UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Računarstvo u oblaku

ms Helena Anišić

Zimski semester 2025/2026.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi

Skladištenje podataka u Docker-u

Pokazni primer [*feedback-app*]

- Kontejnerizovati aplikaciju i pristupiti joj preko veb pretraživača.
- **Scenario A:**
  - Zaustaviti kontejner i ponovo ga pokrenuti.
- **Scenario B:**
  - Obrisati kontejner i ponovo ga pokrenuti.

Koja je razlika između ova dva scenarija?

Vrste podataka u Docker-u

Aplikacija (kod + okruženje)	Privremeni podaci aplikacije	Permanentni podaci aplikacije
Napisan od strane programera	Dobavljeno / Kreirano u pokrenutom kontejneru	Dobavljeno / Kreirano u pokrenutom kontejneru
Dodato u sliku kontejnera prilikom build faze	Skladišteno u memoriji ili privremenim fajlovima	Skladišteno u fajlovima ili u bazi podataka
Fiksno - ne može da se promeni nakon što se slika build-uje	Dinamični i promenljivi	Ne sme da se izgubi ako se kontejner obriše
Read-only, skladišteno u slici	Read + write, privremeni, skladišteni u kontejneru	Read + write, permanentni, skladišteni u kontejneru (volumes)

Odnos podataka i kontejnera

Odnos podataka i kontejnera

Dockerfile instrukcija [VOLUME]

```
VOLUME ["/data"]
```

- `VOLUME` instrukcija kreira tačku *mount*-ovanja na definisanoj putanji u kontejneru
- Sa te putanje se učitavaju fajlovi/direktorijumi svaki put kada se pokrene kontejner
  - fajlovi/direktorijumi mogu biti sa host mašine ili drugih kontejnera
- Mount tačka može da bude zadata kao: JSON array ili string
  - JSON array: `VOLUME ["/var/log/"]`
  - string: `VOLUME /var/log`
- Namena `VOLUME` instrukcije:
  - čuvanje podataka čak iako se kontejner zaustavi ili ukloni,
  - deljenje podataka između kontejnera i
  - lakše upravljanje podacima.

Pokazni primer [*feedback-app*]

- Definisati docker skladište podataka u okviru Dockerfile-a.
- Obezbediti trajno čuvanja tekstualnih fajlovi koji se nalaze u okviru *feedback* direktorijuma u aplikaciji.
- Ponoviti scenario B.

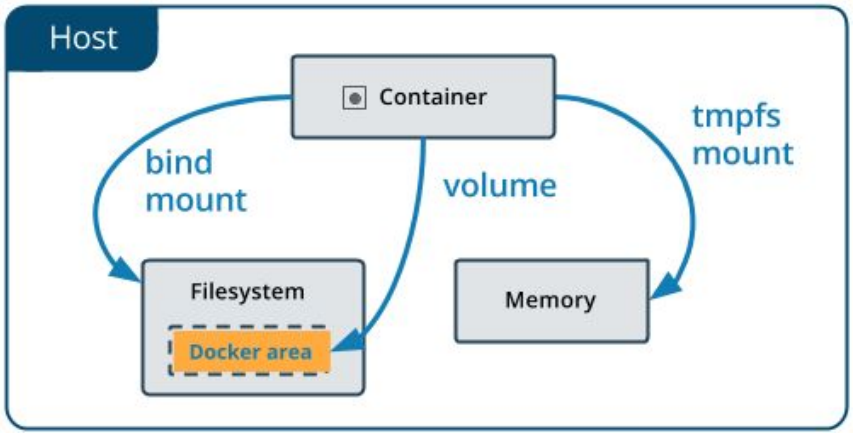
Pokazni primer [*feedback-app*]

- Definisati docker skladište podataka u okviru Dockerfile-a.
- Obezbediti trajno čuvanja tekstualnih fajlovi koji se nalaze u okviru *feedback* direktorijuma u aplikaciji.
- Ponoviti scenario B.

Zašto i dalje nema trajnog čuvanja podataka?

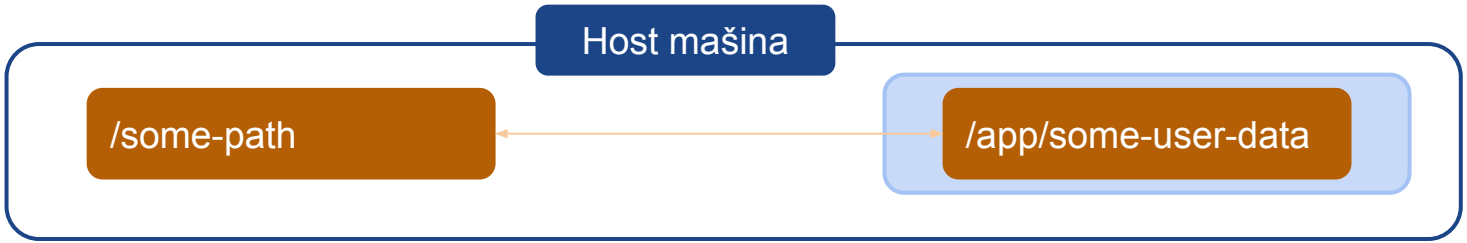
Docker skladišta podataka

- Skladišta podataka u dockeru se dele na:
  - docker skladišta podataka (engl. *volumes*)
  - skladišne tačke uvezivanja (engl. *bind mounts*)
  - privremene tačke uvezivanja (engl. *tmpfs mounts*) - samo za Linux



Docker Volumes

- Docker skladišta podataka (engl. *volumes*) su direktorijumi na mašini domaćina koji su mount-ovani (“omogućeni”, mapirani) na kontejnere
- Podaci koji su skladišteni u docker skladištu ostaju sačuvani i nakon što se kontejner ugasi.
    - Ako se konejner restartuje i mount-uje docker skladišta (volume), svi podaci koji se nalaze unutar docker skladišta postaju dostupni kontejneru.
    - Kontejner može da piše i da čita podatke iz docker skladišta.



