

UNIVERZITET U NIŠU  
ELEKTRONSKI FAKULTET

**KNATIVE – FUNCTION AS A SERVICE PLATFORMA**

Seminarski rad

Predmet: Servisno-orijentisane arhitekture

Student: Mentor:

Jovana Spasić, br. ind. 17407 Prof. dr Dragan Stojanović

Niš, maj 2022. godina

**SADRŽAJ**

[1. UVOD U FUNCTION AS A SERVICE 1](#_Toc103943469)

[**1.1. Dobre prakse pri razvoju funkcija** 1](#_Toc103943470)

[2. KNATIVE 2](#_Toc103943471)

[**2.1. Glavne komponente** 2](#_Toc103943472)

[3. KNATIVE EVENTING 4](#_Toc103943473)

[**3.1. Komponente** 4](#_Toc103943474)

[**Event sources** 5](#_Toc103943475)

[**Broker and Trigger** 5](#_Toc103943476)

[**Channel and Subscription** 5](#_Toc103943477)

[**Event registry** 5](#_Toc103943478)

[**3.2. Usage Patterns** 5](#_Toc103943479)

[**Source to Sink** 5](#_Toc103943480)

[**Channel and Subscription** 6](#_Toc103943481)

[**Broker and Trigger** 6](#_Toc103943482)

[4. KNATIVE SERVING 8](#_Toc103943483)

[**4.1. Komponente** 8](#_Toc103943484)

# **1. UVOD U FUNCTION AS A SERVICE**

*Function as a Service* (FaaS) predstavlja vrstu usluge računarstva u oblaku koja programerima omogućava kreiranje, pokretanje, kao i upravljanje aplikacijama kao funkcijama bez potrebe održavanja sopstvene infrastrukture. FaaS je tzv. *event-driven* model izvršavanja koji radi u *stateless* kontejnerima i kod koga funkcije upravljaju *server-side* logikom.

Aplikacije koje se nalaze na Internetu uobičajeno zahtevaju obezbeđivanje i upravljanje virtuelnim ili fizičkim serverom, kao i upravljanje operativnim sistemom i *web* serverom. Kada se koristi FaaS, tada fizičkim hardverom, operativnim sistemom virtuelne mašine i serverom upravlja *cloud service provider*. Takav princip funkcionisanja omogućava programeru da se fokusira isključivo na pojedinačne funkcije u kodu.

FaaS rešenja su dostupna na svim velikim javnim *cloud* platformama. Neki popularniji primeri uključuju:

* IBM Cloud Functions
* Amazon’s AWS Lambda
* Google Cloud Functions
* Microsoft Azure Functions
* OpenFaaS

## **1.1. Dobre prakse pri razvoju funkcija**

FaaS funkcije je potrebno dizajnirati tako da rade samo deo posla koji odgovara nekom događaju. Pisanje efikasnog, *lightweight* koda limitiranog *scope*-a omogućava brže učitavanje i izvršavanje funkcija.

Funkcije se ne bi trebale međusobno pozivati. Previše poziva funkcija dovodi do povećanja troškova, kao i uklanjanja vrednosti izolacije funkcije.

Korišćenje što je moguće manjeg broja biblioteka u funkcijama sprečava sporo izvršavanje istih. Takođe, ukoliko se unutar funkcije koristi prevelik broj biblioteka, može se znatno otežati skaliranje takve funkcije.

