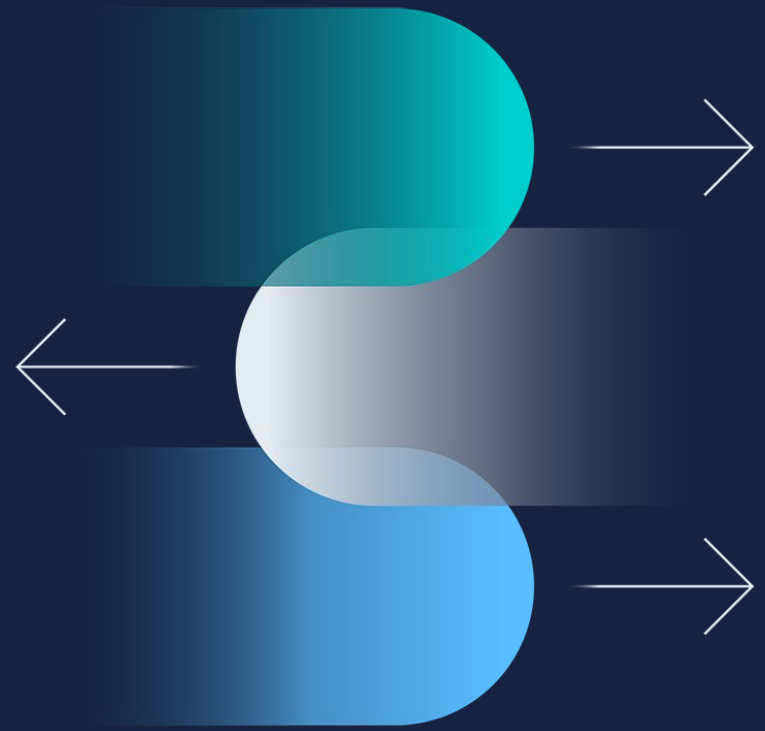
FORM3

# Comprehensive Testing Strategies For Modern Microservice Architectures

Adelina Simion

Let’s make writing tests easier, funner and more efficient!



Oh Hai!

# 👋 About me

🎙 Tech Evangelist @ Form3

💻 Java & Go engineer

🎧 LinkedIn Learning instructor

📙 Author of “Test-Driven Development In Go”


classic_addetz

adelina-simion

addetz

FORM3

Oh Hai!

# 💼 About Form3

- Real-time payments processing platform
- Multi-cloud (AWS, GCP, Azure)
- Go, IaC (Terraform)
- SecDevOps culture
- Fully remote


FORM3

# 🦠 Bug fixing

What percentage of time do you think developers spend fixing bugs?

A) <25%
B) 25-50%
C) >50%

Creator: jenifoto | Credit: Getty Images/iStockphoto
Copyright: jenifoto

FORM3

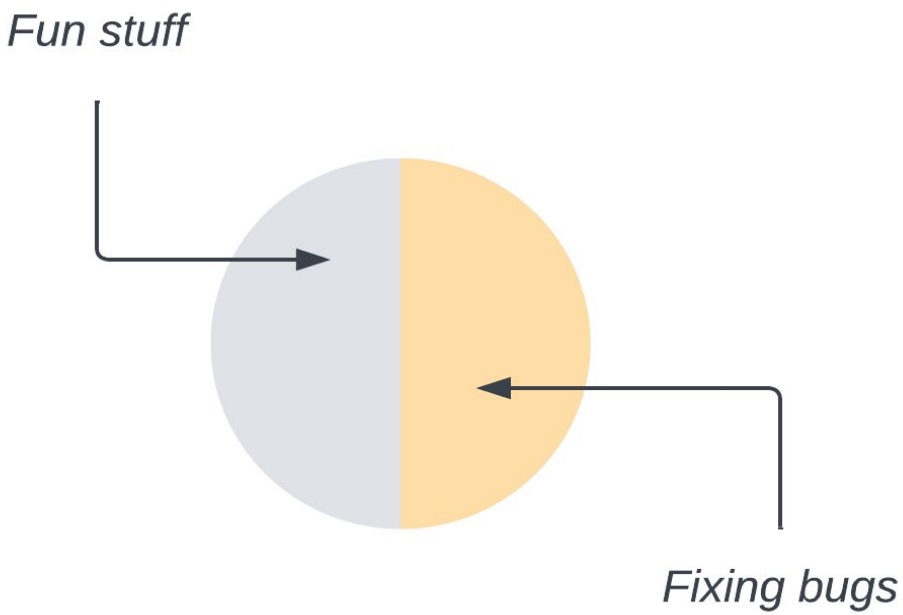
# 🦠 Bug fixing

What percentage of time do you think developers spend fixing bugs?

A) <25%
B) 25-50%
C) >50%

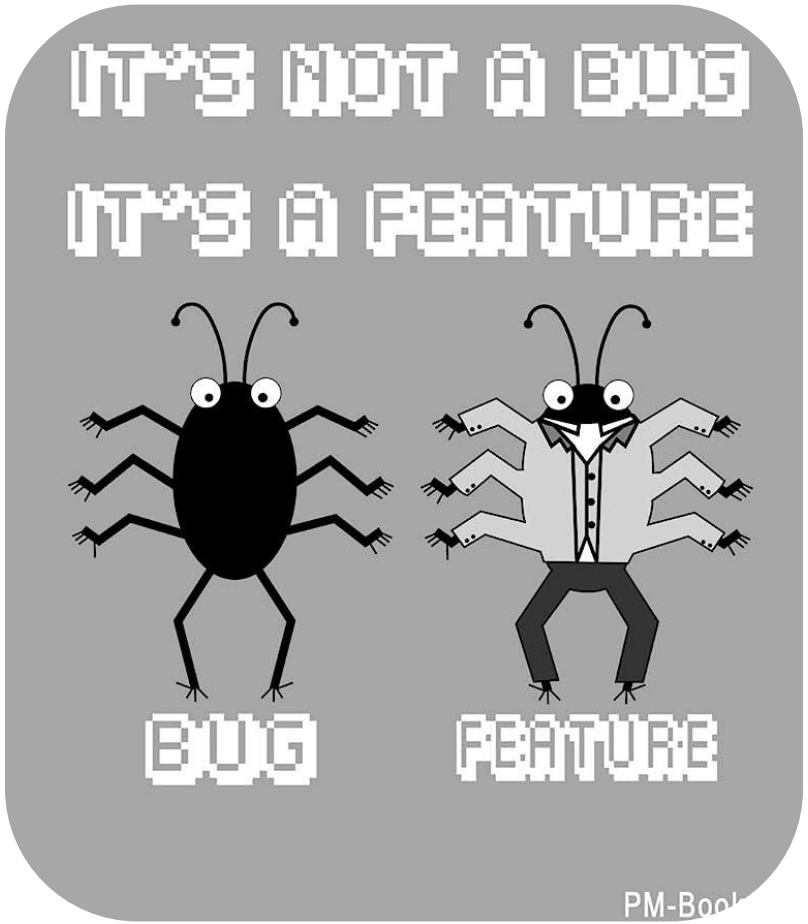
Developers spend 25-50% of their time fixing bugs, with some spending up to 75% of their time bug fixing.

