Kontejneri i orkestracija kontejnera

Osnovni pojmovi, Docker, Docker Compose, Kubernetes

1

Sadržaj

- Teorijski osnovi
- Docker
- Docker Compose
- Kubernetes

2

Teorijski osnovi

3

Kontejnerizacija - uvod

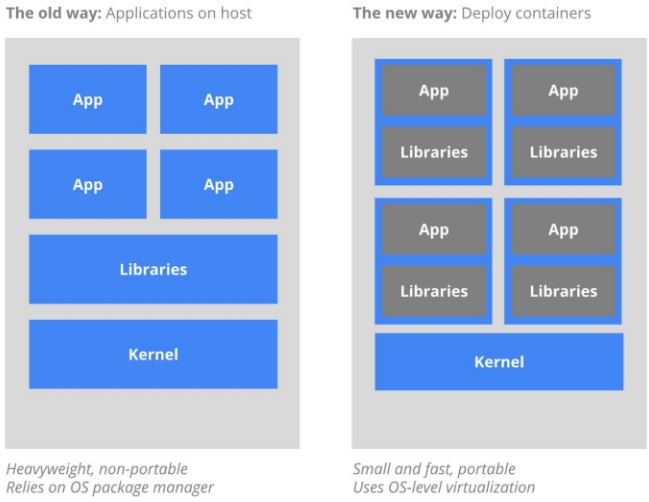
- **Kontejnerizacija** je metod virtualizacije na nivou operativnog sistema
  - koja koristi Kernel operativnog sistema kako bi opskrbila više (distribuiranih) aplikacija pokrenutih u okviru zasebnih kontejnera
- **Kontejner** je izvršiv i izolovan proces u operativnom sistemu
  - koji predstavlja enkapsulaciju aplikacije
  - zajedno sa svim neophodnim dodatnim softverom koji je potreban za izvršavanje aplikacije
- Kontejneri
  - se **ne instaliraju**
    - već se pokreću posredstvom posebnog softvera za rad sa kontejnerima
    - što omogućava lakšu **portabilnost** između različitih hardversko/softverskih platformi
  - omogućavaju lako "pakovanje" aplikativnog koda, konfiguracionih datoteka kao i svih potrebnih biblioteka
    - u lako prenosive i ponovno iskoristive softverske pakete

4

4

Kontejnerizacija - uvod

- Menja i način na koji posmatramo razvoj aplikacija
  - usmeren ka kreiranju manjih, izolovanih celina (**mikroservisa**)
    - iako je moguće kontejnerizovati i monolitne aplikacije
  - kontejneri su najčešće korišćeno sredstvo za izvršavanje mikroservisa

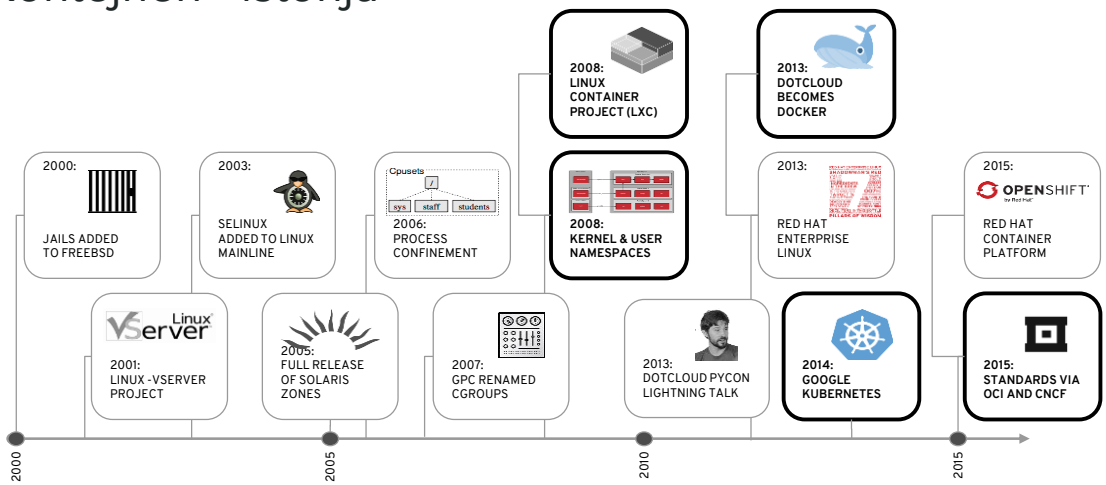


izvor: *Kubernetes Overview, Kubernetes docs* https://kubernetes.io/docs

5

5

Kontejneri - istorija



izvor: *Understanding Container Standards, Scott McCarty* https://medium.com/cri-o/understanding-container-standards-1e1448cbb92c

6

6

Kontejnerizacija

- Korišćenje kontejnera omogućava
  - **konzistentnost okruženja** u kojima se **izvršava** softver
    - uniformno verzionisanje softvera i svih zavisnosti u kontejnerima
    - isti kontejner u razvojnom, testnom i izvršnom okruženju
      - potencijalno sa različitim podešavanjima i parametrima
  - **konzistentnost okruženja** u kojima se **razvija** softver
    - kontejnerizacija svih zavisnosti i razvojnog okruženja
    - kontejnerizacija potrebnih mikroservisa
  - **efikasnije izvršenje aplikacija**
    - brzo pokretanje i zaustavljanje novih instanci iste, kontejnerizovane aplikacije
    - pokretanje izolovanih instanci na istom hardveru pa čak i u okviru istog OS-a
      - više okruženja na istom računaru
  - **bolju kontrolu verzija kôda**
    - kontrolom verzija kontejnera

7

7

Kontejnerizacija - operativni sistemi

- U ovom kursu se najviše bavimo kontejnerizacijom u okviru Linux OS-a
  - većina osnovnih teorijskih principa važi u opštem slučaju
    - i kontejnerizaciji u drugim OS-ovima
  - pojedina odstupanja moguća u okviru Windows OS-a
    - kada se posmatraju navedeni sistemski pozivi i način upravljanja osnovnim delovima Kernela
    - od 2016. godine umesto u virtualnoj mašini, moguća je direktna kontejnerizacija u okviru Windows i Mac OS-ova
      - https://www.docker.com/docker-news-and-press/docker-released-native-mac-and-windows-apps-optimize-developer-experience

8

8

Kontejnerizacija i DevOps

- **DevOps** je praktična primena principa razvoja softvera po kojim inženjeri razvoja softvera (engl. *development engineers*) i inženjeri operative (engl. *operations engineers*) zajedno učestvuju u razvoju softvera tokom celokupnog njegovog životnog ciklusa
  - od projektovanja softvera, preko razvoja pa do njegovog održavanja
- termin inženjer razvoja softvera
  - može se posmatrati kao širi skup od samo softverskih inženjera
    - uključuje sve osobe koji učestvuju u samom razvoju softvera
- termin inženjer operative
  - obuhvata sistemske administratore, administratore baza podataka, inženjere za sigurnost sistema, inženjere za računarske mreže itd.

9

9

Kontejnerizacija i DevOps

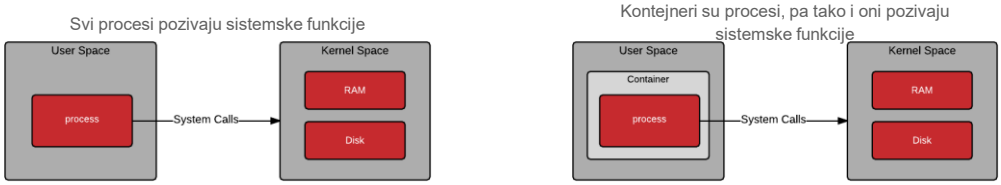
- DevOps promoviše ekosistem razvoja softvera
  - zasnovan na pravilima Agilne i Lean metodologije razvoja softvera
  - u kojem inženjeri razvoja i operative koriste iste alate i tehnike za rad sa softverom
- Infrastruktura kroz kôd (engl. ***Infrastructure as a Code,* IaaC**)
  - obuhvata automatizaciju postavljanja arhitekture na kojoj se izvršavaju programi
  - uspostavljanje arhitekture pomoću konfiguracionih datoteka
    - umesto rada sa hardverskim komponentama
  - **kontejneri** idealni kandidati zbog lake konfiguracije
- Kontinualno dostavljanje kôda (engl. ***Continuous Delivery***)
  - razvoj, testiranje, postavljanje izvršne verzije kôda na brz i automatski način
  - **kontejneri** idealni kandidati zbog mogućnosti automatizacije testiranja posredstvom eksternog softvera

10

10

Kontejneri - osnovni pojmovi

- Tri osnovna postulata:
  - sve aplikacije, uključujući i kontejnere, oslanjaju se na **Kernel** operativnog sistema
    - Kernel predstavlja jezgro operativnog sistema sa potpunom kontrolom nad svim delovima hardversko/softverske platforme
  - Kernel, preko **sistemskih poziva**, aplikacijama nudi API (engl. *Application Programming Interface*)
  - stabilan API je od značaja kontejnerima jer predstavlja osnovu za deterministično izvršavanje
    - dodavanje i ukidanje sistemskih poziva može imati veliki uticaj na kontejnere



izvor: *Architecting Containers, Scott McCarty* https://rhelblog.redhat.com/2015/07/29/architecting-containers-part-1-user-space-vs-kernel-space/

11

11

Kontejneri - osnovni pojmovi

- Prostor Kernela (engl. *Kernel Space*)
  - resursi i procedure Kernela
  - kojima se pristupa putem sistemskih poziva
  - služe za upravljanje hardverom
    - enkapsuliraju pristup hardveru pomoću sistemskih procedura
- Korisnički prostor (engl. *User Space*)
  - u opštem slučaju, obuhvata sav softver koji nije deo Kernela operativnog sistema
  - u terminologiji kontejnerizacije:
    - obuhvata datoteke i programe koji se nalaze i izvršavaju u okviru kontejnera
  - aplikacije mogu da pozivaju sistemske procedure koje se izvršavaju u prostoru Kernela
    - npr. zauzimanje memorije ili otvaranje datoteke
    - **deterministička komunikacija** između dva prostora je neophodna

12

12

Kontejneri - osnovni pojmovi

- Za svaki kontejner se kreira novi imenski prostor u prostoru Kernela
  - pozivom metode **`clone()`**
  - imenski prostori kernela (engl. *Kernel namespace*)
    - omogućavaju da svaki proces kreiran metodom `clone()` ima svoj *hostname*, IP adresu, ID procesa itd.
    - moguće je kreirati novi imenski prostor za svaki kontejner
    - zaustavljanjem kontejnera, brojač imenskih prostora se smanjuje
      - imenski prostori su često uklonjeni i memorija oslobođena

13

13

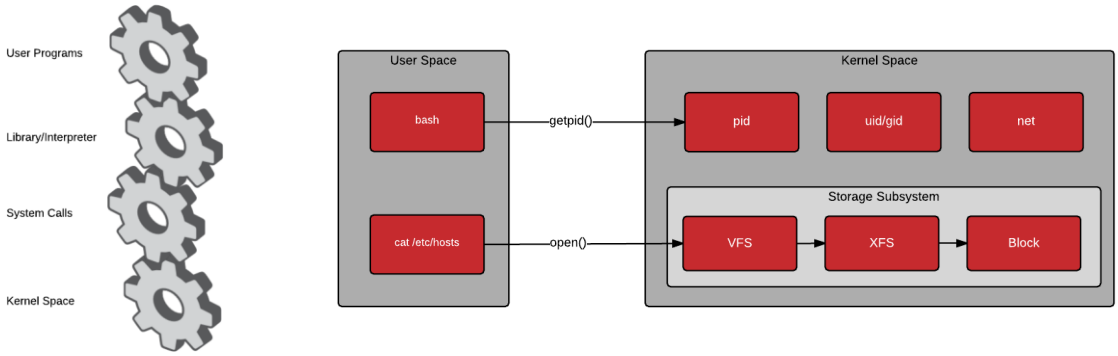
Kontejneri - osnovni pojmovi

- Kontrolne grupe Kernela (engl. *cgroups*)
  - omogućavaju upravljanje resursima (CPU, memorija, mreža, blokovski U/I itd.)
    - postavljanje ograničenja
    - postavljanje prioriteta
  - način kako da se izoluju resursi koje koriste kontejneri na nivou Kernela operativnog sistema

14

14

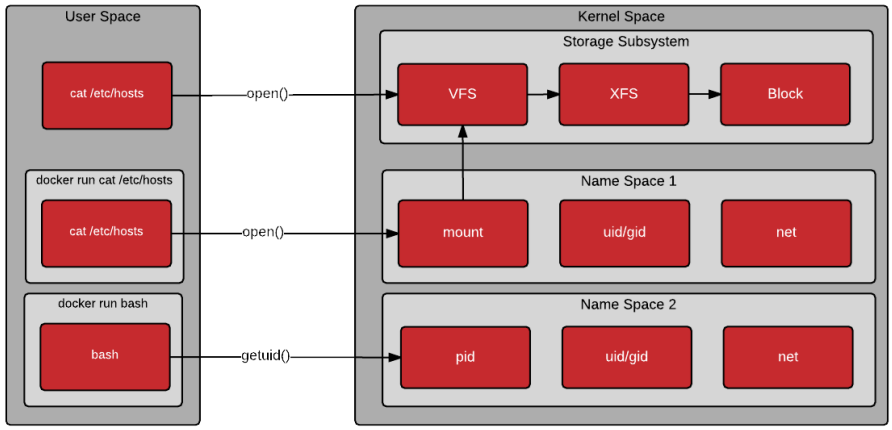
Kontejneri - osnovni pojmovi



izvor: *Architecting Containers, Scott McCarty* https://rhelblog.redhat.com/2015/07/29/architecting-containers-part-1-user-space-vs-kernel-space/

15

Kontejneri - osnovni pojmovi



izvor: *Architecting Containers, Scott McCarty* https://rhelblog.redhat.com/2015/09/17/architecting-containers-part-2-why-the-user-space-matters-2/

16

Kontejneri - osnovni pojmovi

- Aplikacije sadrže poslovnu logiku i oslanjaju se na sistemske pozive
  - kompajlirane aplikacije sadrže pozive sistemskih procedura ugrađene u njihov mašinski kod
  - kod interpretiranih jezika, sistemski pozivi su implementirani na nivou interpretera
    - npr. u slučaju programskog jezika Java, sistemske pozive poziva JVM
- Koristiti kontejnere ne znači zanemariti pojmove korisničkog prostora i prostora Kernela
  - oba prostora su prisutna i korisnici ih trebaju biti svesni
    - iako korisnici ne rade direktno sa sistemskim pozivima niti upravljaju direktno resursima operativnog sistema
  - na taj način će izbeći moguće probleme u prenosivosti kontejnera između različitih sistema
    - usled različitih sistemskih poziva
    - usled evolucije Kernela

17

17

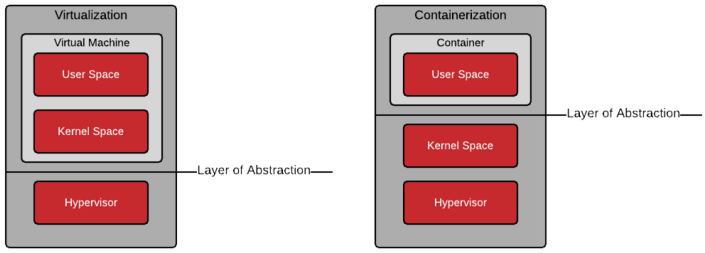
Kontejneri - osnovni pojmovi

- Kontejneri i virtualne mašine
  - ponekad kontejneri mogu predstavljati virtualizaciju operativnog sistema
    - i na taj način pogrešno biti poistovećeni sa virtualnim mašinama
  - **virtualna mašina** (engl. *virtual machine*)
    - predstavlja apstrakciju u odnosu na hardver
    - sastoji se iz dva dela (datoteke):
      - virtuelni disk sa korisničkim prostorom i prostorom Kernela
      - definicija meta-podataka sa specifikacijom resursa (CPU, RAM, grafika)
    - oslanja se na **hipervizore** (engl. *hypervisor*)
      - virtualizacija na nivou hardvera
  - **kontejner**
    - predstavlja apstrakciju u odnosu na operativni sistem
      - virtualizacija na nivou operativnog sistema
    - predstavlja zapakovan samo korisnički prostor u okviru kojeg se izvršava aplikacija

