



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

# Računarstvo u oblaku

ms Helena Anišić

Zimski semester 2022/2023.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi

# Podaci 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

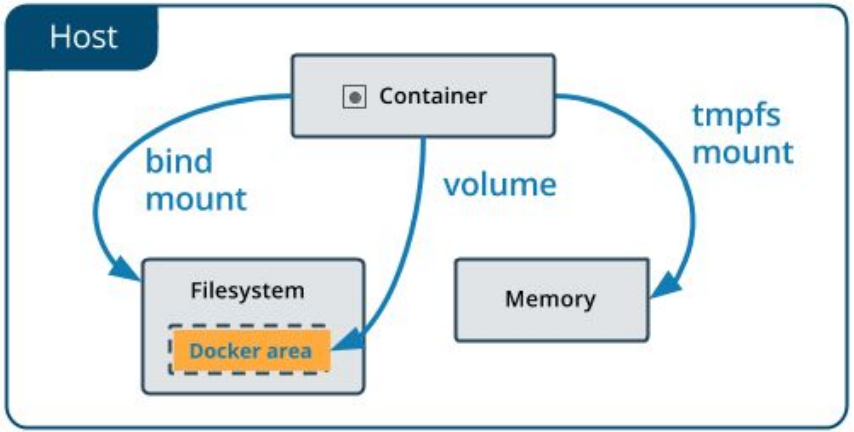
# Odnos podataka i kontejnera

# Primer aplikacije

- Šta se dešava sa fajlom kada se kontejner samo zaustavi i ponovo pokerene?
- Šta se dešava sa fajlom kada se obriše kontejner i kreira novi na osnovu iste slike?

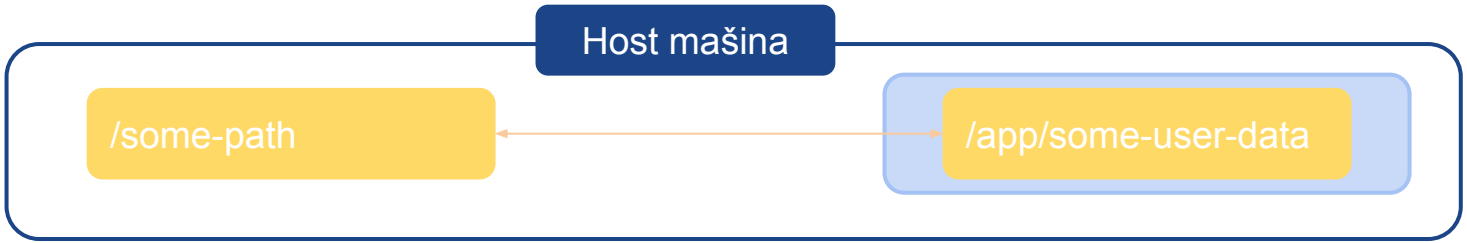
# Docker skladišta podataka

- Skladišta podataka:
  - Docker skladišta (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.



# Dockerfile instrukcija [volume]

- VOLUME instrukcija kreira mount point na definisanoj putanji u kontejneru
- Na tu putanje se mountuju fajlovi/direktorijumi sa domaćina svaki put kada se pokrene kontejner
- Može da bude JSON array ili string
    - VOLUME ["/var/log/"]
    - VOLUME /var/log

# Primer aplikacije

- Šta se desi ako podesimo volume preko Dockerfile-a?

# Anonimni docker volume

- Anonimni docker volume definišu se
  - U Dockerfile-u naredbom **VOLUME <putanja>**
  - Prilikom pokretanja kontejnera opcijom **-v <putanja>**
- Anonimnom docker volume-u Docker dodeljuje naziv
  - Npr: ee403708b35b4b078ac2c1cf91518186f469b9fabae32346683c0a106c33ba1d
- Osobine anonimnih docker volume-a
  - Ako kontejner pokrenemo sa opcijom --rm
    - Volume nestaje kada se kontejner stopira
  - Ako kontejner pokrenemo bez opcije --rm, ali obrišemo kontejner nakon stopiranja
    - Volume ostaje sačuvan, međutim naredno pokretanje kontejnera kreira novi anonimni volume koji ne sadrži podatke prethodnog volume-a

# Docker Volumes

- Docker volumes
  - Anonimni
    - Docker dodeljuje ime ovakvom volume-u (nasumični karakteri)
  - Imenovani
    - Volume ostaje sačuvan i nakon što se kontejner obriše
    - 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.
  - naredba *docker volume* omogućava rad sa tim skladištima

