Microservices (Google Cloud Platform)

Materijali

Osnovna ideja mikroservisa i usluga koje nude mikroservisne platforme (AWS, Azure, Google Cloud Platform) jeste razbijanje monolitne aplikacije na vise komponenti (mikroservisa), od kojih svaka može biti implementirana u proizvoljnoj tehnologiji, od strane sasvim drugog tima developera.

Razlozi mogu biti brojni:

1. drugačiji release-schedule komponenti
2. različite komponente razvijaju različiti timovi
3. komponente koriste drugačiji technology-stack
4. različite potrebe za skaliranjem komponenti
5. itd...

Deploy proces za svaku komponentu pravi dodatni overhead. Dobar način za rešavanje toga jeste serverless pristup. Serverless ne podrazumeva pristup "bez servera", već pristup kod koga se činjenica da na hiljade servera trče u pozadini apstrahuje, što znači da nije neophodno da toga sve vreme budemo svesni.

- nema potrebe za:
  - upravljanjem infrastrukturom (runtime, OS patching, itd...)
  - plaćanjem neiskorišćenog kapaciteta na cloud-u
  - rizikovanjem nedostupnosti servisa u periodu visokog opterećenja
- jedino na šta treba obratiti pažnju jeste pisanje koda
- sve što treba uraditi je deployment servisa
- servis se automatski skalira, njime upravlja "neko drugi" (Google)

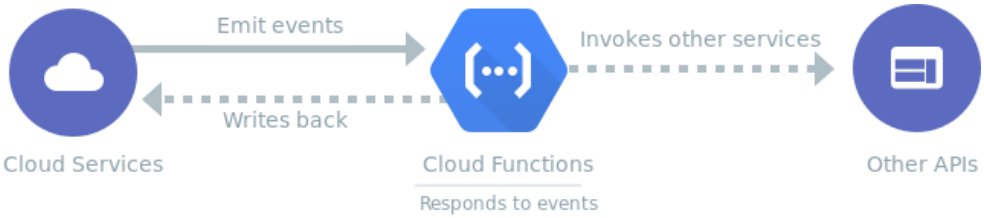
Google Cloud Platform

Slika 1: Google Cloud Platform kao mikroservisna platforma

Google Cloud Platform predstavlja skup cloud servisa, koji pružaju niz modularnih usluga poput remote funkcija, skladištenja podataka, analitike podataka, mašinskog učenja; kao i skup alata za njihovo upravljanje.

Google Cloud Functions (event-driven serverless compute platform) je serverless FaaS (Function as a service) platforma, u okviru Google Cloud sistema, za izgradnju mikroservisa zasnovanih na događajima. Google Cloud funkcije se pišu kao JavaScript komponente, koje se izvršavaju u Node.js runtime-u, uz pomoć Google Cloud platforme. Iako su JavaScript i Node.js (server-side JavaScript runtime environment) ponuđeni kao inicijalno rešenje, Google Cloud pruža podršku i za druge programske jezike. Pored toga, NPM (Node.js Package Manager) ima prilično opsežnu biblioteku modula koji se mogu uključiti u aplikaciju radi korišćenja već gotovh, "out of the box" funkcionalnosti.

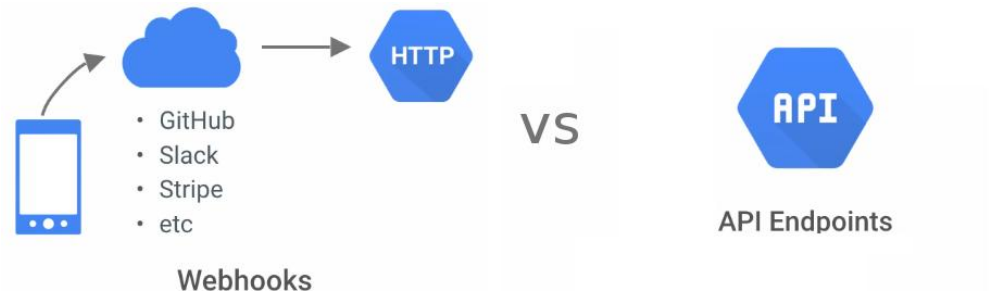
Tipovi Cloud funkcija



Slika 2: Google Cloud funkcije

Google Cloud funkcije su event-driven, a dostupna su dva tipa funkcija, koje se razlikuju po mehanizmu korišćenom za pokretanje njihovog izvršavanja.

