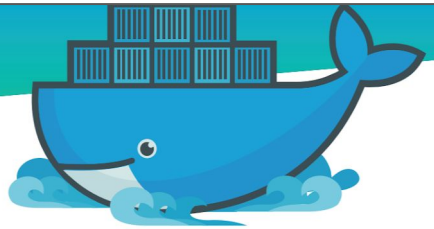
Docker

Instalacija; rezime naredbi i pojmova; zadaci

Instalacija

- Koraci za instalaciju dostupni na: https://docs.docker.com/install/
- Ukoliko se radi instalacija na nekoj distribuciji *Linux* operativnog sistema, opciono se mogu odraditi postinstalacioni koraci: https://docs.docker.com/install/linux/linux-postinstall/
  - *UNIX socket* za koji se vezuje *Docker* pozadinski proces (*daemon*) je u vlasništvu *root* korisnika, zbog čega je neophodno sve *docker* naredbe izvršavati uz komandu *sudo*.
  - Kako bi se ovo izbeglo, može se se napraviti posebna *UNIX* grupa (kolekcija korisnika koji dele datoteke i ostale sistemske resurse) pod nazivom *docker*, u koju je potom neophodno dodati korisnika
  - *docker UNIX* grupa dodeljuje privilegije ekvivalentne onim koje poseduje *root* korisnik
    - Koristiti pažljivo u produkciji - “***Only trusted users should be allowed to control your Docker daemon.”***

Docker Cheat Sheet



ORCHESTRATE

Initialize swarm mode and listen on a specific interface
```
docker swarm init --advertise-addr 10.1.0.2
```

Join an existing swarm as a manager node
```
docker swarm join --token <manager-token> 10.1.0.2:2377
```

Join an existing swarm as a worker node
```
docker swarm join --token <worker-token> 10.1.0.2:2377
```

List the nodes participating in a swarm
```
docker node ls
```

Create a service from an image exposed on a specific port and deploy 3 instances
```
docker service create --replicas 3 -p 80:80 --name web nginx
```

List the services running in a swarm
```
docker service ls
```

Scale a service
```
docker service scale web=5
```

List the tasks of a service
```
docker service ps web
```

BUILD

Build an image from the Dockerfile in the current directory and tag the image
```
docker build -t myapp:1.0 .
```

List all images that are locally stored with the Docker engine
```
docker images
```

Delete an image from the local image store
```
docker rmi alpine:3.4
```

SHIP

Pull an image from a registry
```
docker pull alpine:3.4
```

Retag a local image with a new image name and tag
```
docker tag alpine:3.4 myrepo/myalpine:3.4
```

Log in to a registry (the Docker Hub by default)
```
docker login my.registry.com:8000
```

Push an image to a registry
```
docker push myrepo/myalpine:3.4
```

RUN

```
docker run
          --rm  remove container automatically after it exits
           -it  connect the container to terminal
    --name web  name the container
     -p 5000:80  expose port 5000 externally and map to port 80
-v ~/dev:/code  create a host mapped volume inside the container
   alpine:3.4  the image from which the container is instantiated
      /bin/sh  the command to run inside the container
```

Stop a running container through SIGTERM
```
docker stop web
```

Stop a running container through SIGKILL
```
docker kill web
```

Create an overlay network and specify a subnet
```
docker network create --subnet 10.1.0.0/24 --gateway 10.1.0.1 -d overlay mynet
```

