

Collaborative Contract Driven Development

Billy Korando
Developer Advocate - IBM
@BillyKorando
william.korando@ibm.com

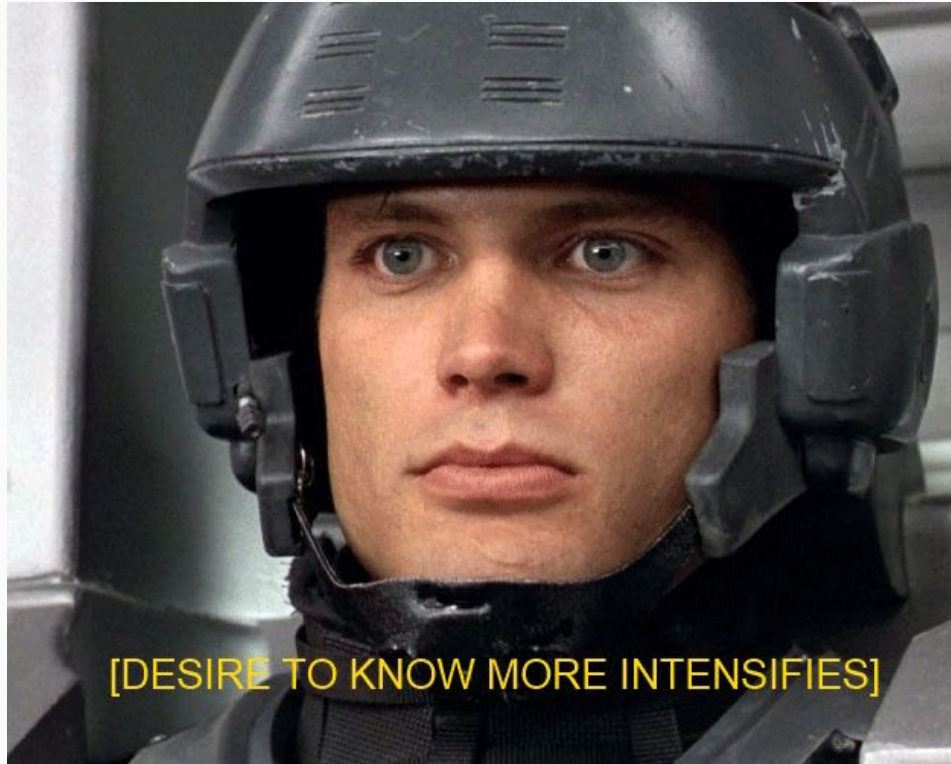


Subscribe to the Java Newsletter

<https://developer.ibm.com/newsletters/java/>

For your Spring & JakartaEE needs

<https://cloud.ibm.com/docs/java>



<https://billykorando.com/>

@BillyKorando

CONTRACT DRIVEN DEVELOPMENT DEFINED

(BRIEFLY)

DEFINING THE BEHAVIOR OF AN API

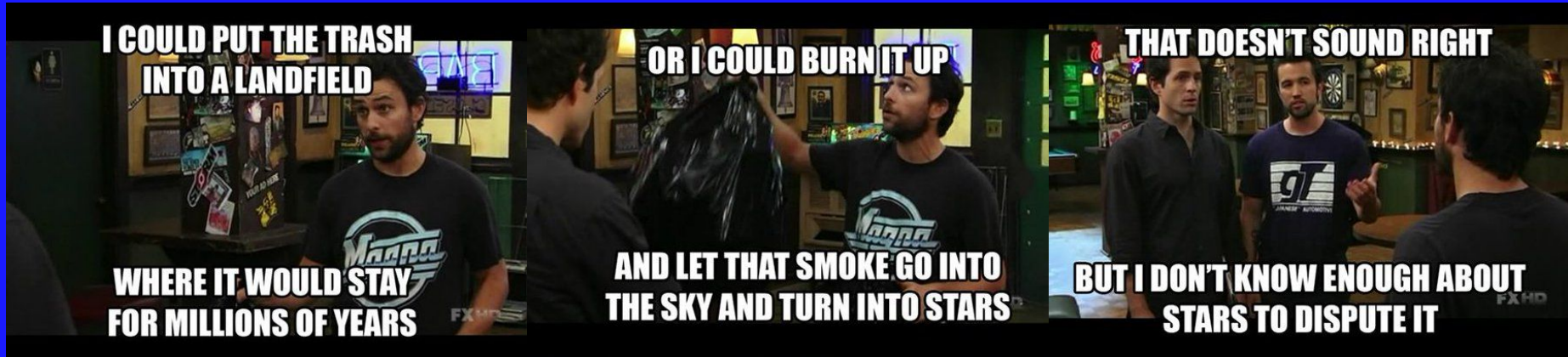


```
Response: 200  
{  
  ...  
}
```

```
Request /api/...  
{  
  ...  
}
```



