

Kontejneri

Docker



prof. dr. sc. Mario Kušek



[@MarioKusek](https://twitter.com/MarioKusek)



mario.kusek@fer.hr

prosinac 2019.

prezentacija i kod se nalaze na: https://github.com/MarioKusek/uvod_u_docker



■ slobodno smijete:

- **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
- **remiksirati** — prerađivati djelo



■ pod sljedećim uvjetima:

- **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
- **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
- **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.



U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela. Najbolji način da to učinite je linkom na ovu internetsku stranicu. Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava. Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licencije preuzet je s <http://creativecommons.org/>.

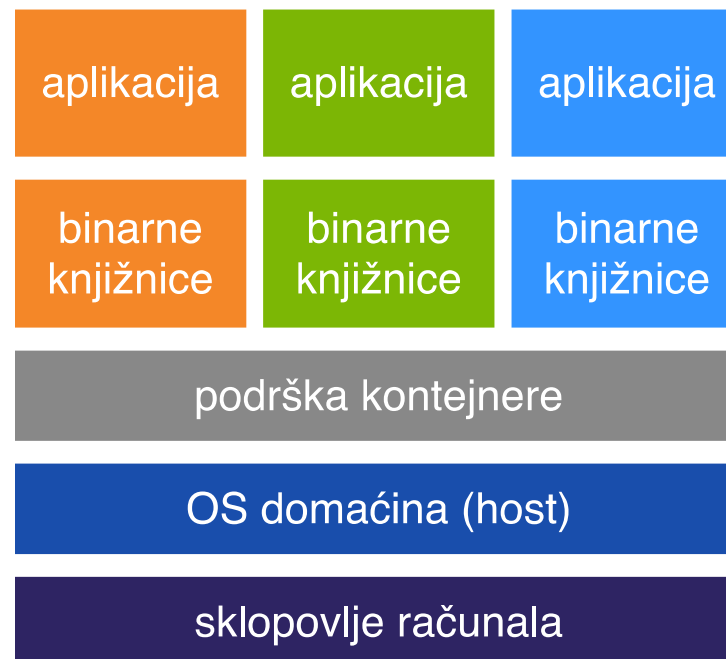
Razlika između kontejnera i virtualnog stroja



virtualni strojevi



kontejneri



♦ Prednosti:

- ne pokreće čitav operacijski sustav
- brže se pokreće
- koristi manje resursa (procesora, memorije i diska)

♦ Mane:

- slabija kontrola korištenja resursa
- slabija izolacija između kontejnera
 - potencijalni sigurnosni problem
- svaki kontejner mora koristiti isti kernel domaćinovog OS-a

- ◆ Imunes – <http://imunes.net> (UniZG – FER – ZZT)
 - prvenstveno za IP mreže
- ◆ Docker – <https://www.docker.com>
 - najpoznatiji
- ◆ runC – <https://github.com/opencontainers/runc>
- ◆ rkt – <https://github.com/coreos/rkt>
 - razvija ga [CoreOS](#)
- ◆ containerd, LXC/LXD, OpenVZ, systemd-nspawn, machinectl, qemu-kvm, lkvm, ...
- ◆ Više se može naći [ovdje](#).

- ◆ <https://www.opencontainers.org>
- ◆ Osnovana 2015. na inicijativu Dockera i ostalih
- ◆ Definira:
 - izvršnu specifikaciju
 - specifikaciju slika

- ◆ Kontejner je zapravo skup izoliranih procesa u korisničkom porostoru operacijskog sustava (Linux/Windows)

- ◆ Tehnologije za izvršavanje na Linuxu:
 - **Namespaces** - ograničava što proces vidi od okoline u kojoj se izvršava:
 - Unix Timesharing System (*hostname*), Process IDs, Mounts, Network, User IDs, ...
 - **chroot**
 - promjena korijenskog direktorija nekog procesa
 - **Cgroups** - limitira korištenje resursa
 - memorija, procesor, I/O (količina podataka prenesena), broj procesa, ...



docker

Uvod

◆ Pojmovi:

- *image* - slika OS-a
- *container* - pokrenuta slika OS-a
- *docker host* - računalo na kojem je pokrenut sustav docker
- *docker hub* - portal za razmjenu slika (registar)
 - postoje i drugi: quay.io, gcr.io, registry.redhat.io
- Dockerfile - tekstualna datoteka s konfiguracijom za izgradnju slike i za pokretanje kontejnera

◆ 3 aspekta Dockera:

■ izrada slika

- svu konfiguraciju stavljamo u datoteku (Dockerfile)
- izgrađujemo sliku na temelju konfiguracije
- sliku možemo spremiti u datoteku
- slike se sastoje od nepromjenjivih slojeva (Union file system)

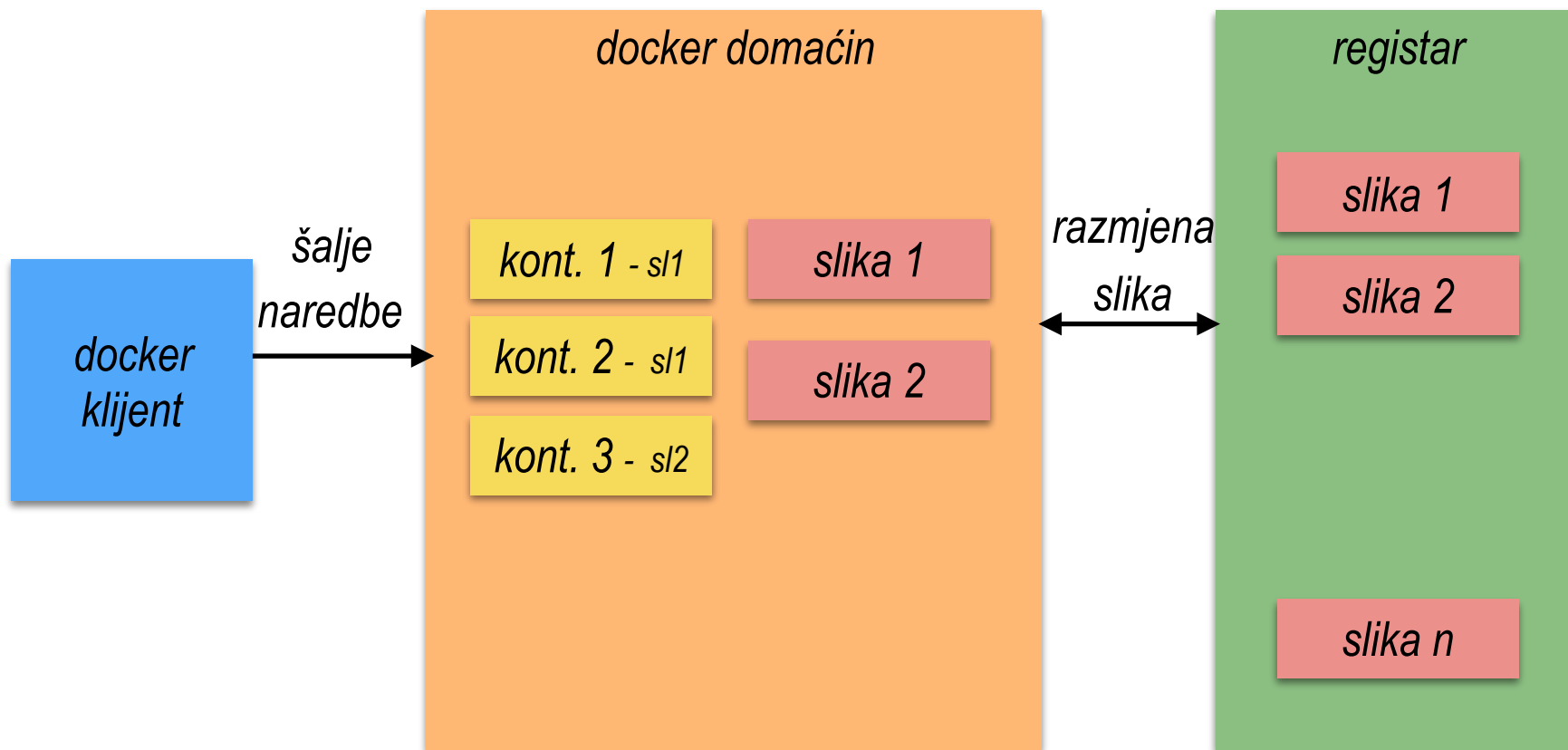
■ isporuka slika

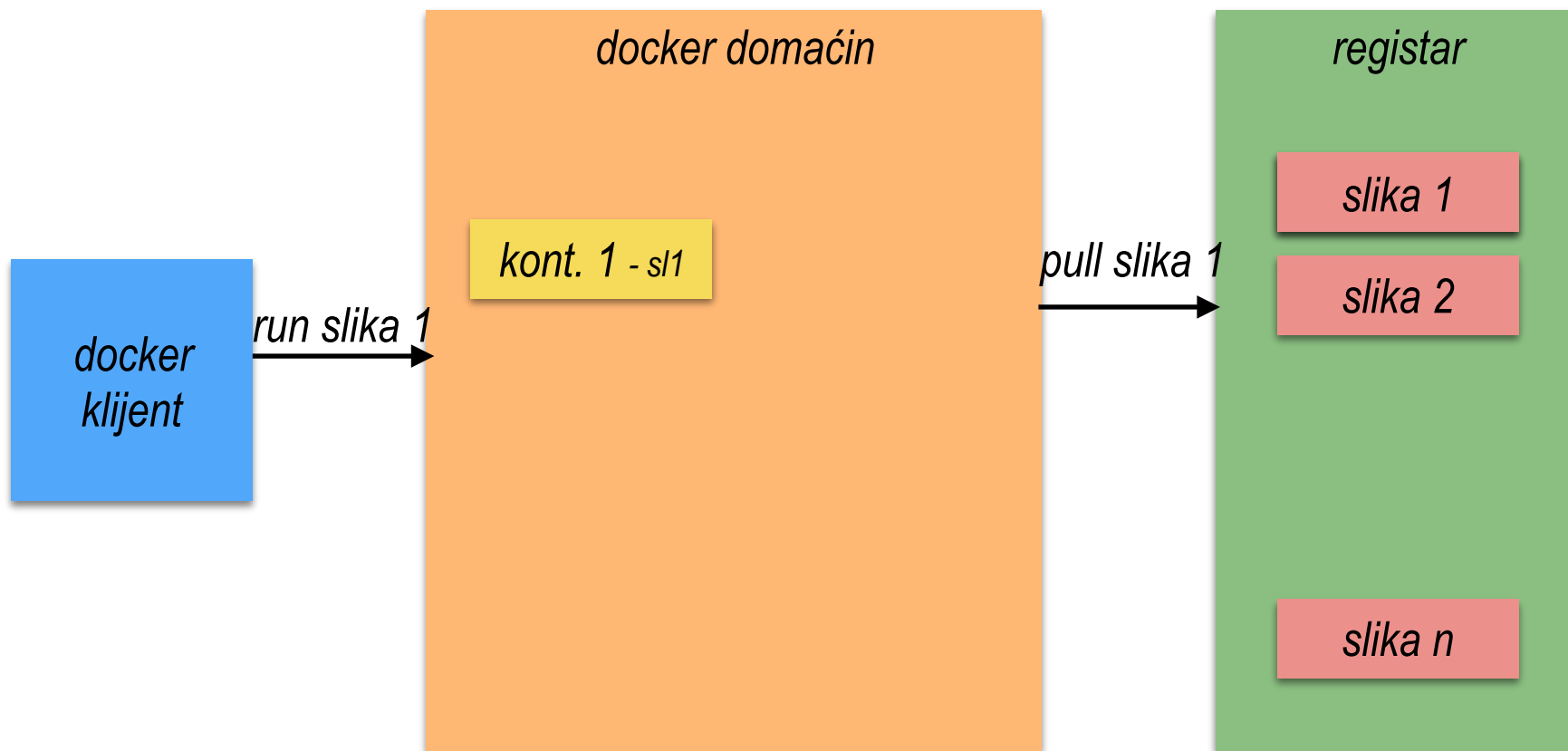
- sliku možemo podijeliti na docker hubu
- lagano možemo pretraživati i skidati slike
- imenovanje slika:
 - slike pojedinaca `<repozitorij>/<proizvod>:<tag>`
 - službene slike `<proizvod>:<tag>`

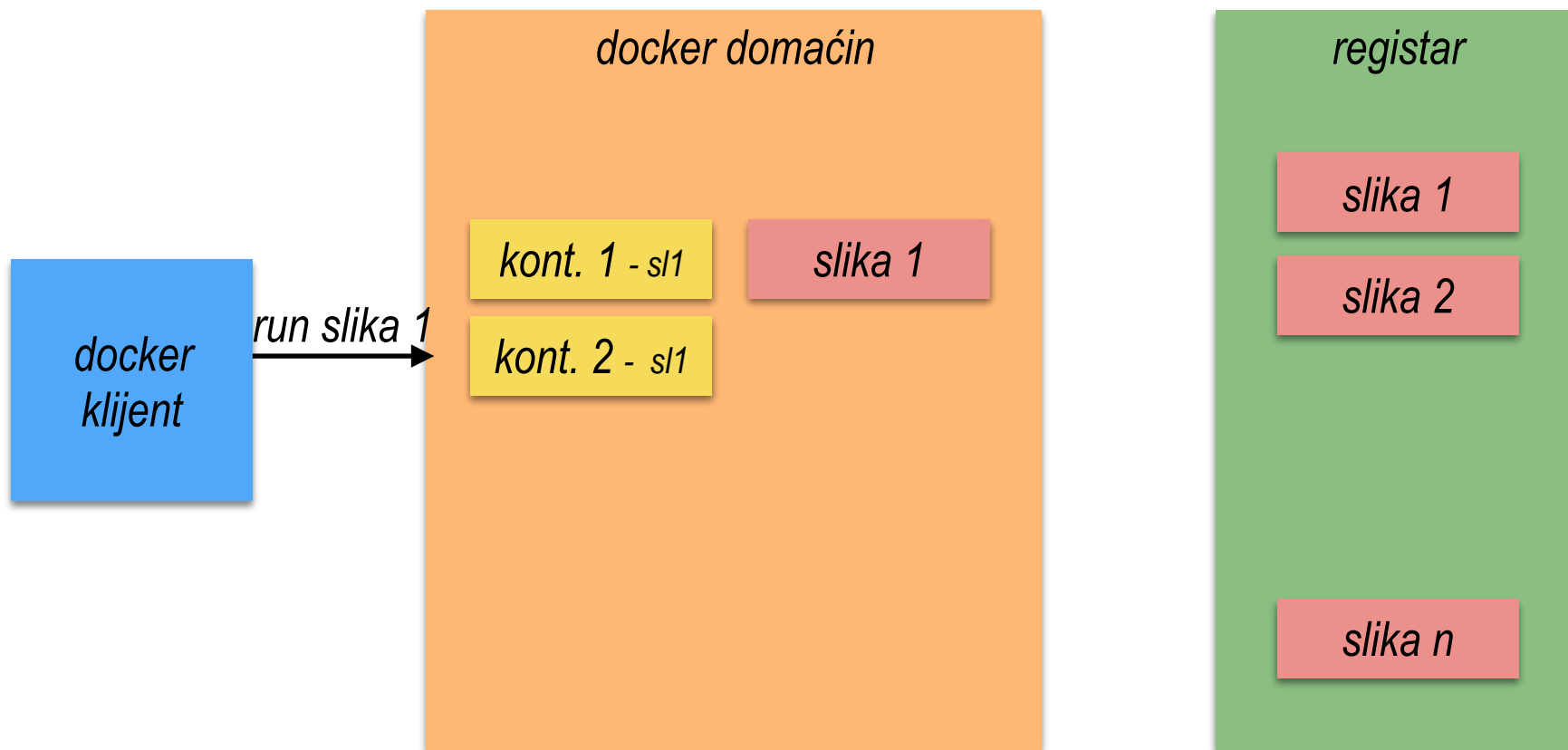
■ pokretanje slika

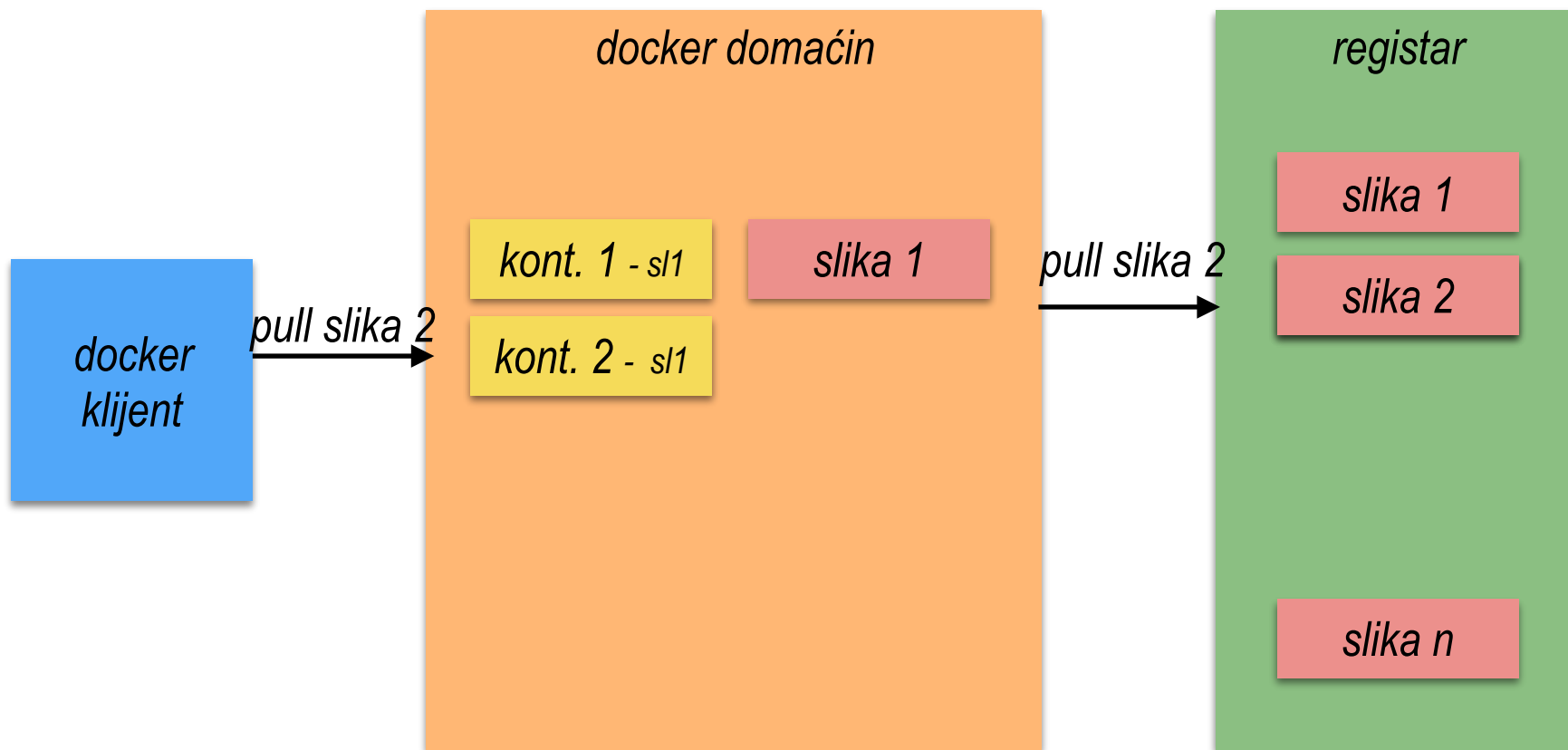
- kontejner je pokrenuta slika

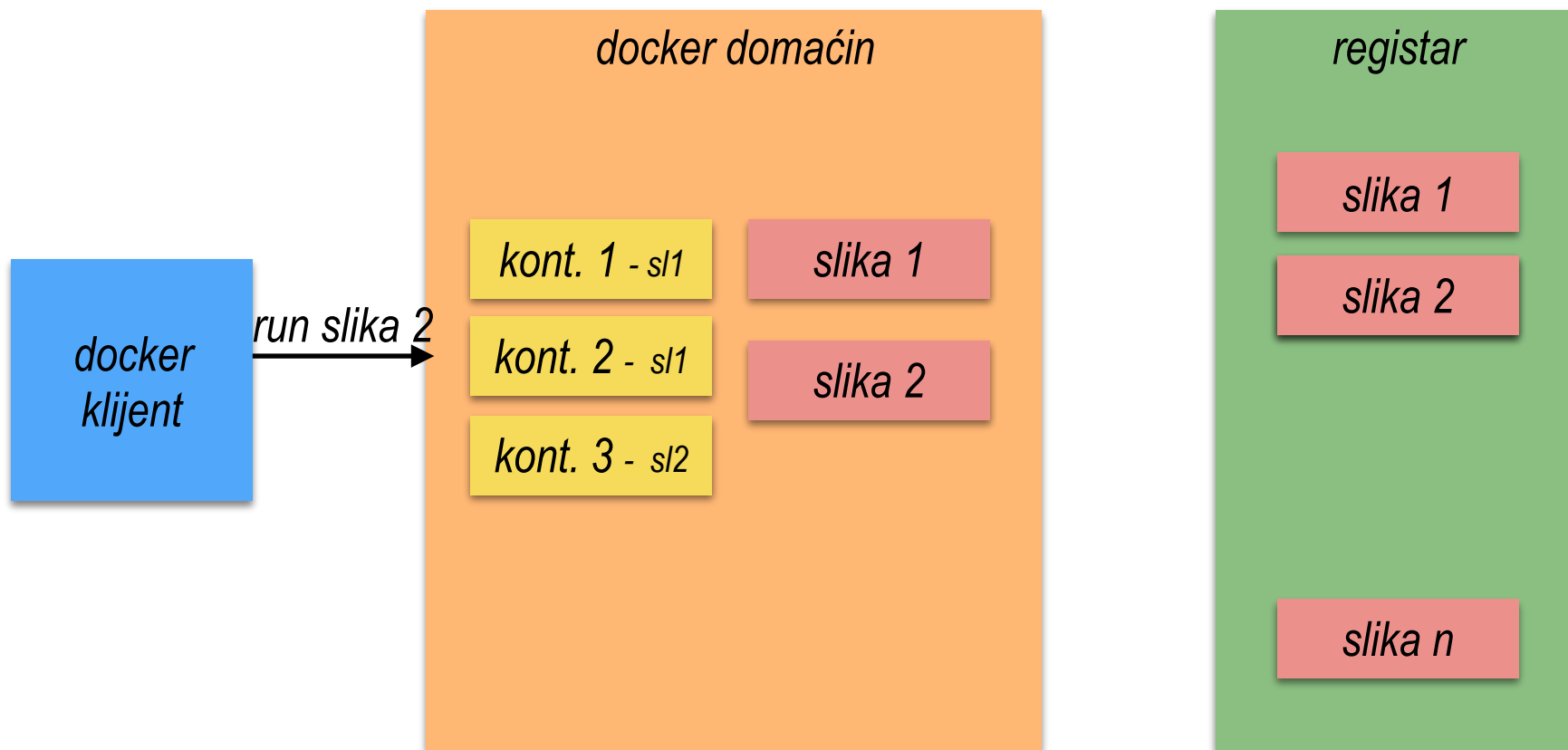
◆ sve su napravili da se može koristiti jednostavno

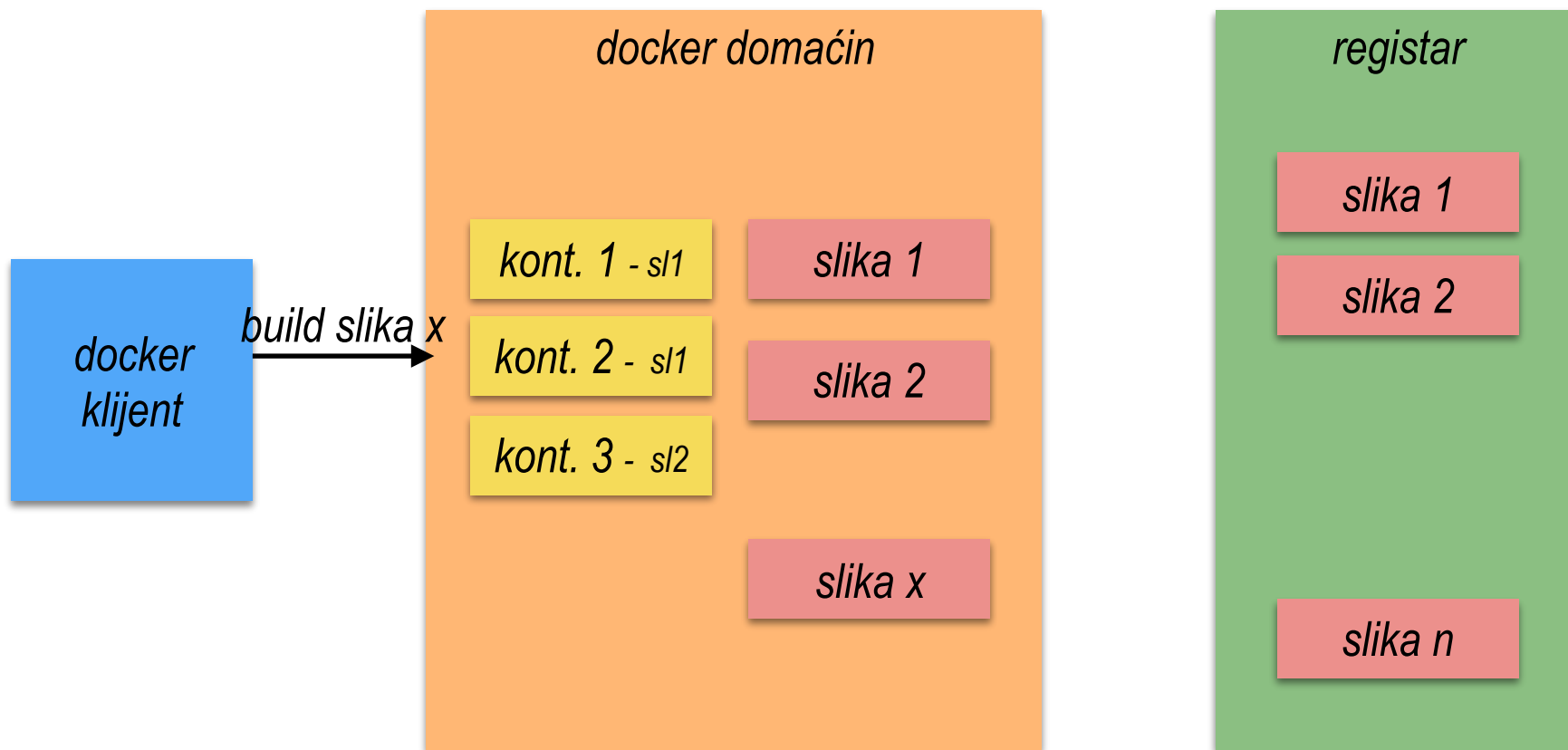


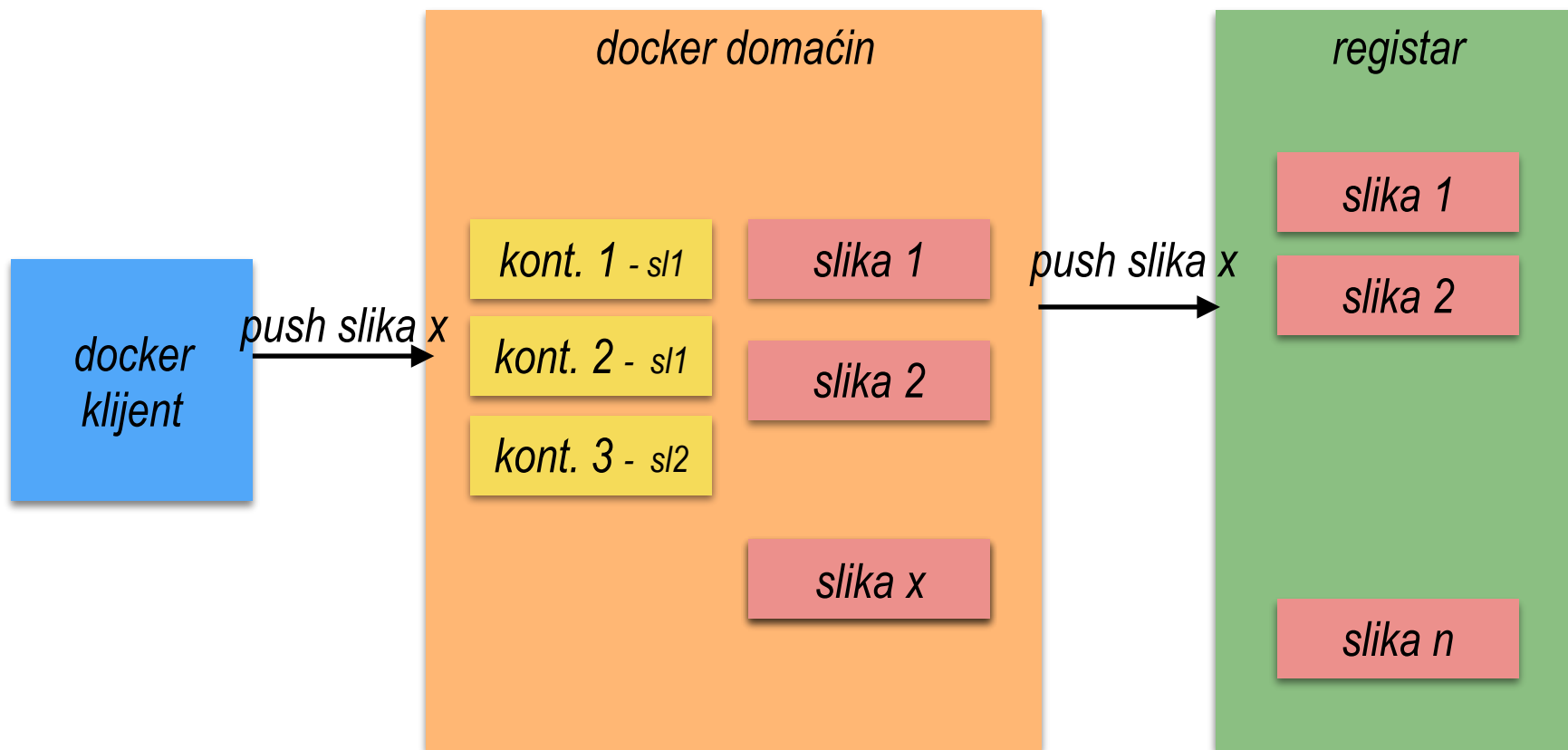












- ◆ Docker se izvršava na Linuxu
 - Od 2017. Windows aplikacije se mogu staviti u docker kontejner i pokretati na Windows strojevima

- ◆ Za izvršavanje na Windowsima ili Macu je potreban virtualni stroj s linuxom:
 - Docker Machine koji koristi VirtualBox za stvaranje stroja (boot2docker Linux distribution) - danas se rijetko koristi
 - konfigurira variable okoline da bi se klijent mogao spojiti na taj stroj
 - Docker Desktop
 - na Macu koristi native Hypervisor.framework
 - na Windowsima koristi Microsoft Hyper-V

- ◆ službene slike su u registru:

- <https://hub.docker.com>

- ◆ Docker CLI - upute za korištenje

- ◆ primjer korištenja slika:

```
$ docker pull imeslike // dohvaća iz dockerovog registra
```

```
// ovo može trajati dugo (alpine je najmanja slika linuxa 6MB)
```

```
$ docker images // lista slika
```

```
$ docker rmi <imeslike> // briše sliku
```

```
$ curl https://registry.hub.docker.com/v1/repositories/<repozitorij slika>/tags | json_pp | grep name
```

```
// dohvaćanje verzija slika
```

```
$ docker commit <ime kontejnera> <ime slike> // stvara sliku iz kontejnera
```

```
$ docker commit --change='CMD ["apachectl", "-DFOREGROUND"]' -c "EXPOSE 80" <ime kontejnera> ...
```

```
// stvara sliku iz kontejnera uz definiranje parametara (mogući parametri: CMD, ENTRYPOINT, ENV, EXPOSE, LABEL, ONBUILD, USER, VOLUME, WORKDIR)
```

```
$ docker <naredba> --help // popis svih opcija pokretanja naredbe
```

♦ primjer korištenja kontejnera:

\$ docker run -it <imeslike> // pokreće kontejner u interaktivnom modu rada

\$ docker run -d <imeslike> // pokreće kontejner u pozadinskom modu rada - za poslužitelje

\$ docker run -it <imeslike> <naredba> // pokreće kontejner u interaktivnom modu rada i u njemu naredbu
// opcije -m 150m, --cpu-quota 110000 (1000 = 1%), -p 80:8080 (host:container vrata)
// pogledati: -e var=value (env. varijable), --rm, --name ime_cont, -v src:dest volume

\$ docker exec <ime/idKontejnera> naredba // pokreće naredbu u kontejneru

\$ docker logs <ime/idKontejnera> // ispisuje log kontejnera (stdout)

\$ docker inspect -f "{{ .Config.Env }}" <ime kontejnera> // ispisuje ENV varijable u kntejneru

\$ docker stats // ispisuje sve kontejnere i koliko koriste resurse

\$ docker ps // lista pokrenutih kontejnera

\$ docker ps -a // lista svih kontejnera

\$ docker rm <ime/idKontejnera> // briše kontejner

\$ docker stop <ime/idKontejnera> // zaustavlja kontejner

\$ docker start <ime/idKontejnera> // pokreće kontejner

\$ docker container cp <lokalna dat.> <ime kontejnera>:<path u kontejneru> // kopira datoteku u kontejner

\$ docker attach <ime/idKontejnera> // spaja se (standardni I/O) na glavni proces u kontejneru

DEMO

Uvod u Docker

Docker

Izgradnja slike

- ◆ **Ručno**
- ◆ **Dockerfile**
- ◆ **Gradle**
- ◆ **Maven**
 - Spotify Maven dockerfile plugin
- ◆ **JIB**
- ◆ **Prilagođen JRE za našu aplikaciju**
 - treba koristiti `jdklink` i više razinski (*multi-stage*) Dockerfile
 - više pogledati ovdje
- ◆ **Quarkus**
 - prevođenje u izvršni kod

- ◆ ima nekoliko REST URI-ja koje možemo pozvati
 - GET /health
 - GET /stats

 - GET, POST /students
 - GET, PUT /students/{sid}
 - GET, POST /students/{sid}/courses
 - PUT /students/{sid}/courses/{cid}
 - GET, POST /courses
 - GET, PUT /courses/{cid}



0.0.1-SNAPSHOT

Docker

Izgradnja slike (ručno)

```
$ docker run openjdk:11.0.5-jre-slim java -version
```

```
$ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	...
561b6be4802d	openjdk:11.0.5-jre-slim	"java -version"	...

```
$ docker container cp build/libs/students-0.0.1-SNAPSHOT.jar  
561b6be4802d:/opt/app.jar
```

```
$ docker commit -c 'CMD ["java", "-jar", "/opt/app.jar"]' -c 'EXPOSE  
8080' 561b6be4802d students:0.0.1-SNAPSHOT
```

```
$ docker images | grep student
```

```
$ docker run -p 8080:8080 students:0.0.1-SNAPSHOT
```

```
$ curl localhost:8080/health
```

```
$ curl localhost:8080/stats
```

Docker

Izgradnja slike (Dockerfile)

◆ Dockerfile

```
FROM openjdk:11.0.5-jre-slim
ADD build/libs/students-0.0.1-SNAPSHOT.jar /opt/app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/opt/app.jar"]
```

◆ Stvaranje slike:

```
docker build -t students-df:0.0.1-SNAPSHOT .
```

```
docker run -p 8080:8080 students-df:0.0.1-SNAPSHOT
```



dockerfile

Docker

Izgradnja slike (Gradle)

Izgradnje slike pomoću plugina za Gradle (1)



- ◆ koristi se: gradle-docker-plugin

- ◆ dodati u build.gradle

```
plugins {
```

```
...
```

```
    id 'com.bmuschko.docker-remote-api' version '6.1.1'
```

```
}
```

```
import com.bmuschko.gradle.docker.tasks.image.Dockerfile
```

```
import com.bmuschko.gradle.docker.tasks.image.DockerBuildImage
```



gradle-dockerimage

Izgradnje slike pomoću plugina za Gradle (2)



```
task createDockerfile(type: Dockerfile) {
    dependsOn bootJar
    from 'openjdk:11.0.5-jre-slim'
    exposePort 8080

    addFile bootJar.outputs.files.singleFile.name, "/opt/app.jar"
    entryPoint "java", "-jar", "/opt/app.jar"

    doLast {
        ant.copy file: bootJar.outputs.files.singleFile,
            todir: "${destDir.get()}"
    }
}

task buildImage(type: DockerBuildImage) {
    dependsOn createDockerfile
    images.add("${project.name}-gradle:${version}")
}

assemble.dependsOn buildImage
```

Docker

Izgradnja slike (JIB)

Izgradnje slike pomoću plugina Google JIB



- ◆ koristi se: Gradle JIB

- ◆ dodati u build.gradle

```
plugins {  
    ...  
    id 'com.google.cloud.tools.jib' version '1.8.0'  
}  
  
jib {  
    from {  
        image = 'openjdk:11.0.5-jre-slim'  
    }  
  
    to {  
        image = "${project.name}-jib:${version}"  
    }  
  
    container {  
        appRoot = '/opt'  
        ports = ['8080']  
        jvmFlags = ['-Djava.security.egd=file:/dev/./urandom']  
    }  
}  
  
assemble.dependsOn jibDockerBuild
```



jib

Povezivanje više kontejnera (*linking*)



h2

Povezivanje više kontejnera (*linking*)



postgres

◆ Pokretanje baze u dockeru:

```
$ docker run --name postgres -e POSTGRES_USER=studentapp -e  
POSTGRES_PASSWORD=jakosiguranpass -e POSTGRES_DB=students -p  
5432:5432 -d postgres:12.1-alpine
```

◆ U application.properties postaviti sljedeću konfiguraciju:

```
spring.datasource.url= jdbc:postgresql://localhost:5432/students  
spring.datasource.username=studentapp  
spring.datasource.password=jakosiguranpass
```



postgres-docker

◆ Pokretanje baze u dockeru:

```
$ docker run --name postgres -e POSTGRES_USER=studentapp -e  
POSTGRES_PASSWORD=jakosiguranpass -e POSTGRES_DB=students -p  
5432:5432 -d postgres:12.1-alpine
```

◆ U application.properties postaviti sljedeću konfiguraciju:

```
spring.datasource.url= jdbc:postgresql://${DB_HOSTNAME:localhost}:$  
{DB_PORT:5432}/${DB_NAME:students}  
spring.datasource.username=${DB_USERNAME:studentapp}  
spring.datasource.password=${DB_PASSWORD:jakosiguranpass}
```

◆ Pokretanje aplikacije u dockeru:

```
$ docker run -d --name students -p 8080:8080 -e DB_HOSTNAME=postgres  
--link=postgres students-jib:0.2.0-SNAPSHOT
```

```
$ docker logs students -f
```



linking

- ◆ Pokretanje baze u dockeru:

```
$ docker run --name postgres -e POSTGRES_USER=studentapp -e  
POSTGRES_PASSWORD=jakosiguranpass -e POSTGRES_DB=students -p  
5432:5432 -d --volume postgres-db-volume:/var/lib/postgresql/data  
postgres:12.1-alpine
```

- ◆ ostalo na isti način pokrećemo

Docker Compose

- ◆ služi za pokretanje čitavog niza kontejnera
- ◆ postupak u 3 koraka
 1. napraviti sliku
 2. definirati usluge koje čine aplikaciju u datoteci `docker-compose.yml`
 3. pokrenuti čitavu aplikaciju pomoću `docker-compose up`



compose

◆ docker-compose.yml

```
version: "3.7"
services:
  students:
    image: students-jib:0.2.0-SNAPSHOT
    ports:
      - "8080:8080"
    depends_on: [ postgres ]
    environment:
      DB_HOSTNAME: postgres
#      DB_PORT: 5432
#      DB_NAME: students
#      DB_USERNAME: studentapp
#      DB_PASSWORD: jakosiguranpass
  postgres:
    ...
```

◆ docker-compose.yml

```
...
postgres:
  image: postgres:12.1-alpine
  volumes:
    - db-data:/var/lib/postgresql/data
#   ports:
#     - "5432:5432"
  environment:
    POSTGRES_USER: studentapp
    POSTGRES_PASSWORD: jakosiguranpass
    POSTGRES_DB: students
volumes:
  db-data:
```

◆ pokretanje

```
$ docker-compose up -d // pokretanje sustava
```

```
$ docker-compose up -d -f <file>.yaml // pokretanje sustava s datotekom
```

```
$ docker-compose ps // pregled pokrenutih stvari
```

```
$ docker-compose stop // zaustavljanje
```

```
$ docker-compose rm // brisanje kontejnera
```

```
$ docker-compose logs // pokazuje logove svih kontejnera
```

- ◆ Spring Boot with Docker
- ◆ Burr Sutter: Docker for Java Developers Hands On Lab - Devnexus 2015
- ◆ Arun Gupta: Package your Java applications using Docker and Kubernetes
- ◆ Containers form Scratch by Liz Rice
- ◆ Docker and Kubernetes Recipes for Java Developers by Arun Gupta
- ◆ Fast and Distroless Java in Containers: the Recipe! | EclipseCon Europe 2018