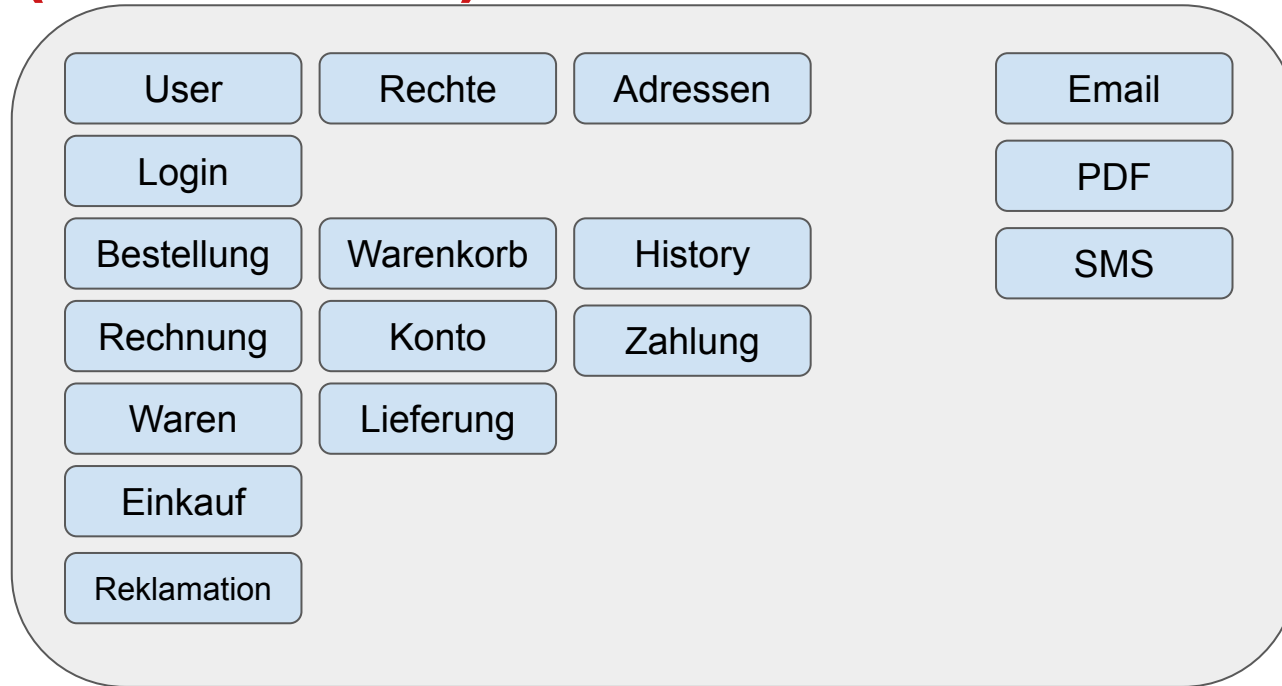

Services - Warum, wann und wie Micro?

