LAB University of Applied Sciences

MillSIGHTS-järjestelmän pilviarkkitehtuurin parantaminen

Virtuaalikoneista ohjelmistokontteihin

LAB-ammattikorkeakoulu
Insinööri (AMK), tieto- ja viestintätekniikka
2022
Henri Backman

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Backman, Henri	Opinnäytetyö, AMK	2022
	Sivumäärä	
	42	

Työn nimi

MillSIGHTS-järjestelmän pilviarkkitehtuurin parantaminen

Virtuaalikoneista ohjelmistokontteihin

Tutkinto ja koulutusala

Insinööri (AMK), Tieto- ja viestintätekniikka

Toimeksiantajan nimi, titteli ja organisaatio (jos opinnäytetyöllä on toimeksiantaja)

Raute Oyj

Tiivistelmä

Työssä luotiin MillSIGHTS-järjestelmää varten uudistettu pilviarkkitehtuuri, jolla voidaan parantaa vanhan arkkitehtuurin puutteita. Vanhaa arkkitehtuuria käytettäessä uusien asiakaskohtaisten palveluympäristöjen luominen on hidasta ja työlästä, eikä erillisiä ympäristöjä voida hallita keskitetysti.

Arkkitehtuuripäivityksessä uudistetun järjestelmän vaatimuksia olivat hallinnan keskittäminen, uusien MillSIGHTS-ympäristöjen pystyttämisen helpottaminen, versionhallinnan parantaminen ja suorituskyvyn skaalautuvuus. Työssä käytettäviksi teknologioiksi valikoitui Docker, sillä luotavat ohjelmistokontit ja Kubernetes-orkestraattori. Lopuksi luotiin vaatimusten perusteella testiympäristö tutkittuja teknologioita käyttäen.

Ohjelmistokonteilla toteutettu testiympäristö luotiin onnistuneesti ja järjestelmän toiminnallisuus todettiin testidatan avulla. Uutta arkkitehtuuria verrattiin vanhaan ja lopuksi tuotiin esille kehitysideoita jatkoa varten.

Työ tehtiin Raute Oyj:lle kevään 2022 aikana. Työn tuloksena tehtyä arkkitehtuuria on tarkoitus jatkokehittää varsinkin tietoturvaan liittyvien asioiden osalta.

Asiasanat

Docker, Kubernetes, Ohjelmistokontti

Abstract

Author(s)	Type of Publication	Published
Backman, Henri	Thesis, UAS	2022
	Number of Pages	
	42	

Title of Publication

Improving the cloud architecture of MillSIGHTS

From virtual machines to software containers

Degree and field of study

Bachelor of Engineering, Information and communication technology

Name, title and organisation of the client (if the thesis work is commissioned by another party)

Raute Oyj

Abstract

The purpose of the thesis was to renew the cloud architecture of the MillSIGHTS system to improve the architecture of the current system. When using the old architecture creating new service environments is slow and time consuming. Centralized management of individual customer environments is also difficult.

The requirements of the improved architecture were centralized management, easier deployment of new MillISGHTS-environments, improved version control and performance scalability. Docker and software containers as well as Kubernetes were chosen as the technologies to be used. The final step was to create a working demo environment based on the researched technologies and the requirements of the improved system.

A new test system that uses software container was created successfully and the systems functionality was tested with data. The new architecture was compared to the old one and finally development ideas were presented for the future.

The thesis was done for Raute Oyj in the spring of 2022. The new architecture will be developed further in the future. Especially issues related to cyber security will need further research.

Keywords

Docker, Kubernetes, Software container

Sisällys

Lyhenneluettelo

ACR	Azure Container Registry, Azuren konttirekisteri
AKS	Azure Kubernetes Service, Azuren Kubernetes-palvelu
IaaS	Infrastructure as a service, infrastruktuuri palveluna
IP	Internet Protocol, Internet-protokolla
PaaS	Platform as a service, ohjelmisto palveluna
PLC	Programmable Logic Contorller, ohjelmoitava logiikka
SaaS	Software as a service, ohjelmisto palveluna
SQL	Structured Query Language, standardoitu kyselykieli
TLS	Transport Layer Security, salausprotokolla
VPN	Virtual Proxy Network, virtuaalinen yksityisverkko

1 Johdanto

Opinnäytetyön toimeksiantaja Raute Oyj on maailmanlaajuisesti toimiva puun jalostukseen tarkoitettujen valmistuslaitteistojen toimittaja. Yrityksen tuote- ja palveluvalikoima voidaan jakaa projektitoimituksiin ja teknologiapalveluihin. Projektitoimitukset sisältävät esimerkiksi yksittäisiä koneita, tuotantolinjoja ja koko tehtaan laajuisia toimituksia. Teknologiapalveluihin kuuluvat esimerkiksi asiantuntijapalvelut tuotannon kehittämiseksi, kunnossapitopalvelut sekä modernisoinnit. (Raute a.)

Yksi Rauten tarjoamista digitaalisista palveluista on datan keräämiseen, analysointiin ja raportointiin tarkoitettu MillSIGHTS. Palvelun tarkoituksena on auttaa asiakasta optimoimaan tuotantoprosessia kerätyn datan avulla, jolloin voidaan esimerkiksi parantaa eri tuotantovaiheita ja ennakoida huollon tarvetta. Prosesseja optimoimalla asiakkaan on mahdollista lisätä kokonaistuotantoaan jopa 10 % sekä lisätä parhaimmanlaatuisen vanerin tuotantoa 15 %. MillSIGHTSin asennus kuuluu oletuksena Rauten toimittamiin uusiin laitteistoihin, mutta se on mahdollista asentaa myös ulkopuolisen tahon tuotantolinjoihin. (Raute b.)

MillSIGHTSin lisääntyneen kysynnän takia uusia asiakasympäristöjä luodaan jatkuvasti. Jokainen asiakas saa oman yksityisen verkkosivustonsa, josta kerättyä dataa voidaan tarkastella interaktiivisten raporttien avulla. Lisääntyvän tilausmäärän takia palvelun toimitusprosessissa on havaittu puutteita, sillä yksittäisen asiakaskohtaisen järjestelmän pystytys on nykyisillä toimintatavoilla hidasta ja vaatii paljon työtunteja sekä manuaalista konfigurointia.

Opinnäytetyön tavoitteena on uudistaa MillSIGHTSin järjestelmäarkkitehtuuria siten, että uusien asiakaskohtaisten kokonaisuuksien toimittaminen olisi jatkossa mahdollisimman nopeaa ja helppoa. Tavoitteina on myös parantaa järjestelmän vikasietoisuutta sekä hallittavuutta. Koko tiedonkeruujärjestelmä koostuu useasta erilaisesta osiosta, mutta työssä keskitytään datan visualisointiin liittyvään järjestelmään. Lisäksi tässä opinnäytetyössä toteutetaan uuden arkkitehtuurin perusteella toimiva testiympäristö, jota on tarkoitus kehittää myöhemmin eteenpäin.

Opinnäytetyössä tutustutaan ensin nykyisen MillSIGHTS-järjestelmän arkkitehtuuriin, tunnistetaan ongelmakohdat ja kartoitetaan uudistetun järjestelmän vaatimukset. Seuraavaksi tutustutaan uuden järjestelmän toteuttamiseksi käytettäviin teknologioihin, ja lopuksi rakennetaan uudenmallinen ympäristö tutkittuja teknologioita hyödyntäen. Palvelu toimitetaan ensisijaisesti pilvipalveluna, mutta täysin paikallisesti toimivan järjestelmän toimitus on myös mahdollista. Työssä keskitytään vain pilvipalveluna toimitettavaan järjestelmään, mutta uudistetussa arkkitehtuurissa käytettyjä teknologioita on myös mahdollista soveltaa paikallisissa asennuksissa.

2 Pilvipalvelut

2.1 Pilvipalvelut yleisesti

Pilvipalveluilla tarkoitetaan Internetissä olevia palveluita, jotka on toteutettu keskittämällä palveluja toteuttavat tietokoneet datakeskuksiin. Yksittäinen datakeskus voi sisältää yli 100 000 eri käytössä olevaa palvelintietokonetta. Pilvi on yleisnimitys tällä tavalla toteutetulle palvelulle. Yleisimpiä palveluntarjoajia ovat Amazon Web Services, Google, Microsoft ja Rackspace. Jokainen palveluntarjoaja veloittaa palveluistaan oman mallinsa mukaisesti, mutta yleisimmin laskutus perustuu vuokrattujen resurssien tunti- tai kuukausihintaan. Pilvipalveluiden tarjoajat voivat lisäksi tarjota myös muita lisäpalveluja kuten varmuuskopiointia sekä sähköpostipalveluja. (Heljanko 2014.)

Pilvipalveluiden tärkeitä ominaisuuksia ovat skaalautuvuus ja vikasietoisuus. Käytettävissä olevia resursseja pitää pystyä lisäämään ja vähentämään käyttötarpeiden mukaan nopeasti, eikä esimerkiksi laitteiston äkillinen rikkoutuminen saa vaikuttaa palvelun toimintaan merkittävästi. Pilvipalvelut toteutetaan yleensä virtuaalikoneilla, jotka ovat toisistaan eristettyjä virtuaalisia ympäristöjä. Virtuaalikone toimii perinteisen tietokoneen tavoin eli se sisältää täyden käyttöjärjestelmän ja kaiken siihen liittyvän toiminnallisuuden. Suurimpana erona on, että useampi virtuaalikone voi käyttää saman fyysisen palvelintietokoneen laskentatehoa samanaikaisesti. Tällä tavalla voidaan jakaa käytössä oleva laskentateho mahdollisimman tehokkaasti asiakkaiden käyttöön. (Heljanko 2014.)

2.2 Erilaisia pilvipalvelumalleja

Erilaiset pilviratkaisut voidaan jakaa yleensä neljään erilaiseen malliin. Ensimmäinen on Public Cloud eli julkinen pilvi. Pääsy julkiseen pilveen on kaikille avoin eli käyttäjien kommunikointia Internetin kautta ei ole erityisesti rajoitettu. Toinen pilvimalli on Private Cloud eli yksityinen pilvi. Yksityiset pilvet ovat yleensä yritysten tai organisaatioiden hallinnoimia, ja käyttäjien pääsy niihin on tarkkaan rajattu esimerkiksi yrityksen työntekijöihin. Kolmas malli on Community Cloud eli yhteisön pilvi. Tällaiset pilviratkaisut sisältävät esimerkiksi tietyn toimialan eri yritysten yhdeksi kokonaisuudeksi yhdistetyt yksityiset pilvet. Viimeinen pilvimalli on Hybrid Cloud eli hybridipilvi. Hybridipilvessä yhdistyvät julkisen sekä yksityisen pilven ominaisuudet eli kokonaisuuteen voi kuulua Internetissä avoimesti toimivia sovelluksia samalla kun osa sovelluksista toimii vain verkon sisäisesti. (Rani & Ranjan 2014, 458–461.)

Pilvipalveluiden yhteydessä puhutaan erilaisista palvelumalleista, jotka jaetaan kolmeen eri tyyppiin. Ensimmäinen on Infrastructure as a Service (IaaS) eli infrastruktuuri palveluna. Tällä tarkoitetaan sitä, että palveluntarjoaja toimittaa pelkästään verkon yli käytettävää

palvelinkapasiteettia ja huolehtii palveluun liittyvistä tiloista, laitteista sekä niiden ylläpidosta. Asiakkaan vastuulle jää kaikki muu toteutus, johon sisältyy esimerkiksi käyttöjärjestelmän asennus, käytettävät sovellukset ja niihin liittyvä tietoturva sekä tietokantojen ylläpito. (Mohammed & Zeebaree 2021, 17–30.)

Platform as a Service (PaaS) eli sovellusalusta palveluna on toinen kolmesta palvelumallista, jossa palveluntarjoaja tarjoaa palveluna sovellusalustoja, joita käytetään yleensä ohjelmistojen kehittämiseen ja julkaisuun. Tässä mallissa asiakas vastaa pelkästään sovelluksista ja niiden konfiguroinnista. Palveluntarjoajan vastuulle jää tällöin kaiken muun vaadittavan toimittaminen ja ylläpito. Näihin kuuluvat esimerkiksi palvelimet, tallennustilat, tietoverkkojen hallinnointi sekä käyttöjärjestelmien ylläpito. (Mohammed & Zeebaree 2021, 17–30.)

Kolmas palvelumalli on nimeltään Software as a Service (SaaS) eli ohjelmisto palveluna. Software as a Service on tutuin pilvipalvelun muoto, sillä mallissa palveluntarjoaja toimittaa valmiin sovelluksen asiakkaan käytettäväksi. Asiakkaan ei tarvitse huolehtia kuin sovelluksen käytöstä. (Mohammed & Zeebaree 2021, 17–30.) Yleisimpiä SaaS-mallisia palveluita ovat esimerkiksi selainpohjaiset sähköpostipalvelut, tekstinkäsittelyohjelmat sekä kalenteripalvelut (Elmubarak, Yousif & Bashir 2017, 65–71).

3 MillSIGHTS-järjestelmän arkkitehtuuri

3.1 Tiedonkeruu

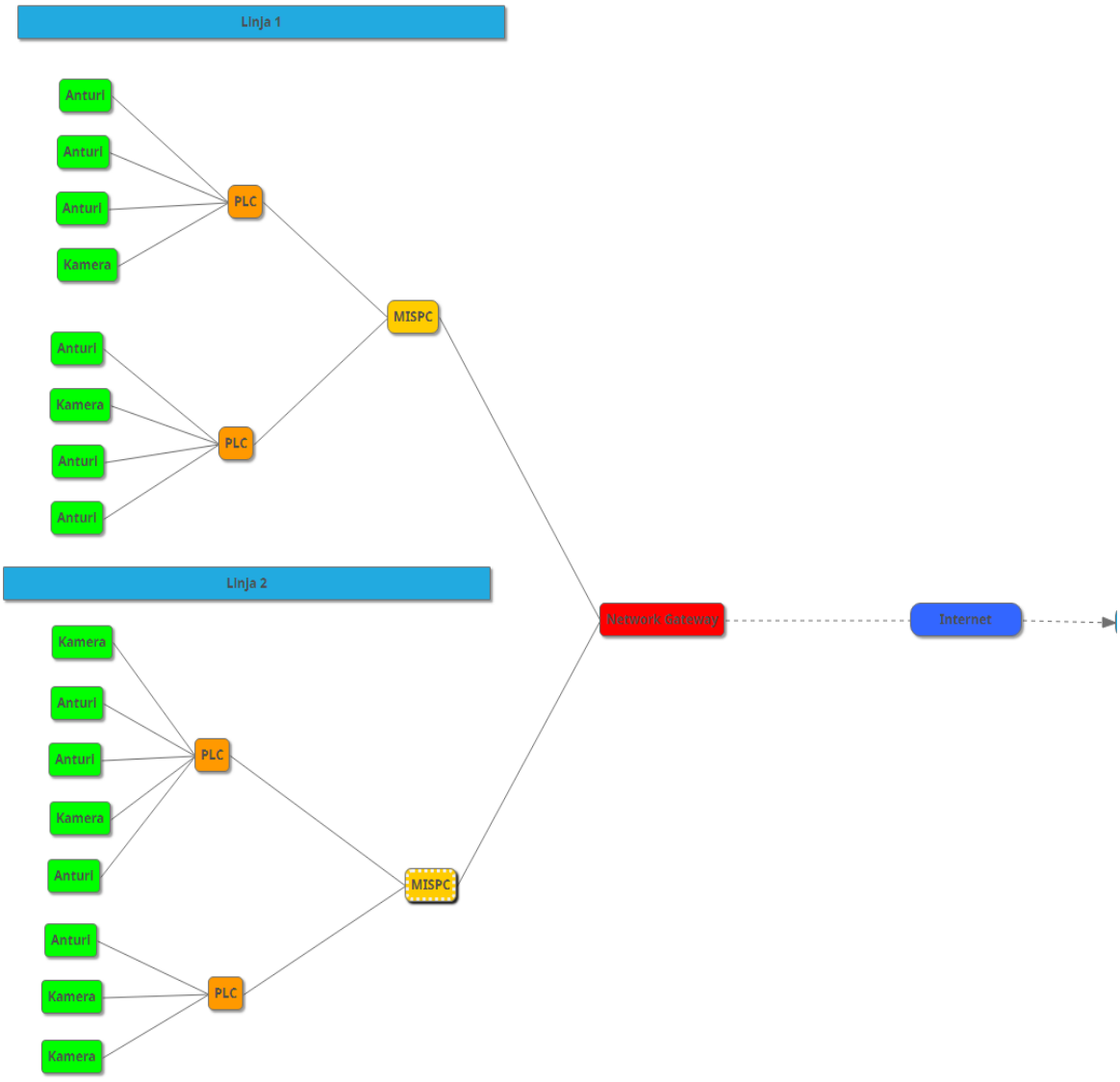
MillSIGHTSia voidaan pitää yksittäisenä palveluna, vaikka järjestelmä rakentuukin useammasta erilaisesta ohjelmistokokonaisuudesta, jotka kommunikoivat keskenään. MillSIGHTS-järjestelmän osat voidaan jakaa toiminnallisuuden perusteella erilaisiin lohkoihin. Tiedonkeruu on yksi järjestelmän tärkeimmistä kokonaisuuksista, mutta tässä opinnäytetyössä sitä tarkastellaan vain pintapuolisesti, koska tarkoituksena on keskittyä pilviarkkitehtuuriin parantamiseen.

