Microservices in Go

Go-Micro vs Go-Kit vs Gizmo vs Kite

Presented by:
Anton Klimenko antklim@gmail.com

Microservice definitions

Microservice is an architectural style that structures an application as a collection of loosely coupled services, which implement business capabilities.

Microservice properties

- Microservices: software that fits in your head. - Dan North
- A microservice implements a single Bounded Context (from DDD). - Martin Fowler, Sam Newman
- Microservices built & deployed independently. Stateless, with state as backing services - 12Factor.net
- Addressable through a service discovery system. - Chris Richardson

Why microservices?

- Smaller simpler applications
- Easier to scale development and deploy new versions of services frequently
- Easier scaling
- Improved fault tolerance and isolation
- Eliminates long-term commitment to a single technology stack

Why not microservices?

- Inter-process communication
- Handling partial failures
- Distributed transactions
- Cross service integration testing
- Managing the development and deployment of features that span multiple services

Why microservices in Go?

- Statically typed
- Fast compiled
- Native binaries
- Garbage collected
- Big standard library
- Baked-in concurrency
- Native networking
- Efficient by default
- Predictable runtime behavior, fast lifecycles

Popular Go Microservice Frameworks

- Go Micro
- Go Kit
- Gizmo
- Kite

Go Micro

Go Micro is a pluggable RPC framework for writing microservices in Go.

Features:

- Service Discovery
- Load Balancing
- Sync & async Communication
- Message Encoding
- RPC Client/Server

Go Micro architecture


```
import "github.com/micro/go-micro"

service := micro.NewService(
   micro.Name("com.example.srv.service"),
   micro.Version("1.0.0"),
)

service.Init()

service.Run()
```

Go Kit

Go Kit is a programming toolkit for building microservices in Go.

Go Kit is a library, designed to be imported into a binary package.

Go Kit (continue)

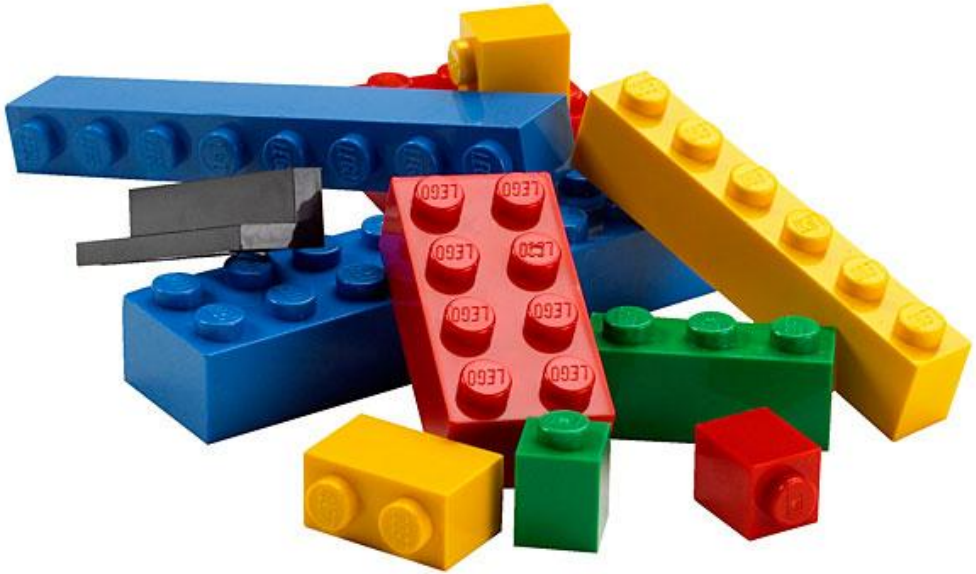
Go Kit philosophy:

- No global state
- Declarative composition
- Explicit dependencies
- Interfaces as contracts
- Domain Driven Design

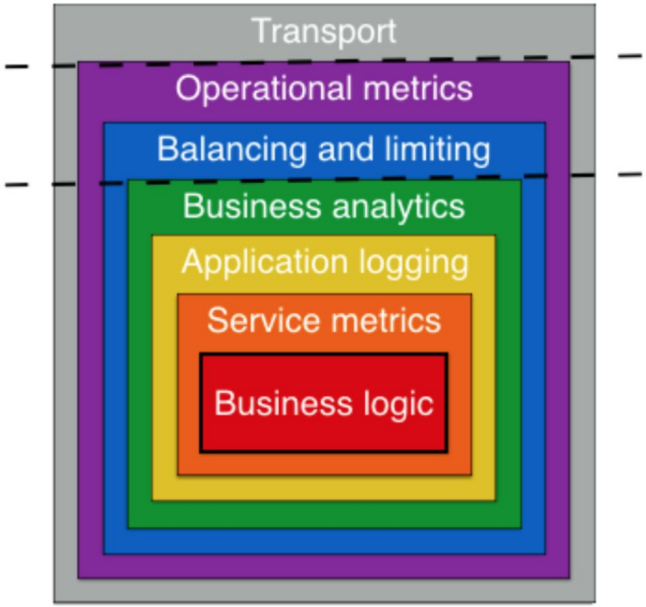
Go Kit goals:

- Operate in a heterogeneous SOA
- RPC as the primary messaging pattern
- Pluggable serialization and transport
- Operate within existing infrastructures

Go Kit architecture



Go Kit architecture (continue)



Gizmo

Gizmo is a microservice toolkit which provides packages to put together server and pubsub daemons.

