

# SERVERLESS

RAČUNARSTVO U OBLAKU  
FAKULTET TEHNIČKIH NAUKA  
UNIVERZITET U NOVOM SADU




# Šta je serverless?



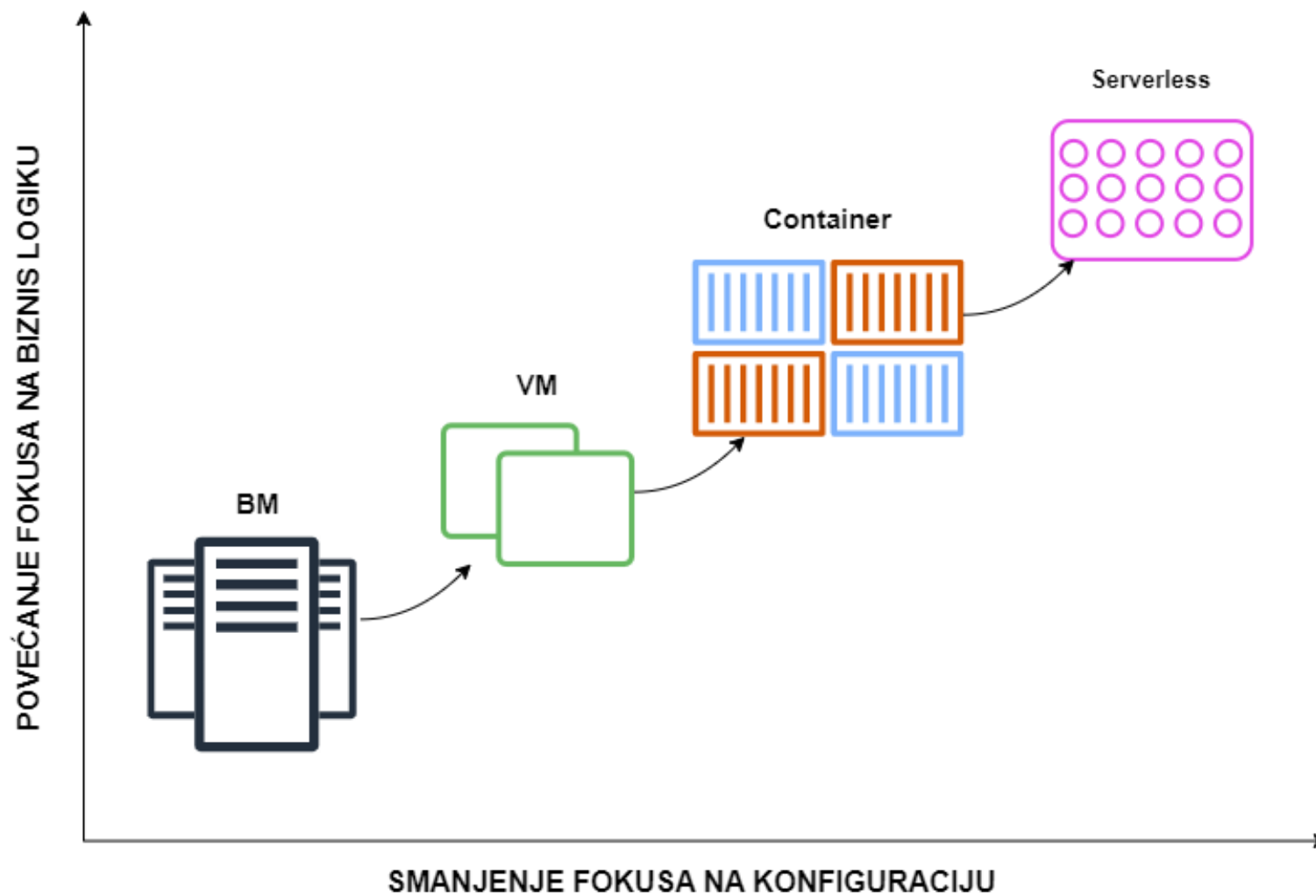
- Pogledi:

1. Arhitektura softverskih rešenja
2. Computing model gde cloud provider vodi računa o „svemu“ (infrastruktura, OS, dodatne instalacije)

- Naš pogled:


- Arhitektura softverskih rešenja koja se oslanja na computing model gde cloud provider vodi računa o „svemu“
- 

# Serverless model





# Serverless model



- **Postoje serveri**, samo njima ne rukuju korisnici
  - Cloud provider u potpunosti rukuje infrastrukturom, OS-om i potrebnim instalacijama
  - Programer samo piše kod
  - *Potpun fokus na **biznis logiku**, minimalan fokus na infrastrukturu*
  - Automatska skalabilnost
  - Visoka dostupnost
  - Pay-for-use model naplate
    - Serverska instanca uvek podignuta i spremna da prima zahteve
      - Plaća se sve vreme
    - Serverless sistem podigne compute instancu tek kada stigne zahtev
      - Plaća se samo vreme kada ima saobraćaja
- 

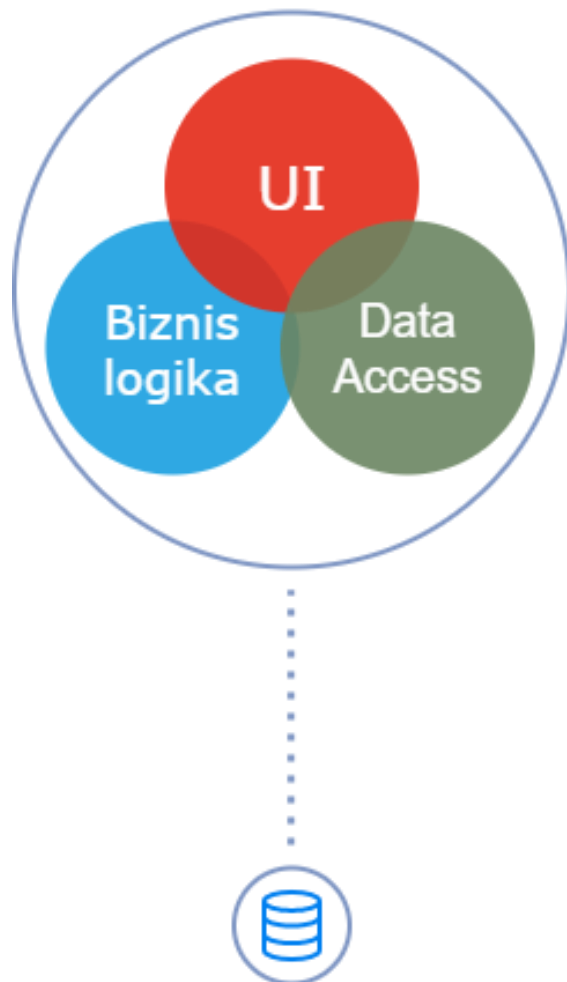
# Serverless arhitektura



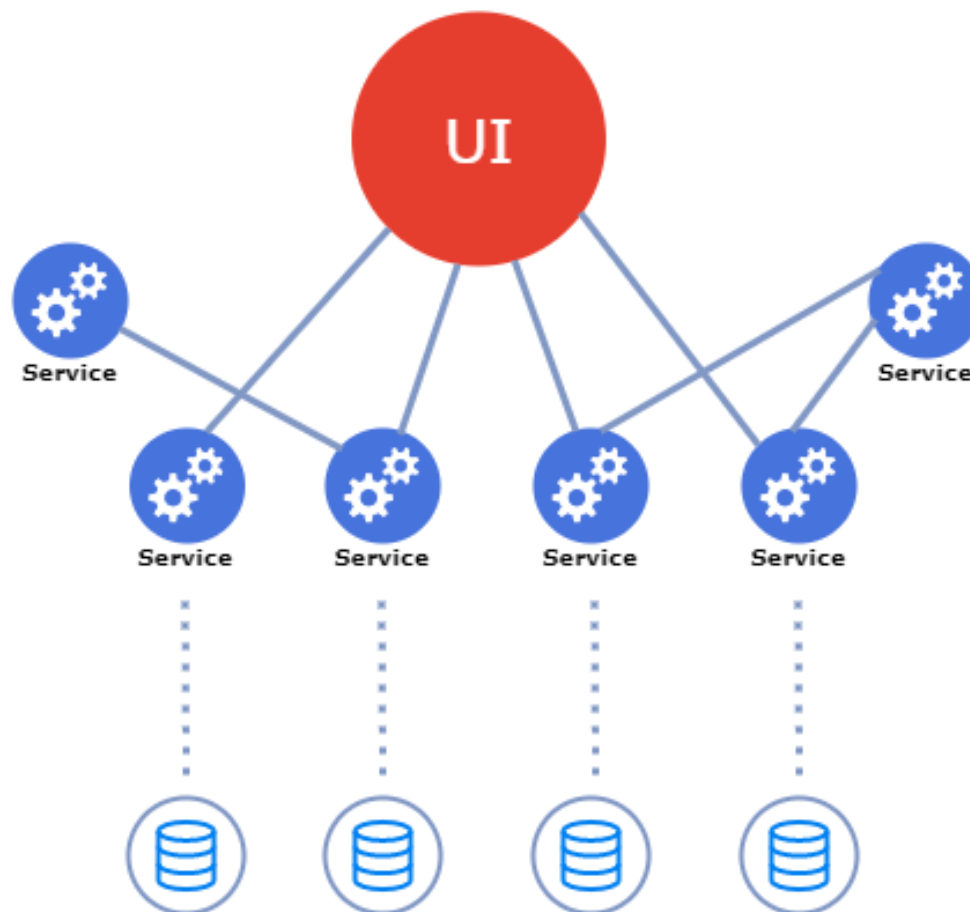
- Serverless sistem podigne compute instancu po potrebi
    - Server nije uvek aktivan
      - Problem **cold-start-a**
    - Server nije uvek dostupan
      - Problem **timeout-a**
    - Ograničeno kratkotrajno izvršavanje
      - Problem **dugačkih operacija**
  - Promena razmišljanja prilikom dizajniranja arhitekture
- 
- 

# Monolitna VS servis orijentisana arhitektura

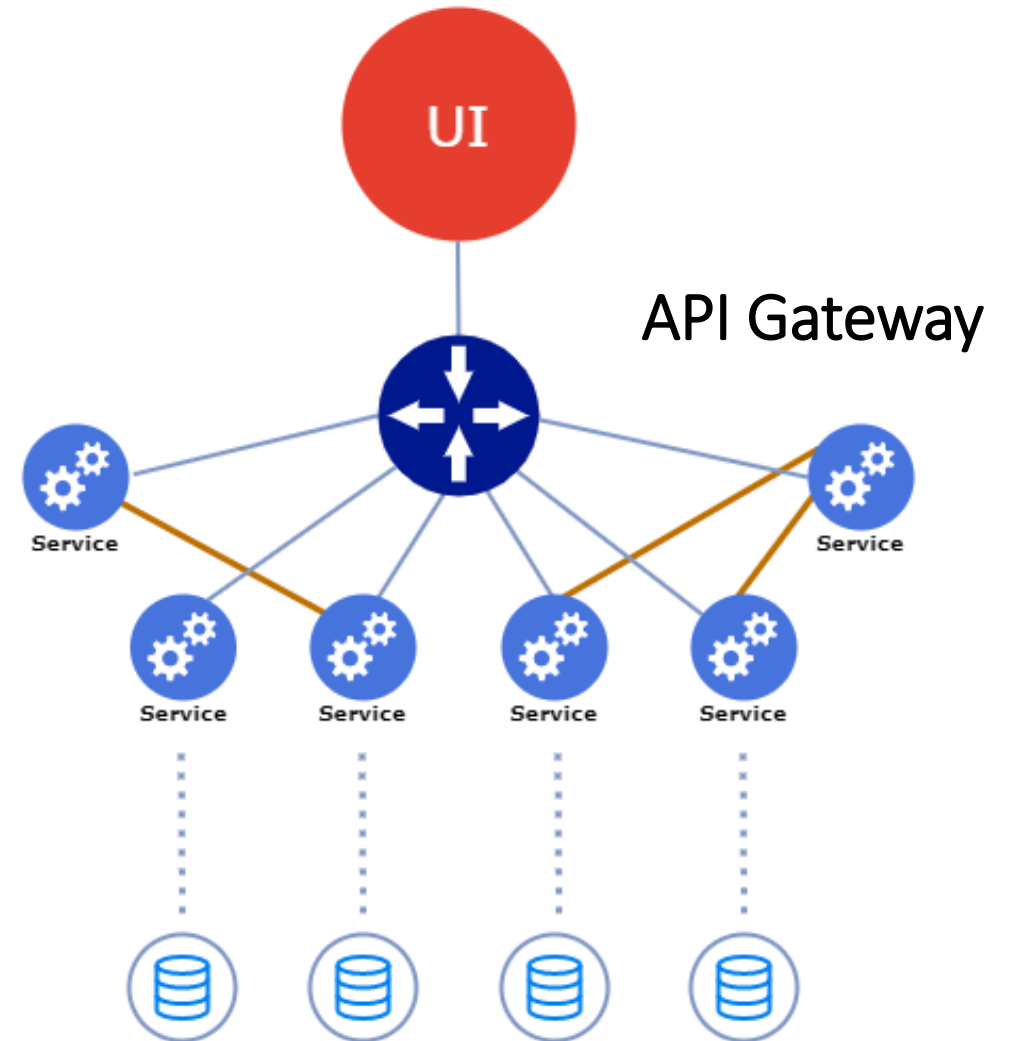
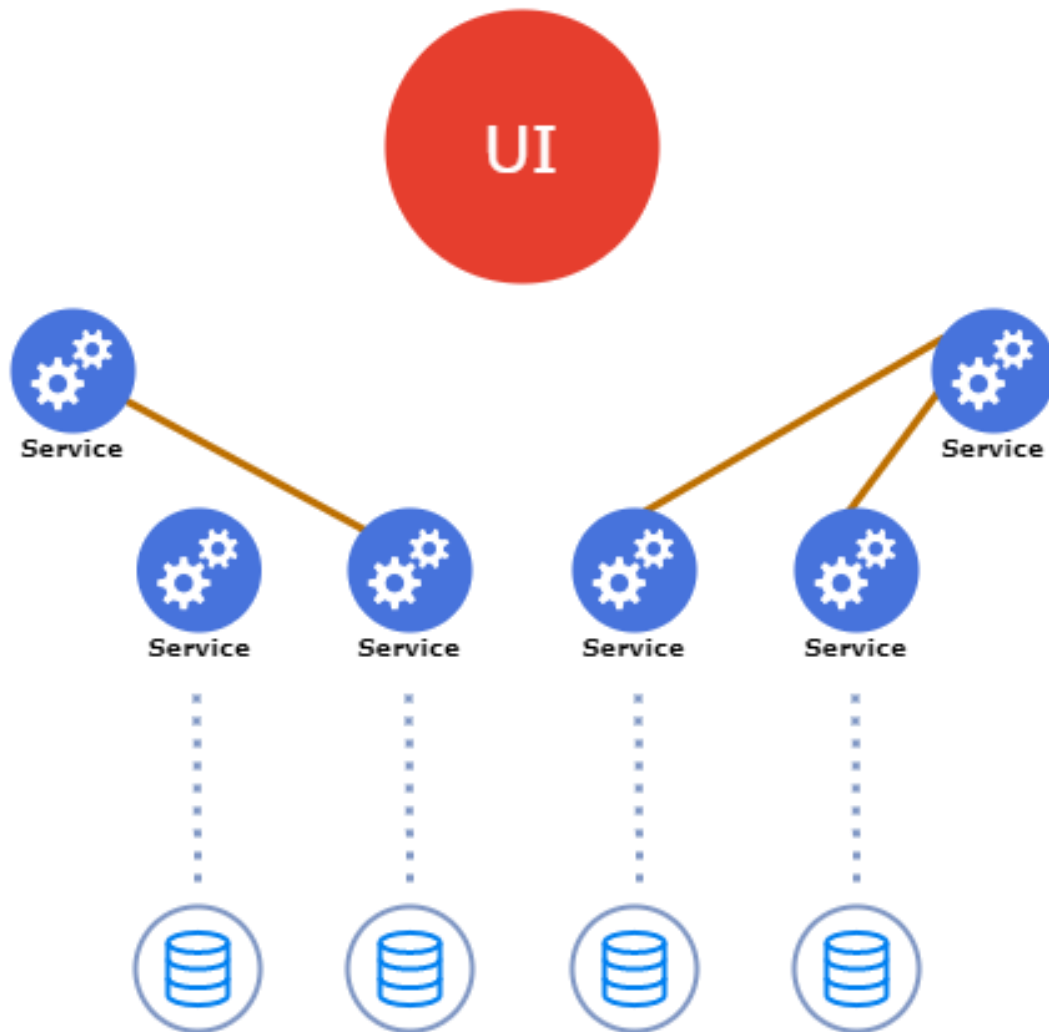
Monolitna arhitektura



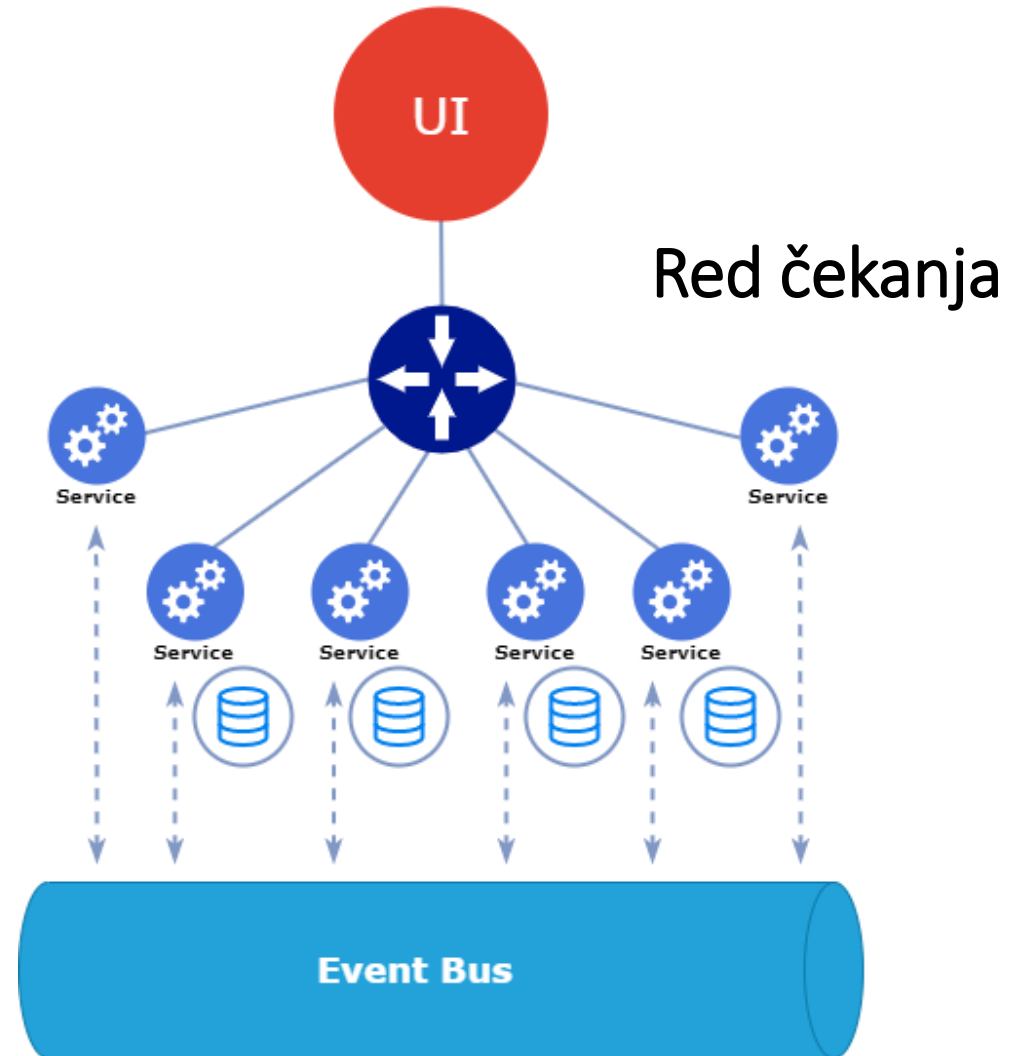
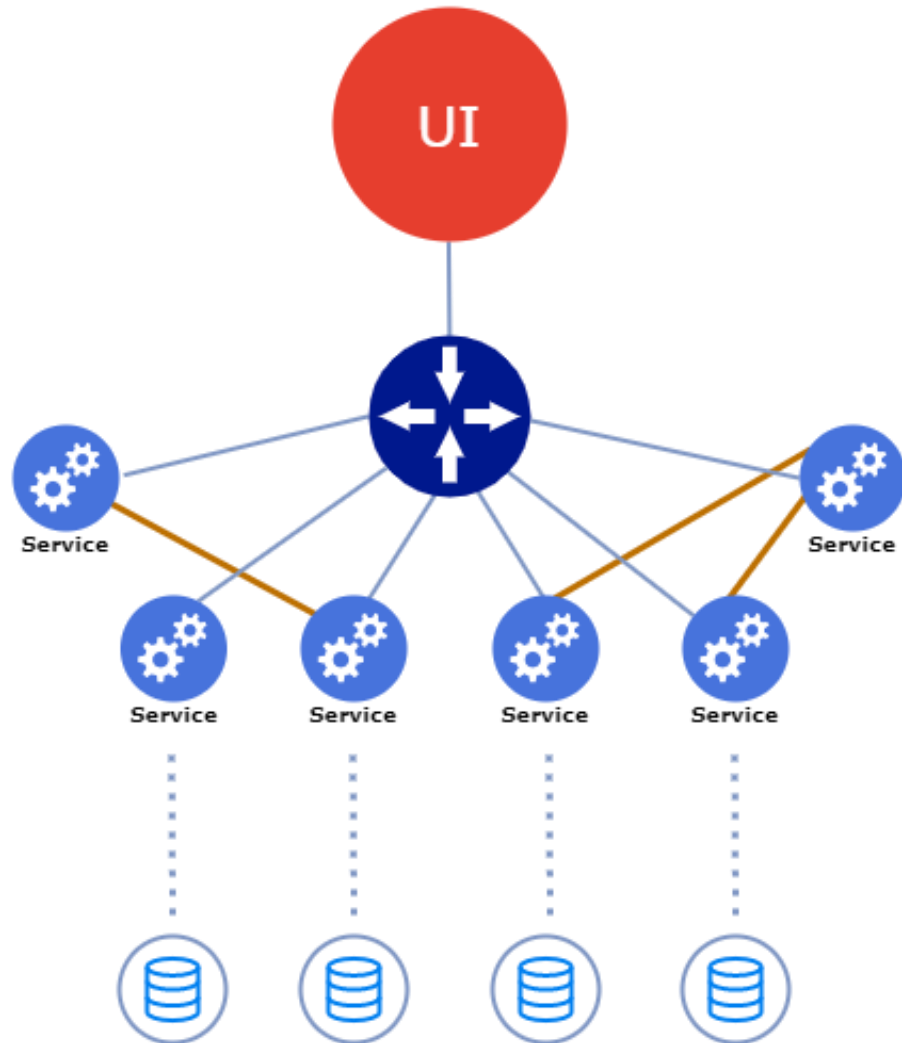
Servisna arhitektura



# Servis orijentisana arhitektura – problem više servisa




# Servis orijentisana arhitektura – problem komunikacije



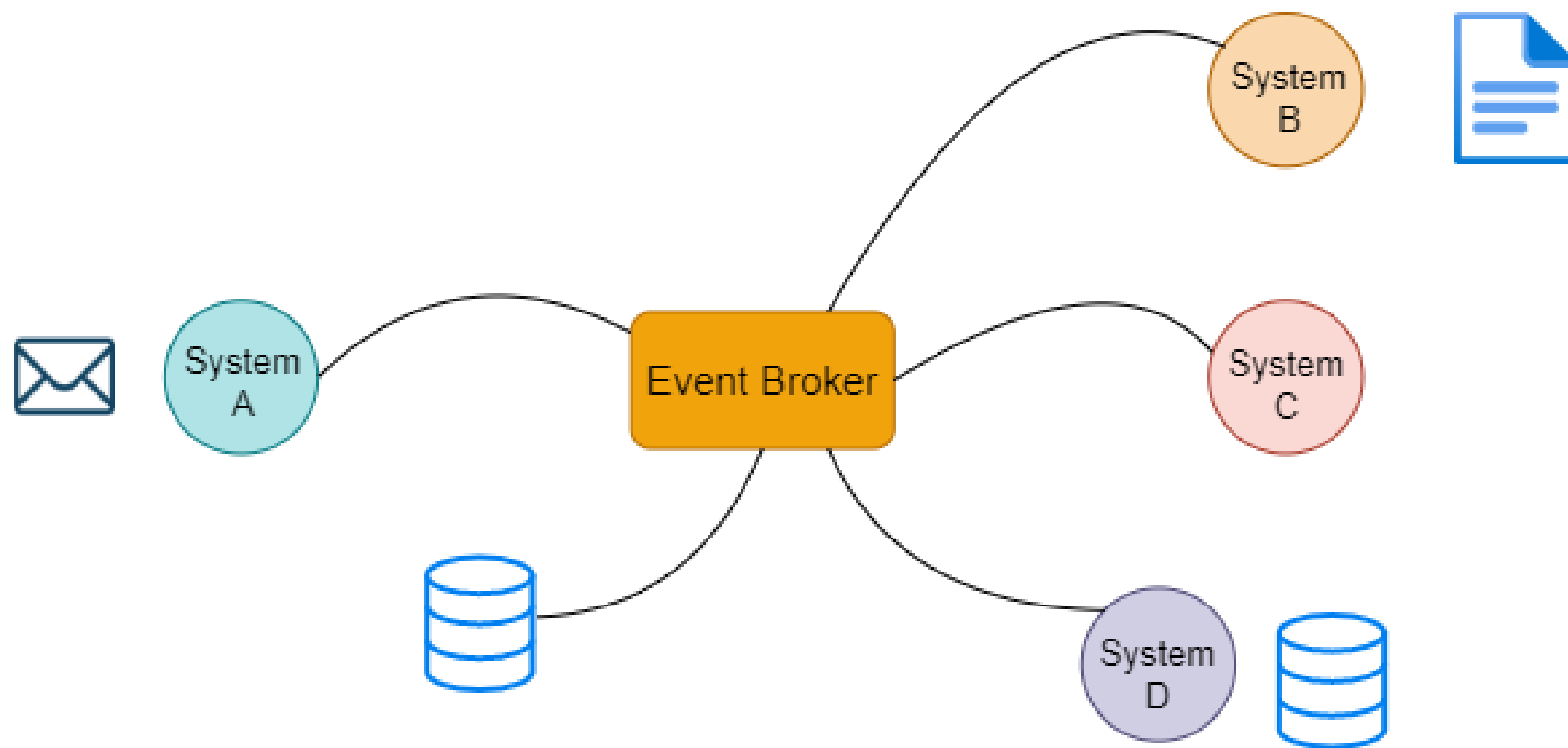


# Koncepti servis orijentisane arhitekture



- Modularnost
    - Više malih celina
  - Nezavisnost
    - Koje ne zavise jedna od druge
  - Asinhronost
    - I ne moraju biti uvek dostupne kako bi se funkcionalnost obavila
  - Event-driven arhitektura
    - Arhitektura bazirana na asinhronoj komunikaciji putem događaja
- 

# Event-driven arhitektura

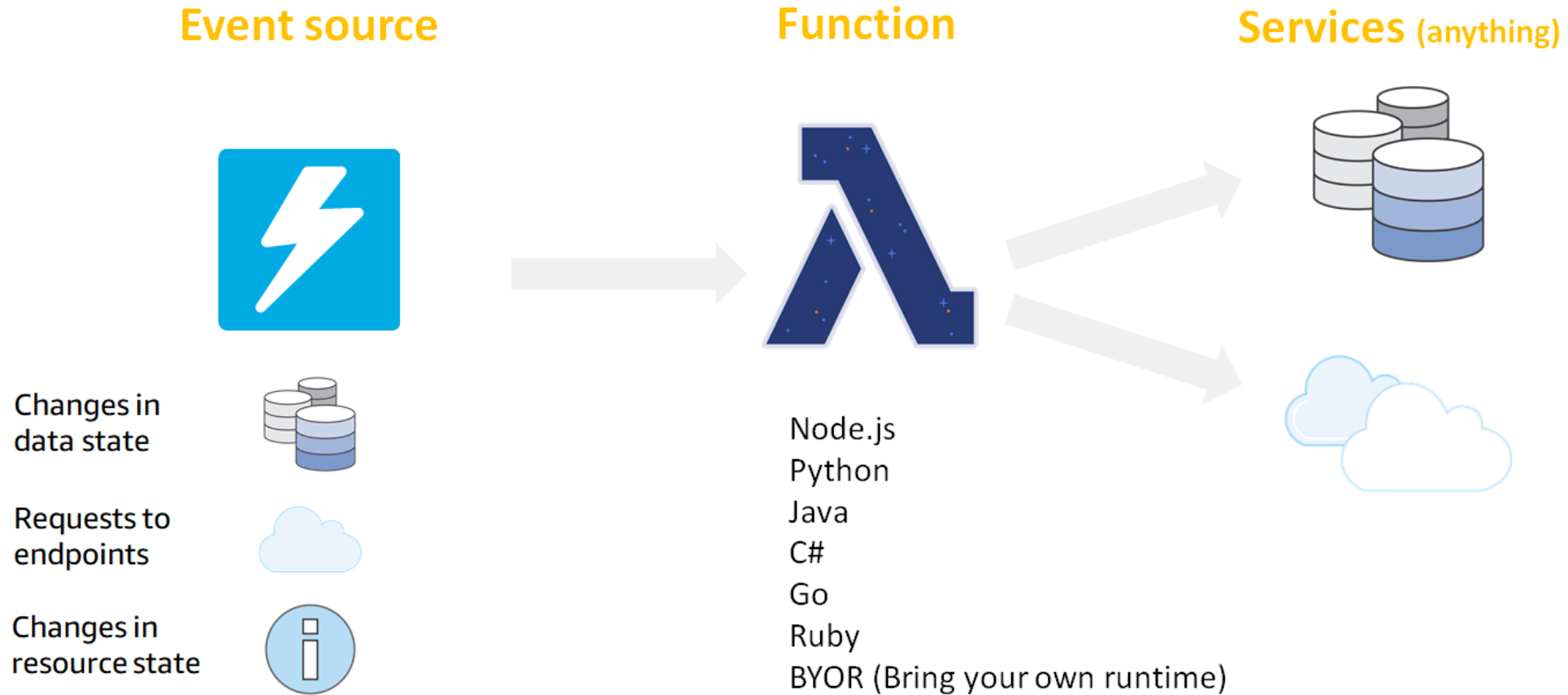


# Serverless VS servis orijentisana arhitektura

- Ograničenja serverless modela:
  - Instance nisu uvek aktivne
    - **Aktiviraju se kao odgovor na neki događaj** (korisnički, servisni, poziv od druge instance)
  - Kratkotrajno izvršavanje
    - Oko 15 minuta maksimalno
- Posledica:
  - Manje komponente od servisa
    - Funkcije
    - **Lambda funkcija**
- Jedan servis ~ jedna servisna klasa
- Jedna serverless funkcija ~ jedna servisna metoda

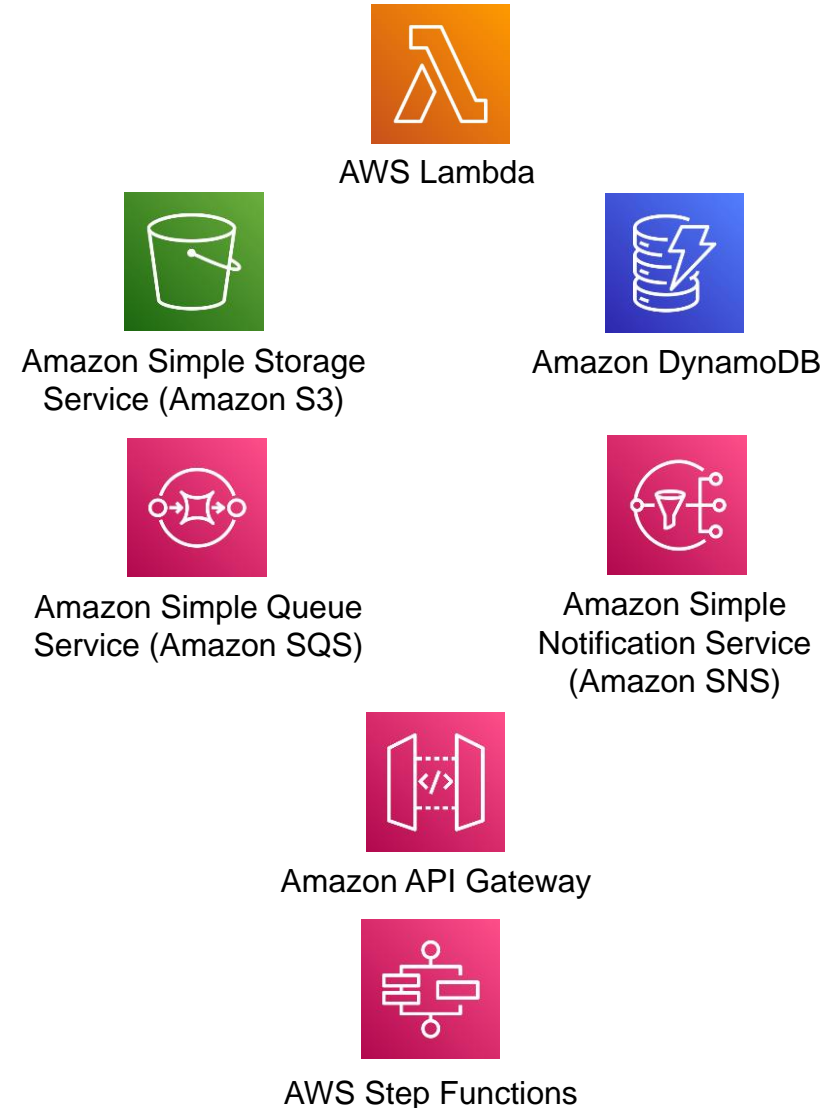


# Event-driven arhitektura + serverless model

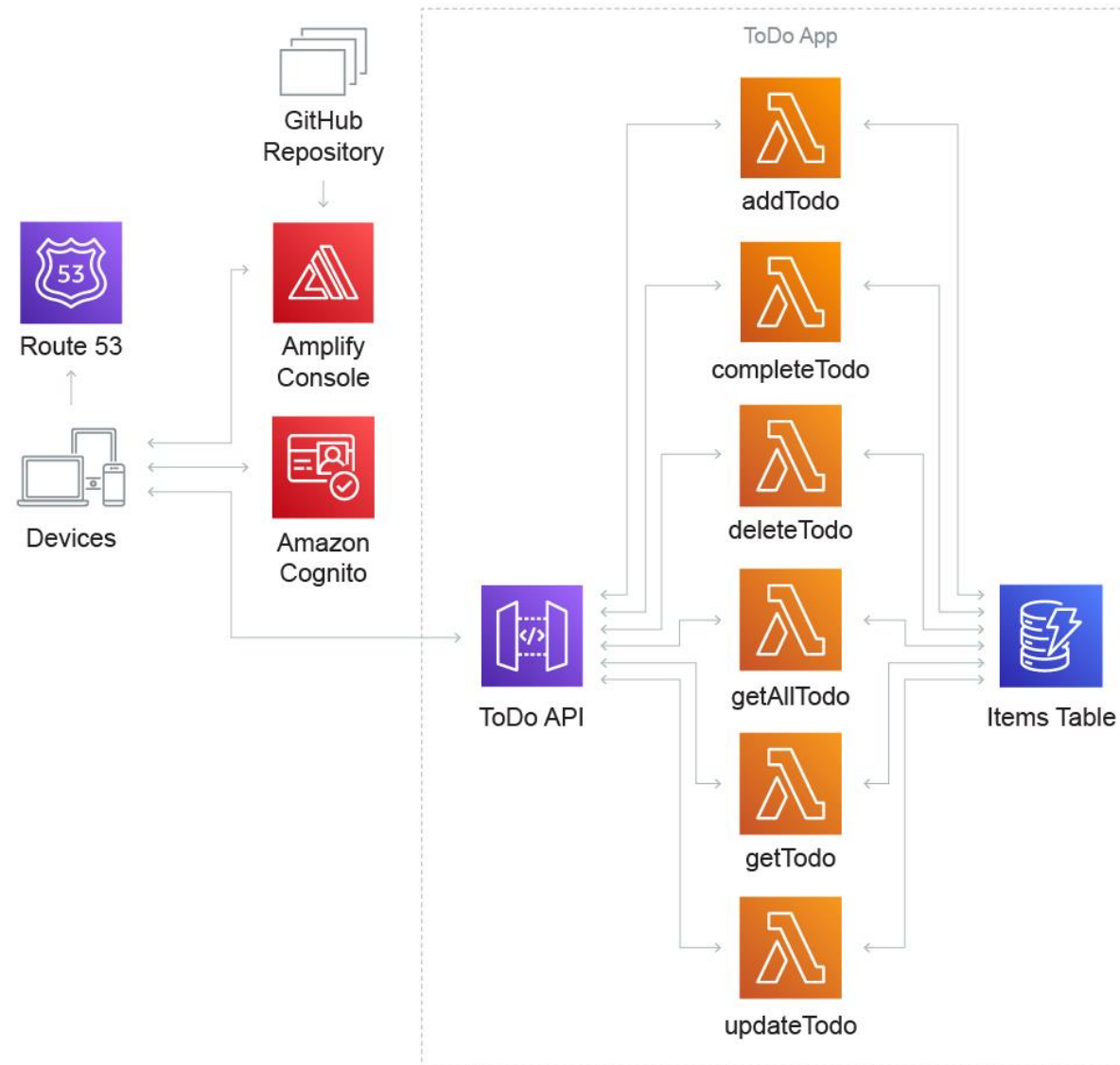


# AWS serverless stack

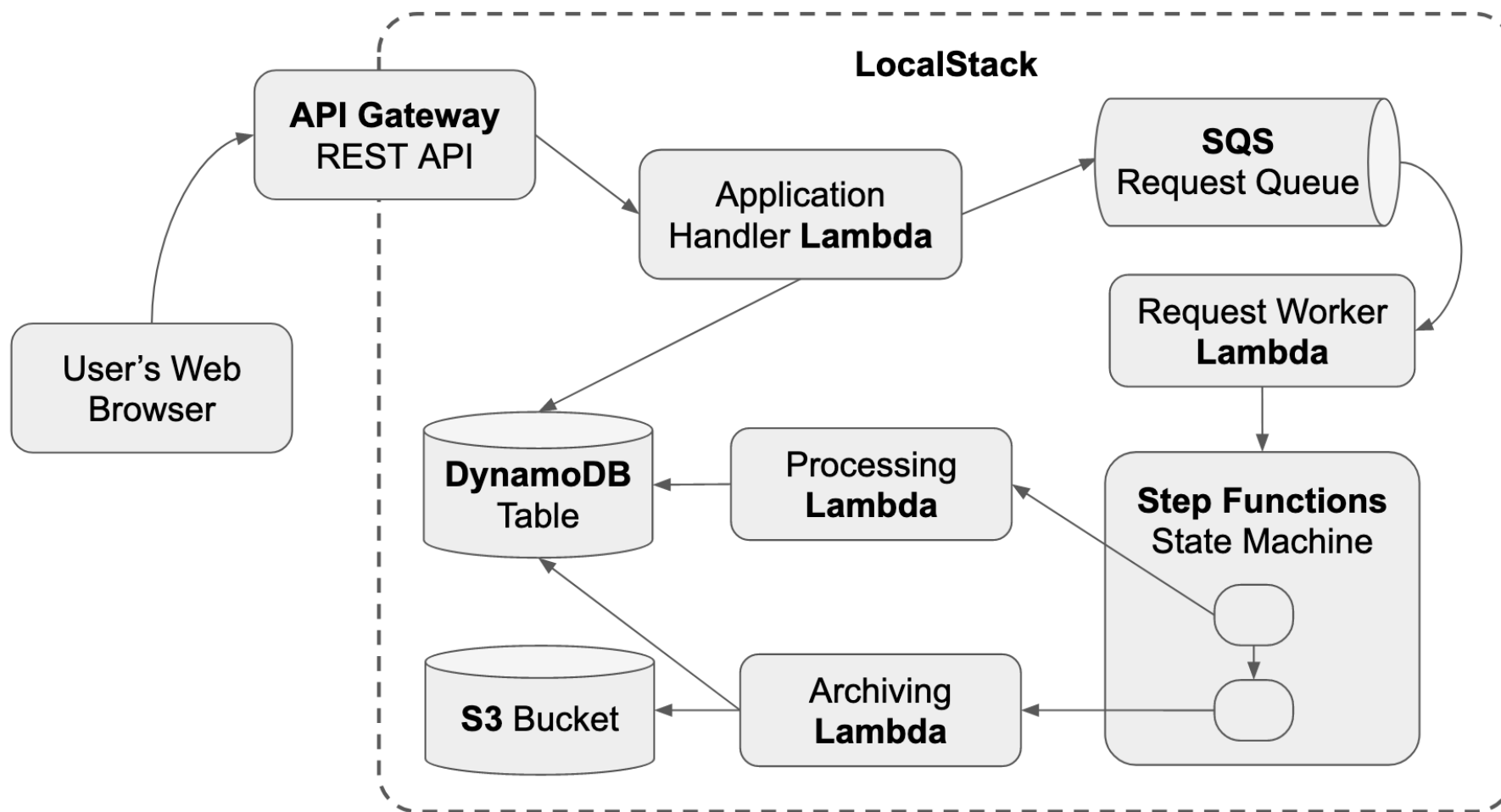
- Compute = Lambda
- Skladište
  - S3
  - DynamoDB
- Komunikacija
  - SQS
  - SNS
- Jedinствена tačka pristupa
  - API Gateway
- Orkestracija
  - Step funkcije



# Jednostavna web aplikacija



# Složenija web aplikacija



# Lambda

- Kod
- Function handler
  - Definiše ulaznu tačku prilikom poziva Lambda funkcije
  - *first\_lambda.hello\_world*
- Event
  - Događaj koji je pokrenuo izvršavanje Lambda funkcije
- Context
  - Kontekst u okviru koga se kod izvršava
    - Verzija funkcije, memorijski kapacitet, environment varijable

```
def hello_world (event, context):  
    print("value1 = " + event['key1'])  
    print("value2 = " + event['key2'])  
    print("value3 = " + event['key3'])  
    return event['key1']
```

*first\_lambda.py*





AWS Lambda



## Serverless – Lambda

# Zadaci



1. Kreirati *HelloWorld* Lambda funkciju u željenom programskom jeziku i isprobati je
2. Izdvojiti celine aplikacije u projektnom zadatku i namapirati ih na Lambda funkcije

\*[AWS Docs](#), [AWS CLI Docs](#), [LocalStack Docs](#)