Einleitung

Microservices sind einer der wichtigsten Software-Architektur-Trends, grundlegende Werke über Microservices gibt es schon. Unter anderem auch das Microservices-Buch (*http://microservices-buch.de*)[1] vom Autor dieses Werks. Warum noch ein weiteres Buch über Microservices?

Es ist eine Sache, eine Architektur zu definieren. Sie umzusetzen, ist eine ganz andere Sache. Dieses Buch stellt technologische Ansätze für die Umsetzung von Microservices vor und zeigt die jeweiligen Vor- und Nachteile.

Dabei geht es um Technologien für ein Microservices-System als Ganzes. Jeder Microservice kann anders implementiert werden. Daher sind die technologischen Entscheidungen für die Frameworks innerhalb der Microservices nicht so wichtig wie die Entscheidungen für das gesamte System. Die Entscheidung für ein Framework kann in jedem Microservice revidiert werden. Technologien für das Gesamtsystem sind kaum änderbar.

Grundlagen

Um Microservices zu verstehen, ist eine Einführung in die Architektur, ihre Vor- und Nachteile und Spielweisen unerlässlich. Die Grundlagen sind in dem Buch soweit erläutert, wie sie für das Verständnis der praktischen Umsetzungen notwendig sind.

Konzepte

Microservices benötigen Lösungen für verschiedene Herausforderungen. Dazu zählen Konzepte zur Integration (*Frontend-Integration, synchrone und asynchrone* Microservices) und zum Betrieb (*Monitoring, Log-Analyse, Tracing*). Microservices-Plattformen wie *PaaS* oder *Kubernetes* stellen vollständige Lösungen für den Betrieb von Microservices dar.

1. Eberhard Wolff: Microservices: Grundlagen flexibler Softwarearchitekturen, dpunkt.verlag, 2015, ISBN 978-3-86490-313-7

Rezepte

Das Buch nutzt Rezepte als Metapher für die Technologien, mit denen die Konzepte umgesetzt werden können. Jeder Ansatz hat viel mit einem Rezept gemeinsam:

- Jedes Rezept ist *praktisch* beschrieben, einschließlich einer technischen Implementierung als Beispiel. Bei den Beispielen liegt der Fokus auf *Einfachheit*. Jedes Beispiel kann leicht nachvollzogen, erweitert und modifiziert werden.
- Das Buch bietet dem Leser eine *Vielzahl von Rezepten*. Der Leser muss aus diesen Rezepten für sein Projekt *eine Auswahl* treffen, so wie ein Koch es für sein Menü tut. Das Buch zeigt verschiedene Optionen. In der Praxis muss fast jedes Projekt anders angegangen werden. Dazu bieten die Rezepte die Basis.
- Zu jedem Rezept gibt es *Rezept-Variationen*. Schließlich kann ein Rezept auf viele verschiedene Arten und Weisen umgesetzt werden. Das gilt auch für die Technologien in diesem Buch. Manchmal sind die Variationen so einfach, dass sie direkt als *Experiment* in dem ablauffähigen Beispiel umgesetzt werden können.

Für jedes Rezept gibt es ein *ablauffähiges Beispiel* mit der konkreten Technologie. Die Beispiele sind einzeln ablauffähig und bauen nicht aufeinander auf. So kann der Leser sich mit den für ihn interessantesten Rezepten und Beispielen beschäftigen, ohne sich dabei mit anderen Beispielen befassen zu müssen.

So liefert das Buch einen *Einstieg*, um einen *Überblick* über die Technologien zu bekommen und einen Technologie-Stack auszuwählen. Danach kann der Leser sich anhand der im Buch enthaltenen Links weiter in die relevanten Technologien vertiefen.

Aufbau des Buchs

Dieses Buch besteht aus drei Teilen.

Teil I – Architektur-Grundlagen

Teil I gibt eine Einführung in die Architektur-Grundlagen, die mit Kapitel 1 beginnt.

- Kapitel 1 klärt den Begriff »Microservice« und Kapitel 3 erläutert Self-contained Systems als besonders praktikablen Ansatz für Microservices.
- In einem Microservices-System gibt es die Ebenen der Mikro- und Makro-Architektur, die globale und lokale Entscheidungen darstellen (Kapitel 2).
- Oft sollen alte Systeme in Microservices migriert werden (Kapitel 4).

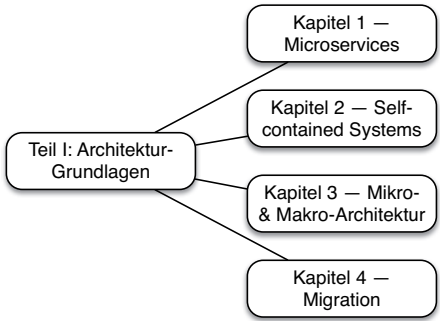

Abb. 1 *Überblick über Teil I*

Teil II – Technologie-Stacks

Technologie-Stacks stehen im Mittelpunkt von Teil II, der mit Kapitel 5 beginnt.

