Simulating APIs for Effective Testing: (Micro)Service Virtualisation

Andrew Morgan
@mogronalol

OpenCredo & Spectolabs
About Me

- Consultant at OpenCredo
- Background mainly with Java and now Go
- Spectolabs

Andrew Morgan
@mogronalol
Agenda

• The problems with operating and testing distributed systems and microservices
• The problems with operating and testing distributed systems and microservices
• Overview of Service Virtualisation / Api Simulation
Agenda

• The problems with operating and testing distributed systems and microservices
• Overview of Service Virtualisation / Api Simulation
• Improving our Tests
Agenda

• The problems with operating and testing distributed systems and microservices
• Overview of Service Virtualisation / Api Simulation
• Improving our Tests
• Data Synchronisation
Agenda

- The problems with operating and testing distributed systems and microservices
- Overview of Service Virtualisation / Api Simulation
- Improving our Tests
- Data Synchronisation
- Summary
Agenda

• The problems with operating and testing distributed systems and microservices
• Overview of Service Virtualisation / Api Simulation
• Improving our Tests
• Data Synchronisation
• Summary
Distributed Systems

-INFOX
Microservices

- Company
- Flights
- Payment
- Auth
- Booking
Microservices

- Company
- Flights
- Payment
- Auth
- Booking

@mogronalol
A natural evolution...

My take on the evolution of software architecture (& Italian food) medium.com/@benorama/the- … #microservices #ravioli

1990's
SPAGHETTI-ORIENTED
ARCHITECTURE
(aka Copy & Paste)

2000's
LASAGNA-ORIENTED
ARCHITECTURE
(aka Layered Monolith)

2010's
RAVIOLI-ORIENTED
ARCHITECTURE
(aka Microservices)
Challenges...
Airline Booking Application
Airline Booking Application

• Microservice architecture
Airline Booking Application

- Microservice architecture
- Integration with third party services
Airline Booking Application

• Microservice architecture
• Integration with third party services
• Integration with legacy internal services
Airline Booking Application

- Microservice architecture
- Integration with third party services
- Integration with legacy internal services
- Heavy resource consumption
Airline Booking Application

- Microservice architecture
- Integration with third party services
- Integration with legacy internal services
- Heavy resource consumption
- A standard CI/CD pipeline
Resources

Booking
Resources

- Flights
- Rewards
- Booking
Resources

- Flights
- Rewards
- Payments

Booking
Resources

- Flights
- Rewards
- Payments
- Booking
- Fraud
Resources

- Flights
- Rewards
- Payments
- Booking
- Fraud

Cost
Licensing
Licensing

Third party

Producer

Consumer
Licensing

Third party

Producer

Consumer

Producer

Consumer
Dependency Constraints
Dependency Constraints

Producer

Consumer
Dependency Constraints

Producer

Consumer
Dependency Constraints
Flakey Dependencies
Flakey Dependencies

![Diagram showing a producer and a consumer with a directed edge between them.](image)
Flakey Dependencies

Producer

Consumer

Producer

Consumer
Doesn't exist
Doesn't exist

Consumer

?
Non Deterministic
Non Deterministic

Booking Service

Consumer
Non Deterministic

Booking Service

Consumer

{  
  "date" : "2013-01-08",
  ...
}

@mogronalol
Non Deterministic

```
Consumer

Booking Service

{  
  "date" : "2013-01-08",
  ...
}

assertThat(booking.getDate()).isEqualTo(LocalDate.of(2013, 1, 8));
```
Non Deterministic

Booking Service

Consumer

```java
assertThat(booking.getDate())
    .isEqualTo(LocalDate.of(2013, 1, 8));
```

```json
{
    "date": "2013-01-08",
    ...
}
```
Non Deterministic

```java
assertThat(booking.getDate()).isEqualTo(LocalDate.of(2013, 1, 8));
```

```json
{  
  "date": "2016-02-07",
  ...
}
```

```
assertThat(booking.getDate()).isEqualTo(LocalDate.of(2013, 1, 8));
```
Fault Injection
Fault Injection

