FOM Hochschule für Ökonomie & Management

Hochschulzentrum Frankfurt am Main

Master Thesis

im Studiengang IT-Management

zur Erlangung des Grades eines

Master of Science (M.Sc.)

über das Thema

Architektonischer Entwurf und Modellierung einer Cloud Native Plattform für den Einsatz von containerisierten Microservices und Anwendungen

von

Dominik Otte

Betreuer :	Dr. phil. Patrick Hedfeld
Matrikelnummer :	585039
Abgabedatum :	6. September 2023

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

CPU Central Processing Unit

PaaS Platform-as-a-Service

RAM Random-Access Memory

SaaS Software-as-a-Service

Symbolverzeichnis

1 Einleitung

Die fortschreitende Digitalisierung und die rasante Entwicklung von Technologien haben die Art und Weise, wie Softwareanwendungen entwickelt und bereitgestellt werden, grundlegend verändert. In der heutigen Arbeitswelt ist Agilität, Skalierbarkeit und Effizienz entscheidend für den Erfolg von Unternehmen. Aus diesem Grund haben sich neben zeitgemäßen Softwarearchitekturen und Technologien auch innovative Arbeitsmethoden und Vorgehensmodelle als grundlegende Elemente zeitgemäßer Softwareentwicklung und Bereitstellung etabliert. Die Weltwirtschaft steht vor einer Ära, in der die nahtlose Bereitstellung von Diensten und Anwendungen unabhängig von räumlichen Einschränkungen von höchster Priorität ist. Die jüngsten Ereignisse, wie die globale COVID-19-Pandemie, haben die Notwendigkeit unterstrichen, dass Unternehmen agil und flexibel auf sich ändernde Marktbedingungen reagieren können. Laut einem Bericht von McKinsey hat die Pandemie die digitale Transformation beschleunigt und die Nachfrage nach Cloud-basierten Lösungen verstärkt, um aus dem HomeOffice heraus zu arbeiten, Geschäftsprozesse umzugestalten und Kunden digital zu erreichen[1].

In einer dynamischen Cloud Umgebung ist es von signifikanter Relevanz, die Methoden und Ansätze zu verstehen, die bei der Gestaltung und Modellierung von Cloud-Native Plattformen für containerisierte Anwendungen angewendet werden können. Die vorliegende Thesis widmet sich genau diesem Thema und beabsichtigt, einen tiefgreifenden Einblick in den architektonischen Entwurf und die Modellierung solcher Plattformen zu bieten. Indem sie aktuelle Nachrichten, Trends und bewährte Praktiken berücksichtigt, strebt die Thesis danach, einen Beitrag zur Weiterentwicklung dieser Marktaktuellen Technologie zu leisten und Leser bei der Realisierung ihrer digitalen Visionen zu unterstützen.

1.1 Ein Medienecho

Der weitreichende und stetig wachsende Themenbereich der Cloud-Technologien hat über die letzten Jahre hinweg eine kontinuierliche Präsenz in einer Vielzahl von Medienkanälen aufrechterhalten, darunter Fachzeitschriften, Online-Blogs, Podcasts sowie verschiedenste andere Formen von Medienplattformen und -Formaten. So behandelt der Artikel von David Linthicum eine skeptische Sicht auf eine von Gartner veröffentlichte Studie, in welcher prognostiziert wird dass bis 2025 mehr als 95% von Anwendungs-Workloads in einer Cloud-Native Plattform laufen werden. Grundlegend möchte der Autor hervorheben, dass

[1] Vgl. *McKinsey*, 2020, o.S.

Unternehmen bei neuen Technologien neben Chancen auch die Risiken berücksichtigt sollen[2]. So werden im Artikel drei Aspekte betont:

- Vendor Lock-in: Anwendungen die gezielt für eine bestimmt Cloud Plattformen entwickelt wurden, lassen sich schwieriger auf andere Plattformen übertragen. Die eingeschränkte Portabilität steht somit zum Teil im Widerspruch dessen was Cloud-Native Anwendungen definiert.
- Skill Gap: Unternehmen ohne Erfahrung stehen vor Herausforderungen, die zusätzliche Schulungen oder Ressourcen erfordern, was zu schlecht konzipierten oder übermäßig komplexen Anwendungen führen kann. Dies wiederum könnte die Effizienz beeinträchtigen und möglicherweise die gesamte Umsetzung gefährden.
- unkontrollierter Kostenanstieg: Die nutzungsabhängige Preisgestaltung kann zu unvorhergesehenen Mehrkosten führen, wenn Anwendungen plötzlich stark frequentiert werden.

