Petteri Mäki

TAUSTAJÄRJESTELMÄ PELIPALVELIMIEN ISÄNNÖIMISEEN PILVIPALVELUNA

Opinnäytetyö

Tekniikan ammattikorkeakoulututkinto

Tieto- ja viestintätekniikan koulutus

2022

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Insinööri (AMK)
Tekijä/Tekijät	Petteri Mäki
Työn nimi	Taustajärjestelmä pelipalvelimien isännöimiseen pilvipalveluna
Toimeksiantaja	Xamk Proto Studio
Vuosi	2022
Sivut	74 sivua
Työn ohjaaja(t)	Niina Mässeli

TIIVISTELMÄ

Tämän opinnäytetyön tavoitteena on kehittää taustajärjestelmä peliin Superraketti. Taustajärjestelmän on tarkoitus isännöidä pelipalvelimia, joille liittymällä pelaajat voivat pelata pelin monipelimuotoa. Pelaajat liittyvät pelipalvelimille internet-yhteyden välityksellä. Taustajärjestelmä kehitetään kokonaisuudessaan pilvipalveluympäristöön.

Työn toimeksiantajana toimii Xamk Proto Studio. Xamk Proto Studio ottaa vastaan opiskelijoiden peli- ja kehitysideoita, joita ryhdytään kehittämään eteenpäin. Superraketti-peli on valittu mukaan kehitykseen. Xamk Proto Studio toimii Kaakkois-Suomen ammattikorkeakoulun alueella.

Opinnäytetyön teoriaosuus toteutettiin tutkivalla lähestymistavalla. Teoriaosuus aloitettiin tutkimalla taustajärjestelmien keskeiset määritelmät ja toimintatavat. Niiden perusteella ryhdyttiin tutkimaan työkaluja ja menetelmiä, joista voisi olla apua toiminnallisessa kehitysosuudessa.

Opinnäytetyön toiminnallisessa kehitystyöosassa luotiin tavoitteiden sekä tutkittujen työkalujen ja rakenteiden perusteella suunnitelma, jonka pohjalta taustajärjestelmä kehitettiin pelille.

Kehitystyössä pelipalvelimesta luotiin ohjelmistokonttiversio Dockerin avulla. Ohjelmistokonteissa ylläpidetään pelipalvelimia. Ohjelmistokonttien hallintaan valittiin työkaluksi Kubernetes-järjestelmä, jonka avulla pelipalvelimien isännöinti voidaan suoraviivaistaa. Agones-kirjaston avulla helpotettiin pelipalvelimien hallintaa Kubernetes-ympäristössä. Koko järjestelmä kehitettiin Google Cloud -pilvipalveluympäristöön.

Opinnäytetyön tuloksena valmistui taustajärjestelmä pelille. Pilvipalveluympäristöön rakennettiin taustajärjestelmä, joka isännöi pelipalvelimia. Taustajärjestelmän kehityksessä käytettiin työkaluja toivotulla tavalla. Pelaajat pystyvät liittymään pelipalvelimille onnistuneesti. Kehitetty järjestelmä on tärkeä osa peliä, koska se mahdollistaa pelin monipelimuodon pelaamisen internet-yhteyden välityksellä. Tulosten perusteella muodostettiin jatkokehitysideoita, joiden avulla järjestelmän toimintaa voidaan laajentaa ja parantaa entisestään.

Asiasanat: taustajärjestelmä, pelipalvelin, pilvipalvelu, Kubernetes

South-Eastern Finland University of Applied Sciences

Degree	Bachelor of Engineering
Author (authors)	Petteri Mäki
Thesis title	Backend system for hosting game servers as a cloud service
Commissioned by	Xamk Proto Studio
Time	May 2022
Pages	74 pages
Supervisor	Niina Mässeli

ABSTRACT

The objective of this thesis was to develop a backend system for the game Superraketti. The main task of the backend system was to host dedicated game servers for the game's multiplayer mode. Players connect to the dedicated game servers through the internet.

The thesis is divided into theoretical and practical parts. The theoretical part consists of examined structures and the perks of backend systems. From there, the theorical part advances to examining tools that are deemed to be useful in the practical part of the thesis.

The practical part of the thesis consists of planning and developing the backend system for the game. A plan for the backend system was created based on the objective of the thesis and the observations in the theoretical part. After that, the backend system is developed based on the plan.

The development work started off by creating a software container version of the game server using Docker. Dedicated game servers were maintained in software containers. The Kubernetes system was used as a tool for managing software containers. The Agones library relieved the management of game servers in the Kubernetes environment. The entire system was developed in the Google Cloud environment.

The result of the thesis was a working background system for the game. The backend system hosts dedicated game servers for the players. Players were able to successfully join the game servers hosted in the backend system. The developed backend system was a success. The developed system is a very important part of the game service because it allows players to play the game through an internet connection. Based on the results, further development ideas were formed. The operations of the backend system can be expanded to achieve an even broader benefit for the game.

Keywords: backend system, dedicated game servers, cloud service, Kubernetes

SISÄLLYS

1 JOHDANTO 6

2 KEHITTÄMISTYÖN TIETOPERUSTA 7

2.1 Tutkimuksen lähtökohdat 7

2.2 Eettisyys ja luotettavuus 8

3 TEORIAOSUUS 9

3.1 Taustajärjestelmä 9

3.2 Rakenne 11

3.3 Virtualisointi 14

3.3.1 Ohjelmistokontit 15

3.3.2 Virtuaalikoneet 16

3.3.3 Ohjelmistokonttien ja virtuaalikoneiden erot 17

3.4 Orkestrointi 18

3.5 Pilvipalvelut 20

4 TOTEUTUKSEN SUUNNITELMA 23

4.1 Tavoitteet 23

4.2 Työkalut 25

4.3 Kehitystyön rakenne 26

5 TOTEUTUS 29

5.1 Suunnitelman testaus 29

5.1.1 Pelipalvelimen testaus 30

5.1.2 Pelipalvelinohjelmistokontti 35

5.1.3 Kubernetes 40

5.2 Taustajärjestelmän kehitys pilvipalveluun 45

5.2.1 Pilvipalvelun käyttöönotto 46

5.2.2 Agones-kirjasto pelipalvelimissa 48

5.2.3 Pilvipalvelun rekisteri 51

5.2.4 Pelipalvelinten Kubernetes-asettelu 52

5.2.5 Pelipalvelimien käynnistys 54
5.2.6 Testaaminen 54
6 TULOKSET 56
6.1 Kehitystyön kulku 57
6.2 Työkalujen käyttö kehitystyössä 59
6.3 Kehitysideat 61
6.4 Johtopäätökset 62
7 POHDINTA 62
7.1 Merkitys ja prosessi 62
7.2 Jatkokehitysmenetelmät 63
7.2.1 Ideaalinen kehitys 63
7.2.2 Vaihtoehtoinen kehitys 65
7.3 Teoria ja työkalut 66
7.4 Agones ja Gcloud 67
LÄHTEET 69
KUVALUETTELO

1 JOHDANTO

Tässä opinnäytetyössä käsitellään taustajärjestelmän suunnittelemista sekä kehittämistä peliin nimeltä Superraketti. Tavoitteena on kehittää pelin monipelimuotoa varten taustajärjestelmä, jonka avulla pelaajat voivat pelata peliä verkkoyhteyden välityksellä. Opinnäytetyössä tullaan kehittämään pelipalvelinjärjestelmä, joka toteutetaan pilvipalvelun ympäristöön hyödyntäen niiden tarjoamia työkaluja. Työssä esitellään taustajärjestelmäkehityksen keskeisiä työkaluja, termejä, menetelmiä sekä ratkaisutapoja, joita hyödynnetään opinnäytetyön toiminnallisen osuuden toteutuksessa. Opinnäytetyön toiminnalliseen osuuteen sisältyy Superraketti-pelin taustajärjestelmän suunnittelu ja kehitys. Opinnäytetyö on tyypiltään kehitystyö.

Superraketti on pelityypiltään kaksiulotteinen pelaaja vastaan pelaaja monipeli. Peliä pelataan avaruusteemaisessa kentässä, jossa pelaajat lentävät avaruusaluksillaan. Pelaajat valitsevat itselleen avaruusaluksen sekä aseen, jolla pelaaja pyrkii tuhoamaan muiden pelaajien alukset. Pelin voittaa pelaaja, joka tuhoaa eniten muita pelaajia. Pelaajien on tarkoitus liittyä pelipalvelimille internetin välityksellä. Peliä voidaan pelata tietokoneella Windows- ja Linux-alustoilla. Opinnäytetyössä kehitetään pelille taustajärjestelmä, jossa ylläpidetään pelipalvelimia. Taustajärjestelmän toiminta pyritään kehittämään mahdollisimman automaattiseksi. Tässä opinnäytetyössä keskitytään vain pelipalvelimien tarjoamiseen, joten pelipalvelimen toimintalogiikka jää ulkopuolelle.

Opinnäytetyön aihe tuli tarpeesta kehittää pelille taustajärjestelmä. Opinnäytetyön kehitystyö on välttämätön osuus pelin toimintaa ajatellen, koska se mahdollistaa pelin pelaamisen maailmanlaajuisesti verkkoyhteyden välityksellä. Lisäksi opinnäytetyön tutkimus antaa toimeksiantajalle ja muille osallisille tietoa taustajärjestelmän suunnittelusta, rakentamisesta, soveltuvuudesta sekä käytettävyydestä.

Työn toimeksiantajana toimii Xamk Proto Studios. Xamk Proto Studio toimii Xamkin, eli Kaakkois-Suomen ammattikorkeakoulun alueella. Xamk Proto Stu-

dio ottaa vastaan opiskelijoiden peli- ja kehitysideoita, joita mahdollisesti lähdetään kehittämään eteenpäin julkaisuun saakka. Superraketti-peli on valittu mukaan kehitykseen.

2 KEHITTÄMISTYÖN TIETOPERUSTA

Tämän opinnäytetyön toiminnallisen kehitystyön osuuden tavoitteena on suunnitella ja kehittää taustajärjestelmä peliin Superraketti. Opinnäytetyössä myös tutustutaan taustajärjestelmän yleiseen rakenteeseen sekä teknologioihin, joita voidaan hyödyntää taustajärjestelmän kehityksessä.

Opinnäytetyön kehitystyön tavoitteissa otetaan huomioon työhön varatut resurssit, joihin lukeutuvat työkalut, arvioitu työmäärä sekä aikataulutus. Lisäksi pelin arvioidun pelaajamäärän perusteella määritellään alustavasti kehitettävän palvelun ylläpitoon kuluvat resurssit.

Pelipalvelimiin on lisätty niiden vaatima asiakas–palvelin-pelilogiikka etukäteen, joten tässä työssä ei käsitellä pelipalvelimien sisäistä toimintaa. Opinnäytetyössä siis keskitytään vain pelipalvelimista koostuvan taustajärjestelmäpalvelun kehitykseen. Opinnäytetyössä kuitenkin muokataan pelipalvelimen logiikkaa, jos taustajärjestelmä sitä vaatii.

2.1 Tutkimuksen lähtökohdat

Tutkimusongelma tarkoittaa keskeistä ongelmaa, johon opinnäytetyöllä pyritään vastaamaan. Tässä opinnäytetyössä tutkimusongelma on taustajärjestelmäpalvelun kehittäminen pelipalvelimille. Työn tutkimuskysymykset johdetaan tutkimusongelmasta. Opinnäytetyössä vastataan tutkimuskysymyksiin:

- Mikä on taustajärjestelmä ja miten se luodaan peleille?
- Miten taustajärjestelmäpalvelu toteutetaan pilvipalveluun?
- Mitä työkaluja ja menetelmiä voidaan hyödyntää taustajärjestelmän kehityksessä?

Tämä opinnäytetyö on tyypiltään toiminnallinen kehittämistyö. Opinnäytetyön kehitystyönä toimii taustajärjestelmän kehittäminen Superraketti-peliin. Työ jakautuu teoreettiseen ja toiminnalliseen osaan. Teoreettisessa osassa tutkitaan

taustajärjestelmiä aiheen ymmärtämiseksi. Toiminnallisissa osuuksissa suunnitellaan ja toteutetaan kehitystyö teorian pohjalta.

Kehitystyössä aineistonkeruumenetelmänä toimii suora havainnointi. Kehitystyössä kerätään kuva- ja tekstihavaintomateriaalia yleisluontoisesti. Kerätty teksti- ja kuvamateriaali konvertoidaan tekstimuotoon selkeän dokumentoinnin luomiseksi. Kuvia käytetään tekstidokumentoinnin tukena. Dokumentoinnin pohjalta voidaan määritellä vastaako toiminnallinen kehitystyö tutkimusongelmiin ja -kysymyksiin.

Opinnäytetyön teoriaosuuden lähteinä käytetään verkkodokumentteja. Verkkodokumenttien avulla pyritään hahmottamaan idea, jolla tutkimusongelma voidaan ratkaista. Verkkodokumenttien avulla kehitystyön tavoite segmentoidaan asiasisältöihin, jotka on helpompi ratkaista.

Tämä opinnäytetyö on jaettu johdanto-, tietoperusta-, teoria-, toteutuksen suunnittelu-, toteutus-, tulokset - sekä pohdintalukuihin. Opinnäytetyö aloitetaan johdantoluvulla, jossa työ esitellään lyhyesti. Johdantoa seuraa opinnäytetyön tietoperustaluku. Tietoperustassa käydään läpi tapauskohtaiset tiedot, joiden perusteella muodostuu opinnäytetyön viitekehys.

Tietoperustaa seuraa teoriaosuusluku, jossa tutkitaan taustajärjestelmäpalveluiden yleisiä rakenteita, työkaluja ja menetelmiä. Teoriaosuuden jälkeen ryhdytään suunnittelemaan kehitystyön taustajärjestelmän toteutusta. Suunnitteluluvussa suunnitellaan taustajärjestelmän toteutus viitekehyksen perusteella hyödyntäen teoriaosuuden sisältöä. Suunnitelmaa seuraa toteutusluku, jossa rakennetaan taustajärjestelmä suunnitelman pohjalta. Tulokset-luvussa tutkitaan kehitystyön tuloksia, joita verrataan suunnitelmaan ja tavoitteisiin. Viimeisenä käydään läpi pohdintaluku, jossa pohditaan työn tuloksen onnistuneisuutta, merkitystä ja mahdollisia kehitysideoita.

2.2 Eettisyys ja luotettavuus

Opinnäytetyön tavoitteena on luoda luotettavasti suunniteltu ja kehitetty taustajärjestelmä. Taustajärjestelmän on tarkoitus tulla viralliseen käyttöön, joka

tukee pyrkimystä luotettavuuteen ja eettisyyteen. Eettisyyttä tukee kehitystyön tekijän halu kehittää osaamista kehitystyön alalla.

Työssä tutkitaan vastaavanlaisten taustajärjestelmäpalveluiden rakenteita sekä työkaluja asiantuntemuksen parantamiseksi. Asiantuntemuksen avulla voidaan kehittää luotettava idea opinnäytetyön tavoitteeseen pääsemiseksi.

Kehitystyön tulos validoidaan tulokset-luvussa suhteuttamalla tutkimuskysymykset kehitystyön prosessiin. Kehitystyön havainto- ja kuvamateriaali kerätään samanaikaisesti kehitystyön edetessä luotettavan dokumentoinnin luomiseksi. Suhteuttaminen tapahtuu vertailemalla kehitystyön havaintomateriaalia tutkimuskysymyksiin.

Opinnäytetyössä käytettävät lähteet pyritään valikoimaan työhön pätevyyden perusteella. Samoja aiheita tutkitaan useista eri lähteistä niiden autenttisuuden sekä pätevyyden määrittelemiseksi. Lähteiden valinnassa suositaan uudempia lähteitä.

3 TEORIAOSUUS

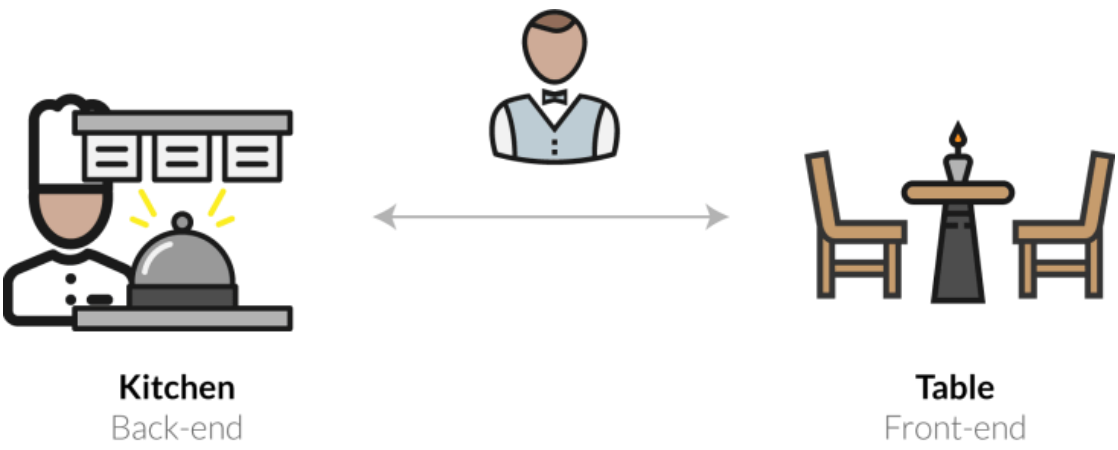
Tässä luvussa esitellään taustajärjestelmissä usein käytettyjä työkaluja, menetelmiä sekä termistöä. Tämän luvun sisältöön kuuluvat pääasiassa asiat, joita voidaan hyödyntää kehitystyön suunnitelmassa ja toteutuksessa. Teoriaosuuden termistö selitetään mahdollisimman yksinkertaisesti ja ymmärrettävästi. Teoriaosiossa esiteltäviä asioita ei selitetä kokonaisuudessaan, vaan niistä mainitaan ainoastaan asiat, jotka koetaan tärkeiksi ja hyödyllisiksi kehitystyössä. Esimerkiksi jos työkalulla on kymmeniä eri toimintoja, ainoastaan kehitystyön kannalta merkityksellisimmät toiminnot esitellään.

3.1 Taustajärjestelmä

Ohjelmistokehityksessä tavoitteena on yleensä luoda ohjelmistotuote, joka palvelee asiakkaita. Palvelu voi esimerkiksi olla verkkosivu verkkokaupalle. Ohjelmistokehityksessä palvelun rakenne jaetaan usein käyttöliittymään ja taustajärjestelmään. Käyttöliittymä on rajapinta, jonka avulla asiakas kommunikoi taustajärjestelmän kanssa. Käyttöliittymänä voi esimerkiksi toimia verkkokaupan verkkosivu tai videopelin ikkuna. Taustajärjestelmä taas koostuu

palvelun tarjoamiseksi suoritetusta ohjelmistologiikasta. Sillä tarkoitetaan taustalla toimivaa ohjelmistosysteemiä ja -logiikkaa, jonka avulla palvelu tarjotaan asiakkaalle. Taustajärjestelmän toiminta on tärkeä osa palvelun rakennetta. Siihen kuuluu muun muassa ohjelmistopalvelimien ajama palvelulogiikka, prosessit ja palvelun komponenttien muodostamat sidokset. Palvelin eli serveri tarkoittaa tietokonetta tai ohjelmistoa, joka palvelee käyttäjää tai ohjelmistoa. (Posey 2021.)

Yleinen tapa havainnollistaa taustajärjestelmän eli ”backendin” ja käyttöliittymän eli ”frontendin” toimintaa on niin sanotulla ”ravintola”-analogialla (Kuva 1). Analogiassa esitetään ravintolatilannetta, jossa asiakas tilaa ruokaa. Asiakas valitsee aterian menusta, jonka tarjoilija toimittaa kokille. Kokki valmistaa aterian, jonka tarjoilija palauttaa asiakkaalle. Asiakkaan käyttämä menu toimii tilanteen käyttöliittymänä eli rajapintana, jonka kanssa asiakas pystyy kommunikoimaan. Asiakas valitsee aterian menusta lukemalla ateriavaihtoehtoja. Asiakas ei sen sijaan kykene kommunikoimaan suoraan kokin kanssa, vaan viestin pitää kulkea tarjoilijan kautta. Tarjoilija toimii välittäjänä asiakkaan ja kokin välillä. Kokin tehtävänä on valmistaa ateria, joka palautetaan asiakkaalle. Kokki toimii analogian taustajärjestelmäpalveluna, joka suorittaa asiakkaan pyytämän palvelun toiminnon. Tarjoilija palauttaa kokin suorittaman toiminnon asiakkaalle. Kokki toimii taustajärjestelmäympäristössä eli ravintolan keittiössä, kun taas asiakas toimii asiakasympäristössä eli ravintolan salissa. Menu on siis analogian käyttöliittymä, jonka avulla asiakas valitsee palvelun, ja kokki taustajärjestelmä, joka suorittaa itse palvelun logiikan, eli aterian. (Codeacademy 2019.; Kononenko 2018.)



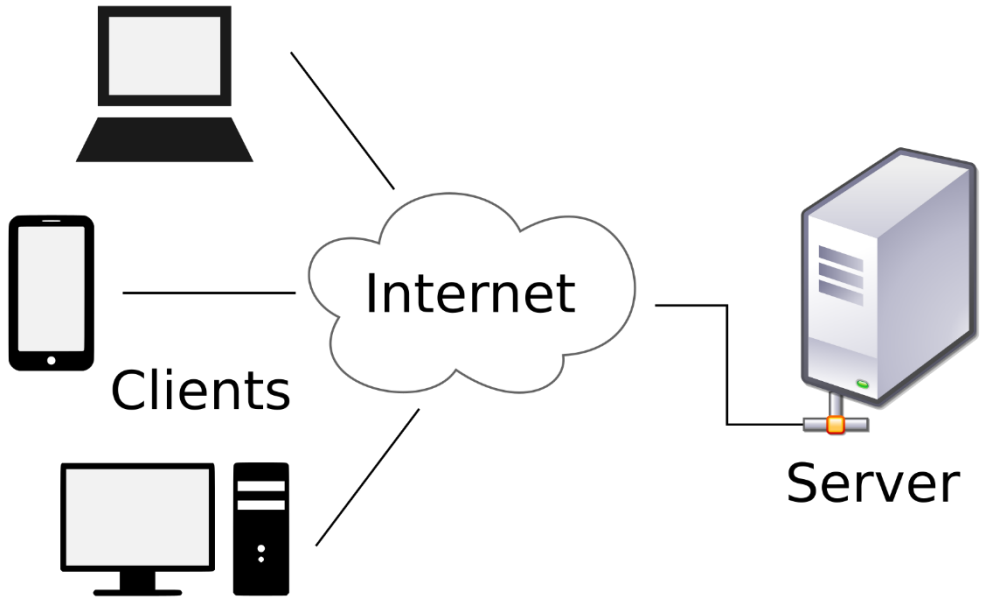
Kuva 1. Ravinto-analogian rakenne

Taustajärjestelmä koostuu yleensä useasta eri rakenneosasta. Rakenneosia voivat esimerkiksi olla palvelimen logiikka sekä tietokannan hallinta. Tässä opinnäytetyössä keskitytään vain pelipalvelimien tarjonnan logiikkaan. Jatkossa opinnäytetyössä käytetään termiä ”ohjelmistopalvelu”, kun halutaan viitata taustajärjestelmän osa-alueeseen, jonka tehtävänä on tarjota ohjelmiston luoma palvelu asiakkaille.

3.2 Rakenne

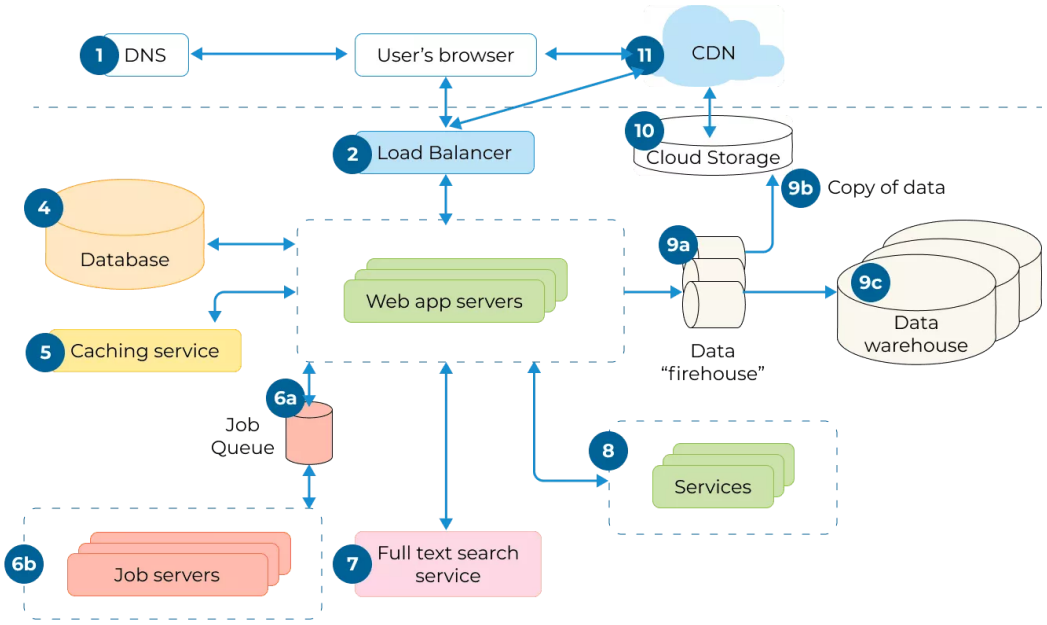
Taustajärjestelmän kehityksessä on tärkeää määritellä ohjelmistopalvelun rakenne. Ohjelmistopalvelun rakenteella tarkoitetaan rakennetta, jossa määritellään ohjelmistopalvelun komponentit ja niiden välinen sidosverkosto. Komponentit ovat palvelun tarjoamiseen tarvittavia rakenneosia, esimerkiksi palvelimia ja tietokantoja. Yksinkertaisuudessaan ohjelmistopalvelun rakenne siis kuvaa kehitettävän ohjelmistopalvelun loogista rakennetta. Ohjelmistopalvelun rakenne voidaan visualisoida toimintakaavioiden ja diagrammien avulla. (Castsoftware s.a.)

