Projektmanagement Aspekte von Microservice-Architekturen

> “Organizations which design systems […] are constrained to produce designs which are copies of the communication structures of these organizations.”

Referent: Sascha Düpre

Agenda

1. Erinnerung: Was ist eine MS-Architektur?
2. Scaling, Teams, Reusability
3. Zwischenstand
4. Wer verwendet MS?

5. Verschiedene technische Aspekte!

Exkurs: Projektmanagement

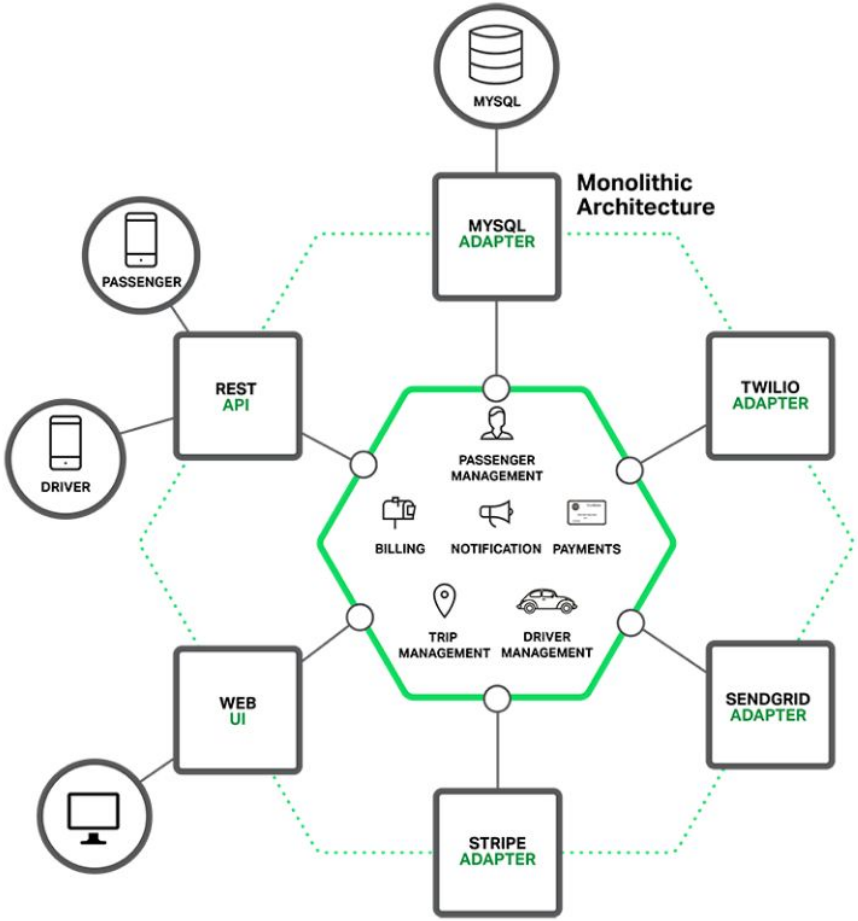
- Phasen: Definition (Kick-Off), Planung, Durchführung und Controlling/Steuerung, Abschluss bzw. Abbruch

- Warum hat die Architektur auswirkungen auf das PM?
- Nicht direkt… Indirekte Auswirkungen: Welche und warum?

- Ziel des PM: Das magische Dreieck im Gleichgewicht halten
  - Zeit, Kosten, Leistung

Monolith-Architektur

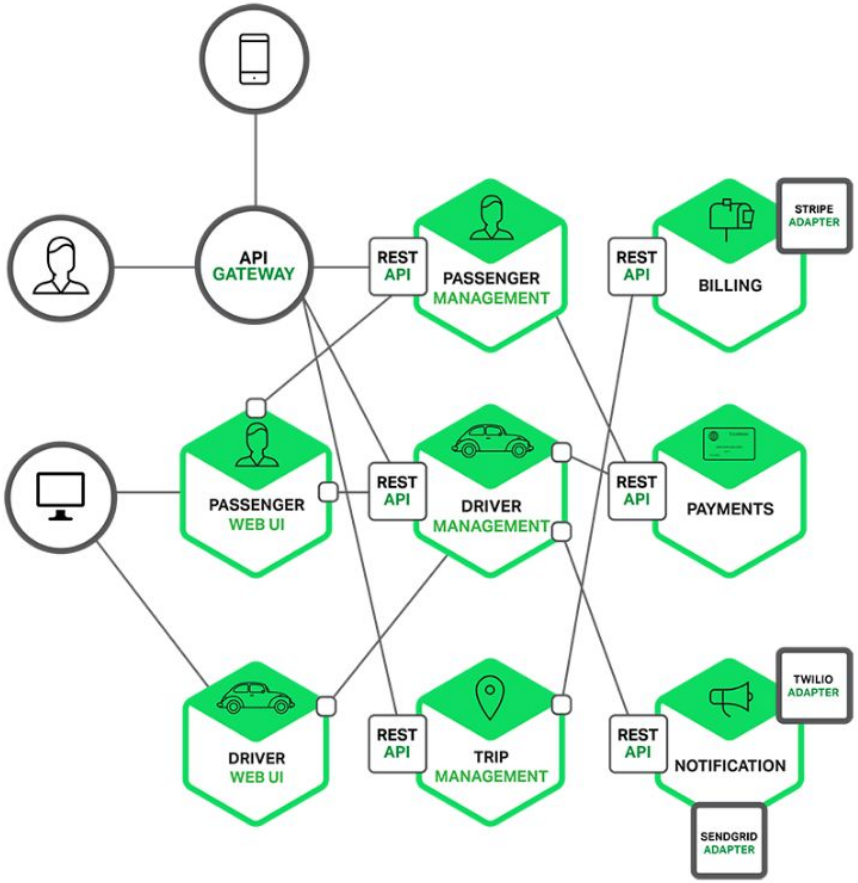
- 1 Server der alle Dienste enthält
- Mehrere Schnittstellen für unterschiedliche Geräte
- 1ne Datenbankschnittstelle



Quelle: https://www.nginx.com/blog/introduction-to-microservices/

Microservice-Architektur

- Mehrere Instanzen pro Service
- Mehrere Datenbanken (nicht ersichtlich)
- Verbunden über wohldefinierte Schnittstellen
- Polyglott presence
- Verteiltes System
  (en: distributed system)



Quelle: https://www.nginx.com/blog/introduction-to-microservices/

Was ist ein Service?

