OPENWHISK

‘SERVERLESS’ PLATFORMA VOĐENA DOGAĐAJIMA

Mentor:
Prof. Dr Dragan Stojanović

Student:
Stefan Aleksić, 16995
Teodora Kocić, 17190

OPENWHISK – OPŠTE KARAKTERISTIKE

- Obezbeđuje skalabilnost pri čemu korisnik ne mora da vodi računa o tome
- Pruža korisnicima korišćenje moćnih alata
- Podrška za rad u raznim programskim jezicima: Python, JavaScript, Go, Java, PHP, Swift, .NET Core, Swift
- Open-source projekat kompanije Apache
- Mogućnost korišćenja za: javne, privatne i hibridne modele

GDE SE KORISTI OPENWHISK?

- **Digitalne aplikacije**

  OpenWhisk se primenjuje za različite mobilne, web i IoT aplikacije u svrhu uprošćavanja orkestracije raznih servisa, vodeći se događajima, pri čemu sada ne postoji poseban bekend za te servise.

- **Big Data**

  Kompleksni podaci mogu biti zabeleženi korišćenjem promena u tokovima podataka, odnosno servisima. Ovo se koristi kod analiza koje je potrebno izvršiti u realnom vremenu.

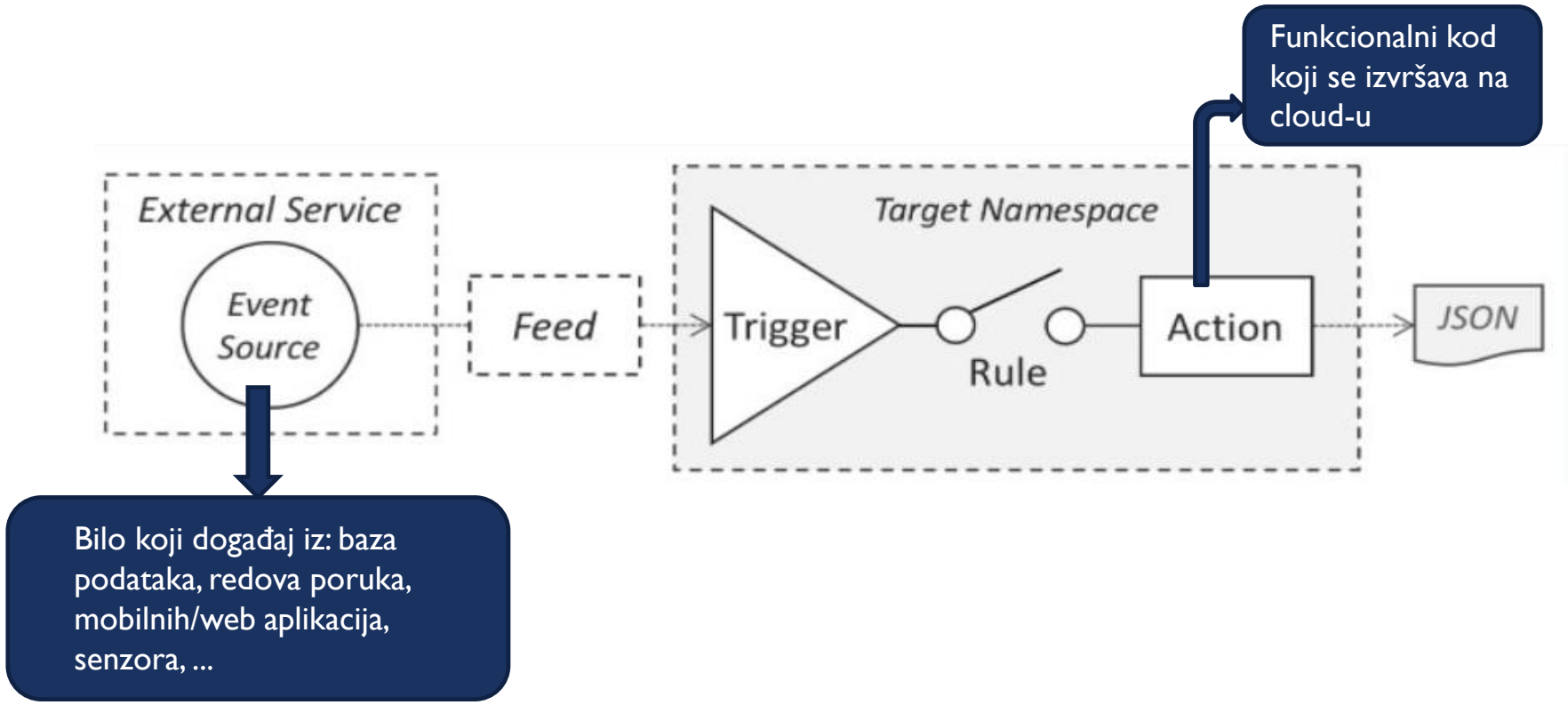
- **DevOps i infrastrustura kao kod**

  OpenWhisk se koristi i za automatizaciju DevOps tokova podataka.

- **Mikroservisi**

  OpenWhisk se korsiti za lako kreiranje mikroservisa na osnovu modela koji treba da ispunjava zadati mikroservis.

MODEL PROGRAMIRANJA

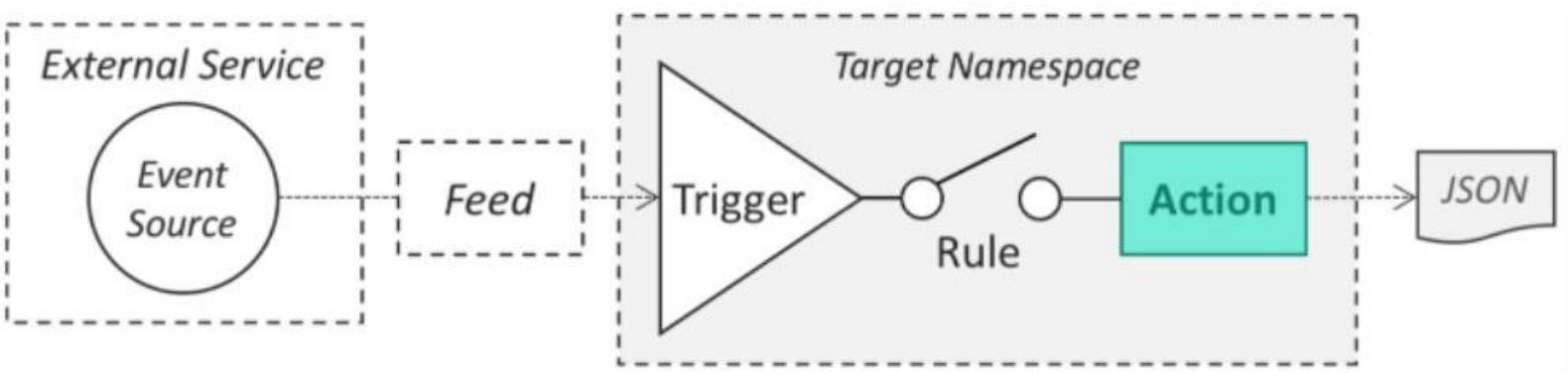


MODEL PROGRAMIRANJA

- Izvor podataka definiše događaj koji se emituje kao *trigere* (triggers). Programer kreira *akcije* (actions) koje imaju zadatak da obrade događaj (handle-uju) pomoću definisanih *pravila* (rules). Takozvani TAR model.

