Virtualizacija na razini operacijskog sustava

Vrban, Marko

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marko Vrban

VIRTUALIZACIJA NA RAZINI OPERACIJSKOG SUSTAVA

DIPLOMSKI RAD

Varaždin, 2022.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marko Vrban

Matični broj: 48803/20–R

Studij: Informacijsko i programsko inženjerstvo

VIRTUALIZACIJA NA RAZINI OPERACIJSKOG SUSTAVA

DIPLOMSKI RAD

Mentor/Mentorica:

Izv. Prof. dr. sc. Ivan Magdalenić

Varaždin, rujan 2022.

Marko Vrban

Izjava o izvornosti

Izjavljujem da je moj završni/diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

_____

Sažetak

Kako sam naslov rada govori u radu će biti obrađena tema virtualizacije na razini operacijskog sustava. Kroz rad biti će napravljen kratki uvod u samu virtualizaciju na razini operacijskog sustava, što se pokušava njome postići te kada se koristi. Zatim će se proći kroz razne metode virtualizacije na razini operacijskog sustava te će biti detaljnije opisani neki od alata koji služe za virtualizaciju na razini operacijskog sustava. Tako će kroz rad osim metoda biti opisani alati Docker, Kubernetes i Podman.

U radu biti će prikazan proces virtualizacije na razini operacijskog sustava tako da će se dvije manje aplikacije koje zajedno čine web aplikaciju pokrenuti unutar kontejnera(eng. containers) koji predstavljaju jednu od metoda virtualizacije na razini operacijskog sustava.

Ključne riječi: Virtualizacija, Operacijski sustav, Linux, kernel, kontejneri, virtualne zone, virtualne jezgre, Docker, Docker hub

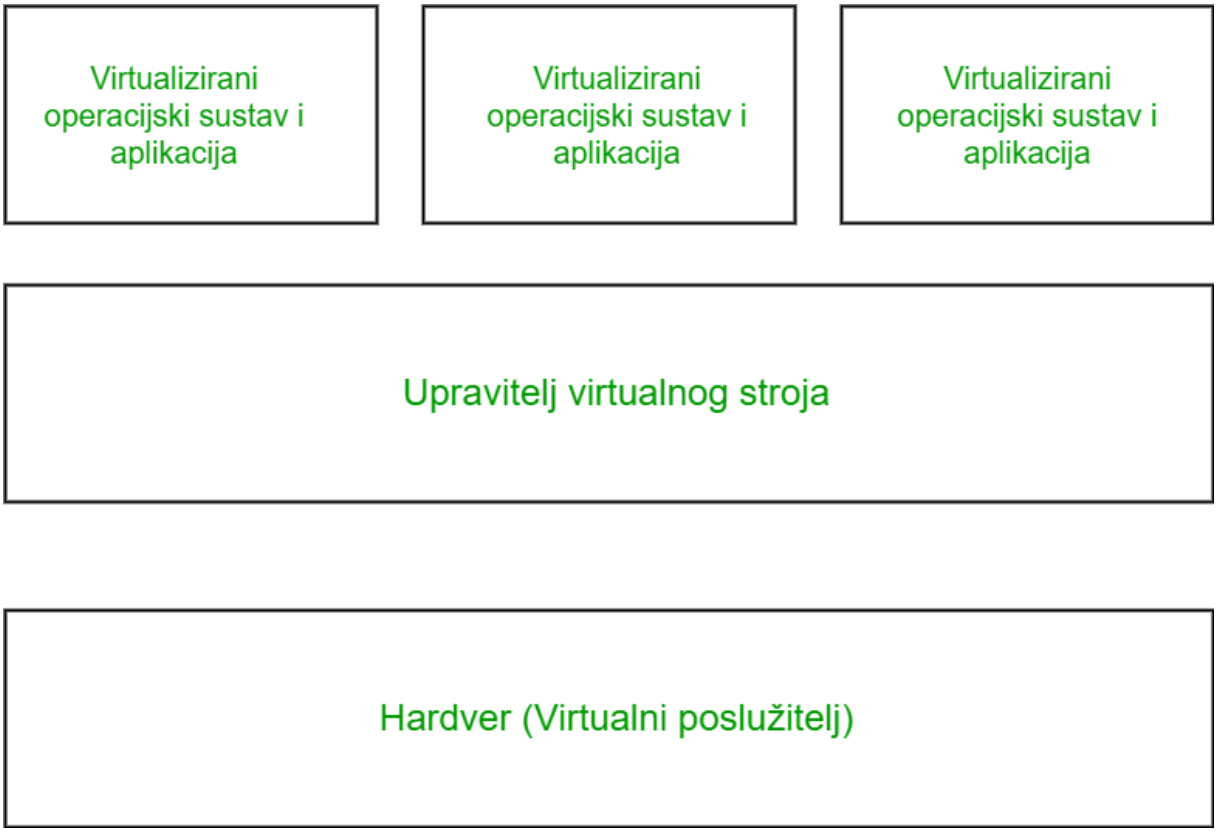
Sadržaj

1. Uvod

Virtualizacija temeljena na operacijskom sustavu odnosi se na značajku operativnog sustava u kojoj jezgra, eng. kernel omogućuje postojanje različitih izoliranih instanci korisničkog prostora. (Slika 1.) Virtualizacija operativnog sustava (virtualizacija OS-a) tehnologija je virtualizacije poslužitelja koja uključuje prilagođavanje standardnog operativnog sustava tako da može pokrenuti različite aplikacije kojima upravlja više korisnika na jednom računalu u isto vrijeme. Operativni sustavi se međusobno ne ometaju iako se nalaze na istom računalu. [1]

U virtualizaciji OS-a operativni sustav se mijenja tako da radi kao nekoliko različitih, pojedinačnih sustava. Virtualizirano okruženje prihvaća naredbe od različitih korisnika koji pokreću različite aplikacije na istom računalu. Virtualizirani operacijski sustav zasebno obrađuje korisnike i njihove zahtjeve. Instalacija softvera za virtualizaciju također se odnosi na virtualizaciju temeljenu na operativnom sustavu. Instalira se preko prethodno postojećeg operativnog sustava i taj se operativni sustav naziva glavni operativni sustav. [1] Takve instance mogu biti:

- kontejneri(LXC, Solaris, Docker, Podman) [1]
- zone(eng. Solaris containers) [1]
- virtualni privatni serveri(OpenVZ) [1]
- particije [1]
- virtualne okoline [1]
- virtualne jezgre [1]
- jails(FreeBSD, chroot jail) [1]



Slika 1. Prikaz virtualnih instanci pomoću virtualizacije na razini operacijskog sustava (Vlastita izrada, 2022.)

Sa stajališta programa koji se u njima izvode takve instance izgledaju kao prava računala. Računalni program koji radi na običnom operativnom sustavu može vidjeti sve resurse (povezane uređaje, datoteke i mape, mrežna dijeljenja, snagu CPU-a, mjerljive hardverske mogućnosti) tog računala. Međutim, programi koji se izvode unutar spremnika mogu vidjeti samo sadržaj spremnika i uređaje koji su dodijeljene tom spremniku, odnosno kontejneru. Izraz spremnik, iako se najčešće odnosi na virtualizacijske sustave na razini OS-a, ponekad se dvosmisleno koristi za označavanje potpunijih okruženja virtualnih strojeva koja rade u različitim stupnjevima usklađenosti s OS-om domaćina, npr. Microsoftovi Hyper-V spremnici. U ovoj vrsti virtualizacije korisnik instalira softver za virtualizaciju u operacijskom sustavu svog sustava kao i svaki drugi program i koristi vezanu aplikaciju za rad i generiranje različitih virtualnih okruženja(eng. Virtual machine) te im dodjeljuje resurse kojima će imati pristup. Na uobičajenim operacijskim sustavima, računalni program može vidjeti (iako možda ne može pristupiti) sve resurse sustava. [2]

U te resurse spada sljedeće:

- Pristup mreži, memoriji i jezgri računala(eng. CPU) [2]
- Podaci koji se mogu čitati ili pisati, kao što su datoteke, mape i mrežna dijeljenja [2]
- Periferni uređaji, poput miša, tipkovnice, web-kamere, pisača, skenera ili faksa [2]

Operacijski sustav ima mogućnost dopuštanja ili uskraćivanja pristupa takvim resursima na temelju kojih ih program zahtijeva i korisničkog računa u kontekstu kojeg se pokreće. Operacijski sustav također može sakriti ove resurse, što dovodi do toga da kada ih računalni program izlistava, oni se ne pojavljuju u rezultatima popisivanja. Ipak, iz perspektive programiranja, računalni program je bio u interakciji s tim resursima, a operativni sustav je upravljao procesom interakcije. Pomoću virtualizacije operacijskog sustava ili kontejnerizacijom, programi se pokreću unutar spremnika(eng. container), kojima su dodijeljeni samo dijelovi ovih resursa. Program od kojeg se očekuje da percipira cijelo računalo, nakon što se pokrene unutar spremnika, može vidjeti samo dodijeljene resurse i vjeruje da su oni sve što je dostupno. Virtualizacija temeljena na operacijskom sustavu je zahtjevna te može povećati zahtjeve i probleme povezane s performansama. [2] Neki od takvih problema mogu biti sljedeći:

- Potreba za većim korištenjem računalne jezgre, memorije i ostalih resursa [2]
- Pozivi povezani s hardverom iz operativnih sustava za goste trebaju prolaziti brojnim slojevima do i od hardvera, što smanjuje ukupne performanse [2]
- Potreba za licencama, kao za temeljni operacijski sustav tako i za svaku virtualnu instancu [2]

2. Metode i tehnike rada

U ovom dijelu rada biti će prikazani i ukratko objašnjeni alati koji su bili korišteni kod pisanja ovog završnog rada. Za razradu teme i pisanje rada korišteni su sljedeći alati:

- **Microsoft Word 2019** – alat koji je bio korišten za pisanje samog rada, oblikovanje teksta i slika

- **Opera browser** – alat koji je bio korišten za pristupanje online sadržaji i literaturi potrebnoj za izradu rada

- **Microsoft Visual Studio Code** – alat koji je bio korišten za pisanje PHP i Python koda

- **Oracle Virtual Machine** – alat korišten za kreiranje virtualne mašine za Linux Mint na kojem su rađeni neki od primjera

- **Docker** – alat koji je bio korišten za kreiranje i pokretanje kontejnera za web aplikaciju korištenu u projektu

- **Mendeley desktop** – alat koji je bio korišten za pravilno navođenje literature i citiranje

3. Načini virtualizacije operacijskog sustava

Virtualizacija na razini operacijskog sustava obično se koristi u okruženjima virtualnog hostinga, gdje je korisna za sigurnu dodjelu ograničenih hardverskih resursa velikom broju korisnika koji nisu povjerljivi. Administratori sustava također ga mogu koristiti za konsolidaciju poslužiteljskog hardvera premještanjem usluga na zasebnim hostovima u spremnike na jednom poslužitelju. Drugi tipični scenariji uključuju odvajanje nekoliko programa u odvojene spremnike za poboljšanu sigurnost, neovisnost o hardveru i dodatne značajke upravljanja resursima. Implementacije virtualizacije na razini operacijskog sustava sposobne za migraciju također se mogu koristiti za dinamičko balansiranje opterećenja spremnika između čvorova u klasteru. [3]

Virtualizacija na razini operacijskog sustava obično nameće manje troškova od hardverske virtualizacije jer programi u virtualnim particijama na razini operacijskog sustava koriste normalno sučelje za pozive operacijskog sustava i ne moraju biti podvrgnuti emulaciji ili pokretanju u srednjem virtualnom stroju, kao što je slučaj s hardverskom virtualizacijom (kao što je VMware ESXi, QEMU ili Hyper-V) i paravirtualizacijom (kao što je Xen ili korisnički način Linuxa). Ovaj oblik virtualizacije također ne zahtijeva hardversku podršku za učinkovit rad. Virtualizacija na razini operacijskog sustava nije tako fleksibilna kao drugi pristupi virtualizaciji budući da ne može ugostiti gostujući operacijski sustav koji se razlikuje od glavnog ili različitu gostujuću jezgru. Na primjer, s Linuxom su različite distribucije u redu, ali drugi operativni sustavi poput Windowsa ne mogu biti ugoščeni. Operacijski sustavi koji koriste sistematiku varijabilnog unosa podliježu ograničenjima unutar virtualizirane arhitekture. [3]

Solaris djelomično nadilazi gore opisano ograničenje svojom značajkom označenih zona, koja pruža mogućnost pokretanja okruženja unutar spremnika koji emulira stariju verziju Solarisa 8 ili 9 u hostu Solarisa 10. Zone s markom Linux (nazivaju se zone s markom "lx") također su dostupne na sustavima Solaris temeljenim na x86, pružajući kompletan korisnički prostor za Linux i podršku za izvođenje Linux aplikacija, Solaris pruža pomoćne programe potrebne za instalaciju Red Hat Enterprise Linux 3.x ili CentOS 3.x Linux distribucija unutar "lx" zona. Međutim, 2010. zone s markom Linux uklonjene su iz Solarisa; u 2014. ponovno su uvedeni u Illumos, koji je open source Solaris fork, podržavajući 32-bitne Linux kernele. Neke implementacije pružaju mehanizme kopiranja na pisanje (CoW) na razini datoteke. (Najčešće se standardni datotečni sustav dijeli između particija, a one particije koje mijenjaju datoteke

automatski stvaraju vlastite kopije.) Ovo je lakše za sigurnosno kopiranje, prostorno učinkovitije i jednostavnije za predmemoriju od kopiranja na razini bloka - pisanje shema uobičajenih za virtualizatore cijelog sustava. Virtualizatori cijelog sustava, međutim, mogu raditi s ne-nativnim datotečnim sustavima i stvarati i vraćati snimke cijelog stanja sustava. [3]

3.1. Kontejneri

Kontejneri(eng. Container) su lagani softverski paketi koji sadrže sve ovisnosti potrebne za izvođenje sadržane softverske aplikacije. Te ovisnosti uključuju stvari kao što su sistemske biblioteke, vanjski paketi kodova trećih strana i druge aplikacije na razini operativnog sustava. Kontejneri su izolirani jedan od drugoga i sadrže vlastiti softver, biblioteke i konfiguracijske datoteke; mogu međusobno komunicirati preko dobro definiranih kanala. Budući da svi kontejneri dijele usluge jedne jezgre operativnog sustava, koriste manje resursa od virtualnih strojeva. Ovisnosti uključene u spremnik postoje na razinama stoga(eng. stack) koje su više od operativnog sustava. Na temelju toga aplikacije upakirane u kontejnere(eng. container) se brzo i pouzdano mogu prebacivati iz jednog računalnog okruženja u drugo. Dobre strane kontejnera(eng. container) su brzina iteriranja, budući da su spremnici lagani i uključuju samo softver visoke razine, vrlo ih je brzo mijenjati i ponavljati, te robusni ekosistem kojega nude. Robusni ekosistem se odnosi na to da većina sustava za izvođenje spremnika nudi hostirano javno spremište unaprijed napravljenih spremnika. Ova spremišta spremnika sadrže mnoge popularne softverske aplikacije poput baza podataka ili sustava za razmjenu poruka i mogu se trenutno preuzeti i izvršiti, čime se štedi vrijeme razvojnim timovima. Loša strana kontejnera(eng. Container) je iskorištavanje dijeljenog hosta zato što svi kontejneri(eng. Container) dijele isti temeljni hardverski sustav ispod sloja operativnog sustava, moguće je da bi eksploatacija u jednom kontejneru mogla izaći iz kontejnera i utjecati na dijeljeni hardver. Najpopularnija okruženja kontejnera imaju javna spremišta unaprijed izgrađenih kontejnera. Postoji sigurnosni rizik u korištenju jedne od ovih javnih slika jer one mogu sadržavati eksploatacije ili mogu biti osjetljive na otimanje zlobnih aktera. [4][5] U nastavku ovog poglavlja govorit će se o nekim najpoznatijim alatima za kreiranje i pokretanje kontejnera kao što su:

