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

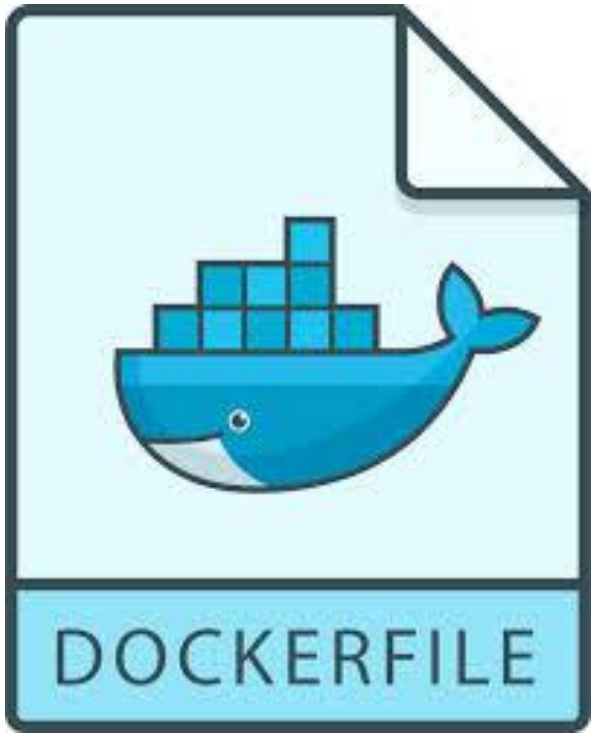
Računarstvo u oblaku

ms Helena Anišić

Zimski semester 2022/2023.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi


DOCKERFILE

Dockerfile

- 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

```
# 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 proađa 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
- FROM instrukcija inicijalizuje novi build stage i postavlje Base Image za dalje instrukcije.
- Base Image može biti bilo koja validna slika
  - Najjednostavnije je da se povuče neka slika za javnog repozitorijuma poput DockerHub-a
    - DockerHub je predefinisani repozitorijum sa kojeg se skidaju slike ukoliko se ne navede neka platforma

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

Dockerfile instrukcija [FROM]

- --platform tag
  - Opcioni tag koji može da specificira tačno platformu za koju da povuče sliku ukoliku FROM referencira multi-platformsku sliku
- tag/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 izbildanu u ovom stadijumu

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 Base image (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 hello-world slika

```
FROM scratch

COPY hello /

CMD ["/hello"]
```

Docker instrukcija [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, biće kreirana iako se ne koristi u naknadnim instrukcijama Dockerfile-a
- Predefinisani radni direktorijum je /
    - U praksi, ako se pravi dockerfile od scratch slike, workdir može vrlo lako da bude podešen od strane Base slike koju koristimo
- Najbolja praksa je definisati radni direktorijum da ne bude iznenađenja

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

Docker instrukcija [WORKDIR]

- WORKDIR instrukcija može da se koristi više puta u Dockerfile-u.
  - Ukoliko je obezbeđena relativna putanja, onda će novi radni direktorijum biti kreiran u odnosu na prethodni radni direktorijum
- PRIMER:

```
WORKDIR /a
WORKDIR b
WORKDIR c
RUN pwd
```

Dockerfile instrukcija [USER]

- 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 runtime-a za relevantne ENTRYPOINT i CMD komande
- Pre nego što se iskoristi komanda USER potrebno je kreiratidatog user-a
  - RUN groupadd -r <group_name> && useradd -r -g <group_name> <user_name>

Dockerfile instrukcija [COPY]

- COPY instrukcija kopira fajlove/direktorijume iz <src> i dodaje ih u fajlsistem kontejnera na putanju <dest>
- <dest> putanja je apsolutna ili relativna u odnosu na WORKDIR
  - <WORKDIR>/ relativeDir/
    - COPY test.txt relativeDir/
  - /absoluteDir/
    - COPY test.txt /absoluteDir/

```
COPY [--chown=<user>:<group>] <src>... <dest>
COPY [--chown=<user>:<group>] ["<src>",... "<dest>"]
```

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 cache i izaziva ponovno build-ovanje datog sloja slike, kao i svih narednih

Dockerfile instrukcija [COPY]

- COPY --link
  - Source fajlovi se kopiraju u prazan destination direktorijum
  - Taj direktorijum se pretvara u sloj koji se linkuje povrh prethodnog stanja
  - Fajlovi ostaju nezavisni i ne postaju invalidirani kada se komande u prethodnom sloju izmene
- PRIMER:
  - #1 i #2 instrukcije izvršavaju isti posao

```
# 1
FROM alpine
COPY --link /foo /bar

#2
FROM alpine
FROM scratch
COPY /foo /bar
```

Dockerfile instrukcija [ADD]

- 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 je apsolutna ili relativna u odnosu na WORKDIR
  - <WORKDIR>/ relativeDir/
  - /absoluteDir/

