Docker

Docker je skup proizvoda platforma kao usluga (PaaS - Platform as a Service) koji koriste virtuelizaciju na nivou OS za isporuku softvera u paketima koji se nazivaju kontejneri.

Postao je ključni alat u svetu razvoja softvera i DevOps prakse, omogućavajući lakše upravljanje aplikacijama i njihovu distribuciju.

Osnovna ideja Docker-a je da omogući konzistentno izvršavanje aplikacija bez obzira na okruženje u kojem se pokreću, eliminisanjem potrebe za brigom o različitim zavisnostima i konfiguracijama sistema.

Kontejneri omogućavaju pakovanje aplikacija zajedno sa svim potrebnim bibliotekama i zavisnostima, pružajući konzistentno i predvidivo okruženje za izvršavanje softvera. Prvi put je pokrenut 2013. godine i razvio ga je Docker, Inc.

Docker kontejneri su laki za prenos i pokretanje na različitim platformama, što olakšava rad na lokalnom razvoju, testiranju i implementaciji u različitim okruženjima, uključujući lokalne servere, virtualne mašine i cloud infrastrukturu. Ova fleksibilnost čini Docker idealnim za modernu agilnu razvojnu metodologiju i podržava kontinualna integraciju i isporuku (CI/CD) procese.

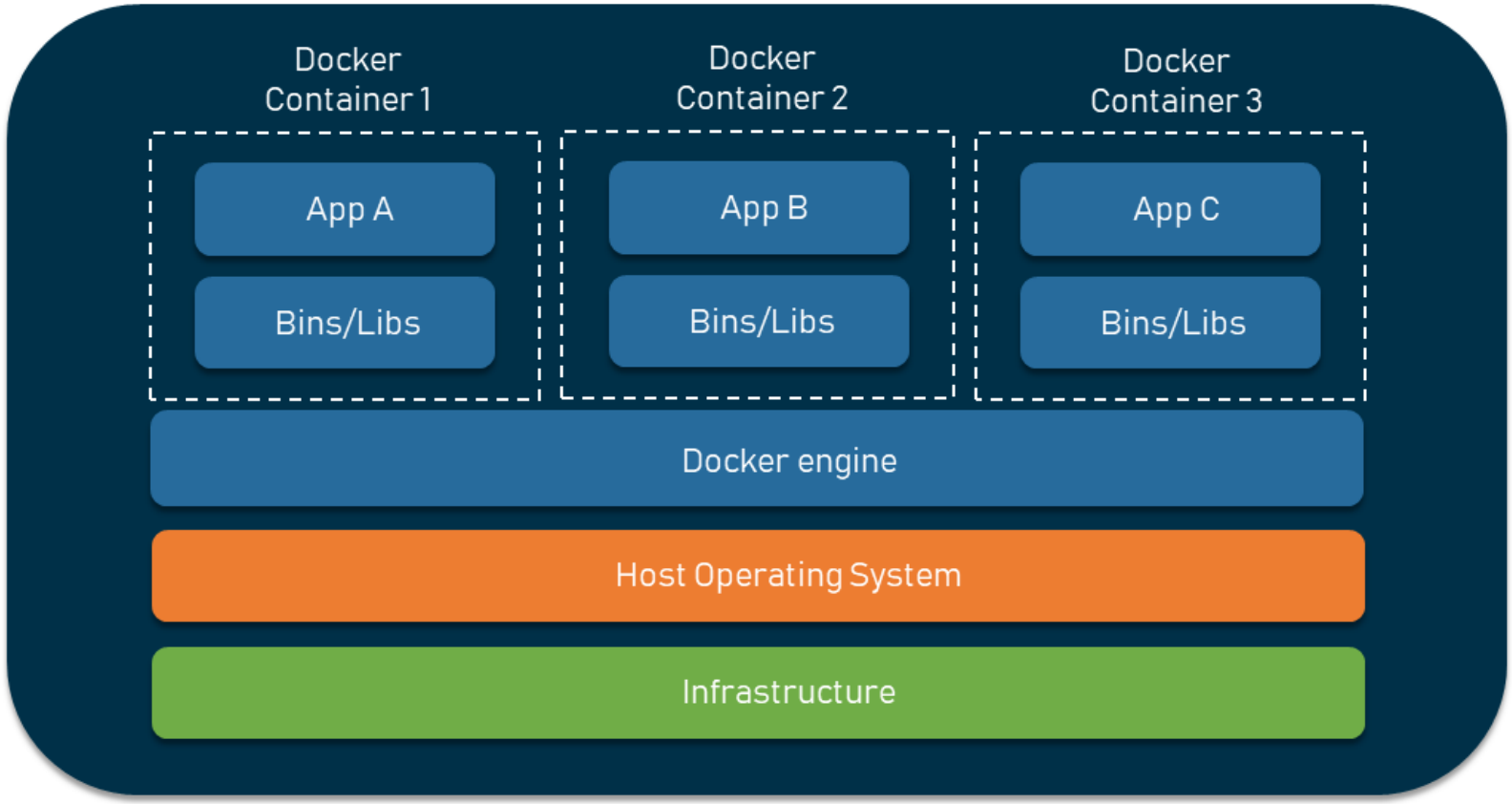
Osim toga, Docker ima bogat ekosistem dodataka (Docker Hub) koji omogućava deljenje i distribuciju gotovih kontejnera, što pojednostavljuje procese deljenja softvera i ubrzava razvoj. Takođe, Docker podržava orkestraciju kontejnera pomoću alata poput Docker Compose, Kubernetes-a ili Docker Swarm-a, omogućavajući efikasno upravljanje i skaliranje kontejnerizovanih aplikacija. Sve ove karakteristike čine Docker ključnim alatom u savremenom softverskom razvoju i DevOps praksi.

2. Docker arhitektura

Docker arhitekturu čine sledeće komponente:

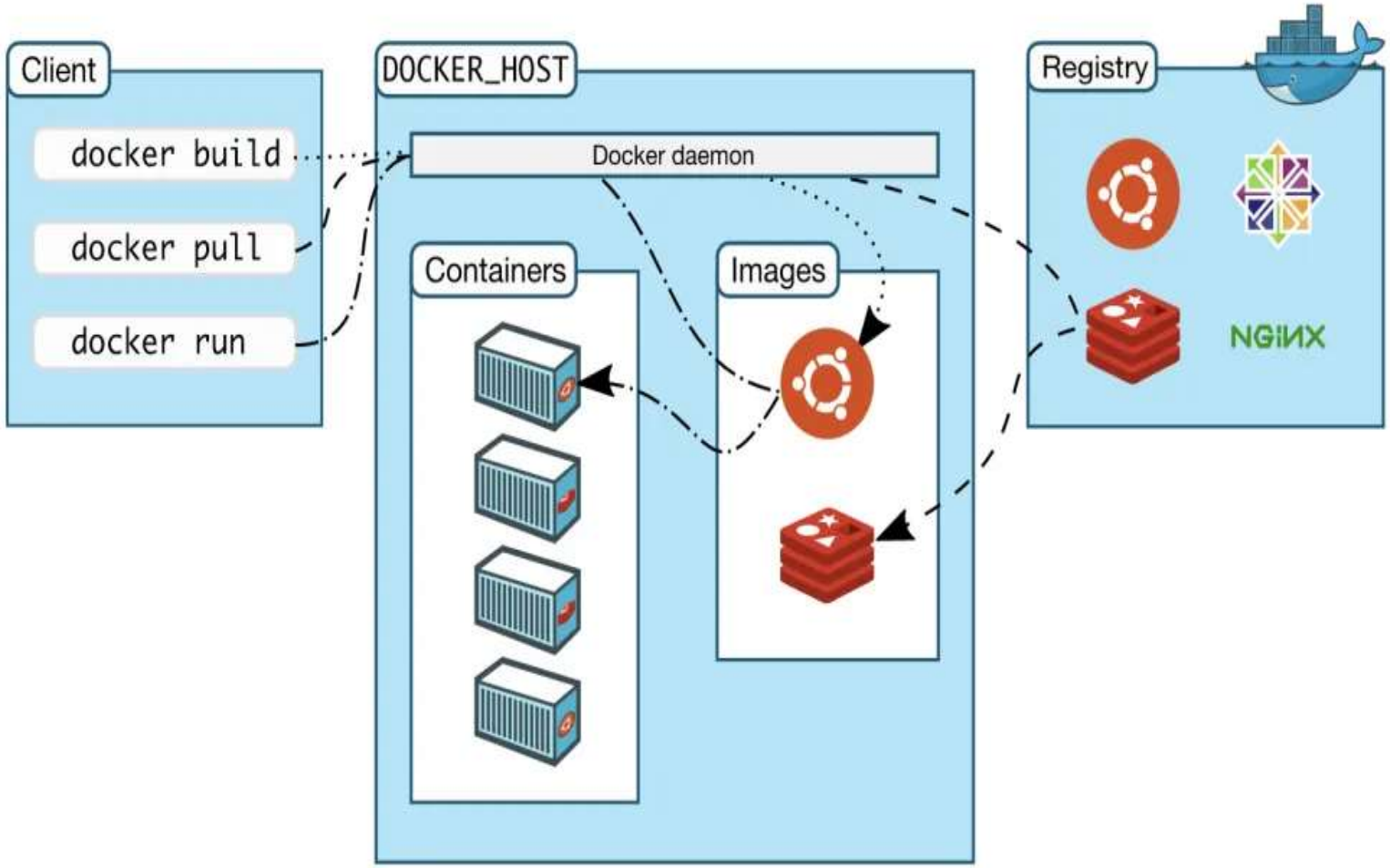
- Docker Daemon: Ovo je pozadinski proces koji upravlja Docker kontejnerima na sistemu. Docker daemon je odgovoran za izvršavanje komandi koje se šalju preko Docker CLI (Command Line Interface).
- Docker CLI (Command Line Interface): Korisnički interfejs koji omogućava korisnicima da komuniciraju s Docker-om. Korisnici mogu izdavati komande za upravljanje Docker kontejnerima, mrežama, volumenima i drugim Docker objektima.
- Docker Client: Docker CLI koristi Docker Client da šalje komande Docker Daemonu. Docker Client može komunicirati s lokalnim Docker Daemonom ili sa Daemonom na udaljenom serveru.
- Docker Images: Docker slike predstavljaju upakovan softver, uključujući aplikacije i sve njihove zavisnosti, u jednu celinu. Slike se koriste kao temelj za kreiranje i pokretanje Docker kontejnera.
- Docker Containers: Kontejneri su instance Docker slika koje se izvršavaju u izolovanom okruženju. Oni predstavljaju pokrenute kopije Docker slika i omogućavaju izolaciju, prenosivost i laku reprodukciju aplikacija.
- Docker Registry: Registry je skladište za Docker slike. Docker Hub je javni registry koji omogućava deljenje i preuzimanje slika. Korisnici mogu takođe koristiti privatne registre za skladištenje internih slika.

DOCKER CONTAINERS



altexsoft

Prikaz izvršavanja docker naredbi



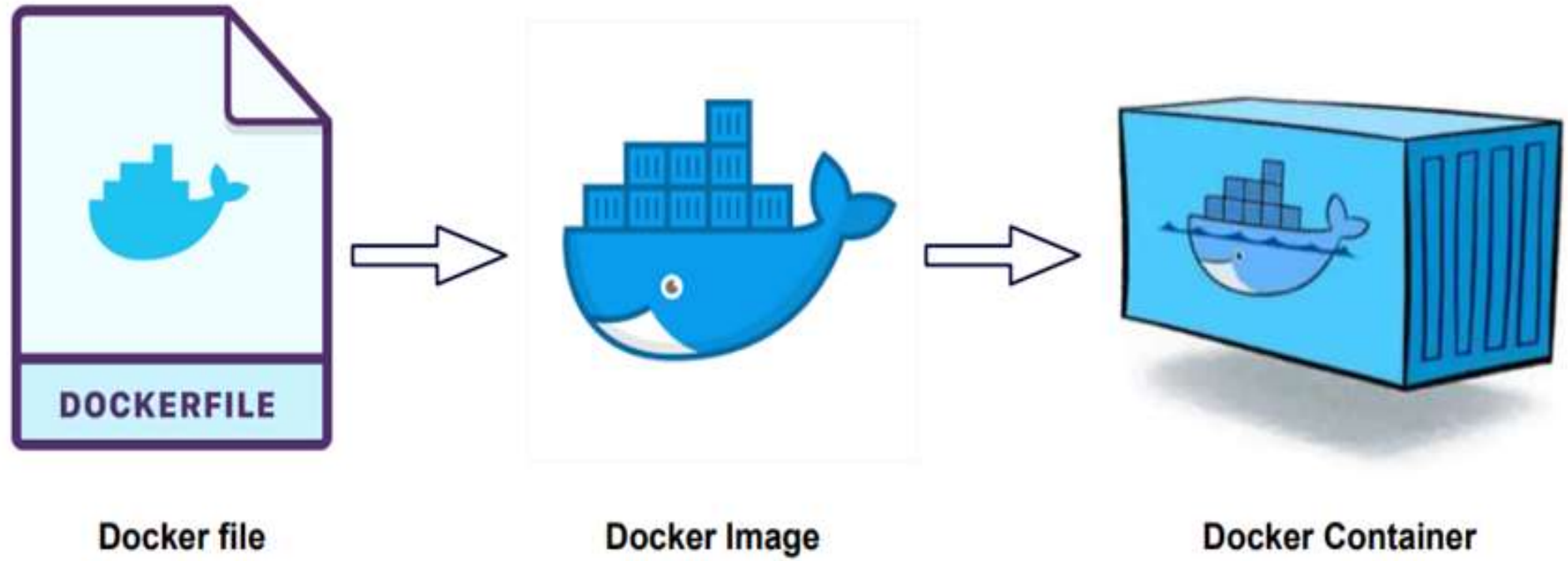
2.1. Uvod u docker slike

Docker slike (eng. Docker images) se koriste za pakovanje i distribuciju softvera, zajedno sa njihovim bibliotekama i konfiguracijama, kao nezavisne jedinice koje se lako mogu implementirati na različitim okruženjima. Slika je u suštini šablon na osnovu koga se prave Docker kontejneri. Korišćenjem Docker slika, programeri mogu da obezbede da njihove aplikacije konzistentno i pouzdano rade na različitim platformama, od razvoja do testiranja i produkcije. Takođe mogu da uštede vreme i trud ponovnom upotrebom postojećih slika iz javnih ili privatnih repozitorijuma, umesto da sve grade od početka.

Da biste kreirali Docker sliku, potrebno je da definišete Dockerfile, što je skripta koja određuje baznu sliku, softverske pakete za instaliranje, datoteke za kopiranje i druge konfiguracione postavke. Nakon što izgradite sliku, možete je skladištiti u registru, kao što su Docker Hub ili privatni repozitorijum, i deliti je sa drugima ili je implementirati na serverima ili cloud platformama, kao što su AWS ili Azure.

U celini, Docker slike pružaju moćan i fleksibilan način za upravljanje i implementiranje aplikacija u kontejnerizovanom okruženju, što može poboljšati prenosivost, skalabilnost i bezbednost.

Mesto Docker slike u Docker strukturi



2.2. Uvod u docker kontejnere

Docker kontejneri su izolovane jedinice za izvršavanje aplikacija, omogućavajući konzistentno funkcionisanje softvera bez obzira na okruženje. Oni pakiraju aplikacije zajedno sa svim potrebnim zavisnostima, bibliotekama i konfiguracijama. Docker kontejneri se koriste za pojednostavljivanje razvoja, testiranja i implementacije aplikacija, čime se omogućava brža isporuka softvera i efikasnije upravljanje infrastrukturom.

