Correct Zipcodes using Serverless Golang

Max and Sebastian

some day in November 2019

What the heck

Share our passion about Go based Microservices:

▶ implementing a REST service
▶ ... in go
▶ ... using a Google Cloud Function

Show me what you've got

use for example:

```
curl -X POST ... -d '{"zipCode":"72205", "placeName":"Barlin"}'
```

to get:

```
"distance":2,
"percentage":81,
"place":{
  "countryCode":"DE",
  "zipCode":"12205",
  "place":"Berlin",
  ...
  "latitude":"52.434",
  "longitude":"13.2945"
}
```

What's the point?

- What's the fuss about FaaS?
- Support EPLASS address detection.
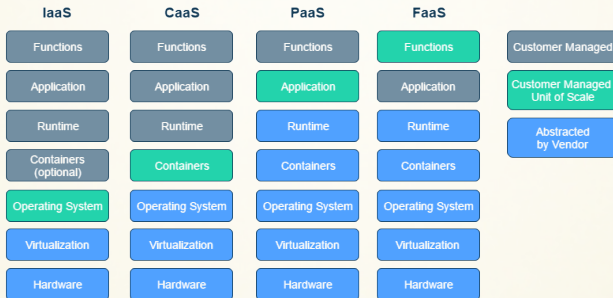- Let's play go.

TOC
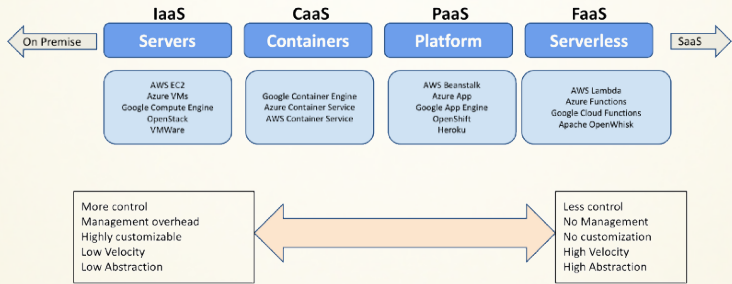
TOC

FaaS

Figure 1: *aaS Stacks

FaaS

Figure 2: *aaS Provider

FaaS here

- AWS Lambda, **Google Cloud Functions**, MS Azure Functions, . . .
- Node.js, Python, **Go 1.11.5**
- Google Cloud Platform, **Google Cloud SDK**, (mirrored) Source Control

TOC

V0 Hello World

v0 implements a service that responds with "Hello World":

- ▶ project layout
- ▶ implementation
- ▶ building and running

V1 Business Logic

v1 extends v0 by zipchecker business logic

- ▶ implementation
    - ▶ request processing
    - ▶ marshaling and unmarshaling
    - ▶ embedding statics
    - ▶ constructors
    - ▶ Levenshtein distance
- ▶ testing

V2 GCP Deployment

$v2$ extends $v1$ by preparing for GCP Deployment

- ▶ GCP preparation
- ▶ GCP deployment

TOC

GCF in Production – Scaling

- ▶ scales by creating new function instances
- ▶ the total number of function instances can be limited
- ▶ function instances are reused
- ▶ global scope may be used to cache across function invocations
- ▶ concurrent requests are processed by different instances
- ▶ response-time depends on hot- or cold start

GCF in Production – Pricing

- ▶ $0.40 / 10^6 invocations
- ▶ $0.0000025 / GB-Second memory
- ▶ $0.0000100 / GHz-Second CPU
- ▶ $0.12 / GB Outbound Data (Egress) traffic

"Free Tier" per month:

- ▶ 2 million invocations
- ▶ 400,000 GB-seconds
- ▶ 200,000 GHz-seconds
- ▶ 5 GB Egress traffic

see: Cloud Functions Pricing

GCF in Production – Price Example

based on 2ms (i.e. 100ms) + 1KB traffic at 128MB and 200MHz:

- ▶ CPU: $\frac{2*10^8 \, \mathrm{MHz\,s}}{0.1s*200 \, \mathrm{MHz}} = 10 * 10^6$

- ▶ memory: $\frac{400{,}000*1024 \, \mathrm{MB\,s}}{0.1s*128 \, \mathrm{MB}} = 32 * 10^6$

- ▶ traffic: $\frac{5*1024*1024 \, \mathrm{KB}}{1 \, \mathrm{KB}} \approx 5.2 * 10^6$

- ▶ invocations: $2 * 10^6$

$\min(10, 32, 5.2, 2) \to 2 * 10^6$ free invocations $\equiv \sim \$4.88$

TOC

Wrapup – Things to do

- trees of functions
- provided services

Wrapup – The next big thing?

- interesting idea
- for small services: easy implementation and deployment

however:

- implementation becomes even more fragmented
- "overly distributed"
- difficult to test
- high delay for logs (sometimes 5-10s)

Wrapup – Readings

- Google Cloud Functions Tutorial Series
- Creating Slack Slash Commands using Go
- this talk, the code . . .

end

```
/whois cool
/whois smart
```