AKCIJE

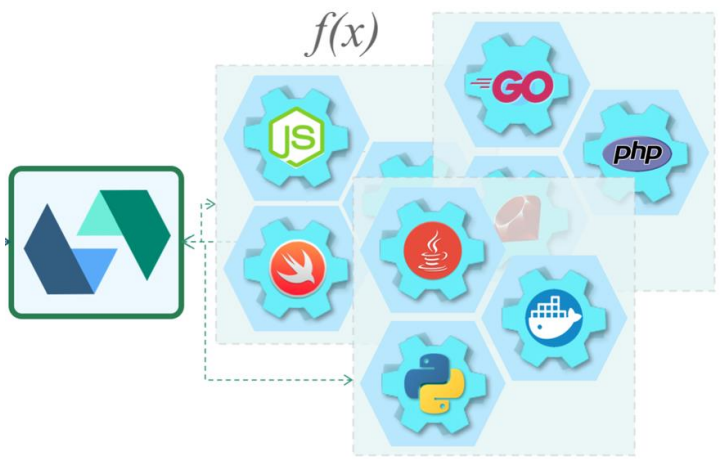
- Delovi koda koji se izvršavaju na OpenWhisk-u. One predstavljaju aplikacionu logiku koja će se izvršavati kao odgovor na događaje. U suštini akcija je odgovr na neki događaj i kao rezultat daje neki izlaz koji korisnici mogu videti i koristiti dalje u implementaciji ukoliko im je potreban taj skup podataka.
- U projekat se dodaju putem: REST API-a OpenWhisk-a, OpenWhisk CLI, korisnički definisanih REST API-a, trigera.



AKCIJE

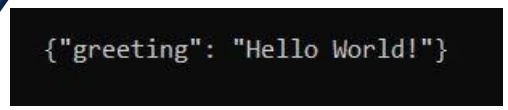
- Da li je bitno u kojem programskom jeziku je *akcija* napisana?
- Šta se dešava u slučaju da jezik u kojem pišemo program nije podržan od strane OpenWhisk-a?

- Ne. Operacije koje se koriste u OpenWhisk-u za kreiranje i upravljanje akcijama su iste nezavisno od jezika u kojem su akcije implementirane.
- OpenWhisk platforma je proširljiva tako da je lako dodati novi programski jezik ili sopstvene pakete (runtime) koristeći doker.



AKCIJE – PRIMERI

- Akcije mogu biti korišćene u svrhu detektovanja lica na slici, mogu predstavljati odgovor na neku promenu u bazi podataka ili odgovor na API poziv ili recimo postavljanje objave na Tviteru.



AKCIJE – PRIMERI

```
C:\Users\Tea\Desktop\VIII semestar\SOA\OpenWhisk\openwhisk-python-twilio>wsk action create --docker textAction tea2904/openwhisk
```

```
C:\Users\Tea\Desktop\VIII semestar\SOA\OpenWhisk\openwhisk-python-twilio>wsk action update textAction --param account_sid "ACfb27ab6cb8399b2b29969a53ba3ac9cc"
 --param auth_token "722ecfc35ffd9f0b0279b983fd03eae2"
```

- Prva komanda postavlja doker OpenWhisk akciju *textAction* koja je implementirana korišćenjem $DOCKER_USER/openwhisk slike sa Docker Hub-a.
- Druga komanda povezuje akciju *textAction* sa nalogom na aplikaciji Twillio i od prethodne komande razlikuje se po tome što ne predstavlja kreiranje akcije, već definiše ažuriranje postojeće akcije *textAction*.

AKCIJE – IZVRŠENJE

- Kada se akcija pozove na izvršenje, sistem zabeleži poziv i nakon toga akcija počinje da se izvršava.
- U slučaju neblokirajućeg poziva, sistem vraća aktivacioni ID kako bi potvrdio da je poziv za izvršenje primljen. Ukoliko je došlo do nekih problem, kao što je pad mreže ili neka greška koja se javila pre primanja HTTP zahteva, moguće je da nastupi situacija u kojoj je OpenWhisk primio i obradio zahtev.
- Kada se u sistemu jednom neka akcija pozove na izvršenje, on je zabeleži i njen status čuva u **aktivacionom zapisniku**. Svaki poziv na izvršenje, koji je uspešno primljen ili za koji se korisniku može naplatiti novcem, u nekom trenutku naći će se u aktivacionom zapisniku.
- U slučaju da nastupi greška programera, akcija se možda delimično izvršila i generisala, te je moglo doći do promene. Na samom korisniku je da prati i ispituje da li je konzistentnost sistema narušena delimičnim izvršenjem akcije i ukoliko jeste pozove akciju na ponovno izvršenje. I neke greške u okviru whisk-a mogu sugerisati korisniku da je akcija počela sa izvršenjem, ali je došlo do pada sistema pre kompletnog izvršenja akcije.

AKCIJE – PRIMER IZVRŠENJA

Sadržaj fajla *manifest.yaml*

a)
```
packages:
    default:
        actions:
            helloPython:
                function: demo.py
```

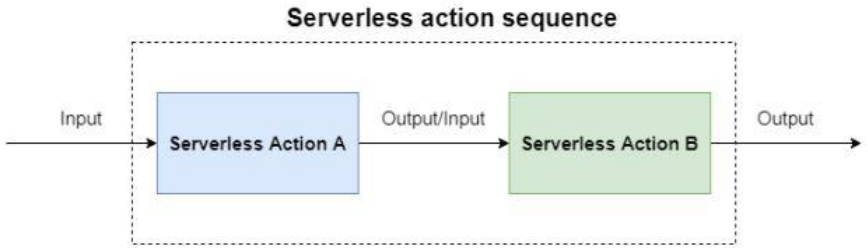
b)
```
packages:
    default:
        actions:
            helloPython:
                code: |
                    import sys
                    def main(args):
                        if 'name' in args:
                            name = args['name']
                        else:
                            name = "stranger"
                        greeting = "Hello " + name + "!"
                        print(greeting)
                        return {"greeting": greeting}
                runtime: python:3
```

```
C:\Users\Tea\Desktop\VIII semestar\SOA\OpenWhisk>wskdeploy -m manifets.yaml
```

