HSB
Hochschule Bremen
City University of Applied Sciences

Hochschule Bremen
Fakultät 4 – Elektrotechnik und Informatik
Internationaler Studiengang Medieninformatik (B.Sc.)

Exposé

Bachelorthesis

– Transformation einer Bestandsanwendung in eine Cloud-Native-Architektur –

Lukas Karsten (*5011712*)

04. November 2021 (Version 1.01)

1 Einleitung

Moderne IT-Systeme folgen bei ihrer Bereitstellung oft den Prinzipien sogenannter Cloud-Native-Architektur und sind damit so gestaltet, dass sie skalierbar, resilient, administrierbar, observierbar und automatisiert sind [**Lee:2021**]. Allerdings kommt es selten vor, dass diese Systeme von Grund auf neu entwickelt werden. Ein Großteil der essentiellen Geschäftsprozesse bestehender Unternehmen läuft bereits in Systemen, bei denen diese architektonischen Erkenntnisse noch keine Anwendung fanden. Eine große Herausforderung ist es daher heute, die bestehende Infrastruktur schrittweise so zu verändern, dass möglichst viele der modernen Prinzipien angewendet werden können, ohne den laufenden Betrieb zu stören oder eine komplette Neuentwicklung der Server-Architektur zu erzwingen [**Fiedelholtz:2021**]. Vor dieser Aufgabe steht auch das Unternehmen *Startnext*, eine Online-Plattform zum Crowdfunding von kreativen Ideen, nachhaltigen Projekten und Start-Ups. Im Rahmen dieser Arbeit wird ein Konzept erarbeitet, um die Startnext-Plattform von der bestehenden, klassischen Architektur schrittweise in eine Cloud-Native-Infrastruktur zu überführen und die Vorteile dieser Technologie effektiv zu nutzen. Die erarbeiteten Methoden werden praktisch umgesetzt, diskutiert und der Erfolg der Maßnahmen bewertet.

2 Problemstellung und Lösungsansatz

2.1 Problemstellung

Moderne Softwareentwicklung stellt hohe Anforderungen an ihre Infrastruktur. Um neue Features zu testen, werden kurzzeitig Testinstanzen benötigt, Änderungen sollen mehrmals täglich ins Produktivsystem eingespielt werden [**IBM:2019**]. Lastspitzen müssen schnell abgefangen und auf schwerwiegende Bugs muss mit zurückrollen zu vorherigen Versionen reagiert werden [Quelle]. Klassische Bereitstellung von Software über selbst- oder fremdgehostete Server schränkt bei Bedarf von mehr Rechenleistung, weiteren Rechen-Instanzen oder Serverstandorten die Anpassungsmöglichkeiten stark ein und lässt Änderungen an der Infrastruktur nur sehr träge zu.
Die Wartung und Aktualisierung der Infrastruktur erfolgt über das Einwählen und manuelle Aufspielen von Updates. Dabei verändert sich der Zustand jeder einzelnen Instanz über die Zeit. Es kommt zum auseinanderdriften von Zuständen der einzelnen Server. Zusätzlich ist der Weg zu diesem Zustand schwer bis nicht nachvollziehbar. Das Updaten der Instanzen wird unvorhersehbar und potentielle Fehler passieren auf produktiven Systemen.

2.2 Lösungsansatz

Einen Lösungsansatz für genau diese Probleme bietet das Konzept von Infrastructure as a Service (IaaS). Als IaaS werden Cloud-Computing-Services bezeichnet, die Rechner-, Speicher- und Netzwerkressourcen auf Anfrage bereitstellen. Dabei wird jede der Ressourcen als separate Komponente angeboten und nur für die Dauer der Nutzung in Rechnung gestellt. Der Aufwand und die Kosten für das Betreiben von eigenen physikalischen Servern

und der zugehörigen Infrastruktur werden also umgangen, während das Einrichten sowie Deinstallieren neuer Komponenten vollautomatisiert und innerhalb weniger Sekunden erfolgen kann.
Durch die alleinige Umstellung zu einem Hosting bei einem Cloud-Anbieter ergeben sich allerdings noch keine signifikanten Vorteile im Betrieb der Anwendung. Um die mit IaaS erworbene Skalierbarkeit, Flexibilität und vereinfachte Verwaltung zu nutzen, bedarf es eines Neudenkens im Design der Infrastruktur und ihren Bereitstellungsprozessen, im allgemeinen als Cloud-Native-Architektur bezeichnet [Quelle]. Allerdings bringen diese neuen Konzepte wiederum Herausforderungen mit sich, die es zu identifizieren und bewältigen gilt.
Im Rahmen der Bachelorthesis werden bereits gängige Konzepte und „Best-Practices“ im Betrieb von cloudnahen Anwendungen evaluiert. Am Beispiel der Startnext-Infrastruktur und durch das Erstellen von Proof of Concepts werden sie auf ihre Anwendbarkeit, Sinnhaftigkeit auf ihre Machbarkeit untersucht und diskutiert.