18

18

Kontejneri - osnovni pojmovi

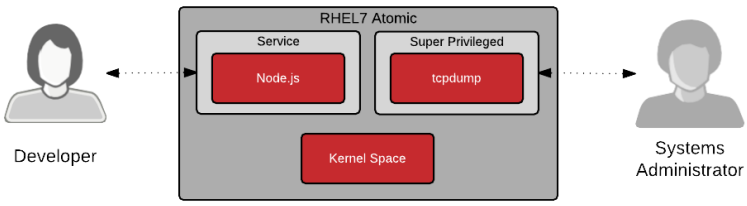


izvor: *Architecting Containers, Scott McCarty* https://rhelblog.redhat.com/2015/09/17/architecting-containers-part-2-why-the-user-space-matters-2/ 19

19

Kontejneri - osnovni pojmovi

- Privilegovani kontejner (engl. *Super Privileged Container*, SPC)
  - sadrži alate za nadgledanje, upravljanje i debug-ovanje drugih kontejnera
  - zahtevaju posebna prava nad sistemom na kojem se izvršavaju
    - *sudo* prava

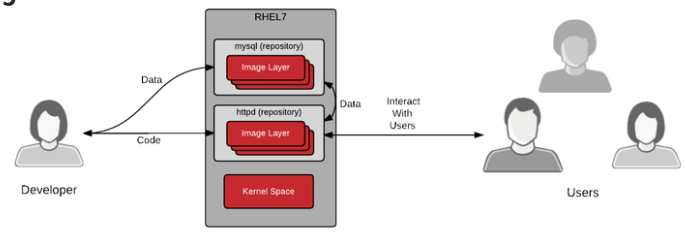


izvor: *Architecting Containers, Scott McCarty* https://rhelblog.redhat.com/2015/11/10/architecting-containers-part-3-how-the-user-space-affects-your-application/ 20

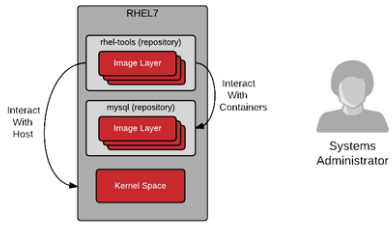
20

Kontejneri - osnovni pojmovi

- Primer aplikativnih kontejnera
  - kontejner 1: aplikativni server
  - kontejner 2: server baze podataka
  - komuniciraju međusobno
  - komuniciraju s korisnicima



- Primer privilegovanog kontejnera
  - SPC kontejner 1: *rhel-tools*
    - sa alatima za nadgledanje saobraćaja i korišćenja resursa
  - kontejner 2: server baze podataka



21

izvor: *Architecting Containers, Scott McCarty* https://rhelblog.redhat.com/2015/11/10/architecting-containers-part-3-how-the-user-space-affects-your-application/

21

Kontejneri - osnovni pojmovi

- Kontejner ima dva stanja:
  - kao i bilo koji, običan, Linux program ili proces
  - **neaktivno stanje**
  - **aktivno stanje**
- Neaktivno stanje
  - u ovom stanju kontejner je sačinjen od jedne ili više datoteka
    - skladištenih na disku
  - ovakav skup datoteka se naziva **slika kontejnera** ili **repozitorijum kontejnera**
    - razlika između ova dva pojma opisana je u nastavku
- Aktivno stanje
  - predstavlja aktivan proces u Linux-u
    - koji se naziva **kontejner**

22

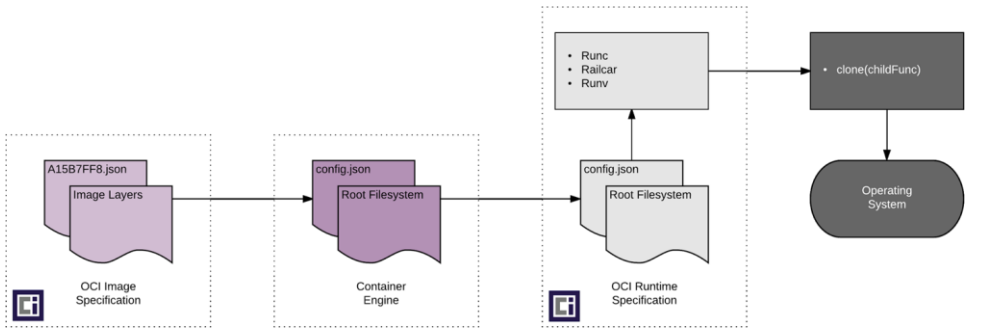
22

Kontejneri - osnovni pojmovi

- Pokretanje kontejnera
  - prebacivanje kontejnera iz neaktivnog u aktivno stanje
  - **obrađivač slike kontejnera** otpakuje sliku kontejnera, uključujući i meta-podatke
  - podaci se prosleđuju Kernelu operativnog sistema
  - vrlo slično pokretanju običnog Linux procesa
    - zahteva poziv ka API-ju Kernela
    - kreira se izolovan proces i kopija datoteka koje su bile deo slike kontejnera

23

Kontejneri - osnovni pojmovi



Slike kontejnera (neaktivno stanje kontejnera).

Obrađivač slike kontejnera otpakuje slike i meta-podatke u sistem datoteka operativnog sistema - domaćina.

Kontejner u izvršnom okruženju (aktivno stanje kontejnera). Izvršno okruženje preuzima datoteke i meta-podatke i šalje kernelu zahtev za kreiranjem kontejnera

Izvršna okruženja mogu biti kreirana za različite operativne sisteme.

izvor: *Understanding Container Standards, Scott McCarty* https://medium.com/cri-o/understanding-container-standards-1e1448cbb92c

24

Kontejneri - osnovni pojmovi

- Standardizacija
  - *Open Container Initiative* (OCI)
  - standardi za format slike kontejnera
    - omogućavaju interoperabilnost između različitih alata za rad sa kontejnerima
    - omogućavaju lakšu saradnju između razvojnih timova i lakši razvoj slika
    - **standard**: *Open Container Initiative* (OCI): *Container Image Format Specification*
      - definiše pravila čuvanja i format slike kontejnera na disku
      - definiše meta-podatke za opis slika kontejnera
  - standard za obrađivač slike
    - **standard**: OCI: *Container Runtime Specification i Reference Runtime Implementation* (runC)

25

25

Kontejneri - terminologija

- Terminologija (osnovna)
  - **Slika kontejnera** (engl. *Container Image*)
  - **Format slike kontejnera** (engl. *Container Image Format*)
  - **Obrađivač slike kontejnera** (engl. *Container Engine*)
  - **Kontejner** (engl. *Container*)
  - **Sistem domaćin** (engl. *Container Host*)
  - **Server registar** (engl. *Registry Server*)
  - **Orkestracija kontejnera** (engl. *Container Orchestration*)

26

26

Slika kontejnera (engl. *Container Image*)

- nepromenljiva, statička datoteka ili skup datoteka koje sadrže “uputstvo” za pokretanje izolovanog procesa na odgovarajućoj platformi
  - sadrži potrebne sistemske biblioteke, alate i podešavanja
    - koji definišu odgovore na pitanja:
      - Kako pokrenuti?, Šta pokrenuti? i Gde pokrenuti?
    - kako bi bilo omogućeno kreiranje kontejnera na nekoj od platformi za kontejnerizaciju
- svaka slika se (najčešće) sastoji od više **slojeva slike**
  - koji omogućavaju veliki stepen ponovne iskoristivosti komponenti
  - obično preuzete sa **servera registra**
  - stoga, termin slika kontejnera se najčešće odnosi na skup slojeva slike zajedno sa meta-podacima o tim slojevima

27

27

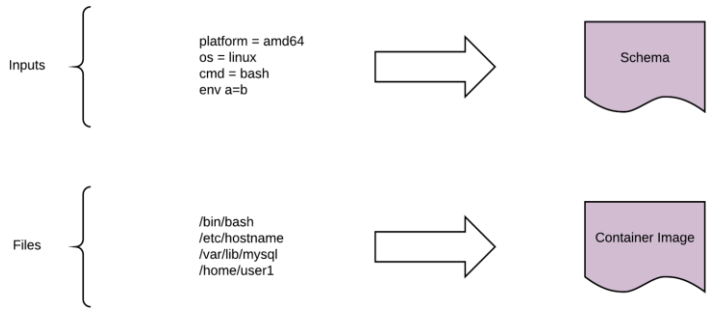
Format slike kontejnera (engl. *Container Image Format*)

- Istorijski gledano, svaki od izvršilaca kontejnera je imao svoj format slike
  - formati: Docker, Appc, LXD
  - neki su definisali postojanje jednog sloja slike
  - neki su definisali postojanje više slojeva slike organizovanih u strukturu tipa stabla
- Danas, skoro svi izvršioci kontejnera podržavaju format slike definisan OCI standardom
  - zasnovan na Docker v2 formatu slika
  - ovaj standard definiše da je svaka slika sastavljena od
    - slojeva (*tar* datoteka) i
    - meta-podataka (*manifest.json* datoteke)

28

28

Format slike kontejnera (engl. *Container Image Format*)

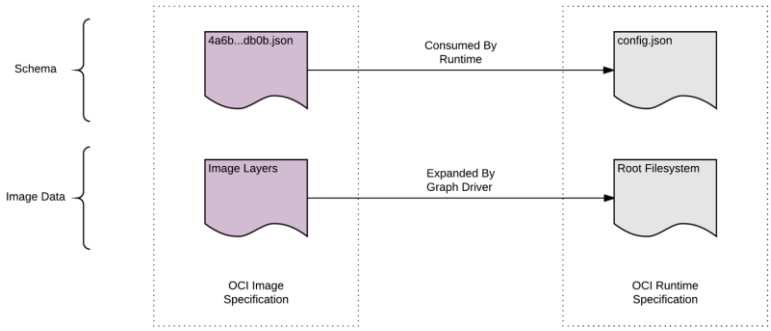


izvor: *Understanding Container Standards, Scott McCarty* https://medium.com/cri-o/understanding-container-standards-1e1448cbb92c

29

29

Format slike kontejnera (engl. *Container Image Format*)

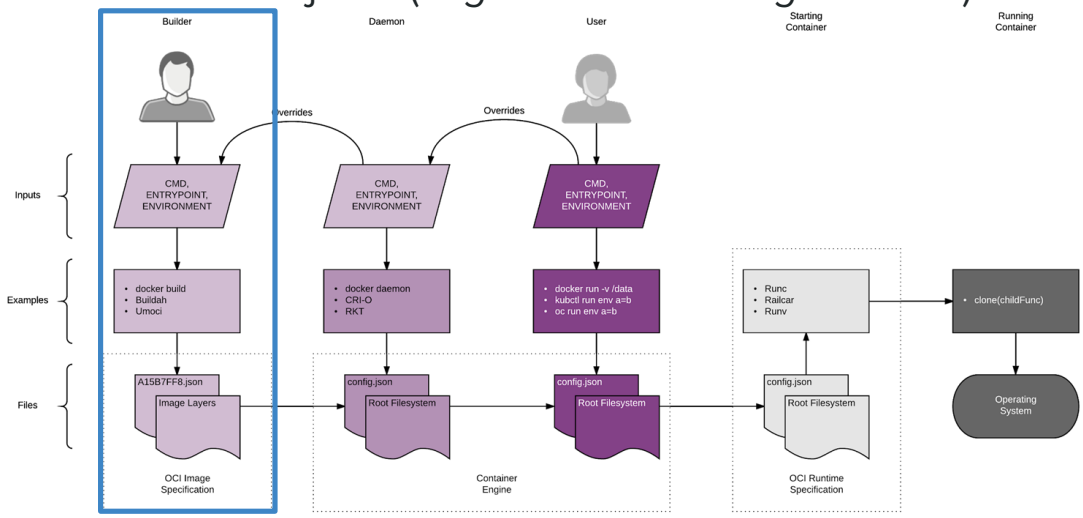


30

izvor: *Understanding Container Standards, Scott McCarty* https://medium.com/cri-o/understanding-container-standards-1e1448cbb92c

30

Format slike kontejnera (*engl. Container Image Format*)



izvor: *Understanding Container Standards, Scott McCarty* https://medium.com/cri-o/understanding-container-standards-1e1448cbb92c

31

31

Obrađivač slike kontejnera (engl. *Container Engine*)

- Softver zadužen za prihvatanje korisničkih uputstava i parametara, preuzimanje slika kontejnera i njihovo otpakivanje u sistemu datoteka **sistema domaćina**
  - ne izvršava kontejnere, već priprema kontejner za izvršenje u **izvršnom okruženju kontejnera**
  - postoji više obrađivača slika: Docker, RTK, CRI-O, LXD, Railcar, LXC
  - takođe, mnogi pružaoci usluga platforme kao servisa (engl. *Platform as a Service*, PaaS) imaju svoje ugrađene izvršioce
  - standardizacija igra ključnu ulogu u interoperabilnosti slika i različitih izvršnih okruženja
    - gde je spona između dve strane upravo izvršilac kontejnera
    - standard OCI

32

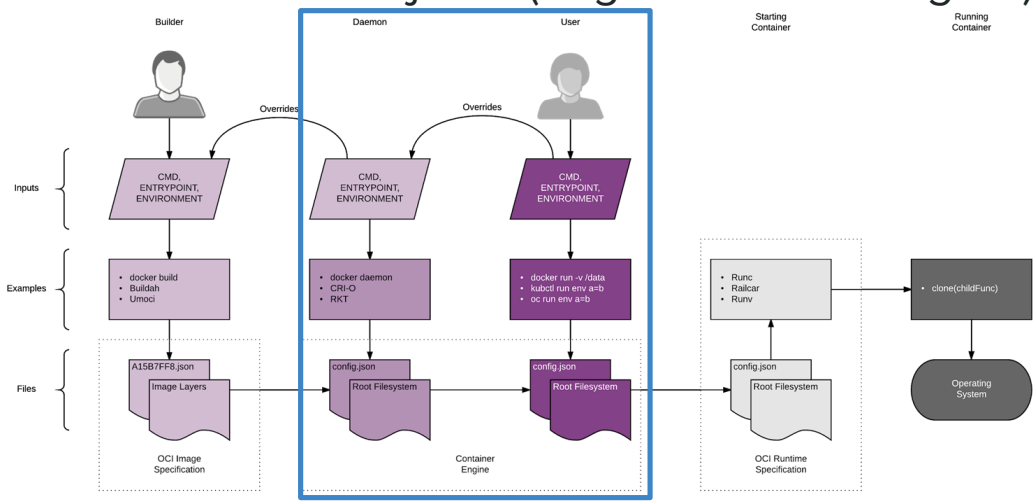
32

Obrađivač slike kontejnera (engl. *Container Engine*)

- Zadužen je za:
  - obradu korisničkih unosa
  - obradu ulaznih parametara dobijenih preko poziva API-ja
    - najčešće poslatih od strane orkestratora kontejnera
  - preuzimanje slika kontejnera sa servera registara
  - otpakivanje slika kontejnera posredstvom graf drajvera
  - priprema meta-podataka koji će biti prosleđeni izvršnom okruženju sa otpakovanom slikom
    - koriste se podrazumevane vrednosti metapodataka za odgovarajuću sliku kontejnera
    - koriste se vrednosti postavljene od strane korisnika
      - npr. CMD, ENTRYPOINT
  - pripremanje tačke uvezivanja (engl. *mount point*) kontejnera u okviru sistema datoteka operativnog sistema domaćina

33

Obrađivač slike kontejnera (engl. *Container Engine*)



izvor: *Understanding Container Standards, Scott McCarty* https://medium.com/cri-o/understanding-container-standards-1e1448cbb92c

34

Kontejner (engl. *Container*)

- Izolovani proces u okviru operativnog sistema domaćina koji predstavlja instancu slike kontejnera
  - pripremljenu od strane obrađivača slike kontejnera
- Na Linux-u
  - kreirani pozivima `clone()` sistemskog poziva
  - izolovani koristeći *cgroups*, *SELinux* ili *AppArmor*
    - mehanizme za izolaciju i upravljanje procesima

35

35

Sistem domaćin (engl. *Container Host*)

- Sistem u okviru kojeg se pokreće kontejnerizovani proces - kontejner
  - operativni sistemi,
  - virtualne mašine,
  - *cloud* platforme,
  - *bare metal* rešenja
- Sadrži lokalni *cache* slika kontejnera

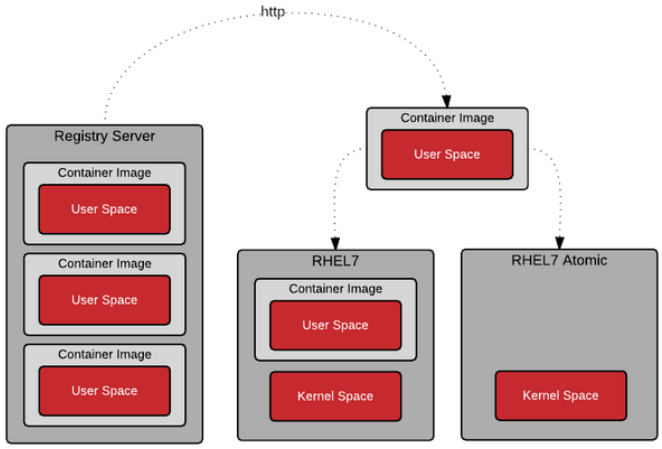
36

36

Server registar (engl. *Registry Server*)

