

Microservices



Max Jonas Werner / @makkes



Was ich mache

The screenshot shows a project management software interface with a dark theme. At the top, there's a navigation bar with a 'CONNECT' button, a 'Menu' dropdown, a search bar, and several icons for notifications, calendar, and user profile.

The main area displays a project titled 'Project New Website'. The navigation bar for the project includes 'Info', 'Microblog' (which is currently selected), 'Drive 22', 'Notes 2', 'Tasks 46', and a '+ Add' button.

The 'Microblog' section shows two posts:

- Thomas Kreye** posted 53 minutes ago: "Came across this nice article about Flat Design. Worth spending a few minutes." with a link to "The Basics of Flat Design".
Reactions: You like this 5 Likes | Comment ↗
- Toni Torpedo** posted 2 hours ago: "Hey everyone, let's get creative. We are looking for some fun ideas for the About Us section of our new Website! Any hint is welcome. Looking forward to your serious and crazy ideas!"
Reactions: Like 11 Likes | 7 comments | Comment ↗
Text: Show 5 more comments

At the bottom of the microblog feed, there's a partial view of another comment by **Benjamin Tittel** posted 15 minutes ago.

The left sidebar contains a 'My Organisation' tree with nodes like 'Groups', 'Developers', 'Project New Website', 'Customers', 'Just Just', 'Concepts', 'Java Talk', 'Scrum Team-A', and 'Marketing & Sales'. A vertical sidebar on the right shows a list of user profiles.

Worum geht's hier?

- Software-Entwicklung
- Software-Administration
- Web-Architektur
- Produktentwicklung
- Bingo



