# Alati za virtualizaciju programske podrške

**Vrdoljak, Stijepo**

**Master's thesis / Diplomski rad**

**2019**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **Josip Juraj Strossmayer University of Osijek, Faculty of Humanities and Social Sciences / Sveučilište Josipa Jurja Strossmayera u Osijeku, Filozofski fakultet**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:142:289471

Sveučilište J.J. Strossmayera u Osijeku
Filozofski fakultet
Dvopredmetni diplomski studij nakladništva i informacijske tehnologije

Stijepo Vrdoljak

# Alati za virtualizaciju programske podrške

Diplomski rad

Mentor: doc.dr.sc. Tomislav Jakopec

Osijek, 2019.

Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet

Odsjek za informacijske znanosti

Dvopredmetni diplomski studij nakladništva i informacijske tehnologije

---

Stijepo Vrdoljak

# Alati za virtualizaciju programske podrške

Diplomski rad

Društvene znanosti, Informacijske i komunikacijske znanosti

Mentor: doc.dr.sc. Tomislav Jakopec

Osijek, 2019.

**Prilog: Izjava o akademskoj čestitosti i o suglasnosti za javno objavljivanje**

Obveza je studenta da donju Izjavu vlastoručno potpiše i umetne kao treću stranicu završnog odnosno diplomskog rada.

**IZJAVA**

Izjavljujem s punom materijalnom i moralnom odgovornošću da sam ovaj rad samostalno napravio te da u njemu nema kopiranih ili prepisanih dijelova teksta tuđih radova, a da nisu označeni kao citati s napisanim izvorom odakle su preneseni.
Svojim vlastoručnim potpisom potvrđujem da sam suglasan da Filozofski fakultet Osijek trajno pohrani i javno objavi ovaj moj rad u internetskoj bazi završnih i diplomskih radova knjižnice Filozofskog fakulteta Osijek, knjižnice Sveučilišta Josipa Jurja Strossmayera u Osijeku i Nacionalne i sveučilišne knjižnice u Zagrebu.

U Osijeku, datum

10.9.2019

Stjepan Vrdoljak, 0122211902

ime i prezime studenta, JMBAG

Sažetak

Rad se osvrće na trenutno stanje alata za virtualizaciju programske podrške. Koji su to alati, kako se razvijaju te budućnost i standardi. Rad se sastoji od dva dijela: teorijski i praktični. U teorijskom dijelu rada govori se o tome što je programska podrška (eng. *software*), virtualizacija, koji su najpopularniji alati (*Wmware* i *Oracle VM VirtualBox*) te *Docker* kao *de facto* standard i budućnost virtualizacije programske podrške. Drugi dio rada je praktičan te će se u tom dijelu govoriti o Docker-u i njegovoj povijesti, izrada Docker kontejnera za različite programske platforme i osvrt na alat za upravljanje kontejnerima *Kubernetes*. Kreiran je kontejner koji servira razvojno okruženje za aplikacije izrađene u PHP programskom jeziku, kontejner koji servira web server *nginx* te kontejner koji servira razvojno okruženje za aplikacije izrađene u JavaScript programskom jeziku na server strani. Svaki korak bit će pomno objašnjen i popraćen slikama i uputama kako izraditi isti. Ova tehnologija omogućava velike uštede resursa za male i velike tvrtke. Docker je vrlo pouzdana tehnologija te omogućava vrlo visoku razinu fleksibilnosti i skaliranja projekta kako isti raste.

**Ključne riječi:** korisnički program, virtualizacija, alati za virtualizaciju, docker, kubernetes

# Sadržaj

# 1. Uvod

Kako bi znali što je virtualizacija programske podrške, treba prvo definirati termin programska podrška - „*programska podrška (eng. software) skup je programa i podataka potrebnih za rad računala. Pod tim se pojmom obično razumijevaju svi ne fizički dijelovi računalnoga sustava, za razliku od sklopovlja, koje obuhvaća sve fizičke dijelove.*“ [1] Mjesta na kojima se ne koristi neki vid programske podrške skoro pa i ne postoje. Programska se podrška nalazi unutar računala, pametnih telefona, TV-a, perilice rublja, hladnjaka, automobila, pametnih satova itd. U 21. stoljeću ponekad nismo svjesni koliko smo okruženi programskom podrškom i da je koristimo, a da toga nismo ni svjesni. Slijed strojnih instrukcija koju procesor (CPU) računala provodi jednu za drugom i time obavlja neku zadaću nazivamo računalni program. Naredbe, odnosno strojne instrukcije su na najnižoj razini nizovi od nula i jedinica. Skup više njih nazivamo bajtovima koje CPU dohvaća u svoje registre. Dekodiranjem istih, sklopovi računalnog procesora određuju koja se operacija treba obaviti, koje operande koristiti i kamo smjestiti rezultat cijele operacije. Instrukcije su vrlo jednostavne. Npr. aritmetičke operacije s dva operanda, usporedba dvaju podataka i sl. Samim time ih je potreban veliki broj za opisivanje složenijih zadaća. Kako su takve instrukcije vrlo apstraktne, pojedine kombinacije jedinica i nula se označe simbolima – program se zapiše u mnemoničkom obliku. To uvelike olakšava izradbu programa u strojnom obliku. Strojni jezik je skup strojnih instrukcija dok se skup njihovih simboličkih zamjena naziva simboličkim jezikom. Interpreter/kompajler prevodi simbolički jezik u strojni jezik koji je razumljiv računalnom procesoru. Različiti korisnički programi pretvaraju računalo u prividne strojeve. Npr. strojevi za pisanje, obradu slike ili teksta, crtanje, pregled baza podataka, igrice itd. Korisnički programi se pokreću pomoću skupa programa koji čine operativni sustav. Operativni se sustav sastoji od niza uslužnih programa kojima se prilagođava rad računala i utječe na izvođenje korisničkih programa. Unaprijed pripremljene operacije koriste se kao gotove programske komponente i pri razvoju korisničkih programa. Priprema programa pomoću višega programskoga jezika obavlja se pomoću funkcija koje omogućava operativni sustav. Popis tih funkcija naziva se sučelje korisničkih programa.[2] Tema rada je virtualizacija programske

[1] Usp. Programska podrška. // Hrvatska enciklopedija. URL: http://www.enciklopedija.hr/natuknica.aspx?ID=50557 (2019-07-13)

[2] Usp. Računalni program. // Hrvatska enciklopedija. URL: http://www.enciklopedija.hr/natuknica.aspx?ID=68626 (2019-07-13)

podrške koja će daljnje u tekstu biti bolje pojašnjena. No, laički rečeno virtualizacija programske podrške nam omogućava da nadiđemo granice računalnog sklopovlja. Pomoću alata i tehnologija za virtualizaciju programske podrške možemo pokrenuti više operativnih sustava na jednom fizičkom sklopovlju ili kako je to slučaj sa Docker-om, možemo imati više odvojenih procesa koji se izvode neovisno jedan od drugog sa svim sistemskim i programskim preduvjetima koji su potrebni za funkcioniranje i izvođenje Docker kontejnera. Tehnologija kojom se koristi Docker je vrlo zanimljiva ako gledamo kroz prizmu zaštite digitalnih objekata jer nam upravo Docker omogućava da imamo sve što nam je potrebno da pokrenemo recimo nekakav stari zapis pomoću tehnologije koja se možda ne koristi više ili je zastarjela i nesigurna. Jednom „upakiran" kontejner se izvodi na svim računalima isto. Jedini preduvjet je dakako instaliran Docker. Važnost ove tehnologije je bitna ne samo za gospodarske grane gdje je informacijska tehnologija (skoro da i ne postoji više gospodarska grana u kojoj se ne koristi programska podrška u nekom obliku) nego i za zaštitu informacije i digitalnih objekata u vremenima prekomjerne količine informacija (eng. *information overload*). Prvi koji je taj pojam definirao je Bertram Gross u kojem opisuje taj pojam kao preopterećenje informacijom za donošenje ispravnih odluka. Information overload je primarno problem World Wide Web-a. Najčešće korišten internetski servis se zove WWW (akronim od eng. *World Wide Web*). Umrežena računala diljem svijeta omogućavaju korisnicima pregledavanje digitalnih dokumenata. Pregled tih dokumenata je moguć pomoću protokola za prijenos hiperteksta, odnosno HTTP-a (akronim od eng. *Hypertext Transfer Protocol*) te jedinstvenim adresama svakog dokumenta iliti URL-a (akronim od eng. *Unifrom Resource Locator*). Mrežna adresa opisuje vrstu protokola, naziv računala ili domene kojima dokument pripada i naziv samog dokumenta. Ti dokumenti su okupljeni u mrežne stranice dok više njih u mrežna mjesta te zatim i portale. Pretraživanjem i pregledavanjem web-a se pronalazi odgovarajući sadržaj mrežnih mjesta.[3] Kako bi buduće generacije mogle neovisno o operacijskom sustavu ili platformi koju koriste imati pristup informacijama prijašnjih generacija, prilikom kojih su to već zastarjele tehnologije, alati i tehnologija za virtualizaciju programske podrške omogućava besprijekornu zaštitu i pristup informacijama. Takva tehnologija garantira da će korisnik imati pristup informaciji.

[3] Usp. WWW. // Hrvatska enciklopedija. URL: http://www.enciklopedija.hr/Natuknica.aspx?ID=66413 (2019-07-13)