Dataa kerätään tuotantolinjoilta lähes jokaisesta tuotantoprosessin vaiheesta erilaisilla antureilla ja kameroilla. Antureita luetaan niihin yhdistetyillä ohjelmoitavilla logiikoilla, joista käytetään yleensä lyhennettä PLC (Programmable Logic Controller). Tiedon kerääminen ja tallentaminen tapahtuu erillisillä tietokoneilla, joista käytetään nimitystä MISPC. Logiikan kanssa kommunikoinnin lisäksi keruutietokoneiden tarkoituksena on puskuroida kerättyä dataa eli tallentaa se tilapäisesti saatavilla olevalle tallennusmedialle. Näin pyritään minimoimaan mahdollista datan häviämistä tilanteissa, joissa yhteys vastaanottavaan palvelimeen ja samalla päätietokantaan katkeaa.

CeCapture ja sen yhteydessä toimivat CeProdControl, CeCapture Configurator ja CeUI ovat Rauten kehittämiä ohjelmia, joilla luetaan logiikan antureista keräämää dataa, kirjoitetaan logiikkaan arvoja ja luodaan operaattorille graafinen käyttöliittymä. Luetusta datasta muodostetaan tapahtumia, joilla ilmaistaan linjan eri toimintojen arvoja mittaushetkellä. CeCapturen asetuksissa määritellään tarkemmin, mitä dataa kerätään ja minne se lähetetään. Kerätyn datan perusteella muodostetut tapahtumat lähetetään yleensä tietokantapalvelimelle jatkokäsittelyä varten. Jos kyseessä on asiakkaan tehtaan sisäverkossa toimiva MillSIGHTSin paikallinen asennus, lähetetään tapahtumat eri linjojen MISPC:iltä suoraan tehdasverkossa sijaitsevalle palvelimelle.

Pilvipalveluna toimitettavien järjestelmien tapahtumien lähetys tapahtuu omalla tavallaan. Tapahtumien lähetys toteutetaan erillisen verkkoyhdyskäytävän (Network gateway) kautta. Yhdyskäytävänä toimii yleensä yksi erillinen tietokone, jonka läpi kaikkien tuotantolinjojen muodostamat tapahtumat ohjataan. Näin vain yhden tietokoneen tarvitsee olla yhdistettynä Internetiin. Jos asiakkaan tehdas on pieni ja MillSIGHTSiin on yhdistetty vain yksittäisiä linjoja, yhdyskäytävänä voidaan käyttää myös yhtä tiedonkeruussa samaan aikaan toimivaa MISPC:tä. Yhdyskäytävänä olevalle tietokoneelle asennetaan erillinen Rauten kehittämä Datatransfer-ohjelma, jonka tehtävänä on salata tuotantolinjoilta kerätystä datasta muodostetut tapahtumat sekä lähettää ne pilvessä olevalle tietokantapalvelimelle.

Tiedonsiirto salataan TLS-protokollaa (Transport Layer Security) käyttäen. TLS:n tarkoitus on todentaa lähettäjän ja vastaanottajan oikeellisuus sekä kryptata lähetettävä data sellaiseen muotoon, että vain oikea vastaanottaja ymmärtää viestin sisällön. Todentamiseen käytetään erilaisia sertifikaattipareja, joiden avulla palvelin varmistaa lähettäjän oikeellisuuden. (Cloudflare.) Kuviossa 1 nähdään, millainen yksinkertaistettu tiedonkeruun arkkitehtuuri voisi olla esimerkiksi kahden tuotantolinjan asennuksessa.



Kuvio 1. Yksinkertaistettu esimerkki tiedonkeruun arkkitehtuurista

3.2 Välityspalvelin

Ensimmäinen järjestelmän pilviarkkitehtuuriin kuuluva osa on käänteinen välityspalvelin (Reverse Proxy Server), joka toimii tuotantotiloissa olevan yhdyskäytävän tavoin datan kulkuväylänä yksityiseen pilveen. Välityspalvelin on pilven ainoa suoraan Internetiin yhdistetty virtuaalikone, jonka tärkeimpänä tehtävänä on välittää tiedonkeruun lähettämät tapahtumat oikealle tietokantapalvelimelle. Palvelin toimii samalla myös palomuurina: se rajoittaa sallittuja yhteyksiä suodattamalla läpi pääsevää liikennettä lähettäjän identifioivan IP- eli Internet Protocol -osoitteen perusteella, jolloin vain tietyistä IP-osoitteista hyväksytään sanomia. Välityspalvelin on myös TLS-salauksen toinen osapuoli, eli lähettäjän sallittu IP-osoite ei vielä riitä liikennöinnin hyväksymiseen, vaan lähettäjältä vaaditaan myös yhteensopiva sertifikaatti. Palvelin toimii arkkitehtuurissa välikätenä ja lisää tietoturvaa, sillä nyt tärkeä tietokantapalvelin pysyy eristettynä pilven virtuaalisen verkon sisällä, eikä tiedonkeruu kommunikoi suoraan tietokannan kanssa.

Välityspalvelin on pieni Linux-pohjainen virtuaalikone, joka ei kuluta paljoa laskentatehoa, joten kustannukset pysyvät palvelimen osalta pieninä. Verkkoliikenteen uudelleenohjaus on toteutettu Nginx- ja Apache-ohjelmistoilla. Nginx on avoimen lähdekoodin hyvin suorituskykyinen HTTP- ja proxy-palvelin (Nginx). Apache on toistaiseksi käytetyin avoimen lähdekoodin HTTP-palvelinohjelmisto (Apache). Nginx vastaanottaa saapuvan verkkoliikenteen ja ohjaa sen Apache HTTP-ohjelmistolle, kun lähettäjän oikeellisuus on ensin varmennettu. Nginx purkaa TLS-salauksen ja siirtää verkkoliikenteen sen jälkeen Apachelle, joka salaa sen uudestaan eri sertifikaattiparia käyttäen. Apache ohjaa verkkoliikenteen eteenpäin tietokantapalvelimelle.

3.3 Tietokantapalvelin

MillSIGHTSin keräämän datan tallentamiseen ja jatkokäsittelyyn käytetään Windows-käyttöjärjestelmällistä virtuaalikonetta. Koska suurin osa datan prosessoinnista tapahtuu tietokantapalvelimella, vaatii virtuaalikone huomattavasti enemmän laskentatehoa käyttöönsä, jotta järjestelmän toiminta olisi mahdollisimman sujuvaa.

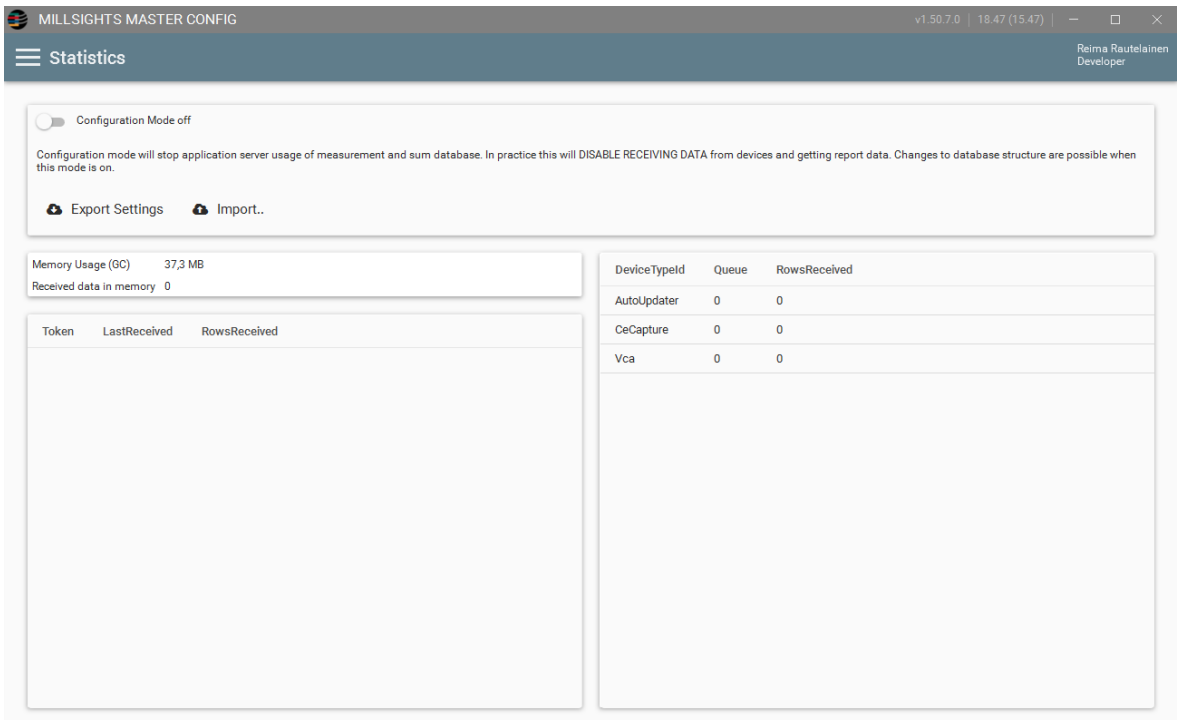
Virtuaalikoneeseen yhdistetään Windows Remote Desktop Connection -ohjelmalla. Näin saadaan muodostettua etäyhteys, jolloin virtuaalikonetta voidaan käyttää normaalin tietokoneen tavoin. Tietokantapalvelin on yhdistettynä pilven sisäiseen virtuaaliseen verkkoon eikä yhteyttä palvelimeen siksi voida muodostaa suoraan Internetin yli. Tämän takia käytetään VPN-yhteyttä (Virtual Private Network), jolloin voidaan muodostaa salattu yhteys Internetin yli pilven virtuaaliseen verkkoon. Gokulakrishnan ja Bain (2014, 3–5) mukaan VPN-yhteys muodostaa salatun tunnelin käyttäjän ja VPN-palvelimen välille, jolloin ulkopuoliset

tahot eivät pääse liikkuvaan dataan käsiksi. Kun yhteys on muodostettu, käyttäjän laite saa uuden IP-osoitteen virtuaalisesta verkosta, jolloin voidaan kommunikoida muiden samaan verkkoon yhdistettyjen laitteiden kanssa.

3.4 MillSIGHTS Application Server ja Master Config

Tietokantapalvelimelle asennetaan Raute Oyj:n kehittämä ja ylläpitämä MillSIGHTS Application Server -ohjelmisto. Application Serverin tärkeimpiä toimintoja ovat tiedonkeruusta tulevien tapahtumien vastaanottaminen, tapahtumien purkaminen tietokantaan sekä kerätystä datasta tehtävien laskujen eli summien laskeminen. Lisäksi Application Serverin vastuulla on muiden ulkoisten määritettyjen API-rajapintojen sekä verkkokäyttöliittymän palveleminen tiedon kyselyissä.

MillSIGHTS Master Config (Kuva 1) on graafinen käyttöliittymä, jonka avulla voidaan muuttaa Application Serverin asetuksia. Master Configilla voidaan esimerkiksi estää tai sallia tapahtumien purkaminen tietokantaan, muokata valmiiksi olemassa olevia summauksia tai lisätä niitä. Käyttöliittymän avulla on myös mahdollista muuttaa tietokannan taulujen määrityksiä, tuoda pakatuista asetustiedostoista aiemmin luotuja raportteja, taulujen ja summien määrityksiä sekä monia muita järjestelmän toimintaan vaikuttavia asetuksia.



Kuva 1. MillSIGHTS Master Configin etusivu

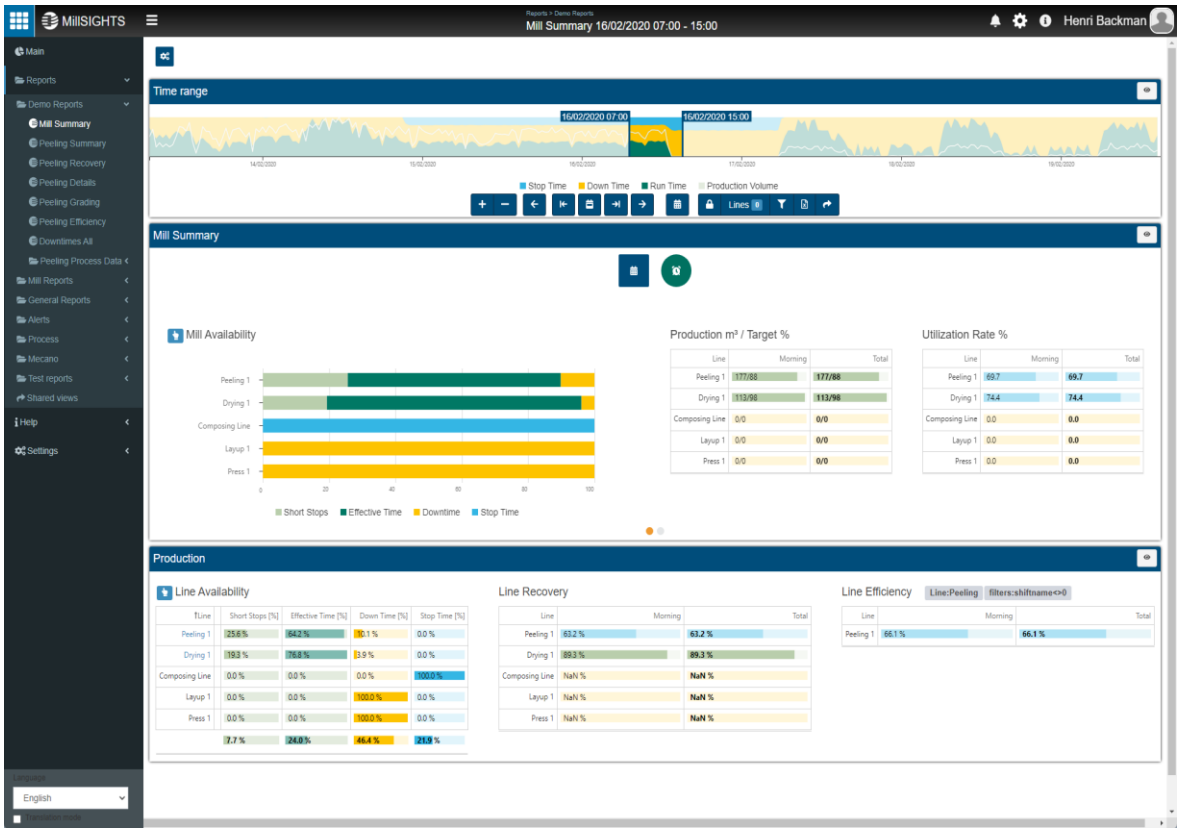
3.5 Tietokanta

Tiedonkeruun tuottama data tallennetaan tietokantapalvelimella datan lähteestä riippuen eri relaatiotietokantoihin. Master Config luo käyttäjän määrittelemät tietokannat automaattisesti ja tekee tietokantojen tauluihin tarvittaessa muutoksia: käyttäjän ei tarvitse itse muistaa tietokantaan tehtäviä SQL-kyselykielen (Structured Query Language) komentoja, vaan kyselyt tehdään automaattisesti. MillSIGHTS-järjestelmä vaatii toimiakseen vähintään kolme eri tietokantaa, mutta yleensä kantoja luodaan useampia esimerkiksi ennakoivasti tulevaisuudessa käyttöön otettavia uusia järjestelmän ominaisuuksia varten. Ensimmäiseen pakolliseen tietokantaan tallennetaan järjestelmän asetuksia ja muita määrityksiä esimerkiksi selaimen kautta näytettäviin raportteihin liittyen. Toinen kanta on tarkoitettu tiedonkeruun tuottaman mittadatan tallentamiseen, ja kolmanteen tietokantaan tallennetaan summattu data.

MillSIGHTSissa käytetään tällä hetkellä kerättävän datan määrästä riippuen pääasiassa kahta eri relaatiotietokantajärjestelmää. Jos dataa kerätään todella monesta eri tuotantolinjasta, käytetään silloin yleensä MSSQL-tietokantaa. MSSQL on Microsoftin kehittämä suurille datamäärille tarkoitettu relaatiotietokantajärjestelmä (Microsoft a). Mikäli kerättävän datan määrä on pientä, eli jos kyseessä on esimerkiksi vain yhden tuotantolinjan MillSIGHTS-asennus, käytetään tietokantajärjestelmänä MariaDB:tä. MariaDB on avoimen lähdekoodin relaatiotietokanta, jonka ovat luoneet suositun MySQL-relaatiotietokantaohjelmiston alkuperäiset kehittäjät (MariaDB).