Docker skladišta podataka

- Docker skladišta podataka (engl. *volumes*) se dele na:
  - **anonimna**
    - docker dodeljuje ime ovakvom skladištu podataka (nasumični karakteri)
  - **imenovana**
    - programer dodeljuje ime ovakvom skladištu podataka
    - smeštanje podataka koje ne želimo da pregledamo niti da menjamo direktno sa host mašine
      - tačna lokacija na host mašini poznata je samo dockeru jer nije namenjeno da programer pristupi tim podacima na host mašini
- Docker postavi direktorijum / putanju negde na host mašini.
  - tačna lokacija je nepoznata programeru.

Anonimno docker skladište podataka

- Anonimno docker skladište podataka se definiše
  - u Dockerfile-u naredbom `VOLUME ["<putanja>"]` ili `VOLUME <putanja>`
  - prilikom pokretanja kontejnera dodaje se opcija `-v <putanja>`
- Anonimnom docker skladištu podataka Docker dodeljuje naziv nasumično
  - npr: ee403708b35b4b078ac2c1cf91518186f469b9fabae32346683c0a106c33ba1d
- Osobine anonimnih docker skladišta podataka
  - ako kontejner pokrenemo sa opcijom `--rm`
    - skladište podataka nestaje kada se kontejner zaustavi
  - ako kontejner pokrenemo bez opcije `--rm`, ali obrišemo kontejner nakon stopiranja
    - skladište podataka ostaje sačuvano, međutim naredno pokretanje kontejnera kreira novo anonimno skladište podataka koje ne sadrži podatke prethodnog skladišta podataka

Imenovano docker skladište podataka

- Imenovano docker skladište podataka (engl. *named volume*) se kreira dodavanjem opcije tokom pokretanja kontejnera
  - naredba `docker run -v <naziv>:<putanja> <IMAGE>`
    - `<naziv>` predstavlja ime skladišta podataka koga kreiramo
    - `<putanja>` predstavlja putanju direktorijuma u okviru kontejnera koji će se mapirati na host mašinu
      - sve što stavim unutar datog direktorijuma u okviru kontejnera i biće sačuvano i nakon brisanja kontejnera

Pokazni primer [*feedback-app*]

- Pokrenuti kontejner sa imenovanim docker skladištem podataka.
- Obezbediti trajno čuvanja tekstualnih fajlovi koji se nalaze u okviru *feedback* direktorijuma u aplikaciji.
- Ponoviti scenario B.

Da li se sada trajno čuvani podaci mogu ponovo učitati u aplikaciju i nakon brisanja kontejnera?

Pokazni primer [*feedback-app*]

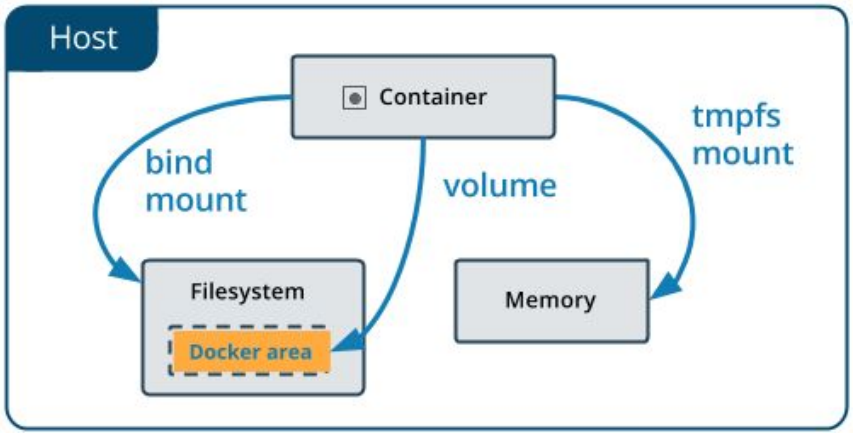
- Obrisati imenovano docker skladište kreirano u prethodnom zadatku.

Docker klijent [volume]

- `docker volume ls`
  - izlistava sva skladišta podataka
- `docker volume create <naziv>`
  - kreira imenovano skladište podataka sa prosleđenim nazivom
    - ova naredba nije nužna, jer ako prilikom pokretanja kontejnera navedeno skladište ne postoji, Docker će ga kreirati
- `docker volume inspect <naziv>`
  - prikazuje informacije vezane za dato docker skladište podataka
- `docker volume prune`
  - briše sva nekorištena skladišta podataka
- `docker volume rm <naziv>`
  - briše skladište podataka sa prosleđenim nazivom

Docker skladišta podataka

- Skladišta podataka u dockeru se dele na:
  - docker skladišta podataka (engl. *volumes*)
  - skladišne tačke uvezivanja (engl. *bind mounts*)
  - privremene tačke uvezivanja (engl. *tmpfs mounts*) - samo za Linux