Producer

Consumer
Fault Injection

Producer

Consumer

Producer

Consumer

@mogronalol
Fault Injection

4xx, 5xx, timeouts
Fault Injection

Operationally challenging to trigger faults

Producer

Consumer

Producer

Consumer

4xx, 5xx, timeouts
Agenda

- The problems with operating and testing distributed systems and microservices
- **Overview of Service Virtualisation / Api Simulation**
- Improving our Tests
- Data Synchronisation
- Summary
Service Virtualisation
Service Virtualisation

• Emulation of real service
Service Virtualisation

• Emulation of real service
• Non-intrusive
Service Virtualisation

- Emulation of real service
- Non-intrusive
- Lightweight - or should be
Service Virtualisation
Service Virtualisation

Consumer
Service Virtualisation

Consumer → Producer
Creating Data
Creating Data

Consumer
Creating Data

Consumer ➔ Producer
Creating Data

Consumer → Service Virtualisation Tool → Producer
Creating Data

```java
stubFor(any(urlPathEqualTo("/everything"))
    .withHeader("Accept", containing("xml"))
    .withCookie("session", matching(".*12345.*"))
    .withQueryParam("search_term", equalTo("WireMock"))
    .withBasicAuth("jeff@example.com", "jeffteenjefftjef")
    .withRequestBody(equalToXml("<search-results />
```

```json
{
  "request": {
    "urlPath": "/everything",
    "method": "ANY",
    "headers": {
        "Accept": {
            "contains": "xml"
        }
    }
  }
}
```
Proxy Mode
Proxy Mode

Consumer -> Producer
Proxy Mode

Consumer — Service Virtualisation Tool — Producer
Webserver / Facade Mode
Webserver / Facade Mode

Consumer
Webserver / Facade Mode

Consumer → Producer
Webserver / Facade Mode

Consumer

Producer
Webserver / Facade Mode
Webserver / Facade Mode

Consumer → Service Virtualisation Tool → Producer

@mogronalol
Not Mocking

Service Under Test

Http

Dependant Service
Not Mocking

Service Under Test

Mocking

Http

Dependant Service
## Not Mocking

<table>
<thead>
<tr>
<th>Service Under Test</th>
<th>Mocking</th>
</tr>
</thead>
</table>

@mogronalol
Not Mocking

Service Under Test

Virtual Service
Not Mocking

Service Under Test

Virtual Service

Mocks bleed into the architectural boundary of your application
Open Source Tooling
Open Source Tooling

Hoverfly
Dependencies without the sting
Open Source Tooling

Hoverfly

Dependencies without the sting

Welcome, friend

mountebank - over the wire test doubles

collected bank - over the wire test doubles

The apothecary

getting started
examples
client libraries
install options
command line

home imposters logs config

mountebank is the first open source tool to provide cross-platform, multi-protocol test double for every platform. Simply point your application under test to mountebank instead of the network, and it will provide the correct response as if the real network were there.
Open Source Tooling

Hoverfly
Dependences without the sting

WireMock
Mock your APIs for fast, robust and comprehensive testing

@mogronalol
Agenda

- The problems with operating and testing distributed systems and microservices
- Overview of Service Virtualisation / Api Simulation
- Improving our Tests
- Data Synchronisation
- Summary
Record and replay
Record and replay

Consumer
Record and replay

Consumer -> External Producer
Record and replay

Consumer → Service Virtualisation Tool → External Producer
Record and replay

Consumer → Service Virtualisation Tool → External Producer
JUnit Example
JUnit Example

```java
@Test
public void shouldBeAbleToRetrieveABooking() throws URISyntaxException {
    // Given
    final int bookingId = 1;

    // When
    final Booking booking = bookingServiceGateway.getBooking(bookingId);

    // Then
    assertThat(booking.getCustomer()).isEqualTo("Andrew Morgan");
    assertThat(booking.getDate()).isEqualTo(LocalDate.of(2016,3,1));
}
```
JUnit Example