- Seine Aufgabe sollte in einem Satz beschrieben werden können
  - Macht “eine Sache”; dafür gut und präzise
  - Geschäftsprozesse sind Übergreifend eher Anwendungsfälle (en: Use-Cases)
- Lässt sich autonom betreiben

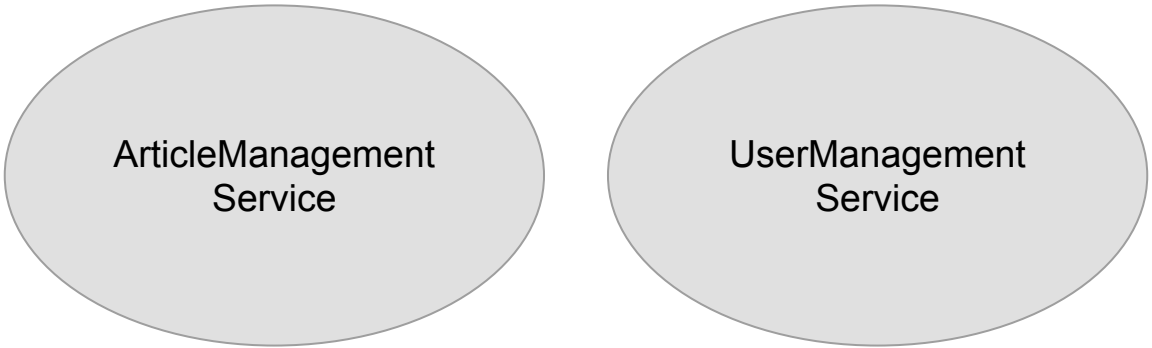
- Eigenschaften [wünschenswert]
  - Responsive -- Antwortbereit
  - Resilience -- Wiederstandsfähigkeit
  - Elastic -- Elastisch
  - (Message-Driven -- Nachrichtenorientiert)

Anwendungsfall -- Shopsystem (1/2)

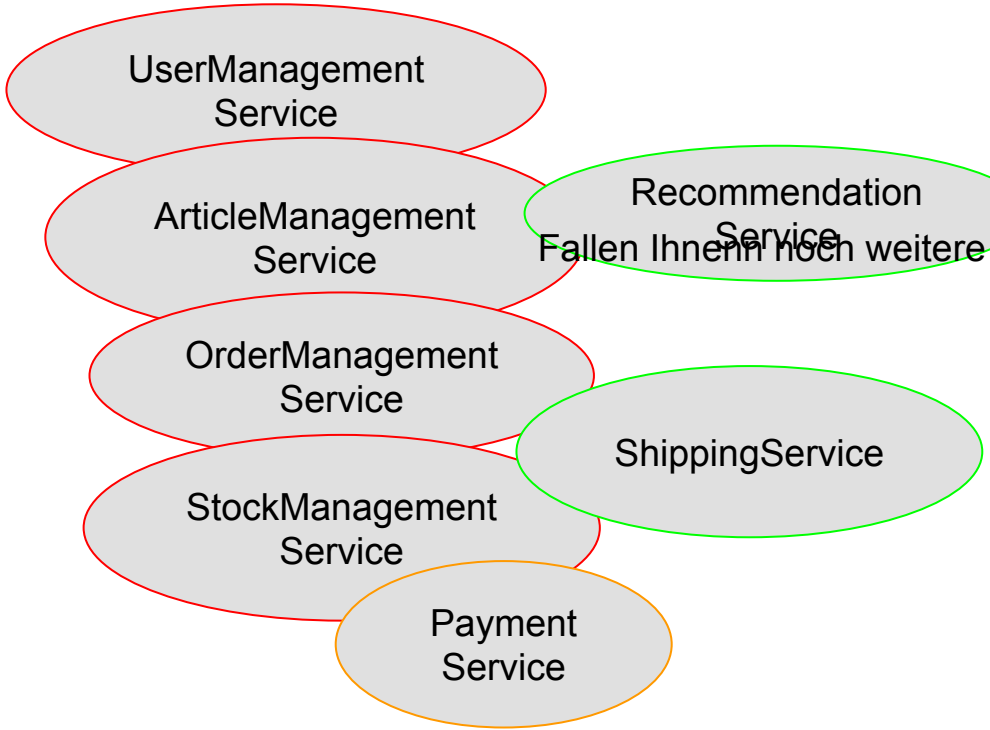
Allgemeines Szenario: Ein *Anwender* wählt ein oder mehrere *Artikel* aus, die er kaufen möchte.