- Server datoteka zadužen za skladištenje slika kontejnera
  - sa dodeljenim DNS-om
  - javni ili privatni
- Obrađivač slika kontejnera povlači slike i meta-podatke iz registra
  - ukoliko nema lokalni *cache* slike
  - ukoliko je lokalni *cache* zastareo
  - povlače se samo izmenjeni slojevi slike
  - postoje podrazumevani serveri za različite obrađivače slika kontejnera
    - moguće ručno dodati nove servere u okviru sistema domaćina
- Potrebno poverenje u server registar
  - potencijalna tačka napada postavljanjem malicioznih slika
  - potencijalni legalni problemi ako slika sadrzi nelegalan softver

Server registar (engl. *Registry Server*)



izvor: *A Practical Introduction to Container Terminology, Scott McCarty* https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/

Orkestracija kontejnera (engl. *Container Orchestration*)

- Obuhvata povezivanje više kontejnera u koherentnu celinu
  - praveći na taj način aplikaciju koja je sastavljena od više servisa
    - umesto jedne monolitne aplikacije
    - npr. mikroservisne arhitekture
  - primeri orkestratora: Docker Swarm, Apache Mesos, Kubernetes
    - Kubernetes je danas *de facto* standard
      - razvijen u Google-u
      - podržan od strane Docker-a i Mesosphere-a
- Orkestrator kontejnera ima dva zadatka:
  - omogućava specifikaciju parametara i veza između kontejnera
    - kreirajući tako aplikaciju koja se sastoji od više kontejnera a čije je izvršenje podešeno specificiranim parametrima
  - dinamičku dodelu zadataka kontejnerima u okviru klastera
    - distribuirana obrada podataka i izračunavanje

39

39

Orkestracija kontejnera (engl. *Container Orchestration*)

- Orkestrator kontejnera omogućava:
  - nezavisno dodeljivanje zadataka kontejnerima koji čine jednu aplikaciju
    - kreiranje velikih klastera sistema domaćina
    - otpornost na greške i bolja skalabilnost
  - lako postavljanje kontejnerizovane aplikacije u novim okruženjima
    - identična slika sa prilagođenim parametrima
      - postavlja se na razvojni računar, test server, produkcioni server
      - potencijalno na različite sisteme domaćine

40

40

Kontejneri - terminologija

- Terminologija (napredna)
  - **Izvršno okruženje kontejnera** (engl. *Container Runtime*)
  - **Sloj slike kontejnera** (engl. *Image Layer*)
  - **Oznaka sloja slike kontejnera** (engl. *Tag*)
  - **Repozitorijum** (engl. *Repository*)
  - **Prostor imenovanja** (engl. *Namespace*)
  - **Prostor imenovanja Kernel-a** (engl. *Kernel Namespace*)
  - **Graf drajveri** (engl. *Graph Driver*)

41

41

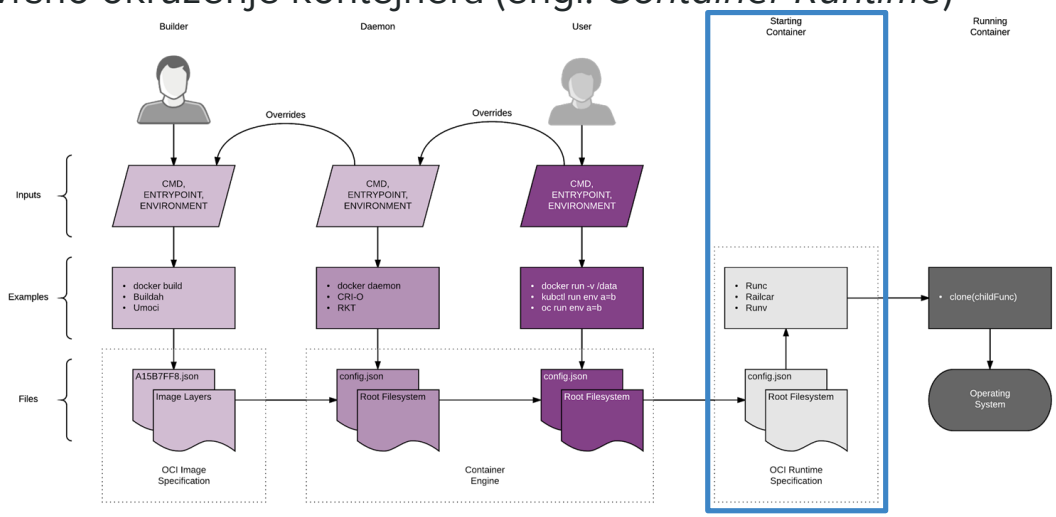
Izvršno okruženje kontejnera (engl. *Container Runtime*)

- Softverska komponenta zadužena za:
  - pokretanje kontejnera na osnovu elemenata (datoteka i meta-podataka) pripremljenih od strane obrađivača slika kontejnera
  - komunikaciju sa Kernel-om
  - izolovanje kontejnera od ostalih procesa operativnog sistema
- Najčešće se koristi **runc** izvršno okruženje kontejnera
  - koriste ga Docker, CRI-O i mnogi drugi
  - nastao na osnovu Docker-ove *libcontainer* biblioteke
  - obuhvaćen je OCI standardom

42

42

Izvršno okruženje kontejnera (engl. *Container Runtime*)



izvor: *Understanding Container Standards, Scott McCarty* https://medium.com/cri-o/understanding-container-standards-1e1448cbb92c

Sloj slike kontejnera (engl. *Image Layer*)

- Slike kontejnera se uobičajeno sastoje od slojeva
  - slojevi se međusobno nalaze u odnosu roditelj-dete
  - svaki sloj predstavlja načinjene izmene u odnosu na roditeljski sloj
  - svaki sloj ima globalno jedinstveni identifikator (engl. *Universally Unique Identifier*, UUID) i oznaku
- Moguće je kreirati kontejner od izolovanog sloja slike
  - uključeni su svi roditeljski slojevi, tranzitivno
  - ne garantuje se izvršivost
- Pravljenje nove slike
  - čuva se razlika u odnosu na prethodno kreiranu sliku
    - snimljena kao novi sloj slike

Oznaka sloja slike kontejnera (engl. *Tag*)

- Alfa-numerička oznaka koju je moguće dodeliti bilo kojem sloju slike kontejnera
  - koju koriste kreatori slike kontejnera
    - da označe one slojeve od kojih je moguće kreirati kontejner
  - uvek postoji rezervisana oznaka *latest*
    - pokazuje na poslednju verziju kontejnera
    - koja se podrazumeva, ako se prilikom kreiranja kontejnera od slike ne navede oznaka sloja koji se preuzima iz registra
  - obično se koriste da označe različite verzije slike
    - koje se obično razlikuju u poslednjih nekoliko slojeva
  - ne postoji ograničenje koji slojevi se mogu označiti
    - **koristiti oznake pažljivo!**

45

45

Repozitorijum (engl. *Repository*) i Prostor imenovanja (engl. *Namespace*)

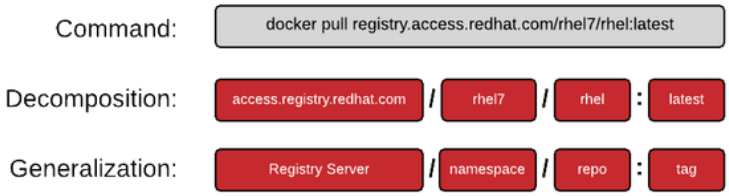
- Repozitorijum predstavlja skup svih slojeva koji mogu sačinjavati slike kontejnera
  - **nadskup pojma slike** jer može da sadrži različite verzije slike
    - tako što sadrži sve slojeve od kojih je moguće izgraditi sliku
  - nalazi se na **serverima registrima**
- Prostor imenovanja predstavlja sredstvo razdvajanja grupa repozitorijuma na serveru registru
  - dozvoljava postojanje istoimenih repozitorijuma i oznaka u različitim prostorima imenovanja

46

46

Repozitorijum (engl. *Repository*) i Prostor imenovanja (engl. *Namespace*)

- Prilikom preuzimanja slike kontejnera
  - navodi se naziv repozitorijuma a ne oznaka slike
  - oznaka sloja iz repozitorijuma od kojeg se pravi kontejner
    - uključujući i njegove roditeljske slojeve
  - **REGISTRY/NAMESPACE/REPOSITORY[:TAG]**
  - pretražuju se svi podešeni serveri registri

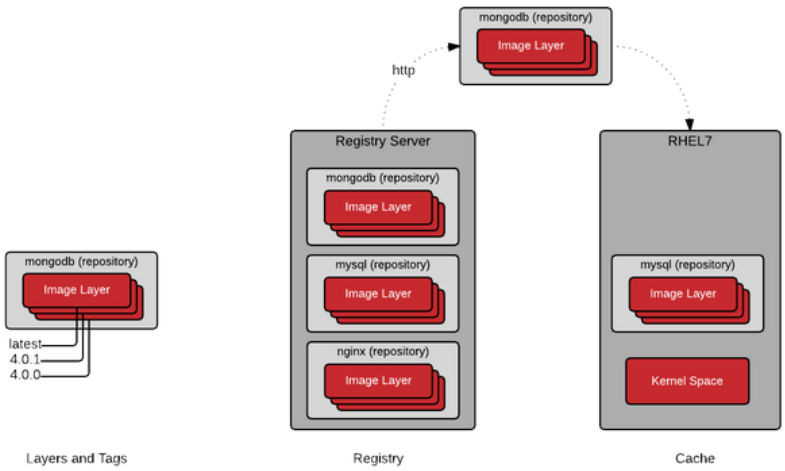


izvor: *A Practical Introduction to Container Terminology, Scott McCarty* https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/

47

47

Repozitorijum (engl. *Repository*) i Prostor imenovanja (engl. *Namespace*)



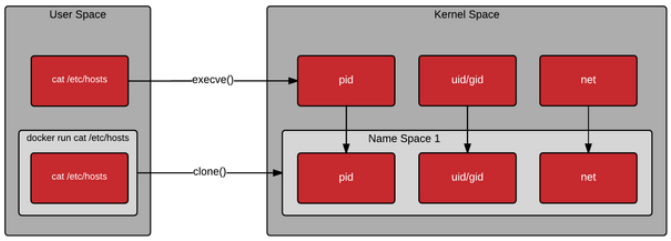
izvor: *A Practical Introduction to Container Terminology, Scott McCarty* https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/

48

48

Prostor imenovanja Kernel-a (engl. *Kernel Namespace*)

- Predstavljaju sredstvo logičkog odvajanja kontejnera unutar Kernela sistema domaćina
  - omogućavaju da svaki kontejner poseduje svoje oznake procesa, oznake korisnika, mrežne interfejse itd.
- Omogućili su kontejnere kakvim ih znamo danas



izvor: *A Practical Introduction to Container Terminology, Scott McCarty* https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/

Graf drajveri (engl. *Graph Driver*)

- Softver koji omogućava raspakivanje potrebnih slojeva slike u strukturu u lokalnoj memoriji sistema domaćina radi pokretanja kontejnera
  - zahteva specifikaciju oznake sloja od kojeg se pravi kontejner
    - podrazumevano *latest*
  - graf drajver otpakuje sve slojeve slike neophodne da bi se raspakovao i označeni sloj
    - ponovo kreira kompletnu predstavu procesa/kontejnera u lokalnoj memoriji
  - moguće je raspakovati slojeve na više vrsta lokacija:
    - direktorijum u sistemu datoteka (npr. *Overlay2* drajver)
    - blokovsko skladište (npr. *Device Mapper* drajver)

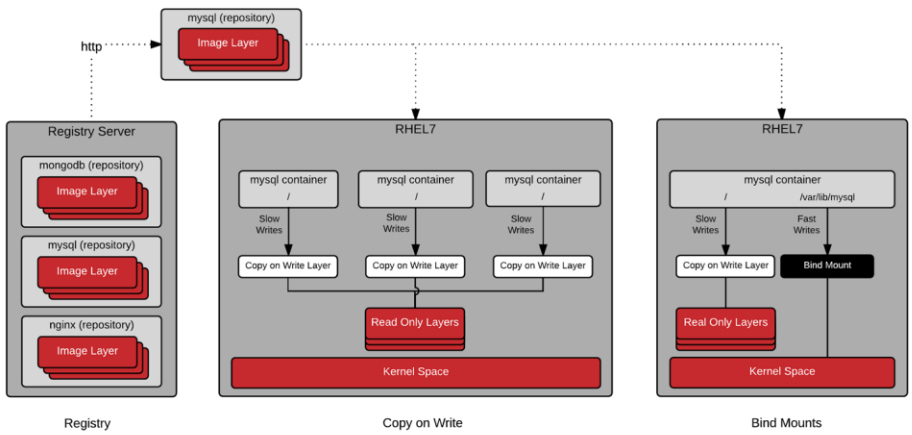
Graf drajveri (engl. *Graph Driver*)

- Prilikom startovanja kontejnera
  - svi slojevi su postavljeni u odgovarajući prostor imenovanja Kernel-a
    - u režimu čitanja
  - otvara se novi, poseban sloj za svaki podignuti kontejner
    - engl. ***copy-on-write layer***
    - u režimu pisanja i čitanja
    - kako bi omogućio upis podataka u kontejneru
    - moguće pokrenuti kontejner i samo u režimu čitanja
      - u tom slučaju je ovaj sloj onemogućen

51

51

Graf drajveri (engl. *Graph Driver*)



izvor: *A Practical Introduction to Container Terminology, Scott McCarty* https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/

52

52

Slučajevi korišćenja kontejnera

- Kontejneri sa aplikacijama (engl. *Application containers*)
  - sadrže aplikacije zajedno sa potrebnim servisima, serverima i bazama podataka
  - obično ne zahtevaju privilegije u sistemu domaćinu da bi bili izvršavani
    - kontejneri u produkciji mogu zahtevati viša prava pristupa
      - kako bi omogućila integraciju sa drugim servisima
- Kontejneri sa operativnim sistemom (engl. *Operating system containers*)
  - obično sadrže ceo operativni sistem zapakovan i pokrenut u obliku kontejnera
  - liče na virtualnu mašinu po svojoj svrsi
    - moguće pokrenuti procese u njima

53

53

Slučajevi korišćenja kontejnera

- Kontejneri sa jednom instaliranom aplikacijom (engl. *Pet containers*)
  - aplikacija kod koje nema potrebe za velikim brojem instaliranja
    - obično se nudi kao SaaS
  - kreiraju se slike kako bi omogućili lakšu prenosivost
    - koristeći postojeće instalacione fajlove i skripte za automatizaciju
- Privilegovani kontejneri (engl. *Super privileged containers*)
  - koriste se u slučajevima kada je potrebno komunicirati sa drugim procesima i kontejnerima u okviru sistema domaćina
    - monitoring, pravljenje kopija, *debug* itd.
  - potencijalno opasni kontejneri za sistem domaćina
    - jer zahtevaju veći stepen povezanosti sa Kernel-om

54

54

Slučajevi korišćenja kontejnera

- Kontejneri sa alatima (engl. *Tools & Operating System Software*)
  - olakšavaju distribuciju softvera
    - između razvojnih timova
    - između testnog i produkcionog okruženja
  - smanjuju opterećenje nad sistemom domaćinom
    - izdvajanjem nepotrebnih aplikacija i alata u kontejnere
      - i njihovim pokretanjem po potrebi
        - npr. samo tokom razvoja softvera

55

55

Vrste slika kontejnera - šabloni

- Slike softverskih rešenja (engl. *Application Images*)
  - sadrže aplikacije namenjene krajnjim korisnicima
  - primeri obuhvataju baze podataka, aplikativne servere i komunikacione magistrale
  - najčešće kreirane od strane proizvođača softvera koji se distribuira
  - najlakše za korišćenje ali najteže za projektovanje, kreiranje i održavanje
- Osnovne slike (engl. *Base Images*)
  - najjednostavnija vrsta slika
  - sastavljene od jednog sloja slike koji ne poseduje roditeljski sloj
  - uobičajeno, ove slike predstavljaju slike operativnih sistema
    - samo sa alatima za instaliranje paketa
  - eksplicitno se nasleđuju od strane drugih slika koji vrše nadogradnju
  - moguće je ručno napraviti sliku koja će biti korišćena kao osnovna slika
    - osnovne slike najčešće prave organizacije i proizvođači koji izdaju operativne sisteme (Debian, Fedora, CentOS, itd.)

