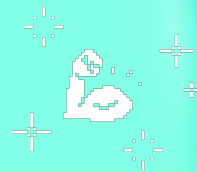


Using Cucumber to test your Go APIs

By Koen Bollen, 2022



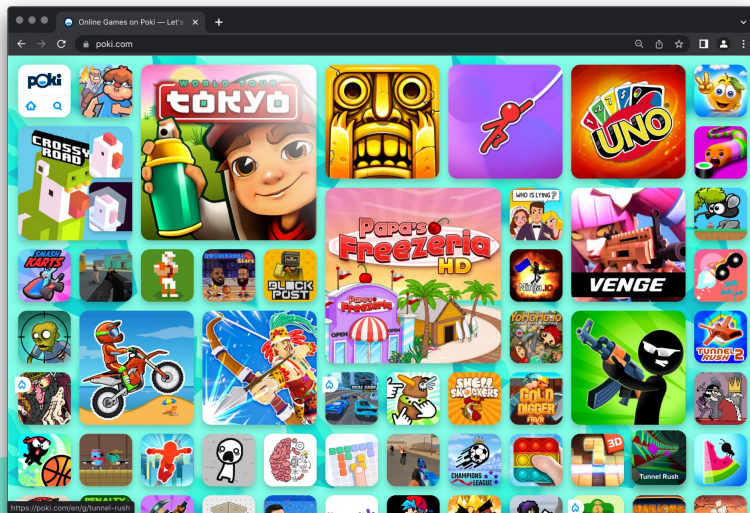
Quick intro!

History of Koen Bollen:







- Taught at University of Applied Sciences in Amsterdam
- Gamedev → PC & Mobile
- Backend Engineer at Blendle
- Started at Poki in 2019

About Poki

- Web game portal
- ±40 million unique users per month
- Work with over 300 game developers on a revenue share basis



Using Cucumber to test your Go APIs

1.  About me & Poki
2.  What is Cucumber
3.  Testing your APIs
4.  Why?
5.  Demo Time
6.  Closing Notes

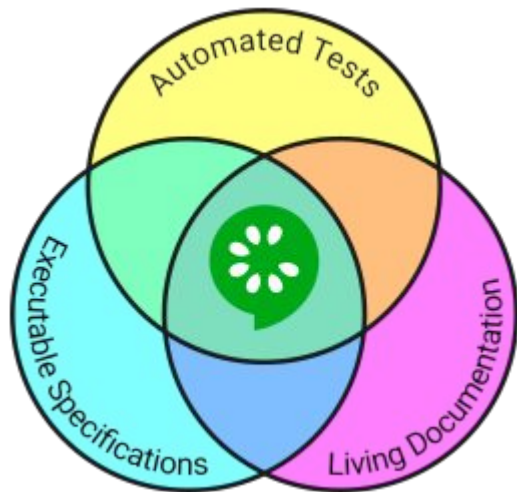


100+



What is Cucumber

What is Cucumber?



```
# Comment
Feature: Eating too many cucumbers may not be good for you
  Eating too much of anything may not be good for you.

Scenario: Eating a few is no problem
  Given Alice is hungry
  When she eats 3 cucumbers
  Then she will be full
```



Running a cucumber scenario

Comment

Feature: Eating too many cucumbers may not be good for you
Eating too much of anything may not be good for you

Scenario: Eating a few is no problem

Given Alice is hungry

When she eats 3 cucumbers

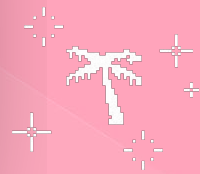
Then she will be full



github.com/cucumber/godog

```
people := make(map[string]int)
var activePerson string
step(`/(.*) is hungry/`, func(name string) {
    people[name] = 0
    activePerson = name
})
```

```
step(`/she eats (\d+) cucumbers/`, func(amount int) {
    people[activePerson] += amount
})
step(`/then she will be full/`, func() error {
    if people[activePerson] < FullnessThreshold {
        return fmt.Errorf("%s is not full", activePerson)
    }
})
```



poki

Examples:

Feature: API Service

Scenario: GET the health endpoint

When the client does a GET request to `"/health"`

Then the response code equals 200 (OK)

And the response header `"Content-Type"` should be `"application/json"`

And the response body should be:

```
"""json
{
  "healthy": true
}
"""
```

Feature:

Scenario: Terms and Conditions effective from 17 May 2022

Given the contract is created

When the contract is ready to sign

Then the Seller agrees to Term and Conditions version 17 May 2022

Feature: Players can create and connect a network of players

Background:

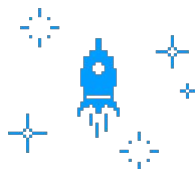
Given the `"signaling"` backend is running

Scenario: A player can create a network to join a game

When `"green"` creates a network for game `"164aae2e-c6e5-4073-80bf-b2a03ad4c9b7"`

Then `"green"` receives the network event `"ready"`

And `"green"` has received the peer ID `"h5yzwyizlwao"`



100+



Testing your APIs

Using Go's standard testing package

```
func TestHandler_ListUsers(t *testing.T) {  
    req := httptest.NewRequest("GET", "/users", nil)  
    resp := httptest.NewRecorder()  
  
    mockStore := &stores.Memory{}  
    fakeClient := &cloudfare.MockClient{}  
  
    mockStore.AddMockUser("1", "john")  
    mockStore.AddMockUser("2", "kate")  
  
    handler := Handler(context.Background(), mockStore, fakeClient)  
  
    handler.ServeHTTP(resp, req)  
  
    if resp.Code != http.StatusOK {  
        t.Errorf("expected status code %d, got %d", http.StatusOK, resp.Code)  
    }  
  
    // test body...  
}
```

Maybe use cucumber?

Feature: List users

Scenario: List all users from a team

Given these "teams" records:

id	name
1	A Team
2	B Team

And these "users" records:

id	team_id	name
1	1	Kate
2	2	John
3	1	Jane

When the client does a GET request to `/teams/1/users`

Then the response code equals 200 (OK)

And the response header `"Content-Type"` should be `"application/json"`

And the response body should be:

```
"""json
[
  {"id": 1, "team_id": 1, "name": "Kate"},
  {"id": 3, "team_id": 1, "name": "Jane"}
]
"""
```



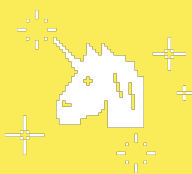
```

scenario.Step(`/^these "([^"]*)" records:$/, givenTheseRecords)

func givenTheseRecords(tableName string, data *godog.Table) error {
    tx, err := database.Begin()
    if err != nil {
        return err
    }
    fields := ExtractTableHeaders(data)
    marks := strings.Repeat("?, ", len(fields)-1) + "?"

    stmt, err := tx.Prepare("REPLACE INTO " + tableName + " (`" + strings.Join(fields, "`", "`") +
        "`) VALUES (" + marks + ")")
    if err != nil {
        tx.Rollback()
        return err
    }
    for i := 1; i < len(data.Rows); i++ {
        var vals []any
        for _, cell := range data.Rows[i].Cells {
            vals = append(vals, cell.Value)
        }
        if _, err = stmt.Exec(vals...); err != nil {
            tx.Rollback()
            return err
        }
    }
    return tx.Commit()
}

```

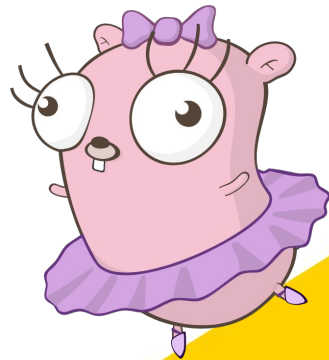


Why?



Pros

- Everyone can read these tests
 - It can double as documentation
 - Frontend developers can read it 🔥
- Super easy to write tests up front before worrying about implementation
- Cucumber can be used in any programming language
- 100% test coverage of documented features!
- We don't run our codebase locally anymore



Cons

- More work upfront implementing *Step Definitions*
- Might run a little slower than native unit tests
- ???





Demo Time

100+



Closing Notes

- We cucumber test our happy-path and some common cases
 - We cover most edge cases in unit tests
- Super useful for regression tests and red/green tests

Thanks! Any questions?

Links

<https://cucumber.io/>

<https://github.com/cucumber/godog>

<https://about.poki.com/>

<https://github.com/egonelbre/gophers>

Code snippets created with: <https://carbon.now.sh/>



fin