List the networks
```
docker network ls
```

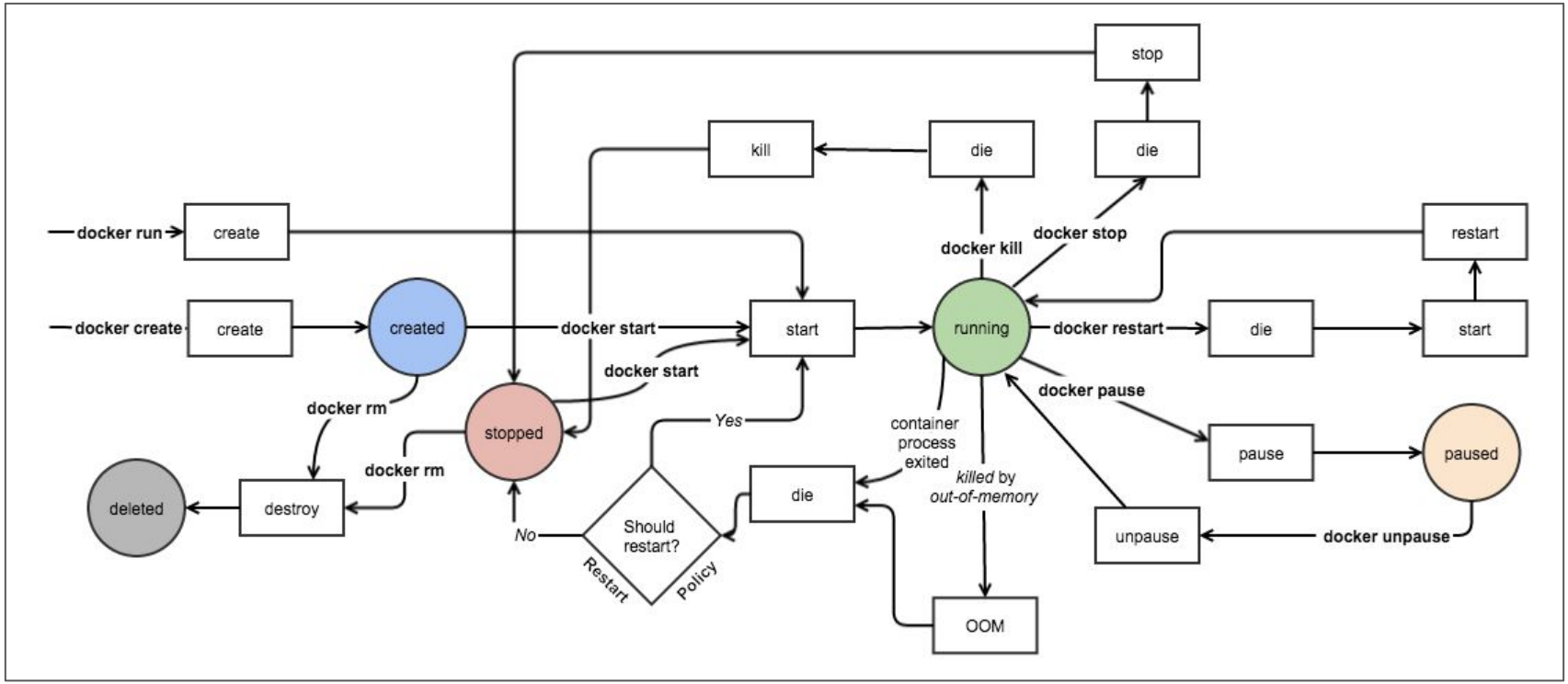
List the running containers
```
docker ps
```

Delete all running and stopped containers
```
docker rm -f $(docker ps -aq)
```

Create a new bash process inside the container and connect it to the terminal
```
docker exec -it web bash
```

Print the last 100 lines of a container's logs
```
docker logs --tail 100 web
```

Doker kontejneri - životni ciklus



Docker slike kontejnera

- Slike se u terminologiji Docker-a dele na
  - **osnovne slike** (*eng. base images*)
    - slike koje nemaju roditeljsku sliku i od kojih se prave druge slike.
  - **zvanične slike** (*eng. official images*)
    - slike softvera koje su kreirali inženjeri u Docker-u
    - obično je njihov naziv sastavljen od jedne reči.
  - **izvedene slike** (*eng. child images*)
    - slike koje predstavljaju doradu osnovnih slika
    - kreirane od strane ostalih korisnika
    - obično je njihov naziv sastavljen od imena korisnika i naziva slike
      - npr. `user/image-name`

Docker slike kontejnera - Dockerfile

- Kreiranje slike kontejnera
  - potrebno je kreirati odgovarajući *Dockerfile*
    - sadrži uputstvo za kreiranje slike
    - potrebno definisati osnovnu sliku
    - potrebno definisati sve naredbe koje prebacuju osnovnu sliku u stanje potrebno za izvršavanje željene aplikacije
    - potrebno ubaciti kod naše aplikacije
  - svaka naredba u okviru *Dockerfile*-a koja menja osnovnu sliku uvodi novi sloj slike u repozitorijum slike

