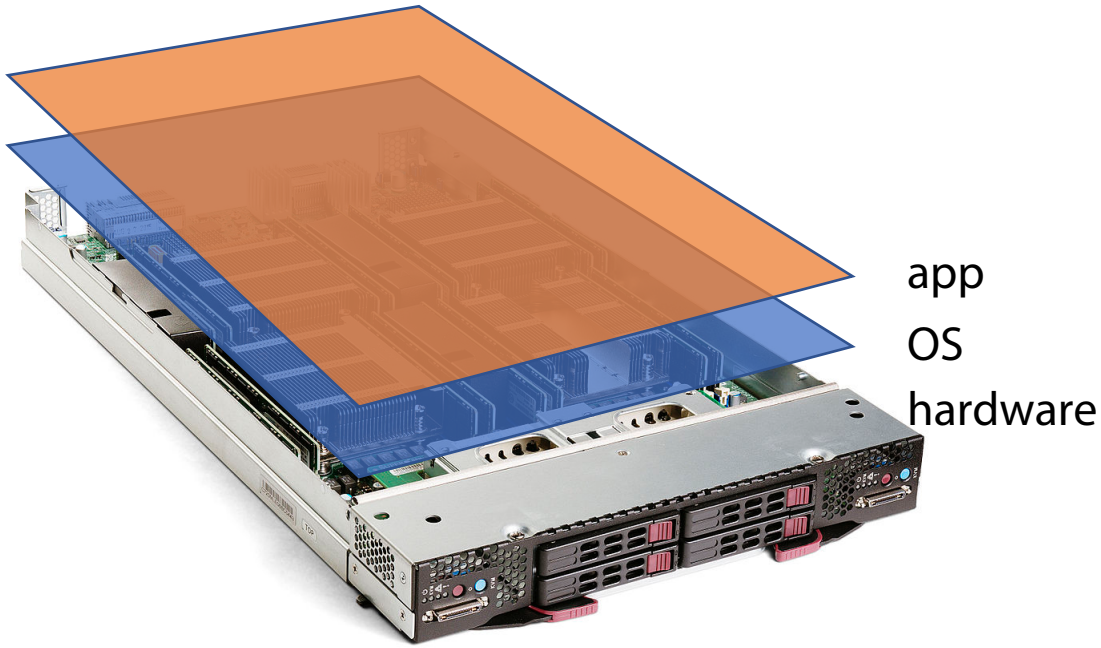
# Serveri i kontejneri

Internet softverske arhitekture

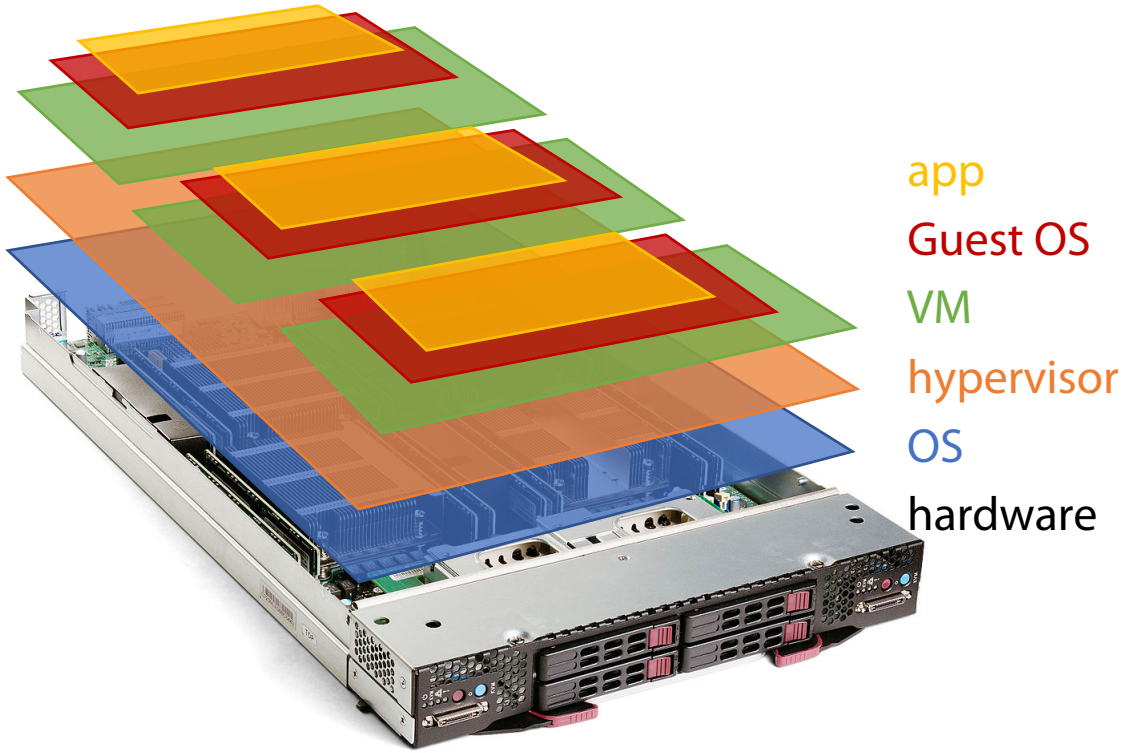
2020.

