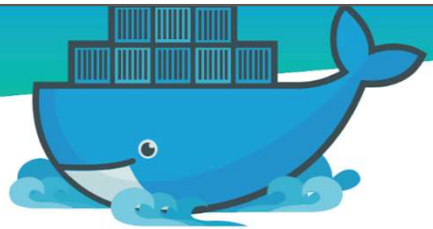
Docker

Instalacija, rezime naredbi i zadaci

Instalacija

- Koraci za instalaciju dostupni na: https://docs.docker.com/install/
- Ukoliko se radi instalacija na Ubuntu distribuciji Linux operativnog sistema, opciono se mogu odraditi postinstalacioni koraci: https://docs.docker.com/install/linux/linux-postinstall/

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
```

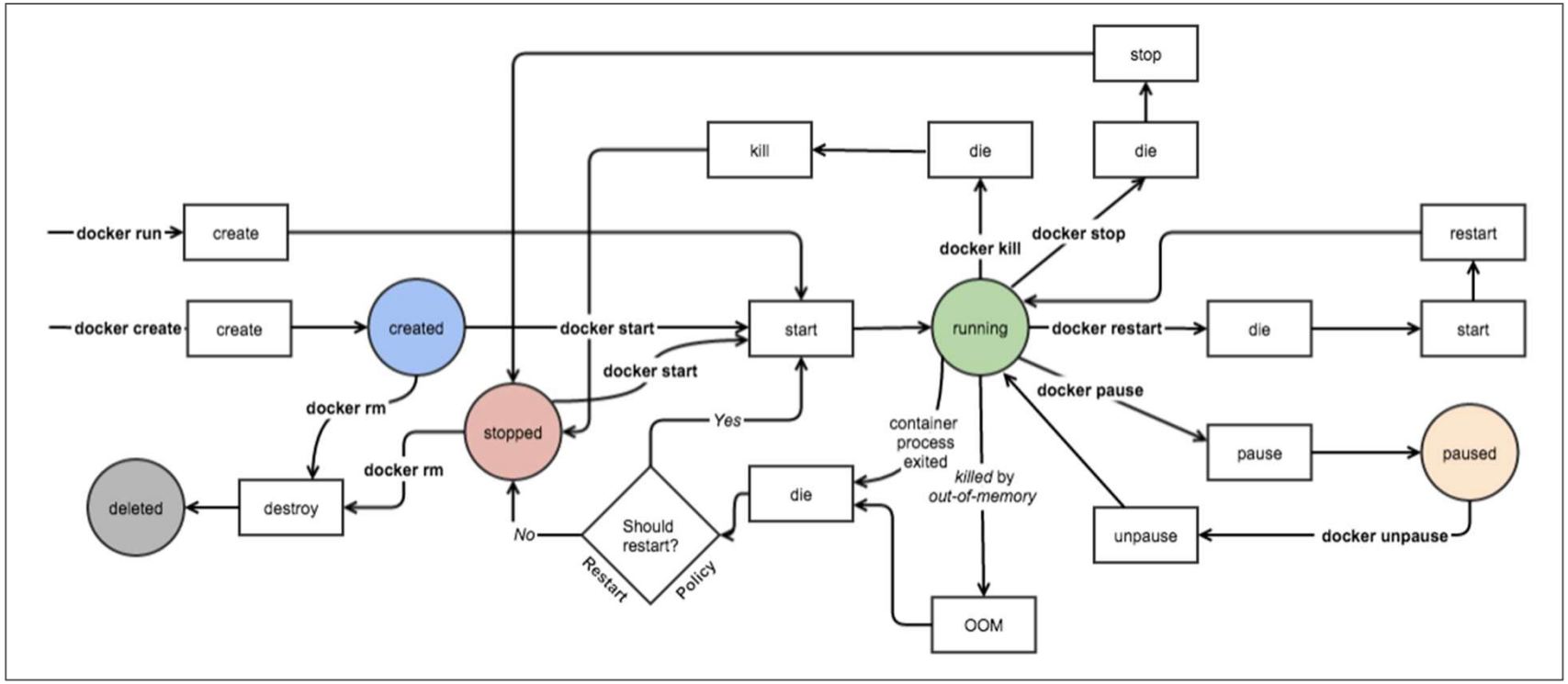
List the running containers
```
docker ps
```

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



Zadatak 1

Napisati jednostavnu statičku *web* stranicu i omogućiti pristup stranici pomoću *nginx web* servera u okviru *docker* kontejnera. 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.

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 (*v2.7-slim*) i radni okvir *Flask*. 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 /app 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

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.

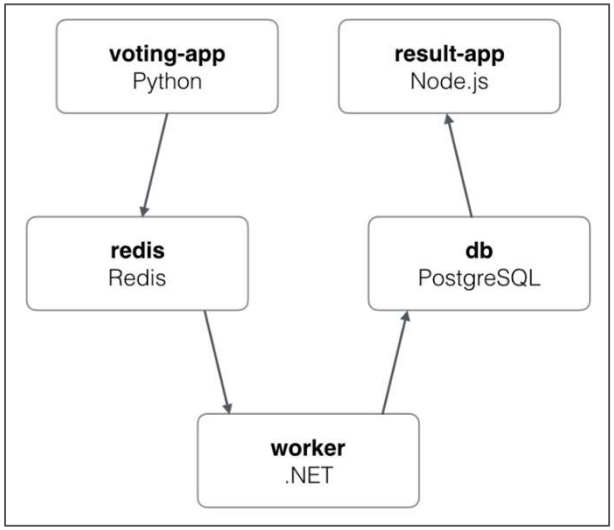
U svrhu realizacije zadatka može biti iskorištena *Dockerfile* datoteka za *python web* servis iz Zadatka 2. Za pokretanje *Redis* kontejnera iskoristiti zvaničnu osnovnu sliku, uz mapiranje porta 6379 na port 6379 i uz kreiranje skladišta podataka koje će preslikavati */data* direktorijum docker kontejnera.

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* serise su dostupne *Dockerfile* datoteke, dok je *Redis* i *PostgreSQL* servise potrebno pokrenuti na osnovu zvaničnih slika.



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

Zadatak 4 - rešenje

ACS