Objekte: *Anwender*, *Artikel* **Aktionen:** auswählen, kaufen

Konkrete Szenarien: *Anwender* wählt *Artikel*; *Anwender* kauft *Artikel*;



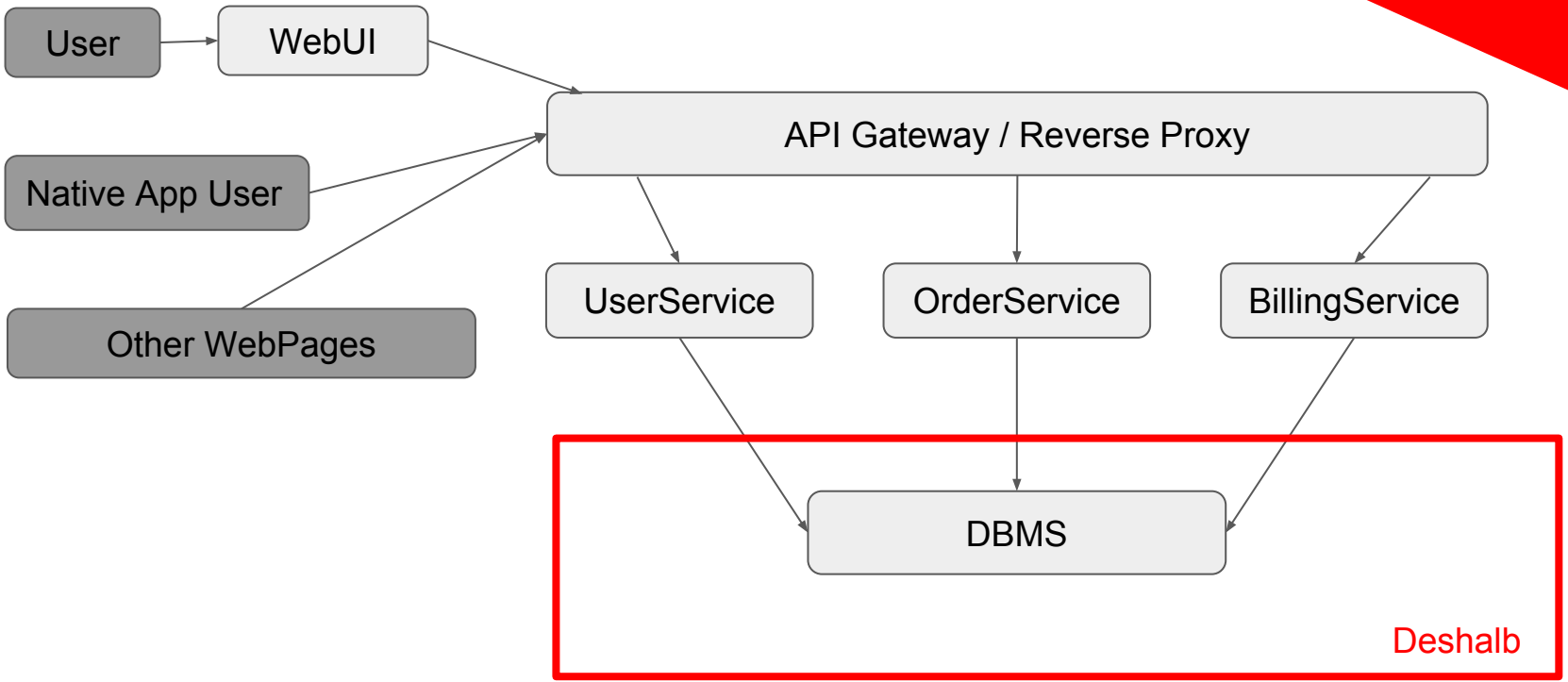
Anwendungsfall -- Shopsystem (2/2)



Fallen Ihnenn noch weitere Services ein?

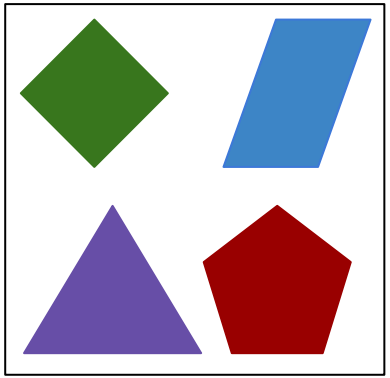
Ist das eine MS-Architektur?