DECONSTRUCTING THE STATUS QUO (i.e. WHY DEVELOPMENT IS DIFFICULT WITHOUT CONTRACTS)



POORLY DEFINED SERVICES



@BillyKorando

POORLY DEFINED SERVICES

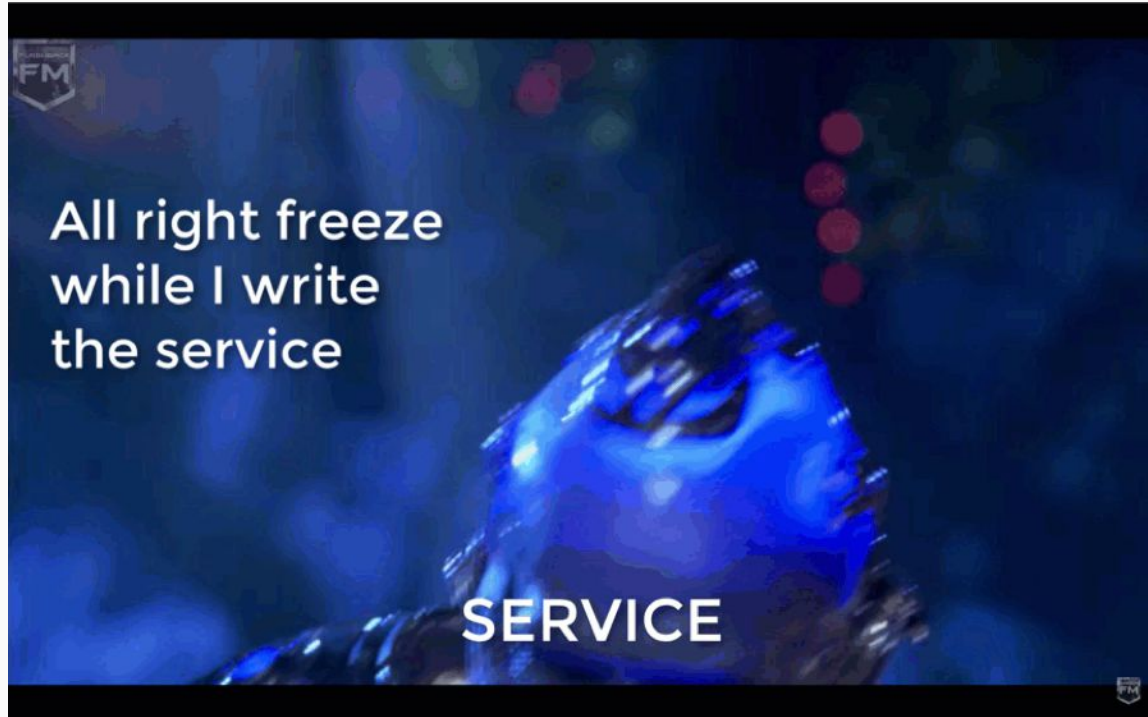


@BillyKorando

SCENE BREAKDOWN

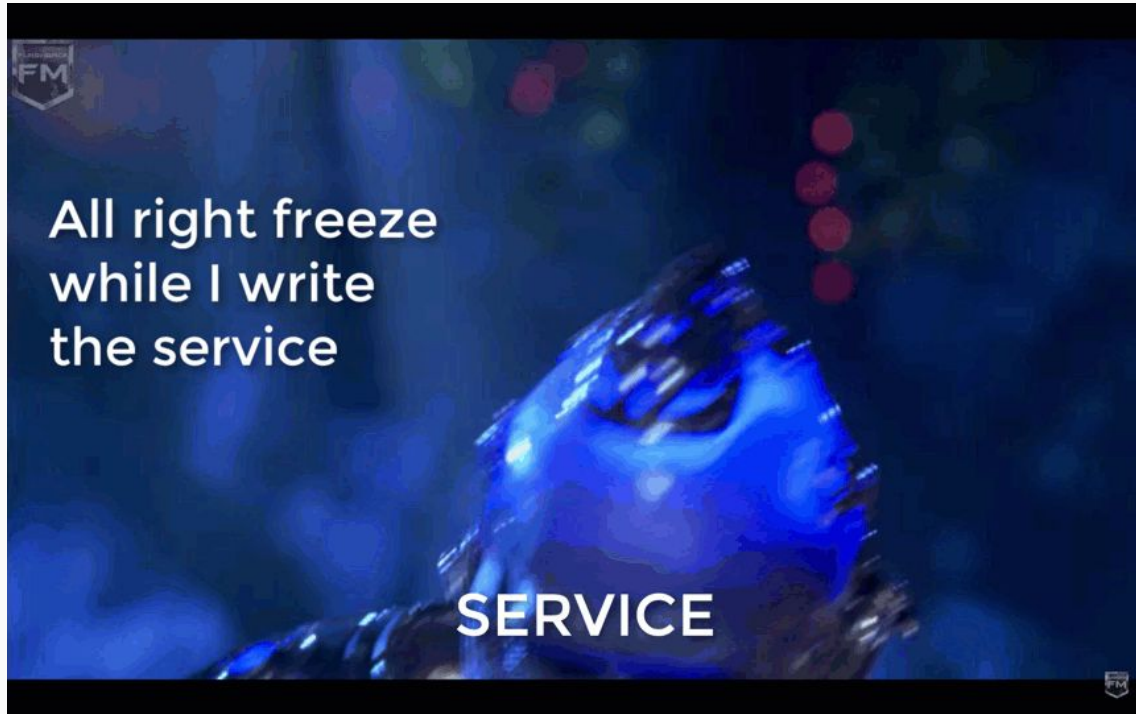
- Poorly defined or undocumented API
- Inconsistent behavior or patterns
- Testing against a live service can lead to inconsistent results

VERY SERIAL DEVELOPMENT



@BillyKorando

VERY SERIAL DEVELOPMENT



@BillyKorando

SCENE BREAKDOWN

- Inability to develop in parallel
- Timeline impacted by delays from both service and client developers