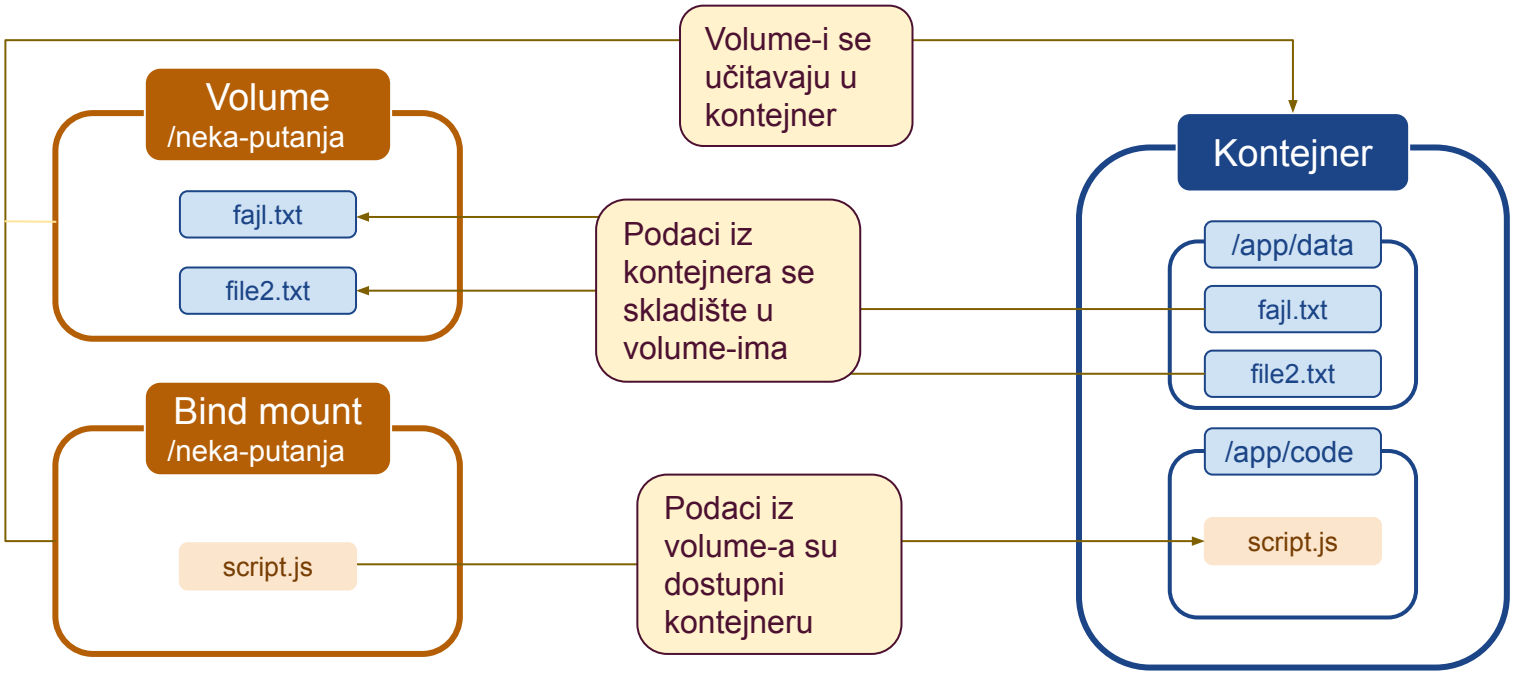
Skladišne tačke uvezivanja

- Skladišna tačka uvezivanja (engl. *bind mount)* je putanja na host mašini koja se mapira na skladište podataka iz kontejnera
- Namenjeno perzistentnim podacima koji mogu da se menjaju vremenom
- `docker run -v <host_putanja>:<kontejner_putanja> <slika>`
  - `<host_putanja>` je putanja na host mašini za kreiranje skladišne tačke uvezivanja
    - mora biti apsolutna putanja
    - skraćenica za Linux:
      - `-v $(pwd):<kontejner_putanja>`
  - `<kontejner_putanja>` je putanja u okviru kontejnera na koju se mapira `<host_putanja>`
  - preporuka je da se koriste dupli navodnici kako bi se izbegle potrencijalne greške
    - `docker run -v “<host_putanja>:<kontejner_putanja>” <slika>`

Pokazni primer [*feedback-app*]

- Napraviti skladišnu tačku uvezivanja koja obezbeđuje da se tekstualni fajlovi koji se čuvaju u feedback direktorijumu u okviru kontejnera, mount-uju i u feedback direktorijum u okviru aplikacije na host mašini.
- Napraviti novi tekstualni fajl upotrebom aplikacije.
- Na host mašini izmeniti naziv datom fajlu, a zatim testirati na koji način reaguje aplikacija na datu promenu.

Odnos kontejnera i skladišta podataka



Pokazni primer [*feedback-app*]

- Da li može da se iskoristi bind mount kako bi se izbegla situacija da svaka izmena koda aplikacije zateva ponovno kreiranje slike kontejnera?

Primer aplikacije [*feedback-app*]

- Šta će se desiti ako omogućimo da se ceo direktorijum projekta na host mašini podesi kao skladišna tačka uvezivanja na kontejner kako bismo mogli uživo da menjamo kod bez da je potrebno sa svakom izmenom koda kreirati novu sliku kontejnera?
  - Pojaviće se problem zbog toga što na putanji koju želimo da koristimo kao skladišnu tačku uvezivanja ne postoji direktorijum node_modules (nastao pokretanjem *npm install* naredbe).
    - direktorijum node_modules je nastao pokretanjem naredbe `RUN npm install` u Dockerfile-u i on je upisan kao jedan sloj slike kontejnera
    - kreiranjem skladišne tačke uvezivanja sve što se nalazi unutar datog direktorijuma na navedenoj host putanji pregaziće sve što se nalazi na putanji unutar kontejner putanje koju smo naveli

Primer aplikacije [*feedback-app*]

- Šta će se desiti ako omogućimo da se ceo direktorijum projekta na host mašini podesi kao skladišna tačka uvezivanja na kontejner kako bismo mogli uživo da menjamo kod bez da je potrebno sa svakom izmenom koda kreirati novu sliku kontejnera?
  - Pojaviće se problem zbog toga što na putanji koju želimo da koristimo kao skladišnu tačku uvezivanja ne postoji direktorijum node_modules (nastao pokretanjem *npm install* naredbe).
    - direktorijum node_modules je nastao pokretanjem naredbe `RUN npm install` u Dockerfile-u i on je upisan kao jedan sloj slike kontejnera
    - kreiranjem skladišne tačke uvezivanja sve što se nalazi unutar datog direktorijuma na navedenoj host putanji pregaziće sve što se nalazi na putanji unutar kontejner putanje koju smo naveli
      - Na koji način se može ovaj problem rešiti?