- *Docker* ist die Basis vieler Microservices-Architekturen (Kapitel 5). Es erleichtert das Ausrollen von Software und den Betrieb der Services.
- Die *technische Mikro-Architektur* (Kapitel 6) beschreibt Technologien, die zur Implementierung eines Microservice genutzt werden können.
- Eine Möglichkeit zur Integration ist das Konzept zur *Integration am Web-Frontend* (Kapitel 7). Die Frontend-Integration führt zu einer losen Kopplung der Microservices und einer hohen Flexibilität.
  - Das Rezept aus Kapitel 8 setzt für die Web-Frontend-Integration auf *Links* und auf *JavaScript* für das dynamische Nachladen von Inhalten. Dieser Ansatz ist einfach realisierbar und nutzt gängige Web-Technologien.
  - Auf dem Server kann die Integration mit *ESI (Edge Side Includes)* erfolgen (Kapitel 9). ESI ist in Caches implementiert, sodass das System eine höhere Performance und Zuverlässigkeit erreichen kann.
- Das Konzept der *asynchronen Kommunikation* steht im Mittelpunkt von Kapitel 10. Asynchrone Kommunikation verbessert die Zuverlässigkeit und entkoppelt das Systems.
  - Eine asynchrone Technologie ist *Apache Kafka* (Kapitel 11), mit der Messages verschickt werden können. Kafka speichert die Nachrichten dauerhaft ab und erlaubt so die Rekonstruktion des Zustands eines Microservices aus den Nachrichten.
  - Die Alternative für asynchrone Kommunikation ist *Atom* (Kapitel 12), ein HTTP- und REST-basiertes Protokoll. Atom nutzt eine REST-Infrastruktur und kann daher sehr einfach umgesetzt werden.
- Kapitel 13 stellt vor, wie das Konzept *synchroner Microservices* umgesetzt werden kann. Die synchrone Kommunikation zwischen Microservices wird in der Praxis sehr häufig genutzt, obwohl dieser Ansatz bei Antwortzeiten und Zuverlässigkeit Herausforderungen bereithalten kann.

- Der *Netflix-Stack* (Kapitel 14) bietet Eureka für Service Discovery, Ribbon für Load Balancing, Hystrix für Resilience und Zuul für Routing. Netflix wird vor allem in der Java Community breit genutzt.
- *Consul* (Kapitel 15) ist eine Alternative für Service Discovery. Consul hat sehr viele Features und kann mit einer breiten Palette an Technologien genutzt werden.

Kapitel 16 erläutert das Konzept der *Microservices-Plattformen*, die den Betrieb und die Kommunikation der Microservices unterstützen.

- *Kubernetes* (Kapitel 17) ist eine Microservices-Plattform, die Docker Container ausführen kann, aber auch Service Discovery und Load Balancing löst. Der Microservice bleibt von dieser Infrastruktur unabhängig.
- *PaaS (Platform as a Service)* ist eine weitere Microservices-Plattform (Kapitel 18), die am Beispiel Cloud Foundry erläutert wird. Cloud Foundry ist sehr flexibel und kann auch im eigenen Rechenzentrum betrieben werden.

