Fadi El-khouri

Konttiratkaisu-Nmap-kurssin harjoituksiin

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

1.6.2023

Tiivistelmä

Tekijä:	Fadi El-khouri
Otsikko:	Konttiratkaisu Nmap-kurssin harjoituksiin
Sivumäärä:	28 sivua + 2 liitettä
Aika:	1.6.2023
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Web-kehitys
Ohjaaja:	Osaamisaluepäällikkö Janne Salonen

Tämä insinöörityö on tehty Metropolia Ammattikorkeakoululle. Työssä toteutettiin eristetty harjoitteluympäristö, jossa opiskelijat kykenevät tekemään Nmap-kurssin harjoituksia.

Tarvittavaa aineistoa kerättiin Nmap.org ja Metropolian ammattikorkeakoulun Moodlen Nmap Scanning Basics kursista. Aineiston ja tehtävän vaatimusten perusteella valittiin työssä käytettävät teknologiat ja työkalut Se toteutettiin konttiratkaisulla, jossa on yksi linux-palvelin ja yksi linux työasema. Toimii siten, että opiskelija voisi tehdä Nmap-kurssin harjoituksia lataamalla tämän harjoitteluympäristön itselleen, jossa on asennettu valmiiksi tarvittavat sovellukset.

Toteutetulla eristetty harjoitteluympäristö on useampia jatkokehitysmahdollisuuksia, esimerkiksi seuraavat: Lisätään Zenmap, kun se on yhteensopiva Python 3 kanssa. Yksinkertaisempi Linux työaseman käynnistäminen. Tulevaisuudessa tulee kyseeseen yhdellä komennolla käynnistää koko harjoitteluympäristö.

Avainsanat: Docker, nmap, konttiteknologia, virtuaalikoneet

Abstract

Author:	Fadi El-khouri
Title:	Container solution for exercises of the Nmap course
Number of Pages:	28 pages + 2 appendices
Date:	6 June 2023
Degree:	Bachelor of Engineering
Degree Programme:	Information and communication technology
Professional Major:	Name of the professional major
Supervisor:	Janne Salonen, Head of School (ICT)

This Bachelor´s thesis was commissioned by Helsinki Metropolia University of Applied Sciences. An isolated training environment was implemented in the work, where students are able to do the exercises of the Nmap course.

The necessary material was collected from Nmap.org and the Nmap Scanning Basics course of Metropolia University of Applied Sciences' Moodle. Based on the material and the requirements of the task, the technologies and tools used in the work were chosen. It was implemented with a container solution with one Linux server and one Linux workstation. It works in such a way that the student can do the exercises of the Nmap course by downloading this training environment, where the necessary applications are already installed.

The implemented isolated training environment has several further development possibilities, for example the following: We will add Zenmap when it is compatible with Python 3. Easier to start a Linux workstation. In the future, it will be possible to start the entire training environment with one command.

Keywords: Docker, virtual machine, nmap, container

Sisällys

Lyhenteet

Nmap: *Network Mapper on ilmainen ohjelma verkon etsintään ja tietoturvatarkastukseen.*

NSE: *Nmap Scripting Engine on erittäin tehokas työkalu tietoturva -ja haavoittuvuustarkastukseen.*

Lua: Skriptikieli, joka on laajasti käytössä sovellusohjelmissa ja tietokonepelien sisäisenä skriptinä.

Fedora 37 Fedora Linux on Red hatin kehittämä ja ylläpitämä Linux pohjainen käyttöjärjestelmä. Luku 37 viittaa versionumeroon.

TCP- Transmission Control Protocol avulla tietokoneet voivat kommunikoida luotettavasti keskenään ja pitävät huolta, että tavujono saapuu oikeassa järjestyksess.

UDP Use Datagram Protocol, joka mahdollistaa tiedon siirron, mutta ei vaadi yhteyttä laitteiden välillä.

Datagrammit Yhteydettömän viestintäpalvelun pakettivälitteissessä verkossa. Verkon ei tarvitse huolehtia datagrammin toimituksesta.

SCTP Stream Control Transmission Protocol varmistaa luotettavan peräkkäisen viestien siirron ja sillä on ruuhkanhallinta.

1 Johdanto

Tämän Opinnäytetyön tavoitteena on luoda Nmap Scanning Basics-kurssin suorittajille mahdollisimman turvallinen harjoitteluympäristö. Harjoitteluympäristö koostuu Linux palvelimesta ja Linux työasemasta. Päättötyön yhteydessä perehdyttiin Metropolian Ammattikorkeakoulun Nmap Scanning Basics kurssin materiaaleihin ja sisältöön perusteellisesti, jotta voimme määrittää mitä tulee olla asennettuna valmiiksi harjoitteluympäristöön.

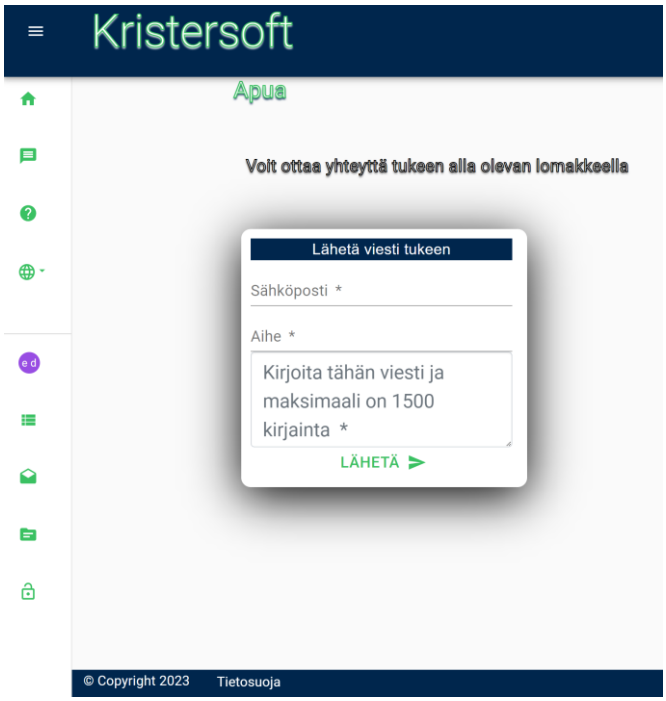
Tähän päättötyöhön kuuluu myös Vaiheittaiset ohjeet opiskelijoiden käyttöön eli miten asennetaan harjoitteluympäristö itselleen ja käytetään.

1.1 Lyhyesti

Nmap C/C++ kielellä kirjoitettu työkalu verkon tutkimiseen ja tietoturvatarkastukseen. Se on verkkoturva-asiantuntijan Gordon ”Fyodor” Lyon luoma ilmaiseen lähdekoodiin perustuva jo vuodesta 1997 Nmap on kehitetty ja paranneltu viimeiset 26 vuotta. Nmap on alun perin kehitetty Linux käyttöjärjestelmille, mutta nykyään sitä on saatavana myös Windows ja mac OS. Nmappiä käynnistetään Linuxissa terminaalissa kirjoittamalla ”sudo nmap”.

1.2 Mihin Nmappiä käytetään

Nmapissa on laaja kirjasto, jonka avulla voimme tiedostella. Listan saa näkyviin kirjoittamalla terminaaliin ”nmap -help”. NSE komentosarjojen avulla voimme tarkistaa järjestelmää hyökkäämällä siihen ja pyrimme etsimään tietoturva aukkoja tai haavoittuvuuksia. Nmappissä on 50 valmista scriptiä, joita voi käyttää ja muokata tarvittaessa Lua:lla.



Kuva 1. Kristersoft sovellus.

Seuraavaksi skannaan minun omaa web-sovellustani, jonka olen itse tehnyt ja kehittänyt mystuff.kristersoft.fi (ks. kuva 1.).

