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

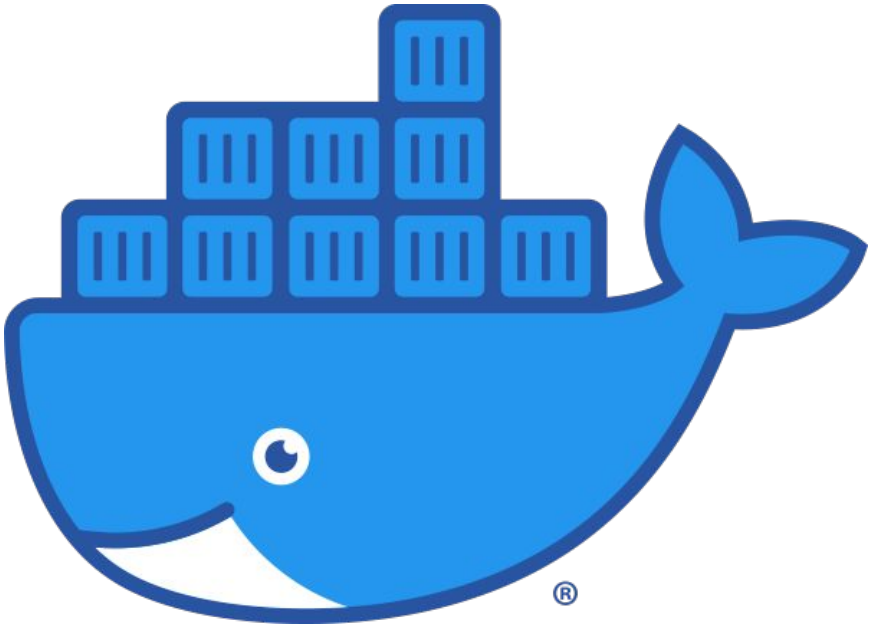
Računarstvo u oblaku

ms Helena Anišić

Zimski semester 2025/2026.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi



Docker - osnovni pojmovi

- Docker je otvorena platforma za razvoj, distribuciju i izvršavanje aplikacija
  - razvijena 2013. godine i napisana u programskom jeziku Go
- Docker nudi mogućnost pakovanja i pokretanja aplikacija u labavo izolovanom okruženju zvanom - **KONTEJNER**.
  - Docker je tehnologija za kontejnerizaciju.
  - Alat za kreiranje i upravljanje kontejnerima.

Docker - osnovni pojmovi

- Kontejner je standardizovano parče softvera.
- **KONTEJNER = KOD + ZAVISNOSTI** (biblioteke, okruženja, itd.)
  - Primer: NodeJS kod + NodeJS runtime
- Isti kontejner će uvek rezultovati istom aplikacijom i istim načinom izvršavanja.
  - Bez obzira gde ili ko pokreće aplikaciju.
  - Primer: Ugrađeno hlađenje u kontejneru