3.6 Web-käyttöliittymä ja asiakasportaali

Summattua dataa tarkastellaan verkkokäyttöliittymän erilaisten raporttien (Kuva 2) avulla. Käyttöliittymä on yksi MillSIGHTSin tärkeimmistä ominaisuuksista ja monesti järjestelmän ainoa pilviarkkitehtuurin käyttäjälle näkyvä osa. Käyttöliittymän avulla on mahdollista määrittää yksityiskohtaisesti, mitä raporteissa näytetään. Selaimen kautta pystytään myös esimerkiksi tilaamaan sähköpostihälytyksiä tiettyjen ehtojen täyttyessä sekä asettamaan linjakohtaisia parametreja, joilla voidaan vaikuttaa summauksiin ja tiettyihin raportteihin.



Kuva 2. MillSIGHTSin web-käyttöliittymä

Asiakkaat pääsevät helposti katsomaan raportteja erillisen asiakasportaalin kautta. Järjestelmään kirjaudutaan henkilökohtaisilla tunnuksilla, ja käyttäjän oikeuksista riippuen kirjautumisen jälkeinen näkymä voi olla erilainen. Yleensä käyttäjällä on pääsy vain yhteen MillSIGHTS-ympäristöön, jolloin kirjautumisen jälkeen selainistunto uudelleenohjataan suoraan oikeaan raporttinäkymään. Joissain tapauksissa käyttäjällä voi kuitenkin olla pääsy useampaan MillSIGHTS-ympäristöön. Silloin haluttu raporttinäkymä valitaan kirjautumisen jälkeen näkyvästä listasta.

4 Arkkitehtuurin päivitys

4.1 Järjestelmän puutteet

Edellisessä luvussa kuvailtiin millaisista eri osista yksittäinen MillSIGHTS asennus koostuu. Vaikka järjestelmä on tällä hetkellä täysin toimiva, on siinä silti huomattavia puutteita ja ongelmia, joihin tässä opinnäytetyössä pyritään löytämään ratkaisuja. Yksittäisen pilviympäristön rakentaminen pitää aloittaa aina kokonaan alusta asti eikä järjestelmän eri komponentteihin ole olemassa valmiita pohjia, joita olisi mahdollista hyödyntää uudelleen.

Uuden MillSIGHTS-ympäristön rakentamista varten on olemassa erilaisia asiakirjoja, joiden ohjeita seuraamalla on mahdollista pystyttää uusi ympäristö ilman aiempaa kokemusta. Ohjeet ovat kuitenkin todella pitkiä ja monimutkaisia, joten jos järjestelmä on ennestään tuntematon, voi uuden ympäristön pystyttämiseen kulua jopa useamman viikon työtunnit. Kokeneemmaltakin työntekijältä uuden MillSIGHTS-ympäristön luomiseen kuluu yleensä noin yhdestä kahteen työpäivää. Lisäksi osa ohjeista on vanhoja, eivätkä niissä kerrotut asiat välttämättä enää pidä paikkaansa. Välillä ohjeiden seuraamisesta huolimatta eteen saattaa tulla ajoittain erilaisia ongelmatilanteita, joiden selvittämiseen voi kulua runsaasti aikaa.

Eniten aikaa uuden ympäristön luomisessa kuluu yleensä käänteisen välityspalvelimen tekemiseen. Verkkoliikenteen uudelleenohjaamiseen käytettävät Nginx ja Apache asennetaan virtuaalikoneelle joka kerta alusta asti. Ohjelmien asentamisessa kestää yleensä useita tunteja, koska palvelimelle on varattu vain vähän laskentatehoa, sillä sitä ei normaalissa toiminnassa tarvita paljoa. Lisäksi TLS-salauksessa käytettävien sertifikaattien siirtäminen ja Nginx:n sekä Apachen asetusten muuttaminen oikeanlaisiksi pitää tehdä kokonaan käsin – molemmat ovat työläitä ja aikaa vieviä prosesseja.

Myös tietokantapalvelimen kanssa menee paljon aikaa hukkaan, sillä sitäkään varten ei ole mitään valmista mallia, josta palvelimelle tulevat ohjelmistot saisi nopeasti pystytettyä. Tietokanta ja Application Server asennetaan joka kerta alusta asti käsin. Application Serverin versionhallintaa varten on olemassa erillinen ohjelmisto, jolla voidaan päivittää sovelluksen versio automaattisesti uudemmaksi, mutta myös päivitysohjelmiston asennus on käyttäjän vastuulla, ja siihenkin kuluu aikaa useampi tunti.

Edellä mainittujen ongelmien lisäksi yksi suurimpia puutteita on keskitetyn hallinnan puute. Koska jokainen MillSIGHTS-ympäristö toimii omalla virtuaalikoneellaan, ovat kaikki järjestelmät omia erillisiä kokonaisuuksiaan. Tämän takia kaikkiin ympäristöihin vaikuttavat päivitykset ja muutokset pitää tehdä käsin yksi ympäristö kerrallaan. Keskitetyn hallinnan puuttumisen lisäksi eri MillSIGHTS-asennuksien kanssa ei ole mahdollista tarkkailla järjestelmien eheyttä, vaan vikatilanteet huomataan yleensä vasta pitkän ajan jälkeen. Joissain

MillSIGHTS-asennuksissa on käytössä automatisoitu login-tester, joka kirjautuu automaattisesti ennalta määriteltyihin ympäristöihin ja lähettää sähköpostihälytyksen, jos kirjautumisessa on ongelmia. Tällä saadaan varmistettua, että testerin tarkkailemat ympäristöt pysyvät asiakkaan saatavilla, mutta muuta toiminnallisuutta sen lisäksi ei voida varmistaa.

4.2 Uudistetun arkkitehtuurin vaatimukset

Luvussa 4.1 tarkasteltiin MillSIGHTS-järjestelmän keskeisimpiä puutteita, ja seuraavaksi laaditaan tässä opinnäytetyössä tehtävälle arkkitehtuurin uudistukselle erilaisia vaatimuksia, joista mahdollisimman monelle etsitään ratkaisu. Tarkoituksena ei ole tehdä uutta täysin valmista lopullista ratkaisua vaan luoda testiympäristö pohjaksi uudelle arkkitehtuurille.

Tärkein vaatimus uudelle arkkitehtuurille on, että uusien asiakasympäristöjen luomisesta tulee helpompaa ja nopeampaa. Uudistetun arkkitehtuurin tulee mahdollistaa uusien MillSIGHTS-ympäristöjen nopea luominen ja poistaminen. Mahdollisimman monen työntekijän tulisi pystyä luomaan uusia ympäristöjä itsenäisesti. Lisäksi pyritään vähentämään käsin tehtävien vaadittavien muutosten määrää tai ainakin nopeuttamaan niiden tekemistä.

Uudistuksella on myös tarkoitus vähentää järjestelmän käyttöön varattavien laskentaresurssien määrää. Virtuaalikonetta luotaessa määritetään, kuinka paljon resursseja kuten prosessorin ytimiä, keskusmuistia ja kiintolevytilaa yksittäinen kone saa käyttöönsä. Tämä aiheuttaa sen, että yhden MillSIGHTS-ympäristön ylläpitokustannukset pysyvät samana riippumatta siitä, kuinka paljon varattua laskentatehoa oikeasti käytetään. Siksi yksi uudistetun arkkitehtuurin vaatimuksista on varattavan laskentatehon tehokkaampi käyttö vaarantamatta nykyistä suorituskykyä.

Arkkitehtuurin uudistuksen yhteydessä huomioidaan myös hallittavuuden parantaminen. Eri MillSIGHTS-ympäristöjä tulee pystyä ohjaamaan keskitetysti samasta paikasta. Keskitetyn hallinnan on tarkoitus auttaa järjestelmän ylläpidossa esimerkiksi Application Serverin versionhallinnan osalta. Lisäksi vähennetään laskentaresurssien varaamista manuaalisesti. Hallinnan keskittämisen myötä pyritään myös parantamaan järjestelmän tarkkailua, jotta palvelu olisi mahdollisimman hyvin asiakkaan saatavilla ja raporteissa näkyvä data ajankohtaista.

4.3 Uudistetun arkkitehtuurin suunnitelma

Päivityssuunnitelman tekeminen aloitetaan muodostamalla edellisen luvun pohjalta tutkimuskysymyksiä, joihin suunnitelman on tarkoitus vastata:

- Miten nopeutetaan ja helpotetaan yksittäisen ympäristön pystyttämistä?

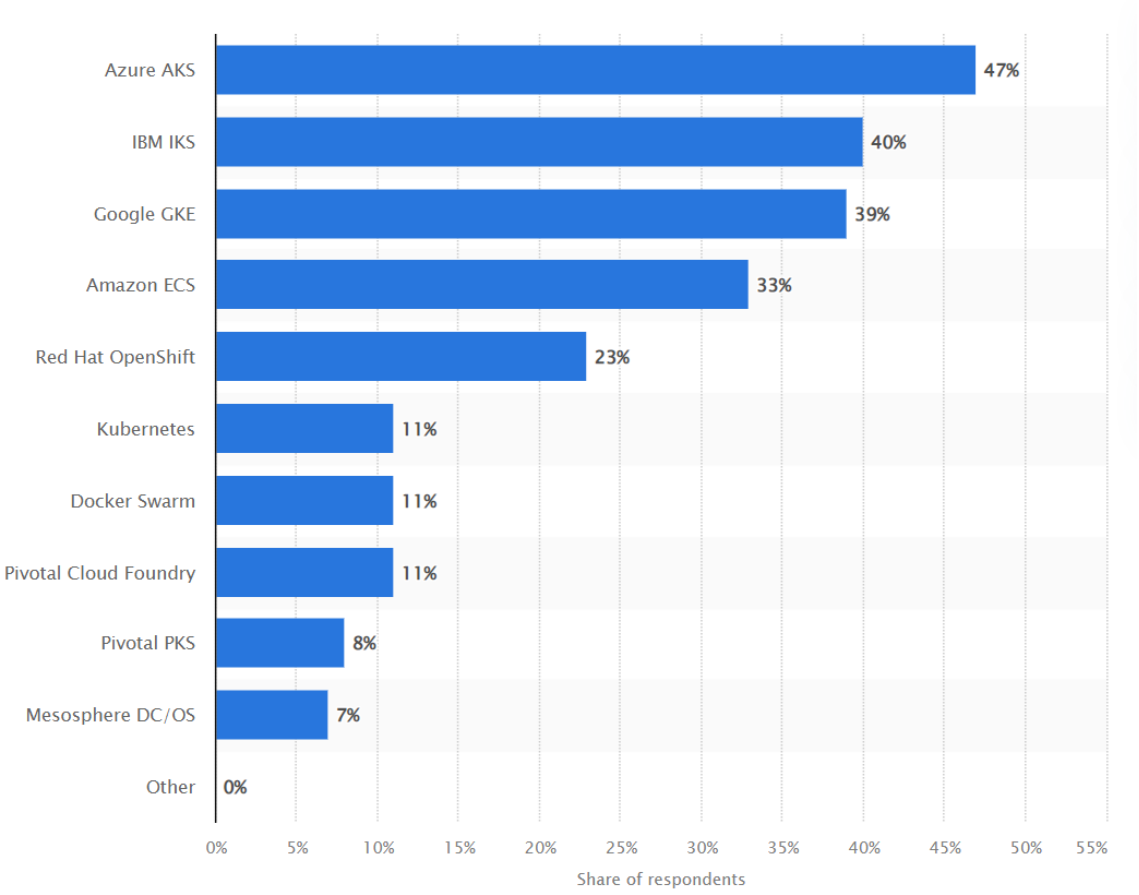
- Miten parannetaan ympäristöjen hallittavuutta sekä versionhallintaa?
- Miten varmistetaan toimivuus ja tarkkaillaan järjestelmää häiriöiden varalta?
- Onko järjestelmää mahdollista skaalata tarvittaessa suorituskyvyn parantamiseksi?
- Voidaanko ohjelmistojen versionhallintaa parantaa jotenkin?

Moni nykyisen järjestelmän ongelmista voidaan korjata muuttamalla arkkitehtuuria siten, että hyödynnetään toteutuksessa erilaisia ohjelmistokontteja (software container). Söderlundin (2019) mukaan ohjelmistokontti on virtuaalinen ympäristö, jota ajetaan isäntäkoneen käyttöjärjestelmän päällä. Kontit käyttävät isäntäkoneen kerneliä, minkä takia konttien luominen ja lopettaminen on nopeaa.

Ohjelmistokontteja käsitellään tarkemmin luvussa 5.1. Ohjelmistokontteja hyödynnettäessä käyttäjän ei tarvitse asentaa jokaista ohjelmistoa eri MillSIGHTS-ympäristöihin itse, vaan asennuksen helpottamiseksi voidaan käyttää valmiiksi luotuja ohjelmistokonttien näköistiedostoja. Tällä tavalla käyttäjän tarvitsee muuttaa käsin vain asiakaskohtaisia asetuksia.

Erillään olevien MillSIGHTS-ympäristöjen hallintaa parannetaan ottamalla käyttöön orkestraattoriohjelmisto. Orkestraatiolla tarkoitetaan tässä tapauksessa tietokonejärjestelmien, sovellusten ja palvelujen automatisoitua konfigurointia, hallintaa ja koordinointia (Red Hat, 2019). Orkestraattorilla on mahdollista hallita useita eri MillSIGHTS-asennuksia keskitetysti. Samalla voidaan tarkkailla käytettyä laskentatehoa sekä päivittää käytössä olevia ohjelmistoja yhdestä paikasta. Tässä opinnäytetyössä on valittu käytettäväksi Kubernetes-orkestraattori.

Ohjelmiston valinta tehtiin vertailemalla yleisimpiä orkestraattoriohjelmistoja ja niiden ominaisuuksia ottaen huomioon MillSIGHTSin vaatimukset. Vertailtavia ohjelmistoja olivat muun muassa Ansible, Terraform, Docker Swarm ja IBM Cloud Pak. Ohjelmistot ovat ominaisuuksiltaan hyvin samanlaisia, ja siksi lopulliseen valintaan vaikutti suuresti kaksi asiaa. Kubernetes on yksi käytetyimmistä orkestraattoriohjelmistoista (Kuvio 2), joten sen käyttämiseen on tarjolla paljon ajankohtaista dokumentaatiota. Kubernetes on alun perin suunniteltu yksinomaan ohjelmistokonttien hallintaan (Vailshery 2022).



Kuvio 2. Yritysten käytetyimmät orkestraattoriohjelmistot vuonna 2019 (Vailshery 2022)

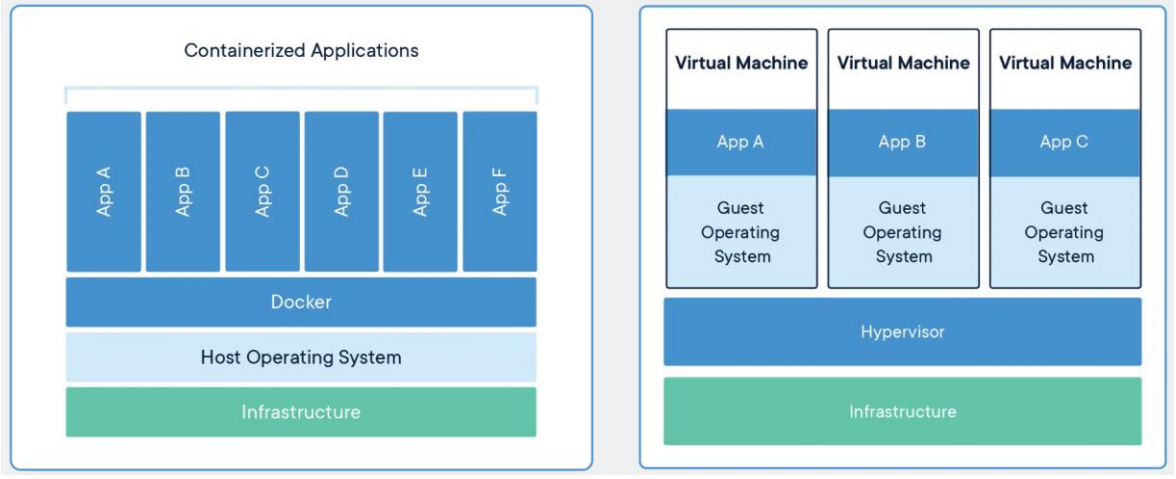
Toinen valintaan suuresti vaikuttanut tekijä on AKS (Azure Kubernetes Service), joka on Microsoftin Azure-pilvipalvelun PaaS-palvelumallin ratkaisu, jolla tarjotaan asiakkaille mahdollisuus käyttää Kubernetesia verkkoselainpohjaisena palveluna. Orkestraattoriohjelmisto on ilmainen, ja käyttäjiä veloitetaan pelkästään käytettyjen resurssien perusteella. (Microsoft b.) Raute Oyj käyttää jo valmiiksi Azure-pilvipalvelua, joten AKS:n käyttöönotto voidaan tehdä helposti. AKS:n kanssa on myös mahdollista käyttää muita Azuren palveluita kuten esimerkiksi ACR:ää (Azure Container Registry), jota voidaan käyttää luotujen ohjelmistokonttien näköistiedostojen säilytykseen. Kubernetesiin tutustutaan tarkemmin luvussa 5.3. Seuraavaksi tutustutaan tässä luvussa esille tuotuihin teknologioihin tarkemmin, minkä jälkeen luodaan uudistettu malli MillSIGHTSin pilviarkkitehtuurin pohjaksi. Sen jälkeen käytetään tutkittuja teknologioita ja luodaan testiympäristö suunnitellun arkkitehtuurimallin mukaan.