Skannaamalla voi selvittää:

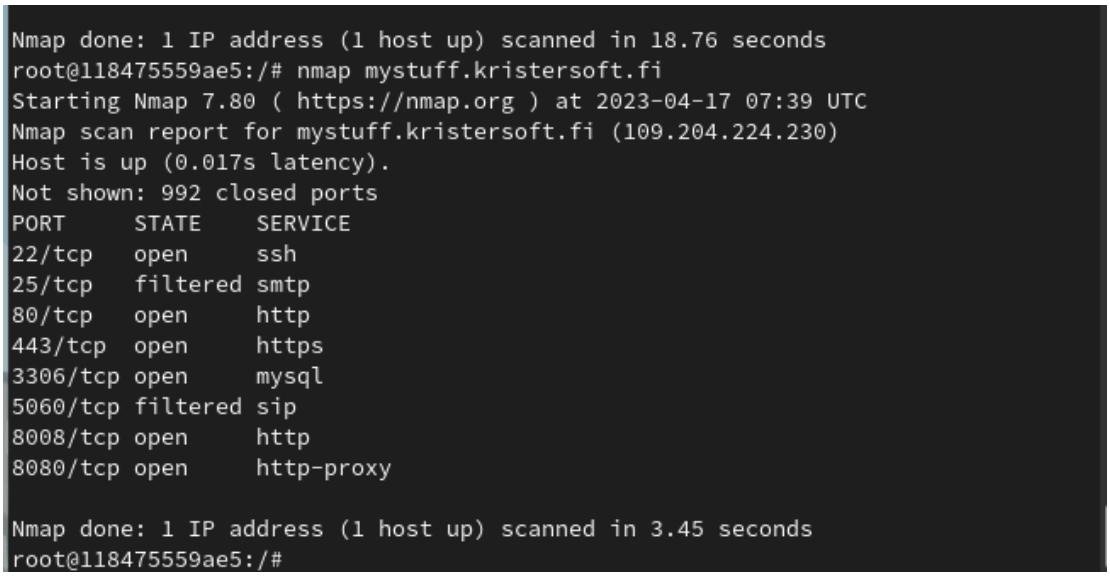
- Mitkä isännät ovat saatavana verkossa.
- Mitä palveluita kyseiset isännät tarjoavat.
- Mitä käyttöjärjestelmiä ne käyttävät.
- Minkä tyyppiset palomuurit ovat käytössä.

Käymme läpi, miten tehdään Quick Port Scanning, jotta saamme yleiskuvan, miten Nmappiä käytetään. Suoritamme Nmappiä kontin sisällä seuraavasti ”nmap mystuff.kristersoft.fi” (ks. Kuva 2).

```
root@118475559ae5:/# nmap mystuff.kristersoft.fi
Starting Nmap 7.80 ( https://nmap.org ) at 2023-04-17 07:39 UTC
```

Kuva 2. Skannaan Nmapillä mystuff.kristersoft.fi sovellusta.

Skannauksen tuloksissa ilmenee skannauksen kohde (mystuff.kristersoft.fi), skannaukseen kuluva aika (3.45 sekuntia), portit ja palvelut (ks. Kuva 3.). Tuloksista kiinnostavimmat ovat taulukosta tila. Monet muut porttiskannerin ovat jakaneet portit avoimeen (open) tai suljettuun (closed) tilaan. Nmapissä portit ovat jakautuneet kuuteen tilaan open (avoin), filtered (suodatettu), closed (suljettu), unfiltered (suodattamaton), open | filtered (avoin | suodatettu) ja closed | filtered (suljettu | suodatettu). Nämä tilat ovat Nmapin oma analyysia eli kertovat millä tavalla tietty portti on Nmapin näkökulmasta.



Kuva 3. Nmapin skannauksen tulokset.

Käsittelemme seuraavaksi, mitä Nmapin tunnistamat tilat tarkoittavat. Portti skannauksen ensisijainen tavoite on löytää avoimet portit ja sulkea ne, jotta vältytään isimme hyökkäyksiltä. Avoin tila tarkoittaa, että sovellus kuuntelee ja hyväksyy TCP-yhteyksiä, UDP- datagrammit tai SCTP-yhteyksiä. Avoimet portit näyttävät käytössä olevia palveluita esimerkiksi kuvassa 4. on avoin portti, jonka numero 3306 ja jossa on käytössä Mysql palvelu.

Kuva 4. Avoimena oleva portti.

Suljettu portti tarkoittaa, että se vastaan ottaa Nmapin koetin paketteja ja vastaa niihin, mutta sovellus ei kuuntele sitä. Niiden tila saattaa muuttua myöhemmin ja siksi järjestelmän kehittäjän olisi syytä estää porttien palomuureilla. Palomuureilla estetyt portit näkyvät suodatettu tilassa, koska Nmap ei pysty määrittelemään onko portti auki. Jos Nmap ei kykene määrittämään, onko portti auki tai kiinni niin se luokitellaan Nmap:ssä suodattamaton. Avoin | suodatettu tila tarkoittaa, että Nmap ei kykene määrittämään, onko portti avoin tai suodatettu ja vastaavasti jos portin tila on suljettu | suodatettu niin Nmap ei pysty määrittämään, onko portti suljettu tai suodatettu.

Kuva 5. Laajempi Nmap skannaus.

Yleensä on ihan riittävää testata sovellusta yksin kertaisella komennolla nmap mystuff.kristersoft.fi, mutta kokeneet Nmapin käyttäjät saattavat hyökätä aggressiivisemmin sovelluksiin käyttämällä kuvan 5 komentoa. Tunnukset tarkoittavat seuraavaa:

-p0- Nmap skannaa kaikki mahdolliset TCP-portit.

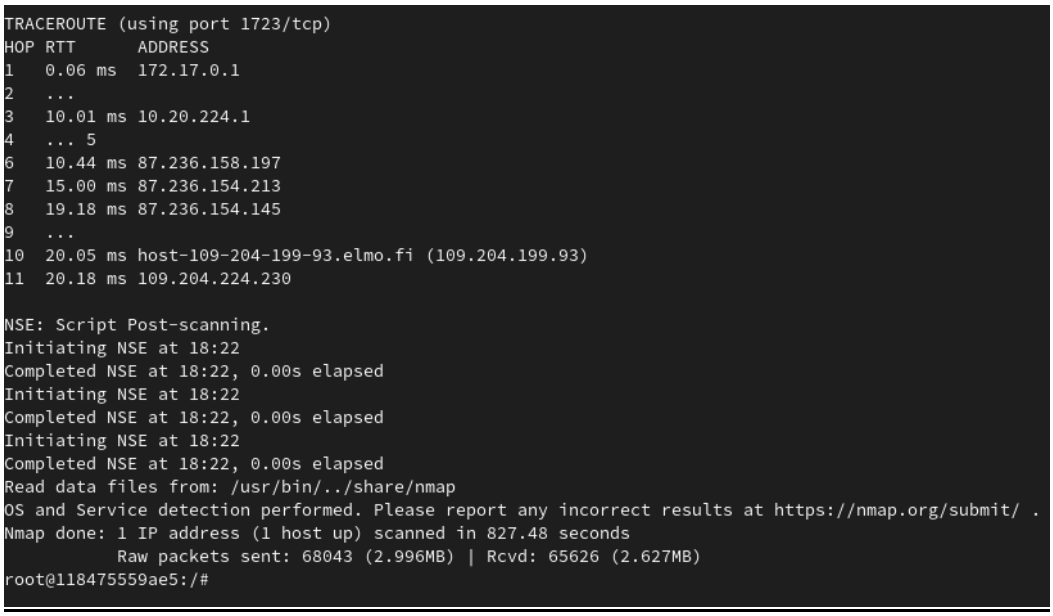
-v Kertoo yksityiskohtaisesti.

-A Mahdollistaa aggressiiviset testit.

-T4 Aggressiivinen skannaus valmistuu nopeammin.