3 Fachliche Überlegungen und Zielsetzung

3.1 Fachliche Überlegungen

Cloud-native dient als Überbegriff für eine Vielzahl an Paradigmen und Prinzipien, die das Entwerfen und Implementieren von Infrastrukturen und Software als skalierbare, resiliente, administrierbare, observierbare und automatisierte Systeme ermöglichen. Eine übergeordnete Rolle in Cloud Native Architekturen spielt das *Immutable Infrastructure* Paradigma, welches die Grundlage für die Implementierung vieler weiterer Prinzipien bildet. Dieser Ansatz zur Software-Verwaltung verfolgt das Ziel eines weitestgehend unveränderlichen Zustands eines bestehenden Systems. Wird also ein Update eines Servers benötigt, wird ein neuer Server mit dem neuen gewünschten Zustand erstellt und die alte Instanz entfernt, anstatt Komponenten im bestehenden System einfach zu verändern Hirschfeld:2018. Dieser Ansatz ist eine direkte Antwort auf das auseinanderdriften von Zuständen und dem dadurch unkalkulierbaren Ausgang von Zustandsänderungen wie in **??** beschrieben. Zusätzlich müssen Änderungen nicht mehr an bereits produktiven Umgebungen vorgenommen werden, deren Ausfall direkte Auswirkungen auf die Anwendung hätte. Im Gegenteil können diese Änderungen ausgeführt, geprüft und anschließend in den produktiven Einsatz geschaltet werden.

3.1.1 Technologien

Bei der Umsetzung von Immutable Infrastructure spielen Werkzeuge und Technologien eine entscheidende Rolle, um die gewünschte Nachvollziehbarkeit, Berechenbarkeit sowie Skalierbarkeit zu gewährleisten.
Um Systemumgebungen reproduzierbar zu erstellen, können Images genutzt werden. Ein Abbild des Speichers einer Maschine mit allen Metadaten und Konfigurationen, die benötigt werden um die Maschine wiederherzustellen oder zu duplizieren. Moderne Tools Ermöglichen das Erstellen von Konfigurationen als Code (As-Code), aus denen Maschinenimages

für unterschiedliche Plattformen erstellt werden können. Durch den „As-Code“-Ansatz werden Images versionierbar. Änderungen an diesen sind nachvollziehbar und widerruflich. Zusätzlich können über verschiedene Instanzen genutzte Grund-Images, auch Golden Images genannt [**GoldenImage:2018**], für eine einheitliche und erleichterte Wartung der Systeme sorgen.
Der „As-Code“-Ansatz ist ebenso bei der Provisionierung von Hosts möglich. Infrastructure-As-Code (IaC) Tools ermöglichen das Erstellen von Umgebungskonfigurationen als Code, die von IaS-Anbietern interpretierbar sind. So wird Idempotenz bei der Bereitstellung von Hosts gewährleistet. Die Provisionierung wird sowohl nachvollziehbar als auch automatisierbar und Änderungen an Umgebungen sind versionier- sowie testbar. Durch Versionierung und in Form von Code vorliegende Konfigurationen aller Bestandteile der Anwendung ist es möglich, den gesamten Bereitstellungsprozess zu automatisieren und den in **??** beschriebenen Anforderungen an moderne Infrastruktur zu entsprechen. Der Prozess vom Provisionieren des Hosts, über das Konfigurieren dieser und dem Aufspielen der Software soll gemeinsam mit entsprechenden Tests jeder dieser Schritte automatisiert ablaufen. So kann auf Anforderungsänderungen schnell reagiert werden und das komplexe System der ~~Startnext-~~Infrastruktur bleibt administrierbar.

Darüber hinaus werden Konzepte und Tools benötigt, welche den Anwendungsbetrieb und ihre automatisierte Bereitstellung sicher, einfach zu administrieren und unanfällig für Fehler machen. Das Verwalten von Geheimnissen, Orchestrieren und Registrieren von Services und Testens sind hier zu nennen.

4 Verwandte Arbeiten

Die ansteigende Nachfrage nach Cloud-nativen Technologien spiegelt sich ~~auch~~ in der wissenschaftlichen Literatur wieder. So findet auch das Thema Immutable Infrastructure als eine der grundlegenden Disziplinen in Cloud Native hohe Aufmerksamkeit in wissenschaftlichen Abhandlungen.