5 Käytettävät teknologiat

5.1 Ohjelmistokontti

Uusien MillSIGHTS-ympäristöjen luomisen helpottamiseksi siirrytään käyttämään ohjelmistokontteja. Virtuaalikoneiden tavoin ohjelmistokontit hyödyntävät isäntäjärjestelmän laskentatehoa. Ohjelmistokontit eroavat virtuaalikoneista siten, että virtuaalikone muodostaa täysin virtualisoidun työpöytäympäristön, jolla on oma käyttöjärjestelmä, koneen käyttöön varattuja prosessorin ytimiä, keskusmuistia sekä kiintolevytilaa. Ohjelmistokonteilla puolestaan pakataan jokin tietty toiminnallisuus kuten esimerkiksi sovellus omaksi pieneksi säiliöksi, jota on helppo monistaa sekä käynnistää ja sammuttaa nopeasti. Jokaisella ohjelmistokontilla on oma tiedostojärjestelmänsä, joka mahdollistaa esimerkiksi ylimääräisten tiedostojen siirtämisen kontin sisälle. Yksi merkittävimpiä ohjelmistokonttien etuja on se, että koska kaikki kontin toimintaan vaadittavat sovellusriippuvuudet pakataan kontin sisälle, ohjelmistokontti toimii samalla tavalla toimintaympäristöstään riippumatta.

Virtuaalikoneilla virtualisoidaan siis koko käyttöjärjestelmän toiminnallisuus. Ohjelmistokonteilla virtualisoidaan vain tiettyjä prosesseja tai sovelluksia. Virtuaalikoneet mahdollistavat usean eri käyttöjärjestelmäinstanssin ajamisen samalla palvelintietokoneella, ja ohjelmistokonteilla pystytään puolestaan monistamaan yksittäisiä ohjelmia, jolloin voidaan käyttää vain yhtä käyttöjärjestelmää usean samanlaisen kontin alustana. Ohjelmistokontit eivät myöskään yleensä varaa käyttöönsä tiettyä määrää isäntäkoneen laskentatehosta, vaan käyttävät sitä tarpeen mukaan. Näin voidaan jakaa käytössä olevaa laskentatehoa tasaisemmin eri konttien välillä. Kuviossa 3 havainnollistetaan edellä mainittujen ohjelmistokonttien ja virtuaalikoneiden eroja.



Kuvio 3. Ohjelmistokonttien ja virtuaalikoneiden eroja (Docker a)

5.2 Docker

Tässä opinnäytetyössä käytetään Docker-ohjelmistokonttiteknologiaa, joka on julkaistu vuonna 2013. Docker hyödyntää olemassa olevia laskentakonsepteja konttien ympärillä erityisesti Linux-maailmassa. Dockerin teknologia keskittyy erottelemaan sovellusriippuvuudet infrastruktuurista ja on siksi ohjelmistokehittäjien sekä järjestelmäoperaattoreiden suosiossa. (Docker a.)

MillSIGHTSin muodostavat ohjelmistot muutetaan ohjelmistokonteiksi Dockeria käyttäen. Esimerkiksi tietokannan ja välityspalvelimen sovelluksia varten on jo olemassa valmiita näköistiedostoja, joita voidaan hyödyntää suoraan. Koska Application Server on Rauten oma ohjelmisto, ei sitä varten ole olemassa valmiita ohjelmistokontteja, joten yksi tämän opinnäytetyön vaiheista on luoda toimiva ohjelmistokontti, johon Application Server asennetaan.

Container image

Ohjelmistokontteja luodaan ja tuhotaan lyhyellä aikavälillä, joten kontti ei ole luonteeltaan pysyvä kuten esimerkiksi perinteiset ohjelmistot. Kun kontti tuhoutuu, kaikki siihen liittyvät tiedostot tuhotaan automaattisesti muutamia poikkeuksia lukuun ottamatta kuten persistent volumes, joita käytetään tässä opinnäytetyössä myöhemmin Application Serverin ja Nginx:n käyttämien asetustiedostojen säilytykseen. Tarvitaan siis keino, jolla voidaan tehdä uusia ohjelmistokontteja loputtomasti.

Konttien pohjana käytetään näköistiedostoa (container image), jota käyttämällä voidaan luoda rajaton määrä keskenään samanlaisia ohjelmistokontteja. Näköistiedostot rakennetaan Docekrilla erityistä dockerfile-tiedostoa käyttäen. Dockerfileä käsitellään seuraavassa luvussa lisää. Tässä opinnäytetyössä muodostetaan kaksi eri näköistiedostoa Nginx:ää ja Application Serveriä varten, jotta samoja pohjatiedostoja voidaan käyttää monessa eri MillSIGHTS-ympäristössä.

Dockerfile

Ohjelmistokonttien näköistiedostot eli container imaget rakennetaan käyttämällä dockerfile-tiedostoa, joka sisältää halutun näköistiedoston rakentamiseen vaadittavat ohjeet. Dockerfilen sisältämä koodi suoritetaan järjestyksessä ensimmäisestä rivistä alkaen. Lopputuloksena on halutunlainen container image, jota voidaan käyttää rajattomasti uusien konttien luomisessa. (Docker b.) Tässä opinnäytetyössä luodaan kaksi eri Dockerfielä, jotta voidaan luoda Nginx:lle ja Application Serverille omat container imaget.

Docker ja esimerkiksi Microsoft tarjoavat valmiita näköistiedostoja, joihin on jo asennettu erilaisia ohjelmistoja. Docker tarjoaa esimerkiksi erilaisia Nginx kontteja, joista yhtä

käytetään tässä opinnäytetyössä välityspalvelimen base imagen luomisessa. Application Serverin ohjelmistokontin rakentaminen pitää kuitenkin aloittaa täysin tyhjästä, koska ohjelmistosta ei aiemmin ole tehty kontteja. Ensimmäinen vaihe container imagen tekemiseen tarkoitetun dockerfilen luomisessa on pohjana käytettävän näköistiedoston eli base imagen valinta. Base imagena voidaan käyttää esimerkiksi aiemmin mainittua Nginx-konttia, jota muokataan dockerfilellä halutulla tavalla. Container image ei vaadi toimiakseen kuin base imagen, mutta jos halutaan käyttää pelkkää base imagea suoraan on dockerfilen käyttö siinä tapauksessa turhaa. On myös tilanteita, joissa halutaan luoda uusi container image, joka on täysin samanlainen tai todella vähän muokattu. Tässä opinnäytetyössä tehtävä Nginx:n näköistiedosto toteutetaan tällä tavalla, ja syitä siihen käsitellään container imagen luonnin yhteydessä myöhemmin. Base image voi olla myös puhdas ohjelmistokonttiympäristöön tarkoitettu käyttöjärjestelmän näköistiedosto ilman ylimääräisiä sovelluksia. Base image määrittää myös kontin pohjana toimivan käyttöjärjestelmän, jolloin Windows-pohjaisia sovelluksia ei voida asentaa Linux-kontteihin ja päinvastoin. Docker tukee molempiin käyttöjärjestelmiin pohjautuvia kontteja (Docker a).

Kun pohjana käytettävä base image on valittu, voidaan määrittää lisää erilaisia asetuksia tai suoritettavia toimintoja. Dockerfilellä voidaan esimerkiksi kopioida haluttuja kansioita tai tiedostoja isäntäkoneen kiintolevyltä kontin omaan tiedostojärjestelmään. Dockerfilellä voidaan myös määrittää container imagen rakennusvaiheessa suoritettavia komentoja, joilla voidaan esimerkiksi asentaa haluttuja ylimääräisiä ohjelmistoja tai lisäominaisuuksia. (Docker b.) Edellä mainittuja dockerfilen ominaisuuksia tullaan käyttämään Application Serverin container imagen luonnissa.

5.3 Kubernetes

Seuraavaksi tutustutaan tässä opinnäytetyössä käytettävään Kubernetes-orkestraattoriin tarkemmin. Kubernetes on alun perin Googlen kehittämä avoimen lähdekoodin ohjelmisto, jonka tärkeimpiä ominaisuuksia ovat helppo skaalautuvuus, itsensä korjaaminen, salasanojen hallinta, automatisoidut päivitykset ja tallennustilan hallinta (Kubernetes a). Kubernetes koostuu erilaisista helposti laajennettavista olevista kokonaisuuksista, jotka muodostavat yhden yhdessä toimivan kokonaisuuden, jota kutsutaan Kubernetes Clusteriksi.

Monista hyvistä ominaisuuksistaan huolimatta Kubenretesissa on kuitenkin yksi selkeästi huono puoli. Orkestraattorin käyttö voi alkuun olla todella haastavaa varsinkin, jos ohjelmisto on ennestään tuntematon. Kubernetesin toimintoja hallitaan tekstipohjaisilla asetustiedostoilla, jotka voivat aluksi vaikuttaa hankalilta. Kun orkestraattorin toimintaa ymmärtää paremmin tulee asetusten muokkaamisesta myös helpompaa.

Pod

Kubernetesin pienin toiminnallinen kokonaisuus, joka koostuu yleensä yhdestä tai useammasta ohjelmistokontista, on nimeltään pod eli kotelo tai kapseli. Jokainen pod sisältää kubeletitksi kutsutun toimijan eli agentin, jonka tehtävänä on tarkkailla podin sisällä olevien konttien tilaa ja ylläpitää tekstitiedostoilla määritettyä Clusterin tilaa. (Kubernetes b.)

Node ja Node Pool

Ohjelmistokontit tarvitsevat toimiakseen isäntäkoneen, jonka laskentatehoa käytetään kontin toimintojen suorittamiseen (Kubernetes b). Kubernetesissa node toimii isäntäkoneen tavoin alustana konteille. Orkestraattori päättää uuden podin luomisvaiheessa, mitä nodea sen isäntänä käytetään luomishetkellä vapaana olevien laskentaresurssien perusteella.

Koska ohjelmistokontit vaativat rakennusvaiheessa aina pohjaksi oikeaan käyttöjärjestelmään pohjautuvan base imagen, myös noden tulee käyttää kontille sopivaa käyttöjärjestelmää. Tästä syystä käytössä voi olla erilaisia nodeja konttien käyttöjärjestelmistä riippuen.

Kubernetes tarvitsee toimiakseen aina vähintään yhden noden, jota käytetään orkestraattorin sisäisten toimintojen suorittamiseen. Kontteja varten luodaan Node Pooleiksi kutsuttuja kokonaisuuksia, jotka koostuvat yhdestä tai useammasta samanlaisesta nodesta. Yksittäiselle nodelle annetaan lupa käyttää rajattu määrä laskentatehoa. Node Pool voidaan asettaa skaalautumaan automaattisesti: kun noden käytettävissä oleva vapaa laskentateho loppuu konttien määrän takia kesken, Kubernetes luo automaattisesti uusia nodeja, jotka saavat saman verran resursseja käyttöönsä kuin Node Poolin ensimmäinen node. Asetuksissa voidaan lisäksi määrittää, kuinka monta kopiota ensimmäisestä nodesta Kubernetes saa automaattisesti tehdä. Tällä tavalla voidaan lisätä konttien käytössä olevaa laskentatehoa, jos jokin konttien suorittama toiminto aiheuttaa hetkellisen piikin laskentaresurssien tarpeessa.

Control Plane

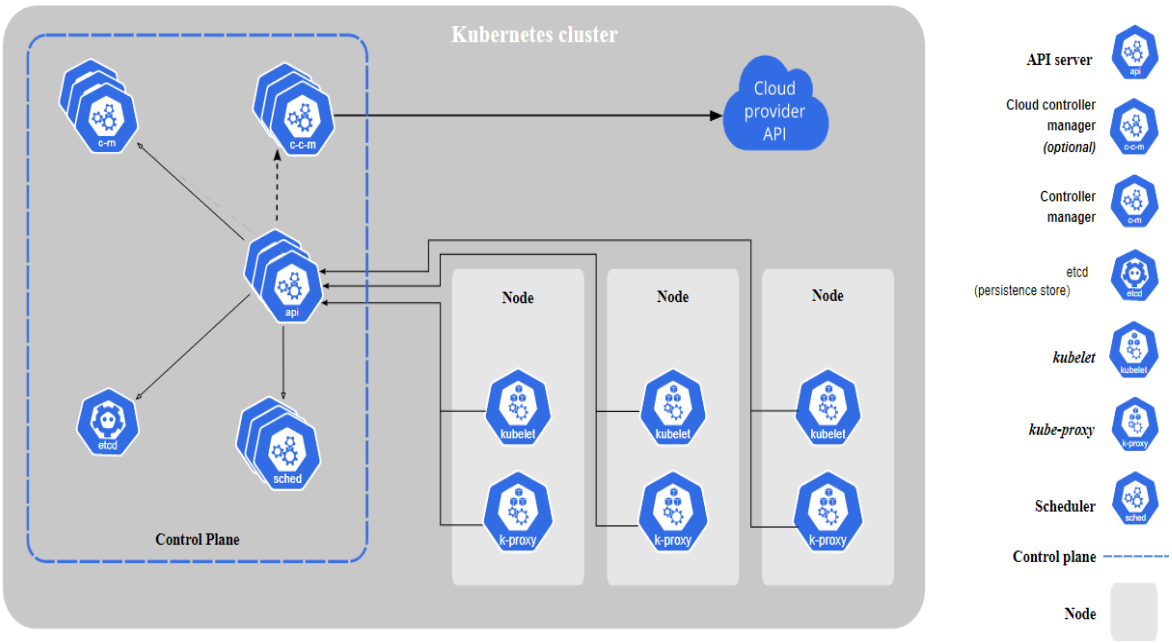
Koko Kubernetes Clusterin tärkeistä päätöksistä vastaa Control Plane, joka pitää sisällään erilaisia hallinnollisia toimintoja. Yksi Control Planen osista on API server, joka toimii rajapintana käyttäjän ja Clusterin välillä. Kubernetesille annettavat komennot tehdään API serverin kautta, joten se on ainoa Control Planen osa, joka on suoraan näkyvissä Clusterin ulkopuolelle. (Kubernetes b.)

Muita Control Planen osia ovat etcd, jonka tehtävänä on tallentaa Clusteriin liittyvää dataa avainarvo pareina sekä Scheduler, jonka tehtävänä on määrittää uusille podeille mahdollisimman sopiva node käyttäjän määritysten perusteella. Control Planeen kuuluu myös

Controller manager, joka esimerkiksi vahtii clusterin podeja, joita se luo sekä poistaa tarvittaessa. (Kubernetes b.)

Cluster

Kaikki edellä mainitut Kubernetesin osat sisältävää kokonaisuutta kutsutaan Kubernetes clusteriksi. Käyttäjä voi halutessaan luoda useamman clusterin eri käyttötarkoituksiin. Yksi käyttötapa on luoda oma Kubernetes cluster eri maanosissa sijaitseville datakeskuksille. Tällä tavalla voidaan vähentää yhteen clusteriin kohdistuvaa kuormaa sekä parantaa käyttäjäkokemusta, koska clusterin palveluita käyttääkseen asiakas yhdistetään maantieteellisesti lähimpänä olevaan palveluun, jolloin käytettävät palvelut toimivat pienemmällä viiveellä. Kuviossa 4 nähdään esimerkki Kubernetes clusterin rakenteesta. Kuviosta on helppo tunnistaa tässä luvussa tarkasteltuja clusterin osia. Kuviossa nähdään lisäksi Kube-proxy, jonka tehtävänä on hallinnoida nodeihin kohdistuvaa verkkoliikennettä.



Kuvio 4. Esimerkki Kubernetes clusterin eri osista (Kubernetes b)

5.4 Microsoft Azure

Tässä opinnäytetyössä käytetään Microsoftin Azure-pilvipalvelua ja siihen kuuluvia erilaisia lisäpalveluita. Arkkitehtuuripäivityksen suunnitelmaan pohjautuva testiympäristö voidaan rakentaa myös ilman pilvipalveluja, mutta skaalautuvuuden ja jatkokehityksen kannalta on parasta käyttää suoraan Azurea testiympäristön alustana.

