Consumer-Driven Contracts

Markus Knittig
@mknittig

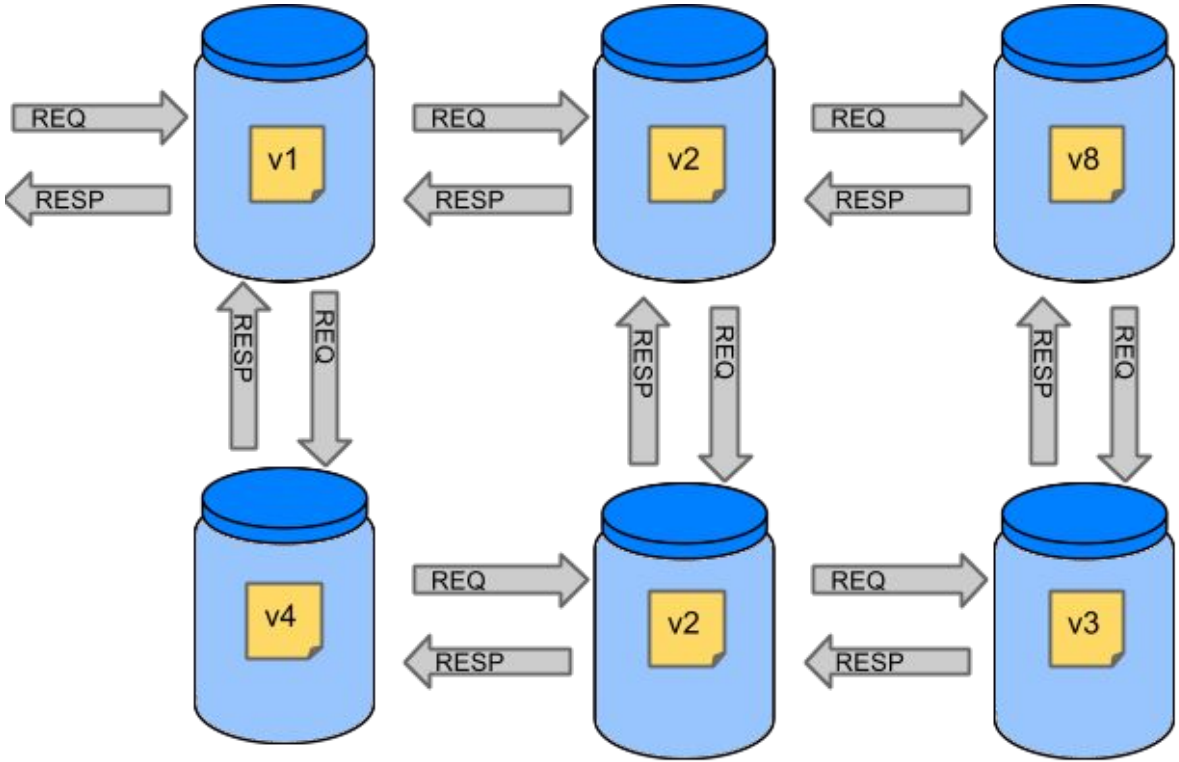
Agenda

- Fundamentals
- How CDC works
- What CDC can do and not do

Service-oriented Architecture

Microservices

Microservices Architecture



Service Consumer

Service Provider