Pokretanje izvršenja akcije korišćenjem *wskdeploy*-a

SEKVENCE I KONDUKTORI AKCIJA

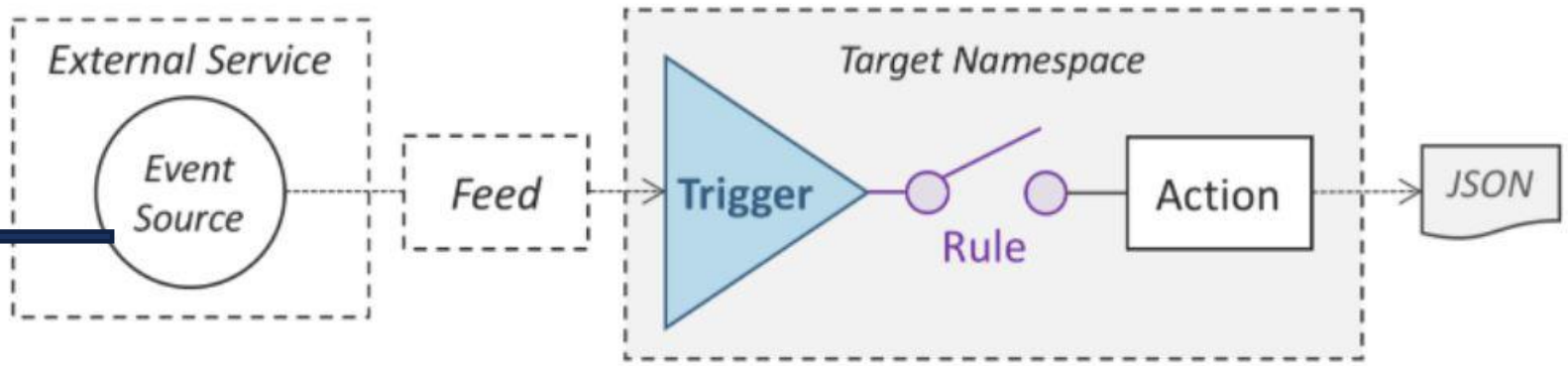
- Veći broj akcija koje mogu biti implementirane u različitim programskim jezicima mogu se kombinovati i pisati zajedno (pipeline) na taj način formirajući sekvencu. Kombinacija akcija očigledno može biti sastavljena iz delova koda napisanih u različitim jezicima što pruža mogućnost odvajanja orkestracije toka podataka među ovim delovima.
- Konduktori, slično kao i sekvence, predstavljaju mogućnost grupisanja većeg broja akcija, koje mogu biti heterogene u pogledu jezika implementacije.
- Razlika između ova dva pojma je što komponente sekvence moraju biti specificirane pre kreiranja same sekvence, dok kod konduktora komponente mogu biti definisane u vreme izvršenja.



TRIGERI I PRAVILA

- Trigeri su imenovani kanali za klase ili vrste događaja koje dolaze iz Event Sourc-a (vidi se na slici ispod).
- Pravila se koriste za povezivanje trigera sa jednom akcijom. Nakon ostvarivanja ove veze, svaki put kada se triger okine, akcija se poziva na izvršavanje.

- Poruke koje stižu na redove poruka
- Promene u bazi podataka
- Promene u bazama dokumenata
- Interakcije između web sajtova/web aplikacija
- Senzori za detekciju i prenos podataka kod IoT uređaja



KREIRANJE TRIGERA

- Primeri trigera:
  1. Triger lokacije ažuriranja događaja
  2. Triger postavljanja dokumenta na web sajt
  3. Triger dobijenog mail-a
- Trigeri mogu biti aktivirani korišćenjem vrednosti iz rečnika (ključ-vrednost par).
- Trigeri se mogu aktivirati eksplicitno (sam korisnik aktivira triger) ili pomoću korisnika preko nekog spoljašnjeg izvora događaja.
- Spoljašnji izvori događaja pobuđuju trigere i to je omogućeno korišćenjem *feed*-a.
- Primeri feed-a:
  1. Svaki put kada se dokument dodaje, odnosno menja u bazi podataka feed za promenu podataka na cloud-u aktivira triger
  2. Feed gita aktivira triger za svaki komit u okviru nekog git repozitorijuma

KORIŠĆENJE PRAVILA

- Korišćenjem odgovarajućeg skupa pravila moguće je jednim trigerom pozvati na izvršenje veći broj akcija, ili se jedna akcija poziva na izvršenje kao posledica na odgovor aktivacije većeg broja trigera.
- Posmatrajmo sistem koji se sastoji iz:
  - klasifikacijaSlika – detektuje objekte na slici i vrši njihovu klasifikaciju
  - naslovnaSlika – kreira se naslovna slika od polazne

  } Akcije

  - noviTvit – aktivira se kada se postavi novi tvit
  - postaviSliku – aktivira se kada se slika postavi na web sajt

  } Trigeri

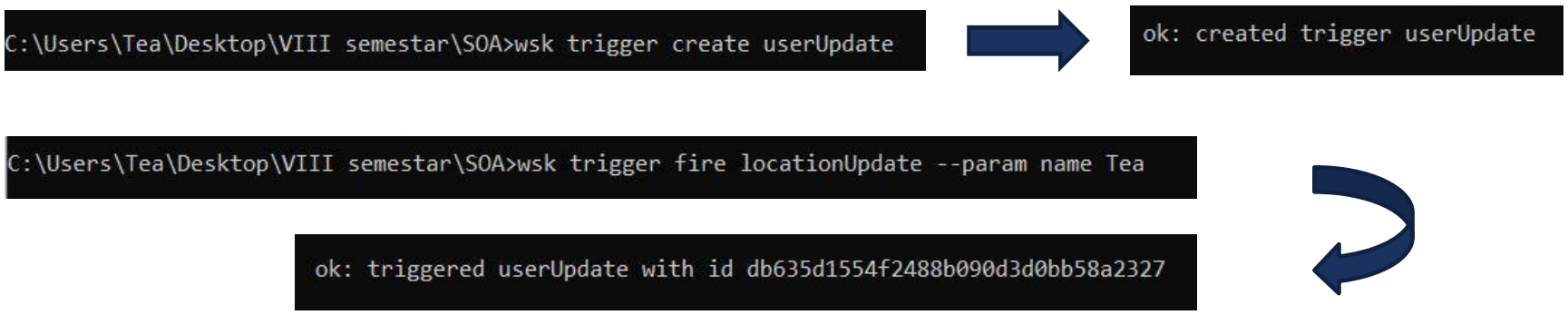
  - noviTvit ⟶ klasifikacijaSlika
  - postaviSliku ⟶ klasifikacijaSlika
  - postaviSliku ⟶ naslovnaSlika

  } Pravila

- Pravila definišu ponašanje sistema: slike u okviru tvita i slika koja je postavljena na web sajt su klasifikovane i izgenerisana je naslovna slika.

