Micro-service delivery - without the pitfalls

Seb Rose
Mastodon: @sebrose@mastodon.scot
Twitter: @sebrose
Blog: https://claysnow.co.uk
E-mail: seb@claysnow.co.uk

Agenda

Introduction
Why software contracts?
Traditional software testing
A taxonomy of contracts (partial)
Handrolled contract tests
Documenting the contract
Micro-services (finally)
Pact, PactBroker and more
“Bidirectional” contract testing
Questions

Micro-service architecture …

… a collection of services that are:

- Independently deployable
- Loosely coupled
- Organised around business capabilities
- Owned by a small team
- Highly maintainable and testable

Benefits

The micro-service architecture enables the rapid, frequent and reliable delivery of large, complex applications. It also enables an organization to evolve its technology stack.

Consumer challenges

Consumer challenges

The
micro-service I depend
on isn't ready

Consumer challenges

The micro-service I depend on isn't ready

The micro-service I depend on doesn't provide detailed documentation

Consumer challenges

The micro-service I depend on isn’t ready

The micro-service I depend on doesn’t provide detailed documentation

I’m not confident the new version of the micro-service will behave exactly the same?

Provider challenges

Provider challenges

We want to update our micro-service, but don't know who is using it

Provider challenges

We want to update our micro-service, but don't know who is using it

We want to fix a defect in our micro-service, but don't know who is depending on it

Provider challenges

We want to update our micro-service, but don't know who is using it

We don't want to manage multiple test instances of our micro-service

We want to fix a defect in our micro-service, but don't know who is depending on it

So what?

If you can’t deploy services independently,
you don’t have micro-services.

Beth Skurrie

So what?

If you can’t deploy services independently, you don’t have micro-services.

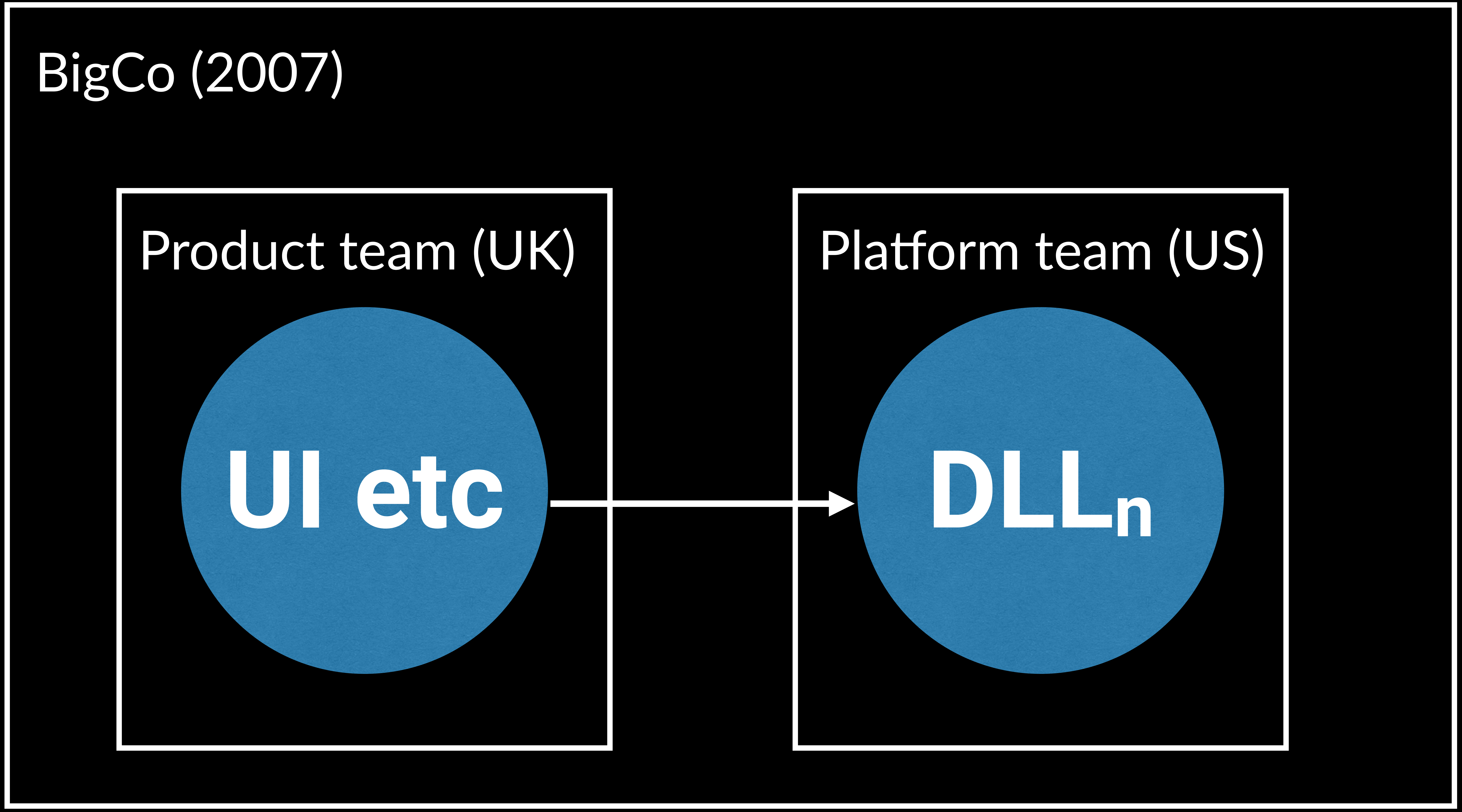
You have a distributed monolith

Beth Skurrie

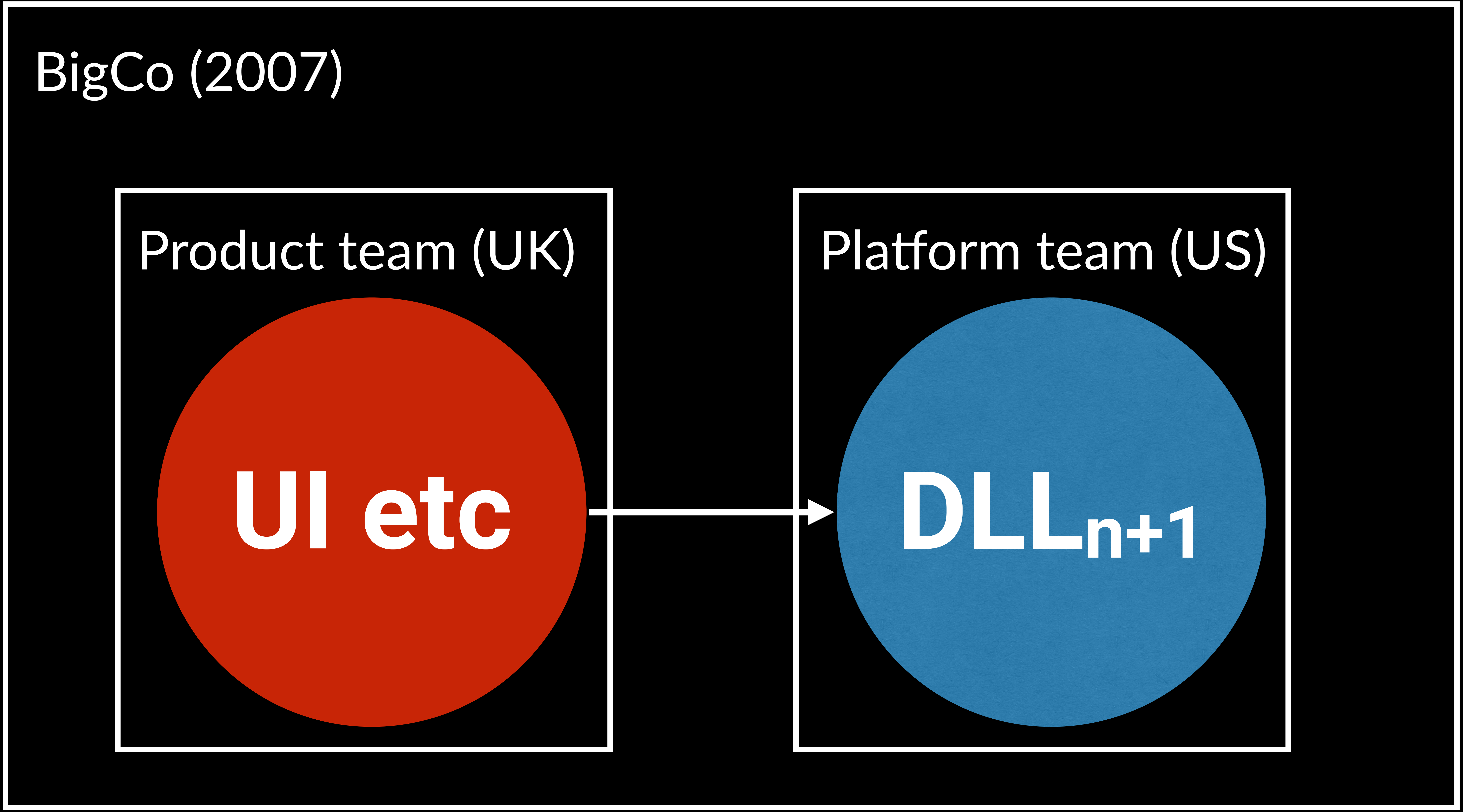
Agenda

Introduction
Why software contracts?
Traditional software testing
A taxonomy of contracts (partial)
Handrolled contract tests
Documenting the contract
Micro-services (finally)
Pact, PactBroker and more
“Bidirectional” contract testing
Questions

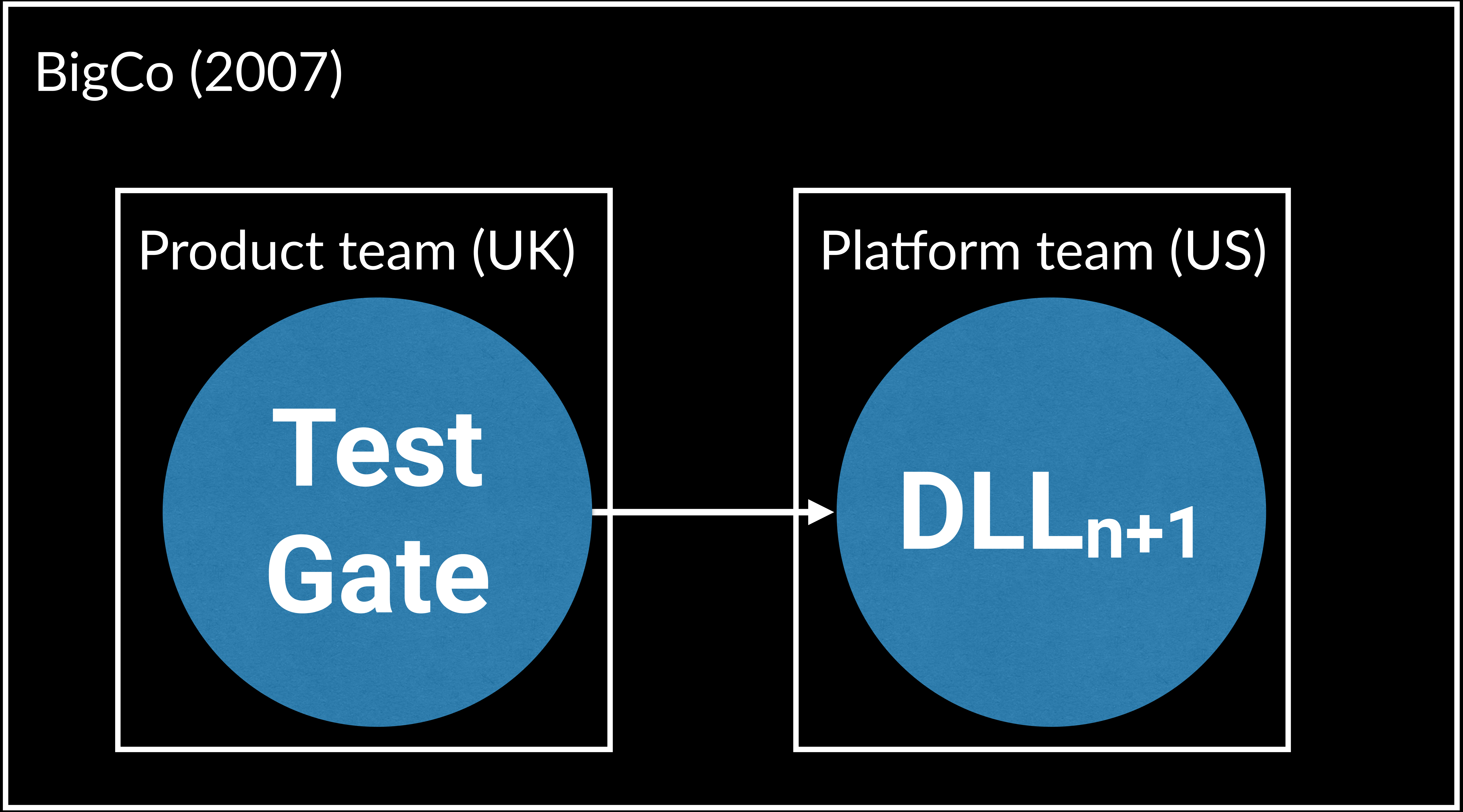
Once upon a time ...



… we couldn't trust DLL updates



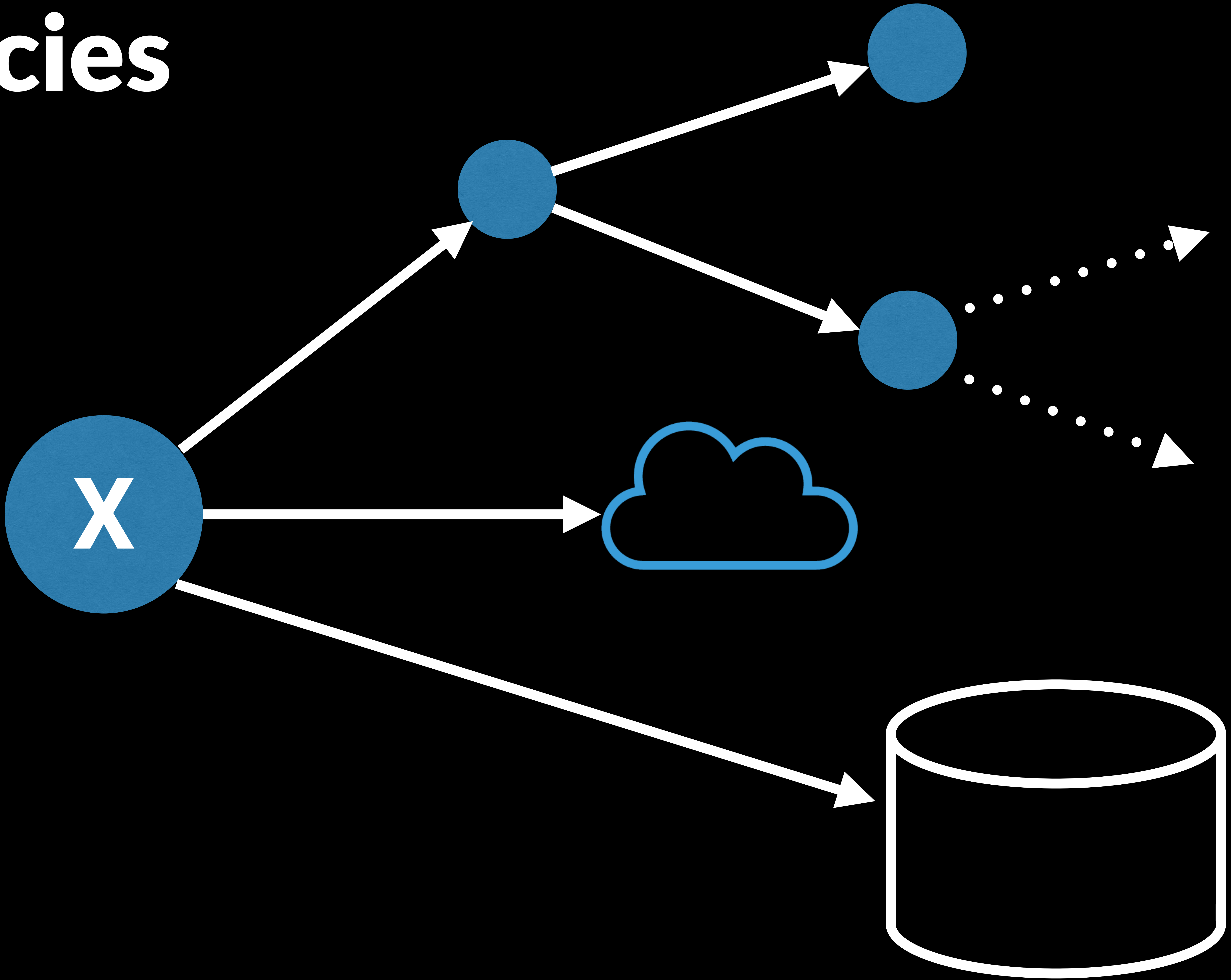
Defensive test suite



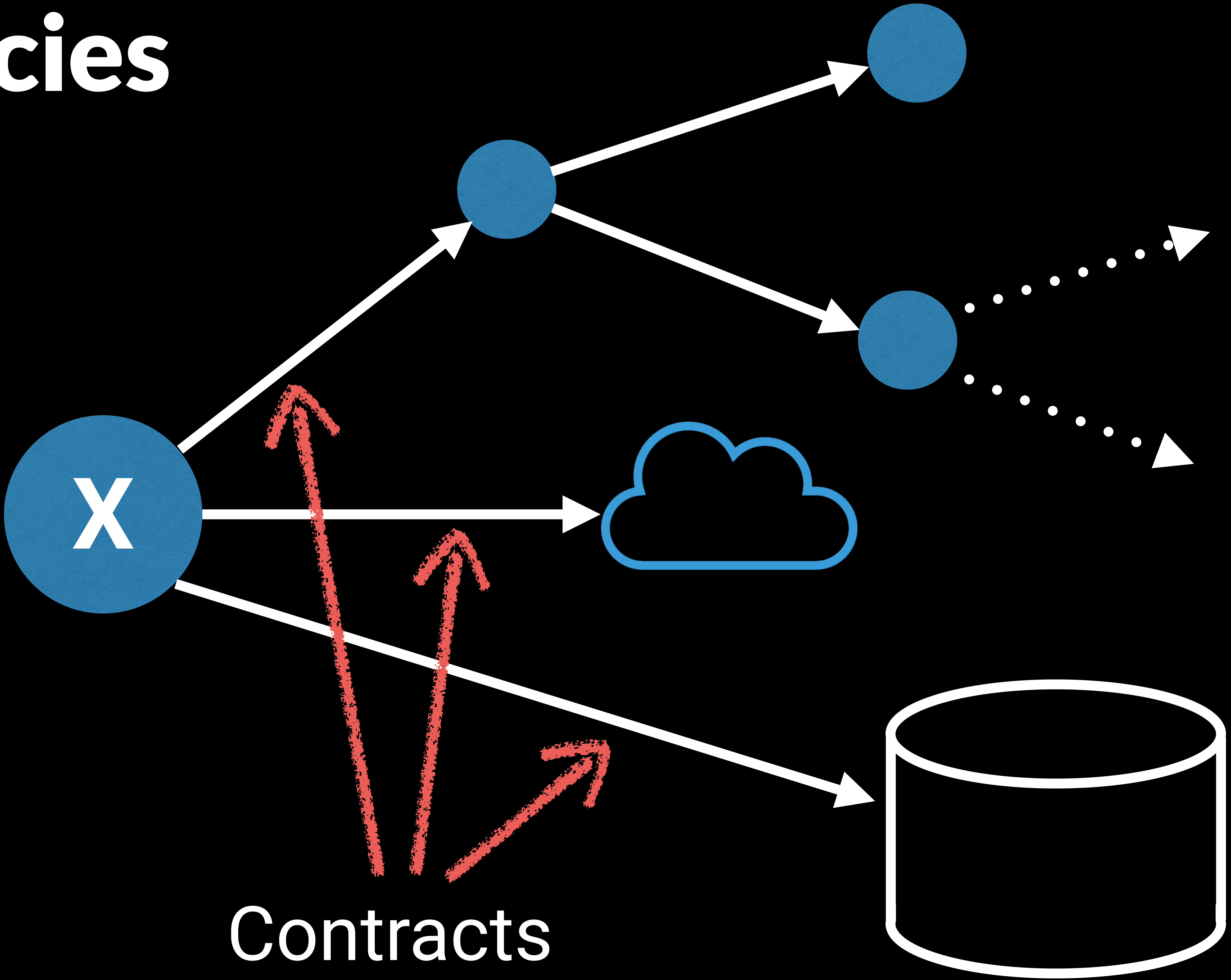
Consequences

- Some data changes between runs
- Failures were a signal for communication
- Product team became more productive
- Contracts became clearer