Kontejneri su posebno korisni u sledećim scenarijima:

- Razvoj i testiranje: Kontejneri omogućavaju programerima da brzo stvore i testiraju aplikacije u konzistentnom okruženju, bez brige o problemima sa kompatibilnošću ili konfliktnim zavisnostima.
- Implementacija: Kontejneri se mogu koristiti za pakovanje i implementaciju aplikacija, obezbeđujući standardizovano i predvidivo okruženje koje se lako može ponoviti na više servera.
- Mikroservisi: Kontejneri se često koriste u kontekstu arhitekture mikroservisa, gde se aplikacije dele na manje, nezavisne servise koji se mogu razvijati, implementirati i skalirati nezavisno.
- Kontinualna integracija i kontinualna implementacija (CI/CD): Kontejneri se mogu koristiti za automatizaciju testiranja, izgradnju i implementaciju aplikacija, omogućavajući efikasniji i jednostavniji proces razvoja.

3. Instalacija Docker-a

3.1. Instalacija na Windows uređajima

Preduslov za instalaciju Docker Desktop-a jeste Enable-ovana virtualizacija koja se može proveriti pokretanjem BIOS-a.

1. Preuzmite program za instalaciju za Docker Desktop s web stranice Docker Hub: https://www.docker.com/products/docker-desktop
2. Pokrenite preuzeti program za instalaciju.
3. Potrebno je slediti uputstva za instalaciju koja su navedena u linku. Kada se instalacija završi, Docker Desktop će se automatski pokrenuti.
4. Ako se ne pokrene automatski, pokrenite Docker Desktop izbornik Start.
5. Nakon pokretanja, Docker Desktop bi trebalo da radi i prikaže grafički interfejs.
6. Proverite je li Docker uspešno instaliran i konfiguriran tako što ćete otvoriti PowerShell i izvršiti sledeću naredbu:

```
docker run hello-world
```

Ukoliko Vam docker ne radi, proverite u BIOS-u da li je uključena Virtualizacija (treba da stoji Enabled).

3.2. Instalacija na Mac uredjajima

1. Preuzmite program za instalaciju za Docker Desktop sa web stranice Docker Hub: https://www.docker.com/products/docker-desktop

2. Kliknite dvaput na preuzeti program za instalaciju da biste otvorili paket.

3. Prevucite Docker Desktop ikonu u Applications direktorijum.

4. Nakon kopiranja, otvorite Applications direktorijum i dvaput kliknite na Docker Desktop aplikaciju.

5. Ako se pojavi upozorenje sigurnosti, kliknite "Open" da biste omogućili pokretanje aplikacije.

6. Nakon pokretanja, Docker Desktop bi trebalo da radi i da prikazuje grafičko korisničko sučeljavanje.

7. Proverite je li Docker uspešno instaliran i konfiguriran tako što ćete otvoriti Terminal i izvršiti sljedeću naredbu:

docker run hello-world

3.3. Instalacija na Linux uredjajima

Uzimajući u obzir to da je Docker izgrađen na osnovi Linux kontejnera instalacija na Linux baziranim operativnim sistemima prilično je jednostavna.

1. Potrebno je ažurirati pakete sistema. Otvoriti terminal i ažurirati pakete sistema pomoću sledeće komande.

```
sudo apt-get update
```

2. Instalirati pakete koji su neophodni za uspešnu instalaciju Dockera.

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

3. Dodati Docker GPG ključ.

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

4. Dodati Docker APT repozitorijum.

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

5. Ažurirati pakete i instalirati Docker.

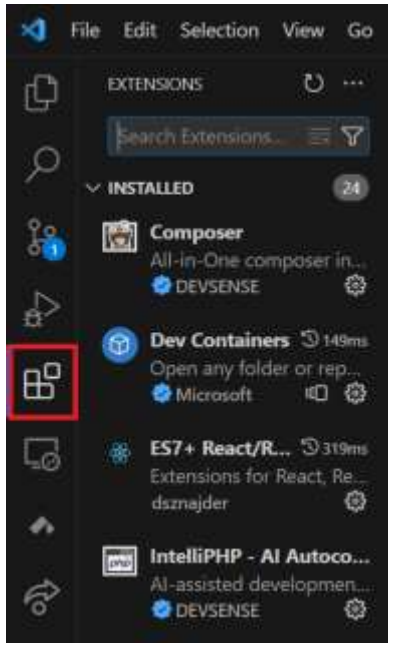
```
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. Proveriti da li je uspešno izvršena instalacija Dockera.

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

3.4. Docker ekstenzija za Visual Studio Code

Otvoriti Visual Studio Code i odabrati sa leve strane deo Extensions koji je na narednoj slici označen crvenim kvadratićem.



Slika 5 - VSCode Extenstion panel

U input polju uneti tekst “docker” (broj 1 na slici) i iz padajuće liste odabrati onu ekstenziju koja je licencirana od strane Microsoft-a I kliknuti na dugme “Install” (broj 2 na slici).

Ukoliko je sve uspešno odrađeno otvoriće se prozor u kome u plavim dugmićima piše “Disable” i “Uninstall” (broj 3 na slici).

Slika 6 - Instalacija docker ekstenzije

4. Docker images

Postoje dve vrste slike: built-in i custome. U nastavku će biti objašnjena oba koncepta slika.

4.1. Built-in images

Built-in Docker slike su unapred definisane slike koje se automatski preuzimaju kada se korisnik odluči za pokretanje određenog kontejnera. Ove slike su obično osnovne slike operativnog sistema, programskog jezika ili aplikacije koje su dostupne putem Docker Hub-a. Primeri ugrađenih slika uključuju operativne sisteme poput Alpine Linux-a, Ubuntu-a, kao i specifične slike za programski jezik poput Python, Node.js, i druge.

Neki od primera kako možete da pokučete slike su:

- Slike za operativne sisteme:
  - `docker pull ubuntu`
  - `docker pull debian`
- Slike za operativne sisteme:
  - `docker pull python`
  - `docker pull node`
- Baze podataka:
  - `docker pull mysql`
  - `docker pull postgres`
- Web serveri i proxy:
  - `docker pull nginx`
  - `docker pull httpd`

4.2. Custome images

Prilagođene Docker slike su kreirane od strane korisnika ili razvojnog tima prema specifičnim potrebama aplikacije ili projekta. Ove slike se obično grade koristeći Dockerfile, koji definiše korake za instalaciju zavisnosti, podešavanje okruženja i postavljanje aplikacije unutar kontejnera. Prilagođene slike omogućavaju potpunu kontrolu nad sadržajem i konfiguracijom kontejnera, čime se prilagođavaju specifičnim zahtevima aplikacije ili tima.

Najlakše je shvatiti na primeru. Projekat možete povući sa github repozitorijuma preteći sledeće korake:

1. git clone https://github.com/elab-development/docker-projects/
2. cd docker-projects/1-node
3. code .

Opis aplikacije: Node.js aplikacija je serverska aplikacija koja koristi JavaScript kao programski jezik i izvršava se na serverskoj strani.

U ovom primeru, koristi se Express.js.

Aplikacija definiše rutiranje za dve rute - prva prikazuje HTML stranicu za dodavanje cilja kursa, dok druga obrađuje POST zahtev za ažuriranje cilja kursa.

Takođe, koristi se body-parser za parsiranje podataka poslatih putem POST zahteva. package.json fajl je konfiguracioni fajl u kojem se definišu zavisnosti i druge informacije o Node.js projektu.

Ovaj fajl sadrži spisak npm (Node Package Manager) paketa koje aplikacija koristi, kao i skripte koje se mogu izvršiti u okviru projekta.

U ovom slučaju, express i body-parser su zavisnosti koje su navedene u package.json fajlu.

Potrebno je, u foleru 1-node, kreirati fajl i nazvati ga **dockerfile** ili **Dockerfile**.

U dockerfile-u otkucati sledeće komande:

```
FROM node:18
WORKDIR /app
COPY package.json /app
RUN npm install
COPY . /app
EXPOSE 80
CMD ["node", "server.js"]
```