- Docker [4]
- Linux Containers ili LXC [4]
- Podman [4]
- Kubernetes [4]

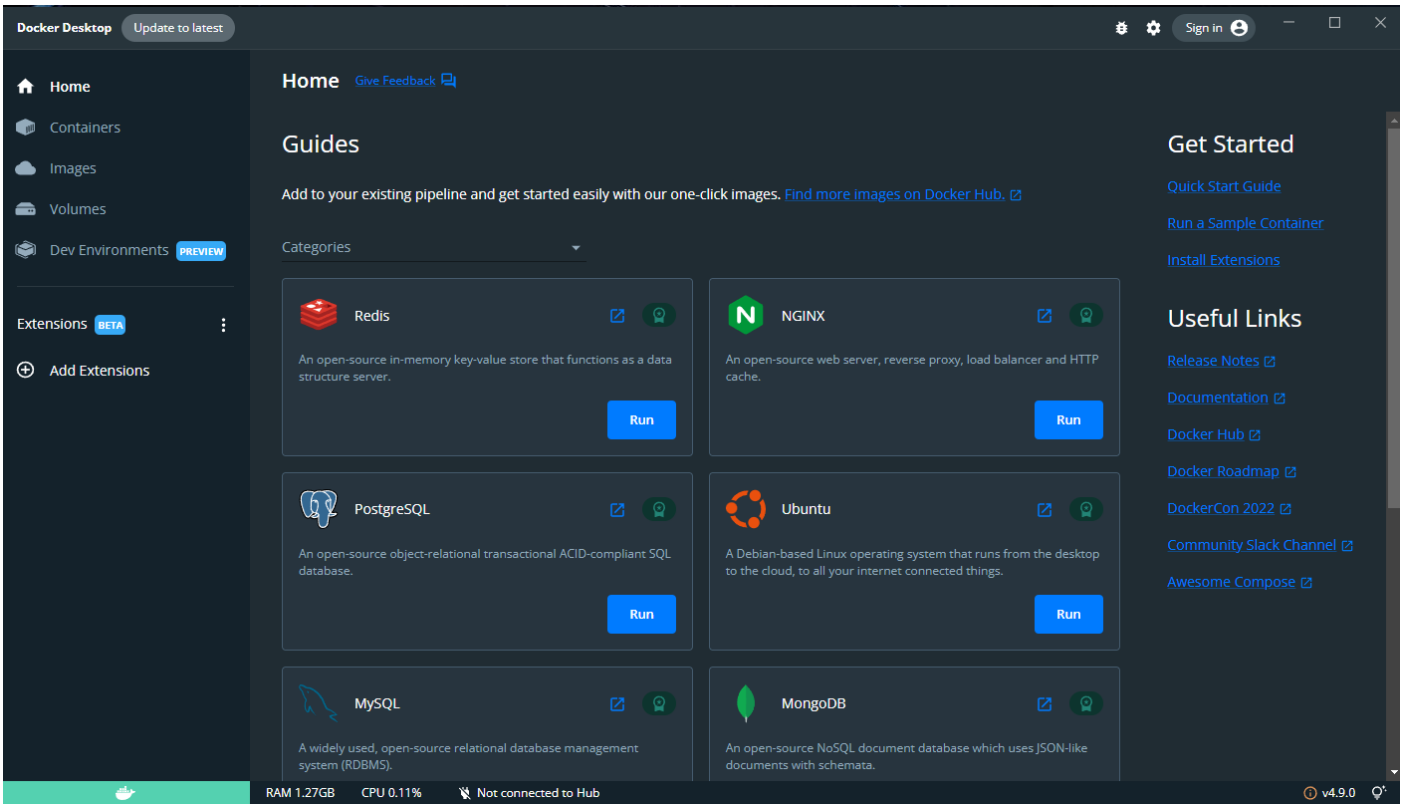
3.1.1. Docker

Docker(Slika 2.) je skup proizvoda platforme kao usluge (PaaS) koji koriste virtualizaciju na razini OS-a za isporuku softvera u paketima koji se nazivaju spremnici ili kontejneri(eng. container). Usluga ima besplatne i premium razine. Softver koji hostira kontejnere zove se Docker Engine. Prvi put je pokrenut 2013. godine, a razvio ga je Docker, Inc. [5]

Slika 2. Docker logo ([5])

Docker može pakirati aplikaciju i njezine ovisnosti u virtualni kontejner koji može raditi na bilo kojem Linux, Windows ili MacOS baziranom računalu. To omogućuje aplikaciji da se izvodi na različitim lokacijama, kao što je lokalno, javno (pogledajte decentralizirano računalstvo, distribuirano računalstvo i računalstvo u oblaku) ili u privatnom oblaku(eng. Private cloud). Kada radi na Linuxu, Docker koristi značajke izolacije resursa Linux kernela (kao što su cgroups i kernel namespaces) i datotečni sustav sposoban za uniju (kao što je OverlayFS) kako bi omogućio rad spremnika unutar jedne Linux instance, izbjegavajući režijske troškove pokretanja i održavanja virtualnih strojeva. Docker na macOS-u koristi Linux virtualni stroj za pokretanje spremnika. Budući da su Docker kontejneri lagani(eng. lightweight), jedan poslužitelj ili virtualni stroj može pokrenuti više kontejnera istovremeno. Analiza iz 2018. pokazala je da tipični slučaj upotrebe Dockera uključuje pokretanje osam kontejnera po poslužitelju(eng. Host), a da četvrtina analiziranih organizacija pokreće 18 ili više kontejnera po poslužitelju(eng. Host). Podrška Linux kernela za imenske prostore(eng. namespace) uglavnom izolira pogled aplikacije na operativno okruženje, uključujući procesna stabla, mrežu, korisničke ID-ove i montirane datotečne sustave, dok kernelove cgroups pružaju ograničenje resursa za memoriju i CPU. [5] Docker softver sastoji se od tri komponente:

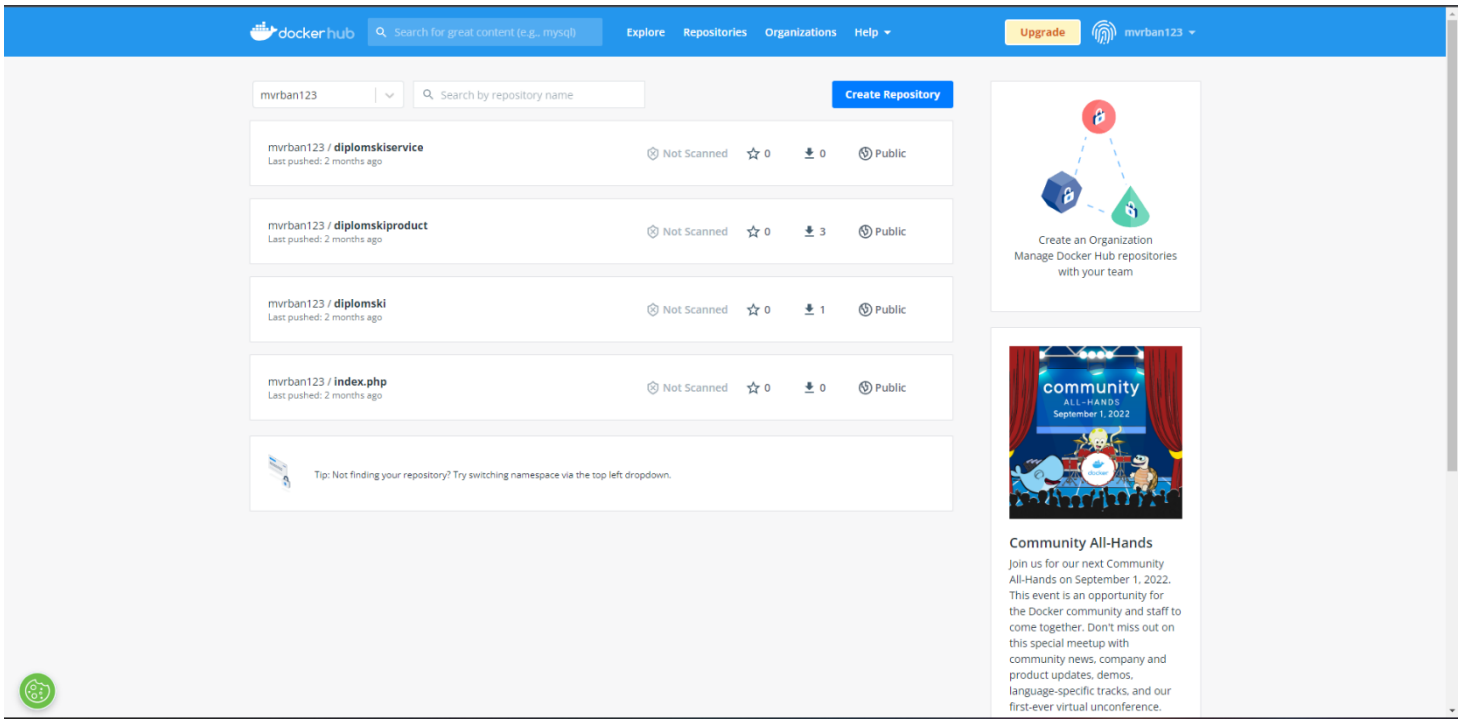
- softvera - Docker daemon, nazvan dockerd, postojan je proces koji upravlja Docker kontejnerima i obrađuje objekte kontejnera. Daemon sluša zahtjeve poslane preko Docker Engine API-ja. Docker klijentski program(Slika 3.), nazvan docker, pruža sučelje naredbenog retka (CLI) koje korisnicima omogućuje interakciju s Docker demonima. [5]



Slika 3. Korisničko sučelje Docker aplikacija, (Vlastita izrada, 2022)

- objekata - Docker objekti su različiti entiteti koji se koriste za sastavljanje aplikacije u Dockeru. Glavne klase Docker objekata su slike(eng. images), kontejneri(eng. container) i usluge. Docker kontejner(eng. container) je standardizirano, enkapsulirano okruženje koje pokreće aplikacije. Kontejnerom se upravlja pomoću Docker API-ja ili CLI-ja. Docker slika(eng. image) je predložak samo za čitanje koji se koristi za izradu kontejnera. Docker slike se koriste za pohranu i slanje aplikacija. Docker usluga omogućuje skaliranje kontejnera na više Docker daemona. Rezultat je poznat kao roj(eng. swarm), skup daemona koji surađuju i komuniciraju putem Docker API-ja. [5]

- registrara - Docker registar je spremište za Docker slike. Docker klijenti povezuju se s registrima za preuzimanje(eng. pull) slika za korištenje ili učitavanje (eng. push) slika koje su izradili. Registri mogu biti javni i privatni. Glavni javni registar je Docker Hub(Slika 4.). Docker Hub je zadani registar u kojem Docker traži slike. Docker registri također omogućuju kreiranje obavijesti na temelju događaja. [5]



Slika 4. Docker hub sučelje (Vlastita izrada, 2022)

Postoje još tri bitna docker alata koja vrijedi opisati a oni su sljedeći:

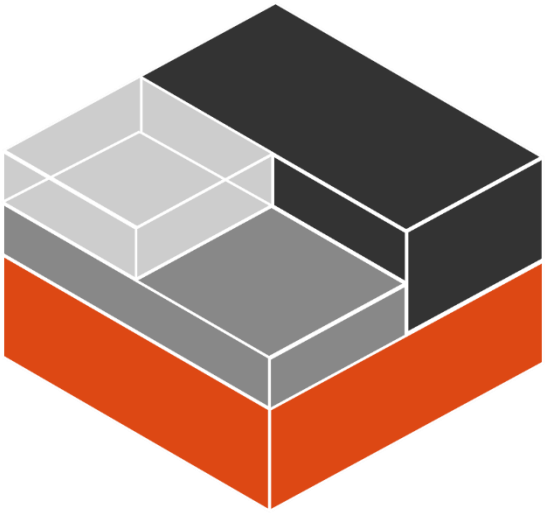
- Docker Compose - alat za definiranje i pokretanje Docker aplikacija s više kontejnera. Koristi YAML datoteke za konfiguriranje usluga aplikacije i izvršava proces stvaranja i pokretanja svih spremnika jednom naredbom. Docker-compose CLI uslužni program omogućuje korisnicima pokretanje naredbi na više kontejnera odjednom, na primjer, stvaranje slika, skaliranje kontejnera, pokretanje kontejnera koji su zaustavljeni i više. Naredbe koje se odnose na manipulaciju slikama ili korisničke interaktivne opcije nisu relevantne u Docker Composeu jer se odnose na jedan kontejner. Datoteka docker-compose.yml koristi se za definiranje usluga aplikacije i uključuje različite opcije konfiguracije. Na primjer, opcija izgradnje definira opcije konfiguracije kao što je Dockerfile putanja, opcija naredbe omogućuje nadjačavanje zadanih naredbi Dockera i više. [5]

- Docker Swarm - pruža nativnu funkcionalnost klasteriranja za Docker kontejnere, što pretvara grupu Docker engine-a u jedan virtualni Docker mehanizam. U Dockeru verziji 1.12 i novijim, Swarm mod je integriran s Docker Engine-om.[35] Uslužni program docker swarm CLI omogućuje korisnicima pokretanje Swarm kontejnera, stvaranje tokena za otkrivanje, popis čvorova u klasteru i još mnogo toga. Docker node CLI uslužni program omogućuje korisnicima pokretanje različitih naredbi za upravljanje čvorovima u roju(eng. cluster), na primjer, ispisivanje čvorova u roju(eng. cluster), ažuriranje čvorova i uklanjanje čvorova iz roja(eng. cluster). Docker upravlja rojevima(eng. cluster) koristeći Raft konsenzusni algoritam. Prema Raftu, da bi se ažuriranje izvršilo, većina Swarm čvorova mora se složiti oko ažuriranja. [5]
- Docker Volume - olakšava neovisnu postojanost podataka, dopuštajući da podaci ostanu čak i nakon što se spremnik izbriše ili ponovno stvori. [5]

U ovom dijelu detaljnije je opisan alat Docker za virtualizaciju na razini operacijskog sustava. U kasnijem poglavlju biti će pomoću ovog alata prikazana uporabu na stvarnom primjeru.