# Nekada davno: jedan server – jedna aplikacija



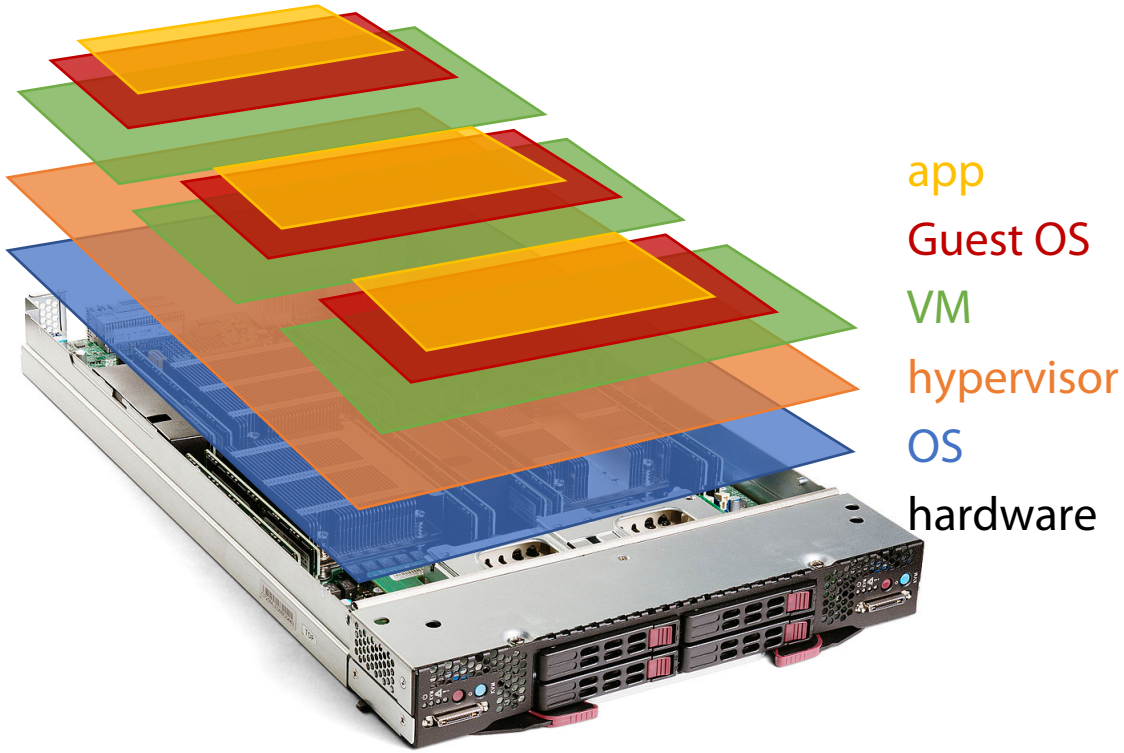

- sporo pokretanje
- veliki troškovi
- rasipanje resursa
- komplikovano skaliranje
- komplikovana migracija
- vendor lock-in

# Manje davno: virtuelne mašine




- jedan fizički server može voziti više aplikacija
- svaka aplikacija radi u virtuelnoj mašini, izolovana od drugih
- bolji resource pooling
- lakše za skaliranje
- VM mogu biti u oblaku
- pay as you go model

