Hochschule für angewandte Wissenschaften München
Fakultät für Informatik und Mathematik

Bachelorarbeit zum Thema:

Entwurf und Bereitstellung von Microservices mit Kubernetes am Beispiel eines CRM-Systems

Zur Erlangung des akademischen Grades Bachelor of Science

Vorgelegt von: Simon Hirner

Matrikelnummer: 02607918

Studiengang: Wirtschaftsinformatik

Betreuer: Prof. Dr. Torsten Zimmer

Abgabedatum: 04.02.2022

Abstract

Bei immer mehr moderne Webanwendungen findet das Architekturmuster der Microservices anwendung. Dieser Trend verändert nicht nur den Entwurf und die Implementierung von Anwendungen, sondern hat auch erhebliche Auswirkungen auf die Bereitstellung und den Betrieb.

In dieser Bachelorarbeit wird der Entwurf und die Bereitstellung von Microservices mithilfe von Kubernetes analysiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Abkürzungsverzeichnis

UNIX Uniplexed Information and Computing Service

CRM Application Programming Interface

CRM Customer Relationship Managment

LoC Lines of Code

SOA Service Oriented Architecture

CI Continuous Integration

CD Continuous Delivery

REST Representational State Transfer

HTTP Hypertext Transfer Protocol

URI Unified Resource Identifier

1 Einleitung

"There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult."

– Tony Hoare

Zunächst wird in diesem Kapitel näher auf die Motivation sowie der Zielsetzung der vorliegenden Bachelorarbeit eingegangen.

Kurzbeschreibung
Gliederung & Struktur der Arbeit

1.1 Motivation

Durch Container, Microservices und Kubernetes hat sich in den letzten Jahren ein erheblicher Wandel in der IT-Branche vollzogen. Containervirtualisierung erleichtert das Bauen großer verteilter Systeme durch das Verbinden von vielen kleinen Services. Im Gegensatz zu monolithischen Anwendungen erleichtern Microservices den Einsatz von verschiedenen Technologien, die Skalierung und den modularen Aufbau (Newman et al., 2015, S. 24 ff.). Um die große Anzahl an Services zu steuern, kommen Orchestrierungssysteme zum Einsatz. In diesem Bereich hat sich in den letzten Jahren Kubernetes zum Standard entwickelt. Das alles sind Trends die DevOps unterstützen und für ein zunehmendes Verschmelzen von Softwareentwicklern und Systemadministratoren sorgen. Heutzutage sind DevOps-Prinzipien tief in modernen Anwendungen verwurzelt und haben erhebliche Auswirkungen auf alle Phasen des Entwicklungszyklus. Vor allem der Entwurf und die Bereitstellung wurden enorm beeinflusst und haben sich grundsätzlich verändert. Den containerisierten, verteilten Systemen gehört die Zukunft (Arundel und Domingus, 2019, S. 1). In dieser Bachelorarbeit wird deshalb der Fokus auf dem Entwurf und der Bereitstellung von modernen Microservice-basierten Anwendungen mit dem Branchenstandard Kubernetes liegen.

1.2 Zielsetzung

Ziel dieser Bachelorarbeit ist es eine Übersicht über
Die
Die Bachelorarbeit widmet sich dem Entwurf und der Bereitstellung von Microservices mit Kubernetes. Die Arbeit wird zuerst die nötigen Methoden und Technologien beschreiben, um diese im Anschluss anhand einer konkreten Fallstudie einzusetzen. Das Ziel der Fallstudie ist es, ein auf Microservices basierendes Customer-Relationship-Management-System (CRM-System) zu entwerfen und dieses mithilfe von Kubernetes bereitzustellen. Dabei soll ein Verfahren, das dem aktuellen Stand der Technik entspricht, vom Entwurf bis zur Bereitstellung von modernen Microservice-basierten Anwendungen implementiert werden.

1.3 Abgrenzung

Die folgenden Aspekte werden nicht betrachtet:

- Sicherheit (Authentifizierung)
- Cloud Computing

Das Ziel dieser Bachelorarbeit ist es mit neuen Methoden und Werkzeugen eine moderne Webanwendung zu gestalten. Dabei soll DevOps-Bezogen der gesamte Softwareprozess von der Entwicklung bis zum Betrieb analysiert und umgesetzt werden.

2 Theoretischer Rahmen

DevOps, Cloud und Container Microservices und Kubernetes
Kubernetes ist der Branchenstandard und die Grundlage für moderne Webanwendungen (Arundel und Domingus, 2019, Vorwort).
Die Verwendungsweise wird maßgeblich beeinflusst durch neue Technologien. (Newman et al., 2015, S. 16)
DevOps, Kubernetes und Microservices sind zwar komplett unterschiedliche Dinge haben jedoch zweifelsfrei Verbindungen und können in Zusammenarbeit ihre Stärken bestmöglich ausnutzen.

