UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Računarstvo u oblaku

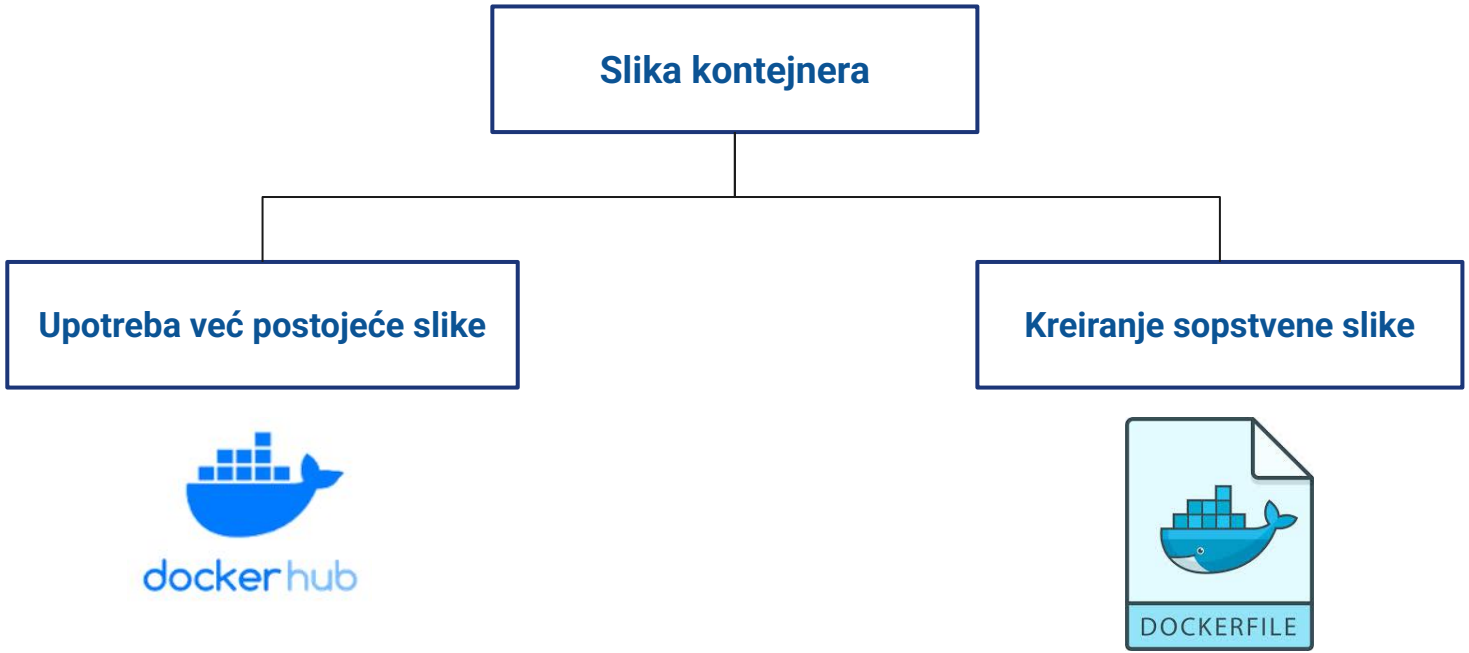
ms Helena Anišić

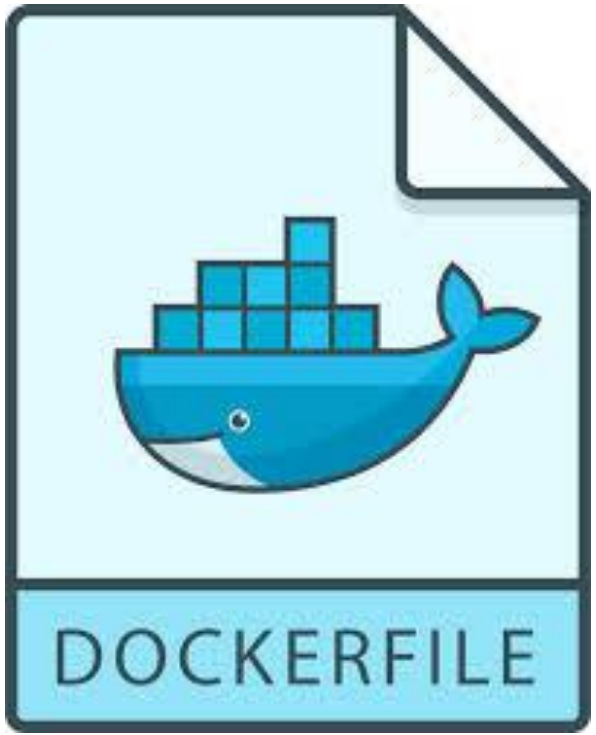
Zimski semester 2025/2026.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi

Dockerfile - osnovni pojmovi




DOCKERFILE

Dockerfile - osnovni pojmovi

- Dockerfile je tekstualni dokument koji sadrži sve potrebne komande za sastavljanje jedne slike kontejnera.
- Docker automatski kreira sliku kontejnera na osnovu instrukcija u Dockerfile-u.
- Docker izvršava instrukcije napisane u Dockerfile-u **redom** kojim su napisani.
- Dockerfile **mora** početi instrukcijom `FROM`.
- Instrukcije nisu case-sensitive, međutim konvencija je da se pišu velikim slovima kako bi se lakše razlikovali od argumenata
- Format Dockerfile-a:

```
# Comment
INSTRUCTION arguments
```

Build context

- Docker build naredba kreira sliku kontejnera na osnovu Dockerfile-a i konteksta
  - Kontekst je sve što se nalazi na putanji koja definisana u okviru naredbe
- Prvi korak jeste da se pošalje kontekst docker daemon-u
  - Ukoliko nešto želimo da izostavimo, definišemo .dockerignore fajl
- PRIMER:

```
docker build .
```

.dockerignore file

- Pre nego što se context pošalje docker daemon-u, docker CLI pokušava da pronađa fajl pod nazivom .dockerignore
  - Fajl mora biti u korenskom direktorijumu context-a
- Na ovaj način mogu da se izbegnu veliki fajlovi, nepotrebni ili oni koji su potencijalno nebezbedni

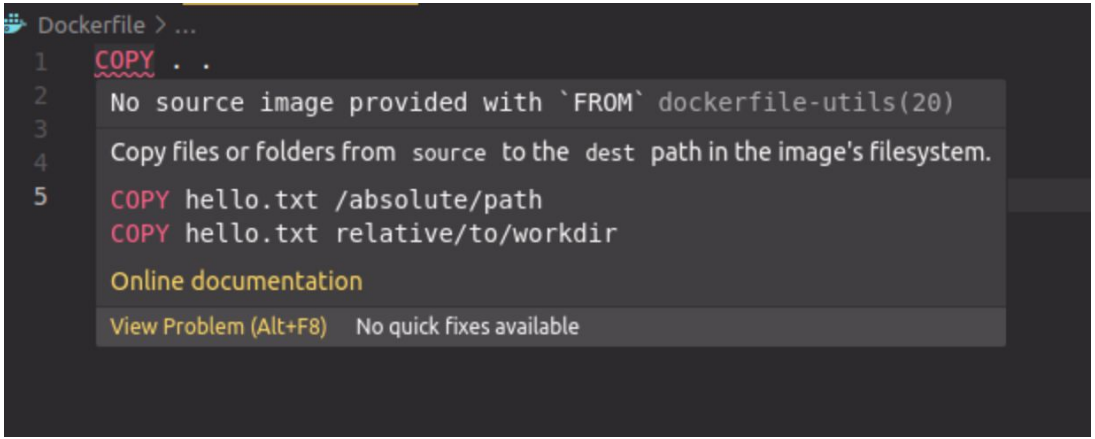
```
# This is just a comment.
README.md
node_modules
```

Dockerfile instrukcije

- FROM
- WORKDIR
- COPY
- ADD
- CMD
- ENTRYPOINT
- RUN
- LABEL
- EXPOSE
- ENV
- ARG
- VOLUME
- USER
- ONBUILD
- STOPSIGNAL
- HEALTHCHECK
- SHELL

Dockerfile instrukcija [FROM]

- Da bi Dockerfile bio validan mora da započne instrukcijom `FROM`.
  - *Visual Studio Code* ekstenzija za Docker vrlo jasno prikazuje nevalidnost Dockerfile-a bez `FROM` instrukcije



Dockerfile instrukcija [FROM]

```
FROM [--platform=<platform>] <image> [AS <name>]
```

```
FROM [--platform=<platform>] <image>[:<tag>] [AS <name>]
```

```
FROM [--platform=<platform>] <image>[@<digest>] [AS <name>]
```

- `FROM` instrukcija inicijalizuje novi *build stage* i postavlja baznu sliku za dalje instrukcije.
- Bazna slika može biti bilo koja validna slika
  - najjednostavnije je da se povuče neka slika sa javnog repozitorijuma poput *DockerHub*-a
    - DockerHub je predefinisani repozitorijum sa kojeg se skidaju slike ukoliko se ne navede neka platforma
- `ARG` je jedina instrukcija koja može da se nađe pre `FROM` naredbe

Dockerfile instrukcija [FROM]