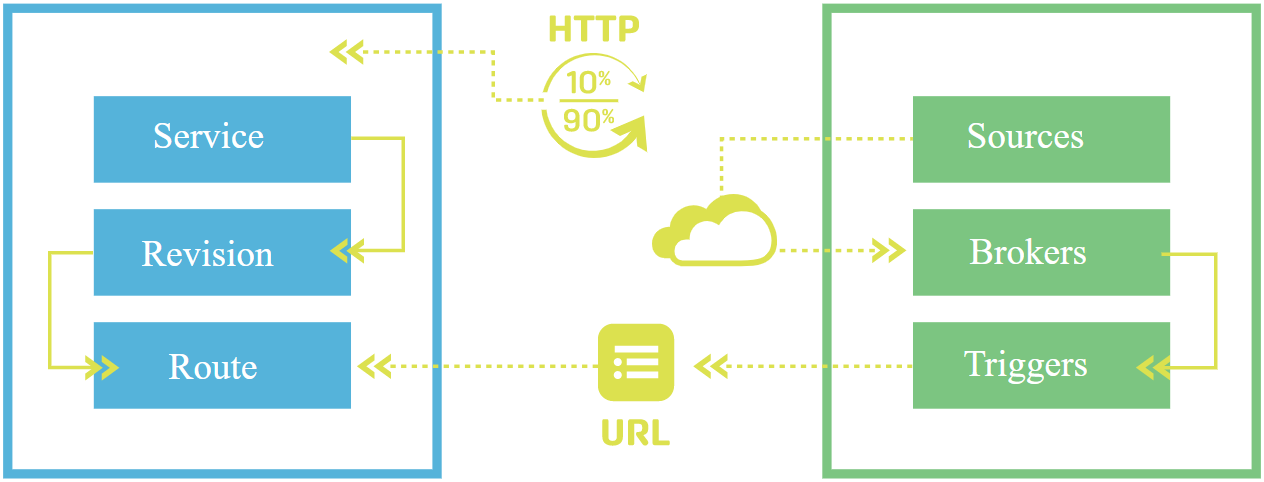
# **2. KNATIVE**

Knative predstavlja platformu baziranu na Kubernetes-u koja služi za razvoj, *deployment* i upravljanje modernim *serverless* rešenjima. Kubernetes, takođe poznat i kao K8s, predstavlja *open-source* sistem za automatizaciju *deployment*-a, skaliranje i upravljanje aplikacijama koje se izvršavaju u kontejnerima. Radi lakšeg upravljanja i otkrivanja, grupiše kontejnere koji čine aplikaciju u logičke jedinice. Knative je takođe kreiran od strane Google-a i pruža set komponenti za razvoj i pokretanje *serverless* aplikacija na Kubernetes-u. Pruža funkcije poput: *scale-to-zero*, automatskog skaliranja, *in-cluster builds* i *eventing framework*-a za *cloud-native* aplikacije na Kubernetes-u. Štaviše, Knative dozvoljava programerima da se fokusiraju na sam proces pisanja koda bez potrebe za brigom o „dosadnim, ali teškim“ delovima *deployment*-a i upravljanja aplikacijama.

Nudi skup *reusable* komponenti koje se fokusiraju na rešavanje svakodnevnih zadataka kao što su *workflows* poput: *build*-ovanja aplikacija od izvornog koda do slika, rutiranja i upravljanja saobraćajem tokom *deployment*-a, automatskog skaliranja ili *bind*-ovanja pokrenutih servisa za rastući ekosistem izvora događaja. Knative prepoznaje kontejnerske slike kao jedinice za *deployment*, što omogućava programerima korišćenje bilo kog jezika ili *framework*-a sa kojim su upoznati.

Obezbeđuje otvoreni API i *runtime environment* koji omogućava pokretanje *serverless workload*-a sa bilo kog „mesta“: Google Cloud-a, Anthos-a na Google Kubernetes Engine-u ili sopstvenog Kubernetes klastera.

## **2.1. Glavne komponente**

Knative poseduje dve glavne komponente koje olakšavaju timovima rad sa Kubernetes-om. Te komponente su Serving i Eventing i one rade zajedno na automatizaciji i upravljanju zadacima i aplikacijama.

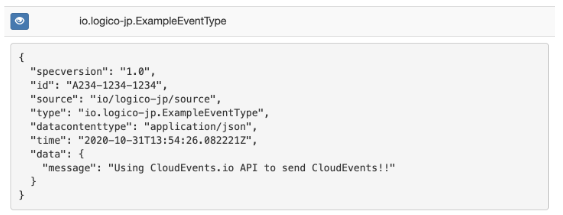
**Slika 2.1.** *Glavne komponente Knative-a: Serving (sa leve strane) i Eventing (sa desne strane)*

Eventing dozvoljava programerima da postave detaljne akcije koje su *trigger*-ovane specifičnim događajem u nekom širem okruženju. Sa druge strane, Serving vrši automatsko upravljanje procesom kreiranja i skaliranja servisa kroz Kubernetes. Obe komponente za cilj imaju smanjenje iskorišćenosti resursa koje bi timovi inače koristili za upravljanje sistemima. Pored toga, reaguju na određene uslove u realnom vremenu, što znači da kompanije plaćaju samo resurse koje koriste, ne i one koje bi možda koristile.

Prethodno pomenuti *scale-to-zero* predstavlja karakteristiku Serving komponente. Ukoliko nema potrebe za nekim servisom koji je pokrenut unutar kontejnera, tada se on automatski isključuje. Kada se javi potreba za servisom, izvršiće se automatsko uključivanje. Ovakav princip smanjuje troškove i pomaže pri upravljanju tehničkim resursima.

# **3. KNATIVE EVENTING**

Knative Eventing pruža alate za rutiranje događaja od *event producer*-a do *sink*-a, odnosno potrošača, što programerima pruža mogućnost korišćenja *event-driven* arhitekture u aplikacijama. Resursi su slabo spregnuti i mogu se razvijati i *deploy*-ovati nezavisno jedni od drugih. Bilo koji proizvođač može generisati događaje pre nego da se javi aktivni potrošač koji osluškuje takve događaje. Takođe, bilo koji potrošač može izraziti interesovanje za nekim događajem ili klasom događaja, pre nego što ih proizvođači kreiraju.