## 2. Virtualizacija

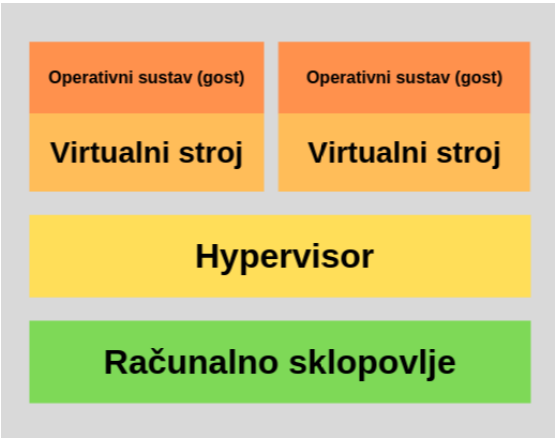
Virtualizacija je proces pokretanja virtualne instance računalnog sustava u sloju koji se izdvaja iz stvarnog sklopovlja. Najčešće se to odnosi na pokretanje više operacijskih sustava na jednom sustavu. Aplikacije koje se izvode na virtualnom stroju funkcioniraju kao da su na „normalnom“ tj. vlastitom računalu. Operativni sustav, programske biblioteke i ostali programi su jedinstveni za virtualni sustav „gosta“ i nisu nikako povezani sa „domaćinom“ odnosno sustavom koji se nalazi ispod njega. [4] Neki od razloga zašto bi se običan korisnik koristio alatima za virtualizaciju programske podrške su: pokretanje aplikacija koje možda nisu dostupne na vlastitom operativnom sustavu, kreiranje specifičnih radnih okolina, testiranje itd. Dok korištenje istih tehnologija za servere ili enterprise korisnike predstavlja uštedu, bolje iskorištavanje resursa, fleksibilnost, skaliranje projekta, osigurava bolju dostupnost usluge, sigurniji backup podataka itd. Kod virtualizacije treba spomenuti pojam *hypervisor* i njegove dvije vrste: „bare metal“ i „hosted“. Hypervisor je program za stvaranje i izvođenje virtualnih strojeva. *Bare metal* se izvodi izravno na hardware-u sustava dok se *hosted* izvode više kao tradicionalne aplikacije koje se mogu pokrenuti i zaustaviti kao bilo koji drugi program. U modernijim sustavima je ova podjela manje zastupljena, osobito kod sustava gdje se koristi Linux. Linux u svom kernelu ima KVM, odnosno kernel-based virtual machine koji može izravno pokrenuti virtualne strojeve, ali se i dalje koristiti kao normalno računalo.[5] Nadalje, imamo virtualne strojeve. Virtualni stroj je emulirani ekvivalent računalnog sustava koji se izvodi na drugom sustavu. Virtualni strojevi mogu imati pristup neograničenom broju resursa: računalna snaga, hardware, ali ograničen pristup računalnom procesoru i memoriji glavnog računala, jedan ili više fizičkih ili virtualnih diskova za pohranu, virtualno ili stvarno mrežno sučelje, kao i sve ostale uređaje kao što su grafičke kartice, USB uređaj ili drugi hardware koji se dijeli sa virtualnim strojem. Ako je virtualni stroj pohranjen na virtualnom disku, to se naziva „slika diska“. Slika diska može sadržavati datoteke za pokretanje virtualnog stroja ili može sadržavati bilo koje druge specifične potrebe za pohranom.[6] Razlika između kontejnera i virtualnog stroja i nisu baš tako male. Iako su konceptualno skoro pa identični i omogućavaju pokretanje aplikacija u izoliranom okruženju te pokretanje više njih na jednom fizičkom stroju, kontejneri ipak nisu u potpunosti neovisni strojevi. Kontejner je samo izolirani proces koji

[4] Usp. What is virtualization? URL: https://opensource.com/resources/virtualization (2019-07-14)

[5] Isto.

[6] Isto.

dijeli isti Linux kernel kao i „domaćin“, odnosno operacijski sustav na kojem se pokreće. Sve programske biblioteke koje su potrebne za pokretanje i mrežno sučelje su unutar kontejnera da mogu funkcionirati kao i virtualni stroj. U pravilu su kontejneri dizajnirani za pokretanje jedne aplikacije dok su virtualni strojevi za emulaciju više individualnih operacijskih sustava na jednom stroju.




**Tip 1 Hypervisor-a**
**(Bare-Metal Architecture)**


**Tip 2 Hypervisor-a**
**(Hosted Architecture)**

**Slika 1.** Dva tipa hypervisor-a[7]

[7] Type 1 and type 2 hypervisor. URL: https://www.nakivo.com/blog/wp-content/uploads/2018/10/Type-1-and-type-2-hypervisor-1024x584.png (2019-07-15)

## 3. Povijest

Povijest virtualizacije datira još iz šezdesetih godina prošlog stoljeća. „*Začetnik virtualizacije šezdesetih godina prošlog stoljeća je bio IBM (1964. godine). Projekt na kojemu su radili se zvao M44/44X gdje se trebao podijeliti fizički sustav na nekoliko različitih virtualnih strojeva. Time bi se poboljšala sama iskoristivost sklopovlja. Centraliziranjem sustava koji može izvoditi više procesa i zadataka koji su stavljeni pred njega bi dovelo do značajne uštede jer u to vrijeme je takvo sklopovlje bilo jako skupo. No da bi virtualizacija bila u potpunosti uspješna, IBM kao vodeća tvrtka toga doba po pitanju virtualizacije je morala imati na umu da se svaki korisnik mora osjećati kao da radi na svome računalu te da programi koje je koristio prije funkcioniraju i sada nakon virtualizacije. Operativni sustav mora biti jednostavan i brz za korištenje, ali najveći cilj je sigurnost toga sustava. Htio se izolirati rad jednog korisnika od drugih koji su radili u isto vrijeme.*“[8] Rane verzije Docker-a bile su „omot“ oko LXC-a[9] uparenog s sindikalnim datotečnim sustavom, ali prihvaćanje i brzina razvoja bili su šokantno brzi. Kasnije je LXC zamijenjen sa njihovim „libcontainer“. U roku od šest mjeseci imao je više od 6700 „zvjezdica“ na GitHub-u i 175 nezaposlenih suradnika. To je dovelo dotCloud (prvotni naziv) da promijeni ime u Docker, Inc. i da ponovno usmjeri svoj poslovni model. Docker 1.0 najavljen je u lipnju 2014. godine i samo 15 mjeseci nakon izlaska verzije 0.1, Docker 1.0 predstavljao je veliki skok u stabilnosti i pouzdanosti – tada je proglašen „spremnim za produkciju“. Iako je već bio na produkciji u nekoliko tvrtki, uključujući Spotify i Baidu. U isto vrijeme Docker je krenuo prema cjelovitoj platformi, a ne samo „motor“ kontejnera. Pokrenut je Docker Hub, javni repozitoriji kontejnera.[10] Danas Docker ima potpuno razrađenu platformu za sve korisnike. Od verzije za zajednicu (Docker Community Edition) do poslovnih rješenja za velike tvrtke (Docker Enterprise), već naveden Docker Hub repozitoriji kontejnera, razni alati, dokumentaciju itd.

---

[8] Zovko, Robert; Žigman, Dubravko. Virtualizacija. // Polytechnic and design 4, 2(2016), str. 196-197. URL: https://hrcak.srce.hr/192099 (2019-08-29)

[9] LXC – sučelje korisničkog prostora za značajke Linux kernela. Kroz moćan API i jednostavne alate, Linux korisnici mogu lako kreirati i upravljati sistemskim ili aplikacijskim kontejnerima. URL: https://linuxcontainers.org/lxc/introduction/ (2019-08-29)

[10] Usp. Mouat, Adiran. Using Docker: The What and Why of Containers. URL: https://www.oreilly.com/library/view/using-docker/9781491915752/ch01.html (2019-08-29)

## 4. VMware

U dostupnim mrežnim izvorima na temu alata za virtualizaciju dobijemo VMware uvijek među prva tri najpoznatija alata.

O tvrtki:

- **Naziv tvrtke**: VMware, Inc.
- **CEO**: Pat Gelsinger
- **Godina osnutka**: 1998.
- **Kratak opis**: VMware je najčešće poznat po sljedećim ključnim riječima – virtualizacija, podatkovni centri, x86 arhitektura [11]

Napomena: svi podaci o tvrtki VMware su preuzeti sa njihove službene stranice. Link: https://www.vmware.com/company.html (2019-07-15)

Sistemski preduvjeti VMware player alata pomoću kojeg se može emulirati drugi operativni sustav na „domaćem“ sustavu:

- CPU: 1 GHz ili brži 64 bitni procesor (preporučeno 2 GHz)
- Memorija: 2GB RAM-a (preporučeno 4GB)
- Diskovni prostor: instalacija zauzima otprilike 150 MB
- Napomena: provjeriti listu podržanih procesora. Procesori proizvedeni 2011. i dalje ne bi trebali imati poteškoće. Opširna lista se nalazi na linku odakle su preuzete informacije.

Izvor: https://www.vmware.com/products/player/faqs.html#installation (2019-07-15)

VMware se može preuzeti sa njihove službene web stranice www.vmware.com pod „Downloads“. (nije naveden link jer su takvi linkovi vrlo često podložni promjenama)

[11] x86 arhitektura – Instruction set architecture (ISA), odnosno serija instrukcija za računalne procesore. Razvila ju je tvrtka Intel Corporation. Definira kako procesor obrađuje i izvršava različite upute koje su prošle operativni sustav i korisničke programe. URL: https://www.techopedia.com/definition/5334/x86-architecture (2019-07-15)

## 5. Oracle VM VirtualBox

Sljedeći predstavnik alata za virtualizaciju programske podrške je VirtualBox od tvrtke Oracle.

O tvrtki:

- **Naziv tvrtke**: Oracle Corporation
- **CEO**: Larry Ellison
- **Godina osnutka**: 1977.
- **Kratak opis**: prema podacima iz 2018. treća najveća tehnološka tvrtka prema dobiti, baze podataka, infrastruktura u oblaku, enterprise programska podrška i dr. su ključne riječi koje opisuju ovu tvrtku.

Napomena: svi podaci o tvrtki su preuzeti s njihove službene web stranice.
Link: https://www.oracle.com/corporate/#info (2019-07-25)

Sistemski preduvjeti Oracle VM VirtualBox alata pomoću kojeg se može emulirati drugi operativni sustav na „domaćem“ sustavu:

- CPU: x86 sklopovlje. Bilo koji nedavno proizveden Intel ili AMD procesor bi trebao zadovoljiti uvjete
- Memorija: minimalno 512 MB, no to varira ovisno o tome koliko domaćin i koliko gostujući OS zahtijeva. Ovisno o tome treba prilagoditi količinu RAM memorije.
- Diskovni prostor: instalacija zauzima 30 MB dok za svaku instancu virtualnog stroja treba osigurati 10 GB prostora i više.
- Napomena: koristiti podržane domaćine i podržane gostujuće operativne sustave.