2.1 DevOps

Das Kofferwort DevOps ist eine Sammlung von Lösungsansätzen wie IT-Leistungen schnell und risikoarm bereitgestellt werden können. Wie das Kofferwort ”DevOps”bereits andeutet, bewirkt es eine stärkere Verschmelzung zwischen Softwareentwicklung (Development) und IT-Betrieb (Operations). Dabei ist DevOps nich klar definiert. DevOps stellt den Kundennutzen in den Mittelpunkt und erinnert dabei an agile Softwareentwicklungsmethoden. DevOps und aglie Entwicklung schließen sich auch nicht gegenseitig aus unterscheiden sich jedoch durch die einbezogenen Gruppen und Abteilungen. Den während DevOps bezieht sich wie bereits erklärt nicht nur auf den Entwicklungsprozess und die Softwareentwickler sondern auf den gesamten Softwareprozess.
Um DevOps in Unternehmen umzusetzen reicht es nicht die entsprechenden Werkzeuge einzuführen, sondern es muss zu einem Kulturwandel kommen.
DevOps ist eine Sammlung an Methoden und Denkweisen, welche sich immer weiter verbreiten. Das Vorgehen in der Fallstudie dieser Arbeit kommen auch DevOps-Werkzeuge zum Einsatz. Da die gewählten Technologien und Architekturmuster sich nicht nur auf den Entwurf sondern auch
Eine der wichtigsten Werkzeuge von DevOps ist Continuous Delivery.

2.2 Microservices

Im Mittelpunkt dieser Arbeit stehen Microservices. Microservices sind
Bei Microservices handelt es sich um ein Architekturmuster zur Modualisierung von Software (Newman et al., 2015, S. 15). Obwohl der Begriff Microservices noch realtiv jung ist, sind die dahinterstehenden Konzepte bereits älter (Newman et al., 2015, S. 15). Große Systeme werden schon lange in kleine Module unterteilt. Die Besonderheit von Microservices liegt darin, dass die Module einzelne Programme sind. Ein solches einzelnes Programme wird als Microservice bezeichnet. Der Begriff Microservice ist nicht fest definiert (Wolff, 2018, S.

2), deshalb werden im nachfolgenden Kapitel die wichtigsten Eigenschaften und Merkmale betrachtet.
Microservices sind ein Architekturmuster Microservices sind verteilte Systeme
Architekturmuster Erklärung Wenn das Architekturmuster für ein System bestimmend ist, nennt man es auch dessen Architekturstil
Erklärung Monolith
Microservices sind ein Architekturmuster, welches konträr zu einer klassischen monolithischen Architektur ist. Dabei heben sich Microservices durch ihre lose Kopplung und die damit einhergehende erhöhte Unabhängigkeit hervor. Obwohl Microservices ein neuer Begriff sind, ist das Konzept schon deutlich länger vorhanden.
Microservices vs Monolithen
Dabei werden Microservices häufig im geschäftlichen Umfeld eingesetzt (Newman et al., 2015, S. 15).
Marktwachstumsprognose: https://www.instanttechnews.com/technology-news/2020/02/16/cloud-microservices-market-2020-trends-market-share-industry-size-opportunities-analysis-and-forecast-by-2026/
Microservices machen Gebrauch von den neu entwickelten Technologien und Verfahren des letzten Jahrzehnts. Vor allem mit DevOps-Methoden und Kubernetes können ...

2.2.1 Merkmale

Microservices sind eigenständige Programme welche über ein Neztwerk miteinander kommunizieren.

Größe Microservices sollen nur eine Aufgabe erledigen, diese jedoch bestmöglich. Dieser Ansatz ist nicht neu und entstammt der UNIX-Philosophie: ”Mache nur eine Sache und mache sie gut”(Douglas McIlroy). Wie der Name ”Microservices”bereits andeutet, handelt es sich dabei offensichtlich um kleine Services. Eine genaue Festlegung wie groß die Services sein sollten gibt es jedoch nicht. Die Anzahl der Codezeilen (Lines of Code) können einen Hinweis geben, jedoch sind derartige Kriterien stark von der Programmiersprache und dem verwendeten Technologie-Stack abhängig. Stattdessen sollte sich die Größe an fachliche Gegebenheiten anpassen. Je kleiner die Services gestaltet werden, umso stärker kommen die in den nachfolgenden Abschnitten beschriebenen Vor- und Nachteile zur Geltung. Des Weiteren sollte ein Microservice nur so Groß sein, dass er von einem einzigen Entwicklerteam betreut werden kann.
Der Name ”Microservices”deutet schon an, dass es sich offensichtlich um kleine Services handelt. Eine objektive Messung der Größe ist jedoch nicht möglich. Um die Größe eines Microservices zu bestimmen, könnte man beispielsweise die Anzahl der Codezeilen (LoC) verwenden. Doch die Lines of Code hängen starkt von der verwendeten Programmiersprache und dem Technologie-Stack ab. Eine Messung der Größe nach rein objektiven Kriterien macht demnach keinen Sinn (Wolff, 2018, S. 31)(Newman et al., 2015, S. 22). Stattdessen sollte sich die Größe an die fachlichen Gegebenheiten richten. Je kleiner die Services werden desto mehr kommen die in den nächsten Abschnitten beschriebenen Vor- und Nachteile zur Geltung. Ein Microservice sollte von einem einzigen Enticklerteam gehandhabt werden (Newman et

al., 2015, S. 23). Falls das nicht mehr möglich ist, könnte es darauf hindeuten, dass der Microservice zu groß ist.

