LAB University of Applied Sciences

Koneoppimistyökalut kontti ympäristössä

Case: ITKO-hanke

LAB-ammattikorkeakoulu
Insinööri (AMK), Tieto- ja viestintätekniikka Ohjelmistotekniikka
2021
Janne Alaranta

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Alaranta, Janne	Opinnäytetyö, AMK	2021
	Sivumäärä	
	22	

Työn nimi

Koneoppimistyökalut kontti ympäristössä

Case: ITKO-hanke

Tutkinto

Insinööri (AMK)

Toimeksiantajan nimi, titteli ja organisaatio

ITKO, LAB Ammattikorkeakoulu

Tiivistelmä

Opinnäytetyössä tavoitteena oli toteuttaa koneoppimis- ja tekoälysovellusten käyttöönotto virtuaalisessa ympäristössä hyödyntäen kiihdytettyä laskentaa. Opinnäytetyö tehtiin osana LAB-ammattikorkeakoulun ITKO-hanketta, jonka tarkoituksena on auttaa kehittämään Päijät-Hämeen alueen pienten ja keskisuurien yrityksien IoT, Big Datan ja koneoppimisen hyödyntämismahdollisuuksia.

Opinnäytetyön teoriaosuudessa tutustutaan koneoppimisessa tarvittaviin teknologioihin, sekä kuinka asiakkaat pääsevät käyttämään kiihdytettyjä resursseja. Teoria osuudessa käydään myös läpi, kuinka kiihdytettyjä resursseja jaetaan.

Käytännön osuudessa perustetaan sovellusalusta kontti ympäristöön. Konttien avulla luodaan monipuolinen ympäristö, mikä koostuu useasta yhdessä toimivasta sovelluksesta. Konttiympäristön avulla saadaan luotua joustava alusta, jolla voidaan ottaa käyttöön valmiita koneoppimistyökaluja kiihdytetyssä laskentaympäristössä.

Lopputuloksena saatiin toteutettua virtuaalinen ympäristö, jossa kiihdytettyä laskentaa hyödyntämällä voidaan suorittaa koneoppimista tai tekoälymallien kouluttamista. Näitä työkaluja käyttämällä voidaan analysoida, sekä visualisoida IoT pipelinessä kerättyä dataa.

Asiasanat

Koneoppiminen, Tekoäly, Kontti, Kiihdytetty laskenta

Abstract

Author(s)	Type of Publication	Published
Alaranta, Janne	Thesis, UAS	2021
	Number of Pages	
	22	

Title of Publication

Machine learning tools in container environment

Case: ITKO Project

Name of Degree

Engineer (UAS)

Name, title and organization of the client

ITKO, LAB University of Applied Sciences

Abstract

The goal in this thesis was to implement the introduction of machine learning and artificial intelligence software in a virtual environment by utilizing accelerated computing. This thesis was donea as part of LAB University of Applied Sciences' ITKO project, the purpose of which is to help develop the utilization of IoT, Big Data and machine learning for small and medium sized enterprises in Päijät-Häme region.

The theoretical part of this thesis introduces technologies needed in machine learning, as well as how customers can access accelerated resources. This part also covers how those accelerated resources are distributed.

In the practical part, the application platform is set up in a container environment. Containers create a versatile environment that consists of several application that work together. The container environment provides a flexible platform for deploying ready-made machine learning tools in an accelerated computing environment.

The result of this thesis was a virtual environment in which machine learning or training of artificial intelligence models can be performed utilizing accelerated computing. Using these tools, the data collected in the IoT pipeline can be analyzed and visualized.

Keywords

Machine learning, Artificial intelligence, Containers, Accelerated computing

Sisällys

1 Johdanto ... 3
2 Toimintaympäristö ... 4
2.1 Yleiskuva ... 4
2.2 Palvelin ... 4
2.3 Alustetut virtuaalikoneet ... 5
3 Teknologiat ... 7
3.1 NVIDIA A100 (MIG) ... 7
3.1.1 Tensor ydin ... 7
3.1.2 MIG ... 9
3.2 Docker ... 11
3.2.1 Virtuaalikoneiden ja konttien eroavaisuus ... 12
3.2.2 Kontti ... 12
3.3 Käänteinen välityspalvelin ... 13
3.4 Jupyter ... 14
4 Resurssien allokointi ... 16
4.1 Kiihdytetty docker virtuaalikone ... 16
4.2 MIG allokointi ... 16
5 Docker ympäristö ... 20
5.1 Käyttötarkoitus ... 20
5.2 Docker CLI ... 20
5.3 Portainer ... 21
5.4 Jupyter ... 22
5.5 NginX ... 22
6 Yhteenveto ... 24
Lähteet ... 25

Sanasto

CLI	Command Line Interface (Komentorivi käyttöliittymä)
GPU	Graphics processing unit (Grafiikka suoritin)
HBM	High-Bandwidth memory
HPC	High performance computing (Suurteholaskenta)
MIG	Multi-instance GPU (Moni-Instanssinen GPU)
PCI-e	Peripheral Component Interconnect Express
SM	Streaming Multiprocessor

1 Johdanto

Koneoppiminen on tulossa yleiseksi monilla aloilla, mutta kaikilla ei ole ideaa mistä aloittaa. Tarjoamalla koneoppimiseen soveltuvia työkaluja, osaamista ja tietotaitoa, voidaan kannustaa yrityksiä kokeilemaan koneoppimista. Laskentaintensiivisissä sovelluksissa resursseja voidaan lisätä käyttämällä kiihdytettyä laitteistoa.

Opinnäytetyö on suoritettu osana ITKO-hanketta (Yrityslähtöiset IoT-ratkaisut ja koneoppiminen) joka on LAB-ammattikorkeakoulun 2 vuotinen hanke, joka alkoi vuonna 2019 ja loppuu 2021. Tarkoituksena on auttaa kehittämään Päijät-Hämeen alueen pienten ja keskisuurien yrityksien IoT:n, Big Datan ja koneoppimisen hyödyntämismahdollisuuksia.

Tavoitteena on ottaa käyttöön kiihdytettyä laskentaa virtuaalisessa ympäristössä. Käyttämällä kiihdytettyä laskentakorttia, jonka laskentaresursseja on mahdollista jakaa, voidaan luoda useammalle käyttäjälle mahdollisuus hyödyntää laitteistoresursseja. Kiihdytetyn laskennan suorittamiseen käytetään sovelluksia, joilla voidaan suorittaa koneoppimista, sekä tekoälyn koulutusta. Data on mahdollista tallentaa paikallisesti palvelimelle, josta se voidaan myöhemmin hakea datan analysointia, visualisointia tai kouluttamista varten. Jotta asiakkaat voivat suorittaa kiihdytettyä laskentaa, palveluiden on oltava saatavilla verkon kautta.

Tämän opinnäytetyön teknologiat osuudessa käsitellään, mitä teknologioita tarvitaan koneoppimisympäristön käyttöönottoon palvelimella. Käytössä oleva NVIDIA A100 kiihdytinkortti on mahdollista jakaa useampaan itsenäiseen virtuaaliseen laitteeseen, joita hyödyntämällä asiakkaille voidaan allokoida yksilölliset laskentaresurssit käytettäväksi. Näiden resurssien hyödyntämiseksi verkon välityksellä yhden yhtenäisen osoitteen kautta käytetään web-sovelluksia, jotka ovat käänteisen välityspalvelimen takana.

2 Toimintaympäristö

2.1 Yleiskuva

Toimintaympäristönä toimi palvelin, joka sijaitsee datakeskuksessa. Lähtökohtana tällä palvelimella oli asennettuna avoimen-lähdekoodin virtualisointialusta Proxmox. Tällä voidaan luoda virtuaalikoneita ja LXC-kontteja (Linux container). Virtuaalikoneet ja LXC-kontit ovat virtuaalisesti luotuja instansseja, jotka imitoivat fyysistä tietokonetta. Näille on sitten allokoitu pienempiä osia kaikista palvelimella olevista resursseista. Näin laskentaresurssit voidaan jakaa tehokkaasti tarpeiden mukaan.