```
ADD [--chown=<user>:<group>] [--checksum=<checksum>] <src>... <dest>
ADD [--chown=<user>:<group>] ["<src>",... "<dest>"]
```

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

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.

PRIMERI

- Primer bez definisanog WORKDIR
  - Prebacivanje fajla
    - sa /
    - bez /
    - sa .
    - samo /
  - Prebacivanje foldera
    - sa /
    - bez /
    - sa .
    - samo /
- Primeri sa definisanim WORKDIR
  - isto

Dockerfile instrukcija [CMD]

- CMD instrukcija specificira naredbu koja treba da se izvrši kada docker kontejner započne sa izvršavanjem
  - Glavna poenta CMD instrukcije je da pokrene softver koji se nalazi se u kontejneru.
  - Primer:
    - Pokretanje .exe fjla
    - 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

Dockerfile instrukcija [CMD]

1. CMD ["executable","param1","param2"] (*exec* form, this is the preferred form)
2. CMD ["param1","param2"] (as *default parameters to ENTRYPOINT*)
3. CMD command param1 param2 (*shell* form)

- Exec forma (1.) se parsira kao JSON array i to znači da mora biti unutar dvostrukih navodnika
- U drugoj opciji (2.) se ne koristi executable, te se ona(executable) mora sepcificirati u okviru ENTRYPOINT-a
- Treća opcija (3.) - shell form indukuje shell processing
  - Exec forma to ne radi - ne inicira shell processing
    - CMD [“sh”, “-c”, “echo $HOME”] - za pokretanje preko shell-a
- Specificiranjem argumenata kod *docker run* naredbe možemo pregaziti CMD naredbu napisanu u Dockerfile-u

Dockerfile instrukcija [ENTRYPOINT]

- Exec forma
  - ENTRYPOINT ["executable", "param1", "param2"]
- Shell forma
  - ENTRYPOINT command param1 param2
- 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

Dockerfile instrukcija [RUN]

- RUN instrukcija izvršava bilo kakvu komandu u novom sloju povrh trenutne slike i komituje rezultat.
  - Rezultujuća komitovana slika će se koristiti za naredni korak u Dockerfile-u.
  - Raslojavanje docker slike pomoću RUN instrukcije i generisanje komitova odgovara konceptu dockera jer su komitovi jeftiniji i kontejneri mogu da budu kreirani sa bilo koje tačke u istoriji slike.

1. RUN <command>
2. RUN ["executable", "param1", "param2"]

CMD VS. RUN

- RUN i CMD nisu iste instrukcije
- RUN izvršava instrukciju tokom build-ovanja slike i komituje rezultat izvršavanja
- CMD ne izvršava istrukciju tokom build-ovanja slike, već tokom pokretanja kontejnera

Docker instrukcija [LABEL]

- Dodaje metapodatke slici kontejnera
  - Par ključ:vrednost
- Slika može da ima više od jedne labele
- Labele koju su uključene u Base Image ili roditeljsku siku se nasleđuju
- Za pregled labele slike koristiti naredbu ***docker image inspect***

```
LABEL version="1.0"
LABEL description="This text illustrates \
that label-values can span multiple lines."

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

Docker instrukcija [EXPOSE]

- EXPOSE instrukcija informiše Docker o tome da kontejner sluša na određenom mrežnom portu tokom runtime-a.
  - Može se specificirati TCP/UDP
    - TCP je predefinisani
- EXPOSE instrukcija ne objavljuje zapravo port, već više služi kao dokumentacija
- Za stvarno objavljivanje porta koristit **-p** opciju prilikom docker run naredbe

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

Dockerfile instrukcija [VOLUME]

- VOLUME instrukcija kreira mount point na definisanoj putanje u kontejneru
- Sa te putanje se mountuju fajlovi/direktorijumi svaki put kada se pokrene kontejner
- Može da bude JSON array ili string
    - VOLUME ["/var/log/"]
    - VOLUME /var/log

Dockerfile instrukcija [ARG]

- ARG instrukcija definiše promenljivu koju korisnik prosleđuje tokom poziva *docker build* komande upotrebom **--build-arg <varname>=<value>** opcije
  - Ako korisnik prosledi argument koji nije definisan u dockerfile-u, build će da izbaci upozorenje
- Moguće je definisani više ARG instrukcija
- ARG instrukcija

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

Dockerfile instrukcija [ARG]

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

- Šta se desi ako nema predefinisane vrednosti i ništa se ne prosledi tokom build naredbe?

Dockerfile instrukcija [ENV]