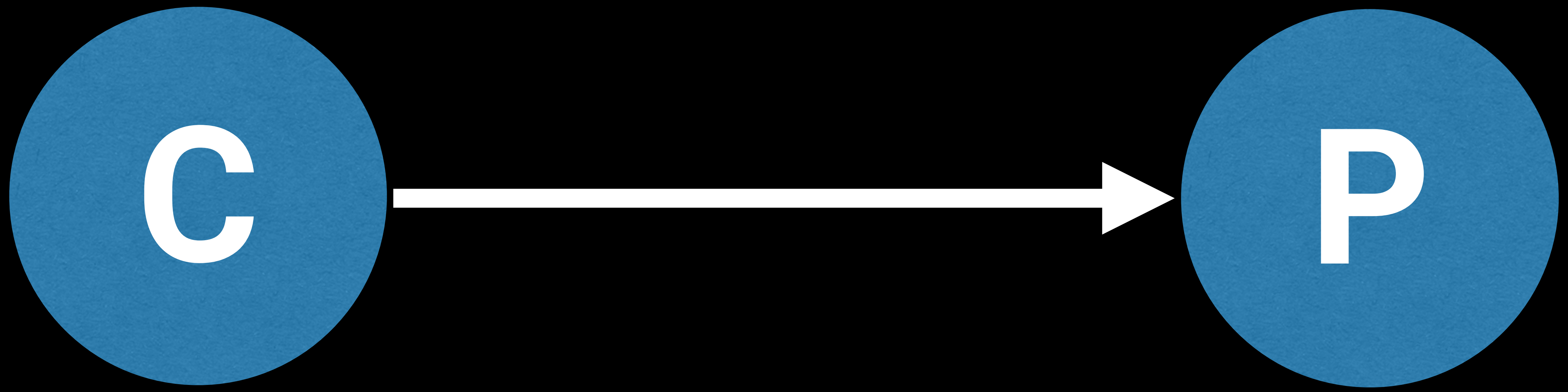
Dependencies



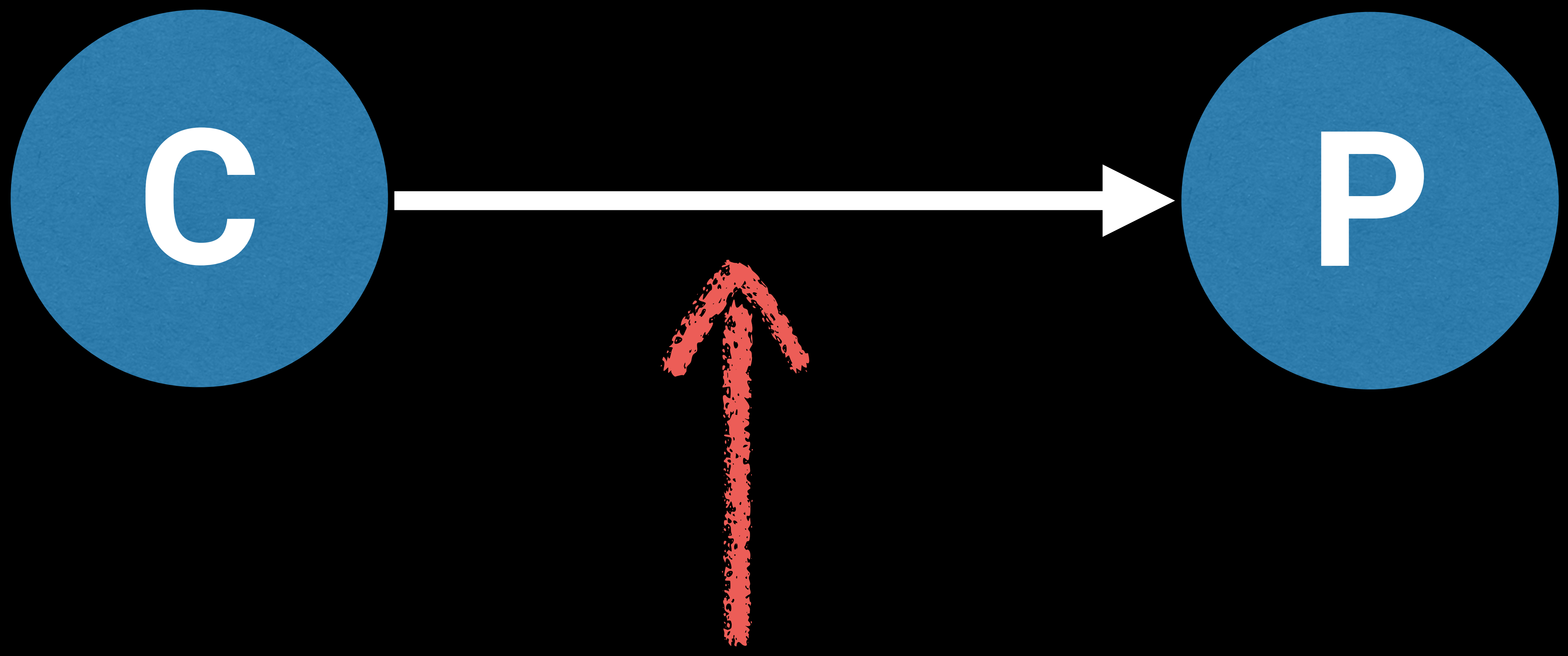
Dependencies



Consumer & provider



Consumer & provider



Contract

So what?

Contract
- an agreement between client and supplier

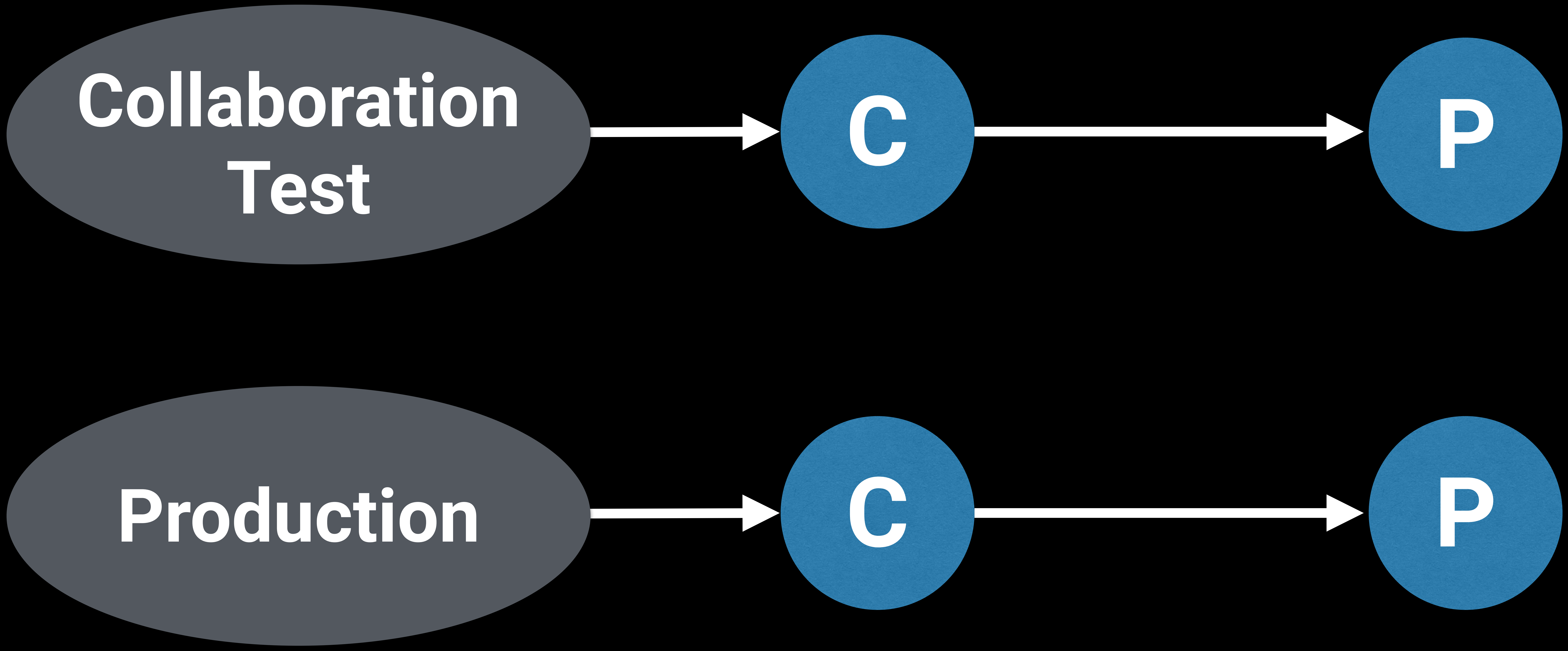
Characteristics
- expect some benefits
- incur some obligations

OBJECT-ORIENTED
SOFTWARE CONSTRUCTION
SECOND EDITION
Winner Software Development Jolt Product Excellence Award 1997
CD-ROM INCLUDED
The Most Comprehensive, Definitive O-O Reference Ever Published
An O-O Tour de Force by a Pioneer in the Field
CD-ROM Includes Complete Hypertext Version of Book AND Object-Oriented Development Environment
BERTRAND MEYER

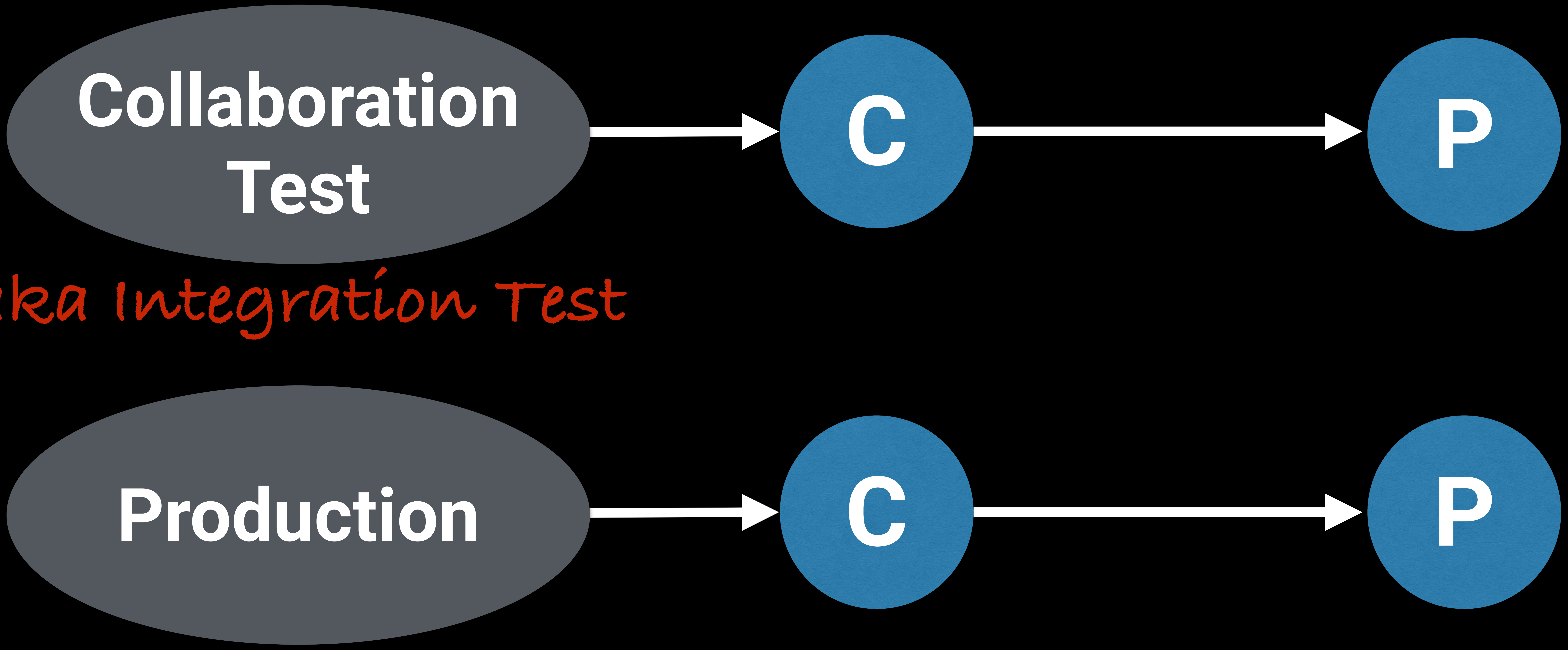
Agenda

Introduction
Why software contracts?
Traditional software testing
A taxonomy of contracts (partial)
Handrolled contract tests
Documenting the contract
Micro-services (finally)
Pact, PactBroker and more
“Bidirectional” contract testing
Questions