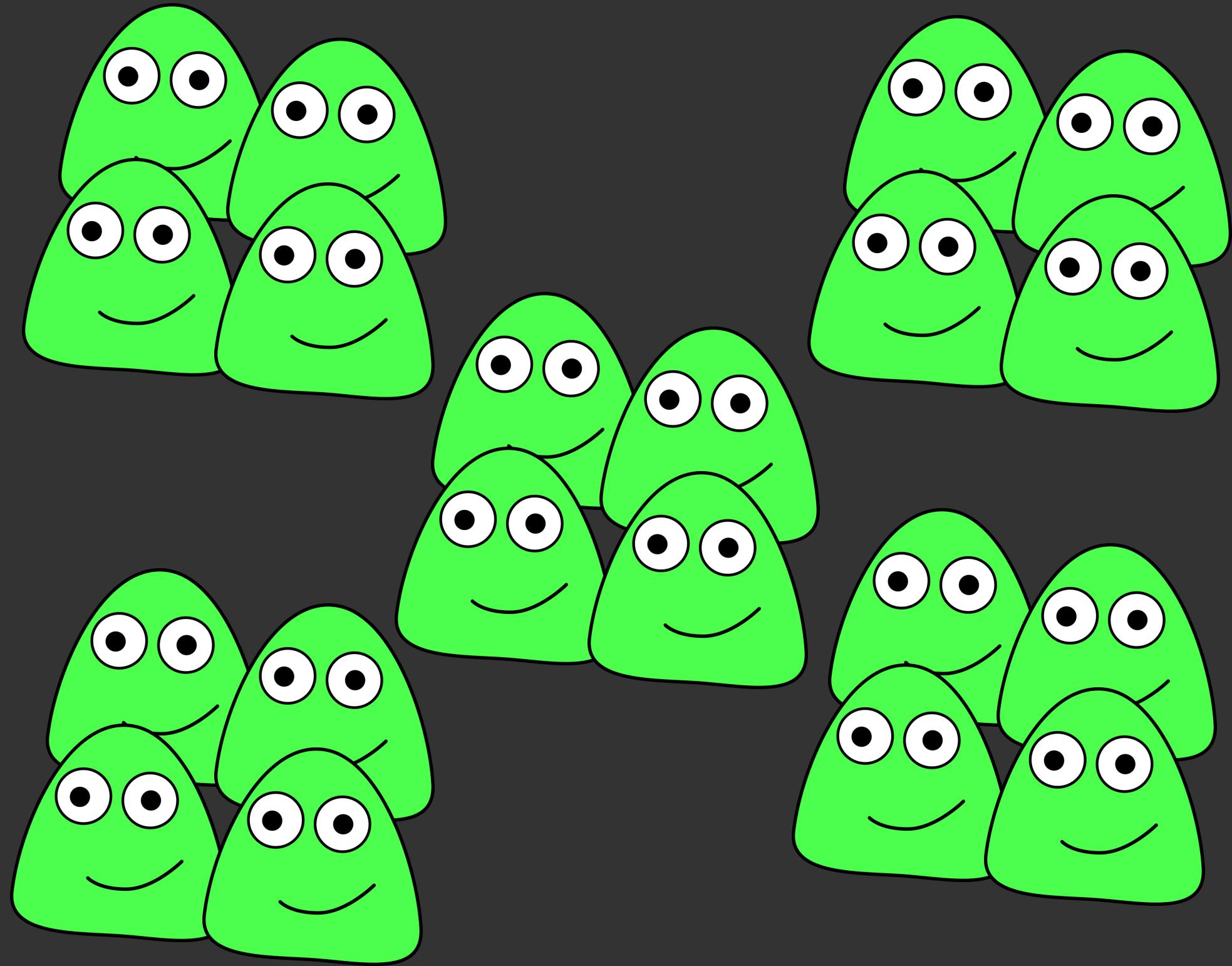


Unser Ansatz

- Kurze Entscheidungswege
- Agile Softwareentwicklung
- Nutzung von Open Source
- Vertrauen und Respekt
- Transparente Kommunikation
- Spaß bei der Arbeit