AKS (Azure Kubernetes Service)

Kubernetesin hallintaan käytetään AKS:ää, joka on Microsoftin tarjoama palvelu, jolla helpotetaan orkestraattorin käyttöä. Microsoft hoitaa Kubernetes clusteriin liittyvän teknisen puolen, jolloin käyttäjän tarvitsee huolehtia vain clusterin sisäisestä konfiguroinnista. (Microsoft b.) AKS:n käyttö on hyödyllistä myös siksi, että palveluun voidaan yhdistää muita Azuren ominaisuuksia kuten erityyppistä tallennustilaa sovellusten tarpeiden mukaan. AKS käyttää konttien pyörittämiseen Azuren tarjoamaa laskentatehoa, joten clusterin suorituskyvyn lisääminen tarpeita vastaavaksi on nopeaa.

ACR (Azure Container Registry)

Dockerilla tehdyt container imaget tallennetaan ACR:ään, joka on Azuren ohjelmistokonttien näköistiedostojen säilytykseen tarkoitettu palvelu (Microsoft c). Kubernetesia käytetään konttien hallinnointiin, mutta uusien konttien luomista varten tarvitaan pohjana toimiva container image. Tähän tarkoitukseen on mahdollista käyttää esimerkiksi Dockerin omaa arkistoa, mutta ACR:ää käytettäessä Kubernetesin tarvitsemat container imaget sijoitetaan lähemmäs clusteria, mikä mahdollistaa näköistiedostojen nopeamman lukemisen. Näin voidaan nopeuttaa uusien ohjelmistokonttien luomista ja keskittää clusterin käyttämät resurssit mahdollisimman hyvin yhteen paikkaan.

Managed Disks

MillSIGHTSin ohjelmistokontit tarvitsevat kiinteää tallennustilaa, johon tallennetaan konttien käyttämät asiakaskohtaiset tiedostot, joiden täytyy pysyä tallessa, vaikka niitä käyttävä kontti tuhoutuisi ja saisi tilalleen uuden kontin. AKS voi käyttää kolmea erityyppistä Azuren tallennustilaa, jotka ovat tiedostojärjestelmä operaatioille tarkoitettu Blob Storage, tiedostojen jakamiseen tarkoitettu Azure files sekä suurille datamäärille tarkoitettu Azure Disks (Microsoft 2021a). Tietokantojen tallennustilana käytetään Azure diskejä, koska muut tallennustilatyypit eivät sovellu tietokannan tarvitsemille datamäärille.

File Share

Nginx:n ja Application Serverin kontit käyttävät Azure filesia, koska tallennettavia tiedostoja ei tarvitse lukea niin usein kuin tietokannan kohdalla. Tiedostoja käytetään konttien asetuksien tallentamiseen. Kontit käyttävät Azure filesia myös lokitietojen tallennuspaikkana. Yksi tärkeä syy Azure Filesin käyttöön on se, että tallennustilaa voidaan käyttää perinteisen verkkolevyn tavoin tai vaihtoehtoisesti Azuren verkkoselainnäkymän kautta, jolloin toiminnallisuus on rajoitetumpaa (Microsoft 2022a).

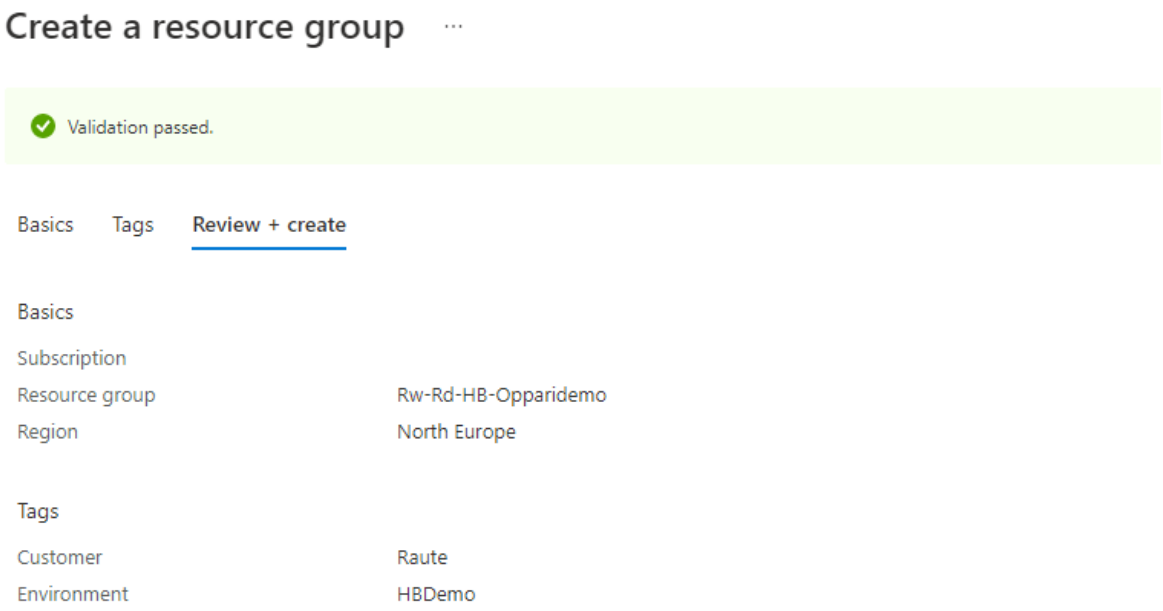
6 Testiympäristön rakentaminen

6.1 AKS-clusterin luominen

Ensimmäinen vaihe testiympäristön rakentamisessa on Kubernetes clusterin luominen, joka tehdään Azure portalin kautta. Testiympäristön luominen voidaan aloittaa myös rakentamalla container imaget. Cluster luodaan kuitenkin ensimmäisenä, koska se helpottaa varsinkin Application Serverin imagen rakentamisessa myöhemmin.

Resource Group

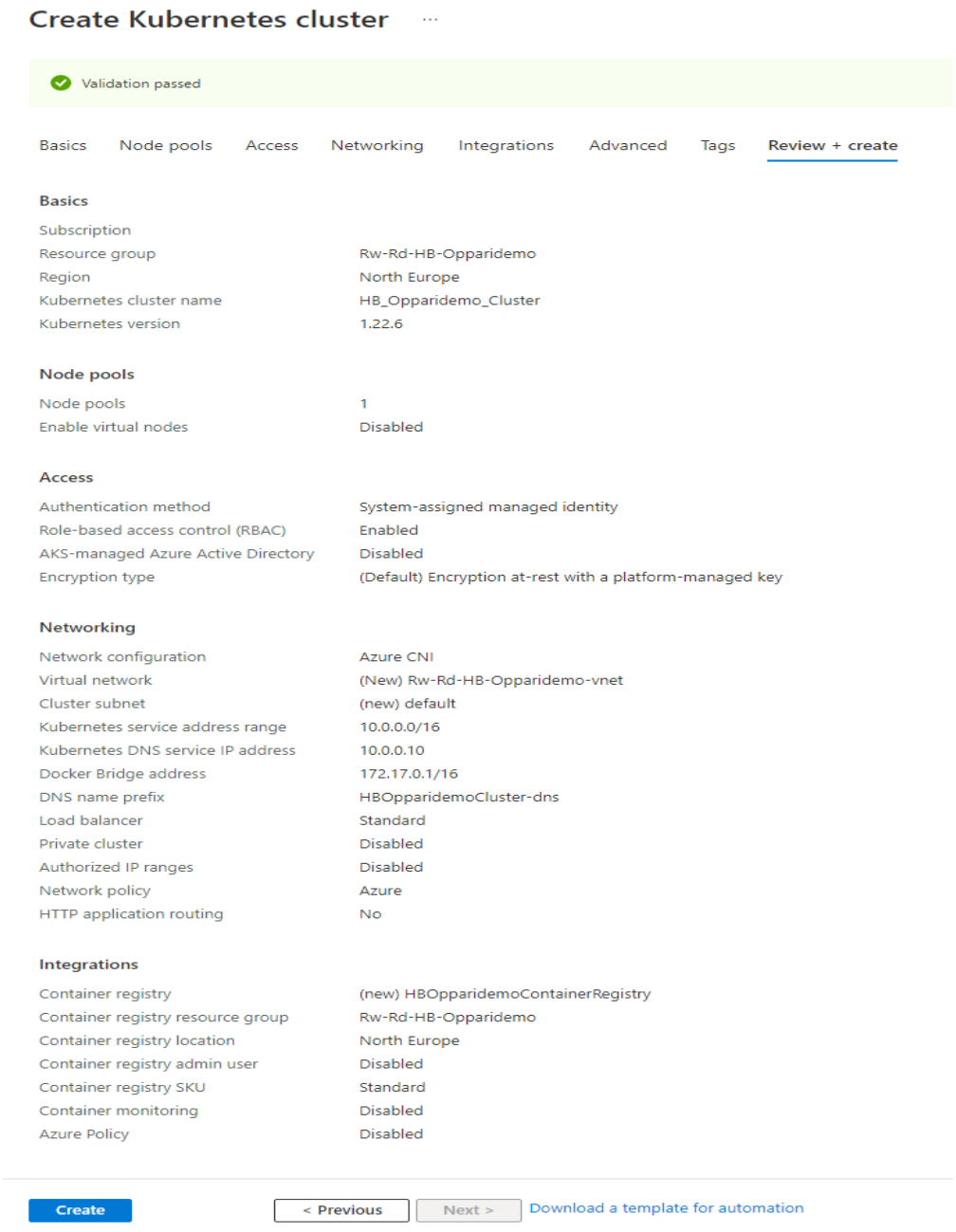
Aloitetaan luomalla Resource group, joka auttaa hallitsemaan Azureen varattavia resursseja. Kuvassa 3 nähdään resource groupin tekemiseen käytettävät asetukset. Tässä kohtaa on kiinnitettävä erityistä huomiota groupin nimeämiseen. Nimen tulee olla uniikki ja helposti tunnistettava, jolloin resurssien hallinta on helpompaa, kun resource groupeja on paljon. Toinen tärkeä asetus on Region eli maantieteellinen sijainti, joka määrittää mihin datakeskukseen groupin resurssit luodaan.



Kuva 3. Resource groupin asetukset

Kubernetes Service

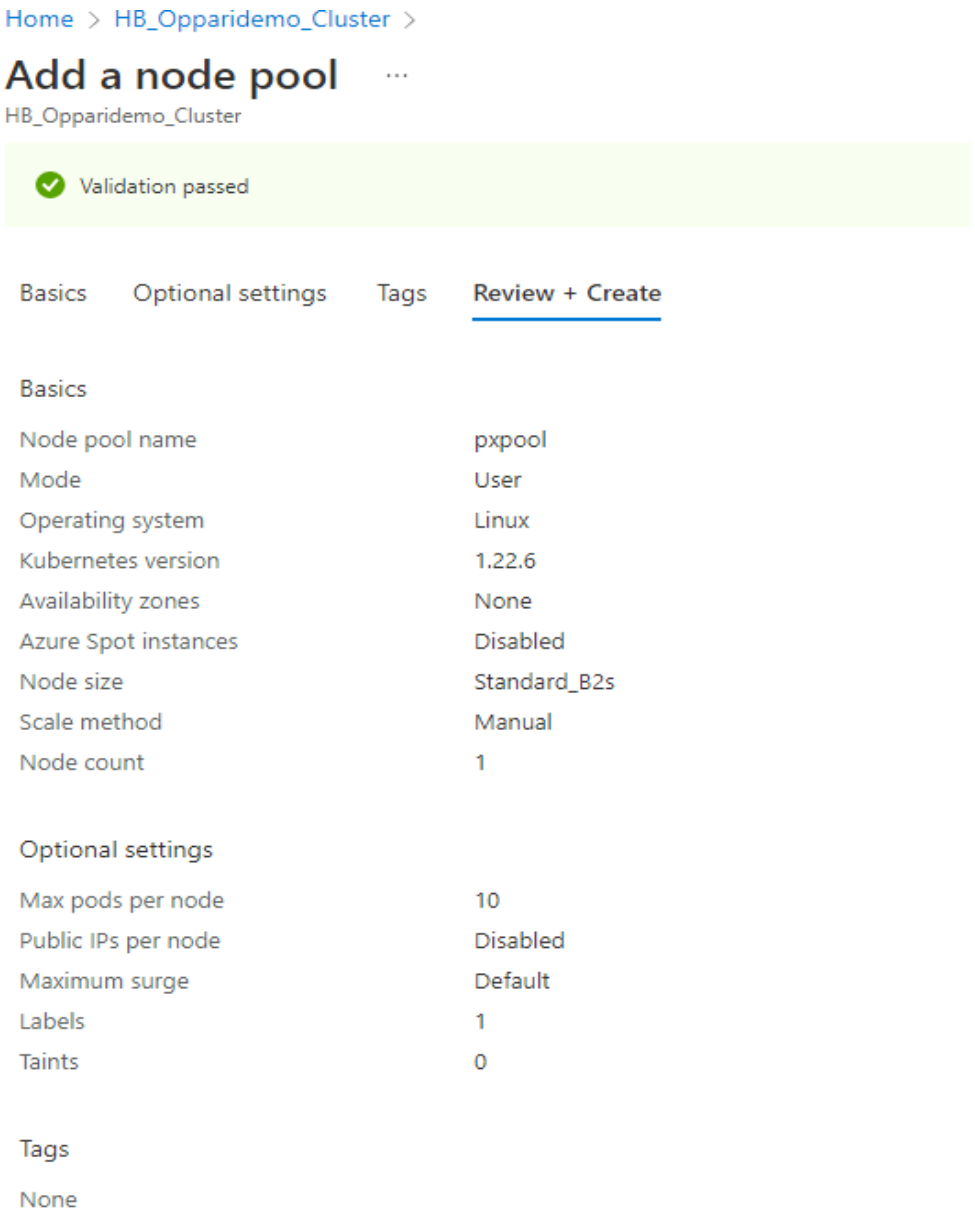
Juuri luotuun resource groupiin tehdään uusi Kubernetes Service -resurssi, jonka kohdalla pitää varmista, että käytettävä region on sama kuin resource groupilla. Samalla valitaan clusterin control planen käyttämän noden laskentateho. Cluster tarvitsee myös ACR-arkiston, josta container imaget haetaan, joten luodaan samalla uusi ACR-arkisto clusterin käyttöön. AKS-clusteria varten tarvitaan lisäksi Azuren virtuaaliverkko, joka tehdään myös clusterin luomisen yhteydessä. Kuvassa 4 nähdään clusterin tekemiseen käytettävät asetukset.



Kuva 4. Kubernetes clusterin asetukset

Node Poolit

Seuraavaksi määritetään clusterin node poolit. Käytettäviä kontteja tulee olemaan kolmea erilaista, joten erilaisille konteille tehdään omat node poolit. Jokaiselle eri poolille asetetaan omanlainen Label, joiden perusteella voidaan pakottaa kontit käyttämään haluttuja node pooleja. Lisäksi valitaan haluttu laskentateho, jota yksittäinen node saa käyttää. Kuvassa 5 on esimerkki node poolin luomisessa käytetyistä asetuksista.



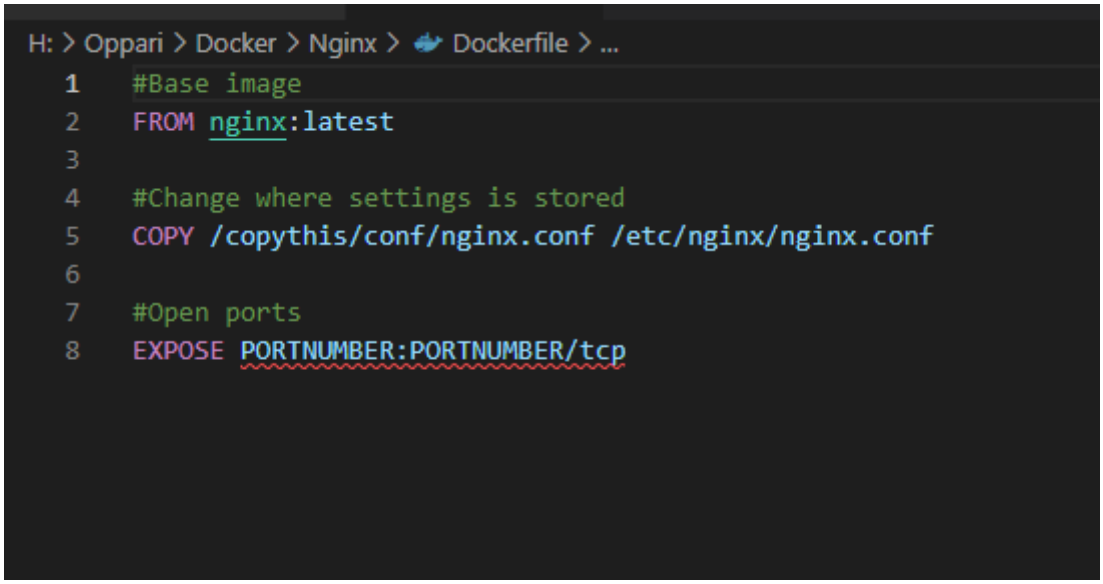
Kuva 5. Node poolin luomisessa käytetyt asetukset

6.2 Testiympäristön container imaget

Seuraavaksi luodaan käytettävistä ohjelmistoista container imaget, jotta Kubernetes voi luoda halutusta pohjasta uusia kontteja. Ensimmäisenä tehdään Nginx:n imagen rakentamiseen tarkoitettu dockerfile, koska sen tekeminen on helpompaa kuin Application Serverin. Tietokantapalvelimena käytettävää MariaDB:tä varten ei tehdä omaa dockerfileä, vaan container image haetaan suoraan Kubernetesin kautta.