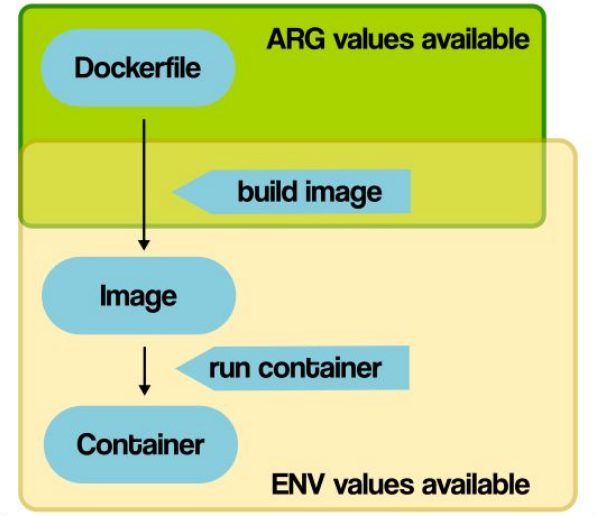
- Šta su environment varijable?
- ENV instrukcija postavlja environment varijablu <key> na vrednost <value>
- Moguće je postavljati više varijabli odjednom
- Varijabla ostaje sačuvana kada se kontejner pokrene na osnovu date slike
    - docker inspect naredba daje uvid u varijable
    - docker run –env <key>=<value> omogućava izmenu varijabli tokom pokretanja kontejnera

```
ENV 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 za buduće kontejnere, a ARG je za izgradnju slike kontejnera
- ENV je pre svega namenjeno za predefinisane vrednosti budućih varijabli
  - Rad dokerizovane aplikacije može da pristupi environment varijablama
  - Odličan način da se prosleđue konfiguracione vrednosti u projekat
- ARG vrednosti nisu dostupne nakon što se slika izbilda
  - Kontejner koji radi nema pristup ARG vrednostima
- ARG i ENV se preklapaju tokom image build-a

ARG VS ENV


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

Dockerfile instrukcija [STOPSIGNAL]

- STOPSIGNAL instrukcija postavlja signal za sistemski poziv koji će biti poslat kontejneru prilikom exit-a
- Signal može biti naziv signala u formatu SIG<NAME> npr SIGKILL
- Signal može biti unsigned broj koji odgovara poziciji u syscall tabeli kernels (9)
    - Predefinisani je SIGTERM
- Predefinisani stopsignal može da bude pregažen po pojedinačnom kontejneru upotrebom --stop-signal opcije na docker run i docker create naredbe

Dockerfile instrukcija [HEALTHCHECK]

- HEALTHCHECK instrukcija ima dve forme:
  - HEALTHCHECK [OPTIONS] CMD command
    - Proverava zdravlje kontejnera pokretanjem naredbe unutar kontejnera
  - HEALTHCHECK NONE
    - Onesposobi bilo kakav healthcheck nasleđen od strane Base slike
- HEALTHCHECK instrukcija govori dockeru kako da testira kontejner kako bi proverio da 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 health status
  - Inicijalno je starting
- Kada health check prođe, postaje healthy
- Posle nekoliko uzastopnih neuspeha postaje unhealthy
- Može da postoji samo jedna healthcheck instrukcija u dockerfile-u

Dockerfile instrukcija [HEALTHCHECK]

- Opcije:
  - --interval=DURATION (default: 30s)
  - --timeout=DURATION (default: 30s)
  - --start-period=DURATION (default: 0s)
  - --retries=N (default: 3)
- Primer koji proverava na svakih 5 minuta da veb server može da servira glavnu stranicu sajta u roku od 3 sekunde:

```
HEALTHCHECK --interval=5m --timeout=3s \
  CMD curl -f http://localhost/ || exit 1
```

Dockerfile instrukcija [HEALTHCHECK]

- Interval
  - Vremenski interval između svake provere zdravlja, kao i vremenski interval između pokretanja kontejnera i prve provere zdravlja
- Timeout
  - Vremeski period nakon čijeg isteka provera zdravlja se smatra neuspešnom
- Retries
  - Broj ponavljanja provere zdravlja pre nego se provera smatra neuspešnom
- Start period
  - 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.

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 [ONBUILD]

- ONBUILD instrukcija dodaje trigger instrukciju na sliku kontejnera
  - Taj trigger će se izvršiti kada se slika kontejnera bude koristila kao Base slika za neku drugu sliku kontejnera
- Kako radi ONBUILD:
  - Kada se naiđe na ONBUILD instrukciju, builder dodaje trigger-e 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 dve triggere date slike pod ključem OnBuild
  - Kada se data slika koristi kao Base slika u okviru naredbe FROM, builder traži ONBUILD triggere i izvršava ih u istom redosledu kako su navedeni. Ako neki trigger ne uspe da se izvrši, akcija se abortira. Ako se uspešno izvrši, nastavlja se na narednu instrukciju nakon FROM-
  - Triggeri se ne nasleđuju.

```
ONBUILD <INSTRUCTION>
```

Dockerfile instrukcija [ONBUILD]

- https://blog.frankel.ch/onbuild-overlooked-docker-directive/
- https://github.com/carlossg/docker-maven/blob/8ab542b907e69c5269942bcc0915d8dffcc7e9fa/jdk-8/onbuild/Dockerfile

Dockerfile instruction [SHELL]

- SHELL instrukcija omogućava da se pregazi predefinisani shell za shell formu komande.
  - Predefinisani shell za linux je ["/bin/sh", "-c"]
  - Predefinisani shel na Windowsu je ["cmd", "/S", "/C"]
- Shell instrukcija mora biti napisana u JSON formatu
- Ova instrukcija je posebno zanimljiva na Windows OS-u gde postoje dva 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]

```
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/