- Client developer will have to get back up to speed

I AM ALTERING THE API



@BillyKorando

I AM ALTERING THE API



@BillyKorando

SCENE BREAKDOWN

- Breaking API changes not caught until production
- API design governed almost entirely by service developer

CONTRACT DRIVEN DEVELOPMENT EXPLAINED USING SPRING CLOUD CONTRACT

@BillyKorando

SPRING CLOUD CONTRACT

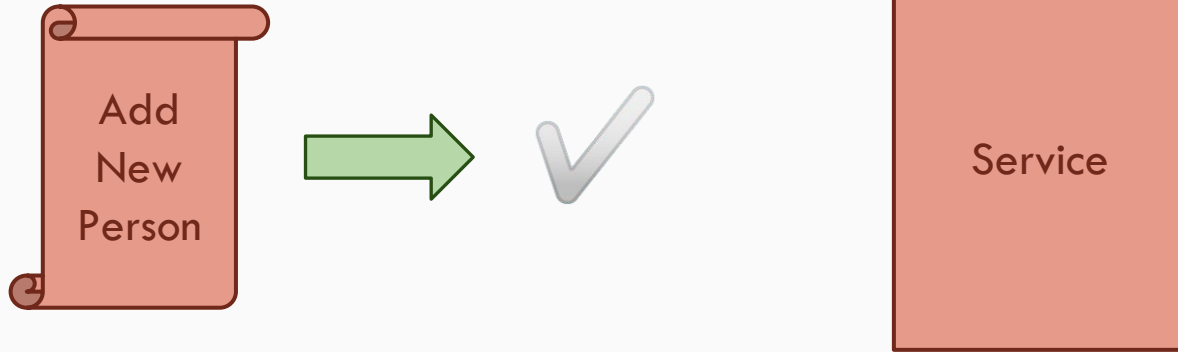
- Very active project
- Inter-operable with other tools/standards: OpenAPI, Pact, Swagger
- Flexible documentation support with Spring REST Docs
- Polyglot support