Rob Hirschfeld diskutiert auf der SREcon-Konferenz 2018 Immutable Deployments und ihren Mehrwert [**Hirschfeld:2018**]. Er arbeitet heraus, dass der Ansatz Immutable Infrastructure zusammen mit Cloud Computing zum kosteneffizienten Ansatz für das Bereitstellen von Laufzeitumgebungen wird. Hirschfeld teilt Anwendungsweisen von ImmuInfra in drei verschiedene Immutable Patterns ein:

- Einfaches Zurücksetzen und Neubespielen der Maschine
- Live-Booten der Maschine und anschließendes Konfigurieren
- Erstellen eines fertigen Images und aufspielen des Images

Diese drei Immutable Patterns gehen alle mit eigenen Vor- sowie Nachteilen einher. Während der erste Ansatz keine vollständige ImmuInfra-Lösung ist, bietet er dennoch mehr Nachvollziehbarkeit als klassiche Deployments. Pattern 2 und 3 unterscheiden sich hauptsächlich in der Nutzung von Images, die eine Wiederverwendbarkeit, Zurückrollen und ein zeitlich effektives Aufspielen ermöglichen. Der Beitrag stellt klar, dass die Nutzung von

Images die meisten Vorteile ~~bereitstellt~~, allerdings auch die höchste Komplexität mit sich bringt.

Mit der Gestaltung eines resilienten und skalierenden Cloud-Native Systems durch eine selbstadministrierende Anwendung beschäftigt sich Toffetti [**Toffetti:2017**]. Toffetti erstellt ein Cloud-Native Design, welches in der Lage ist, selbständig zu skalieren und sich selbst zu heilen. Die Architektur beinhaltet neben Cloud Orchestrierung und verteiltem Konfigurationsmanagement ein verteiltes Datenbanksystem sowie die Fähigkeit der Leader-Election. Anschließend wird dieses Design auf eine bestehende Legacy Web-App angewandt. Er und sein Team emulieren Fehler sowohl in den containerisierten Teilen der Anwendung als auch in den VMs, welche die Container hosten. Sie messen die Reaktion des Systems auf Fehler und diskutieren diese. Toffettis Journal bietet einen robusten Leitfaden für die Wahl von DesignPatterns und eine Orientierung für das Erstellen eines selbstgemanagten Cloud-Native Systems.

5 Konkrete Aufgaben und Zeitplan

Die Ausführung der Bachelorarbeit ist zeitlich wie folgt geplant:
Dauer: 9 Wochen

- Woche 1: Literaturrecherche,Einleitung, Einrichten des Dokuments
- Woche 2: Verfassen der Problemstellung, Zielsetzung und möglichen Lösungsansätzen
- Woche 3: Entwickeln der Anforderungsanalyse
- Woche 4: Grundlagen und verwandte Arbeiten - Verfassen und Recherche
- Woche 5: Konzeption - Entwickeln eines Entwurfs zur Lösung der Problemstellung
- Woche 6-7: Prototypische Umsetzung
- Woche 8: Evaluation - Bewertung und Diskussion der Ergebnisse
- Woche 9: Überarbeitung, Korrektur und Druck des Dokuments

6 Vorläufige Gliederung

Eigenständigkeitserklärung

Zusammenfassung / Abstract

1. Einleitung
   1.1. Problemfeld
   1.2. Ziesetzung
   1.3. Lösungsansatz
   1.4. Aufbau der Arbeit
2. Anforderungsanalyse
   2.1. Funktionale Anforderungen
   2.2. Nicht-funktionale Anforderungen
   2.3. Zusammenfassung
3. Grundlagen und verwandte Arbeiten
   3.1. Immutable Infrastructure
   3.2. HashiCorp - Packer
   3.3. Selfmanaged Infrastructure
   3.4. Monitoring von Infrastruktur
   3.5. Automatisierte Orchestrierung und Konfiguration von Infrastruktur
   3.6. HashiCorp - Nomad & Consul

7 Unterschriften

Student:

Name: *Lukas Karsten*

Adresse: *Bismarckstraße 156, 28205 Bremen*

Telefon: *0157/81733962*

E -Mail: *lkarsten@stud.hs-bremen.de*

Unterschrift (Student_in)

Erstgutachter_in / Betreuer_in: *Prof. Dr. Lars Braubach?/Hans Mündelein?*

Unterschrift (Erstgutachter)

Zweitgutachter_in:

Literatur