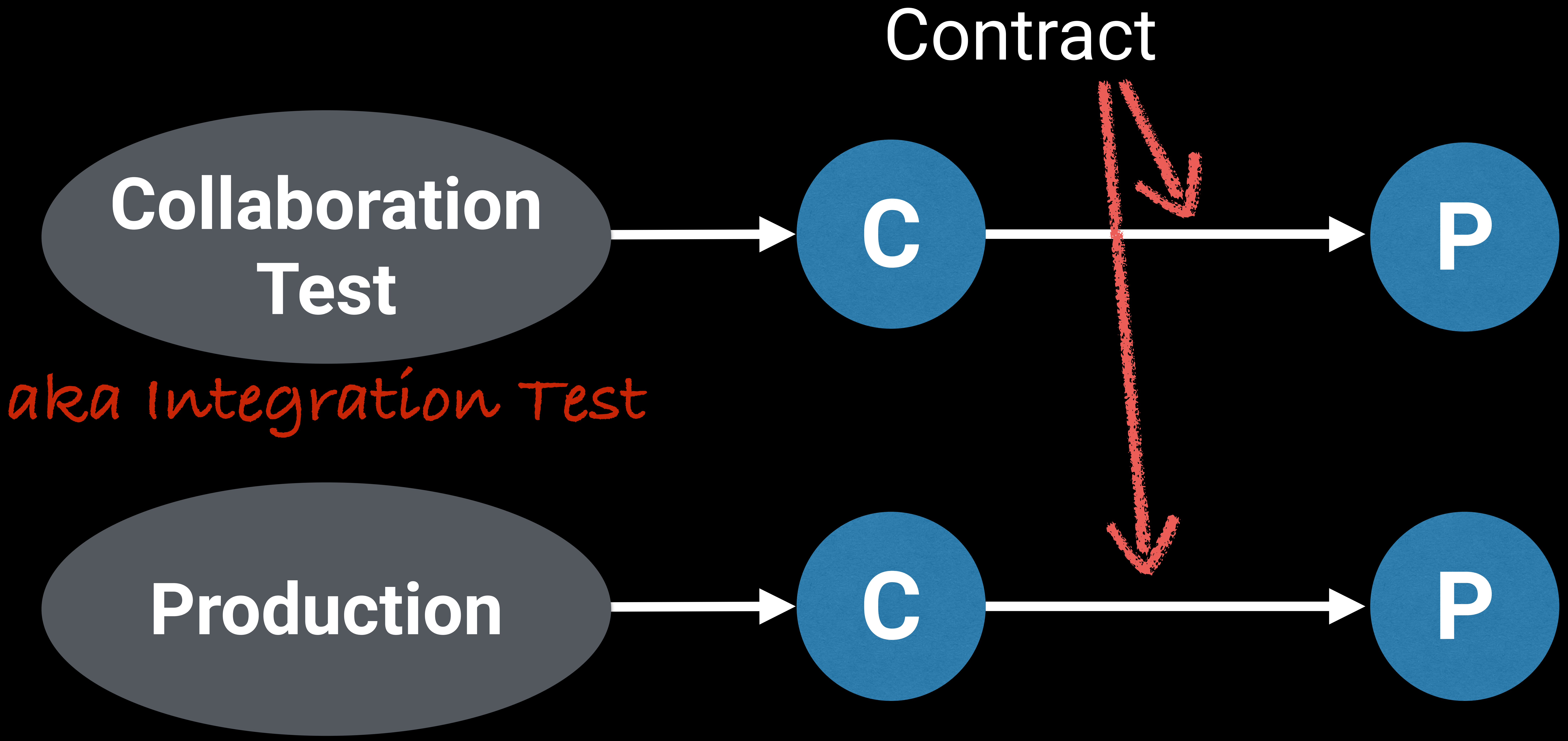
Naïve approach



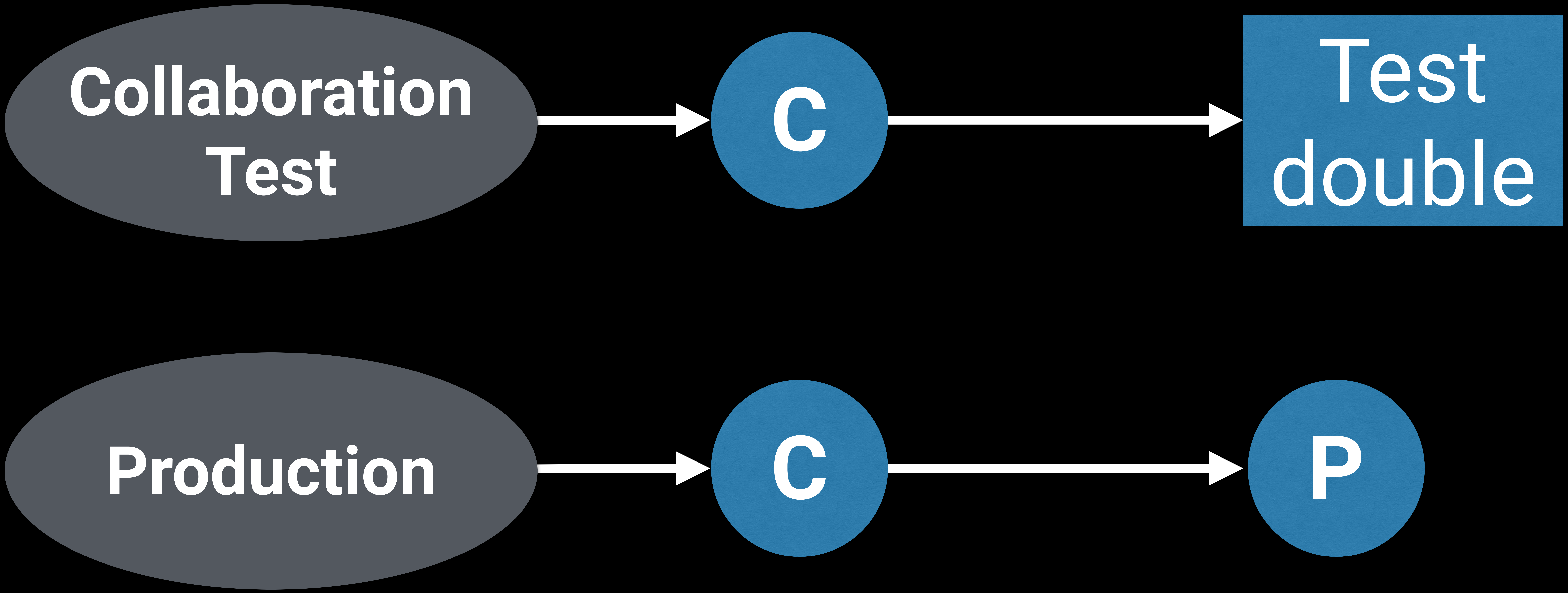
Naïve approach



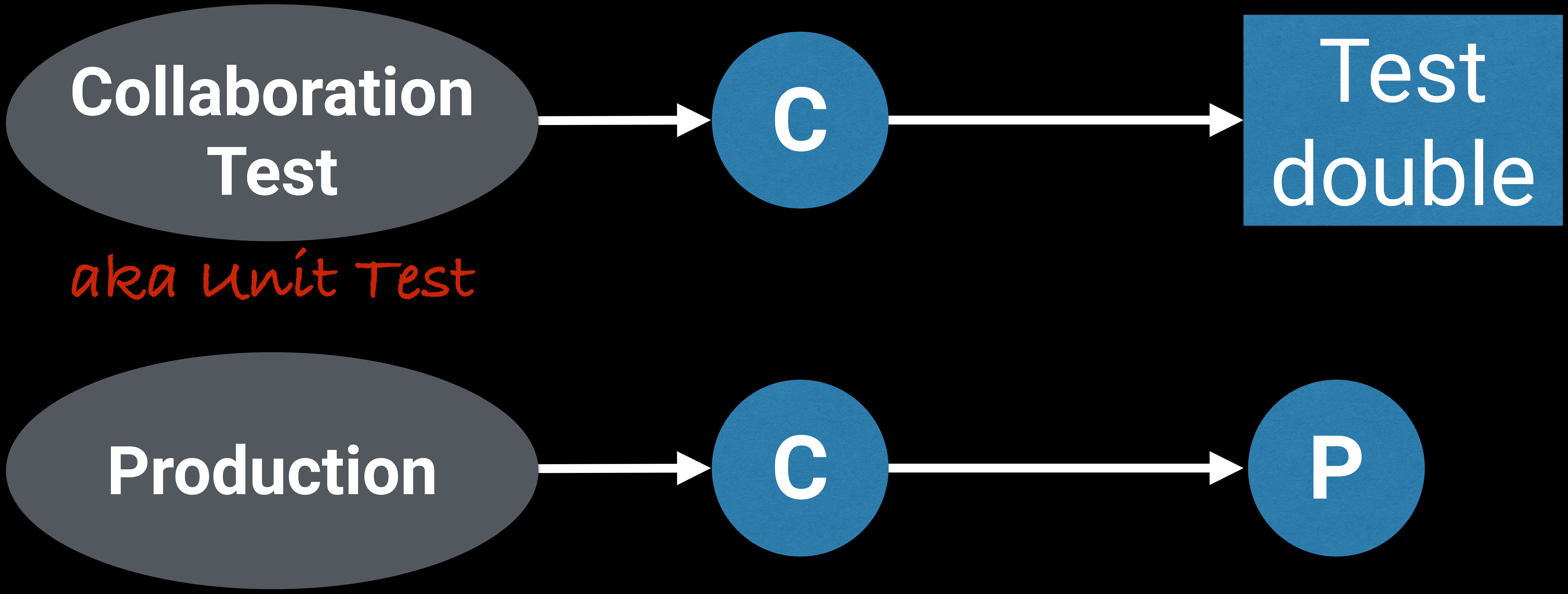
Naïve approach



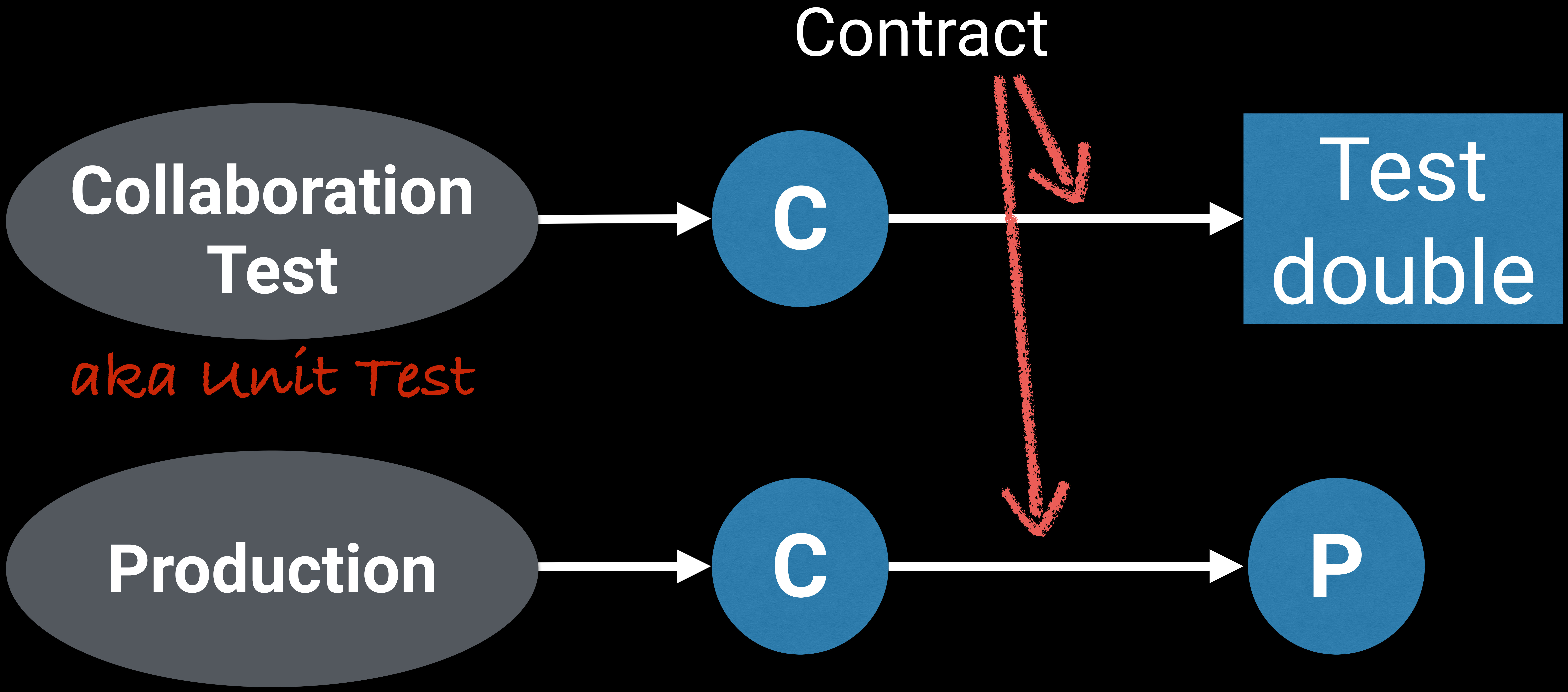
Decoupled approach



Decoupled approach

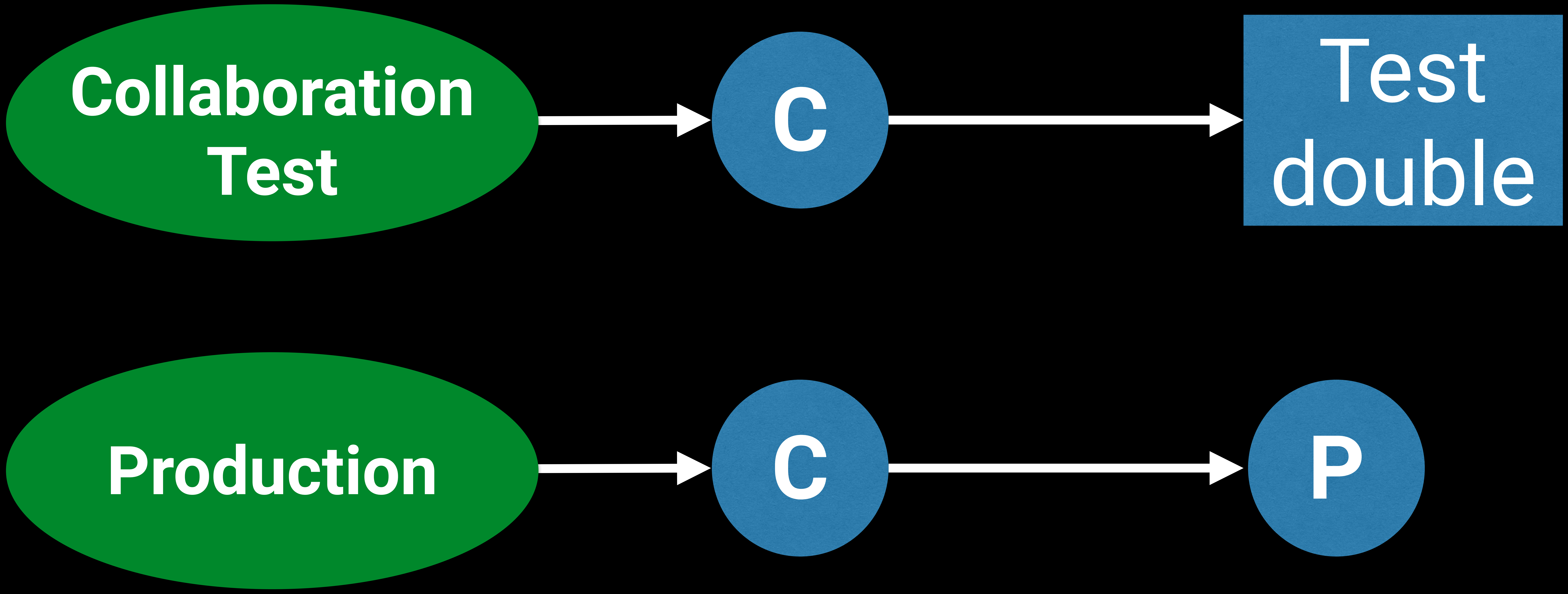


Decoupled approach



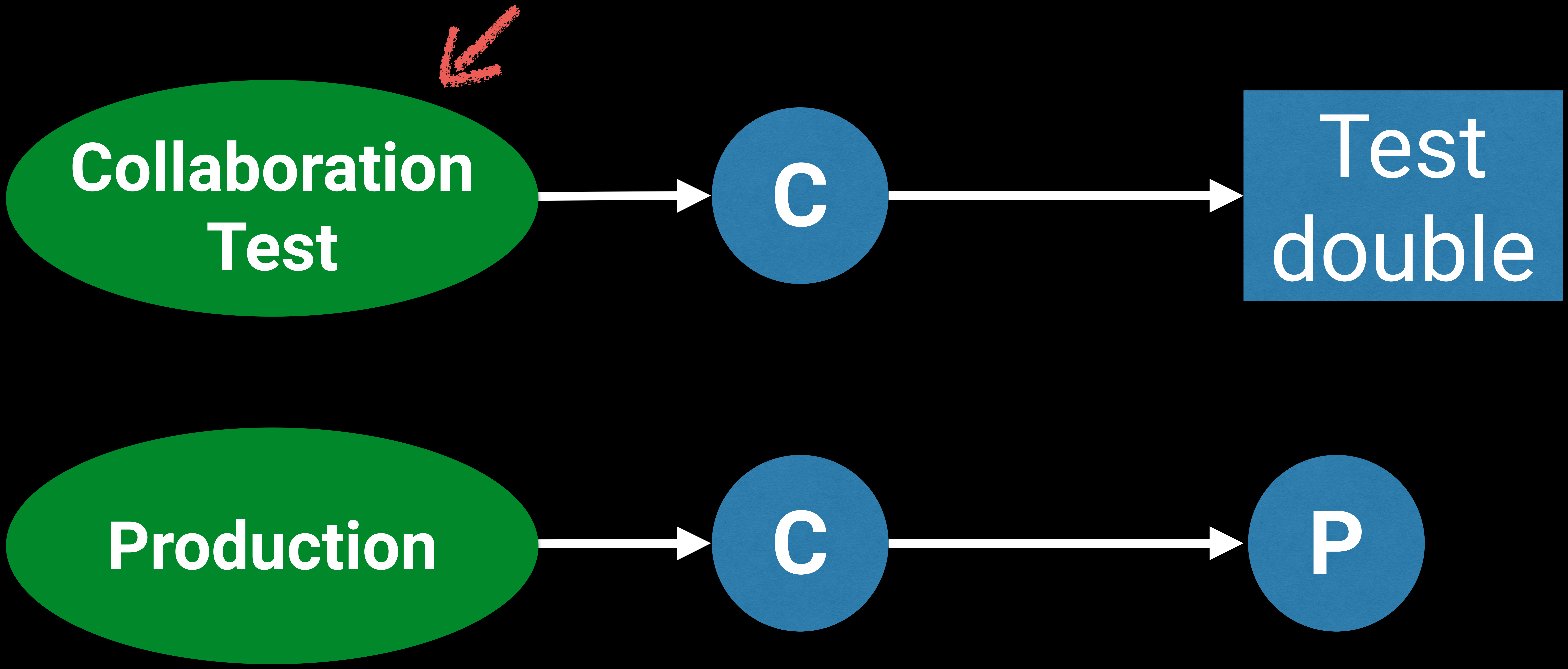
What could possibly go wrong?

Success!



Success!

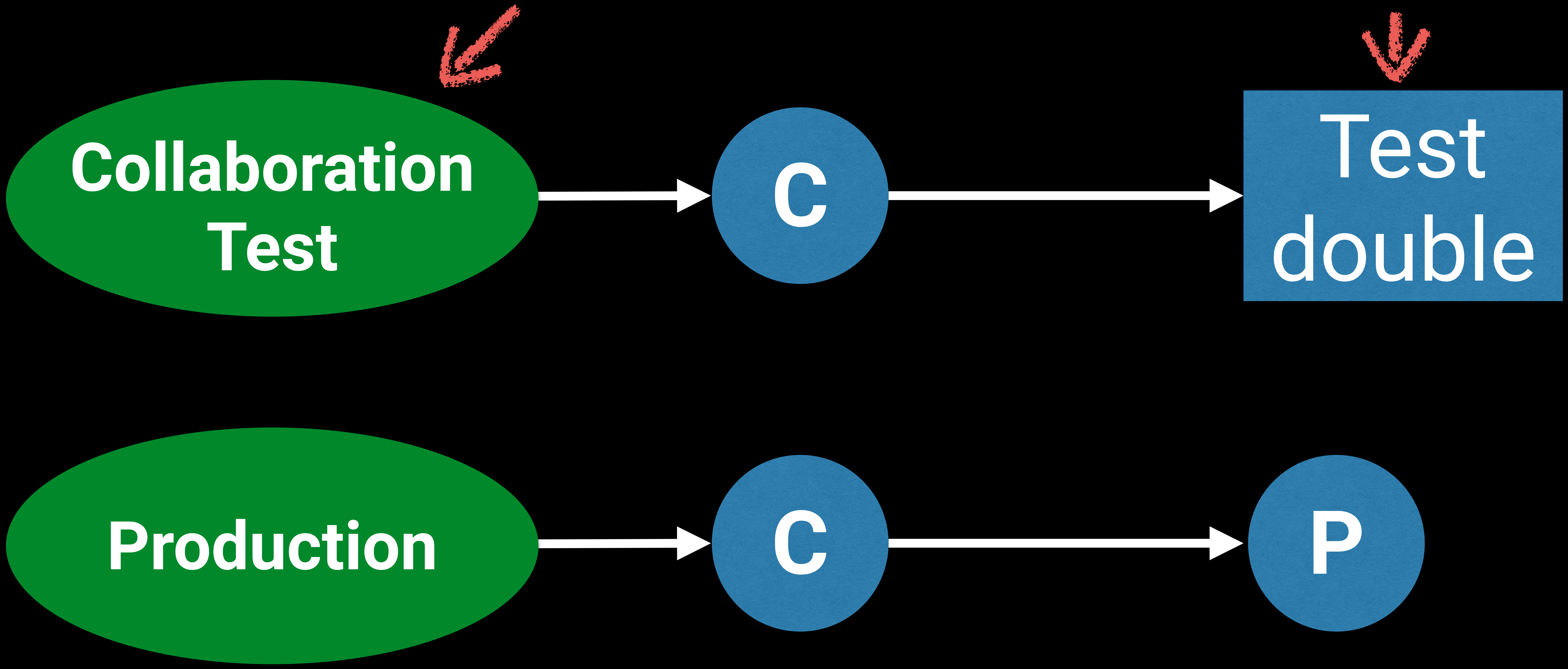
Makes assumptions about the contract



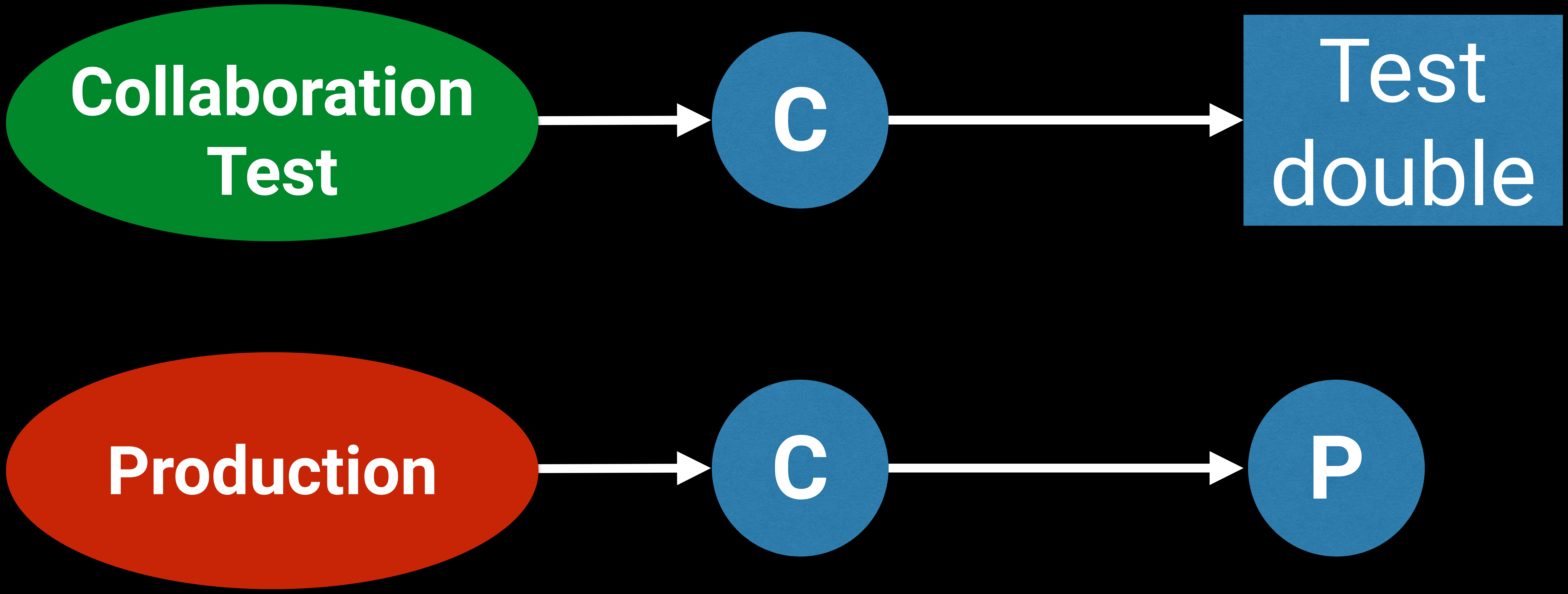
Success!

Makes assumptions about the contract

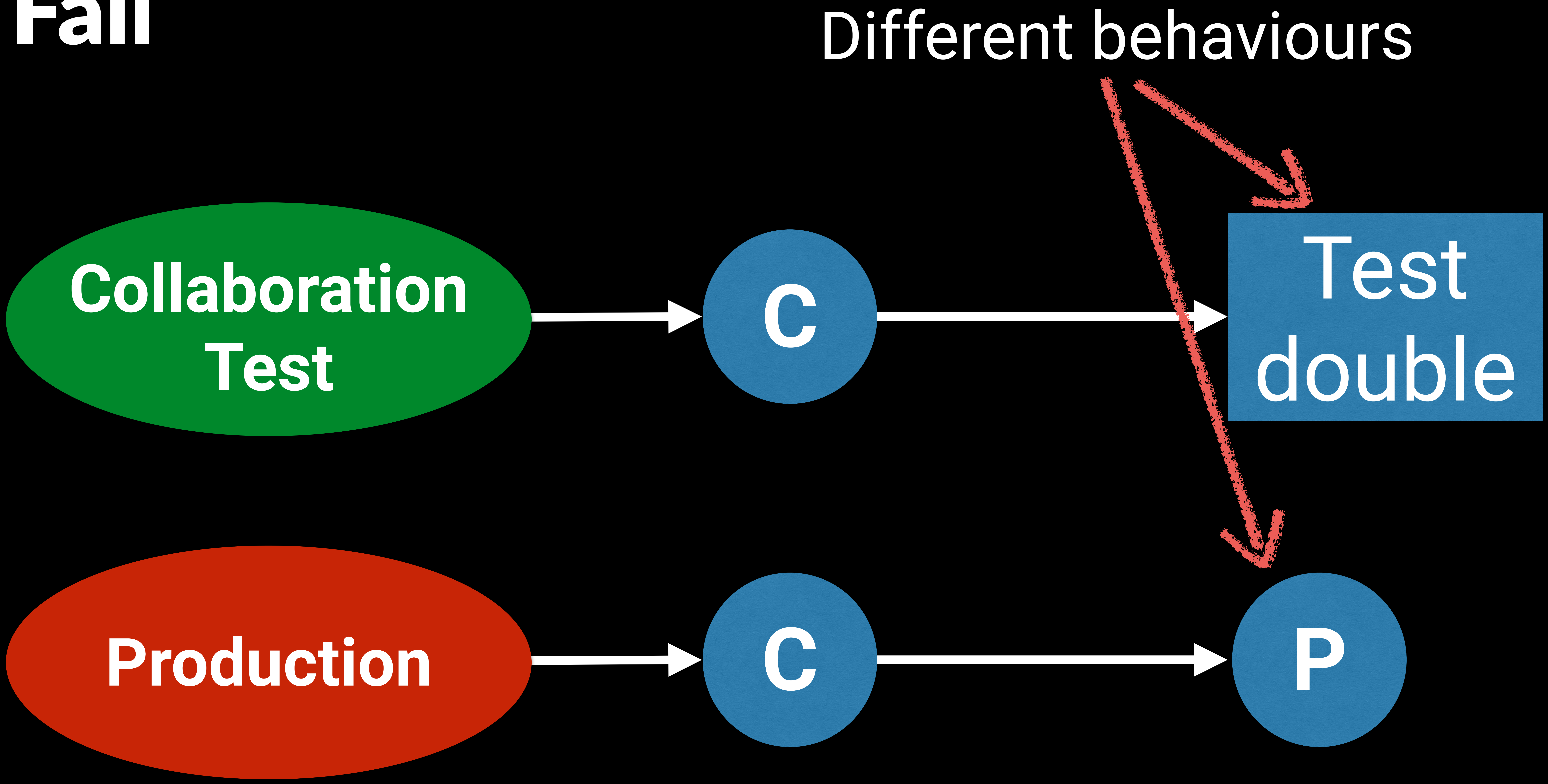
Satisfies those assumptions



Fail



Fail



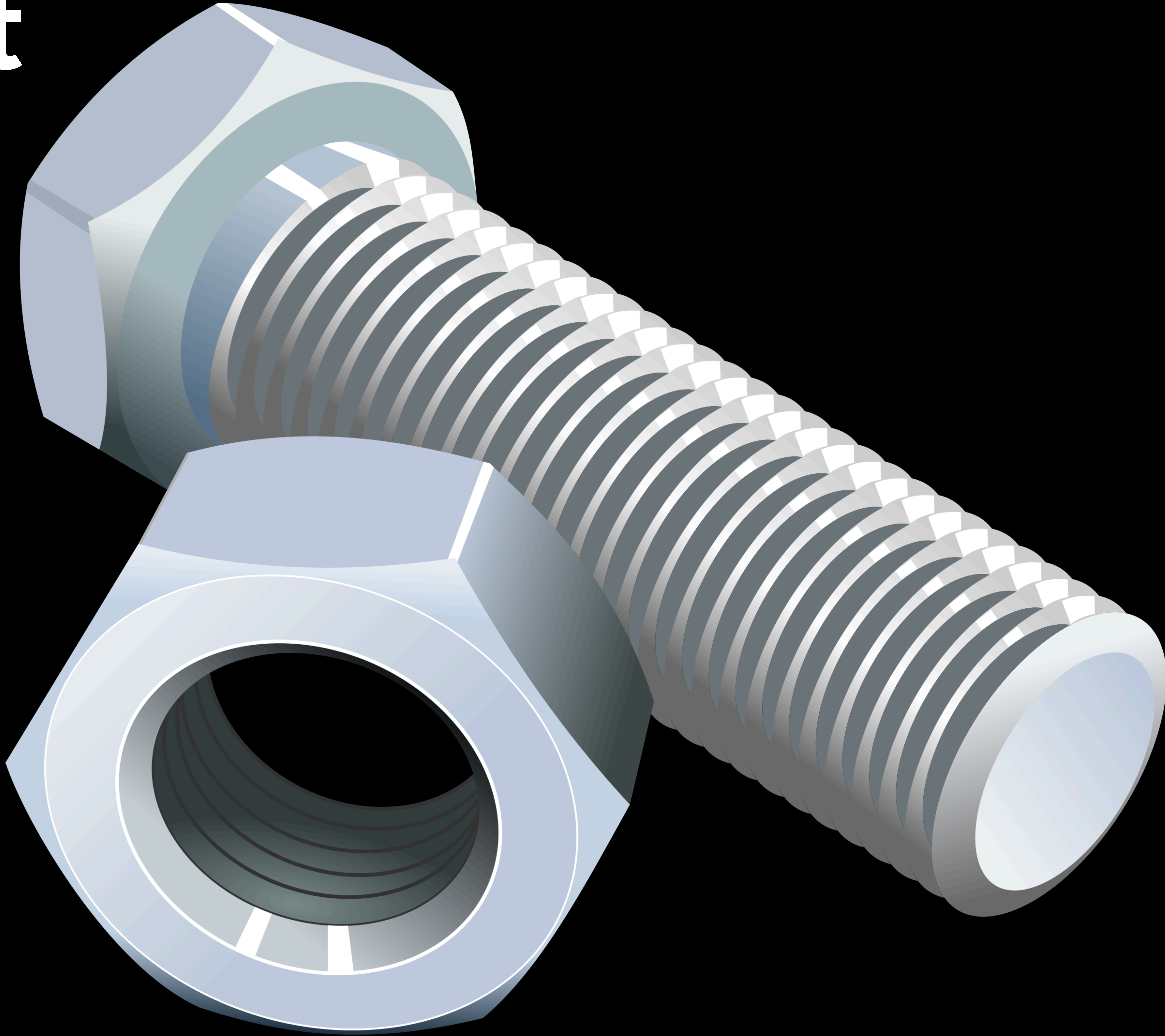
Different behaviours
Collaboration Test
C
Test double
Production
C
P

So what?

