
PUZZLE ITC
changing IT for the better

Spring Cloud Contract: Consumer Driven Contracts

Dmitri Karpovich,
Puzzle ITC

Agenda

16.11.2017

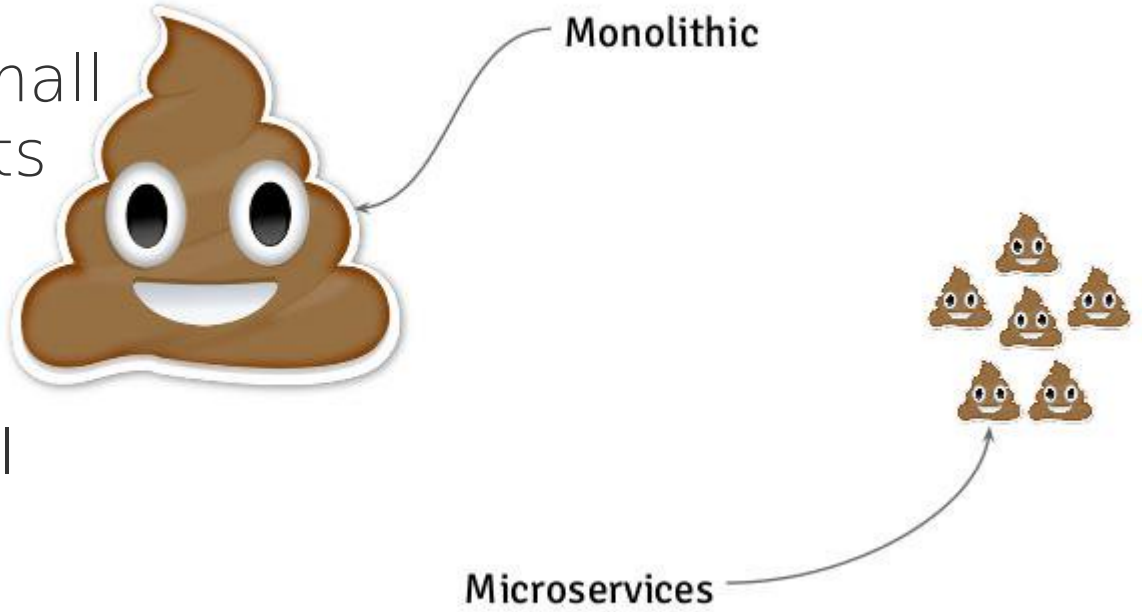
1. Short theory intro
2. Sample project description
3. Problem statement
4. Finding a solution
5. Seeing it in code
6. Q&A
7. Go back to Zurich

1

Short theory intro

Microservices architecture

microservice architectural style - developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API



2

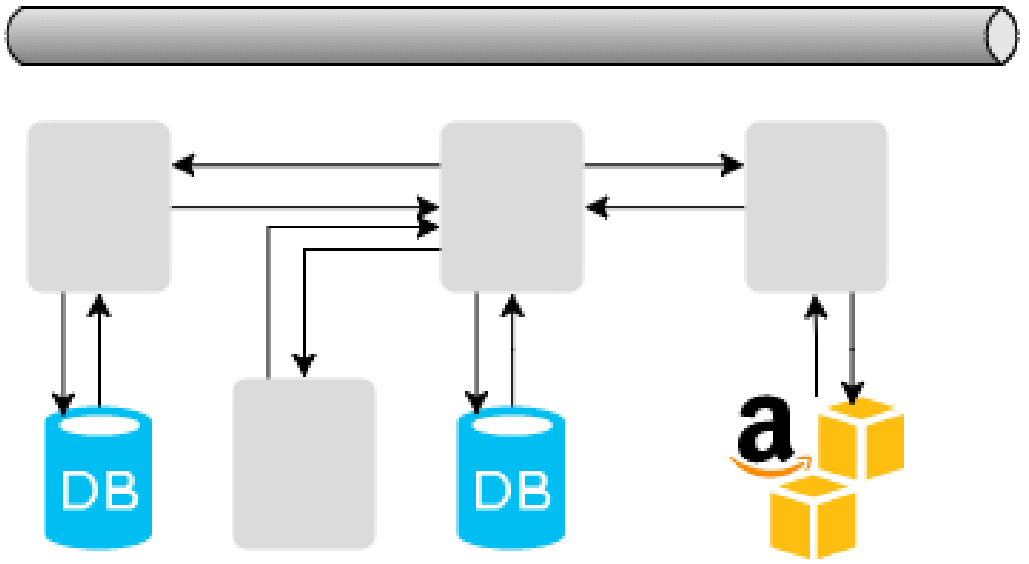
Sample project

BeerWithSmb

Simplifies finding someone to have a beer in the evening.