Nginx Dockerfile

Dockerfile tehdään Dockerin (b) ohjeen mukaan. Ensin valitaan pohjana käytettävä base image, joka on tässä tapauksessa uusin Dockerin tarjoama valmis Nginx image. Container imagesta luodut kontit ovat keskenään identtisiä, joten niitä pitää pystyä muokkaamaan myöhemmin. Jokainen Nginx-kontti pitää eriyttää asiakaskohtaiseksi muuttamalla konttien käyttämiä asetuksia. Base imagen valinnan jälkeen dockerfileen merkitään, että yksi Nginx:n asetustiedostoista korvataan muokatulla tiedostolla, jolla vaihdetaan käytettävän asetustiedoston sijaintia. Uutena sijaintina käytetään jokaiselle kontille myöhemmin määritettävää Azure Filesia. Näin asiakaskohtaiset tiedot säilyvät, vaikka niitä käyttävä kontti tuhoutuisi. Dockerfilen lopuksi määritetään vielä portti, joka avataan kontin ulkopuolista verkkoliikennettä varten. Kuvassa 6 nähdään valmis Nginx:n dockerfile.



Kuva 6. Nginx:n container imagea varten tehty dockerfile (mukailtu Docker b)

Application Server Dockerfile

Seuraava dockerfile (Kuva 7) tehdään myös Dockerin (b) ohjeen mukaan. Application Server on Windows-pohjainen sovellus, joten base imagen pitää myös pohjautua Windows-käyttöjärjestelmään. Base imagen valinta on tehtävä tarkasti, sillä Kubernetesin noden käyttöjerjestelmän version pitää olla täysin sama kuin kontin. Jos versiot eivät täsmää, tuloksena on kontti, jota Kubernetes ei voi käynnistää. Seuraavaksi dockerfilessa käytetään PowerShell-komentoa, jolla asennetaan Application Serverin tarvitsemat Windowsin lisäominaisuudet, joita ei ole asennettu valmiiksi pohjaksi valitussa base imagessa.

Seuraavaksi Application Server asennetaan kontin sisälle käyttämällä MIS2.msi-asennuspakettia. Dockerfilen perusteella luodaan ensin Application serverin käyttämät kansiot, minkä jälkeen sovellus asennetaan kontin sisälle. Tämän jälkeen avataan sovelluksen käyttämät portit kuten Nginx:n kanssa.

Application Serverin konttien eriyttäminen toteutetaan hieman eri tavalla kuin Nginx:n tapauksessa. Application Server saa käyttöönsä Nginx:n tavoin kiinteää tallennustilaa asetusten säilömiseen. Jokainen Application Server käyttää asiakaskohtaisia TLS-sertifikaatteja, joiden asennus kontin sisälle pitää tehdä vasta, kun kontti on jo luotu. Tämä toteutetaan käyttämällä ENTRYPOINT-komentoa, jolloin kontin sisällä voidaan ajaa erilaisia komentoja kontin luomisen jälkeen (Docker b). Sertifikaattien asennus tehdään ulkoisella powershell skritpillä, jolloin voidaan esimerkiksi muuttaa asennettavia sertifikaatteja.

```
#Base image
FROM mcr.microsoft.com/windows/servercore:1809

#Windows features installation
RUN powershell "Set-Service -Name wuauserv -StartupType Manual; Install-WindowsFeature -Name Web-Asp-Net -Verbose;

#Appsrvr installation
RUN mkdir C:\DIRNAME
WORKDIR C:\\DIRNAME
COPY MIS2.msi C:\\DIRNAME\\MIS2.msi
RUN mkdir C:\\DIRNAME
RUN mkdir C:\\DIRNAME
RUN mkdir C:\\DIRNAME
RUN mkdir C:\\DIRNAME

#Ports
EXPOSE PORT:PORT/tcp
EXPOSE PORT:PORT/tcp
EXPOSE PORT:PORT/tcp
EXPOSE PORT:PORT/tcp

#Certificates
ENTRYPOINT powershell.exe "C:\\DIRNAME\\PSSCRIPTNAME.ps1"; ping -t localhost
```

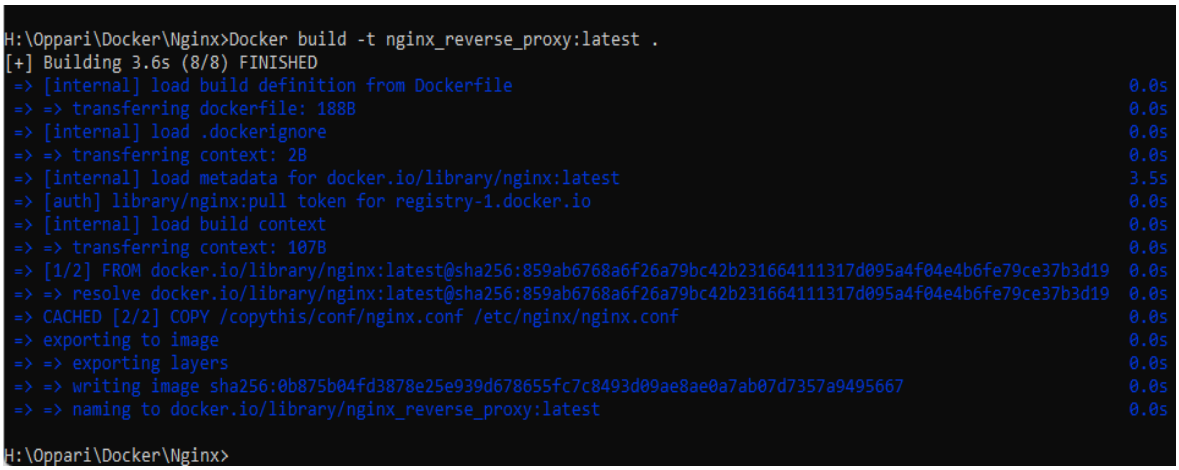
Kuva 7. Application Serverin dockerfile (mukailtu Docker b)

Docker Build

Testiympäristöä varten tarvittavat container imaget rakennetaan käyttämällä docker build -komentoa, jolla rakennetaan uusi container image dockerfilen ohjeiden perusteella. Dockerin (c) dokumentaation mukaan rakennettaville container imageille pitää antaa asianmukainen nimi ja uniikki tag, jota käytetään eri image versioiden tunnistamiseen.

Nginx:n container image

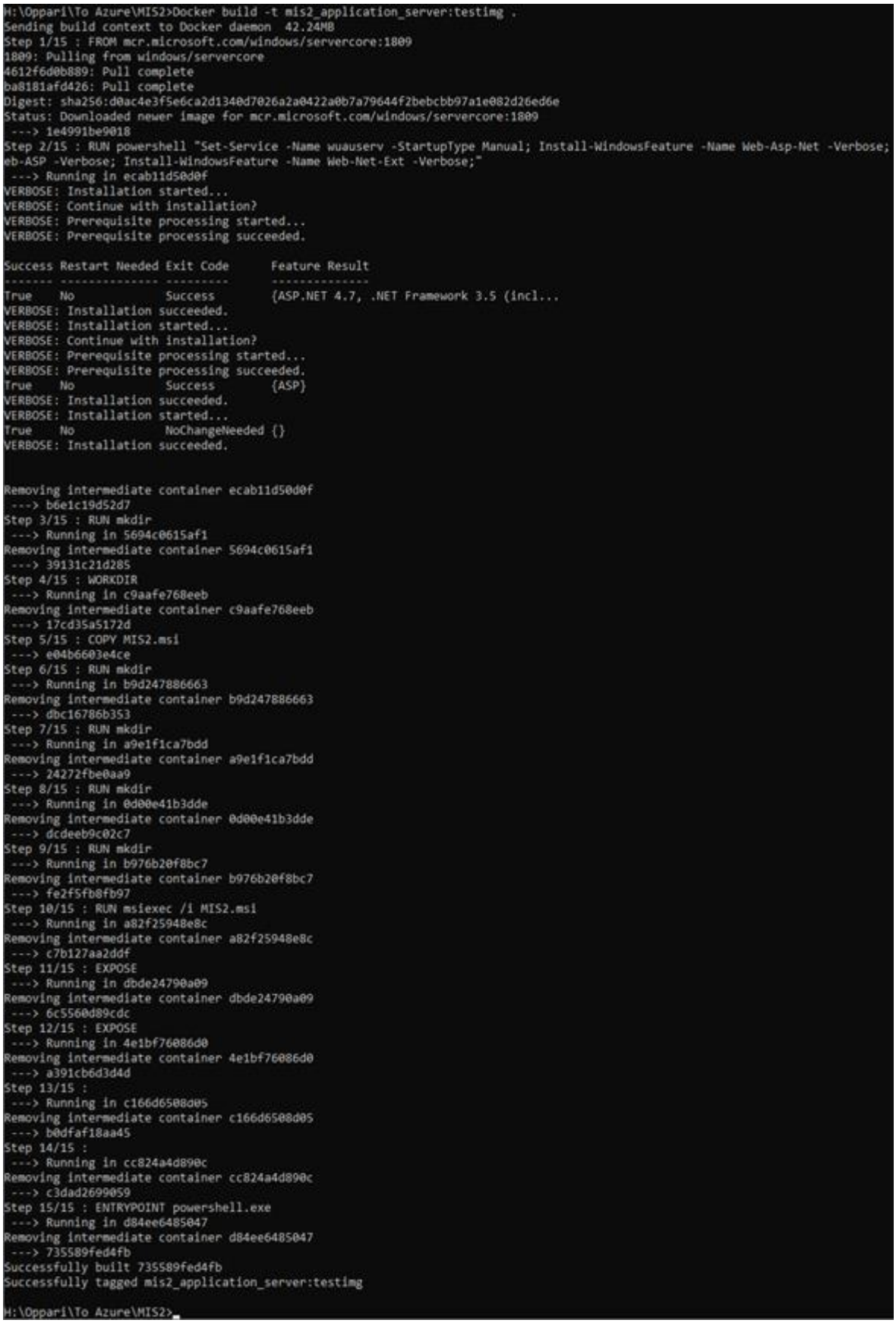
Ensimmäinen container image rakennetaan edellisessä luvussa tehdystä dockerfilesta. Dockerille annetaan komentoja Windowsin komentorivin kautta. Aluksi siirrytään samaan kansioon, johon Dockerfile on tallennettu. Sitten käytetään docker build -komentoa, minkä jälkeen Docker aloittaa container imagen rakentamisen. Kuvassa 8 nähdään Windowsin komentorivin näkymä sen jälkeen, kun container image on luotu onnistuneesti.



Kuva 8. Komentorivin näkymä onnistuneen docker build -komennon jälkeen

Application server container image

Seuraavaksi rakennetaan Application Serverin container image samalla tavalla kuin edellä oleva Nginx:n image. Application Serverin Dockerfile on paljon monimutkaisempi kuin Nginx:n, joten docker build -komennon suorittamiseen kuluu huomattavasti enemmän aikaa. Tämä on nähtävissä selkeästi, kun verrataan Application Serverin (Kuva 9) ja Nginx:n komentorivien näkymiä onnistuneen docker build -komennon jälkeen.



Kuva 9. Komentorivin näkymä Application Serverin container imagen luomisen jälkeen

ACR-arkisto

Aiemmin luodut container imaget siirretään seuraavaksi luvussa 6.1 tehtyyn uuteen ACR-arkistoon. Microsoftin (2022b) mukaan container imaget pitää kuitenkin ensin nimetä tietyllä tavalla ennen kuin ne voidaan puskea ACR:ään. Molempien sovellusten container imaget nimetään Microsoftin (2022b) dokumentaation mukaan käyttämällä docker tag -komentoa. Seuraavaksi kirjaudutaan ACR-arkistoon, minkä jälkeen container imaget pusketaan sinne käyttämällä docker push -komentoa. Kuvassa 10 on komentorivin ruutukaappaus, josta voidaan nähdä käytetyt komennot. Kuvasta huomataan myös, että container imageja ei lähetetä ACR-arkistoon kokonaisina tiedostoina, vaan kerroksittain. Tällöin lähetyksen keskeytyessä esimerkiksi Internet-yhteyden katkeamisen takia container imageja ei tarvitse lähettää kokonaan uudestaan. Riittää, että vain puuttuvat kerrokset lähetetään uudestaan. Kuvassa näkyvät imaget on nimetty uudelleen ennen lähettämistä.

```
C:\WINDOWS\system32>docker push hbopparidemocontainerregistry.azurecr.io/mis2_application_server:testimg
The push refers to repository [hbopparidemocontainerregistry.azurecr.io/mis2_application_server]
780c01499418: Layer already exists
722e831e9f92: Layer already exists
30a1b255cc14: Layer already exists
3a951d835069: Layer already exists
4055f136f8d4: Layer already exists
6f9f4b3cd5b5: Pushed
77abca223533: Layer already exists
b9d553a4b39b: Layer already exists
bc6a560d1c1b: Layer already exists
00a2da15b6db: Layer already exists
3b9c7dcdcedd: Layer already exists
3ab13667d280: Layer already exists
bd2cfa06dce6: Layer already exists
4d134aa4c0cf: Pushed
c6723851d2c1: Pushed
a7ba3db29ebb: Pushed
testimg: digest: sha256:09274958a35e5f23e5c39a61dd2e68f9c16e8f83f52ede599c12372bc9a005e0 size: 3680

C:\WINDOWS\system32>docker push hbopparidemocontainerregistry.azurecr.io/nginx_reverse_proxy:latest
The push refers to repository [hbopparidemocontainerregistry.azurecr.io/nginx_reverse_proxy]
a90bc4824104: Pushed
b6812e8d56d6: Pushed
7046505147d7: Pushed
c876aa251c80: Pushed
f5ab86d69014: Pushed
4b7fffa0f0a4: Pushed
9c1b6dd6c1e6: Pushed
latest: digest: sha256:1910dbeddbbb6453aa6e7483f3a71f3b4fad324017d3014f754c5b0c9caa239d size: 1777

C:\WINDOWS\system32>
```

Kuva 10. Ruutukaappaus komentorivin näkymästä docker push -komennon jälkeen

6.3 AKS-Clusterin asetukset

Kaikkien lukujen 6.1 ja 6.2 sisältämien vaiheiden jälkeen voidaan siirtyä Kunernetes clusterin konfigurointiin, joka tehdään yaml-tiedostoja käyttäen. Testiympäristön toimintaan tarvitaan viittä eri Kubernetesin ominaisuutta. Ensimmäinen on nimeltään deployment, jolla määritetään mitä kontteja Kubernetesin halutaan luovan ja kuinka monta samanlaista konttia luodaan. Samalla tehdään esimerkiksi käytettävään tallennustilaan liittyviä varauksia, nimetään kontteja ja tehdään muita haluttuja määrityksiä. (Kubernetes c.)

Deploymenteilla kerrotaan orkestraattorille konttien pyörittämiseen liittyvät määritykset, mutta deploymentit luovat vain toisistaan täysin eristyksissä olevia kontteja, jotka eivät voi kommunikoida mitenkään keskenään tai ulkoiseen verkkoon. Tästä syystä käytetään Kubernetesin servicejä, joilla mahdollistetaan konttien kanssa kommunikointi. Serviceille voidaan myös asettaa julkisia IP-osoitteita, jolloin clusterin konttien kanssa voidaan kommunikoida myös Internetin yli. (Kubernetes d.)

Edellisten määritysten jälkeen kontit kykenevät kommunikoimaan oman eristetyn ympäristönsä ulkopuolella, mutta tarvitsevat silti keinon tallentaa data pysyvästi. Tätä tarkoitusta varten voidaan Kubernetesin (e) mukaan käyttää storage class- ja persisten volume claim -toimintoja. Storage classilla määritetään millaista muistia Kubernetes saa varata clusterin käyttöön, ja persistent volume claimeja käytetään vapaana olevan tallennustilan varaamiseen konttien käyttöön. Clusterin käytettävissä oleva tallennustila voidaan määrittää myös manuaalisesti käyttämällä persistent volume -ominaisuutta, mutta silloin fyysinen tallennustila pitää olla jo valmiiksi olemassa. Storage classin avulla Kubernetes voi varata automaattisesti tallennustilaa tarpeiden mukaan ilman, että käyttäjän tarvitsee tehdä muita toimenpiteitä.

