

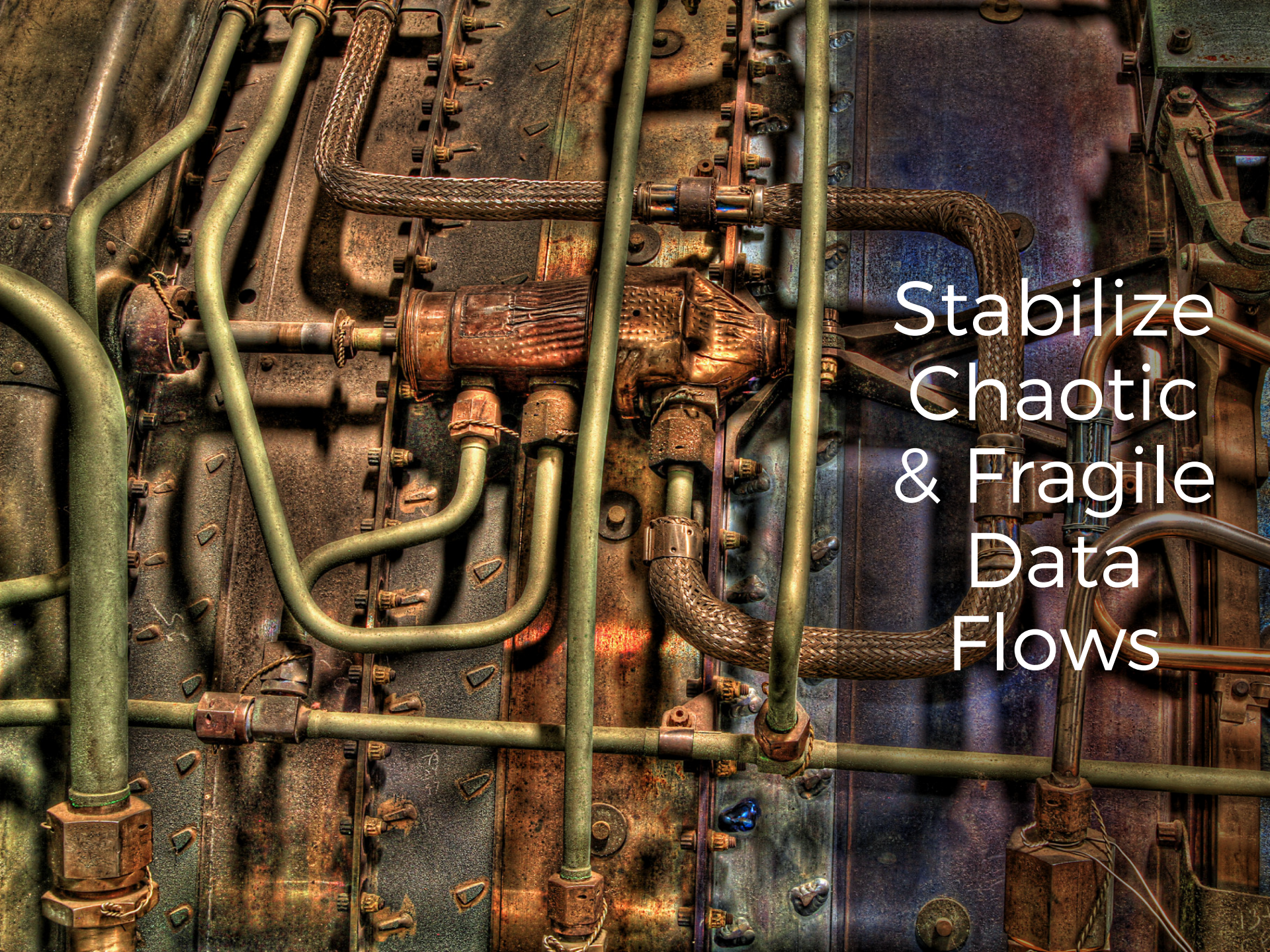


Spring Cloud Contract

Consumer-Driven Contracts Architecture

@RoboNovotny

December 12nd, 2018

A detailed photograph of a complex industrial machine, possibly a high-pressure reactor or a specialized engine. The machine is constructed from dark, weathered metal plates, likely steel or aluminum, which show signs of use and corrosion. A central component is a large, cylindrical copper or brass vessel with a textured, possibly knurled surface. This central vessel is connected to a network of pipes and hoses. Some pipes are smooth and light-colored, while others are braided metal hoses. The connections are made with various fittings, including hex nuts and flanges. The overall appearance is that of a well-used, intricate piece of engineering. The lighting is dramatic, highlighting the textures and colors of the different materials.

Stabilize
Chaotic
& Fragile
Data
Flows

A close-up photograph of a human hand reaching out to touch the paw of a large bear. The bear's paw is covered in dark, shaggy fur and has four long, sharp, black claws. The background is slightly blurred, showing green foliage and a clear blue sky. The text "Consumer-Driven Contract" is overlaid in white on a semi-transparent dark band across the bottom of the image.

Consumer-Driven Contract



Spring Cloud Contract

1. Define contract in YAML | Groovy
2. Make sure server and client adhere to the contract

request:

method: POST

url: /check

body:

text: bacardi, martel, hennessy...

headers:

Content-Type: application/json;charset=UTF-8

response:

status: 200

body:

match: 100

headers:

Content-Type: application/json;charset=UTF-8



Supported Technologies

- Spring Boot
- HTTP
- Spring Integration
- Spring Cloud Stream
- Camel
- RabbitMQ/AMQP



Server side: HTTP

- Write down the contract
- Release as an artifact



Client Side: HTTP

- Code client-side
- Verify contract via integration test



Client Side: Integration Tests

```
@AutoConfigureStubRunner(ids = {  
    "sk.eastcode:scc-producer:+:stubs:6565"  
})
```



Client Side: Integration Tests

- **Stub Runner:**
 - download contracts from repo
 - parse contract
- **WireMock**
 - mimic a HTTP server
- **RestTemplate:**
 - Spring client with mock client



Server Side: HTTP

- Implement server-side
- Verify contract via integration test



Server Side: Integration Tests

- **Maven/Gradle Plugin :**
 - autogenerate tests according to contract
- **Spring MockMVC + RestAssured**
 - run tests against HTTP endpoint implementation



Contract Changes?

- **release!**

 - a new contract artifact to repo

- **break'em**

 - client tests and server-side tests
will possibly break!

- **fix'em!**

The image features a dark grey background with several overlapping, semi-transparent green polygons of various shapes and sizes on the left side. The polygons are arranged in a cluster, with some overlapping others, creating a layered effect. The colors range from a bright lime green to a darker, forest green.

Thanks!

@RoboNovotny