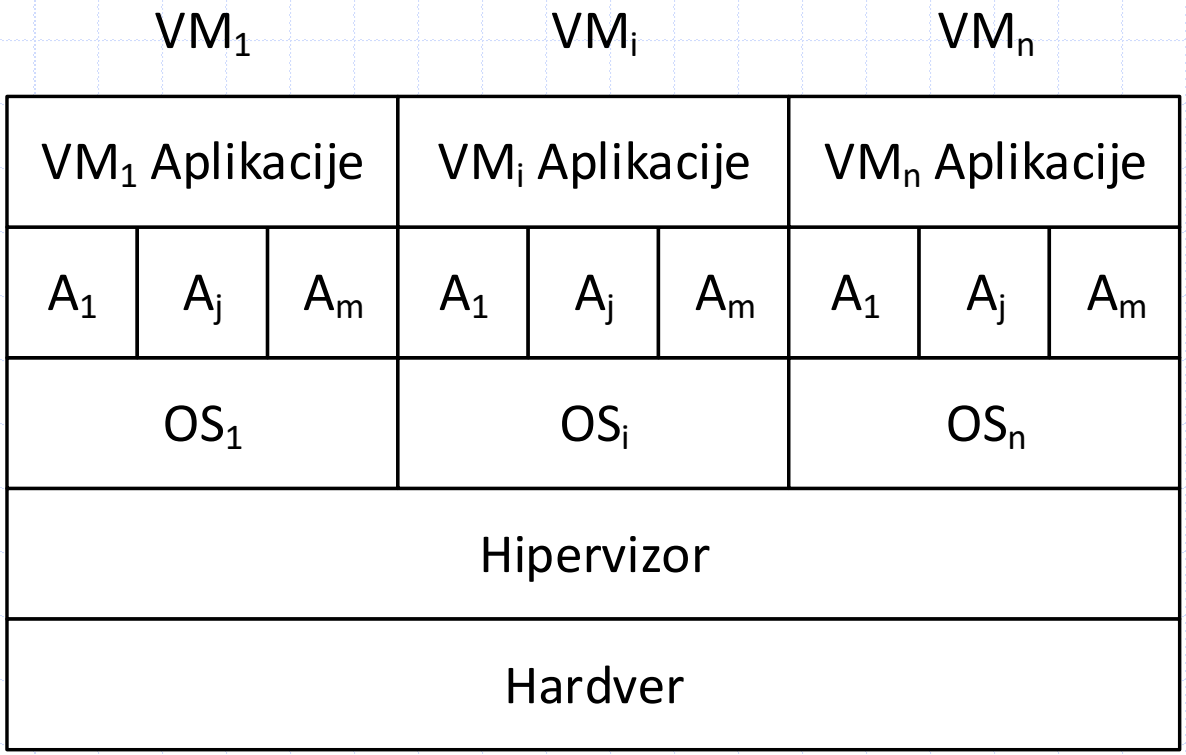
Kontejneri

- Koncept kontejnera
- Docker kontejneri

Prednosti i nedostaci virtuelnih mašina (VM) (1/3)

- Glavna prednost VM:
  - VM podržava proizvoljan OS
- VM hipervizor (ili VM monitor, VMM):
  - Emulira procesor
  - OS bez izmena može da se izvršava na VM
  - Tehnike za direktno izvršenje „punom brzinom HW“
- Jedan VM hipervizor može da pokrene više VM:
  - Sa različitim OS na različitim VM
- Korisnik koji iznajmi VM
  - Bira OS i aplikacije koje će se na njemu izvršavati

Prednosti i nedostaci virtuelnih mašina (VM) (2/3)



- VMM pokrenuo *n* VM, svaka izvršava svoj OS
  - Na svakom OS se izvršava *m* (različitih) aplikacija

Prednosti i nedostaci virtuelnih mašina (VM) (3/3)

- Nedostaci VM:
  - Pravljenje VM traje, svaki OS mora da se napuni
  - Dodatna režija na serveru
- Svaki OS na VM izaziva 2 forme režija:
  - Dva raspoređivača: prvi u VMM i drugi u OS
  - Pozadinski sistemski procesi (za podršku korisničkih app) u svakom OS; *n* skupova ovih procesa
  - Primer pozadinskog procesa: obrada mrežnih paketa
- Režija:
  - Punjenje OS + raspoređivanje apps i sis. procesa

Tradicionalne app i elastično računanje (1/2)

- VM pristup je dobar:
  - Kad VM dugo radi (npr. danima)
  - Kad je korisniku potreban izbor ciljnog OS
- Međutim, često korisnik:
  - Ima samo jednu app, ne koristi OS, i koristi uslugu elastičnog računanja koja pokreće kopije app
  - Ovde VM pristup sa punjenjem OS nije adekvatan
- OS već ima podršku za konkurentne procese
  - Ideja: Koristiti OS za pokretanje kopija app
  - Umesto VM-OS koristiti OS sa procesima (tzv. „lakim" serverima) za kopije app

Tradicionalne app i elastično računanje (2/2)

- Na žalost OS ne rešava problem u potpunosti:
  - OS ne obezbeđuje izolaciju za više-zakupaca
- Izolacija nije obezbeđena jer:
  - Procesi dele pristup mreži
  - Procesi dele sistem datoteka
  - Proces od OS može da dobije listu drugih procesa i informaciju o resursima koje koriste (špijuniranje)

Mehanizmi izolacije procesa u OS

- Virtuelna memorija:
  - Svaki proces dobija svoj izolovan adresni prostor
- Izolacija na osnovu ID korisnika:
  - Svakom procesu i datoteci se pridružuje ID vlasnika
  - Procesu čiji vlasnik je jedan korisnik je sprečen pristup objektima čiji vlasnik je drugi korisnik
- Ali izolacija na bazi ID nije skalabilna
  - Zato je OS komuna istraživala druge načine

Linux prostori-imena (1/2)

- Prostori-imena su:
  - Skup mehanizama za izolaciju raznih aspekata app
  - Kontejneri koriste 7 glavnih prostora-imena

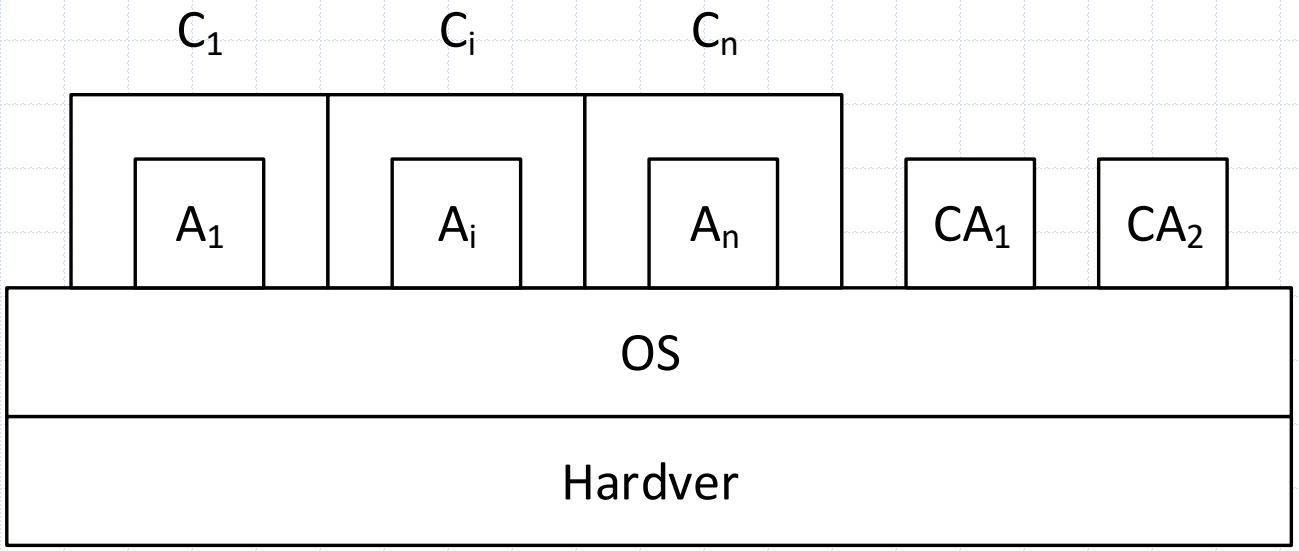
Prostor-imena	Kontroliše
Mount	Tačke montiranja sistema datoteka
Process ID	Identifikatori procesa
Network	Vidljivi mrežni interfejsi
Interprocess Comm.	Komunikacija između procesa
User ID	Korisnički i grupni identifikatori
UTS	Imena domaćina (hosts) i domena
Control group	Grupa procesa

Linux prostori-imena (2/2)

- Razni aspekti izolacija:
  - Prostor ID procesa: svaka app ima svoj skup ID-ova
  - Mrežni prostor: svaka app ima svoju IP adresu, na virt. mreži koja se razlikuje od mreže OS domaćina
  - Mount prostor: svaka app ima svoj sistem datoteka
  - Prostor grupe proc.: svaka app ima svoju grupu proc.
- Napomena:
  - Ista vrednost ID-a, npr. p, u izolovanim prostorima za različite app se preslikava u različite vrednosti

Izolacija na bazi kontejnera (1/2)

- Kontejner:
  - Izolovano okruženje za app, koje je podržano od OS



Izolacija na bazi kontejnera (2/2)

- Napomene:
  - Na OS mogu da se pored kontejnera izvršavaju i druge konvencionalne app
  - Izolacija nije savršena jer te app mogu dobiti neku info o procesima u kontejnerima
  - Zato se konvencionalne app ograničavaju samo na kontrolni SW za pravljenje i rukovanje kontejnerima

Doker kontejneri (1/2)

- Glavne prednosti doker (docker) tehnologije su:
  - Alati za brz i jednostavan razvoj kontejnera
  - Širok registar SW za korišćenje sa kontejnerima
  - Tehnike za brzu instancijaciju izolovanih app
  - Ponovljivo izvršenje na raznim domaćinima
- Alati:
  - Programer kombinuje već povezane module koda u jednu sliku koja se raspoređuje na kontejner
  - Sav potreban SW, uključujući app, se kontejner-izuje (tj. smešta) u jednu sliku
  - Posebna slika mora biti napravljena za svaku app

Doker kontejneri (2/2)

- Postoji širok registar SW, tzv. „Docker Hub“:
  - Biblioteka rasporedivih app, kao npr. Web server
  - Kombinuju se kao programi iz biblioteke
- Brza instancijacija
  - Kontejner je mnogo manji od VM jer ne zahteva OS
  - Sav SW je već povezan u jednu sliku, pa se kontejer može praktično trenutno pokrenuti
- Ponovljivo izvršenje
  - Slika je neizmenljiva (immutable)
  - Slika ostaje ista na svim sistemima, pa izvršenje uvek daje iste rezultate

Doker termini i razvoji alati (1/2)

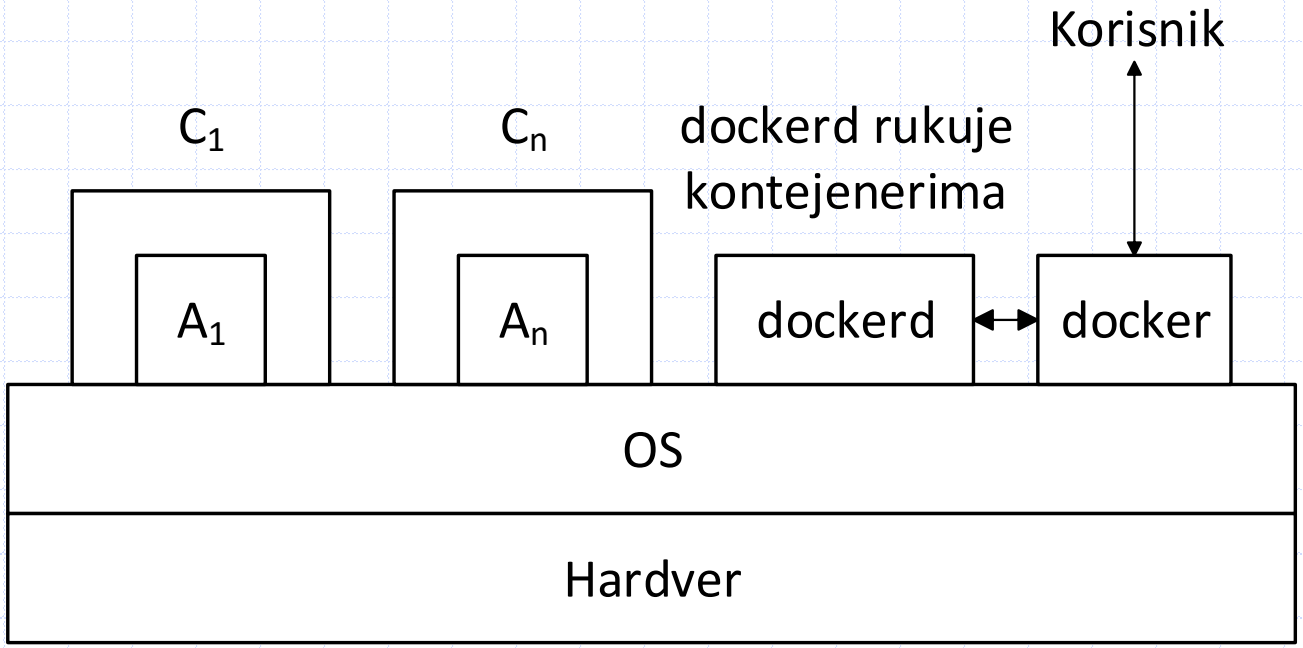
- Alati za:
  - Pravljenje i lansiranje app
  - Raspoređivanje i kontrolu kontejnera u radu
- Glavni doker termini:
  - Slika, sloj, kontejner, doker-datoteka, doker-povezivanje, doker-izvršenje
- Sledi tabela termina sa 3 kolone:
  - Izvorni termin na engleskom, objašenje, i analogija sa konvencionalnim OS

Doker termini i razvoji alati (2/2)

Termin (eng.)	Objašnjenje	Analogija
image	Datoteka sa binarnim kodom za kontejner zajedno sa njegovim zavisnostima	a.out datoteka (Linux) .exe datoteka (Windows)
layer	Jedan deo SW dodat u sliku	SW modul
container	Instanca slike	Proces u izvršenju
dockerfile	Specifikacija kako da se poveže slika	Makefile (Linux)
docker build	Komanda koja konstruiše sliku u skladu sa dockerfile	„make" program (Linux)
docker run A	Komanda za pokretanje slike A u kontejneru	Lansiranje app A

Arhitektura dokera (1/2)

- Tzv. „docker engine“ sadrži 2 komponente:
  - Dockerd (docker daemon) rukuje kontejnerima
  - Docker je korisnički interfejs



Arhitektura dokera (2/2)

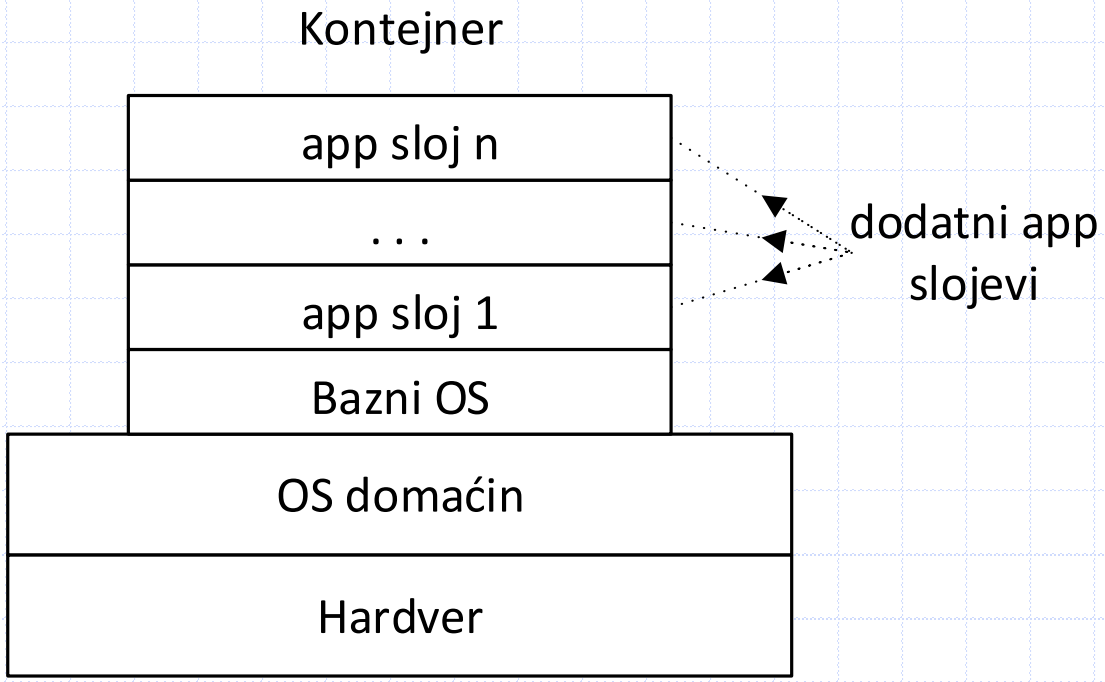
- Tri podsistema app dockerd:
  - Lansiranje i terminiranje kontejnera, pravljenje slika, preuzimanje stavki iz registra
- Dockerd podržava 2 korisničke sprege:
  - RESTful sprega preko HTTP za aplikacije
  - CLI (Command Line Interface) za ljude
- CLI komande: docker komanda argumenti
  - Komande: 15 za rukovanje + 40 dodatnih
  - Primeri u sledećoj tabeli

Primeri docker komandi

Komanda	Značenje
docker build .	Napravi sliku na osnovu docker-datoteke u tekućem direktorijumu
docker images	Izlistaj sve slike na lokalnoj mašini
docker run	Napravi i pokreni kontejner za izvršenje zadate slike
docker ps	Izlistaj sve aktivne kontejnere na sistemu
docker pull	Preuzmi kopiju slike iz registra
docker stop	Zaustavi jedan ili više kontejnera
docker start	(Ponovo)pokreni jedan ili više zaustavljenih kontejnera

Bazni OS i datoteke

- Bazni OS i datoteke:
  - Bazni OS je tanak sloj između app slojeva i OS
  - Datoteke u FS baznog OS nisu perzistentne!



Doker-datoteka (1/2)

- Specificira korake za pravljenje slike
- Glavne ključne reči doker-datoteke:
  - FROM specificira bazni OS, npr. FROM alpine
  - RUN specificira prog koji dodaje novi sloj, npr.
    RUN apk add py2-pip
    apk treba da doda Python i Pip u sliku
  - ENTRYPOINT zadaje ime start prog. sa argumentima
  - CMD zadaje komadnu koju izvršava kom. procesor (to je alternativa za ENTRYPOINT)
  - COPY (i starija ADD) dodaje direktorijume i datoteke u sistem datoteka slike (raspoloživ u slici)

Doker-datoteka (2/2)

- Glavne ključne reči doker-datoteke - nastavak:
  - COPY kopira datoteke sa računara na kom se pravi slika; npr. kopiranje Python skripte u sliku
  - ADD kopira lok ili udaljene dat, podržava kopiranje iz arhiva, ali može napraviti veću sliku
  - EXPOSE i VOLUME za povezivanje kontejnera sa spoljnim svetom
  - EXPOSE zadaje br porta, npr. EXPOSE 80
  - VOLUME zadaje tačku na koju se može montirati spoljni FS sa perzistentnim datotekama

Primer doker-datoteke

- Primer:

```
FROM alpine
ENTRYPOINT [“bin/echo”, “Hello, world!”]
```