@Rule
public HoverflyRule hoverflyRule = HoverflyRule.inCaptureMode("src/test/resources/external.json").build();

@Test
public void shouldBeAbleToRetrieveABooking() throws URISyntaxException {
    // Given
    final int bookingId = 1;

    // When
    final Booking booking = bookingServiceGateway.getBooking(bookingId);

    // Then
    assertThat(booking.getCustomer()).isEqualTo("Andrew Morgan");
    assertThat(booking.getDate()).isEqualTo(LocalDate.of(2016,3,1));
}
JUnit Example

```json
{
  "data": [
    {
      "request": {
        "path": "/api/bookings/1",
        "method": "GET",
        "destination": "www.my-test.com",
        "query": null,
        "body": ""
      },
      "response": {
        "status": 200,
        "body": "{"customer":"Andrew Morgan","date":"2016-03-01"}
      },
      "headers": {
        "Content-Type": ["application/json"
      }
    }
  ]
}
```
JUnit Example

@Rule
public HoverflyRule hoverflyRule = HoverflyRule.buildFromClassPathResource("external.json").build();

@Test
public void shouldBeAbleToRetrieveABooking() throws URISyntaxException {
    // Given
    final int bookingId = 1;

    // When
    final Booking booking = bookingServiceGateway.getBooking(bookingId);

    // Then
    assertThat(booking.getCustomer()).isEqualTo("Andrew Morgan");
    assertThat(booking.getDate()).isEqualTo(LocalDate.of(2016,3,1));
}

@Rule
public HoverflyRule hoverflyRule = HoverflyRule.buildFromClassPathResource("external.json").build();

@Test
public void shouldBeAbleToRetrieveABooking() throws URISyntaxException {
    // Given
    final int bookingId = 1;

    // When
    final Booking booking = bookingServiceGateway.getBooking(bookingId);

    // Then
    assertThat(booking.getCustomer()).isEqualTo("Andrew Morgan");
    assertThat(booking.getDate()).isEqualTo(LocalDate.of(2016,3,1));
}
Advantages
Advantages

• Isolation
Advantages

- Isolation
- Flaky dependencies
Advantages

• Isolation
• Flaky dependencies
• Rate limiting
Advantages

• Isolation
• Flaky dependencies
• Rate limiting
• Deterministic
Advantages

• Isolation
• Flaky dependencies
• Rate limiting
• Deterministic
• Licensing
Deployment and Infrastructure
Deployment and Infrastructure

Consumer -> External Producer
Deployment and Infrastructure

- Internal Producer
- Internal Producer
- Internal Producer
- Consumer
- External Producer

@mogronalol
Deployment and Infrastructure

- Consumer
  - Virtual Service
  - Virtual Service
  - Virtual Service

@mogronalol
CLI Example
CLI Example

$ hoverctl start --proxy-port 8080
CLI Example

$ hoverctl start --proxy-port 8080
$ hoverctl import accounts-service-simulation.json
CLI Example

$ hoverctl start --proxy-port 8080
$ hoverctl import accounts-service-simulation.json

$ hoverctl start --proxy-port 8081
CLI Example

$ hoverctl start --proxy-port 8080
$ hoverctl import accounts-service-simulation.json

$ hoverctl start --proxy-port 8081
$ hoverctl import payments-service-simulation.json
CLI Example

$ hoverctl start --proxy-port 8080
$ hoverctl import accounts-service-simulation.json

$ hoverctl start --proxy-port 8081
$ hoverctl import payments-service-simulation.json


[{"name":"foo"}]
$ hoverctl start --proxy-port 8080
$ hoverctl import accounts-service-simulation.json

$ hoverctl start --proxy-port 8081
$ hoverctl import payments-service-simulation.json


[{"name":"foo"}]