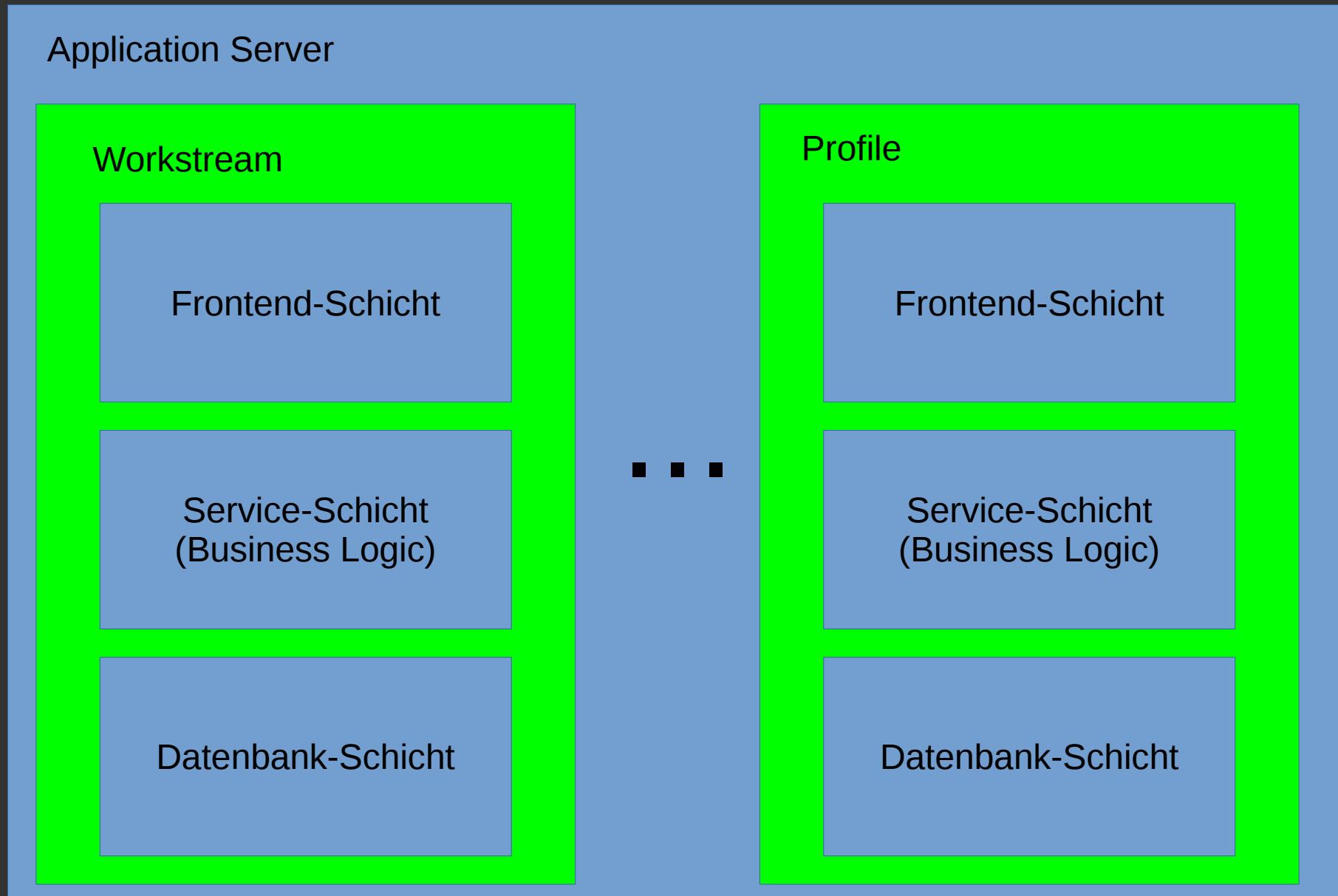


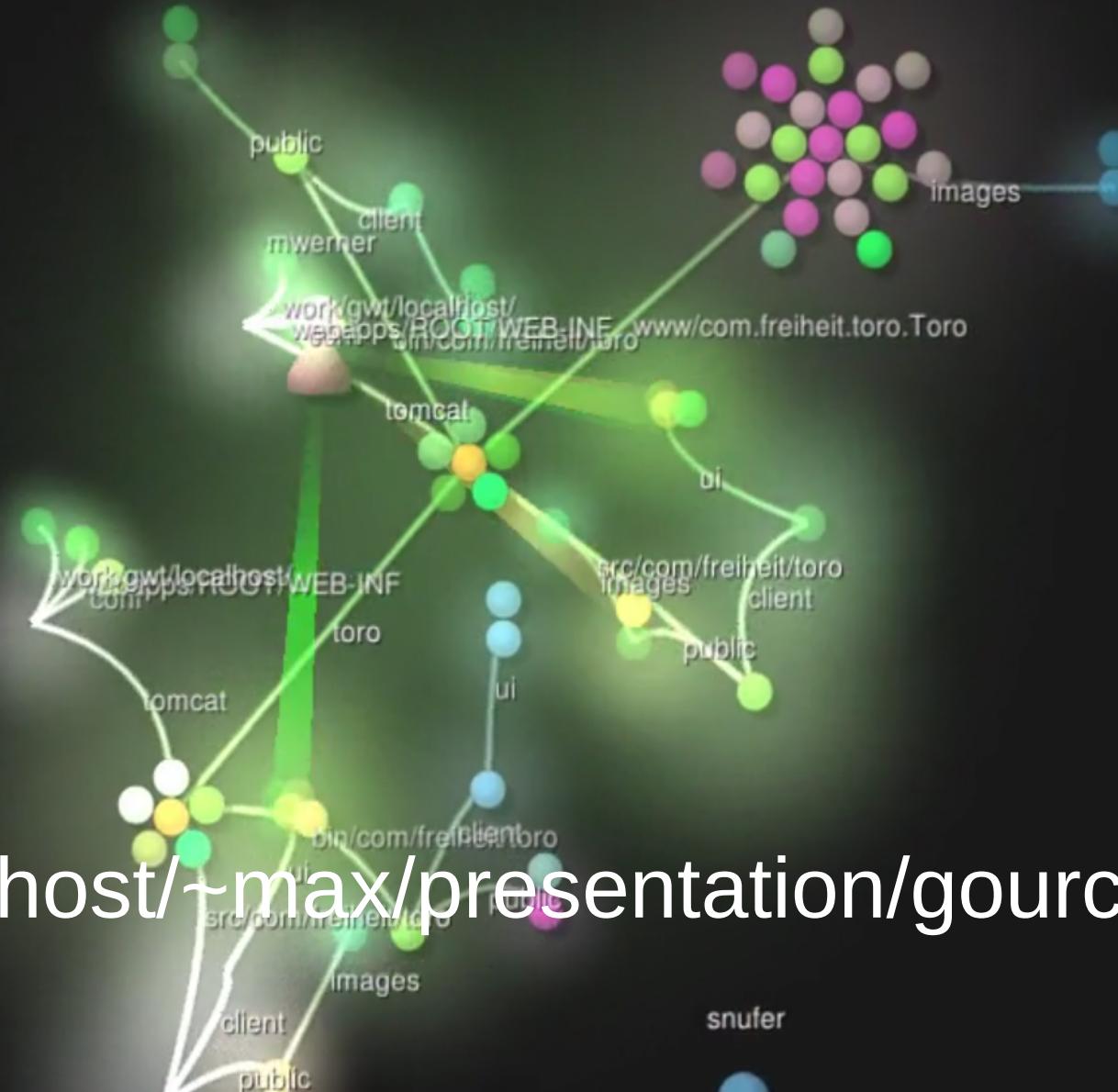


Trotzdem

- Keine individuelle Code-Verantwortung
- Teams können wenig selbst entscheiden
- Langsame Entwicklung
- Einsatz von modernen Tools schwierig
- Skalierbarkeit schwierig