Schwabeneder Markus

(Modularer) Monolith



Service

User

Rechte

Adressen

Login

Bestellung

Warenkorb

History

Rechnung

Konto

Zahlung

Waren

Lieferung

Einkauf

Reklamation

Queues

Bus

Email

PDF

SMS

Microservice

User

Rechte

Adressen

Login

Bestellung

Warenkorb

History

Rechnung

Konto

Zahlung

Waren

Lieferung

Einkauf

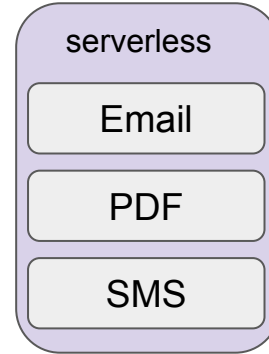
Reklamation

Email

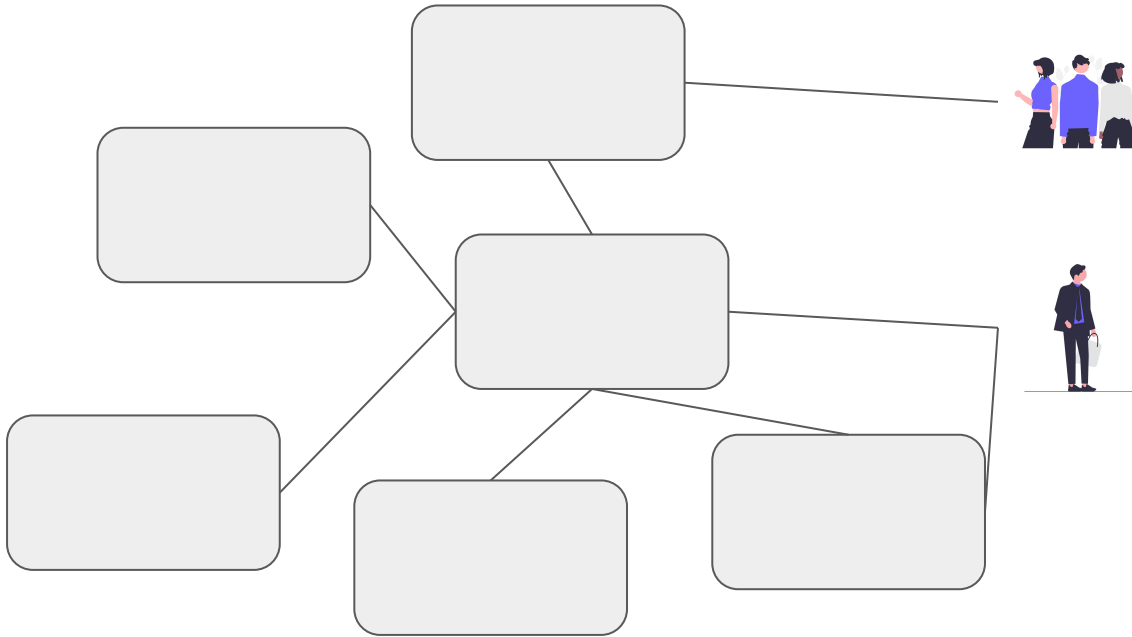
PDF

SMS

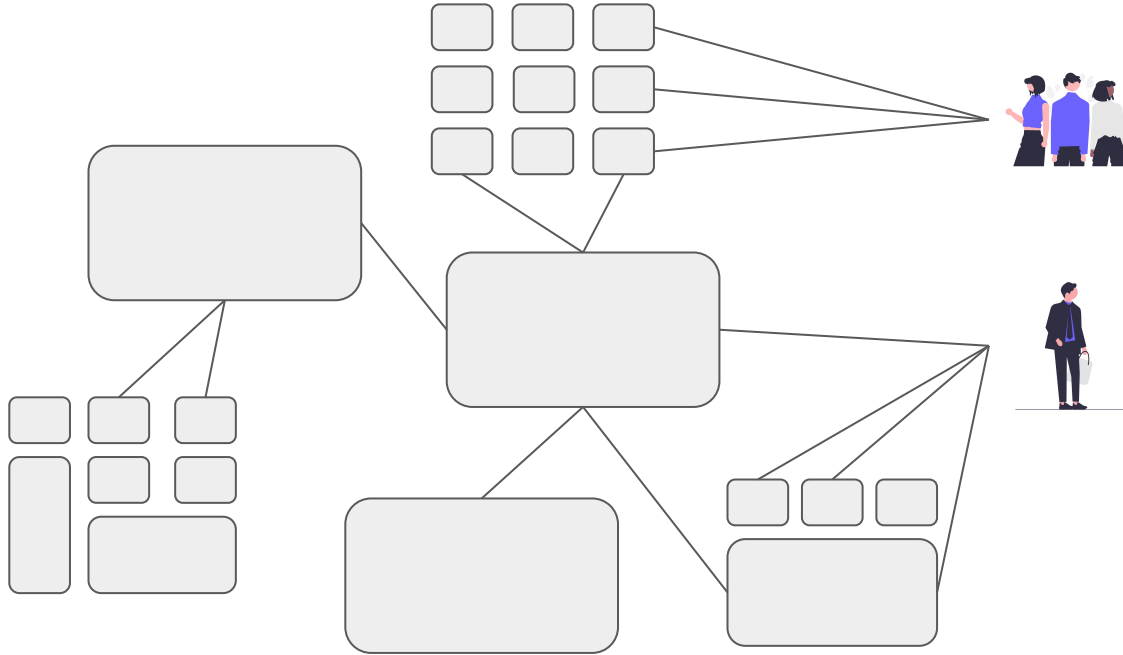
Microservice



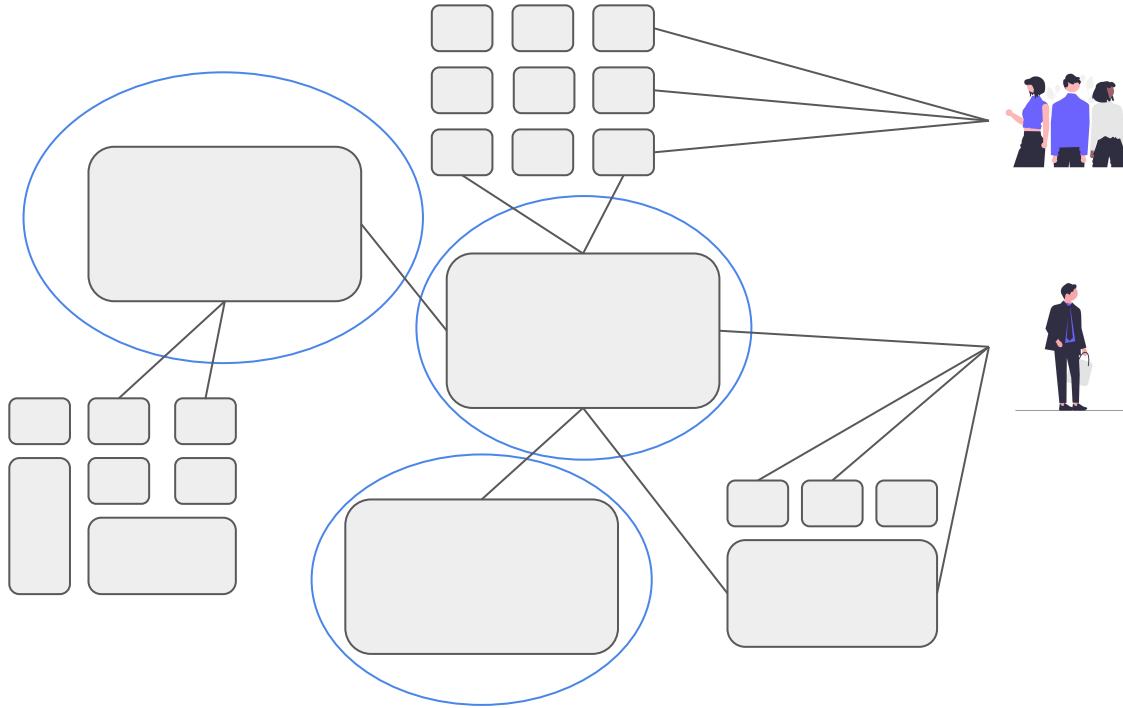
Architekturen und Scope



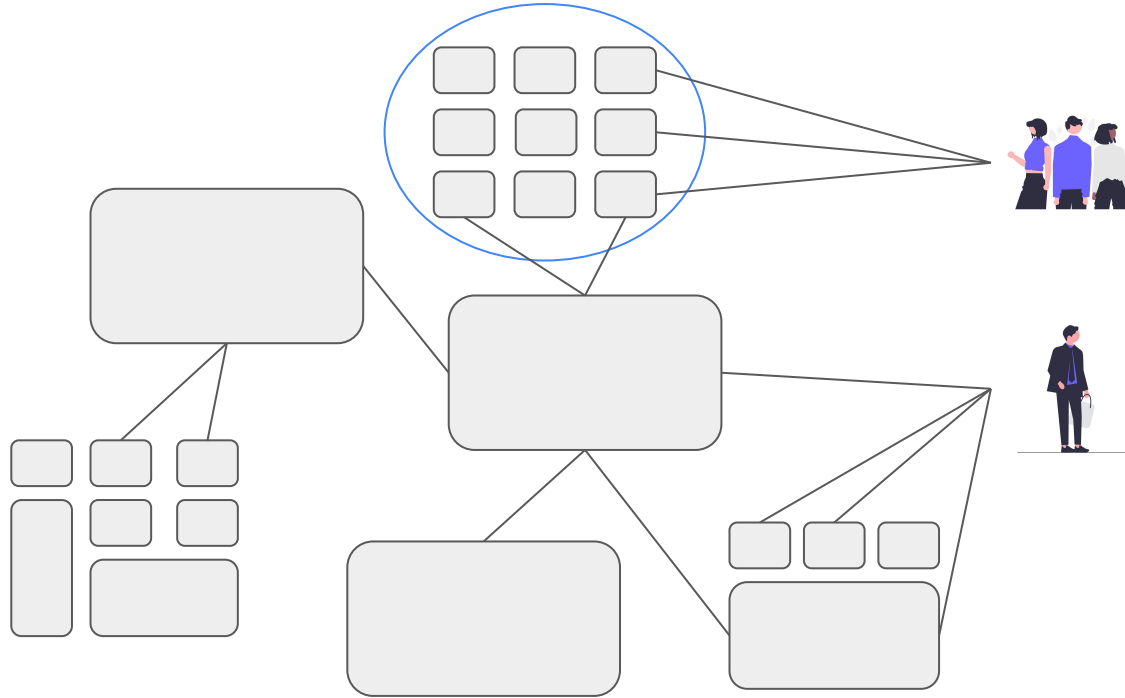
Architekturen und Scope



Monolith



Microservice Architecture



Vorteile des Monolithen

- Kein Kommunikations-Overhead

Vorteile des Monolithen

- Kein Kommunikations-Overhead
- Einfaches Deployment

Vorteile des Monolithen

- Kein Kommunikations-Overhead
