietoa
Git	Versionhallintajärjestelmä
IaaS	Infrastruktuuri palveluna. Tarjoaa käyttäjälle palvelimet, virtuaalikoneet, tietoverkot ja muistin.
Klusteri	Tietokoneryväs. Joukko tietokoneita, jotka jakavat tehtäviä.
Node	Joko virtuaalinen tai fyysinen kone, joka tarjoaa podille ajonaikaisen ympäristön
PaaS	Sovellusalusta palveluna. Tarjoaa käyttäjille alustan, jolla kehittää, suorittaa ja hallita sovelluksia.
Podi	Yksi tai useampi kontti, joilla on jaettu muisti ja tietoverkkoresurssit sekä määrittelyt, kuinka kontteja ajetaan.
Portti	Portti määrittelee ohjelman, johon otetaan yhteyttä tietokoneessa

1 Johdanto

Konttiteknologioiden yleistyessä OpenShift on varteenotettava vaihtoehto yrityksille siirtyä konttimaailmaan turvallisesti. Se tarjoaa tavan hallita kontitettuja sovelluksia laajemmalla skaalalla tehden sovellusten käyttöönotosta helpompaa ja nopeampaa.

Opinnäytetyössä on tarkoituksena käydä läpi mikä OpenShift on, sen peruskäsitteitä ja toimintoja sekä vastata tutkimuskysymykseen; kuinka sovellus asennetaan OpenShiftille. Opinnäytetyö on suunnattu aloitteleville OpenShift-käyttäjille, auttamaan heitä sisäistämään peruskäyttöä ja konsepteja.

Kyseinen aihe valikoitui sekä omasta että kollegoiden tarpeesta ymmärtää OpenShiftiä. Alusta tuli itselleni ja muille uutena jokapäiväiseen käyttöön syksyllä 2020. Kubernetestä aiemmin käyttäneille alusta oli helpompi sisäistää, kun taas osalle, kuten minulle, alustan tuomat konseptit olivat täysin uusia.

Opinnäytetyön teoriaosiossa käydään läpi OpenShiftin historiaa, arkkitehtuuria, peruskäsitteitä ja tapoja asentaa sovelluksia. Tutkimusosiossa käydään läpi kuinka asentaa Drupal ja MariaDB OpenShiftille. Sovellukseksi valikoitui Drupal, koska se on suosittu alusta yritysten verkkosivuille. Drupalille löytyy myös valmiita Docker-levykuvia, joita muokkaamalla saadaan OpenShiftillä toimiva kontti.

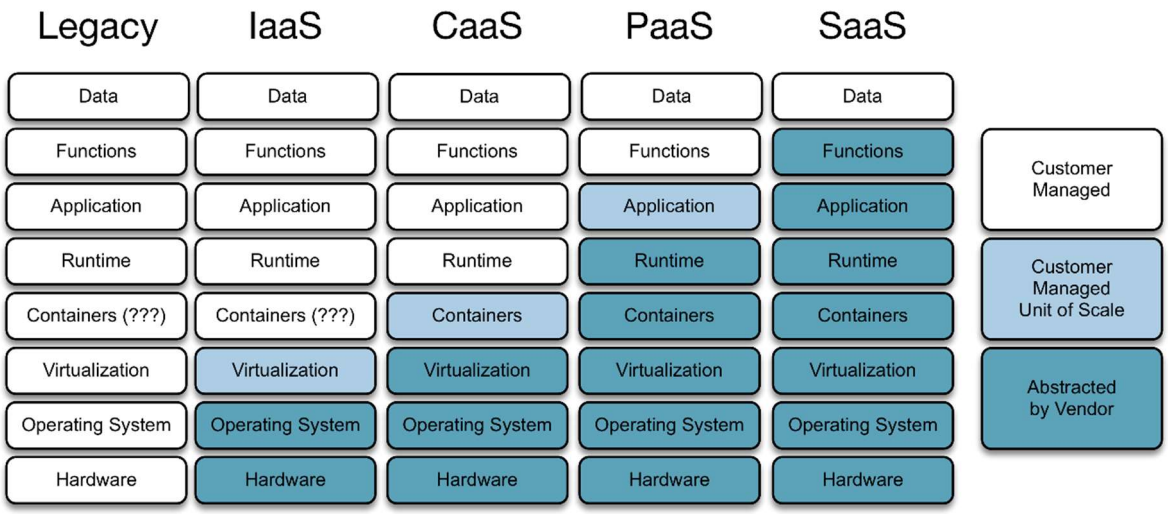
2 Mikä on OpenShift

2.1 OpenShift

OpenShift on sovelluksien kehitykseen ja julkaisuun kehitetty alusta. Julkaistava sovellus voi olla julkinen web-sovellus tai backend-sovellus, sisältäen myös mikropalvelut ja tietokannat. Sovelluksen ohjelmointikielellä ei ole väliä, mutta sovellus pitää pystyä ajamaan kontissa. (Dumpleton 2018, luku 1.)

Puhuttaessa pilvipalveluiden palvelumalleista, OpenShift voidaan sijoittaa sekä perinteisempään sovellusalusta palveluna (Platform as a Service) että uudempaan kontti palveluna (Container as a Service) -luokitteluun. OpenShift tarjoaa tavan ottaa käyttöön sovellus rakentamalla konttilevykuvan (container image) suoraan sovelluksen lähdekoodista. Tästä on lähtöisin ajattelu OpenShiftin kuulumisesta alusta palveluna -luokitteluun. (OpenShift Cookbook a.)

OpenShiftin luokittelu kontti palveluna -palvelumalliin lähtee mahdollisuudesta antaa OpenShiftille valmis konttilevykuva, joka sisältää sovelluksen sekä sen tarvitsemat kirjastot ja riippuvuudet. Palvelumallia voidaan pitää samankaltaisena infrastruktuuri palveluna (Infrastructure as a Service) -mallin kanssa. Erona on, että virtuaalikoneen sijaan tarjotaan kontti. (Dumpleton 2018, luku 1.) Alla olevassa kuvassa (kuva 1) näkyy, mihin kontti palveluna (CaaS) sijoittuu perinteisempien mallien joukossa.

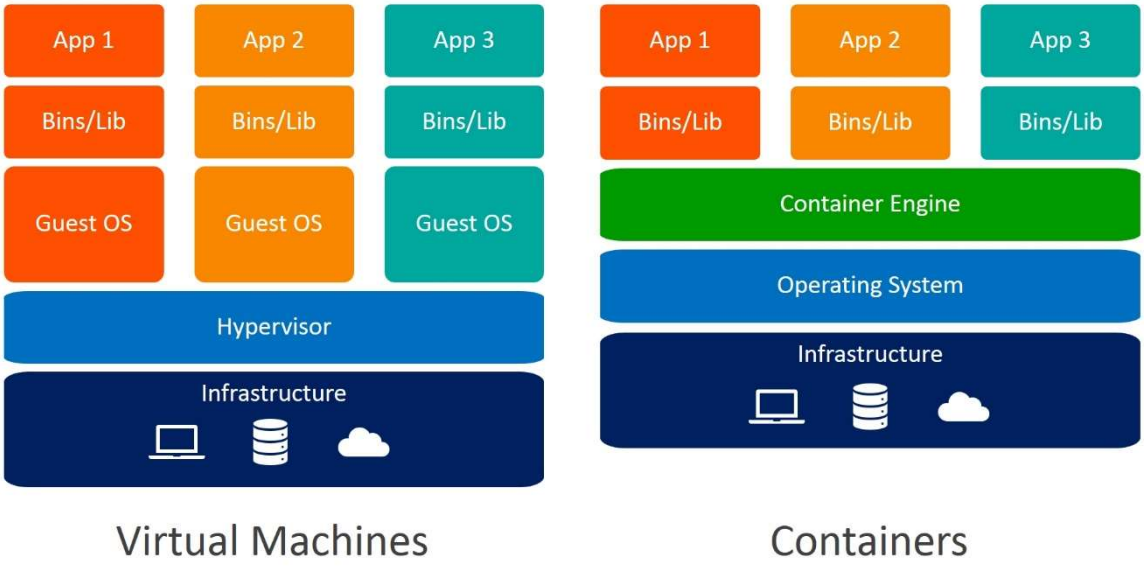


Kuva 1. Pilvipalvelumallit (Deploying OpenShift)

2.2 Kontit

Ymmärtääkseen OpenShiftiä, pitää tietää mitä kontit ovat. Ne ovat suoritettavia ohjelman yksiköitä, joissa sovelluksen koodi, kirjastot ja riippuvuudet paketoidaan niin, että ohjelma voidaan suorittaa missä tahansa tietokoneessa tai jopa pilvessä (IBM 2019).

Kontteja ymmärtää ehkä paremmin oppimalla, mikä erottaa ne virtuaalikoneista. Perinteisessä virtuaalikoneessa on vieraskäyttöjärjestelmä, virtuaalinen kopio laitteistosta, jota käyttöjärjestelmä tarvitsee, sekä sovellus ja sen kirjastot ja riippuvuudet. Konteissa ei virtualisoida laitteistoa, vaan koneen oma käyttöjärjestelmä niin, että kontti sisältää vain sovelluksen sekä sen kirjastot ja riippuvuudet. Vieraskäyttöjärjestelmän puuttuminen tekee konteista kevyitä, nopeita ja siirreltäviä. (IBM 2019.)



Kuva 2. Virtuaalikoneet vs. kontit (Linux Pipeline)

2.3 Docker

Konttiteknologia on ollut olemassa jo pitkään, mutta sen suosio alkoi kasvaa vasta vuonna 2013, kun dotCloud toi markkinoille Dockerin. Docker auttoi ratkaisemaan kaksi tärkeää ongelmaa kontituksessa: konttilevykuvan paketoinnin formaatin määrittäminen ja kuvan rakentamiseen tarvittavien työkalujen tarjoaminen. Dockerin avulla pystyttiin siis helposti luomaan sovelluksista levykuvia, joita pystyttiin siirtelemään eri järjestelmien välillä ja sitten ajamaan konteissa. (Dumpleton 2018, luku 1.)

OpenShift, jonka ensimmäinen versio julkistettiin vuonna 2011, käytti versioissaan v1 ja v2 Linux-kontteja sovellusten ajamiseen ja julkaisemiseen. Kun Docker julkaistiin, Red Hat

alkoi tukea Dockerin avoimen lähdekoodin projektia ja toi Docker-tuen Red Hat Enterprise Linux 7:ään, josta tulikin pohja OpenShift 3:lle. (Fernandes 7.11.2016.)

2.4 Kubernetes

Vuonna 2014 Google ilmoitti kehittävänsä Kubernetestä, avoimen lähdekoodin ohjelmaa, joka automatisoi kontitettujen sovellusten julkaisun, skaalauksen ja hallinnoinnin. Red Hat, joka oli pyrkinyt samaan rakentamalla omaa orkestrointikerrosta Dockerin ympärille, päätti lopettaa oman projektinsa ja ottaa käyttöön Kuberneteksen. Tämä ratkaisi OpenShiftin ongelman pyörittää suurempaa määrää kontteja. (Dumpleton 2018, luku 1.)

2.5 OpenShift V3 ja V4

Kehittäessään OpenShiftin kolmatta versiota Red Hat siis otti pohjaksi sekä Dockerin konttiteknologian että Kuberneteksen tuoman orkestrointikyvykkyyden. Näiden päälle OpenShiftiin rakennettiin myös käyttöliittymä, jolla pystyy nopeasti luomaan ja julkaisemaan sovelluksia Source-to-image ja julkaisuputki (pipeline) -teknologioilla. (McConville & Wagoner 1.8.2020.)

OpenShiftin nykyinen versio on OpenShift 4. Siihen on lisätty monia uusia ominaisuuksia kuten OpenShift Service Mesh, Red Hat Enterprise Linux CoreOS, pilviautomaatio ja paljon muuta. OpenShift Service Mesh esimerkiksi yhdistää Jaegerin, Istion ja Kialin yhdeksi kyvykkyydeksi, joka auttaa mm. analysoinnissa, monitoroinnissa, mikropalveluiden yhdistämisessä ja näkyvyydessä sekä tarjoaa topologian ja tiedon mikropalveluiden terveydestä. Aiemmin palvelimet vaativat yksilöityä huolenpitoa, mutta nyt Red Hat Enterprise Linux CoreOS mahdollistaa viallisen palvelimen korvaamisen suoraan toisella palvelimella. Kolmantena esimerkkinä OpenShift 4:n uusista ominaisuuksista on pilviautomaatio, jolla tarkoitetaan sitä, että OpenShift 4 on suunniteltu sekä monenlaisille eri pilviympäristöille ja hybridi-pilviympäristöille, jotka voivat olla sekä omassa konesalissa (on-premise) että pilvessä. (Dharmalingam 11.10.2019.) Nämä ovat vain muutamia OpenShift 4:n uusista ominaisuuksista.

2.6 OpenShift vs. Kubernetes

Konttiorkestraatioalustoista tunnetuimpia ovat Kubernetes ja OpenShift. Näitä vertaillessa on hyvä muistaa, että Kubernetes on osa OpenShiftiä, mutta ei toisinpäin. Molemmat ovat siis avoimen lähdekoodin alustoja, jotka ovat tarkoitettu sovellusten kehittämiseen ja hallintaan. Kubernetes ryhmittelee kontit loogisiin klustereihin hallinnan ja löytymisen

helpottamiseksi. OpenShift taas antaa kehittäjien rakentaa ja ottaa käyttöön Docker-muotoisia kontteja, joita hallitaan Kubernetes-alustan avulla. (Maayan 6.4.2020.)

Kubernetes on avoimen lähdekoodin projekti, kun taas OpenShift on kaupallinen tuote, jolle saa maksullisen tuen. OpenShiftillä on vahvemmat turvallisuuskäytänteet kuin Kuberneteksella, mutta nämä käytänteet vaativat oppimista. Autentikoinnin asennus ja konfigurointi vaatii Kuberneteksella enemmän työtä kuin OpenShiftillä, jossa on integroitu palvelin autentikaatiota varten. (Maayan 6.4.2020.)

