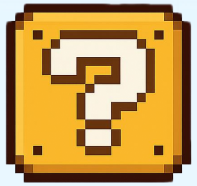
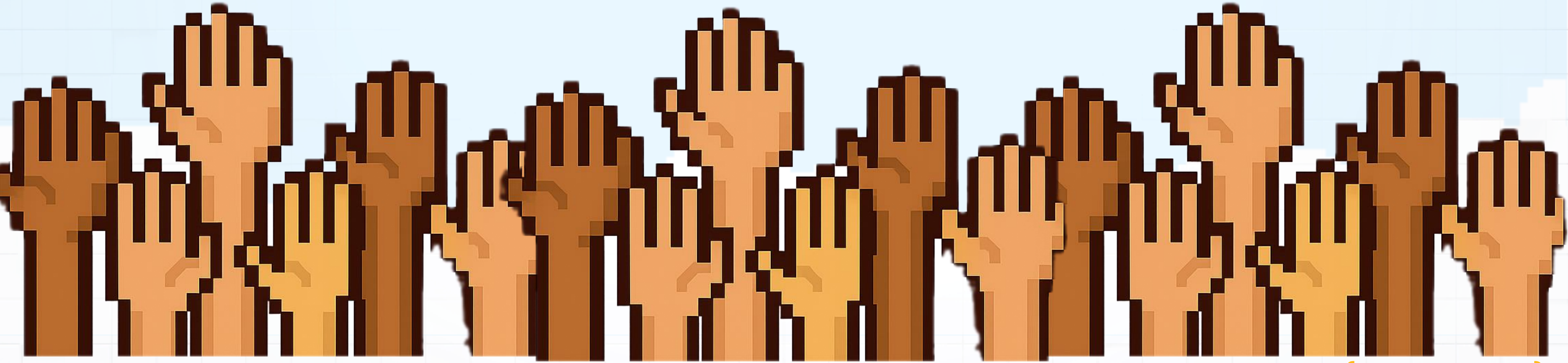
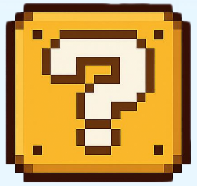
Have You Ever Written a Software Test Before?





{CHECK.conf}

What Was Your Motivation to Write Tests?



Catch bugs before they hit production

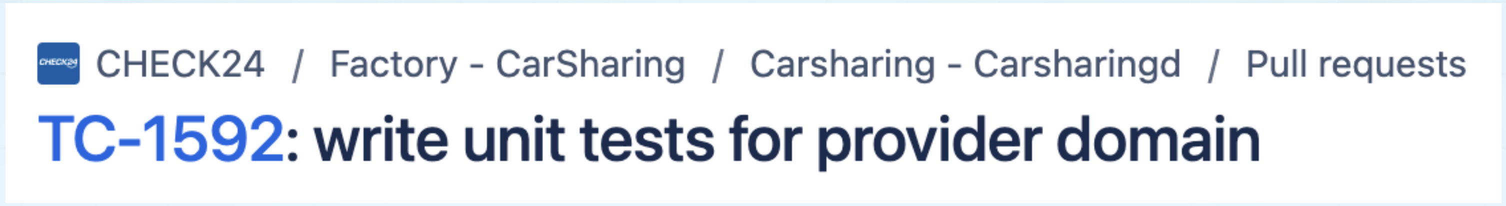
Gain confidence while refactoring

Document expected behavior

Because your team lead told you to

To bump the test coverage metric

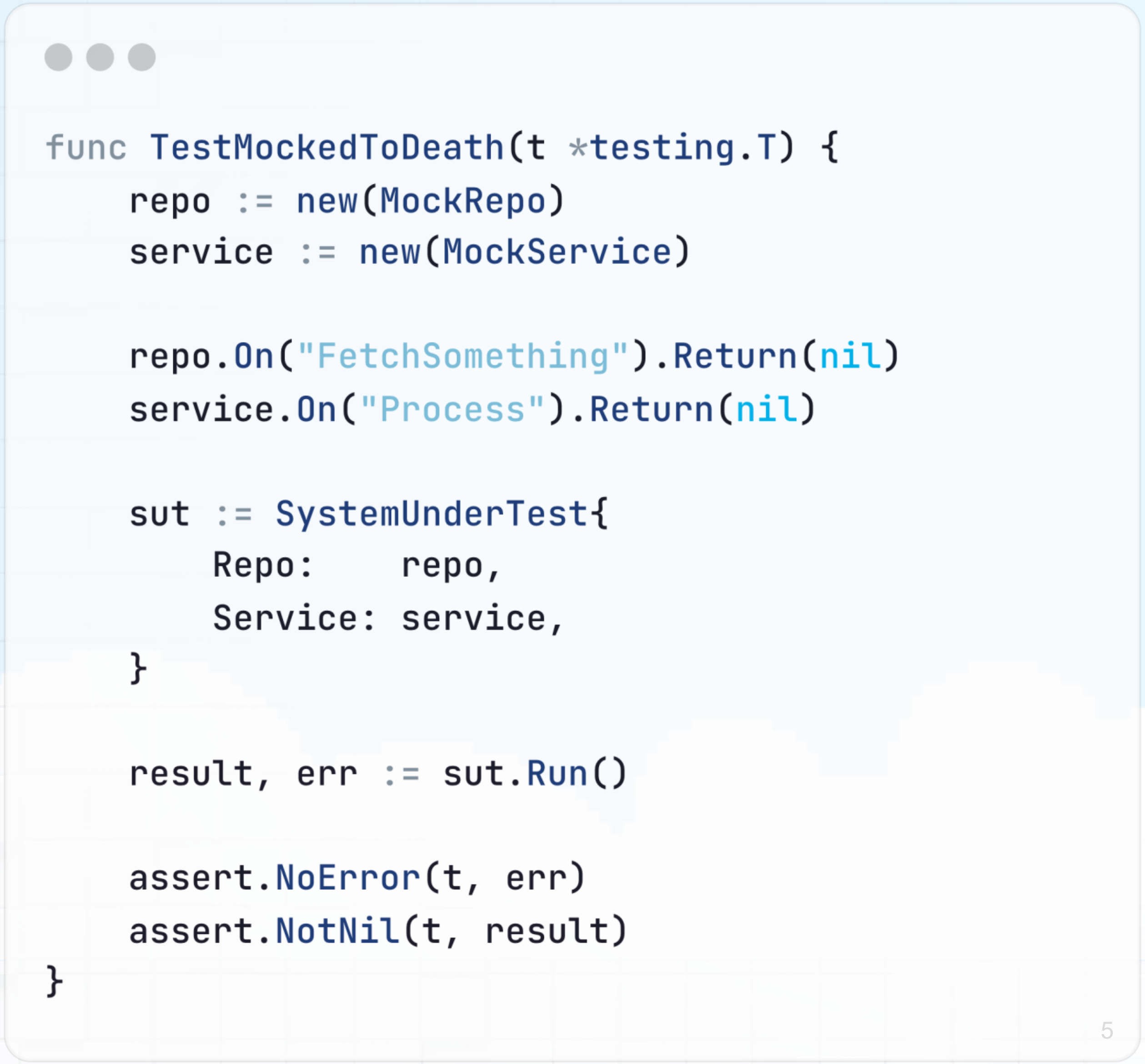
Perception of "tests are good"



CHECK24 / Factory - CarSharing / Carsharing - Carsharingd / Pull requests

TC-1592: write unit tests for provider domain

Simplified Version of the Tests I Found



```go
func TestMockedToDeath(t *testing.T) {
    repo := new(MockRepo)
    service := new(MockService)

    repo.On("FetchSomething").Return(nil)
    service.On("Process").Return(nil)

    sut := SystemUnderTest{
        Repo:    repo,
        Service: service,
    }

    result, err := sut.Run()

    assert.NoError(t, err)
    assert.NotNil(t, result)
}
```

- We mock ***everything***.
- We assert that ***something*** was returned.
- We assert that an error ***didn't*** happen.

{CHECK.conf}

Those Tests Weren't Written to Provide Value

- They were written to **complete the Jira ticket**.
- They were written to **increase the test coverage**.
- They **weren't** written **to highlight any bugs**.
- They were written **just for the sake of writing tests**.

*3 critical bugs dormant in the **covered** code was found later…*

GAME OVER

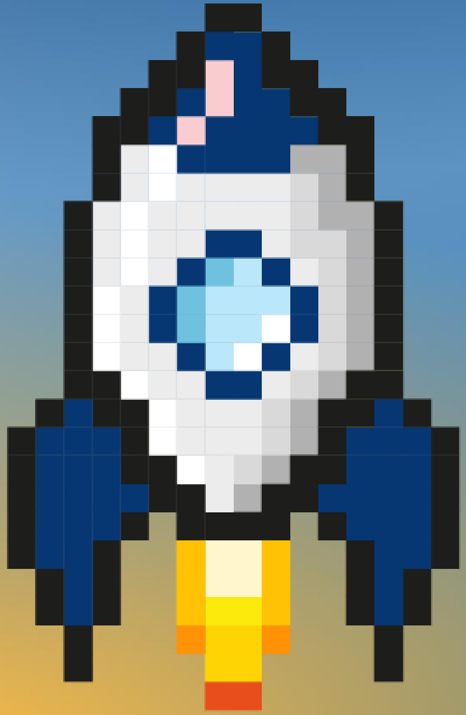
DON'T WRITE TESTS JUST FOR THE SAKE OF IT