2.2 Palvelin

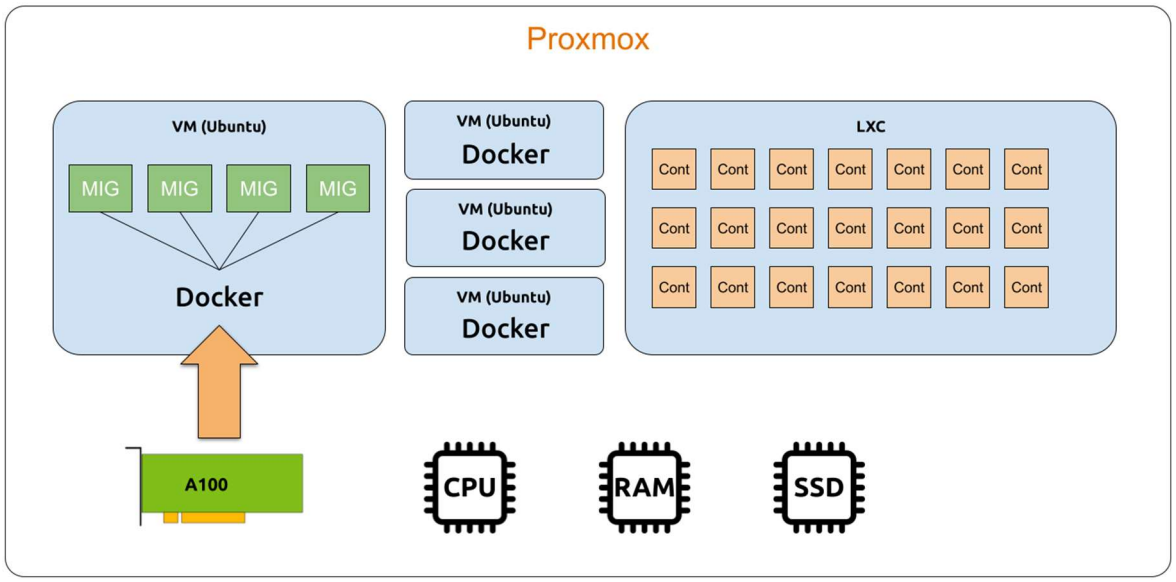
Palvelimena toimii Dell R7525, jonka kokoonpanoon kuului kaksi AMD EPYC 7502P prosessoria, joissa on 32 ydintä/64 säiettä. Keskusmuistia oli 8 muisti kampaa, joista jokainen oli 32GB eli yhteensä 256GB. Tallennustilana toimi kaksi 3.84TB SAS-väyläistä SSD-levyä (Solid State Drive). Myöhemmin kokoonpanoon lisättiin kuusi Samsung PM883 480GB levyä. Kaikki kahdeksan levyä ovat Read-Intensive painotteisia, joka tuo kokonaistallennustilan noin 12 Teratavun kapasiteettiin. Levyille väyläkorttina toimi Dell HBA330-isäntäväyläsovitin (Host Bus Adapter). Jotta verkkoliikenne saataisiin kytkettyä kahteen eri kytkimeen, palvelimeen lisättiin Broadcom 57416 NIC Dual Port 10GbE SFP+ verkkokortti mahdollistaen vikasietoisen verkon. Palvelimen etähallintaan ja monitorointiin käytettiin Dell:n IDRAC9 (Integrated Dell Remote Access Controller) Enterprise X5 etähallinta ohjainta. Virtalähteenä toimi Dell:n vikasietoinen 1+1 1100W virtalähde. Kuvasta 1 nähdään fyysinen palvelin.

Kuva 1. Fyysinen palvelin

2.3 Alustetut virtuaalikoneet

Virtualisointialustalle oli valmiiksi asennettu virtuaalikone, jolla kiihdytettyä laskentaa oli tarkoitus toteuttaa. Käyttöjärjestelmänä virtuaalikoneella toimi Ubuntu 18.04.5 LTS (Long Term Support). Virtuaalikoneelle oli myös annettu NVIDIAn kiihdytyskortti käyttöön pass-through teknologian avulla ja asennettu sille sopivat ajurit. Kiihdytyskortin resurssien jakamiseksi virtuaalikoneelle oli asennettu Docker ja niitä yhdistämään vaadittu nvidia-container-runtime, jonka avulla kontit voivat hyödyntää kiihdytettyjä resursseja. Tämän ajoympäristön avulla voidaan luoda kontteja, jotka voivat hyödyntää grafiikka suoritinta riippumatta käyttöönottoympäristöstä (NVIDIA 2021a).

Kuvassa 2 nähdään keskellä kolme Ubuntu Docker yksikköä, jotka kuvastavat testikäytössä olevaa kolmea virtuaalikonetta. Nämä virtuaalikoneet ovat identtisiä tuotantokäytössä olevaan virtuaalikoneeseen nähden, mutta niillä ei ole mahdollisuutta käyttää kiihdytettyjä resursseja. Näiden virtuaalikoneiden päätarkoitus oli sovellustyökalujen toimivuuden varmistaminen ja käyttöönoton testaaminen. Testaaminen tapahtui toisella virtuaalikoneella kiihdytettyjen laskentaresurssien säästämiseksi ja käyttökatkosten ehkäisemiseksi.



Kuva 2. Proxmox toimintaympäristö

3 Teknologiat

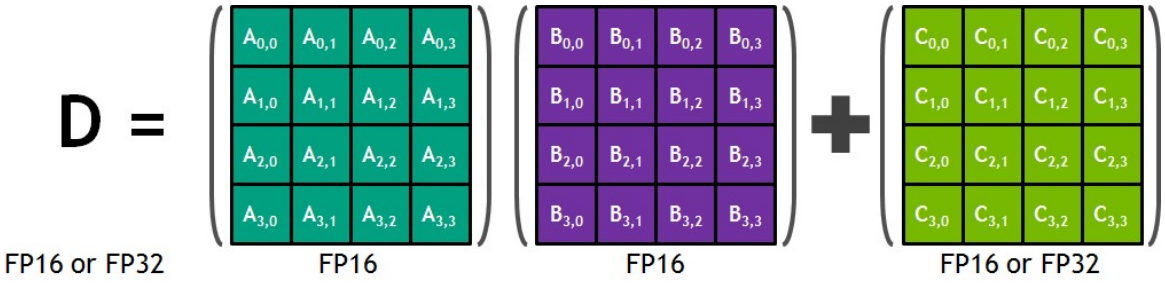
3.1 NVIDIA A100 (MIG)

Palvelimelle oli lisätty NVIDIA A100 kiihdytinkortti, jota käytetään kiihdytetyn laskennan suorittamiseen. Kortilla on 40GB HBM2 (High Bandwidth Memory) muistia ja grafiikkasuorittimen muistikaistanleveyttä on 1,5TB/s. Suurin suunniteltu lämpäteho (Thermal Design Power) on 250W. Pääkäyttötarkoituksiin kuuluu tekoäly, data analytiikka sekä hpc (High Performance Computing) sovellukset. NVIDIA A100 kiihdytinkortti on lähes 250 kertaa nopeampi verrattuna vain prosessorilla tehtyyn BERT-LARGE (Bidirectional Encoder Representations from Transformers) inference testiin. (NVIDIA 2021c). BERT on menetelmä, jolla voidaan kouluttaa yleiskäyttöinen kielen ymmärtämisen malli. Tällä saadaan tuotettua huippuluokan tuloksia luonnollisen kielen käsittelyssä. (HPCAC 2020). NVIDIA A100 kiihdytinkortti julkaistiin 14 toukokuuta 2020. Kortti perustuu ampere arkkitehtuuriin ja käyttää kolmannen sukupolven tensor ytimiä. Ampere on koodinimi NVIDIAn GPU korteissa käytetylle mikroarkkitehtuurille.

3.1.1 Tensor ydin

Koneoppimisessa ja tekoälyn kehityksessä työskennellään paljon neuroverkkojen kanssa, jotka ovat matriisilasku painotteisia. Tensor ytimet ovat erikoistuneet matriisi laskujen tekemiseen, joka tekee niistä erinomaisen valinnan koneoppimiseen ja tekoälyn kehitykseen. Kolmannen sukupolven Tensor ytimet voivat tehdä 256 FP16/FP32 (16/32 bit Floating Point) FMA (Fused multiply-add) operaatiota kellojaksoa kohden. A100 kortissa on neljä kolmannen sukupolven Tensor ydintä jokaista SM:a (Streaming Multiprocessor) kohden, joista tulee yhteensä 1024 FP16/FP32 FMA operaatiota kellojaksoa kohden. A100 kiihdytinkortti sisältää 108 SM:a ja 432 kolmannen sukupolven Tensor ydintä. (Krashinsky 2020.)

Vuonna 2008, IEEE 754 standardia muutettiin sisältämään FMA operaation. FMA operaation laskee rn(A x B + C) yhdellä aritmeettisella pyöristyksellä. Ilman FMA operaatiota, tulos laskettaisiin rn(rn(A x B) + C) kahdella aritmeettisella pyöristyksellä. Yksi pyöristys kertolaskulle ja toinen yhteenlaskulle. Kun tämä pyöristys tehdään vain kerran, saadaan tarkempi lopputulos. (Whitehead & Fit-Florea 2011). Kuvassa 3 nähdään MAC (Multiply-accumulate) matriisi lasku, jonka tulos pyöristämällä saadaan FMA lasku.