More: <https://spring.io/projects/spring-cloud-contract>

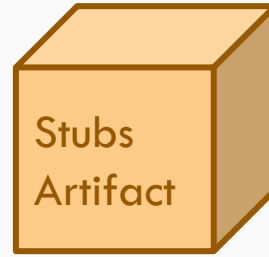
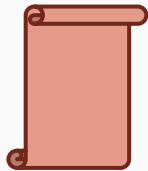
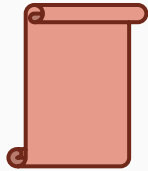
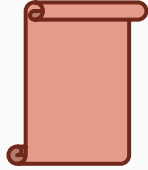
WRITING THE CONTRACT

```
Contract.make {
  label "Add New Person"
  request {
    method 'POST'
    url '/api/persons'
    body([ fName: "John"
         lName: "Doe"])
    headers {
      header(HttpHeaders.CONTENT_TYPE,
             MediaType.APPLICATION_JSON_UTF8_VALUE)}
  } response {
    status 201
    headers {
      header(HttpHeaders.LOCATION, 'api/persons/1')}
  }
}
```

VALIDATING THE CONTRACT



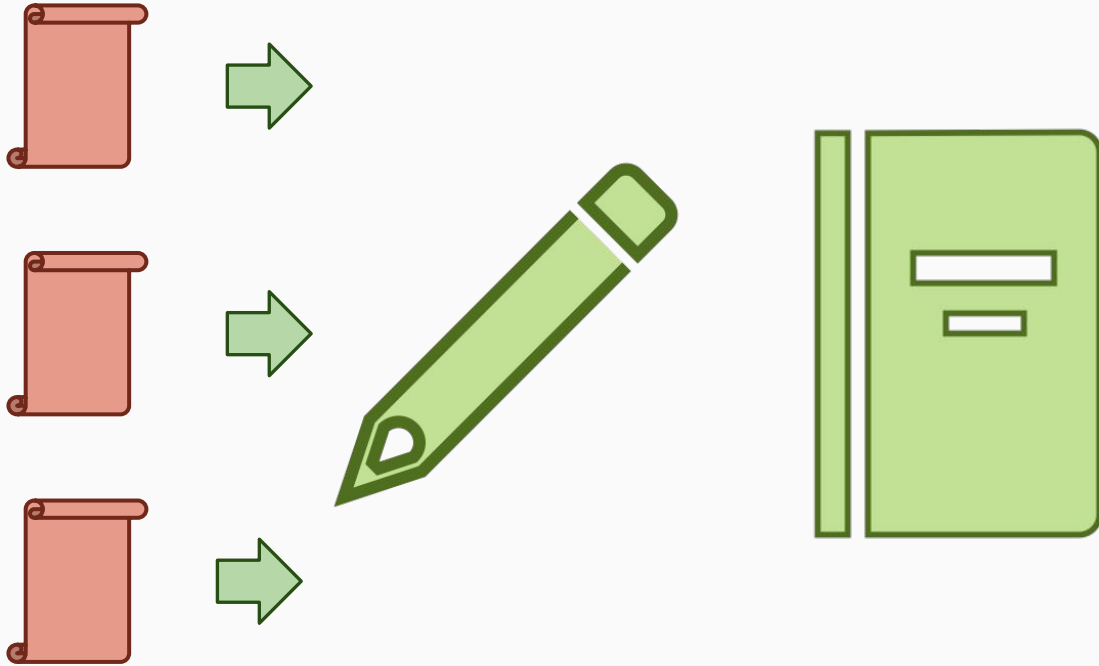
SHARING THE CONTRACT



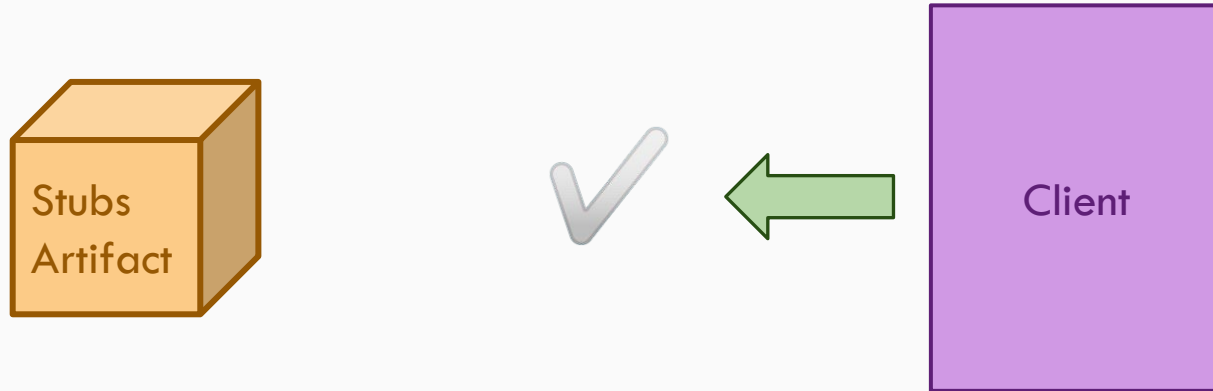