Kombinovanje različitih tipova skladišta podataka

- Prilikom definisanja skladišta podataka može da pojaviti konflikt
  - primer: dva različita skladišta sa host-a se mapiraju na istu putanju (ili putanju koja je dete date putanje) u kontejneru
    - `-v "/home/helena/Desktop/data-volumes-01-starting-setup":/app` (bind mount)
    - `-v /app/node_modules` (anonimni volume)
    - `-v feedback:/app/feedback` (imenovani volume)
- Konflikt se razrešava tako što duža putanja pobeđuje
  - Primer: Bind mount će pregaziti sve na putanji `/app` osim onog što je na `/app/node_modules` i `/app/feedback`
- U ovakvim situacijama definisati anonimne volume-e kao deo naredbe docker run, a ne u Dockerfile-u

Pokazni primer [*feedback-app*]

- Koliko je bezbedno da dozvolimo da se iz kontejnera mogu menjati fajlovi na host mašini?

Read-only docker skladišta podataka

- Omogućeno je podešavanje read-only docker skladišta podataka kako bi se onemogućilo da kontejner menja fajlove na host mašini koji su mu dostupni preko skladišne tačke uvezivanja
    - naredba `docker run -v <host_putanja>:<kontejner_putanja>:ro <slika>`

Pokazni primer [*feedback-app*]

- Napraviti da je skladišna tačka uvezivanja read-only u primeru feedback-app kontejnera?

Pokazni primer [*feedback-app*]

- Napraviti da je skladišna tačka uvezivanja read-only u primeru feedback-app kontejnera?

Koji problem je nastao sada?

Pokazni primer [*feedback-app*]

- Koji problem nastaje ako napravimo da je skladišna tačka uvezivanja read-only u primeru feedback-app kontejnera?

Primer aplikacije [*feedback-app*]

- Koji problem nastaje ako napravimo da je skladišna tačka uvezivanja read-only u primeru feedback-app kontejnera?
    - Nije više moguće kreiranje tekstualnih fajlova i njihovo smeštanje u feedback ili temp direktorijum jer je podešen read-only režim rada.

Primer aplikacije [*feedback-app*]

- Koji problem nastaje ako napravimo da je skladišna tačka uvezivanja read-only u primeru feedback-app kontejnera?
  - Nije više moguće kreiranje tekstualnih fajlova i njihovo smeštanje u temp direktorijum jer je podešen read-only režim rada.
    - Kako se dati problem može rešiti tako da skladišna tačka uvezivanja i dalje bude read-only ali da se omogući kreiranje i smeštanje fajlova u temp i feedback direktorijum?

Primer aplikacije [*feedback-app*]

- Koji problem nastaje ako napravimo da je skladišna tačka uvezivanja read-only u primeru feedback-app kontejnera?
  - Nije više moguće kreiranje tekstualnih fajlova i njihovo smeštanje u temp direktorijum jer je podešen read-only režim rada.
    - Kako se dati problem može rešiti tako da skladišna tačka uvezivanja i dalje bude read-only ali da se omogući kreiranje i smeštanje fajlova u temp i feedback direktorijum?
      - kreiranjem anonimnog docker skladišta podataka sa dužim putanjom

Pokazni primer [*feedback-app*]

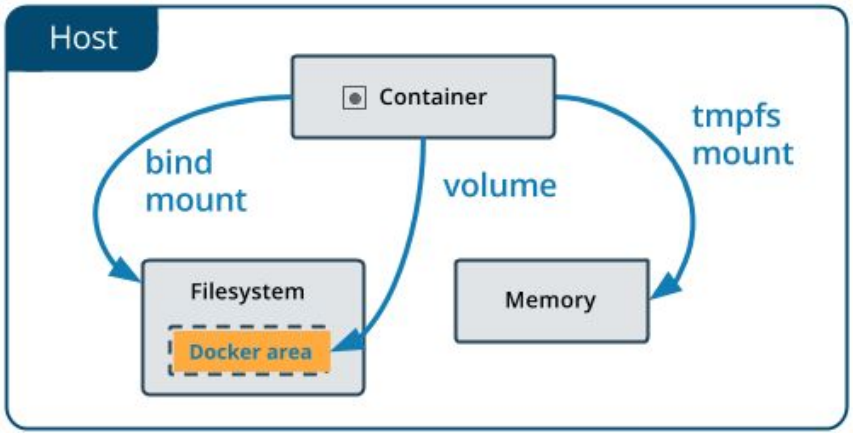
- Zašto uopšte koristimo `COPY` instrukciju u Dockerfile-u ako možemo samo da kreiramo skladišnu tačku uvezivanja i na taj način dopremimo kod unutar kontejnera?

Pokazni primer [*feedback-app*]

- Zašto uopšte koristimo `COPY` instrukciju u Dockerfile-u ako možemo samo da kreiramo skladišnu tačku uvezivanja i na taj način dopremimo kod unutar kontejnera?
  - Za potrebe aplikacije u produkciji ne želimo da imamo skladišnu tačku uvezivanja, jer ne želimo u produkciji da menjamo postojeći kod.
  - Za potrebe aplikacije u produkciji želimo *snapshot* koda aplikacije.
  - Dok razvijamo aplikaciju (kako ne bismo morali da zbog svake izmene koda kreiramo novu sliku kontejnera) koristimo skladišnu tačku uvezivanja i `COPY` naredba nam nije potrebna
  - Kada se aplikacija pusti u produkciju, kopiramo kod putem `COPY` naredba, pravimo snapshot naše aplikacije, a skladišna tačka uvezivanja nam nije potrebna.