Kuva 3. MAC matriisi lasku (NVIDIA 2021)

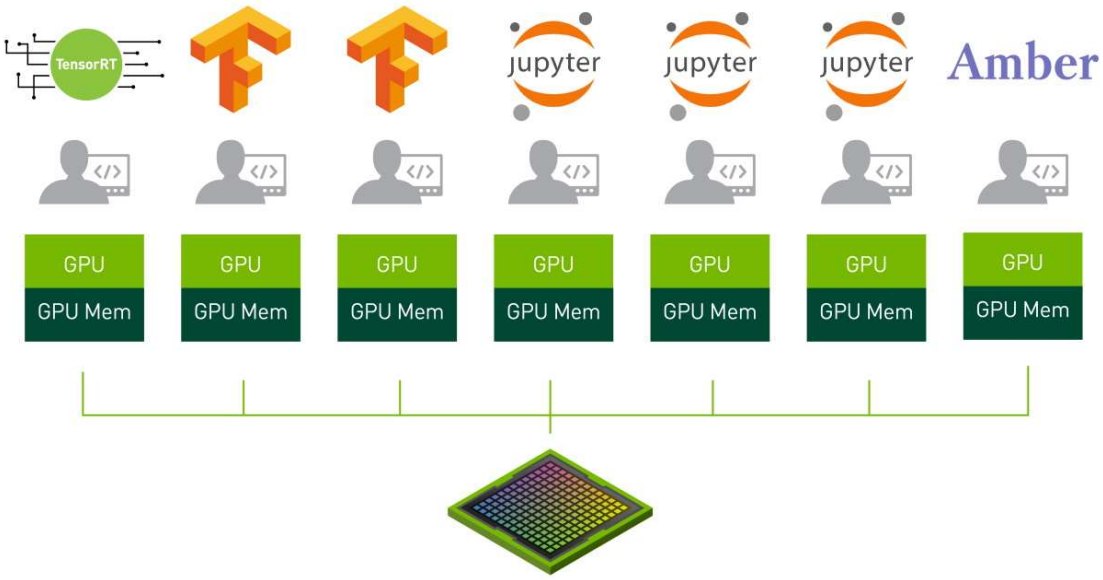
Kuvassa 4 on yksi SM ja sen sisältö. Tämä SM sisältää 4 prosessointi lohkoa, jotka jakavat 192 kilotavun L1 välimuistin. Tätä välimuistia käytetään myös jaettuna muistina, jotta lohkot voivat jakaa dataa keskenään. Jokainen prosessointi lohko sisältää yhden warp vuorontajan (scheduler), 16 INT32 (32bit integer) CUDA ydintä, 16 FP32 CUDA ydintä, 8 FP64 CUDA ydintä sekä tensor ytimen matriisi laskujen suorittamiseen. Prosessointi lohkot ovat jaettu pienempii warp osioihin, jotka sisältävät 32 säiettä (Hui 2020.) Warp vuorontaja määrittää, mikä warp osio suorittaa seuraavan tehtävän. Yksi SM voi sisältää 64 warp osiota ja yhteensä 2048 säiettä. (NVIDIA 2020.)

Kuva 4. Ampere SM arkkitehtuuri (Hui 2020)

3.1.2 MIG

MIG teknologialla voidaan jakaa A100 kortin resurssit, jopa seitsemään itsenäiseen MIG laitteeseen, jotka ovat täysin eristetty ja joilla on omat laajakaistaiset muistit, välimuisti ja laskenta ytimet. MIG laite koostuu yhdestä GPU- ja laskenta instanssista. Koska jokainen laite on eristetty, virheet yhdessä laitteessa ei vaikuta muihin, jotka ovat samassa

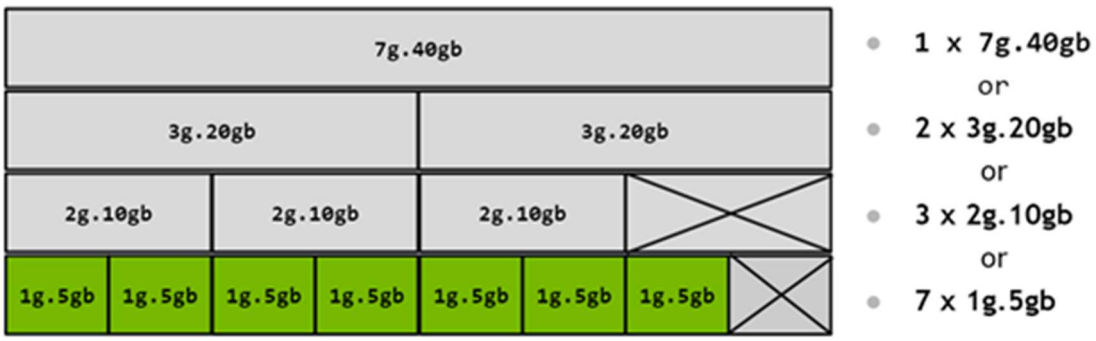
fyysisessä GPU:ssa (Chandrasekaran 2021.) Kuvassa 5 nähdään kuinka A100 kortti on jaettu seitsemään osaan, joissa jokaisella on oma käyttäjä.



Kuva 5. MIG toiminta malli (Chandrasekaran 2021)

GPU instanssi muodostuu yhdestä GPU lohkosta ja GPU muistin lohkosta. GPU lohkon koko on noin seitsemäs osa kaikista SM:en määrästä. GPU muisti lohkon koko on noin kahdeksas osa kaikesta muistista eli 5 gigabittiä sisältäen muisti ohjaimen ja välimuistin. GPU instanssin voi jakaa useampaan laskenta instanssiin. Jokainen laskenta instanssi sisältää osan isäntä GPU instanssin SM:sta ja GPU moottoreita (NVDEC, DMA jne.). (NVIDIA 2021b.)

Kuvassa 6 nähdään eri profiilit, joita A100 tukee MIG tilassa. Näitä profiileja käytetään, kun toteutetaan kortin jakaminen. GPU lohkojen määrän puutteessa verrattuna GPU muistin lohkojen määrään, GPU instanssien määrä jää vajaaksi 7x 1g.5gb konfiguraatiossa. Muisti lohkojen koko on ennalta määrätty ja niiden koko ei voi muuttaa, joten muistia jää käyttämättä. Sama tapahtuu 3x 2g.10gb konfiguraatiossa. Jotta kaikki instanssit olisivat identtisiä yksi GPU lohko ja 2 muisti lohkoa on jätettävä pois. Näitä käyttämättömiä lohkoja voidaan allokoida myös. Tämä muodostaisi 3x 2g.10gb + 1x 1g.5gb konfiguraation.

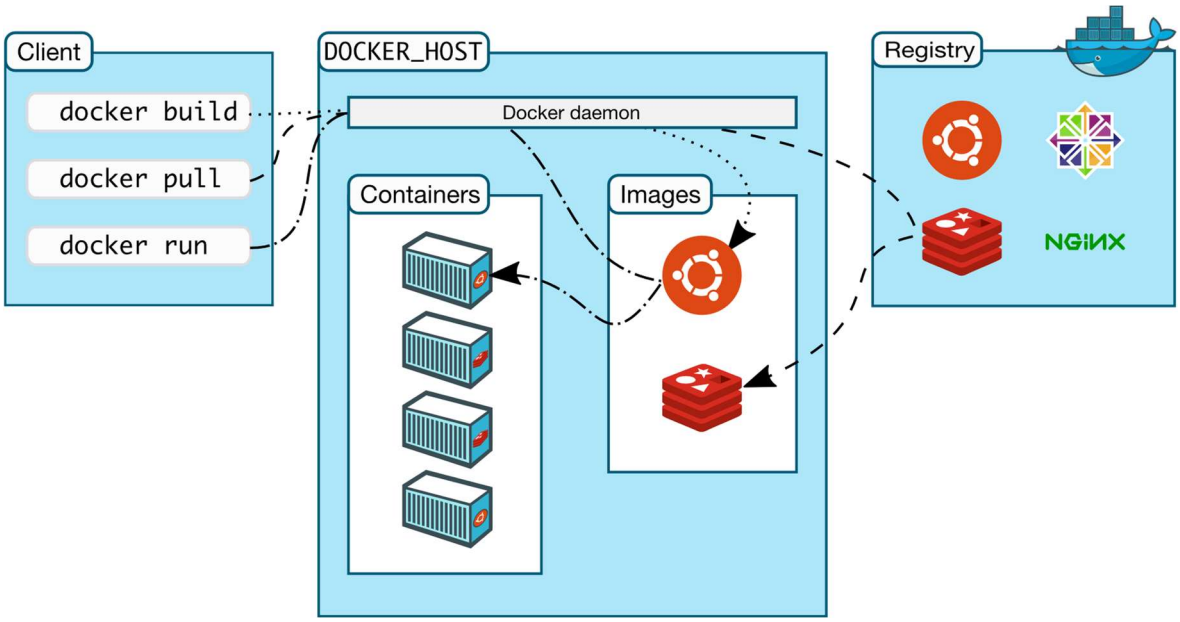


Kuva 6. Mig profiili esimerkki (NVIDIA 2021b)

3.2 Docker

Docker on virtualisointi työkalu, jolla voidaan kehittää, toimittaa ja suorittaa sovelluksia. Näitä sovelluksia ajetaan konteissa, jotka ovat löysästi (loosely) eristetty ympäristöstään (Docker 2021a). Docker toimii komentorivillä mutta siitä on myös saatavilla visuaalisen käyttöliittymän kanssa.

Kuvassa 7 nähdään Dockerin arkkitehtuuri. Docker toimii client-server arkkitehtuurilla. Docker client (vasemmalla) keskustelee docker palveluprosessin (daemon) kanssa (keskellä), joka suorittaa kaiken työn taustalla. Docker palveluprosessi voi sijaita samalla laitteella kuin client mutta voi myös sijaita toisella laitteella. Nämä keskustelevat keskenään käyttäen REST API:a (Application Programming Interface).



