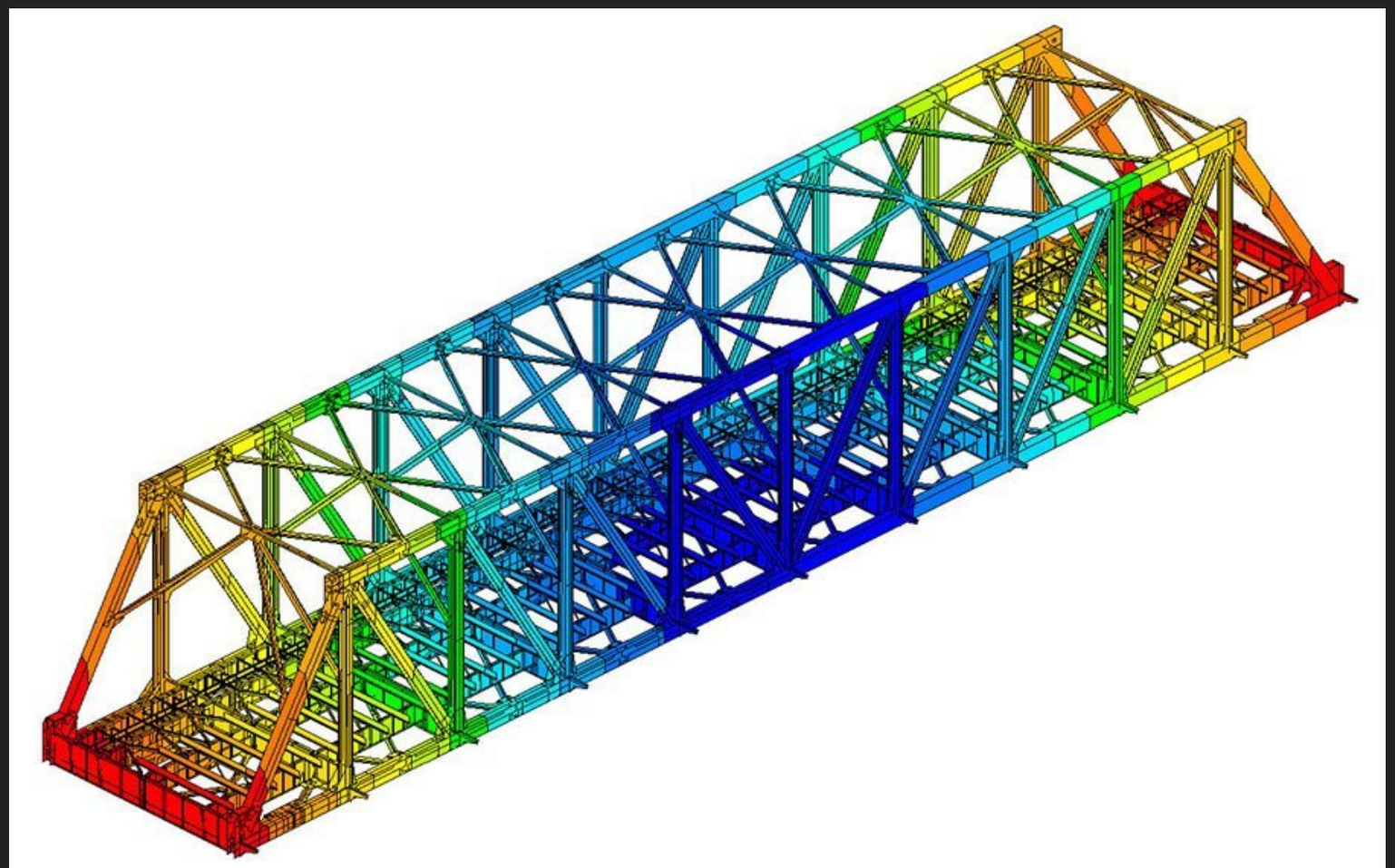


API TESTING VIA

WIREMOCK

MATT HAVILAH (mhavilah@gmail.com)

MARCH, 2019

API DEVELOPMENT + TESTING CHALLENGES

- How do you test your application when.....

API DEVELOPMENT + TESTING CHALLENGES

- How do you test your application when.....
  - you are integrating with an API/Remote Service ?

API DEVELOPMENT + TESTING CHALLENGES

- How do you test your application when.....
  - you are integrating with an API/Remote Service ?
  - and that service may be offline/unavailable

API DEVELOPMENT + TESTING CHALLENGES

- How do you test your application when.....
  - you are integrating with an API/Remote Service ?
    - and that service may be offline/unavailable
    - or slow (or costly) to test against

API DEVELOPMENT + TESTING CHALLENGES

- How do you test your application when.....
  - you are integrating with an API/Remote Service ?
    - and that service may be offline/unavailable
    - or slow (or costly) to test against
    - or may not even exist yet (ie, work in progress)

MOCKING REVIEW

- Traditional Service Mocking creates a “Test Double”

MOCKING REVIEW

- Traditional Service Mocking creates a “Test Double”
  - allows client interaction with service behaviour to be tested
  - allows behaviour with failure scenarios to be examined