```
FROM node:18
WORKDIR /app
COPY package.json /app
RUN npm install
COPY . /app
EXPOSE 80
CMD ["node", "server.js"]
```

Sada će biti objašenjena svaka linije ovog dockerfile-a.

1. **FROM node:18** - Ova komanda postavlja osnovnu Docker sliku na Node.js sliku verzije 18. To znači da će Docker slika biti bazirana na operativnom sistemu koji uključuje Node.js verziju 18.
2. **WORKDIR /app** - Postavlja radni direktorijum u Docker slici na "/app". Sve naredne komande će se izvršavati unutar ovog direktorijuma.
3. **COPY package.json /app** - Kopira lokalni package.json fajl u direktorijum "/app" u Docker slici. Ovo se radi odvojeno kako bi se omogućila optimalna upotreba keša tokom izvršavanja npm install koraka.
4. **RUN npm install** - Izvršava npm install komandu unutar "/app" direktorijuma. Ova komanda instalira sve zavisnosti navedene u package.json fajlu i takođe izvršava unutar Docker slike.
5. **COPY . /app** - Kopira sve fajlove iz trenutnog lokalnog direktorijuma (gde se nalazi Dockerfile) (prvi parametar) u "/app" direktorijum (drugi parametar) u Docker slici. Ovo uključuje sve fajlove aplikacije.
6. **EXPOSE 80** - Ova komanda označava da će kontejner, zasnovan na ovoj slici, biti dostupan na portu 80. Ova linija nije obavezna za internu komunikaciju unutar kontejnera, ali je informativna i može se koristiti prilikom mapiranja portova na matičnoj mašini.
7. **CMD ["node", "server.js"]** - Postavlja podrazumevanu komandu koja će se izvršiti kada se kontejner pokrene. U ovom slučaju, pokreće se Node.js aplikacija "server.js" pomoću node komande.

Komanda docker build koristi se za izgradnju Docker slike iz Dockerfile-a. Ovo je osnovna sintaksa naredbe:

docker build -t IME_SLIKE:OZNAKA PUTANJA_DO_DOCKERFAJLA

Gde je:

- -t IME_SLIKE:OZNAKA: Opcija -t omogućava označavanje (naming) slike sa zadatim imenom i oznakom.
- PUTANJA_DO_DOCKERFAJLA: Putanja do direktorijuma koji sadrži Dockerfajl.

Pokretanjem komande u terminalu, sa putanje na kojoj se nalazi dockerfile, dobijamo kreiranu sliku:

docker build -t wishlist-app .

Brojevi na slici predstavljaju:

1. **Učitavanje definicije izgradnje iz datoteke Dockerfile:** Dockerfile je datoteka koja sadrži instrukcije za izgradnju slike. U ovom slučaju, Dockerfile se nalazi u istom direktorijumu kao i terminal.
2. **Učitavanje datoteke dockerignore:** Ako postoji datoteka .dockerignore, Docker će je koristiti kako bi odredio koje fajlove ili direktorijume treba ignorisati tokom kopiranja.
3. **Preuzimanje slike osnovnog kontejnera**: Docker preuzima osnovnu sliku (u ovom slučaju node:18) sa Docker Hub-a ako već nije dostupna lokalno.
4. **Postavlja radni direktorijum unutar slike na "/app"**.
5. **Kopira lokalni package.json fajl u "/app" direktorijum unutar slike**.
6. **Izvršavanje npm install za instalaciju zavisnosti paketa**.
7. **Kopira sve fajlove iz lokalnog direktorijuma (gde se nalazi Dockerfile) u "/app" direktorijum unutar slike.**
8. **Dodeljuje ID novokreiranoj slici**: b63aa…27f4
9. **Dodeljuje naziv i tag** (u ovom slučaju "wishlist-app") novokreiranoj Docker slici.

Primećujemo da se nisu izvršile sve komande koje su napisane u dockerfile-u: **EXPOSE** i **CMD**. Ove komande se NE izvršavaju pri kreiranju slike, već pri kreiranju kontejnera.

Ukoliko ponovo pokrenemo istu komandu `docker build -t wishlist-app .` primetićemo drugačiji ispis.

```
[+] Building 0.7s (10/10) FINISHED
 =>1[internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 36B                                                                                 0.0s
 =>2[internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/node:18                                                          0.6s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 184B                                                                                   0.0s
 =>3[1/5] FROM docker.io/library/node:18@sha256:aa329c613f0067755c0787d2a3a9802c7d95eecdb927d62b910ec1d28689882f     0.0s
 =>4CACHED [2/5] WORKDIR /app                                                                                       0.0s
 =>5CACHED [3/5] COPY package.json /app                                                                             0.0s
 =>6CACHED [4/5] RUN npm install                                                                                    0.0s
 =>7CACHED [5/5] COPY . /app                                                                                        0.0s
 => exporting to image                                                                                              0.0s
 => => exporting layers                                                                                             0.0s
 =>8=> writing image sha256:b63aa2140eef0b44c9c15ae9b4bf5d77566c5222adf396d8a0495dd9fc27f4fe                        0.0s
 =>9=> naming to docker.io/library/wishlist-app                                                                     0.0s
```

Slika 8 - ponovno build-ovanje slike bez ikakvih izmena

Koraci 1, 2, 8 i 9 su isti.

3. Kratka izlazna poruka nakon ove komande ukazuje na to da slika već postoji na lokalnom računaru i da će se ona koristiti, umesto da se ponovo preuzima s Docker Hub-a.

Koraci 4, 5, 6 i 7 imaju oznaku **[CACHED]** ispred, to znači da Docker koristi keširane rezultate iz prethodnih build-ova umesto da ponovo izvršava te korake. Ovo je deo mehanizma keširanja koji pomaže u ubrzanju procesa izgradnje slike.

Ukoliko promenimo nešto u kodu, npr. na 24. liniju u server.js fajlu dodamo još jedan uzvičnik i ponovo build-ujemo sliku primetićemo da nisu baš sve komande keširane. Kada dođe do izmene u kodu Docker primenjuje invalidaciju keša (**cache invalidation**) za korake koji slede nakon te linije. To znači da će se koraci koji se odnose na kopiranje fajlova ili izvršavanje komandi koje dolaze posle promenjenih fajlova ponovno izvršiti.

```
<h2>Moja lista želja!</h2>
<h2>Moja lista želja!</h2>
```

```
[+] Building 1.3s (11/11) FINISHED
 =>1[internal] load build definition from Dockerfile                                                0.0s
 => => transferring dockerfile: 36B                                                                 0.0s
 =>2[internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load metadata for docker.io/library/node:18                                          1.2s
 => [auth] library/node:pull token for registry-1.docker.io                                         0.0s
 => [internal] load build context                                                                   0.0s
 => => transferring context: 1.21kB                                                                 0.0s
 =>3[1/5] FROM docker.io/library/node:18@sha256:aa329c613f0067755c0787d2a3a9802c7d95eecdb927d62b910ec1d28689882f     0.0s
 =>4CACHED [2/5] WORKDIR /app                                                                       0.0s
 =>5CACHED [3/5] COPY package.json /app                                                             0.0s
 =>6CACHED [4/5] RUN npm install                                                                    0.0s
 =>7[5/5] COPY . /app                                                                               0.0s
 => exporting to image                                                                              0.0s
 => => exporting layers                                                                             0.0s
 =>8=> writing image sha256:a4e145452eb45eb5a1de9258a6c86f3707fb2af5ef50e435b1bae8ebef70dd24        0.0s
 =>9=> naming to docker.io/library/wishlist-app                                                     0.0s
```

Ukoliko promenimo nešto u fajlu package.json i izvršimo komandu
`docker build -t wishlist-app:v2 .` imaćemo drugačiji ispis.

Slika 1 - izmena koda u package.json fajlu