Docker skladišta podataka

- Skladišta podataka u dockeru se dele na:
  - docker skladišta podataka (engl. *volumes*)
  - skladišne tačke uvezivanja (engl. *bind mounts*)
  - privremene tačke uvezivanja (engl. *tmpfs mounts*) - samo za Linux



tmpfs mount

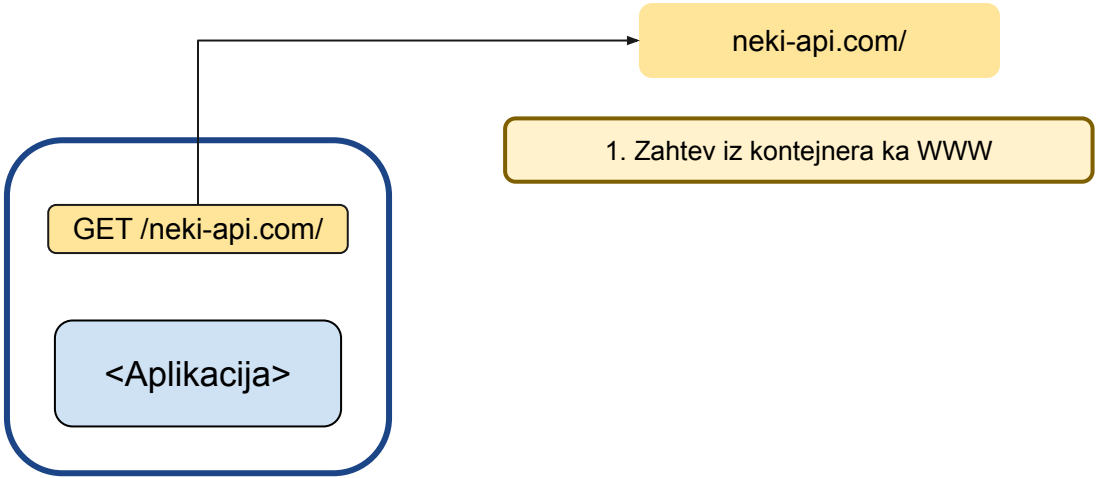
- tmpfs (engl. *temporary file system*) mount
  - omogućena samo Linux korisnicima
- Kontejner čuva podatke izvan kontejnera
  - NE čuva u fajl sistemu host mašine
  - Čuva privremeno
    - Kada se kontejner stopira tmpfs mount nestaje
    - Ne može da se deli između više kontejnera
- Pogodno je za osetljive podatke koji ne treba da se čuvaju ni na host-u ni u kontejneru
- Više informacija na linku: https://docs.docker.com/storage/tmpfs/

Umrežavanje kontejnera u Docker-u

Umrežavanje kontejnera

- Postoje tri različite kategorije umrežavanja u radu sa kontejnerima
  - Komunikacija **kontejner - WorldWideWeb**
  - Komunikacija **kontejner - lokalna mašina**
  - Komunikacija **kontejner - kontejner**