Izvor: https://www.virtualbox.org/wiki/End-user_documentation (2019-07-25)

VirtualBox se može preuzeti sa njihove službene web stranice https://www.virtualbox.org/ pod „Downloads“. (nije naveden link jer su takvi linkovi vrlo često podložni promjenama)

## 6. Što je to Docker?

Docker je alat dizajniran da olakša stvaranje, implementaciju i pokretanje aplikacija pomoću kontejnera. Kontejneri omogućavaju programeru da „upakira“ aplikaciju sa svim potrebnim instancama, kao što su *libraries* i druge potrebne instance za uspješno izvršavanje. Sve bude otpremljeno kao jedan paket. Na taj način, programer može biti siguran da će aplikacija ispravno raditi na bilo kojem drugom računalu koje pokreće Linux bez obzira na druge postavke koje računalo može imati a koje bi se mogle razlikovati od uređaja koji se koristi za pisanje i testiranje koda. Docker je poput virtualnih strojeva. No, u odnosu na virtualni stroj, umjesto kreiranja potpuno novog operativnog sustava, Docker omogućava korištenje istog Linux kernela koje i sam domaćin koristi. To daje značajno povećanje performansi i smanjuje veličinu aplikacije. Također treba naglasiti kako je Docker otvorenog koda. To znači da svi mogu sudjelovati u razvoju istog ili prilagoditi isti svojim potrebama u smislu dodatnih mogućnosti s kojim se Docker ne isporučuje. Docker je alat dizajniran za developere i sistem administratore. Developeri se mogu fokusirati na pisanje koda bez da moraju razmišljati o tome hoće li se aplikacija izvršavati na sistemu na kojem će biti isporučena. Također omogućuje brzo kreiranje radne okoline pomoću tisuće unaprijed kreiranih programa koji su već dizajnirani za pokretanje u Docker kontejneru kao dio njihove aplikacije. Docker omogućava fleksibilnost i potencijalno smanjuje broj potrebnih sustava te štedi novac. [12] Koliko je tehnologija kontejnera u kontekstu ove tehnologija snažna pokazuju broje za prošlu 2018. godinu i predikcije rasta za sljedećih 5 godina. Naime, 2018. godine globalna vrijednost aplikacija pomoću ove tehnologije je iznosila 1,5 milijardi američkih dolara. Dok se za razdoblje od 2019. – 2025. godine predviđa rast od 26,5 %. [13] Ako uzmemo u obzir da je prva verzija Docker-a izašla 2013. godine, odnosno prije 6 godina i da je ta tehnologija dosegla toliku vrijednost na tržištu je zaista impresivno i govori o tome kakav je ovo skok u odnosu na dosadašnju tehnologiju. Velike multinacionalne kompanije kao što su PayPal, VISA, Societe Generale, Citizens Bank i dr. su samo neki od uspješnica s kojim se Docker hvali na vlastitoj web stranici. [14]

---

[12] Usp. What is Docker? URL: https://opensource.com/resources/what-docker (2019-08-13)

[13] Usp. Application Container Market Analysis Report, 2019. URL: URL: https://www.grandviewresearch.com/industry-analysis/application-container-market (2019-08-13)

[14] Usp. Customer Success, 2019. URL: https://www.docker.com/customers (2019-08-13)

Prilog 1. Docker u brojkama

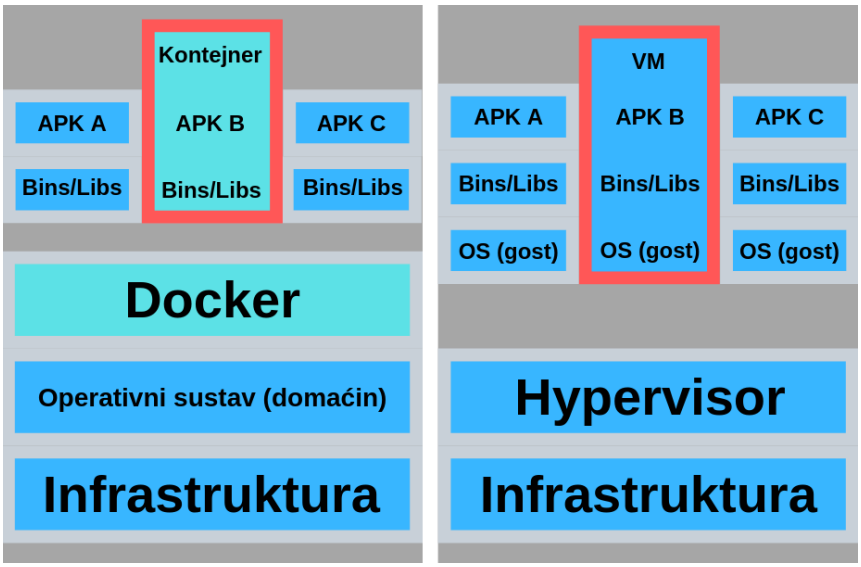
Docker u brojkama:

- 105 milijardi preuzimanja kontejnera
- 750+ Docker Enterprise klijenata
- 200+ konferencija diljem svijeta
- 32 000+ GitHub zvjezdica
- 5.8 milijuna aplikacija na Docker Hub-u
- 100 000 + projekata koji koriste Docker [15]

O tvrtki:

- Naziv tvrtke: Docker, Inc.
- Datum osnutka: 2013. godina
- CEO: Rob Bearden (2019.)
- Gihub Docker-a: https://github.com/docker/docker-ce

Razlika Docker-a u odnosu na virtualni stroj (VM):




**Slika 2.** Docker kontejner vs. virtualni stroj[16]

Na slici 18. se može vidjeti najveća razlika između Docker-a i virtualnog stroja. Docker koristi isti Linux kernel kao i operativni sustav na kojem se nalazi dok virtualni stroj kreira potpuno

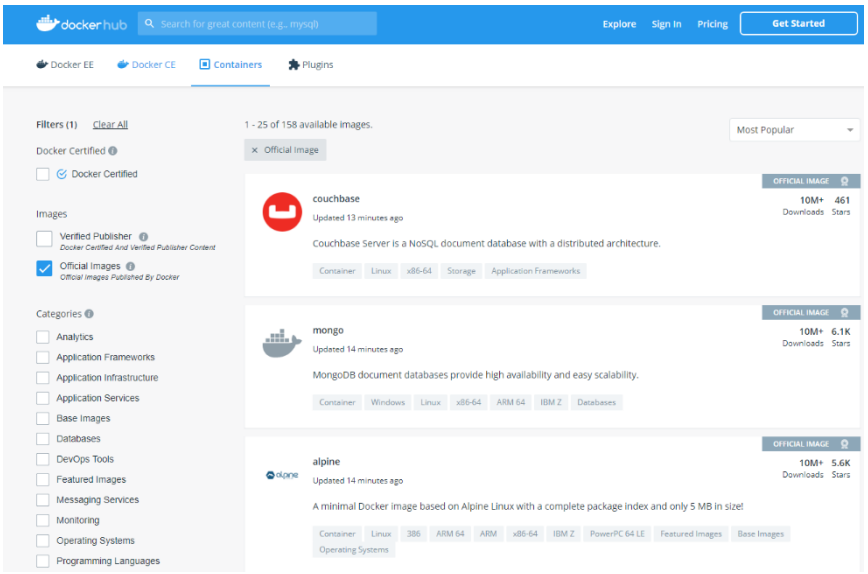
[15] Usp. About Docker. URL: https://www.docker.com/company (2019-08-13)

[16] Usp. Get Started, Part 1: Orientation and setup. URL: https://docs.docker.com/get-started/ (2019-08-14)

novi operativni sustav u ulozi gosta koji se zatim prilagođava ovisno o potrebama aplikacije koja će se izvršavati.

## 6.1 Docker Hub

Docker hub je repozitoriji slika pomoću kojih se dalje nadograđuje, modificira i isporučuje aplikacija u produkciju. Kao tipičan primjer jedne slike unutar Docker Hub-a je Ubuntu slika. Što znači da je Ubuntu slika baza za sve ostalo što će se nadograđivati na temelj koji je u ovom slučaju Ubuntu. Što je na kraju krajeva Docker slika? Docker slika je predložak („Dockerfile“) koji služi da bi definirali sve što će kontejner sadržavati i pokrenuti. Docker kontejner je zapravo Docker slika koja se izvršava dok je Docker slika paket koji uključuje sve što je potrebno za pokretanje aplikacije – kod, životni vijek aplikacije, instance koje su potrebne i konfiguracijske datoteke. Popis svih kontejnera se može provjeriti sa naredbom „*docker ps*“.[17]




**Slika 3.** Docker Hub repozitorij slika[18]

Najpopularnije službene Docker slike su: couchbase, mongo, apline, postgres, redis, ubuntu, node, nginx, mysql, httpd, mariadb, centos, wordpress i mnogi drugi. Docker je vrlo fleksibilan

[17] Get Started, Part 1: Orientation and setup. URL: https://docs.docker.com/get-started/ (2019-08-14)

[18] Usp. Docker Hub: Containers. URL: https://hub.docker.com/search?q=&type=image (2019-08-14)

te se pokreće na svim glavnim platformama a to su Linux, Windows i MacOS. Iako treba naglasiti kako se primarno razvija za Linux zbog korištenja jedinstvenih mogućnosti kernel-a.

## 6.2 Od virtualnih strojeva do kontejnera

U ovom dijelu rada daje se gledište na trenutno stanje tržišta virtualizacije, virtualnih strojeva i tehnologije kontejnera. Najbitnija stavka kod prelaska sa virtualnih strojeva na kontejnere je biti dobro informiran o svim prednostima i nedostacima koje dolaze sa migracijom. O prednostima i nedostacima kod Docker-a će biti riječ dalje u radu. Svakako treba analizirati treba li uopće raditi migraciju. Vrlo često kada se pojavi neka nova *disruptivna* tehnologija (tehnologija koja iz temelja mijenja dosadašnje pristupe i rješenja nekom problemu) slijede migracije koje možda nisu u potpunosti opravdane niti potrebne. Najbolji primjer za to su pojava umjetne inteligencije (Artificial Intelligence - AI). Količina korištenja riječi „umjetna inteligencija“ ili „AI“ u raznoraznim područjima koje to zaista ne koriste ili koriste osnove su više-manje marketing i njegovi trikovi da bi se pokazalo vanjskom svijetu da tvrtka prati trendove i da je „cool“. To je nekad dobro, ali vrlo često i pomalo neodgovorno. Tehnologija kontejnera nije ništa novo, ali su novi pristupi, alati te ponajviše Docker kao takav zaslužan za ogromnu popularnost i mijenjanje tržišta virtualizacije. Ta tehnologija je stara već više od pola stoljeća i polako izrasta u industrijski standard (ako to već nije). Virtualni strojevi su kao takvi i dalje vrlo korisna tehnologija i alati koji se koriste danas će se koristiti i u budućnosti, no fleksibilnost kod odabira prave tehnologije je danas puno veća u odnosu na prije par godina. I jedna i druga tehnologija su tu da ostanu i dominiraju tržištem te se na njih ne treba nužno gledati kao konkurencija, nego kao sazrijevanje tržišta, razvoj i ušteda.

## 6.3 Virtualni stroj

Virtualni stroj je programska implementacija fizičkog stroja koji može imati svoj vlastiti operativni sustav i aplikacije. [19] Virtualni stroj kao takav je identičan kao i fizički stroj dok je najveća prednost iskoristivost jednog fizičkog računalnog sklopovlja sa više različitih operativnih sustava. Preduvjet za virtualizaciju je Hypervisor ili VMM (virtual machine monitor). Popek i Goldberg definiraju tri glavne značajke VMM-a:

- Jednakost - VMM pruža okruženje za programe koje je identično originalnom stroju
- Učinkovitost – Programi koji se izvode u ovom okruženju pokazuju u najgorem slučaju samo neznatno smanjenje performansi u usporedbi s istim programima na originalnom stroju s istim hardware-skim značajkama.
- Kontrola resursa – VMM je u potpunoj kontroli resursa sustava.[20]

Virtualni stroj u konačnici emulira operativni sustav koji ima određene minimalne sistemske zahtjeve koje treba osigurati te se zatim može koristiti prema potrebama kao i operativni sustav u ulozi „domaćina“. Razlike nema.