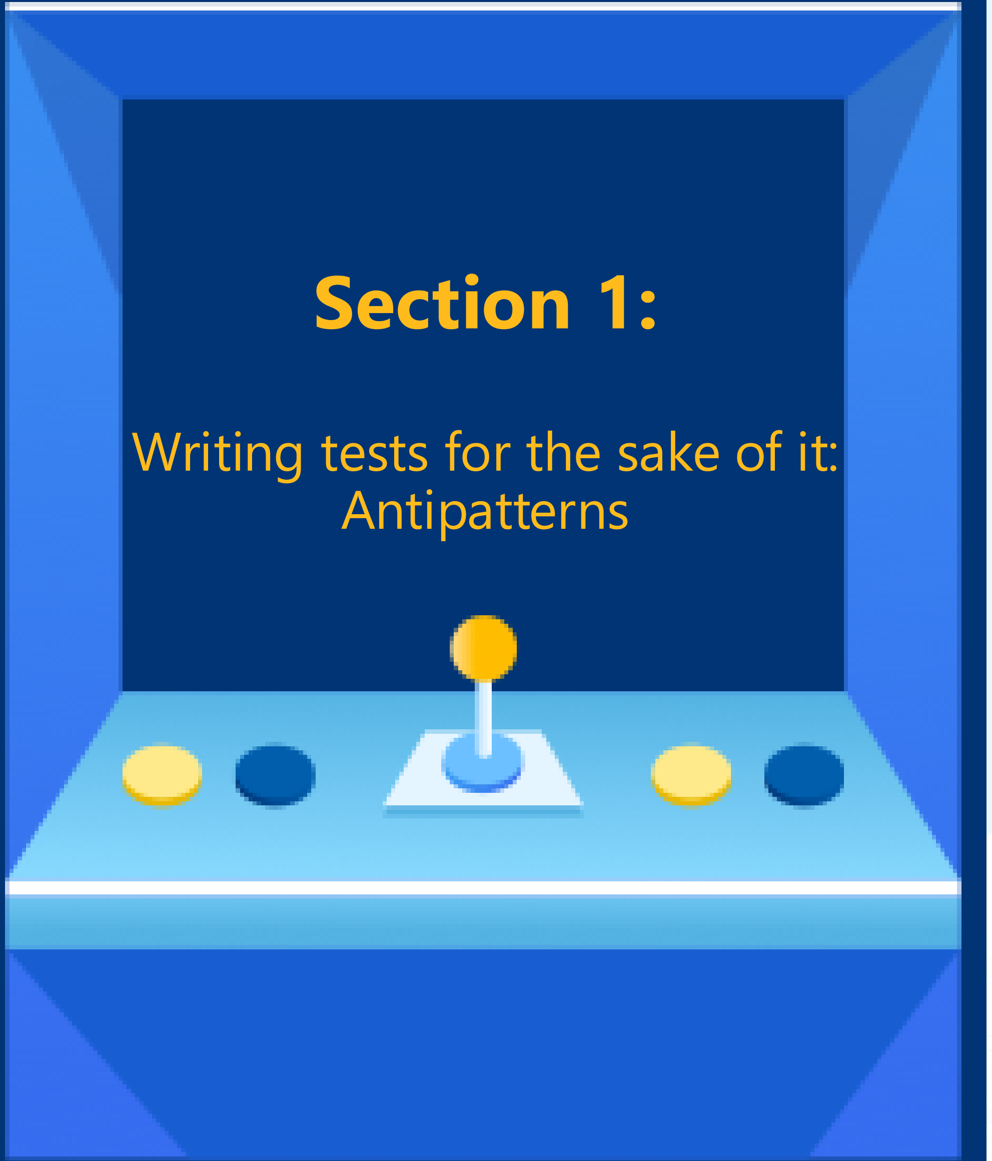
27 June 2025 – 14.45

Celil Yigit

{CHECK.conf}

{AGENDA}





Section 1:

Writing tests for the sake of it: Antipatterns

1. *The Illusion of "Cost-Free" Testing*
2. *More Tests = Better Quality (!)*
3. *Treating Test Coverage as the Goal*
4. *Over-specific Tests*
5. *Sticking to "Easy" Tests*

Antipattern 1: The Illusion of "Cost-Free" Testing

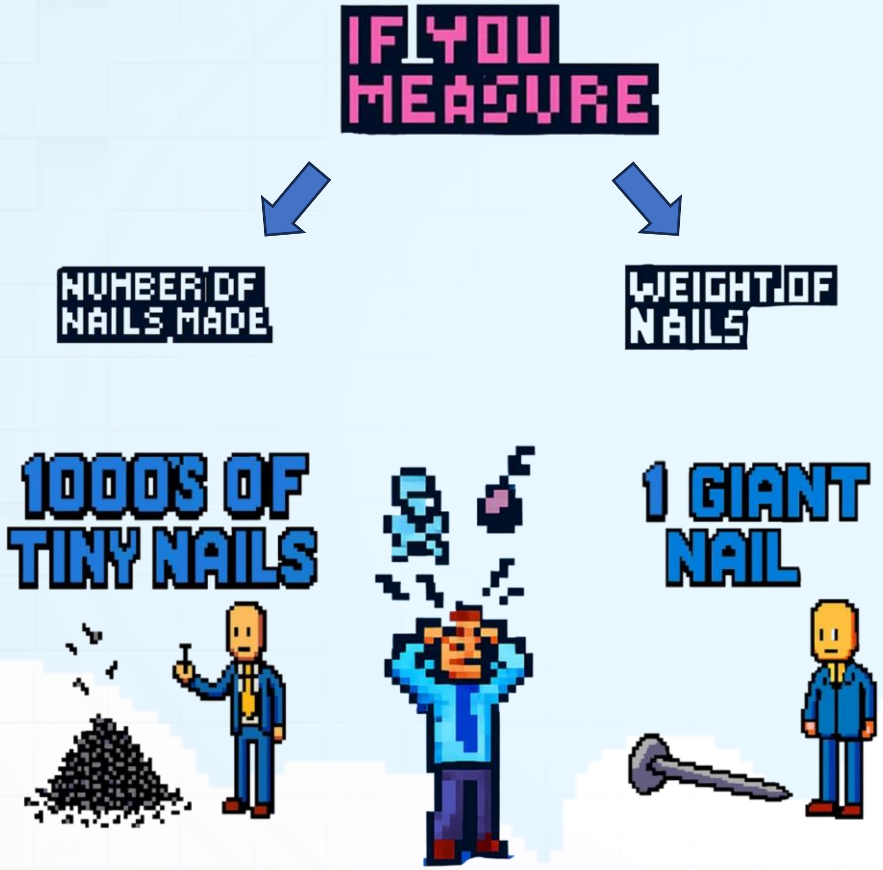
- **Writing Cost:** Initial development effort.
- **Running Cost:** CI/CD time & resources.
- **Maintaining Cost:** Ongoing updates & fixes (often the highest).
- **Understanding Cost:** Cognitive load to interpret & debug.

{CHECK.conf}

🪨 VS 🧈 Antipattern 2: More Tests = Better Quality (!)

- **Assertion Value:** Tests are as valuable as their assertions.
- **Real Confidence:** Derived from quality, not just high test counts.
- **Misleading Metric:** Test count alone says little about effectiveness.

Antipattern 3: Treating Test Coverage as the Goal



- **Indicator, Not Goal:** Coverage shows what's run, not what's verified.
- **Quality Blindspot:** High coverage can mask low-quality or missing assertions.
- **Easy to fake:** Even minimally asserted tests boost coverage.

Antipattern 4: Over-specific Tests

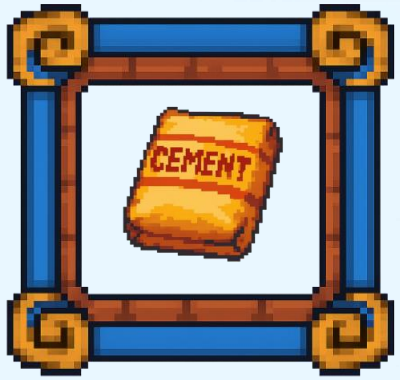
- **Testing every detail:** Testing the static strings, already tested libraries
- **Check Code, Not the Contract:** Tests focus on how things are done, not what is delivered.
- **Tests Fear Change:** Tests resist or break upon any change.

Antipattern 5: Sticking to "Easy" Tests

- **Value in Layers:** Integration/E2E tests catch different, critical bugs.
- **Isolation Limits:** Unit tests alone can't verify component interactions.
- **Complexity Tradeoff:** Harder tests often provide higher confidence.