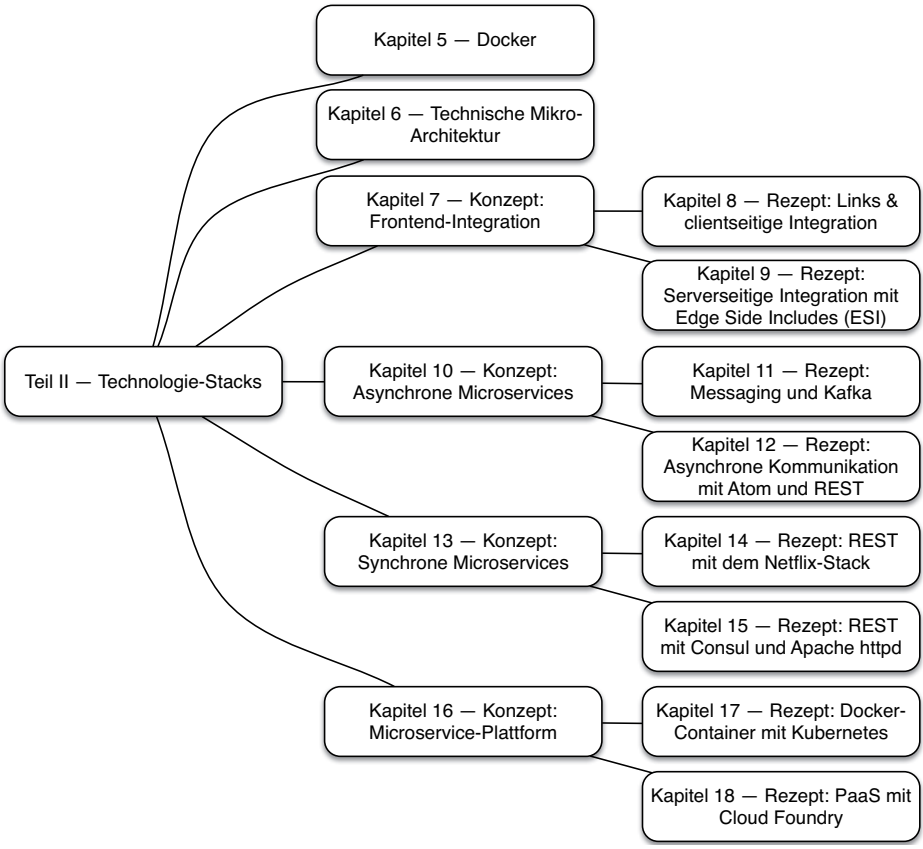

Abb. 2 *Überblick über Teil II*

Teil III – Betrieb

Den *Betrieb* einer Vielzahl von Microservices sicherzustellen, ist eine große Herausforderung. Teil III (ab Kapitel 19) diskutiert mögliche Rezepte zur Lösung.

- Das Kapitel 19 erläutert *Grundlagen*, und warum der Betrieb von Microservices so schwierig ist.
- Im Kapitel 20 geht es um *Monitoring* und das Werkzeug Prometheus. Prometheus unterstützt multidimensionale Datenstrukturen und kann die Monitoring-Werte auch von vielen Microservice-Instanzen analysieren.
- Die *Analyse von Log-Daten* steht im Mittelpunkt von Kapitel 21. Als Werkzeug zeigt das Kapitel den Elastic Stack. Dieser Stack ist sehr weit verbreitet und stellt eine gute Basis für die Analyse auch großer Log-Datenmengen dar.
- *Tracing* verfolgt Aufrufe zwischen Microservices (Kapitel 22). Dazu kommt Zipkin zum Einsatz. Zipkin unterstützt verschiedene Plattformen und stellt einen De-facto-Standard für Tracing dar.

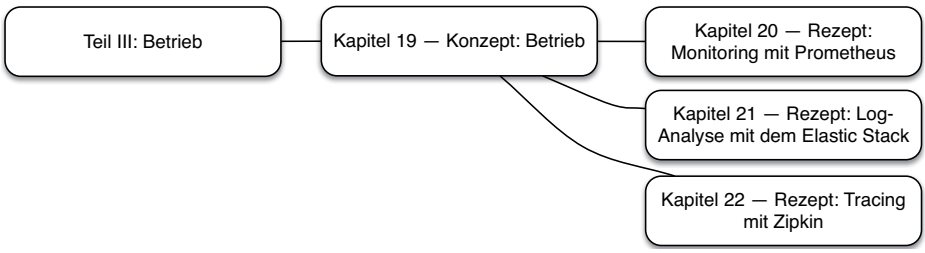

Abb. 3 *Überblick über Teil III*

Abschluss und Anhänge

Abschließend bietet das Kapitel 23 noch einen *Ausblick*.

Die Anhänge erklären die Installation der Software (Anhang A), die Benutzung des Build-Werkzeugs Maven (Anhang B) sowie Docker und Docker Compose (Anhang C), mit denen die Umgebungen für die Beispiele betrieben werden.

Zielgruppe

Das Buch erläutert Grundlagen und technische Aspekte von Microservices. Es ist für verschiedene Zielgruppen interessant:

- *Entwicklern* bietet Teil II eine Hilfe bei der Auswahl eines geeigneten Technologie-Stacks. Die Beispielprojekte dienen als Basis für das Einarbeiten in die Technologien. Die Microservices in den Beispielprojekten sind in Java mit dem Spring-Framework geschrieben, aber die Technologien in den Beispielen dienen zur Integration von Microservices, sodass weitere Microservices in anderen Sprachen ergänzt werden können. Teil III rundet das Buch in Rich-

tung Betrieb ab, der für Entwickler immer wichtiger wird, und Teil I erläutert die grundlegenden Architektur-Konzepte.