TRIGERI – PRIMER

- Kreiraćemo triger koji ažurira ime korisnika i sami ćemo pozvati triger na izvršenje.



- Triger koji je aktiviran bez pravila koje ga definiše nema vidljivog efekta i nakon izvršenja.
- Trigeri se ne mogu kreirati u okviru paketa. Moraju se direktno kreirati u okviru namespace-a.

POVEZIVANJE TRIGERA I AKCIJA KORIŠĆENJEM PRAVILA – PRIMER

- Pravila se koriste za povezivanje odgovarajućih akcija i pravila. Svaki put kada se aktivira triger, akcija se poziva na izvršenje sa parametrima koji se definišu u datom događaju. Postrajmo primer gde će se akcija pozivati uvek kada korisnik ažurira svoje ime:

1) Provera da i akcija i triger koje ćemo koristiti dalje postoje.

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk trigger update userUpdate
```

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk action update helloPython demo.py
```

2) Kreiranje pravila. Pravilo odmah postaje dostupno da odgovori na aktivaciju trigera. Eskplicitno moramo onemogućiti pravilo ako to želimo.

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk rule create myRule userUpdate helloPython
```

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk rule disable myRule
```

3) Uvek kada se desi događaj, akcija se poziva na izvršenje sa dodeljenim parametrima.

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk trigger fire userUpdate --param name Tea
```

```
ok: triggered userUpdate with id 648998285bbc567a8998285bbc967a41
```

PAKETI – KREIRANJE I KORIŠĆENJE

- U OpenWhisk-u skup povezanih akcija se može „skupiti zajedno“ i deliti sa ostalim korisnicima, a sve to je moguće zahvaljujući paketima (*packages*).
- Paketi uključuju akicje i feed-ove.
- Svaki entitet u okviru OpenWhisk-a, uključujući i pakete, pripada nekom prostoru imena (*namespace*). Tako za svaki od entiteta važi konvencija imenovanja: /namespaceName[/packageName]/entityName
- Nekoliko paketa dolazi u okviru OpenWhisk-a. Listu paketa u prostoru imena, listu entita u paketu, kao i opise individualnih entiteta u okviru paketa možemo dobiti izvršavanjem sledeće komande:

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk package list /whisk.system
```

Pretraga paketa u prostoru imena whisk.system

```
packages
/whisk.system/cloudant                shared
/whisk.system/alarms                  shared
/whisk.system/watson                  shared
/whisk.system/websocket               shared
/whisk.system/weather                 shared
/whisk.system/system                  shared
/whisk.system/utils                   shared
/whisk.system/slack                   shared
/whisk.system/samples                 shared
/whisk.system/github                  shared
/whisk.system/pushnotifications       shared
```

KREIRANJE PAKETA

- Paketi se koriste za organizaciju skupova međusobno vezanih akcija i feed-ova. Pored toga pruža mogućnost da su parametri vidljivi svim entitetima u paketu.

1) Kreiranje paketa.

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk package create custom
```

```
ok: created package custom
```

2) Kreirati jednostavnu akciju koja vraća ulazne parametre koji su joj prosleđeni. Ukoliko se akcija kreira u paketu onda se imenu akcije dodaje prefiks imena paketa u kojem se ona kreira. Nije dozvoljeno ugnježdavanje paketa. Paket ne sme da poseduje druge pakete.

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk action create custom/identity identity.py
```

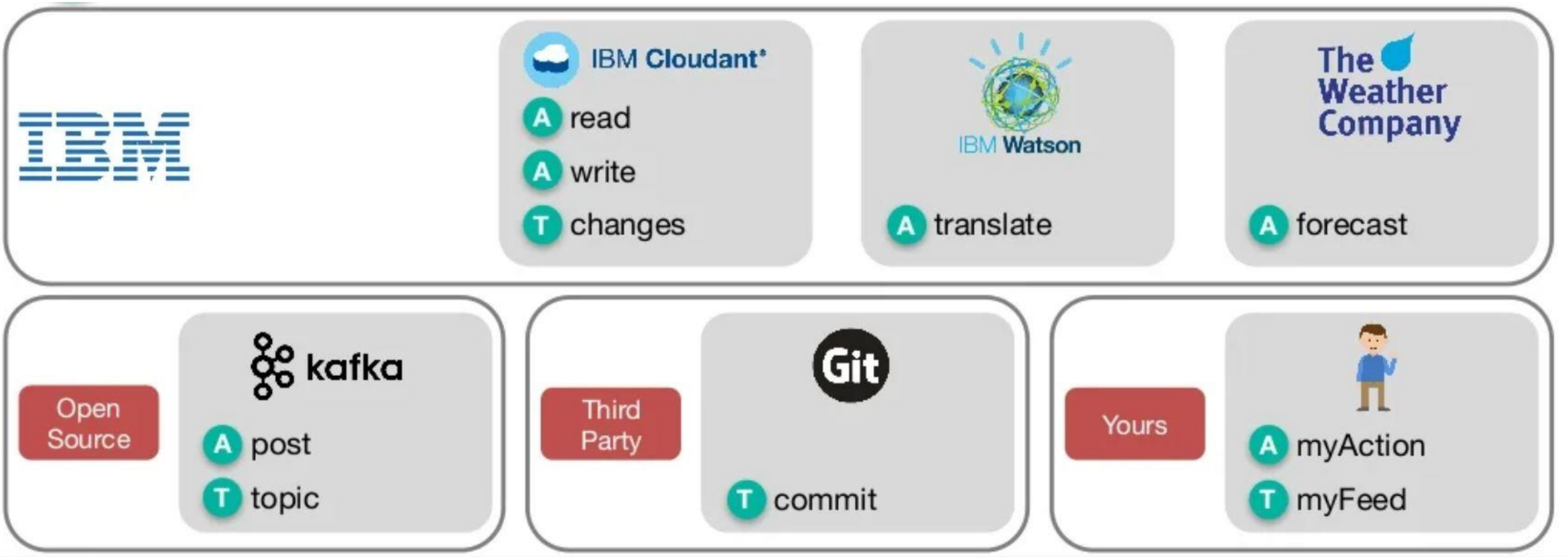
```
ok: created action custom/identity
```

3) Dodavanje kreirane akcije paketu.