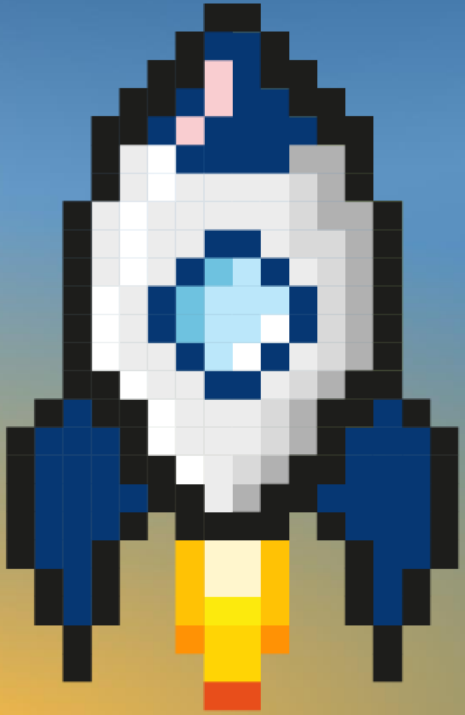
{CHECK.conf}

We covered 5 types of Testing Antipatterns

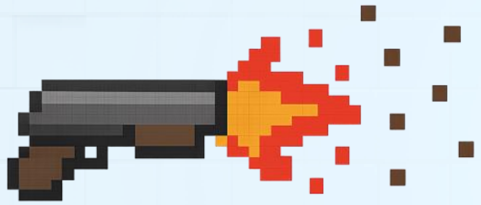


Let's see their consequences in CHECK24

{AGENDA}



1. *140 Tests Later... CI/CD Was On Fire.*



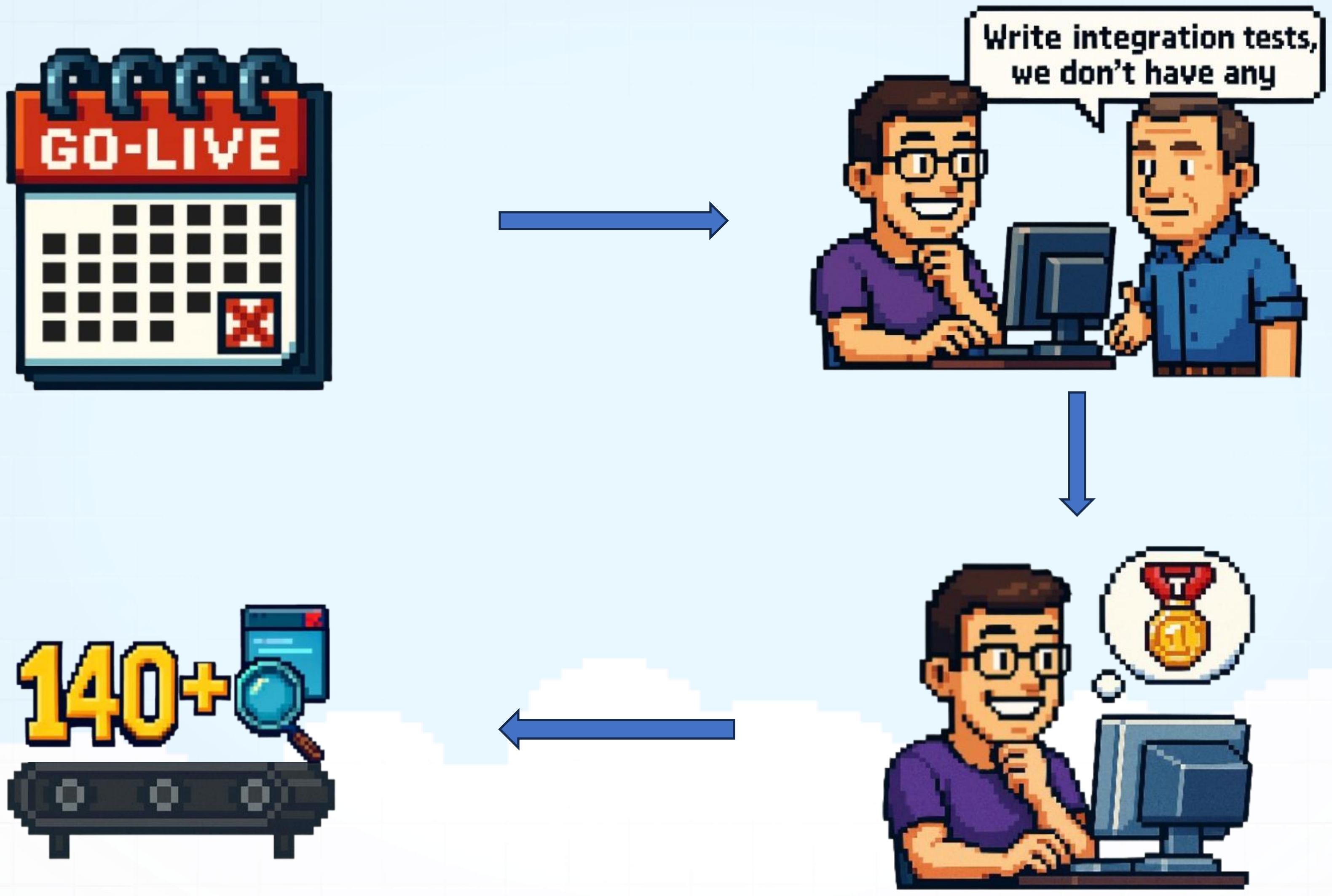
2. *One Symbol. 300 Test Failures.*

3. *27,000 Tests. Green CI. Broken App.*

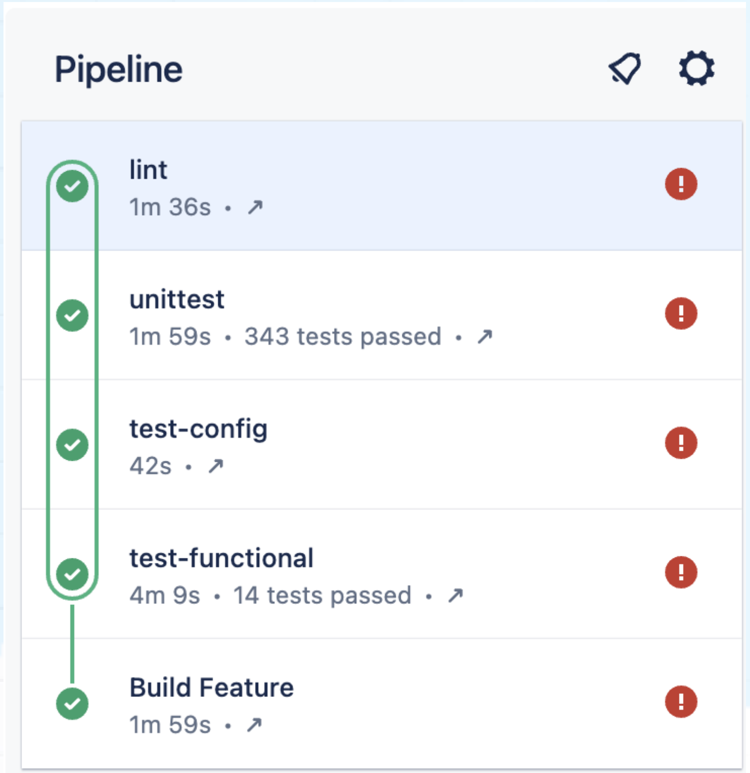
Section 2:

Three Stories: Results of these Antipatterns in CHECK24

140 Tests Later… CI/CD Was On Fire



140 Tests Later... CI/CD Was On Fire


Pipeline

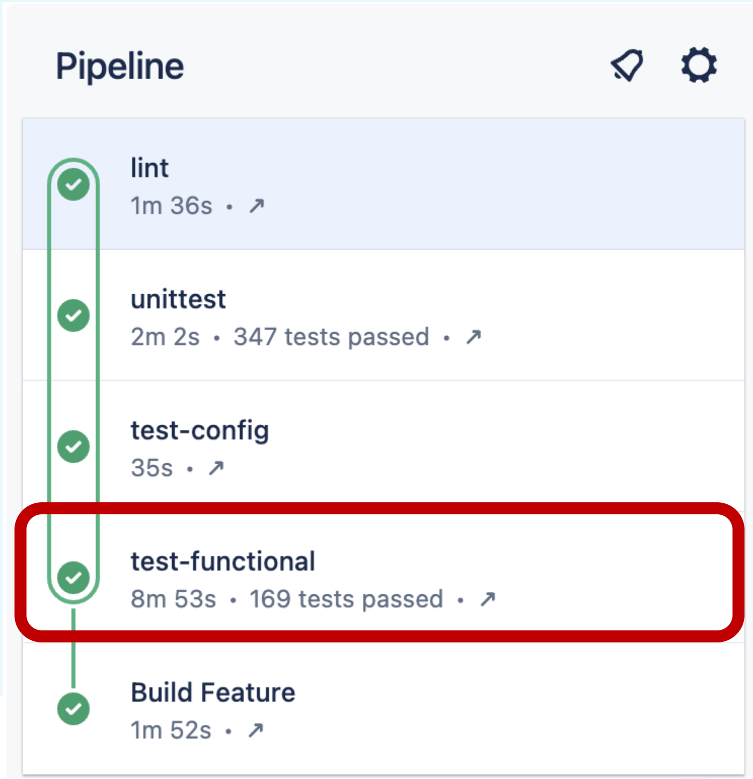
lint
1m 36s • ↗

unittest
1m 59s • 343 tests passed • ↗

test-config
42s • ↗

test-functional
4m 9s • 14 tests passed • ↗

Build Feature
1m 59s • ↗


Pipeline

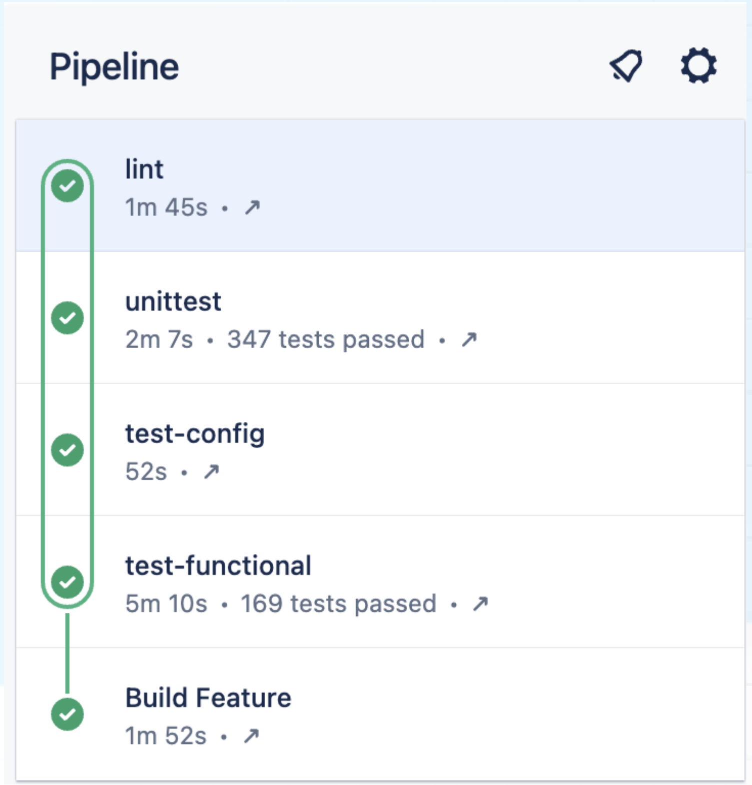
lint
1m 36s • ↗

unittest
2m 2s • 347 tests passed • ↗

test-config
35s • ↗

test-functional
8m 53s • 169 tests passed • ↗

Build Feature
1m 52s • ↗


Pipeline

lint
1m 45s • ↗

unittest
2m 7s • 347 tests passed • ↗

test-config
52s • ↗

test-functional
5m 10s • 169 tests passed • ↗

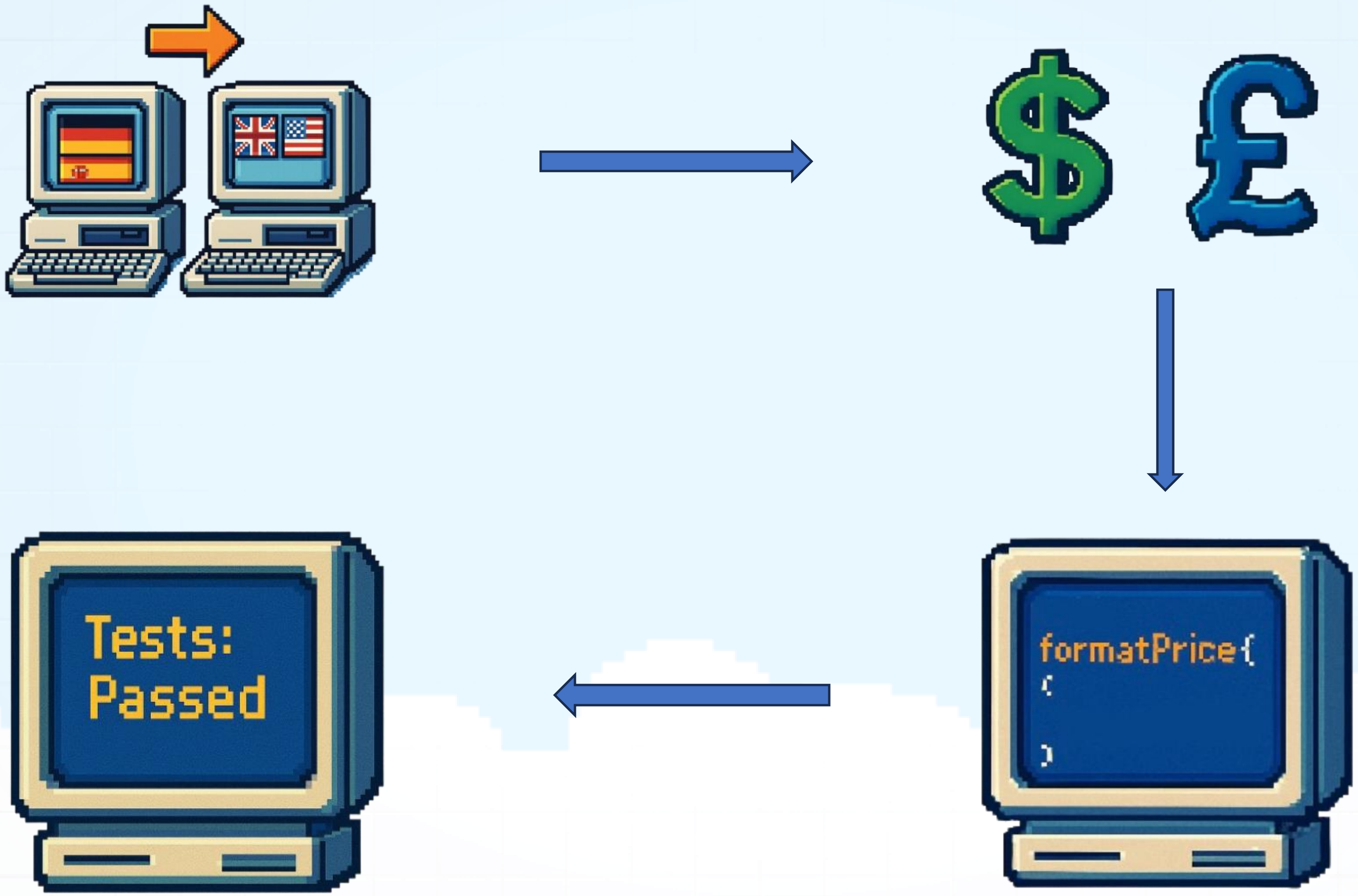
Build Feature
1m 52s • ↗

{CHECK.conf}

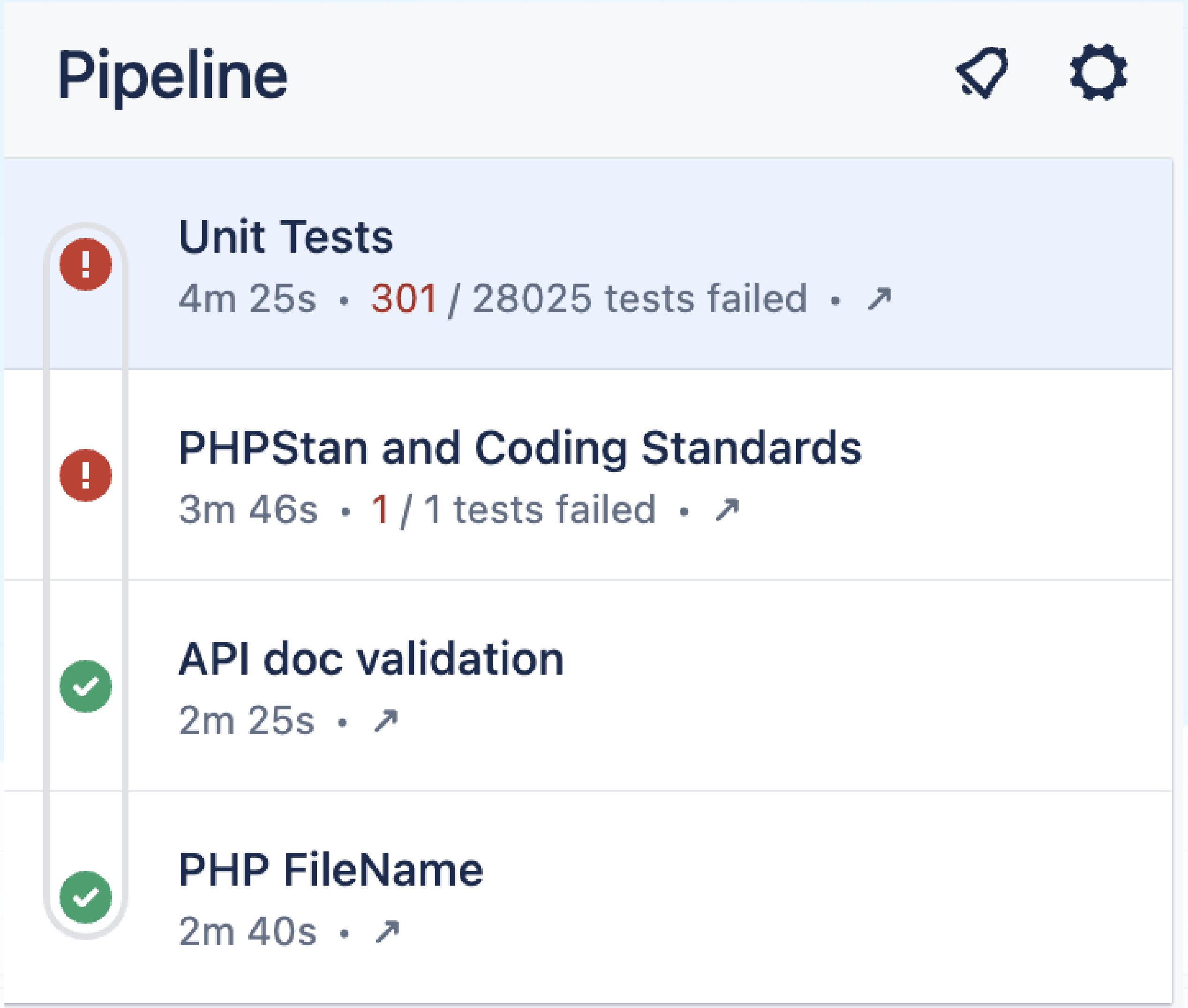
🔥 140 Tests Later... CI/CD Was On Fire

{CHECK.conf}

One Change. 300 Test Failures:



One Change. 300 Test Failures:


Pipeline

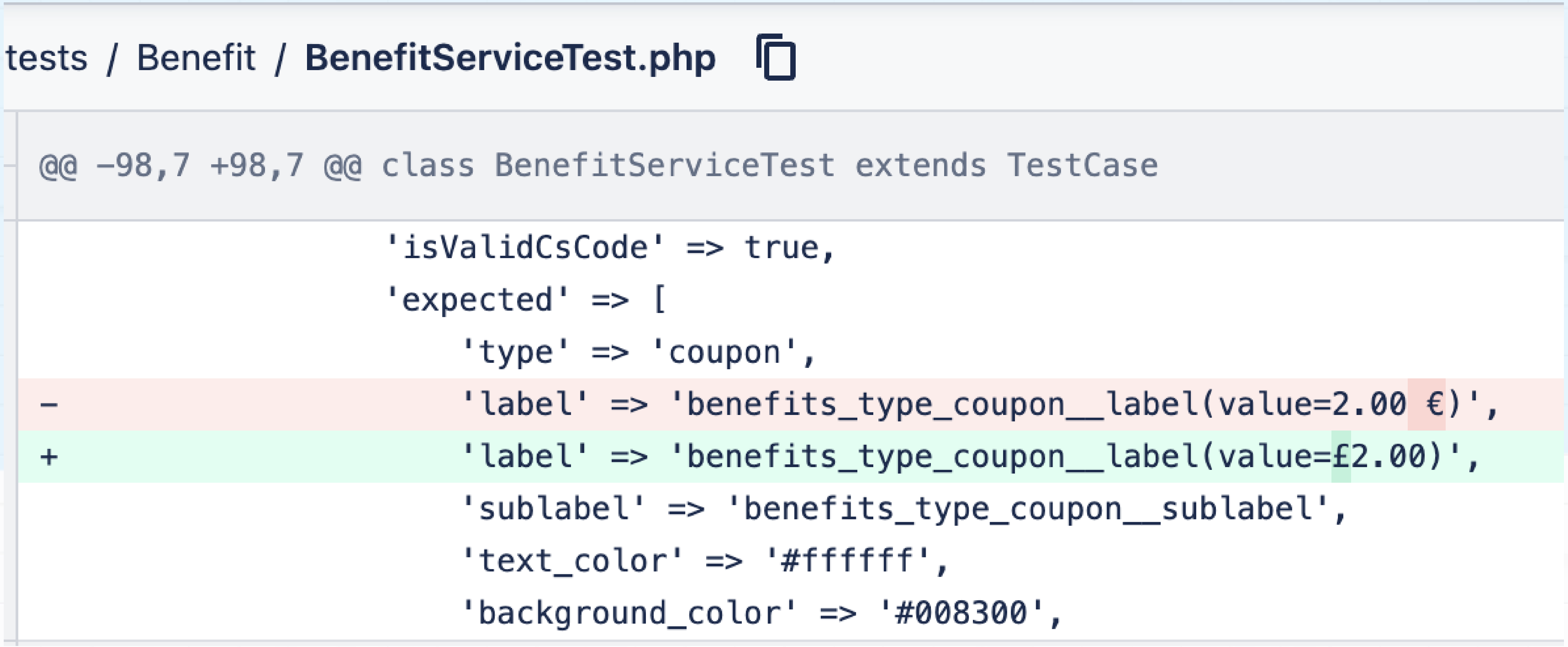
Unit Tests
4m 25s • 301 / 28025 tests failed • ↗

PHPStan and Coding Standards
3m 46s • 1 / 1 tests failed • ↗

API doc validation
2m 25s • ↗

PHP FileName
2m 40s • ↗

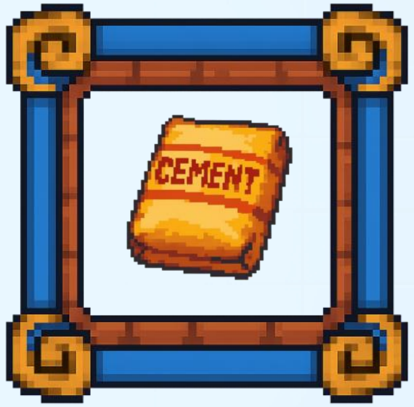
One Change. 300 Test Failures:



tests / Benefit / **BenefitServiceTest.php**

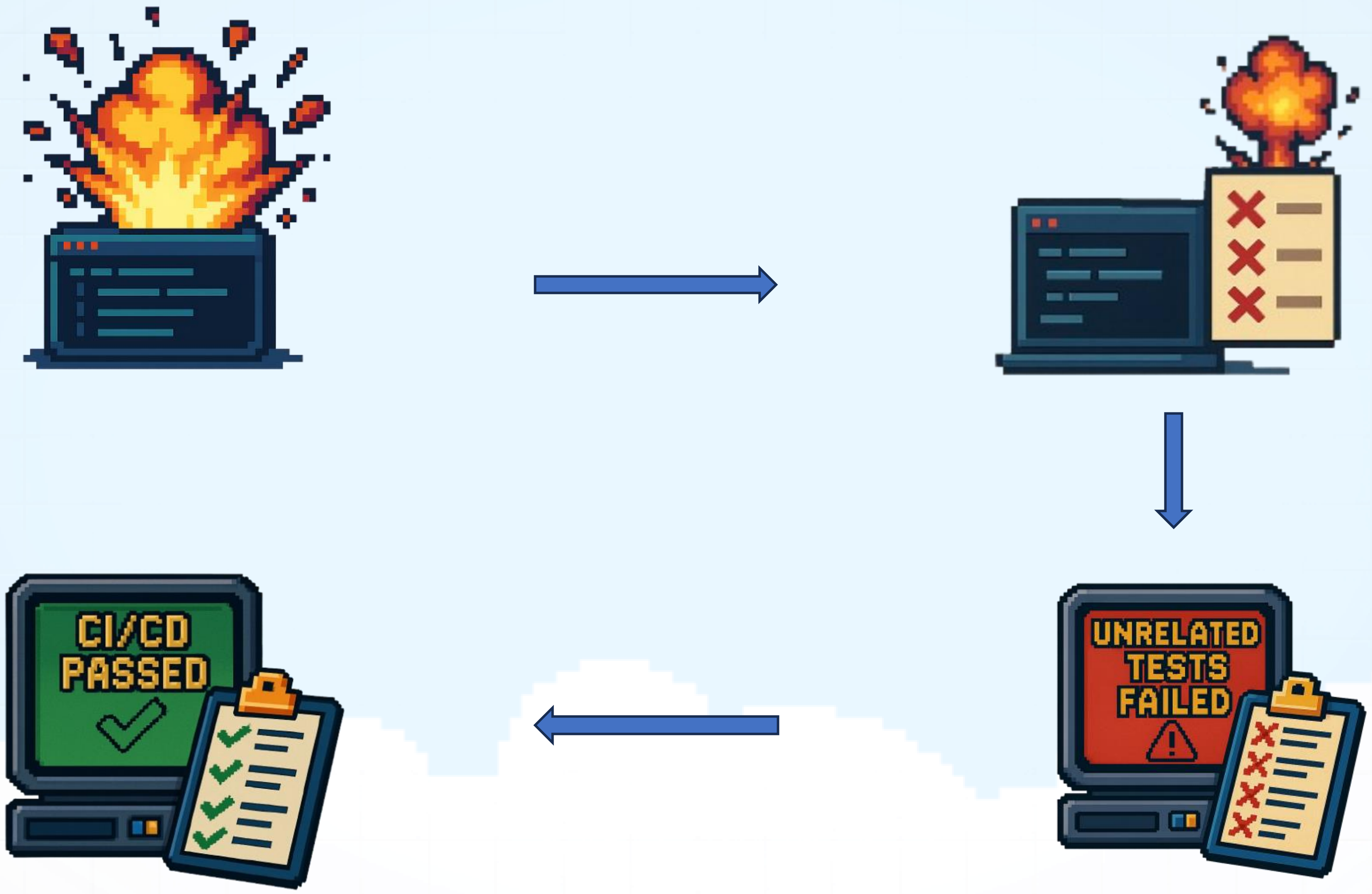
```
@@ -98,7 +98,7 @@ class BenefitServiceTest extends TestCase
                    'isValidCsCode' => true,
                    'expected' => [
                        'type' => 'coupon',
-                       'label' => 'benefits_type_coupon__label(value=2.00 €)',
+                       'label' => 'benefits_type_coupon__label(value=£2.00)',
                        'sublabel' => 'benefits_type_coupon__sublabel',
                        'text_color' => '#ffffff',
                        'background_color' => '#008300',
```

One Change. 300 Test Failures:

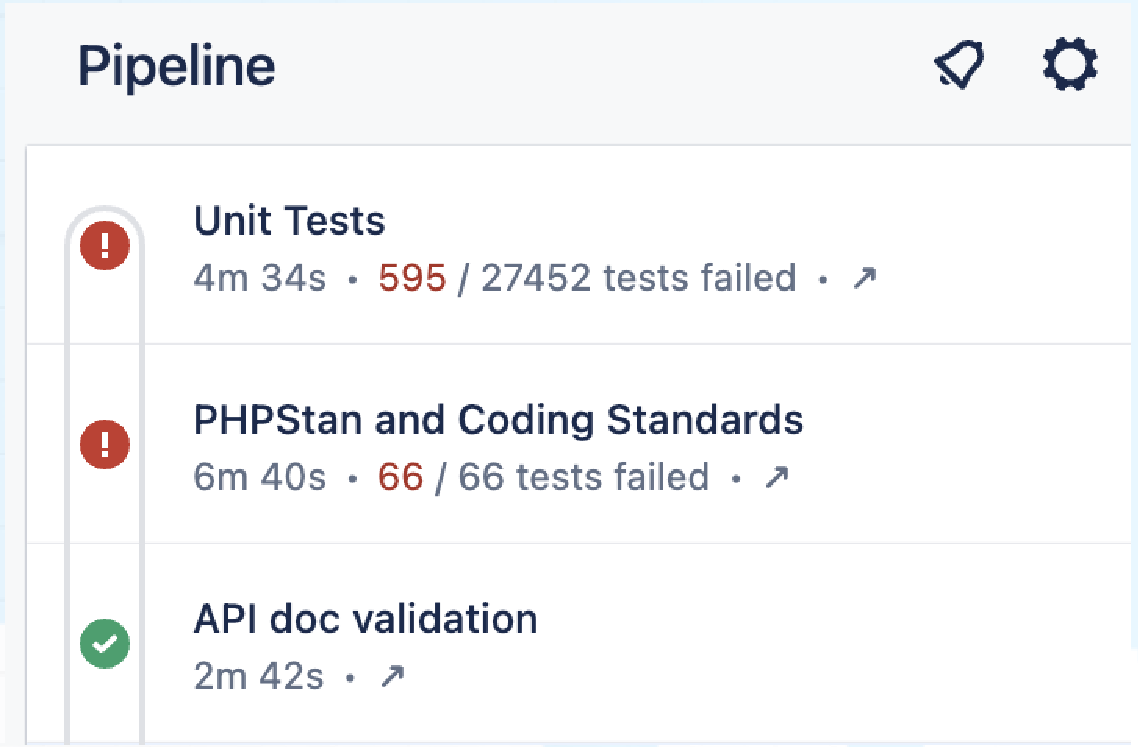


{CHECK.conf}

27,000 Tests. Green CI. Broken App.



27,000 Tests. Green CI. Broken App.

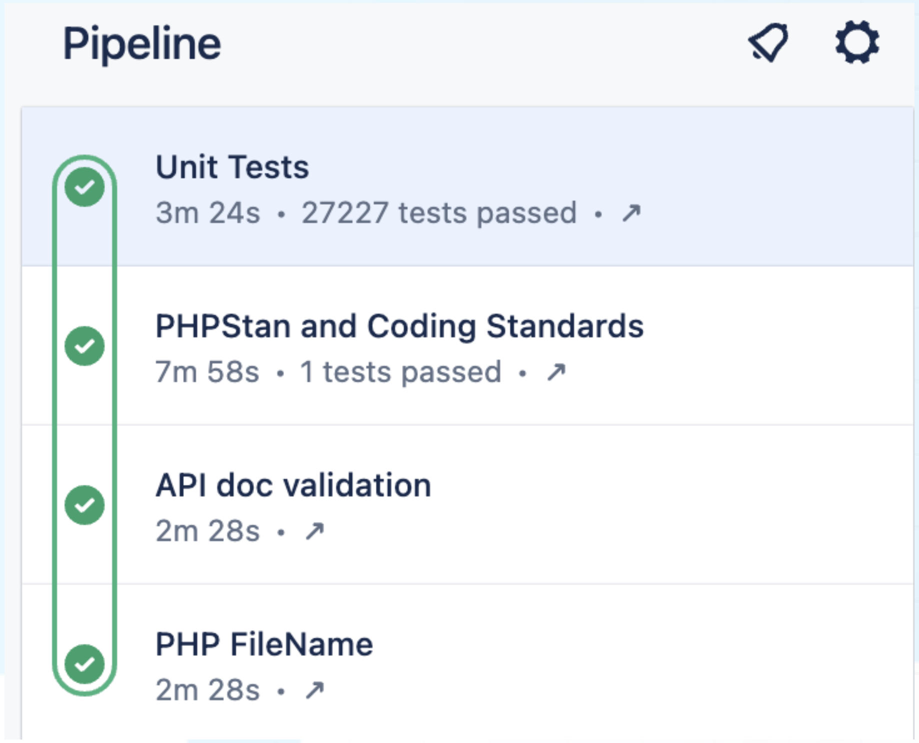


Pipeline

Unit Tests
4m 34s • 595 / 27452 tests failed • ↗

PHPStan and Coding Standards
6m 40s • 66 / 66 tests failed • ↗

API doc validation
2m 42s • ↗



Pipeline

Unit Tests
3m 24s • 27227 tests passed • ↗

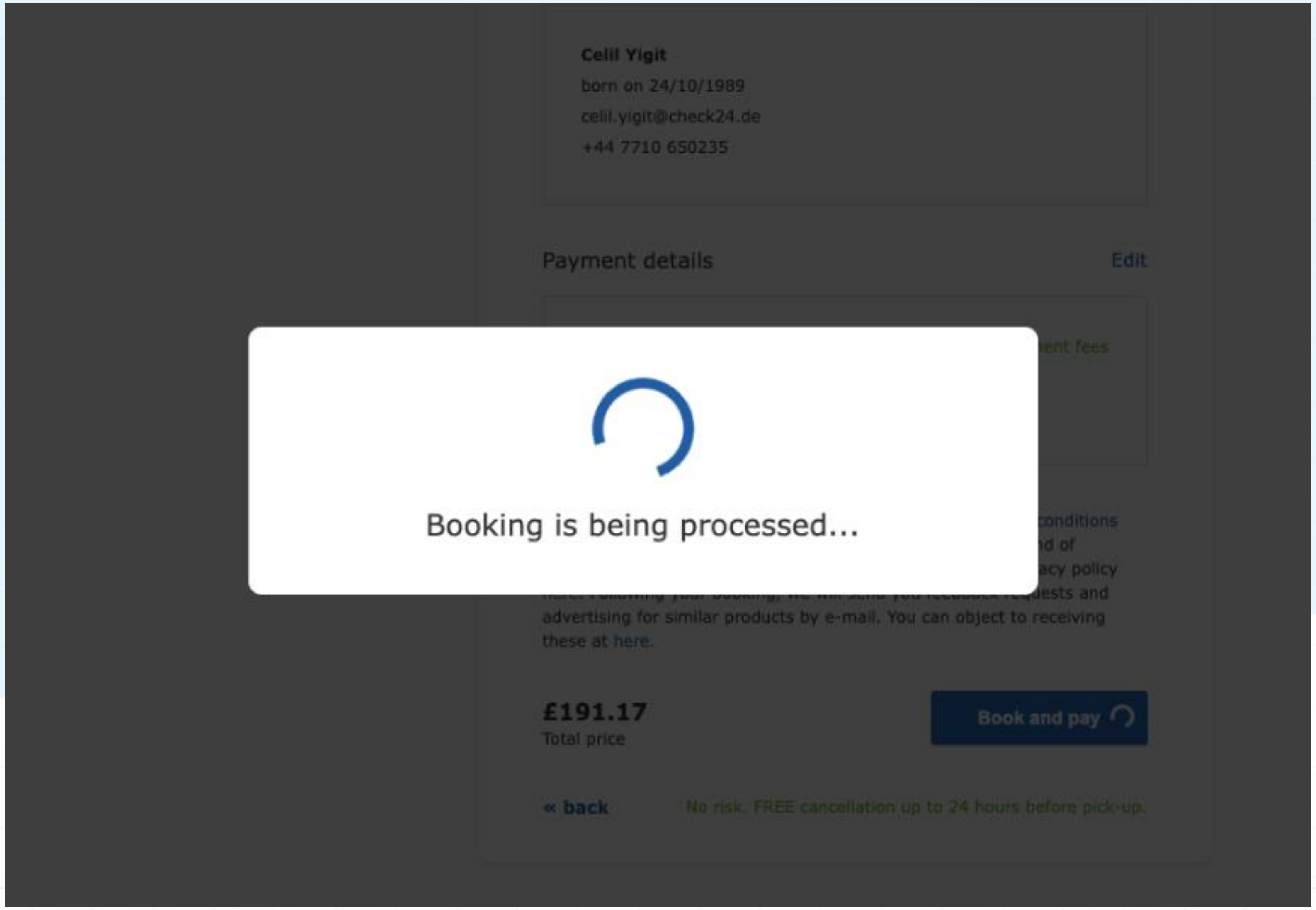
PHPStan and Coding Standards
7m 58s • 1 tests passed • ↗

API doc validation
2m 28s • ↗

PHP FileName
2m 28s • ↗

{CHECK.conf}

27,000 Tests. Green CI. Broken App.



