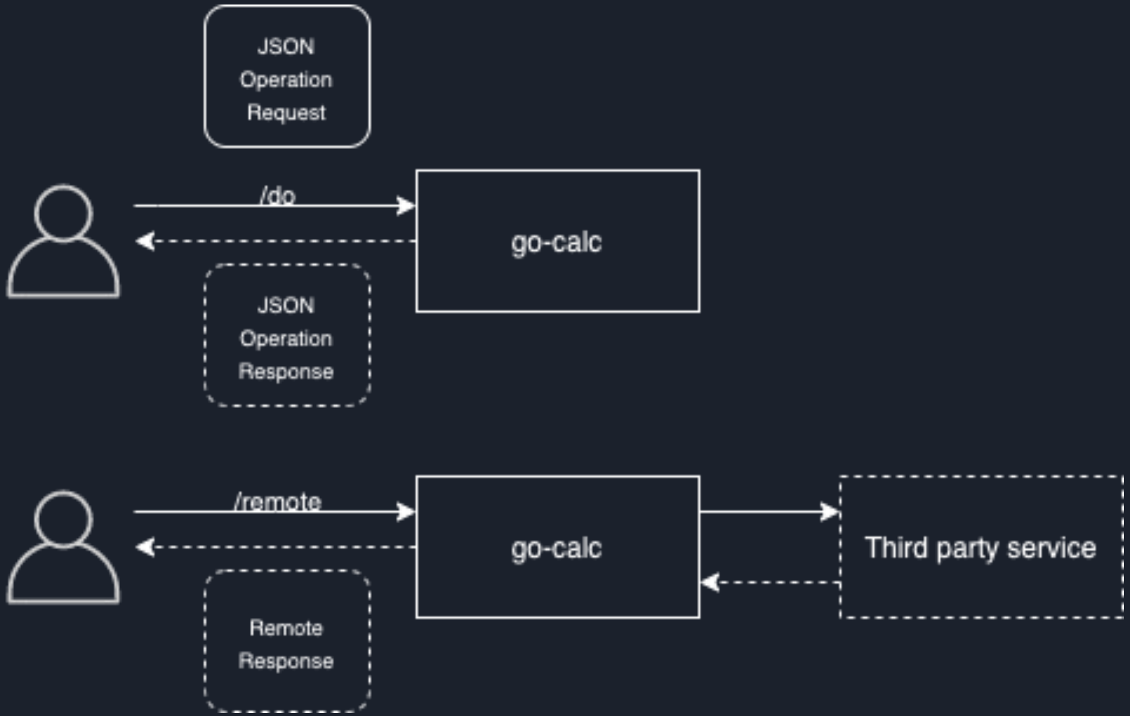
Testing HTTP service in Go

Presented by:
Anton Klimenko antklim@gmail.com

Typical HTTP service

- REST interface
- Business logic implementation
- External service call orchestration

Sample service “go-calc”



Useful design principles

- Use interfaces to decouple your code from external services implementation
- Dependency injections are unidirectional, from outer layers to inner layers

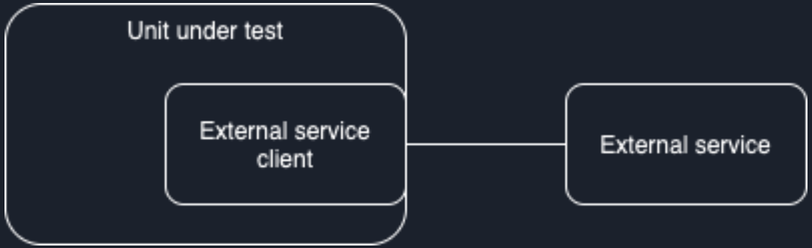
Testing route handlers

*func Handler(w http.ResponseWriter, r *http.Request)*

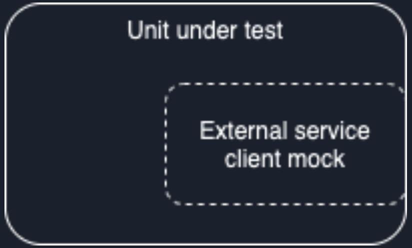
Use **net/http/httptest** to test HTTP clients and servers

- **NewRecorder** - creates new ResponseRecorder that implements **http.ResponseWriter**
- **NewRequest** - creates new **http.Request**
- **NewServer** - creates new **httptest.Server**