Testattiin sovellustamme kuvan 5 komennon mukaan eli aggressiivisemmin. Saamamme kuvaus on yksityiskohtaisesti lueteltuna kaikki mitä Nmap tekee ja saa selville sovelluksesta. Prosessiin kului aikaa 827,48 sekuntia eli noin 14

minuuttia. Kaikkia tietoja emme laita mukaan, koska ne ovat useamman sivun pituinen raportti, joten emme laita sitä kokonaisuudessa tähän. Osa tuloksista on nähtävänä kuvassa 6:ssä.



Kuva 6. Laajempi Quick Port Scanning tulokset.

Skannauksen mystuff.kristersoft.fi sovelluksen tuloksena huomattiin, että useampi portti on auki ja ne ovat tietoturva riskiä. Meidän pitäisi lisätä turvallisuutta palomuureilla, jotta voisimme suojata näitä portteja hyökkäyksiltä.

Tällä tavalla Nmapiä käytetään, jotta voidaan testata web-sovelluksien tietoturvariskejä ja korjaamaan niitä. Nmapillä on monia muita ominaisuuksia, mutta Quick Port Scanning on Nmapin yleisin käytetty työkalu.

Nmap:ssa aloittelijoilla on käytössä tarvittaessa Zenmap. Se on Nmap-sovelluksen graafinen käyttöliittymä ja siinä on samat ominaisuudet kuin terminaalia käytettäessä. Zenmap:ssä on myös hakuvaihtoehtoja kokeneille käyttäjille. Se on saatavilla useammille käyttöjärjestelmissä esimerkiksi Windows, Linux, Mac OS X ja monet muut, jotka tukevat Nmappiä.

1.3 Nmap on suuressa suosiossa

Nmap on suosittu useasta syystä: Se on yksinkertainen ja helppo käyttää. Siltä löytyy tehoa ja nopeutta skannata suuria verkkoja samaan aikaan ja se toimii myös hyvin yksittäisten isäntien kohdalla. Sillä voidaan tehdä myös haamu palveluja, siten että kysyjä on näkymätön.

2 Konttiratkaisu

Jos emme käytä konttiteknologioita ja haluamme, että sovellus toimii oikein sovelluskehittäjien välillä niin kaikilla jäsenillä pitää olla samat versiot sovelluksista ja konfiguraatioista jne. Konttiteknologioiden tarkoitus on juuri sitä, että ei tarvitse olla samat vaan kaikki kehittäjät saavat ladattua suoraan toimivan sovelluksen ilman, että tarvitsee huolehtia konfiguraatiosta tai sovelluksien asetuksista jne. Konttien rakenteissa ja käyttökohteissa on eroavaisuuksia.

Konttiteknologian tarkoitus on luoda eristetty käyttöjärjestelmän tai sovelluksen. Se on helppo siirtää, muokata ja päivittää. Se on yksi kokonaisuus, jossa on asennettu kaikki tarvittavat ohjelmistot ja työkalut, joita tarvitaan sen sovelluksen tai käyttöliittymän suorittamiseen testausympäristössä tai tuotantoympäristössä. Se on myös turvallinen tapa saada ohjelman toimimaan eri järjestelmissä.

2.1 Docker

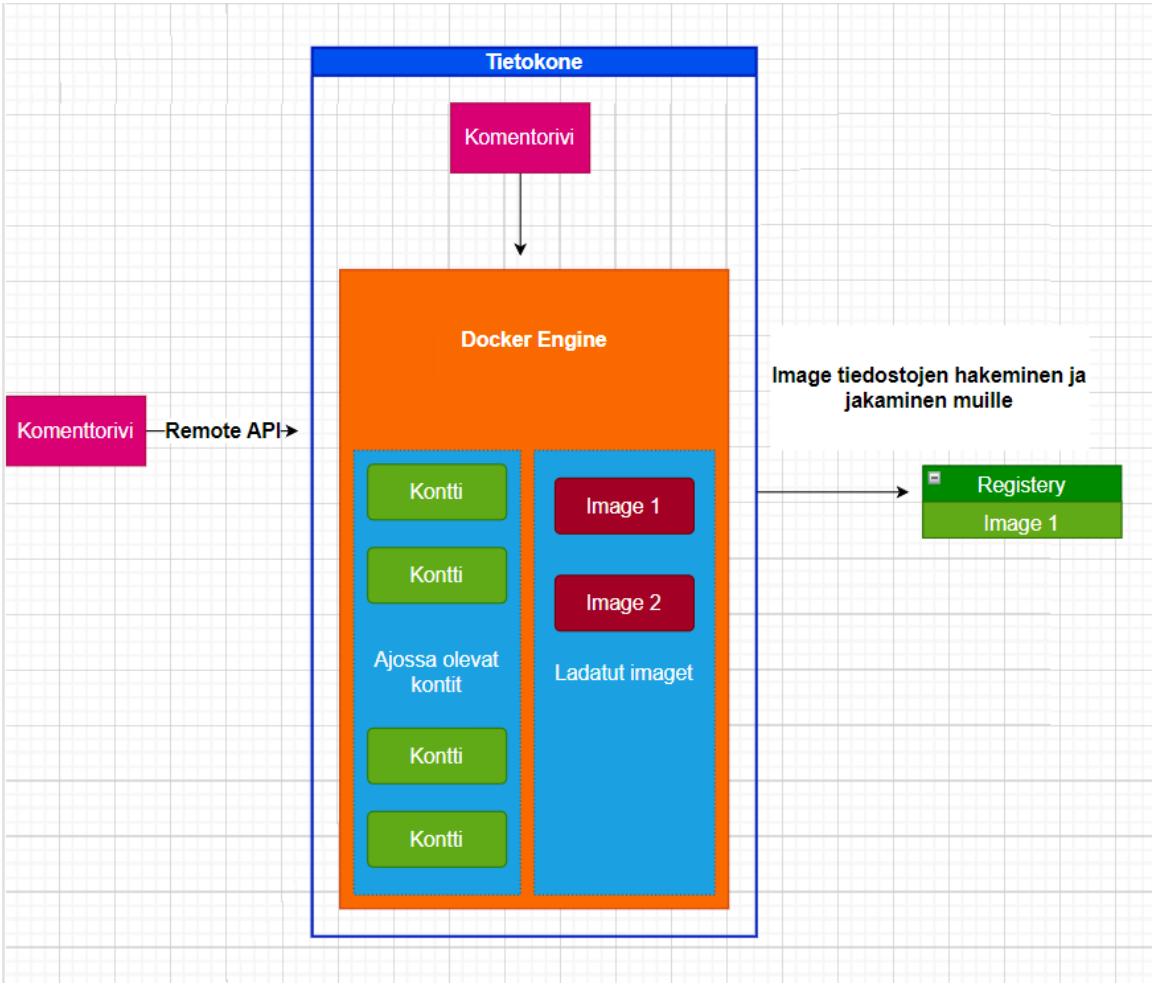
Docker on kehitetty LXC:stä avoimen koodin projekti Linuxille, mutta nykyään sen kehityksestä huolehtii Docker Inc. Sitä on saatavana ilmaisena versiona (Docker Community) tai maksullinen versiota (Docker Enterprise). Docker on saatavana kaikille käyttöjärjestelmille esimerkiksi Windows, Linux ja Mac OS X.

2.2 Dockerin rakenne

Käymme tässä luvussa perusteet Dockerin rakenteesta ja toiminnallisuudesta. Dockeri on pakattu yhteen tiedostoon (container Image), joka koostuu useammasta imagesta päällekkäin. Se sisältää kaiken tarvittavan esimerkiksi web-sovelluksen ja sen tarvittavat ohjelmistot, käyttöjärjestelmä ja konfiguraation (kertoo, miten Docker Imagetta rakennetaan), jotta kyseistä sovellusta pystytään käynnistämään. Sitä voi kuljettaa helposti eri ympäristöissä.

Docker Engine on konttien hallinta ohjelma, joka hallinnoi palvelimessa ajattavia kontteja (ks. Kuva 7). Sitä taas hallitaan Docker API:n kautta, jota voidaan käyttää terminaalissa. Docker Engine kautta voidaan pakata sovelluksen tarvittavat ohjelmat imageksi ja suorittaa ne tarvittaessa.

