Einführung in Microservices

Sebastian Heuer

Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg

03.03.2022

Zielsetzung

- Einführung in das Thema Microservices
- Überblick über Herausforderungen
- Nachvollziehen eines Definitionsversuchs
- Wertung

Zielsetzung

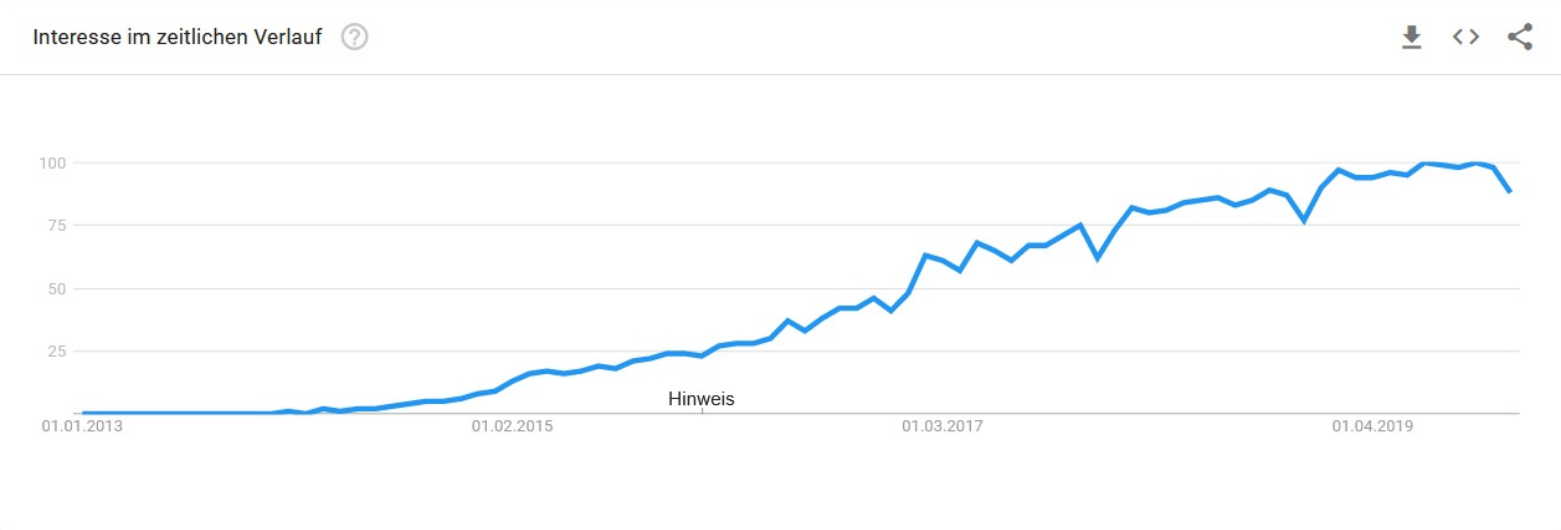
Folgende Punkte werden (fast) nicht behandelt:

- Vor- und Nachteile
- Performance
- Service-Komposition
- Frameworks, Docker und Kubernetes

Geschichte

- 1999 - "resource oriented computing" - Dexter research project, Hewlett Packard Labs - REST als Teil von ROC abstraction
- 2005 - "Software components are Micro-Web-Services" - Dr. Peter Rodgers, Web Services Edge conference
- 2009 - Netflix spaltet den Monolithen und geht in die Cloud
- 2011 - "Microservices" - erste Erwähnung, workshop for software architects
- 2012 - "Microservices" - Namensgebung, workshop for software architects
- 2014 - "Microservices, a definition of this new architectural term" - Lewis & Fowler, Fachartikel für Thoughtworks
- 2015 - "Building Microservices" - Buch von Sam Newman, Nachschlagewerk

Google Trends



Quelle: Google Trends Suche nach "Microservices"

Grundlagen

Service-Oriented Architecture

Softwarekomponenten sind eigene Applikationen in einem verteilten System.

Agile Entwicklung

- Projektmanagementstrategie aus der Industrie
- kurze Entwicklungszyklen, kleine Teams
- enge Zusammenarbeit mit dem Kunden
- Reaktion auf Änderungen anstatt an einem Plan festhalten

Grundlagen

DevOps

- Methodensammlung zum Kombinieren von Entwicklung (Dev) und Management(Ops)
- Effektive Prozesse und Werkzeuge zur Steigerung der Qualität und Geschwindigkeit
- Betrifft: Entwicklung, Qualitätsmanagement, Auslieferung, Wartung, Kommunikation, ...
- Beispiele: Virtualisierung, Automatisierung, Versionsverwaltung, Infrastructure as Code

CI/CD

Continuous Integration, Continuous Deployment
Durchgehende Auslieferung von kleinen Änderungen anstatt Sammeln von Änderungen in größeren Patches

Motivation

Wir haben eine Sammlung von Anforderungen an die Softwareentwicklung

Organisation

- Wir betrachten große, komplexe Anwendungen.
- Wir wollen kleine, autonome Teams.