- `[--platform=<platform>]`
  - opcioni tag koji može da specificira tačno platformu za koju da povuče sliku ukoliku `FROM` referencira multi-platformsku sliku
- `<image>[:<tag>]` ili `<image>[@<digest>]`
  - opcione vrednosti koje ukoliko se ne navedu, docker povlači sliku sa *latest* tagom
  - docker vraća grešku ako ne nađe sliku sa datim tagom na repozitorijumu.
- `[AS <name>]`
  - opciono dodavanje naziva *build stage*-u
  - naziv može da se koristi u narednim `FROM` i `COPY --from=<name>` instrukcijama kako bi se referencirali na sliku napravljenu u ovom stadijumu

Dockerfile instrukcija [FROM]

- Primer upotrebe *multistage build*-a

```
FROM node:12.13.0-alpine as build

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build



FROM nginx

EXPOSE 3000

COPY ./nginx/default.conf /etc/nginx/conf.d/default.conf

COPY --from=build /app/build /usr/share/nginx/html
```

Dockerfile instrukcija [FROM]

- PRIMERI različitih varijanti slika:
  - `FROM` python
    - Povući će python sliku sa tagom latest
  - `FROM` python:<version>
    - Primer: `FROM` python:3.10
    - Neki od ovih tagova mogu imati imena poput *bullseye* ili *buster*. Ovo su nazivi *Debian release-a* na kojima su zasnovane ove slike.
  - `FROM` python:<version>-slim
    - Slika kontejnera koja sadrži samo minimum potrebnih paketa koji su potrebni da bi python radio.
  - `FROM` python:<version>--alpine
    - Zasnovano na Alpine linux-u

Alpine Linux

- Alipine Linux je *lightweight* distribucija Linux-a dizajnirana da bude mala, jednostavna i bezbedna.
- Za razliku od većine Linux distribucija, Alpine Linux koristi musl, BusyBox i OpenRC umesto češće korištenih glibc, GNU Core Utilities i systemd-a respektivno.
- Slika kontejnera za Apline Linux je samo 5 MB
    - U poređenju sa Ubuntu slikom kontejnera koja je 75 MB

Dockerfile instrukcija [FROM]

- `FROM` scratch
  - Eksplicitno prazna slika
  - Najkorisnija kada se prave bazna slika (kao *debian* ili *busybox*) ili kada se prave super minimalne slike kao što je *hello-world* koje sadrže samo jedan binarni fajl i šta god je još potrebno
  - Neće kreirati dodatni sloj u slici kontejnera
  - Upotrebom scratch slike signaliziramo build procesu da želimo da naredna komanda u Dockerfile-u bude prvi sloj fajl sistema u slici kontejnera

Docker instrukcija [WORKDIR]

```
WORKDIR /path/to/workdir
```

- `WORKDIR` naredba kreira radni direktorijum za bilo koju `RUN`, `CMD`, `ENTRYPOINT`, `COPY` i `ADD` instrukciju koja sledi u Dockerfile-u
- Ukoliko `WORKDIR` naredba ne postoji, radni direktorijum će svakako biti kreiran
  - naziv radnog direktorijuma u toj situaciji će biti /
  - njegova pozicija će biti relativna u odnosu na baznu sliku koju koristimo
    - osim u situaciji kada ne koristimo ni jednu baznu sliku (`FROM scratch`)
- Najbolja praksa je definisati radni direktorijum da ne bude iznenađenja

Docker instrukcija [WORKDIR]

- `WORKDIR` instrukcija može da se koristi više puta u Dockerfile-u.
  - nakon što se pozicija radnog direktorijuma redefiniše, sve naredne naredbe će se odnositi na taj novi radni direktorijum
- PRIMER:

```
FROM ubuntu:latest
WORKDIR /my-workdir
RUN echo "work directory 1" > file1.txt
WORKDIR /my-workdir2
RUN echo "work directory 2" > file2.txt
```

Docker instrukcija [WORKDIR]

- Definicija putanje radnog direktorijuma:
  - relativna putanja
    - ukoliko je obezbeđena relativna putanja, onda će novi radni direktorijum biti kreiran u odnosu na prethodni radni direktorijum
  - apsolutna putanja
    - VSCode Docker ekstenzija upozorava da je preporučljivo koristiti apsolutne putanje

```
FROM ubuntu:latest
WORKDIR /a
WORKDIR b
WORKDIR c
RUN pwd
```

Dockerfile instrukcija [USER]