- Einfaches Deployment
- Als Gesamtsystem weniger kompliziert

Vorteile von service-basierten Systemen

- Viel besser skalierbar

Vorteile von service-basierten Systemen

- Einzelne Services sind einfach

Vorteile von service-basierten Systemen

- Verschiedene Teams arbeiten unabhängig

Vorteile von service-basierten Systemen

- Verschiedene Technologien

Vorteile von service-basierten Systemen

- Unterschiedliche Security-Levels

Vorteile von service-basierten Systemen

- Unterschiedliche SLAs

Vorteile von service-basierten Systemen

- Keine Big-Bang-Deployments

Vorteile von service-basierten Systemen

- A/B Testing einfach umsetzbar

Vorteile von service-basierten Systemen

- Viel besser skalierbar
- Einzelne Services sind einfach
- Verschiedene Teams arbeiten unabhängig
- Verschiedene Technologien
- Unterschiedliche Security-Levels
- Unterschiedliche SLAs
- Keine Big-Bang-Deployments
- A/B Testing einfach umsetzbar

Microservicearchitecture

- kein zentrales “Steuer”-Service

Microservicearchitecture

- kein zentrales “Steuer”-Service
- deutlich robuster und ausfallsicherer

Microservicearchitecture

- kein zentrales “Steuer”-Service
- deutlich robuster und ausfallsicherer
- komplexer und schwieriger zu überblicken



**1. Dokumentiere oder noch besser lass den Code
dokumentieren!**

Swagger

store Access to Petstore orders		^
GET	/store/inventory Returns pet inventories by status	🔒 ✓
POST	/store/order Place an order for a pet	✓
GET	/store/order/{orderId} Find purchase order by ID	✓
DELETE	/store/order/{orderId} Delete purchase order by ID	✓
user Operations about user		Find out more about our store ^
POST	/user/createWithList Creates list of users with given input array	✓
GET	/user/{username} Get user by user name	✓