Umrežavanje kontejnera

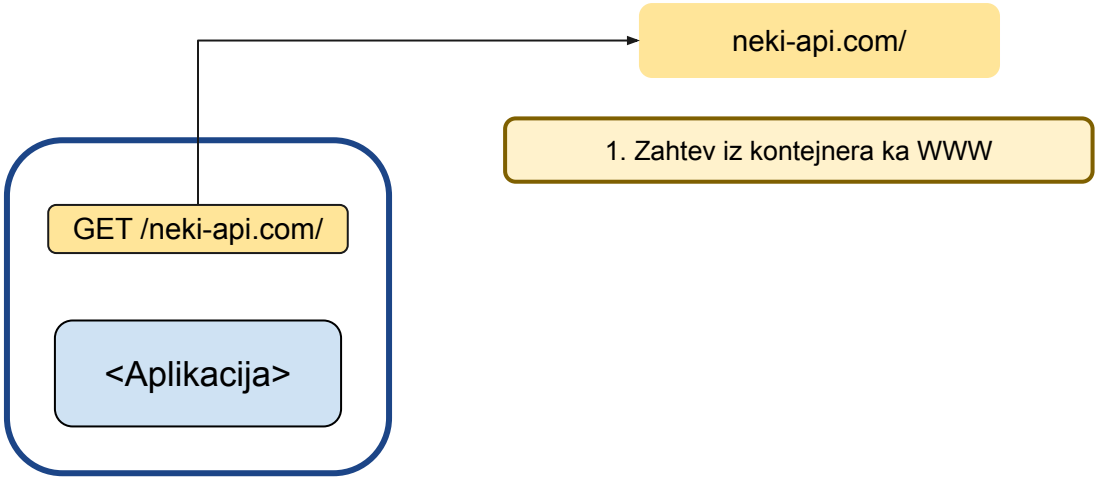


Komunikacija kontejner - *WorldWideWeb*

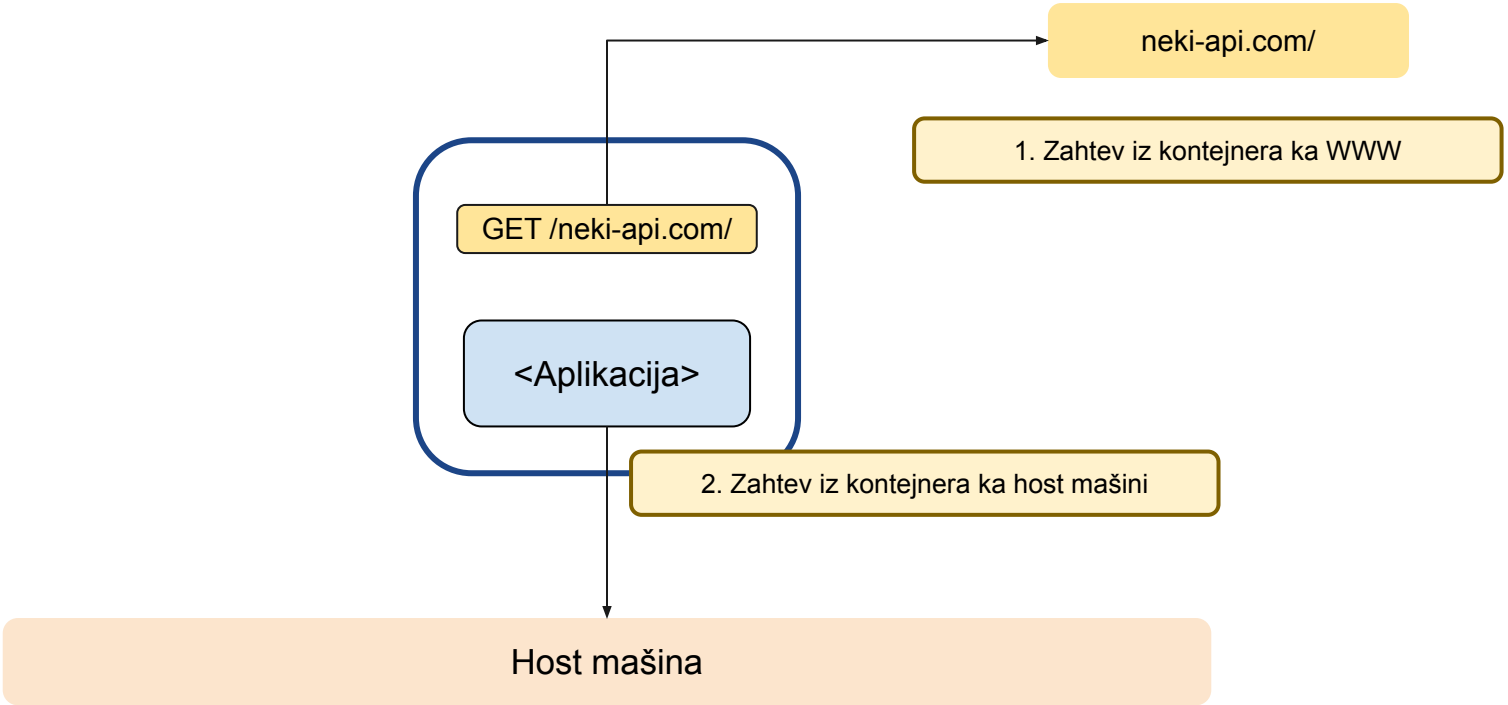
- Nije potrebno nikakvo dodatno podešavanje da bi kontejner komunicirao (slao zahteve) ka *WorldWideWeb*-u
- Primer koji pokazuje get zahtev napisan u NodeJS-u pomoću *axios* biblioteke:

```
app.get('/movies', async (req, res) => {
 try {
   const response = await axios.get('https://swapi.dev/api/films');
   res.status(200).json({ movies: response.data });
 } catch (error) {
   res.status(500).json({ message: 'Something went wrong.' });
 }
});
```

Umrežavanje kontejnera



Umrežavanje kontejnera



Pokazni primer [*favorites-app*]

- Omogućiti da aplikacija pokrenuta u kontejneru komunicira sa bazom na lokalnoj mašini.

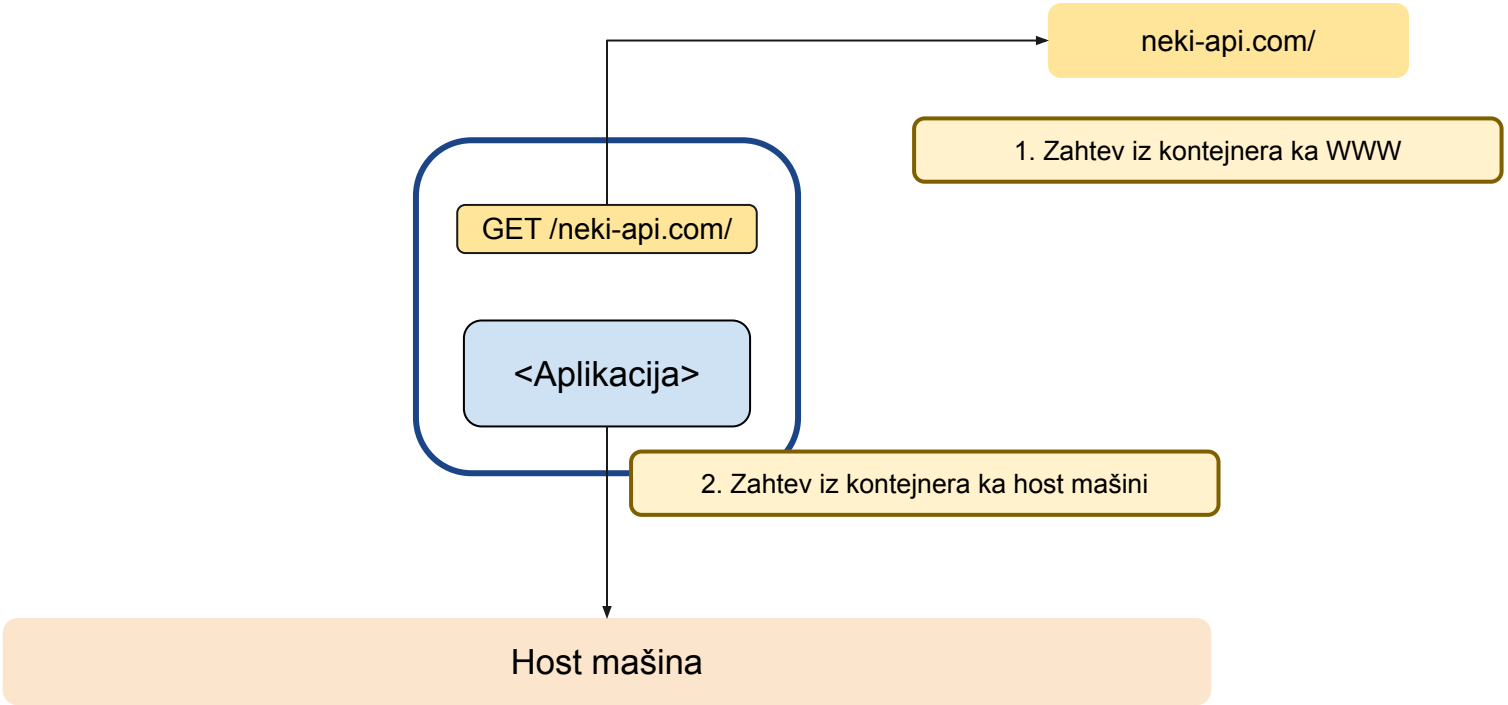
Pokazni primer [*favorites-app*]