*(Source: Rollbar survey)*



# Common bug causes

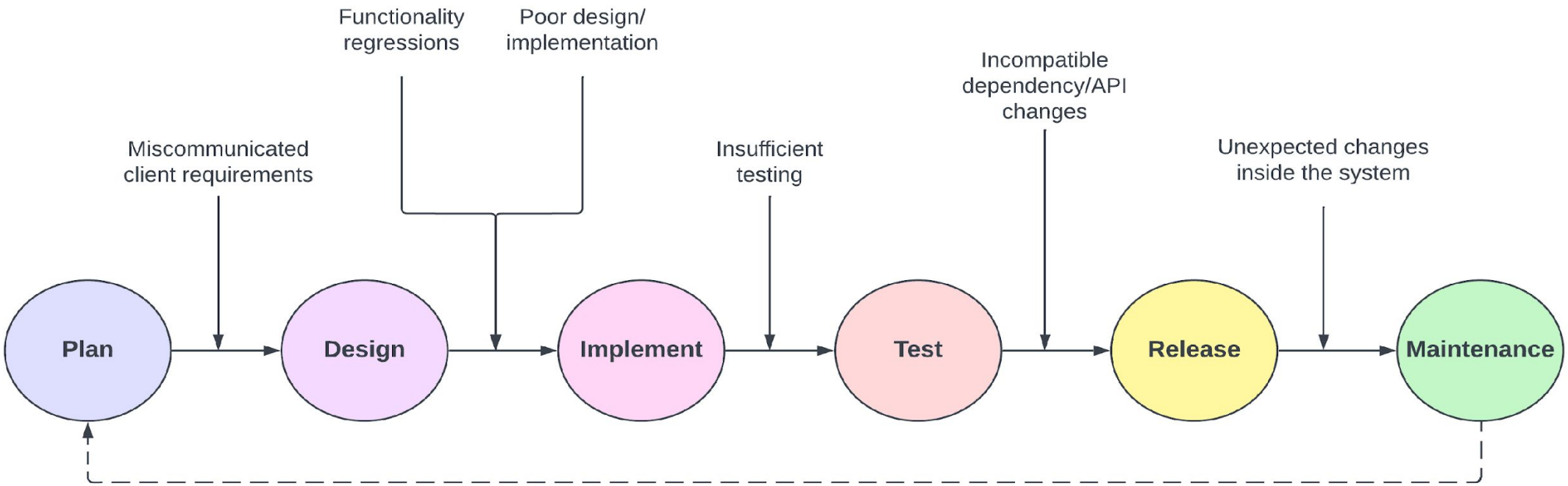
- Miscommunicated client requirements
- Poor design/implementation
- Insufficient testing
- Functionality regressions
- Incompatible dependency/API changes
- Unexpected changes inside the system