56

56

Vrste slika kontejnera - šabloni

- Parametrizovane razvojne slike (engl. *Builder Images*)
  - slike koje sadrže većinu alata i biblioteka potrebnih za kreiranje upotrebljive slike softverskog rešenja
    - ne sadrže kôd konkretne aplikacije
  - kôd aplikacije se prosleđuje kao parametar već pokrenutom kontejneru sa parametrizovanom razvojnom slikom
    - kontejner pripremi kôd za izvršenje
      - npr. dobavljanje neophodnih biblioteka i kompajliranje
    - rezultat je nova slika softverskog rešenja od koje je moguće kreirati kontejner
    - npr. *source-to-image* https://github.com/openshift/source-to-image

57

57

Vrste slika kontejnera - šabloni

- Slike sa komponentama sistema (engl. *Containerized Components*)
  - kontejneri koji su namenjeni pokretanju zajedno sa drugim kontejnerima
    - obično se ne pokreću samostalno, mada nije zabranjeno
    - imaju veću upotrebnu vrednost kao deo celine nego samostalno
  - upotreba mikroservisnim arhitekturama
  - upotreba u slučaju kada nije moguće kontejnerizovati sve komponente aplikacije
- Slike za upravljanje kontejnerima (engl. *Deployer Images*)
  - slike koje, kada se pokrenu kao kontejneri, služe za upravljanje drugim kontejnerima
    - npr. u slučajevima kada je potrebno pokrenuti kontejnere u određenom redosledu

58

58

Vrste slika kontejnera - šabloni

- Posredničke slike (engl. *Intermediate Images*)
  - bilo koja slika nastala od osnovne slike a koja se zatim koristi za pravljenje novih slika
    - koristi se umesto osnovne slike
  - obuhvata dodatne slojeve u odnosu na osnovnu sliku
- Slike sa sistemskim procesima (engl. *System Containers*)
  - sadrže sistemski softver zapakovan u obliku slike kontejnera
    - ovakve slike se obično pokreću u obliku privilegovanih kontejnera
- Hibridne slike (engl. *Intermodal Container Images*)
  - slike za koje se može reći da pripadaju nekoliko prethodno navedenih vrsta
  - npr. slika sa *Apache* aplikativnim serverom i radnim okvirom *Ruby on Rails*
    - ovakva slika se može posmatrati i kao slika softverskog rešenja i kao parametrizovana razvojna slika

59

59

Docker

60

Docker - Arhiterktura

- Docker je platforma za razvoj, distribuciju i izvršavanje aplikacija
  - napisana u programskom jeziku GO
  - koja omogućava kontejnerizaciju aplikacija
  - i na taj način omogućavajući bržu isporuku softvera
- Docker omogućava “pakovanje” i izvršavanje aplikacija u vidu kontejnera
  - koji su izolovani od ostatka OS-a na kojem se izvršavaju
- Licenca
  - licenca otvorenog koda Apache 2.0

61

61

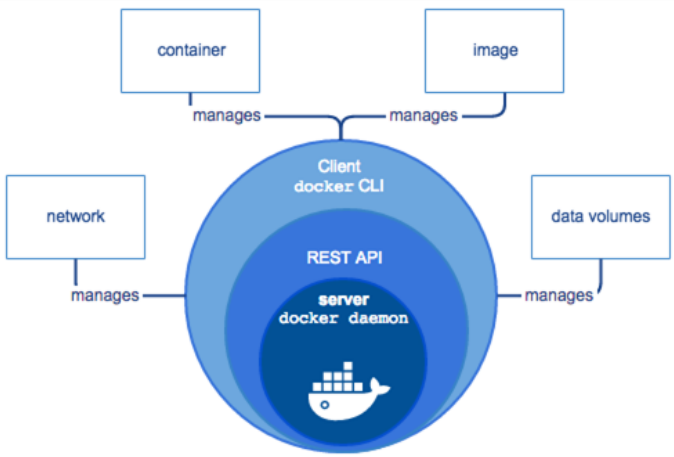
Docker - Arhiterktura

- Docker izvršno okruženje (engl. *Docker Engine*)
  - predstavlja aplikaciju koja prati klijent-server arhitekturu koja se sastoji od sledećih komponenti:
    - **docker pozadinski proces** (engl. *docker daemon*) - pozadinski proces operativnog sistema domaćina koji sadrži sve potrebne metode za rad sa kontejnerima, slikama, mrežama i diskovima
      - predstavlja server u ovoj arhitekturi
      - pristupa mu se direktno pomoću naredbe `dockerd`
    - **REST API** - koji predstavlja specifikaciju interfejsa sa metodama koje koriste docker klijenti kako bi komunicirali sa docker pozadinskim procesom
    - **docker klijent** - Program koji se izvršava kroz terminal operativnog sistema. Pristupa mu se direktno pomoću naredbe `docker`

62

62

Docker - Arhiterktura



izvor: *Docker Overview, docker docs* https://docs.docker.com/

63

63

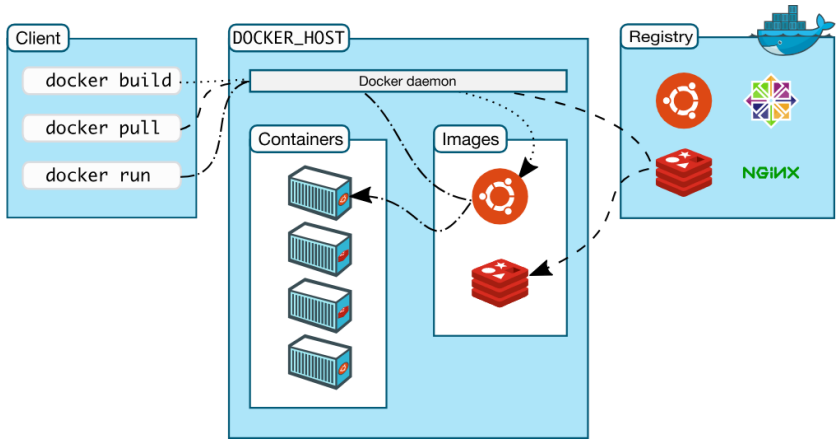
Docker - Arhiterktura

- Docker registar
  - skladište slika kontejnera koje se mogu koristiti uz docker
  - javni registar
    - Docker Hub https://hub.docker.com/explore/
      - podrazumevani registar za Docker
      - Docker će bez dodatnog podešavanja pokušati da u ovom registru pronađe sve slike
  - privatni registri
    - mogu se postaviti u okviru Docker Cloud servisa
      - https://cloud.docker.com/
    - mogu se postaviti samostalno na bilo koji server dostupan preko mreže
      - Npr. kompanijski Artifactory registar
  - prodavnica slika kontejnera
    - Docker Store https://store.docker.com/

64

64

Docker - Arhiterktura



izvor: *Docker Overview, docker docs* https://docs.docker.com/

65

65

Docker - Arhiterktura

- Imenski prostori (**namespaces**)
  - za svaki kreirani Docker kontejner kreira se skup imenskih prostora
    - u okviru prostora Kernel-a Linux OS-a
  - imenski prostor **pid** - izolacija procesa (PID: *Process ID*).
  - imenski prostor **net** - upravljanje mrežnim interfejsima (NET: *Networking*).
  - imenski prostor **ipc** - upravljanje međuprostornom komunikacijom (IPC: *InterProcess Communication*).
  - imenski prostor **mnt** - upravljanje sistemom datoteka (MNT: *Mount*).
  - imenski prostor **uts** - izolacija na nivou Kernel-a. (UTS: *Unix Timesharing System*).
- Kontrolne grupe (**cgroups**)
  - ograničavaju aplikacije na određen skup resursa
  - omogućavaju kontejnerima da dele isti hardver
  - omogućava uspostavljanje ograničenja nad resursima

66

66

Docker - Arhiterktura

- UnionFS
  - servis u Linux OS-u za priključivanje novih sistema datoteka:
    - npr. AUFS, btrfs, vfs, and DeviceMapper
  - omogućava kreiranje slika i kontejnera od više slojeva
    - brzo i efikasno
- Izvršenje kontejnera
  - **libcontainer** biblioteka

67

67

Docker pozadinski proces (dockerd)

- Pokretanje pozadinskog procesa
  - uobičajeno se pokreće automatski prilikom podizanja OS-a
  - uz korišćenje programa samog OS-a, ako je docker instaliran kao servis
    - `sudo systemctl start docker`
    - `sudo service docker start`
  - ručno
    - `dockerd`
- Moguće konfigurisati pozadinski proces
  - https://docs.docker.com/engine/reference/commandline/dockerd/

```
{
  "debug": true,
  "tls": true,
  "tlscert": "/var/docker/server.pem",
  "tlskey": "/var/docker/serverkey.pem",
  "hosts": ["tcp://192.168.59.3:2376"]
}
```

68

68

Docker klijent [run] - *Hello World*

- *Hello world* primer
  - ukoliko slika nije pronađena lokalno, docker je preuzima iz podrazumevanog registra
    - podrazumevano DockerHub

```
$ docker run hello-world

Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
9db2ca6ccae0: Pull complete
Digest: sha256:4b8ff392a12ed9ea17784bd3c9a8b1fa3299cac44aca35a85c90c5e3c7afacdc
Status: Downloaded newer image for hello-world:latest

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

69

69

Docker klijent [run] - *Hello World*

- *Hello world* primer
  - ukoliko je slika pronađena lokalno, Docker koristi tu sliku

```
$ docker run hello-world

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

70

70

Docker klijent [pull] - *Busybox*

- *Busybox*
  - skup alata koji se mogu koristiti iz komandne linije POSIX okruženja
    - https://en.wikipedia.org/wiki/BusyBox
  - napravljeni za ugrađene (engl. *embedded*) Linux sisteme
- Preuzimanje slike iz registra (Docker Hub)
  - naredba **`docker pull [OPTIONS] NAME[:TAG|@DIGEST]`**
  - opciono, iza naziva slike može se navesti i verzija koju preuzimamo
    - razdvojena simbolom ":"

```
$ docker pull busybox:latest

latest: Pulling from library/busybox
75a0e65efd51: Pull complete
Digest: sha256:d21b79794850b4b15d8d332b451d95351d14c951542942a816eea69c9e04b240
Status: Downloaded newer image for busybox:latest
```

71

Docker klijent [images] - *Busybox*

- Prikazivanje preuzetih docker slika
  - naredba **`docker images [OPTIONS] [REPOSITORY[:TAG]]`**
  - moguće prikazati i ID slike
    - navođenjem opcije `-a`

```
$ docker images
REPOSITORY                  TAG                 IMAGE ID            CREATED             SIZE
busybox                     latest              22c2dd5ee85d        10 days ago         1.16MB
hello-world                 latest              2cb0d9787c4d        2 weeks ago         1.85kB
...
```

72

Docker klijent [run] - *Busybox*

- Pokretanje kontejnera na osnovu preuzete slike
  - naredbom **`docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]`**
- kontejner je pokrenut, izvršio je praznu naredbu i zaustavio se
  - pošto je *Busybox* skup CLI alata, zahteva prosleđivanje naredbe koja treba biti izvršena
- Moguće proslediti Linux naredbu podržanu od strane aplikacije
  - nakon naredbe za pokretanje kontejnera
  - npr. `echo`

```
$ docker run busybox


$ docker run busybox echo "hello from busybox"
hello from busybox
```

73

73

Docker klijent [ps] - *Busybox*

- Ispisivanje kontejnera
  - naredba **`docker ps [OPTIONS]`**
  - opcija `-a` prikazuje sve kontejnere
    - pokrenute i nepokrenute
- Svako pokretanje slike kreira novi kontejner

```
$ docker ps
CONTAINER ID   IMAGE    COMMAND                  CREATED        STATUS                     PORTS     NAMES

$ docker ps -a
CONTAINER ID   IMAGE    COMMAND                  CREATED        STATUS                     PORTS     NAMES
69bc53929b3c   busybox  "echo 'hello from ..."   3 minutes ago  Exited (0) 2 minutes ago             zen_ritchie
f4d727e4b99e   busybox  "sh"                     6 minutes ago  Exited (0) 6 minutes ago             vigilant_thompson
```

74

74

Docker klijent [run] - *Busybox*

- Pokretanje kontejnera u interaktivnom režimu sa otvorenim terminalom
  - naredba docker `run -it`
  - opcija -i spaja se na STDIN
  - opcija -t alocira pseudo terminal

```
$ docker run -it busybox sh
/ # ls
bin   dev   etc   home  proc  root  sys   tmp   usr   var
/ # uptime
 14:24:15 up 1 day,  4:36,  0 users,  load average: 0.58, 0.72, 0.95
/ # exit
```

75

75

Docker klijent [rm]

- Brisanje kontejnera
  - naredba **`docker rm [OPTIONS] CONTAINER [CONTAINER...]`**
  - briše kontejnere čiji su ID-jevi prosleđeni

```
$ docker ps -a
CONTAINER ID    IMAGE         COMMAND    CREATED          STATUS               PORTS NAMES
d82779ec77f9    hello-world   "/hello"   4 seconds ago    Exited (0) 1 second ago    brave_dijkstra
...

$ docker rm d82779ec77f9
d82779ec77f9
```

76

76

Docker klijent [rm]

- Brisanje kontejnera
  - moguće je i kombinovati naredbe
    - nije karakteristično samo za docker `rm` već i za druge naredbe
  - opcija `-q` vraća numeričke ID-jeve
  - opcija `-f` filtrira rezultat prema zadatom uslovu

```
$ docker rm $(docker ps -a -q -f status=exited)
b70668e66275
69bc53929b3c
f4d727e4b99e
```

77

77

Docker klijent [container prune]

- Brisanje nekorišćenih kontejnera
  - naredba **`docker container prune`**
  - briše sve zaustavljene kontejnere
    - ali ne i povezane elemente kao što su skladišta ili mrežne konfiguracije

```
$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
14f4d283cdd1006f07b4fsdf1ad1ea75b826b927cc508a3f22a0bb3934625f69
bf1120889cc03b2854bb7ef4a1t89th6adf31fb35b1c292e7dd3f6a50a669805
...
```

78

78

Docker klijent [rmi]

- Brisanje slika kontejnera
  - naredba **`docker rmi [OPTIONS] IMAGE [IMAGE...]`**
  - briše slike čiji su ID-jevi prosleđeni

```
$ docker images -a
REPOSITORY                TAG                 IMAGE ID            CREATED             SIZE
hello-world               latest              2cb0d9787c4d        2 weeks ago         1.85kB
…

$ docker rmi 2cb0d9787c4d
Untagged: hello-world:latest
Untagged: hello-world@sha256:4b8ff392a12ed9ea17784bd3c9a8b1fa3299cac44aca35a85c90c5e3c7afacdc
Deleted: sha256:2cb0d9787c4dd17ef9eb03e512923bc4db10add190d3f84af63b744e353a9b34
Deleted: sha256:ee83fc5847cb872324b8a1f5dbfd754255367f4280122b4e2d5aee17818e31f5
```


Docker klijent [image prune]

- Brisanje nekorišćenih slika kontejnera i slojeva
  - naredba **`docker image prune`**
  - briše sve *dangling* slojeve slika
    - slojeve koji više nisu vezani za konkretne slike kontejnera
  - moguće obrisati i sve slike koje se ne koriste

```
$ docker image prune
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] y

Deleted Images:
untagged: hostname/image@sha256:877ccf9171518a9deaaabf3c9d4f5bc4dd57d19b37080dd7bf5c0fd4acd98484
deleted: sha256:71b2e074cf93e41cd15828965asddd899b54a45b5713c34a20904d4e356d0335
deleted: sha256:adf5ce285a3929080f17ca8dfa33fee2cd97e415b91548aa089655bd5c55029c
deleted: sha256:654c896ec44c926792e85097c39bda213a9faba3dc5a28487c7fcd6dc5022ac2
deleted: sha256:3d062c5001645ae4eb853a525409d6e2c1f12s321bf5f2d13cf82d69b1341d9a
...
```

Docker klijent [run] - web app

- Pokretanje slike kontejnera u pozadini
  - opcija -d specificira da se kontejner ne kači na STDOUT
    - dobra opcija za kontejnere u kojima se proces neprekidno izvršava
    - npr. web servere i servere baze podataka
  - nakon pokretanja proveriti http://localhost:32769/
- Dodatne opcije
  - opcija -P dodeljuje nasumične portove iz intervala 32768-61000 na exportovane portove kontejnera
  - moguće i dodeliti konkretne portove opcijom -p
  - opcija `--name` dodeljuje ime kontejneru

```
$ docker run -d -P --name static-site prakhar1989/static-site
0988c7f5c91a893210fb788a4d0183559dba41bbf2a2ab7e95b9f7b86d72b77f

$ docker port static-site
80/tcp -> 0.0.0.0:32769
443/tcp -> 0.0.0.0:32768
```