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk action invoke --result custom/identity
```

```
{}
```

- Korisnik može postaviti inicijalne parametre za sve entitete u paketu. Ovo se postiže postavljanjem tzv. *package-level* parametara koje preuziama svaka akcija iz paketa.



DELJENJE PAKETA

Nakon dodavanja svih željenih akcija i feed-ova u paket, vrši se debug-ovanje i testiranje paketa. Ukoliko nema grešaka takav paket može da se podeli i postane javan za sve korisnike OpenWhisk-a. Deljenjem paketa ostali korisnici moći će da ga koriste dodavanjem akcija, pravila ili kreiranjem novih sekvenci.

DELJENJE PAKETA – PRIMER

1) Deljenje paketa sa svim korisnicima.

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk package update custom --shared yes
```

```
ok: updated package custom
```

2) Provera da li je paket uspešno publikovan na OpenWhisk platformi.

```
C:\Users\Tea\Desktop\VIII semestar\SOA>wsk package get custom
```

```
ok: got package custom
...
"publish": true,
...
```

- Ostali korisnici sada mogu koristiti paket *custom*, uključujući i mogućnost izmene samog paketa dodavanjem akcija, itd. Da bi izmenili paket moraju znati puno ime paketa. Sve pojedinačne akcije i feed-ovi u ovom paketu takođe su javni. Ukoliko je paket privatni, to će važiti i za čitav sadržaj tog paketa.

https://github.com/apache/openwhisk/blob/master/docs/reference.md#openwhisk-entities

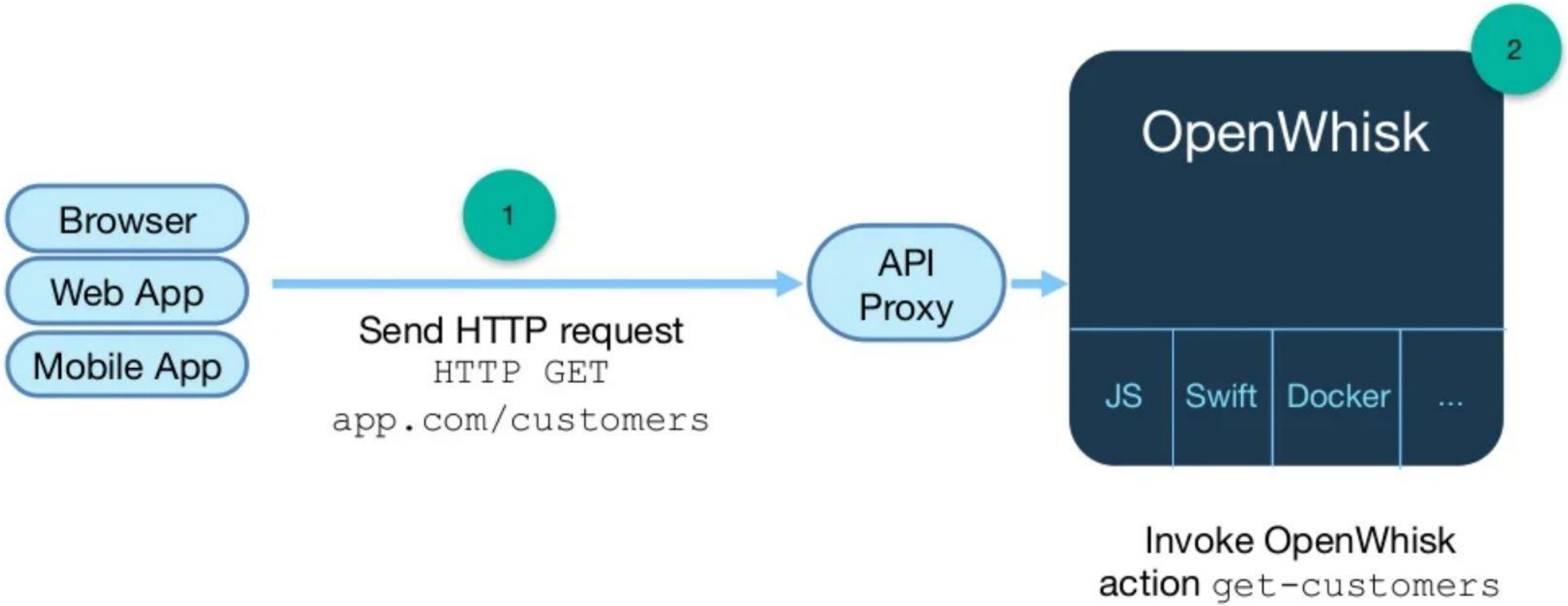
REST API I OPENWHISK

- Nakon što se postavi OpenWhisk okruženje ono se može iskoristiti za API pozive u okviru mobinih ili web aplikacija.
- Sve karakteristike sistema dostupne su preko REST API-a. Postoji kolekcija endpoint-a za akcije, trigere, pravila, pakete i prostore imena.

  - https://$APIHOST/api/v1/namespaces
  - https://$APIHOST/api/v1/namespaces/{namespace}/actions
  - https://$APIHOST/api/v1/namespaces/{namespace}/triggers
  - https://$APIHOST/api/v1/namespaces/{namespace}/rules
  - https://$APIHOST/api/v1/namespaces/{namespace}/packages
  - https://$APIHOST/api/v1/namespaces/{namespace}/activations
  - https://$APIHOST/api/v1/namespaces/{namespace}/limits