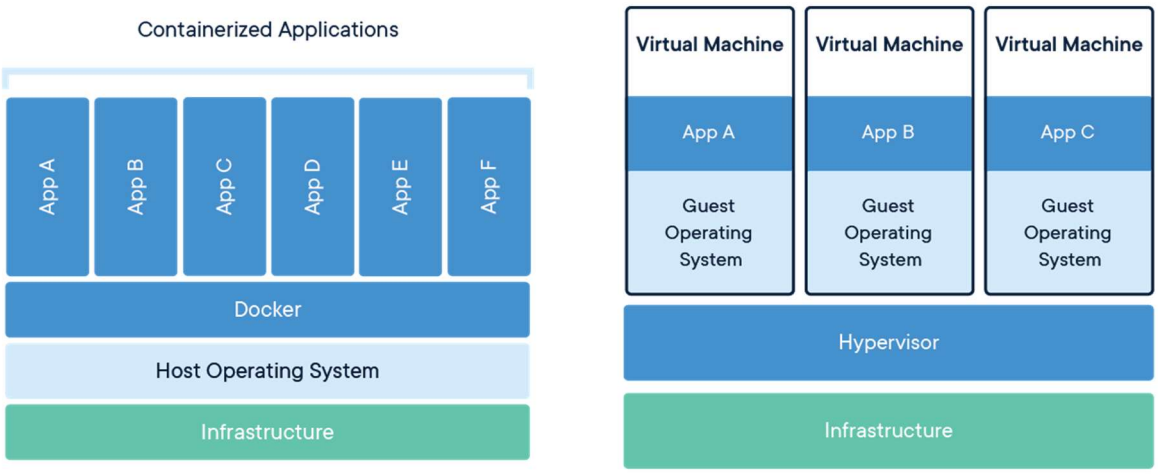
Kuva 7. Docker arkkitehtuuri (Docker 2021a)

Docker komentorivityökalulla voidaan luoda kontti levykuvasta. Levykuva on joukko ohjeita, joita käytetään kontin luomiseen. Levykuva on verrattavissa virtuaalikoneen

tilannevedokseen (snapshot) (Gillis 2021). Konttiin voi luonti vaiheessa liittää tallennustilaa, jotta tiedot eivät katoa, liittää verkkoon, jotta se voi keskustella muiden konttien kanssa tai luoda uuden imagen perustuen kontin nykyiseen tilaan. Docker tarjoaa rekisterin nimeltä DockerHub, jossa voidaan jakaa kontti levykuvia. Tässä palvelussa voidaan esimerkiksi puskea ja hakea levykuvia, sekä luoda yksityisiä varastoja (repository). (Docker 2021b.)

3.2.1 Virtuaalikoneiden ja konttien eroavaisuus

Virtuaalikoneet virtualisoivat alla olevan laitteiston, jotta voidaan ajaa useampaa käyttöjärjestelmä instanssia. Jokaisella virtuaalikoneella on pääsy sille allokoituihin resursseihin, jotka kuvastavat alla olevaa laitteistoa. (Azure 2021.) Kontit toisaalta virtualisoivat alla olevan käyttöjärjestelmän ja saavat kontit näkemään tämän käyttöjärjestelmän omanaan, mukaan lukien alla olevat resurssit kuten: prosessori, muisti, tiedosto muisti ja verkko. Kontit luodaan levykuvasta, joka on kuin tietokoneen varmuuskopio ja koko on yleensä kymmenissä megatavuissa. Levykuva voidaan luoda Dockerfile tiedostosta, joka sisältää ohjeet siitä, mitä levykuvassa on. Kuvassa 8 nähdään konttien ja virtuaalikoneiden erot.



Kuva 8. Konttien ja virtuaalikoneiden ero (Docker 2021c)

3.2.2 Kontti

Kontit ovat tapa pakata sovellukset yhteen ajettavaan(executable) pakettiin, jossa on sovellus ja sen tarvittavat tiedostot, kirjastot sekä riippuvuudet, jotka vaaditaan sovelluksen ajamiseen. Nämä sovellukset on eristetty siten, ettei niissä tule käyttöjärjestelmää mukana. Tämän sijaan isäntä koneelle asennetaan Docker runtime engine, jonka kautta kontit voivat jakaa kerneleitä keskenään. (IBM 2021.)

Docker kontit erikoistuvat yhden sovelluksen pakkaamiseen konttiin, kun LXC erikoistuu käyttöjärjestelmän pakkaamiseen konttiin. LXC kontit jakavat isäntäkoneen kernelin. Nämä

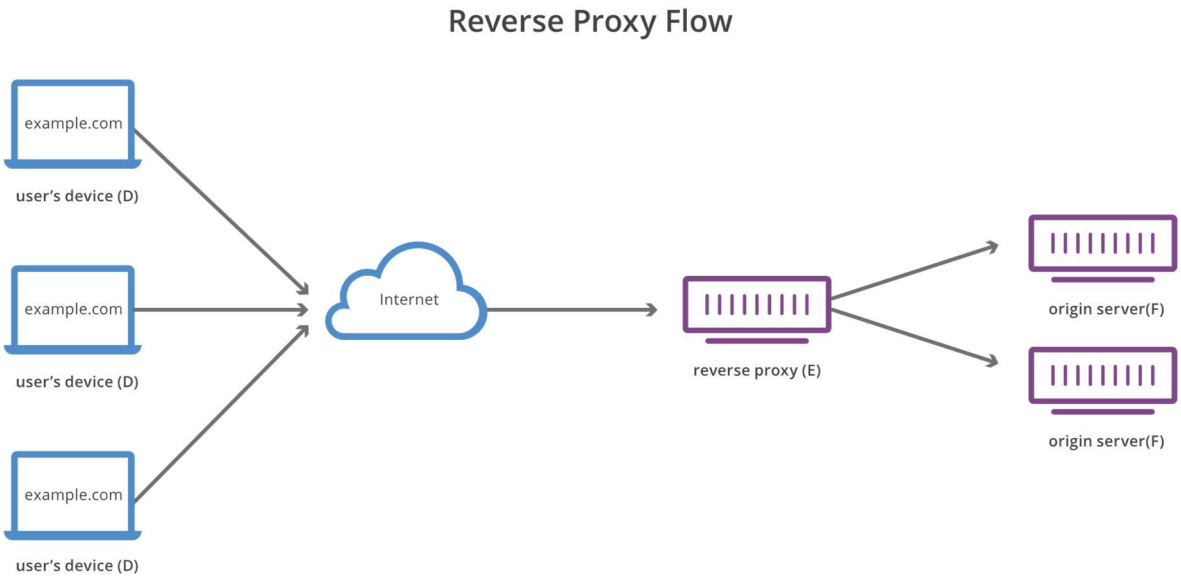
käyttöjärjestelmät toimivat kuten tavallinen virtuaalikone, joten sinne voi ottaa yhteyden SSH:n (Secure Shell) kautta ja asentaa ohjelmia kuin tavalliselle koneelle. (Gikonyo 2021.)

Kontti teknologia valittiin, koska konteilla voidaan hyödyntää tietokoneen resursseja tehokkaammin kuin virtuaalikoneilla. Virtuaalikoneita luodessa niille täytyy antaa kiinteä(fixed) määrä RAM muistia, kun kontit käyttävät muistia joustavasti. Dockeria ei asennettu LXC konttiin, koska vastaan tulee ongelmia tiedon tallentamisen kanssa.

Jotta kontteihin pääsee käsiksi ilman pääsyä docker ympäristöön, konttien portteja on paljastettava isäntäkoneelle, jonka jälkeen voidaan käyttää käänteistä välityspalvelinta (reverse proxy) pyyntöjen reitittämiseen oikeaan paikkaan. Kontin näkökulmasta sillä on oma verkko interface (network interface) jossa sillä on oma IP osoite, yhdyskäytävä, reititystaulukko jne.