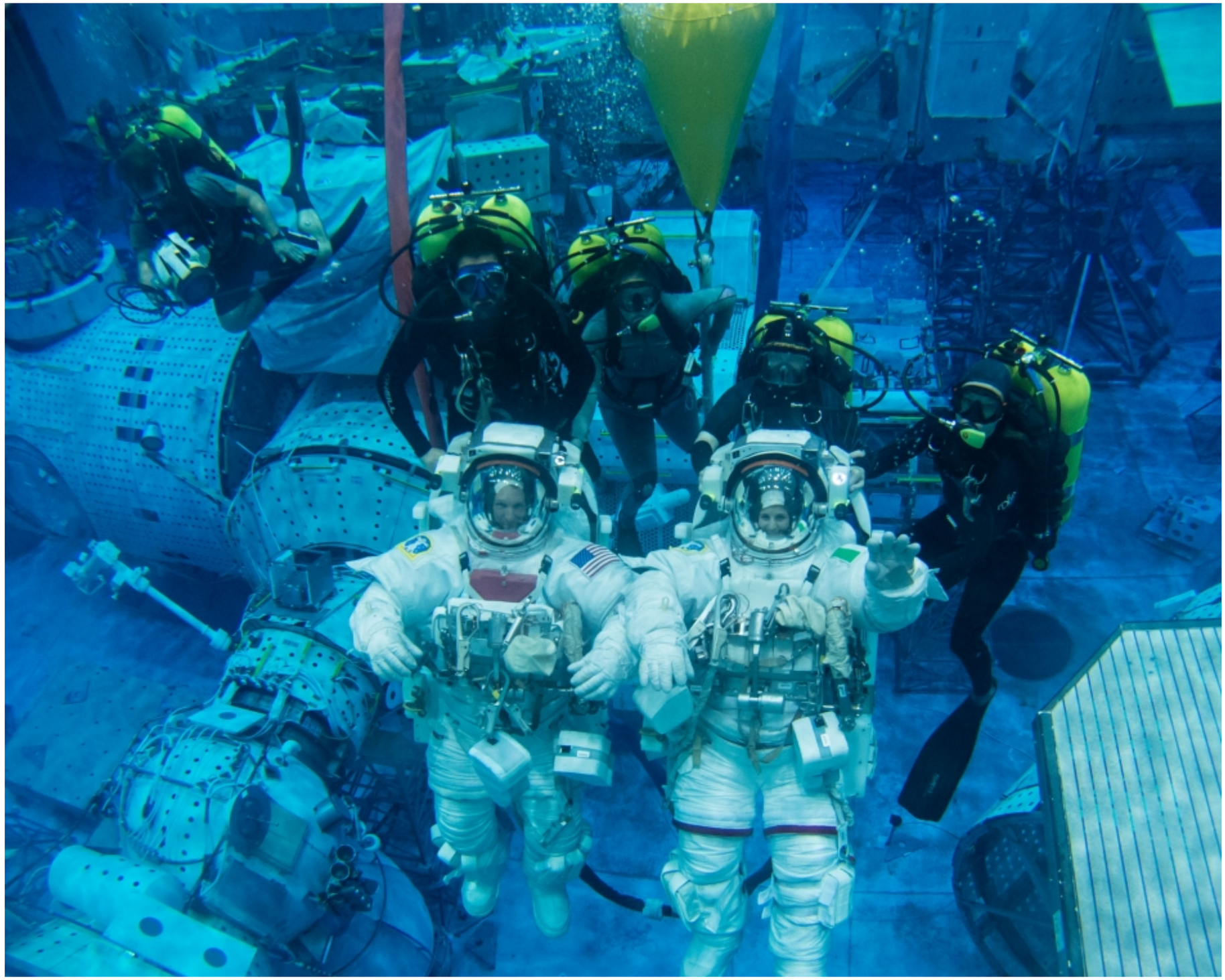
MOCKING REVIEW

- Traditional Service Mocking creates a “Test Double”
  - allows client interaction with service behaviour to be tested
  - allows behaviour with failure scenarios to be examined
  - *generally* avoids complex logic associated with “simulation”
    - statefulness, (long-lived) side-effects etc

MOCKING REVIEW

- Traditional Service Mocking creates a “Test Double”
  - allows client interaction with service behaviour to be tested
  - allows behaviour with failure scenarios to be examined
  - *generally* avoids complex logic associated with “simulation”
    - statefulness, (long-lived) side-effects etc
- Cheap and easy to generate

REMOTE API MOCKING



REMOTE API MOCKING

- We can take the same approach with *remote* services

REMOTE API MOCKING

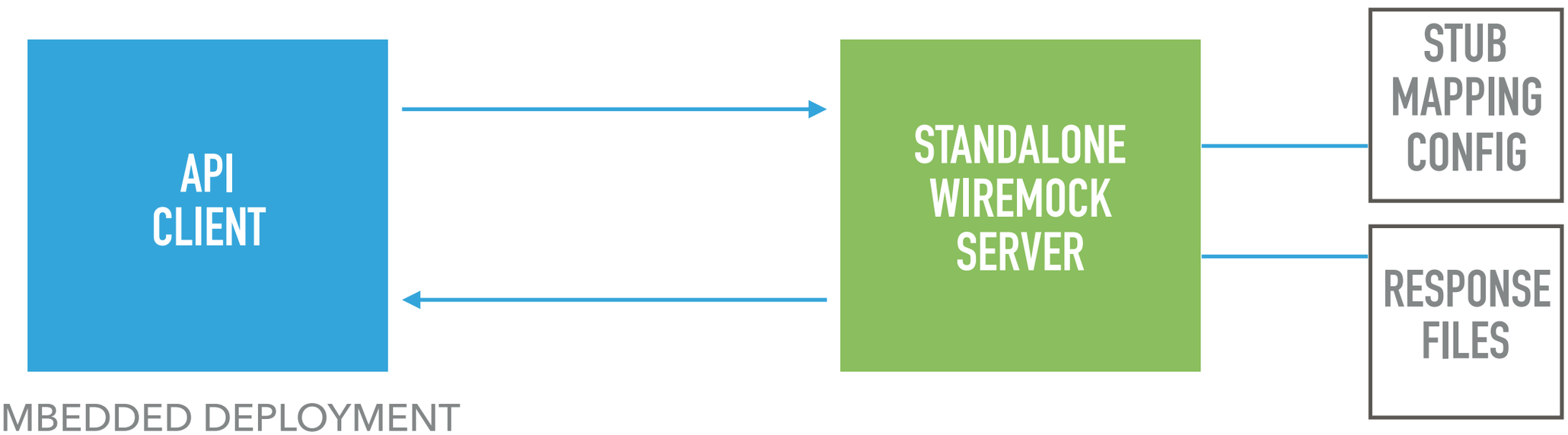
- We can take the same approach with *remote* services
- Several approaches:
  - Write *stubbed* responses to API requests