Creator: PM Books

FORM3

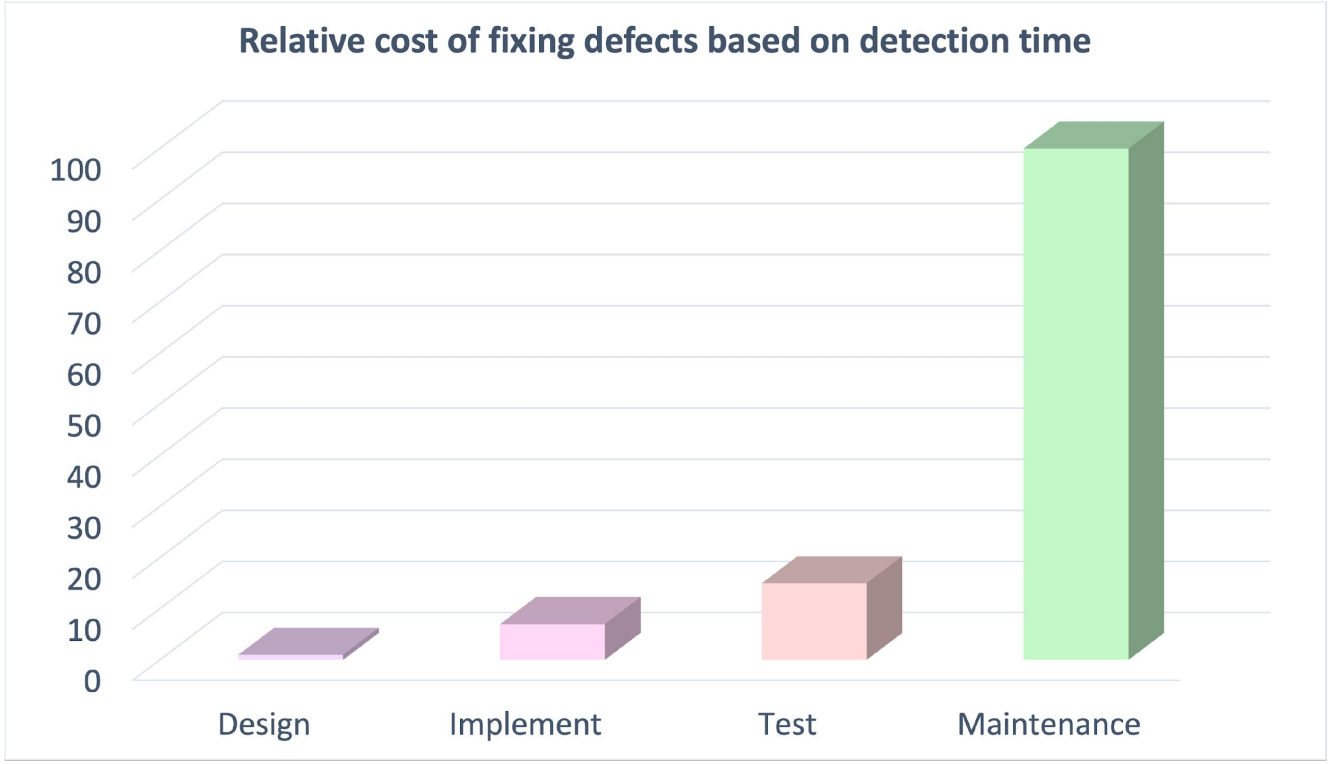
# Bugs in the SDLC



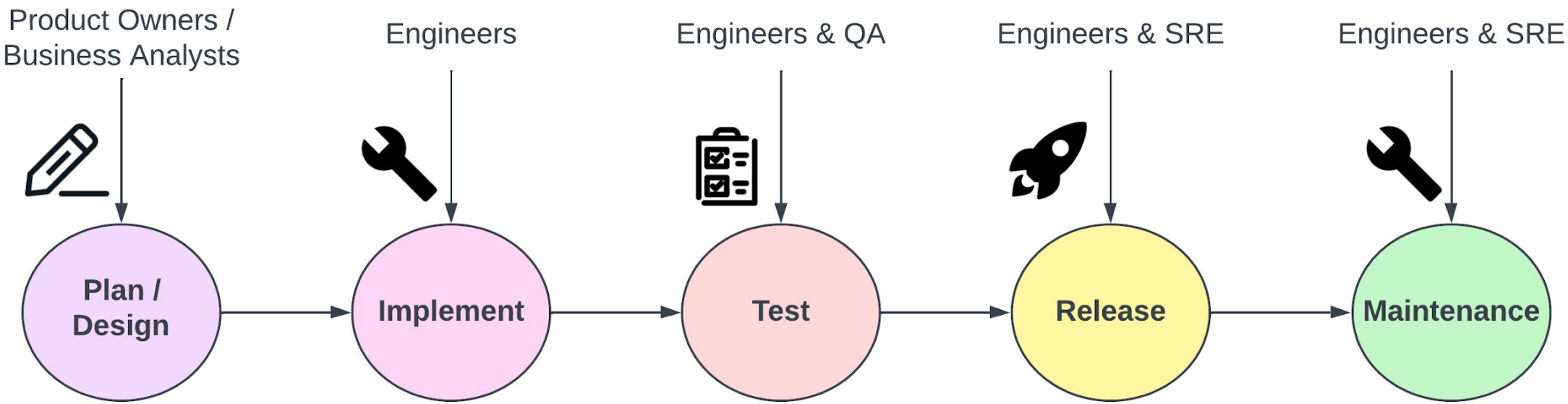
# 💰 Exponential costs

- **Detect early, detect often** to avoid delays and increased costs (the **“shift left”** motto)
- The **cost of fixing bugs increases up to 100 times** when identified after release.

(Source: *IBM study*)



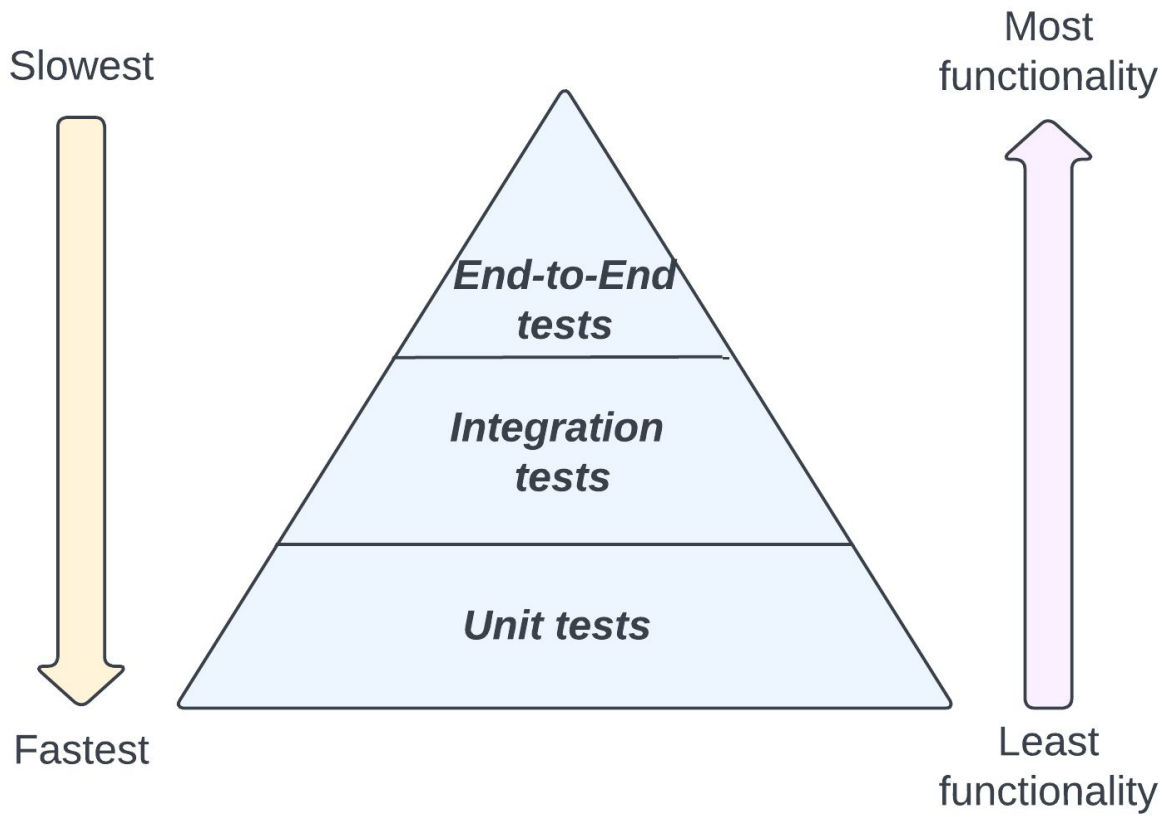
# 💻 An engineer's life ...




Main players in the SDLC

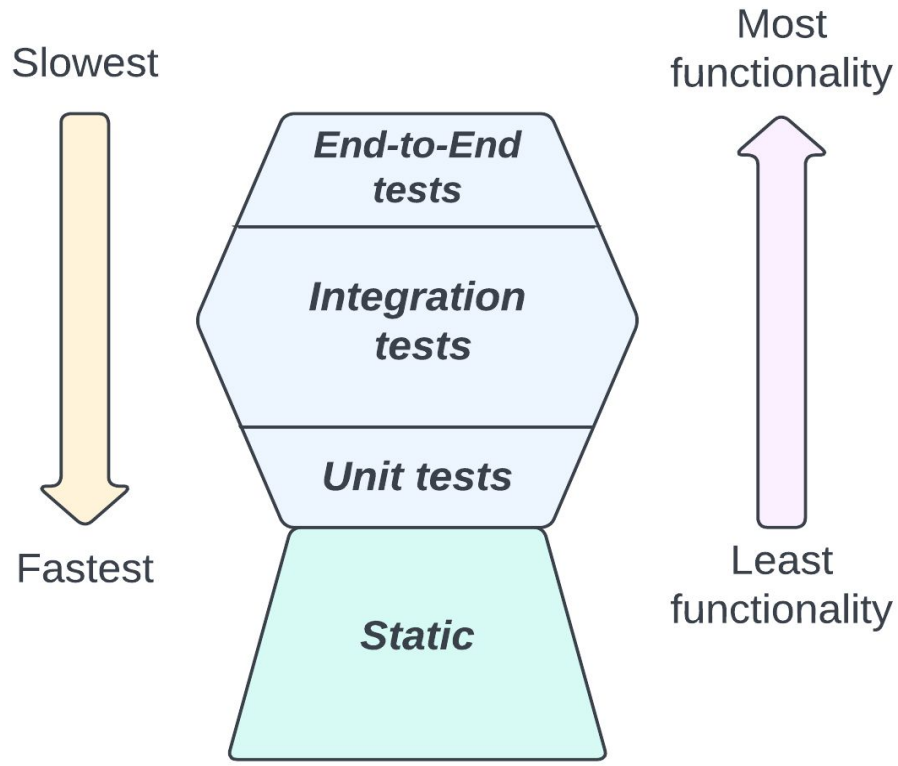
# 🔺 The testing pyramid

- **Unit tests** verify internal code mechanisms.
- **Integration tests** verify multiple units work well together.
- **End-to-end tests** verify user interactions.