SHARING THE CONTRACT




DOCUMENTING THE API



VALIDATING AGAINST THE CONTRACT



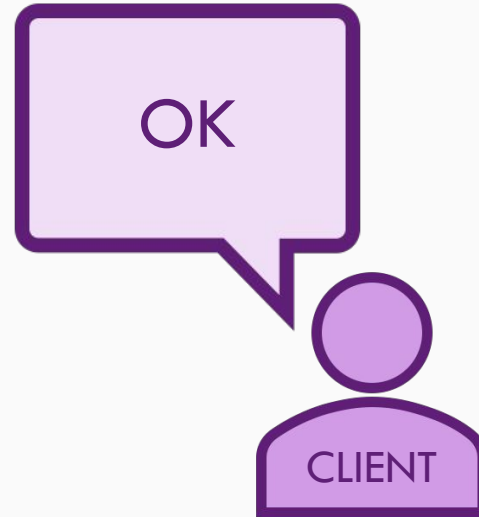
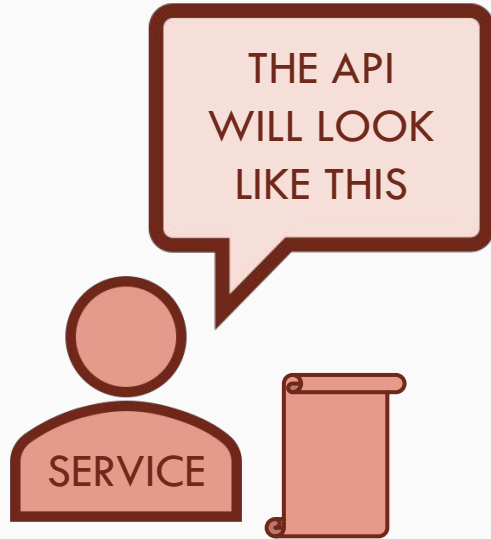
CONTRACT DRIVEN DEVELOPMENT IN REVIEW

1. Write contract(s) to define API behavior
 2. Contracts validate API matches behavior
 3. Bundle contracts as shareable artifact
 4. Generate documentation from contracts
 5. Clients can test/develop against artifact
- 
- Automated

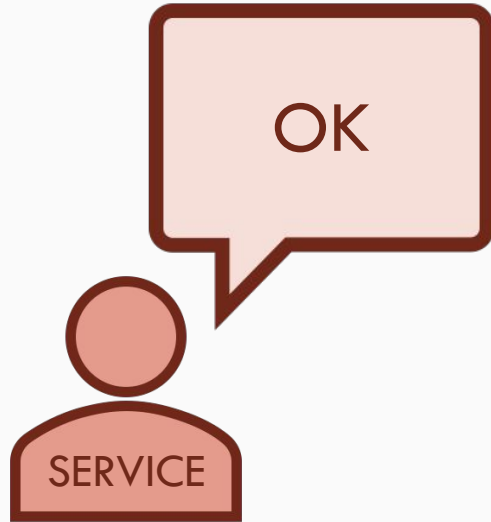


@BillyKorando

PRODUCER DRIVEN CONTRACTS



CONSUMER DRIVEN CONTRACTS



WHY COLLABORATION IS IMPORTANT

- Better utilization of developer time
- More input from parties on API design
- More buy-in from parties on API design