`USER <user>[:<group>]` ili `USER *UID*[:*GID*]`

- Predefinisano je da je Docker kontejner root korisnik.
  - ovo može da predstavlja opasnost po bezbednost aplikacija koje se nalaze u kontejnerima
- Instrukcija `USER` omogućava izmenu korisnika unutar Docker kontejnera
- USER instrukcija postavlja user name (UID) i opciono user group (GID) za predefinisani user i group koji će važiti u nastavku izvršavanja instrukcija Dockerfile-a
  - Specificirani korisnik se koristi za `RUN` instrukcije i tokom rada kontejnera za relevantne `ENTRYPOINT` i `CMD` komande
- Pre nego što se iskoristi komanda `USER` potrebno je kreirati datog datog korisnika naredbom:
  - `RUN groupadd -r <group_name> && useradd -r -g <group_name> <user_name>`

Dockerfile instrukcija [USER]

- Primer upotrebe USER naredbe:
  - radi provere uspešnosti postavljanja novog korisnika pokrenuti naredbu id u kontejneru

```
FROM ubuntu:latest
RUN apt-get -y update
RUN groupadd -r user && useradd -r -g user user
USER user
```

Dockerfile instrukcija [COPY]

- `COPY` naredba ima dve forme:
  - `COPY [--chown=<user>:<group>] [--chmod=<perms>] <src> ... <dest>`
  - `COPY [--chown=<user>:<group>] [--chmod=<perms>] ["<src>", ... "<dest>"]`
- `COPY` instrukcija kopira fajlove/direktorijume iz <src> putanje i dodaje ih u fajlsistem kontejnera na putanju <dest>
- <dest> putanja može da bude apsolutna ili relativna
  - Apsolutna putanja kopira fajlove direktno na tu putanju
    - `COPY test.txt /absoluteDir/`
  - Relativna putanja kopira fajlove relativno od postavljenog radnog direktorijuma
    - `COPY test.txt relativeDir/`
      - lokacija fajla će biti WORKDIR/relativeDir/

Dockerfile instrukcija [COPY]

- <src> putanja mora da bude unutar *build context*-a
  - prvi korak docker build naredbe jeste da se pošalje context direktorijum docker daemon-u
- Ako je <src> direktorijum, kompletan sadržaj se kopira zajedno sa metapodacima fajlsistema
  - direktorijum sam po sebi se ne kopira
- Ako je <src> bilo koja druga vrsta fajla, kopira se pojedinačno zajedno sa metapodacima
  - Ako se <dest> završava sa /, smatraće se direktorijumom, a sadržaj <src> će biti upisan na <dest>/base(<src>)
  - Ako se <dest> ne završava /, smatraće se regularnim fajlom i sadržaj <src> će se upisati na <dest>
- Ako <dest> ne postoji, kreiraće se
- Svaka izmena fajlova koji se kopiraju poništava prethodni keš i izaziva ponovnu izmenu datog sloja slike, kao i svih narednih

Dockerfile instrukcija [ADD]

- `ADD` naredba ima dve forme:
  - `ADD [--chown=<user>:<group>] [--chmod=<perms>] <src> ... <dest>`
  - `ADD [--chown=<user>:<group>] [--chmod=<perms>] ["<src>", ... "<dest>"]`
- `ADD` instrukcija kopira nove fajlove, direktorijume ili udaljene fajl URL-ove sa <src> putanje i dodaje ih u fajlsistem slike na putanji <dest>
- Putanje se interpretiraju relativno u odnosu na source of the context of the build
- <dest> putanja može da bude apsolutna ili relativna
  - Apsolutna putanja kopira fajlove direktno na tu putanju
    - `ADD test.txt /absoluteDir/`
  - Relativna putanja kopira fajlove relativno od postavljenog radnog direktorijuma
    - `ADD test.txt relativeDir/`
      - lokacija fajla će biti WORKDIR/relativeDir/

Dockerfile instrukcija [ADD]

- <src> putanja mora da bude unutar build context-a
  - Prvi korak docker build naredbe jeste da se pošalje context direktorijum docker daemon-u
- Ako je <src> URL, a <dest> se ne završava sa /, fajl će biti pruzet i kopiran u <dest>
- Ako je <src> direktorijum, kompletan sadržaj se kopira zajedno sa metapodacima fajlsistema
  - Direktorijum sam po sebi se ne kopira
- Ako je <src> lokalna tar arhiva, raspakovaće se.
- Ako je <src> bilo koja druga vrsta fajla, kopira se pojedinačno zajedno sa metapodacima
  - Ako se <dest> završava sa /, smatraće se direktorijumom, a sadržaj <src> će biti upisan na <dest>/base(<src>)
  - Ako se <dest> ne završava /, smatraće se regularnim fajlom i sadržaj <src> će se upisati na <dest>
- Ako <dest> ne postoji, kreiraće se

Dockerfile instrukcija [ADD]

Primer `ADD` instrukcije:

```
FROM scratch
ADD ubuntu-bionic-oci-amd64-root.tar.gz /
CMD ["ls"]
```

[ADD] VS. [COPY]

COPY	ADD
Kopira fajlove sa putanje na lokalnoj mašini na destinaciju docker kontejnera	Kopira fajlove sa putanje na lokalnoj mašini na destinaciju docker kontejnera
COPY <src> <dest>	ADD <src> <dest>
Kopira fajlove samo sa lokalne mašine.	Može da radi preuzimanje sa URL i onda kopiranje.
Ne može da radi raspakivanje arhiva.	Radi i raspakivanje tar arhiva.