Nein

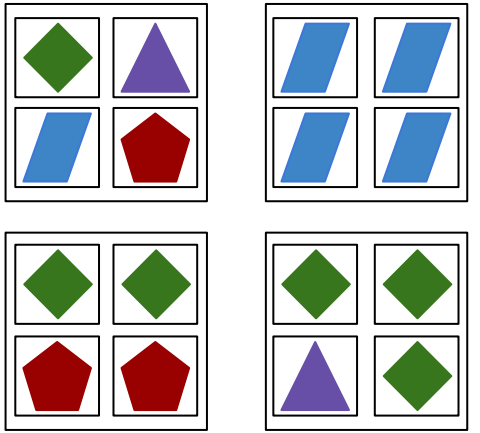


Skalierung - Vertikal vs. Horizontal

1ne leistungsstarke Maschine mit einer Instanz

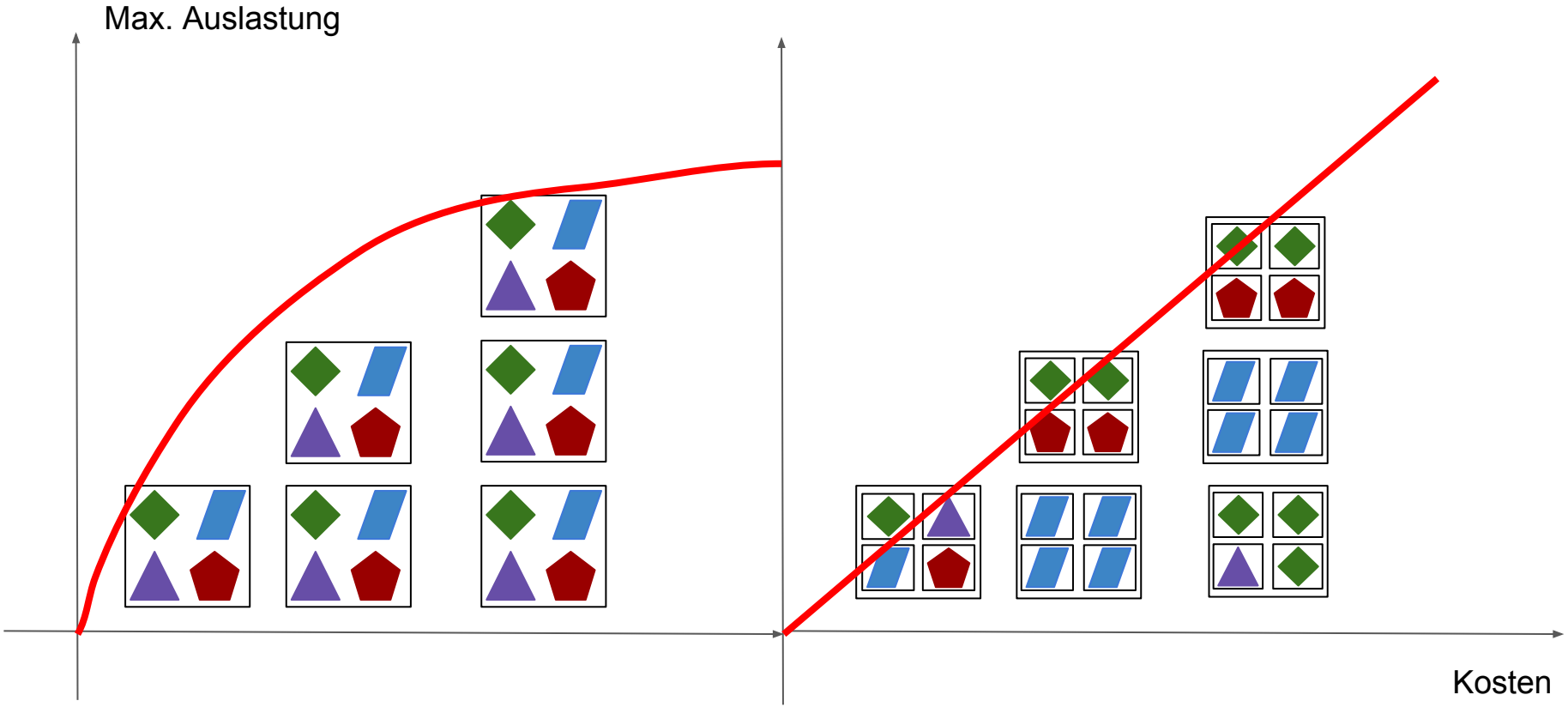


4 Maschinen mit je 4 Services



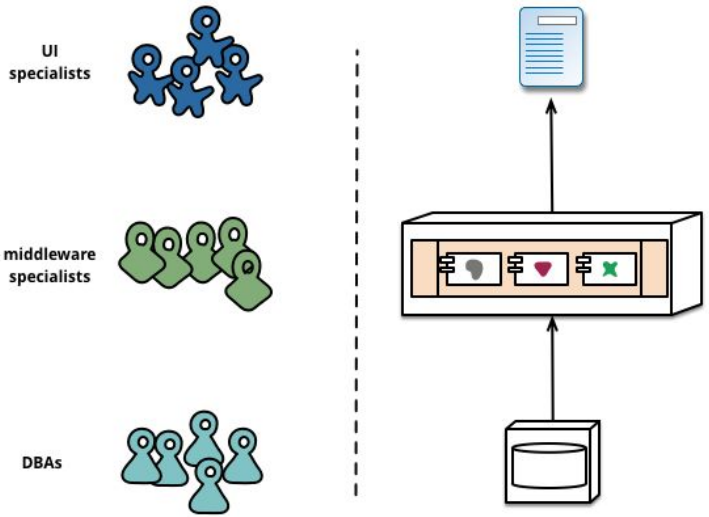
Leistung erhöhen vs. Knotenanzahl erhöhen

Warum ist Skalierung interessant?

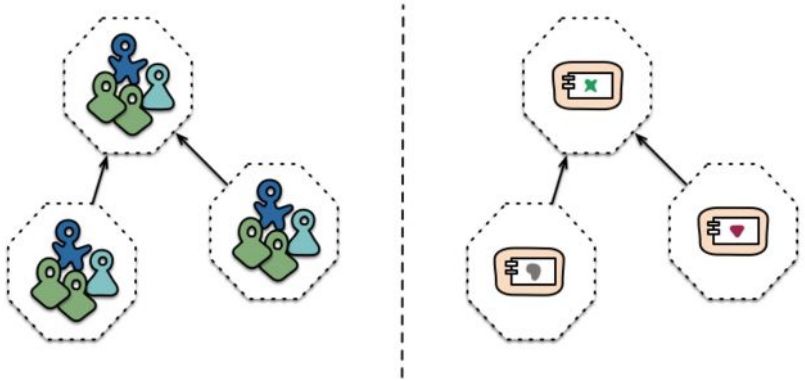


Conways Law

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



vs.



Source: http://martinfowler.com/articles/microservices.html

Teams (1/2)

- Wie groß sollte ein Team pro Service sein?