# 🏆 The testing trophy

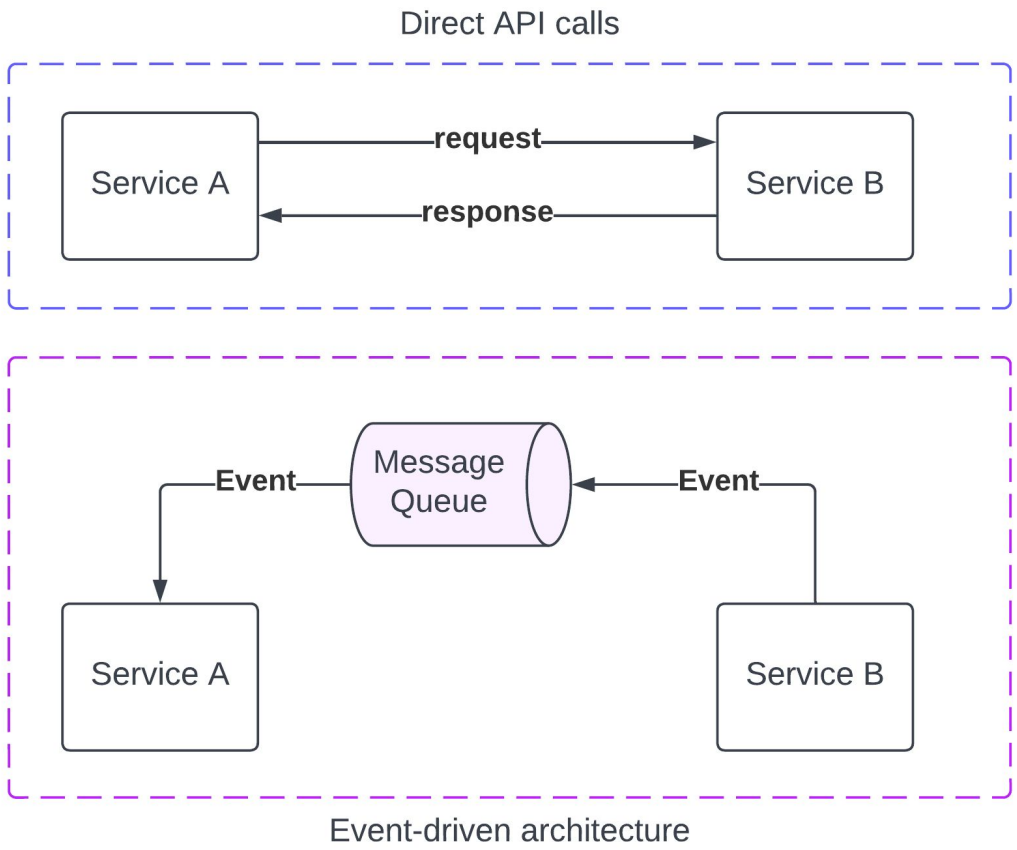
- Tests have diminishing returns and can also add a maintenance burden.
- Common static checks are:
  - Reviews
  - Static analysis
  - Checkstyle
  - Linters
- Code coverage should be **in 80-90% range**.




FORM3

# 🧐 A closer look...

- Microservices change without any central oversight and are released at different times.
- They might directly integrate with each other or use event driven architectures such as event buses/queues.




FORM3

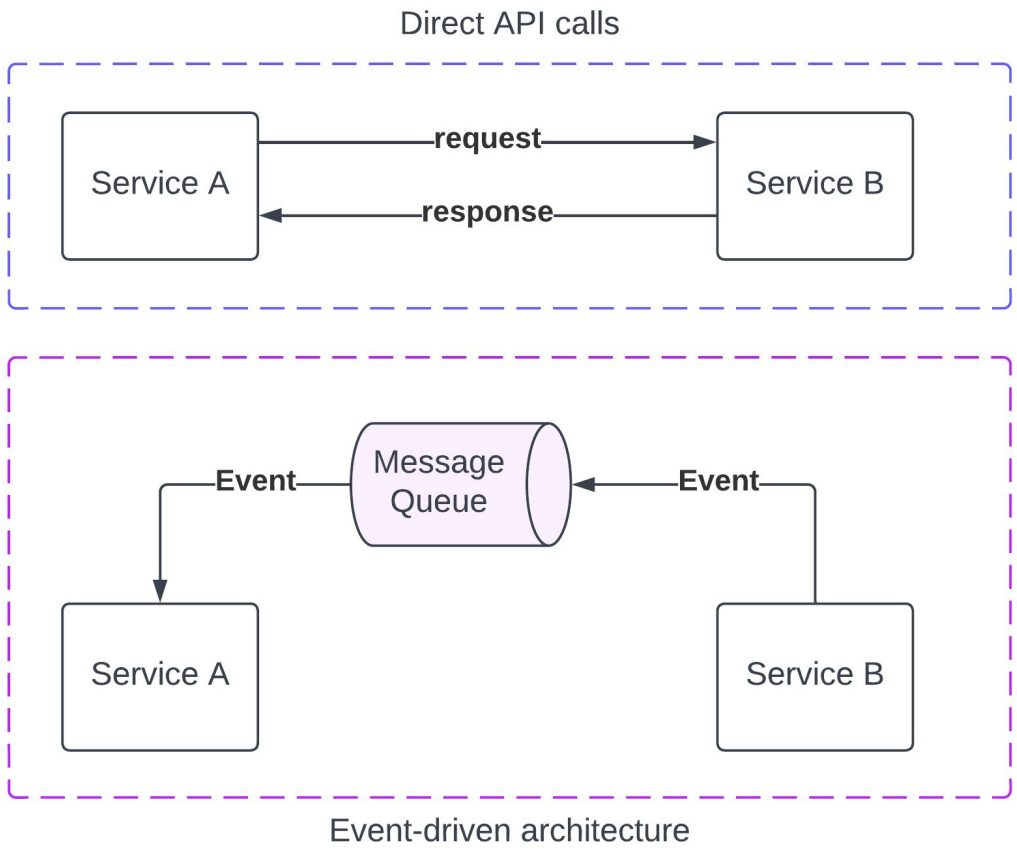
# 😱 Everything can go wrong!