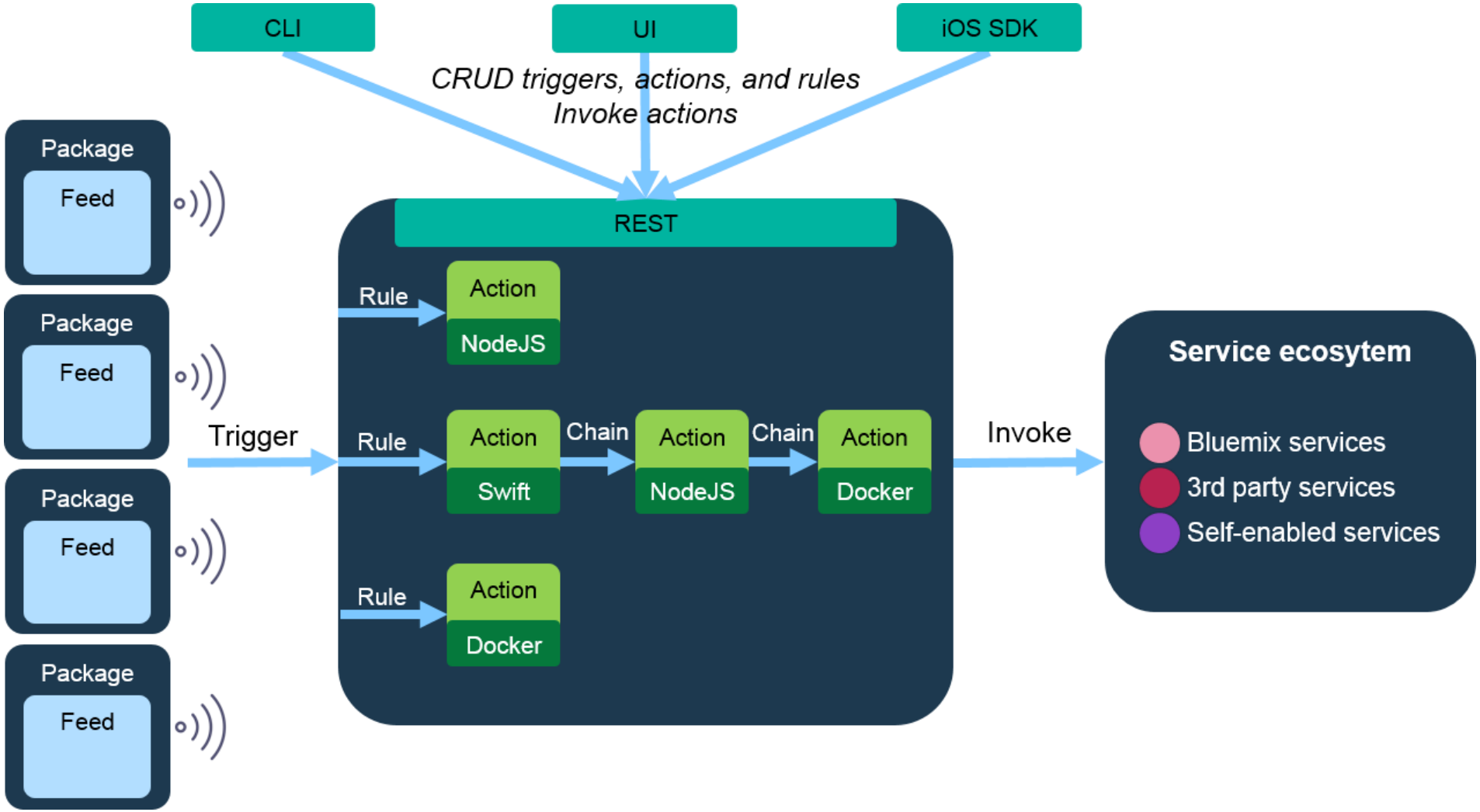
Kolekcija endpoint-ova

REST MIKROSERVIS



REST API I OPENWHISK

- Prostori imena kao i aktivacioni endpoint-i podržavaju jedino GET zahteve.
- Endpoint-i akcija, trigera, pravila i paketa podržavaju: GET, PUT i DELETE zahteve.
- Važi da endpoint-i akcija, trigera i pravila podržavaju i POST zahteve, koji se koriste da aktiviraju akcije i trigere ili da omoguće, odnosno neomoguće pravila.
- Svaki API je zaštićen *HTTP Basic* autentifikacijom. Ukoliko korisnik želi da generiše novi prostor imena i autentifikaciju to može učiniti koristeći se *wskadmin* alatom.
- OpenWhisk API podržava „zahtev-odgovor“ pozive od strane web klijenata. OpenWhisk odgovara na *OPTIONS* zahteve hederima **Cross-Origin Resource Sharing**.
- Trenutno su dostupan svaki origin (Access-Control-Allow-Origin je „*“), kao i standardni skup metoda (Access-Control-Allow-Methods je „GET, POST, PUT, DELETE, HEAD“) i Access-Control-Allow-Headers podržava „Authorization, Origin, X-Requested-With, Content-Type, Accept, User-Agent„.
- OpenWhisk za sada podržava samo jedan ključ po prostoru imena, te se ne preporučuje korišćenje CORS-a za jednostavnije projekte. U tom slučaju se koriste *API Gateway* ili *Web Actions*.



Arhitektura OpenWhisk platforme

ARHITEKTURA I SISTEM RADA

- Kretanje događaji koji dolaze iz spoljašnjih i unutrašnjih izvora događaja kontrolisano je trigerima, dok pravila omogućavaju akcijama da na odgovarajući način reaguju na te događaje.
- Akcije u OpenWhisk-u se izvršavaju u trenutku aktiviranja trigera. Što se više trigera aktivira, više akcija se poziva na izvršenje. Ako nema aktiviranja trigera, kod koji predstavlja akciju se ne izvršava, pa samim tim nema troškova.
- Akciju za odgovarajući triger možemo povezati korišćenjem OpenWhisk CLI, API ili iOS SDK. Kombinovanje više akcija se postiže korišćenjem sekvenci, nema potrebe to definisati u samom kodu.
- Kako se akcije izvršavaju jedino kada se triger aktivira, OpenWhisk postiže optimizaciju, skalabilnost i efikasno korišćenje.
- Dodatni servisi i izvori događaja se dodaju preko paketa. Postojeći skup paketa čini stvaranje aplikacija lakšim, jerti paketi sadrže korisne mogućnosti i pristup spoljašnjim servisima u ekosistemu (Cloudant, The Weather Company, Slack and GitHub).
- Zaključujemo da arhitektura OpenWhisk platforme nije nimalo jednostavna.