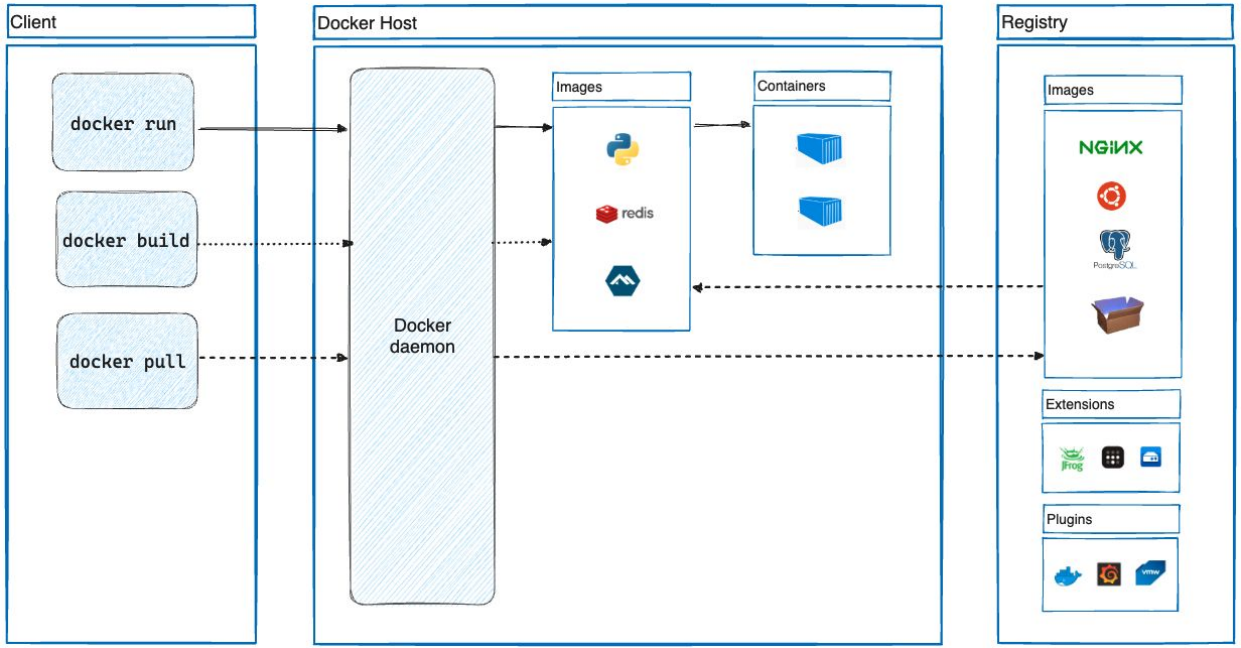
Docker - osnovni pojmovi

- Zašto nam trebaju nezavisni, standardizovani aplikacioni paketi?
    1. Različita produkciona i razvojna okruženja.
    2. Različita razvojna okruženja u okviru firme/tima.
    3. Alati i verzije koje se sudaraju između različitih projekata.

Docker - arhitektura

- Docker izvršno okruženje (engl. *Docker Engine*)
  - predstavlja aplikaciju koja prati **klijent-server** arhitekturu koja se sastoji od sledećih komponenti:
    - **docker pozadinski proces** (engl. *docker daemon*) - pozadinski proces operativnog sistema domaćina koji sadrži sve potrebne metode za rad sa kontejnerima, slikama, mrežama i diskovima
      - predstavlja server u arhitekturi
      - pristupa mu se naredbom `dockerd`
    - **REST API** - koji predstavlja specifikaciju interfejsa sa metodama koje koriste docker klijenti kako bi komunicirali sa docker pozadinskim procesom
    - **docker klijent** - program koji se izvšava kroz terminal operativnog sistema.
      - pristupa mu se direktno preko naredbe `docker`
      - docker klijent i *daemon* mogu, ali i ne moraju biti na istom sistemu

Docker arhitektura



Docker - arhitektura

- Docker *daemon* (`dockerd`)
  - čeka na Docker API zahteve
  - upravlja docker objektima kao što su slike, kontejneri, mreže i volume-i
  - *daemon* može da komunicira sa drugim *daemon*-om radi upravljanja Docker servisima

Docker - arhitektura

- Pokretanje pozadinskog procesa
  - Uobičajeno se pokreće automatski prilikom podizanja OS-a
  - Uz korišćenje programa samog OS-a, ako je docker instaliran kao servis
    - `$ sudo systemctl start docker`
    - `$ sudo service docker start`
  - ručno
    - `$ dockerd`

Docker - arhitektura

- Docker klijent (`docker`)
  - primarni način za interakciju sa Docker-om
  - kada se izvrši komanda poput `docker run` u okviru terminala, klijent šalje ovu komandu docker *daemon*-u koji ih zatim izvršava
  - docker koristi Docker API
  - docker klijent može da komunicira sa više *daemon*-a
  - *Docker Compose* je takođe docker klijent
    - omogućava lakši rad sa aplikacijama koje se sastoje iz više klijenata

Docker - arhitektura

- Docker registar je skladište slika kontejnera koje se mogu koristiti uz docker
  - JAVNI REGISTAR
    - Docker Hub https://hub.docker.com/
      - podrazumevani registar za Docker
      - Docker će bez dodatnog podešavanja pokušati da u ovom registru pronađe slike
  - PRIVATNI REGISTAR
    - Docker Hub
    - Mogu se postaviti na bilo koji server dostupan na mreži