Yleinen internetin välityksellä toimiva palvelu, esimerkiksi verkkosivu, saattaa käyttää perinteistä palvelin–asiakas-rakennetta. Palvelin–asiakas-rakenteessa asiakas ottaa tietokonelaitteellaan yhteyden palveluun, joka palauttaa halutun toiminnon asiakkaalle (Kuva 2). Asiakas voi esimerkiksi tehdä palvelupyynnön verkkoselaimen välityksellä verkkokauppapalveluun, joka palauttaa verkkokauppasivun asiakkaan selaimeen. Palvelun käytön aikana asiakas tekee usein useita eri pyyntöjä palvelimelle. Edellisen esimerkin asiakas voi ostaa verkkokaupan sivuilla tuotteita. Ostohetkellä asiakas ottaa yhteyden palvelimeen, joka tallentaa ostotapahtuman tiedot palvelun tietokantaan. (CIO Wiki s.a.)



Kuva 2. Yksinkertainen asiakas–palvelin-rakenne

Ohjelmistopalvelun rakenne on usein kuvattu monimuotoisemmin kuin edellisessä esimerkissä (Kuva 2). Ohjelmistopalvelurakenteen kuvauksessa voidaan esimerkiksi havainnollistaa palvelun sisäisiä komponentteja ja osa-alueita. Laajamittaisessa suunnitelmassa on kuvattu kaikki palvelun tarvitsemat komponentit, niiden tehtävät sekä verkostot. Ohjelmistopalvelurakenteen kuvauksen hahmottelu aloitetaan jo palvelun suunnittelun yhteydessä. Tarkasti suunniteltu ja kuvattu rakenne auttaa ohjelmistopalvelun kehityksessä ja helpottaa mahdollisten ongelmakohtien löytämisessä. Alapuolen Kuva 3 kuvataan tarkasti suunnitellun ohjelmistopalvelun rakennetta. Kuvassa hahmotellaan palvelun rakenteen eri komponentit ja niiden väliset sidokset.



Kuva 3. Esimerkki tarkasti kuvatusta ohjelmistopalvelun rakenteesta. Kuvassa yläpuolen käyttöliittymä ja alapuolen taustajärjestelmä on jaettu vaakasuuntaisella katkoviivalla. Komponentit on kuvattu erivärisillä laatikoilla ja lieriöillä. Nuoliviivat kuvaavat komponenttien välisiä sidoksia.

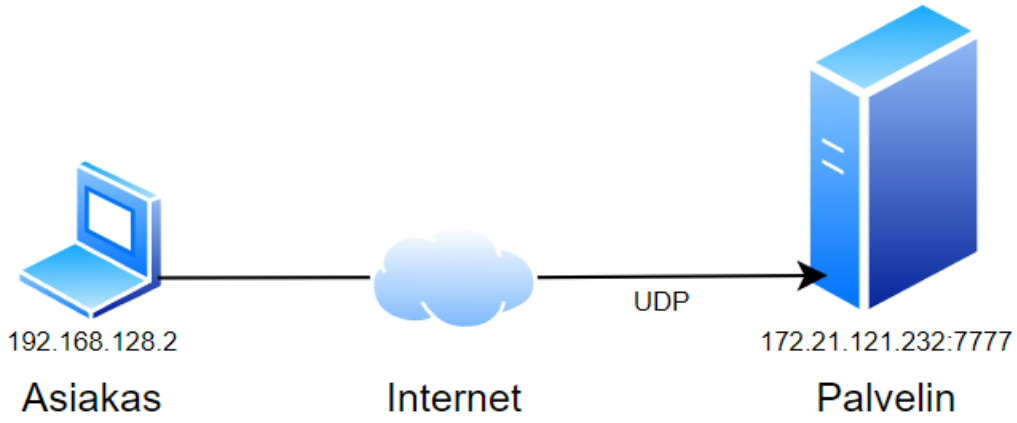
Laitteiden välinen kommunikointi

Ohjelmistopalvelun sidosverkot muodostuvat ohjelmistorajapintojen tai fyysisten osoitteiden avulla laitteiden välille. Jokaisella laitteella on oma osoitteensa, jota kutsutaan IP-osoitteeksi. IP-osoite tarkoittaa tietokonelaitteen uniikkia osoitetta verkossa. Sitä tarvitaan, jos laitteen kanssa halutaan kommunikoida ulkopuolelta. IP-osoitetta tarvitaan esimerkiksi silloin, kun asiakas haluaa viestiä palvelimen kanssa. Asiakas ottaa yhteyden IP-osoitteella palvelimeen, joka palauttaa vastauksen asiakkaan laitteelle. IP-osoite versiolla 4 (IPv4) koostuu numeroista, jotka eritellään pisteillä. IP-osoitteet on merkitty Kuva 4 laitteiden alapuolelle. (Jena 2022.)

Tietoliikenneprotokolla (TCP/IP) tarkoittaa tietokoneviestinnässä määriteltyä viestintämenetelmää. IP-osoite on vain yksi osa tietoliikenneprotokollan rakennetta. Tietoliikenneprotokolla muodostuu useista kerroksista, jotka mahdollistavat kommunikoinnin laitteiden välillä. Tietoliikenneprotokollan avulla laitteet voivat kommunikoida keskenään internetissä tai verkossa. IPv4 on osa verkkokerrosta. Verkkokerroksen päällä on kuljetuskerros, jossa määritellään liikennöinnissä käytettävä protokolla. Kuljetuskerroksen mahdollistavat ohjelmistojen kommunikoinnin keskenään verkon välityksellä. Kuljetuskerroksen protokollia ovat esimerkiksi TCP ja UDP. Eri protokollilla on eri ominaisuudet

ja tilanteet, joissa niitä käytetään. Kuljetuskerroksen protokolla on esitetty Kuva 4 nuolena palvelimelle. (IBM 2022.)

Portti on tietokoneiden välisen viestinnän alku- ja loppupiste. Portti määrittelee tietokonelaitteen sijainnin, jossa jokin prosessi toimii. Käyttäjä pystyy kommunikoimaan prosessin kanssa yhdistymällä porttiin. Portti ilmaistaan numeromuodossa, joka voi esimerkiksi olla Kuva 4 mukaisesti 7777. (Jena 2022.)



Kuva 4. Asiakkaan palvelupyyntö IP-osoitteilla. Palvelimen IP-osoitteen perässä eritelty portti ":"-merkillä. Kuljetuskerroksena toimii UDP-protokolla.

Tietokoneiden välistä viestintää voidaan havainnollistaa junaliikenneanalogian avulla. Kuvitellaan junarata, jonka matkalla on useita asemia. Jokainen matkalla oleva juna-asema on tietokone. Juna-asemien IP-osoitteet toimivat asemien osoitteina ja portit liikenteelle varattuina laitureina. Kuljetuskerroksen protokolla toimii rahdin tyyppinä. Laiturin ja rahdin täytyy olla yhteensopivat: tavararahtia ei voida jättää matkustajalaiturille. Junan loppuosoitteeksi annetaan juna-aseman IP-osoite, laituriksi portin numero ja rahdiksi palvelimen kuljetusprotokolla. Jos asema, laituri ja rahti sopivat yhteen, tietokoneiden välinen viestintä onnistuu.

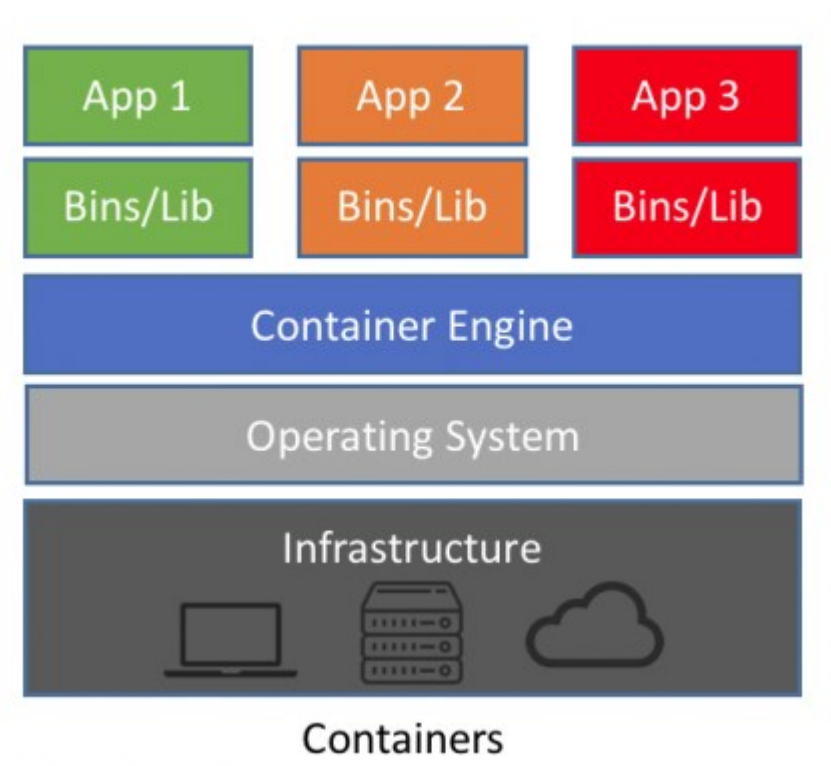
3.3 Virtualisointi

Virtualisointi tarkoittaa tietokonelaitteen, tietokoneresurssin tai sovelluksen erittelyä fyysisestä laitteesta. Halutusta laitteesta, esimerkiksi tietokoneesta,

luodaan virtuaalinen versio, joka toimii isäntälaitteen ”päällä”. Isäntälaite tarkoittaa tietokonetta, joka ylläpitää virtualisoitua laitetta. Virtualisoidut laitteet kuluttavat isäntälaitteen resursseja. Samalla isäntälaitteella voidaan isännöidä useita virtuaalilaitteita samaan aikaan. Virtualisoinnin avulla isäntälaitteen resurssit voidaan jakaa tehokkaasti yhden tai useamman virtualisoidun laitteen kesken. Virtualisoidut laitteet toimivat lähes samalla tavalla kuin niiden fyysiset versiot. Lisäksi virtuaalikoneilla voidaan ajaa käyttöliittymiä ja ohjelmistoja, joiden ajaminen ei ole mahdollista isäntälaitteella. Virtualisoituja laitteita ovat esimerkiksi virtuaalikoneet (virtual machine) ja ohjelmistokontit (container), joista kerrotaan seuraavaksi enemmän. (opensource.com s.a.; Red Hat 2018.)

3.3.1 Ohjelmistokontit

Ohjelmistokonttiteknologialla tarkoitetaan ohjelmistokonttien hyödyntämistä ohjelmistokehityksessä. Teknologiassa haluttu ohjelmisto pakataan standardoituun konttiin. Kontteihin pakataan ohjelmiston osa tai kokonaisuus mukaan lukien tarvittavat sidoskirjastot, eli kaikki sisältöohjelmiston ajamiseen tarvittavat komponentit. Ohjelmistokontissa voidaan esimerkiksi ajaa verkkokauppapalvelinta tai pelipalvelinta. Ohjelmistokontit ovat siis mukautettuja ympäristöjä, joissa niiden sisältöohjelmisto ajetaan. Kontit toimivat erillisesti virtualisoituna isäntälaitteen päällä, joten niitä on helppo siirtää kehitysympäristöstä toiseen (Kuva 5). Ohjelmistokontit eivät myöskään yleensä vaadi muutoksia isäntäympäristön muuttuessa. Konttien käytön avulla tietokoneen resurssit voidaan jakaa tehokkaasti ohjelmiston osien kesken. Kontteja voidaan käynnistää, muuttaa sekä poistaa käytöstä tarvittaessa. Kontit ovat siis toimintatavoiltaan samantapaisia, kuin logistiikassa käytetyt kuljetuskontit. Docker on yleisesti käytetty ohjelmistokonttityökalu. (Section.io 2021.; Wallenius 2019.)

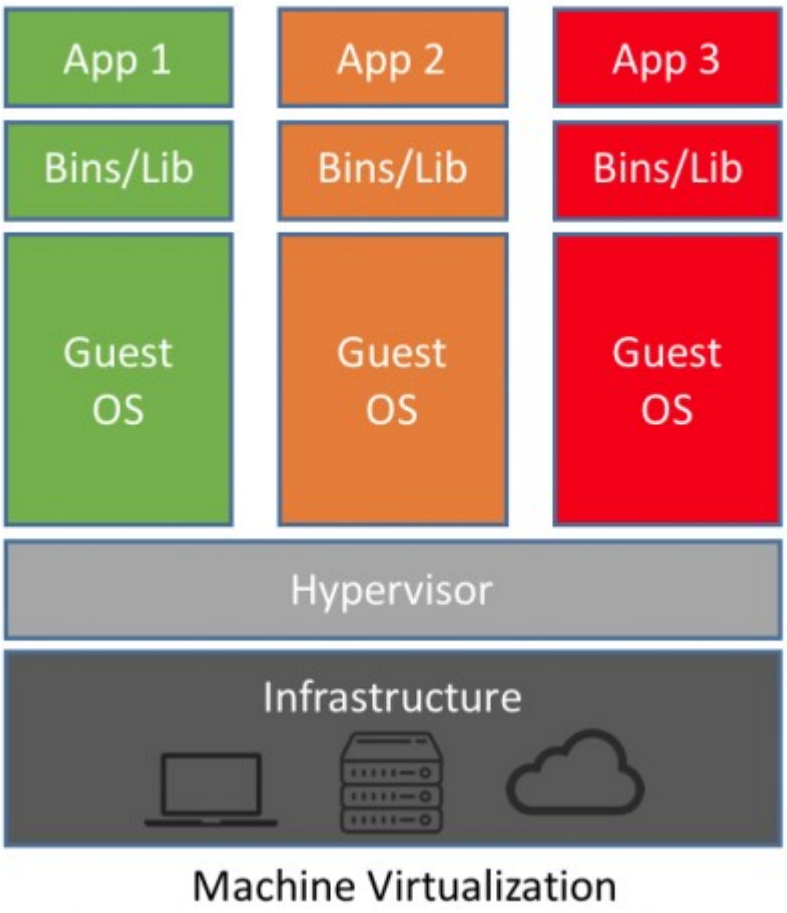


Kuva 5. Konttijärjestelmän rakennekaavio

Kuva 5 kuvaa ohjelmistokontin rakennetta. Pohjakerroksessa on isäntälaiteympäristö ”infrastructure”, jonka päällä toimii käyttöjärjestelmä ”operating system”. Seuraavana rakenteessa tulee konttimoottori-kerros ”container engine”, joka hallitsee kontteja ja käsittelee käyttäjän antamat konttisyötteet. Kontit sijaitsevat erillään toisistaan ylimmissä riveissä. Kontit koostuvat niiden ohjelmistoista ”App” ja sidoskirjastoista ”Bins/Lib”.

3.3.2 Virtuaalikoneet

Virtuaalikone eli ”virtual machine” tarkoittaa emuloitua virtuaalitietokonejärjestelmää. Virtuaalikoneet ovat toiminnaltaan samankaltaisia kuin ohjelmistokontit. Virtuaalikone on ikään kuin erillinen tietokone isäntälaitteessa. Virtuaalikoneet rakennetaan isäntäjärjestelmän päälle, jossa niitä voi olla useita. Niille voidaan varata haluttu määrä isäntälaitteen resursseja. Jokaisella virtuaalikoneella on oma käyttöjärjestelmänsä, jonka päälle käyttäjä voi rakentaa oman ohjelmiston tai sen osan (Kuva 6). Virtuaalikoneet toimivat omina ympäristöinään. Virtuaalikoneiden etuja ovat muun muassa kyky ajaa eri käyttöjärjestelmää kuin isäntälaitteella sekä isäntälaitteen resurssien jako eri virtuaalikoneympäristöihin. (What is a virtual machine? s.a.)



Kuva 6. Virtuaalikoneen rakenne

Kuva 6 kuvaa virtuaalikoneen rakennetta. Pohjakerroksessa on isäntälaiteympäristö ”infrastructure”. Sen yläpuolella on ”hypervisor”-kerros, joka hallitsee virtuaalikoneita, esimerkiksi toteuttamalla virtuaalikoneiden luomisen ja poiston. Seuraavana yläpuolella ovat pystysuunnassa jaetut virtuaalikoneinstanssit. Jokaisella virtuaalikoneella on oma käyttöjärjestelmä ”guest os”, ohjelmisto ”app” sekä sidoskirjastot ”bins/lib”.

3.3.3 Ohjelmistokonttien ja virtuaalikoneiden erot

Konttien ja virtuaalikoneiden keskeisin ero on niiden käyttöjärjestelmän toiminnassa. Kontit jakavat yhteisen käyttöjärjestelmän, kun taas kaikille virtuaalikoneille määritellään oma käyttöjärjestelmä. Seurauksena kontit ovat kooltaan kevyempiä. Konttien luominen ja poistaminen on siis nopeampaa. Konttien etuna on niiden rakenteellinen keveys, hallinnan helppous ja kyky siirtyä ympäristöstä toiseen. Virtuaalikoneet taas ovat turvallisempia, koska jokaisella virtuaalikoneella on oma käyttöjärjestelmä. Yhdelle virtuaalikoneelle murtautuminen ei siis tarkoita, että kaikkien virtuaalikoneiden tietoihin olisi päästy käsiksi. (geeksforgeeks 2020.)

3.4 Orkestrointi

Orkestroinnilla tarkoitetaan prosessia, jossa hallitaan useita ohjelmistopalvelun osia ja siihen liittyvää dataa työnkulun helpottamiseksi. Työnkululla tarkoitetaan ohjelmiston prosesseista koostuvaa kokonaistoimintaa. Orkestroinnin avulla voidaan luoda ympäristö, jossa virtuaaliresursseja hallitaan. Orkestroinnin avulla voidaan automatisoida yksittäisiä tehtäviä tai palvelu kokonaisuudessaan. Automatisoidut tehtävät yhdistetään muuhun työnkulkuun yhden virtaviivaisen prosessin luomiseksi. Orkestroinnin avulla voidaan esimerkiksi automatisoida ohjelmistopalvelun skaalautuminen, vikasieto sekä resurssien hallinta. Sen avulla myös vähennetään ohjelmistopalvelun ylläpitoon kuluvaa aikaa sekä resursseja. Orkestrointi on hyödyllinen silloin, kun palvelun rakenne muodostuu useista eri komponenteista. Orkestroinnin avulla helpotetaan palvelun komponenttien yhteistä toimintaa. Lisäksi sen avulla voidaan kerätä dataa palvelun tapahtumista. Datan avulla voidaan esimerkiksi havaita tapahtuvia virheitä tai seurata palvelun kuormitusta. (Red Hat 2019b.)

Kubernetes

Kubernetes on laajasti käytetty avoimen lähdekoodin ympäristö orkestrointiin. Kubernetestä käytetään usein ohjelmistokonttipohjaisten palveluiden kehityksessä, koska sen työkaluilla voidaan kehittää ohjelmistopalvelu kokonaisuudessaan. Se tarjoaa muun muassa ympäristön palvelun kehitykseen ja ylläpitoon, työkaluja esimerkiksi palvelun skaalaamiseen, datan keruun ajon aikana, resurssien optimoinnin sekä vikasietotilan. Käytännössä palveluohjelmistoa ylläpitävät ohjelmistokontit rakennetaan Kubernetes-ympäristöön, jossa niitä voidaan kehittää, hallita ja automatisoida. Lisäksi Kubernetes tarjoaa työkaluja ohjelmistopalvelun julkistamiseen. Julkistamisella tarkoitetaan menetelmää, jonka avulla ulkoinen liikenne ohjataan ohjelmistokontteihin. Kubernetes tukee myös useita kolmannen osapuolen luomia työkaluja, joka tekee siitä joustavan eri tilanteissa. Kubernetes toimii myös pilvessä. (Kubernetes.io 2022a.)

Kubernetes-rakenne

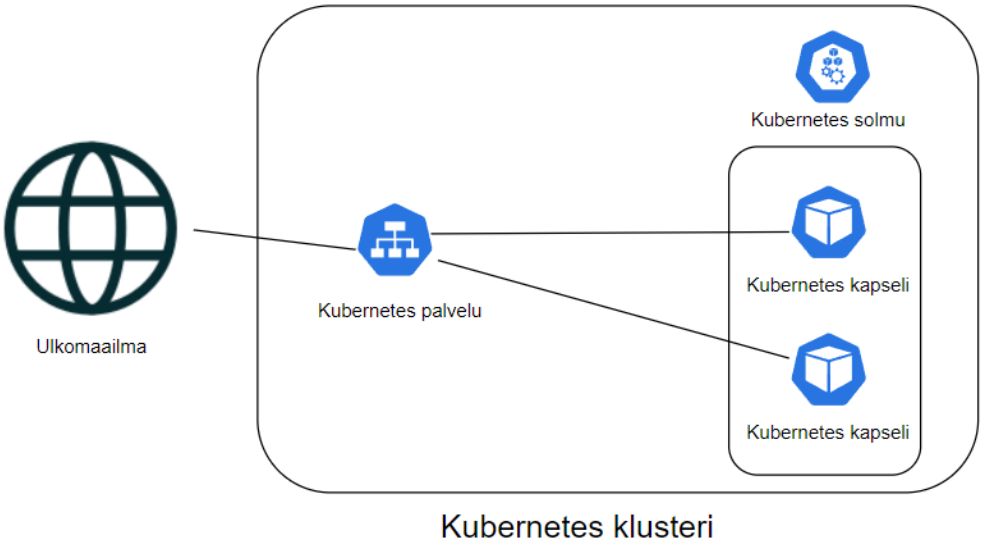
Kubernetes-ympäristön käyttö aloitetaan luomalla Kubernetes-klusteri (Kubernetes cluster). Kubernetes-klusteri koostuu solmuista, joissa ylläpidetään palvelimia. Palvelimia ylläpitäviä solmuja kutsutaan työläissolmuksi (worker node) ja niitä hallitsevaa solmua pääsolmuksi (master node). Klusteriin lisätään Kuberneteksen toimintaan tarvittavat komponentit, jonka jälkeen sinne voidaan lisätä solmut ja palvelun ohjelmistokontit. Jokaisella komponentilla on oma IP-osoitteensa, jonka avulla sen kanssa voidaan kommunikoida. Klusteri siis toimii omana ympäristönä, johon Kubernetes komponentit rakennetaan. (What is a Kubernetes cluster? s.a.)

Kubernetes-kapseli (Kubernetes pod) on Kuberneteksen matalimman tason komponentti. Kapselissa ajetaan ohjelmistokonttia, jossa voi olla esimerkiksi palvelin. Kapseleille luodaan pohjamalli, jonka avulla niitä luodaan. Niiden toiminta on hyvin saman kaltainen Docker-ohjelmistokonttien kanssa. Ohjelmistokonttiin rakennettu palvelin lisätään kapseliin, josta Kubernetes voi hallita sitä. Ohjelmistokontin ohjelman suorituksen jälkeen kapseli tuhotaan. Kapseleita voidaan lisätä ja poistaa. Samassa ympäristössä on usein useita kapseleita, joiden välille palvelun asiakasliikenne jaetaan. Kapselit sijaitsevat Kubernetes-solmuissa. Kuva 7 on kuvattu kapseleita klusterissa. (Kubernetes.io 2022b.)

Kubernetes-solmu (Kubernetes-node) toimii ympäristönä Kubernetes-kapseleille. Solmut luodaan Kubernetes-klusteriin, jossa niitä voi olla useita. Solmu toimii omana fyysisenä tai virtuaalisena koneena. Se sisältää kaikki kapselin ajamiseen tarvittavat toiminnot. Kapseleille liikennöinti tapahtuu solmujen osoitteiden kautta. (Kubernetes.io 2022c.)