OpenShiftillä on käyttöliittymä, johon kirjautumalla näkee helposti kaiken tarvittavan ja jossa pystyy tekemään muutoksia. Kubernetekseen saa myös käyttöliittymän, mutta se pitää asentaa erikseen. Kirjautumiseen tarvittava tunnisteväline (token) pitää luoda manuaalisesti, koska Kuberneteksella ei ole kirjautumissivua toisin kuin OpenShiftillä. OpenShiftillä on myös sisäinen levykuvarekisteri, jollaista Kuberneteksella ei ole, mutta Kubernetekselläkin voi halutessaan asentaa oman Docker-rekisterin. (Maayan 6.4.2020.)

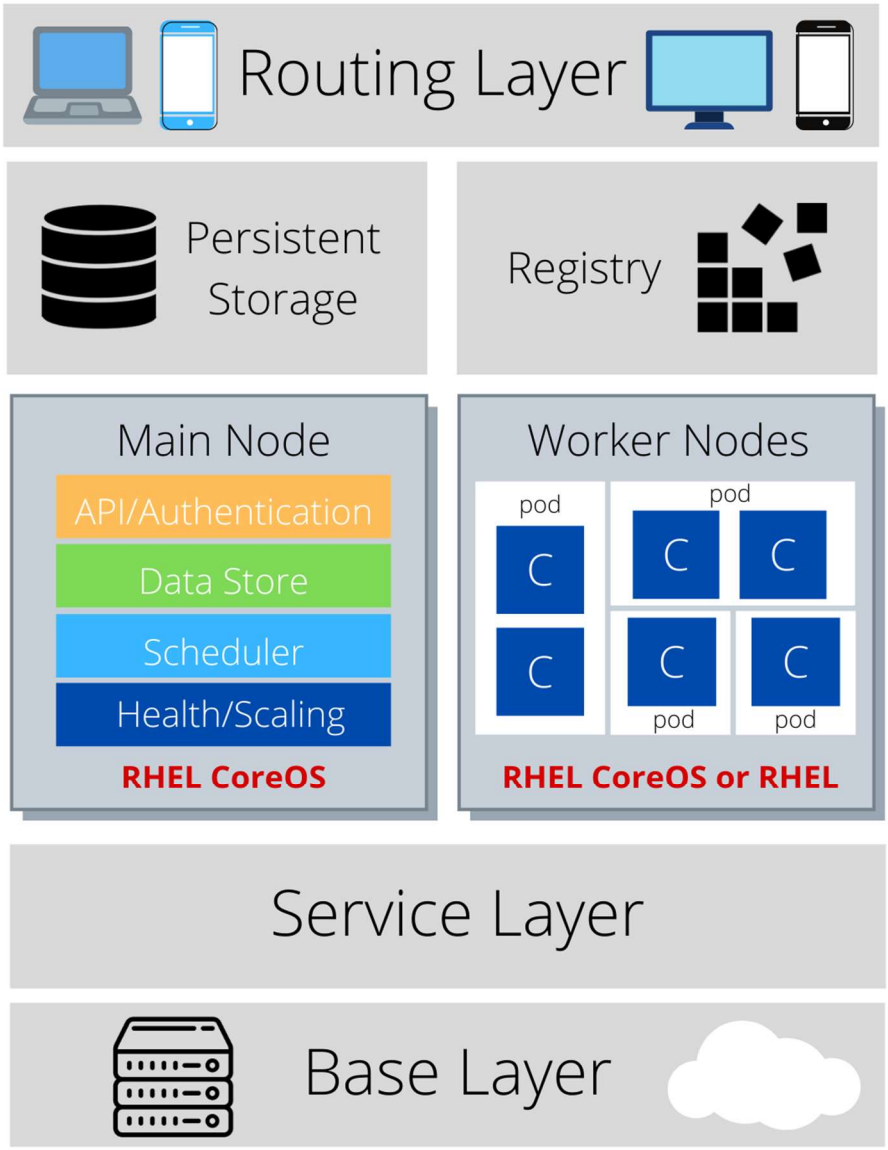
Yksi isoimmista OpenShiftin eduista Kubernetekseen verrattuna on kuitenkin asennuksen helppous. Kubernetes on avoimen lähdekoodin ohjelma, mutta sitä ei kannata asentaa suoraan lähdekoodista. Kubernetes-asennus vaatii paljon aikaa ja monia komponentteja ja sen konfiguraatio on monimutkainen. Tämän vuoksi Kuberneteksen asentamiseen käytetään Kubernetes-jakeluja, kuten esimerkiksi OpenShiftiä, Canonical Kubernetestä tai Rancheria. (Tozzi 4.3.2020.)

Kubernetes on saatavissa palveluna mihin tahansa isolle pilvialustalle esimerkiksi Azurelle tai AWS:lle (Amazon Web Services) tai voidaan asentaa mille tahansa Linux-jakelulle esim. Ubuntu ja Debian. OpenShift vaatii Red Hat Enterprise Linux (RHEL), Red Hat CoreOS tai CentOS jakelun asennukseen. (Sengupta 3.11.2020.)

3 OpenShiftin arkkitehtuuri

3.1 Arkkitehtuuri

OpenShiftin arkkitehtuuri koostuu useasta eri komponentista. On infrastruktuurikerros (infrastructure layer), palvelukerros (service layer), päänode (main node), työntekijänodet (worker nodes), rekisteri (registry), pysyväistallennus (persistent storage) ja reitityskerros (routing layer). (Abushamleh 19.8.2020.)



Kuva 3. OpenShift arkkitehtuuri. IBM Developer Blog. 2020.

Infrastruktuurikerroksessa, joka on kuvassa (kuva 3) nimellä Base Layer, on sovelluksen hostaus palvelimilla. Nämä palvelimet voivat olla fyysisiä, virtuaalisia, tai yksityisessä tai julkisessa pilvessä. Infrastruktuuri kerroksen päällä on palvelukerros, joka antaa podeille pysyvän IP-osoitteen ja isäntänimen (host name). Se myös yhdistää eri sovellukset ja tarjoaa sisäisen kuormituksen tasaajan (load balancing). (Abushamleh 19.8.2020.)

Päänode hallinnoi klusteria ja työntekijänodeja. Sillä on neljä päätehtävää: 1. API-pyynnöistä ja autentikoinnista huolehtiminen; 2. ympäristöön ja sovellukseen liittyvän informaation ja niiden tilan varastointi; 3. podien sijoittaminen muistin, prosessorin ja muiden ympäristötekijöiden käyttöasteen perusteella työntekijänodeille; 4. podien terveyden monitorointi ja niiden skaalaaminen käyttöasteen perusteella. (Abushamleh 19.8.2020.)

Päänoden lisäksi on olemassa työntekijänodeja. Jokainen työntekijänode sisältää kolme asiaa: 1. CRI-O-konttimoottorin (container engine), joka ajaa ja hallinnoi kontteja; 2. kubeletin, joka ottaa vastaan konttien hallinnointiin liittyviä pyyntöjä päänodelta; 3. palvelun välityksen (service proxy), joka vastaa eri podien välisestä kommunikaatiosta kaikkien työntekijänodejen välillä. (Red Hat OpenShift Documentation a.) Työntekijänodet hallinnoivat podeja edellä mainittujen työkalujen avulla. Podi puolestaan on pienin määriteltävissä oleva, käyttöön otettava ja hallinnoitava yksikkö. Podissa voi olla yksi tai useampi kontti. Kaikilla samassa podissa olevilla konteilla on sama IP-osoite. (Abushamleh 19.8.2020.)

Arkkitehtuuriin kuuluvat myös rekisteri ja pysyväismuisti (persistent volume). Rekisteri tallentaa levykuvat paikallisesti klusteriin, kun taas pysyväismuisti on paikka, johon kaikki säilytettävä data tallennetaan. Pysyväismuisti on myös liitettynä kontteihin, sillä kun kontteja käynnistetään uudestaan tai poistetaan, niiden data katoaa, ellei sitä ole liitetty pysyväismuistiin. (Abushamleh 19.8.2020.)

Ylimpänä kuvassa 3 näkyy reitityskerros. Se tarjoaa pääsyn ulkoisen pääsyn klusterin sovelluksiin miltä tahansa päätelaitteelta (Abushamleh 19.8.2020).

4 Peruskonsepteja

4.1 Deployments

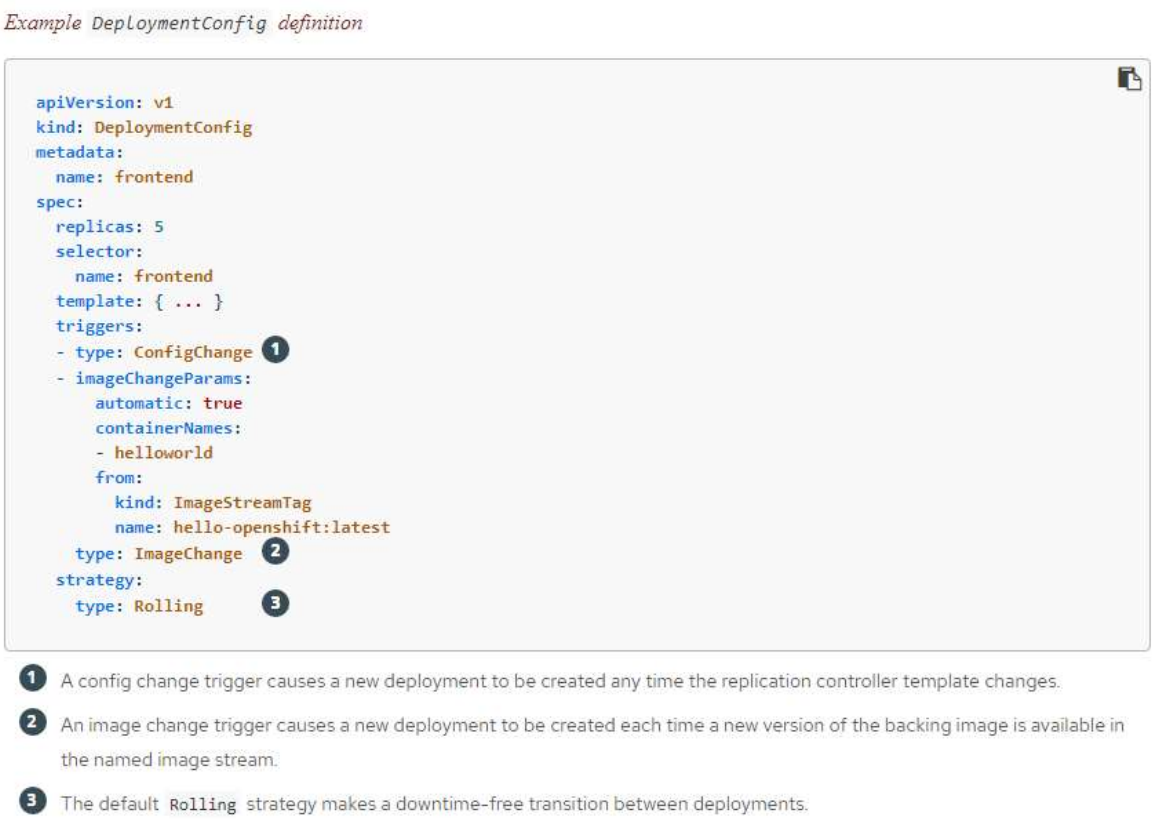
Deploymentit ovat Kuberneteksestä lähtöisin oleva tapa hallita sovelluksen päivitystä. Tyypillisesti osana deploymentia on ReplicaSet ja podit. ReplicaSetin tehtävä on varmistaa, että tarvittava määrä podeja on koko ajan käytössä. Jos yksi pod kuolee, ReplicaSet luo toisen sen tilalle. Deployment tehdään siis tyypillisesti, jos tarvitsee esimerkiksi päivittää uudempi image podeille. Deploymentin avulla varmistetaan, että muutos ei aiheuta katkoksia palveluun. Päivitys, jota kutsutaan nimellä ”rolling update”, korvaa kaikki podit yksi kerrallaan uudemmilla versioilla, pitäen kuitenkin elossa vähintään niin monta podia kuin Deploymentissa on määritelty. Se varmistaa myös samalla, että kaikki liikenne kohdistetaan toimiviin podeihin. (Docs SCS 2020.)