Celil Yigit
born on 24/10/1989
celil.yigit@check24.de
+44 7710 650235

Payment details Edit

Booking is being processed...

advertising for similar products by e-mail. You can object to receiving these at here.

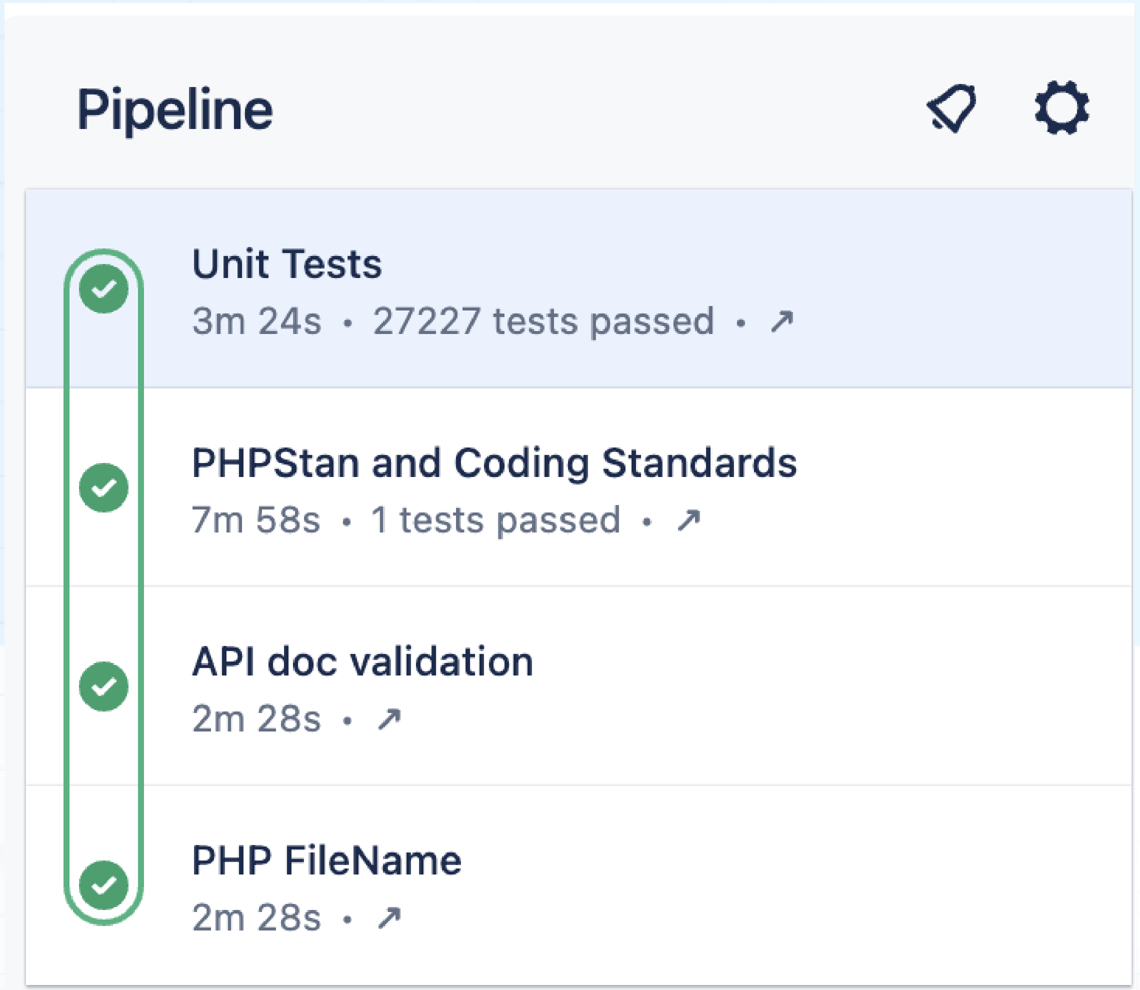
£191.17
Total price

Book and pay

« back

No risk. FREE cancellation up to 24 hours before pick-up.

27,000 Tests. Green CI. Broken App.

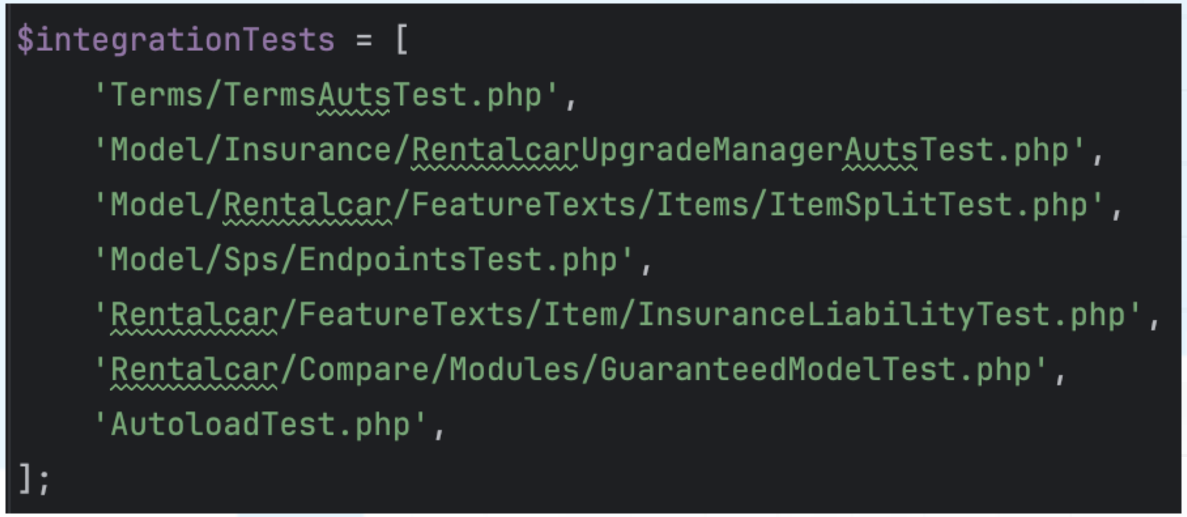

Pipeline

Unit Tests
3m 24s • 27227 tests passed • ↗

PHPStan and Coding Standards
7m 58s • 1 tests passed • ↗

API doc validation
2m 28s • ↗

PHP FileName
2m 28s • ↗


```php
$integrationTests = [
    'Terms/TermsAutsTest.php',
    'Model/Insurance/RentalcarUpgradeManagerAutsTest.php',
    'Model/Rentalcar/FeatureTexts/Items/ItemSplitTest.php',
    'Model/Sps/EndpointsTest.php',
    'Rentalcar/FeatureTexts/Item/InsuranceLiabilityTest.php',
    'Rentalcar/Compare/Modules/GuaranteedModelTest.php',
    'AutoloadTest.php',
];
```

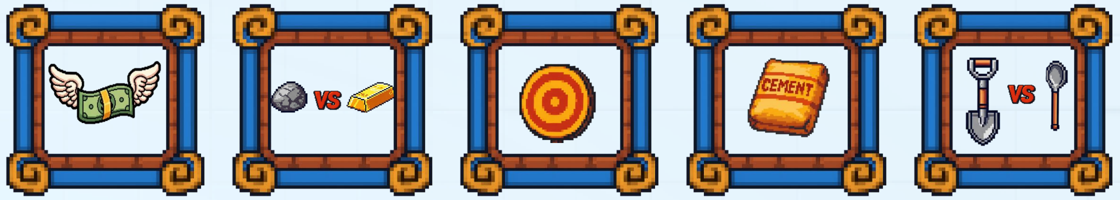
44 Integration Tests in Total

{CHECK.conf}

27,000 Tests. Green CI. Broken App.

{CHECK.conf}

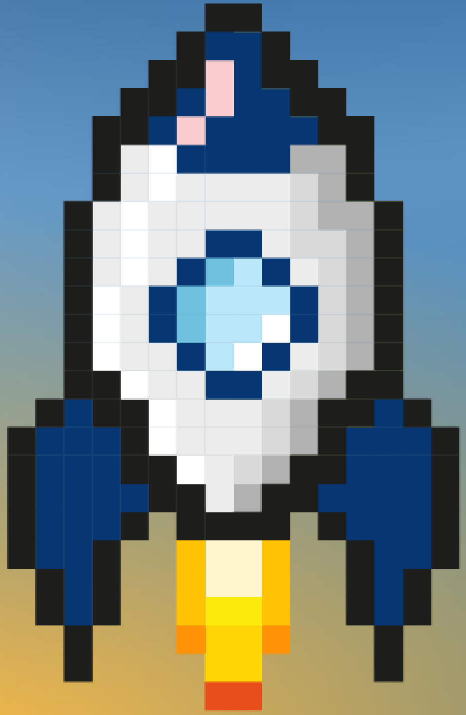
We Talked About Antipatterns



We Talked About Their Effects in CHECK24

Let's Talk About What We Can Do to Our Tests to LEVEL UP

{AGENDA}

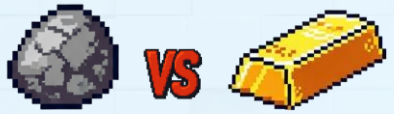


01 **Writing tests for the sake of it: Antipatterns**
Misconceptions / Bad habits that lead to bad test suites.