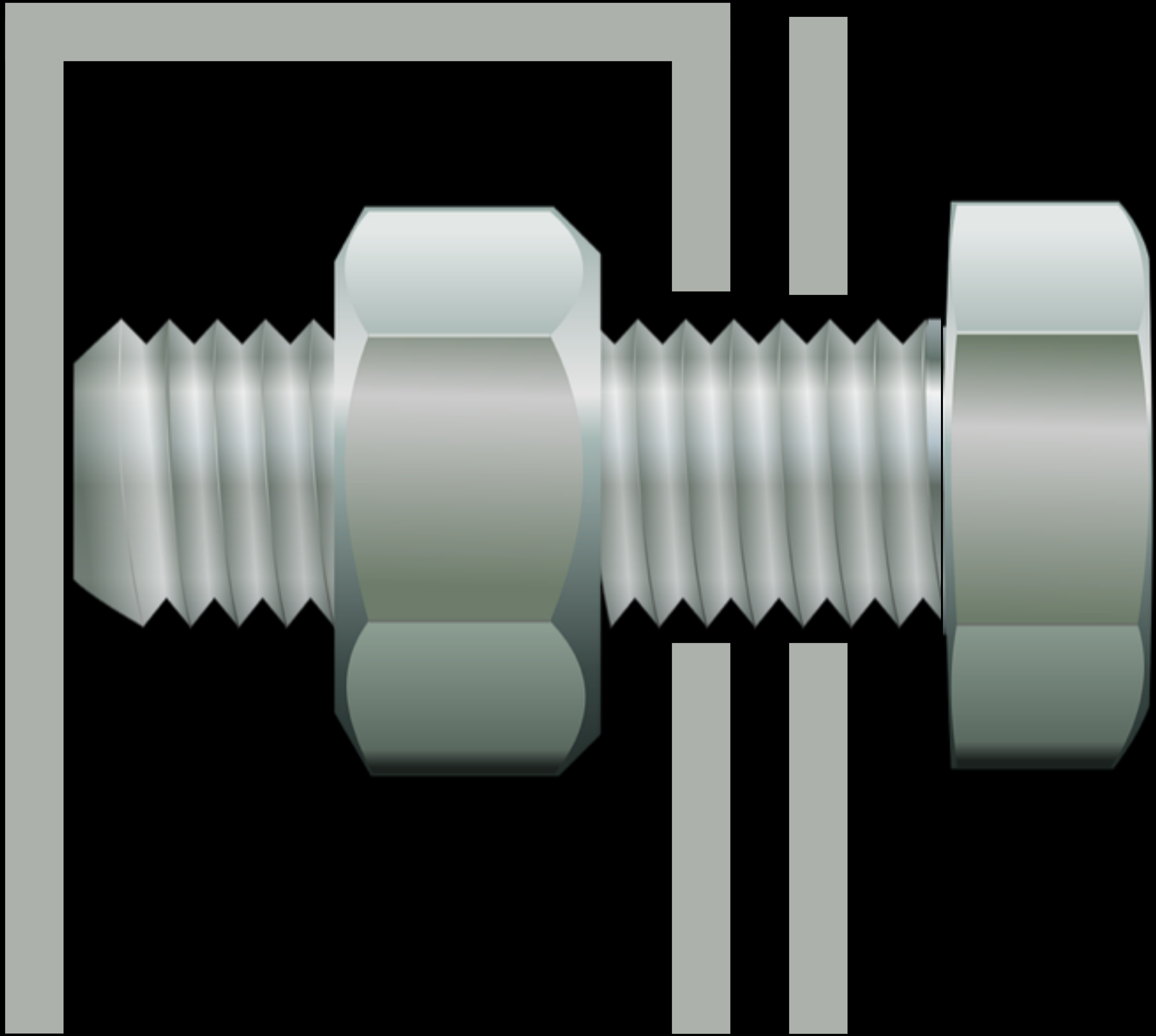
- Test doubles enable isolated testing of components
- Test doubles don’t necessarily behave in the same way as the component they replace
- Testing that uses test doubles can give a false sense of security

Agenda

Explicit contract

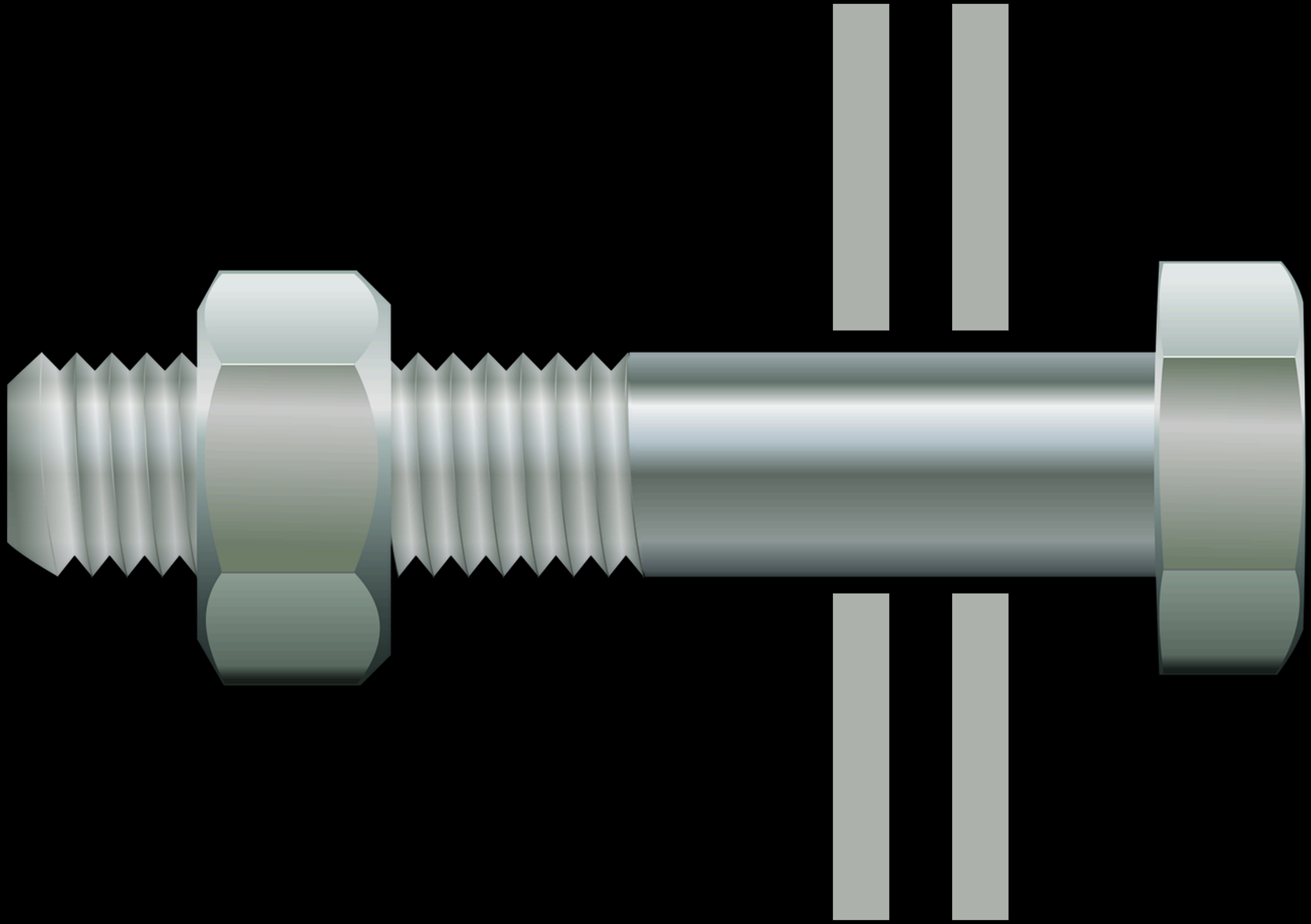


Implicit contract



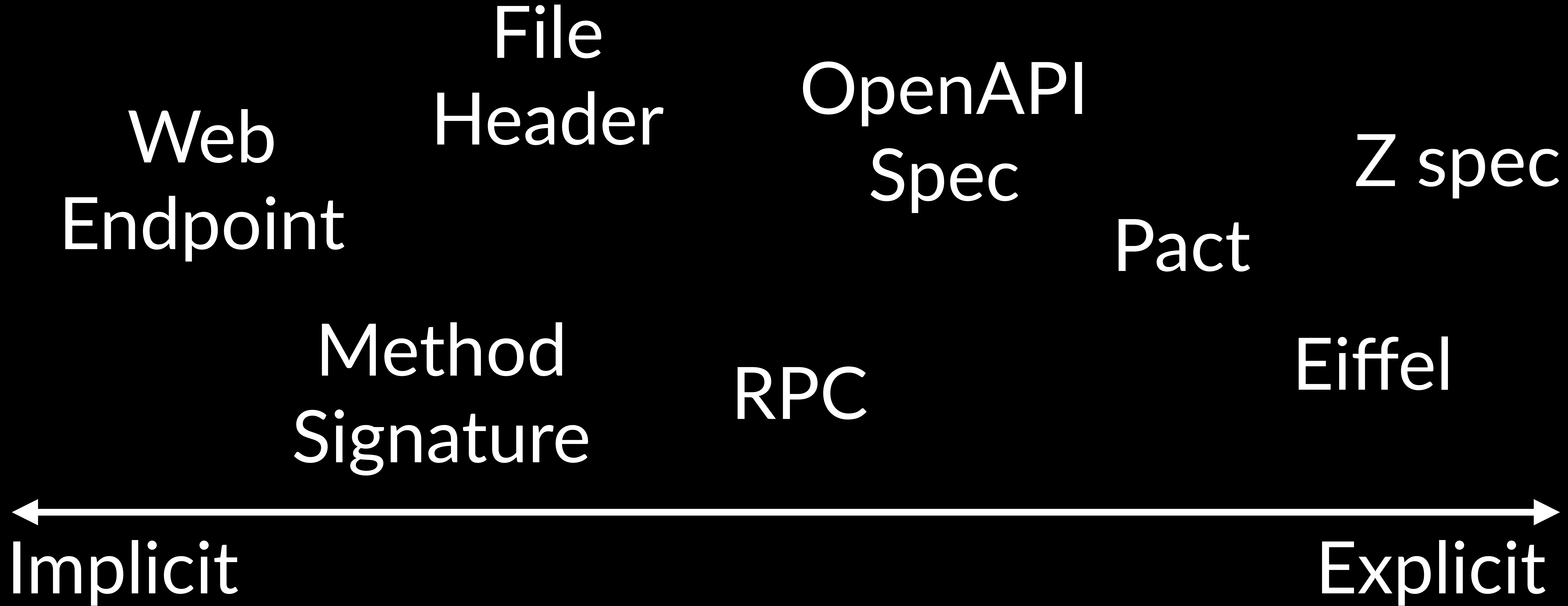
https://www.slidebackgrounds.com/bolt/jkM2jm8-bolt-nut-screw-backgrounds/840

Implicit contract



https://www.slidebackgrounds.com/bolt/jkM2jm8-bolt-nut-screw-backgrounds/840

Contracts as a spectrum



Another categorisation

Another categorisation

- **Informal / absent** - implicit
- **Unilateral** - partially explicit
- **Consumer driven** - partially explicit
- **Formal** - explicit

Another categorisation

- **Informal / absent** - implicit
- **Unilateral** - partially explicit
- **Consumer driven** - partially explicit
- **Formal** - explicit

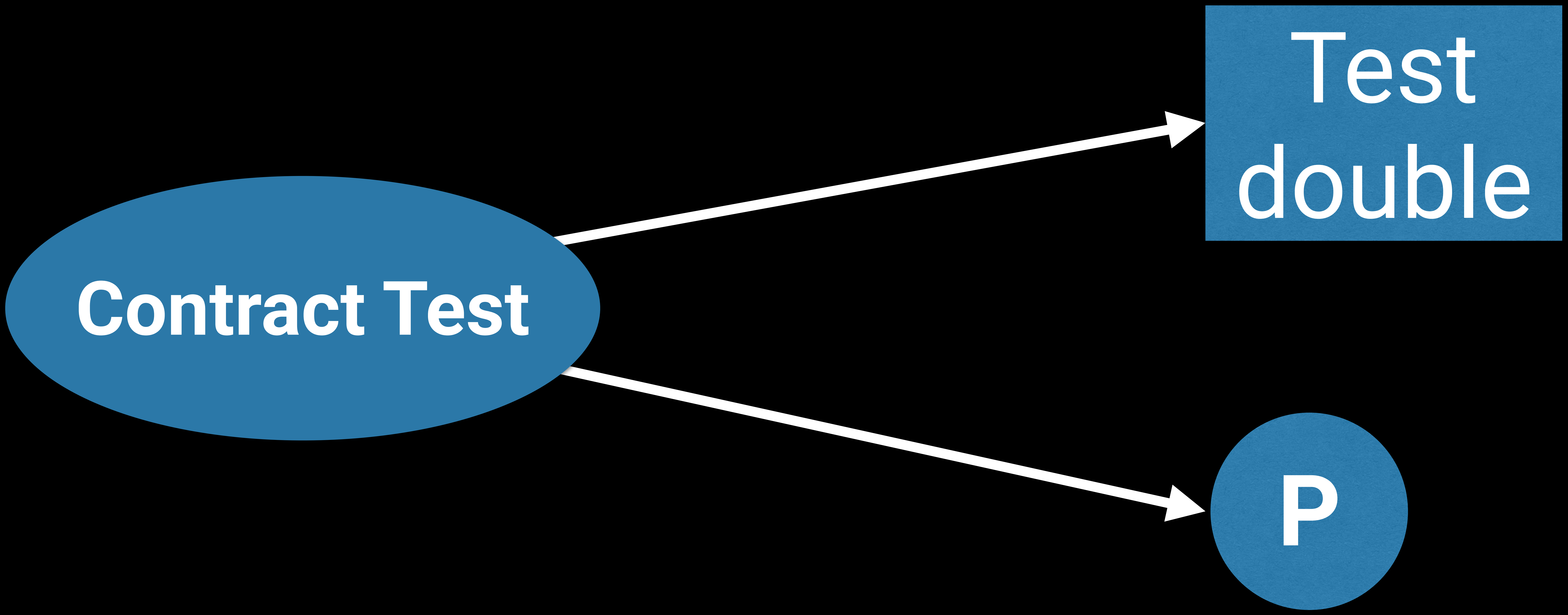
So what?

- All interaction between components is governed by contracts
- Contracts may be explicit, implicit, or a combination of both

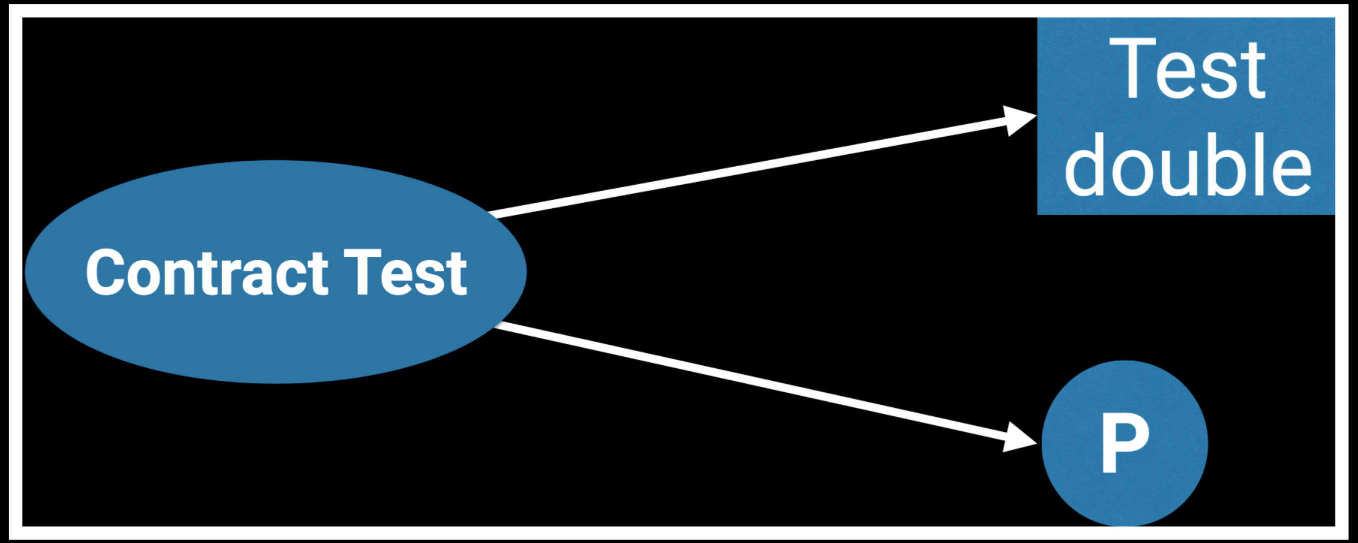
Agenda

Introduction
Why software contracts?
Traditional software testing
A taxonomy of contracts (partial)
Handrolled contract tests
Documenting the contract
Micro-services (finally)
Pact, PactBroker and more
“Bidirectional” contract testing
Questions