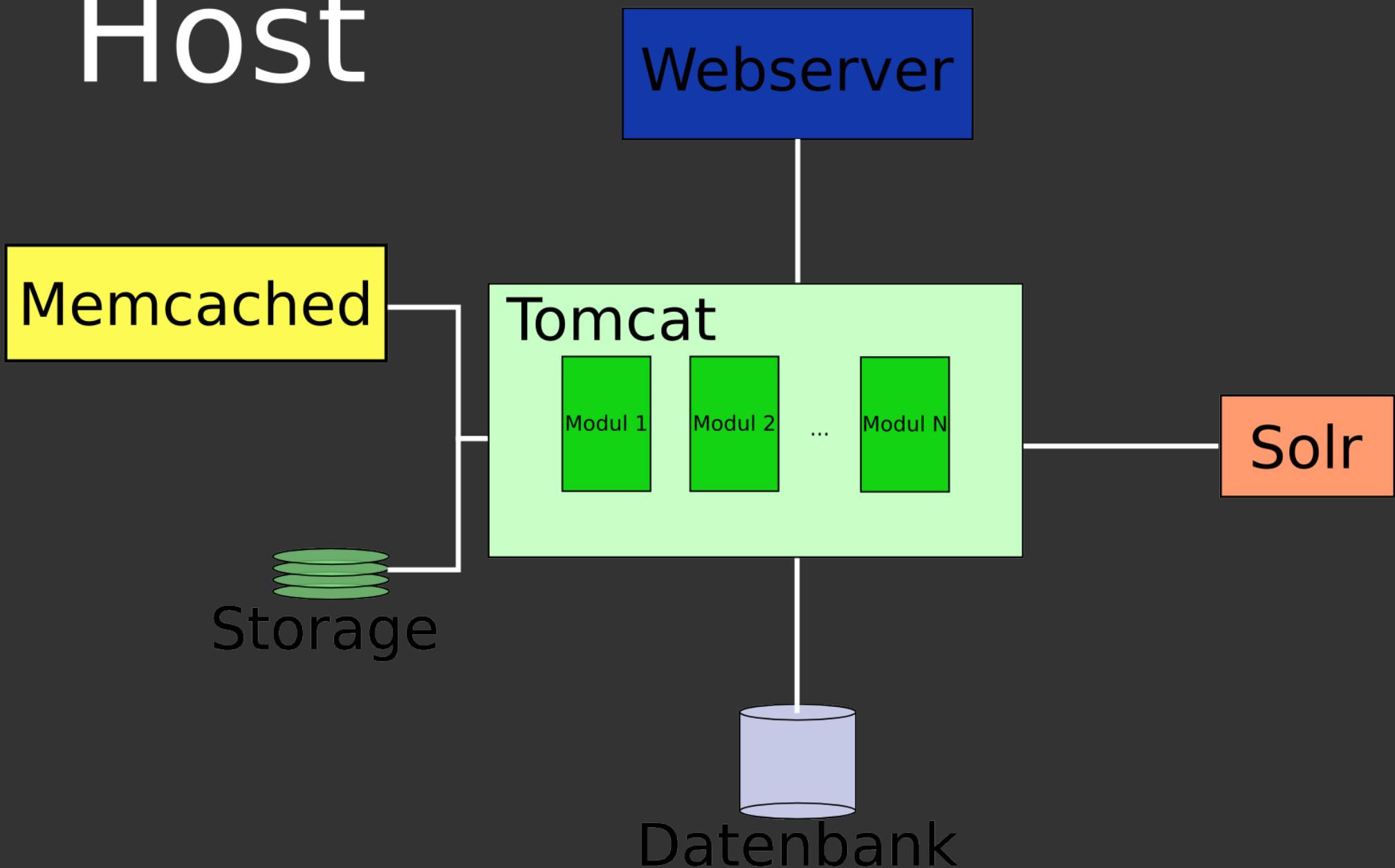
Der Monolith