Docker slike kontejnera - Dockerfile

- Izgradnju slike obavlja *docker* pozadinski proces
  - docker klijent šalje zapakovan sadržaj direktorijuma koji se ubacuje u sliku
  - ovaj direktorijum mora da poseduje Dockerfile
    - ali može imati i dodatne datoteke i direktorijume
      - dodaju se u naredbom *ADD* ili *COPY*.
  - **napomena**: Celokupan direktorijum u kojem je pokrenuta naredba se pakuje i šalje pozadinskom procesu
    - obratiti pažnju da folder sadrži samo zaista potrebne fajlove

Dockerfile - rezime naredbi

Command	Description
FROM	Sets the base image for subsequent
MAINTAINER	Sets the author field of the generated images
RUN	Execute commands in a new layer on top of the current image and commit the results
CMD	Allowed only once (if many then last one takes effect)
LABEL	Adds metadata to an image
EXPOSE	Informs container runtime that the container listens on the specified network ports at runtime
ENV	Sets an environment variable
ADD	Copy new files, directories, or remote file URLs from into the filesystem of the container
COPY	Copy new files or directories into the filesystem of the container
ENTRYPOINT	Allows you to configure a container that will run as an executable
VOLUME	Creates a mount point and marks it as holding externally mounted volumes from native host or other containers
USER	Sets the username or UID to use when running the image
WORKDIR	Sets the working directory for any RUN, CMD, ENTRYPOINT, COPY, and ADD commands

Dockerfile - primeri

- Primer Node.js web aplikacije
  - zapakovati prostu, Hello World Node.js aplikaciju kao docker sliku
  - odgovarajući *Dockerfile*:

```
FROM node:8

# Create app directory
WORKDIR /usr/src/app

# Install app dependencies
# A wildcard is used to ensure both package.json AND package-lock.json are copied where available
COPY package*.json ./
RUN npm install

# Bundle app source
COPY . .

EXPOSE 8080
CMD [ "npm", "start" ]
```

Docker slike kontejnera - Dockerfile primeri

```
FROM ubuntu

RUN apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
RUN echo "deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen" | tee -a
/etc/apt/sources.list.d/10gen.list
RUN apt-get update
RUN apt-get -y install apt-utils
RUN apt-get -y install mongodb-10gen

CMD ["/usr/bin/mongod", "--config", "/etc/mongodb.conf"]
```

Docker slike kontejnera - Dockerfile primeri

```
FROM golang:1.9.2-alpine3.6 AS build

# Install tools required for project
# Run `docker build --no-cache .` to update dependencies
RUN apk add --no-cache git
RUN go get github.com/golang/dep/cmd/dep

# List project dependencies with Gopkg.toml and Gopkg.lock
# These layers are only re-built when Gopkg files are updated
COPY Gopkg.lock Gopkg.toml /go/src/project/
WORKDIR /go/src/project/
# Install library dependencies
RUN dep ensure -vendor-only

# Copy the entire project and build it
# This layer is rebuilt when a file changes in the project directory
COPY . /go/src/project/
RUN go build -o /bin/project

# This results in a single layer image
FROM scratch
COPY --from=build /bin/project /bin/project
ENTRYPOINT ["/bin/project"]
CMD ["--help"]
```

Dockerfile - smernice

- Smernice za pravljenje Dockerfile-a
  - smanjiti broj koraka u Dockerfile-u
    - povećava performanse pravljenja i preuzimanja slika (smanjuje broj slojeva slike)
  - sortirati delove naredbe unutar višelinijskih naredbi
  - početi Dockerfile sa koracima koji imaju najmanje izgleda da će biti promenjeni
    - instalirati prvo alate potrebne da aplikacija radi
    - instalirati sve potrebne biblioteke i pakete
    - kompajlirati aplikaciju
  - koristiti *.dockerignore* fajl kako bi se izbeglo nepotrebno kopiranje
  - kreirati slike tako da budu lako pokrenute i kontejneri lako zaustavljeni
    - *ephemeral* slike tj. slike koje ne čuvaju stanje u sebi već putem skladišta podataka

```
FROM alpine:3.4

RUN apk update && apk add \
    curl \
    git \
    vim
```

Napomena za rad u učionici

- Posto je svlacenje slika neophodnih za realizaciju primera, uz zadatke su date i odgovarajuce slike
- Slike su date u vidu .tar datoteka koje je neophodno ucitati na sledeci nacin:

```
docker load --input image_name.tar
```

Zadatak 1

Napisati jednostavnu statičku *web* stranicu i omogućiti pristup stranici pomoću *nginx web* servera u okviru *docker* kontejnera. U ovu svrhu, potrebno je iskoristiti zvaničnu *docker* sliku *nginx*. Takođe, potrebno je potrebno omogućiti pristup *web* stranici i izvan kontejnera.