**Direct API calls**

- Request & response format issues
- Incorrect service behaviour
- Service outages & high latency

**Event-driven architecture**

- Event payload format issues
- Message broker outages
- Events acknowledged but not processed

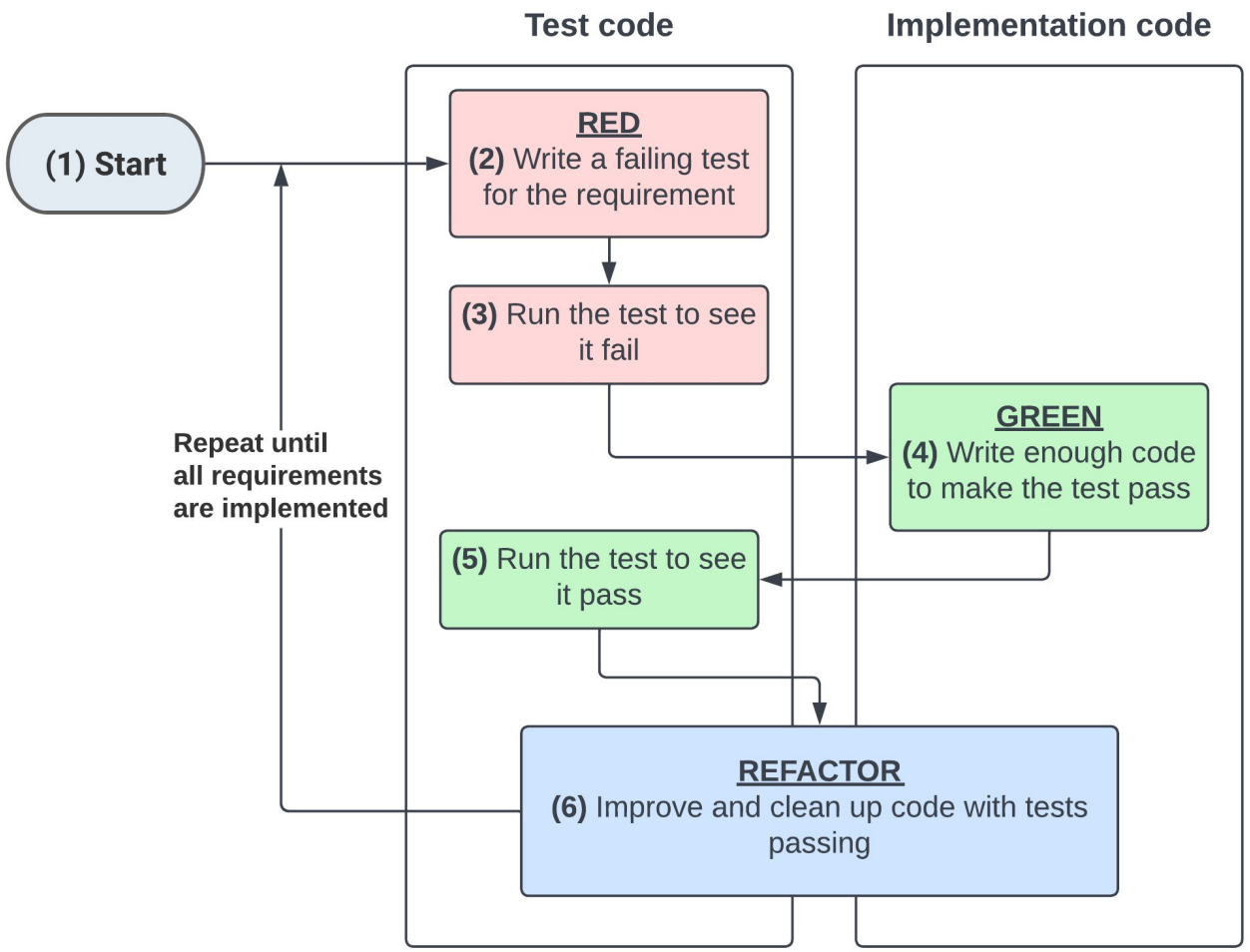



FORM3

# ⚒️Let's start pulling the testing lever!

## Unit tests

- Tests are written alongside implementation using **TDD**
- Asserts the logic of the **Unit Under Test (UUT)** in isolation
- Test setup typically follows the **Arrange-Act-Assert (AAA)** structure




FORM3

# 1 Unit testing in Go

- In Go, packages are the smallest unit under test as they are a self-contained mini-API.
- Tests are the first clients/ consumers of our packages.
- Tests are functions that satisfy a few conventions.

```go
func SayHello(name string) string {
  return fmt.Sprintf("Hello, %s!", name)
}

func TestSayHello(t *testing.T) {
  names := []string{"Anna", "Belle"}
  for _, n := range names {
    t.Run(n, func(t *testing.T) {
      want := "Hello, " + n + "!"
      got := SayHello(n)
      if got != want {
        t.Fatalf("got: %s; want: %s",
          got, want)
      }
    })
  }
}
```

# 🧐 Fuzz testing

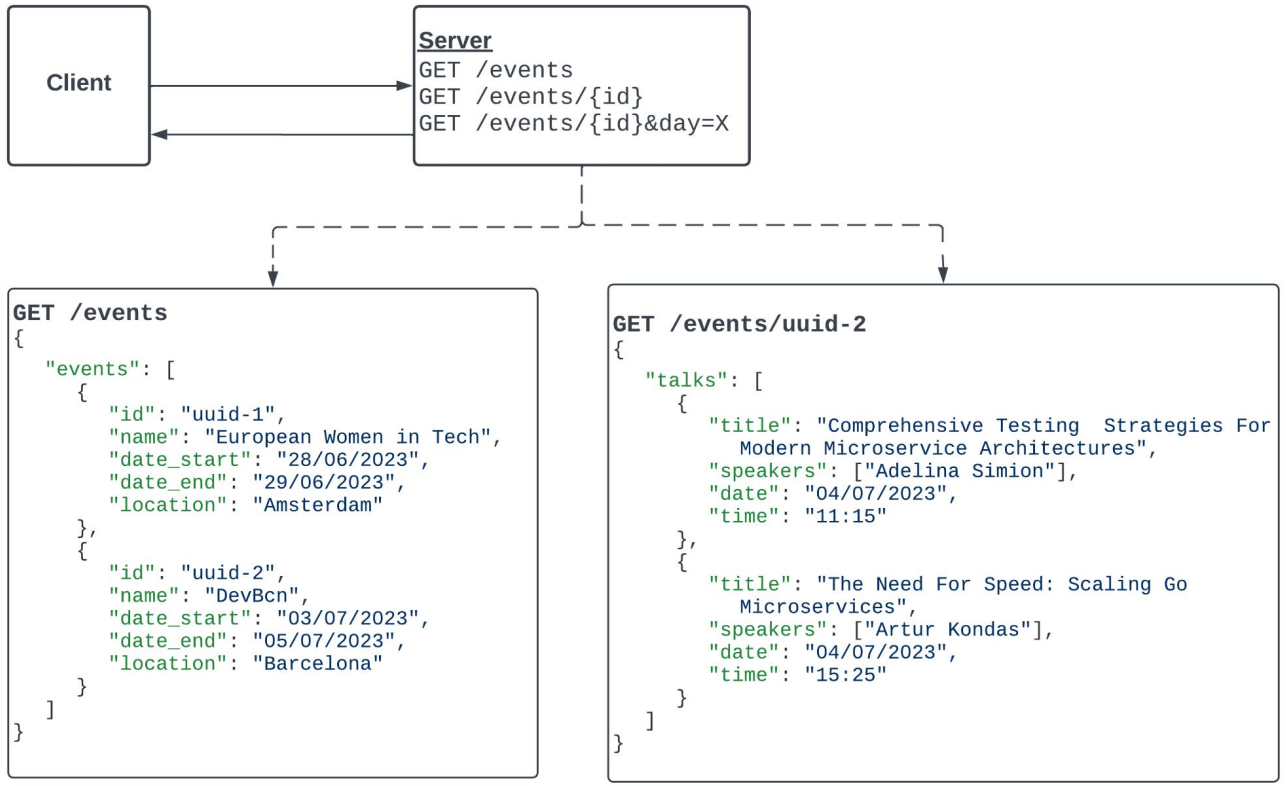
**Edge cases coverage through unit tests**