Dockerfile instrukcija [CMD]

- `CMD` instrukcija može da bude zadata u tri forme:
  - `CMD ["executable","param1","param2"]` (*exec* forma, preferirani način)
    - parsira se kao JSON array što znači da mora biti unutar dvostrukih navodnika
  - `CMD ["param1","param2"]` (predefinisani parametri za `ENTRYPOINT` instrukciju)
    - ne koristi izvršnu naredbu (*executable*), te se ona mora specificirati u okviru `ENTRYPOINT`-a
  - `CMD command param1 param2` (*shell* forma)
    - komanda se izvršava u shell-u
- Zadatak `CMD` instrukcije je da pokrene softver koji se nalazi u kontejneru ili da izvrši neku naredbu.
  - pokretanje .exe fajla
  - pokretanje bash terminala
- Može da postoji samo jedna `CMD` instrukcija u Dockerfile-u
  - ukoliko se navede više od jedne `CMD` instrukcije, izvršiće se poslednja u nizu
  - najveći prioritet ima komanda napisana tokom pokretanja kontejnera

Dockerfile instrukcija [ENTRYPOINT]

- `ENTRYPOINT` instrukcija može biti zadata u dve forme:
  - `ENTRYPOINT ["executable", "param1", "param2"]` (*exec* forma, preferirani način)
  - `ENTRYPOINT command param1 param2` (*shell* forma)
- `ENTRYPOINT` ima sličnu ulogu kao `CMD`
- Kada se pokrene kontejner sa naredbom `docker run <image>` svi elementi u exec formi `ENTRYPOINT` naredbe se nadodaju na kraj (i pregaze bilo kakvu `CMD` naredbu definisanu u Dockerfile-u)

[CMD] VS [ENTRYPOINT]

- `CMD` i `ENTRYPOINT` definišu komandu koja treba da se izvrši kada se kontejner pokrene
- Nekoliko pravila za njihovu upotrebu:
  - Dockerfile mora da sadrži ili `CMD` ili `ENTRYPOINT`
  - `CMD` komanda će biti pregažena kada se kontejner pokrene sa nekim drugim argumentima
  - `CMD` bi trebao da se koristi kao način da se definišu predefinisani argumenti za `ENTRYPOINT` ili za izvršavanje predefinisanih komandi
  - `ENTRYPOINT` treba da se koristi kada se kontejner koristi kao *executable*
  - Dobra praksa:
    - `ENTRYPOINT` koristiti kao putanju za izvršni fajl
    - `CMD` koristiti kao predefinisane argumente koji se prosleđuju

Dockerfile instrukcija [RUN]

- `RUN` instrukcija ima dve forme:
  - `RUN <command>` (*shell* forma, komanda je pokrenuta u shell-u, predefinisani za Linux je /bin/sh -c, cmd /S /C za Windows)
  - `RUN ["executable", "param1", "param2"]`
    - Ako je potrebno izvršiti naredbu u *shell*-u onda se u okviru exec forme mora navesti *shell* za izvršavanje
      - `RUN ["/bin/bash", "-c", "echo $HOME"]`
- `RUN` instrukcija izvršava bilo kakvu komandu u novom sloju povrh trenutne slike i *commit*-uje rezultat.
  - Rezultujuća *commit*-ovana slika će se koristiti za naredni korak u Dockerfile-u.
  - Raslojavanje docker slike pomoću `RUN` instrukcije i generisanje *commit*-ova odgovara konceptu dockera jer su *commit*-ovi jeftiniji i kontejneri mogu da budu kreirani sa bilo koje tačke u istoriji slike.