CONTRACT

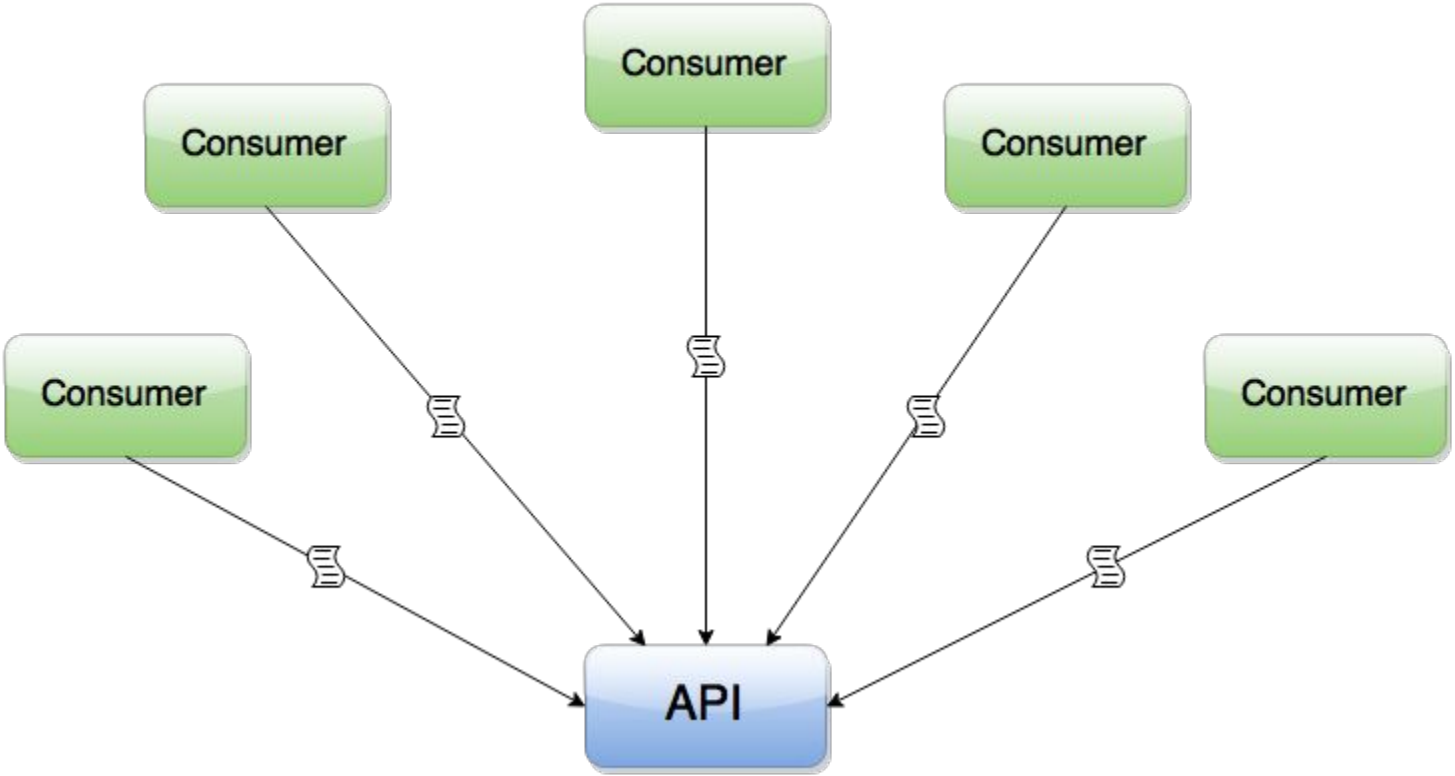
Provider Contracts

- Document schemas
- Interfaces
- Conversations
- Policy
- Quality of service characteristics

… and Consumer Contracts?

Consumer-Driven

Contracts



Pact and Spring Cloud Contracts to the rescue!