Gizmo (continue)

Features:

- Standardized configuration and logging
- Health check endpoints with configurable strategies
- Configuration for managing pprof endpoints and log levels
- Useful metrics for endpoints
- Graceful shutdowns

Gizmo packages

- Server: SimpleServer & RPCServer
- server/kit
- config
- Pub/sub
- Pubsub/pubsubtest
- web

Kite

Kite is a framework for developing microservices in Go.

Kite includes both an RPC server and a client.

Kite services can discover each other via Kontrol and establish bidirectional communication.

And, it only works well with other kites via Kontrol SD within its own ecosystem.

Objective comparison

	Go Micro	Go Kit	Gizmo
License	Apache License 2.0	MIT	Apache License 2.0
Created	3 years ago (Jan, 2015)	3 years ago (Feb, 2015)	2.5 years ago (Dec, 2015)
Releases	11	7	3
Author	Asim Aslam	Peter Bourgon	JP Robinson
Stars	3381	9700	2171
Open/Closed Issues	0 / 128	35 / 270	7 / 32
Last update	April, 2018	April, 2018	April, 2018

Subjective comparison - Docs & Examples

There is a good Go Micro blog. Blog posts are also available in Medium. Examples available in the repo.

Go Kit - more information available in Peter Bourgon blog. And the better examples are in ru-rocker blog.

Gizmo - the best documentation is a source code. Examples might not work due to outdated gorilla/mux dependencies.

Subjective comparison - Users & Community

Go Micro. Used and sponsored by Sixt. 26 contributors and 363 forks in Github.

Go Kit. Development supported by DigitalOcean. 120 contributors and 996 forks in Github.

Gizmo. The main user is The New York Times. 31 contributors and 136 forks in Github.

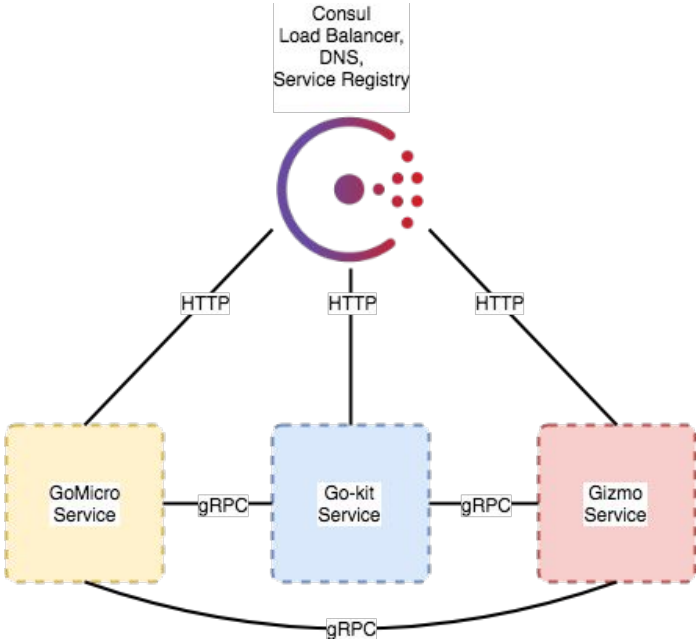
Subjective comparison - Code quality

Go Micro does not provide coverage information, but has great Go report rating.

Go Kit has good coverage 79% and excellent Go report rating.

Gizmo has the coverage rate 46% and excellent Go report rating.

Example architecture



Go Micro verdict

Services created from the protobuf declaration. Wherein, the custom compiler used to automate service code generation.

Enforces to use service discovery and pluggable architecture.

Has large collection of plugins for message encoding/decoding, RPC, pubsub, etc.

Good starting point for RPC based services. Also, comes with a Sidecar.

Go Kit verdict

Feels like a swiss-knife for microservices creation.

Requires good architecture skills to be able to compose reliable services.

Provided examples have too much boilerplate code to support layers separation. The code is mostly in copy/paste style.

Could be a great basement to build you own framework or microservice generators on top of it. Good examples are Gizmo and Microgen.

Gizmo verdict

Specific implementation of the microservice architecture. Was created with the New York Times needs in mind.

Provides several types of servers - JSON over HTTP, RPC.

Some of the server types use Go Kit.

Pubsub via AWS SNS/SQS, GCP, Kafka topics and HTTP.

Would be good to use in case of better documentation and examples.

References

- Go-microservices example source code
- Go Micro blog
- Go Micro Medium
- Go Kit: Go in the modern enterprise
- Go microservices by Peter Bourgon
- Microservices.io