

Docker Compose

Instalacija, rezime naredbi i zadaci

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 (*eng. links*)
 - olakšava rad sa kontejnerima
 - *.yaml* 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

Compose YAML datoteka

Compose datoteka je *YAML* datoteka koja definiše servise, mreže i skladišta, neophodne kako bi se pokrenuo čitav ekosistem servisa.

Podrazumevana putanja do *Compose* datoteke je `./docker-compose.yml`.

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

Više informacija na: [Compose File v3 Reference](#)

Osobine

- 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 (*eng. environmental variables*)
 - za parametrizaciju podešavanja kontejnera
 - povećava stepen ponovne iskoristivosti specifikacija
 - za različita okruženja

Slučajevi korišćenja

- Slučajevi korišćenja alata *Docker Compose*
 - 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

Instalacija

- Koraci za instalaciju dostupni na: <https://docs.docker.com/compose/install/>

Rezime *cli* naredbi

```
docker-compose up      # Izgrađuje, kreira, pokreće i povezuje kontejnere
docker-compose down    # Zaustavlja kontejnere (opciono uklanja kontejnere, mreže,
                        # skladišta i slike kreirane korišćenjem naredbe up (--rmi 'all', -v...))

docker-compose start   # Pokreće postojeće kontejnere
docker-compose stop    # Zaustavlja pokrenute kontejnere bez uklanjanja
docker-compose kill     # Prisilno zaustavlja pokrenute kontejnere (force stop)
docker-compose restart # Restartuje pokrenute kontejnere

docker-compose logs    # Prikazuje logove
```

Više informacija na: <https://docs.docker.com/compose/reference/>

Zadatak 1

Korišćenjem alata ***docker-compose*** odraditi *deployment* jednostavne dinamičke *web* aplikacije koja izračunava i vrši prikaz broja poseta stranici aplikacije. Aplikacija se sastoji od dva dela: prvi deo je *web* servis napisan na *python* programskom jeziku, a drugi deo predstavlja baza podataka *Redis* koja čuva informaciju o broju pristupa sajtu.

Za pokretanje *Redis* kontejnera iskoristiti zvaničnu osnovnu sliku, uz mapiranje porta 6379 na port 6379 i uz kreiranje skladišta podataka koje će preslikavati */data* direktorijum docker kontejnera. Za pokretanje *python web* aplikacije iskoristiti dostupnu *Dockerfile* datoteku; port 80 namapirati na port 8085; ***PERSON*** **env varijablu** postaviti na proizvoljnu vrednost. Pokrenute *docker* kontejnere potrebno je povezati na *webnet* mrežu.

Zadatak 1

Dodatni zadaci:

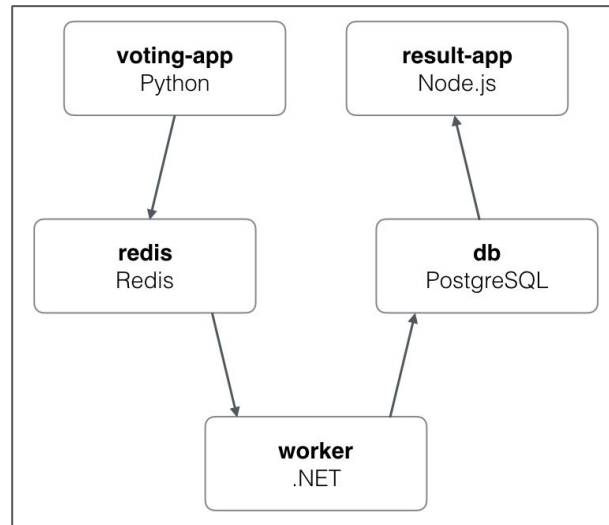
1. Izdvojiti env varijablu u zaseban fajl
2. Pokrenuti *Redis* samostalno, pa ga povezati sa *web* aplikacijom
3. Specificirati fajl prilikom pokretanja *docker-compose up* naredbe
4. Pomoću nasleđivanja specificirati drugačije mapiranje porta i drugu vrednost env varijable za produkciono okruženje u odnosu na development okruženje
5. Obrisati imenovani volume prilikom pokretanja *docker-compose down* naredbe

Napomena za rad u učionici: potrebno je dodati *proxy* podešavanja pri pokretanju *pip install* naredbe:

```
--proxy "http://192.168.18.1:8080"
```

Zadatak 2

Odraditi *deployment* aplikacije za glasanje čija je arhitektura prikazana na slici. Aplikacija se sastoji od 5 različitih mikroservisa i svaki je potrebno pokrenuti u zasebnom kontejneru. Za *voting*, *result* i *worker* serise su dostupne *Dockerfile* datoteke, dok je *Redis* i *PostgreSQL* servise potrebno pokrenuti na osnovu zvaničnih slika.



Deployment odraditi korišćenjem alata *docker-compose*.

Zadatak 2

Aplikacija se sastoji od 5 mikroservisa:

- *Python web* aplikacija koja omogućava glasanje između dve ponuđene opcije (pasa i mačaka :))
- *Redis* red pomoću kog se sakupljaju novi glasovi
- *.NET worker* koji preuzima glasove i skladišti ih u bazu podataka
- *Postgres* baza podataka podržana *Docker* skladištem
- *Node.js web* aplikacija koja omogućava prikaz rezultata glasanja u realnom vremenu

Zadatak 2

Arhitektura podrazumeva da postoje dve mreže (*network*) - mreža pozadinskog nivoa, koja služi za komunikaciju svih servisa (svi servisi bi trebalo da budu povezani na nju) i mreža prednjeg nivoa koja služi za povezivanje *vote* i *result* servisa aplikacije.

Za *Redis* servis je pri pokretanju potrebno namapirati port 6379 na port 6379; za *result* servis je potrebno namapirati port 5000 na port 80, kao i port 5858 na port 5858; za *vote* servis je potrebno namapirati port 5001 na port 80.

Za *PostgreSQL* servis je potrebno obezbediti skladište podataka koje se mapira na `/var/lib/postgresql/data` direktorijum, kao i zadati vrednosti za `POSTGRES_USER` i `POSTGRES_PASSWORD` *environment* varijable.

Za *vote* i *result* servis skladišta koja mapiraju `/app` direktorijum na *vote*, odnosno *result* direktorijum *host* operativnog sistema.

Zadatak 2 - napomena za rad u učionici

Napomene za rad u učionici - *proxy* podešavnja:

- u *vote* servisu, u *Dockerfile* pri pokretanju ***pip install*** naredbe:

```
--proxy "http://192.168.18.1:8080"
```

- u *result* servisu, u *Dockerfile* odmah nakon FROM dodati:

```
RUN npm config set proxy http://192.168.18.1:8080
```

- u *docker-compose.yml* datoteci *worker* servisu dodati dve sistemske varijable:

```
HTTP_PROXY = "http://192.168.18.1:8080"
```

```
HTTPS_PROXY = "https://192.168.18.1:8080"
```

Zadatak 2

Dodatni zadaci:

1. *Postgre* i *Redis* servisi trebalo bi da se pokrenu isključivo ukoliko je aktivan profil pod nazivom *local_db*
2. Za definisanje profila *Postgre* servisa iskoristiti *YAML* sidra i alijase
3. Za *Result* servis verziju slike proslediti kroz argument izgradnje slike

Zadatak 3

Odraditi *deploy* troslojne *web* aplikacije koja se sastoji od:

- MySQL baze podataka,
- *Java Spring boot* aplikacije i
- *Angular* aplikacije.

Pomoćni resursi:

- [Angular apps as docker containers](#)
- [Spring boot docker](#)