Docker - instalacija

- Postoji više načina za instalaciju docker-a
  - Instalacija *Docker Desktop*-a
    - https://docs.docker.com/get-docker/
    - sadrži sve potrebne alate za rad sa dockerom + GUI aplikaciju
  - Instalacija *Docker Engine*-a
    - https://docs.docker.com/engine/install/
  - Objašnjenje razlike između dva načina instalacije za Linux platformu na sledećem linku: https://docs.docker.com/desktop/faqs/linuxfaqs/#what-is-the-difference-between-docker-desktop-for-linux-and-docker-engine

Docker klijent [run] - *Hello World*

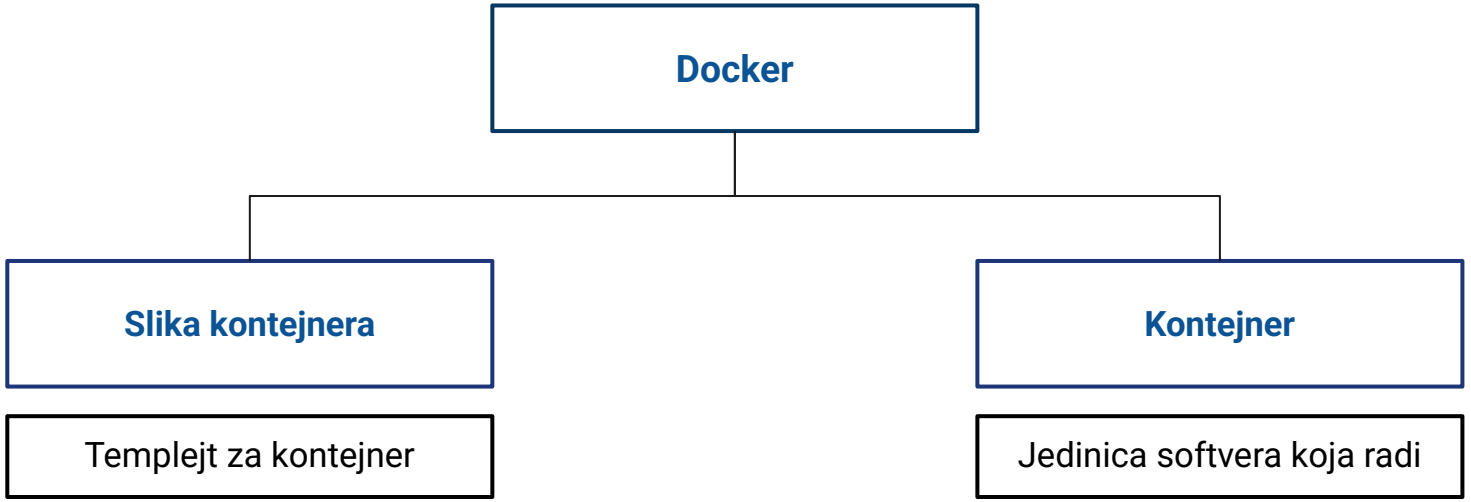
```
$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.
```

Docker klijent [run] - *Hello World*

- *Hello World* primer
  - provera da li docker radi
  - ukoliko slika nije pronađena lokalno, docker je preuzima iz podrazumevanog registra
    - podrazumevano je to DockerHub

```
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.
…
```

Docker - osnovni pojmovi



Docker klijent [pull]

- Preuzima sliku ili repozitorijum iz registra (*DockerHub* je predefinisani registar)
  - naredba `$ docker pull [OPTIONS] NAME[:TAG|@DIGEST]`
- Ukoliko se ne navede tag, *Docker Engine* uzima **:latest** tag kao predefinisani

```
$ docker pull ubuntu

Using default tag: latest
latest: Pulling from library/ubuntu
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:e7d38b3517548a1c71e41bffe9c8ae6d6d29546ce46bf62159837aad072c90aa
Status: Downloaded newer image for ubuntu:latest
```

Docker klijent [pull]