Swagger

POST
/pet
Add a new pet to the store

Parameters

Name

Description

body

object

(body)

required

Pet object that needs to be added to the store

Example Value

Model

```
{
  "id": 0,
  "category": {
    "id": 0,
    "name": "string"
  },
  "name": "doggie",
  "photoUrls": [
    "string"
  ],
  "tags": [
    {
      "id": 0,
      "name": "string"
    }
  ],
  "status": "available"
}
```

Parameter content type

application/json

Responses

Response content type

application/json

Code

Description

405

Invalid input

© SEQIS GmbH, 2024

Swagger

GET

/pet/findByStatus Finds Pets by status

🔒 ⬆

Multiple status values can be provided with comma separated strings

Parameters

Try it out

Name	Description
status * required array[string] (query)	Status values that need to be considered for filter Available values : available, pending, sold <div> <div>available</div> <div>pending</div> <div>sold</div> </div>

Responses

Response content type application/json

Code	Description
200	successful operation <div> <div>Example Value</div> <div>Model</div> </div> <pre> { "id": 0, "category": { "id": 0, "name": "string" }, "name": "doggie", "photoUrls": ["string"], "tags": [{ "id": 0, "name": "string" }], "status": "available" } </pre>
400	Invalid status value



2. Teste die Verträge! Automatisiert!

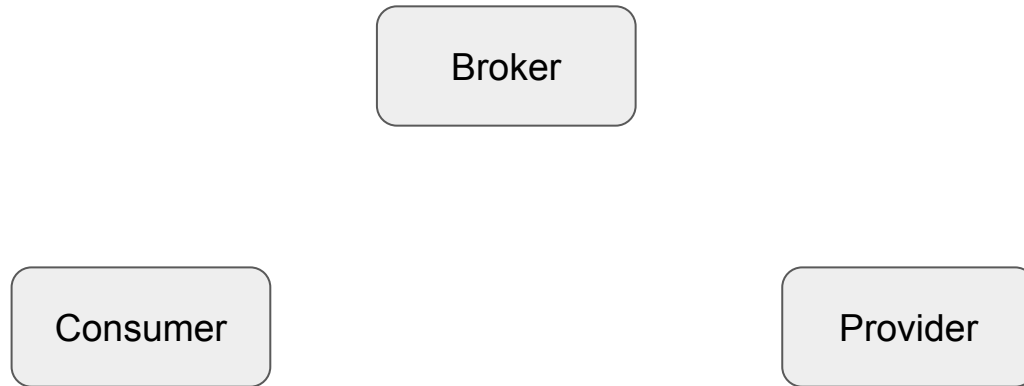
- Contract Testing
- API Testing

Contract Testing



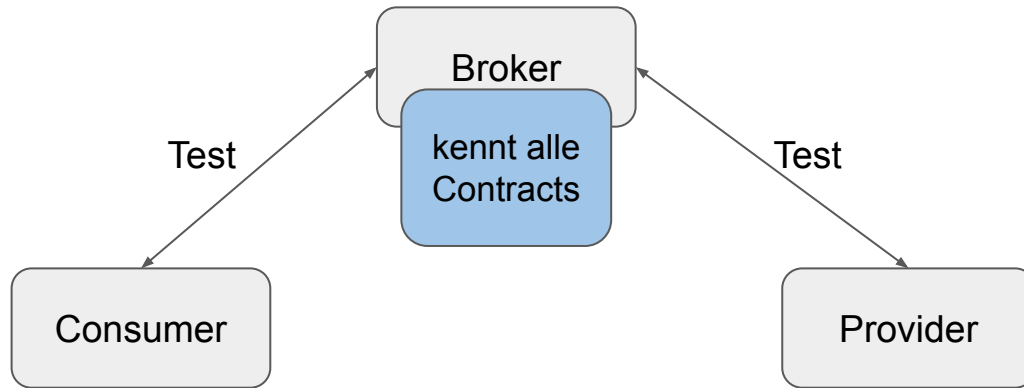
Contract Testing

Beispiel pact.io



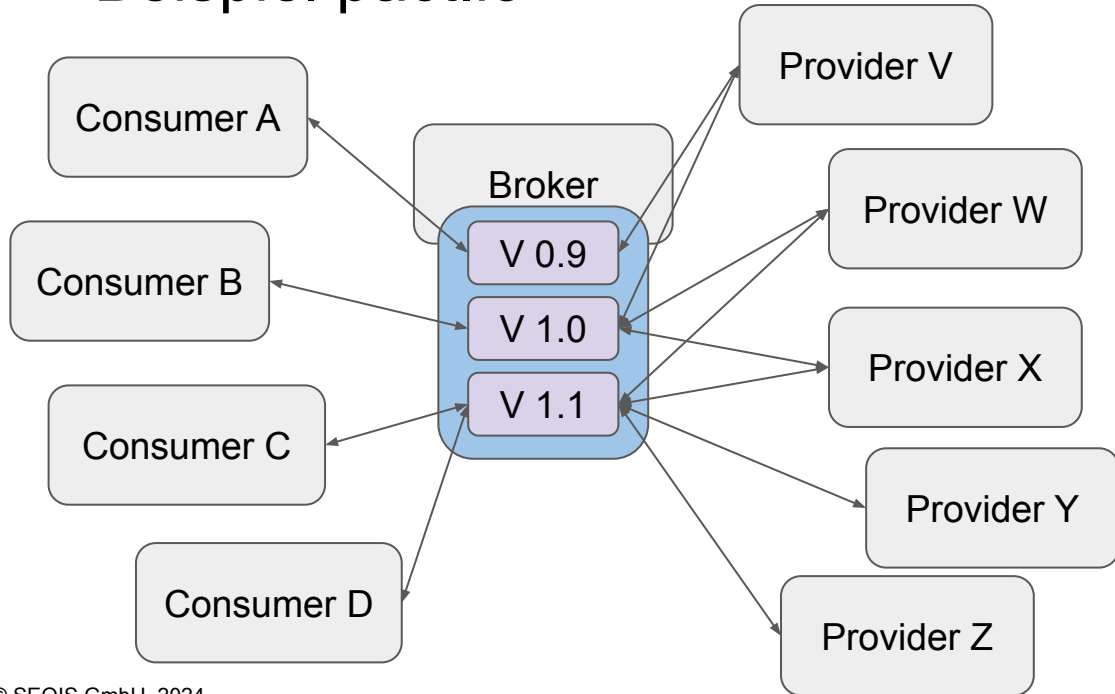
Contract Testing

Beispiel pact.io



Contract Testing

Beispiel pact.io



API-Testing

- Contracts beschreiben keine Logik



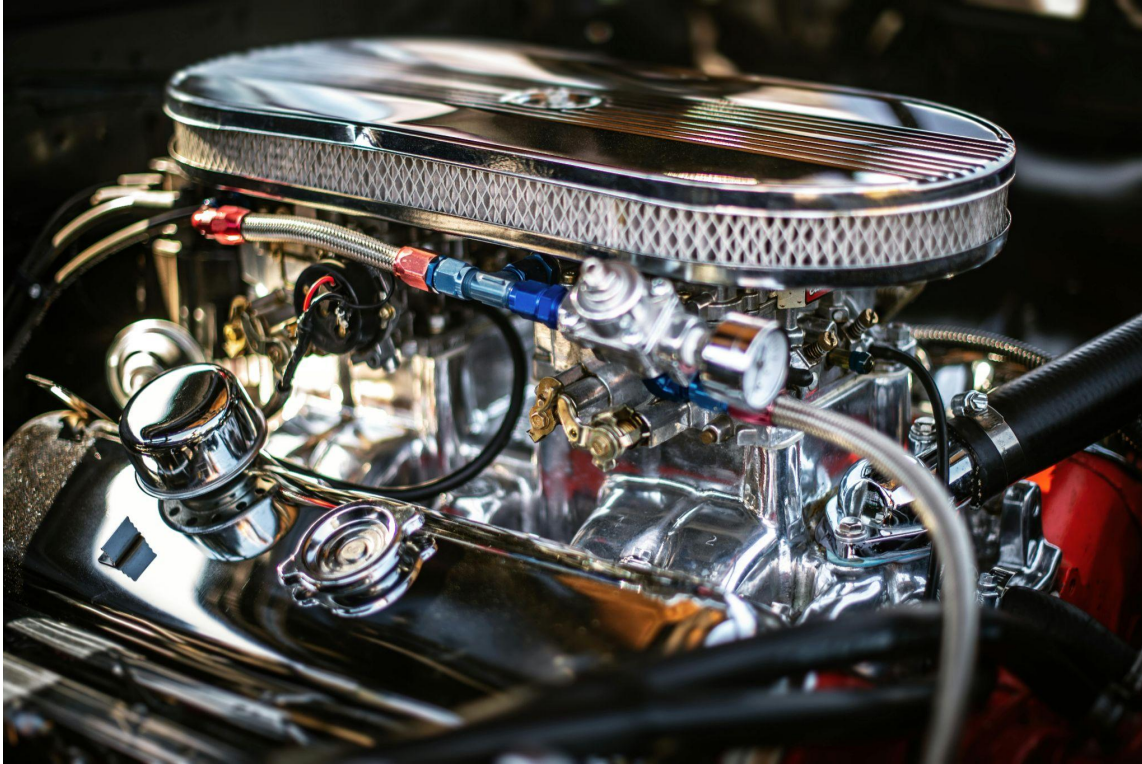
3. Automatisiere dein Deployment!



4. Microservices brauchen kein Mikromanagement!



**5. Löse deine Probleme bevor du neue Herausforderungen
angehst!**



(Micro-) Sevices

- Bieten Lösung für gewisse technische Probleme

(Micro-) Sevices

- Bieten Lösung für gewisse technische Probleme
- Lösen keine organisatorischen Probleme

(Micro-) Sevices

- Bieten Lösung für gewisse technische Probleme
- Lösen keine organisatorischen Probleme
- Lösen keine prozesstechnischen Probleme

(Micro-) Sevices benötigen

- mehr Know-how

(Micro-) Sevices benötigen

- mehr Know-how
- höheren Automatisierungsgrad

(Micro-) Sevices benötigen

- mehr Know-how
- höheren Automatisierungsgrad
