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

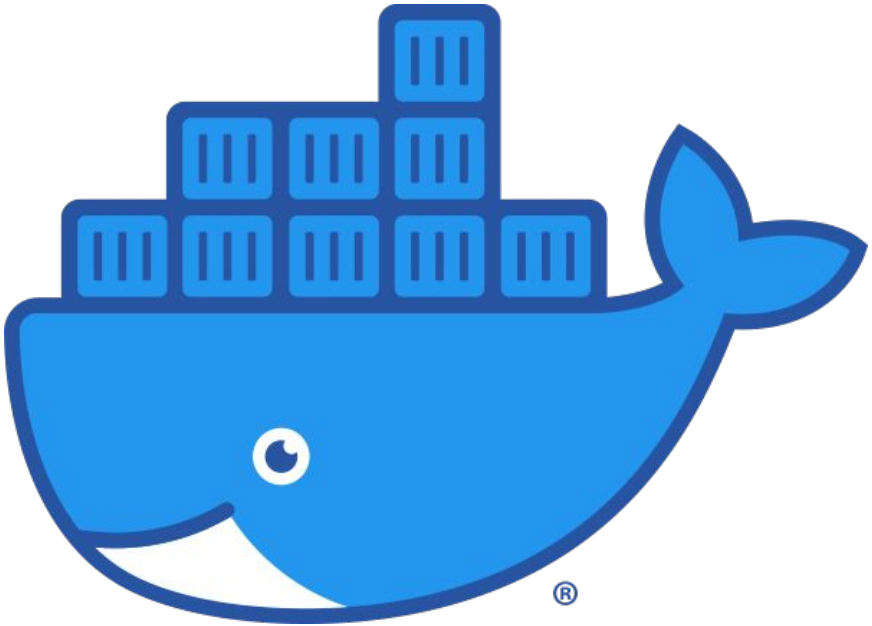
Računarstvo u oblaku

ms Helena Anišić

Zimski semester 2022/2023.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi



Šta je Docker?

- Docker je skup PaaS (platform-as-a-service) produkata koji koriste virtualizaciju na nivou OS-a sa ciljem dostavljanja softvera u paketima koji se zovu kontejneri.
- Kontejneri su izolovani od ostatka OS-a na kojem se izvršavaju.
- Docker je tehnologija za kontejnerizaciju. Alat za kreiranje i upravljanje kontejnerima.
- Razvije je 2013. godine i napisan u programskom jeziku Go.

Kontejner

Standardizovano parče softvera.

CONTAINER = CODE + DEPENDECIES

Prednosti? -> Isti kontejner će uvek rezultovati istom aplikacijom i istim načinom izvršavanja. Bez obzira gde ili ko pokreće aplikaciju.

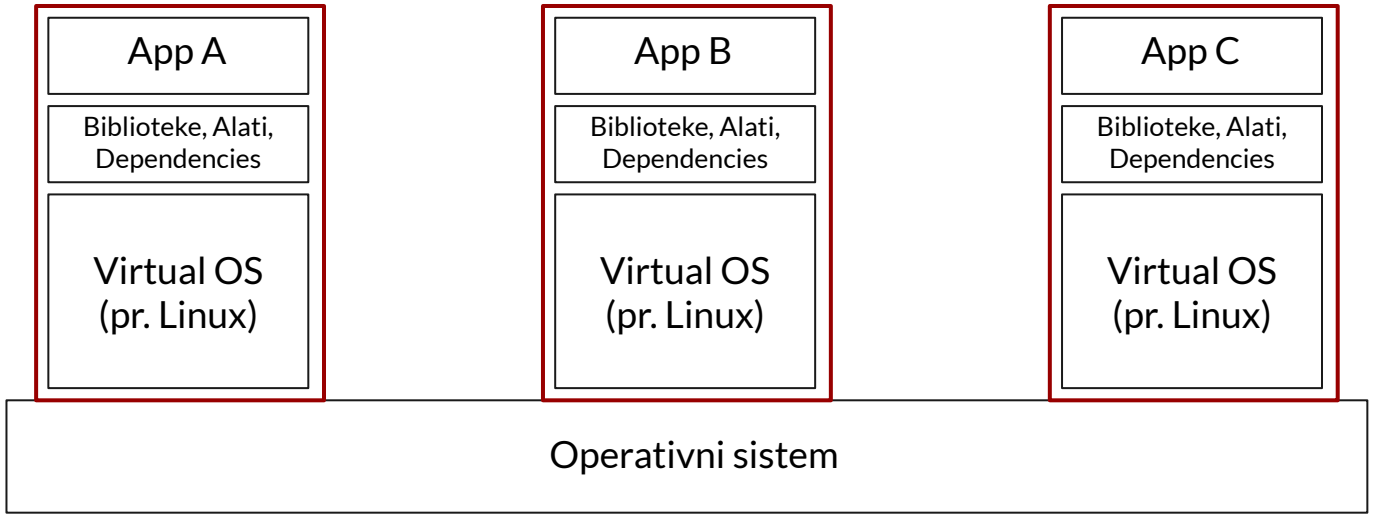
- Primer sa ugrađenim hlađenjem u kontejneru

Čemu kontejneri?