81

81

Docker klijent [stats] - web app

- Provera zauzeća resursa kontejnera
  - naredba **`docker stats [OPTIONS] [CONTAINER...]`**
  - opcija `--no-stream` vraća samo prvi rezultat

```
$ docker stats --no-stream static-site
CONTAINER    CPU %     MEM USAGE / LIMIT     MEM %       NET I/O           BLOCK I/O    PIDS
static-site  0.00%     2.145MiB / 13.69GiB   0.02%       8.03kB / 0B       0B / 0B      2
```

82

82

Docker klijent [inspect] - web app

- Uvid u detalje pokrenutog kontejnera
  - naredba **`docker inspect [OPTIONS] NAME|ID [NAME|ID...]`**

```
$ docker inspect static-site
[
    {
        "Id": "e101b096fdc7ee46aff9d30598f4ca4ada4a63ab4b692f0b1b0ac9799d429b41",
        "Created": "2018-07-28T13:27:08.965455832Z",
        "Path": "./wrapper.sh",
        "Args": [],
        "State": {
            "Status": "exited",
            "Running": false,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 0,
...
```

83

Docker klijent [exec] - web app

- Izvršenje proizvoljne naredbe u pokrenutom kontejneru
  - naredba **`docker exec [OPTIONS] CONTAINER COMMAND [ARG...]`**
  - vrlo često je potrebno izvršiti samo jednu naredbu u kontejneru
    - nema potrebe držati otvorenu sesiju i raditi u terminalu kontejnera
  - primer dat na ovom slajdu
    - pokretanje i kačenje na terminal u kontejneru
    - naredba koja se izvršava u kontejneru je `bash`

```
$ docker exec -it static-site  bash

root@4e0fca77c77a:/# ls
bin  boot  dev            etc  home  lib           lib64  media  mnt  opt proc  root  run  sbin  srv  sys  tmp  usr
var          wrapper.sh

root@4e0fca77c77a:/# exit
exit
```

84

Docker klijent [attach] - web app

- Povezivanje na kontejner koji je pokrenut u interaktivnom režimu
  - kontejner pokrenut naredbom **`docker run -it`**
  - zatvorena komunikacija sa Ctrl + P + Q
  - moguće je ponovo pokrenuti interaktivni režim
    - naredba **`docker attach [OPTIONS] CONTAINER`**

```
$ docker attach static-site
```

85

85

Docker klijent [stop] - web app

- Zaustavljanje kontejnera koji je u pokrenutom (UP) stanju
  - naredba **`docker stop [OPTIONS] CONTAINER [CONTAINER...]`**
  - navodi se ID kontejnera koji se zaustavlja

```
$ docker ps -a
CONTAINER ID        IMAGE                     COMMAND             CREATED              STATUS              PORTS
NAMES
0988c7f5c91a        prakhar1989/static-site   "./wrapper.sh"      About a minute ago   Up About a minute
0.0.0.0:32769->80/tcp, 0.0.0.0:32768->443/tcp   static-site

$ docker stop 0988c7f5c91a
0988c7f5c91a
```

86

86

Docker klijent [start] - web app

- Pokretanje prethodno zaustavljenog kontejnera
  - naredba **`docker start [OPTIONS] CONTAINER [CONTAINER...]`**
  - navodi se ID kontejnera koji se ponovo pokreće

```
$ docker start 0988c7f5c91a
0988c7f5c91a
```

87

87

Docker slike kontejnera

- Prikazane osobine slike kontejnera
  - REPOSITORY - naziv repozitorijuma koji predstavlja datu sliku
  - TAG - oznaka verzije slike kontejnera koja je preuzeta
  - IMAGE ID - jednoznačna oznaka slike
    - u prikazu je skraćena verzija, za punu pozvati sa opcijom `--no-trunc`
  - CREATED - datum kreiranja slike
  - SIZE - veličina slike

```
$ docker images -a
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
hello-world             latest              2cb0d9787c4d        2 weeks ago         1.85kB
…
```

88

88

Docker slike kontejnera

- Slike se u terminologiji Docker-a dele na
  - **osnovne slike** (engl. *base images*)
    - slike koje nemaju roditeljsku sliku i od kojih se prave druge slike.
  - **zvanične slike** (engl. *official images*)
    - slike softvera koje su kreirali inženjeri u Docker-u
    - obično je njihov naziv sastavljen od jedne reči.
  - **izvedene slike** (engl. *child images*)
    - slike koje predstavljaju doradu osnovnih slika.
      - kreirane od strane korisnika Docker-a
    - obično je njihov naziv sastavljen od imena korisnika i naziva slike
      - npr. `user/image-name`

89

89

Docker slike kontejnera - Dockerfile

- Kreiranje slike kontejnera
  - potrebno je kreirati odgovarajući Dockerfile
    - sadrži uputstvo za kreiranje slike
    - potrebno definisati osnovnu sliku
    - potrebno definisati sve naredbe koje prebacuju osnovnu sliku u stanje potrebno za izvršavanje željene aplikacije
    - potrebno ubaciti kod naše aplikacije
  - svaka naredba u okviru Dockerfile-a koja menja osnovnu sliku uvodi novi sloj slike u repozitorijum slike

90

90

Docker slike kontejnera - Dockerfile

- Izgradnju slike obavlja docker pozadinski proces
  - docker klijent šalje zapakovan sadržaj direktorijuma koji se ubacuje u sliku
  - ovaj direktorijum mora da poseduje Dockerfile
    - ali može imati i dodatne datoteke i direktorijume
      - dodaju se u naredbom ADD ili COPY.
  - **napomena**: Celokupan direktorijum u kojem je pokrenuta naredba se pakuje i šalje pozadinskom procesu
    - obratiti pažnju da folder sadrži samo zaista potrebne fajlove

Docker slike kontejnera - Dockerfile

- Primer Node.js web aplikacije
  - zapakovati prostu, Hello World Node.js aplikaciju kao docker sliku
  - odgovarajući Dockerfile:

```
FROM node:8

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied where applicable
COPY package*.json ./
RUN npm install

# Bundle app source
COPY . .

EXPOSE 8080
CMD [ "npm", "start" ]
```

https://github.com/vdimitrieski/asvsp/tree/master/02_containers/docker-node-js-app

Docker slike kontejnera - Dockerfile

- Kopiranje nepotrebnih datoteka može biti izbegnuto
  - datoteka *.dockerignore*

```
node_modules
npm-debug.log
```

- Pravljenje slike i dodeljivanje oznake
  - naredba **`docker build [OPTIONS] PATH | URL | -`**
  - opcija **`-t`** služi za dodavanje oznake slici

93

93

Docker slike kontejnera - Dockerfile

```
$ docker build -t vdimitrieski/node-web-app .
Sending build context to Docker daemon  6.144kB
Step 1/7 : FROM node:8
 ---> ed145ef978c4
Step 2/7 : WORKDIR /usr/src/app
 ---> Using cache
 ---> db58b137c74f
Step 3/7 : COPY package*.json ./
 ---> Using cache
 ---> 23085fcd6fb7
Step 4/7 : RUN npm install
 ---> Using cache
 ---> ba377483612b
Step 5/7 : COPY . .
 ---> Using cache
 ---> 2a36c815ad09
Step 6/7 : EXPOSE 8080
 ---> Using cache
 ---> 7a93dd83dfef
Step 7/7 : CMD npm start
 ---> Using cache
 ---> e1c87eb88b8a
Successfully built e1c87eb88b8a
Successfully tagged vdimitrieski/node-web-app:latest
```

94

94

Docker slike kontejnera - Dockerfile

- Pokrenuti kreiranu sliku

```
$ docker run -p 49160:8080 -d vdimitrieski/node-web-app
```

- Otvoriti stranicu u internet pretraživaču
  - http://localhost:49160/

Docker slike kontejnera - Dockerfile

- Pregledanje logova za pokrenuti kontejner
  - naredba **`docker logs [OPTIONS] CONTAINER`**

```
$ docker ps
CONTAINER ID        IMAGE                      COMMAND            CREATED             STATUS              PORTS
NAMES
a77e1d94599f        vdimitrieski/node-web-app   "npm start"        3 minutes ago       Up 3 minutes
0.0.0.0:49160->8080/tcp   stoic_liskov

$ docker logs a77e1d94599f

> docker_web_app@1.0.0 start /usr/src/app
> node server.js

Running on http://0.0.0.0:8080
```

Docker slike kontejnera - Dockerfile

- Zaustaviti kontejner

```
$ docker stop a77e1d94599f
a77e1d94599f
```

97

Docker slike kontejnera - Dockviz

- Prikaz slojeva kreirane slike
  - od docker-a 1.7 ne postoji ugrađen alat za prikaz slojeva slike
  - može se iskoristiti alat Dockviz
    - preuzeti ga kao docker kontejner
      - dostupan na GitHub-u https://github.com/justone/dockviz
    - zgodno napravili alias naredbu koja pokreće docker kontejner

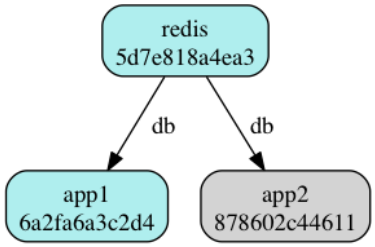
```
$ docker pull nate/dockviz
Unable to find image 'nate/dockviz:latest' locally
latest: Pulling from nate/dockviz
d6bcd48223ce: Pull complete
Digest: sha256:0157f9ec324cc8970a7034712c6ccbcfda4778cb7d0c004bbe68e670fbe61b4d
Status: Downloaded newer image for nate/dockviz:latest

$ alias dockviz="docker run -it --rm -v /var/run/docker.sock:/var/run/docker.sock nate/dockviz"
```

98

Docker slike kontejnera - Dockviz

- Dockviz
  - može se koristiti za vizualizaciju odnosa između kontejnera ili odnosa između slojeva slika

```
$ dockviz containers -d | dot -Tpng -o containers.png
```

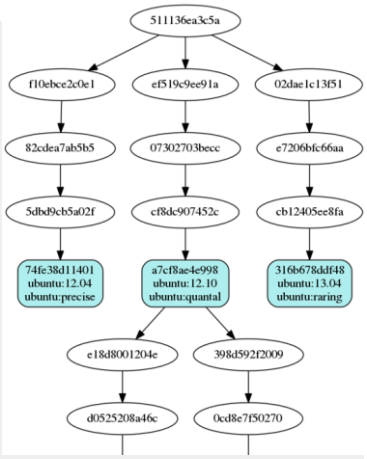


99

99

Docker slike kontejnera - Dockviz

```
$ dockviz containers -d | dot -Tpng -o containers.png

$ dockviz images -t -i
├─<missing> Size: 126.8 MB
│ └─<missing> Size: 0.0 B
│   └─<missing> Size: 41.1 MB
│     └─<missing> Size: 0.0 B
│       └─<missing> Size: 123.0 MB
│         └─<missing> Size: 320.1 MB
│           └─<missing> Size: 335.1 KB
│             └─<missing> Size: 135.6 KB
│               └─<missing> Size: 0.0 B
│                 └─<missing> Size: 56.6 MB
│                   └─<missing> Size: 0.0 B
│                     └─<missing> Size: 4.5 MB
│                       └─ed145ef978c4 Size: 0.0 B Tags: node:8
│                         └─db58b137c74f Size: 0.0 B
│                           └─23085fcd6fb7 Size: 265.0 B
│                             └─ba377483612b Size: 3.0 MB
│                               └─2a36c815ad09 Size: 884.0 B
│                                 └─7a93dd83dfef Size: 0.0 B
│                                   └─e1c87eb88b8a Size: 0.0 B Tags: vdimitrieski/node-web-app:latest
```



100

100

Docker slike kontejnera - Dockerfile primeri

```
FROM ubuntu

RUN apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
RUN echo "deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen" | tee -a
/etc/apt/sources.list.d/10gen.list
RUN apt-get update
RUN apt-get -y install apt-utils
RUN apt-get -y install mongodb-10gen

CMD ["/usr/bin/mongod", "--config", "/etc/mongodb.conf"]
```

101

101

Docker slike kontejnera - Dockerfile primeri

```
FROM golang:1.9.2-alpine3.6 AS build

# Install tools required for project
# Run `docker build --no-cache .` to update dependencies
RUN apk add --no-cache git
RUN go get github.com/golang/dep/cmd/dep

# List project dependencies with Gopkg.toml and Gopkg.lock
# These layers are only re-built when Gopkg files are updated
COPY Gopkg.lock Gopkg.toml /go/src/project/
WORKDIR /go/src/project/
# Install library dependencies
RUN dep ensure -vendor-only

# Copy the entire project and build it
# This layer is rebuilt when a file changes in the project directory
COPY . /go/src/project/
RUN go build -o /bin/project

# This results in a single layer image
FROM scratch
COPY --from=build /bin/project /bin/project
ENTRYPOINT ["/bin/project"]
CMD ["--help"]
```

102

102

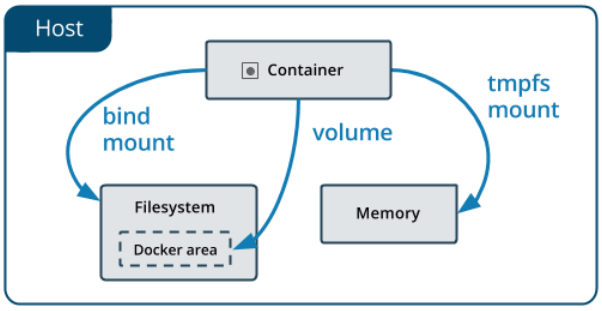
Docker slike kontejnera - Dockerfile

- Smernice za pravljenje Dockerfile-a
  - smanjiti broj koraka u Dockerfile-u
    - povećava performanse pravljenja i preuzimanja slika
      - **smanjuje broj slojeva slike**
  - sortirati delove naredbe unutar višelinijskih naredbi
  - početi Dockerfile sa koracima koji imaju najmanje izgleda da će biti promenjeni
    - instalirati prvo alate potrebne da aplikacija radi
    - instalirati sve potrebne biblioteke i pakete
    - kompajlirati aplikaciju
  - koristiti *.dockerignore* fajl kako bi se izbeglo nepotrebno kopiranje
  - kreirati slike tako da budu lako pokrenute i kontejneri lako zaustavljeni
    - *ephemeral* slike tj. slike koje ne čuvaju stanje u sebi već putem skladišta podataka

```
FROM alpine:3.4

RUN apk update && apk add \
    curl \
    git \
    vim
```


Docker slike kontejnera - skladišta podataka

- Skladišta podataka
  - Docker skladišta (engl. *volumes*)
  - skladišne tačke uvezivanja (engl. *bind mounts*)
  - privremene tačke uvezivanja (engl. *tmpfs mounts*)



izvor: *Docker Overview, Docker Volumes* https://docs.docker.com/storage/volumes/

Docker slike kontejnera - skladišta podataka

- Skladišta podataka
  - Docker skladišta (engl. *volumes*)
    - kreirana i upravljana od strane Docker-a (`/var/lib/docker/volumes/`)
      - naredba **`docker volume create [OPTIONS] [VOLUME]`**
    - registruju se i upravljaju na sličan način kao kontejneri i slike kontejnera
    - izolovano skladište od ostatka sistema datoteka
      - mogu ih menjati samo kontejneri kojima su skladišta dodeljena
    - mogu se deliti među kontejnerima
    - moraju se ručno osloboditi
      - naredba **`docker volume prune [OPTIONS]`**

105

105

Docker slike kontejnera - skladišta podataka

- Skladišta podataka
  - stalne tačke uvezivanja (engl. *bind mounts*)
    - omogućavaju korišćenje datoteka i direktorijuma sistema domaćina unutar kontejnera
    - mogu ih menjati i kontejneri i procesi sistema domaćina
  - privremene tačke uvezivanja (engl. *tmpfs mounts*)
    - nisu stalno skladištene ni u sistemu domaćinu ni u kontejnerima
    - najčešće ih koriste kontejneri za čuvanje privremenih rezultata sračunavanja ili veoma osetljive podatke koji nestaju nakon zaustavljanja kontejnera
- Dodeljivanje skladišta kontejnerima
  - Docker skladišta i stalne tačke uvezivanja se dodeljuju u naredbi run
    - opcija `-v` ili `--volume`
    - identično kao i kreiranje pomoću VOLUME naredbe u Dockerfile-u
  - privremene tačke uvezivanja se dodeljuju u naredbi `run`
    - opcija `--tmpfs`