Hand-rolled contract tests



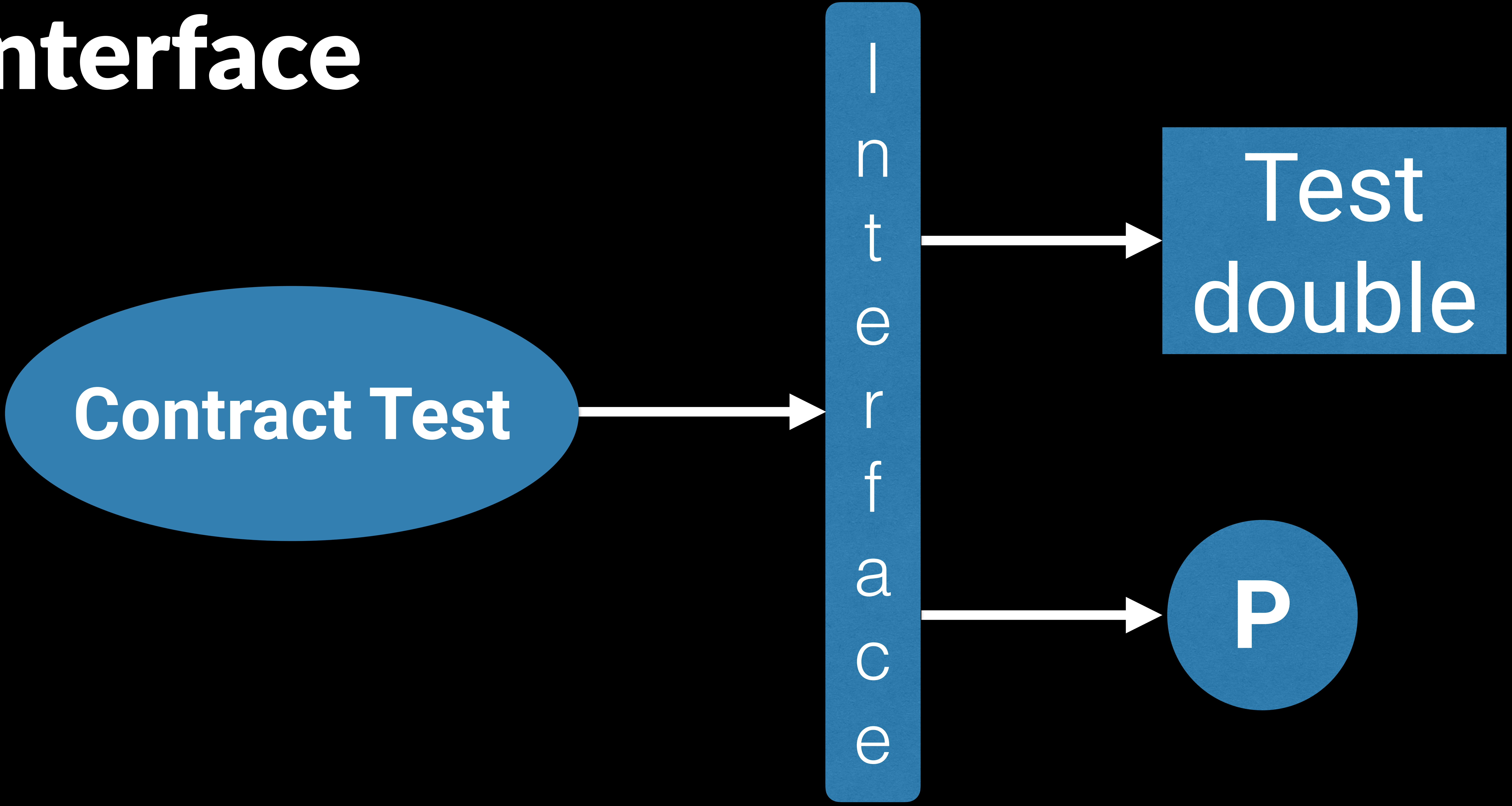
Systematic contract testing

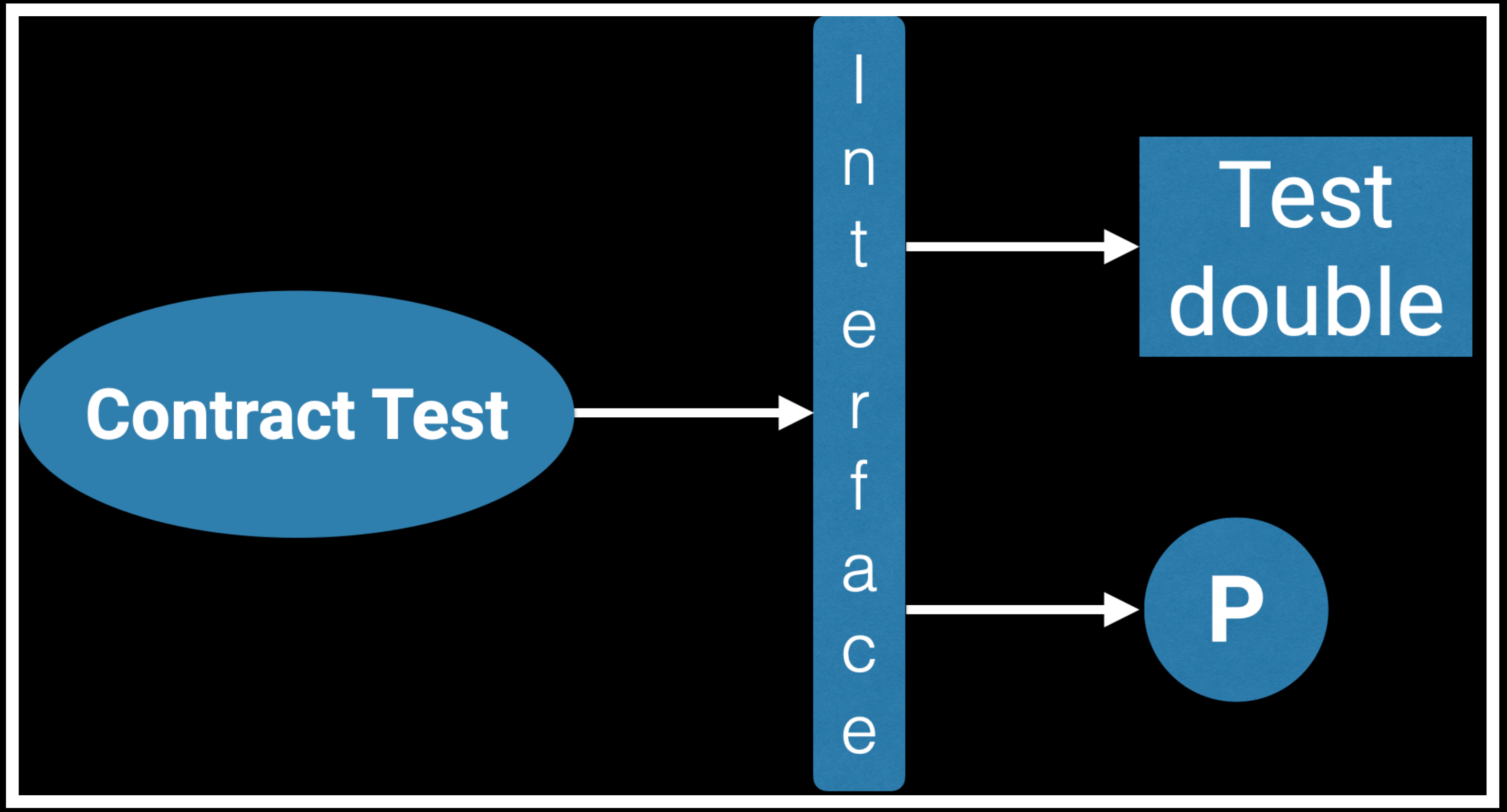


- Collaboration tests make assumptions about the contract
- Contract tests try to justify those assumptions

JB Rainsberger, via GOOS mailing list, “Unit-test mock/stub assumptions rots”
15 March 2012

Contract test through an interface

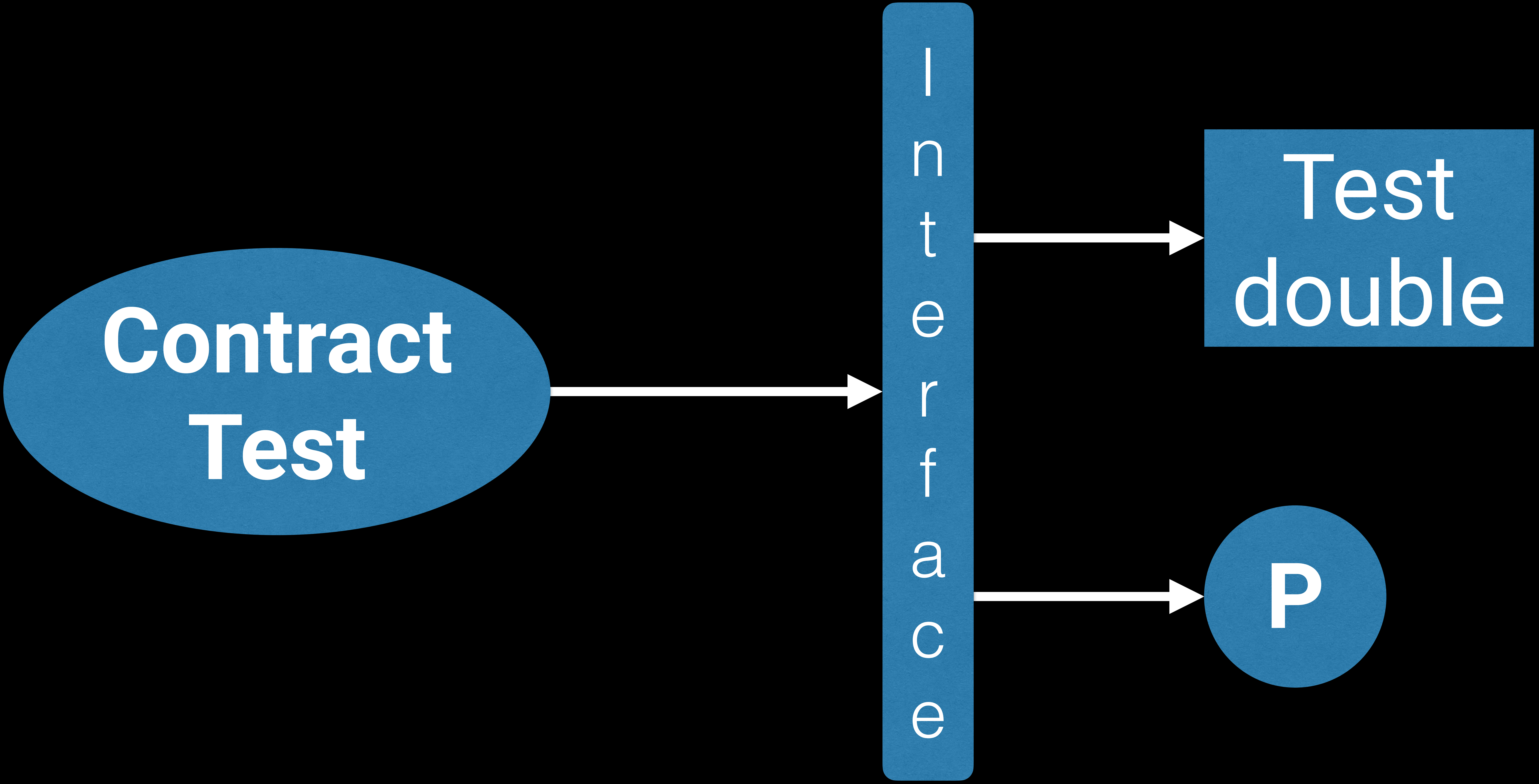




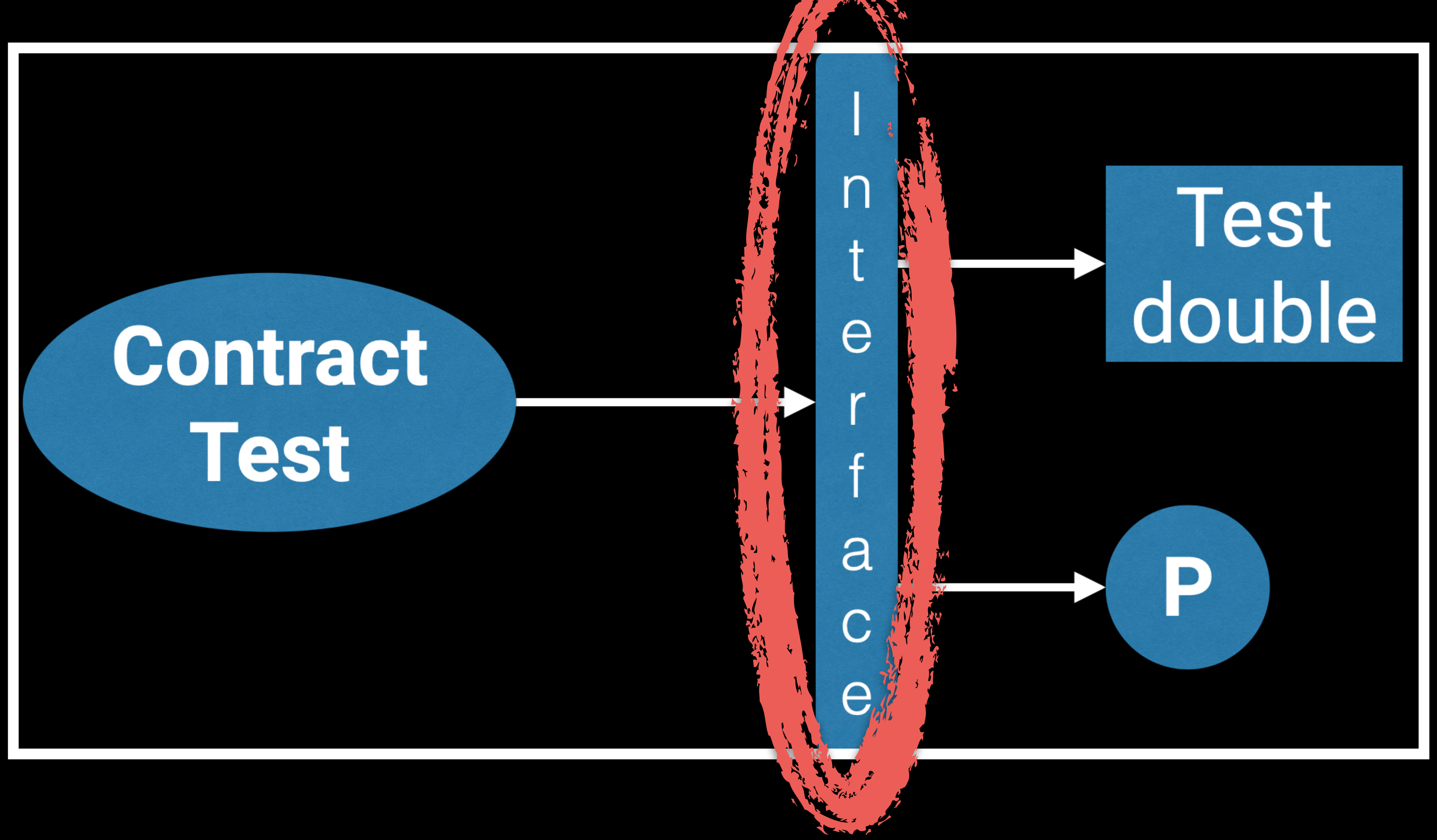
“Don’t mock what you don’t own”

Joe Walnes

From first principles …



Our *explicit* contract



```
public interface IRevenueProvider
{
    decimal GetRevenue(int customerId);
}
```

What’s our *implicit* contract?

What’s our *implicit* contract?

- Pass in a valid customer ID, return a decimal value

What’s our *implicit* contract?

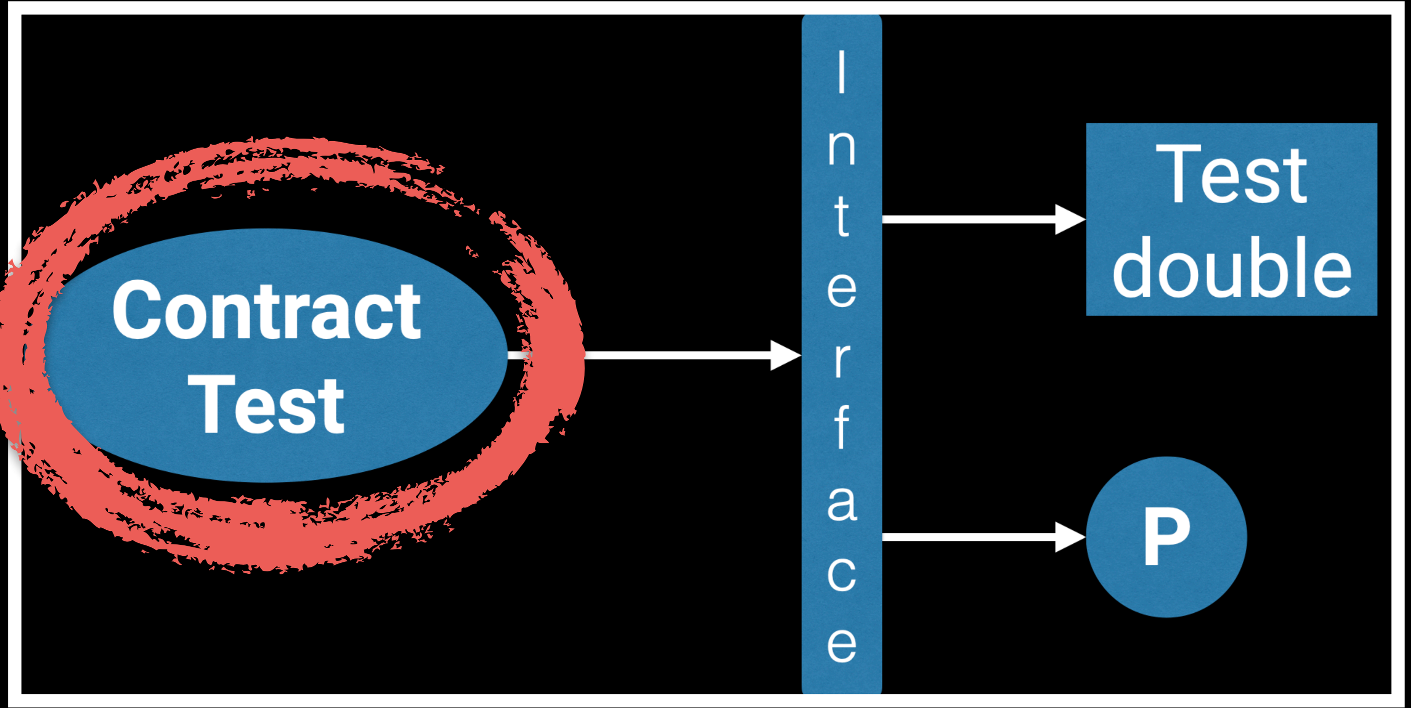
- Pass in a valid customer ID, return a decimal value
  - Can we say anything about the magnitude of the returned value?

What’s our *implicit* contract?

- Pass in a valid customer ID, return a decimal value
  - Can we say anything about the magnitude of the returned value?

- Pass in an invalid customer ID…. then what?

Documenting the contract



```
[TestFixture]
public abstract class RevenueProviderContract
{
    private IRevenueProvider revenueProvider;

    protected abstract
        IRevenueProvider GetRevenueProvider();
```

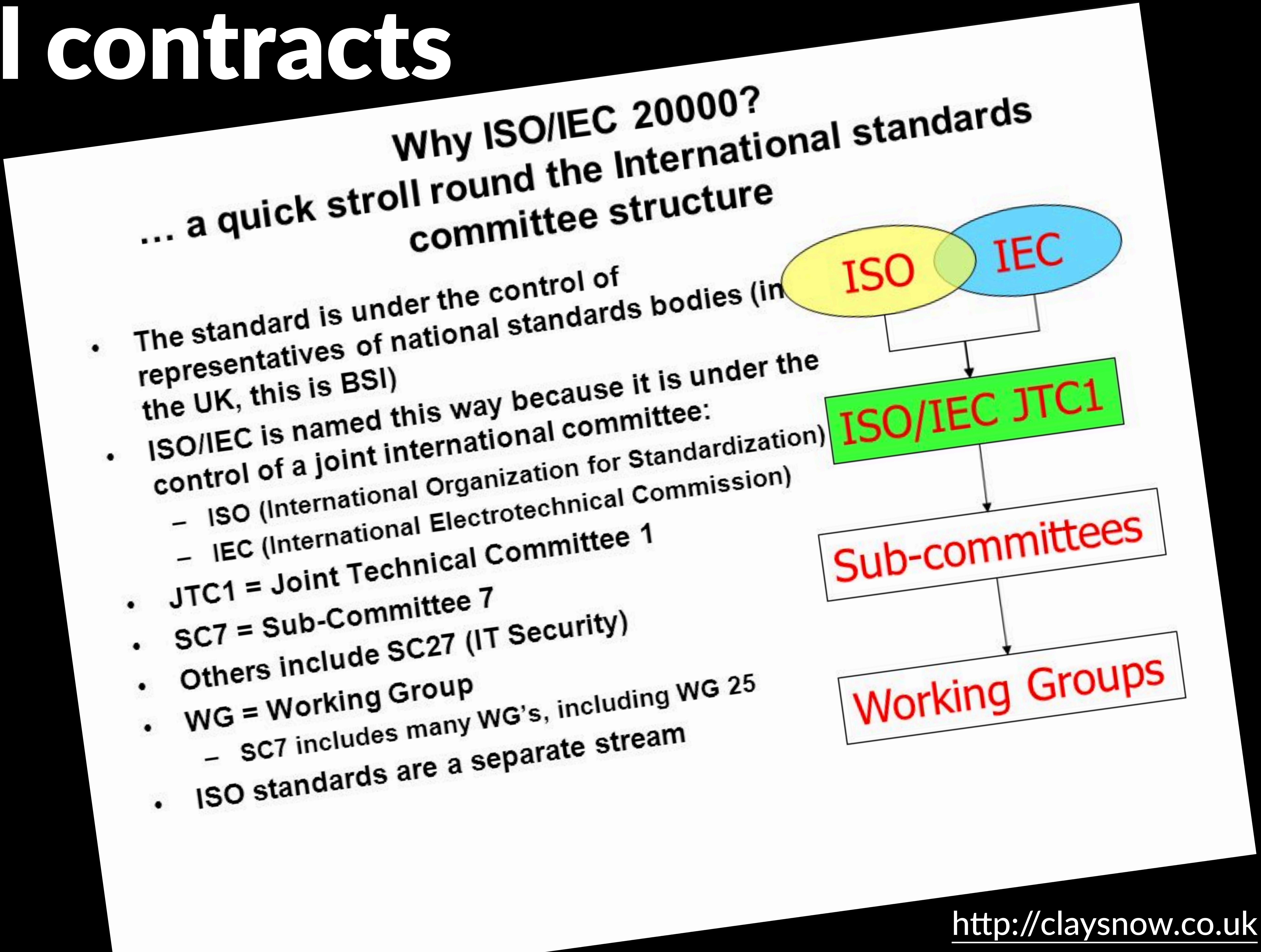
```
private IRevenueProvider revenueProvider;

protected abstract
    IRevenueProvider GetRevenueProvider();


[SetUp]
public void setup()
{
    revenueProvider = GetRevenueProvider();
}

[Test]
public void valid_customer_id()
{
    [Test]
    public void valid_customer_id()
    {
        revenueProvider.GetRevenue(VALID_ID);
    }

    [Test]
    public void invalid_customer_id()
    {
        Assert.Throws<CustomerIdException>(() =>
         revenueProvider.GetRevenue(INVALID_ID));
    }
}
```

So what?