Deployment definition

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-openshift
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-openshift
  template:
    metadata:
      labels:
        app: hello-openshift
    spec:
      containers:
      - name: hello-openshift
        image: openshift/hello-openshift:latest
        ports:
        - containerPort: 80
```

Kuva 4. Esimerkki Deploymentin määrittelystä OpenShiftin yaml-konfiguraatiossa (OpenShift Documentation)

DeploymentConfig on OpenShiftin oma tapa hoitaa sovelluksen päivitystä. ReplicaSetin sijaan käytetään ReplicationControlleria, joka käynnistää ja terminoi podeja deploymentconfig.yaml:in määrittelyjen mukaan. (Docs SCS 2020.) DeploymentConfig tarjoaa myös erilaisia liipaisimia (trigger) deploymentin automaattiseen luontiin ja strategian deploymentien väliseen siirtymiseen (Red Hat OpenShift Documentation b).



Example `DeploymentConfig` definition

```yaml
apiVersion: v1
kind: DeploymentConfig
metadata:
  name: frontend
spec:
  replicas: 5
  selector:
    name: frontend
  template: { ... }
  triggers:
  - type: ConfigChange (1)
  - imageChangeParams:
      automatic: true
      containerNames:
      - helloworld
      from:
        kind: ImageStreamTag
        name: hello-openshift:latest
    type: ImageChange (2)
  strategy:
    type: Rolling (3)
```

(1) A config change trigger causes a new deployment to be created any time the replication controller template changes.

(2) An image change trigger causes a new deployment to be created each time a new version of the backing image is available in the named image stream.

(3) The default `Rolling` strategy makes a downtime-free transition between deployments.

Kuva 5. Esimerkki DeploymentConfig yaml-konfiguraatiosta (OpenShift Documentation)

Deploymentilla ja DeploymentConfigillä on muutamia eroja, joita kannattaa miettiä valitessaan kumpaa haluaa käyttää sovelluksen hallintaan. DeploymentConfig suosii johdonmukaisuutta, kun taas Deployment valitsee ennemmin saavutettavuuden. DeploymentConfigin osalta tämä tarkoittaa sitä, että jos käyttöönottoa suorittava node kaatuu, sitä ei korvata vaan odotetaan, että se joko tulee takaisin käytettäväksi tai poistetaan manuaalisesti. Deploymentissa taas käyttöönottoa hallitsee controller manager, joka tekee algoritmiensa mukaan valintoja, jotka suosivat saavutettavuutta sen sijaan, että jouduttaisiin odottamaan tiettyä nodea. (Red Hat OpenShift Documentation b.)

Aiemmin mainittujen liipaisimien lisäksi, DeploymentConfig spesifisiä ominaisuuksia ovat automaattinen vanhan version palautus ja käyttäjien spesifioimat deployment-strategiat. Deployment spesifisiä ominaisuuksia ovat rolloverit, suhteutettu skaalaus ja pysäyttäminen kesken käyttöönoton. (Red Hat OpenShift Documentation b.)

4.2 ConfigMaps ja Secrets

Sovellukset tarvitsevat usein konfigurointia, joka on yhdistelmä asetustiedostoja, komentoriviargumentteja ja ympäristömuuttujia. OpenShiftissä konfigurointi on irrotettu levykuvista, jotta kontitetut sovellukset ovat liikuteltavissa. Tähän ratkaisuna käytetään ConfigMappeja, joiden avulla kontteihin saadaan konfiguraatiotiedot pitämällä kontit samalla riippumattomina OpenShift-alustasta. (Red Hat OpenShift Documentation c.)

ConfigMapissa konfiguraatiodataa varastoidaan avain-arvo pareissa, joko podien tai muiden järjestelmän osien kuten esimerkiksi controllerien käytettäväksi. ConfigMap on hyvin samanlainen Secretin kanssa, mutta ConfigMappia ei ole suunniteltu sisältämään arkaluontoista tietoa. (Red Hat OpenShift Documentation c.)

Kuva 6. Esimerkki ConfigMapin sisällöstä (Red Hat OpenShift Documentation)

Secretit on luotu varta vasten arkaluontoisen datan, kuten esimerkiksi salasanojen, varastoimiseen. Secretejä voidaan käyttää podin ympäristömuuttujissa tai antaa ne podille tiedostomuodossa. (Red Hat OpenShift Documentation d.)

Secretiä luodessa voidaan antaa datalle tyyppi (type), joka määrittelee avain-arvo parien rakennetta. Jos datalle ei haluta rakenne validointia, on hyvä valita tyypiksi opaque, joka on myös oletusarvo. Avaimiin liittyvät arvot voidaan antaa joko suoraan base64-koodattuina tai käyttää ”stringDataa”, jolloin arvot voi antaa selkokielisinä, mutta OpenShift muuntaa ne base64-koodatuiksi. (Red Hat OpenShift Documentation d.)

4.3 Sovelluksen tietoliikenne (Networking)

OpenShift käyttää Kubernetestä takaamaan sen, että podit pystyvät kommunikoimaan toistensa kanssa. Kubernetes antaa jokaiselle podille sisäverkon IP-osoitteen. Tällä tavoin kaikki kontit podin sisällä näyttäytyvät ulospäin kuin ne olisivat samalla laitteella. IP-osoitteiden antaminen podeille tarkoittaa myös sitä, että niillä voidaan käyttää portteja, verkkoa, palveluiden löytämistä (service discovery), kuorman tasaamista ja sovelluksen konfiguroimista samoin kuin fyysisellä palvelimella tai virtuaalikoneella. (Red Hat OpenShift Documentation e.)

Service on tapa, jolla Kuberneteksessä (ja OpenShiftissä) määritellään yhteenkuuluvat podit sekä toimintaperiaate, joilla niihin päästään käsiksi. Servicessä määritellään muun muassa selector, joka määrittelee mitä podeja service koskee. Siinä määritellään myös portit sekä tietoliikenneprotokolla. Servicen voi luoda myös ilman selectoria. Tällöin voidaan servicelle antaa esimerkiksi toisen klusterin IP-osoite, jonne liikenne ohjataan. (Kubernetes GitHub.)

Esimerkki yaml servicen luonnista:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: example
  namespace: example-namespace
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

OpenShiftillä on myös oma sisäänrakennettu DNS. Tämä ominaisuus on otettu käyttöön, koska käyttäjillä saattaa olla pyörimässä OpenShiftillä monia palveluita, esimerkiksi frontend ja backend -palvelut, joilla on eri podeja. Jotta frontend-podit saavat

kommunikoitua backend-podien kanssa klusterin sisäisesti, tarvitaan ympäristömuuttujia, kuten esimerkiksi käyttäjänimiä ja palveluiden IP:t. Jos service poistetaan ja sen tilalle luodaan uusi, servicelle luodaan uusi IP. Tällöin frontend-podit pitäisi luoda uudestaan, jotta uusi IP saataisiin käyttöön. Myös backend-service pitäisi luoda ennen frontendiä, jotta se voidaan antaa frontendille ympäristömuuttujissa. Tätä varten OpenShift käyttää DNS:ää, jotta serviceen saadaan yhteys myös nimen kautta, eikä vain IP:n. (Red Hat OpenShift Documentation e.)

Servicen lisäksi OpenShiftissä on reittejä. Reiteillä julkaistaan service antamalla sille isäntänimi, jolla päästään sovellukseen julkisesta verkosta. Reitti koostuu reitin nimestä, service selectorista ja valinnaisena valittavasta turvallisuuteen liittyvästä konfiguroinnista. (Red Hat OpenShift Documentation f.)

4.4 Storage