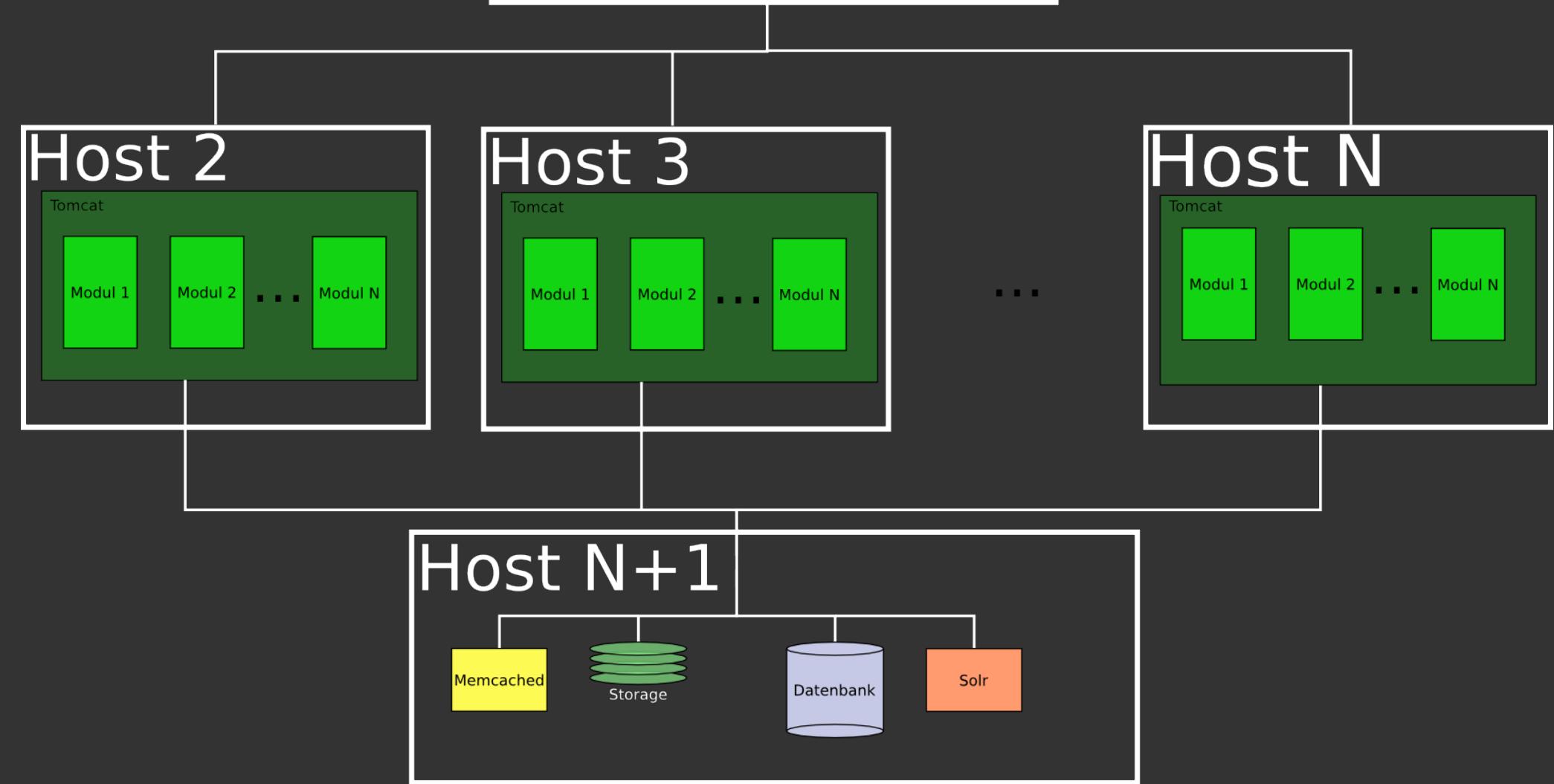
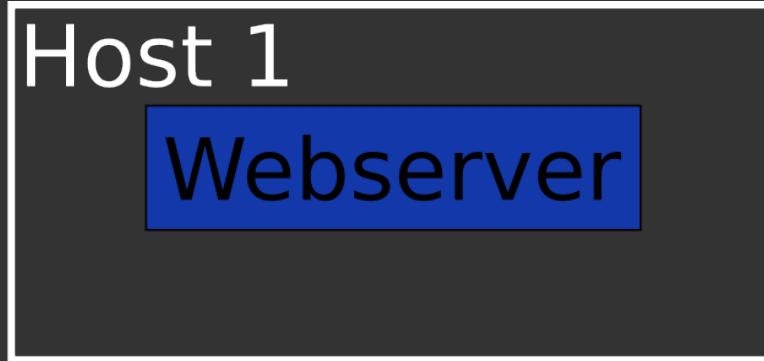