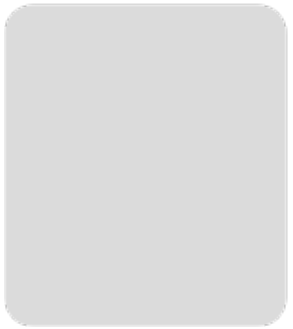
Consists of

- Several microservices
- External systems (file storages, DBs, Calendar connectors etc)



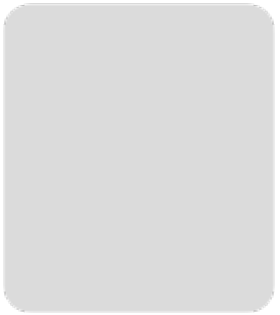
BeerWithSmb - Zoom in

intensions



Processes expressed desires of person A having a drink with person B

buddies



Manages persons, their contact information etc.

How it works

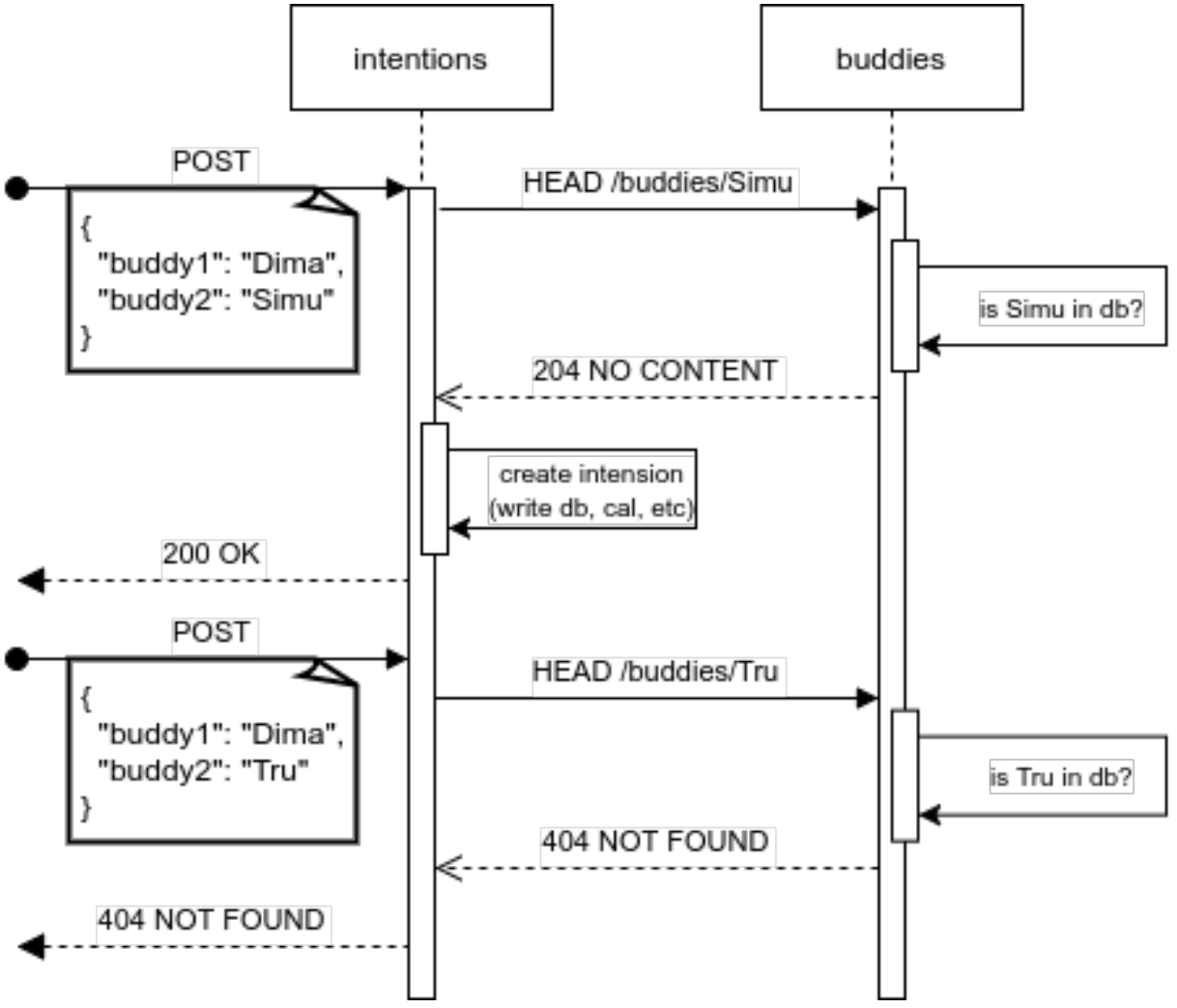
User sends a POST-request with JSON containing his name and name of a desired drink-buddy to intentions-microservice.