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.

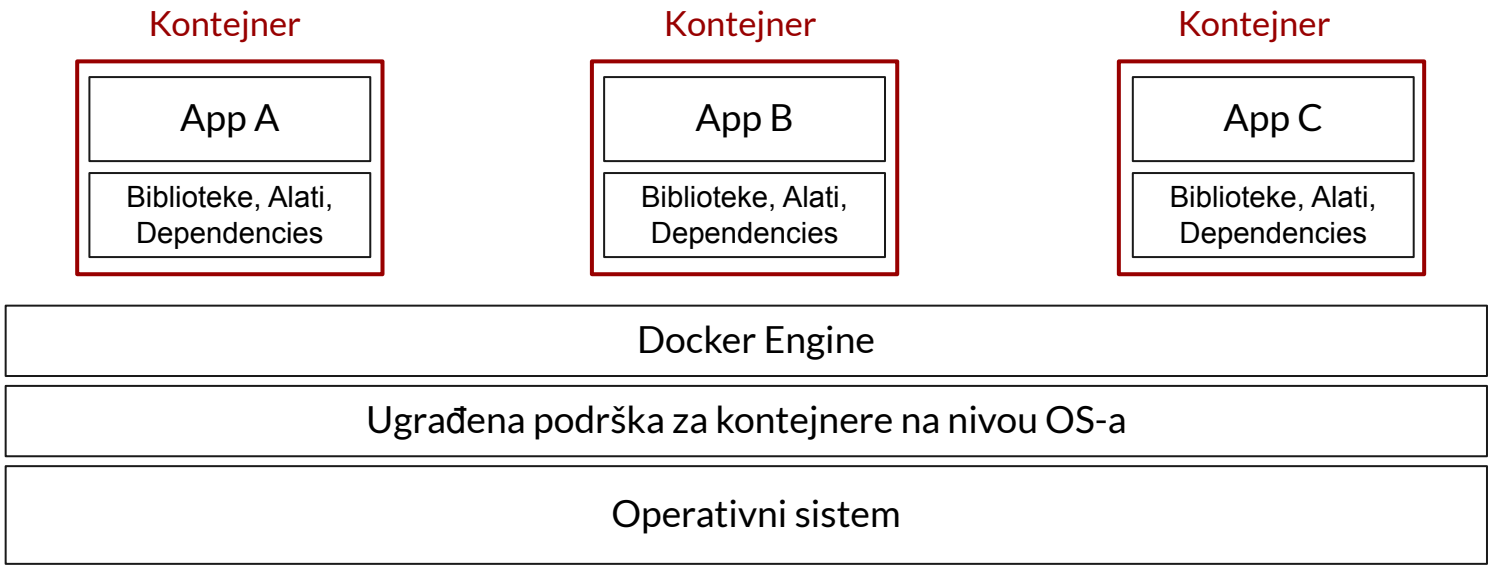
Zašto ne koristimo VM?



VM za i protiv

ZA	PROTIV
Odvojena okruženja	Dupliciranje, prevelika potrošnja memorije
Konfiguracije koje su specifične za okruženje	Performanse mogu da budu sporije
Konfiguracije okruženja mogu da se pouzdano dele i reprodukuju	Reproduciranje na drugom računaru/serveru je moguće ali je komplikovano