REMOTE API MOCKING

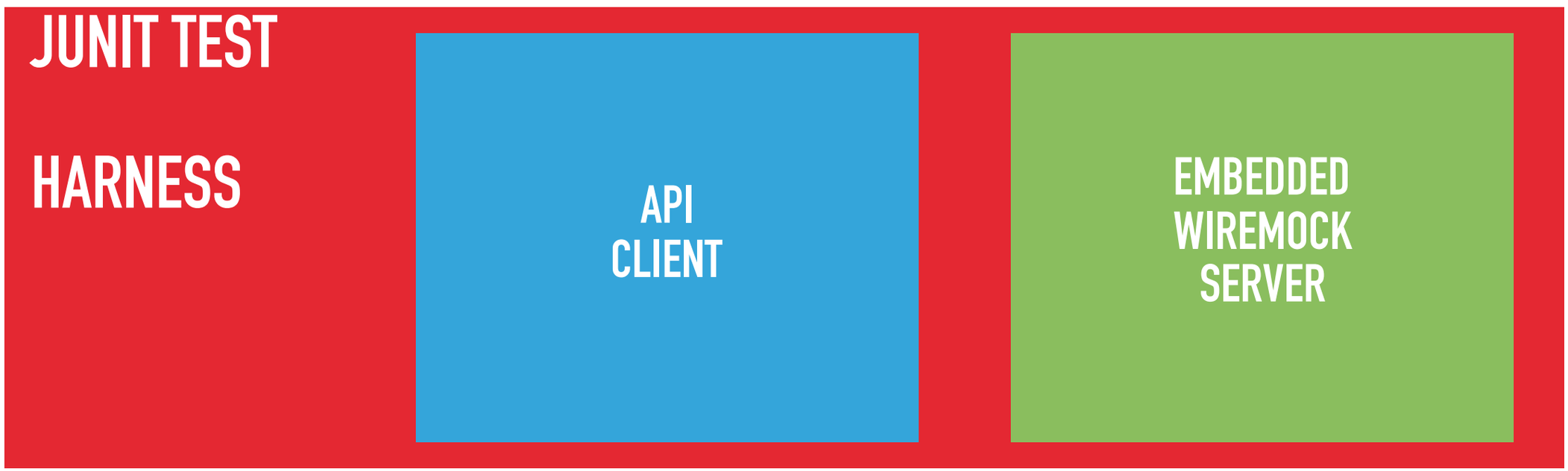
- We can take the same approach with *remote* services
- Several approaches:
  - Write *stubbed* responses to API requests
- or,
  - Use a *proxy to* Record/Playback requests to a live API

API STUBBING

STANDALONE DEPLOYMENT



EMBEDDED DEPLOYMENT



API STUBBING

- Use a set of "request matcher" rules to define an API
  - match by request URL (or URL pattern)
  - HTTP request method
  - HTTP request headers, cookies etc
  - Request body/query parameters