# Manje davno: virtuelne mašine

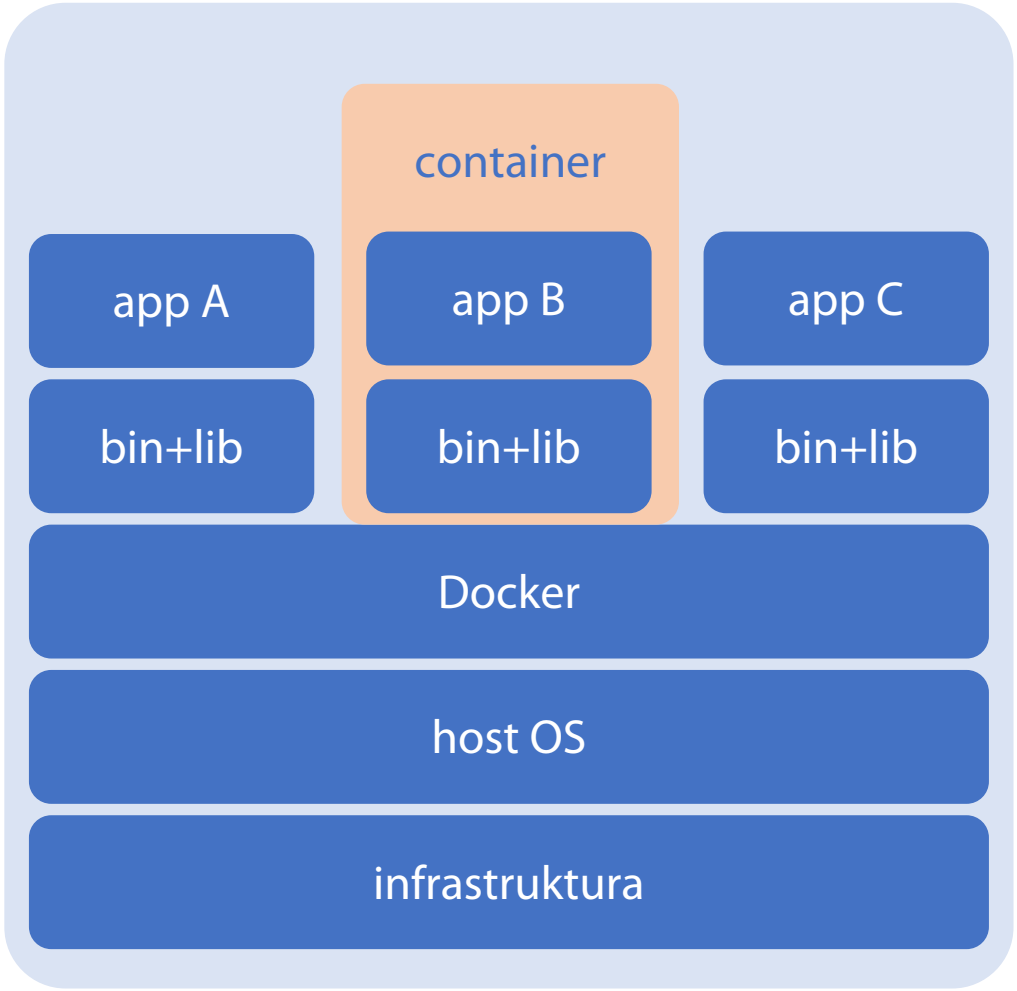



- svaka VM traži
  - svoja CPU jezgra
  - svoju RAM memoriju
  - svoj prostor na disku
  - ceo guest OS
- isti guest OS radi na različitim VM istovremeno - nepotrebno