Zadatak 1 - rešenje

Kako bi se zadatak realizovao, potrebno je:

1. specificirati *Dockerfile* datoteku koja će za osnovnu sliku iskoristiti zvaničnu sliku pod nazivom *nginx*,
2. prekopirati *web* stranicu u /usr/share/nginx/html direktorijum kontejnera i
3. izložiti port 80 spoljnom okruženju.

Pri pokretanju kontejnera potrebno je namapirati port 80 na port 8080.

Pokušati isto uz korišćenje *WORKDIR* naredbe.

Zadatak 2

U okviru *docker* kontejnera servirati jednostavnu dinamičku *web* aplikaciju napisanu na *python* programskom jeziku, datu u ***app.py*** datoteci. Pokretanje aplikacije zahteva da su u okviru *docker* kontejnera dostupni *python* interpreter (iskoristiti zvaničnu sliku *python:v2.7-slim*) i radni okvir *Flask* (naveden u *requirements.txt* datoteci). Sadržajem *web* stranice upravlja se podešavanjem vrednosti sistemske promenljive *PERSON*.

Zadatak 2 - rešenje

Cilj ovog zadatka može se postići realizacijom sledećih koraka:

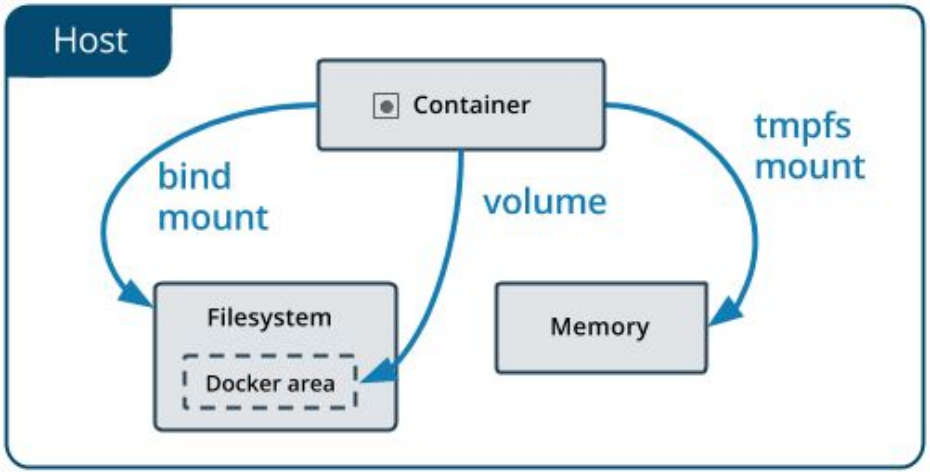
1. U novi direktorijuma prekopirati app.py i requirements.txt datoteke
2. U okviru istog foldera kreirati Dockerfile datoteku pomoću koje će se:
   a. zvanična python:2.7-slim slika postaviti kao osnova za izgradnju nove slike
   b. /app postaviti za radni direktorijum kontejnera
   c. u radni direktorijum kontejnera prekopirati sadržaj direktorijuma u kom se nalaze prekopirani fajlovi i Dockerfile
   d. pokrenuti `pip install --trusted-host pypi.python.org -r requirements.txt` naredba **u toku kreiranja nove slike**
   e. omogućiti pristup portu 80 iz spoljnog okruženja kontejnera
   f. definisati environment promenljiva pod nazivom PERSON kojoj je potrebno zadati proizvoljnu vrednost
   g. pokrenuti *python* skripta nakon pokretanja kontejnera
      i. pomoć: naredba za pokretanje: `python app.py`

Zadatak 2 - rešenje

3. Izvršiti izgradnju nove slike na osnovu kreirane Dockerfile datoteke uz tagovanje slike
4. Pokrenuti tagovanu docker sliku uz mapiranje porta 80 na port 8085