- Automated tests that are run using the same testing commands.
- Fuzzed tests generate values, making it easy to write many unit tests without having to manually write test cases.
- Help us to ensure **code robustness**

```
func FuzzSayHello(f *testing.F) {
  f.Add("Anna")
  f.Add("Belle")
  f.Fuzz(func(t *testing.T, n string) {
    want := "Hello, " + n + "!"
    got := SayHello(n)
    if got != want {
      t.Fatalf("got: %s; want: %s",
        got, want)
    }
  })
}
```

FORM3

# 🗣️Introducing the Conference Talks service!




FORM3

# 🥳 Show me the code!

github.com/addetz/testing-strategies-demo


FORM3

# 🔼 Moving up the testing pyramid!

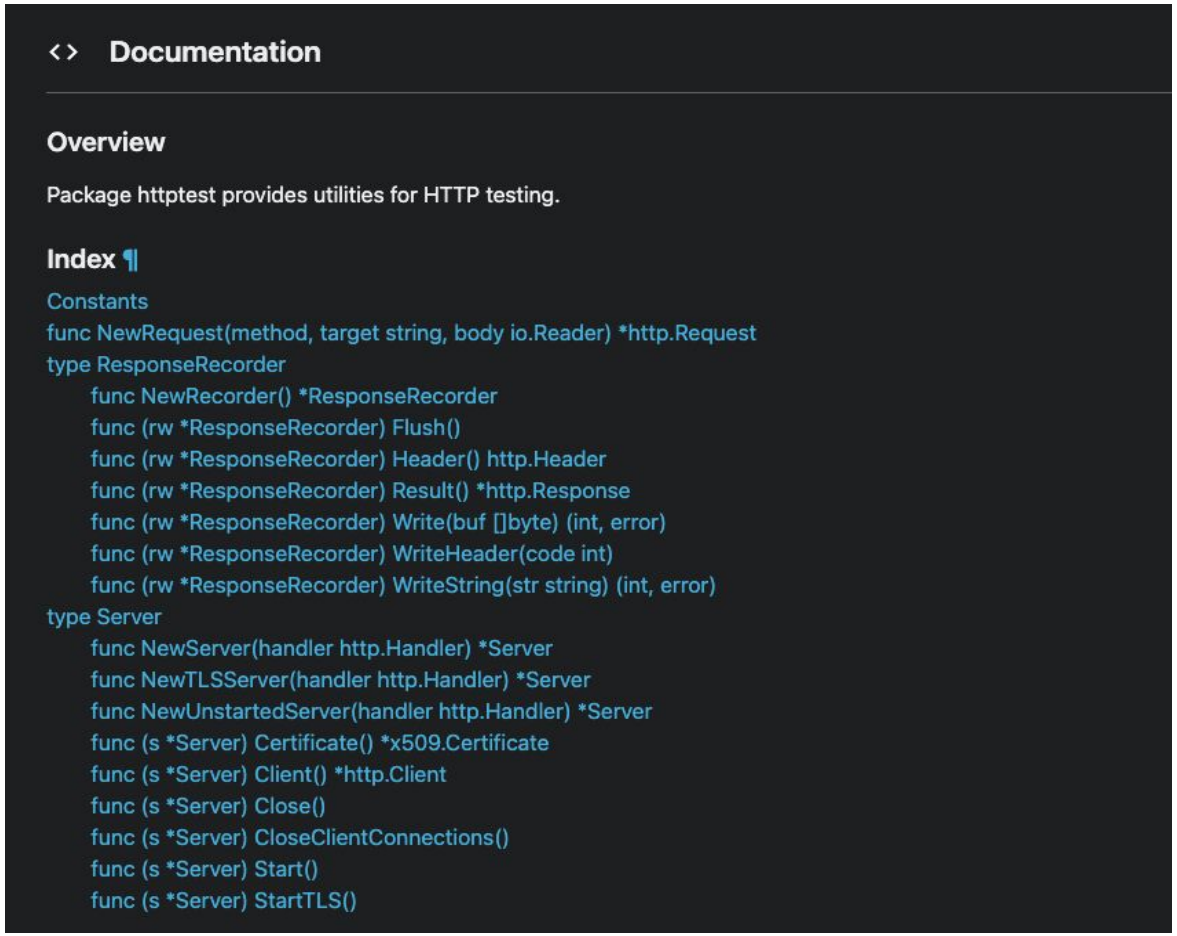
| | **Integration tests** | **End-to-end tests** |
|---|---|---|
| **Purpose** | Test integration with external and internal modules | Test user flow and complete user experience |
| **Cost** | + Fast & cheap | - Slow & expensive |
| **Timing** | + Performed earlier in the SDLC | - Performed at the end of the development cycle, when application is running |
| **Scope** | + Ensure that a component works with code that it does not own | - Ensure that specific user workflows perform correctly |

FORM3

# 💎 The `httptest` library

- Starts a test server under the hood and allows us to verify HTTP requests and responses.
- Tests look very similar to client code, making it easier for us design client & server integrations too.