Koristi standardne HTTP POST zahteve za slanje i prijem događaja između proizvođača događaja i potrošača. Ti događaji su u skladu sa CloudEvents specifikacijama, što omogućava kreiranje, parsiranje, slanje i prijem događaja u bilo kom programskom jeziku.

**Slika 3.1.** *Primer CloudEvents specifikacije*

## **3.1. Komponente**

*Event-driven* arhitektura je bazirana na konceptu razdvajanja veza između proizvođača događaja i potrošača tih događaja ili *sinks*-a koji primaju te događaje. *Sink* se koristi pri specifikaciji toga gde će događaji sa izvorišta biti poslati. Reč je o tzv. *Addressable* ili *Callable* resursu koji može primiti dolazeće događaje iz drugih resursa. Primeri *sink*-a su: Knative Services, Channels i Brokers.

*Addressable* objekti primaju i potvrđuju događaj koji je pristigao preko HTTP protokola na adresu definisanu u njihovom **status.address.url** polju. Nasuprot njima, *Callable* objekti mogu primiti događaj koji je pristigao preko HTTP protokola i transformisati ga, vraćajući 0 ili 1 novi događaj kroz HTTP odgovor. Vraćeni događaji se mogu dalje obrađivati na isti način na koji se obrađuju događaji iz spoljašnjeg izvora događaja.

### ***Event sources***

*Event sources* predstavljaju primarne proizvođače događaja. Događaji se šalju na *sink* ili pretplatniku.

### ***Broker and Trigger***

Pružaju tzv. *event mesh* model, koji dozvoljava proizvođaču događaja da dostavi događaje *Broker*-u, koji ih zatim uniformno distribuira potrošačima korišćenjem *Trigger*-a. Dakle, potrošači se mogu registrovati na specifične tipove događaja bez potrebe da direktno pregovaraju sa proizvođačima događaja. Takođe, moguće je izvršiti optimizaciju rutiranja događaja.

### ***Channel and Subscription***

Pružaju tzv. *event pipe* model koji transformiše i rutira događaje između *Channel*-a korišćenjem *Subscription*-a. Ovakav model je pogodan za *event pipeline* kod koga događaji iz jednog sistema trebaju biti transformisani i preusmereni na neki drugi proces.

### ***Event registry***

Knative Eventing definiše EventType objekat kako bi olakšao potrošačima „otkrivanje“ tipova događaja koje mogu koristiti iz *Broker*-a. Registar se sastoji od kolekcije tipova događaja. Tipovi događaja koji se nalaze unutar registra sadrže potrebne informacije za potrošača, kako bi kreirao *Trigger* bez pribegavanja nekom drugom mehanizmu.

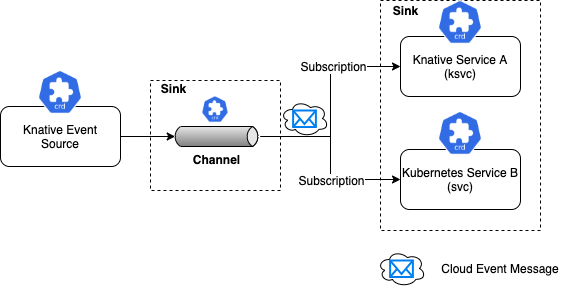
## **3.2. Usage Patterns**

### ***Source to Sink***

Pruža najjednostavnije iskustvo sa Knative Eventing komponentom. Obezbeđuje jedan *Sink* koji predstavlja servis koji prima događaj, bez redova čekanja i filtriranja. Ovakav obrazac korišćenja ne podržava odgovore, što znači da će odgovor servisa koji prima događaj biti ignorisan. Odgovornost *Event Source*-a je da samo dostavi poruku bez čekanja na odgovor.

**Slika 3.2.** *Source to Sink obrazac korišćenja*

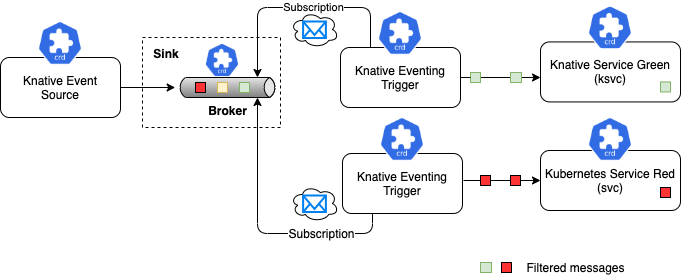
### ***Channel and Subscription***

Sa ovim obrascem korišćenja, Knative Eventing sistem definiše *Channel* koji se može povezati na razne servise, npr. *In-Memory*, Kafka i *PubSub*. Svaki *Channel* može imati jednog ili više pretplatnika u formi *Sink* servisa, kao što je prikazano na slici 3.3. Svaki *Sink* servis može primiti poruke o događajima i po potrebi ih obraditi. Svaka poruka koju *Channel* šalje je formatirana tako da zadovoljava CloudEvents specifikaciju. Ovaj obrazac takođe nema podršku za filtriranje poruka.