## 6.4 Kontejner

Docker slike kreiraju Docker kontejner. Kontejneri sadržavaju sve komponente koje su potrebne za uspješan rad aplikacije pa se aplikacija može izvoditi na izolirani način. Na primjer, pretpostavimo da koristimo sliku sa Ubuntu operativnim sustavom koji unutar sebe pokreće SQL server. Kada se ova slika pokrene sa naredbom *docker run*, kreirat će se kontejner koji će

[19] Usp. Aghaalitari, Pooriya. Development of a virtualization systems architecture course for the information sciences and technologies department at the Rochester Institute of Technology (RIT). Rochester: Rochester Institute of Technology, str.24. URL: https://pdfs.semanticscholar.org/a6eb/6e2055a1e76d9f4e5408aaadc532bdc9ff2d.pdf (2019-08-30)

[20] Usp. Aghaalitari, Pooriya. Development of a virtualization systems architecture course for the information sciences and technologies department at the Rochester Institute of Technology (RIT). Rochester: Rochester Institute of Technology, str. 25. URL: https://pdfs.semanticscholar.org/a6eb/6e2055a1e76d9f4e5408aaadc532bdc9ff2d.pdf (2019-08-30)

pokrenuti SQL server na Ubuntu operativnom sustavu. [21] Kada imamo temeljnu sliku s kojom možemo raditi, nastavljamo graditi aplikaciju prema potrebama i zahtjevima. Fleksibilnost, brzina isporuke te brzina u odnosu na virtualni stroj su glavne značajke *kontejnerizacije* iliti u žargonu Docker zajednice *dockerization.* Ova tehnologija uvelike ubrzava development i isporuku. Štedi vrijeme a samim time i novac.

[21] Usp. Ahmadi, Mohammad; Bashari Rad, Babak; Bhatti, John. An Introduction to Docker and Analysis of its Performance // International Journal of Computer Science and Network Security 17, 3 (2017), str. 228-235. URL: http://paper.ijcsns.org/07_book/201703/20170327.pdf (2019-08-30)

# 7. Pokretanje Docker kontejnera i osnovne naredbe

U ovom dijelu rada će biti prikazano kako pokrenuti Docker kontejner i osnovne naredbe koje idu s tim postupkom. Pokrenut će se kako preporučuje i sama dokumentacija „hello world“ aplikacija koja ako uspješno pokrenuta potvrdi ispravnost instalacije.

stijepo@airwing:~$ **docker run hello-world**

```
stijepo@airwing: ~
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:451ce787d12369c5df2a32c85e5a03d52cbcef6eb3586dd03075f3034f10adcd
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

stijepo@airwing:~$
```

**Slika 4.** Pokretanje hello-world programa

Nakon preuzimanja slike i pokretanjem kontejnera. Poruka „This message shows that your installation appears to be working correctly“ potvrđuje da sve ispravno radi.

stijepo@airwing:~$ **docker images**

```
stijepo@airwing: ~
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
hello-world         latest              fce289e99eb9        8 months ago        1.84kB
stijepo@airwing:~$
```

**Slika 5.** Prikaz Docker slika

stijepo@airwing:~$ **docker container ls -a**

```
stijepo@airwing: ~
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
8e16eb8300bc        hello-world         "/hello"            3 minutes ago       Exited (0) 3 minutes ago                        agitated_gagarin
stijepo@airwing:~$
```

**Slika 6.** Prikaz Docker kontejnera

stijepo@airwing:~$ **docker container rm agitated_gagarin**

```
stijepo@airwing: ~
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~$ docker container ls -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS                      PORTS               NAMES
8e16eb8300bc        hello-world         "/hello"            3 minutes ago       Exited (0) 3 minutes ago                        agitated_gagarin
stijepo@airwing:~$ docker container rm agitated_gagarin
agitated_gagarin
stijepo@airwing:~$
```

**Slika 7.** Brisanje kontejnera

Kod brisanja Docker slike treba navesti „NAME“, odnosno ime slike koje se nalazi u zadnjem desnom stupcu.

## 7.1 Instalacija Apache web servera, PHP-a i MySQL-a pomoću Docker-a

U ovom dijelu instalacije će biti prikazana instalacija kontejnera za PHP developer-a. Pomoću ove instalacije se mogu kreirati PHP web aplikacije. Može služiti za razvoj, testiranje, otklanjanje poteškoća i serviranje klijentu.

stijepo@airwing:~$ **mkdir docker-diplomski**

stijepo@airwing:~$ **cd docker-diplomski/**

stijepo@airwing:~/docker-diplomski$ **nano docker-compose.yml**

Kako bi započeli, prvo se treba kreirati mapa u kojoj će se nalaziti sve ostale datoteke. To radimo sa naredbom mkdir (kreiranje mape) te docker-diplomski (naziv mape). Kako bi se pozicionirali na mapu „docker-diplomski“ koristimo naredbu cd (change directory). Nakon što smo pozicionirani unutar novo kreirane mape, potrebno je kreirati novu datoteku pomoću naredbe nano te naziv datoteke (docker-compose.yml).

```
stijepo@airwing: ~/docker-diplomski
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~$ mkdir docker-diplomski
stijepo@airwing:~$ cd docker-diplomski
stijepo@airwing:~/docker-diplomski$ nano docker-compose.yml
```

**Slika 8.** Kreiranje mape *docker-diplomski* i datoteke *docker-compose.yml*

Pritiskom na gumb enter, kreirat će se nova datoteka te možemo nastaviti sa kreiranjem kontejnera i svih potrebnih servisa. Unutar datoteke trebamo temeljnu sliku koja će nam služiti za daljnje kreiranje kontejnera za PHP developera. Repozitorij slika se nalazi na sljedećem linku: https://hub.docker.com/ (2019-08-24).

Službene slike za PHP se isto nalaze u repozitoriju pa je potrebno samo ukucati u tražilicu „PHP“. Slike za PHP se nalaze na sljedećem linku: https://hub.docker.com/_/php (2019-08-24).

stijepo@airwing:~/docker-diplomski$ **mkdir php**

stijepo@airwing:~/docker-diplomski$ **cd php**

stijepo@airwing:~/docker-diplomski/php$ **nano Dockerfile**

```
stijepo@airwing: ~/docker-diplomski/php
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~/docker-diplomski$ mkdir php
stijepo@airwing:~/docker-diplomski$ cd php
stijepo@airwing:~/docker-diplomski/php$ nano Dockerfile
```

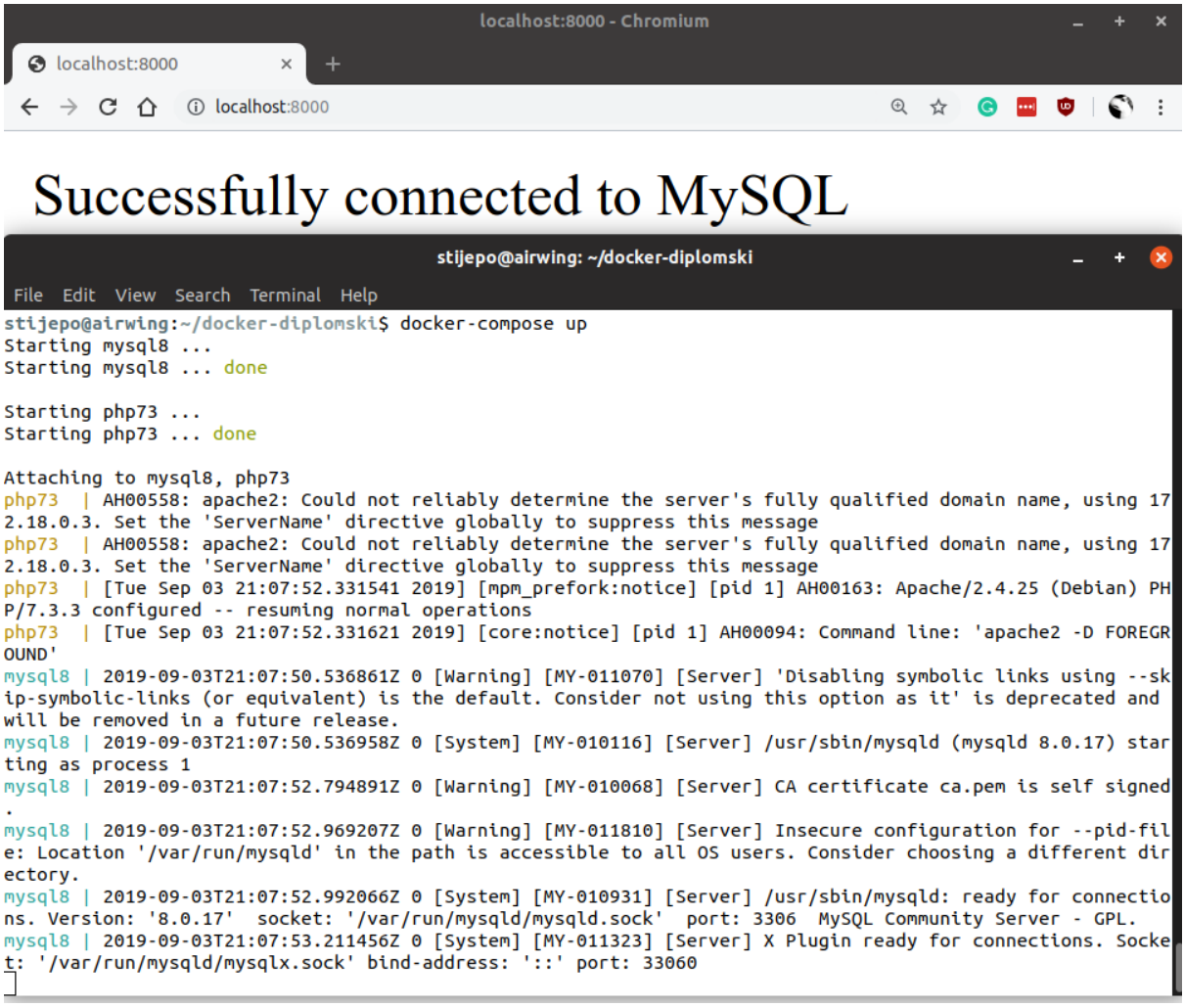
**Slika 9.** Kreiranje mape *php* i datoteke *Dockerfile*

stijepo@airwing:~/docker-diplomski/php$ **nano index.php**

```
stijepo@airwing: ~/docker-diplomski/php
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~/docker-diplomski/php$ nano index.php
```

**Slika 10.** Kreiranje datoteke *index.php*

stijepo@airwing:~/docker-diplomski$ **docker-compose up**




**Slika 11.** Uspješno pokrenuta web aplikacija

Nakon što se sve tri datoteke kreirane, naredbom *docker-compose up* se kreiraju kontejneri te pokretanjem aplikacije na zadanom port-u možemo vidjeti da aplikacija uspješno radi. Sve datoteke i popratni komentari se nalaze na sljedećem link-u: https://github.com/svrdoljak/Docker (2019-08-24).