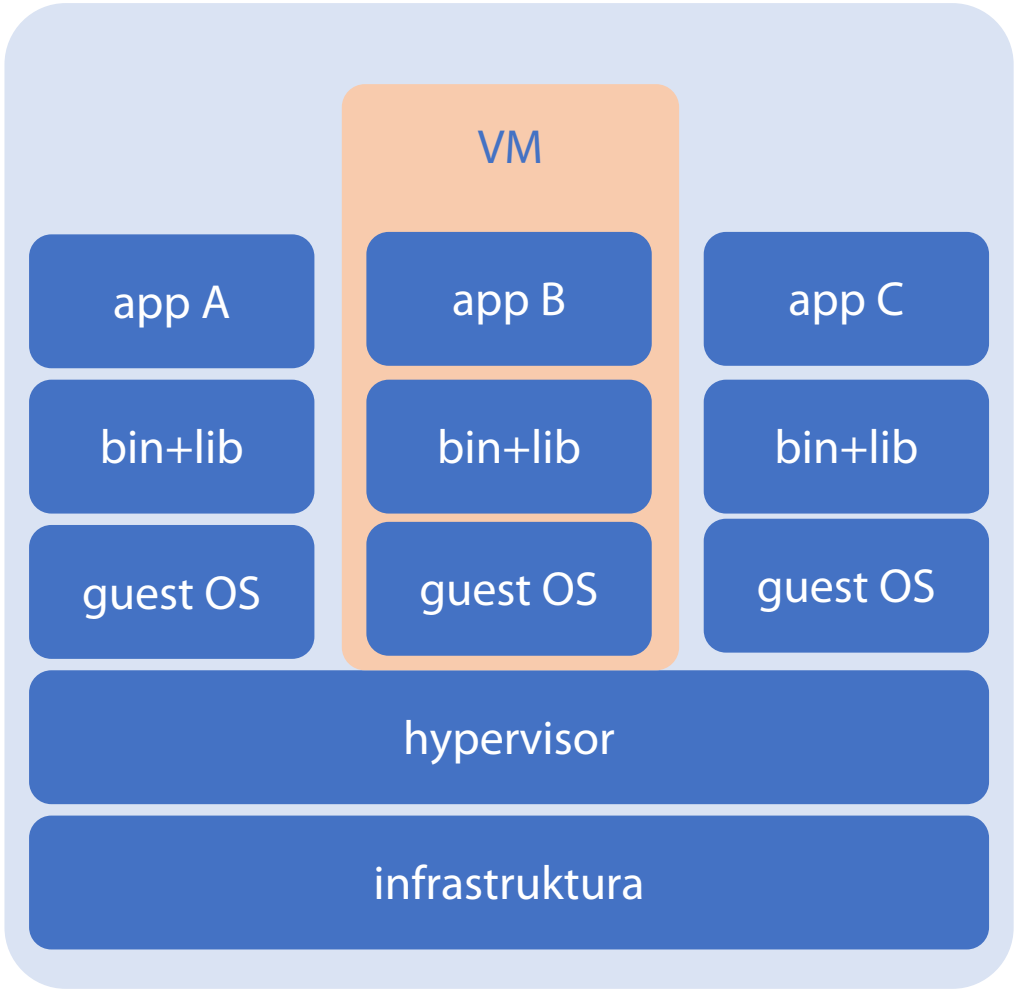
# Nedavno: kontejneri


- standardizovano pakovanje softvera i zavisnosti
- međusobna izolacija kontejnera
- deli se isti OS kernel