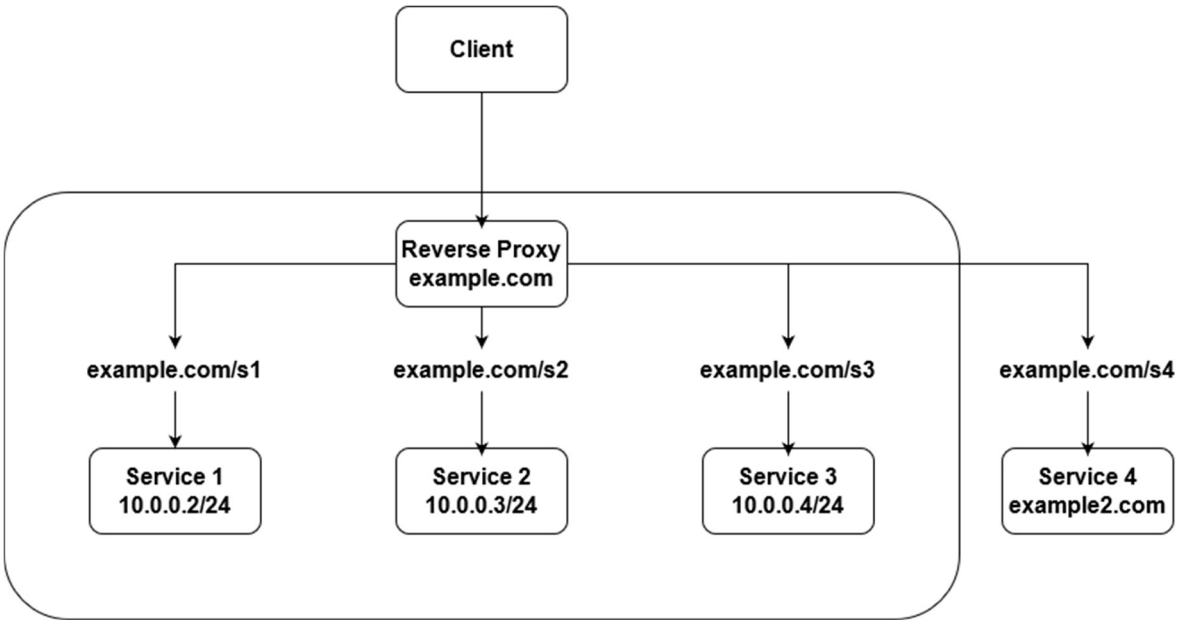
3.3 Käänteinen välityspalvelin

Välityspalvelimet ohjaavat liikennettä asiakkaan ja toisen palvelimen välillä. Tekemällä näin palvelin voi säädellä liikennettä ennalta asetettujen sääntöjen mukaisesti. Säännöillä voidaan esimerkiksi peittää asiakkaan IP-osoite, sekä estää tuntematonta liikennettä suojellen asiakasta samanaikaisesti (Zalman 2021). Käänteinen välityspalvelin (reverse proxy) vastaanottaa asiakkaalta pyynnön, välittää sen toiselle palvelimelle ja palauttaa vastauksen takaisin asiakkaalle ikään kuin välityspalvelin olisi itse käsitellyt pyynnön. Se myös varmistaa, että useita palvelimia voidaan käyttää yhdestä osoitteesta riippumatta paikallisverkon rakenteesta. Kuvasta 9 nähdään, miten käänteinen välityspalvelin sijaitsee muiden palvelimien edessä.



Kuva 9. Käänteinen välityspalvelin palvelimien edessä (Cloudflare 2021b)

Käänteinen välityspalvelin toimii myös palomuurina tämän takana sijaitseville palvelimille, jotka ovat kuvassa nimellä origin server. Koska kaikki liikenne kulkee käänteisen välityspalvelimen läpi, tällä voidaan suodattaa haitallista liikennettä. Tämä myös mahdollistaa kyselyiden tekemisen palveluihin, jotka ovat samassa sisäverkossa palvelimen kanssa. Kuvasta 10 nähdään miten asiakas ottaa yhteyttä käänteiseen välityspalvelimeen ja se lähettää pyynnön eteenpäin riippuen mitä URL-osoitteen (Uniform Resource Locator) perään kirjoitetaan.



Kuva 10. Käänteinen välityspalvelin

Kun otetaan käyttöön web sovelluksia käänteisen välityspalvelimen takana, on otettava huomioon, että sovellukset saattavat luulla olevansa suoraan host osoitteen takana osoitteessa example.com. Käyttäen kuvaa 10 esimerkkinä, jos sovellus on osoitteessa 10.0.0.2 ja asiakas ottaa yhteyttää osoitteeseen example.com/s1, sovellus ei välttämättä tiedä mistä polusta löytää tiedostoja toimiakseen ja palauttaa virheen asiakkaalle. Monessa web sovelluksessa on mahdollisuus asettaa base url, joka kertoo sovellukselle, että ”etsi tiedostoja tästä osoitteesta”. Tässä esimerkissä, jos base url asetettaisiin olevan s1, sovellus tietäisi mistä etsiä tiedostoja ja toimisi odotetusti.

3.4 Jupyter

Jupyter on web-pohjainen avoimen lähdekoodin sovellus, jonka avulla voidaan luoda ja jakaa dokumentteja (notebook), joissa voi olla live koodia, yhtälöitä, visualisoitua dataa sekä selostavaa tekstiä. Pääkäyttötarkoituksiin kuuluu esimerkiksi: tilastollinen mallintaminen, datan visualisointi sekä koneoppiminen. Jupyter tukee yli 40 ohjelmointikieltä mukaan lukien: Python, R ja Javascript. (Jupyter 2021b.)

Vuonna 2014 IPythonista ilmoitettiin tulevan sivu projekti nimeltä Project Jupyter (Perez 2014). Ensimmäinen vakaa versio julkistettiin 20. helmikuuta 2018.

Dokumentit edustavat kaikkea, mikä nähdään web käyttöliittymässä kuten: koodin syötteet ja tulosteet, selostavat tekstit, matemaattiset yhtälöt, sekä datan visualisoinnit. Nämä tiedot tallennetaan ipynb muotoiseen tiedostoon, joka on todellisuudessa JSON (JavaScript Object Notation) tiedosto. Tämä tekee siitä helposti jaettavan ja versiohallittavan tiedoston. (The Jupyter Notebook 2015.)

Dokumentit ovat jaettu soluihin, jotka voivat sisältää tekstiä tai koodia. Kun dokumentti suoritetaan, kaikki solut käydään läpi yksi kerrallaan, tulostaen solujen tulostuksen niiden alle. Tämä tuloste voi olla esimerkiksi muuttujan arvo tai kaavio datasta. Dokumenttien oletus ohjelmointi kieli on Python, mutta Jupyteriin voi asentaa muita kerneleitä, jotka mahdollistavat muiden ohjelmointikielien käyttämisen. Nämä muut kernelit ovat yhteisön ylläpitämiä (Jupyter 2021a).

Kernelillä tässä tapauksessa ei ole mitään yhteistä käyttöjärjestelmän kernelin kanssa. Kerneli tässä tapauksessa on osa taustajärjestelmää (backend), joka on vastuussa käyttäjän kirjoittaman koodin suorittamisesta. (Renou 2019.)

4 Resurssien allokointi

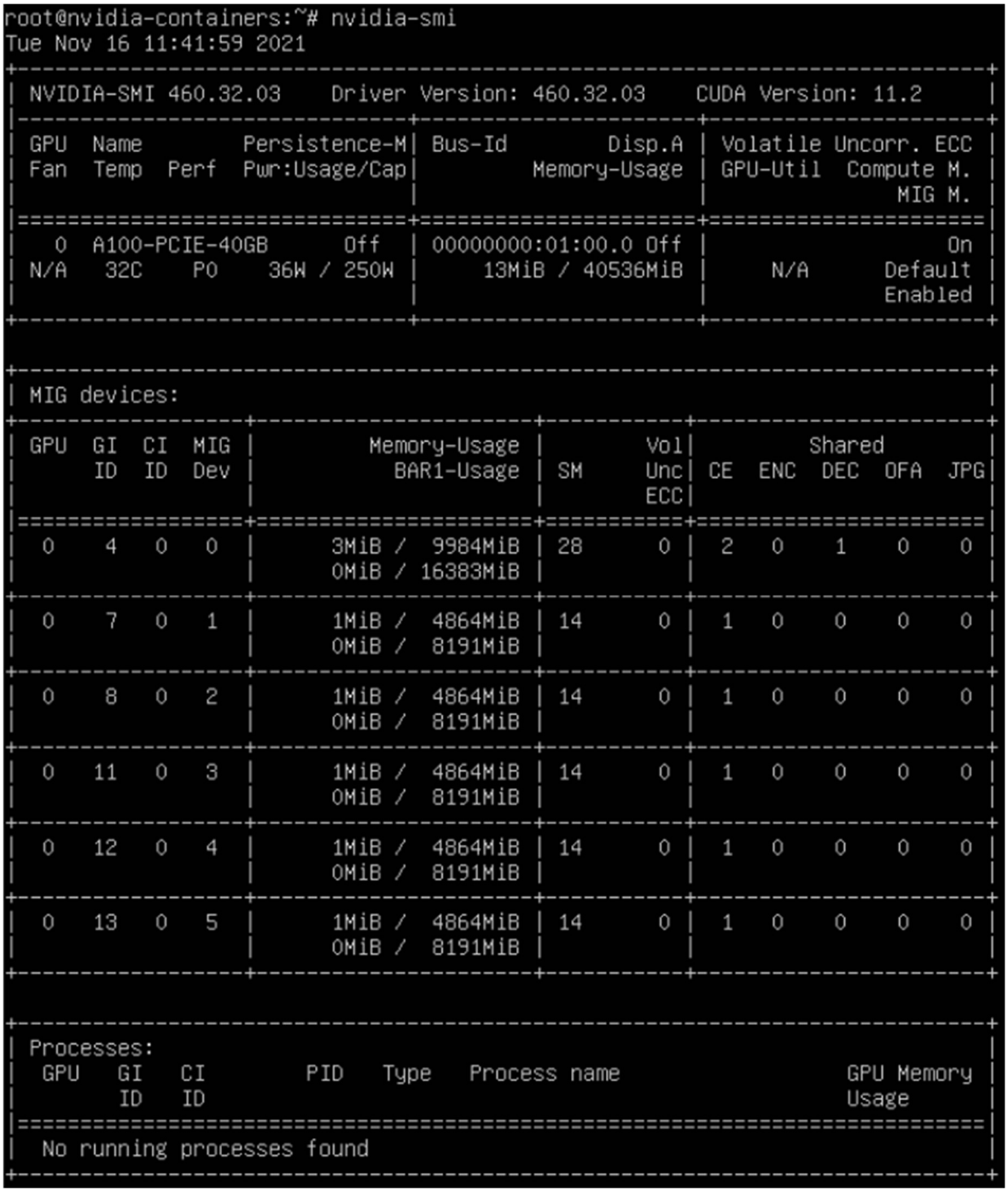
4.1 Kiihdytetty docker virtuaalikone

Proxmox palvelimella olevat virtuaalikoneet ja LXC kontit käyttävät Ubuntu 20.04 LTS käyttöjärjestelmää, koska tiimillä oli eniten kokemusta kyseisestä käyttöjärjestelmästä, jolloin ylläpidon osaamisen tarve on vähäisempi, kun on vain yksi käyttöjärjestelmä käytössä. Virtuaalikoneelle, jossa ajetaan koneoppimista, on annettu runsaat resurssit, jotta useampi käyttäjä voi käyttää alustaa ilman huomattavaa suorituskyvyn laskua, vaikka palvelin olisi suuremman kuorman alla. Jotta A100 kortti saadaan käyttöön virtuaalikoneessa, se täytyy antaa käyttöön pass-through tilassa. Tämä tarkoittaa, että virtuaalikone saa pääsyn PCI-e (Peripheral Component Interconnect Express) laitteeseen suoraan, jotta saadaan lähes täysi suorituskyky virtuaalikoneen sisällä. Proxmoxin puolella vaaditaan konfigurointia, jotta PCI-e laitteen saa käyttöön virtuaalikoneen sisällä, mutta tämä oli tehty ennen hankkeeseen liittymistä.