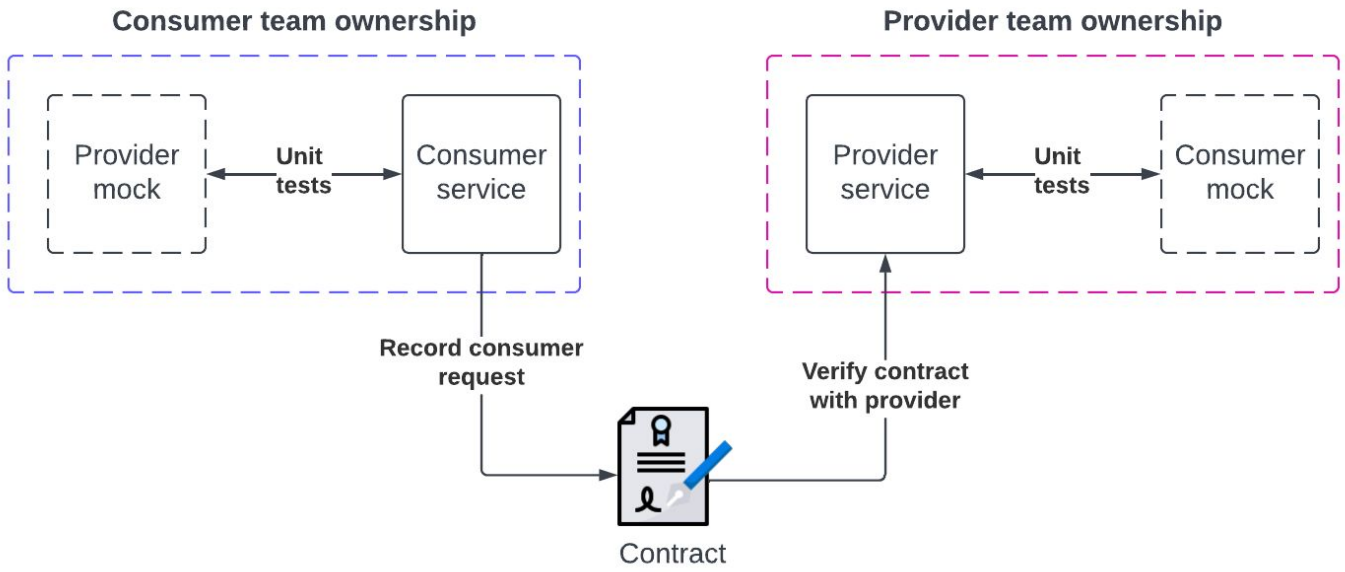
pkg.go.dev/net/http/httptest




FORM3

# 🤝 Contract testing

- The **consumer** issues the request for data from either a queue or service.
- The **provider** responds to the request and sends the data.
- The **contract** records the expectations of both the consumer and provider, allowing us to establish a common language.

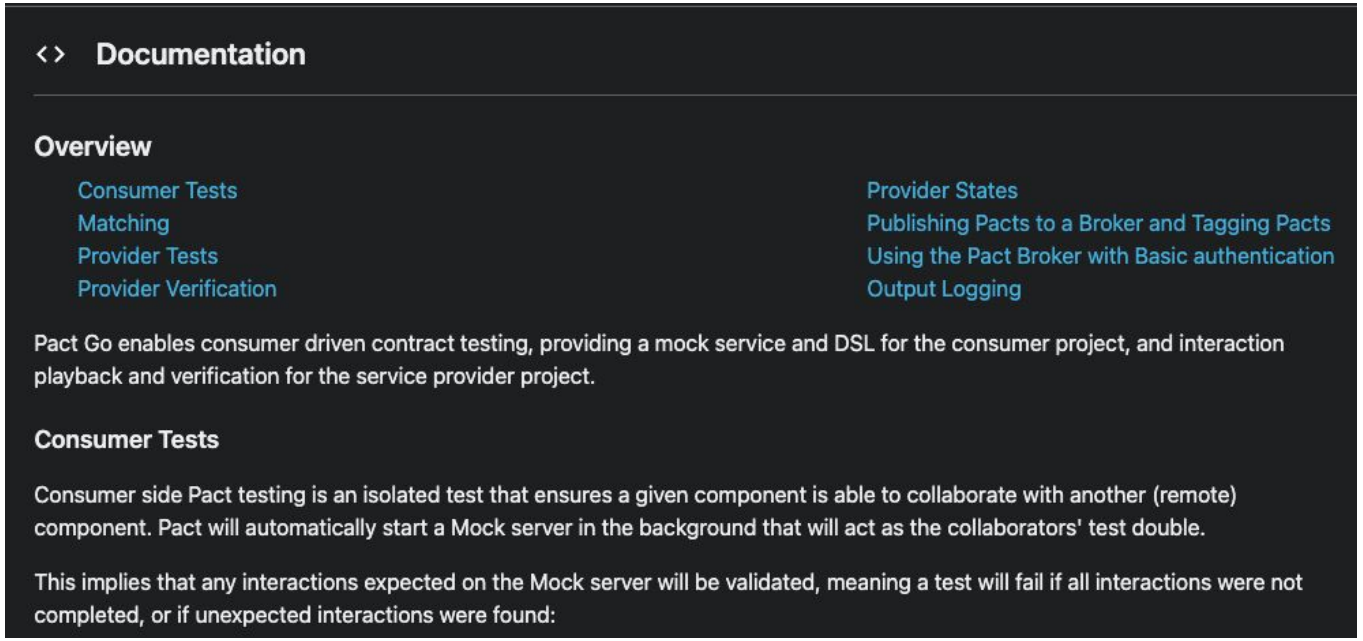



FORM3

# 💎 The `pact` tool

- Contract testing framework with libraries in over 10 languages.
- Provides:
  - mocking
  - verification
  - domain-specific language (DSL)
  - playback capabilities