Prozesse

- Agile Methoden
- CI/CD
- DevOps

Architektur

- Skalierbar
- Ausfallsicher

Was sind Microservices?

- Architekturstil basierend auf SOA
- Applikation wird aufgesplittet in voneinander unabhängig auslieferbare Services
- Hohe Beliebtheit durch Kompatibilität mit modernen DevOps Methoden
- Stil entspringt der Beobachtung von ähnlichen Vorangehensweisen aus der Industrie
- Formale Definition fehlt
- Gemeinsame Merkmale werden in dem Stil zusammengefasst
- Näherungsdefinition durch Vergleiche zu anderen Stilen Insbesondere Unterschiede zum Monolithischen Stil

Gemeinsame Merkmale

Unix Philosophie

Do one thing and do it well

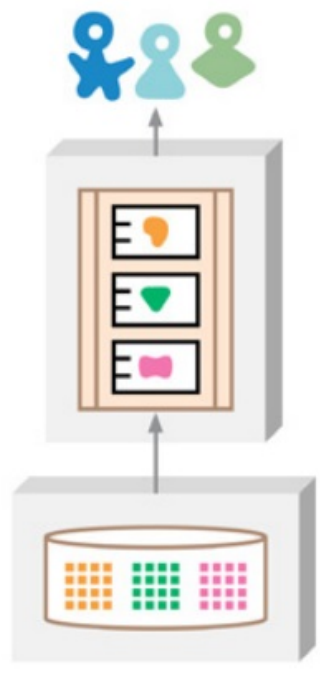
Für jeden Service gilt:

- dezentral
- eigene Codebasis
- unabhängiges Deployment
- lose Kopplung
- verwaltet seine benötigten Daten selbst
- Kommunikation durch Netzwerkprotokolle

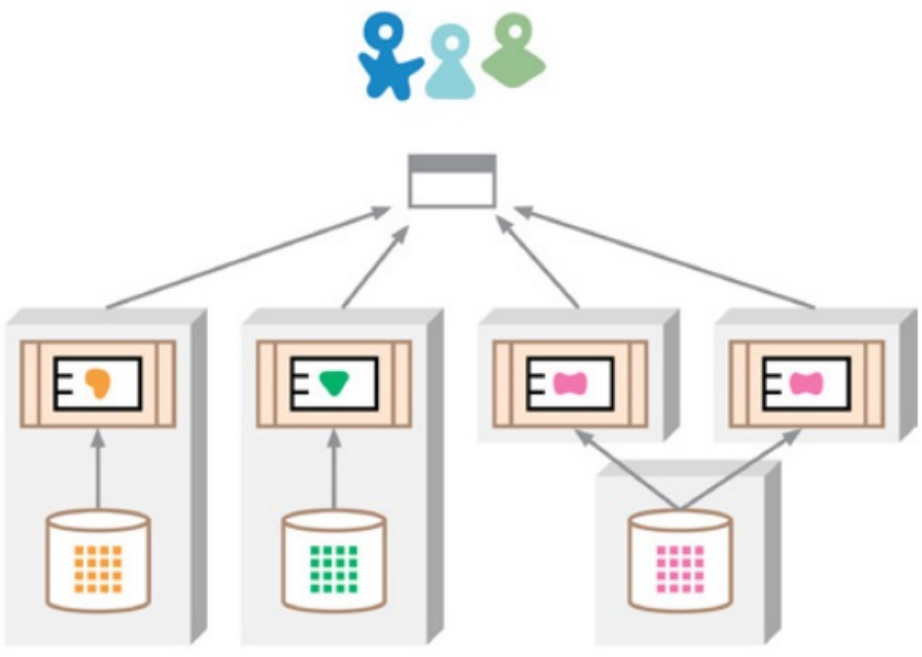
Unterschiede zu Monolithen

- Unterschiedliche Technologien je nach Service
- Fehleranfälligkeit
- Skalierbarkeit
- Erweiterbarkeit
- Flexiblere Organisationsstruktur
- dezentrale Datenverwaltung
- Austauschbarkeit der Komponenten

Monolith vs Microservices



Monolith – eine Datenbank



Microservices – mehrere Anwendungsdatenbanken

Quelle: Fowler & Lewis - Microservices: Nur ein weiteres Konzept in der Softwarearchitektur oder mehr?