- Omogućiti da aplikacija pokrenuta u kontejneru komunicira sa bazom na lokalnoj mašini.
  - Pokrenuti kontejner sa naredbom `docker run --network=host favorites-node`

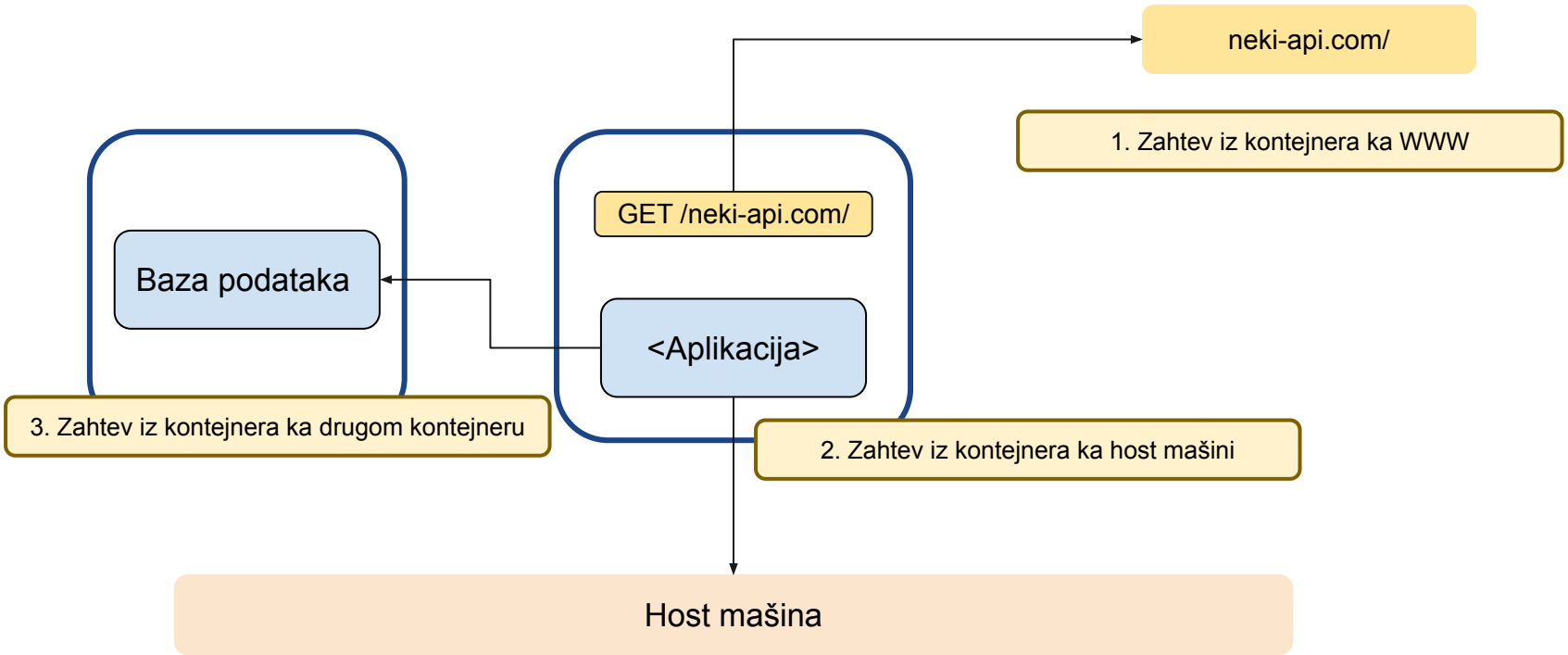
Komunikacija kontejner - lokalna mašina

- Docker kontejner je po definiciji izolovana celina od ostatak mašine na kojoj je pokrenuta, te zbog toga
  - Docker kontejner ima i izolovani mrežni namespace od ostatka mašine
    - Aplikacije unutar kontejner ne vide mrežni interfejs lokalne mašine, već samo virtuelni koji je kreiran za njih
- Kako bi se obezbedila komunikacija između kontejnera i nekog servisa (npr. baze podataka) koji se nalazi na lokalnoj mašini potrebno je:
  - pokrenuti kontejner sa dodatnom opcijom: `docker run --network=host <image_name>`
    - mreža kontejnera više nije izolovana od mreže host računara (koriste isti namespace)
      - Nema potrebe raditi mapiranje portova pomoću parametra ***-p*** prilikom pokretanja kontejnera
        - Dobije se sledeće upozorenje: WARNING: Published ports are discarded when using host network mode
  - zameniti `localhost` sa `host.docker.internal` prilikom komunikacije sa lokalnom mašinom
    - Ova opcija sigurno radi na Mac i Windows OS-u
      - Od Docker 20.10. verzije radi i na Linux-u uz dodatak opcije `--add-host=host.docker.internal:host-gateway` prilikom pokretanja kontejnera