Example with Spring Cloud Contracts

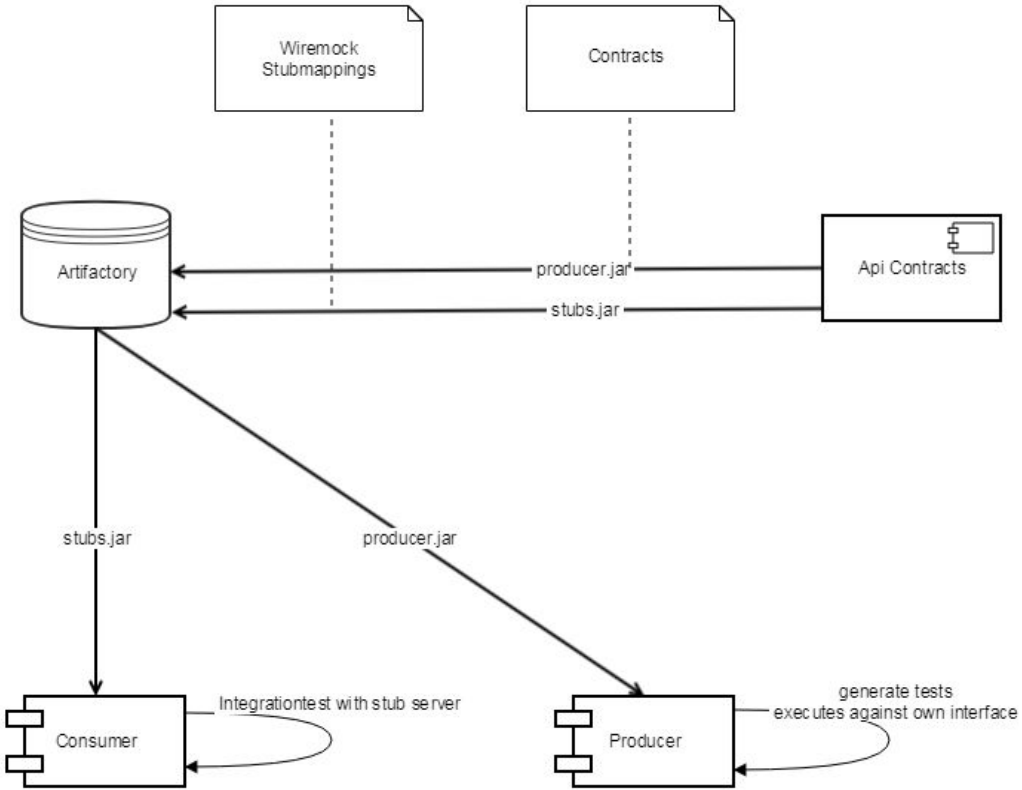
```
package contracts

org.springframework.cloud.contract.spec.Contract.make {
    request {
        method 'GET'
        url '/uaa/v1/me'
        headers {
            header('Content-Type': consumer(regex('application/*json*')))
        }
    }
    response {
        status 200
        body([
                username    : value(producer(regex('[A-Za-z0-9]+'))),
                firstName   : value(producer(regex('[A-Za-z]+'))),
                lastName    : value(producer(regex('[A-Za-z]+'))),
                email       : value(producer(regex('[A-Za-z0-9]+\\@[A-Za-z0-9]+\\.[A-Za-z]+'))),
                createdAt   : value(producer(regex('[0-9]+'))),
                lastModified: value(producer(regex('[0-9]+'))),
                id          : value(producer(regex('[0-9]+')))
        ])
        headers {
            header('Content-Type': value(
                    producer('application/json;charset=UTF-8'),
                    consumer('application/json;charset=UTF-8'))
            )
        }
    }
}
```