- Preuzimanje slike preko *digest*-a
  - upotreba tagova za preuzimanje slika podrazumeva najnoviju verziju date slike
  - u nekim situacijama potrebno je preuzeti tačno određenu verziju neke slike

```
$ docker pull ubuntu:20.04

20.04: Pulling from library/ubuntu
16ec32c2132b: Pull complete
Digest: sha256:82becede498899ec668628e7cb0ad87b6e1c371cb8a1e597d83a47fac21d6af3

$ docker pull ubuntu@sha256:82becede498899ec668628e7cb0ad87b6e1c371cb8a1e597d83a47fac21d6af3
```

Docker klijent [images]

- Pregled postojećih slika kontejnera
  - naredba `$ docker images [OPTIONS] [REPOSITORY[:TAG]]`
- OPTIONS: --filer, --format, ..

```
$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
<none>        <none>    8794538e13a1   24 minutes ago   1GB
node          latest    cb9aad9080ca   23 hours ago     994MB
ubuntu        latest    a8780b506fa4   6 days ago       77.8MB
python        latest    00cd1fb8bdcc   2 weeks ago      932MB
hello-world   latest    feb5d9fea6a5   13 months ago    13.3kB
```

Docker klijent [run]

- Pokretanje kontejnera na osnovu preuzete slike
  - naredba `$ docker run [OPTIONS] IMAGE [COMMAND] [ARG ...]`
  - kreira novi kontejner za razliku od naredbe **`start`**
- OPTIONS = -it
  - Pokretanje kontejnera u interaktivnom režimu sa otvorenim terminalom
  - Opcija -i spaja se na STDIN
  - Opcija -t alocira pseudo terminal

```
$ docker run -it node
Welcome to Node.js v19.0.1.
Type ".help" for more information.
> 1+1
2
```

Docker klijent [run]

- OPTIONS = --name
  - definisanje naziva kontejnera
- OPTIONS = --rm
  - omogućava automatsko brisanje kontejnera čim se zaustavi

Docker klijent [ps]

- Prikaz kontejnera (podrazumevano samo one kontejnere koji rade)
  - naredba `$ docker ps [OPTIONS]`
  - svako pokretanje slike kreira novi kontejner
- OPTIONS = --all, -a
  - prikazuje i kontejnere koji su trenutno pokrenuti, kao i one koju su zaustavljeni
- OPTIONS = --no-trunc
  - potpuni ispis
- OPTIONS = --size, -s
  - Prikazuje veličinu zauzetog skladišta
    - size - količina podataka na disku koja je upotrebljena za *writable* sloj kontejnera
    - virtual size - ukupna količina prostora na disku koja se koristi za read-only podatke slike upotrebljene od strane kontejnera i *writable* sloja

Docker klijent [ps]

```
$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS     PORTS NAMES

$ docker ps -a
CONTAINER ID   IMAGE           COMMAND     CREATED           STATUS                    PORTS NAMES
eecf1a0eca93   hello-world   "/hello"   21 minutes ago   Exited (0) 21 minutes ago      gallant
dfb15341f798   hello-world   "/hello"   22 minutes ago   Exited (0) 22 minutes ago       moser
aee9d8250ed9   hello-world   "/hello"   26 minutes ago   Exited (0) 26 minutes ago      poitras
```

Docker klijent [ps]

- OPTIONS = --filter, -f
  - format je u obliku key=value
  - može da se prosledi više filtera (primer. --filter "foo=bar" --filter "bif=baz")

```
$ docker ps --filter "name=nostalgic_stallman"
$ docker ps -a --filter 'exited=0'
$ docker ps -a --filter 'exited=0'
$ docker ps -a --filter 'exited=137'
$ docker ps --filter status=running
$ docker ps -f before=9c3527ed70ce
```

Docker klijent [ps]