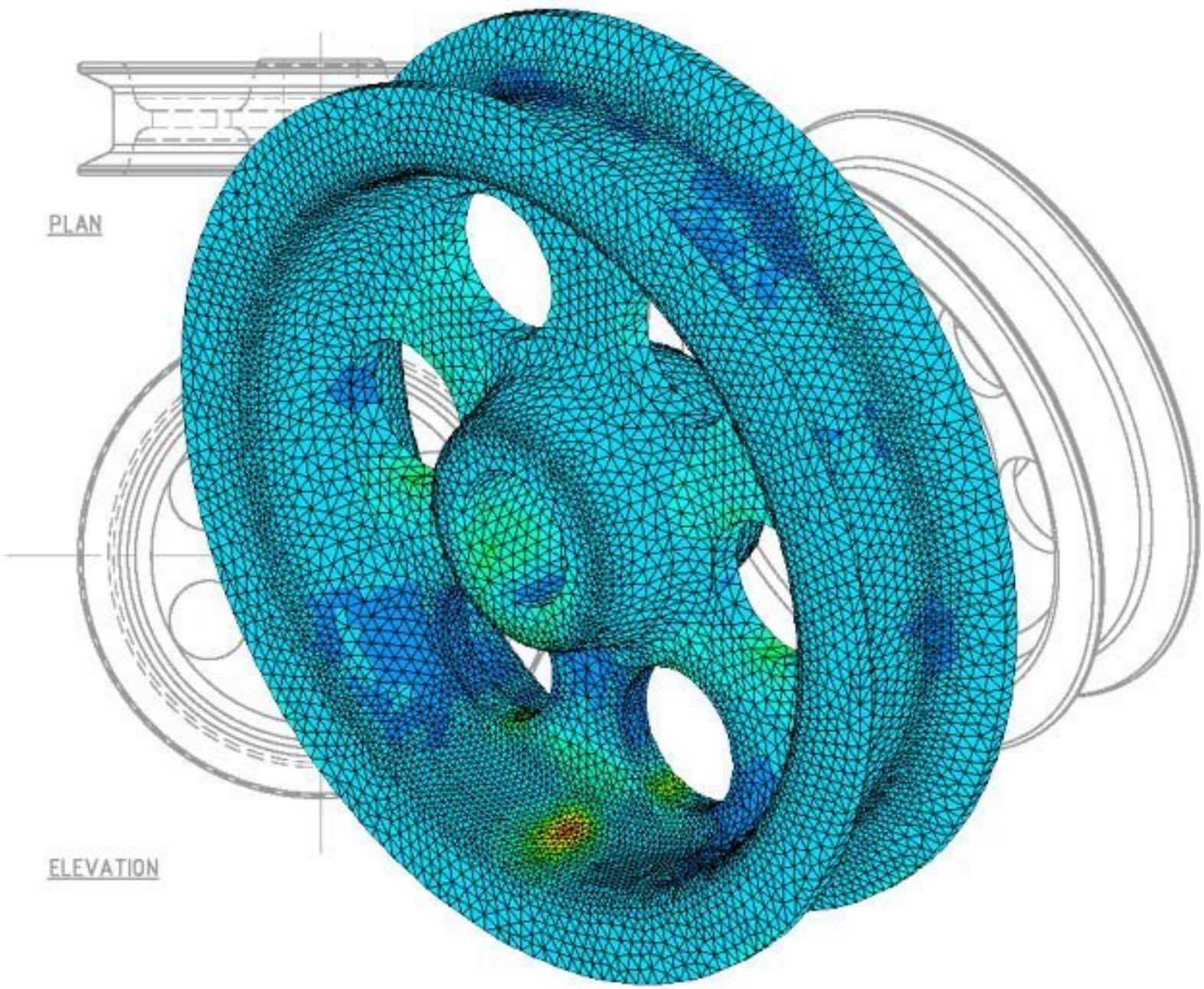
API STUBBING – EXAMPLE

- Cafe Ordering Application
  - As a caffeine fan I want to ask what styles of coffee are available so that I can make an order
    - GET /products
- As a caffeine fan I want to order a coffee so that I can feel warm and fuzzy
  - POST /order ==> order ID

API STUBBING – EXAMPLE

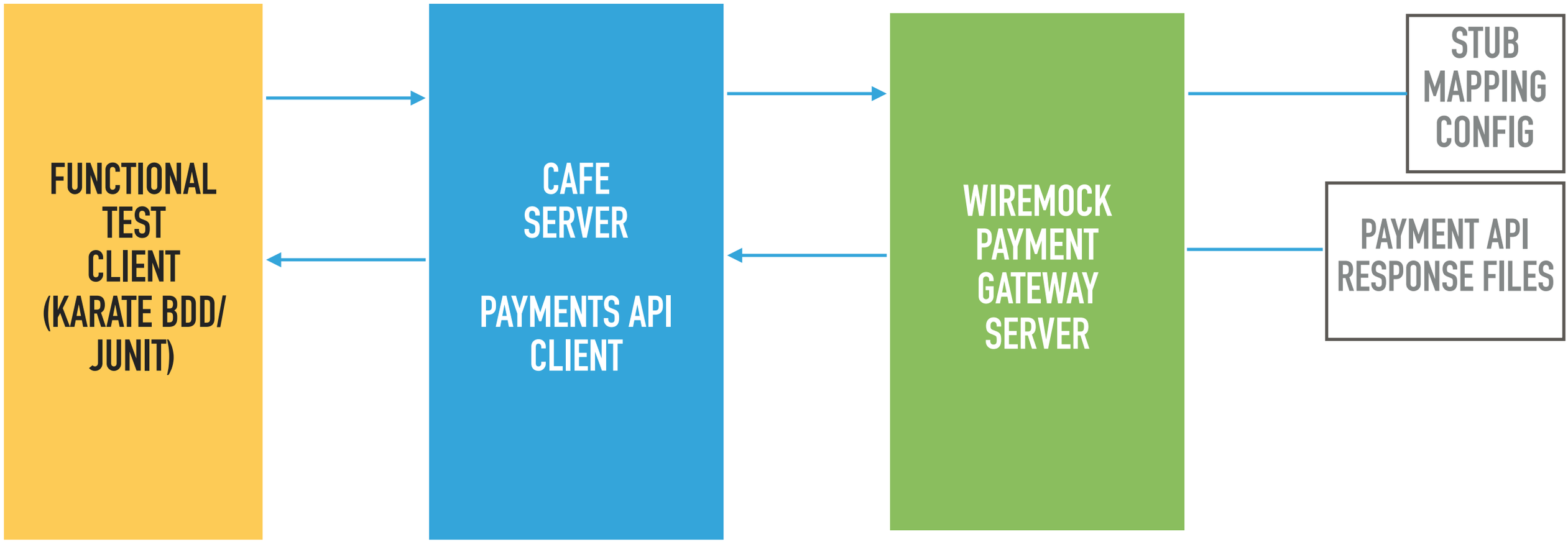
- Cafe Ordering Application con'd
  - As a caffeine fan I want check the status of my order
    - GET /order/<orderID>/status
  - As a caffeine fan I want to amend my order so that I can get some tasty treats
    - PATCH /order/<orderID>
  - As a caffeine fan I want to cancel my order so that I can catch my early running train
    - DELETE /order/<orderID>