Rekisteri on imageitten varasto, jossa imaget säilytetään. Rekisterin tarkoitus on varastoida, hallinnoida ja jakaa imageita. Kun halutaan käynnistää kontin, niin kone etsii sitä ensin paikallisesti ja käynnistää sen, jos löytyy ja jos ei löydy niin kone etsii sitä rekisteristä ja jos löytyy, se lataan sen ja käynnistää. Se voi olla yksityinen tai julkinen varasto. Julkinen varasto esimerkiksi Dockerin omistama Docker Hub, jossa on tuhansia ilmaisia imageita, joita voi ladata ilmaiseksi. Yksityinen varasto on sellainen, jossa ei ole muita kuin yrityksen omia imageita.



Kuva 7. Dockerin toiminnan rakenne.

Yhdestä imagesta voidaan käynnistää montaa rinnakkaista konttia (Docker Container). Konteille ei aseteta mitään resursseja. Tarpeen mukaan kontti pyytää käyttöjärjestelmältä resursseja. Halutessamme voidaan käyttää docker compose. Sillä voidaan yhdistää koko sovelluksen esimerkiksi frontend, backend ja tietokanta ja käynistää koko sovelluksen yhdellä komennolla.

2.3 Dockerin hallinta

Dockeria voidaan hallinnoida automaattisesta apu ohjelmistolla, kun tarvitaan useampi palvelin käynnistämään kontteja eli orchestrator. Hallinta ohjelmistoja on monenlaisia ja niissä on paljon eroavaisuuksia, jotta voisimme valita oikean ja juuri sopivan Dockerin hallinta ohjelmistoa niin pitäisi perehtyä ja verrata niitä

keskenään. Valitsemme seuraavat Dockerin hallinta ohjelmistoa, jotka ovat suosiossa esimerkiksi Dockerin oma hallinta ohjelma Docker Swarm tai Googlen kehittämä Kubernetes.

Docker Swarm koostuu useista Docker-isännistä. Se voi olla johtaja, työntekijä tai suorittaa molemmat roolit.

Luodessa palvelun, niin määrität sille tietyn tilan eli annat sille ominaisuuksia esimerkiksi:

- Kopioiden määrän.
- Käytettävissä olevat verkko- ja tallennusresurssit.
- Mitkä portit näkyvät maailmalle.

Docker yrittää säilyttää haluttua tilaa esimerkiksi työntekijä Node ei ole käytettävissä, Docker siirtää kyseisen node tehtävät muihin nodeihin. Nodella tarkoitetaan fyysistä tai virtuaalista palvelinta.

Docker Swarm:llä voidaan luoda ja hallinnoida Docker klustereita. Kluster muodostuu useasta erillisestä koneesta, käyttöjärjestelmistä tai sovelluksista, jotka on yhdistetty toisiinsa. Klusterin käytöllä on paljon hyötyjä konttiteknologiassa seuraavat ominaisuudet:

- Järjestelmien päivitys onnistuu helposti.
- Voidaan hyödyntää avoimen lähdekoodin alustoja.
- Emme ole riippuvaisia laitevalmistajista.
- Varmempaa palveluita ja ympäristön muokkaus.
- Laitteiden kuormitus jaetaan tasan, jotta ei rasiteta yksittäisiä laitteita.

Googlen kehittämä vuonna 2014 julkaisema Kubernetes. Se on kontitettujen sovelluksien ja palveluiden hallinnoimiseen. Se on yksi käytetyimmästä hallintaohjelmasta ja sen käyttäjä määrä on jatkuvassa nousussa.

2.4 Dockerin hyödyt

Tässä luvussa vertaamme Dockerin hyötyjä sovelluskehityksessä ja liiketoiminnalle perinteisiin palvelimiin verrattuna. Docker on suosituin konttiteknologioista, koska se on käyttäjäystävällinen ja se on helppo käyttää.

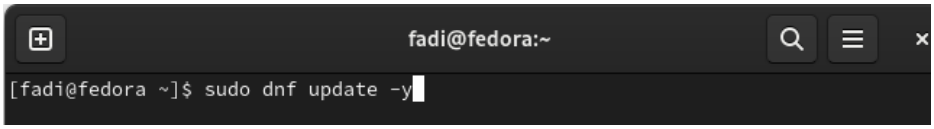
Dockerin käyttö on yleistynyt viimeiset vuodet, koska yrityksille on hyötyä niistä runsaasti. Docker mahdollistaa liiketoiminnalle pienemmät kulut ja sen avulla voimme kehittää sovelluksia nopeammin, jolloin voimme viedä ne markkinoille nopeammin kuin perinteisiä palvelimia verrattuna.

Kontit eivät varaa resursseja vaan pyytävät tarvittaessa ajoa varten ja jakavat yhden yhteisen käyttöjärjestelmän. Kontteihin asennetaan vain sovellukset, joita tarvitaan sovelluksen ajoa varten ja siksi ne ovat erittäin tehokkaita ja nopeita. Docker on kevyt ja helppo siirtää kehittäjien koneilta testiympäristöön ja sieltä tuotantoon, koska se on paketoitu yhdeksi tiedostoksi. Se on valmiskäytettäväksi erilaisissa ympäristöissä huolimatta. Kontit voidaan käynnistää yksittäisinä tai kaikki samanaikaisesti.

3 Harjoitteluympäristön toteutus

Tässä kappaleessa selitämme, miten harjoitteluympäristö asennettiin, testattiin ja myös mahdolliset ongelmat, joita tuli asennuksen yhteydessä vaiheittain. Käytämme tässä työssä Fedora käyttöjärjestelmää.

3.1 Dockerin asennus



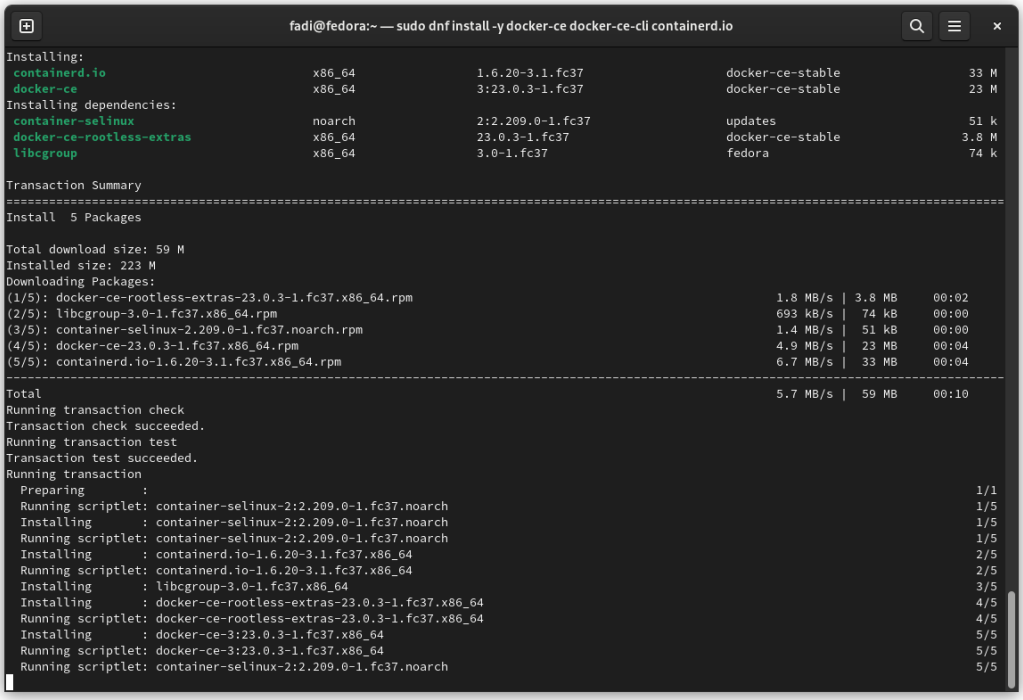
Kuva 8. Fedora käyttöjärjestelmän päivitys.