Kubernetes tarjoaa erityisiä Kubernetes-palvelu-työkaluja (Kubernetes services), joiden avulla hallitaan liikennettä klusterin sisäverkkoon. Kubernetes-palveluiden tarkoitus on kääntää klusterin ulkopuolelta tuleva liikenne sisäverkon puolelle. Ne siis mahdollistavat palvelupyyntöjen ohjaamisen ohjelmistopalvelua ylläpitäviin kapseleihin. Kubernetes tarjoaa kolme eri palvelutyyppiä, joiden toiminta eroaa toisistaan. Palveluiden tyypit ovat ”loadbalancer”, ”nodeport” ja

”clusterservice”. Kuva 7 on kuvattu Kubernetes-palvelun välityksellä julkaistuja kapseleita. (Kubernetes.io 2022d.)



Kuva 7. Kubernetes-rakenne yksinkertaisuudessaan

Kubernetes-asettelu (Kubernetes deployment) on määritelmä Kubernetes-komponenttien joukosta, josta palvelu voidaan luoda kerralla kokonaisuudessaan. Käytännössä asettelun avulla määritellään Kubernetes-palvelun haluttu tila. Asettelussa voidaan esimerkiksi alustaa haluttu määrä kapseleita ja palveluita, jotka luodaan asettelun käynnistyksen yhteydessä. Sen avulla vältytään tarpeelta luoda jokainen Kubernetes-komponentti yksitellen, koska asettelu rakentaa kaikki komponentit sidoksineen kerralla. Asettelun asetuksia voidaan muokata ajon aikana. Asettelu siis toimii toimintasuunnitelmana Kubernetes-pohjaisen ohjelmistopalvelun rakenteelle. (Kubernetes.io 2022e.)

3.5 Pilvipalvelut

Pilvipalvelut ovat internetin välityksellä tarjottavia tietokoneresursseja sekä palveluita, jotka ovat erityisen suosittuja ohjelmistopalvelujen kehityksessä. Pilvipalvelut tarjoavat virtualisoituja ohjelmistoja, alustoja, ympäristöjä sekä työkaluja asiakkaiden käyttöön. Asiakas voi esimerkiksi rakentaa ohjelmiston pilvipalveluun, jolloin palvelun ylläpito tapahtuu pilvipalvelun ympäristössä. Asiakas välttyy tarpeelta hankkia fyysisiä laitteita palvelun tarjoamiseen, koska

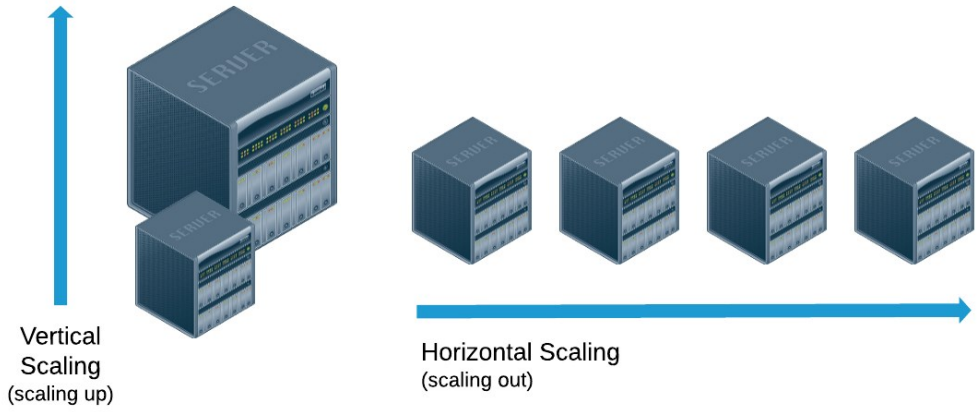
ohjelmisto käyttää pilvipalvelun resursseja. Asiakas maksaa pilvipalvelun tarjoajalle käytetyistä resursseista. Pilvipalvelut tarjoavat hyödyllisiä ratkaisuja ja työkaluja sovelluskehitykseen. Niihin on usein integroitu useita sovelluskehityksessä tarvittavia komponentteja, esimerkiksi tietokantoja ja tietokoneita. (IBM Cloud Team 2020.; Red Hat 2019a.)

Pilvipalveluja voidaan hyödyntää eri tavoilla tilanteesta riippuen. Pilvipalveluiden palvelumallit on usein jaettu kolmeen eri kategoriaan: ”IaaS”, ”Paas” ja ”SaaS”. Palvelumalleissa asiakkaan ja pilvipalvelun vastuut vaihtelevat mallin perusteella. Iaas (Infrastructure as a Service) eli infrastruktuuri palveluna -mallissa pilvipalvelu tarjoaa asiakkaalle ympäristön ja tietokoneresursseja. Asiakkaan on itse rakennettava ohjelmistopalvelu pilvipalveluympäristöön sen tarjoamilla tietokoneresursseilla. Asiakas on siis vastuussa ohjelmistopalvelun toimintalogiikasta, rakenteesta sekä palvelulogiikasta. Paas (Platforms as a Service), eli alusta palveluna -mallissa pilvipalvelu tarjoaa alustat asiakkaan käyttöön. Alustoja ovat esimerkiksi Kubernetes ja jotkin tietokannat. Lisäksi pilvipalvelu on vastuussa käyttöjärjestelmästä. Pilvipalvelu on siis vastuussa infrastruktuurin lisäksi ohjelmistopalvelun alustoista ja niiden lisensseistä. Saas (Software as a Service), eli ohjelmisto palveluna -mallissa pilvipalvelu tarjoaa asiakkaan ohjelmistolle kaiken tarvittavan ohjelmistopalvelun ylläpitämiseen. Infrastruktuurin ja alustan lisäksi Saas tarjoaa itse sovelluksen asiakkaan käyttöön. Palvelu on siis käytännössä täysin pilvipalvelun ylläpitämä. (Watts, S & Raza, M. 2019.)

Pilvipalvelun käytöllä on useita etuja. Asiakas voi siirtää ohjelmiston osan tai kokonaisuuden pilvipalveluun säästääkseen resursseja. Esimerkiksi tilaa vievän tietokannan voi rakentaa pilveen, jolloin asiakkaan ei tarvitse enää kuluttaa omia tietokoneresurssejaan sen ylläpitämiseen. Lisäksi ohjelmistoa ylläpitävän infrastruktuurin vastuu siirtyy pilvipalvelun ylläpitäjälle. Pilveen voidaan myös rakentaa testi- ja kehitysympäristöjä. Pilvipalvelut myös tarjoavat analytiikka-apuvälineitä, joiden avulla voidaan seurata sovelluksen suorituskykyä.

Skaalaaminen

Asiakas maksaa pilvipalvelulle varatuista ja käytetyistä resursseista, jonka vuoksi kehitetyn sovelluksen resurssienkulutus kannattaa optimoida. Skaalaamisella tarkoitetaan resurssien varausta sovellukselle tarpeen mukaan. Ohjelmistopalvelun resursseja lisätään ja vähennetään tarvittaessa. Jos esimerkiksi ohjelmistopalvelulla on paljon liikennettä, ohjelmistopalvelun resursseja lisätään, kunnes kaikkia asiakkaita voidaan palvella. Jos palvelulla ei ole liikennettä, resurssien määrää vähennetään kulujen säästämiseksi. Skaalaaminen toimii siis molempiin suuntiin. Skaalaamisella pyritään löytämään tasapaino palvelukyvyn ja palvelun ylläpitokustannuksien välille. Ohjelmistopalvelujen päivittäinen kuorma vaihtelee, jonka vuoksi skaalaamisen tulisi tapahtua automaattisesti palvelunpyyntöjen määrän perusteella. Ohjelmiston rakenteesta riippuen sovelluksen skaalausmenetelmä jaetaan usein kahteen vaihtoehtoon: vertikaaliseen ja horisontaaliseen. (Cloudzero 2021.; Section.io 2020.)



Kuva 8. Pilvipalveluiden skaalaussuunnat

Vertikaalinen skaalaus tarkoittaa sovellusta ylläpitävän laitteen resurssimäärän muuttamista tilanteen mukaan. Laitteiston tietokoneresurssien määrää lisätään tai vähennetään vastaamaan palvelupyyntöjen määrää. Laitteen skaalaaminen suuremmaksi lisää laitteen resursseja, joka parantaa laitteistolla ylläpidetyn palvelun kykyä palvella asiakkaita. Laitteen skaalaaminen pienemmäksi taas heikentää palvelukykyä, samalla vähentäen ylläpidon kustannuksia. Vertikaalinen skaalaus on usein helpompi ja nopeampi tapa skaalata, koska se ei vaikuta sovelluksen logiikan ajoon. (Kuva 8) (Cloudzero 2021.; Section.io 2020.)

Horisontaalinen skaalaus tarkoittaa palvelun palvelulaitteiden lisäämistä ja poistamista tarpeen mukaan. Palvelupyyntöjen kasvaessa laitteita lisätään vastaamaan kysyntää, kun taas pyyntöjen vähentyessä laitteita vähennetään. Horisontaalisessa skaalauksessa laitteiston ylläpitämän ohjelmistopalvelun logiikkaa täytyy usein muokata tukemaan useita palvelimia. Koska palvelulaitteiden määrää lisätään, palvelun osat jakautuvat. Sen vuoksi ohjelmistopalveluun tarvitaan logiikka, joka ohjaa liikennettä myös toisille laitteille. Palvelun osien jakautuminen voi myös aiheuttaa muita rakenteellisia ongelmia. Horisontaalisen skaalauksen etuna on sen kyky kasvaa hyvin suureksi, jonka vuoksi palvelu voi palvella erittäin suurta asiakasmäärää. Vertikaalinen skaalaus taas rajoittuu yhden laitteen maksimi resurssikapasiteettiin. Lisäksi horisontaalisen skaalauksen virheensietokyky on parempi, koska yhden laitteen kaatuessa koko palvelu ei hajoa. Hajonneen laitteen liikenne voidaan myös jakaa muiden laitteiden välille. (Kuva 8) (Cloudzero 2021.; Section.io 2020.)

4 TOTEUTUKSEN SUUNNITELMA

Tässä osuudessa suunnitellaan kehitystyön toteutus. Kehitystyönä toimii taustajärjestelmän kehitys peliin Superraketti. Suunnitelma aloitetaan määrittelemällä kehitystyön tavoitteet. Tavoitteina määrittelemisessä hyödynnetään aikaisemmin luotua viitekehystä. Niiden jälkeen valitaan menetelmät ja työkalut, joiden avulla ohjelmistopalvelu kehitetään. Seuraavana hahmotellaan palvelun rakenne tavoitteiden ja työkalujen perusteella. Rakenteeseen kuuluvat ympäristöt, palvelun komponentit sekä niiden väliset sidosverkostot. Rakenteesta tullaan luomaan visuaaliset kaaviot hahmotuksen helpottamiseksi. Työkalut valitaan työhön sopivuuden mukaan. Työkalujen ja niihin liittyvien menetelmien toiminta on selitetty teoriaosuudessa.

4.1 Tavoitteet

Suunnitelma on hyvä aloittaa listaamalla kehitystyön tavoitteet. Kuten aikaisemmin mainittiin, työn lopputavoitteena on luoda taustajärjestelmä Superraketti-peliin. Taustajärjestelmään luodaan palvelu, joka tarjoaa pelin pelipalvelimet pelaajille. Alapuolelle on listattu kehitystyön sisällölliset tavoitteet, joista taustajärjestelmä tulee koostumaan.

Pelaajien halutaan pystyvän liittymään pelin pelipalvelimille internetyhteyden kautta, jotta pelaajat voivat pelata peliä mistä tahansa sijainnista internetyhteyden välityksellä. Tämän avulla pelaajia voidaan palvella globaalisti. Lisäksi se mahdollistaa järjestelmän sijoittamisen yhteen sijaintiin.

Ohjelmistopalvelu halutaan rakentaa pilveen, koska pilvipalvelun tarjoajat tarjoavat tietokone resursseja asiakkaiden käyttöön kohtuulliseen hintaan. Pilvipalvelun avulla voidaan julkaista ohjelmistopalvelu asiakkaille. Ohjelmistopalvelu voidaan myös keskittää kokonaisuudessaan pilvipalvelun ympäristöön. Lisäksi pilvipalvelut tarjoavat käyttöön useita työkaluja. Palvelu isännöidään pilvipalvelun tarjoamassa tilassa, mikä poistaa tarpeen hankkia kalliita tietokoneita ja muita laitteita palvelun isännöintiin. Pilven käyttö poistaa vastuun fyysisen palvelulaitteiston ylläpidosta. Huonona puolena on tietenkin se, että ohjelmistopalvelun kehittäjä ei pysty korjaamaan ylläpidon ongelmia, koska niistä vastaa pilvipalvelun ylläpitäjä.

Pelipalveluun halutaan luoda pelipalvelin-instansseja, joihin pystyy liittymään tietty määrä pelaajia. Jokainen pelipalvelin-instanssi siis toimii peliaulana. Peli-instanssi on pelipalvelin, joka lisätään ohjelmistokonttiympäristöön. Pelipalvelimellisesta ohjelmistokontista luodaan malli, jonka pohjalta muut ohjelmistokontit luodaan. Jokaiseen ohjelmistokonttiin lisätään yksi pelipalvelin-instanssi. Ohjelmistokonttien avulla voidaan palvella suuria määriä pelaajia samankokoisissa peliauloissa. Lisäksi niitä on helppo lisätä ja poistaa. Kun pelipalvelin-instanssi on täynnä, siihen ei enää päästetä uusia pelaajia. Pelaajat voivat poistua instanssista vapauttaen paikan uudelle pelaajalle.

Peli-instanssien dataa halutaan kerätä. Instanssista kerättävään dataan voisi kuulua muun muassa aktiivisuus, resurssien kulutus, virheilmoitukset ja pelaajamäärä. Datan kerääminen on hyvin tärkeää palvelun ylläpidon kannalta. Kerätyn datan perusteella voidaan reagoida erilaisiin tilanteisiin. Esimerkiksi pelaajamäärä auttaa selvittämään, missä pelipalvelin-instansseissa on tilaa. Lisäksi virheilmoituksilla mahdollisesti hajonneet instanssit voidaan poistaa. Aktiivisuuden avulla taas voidaan sulkea turhia instansseja toiminnasta resurssien säästämiseksi.

Pelipalvelimia halutaan orkestroida. Orkestroinnin avulla voidaan hallita ohjelmistopalvelun toimintaa kokonaisuudessaan. Sen avulla vältytään tarpeelta muuttaa ohjelmistopalvelun tilaa komponentti kerrallaan. Käytännössä lopputavoitteena olisi täysin automatisoitu järjestelmä. Orkestroinnin avulla voidaan hallita ohjelmistokontteja ja palvelun tilaa kokonaisuudessaan. Lisäksi yksittäisiä tehtäviä voidaan automatisoida. Automatisoinnin avulla voidaan esimerkiksi luoda menetelmä, joka käynnistää kaatuneet pelipalvelimet automaattisesti uudelleen. Orkestroinnin avulla voidaan myös kehittää skaalaamismenetelmä ohjelmistopalvelulle. Lisäksi orkestrointiin kuuluu ohjelmistopalvelun datan keruu.

4.2 Työkalut

Edellisessä alaluvussa käytiin läpi tavoitteet. Tavoitteiden perusteella tiedetään, minkä tyyppisiä työkaluja taustajärjestelmän kehitykseen tarvitaan. Tässä alaluvussa käydään läpi työkaluja, joita taustajärjestelmän kehityksessä hyödynnetään.

Superraketti-peli on luotu Unity-pelieditorissa, joten pelipalvelimet rakennetaan siinä. Pelipalvelimet tulevat olemaan Linux-pohjaisia. Linux-pohjaiset pelipalvelimet tullaan lisäämään ohjelmistokontteihin, joista ne palvelevat asiakkaita. Toisena vaihtoehtoina on Windows-pohja. Linux-pohjan etuna Windowsiin nähden on sen turvallisuus, matala resurssien kulutus ja korkea suorituskyky. Lisäksi Linux on ilmainen, kun taas Windowsin käytöstä saattaa joutua maksamaan Microsoftille. (Simic 2021.)

Taustajärjestelmän kehitykseen tarvitaan ohjelmistokonttikirjasto. Ohjelmistokonttikirjaston avulla luodaan ohjelmistokontteja, joiden ympäristössä ajetaan pelipalvelimia. Ohjelmistokonttityökirjastoksi valitaan Docker. Docker tukee pelipalvelimien käyttämää käyttöjärjestelmää, joka mahdollistaa pelipalvelimien ajon ohjelmistokonttiympäristössä. Ohjelmistokontista luodaan vedos, jonka pohjalta uudet kontit luodaan. Vedoksessa määritellään ohjelmistokontin ajokomennot ja asetukset. Asetusten yhteydessä määritellään ohjelmistokontin portti, jonka kautta asiakkaat yhdistetään palvelimelle. Jokaisella ohjelmistokontilla on oma IP-osoitteensa. Lisäksi Docker on hyvin dokumentoitu, päivitetty, helposti käytettävä sekä laajasti tuettu kirjasto.

Ohjelmistokonttien hallintaan tarvitaan orkestrointikirjasto. Orkestrointikirjastoksi valitaan Kubernetes, koska se tukee Docker-pohjaisia ohjelmistokontteja. Kuberneteksen avulla luodaan alusta, johon ohjelmistokonttipohjainen palvelu rakennetaan. Kubernetes tarjoaa toimintoja ja työkaluja ohjelmistopalvelun kehitykseen sekä julkaisuun. Sen avulla luodaan taustajärjestelmän palvelu kokonaisuudessaan. Taustajärjestelmän osat liitetään yhteiseen työnkulkuun, joka pyritään automatisoimaan. Lisäksi Kuberneteksen avulla pelipalvelimia voidaan skaalata horisontaalisesti.

Pilvipalveluksi valitaan Google Cloud, eli Gcloud. Gcloud valitaan, koska siitä löytyy tarvittavat työkalut pelipalvelimien ajoon pilvipalvelun ympäristössä. Ohjelmistopalvelu tullaan kokonaisuudessaan rakentamaan Gcloud-ympäristöön. Gcloud tarjoaa uusille asiakkaille 300 dollaria ilmaista luottoa, jonka avulla taustajärjestelmää voidaan kehittää ja ylläpitää ilmaiseksi jonkin aikaa. Gcloudin dokumentointi on kattavaa, jonka lisäksi se tukee Kubernetes-pohjaisia ohjelmistoja. Gcloudissa pelipalvelinkontit voidaan lisätä Kubernetes-ympäristöön, josta niitä voidaan hallita. Lisäksi Gcloudin tukema Agones-niminen kirjasto mahdollistaa pelipalvelimien ja Kubernetes-alustan välisen kommunikoinnin. Gcloud sekä Kubernetes ovat molemmat Googlen tarjoamia, joten niiden yhteistoiminta on taattu. Pilvipalvelun palvelumalliksi tulee siis PaaS eli platform as a service.

Koska taustajärjestelmä rakennetaan Gcloud-ympäristöön, siinä tullaan käyttämään Agones-kirjastoa. Agones on Googlen luoma kirjasto, jonka avulla luodaan mukautettuja Kubernetes-komponentteja, jotka erikoistuvat pelipalvelinpalvelun tarjontaan. Käytännössä Agones toimii lisäosana Kubernetes-pohjaiselle taustajärjestelmälle. Agoneksen avulla luodaan erityisiä pelipalvelinkapseleita, joissa pelipalvelinta ajetaan. Lisäksi Agoneksen tarjoaa mukautetun palvelun pelipalvelimien tarjoamiseksi. (Agones.dev 2022.)

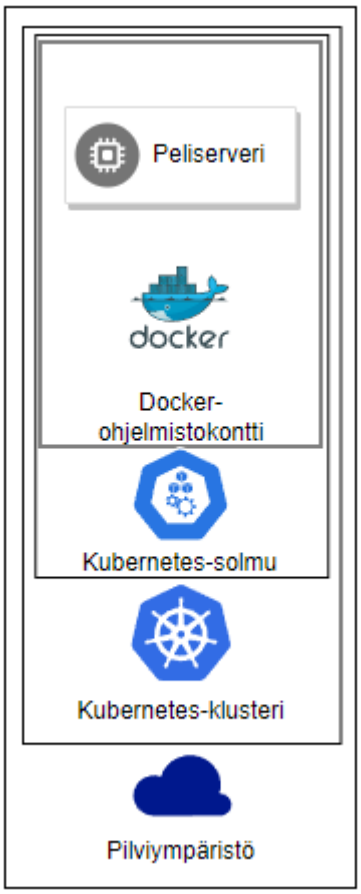
4.3 Kehitystyön rakenne

Superraketin taustajärjestelmä halutaan rakentaa pilvipalveluympäristöön, jotta taustajärjestelmän ylläpitoon ei tarvitse käyttää omia tietokoneresursseja.

Lisäksi pilvipalvelun tarjoamat työkalut saadaan käyttöön, joka helpottaa palvelun kehitystä. Taustajärjestelmälle luodaan kehitysympäristö pilveen, johon kaikki taustajärjestelmän komponentit tullaan rakentamaan. Taustajärjestelmä rakennetaan siis kokonaisuudessaan pilvipalveluun. Ohjelmistopalveluun kuuluu tässä tapauksessa asiakkaille tarjottavien pelipalvelimien kehitys ja ylläpito.

Tavoitteena on luoda ohjelmistokonttipohjainen pelipalvelinjärjestelmä. Jokaiseen ohjelmistokonttiin tullaan sisällyttämään yksi pelipalvelininstanssi. Pelipalvelininstanssi on pelistä luotu yksittäinen palvelin, joka palvelee pelaajia. Jokaisessa instanssissa toimii peli, johon mahtuu maksimissaan 8 pelaajaa. Palvelimen pelilogiikka on luotu valmiiksi, joten pelipalvelin täytyy vain rakentaa pelieditorissa. Pelipalvelinten kuuntelema portti on 7777 ja liikenneprotokolla UDP. Pelipalvelimien alustana toimii Linux-käyttöjärjestelmä. Pelipalvelimilla ei tarvitse olla visuaalista käyttöliittymää, joten niistä luodaan käyttöliittymätön versio. Käyttöliittymätön versio pienentää pelipalvelimen kokoa, joka nopeuttaa sen käynnistystä.

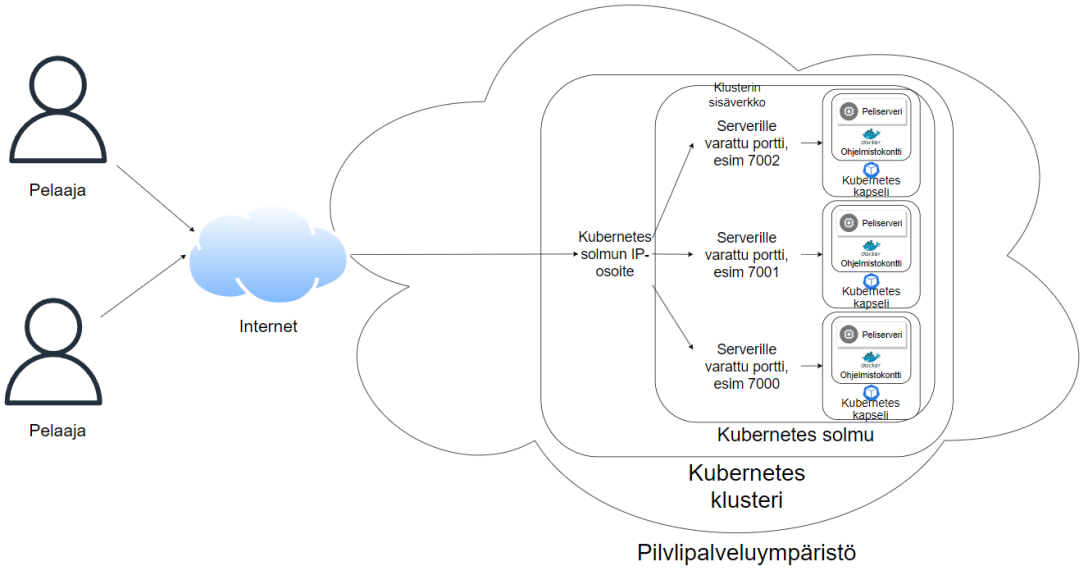
Dockerin avulla luodaan ohjelmistokontteja, joihin pelipalvelimet sijoitetaan. Jokainen kontti toimii ympäristönä yhdelle pelipalvelininstanssille. Ohjelmistokontti tulee olemaan osa Kubernetes-kapselia, jota Kubernetes voi hallita. Kubernetes-ympäristöä taas ylläpidetään pilvipalvelussa (Kuva 9). Pelipalvelimia tulee olla tarpeeksi pelaajien palvelimiseen.