```
[+] Building 1.3s (11/11) FINISHED
 => [internal] load build definition from Dockerfile                                                                0.0s
 => => transferring dockerfile: 36B                                                                                 0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/node:18                                                          1.2s
 => [auth] library/node:pull token for registry-1.docker.io                                                         0.0s
 => [internal] load build context                                                                                   0.0s
 => => transferring context: 184B                                                                                   0.0s
 => [1/5] FROM docker.io/library/node:18@sha256:aa329c613f0067755c0787d2a3a9802c7d95eecdb927d62b910ec1d28689882f     0.0s
 => CACHED [2/5] WORKDIR /app                                                                                       0.0s
 => CACHED [3/5] COPY package.json /app                                                                             0.0s
 => CACHED [4/5] RUN npm install                                                                                    0.0s
 => CACHED [5/5] COPY . /app                                                                                        0.0s
 => exporting to image                                                                                              0.0s
 => => exporting layers                                                                                             0.0s
 => => writing image sha256:a4e145452eb45eb5a1de9258a6c86f3707fb2af5ef50e435b1bae8ebef70dd24                        0.0s
 => => naming to docker.io/library/wishlist-app:v2                                                                  0.0s
```

Iako smo 3 puta izvršili komandu za kreiranje slike sa nazivom "wishlist-app", na kraju postoje dve slika sa nazivom " wishlist -app".

Razlog za to je što su nazivi slika jedinstveni, i svaki put kada izvršimo komandu, stara slika će biti zamenjena novom slikom.

Poslednja komanda se ipak malo razlikuje zato što smo pored naziva dodali :v2 što će postaviti tag slike na v2.

Tag se koristi za razlikovanje različitih verzija iste slike. To može biti verzija softvera, datum kreiranja ili bilo koji drugi identifikator koji vam pomaže da identifikujete konkretnu verziju slike.

Ukoliko se ne navede tag po defaultu će stojati latest. Međutim, preporučljivo je izbegavati korišćenje 'latest' taga, već umesto toga eksplicitno navesti odgovarajuću verziju koja odgovara vašoj aplikaciji.

Da bismo potvrdili ovu tvrdnju, možemo prikazati sve dostupne slike pomoću komande

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
wishlist-app	latest	a4e145452eb4	10 minutes ago	1.1GB
wishlist-app	v2	a4e145452eb4	10 minutes ago	1.1GB

U ispisu primećujemo sledeće kolone:

- REPOSITORY: Ovo je naziv slike.
- TAG: Tag se koristi za razlikovanje različitih verzija iste slike. To može biti verzija softvera, datum kreiranja ili bilo koji drugi identifikator koji vam pomaže da identifikujete konkretnu verziju slike.
- IMAGE ID: Ovo je jedinstveni identifikator (ID) slike. Svaka slika ima svoj jedinstveni ID koji se koristi za identifikaciju prilikom upotrebe Docker komandi.
- CREATED: Ova kolona pokazuje datum i vreme kada je slika kreirana. Pomaže vam da pratite vreme kada su slike dodavane ili ažurirane.
- SIZE: Veličina slike u bajtovima. Ova vrednost prikazuje koliko prostora zauzima slika na disku. Važno je imati na umu veličinu slika kako biste upravljali prostorom na disku i optimizovali resurse.

Docker komanda **docker inspect** možemo da istražimo konkretne specifikacije slike:

docker inspect IMAGE_ID_or_NAME

gde je:

- **IMAGE_ID_or_NAME:** ID ili ime Docker slike.

Na našem primeru: `docker inspect wishlist-app`

Na sledećoj slici biće prikazane samo ključne stavke koje možemo da pročitamo iz slike:

- **Id**: identifikator slike.
- **RepoTags**: popis oznaka slike.
- **Created**: datum i vreme kada je slika kreirana.
- **ExposedPorts**: koji port će biti izložen iz kontejnera kada se slika ubaci u njega
- **Env**: spisak *environment* varijabli
- **Cmd**: spisak komandi koje se izvršavaju kada se kada se slika ubaci u kontejner
- **WorkingDir**: glavni direktorijum u slici
- **Os**: operativni sistem na kome će se pokretati aplikacija
- **Size**: ukupna veličina slike, uključujući sve slojeve (layers) i podatke koji se nalaze u slici. Prikazano je u bajtovima.
- **Layers**: slojevi koji čine Docker sliku. Svaki korak u Dockerfile-u čini jedan sloj. Slojevi omogućavaju efikasno deljenje i keširanje, jer se samo izmenjeni slojevi ponovo izrađuju prilikom ažuriranja slike. Možemo primetiti da u Dockerfile-u postoji 7 linija koda, što u ovom nizu layers-a predstavlja poslednjih 7 elemenata niza. Prvih 5 layers-a u nizu se odnose na node sliku i na linux sliku zato što i oni imaju slojeve koje treba izvršiti.

```
[
    {
        "Id": "sha256:a4e145452eb45eb5a1de9258a6c86f3707fb2af5ef50e435b1bae8ebef70dd24",
        "RepoTags": [
            "wishlist-app:latest",
            "wishlist-app:v2"
        ],
        "RepoDigests": [],
        "Parent": "",
        "Comment": "buildkit.dockerfile.v0",
        "Created": "2024-03-11T12:44:03.438903214Z",
      "ExposedPorts": {
          "80/tcp": {}
      },
      "Tty": false,
      "OpenStdin": false,
      "StdinOnce": false,
      "Env": [
          "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin",
          "NODE_VERSION=18.19.1",
          "YARN_VERSION=1.22.19"
      ],
      "Cmd": [
          "node",
          "server.js"
      ],
      "ArgsEscaped": true,
      "Image": "",
      "Volumes": null,
      "WorkingDir": "/app",
```

Docker komanda **docker tag** koristi se za dodelu ili promenu oznake Docker slike. Ovo je osnovna sintaksa naredbe:

docker tag SOURCE_IMAGE[:TAG] TARGET_IMAGE[:TAG]

gde je:

- **SOURCE_IMAGE**: Slika koju želite označiti.
- **TAG** (opciono): Oznaka koju želite dodeliti slici. Ako nije navedena, koristi se "latest".
- **TARGET_IMAGE**: Novo ime slike.

Na našem primeru: `docker tag wishlist-app:v2 wishlist-app:v1`

Objašnjenje: Dobili smo novu sliku, wishlist-app:v1, koja je kopija slike wishlist -app:v2.

Docker slika, koju NE KORISTI ni jedan kontejner, se može obrisati pomoću naredbe **docker rmi** (remove image). Ovo je osnovna sintaksa naredbe:

docker rmi IMAGE[:TAG]

gde je:

- **IMAGE** je ime slike koju želite izbrisati.
- **TAG** (opcionalno) je oznaka slike koju želite izbrisati. Ako nije navedena, pretpostavlja se "latest"

Na našem primeru: `docker rmi wishlist-app:v1`

Objašnjenje: Obrisana je slika sa nazivom goal-app

Docker slika, koju KORISTI neki kontejner, se može obrisati pomoću naredbe:

docker rmi -f IMAGE[:TAG] ili docker rmi --force IMAGE[:TAG]

Na našem primeru: `docker rmi -f wishlist-app:v2` ili `docker rmi --force wishlist-app:v2`

Objašnjenje: Obrisana je slika wishlist -app, ali se kontejner neće obrisati. Međutim, takva akcija može rezultirati neobičnim ponašanjem kontejnera, jer mu se oduzima osnovna slika.

4.3. Dodatne komande koje treba znati

Postoje još neke ključne reči koje nisu pomenute gore a mogu se koristiti za pisanje Dockerfile-a.

- **ARG:** koristi se samo u Dockerfile-u i koristi se za definisanje promenljivih koje se mogu koristiti prilikom izgradnje Docker slike. Ove promenljive su dostupne samo tokom faze izgradnje i neće biti dostupne u kontejneru koji se pokreće iz te slike.
- **ENV:** koriste se u Dockerfile-u, unutar projekta i kontejnera. ENV varijable se postavljaju u Dockerfile-u ili prilikom pokretanja kontejnera pomoću **docker run -e** opcije. One su dostupne u okruženju kontejnera tokom njegovog izvršavanja i mogu se koristiti u projektu ili aplikaciji unutar kontejnera, na isti način kao i bilo koja druga sistemska varijabla.
- **VOLUME:** Definiše volumene koje će kontejner deliti sa drugim kontejnerima ili spoljašnjim sistemom.

Primer će biti prikazan u delu za kontejnere.

5. Docker containers

Za pokretanje kontejnera su nam potrebne slike. Koristićemo već kreiranu sliku pod nazivom goal-app.

Komanda **docker run** se koristi za pokretanje Docker kontejnera iz Docker slike. Ovo je osnovna sintaksa naredbe:

docker run [OPTIONS] IMAGE[:TAG] [COMMAND] [ARG...]

gde su:

- **OPTIONS**: Dodatne opcije koje konfigurišu ponašanje kontejnera.
- **IMAGE**: Ime Docker slike koja će biti korišćena za pokretanje kontejnera.
- **TAG** (opciono): Oznaka slike (version tag). Ako nije navedena, koristi se "latest".
- **COMMAND**: Komanda koja će se izvršiti unutar kontejnera.
- **ARG...**: Argumenti prosleđeni komandi unutar kontejnera.

Primeri:

```
docker run ubuntu
```

Objašnjenje: Ova komanda će pokrenuti kontejner zasnovan na Ubuntu slici. Ubuntu slika ne postoji na lokalnoj mašini, pa ju je povukao sa docker hub-a.

```
Unable to find image 'ubuntu:latest' locally
latest: Pulling from library/ubuntu
57c139bbda7e: Pull complete
Digest: sha256:e9569c25505f33ff72e88b2990887c9dcf230f23259da296eb814fc2b41af999
Status: Downloaded newer image for ubuntu:latest
```

Slika 15 - rezultat izvršavanje komande docker run ubutnu

```
docker run -it ubuntu bash
```

gde je:

- **-it ubuntu**: Opcije -it omogućavaju interaktivni režim
- **bash**: komanda koja se pokreće unutar kontejnera.

Objašnjenje: Komandom **docker run -it ubuntu bash**, korisnik započinje interaktivnu sesiju unutar Docker kontejnera. Ubuntu slika se koristi kao osnovna platforma, a Bash je postavljen kao način interakcije s korisnikom unutar kontejnera. Kroz ovu sesiju, korisnik može izvršavati komande, pregledavati datoteke, kreirati direktorijume i obavljati druge zadatke unutar Ubuntu kontejnera. Ovo je korisno, na primer, za eksperimentisanje sa specifičnostima kontejnerizacije i radom u izolovanom okruženju.

Slika 16 - navigiranje i kreiranje direktorijuma u kontejneru sa ubutnu slikom

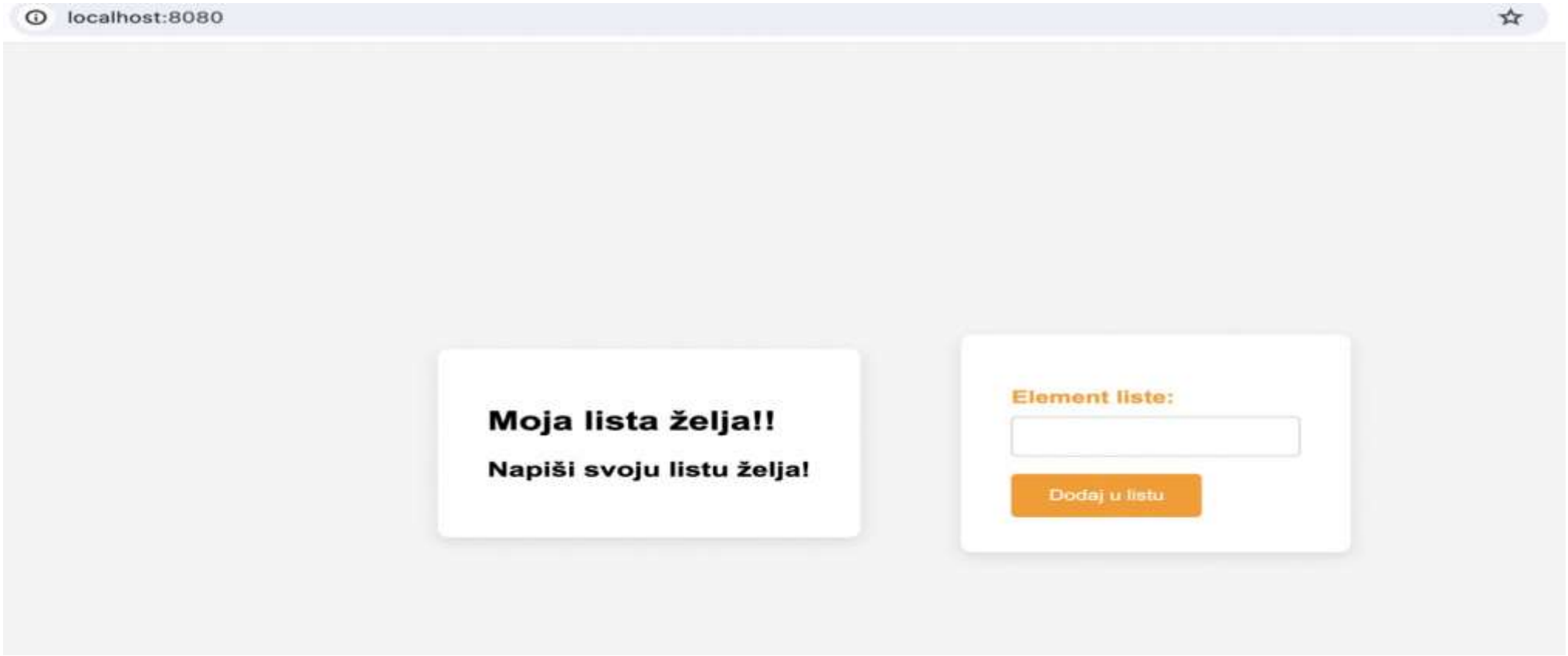
3. docker run -p 8080:80 wishlist-app

docker run -p HOST_PORT:CONTAINER_PORT

gde se:

- HOST_PORT: Port na lokalnom sistemu na koji će biti mapiran port iz kontejnera.
- CONTAINER_PORT: Port unutar kontejnera koji će biti mapiran na lokalni sistem.

Objašnjenje: U našem primeru, -p 8080:80 znači da će port 80 unutar kontejnera biti mapiran na port 8080 na lokalnom sistemu. Ovo omogućava pristup node veb serveru unutar kontejnera putem lokalnog port 8080. Kada se pokrene komanda u terminalu, terminal će da blokira i nije moguć dalji unos teksta u terminal. Otvorite brauzer i otkucajte u putanji localhost:8080. To će omogućiti pristup aplikaciji pokrenutoj unutar kontejnera.



4. **Moramo da otvorimo novi terminal i u njemu otkcamo komandu: docker ps**

Objašnjenje: Komanda docker ps koristi se za prikazivanje trenutno pokrenutih Docker kontejnera na sistemu.

Prvi kontejner pokrenut komandom docker run ubuntu nije interaktivni i nije imao pokrenut proces koji bi održavao kontejner aktivan, što znači da se kontejner automatski zatvorio nakon startovanja jer nije bilo nikakvog aktuelnog zadatka.

Drugi kontejner, pokrenut sa docker run -it ubuntu bash, bio je interaktivni kontejner sa Bash-om. Kada smo unutar njega pokrenuli Bash, kontejner je ostao aktivan i interaktivan dok je Bash bio aktivan. Međutim, nakon što smo otkucali exit i izašli iz Bash-a, kontejner se zatvorio jer nije bilo daljih procesa koji bi održavali kontejner aktivan. Nakon toga, kada smo pokrenuli docker run -p 8080:80 wishlist-app, koji predstavlja kontejner sa slikom wishlist -app koja pokreće Node server, taj kontejner je zauzeo terminal i blokirao ga. U ovom slučaju, Node server je aktivan unutar kontejnera, a terminal je blokiran sve dok server radi.

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
d6b35921e97d	wishlist-app	"docker-entrypoint.s…"	2 minutes ago	Up 2 minutes	0.0.0.0:8080->80/tcp	distracted_moore

Slika 18 - spisak svih trenutno pokrenutih Docker kontejnera

- *CONTAINER ID: Ovo je jedinstveni identifikator (hash) za svaki kontejner. Koristi se za jednoznačno identifikovanje kontejnera.*
- *IMAGE: Ovo je ime Docker slike na osnovu koje je kreiran kontejner.*
- *COMMAND: Ova kolona prikazuje komandu koja je pokrenuta unutar kontejnera prilikom njegovog startovanja.*
- *CREATED: Ova kolona pokazuje vreme koje je prošlo od kreiranja kontejnera.*
- *STATUS: Ova kolona pokazuje trenutni status kontejnera, tj. da li je pokrenut (Up) ili zaustavljen (Exited).*
- *PORTS: Ova kolona prikazuje mapiranje portova između lokalnog sistema i kontejnera. Na primer, "0.0.0.0:8080 -> 80/tcp" znači da je port 8080 na lokalnom sistemu mapiran na port 80 unutar kontejnera.*
- *NAMES: Ovo je ime kontejnera. Kontejnerima se automatski dodeljuju jedinstvena imena ukoliko im nije eksplicitno dodeljeno prilikom kreiranja.*

5. Takođe u novom terminalu: `docker stop d6b` ili `docker stop distracted_moore`

docker stop CONTAINER_ID_or_NAME

gde je:

- **CONTAINER_ID_or_NAME**: ID ili ime Docker kontejnera.

Objašnjenje: Komanda **docker stop** koristi se za zaustavljanje pokrenutih Docker kontejnera. Ova komanda zaustavlja izvršavanje procesa unutar kontejnera.

1. `docker ps`

Objašnjene: Ponovnim pokretanjem komande vidimo da nema ni jednog aktivnog kontejnera.

