Operativni Sistemi

II Semestar - 2022/23 - Vježbe

Sedmica 11

Handout za Vježbe

Agenda:

- Docker, instalacija na Linux
- Rad s Ngnix serverom kroz Docker s primjerima
- Lab. zadaci

Kontakt:

Narcisa.hadzajlic@size.ba (B Grupa)

Adin.jahic2019@size.ba (A Grupa)

Docker, instalacija na Linux

Docker je popularni open-source alat koji se koristi za upravljanje kontejnerima, što omogućuje brzu i jednostavnu distribuciju aplikacija u različitim okruženjima. Kontejneri su izolirane okoline u kojima aplikacije mogu raditi neovisno o drugim aplikacijama i sistemima, što ih čini pogodnim za razvoj i distribuciju softvera.

Docker datoteka, poznata i kao Dockerfile, je skripta koja opisuje kako se aplikacija ili servis može pakirati i pokrenuti u Docker kontejneru. Ona definira koji se paketi i ovisnosti (eng. dependencies) moraju instalirati, koje datoteke treba kopirati u kontejner, te koje naredbe treba pokrenuti kako bi se aplikacija pokrenula.

Instalacija Docker-a na Linux je relativno jednostavna i može se obaviti u nekoliko koraka.

1. Provjera zahtjeva sistema

Prvo, provjerite jesu li vaš Linux sistem i kernel kompatibilni s Docker-om:

```
$ uname -r
```

Ako se izlaz naredbe podudara s nekom od podržanih verzija kernela navedenih u dokumentaciji za Docker, možete nastaviti s instalacijom.

2. Instalacija preduvjeta

Prije instalacije Docker-a, potrebno je instalirati preduvjete. Ovisno o vašoj distribuciji Linux-a, postupak instalacije preduvjeta se može razlikovati. Na primjer, za Ubuntu:

```
$ sudo apt-get update

$ sudo apt-get install -y apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

3. Instalacija Docker-a

Nakon instalacije preduvjeta, možete nastaviti s instalacijom Docker-a. Za to, dodajte Docker GPG ključ:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Dodajte Docker APT repozitorij:

```
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

Ažurirajte repozitorij:

```
$ sudo apt-get update
```

Instalirajte Docker:

```
$ sudo apt-get install -y docker-ce docker-ce-cli containerd.io
```

4. Provjera instalacije

Nakon instalacije, provjerite je li Docker ispravno instaliran pokretanjem naredbe:

```
$ sudo docker run hello-world
```

Ako se prikaže poruka koja govori da je instalacija uspješna, Docker je ispravno instaliran i možete započeti s korištenjem.

Rad s ngnix kroz Docker

Primjer jednostavnog Dockerfile-a koji pokazuje osnovni koncept Docker-a može izgledati ovako:

```
FROM ubuntu:latest

RUN apt-get update && apt-get install -y nginx

COPY index.html /var/www/html/

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

U ovom primjeru, Dockerfile definira da se kao osnovna slika (base image) koristi najnovija verzija Ubuntu Linux operativnog sustava. Nakon toga, skripta instalira Nginx web server, kopira datoteku index.html u direktorij /var/www/html/ u kontejneru, otvara port 80 za vanjske zahtjeve i na kraju pokreće Nginx kao uslugu u pozadini.

Ovaj primjer se može pokrenuti koristeći sljedeće naredbe:

```
docker build -t mynginx .

docker run -d -p 8080:80 mynginx
```

Prva naredba (docker build) gradi Docker sliku s imenom mynginx koristeći Dockerfile koji se nalazi u trenutnom direktoriju. Druga naredba (docker run) pokreće kontejner koji koristi sliku mynginx, mapira lokalni port 8080 na port 80 u kontejneru, te ga pokreće u pozadini.

Nakon što se kontejner pokrene, može se otvoriti web preglednik i upisati adresa "localhost:8080" kako bi vidjeli Nginx web stranicu koja se nalazi unutar kontejnera. Ovo je samo jedan jednostavan primjer upotrebe Docker-a i Dockerfile-a, a Docker ima mnogo drugih mogućnosti i funkcija koje ga čine popularnim alatom za razvoj i distribuciju softvera.

Još primjera rada s ngnix kroz Docker

1. Pokretanje Nginx web servera u Docker kontejneru:

```
docker run -p 8080:80 nginx
```

Ova naredba će preuzeti najnoviju verziju Nginx s Docker Hub-a i pokrenuti ga u kontejneru. Vanjski port 8080 će biti mapiran na unutarnji port 80 kontejnera. Nakon što se kontejner pokrene, korisnik može pristupiti Nginx stranici putem "localhost:8080" u web pregledniku.

2. Pokretanje Nginx web servera u Docker kontejneru s prilagođenom konfiguracijskom datotekom:

javascript

```
docker run -p 8080:80 -v /putanja/do/konfiguracijske/datoteke:/etc/nginx/conf.d nginx
```

Ova naredba će pokrenuti Nginx u kontejneru s prilagođenom konfiguracijskom datotekom koja se nalazi na lokalnom sistemu. Vanjski port 8080 će biti mapiran na unutarnji port 80 kontejnera. Konfiguracijska datoteka se nalazi u lokalnom direktoriju /putanja/do/konfiguracijske/datoteke i mapirana je na direktorij /etc/nginx/conf.d unutar kontejnera.

3. Pokretanje više instanci Nginx web servera korištenjem Docker Compose-a:

*yaml