Jotta kontit voivat hyödyntää tätä A100 korttia täytyy asentaa laitevalmistajan ajurit, nvidia-cuda ja dockerille nvidia runtime, mutta kuten proxmoksissakin, tämä oli tehty etukäteen. A100 kortti ei ole oletuksena jaettu osiin, mutta käyttämällä nvidia-smi työkalua, joka tulee nvidia ajureiden mukana, kortti voidaan jakaa useampaan laitteeseen, joita kontit voivat sitten hyödyntää.

4.2 MIG allokointi

Jotta MIG:ä voidaan käyttää, on A100 kortti jaettava GPU- ja laskenta instansseihin käyttäen NVIDIAn tarjoamia työkaluja. Nämä työkalut ovat tarjolla muissakin NVIDIAn näytönohjaimissa, mutta tarjoavat vähemmän ominaisuuksia. Nvidia-smi on komentorivi työkalu, joka auttaa hallinnoimaan ja monitoroimaan NVIDIA GPU laitteita. Tällä komennolla voidaan nähdä esimerkiksi jokaisen laitteen käyttöaste, prosessit, jotka käyttävät A100 korttia sekä käytössä olevat resurssit. Kuvassa 11 nähdään nvidia-smi komennon tulostus konsolissa.



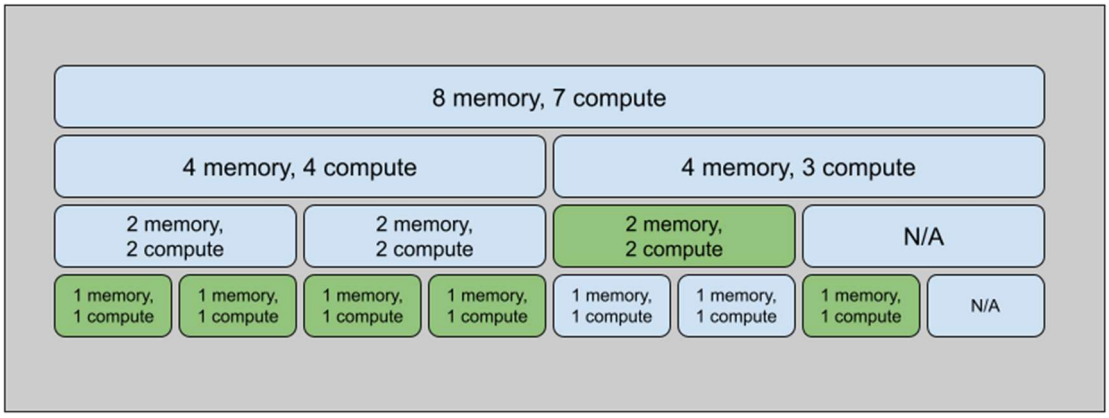
Kuva 11. Nvidia-smi komennon tulostus

Tällä työkalulla A100 kortti jaetaan kuuteen osaan, joista yksi osa on tehokkaampi kuin muut. Näin varmistetaan, että mahdollisimman paljon resursseja on käytössä monelle käyttäjälle ja yksi tehokkaampi osio niille, joiden työmäärä on liian suurin pienimmälle osiolle kiihdytyskortista. Ennen kuin A100 voidaan jakaa instansseihin, täytyy tietää mitä mahdollisuuksia on. Ajamalla nvidia-smi mig -lgip komento voidaan nähdä eri profiilien ID:t, joita käytetään kiihdytyskortin jakamisessa sekä kuinka monta SM:ä jokaisella profiililla on. Kuvasta 12 nähdään komennon tulostus ja eri profiilien ID:t.

```
root@nvidia-containers:~# nvidia-smi mig -lgip
+-----------------------------------------------------------------------------+
| GPU instance profiles:                                                      |
| GPU   Name          ID    Instances   Memory     P2P    SM    DEC   ENC    |
|                           Free/Total   GiB              CE    JPEG  OFA    |
|=============================================================================|
|   0  MIG 1g.5gb     19     0/7         4.75       No     14     0     0    |
|                                                           1     0     0    |
+-----------------------------------------------------------------------------+
|   0  MIG 2g.10gb    14     0/3         9.75       No     28     1     0    |
|                                                           2     0     0    |
+-----------------------------------------------------------------------------+
|   0  MIG 3g.20gb     9     0/2         19.62      No     42     2     0    |
|                                                           3     0     0    |
+-----------------------------------------------------------------------------+
|   0  MIG 4g.20gb     5     0/1         19.62      No     56     2     0    |
|                                                           4     0     0    |
+-----------------------------------------------------------------------------+
|   0  MIG 7g.40gb     0     0/1         39.50      No     98     5     0    |
|                                                           7     1     1    |
+-----------------------------------------------------------------------------+
```

Kuva 12. Nvidia-smi profiilien ID:t

Instanssit tullaan jakamaan Kuvan 13 mukaisesti. Kuvaa katsottaessa on huomioitava, että vihreät osiot eivät saa mennä päällekkäin muuten samat resurssit allokoitaisiin useampaan instanssiin.



Kuva 13. MIG allokointi

MIG instanssien luominen tapahtuu käyttäen nvidia-smi mig -cgi komentoa. Tämä komento ottaa argumentteina listan profiili ID:tä. Yllä olevia kuvia hyväksi käyttäen voidaan luoda kuusi instanssinen jako.

Kuvassa 14 on komento, jolla luotiin MIG laitteet. Tämän komennon cgi (Create GPU Instance) optiolla määritellään, miten A100 kortti jaetaan. Tämä määrittelyn voi tehdä kolmella tavalla (NVIDIA 2021b):

- Antamalla profiili ID:n (esim. 19, 14, 9)
- Profiilin lyhyen nimen (esim. 2g.10gb)
- Profiilin koko nimen (esim. 3g.20gb)

```
$ nvidia-smi mig -cgi 19,19,19,19,14,19 -C
```

Kuva 14. MIG instanssien jakamiseen käytetty komento

Nämä tiedot ovat saatavilla kuvan 12 komennosta. Käyttämällä C optiota nvidia-smi luo laskenta instanssit. Jos laskenta instansseja ei luoda, A100 ei voi suorittaa CUDA laskentoja, jolloin ei voida suorittaa kiihdytettyä laskentaa koneoppimissovelluksissa. Lopputuloksena GPU- ja laskenta instanssien luomisesta on MIG laite. Jotta nämä laitteet saatiin kontin sisällä käyttöön, kontti oli luotava Docker CLI:n (Command Line Interface) kautta.

5 Docker ympäristö

5.1 Käyttötarkoitus

Dockeria käytetään yleensä CPU (Central Processing Unit) painotteisten sovellusten käyttöönottoon (NVIDIA 2021d). Käyttämällä nvidia-container-runtimea on mahdollista käyttää A100 kiihdytinkorttia konttien sisällä ja ajaa GPU painotteisia sovelluksia. Etuina GPU-sovellusten suorittamisessa konteissa on:

- käyttöönoton helppous
- NVIDIA ajurit täytyy asentaa vain kerran isäntäkoneelle
- yksittäisten laitteiden eristäminen
- suorittaminen heterogeenisessä ympäristössä

5.2 Docker CLI

Docker CLI toimii pääsääntöisenä käyttöliittymänä, kun Docker on otettu käyttöön palvelimella, visuaalisen käyttöliittymän puutteen vuoksi. Dockerilla luodut kontit voidaan suorittaa oletuksena edustalla (foreground) tai taustalla (background) detached tilassa. Edustalla suoritettavien konttien standard input ja output liitetään isäntäkoneen konsoliin. Detached tilassa suoritettavat kontit menevät taustalle ja Docker tulostaa kontin id:n. (Docker 2021d.)

Docker run on komento, jolla luodaan kontit. Tämä komento ottaa vastaan optioita, levykuvan ja mahdollisen komennon argumentteineen, joka ajetaan kontin käynnistyessä.