```
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

Slika 1 - spisak aktivnih kontejnera (prazan)

2. `docker ps -a` ili `docker ps --all`

gde je:

- **-a** ili **--all**: prikazuju se svi kontejneri, bez obzira na to da li su trenutno pokrenuti ili zaustavljeni.

Objašnjenje: Komanda **docker stop** koristi se za zaustavljanje pokrenutih Docker kontejnera. Ova komanda zaustavlja izvršavanje procesa unutar kontejnera.

```
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATUS                      PORTS
               NAMES
d6b35921e97d   wishlist-app         "docker-entrypoint.s…"   4 minutes ago    Up 4 minutes                0.0.0.0:8080->80/tcp
               distracted_moore
5c0d3396baa3   hello-world          "/hello"                 45 minutes ago   Exited (0) 45 minutes ago
               cranky_buck
92225daa62dd   hello-world          "/hello"                 12 days ago      Exited (0) 12 days ago
               admiring_euler
```

docker rm CONTAINER_ID_or_NAME

gde je:

- **CONTAINER_ID_or_NAME**: ID ili ime Docker kontejnera.
  `docker rm 5c0` ili `docker rm cranky_buck`

Objašnjenje: Komanda **docker rm** koristi se za uklanjanje jednog ili više Docker kontejnera sa sistema. Kada izvršite docker rm, trajno brišete kontejnere, oslobađate resurse i oslobađate prostor na disku. Napomena – ne možete obrisati pokrenut kontejner, već ga prvo morate zaustaviti!

9. `docker start d6b` ili `docker start distracted_moore`

docker start CONTAINER_ID_or_NAME

gde je:

- **CONTAINER_ID_or_NAME**: ID ili ime Docker kontejnera.

Objašnjenje: Komanda **docker start** koristi se za ponovno pokretanje ranije zaustavljenih Docker kontejnera. Ova komanda pokreće procese unutar kontejnera i omogućava ponovno korišćenje kontejnera koji je prethodno bio zaustavljen. Kontejner se pokreće u neblokirajućem procesu, tj. u pozadini. Odradićemo komandu *docker stop distracted_moore* kako bismo pokazali neke nove načine pokretanja kontejnera.

10. `docker inspect d6b` ili `docker inspect distracted_moore`

docker inspect [OPTIONS] CONTAINER [CONTAINER...]

gde je:

- **OPTIONS:** Dodatne opcije koje se mogu koristiti prilikom ponovnog inspekcije kontejnera.
- **CONTAINER [CONTAINER...]:** Identifikator ili ime kontejnera čije informacije želite inspekciju.

Objašnjenje: Komanda **docker inspect** se koristi za dobijanje detaljnih informacija o Docker objektima kao što su kontejneri, slike, mreže i volumeni. Kada se koristi za kontejner, ova komanda pruža sveobuhvatne informacije o konfiguraciji kontejnera, uključujući mrežna podešavanja, postavke resursa, i druge relevantne informacije.

- **ID**: Jedinstveni identifikator kontejnera.
- **Created**: Datum i vreme kreiranja kontejnera.
- **Path**:
- **Args**:
- **State:** opisuje trenutno stanje kontejenra
  - **Status**: Stanje kontejnera (opis je na sledećoj stranici **).
- **Image**: Jedinstveni identifikator slike
- **Name**: Ime kontejnera.
- **RestartCount**: Broj puta da je kontejner povremeno pokrenut.
- **Driver**: Kontejner runtime koji se koristi za pokretanje kontejnera.
- **Platform**: Operativni sistem na kom se kontejner pokreće.
- **Config**: Konfiguracija kontejnera (hostname, exposed ports, environment variables).
  - **Hostname:** Skraćen zapis id-ja koji kontejner koristi unutar sebe i u komunikaciji sa drugim kontejnerima. U primeru iznad, hostname je "614388844e17".
  - **ExposedPorts:** Spisak portova koje kontejner izlaže za spoljašnju komunikaciju. U primeru iznad, port 80/tcp je izložen.
  - **Env:** Varijable okruženja dostupne aplikaciji unutar kontejnera. U primeru iznad, definisane su PATH, NODE_VERSION i YARN_VERSION.
  - **Cmd:** Lista komandi koje se izvršavaju pri pokretanju kontejnera. U primeru iznad, izvršava se node server.js.
  - **Image:** Slika Docker-a na kojoj je baziran kontejner. U primeru iznad, slika je "goal-app"**.**
  - **Volumes:** Spisak fizičkih direktorijuma na matičnoj mašini koji su montirani unutar kontejnera. Ovo omogućava trajne podatake nezavisno od životnog veka kontejnera.
  - **Workdir:** Direktorijum unutar kontejnera u kojem se izvršavaju procesi.
  - **Entrypoint:** Lista komandi koje se izvršavaju kao glavni proces kontejnera pri pokretanju. Ima prioritet nad Cmd.
- **NetworkSettings**: Mrežna podešavanja kontejnera (ovo će biti detaljnije objašnjeno u delu mreže):
  - **bridge**: Naziv mreže kojoj je kontejner priključen. U većini slučajeva, ova mreža je podrazumevana "bridge" mreža Docker-a.
  - **ports**: Mapa koja povezuje portove izložene unutar kontejnera sa portovima na matičnoj mašini. U ovom primeru nema mapiranja portova.
  - **macaddress**: MAC adresa dodeljena kontejneru za komunikaciju na mreži.

(**) Tipovi statusa kontejnera i njihova načenja:

- **Up**: Kontejner je pokrenut i trenutno se izvršava.
- **Exited**: Kontejner je završio izvršavanje i izašao sa ishodnim kodom.
- **Paused**: Kontejner je pauziran pomoću "docker pause" naredbe. To znači da su svi procesi unutar kontejnera pauzirani i ne rade.
- **Restarting**: Kontejner se ponovo pokreće.
- **Creating**: Kontejner se kreira i još uvek nije pokrenut.
- **Removing**: Kontejner se uklanja pomoću "docker rm" naredbe.