Intentions-microservice validates the name of the buddy by querying it from buddies-microservice and creates an intention if validation is OK.

Otherwise the intention is not created.

How it works - 2

Let us say buddies microservice knows “Simu” and doesn’t know “Tru”



3

What’s the problem?

Problem is quite big

How do I test the interaction between *intensions* and *buddies*?

How can I be sure that they can talk to each other and „**understand**“ each other during continuous development process?

What if I‘d really like to change the structure of JSON that *buddies* return? Will *intensions* still work or crash in production?

4

So what do we do?

Solution 1

Start both microservices and do pure e2e-testing

+ simulates prod. Environment

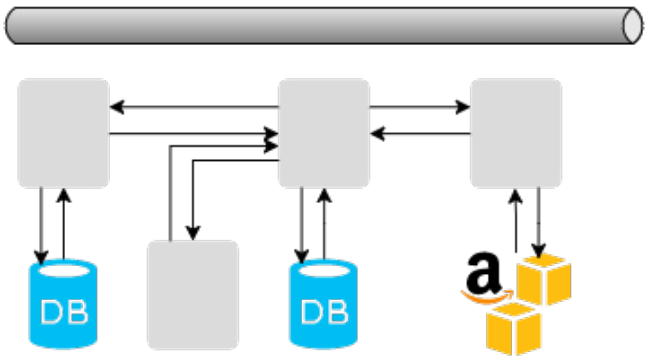
– can be slow

- can be expensive

- what do I do with external systems?

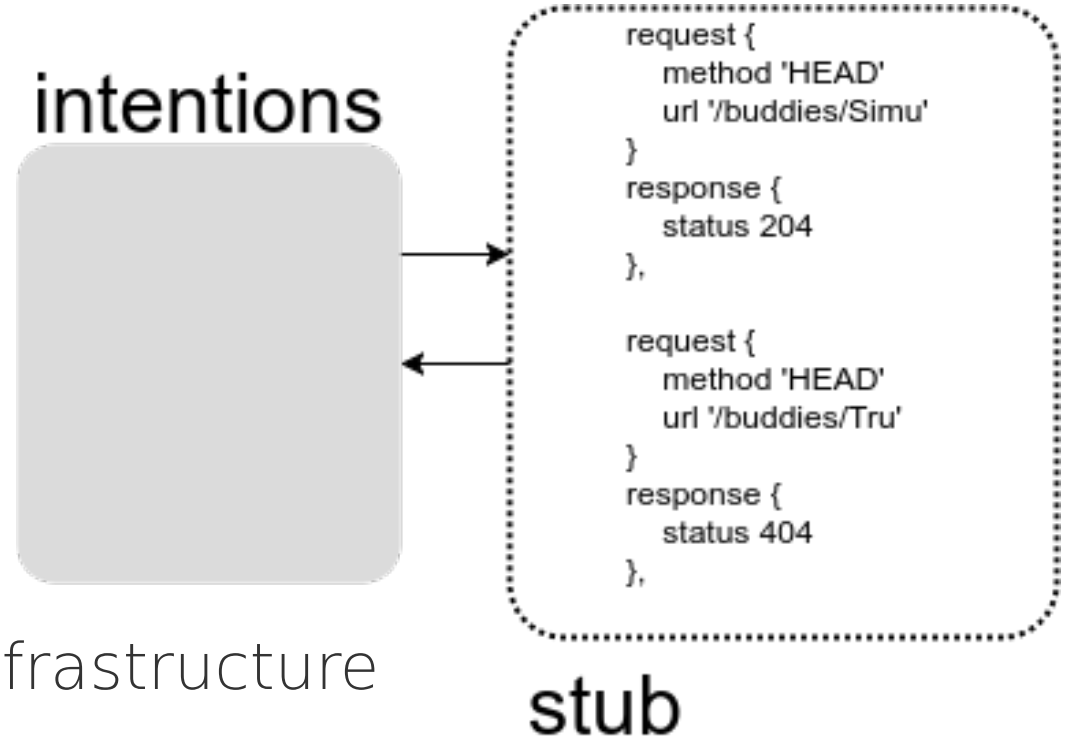
- what about other needed microservices?