Kuvassa 15 on komento, jolla on luotu kontti, joka voi hyödyntää A100 kiihdytinkorttia MIG tilassa. Käyttämällä -d optiota määritetään kontin olevan detached tilassa. MIG laitteen hyödyntämiseksi kontin runtime optio on asetettava nvidia tilaan. NVIDIA kehitti tämän runtime option, jotta voitaisiin luoda ajuri agnostisia CUDA levykuvia (Calmes 2018). Optiolla gpus voidaan määrittää, mitä MIG laitetta kontti voi hyödyntää. Tässä tapauksessa kontti voi hyödyntää MIG laitetta, jolla on id 4.

```
$ docker run -d --runtime=nvidia --gpus '"device=0:4"' --name mig-4 hello-world
```

Kuva 15. MIG laitteen hyödyntäminen kontissa

Kontit, joissa on MIG laite, pystyvät hyödyntämään nvidi-smi komentoa nähdäkseen niille määritetyn MIG laitteen käyttöasteen. Kontin sisällä suoritettu nvidia-smi komento ei näytä prosesseja, koska ajurit eivät ole tietoisia PID (Process ID) nimiavaruudesta. Kun kontti on luotu, sitä voidaan hallita Portainer sovelluksella.

5.3 Portainer

Portainer on web-sovellus, jolla voidaan hallita Docker kontteja. Tämän sovelluksen avulla on mahdollista luoda, hallita, muokata sekä monitoroida kontteja. Portainerilla ei voida asettaa nvidia runtime optiota, joten kontit, jotka haluavat käyttää kiihdytinkorttia, täytyy luoda CLI:n avulla.

Ennen Portainerin asentamista sille on luotava volume. Volumet ovat Dockerin hallitsemia hakemistoja isäntäkoneella, jotka sijaitsevat yleensä polussa /var/lib/docker/volumes/. Volumet ovat helppoja varmuuskopioida tai siirtää palvelimelta toiselle. Volumen ollessa tyhjä, sille asetettu kontti täyttää sen käyttäjän määrittämän kontin sisäisen polun tiedostoilla. Kontin data jatkaa olemassaoloa kontin tuhoamisen jälkeen. Luodakseen kontteja, Portainerin täytyy pystyä keskustelemaan Dockerin palveluprosessin kanssa. Tämä voidaan tehdä bind mountilla. Bind mountilla voidaan liittää tiedosto tai kansio isäntäkoneelta konttiin. Liittämällä Dockerin unix socketin käyttäen bind mountia, kontti voi keskustella isäntäkoneen Dockerin kanssa. Oletuksena kontissa olevaa sovellusta ei voi käyttää verkon kautta, eikä mitään porttia ole julkaistu ulkomaailmalle. Käyttämällä p optiota voidaan ohjata (map) esimerkiksi kontin portin 8000 liikenne isäntä koneen porttiin 8000. Tämän tehtyä sovelluksen web-käyttöliittymä on käytettävissä isäntäkoneen portissa 8000. Kuvassa 16 luodaan volume sekä Portainer kontti.

```
docker volume create portainer_data

docker run -d -p 8000:8000 --name portainer \
--restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v portainer_data:/data  \
portainer/portainer-ce:latest
```

Kuva 16. Portainer kontin luonti

Kloonaamalla kontin, jossa on MIG laite käytössä, voidaan luoda uusi kontti, jolla on MIG laite käytössä. Näin voidaan luoda kontteja, jotka hyödyntävät MIG laitteita Portainerilla.

Jupyterista on olemassa monta erilaista levykuvaa. Levykuvaksi valittiin tensorflow/tensorflow:latest-gpu-jupyter. Tässä levykuvassa koneoppimis ympäristö on asennettu valmiina ja sisältää valmiita esimerkkejä, jotka voivat hyödyntää A100 GPU:ta ilman erillisiä asennuksia tai asetuksien muuttamista.

5.4 Jupyter

Jupyter kontin luonti tapahtuu samoin kuin aiemmin luoduissa konteissa pienillä muutoksilla. Asettamalla detached tilan, runtimen, MIG laitteen, volumen sekä portin, Jupyteriä voisi käyttää tässä vaiheessa, mutta asetuksien muuttaminen ei onnistuisi. Asetuksien muuttaminen voi tapahtua kontin sisällä tai kontin luonti vaiheessa. Sovelluksen asetus tiedostot tässä levykuvassa ovat jääneet luomatta. Käyttämällä e optiota voidaan määrittää ympäristömuuttujia kontille. Jokaisessa kontissa on erilaiset ympäristömuuttujat. Ympäristömuuttujat ovat avain-arvopareja, joilla voidaan vaikuttaa prosessin käytökseen.

Kuvasta 17 nähdään, miten Jupyter kontti luotiin. Antamalla e optiolle JUPYTER_CONFIG_DIR avaimen ja arvona /tf/.jupyter voidaan asettaa Jupyter asetus tiedostojen sijainti arvossa määriteltyyn polkuun. Kontin luonnin komento-osiossa käynnistetään Jupyter serveri kontin sisällä. Jupyterin asetuksia on mahdollista vaihtaa tässä vaiheessa. Optiolla ip määritetään, mitä osoitteita Jupyter kuuntelee. 0.0.0.0 tarkoittaa Jupyterin kuuntelevan kaikkia osoitteita lokaalilla laitteella eli kontissa. No-browser ei avaa selainta, kun Jupyter prosessi käynnistyy. Allow-root ajaa jupyter prosessin root käyttäjänä kontin sisällä. NotebookApp.base_url asettaa Jupyter serverin base url. Base url on osa verkko osoitetta, joka ei muutu kyselyiden välillä.

```
docker run -d --runtime=nvidia --gpus '"device=0:2"' --name
tensorflow-gpu -p 50001:8888 -p 50000:6006 -e
"JUPYTER_CONFIG_DIR=/tf/.jupyter" -v tensorflow:/tf
tensorflow/tensorflow:latest-gpu-jupyter jupyter notebook --ip 0.0.0.0
--no-browser --allow-root --NotebookApp.base_url=/jupyter
```

Kuva 17. Jupyter kontin luonti komento

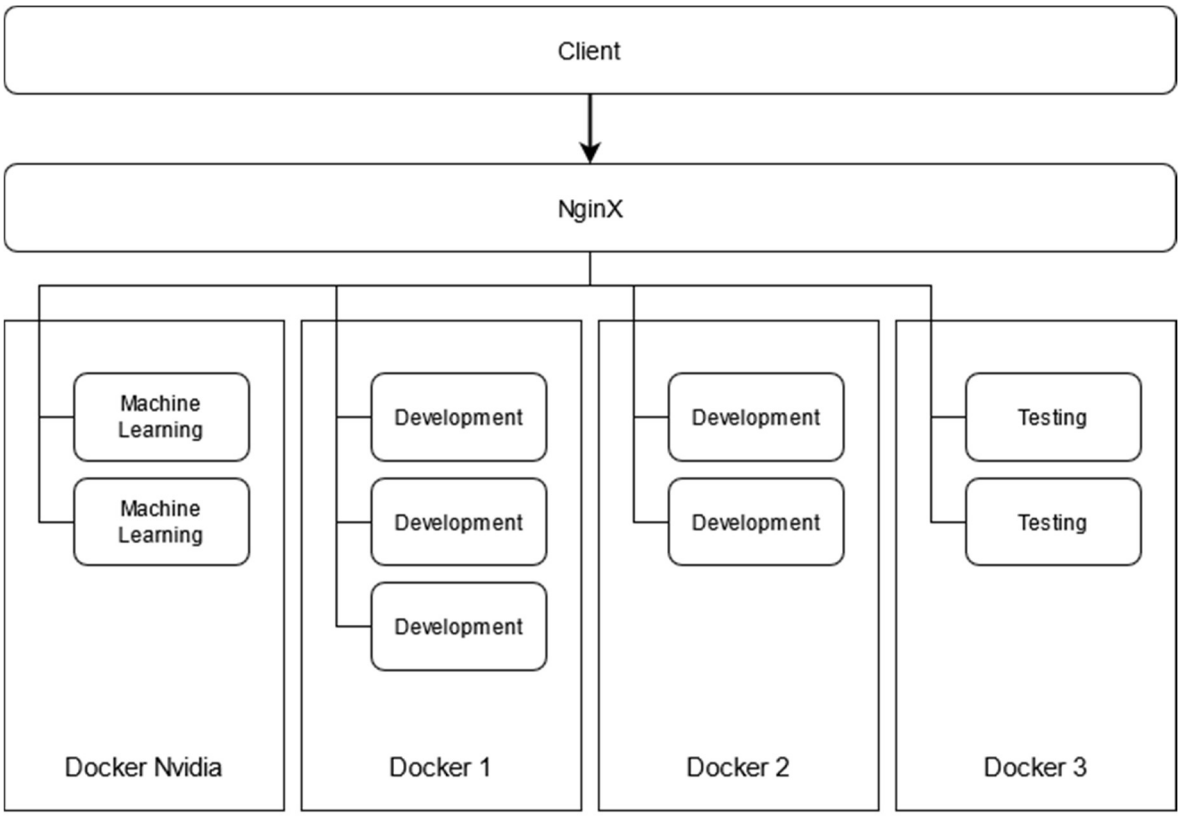
5.5 NginX

Käänteisen välityspalvelimen toteuttamiseen käytettiin NginX. NginX on web palvelin, jolla voidaan toteuttaa esimerkiksi verkkosivujen jakamista verkkoon.