`docker stop d6b` ili `docker stop distracted_moore`

docker stop CONTAINER_ID_or_NAME

gde je:

- **CONTAINER_ID_or_NAME**: ID ili ime Docker kontejnera.

Objašnjenje: Komanda **docker stop** koristi se za zaustavljanje pokrenutih Docker kontejnera. Ova komanda zaustavlja izvršavanje procesa unutar kontejnera.

`docker start -a d6b` ili `docker start -a distracted_moore`

docker start -a CONTAINER_ID_or_NAME

gde je:

- **-a**: označava da će se pri pokretanju kontejnera prikažu izlaz kontejnera.

Objašnjenje: Komanda **docker start -a** se koristi za pokretanje prethodno zaustavljenog Docker kontejnera i prikazivanje njegovog izlaza u trenutnom terminalu. Kada se koristi ova opcija (*eng. flag*) **-a**, izlaz kontejnera će biti prikazan u terminalu. Nakon pokretanja kontejnera, koristite veb pretraživač da pristupite aplikaciji na putanji **localhost:8080**, a zatim unesite "Novi laptop!" u odgovarajuće input polje. Vratite se u konzolu i primetićete ispis " Novi laptop!". U novom terminalu otkucajte *docker stop distracted_moore* kako bismo oslobodili terminal.

1. Moramo da otvorimo novi terminal i u njemu otkcamo komandu: `docker restart d6b` ili `docker restart distracted_moore`

```
docker restart [OPTIONS] CONTAINER [CONTAINER...]
```

gde je:

- **OPTIONS:** Dodatne opcije koje se mogu koristiti prilikom ponovnog pokretanja kontejnera.
- **CONTAINER [CONTAINER...]:** Jedan ili više identifikatora ili imena kontejnera koje želite ponovo pokrenuti.

Objašnjenje: Komanda **docker restart** se koristi za ponovno pokretanje već pokrenutog Docker kontejnera, čime omogućava osvežavanje procesa unutar kontejnera ili primenu promena u konfiguraciji. Kontejner se pokreće u neblokirajućem procesu, tj. u pozadini, nakon čega su svi procesi unutar kontejnera ponovo aktivirani.

2. `docker logs d6b` ili `docker logs distracted_moore`

```
docker logs [OPTIONS] CONTAINER_ID_or_NAME
```

gde je:

- **OPTIONS:** Dodatne opcije koje se mogu koristiti prilikom prikazivanja logova.
- **CONTAINER:** Identifikator ili ime kontejnera čije logove želite prikazati.

Objašnjenje: Komanda **docker logs** se koristi za prikazivanje izlaza kontejnera, odnosno prikazivanje logova generisanih tokom izvršavanja procesa unutar Docker kontejnera. Ova komanda omogućava korisniku da prati izlazne podatke ili logove iz kontejnera, što može biti korisno za dijagnostiku, praćenje rada aplikacija ili pronalaženje grešaka. Nakon što izvršite ovu komandu, biće vam prikazani logovi koji su generisani unutar odabranog kontejnera.

```
Novi laptop!
```

Slika 1 - Ispis u konzoli nakon izvršavanja komande docker logs

15. `docker attach d6b` ili `docker attach distracted_moore`

docker attach [OPTIONS] CONTAINER_ID_or_NAME

gde je:

- **OPTIONS:** Dodatne opcije koje se mogu koristiti prilikom priključivanja na kontejner.
- **CONTAINER:** Identifikator ili ime kontejnera na koji želite priključiti terminal.

Objašnjenje: Komanda **docker attach** se koristi za priključivanje terminala na pokrenuti Docker kontejner, omogućavajući korisniku interaktivnu sesiju sa procesima unutar kontejnera. Nakon izvršenja ove komande, korisnik će biti preusmeren na terminal unutar kontejnera, gde može izvršavati komande i pratiti izlazne podatke. Važno je napomenuti da će se nakon izlaska iz sesije (npr. pritiskom na Ctrl+C) procesi unutar kontejnera nastaviti izvršavati.

1. Takođe u novom terminalu: `docker stop d6b` ili `docker stop distracted_moore`

```
docker stop CONTAINER_ID_or_NAME
```

gde je:

- **CONTAINER_ID_or_NAME**: ID ili ime Docker kontejnera.

Objašnjenje: Komanda **docker stop** koristi se za zaustavljanje pokrenutih Docker kontejnera. Ova komanda zaustavlja izvršavanje procesa unutar kontejnera.

2. `docker run --rm -d -p 8080:80 --name web-server wishlist-app`

```
docker run –rm -d -p HOST_PORT:CONTAINER_PORT --name CONTAINER_NAME IMAGE_NAME
```

gde su:

- **--rm**: označava da se kontejner automatski briše nakon što se zaustavi. Ovo je korisno ako ne želite zadržavati kontejner nakon što završi njegova upotreba, što pomaže u čišćenju sistema i oslobađanju resursa.
- **-d**: pokreće kontejner u pozadinskom režimu, što znači da će se kontejner pokrenuti u pozadini i osloboditi terminal. To omogućava kontejneru da nastavi raditi u pozadini dok korisnik može nastaviti koristiti terminal za druge zadatke.
- **--name web-server**: dodeljuje ime "web-server" kontejneru. Ovo ime se može koristiti umesto ID-a kontejnera prilikom upravljanja kontejnerom, što olakšava identifikaciju i referenciranje kontejnera.

Objašnjenje: Pokretanjem ove komande biće vam prikazan jedinstven ID kontejnera, koji je obično dugačak niz karaktera. Ova komanda je pokrenuta u pozadinskom režimu (**-d** opcija), što znači da kontejner radi u pozadini, omogućavajući vam da nastavite sa kucanjem drugih komandi u ovom terminalu dok kontejner nastavlja sa radom. Korišćenjem komande **docker ps**, možete pregledati spisak svih aktivnih kontejnera na vašem sistemu. Komandom **docker stop web-server** zaustavljamo kontejner. Pokrenućemo komandu **docker ps -a** za izlistavanje svih kontejnera i primetićemo da tamo nema web-server kontejnera zato što se on obrisao opcijom **--rm**.

```
563b4f7984976d0feb2d1d2944140f555d6103d0f5209c36e53879ab5c1b3ff7
```

Slika 1 – ID kontejnera prikazan nakon komande docker run

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
563b4f798497	wishlist-app	"docker-entrypoint.s…"	48 seconds ago	Up 47 seconds	0.0.0.0:8080->80/tcp	web-server

Slika 25 - spisak aktivnih kontejnera

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
563b4f798497	wishlist-app	"docker-entrypoint.s…"	About a minute ago	Up About a minute	0.0.0.0:8080->80/tcp	web-server
d6b35921e97d	wishlist-app	"docker-entrypoint.s…"	26 minutes ago	Exited (137) About a minute ago		distracted_moore
5c0d3396baa3	hello-world	"/hello"	About an hour ago	Exited (0) About an hour ago		cranky_buck

Slika 26 - spisak svih kontejnera

18. `docker rm d6b` ili `docker rm distracted_moore`

docker rm [OPTIONS] CONTAINER [CONTAINER...]

gde je:

- **OPTIONS:** Dodatne opcije koje se mogu koristiti prilikom brisanja kontejnera.
- **CONTAINER [CONTAINER...]:** Jedan ili više identifikatora ili imena kontejnera koje želite ukloniti.

Objašnjenje: Komanda **docker rm** trajno uklanja Docker kontejner(e) sa sistema, oslobađajući resurse koje su kontejneri koristili. Kontejneri se brišu bez mogućnosti povratka, pa je važno pažljivo koristiti ovu komandu, posebno kada se brišu važni ili aktivni kontejneri. Pri korišćenju ove komande, svi podaci i promene unutar kontejnera će biti izgubljeni.