3.1.2. Linux Containers, LXC

Linux Containers (LXC) (Slika 5.) je metoda virtualizacije na razini operacijskog sustava za pokretanje više izoliranih Linux kontejnera na kontrolnom hostu koristeći jedan Linux kernel. Linux kernel pruža funkcionalnost cgroups koja omogućuje ograničavanje i prioritizaciju resursa (CPU, memorija, blok I/O, mreža, itd.) bez potrebe za pokretanjem virtualnih strojeva, kao i funkciju izolacije imenskog prostora(eng. namespace) koja omogućuje potpuni izolirani pogled aplikacije na radnu okolinu, uključujući stabla procesa, umrežavanje, korisničke ID-ove i montirane datotečne sustave. LXC kombinira cgroups kernel i podršku za izolirane imenske prostore(eng. namespace) kako bi pružio izolirano okruženje za aplikacije. Rane verzije Dockera koristile su LXC kao upravljački program za izvršavanje spremnika, iako je LXC postao izboran u v0.9, a podrška je ukinuta u Dockeru v1.10. Reference na Linux spremnike obično se odnose na Docker spremnike koji rade na Linuxu. [6]



Slika 5. Linux Containers logo ([6])

LXC(Linux Containers) je sličan drugim tehnologijama virtualizacije na razini operacijskog sustava na Linuxu kao što su OpenVZ i Linux-VServer, kao i onima na drugim operativnim sustavima kao što su FreeBSD jails, AIX Workload Partitions i Solaris Containers. Za razliku od OpenVZ-a, LXC radi u originalnome(eng. vanilla) Linux kernelu ne zahtijevajući nikakve dodatne zakrpe za primjenu na izvorima kernela. Verzija 1 LXC-a, koja je objavljena 20. veljače 2014., dugoročno je podržana verzija i namijenjena je za pet godina. [6]

3.1.3. Podman

Podman(Slika 6.) je izvorni Linux alat otvorenog koda bez daemona koji olakšava pronalaženje, pokretanje, izgradnju, dijeljenje i implementaciju aplikacija pomoću kontejnera i slika kontejnera Open Containers Initiative (OCI). Podman pruža sučelje naredbenog retka (CLI) poznato svakome tko je koristio Docker Container Engine. Većina korisnika može jednostavno promijeniti Docker u Podman (alias docker=podman) bez ikakvih problema. Slično ostalim uobičajenim Container Engine-ovima (Docker, CRI-O, containerd), Podman se oslanja na OCI kompatibilan Container Runtime (runc, crun, runv, itd.) za povezivanje s operativnim sustavom i stvaranje kontejnera koji rade. Zbog toga se tekući spremnici koje je izradio Podman gotovo ne razlikuju od onih koje stvara bilo koji drugi alat za stvaranje kontejnera. [7]

Slika 6. Podman logo ([7])

Kontejnere pod kontrolom Podmana može pokrenuti root ili neprivilegirani korisnik. Podman upravlja cijelim ekosustavom kontejnera koji uključuje podove, kontejnere, slike kontejnera i volumene kontejnera pomoću biblioteke libpod. Podman je specijaliziran za sve naredbe i funkcije koje vam pomažu održavati i mijenjati slike OCI kontejnera, kao što su povlačenje(eng. pull) i označavanje(eng. label). Omogućuje stvaranje, pokretanje i održavanje tih kontejnera i slika kontejnera u proizvodnom okruženju. Postoji RESTFul API za upravljanje kontejnerima. Postoji udaljeni Podman klijent koji može komunicirati s RESTFul uslugom. Trenutno postoji podrška za Linux, MacOS i Windows. Usluga RESTFul podržana je samo na Linuxu. [7]

3.1.4. Kubernetes

Kubernetes (ponekad se naziva i K8s) popularna je platforma otvorenog koda koja upravlja sustavima za izvršavanje kontejnera preko klastera umreženih resursa. Kubernetes se može koristiti s ili bez Dockera. Platformu je izvorno razvio Google, koji je trebao novi način pokretanja milijardi kontejnera tjedno u velikim razmjerima. Google je 2014. godine Kubernetes objavio kao otvoreni kod i sada se naširoko smatra vodećim na tržištu i standardnim alatom za orkestraciju kontejnera i raspodjelu aplikacija. [8][9]

Slika 7. Kubernetes logo ([8])

Google napominje da je Kubernetes-ov glavni cilj dizajna olakšati implementaciju i upravljanje složenim distribuiranim sustavima, dok još uvijek ima koristi od poboljšane upotrebe koju omogućuju kontejneri. Kubernetes povezuje skup kontejnera u grupu kojom upravlja na istom stroju kako bi smanjio opterećenje mreže i povećao učinkovitost korištenja resursa. Primjer skupa kontejnera je poslužitelj aplikacija, redis pred memorija i SQL baza podataka. Kubernetes je posebno koristan za DevOps timove budući da nudi otkrivanje usluga, balansiranje opterećenja unutar klastera, automatizirano uvođenje i vraćanje na staro stanje, samo ozdravljenje kontejnera koji ne uspiju i upravljanje konfiguracijom. Osim toga, Kubernetes je ključni alat za izgradnju robusnih DevOps CI/CD cjevovoda. Međutim, Kubernetes nije potpuna platforma kao usluga (PaaS) i postoje mnoga razmatranja koja treba imati na umu prilikom izgradnje i upravljanja Kubernetes klasterima. Složenost koja dolazi s upravljanjem Kubernetesom veliki je čimbenik zašto se mnogi korisnici odlučuju koristiti upravljane usluge Kubernetesa dobavljača oblaka. Sustav je često opisan kao "Linux u oblaku" te je najpopularnija platforma za orkestraciju spremnika s razlogom. [8][9] Evo nekih od razloga zašto:

- automatizirane operacije - dolazi s moćnim API-jem i alatom za naredbeni redak, nazvanim kubectl, koji obrađuje većinu posla koji ide u upravljanje kontejnerom dopuštajući automatiziranje operacija. Uzorak kontrolera u Kubernetesu osigurava da kontejneri i aplikacije unutar njih rade točno kako je navedeno. [8]
- apstrakcija infrastrukture - upravlja resursima koji su dostupni. Ovaj pristup oslobađa razvojne programere da se usredotoče na pisanje koda aplikacije, a ne na temeljnu računalnu ili mrežnu infrastrukturu. [8]

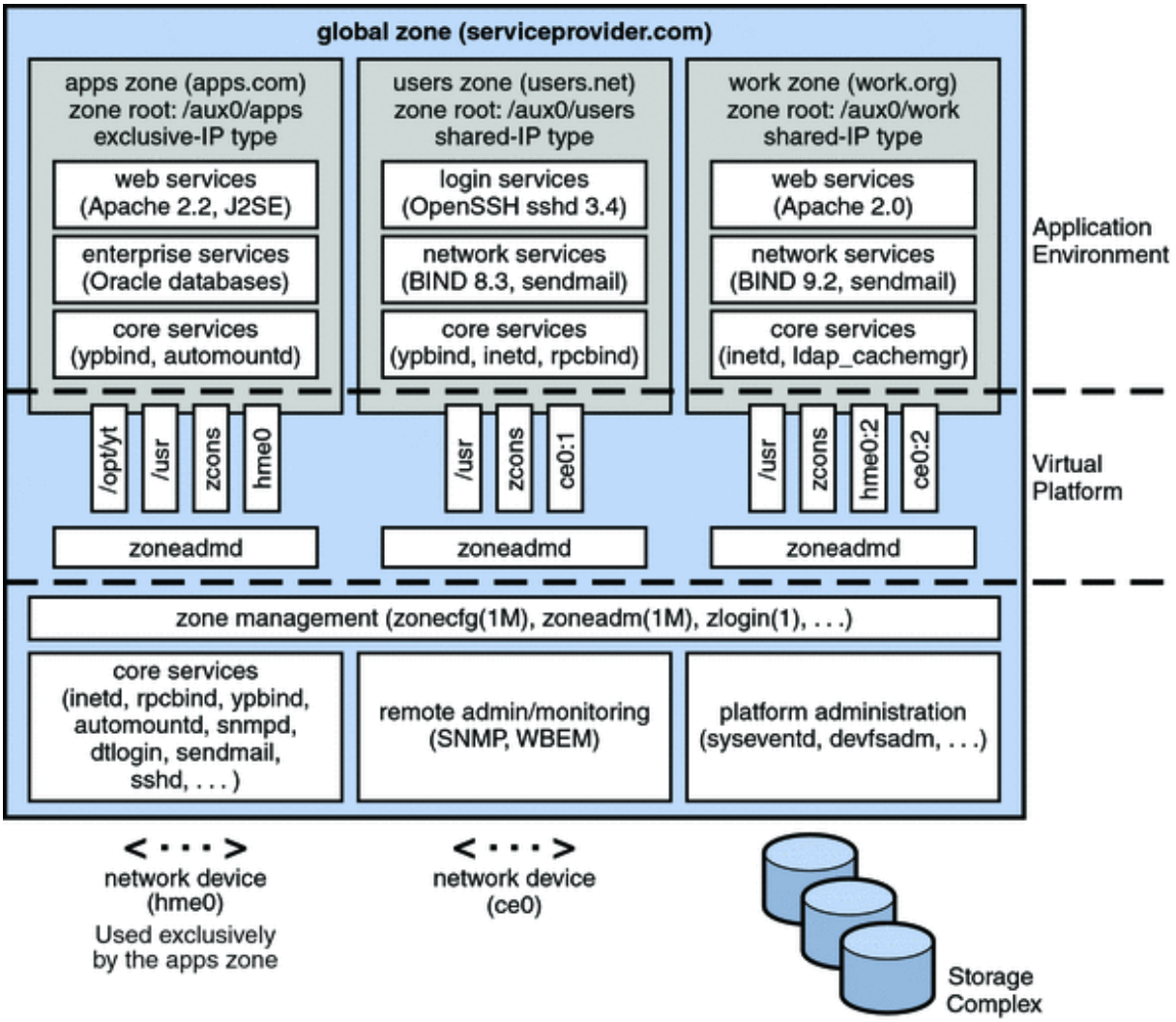
- praćenje ispravnosti usluge - prati radnu okolinu i uspoređuje ju sa željenim stanjem. Izvodi automatske provjere ispravnosti usluga i ponovno pokreće kontejnere koji nisu uspjeli ili su zaustavljeni. Kubernetes čini usluge dostupnima samo kada su pokrenute i spremne. [8]

U ovom dijelu opisana je platforma Kubernetes za virtualizaciju na razini operacijskog sustava, koji se smatra jednim od najpopularnijih alata za upravljanje kontejnerima na visokoj razini kada ih je puno potrebno.

3.2. Virtualne zone

Zona je virtualizirano okruženje operacijskog sustava stvoreno unutar jedne instance operacijskog sustava Solaris. Tehnologija particioniranja Solaris Zones koristi se za virtualizaciju usluga operativnog sustava i pružanje izoliranog i sigurnog okruženja za pokretanje aplikacija. Kada kreirate zonu, proizvodite okruženje za izvršavanje aplikacije u kojem su procesi izolirani od ostatka sustava. Ova izolacija sprječava procese koji se izvode u jednoj zoni da nadziru ili utječu na procese koji se izvode u drugim zonama. Čak ni proces koji se izvodi s vjerodajnicama super korisnika ne može vidjeti niti utjecati na aktivnosti u drugim zonama. Zona također pruža apstraktni sloj koji odvaja aplikacije od fizičkih atributa stroja na kojem su raspoređene. Primjeri ovih atributa uključuju staze fizičkih uređaja. Zone se mogu koristiti na bilo kojem stroju koji pokreće barem izdanje Solarisa 10. Gornja granica za broj zona na sustavu je 8192. Broj zona koje se mogu učinkovito smjestiti na jedan sustav određen je ukupnim zahtjevima za resursima aplikacijskog softvera koji radi u svim zonama. U izdanju Solarisa 10 postoje dvije vrste modela korijenskih datotečnih sustava ne globalnih zona: rijetki(eng. sparse) i cijeli(eng. whole) korijenski. Model rijetke(eng. sparse) korijenske zone optimizira dijeljenje objekata. Cijeli(eng. whole) model korijenske zone pruža maksimalnu konfigurabilnost. Zone su idealne za okruženja koja konsolidiraju brojne aplikacije na jednom

poslužitelju. Trošak i složenost upravljanja brojnim strojevima čine konsolidaciju nekoliko aplikacija na većim, skalabilnijim poslužiteljima korisnim. [10]



Slika 8. Sustav sa 4. zone ([10])

Slika 8. prikazuje sustav s četiri zone. Svaka zona sastavljena od aplikacije, korisnika i posla izvodi radno opterećenje koje nije povezano s radnim opterećenjima drugih zona, u oglednom konsolidiranom okruženju. Ovaj primjer ilustrira da se različite verzije iste aplikacije mogu izvoditi bez negativnih posljedica u različitim zonama, kako bi se udovoljilo zahtjevima konsolidacije. Svaka zona može pružiti prilagođeni skup usluga. Zone omogućuju učinkovitije korištenje resursa na vašem sustavu. Dinamička preraspodjela resursa dopušta premještanje neiskorištenih resursa u druge spremnike prema potrebi. Izolacija kvarova i sigurnosti znači da aplikacije koje se loše ponašaju ne zahtijevaju namjenski i nedovoljno iskorišten sustav. Korištenjem zona ove se aplikacije mogu objediniti s drugim aplikacijama. [10]

Zone omogućuju delegiranje nekih administrativnih funkcija uz održavanje ukupne sigurnosti sustava. Postoje dvije vrste zona, globalne(eng. global) i ne globalne(eng. non-global). U nastavku bit će nabrojane karakteristike svake vrste navedenih zona. Karakteristike globalnih zona su sljedeće: [10]

- Sustav mu dodjeljuje ID 0 [10]
- Pruža jednu instancu jezgre Solarisa koja se može pokrenuti i izvoditi na sustavu [10]
- Sadrži kompletnu instalaciju softverskih paketa sustava Solaris [10]
- Može sadržavati dodatne softverske pakete ili dodatni softver, direktorije, datoteke i druge podatke koji nisu instalirani putem paketa [10]
- Pruža potpunu i dosljednu bazu podataka proizvoda koja sadrži informacije o svim softverskim komponentama instaliranim u globalnoj zoni [10]
- Sadrži informacije o konfiguraciji specifične samo za globalnu zonu, kao što je naziv hosta globalne zone i tablica sustava datoteka [10]
- Jedina je zona koja je svjesna svih uređaja i svih sustava datoteka [10]
- Jedina je zona sa znanjem o postojanju i konfiguraciji ne globalne zone [10]
- Jedina je zona iz koje se ne globalna zona može konfigurirati, instalirati, upravljati ili deinstalirati [10]

S druge strane karakteristike ne globalnih(eng. non-global) su sljedeće:

- Sustav mu dodjeljuje ID zone kada se zona pokrene [10]
- Dijeli operacije pod Solaris jezgrom pokrenutom iz globalne zone [10]
- Sadrži instalirani podskup kompletnih softverskih paketa operacijskog sustava Solaris [10]
- Sadrži softverske pakete Solaris koji se dijele iz globalne zone [10]
- Može sadržavati dodatne instalirane softverske pakete koji se ne dijele iz globalne zone [10]
- Može sadržavati dodatni softver, direktorije, datoteke i druge podatke stvorene u ne globalnoj zoni koji nisu instalirani putem paketa niti se dijele iz globalne zone [10]
- Ima potpunu i dosljednu bazu podataka proizvoda koja sadrži informacije o svim softverskim komponentama instaliranim u zoni, bilo da su prisutne u ne globalnoj zoni ili dijeljene samo za čitanje iz globalne zone [10]
- Nije svjesna postojanja drugih zona [10]

- Ne može instalirati, upravljati ili deinstalirati druge zone, uključujući sebe [10]
- Sadrži informacije o konfiguraciji specifične samo za tu ne globalnu zonu, kao što je naziv hosta ne globalne zone i tablica sustava datoteka [10]
- Može imati vlastitu postavku vremenske zone [10]

Administrator ima privilegije super korisnika ili ulogu primarnog administratora. Kada je prijavljen u globalnu zonu, globalni administrator može nadzirati i kontrolirati sustav u cjelini. Ne globalnom zonom može upravljati administrator zone. Globalni administrator dodjeljuje Zone Management profil administratoru zone. Privilegije administratora zone ograničene su na ne globalnu zonu. [10]