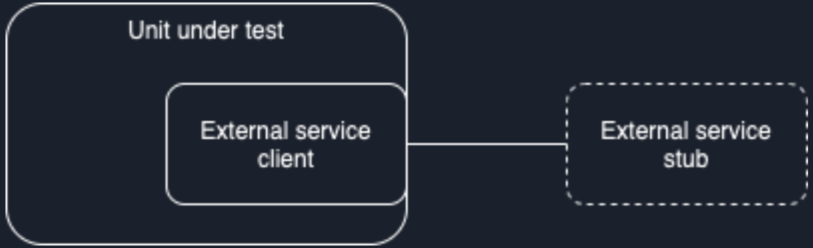
Testing HTTP calls



Option 1.



Option 2.



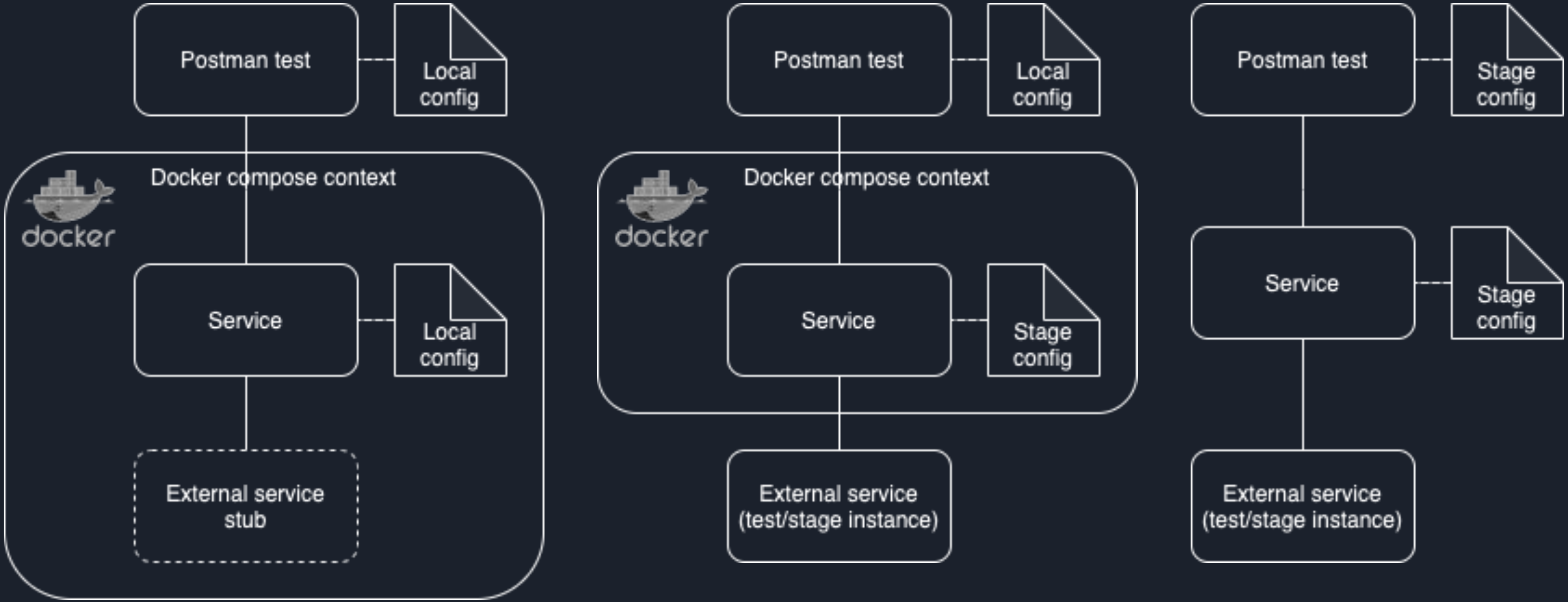
Testing HTTP calls

- Testing using client mock
  - Use mockery to generate interfaces mocks
- Testing using stub service

Testing using Postman and Newman

- Blackbox testing
- The service and its dependencies launched and connected
- Requires Docker and docker-compose

Postman testing variations



References

- http and httptest - Go HTTP package and test utilities
- testify - package with common assertions and mocks
- mockery - mock generator for Go interfaces
- Postman and Newman - API client and test tools
- Test Pyramid, test doubles - testing theory and terminology
- go-calc - sample HTTP service in Go
- Testing HTTP services in Go

Thank you

Anton Klimenko antklim@gmail.com