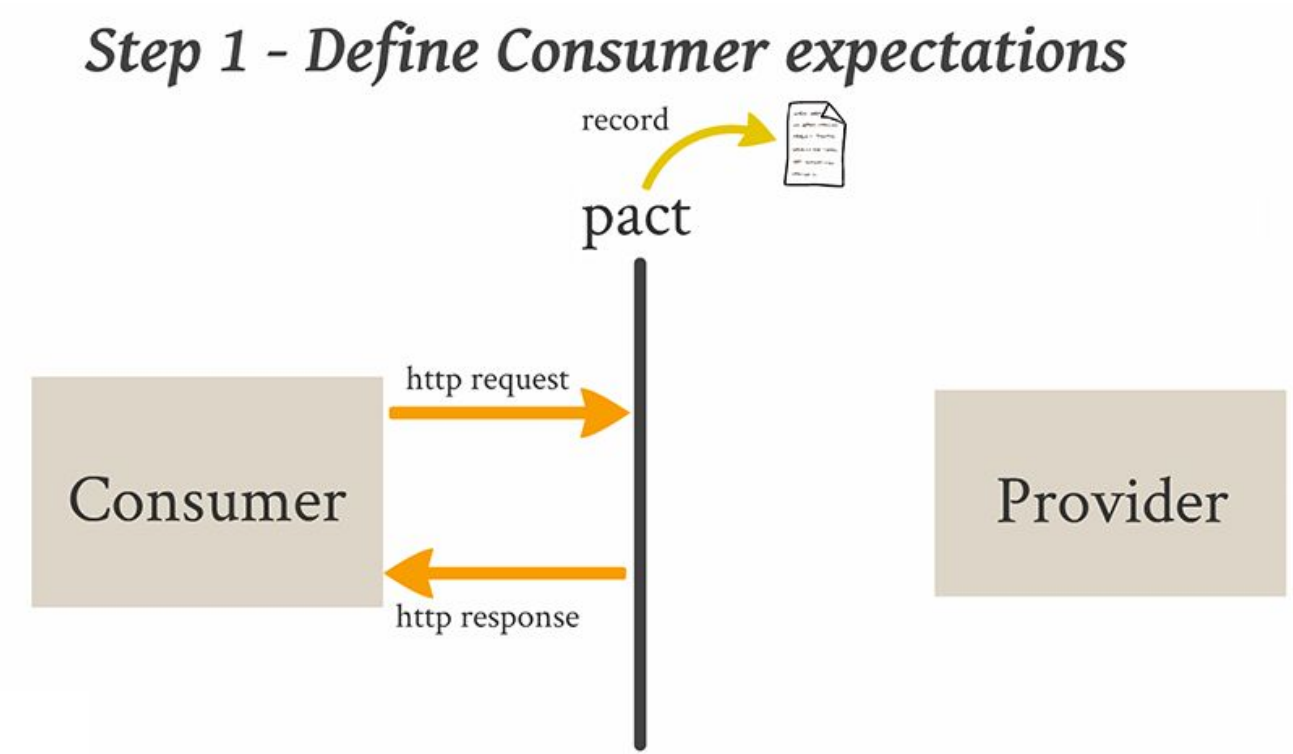
Example with Pact

```
animal_service.given("an alligator named Mary exists").
 upon_receiving("a request for an alligator").
 with(
   method: "get",
   path: "/alligators/Mary",
   headers: {"Accept" => "application/json"}).
 will_respond_with(
   status: 200,
   headers: {"Content-Type" => "application/json"},
   body: {
     name: "Mary",
     dateOfBirth: Pact.term(
       generate: "02/11/2013",
       matcher: /\d{2}\/\d{2}\/\d{4}/)
   })
```