106

106

Docker slike kontejnera - skladišta podataka

- Kada koristiti skladišta podataka
  - Docker skladišta
    - kada je potrebno podeliti datoteke između više kontejnera
    - kada postoji verovatnoća da sistem domaćin neće imati traženu strukturu direktorijuma da bi se napravile skladišne tačke uvezivanja
    - za smeštanje datoteka kontejnera na udaljene servere
    - u slučaju kada je kreiranje sigurnosne kopije i migracija podataka kontejnera potrebna
  - skladišne tačke uvezivanja
    - za deljenje konfiguracionih datoteka između sistema domaćina i kontejnera
    - kada je struktura direktorijuma u sistemu domaćinu zagarantovana
  - privremene tačke uvezivanja
    - kada god nam podaci iz kontejnera nisu potrebni između dva pokretanja

107

107

Docker slike kontejnera - Docker skladište primer

```
$ docker run -it -v /data --name container1 busybox

/ # cd data
/data # touch file1.txt
/data # ls
file1.txt
/data # exit

$ docker inspect container1
...
"Mounts": [
       {
       "Type": "volume",
       "Name": "568258a7940182c5ac52f0637c60c1d1f81e9ec77f3e4694647b4879c2ff003c",
       "Source": "/var/lib/docker/volumes/568258a7940182c5ac52f0637c60c1d1f81e9ec77f3e4694647b4879c2ff003c/_data",
       "Destination": "/data",
       "Driver": "local",
       "Mode": "",
       "RW": true,
       "Propagation": ""
       }
],
...
```

108

108

Docker slike kontejnera - Docker skladište primer (2)

```
$ docker restart container1

$ docker attach container1
/ # ls
bin   data  dev   etc   home  proc  root  sys   tmp   usr   var
/ # cd data
/data # ls
file1.txt
/data #

$ docker rm -v container1
```

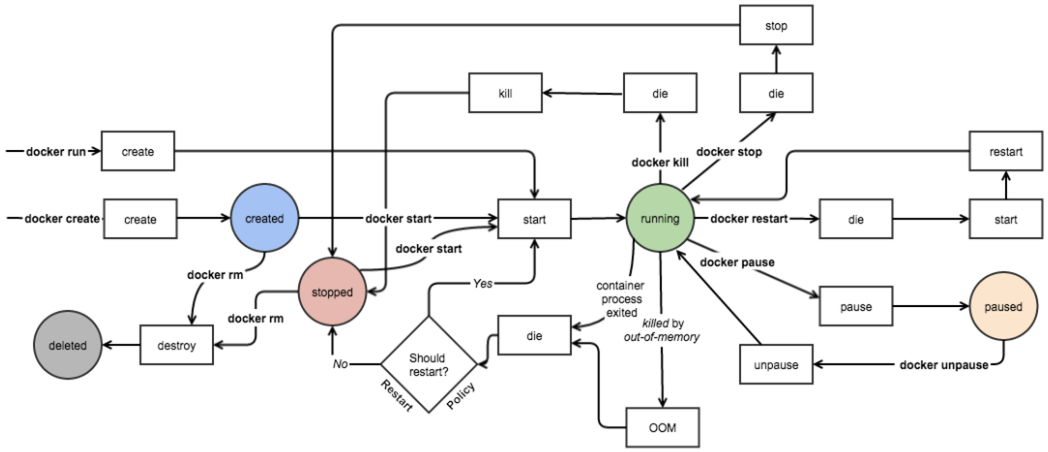
109

Docker slike kontejnera - skladišna tačka uvezivanja primer

```
$ mkdir ~/data
$ echo TEST TEST > ~/data/test.txt

$ docker run -it --name container1 -v ~/data:/datavol busybox
/ # cd datavol/
/datavol # ls
test.txt
/datavol # cat test.txt
TEST TEST
/datavol # exit
```

110

Doker kontejneri - životni ciklus



izvor: *Docker Internals*, http://docker-saigon.github.io

111

111

Docker Compose

112

Docker Compose

- Docker Compose je alat za pokretanje kontejnerizovanih aplikacija
  - kontejnerizovana aplikacija - aplikacija koja se sastoji od jednog ili više Docker kontejnera
  - kontejneri mogu biti povezani
    - pomoću skladišta ili simboličkih veza (engl. *links*)
  - olakšava rad sa kontejnerima
    - jedan *.yml* fajl sa svim podešavanjima
- Koraci za korišćenje alata Docker Compose
  - specifikacija kontejnera koji se koriste u kontejnerizovanoj aplikaciji
    - mora postojati Dockerfile ili slika mora biti dostupna na nekom od servera registara
  - konfiguracija kontejnera aplikacije
  - pokretanje kontejnerizovane aplikacije
    - naredbom **`docker-compose up [options] [--scale SERVICE=NUM...] [SERVICE...]`**

113

Docker Compose

- Primer *docker-compose.yml* datoteke

```
version: '3'
services:
  web:
    build: .
    ports:
    - "5000:5000"
    volumes:
    - .:/code
    - logvolume01:/var/log
    links:
    - redis
  redis:
    image: redis
volumes:
  logvolume01: {}
```

114

Docker Compose

- Osobine alata Docker Compose
  - omogućava pokretanje izolovanih izvršnih okruženja na istom sistemu domaćinu
    - npr. moguće je na razvojnoj mašini pokrenuti paralelno različite verzije istog softvera kako bi se lakše testirale funkcionalnosti
  - čuva podatke u skladištima između pokretanja
    - svi podaci smešteni u trajna skladišta su sačuvana
  - kešira podešavanje kontejnera između pokretanja
    - rekreiraju se samo kontejneri koji su u međuvremenu izmenjeni
    - ukoliko nije bilo izmena, docker compose ponovo koristi stari kontejner
  - dozvoljava korišćenje promenljivih iz izvršnog okruženja (engl. *environmental variables*)
    - za parametrizaciju podešavanja kontejnera
    - povećava stepen ponovne iskoristivosti specifikacija
      - za različita okruženja

115

115

Docker Compose

- Slučajevi korišćenja
  - pokretanje aplikacije na razvojnom računaru
    - jednom konfigurisane servise koje aplikacija koristi potrebno samo pokretati i zaustavljati
      - baze podataka, *cache*-ove, eksterne API-je itd.
    - lako postavljanje i uklanjanje izolovanih testnih okruženja
  - pokretanje automatizovanih testova
    - usled izolacije izvršnog okruženja, pogodni za testiranje
    - sve izmene načinjene tokom testiranja se lako mogu poništiti
  - postavljanje aplikacije na server sačinjen od jednog računara
    - ne postoji prava orkestracija
    - orkestraciju je moguće postići uz pomoć Docker Swarm alata

116

116

Docker Compose

- Napredni koncepti
  - nasleđivanje Docker Compose specifikacija
    - nasleđivanje cele specifikacije korišćenjem posebnih datoteka
    - naseđivanje pojedinačnih servisa korišćenjem polja **`extends`**

Docker Compose - Primer

- Primer korišćenja alata Docker Compose
  - za pokretanje Python Flask web aplikacije
  - nije potrebno predznanje Python jezika niti Flask radnog okvira
  - koristi Redis bazu podataka
    - za *cache* podataka
    - pokrenutu kao zaseban kontejner
      - nema potrebe instalirati Redis na razvojni računar
    - kao adresa Redis servera koristi se simbolička veza *redis* postavljena u datoteci *docker-compose.yml* i standardni port 6379

https://github.com/vdimitrieski/asvsp/tree/master/02_containers/docker-compose-flask-app

Docker Compose - Primer

- Primer *docker-compose.yml* datoteke

```
version: '3'
services:
  web:
    build: .
    ports:
     - "5000:5000"
  redis:
    image: "redis:alpine"
```

119

Docker Compose - Primer

- Izvršavanje aplikacije naredbom `docker-compose up`
  - zaustavljanje CTRL+C

```
$ docker-compose up
Starting docker-compose-flask-app_redis_1 ... done
Starting docker-compose-flask-app_web_1   ... done
Attaching to docker-compose-flask-app_redis_1, docker-compose-flask-app_web_1
redis_1  | 1:C 01 Aug 15:34:07.492 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
redis_1  | 1:C 01 Aug 15:34:07.492 # Redis version=4.0.10, bits=64, commit=00000000, modified=0, pid=1, just started
redis_1  | 1:C 01 Aug 15:34:07.492 # Warning: no config file specified, using the default config. In order to specify a
config file use redis-server /path/to/redis.conf
...
redis_1  | 1:M 01 Aug 15:34:07.495 * DB loaded from disk: 0.000 seconds
redis_1  | 1:M 01 Aug 15:34:07.495 * Ready to accept connections
web_1    |  * Serving Flask app "app" (lazy loading)
...
web_1    |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
web_1    |  * Restarting with stat
web_1    |  * Debugger is active!
web_1    |  * Debugger PIN: 898-272-666
```

120

Docker Compose - Primer

- U drugom terminalu
  - pokretanjem odgovarajućih naredbi možemo videti sve pokrenute kontejneri i preuzete slike

```
$docker ps

CONTAINER ID        IMAGE                          COMMAND                  CREATED             STATUS
PORTS                    NAMES
60288b0a3efb        docker-compose-flask-app_web   "python app.py"          8 minutes ago       Up 7 minutes
0.0.0.0:5000->5000/tcp   docker-compose-flask-app_web_1
503139beeb14        redis:alpine                   "docker-entrypoint..."   8 minutes ago       Up 7 minutes
6379/tcp                 docker-compose-flask-app_redis_1

$ docker image ls

REPOSITORY                                TAG                 IMAGE ID            CREATED             SIZE
docker-compose-flask-app_web              latest              358bb6ea506b        6 minutes ago       77.8MB
python                                    3.4-alpine          49fd33cb8033        5 days ago          66.5MB
redis                                     alpine              80581db8c700        3 weeks ago         28.6MB
...
```

121

Docker Compose - Primer

- Primer *docker-compose.yml* datoteke
  - sa mogućnošću direktne izmene koda
  - uvođenjem stalne tačke uvezivanja

```
version: '3'
services:
  web:
    build: .
    ports:
     - "5000:5000"
    volumes:
     - .:/code
  redis:
    image: "redis:alpine"
```

https://github.com/vdimitrieski/asvsp/tree/master/02_containers/docker-compose-flask-app-advanced

122

Docker Compose - Primer

- Izvršavanje aplikacije u pozadini
  - pokretanje naredbom `docker-compose up -d`
  - zaustavljanje naredbom docker-compose stop

```
$ docker-compose up -d
Starting docker-compose-flask-app_web_1   ... done
Starting docker-compose-flask-app_redis_1 ... done

$ docker-compose stop
Stopping docker-compose-flask-app_web_1   ... done
Stopping docker-compose-flask-app_redis_1 ... done
```

- Zaustavljanje aplikacije i uklanjanje svih podataka
  - naredba `docker-compose down`

123

123

Docker Compose

- Napredni koncepti (nasleđivanje)
  - nasleđivanje cele specifikacije korišćenjem posebnih datoteka

```
web:
  image: example/my_web_app:latest
  links:
    - db
    - cache
db:
  image: postgres:latest
cache:
  image: redis:latest
```

docker-compose.yml

```
web:
  ports:
    - 80:80
  environment:
    PRODUCTION: 'true'

cache:
  environment:
    TTL: '500'
```

docker-compose.prod.yml

```
$ docker-compose \
             -f docker-compose.yml
\
             -f docker-
compose.prod.yml \
             up -d
```

124

124

Docker Compose

- Napredni koncepti (nasleđivanje)
  - naseđivanje pojedinačnih servisa korišćenjem polja extends

```
app:
  build: .
  environment:
    CONFIG_FILE_PATH: /code/config
    API_KEY: xxxyyy
  cpu_shares: 5
```

common-services.yml

```
webapp:
  extends:
    file: common-services.yml
    service: app
  command: /code/run_web_app
  ports:
    - 8080:8080
  links:
    - queue
    - db
queue_worker:
  extends:
    file: common-services.yml
    service: app
  command: /code/run_worker
  links:
    - queue
```

docker-compose.yml

```
$ docker-compose up -d
```

125

Docker Compose - Primer

- Napredni koncepti (mrežna podešavanja)
  - portovi i simboličke veze
  - kreiranjem mreža
    - kojima se dodaju servisi
    - ipv4_address
    - ipv6_address

```
version: "3"
services:

  proxy:
    build: ./proxy
    networks:
      - frontend
  app:
    build: ./app
    networks:
      - frontend
      - backend
  db:
    image: postgres
    networks:
      - backend
networks:
  frontend:
    # Use a custom driver
    driver: custom-driver-1
    ipv4_address: 172.16.238.10
  backend:
    # Use a custom driver which takes special options
    driver: custom-driver-2
    driver_opts:
      foo: "1"
      bar: "2"
```

126

Docker Compose

- Napredni koncepti (redosled pokretanja/kreiranja kontejnera)
  - upotrebom polja **depends_on**
  - napomena: *compose* ne čeka dok kontejner ne bude spreman da pokrene sledeći
    - samo ih pokreće u odgovarajućem redosledu
    - rešenje: implementirati *.sh* skriptu da bi se postigla ova funkcionalnost
    - bolje rešenje: napraviti robusnu aplikaciju koja pokušava ponovno uspostavljanje veze kada jedan od servisa nije dostupan

```
version: "2"
services:
  web:
    build: .
    ports:
      - "80:8000"
    depends_on:
      - "db"
    command: ["./wait-for-it.sh", "db:5432", "--", "python", "app.py"]
  db:
    image: postgres
```

27

Docker Compose - dodatni primeri (rails)

```
FROM ruby:2.5
RUN apt-get update -qq && apt-get install -y
build-essential libpq-dev nodejs
RUN mkdir /myapp
WORKDIR /myapp
COPY Gemfile /myapp/Gemfile
COPY Gemfile.lock /myapp/Gemfile.lock
RUN bundle install
COPY . /myapp
```

```
version: '3'
services:
  db:
    image: postgres
    volumes:
      - ./tmp/db:/var/lib/postgresql/data
  web:
    build: .
    command: bundle exec rails s -p 3000 -b '0.0.0.0'
    volumes:
      - .:/myapp
    ports:
      - "3000:3000"
    depends_on:
      - db
```

128

Docker Compose - dodatni primeri (WordPress)

```
version: '3.3'
services:
   db:
     image: mysql:5.7
     volumes:
       - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress
   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
volumes:
    db_data:
```


129

Kubernetes

130

Orkestracija - alternative

- Docker Swarm
  - alat za orkestraciju koji je napravio i održava Docker tim
  - usko povezana sa Docker datotekama i alatom Docker Compose
  - jednostavan za korišćenje i instaliranje i izuzetno dobro se integriše sa postojećim Docker tehnologijama
  - najčešće dovoljan samo za jednostavnije aplikacije
- **Kubernetes (K8s)**
  - obuhvata nekoliko izvršnih okruženja za kontejnere pored Docker-a
  - najfleksibilniji i najveći skup funkcionalnosti
  - kompleksnije instaliranje i korišćenje
- Apache Mesos
  - postavljanje centara sa podacima (engl. *data centers*) gde postoji više aplikacija koje se istovremeno koriste
  - dozvoljava upotrebu bilo koje tehnologije za orkestraciju aplikacija

131

131

Kubernetes - osnovi

- Razvijen u Google-u
  - sredinom 2014. godine
  - projekat *Borg* (https://ai.google/research/pubs/pub43438)
- Kubernetes (gr. *κυβερνήτης* - kormilar)
  - platforma otvorenog koda za upravljanje kontejnerizovanim aplikacijama, koja omogućava:
    - automatizaciju puštanja aplikacije u produkciju
    - skaliranje pojedinačnih servisa (kontejnera) u okviru aplikacije
      - tj. ekosistema kontejnera
    - upravljanje konfiguracijama na nivou pojedinačnih servisa
    - deli kontejnerizovanu aplikaciju u manje logičke jedinice zasnovane na kontejnerima
      - radi lakšeg upravljanja
  - platforma za upravljanje mikroservisima
    - predstavljenim pomoću kontejnera

132

132

Kubernetes - osnovi

- Kubernetes je platforma
  - za kreiranje ekosistema komponenti i alata
  - u cilju što lakšeg puštanja komponenti u produkciju, njihovog skaliranja i upravljanja
  - orkestrira kontejnere tako što pojednostavljuje i apstrahuje servise niskog nivoa i skriva ih od krajnjeg korisnika
    - olakšava upravljanje resursima, umrežavanje, infrastrukturu za skladištenje podataka itd.
    - nudi jednostavnost PaaS sa fleksibilnošću IaaS
  - omogućava prenosivost infrastrukture na druge IaaS sisteme