- Ein paar allgemeine Verantwortungen/benötigte Kompetenzen
  - Testing
  - UI
  - Logic
  - DB(s)
  - Middleware
  - Betrieb

Teams (2/2)

- Daumenregel: < 10
- Jeff Bezos (Amazon): 5-7 max. -- Zwei-Pizza-Regel

- Semantisch:
  - Nicht zu groß, um “überflüssige” Kommunikation zu vermeiden
    - Nicht jedes Projektmitglied interessiert jede Information
  - Nötige Kompetenzen müssen vertreten sein

Bedeutung für das PM

- Agile Teams (erfordern Agile SW-Entwicklung)
- Schnellere (Zwischen)Ergebnisse
- Aufwände lassen sich klar zuordnen (Pro Team/Service)

=> Kunde kann zwischendurch Ergebnisse validieren (ggf. Testen)
-> Fehler/Probleme & Missverständnisse frühzeitig erkennen

=> Teile des Projekt können abgebrochen werden (non-core-services)

Team roles

- Operators are needed
- Developers are needed
- Consider DevOps

- General Problem: From the moment the product runs in production you will need to have a team member for emergency cases available.
  - “I work from 9-5; no interest in working more besides overtime.” - Random Employee
  - “I love my work. I like to work more if it’s interesting.” - Another Random Employee
- People don’t want to work on attendance (“9-5”, “Not interesting”)
- In the US: One may get fired if they won’t do it.
- In Germany: It’s on free will; one needs a “fair” model or have an explicit operator.
- Solution : DevOps (expensive)

Exkurs: Agile Software-Entwicklung

- Es gibt verschiedene Vorgehensmodelle, aber im Kern...
  - Iterativ
  - “Stromlinienförmig”
  - Haben Zeitfenster
  - Hoher Grad an Kollaboration

- Beispiele
  - Scrum
  - Kanban
  - Crystal Light (Oberbegriff für mehrere Agile Methoden)

Wiederverwendbarkeit

- Services können wiederverwendet werden
  - Müssen autonom laufen
- Dadurch sind sie auch austauschbar

- Gilt speziell für Dienste die querschneidende Belange umsetzen. Z.b:
  - Logging
  - Authorization
  - Authentication
  - UserManagement

Im Betrieb bzw. Durchführung

- Continuous Integration und Continuous Delivery
- Cluster Management (Docker Swarm, Kubernetes, VMs, …)
- Load Balancing und andere querschneidende Belange hinsichtlich Infrastruktur bzw. Betrieb

Zwischenstand

- Kostenoptimiertes Skalieren
- Teamkommunikation und -koordination
- Agile Entwicklung
- Wiederverwendbarkeit / Austauschbar

- Längere Time2Market im Gegensatz zur Monolith-Architektur
- Erhöhter Aufwand im Bereich “Betrieb” (en: Operations) (Overhead)
- Ohne Erfahrung: Erhöhtes Risiko
- Microservices sind Komplex (nicht zwingend kompliziert)
- YAGNI - **Y**ou **A**ren’t **G**onna **N**eed **I**t

Und nun?

- Wenn man MS-A benutzen möchte:
  - Haben wir die Kapazitäten (Ressourcen)?
  - Haben wir die organisatorische Komplexität?
  - Benötigen wir die Skalierungsmöglichkeiten?
  - Müssen wir das Produkt schneller liefern und regelmäßige Updates machen?

- Offensichtlich sollte man für jedes Projekt separat entscheiden, ob der Weg über Microservices Sinn ergibt und nicht blind den Vorteilen erliegen.

Wer verwendet Microservices? (1/4)

- Walmart
  - Kauften OneOps und arbeiteten 2 Jahre an einer DevOps API
  - OpenSource: http://oneops.com
- “Four years ago, the Walmart Global eCommerce system was a monolithic system, deployed every 2 months” - walmart.com [01.2016]

- “On a typical day there are now over 1,000 deployments, executed on-demand by development teams, each taking only minutes on average to complete.”

Wer verwendet Microservices? (2/4)

- Netflix

- Viele Beiträge zur OpenSource Community
  - Hystrix
  - Chaos Monkey
  - https://github.com/Netflix/ & https://netflix.github.io/

Wer verwendet Microservices? (3/4)

- Twitter
  - http://twitter.github.io
  - Finagle
  - Eigener MySQL Fork

- Geschichte
  - Beginn: 3-Schichten-Modell, RubyOnRails Monolith
  - Langsamer wechsel zu einem SO-Ansatz
  - Entwicklung von Microservices
  - Stand: Haben vollständig auf MS umgestellt. Für das Front-End RubyOnRails

Wer verwendet Microservices? (4/4)

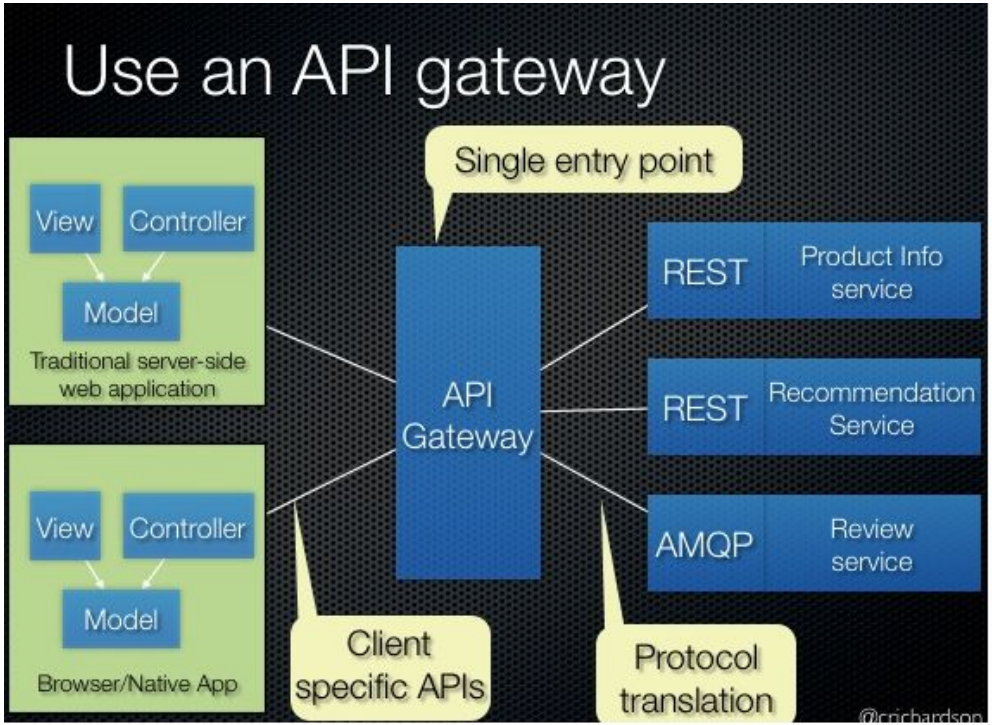
- Viele Mehr
  - Amazon
  - Ebay
  - Google
  - PayPal
  - Soundcloud