[{"amount":"£23.27"}]
Fault Injection
Fault Injection

Consumer -> Service Virtualisation Tool -> External Producer
Fault Injection

Consumer → Service Virtualisation Tool → Middleware → External Producer
def main():
    data = sys.stdin.readlines()
    # this is a json string in one line so we are interested in that one line
    payload = data[0]
    logging.debug(payload)

    payload_dict = json.loads(payload)

    payload_dict['response']['status'] = 201
    payload_dict['response']['body'] = "body was replaced by middleware"

    # now let's sleep for 2 seconds
    sleep(2)

    # returning new payload
    print(json.dumps(payload_dict))
Advantages
Advantages

• Transparent
Advantages

• Transparent
• Configurable
Advantages

• Transparent
• Configurable
• Operationally simple
Documentation-Driven

Consumer

?
Documentation-Driven

Consumer
Documentation-Driven

Consumer

Transform

swagger
Documentation-Driven

Consumer -> Virtual Service

Transform

@mogronalol
Advantages
Advantages

• Documentation Driven Development
Advantages

• Documentation Driven Development
• Work in Parallel
• The problems with operating and testing distributed systems and microservices
• Overview of Service Virtualisation / Api Simulation
• Improving our Tests
• Data Synchronisation
• Summary
Data Synchronisation Issues
Data Synchronisation Issues

![Diagram showing data synchronisation issue between Consumer and Virtual Service]
Data Synchronisation Issues

Consumer — X — Real Service
## Ownership

<table>
<thead>
<tr>
<th>Own Consumer</th>
<th>Own Producer</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>✗</td>
<td></td>
</tr>
<tr>
<td>✗</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>✗</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>

@mogronalol
## Ownership

<table>
<thead>
<tr>
<th>Own Consumer</th>
<th>Own Producer</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>✗</td>
<td>Integration</td>
</tr>
<tr>
<td>✗</td>
<td>✔️</td>
<td>Contract Testing</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>
Integration Contract Testing

Scheduled Build

http://martinfowler.com/bliki/IntegrationContractTest.html

@mogronalol
Integration Contract Testing

Scheduled Build

Reproduce Simulations

http://martinfowler.com/bliki/IntegrationContractTest.html

@mogronalol
Integration Contract Testing

Scheduled Build → Reproduce Simulations → Diff

http://martinfowler.com/bliki/IntegrationContractTest.html

@mogronalol
Integration Contract Testing

Scheduled Build → Reproduce Simulations → Diff

Integration Testing or Contract Diff

http://martinfowler.com/bliki/IntegrationContractTest.html

@mogronalol
Integration Contract Testing

Scheduled Build → Reproduce Simulations → Diff

Integration Testing
or
Contract Diff

No diff?

http://martinfowler.com/bliki/IntegrationContractTest.html

@mogronalol
Integration Contract Testing

http://martinfowler.com/bliki/IntegrationContractTest.html

@mogronalol
# Ownership

<table>
<thead>
<tr>
<th>Own Consumer</th>
<th>Own Producer</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Checkmark" /></td>
<td><img src="https://via.placeholder.com/15" alt="X" /></td>
<td>Integration Contract Testing</td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="X" /></td>
<td><img src="https://via.placeholder.com/15" alt="Checkmark" /></td>
<td></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Checkmark" /></td>
<td><img src="https://via.placeholder.com/15" alt="Checkmark" /></td>
<td></td>
</tr>
</tbody>
</table>
## Ownership

<table>
<thead>
<tr>
<th>Own Consumer</th>
<th>Own Producer</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✗</td>
<td>Integration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contract Testing</td>
</tr>
<tr>
<td>✗</td>
<td>✓</td>
<td>Virtual Service as First Class Citizen</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Virtual Service as First Class Citizen
Virtual Service as First Class Citizen

Run and Capture Service Tests
Virtual Service as First Class Citizen

Run and Capture Service Tests → Transform to simulation
Virtual Service as First Class Citizen

- Run and Capture Service Tests
- Transform to simulation
- Publish Virtual Service
Virtual Service as First Class Citizen