<http://localhost/~max/presentation/gource.html>

Host





Technische Probleme

- Deployments riskant
- Deployments mit hoher Downtime
- Langer Release-Prozess
- Testing aufwändig (App-Context hochfahren etc.)
- Komplexität im Code kaum noch beherrschbar

Technische Probleme

- Continuous Delivery fast unmöglich
- Wie führen wir neue Technologien ein?
 - Programmiersprachen
 - Frameworks
 - ...
- Was passiert, wenn wir wachsen?
- Skalierbarkeit der Organisation schwierig

Conway's Law

“Organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations.” – Melvin E. Conway

Microservices

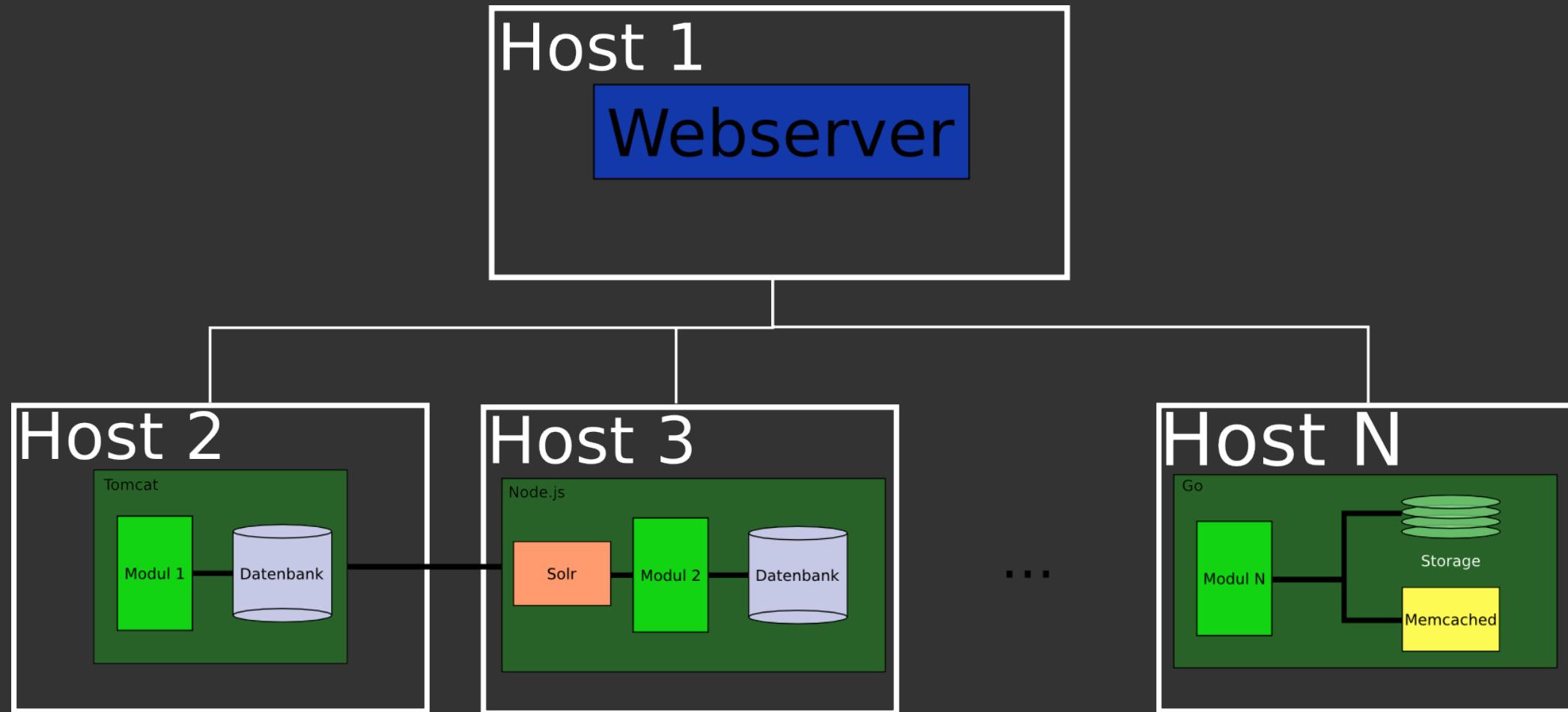
- Was, wenn jeder Service in sich geschlossen wäre?
- Services werden lose gekoppelt
- Kommunikation über definierte APIs
- Vom Monolithen zum verteilten System

Das hatten wir doch schon mal?

- SOA
- CORBA
- Warum hat das nicht funktioniert?

Definition

"In short, the microservice architectural style is an approach to 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. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies." — Martin Fowler



Vorteile

Unterschiedliche Technologien

Java

Node.js
MongoDB

Python

Go

Vorteile

Ausfallsicherheit



Vorteile

Individuelle Skalierung (Cloud)



Vorteile

Individuelles Deployment

v1.2

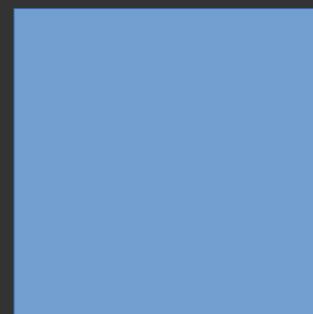
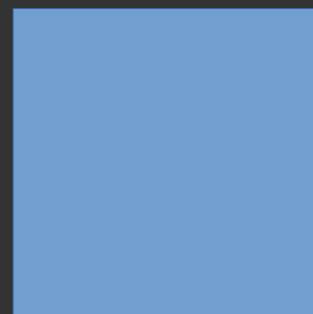
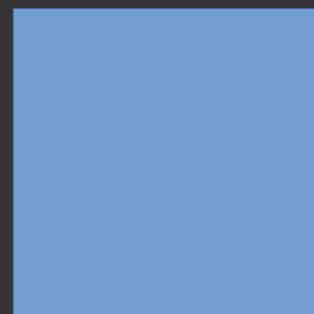
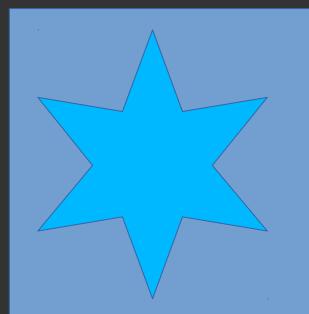
v4.3