# Imenovani volume

- Ne stavlja se instrukcija u Dockerfile
- Dodaje se opcija tokom pokretanja kontejnera
  - docker run ***-v <naziv>:<putanja>*** image
    - Naziv predstavlja ime volume-a koga kreiramo
    - Putanja predstavlja putanju direktorijuma u okviru kontejnera koji će se mapirati na na host mašinu i gde će nalaziti podaci za permanentno čuvanje

# Docker klijent [volume]

- ***docker volume ls***
  - Izlistava sve volume-a
- ***docker volume create <naziv>***
  - Kreira imenovani volume sa prosleđenim nazivom
    - Ova naredba nije nužna, jer ako prilikom pokretanja kontejnera navedeni volume ne postoji, Docker će ga kreirati
- ***docker volume inspect <naziv>***
  - Prikazuje informacije vezane za dati docker volume
- ***docker volume prune***
  - Briše sve nekorišćene volume-e
- ***docker volume rm <naziv>***
  - Briše prosleđeni volume

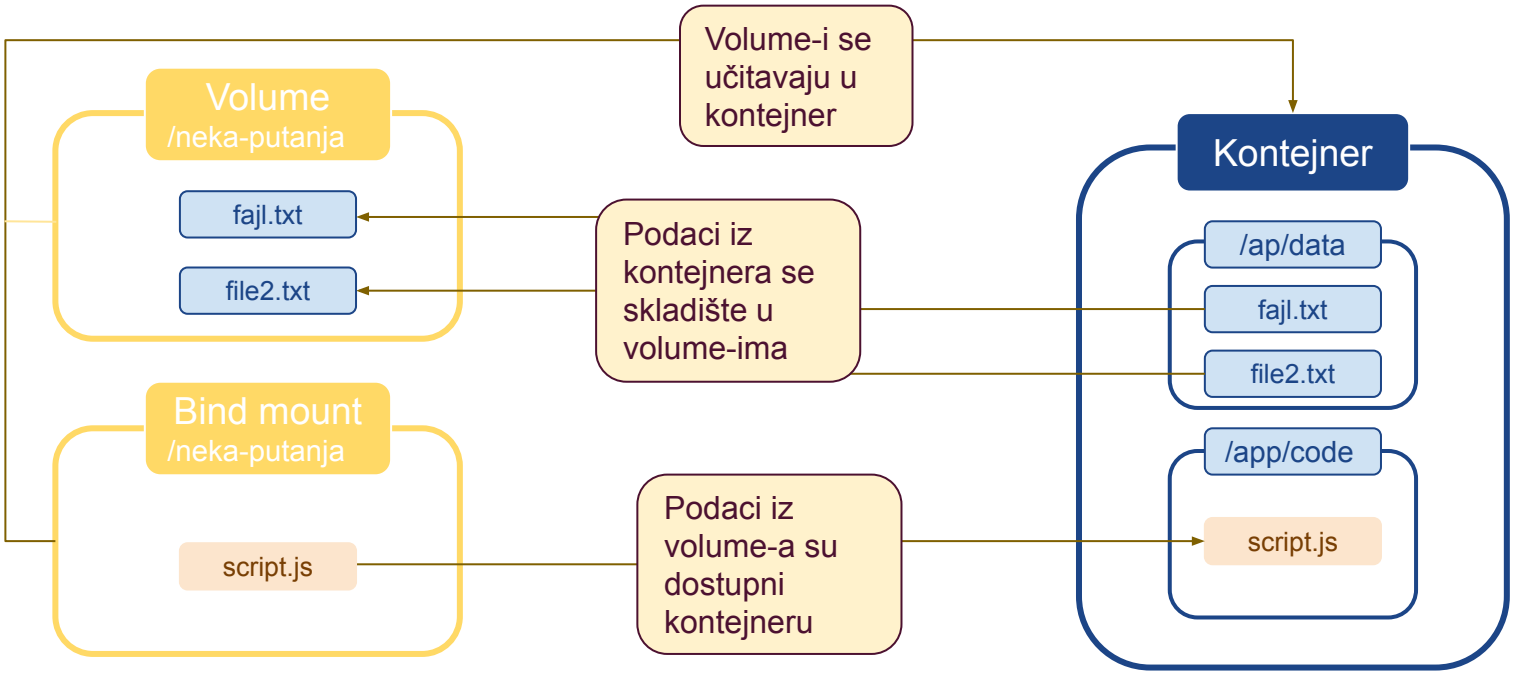
# Bind mounts

- Programer definiše putanju host mašine na koju želi da se mapira 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 bind mount-a
    - apsolutna putanja
      - Može da se stavi pod duple navodnike
    - Skraćenica za Linux:
      - -v $(pwd) : /app
  - *kontejner_putanja* je putanja u okviru kontejnera na koju se mapira host_putanja