Docker skladišta podataka

- Skladišta podataka
  - Docker skladišta (*eng. volumes*)
  - skladišne tačke uvezivanja (*eng. bind mounts*)
  - privremene tačke uvezivanja (*eng. tmpfs mounts*)



izvor: *Docker Overview, Docker Volumes* https://docs.docker.com/storage/volumes/

Docker skladišta podataka

- Skladišta podataka
  - Docker skladišta (*eng. volumes*)
    - kreirane i upravljane od strane Docker-a (`/var/lib/docker/volumes/`)
    - registruju se i upravljaju na sličan način kao kontejneri i slike kontejnera
    - izolovano skladište od ostatka sistema datoteka
      - mogu ih menjati samo kontejneri kojima su skladišta dodeljena
    - mogu se deliti među kontejnerima
    - moraju se ručno osloboditi
  - stalne tačke uvezivanja (*eng. bind mounts*)
    - omogućavaju korišćenje datoteka i direktorijuma sistema domaćina unutar kontejnera
    - mogu ih menjati i kontejneri i procesi sistema domaćina
  - privremene tačke uvezivanja (*eng. tmpfs mounts*)
    - nisu stalno skladištene ni u sistemu domaćinu ni u kontejnerima
    - najčešće ih koriste kontejneri za čuvanje privremenih rezultata sračunavanja ili veoma osetljive podatke koji nestaju nakon zaustavljanja kontejnera

Docker skladišta podataka

- Kada koristiti skladišta podataka
  - Docker skladišta
    - kada je potrebno podeliti datoteke između više kontejnera
    - kada postoji verovatnoća da sistem domaćin neće imati traženu strukturu direktorijuma da bi se napravile skladišne tačke uvezivanja
    - za smeštanje datoteka kontejnera na udaljene servere
    - u slučaju kada je kreiranje sigurnosne kopije i migracija podataka kontejnera potrebna
  - skladišne tačke uvezivanja
    - za deljenje konfiguracionih datoteka između sistema domaćina i kontejnera
    - kada je struktura direktorijuma u sistemu domaćinu zagarantovana
  - privremene tačke uvezivanja
    - kada god nam podaci iz kontejnera nisu potrebni između dva pokretanja

Docker skladište

- Kreiranje skladišta
  - Naredba **`docker volume create [OPTIONS] [VOLUME]`**
  - Neophodno ručno dodeliti skladište kontejneru prilikom njegovog pokretanja
    - Opcije ***-v*** | ***--volume*** ili ***--mount***
  - Ukoliko se prilikom pokretanja kontejnera pomoću specificira skladište koje ne postoji, biće kreirano novo skladište sa zadatim nazivom
  - Primer:

```
$ docker run -d --name devtest --mount source=myvol2,target=/app nginx:latest

$ docker run -d --name devtest -v myvol2:/app nginx:latest
```

- Oslobađanje skladišta - mora ručno
  - Naredba **`docker volume prune [OPTIONS]`**
  - Naredba **`docker volume rm`** ***`volume-name`***

Docker skladišče - primer (1)

```
$ docker run -it -v /data --name container1 busybox

/ # cd data
/data # touch file1.txt
/data # ls
file1.txt
/data # exit

$ docker inspect container1
...
"Mounts": [
       {
       "Type": "volume",
       "Name": "568258a7940182c5ac52f0637c60c1d1f81e9ec77f3e4694647b4879c2ff003c",
       "Source": "/var/lib/docker/volumes/568258a7940182c5ac52f0637c60c1d1f81e9ec77f3e4694647b4879c2ff003c/_data",
       "Destination": "/data",
       "Driver": "local",
       "Mode": "",
       "RW": true,
       "Propagation": ""
       }
],
...
```

Docker skladište - primer (2)

```
$ docker restart container1

$ docker attach container1
/ # ls
bin   data  dev   etc   home  proc  root  sys   tmp   usr   var
/ # cd data
/data # ls
file1.txt
/data #

$ docker rm -v container1
```

Docker mreže

- Docker mreže koriste se za povezivanje kontejnerizovanih servisa
- Unutar mreže, pojedinačni kontejneri dostupni su
  - na osnovu dodeljene IP adrese ili
  - na osnovu **naziva kontejnera** datog prilikom pokretanja
- Naredba za kreiranje docker mreze

```
docker network create naziv_mreze
```