- selbständige Teams und Partner mit hohem Qualitätsbewusstsein

(Micro-) Sevices benötigen

- mehr Know-how
- höheren Automatisierungsgrad
- selbständige Teams und Partner mit hohem Qualitätsbewusstsein
- komplexere Infrastruktur



6. Führe Quality Gates für Services ein!

Quality Gates

- Test Coverage

Quality Gates

- Test Coverage
- Statische Codeanalyse

Quality Gates

- Test Coverage
- Statische Codeanalyse
- Security Tests

Quality Gates

- Test Coverage
- Statische Codeanalyse
- Security Tests
- nicht-funktionale Tests

Unterschiedlich für jedes Service

- intern vs. extern (Entwicklung)

Unterschiedlich für jedes Service

- intern vs. extern (Entwicklung)
- intern vs. extern (Benutzer)

Unterschiedlich für jedes Service

- intern vs. extern (Entwicklung)
- intern vs. extern (Benutzer)
- Kritikalität des Services

Unterschiedlich für jedes Service

- intern vs. extern (Entwicklung)
- intern vs. extern (Benutzer)
- Kritikalität des Services
- Zugriffsmöglichkeiten



7. Mache Last- und Performancetests sowohl für die einzelnen Services als auch für das Gesamtsystem.



8. Viele Wege führen nach Rom, nimm den richtigen!



9. Nicht ohne triftigen Grund Micro



10. Kein Gold Plating

Modulare Entwicklung