# Primer aplikacija

- Šta će se desiti ako omogućimo da se ceo direktorijum projekta na host mašini podesi kao bind mount na kontejner ( /app )?
    - Kako bismo mogli menjati kod bez potrebe da ponovo build-ujemo sliku kontejnera

# Interakcija kontejner - volume



# Volume + Bind mount

- Prilikom definisanja skladišta podataka može da se pojavi 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

# Read-only Volumes

- Omogućeno je podešavanje read-only volume-a
  - Kako bi se onemogućilo da kontejner menja fajlove na host mašini koji su mu dostupni preko bind mount-a
  - ***-v <host_putanja>:<kontejner_putanja>:ro***

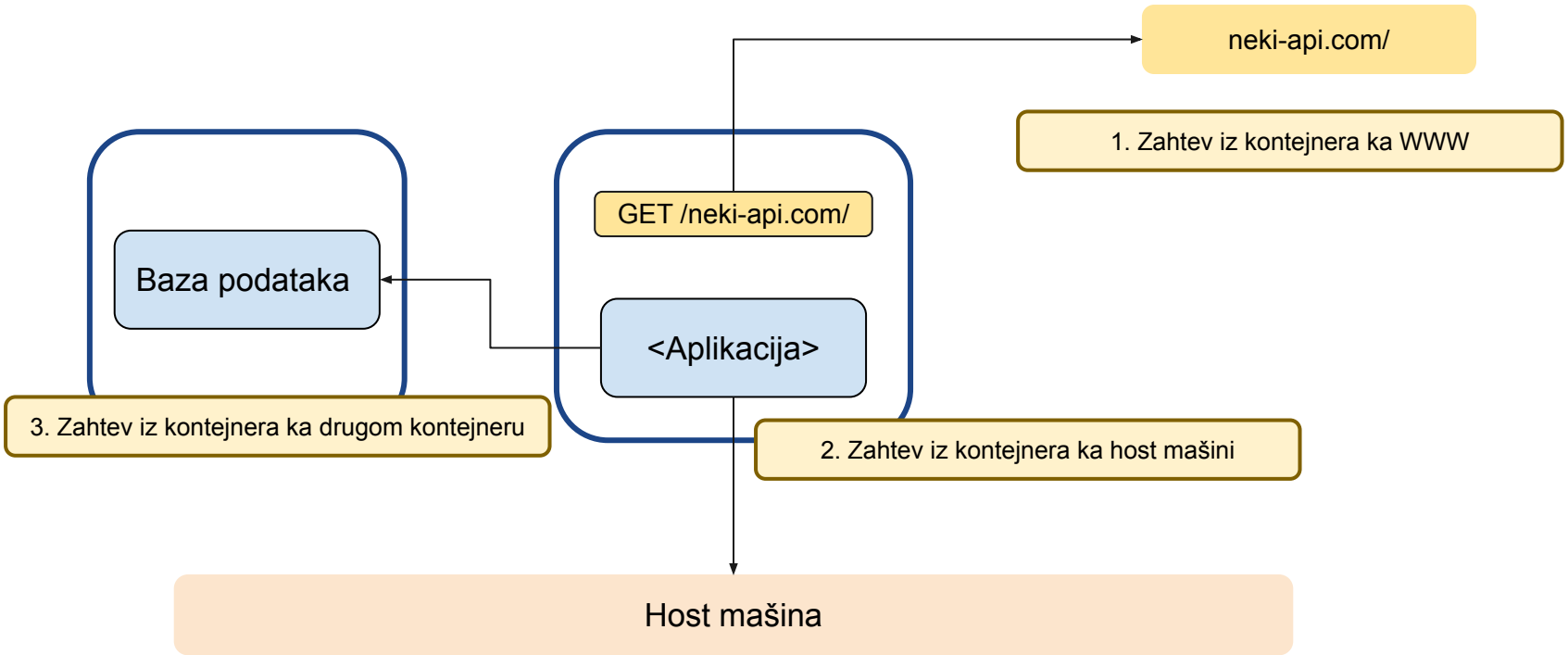
# COPY VS. Bind Mount

- Zašto koristiti COPY u Dockerfile-u za kreiranje slike kontejnera ako se može iskoristiti bind mount?