3.3. Virtualni privatni poslužitelji

Virtualni privatni poslužitelj (eng. Virtual Private Servers, VPS), koji se naziva i virtualni namjenski poslužitelj (eng. Virtual Dedicated Server, VDS), virtualni je poslužitelj koji se korisniku čini kao namjenski poslužitelj, ali je zapravo instaliran na računalu koje poslužuje više web stranica. Jedno računalo može imati nekoliko VPS-ova, svaki sa svojim operativnim sustavom (OS) koji pokreće softver za hosting za određenog korisnika. Pružatelj usluga VPS hostinga oslanja se na softver za virtualizaciju, koji se naziva hipervizor, kako bi apstrahirao resurse na fizičkom poslužitelju i korisnicima omogućio pristup emuliranom poslužitelju, koji se naziva virtualni stroj (VM). Svaki virtualni stroj pokreće kompletan operacijski sustav i ima ograničen pristup dijelu računala, memorije i resursa za pohranu fizičkog poslužitelja. Kupci imaju pristup operacijski sustavu VM-a, ali ne i fizičkom poslužitelju. Iako više zakupaca može dijeliti VM-ove koji se nalaze na istom fizičkom poslužitelju, tim VM-ovima je ograničena interakcija s VM-ovima u vlasništvu drugih zakupaca, čime se stvara poslužitelj koji je logički privatan, ali nije fizički odvojen. Jedan alat koji omogućuje stvaranje virtualnih privatnih poslužitelja je OpenVZ. [11]

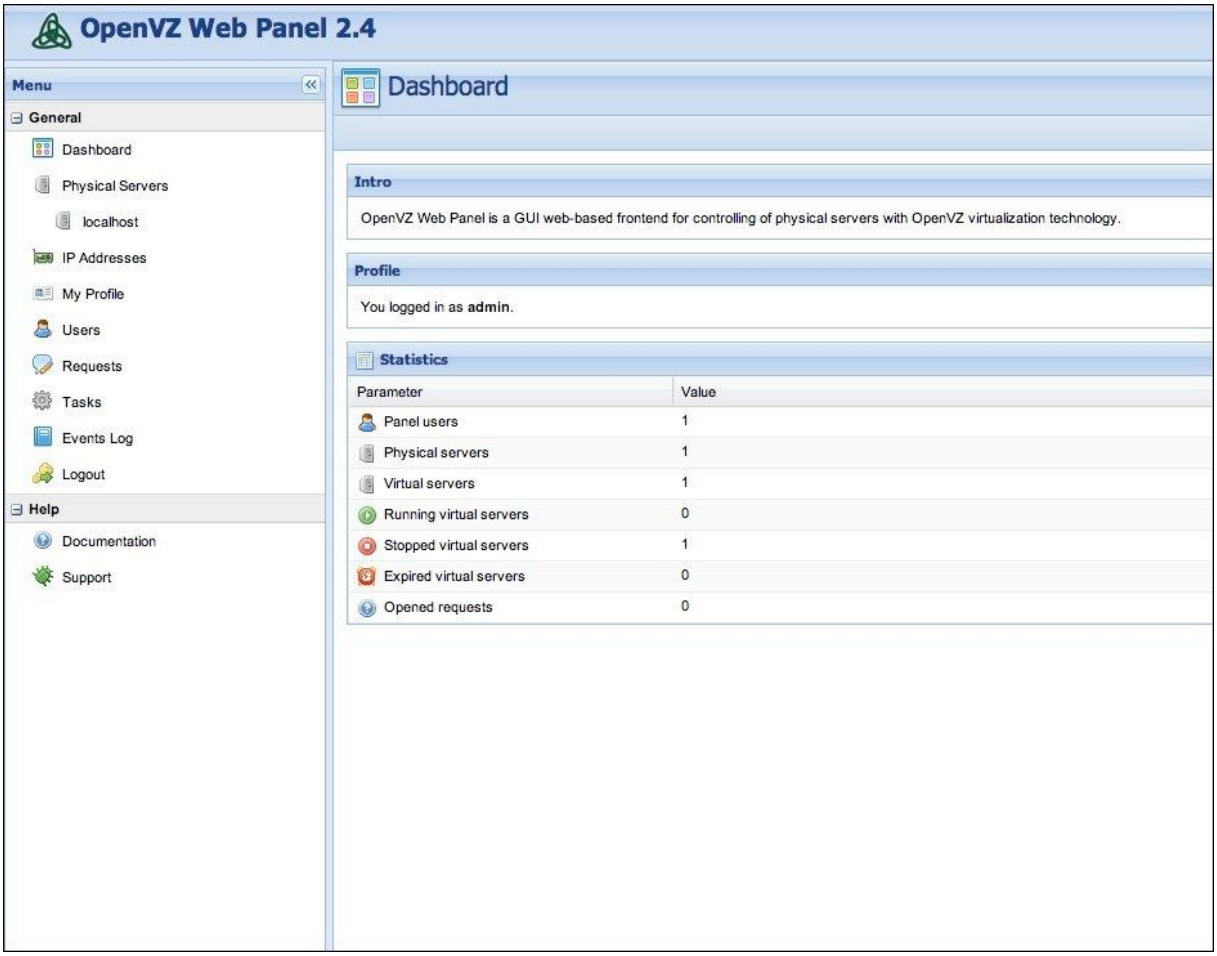
3.3.1. OpenVZ

OpenVZ (Open Virtuozzo) (Slika 9.) je virtualizacijska tehnologija na razini operativnog sustava za Linux. Omogućuje fizičkom poslužitelju pokretanje više izoliranih instanci operativnog sustava, zvanih spremnici, virtualni privatni poslužitelji (VPS) ili virtualna okruženja (VE). OpenVZ je sličan Solaris Containers i LXC. [12]

Slika 9. OpenVZ logo ([12])

Dok tehnologije virtualizacije kao što su VMware, Xen i KVM pružaju potpunu virtualizaciju i mogu pokrenuti više operativnih sustava i različite verzije kernela, OpenVZ koristi jednu Linux kernel i stoga može pokrenuti samo Linux. Svi OpenVZ spremnici dijele istu arhitekturu i verziju kernela. To može biti nedostatak u situacijama u kojima gosti zahtijevaju različite verzije kernela od one glavnog računala. Međutim, budući da nema dodatne troškove kao pravi hipervizor, vrlo je brz i učinkovit. [12]

Dodjela memorije s OpenVZ-om je meka utoliko što memoriju koja se ne koristi u jednom virtualnom okruženju mogu koristiti drugi ili za predmemoriju diska. Dok su stare verzije OpenVZ-a koristile zajednički datotečni sustav (gdje je svako virtualno okruženje samo direktorij datoteka koje su izolirane pomoću chroota), trenutne verzije OpenVZ-a dopuštaju svakom spremniku da ima vlastiti datotečni sustav. OpenVZ kernel je Linux kernel, modificiran za dodavanje podrške za OpenVZ kontejnere. Modificirana jezgra omogućuje virtualizaciju, izolaciju, upravljanje resursima i kontrolne točke. Svaki kontejner je zasebna cjelina i ponaša se uglavnom kao fizički poslužitelj, odnosno server. Prema zadanim postavkama, OpenVZ ograničava pristup kontejneru na stvarne fizičke uređaje (čime kontejner postaje neovisan o hardveru). Administrator OpenVZ-a može omogućiti pristup kontejneru različitim stvarnim uređajima, kao što su diskovni pogoni, USB priključci, PCI uređaji ili fizičke mrežne kartice. Grafičko korisničko sučelje nazvano EasyVZ pokušano je izraditi 2007., ali nije napredovalo dalje od verzije 0.1. Do verzije 3.4, Proxmox VE se mogao koristiti kao virtualizacijsko okruženje poslužitelja temeljeno na OpenVZ-u s GUI-jem (Slika 10.), iako su kasnije verzije prešle na LXC. [12]



Slika 10. Grafičko sučelje OpenVZ-a (Vlastita izrada, 2022.)

3.4. Virtualna jezgre

Arhitektura virtualne jezgre (vkernel) je paradigma virtualizacije operacijskog sustava gdje se kod jezgre može kompilirati za izvođenje u korisničkom prostoru, na primjer, kako bi se olakšalo otklanjanje pogrešaka različitih komponenti na razini jezgre, uz virtualizaciju opće namjene i kompartmentalizaciju resursa sustava. Koristi ga DragonFly BSD(Slika 11.) u svojoj implementaciji vkernela od DragonFly 1.7, prvi put je otkriven u rujnu 2006. (prije 15 godina) i prvi put objavljen u stabilnoj grani s DragonFly 1.8 u siječnju 2007. (prije 15 godina). Dugoročni cilj, uz olakšavanje razvoja kernela, je olakšati podršku klasterima računala povezanih s internetom bez ugrožavanja lokalne sigurnosti. Slični koncepti postoje i u drugim operativnim sustavima, u Linuxu je sličan koncept virtualizacije poznat kao Linux u korisničkom načinu rada, dok je u NetBSD-u od ljeta 2007. to bio početni fokus rump kernel infrastrukture. [13.]

Slika 11. Logo DragonFly BSD alata ([13])

Koncept virtualne jezgre gotovo je potpuna suprotnost konceptu unikernela — s vkernelom, komponente jezgre mogu se izvoditi u korisničkom prostoru kako bi se olakšao razvoj jezgre i otklanjanje pogrešaka, uz podršku regularnog kernela operativnog sustava; dok s unikernelom, komponente na razini korisničkog prostora mogu se izvoditi izravno u prostoru jezgre za dodatnu izvedbu, podržanu bare metalnim hardverom ili sklopom hardverske virtualizacije. Međutim, i vkerneli i unikerneli mogu se također koristiti za slične zadatke, na primjer, za samostalni softver u virtualiziranom okruženju s niskim opterećenjem. Zapravo, krnja jezgra NetBSD-a, koja je izvorno bila fokusirana na pokretanje komponenti jezgre u korisničkom prostoru, od tada se također pomaknula u prostor unikernela (idući nakon nadimka bilo koje jezgre jer podržava obje paradigme). [13.]

3.5. FreeBSD jail

Jail mehanizam je implementacija FreeBSD-ove(Slika 12.) virtualizacije na razini operacijskog sustava koja omogućava administratorima sustava da particioniraju računalni sustav izveden iz FreeBSD-a u nekoliko nezavisnih mini-sustava koji se nazivaju zatvorima(eng. jail), a svi dijele istu jezgru, uz vrlo malo troškova. Implementiran je kroz sistemski poziv, jail, kao i korisnički pomoćni program. Funkcionalnost je 1999. godine u FreeBSD unio Poul-Henning Kamp nakon određenog razdoblja proizvodne upotrebe od strane

davatelja usluga hostinga, a prvi put je objavljena s FreeBSD 4.0, stoga je podržana na brojnim FreeBSD potomcima, uključujući DragonFly BSD, do danas. [14]

Slika 12. FreeBSD logo ([14])

Potreba za jail FreeBSD-ovima proizašla je iz želje malog pružatelja usluga hostinga dijeljenog okruženja (vlasnik R&D Associates, Inc., Derrick T. Woolworth) da uspostave čistu, jasnu razliku između vlastitih usluga i usluga svojih kupaca, uglavnom zbog sigurnosti i lakše administracije. Umjesto dodavanja novog sloja detaljnih konfiguracijskih opcija, rješenje koje je usvojio Poul-Henning Kamp bilo je segmentiranje sustava – i njegovih datoteka i resursa – na takav način da samo pravim osobama bude omogućen pristup pravim odjeljcima. [14.]

FreeBSD jail imaju tri cilja:

- virtualizacija - svaki jail je virtualno okruženje koje radi na glavnom računalu sa svojim datotekama, procesima, korisničkim i super korisničkim računima. Unutar zatvorenog procesa, okruženje se gotovo ne razlikuje od stvarnog sustava. [14]
- Sigurnost - svaki jail je zapečaćen od ostalih, čime se pruža dodatna razina sigurnosti. [14]
- Jednostavnost delegiranja - ograničeni opseg jail-a omogućuje administratorima sustava delegiranje nekoliko zadataka koji zahtijevaju pristup super korisnika bez prenošenja potpune kontrole nad sustavom. [14]

S jail-om je moguće kreirati različite virtualne strojeve, od kojih svaki ima vlastiti skup instaliranih uslužnih programa i vlastitu konfiguraciju. To ga čini sigurnim načinom isprobavanja softvera. Na primjer, moguće je pokrenuti različite verzije ili isprobati različite konfiguracije paketa web poslužitelja u različitim jail-ovima. A budući da je jail ograničen na

uski opseg, učinci pogrešne konfiguracije ili pogreške (čak i ako ih je napravio superkorisnik u zatvoru) ne ugrožavaju ostatak integriteta sustava. Budući da zapravo ništa nije modificirano izvan zatvora, "promjene" se mogu odbaciti brisanjem kopije stabla direktorija jail-a. Virtualizacija je vrijedna za pružatelje usluga koji žele svojim korisnicima ponuditi mogućnost prilagođenih konfiguracija, a da cijeli sustav bude jednostavan za održavanje. Na primjer, dva različita korisnika mogu trebati različite verzije istog softvera. Bez jail-a, konfiguriranje više verzija softvera u različitim direktorijima i osiguravanje da ne zadiru jedna u drugu nije uvijek moguće ili lako za održavanje. Jails s druge strane dopuštaju softverskim paketima da gledaju na sustav zasebno, kao da svaki paket ima stroj za sebe. Jail-ovi također mogu imati svoje, neovisne, zatvorene superkorisnike. FreeBSD zatvor ipak ne postiže pravu virtualizaciju, ne dopušta virtualnim strojevima pokretanje različitih verzija kernela od onog osnovnog sustava. Svi virtualni poslužitelji dijele isti kernel i stoga otkrivaju iste greške i potencijalne sigurnosne rupe. Ne postoji podrška za klasteriranje ili migraciju procesa, tako da su kernel glavnog računala i glavno računalo još uvijek jedinstvena točka kvara za sve virtualne poslužitelje. Moguće je koristiti jail-ove za sigurno testiranje novog softvera, ali ne i novih kernela. FreeBSD jails su učinkovit način za povećanje sigurnosti poslužitelja zbog odvojenosti između zatvorene okoline i ostatka sustava (drugih jail-ova i osnovnog sustava). Na primjer, u nezatvorenom sustavu, web poslužitelj koji radi kao korisnik www koji uvodi PHP-include ranjivost ugrozio bi cijeli sustav. Napadač bi imao prava korisnika www koji obično može mijenjati datoteke na web poslužitelju, prolaziti kroz stablo direktorija i prikupljajte informacije, kao što je potpuni popis korisnika, ljuska i početni direktorij iz /etc/passwd. Ali ako je web poslužitelj zatvoren, opseg korisnika www je ograničen na taj jail, koji zauzvrat može biti dovoljno minimalističan da ne oda previše. Čak i ako je napadač dobio pristup super korisničkom računu tog jail-a, mogao je modificirati samo taj jail, a ne cijeli sustav. [14]

4. Primjer virtualizacije na razini operacijskog sustava

Sada kada smo prošli kroz koncepte virtualizacije na temelju operacijskog sustava i opisali neke od alata u prijašnjem poglavlju u ovome poglavlju ćemo prikazati praktični primjer gdje će biti prikazana instalacija Docker alata na Linux Mint sustav. Nakon toga bit će prikazana kreacija i pokretanje kontejnera za dvije aplikacije, jedna PHP, a druga Python koje međusobno komuniciraju, gdje Python aplikacija služi kao API iz kojeg PHP aplikacija dohvaća podatke za prikaz na stranici. Za kraj bit će prikazano još upravljanje kontejnerima pomoću Kubernetesa, odnosno kreirat ćemo klaster(eng. cluster) kontejnera te prikazati upravljanje njima.