Herausforderungen

- Komplexität von verteilten Systemen
- Abhängigikeiten zwischen Services gering halten
- Datenkonsistenz da Services Kopien von benötigten Daten verwalten
- Sicherstellen von Transaktionen auf Geschäftsebene schwierig
- sehr viele bewegliche Teile im System
- Wo zieht man die Grenzen zwischen den (Micro-)Services?

Nachteile

- Code über Servicegrenzen zu Ändern ist sehr teuer
- Services sind Informationsbarrieren
- Verlockung für gemeinsam genutzen Code sehr groß
- Unklare API Spezifizierungen sind ein großes Problem
- Mehr Entscheidungsmöglichkeiten führen zu mehr Fehlern
- Erhöhte Latenz im Vergleich zu In-Prozess Kommunikation

Definition?

- Wichtiger Teil der Definition fehlt noch!
- Wie groß ist ein "Micro"-Service?
- Unterschiedliche Ansichten und Meinungen führen zu technical debt!
- Das Verständnis an welchen Grenzen ein System in Microservices eingeteilt wird ist sehr wichtig

Versuch 1: Komponententrennung durch Services

- Idee: Aufspalten des Monoliths anhand von verwendeten Softwarekomponenten in Services
- Meist hindern nur Dokumentation und Disziplin dass Komponenten nicht enger gekoppelt werden
- Bibliotheken bestehen aus Komponenten die sich in-memory aufrufen
- Änderungen führen dazu dass der gesamte Monolith neu deployed werden muss
- Services sind out-of-process Komponenten
- Wir wollen dass nur einzelne Services neu deployed werden müssen!
- Erste Annäherung. Ein Service kann aus mehreren Komponenten bestehen die zusammen deployed werden

Versuch 2: Strukturierung nach technischen Schichten

- Idee: Aufspalten des Monoliths in technische Schichten als Services
- einzelne Teams für User-Interface-, Backend-, und Datenbankbereiche
- Änderungen an einer Schicht führen meist Änderungen in anderen technischen Schichten nach sich
- Diese Änderungen sind zwangsläufig Service-übergreifend und Team-übergreifend und damit extrem teuer!

Versuch 3: Zerlegen anhand von Geschäftsstrukturen

- Idee: Aufspalten des Monoliths in fachlich getrennte Geschäftsstrukturen
- Implementation des kompletten Technologiestacks für fachlich getrennte Geschätseinheiten in einem Service
- Benutzeroberfläche, Datenspeicherung und Kommunikation für einen Geschätsbereich
- Änderungen der Anforderungen bleiben meist service-intern
- Teams arbeiten funktionsübergreifend und haben Kompetenzen in allen Bereichen

Bewertung Microservices

Microservices:

- Moderner Ansatz zur Lösung von Skalierungs- und Maintenanceproblemen von großen Softwareprojekten
- Man sollte sich dem Aufwand und den Herausforderungen bewusst sein
- Projekte sollten durch Spaltung in diese Architektur überführt werden
- Neuentwicklung mit dieser Architektur kann schwierig und frustrierend sein
- Sind die Hürden überwunden ist die Skalierung und Erweiterung sehr einfach!

Bewertung Quellen

Quellen:

- Sam Newman - "Building Microservices" - Sehr gutes Nachschlagewerk
- Die Veröffentlichungen von Fowler haben maßgeblich zur Popularität von Microservices beigetragen
- Beide Quellen werden of als Standartwerke zitiert.

Zusammenfassung

Microservices

- Beschreibung eines Architekturstils
- Keine formale Definition
- Definitionsversuch über gemeinsame Merkmale
- Aufbrechen von vorhandener Software entlang von geschäftsrelevanten Grenzen

Zusammenfassung

Microservices

- Geringe Kosten für Erweiterung, Skalierung
- Automatisierung von Deployment
- Hoher Einarbeitungsaufwand
- Kenntnisse über Geschäftsprozesse nötig

Danke für Ihre Aufmerksamkeit

Fragen?

Quellen

- Fowler & Lewis - Microservices: Nur ein weiteres Konzept in der Softwarearchitektur oder mehr?
- Sam Newman - Building Microservices

Wann sollte man keine Microservices verwenden?

- Geschäftsprozesse müssen bekannt sein.
  Änderungen in Service-Service Beziehungen sind teuer!
- Bei Neuentwicklungen sind die Abhängigkeiten zwischen Geschäftsprozessen nicht ausreichend bekannt.
  Man sollte mit einem Monolithen starten!
- Probleme die durch Microservices verursacht werden verschlimmern sich mit der Skalierung.
  Implementierung von Microservices sollte graduell passieren, je nach Änderungsvermögen des Systems!

Besondere Services

- Ingress
- API Gateway
- Eventbus
- Dynamic Service Registry
- Management/Orchestration
- Monitoring

Aufbau eines Microservices

- Cluster IP
- Cache
- Security
- Safety