# Tmpfs mount

- tmpfs (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

# Networking


neki-api.com/
1. Zahtev iz kontejnera ka WWW
GET /neki-api.com/
Baza podataka
<Aplikacija>
3. Zahtev iz kontejnera ka drugom kontejneru
2. Zahtev iz kontejnera ka host mašini
Host mašina

# Networking

- Postoje tri različite kategorije na umrežavanja u radu sa kontejnerima
  - Komunikacija između kontejnera i WWW
  - Komunikacija između kontejnera i lokalne mašine
  - Komunikacija između dva kontejnera

# Komunikacija između kontejnera i WWW

- Nije potrebno nikakvo dodatno podešavanje da bi kontejner komunicirao (slao zahteve) ka WWW

```
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.' });
 }
});
```

# Komunikacija između kontejnera i lokalne mašine

- docker run ***--network=host*** <image_name>
  - mreža kontejnera više nije izolovana od mreže host računara (koriste isti namespace)
    - Kontejner nema svoju IP adresu
    - 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
- docker run ***--add-host host.docker.internal:host-gateway*** <image>
  - IP adresa = ***host.docker.internal***

# Primer aplikacije

- Obezbediti da Django aplikacija radi sa bazom na host mašini

# Komunikacija između dva kontejnera

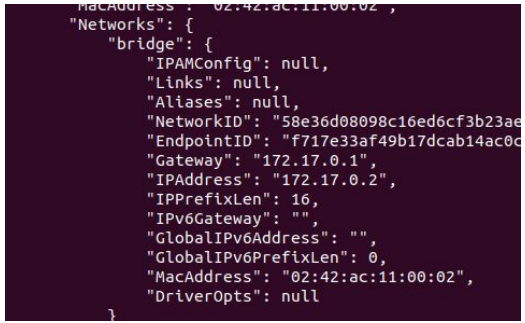
- Direktnom komunikacijom (hard-kodiranje IP adresa)
  - Programer je zadužen da obezbedi da kontejner kontaktira drugi preko tačne IP adrese
    - Naredba ***docker container inspect <container_id>***
      - Omogućava pregled informacija o kontejneru među kojima je i IP adresa datog kontejnera
  - IP adresa je nepredvidiva
    - prilikom pokretanja kontejnera Docker dodeljuje IP adresu
- Kreiranjem mreže kontejnera (mreže)
  - Svi kontejneri u okviru jedne mreže mogu da komuniciraju jedan sa drugim
    - Docker je odgovoran za razrešavanje IP adresa koje je programer ručno radio prilikom direktne komunikacije
  - Naredba ***docker run --network my_network...***

# Zadatak

- Obezbediti hard-kodiran način komunikacije između dva kontejnera
  - Django app
    - Napisati Dockerfile
      - Lakša opcija - napisati prvo Dockerfile da radi sa sqlite bazom
  - Postgres baza podataka
    - Naredba za pokretanje kontejnera sa bazom:
      - docker run -e POSTGRES_PASSWORD=<password> -d postgres

# Komunikacija između dva kontejnera

- Primer direktne komunikacije
  - Za komunikaciju između Django aplikacije i Postgres baze potrebno je
    - proveriti na kojoj IP adresi se nalazi Postgres baza (slika 1)
    - Definisati datu IP adresu prilikom kreiranja konekcije za bazu u Django aplikaciji (slika 2)




slika 1

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'postgres',
        'USER': 'postgres',
        'PASSWORD': 'admin',
        'HOST': '172.17.0.2',
        'PORT': '5432',
    }
}
```

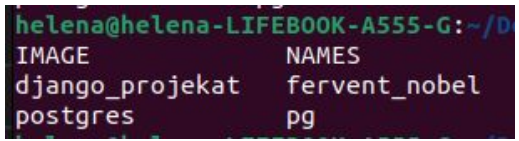
slika 2

# Komunikacija između dva kontejnera

- Mreža kontejnera
  - Naredba: ***docker network create <naziv_mreze>*** kreira novu mrežu
    - Mreža mora da se kreira pre naredbe docker run
      - Docker neće sam kreirati mrežu koja ne postoji kao što kreira volume ako ne postoji
  - Naredba: ***docker network ls*** izlistava sve postojeće mreže
  - Naredba: ***docker run --network <naziv_mreze> <naziv_slike>*** kreira kontejner i taj kontejner umrežava u mrežu koja je prosleđena parametrom --network

# Komunikacija između dva kontejnera

- Primer mreže kontejnera
  - Za komunikaciju između Django aplikacije i Postgres baze potrebno je
    - kreirati novu mrežu
    - pokrenuti Postgres bazu u novo kreiranoj mreži
    - Umesto hard-kodiranja IP adresa uneti naziv kontejnera na mesto IP adrese u Django aplikaciji
    - Pokrenuti Django aplikaciju u novo kreiranoj mreži




```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'postgres',
        'USER': 'postgres',
        'PASSWORD': 'admin',
        'HOST': 'pg',
        'PORT': '5432',
    }
}
```

# Zadatak

- Obezbediti da se bez build-ovanja slike može menjati kod Django aplikacije
- Obezbediti da se ne može pristupiti ni jednom drugom delu projekta na host mašini osim folderu u koji se upload-uju slike
    - Pokušati preko bind mounta i preko volume-a

# Materijali:

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