Filter	Description
`id`	Container's ID
`name`	Container's name
`label`	An arbitrary string representing either a key or a key-value pair. Expressed as `<key>` or `<key>=<value>`
`exited`	An integer representing the container's exit code. Only useful with `--all`.
`status`	One of `created`, `restarting`, `running`, `removing`, `paused`, `exited`, or `dead`
`ancestor`	Filters containers which share a given image as an ancestor. Expressed as `<image-name>[:<tag>]`, `<image id>`, or `<image@digest>`
`before` or `since`	Filters containers created before or after a given container ID or name
`volume`	Filters running containers which have mounted a given volume or bind mount.
`network`	Filters running containers connected to a given network.
`publish` or `expose`	Filters containers which publish or expose a given port. Expressed as `<port>[/<proto>]` or `<startport-endport>/[<proto>]`
`health`	Filters containers based on their healthcheck status. One of `starting`, `healthy`, `unhealthy` or `none`.
`isolation`	Windows daemon only. One of `default`, `process`, or `hyperv`.
`is-task`	Filters containers that are a "task" for a service. Boolean option (`true` or `false`)

Docker klijent [ps]

- OPTIONS = --format
  - Opcija omogućava uređen ispis informacija o kontejnerima upotrebom Go template-a
  - Ukoliko se u okviru templejta navede **table**, ispis će uključivati i zaglavlja tabele
  - Za razmak između pojedinih kolona u ispisu potrebno je staviti **\t**

```
$ docker ps --format "{{.ID}}"
$ docker ps --format "{{.ID}} \t {{.Image}}"
$ docker ps --format "table {{.ID}} \t {{.Image}}"
```

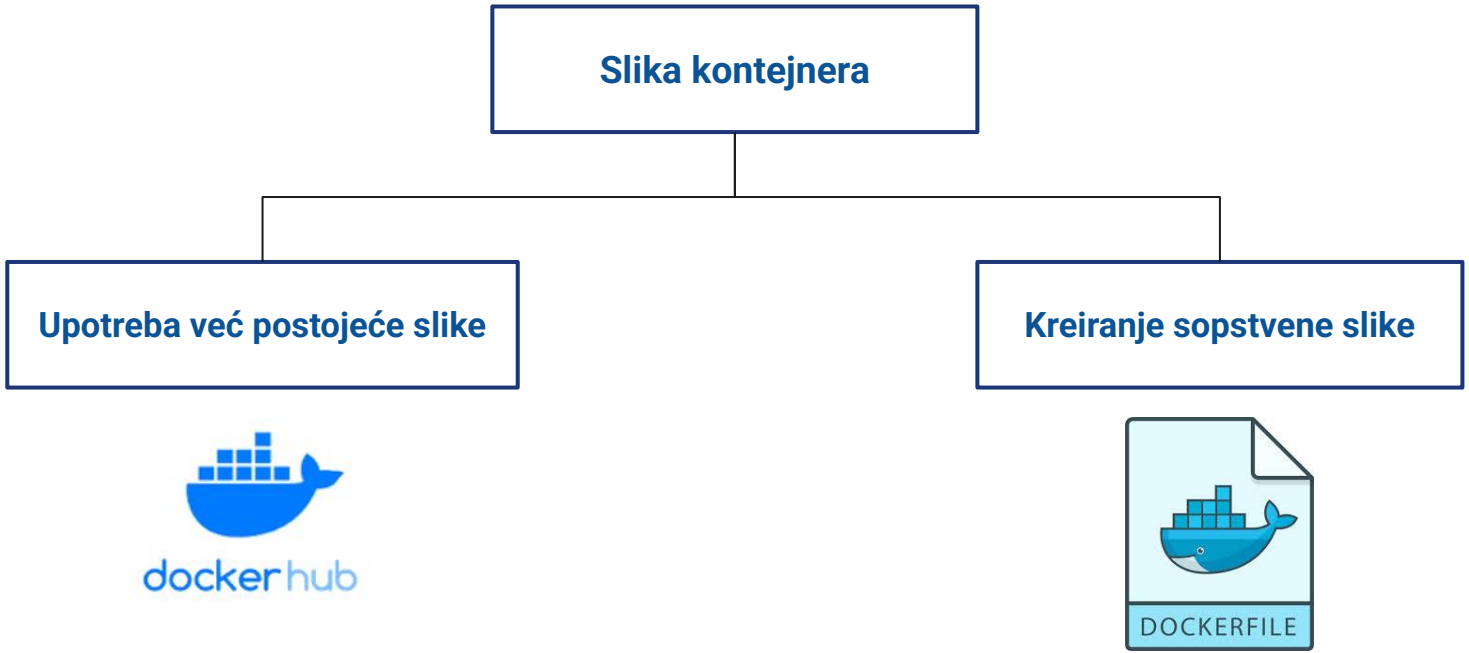
Docker klijent [ps]

```
$ docker ps -a

CONTAINER ID   IMAGE           COMMAND      CREATED           STATUS                     PORTS  NAMES
eecf1a0eca93   hello-world   "/hello"   21 minutes ago   Exited (0) 21 minutes ago      gallant
dfb15341f798   hello-world   "/hello"   22 minutes ago   Exited (0) 22 minutes ago       moser
aee9d8250ed9   hello-world   "/hello"   26 minutes ago   Exited (0) 26 minutes ago      poitras

$docker ps -a --format "table {{.ID}} \t {{.Image}}"

CONTAINER ID       IMAGE
8abcdfc35f46       hello-world
```