API STUBBING – DEMO

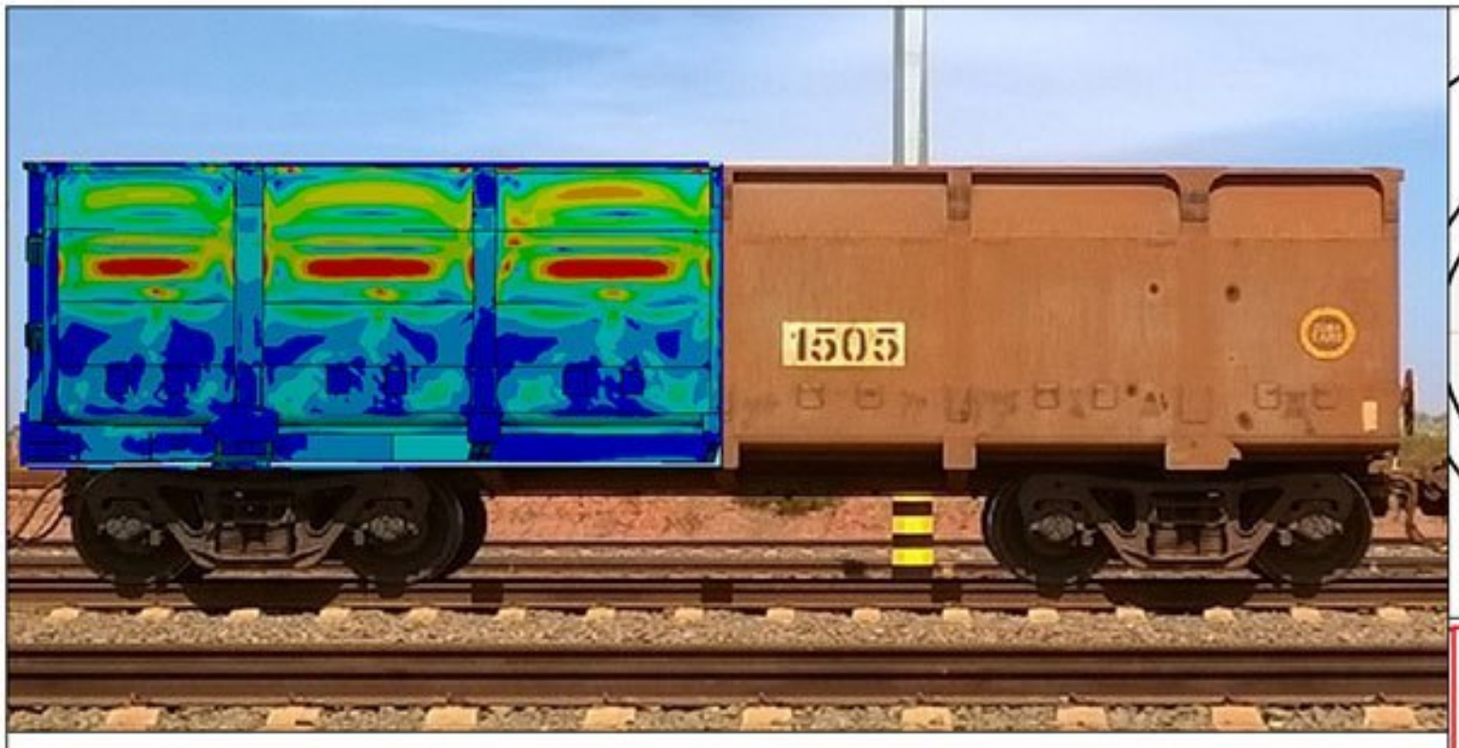


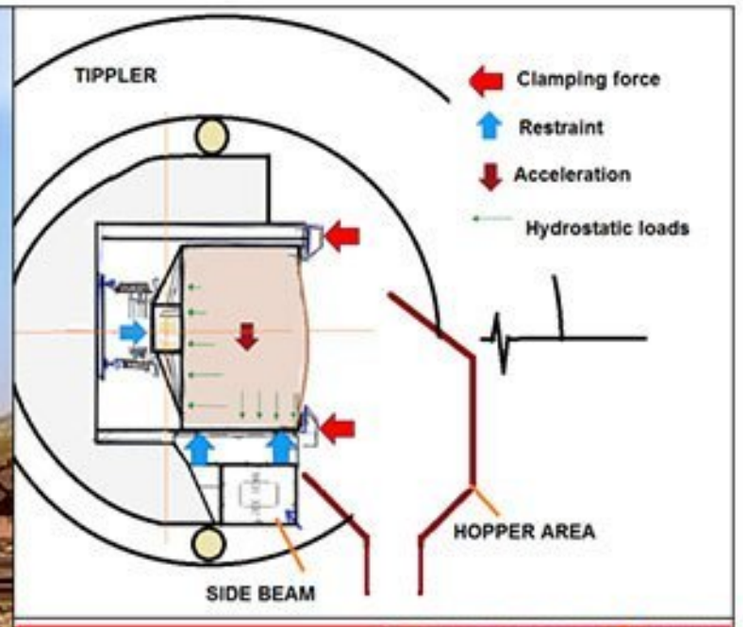
CAFE APPLICATION TEST USAGE

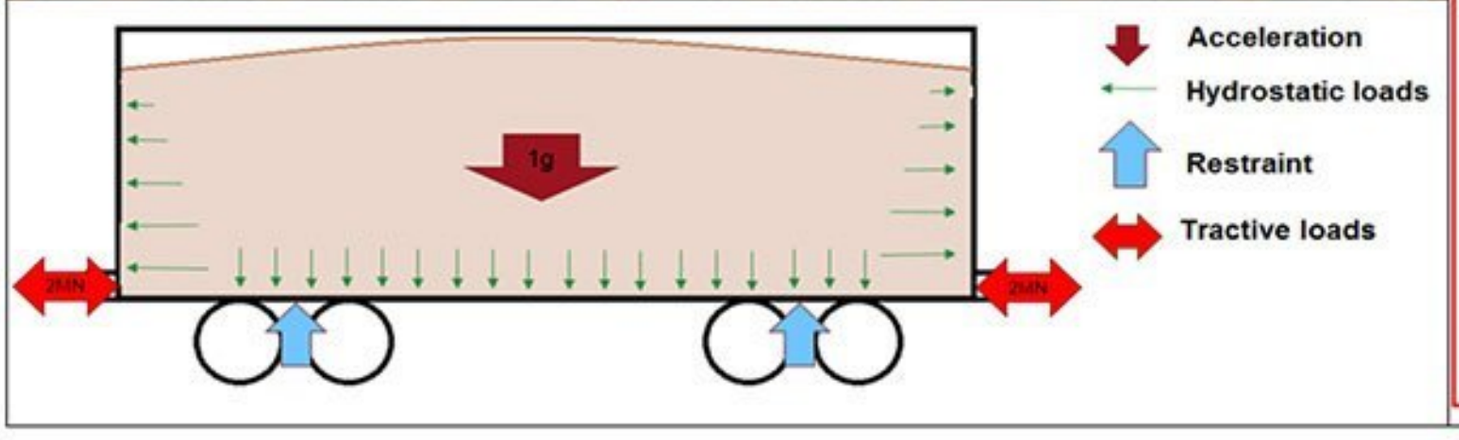
- FUNCTIONAL TESTING/STANDALONE DEPLOYMENT