Asennettiin Docker ja sen tarvittavat kirjastot ja työkalut terminaalia käyttäen. Asennus oli sujuvaa ja ei ollut ongelmia. Käymme tässä luvussa läpi yleistä Dockerin asennuksesta. Tässä on yksi esimerkki, miten terminaalia käytetään kuvan 8 komennolla tarkistamme, että järjestelmä on ajan tasalla.

Selitämme sana kerrallaan, mitä kuvan 8 Komento tarkoittaa, jotta saamme käsityksen, miten komennot muodostetaan ja käytetään seuraavasti:

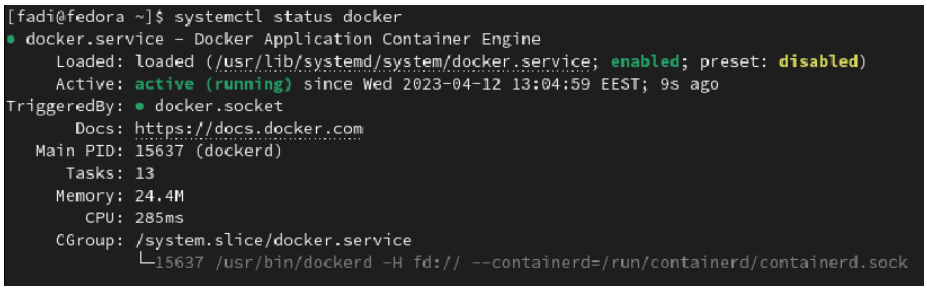
- Annetaan käyttöoikeudet järjestelmänresursseihin (sudo).
- Ohjelmistojenpakettien hallintaohjelma, joka asentaa, päivittää ja poistaa paketteja (dnf).
- Päivitä järjestelmä vastaamalla kehotteisiin kyllä keskeyttämättä prosessia (update -y).

Docker ja sen riippuvuudet voimme asentaa komennolla dnf install / yum install -y docker-ce docker-ce-cli containerd.io (ks. Kuva 9.).



Kuva 9. Dockerin asennuksen aloitus.

Asennettuamme Docker ohjeiden mukaisesti niin huomaamme, että sen tila on aktiivinen (ks. Kuva 10). Dockerin tilan saadaan näkyviin sudo systemctl status docker komennolla.



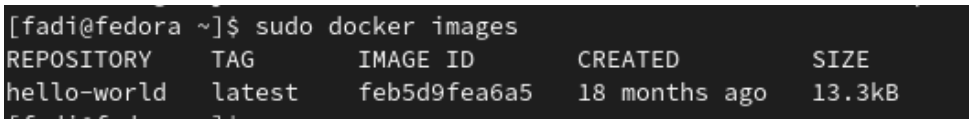
Kuva 10. Dockerin tila.

Jos Käyttämäsi järjestelmä ei käytä systemctl niin voit korvata kaikki komennot systemctl service:llä: Taulukossa 1. on koottu kaikki yleiset Dockerin komennot esimerkiksi käynnistys, pysähdys jne.

Taulukko 1. Dockerin yleiset komennot

Komennot	**Merkitys**
sudo systemctl status docker	Tilan tarkastus
sudo systemctl start docker	Serverin käynnistäminen
sudo systemctl restart docker	Serverin uudelleen käynnistäminen
sudo systemctl stop docker	Serverin pysäyttäminen
sudo systemctl enable --now docker	Koneen käynnistäessä käynnistää serverin
docker --version	Dockerin tiedot

Lopuksi voisimme testata, että juuri asennettu Docker toimii. Lataamme kuvan Docker hub -arkistosta ja kuvasta tehdään kontti, jossa näkyy ”hello world” (ks. kuva 11).



Kuva 11. Listattu kaikki ladatut imageet.

3.2 Luodaan Linux pohjainen käyttöjärjestelmä

Saatuamme Dockerin toimimaan niin seuraavaksi rakennamme oman Docker imageen. Dockeriin asennetaan tarvittavat sovellukset Dockerfile käyttäen. Voimme luoda sen tiedoston käyttäen terminaalia tai helpompi tapa on käyttää teksti editoria ja tässä tilanteessa käytämme Visual Studio Code.

```
FROM fedora
RUN dnf update  -y && dnf upgrade -y
RUN dnf install -y wget
RUN dnf install -y whois
RUN dnf install -y bind-utils
RUN dnf install -y iputils
RUN dnf install -y hping2
RUN dnf install -y ripgrep
RUN dnf install python3
RUN dnf install -y nmap

VOLUME ["/tmp", "/run", "/run/lock"]
ENTRYPOINT ["tail", "-f", "/dev/null"]
```

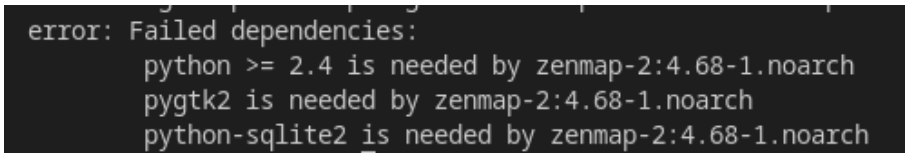
Esimerkkikoodi 1. Sovelluksien asennus Dockeriin.

Työkaluiden käyttö helpottaa merkittävästi Dockerin sovelluksien asennuksessa ja käyttöönotossa. Dockerfile:ssä annetaan peräkkäisiä komentoja, joita suoritetaan siinä järjestyksessä, missä ne ovat (ks. Esimerkkikoodi 1.). Käytettiin tässä pohjana Moodlen Nmap Scanning Basics kurssin sisältöä, jotta selvitetään, mitä työkaluista tarvitaan. Fedora:ssa asennetaan sovelluksia ja työkaluita käyttäen dnf install komentoa tai yum install. Käymme seuraavaksi, joitakin komentoja ja selitämme, mitä merkitystä näiden työkaluilla on. Nmap:n esimerkeissä on työkaluista, joita ei löydy asennettuina Fedora:sta valmiina ja jotta helpottaaksemme opiskelijoitten esimerkkien seuraaminen niin päätettiin asentaa valmiiksi tarvittavat työkalut.

Taulukko 2. Asennettujen sovelluksien ja työkaluiden tarkoitus

Työkalut	**Tarkoitus**
wget	Tiedostojen lataukseen
whois	Tietojen keruuseen
Python3	Nmap tarvitsee python3 toimiakseen
bind-utils	host työkalu, jos selvitetään isäntä
nmap	Nmap sovellus

Dockerfile FROM tarkoittaa, että mistä aloitetaan sen rakentamista ja tämän Docker aloitus on Fedora pohjainen. Pohjan määrityksen jälkeen annamme RUN:lla, mitä sovelluksia asennamme ja missä järjestyksessä. Se on tärkeä, koska asentamamme sovellukset ovat riippuvaisia muista sovelluksista, joten ennen kuin tietty sovellus asennetaan, niin sovellus tarkistaa, että onko asennettu kaikki sovellukset, joista se on riippuvainen. Docker luo jokaisesta rivistä oman kontin ja jos haluaa samaan konttiin, niin pitäisi kirjoittaa komennot samalle riville. Voimme rakentaa Dockeria commenolla ”docker build -t <nimi> <piste>”.



Kuva 12. Zenmap virhe ilmoitus.

Yritin noin viikon asentaa Zenmap, joka on Nmap:n käyttöliittymä. En onnistunut asentamaan toimivaksi, niin poistin sen dockerfile:sta. Totesin, että se ei ole yhteensopiva Nmapin kanssa, koska se vaatii ainakin Python 2.4, mutta ei toimi Python 3 kanssa ja viimeisin versio Nmapistä vaatii Python 3 (ks. Kuva 12). Nmapin harjoituksia pystytään tekemään ilman Zenmappiä ja tässä Dockerissa ei ole asennettuna sitä.