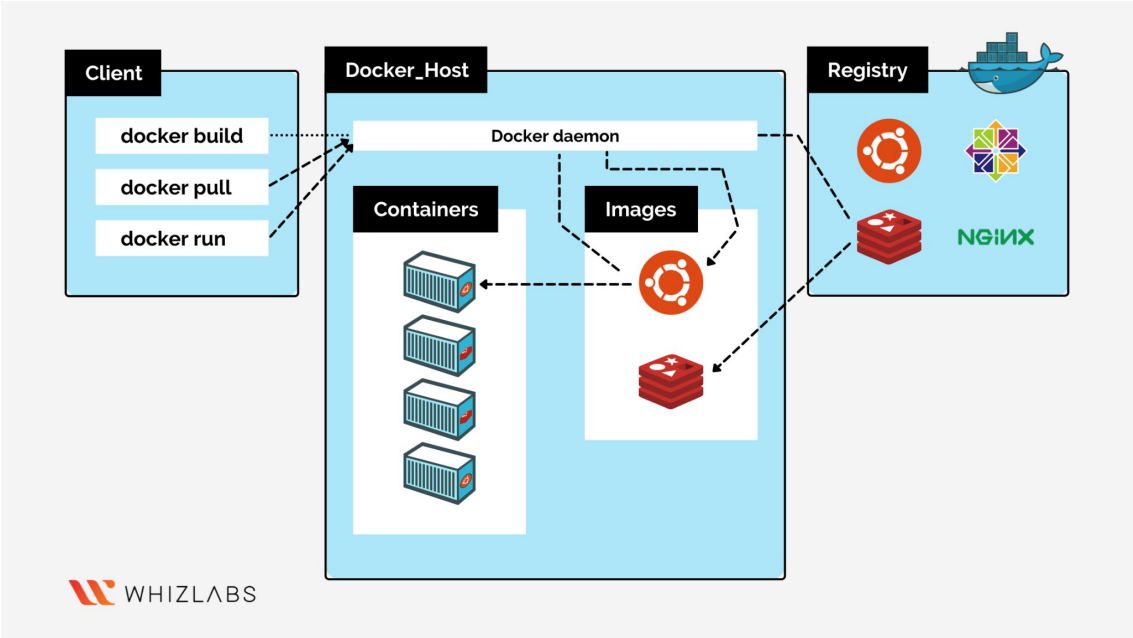
Kontejneri


Kontejner
App A
Biblioteke, Alati, Dependencies
Kontejner
App B
Biblioteke, Alati, Dependencies
Kontejner
App C
Biblioteke, Alati, Dependencies
Docker Engine
Ugrađena podrška za kontejnere na nivou OS-a
Operativni sistem

Kontejneri vs. VM

KONTEJNERI	VM
Mali uticaj na OS, vrlo brzi, minimalna upotreba diska	Veći uticaj na OS, sporiji, veća upotreba diska
Lako deljenje, ponovno build-ovanje i distribucija	Zahtevno deljenje, ponovno build-ovanje i distribucija
Enkapsuliranje aplikacija / okruženja umesto celih mašina	Enkapsuliranje cele mašine umesto jedne aplikacije / okruženja

Docker arhitektura



Docker arhitektura

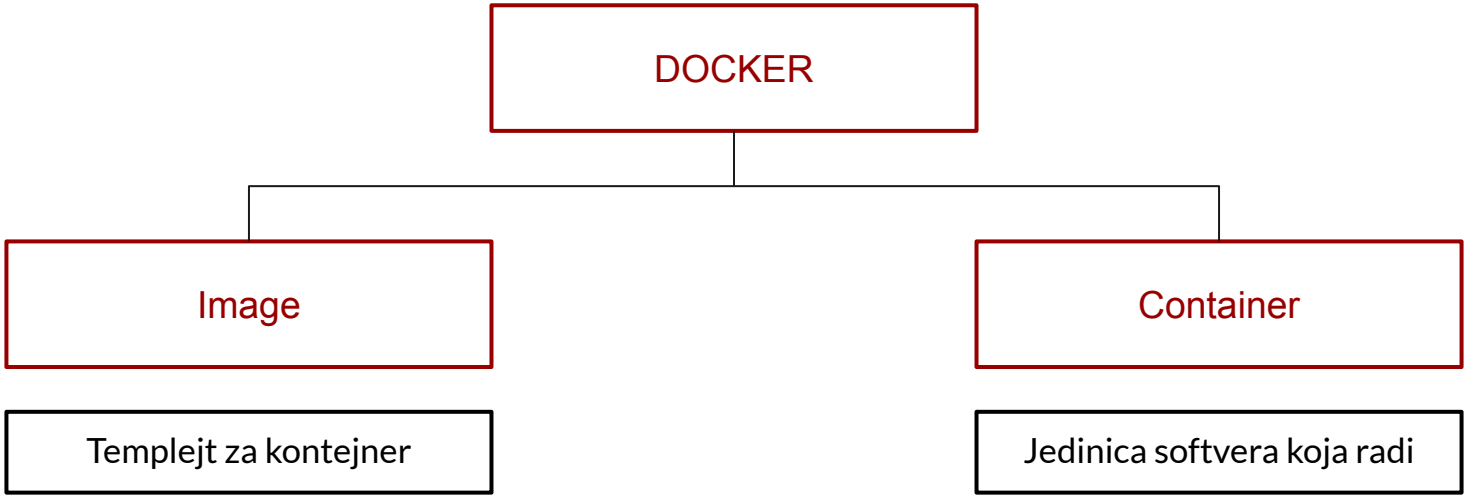
- Docker izvršno okruženje (engl. Docker Engine)
  - predstavlja aplikaciju koja prat klijent-server arhitekturu koja se sastoji od sledećih komponenti:
    - **docker pozadinski proces** (engl. Docker daemon) - pozadinski proces operativnog sistema domaćina koji sadrži sve potrebne metode za rad sa kontejnerima, slikama, mrežama i diskovima
      - predstavlja server u arhitekturi
      - pristupa mu se naredbom dockerd
    - **REST API** - koj 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