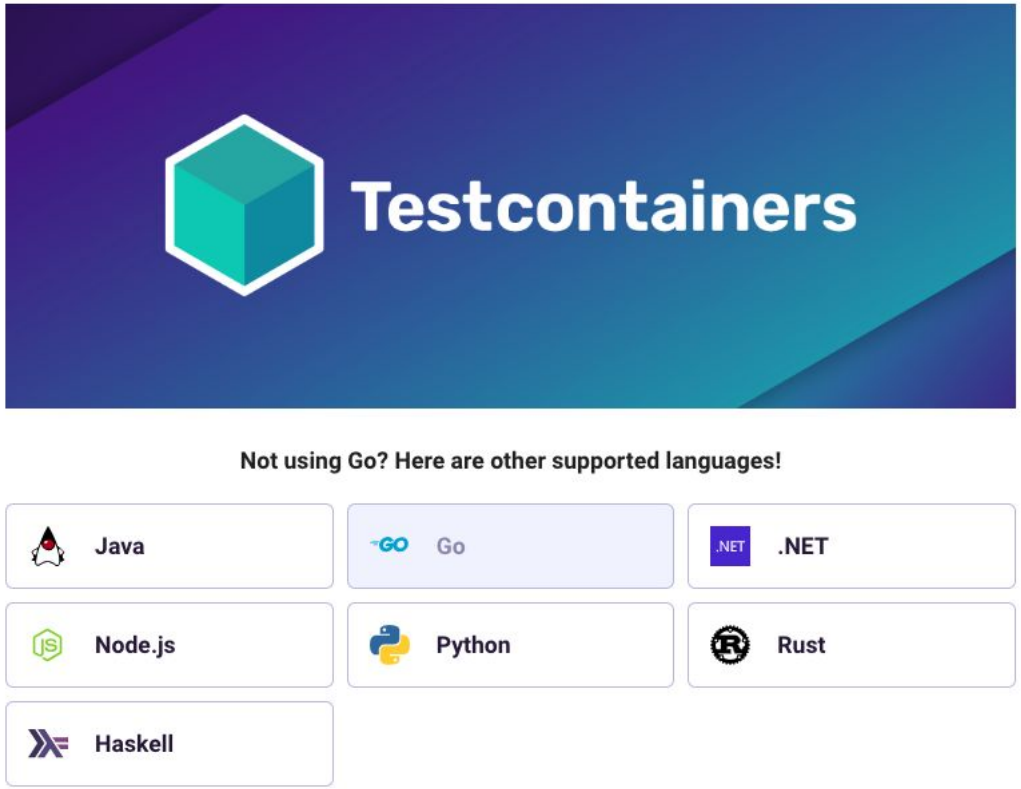
github.com/pact-foundation/pact-go/v2



# 💎 The `testcontainers` project

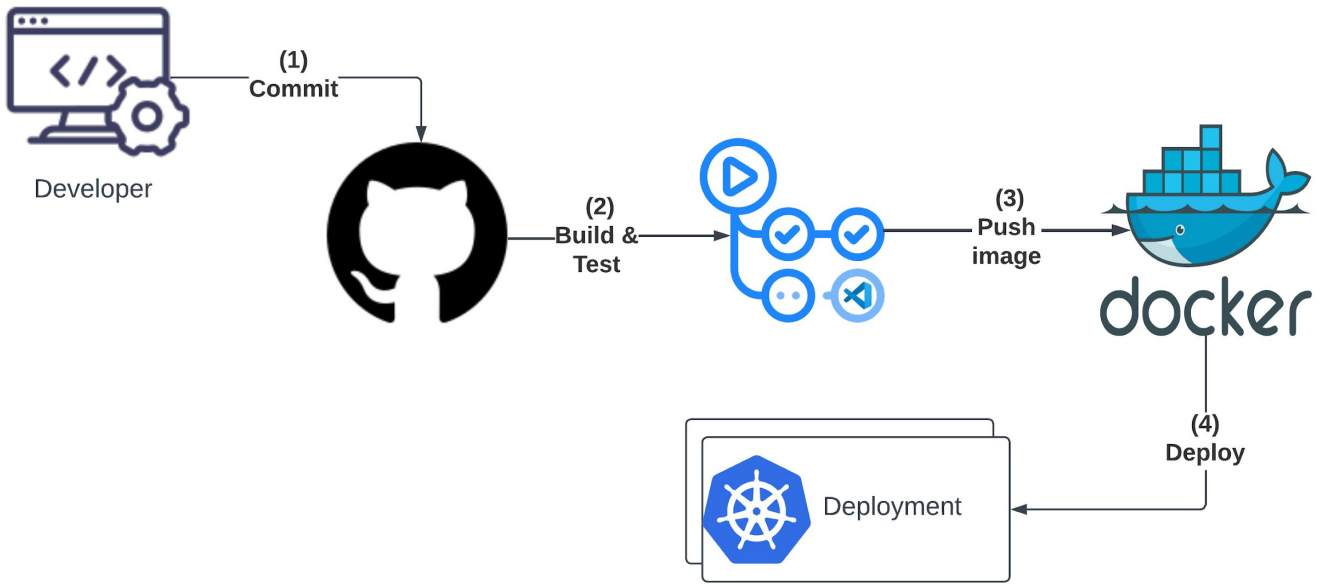
- A project which makes it easy to manage containers used for integration and end-to-end tests.
- Bundles together container managements together with code writing, simplifying test setup.

github.com/testcontainers/testcontainers-go



# 🚦 Making tests quality gates

- For consistency, tests should be run as part of the CI/CD pipelines to avoid outages, bugs and regressions.
- Unit tests are usually run on all branches, while the longer running tests are run on the **main branch only**.
- **GitHub Actions** are a popular solution for creating custom CI/CD pipelines.



# 🥳 Show me the code!

github.com/addetz/testing-strategies-demo


FORM3

# ✅ Summary

- **Unit tests** verify the behaviour of the isolated parts of our code.
- **Integration tests** extend test scope without losing performance.
- **Contract testing** verifies the API integration between microservices.

# ✅ Summary

- **Unit tests** verify the behaviour of the isolated parts of our code.
- **Integration tests** extend test scope without losing performance.
- **Contract testing** verifies the API integration between microservices.

- Tests should be your quality gates to prevent outages.

# ✅ Summary

- **Unit tests** verify the behaviour of the isolated parts of our code.
- **Integration tests** extend test scope without losing performance.
- **Contract testing** verifies the API integration between microservices.

- Tests should be your quality gates to prevent outages.

- **Embrace a testing and quality culture in your teams!**

# Thank you for your time!

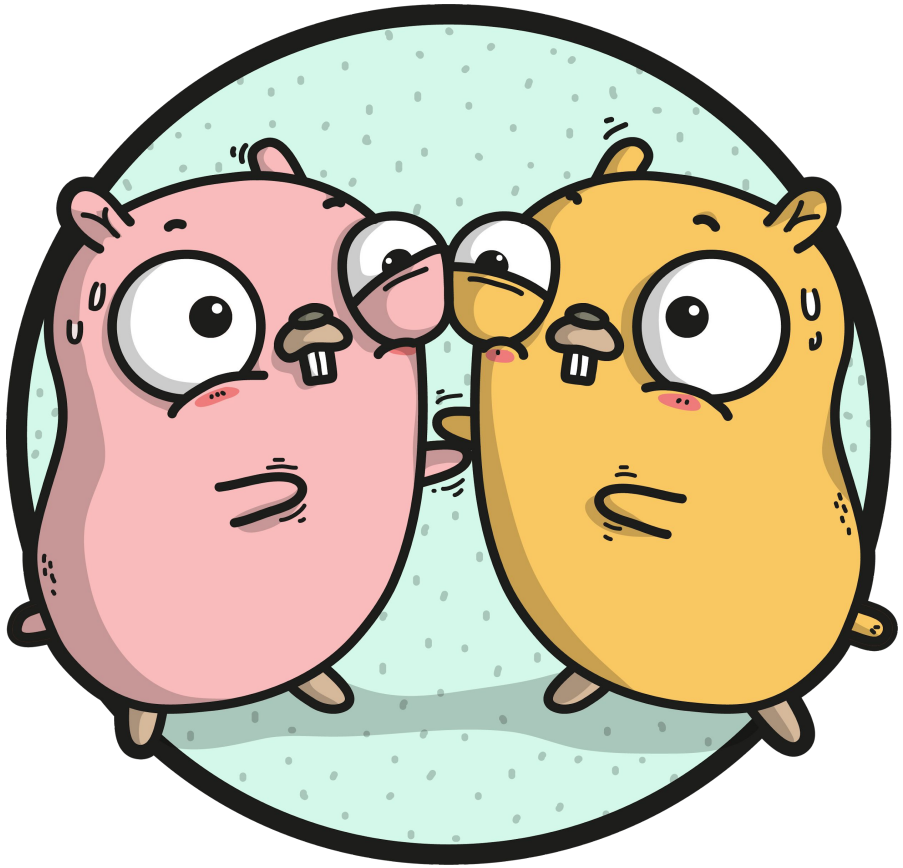
Hope you have fun testing your applications ❤️

 github.com/addetz/testing-strategies-demo

FORM3

# ❤️Let’s be pals!

- classic_addetz
- adelina-simion
- addetz



Creator: https://github.com/ashleymcnamara/gophers

FORM3