Kuva 9. Pelipalvelimen ympäristörakenne

Pelipalvelu tullaan rakentamaan Gcloud-pilvipalvelun ympäristöön. Pilvipalvelu tarjoaa Kubernetes-alustan, jossa pelipalvelu voidaan kehittää. Kubernetek-sen avulla luodaan klusteri, joka toimii pelipalvelun ympäristönä. Klusteriin ra-kennetaan pelipalvelinkapseleita sekä palvelu, jonka avulla pelipalvelimet jul-kaistaan asiakkaille. Kubernetes-klusteriin luodaan Kubernetes-solmu, joka toimii toimintaympäristönä kapseleille. Solmu toimii kapseleiden isäntänä, jonka kautta kapselit julkaistaan. Kapselien luontiin käytetään Agoneksen mu-kautettua pelipalvelinkapselia, joka parantaa Kuberneteksen ja pelipalvelimen välistä toimintaa.

Tavoitteena on järjestelmä, jossa pelaajat liittyvät pelipalvelimille internetin vä-lityksellä. Kubernetes-pelipalvelinkapselit julkaistaan niitä ylläpitävän Kuberne-tes-solmun kautta. Pelipalvelinkapseleiden osoitteena toimii solmun IP-osoite ja porttina niille varattu solmun portti (Kuva 10).



Kuva 10. Taustajärjestelmän rakenne pilvipalvelussa

Taustajärjestelmän kehitys aloitetaan ensin paikallisessa tietokonejärjestelmässä suunnitelman testaamiseksi. Suunnitelman testaus suoritetaan paikallisesti, jotta voidaan varmistua sen toimivuudesta ennen kuin siirrytään pilvipalveluympäristöön. Aikaisemmin löydetyt ongelmat ovat helpompia ratkaista. Ohjelmistokonttien luominen ja ajaminen sekä Kubernetes-kapseleiden toiminta testataan paikallisesti. Testaus suoritetaan paikallisella tietokoneella. Tietokoneelle täytyy ensin asentaa Linux-käyttöjärjestelmä, koska pelipalvelimet toimivat vain Linux-pohjaisissa järjestelmissä. Testauksen jälkeen taustajärjestelmä rakennetaan pilvipalveluun.

5 TOTEUTUS

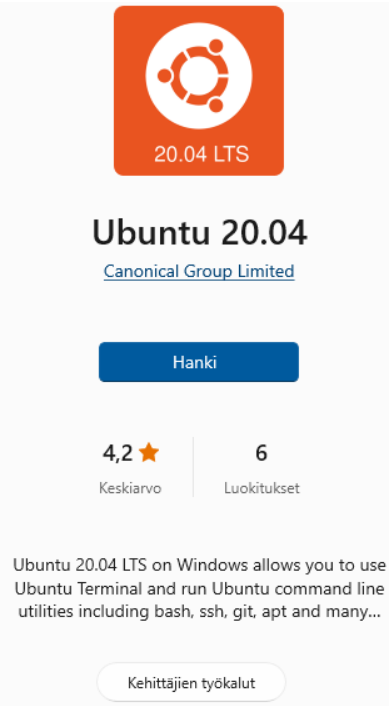
5.1 Suunnitelman testaus

Kuten suunnitelmassa mainittiin, toteutus aloitetaan testaamalla työkalujen toimintaa lokaalisti. Testin avulla kootaan kokonaisidea taustajärjestelmän toimivuudesta ja rakenteesta. Lisäksi työkaluihin tutustuminen lokaalisti nopeuttaa taustajärjestelmän rakennusprosessia pilvipalveluun, joka vähentää pilvipalvelun kustannuksia kehityksen aikana. Lokaalin testauksen kohteena on pelipalvelinten sisältäminen ohjelmistokonttiin, pelipalvelimien toimivuus ohjelmistokonteissa sekä Kubernetes-ympäristön toiminta pelipalvelinkonttien kanssa.

Pelipalvelimet rakennetaan Linux-käyttöjärjestelmän pohjalle. Kehitystyön tietokoneen käyttöliittymänä toimii Windows. Windows-pohjaiset laitteet eivät tue

Linuxin toimintaa, joten työssä hyödynnetään WSL-alijärjestelmää. WSL-alijärjestelmä luo oman Linux-ympäristön Windows-pohjaiselle laitteelle. Luodussa Linux-ympäristössä voidaan ajaa Linux-pohjaisia ohjelmia, joka mahdollistaa pelipalvelinten, ohjelmistokonttien sekä Kuberneteksen ajon. WSL ladataan koneelle komentokehotteessa komennolla `wsl –install`.

Seuraavaksi Linux-alijärjestelmään täytyy ladata Linux-distribuutio. Tässä työssä distribuutioksi valitaan Ubuntu versio 20.04, joka on yleisesti käytetty ja dokumentoitu Linux-distribuutio. Ubuntu voidaan ladata verkosta tai Microsoft Storesta (Kuva 11). Asennuksen yhteydessä Ubuntuun luodaan tunnukset, jotka kirjoitetaan talteen. Joissain Windows-laitteissa täytyy sallia WSL-asetus, ennen kuin alijärjestelmää voidaan käyttää. WSL version tulee olla WSL2.

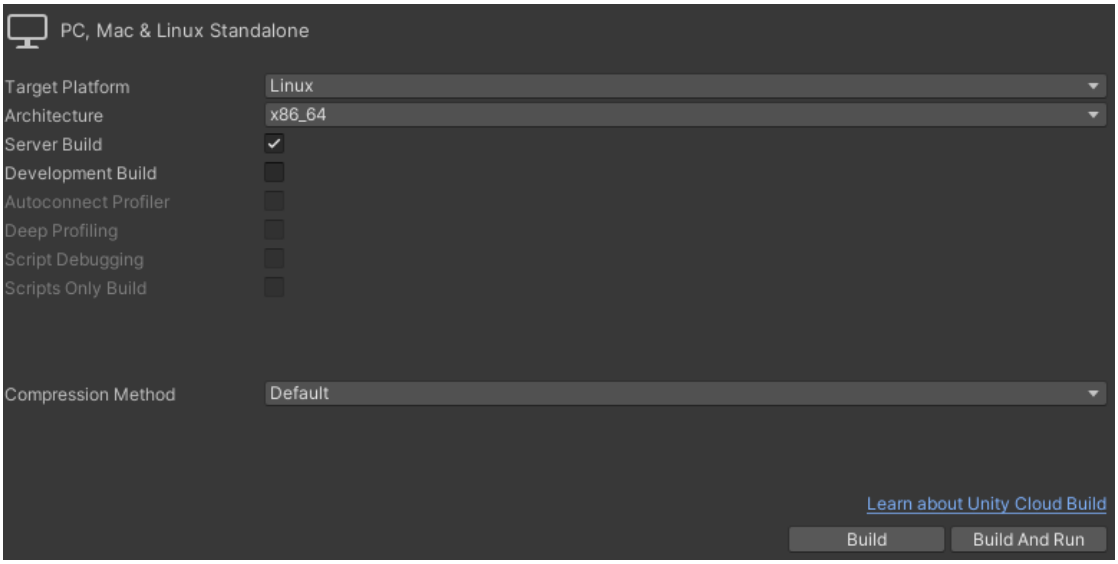


Kuva 11. Ubuntu Microsoft Storessa

5.1.1 Pelipalvelimen testaus

Seuraavaksi rakennetaan pelipalvelimet. Kuten aikaisemmin mainittiin, pelinpalvelimien toimintalogiikka on jo tehty, joten se ei sisälly tähän opinnäytetyöhön. Pelipalvelin rakennetaan Unity-pelieditorin ”build settings” -ikkunassa. Rakentamisella tarkoitetaan Unityn luomaa versiota pelistä. Unity rakentaa valittujen asetusten perusteella kansion, johon sisältyy pelin ajamiseen tarvitta-

vat tiedostot. Rakennusvalikosta asetuksiksi valitaan ”Linux”- ja ”Server”-valinnat, jotka määrittävät pelin versioksi pelipalvelimen sekä alustaksi Linux-käyttöjärjestelmän (Kuva 12). Pelipalvelimen kansio nimetään ja rakennetaan ”Build”-painikkeella.



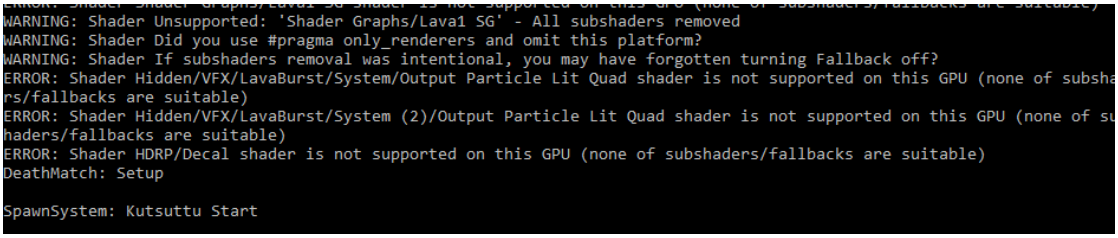
Kuva 12. Pelipalvelimen rakennus Unity-pelieditorissa

Seuraavaksi pelipalvelimen kansio täytyy siirtää Windows-järjestelmästä alijärjestelmään. Kansio saadaan siirrettyä Ubuntun terminaalista `cp` -komennolla. Komento kopioi kansion valitusta sijainnista uuteen sijaintiin. Komento on tässä tilanteessa kokonaisuudessaan `cp -a /mnt/d/Users/Petu_/Desktop/'Superraketti exe'/gameserver/. /home/petteri/server/gameserver/`. Kopion lähde- ja loppusijainti eritellään komennon keskellä olevan pisteen jälkeisellä välilyönnillä. Kopiointi voidaan varmistaa `ls server/gameserver` komennolla, joka tulostaa kansion sisältämät tiedostot terminaaliin (Kuva 13).

Kuva 13. ”gameserver” kansion tulostaminen terminaaliin

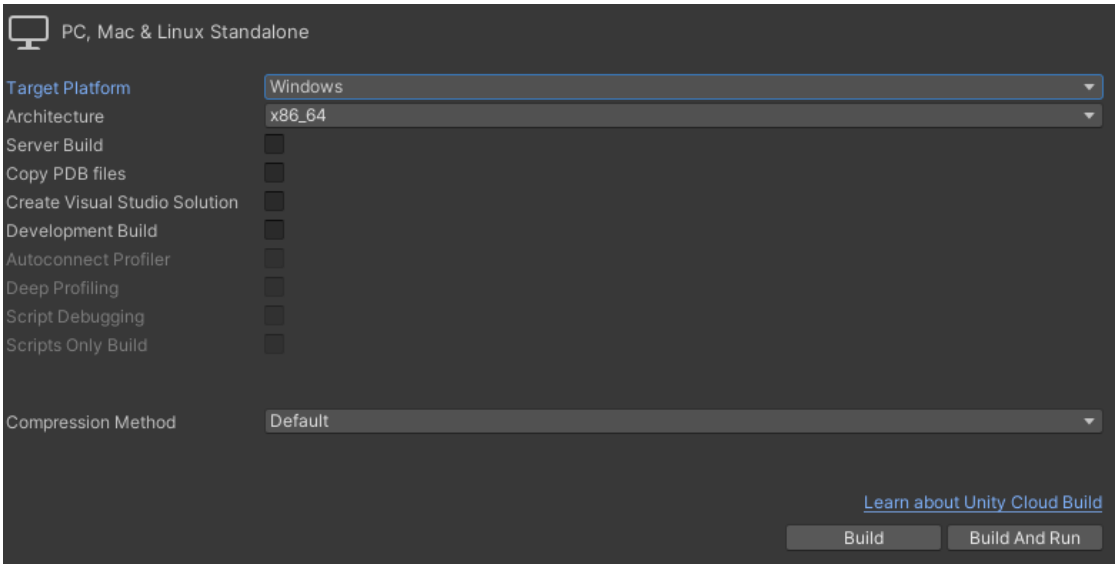
Kun pelipalvelimen kansio on kopioitu alijärjestelmään, voidaan pelipalvelimen toimintaa testauttaa. Pelipalvelimen rakennuksen yhteydessä luotiin exe-tiedosto, jonka avulla pelipalvelin voidaan käynnistää. Ensin navigoidaan terminaalissa pelipalvelimen kansioon. Navigointi tapahtuu `cd` -komennolla, jonka

avulla siirrytään terminaalissa valittuun alakansioon. Komennolla `cd server/gameserver` siirrytään pelipalvelimen alakansioon. Seuraavaksi palvelintiedosto merkitään ajettavaksi `chmod` -komennolla. Komento `chmod +x server_13012022.x86_64` ilmoittaa, että pelipalvelin käynnistetään komennolla terminaalissa. Sen jälkeen pelipalvelin voidaan käynnistää komennolla `./ server_13012022.x86_64`. Käynnistyksen yhteydessä terminaaliin ilmestyy pelipalvelimen luoma loki pelin käynnistysprosessista (Kuva 14). Tässä tilanteessa pelipalvelimen lokien ilmestyminen terminaaliin tarkoittaa onnistunutta käynnistystä.



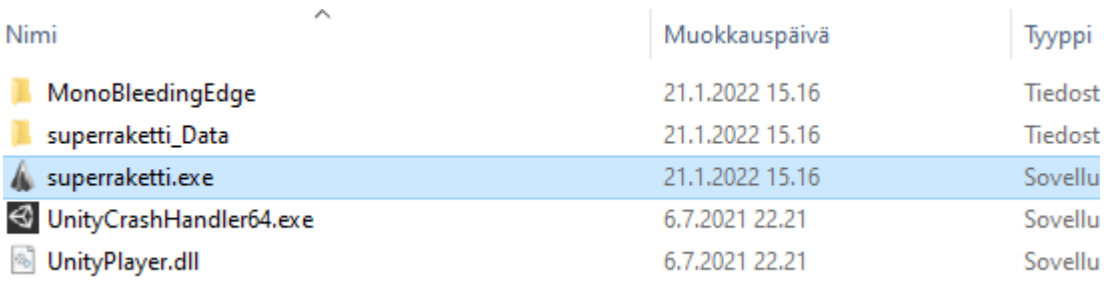
Kuva 14. Pelipalvelimen käynnistyksen loki

Nyt kun pelipalvelin on käynnistetty, sen toimintaa voidaan testata pelin asiakasversiolla. Asiakasversio on sovellusversio pelistä, jolla pelaajat pelaavat peliä. Asiakasversio luodaan samassa Unity-pelieditorin ikkunassa, jossa palvelinversio rakennettiin. Rakennusasetuksissa alustaksi asetetaan Windows, jonka lisäksi server-asetus otetaan pois käytöstä (Kuva 15). Edellä mainittujen asetusten pohjalta rakennetaan asiakasversio Windows-alustalle. Asiakasversio halutaan Windows-pohjaiseksi, koska alijärjestelmällä ei ole graafista käyttöliittymää, joten peliruutua ei voida tulostaa ruudulle. Asiakasversion käyttö siis tapahtuu tietokoneen Windows-alustalla. Asiakasversio tallennetaan halutuun sijaintiin, josta se voidaan käynnistää.

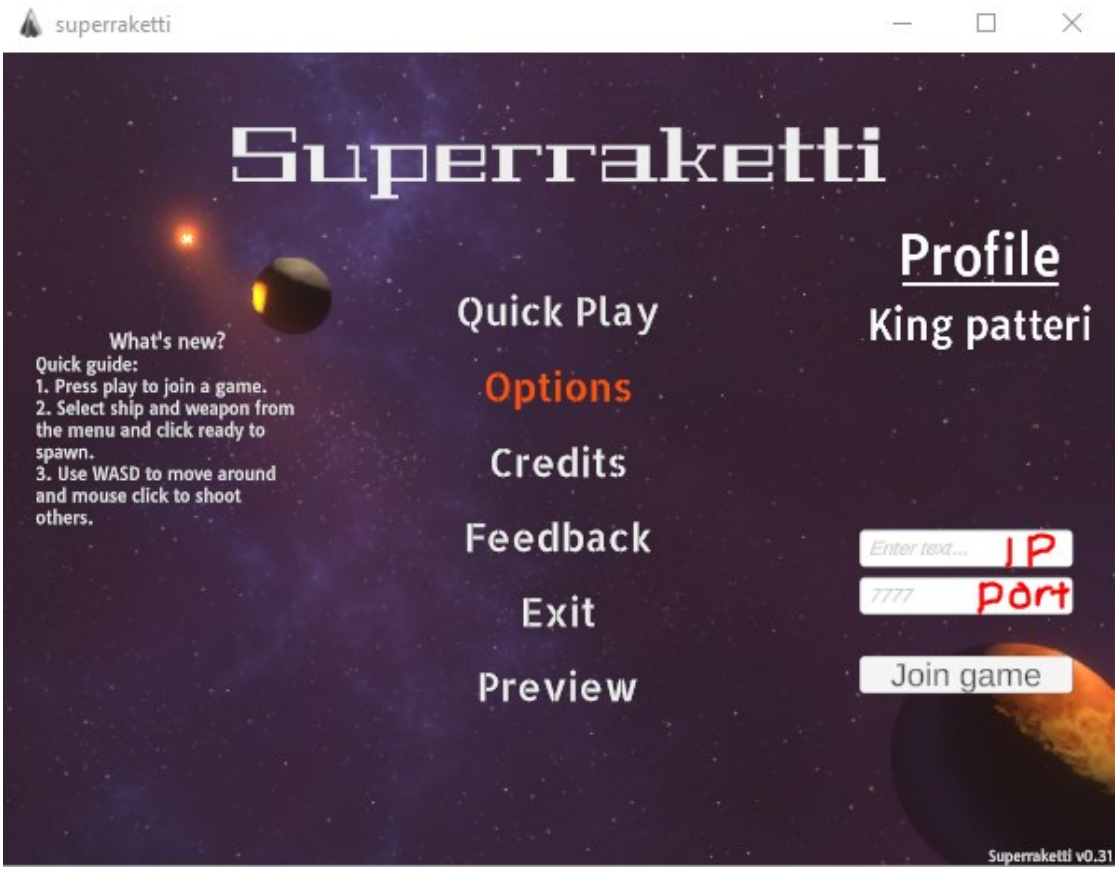


Kuva 15. Windows-pohjaisen asiakasversion asetukset

Asiakasversio käynnistetään asiakasversiokansion .exe-tiedostolla (Kuva 16). Asiakasversion käynnistyttyä pelin ikkuna avautuu. Peli alkaa päävalikosta, jonka oikeassa alareunassa on kaksi tekstikenttää: ylempi IP-osoitteelle ja alempi portille (Kuva 17). IP-osoitteen tekstikenttään syötetään tietokoneen IP-osoite. IP-osoitteen saa terminaalissa komennolla `ifconfig`. Tässä tilanteessa IP-osoite on 172.29.180.210 (Kuva 18**Virhe. Viitteen lähdettä ei löytynyt.**). Portin tekstikenttään syötetään `7777`, koska se on määritelty pelipalvelimen portiksi. ”Join game” -painikkeella pelaaja siirtyy peliin. Pelaajan liittyessä peliin pelipalvelimen terminaaliin alkaa ilmestymään pelikoodissa määritettyjä lokeja (Kuva 19). Jos asiakasversio ei pysty liittymään pelipalvelimelle, ruudulle tulostetaan virheilmoitus (Kuva 20).



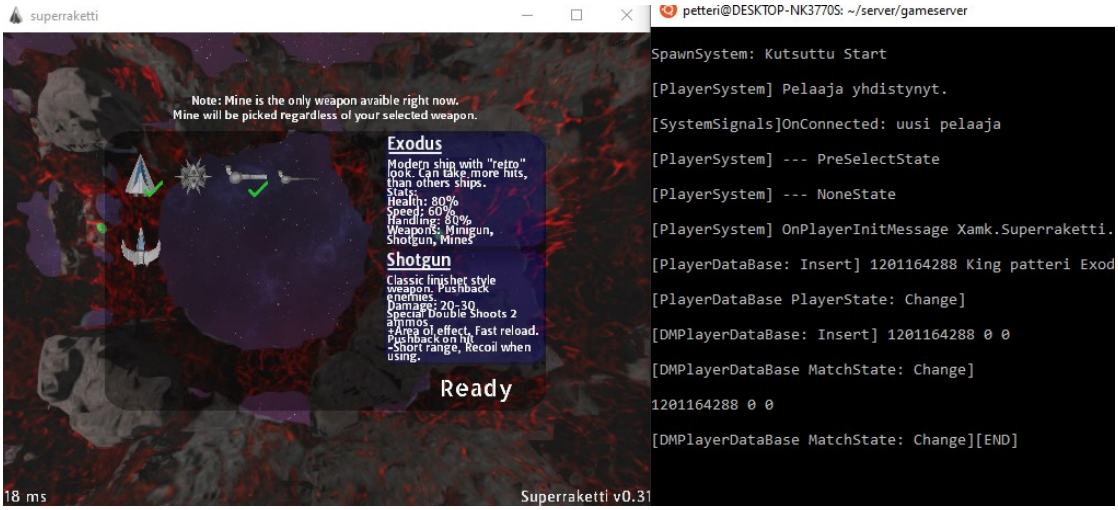
Kuva 16. Asiakasversion rakennettu kansio