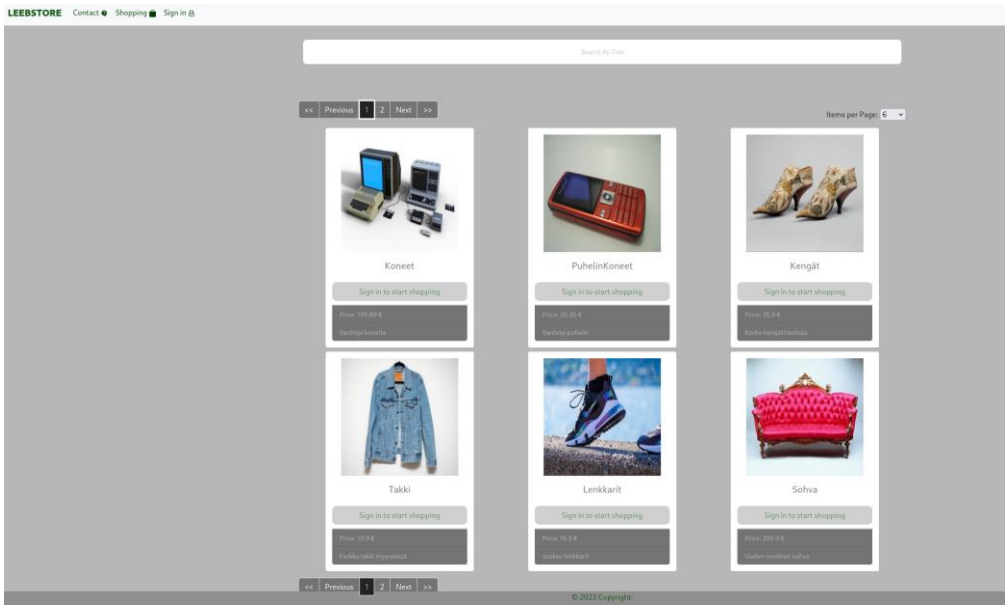
Yritin lähestyä ongelmaa seuraavasti:

1. Googletin ongelman.
2. Kokeilin eri käyttöjärjestelmiä alustoina.
3. Asensimme useamman python version samaan Dockeriin.

Googlettamalla ja löysin useamman ratkaisu Stack overflow:sta, jotka saattaisivat ratkaista ongelman, mutta valitettavasti eivät toimineet. Yritin myös eri Linux pohjaisista alustoista rakentaa dockerfile esimerkiksi Kali Linux, Ubuntu,Debian ja CentosOS jospa Zenmap toimisi. Eri alustoiden yhteydessä asensin useamman Python version 2.7 ja 3 ja tarvittavat työkalu. Yritin myös pakottaa, että Zenmap käyttäisi Python 2.7, mutta ei se toiminut. Sain silti saman virheilmoituksen huolimatta (ks. Kuva 12), mikä järjestelmä on Dockerin alustana tai mitä keinoja yritin.

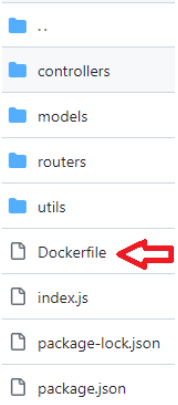
3.3 Luodaan Kontit sovellukselle

Tässä kappaleessa kerron, miten sovelluksesta tehtiin kontit eli jaoin sovellusta moneen konttiin ja määriteltiin, miten kommunikoivat keskenään.



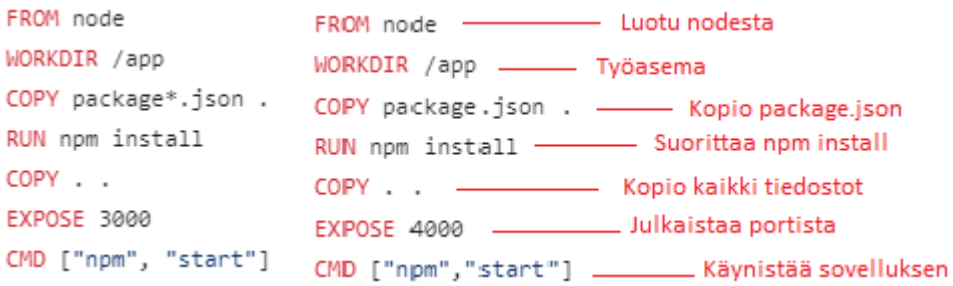
Kuva 13. Linux pohjainen server Leebstore sovellus.

Olen tehnyt sovelluksen, joka on rakennettu frontend React hookilla, backend Node js:lla ja tietokanta Mongodb:llä. Siihen on myös autentikointia eli varmistetaan, että olet kirjautunut järjestelmään ja että tietyllä käyttäjällä on oikeuksia tehdä muutoksia päivittää, lisätä ja poistaa tuotteita. Sovelluksessa käyttämäni kuvilla on Creative Commons -käyttöluvat eli kuvat saa käyttää vapaasti ja en riko tekijä oikeuksia. Frontend:lle ja backend:lle luodaan omat Dockerfile-tiedostot. Nämä tiedostot luodaan niiden pohjatiedostoihin (ks. Kuva 14.). Käytämme valmista tietokanta image Mongodb, jonka määritämme Docker compose -tiedostossa.



Kuva 14. Dockerfile:lle tiedoston luonti pohjatiedostoon.

Kyseessä sovellukselle luotu dockerfile. Meidän pitää määrittää esimerkiksi, mitä tiedostoja kopioidaan mukaan, missä portissa sovellus pyörii ja lopuksi pitää käynnistää sovellusta (ks. Kuva 15).



Kuva 15. Dockerfile frontend:lle (vasen) ja backend:lle (oikea).

Ensiksi rekisteröidään Docker hub:n käyttäjätunnuksella elkhfad ja seuraavaksi luodaan docker-compose.yml -tiedosto, joka luodaan koko projectin pohjatiedostoon. Docker-compose:lla yhdistämme frontend, backend ja Mongodb:n toisiinsa (ks. Liite 1(1)).

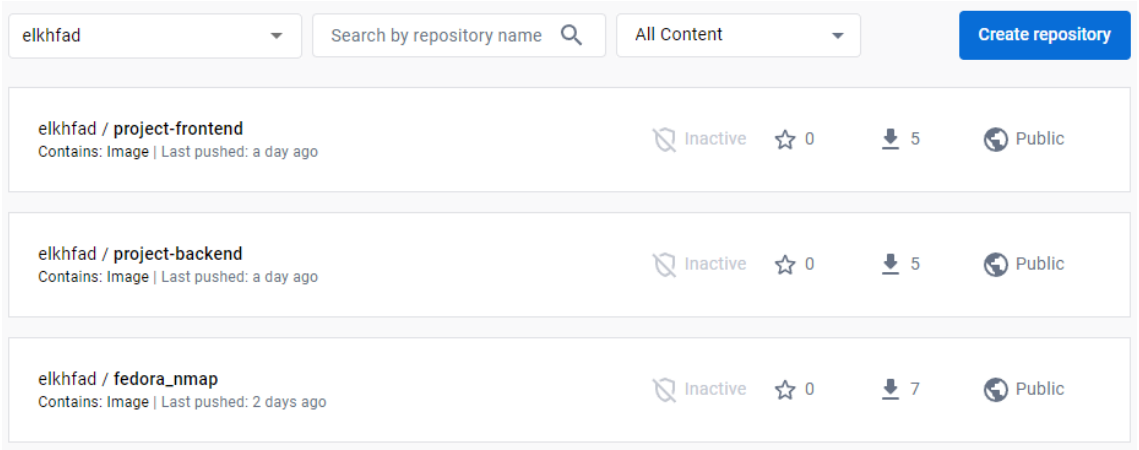
Käydään seuraavaksi, joitakin komentoja läpi. Volumes:lla voimme määrittää esimerkiksi, missä on tietokanta ja mihin se laittaa ne kontissa. Image:lla "elkhfad/project-frontend" docker hubin käyttäjän nimi/repositorion nimi eli siinä määrittelemme, dockerille mihin nämä pusketaan Docker hubissa. Depend komennolla määrittelemme, että tämä kontti on riippuvainen tästä toisesta kontista. Build komennolla määrittelemme, mistä Docker- compose hakee dockerfilelit ja luo niistä kontit.

