AARHUS UNIVERSITET

Microservices and DevOps

Scalable Microservices

Test Double Services

Henrik Bærbak Christensen

AARHUS UNIVERSITET

Motivation

- Integration testing again
  - … and service tests, and CDTs
  - The need for Meszaro's **test doubles**

- Easiest case
  - Use programmatic test doubles or mock libraries (in-process)
    - Requirement: *Control of the dependency injection*
      - You have to tell your service to *use another implementation of a service interface*
        - » *Program to an interface, use object composition, dep. injection*

AARHUS UNIVERSITET

In Architectures

- Meszaros (2007) has a special category of doubles
  - *Saboteurs* *stubs that behave badly!*
- The ones we need for testing Nygard **Stability Patterns.**
- Decorator pattern can be used, to *add saboteur* behavior

```
17 public class SlowResponseCaveStorage implements CaveStorage {
18
19   private CaveStorage decoratee;
20
21   public SlowResponseCaveStorage() {
22     decoratee = new FakeCaveStorage();
23   }
24
25   public RoomRecord getRoom(String positionString) {
26     boolean dummy =
27         JOptionPane.YES_OPTION
28         ==
29         JOptionPane.showConfirmDialog(null,
30                                       "Is 'getRoom()' slow?",
31                                       "Slow Response Anti pattern",
32                                       JOptionPane.YES_NO_OPTION );
33
34     return decoratee.getRoom(positionString);
35   }
```

Motivation

Bad Architecture, IMO

- But it is not always possible
  - Not programmed to an interface, no dependency injection
  - Low cohesion design
    - Service is accessed in 7.463 places all over the code…
  - Non “programmatic” interfaces (REST, UDP, …)

- In a microservice/SOA context, services are remotely deployed services, and we need to test that as well
  - CDT and connector testing within a staging environment that is a real distributed system, but with saboteur doubles…
    - Can test latency issues, availability, etc.

AARHUS UNIVERSITET

Solution

- Make test doubles that are *deployed services*
  - *Support 'out-of-process service tests'*
- That is
  - Conceptually identical
  - Difference: The interface
    - In-process service test doubles:
      - Java class implementing interface
      - Or Mock library generated class, with expectations set
    - Out-of-process service test doubles:
      - Deployed remote service on given port, programmed to respond to network calls, using 'canned' responses

How?

- Well, of course you may just code it!

- Make a (SparkJava) REST service that produce
  - canned responses
  - is slow responding
  - fail, drop network connection, send illegal JSON, send 1TB random byte array, return 1.000.000 item JSON array, …

- Alternative: A programmable REST service…

AARHUS UNIVERSITET

Mountebank

A Nice Technology Choice

AARHUS UNIVERSITET

Demo

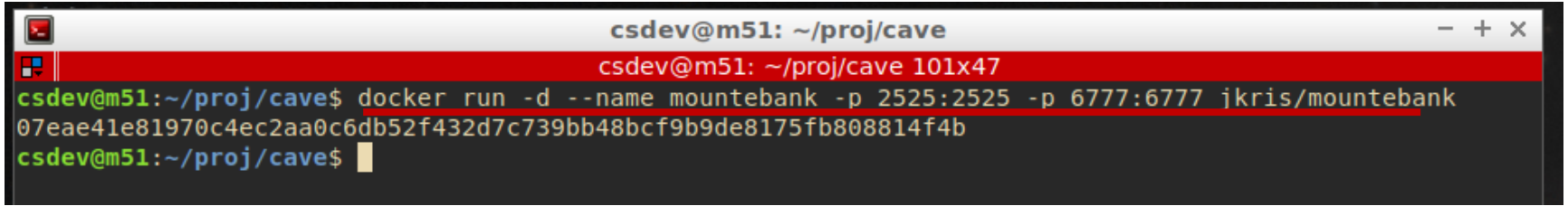
- Goal:
  - Issue: The QuoteService returns random quotes
    - Test before we have developed this *tricky service* ☺
  - Goal: Make a deterministic test stub

```
csdev@m51:~/proj/cave$ http localhost:6777/msdo/v1/quotes/7
HTTP/1.1 200 OK
Connection: close
Content-Type: application/json
Date: Wed, 01 Apr 2020 13:17:59 GMT
Transfer-Encoding: chunked

{
    "author": "Albert Einstein",
    "number": 7,
    "quote": "The true sign of intelligence is not knowledge but imagination."
}

csdev@m51:~/proj/cave$
```

AARHUS UNIVERSITET

Installing? Nay

- Mountebank is a NodeJS service ☹
- Docker, help me…
  - 2525 is Mountebank's port
  - Must port map the *imposter port* as well