Viimeinen tärkeä ominaisuus on Namespace, jota käytetään eri ominaisuuksien ryhmittelyssä. Namespacet auttavat clusterin hallinnassa ja niitä voidaan hyödyntää esimerkiksi käyttöoikeuksien yhteydessä rajaamaan pääsyä clusterin ominaisuuksiin ja palveluihin (Kubernetes f).

Testiympäristön namespace

Testiympäristön kannalta namespacen määrittäminen ei ole pakollista, mutta ominaisuutta käytetään tulevaisuuden tarpeita ajatellen. Aluksi clusteriin luodaan uusi namespace, jota käytetään muiden ominaisuuksien määrityksissä. Namespace luodaan kuvassa 11 näkyvällä tavalla. Namespace on luotu Kubernetesin dokumentaation ohjeistuksen mukaan (Kubernetes f).

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: demonamespace
  labels:
    name: demospace1
```

Kuva 11. Uuden namespacen määritys (mukailtu Kubernetes f)

Persistent volume claim ja storage class

Seuraavaksi tehdään konttien kiinteään tallennustilaan liittyvät määritykset. Aloitetaan luomalla kaksi eri storage classia, koska Nginx ja Application Server käyttävät erityyppistä tallennustilaa kuin MariaDB. Storage classien määritykset tehdään Kubernetesin (e) dokumentaation mukaan. Kuvassa 12 nähdään testiympäristön storage class määritykset.

```yaml
1  apiVersion: storage.k8s.io/v1
2  kind: StorageClass
3  metadata:
4    name: testclass1
5  provisioner: file.csi.azure.com
6  parameters:
7    skuName: Premium_LRS
8  reclaimPolicy: Retain
9  volumeBindingMode: WaitForFirstConsumer
10 allowVolumeExpansion: true
11 ---
12 apiVersion: storage.k8s.io/v1
13 kind: StorageClass
14 metadata:
15   name: testclass2-db
16 provisioner: disk.csi.azure.com
17 parameters:
18   skuName: Premium_LRS
19 reclaimPolicy: Retain
20 volumeBindingMode: WaitForFirstConsumer
21 allowVolumeExpansion: true
```

Kuva 12. Testiympäristön storage classit (mukailtu Kubernetes e)

Storage classien jälkeen määritetään käytettäville konteille persistent storage claimit, joita käytetään yhdessä storage classien kanssa tarvittavan tallennustilan varaamiseksi. Storage class määrää minkälaista tallennustilaa käytetään ja persistent volume claim kuinka paljon tilaa varataan. Näiden kahden määrityksen perusteella Kubernetes varaa automaattisesti tarvittavan määrän tallennustilaa Azuresta. Kuvassa 13 nähdään testiympäristöön tehtävät persistent volume claimit. Kuvasta voidaan havaita mitä claimia yksittäiset kontit tulevat käyttämään metadatan alla olevasta nimestä. Kuvasta nähdään myös käytettävät storage classit ja varattavan tilan määrä sekä namespace määritys. Kuvassa näkyvät määritykset on tehty hyödyntäen Kubernetesin (e) dokumentaatiota.

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: nginx-reverse-proxy-pvc
  namespace: demonamespace
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: testclass1
  resources:
    requests:
      storage: 1Gi
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mariadb-pvc
  namespace: demonamespace
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: testclass2-db
  resources:
    requests:
      storage: 100Gi
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mis2-pvc
  namespace: demonamespace
spec:
  accessModes:
  - ReadWriteOnce
  storageClassName: testclass1
  resources:
    requests:
      storage: 4Gi
```

Kuva 13. Testiympäristön persistent volume claimien määritykset (mukailtu Kubernetes e)

Kubernetes servicet ja deploymentit

Testiympäristön toiminta vaatii vielä omat määritykset deplymenteille ja serviceille. Ensin tehdään service-määritykset. Jokaiselle kontille tehdään uniikki service, jonka avulla kontit voivat kommunikoida keskenään ja clusterin ulkopuolelle. Kuvassa 14 on esitetty serviceiden pohjaksi tehty asetustiedosto, jota muokataan asiakaskohtaiseksi, kun järjestelmään halutaan lisätä uusia MillSIGHTS-ympäristöjä. Asetustiedostossa on käytetty pohjana Kubernetesin (d) dokumentaatiota. Erityistä huomiota kiinnitetään Nginx:n service määrittelyyn, jossa ei käytetä staattista IP-määritystä, vaan service saa Kubernetesilta julkisen IP-osoitteen, jota käytetään esimerkiksi tiedonkeruun lähettämän datan vastaanottamiseen. Application Serverin servicen asetukset ovat konttikohtaisia asioita lukuun ottamatta samanlaiset kuin, kuvassa näkyvän MariaDB:n.

```
66  apiVersion: v1
67  kind: Service
68  metadata:
69    name: nginx-reverse-proxy-service
70    namespace: demonamespace
71  spec:
72    type: LoadBalancer
73    selector:
74      app: mis2-px
75    ports:
76    - name:
77      protocol: TCP
78      port: PORTNUMBER
79      targetPort: PORTNUMBER
80  ---
81  apiVersion: v1
82  kind: Service
83  metadata:
84    name: mariadb-service
85    namespace: demonamespace
86    annotations:
87      service.beta.kubernetes.io/azure-load-balancer-internal: "true"
88  spec:
89    type: LoadBalancer
90    loadBalancerIP: STATICSERVICEIP
91    selector:
92      app: maria
93    ports:
94    - name:
95      protocol: TCP
96      port: PORTNUMBER
97      targetPort: PORTNUMBER
```

Kuva 14. Testiympäristön service määritykset (mukailtu Kubernetes d)

Lopuksi tehdään vielä asetustiedosto (Kuva 15) järjestelmän deploymentteja varten Kubernetesin (c) dokumentaation mukaan. Kubernetes luo kontteja deploymentissa ilmoitettujen ohjeiden perusteella. Deploymentissa määritetään käytettävät container imaget, sovellusten käyttämät portit sekä luotavien konttien nimet. Aiemmin tehdyt persistent volume claimit liitetään oikeisiin kontteihin. MariaDB:n deployment asetukset (Kuva 16) ovat hieman erilaiset, koska kontin luomisessa käytetään Dockerin tarjoamaa container imagea ACR-arkiston sijaan. Tietokannan root- eli pääkäyttäjän salasana asetetaan suoraan deployment tiedostossa (Docker d). Tietoturvan kannalta tämä menettely ei ole turvallinen, joten salasana ilmoitetaan Kubernetesille jatkossa secret-ominaisuuden avulla (Kubernetes g). Secretin käyttö ei vaikuta testiympäristön toimintaan, joten sitä ei käytetä vielä tässä vaiheessa arkkitehtuuripäivitystä.

```
106  apiVersion: apps/v1
107  kind: Deployment
108  metadata:
109    name: mis2-proxy
110    namespace: demonamespace
111    labels:
112      app: mis2-px
113  spec:
114    replicas: 1
115    selector:
116      matchLabels:
117        app: mis2-px
118    template:
119      metadata:
120        labels:
121          app: mis2-px
122      spec:
123        nodeSelector:
124          Node: Proxy
125        containers:
126        - name: mis2-demo-proxy
127          image: hbopparidemocontainerregistry.azurecr.io/nginx_reverse_proxy:latest
128          ports:
129          - containerPort: PORTNUMBER
130          volumeMounts:
131          - name: nginx-volume
132            mountPath: etc/nginx/conf.d/
133        volumes:
134        - name: nginx-volume
135          persistentVolumeClaim:
136            claimName: nginx-reverse-proxy-pvc
```

Kuva 15. Osa testiympäristön deployment tiedostoa (mukailtu Kubernetes c)