Pokemon Go, ein Spiel welches 2016 für Android und iOS Gräte erschien also für Smartphones und Tablets, setzt ebenfalls auf die Cloud Technologie von Google. Im News Blog von Google wird dargelegt wie es den Entwicklern möglich war mithilfe der bereitgestellten Cloud Technologie Live-Events im Spiel mit einem Transaktionsvolumen von 400.000 bis fast zu einer Millionen Transaktionen pro Sekunde umzugehen. Weiterhin wird ausgeführt, dass im Backend der Infrastruktur Services flexibel, nach bedarf skalieren. Täglich werden 5-10 Terabyte an Daten im Rahmen Datenanalysen verarbeitet. Außerdem wird hervorgehoben, dass die Stabilität und Gesundheit durch umfassendes Logging, Monitoring und umfangreiche Dashboards sichergestellt wird[3].

1.2 Problemstellung und Zielsetzung der Thesis

Die traditionelle monolithische Anwendungsarchitektur wird zunehmend von einer auf Microservices[4] basierenden Architektur abgelöst, die es ermöglicht, Anwendungen in kleinere, eigenständige Komponenten zu zerlegen. Diese als containerisierte Microservices bezeichneten Software Komponenten, können unabhängig voneinander entwickelt, bereitgestellt und skaliert werden. Cloudlösungen werben damit, eine ideale Umgebung für den Einsatz solcher Microservices zu bieten, da sie unter anderem Ressourcen elastische bereitstellen und automatische skalieren können[5].

[2] Vgl. *Linthicum*, *D.*, 2023, o.S.
[3] Vgl. *Priyanka Vergadia*, *J. P.*, 2021, o.S.
[4] Vgl. *Wolff*, *E.*, 2018, S.4.
[5] Vgl. *Henneberger*, *M.*, 2016, S.8-19.

Der Einsatz von containerisierten Microservices in der Cloud bringt jedoch auch neue Herausforderungen mit sich. Wie das Wort Microservice bereits indiziert, liegt die Vermutung nahe, dass in einer Anwendungslandschaft zahlreiche weitere Microservices existieren, die als Konglomerat einen ganzheitlichen Service bereitstellen. Für den Betrieb im Produktiven Umfeld bedeutet dementsprechend, dass der Monitoring aufwand für Microservices steigt. Eine entscheidende Frage ist die Auswahl geeigneter Technologien und Tools, um die Microservices effizient zu verwalten und orchestrieren[6]. Container-Orchestrierungssysteme wie Kubernetes haben sich als Industriestandard etabliert und bieten zudem eine Vielzahl an Konfigurationsmöglichkeiten. Die Gewährleistung von Skalierbarkeit ist ein weiterer essenzieller Aspekt im Cloud Kontext. Um dynamische Workloads effizient zu bewältigen und Ressourcenverschwendung zu vermeiden, muss eine Cloud-Native Plattform in der Lage sein, Anwendungen flexibel zu skalieren. Eine effektive Verwaltung und Auslastung der Ressourcen in der Cloud sind entscheidend für die Wirtschaftlichkeit und Leistungsfähigkeit der Plattform[7].

Die Thesis beschäftigt sich mit der Forschungsfrage: „Wie kann ein architektonischer Entwurf für eine Cloud-Native Plattform aussehen um den neuen Herausforderungen in einer Cloud-Native Umgebung zu begegnen“? Hauptaugenmerk liegt im Vergleich zu klassischen Architekturen nicht auf der Hardware Infrastruktur und der Ressourcen Ausstattung wie Central Processing Unit (CPU), Random-Access Memory (RAM) und Speicherplatz, sondern Aspekte im Sinne der Application Observability, die für den reibungslosen Betrieb der Plattform und den darauf ansässigen Anwendungen von Belang sind. Weiterhin betrachtet die Thesis eine eigenständig administrierte, on-Premise betriebene Plattform und keinen Dienstleistungsbezug von einem Service Provider in Form einer Software-as-a-Service (SaaS) oder Platform-as-a-Service (PaaS) Lösung.

Das vorrangige Ziel dieser Forschungsarbeit besteht darin, eine eingehende Untersuchung des Themenfelds Application Observability im Kontext von Cloud-Native Anwendungen durchzuführen, wobei besonderes Augenmerk auf die Identifizierung und Analyse relevanter Aspekte gelegt wird, die für den effizienten Betrieb von Anwendungen relevant sind. Darüber hinaus beabsichtigt diese Arbeit, die gewonnenen Erkenntnisse und Einsichten in ein umfassendes Architekturmodell zu überführen, das als Leitfaden und Rahmenwerk für die Implementierung und das Management von Application Observability in Cloud Native Umgebungen dienen soll.

[6] Vgl. *Wolff, E.*, 2018, S.12-16.
[7] Vgl. *Kubernetes*, 2022, o.S.