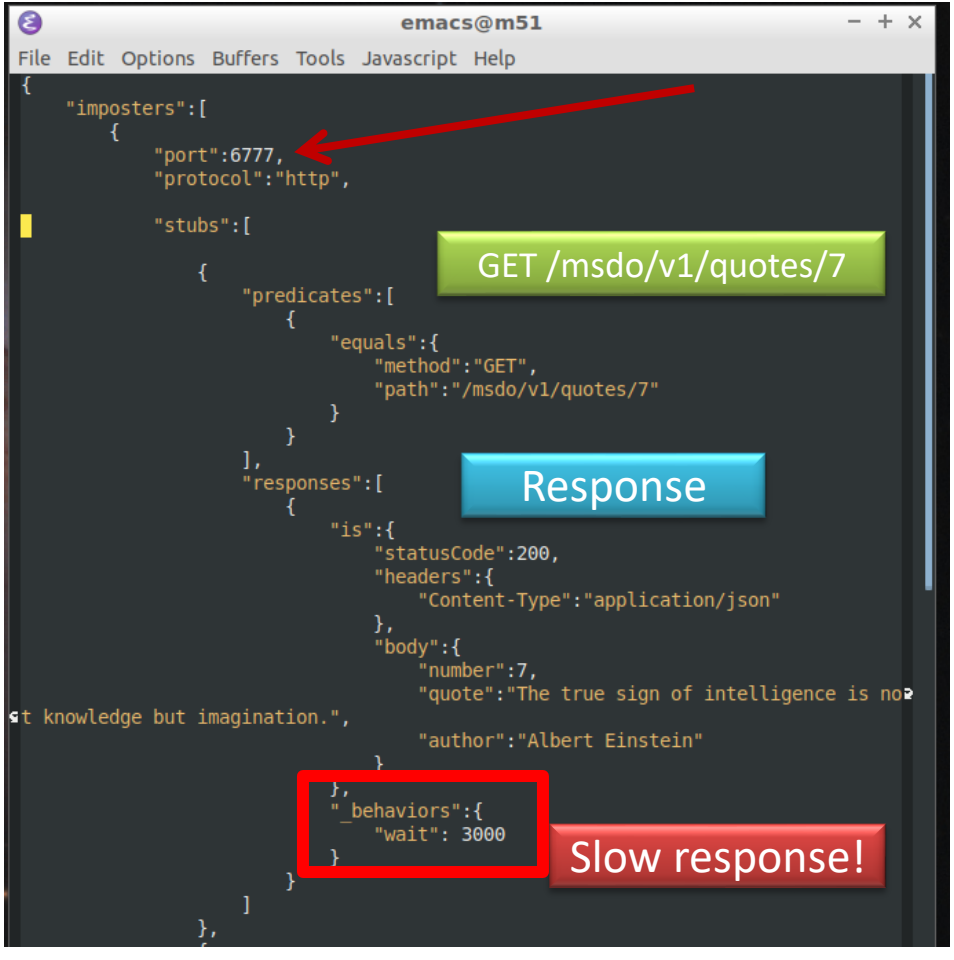


Using one of several mountebank images…

AARHUS UNIVERSITET

Define Behavior

- Next we have to *program* the imposter…
  - Basically by POSTing a JSON configuration files to Mountebank



AARHUS UNIVERSITET

Step 2

- Here I use ‘httpie’ to PUT the JSON to mountebank

```
csdev@m51:~/proj/cave$ http PUT localhost:2525/imposters < integration/src/integration/resources/moun
tebank/quote-service.json
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Length: 222
Content-Type: application/json; charset=utf-8
Date: Wed, 01 Apr 2020 13:16:49 GMT

{
    "imposters": [
        {
            "_links": {
                "self": {
                    "href": "http://localhost:2525/imposters/6777"
                }
            },
            "numberOfRequests": 0,
            "port": 6777,
            "protocol": "http"
        }
    ]
}
```

AARHUS UNIVERSITET

Ok - Testing

```
csdev@m51:~/proj/cave$ http localhost:6777/msdo/v1/quotes/7
HTTP/1.1 200 OK
Connection: close
Content-Type: application/json
Date: Wed, 01 Apr 2020 13:17:59 GMT
Transfer-Encoding: chunked

{
    "author": "Albert Einstein",
    "number": 7,
    "quote": "The true sign of intelligence is not knowledge but imagination."
}

csdev@m51:~/proj/cave$
```

Long wait here!

Now: A *slow responding deterministic test stub service.*

AARHUS UNIVERSITET

Testing

```
csdev@m51:~/proj/cave$ ./gradlew daemon -Pcpf=quote-service-local.cpf
                                csdev@m51: ~/proj/cave 101x23
csdev@m51:~/proj/cave$ ./gradlew cmd
```

```
== Welcome to SkyCave, player Mikkel ==
Entering command loop, type "q" to quit, "h" for help.
> quote 7
The true sign of intelligence is not knowledge but imagination. - Albert Einstein

> quote 3
*The requested quote was not found*

> quote 8
*The requested quote was not found*

>
```

AARHUS UNIVERSITET

Advanced Features

- Lots
  - Mock verifications
  - Variable behavior
    - Add more responses, taken in order
  - And on and on

```
POST /imposters HTTP/1.1
Host: localhost:30465
Accept: application/json
Content-Type: application/json

{
  "port": 7777,
  "protocol": "http",
  "stubs": [
    {
      "responses": [
        {
          "is": {
            "body": "This will repeat 2 times"
          },
          "_behaviors": {
            "repeat": 2
          }
        },
        {
          "is": {
            "body": "Then this will return"
          }
        }
      ]
    }
  ]
}
```

- Take care:
  - Doubles should be simple, or the bug will be in the double ☺
    - I spend quite some time debugging my JSON to find that ‘,’ I had misplaced ☹

AARHUS UNIVERSITET

TestContainers Context

- I had a *hell of a troubled time* to get mountebank running in a TestContainers context ☹
  - Issue:
    - TC will await the port 6777 port opening for the doubled service
    - Which will **not** open until it programmed
      - Deadlock: waiting for an event that cannot happen ☹
  - Solution:
    - Do not use PUT/POST for programming, instead volume mount a ‘config.json’ file with the double behavior specification, and tell mountebank to read that.

AARHUS UNIVERSITET

Conclusion

- We can replace programmatic test doubles with service test doubles when
  - The UUT has no well encapsulated API to the service and/or no dependency injection in place
  - We want to make integration testing in a real distributed staging environment
- Mountebank is a really nice and flexible tool
- Use it in the mandatory on *safe failure modes…*