**docker-compose.yml datoteka**

```
# Verzija docker-a
version: '3.3'
# Servisi
services:
  # Web servis
  web:

    build:
      # Lokalna mapa u kojoj će se nalaziti web aplikacija
      context: ./php
      # Docker datoteka
      dockerfile: Dockerfile
    # Naziv php kontejnera
    container_name: php73
    # Ovaj kontejner ovisi o kontejneru "db", odnosno bazi podataka
    depends_on:
     - db
    # Lokalna mapa "php" će biti proecirana na apache web server unutar
/var/www/html/ radi izvršavana PHP koda
    volumes:
      - ./php:/var/www/html/
    # Port 8000 na kojem se izvršava aplikacija
    ports:
      - 8000:80
  # Servis za bazu podataka
  db:
    # Naziv kontejnera za bazu podataka
    container_name: mysql8
    # Službena mysql slika sa https://hub.docker.com
    image: mysql:8.0
    # Postavke baze podataka
    command: --default-authentication-plugin=mysql_native_password
    # Kad god dode do promjena, restart servisa se radi
    restart: always
    # Dodatne postavke baze podataka
    environment:
       # root lozinka
       MYSQL_ROOT_PASSWORD: root
       # Naziv baze podataka
       MYSQL_DATABASE: test_db
       # Korisnicko ime koje će se koristiti za spajanje na bazu podataka
       MYSQL_USER: svrdoljak
       # Lozinka koja će se koristiti za spajanje na bazu podataka
       MYSQL_PASSWORD: docker123
    # Port 6033 na kojem se izvršava mysql baza podataka
    ports:
      - 6033:3306
```

### Dockerfile datoteka

```
# FROM označava koju temeljnu sliku koristimo. U ovom slučaju php:7.3.3-apache
FROM php:7.3.3-apache
# Ažuriranje sustava
RUN apt-get update && apt-get upgrade -y
# Instalacija mysqli ekstenzije za rad s bazom podataka
RUN docker-php-ext-install mysqli
# Port web aplikacije
EXPOSE 80
```

### index.php datoteka

```
<?php

// Naziv servisa iz docker-compose.yml datoteke
$host = "db";
// Korisničko ime za spajanje na bazu podataka
$user = "svrdoljak";
// Lozinka za spajanje na bazu podataka
$password = "docker123";
// Naziv tablice
$db = "test_db";

// Kreiranje veze na bazu podataka
$conn = new mysqli($host,$user,$password,$db);
// Ako veza ne uspije, poruka "Veza nije uspjela" i greška se ispisuje
if($conn->connect_error){
    echo "Connection failed" . $conn->connect_error;
}
// Ako veza uspije, poruka "Uspješno spojen sa MySQL" se ispisuje
echo "Successfully connected to MySQL";


// PDO primjer
// try {
//  $dbc = new PDO('mysql:host=db;dbname=test_db', 'svrdoljak', 'docker123');
//     $dbc->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
//  $dbc->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY, true);
// } catch(PDOException $e) {
//     echo 'ERROR: ' . $e->getMessage();
// }
// $dbc->query("set names utf8");
```

## 7.2 Instalacija nginx web servera

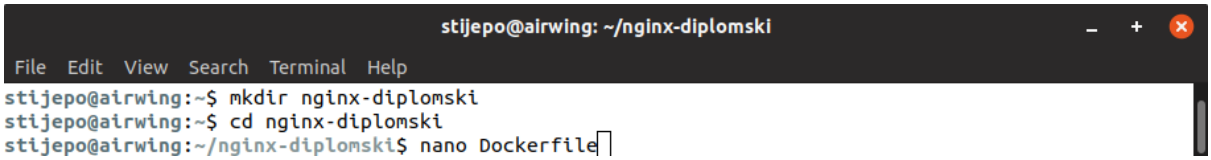
Ovdje će biti prikazan postupak postavljanja nginx web servera kao Docker kontejnera. Nginx (eng. - *engine-x*) je proxy poslužitelj otvorenog koda za HTTP, HTTPS, SMTP, POP3 i IMAP protokole kao i *load balancer* (menadžment opterećenja servera), HTTP pred memorija i web poslužitelj. Projekt nginx započeo je s jakim fokusom na visoku konkurentnost, visoke performanse i malu potrošnju memorije. Pod BSD licencom je, a radi na Linuxu, BSD inačicama, Mac OS X, Solaris, AIX, HP-UX, kao i na drugim *nix inačicama. [22]

stijepo@airwing:~$ **mkdir nginx-diplomski**

stijepo@airwing:~$ **cd nginx-diplomski/**

stijepo@airwing:~/nginx-diplomski$ **nano Dockerfile**

Kako bi započeli, prvo se treba kreirati mapa u kojoj će se nalaziti sve ostale datoteke. To radimo sa naredbom mkdir (kreiranje mape) te nginx-diplomski (naziv mape). Kako bi se pozicionirali na mapu „nginx-diplomski“ koristimo naredbu cd (change directory). Nakon što smo pozicionirani unutar novo kreirane mape, potrebno je kreirati novu datoteku pomoću naredbe nano te naziv datoteke (Dockerfile).




**Slika 12.** Kreiranje mape *nginx-diplomski* i datoteke *Dockerfile*

Pritiskom na enter, kreirat će se nova datoteka te možemo nastaviti sa kreiranjem kontejnera i svih potrebnih servisa. Unutar datoteke trebamo temeljnu sliku koja će nam služiti za daljnje kreiranje kontejnera nginx web servera. Repozitorij slika se nalazi na sljedećem linku: https://hub.docker.com/ (2019-08-30).

Službene slike za nginx se nalaze u repozitoriju  slika pa je potrebno samo ukucati u tražilicu „nginx“. Slike za nginx se nalaze na sljedećem linku: https://hub.docker.com/_/nginx (2019-08-30).

[22] Usp. nginx: Docker Official Images. URL: https://hub.docker.com/_/nginx (2019-08-30)

Nakon odabrane temeljne slike, treba kreirati mapu za statični HTML koji će biti prikazan. Kreirana je mapa *nginx-html* te unutar mape *index.html*.

stijepo@airwing:~/nginx-diplomski$ **mkdir nginx-html**

stijepo@airwing:~nginx-diplomski$ **cd nginx-html/**

stijepo@airwing:~/nginx-diplomski/nginx-html$ **nano index.html**

```
stijepo@airwing: ~/nginx-diplomski/nginx-html
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~/nginx-diplomski$ mkdir nginx-html
stijepo@airwing:~/nginx-diplomski$ cd nginx-html/
stijepo@airwing:~/nginx-diplomski/nginx-html$ nano index.html
```

**Slika 13.** Kreiranje mape *nginx-html* i datoteke *index.html*

Nakon što je kreirano sve potrebno, potrebno je pozicionirati se na početnu mapu (nginx-diplomski) te izvesti sljedeću naredbu:

stijepo@airwing:~/nginx-diplomski$ **docker build -t nginx-diplomski .**

```
stijepo@airwing: ~/nginx-diplomski
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~/nginx-diplomski$ docker build -t nginx-diplomski .
```

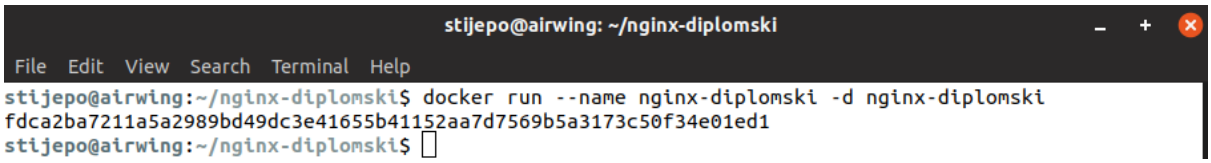
**Slika 14.** Kreiranje kontejnera

Nakon što se naredba izvrši, treba biti prikazana ovakva obavijest:

```
stijepo@airwing: ~/nginx-diplomski
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~/nginx-diplomski$ docker build -t nginx-diplomski .
Sending build context to Docker daemon  3.584kB
Step 1/2 : FROM nginx
latest: Pulling from library/nginx
1ab2bdfe9778: Pull complete
a17e64cfe253: Pull complete
e1288088c7a8: Pull complete
Digest: sha256:53ddb41e46de3d63376579acf46f9a41a8d7de33645db47a486de9769201fec9
Status: Downloaded newer image for nginx:latest
 ---> 5a3221f0137b
Step 2/2 : COPY nginx-html /usr/share/nginx/html
 ---> cebace0296b7
Successfully built cebace0296b7
Successfully tagged nginx-diplomski:latest
stijepo@airwing:~/nginx-diplomski$
```

**Slika 15.** Ispis nakon što je kreiran kontejner

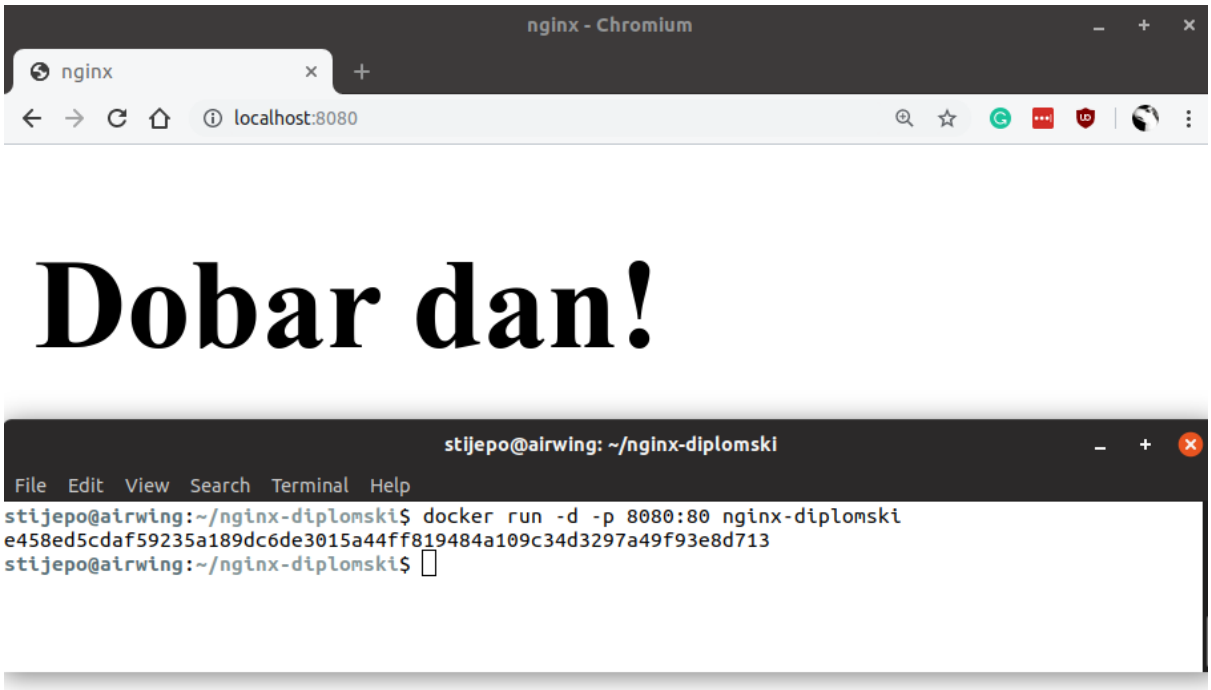
stijepo@airwing:~/nginx-diplomski$ **docker run –name nginx-diplomski -d nginx-diplomski**