- *Architekten* vermittelt Teil I das grundlegende Wissen über Microservices. Teil II und Teil III zeigen praktische Rezepte und Technologien, um die Architekturen umzusetzen. Damit geht das Buch weiter als ein reines Microservices-Architektur-Buch.
- Für Experten aus den Bereichen *DevOps* und *Betrieb* stellen die Rezepte in Teil III eine Basis für eine Technologie-Bewertung von Betriebsaspekten wie Log-Analyse, Monitoring und Tracing von Microservices dar. Teil II zeigt Technologien für Deployment wie Docker, Kubernetes oder Cloud Foundry. Teil I beschreibt als Hintergrund die Konzepte hinter dem Microservices-Architektur-Ansatz.
- *Manager* bekommen in Teil I einen Überblick über die Vorteile des Architektur-Ansatzes und die besonderen Herausforderungen. Sie können bei Interesse an technischen Details Teil II und Teil III lesen.

Vorwissen

Das Buch setzt grundlegendes Wissen über Software-Architektur und Software-Entwicklung voraus. Die praktischen Beispiele sind so dokumentiert, dass sie mit wenig Vorwissen ausgeführt werden können. Das Buch fokussiert auf Technologien, die für Microservices in verschiedenen Programmiersprachen genutzt werden können. Die Beispiele sind in Java mit den Frameworks Spring Boot und Spring Cloud geschrieben, sodass für Änderungen an dem Code Java-Kentnisse notwendig sind.

Quick Start

Das Buch vermittelt vor allem Technologien. Zu jeder Technologie in jedem Kapitel gibt es ein Beispiel. Um schnell praktische Erfahrungen mit den Technologien zu sammeln und anhand der Beispiele nachzuvollziehen, gibt es einen Quick Start:

- Zunächst muss auf dem Rechner die notwendige Software *installiert* sein. Die Installation beschreibt Anhang A.
- Der Build der Beispiele erfolgt mit *Maven*. Den Umgang mit Maven erläutert Anhang B.
- Die Beispiele setzen alle auf *Docker* und *Docker Compose* auf. Das Anhang C beschreibt die wichtigsten Befehle für Docker und Docker Compose.

Sowohl für den Build mit Maven als auch für Docker und Docker Compose enthalten die Kapitel Anleitungen zum Troubeshooting.

Die Beispiele sind in folgenden Abschnitten erläutert:

Konzept	Rezept	Abschnitt
Frontend-Integration	Links & clientseitige Integration	8.2
Frontend-Integration	Edge Side Includes (ESI)	9.2
Asynchrone Microservices	Kafka	11.4
Asynchrone Microservices	REST & Atom	12.2
Synchrone Microservices	Netflix-Stack	14.1
Synchrone Microservices	Consul & Apache httpd	15.1
Microservices-Plattform	Kubernetes	17.3
Microservices-Plattform	Cloud Foundry	18.3
Betrieb	Monitoring mit Prometheus	20.4
Betrieb	Log-Analyse mit Elastic Stack	21.3
Betrieb	Tracing mit Zipkin	22.2

Die Projekte sind alle auf GitHub verfügbar. In den Projekten gibt es jeweils eine Datei `WIE-LAUFEN.md` mit einer Schritt-für-Schritt-Anleitung, wie die Demos installiert und gestartet werden können.

Die Beispiele bauen nicht aufeinander auf. Dadurch ist es möglich, mit einem beliebigen Beispiel loszulegen.

Danksagung

Ich möchte allen danken, mit denen ich über Microservices diskutiert habe, die mir Fragen gestellt oder mit mir zusammengearbeitet haben. Es sind viel zu viele, um sie alle zu nennen. Der Dialog hilft sehr und macht Spaß!

Viele der Ideen und auch die Umsetzungen sind ohne meine Kollegen bei der innoQ nicht denkbar. Insbesondere möchte ich Alexander Heusingfeld, Christian Stettler, Christine Koppelt, Daniel Westheide, Gerald Preissler, Jörg Müller, Lucas Dohmen, Marc Giersch, Michael Simons, Michael Vitz, Philipp Neugebauer, Simon Kölsch, Sophie Kuna und Stefan Lauer danken.

Weiteres wichtiges Feedback kam von Merten Driemeyer und Olcay Tümce.

Schließlich habe ich meinen Freunden, Eltern und Verwandten zu danken, die ich für das Buch oft vernachlässigt habe – insbesondere meiner Frau.

Und natürlich gilt mein Dank all jenen, die an den in diesem Buch erwähnten Technologien gearbeitet und so die Grundlagen für Microservices gelegt haben.

Bei den Entwicklern der Werkzeuge von *https://www.softcover.io/* möchte ich mich ebenfalls bedanken.

Last but not least möchte ich dem dpunkt.verlag und René Schönfeldt danken, der mich sehr professionell bei der Erstellung des Buchs unterstützt hat.

Website

Die Website zum Buch ist *http://microservices-praxisbuch.de/*. Dort finden sich die Errata und Links zu den Beispielen.