Sepzialisierung

Eigenständigkeit Microservices laufen als eigenständige Programme und müssen unabhängig voneinander deploybar sein. Jeder Microservice ist ein eigener Prozess, welcher isoliert von den anderen abläuft. Die Isolierung trägt dazu bei, dass verteilte System besser zu verstehen aber auch keine unbemerkten Abhängigkeiten zwischen den Services entstehen zu lassen. Neue Technologien wie Containervirtualisierung können dies erleichtern. Die Kommunikation der Services erfolgt über ein Netzwerk und mittel definierter Schnittstellen (API).
Eine ausreichende Eigenständigkeit ist nur gegeben, sobald die Services unabhängig voneinander verändert und bereitgestellt werden können. Ein Entwicklerteam sollte sich beim

Ersetzbarkeit

2.2.2 Vorteile

Das Aufteilen von Software bringt wichtige Vorteile mit sich.

Modularisierung Bei klassischen Software-Monolithen, welche aus Komponenten zusammengestellt wird, entstehen schnell unerwünschte Abhängigkeiten. Die viele Abhängigkeiten erschweren die Wartung oder Weiterentwicklung. Da die einzelnen Microservices eigene Programme sind, herrscht eine starke Modularisierung. Die Programme sind eigenständig und kommunizieren nur über explizite Schnittstellen. Ungewollte Abhängigkeiten entstehen hier deutlich schwerer. (Wolff, 2018, S. 3)
In der Praxis wird die Architektur von Deployment-Monolithen meistens zunehmend schlechter. (Wolff, 2018, S. 3)

Ersetzbarkeit Da Microservices nur über eine explizite Schnittstelle genutzt werden, können sie einfach durch einen Service, der die selbe Schnittstelle anbietet ersetzt werden. Bei der Ersetzung ist der neue Service nicht an den Technologie-Stack des alten Service gebunden. Auch die Risiken werden geringer, da bei schwerwiegenden Fehlentscheidungen in der Entwicklung, ein Austausch mit weniger Aufwand verbunden ist. (Wolff, 2018, S. 4)
Sie können auch deutlich schneller eingesetzt werden. (Time to Market)

Skalierbarkeit Durch die Unabhängigkeit der Microservices können sie auch unabhängig voneinander skaliert werden. So kann eine einzelne Funktionalität, welche stärker genutzt wird, skaliert werden, ohne das gesamte System zu skalieren. (Wolff, 2018, S. 5)

Technologiefreiheit Des Weiteren führt die Unabhängigkeit auch zu einer großen Technologiefreiheit. Die verwendeten Technologien müssen schließlich nur in der Lage sein die explizite Schnittstelle anzubieten. (Wolff, 2018, S. 5)

Continuous Delivery Ein wesentlicher Grund für die Einführung ist Continuous Delivery. Die kleinen Microservices können leichter deployt werden und das Deployment bietet weniger Gefahren und ist einfacher abzusichern, als bei einem Monolithen. (Wolff, 2018, S. 5)

2.2.3 Herausforderungen

Versteckte Beziehungen

Refactoring

Fachliche Architektur

Komplexität

Verteilte Systeme

2.2.4 Architektur

2.2.5 Integration

2.2.6 Bereitstellung / Deployment und Betrieb

2.2.7 Technologien

2.3 Containervirtualisierung

2.4 Kubernetes

3 Fallstudie

3.1 Problembeschreibung

3.2 Entwurf

3.3 Implementierung

3.4 Bereitstellung

4 Schlussbetrachtung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Literatur

Arundel, J., & Domingus, J. (2019). *Cloud Native DevOps mit Kubernetes: Bauen, Deployen und Skalieren moderner Anwendungen in der Cloud* (1. Auflage). dpunkt.verlag.

Newman, S., Lorenzen, K., & Newman, S. (2015). *Microservices: Konzeption und Design* (1. Aufl). mitp-Verl.

Wolff, E. (2018). *Microservices: Grundlagen flexibler Softwarearchitekturen* (2., aktualisierte Auflage). dpunkt.verlag.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst und keine anderen Hilfsmittel als die angegebenen verwendet habe.

München, den 18. Januar 2022