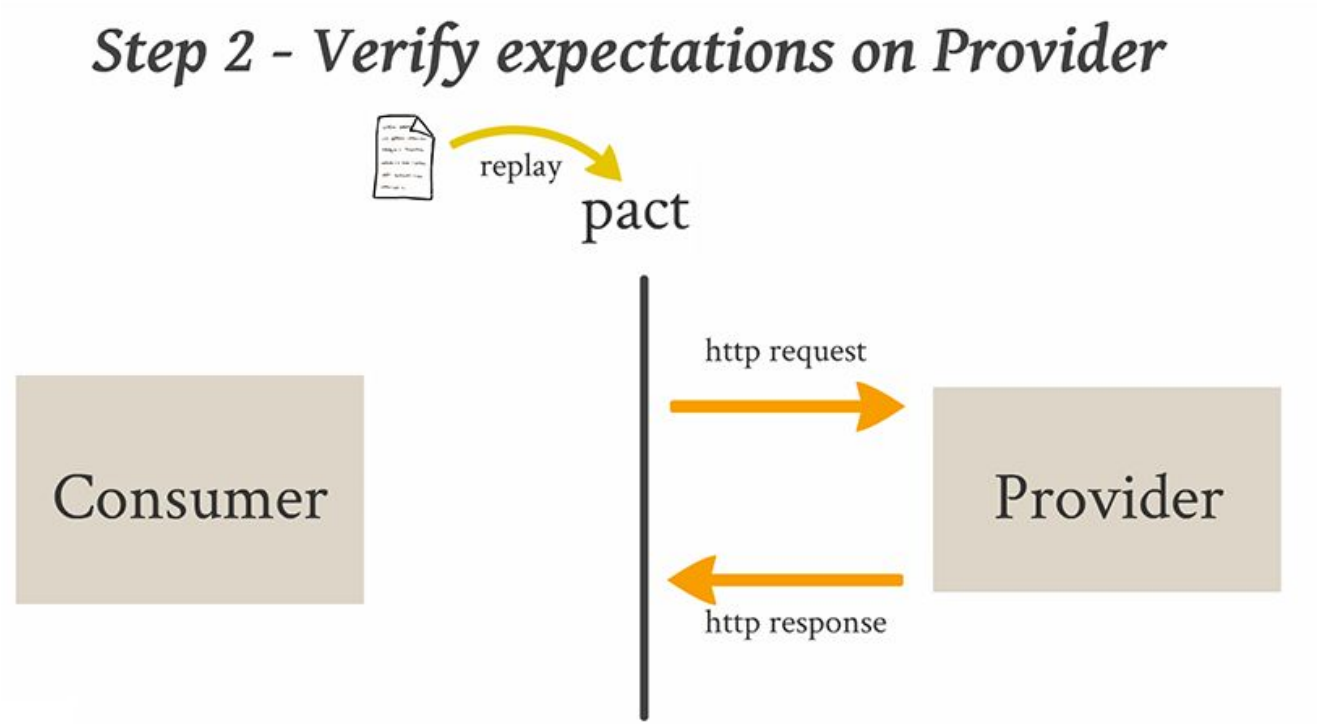
Overview with Spring Cloud Contracts



Consumer view with Pact



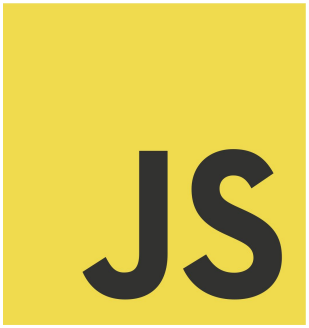
Provider view with Pact

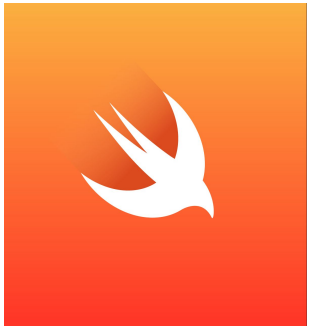


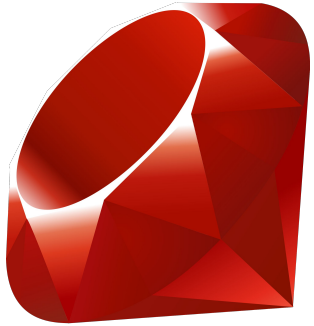
Should I use Pact or Spring Cloud Contracts?

Simple: Use Spring Cloud Contracts if you want to test Spring (Boot) projects (you can consume Pact files too)

Pact Implementations







Share contracts

Pact Broker

- Matrix of compatible consumer/provider versions
- Tagging of Pact versions e.g. “production”, “feature-branch”
- Hooks when Pact changes e.g. run provider verifications
- Provider verification results
- Diffs between Pact versions
- And some more...

Pact Broker Index

Pacts

Consumer	Provider	Latest pact published	Last verified
Foo	Animals	2 minutes ago	2 days ago
Foo	Bar	7 days ago	15 days ago ⚠
Foo	Hello World App	1 day ago	
Foo	Wiffles	less than a minute ago	7 days ago
Some other app	A service	26 days ago	less than a minute ago
The Android App	The back end	less than a minute ago	

Pact Broker Contract Details

A pact between Zoo App and Animal Service

Zoo App version: 1.0.0

Date published: 11/11/2014 8:56PM +11:00

View in HAL Browser

Requests from Zoo App to Animal Service

- A request for an alligator given there is an alligator named Mary
- A request for an alligator given there is not an alligator named Mary
- A request for an alligator given an error occurs retrieving an alligator