3.4 Kontit Docker hubiin

Tässä kerron, miten laitoin molemmat serverit Docker hubiin.

Siirsin ensiksi Linux palvelimen ja sitten Linux työaseman seuraavasti:

1. Kirjauduttiin Docker hubiin komennolla: sudo docker login.
2. Rakennetaan sovelluksista kontit komennolla: sudo docker-compose build --pull.
3. Pusketaan sovellusta sudo docker hubiin komennolla: sudo docker-compose push.
4. Luodaan dockeria komennolla sudo docker build -t elkhfad/fedora_nmap.
5. Työaseman puskettiin komennolla: sudo docker push elkhfad/fedora_nmap.
6. Kirjaudutaan ulos komennolla: sudo docker logout.



Kuva 15. Kontit ovat varastoituneena docker hubissa.

Tein samat vaiheet useampaan kertaa, jotta kaikki määritykset toimivat. Tämä vaihe oli työläs, koska tein pieniä muutoksia tiedostoihin ja sitten suoritin yllä mainitut vaiheet uudestaan niin kauan, että projekti pyörii ilman virheitä ja jotta varmistan, että kaikki toimii (ks. Kuva 15).

4 Harjoitteluympäristön käyttöönotto

Annan tässä luvussa ohjeita, miten ladataan ja käytetään vaiheittain ja mahdolliset virheet asennuksen aikana.

4.1 Dockerin asennus ja käynnistys

Ladataan ja käynnistetään dockeria seuraavasti:

1. Ensiksi asentakaa Docker kahdella tavalla a tai b

    a. Kappaleen 4.1 Dockerin asennus.

    b. Käytössä jokin muu kuin Fedora käyttöjärjestelmä niin kehottaisin katsomaan, miten asennetaan Docker osoitteesta https://docs.docker.com/get-docker/ .

2. Varmista, että docker on päällä komennolla systemctl status docker.

```
[fadi@fedora Metropolia_nmap_project]$ sudo systemctl status docker
○ docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset:
     Active: inactive (dead)
TriggeredBy: ○ docker.socket
       Docs: https://docs.docker.com
```

Kuva 16. Docker inactive eli ei ole päällä.

3. Jos on inactive niin laittakaa päälle docker systemctl start docker (ks.Kuva 16).

4. Tarkista onko Docker päällä komennolla docker systemctl status (ks. Kuva 17).

```
[fadi@fedora ~]$ systemctl status docker
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; preset: disabled)
     Active: active (running) since Wed 2023-04-12 13:04:59 EEST; 9s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
   Main PID: 15637 (dockerd)
      Tasks: 13
     Memory: 24.4M
        CPU: 285ms
     CGroup: /system.slice/docker.service
             └─15637 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
```

Kuva 17, Docker on active eli on päällä.

4.2 Ladataan Linux palvelin

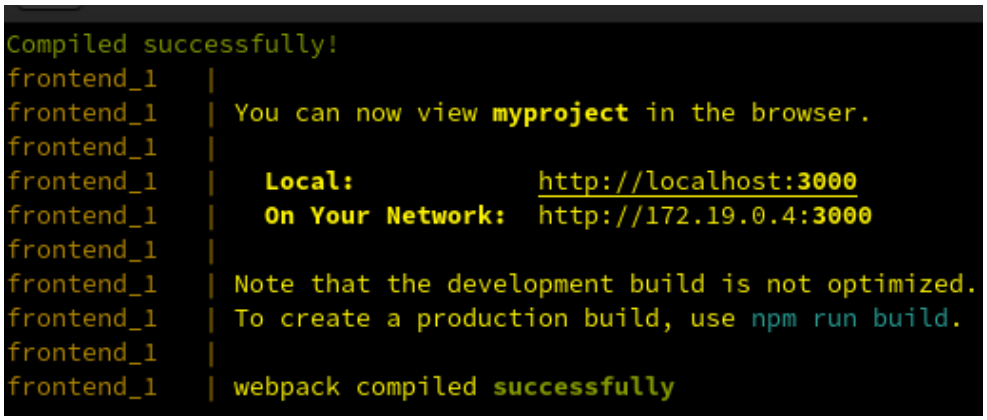
Ladataan ja käynnistetään Linux palvelin seuraavasti:

1. Hae tiedosto git hubista osoitteesta https://github.com/elkhfad/Metropolia_nmap_project ja lataa.

2. Siirry lataamaasi tiedostoon Metropolia_nmap_project ja suorita komento sudo docker-compose up.

3. Hetken kuluttua pitäisi näkyä (ks. Kuva 18.).

4. Jatkossa riittää, että menee siihen Metropolia_nmap_project tiedostoon ja kirjoittaa sudo docker-compose up ja jos haluaa sammuttaa kirjoittaa sudo docker-compose down tai ctrl + c.



Kuva 18. Linux palvelin on käynnistynyt.

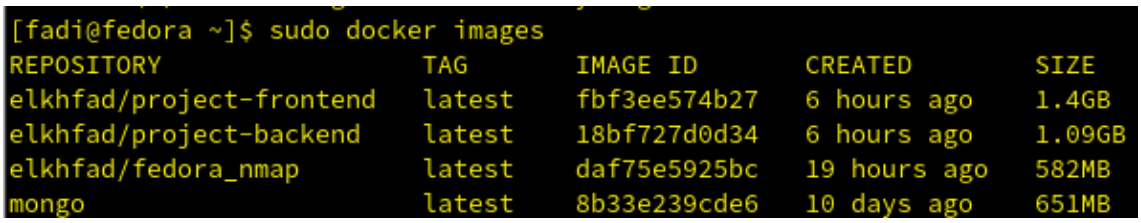
Kuvassa 18 on ilmoitus, että sovellus on käynnissä paikallisesti osoitteessa http://localhost:3000. Paina osoitteeseen niin pitäisi avata uusi selain (ks. Kuva 13).

4.3 Miten lataamme Linux työpöydän

Ladataan ja käynnistetään Linux työasema seuraavasti:

1. Kirjoittamalla terminaaliin sudo docker pull elkhfad/fedora_nmap.

2. Kirjoita seuraavaksi sudo docker images (ks. Kuva 19).



Kuva 19. Dockerin kuvakkeet.

3. Käynnistetään kuvake komennolla:

   sudo docker run -d elkhfad/fedora_nmap tail -f /dev/null, jotta pidetään kontti ajossa.

4. Kirjoita komento: sudo docker ps, tämä komento näyttää ajossa olevat kontit ja jos haluaa nähdä kaikki niin sudo docker ps -a.

Kuva 20. Kontin sisällä.

5. Terminaaliin komento sudo docker exec -it <containerin id> /bin/bash, tämän komennon perusteella pääset suoraan kontitn sisälle (ks. Kuva 20).

6. Kontin sammutetaan sudo docker stop <containerin id>

7. Vastaavasti käynnistetään sudo docker start <containerin id>, ja sen jälkeen komento sudo docker exec -it <containerin id> /bin/bash. Jos ei tiedä, mikä se id on katso kuvaa 20 ja siellä on sarake Container ID (01460729f6cf). Kuvassa 20 ”image” sarakkeen kohdassa on kontin nimi ja jos kontti on käynnissä, lukee sarakkeen status kohdassa up ja aika ja jos ei ole niin exited ja aika perässä.

8. Poistu kontista kirjoittamalla exit.

4.4 Miten voimme käyttää Nmappi

Tässä vaiheessa kerromme, miten voitte käyttää Linux serveriä (sovellus) ja Linux käyttöjärjestelmän keskenään.