Dockerfile instrukcija [RUN]

- Primeri `RUN` instrukcije:

```
FROM ubuntu:latest
RUN apt-get update
RUN apt-get install -y apache2

# \ za prelazak u naredni liniju
RUN /bin/bash -c 'source $HOME/.bashrc && \
echo $HOME'
# shell forma
RUN /bin/bash -c 'source $HOME/.bashrc && echo $HOME'
# exec forma
RUN ["/bin/bash", "-c", "echo hello"]
```

[CMD] VS. [RUN]

- `RUN` i `CMD` nisu iste instrukcije
- `RUN` izvršava instrukciju tokom kreiranja slike i *commit*-uje rezultat izvršavanja
- `CMD` ne izvršava instrukciju tokom kreiranja slike, već tokom pokretanja kontejnera

Docker instrukcija [LABEL]

`LABEL <key>=<value> <key>=<value> <key>=<value> …`

- Dodaje metapodatke slici kontejnera
  - par ključ:vrednost
- Slika može da ima više od jedne labele
- Ako je potrebno da labela sadrži razmak - koristiti navodnike
- Koristiti duple navodnike posebno u sitaciji kada string interpolacija (`LABEL example="foo-$ENV_VAR"`)
  - single navodnici će pokupiti string takav kakav jeste bez njegovog raspakivanja
- Ako je potrebno da labela bude napisana u više redova koristiti \
- Labele koju su uključene u baznu sliku ili roditeljsku siku se nasleđuju
- Za pregled labele slike koristiti naredbu `docker image inspect`
  - Dodatno može da formatira ispis kako bi samo ispisao informacije o labelama
    - `docker image inspect --format='{{json .Config.Labels}}' myimage`

Docker instrukcija [LABEL]

- Primer upotrebe labele:

```
LABEL version="1.0"

LABEL multi.label1="value1" multi.label2="value2" other="value3"

LABEL description="This text illustrates \
that label-values can span multiple lines."

LABEL example="foo-$ENV_VAR"
```

Docker instrukcija [EXPOSE]

`EXPOSE <port> [<port>/<protocol>...]`

- `EXPOSE` instrukcija služi kao svojevrsna dokumentacija između osobe koja je napisala sliku za kontejner i osobe koja pokreće kontejner na osnovu date slike
  - Za stvarno objavljivanje porta koristit **-p** opciju prilikom `docker run` naredbe
  - Može se specificirati TCP/UDP
    - TCP je predefinisani

```
EXPOSE 80
EXPOSE 80/udp
```

Dockerfile instrukcija [VOLUME]

```
VOLUME ["/data"]
```

- `VOLUME` instrukcija kreira tačku *mount*-ovanja na definisanoj putanji u kontejneru
- Sa te putanje se učitavaju fajlovi/direktorijumi svaki put kada se pokrene kontejner
  - fajlovi/direktorijumi mogu biti sa host mašine ili drugih kontejnera
- Mount tačka može da bude zadata kao: JSON array ili string
  - JSON array: `VOLUME ["/var/log/"]`
  - string: `VOLUME /var/log`
- Namena `VOLUME` instrukcije:
  - čuvanje podataka čak iako se kontejner zaustavi ili ukloni,
  - deljenje podataka između kontejnera i
  - lakše upravljanje podacima.

Dockerfile instrukcija [VOLUME]

- Primer `VOLUME` instrukcije:

```
FROM ubuntu
RUN mkdir /myvol
RUN echo "hello world" > /myvol/greeting
VOLUME /myvol
```

Dockerfile instrukcija [ARG]

`ARG <name>[=<default value>]`

- `ARG` instrukcija definiše argument koji se koristi tokom kreiranja slike kontejnera
- `ARG` instrukcija u Dockerfile-u definiše predefinisanu vrednost argumenta
  - `docker build` naredba pozvana sa opcijom `--build-arg <varname>=<value>` omogućava zadavanje neke druge vrednosti datom argumentu
- Moguće je definisani više `ARG` instrukcija
- Definisani argument može da se koristi u instrukcijama Dockerfile-a koji slede nakon njegove definicije
- Postoji skup predefinisanih argumenta koji se mogu koristiti bez njihove prethodne definicije u Dockerfile-u (https://docs.docker.com/engine/reference/builder/#predefined-args)
  - za upotrebu datih argumenata dovoljno je samo izvršiti sledeću naredbu (primer HTTP_PROXY):
    - `$ docker build --build-arg HTTPS_PROXY=https://my-proxy.example.com .`

Dockerfile instrukcija [ARG]

- Primer upotrebe ARG instrukcije:

```
FROM ubuntu:latest
ARG GREET=FTN
RUN echo "Hey there! Welcome to $GREET" > greeting.txt
CMD cat greeting.txt
```

Dockerfile instrukcija [ENV]