**Slika 16.** Pokretanje kontejnera u pozadini

Posljednja naredba za pokretanje kontejnera, odnosno nginx web poslužitelja je:

stijepo@airwing:~/nginx-diplomski$ **docker run -d -p 8080:80 nginx-diplomski**




**Slika 17.** Uspješno pokrenuta web aplikacija

Sve datoteke se nalaze na sljedećem GitHub repozitoriju: https://github.com/svrdoljak/nginx (2019-08-31)

**Dockerfile**

```
# Temeljna slika
FROM nginx
# Kopiranje statičnog html-a u /usr/share/nginx/html za izvršavanje index.html
datoteke
COPY nginx-html /usr/share/nginx/html
```

**index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>nginx</title>
</head>
<body>
    <h1>Dobar dan!</h1>
</body>
</html>
```

## 7.3 Instalacija Node.js programske platforme za JavaScript aplikacije

Ovdje će biti prikazan postupak postavljanja node.js programske platforme kao Docker kontejnera. Node.js programska platforma za skalabilne aplikacije na strani poslužitelja i umrežavanje aplikacija. Node.js aplikacije su napisane u JavaScript programskom jeziku i pokreću se unutar Node.js *runtime-a* na Mac OS X, Windows i Linux bez problema. Node.js aplikacije dizajnirane su za maksimiziranje propusnosti i učinkovitosti. Node.js aplikacije se obično koriste za aplikacije u stvarnom vremenu zbog svoje asinkrone prirode. Interno koristi Google V8 JavaScript *engine* za izvršavanje koda. Velik postotak osnovnih modula je napisan u JavaScript-u. Podrška za HTTP i *socket* omogućava Node.js-u da djeluje kao web poslužitelj bez dodatnog softvera poput Apache ili nginx web servera.

stijepo@airwing:~$ **mkdir nodejs-diplomski**

stijepo@airwing:~$ **cd nodejs-diplomski/**

```
stijepo@airwing: ~/nodejs-diplomski
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~$ mkdir nodejs-diplomski
stijepo@airwing:~$ cd nodejs-diplomski
stijepo@airwing:~/nodejs-diplomski$
```

**Slika 18.** Kreiranje mape *nodejs-diplomski* te pozicioniranje na istu

Kako bi započeli, prvo se treba kreirati mapa u kojoj će se nalaziti sve ostale datoteke. To radimo sa naredbom mkdir (kreiranje mape) te nodejs-diplomski (naziv mape). Kako bi se pozicionirali na mapu „nodejs-diplomski“ koristimo naredbu cd (change directory). Nakon što smo pozicionirani unutar novo kreirane mape, potrebno je kreirati *package.json* pomoću naredbe „npm init“.

stijepo@airwing:~/nodejs-diplomski$ **npm init**

```
stijepo@airwing: ~/nodejs-diplomski
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~/nodejs-diplomski$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (nodejs-diplomski) nodejs_docker_diplomski
version: (1.0.0)
description: Nodejs na Dockeru
entry point: (index.js)
test command: node index.js
git repository:
keywords:
author: Stijepo Vrdoljak
license: (ISC)
About to write to /home/stijepo/nodejs-diplomski/package.json:

{
  "name": "nodejs_docker_diplomski",
  "version": "1.0.0",
  "description": "Nodejs na Dockeru",
  "main": "index.js",
  "scripts": {
    "test": "node index.js"
  },
  "author": "Stijepo Vrdoljak",
  "license": "ISC"
}


Is this ok? (yes)
```

**Slika 19.** Kreiranje *package.json* datoteke

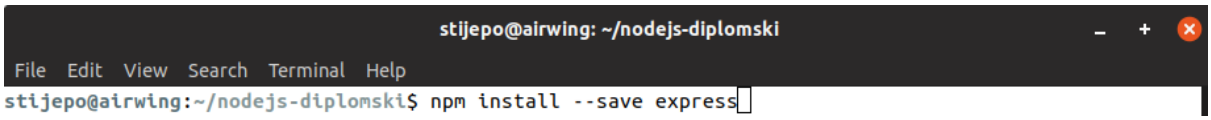
Pritiskom na enter, slijede pitanja o tome što sve *package.json* treba sadržavati. Pritiskom samo tipke enter, postavit će se zadane vrijednosti te se zatim može nastaviti sa kreiranjem kontejnera. Unutar datoteke trebamo temeljnu sliku koja će nam služiti za daljnje kreiranje

Node.js kontejnera. Repozitorij slika se nalazi na sljedećem linku: https://hub.docker.com/ (2019-08-30).

Službene slike za Node.js se nalaze u repozitoriju slika pa je potrebno samo ukucati u tražilicu „node“. Slike za Node.js se nalaze na sljedećem linku: https://hub.docker.com/_/node (2019-08-30).

Sljedećom naredbom će se postaviti Express JavaScript framework:

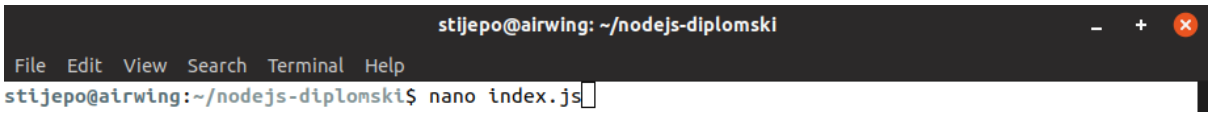
stijepo@airwing:~/nodejs-diplomski$ **npm install –save express**




**Slika 20.** Instalacija Express JavaScript framework-a

Nakon postavljanja Express framework-a slijedi kreiranje *indeks.js* datoteke.

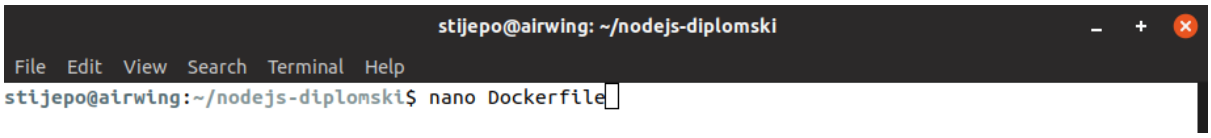
stijepo@airwing:~/nodejs-diplomski$ **nano index.js**




**Slika 21.** Kreiranje *index.js* datoteke

Nakon kreiranja *index.js* datoteke slijedi kreiranje *Dockerfile* datoteke.

stijepo@airwing:~/nodejs-diplomski$ **nano Dockerfile**




**Slika 22.** Kreiranje *Dockerfile* datoteke

Kreiranje kontejnera.

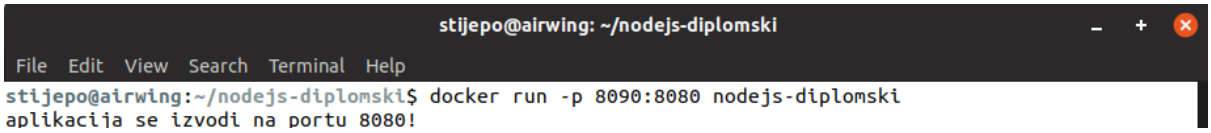
stijepo@airwing:~/nodejs-diplomski$ **docker build -t nodejs-diplomski .**

```
stijepo@airwing: ~/nodejs-diplomski
File  Edit  View  Search  Terminal  Help
stijepo@airwing:~/nodejs-diplomski$ docker build -t nodejs-diplomski .
Sending build context to Docker daemon  2.047MB
Step 1/7 : FROM node:10
10: Pulling from library/node
9cc2ad81d40d: Pull complete
e6cb98e32a52: Pull complete
ae1b8d879bad: Pull complete
42cfa3699b05: Pull complete
053cac798c4e: Pull complete
e11ff976ff71: Pull complete
710b6060865b: Pull complete
ce1774bcd326: Pull complete
753f76cbf7de: Pull complete
Digest: sha256:8eab6272219cb2eef3106a2b2c72ba85627008ebe5f1471dabc76c10e688ad59
Status: Downloaded newer image for node:10
 ---> f7949667ac49
Step 2/7 : WORKDIR /app
 ---> Running in 2fd42da40fb7
Removing intermediate container 2fd42da40fb7
 ---> 2ffa68b27937
Step 3/7 : COPY package.json /app
 ---> b0bb9cf716c6
Step 4/7 : RUN npm install
 ---> Running in 46447a2972fd
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN nodejs_docker_diplomski@1.0.0 No repository field.

added 50 packages from 37 contributors and audited 126 packages in 2.469s
found 0 vulnerabilities

Removing intermediate container 46447a2972fd
 ---> 089ee076b6b3
Step 5/7 : COPY . /app
 ---> 68edee1a4974
Step 6/7 : CMD node index.js
 ---> Running in a2f52a5ef525
Removing intermediate container a2f52a5ef525
 ---> 0e39c22c1237
Step 7/7 : EXPOSE 8090
 ---> Running in 3f8be860310f
Removing intermediate container 3f8be860310f
 ---> e800d5676ccd
Successfully built e800d5676ccd
Successfully tagged nodejs-diplomski:latest
stijepo@airwing:~/nodejs-diplomski$
```

**Slika 23.** Kreiranje kontejnera

Pokretanje aplikacije.

stijepo@airwing:~/nodejs-diplomski$ **docker run -p 8090:8080 nodejs-diplomski**




**Slika 24.** Pokretanje web aplikacije

**Napomena**: iako je obavijest lokalne datoteke da se izvodi na portu 8080, tijekom kreiranja kontejnera smo mapirali port 8090 sa portom 8080. To znači da će se aplikacija izvoditi nakon pokretanja aplikacije na portu 8090. Samim time smo sigurni da se Docker kontejner uspješno pokrenuo.


**Slika 25.** Uspješno pokrenuta web aplikacija

Sve datoteke se nalaze na sljedećem GitHub repozitoriju: https://github.com/svrdoljak/nodejs (2019-09-01)

**Dockerfile**

```
# Temeljna slika
FROM node:10
# Radna mapa
WORKDIR /app
# Kopiranje "package.json" u radnu mapu
COPY package.json /app
# Instalacija svih modula koje su potrebne/zadane u "ovisnostima" u datoteci
package.json
RUN npm install
# Kopiranje svih potrebnih datoteka aplikacije u radnu mapu
COPY . /app
# Pokretanje aplikacije
CMD node index.js
# Port na kojem se aplikacija izvršava
EXPOSE 8090
```

**package.json**

```
{
  "name": "nodejs_docker_diplomski",
  "version": "1.0.0",
  "description": "Nodejs na Dockeru",
  "main": "index.js",
  "scripts": {
    "test": "node index.js"
  },
  "author": "Stijepo Vrdoljak",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