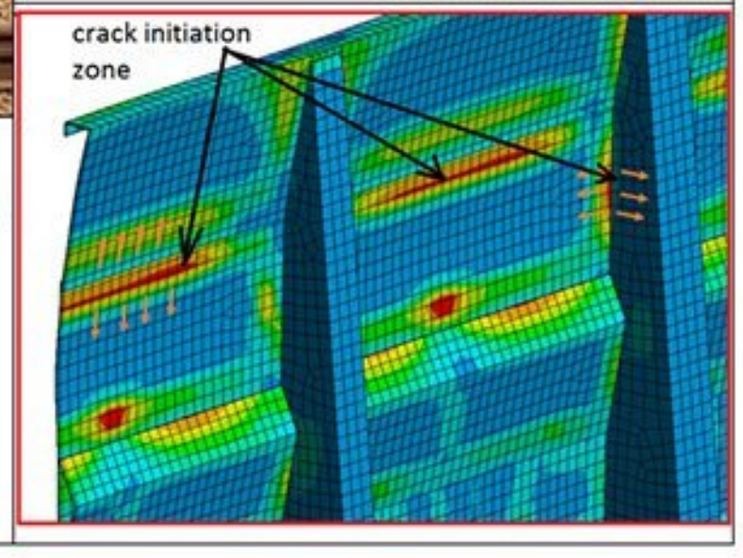


PROXYING


1505

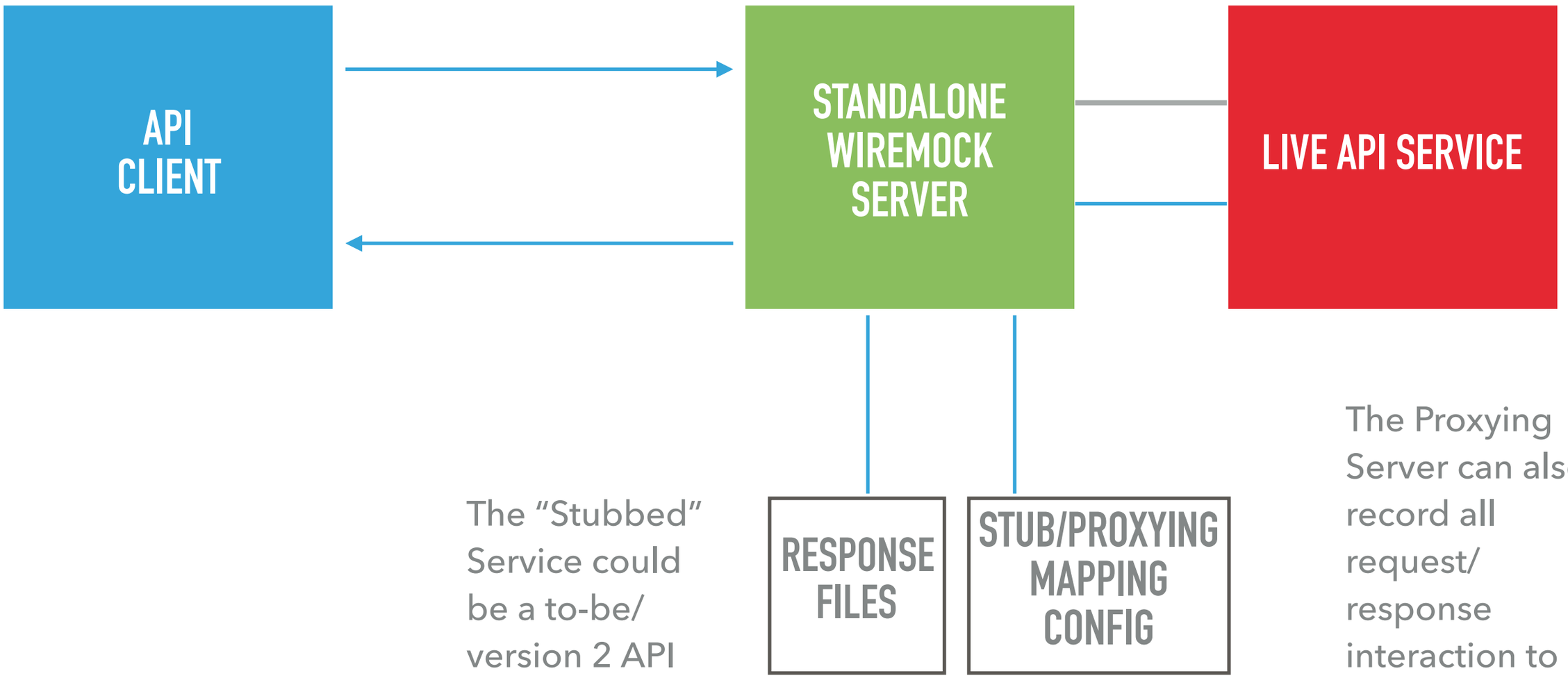

TIPPLER
Clamping force
Restraint
Acceleration
Hydrostatic loads
HOPPER AREA
SIDE BEAM


