

Alati za razvoj softvera

Docker compose, Docker volumes



Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Uvod

- ▶ Postavlja se pitanje sta raditi ukoliko imamo više aplikacija, od kojih je neke neophodno pokrenuti u vise instanci (kontejnera), koji moraju da komuniciraju
- ▶ Tada posao pojedinačnog kreiranja slika i pokretanja kontejnera nije baš idealan
- ▶ Za ove potrebe se koristi alat *docker-compose* koji nam značajno olakšava stvari po tom pitanju
- ▶ Omogucuje nam:
 - ▶ pokretanje i zaustavljanje aplikacija
 - ▶ zajednički ispis logova svih aplikacija na jedan pseudo terminal
 - ▶ jednostavno održavanje mreže i DNS-a

- ▶ Sve što je neophodno jeste da se instalira alat
- ▶ Zatim je potrebno da kreiramo fajl pod nazivom *docker-compose.yml*
- ▶ On sadrži specifikaciju koje sve aplikacije (kontejnere) je potrebno pokrenuti, i kako organizovati mrežu itd.
- ▶ Ceo postupak nam omogućava da lako pokrenemo povezane aplikacije i da ih testiramo

Docker compose YAML

- ▶ Ovaj fajl je relativno jednostavan, i sadrži dosta direktiva koje možemo upotrebiti:
 - ▶ **version** ovde naglašavamo koju verziju formata želimo da koristimo — dovoljno je navesti verziju 3 (poslednja verzija formata)
 - ▶ **services** definiše niz objekata gde svaki predstavlja servis, odnosno kontejner
 - ▶ Obe sekcije su obavezne
 - ▶ Pored ove dve sekcije možemo definiati volumes, mreže

Services

- ▶ **services** definiše niz objekata gde svaki predstavlja servis, odnosno kontejner
- ▶ Ovaj element ima složenu strukturu, dalje unutar njega definisemo:
 - ▶ **build** ova direktiva ako je definisana, govori da je neophodno kreirati slike pri čemu se definišu odnosno putanja do direktorijuma na kojoj se nalazi Dockerfile (može . ako se nalazi na istoj lokaciji kao i Dockerfile)
 - ▶ **image** definiše naziv slika koja će nastati prilikom build-ovanja
 - ▶ **container-name** definiše naziv kontejnera koji će biti pokrenut
 - ▶ **restart** definiše pod kojim okolnostima kontejner treba restartovati
 - ▶ **networks** definiše mrežu (mreže) u kojoj kontejner treba da se nalazi
 - ▶ **ports** vrši se mapiranje portova (host:kontejner)
 - ▶ **environments** postavlja vrednost environment varijable koje se nalaze u kontejneru
 - ▶ **volume** definiše volume za koje se kontejner kači **depends_on** nam govori prilikom pokretanja servisa koje su zavisnosti, odnosno koji servisi moraju biti pokrenuti pre nego što se pokrene konkretan servis

Docker compose pokretanje

- ▶ Pokretanje i zaustavljanje servisa je relativno jednostavno
- ▶ Pozicioniramo se na putanju do direktorijuma u kojem se nalazi `docker-compose.yml`
- ▶ Pozovemo naredbu: `docker-compose up --build`
- ▶ Sa ovim pokrećemo sve nase servise (kontejnere)
- ▶ Rezultat izvršavanja `docker-compose` naredbe nam je u pseudo terminalu spojio logove sa svih pokrenutih servisa
- ▶ Pozovemo naredbu: `docker-compose down` zaustavljamo izvršavanje servisa

Docker compose YAML primer

```
version: '3'
services:
  consul:
    image: consul
    ports:
      - "8500:8500"
      - "8600:8600/tcp"
      - "8600:8600/udp"
    command: "agent -server -ui -node=server-1 -bootstrap-expect=1 -client=0.0.0.0"
    volumes:
      - ~/Desktop/consul:/consul/data
  app:
    build: .
    restart: always
    ports:
      - "8000:8000"
    depends_on:
      - consul
    environment:
      - DB=consul
      - DBPORT=8500
volumes:
  consul:
```