**index.js**

```
var express = require('express')
var app = express()

app.get('/', function (req, res){
    res.send('Dobar dan!')
})

app.listen(8080, function(){
    console.log('aplikacija se izvodi na portu 8080!')
})
```

# 8. Docker konkurencija

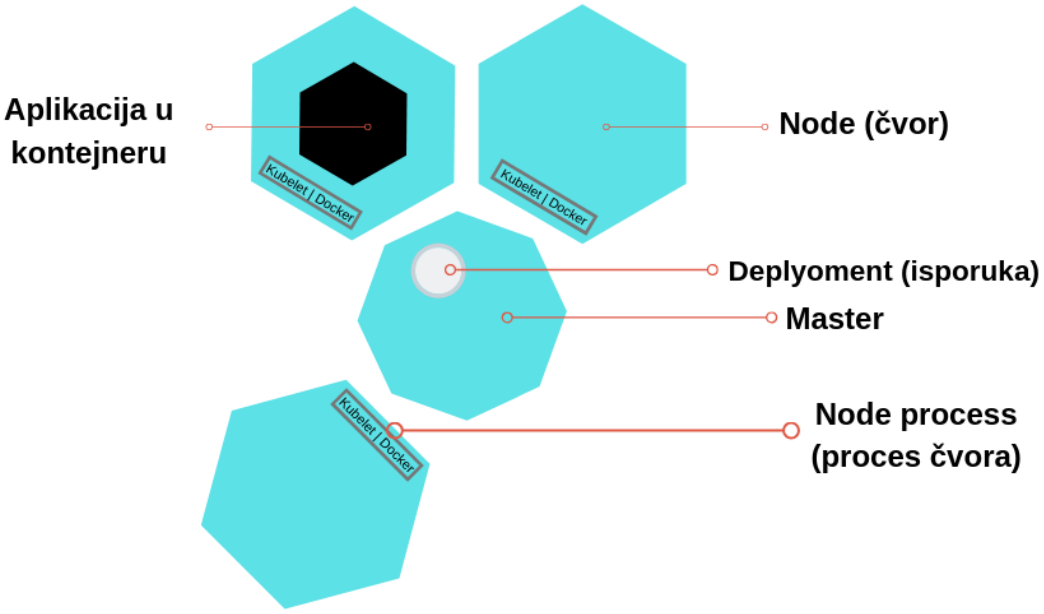
Iako na početku rada piše da je Docker *de facto* standard, postoje i alternativna rješenja koja se mogu koristiti. Konkurencija je uvijek dobra, te samo može poboljšati samu tehnologiju. Kao glavni konkurent Docker-u je isplivao ReactOS. Tehnološki gigant RedHat preuzeo je ReactOS početkom 2018. godine. [23] RedHat u tehnološkom svijetu ne treba previše predstavljati jer njihov biznis model otvorenog koda je zaista vrlo uspješan. Neki od glavnih projekata su RedHat Linux, Fedora, GNOME (desktop okruženje), LibreOffice itd. Sudjeluju aktivno na mnogo projekata otvorenog koda a sve proizvode i usluge je najbolje iščitati s njihove službene web stranice. Dalje, imamo Kubernetes. Treba naglasiti kako Kubernetes nije izravna konkurencija Docker-u. Naime, možemo reći kako postoji sinergija između ove dvije tehnologije. Jedna može funkcionirati bez druge, ali tržište ih je spojilo. Kubernetes je prijenosna i proširiva platforma otvorenog koda za upravljanje kontejnerima, opterećenjima servera i uslugama za deklarativne konfiguracije i automatizaciju. Ima veliki te brzo rastući ekosustav. Kubernetes usluge, podrška i alati su široko dostupni. Ime Kubernetes potječe od grčkog, što znači kormilar ili pilot. Google je dao izvorni kod javnosti 2014. godine. Kubernetes u svojih 15 godina iskustva zrači povjerenjem u kombinaciji s najboljim idejama i najboljim praksama iz zajednice. [24] Ova tehnologija uz Docker je tu da ostane. Isporuka kontejnera u „oblak“ će prema svemu sudeći postati standard u bliskoj budućnosti te bi svaka tvrtka trebala početi adaptirati ovu tehnologiju u vlastitim aplikacijama.

[23] Usp. Red Hat to Acquire CoreOS, Expanding its Kubernetes and Containers Leadership, 2018. URL: https://www.redhat.com/en/about/press-releases/red-hat-acquire-coreos-expanding-its-kubernetes-and-containers-leadership (2019-08-27)

[24] Usp. Kubernetes. URL: https://github.com/kubernetes/kubernetes (2019-09-01)

## 9. Kubernetes

Kubernetes je sustav otvorenog koda za upravljanje kontejnerskim aplikacijama kroz više domaćina, pružajući osnovne mehanizme za implementaciju, održavanje i skaliranje aplikacija. Kubernetes se temelji na desetljeću i pol iskustva u upravljanju Google-ovim proizvodnim opterećenjima korištenjem sustava zvanog Borg u kombinaciji s najboljim idejama i praksama iz zajednice. Kubernetes pokreće Cloud Native Computing Foundation (CNCF).[25] Treba naglasiti kako Kubernetes ne radi samo sa Docker-om nego generalno sa tehnologijom kontejnera.




**Slika 26.** Kubernetes klaster (eng. *cluster*)[26]

[25] Isto.

[26] Usp. Using Minikube to Create a Cluster, 2018. URL: https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/ (2019-09-02)

Kubernetes omogućava sljedeće:

- **Otkrivanje usluge i uravnoteženje opterećenja –** Kubernetes može izložiti kontejner koristeći DNS naziv ili koristeći vlastitu IP adresu. Ako je promet kontejnerom velik, Kubernetes je u stanju rasporediti opterećenje na više servera i distribuirati mrežni promet tako da je usluga stabilna.
- **Orkestracija skladištenja –** Kubernetes omogućuje automatsko montiranje sustava pohrane po izboru, poput lokalnih spremišta, javnih pružatelja usluga oblaka i još mnogo toga.
- **Automatizirano vraćanje sustava i sigurnosne kopije –** omogućava promjenu stanja kontejnera iz trenutnog u željeno stanje. Npr. Automatizirano stvaranje novih kontejnera za implementaciju, uklanjanje postojećih kontejnera i usvajanje svih njihovih resursa u novi kontejner.
- **Automatsko pakiranje –** Kubernetes omogućuje da se odredi koliko CPU snage i RAM memorije treba svakom kontejneru. Kad su zadani zahtjevi kontejnera, Kubernetes može donijeti bolje odluke u upravljanju resursima za iste.
- **Samo-oporavak –** Kubernetes ponovno pokreće kontejnere koje se nisu pokrenuli, zamjenjuje kontejnere, gasi kontejnere koji ne odgovaraju na zadane postavke oporavka i provjere te ih ne isporučuje klijentima sve dok nisu spremni za posluživanje
- **Konfiguriranje sigurnosti –** Kubernetes omogućuje pohranjivanje i upravljanje osjetljivim podacima poput lozinki, OAuth tokena i SSH ključeva. Može se implementirati i ažurirati konfiguracija aplikacije bez ponovnog kreiranja slike kontejnera i otkrivanja istih u postavkama.[27]

## 9.1 Kubernetes objekti

Vrlo često se uz Docker veže i Kubernetes. Kubernetes je alat koji služi za automatsku isporuku kontejnera, skaliranje i upravljanje istim. Docker ima također svoj alat za upravljanje kontejnerima pod nazivom Docker Swarm. U svojoj srži su skoro pa identični alati. No, Kubernetes podržava veći promet uz veću kompleksnost dok Docker Swarm nudi jednostavno rješenje s kojim je lako započeti. Docker Swarm je bio poprilično popularan među programerima koji preferiraju brze implementacije i jednostavnost. Istovremeno, Kubernetes

[27] Usp. What is Kubernetes, 2019. URL: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ (2019-09-01)

koriste u proizvodnim okruženjima popularnih tvrtki koje vode neke od najpopularnijih usluga danas.[28] Međutim tržište je prigrlilo kombinaciju Kubernetes + Docker kontejnere. Kako bi demistificirali osnovnu terminologiju oko Kubernetes-a, slijede osnovni objekti istog:

- **Pod –** osnovna izvršna jedinica Kubernetes aplikacije. Najmanja i najjednostavnija jedinica u Kubernetes-ovom modelu objekata koja se kreira ili implementira. Pod predstavlja procese koji se izvode na klasteru (*eng. cluster*). Klaster je skup strojeva, nazvanih *čvor* koji pokreću kontejnerske aplikacija kojima upravlja Kubernetes. Klaster ima najmanje jedan radni čvor i najmanje jedan glavni čvor.[29]
- **Service –** apstraktni način eksponiranja aplikacije koja se izvodi na skupu Pod-ova kao mrežna usluga. S Kubernetes-om se ne mora mijenjati aplikacija za upotrebu nepoznatog mehanizma otkrivanja usluge. Kubernetes daje Pod-ovima vlastite IP adrese i jedinstveni DNS za skup Pod-ova te ih može međusobno balansirati.
- **Volume –** datoteke na disku u kontejneru su prolazne što predstavlja probleme za netrivijalne aplikacije tijekom pokretanja u kontejneru. Prvo, ako se kontejner sruši, kubelet će ga ponovno pokrenuti, ali datoteke će se izgubiti – kontejner započinje s čistim stanjem. Drugo, pri pokretanju kontejnera zajedno u pod-u često je potrebno dijeliti datoteke između tih kontejnera. Kubernetes Volume apstrakcija rješava oba problema.
- **Namespace –** Kubernetes podržava više virtualnih klastera poduprtih istim fizičkim klasterom. Ti virtualni klasteri se nazivaju *namespace.*[30]

[28] Usp. Kubernetes vs. Docker Swarm: What's the Difference?, 2018. URL: https://thenewstack.io/kubernetes-vs-docker-swarm-whats-the-difference/ (2019-09-01)

[29] Usp. Standardized Glossary, 2019. URL: https://kubernetes.io/docs/reference/glossary/?all=true#term-cluster (2019-09-02)

[30] Usp. Concepts, 2019. URL: https://kubernetes.io/docs/concepts/ (2019-09-02)

## 10. Prednosti i nedostaci Docker-a

Kako bi lakše razumjeli s kakvom tehnologijom radimo te koje su njene prednosti i nedostaci, slijede ključne točke za i protiv. Ovo može poslužiti kao kratak vodič, treba li uvesti Docker u okruženje ili ipak ne.