1g
2MN
2MN
Acceleration
Hydrostatic loads
Restraint
Tractive loads


crack initiation
zone

PROXYING

STANDALONE, PROXYING DEPLOYMENT



The "Live" Service could be an as-is/ version 1 API

The "Stubbed" Service could be a to-be/ version 2 API

The Proxying Server can also record all request/ response interaction to "build" the mappings

PROXYING/RECORDING – EXAMPLE

- To set up Wiremock to proxy and record requests to: http://api.service.com

```
$ java -jar lib/wiremock-standalone-2.19.0.jar --port 9000 --proxy-all="http://api.service.com" --record-mappings

Note: The following script is available in the github repo listed at the end of the presentation
$ ./startProxy.sh
Starting Wiremock proxy.
Listening on port: 9000
Proxying API Service: https://api.willyweather.com.au/v2/
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
 /$$      /$$ /$$                     /$$      /$$                     /$$
| $$  /$ | $$|__/                    | $$$    /$$$                    | $$
| $$ /$$$| $$ /$$  /$$$$$$   /$$$$$$ | $$$$  /$$$$  /$$$$$$   /$$$$$$$| $$   /$$
| $$/$$ $$ $$| $$ /$$__  $$ /$$__  $$| $$ $$/$$ $$ /$$__  $$ /$$_____/| $$  /$$/
| $$$$_  $$$$| $$| $$  \__/| $$$$$$$$| $$  $$$| $$| $$  \ $$| $$      | $$$$$$/
| $$$/ \  $$$| $$| $$      | $$_____/| $$\  $ | $$| $$  | $$| $$      | $$_  $$
| $$/   \  $$| $$| $$      |  $$$$$$$| $$ \/  | $$|  $$$$$$/|  $$$$$$$| $$ \  $$
|__/     \__/|__/|__/       \_______/|__/     |__/ \______/  \_______/|__/  \__/

port:                           9000
proxy-all:                      https://api.willyweather.com.au/v2/
preserve-host-header:           false
enable-browser-proxying:        false
disable-banner:                 false
record-mappings:                true
match-headers:                  []
no-request-journal:             false
verbose:                        false

Requests to: http://localhost:9000 will be forwarded to: https://api.willyweather.com.au/v2/
```

PROXYING/RECORDING – EXAMPLE

- All requests sent to the Wiremock proxy (on localhost) will be recorded as "JSON mappings":

```
$ curl -X GET -o - "http://localhost:9000/locations/2251/weather.json?
forecasts=rainfallprobability&days=5&startDate=2017-03-27"
{"location":
{"id":2251,"name":"Queenscliff","region":"Sydney","state":"NSW","postcode":"2096","timeZone":"Australia\/
Sydney","lat":-33.7846,"lng":151.28782,"typeId":1},"forecasts":{"rainfallprobability":{"days":[],"units":
{"percentage":"%"},"issueDateTime":"2019-04-04 16:37:37","carousel":{"size":8,"start":-736}}}}

$ ls mappings
mapping-2251-weather.json-bELVn.json

$ more mappings/mapping-2251-weather.json-bELVn.json
{
  "id" : "0cf105e3-bd4b-3ad3-a7f9-17a39839a48c",
  "request" : {
    "url" : "/locations/2251/weather.json?forecasts=rainfallprobability&days=5&startDate=2017-03-27",
    "method" : "GET"
  },
  "response" : {
    "status" : 200,
    "bodyFileName" : "body-2251-weather.json-bELVn.json",
    "headers" : {
      "Date" : "Thu, 04 Apr 2019 05:46:29 GMT",
      "Content-Type" : "application/json",
      ……
      "Vary" : "Accept-Encoding,User-Agent"
    }
  },
  "uuid" : "0cf105e3-bd4b-3ad3-a7f9-17a39839a48c"
}
```

Recorded Request URL path

Captured Response Content File

PROXYING/RECORDING – CON'D

- API Service response content will be saved in the Wiremock "__files" sub-folder:

```
$ ls __files
body-2251-weather.json-bELVn.json

$ more __files/body-2251-weather.json-bELVn.json  | jq .
{
  "location": {
    "id": 2251,
    "name": "Queenscliff",
    "region": "Sydney",
    "state": "NSW",
    "postcode": "2096",
    "timeZone": "Australia/Sydney",
    "lat": -33.7846,
    "lng": 151.28782,
    "typeId": 1
  },
  "forecasts": {
     "rainfallprobability": {
      "days": [
        {
          "dateTime": "2019-04-04 00:00:00",
          "entries": [
            {
              "dateTime": "2019-04-04 02:00:00",
              "probability": 5
            },
            {
              "dateTime": "2019-04-04 05:00:00",
              "probability": 10
            },
.....
```

REFERENCES

- Wiremock

http://wiremock.org/
http://wiremock.org/docs/
http://wiremock.org/docs/stubbing/
http://wiremock.org/docs/proxying/
http://wiremock.org/docs/record-playback/
https://github.com/tomakehurst/wiremock

  - Extensions

https://handlebarsjs.com/block_helpers.html
https://github.com/opentable/wiremock-body-transformer

https://github.com/jknack/handlebars.java/blob/master/handlebars/src/main/java/com/github/jknack/handlebars/helper/StringHelpers.java
https://github.com/tomakehurst/wiremock/tree/master/src/test/java/com/github/tomakehurst/wiremock/extension/responsetemplating **