# VM vs kontejneri



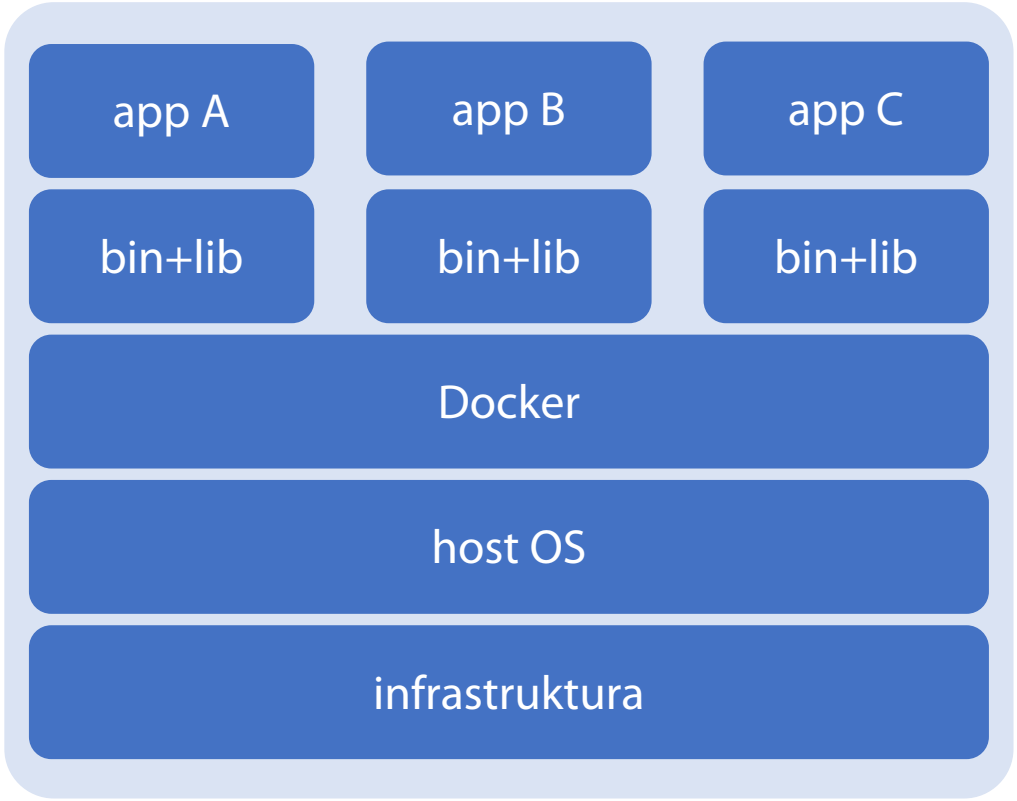

kontejner: izolacija na nivou aplikacija u okviru iste mašine



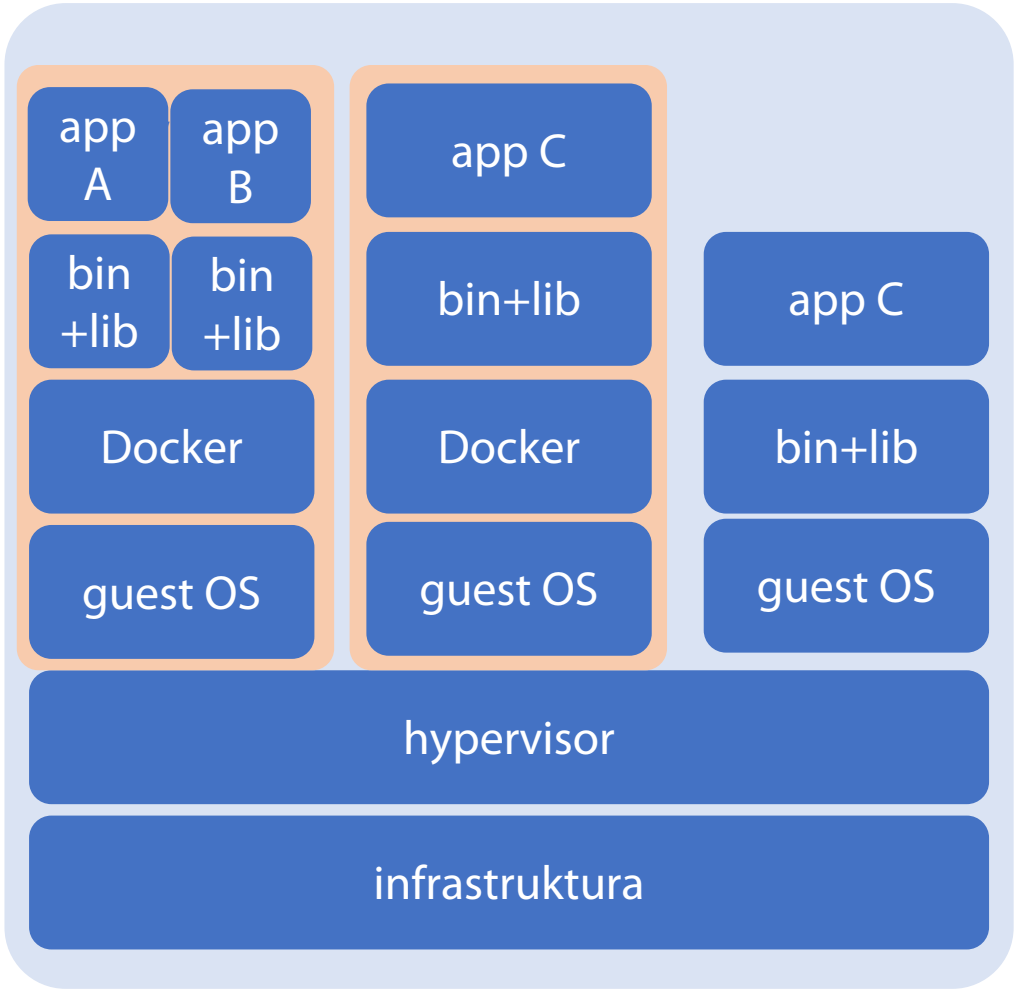

VM: izolacija više mašina na istom hardveru

# VM + kontejneri

- mogu i zajedno
- velika fleksibilnost




DEV




PROD

# Prednosti kontejnera

- brzina
  - pokretanje ne zahteva startovanje OS-a
- prenosivost
  - manje zavisnosti između slojeva
  - veća pokretljivost unutar infrastrukture
- efikasnost
  - manji OS overhead
  - povećana gustina VM