Docker klijent [ps]

Placeholder	Description
`.ID`	Container ID
`.Image`	Image ID
`.Command`	Quoted command
`.CreatedAt`	Time when the container was created.
`.RunningFor`	Elapsed time since the container was started.
`.Ports`	Exposed ports.
`.State`	Container status (for example; “created”, “running”, “exited”).
`.Status`	Container status with details about duration and health-status.
`.Size`	Container disk size.
`.Names`	Container names.
`.Labels`	All labels assigned to the container.
`.Label`	Value of a specific label for this container. For example `'{{.Label "com.docker.swarm.cpu"}}'`
`.Mounts`	Names of the volumes mounted in this container.
`.Networks`	Names of the networks attached to this container.

Docker - osnovni pojmovi



Primer

- Pisanje Dockerfile-a za NodeJS aplikaciju

```
FROM node

WORKDIR /app

COPY . /app

RUN npm install

EXPOSE 80

CMD ["node", "server.js"]
```

Docker klijent [build]

- Kreiranje docker slike na osnovu Dockerfile-a
  - naredba `$ docker build [OPTIONS] PATH | URL | -`
  - tipično se docker slike preuzete sa *DockerHub*-a nadograđuju
    - u te svrhe se koristi Dockerfile

```
$ docker build .
Sending build context to Docker daemon  13.82kB
Step 1/6 : FROM node
---> cb9aad9080ca
Step 2/6 : WORKDIR /app
---> Using cache
---> 341a97a3ea14
Step 3/6 : COPY . /app
---> Using cache
---> 06434e267b35
Step 4/6 : RUN npm install
---> Using cache
---> f8490054a347
Step 5/6 : EXPOSE 80
---> Using cache
---> 3f660d3530bb
Step 6/6 : CMD ["node", "server.js"]
---> Using cache
---> 8794538e13a1
Successfully built 8794538e13a1
```

Docker - osnovni pojmovi

- Slike kontejnera su slojevite arhitekture
  - Svaka instrukcija Dockerfile-a kreira jedan sloj
    - Utiče na brzinu kreiranja slike
    - Prilikom ponovnog build-ovanja slike proverava se da li desila neka izmena
      - Ako nije - koristi se keširani sloj
      - Ako jeste - ponovo se build-a dati sloj
        - Svi slojevi nakon njega se takođe ponovo builduju
          - VAŽAN REDOSLED INSTRUKCIJA
- Slike kontejnera su **READ-ONLY**
  - ukoliko se izmeni kod potrebno je ponovo build-ovati sliku, jer u suprotnom izmene neće biti primenjene

Docker - imenovanje i tagovanje slika i kontejnera

- Imenovanje slika kontejnera
  - naziv slike kontejnera sastoji se iz dva dela **name : tag**
    - name - definiše grupu (npr. node)
    - tag - definiše specijalizovanu sliku iz grupe (npr. 14)
  - naredba `$ docker  build -t name:tag PATH | URL | -`
- Imenovanje kontejnera
  - naredba `$ docker run --name [naziv] [imageID]`

Docker klijent [start]

- Pokretanje prethodno zaustavljenog kontejnera
  - naredba `$ docker start [OPTIONS] CONTAINER [CONTAINER ... ]`
  - navodi se **ID** kontejnera ili **naziv** kontejnera

```
$ docker start  youthful_chatele
youthful_chatele
```

