

# Correct Zipcodes using Serverless Golang

Max and Sebastian

Nov 08 2019

**thinkproject**

A decorative graphic on the right side of the slide consisting of several overlapping diagonal bars. From top to bottom, the colors are white, light green, and dark green. The bars are of varying lengths and are positioned to create a sense of depth and movement.

# What the heck

Share our passion about Go based Microservices:

- ▶ implementing a REST service
- ▶ ... in go using golang
- ▶ ... deploying it as a Google Cloud Function

# Show me what you've got

use for example:

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

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 (CoP ML project).
- ▶ Let's play go.

# Why go?

- ▶ Typed Language
- ▶ Minimal and Lightweight
- ▶ No external http server required
- ▶ Comprehensive concurrency and async
- ▶ Easily cross-compiled
- ▶ gofmt formatting

# TOC

1. FaaS
2. V0, V1 and V2 of the zipchecker
3. GCF in Production
4. Wrapup

# TOC

1. **FaaS**
2. V0, V1 and V2 of the zipchecker
3. GCF in Production
4. Wrapup

# FaaS

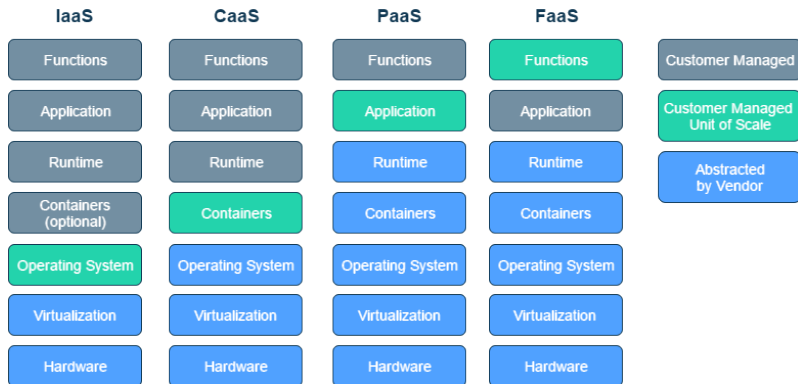


Figure 1: \*aaS Stacks



# FaaS

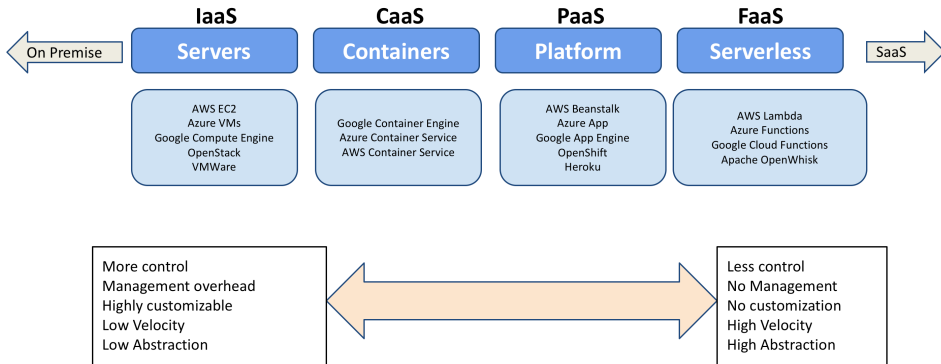


Figure 2: \*aaS Provider

# 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

1. FaaS
2. **V0, V1 and V2 of the zipchecker**
3. GCF in Production
4. Wrapup

# 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 (`net/http`)
  - ▶ marshaling and unmarshaling (JSON and CSV)
  - ▶ embedding statics
  - ▶ constructors
  - ▶ Levenshtein distance
  - ▶ error handling
- ▶ testing

# V2 GCP Deployment

v2 extends v1 by GCP deployment:

- ▶ GCP preparation
- ▶ GCP admin console
- ▶ GCP deployment
- ▶ GCP logs

# TOC

1. FaaS
2. V0, V1 and V2 of the zipchecker
3. **GCF in Production**
4. Wrapup

# GCF in Production – Scaling

- ▶ scales by creating new function instances
- ▶ the total number of function instances can be limited
- ▶ function instances are reused (watch out here)
- ▶ 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:  $\frac{2 \cdot 10^8 \text{ MHz s}}{0.1 \text{ s} \cdot 200 \text{ MHz}} = 10 \cdot 10^6$
- ▶ memory:  $\frac{400,000 \cdot 1024 \text{ MB s}}{0.1 \text{ s} \cdot 128 \text{ MB}} = 32 \cdot 10^6$
- ▶ traffic:  $\frac{5 \cdot 1024 \cdot 1024 \text{ KB}}{1 \text{ KB}} \approx 5.2 \cdot 10^6$
- ▶ invocations:  $2 \cdot 10^6$

$\min(10, 32, 5.2, 2) \rightarrow 2 \cdot 10^6 \text{ free invocations} \equiv \sim \$4.88$

# TOC

1. FaaS
2. V0, V1 and V2 of the zipchecker
3. GCF in Production
4. **Wrapup**

## Wrapup – Things to do

- ▶ trees of functions
- ▶ provided services

## Wrapup – The next big thing?

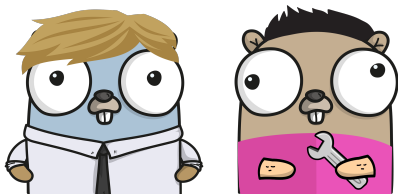
- ▶ interesting idea
- ▶ for small services: easy implementation and deployment

however:

- ▶ implementation becomes even more fragmented
- ▶ “overly distributed”
- ▶ difficult to test
- ▶ high delay for logs (sometimes 5-10s)
- ▶ cost: hard to predict and might get out of hand

# Wrapup – Readings

- ▶ Go Playground
- ▶ Google Cloud Functions Tutorial Series
- ▶ Service Setup with Gin, Auth0
- ▶ Service Monitoring with Go and DataDog



<https://github.com/sebogh/zipchecker>