Umrežavanje kontejnera



Umrežavanje kontejnera



Pokazni primer [*favorites-app*]

- Omogućiti da aplikacija pokrenuta u kontejneru komunicira sa bazom koja je pokrenuta u zasebnom kontejneru.

Pokazni primer [*favorites-app*]

- Primer direktne komunikacije
  - Za komunikaciju između *node* aplikacije i *mongo* baze potrebno je
    - proveriti na kojoj IP adresi se nalazi pokrenuti *mongo* kontejner (slika 1)
      - naredba `docker inspect mongodb`
    - Definisati datu IP adresu prilikom kreiranja konekcije za bazu u *node* aplikaciji (slika 2)

Slika 1

Slika 2

Primer aplikacije [*favorites-app*]

- Primer mreže kontejnera
  - Za komunikaciju između *node* aplikacije i *mongo* baze potrebno je
    - kreirati novu mrežu naredbom `docker network create favorites-net`
    - pokrenuti *mongo* bazu u novo kreiranoj mreži `docker run --name mongodb --network favorites-net mongo`
    - u *node* aplikaciji zameniti IP adresu sa nazivom pokrenutog *mongo* kontejnera (slika 1)
    - pokrenuti *node* aplikaciju u istoj mreži `docker run --name favorites-app --network favorites-net favorites:node`

```
70  mongoose.connect(
71    'mongodb://mongodb:27017/swfavorites',
72    { useNewUrlParser: true },
73    (err) => {
74      if (err) {
75        console.log(err);
76      } else {
77        app.listen(3000);
78      }
79    }
80  );
```

Slika 1

Komunikacija kontejner - kontejner

- U velikom broju situacija radi se sa više od jednog kontejnera iz dva razloga:
  - Smatra se dobrom praksom razdvojiti kontejnere tako da svaki izvršava jedan zadatak
  - Teško je konfigurisati kontejner tako da radi sa više zadataka od jednom (npr. Izvršava i veb aplikaciju i bazu podataka)
- Komunikacija između dva kontejnera se može obezbediti na dva načina:
  - Direktno (hard-kodiranjem IP adresa)
  - Kreiranjem mreže kontejnera (mreže)

Komunikacija kontejner - kontejner

- Direktno (hard-kodiranjem IP adresa)
  - Programer je zadužen da obezbedi da kontejner kontaktira drugi preko tačne IP adrese
    - Naredba `docker container inspect <container_id>`
      - omogućava pregled informacija o kontejneru među kojima je i IP adresa datog kontejnera
    - Mane ovakvog pristupa: IP adresa je nepredvidiva jer je dodeljuje Docker prilikom pokretanja kontejnera
- Kreiranjem mreže kontejnera
  - Svi kontejneri u okviru jedne mreže mogu da komuniciraju jedan sa drugim
  - Mreža se kreira naredbom: `docker network create <naziv_mreze>`
    - Docker neće sam kreirati mrežu koja ne postoji kao što kreira *volume* ako ne postoji
    - naredba `docker network ls` izlistava sve postojeće mreže
  - Prilikom pokretanja kontejnera potrebno je dodati dati kontejner u kreiranu mrežu naredbom `docker run --network <naziv_mreze> <naziv_slike>`
  - Podesiti da kontejner koji šalje zahtev ka drugom kontejner kao IP adresu koristi naziv kontejnera koji se nalazi u istoj mreži i sa kojim želi da komunicira

Zadatak

1. Obezbediti direktnu komunikaciju između Django aplikacije i Postgres baze podataka
2. Obezbediti komunikaciju Django aplikacije i Postgres baze kreiranjem mreže
3. Obezbediti da se može menjati kod Django aplikacije bez potrebe za ponovnim kreiranjem slike
4. Obezbediti da se iz kontejnera ne može pristupiti ni jednom drugom delu projekta na host mašini osim direktorijuma u koji se upload-uju slike

Zadatak

- Pomoć za rešavanje zadatka:
  - Koraci za pokretanje Django aplikacije su sledeći:
    1. Obezbediti python verziju 3.10
    2. Izvršiti instalaciju potrebnih biblioteka naredbom `pip install -r requirements.txt`
    3. Primeniti migracije na bazu podataka naredbom `python3 manage.py migrate`
    4. Pokrenuti Django aplikaciju naredbom `python3 manage.py runserver 0.0.0.0:8081`
  - Naredba za pokretanje kontejnera u kome se nalazi postgres baza
    - `docker run -e POSTGRES_PASSWORD=<password> -d postgres`
      - HINT: na putanji mysite/settings.py unutar Django projekta se nalaze podešavanja za konekciju ka bazi podataka
  - Na putanji meetups/templates/meetups/index.html promeniti tekst unutar h2 tag-a kako bi se proverila tačnost zadatka
  - Direktorijum u koji se upload-uju slike se nalazi na putanji uploads/images unutar Django projekta

Materijali:

- https://docs.docker.com/engine/reference/
- https://www.igordejanovic.net/courses/tech/docker/