`ENV <key>=<value> ...`

- `ENV` instrukcija postavlja environment varijablu <key> na vrednost <value>
- Moguće je postavljati više varijabli u jednoj liniji
- Kao i kod labela, environment varijable se nasleđuju od bazni ili roditeljskih slika
- Vrednost environment varijable se može koristiti u instrukcijam koje slede nakon nje
- Varijabla ostaje sačuvana kada se kontejner pokrene na osnovu date slike
    - `docker inspect` naredba daje uvid u varijable
    - `docker run –env <key>=<value>` omogućava izmenu varijabli tokom pokretanja kontejnera
- `ENV <key> <value>`
    - Sintaksa koja je nekada bila validna, ali se ne preporučuje

Dockerfile instrukcija [ENV]

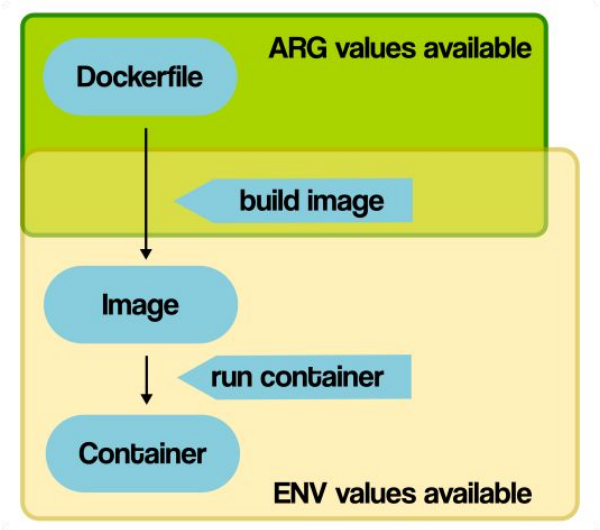
- Primer postavljanja environment varijabli:

```
ENV MY_NAME="John Doe"
ENV MY_DOG=Rex\ The\ Dog
ENV MY_CAT=fluffy
ENV MY_NAME="John Doe" MY_DOG=Rex\ The\ Dog \
      MY_CAT=fluffy
```

[ARG] VS [ENV]

- `ENV` je pre svega namenjen kontejnerima koji će biti izgrađeni na osnovu slike kontejnera, dok se `ARG` koristi za izgradnju slike kontejnera
- `ENV` je namenjen za predefinisane vrednosti budućih varijabli
  - dokerizovane aplikacija može da pristupi environment varijablama koje su definisane u slici kontejnera
  - preporučeni način da se prosleđue konfiguracione vrednosti u projekat
- `ARG` vrednosti nisu dostupne nakon što se slika kreira
  - kontejner koji radi nema pristup `ARG` vrednostima
- `ARG` i `ENV` se preklapaju tokom procesa kreiranja slike kontejnera

[ARG] VS [ENV]


ARG values available
Dockerfile
build image
Image
run container
Container
ENV values available

Dockerfile instrukcija [STOPSIGNAL]

```
STOPSIGNAL signal
```

- `STOPSIGNAL` instrukcija postavlja signal za sistemski poziv koji će biti poslat kontejneru prilikom završetka rada kontejnera
- Signal može biti:
  - naziv signala u formatu SIG<NAME> (npr. SIGKILL)
  - neoznačeni broj koji odgovara poziciji u syscall tabeli kernel-a (npr. 9)
- Predefinisani signal je SIGTERM
- Predefinisani `STOPSIGNAL` može da bude pregažen po pojedinačnom kontejneru upotrebom `--stop-signal` naredbe prilikom pokretanja kontejnera

Dockerfile instrukcija [HEALTHCHECK]

- `HEALTHCHECK` instrukcija ima dve forme:
  - `HEALTHCHECK [OPTIONS] CMD command`
    - proverava zdravlje kontejnera pokretanjem naredbe unutar kontejnera
    - komanda nakon `CMD` može biti ili u *shell* komanda (`HEALTHCHECK CMD /bin/check-running`) ili *exec* array
  - `HEALTHCHECK NONE`
    - onesposobljavanje bilo kakve provere zdravlja nasleđene od strane bazne slike
- `HEALTHCHECK` instrukcija govori dockeru kako da testira kontejner kako bi proverio da li i dalje radi
  - može da detektuje slučajeve kao što je veb server koji se zaglavio u beskonačnoj petlji i nije u mogućnosti da obradi naredne konekcije iako proces servera i dalje radi
- Instrukcija zahteva da se pored normalnog statusa ispiše i status koji opisuje zdravlje kontejnera
  - inicijalni status je *starting*
  - kada provera zdravlja prođe uspešno status zdravlja postaje *healthy*
  - posle nekoliko uzastopnih neuspeha status postaje *unhealthy*
- Može da postoji samo jedna `HEALTHCHECK` instrukcija u dockerfile-u
  - ako se navede više `HEALTHCHECK` instrukcija, važi poslednja napisana