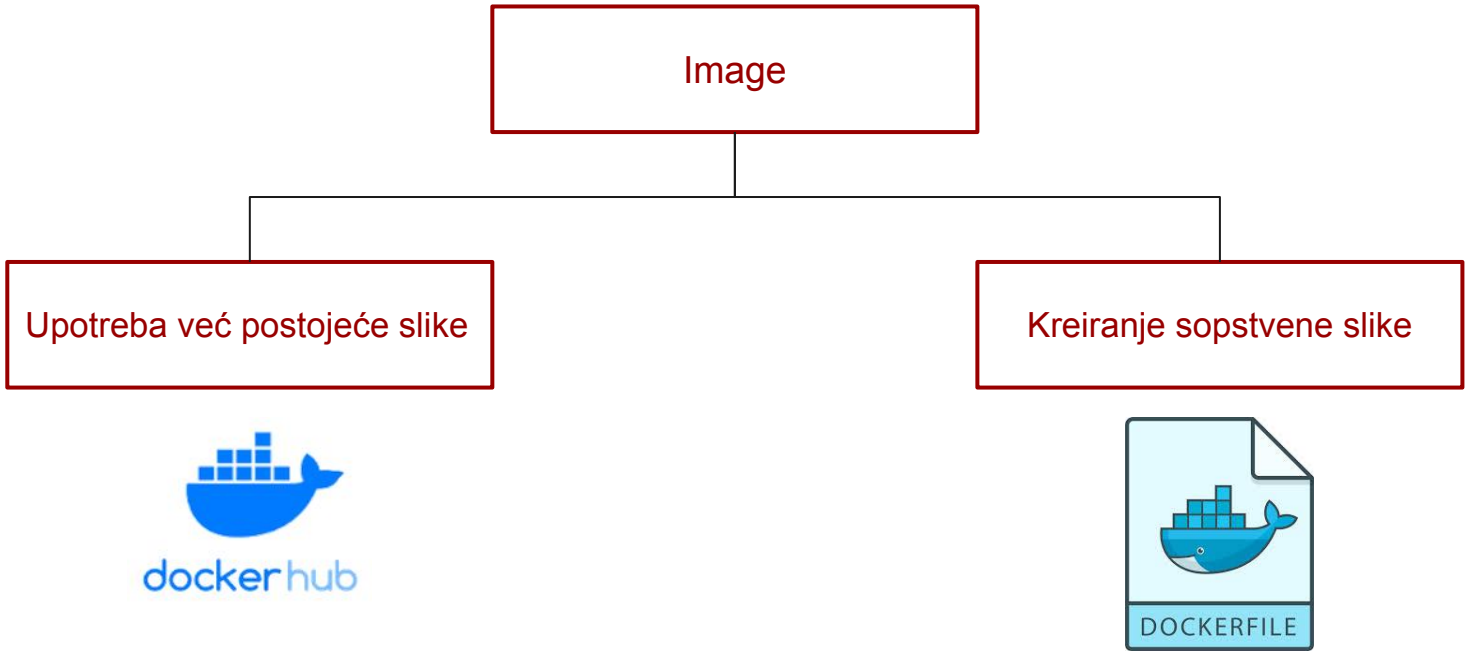
- Docker daemon
  - Sluša na Docker API zahteve i upravlja objektima poput slika, kontejnera, mreže i docker skladišta podataka
  - Može da komunicira i sa drugim Docker daemon-ima
- Docker klijent
  - Primarni način za komunikaciju sa docker-om
  - kada se koristi komanda poput docker run, klijent pošalje komandu dockerd, koji ih zatim izvršava.