4.1. Instalacija Docker alata

Prvo kako bi se instalirao Docker alat potrebno je ažurirati sistemske pakete Linux operacijskog sustava, što se radi pomoću pokretanja sljedeće linije u naredbenom retku operacijskog sustava: [15]

```
sudo apt update
```

Nakon ažuriranja sistemskih paketa instalirat ćemo Docker alat pomoću Debian paketa. Kako bi to učinili potrebno je prvo preuzeti Debian pakete za containered.io.deb, Docker-ce-cli.deb, i docker-ce.deb pakete. Za to su korištene sljedeće naredbe. Za skidanje containered.io.deb paketa potrebno je pokrenuti sljedeću naredba: [15]

```
wget
https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/cont
ainerd.io_1.6.4-1_amd64.deb
```

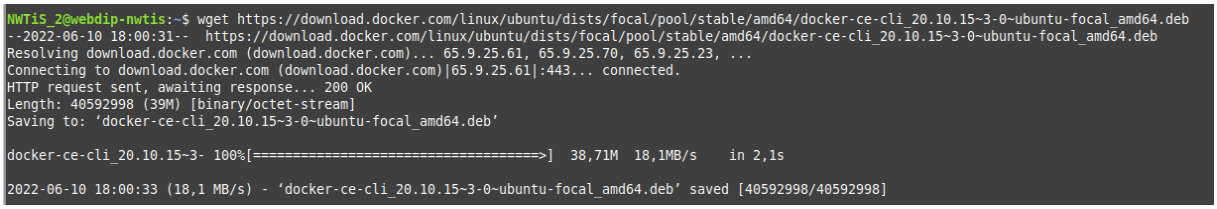
```
NWTiS_2@webdip-nwtis:~$  wget  https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/containerd.io_1.6.4-1_amd64.deb
--2022-06-10 18:00:10--  https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/containerd.io_1.6.4-1_amd64.deb
Resolving download.docker.com (download.docker.com)... 65.9.25.61, 65.9.25.70, 65.9.25.23, ...
Connecting to download.docker.com (download.docker.com)|65.9.25.61|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 28122634 (27M) [binary/octet-stream]
Saving to: ‘containerd.io_1.6.4-1_amd64.deb’

containerd.io_1.6.4-1_amd 100%[===================================>]  26,82M  26,7MB/s    in 1,0s

2022-06-10 18:00:12 (26,7 MB/s) - ‘containerd.io_1.6.4-1_amd64.deb’ saved [28122634/28122634]
```

Slika 13. Rezultat pokretanja komande (Vlastita izrada, 2022.)

Na slici 13. prikazan je rezultat pokretanja navedene komande.

U nastavku je potrebno pokrenuti komandu za preuzimanje Docker-ce-cli.deb paketa, za to nam je potrebna sljedeća naredba: [15]

```
wget
https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/docker-ce-cli_20.10.15~3-0~ubuntu-focal_amd64.deb
```

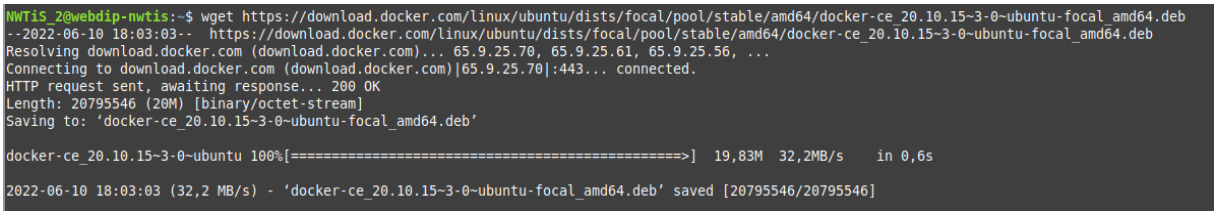


Slika 14. Rezultat pokretanja komande (Vlastita izrada, 2022.)

Na slici 14. prikazan je rezultat pokretanja navedene komande.

Prije same instalacije Docker alata potrebno je još pokrenuti komandu kako bi se skinuo docker-ce.deb paket, za to je potrebna sljedeća naredba: [15]

```
wget
https://download.docker.com/linux/ubuntu/dists/focal/pool/stable/amd64/docker-ce_20.10.15~3-0~ubuntu-focal_amd64.deb
```



Slika 15. Rezultat pokretanja komande (Vlastita izrada, 2022.)

Kada smo izvršili sve navedene naredbe možemo instalirati Docker alat na naš Linux operacijski sustav. To se radi pomoću sljedeće naredbe: [15]

```
sudo apt install ./*.deb
```

Na slici 16. prikazan je rezultat pokretanja komande za instalaciju Docker alata na naš željeni operacijski sustav. Kada smo izvršili sve te korake pomoću sljedeće naredbe: [15]

```
sudo docker version
```

možemo verificirati da se instalacija uspješno izvršila. Rezultat te komande prikazan je na slici 17. Sada nakon što smo pripremili Docker alat za korištenje možemo krenuti na kreiranje i pokretanje Docker alata. U nastavku će prvo biti prikazan izvorni kod aplikacija a potom svi potrebno koraci kako bi se kreirali i pokretali kontejneri pomoću Docker alata. [15]

```
NWTiS_2@webdip-nwtis:~$ sudo apt install ./*.deb
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'containerd.io' instead of './containerd.io_1.6.4-1_amd64.deb'
Note, selecting 'docker-ce' instead of './docker-ce_20.10.15~3-0~ubuntu-focal_amd64.deb'
Note, selecting 'docker-ce-cli' instead of './docker-ce-cli_20.10.15~3-0~ubuntu-focal_amd64.deb'
The following additional packages will be installed:
  docker-scan-plugin libseccomp2
Suggested packages:
  aufs-tools cgroupfs-mount | cgroup-lite
The following NEW packages will be installed:
  docker-scan-plugin
The following packages will be upgraded:
  containerd.io docker-ce docker-ce-cli libseccomp2
4 upgraded, 1 newly installed, 0 to remove and 529 not upgraded.
Need to get 93,1 MB of archives.
After this operation, 24,3 MB disk space will be freed.
Do you want to continue? [Y/n] y
Get:1 https://download.docker.com/linux/ubuntu focal/stable amd64 containerd.io amd64 1.6.4-1 [28,1 MB]
Get:2 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libseccomp2 amd64 2.5.1-1ubuntu1~20.04.2 [42,5 kB]
Get:3 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-ce-cli amd64 5:20.10.15~3-0~ubuntu-focal [40,6 MB]
Get:4 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-ce amd64 5:20.10.15~3-0~ubuntu-focal [20,8 MB]
Get:5 https://download.docker.com/linux/ubuntu focal/stable amd64 docker-scan-plugin amd64 0.17.0~ubuntu-focal [3.521 kB]
Fetched 93,1 MB in 4s (23,1 MB/s)
(Reading database ... 371240 files and directories currently installed.)
Preparing to unpack .../libseccomp2_2.5.1-1ubuntu1~20.04.2_amd64.deb ...
Unpacking libseccomp2:amd64 (2.5.1-1ubuntu1~20.04.2) over (2.4.3-1ubuntu3.20.04.3) ...
Setting up libseccomp2:amd64 (2.5.1-1ubuntu1~20.04.2) ...
(Reading database ... 371240 files and directories currently installed.)
Preparing to unpack .../containerd.io_1.6.4-1_amd64.deb ...
Unpacking containerd.io (1.6.4-1) over (1.4.3-1) ...
Preparing to unpack .../docker-ce-cli_5%3a20.10.15~3-0~ubuntu-focal_amd64.deb ...
Unpacking docker-ce-cli (5:20.10.15~3-0~ubuntu-focal) over (5:20.10.3~3-0~ubuntu-focal) ...
Preparing to unpack .../docker-ce_5%3a20.10.15~3-0~ubuntu-focal_amd64.deb ...
Unpacking docker-ce (5:20.10.15~3-0~ubuntu-focal) over (5:20.10.3~3-0~ubuntu-focal) ...
Selecting previously unselected package docker-scan-plugin.
Preparing to unpack .../docker-scan-plugin_0.17.0~ubuntu-focal_amd64.deb ...
Unpacking docker-scan-plugin (0.17.0~ubuntu-focal) ...
Setting up docker-scan-plugin (0.17.0~ubuntu-focal) ...
Setting up containerd.io (1.6.4-1) ...
Installing new version of config file /etc/containerd/config.toml ...
Setting up docker-ce-cli (5:20.10.15~3-0~ubuntu-focal) ...
Setting up docker-ce (5:20.10.15~3-0~ubuntu-focal) ...
Processing triggers for systemd (245.4-4ubuntu3.4) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.2) ...
```

Slika 16. Rezultat naredbe za instalaciju Docker alata(Vlastita izrada, 2022.)

```
NWTiS_2@webdip-nwtis:~$ sudo docker version
Client: Docker Engine - Community
 Version:           20.10.15
 API version:       1.41
 Go version:        go1.17.9
 Git commit:        fd82621
 Built:             Thu May  5 13:19:23 2022
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.15
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.17.9
  Git commit:       4433bf6
  Built:            Thu May  5 13:17:28 2022
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.4
  GitCommit:        212e8b6fa2f44b9c21b2798135fc6fb7c53efc16
 runc:
  Version:          1.1.1
  GitCommit:        v1.1.1-0-g52de29d
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

Slika 17. Rezultat naredbe za verifikaciju instalacije(Vlastita izrada, 2022.)

4.2. Izvorni kod korištene aplikacije

Izvori kod PHP aplikacije:

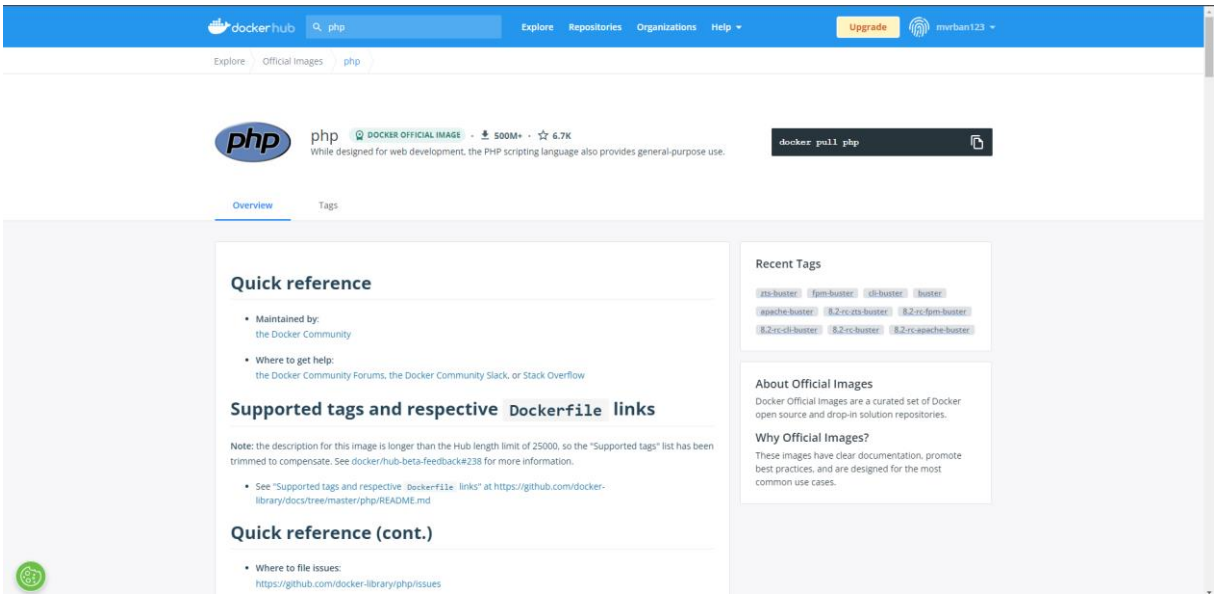
```
<html>
    <head>
        <title>My Shop</title>
    </head>

    <body>
        <h1>Welcome to my shop</h1>
        <ul>
            <?php

            $json = file_get_contents('http://product-service/');
            $obj = json_decode($json);

            $products = $obj->products;

            foreach ($products as $product) {
                echo "<li>$product</li>";
            }

            ?>
        </ul>
    </body>
</html>
```

Izvorni kod Python aplikacije:

```
from flask import Flask
from flask_restful import Resource, Api

# Instantiate the app
app = Flask(__name__)
api = Api(app)

class Product(Resource):
    def get(self):
        return {
            'products': ['Ice cream', 'Chocolate', 'Fruit', 'Eggs']
        }

# Create routes
api.add_resource(Product, '/')

# Run the application
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80, debug=True)
```

4.3. Kreiranje i pokretanje kontejnera

U ovome dijelu poglavlja kreirat ćemo iz našeg izvornog koda aplikacija dva zasebna kontejnera koji zajedno komuniciraju i čine našu jednostavnu web aplikaciju. Za početak kako bi kreirali kontejnere pomoću Docker alata prije toga još moramo kreirati takozvane „Dockerfile“ datoteke u koje se nalaze definirane sve potrebne ovisnosti koje moraju biti uključene u kontejnere kako bi aplikacija unutar njega ispravno radila. Na temelju „Dockerfile“ datoteke kreirat će se slika Docker kontejnera kojega onda možemo pokrenuti te samim time kreirati našu aplikaciju unutar Docker kontejnera koja je spremna za korištenje. Kako bi kreirali PHP aplikaciju u Docker kontejneru potrebna nam je okolina s instaliranim PHP-om i Apache servisom, što ćemo definirati u „Dockerfile“ datoteci. Kako već postoje gotove slike na Docker hub repozitoriju ne moramo početi od nule nego otići na Docker hub stranicu i pretražiti već postojeću PHP sliku(Slika 18.) i nastaviti graditi na nju. Kada nađemo sliku u sklopu nje već postoji definirano kako bi naš „Dockerfile“ trebao izgledati.



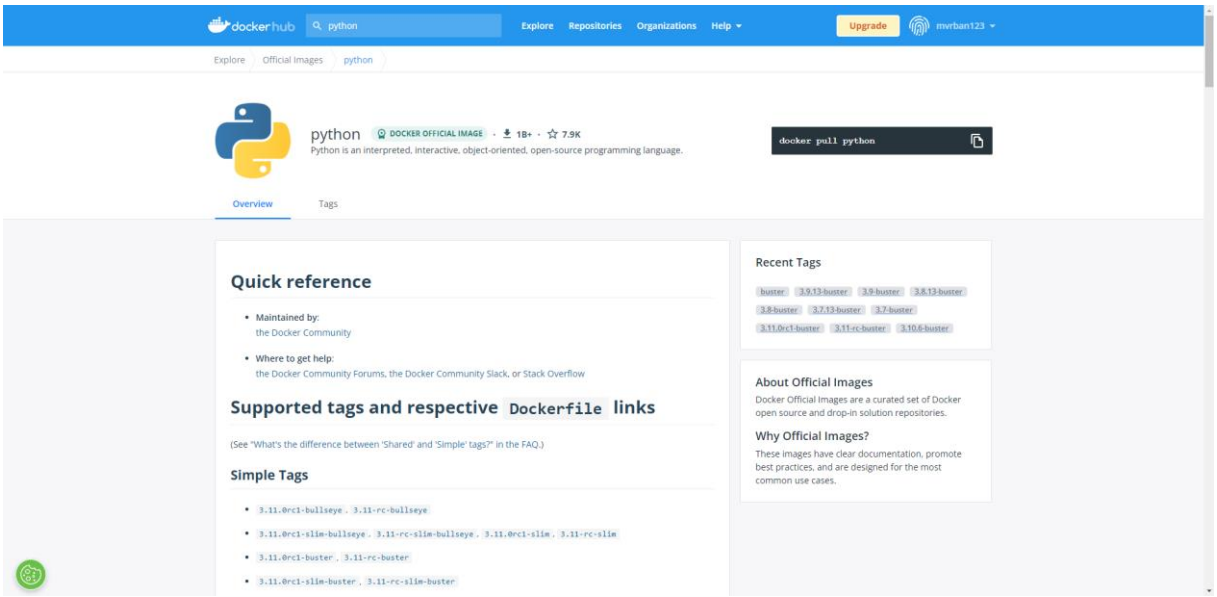
Slika 18. Postojeća Docker hub PHP slika (Vlastita izrada, 2022.)

Na temelju toga „Dockerfile“ datoteka za PHP aplikaciju izgleda ovako:

```
FROM php:7.0-apache
COPY . /var/www/html
EXPOSE 80
```

Prva linija koda predstavlja službenu sliku koju smo pronašli i koju referenciramo zajedno sa kojom verzijom PHP i apache servisa želimo da se instalira. Sljedeća linija označava na koju lokaciju da se trebaju kopirati dijelovi aplikacije unutar slike kako bi se ispravno izvršavali dok zadnja linija označava na kojem portu localhosta želimo da aplikacija bude dostupna. Sukladno

tome potrebno je još na jednaki način kreirati „Dockerfile“ za Python aplikaciju. Postojeću sliku jednako tako možemo pronaći na Docker hub repozitoriju što je prikazano na slici 19.



Slika 19. Postojeća Docker hub Python slika (Vlastita izrada, 2022.)

Sukladno tome „Dockerfile“ datoteka za Python sliku izgledati će sljedeće:

```
FROM python:3-onbuild
COPY . /usr/src/app
CMD ["python", "product.py"]
```

Gdje prva linija predstavlja na koju postojeću sliku gradimo našu te koju verziju Pythona želimo za našu aplikaciju. Sljedeća kamo se trebaju kopirati dijelovi aplikacije unutar kontejner slike kako bi ona uspješno radila. Na kraju je definira se koja komanda će se pokrenuti kako bi naša aplikacija počela s radom. Kada smo kreirali naše „Dockerfile“ datoteke za kreiranja Docker slika koje će na kraju postati kontejneri možemo krenuti na sljedeće poglavlje gdje ćemo napraviti upravo to.

4.3.1. Kreiranje i pokretanje kontejnera pomoću Docker alata

Kada smo pripremili sve što nam je potrebno možemo konačno krenuti s kreiranjem Docker slike koje pokretanjem postaju Docker kontejneri. Prvo ćemo kreirati Docker sliku za Python aplikaciju, pokrenuti tu sliku kako bi dobili kontejner u kojem radi aplikacija te potom ćemo pokušati pristupiti toj aplikaciji. Kako bismo to napravili potrebno je pokrenuti sljedeću naredbu:

```
docker build -t product.py . --file=Dockerfile
```

```
NWTiS_2@webdip-nwtis:~/diplomski/product$ docker build -t product.py . --file=Dockerfile
Sending build context to Docker daemon  5.632kB
Step 1/3 : FROM python:3-onbuild
# Executing 3 build triggers
 ---> Using cache
 ---> Using cache
 ---> Using cache
 ---> 2dd0b97fdb7e
Step 2/3 : COPY . /usr/src/app
 ---> Using cache
 ---> 1c78c29b70f8
Step 3/3 : CMD ["python", "product.py"]
 ---> Using cache
 ---> e44f2bdec7b5
Successfully built e44f2bdec7b5
Successfully tagged product.py:latest
```

Slika 20. Kreiranje Docker slike za Python aplikaciju (Vlastita izrada, 2022.)

Na slici 20. prikazan je rezultat pokretanja naredbe za kreiranje Docker slike te vidimo kako se ona uspješno kreirala. Pokretanjem naredbe:

```
docker image ls
```

Možemo izlistati sve kreirane Docker slike. Što je prikazano na slici 21. te tamo vidimo kako se kreirala naša slika „python.py“, koju možemo onda pokrenuti.

```
NWTiS_2@webdip-nwtis:~/diplomski/product$ docker image ls
REPOSITORY                  TAG          IMAGE ID       CREATED         SIZE
diplomski_website           latest       dfcc88cacc87   6 weeks ago     368MB
diplomski_product-service   latest       6b975932a4aa   6 weeks ago     702MB
mvrban123/diplomskiservice  latest       6b975932a4aa   6 weeks ago     702MB
diplomskiproduct            latest       e44f2bdec7b5   6 weeks ago     702MB
product.py                  latest       e44f2bdec7b5   6 weeks ago     702MB
mvrban123/diplomskiproduct  latest       e44f2bdec7b5   6 weeks ago     702MB
<none>                      <none>       04b1379475fc   2 months ago    702MB
diplomski                   latest       87e5d46b0fb1   2 months ago    368MB
index.php                   latest       87e5d46b0fb1   2 months ago    368MB
mvrban123/diplomski         latest       87e5d46b0fb1   2 months ago    368MB
mvrban123/index.php         latest       87e5d46b0fb1   2 months ago    368MB
<none>                      <none>       eac8d616a3ef   2 months ago    702MB
<none>                      <none>       675f5c158199   2 months ago    702MB
<none>                      <none>       851eab2b4fde   2 months ago    702MB
<none>                      <none>       2e630faae55c   2 months ago    702MB
<none>                      <none>       54bc44e1730a   2 months ago    702MB
<none>                      <none>       bd7a4a27465a   2 months ago    368MB
products.php                latest       e5c3a2a95643   2 months ago    368MB
santosomar/webgoat          latest       cd4f5cf4ef0c   2 years ago     477MB
hello-world                 latest       bf756fb1ae65   2 years ago     13.3kB
php                         7.0-apache   aa67a9c9814f   3 years ago     368MB
python                      3-onbuild    292ed8dee366   4 years ago     691MB
```

Slika 21. Prikaz svih kreiranih Docker slika (Vlastita izrada, 2022.)

Novo kreiranu sliku možemo pokrenuti putem sljedeće naredbe:

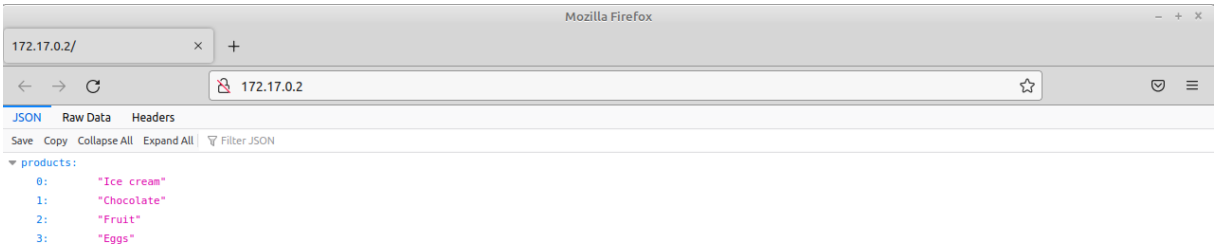
```
docker run -p 81:81 product.py
```

```
NWTiS_2@webdip-nwtis:~/diplomski/product$ docker run -p 81:81 product.py
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:80/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 509-160-763
172.17.0.1 - - [28/Aug/2022 14:16:09] "GET / HTTP/1.1" 200 -
```

Slika 22. Pokretanje Docker kontejnera (Vlastita izrada, 2022.)

Kako bi se pokrenuo Docker kontejner na temelju slike potrebno je specificirati u naredbi koja slika će se pokrenuti te na kojem portu unutar kontejnera će se aplikacija izvoditi. Na slici 22. možemo vidjeti rezultat pokretanja naredbe te na kojoj adresi možemo pristupiti aplikaciji što je na adresi:

http://172.17.0.2:80



Slika 23. Prikaz aplikacije pokrenute unutar kontejnera (Vlastita izrada, 2022.)

Na slici 23. vidimo prikaz aplikacije pokrenuto unutar Docker kontejnera. Kada smo to napravili sada još moramo kreirati Docker sliku i kontejner za nju za našu PHP aplikaciju. Za to moramo ponoviti sve korake ponovno koje smo napravili za Python aplikaciju uz samo manje izmjene. Kako bismo kreirali Docker sliku za PHP aplikaciju trebamo pokrenuti sljedeću naredbu:

```
docker build -t index.php .
```

```
NWTiS_2@webdip-nwtis:~/diplomski/website$ docker build -t index.php .
Sending build context to Docker daemon  3.584kB
Step 1/3 : FROM php:7.0-apache
 ---> aa67a9c9814f
Step 2/3 : COPY . /var/www/html
 ---> Using cache
 ---> 1d1a5c6f123d
Step 3/3 : EXPOSE 80
 ---> Using cache
 ---> 87e5d46b0fb1
Successfully built 87e5d46b0fb1
Successfully tagged index.php:latest
```

Slika 24. Kreiranje Docker slike za PHP aplikaciju (Vlastita izrada, 2022.)

Na slici 24. možemo vidjeti rezultat izvršavanja navedene naredbe i sada ako opet pokrenemo naredbu za izlistavanje svih kreiranih slika možemo vidjeti našu novo kreiranu sliku. To možemo vidjeti na slici 25.

```
NWTiS_2@webdip-nwtis: ~/diplomski/website
File  Edit  View  Search  Terminal  Help
NWTiS_2@webdip-nwtis:~/diplomski/website$ docker image ls
REPOSITORY                   TAG          IMAGE ID       CREATED        SIZE
diplomski_website            latest       dfcc88cacc87   6 weeks ago    368MB
diplomski_product-service    latest       6b975932a4aa   6 weeks ago    702MB
mvrban123/diplomskiservice   latest       6b975932a4aa   6 weeks ago    702MB
diplomskiproduct             latest       e44f2bdec7b5   6 weeks ago    702MB
product.py                   latest       e44f2bdec7b5   6 weeks ago    702MB
mvrban123/diplomskiproduct   latest       e44f2bdec7b5   6 weeks ago    702MB
<none>                       <none>       04b1379475fc   2 months ago   702MB
diplomski                    latest       87e5d46b0fb1   2 months ago   368MB
index.php                    latest       87e5d46b0fb1   2 months ago   368MB
mvrban123/diplomski          latest       87e5d46b0fb1   2 months ago   368MB
mvrban123/index.php          latest       87e5d46b0fb1   2 months ago   368MB
<none>                       <none>       eac8d616a3ef   2 months ago   702MB
<none>                       <none>       675f5c158199   2 months ago   702MB
<none>                       <none>       851eab2b4fde   2 months ago   702MB
<none>                       <none>       2e630faae55c   2 months ago   702MB
<none>                       <none>       54bc44e1730a   2 months ago   702MB
<none>                       <none>       bd7a4a27465a   2 months ago   368MB
products.php                 latest       e5c3a2a95643   2 months ago   368MB
santosomar/webgoat           latest       cd4f5cf4ef0c   2 years ago    477MB
hello-world                  latest       bf756fb1ae65   2 years ago    13.3kB
php                          7.0-apache   aa67a9c9814f   3 years ago    368MB
python                       3-onbuild    292ed8dee366   4 years ago    691MB
```

Slika 25. Prikaz svih kreiranih Docker slika (Vlastita izrada, 2022.)

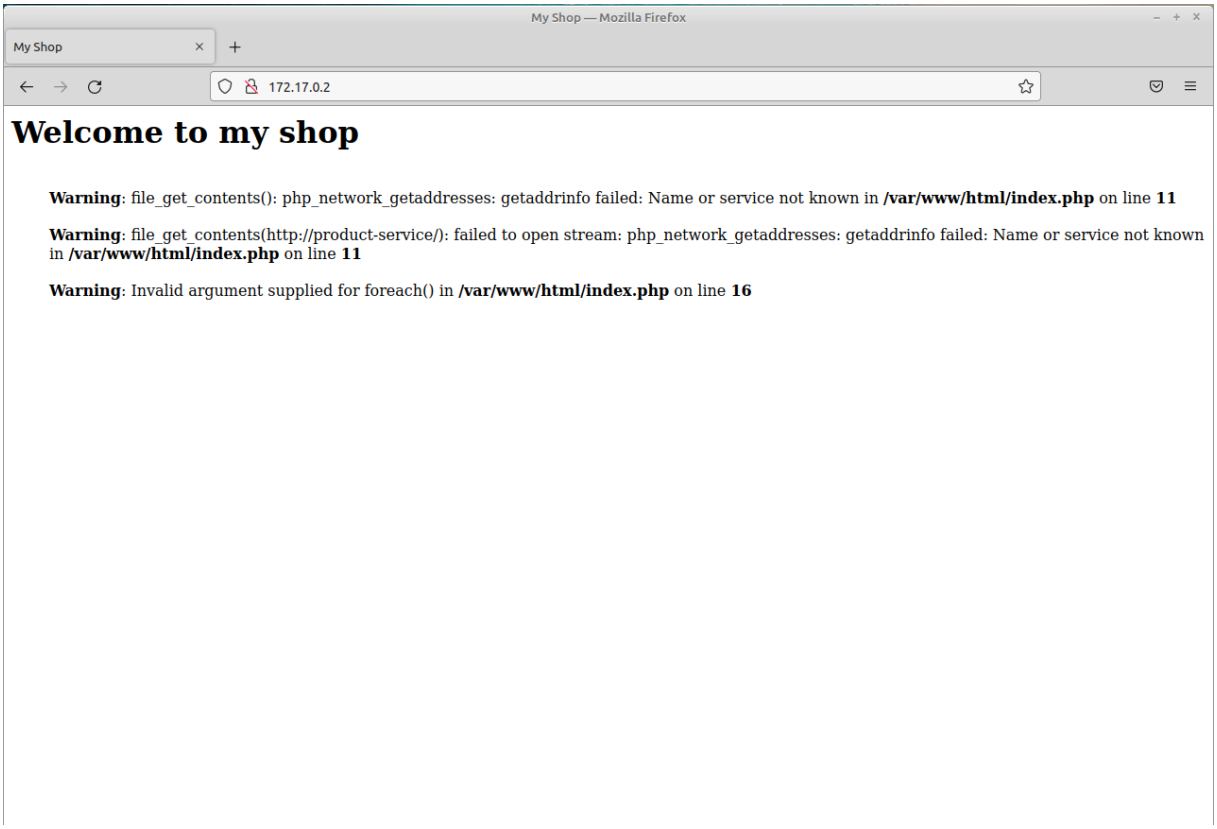
Sada kada imamo kreiranu i sliku PHP aplikacije možemo pokrenuti taj kontejner i vidjeti kako radi. Kontejner PHP aplikacije pokrećemo preko sljedeće naredbe:

```
docker run -p 80:80 index.php
```

```
NWTiS_2@webdip-nwtis: ~/diplomski/website
File  Edit  View  Search  Terminal  Help
NWTiS_2@webdip-nwtis:~/diplomski/website$ docker run -p 80:80 index.php
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive
globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive
globally to suppress this message
[Sun Aug 28 15:48:13.558997 2022] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.25 (Debian) PHP/7.0.33 configured -- resuming normal op
erations
[Sun Aug 28 15:48:13.559143 2022] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
```

Slika 26. Pokretanje Docker kontejnera PHP aplikacije (Vlastita izrada, 2022.)

Na slici 26. možemo vidjeti rezultat pokretanja kontejnera koji sadrži PHP aplikaciju te na kojoj adresi možemo pristupiti aplikaciji u kontejneru. Ako pristupimo aplikaciji na adresi, što vidimo na slici 27. vidimo kako aplikacija zapravo ne radi kako treba. Razlog tome je to što očekuje podatke od `'http://product-service/'` adrese no ona još nije vidljiva našoj aplikaciji.



Slika 27. Prikaz PHP aplikacije pokrenute unutar kontejnera (Vlastita izrada, 2022.)

Kako bi to omogućili sada kad smo vidjeli kako kreirati i pokrenuti Docker kontejnere koristit ćemo Docker alat nazvan docker-compose. On nam omogućava pokretanje više kontejnera od jednom i ujedno razmjenu podataka između kontejnera na temelju serverskih imena. Prvo kako bismo zaustavili rad trenutnih kontejnera samo treba unutar naredbenog retka pokrenuti naredbu za prekid što je u Linux operacijskom sustavu kombinacija: `CTRL + C`.

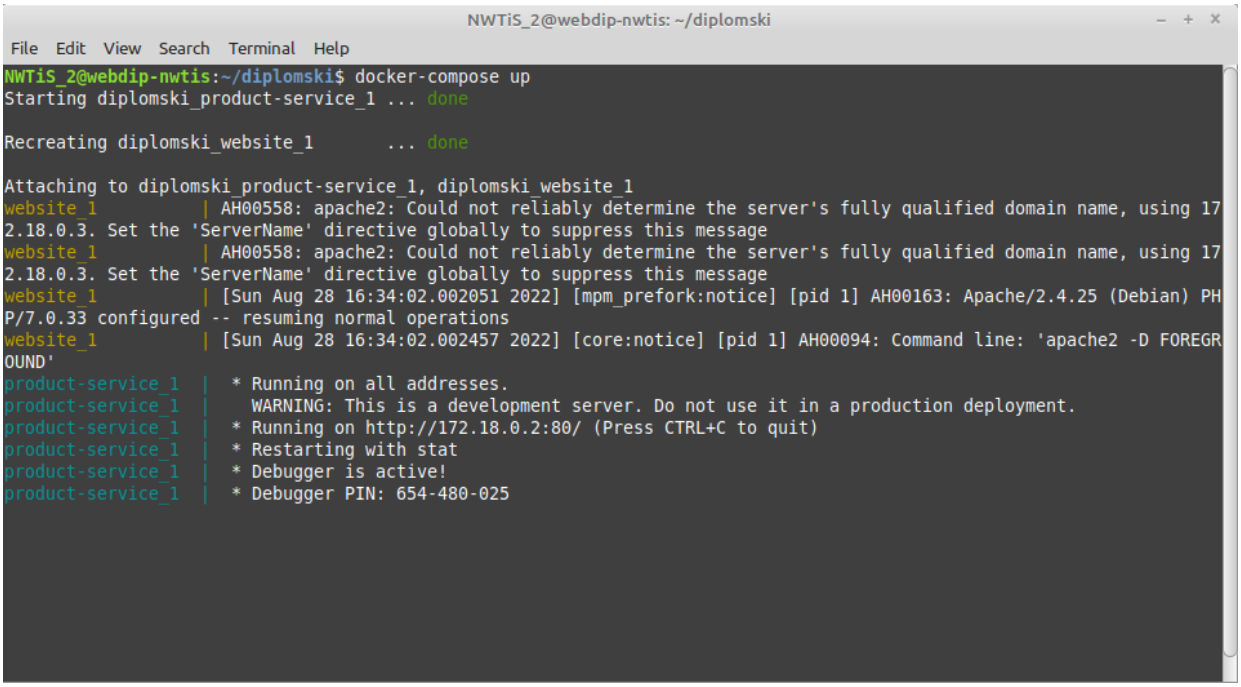
Kako bismo pokrenuli kontejnere pomoću docker-compose alata moramo kreirati „docker-compose.yml“ datoteku u kojem ćemo specificirati sve potrebne ovisnosti koje su bitne kako bismo mogli pokrenuti aplikacije u dva zasebna kontejnera istovremeno.

Sadržaj naše datoteke je sljedeći:

```
version: '3'

services:
  product-service:
    build: ./product
    volumes:
      - ./product:/usr/src/app

  ports:
      - 5001:80

  website:
    build: ./website
    volumes:
      - ./website:/var/www/html
    ports:
      - 5000:80
    depends_on:
      - product-service
```

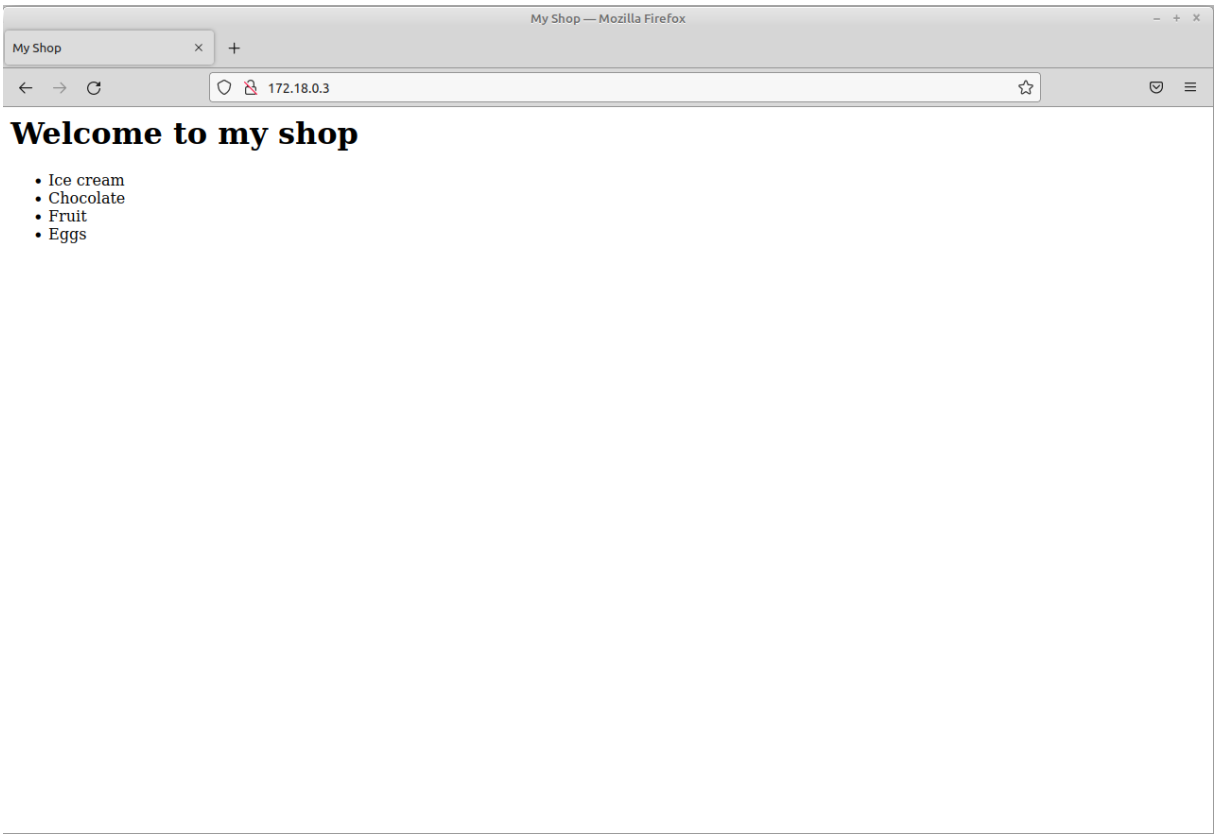
Prvo je trebalo specificirati verziju formata „compose“ datoteke. Nakon toga specificiramo servise koje želimo imati i kao što vidite tu ćemo specificirati „product-service“ koji nam je prije nedostajao da bi naša PHP aplikacija ispravno radila. Zatim specificiramo potrebne volumene kamo želimo da se naša aplikacija kopira što nam olakšava daljnji rad jer na taj način kontejner vidi promjene koda pa ne treba za svaku promjenu kreirati novi kontejner. Jednako tako radimo za portove moramo mapirati koji port na hostu će biti mapiran kojem hostu unutar kontejnera. Kao što smo napravili za PHP dio aplikacije tako sada moramo i za PHP aplikaciju dodati kod unutar datoteke na isti način. Kod ovog dijela moramo povezati servis iz prvog dijela s našom PHP aplikacijom. Kada smo to sve napravili možemo pokrenuti naše aplikacije unutar kontejnera pomoću docker-compose alata. To radimo pomoću sljedeće naredbe:

```
docker-compose up
```



Slika 28. Rezultat pokretanja aplikacije pomoću docker-compose alata (Vlastita izrada, 2022.)

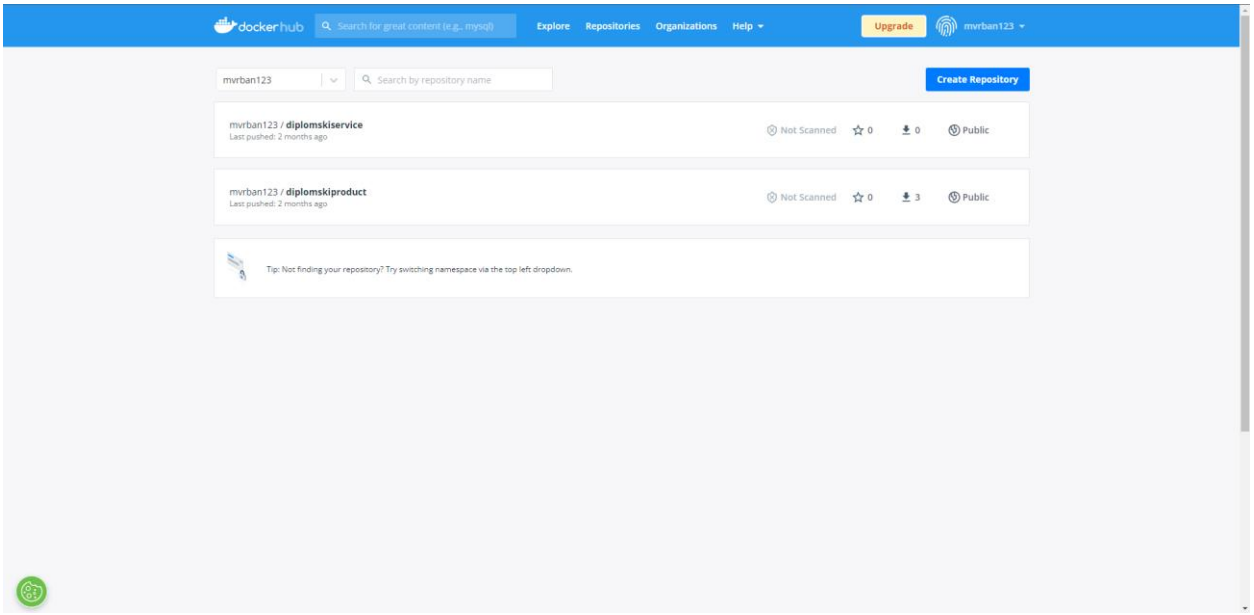
Na slici 28. možemo vidjeti rezultat pokretanja naredbe te na kojoj adresi možemo pristupiti našoj aplikaciji.



Slika 29. PHP aplikacija pokrenuta unutar kontejnera (Vlastita izrada, 2022.)

Sada na slici 29. možemo vidjeti kako naša PHP aplikacija zajedno sa Python aplikacijom radi unutar dva zasebna kontejnera i kako one međusobno komuniciraju. Python aplikacija šalje

podatke a PHP aplikacija prima i ispisuje podatke u listu. Kada smo prošli kroz sve te korake prvo ćemo kreirane slike postaviti(eng. push) u docker hub repozitorije. Za to je prvo potrebno kreirati dva zasebna repozitorija koji su prikazani na slici 30.



Slika 30. Kreirani repozitoriji na Docker hubu (Vlastita izrada, 2022.)

Kako bismo kreirane Docker slike postavili u Docker hub repozitorije potrebno je pokrenuti sljedeće naredbe u naredbenom retku. Za postavljanje Docker slike PHP aplikacije koristi se sljedeća metoda:

```
docker push mvrban123/diplomskiproduct
```

Slika 31. Rezultat pokretanja naredbe za postavljanje slike u Docker hub (Vlastita izrada, 2022.)

Na slici 31. vidimo rezultat postavljanja slike u Docker hub. Sada još moramo napraviti isto za drugu Docker sliku kako bismo ju postavili u Docker hub repozitorij. Za to nam je potrebna sljedeća naredba:

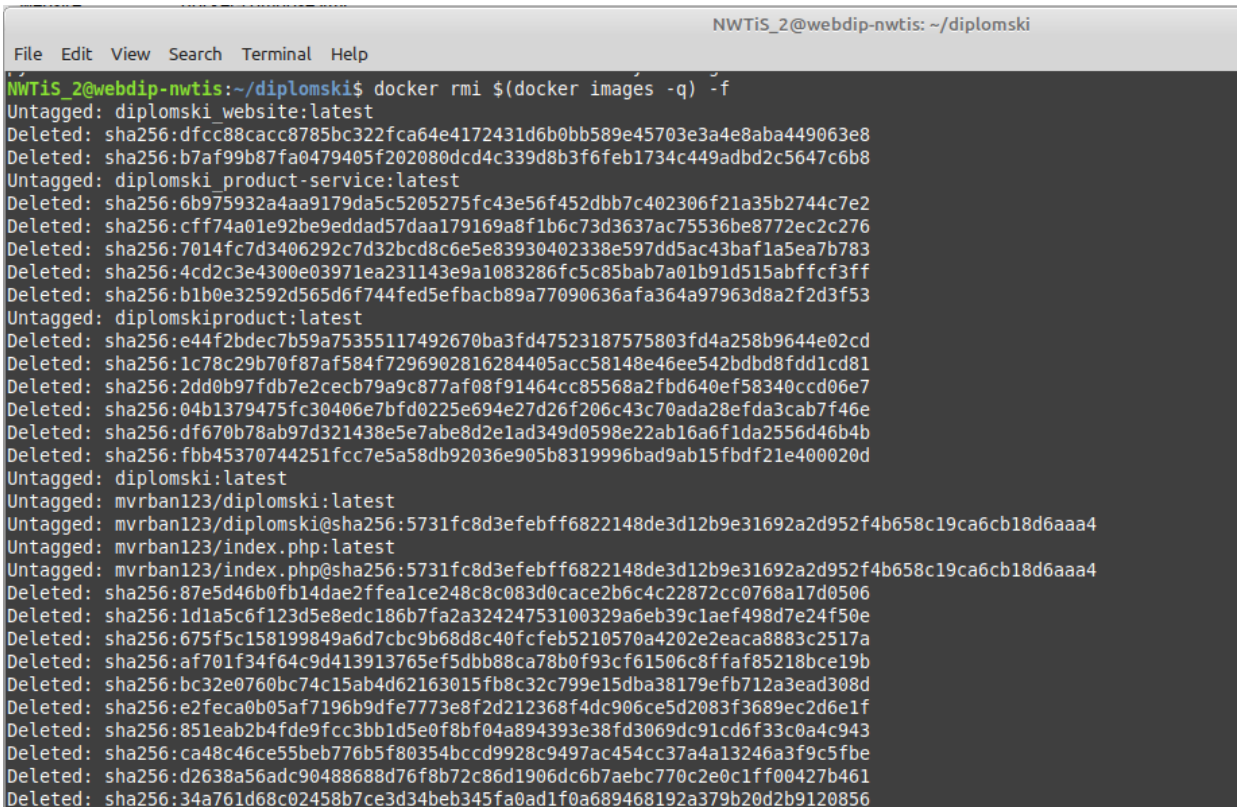
```
docker push mvrban123/diplomskiservice
```

Slika 32. Rezultat pokretanja naredbe za postavljanje slike u Docker hub (Vlastita izrada, 2022.)

Sada kada smo postavili slike u Docker hub repozitorije možemo obrisati slike sa računala. To ćemo učiniti pomoću sljedeće naredbe:

```
docker rmi $(docker images -q) -f
```

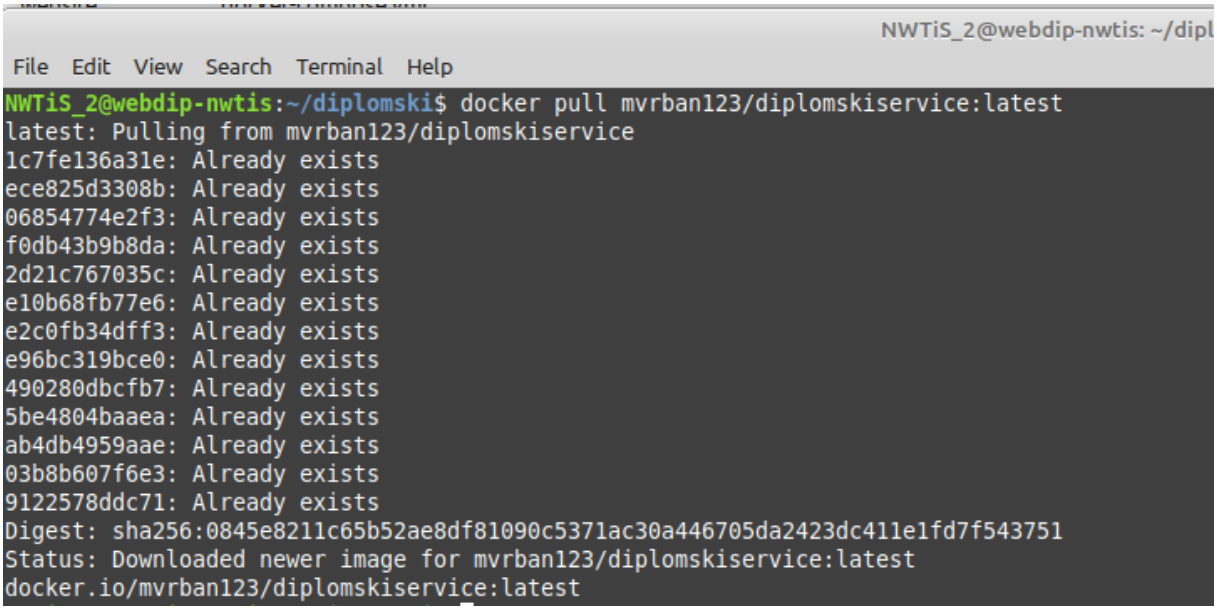
Funkcija rmi služi za brisanje slike a dio koda `$(docker images -q)` služi za dohvaćanje svih slika kako ne bismo trebali specificirati ID svake slike zasebno. Na slici 33. vidimo rezultat brisanja svih slika.



Slika 33. Rezultat brisanja svih Docker slika (Vlastita izrada, 2022.)

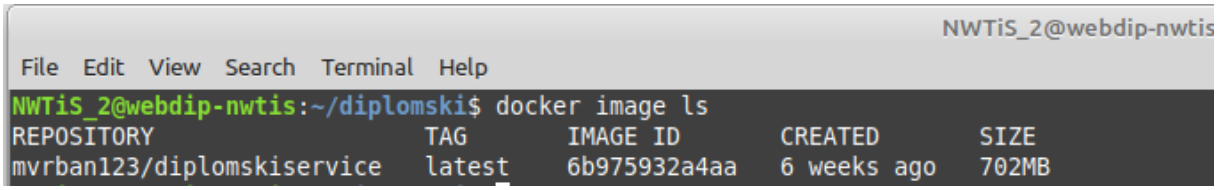
Kada smo obrisali sve slike naše postojeće možemo ponovno povući i dalje koristiti sa Docker hub repozitorija. Kako bi povukli Python aplikaciju potrebna nam je sljedeća naredba:

```
docker pull mvrban123/diplomskiservice:latest
```



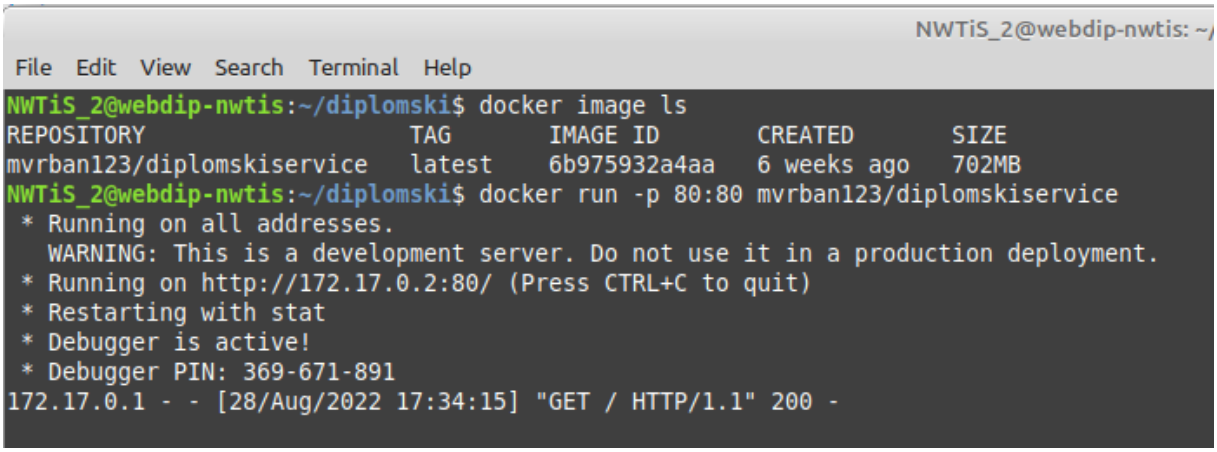
Slika 34. Rezultat naredbe za povlačenje Python aplikacije (Vlastita izrada, 2022.)

Na slici 34. vidimo rezultat pokretanja naredbe za povlačenje Python aplikacije i sada ako ponovno izlistamo sve Docker slike možemo vidjeti kako se samo ona nalazi na našem računalo te ju možemo pokrenuti opet.

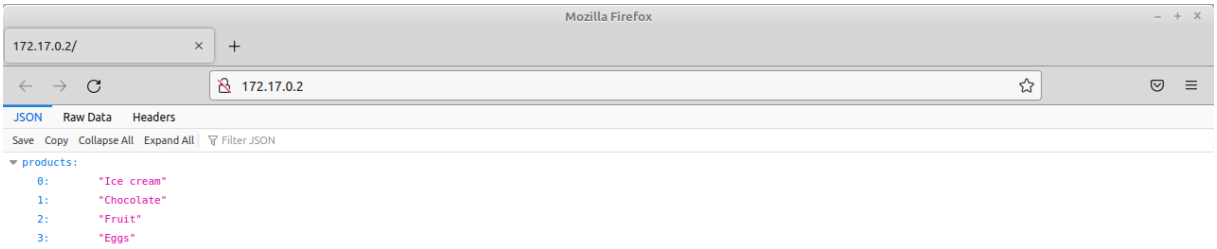


Slika 35. Rezultat izlistavanja svih slika na računalu (Vlastita izrada, 2022.)

Na slici 35. možemo vidjeti rezultat izlistavanja svih Docker slika na računalu, a na slikama 36. i 37. možemo vidjeti kako ponovno možemo samo pokrenuti povučenu sliku na isti način kao i prije.



Slika 36. Rezultat pokretanja Python aplikacije u kontejneru (Vlastita izrada, 2022.)



Slika 37. Prikaz pokrenute Python aplikacije u web pregledniku (Vlastita izrada, 2022.)

Sada kada smo prošli kroz sve vezano uz Docker u sljedećem poglavlju napravit ćemo analizu dobrih i loših strana Docker kontejnera.

4.4. Prednosti i nedostaci

Nakon što smo sve prošli u ovom poglavlju će biti nabrojane neke dobre i loše strane korištenja Dockera koje smo mogli uočiti iz našeg primjera. Dobre strane Dockera su sljedeće:

- Konzistentnost – aplikacije su jednako preko svih operacijskih sustava jednom kada se stvori slika.
- Stabilnost – svaki kontejner se bazira na Linux kernelu te stoga nikad nema problema s ažuriranjima.
- Mogućnost brzog razvoja aplikacija – zbog toga što se aplikacija ne pokreće na operacijskom sustava te se stoga brže mogu vidjeti načinjene promjene.
- Sigurnost aplikacije – aplikacije unutar kontejnera su segregirane jedna od druge i jednako tako od operacijskog sustava.

S druge strane neke od loših strana Dockera su sljedeće:

- potreba za nekom aplikacijom koja manipulira kontejnerima
- Ne mogu se pokrenuti aplikacije koje imaju neko grafičko sučelje, s toga postaje komplicirano s desktop aplikacijama koje imaju grafičko sučelje
- Docker iziskuje veliku krivulju učenja kako bi se potpuno moglo koristiti njime
- Jednostavno alat za pregled statistike kontejnera
- Pokretanje više kontejnera odjednom kada postoji potreba za njima

To su dobre i loše Docker alata koje sam uočio prilikom izrade praktičnog dijela ovog rada.

5. Zaključak

U radu je bila obrađena tema virtualizacije na razini operacijskog sustava. U početku bio je napravljen uvod u sami pojam virtualizacije te čemu on služi. Na koje sve razne načine se može pokrenuti neka aplikacija ili operacijski sustav na već postojećem domaćinu(eng. hostu).

U nastavku bile su detaljno opisane razne metode za virtualizaciju na razini operacijskog sustava kao što su kontejneri(eng. container), virtualne zone, virtualni privatni poslužitelji, odnosno serveri, virtualne jezgre i FreeBSD jail. Najviše pažnje u radu bilo je dano metodi virtualizacije na razini operacijskog sustava kontejnerizaciji (eng. containerization) budući da je ona najpopularnija i najviše se koristi u današnje vrijeme. Za tu metodu bila su detaljnije opisana četiri alata koja su Docker, Linux Containers, Podman i Kubernetes. Najviše pažnje bilo je posvećeno Docker alatu budući da je trenutno najpopularniji od svih.

Nakon što su detaljno bile opisane sve metode virtualizacije na razini operacijskog sustava i neki od alata koji ju omogućuju napravljen je praktični primjer u kojem se primarno koristio Docker alat za kreiranje i pokretanje kontejnera. Za tu svrhu bile su napravljene dvije jednostavne aplikacije, jedna pomoću PHP programskog jezika, a druga pomoću Python programskog jezika, koje međusobno komuniciraju te zajedno čine jednu web aplikaciju. Kada je bila napravljena aplikacija pomoću Docker alata bila je prikazana moguća manipulacija kontejnerima kako bi se aplikacija pokrenula i pristupila u virtualnom okruženju unutar domaćina operacijskog sustava(eng. hosta). Za kraj nakon što se prikazao rad s Docker alatom i kreiranje kontejnera u kojoj je pokrenuta aplikacija napravljena je kratka usporedba dobrih i loših strana Docker alata.

Tema rada mi je bila zabavna i sretan sam što sam odabrao ju odabrao čime sam dobio puno dublje razumijevanje u tehnologiju koja je u današnje vrijeme neophodna u razvoju softvera. Obrađujući temu naučio sam kako se može ubrzati razvoj softvera te kako na brzi i efikasan način pustiti neku aplikaciju u pogon i omogućiti njen pristup korisniku. Ujedno naučio sam kako aplikacije testirati i razvijati na siguran način budući da kontejneri pružaju zasebne virtualne okoline za svaku aplikaciju bez ugrožavanja cijelog sustava. Nadam se da u radu nisam propustio neku bitnu stavku virtualizacije na razini operacijskog sustava budući da je tema opširna i može se još pisati o puno toga.

Popis literature

[1.] „Software Containers: Used More Frequently than Most Realize“ [Online], Hogg S., 2014,
Dostupno: https://www.networkworld.com/article/2226996/software-containers--used-more-frequently-than-most-realize.html
[Pristupljeno: 15-08-2022.]

[2.] „Operating system based Visualization“ [Online], 2021.,
Dostupno: https://www.geeksforgeeks.org/operating-system-based-virtualization/,
[Pristupljeno: 15-08-2022.]

[3.] „Operating System Virtualization(OS Virtualization)“ [Online], 2017.,
Dostupno: https://www.techopedia.com/definition/660/operating-system-virtualization-os-virtualization,
[Pristupljeno: 15-08-2022.]

[4.] „Containers vs. virtual machines“ [Online],
Dostupno: https://www.atlassian.com/microservices/cloud-computing/containers-vs-vms,
[Pristupljeno: 15-08-2022.]

[5.] „Docker“ [Online],
Dostupno: https://www.docker.com/resources/what-container/,
[Pristupljeno: 15-08-2022.]

[6.] „Linux Containers“,
Dostupno: https://linuxcontainers.org/lxd/introduction/,
[Pristupljeno: 16-08-2022.]

[7.] „Podman“,
Dostupno: https://docs.podman.io/en/latest/,
[Pristupljeno: 16-08-2022.]

[8.] „Kubernetes“,
Dostupno: https://kubernetes.io/docs/home/,
[Pristupljeno: 16-08-2022.]

[9.] „Kubernets vs Docker“,
Dostupno: https://www.atlassian.com/microservices/microservices-architecture/kubernetes-vs-docker,

[Pristupljeno: 16-08-2022.]

[10.] „Introduction to Solaris Zones“,
Dostupno: https://docs.oracle.com/cd/E19455-01/817-1592/zones.intro-1/index.html,
[Pristupljeno: 16-08-2022.]

[11.] „Virtual private server (VPS) or Virtual dedicated server (VDS)“,
Dostupno: https://www.techtarget.com/searchitoperations/definition/virtual-private-server-VPS-or-virtual-dedicated-server-VDS,
[Pristupljeno: 17-08-2022.]

[12.] „OpenVZ“
Dostupno: https://wiki.openvz.org/Main_Page,
[Pristupljeno: 17-08-2022.]

[13.] „DragonFly BSD“,
Dostupno: https://www.dragonflybsd.org/features/,
[Pristupljeno: 17-08-2022.]

[14.] „FreeBSD“
Dostupno: https://www.freebsd.org/,
[Pristupljeno: 17-08-2022.]

[15.] „How to install Docker on Linux Mint",
Dostupno: https://linuxhint.com/install_docker_linux_mint/,
[Pristupljeno: 18-08-2022.]

Popis slika