Pysyväistallennuslaite on mikä tahansa dataa varastoiva laite, joka säilyttää datansa laitteen sulkemisen jälkeen. Konttiympäristössä pysyväistallennus viittaa volumeihin, jotka ovat saatavilla myös konttien terminoituessa. Pysyväistallennuksen vastakohtana voidaan pitää lyhytaikaistallennus (ephemeral storage) -volumeja, jotka ovat toiminnassa vain kun konttikin on toiminnassa. (NetApp.)

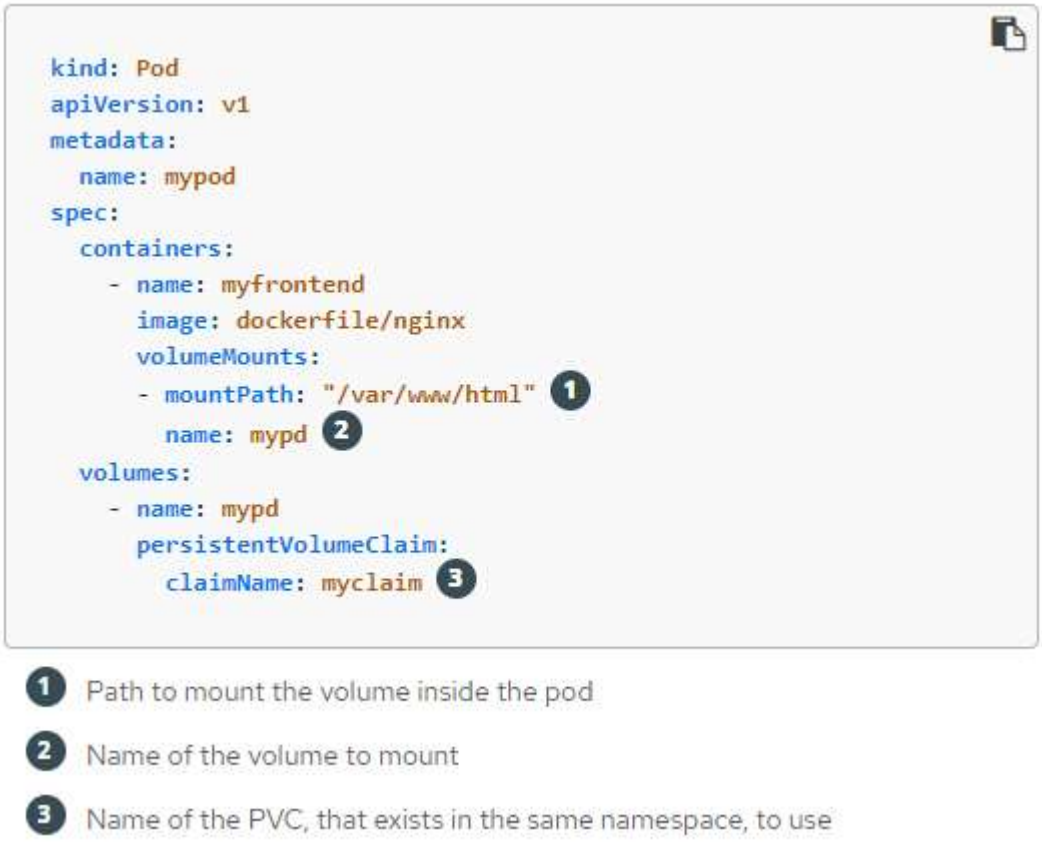
PersistentVolumet (PV) ovat siis resursseja klusterilla ja niiden elinkaari ei riipu kyseistä volumea käyttävän podin elinkaaresta. PersistentVolume on käytännössä ”fyysinen” volume palvelimella, joka varastoi dataa. PersistentVolumeClaimeja (PVC) käytetään tekemään pyyntö luoda PersistentVolume ja kiinnittämään se podeihin. Volume voidaan luoda joko staattisesti tai dynaamisesti ja sille voidaan antaa eri ominaisuuksia liittyen esimerkiksi suorituskykyyn, kokoon ja access modeihin. (NetApp.)

PersistentVolumeClaimia tehdessä määritellään volumen haluttu koko, tarvittava access mode sekä luodaan storage class, jolla kuvaillaan ja luokitellaan storagea. Access mode kuvailee volumen kyvykkyyksiä ja sen arvona voi olla joko ReadWriteOnce (RWO), ReadOnlyMany (ROX) tai ReadWriteMany (RWX). Jos access mode on RWO, volumea voi lukea ja kirjoittaa vain yksi node. ROX access modessa olevaa volumea voi lukea monet nodet. RWX volumea voi kirjoittaa ja lukea monet nodet. (Red Hat OpenShift Documentation g.)

Volumeille on mahdollista käyttää eri plugineita, kuten esimerkiksi NFS, AWS EBS, Azure File, Azure Disk ja Red Hat OpenShift Container Storage. Näillä plugineilla on tarjolla eri access modeja. Esimerkiksi Azure Disk tarjoaa vain RWO access moden, joten jos

käytössä on kyseisen kaltainen volume, ei voi käyttää muita access modeja. NFS puolestaan tarjoaa kaikkia access modeja, mutta RXO:ta käytettäessä claim pitää merkata read-only:ksi. (Red Hat OpenShift Documentation g.)

PersistentVolumeClaimin pitää olla samassa nimiavaruudessa (namespace) kuin podi, joka käyttää volumea. Tällöin klusteri löytää claimin ja käyttää sitä löytääkseen claimia vastaavan PersistentVolumen. Volume liitetään palvelimelle ja podin sisälle esimerkiksi kuvan seitsemän esittämällä tavalla. (Red Hat OpenShift Documentation g.)



Kuva 7. Volumen liittäminen podille (Red Hat OpenShift Documentation)

4.5 Levykuvien rakentaminen

Rakentaminen on prosessi, jossa input parametrit muutetaan objektiksi. Useimmiten tätä prosessia käytetään muuttamaan lähdekoodi ajettavaksi levykuvaksi. BuildConfig määrittelee kuinka rakentaminen tapahtuu. OpenShift käyttää Kubernetestä luomaan kontteja ja viemään nämä kontit levykuvarekisteriin. (Red Hat OpenShift Documentation h.)

BuildConfig siis määrittelee kuinka rakentaminen tehdään sekä mikä käynnistää uuden rakennuksen tekemisen. Riippuen sovelluksen luontitavasta OpenShift yleensä generoi BuildConfigin automaattisesti. BuildConfigissa määritellään useita eri asioita kuten

runPolicy, liipaisimet, rakennuksen lähde, strategia ja output. (Red Hat OpenShift Documentation h.)

RunPolicyllä hallinnoidaan voidaanko samasta rakennuksen konfiguraatiosta luotuja rakennuksia ajaa samanaikaisesti. Liipaisin kertoo, mikä aiheuttaa uuden rakennuksen luomisen. Lähde määrittelee rakennuksen lähteen eli onko rakennuksen tyyppi esimerkiksi Git, Dockerfile vai Binääri sekä esimerkiksi Git:in tapauksessa koodin säilytyspaikan. Rakentamisen strategiana voi olla Source, Docker tai Custom. Esimerkiksi Docker-strategiaa käytettäessä OpenShift kutsuu docker build -käskyä ja odottaa saavansa repositorion, josta löytyy sekä Dockerfile että kaikki tarpeelliset artifaktit ajettavan levykuvan luomiseen. Output määrittelee mihin konttirekisteriin onnistuneen rakentamisen tuloksena syntynyt levykuva viedään. (Red Hat OpenShift Documentation h.)

4.6 Operaattorit

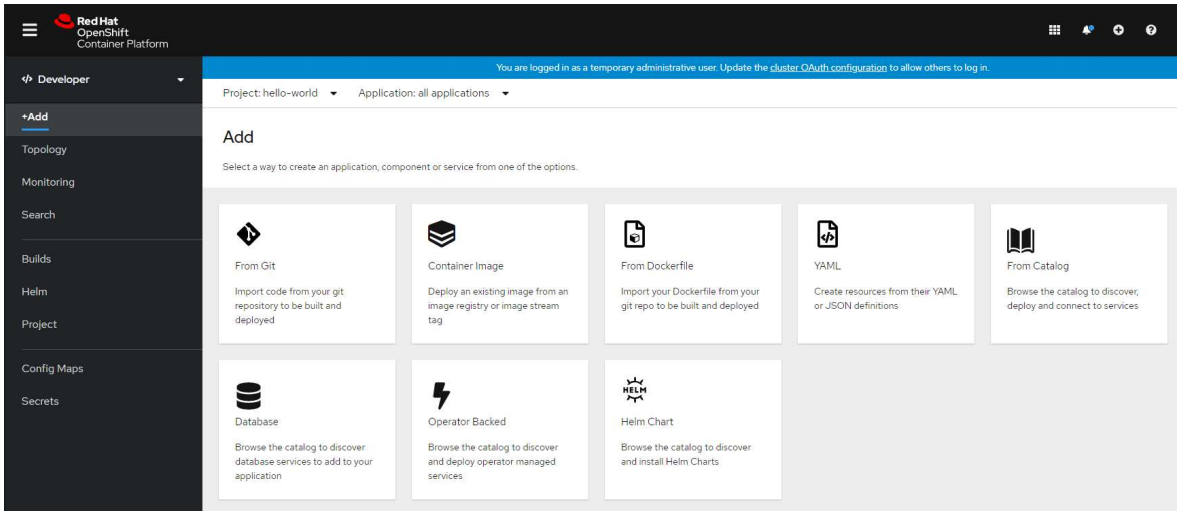
Operaattorit ovat tapa paketoida, ottaa käyttöön ja hallita Kubernetes-APIa ja kubectl-työkaluja käyttäviä sovelluksia. Operaattorit ovat toisin sanottuna ohjelmia, jotka helpottavat toisen ohjelman ajamiseen liittyvää operationaalista monimutkaisuutta. Ne tarkkailevat OpenShift-ympäristöä ja tekevät päätöksiä ympäristön nykyisen statuksen perusteella. (Red Hat OpenShift Documentation i.)

Operaattoreita voi asentaa klusterin järjestelmänvalvoja tai käyttäjä, jolle järjestelmänvalvoja on antanut oikeudet asentaa operaattoreita. Kuitenkin vain järjestelmänvalvoja voi asentaa operaattoreita koko klusterin laajuisesti niin, että operaattori tulee käyttöön kaikille klusterin projekteille. Käyttäjä pystyy asentamaan operaattoreita vain projektikohtaisesti. (Red Hat OpenShift Documentation j.)

Operaattorit löytyvät OperatorHubista, joka on OpenShift käyttöliittymässä (web console) navigaatiossa Operators-välilehden alla. OperatorHub toimii yhteistyössä Operator Lifecycle Managerin kanssa (OLM), joka asentaa ja hallinnoi operaattoreita klusterilla. (Red Hat OpenShift Documentation j.) Operaattori voi olla Red Hatin, sertifioidun sovellusmyyjän, yhteisön tekemä tai sellaisen voi tehdä myös itse (Red Hat OpenShift Documentation k).

5 Sovelluksien luominen

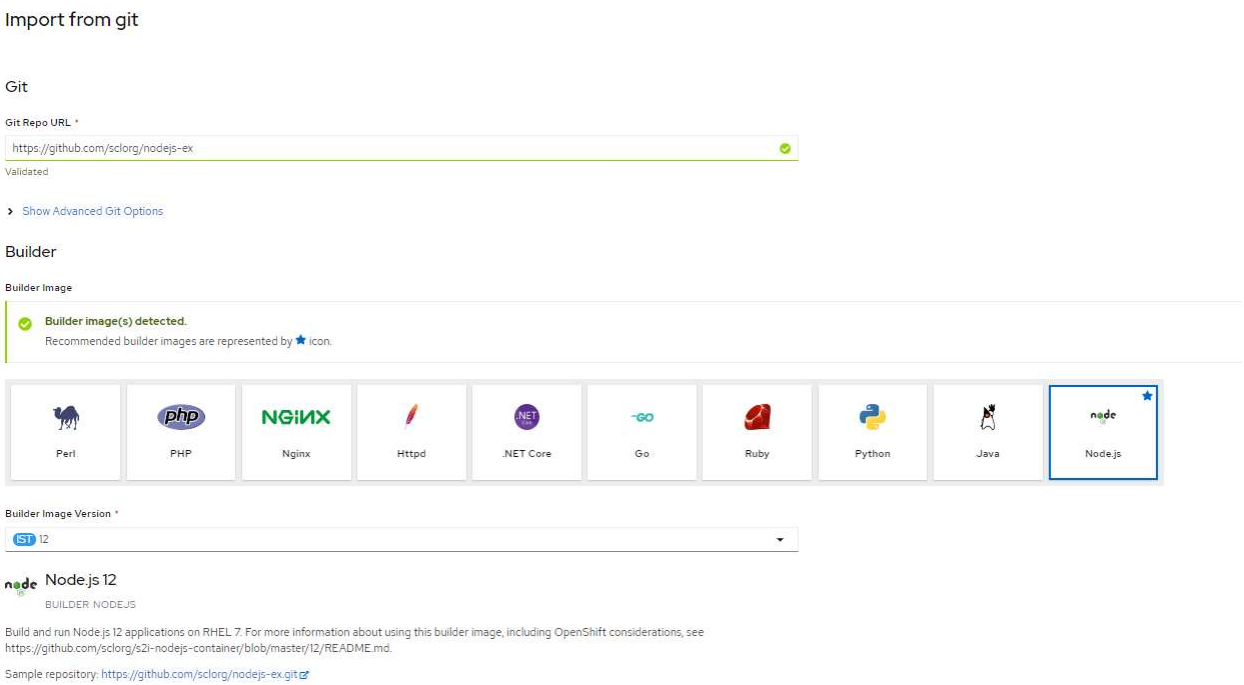
OpenShift tarjoaa monta tapaa luoda uusi sovellus klusterille. Käyttöliittymässä on näkymä (kuva 8), josta voi valita haluamansa tavan lisätä sovellus. Näitä ovat muun muassa: lisääminen Gitistä, konttilevykuvan käyttö, Dockerfilen käyttö, YAML, katalogi, tietokanta, operaattorin avulla asennettavat palvelut ja Helm Chart. (Red Hat Customer Portal.) Näistä käydään tässä läpi viisi ensimmäistä.



Kuva 8. Sovelluksen lisääminen OpenShiftin käyttöliittymästä

5.1 Lisääminen Gitistä ja Source-to-Image

Gitissä olevan koodipohjan voi tuoda OpenShiftille, jotta voi luoda, rakentaa (build) ja ottaa käyttöön (deploy) sovelluksen. Käytännössä OpenShiftille annetaan Git-repositorion URL, jonka OpenShift validoi ja havaitsee automaattisesti sovellukselle sopivan rakentajan (builderin) (kuva 9). Sovellukselle annetaan myös nimi, jota käytetään kyseiseen sovellukseen liittyvien resurssien tunnistamiseen sekä ryhmänimi eri komponenttien yhdistämiseen kokonaisuudeksi. (Red Hat Customer Portal.)



Kuva 9. Git-repositorion validointi ja rakentajan valinta

Seuraavaksi täytyy tehdä valinta resurssien tyypistä Deploymentin tai DeploymentConfigin välillä. Molemmat näistä tarjoavat hieman erilaisen tavan hallita sovelluksia. Kumpikin näistä kuitenkin kuvaavat tarvittavia asetuksia podin käyttöönotolle. (Red Hat OpenShift Documentation b.)

Oletuksena lomakkeella on myös valittuna reitin (route) luonti, joka siis mahdollistaa sovellukseen pääsyn julkisesti URL:lla selaimen kautta. Halutessaan voi myös räätälöidä sovellusta vielä enemmän tekemällä esimerkiksi reititykselle, skaalaukselle tai resursseille haluttuja valintoja. Lopuksi, kun painetaan ”Create”, näkyy rakennuksen tila esimerkiksi ”Topology” näkymässä. (Red Hat Customer Portal.)