v2.7

v12.0

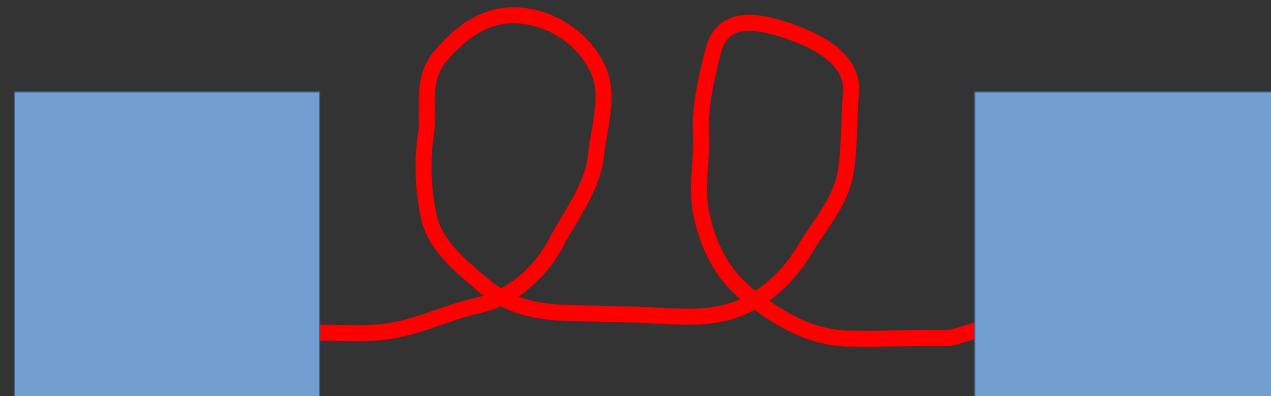
Vorteile

Neue Features



Risiken

Kommunikation ist langsam



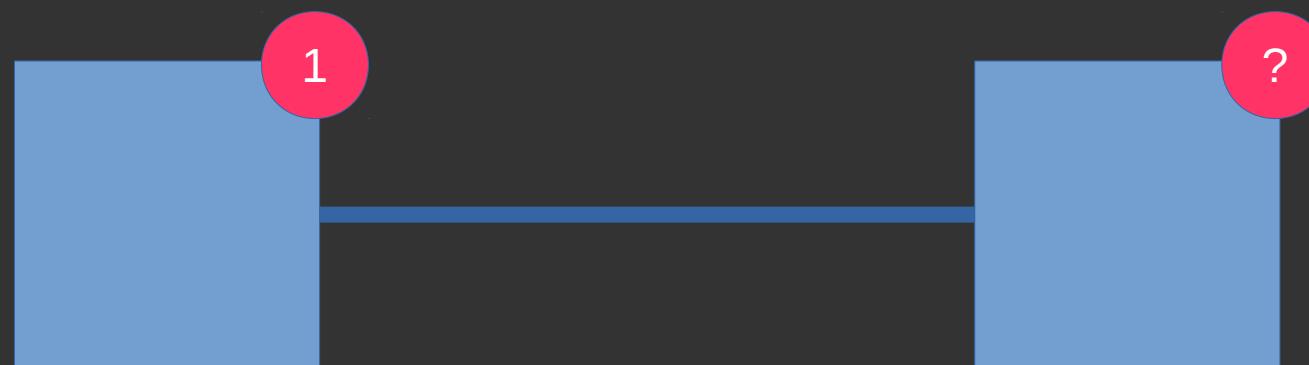
Risiken

Kommunikation ist fehleranfällig



Risiken

Datenhaltung inkonsistent



Risiken

Betriebskomplexität

- Deployment
- Monitoring
- Debugging
- Orchestrierung

Prinzipien

- Redundante Datenhaltung
- Keine synchronen Requests während einer Interaktion
- Möglichst asynchrone Kommunikation
- Lose Kopplung
- Stabile APIs

Technologien

- Inter-server-Kommunikation: G-RPC, Thrift, HTTP/REST
- Datenserialisierung: JSON, XML, ProtoBuf, Hypermedia
- Deployment: Docker

Management

- Autonome Teams
- Vertrauen
- 100% Verantwortung (“you build it, you run it”)
- DevOps-Mentalität

Wann Microservices?

- “There is no silver bullet”
- Start mit Monolith (Time to Market)
- Prepare for Failure
- Lerne auf dem Weg



max@just.social