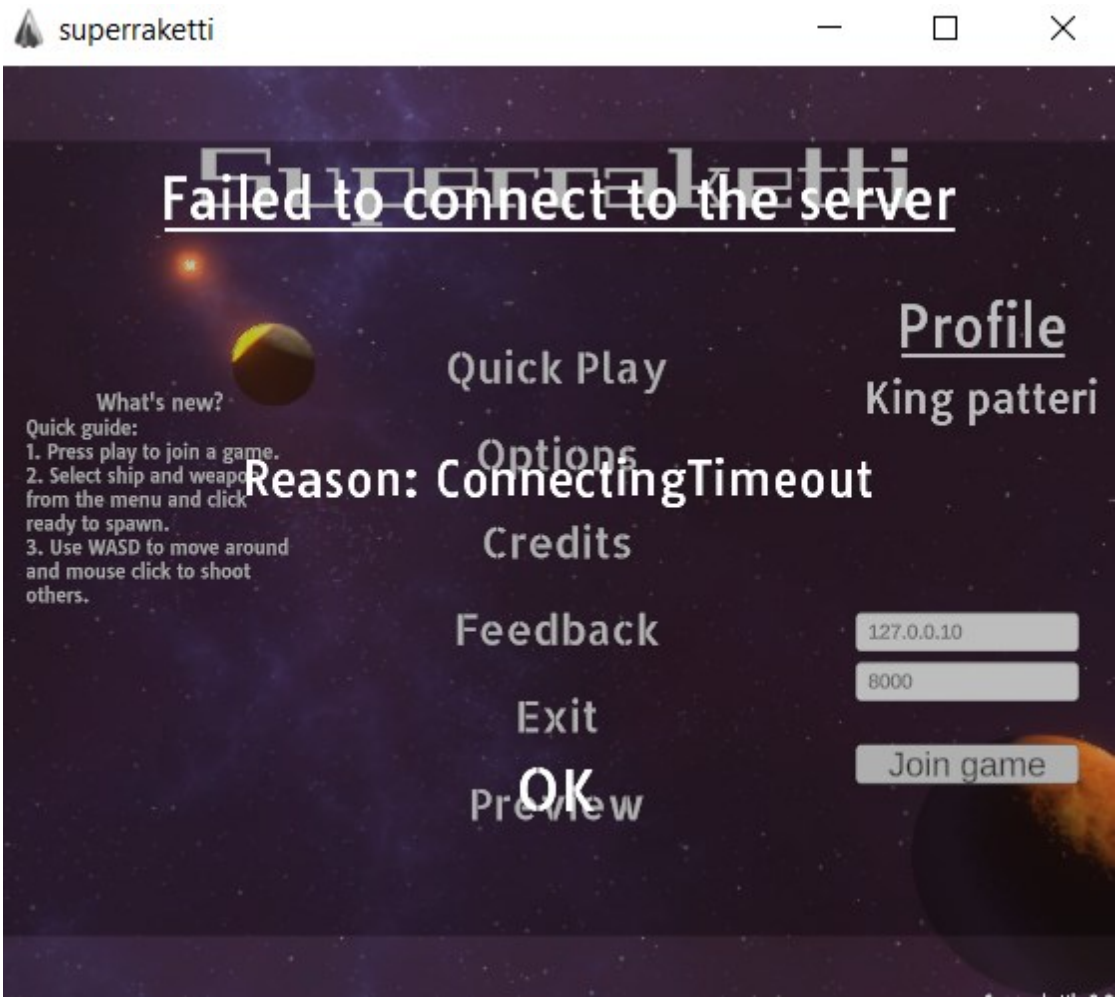
Kuva 17. Asiakasversion päävalikko

```
petteri@DESKTOP-NK3770S:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.29.180.210  netmask 255.255.240.0  broadcast 172.29.191.255
        inet6 fe80::215:5dff:fe48:bc52  prefixlen 64  scopeid 0x20<link>
        ether 00:15:5d:48:bc:52  txqueuelen 1000  (Ethernet)
        RX packets 2275  bytes 332657 (332.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 18  bytes 1356 (1.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Kuva 18. IP-osoitteiden haku komennolla terminaalissa



Kuva 19. Asiakasversio liittyneenä palvelimelle



Kuva 20. Asiakasversion virheilmoitus

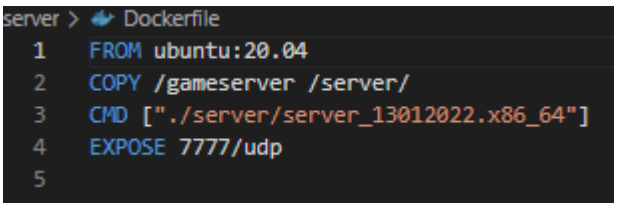
5.1.2 Pelipalvelinohjelmistokontti

Seuraavaksi luodaan ohjelmistokontti pelipalvelimelle Dockerin avulla. Docker on työkalu, jonka avulla ohjelmistokontit luodaan. Dockerin avulla luodaan standardoitu ohjelmistokontti, joka sisältää pelipalvelimen. Dockerin avulla luodaan pohja, jossa määritellään ohjelmistokontin asetukset. Ohjelmistokontin asetuksissa määritellään sisältöohjelman ajoon tarvittavat asetukset. Käytännössä Dockerin avulla luodaan ohjelmistokonttipohja, jonka avulla pelipalvelimet alustetaan ja käynnistetään pelaajien käyttöön automaattisesti. Docker on ladattu ja asennettu Docker-verkkosivujen ohjeiden (docs.docker) mukaan alijärjestelmään.

Ohjelmistokontit luodaan Docker-vedosmallista, joka toimii pohjamallina ohjelmistokonteille. Siinä konfiguroidaan ohjelmistokontin toimintaan liittyvät asetukset, esimerkiksi kontin pohjajärjestelmä ja ajokäskyt. Vedosmalli luodaan

Dockerfile-tiedoston pohjalta. Dockerfilessa määritellään vedoksen konfigurointiasetukset. Pelipalvelin kopiotiin ”server”-nimisen hakemiston ”gameserver”-nimiseen alahakemistoon. Terminaalissa navigoidaan ”server”-hakemistoon, jossa Dockerfile-tiedosto luodaan kirjoittamalla terminaaliin Linux komento `touch Dockerfile.dockerfile`.

Dockerfile-tiedosto avataan koodieditorissa, jotta sen sisältöä voidaan muokata. Tässä tilanteessa editorina käytetään Visual Code -editoria. Dockerfile-tiedostoon lisätään kentät, joilla määritetään kontin lähdekansio, käyttöjärjestelmä sekä pelipalvelimen ajoon vaadittavat komennot (Kuva 21).



Kuva 21. Pelipalvelimen Dockerfile-tiedosto

Kuva 21 tiedoston `FROM`-arvona on `ubuntu:20.04`, joka määrittää ohjelmistokontin käyttämän pohjajärjestelmän. Pohjajärjestelmäksi valitaan `ubuntu:20.04`, koska pelipalvelimen toimintaa testattiin alijärjestelmässä kyseisellä distribuutiolla aikaisemmin, joten sen toimivuus on varmistettu.

`COPY`-arvolla määritellään kopioitava sisältöhakemisto, joka on tässä tilanteessa `/gameserver /server/`. `COPY`-arvoksi siis annetaan pelipalvelimen hakemisto, joka kopioidaan ohjelmistokonttiin. `COPY`-arvossa määritellään kopioitava kansio sekä ohjelmistokontin hakemisto, johon pelipalvelin kopioidaan. Edellä annetulla `COPY`-arvolla kopioidaan ”gameserver”-nimisen kansion sisältö ohjelmistokontin ”server”-nimiseen hakemistoon.

`CMD`-arvo määrittelee kontissa ajettavan terminaalikomennon. Kun ohjelmistokontti on luotu, terminaalikomento ajetaan ohjelmistokontissa. `CMD`-arvoa halutaan käyttää ohjelmistokontin pelipalvelimen käynnistämiseen. Ohjelmistokontin pelipalvelin käynnistetään samalla tavalla kuin aikaisemmassa testissä eli komennolla `./server/server_13012022.x86_64`.

Viimeisenä määritellään `EXPOSE`-arvo. `EXPOSE`-arvo määrittelee ohjelmistokontin julkaisussa käytetyn vakioportin ja -protokollan. Kuten aikaisemmin mainittiin, pelipalvelin käyttää kommunikointiin porttia 7777. Pelipalvelin käyttää kuljetusprotokollana UDP-protokollaa, joten se lisätään porttinumeron loppuun. Portti ja protokolla eritellään jakomerkin avulla.

Dockerfile-tiedostosta luodaan seuraavaksi Docker-vedosmalli. Docker-vedosmallin luomiseksi terminaalissa täytyy navigoida hakemistoon, jossa Dockerfile-tiedosto sijaitsee. Docker-vedosmalli luodaan terminaalissa komennolla `docker build -t sr_server .`. Komento luo hakemiston Dockerfile-tiedoston asetusten pohjalta vedoksen, jonka nimeksi annetaan ”sr_server”. Vedosmallit voidaan listata terminaaliin komennolla `docker images`. Vedosmallin rakentaminen onnistui, jos listauksessa on mukana ”sr_sever” (Kuva 22).

```
petteri@DESKTOP-NK3770S:~/server$ docker images
REPOSITORY                    TAG       IMAGE ID       CREATED         SIZE
sr_server                     latest    c923b526e6c9   3 days ago      424MB
```

Kuva 22. Kuvassa listattu docker image ja sen tiedot

Seuraavaksi luodaan ohjelmistokontti vedoksen pohjalta. Vedosmallista käynnistetään ohjelmistokontti terminaalissa komennolla `docker run`. Komentoon voidaan lisätä ohjelmistokontin ajoasetuksia. Tässä tilanteessa käytetään komentoa `docker run -dp 8000:7777/udp sr_server`. Edellisen komennon ajoasetukset käydään seuraavaksi läpi.

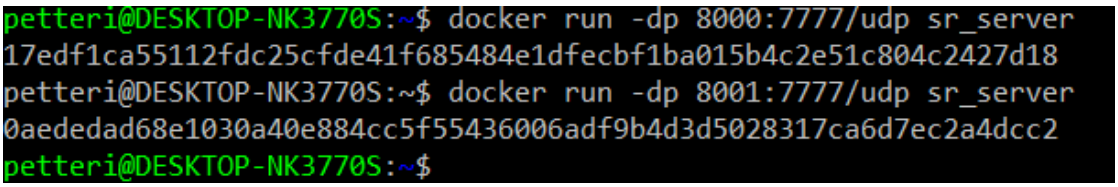
Komennon `-dp` osa määrittää kaksi asetusta. Ensimmäinen asetus on ”-d” (detached mode), joka estää pelipalvelinta tulostamasta lokeja terminaaliin. Toinen asetus on ”-p” (publish), joka ilmoittaa ohjelmistokontin portin julkaisusta.

Asetuksia seuraa konfigurointi vaihe, jossa kartoitetaan kontin julkaistava portti. Konfiguroinnin 8000:7777 osassa määritellään portinsiirto. Portinsiirron avulla siirretään liikenne portista toiseen porttiin. Tässä tilanteessa alijärjestelmän portin 8000 liikenne siirretään pelipalvelimen käyttämään ohjelmistokon-

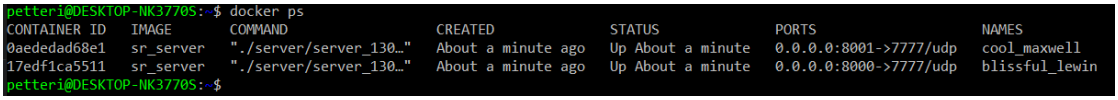
tin porttiin eli porttiin 7777. Ohjelmistokontin pelipalvelin siis käyttää alijärjestelmän porttia 8000 kommunikointiin pelaajien kanssa. Portinsiirron jälkeen ohjelmistokontin liikenneprotokollaksi määritellään UDP-protokolla.

Viimeisenä käskyssä määritetään vedos, jota halutaan käyttää. Ohjelmistokontti käynnistetään aikaisemmin luodun ”sr_server”-vedoksen pohjalta.

Seuraavaksi testataan ohjelmistokonttien toimintaa. Ensimmäisenä Docker-vedosmallista käynnistetään kaksi ohjelmistokontti-instanssia, jotka käyttävät eri portteja (Kuva 23). Ohjelmistokonttien tiedot voidaan tulostaa terminaaliin komennolla `docker ps` (Kuva 24). Kuvassa nähdään, että ohjelmistokontit ovat käynnistyneet onnistuneesti.



Kuva 23. Kaksi terminaalissa luotua ohjelmistokonttia, jotka kuuntelevat portteja 8000 ja 8001



Kuva 24. Ohjelmistokonttien tulostaminen terminaaliin ”docker ps” -komennolla

Ohjelmistokonttien toimintaa testataan samalla tavalla, kuin aikaisempaa pelipalvelinta. Asiakasversiolla liitytään ohjelmistokontin pelipalvelimelle. Pelistä käynnistetään asiakasversio, johon syötetään IP-osoite ja portti. Koska ohjelmistokontit rakennetaan alijärjestelmään, IP-tekstikenttään lisätään tietokoneen määrittelemä alijärjestelmän IP-osoite. IP-osoite löydetään tietokoneen komentokehotteesta komennolla `ipconfig` WSL-osion alta (Kuva 25). Portin tekstikentän arvoksi annetaan alijärjestelmän portti, joka siirtää liikenteen ohjelmistokontin pelipalvelimelle. Portit tulostetaan terminaali-ikkunaan komennolla `docker ps` (Kuva 26).

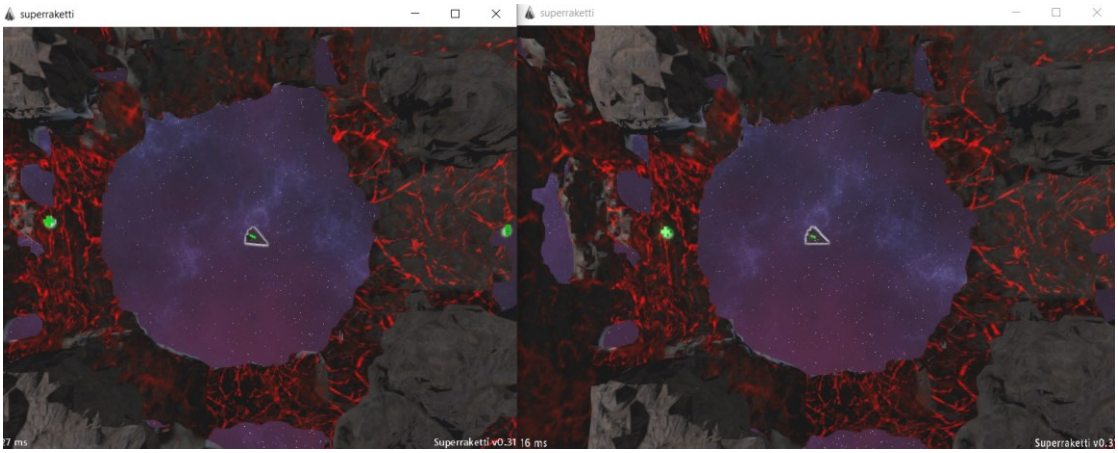
```
Ethernet adapter vEthernet (WSL):

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::8d1a:35ae:1275:9a%43
   IPv4 Address. . . . . . . . . . . : 172.29.176.1
   Subnet Mask . . . . . . . . . . . : 255.255.240.0
   Default Gateway . . . . . . . . . :
```

Kuva 25. WSL:n IP-osoite tulostettuna komentokehotteeseen

```
PORTS
0.0.0.0:8001->7777/udp
0.0.0.0:8000->7777/udp
```

Kuva 26. Ohjelmistokonttien portit

Asiakasversiossa liitytään peliin painamalla ”Join game” -painiketta. Ohjelmistokontti toimii oikein, jos pelaaja liittyy peliin. Sama testi suoritetaan myös toiselle ohjelmistokontille. Kuva 27 kaksi asiakasversiota on liittynyt eri ohjelmistokonttien pelipalvelimille. Lopuksi kontit pysäytetään ja poistetaan Kuva 28 mukaisesti `docker stop` ja `docker rm` komennoilla.



Kuva 27. Kaksi pelaajaa liittyneenä eri ohjelmistokonttien pelipalvelimille

```
petteri@DESKTOP-NK3770S:~$ docker stop cool_maxwell blissful_lewin
cool_maxwell
blissful_lewin
petteri@DESKTOP-NK3770S:~$ docker rm cool_maxwell blissful_lewin
cool_maxwell
blissful_lewin
petteri@DESKTOP-NK3770S:~$
```

Kuva 28. Ohjelmistokontit pysäytetään ja poistetaan komennoilla

5.1.3 Kubernetes

Seuraavaksi lähdetään testaamaan Kubernetes-järjestelmää. Kubernetes tarjoaa alustan ohjelmistokonteille, josta niitä voidaan hallita. Kubernetes-työkalujen käyttö myös helpottaa taustajärjestelmän työnkulun kehitystä. Tavoitteena on testata pelipalvelimellisten ohjelmistokonttien toimintaa ja hallintaa lokaalissa Kubernetes-ympäristössä.

Kubernetes ladataan terminaalissa komennolla `curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl`. Kuberneteksen käyttöön terminaalissa tarvitaan ”kubectl”-niminen työkalu. Se mahdollistaa Kubernetes-komentojen ajamisen terminaalissa. Työkalu ladataan ja asennetaan komennolla `sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl`. Lopuksi kubectl-työkalun varmistetaan Kubernetes-komennon `kubectl cluster-info` avulla. Jos terminaaliin tulostuu linkkejä, työkalu on asennettu oikein (Kuva 29).

```
petteri@DESKTOP-NK3770S:~$ kubectl cluster-info
Kubernetes control plane is running at https://kubernetes.docker.internal:6443
CoreDNS is running at https://kubernetes.docker.internal:6443/api/v1/namespaces/kube-

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
petteri@DESKTOP-NK3770S:~$
```

Kuva 29. Klusterin tierojen tulostus terminaaliin

Kuberneteksen käyttöön lokaalisti vaaditaan Minikube-niminen kirjasto. Minikube luo lokaalin klusterin, jonka pohjalle Kubernetes-komponentit rakennetaan. Minikube soveltuu vain yhden lokaalin klusterin testikäyttöön. Tarkoituksena on testata pelipalvelimien toimintaa klusterin ympäristössä.

Minikube ladataan komennolla `curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64` ja asennetaan komennolla `sudo install minikube-linux-amd64 /usr/local/bin/minikube`. Latauksen ja asennuksen jälkeen Minikube käynnistetään `Minikube start` -komennolla. Minikube luo uuden klusterin, johon on valmiiksi integroitu Docker-kirjasto.

Seuraavaksi luodaan Kubernetes-kapseli, johon rakennetaan aikaisemmin luotu ohjelmistokontti. Kapseli luodaan YAML-tiedoston pohjalta, joka on sisällöltään saman tyylinen, kuin aikaisemmin luotu Dockerfile-tiedosto. YAML-tiedostoon kirjoitetaan asetukset, joiden pohjalta kapseli rakennetaan. Kapselin asetuksiin sisältyy esimerkiksi ajonaikaiset asetukset sekä ohjelmistokontin vedosmalli. YAML-tiedosto kapseleille luodaan terminaalissa komennolla `touch pod.YAML`.

Kapseleiden pod.YAML tiedostoon lisätään editorissa Kuva 30 olevat arvot. Kuvaan on numeroitu punaisella tekstillä tärkeimmät arvot, jotka esitellään seuraavaksi.

1. Kubernetes-API versio, jota halutaan käyttää.
2. Kubernetes-komponentin tyyppi. Tässä tilanteessa luodaan kapseli (pod).
3. Kubernetes-kapselille annettava typpinimi. Tyyppinimi on tärkeä useiden kapseleiden hallinnassa.
4. hostNetwork-arvo mahdollistaa osoitteen määrittelemisen kapselille. Sen arvoksi annetaan ”true”, jotta kapselin kanssa voidaan kommunikoida.
5. Kapselille annettava nimi.
6. Docker-ohjelmistokonttivedos, jonka pohjalta kapseli luodaan. Vedoksena käytetään aikaisemmin luotua pelipalvelinvedosta.
7. imagePullPolicy-arvolla määritellään, mistä ohjelmistokontin vedos etsitään. ”Never”-arvo määrittää, että kapselia ei etsitä verkon välityksellä reksiteristä, vaan lokaalista vedoslistasta. Vedos luotiin Minikuben Docker-ympäristöön.
8. Ohjelmistokontin portti, johon kapselin pelaajaliikenne siirretään. Kuten aikaisemmin mainittiin, pelipalvelin käyttää porttia 7777.
9. Ohjelmistokontin käyttämä protokolla. Pelipalvelimet käyttävät UDP-protokollaa kommunikointiin.

Kuva 30. Kubernetes-pelipalvelinkapselin YAML-tiedosto

Kapselia ei voi kuitenkaan vielä rakentaa, koska pelipalvelinohjelmistokontin vedosta ei ole vielä lisätty Minikuben ympäristöön. Minikubessa ajetaan omaa Docker-ympäristöä, josta kapselin vedos haetaan. Seuraavaksi pelipalvelinkontin vedos lisätään Minikuben Docker-ympäristöön. Minikuben Docker-ympäristöön siirrytään komennolla `eval $(minikube docker-env)`, jonka jälkeen ohjelmistokontti rakennetaan samalla tavalla kuin aikaisemmin eli `docker build -t sr_server .` -komennolla.

Seuraavaksi luodaan kapseli Kubernetes-ympäristöön. Kapseli luodaan komennolla `kubectl apply -f pod-YAML`. Komento luo kapselin ”pod.YAML” -tiedoston pohjalta. Kapselit ja niiden tilat tulostetaan ruudulle komennolla `kubectl get pods` (Kuva 31). Jos kapselin status on ”running”, kapselin ja pelipalvelimen rakennus onnistui. Asiakasversiolla ei pysty liittymään pelipalvelimille tässä testissä, koska Kuberneteksen vakiopalvelimet eivät tue pelipalvelimien vaatimaa yhteystyyppiä. Pelaajat tarvitsevat pitkäaikaisen yhteyden samalle pelipalvelimelle, jotta peliä voidaan pelata. Kubernetes-palvelut eivät kuitenkaan pysty välittämään pitkäkestoisia yhteyksiä. Pelipalvelimelle täytyisi luoda suora yhteys, jonka tarjoamiseen vaaditaan Agones-kirjasto. Minikube rakennettiin uudempaan Kubernetes-versioon, joka ei vielä tue Agonesta. Lopuksi kapseli poistetaan komennolla `kubectl delete pod srserver`.

```
petteri@DESKTOP-NK3770S:~$ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
srserver   1/1     Running   0          85s
```

Kuva 31. Kapselin tietojen tulostus komennolla

Viimeisenä testataan Kubernetes-asettelun toimintaa. Asettelun avulla luodaan useita kapseleita, joita hallitaan keskitetysti. Myös asettelu luodaan YAML-tiedoston avulla. Pohjustuksen YAML-tiedosto luodaan komennolla `touch deployment.YAML`. Tiedostoon lisätään Kuva 32 olevat arvot. Kuvaan on numeroitu tärkeät kohdat punaisella värillä, jotka esitellään seuraavaksi.

1. Kubernetes-API versio, jota halutaan käyttää.
2. Kubernetes-komponentin tyyppi. Tässä tilanteessa luodaan Kubernetes-asettelu (deployment).
3. Kubernetes-asettelulle annettava nimi.
4. Kubernetes-asettelulle annettava typpinimi. Tyyppinimeä hyödynnetään hallinnassa.
5. Rakennettavien Kubernetes-komponenttien määrä. Tässä tilanteessa luodaan 3 pelipalvelinkapselia.
6. Kapselille annettava nimi.
7. Docker-ohjelmistokonttivedos, jonka pohjalta kapseli luodaan.
8. imagePullPolicy-arvolla määritellään, mistä ohjelmistokontin vedos etsitään. Never-arvo kertoo, että kapselia ei etsitä verkon välityksellä rekisteristä, vaan lokaalista vedoslistasta. Vedos luotiin Minikuben Docker-ympäristöön.
9. Ohjelmistokontin portti, johon kapselin pelaajaliikenne siirretään. Kuten aikaisemmin mainittiin, pelipalvelin käyttää porttia 7777.
10. Ohjelmistokontin käyttämä protokolla. Pelipalvelimet käyttävät UDP-protokollaa kommunikointiin.

Kuva 32. Kubernetes-asettelun YAML-tiedosto

Kubernetes-asettelu käynnistetään komennolla `kubectl apply -f deployment.YAML`. Asettelu käynnistää 3 kapselia, joissa ajetaan pelipalvelimia. Kapselien status tulostetaan terminaaliin samalla komennolla, kuin aikaisemmassa yksittäisen kapselin testissä (Kuva 33). Kuvasta nähdään, että kaikki kapselit käynnistyivät oikein. Seuraavaksi testataan pohjustuksen hallintaa `kubectl scale` komennon avulla. Komennon avulla skaalataan kapseleiden määrää käynnissä olevassa pohjustuksessa. Kuva 34 komennolla muutetaan kapseleiden määrä kahteen, jonka jälkeen tulostetaan kapseleiden tiedot. Kuvasta huomataan, että yhden kapselin tila muuttui ”Terminating”-tilaan, jonka seurauksena se poistetaan. Asettelu poistetaan komennolla `kubectl delete deployment sr-deployment`.