- Gemeinsamkeit: Haben alle mit einem Monolithen gestartet

Technische Aspekte

- Gerade die übergreifenden Technologien haben auswirkungen für die Zukunft
  - Es folgen ein paar grundlegende Aspekte die im Zusammenhang mit MS-A relevant sind.
- Allgemeine Architektur-Patterns
- Service Discovery
- Autorisierung/Authentifizierung (via Token)
- Kommunikation

API Gateway Pattern



- Versteckt die innere API
- Kann verschiedene APIs für verschiedenen Clients anbieten
- Vereinfacht den wechsel zwischen API Versionen
- Weniger overhead
- Kann sich um Service Discovery kümmern

Quelle: http://microservices.io/patterns/apigateway.html

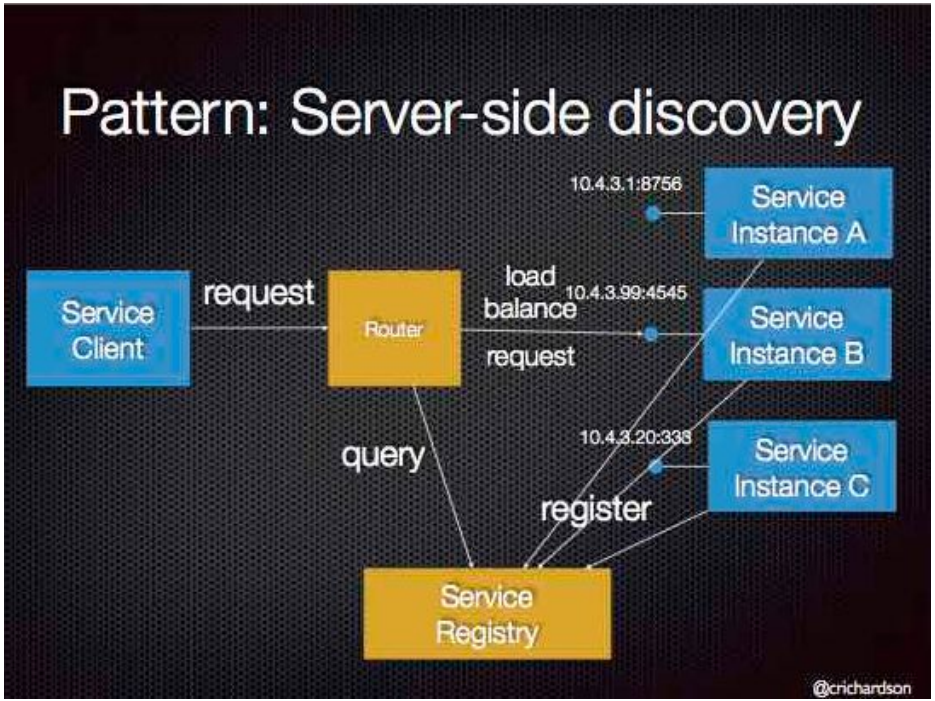
Service-Discovery

- Die laufenden Service-Instanzen werden automatisiert verwaltet
  - ➢ Beliebige Anzahl an laufenden Services, die unterschiedliche IP-Adressen besitzen
- Woher weiß der Client, wohin sein Request gehen muss?

- Server-Side Service-Discovery
- Client-Side Service-Discovery

“Kubernetes is an open-source system for automating deployment, operations, and scaling of containerized applications” - kubernetes.com