Napomena

- ▶ Ako je potrebno da se kontejneru pošalju nekakvi (meta)podaci, nije loša praksa da se to uradi preko env varijabli OS-a
- ▶ Ovo treba navesti prilikom startovanja
- ▶ Unutar kontejnera, možemo ih dobiti prostim naredbama programskog jezika
- ▶ Ako pogledamo prethodni primer, našem servisu smo parametre za konekciju na bazu prosledili na ovaj način

```
environment:
  - DB=consul
  - DBPORT=8500
```

- ▶ *DB* i *DBPORT* su ključevi po kojima možemo unutar kontejnera da dobijemo ove vrednosti

- ▶ Docker compose za nas u pozadini održava DNS server
- ▶ Što je bitno, zato što servisima možemo da pristupimo preko imena a ne preko adrese
- ▶ Ako pogledamo docker compose YAML fajl, naziv svakog servisa će zapravo biti njegova adresa
- ▶ Koristeći taj naziv možemo da pristupimo drugim servisima preko mreže
- ▶ Naša aplikacija se povezuje sa Consul bazom
- ▶ Adresa ove baze je ime Consul servisa (consul), a port je onaj port koji smo otvorili za komunikaciju

Uvod

- ▶ Kada smo opisivali docker slike, bilo je reci o read-only slojevima i read-write sloju
- ▶ Sve promene i sav sadržaj se upisuju u taj sloj
- ▶ Problem sa tim jeste da kada se kontejner obriše, promene će biti potpuno izgubljene
- ▶ Kontejner je read-only sistem, i sve podatke koje treba da sačuvamo, ne možemo čuvati u kontejneru
- ▶ Moramo nekako naći način da te podatke izbacimo van kontejnera i da ih nekako *zakačimo* za kontejner

- ▶ Zato je Docker uveo nov koncept pod nazivom *volumes*
- ▶ Da bi mogli da cuvamo konkretan sadrzaj, i po potrebi ga delimo
- ▶ Kreiramo poseban *volume* koji je prosto rečeno, ništa drugo do skup direktorijuma/fajlova koji se nalaze izvan UFS-a
- ▶ Ovaj skup koji naravno postoji kao direktorijumi/fajlovi na host fajlsistemu
- ▶ Vaš kontejner ima utisak kao da se ti fajlovi/direktorijumi nalaze u njegovom FS-u
- ▶ Nakon zaustavljanja kontejnera svi podaci su bezbedni, tj. neće nestati kada se kontejner ugasi
- ▶ Ovo je jako zgodno za baze podataka, i podatke koje planiramo da čuvamo u njima

Upotreba

- ▶ Kreiranje *volume-a* je moguće odraditi sa komandom

```
docker volume create naziv
```

- ▶ Mount-ovanje se radi prilikom pokretanja sa flegom `-volume` ili `-v`

```
docker run -i -t -v primer1:/nekiPodaci ubuntu /bin/bash
```

- ▶ Dakle najobičnija komanda (koju smo vec videli), prosirena flegom `-v`
- ▶ Zadali smo naziv volume-a i gde će biti izvršeno mount-ovanje u okviru samog kontejnera

- ▶ Ovaj proces moramo da uraidmo i kada koristimo docker compose
- ▶ Samo je on malo jednostavniji
- ▶ Ako pogledamo primer datog YAML fajla, vidimo da smo kreirali volumes tako da podaci iz consul-a budu *eksportovani* van kontejnera

```

...
services:
  ...
  consul:
    volumes:
      - ~/Desktop/consul:/consul/data
volumes:
  consul:

```

- ▶ Volumes nije samo zgodan za podatke baze podataka
- ▶ Možemo da koristiti i za razne konfiguracije
- ▶ Ako se naši servisi konfigurišu kroz neke konfiguracione datoteke, njih možemo da eksportujemo van kontejnera i da ih povežemo sa kontejnerom
- ▶ Na taj način smo slobodni da podešamo našistem ekternalizovano
- ▶ Možemo da menjamo konfiguracije kako nam bude potrebno
- ▶ Ova fleksibilnost je jako bitna

Dodatni materijali

- ▶ Docker compose docs
- ▶ Docker volumes docs
- ▶ Building api with golang and docker compose

Kraj predavanja

Pitanja? :)