Edellä mainittu tapa käyttää suoraan Gitissä olevaa lähdekoodia sovelluksen luomiseen ilman Dockerfilea kutsutaan Source-to-Image (S21) rakennukseksi. Lähdekoodi käytännössä injektoidaan konttilevykuvaan ja tämä kontti valmistelee lähdekoodin suoritettavaksi. Valmistelu voi tapahtua esimerkiksi asentamalla riippuvuudet ja siirtämällä lähdekoodin oikeaan kansioon. (Source-to-image.)

5.2 Konttilevykuva

Sovellus voidaan myös ottaa käyttöön käyttämällä valmista konttilevykuvaa. Tätä varten levykuvan täytyy olla hostattuna joko ulkoisessa levykuvarekisterissä (image registry),

esim. Docker Hub, tai OpenShiftin sisäisessä levykuvarekisterissä (OpenShift Cookbook b). Levykuvarekisterillä tarkoitetaan palvelinta, johon varastoidaan ja josta ladataan konttilevykuvia (Red Hat OpenShift Documentation l).

Konttilevykuva voidaan ottaa käyttöön käyttämällä joko levykuvarekisteriä tai image streamiä (Red Hat Customer Portal). Image stream on kokoelma konttilevykuviin liittyviä versioita, jotka ovat tunnistettavissa tageilla. Image stream ei siis sisällä levykuvaa itsessään vaan viitteet toisiinsa liittyviin konttilevykuviin. Levykuvat ovat edelleen tallennettuna levykuvarekisteriin. (Red Hat OpenShift Documentation l.)

Etuja image streamin käyttämisessä levykuvarekisterin sijaan on monia. Esimerkiksi, kun levykuvasta luodaan uusi versio, voidaan se ottaa automaattisesti käyttöön. Myöskin vanhan version palautus onnistuu helposti. (Red Hat OpenShift Documentation l.)

5.3 Dockerfile

Yksi tapa luoda sovellus on käyttää Dockerfilea. Dockerfile on tekstitiedosto, joka sisältää kaikki levykuvan kokoamiseen tarvittavat käskyt. Dockerfile alkaa FROM-käskyllä, joka asettaa alkulevykuvan (base imagen) tai pohjalevykuvan (parent imagen) seuraavia käskyjä varten. (Docker Docs.) Alkulevykuvan ja pohjalevykuvan ero on, että alkulevykuvalla ei ole pohjalevykuvaa, johon se perustaa rakennuksen. Pohja- tai alkulevykuvaa käytetään siis pohjana lopulle tiedostolle ja se tarjoaa olennaisia rakennuspalikoita kontitetulle ympäristölle (Kisller 29.6.2020).

FROM-käskyn lisäksi yleisesti käytettyjä käskyjä ovat mm. RUN, COPY ja ADD. Taulukosta 1 näkyy yleisimpiä käskyjä ja niiden tarkoitus.

Taulukko 1. Yleisimpiä Dockerfilessa käytettyjä käskyjä (Kisller 2020)

Command	Purpose
FROM	To specify the parent image.
WORKDIR	To set the working directory for any commands that follow in the Dockerfile.
RUN	To install any applications and packages required for your container.
COPY	To copy over files or directories from a specific location.

ADD	As COPY, but also able to handle remote URLs and unpack compressed files.
ENTRYPOINT	Command that will always be executed when the container starts. If not specified, the default is /bin/sh -c
CMD	Arguments passed to the entrypoint. If ENTRYPOINT is not set (defaults to /bin/sh -c), the CMD will be the commands the container executes.
EXPOSE	To define which port through which to access your container application.
LABEL	To add metadata to the image.

OpenShiftillä on myös ominaisuuksia, joiden takia esimerkiksi Dockerilla toimiva kontti ei toimi OpenShiftissä. Tästä esimerkkinä on se, että OpenShiftilla pyörivässä kontissa käyttäjänä on tietyltä numeroväliltä arvottu numerosarja. Eri projekteissa käytetään eri numeroväleiltä arvottuja käyttäjiä. Tämän tarkoituksena on varmistaa, että jos yhdessä sovelluksessa on tietoturva-aukko ja sovellus pääsisi kontista palvelimelle, sovellus ei kuitenkaan pysty vaikuttamaan muihin sovelluksiin. (OpenShift Cookbook c.)

Kontteja ei siis ajeta root-käyttäjänä, joten Dockerfilessa täytyy erikseen antaa oikeudet tarvittaviin tiedostoihin ja kansioihin root-ryhmään kuuluville käyttäjille, johon myös OpenShiftin arvotut käyttäjät kuuluvat. Kontissa ei myöskään voida käyttää portteja alle 1024, koska nämä ovat etuoikeutettuja (privileged) portteja, joiden käyttämiseen tarvitaan etuoikeutettu käyttäjä, jollainen OpenShiftin arpoma käyttäjä ei ole. (Red Hat OpenShift Documentation m.)

5.4 YAML ja Katalogi

Sovelluksia voi myös luoda ja muokata suoraan YAML-tiedostoista. Toinen helppo tapa on käyttää Developer-katalogia, josta löytyy valmiina levykuvia sovelluksille, palveluille ja source-to-image rakentajille. Myös esimerkiksi tietokannan voi helposti asentaa katalogin kautta. (Red Hat Customer Portal.)

5.5 Topologianäkymä

Kun sovellus on luotu, OpenShift ohjaa käyttäjän automaattisesti topologianäkymään. Topologia tarjoaa visuaalisen näkymän kaikista projektin sovelluksista, niiden

rakentamisen statuksesta ja näihin liittyvistä komponenteista ja palveluista. Podin statusta kuvataan eri värein ja sanoin. Mahdollisia eri statuksia ovat: Running, Not Ready, Warning, Failed, Pending, Succeeded, Terminating ja Unknown. Sovellusta luotaessa, kun levykuva on otettu käyttöön, status on pending. Jos sovellus on rakennettu ilman ongelmia, tilaksi muuttuu running. (Red Hat OpenShift Documentation n.)

Kuva 10. Running-tilassa oleva sovellus Topologianäkymässä (OpenShift Documentation)

Topologianäkymässä voi myös klikata komponenttia, jolloin oikealla tulee näkyviin paneeli, josta näkee tietoja kyseisen komponentin podeista, rakentamisen statuksesta, Serviceistä ja reiteistä. Paneelista pystyy myös muuttamaan podien määrää ylös ja alas nuolilla sekä pääsee katselemaan lokeja. (Red Hat OpenShift Documentation n.)

6 Tutkimuskysymykset ja menetelmät

Tutkimuskysymykseni on, miten sovellus asennetaan OpenShiftille? Tapoja on toki monia, mutta käyn kokonaisuudessaan läpi yhden tavan asentaa toimiva sovellus. Tutkin mitä sovelluksen asentamiseen ja toimimaan saamiseen tarvitaan käytettäessä OpenShiftin käyttöliittymää. Tarkastelen myös mitä mahdollisia ongelmia asennuksen yhteydessä tulee ja etsin näihin ratkaisuja. Tutkimukseni tavoitteena on tarjota tietoa OpenShiftin käytöstä ihmisille, jotka eivät ole kyseistä alustaa käyttäneet joko ollenkaan tai ovat sen käytössä aloittelijoita.

Asennettavaksi sovellukseksi valitsin Drupal 9.1:n, sillä se on suosittu alusta isompien yritysten verkkosivuille. Esimerkiksi Cisco, Tesla ja Australian hallitus käyttävät Drupalia nettisivujensa ylläpitoon (Created by Cocoon 2019). Drupal vaatii myös tietokannan asentamisen toimiakseen, mikä on hyvin realistinen vaatimus monille sovelluksille. Tutkimus lähtee myös olettamuksesta, että asentajalla on käytössään toimiva OpenShift-klusteri.

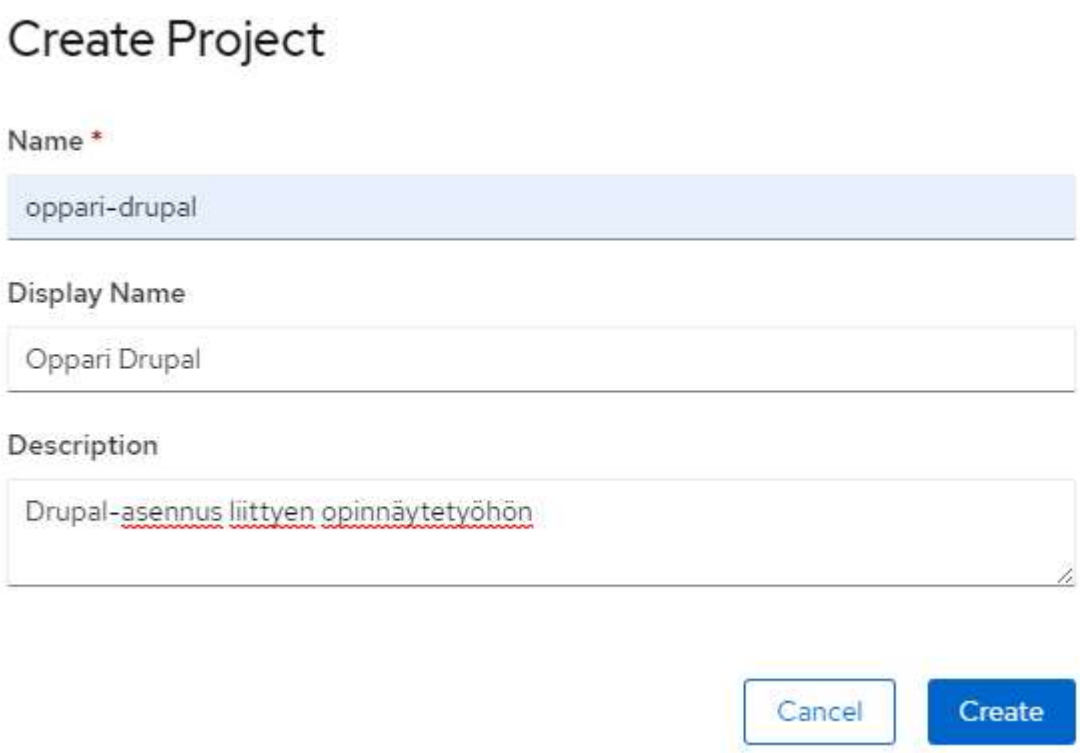
Tutkimus itsessään on kvalitatiivinen tutkimus, sillä tutkimuksessa pyritään tutkimaan yhden sovelluksen asennusta käytännössä OpenShift-alustalle. Aineistoa ei myöskään analysoida lukuina vaan pyritään ymmärtämään käytännön toimintaa.

7 Empiria