133

133

Kubernetes - osnovi

- Osobine Kubernetes-a
  - omogućava **optimizaciju opterećenja kontejnera**
    - pokretanjem novih instanci kontejnera po potrebi, pri tome ne narušavajući dostupnost, sigurnost i konfigurisana pravila
  - dozvoljava **horizontalno skaliranje** (ručno ili automatski)
  - omogućava **automatizovano ažuriranje kontejnerizovane aplikacije** kao i njeno vraćanje na stariju verziju
    - izmene se propagiraju inkrementalno, kontejner po kontejner
    - uz nadgledanje statusa celokupnog sistema
  - omogućava automatizovano priključivanje i izmenu skladišta podataka
    - bez zaustavljanja aplikacije

134

134

Kubernetes - osnovi

- Osobine Kubernetes-a
  - poseduje mehanizam za brzi oporavak
    - ponovno pokretanje kontejnera koji se više ne izvršavaju ili za koje korisnički kreirana provera ne prolazi
    - šalje spoljnom svetu samo informacije o pokrenutim i dostupnim kontejnerima
  - poseduje mehanizam za otkrivanje servisa (engl. *service discovery*) i izjednačavanje opterećenja (engl. *load balancing*) kontejnera
  - omogućava razdvajanje aplikacija od infrastrukture
    - članovi razvojnog tima brinu se samo o aplikacijama ali ne i o potrebnim hardversko/komunikacionim elementima
  - omogućava lako upravljanje klasterom
    - nema ga smisla koristiti na jednom računaru
      - osim za testiranje i edukaciju

135

135

Kubernetes - osnovne komponente

- Osnovne komponente Kubernetes klastera:
  - **glavni čvor** (engl. *master*)
    - glavni čvor ekosistema, koji upravlja i nadgleda jedan ili više podređenih čvorova
  - **podređeni čvor** (engl. *node, minion*)
    - čvor koji izvršava zadatke koje mu dostavi glavni čvor ili korisnik
  - **kapsula** (engl. *pod*)
    - aplikacija ili njen deo koja se izvršava na podređenom čvoru
    - najmanja jedinica manipulacije u Kubernetes-u
  - **upravljač replikacijom** (engl. *replication controller*)
    - komponenta koja obezbeđuje da se definisani broj kapsula izvršava na podređenim čvorovima u svakom trenutku
    - u novijim verzijama alata, zamenjena replikacionim skupom i njegovim postavkama

136

136

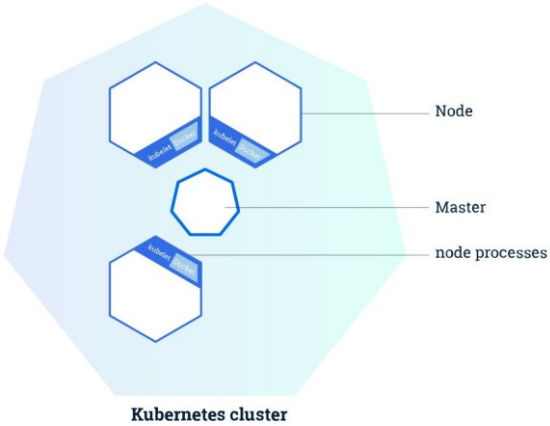
Kubernetes - osnovne komponente

- Osnovne komponente Kubernetes klastera:
  - **labela** (engl. *label*)
    - proizvoljni par ključ:vrednost
    - koji upravljač replikacijom koristi za obezbeđivanje otkrivanja servisa
  - **servis** (engl. *service*)
    - komponenta koja obezbeđuje izjednačavanje opterećenja u okviru replicirane grupe kapsula
  - **skladišta podataka** (engl. *volumes*)
    - komponente koje se koriste za skladištenje podataka

137

137

Kubernetes - čvorovi

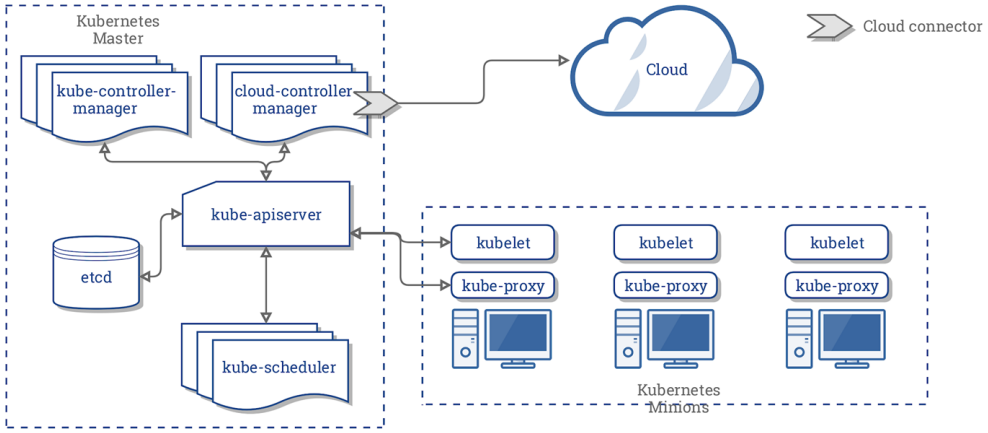


izvor: *Kubernetes Basics, Kubernetes docs* https://kubernetes.io/docs

138

138

Kubernetes - arhitektura



izvor: *Concepts Underlying the Cloud Controller Manager, Kubernetes docs* https://kubernetes.io/docs

Kubernetes - glavni čvor (*Master*)

- Glavni čvor
  - omogućava upravljanje klasterom
  - mesto gde je implementirana logika reagovanja na događaje u klasteru
  - može biti implementiran na bilo kojoj mašini u klasteru
    - ali obično postoji poseban računar za glavni čvor
- Komponente glavnog čvora
  - *kube-apiserver*
    - Kubernetes API za rad sa Kubernetes komponentama
    - glavni čvor kreira kontejnere koji mogu da odgovore na postavljene zahteve koji bi trebalo da se izvrše
    - dizajniran za horizontalno skaliranje

Kubernetes - glavni čvor (*Master*)

- Komponente glavnog čvora
  - *etcd*
    - konzistentno i visoko raspoloživo skladište podataka
    - interno ga koristi Kubernetes
    - veoma bitno redovno praviti rezervne kopije ovog skladišta
  - *kube-scheduler*
    - komponenta koja prati novokreirane kapsule koje nisu dodeljene čvorovima
      - i dodeljuje ih odgovarajućem čvoru u klasteru
    - uzima u obzir lokalne i globalne potrebe za resursima, lokalnost podataka, dostupnost delova mreže itd.

141

141

Kubernetes - glavni čvor (*Master*)

- Komponente glavnog čvora
  - *kube-controller-manager*
    - upravljač kontrolerima (upravljačima) u okviru glavnog čvora:
      - upravljač replikacijom - obezbeđuje dostupnost delova sistema
      - upravljač servisima - upravlja servisima i povezuje ih sa kapsulama
      - upravljač čvorovima - posmatra status čvorova
      - upravljač nalozima i tokenima za servise
  - *cloud-controller-manager*
    - upravljač kontrolerima koji komuniciraju sa pružaocem usluga infrastrukture
      - na kojoj se Kubernetes nalazi
      - obuhvata kontrolere za rad sa mrežom, skladištima podataka itd.

142

142

Kubernetes - podređeni čvor (*Node*)

- Komponente podređenog čvora
  - *kubelet*
    - softverski agent koji obezbeđuje da su kontejneri pokrenuti u kapsulama
    - pokreće, zaustavlja i nadgleda izvršavanje kontejnera
  - *kube-proxy*
    - omogućava kreiranje i apstrakciju API-ja kontejnera pomoću servisa
  - izvršno okruženje kontejnera
    - omogućava pokretanje kontejenera

143

143

Kubernetes - komunikacija

- Komunikacija u smeru od podređenih ka glavnom čvoru
  - preko HTTPS
  - mora biti uspostavljen neki oblik autentifikacije
  - komunicira se isključivo sa *kube-apiserver*
  - obuhvata siganlizaciju, događaje i zahteve kapsula
- Komunikacija u smeru od glavnog ka podređenim čvorovima
  - *kube-apiserver -> kubelet*
    - preuzimanje logova, dodela kapsula čvorovima, preusmerenje portova
  - *kube-apiserver* -> čvorovi, kapsule, i servisi
    - komunikacija po HTTP protokolu (može i HTTPS)

144

144

Kubernetes - objekti

- Objekti su glavne gradivne komponente u Kubernetes-u
  - trajni entiteti u ekosistemu koje Kubernetes koristi kako bi opisao stanje klastera
    - objekti predstavljaju zapis o nameri
      - tj. pomoću njih opisujemo kako želimo tačno da klaster izgleda
  - opisuju:
    - koje aplikacije se izvršavaju i na kojim čvorovima
    - koje resurse aplikacije koriste
    - polise dodeljene tim aplikacijama

145

145

Kubernetes - objekti

- Polja Kubernetes objekta
  - *spec*
    - sadrži specifikaciju željenog stanja objekta,
      - kroz podešavanje njegovih karakteristika
  - *status*
    - sadrži pravo stanje objekta
      - koje održava sam Kubernetes
- Kubernetes nadgleda i poredi vrednosti u dva polja
  - u slučaju razlike reaguje na odgovarajući način
    - npr. podiže dodatnu instancu ukoliko je broj instanci u polju status manji od broja instancu u polju spec

146

146

Kubernetes - objekti

- Kreiranje objekata
  - naredba **`kubectl create`**
  - opcija -f za fajl

147

147

Kubernetes - labele i selektori

- Labela (engl. *label*)
  - predstavlja par ključ:vrednost
    - koji korisniku sistema služi za identifikaciju i pretragu objekata
    - ne nosi semantiku u Kubernetes ekosistemu
  - može biti dodeljen bilo kom Kubernetes objektu
    - prilikom kreiranja objekta ili u bilo kojem trenutku njegove eksploatacije
  - nisu jedinstvene
    - više objekata može imati isu labelu
- Selektor (engl. *selector*)
  - koristi se za odabir objekata koji zadovoljavaju uslov selekcije
    - npr. u okviru servisa se moraju odabrati kapsule na koje se servis odnosi
  - selektor po jednakosti
    - selektor koji koristi operaciju jednakosti
  - skupovni selektor
    - selektor koji koristi operacije nad skupovima (`in`, `notin` i `exists`)

148

148

Kubernetes - labele i selektori

```
"metadata": {
  "labels": {
    "key1" : "value1",
    "key2" : "value2"
  }
}
```

labele

```
environment = production
tier != frontend
```

selekcija po jednakosti

```
environment in (production, qa)
tier notin (frontend, backend)
partition
!partition
```

skupovni selektor

149

149

Kubernetes - objekti

- Moguće je kreirati sledeće objekte (dati najčešći):
  - kapsula
  - upravljač replikacijom
  - postavka replikacionog skupa
  - servis
  - ingress

150

150

Kubernetes - kapsule

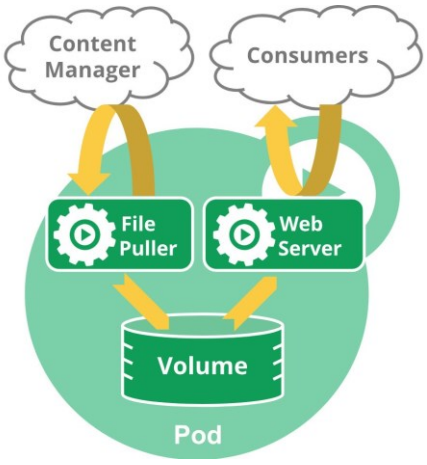
- Kapsule
  - predstavljaju najmanje i najjednostavnije objekte koji se mogu kreirati i pokrenuti
  - predstavljaju procese koji se izvršavaju u okviru Kubernetes klastera
  - enkapsuliraju kontejnerizovanu aplikaciju sa:
    - jednim ili više kontejnera
      - najčešće sa jednim kontejnerom
    - resursima za skladište podataka
    - jedinstvenom IP adresom
    - parametrima koji upravljaju izvršavanjem kontejnera
  - predstavljaju apstrakciju u odnosu na kontejnere
    - jer Kubernetes može da radi sa više izvršnih okruženja kontejntera

151

151

Kubernetes - kapsule

- Kapsule
  - izvršavaju samo jednu instancu kontejnerizovane aplikacije
  - moguće horizontalno skaliranje (replikacija)
  - specificiraju se u vidu šablona koji se pišu u **spec** polju Kubernetes objekta
- Kapsule sa više kontejnera
  - uvek su vezane za jedan čvor
  - zahtevaju dosta više podešavanja
  - koriste se samo kada su kontejneru u aplikaciji tesno povezani

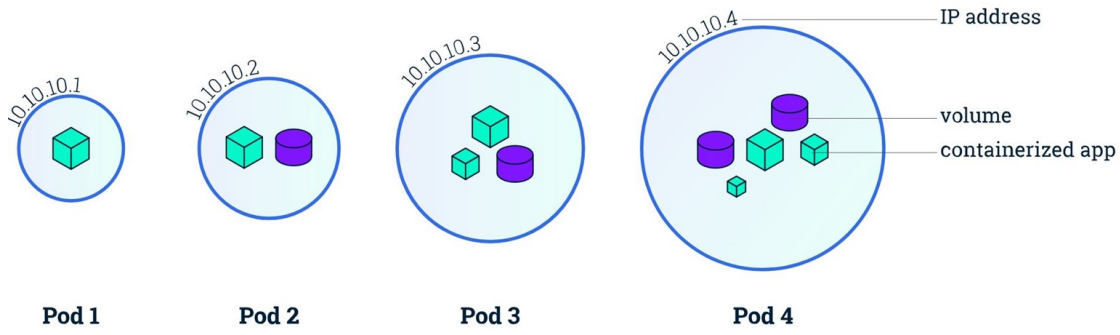


izvor: *Pod Overview, Kubernetes docs* https://kubernetes.io/docs

152

152

Kubernetes - kapsule

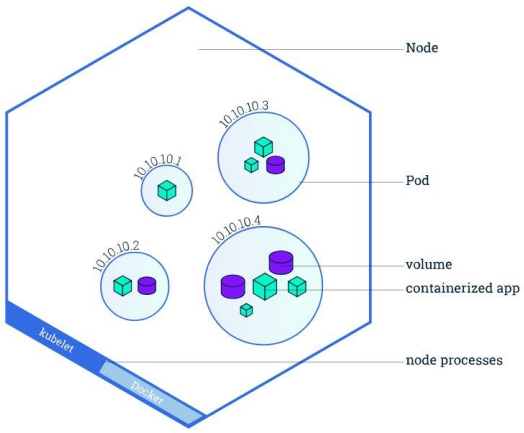


izvor: *Kubernetes Basics, Kubernetes docs* https://kubernetes.io/docs

153

153

Kubernetes - čvorovi i kapsule



izvor: *Kubernetes Basics, Kubernetes docs* https://kubernetes.io/docs

154

154

Kubernetes - kapsule

- Kreiranje kapsula
  - kao i kreiranje bilo kod drugog objekta
  - naredbom **`kubectl create`**
- Prikaz kapsula
  - naredba **`kubectl get pods`**
  - opcija -l omogućava specifikaciju labele za pretragu kapsula
    - `kubectl get pods -l env=test`
- Brisanje kapsula
  - naredbom **`kubectl delete pod`**
  - opcija -l omogućava specifikaciju labele za brisanje kapsula
    - `kubectl delete pod -l env=test`

155

155

Kubernetes - kapsule

```
application/deployment.yaml
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

```
$ kubectl create -f deployment.yaml
deployment "nginx-deployment" created
$ kubectl delete nginx-deployment
```

156

156

Kubernetes - upravljači replikacijom

- Upravljač replikacijom
  - Kubernetes objekat koji osigurava da se u svakom trenutku izvršava željeni broj kapsula
    - ukoliko je pokrenut preveliki broj kapsula, upravljač replikacijom zaustavlja kapsule
    - ukoliko je pokrenut premali broj kapsula, upravljač replikacijom pokreće nove kapsule
    - u slučaju greške, zaustavljene kapsule kojima upravlja upravljač replikacijom se automatski ponovo pokreću
      - za razliku od ručno pokrenutih kapsula
      - koristiti upravljač replikacijom čak i kada imamo jednu kapsulu
    - trenutno posmatra samo pokrenute i zaustavljene kapsule
      - u budućnosti će biti dodat i detaljniji status kapsula

157

157

Kubernetes - upravljači replikacijom

- Upravljač replikacijom koristi se za
  - **ponovno pokretanje** (engl. *rescheduling*)
    - bilo da imamo jednu ili više kapsula, u slučaju zaustavljanja kapsule upravljač će je ponovo pokrenuti
  - **skaliranje** (engl. *scaling*)
  - **postepeno ažuriranje** (engl. *rolling updates*)
    - omogućava postepeno ažuriranje servisa
      - kapsule su zamenjene jedna po jedna
      - ne narušavajući ukupnu performantnost i dostupnost sistema
      - naredba kubectl rolling-update
  - **održavanje višestrukih verzija aplikacije** (engl. *multiple release tracks*)

158

158

Kubernetes - objekti, primer upravljača replikacijom