- Parse Contract
- Run and Capture Service Tests
- Transform to simulation
- Publish Virtual Service
Virtual Service as First Class Citizen

1. Parse Contract
2. Run and Capture Service Tests
3. Parse Documentation
4. Transform to simulation
5. Publish Virtual Service
### Ownership

<table>
<thead>
<tr>
<th>Own Consumer</th>
<th>Own Producer</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>✗</td>
<td>Integration</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Contract Testing</td>
</tr>
<tr>
<td>✗</td>
<td>✔️</td>
<td>Virtual Service as</td>
</tr>
<tr>
<td></td>
<td></td>
<td>First Class Citizen</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
<td></td>
</tr>
</tbody>
</table>

@mogronalol
# Ownership

<table>
<thead>
<tr>
<th>Own Consumer</th>
<th>Own Producer</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>❌</td>
<td>Integration Contract Testing</td>
</tr>
<tr>
<td>❌</td>
<td>✔️</td>
<td>Virtual Service as First Class Citizen</td>
</tr>
<tr>
<td>✔️</td>
<td>✔️</td>
<td>Consumer Driven Contract Testing</td>
</tr>
</tbody>
</table>
Consumer Driven Contract Testing

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

Consumer Defines Contract

Generate Producer Stubs

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

Consumer Defines Contract
Generate Producer Stubs
Implement Consumer

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

Consumer Defines Contract

Generate Producer Stubs

Implement Consumer

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

Consumer Defines Contract

Generate Producer Stubs

Implement Consumer

Generate Producer Contract Tests

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

- Consumer Defines Contract
- Generate Producer Stubs
- Implement Consumer

- Generate Producer Contract Tests
- Implement Producer

http://www.martinfowler.com/articles/consumerDrivenContracts.html
@mogronalol
Consumer Driven Contract Testing

Consumer Defines Contract

Generate Producer Stubs

Implement Consumer

Generate Producer Contract Tests

Implement Producer

Run Consumer Contract Tests

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

Consumer Defines Contract

Generate Producer Stubs

Implement Consumer

Generate Producer Contract Tests

Implement Producer

Run Consumer Contract Tests

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Consumer Driven Contract Testing

Change producer

Run Consumer Contract Tests

http://www.martinfowler.com/articles/consumerDrivenContracts.html

@mogronalol
Spring Cloud Contract

Spring Cloud Contract is an umbrella project holding solutions that help users in successfully implementing the Consumer Driven Contracts approach. Currently, Cloud Contract consists of the Spring Cloud Contract Verifier project. Spring Cloud Contract Verifier is a tool that enables Consumer Driven Contract development of JVM-based applications. It is shipped with Contract Definition
Tools

Spring Cloud Contract

Spring Cloud Contract is an umbrella project holding solutions that help users in successfully implementing the Consumer Driven Contracts approach. Currently, Cloud Contract consists of the Spring Cloud Contract Verifier project. Spring Cloud Contract Verifier is a tool that enables Consumer Driven Contract development of JVM-based applications. It is shipped with Contract Definition Language (CDL) support.

DiUS / pact-jvm

JVM version of Pact. Enables consumer driven contract interaction playback and verification for the service provider.
Testing Pyramid

End to end

Integration / Service Testing

Unit Testing
Testing Pyramid

- Unit Testing
- Integration / Service Testing
- Consumer Driven Contract Testing
- End to end
Testing Pyramid

End to end

Consumer Driven
Contract Testing

Integration / Service Testing

Unit Testing
Summary

• Cost reduction from simulated services
Summary

• Cost reduction from simulated services
• Easy deployment of simulations
Summary

• Cost reduction from simulated services
• Easy deployment of simulations
• Less dependant on end to end testing
Summary

• Cost reduction from simulated services
• Easy deployment of simulations
• Less dependant on end to end testing
• Better feedback cycle
Questions?

@mogronalol

https://github.com/mogronalol