- Naredba za povezivanje kontejnera na mrežu
  - Može se koristiti i `--net naziv_mreze` pri pokretanju `docker run` naredbe

```
docker network connect naziv_mreze naziv_kontejnera
```

Zadatak 3

Odraditi *deployment* jednostavne dinamičke *web* aplikacije koja izračunava i vrši prikaz broja poseta stranici aplikacije. Aplikacija se sastoji od dva dela: prvi deo je *web* servis napisan na *python* programskom jeziku, dat u *app.py* datoteci, a drugi deo predstavlja baza podataka *Redis* koja čuva informaciju o broju pristupa sajtu. Oba dela aplikacije potrebno je pokrenuti u okviru zasebnih *docker* kontejnera, dok je pokrenute *docker* kontejnere potrebno povezati na kreiranu *webnet* mrežu.

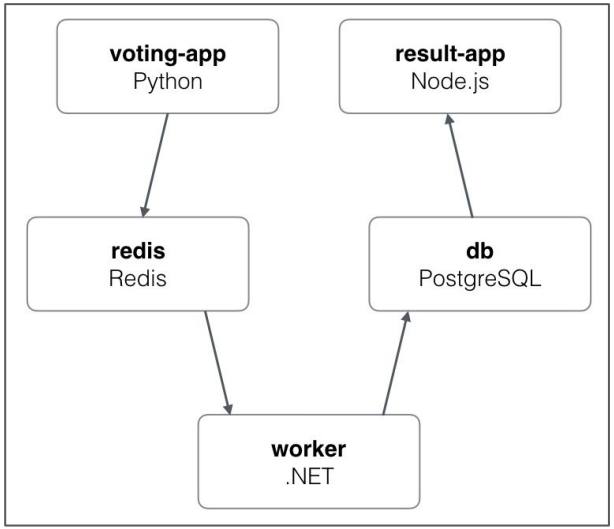
U svrhu realizacije zadatka može biti iskorištena *Dockerfile* datoteka za *python web* servis iz Zadatka 2. Za pokretanje *Redis* kontejnera iskoristiti zvaničnu osnovnu sliku, uz mapiranje porta 6379 na port 6379 i uz kreiranje skladišta podataka koje će preslikavati */data* direktorijum docker kontejnera.

Zadatak 3 - rešenje

ACS

Zadatak 4

Odraditi *deployment* aplikacije za glasanje čija je arhitektura prikazana na slici. Aplikacija se sastoji od 5 različitih mikroservisa i svaki je potrebno pokrenuti u zasebnom kontejneru. Za *voting, result* i *worker* servise dostupne su *Dockerfile* datoteke, dok je *Redis* i *PostgreSQL* servise potrebno pokrenuti na osnovu zvaničnih slika.



Napomena: kontejnerima davati imena po odgovarajućim zadebljanim rečima sa slike.

Zadatak 4

Aplikacija se sastoji od 5 mikroservisa:

- *Python web* aplikacija koja omogućava glasanje između dve ponuđene opcije (pasa i mačaka :))
- *Redis* red pomoću kog se sakupljaju novi glasovi
- .NET *worker* koji preuzima glasove i skladišti ih u bazu podataka
- *Postgres* baza podataka podržana *Docker* skladištem
- *Node.js web* aplikacija koja omogućava prikaz rezultata glasanja u realnom vremenu

Zadatak 4

Arhitektura podrazumeva da postoje dve mreže (*network*) - mreža pozadinskog nivoa, koja služi za komunikaciju svih servisa (svi servisi bi trebalo da budu povezani na nju) i mreža prednjeg nivoa koja služi za povezivanje *vote* i *result* servisa aplikacije.

Za *Redis* seris je pri pokretanju potrebno namapirati port 6379 na port 6379; za *result* servis je potrebno namapirati port 5000 na port 80, kao i port 5858 na port 5858; za *vote* servis je potrebno namapirati port 5001 na port 80.

Za *PostgreSQL* servis je potrebno obezbediti skladište podataka koje se mapira na */var/lib/postgresql/data* direktorijum, kao i zadati vrednosti za POSTGRES_USER i POSTGRES_PASSWORD (postgres) *environment* varijable.

Za *vote* i *result* servis skladišta koja mapiraju */app* direktorijum na *vote*, odnosno *result* direktorijum *host* operativnog sistema.

Zadatak 4 - rešenje

ACS