```
apiVersion: v1 #replication.yaml
kind: ReplicationController
metadata:
  name: nginx
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

```
$ kubectl create -f https://k8s.io/examples/controllers/replication.yaml
replicationcontroller "nginx" created
```

159

159

Kubernetes - objekti, primer upravljača replikacijom

```
$ kubectl describe replicationcontrollers/nginx
Name:        nginx
Namespace:   default
Selector:    app=nginx
Labels:      app=nginx
Annotations:    <none>
Replicas:    3 current / 3 desired
Pods Status: 3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:       app=nginx
  Containers:
   nginx:
    Image:              nginx
    Port:               80/TCP
    Environment:        <none>
    Mounts:             <none>
  Volumes:              <none>
Events:
FirstSeen       LastSeen     Count    From                    SubobjectPath    Type      Reason              Message
---------       --------     -----    ----                    -------------    ----      ------              -------
20s             20s          1                {replication-controller }      Normal    SuccessfulCreate
Created pod: nginx-qrm3m
...
```

160

160

Kubernetes - replikacioni skup

- Replikacioni skup (engl. *Replica Set*)
  - nova verzija upravljača replikacijom
  - jedina razlika u odnosu na upravljač se ogleda u podršci za naprednije upite u polju selector
    - mogu se koristiti i skupovni slektori
  - obično se ne koriste direktno, već isključivo posredstvom **postavke replikacionog skupa**
    - koji nudi deklarativniji mehanizam za upravljanje replikacionim skupovima

161

161

Kubernetes - postavka replikacionog skupa

- Postavka replikacionog skupa (engl. *Deployment*)
  - enkapsulacija replikacionih skupova
    - nudi mogućnost deklarativne specifikacije ažuriranja za kapsule i replikacione skupove
  - izmenom specifikacije željenog stanja objekta postavke, upravljač postavkama menja trenutno stanje kapsula u željeno
    - na kontrolisan način

162

162

Kubernetes - postavka replikacionog skupa

- Postavka replikacionog skupa omogućava:
  - upravljanje replikacionim skupovima
  - deklarisanje novog stanja kapsula, ažuriranjem specifikacije njihovog objekta
    - kreira se novi replikacioni skup i kapsule se prebacuju postepeno iz starog skupa u novi
  - povratak na prethodnu verziju postavke
  - pauziranje izvršavanja kapsula
    - radi velikih postepenih izmena
  - čišćenje nepotrebnih replikacionih skupova

163

163

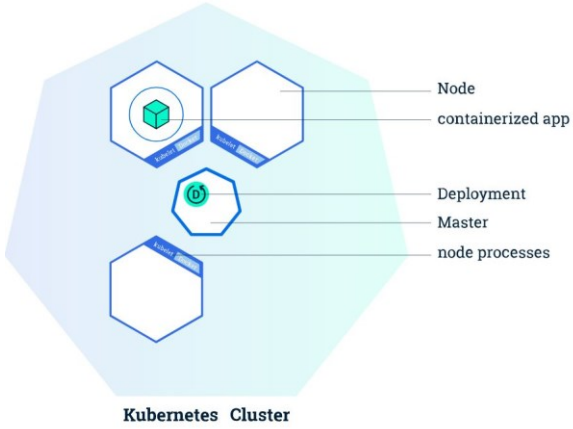
Kubernetes - postavka replikacionog skupa

- Kreiranje kapsula
  - kao i kreiranje bilo kod drugog objekta
  - naredbom **`kubectl create`**
- Prikaz postavki
  - naredba **`kubectl get deployment`**
- Ažuriranje postavki putem konfiguracione datoteke
  - **`kubectl apply -f https://k8s.io/examples/application/deployment-update.yaml`**
- Brisanje kapsula
  - naredbom **`kubectl delete deployment`**

164

164

Kubernetes - postavka

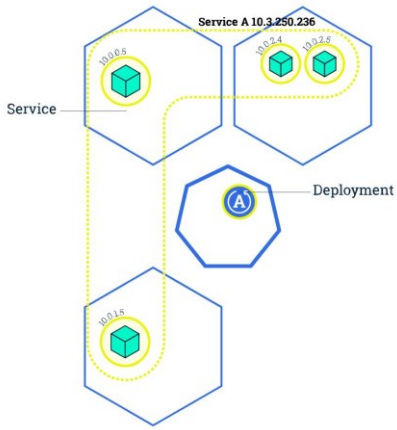


izvor: *Kubernetes Basics, Kubernetes docs* https://kubernetes.io/docs

165

165

Kubernetes - replikacija

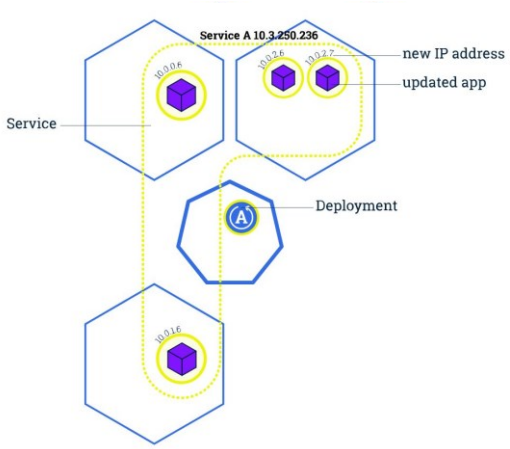


izvor: *Kubernetes Basics, Kubernetes docs* https://kubernetes.io/docs

166

166

Kubernetes - postepeno ažuriranje



izvor: *Kubernetes Basics, Kubernetes docs* https://kubernetes.io/docs

167

Kubernetes - objekti, primer postavke

```
apiVersion: apps/v1 #nginx-deployment.yaml
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

168

Kubernetes - objekti, primer postavke

```
$ kubectl create -f  nginx-deployment.yaml
$ kubectl get deployments
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3         0         0            0           1s

$ kubectl rollout status deployment/nginx-deployment
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
deployment "nginx-deployment" successfully rolled out

$ kubectl get deployments
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3         3         3            3           18s

$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-2035384211   3         3         3       18s

$ kubectl get pods --show-labels
NAME                                READY     STATUS    RESTARTS   AGE       LABELS
nginx-deployment-2035384211-7ci7o   1/1       Running   0          18s       app=nginx,pod-template-hash=2035384211
nginx-deployment-2035384211-kzszj   1/1       Running   0          18s       app=nginx,pod-template-hash=2035384211
nginx-deployment-2035384211-qqcnn   1/1       Running   0          18s       app=nginx,pod-template-hash=2035384211
```

169

169

Kubernetes - objekti, primer ažuriranja postavke

```
#from 1.7.9 to 1.9.1
$ kubectl set image deployment/nginx-deployment nginx=nginx:1.9.1
deployment "nginx-deployment" image updated
#-----rucno izmeniti datoteku i pokrenuti
# $ kubectl edit deployment/nginx-deployment
# deployment "nginx-deployment" edited

$ kubectl rollout status deployment/nginx-deployment
Waiting for rollout to finish: 2 out of 3 new replicas have been updated...
deployment "nginx-deployment" successfully rolled out

$ kubectl get deployments
NAME               DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment   3         3         3            3           36s

$ kubectl get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-1564180365   3         3         3       6s
nginx-deployment-2035384211   0         0         0       36s

$ kubectl get pods
NAME                                READY     STATUS    RESTARTS   AGE
nginx-deployment-1564180365-khku8   1/1       Running   0          14s
nginx-deployment-1564180365-nacti   1/1       Running   0          14s
nginx-deployment-1564180365-z9gth   1/1       Running   0          14s
```

170

170

Kubernetes - servisi

- Servisi
  - predstavlja apstrakciju koja definiše skup kapsula i način pristupa
    - skriva infrastrukturalne detalje od korisnika usluge
    - uvek se pristupa servisu koji zatim prosleđuje zahtev odgovarajućoj kapsuli
      - kapsula se nalazi na IP adresi koja je izmenljiva usled pokretanja i spuštanja prouzrokovanih radom upravljača replikacijom ili replikacionog skupa
  - termin koji se još koristi: **mikroservis**
  - određen specificiranim selektorom labele
    - koji odabira kapsule koje će biti enkapsulirane servisom
  - moguće je implementirati servis bez selektora
    - za prosleđivanje zahteva aplikacijama koje nisu deo Kubernetes ekosistema
    - potrebno kreirati krajnju tačku (engl. *endpoint*) za definisanje IP adrese eksternog servisa
  - svakom servisu je dodeljena IP adresa kako bi se sprečila kolizija između servisa
  - moguće je dodeliti sertifikate servisu

171

171

Kubernetes - servisi

- Servisi
  - mogu biti za internu upotrebu unutar klastera
    - kada kapsule pristupaju drugim kapsulama jer zajedno čine neku aplikaciju
    - tip servisa **ClusterIP**
  - mogu se otvoriti za saobraćaj izvan klastera
    - tip servisa **NodePort** definiše statičku adresu kojoj će biti pristupano spolja
      - automatski se kreira ClusterIP servis na koji se prosleđuju pozivi
    - tip servisa **LoadBalancer** pruža usluge izjednačavanja opterećenja u pristupu kapsulama
      - automatski se kreiraju NodePort i ClusterIP servisi na koje se prosleđuju pozivi
    - tip servisa **ExternalName** vraća CNAME zapis na DNS serveru
      - zahteva verziju >=1.7 *kube-dns* komponente

172

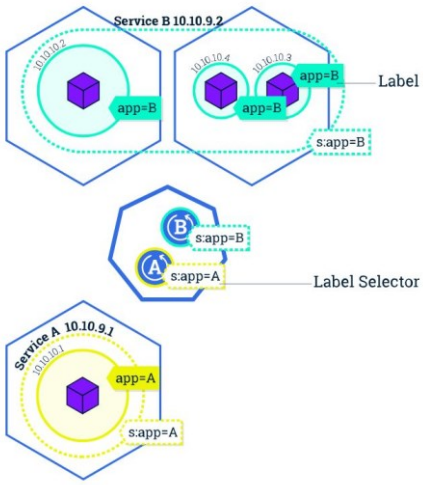
172

Kubernetes - servisi

- Kreiranje servisa
  - kao i kreiranje bilo kog drugog objekta
  - naredbom **`kubectl create`**
- Prikaz servisa
  - naredba **`kubectl get services`**
- Brisanje servisa
  - naredbom **`kubectl delete service nginx-service`**

173

Kubernetes - postavka



izvor: *Kubernetes Basics, Kubernetes docs* https://kubernetes.io/docs

174

Kubernetes - servisi

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

```
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

```
kind: Endpoints
apiVersion: v1
metadata:
  name: my-service
subsets:
  - addresses:
      - ip: 1.2.3.4
    ports:
      - port: 9376
```

175

Kubernetes - servisi, primer

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 2
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      containers:
      - name: my-nginx
        image: nginx
        ports:
        - containerPort: 80
```

```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  ports:
  - port: 80
    protocol: TCP
  selector:
    run: my-nginx
```

176

Kubernetes - servisi, primer

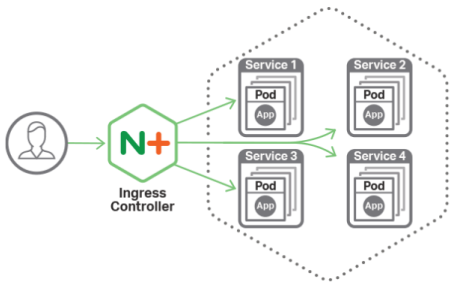
```
$ kubectl create -f ./run-my-nginx.yaml
$ kubectl get pods -l run=my-nginx -o wide
NAME                        READY     STATUS    RESTARTS   AGE       IP            NODE
my-nginx-3800858182-jr4a2   1/1       Running   0          13s       10.244.3.4    kubernetes-minion-905m
my-nginx-3800858182-kna2y   1/1       Running   0          13s       10.244.2.5    kubernetes-minion-ljyd
$ kubectl get pods -l run=my-nginx -o yaml | grep podIP
    podIP: 10.244.3.4
    podIP: 10.244.2.5


$ kubectl expose deployment/my-nginx
service "my-nginx" exposed
# ---------- or creating by .yml
# $ kubectl create -f ./nginx-svc.yaml

$ kubectl get svc my-nginx
NAME       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
my-nginx   ClusterIP   10.0.162.149   <none>        80/TCP    21s

$ kubectl get ep my-nginx
NAME       ENDPOINTS                     AGE
my-nginx   10.244.2.5:80,10.244.3.4:80   1m
```

177

177

Kubernetes - ingress

- Ingress
  - komponenta koja dozvoljava preusmeravanje saobraća sa interneta ka unutrašnjim čvorovima u klasteru
  - skup pravila za rutiranje dolaznog saobraćaja
  - omogućava dodelu eksternih URL-ova resursima
  - omogućava izjednačavanje opterećenja



izvor: *Nginx Ingress, Official docs*
https://www.nginx.com/products/nginx/kubernetes-ingress-controller/

178

178

Kubernetes - ingress

- Ingress tipovi
  - Ingress za jedan servis (engl. *Single Service Ingress*)
    - alternativa za servise tipa NodePort i LoadBalancer
  - jednostavni izjednačavač opterećenja (engl. *Simple fanout*)
    - izjednačavač opterećenja za spoljni saobraćaj u odnosu na klaster
    - omogućava smanjenje broja izjednačavača u samom klasteru
  - Ingress za virtualni hosting zasnovan na imenu (engl. *Name based virtual hosting*)
    - dozvoljava postojanje više URL-ova za istu IP adresu

179

179

Kubernetes - ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```

primer Ingress specifikacije

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test-ingress
spec:
  backend:
    serviceName: testsvc
    servicePort: 80
```

```
$ kubectl get ing
NAME                RULE          BACKEND        ADDRESS
test-ingress        -             testsvc:80
107.178.254.228
```

Ingress za jedan servis

180

180

Kubernetes - ingress

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - path: /foo
        backend:
          serviceName: s1
          servicePort: 80
      - path: /bar
        backend:
          serviceName: s2
          servicePort: 80
```

```
$ kubectl get ing
NAME      RULE          BACKEND   ADDRESS
test      -
          foo.bar.com
          /foo          s1:80
          /bar          s2:80
```

jednostavni izjednačavač opterećenja

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: test
spec:
  rules:
  - host: foo.bar.com
    http:
      paths:
      - backend:
          serviceName: s1
          servicePort: 80
  - host: bar.foo.com
    http:
      paths:
      - backend:
          serviceName: s2
          servicePort: 80
```

Ingress za virtualni hosting zasnovan na imenu

181

181

Kubernetes - skladišta podataka

- Skladišta podataka (engl. *volumes*)
  - omogućavaju čuvanje podataka između pokretanja kontejnera
  - omogućavaju deljenje sadržaja između kontejnera/kapsula
  - moćniji koncept od Docker skladišta
    - mogućnost specificiranja životnog veka skladišta
    - podržan veliki broj različitih tipova skladišta
    - mogućnost dodele više skladišta kontejnerima
      - ukoliko kontejneri to podržavaju
  - deo su kapsule
    - traju koliko i kapsula i životni vek im nije povezan sa životnim vekom kontejnera u kapsuli

182

182

Kubernetes - skladišta podataka

```
apiVersion: v1
kind: Pod
metadata:
  name: test-ebs
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-ebs
      name: test-volume
  volumes:
  - name: test-volume
    # This AWS EBS volume must already exist.
    awsElasticBlockStore:
      volumeID: <volume-id>
      fsType: ext4
```

183

183

Docker Swarm i Mesos

- Online Kubernetes laboratorija
  - https://labs.play-with-k8s.com/
  - https://www.katacoda.com/courses/kubernetes/playground
- Preporučujemo da se upoznate i sa ostalim rešenjima
  - Docker Swarm
    - https://docs.docker.com/engine/swarm/
  - Apache Mesos
    - http://mesos.apache.org/
    - Testni poligon https://dcos.io/

184

184

Literatura

- https://rhelblog.redhat.com/2015/07/29/architecting-containers-part-1-user-space-vs-kernel-space/
- https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/#h.dqlu6589o789
- https://github.com/veggiemonk/awesome-docker
- https://github.com/ramitsurana/awesome-kubernetes
- https://cloud.google.com/kubernetes-engine/kubernetes-comic/

185