** Contains lots of good usage examples of the XML, JSON and "Handlebars"-based response template helpers

- This Presentation's Source Code

https://github.com/mhavilah/CafeService

REFERENCES

- Tutorials (Java API)

https://www.petrikainulainen.net/wiremock-tutorial/
https://www.petrikainulainen.net/programming/testing/wiremock-tutorial-introduction/
https://www.petrikainulainen.net/programming/testing/wiremock-tutorial-introduction-to-stubbing/

* Request URL/Method Matching
https://www.petrikainulainen.net/programming/testing/wiremock-tutorial-request-matching-part-one/

* Request Parameter/Cookie Matching
https://www.petrikainulainen.net/programming/testing/wiremock-tutorial-request-matching-part-two/

* JSON Request Matching
https://www.petrikainulainen.net/programming/testing/wiremock-tutorial-request-matching-part-three/

* XML Request Matching
https://www.petrikainulainen.net/programming/testing/wiremock-tutorial-request-matching-part-four/

- Proxying Mode

https://dzone.com/articles/mocking-rest-api-with-wiremock-using-recording-mod

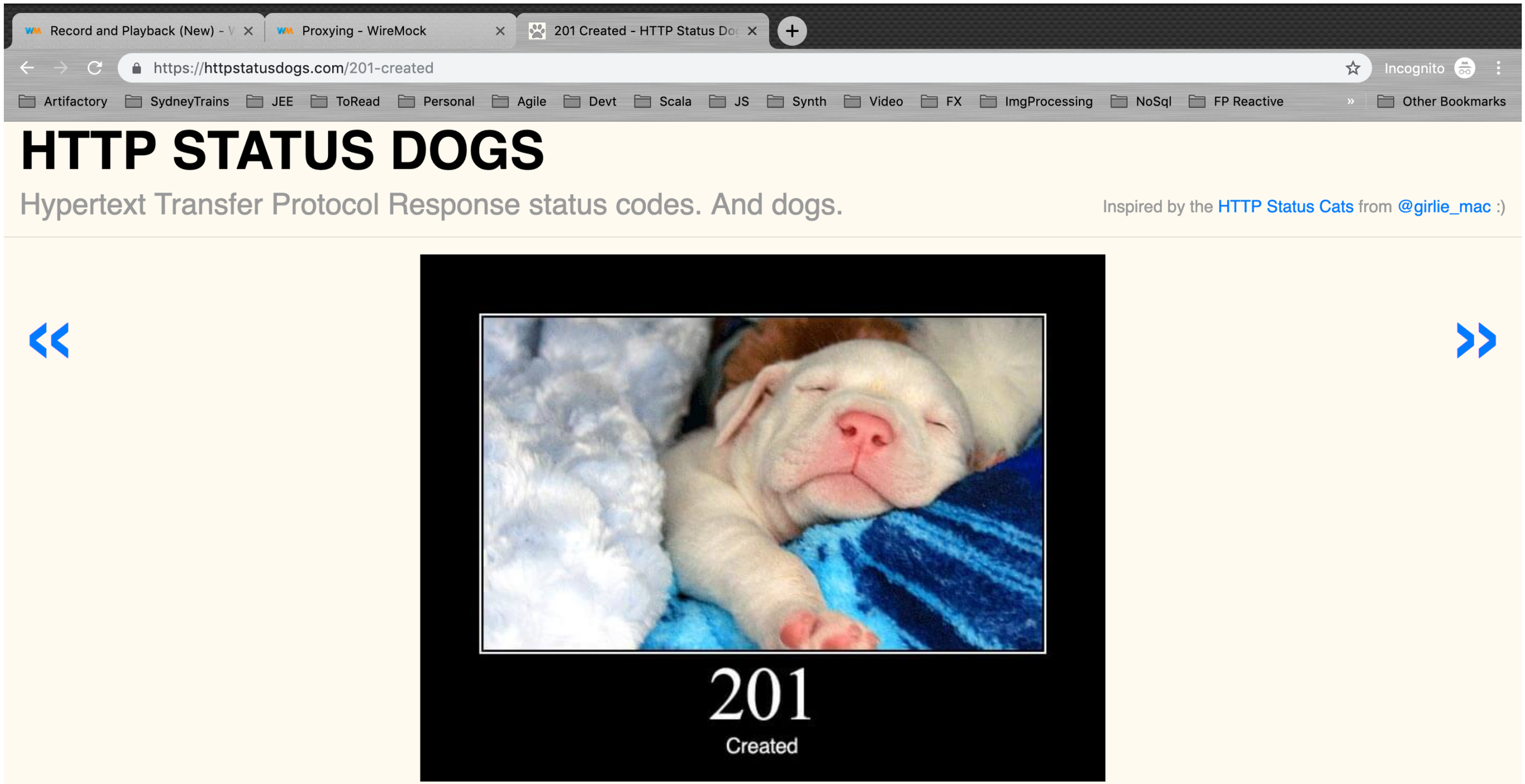
- Related Projects

https://github.com/telegraph/wiremock-swagger

- Clip Art

https://www.nasa.gov/mission_pages/station/multimedia/gallery/jsc2012e238476.html#.XJoH5-szbOQ

THANKS


Record and Playback (New) -
Proxying - WireMock
201 Created - HTTP Status Do
https://httpstatusdogs.com/201-created
Incognito
Artifactory
SydneyTrains
JEE
ToRead
Personal
Agile
Devt
Scala
JS
Synth
Video
FX
ImgProcessing
NoSql
FP Reactive
Other Bookmarks
HTTP STATUS DOGS
Hypertext Transfer Protocol Response status codes. And dogs.
Inspired by the HTTP Status Cats from @girlie_mac :)
201
Created
201 Created: The request has been fulfilled and resulted in a new resource being created.
Photo by Beverly & Pack.