```
petteri@DESKTOP-NK3770S:~$ kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
sr-deployment-7565b6444b-dnjz5   1/1     Running   0          25s
sr-deployment-7565b6444b-frvv9   1/1     Running   0          25s
sr-deployment-7565b6444b-sqj96   1/1     Running   0          25s
petteri@DESKTOP-NK3770S:~$
```

Kuva 33. Pohjustuksen luomat kapselit tulostettuna terminaaliin

```
petteri@DESKTOP-NK3770S:~$ kubectl scale --replicas=2 deployment sr-deployment
deployment.apps/sr-deployment scaled
petteri@DESKTOP-NK3770S:~$ kubectl get pods
NAME                                READY   STATUS        RESTARTS   AGE
sr-deployment-7565b6444b-dnjz5   1/1     Running       0          31m
sr-deployment-7565b6444b-frvv9   1/1     Terminating   0          31m
sr-deployment-7565b6444b-sqj96   1/1     Running       0          31m
petteri@DESKTOP-NK3770S:~$
```

Kuva 34. Komennolla skaalataan kapseleiden määrää pohjustuksessa

Lopuksi Minikube sammutetaan komennolla `minikube stop`. Edellisten testien perusteella pelipalvelimet toimivat suunnitellusti taustajärjestelmässä. Pelipalvelimia ei kuitenkaan voida julkaista ilman Agones-kirjastoa. Seuraavaksi siirrytään rakentamaan taustajärjestelmää pilvipalvelun ympäristöön Agoneksen kanssa.

5.2 Taustajärjestelmän kehitys pilvipalveluun

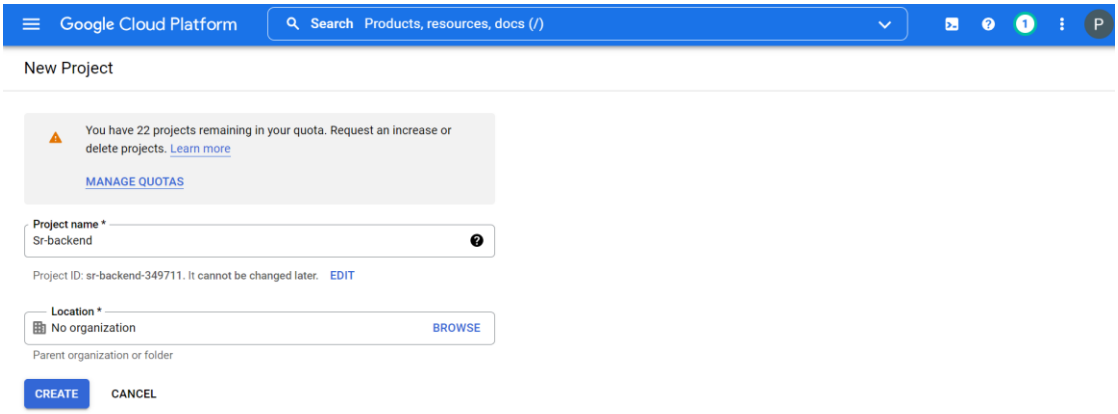
Taustajärjestelmä halutaan rakentaa pilvipalveluun. Pilvipalvelun käytön avulla taustajärjestelmän rakentamisessa vältytään tarpeelta ostaa fyysisiä tietokoneita järjestelmän ylläpitoon, koska tietokoneresurssit ostetaan virtuaalisesti pilvipalvelulta. Lisäksi pilvipalvelun tarjoama kehitysympäristö työkaluineen helpottaa taustajärjestelmän kehitystä. Pilvipalvelun avulla mahdollistetaan taustajärjestelmän skaalautumiskyky ja mahdollistetaan maailmanlaajuinen pelipalvelimien tarjonta.

Pilvipalveluksi valitaan Google Cloud eli Gcloud. Gcloud on Googlen kehittämä pilvipalvelu, joka tarjoaa uusille käyttäjille 300 dollaria ilmaista krediittiä. Krediitti kuluu asiakkaan käyttäessä pilvipalvelun palveluita. Krediitti siis toimii tavallaan pilvipalvelun koeaikana. Krediitin loppuessa asiakas joutuu maksamaan pilvipalvelun toiminnoista.

Gcloud tarjoaa alustan Kubernetes-pohjaisten sovellusten kehitykseen. Gcloudilla on paljon ohjeita ja dokumentteja palveluiden kehittämiseen pilviympäristössä. Tässä työssä käytetään Agones-nimistä ohjelmistokehityspakettia pelipalvelinjärjestelmän kehitykseen. Agones toimii Kuberneteksen lisäosana, joka tarjoaa mukautettuja Kubernetes-komponentteja pelipalvelinpohjaisille taustajärjestelmille. Agoneksen tarjoamat mukautetut komponentit helpottavat taustajärjestelmän kehitystä huomattavasti. Lisäksi Agones mahdollistaa pelipalvelimien ja Kuberneteksen välisen kommunikoinnin.

5.2.1 Pilvipalvelun käyttöönotto

Gcloudin käyttö aloitetaan Gcloudin-verkkokonsolissa, jossa ensimmäiseksi luodaan projekti (Kuva 35). Gcloud-ympäristön hallinta tapahtuu verkkokonsolin tai terminaalin kautta. Tässä työssä käytetään enemmän terminaalia. Terminaalin avulla luodaan tarvittavat Kubernetes-komponentit taustajärjestelmään. Gcloudin terminaali avataan verkkokonsolin oikean yläreunan terminaalipainikkeesta (Kuva 36). Gcloudin terminaali toimii samaan tapaan kuin aikaisemmin käytetty alijärjestelmän terminaali.



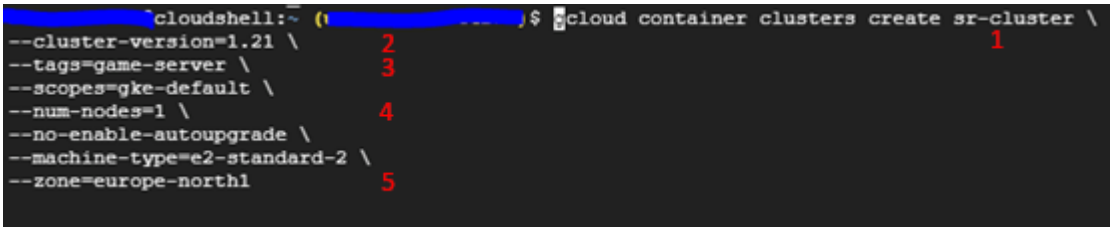
Kuva 35. Gcloudin-verkkokonsolisivu, jossa luodaan projekti

Kuva 36. Gcloud-verkkokonsolin terminaalipainike

Pilvipalvelun taustajärjestelmän kehitys aloitetaan Gcloud-terminaalissa komennolla `gcloud container cluster create`, joka luo pilvipalveluun ohjelmistokonttiklusterin pelipalvelimia varten. Komennolle syötetään asetuk-

sia, joiden pohjalta klusteriin luodaan (Kuva 37). Kuvaan on merkitty punaisella numerot tärkeimpien asetusten rinnalle. Numeroidut asetukset käydään läpi seuraavaksi:

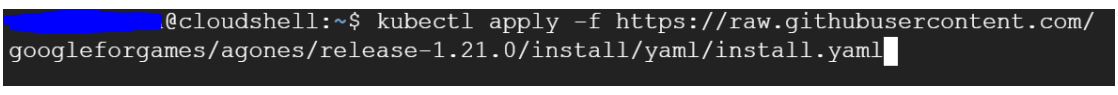
1. Klusterin nimeksi annetaan ”sr-cluster”.
2. Klusterin Kubernetes-versioksi annetaan 1.21, joka tukee viimeisintä Agones versiota.
3. Klusterin tunnisteeksi annetaan ”game-server”. Tunnistetta tarvitaan myöhemmin palomuuriasetuksiin.
4. Klusterin solmujen määrä. Solmu toimii ympäristönä pelipalvelinkapseleille. Arvoksi annetaan 1.
5. Klusterin alueeksi annetaan ”europe-north1”, joka sijaitsee Suomessa. Sijainnin avulla pyritään minimalisoimaan pelaajien viive.



Kuva 37. Pilvipalvelun Kubernetes-klusterin asetukset

Kuva 37 Gcloudin terminaalissa on sinisellä värillä yliviivattuja tekstejä. Sinisellä värillä tuhrataan kaksi arvoa: sähköpostiosoite ja projektin ID-arvo. Arvot tuhrataan varmuuden vuoksi turvallisuussyistä. Jatkossa ajettavien komentojen tuhratut osat sisältävät Gcloud-projektin ID-arvon.

Gcloud-pilvipalvelu hyödyntää pelipalvelinjärjestelmässä Agones-nimistä kirjastoa. Agones koostuu Kubernetes- ja pelipalvelinosista. Taustajärjestelmän osaan sisältyy kaikki Kuberneteksen puolen Agones-komponentit. Pelipalvelimen osaan taas sisältyy pelipalvelimeen asennettava kirjasto, joka kommunikoi taustajärjestelmän Kuberneteksen kanssa. Pelipalvelinosan avulla Kubernetekselle ilmoitetaan esimerkiksi pelipalvelimen tila. Taustajärjestelmän osa ladataan ja asennetaan pilvipalvelun terminaalissa (Kuva 38). Agones-versioksi valitaan 1.21, joka on tekohetkellä uusin Kubernetestä tukeva versio.



Kuva 38. Gcloud-pilvipalveluun lisätään terminaalissa Agones kirjasto

Seuraavaksi luodaan pelipalvelinpiiri. Pelipalvelinpiiri on pelipalvelimille luotujen klusterien ryhmittymä. Piirien avulla pelipalvelimet jaotellaan useisiin

maantieteellisiin osiin palvelun tarjoamiseksi. Piirejä ylläpidetään eri puolilla maailmaa, jotta pelaajien viive voidaan minimoida. Tässä työssä luodaan vain yksi piiri mahdollisimman lähelle. Pelipalvelin luodaan Kuva 39 komennolla. Komennossa sijainniksi annetaan ”europe-north1”, joka sijaitsee Suomessa.

```
gcloud game servers realms create sr-realm --time-zone EET --location europe-north1
```

Kuva 39. Pilvipalveluun pelipalvelinpiirin luominen komennolla

Äsken luotuun pelipalvelinpiiriin rakennetaan klusteri. Klusteri luodaan Kuva 40 komennolla. Komennossa määritellään klusterin nimi, piiri ja sijainti.

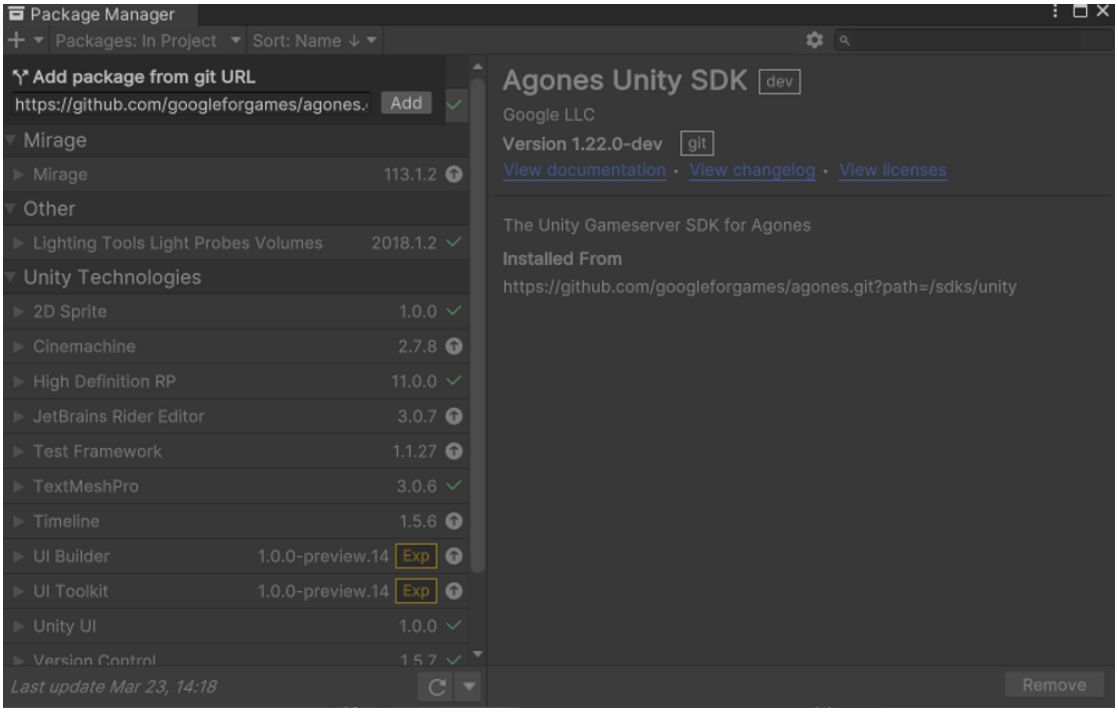
```
[redacted]cloudshell:~ ([redacted])$ gcloud game servers clusters create sr-cluster \
--realm=sr-realm \
--gke-cluster locations/europe-north1/clusters/sr-cluster \
--namespace=default \
--location europe-north1 \
--no-dry-run
```

Kuva 40. Klusteri luodaan komennolla valittuun piiriin

5.2.2 Agones-kirjasto pelipalvelimissa

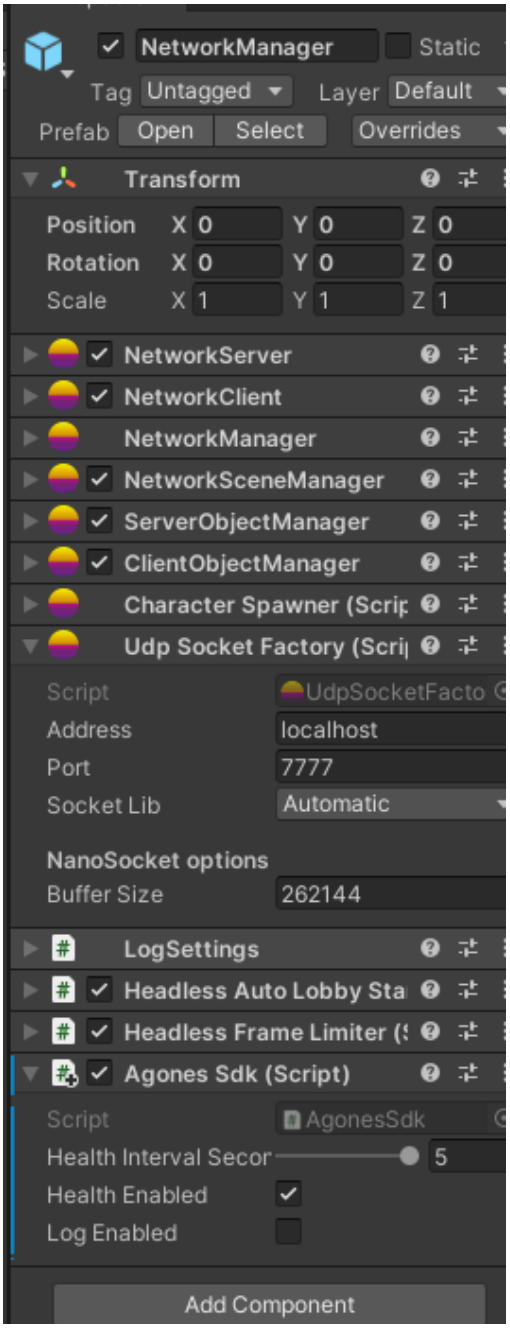
Aikaisemmin mainittiin, että Agones koostuu Kubernetes- ja pelipalvelinpuolen kirjastoista. Seuraavaksi pelipalvelimeen lisätään Agoneksen pelipalvelinpuolen kirjasto. Agoneksen pelipalvelinkirjasto mahdollistaa Kuberneteksen ja pelipalvelimen välisen kommunikoinnin. Sen avulla pelipalvelin voi esimerkiksi ilmoittaa pelipalvelinohjelman virheestä Kubernetekselle, jonka jälkeen Kubernetes käynnistää kapselin uudelleen. Lisäksi skaalaustilanteessa Kubernetes odottaa pelipalvelinpelin loppumiseen saakka, ennen kuin kapseli poistetaan. Pelipalvelimien ja Kuberneteksen välinen kommunikointi siis parantaa pelaajien pelikokemusta ja optimoi resurssien kulutusta.

Agoneksen pelipalvelinkirjasto lisätään pelipalvelimeen Unity-pelieditorissa. Pelieditorissa avataan ”Package Manager” -ikkuna, jossa kirjasto ladataan ja asennetaan URL-osoitteen avulla (Kuva 41). Unity lataa ja asentaa kirjaston pelipalvelimen tiedostoihin, jonka jälkeen Agones-kirjaston metodeja voidaan kutsua koodista. Metodit ovat koodissa ajettavia toimintoja.



Kuva 41. Agones SDK lisätään Unityn package managerissa URL osoitteen kautta

Pelipalvelimen nettilogiikka on lisätty Unity-editorissa ”NetworkManager”-nimiseen olioon. NetworkManager-olio hallitsee pelipalvelimen tilaa, joten Agones-metodit ajetaan siellä. Agones-kirjaston ”Agones sdk” -niminen skripti lisätään ”NetworkManager” olioon (Kuva 42). Skriptin lisääminen olioon mahdollistaa Agones-metodien kutsumisen pelipalvelimen koodista.



Kuva 42. Agones SDK skripti lisättynä ”NetworkManager”-nimiseen olioon Unityn editorissa. NetworkManager-olio hallitsee pelin nettilogiikaa.

NetworkManager-oliossa oleva “Headless Auto Lobby Start” -niminen skripti avataan koodieditorissa. Skriptiin lisätään Kuva 43 näkyvät ”Connect-ToAgones” ja ”ServerReadyForPlayers” -metodit. ConnectoToAgones-metodi luo yhteyden taustajärjestelmän Agones-kirjastoon, joka mahdollistaa kommunikoinnin pelipalvelimen ja Kuberneteksen välillä. ServerReadyForPlayers-metodi taas ilmoittaa Kubernetekselle, että pelipalvelin on valmis ottamaan pelaajia vastaan. Seurauksena Kubernetes julkaisee pelipalvelimen osoitteen, jonka

jälkeen pelaajat pystyvät liittymään pelipalvelimelle. Agones-kirjastolla on muitakin hyödyllisiä metodeja, mutta niitä ei tässä vaiheessa lisätä enempää. Kaikki Agoneksen tarjoamat metodit löytyvät ”agones.dev” -verkkosivuilta.

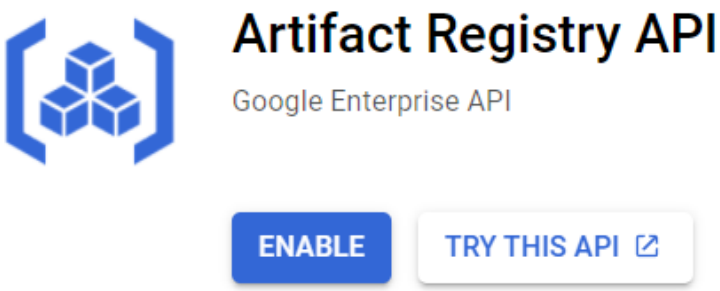
```
//Link to agones sdk unity guide https://agones.dev/site/docs/guides/client-sdks/unity/
1 reference
private async void ConnectToAgones()
{
    bool ok = await agones.Connect();
    Debug.Log("Attempted to connect to agones sdk server, result: " + ok);
}

1 reference
private async void ServerReadyForPlayers()
{
    bool ok = await agones.Ready();
    Debug.Log("Attempted to set server state to receive players, result: " + ok);
}
```

Kuva 43. Nettilogiikan koodiin lisätyt metodit

Lopuksi pelipalvelin rakennetaan Unity-pelieditorissa, jonka jälkeen siitä luodaan vedos ohjelmistokonteille alijärjestelmässä. Tällä kertaa vedoksen nimeksi annetaan ”sr_server_v3”. Pelipalvelimien rakentaminen ja ohjelmistokontin luonti käytiin läpi edellisessä alaluvussa 5.1.

5.2.3 Pilvipalvelun rekisteri

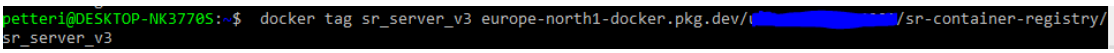
Jotta pelipalvelinohjelmistokontteja voidaan käyttää Kuberneteksessä, niiden vedos täytyy lisätä pilvipalveluun. Gcloud tarjoaa ”Artifact Registry” -nimisen rekisterin, johon ohjelmistokonttien vedokset voidaan tallentaa. Kubernetes hakee vedokset rekisteristä. Gcloudin verkkokonsolissa navigoidaan ”Artifact Registry” -sivulle, jossa rekisteri otetaan käyttöön ”ENABLE”-painikkeella (Kuva 44). Käyttöönoton jälkeen verkkokonsolissa luodaan” sr-container-registry” -niminen rekisteri. Nyt pilvipalvelun rekisteriin voidaan lisätä ohjelmistokontteja.



Kuva 44. Artifact Registry Gcloudin verkkokonsolissa

Pelipalvelimien ohjelmistokonttien lisääminen pilvipalvelun rekisteriin aloitetaan lokaalin alijärjestelmän terminaalissa. Jotta vedos voidaan puskea pilvipalvelun rekisteriin, alijärjestelmään täytyy ladata Gcloud-pilvipalvelun ohjelmistokehityspaketti (Gcloud CLI). Paketin avulla voidaan kirjautua pilvipalveluun sekä ajaa sen komentoja lokaalissa terminaalissa. Gcloud CLI on ladattuna valmiiksi Gcloudin ohjeiden mukaan. Gcloud terminaalityökalu asennusohjeet löytyvät Gcloudin verkkosivuilta (Gcloud docs s.a.).

Pelipalvelimen vedoksen tunnistetta (tag) täytyy muokata, jotta se voidaan puskea pilvipalvelun rekisteriin verkon välityksellä. Vedoksen tunnistetta muutetaan komennolla `docker tag`. Vedoksen tunniste muutetaan Kuva 45 mukaiseksi. Kuvassa tunnisteeksi annetaan ”rekisterin sijainti-docker.pkg.dev/projektin ID/rekisterin nimi/vedos”.



Kuva 45. Vedoksen tunnisteen muutos komennolla

Lopuksi vedos pusketaan pilvipalvelun rekisteriin komennolla `docker push europe-north1-docker.pkg.dev/projektin ID/sr-container-registry/sr_server_v3`.

5.2.4 Pelipalvelinten Kubernetes-asettelu

Kubernetes-asettelussa määritellään pelipalvelinten määrä ja niiden asetukset. Asettelun avulla pelipalvelinten kokonaisuuteen voidaan tehdä deklaratiivisia päivityksiä. Käytännössä asettelu luo pelipalvelinten kokonaisuuden,

jonka kautta niitä voidaan hallita. Kubernetes-asettelua päivitetään YAML-tiedoston pohjalta. Asettelu luodaan pilvipalveluun terminaalissa Kuva 46 komennolla.

```
gcloud game servers deployments create sr-deployment
```

Kuva 46. Komennon avulla luodaan ”sr-deployment” -niminen Kubernetes-asettelu

Seuraavaksi luodaan YAML-pohjatiedosto, joka toimii muokattavana pohjatiedostona konfigurointitiedostoille. Konfigurointitiedostot toimivat Kubernetes-asettelun tiloina. Konfigurointitiedostoissa kirjataan Kubernetes-asettelun asetukset samaan tapaan kuin aikaisemmassa testissä. Pohjatiedostoon siis kirjataan asettelun asetukset, josta konfigurointitiedosto luodaan. Pohjatiedosto luodaan Gcloud-terminaalissa komennolla touch `fleet-config.YAML`, joka luo ”fleet-config.YAML” nimisen tiedoston. Tiedostoa muokataan terminaalissa komennolla `nano fleet-config.YAML`. Tiedostoon lisätään Kuva 47 mukaiset arvot. Tiedostoon on määritelty käynnistettävien pelipalvelinten määrä (replicas), portti asetukset sekä vedoksen osoite. Vedoksen osoitteena toimii rekisteriin lisätyn vedoksen osoite.

```
  GNU nano 5.4
replicas: 2
template:
  metadata:
    labels:
      foo: bar
  spec:
    ports:
    - name: default
      portPolicy: Dynamic
      containerPort: 7777
    template:
      spec:
        containers:
        - name: sr-server
          image: europe-north1-docker.pkg.dev/[redacted]/sr-container-registry/sr_server_v3

^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^C Location
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^_ Go To Line
```

Kuva 47. Kubernetes-asettelun YAML-tiedosto

Konfigurointitiedosto luodaan Kuva 48 komennolla. Komento luo konfigurointitiedoston (config-1) pohjatiedoston (fleet-config) pohjalta. Konfigurointitiedot toimivat Kubernetes-asettelun tiloina. Konfigurointitiedostoja voidaan luoda useita eri tiloille.

```
gcloud game servers configs create config-1 --deployment sr-deployment --fleet-configs-file fleet-config.yaml
```

Kuva 48. Konfigurointitiedosto luodaan pelipalvelimien Kubernetes-asettelua varten

5.2.5 Pelipalvelimien käynnistys

Pelipalvelimet ovat valmiita käynnistettäväksi. Aikaisemmin pilvipalveluun luotiin Kubernetes-asettelu, jonka nimeksi annettiin ”sr-deployment”. Asettelu ei toistaiseksi ylläpidä palvelua, koska sille ei ole määritelty konfigurointitiedostoa. Konfigurointitiedoston ”config-1” konfigurointi asetetaan asettelun tilaksi Kuva 49 komennolla. Kubernetes käynnistää konfigurointitiedostossa määritetyt kaksi pelipalvelinkapselia.

```
gcloud game servers deployments update-rollout  sr-deployment --default-config config-1 --no-dry-run
```

Kuva 49. Terminaalissa Kubernetes-asettelun tilaksi asennetaan ”config-1” -tiedoston tila

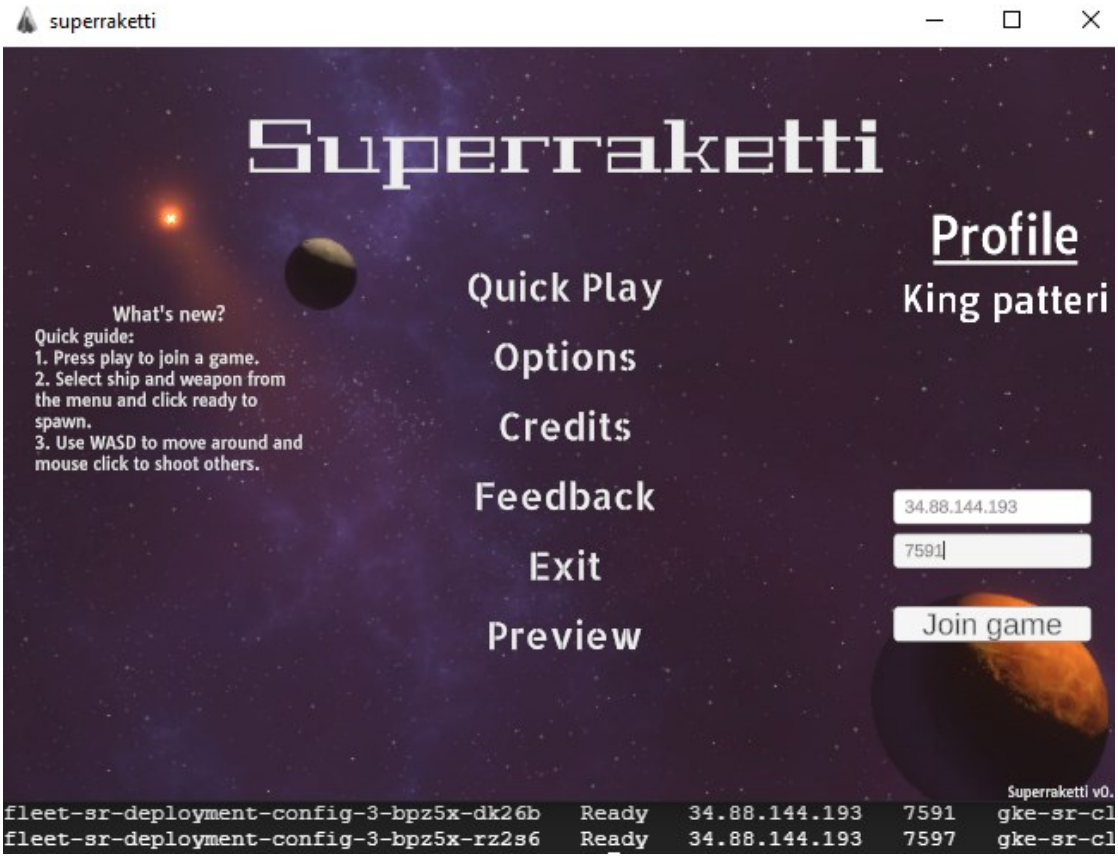
Lopuksi pelipalvelimille täytyy lisätä palomuuriasetukset. Kubernetes estää ulkopuolisen UDP-protokollaliikenteen vakiona. Liikenne pelipalvelimille hyväksytään Kuva 50 komennolla. Nyt pelipalvelimet ovat valmiita pelaajaliikenteelle.