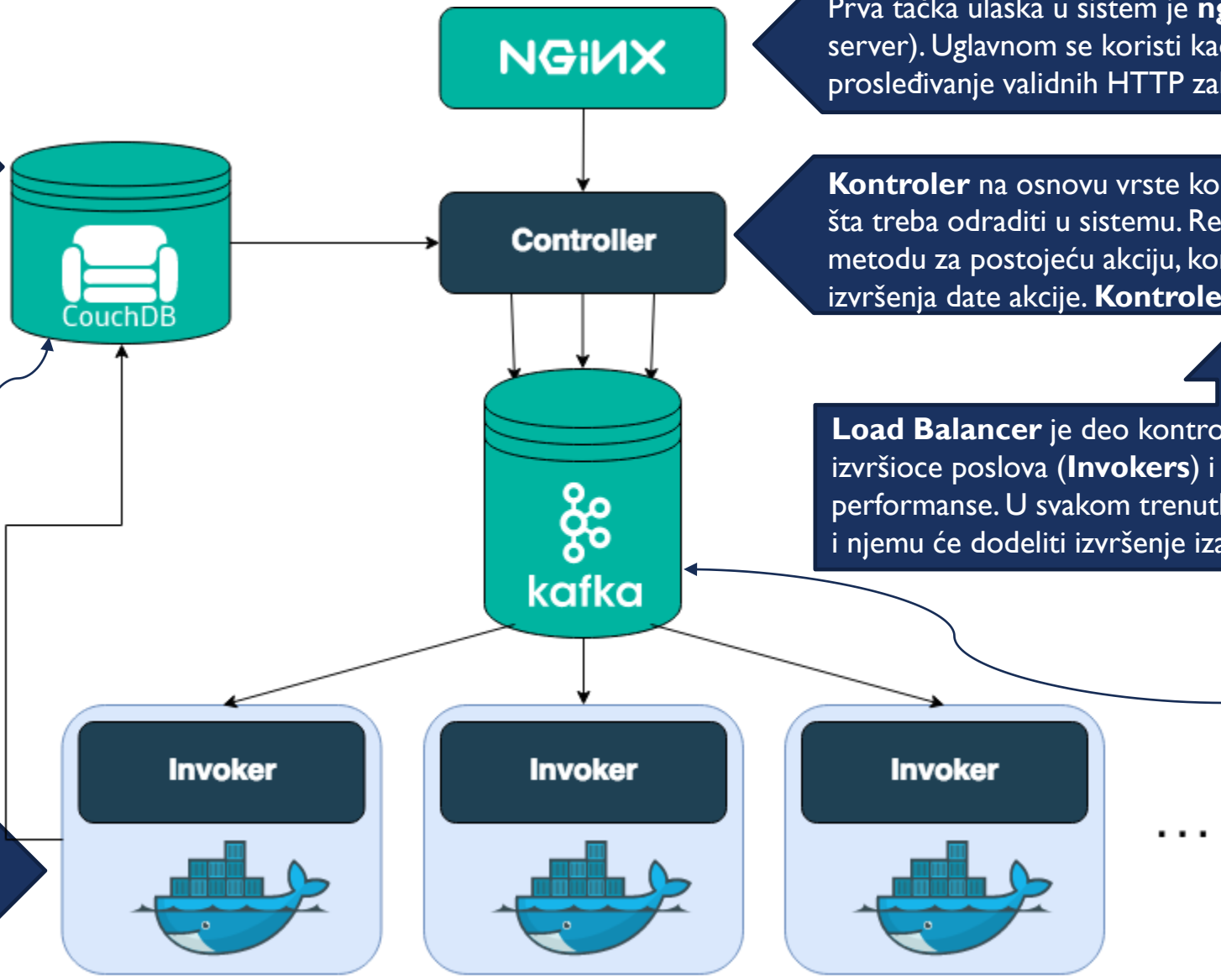
Nakon što kontroler izvrši autentifikaciju korisnika, podaci korisnika se verifikuju u bazi podataka subjekata u instanci **CouchDB**-a. Nakon ovih podataka i odabrana akcija se učitava u **CouchDB**.

Prva tačka ulaska u sistem je **nginx** (HTTP i suprotan proxy server). Uglavnom se koristi kao krajnja tačka kod SSL-a ili za prosleđivanje validnih HTTP zahteva narednoj komponenti.

Kontroler na osnovu vrste korisničkog HTTP zahteva saznaje šta treba odraditi u sistemu. Recimo da korisnik koristi POST metodu za postojeću akciju, kontroler to prevodi na poziv izvršenja date akcije. **Kontroler** je centralna tačka sistema.

Dobijeni rezultat od strane Invoker-a se smešta u bazu aktivacija. Ova baza podataka „živi“ u CouchDB-u.

Load Balancer je deo kontrolera koji ima globalni pogled na izvršioce poslova (**Invokers**) i u kontinuitetu prati njihovo stanje i performanse. U svakom trenutku zna koji od izvršioca je slobodan i njemu će dodeliti izvršenje izabrane akcije.



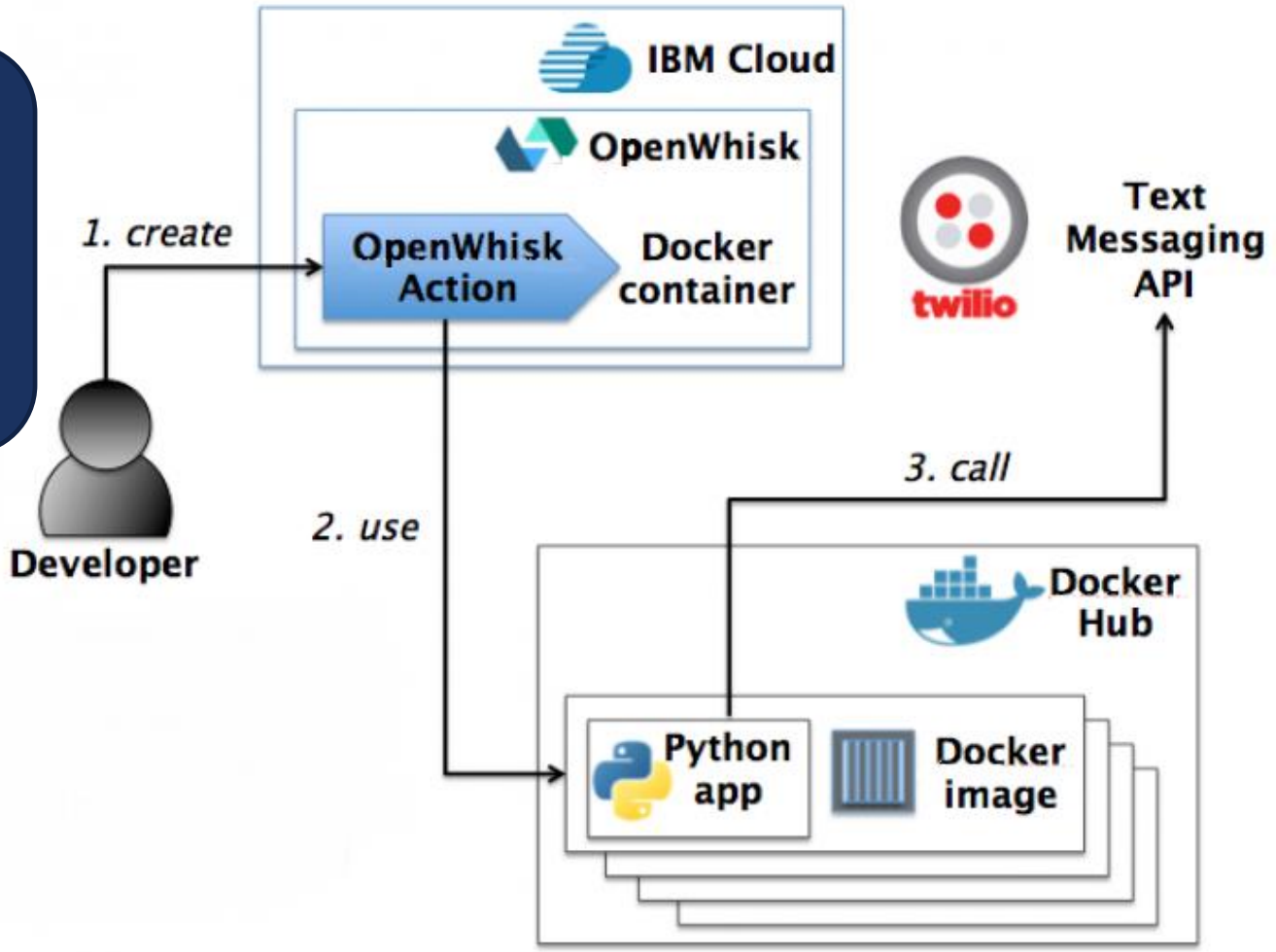
Kafka je visoko propusni, distribuirani publish-subscribe sistem. Predstavlja bafer za sve poruke u sistemu, tako da postoji rizik od nedostatka memorijskog prostora. Takođe vodi računa o tome da se pojedine poruke ne izgube u slučaju da dođe do pada sistema.

