Serverless aplikacije

Računarstvo u oblaku
(Cloud Computing)

Sadržaj

Uvod

- Serverless arhitekture je stil arhitekture aplikacija koji uključuje “Backend as a Service” (BaaS) usluge, ili/i sopstveni kod koji se izvršava u upravljanim izvršnim okruženjima na “Functions as a Service” (FaaS) platformi.
- Ove ideje, uz odgovarajući *front-end* (poput jednostraničnih aplikacija) omogućava da se odstupi od klasične instalacije na servere i potrebe da se aplikacije stalno održavaju na serverima.
- Mogu znatno smanjiti operativne troškove, mogu biti jednostavnije od klasičnih arhitektura, i mogu skratiti vreme razvoja aplikacija uz oslobađanje programera potrebe da se bave konfigurisanjem infrastrukturnih komponenti.
- Ali vas trajno vezuju za određenu platformu.

Uvod

- AWS, Google i Microsoft svi nude svoje servise za formiranje serverless aplikacija
  - https://aws.amazon.com/serverless/
  - https://cloud.google.com/functions
  - https://azure.microsoft.com/en-gb/products/functions
- Kao i sve drugo ni ovo nije „silver bullet“, za neke aplikacije nisu dobro rešenje

Dakle šta je serverless?

- Dva različita pristupa koja se do određene mere preklapaju
  - Serverless pristup je isprva opisivao aplikacije koje su se oslanjale na već gotove servise. Tipično „rich client“ koji kao backend koristi neke od ponuđenih cloud servica – BaaS
    - često API za pristup cloud hosted bazama (Firebase), servise za autentikaciju (AuthO).
  - Danas kada se govori o serverless aplikacijama često se misli na aplikacije kod kojih postoji dodatna, specifična programska logika, ali koja se izvršava u stateless kontejnerima koje obezbeđuje ponuđač usluge, bez potrebe da se razvojni tim bavi infrastrukturnim zadacima - FaaS

Primer – online shop aplikacija u klasičnom obliku

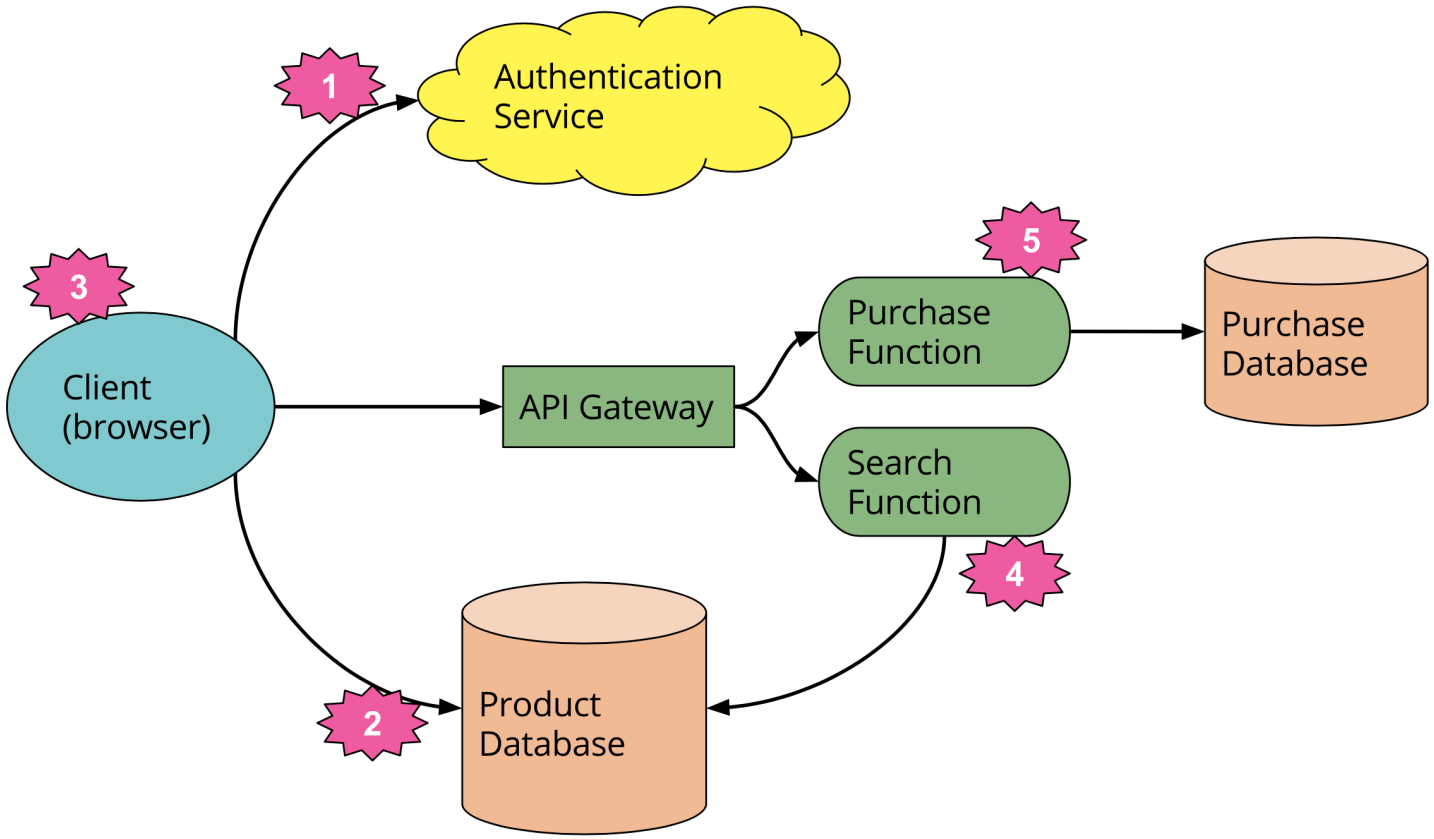
- Klasična troslojna aplikacija – npr. online prodavnica
  - Relativno jednostavna klijent
  - Backend server obavlja sve ključne operacije - autorizaciju, navigaciju, pretraživanje, kupovne transakcije

Primer – online shop aplikacija u serverless implementaciji

- U serverless implementacija aplikacija bi mogla da izgleda ovako:

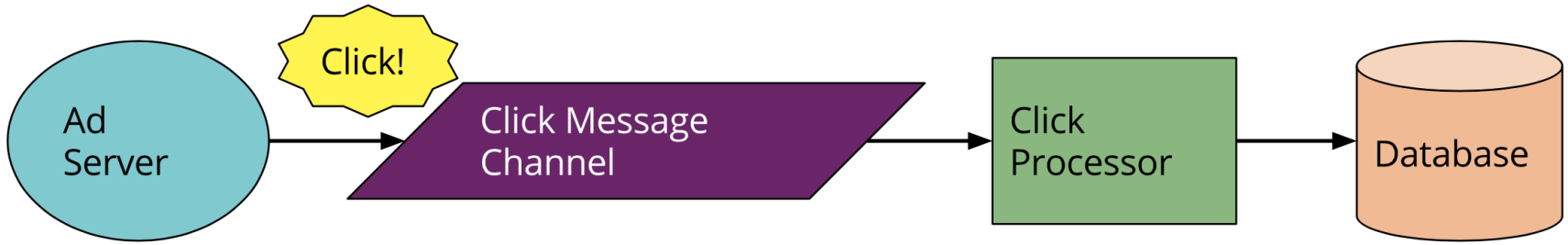
1. Autorizaciju sada obavlja specijalizovani servis
2. Klijent preko API ja ima pristup bazi proizvoda
3. API Gateway upravlja rutiranjem
4. Funkcionalnosti kupovine i pretraživanja su implementirane kao posebne funkcije (FaaS)

U serverless arhitekturi više nema centralnog modula koji upravlja svim operacijama – skup servisa sa jasnom namenom



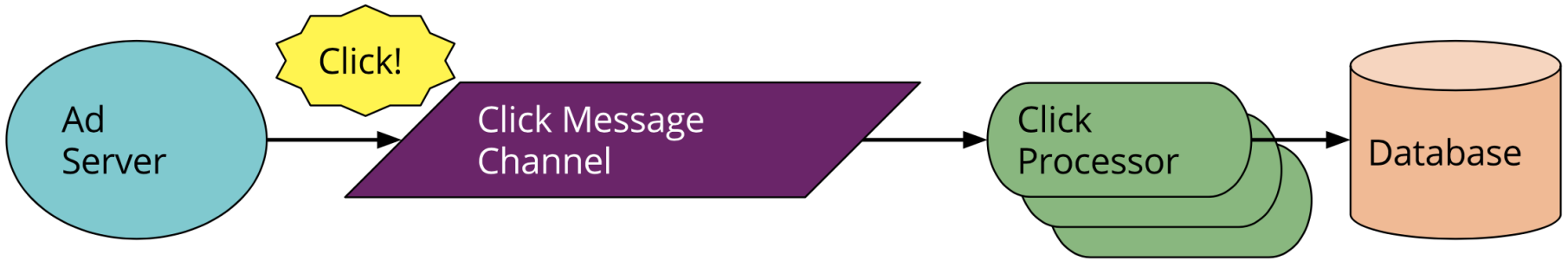
Primer – message driven aplikacija u klasičnom obliku

- Npr. aplikacija za procesiranje podataka o akcijama korisnika
  - Korisnički usmerena – npr. oglašavanje / želite brzu rekaciju na klik korisnika, usmeravanje na ciljani oglas, ali istovremeno i da zabeležite klik kako bi se evidentirao, uradila analiza korisničih akcija i naplatilli klikovi na oglase



Primer – obrada klikova na oglase u serverless implementaciji

- U serverless implementacija aplikacija bi mogla da izgleda ovako:



  - Ovde je izmena arhitekture mnogo manja, dugotrajni potrošač poruka koji je postojao u prethodnoj arhitekturi sada je zamenjen funckijama koje se okidaju na određene događaje
  - Cloud provider obezbeđuje i usluge message brokera i sam FaaS kontejner
  - Paralelizacija (istovremeno izvršavanje više instanci) je ovde ugrađeno svojstvo

Ključne osobine FaaS pristupa

- Suštinski obezbeđuje mogućnost izvršavanja backend koda bez opterećenja održavanja sopstvenog serverskog sistema i dugotrajnih (*long lived*) serverskih aplikacija. Funkcije se aktiviraju po potrebi i daktiviraju kada su neaktivne (razlika u odnosu na PaaS)
- Ne zahteva se kodiranje u skladu sa nekim posebnim razvojnim okvirom. Standardni kod funkcija. Ali mora biti napisan u podržanim jezicima (na AWS JavaScript/TypeScript, Python, Go, bilo koji JVM jezik, ili bilo koji .NET jezik).
  - Jedina izmena u odnosu na kod koji bi se izvršavao u drugom okruženju je neophodnost pisanja specifične event handler „ulazne“ funkcije

Ključne osobine FaaS pristupa

- Instalacija je specifična
  - Uploaduje se sam kod funkcije
- Horizontalno skaliranje je u potpunosti automatizovano, elastično i obezbeđuje ga ponuđač usluge
- Okidanje funkcije je tipično na osnovu tipova događaja koje ponuđač usluge definiše
- Moguće je i okidanje na HTTP zahtev – preko API Gateway-a

Osobine funkcija u FaaS

- State – stanje
  - Funkcija nema stanje i ne možete se osloniti na čuvanje nekih podataka između dve aktivacije funkcija ili na deljenje podataka između više pokrenutih instanci funkcija
  - Ukoliko nešto od podataka treba trajno da se čuva ta operacija mora biti eksternalizovana na neki dodatni servis
- Trajanje izvršavanja
  - Postoji limit koliko dugo funkcija može da se izvršava
    - Određeni zadaci koji zahtevaju funkcije koje dugo ostaju u aktivnom stanu nisu pogodne za ovakvu implementaciju

Osobine funkcija u FaaS

- Kašnjenje pri pokretanje i „hladni start“
  - Uvek postoji kašnjenje dok FaaS platforma inicijalizuje funkciju
  - Hladni start ako ne postoji nijedna instanca funkcije koja je već inicijalizovana, a trenutno ne radi ništa
  - Topli start – iskoristi se već inicijalizovana funkcija, koja posle završetka ostaje izvesno vreme učitana u memoriji
- Upotreba API gatewaya
  - Rutiraju saobraćaj i ako je ruta mapirana na FaaS funkciju prepakuju podatke iz zahteva u oblik koji funkcija očekuje i prosleđuju joj te podatke, aktivirajući događaj na koji funkcija reaguje

Prednosti ovog pristupa

- Smanjeni operativni troškovi
- Smanjeni troškovi razvoja
- Dobro reagovanje na nekonzistentne obrasce opterećenja
  - Povremeni saobraćaj ili nagli skokovi opterećenja
- Smanjena kompleksnost isporuke (packaging and deployment)
- „Greener computing“?

Nedostaci ovog pristupa

- Čvrsto vezivanje uz platformu određenog isporučioca
- Multitenant problemi
- Pitanje bezbednosti
  - Povećan broj komponenti za koje je potrebno obezbediti dobru kontrolu prava pristupa
  - BaaS pristup direktno bazama preko API-ja uklanja “štit“ koji naš specifičan backend može da kreira
- Nepostojanje čuvanja stanja na serverskoj strani

Zaključak

- Pitanja?