Interactions

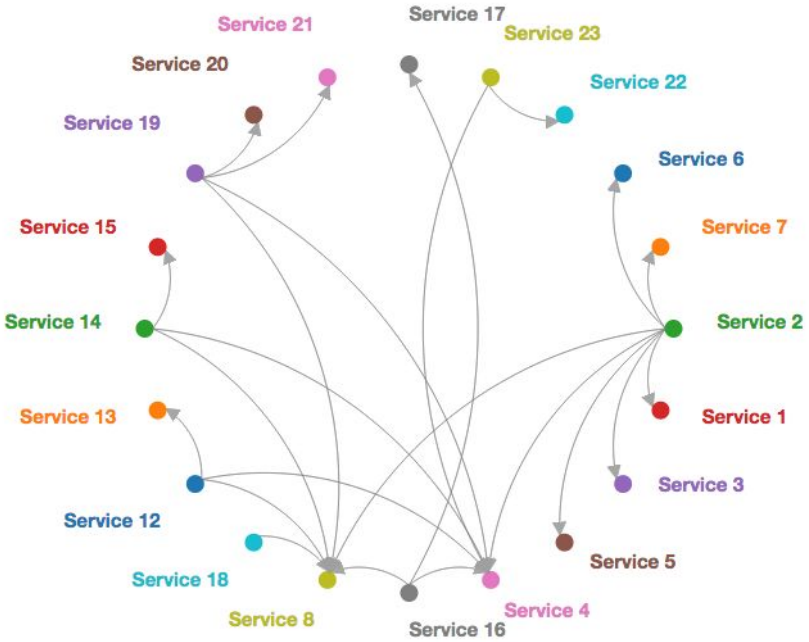
Given **there is an alligator named Mary**, upon receiving **a request for an alligator** from Zoo App, with

```
{
  "method": "get",
  "path": "/alligators/Mary",
  "headers": {
    "Accept": "application/json"
  }
}
```

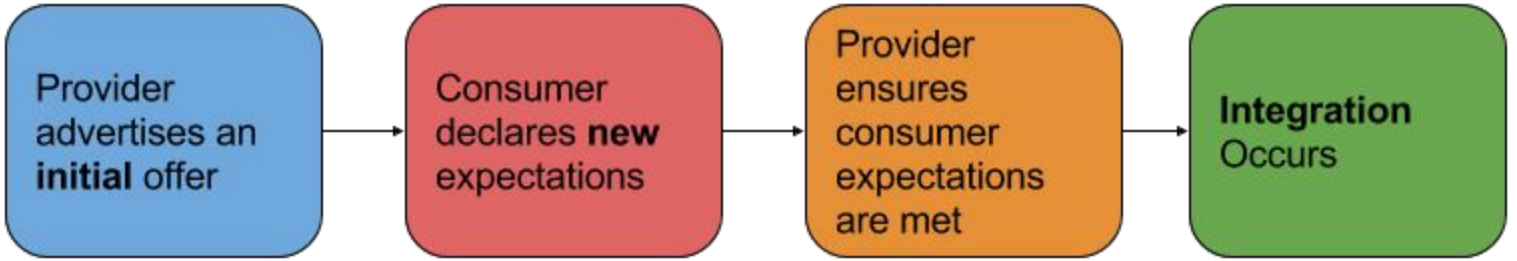
Animal Service will respond with:

```
{
  "status": 200,
  "headers": {
    "Content-Type": "application/json;charset=utf-8"
```