Invoker predstavlja srce OpenWhisk-a. Glavni zadatak ove komponente je da pozove akciju na izvršenje. Implementiran je u *Scala*-i. Najbolji način da se akcija izvršava na izolovani i sigurni način je korišćenjem *Docker*-a.

Internal flow OpenWhisk platforme

ARHITEKTURA – PRIMER

Korisnik kreira akciju OpenWhisk-a na IBM Cloud-u za pokretanje doker kontejnera koristeću sliku sa Docker Hub-a onda kada se aplikacija pozove na izvršenje. Aplikacija potom upućuje poziv Twillio API-u i na kao rezultat na korisnikov telefon stiže SMS.



KOD AKCIJE – PRIMER

- Kod za OpenWhisk akciju nalazi se u pajton fajlu. Postoje dve funkcije, *init* i *run* koje odgovaraju rutama aplikacije u Flask-u /init i /run.
- Za slanje poruke koristi se se OpenWhisk CLI kako bi se prosledili parametri SMS-a.

```
5
6   @app.route('/init', methods = ['POST'])
7   def init():
8       try:
9           json = request.get_json()
10
11          response = [
12              {'success': 'true'}
13          ]
14          return jsonify(status=response), 200
15
16      except Exception as e:
17          err = "Error: " + str(e)
18
19          response = [
20              {'success': 'false'},
21              {'msg': err}
22          ]
23
24          return jsonify(status=response), 500
```

Funkcija *init* se poziva na HTTP POST zahtev i OpenWhisk platforma vraća HTTP 200 status kod ako je sve u redu, u suprotnom će izbaciti poruku o grešci koja se javila.

Funkcija *run* proverava da je dolzeći HTTP POST zahtev JSON dokument, koji sadrži konfiguracione parametre za Twillio nalog (twillio sid, twillio authentification token and twillio phone number), kao i tekst poruke koja se šalje korisniku. Nakon konfiguracije parametara za Twillio klijenta i slanja poruke funkcija vraća HTTP 200 status kod, kao i JSON dokument koji potvrđuje da je akcij uspešno obavljena.

```
26  @app.route('/run', methods = ['POST'])
27  def run():
28      try:
29          json = request.get_json()
30
31          assert json
32          assert "value" in json, "Missing value key in the input JSON"
33          assert "account_sid" in json["value"], "Missing Twilio account SID"
34          assert "auth_token" in json["value"], "Missing Twilio Authentication Token"
35
36          account_sid = json["value"]["account_sid"]
37          auth_token  = json["value"]["auth_token"]
38
39          client = TwilioRestClient(account_sid, auth_token)
40
41          assert "from" in json["value"], "Missing from phone number"
42          assert "to" in json["value"], "Missing to phone number"
43
44          assert "msg" in json["value"], "Missing the body of the message"
45
46          fromNumber = json["value"]["from"]
47          toNumber = json["value"]["to"]
48          msg = json["value"]["msg"]
49
50          message = client.messages.create(body=msg,
51              to=toNumber,
52              from_=fromNumber)
53
54          assert message.sid, "Unable to retrieve Twilio SMS message SID"
```

KREIRANJE I TESTIRANJE AKCIJE – PRIMER

- Najpre se kreira akcija koristeći sliku iz Docker Hub-a, a potom se akciji dodaju parametri vezani za Twillio nalog sa kojeg će biti poslat SMS.

1) wsk action create --docker textAction tea2904/openwhisk
2) wsk action update textAction --param account_sid "ACfb27ab6cb8399b2b29969a53ba3ac9cc" --param auth_token "722ecfc35ffd9f0b0279b983fd03eae2"

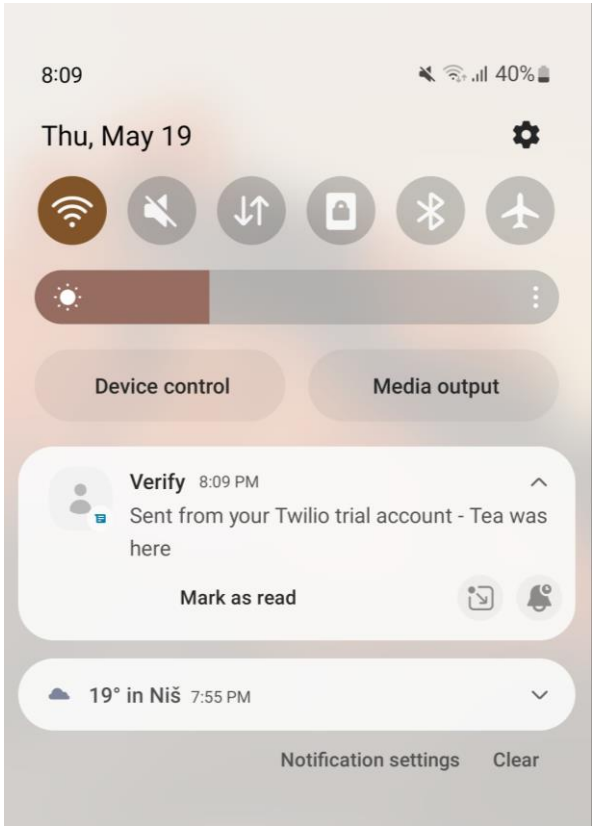
- Na kraju da bismo poslali poruku izvršiti sledeću komandu:

1) wsk action invoke --blocking --result -p from "+17655607120" -p to „+381641119359" -p msg "Tea was here" textAction

KREIRANJE I TESTIRANJE AKCIJE – PRIMER

- Ukoliko je uspešno izvršena akcija odgovor trebalo bi dobiti odgovor sličan ovome:


```
{
  "status": [
    {
      "success": "true"
    },
    {
      "message_sid": "SMf2ccc98d98ab47ed84ba52c7bf5f6671"
    }
  ]
}
```

Primljenja poruka na telefonu

LITERATURA

1. Raymond Camden, " Developing Serverless Applications – A Practical Introduction with Apache OpenWhisk" – O'Reilly
2. Apache OpenWhisk Documentation, https://openwhisk.apache.org/documentation.html
3. Apache GitHub account, https://github.com/apache/openwhisk