```
version: '3'
services:
  web:
    image: nginx
    ports:
      - "8080:80"
  web2:
    image: nginx
    ports:
      - "8081:80"
```

Ovaj Docker Compose datoteka opisuje dva servisa (web i web2) koji koriste Nginx sliku. Svaki servis je mapiran na drugi vanjski port (8080 i 8081) i oba će pokrenuti Nginx u kontejneru. Korisnik može pokrenuti ova dva servisa koristeći sljedeću naredbu:

```
docker-compose up
```

Nakon pokretanja, korisnik može pristupiti Nginx stranicama putem "localhost:8080" i "localhost:8081" u web pregledniku.

*NAPOMENA:

YAML (YAML Ain't Markup Language) je format za serijalizaciju podataka koji se može čitati i pisati ljudima. To je zapravo vrlo jednostavan način pisanja strukturiranih podataka u obliku teksta, koji se može koristiti za definiranje konfiguracijskih datoteka, datoteka za spremanje podataka, skripti za automatizaciju i druge svrhe. YAML datoteke često se koriste u kombinaciji s Dockerom i drugim tehnologijama koje se koriste za izgradnju i upravljanje aplikacijama.

Zadaci za vježbu:

I. nivo

1. Napravite Docker sliku s Nginx-om i učitajte je na Docker Hub. Zatim pokrenite tu sliku kao kontejner na svom lokalnom računalu i provjerite radi li Nginx poslužitelj ispravno.

2. Postavite Nginx poslužitelj kao reverse proxy za drugi web poslužitelj (na primjer, Apache) pomoću Docker kontejnera. Pokrenite oba kontejnera i provjerite jesu li povezani tako da Nginx preusmjerava zahtjeve na Apache.

3. Konfigurirajte Nginx kao poslužitelj za statičke datoteke (na primjer, HTML, CSS i JavaScript). Postavite Nginx kontejner tako da poslužuje jednostavnu HTML stranicu koju ste prethodno pripremili.

4. Postavite Nginx da radi kao load balancer za više instanci drugog web poslužitelja (na primjer, Apache) pomoću Docker kontejnera. Pokrenite nekoliko Apache kontejnera i provjerite jesu li povezani tako da Nginx usmjerava zahtjeve na sve instance.

5. Konfigurirajte Nginx kontejner tako da radi kao SSL/TLS proxy za drugi web poslužitelj (na primjer, Apache). Postavite SSL certifikat i konfigurirajte Nginx da ga koristi kako bi zaštitio komunikaciju između klijenta i poslužitelja.

II. nivo

1. Postavite više kontejnera s Nginx-om i Apache-om, postavite ih kao dio iste mreže u Dockeru i konfigurirajte Nginx tako da usmjerava zahtjeve na više Apache instanci. Zatim provjerite kako se Nginx ponaša kada jedan od Apache kontejnera padne ili se ponovno pokrene.

2. Postavite Jenkins server kao Docker kontejner i konfigurirajte Nginx tako da poslužuje Jenkins web sučelje preko SSL-a. Generirajte i instalirajte SSL certifikat za Jenkins, zatim postavite Nginx da ga koristi kako bi osigurao sigurnu komunikaciju između klijenta i poslužitelja.

3. Napravite Docker sliku koja se sastoji od Nginx-a i aplikacijskog poslužitelja (na primjer, Node.js ili Ruby on Rails). Konfigurirajte Nginx da preusmjerava zahtjeve na aplikacijski poslužitelj i dodajte neke jednostavne HTML stranice u aplikacijski kod. Pokrenite sliku kao kontejner na svom lokalnom računalu i provjerite možete li dohvatiti stranice preko Nginx-a.

III. nivo (Samo za one koji se žele više poigrati!)

1. Postavite Docker sliku koja sadrži Nginx, Node.js i MongoDB te konfigurirajte ih tako da stvore jednostavnu aplikaciju za upravljanje zadacima (Task Manager). Konfigurirajte Nginx kao proxy server koji usmjerava zahtjeve na Node.js aplikaciju, koja zatim koristi MongoDB za spremanje podataka. Postavite aplikaciju tako da može prihvatiti zahtjeve putem REST API-ja i koristite Swagger za dokumentiranje API-ja.

2. Napravite Docker sliku koja se sastoji od Nginx-a, PHP-a i MySQL-a te postavite jednostavnu web aplikaciju koja koristi PHP i MySQL za prikazivanje i uređivanje podataka. Konfigurirajte Nginx da usmjerava zahtjeve na PHP aplikaciju, koja zatim koristi MySQL za spremanje podataka. Postavite web aplikaciju tako da može prihvatiti korisničke unose putem HTML formi i provjerite kako se podaci sprema u MySQL bazu podataka.

3. Postavite Docker sliku koja se sastoji od Nginx-a, Apachea, PHP-a i PostgreSQL-a te postavite složeniju web aplikaciju za upravljanje projektima. Konfigurirajte Nginx kao proxy server koji usmjerava zahtjeve na Apache aplikaciju, koja zatim koristi PHP i PostgreSQL za spremanje i dohvaćanje podataka. Postavite aplikaciju tako da podržava autentifikaciju korisnika, omogućava stvaranje novih projekata, upravljanje korisničkim pravima, te dohvaćanje i uređivanje zadataka u okviru projekta.

Za sve eventualne primjedbe, komentare, sugestije obratiti se na mail: narcisa.hadzajlic@dl.unze.ba; adin.jahic2019@size.ba