Asennettavaksi sovellukseksi valitsin Drupalin, joka on sisällönhallintajärjestelmä ja tarjoaa hyvän alustan nettisivujen kehittämiseen. Drupalille löytyy valmiita kontteja Docker Hubista, joita muokkaamalla voi saada toimivan OpenShift-sovelluksen. Tässä tutkimuksessa käytetään valmista Drupal 9.1 -konttilevykuvaa, jonka pohjana on Buster eli Debian 10-käyttöjärjestelmä. Konttilevykuva sisältää myös PHP:n ja Apache HTTPD-palvelimen (myöhemmin pelkkä Apache). Drupal-kontin lisäksi tarvitaan tietokanta, joka on tässä tapauksessa MariaDB. MariaDB-kontti voidaan asentaa helposti OpenShiftiltä löytyvästä katalogista suoraan.

7.1 Projektin luonti OpenShiftille

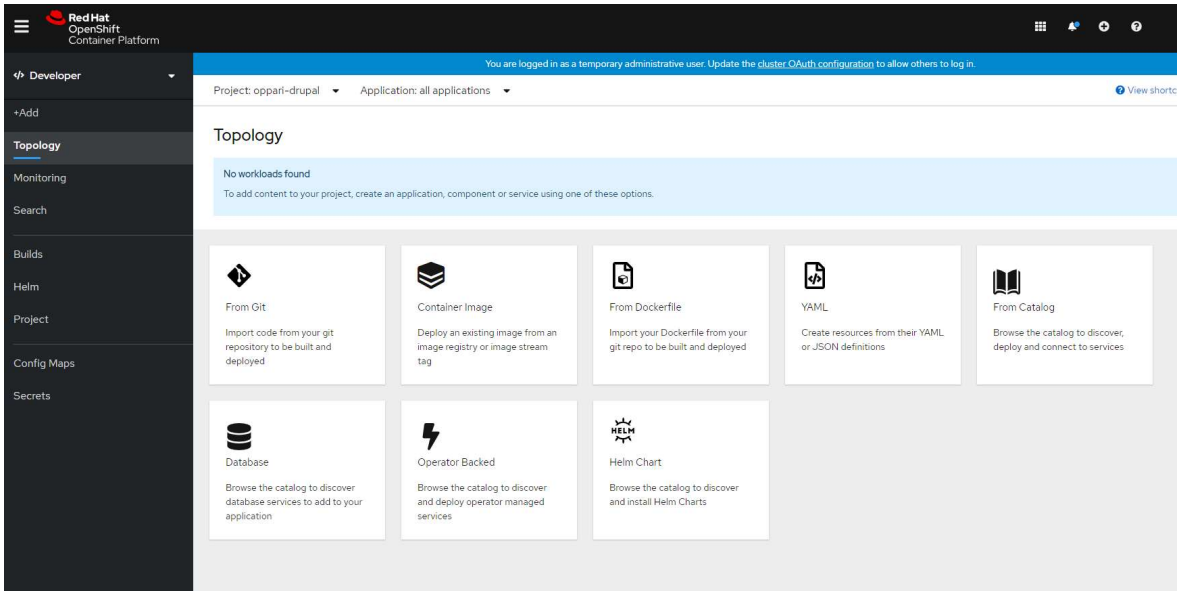
Ennen kun voidaan asentaa sovellusta OpenShiftille, pitää sovellusta varten luoda projekti. Sen voi luoda menemällä vasemmasta valikosta Home-välilehden alta Projects-välilehdelle. Oikeasta yläkulmasta löytyy ”Create Project” -painike. Avautuvalla lomakkeella (kuva 11) ainoastaan nimi on pakollinen, mutta muutkin tiedot on hyvä lisätä. Kun projektin luonti on valmis, tulee näkyviin automaattisesti projektin yleiskatsaus (Overview) -sivu.



Kuva 4. Projektin luonti

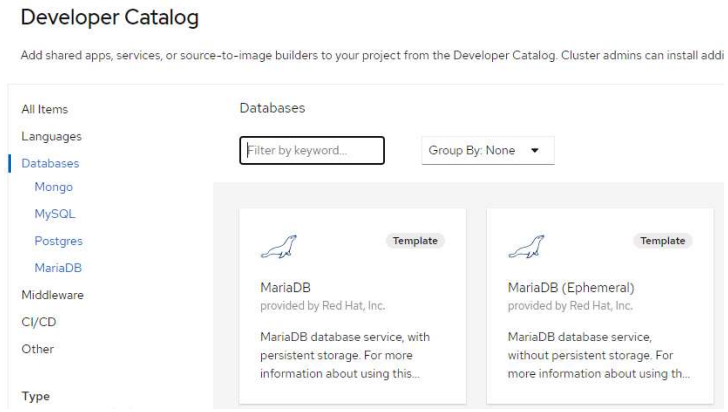
7.2 MariaDB:n asennus

Aloitetaan sovelluksen asentaminen MariaDB:llä, sillä se on helpompi asentaa, mutta sitä myös tarvitaan Drupalin toimimiseen. OpenShift-konsolissa siirrytään ensin Developer-puolelle. Koska projektiin ei ole vielä asennettu mitään, tulee vastaan suoraan näkymä, josta voi valita sovelluksen asennustavan.



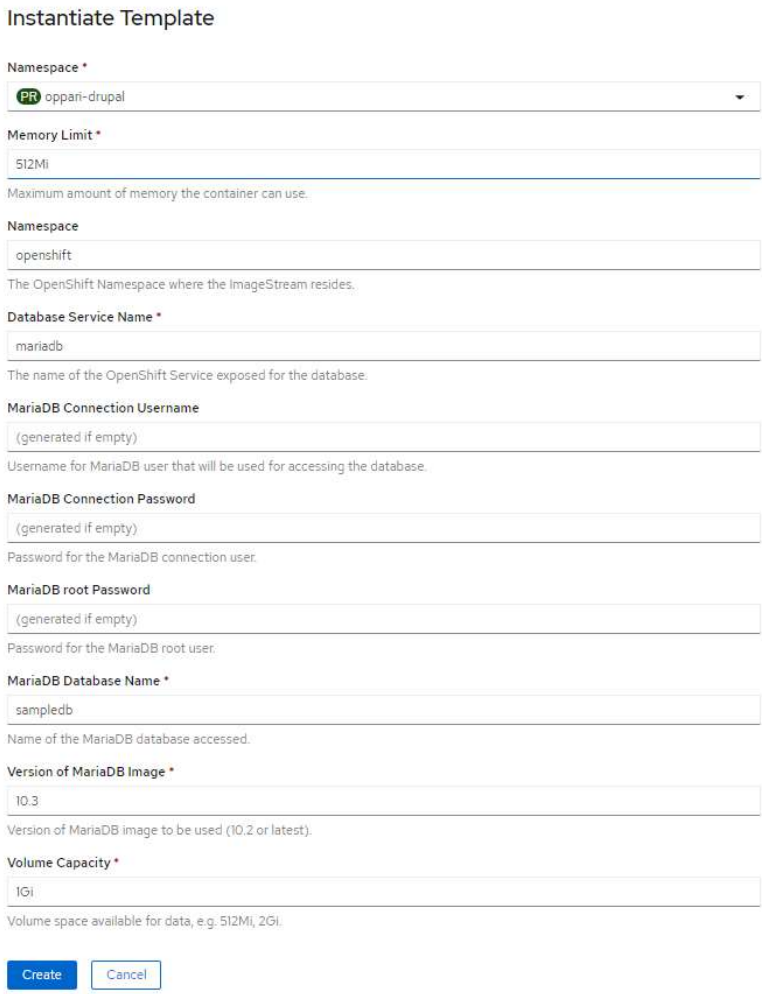
Kuva 12. Topologianäkymä ennen kuin projektiin on asennettu mitään

MariaDB:n voi hakea joko katalogista tai painaa ”Database” -vaihtoehtoa, josta pääsee suoraan katalogin tietokantavaihtoehtoihin. Tässä valitsemme ensimmäisenä tulevan MariaDB-templaten sillä se luo automaattisesti tietokannalle myös pysyväistallennuksen.



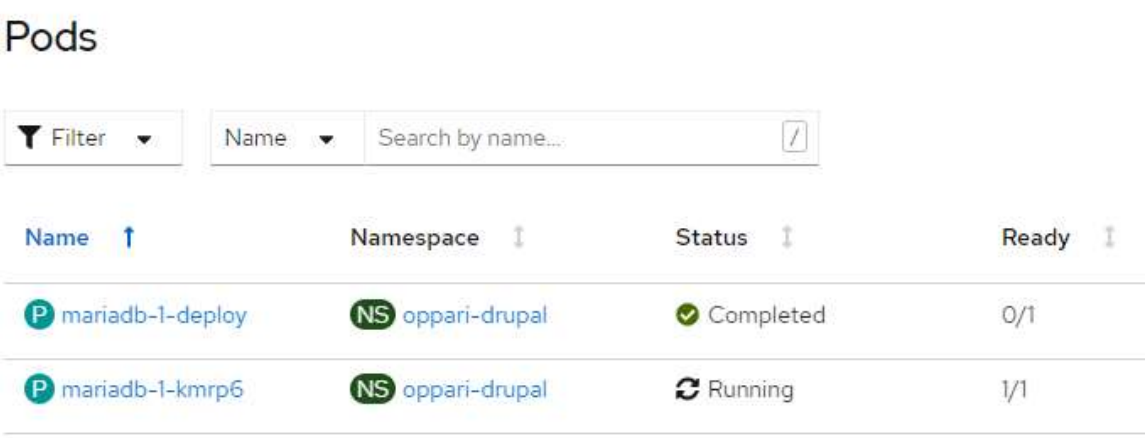
Kuva 13. MariaDB-templaatit katalogissa

Seuraavaksi pääsee valitsemaan tietokannalle asetukset kuten muistin koon, tietokannan nimen, käyttäjän, salasanan, version ja pysyväismuistin koon. Templaatti luo DeploymentConfigin, PersistentVolumeClaimin, Secretin ja Servicen, kuten lukee myös sivulla, josta templaatti luodaan. Tätä projektia varten muutetaan ainoastaan MariaDB:n versiota oletuksena tulevasta 10.2:sta 10.3:een, sillä käyttämämme Drupal 9 vaatii MariaDB:stä vähintään versiota 10.3.7+. Salasanat ja käyttäjätunnus generoidaan, jos niitä ei syötä itse.



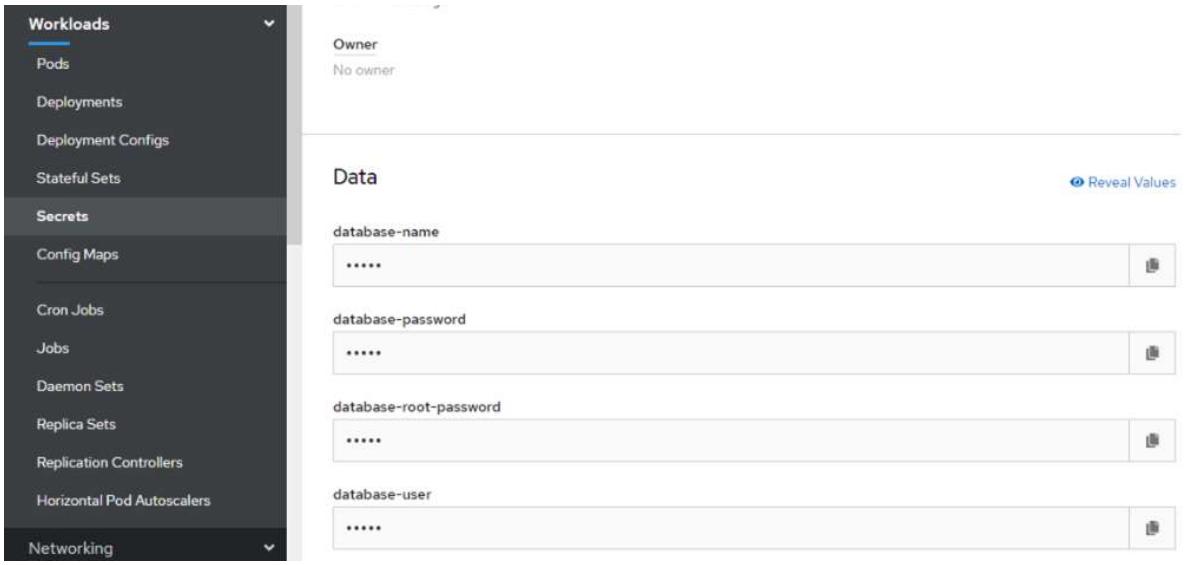
Kuva 14. MariaDB-templaatin asetusvalinnat

Painettaessa Create-painiketta siirtyy näkymä topologianäkymään, josta voi seurata, milloin tietokanta on valmis. Voi myös siirtyä Administrator-näkymään, josta valitsemalla Workloads ja Pods, näkee tietokannan asennuksen nykyisen vaiheen. Alla olevassa kuvassa (kuva 15) MariaDB näkyy olevan Running-tilassa, joten voimme testata tietokantaa, käymällä katsomassa secreteistä MariaDB:lle generoidun käyttäjän ja kirjautumalla sisään tietokantaan kontin terminaalista.



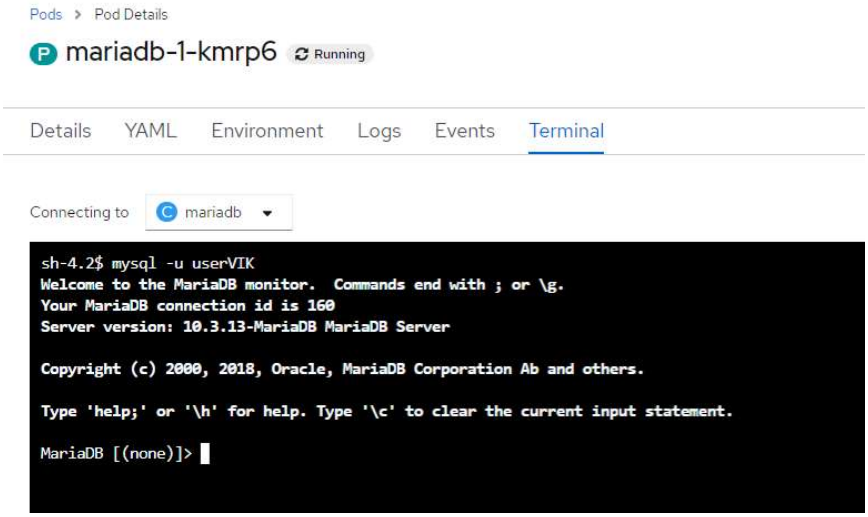
Kuva 15. MariaDB-podien tila

Secrets-näkymästä löytyy MariaDB-niminen secret, jonka sisältä löytyvät salasanat, tietokannan nimi ja käyttäjätunnus. Voi painaa ”Reveal Values”, jos haluaa nähdä näiden sisällöt tai vain kopioida esimerkiksi database-userin sisällön siihen tarkoitetusta painikkeesta.



Kuva 16. MariaDB secretit

Siirrytään uudestaan Pods-välilehdelle ja valitaan MariaDB:n käynnissä oleva kontti. Täältä voi valita Terminal-välilehden, josta pystyy kirjautumaan sisälle tietokantaan. Koska kyseessä on MariaDB:n oma kontti, josta kirjaudutaan sisään, tarvitsee tähän vain käyttäjätunnuksen. Kuten kuvasta 17 näkyy, MariaDB on toiminnassa ja siihen kirjautuminen onnistui.



Kuva 17. Toimiva MariaDB

7.3 Drupalin asennus

GitHubista löytyy Drupalin eri versioille valimiita Dockerfilejä, joita voi käyttää sovelluksen asentamiseen. Nämä tiedostot on tehty Dockeria varten, joten tarvitaan pieniä muutoksia, jotta kyseisiä Dockefilejä voidaan käyttää OpenShiftissä. Kyseisestä GitHub-repositoriosta voi onneksi tehdä forkin ja tehdä muutokset forkattuun repositorioon.

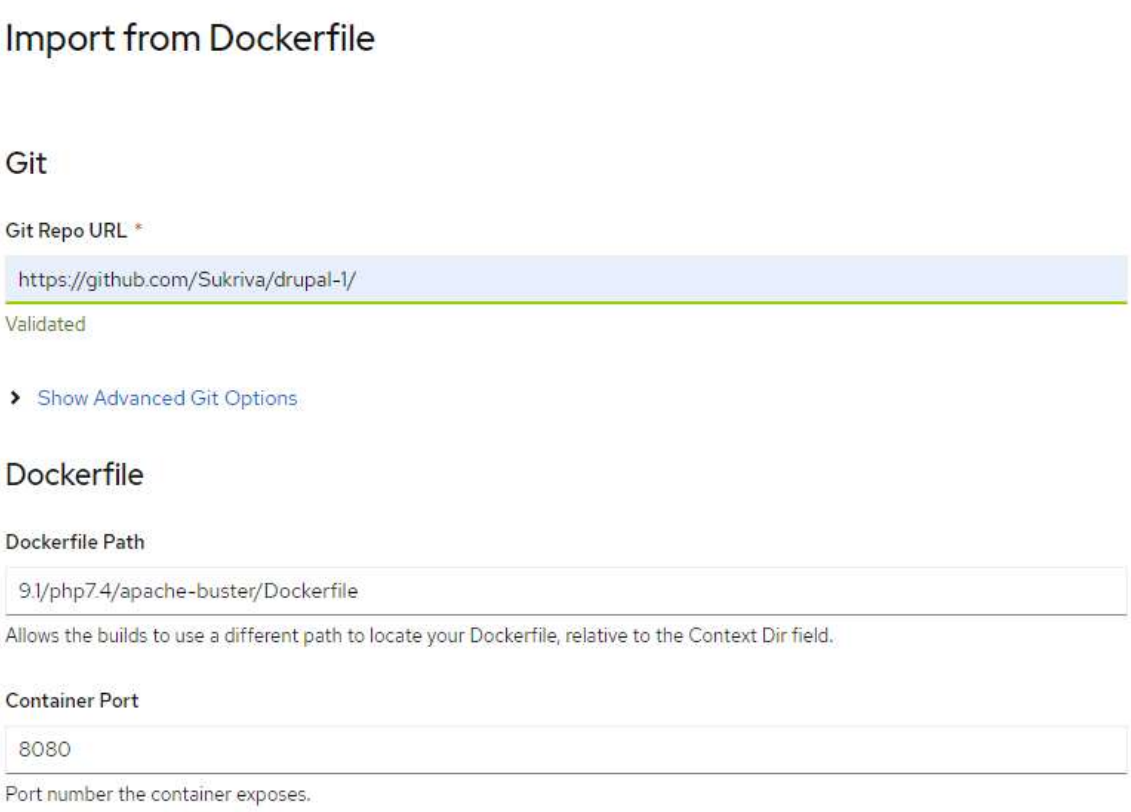
Tässä käytetty Drupal käyttää Apachea ja Apache kuuntelee oletuksena porttia 80. OpenShift ei kuitenkaan salli portteja alle 1024:n, joten Dockerfileen pitää lisätä kohta, joka muuttaa Apachen kuunteleman portin 80:stä esimerkiksi porttiin 8080. Tämän voi tehdä esimerkiksi käyttämällä sed:iä, jolla voi muokata tiedostoja kertomalla mitä sieltä löytyy ja minkälaiseksi se pitää muokata.

Apachen porttikonfiguraatio löytyy polusta /etc/apache2/ports.conf ja tiedoston sisältä löytyy ”Listen 80”, joka halutaan muuttaa. Dockerfileen voidaan siis lisätä
RUN sed -i "s/Listen 80/Listen 8080/g" /etc/apache2/ports.conf
-käsky, joka muuttaa kuunneltavan portin. Koska kyseessä on Dockerfile, käskyn alussa pitää olla ”RUN”, jotta käsky suoritetaan. Tämän jälkeen normaali sed-käsky, jonka voisi kirjoittaa tavallisesti terminaaliin. Docker-tiedoston loppuun myös lisätään käsky ”EXPOSE 8080”, jotta Drupal-konttiin päästään käsiksi tästä portista.

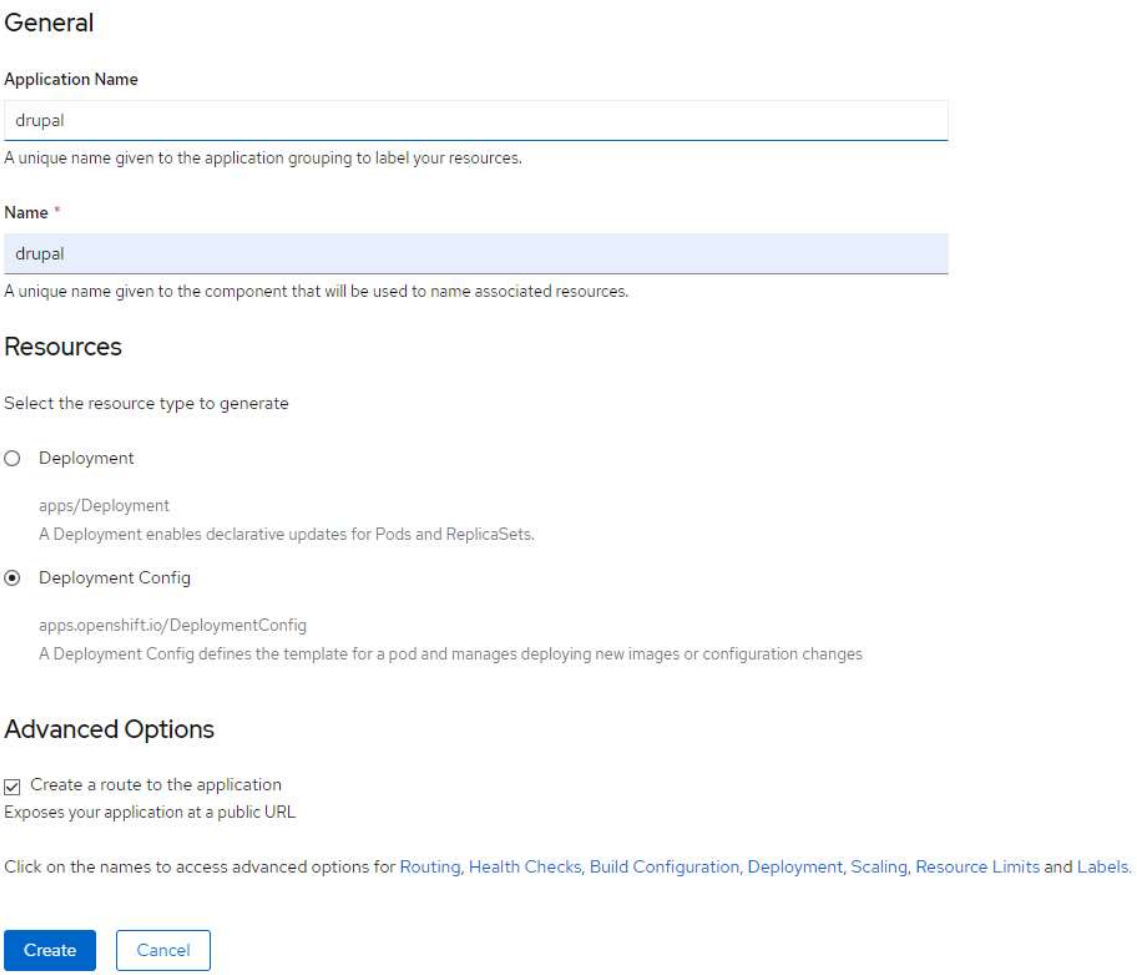
Kun Dockerfileen on tehty muutokset, voidaan asentaa sovellus OpenShiftille käyttämällä tätä tiedostoa. Voidaan mennä taas siis Developer-näkymään. Tällä kertaa, koska projektissa on jo jotain asennettuna (MariaDB), tullaan suoraan topologianäkymään. Jotta

päästään asentamaan sovellusta, valitaan vasemmalta valikosta ”+Add”. Tällä kertaa avautuvasta näkymästä ei valita ”Database” vaihtoehtoa, vaan ”From Dockerfile”.

Dockerfilen tuomiseksi annetaan GitHub-repositorion osoite, jonka OpenShift ensin validoi. Seuraavaksi annetaan polku, josta haluttu Dockerfile löytyy repositoriossa, sekä avattava portti, joka mainittiin myös Dockerfilessä.



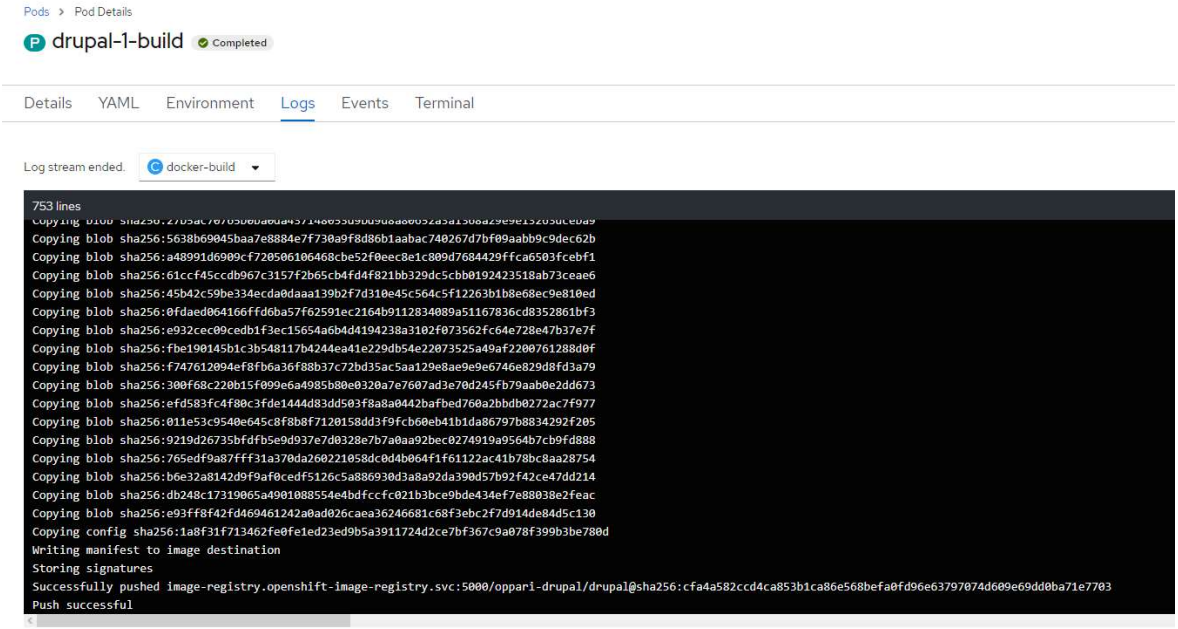
Kuva 18. Dockerfilen tuominen OpenShiftille 1/2



Kuva 19. Dockerfilen tuominen OpenShiftille 2/2

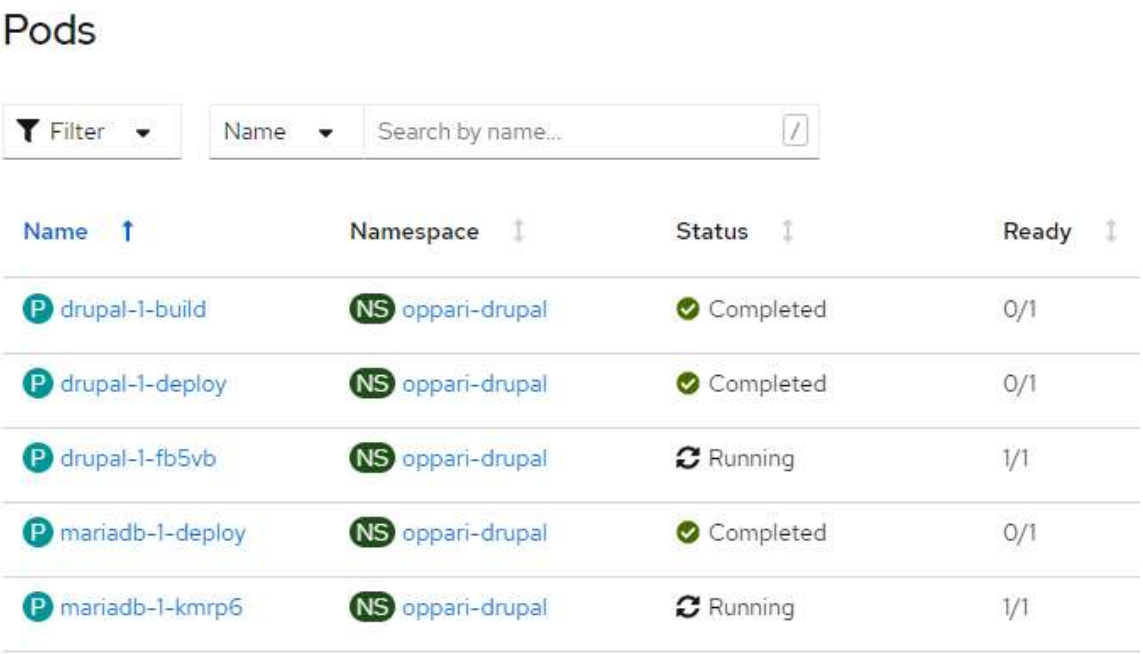
Sovellukselle annetaan nimi, sekä ryhmänimi, jolla eri komponentit liitetään kyseiseen sovellukseen. Resursseissa voi valita joko Deploymentin tai DeploymentConfigin. Tässä on valittu DeploymentConfig siltä varalta, että konfiguraatioon joudutaan tekemään vielä muutoksia. Advanced Options -kohdassa on oletuksena valittu reitin luominen, mutta sen voi halutessaan ottaa pois päältä. Tässä kuitenkin haluamme saada reitin eli URL:in, jonka kautta pääsemme Drupalin itsensä asennusikkunaan. Halutessaan sovellukselle voi tehdä vielä muitakin asetuksia, mutta tässä kohtaa meille riittää jo annetut asetukset ja voidaan painaa ”Create”.

Administrator-näkymän Workloads-välilehden alta löytyvästä Pods:ista voi taas katsoa kuinka asennus edistyy. Jos asennuksen kanssa on ongelmia, on hyödyllistä klikata podia, esimerkiksi tässä tapauksessa nimellä ”drupal-1-build”, ja valita välilehdiltä joko ”Logs” tai ”Events”. Molemmat ovat hyviä paikkoja, joista aloittaa ongelmanselvitys, jos levykuvan rakentaminen ei ole onnistunut. Kuten alta näkyy (kuva 20), kyseinen asennus kuitenkin oli onnistunut.