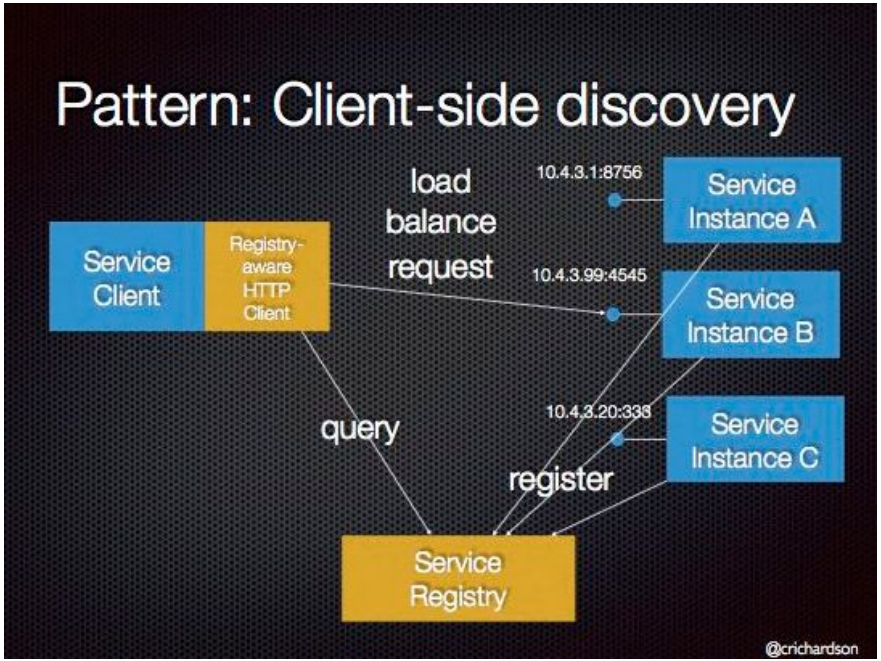
Server-Side Service-Discovery



AWS Elastic Load Balancer

Quelle: http://microservices.io/

Client-Side Service-Discovery

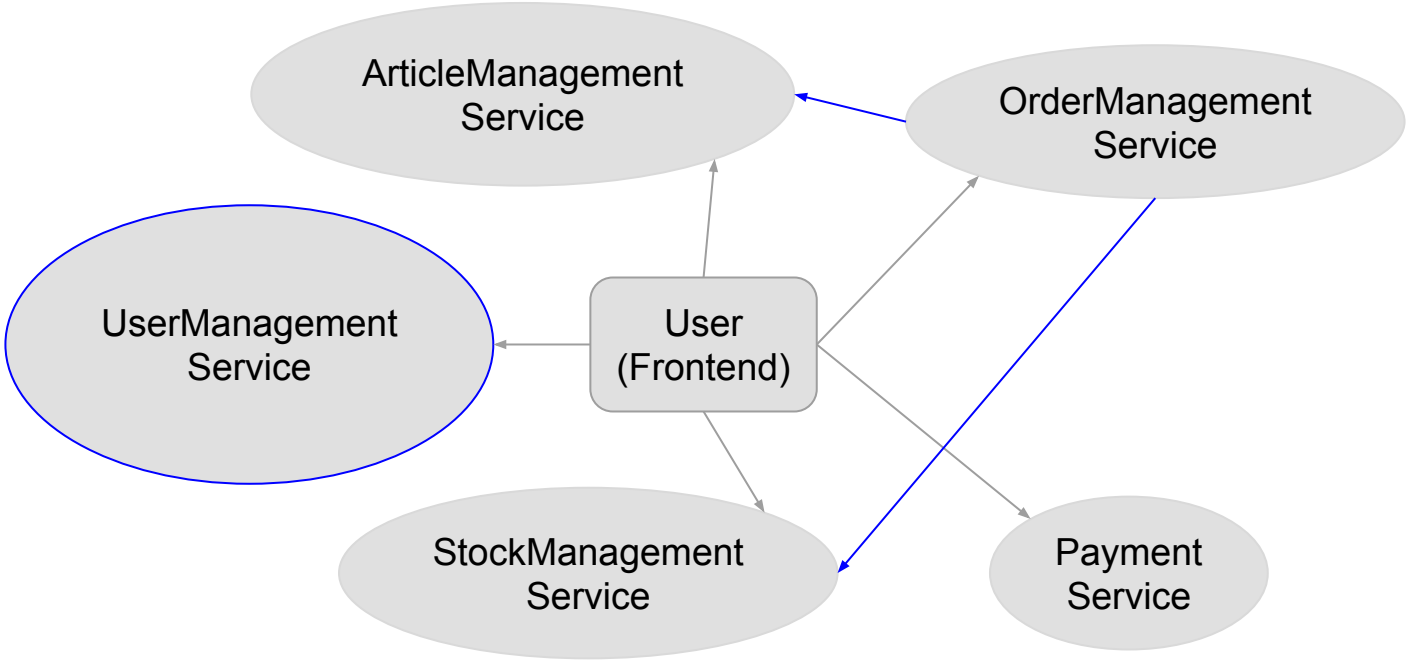


NETFLIX OSS

Eureka - Service Registry

Ribbon client - REST
Client querying Eureka

Kommunikation zwischen Services

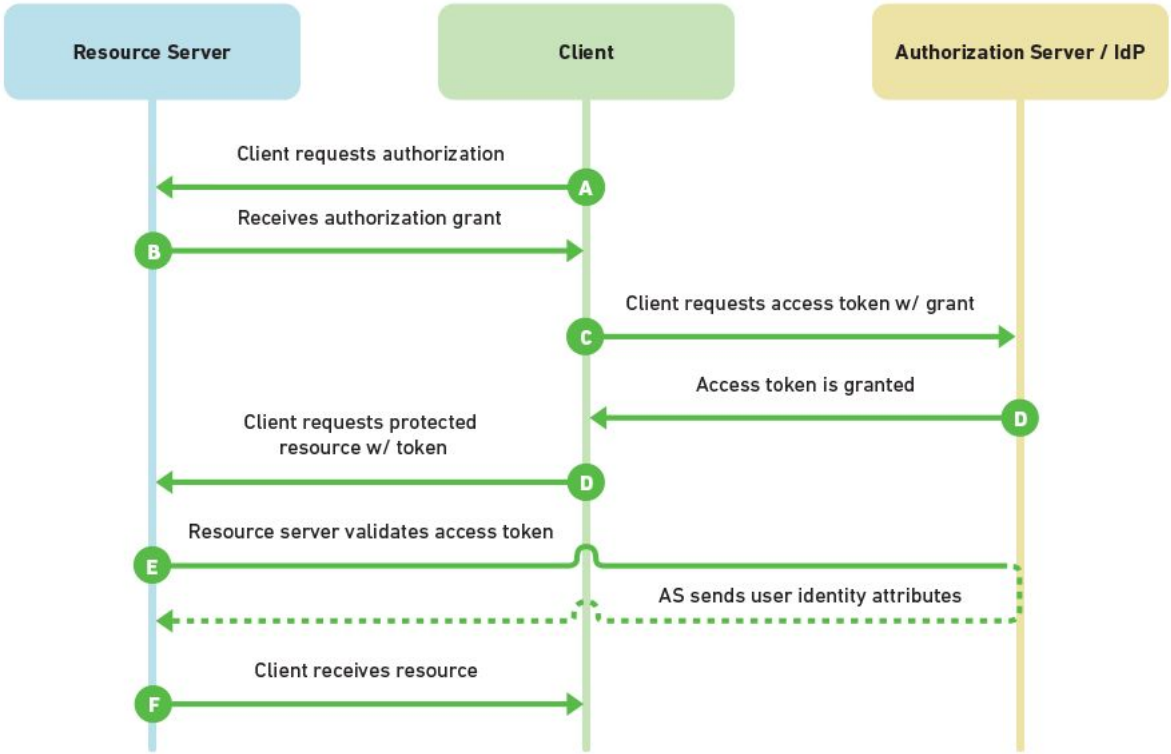


Kommunikation zwischen Services

- Optimalerweise Stateless
- REST
- SOAP (XML-RPC)
- Message Service/Pipes/Streams
- etc.