- tests can fail mysteriously



Solution 2

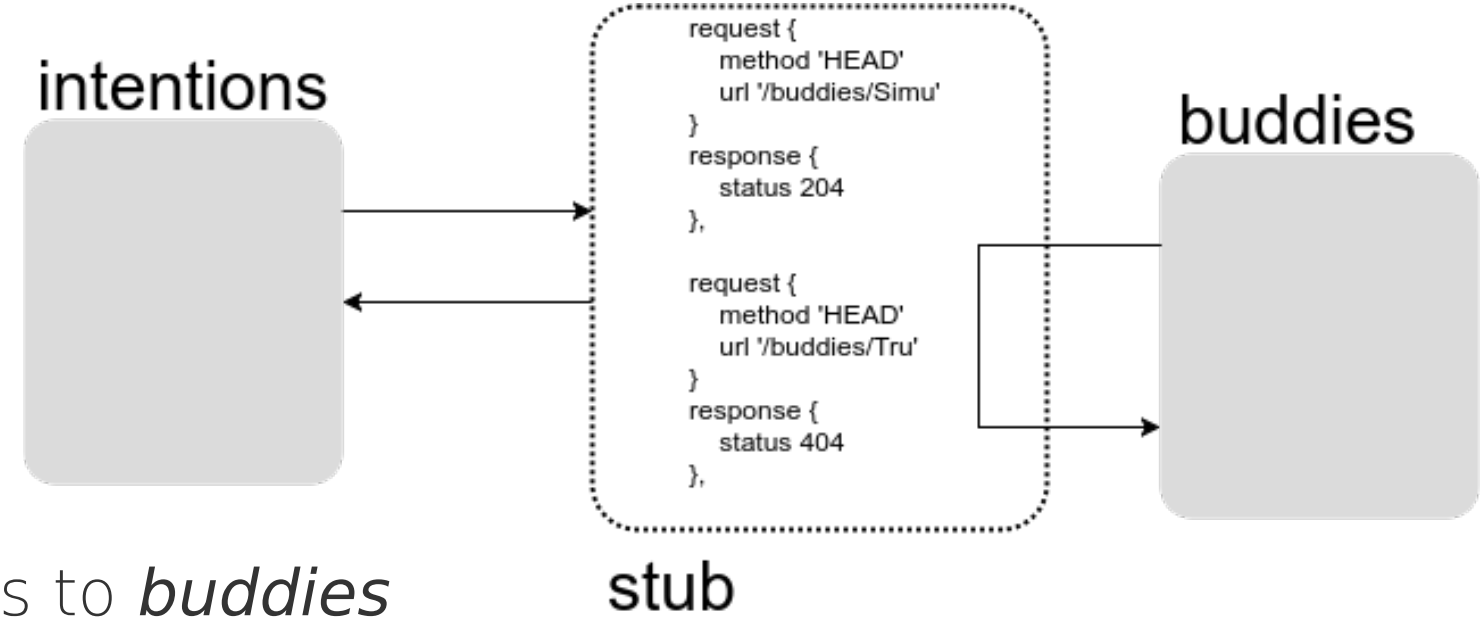
Mock *buddies* with a stub that imitates it and run tests in *intensions* against mocked *buddies*.



\+ no need in starting whole infrastructure

– no guaranty that stub is actual and reflects the latest state of *buddies*

Solution 3



1. Move the stubs to ***buddies***
2. Validate ***buddies*** against the stubs
3. ***Intentions*** imports stubs and runs tests against them

Spring Cloud Contract

In buddies:

Stubs are groovy DSL files stored in *buddies*

Generates unit-tests based on stubs so that *buddies* is validated

Exports *buddies-stubs.jar* as a separate artifact

In intentions:

Grabs *buddies-stubs.jar*

Starts the mock server that behaves exactly as written in stubs, thus allowing appropriate tests from *intentions* to be executed

5

Show me some code

Example scenario: HEAD → GET

Generated tests in buddies will fail → have to fix the contract

New buddies-stubs.jar is installed to localrepo

Intentions grabs new stubs from localrepo

Mock server is started with new buddies-stubs.jar

Tests in intentions fail because they expect valid response on HEAD-requests → have to fix intentions

6

Q&A

Thank you!

...for not having slept during the presentation