- Contract tests run the same test code against the production component AND the test double
- Doing this by hand is onerous

Agenda

Unilateral contracts



Why ISO/IEC 20000?

… a quick stroll round the International standards committee structure

- The standard is under the control of representatives of national standards bodies (in the UK, this is BSI)
- ISO/IEC is named this way because it is under the control of a joint international committee:
  - ISO (International Organization for Standardization)
  - IEC (International Electrotechnical Commission)
- JTC1 = Joint Technical Committee 1
- SC7 = Sub-Committee 7
- Others include SC27 (IT Security)
- WG = Working Group
  - SC7 includes many WG's, including WG 25
- ISO standards are a separate stream

Unilateral contracts

Why ISO/IEC 20000?
… a quick stroll round the International standards committee structure

- The standard is under the control of representatives of national standards bodies (in the UK, this is BSI)
- ISO/IEC is named this way because it is under the control of a joint international committee:
  - ISO (International Organization for Standardization)
  - IEC (International Electrotechnical Commission)
- JTC1 = Joint Technical Committee 1
- SC7 = Sub-Committee 7

- **Closed and complete** Unilateral contracts express a service's business function capabilities in terms of the complete set of exportable elements available to consumers, and as such are closed and complete with respect to the functionality available to the system.

- **Singular and authoritative** Unilateral contracts are singular and authoritative in their expression of the business functionality available to the system.

- **Bounded stability and immutability** A Unilateral contract is stable and immutable for a bounded period and/or locale. Provider contracts typically use some form of versioning to differentiate differently bounded instances of the contract.

https://martinfowler.com/articles/consumerDrivenContracts.html

How do you agree a Unilateral contract?

- [Optional] Read the T&Cs
- Click “AGREE”

Consumer driven contracts (CDC)

- Open and incomplete Consumer contracts are open and incomplete with respect to the business functionality available to the system. They express a subset of the system's business function capabilities in terms of the consumer's expectations of the provider contract.

- Multiple and non-authoritative Consumer contracts are multiple in proportion to the number of consumers of a service, and each is non-authoritative with regard to the total set of contractual obligations placed on the provider. Consumers may evolve at different rates.

- Bounded stability and immutability Like provider contracts, consumer contracts are valid for a particular period of time and/or location.

How do you agree a CDC?

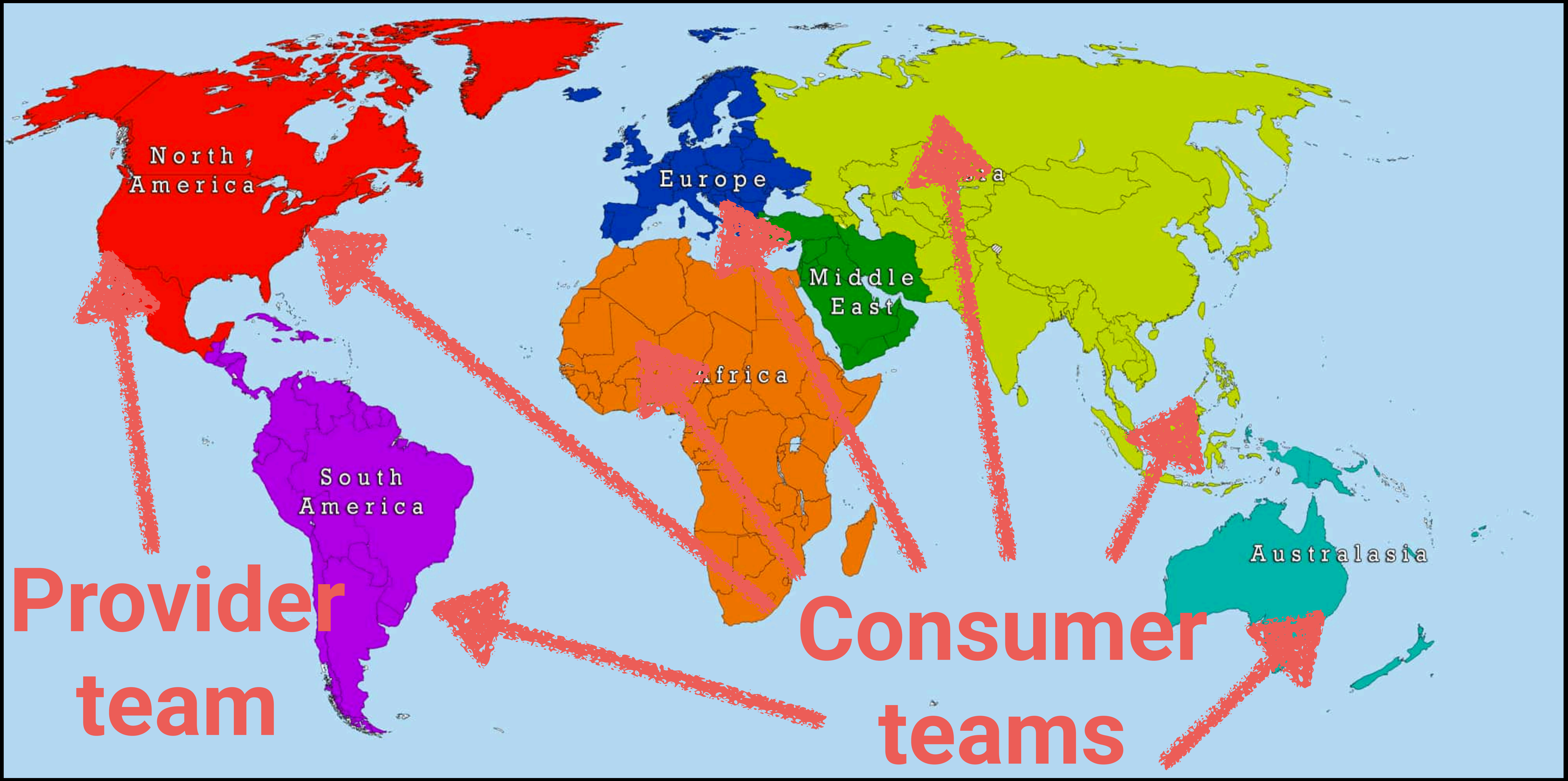
How do you agree a CDC?

How do you agree a CDC?

Consumer team

Provider team

How do you agree a CDC?



How do you agree a CDC?

You negotiate and document the contract.

Consumer driven contracts

Consumer driven contracts

- Each consumer captures their expectations of the provider in a separate contract.

Consumer driven contracts

- Each consumer captures their expectations of the provider in a separate contract.
- All of these contracts are shared with the provider so they gain insight into the obligations they must fulfil for each individual client.

Consumer driven contracts

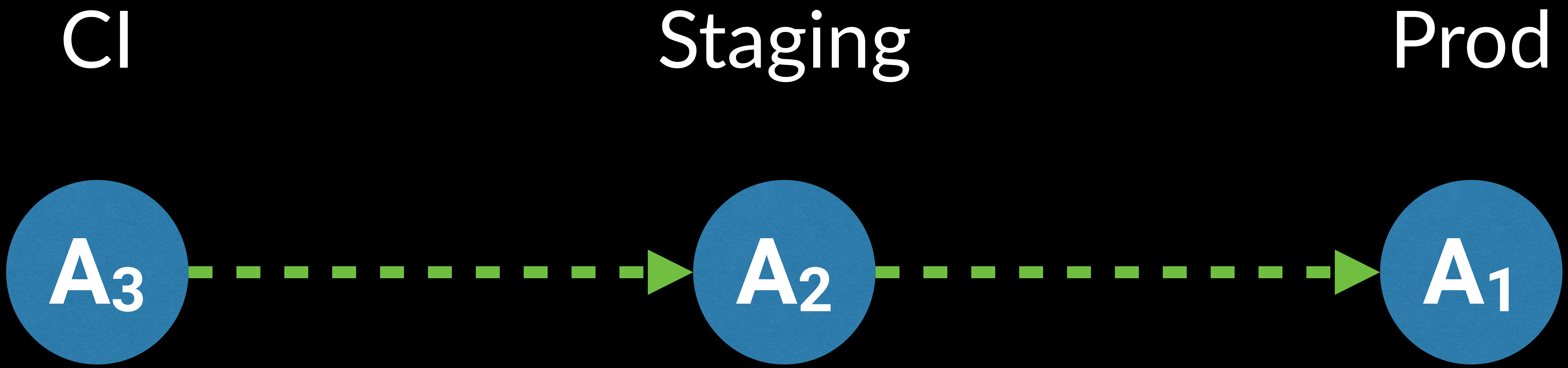
- Each consumer captures their expectations of the provider in a separate contract.
- All of these contracts are shared with the provider so they gain insight into the obligations they must fulfil for each individual client.
- The provider can create a test suite to validate these obligations.

So what?

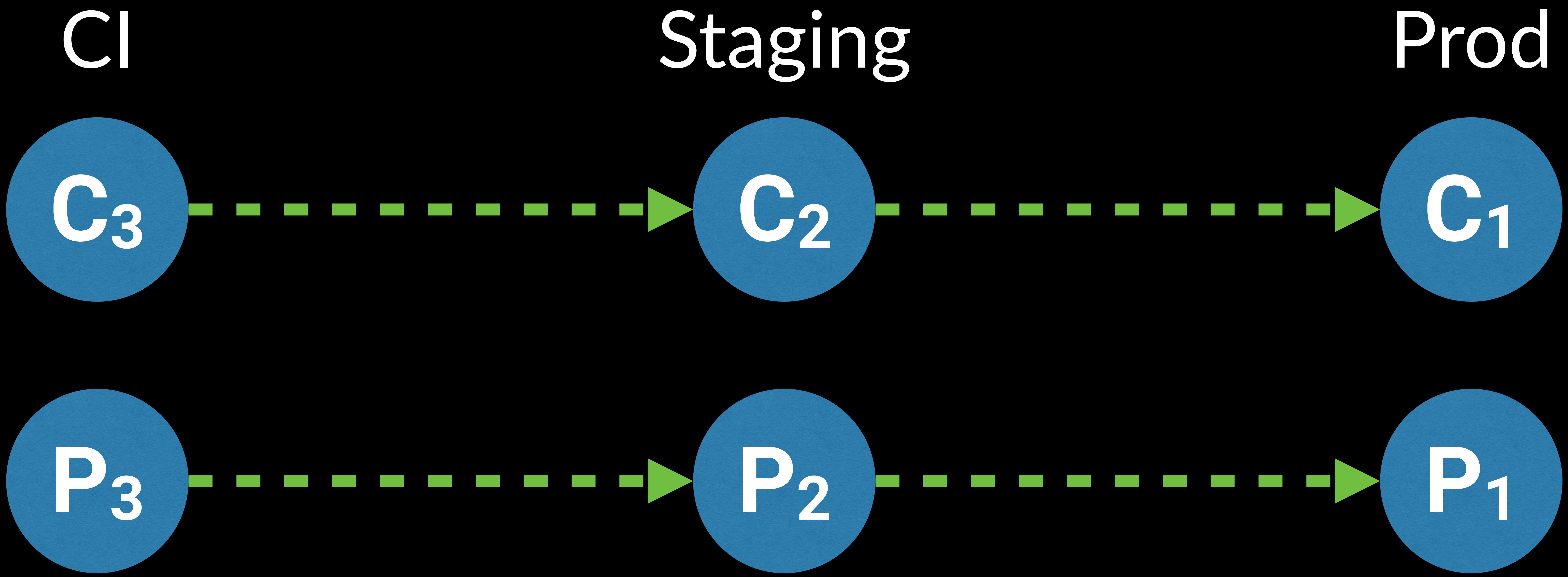
- You either accept what you're given OR you have to negotiate between producer and consumer
- Consumer-driven contracts work well within globally distributed organisations
  - Automation eases the burden, but communication is still essential

Agenda

Continuous Integration (CI)



CI - consumer & provider


$$\begin{tikzcd}
\text{CI} & \text{Staging} & \text{Prod} \\
C_3 \arrow[r, dashed] & C_2 \arrow[r, dashed] & C_1 \\
P_3 \arrow[r, dashed] & P_2 \arrow[r, dashed] & P_1
\end{tikzcd}$$

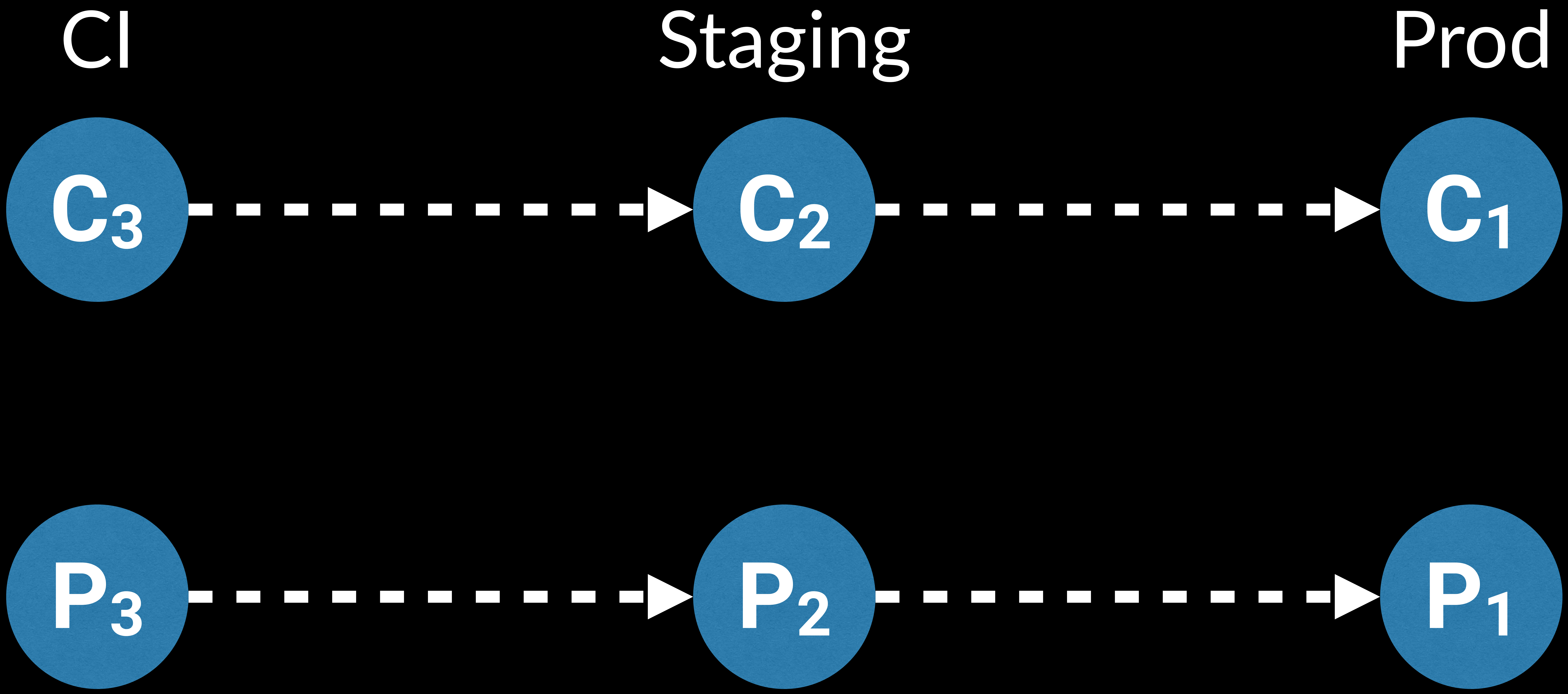
Remember this?

If you can’t deploy services independently, you don’t have micro-services.

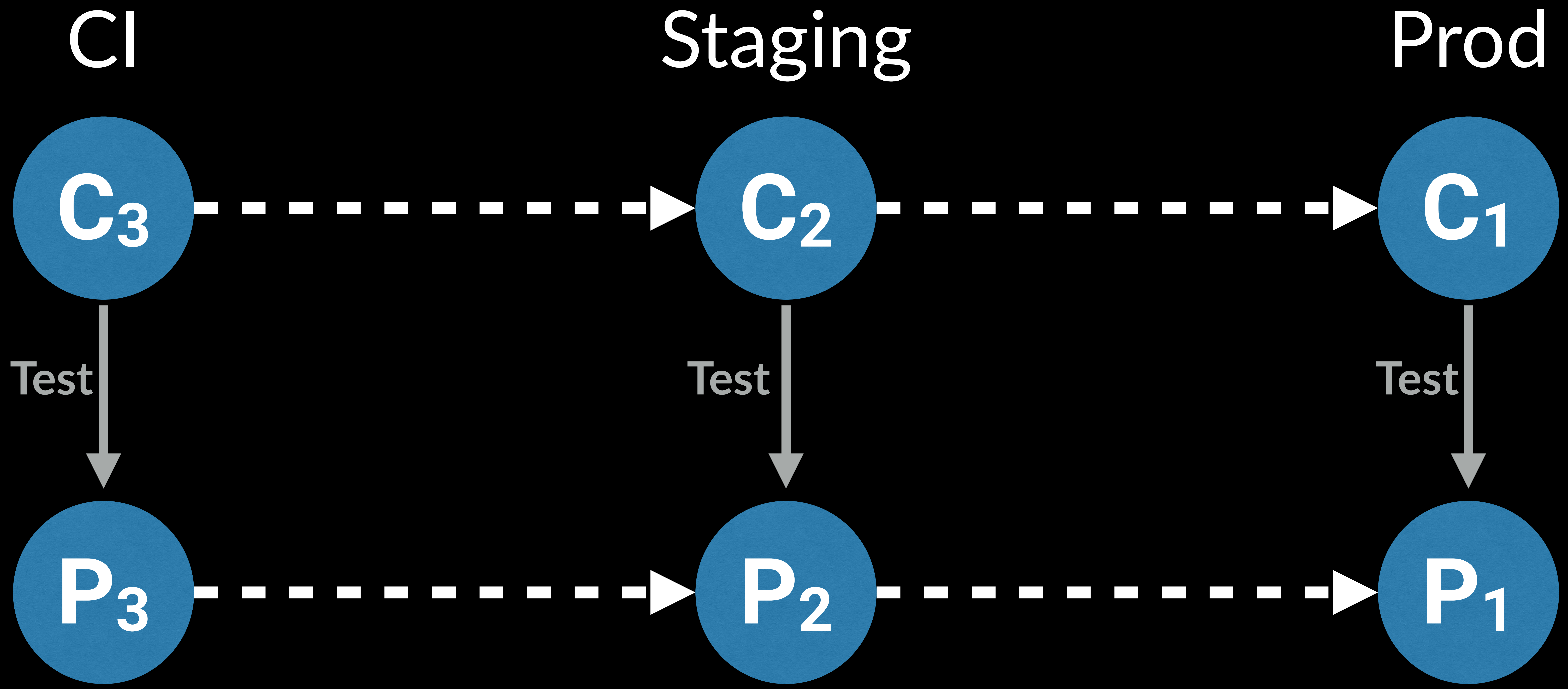
You have a distributed monolith

Beth Skurrie

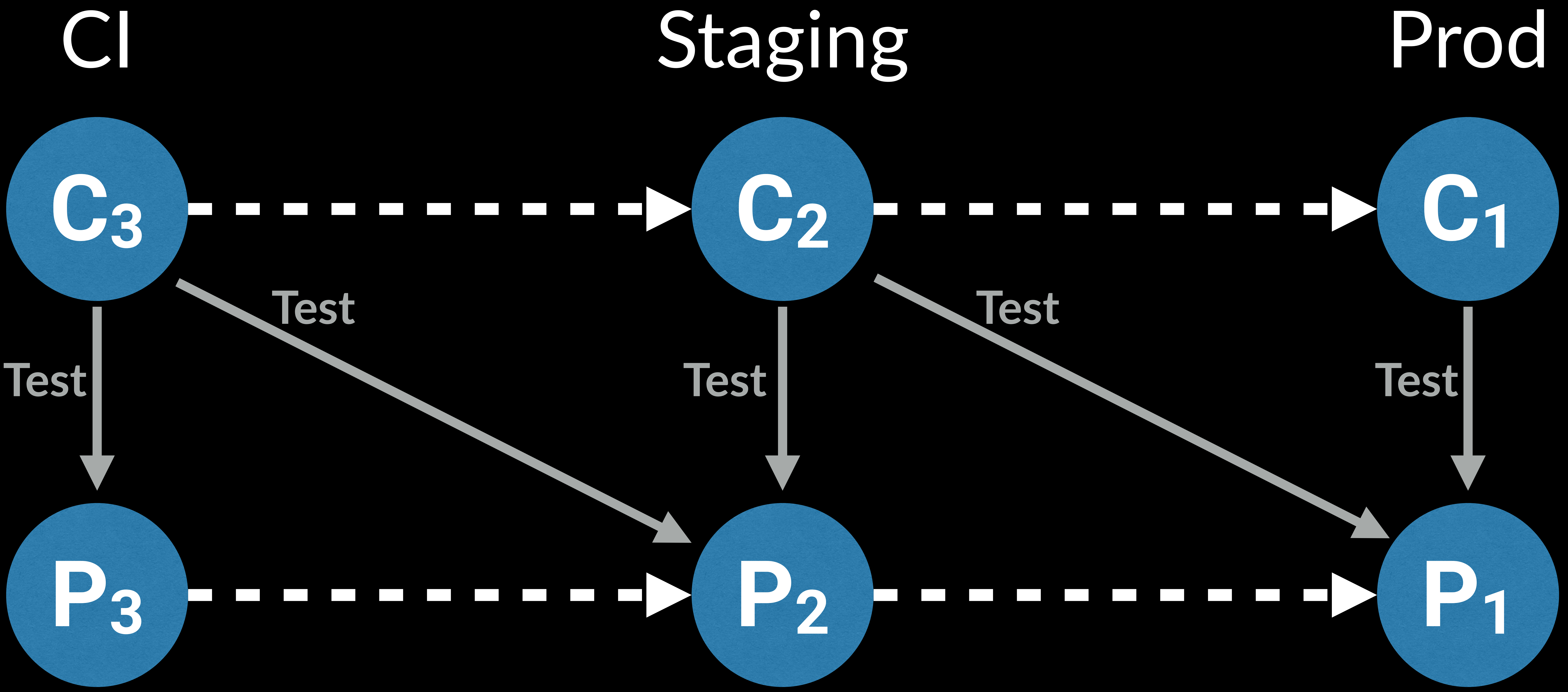
CI - consumer & provider



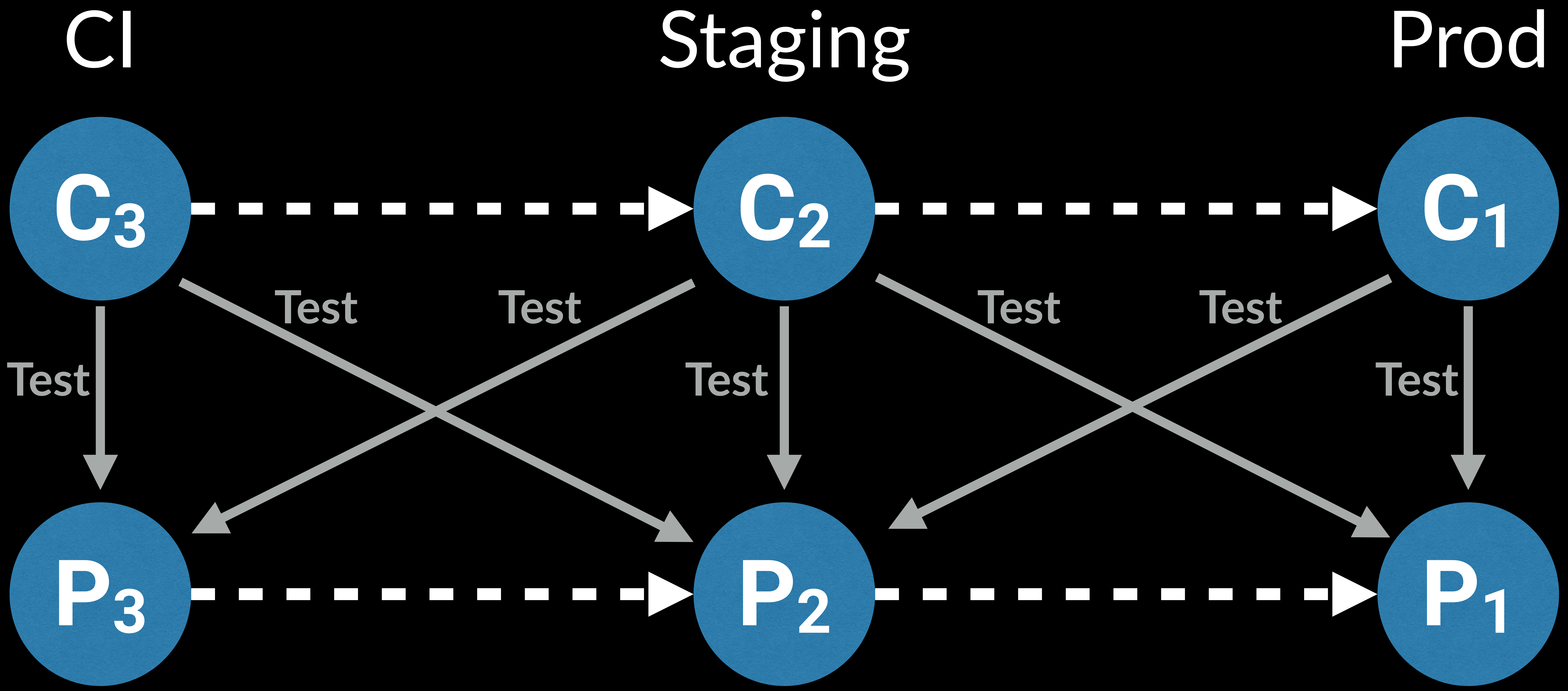
CI - consumer & provider


$$\begin{tikzcd}
\text{CI} & \text{Staging} & \text{Prod} \\
C_3 \arrow[r, dashed] \arrow[d, "\text{Test}"'] & C_2 \arrow[r, dashed] \arrow[d, "\text{Test}"'] & C_1 \arrow[d, "\text{Test}"'] \\
P_3 \arrow[r, dashed] & P_2 \arrow[r, dashed] & P_1
\end{tikzcd}$$

CI - consumer & provider



CI - consumer & provider



So what?

- Micro-services MUST be independently deployable
- There are many environments in a deployment pipeline
- Many micro-services leads to the need for many compatibility tests between components in many environments

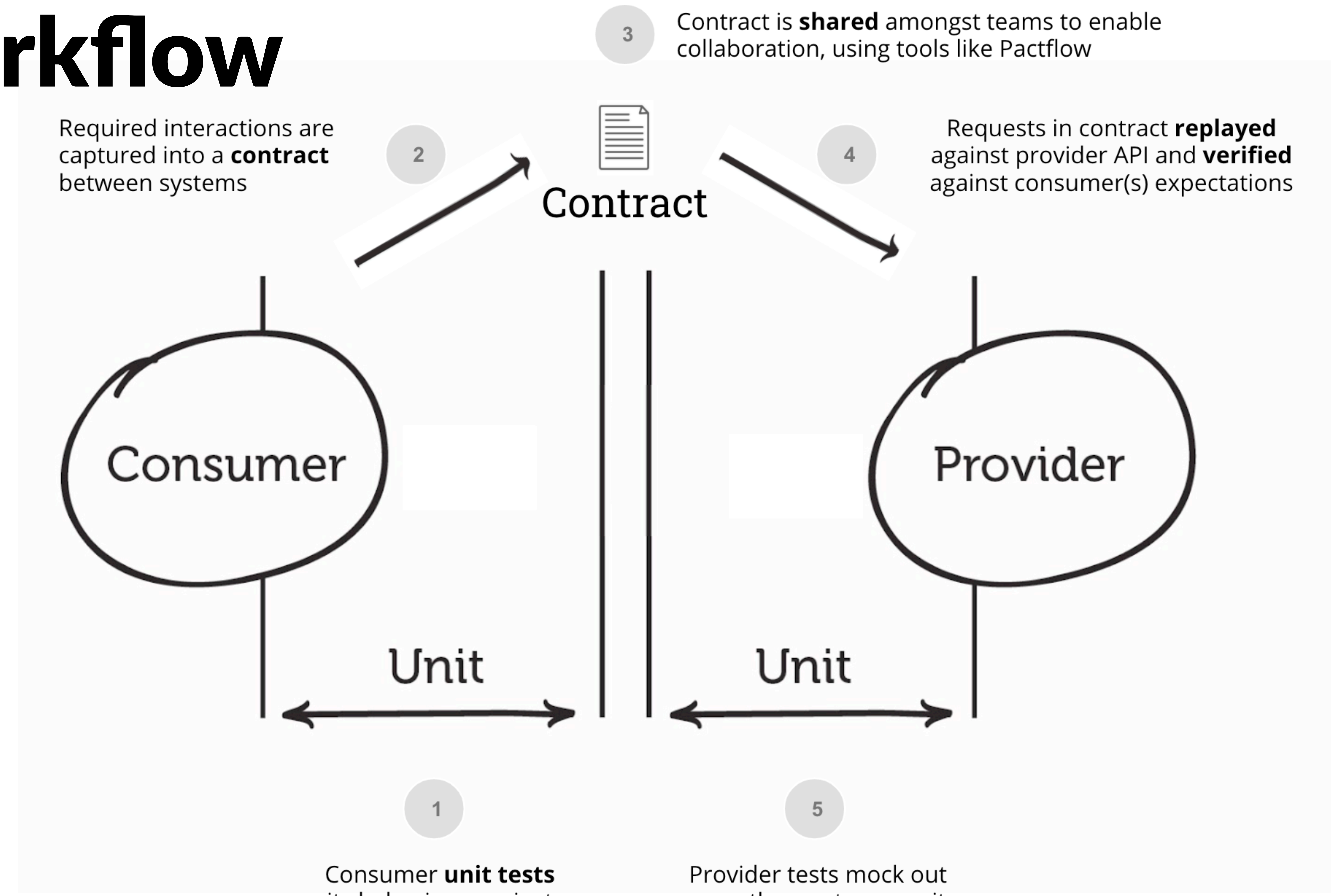
Agenda

Introduction
Why software contracts?
Traditional software testing
A taxonomy of contracts (partial)
Handrolled contract tests
Documenting the contract
Micro-services (finally)
Pact, PactBroker and more
“Bidirectional” contract testing
Questions

Simplifying consumer driven contracts

Pact provides a mechanism for creating a contract between a service consumer and a service provider, and then providing the tools to validate that the consumer and provider adhere to the contact independently of each other.

Pact workflow


3
Contract is **shared** amongst teams to enable collaboration, using tools like Pactflow
2
Required interactions are captured into a **contract** between systems
Contract
4
Requests in contract **replayed** against provider API and **verified** against consumer(s) expectations
Consumer
Provider
Unit
Unit
1
5

Consumer **unit tests** its behaviour against provider mock

Provider tests mock out any other systems, so it can be **tested in isolation**

@sebrose

https://docs.pact.io/

```
[Fact]
public void ItHandlesNoData()
{
    // Arrange
    _mockProviderService.Given("There is no data")
                        .UponReceiving("A valid GET request for Date Validation")
                        .With(new ProviderServiceRequest
                        {
                            Method = HttpVerb.Get,
                            Path = "/api/provider",
                            Query = "validDateTime=04/04/2018"
                        })
                        .WillRespondWith(new ProviderServiceResponse {
                            Status = 404
                        });

    // Act
    var result = ConsumerApiClient.ValidateDateTimeUsingProviderApi("04/04/2018", _mockProviderService
    var resultStatus = (int)result.StatusCode;

    // Assert
    Assert.Equal(404, resultStatus);
}
```

https://github.com/pact-foundation/pact-workshop-dotnet-core-v1

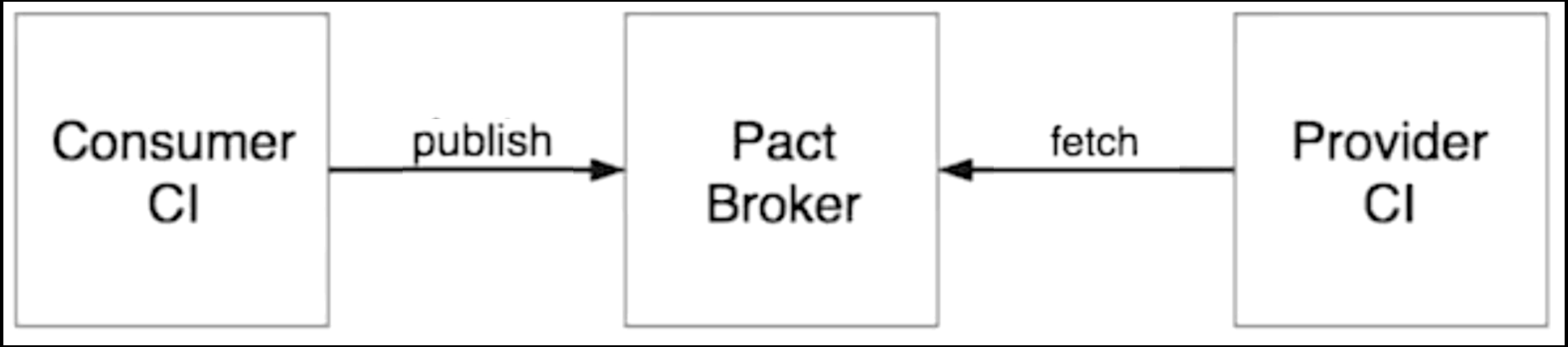
```
{
  "description": "A valid GET request for Date Validation",
  "providerState": "There is no data",
  "request": {
    "method": "get",
    "path": "/api/provider",
    "query": "validDateTime=04/04/2018"
  },
  "response": {
    "status": 404,
    "headers": {
    }
  }
},
```

https://github.com/pact-foundation/pact-workshop-dotnet-core-v1

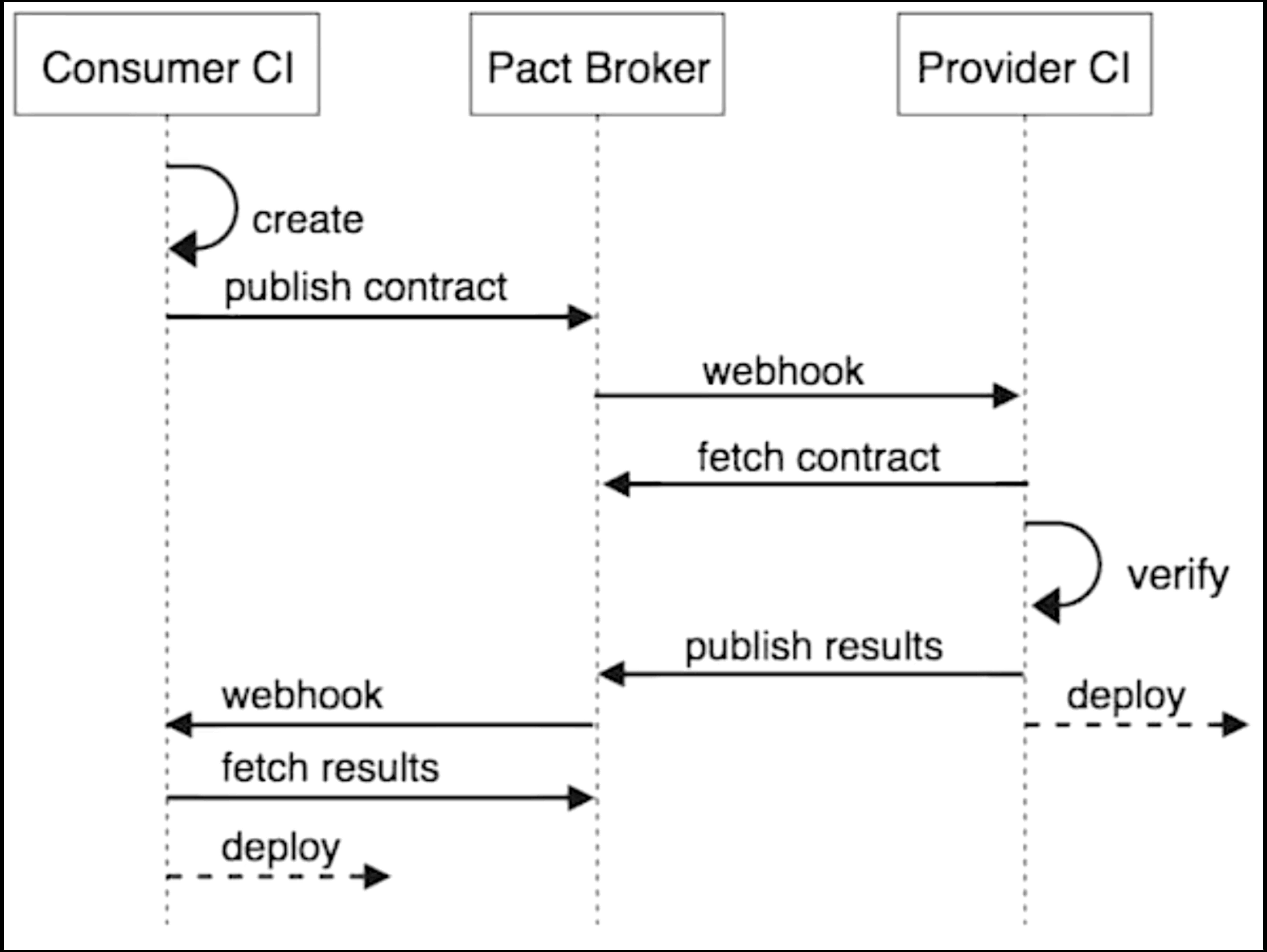
Pact - key points

- Consumer creates contracts using Pact DSL
- When consumer tests are run:
  - Pact creates a mock HTTP server
  - a Pact file is created
- Provider uses Pact file to verify compatibility
- Provider will need to provide “known” test data