# Docker terminologija

- image
  - sadržaj (aplikacija+bin+lib) pripremljen za izvršavanje
- container
  - trčeći image
- engine
  - softver koji izvršava komande upućene kontejnerima
- registry
  - katalog, skladište, distribucija image-a
- control plane
  - upravljanje i orkestracija image-ima

# Primer: pokretanje gotovog image-a

```
$ docker run
    --name mydb                                  naziv kontejnera
    --volume ~/tmp/mydb:/var/lib/mysql  folder na hostu se mapira na folder na image-u
    --volume ~/tmp/initdb:/docker-entrypoint-initdb.d
    --env MYSQL_ROOT_PASSWORD=***          promenljive okruženja
    --env MYSQL_USER=zika
    --env MYSQL_PASSWORD=***
    --env MYSQL_DATABASE=mydatabase
    --detach                                     pokreni kontejner u pozadini
    mysql:5.7.21                                 naziv image-a u katalogu
```

# Primer: kreiranje image-a i pokretanje

Dockerfile

```
FROM openjdk:8-jdk-alpine                  počni od ovog image-a
ARG JAR_FILE=target/*.jar                  definiši argument i njegovu default vrednost
COPY ${JAR_FILE} app.jar                   kopiraj fajl sa hosta u image
ENTRYPOINT ["java","-jar","/app.jar"]      definiši program koji se pokreće sa parametrima
```

```
napravi Spring Boot aplikaciju
$ gradle build

napravi image na osnovu Dockerfile
$ docker build --build-arg JAR_FILE=build/libs/*.jar –t isa:myapp

pokreni kontejner na osnovu image-a
$ docker run --name myapp1 --detach –-expose 8080:8080 isa:myapp
```

# Primer: pokretanje više kontejnera iz istog image-a

```
$ docker run --name myapp1 --detach –-expose 8080:8080 isa:myapp
$ docker run --name myapp2 --detach –-expose 8080:8081 isa:myapp
$ docker run --name myapp3 --detach –-expose 8080:8082 isa:myapp
$ docker run --name myapp4 --detach –-expose 8080:8083 isa:myapp
$ docker run --name myapp5 --detach –-expose 8080:8084 isa:myapp
```

# Primer: start/stop kontejnera

```
napravi i pokreni
$ docker run --name myapp1 --detach –-expose 8080:8080 isa:myapp

izlistaj aktivne kontejnere
$ docker ps

zaustavi
$ docker stop myapp1

pokreni
$ docker start myapp1

zaustavi, pa obriši kontejner
$ docker stop myapp1
$ docker rm myapp1
```

# Kontejner je efemeran

- kontejner može biti zaustavljen i uklonjen
- novi se može napraviti iz image-a uz minimalan napor

- kada se kontejner ukloni, svi podaci iz njega nestaju
- trajne podatke treba čuvati izvan kontejnera
  - u fajl sistemu hosta
  - u bazi podataka
  - ...

# Dva kontejnera u saradnji

```
pokreni MySQL
$ docker run --name mydb ... --detach mysql:5.7.21

pokreni Spring Boot aplikaciju
$ docker run --name myapp1 --detach –-expose 8080:8080 --link mydb isa:myapp

drugi kontejner će videti prvog pod imenom mydb
na primer, moći će da otvori vezu prema jdbc:mysql://mydb:3306/
```

# docker-compose: konfigurisanje složenih aplikacija

docker-compose.yml

```
version: "3.3"
services:
  mydb:
    image: mysql:5.7.21
    container_name: "mydb"
    volumes:
      - /path/to/host/dir:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ****
      MYSQL_DATABASE: myappdb
      MYSQL_USER: myapp
      MYSQL_PASSWORD: ****

  myapp:
    depends_on:
      - mydb
    image: isa:myapp
    container_name: "myapp"
    restart: always
```

```
pokreni sve kontejnere
$ docker-compose up -d

zaustavi sve
$ docker-compose down
```

1) definiši kontejnere

2) pokreni sve,
u pravilnom redosledu