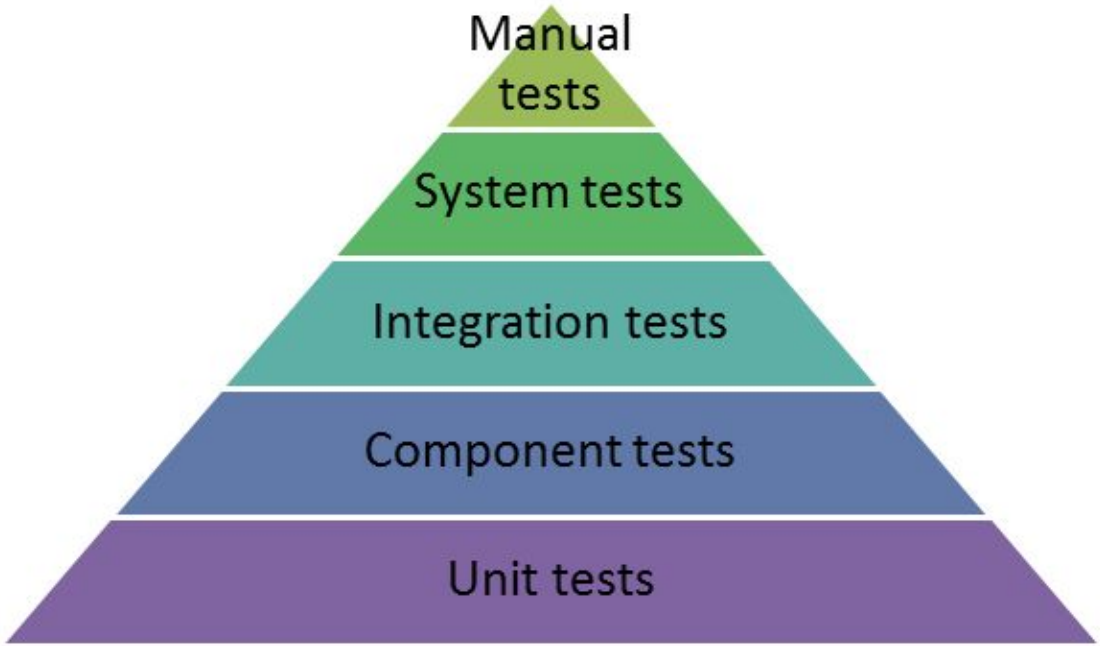
Pact Broker Dependency Graph

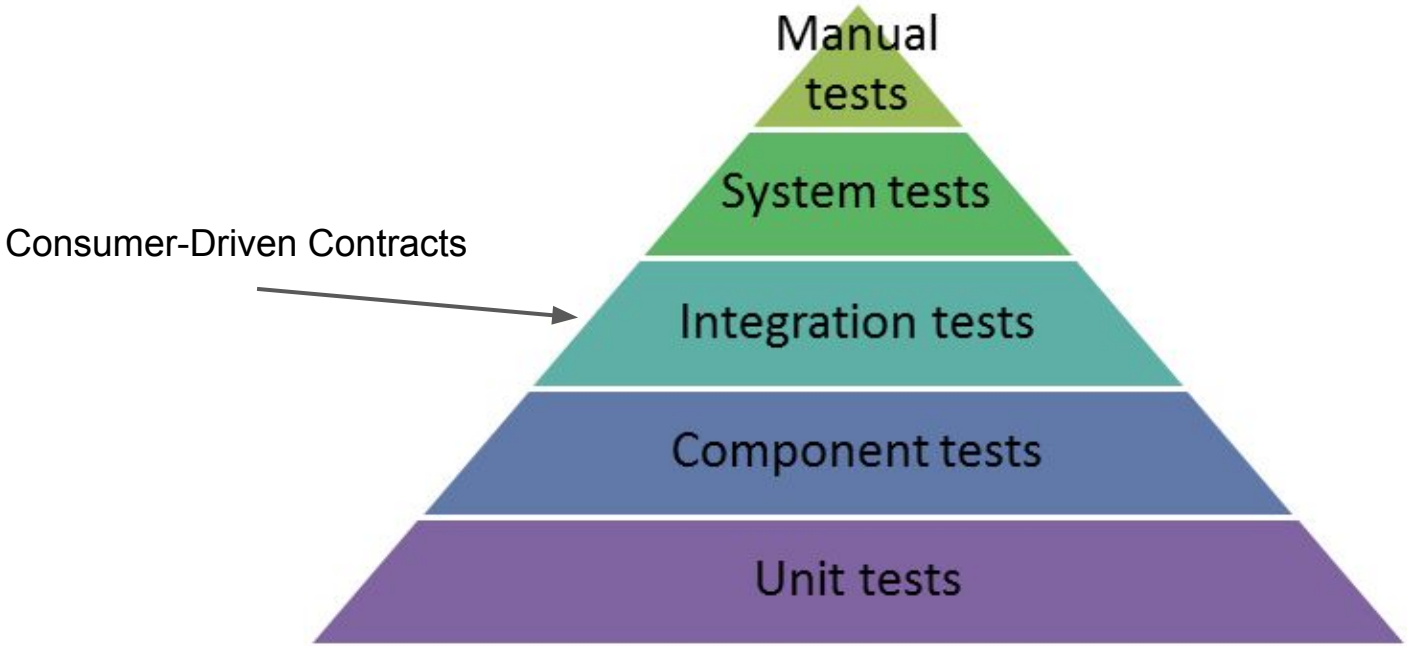


(Possible) CDC Flow



Contract Tests vs Functional Tests


Manual tests
System tests
Integration tests
Component tests
Unit tests


Consumer-Driven Contracts
Manual tests
System tests
Integration tests
Component tests
Unit tests

```
Given "there is no user called Mary"
When "creating a user with username Mary"
 POST /users { "username": "mary", email: "...", ... }
Then
 Expected Response is 200 OK
```

```
Given "there is already a user called Mary"
When "creating a user with username Mary"
  POST /users { "username": "mary", email: "...", ... }
Then
  Expected Response is 409 Conflict
```

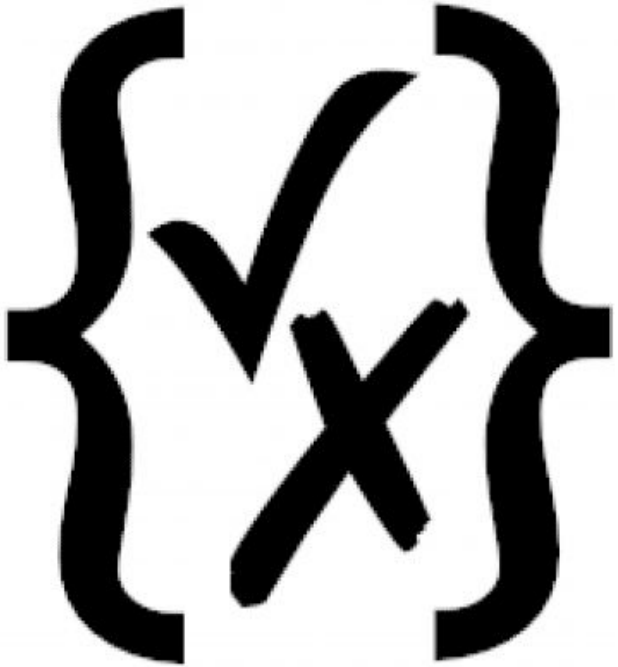
```
When "creating a user with a username with 21 characters"
  POST /users { "username": "thisisalooongusername", email: "...",... }
Then
  Expected Response is 400 Bad Request
  Expected Response body is { "error": "username cannot be more than 20 characters" }
```

```
When "creating a user with a username containing numbers"
  POST /users { "username": "us3rn4me", email: "...", ... }
Then
  Expected Response is 400 Bad Request
  Expected Response body is { "error": "username can only contain
letters" }
```

```
Given that username "bad_username" is invalid
When "creating a user with an invalid username"
  POST /users { "username": "bad_username", ... }
Then
  Response is 400 Bad Request
  Response body is { "error": "<any string>" }
```

CDC vs. Swagger

CDC vs. JSON Schema



JSON Schema

Where CDC can help

- Good tool for communication about interfaces between consumer and provider
- Simple structure / naming / type checks in request / response on lower test level
- Simple stubs for the consumer which can be used to use consumer even if the real implementation is missing

Do not use CDC...

- ...to replace end-to-end / system tests. “We have CDC! Why does integration break?”
- ...to test business logic. “For this request I expect two offerings, for this request one..”
- ...to test (master) data. “I need to ensure all different currencies are supported.”
- … to test cross-cutting concerns like security, token handling etc.

CDC Users


accenture
carsales
lastminute.com
portbase
NTT DATA
arquillian
shutterstock
itv
redhat
IBM
Pivotal Labs
Sainsbury's
SOUNDCLOUD

Thank you for your attention!
Questions and comments are welcome

Markus Knittig
@mknittig