1.3 Abgrenzung

Die Festlegung der Abgrenzung der vorliegenden Thesis ist von entscheidender Bedeutung, um den Umfang der Arbeit klar zu definieren und den Fokus gezielt auf die Themenspezifischen Aspekte des architektonischen Entwurfs und der Modellierung einer Cloud-Native Plattform für containerisierte Microservices und Anwendungen zu lenken. Durch die klare Definition des Umfangs werden die Grenzen der Untersuchung im weitreichenden Cloud-Native Themenkomplex abgesteckt und die Aufmerksamkeit auf die relevanten Themenbereiche gelenkt, die im Kontext dieser Arbeit von Interesse sind.

- **Technologische Ausrichtung:** Die Hausarbeit konzentriert sich auf Cloud-Native Technologien und -Ansätze, insbesondere auf die Verwendung von Containern und Microservices. Andere Ansätze, wie beispielsweise virtuelle Maschinen oder herkömmliche monolithische Architekturen, werden nicht im Detail behandelt.

- **Plattformfokus:** Die Arbeit legt den Schwerpunkt auf die Entwicklung einer Cloud-Native-Plattform, die den spezifischen Anforderungen von containerisierten Microservices und Anwendungen gerecht wird. Dabei werden Aspekte wie Architekturdesign, Auswahl geeigneter Technologien und Integration von Monitoring-, Logging- und Tracing-Funktionalitäten berücksichtigt. Die Implementierung und konkrete Umsetzung der Plattform in einer bestimmten Cloud-Umgebung oder mit spezifischen Tools wird jedoch nicht im Detail behandelt.

- **Anwendungsbereich:** Die Hausarbeit fokussiert sich auf den generellen architektonischen Entwurf und die Modellierung einer Cloud-Native-Plattform für containerisierte Microservices und Anwendungen. Es werden keine spezifischen Anwendungsdomänen oder Industrien betrachtet. Die vorgeschlagene Architektur und Modellierung sollten jedoch auf verschiedene Anwendungsfälle und Branchen anwendbar sein.

- **Zeitliche Betrachtung:** Die Hausarbeit bezieht sich auf den aktuellen Stand der Technologie und Best Practices zum Zeitpunkt der Erstellung der Arbeit. Zukünftige Entwicklungen oder Trends im Bereich der Cloud-Native-Architektur und Containerisierung können nicht berücksichtigt werden.

[exkludiert sind Managed Services wie (AWS, GKE, etc.) Zusammenfassung der Abgrenzung]

1.4 Methodik und Vorgehensweise

Literaturrecherche, Modellentwicklung auf Basis der Literatur, Anforderungsanalyse, Prototyp Entwicklung in Form einer praxis Implementierung

2 Softwarearchitektur

2.1 Begriffsdefinition

Informationssysteme sind Systeme, die aus menschlichen und maschinellen Teilsystemen bestehen. Diese werden für den Zweck der optimalen Bereitstellung der richtigen Informationen im wirtschaftlichen Kontext eingesetzt[8]. Merkmale von Informationssystemen sind Offenheit, Dynamik und Komplexität. Um solche komplexe Systeme zu steuern werden sogenannte IT-Architekturen entwickelt[9].

Der Begriff der Architektur beschreibt im traditionellen Sprachgebrauch die grundlegende Gestaltung und Struktur eines Gebäudes, wie beispielsweise die Materialauswahl, Werkzeuge oder Skizzen[10]. Analog beschreibt die Architektur von Informationssystemen verschiedene Elemente, wie Dokumentationen und Prozesslandkarten[11].

IT-Architekturen beschreiben modellhaft den grundsätzlichen Aufbau und die Struktur aller Komponenten in einem System, sowie deren Beziehung zueinander[12]. Aus der Sicht der Implementierung kann die IT-Architektur als eine Struktur informationstechnischer Systeme beschrieben werden[13].

2.2 Architekturmodelle

2.2.1 3-Tier Architektur

2.2.2 Monolith

2.2.3 Microservices

In einer Microservice-Architektur wird eine Anwendung als eine Reihe kleiner monofunktionaler Module, den Microservices, realisiert. Jeder Microservice führt dabei einen eigenständigen Prozess aus, wobei die Kommunikation über Schnittstellen verläuft[14]. Im Gegensatz zu traditionellen monolithischen Architekturen, in denen verschiedene Module

[8] Vgl. *Becker*, *A.*, 2011, S.8.
[9] Vgl. ebd., S.11.
[10] **schuetz2017**.
[11] **schuetz2017**.
[12] Vgl. *Knoll*, *M.*, 2018, S.889.
[13] Vgl. ebd., S.890.
[14] Vgl. *Filho*, *M.* et al., 2021, S.1.

und Subsysteme in einer Anwendung integriert sind und zentral zusammenarbeiten, werden die Microservices unabhängig voneinander entwickelt, bereitgestellt und skaliert[15].