CAN'T SPELL "COLLABORATIVE"
WITHOUT "POLYGLOT"

LANGUAGE INDEPENDENT TOOLS

Java Developers

JDK



Maven



Java friendly IDE



Docker



Text editor



Javascript Developers

JDK



Maven



Java friendly IDE



Docker



Text editor



Other Developers

JDK



Maven



Java friendly IDE



Docker



Text editor



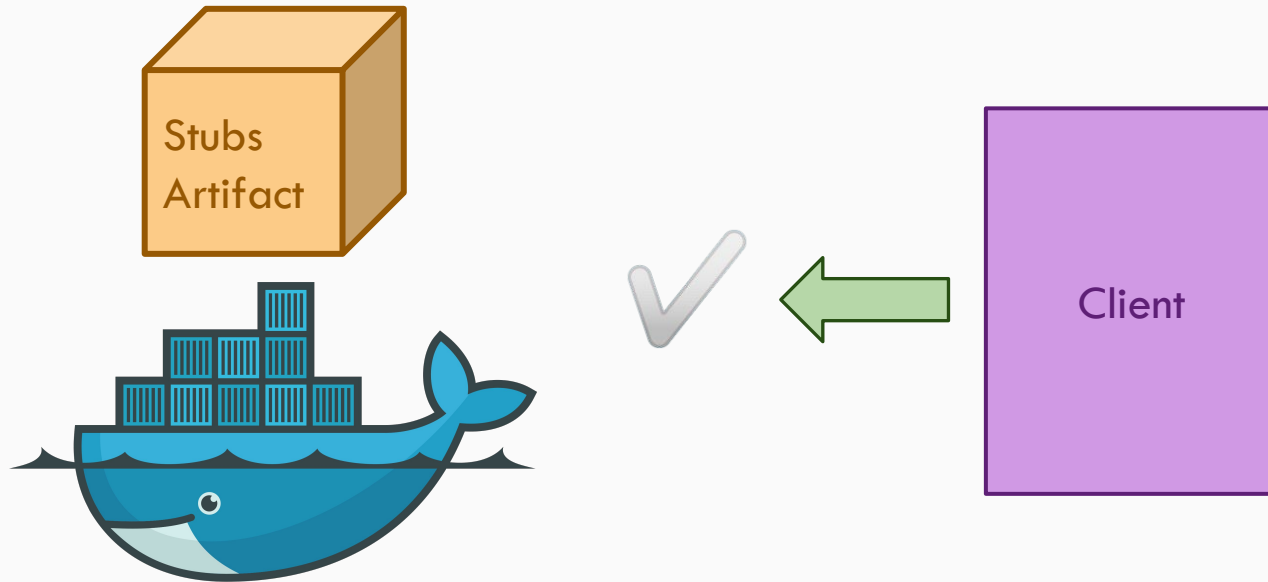
YAML CONTRACT

```
request:
  label: "Add New Person"
  method: POST
  url: /api/persons
  body:
    fName: "John"
    lName: "Doe"
  headers:
    Content-Type: application/json
response:
  status: 201
  headers:
    Location: api/persons/1
```