```
[redacted]@cloudshell:~ ([redacted]) $ gcloud compute firewall-rules create sr-gameserver-firewall \
--allow udp:7000-8000 \
--target-tags game-server \
--description "Firewall to allow game server udp traffic"
```

Kuva 50. Pelipalvelimille sallitaan liikenne palomuuriasetuksista

5.2.6 Testaaminen

Kun pelipalvelimet on käynnistetty, niiden toimintaa voidaan testata. Pelistä avataan asiakasversio, jolla pelipalvelimelle liitytään. Pelipalvelimelle liitytään kirjoittamalla asiakasversion tekstikenttiin pelipalvelimen IP-osoite ja portti. Pelipalvelimien tiedot saadaan Gcloud-terminaalissa komennolla `kubectl get gameservers`. Kuva 51 näkyy asiakasversion ikkuna, jonka alapuolelle on tulostettu pelipalvelimien tiedot terminaalissa.



Kuva 51. Pelin asiakasversio, jonka alapuolelle on tulostettu pelipalvelimien tiedot

Pelipalvelimelle liitytään ”Join game” -painikkeella. Jos pelipalvelinten konfigurointi on tehty oikein, pelaaja siirtyy pelipalvelimelle (Kuva 52). Seuraavaksi siirrytään Gcloudin verkkokonsoliin, jossa navigoidaan ”Kubernetes Engine” -sivulle. Sivulla siirrytään ”Applications”-paneeliin, josta valitaan kapseli. Kapselin ”logs”-valikko avaa ruudulle pelipalvelimen lokin. Lokiin tulostetaan pelipalvelimen logiikan tulostukset. Kuva 53 huomataan, että pelipalvelimen lokiin on tulostettu pelaajan liittymisen tiedot. Sama testi tehdään myös toiselle pelipalvelimelle.

Kuva 52. Pelaaja liittyneenä pilvipalvelussa ylläpidetylle pelipalvelimelle onnistuneesti