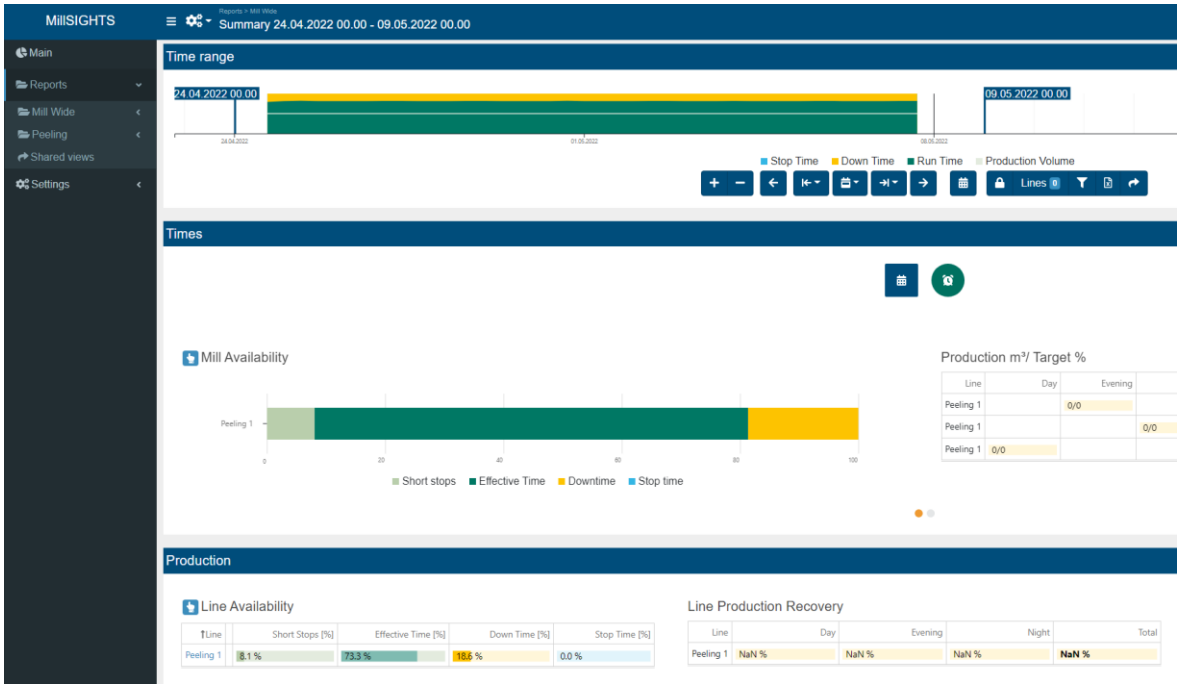
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mariadb
  namespace: demonamespace
  labels:
    app: maria
spec:
  replicas: 1
  selector:
    matchLabels:
      app: maria
  template:
    metadata:
      labels:
        app: maria
    spec:
      nodeSelector:
       Node: mariadb
      containers:
      - name: mariadb
        image: mariadb:10.4.24
        imagePullPolicy: "Always"
        env:
        - name: MYSQL_ROOT_PASSWORD
          value: THISISYOURPASSWORD
        ports:
        - containerPort: PORTNUMBER
        volumeMounts:
        - mountPath: /var/lib/mysql
          name: mariadb-data
      volumes:
      - name: mariadb-data
        persistentVolumeClaim:
          claimName: mariadb-pvc
```

Kuva 16. MariaDB:n deployment (mukailtu Kubernetes c)

7 Testiympäristön toiminnan testaus

Edellisen luvun ja sen alalukujen vaiheiden jälkeen tuloksena on yksi toimiva ohjelmistokonteilla toteutettu MillSIGHTS-ympäristö. Testiympäristön toiminnan testausta ja MillSIGHTS-ympäristöjen konfigurointia varten tarvitaan vielä VPN-yhteys Azuren virtuaaliverkkoon yhdistämistä varten. Yhteys muodostetaan Microsoftin (2021b) ohjeen mukaan. Azureen tehdään ensin VPN-yhdyskäytävä ja yhteyden muodostuksessa käytettävät sertifikaatit. Lopuksi asennetaan Azuresta ladattava VPN-client, jota käytetään yhteyden muodostuksessa.

Yhteyden muodostamisen jälkeen MillSIGHTS Master Configia voidaan käyttää järjestelmän konfigurointiin samalla tavalla kuin virtuaalikoneita käyttävässä ympäristössä. Ohjelmistokonteilla toteutetun ympäristön toimivuus varmistetaan lähetämällä järjestelmälle testidataa. Vastaanotettu data summataan, jonka jälkeen verkkoselaimen raporteissa nähdään muutoksia (Kuva 17).

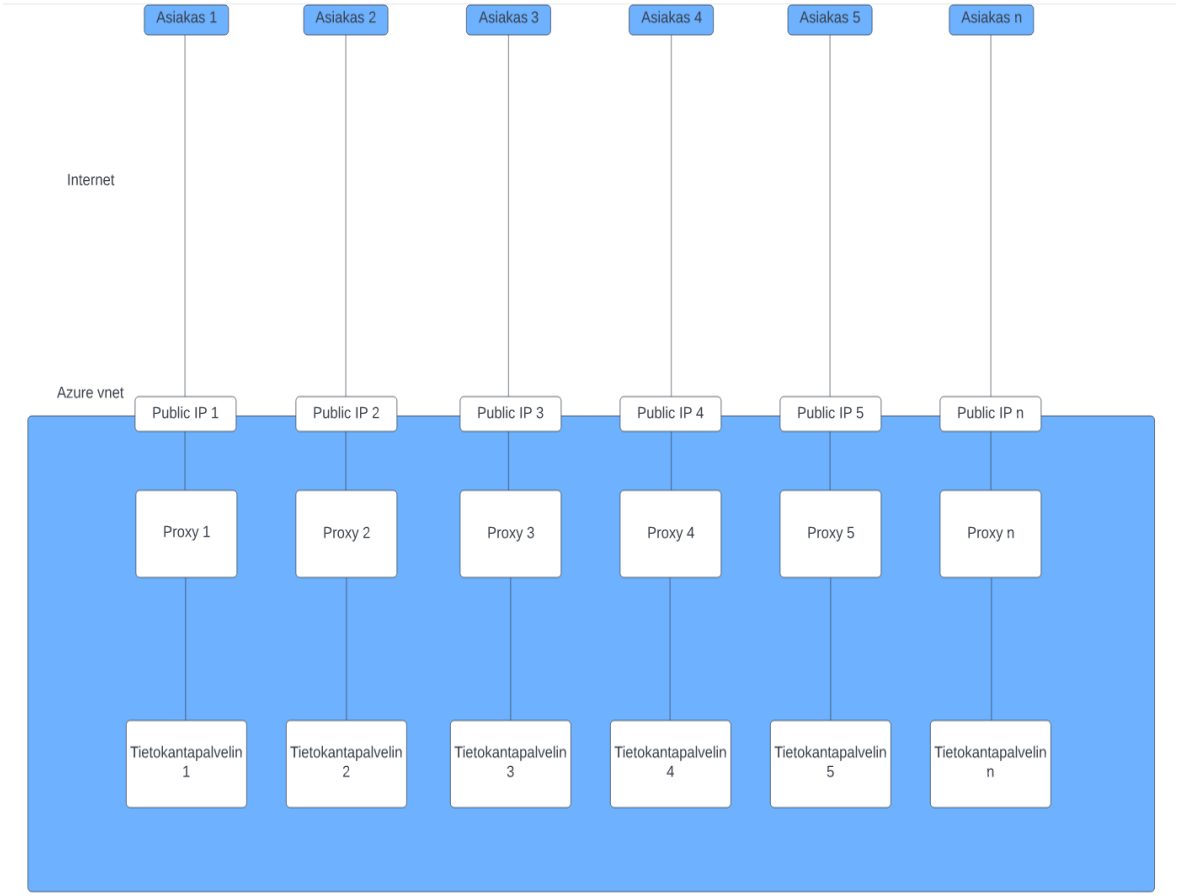


Kuva 17. Raporttinäkymä testidatan summaamisen jälkeen

8 Uuden ja vanhan arkkitehtuurin vertailu

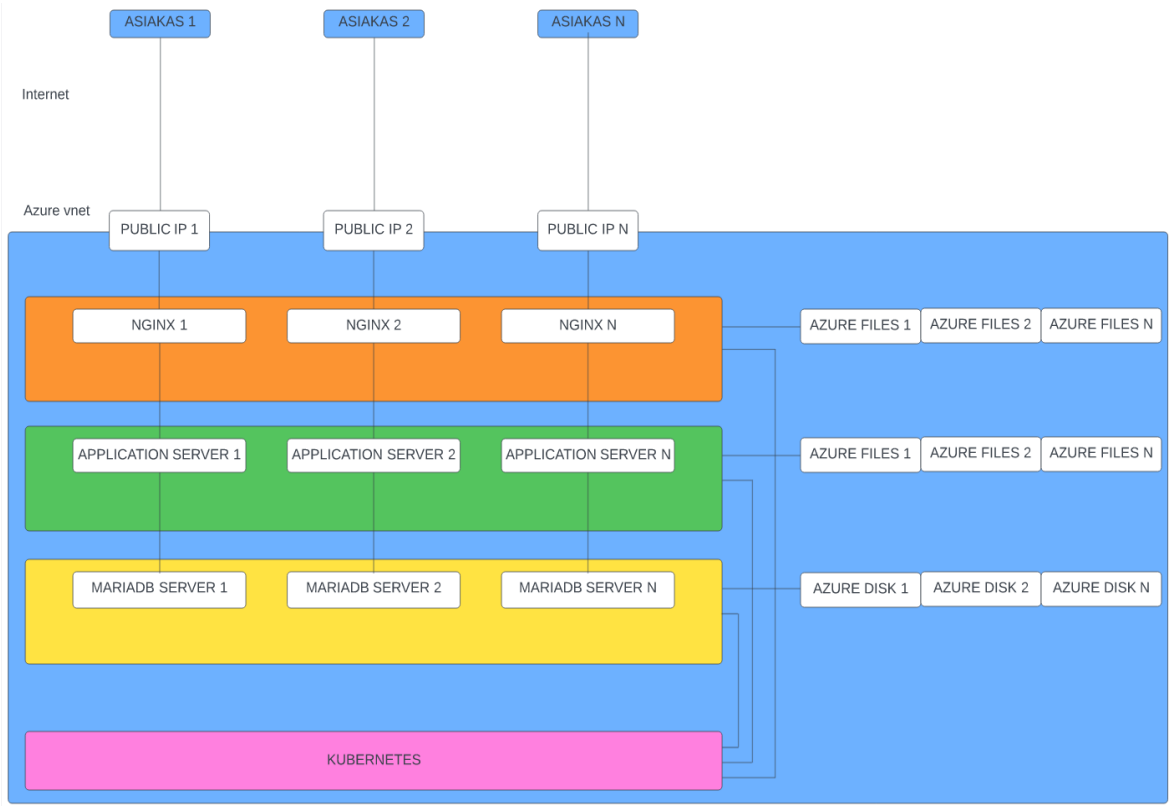
Topologia

Aloitetaan kahden eri arkkitehtuurin vertailu tarkastelemalla ensin yksinkertaistettuja topologiakarttoja. Kuviossa 5 esitetään, miltä vanha arkkitehtuuri pääpiirteissään näyttää. Kuviosta nähdään, että jokainen ympäristö on omana erillisenä kokonaisuutenaan. Tämän seurauksena koko järjestelmää koskevat muutokset pitää tehdä erikseen jokaiselle ympäristölle. Järjestelmää voidaan hallita rajatusti Azuren portalin kautta, mutta silloinkin jokaista virtuualikonetta täytyy käsitellä erikseen.



Kuvio 5. MillSIGHTSin pilviarkkitehtuurin topologia

Kuviossa 6 on esitetty, miltä arkkitehtuuri näyttää tässä opinnäytetyössä tehdyn uudistuksen jälkeen. Järjestelmän jokaista osaa voidaan hallita samasta paikasta keskitetysti. Kubernetes huolehtii eri osien toimivuudesta ja ylläpidosta. Topologiakartta on monimutkaisempi kuin vanhan arkkitehtuurin kohdalla, koska kartassa on eritelty järjestelmän osat tarkemmin. Värilliset pohjat kuvaavat eri node pooleja, joihin kontteja on sijoitettu.



Kuvio 6. MillSIGHTSin uudistettu topologia

Uudistetulla järjestelmällä on selkeitä etuja vanhaan nähden. Uutta ohjelmistokontteihin pohjautuvaa järjestelmää voidaan hallita keskitetysti Kubernetes-orkestraattorilla, jolloin voidaan esimerkiksi päivittää kaikkien käytössä olevien Application Server -konttien ohjelmistoversiot samaan aikaan. Arkkitehtuurin muutos ei myöskään vaikuta ulkoisiin järjestelmän osiin mitenkään, joten esimerkiksi tiedonkeruuseen ei tarvitse tehdä muutoksia. Yksi keskitetyn hallinnan huonoista puolista on kuitenkin se, että orkestraattorin control planen vaarantuessa hyökkääjä pääsee käsiksi kaikkiin clusterin eri MillSIGHTS-ympäristöihin. Hyviä tietoturvakäytäntöjä noudattamalla tämä ei tule olemaan ongelma, mutta siihen on hyvä kiinnittää huomiota jatkossa.

Yksi arkkitehtuurin uudistuksen tavoitteista oli nopeuttaa ja helpottaa uusien MillSIGHTS-ympäristöjen pystyttämistä. Vanhan järjestelmän luominen aloitetaan aina alusta, ja ohjelmistojen asentamisessa virtuaalikoneille kuluu useampi työpäivä. Varsinkin välityspalvelimen ohjelmistojen asennus vie paljon aikaa, minkä takia suurin osa uuden MillSIGHTS-ympäristön pystyttämisestä kuluu proxyn pystytykseen. Ohjelmistokonteilla tehdyn järjestelmän kanssa samanlaista ongelmaa ei ole, koska kontit luodaan container imageista, joihin tarvittavat sovellukset on jo asennettu valmiiksi. Näin uuden MillSIGHTS-ympäristön

rakentamiseen kuluu korkeintaan tunteja, vaikka koko järjestelmä pitäisi tehdä alusta asti uudestaan.

Uudistetussa arkkitehtuurissa voidaan myös hajauttaa laskentatehoa paremmin järjestelmän eri osien välillä. Kontit on jaettu ohjelmistojen mukaan eri node pooleihin, jolloin voidaan tarvittaessa parantaa tietyn järjestelmän osan suorituskykyä keskitetysti. Esimerkiksi yksittäiselle nodelle voidaan määrittää saman verran laskentatehoa kuin vanhassa arkkitehtuurissa on annettu koko virtuaalikoneelle. Yksi node voi toimia usean kontin isäntänä, jolloin noden laskentateho jaetaan kaikkien konttien kesken. Kubernetes voi asetuksista riippuen lisätä automaattisesti järjestelmään uusia nodeja, jolloin palveluiden suorituskyky pysyy mahdollisimman samanlaisena, jos laskentatehon tarve lisääntyy hetkellisesti.

9 Yhteenveto ja pohdinta

Tämän opinnäytetyön tavoitteena oli parantaa MillSIGHTS-järjestelmän pilviarkkitehtuuria, jotta uusien ympäristöjen rakentamisesta tulisi helpompaa ja nopeampaa. Yhtenä tavoitteena oli keskittää järjestelmän hallinta ylläpidon helpottamiseksi. Arkkitehtuuripäivityksessä tutkittiin erilaisia teknologioita, joita käytettiin testiympäristön luomisessa. Lopputuloksena syntynyt testiympäristö on toimiva, ja ohjelmistokonttien takia uusien asiakaskohtaisten ympäristöjen luominen on huomattavasti helpompaa ja nopeampaa kuin vanhaa arkkitehtuuria käytettäessä. Uudistettu järjestelmä hyödyntää myös käytössä olevaa laskentatehoa vanhaa järjestelmää tasaisemmin. Päivitettyä arkkitehtuuria käytettäessä uusien ympäristöjen luomiseksi vaaditaan vähemmän manuualista työtä, koska Kubernetesille annettavat asetukset syötetään tekstitiedostoilla. Kaikki clusteriin liittyvät määritykset voidaan antaa myös suoraan Azuren oman komentorivin kautta, jolloin ympäristöjen luominen on mahdollista automatisoida lähes kokonaan asiakaskohtaisten parametrien syöttämistä lukuun ottamatta.

Järjestelmän hallinta keskitettiin ottamalla käyttöön Kubernetes-orkestraattori, joka on vastuussa kaikkien eri ympäristöjen toiminnasta ja eheydestä. Orkestraattori pyrkii ylläpitämään käyttäjän tekemiä määrityksiä pyöritettävien konttien ja ominaisuuksien suhteen. Orkestraattori parantaa konteilla tarjottavien palveluiden saatavuutta, sillä esimerkiksi häiriötilanteissa tuhoutuneet kontit luodaan mahdollisimman nopeasti uudestaan.

Orkestraattorin käyttö helpottaa käytettävien ohjelmistojen versionhallinnassa. Koska kaikkia kontteja hallitaan yhdestä paikasta, koko järjestelmän laajuisia päivityksiä ja korjauksia voidaan tehdä keskitetysti. Kaikkien käytössä olevien ohjelmistojen päivitykset voidaan toteuttaa samaan aikaan, mutta yksittäisiä ympäristöjä koskevat muutokset ovat edelleen mahdollisia.

Kubernetesin käyttämien node poolien avulla voidaan varmistaa järjestelmän optimaalinen suorituskyky. Kontit jakavat keskenään nodelle määritellyn laskentatehon, joten resurssien varaamista voidaan skaalata nopeasti tarpeen mukaan. Haluttua resurssimäärää voidaan hallita joko täysin manuaalisesti tai Kubernetesin voidaan antaa skaalata nodeja automaattisesti ennalta määritettyjen arvojen mukaan. Yksittäisiä kontteja pystytään tarvittaessa pakottamaan täysin omille nodeille, jolloin voidaan varmistaa riittävä suorituskyky ja eristää ympäristöjä toisistaan vielä paremmin.

Tässä opinnäytetyössä tutkittuja ja käytettyjä teknologioita on mahdollista soveltaa muissakin yhteyksissä. Container imageja voidaan luoda halutuista ohjelmistoista, kun Dockerin toimintaan pääsee tutustumaan paremmin. Kubernetesia voidaan myös käyttää helposti

erilaisiin tarkoituksiin, mutta ohjelmiston toimintaperiaatteiden ymmärtäminen vaatii paljon aikaa ja harjoittelua.

Testiympäristöä on tarkoitus käyttää pohjana arkkitehtuurin jatkokehitykselle. Tässä opinnäytetyössä toteutetun testiympäristön tietoturvaan ei ole vielä kiinnitetty erityistä huomiota, koska työn tavoitteena oli etsiä ratkaisuja vanhan arkkitehtuurin puutteisiin. Esimerkiksi osa Master Configin ominaisuuksista ei vielä toimi oikein ohjelmistokontteja käyttävän järjestelmän kanssa. Tarvittavat muutokset ovat pieniä eikä niiden pitäisi vaikuttaa suuresti järjestelmän toimintaan. Uudistettua arkkitehtuuria on tarkoitus kehittää jatkossa pidemmälle, jolloin tietoturvaan ja järjestelmään liittyviin yksityiskohtiin kiinnitetään enemmän huomiota. Uudistetun järjestelmän käyttöönottoa harkitaan vasta, kun sen toimivuus on testattu laajasti ja varsinkin tietoturvaan liittyvät puutteet on korjattu.

Lähteet

Apache. The Number One HTTP Server On The Internet. Viitattu 28.4.2022. Saatavissa https://httpd.apache.org/

Cloudflare. What is an TLS (Transport Layer Security)? Viitattu 12.5.2022. Saatavissa https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/

Docker a. Use containers to Build, Share and Run your applications. Viitattu 1.5.2022. Saatavissa https://www.docker.com/resources/what-container/

Docker b. Dockerfile reference. Viitattu 1.5.2022. Saatavissa https://docs.docker.com/engine/reference/builder/

Docker c. Docker build. Viitattu 6.5.2022. Saatavissa https://docs.docker.com/engine/reference/commandline/build/

Docker d. mariadb. Viitattu 7.5.2022. Saatavissa https://hub.docker.com/_/mariadb

Elmubarak, S., Yousif, A. & Bashir, M. 2017. Performance based ranking model for cloud SaaS services. International Journal of Information Technology and Computer Science. Vol. 9 (1), 2017, 65–71. Viitattu 27.4.2022. Saatavissa https://www.mecs-press.org/ijitcs/ijitcs-v9-n1/IJITCS-V9-N1-8.pdf

Gokulakrishnan, J. & Bai, V. 2014. A survey report on VPN security & its technologies. Indian Journal of Computer Science and Engineering. Vol. 5 (4), 3–5. Viitattu 28.4.2022. Saatavissa http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.643.7609&rep=rep1&type=pdf

Heljanko, K. 2014. Virtuaalipilvet tietotekniikassa: mitä pilvipalvelu tarkoittaa? Aalto-yliopisto. Viitattu 27.4.2022. Saatavissa https://www.researchgate.net/profile/Keijo-Heljanko/publication/259828505_Virtuaalipilvet_tietotekniikassa_mita_pilvipalvelu_tarkoittaa_Presentation_at_Finnish_Technology_Days_2014/links/02e7e52e0f0e984fcd000000/Virtuaalipilvet-tietotekniikassa-mitae-pilvipalvelu-tarkoittaa-Presentation-at-Finnish-Technology-Days-2014.pdf

Kubernetes a. What is Kubernetes? Viitattu 3.5.2022. Saatavissa https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Kubernetes b. Kubernetes Components. Viitattu 3.5.2022. Saatavissa https://kubernetes.io/docs/concepts/overview/components/

Kubernetes c. Deployments. Viitattu 7.5.2022. Saatavissa https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Kubernetes d. Services, Load Balancing, and Networking. Viitattu 7.5.2022. Saatavissa https://kubernetes.io/docs/concepts/services-networking/

Kubernetes e. Dynamic Volume Provisioning. Viitattu 7.5.2022. Saatavissa https://kubernetes.io/docs/concepts/storage/dynamic-provisioning/

Kubernetes f. Share a Cluster with Namespaces. Viitattu 7.5.2022. Saatavissa https://kubernetes.io/docs/tasks/administer-cluster/namespaces/

Kubernetes g. Secrets. Viitattu 7.5.2022. Saatavissa https://kubernetes.io/docs/concepts/configuration/secret/

MariaDB. MariaDB Server: The open source relational database. Viitattu 28.4.2022. Saatavissa https://mariadb.org/

Microsoft. 2021a. Best practices for storage and backups in Azure Kubernetes Service (AKS). Viitattu 5.5.2022. Saatavissa https://docs.microsoft.com/en-us/azure/aks/operator-best-practices-storage

Microsoft. 2021b. Configure a Point-to-Site connection by using certificate authentication (classic). Viitattu 7.5.2022. Saatavissa https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-point-to-site-classic-azure-portal

Microsoft. 2022a. What is Azure Files? Viitattu 5.5.2022. Saatavissa https://docs.microsoft.com/en-us/azure/storage/files/storage-files-introduction

Microsoft. 2022b. Push your first image to your Azure container registry using the Docker CLI. Viitattu 6.5.2022. Saatavissa https://docs.microsoft.com/en-us/azure/container-registry/container-registry-get-started-docker-cli?tabs=azure-cli

Microsoft a. What you'll love about SQL Server 2017. Viitattu 28.4.2022. Saatavissa https://www.microsoft.com/en-us/sql-server/sql-server-2017

Microsoft b. Azure Kubernetes Service (AKS). Viitattu 30.4.2022. Saatavissa https://azure.microsoft.com/en-us/services/kubernetes-service/#overview

Microsoft c. Azure Container Registry. Viitattu 5.5.2022. Saatavissa https://azure.microsoft.com/en-us/services/container-registry/#features

Mohammed, C. & Zeebaree, S. 2021. Sufficient comparison among cloud computing services: IaaS, PaaS and SaaS: A review. International Journal of Science and Business.

Vol. 5 (2), 17–30. Viitattu 27.4.2022. Saatavissa https://ideas.repec.org/a/aif/journl/v5y2021i2p17-30.html

Nginx. Welcome to NGINX Wiki! Viitattu 28.4.2022. Saatavissa https://www.nginx.com/resources/wiki/

Raute a. Rauten ratkaisut. Viitattu 26.4.2022. Saatavissa https://www.raute.fi/sijoittajat/raute-sijoituskohteena/rauten-ratkaisut/

Raute b. MillSIGHTS. Viitattu 27.4.2022. Saatavissa https://www.raute.com/software/millsights/

Rani, D. & Ranjan, R. 2014. A comparative study of SaaS, PaaS and IaaS in cloud computing. International Journal of Advanced Research in Computer Science and Software Engineering. Vol. 4 (6), 458–461. Viitattu 27.4.2022. Saatavissa https://www.academia.edu/29455291/A_Comparative_Study_of_SaaS_PaaS_and_IaaS_in_Cloud_Computing

Red Hat. 2019. What is orchestration? Viitattu 30.4.2022. Saatavissa https://www.redhat.com/en/topics/automation/what-is-orchestration

Söderlund A. 2019. Kontit ja virtuaalipalvelimet pilvipalvelussa. Jyväskylän yliopisto. Viitattu 30.4.2022. Saatavissa https://jyx.jyu.fi/bitstream/handle/123456789/65884/1/URN%3ANBN%3Afi%3Ajyu-201910154462.pdf

Vailshery, L. 2022. Container orchestration tool used by organizations most frequently as of 2019. Statista. Viitattu 30.4.2022. Saatavissa https://www.statista.com/statistics/588827/worldwide-container-technology-orchestration-tool-use/