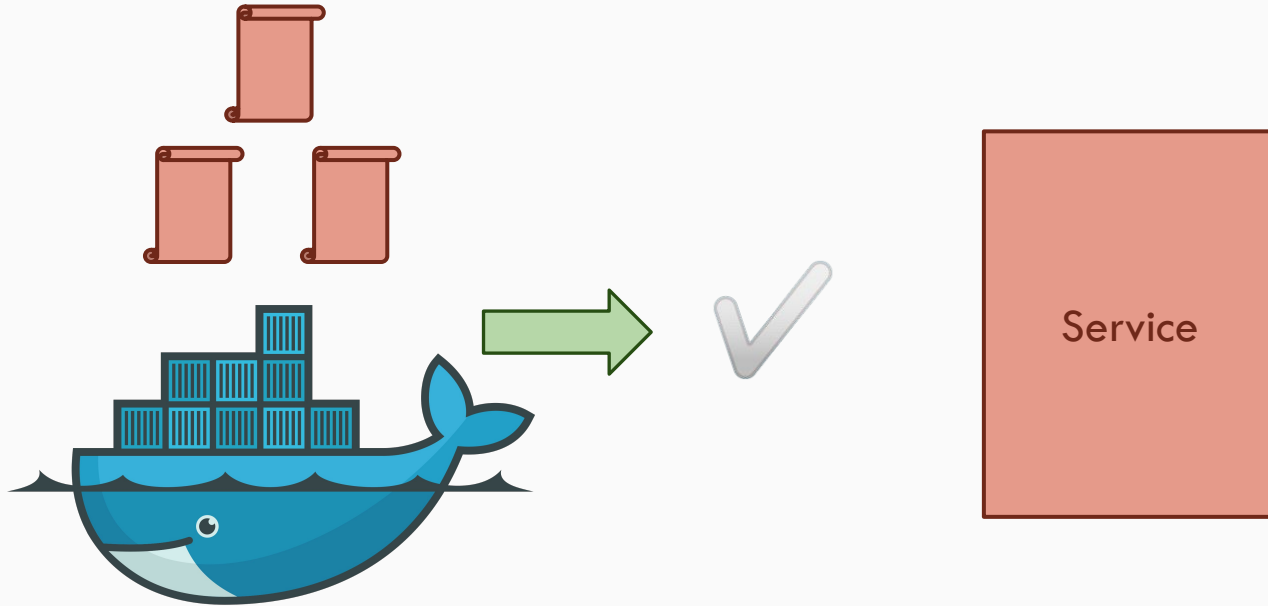
DOCKER IMAGES FOR READING CONTRACTS

- Encapsulates all Java and Spring Cloud Contract logic within a Docker container
- Images for both consumers and producers
- Configured by passing in environment variables
- Example of using producer image [link](#)
- Example of using consumer image [link](#)

CONTRACT TESTING WITH DOCKER

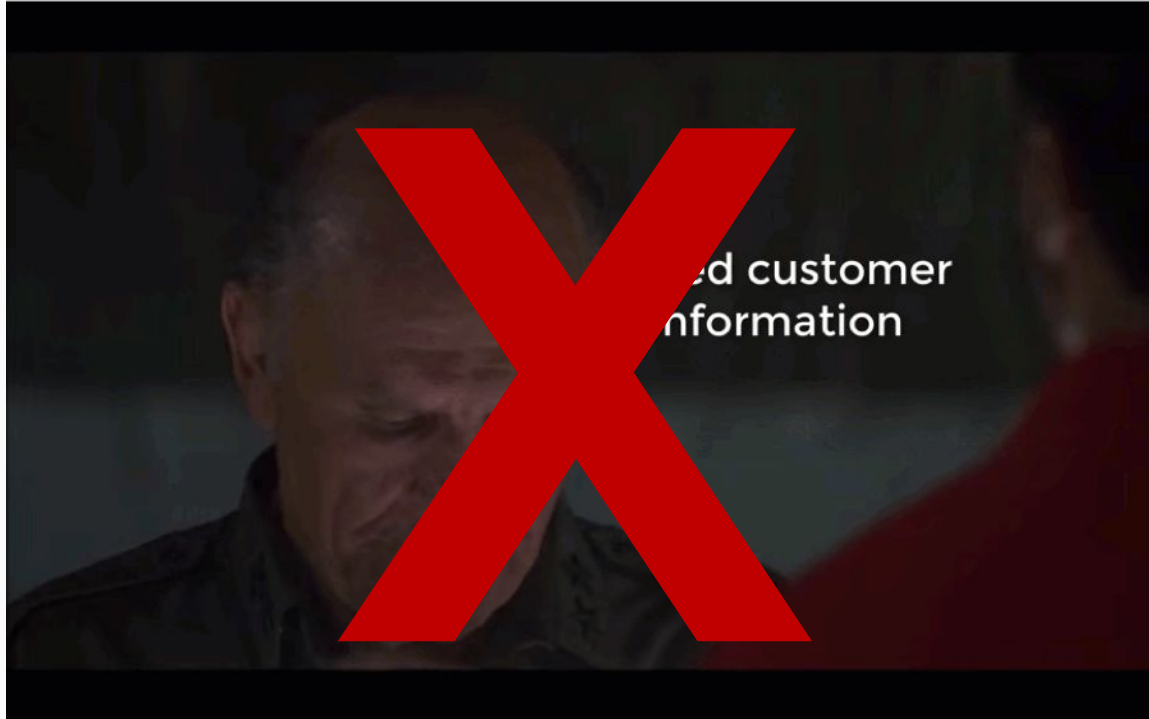


CONTRACT TESTING WITH DOCKER



DEMO TIME!