Dockerfile instrukcija [HEALTHCHECK]

- Opcije:
  - --interval=DURATION (default: 30s)
    - vremenski interval između svake provere zdravlja, kao i vremenski interval između pokretanja kontejnera i prve provere zdravlja
  - --timeout=DURATION (default: 30s)
    - vremeski period nakon čijeg isteka provera zdravlja se smatra neuspešnom
  - --start-period=DURATION (default: 0s)
    - početni period obezbeđuje vreme inicijalizacije za kontejnere kojima je potrebno vreme za pokretanje. Neuspeh tokom tog perioda neće se računati u maksimalan broj ponovnih pokušaja. Međutim, ako provera zdravlja uspe tokom početnog perioda, kontejner se smatra pokrenutim i svi uzastopni neuspesi će se računati u maksimalan broj ponovnih pokušaja.
  - --start-interval=DURATION (default: 5s)
    - vremeski period između provera zdravlja tokom početnog perioda
  - --retries=N (default: 3)
    - broj ponavljanja provere zdravlja pre nego se provera smatra neuspešnom

Dockerfile instrukcija [HEALTHCHECK]

- Exit status pokazuje status zdravlja kontejnera:
  - 0 : success - kontejner je zdrav i spreman za upotrebu
  - 1 : unhealthy - kontejner ne radi kako treba
  - 2 : reserved - ne koristiti ovaj exit code
- Za pregled statusa zdravlja kontejnera koristiti naredbu `docker inspect`

Dockerfile instrukcija [HEALTHCHECK]

- Primer:
  - kreiran je mali veb server i potrebno je dodati proveru zdravlja kako bismo bili sigurni da je veb server funkcionalan
    - koristiće se komanda *curl* za proveru odziva veb servera
    - potrebno je na svakih 30se proveriti da li se veb server odaziva
    - veb server ima 3s da odgovori na zahtev
    - ako ne odgovori u 3 pokušaja na zahtev, postaviti exit status na unehalthy - 1

Dockerfile instrukcija [ONBUILD]

`ONBUILD` *`INSTRUCTION`*

- `ONBUILD` instrukcija dodaje okidačku instrukciju (engl. *trigger*) na sliku kontejnera
  - taj okidač će se izvršiti kada se slika kontejnera bude koristila kao bazna slika za neku drugu sliku kontejnera
- Kako radi `ONBUILD`:
  - kada se naiđe na `ONBUILD` instrukciju, docker dodaje okidače u metapodatke slike koja se kreira.
    - ni u kakvom drugom smislu ova instrukcija ne utiče na trenutno kreiranu sliku
  - `docker inspect` naredba omogućava uvid u sve okidače date slike pod ključem *OnBuild*
  - kada se data slika koristi kao bazna slika u okviru naredbe `FROM`, builder traži `ONBUILD` okidače i izvršava ih u istom redosledu kako su navedeni.
    - ako neki okidač ne uspe da se izvrši, akcija se abortira.
    - ako se uspešno izvrši, nastavlja se na narednu instrukciju nakon `FROM`
  - okidači se ne nasleđuju

Dockerfile instrukcija [ONBUILD]

- Primer na linku: https://blog.frankel.ch/onbuild-overlooked-docker-directive/

Dockerfile instruction [SHELL]

```
SHELL ["executable", "parameters"]
```

- `SHELL` instrukcija omogućava da se pregazi predefinisani *shell* za *shell* formu komande.
  - predefinisani *shell* za Linux OS je ["/bin/sh", "-c"]
  - predefinisani *shell* na Windows OS je ["cmd", "/S", "/C"]
- `SHELL` instrukcija mora biti napisana u JSON formatu
- Ova instrukcija je posebno zanimljiva na Windows OS-u gde postoje dve često upotrebljene i vrlo različite vrste *shell*-a: *cmd* i *powershell*.
- `SHELL` instrukcije mogu da se pojave više puta, pri čemu svaka naredna shell instrukcija pregazi sve prethodne i utiče na naredne instrukcije (`RUN`, `CMD`, `ENTRYPOINT`)

Dockerfile instrukcija [SHELL]

- Primer `SHELL` instrukcije:

```
FROM microsoft/windowsservercore

# Executed as cmd /S /C echo default
RUN echo default

# Executed as cmd /S /C powershell -command Write-Host default
RUN powershell -command Write-Host default

# Executed as powershell -command Write-Host hello
SHELL ["powershell", "-command"]
RUN Write-Host hello

# Executed as cmd /S /C echo hello
SHELL ["cmd", "/S", "/C"]
RUN echo hello
```

Materijali:

- https://docs.docker.com/engine/reference/builder/
- https://www.igordejanovic.net/courses/tech/docker/