Docker - attached VS. detached kontejner

- **Attached** način rada
  - terminal čeka na output iz kontejnera (primer console.log() naredba)
  - `$ docker run` naredba predefinisano koristi attached režim rada
  - ukoliko je potrebno `$ docker run` naredbu izvršiti u detached režimu, dodati opciju **`-d`**
- **Detached** način rada
  - terminal ne čeka na output iz kontejnera
  - kontejner radi u pozadini
  - za naknadno attach-ovanje koristiti `$ docker attach CONTAINER`

```
$ docker attach youthful_chatele
```

Docker klijent [logs]

- Prikazuje logove kontejnera
  - naredba `$ docker logs [OPTIONS] CONTAINER`
- OPTIONS = --follow, -f
  - ispisivanje narednih logova - ponovno attach-ovanje na kontejner
- OPTIONS = --since
  - ispisivanje logova nakon određenog timestamp-a
- OPTIONS = --until
  - ispisivanje logova pre određenog timestamp-a

```
$ docker logs youthful_chatele
```

Primer

- Pisanje Dockerfile-a za Python aplikaciju

```
FROM python

WORKDIR /app

COPY . /app

CMD ["python", "rng.py"]
```

Docker klijent - interaktivni režim rada

- Attached režim rada omogućava da se na terminalu prati ispis kontejnera
  - Da bi se omogućio i unos preko STDIN-a potrebno je dodati sledeće opcije:
    - **-i -t** prilikom naredbe `$ docker run`
      - `$ docker run -it [IMAGE]`
    - **-i -a** prilikom naredbe `$ docker start`
      - `$ docker start -ia [CONTAINER]`

Docker klijent [rm]

- Brisanje kontejnera
  - naredba `$ docker rm [OPTIONS] CONTAINER [CONTAINER ... ]`
  - briše kontejnere čiji su ID-jevi ili nazivi prosleđeni
    - brisanje više kontejnera - navođenje kontejnera sa razmakom između
  - predefinisano je da se ne mogu obrisati kontejneri koji su pokrenuti
- OPTIONS = --force , -f
  - omogućava uklanjanje kontejnera koji su pokrenuti (koristi SIGKILL)
- Ako pokrenemo kontejner sa opcijom -rm on će se automatski obrisati kada se zaustavi

```
$ docker rm 80153f8c53f8
80153f8c53f8
```

Docker klijent - Kombinovanje naredbi

- Kombinovanje se može izršiti sa bilo kojom Docker klijent naredbom

```
$ docker rm $(docker ps -a -q -f status=exited)
5236ac8a8f1f
4098e9ee670b
9b333327baa9
c52b6c5fba60
987c4b4918cd
1f28605c55d8
cc3a3baa4ad4
4063c173e851
0eeb596d5f4c
```

Docker klijent [rmi]

- Brisanje slika kontejnera
  - naredba `$ docker rmi [OPTIONS] IMAGE [IMAGE ...]`
  - slika kontejnera može da se obriše samo ukoliko ne postoji nijedan kontejner koji koristi tu sliku
    - zaustavljen kontejner se i dalje smatra kontejnerom koji koristi neku slike, te je potrebno i zaustavljene kontejnere obrisati, pa tek nakon toga obrisati slike kontejnera

```
$ docker rmi ubuntu
Untagged: ubuntu:latest
Untagged: ubuntu@sha256:4b1d0c4a2d2aaf63b37111f34eb9fa89fa1bf53dd6e4ca954d47caebca4005c2
Deleted: sha256:a8780b506fa4eeb1d0779a3c92c8d5d3e6a656c758135f62826768da458b5235
Deleted: sha256:f4a670ac65b68f8757aea863ac0de19e627c0ea57165abad8094eae512ca7dad
```

Docker klijent [container prune]

- Brisanje nekorišćenih kontejnera (stopiranih)
  - naredba `$ docker container prune [OPTIONS]`

```
$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
c17cdbd9445d026d4bee6293f1e9fb9d1c455915c7fdfb01e78edbf6a05cd326

Total reclaimed space: 9.084MB
```