POORLY DEFINED SERVICES

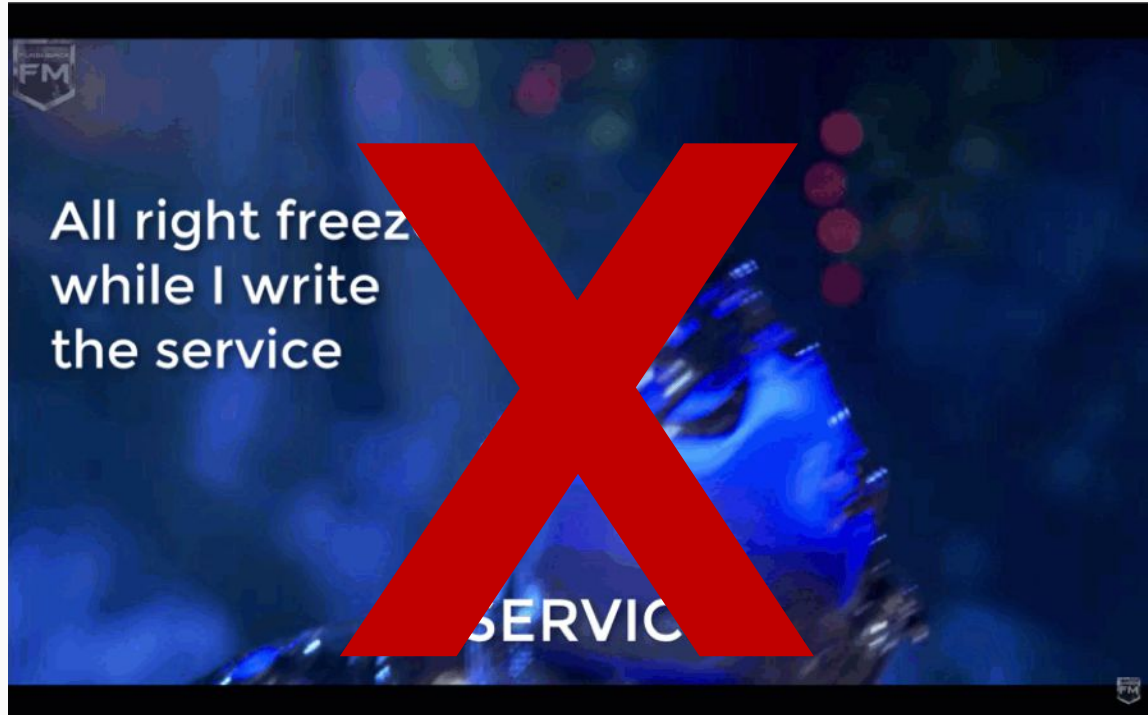


@BillyKorando

SCENE BREAKDOWN

- API is documented automatically as a result of the process
- Accurate documentation encourages discussion on design/consistency
- Clients can develop/test against contracts which give consistent results

VERY SERIAL DEVELOPMENT



@BillyKorando

SCENE BREAKDOWN

- Client and service developers can work in parallel once contract is written
- Process encourages discussion on API design

I AM ALTERING THE API



@BillyKorando

SCENE BREAKDOWN

- API is checked as part of build process
- Changes will cause either service or client to fail
- Prevents “rogue” changes from reaching PROD and becoming “finalized”

FINAL POINTS

- Not just REST/HTTP, can also write contracts for messages
- It's ok to change contracts
- Contract tests are not acceptance tests nor replace end-to-end tests
 - Not tested:
 - Resiliency
 - Timeout
 - Performance
 - Edge cases

Q & A



@BillyKorando

SOURCES

Code:

<https://github.com/wkorando/collaborative-contract-driven-development-2-0>

<https://cloud.spring.io/spring-cloud-contract/single/spring-cloud-contract.html>

<https://spring.io/blog/2018/02/13/spring-cloud-contract-in-a-polyglot-world>

Contact me:

Twitter: @BillyKorando

Email: William.Korando@ibm.com