**Slika 3.3.** *Channels and Subscriptions*

### ***Broker and Trigger***

Ovaj obrazac korišćenja je sličan obrascu *Channel and Subscription*, s tom razlikom da podržava filtriranje događaja. *Event filtering* je metoda koja dozvoljava pretplatnicima da iskažu interesovanje samo za određeni skup poruka koje dolaze u *Broker*. Za svaki *Broker*, Knative Eventing implicitno kreira *Knative Eventing Channel*. Sa slike 3.4. je moguće videti da se *Trigger* pretplaćuje na *Broker* i primenjuje filter na poruke iz tog *Broker*-a. Filteri se primenjuju nad CloudEvent atributima poruke, pre nego što se dostave zainteresovanim pretplatnicima.

**Slika 3.4.** *Brokers and Triggers*

# **4. KNATIVE SERVING**

Idealan je za pokretanje aplikacionih servisa unutar Kubernetes-a, omogućavajući jednostavniju sintaksu za *deployment* sa automatskim *scale-to-zero* principom. Sam Knative će upravljati *deployment*-om, revizijama i skaliranjem.

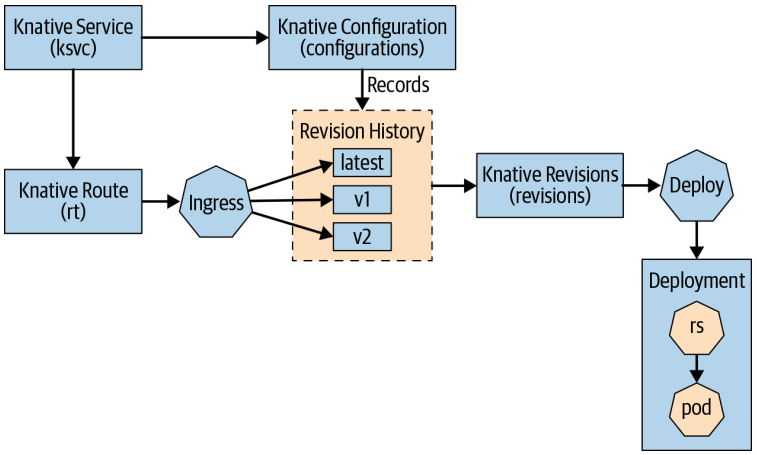
Servis je *expose*-ovan preko HTTP URL-a i ima dosta podrazumevanih podešavanja u konfiguracijama. Za većinu praktičnih upotreba, može se desiti da je neophodno promeniti neka podrazumevana podešavanja. Takođe, ponekad je potrebno podesiti distribuciju saobraćaja između revizija servisa.

Knative Serving obezbeđuje komponente koje omogućavaju:

* Brz *deployment serverless* kontejnera
* Automatsko skaliranje, uključujući *scale-to-zero*
* Podršku za razne mrežne slojeve, kao što je Istio, za integraciju sa postojećim okruženjima
* *Snapshot*-ove koda i konfiguracija

Podržava i HTTP i HTTPS protokol.

## **4.1. Komponente**

Tokom *deployment*-a Knative Serving servisa, Knative Serving kontroler kreira: *Configuration*, *Revision* i *Route*.

**Slika 4.1.** *Knative Serving komponente*

*Configuration* održava željeno stanje *deployment*-a, tako što omogućava odvajanje koda i konfiguracije korišćenjem *twelve-factor app* principa razvoja. Na osnovu željenog stanja, *Knative Configuration* kontroler kreira novi *Kubernetes Deployment* za aplikaciju. Treba imati na umu da svaka promena konfiguracije rezultuje kreiranjem novog *Kubernetes Deployment*-a.

Pošto se koriste *twelve-factor app* principi, svaka promena konfiguracije aplikacije takođe kreira i nov *Knative Revision* koji se može posmatrati kao *version control tag*. Nije moguće promeniti reviziju. Svaki *Knative Revision* ima odgovarajući *Kubernetes Deployment* sa kojim je povezan, što dozvoljava aplikaciji vraćanje na prethodnu konfiguraciju za koju se zna da je bila validna.

*Knative Route* predstavlja URL preko koga je moguće pristupiti Knative servisu.