Docker klijent [container prune]

- OPTIONS = --force , -f
  - na silu obriše sve stopirane kontejnere, bez prethodnog upita u kome se traži potvrda korisnika
- OPTIONS = --filter
  - format je u obliku **key=value**
  - može da se prosledi više filtera (primer. --filter "foo=bar" --filter "bif=baz")
  - Trenutno podržani filteri:
    - until (<timestamp>) - briše sve kontejnere koji su kreirani pre određenog timestamp-a
    - label (label=<key>, label=<key>=<value>, label!=<key>, or label!=<key>=<value>) - briše sve kontejnere sa (ili bez, u slučaju label!=...) specificiranom labelom

```
$ docker container prune --force --filter "until=5m"
$ docker container prune --force --filter "until=2017-01-04T13:10:00"
```

Docker klijent [image prune]

- Brisanje nekorišćenih slika kontejnera i slojeva
  - naredba `$ docker image prune [OPTIONS]`
  - briše sve *dangling* slojeve slika
    - slojeve koji više nisu vezani za konkretne slike kontejnera i koji nisu tagovani
    - detaljnije objašnjenje https://www.howtogeek.com/devops/what-are-dangling-docker-images/
  - moguće je obrisati i sve slike koje se ne koriste

```
$ docker image prune
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] y
Deleted Images:
deleted: sha256:b0e0fb50d2bda9a6a643027436202aef56c9c86c8e752e0da391e38aa4a41e11
```

Docker klijent [image prune]

- OPTIONS = --all , -a
- OPTIONS = --filter
  - isto kao kod [container prune]
- OPTIONS = --force, -f
  - isto kao kod [container prune]

```
$ docker image prune -a
WARNING! This will remove all images without at least one container associated to them.
Are you sure you want to continue? [y/N] y
Deleted Images:
untagged: hello-world:latest
untagged: hello-world@sha256:e18f0a777aefabe047a671ab3ec3eed05414477c951ab1a6f352a06974245fe7
```

Docker klijent [system prune]

- Brisanje svih nekorištenih kontejnera, mreža, slika (i *dangling* i nereferenciranih) i opciono Docker skladišta (engl. *volumes*)
  - naredba `$ docker system prune [OPTIONS]`

```
$ docker system prune
WARNING! This will remove:
 - all stopped containers
 - all networks not used by at least one container
 - all dangling images
 - all dangling build cache

Are you sure you want to continue? [y/N] y
```

Docker klijent [system prune]

- OPTIONS = --all , -a
  - Briše sve slike nekorištene slike, ne samo *dangling*
- OPTIONS = --filter
  - isto kao kod [container prune]
- OPTIONS = --force, -f
  - isto kao kod [container prune]
- OPTIONS = --volumes
  - Briše i podatke u Docker skladištu

Docker klijent [stats]

- Proverava zauzeća resursa kontejnera
  - naredba `$ docker stats [OPTIONS] [CONTAINER ... ]`

```
$ docker stats

CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT     MEM %     NET I/O
68e234c3f992   dreamy_mirzakhani   0.00%     19.84MiB / 7.677GiB   0.25%     182kB / 2.44kB
```

Docker klijent [inspect]

- Uvid u detalje pokrenutog kontejnera
  - naredba `$ docker inspect [OPTIONS] NAME|ID [NAME|ID ...]`

```
$ docker inspect dreamy_mirzakhani
[
    {
        "Id": "68e234c3f9926dba5bdff7ae36f0b93482e4d942e6c44d4b60203f4b2a65b33d",
        "Created": "2022-11-09T17:01:57.63262032Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "node",
            "Server.js"
…
```

Docker klijent - dokumentacija

- Dodatne naredbe kao i opcije svih naredbi dokumentovane su:
  - Docker dokumentacija na sledećem linku: https://docs.docker.com/reference/
  - naredba `--help` nakon bilo koje naredbe

```
$  docker run --help

Usage:  docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
Create and run a new container from an image
Aliases:
  docker container run, docker run
Options:
  -a, --attach list                    Attach to STDIN, STDOUT or STDERR
  -d, --detach                         Run container in background and print container ID
```

Materijali

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