1. Käynnistä seuraavaksi sovellusta. Mene tiedostoon Metropolia_nmap_projec ja kirjoita komento sudo docker-compose up (ks. Kuva 18).

2. Avaa uusi terminaali ja kirjoita komento sudo docke ps -a ja etsi sieltä kontin elkhfad/fedora_nmap status. Jos status on up niin ei tarvitse tehdä mitään, mutta jos status on exited niin käynnistä konttia komennolla sudo docker start container id (ks. Kuva 20).

3. Jotta päästään käyttämään Linux työpöytää niin kirjoita sudo docker exec -it /bin/bash.

4. Linux työpöytä ja serveri pitäisi olla nyt käynnissä.

```
[root@bff8a7ef42ef /]# nmap 172.19.0.1
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-15 17:45 UTC
Nmap scan report for 172.19.0.1
Host is up (0.0000080s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE
3000/tcp open  ppp
4000/tcp open  remoteanything

Nmap done: 1 IP address (1 host up) scanned in 0.11 seconds
```

Kuva 21. Testataan toimiiko Nmap paikallisesti.

5. Käytetään Linux työpöytää komennolla nmap 172.19.01 (ks. Kuva 21). Olemme liittäneet komennon ”nmap -p0 -v -A -T4 172.19.01” tulokset (ks. Liite 1 (2)).

Netti on täynnä eri komentoja, joita voi käyttää, mutta jotkut niistä loivat minulle uuden kontin ja minä en halunnut sitä. Minun mielestäni tuo mainitsemani komennot ovat hyvä siksi, että ne pitää sen kontin käynnissä, vaikka poistuisit siitä.

5 Yhteenveto

Onnistuttiin toteuttamaan konttiratkaisun, jossa on yksi Linux pohjainen-palvelin ja työasema. Käytännössä toimii siten, että opiskelija voisi tehdä Nmap-kurssin harjoituksia. Lataamalla docker composen tiedoston git hubista ja sen kautta asentaa itselleen Linux palvelimen. Linux aseman voi ladata suoraan docker hubista. Harjoitteluympäristössä on asennettu valmiiksi tarvittavat sovellukset ja työkalut.

Kehityssuunnat ovat laajat mistä voisi lähteä kehittämään, mutta koko harjoitteluympäristön käynnistäminen yhdellä komennolla voisi olla ensisijainen tavoite. Sen voisi toteuttaa monella eri tavalla, mutta helpoin sen voisi tehdä tekemällä skriptin, joka käynnistää molemmat Linux palvelin ja työaseman samanaikaisesti. Seuraavassa versiossa voisi harkita, jos Zenmap on päivitetty niin voisi tuoda sen tähän eristettyyn työympäristöön mukaan.

Työ oli haastava, mutta mielenkiintoista, kun asiat sujuivat. Valitsin tämän aiheen, koska minulla ei ole Nmapistä tai konttiteknologioista aikaisempaa osaamista. Tässä opinnäytetyössä pääsin tutustumaan ja oppimaan paljon uutta. Konttiteknologioista ja tietoturvallisuudesta. Ne ovat nykypäivää ja työelämässä niiden osaamista tarvitaan jatkuvasti lisää. Lisäksi uskon, että näiden uusien teknologioiden oppiminen lisää minulle mahdollisuus kehittyä urallani.

Lähteet

1 Lyon, Gordon. 2009. Nmap Network Scanning. https://nmap.org/book/toc.html

2 SCTP Overview. 2021. https://www.juniper.net/documentation/us/en/software/junos/gtp-sctp/topics/topic-map/security-gprs-sctp.html

3 Lua. 2022. https://www.lua.org

4 Wallenius, Niklas. 23.2.2022. Mikä on Docker ja mitä hyötyä siitä on? 2022. https://niklaswallenius.fi/mika-on-docker/

5 Docker.2023. https://docs.docker.com/get-started/overview/

6 Suominen, Tuomas.2022. Sovelluskonttien hallinta. Satakunnan ammattikorkeakoulu. Thesus-tietokanta

7 Docker Docs.2023. Docker Swarm. https://docs.docker.com/engine/swarm/key-concepts/#nodes

8 Kubernetes.io.2020. https://kubernetes.io/docs/concepts/

Tiedoston sisältö

Tähän liitteeseen on laitettu koko Docker-compose.yml tiedoston sisällön, jotta sitä voisi tarkastella ja tutkia.

#docker compose versio on 3.8

```
version: "3.8"

# services: lista konteista
services:
  # mongodb nimi
  mymongodb:
    image: "mongo"
    container_name: mymongodb
    # Julkinen mongodb portti
    ports:
      - 27017:27017
    # pitää meidän data
    volumes:
      - /home/leebstores/mongodb/database:/data/db

  backend:
    build: "./backend"
    image: "elkhfad/project-backend"
    ports:
      - 4000:4000
    extra_hosts:
      - host.docker.internal:host-gateway
    depends_on:
      - mymongodb
    environment:
      - SE-
CRET=0c63a75b845e4f7d01107d852e4c2485c51a50aaaa94fc61995e71bbee983a2ac
3713831264adb47fb6bd1e058d5f004
      - MONGODB_URI=mongodb://mymongodb:27017/leebstores
      - PORT=4000

  frontend:
    build: "./frontend"
    image: "elkhfad/project-frontend"
    ports:
      - 3000:3000
    stdin_open: true
    volumes:
       ./frontend/src:/app/src
    tty: true
    extra_hosts:
      - host.docker.internal:host-gateway
    depends_on:
      - backend

volumes:
  data:
```

Esimerkkikoodi 1. Docker compose tiedosto.

Koko Nmap Quick Port Scanning tulokse

```
[root@bff8a7ef42ef /]# nmap -p0 -v -A -T4 172.19.0.1
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-15 17:47 UTC
NSE: Loaded 155 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 17:47
Completed NSE at 17:47, 0.00s elapsed
Initiating NSE at 17:47
Completed NSE at 17:47, 0.00s elapsed
Initiating NSE at 17:47
Completed NSE at 17:47, 0.00s elapsed
Initiating Ping Scan at 17:47
Scanning 172.19.0.1 [4 ports]
Completed Ping Scan at 17:47, 0.01s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 17:47
Completed Parallel DNS resolution of 1 host. at 17:47, 0.01s elapsed
Initiating SYN Stealth Scan at 17:47
Scanning 172.19.0.1 [1 port]
Completed SYN Stealth Scan at 17:47, 0.01s elapsed (1 total ports)
Initiating Service scan at 17:47
Initiating OS detection (try #1) against 172.19.0.1
Retrying OS detection (try #2) against 172.19.0.1
Initiating Traceroute at 17:47
Completed Traceroute at 17:47, 0.01s elapsed
Initiating Parallel DNS resolution of 1 host. at 17:47
Completed Parallel DNS resolution of 1 host. at 17:47, 0.01s elapsed
NSE: Script scanning 172.19.0.1.
Initiating NSE at 17:47
Completed NSE at 17:47, 0.00s elapsed
Initiating NSE at 17:47
Completed NSE at 17:47, 0.00s elapsed
Initiating NSE at 17:47
Completed NSE at 17:47, 0.00s elapsed
Nmap scan report for 172.19.0.1
Host is up (0.00012s latency).

PORT  STATE  SERVICE VERSION
0/tcp closed unknown
Too many fingerprints match this host to give specific OS details
Network Distance: 1 hop

TRACEROUTE (using port 0/tcp)
HOP RTT     ADDRESS
1   0.06 ms 172.19.0.1

NSE: Script Post-scanning.
Initiating NSE at 17:47
Completed NSE at 17:47, 0.00s elapsed
Initiating NSE at 17:47
Completed NSE at 17:47, 0.00s elapsed
Initiating NSE at 17:47
Completed NSE at 17:47, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 2.94 seconds
           Raw packets sent: 33 (2.664KB) | Rcvd: 22 (1.992KB)
```