Kuva 20. Drupal-rakennuksen loki

Pods-näkymää katsottaessa (kuva 21) näkyy myös, että sovelluksen käyttöönotto on onnistunut ja että Drupalista on pyörimässä kontti.

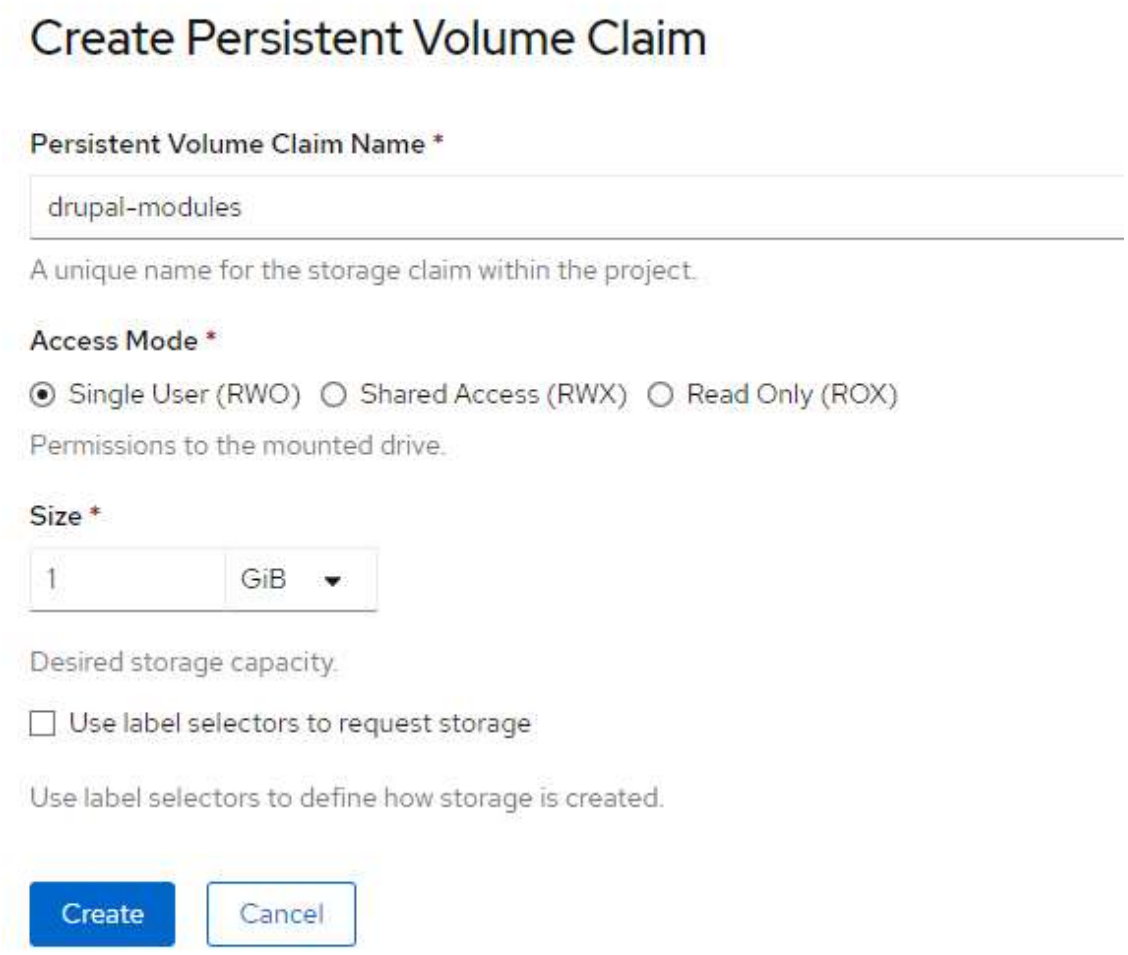


Kuva 21. Näkymä projektin podeihin

Halutessaan voi käydä katsomassa Networking-välilehden alta löytyvältä Routes-välilehdeltä Drupalia varten luodun reitin ja klikata sitä, jotta näkee, että se toimii. Tällöin pitäisi tulla näkyviin Drupalin asennusikkuna. Ennen kuin Drupalia voi lähteä asentamaan, täytyy kuitenkin Drupalille luoda OpenShiftillä pysyväismuisti, jotta tarvittavat tiedostot ja konfiguraatiot saadaan pysymään muistissa, myös podin kaatuessa.

Yleisesti pidetään hyvänä, että Drupalissa volumeina olisivat /var/www/html/modules, /var/www/html/profiles, ja /var/www/html/themes. Hieman monimutkaisempi on hakemisto /var/www/html/sites, jota tarvitaan jo Drupalia asennettaessa. (Docker Hub.) Aloitetaan kuitenkin tekemällä kolmelle ensimmäiselle volumet.

OpenShiftin sivupaneelista Storagen alta löytyy Persistent Volume Claims. Tässä vaiheessa sieltä löytyy MariaDB:n volume claim, jonka MariaDB:n templaatti loi automaattisesti. Painetaan ”Create Persistent Volume Claim”. Kuten kuvassa 22 näkyy, luodaan ensin Drupalin moduuleille volume ja annetaan tälle uniikki nimi. Seuraavaksi määritellään access mode. Tässä tapauksessa Single User (RWO) on hyvä, koska kyseisellä klusterilla on käytössä vain yksi node. Volumen kooksi riittää myös 1 GiB. Kun painetaan ”Create”, tulee esille näkymä, jossa on volume claimin tiedot. Tehdään sama vielä profilelle ja themesille.



Kuva 22. Persistent Volume Claimin luominen

Kun volume claimit on luotu, volumet pitää vielä lisätä Drupalin DeploymentConfigiin, jotta Drupalin podi osaa hyödyntää volumeja. DeploymentConfigiin lisätään siis seuraavanlainen kohta:

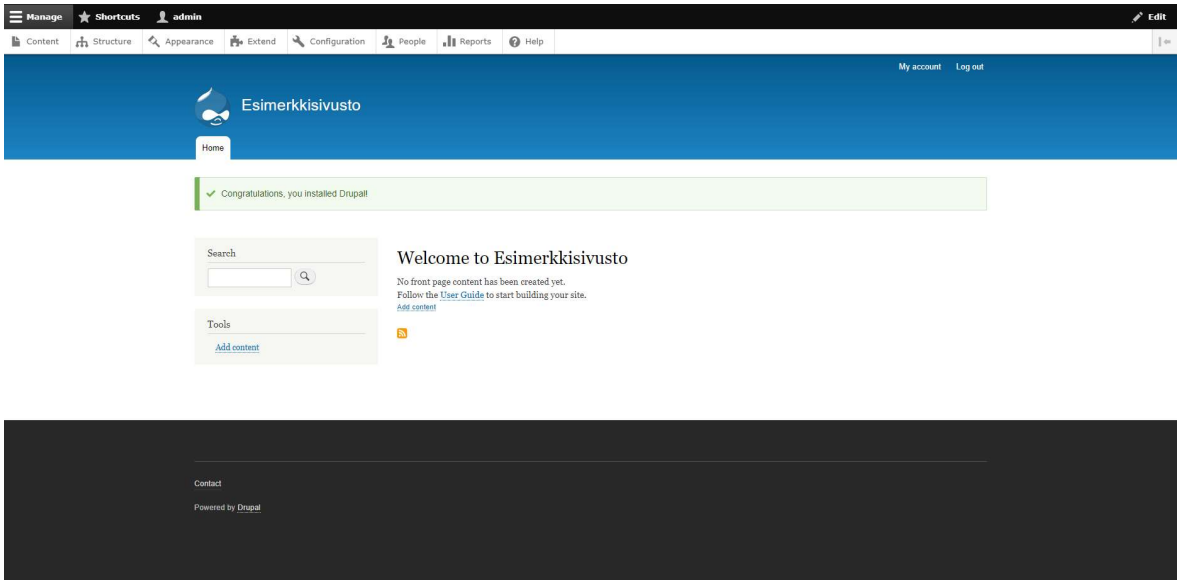
```
    volumes:
      - name: drupal-themes
        persistentVolumeClaim:
          claimName: drupal-themes
      - name: drupal-profiles
        persistentVolumeClaim:
          claimName: drupal-profiles
      - name: drupal-modules
        persistentVolumeClaim:
          claimName: drupal-modules
    containers:
      - name: drupal
        image: >-
          <linkki-rekisteriin>
        resources: {}
        volumeMounts:
          - name: drupal-themes
            mountPath: /opt/drupal/web/themes/
          - name: drupal-profiles
            mountPath: /opt/drupal/web/profiles/
          - name: drupal-modules
            mountPath: /opt/drupal/web/modules/
```

Sites-kansio on hieman erilainen, koska sites:n pitää olla volume, mutta Drupal tarvitsee sites-kansioon myös levykuvan mukana tulleet tiedostot, jotta asennus onnistuu. Volumen liittäminen kuitenkin ylikirjoittaa koko kansion ja volume on tällöin kontin käynnistyessä tyhjä. Pitää siis keksiä tapa, jolla siirtää levykuvan sites-kansiossa olevat tiedostot sites-volumeen. Tämä vaatii hieman ylimääräistä vaivaa.

Voidaan siis tehdä ensin volume nimellä ”sites” samalla tavalla kuin profiles, modules ja themes volumet tehtiin. Drupal DeploymentConfigissa kuitenkin laitetaan mountPathiin esimerkiksi sites2, ei sites. Tällöin Drupalista löytyy levykuvan mukana tullut sites, sekä tyhjä volume hakemistosta sites2. Käydään Drupalin terminaalissa kopioimassa sites-kansion sisältö volumeen sites2 käskyllä ”cp -r sites/* sites2”. Kun tiedostot ovat volumen mountPath:ssa sites2, voidaan käydä muuttamassa mountPath sites-nimelle, jolloin Drupalilla on volume sites, josta löytyy levykuvan mukana tulleet tiedostot.

Tämän jälkeen voi mennä sivupaneelista Networking-välilehden alta Routes-sivulle ja sieltä klikata sivun URL:ia, joka vie Drupalin asennussivulle. Asennus on helppo muutoin,

mutta Database-kohdassa pitää muistaa käydä katsomassa tietokannan tiedot OpenShiftiltä secreteistä, sekä advanced optioneissa pitää muuttaa host OpenShiftille asennetun MariaDB:n sisäiseksi osoitteeksi, joka on muotoa <MariaDB:n-servicen-nimi>.<projektin-nimi>.svc.cluster.local. Drupalin asennettua pääsee Drupalin pääsivulle, jota voi sitten muokata haluamakseen.



Kuva 23. Asennettu Drupal

8 Päätelmät ja johtopäätökset

OpenShiftillä on paljon eri ominaisuuksia ja asennuksia pystyy tekemään monin eri tavoin. Halutessaan voi tehdä nopeita asennuksia templaattien avulla, mutta oikeissa projekteissa asiat eivät yleensä mene näin yksinkertaisesti. Esimerkiksi Drupalin asennus vaatii jo vähän enemmän työtä, vaikka sillekin löytyy Docker Hubista valmiita levykuvia.

Käytännössä Dockerfilea kirjoitettaessa pitää osata ajatella tavallisen Docker-pohjaisen kontin luomisen lisäksi myös OpenShiftin omia sääntöjä. Jos ei tiedä esimerkiksi, että portteja alle 1024 ei voi käyttää, tai että konttia ei voida ajaa root-käyttäjänä, jolloin tarvittaviin kansioihin pitää antaa oikeudet OpenShiftin arpomalle käyttäjälle jo Dockerfilessä, voi OpenShiftin käyttö tuottaa haasteita. Toisaalta jos Kubernetes on entuudestaan tuttu käyttäjälle, voi OpenShiftin käytön sisäistäminen käydä helpostikin.

Tässä opinnäytetyössä tehty tutkimus kattaa vain yhden tavan tehdä asennus eikä asennuksessa tehty mitään ylimääräistä. Se tuo kuitenkin esille asennuksen perusteita, joihin törmää varmasti kaikissa asennuksissa oli tapa asentaa mikä tahansa. Asennuksiin voi tuoda mukaan paljon muitakin ominaisuuksia riippuen sovelluksesta ja sen tarpeista. Tämä tutkimus on pyritty pitämään melko yksinkertaisena, jotta se olisi aloittelijaystävällinen. Kuten sanottu, oikeissa projekteissa on kuitenkin paljon enemmän ominaisuuksia, ja näin tulee myös ongelmiakin matkalla vastaan enemmän.

Kirjallisuutta OpenShiftin neljännestä versiosta ei löydy paljoa. Hyvin paljon materiaalista, joka löytyy sekä kirjallisena että internetistä, on kirjoitettu OpenShiftin kolmannelle versiolle. Osa tästä materiaalista pitää edelleen paikkansa, mutta materiaalista pitää osata suodattaa oikeat asiat. OpenShiftistä löytyy hyvin dokumentaatiota sekä kolmanteen että neljänteen versioon, mutta paikoin tämä dokumentaatio keskittyy OpenShiftin käyttämisen ohjeistamiseen ja jättää selittämättä mitä jotkin keskeiset konseptit tarkoittavat.

Tällä opinnäytetyöllä oli tarkoitus vastata tarpeeseen saada suomenkielistä dokumentaatiota OpenShiftiin liittyen varsinkin aloittelijoille, jolloin myös OpenShiftin omasta dokumentaatiosta puuttuvia konsepteja oli tarpeen selittää. Myöskin aloittelija pystyy opinnäytetyön luettuaan asentamaan yksinkertaisen sovelluksen OpenShiftille tai vähintäänkin samanlaisen sovelluksen, joka on tässä opinnäytetyössä toteutettu. Aloittelijan tarpeeseen tämä työ varmasti vastaa hyvin.

Lähteet