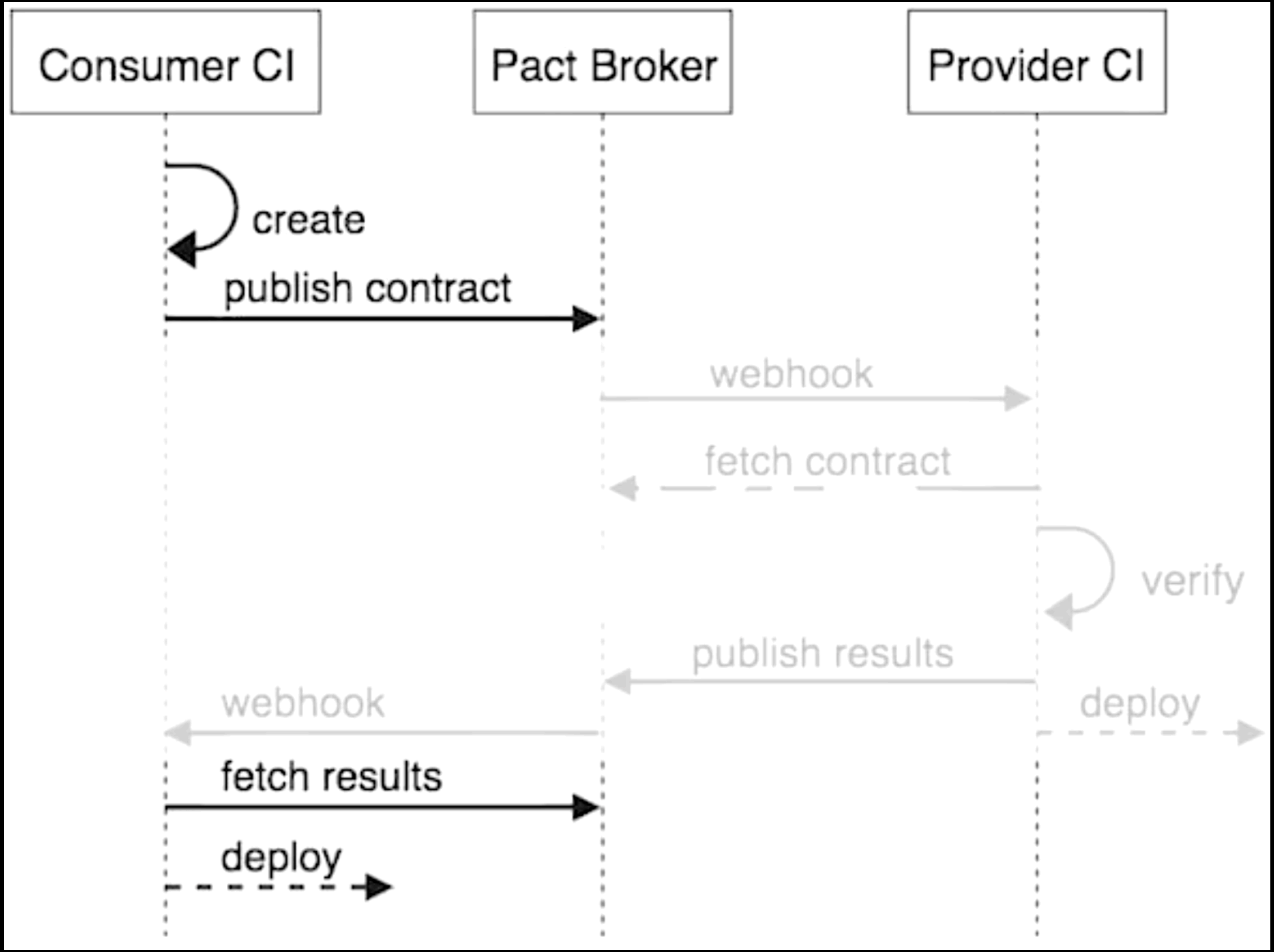
Pact broker



https://www.youtube.com/watch?v=79GKBYSqMIo

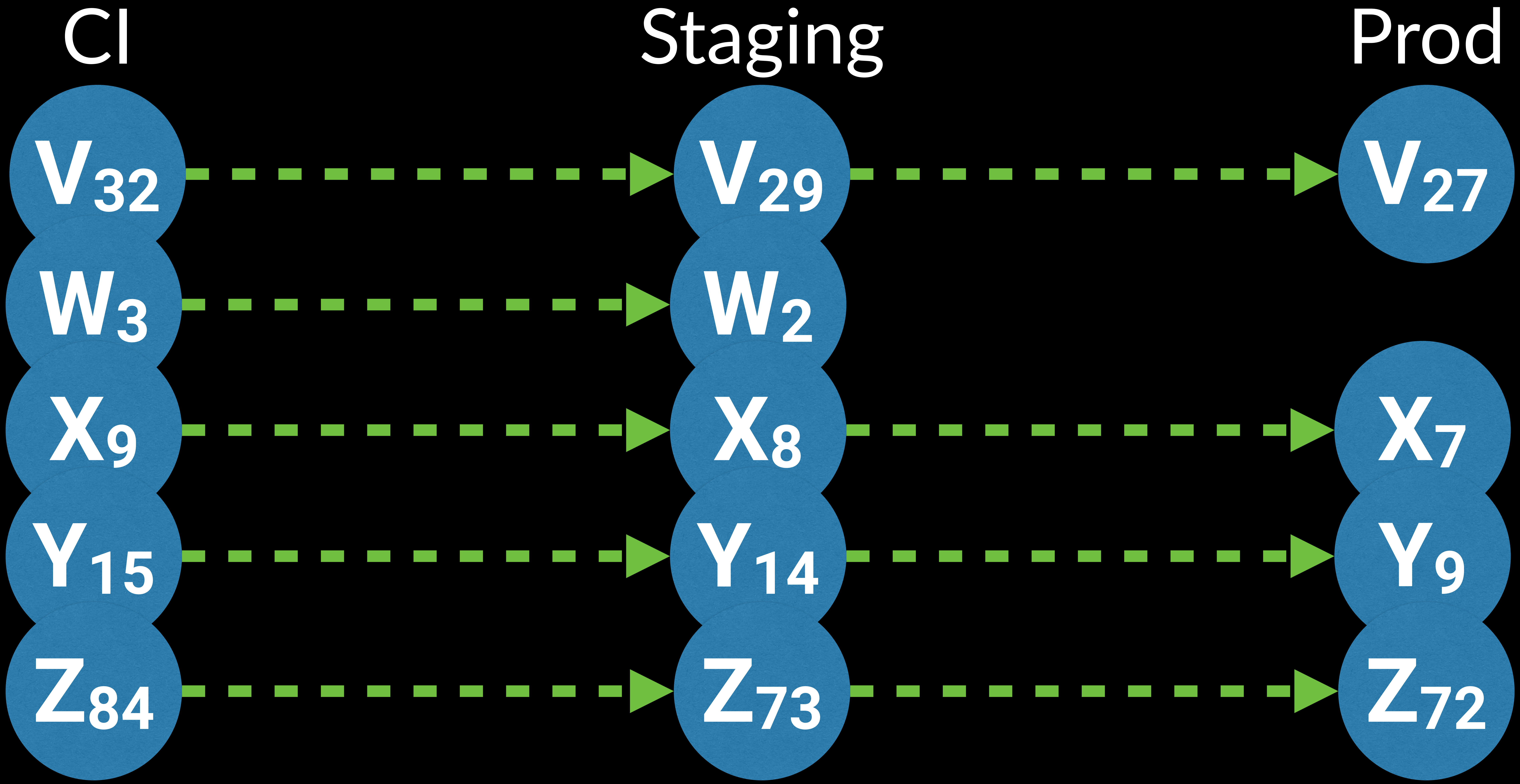


https://www.youtube.com/watch?v=79GKBYSqMIo

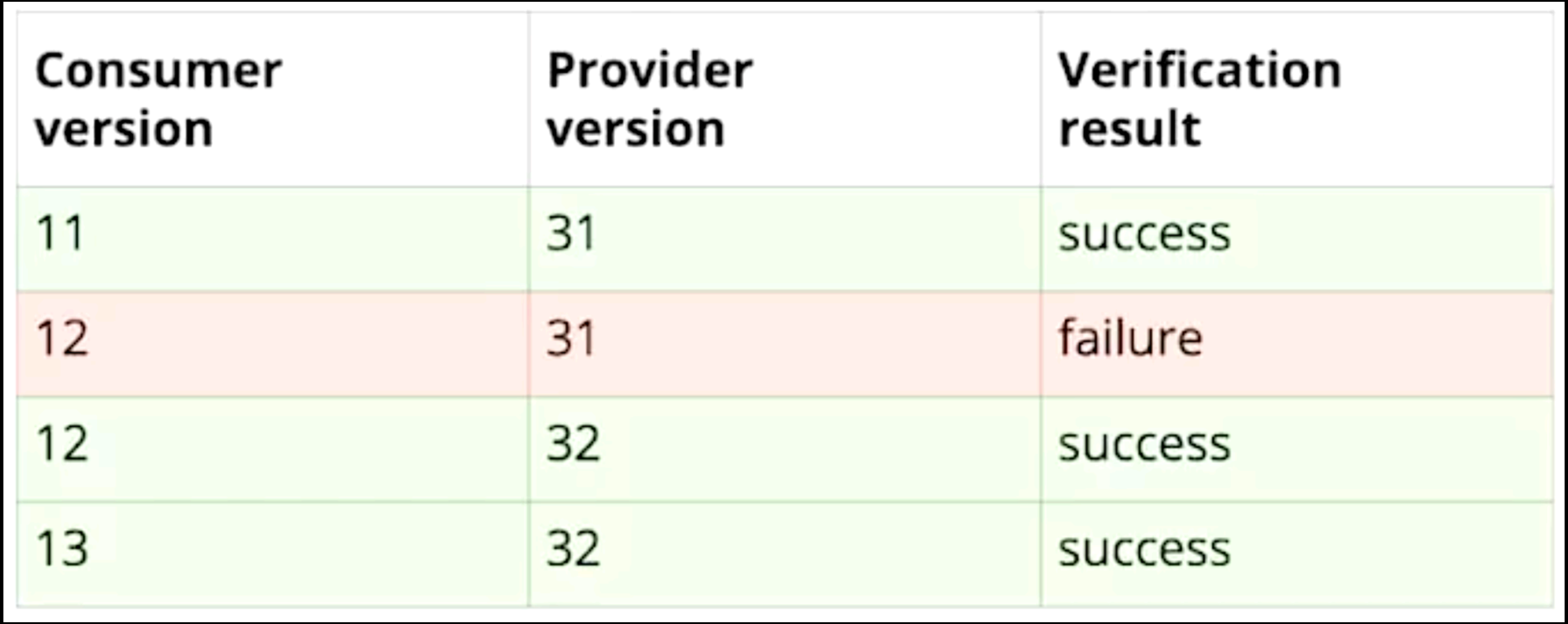


https://www.youtube.com/watch?v=79GKBYSqMIo

Complications



Pact broker "Matrix"



Consumer version	Provider version	Verification result
11	31	success
12	31	failure
12	32	success
13	32	success

https://www.youtube.com/watch?v=79GKBYSqMIo

`can-i-deploy`

Pact broker - key points

- Pacts are published by Consumer
- Pacts are fetched by Provider
- Results are stored in the “Matrix”
- “Matrix” supports independent deployment

So what?

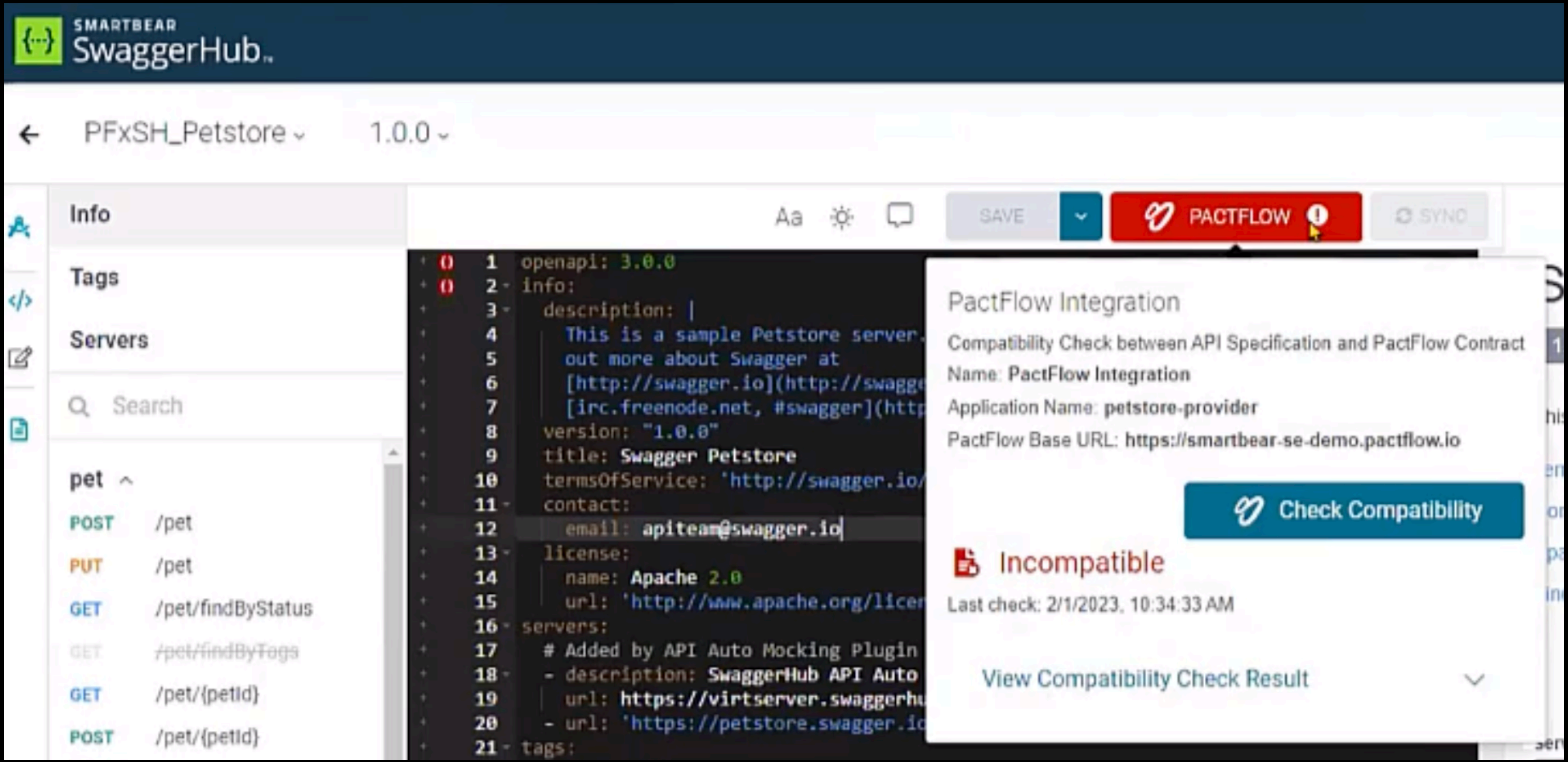
- Pact is a free, open source tool
- Pact provides the DSL and runtime to make the cost of contract testing manageable
- `can-i-deploy` helps with deployment
- The provider team needs to provide known test data states

Agenda

Introduction
Why software contracts?
Traditional software testing
A taxonomy of contracts (partial)
Handrolled contract tests
Documenting the contract
Micro-services (finally)
Pact, PactBroker and more
“Bidirectional” contract testing
Questions

PACTFLOW
A SMARTBEAR COMPANY

Bi-directional contract testing


SMARTBEAR SwaggerHub

PFxSH_Petstore 1.0.0

Info
Tags
Servers
Search

pet
POST /pet
PUT /pet
GET /pet/findByStatus
~~GET /pet/findByTags~~
GET /pet/{petId}
POST /pet/{petId}

SAVE PACTFLOW SYNC

```
openapi: 3.0.0
info:
  description: |
    This is a sample Petstore server.
    out more about Swagger at
    [http://swagger.io](http://swagge
    [irc.freenode.net, #swagger](http
  version: "1.0.0"
  title: Swagger Petstore
  termsOfService: 'http://swagger.io/
  contact:
    email: apiteam@swagger.io
  license:
    name: Apache 2.0
    url: 'http://www.apache.org/licen
servers:
  # Added by API Auto Mocking Plugin
  - description: SwaggerHub API Auto
    url: https://virtserver.swaggerhu
  - url: 'https://petstore.swagger.io
tags:
```

PactFlow Integration
Compatibility Check between API Specification and PactFlow Contract
Name: PactFlow Integration
Application Name: petstore-provider
PactFlow Base URL: https://smartbear-se-demo.pactflow.io

Check Compatibility

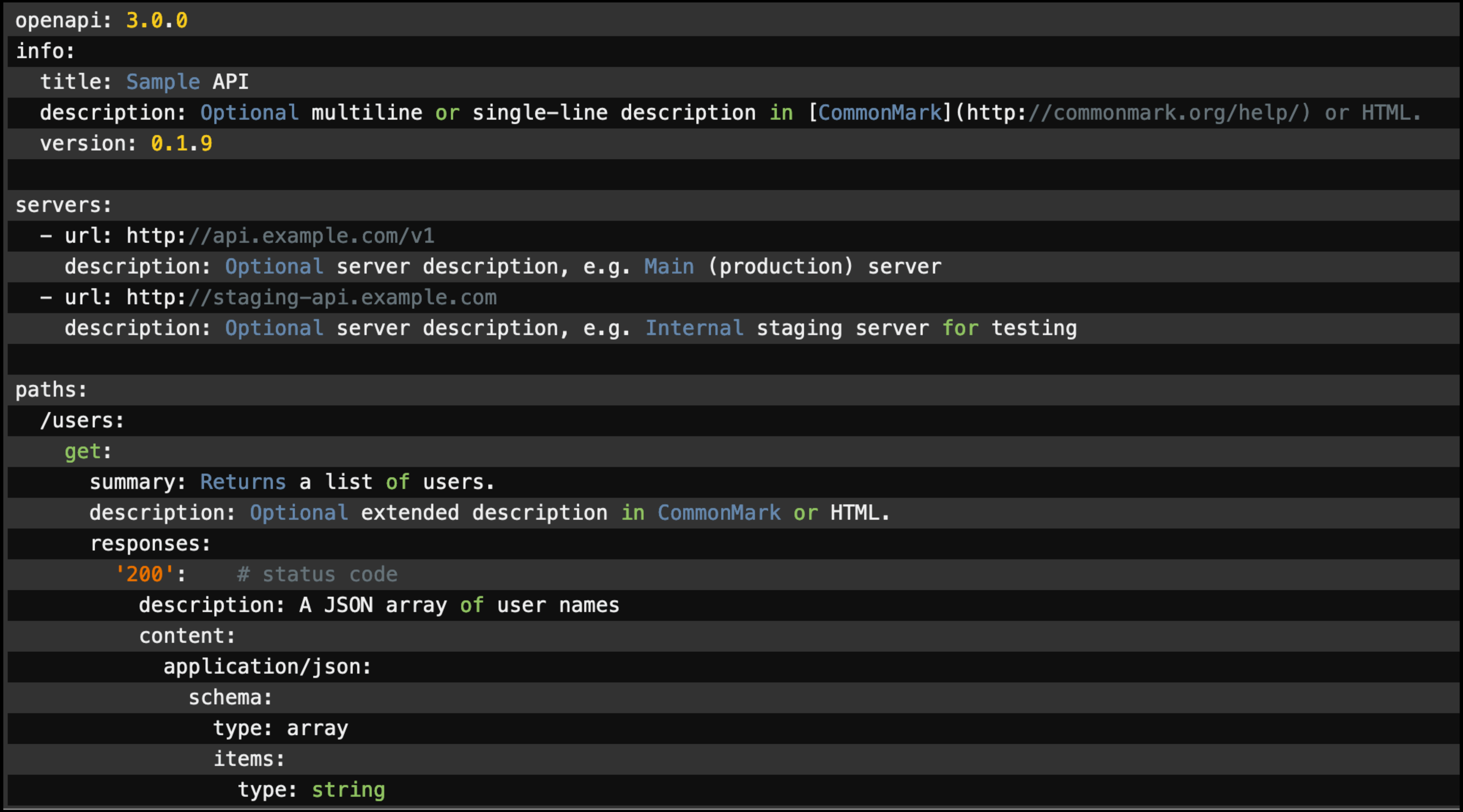
Incompatible
Last check: 2/1/2023, 10:34:33 AM

View Compatibility Check Result

https://github.com/pactflow/swagger-mock-validator

@sebrose https://pactflow.io/bi-directional-contract-testing http://claysnow.co.uk

OpenAPI specification



```
openapi: 3.0.0
info:
  title: Sample API
  description: Optional multiline or single-line description in [CommonMark](http://commonmark.org/help/) or HTML.
  version: 0.1.9

servers:
  - url: http://api.example.com/v1
    description: Optional server description, e.g. Main (production) server
  - url: http://staging-api.example.com
    description: Optional server description, e.g. Internal staging server for testing

paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in CommonMark or HTML.
      responses:
        '200':    # status code
          description: A JSON array of user names
          content:
            application/json:
              schema:
                type: array
                items:
                  type: string
```

https://swagger.io/docs/specification/basic-structure/

So what?

- Pactflow (PF) is a commercial tool
- Bidirectional contract testing (BDC) statically compares Pact files to OpenAPI specifications
- BDC reduces the burden on the provider team, but with weaker guarantees
- BDC can be achieved without using PF

Take aways

Take aways

- All interactions between software components are governed by contracts

Take aways

- All interactions between software components are governed by contracts
- Contract testing ensures that both components have the same expectations

Take aways

- All interactions between software components are governed by contracts
- Contract testing ensures that both components have the same expectations
- Contract tests should be written by the developers

More take aways

Contract testing:

More take aways

Contract testing:

- Increases agility and confidence

More take aways

Contract testing:

- Increases agility and confidence
- Reduces need for integration tests

More take aways

Contract testing:

- Increases agility and confidence
- Reduces need for integration tests
- Speeds up development

More take aways

Contract testing:

- Increases agility and confidence
- Reduces need for integration tests
- Speeds up development

- No substitute for communication

More take aways

Contract testing:

- Increases agility and confidence
- Reduces need for integration tests
- Speeds up development

- No substitute for communication
- Is a development activity

More take aways

Contract testing:

- Increases agility and confidence
- Reduces need for integration tests
- Speeds up development

- No substitute for communication
- Is a development activity
- Does not replace other forms of testing

QUESTIONS?

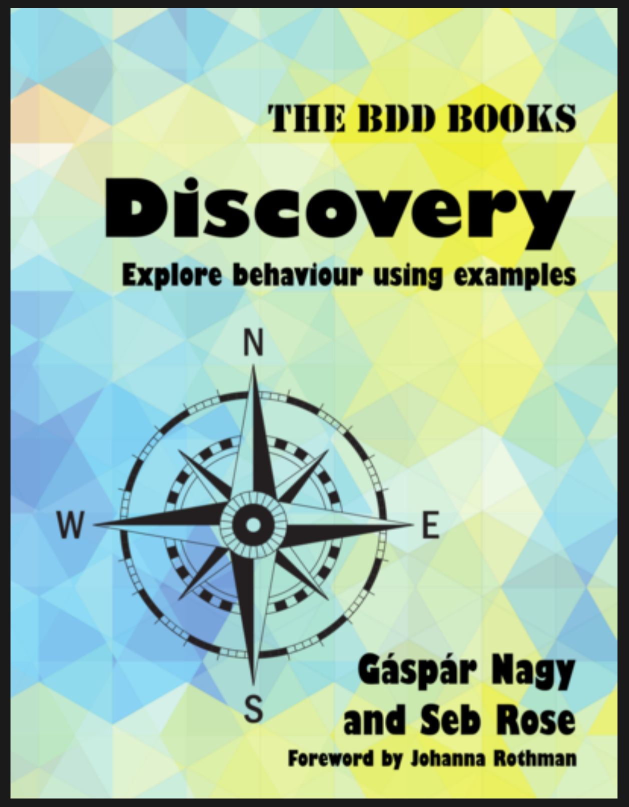
Seb Rose

Mastodon: @sebrose@mastodon.scot

Twitter: @sebrose

Blog: https://claysnow.co.uk

E-mail: seb@claysnow.co.uk



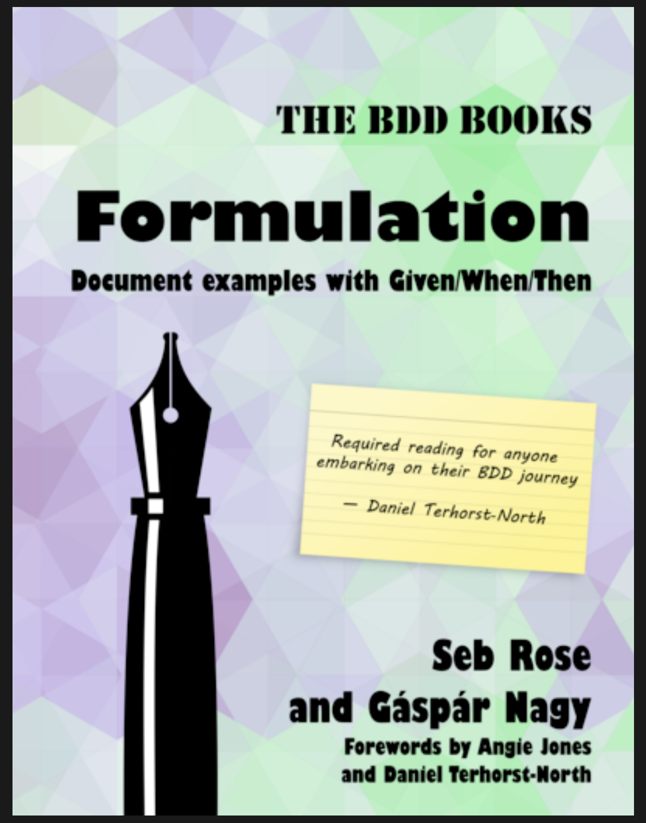
THE BDD BOOKS

Discovery

Explore behaviour using examples

Gáspár Nagy
and Seb Rose

Foreword by Johanna Rothman



THE BDD BOOKS

Formulation

Document examples with Given/When/Then

Required reading for anyone embarking on their BDD journey

— Daniel Terhorst-North

Seb Rose
and Gáspár Nagy

Forewords by Angie Jones
and Daniel Terhorst-North

https://bddbooks.com