Autorisierung / Authentifizierung

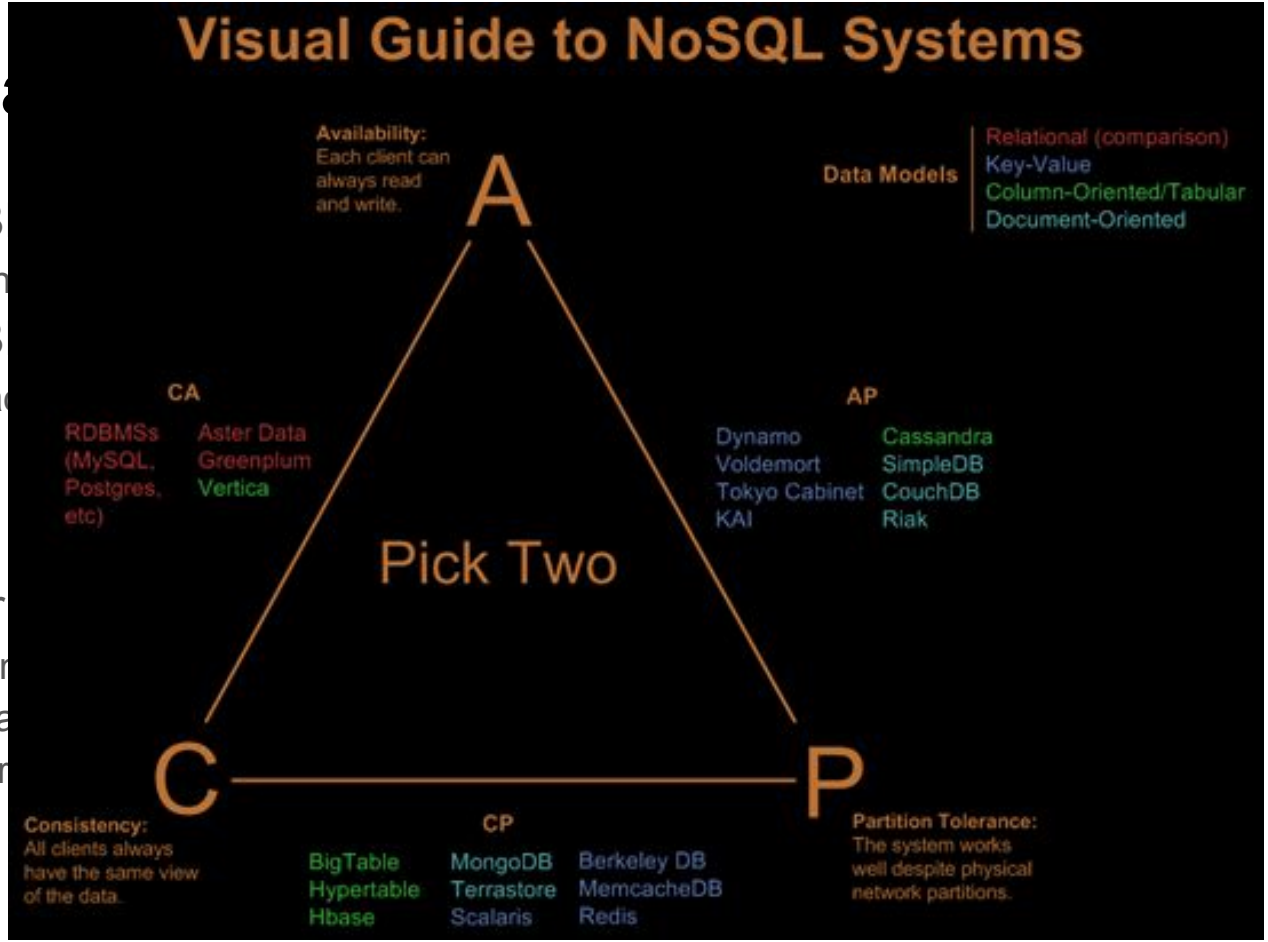
z.B. OAuth via Token



Datenba

- Ein DB
  - Min
- Ein DB
  - Ma
- Kleiner
  - Cor
  - Ava
  - Par

(C,A,P sind



Frameworks und Toolkits

Web-Frameworks/Toolkits

- Vertx.io
  - Java, Javascript, Groovy, Ruby, Ceylon
- Play Framework
  - Java, Scala
- WildFly
  - Java (JEE 7 Compliant)

(REST)-Frameworks

- Finagle (Twitter)
- Spray.io / Akka.io
- Express JS (NodeJS)

Quellen

- http://martinfowler.com/articles/microservices.html
- https://www.nginx.com/blog/introduction-to-microservices/
- http://microservices.io/patterns/monolithic.html
- http://microservices.io/patterns/microservices.html
- http://microservices.io/patterns/microservice-chassis.html
- http://thenewstack.io/agile-management-how-to-manage-microservices-with-your-team/
- http://www.baselinemag.com/enterprise-apps/walmart-embraces-microservices-to-get-more-agile.html
- http://techblog.netflix.com/
- http://www.artima.com/scalazine/articles/twitter_on_scala.html

Vielen Dank für Ihre Aufmerksamkeit.

Projektmanagement
Aspekte von
Microservice-Architekturen

Referent: Sascha Düpre