Abushamleh, M. 19.8.2020. OpenShift 101: Introduction, architecture, and operators. IBM Developer Blog. Luettavissa: https://developer.ibm.com/blogs/openshift-101-architecture/. Luettu: 22.11.2020.

Created by Cocoon. Top 10 Uses for Drupal in 2019. Luettavissa: https://createdbycocoon.com/post/top-uses-for-drupal. Luettu: 26.4.2021.

Dharmalingam, N. 11.10.2019. What are the New Features of OpenShift 4? Whizlabs Blog. Luettavissa: https://www.whizlabs.com/blog/openshift-4-new-features/. Luettu: 22.11.2020.

Docker docs. Dockerfile reference. Luettavissa: https://docs.docker.com/engine/reference/builder. Luettu: 9.1.2021.

Docker Hub. Drupal. Luettavissa: https://hub.docker.com/_/drupal/. Luettu: 27.4.2021

Docs SCS. Kubernetes and OpenShift concepts. Luettavissa: https://docs.csc.fi/cloud/rahti/concepts/. Luettu: 14.3.2021.

Dumpleton, G. 2018. Deploying to OpenShift. O'Reilly Media, Inc. Luettavissa: https://learning.oreilly.com/library/view/deploying-to-openshift/9781491957158/. Luettu: 27.10.2020.

Fernandes, J. 7.11.2016. Why Red Hat Chose Kubernetes for OpenShift. Red Hat OpenShift blogi. Luettavissa: https://www.openshift.com/blog/red-hat-chose-kubernetes-openshift. Luettu: 27.10.2020.

IBM 2019. Containers. Luettavissa: https://www.ibm.com/cloud/learn/containers. Luettu: 27.10.2020.

IBM Developer Blog. OpenShift arkkitehtuuri. Luettavissa: https://developer.ibm.com/blogs/openshift-101-architecture/. Luettu: 22.11.2020.

Kisller, E. 29.6.2020. A Beginner's Guide to Understanding and Building Docker Images. JFrog. Luettavissa: https://jfrog.com/knowledge-base/a-beginners-guide-to-understanding-and-building-docker-images/#AnatomyofaDockerImage. Luettu: 9.1.2021.

Kubernetes GitHub. Services in Kubernetes. Luettavissa: https://github.com/kubernetes/kubernetes/blob/release-1.0/docs/user-guide/services.md. Luettu: 13.3.2021.

Linux Pipeline. Running a virtual machine on Linux. Luettavissa: https://www.linuxpipeline.com/running-a-virtual-machine-on-linux/. Luettu: 5.11.2020.

Maayan, G. 6.4.2020. OpenShift vs. Kubernetes: The Seven Most Critical Differences. Dataversity. Luettavissa: https://www.dataversity.net/openshift-vs-kubernetes-the-seven-most-critical-differences/. Luettu: 13.5.2021.

McConville, A. & Wagoner, O. 1.8.2019. A brief history of Kubernetes, OpenShift, and IBM. IBM Developer Blog. Luettavissa: https://developer.ibm.com/technologies/containers/blogs/a-brief-history-of-red-hat-openshift/. Luettu: 5.11.2020.

NetApp. What Is Persistent Storage? Luettavissa: https://www.netapp.com/data-management/max-memory-accelerated-data/persistent-storage/. Luettu: 7.3.2021.

OpenShift Cookbook a. What is the OpenShift container platform? Luettavissa: https://cookbook.openshift.org/openshift-container-platform/what-is-the-openshift-container-platform.html. Luettu 27.10.2020.

OpenShift Cookbook b. How do I deploy a pre-existing application image? Luettavissa: https://cookbook.openshift.org/deploying-applications-from-images/how-do-i-deploy-a-pre-existing-application-image.html. Luettu 9.1.2021.

OpenShift Cookbook c. Why do my applications run as a random user ID? Luettavissa: https://cookbook.openshift.org/users-and-role-based-access-control/why-do-my-applications-run-as-a-random-user-id.html. Luettu 14.1.2021.

Red Hat Customer Portal. CHAPTER 2. APPLICATION LIFE CYCLE MANAGEMENT. Luettavissa: https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/applications/application-life-cycle-management. Luettu: 3.12.2020.

Red Hat OpenShift Documentation a. The OpenShift Container Platform control plane. Luettavissa: https://docs.openshift.com/container-platform/4.5/architecture/control-plane.html. Luettu: 22.11.2020.

Red Hat OpenShift Documentation b. Understanding Deployments and DeploymentConfigs. Luettavissa: https://docs.openshift.com/container-platform/4.6/applications/deployments/what-deployments-are.html. Luettu: 8.12.2020.

Red Hat OpenShift Documentation c. Creating and using config maps. Luettavissa: https://docs.openshift.com/container-platform/4.5/authentication/configmaps.html. Luettu: 28.3.2021.

Red Hat OpenShift Documentation d. Providing sensitive data to pods. Luettavissa: https://docs.openshift.com/container-platform/4.5/nodes/pods/nodes-pods-secrets.html. Luettu: 28.3.2021.

Red Hat OpenShift Documentation e. Understanding networking. Luettavissa: https://docs.openshift.com/container-platform/4.5/networking/understanding-networking.html. Luettu: 13.3.2021.

Red Hat OpenShift Documentation f. Routes. Luettavissa: https://docs.openshift.com/enterprise/3.0/architecture/core_concepts/routes.html. Luettu: 13.3.2021.

Red Hat OpenShift Documentation g. Understanding persistent storage. Luettavissa: https://docs.openshift.com/container-platform/4.5/storage/understanding-persistent-storage.html. Luettu: 7.3.2021.

Red Hat OpenShift Documentation h. Understanding image builds. Luettavissa: https://docs.openshift.com/container-platform/4.5/builds/understanding-image-builds.html. Luettu: 5.3.2021.

Red Hat OpenShift Documentation i. What are Operators? Luettavissa: https://docs.openshift.com/container-platform/4.5/operators/understanding/olm-what-operators-are.html. Luettu: 10.4.2021.

Red Hat OpenShift Documentation j. Installing Operators in your namespace. Luettavissa: https://docs.openshift.com/container-platform/4.5/operators/user/olm-installing-operators-in-namespace.html. Luettu: 10.4.2021.

Red Hat OpenShift Documentation k. Understanding OperatorHub. Luettavissa: https://docs.openshift.com/container-platform/4.5/operators/understanding/olm-understanding-operatorhub.html. Luettu: 10.4.2021.

Red Hat OpenShift Documentation l. Understanding containers, images, and imagestreams. Luettavissa: https://docs.openshift.com/container-platform/4.6/openshift_images/images-understand.html. Luettu: 9.1.2021.

Red Hat OpenShift Documentation m. Creating images. Luettavissa: https://docs.openshift.com/container-platform/4.6/openshift_images/create-images.html#images-create-guide-openshift_create-images. Luettu: 14.1.2021.

Red Hat OpenShift Documentation n. Viewing application composition using the Topology view. Luettavissa: https://docs.openshift.com/container-platform/4.6/applications/application_life_cycle_management/odc-viewing-application-composition-using-topology-view.html. Luettu: 6.2.2021.

Sengupta, S. 3.11.2020. Kubernetes vs OpenShift: What's The Difference? Bcm blogs. Luettavissa: https://www.bmc.com/blogs/kubernetes-vs-openshift/. Luettu: 13.5.2021.

Source-To-Image. OpenShift/source-to-image GitHub. Luettavissa: https://github.com/openshift/source-to-image. Luettu: 8.1.2021.

Tozzi, C. 4.3.2020. How to Choose the Right Kubernetes Distribution. ITPro Today. Luettavissa: https://www.itprotoday.com/hybrid-cloud/how-choose-right-kubernetes-distribution. Luettu: 13.5.2021.