Docker daemon

- 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
Image
Container
Templejt za kontejner
Jedinica softvera koja radi

Docker Image



Docker registar

- 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 sve slike
  - PRIVATNI REGISTAR
    - Docker Hub
    - Mogu se postaviti na bilo koji server dostupan preko mreže

Docker klijent [run]

```
$ 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 [pull]

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

```
$ docker pull debian

Using default tag: latest
latest: Pulling from library/debian
fdd5d7827f33: Pull complete
a3ed95caeb02: Pull complete
Digest: sha256:e7d38b3517548a1c71e41bffe9c8ae6d6d29546ce46bf62159837aad072c90aa
Status: Downloaded newer image for debian: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 [pull]

- OPTIONS = --all-tags, -a
  - Preuzima sve slike koje se nalaze u datom repozitorijumu
- CTRL + C
  - Prekidanje preuzimanja

Docker klijent [ps]

- Prikaz kontejnera (podrazumevano samo one kontejnere koji rade)
  - naredba $ docker ps [OPTIONS]
- 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]

- 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
```

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}}"
```

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 klijent [run]

- Pokretanje kontejnera na osnovu preuzete slike
  - naredba $ docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
- 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 [build]

- Kreiranje slike na osnovu Dockerfile-a
  - naredba $ docker build [OPTIONS] PATH | URL | -

Zadatak 1 - nodejs primer

```
FROM node

WORKDIR /app

COPY . /app

RUN npm install

EXPOSE 80

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

```
$ 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 klijent [images]

- Pregled 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 image

- Slike kontejnera su READ-ONLY
  - ukoliko se izmeni kod potrebno je ponovo build-ovati sliku, jer u suprotnom izmene neće biti primenjene

Docker klijent [stop]

- Zaustavljanje jednog ili više pokrenutih kontejnera
  - naredba $ docker stop [OPTIONS] CONTAINER [CONTAINER...]
  - Navodi se CONTAINER ID ili naziv kontejnera

```
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
NAMES
68e234c3f992   8794538e13a1   "docker-entrypoint.s…"   25 minutes ago   Up 25 minutes
0.0.0.0:8001->80/tcp, :::8001->80/tcp   dreamy_mirzakhani

$ docker stop -t 2 68e234c3f992
68e234c3f992
```

Docker klijent [start]

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

```
$ docker start dreamy_mirzakhani
dreamy_mirzakhani
```

Attached VS. Detached Kontejner

- **Attached** način rada
  - Terminal čeka na output iz kontejnera (primer. console.log())
  - $ docker run naredba
  - 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 klijent [attach]

```
$ docker attach dreamy_mirzakhani
```

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 dreamy_mirzakhani
$ docker logs -f dreamy_mirzakhani
```

Zadatak 2 - python-primer

```
FROM python

WORKDIR /app

COPY . /app

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

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 opcije **-i -t** prilikom naredbe $ docker run
  - Da bi se omogućio i unos preko STDIN-a potrebno je dodati opcije **-i -a** prilikom naredbe $ docker start

Docker klijent [rm]

- Brisanje kontejnera
  - naredba $ docker rm [OPTIONS] CONTAINER [CONTAINER...]
  - Briše kontejnere čiji su prosleđeni ID-jevi ili nazivi (brisanje više kontejnera njihovim navođenjem 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)

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

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 [run]

- OPTIONS = --name
  - Definisanje naziva kontejnera
- OPTIONS = --rm
  - $ docker run **--rm** [IMAGE]
  - omogućava automatsko brisanje kontejnera čim se zaustavi

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 prompta 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
  - 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 [cp]

- Kopiranje fajlova/foldera između kontejnera i lokalnog fajl sistema
  - naredba $ docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH|-
  - Primer log fajlovi mogu da postanu dostupni i van kontejnera

```
$ docker cp test/. dreamy_mirzakhani:/test
$ docker cp dreamy_mirzakhani:/test test
```

Imenovanje i tagovanje slika i kontejnera

- Imenovanje slika kontejnera
  - Naziv slike kontejnera sastoji se iz dva dela **name : tag**
  - Naredba $ docker build **-t name:tag** PATH | URL | -
- Imenovanje kontejnera
  - Naredba $ docker run **--name [naziv]** [imageID]

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"
…
```

Zadatak 3

Dokerizujte obe aplikacije - i Python i Node aplikaciju

1. Kreirajte odgovarajuće slike za obe aplikacije .
2. Pokrenite kontejner za svaku sliku i proverite da li aplikacije rade.
3. Ponovo pokrenite kontejnere sa dodeljenim novim imenima.
4. Uklonite sve zaustavljene i pokrenute kontejnere kao i slike kontejnera.
5. Ponovo build-ujte slike, ali sada sa novo dodeljenim imenima i tagovima.
6. Ponovo pokrenite kontejnere uz pomoć novo izbildovanih slika tako da se ti kontejneri automatski obrišu kada se zaustave.

Materijali:

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