Jeder Microservice übernimmt stets eine klar abgegrenzte Aufgabe und hat einen in sich abgeschlossenen Funktionsumfang. So wirken sich Ausfälle nicht direkt auf das Gesamtsystem aus, sondern nur auf den Funktionsumfang des jeweiligen Microservices. Ebenfalls hat jeder Microservice eine eigene Bedieneroberfläche[16]. Microservices können verschiedene Technologien, wie zum Beispiel verschiedene Programmiersprachen oder Plattformen nutzen, da der interne Aufbau von der Außenwelt abgeschirmt ist. Ebenfalls besitzt jeder Microservice eine eigene Datenbank[17]. Die einzelnen Microservices werden in einem Container zusammengefasst oder auf virtuellen Maschinen installiert[18].

Microservices bilden eine abgeschlossene Einheit und bestehen aus einer Datenschicht, einer Funktionsschicht und einer eigenen Präsentationsschicht (Abbildung **??**): In der Datenschicht verwaltet jeder Microservice seine eigene Datenbank. So können für verschiedene Microservices Datenbanken mit unterschiedlichen Technologien genutzt werden. Auch in der Funktionsschicht können je nach Microservice verschiedene Programmiersprachen und Technologien verwendet werden. Die Präsentationsschicht stellt sicher, dass jeder Microservice als abgeschlossene Einheit voll funktionsfähig ist[19].

Des Öfteren wird mit Microservices das Gesetz von Conway in Zusammenhang gebracht: Es besagt, dass die entwickelte Architektur eine Kopie der Kommunikationsstruktur des Unternehmens darstellt[20]. Werden Projekte somit streng nach Organisationsaufbau entwickelt, sodass es ein Team für Präsentationsschicht, Funktionsschicht und Datenschicht gibt, entsteht auch ein streng monolithisches Softwaresystem. Demnach ist der Abstimmungsaufwand untereinander vergleichsweise sehr hoch[21].

Im Unterschied dazu, wird in der Microservice-Architektur ein System nach fachlichen Komponenten aufgeteilt (Abbildung **??**)[22]. Ein Team entwickelt somit einen Microservice inklusive der Präsentationsschicht, Funktionsschicht und Datenschicht. Daraus resultiert, dass das Entwicklerteam unabhängig ist; Abstimmungen und Kommunikationsbedarf sind innerhalb des kleinen Teams effizienter möglich[23]. Zwischen Teams verschiedener Microservices besteht ebenfalls nur noch geringer Abstimmungsaufwand[24].

[15] Vgl. *Laigner*, *R.* et al., 2021, S.1.
[16] Vgl. *Albrecht*, *W.*, 2020, S.79.
[17] **wolff2018**.
[18] Vgl. *Albrecht*, *W.*, 2020, S.79.
[19] Vgl. ebd., S.81 f.
[20] **fowler2015**.
[21] Vgl. *Albrecht*, *W.*, 2020, S.81.
[22] Vgl. ebd., S.81 f.
[23] **wolff2018**.
[24] Vgl. *Albrecht*, *W.*, 2020, S.81.

2.3 Container

2.4 Automation und Orchestration

3 Cloud Computing

3.1 Begriffsdefinition

3.2 Grundlagen der Cloud-Technologie

3.3 Cloud Native Plattform

Begriffsdefinition:[25]

3.4 Kubernetes

[25] Vgl. *Kratzke, N.*, 2021, S. 33-34.

4 Application Observability

4.1 Begriffsdefinition

4.2 Grundbausteine der Application Observability

4.2.1 Protokolle und Log-Dateien

4.2.2 Metriken und Performance Daten

4.2.3 Tracing

4.3 Herausforderungen in Cloud Native Umgebungen

5 Methodische Vorgehensweise

5.1 Auswahl Prototyping-Ansatz(Begründung Wahl der Methodik)

5.2 Identifizierung der Kernanforderungen an eine Cloud Native Plattform

5.3 Planung und Design

6 Theoretischer Entwurf eines Modells

6.1 Ausprägungsmerkmale des Modells

6.2 Datenflussdiagramm

7 Prototypentwicklung aus dem theoretischen Modell

7.1 Aufbau der Laborumgebung

7.2 Iterative Prototyping Schleife (Auswahl der Tools)

7.3 Validierung und Test

8 Ergebnisse und Diskussion

8.1 Zusammenfassung der Ergebnisse

8.2 Handlungsempfehlung bei der Implementierung

9 Kritische Betrachtung

9.1 Limitation der angewandten Methodik

9.2 Limitation der Ergebnisse

9.3 Ausblick für künftige Forschungsarbeiten

10 Fazit und Ausblick

Anhang

Literaturverzeichnis

Albrecht, *Wolfgang* (2020): