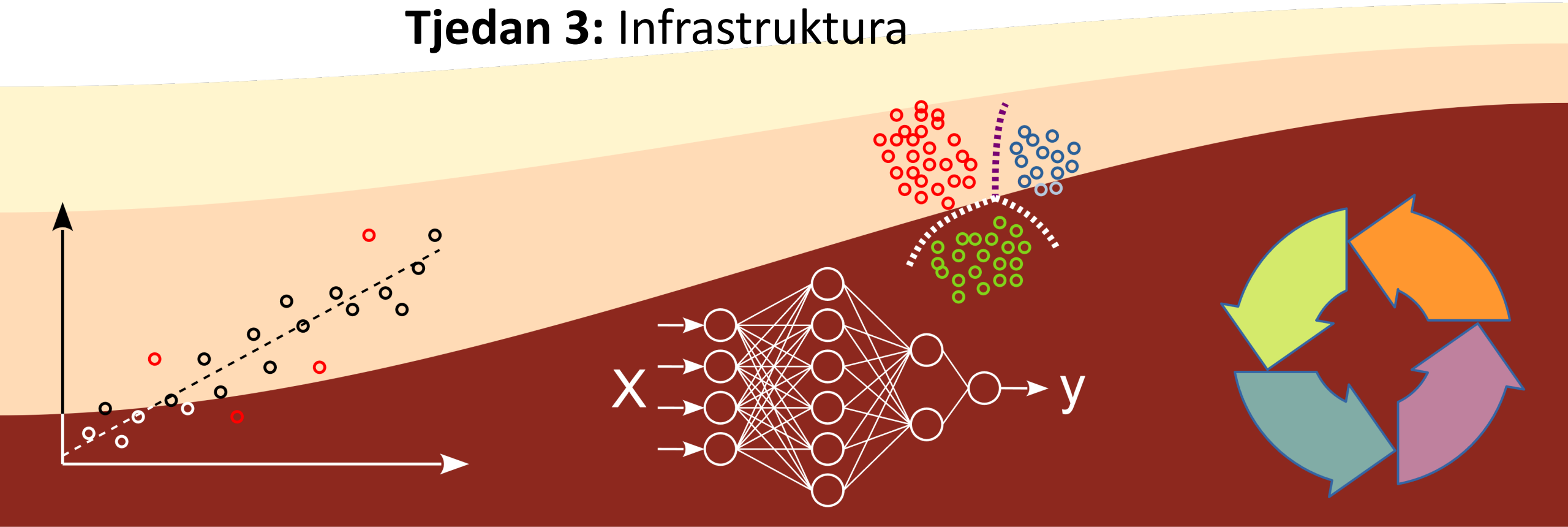
Arhitektura i Razvoj Inteligentnih Sustava

Tjedan 3: Infrastruktura

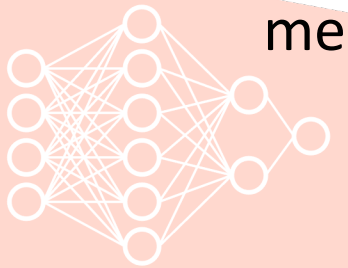


Creative Commons

- slobodno smijete:
  - dijeliti — umnožavati, distribuirati i javnosti priopćavati djelo
  - 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.

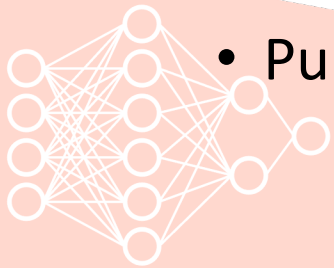
Što možemo očekivati?

- U klasičnim informacijskim sustavima imamo infrastrukturu koja se projektira kako bi zadovoljila sljedeće zahtjeve:
  - Sigurnost – autentifikacija, autorizacija, zaštita od neovlaštenog upada
  - *Fail-safe* – pad sustava koji ne uzrokuje gubitke
  - Visoka dostupnost – u slučaju problema sa serverima i strojnom opremom, sustav je i dalje dostupan
  - *Disaster recovery* – osiguranje kontinuiteta usluge u slučaju prirodne ili ljudske katastrofe održavanjem virtualne infrastrukture na udaljenoj, sigurnoj lokaciji
  - Performanse – broj konkurentnih korisnika, vrijeme odaziva
  - Skalabilnost – adaptacija sustava za povećanje resursa, kapaciteta, performansi
  - Određeni kapacitet i resursi – koliko diskovnog prostora, memorije (out of memory issues?)



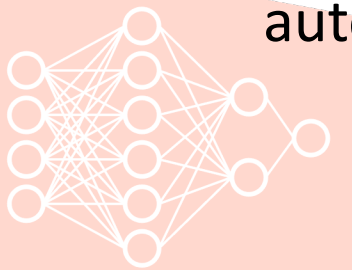
Sigurnost

- Generalno
  - Autentifikacija - ograničavanje pristupa za poznate korisnike
  - Autorizacija – ograničavanje pristupa resursima i funkcionalnostima
- Uspostava kontrole pristupa ovisi o tehnologiji informacijskog sustava
  - Klasični samostalni sustavi (aplikacije) s debelim klijentom
  - Sustavi s tankim klijentima – npr. HTML/JS, Dojo, ReactJS, ...
  - Servisne pozadine – npr. RMI/IIOP, SOAP/XML, REST/XML, REST/JSON
    - Pristup dosta ovisi da li imamo persistenciju unutarnjeg stanja
- LDAP (MS Active Directory) – baze korisnika i grupa
  - Nezaobilazni i u modernim infrastrukturama
  - Puno sustava replicira te podatke i koristi ih za konverziju u sjednice



Sigurnost

- Autorizacijski serveri
  - Oslanjaju se na standardne baze korisnika i grupa
  - Dodaju svoje autorizacijske podatke – imaju svoje baze podataka
  - Izdaju tokene
- Single-Sign On princip
  - Korisnik se usmjerava na sustav koji ga autentificira i koji mu izda token
  - Koncept tokena – upotrebljiv unutar sigurnosne domene
  - Svaki sustav unutar sigurnosne domene koji prihvaća token ne traži ponovnu autentifikaciju i autorizaciju korisnika
  - Neki sustavi povremeno provjeravaju ispravnost tokena i autorizacije sa autorizacijskim serverom

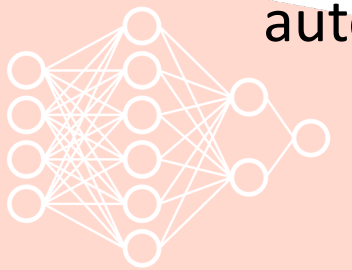


Sigurnost

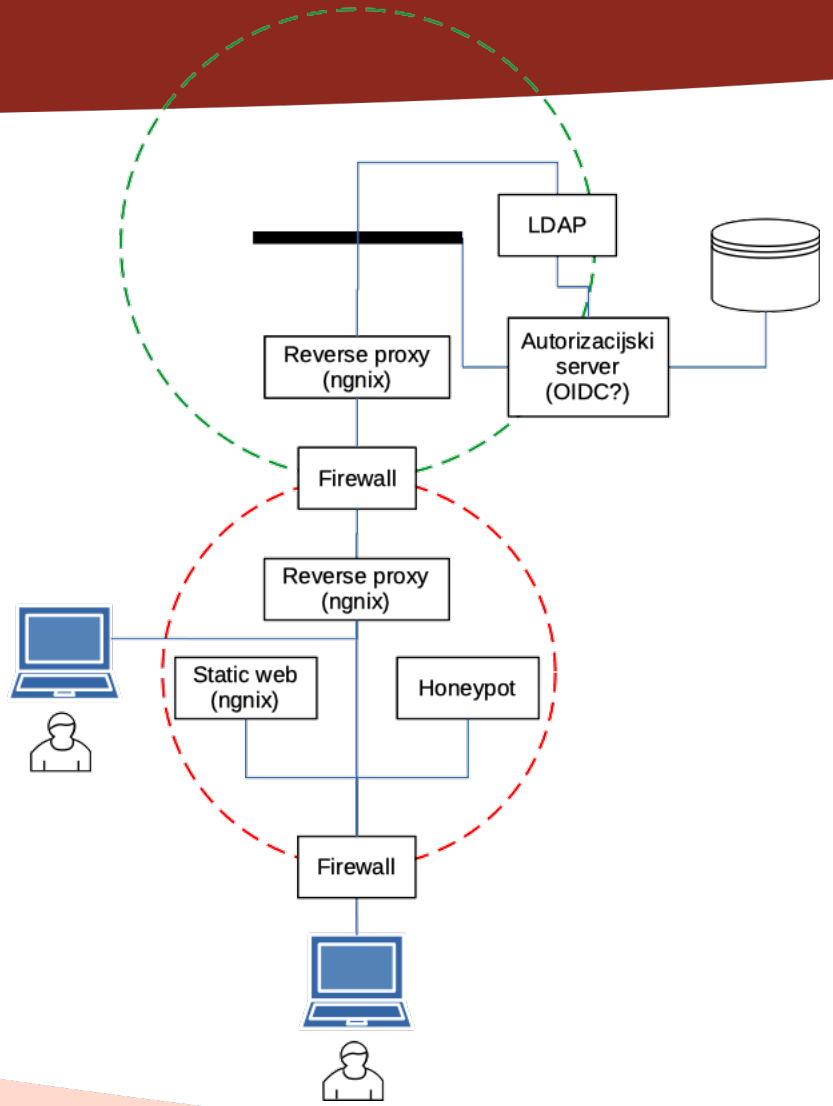
- Javno dostupni sustavi – sve ono što se može iskoristiti za napad na druge sustave – web serveri, e-mail serveri, ...
  - Problem razlikovanja javno dostupnih resursa i resursa koji moraju biti dostupni samo autoriziranim korisnicima
  - Koncept demilitarizirane zone – sustav *firewall-a* i *reverse proxy*-a
    - Honeypot
  - Praćenje DoS i DDoS napada

Sigurnost

- Autorizacijski serveri
  - Oslanjaju se na standardne baze korisnika i grupa
  - Dodaju svoje autorizacijske podatke – imaju svoje baze podataka
  - Izdaju tokene
- Single-Sign On princip
  - Korisnik se usmjerava na sustav koji ga autentificira i koji mu izda token
  - Koncept tokena – upotrebljiv unutar sigurnosne domene
  - Svaki sustav unutar sigurnosne domene koji prihvaća token ne traži ponovnu autentifikaciju i autorizaciju korisnika
  - Neki sustavi povremeno provjeravaju ispravnost tokena i autorizacije sa autorizacijskim serverom



Sigurnost

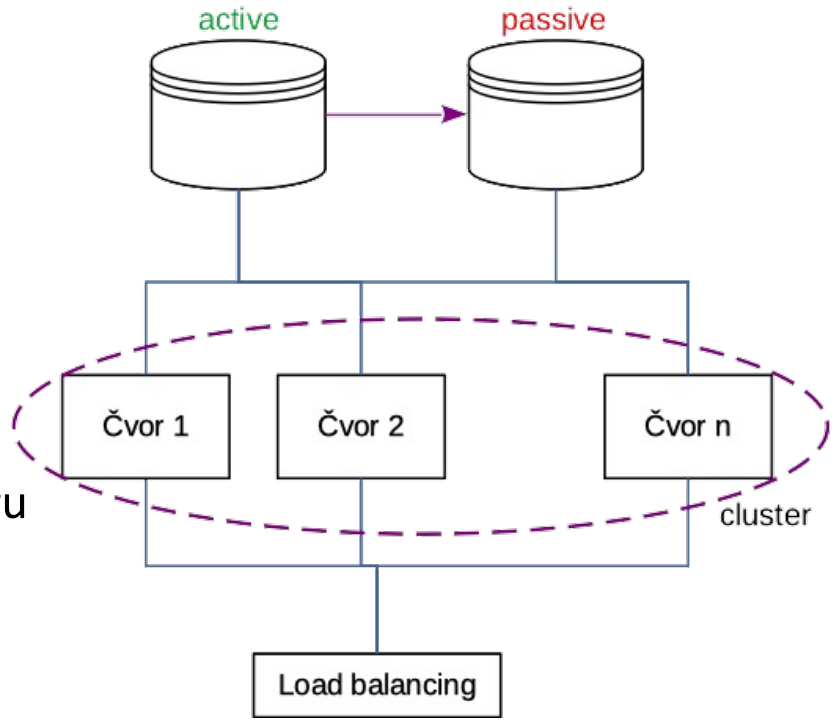

LDAP
Reverse proxy (ngnix)
Autorizacijski server (OIDC?)
Firewall
Reverse proxy (ngnix)
Static web (ngnix)
Honeypot
Firewall

Fail-safe

- U slučaju razrušenja sustava, potrebno je osigurati ponovni povratak podataka i resursa
  - Redovito arhiviranje - automatizirano
    - Baze informacijskih sustava
    - Repozitoriji source koda i ostalih resursa
    - Korisnici, grupe
  - Infrastruktura generalno ili pojedinačni sustav

Visoka dostupnost

- Strojna oprema zakazuje – čak i u slučaju dobro vođenog računskog centra (*server room*)
  - Kvarovi na serverima
  - Kvarovi na diskovima
  - Mrežna oprema
- Uspostava redundancije
  - Logičke / Virtualne
  - Fizičke
    - Dvije instance servera nikad ne držite fizički na jednom serveru
    - Dva servera nikad nisu na jednom UPS-u
    - Dva servera nikad nisu na jednoj mrežnoj opremi
    - RAID, dva fizički odvojena RAID-a
    - Kompletna dualnost (najmanje), logička i fizička
  - Pitanje sjedine i održavanja unutarnjeg stanja

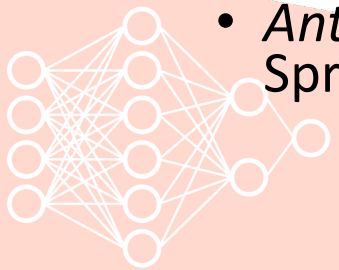


Disaster recovery

- Sekundarna lokacija
  - Skalirana infrastruktura – podržavanje osnovnih funkcionalnosti
  - Možemo tolerirati smanjenje QoS
  - Preuzima rad do dok se na uspostavi primarna lokacija
    - *No service gap* tako dugo dok se ne uspostavi mrežna redirekcija

Performanse

- Broj konkurentnih korisnika direktno utječe na ovaj problem
  - Infrastruktura nije ista za 20 paralelnih korisnika ili 5000 paralelnih korisnika
  - Responsivnost je bitan faktor – korisnici ne žele za svaki klik mišem čekati 20 sekundi
- Visoka dostupnost djelomično rješava ovaj problem
  - Masivni paralelni sustavi responsivniji
  - Koncept elastičnosti – često povezivan s *cloud* infrastrukturom
- Kvaliteta strojne opreme
- Dodijeljeni resursi
  - Disk i memorija
- Kvaliteta izvedbe informacijskog sustava
  - Programski jezici
  - Korišteni okvir (*framework*)
  - *Anti-pattern* – Hrpa slojeva apstrakcije kako bi se programerima olakšao posao (npr. Spring, eclipse)



Skalabilnost

- Povećanje infrastrukture kako bi se podržalo veće opterećenje na sustav
  - Povećanje broja korisnika
  - Povećanje količine podataka koji se obrađuju
- Sličan koncept kao i kod visoke dostupnosti
  - Više čvorova, više korisnika
- Kod obrade podataka
  - Masivno paralelno procesiranje
  - Map/Reduce koncept
- *Cloud* infrastruktura prirodno omogućava skalabilnost
  - Koncept elastičnosti
- Skalabilnost kod baza podataka?

Resursi

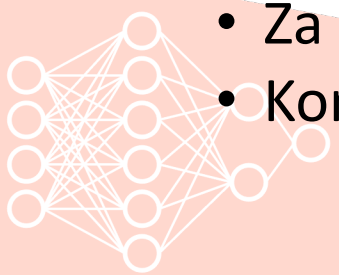
- Recimo baze podataka koje se drže u radnoj memoriji
  - Idemo na performanse
  - Ali zahtijevaju dosta radne memorije
  - Dosta BI alata i DWH koncepata funkcionira ovako
    - *In-memory data cube*
- Neka rješenja troše dosta memorije i diskovnog prostora
  - *Full-blown* aplikacijski serveri
    - Sama infrastruktura aplikacijskog servera zauzme poprilično resursa

Inteligentni sustav

- Osnovno pitanje: da li je MLOps pipeline poslovno kritičan?

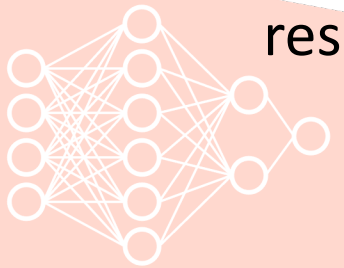
Izvori podataka

- Inteligentni sustavi se oslanjaju na vanjske izvore podataka
- Baze podataka – obično nije transakcijska baza za inf. sustav
  - Rezultat čišćenja i transformacije podataka za proces učenja
    - Sigurnost: bazi fizički pristupa ograničeni broj korisnika i sustava, te sama baza može biti izolirana od bilo kakvog javnog pristupa.
    - Performanse, visoka dostupnost i skalabilnost: Ovisno da li je MLOps pipeline poslovno kritičan i koliki je *throughput* podataka za učenje.
      - Skalabilnost baze podataka? Pitanje konzistencije. Primjer Apache Cassandra.
    - Resursi: Potreban izračun diska s obzirom na to da se mogu spremiti povijesni podaci – podaci koji su korišteni za prethodna učenja.
- Tokovi podataka
  - Fokus na performanse i skalabilnost
  - Za sustave koji generiraju masivne količine podataka
  - Kontinuirano učenje?



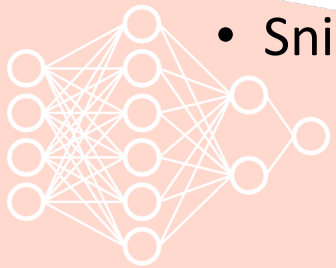
Čišćenje i transformacija podataka

- Automatski postupci za čišćenje i transformaciju podataka - ETL
- S obzirom na količinu podataka koji mogu stizati iz informacijskih sustava, postoji nekoliko točaka za razmotriti
  - Skalabilnost – Potreba skaliranja sustava radi ubrzanja procesiranja velike količine podataka
    - Apache Spark – Map/Reduce princip, *master/worker* skup čvorova
  - Performanse – Kritična stavka s obzirom na potencijalnu količinu podataka koji se obrađuju
  - Resursi – Očekuje se potrošnja memorije, kao i značajnije korištenje mrežnih resursa
    - Koncept lokalnosti resursa – npr. lokalnost koda
  - *Fail-safe* se odnosi na spremanje samih ETL postupaka – repozitorij kodova i resursa



Učenje modela

- Eksperimentiranje ili iteracija MLOps-a
  - Kod eksperimentiranja se možemo koristiti dijelom podataka (npr. KDDCup ‘99, 10%)
  - Iteracija MLOps-a često može sadržavati kombinaciju podataka iz starijih verzija i novih podataka
  - Najkritičniji zahtjev je skalabilnost – ne želimo da nam zbog količine podataka učenje traje dugo
  - Performanse – Korištenje grafičkih procesora (CUDA)
- *Large-scale Machine Learning*
  - Koncept u kojem kroz tok podataka stiže velika količina podataka – konstantno učenje
  - Pojednostavnjeni modeli
  - Stalno mijenjamo hiperparametre – kontinuirano učenje
  - Snimak modela se instalira na poslužitelje modela

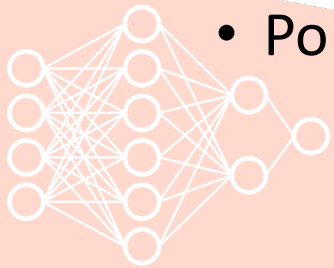


Repozitorij eksperimenata i modela

- Generalno kod repozitorija
  - Najkritičniji zahtjev je *fail-safe* – ne želimo izgubiti source kod, modele i logove eksperimenata
  - Imamo repozitorij eksperimenata (MLflow) i modela (npr. minIO)
    - Repozitorij eksperimenata je osjetljiviji
    - Repozitorij modela – kao među-spremnik za poslužitelj modela
      - Modeli se uvijek mogu izvaditi iz repozitorija eksperimenata

Poslužitelj modela (Seldon Core)

- Poslužitelj modela (posebno u produkcijskoj okolini) ima dosta generalne zahtjeve
  - Sigurnost – modelima treba ograničiti pristup – obično se pristupa iz unutarnje mreže
  - Visoka dostupnost – Želimo da su nam mikro-servisi modela dostupni bez obzira na da li je neki čvor u kvaru
  - Skalabilnost – Ovisi o broju poziva mikro-servisa modela i poslovnim procesima osnovnih informacijskih sustava, treba računati na promjenjivo opterećenje
  - Performanse - Za slučaj modela koji raspoznaju audio, video, slike, preporučljivo je imati grafičke procesore (CUDA)
  - Po pitanju resursa značajna je jedino potrošnja radne memorije



Monitoring modela (Prometheus, Grafana)

- Generalno, sustav za upravljanje vremenskim bazama podataka (TSDB) i prikupljanje metrike
  - Visoka potrošnja radne memorije zbog *time-series* baze podataka
  - Ovisi i o intervalima uzorkovanja
  - Sam monitoring nema većeg utjecaja osim na MLOps iteracije

MLOps pipeline workflow engine (Apache Airflow, ...)

- Ako je premisa da MLOps iteracije nisu poslovno kritične
  - Stoga visoka dostupnost i skalabilnost automatizacije pipeline-a nisu potrebne
  - Sama automatizacija pipeline-a je opcionalna
  - Sve ostalo se može skalirati prema dolje, pa se postupci MLOps iteracija mogu izvoditi ručno
  - U većim sustavima s više podataka je poželjno automatizirati MLOps iteracije, kao i izvesti integracija sustava
    - U tom slučaju nije loše razmisliti barem o visokoj dostupnosti MLOps pipeline workflow engine-a

Kubernetes cluster

- Enterprise infrastruktura na bilo kakvoj strojnoj opremi
  - Application management
  - PaaS – API za upravljanje deploymentom, platformom i aplikacijama
  - Jaka skalabilnost, visoka dostupnost, ...
  - Generalno osnova za cloud servisa
  - Više o tome kasnije