Prednosti:

- **Povratak investicije i štednj**a – Docker-ova glavna prednost je povratak investicije. To se pogotovo očituje kod velikih tvrtki. Spuštanje troškova te rast profita. Razlog tomu je potreba za manjom količinom računalne snage u odnosu na virtualne strojeve.
- **Brza isporuka –** isporuka proizvoda/usluge se može svesti na sekunde. To je zbog činjenice što se može kreirati kontejner za svaki proces bez potrebe za novim operativnim sustavom.
- **Sigurnost –** Docker osigurava da se aplikacije koje su „dockerizirane“, odnosno nalaze u zasebnim kontejnerima potpuno izolirane jedna od druge. Gledajući kroz prizmu sigurnosti, to nam daje potpunu kontrolu protoka prometa i upravljanja njime.
- **Jednostavnost i brža konfiguracija –** način na koji Docker pojednostavljuje stvari je jedno od njegovih glavnih prednosti. Korisnicima daje fleksibilnost tijekom konfiguriranja, implementacije koda te isporuke bez ikakvih problema. Također, kao jedno od velikih prednosti je mogućnost funkcioniranja Docker-a na različitim platformama i okruženjima.
- **Slika kontejnera -** uz pomoć Docker-a se mogu izraditi slike koje se mogu koristiti za svaki proces tijekom daljnjeg razvoja aplikacije. Prednost u tome je mogućnost odvajanje neovisnih koraka i njihovo paralelno izvođenje. Uz to, potrebno vrijeme za izradu aplikacije do njene isporuke se znatno smanjuje.
- **Kontinuirana integracija –** Docker se kontinuirano razvija te se izrađuju i alati koju pomažu tijekom testiranja, izrade i isporuke aplikacije kao što su Travis, Jenkins i Wrecker.[31]

[31] Usp. Advantages and Disadvantages of Docker – Learn Docker, 2018. URL: https://data-flair.training/blogs/advantages-and-disadvantages-of-docker/ (2019-08-27)

Nedostaci:

- **Nedostatak značajki** – u tijeku je mnoštvo zahtjeva za dodatnim značajkama/mogućnostima kao što je samo-registracija kontejnera, kopiranje datoteka s „domaćina“ na kontejner i mnogi drugi.

- **Podaci u kontejneru –** ponekad kontejner neće raditi te za to treba sigurnosna kopije i oporavak. Iako postoji nekoliko rješenja, ona nisu automatizirana i skalabilna.

- **Brzina izvršavanja aplikacije –** ako usporedimo brzinu izvođenja Docker-a u odnosu na virtualne strojeve, Docker je brži, ali ne brži od izvođenja aplikacije na vlastitom računalnom sklopovlju i potrebnim instancama. Ovdje se i dalje radi o virtualizaciji programske podrške.

- **Podrška za više platformi –** jedan od najvećih problema kod izvođenja Docker-a je ako podržava jednu platformu (recimo Windows), ta ista aplikacija neće raditi na Linux-u i obrnuto. Virtualni strojevi nemaju takva ograničenja.

- **Pokretanje aplikacija sa grafičkim sučeljem –** Docker je dizajniran za pokretanje aplikacija u znakovnom okruženju. Iako postoje mogućnosti za pokretanje kontejnera koji pokreću grafička sučelja, to još nije na zavidnoj razini. Dakle, za aplikacije koje zahtijevaju bogata grafička sučelja, Docker nije najbolje rješenje.

- **Problem sigurnosti –** još uvijek treba evaluirati sve sigurnosne probleme koje Docker nosi sa sobom. Razlog tomu su novi izazovi koje on nosi sa sobom te praćenje svih komponenti u velikim i dinamičnim okruženjima.[32]

Prednosti kao i nedostataka ima sa svakom tehnologijom i alatom. Kako bi izbjegli neugodne situacije ili situacije u kojima se može izgubiti novac, potrebno je biti informiran i donijeti ispravnu poslovnu odluku. Docker donosi odlične prednosti sa sobom, ali također nosi i određenu fazu prilagodbe i učenja. Adaptirati novu tehnologiju znači ulaganje u zaposlenike i njihov razvoj. To dakako košta. Proces učenja potaknut iskustvom kreiranja kontejnera za jednostavne stvari može biti jednostavan i brz, ali za puno kompleksnije i veće sustave donosi ozbiljnu količinu vremena i učenja koju je potrebno utrošiti. Za velike sustave to znači cijele

[32] Isto.

timove koji rade na implementaciji ove tehnologije u kombinaciji sa nekim upraviteljem kontejnera (npr. Kubernetes). Fleksibilnost je ogromna, ali je bitna i stručnost prilikom implementacije iste.

# 11. Zaključak

Tehnologija virtualizacije nije ništa novo. Baš naprotiv, ista je stara više od 50 godina. No kontinuiranim razvojem te boljim implementacijama i sve bolji alati su omogućili virtualizaciju programske podrške na do sad neviđene razine. Da bi shvatili koliko je razvoj tehnologije virtualizacije napredovao treba se okrenuti tvrtkama kao što su Amazon, Google i Microsoft. To su glavni tehnološki giganti koji pokreću internet i u velikoj mjeri drže njegovu infrastrukturu. Ne treba previše predstavljati iste jer su kreirali cijela carstva gledajući kroz prizmu informacijske tehnologije. Ovisnost o njihovim proizvodima i uslugama možda i nije u potpunosti jasna, ali je od iznimne važnosti modernom društvu. Upravo je i virtualizacija jedan od razloga zašto su gore navedene tvrtke uspješne. Izlaskom Docker-a u verziji 1.0 se tehnologija virtualizacije popela na jednu stepenicu više u odnosu na sve dosadašnje viđeno. *Kontejnerizacija* programske podrške je novi primjer *disruptivne* tehnologije. Iako ova tehnologija postoji već duže vrijeme, tek pojavom novih tvrtki kao što su Docker se počelo sve više govoriti o istoj. Razlog tomu je jednostavnost implementacije i rješavanje problema na svim instancama prilikom kreiranja, isporuke i kontrole. Dakle, postoji cijeli ekosustav i podrška koja se nalazi oko Docker kontejnera. Docker kontejneri su danas jedna od glavnih tema u tehnološkom svijetu i trend koji će svi slijediti. Uštede i prednosti koje Docker nosi su jasne i vidljive te u pravilu nema argumenata za ne implementiranje ove tehnologije u vlastitim redovima. Docker pokaže najviše u najvećim sustavima koji su kompleksni te ih treba pojednostaviti implementacijom kontejnera i mikro-servisa. Kada broj kontejnera dosegne određenu kritičnu „masu“ dolaze na vidjelo alati za upravljanje, skalabilnost i isporuku. Trenutno najpoznatiji alat za to je Kubernetes koji omogućava sjajnu fleksibilnost i kontrolu sve u svrhu bolje dostupnosti usluge i lakše ažuriranje od strane developera. Tržište je spojilo možemo reći konkurente te zajedno funkcioniraju i najčešće se jedan veže uz drugi. Docker i Kubernetes su „in“. S razlogom su u trendu jer omogućavaju sjajnu kontrolu, nadzor, upravljanje i razvoj. Tržište virtualizacije je trenutno vrlo turbulentno te su predviđanja za rast vrlo pozitivna i u skoroj budućnosti se očekuje isključivo rast. Može se zaključiti kako je trenutno stanje virtualizacije programske podrške od svojih začetaka do danas u fazi rasta i iznimne popularnosti. Tehnologija je dovoljno zrela da velike korporacije objeručke prihvaćaju sve što im nudi. Budućnost je vrlo optimistična za sve dionike te se može samo nestrpljivo iščekivati što nam nosi u budućnosti.

## Literatura


1. About Docker. URL: https://www.docker.com/company (2019-08-13)
2. Advantages and Disadvantages of Docker – Learn Docker, 2018. URL: https://data-flair.training/blogs/advantages-and-disadvantages-of-docker/ (2019-08-27)
3. Aghaalitari, Pooriya. Development of a virtualization systems architecture course for the information sciences and technologies department at the Rochester Institute of Technology (RIT). Rochester: Rochester Institute of Technology, 2014. URL: https://pdfs.semanticscholar.org/a6eb/6e2055a1e76d9f4e5408aaadc532bdc9ff2d.pdf (2019-08-30)
4. Ahmadi, Mohammad; Bashari Rad, Babak; Bhatti, John. An Introduction to Docker and Analysis of its Performance // International Journal of Computer Science and Network Security 17, 3 (2017), str. 228-235. URL: http://paper.ijcsns.org/07_book/201703/20170327.pdf (2019-08-30)
5. Application Container Market Analysis Report, 2019. URL: https://www.grandviewresearch.com/industry-analysis/application-container-market (2019-08-13)
6. Concepts, 2019. URL: https://kubernetes.io/docs/concepts/ (2019-09-02)
7. Customer Success, 2019. URL: https://www.docker.com/customers (2019-08-13)
8. Docker Hub: Containers. URL: https://hub.docker.com/search?q=&type=image (2019-08-14)
9. Get Started, Part 1: Orientation and setup. URL: https://docs.docker.com/get-started/ (2019-08-14)
10. Kubernetes vs. Docker Swarm: What's the Difference?, 2018. URL: https://thenewstack.io/kubernetes-vs-docker-swarm-whats-the-difference/ (2019-09-01)
11. Kubernetes. URL: https://github.com/kubernetes/kubernetes (2019-09-01)
12. Mouat, Adiran. Using Docker: The What and Why of Containers. URL: https://www.oreilly.com/library/view/using-docker/9781491915752/ch01.html (2019-08-29)
13. nginx: Docker Official Images. URL: https://hub.docker.com/_/nginx (2019-08-30)

14. Programska podrška. // Hrvatska enciklopedija. URL: http://www.enciklopedija.hr/natuknica.aspx?ID=50557 (2019-07-13)
15. Računalni program. // Hrvatska enciklopedija. URL: http://www.enciklopedija.hr/natuknica.aspx?ID=68626 (2019-07-13)
16. Red Hat to Acquire CoreOS, Expanding its Kubernetes and Containers Leadership, 2018. URL: https://www.redhat.com/en/about/press-releases/red-hat-acquire-coreos-expanding-its-kubernetes-and-containers-leadership (2019-08-27)
17. Standardized Glossary, 2019. URL: https://kubernetes.io/docs/reference/glossary/?all=true#term- (2019-09-02)
18. Type 1 and type 2 hypervisor. URL: https://www.nakivo.com/blog/wp-content/uploads/2018/10/Type-1-and-type-2-hypervisor-1024x584.png (2019-07-15)
19. Using Minikube to Create a Cluster, 2018. URL: https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-intro/ (2019-09-02)
20. What is Docker? URL: https://opensource.com/resources/what-docker (2019-08-13)
21. What is Kubernetes, 2019. URL: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/ (2019-09-01)
22. What is virtualization? URL: https://opensource.com/resources/virtualization (2019-07-14)
23. What's LXC? URL: https://linuxcontainers.org/lxc/introduction/ (2019-08-29)
24. WWW. // Hrvatska enciklopedija. URL: http://www.enciklopedija.hr/Natuknica.aspx?ID=66413 (2019-07-13)
25. x86 Architecture. // Techopedia. URL: https://www.techopedia.com/definition/5334/x86-architecture (2019-07-15)
26. Zovko, Robert; Žigman, Dubravko. Virtualizacija. // Polytechnic and design 4, 2(2016), str. 196-200. URL: https://hrcak.srce.hr/192099 (2019-08-29)