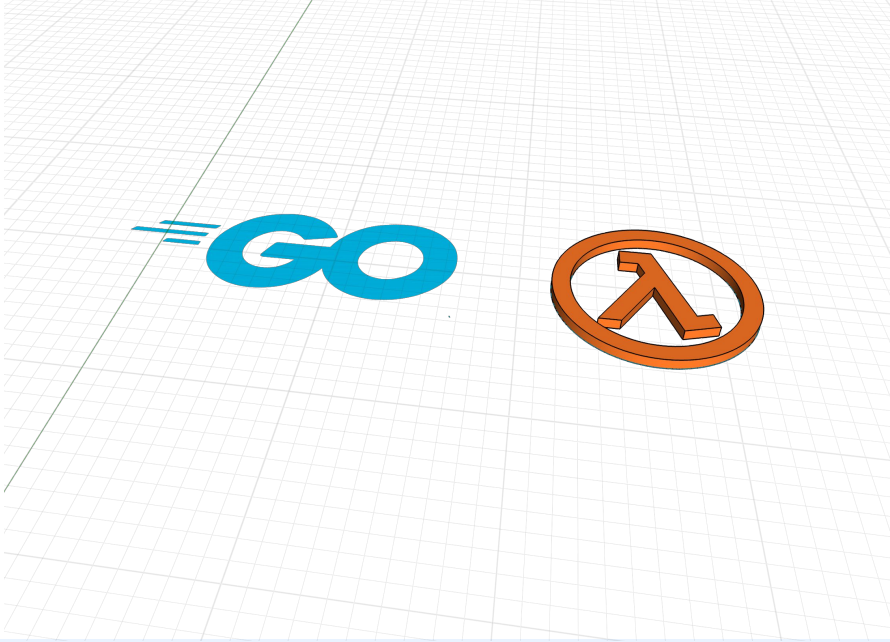
Go Serverless

A Case Study



Richard Szabo

Shapr3D

Intro

- Shapr3D is a 3D modeling app for iPad
- Our team: backend services

  User and Subscription Management

  Analytics / Telemetry

  Collecting App Crash info

  Support for business / financial reports

  3rd party integrations: Marketing tools, Analytical services, Payment systems

  Webhooks, Slackbots, etc.
- AWS, Microservices, Lambda, Go

Shapr3D

Our Stack

- SAM / Cloudformation: AWS resources, infra in code

  API Endpoints, DB, Alerting, Networking, Firewall, almost everything
- Dockerized local dev env + CI/CD
- Alerting, Monitoring

  CloudWatch, PagerDuty
- Lambda functions written in Go
- + Scripts written in Go

  We can reuse already existing business logic in scripts

Shapr3D

Why Serverless?

- Low upfront + maintenance costs in DevOps

  Team started in small, no need for dedicated DevOps / Infra team

  No need to keep servers / clusters up-to-date, monitor disk usage, etc.
- Pay as you go
- Scales automatically up/down based on load

  User Logins, Payments doesn't happen with a constant load,

  but we can handle spikes as well (but be aware of non-serverless resources!)

Why Go?

- Low memory footprint
- Fast startup

  Reduces cold-start time in lambda (No VM like Node.js, Java, ...)
- These affect costs as well

  Execution time, GB-second
- Static typing

  Huge help in terms of maintenance, refactoring
- Simplicity

  Quick onboarding for new members

Why Go? / 2

- Explicit

  Clear is better than clever.
- Fast compilation
- Easy Cross compilation

  Build scripts locally for execution on server
- Built-in test framework

  Also has great support for debugging e.g. in VS Code or GoLand

Why NOT Serverless?

Serverless caveats, pitfalls, good to know

- **Execution on single core**

  We get less than 1 vCPU, cannot really use the full potential of Go in this regard

- **Timeout**

  Max. 15 minutes, not a good fit for long running tasks (e.g. mesh conversion);

  There are alternative serverless solutions of course (e.g. AWS Fargate)

- **Cold start**

  Login or other requests may be slow, however currently doesn't affect UX

Why NOT Serverless? /2

Serverless caveats, pitfalls, good to know

- **Stateless, but somewhat stateful**

  Lambda container might keep running after single exec

  Try to keep resources (e.g. DB connection, authorization token)

- **One request – One lambda instance**

  This might cause too many DB connections when not handled properly

Why NOT Go?

Go caveats, pitfalls, good to know

- Zero values

  E.g. can cause problems in integration with 3rd party services, when converting structs to JSON
- Verbosity
- Consts

  Only simple values can be consts + Consts cannot be addressed
- Error handling

  Repetitive + stack trace needs to be added manually
- No generics :(

  You have to use sometime interface{} or reimplement algorithms for multiple types
- Immature(-ish) community / ecosystem

Shapr3D

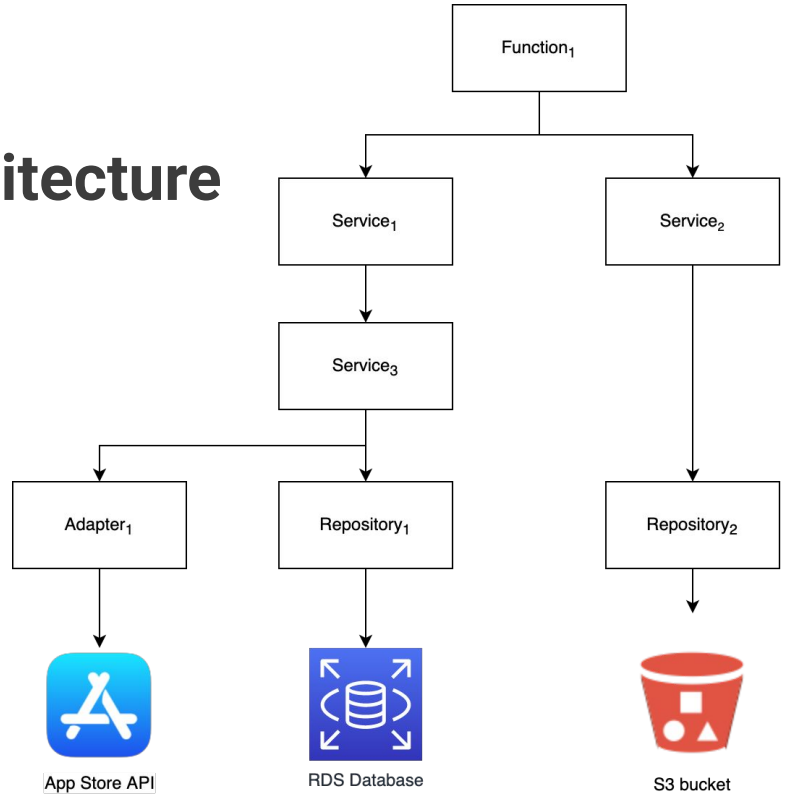
Application architecture

- functions

  This is the only one Lambda specific part!
- services
- repositories

  Raw SQL queries (prepared statements)
- adapters

  Integration with 3rd party services



Shapr3D

Example: User Management System

- 33 functions

  login, logout, register, App Store Webhook, subscription updater job, etc
- 50+ services

  App Store Reporting, User Registration, EDU Request Registration, Analytics, etc.
- 18 repositories

  Users, Subscriptions, Payment Transactions, FX rates, S3 files, etc
- 20 adapters

  App Store receipt verification, Zendesk integration, Sendgrid integration
- + scripts

  E.g. different kind of backfill scripts for analytics

Shapr3D

Q&A

Shapr3D