02 **Three Stories: Results of these Antipatterns in CHECK24**
False sense of security, bloating pipelines, and problem in refactoring

03 **Revisiting Antipatterns: Leveling up our tests**
Clear strategies to write fewer, better, high-value tests

1. *Have control over test costs!*

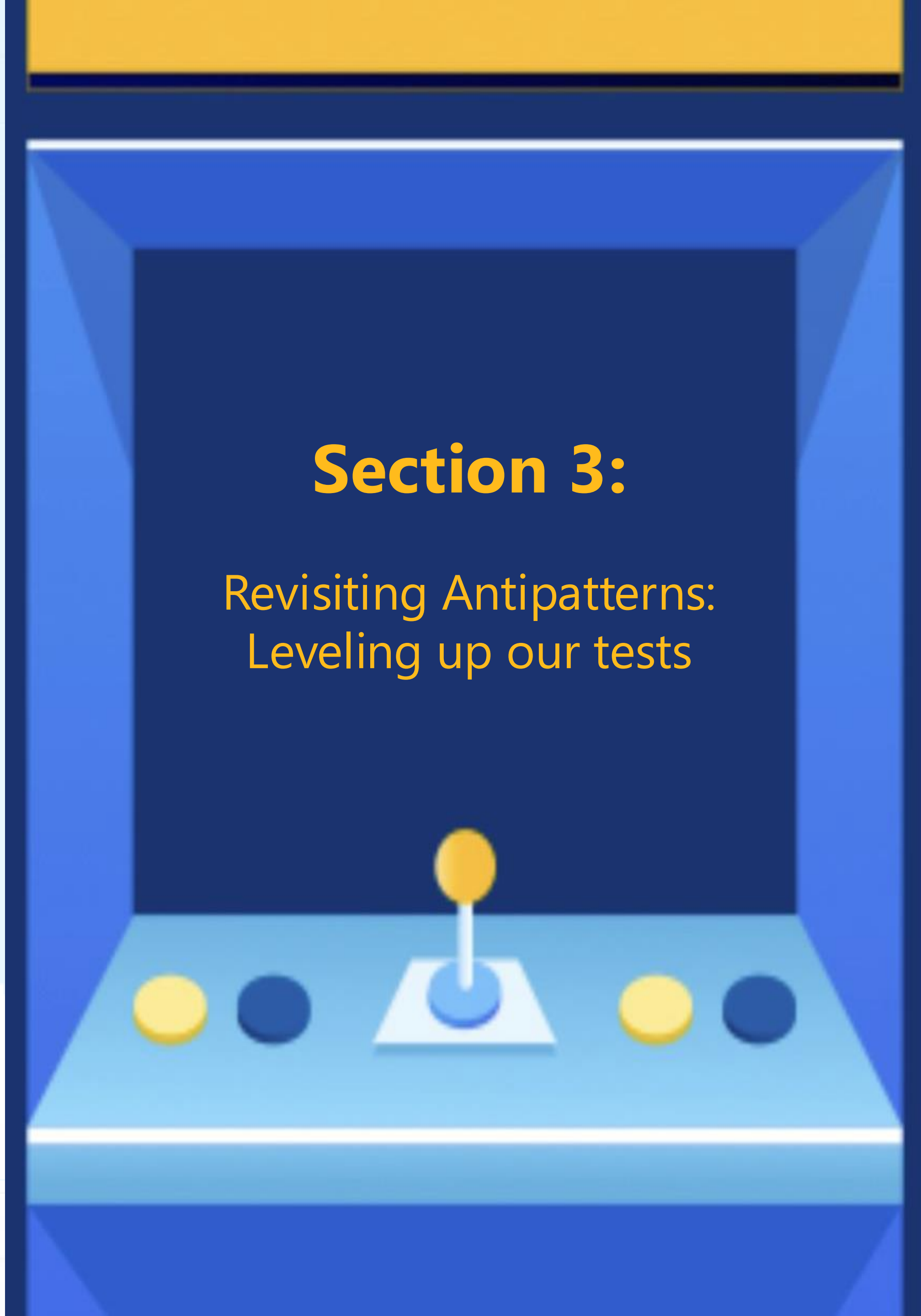


2. *Focus on Quality, not the Quantity*

3. *Use Test Coverage the right way*

4. *Adaptable Tests for Evolving Code*

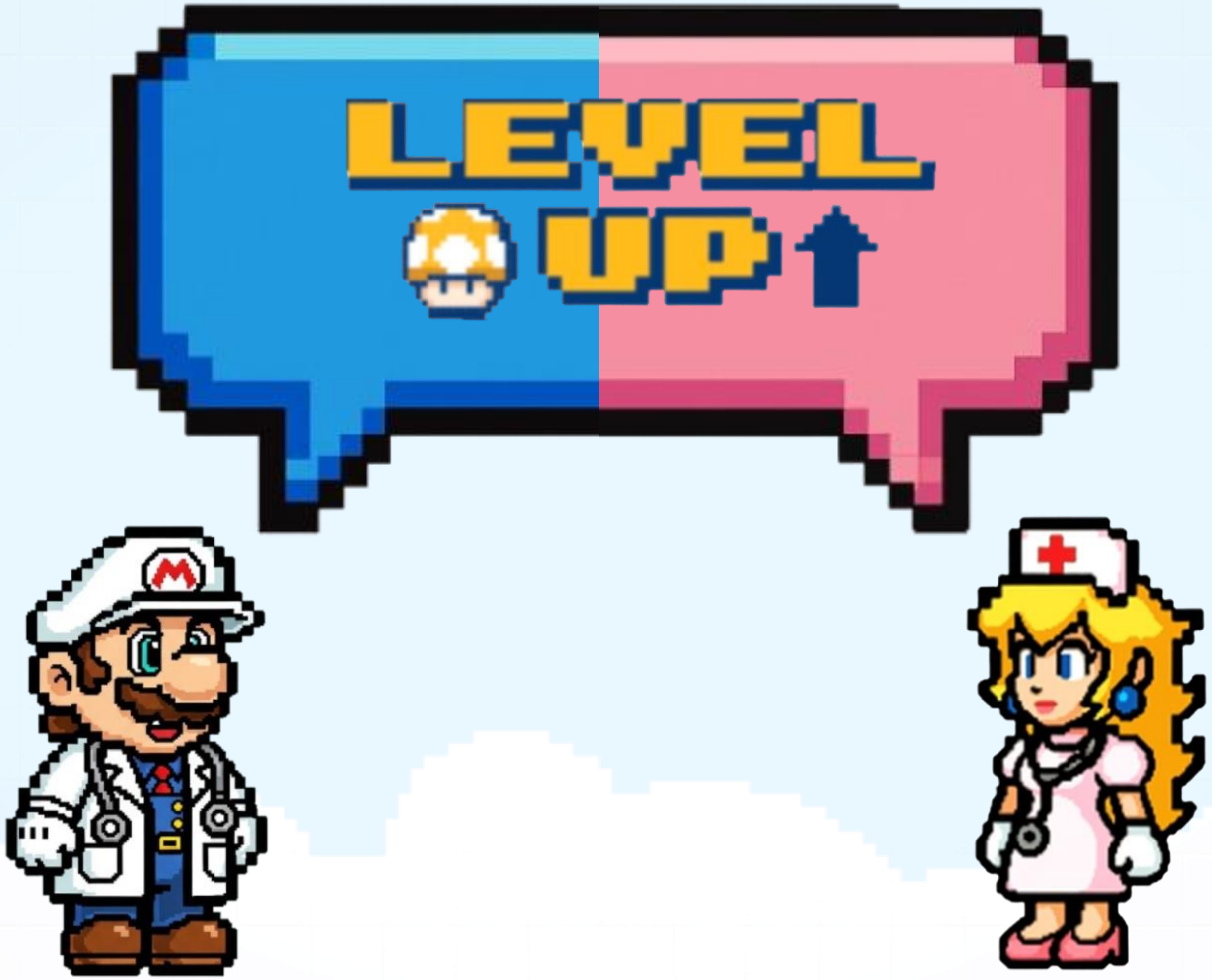
5. *Leverage Every Test Type*



Section 3:

Revisiting Antipatterns:
Leveling up our tests

Level Up Testing with Doctor Mario and Nurse Peach



Diagnosis: The Illusion of "Cost-Free" Testing

Do your CI/CD pipelines run longer than your coffee breaks?

*Ever thought, "**these tests cost more trouble than they're worth**"?*

Level Up: Have control over test costs!