Portti on virtuaalinen piste, jossa verkkoliikenne alkaa ja loppuu (Cloudflare 2021a). Porttien avulla voidaan tunnistaa, mikä sovellus lähettää tai vastaanottaa dataa. Portteja 80 ja 443 käytetään HTTP (Hypertext Transfel Protocol) ja HTTPS (HTTP Secure) pyyntöjen välittämiseen. Tämän tyyppisiin pyyntöihin vastataan tavallisimmin HTML-sivulla. NginX voi varata molemmat portit itselleen ja välittää dataa muista porteista ulkomaailmaan toimimalla välittäjänä datalle. Kuvassa 8 nähdään miten palveluita, joilla on vain IP-osoite, on

mahdollista jakaa nimetyn osoitteen kautta käyttämällä käänteistä välityspalvelinta. Näin asiakkaiden ei tarvitse muistaa IP-osoitteita ja portti numeroita päästäkseen palveluihin selaimessa.

NginX asennettiin LXC konttiin, jotta sille voidaan antaa oma domain. Tämän avulla asiakkaat ottavat NginX palvelimeen yhteyden ja palvelin välittää pyynnöt oikeisiin osoitteisiin. Kuvassa 18 nähdään, miten data liikkuu päätasolla.



Kuva 18. NginX datan kulku päätasolla

Asiakas saapuu järjestelmän tietäen NginX palvelimen osoitteen, joka ohjaa Heimdall palveluun. Heimdall toimii tässä työssä etusivuna, josta löytyy linkit eri palveluihin. Tämä palvelu toimii kuten kirjanmerkit selaimessa. Sille annetaan osoitteita ja niiden takan olevia palveluita kuvastavat nimet. Tämän jälkeen asiakkaan ei tarvitse muistaa kuin NginX palvelimen osoite ja he näkevät visuaalisesti mitä palveluita on tarjottavana.

6 Yhteenveto

Opinnäytetyön tavoitteena oli koneoppimis- ja tekoälysovellusten testaus ja käyttöönotto virtuaalisessa ympäristössä. Tavoitteessa pyrittiin hyödyntämään virtuaalikoneita ja kontitusteknologiaa.

Lopputuloksena saatiin luotua virtuaalikone, jonne asennettiin Docker. Ensimmäinen kontti, joka otettiin käyttöön, oli Portainer. Portainerin avulla voitiin hallita ja monitoroida kontteja sekä korjata mahdollisia virheitä konteissa. Asiakkailla oli mahdollisuus luoda omia kontteja testiympäristöön. Asiakkaat pystyivät hallitsemaan konttejaan samalla sovelluksella.

Opinnäytetyössä saatiin myös asennettua Jupyter koneoppimista ja tekoälyä varten. Hyödyntämällä kiihdytettyä laskenta korttia, koneoppimis mallien kouluttaminen tapahtui huomattavasti nopeammin verrattuna vain prosessorilla tehtyyn kouluttamiseen. Luomalla esimerkkejä, jotka hyödyntävät kiihdytettyä laskentaa ja suorittamalla ne, saatiin varmistettua konttien käyttävän laskenta korttia hyödyntäen NVIDIAn tarjoamaa työkalua NVIDIA laitteiden hallintaan ja monitorointiin.

Työn aikana konfiguroidut sovellukset käyttivät omaa käyttäjän todennus metodia. Tämä johti monien tunnuksien luomiseen eri järjestelmissä. Käyttämällä yhtenevää todennusprotokollaa, voitaisiin luoda yksi tunnus käyttäjää kohden, joka on käytettävissä eri sovellusten välillä. ITKO-hanke loppuu vuoden 2021 lopussa.

Lähteet

Azure. 2021. What is a container. Blogi. Viitattu 19.11.2021. Saatavissa https://azure.microsoft.com/en-us/overview/what-is-a-container/#overview

Calmes, J. Abecassis, F. Ramarao, P. 2018. Enabling GPUs in the Container Runtime Ecosystem. Blogi. Viitattu 20.11.2021. Saatavissa: https://developer.nvidia.com/blog/gpu-containers-runtime/

Chandrasekaran, S. 14.5.2020. Ride the Fast Lane to AI Productivity with Multi-Instance GPUs. Blogi. Viitattu 27.9.2021. Saatavissa https://blogs.nvidia.com/blog/2020/05/14/multi-instance-gpus/

Cloudflare. 2021a. What is a computer port? | Ports in networking. Blogi. Viitattu 24.11.2021. Saatavissa https://www.cloudflare.com/en-ca/learning/network-layer/what-is-a-computer-port/

Cloudflare. 2021b. What is a reverse proxy? | Proxy servers explained. Blogi. Viitattu 19.11.2021. Saatavissa https://www.cloudflare.com/en-ca/learning/cdn/glossary/reverse-proxy/

Docker. 2021a. Docker overview. Dokumentaatio. Viitattu 15.7.2021. Saatavissa https://docs.docker.com/get-started/overview/

Docker. 2021b. Docker Hub Quickstart. Dokumentaatio. Viitattu 19.11.2021. Saatavissa https://docs.docker.com/docker-hub/

Docker. 2021c. Use containers to Build, Share and Run your applications. Blogi. Viitattu 19.11.2021. Saatavissa https://www.docker.com/resources/what-container

Docker. 2021d. Docker run reference. Dokumentaatio. Viitattu 20.11.2021. Saatavissa https://docs.docker.com/engine/reference/run/

Gillis, A. 2021. Docker image. Viitattu 21.11.2021. Saatavissa https://searchitoperations.techtarget.com/definition/Docker-image

HPCAC. 2020. HPC-AI Competition BERT-LARGE Benchmark Guidelines. Dokumentaatio. Viitattu 7.12.2021. Saatavissa https://www.hpcadvisorycouncil.com/events/2020/APAC-AI-HPC/pdf/HPC-AI_Competition_BERT-LARGE_Benchmark_guidelines.pdf

Hui, J. 11.8.2020. AI Chips: A100 GPU with Nvidia Ampere architecture. Blogi. Viitattu 27.11.2020. Saatavissa https://jonathan-hui.medium.com/ai-chips-a100-gpu-with-nvidia-ampere-architecture-3034ed685e6e

IBM Cloud Education. 2021. Containerization. Viitattu 19.11.2021. Saatavissa https://www.ibm.com/cloud/learn/containerization

Gikonyo, J.N. 2021. Linux Containers vs Docker - What is the Difference and Why Docker is Better. Blogi. Viitattu 19.11.2021. Saatavissa https://www.section.io/engineering-education/lxc-vs-docker-what-is-the-difference-and-why-docker-is-better/

Jupyter. 2021a. What is the Jupyter Notebook? Dokumentaatio. Viitattu 22.11.2021. Saatavissa https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/What%20is%20the%20Jupyter%20Notebook.html#Kernels

Jupyter. 2021b. The Jupyter Notebook. Viitattu 24.11.2021. Saatavissa https://jupyter.org/

Krashinsky, R. Giroux, O. Jones, S. Stam, N. Ramaswamy, S. 14.5.2020. NVIDIA Ampere Architecture In-Depth. Dokumentaatio. Viitattu 19.11.2021. Saatavissa https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

NVIDIA. 2021a. NVIDIA Container Runtime. Dokumentaatio. Viitattu 19.11.2021. Saatavissa https://developer.nvidia.com/nvidia-container-runtime

NVIDIA. 2021b. Nvidia Multi-Instance GPU User Guide. Dokumentaatio. Viitattu 27.9.2021. Saatavissa https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html

NVIDIA. 2021c. NVIDIA A100 TENSOR CORE GPU. Viitattu 19.11.2021. Saatavissa https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf

NVIDIA. 2021d. Concepts. Dokumentaatio. Viitattu 19.11.2021. Saatavissa https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/concepts.html

NVIDIA. 2020. NVIDIA A100 Tensor Core GPU Architecture. Viitattu 27.11.2021. Saatavissa https://images.nvidia.com/aem-dam/en-zz/Solutions/data-center/nvidia-ampere-architecture-whitepaper.pdf

Perez, F. 8.7.2014. Project Jupyter. Viitattu 19.11.2021. Saatavissa https://speakerdeck.com/fperez/project-jupyter

Renou, M. 9.1.2019. A new Python kernel for Jupyter. Blogi. Viitattu 19.11.2021. Saatavissa https://blog.jupyter.org/a-new-python-kernel-for-jupyter-fcdf211e30a8

The Jupyter Notebook, 2015. Viitattu 19.11.2021. Saatavissa https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

Whitehead, N. Fit-Florea, A. 2011. Precision & Performance: Floating Point and IEEE 754 Compliance for NVIDIA GPUs. Nvidia. Dokumentaatio. Viitattu: 19.11.2021. Saatavissa: https://developer.nvidia.com/sites/default/files/akamai/cuda/files/NVIDIA-CUDA-Floating-Point.pdf

Zalman, L. 2021. What is the difference between Proxy and Reverse Proxy? | How To Use Reverse Proxy for Access Management Control. StrongDM. Blogi. Viitattu 19.11.2021. Saatavissa https://www.strongdm.com/blog/difference-between-proxy-and-reverse-proxy