1. HTTP endpoints/functions - funckije čije se izvršavanje pokreće HTTP zahtevima, najčešće se koriste za:
    a. implementaciju API-ja
    b. implementaciju HTTP callback-a (tzv. "webhooks", pored standardih trigger-a, mogu da handle-uju niz događaja poput file uplod-a na GoogleDrive, pristigle pošta u GMail-u, događaje u GoogleCalendar-u, plaćanje platnim karticama vezanim za Google nalog, slanje SMS poruke, itd)



Slika 3: Primer upotrebe HTTP funkcija

2. Backgroud functions - tzv. pozadinske funkcije koje osluškuju događaje na cloud-u
    a. Cloud Storage događaji
    b. Cloud Pub/Sub događaji

Primer HTTP funkcije

S obzirom na činjenicu da cloud funkcije apstrahuju infrastrukturu, nije potrebno pisati mnogo koda, niti konfiguracije okruženja za potrebe pokretanja jednostavnog mikroservisa. Cloud funkcija automatski parsira telo dolazećeg HTTP zahteva (npr. JSON objekat).

```
exports.helloHttp = function helloHttp(request, response) {
      let message = 'Hello, ' + request.body.name;
      response.status(200).send(message);
}
```

Za pokretanje izvršavanja prethodno definisane funkcije, dovoljno ju je deploy-ovati i uputiti HTTP POST zahtev na adresu HTTP endpoint-a, sa sledećim sadržajem:

```
{ "name":  "Pera Perić"}
```

Request i response objekti su strogo tipizirani (makar u meri stroge tipiziranosti koja se može ostvariti u JavaScript-u), što znači da je moguć direktan pristup URL parametrima, parametrima forme, parametrima iz HTTP body-ja, JSON zahteva kao direktnim property-jima request objekta:

- req.baseUrl - URL putanja router-a
- req.body - key-value parovi iz tela HTTP zahteva
- req.cookies - objekat koji sadrži cookie-je iz HTTP zahteva
- req.params - parametri putanje
- req.path - path deo URL-a upućenog zahteva
- req.query - objekat koji sadrži mapu svih query string parametara
- itd...

Na sličan način je moguće pristupiti svojstvima response objekta sa ciljem postavljanja tela HTTP odgovora, status koda, i sl.

Upravljanje dependency-jima

Upravljanje dependency-jima u Cloud funkcijama je isto kao i u bilo kojoj drugoj Node.js aplikaciji. Za kreiranje package.json fajla potrebno je inicijalizovati Node.js projekat u tekućem direktorijumu, izvršavanjem sledeće komande:

```
$ npm init

name: (helloHttp)
version: 0.0.1
```

```
...
author: Igor Cverdelj-Fogarasi
licence: (ISC) Apache-2.0
About to write package.json:

Is this ok? (yes)
```

Rezultat izvršavanja prethodne komande je sledeći package.json fajl:

package.json

```
{
  "name": "sort",
  "version": "0.0.1",
  "description": "Here goes short package description.",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Igor Cverdelj-Fogarasi",
  "license": "Apache-2.0",
  "dependencies": {
    "@google/cloud-debug": "^0.9.3"
  }
}
```

U prethodnom primeru dodata je referenca ka Stackdriver Debugger-u, Node.js modulu koji je moguće instalirati lokalno i uključiti u package.json deklaraciju na sledeći način:

```
$ npm install @google/cloud-debug --save
```

Prethodna komanda će referencirani Node.js modul instalirati lokalno (u tekućem direktorijumu), dok će --save automatski dodati dependency u package.json. Kako bismo prethodni modul referencirali iz naše cloud funkcije dovoljno ga je uključiti na sledeći način:

index.js

```
require('@google/cloud-debug');
exports.helloHttp = function helloHttp(req, res) {

  let name = req.query.name || req.body.name;

  if (name === undefined) {
    console.warn('Bad request: No name is provided.');
    res.status(400).send('Name is not defined!');

  } else {
    console.log('Sending a greeting to: ' + name);
    res.status(200).send('Hello ' + name + '!');
```

```
  }
};
```

Deployment funkcije na cloud

Cloud funkcije se mogu pisati direktno, u okviru Cloud Console web interfejsa, deploy-ovati sa git repozitorijuma ili lokalne radne stanice. Pored deployment-a mikroservisa na cloud (iz Google Cloud SDK-a, ili Cloud Shell web interfejsa), na raspolaganju je i open-source emulator (Google Cloud Functions Emulator), Node.js aplikacija koja implementira Google Cloud Functions API, a uključuje i CLI (command line interface) za upravljanje aplikacijom.

Iz Google Cloud SDK-a:

- Instalacija Cloud SDK-a[1]
- Pokretanje SDK Shell-a i inicijalizacija
  `$ gcloud init`
- Deployment funkcije (trigger HTTP omogućava testiranje funkcije "iz browsera")
  `$ gcloud functions deploy helloHttp --trigger-http`

ili iz emulatora:

- Instalacija emulatora[2]
  `$ npm install -g @google-cloud/functions-emulator`
- Inicijalizacija emulatora
- `$ functions config set projectId cloud-demo`
- Pokretanje emulatora
- `$ functions start`
- Lokalno deploy-ovanje funcije
  `$ functions deploy helloHttp --trigger-http`

Kada smo se uverili da funkcija radi kako treba, možemo je deployovati na Google Cloud platformu. Prilikom deploymenta cloud funkcije dobijamo sledeće:

- HTTP endpoint sa kvalifikovanim domenom
- dinamički generisani SSL/TLS sertifikat (sav saobraćaj možemo slati kroz HTTPS)
- automatsko skaliranje (serverless)
- ne trebamo da brinemo o infrastrukturi (serverless)

Testiranje HTTP funkcija

[1] Google Cloud SDK, https://cloud.google.com/sdk/downloads
[2] Google Cloud Functions Emulator, https://github.com/GoogleCloudPlatform/cloud-functions-emulator

Testiranje HTTP endpointa se svodi na izvršavanje HTTP zahteva upućenih na adresu na kojoj je deploy-ovana cloud funkcija. Uglavnom je to URL definisan po sledećem šablonu:

`http://{host}:{supervisorPort}/{projectId}/{region}/{functionName}` tj.
`http://localhost:8010/cloud-demo/us-central1/helloHttp`

Za više detalja o konfiguraciji emulatora izvršiti sledeću komandu:
`$ functions config list`

Cloud Functions emulator vodi detaljan log aktivnosti, kome se može pristupiti sledećom komandom:
`$ functions logs read --min-log-level=info`

Za prikaz pune putanje do log fajla izvršiti sledeću komandu iz CLI:
`$ functions status`

Primer Backgroud funkcija

Backgroud funkcije predstavljaju pozadinske funkcije koje osluškuju događaje na cloud-u. Događaji koje ove funkcije obrađuju mogu biti:

- Cloud Storage događaji
- Cloud Pub/Sub događaji

Cloud Storage događaji

Deployment je vrlo sličan deployment-u HTTP triggerovanih funkcija.

```
$ gcloud functions deploy filewatch --stage-bucket functions-src --trigger-bucket watch-bucket
```

Jedina razlika je u trigger bucket-u, koji predstavlja blob storage koji funkcija nadgleda, svaka izmena nad bucketom okida izvršavanje pozadinske funckije.

cloudstorage.js

```
// Function triggered via a file change event
exports.analyzeImage = function(context, data) {
	console.log('Processing image: ' + data.name);
	// Image processing happens here...
	if (hasError) {
		contextFailure('Error when analyzing image: ' + errMsg);
	} else {
		console.log('Image results: ' + results);
		context.success();
	}
};
```

Function signature je drugačiji u odnosu na cloud funkcije tipa HTTP endpoint-a. Parametar context predstavlja objekat koji omogućava posrednika u generisanju odgovora pozadinske funkcije (uspešno ili neuspešno izvršavanje pozadinske funkcije). Parametar data predstavlja ulazne podatke preuzete iz događaja koji je trigger-ovao izvršavanje pozadinske funkcije (npr. file change event from cloud storage).

Cloud Pub/Sub događaji

Predstavljaju Publisher/Subscriber message queuing sistem na Google Cloud platformi.

- predstavlja potpuno upravljiv, skalabilan sistem čekanja poruka
- bilo šta što može da generiše HTTP request, može biti publisher tj. subscriber

Deployment je vrlo sličan prethodnom primeru deployment-a kod Cloud storage event-ova.

```
$ gcloud functions deploy filewatch --stage-bucket functions-src --trigger-topic football-news
```

Mehanizam razmene poruka je vrlo sličan JMS-u (Java Messaging Service). Kada se poruka prosledi u Pub/Sub queue (koji je povezan sa topic-om), svi subscriberi koji su pretplaćeni na dati topic bivaju trigerovani pristiglom porukom. Svaki put kada poruka stigne u referencirani topic, pozadinska cloud funckija se izvršava.

pubsub.js

```
// Function triggered when a Pub/Sub message is published
exports.subscribe = function(context, data) {
	console.log('Stats for ' + data.message.teamName);
	// The rest of the logic is here
	context.success();
};
```

Pub/Sub queue (najcešće JSON objekat) je u potpunosti prilagodljiv kontekstu, bilo šta što treba da se prosledi pozadinskoj funkciji može da se stavi u topic. Data argument za prethodni Pub/Sub event bi mogao da bude formiran kao JSON objekat na sledeći način:

```
{
	"message": {
		"teamName": "Fightin' Footballers",
		"location": "London, UK",
		"currentRank": 1
	}
}
```

Primeri cloud funkcija

1. **helloHttp** - jednostavan primer Google Cloud HTTP endpointa/funkcije
2. **sortContacts** - primer složenijeg HTTP endpoint-a, koji vrši sortiranje prosleđene JSON kolekcije objekata po dinamički zadatom kriterijumu.

Zadatak

U okviru sistema za rezervaciju smeštaja, implementirati podsistem za komentarisanje i ocenjivanje kao mikroservis, uz pomoć HTTP cloud funkcije. Omogućiti korisniku da nakon potvrde boravka i korišćenja smeštaja unese svoj komentar i ocenu. Perzistenciju komentara i ocena implementirati uz oslonac na Google Cloud platformu (cloud storage servise, Cloud SQL ili Cloud Datastore). U okviru podsistema omogućiti i pretraživanje komentara po oceni, kao i preuzimanje prosečne ocene za dati smeštaj.