WRITING COST
RUNNING COST
COGNITIVE COST
MAINTENANCE COST

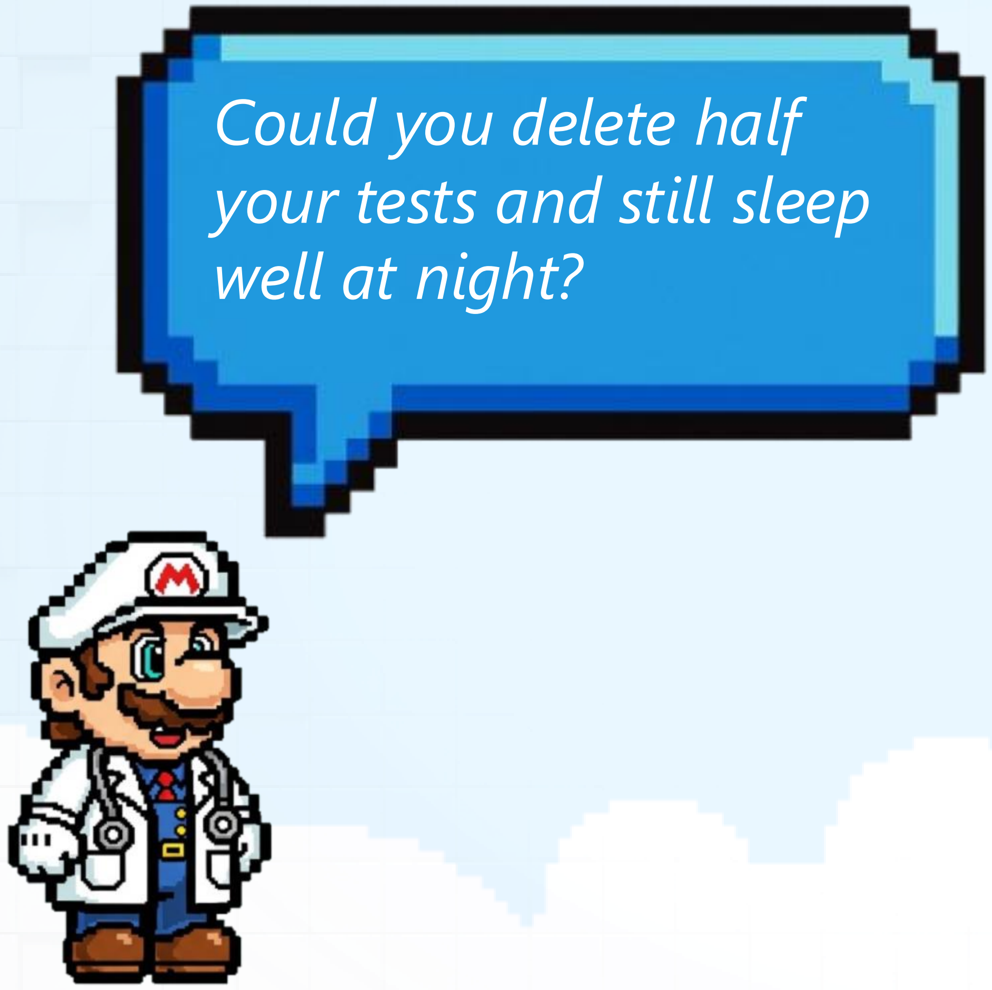
Consider all cost dimensions

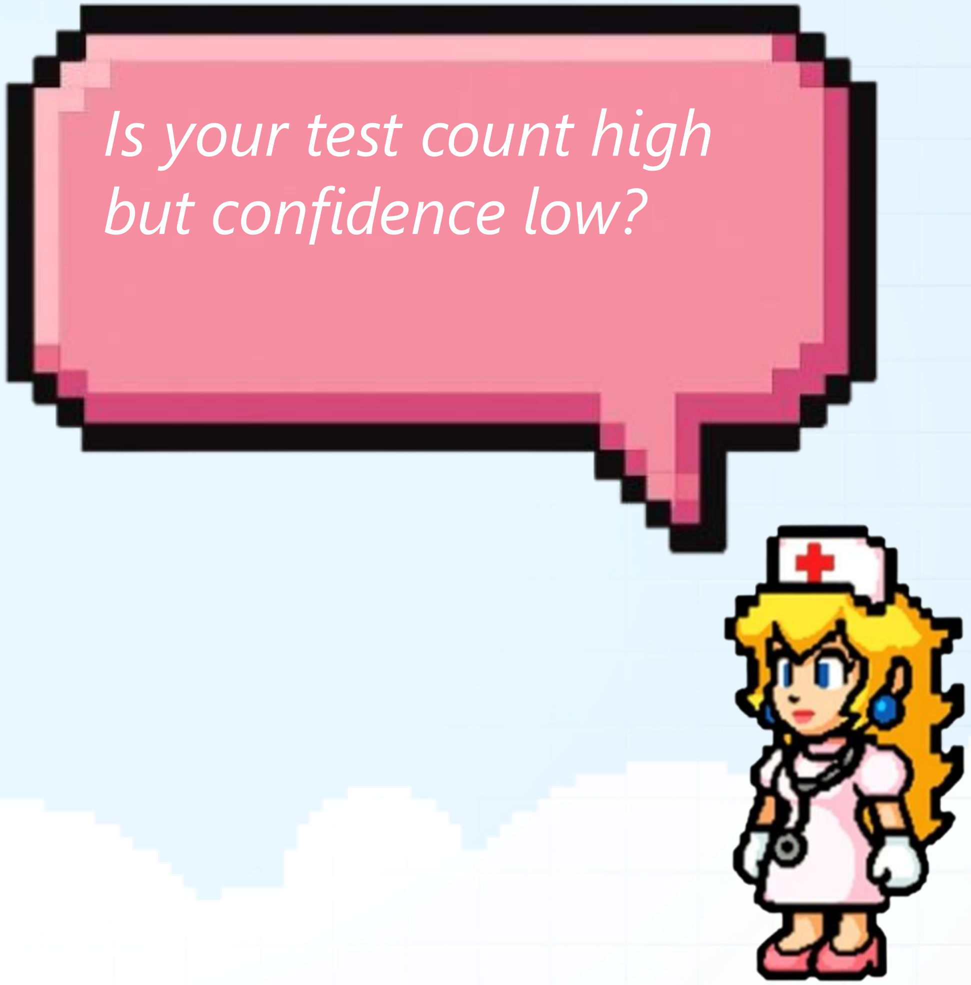
Define clear performance metrics

Be brave – Prune your tests

{CHECK.conf}

Diagnosis: More Tests = Better Quality (!)

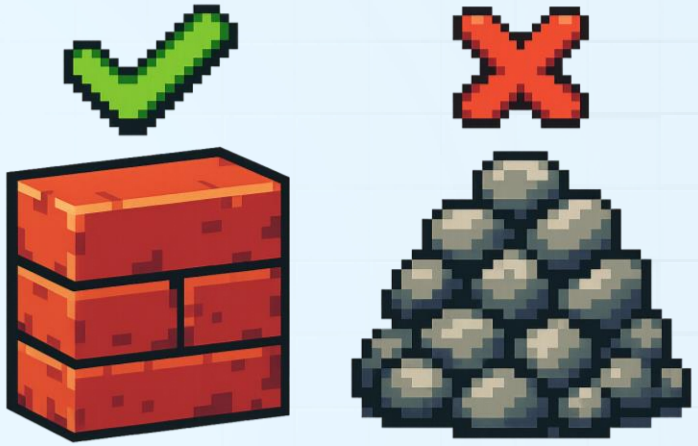

Could you delete half your tests and still sleep well at night?



Level Up: Focus on Quality, not the Quantity

Always focus on assertion quality

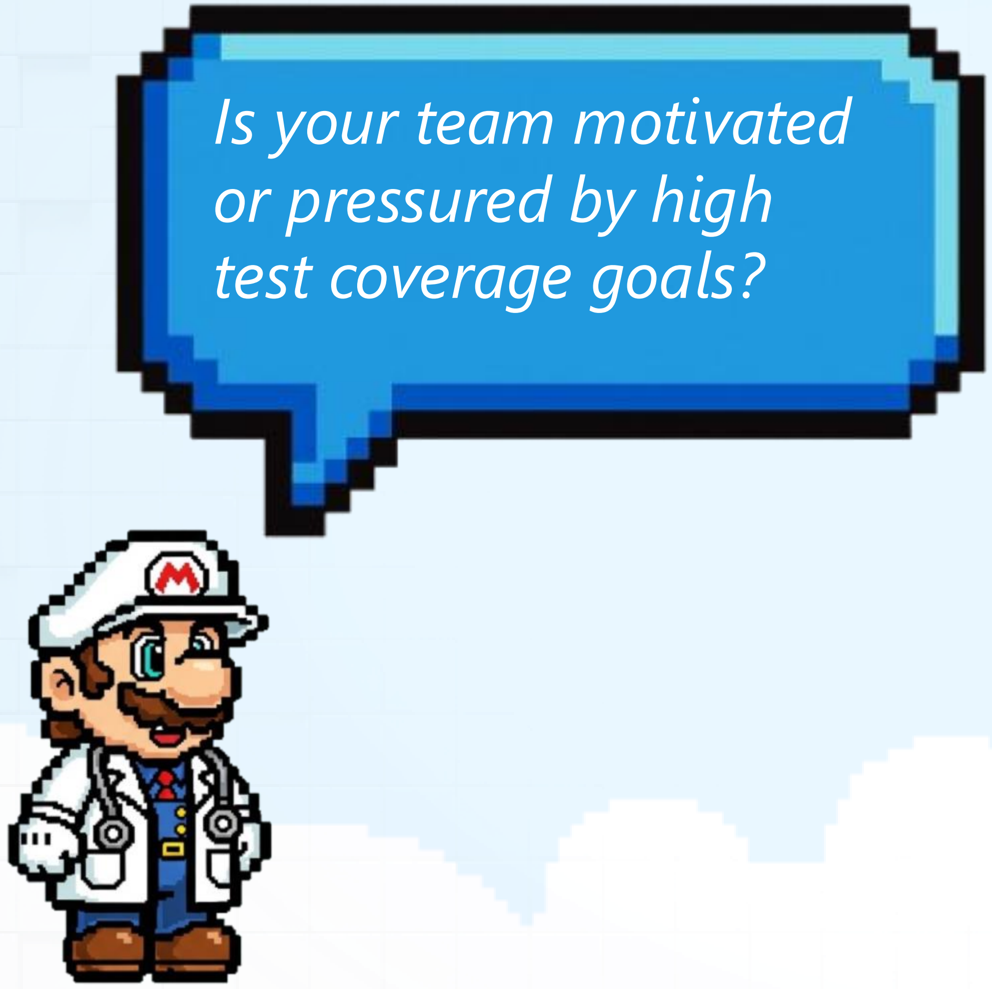
Have a clear goal/assertion in mind



Fewer, smarter tests for the same confidence

{CHECK.conf}

Diagnosis: Treating Test Coverage as the Goal


Is your team motivated or pressured by high test coverage goals?

Have you ever written a test just to make a coverage number go up?

{CHECK.conf}

Level Up: Use Test Coverage the right way

Acknowledge Test Coverage is a map, not a trophy

Try not to employ a rigid coverage goal

Inspect bug leakages from "covered" code

{CHECK.conf}

Diagnosis: Over-specific Tests

Do you have "Refactoring Anxiety" because of how many tests will break?

Are your tests sensitive to formatting changes in output strings?

{CHECK.conf}

Level Up: Adaptable Tests for Evolving Code

Test public APIs, treat internals as black-box

Assert the key data only, not static strings

Use builders for flexible test data

{CHECK.conf}

Diagnosis: Sticking to "Easy" Tests

Do your tests run in perfect isolation, while your app breaks in integration?

Is your test pyramid more of a pancake or a spike?

{CHECK.conf}

Level Up: Leverage Every Test Type

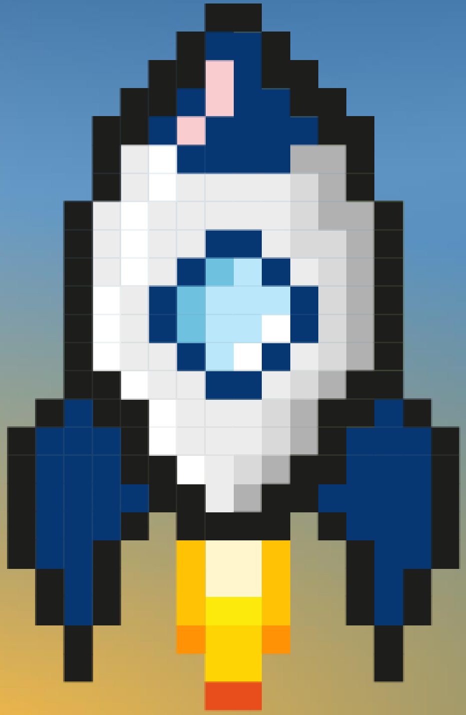
Test Types Collaborate, Not Compete

E2E Tests: Costly Upfront, Priceless in Reliability

Testing individual units is never enough

{CHECK.conf}

{AGENDA}



01 Writing tests for the sake of it: Ant

Misconceptions / Bad habits that uites.

02 T s: Results ns in CHECK24

security s, and problem in

03 Rev up our tests

Clear st , better, high-value tests

{CHECK.conf}

Final Takeaways

- Prioritize ***BEHAVIOR*** & ***VALUE***, Not Just Coverage or Count.
- Treat Tests as ***CRITICAL CODE***: Design, Review, Maintain.
- If a Test Offers ***NO REAL CONFIDENCE***, Be Brave: ***DELETE IT***.
- Strategically Test ***INTEGRATIONS***, Not Just Isolated Units.
- *Don't write tests just for the sake it!*

{CHECK.conf}

QUESTIONS & ANSWERS

Vielen Dank für Ihre Aufmerksamkeit