```
2022-03-18T13:12:37.247117391Z  [PlayerSystem] Pelaaja yhdistynyt.
2022-03-18T13:12:37.247232045Z  {}
2022-03-18T13:12:37.247243981Z  [SystemSignals]OnConnected: uusi pelaaja
2022-03-18T13:12:37.247250749Z  {}
2022-03-18T13:12:40.248421412Z  [PlayerSystem] --- PreSelectState
2022-03-18T13:12:40.248470182Z  {}
2022-03-18T13:12:40.248479662Z  [PlayerSystem] --- NoneState
2022-03-18T13:12:40.248486715Z  {}
2022-03-18T13:12:40.298787590Z  [PlayerSystem] OnPlayerInitMessage Xamk.Superraketti.Network.Messages.PlayerInitMessage
2022-03-18T13:12:40.298837051Z  {}
```

Kuva 53. Pelipalvelimen loki pilvipalvelussa

Pelipalvelimet toimivat toivotulla tavalla pilvipalvelun Kubernetes-ympäristössä. Pelipalvelimien määrää voidaan muuttaa samalla tavalla kuin lokaalissa testiosiossa eli `kubectl scale` -komennolla. Opinnäytetyön toteutusosuus päättyy tähän.

6 TULOKSET

Tässä osiossa käydään läpi kehitystyön tulokset. Osiossa tutkitaan kehitystyön lopputulosta, jota verrataan suunnitelmaan. Samalla myös selvitetään,

käytettiinkö kehitystyössä työkaluja suunnitellulla tavalla. Lisäksi vastataan kehitystyö tutkimusongelmaan ja tutkimuskysymyksiin. Lopuksi käydään läpi kehitysideat ja johtopäätökset.

Opinnäytetyön tavoitteena oli suunnitella ja kehittää taustajärjestelmä Superraketti-pelille. Taustajärjestelmä tarkoittaa ohjelmistopalvelulle kehitettyä järjestelmää, jota käyttäjät eivät suoraan näe, mutta jonka toiminta on palvelulle tärkeää. Taustajärjestelmään voi esimerkiksi kuulua ohjelmistopalvelun logiikka ja palvelun muodostavien komponenttien rakenne. Taustajärjestelmä koostuu usein muistakin toiminnoista, joita tarvitaan palvelun tarjoamiseen.

Tässä tilanteessa taustajärjestelmään oli tarkoitus kehittää toiminto, joka tarjoaa pelipalvelimet asiakkaiden käyttöön internetin välityksellä. Muita pelin vaatimia toimintoja olisivat automaattinen skaalaus sekä pelaajien välittäjä, joista puhutaan enemmän kehitysideat-alaluvussa.

Taustajärjestelmä rakennettiin Gcloud-pilvipalveluun Kubernetes-alustalle. Kuberneteksen avulla luotiin työnkulku, jonka pohjalta pelipalvelimet tarjotaan. Pelipalvelimet toimivat ohjelmistokonteissa. Ohjelmistokontteja ylläpidetään pilvipalvelun Kubernetes-ympäristössä, jossa ne palvelevat pelaajia. Pilvipalvelussa ylläpidetään useita pelipalvelimia saman aikaisesti. Pelaajat liittyvät pelipalvelimille internetin välityksellä.

Lisäksi taustajärjestelmä haluttiin rakentaa pilvipalveluun, jotta pelipalvelimet voidaan tarjota maailmanlaajuisesti. Lisäksi sen avulla vältyttiin tarpeelta hankkia fyysisiä laitteita järjestelmän ylläpitoon.

6.1 Kehitystyön kulku

Kehitystyö aloitettiin luomalla pelipalvelin Unity-pelieditorissa. Samalla Unityssä luotiin pelille asiakasversio, joka toimii pelaajien käyttöliittymänä. Dockerin avulla pelipalvelimelle luotiin ohjelmistokontti. Ohjelmistokonteille luotiin vedostiedosto, jonka pohjalta ne luodaan. Ohjelmistokontit käynnistävät pelipalvelimen automaattisesti kontin luonnin yhteydessä. Pelaajien asiakasversiolla pelipalvelimien sekä ohjelmistokonttien toimintaa testattiin onnistuneesti.

Sen jälkeen siirryttiin testaamaan Kubernetes-ympäristön toimintaa. Kubernetekseen luotiin Kubernetes-klusteri, johon rakennettiin Kubernetes-solmu. Solmu ylläpitää Kubernetes-kapseleita, joissa pelipalvelimet sijaitsevat. Kubernetes-komponentit luotiin YAML-tiedostojen pohjalta. YAML-tiedostoon kirjoitettiin komponentin asetukset. Pelipalvelinohjelmistokonteista luotiin pelipalvelinkapseleita. Pelipalvelinkapselit luotiin Kubernetes-asettelussa, joka toimi tilana pelipalvelinjärjestelmälle. Kapseleita hallittiin alustuksen avulla. Kubernetes-kapseleiden pelipalvelimet toimivat toivotusti, vaikka pelipalvelimien toimintaa ei voitu testata asiakasversiolla. Pelipalvelimet jakamiseen täytyi kuitenkin löytää ratkaisu. Lyhyen tutkinnan jälkeen ongelman ratkaisuksi valittiin Agones-niminen kirjasto. Agones mahdollistaa pelipalvelimien julkaisun ulkomaailmalle Kubernetes-ympäristöstä.

Seuraavaksi siirryttiin pilvipalveluun. Gcloud valittiin Agones-tuen ja sen tarjoamien työkalujen vuoksi. Gcloud-pilvipalveluun luotiin projekti, johon taustajärjestelmä rakennettiin. Projektiin luotiin Kubernetes-piiri ja klusteri. Klusteriin luotiin solmu, jossa ylläpidetään Kubernetes-asettelua. Asettelussa ei vielä ajettu pelipalvelinkapseleita.

Agones-kirjasto lisättiin pilvipalvelun Kubernetes-ympäristöön. Samalla pelipalvelimen koodiin lisättiin Agones-metodeja, jotka mahdollistavat kommunikoinnin taustajärjestelmän Kuberneteksen kanssa. Pelipalvelin ilmoittaa Kubernetekselle, kun se on valmis ottamaan vastaan liikennettä. Pelipalvelin rakennettiin uudelleen, jonka jälkeen siitä luotiin vedos ohjelmistokonteille.

Uusi vedos täytyi puskea pilvipalvelun rekisteriin, jotta taustajärjestelmän Kubernetes pystyi käyttämään sitä. Puskun jälkeen Kubernetekseen luotiin konfigurointitiedosto Kubernetes-asettelua varten. Konfigurointitiedostossa määriteltiin asetukset, jotka muodostivat Kubernetes-palvelun tilan. Konfigurointitiedostoon lisättiin pelipalvelinkapseleiden asetukset, jonka jälkeen asettelu käynnistettiin konfigurointitiedoston avulla. Samalla Kubernetes julkaisi pelipalvelinkapselit Kubernetes-solmun kautta. Pelipalvelinkapseleille määritettiin solmun portit, jota ne käyttivät yhteyden muodostukseen pelaajien kanssa. Lopuksi pilvipalvelussa ajettavia pelipalvelimia testattiin asiakasversiossa onnistuneesti.

Kehitystyön toteutus eteni suunnitteluluvussa kehitetyn suunnitelman mukaisesti. Kehitystyön taustajärjestelmän rakenne vastaa luotua suunnitelmaa (Kuva 10). Työkaluja käytettiin suunnitelman mukaisesti. Suunnitelmaan kuulunut kehitystyön suunnitelman testaus -luku osoittautui merkittäväksi. Suunnitelman testin avulla saatiin tuntuma taustajärjestelmässä käytettäville työkaluille, joka helpotti taustajärjestelmän kehitystä pilvipalveluun. Lisäksi suunnitelman testissä huomattiin, että pelipalvelimien tarjoamiseen vaaditaan kolmannen osapuolen Kubernetes-kirjasto, joka mahdollistaa pelipalvelimien julkaisun. Ongelman ratkaisuksi valittiin Agones-kirjasto.

6.2 Työkalujen käyttö kehitystyössä

Taustajärjestelmän kehitykseen vaaditaan usein työkaluja, joiden avulla haluttu palvelu halutaan kehittää. Kuten aikaisemmin mainittiin, kehitystyön tavoitteena oli luoda pelipalvelinten ylläpitojärjestelmä. Seuraavaksi kerrotaan työkaluista, joita hyödynnettiin taustajärjestelmän kehitystyössä.

Kehitystyössä käytettyjä työkaluja ovat Docker, Kubernetes, Gcloud, Agones ja Unity. Kehitystyöhön valitut työkalut sopivat tavoitteen mukaisen järjestelmän kehitykseen. Kehitystyössä käytettiin työkaluja toivotulla tavalla. Työkalut tukivat toisiaan kehityksessä. Kaikilla työkaluilla on omat tehtävät, joiden pohjalta taustajärjestelmä koostuu.

Unity toimi pelieditorina, jonka avulla pelipalvelimet ja asiakasversiot luotiin.

Dockerin tehtävänä oli pakata pelipalvelin standardoituun ohjelmistokonttiin. Ohjelmistokontti toimi omana pelipalvelinympäristönä, jota oli helppo hallita. Ohjelmistokontin tiedostossa määriteltyjen asetusten ansiosta pakattu pelipalvelin voitiin alustaa. Se osasi esimerkiksi käynnistää pelipalvelimen automaattisesti luonnin yhteydessä.

Kubernetes toimi taustajärjestelmän pelipalvelun pohjana ja pelipalvelinohjelmistokonttien hallintatyökaluna. Kubernetes-alusta mahdollistaa pelipalvelun skaalaamisen sekä keskitetyn hallinnan. Lisäksi se mahdollisti pelipalvelimien julkaisun pelaajille. Kuberneteksen avulla pelipalvelimien osoitteet eriteltiin,

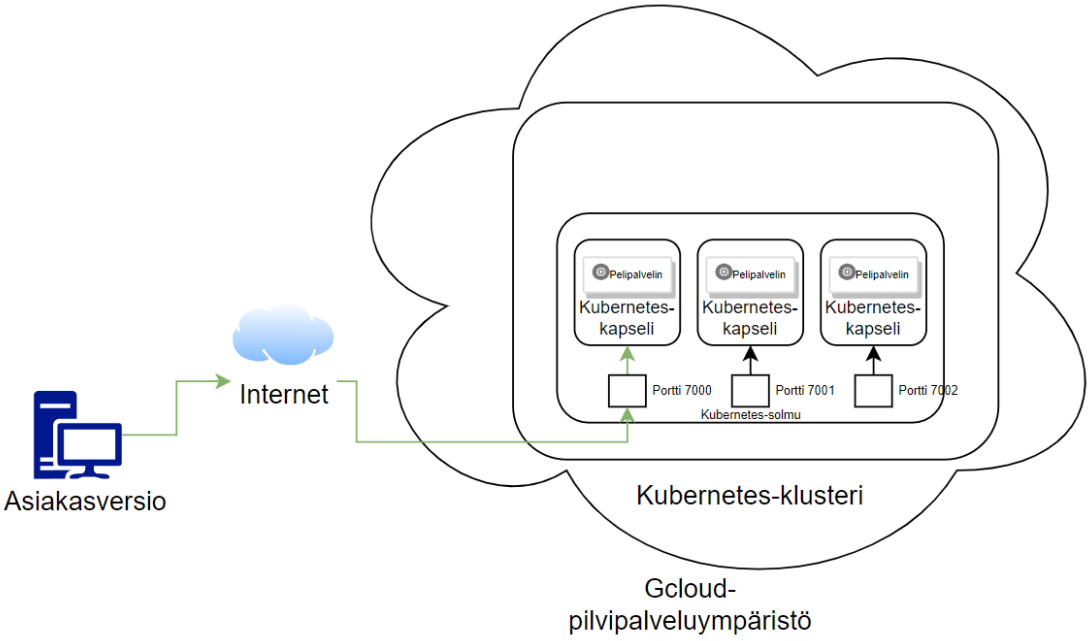
joka mahdollisti useiden pelipalvelimien tarjoamisen saman aikaisesti. Kubernetes avulla luotiin klusteri, johon lisättiin pelipalvelinohjelmistokontteja ylläpitävä solmu. Pelipalvelimet julkaistiin Kubernetes-solmun avulla.

Agones toimii Kuberneteksen lisäosana, joka erikoistuu pelipalvelinpohjaisen Kubernetes-järjestelmän tarjoamiseen. Agones mahdollistaa pelipalvelintilojen kommunikoinnin Kubernetekselle, josta niitä hallitaan. Agoneksen ansiosta pelipalvelin ilmoittaa taustajärjestelmän Kubernetekselle, kun se on valmis vastaanottamaan pelaajia. Pelipalvelimet sijoitettiin Agoneksen tarjoamiin mukautettuihin Kubernetes-pelipalvelinkapseleihin.

Gcloud-pilvipalvelu tarjoaa vakaan infrastruktuuripohjan taustajärjestelmälle. Gcloudin työkalut helpottivat taustajärjestelmän hallintaa ja mahdollistivat pelipalvelimien tarjoamisen verkkoyhteyden välityksellä. Erityisesti Gcloudin tarjoama Kubernetes-alusta oli erityisen tärkeä kehityksen kannalta.

Taustajärjestelmä on yksinkertaisuudessaan rakenteeltaan asiakas–palvelinrakenne. Pelipalvelimet ovat pilvipalvelussa, jossa niitä hallitsee Kubernetes. Pelaajat liittyvät pelipalvelimille internetin välityksellä. Pelipalvelin ja asiakasversio kommunikoivat keskenään pelin tarjoamiseksi.

Kuva 54 kuvataan kehitetyn taustajärjestelmän toimintaa. Pelaaja ottaa asiakasversiolla yhteyden pelipalvelimeen internet-yhteyden välityksellä. Käyttäjä liittyy pelipalvelimeen Kubernetes-solmun osoitteen avulla. Portti määrittelee, mille pelipalvelimelle pelaaja liittyy.



Kuva 54. Kehitystyön taustajärjestelmän rakenne

6.3 Kehitysideat

Taustajärjestelmästä puuttuu välittäjäpalvelin, joka ohjaa pelaajat pelipalvelimille. Lisäksi pelipalvelimien määrää ei skaalata automaattisesti. Taustajärjestelmä on kehitetty tavalla, joka mahdollistaa näiden kahden toiminnon kehittämisen. Pohdintaluvussa käydään asiaa tarkemmin läpi.

Taustajärjestelmä on tähän asti kehitetty pääasiassa toiminnallisuuden kannalta. Sen tietoturvallisuuteen ei ole kiinnitetty erityistä huomiota. Taustajärjestelmän kehityksen aikana pyrittiin välttämään tietoturvaongelmia taitotason mukaan, mutta se ei välttämättä riitä. Taustajärjestelmä voisi hyötyä tietoturvaselvityksestä.

Ohjelmistokontit rakennetaan Ubuntu-pohjalle. Ohjelmistokonttien pohja asennetaan jokaiseen ohjelmistokonttiin. Sen vuoksi ohjelmistokonttien pohjan kannattaa olla kooltaan mahdollisimman pieni. Mitä pienempi ohjelmistokontin pohja on, sitä vähemmän resursseja se vie. Lisäksi pienemmät ohjelmistokontit ovat nopeampia rakentaa. Esimerkiksi Alpine olisi Ubuntua kevyempi pohja ohjelmistokonteille. Ohjelmistokonttien pohjat löytyvät ”Docker Hub” -verkkosivuilta.

Taustajärjestelmän toimintaa testattiin vain muutamilla pelipalvelimilla. Teoreettinen pelaajakapasiteetti on toistaiseksi 8 000 pelaajaa. Klusteriin luotiin yksi solmu, joka avaa 1 000 porttia pelipalvelimille. Toisin sanoen pelipalvelinkapseleita voi olla yhdessä solmussa maksimissaan 1 000 kappaletta. Jokaiseen pelipalvelimeen voi liittyä 8 pelaajaa, eli pelaajakapasiteetti saadaan kertomalla 1 000 pelipalvelinta 8 pelaajalla. Solmut eivät kuitenkaan mahdollisesti jaksa ylläpitää kaikkia 1 000:ta palvelinta, joten niiden ylläpitokykyä voitaisiin testata. Klusteriin voi lisätä solmuja pelaajakapasiteetin lisäämiseksi. Tällä tavalla mahdollistetaan mahdollisimman suuren pelaajamäärän palvelu.

6.4 Johtopäätökset

Taustajärjestelmän toiminnallisessa osaan sisältyi taustajärjestelmän suunnittelu ja toteutus. Teoriaosuuden pohjalta luotiin taustajärjestelmän toiminta- ja rakennesuunnitelma. Toteutusosuudessa suunnitelma pantiin käytäntöön, jonka lopputuloksena syntyi tavoiteltu järjestelmä.

Kehitystyön tuloksen valmistui taustajärjestelmä, jossa ylläpidetään pelipalvelimia. Taustajärjestelmälle asetettuihin sisällöllisiin tavoitteisiin päästiin. Kehitystyössä käytettiin useita eri työkaluja työnkulun helpottamiseksi.

Lisäksi taustajärjestelmä on kehitetty tavalla, joka mahdollistaa jatkokehityksen. Tuloksena syntynyt taustajärjestelmä ylläpitää pelipalvelimia, joissa pelaajat voivat pelata Superraketti-peliä. Taustajärjestelmään voi kuitenkin lisätä muitakin hyödyllisiä palveluita ja toimintoja. Taustajärjestelmän muita mahdollisia jatkotoimintoja käydään luvussa 7.2.

7 POHDINTA

7.1 Merkitys ja prosessi

Taustajärjestelmän tuloksen merkitys on suuri pelin kannalta. Järjestelmä on käyttökelpoinen. Kehitetty taustajärjestelmä tarjoaa pelipalvelimet pelaajien käytettäväksi. Pelaajat siis voivat liittyä pelipalvelimiin verkkoyhteyden välityksellä. Peli on pelityypiltään monipeli, joten pelipalvelimien toiminta verkkoyhteyden välityksellä on tärkeä. Kehitystyön tulosta kuitenkin varjostaa se, että pelaajilla ei ole asiakasversiossa suoraa tapaa etsiä palvelimia. Liittyäkseen

pelaajan täytyy tietää pelipalvelimen IP-osoite ja portti. Taustajärjestelmä ei siis kokonaisuudessaan koostu pelkästään tästä kehitystyöstä. Taustajärjestelmä on rakennettu tavalla, joka mahdollistaa välittäjäpalvelin-järjestelmän kehittämisen samaan ympäristöön jatkossa. Jatkokehitystä käsitellään seuraavassa alaluvussa.

Opinnäytetyön alkuvaiheilla pohdittiin, sisältyvätkö välittäjäpalvelu (matchmaker) ja automaattinen skaalaussysteemi opinnäytetyöhön. Lopuksi niitä ei sisälletty tähän opinnäytetyöhön työn laajuuden vuoksi. Skaalautumiselle sekä välittäjäpalvelulle kuitenkin luotiin alustava pohjasuunnitelma, jota esitellään seuraavaksi.

Opinnäytetyön taustajärjestelmän kehitys onnistui hyvin. Kehitetty taustajärjestelmä vastaa opinnäytetyölle annettuja tavoitteita. Kehitystyössä kehitetyn taustajärjestelmän suunnittelu ja toteutus onnistui yllättävän hyvin. Yksi ongelma, joka tuli vastaan kehitystyön toteutuksen aikana oli pelipalvelinten julkaisu Kubernetes-ympäristössä. Kubernetes-palveluiden avulla ei voitu julkaista pelipalvelimia muulle maailmalle. Tästä johtuen Kubernetes-ympäristöön lisättiin Agones-kirjasto, joka mahdollisti pelipalvelimien julkaisun. Kun kehitystyössä siirryttiin taustajärjestelmän rakentamiseen pilvipalveluun, uusia isoja ongelmia ei tullut ilmi.

Kehitystyössä dokumentoitiin varsin kattavasti. Kehitystyössä käytettiin lähes yksinomaan teoriaosuudessa esiteltyjä menetelmiä ja työkaluja. Kaikilla työkaluilla on perustellut tehtävät, joiden avulla taustajärjestelmän kehitys mahdollistettiin. Toteutusosuudessa esiteltiin kaikki kehityksen vaiheet kuvien tukemana. Kehitystyössä tehdyt valinnat pyrittiin perustelemaan. Toteutusosuuden dokumentoinnin perusteella samanlaisen järjestelmän kehitys on mahdollista myös muille peleille.

7.2 Jatkokehitysmenetelmät

7.2.1 Ideaalinen kehitys

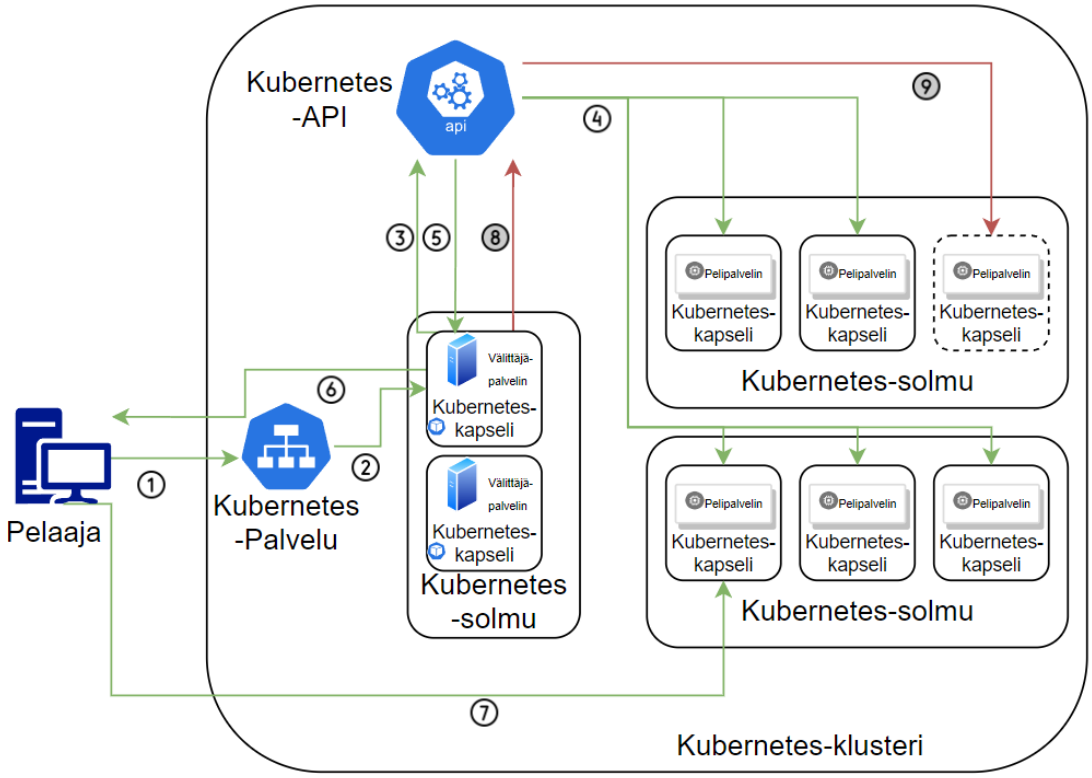
Vaikka taustajärjestelmässä onnistuttiin luoman järjestelmä pelipalvelimien tarjoamiseen, ei se käytännössä ole vielä tarpeeksi kattava. Tämänhetkisestä

taustajärjestelmän versiosta on hyötyä lähinnä rajatulla pelaajamäärällä. Kuten aikaisemmin mainittiin, pelin asiakasversiolla ei ole suoraa tapaa etsiä pelipalvelimia, vaan pelaajan täytyy tietää pelipalvelinten osoitteet. Toistaiseksi ainoastaan Kubernetes tietää pelipalvelimien osoitteet. Ensimmäisenä ongelmana on siis pelaajien ohjaaminen pelipalvelimille. Taustajärjestelmään pitäisi luoda välittäjäpalvelu, jonka avulla pelaajat pystyvät liittymään pelipalvelimille. Välittäjäpalvelun tarkoituksena olisi etsiä vapaat pelipalvelimet ja ohjata pelaajat niihin. Välittäjäpalvelin helpottaisi pelaajien pelikokemusta huomattavasti, koska heidän ei tarvitsisi jatkuvasti etsiä ja kirjoittaa pelipalvelimien osoitteita.

Toisena ongelmana on skaalautuminen. Toistaiseksi pelipalvelinkapseleita täytyy lisätä ja poistaa manuaalisesti. Pelipalvelinkapseleiden luonti- ja poistoprosessit halutaan automatisoida resurssien ja työmäärän minimoimiseksi. Pelipalvelimien skaalaussysteemin ja välittäjäpalvelun kehittäminen loisi itseään ylläpitävän taustajärjestelmän. Lopputavoitteena olisi Kuva 55 mukainen taustajärjestelmä.

Kuva 55 Kubernetes-klusteriin on lisätty useita solmuja pelipalvelimille. Lisäksi on luotu yksi solmu, jossa sijaitsee välittäjäpalvelimet, joilla on kaksi tehtävää. Ensimmäinen tehtävä on etsiä pelipalvelimet ja toinen tehtävä on skaalata pelipalvelu. Kuvassa on numeroitu komponenttien väliset kommunikoinnit. Valkotaustaiset numerot (1–7) kuvaavat liittymisprosessin työnkulkua, kun taas harmaataustaiset (8–9) kuvaavat pelipalvelun skaalautumista. Seuraavaksi listataan numeroiden tehtävät.

1. Pelaaja ottaa yhteyden Kubernetes-palveluun.
2. Kubernetes-palvelu jakaa asiakasliikenteen välittäjäpalvelinkapselien välille.
3. Välittäjäpalvelin pyytää pelipalvelimien osoitteita Kubernetes-API:n kautta.
4. Kubernetes-API etsii pelipalvelimet ja niiden osoitteet.
5. Kubernetes-API palauttaa pelipalvelimien osoitteet välittäjäpalvelimiin.
6. Välittäjäpalvelin valitsee sopivan pelipalvelimen, jonka osoitteen se lähettää pelaajalle.
7. Pelaaja liittyy pelipalvelimelle saadun osoitteen avulla.
8. Välittäjäpalvelin havaitsee, että pelipalvelimia on liikaa tai liian vähän kaikkien pelaajien palvelemiseen, joten se lähettää skaalauspyynnön Kubernetes-API:lle.
9. Kubernetes-API poistaa tai lisää pelipalvelinkapselin.



Kuva 55. Lopullisen taustajärjestelmän Kubernetes-rakenne

7.2.2 Vaihtoehtoinen kehitys

Asiakasversioon voitaisiin tämänhetkisessä versiossa luoda hyvin yksinkertainen liittymissysteemi pelipalvelimelle. Aikaisemmin mainittiin, että taustajärjestelmän pelipalvelinkapselit sijaitsevat samassa Kubernetes-solmussa, jolla on oma staattinen IP-osoitteensa. Pelipalvelinkapselit käyttävät solmun portteja pelipalvelinten julkaisuun. Tämä tarkoittaa sitä, että pelaajat käyttävät toistaiseksi aina samaa IP-osoitetta (solmun IP-osoite) pelipalvelimille liittymiseen. Ainoa vaihtuva arvo siis on portti. Lisäksi pelipalvelimet käyttävät ainoastaan solmun portteja 7000–8000. Käytännössä asiakasversion palvelimen etsintäkoodia voitaisiin muokata tavalla, jossa asiakasversio etsii pelipalvelimia ainoastaan ”kovakoodatusta” solmun IP-osoitteesta. Asiakasversio testaisi solmun portteja 7000–8000 yksitellen, kunnes vapaa palvelin löytyisi. Esitelty ratkaisu ei kuitenkaan olisi vakaa. Ongelmia olisivat mahdolliset pelipalvelimen pitkät etsintäajat, systeemi ei skaalautuisi solmuja lisätessä ja pelipalvelimet eivät välttämättä täyty järjestyksessä.

Taustajärjestelmään voisi myös luoda skaalautumismenetelmän. Taustajärjestelmän pelipalvelimien skaalautuminen tapahtuisi Kubernetes-rajapinnan avulla. Skaalautumista varten voitaisiin rakentaa oma palvelin, joka lisäisi ja

poistaisi pelipalvelinkapseleita tarpeen mukaan. Skaalautumista varten pelipalvelimille kirjoitettaisiin koodinpätkä, joka palauttaisi pelipalvelimen pelaajien määrän. Skaalauspalvelin lähettäisi pelipalvelimille pyyntöjä, jotka palauttaisivat skaalauspalvelimelle tiedon pelaajamäärästä. Pelaajamäärän perusteella palvelin päättelisi, tarvitseeko pelipalvelinkapseleita lisätä vai poistaa. Sen avulla taustajärjestelmän pelipalvelinpalvelu voitaisiin skaalata automaattisesti.

7.3 Teoria ja työkalut

Opinnäytetyön osat tukivat toinen toisiaan hyvin. Teoriaosuudessa käytettyjä menetelmiä ja työkaluja hyödynnettiin kehitystyön suunnitelmassa sekä sen toteutuksessa. Valittujen työkalut toimivat hyvin yhteistyössä toistensa kanssa. Esimerkiksi Gcloud-pilvipalvelu tukee Kubernetes-ympäristöä, joka taas tukee Docker-ohjelmistokonttipohjaisia pelipalvelimia. Työkaluilla siis oli omat tehtävänsä, joista koko järjestelmän toiminta muodostui.

Teorialuvussa käsiteltiin tutkittua taustajärjestelmän yleistermistöä, keskeisiä työkaluja sekä menetelmiä, joista uskottiin olevan hyötyä taustajärjestelmän kehityksessä. Teoriaosuudessa määriteltiin, mikä on taustajärjestelmä ja mistä se koostuu. Sitä seurasi taustajärjestelmärakenteiden esittely. Rakenne-esittelyä seurasi taustajärjestelmän työkalujen esittely. Tässä työssä esiteltiin vain muutamia tavoitteiden saavuttamiseen vaadittavia työkaluja. Todellisuudessa taustajärjestelmä voi koostua hyvin useista eri komponentista erilaisiin taustajärjestelmiin. Lopuksi esiteltiin virtualisointimenetelmät, orkestrointi sekä pilvipalvelut. Kaikilla teoriaisuudessa esitellyillä asioilla oli jollain tavalla merkitystä kehitystyössä.

Työkaluja esiteltiin eritoten Kuberetes-pohjaisten järjestelmien näkökannasta. Syynä Kuberneteksen suosimiseen on sen rakennemahdollisuuksissa. Kehitystyössä keskityttiin pelipalvelimien tarjoamiseen, mutta jatkokehityksen vuoksi skaalautuminen otettiin huomioon. Teoriaosuudessa selitettiin eri skaalaussuunnat. Kubernetes-pohja mahdollistaa horisontaalisen skaalautuvuuden taustajärjestelmässä Kubernetes-solmujen ja kapseleiden avulla. Klusteriin pystytään aina lisäämään solmuja, joissa ylläpidetään yksittäisiä palvelimia

kapseleissa. Solmuja ja niiden kapseleita voidaan lisätä ja poistaa tarvittaessa. Tämäntyyppinen skaalaussysteemi pystyy laajentumaan lähes loputtomiin, joka mahdollistaa minkä tahansa pelaajamäärän palvelun resurssitehokkaasti. Resurssitehokkuus on erittäin tärkeä osa taustajärjestelmää, koska taustajärjestelmän kulut riippuvat resurssien kulutuksesta.

Mainittakoon, että kehitystyössä kehitetyn taustajärjestelmän rakennetta kutsutaan mikropalveluksi (microservice). Mikropalvelurakenteiset ohjelmistot koostuvat useista erinäisistä itsenäisistä toiminnoista palvelun tarjoamiseksi. Tässä tilanteessa jokainen Kubernetes-kapseli toimii erillisenä pelipalvelimella. Lisäksi Kubernetes-solmut toimivat erillään toisistaan. (Kontsevoy 2021.)

7.4 Agones ja Gcloud

Agones on toistaiseksi työn alla oleva työkalu. Sen kaikki toiminnot eivät koske Unity-pohjaisia pelejä toistaiseksi (Kuva 56). Jatkotoimien kannalta olisi hyödyllistä, että kaikkia Agoneksen pelipalvelinpuolen kirjaston metodeja tuettaisiin Unity-editorissa. Agones ei vielä tue opinnäytetyön tekohetkellä Unity-pelien ”Player Tracking” -metodeja. Kyseiset metodit olisivat tosi hyödyllisiä pelipalvelimien skaalautumista ajatelleen. Käytännössä ne mahdollistavat pelipalvelimen pelaajamäärän tietojen ilmoittamisen Kubernetes-järjestelmälle. Kubernetes pystyisi pelaajamäärän perusteella skaalaamaan pelipalvelimien määrää.

Agoneksen ja Gcloudin toiminnassa on yksi ongelma, joka täytyy ratkaista ennen kuin taustajärjestelmän pelaajavälittäjäpalvelin voidaan kehittää. Pelipalvelimet etsitään Agoneksen luoman mukautetun Kubernetes-API:n komennon avulla, koska ne ovat Agoneksen luomia mukautettuja Kubernetes-kapseleita. Ongelmana on se, että Gcloudin Kubernetes-API:n avulla ei pystytä etsimään näitä mukautettuja pelipalvelin kapseleita. Välittäjäpalvelimen pitäisi siis saada yhteys Agoneksen API:iin, jotta pelipalvelimet voidaan etsiä välittäjäpalvelimessa.

SDK Functionality

Area	Action	Implemented
Lifecycle	Ready	✔
Lifecycle	Health	✔
Lifecycle	Reserve	✔
Lifecycle	Allocate	✔
Lifecycle	Shutdown	✔
Configuration	GameServer	✔
Configuration	Watch	✔
Metadata	SetAnnotation	✔
Metadata	SetLabel	✔
Player Tracking	GetConnectedPlayers	✘
Player Tracking	GetPlayerCapacity	✘
Player Tracking	GetPlayerCount	✘
Player Tracking	IsPlayerConnected	✘
Player Tracking	PlayerConnect	✘
Player Tracking	PlayerDisconnect	✘
Player Tracking	SetPlayerCapacity	✘

Kuva 56. Agones-kirjaston tuki Unity-peleille

Gcloud on palveluna erinomainen opinnäytetyön kehitystyön tyyppisen taustajärjestelmän kehittämiseen. Gcloudin tarjoamat työkalut helpottivat taustajärjestelmän kehitystä huomattavasti. Lisäksi Gcloud myös mahdollistaa taustajärjestelmän jatkokehityksen. Gcloudin ongelma on se, että kattava pilvipalvelu tulee myös kattavilla laskuilla. Gcloudin kuluja seurattiin kehitystyön edetessä. Taustajärjestelmän ylläpitoon kului noin 7 euroa päivässä. Kulut sinänsä eivät ole hirveän korkeat, mutta halvempiakin vaihtoehtoja löytyy. Koko kehitystyön ajan kulutettiin Gcloudin tarjoaman koeajan krediittiä rahan sijaan. Muiden pilvipalveluiden ongelmana on niiden dokumentoinnin epäselvyys sekä toimintojen puute. Yksi suurimmista syistä Gcloudin valintaan on sen tarjoama Agones-tuki. Muita vaihtoehtoja ei kuitenkaan kannata sulkea pois.

LÄHTEET

Agones.dev. 2022. Overview. WWW-dokumentti. Saatavissa: https://agones.dev/site/docs/overview/ [Viitattu 16.5.2022].

Castsoftware. s.a. What Is Software Architecture?. WWW-dokumentti. Saatavissa: https://www.castsoftware.com/glossary/what-is-software-architecture-tools-design-definition-explanation-best [Viitattu 16.5.2022].

CIO Wiki. s.a. Client Server Architecture. WWW-dokumentti. Saatavissa: https://cio-wiki.org/wiki/Client_Server_Architecture#What_is_Client_Server_Architecture.3F [Viitattu 28.11.2021].

Cloudzero. 2021. Horizontal Vs. Vertical Scaling: How Do They Compare?. WWW-dokumentti. Saatavissa: https://www.cloudzero.com/blog/horizontal-vs-vertical-scaling [Viitattu: 9.12.2021].

Codeacademy. 2019. What is back-end?. WWW-video. Saatavissa: https://www.youtube.com/watch?v=WwbBOQaM0Zw [Viitattu 18.11.2021].

geeksforgeeks.com. Nimimerkki: sam816g. 2020. WWW-dokumentti .Difference between Virtual Machines and Containers. Saatavissa: https://www.geeksforgeeks.org/difference-between-virtual-machines-and-containers/ [Viitattu 28.11.2021].

IBM. 2022. Internet Transport-Level Protocols. WWW-dokumentti. Saatavissa: https://www.ibm.com/docs/en/aix/7.1?topic=protocols-internet-transport-level [Viitattu: 11.5.2022].

IBM Cloud Team, IBM Cloud. 2020. Top 7 Most Common Uses of Cloud Computing. WWW-dokumentti. Saatavissa: https://www.ibm.com/cloud/blog/top-7-most-common-uses-of-cloud-computing [Viitattu 7.12.2021].

Jena, S. 2022. Difference between IP address and Port Number. WWW-dokumentti. Saatavissa: https://www.geeksforgeeks.org/difference-between-ip-address-and-port-number/ [Viitattu: 2.5.2022].

Kononenko, Kevin. 2018. Front End v. Back End Explained by Waiting Tables At A Restaurant. WWW-dokumentti. Saatavissa: https://blog.codeanalogies.com/2018/04/07/front-end-v-back-end-explained-by-waiting-tables-at-a-restaurant/ [Viitattu 18.11.2021].

Kontsevoy, Ev. 2021. Microservices, Containers and Kubernetes in 10 minutes. WWW-dokumentti. Saatavissa: https://goteleport.com/blog/microservices-containers-kubernetes/ [Viitattu: 16.5.2022].

Kubernetes.io. 2022a. What is Kubernetes?. WWW-dokumentti. Saatavissa: https://Kubernetes.io/docs/concepts/overview/what-is-Kubernetes/ [Viitattu: 11.5.2022].

Kuberneters.io. 2022b. Pods. WWW-dokumentti. Saatavissa: https://Kubernetes.io/docs/concepts/workloads/pods/ [Viitattu: 2.5.2022].

Kubernetes.io. 2022c. Nodes. WWW-dokumentti. Saatavissa: https://Kubernetes.io/docs/concepts/architecture/nodes/ [Viitattu: 11.5.2022].

Kuberneters.io. 2022d. Service. WWW-dokumentti. Saatavissa: https://Kubernetes.io/docs/concepts/services-networking/service/ [Viitattu: 2.5.2022].

Kuberneters.io. 2022e. Deployments. WWW-dokumentti. Saatavissa: https://Kubernetes.io/docs/concepts/workloads/controllers/deployment/ [Viitattu: 2.5.2022].

opensource.com. s.a. What is virtualization?. WWW-dokumentti. Saatavissa: https://opensource.com/resources/virtualization [Viitattu 19.11.2021].

Posey, B. 2021. What is a Server?. WWW-dokumentti. Saatavissa: https://www.techtarget.com/whatis/definition/server [Viitattu: 22.4.2022].

Red Hat. 2018. What is virtualization?. WWW-dokumentti. Saatavissa: https://www.redhat.com/en/topics/virtualization/what-is-virtualization [Viitattu 19.11.2021].

Red Hat. 2019a. What are cloud services?. WWW-dokumentti. Saatavissa: https://www.redhat.com/en/topics/cloud-computing/what-are-cloud-services [Viitattu: 7.12.2021].

Red Hat. 2019b. What is orchestration?. WWW-dokumentti. Saatavissa: https://www.redhat.com/en/topics/automation/what-is-orchestration [Viitattu: 11.5.2022].

Section.io. 2020. Scaling Horizontally vs. Scaling Vertically. WWW-dokumentti. Saatavissa: https://www.section.io/blog/scaling-horizontally-vs-vertically/ [Viitattu: 9.12.2021].

Section.io. 2021. Why is Docker so Popular. WWW-dokumentti. Saatavissa: https://www.section.io/engineering-education/why-is-docker-so-popular/ [Viitattu 11.11.2021].

Simic, S. 2021. Linux vs. Windows Server: The Ultimate Comparison. WWW-dokumentti. Saatavissa: https://phoenixnap.com/blog/linux-vs-microsoft-windows-servers [Viitattu: 11.5.2022].

Wallenius, N. 2019. Konttiteknologia – mitä kontit ovat ja mitä hyötyä niistä on?. WWW-dokumentti. Saatavissa: https://niklaswallenius.fi/konttiteknologia-mita-hyotya/ [Viitattu 11.11.2021].

Watts, S & Raza, M. 2019. SaaS vs PaaS vs IaaS: What's The Difference & How To Choose. WWW-dokumentti. Saatavissa: https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/ [Viitattu 7.12.2021].

What is a Kubernetes cluster?. s.a. VMware. WWW-dokumentti. Saatavissa: https://www.vmware.com/topics/glossary/content/Kubernetes-cluster.html [Viitattu: [11.5.2022].

What is a virtual machine?. s.a. VMware. WWW-dokumentti. Saatavissa: https://www.vmware.com/topics/glossary/content/virtual-machine [Viitattu 28.11.2021].

KUVALUETTELO

Kuva 1. Ravinto-analogian rakenne. Saatavissa: https://blog.codeanalogies.com/2018/04/07/front-end-v-back-end-explained-by-waiting-tables-at-a-restaurant/.

Kuva 2. Yksinkertainen asiakas–palvelin-rakenne. Saatavissa: https://en.wikipedia.org/wiki/Client%E2%80%93server_model.

Kuva 3. Esimerkki tarkasti kuvatusta ohjelmistopalvelun rakenteesta. Kuvassa yläpuolen käyttöliittymä ja alapuolen taustajärjestelmä on jaettu vaakasuuntaisella katkoviivalla. Komponentit on kuvattu erivärisillä laatikoilla ja lieriöillä. Nuoliviivat kuvaavat komponenttien välisiä sidoksia. Saatavissa: https://litslink.com/blog/web-application-architecture.

Kuva 4. Asiakkaan palvelupyyntö IP-osoitteilla. Palvelimen IP-osoitteen perässä eritelty portti ":"-merkillä. Kuljetuskerroksena toimii UDP-protokolla.

Kuva 5. Konttijärjestelmän rakennekaavio. Saatavissa: https://blog.netapp.com/blogs/containers-vs-vms/.

Kuva 6. Virtuaalikoneen rakenne. Saatavissa: https://blog.netapp.com/blogs/containers-vs-vms/.

Kuva 7. Kubernetes-rakenne yksinkertaisuudessaan.

Kuva 8. Pilvipalveluiden skaalaussuunnat. Saatavissa: https://www.section.io/blog/scaling-horizontally-vs-vertically/.

Kuva 9. Pelipalvelimen ympäristörakenne.

Kuva 10. Taustajärjestelmän rakenne pilvipalvelussa.

Kuva 11. Ubuntu Microsoft Storessa.

Kuva 12. Pelipalvelimen rakennus Unity-pelieditorissa.

Kuva 13. ”gameserver” kansion tulostaminen terminaaliin.

Kuva 14. Pelipalvelimen käynnistyksen loki.

Kuva 15. Windows-pohjaisen asiakasversion asetukset.

Kuva 16. Asiakasversion rakennettu kansio.

Kuva 17. Asiakasversion päävalikko.

Kuva 18. IP-osoitteiden haku komennolla terminaalissa.

Kuva 19. Asiakasversio liittyneenä palvelimelle.

Kuva 20. Asiakasversion virheilmoitus.

Kuva 21. Pelipalvelimen Dockerfile-tiedosto.

Kuva 22. Kuvassa listattu docker image ja sen tiedot.

Kuva 23. Kaksi terminaalissa luotua ohjelmistokonttia, jotka kuuntelevat portteja 8000 ja 8001.

Kuva 24. Ohjelmistokonttien tulostaminen terminaaliin ”docker ps” -komennolla.

Kuva 25. WSL:n IP-osoite tulostettuna komentokehotteeseen.

Kuva 26. Ohjelmistokonttien portit.

Kuva 27. Kaksi pelaajaa liittyneenä eri ohjelmistokonttien pelipalvelimille.

Kuva 28. Ohjelmistokontit pysäytetään ja poistetaan komennoilla.

Kuva 29. Klusterin tierojen tulostus terminaaliin.

Kuva 30. Kubernetes-pelipalvelinkapselin YAML-tiedosto.

Kuva 31. Kapselin tietojen tulostus komennolla.

Kuva 32. Kubernetes-asettelun YAML-tiedosto.

Kuva 33. Pohjustuksen luomat kapselit tulostettuna terminaaliin.

Kuva 34. Komennolla skaalataan kapseleiden määrää pohjustuksessa.

Kuva 35. Gcloudin-verkkokonsolisivu, jossa luodaan projekti.

Kuva 36. Gcloud-verkkokonsolin terminaalipainike.

Kuva 37. Pilvipalvelun Kubernetes-klusterin asetukset.

Kuva 38. Gcloud-pilvipalveluun lisätään terminaalissa Agones kirjasto.

Kuva 39. Pilvipalveluun pelipalvelinpiirin luominen komennolla.

Kuva 40. Klusteri luodaan komennolla valittuun piiriin.

Kuva 41. Agones SDK lisätään Unityn package managerissa URL osoitteen kautta.

Kuva 42. Agones SDK skripti lisättynä ”NetworkManager”-nimiseen olioon Unityn editorissa. NetworkManager-olio hallitsee pelin nettilogiikaa.

Kuva 43. Nettilogiikan koodiin lisätyt metodit.

Kuva 44. Artifact Registry Gcloudin verkkokonsolissa.

Kuva 45. Vedoksen tunnisteen muutos komennolla.

Kuva 46. Komennon avulla luodaan ”sr-deployment” -niminen Kubernetes-asettelu.

Kuva 47. Kubernetes-asettelun YAML-tiedosto.

Kuva 48. Konfigurointitiedosto luodaan pelipalvelimien Kubernetes-asettelua varten.

Kuva 49. Terminaalissa Kubernetes-asettelun tilaksi asennetaan ”config-1” -tiedoston tila.

Kuva 50. Pelipalvelimille sallitaan liikenne palomuuriasetuksista.

Kuva 51. Pelin asiakasversio, jonka alapuolelle on tulostettu pelipalvelimien tiedot.

Kuva 52. Pelaaja liittyneenä pilvipalvelussa ylläpidetylle pelipalvelimelle onnistuneesti.

Kuva 53. Pelipalvelimen loki pilvipalvelussa.

Kuva 54. Kehitystyön taustajärjestelmän rakenne.

Kuva 55. Lopullisen taustajärjestelmän Kubernetes-rakenne.

Kuva 56. Agones-kirjaston tuki Unity-peleille.