Go for Microservices?

In this lesson, we'll see how Go fits for usage in the implementation of Microservices according to the criteria specified in the Requirements lesson.
Let's Begin!

We'll cover the following

GO Lang and Microservices

The criteria from the second lesson (https://www.educative.io/collection/page/10370001/6518081205567488/5806490545815552) of this chapter for the implementation of microservices can serve as a basis to assess Go's suitability as a microservices programming language.

Communication

Go supports **REST** in the standard libraries. Libraries are also available for messaging systems such as **AMQP**, for example https://github.com/streadway/amqp (https://github.com/streadway/amqp).

There is also a library for messaging with Redis (https://github.com/go-redis/redis).

Due to the widespread use of Go, there is hardly any communication infrastructure that does not support Go.

Operation

Go also offers many options for operation.

Deployment

- The **deployment** in a Docker container is very easy with Docker multi stage builds, as already illustrated.

Configuration

- Libraries like Viper (https://github.com/spf13/viper) support the **configuration** of Go applications. This library supports formats such as **YAML** or **JSON**.

Logs

- Go itself already offers support for **logs**. The Go microservices framework Go Kit contains additional features for logs (https://godoc.org/github.com/go-kit/kit/log) in more complex scenarios.

Metrics

- For **metrics**, Go Kit (https://godoc.org/github.com/go-kit/kit) supports a plethora of tools such as Prometheus, but also **Graphite** or **InfluxDB**.

New microservices

For a new microservice, it is enough to create the Docker build and then write the source code.

Resilience

Go Kit contains an implementation of resilience patterns such as Circuit Breaker (https://godoc.org/github.com/go-kit/kit/circuitbreaker). In addition, there is a port of the Hystrix library (https://github.com/afex/hystrix-go) for Go.

Microservices have to **communicate** with *other microservices*. This requires a **UI integration** in the **web UI** or **protocols** such as **REST** or **messaging**.

> It is a *macro architecture decision* which communication protocol is used (see chapter 2 (https://www.educative.io/collection/page/10370001/6518081205567488/4998953437233152)).

Q U I

7

1 What is `viper`?

○ A) A library used for the configuration of Go applications

○ B) A feature for logs

○ C) A library that supports Go output to YAML and JSON files

Submit Answer

Question 1 of 3
0 attempted

Reset Quiz

In the *next lesson*, we'll discuss variations in the implementation of Microservices.

Stay tuned!

← Back

Next →

Go

Variations

✓ Mark as Completed

Report an Issue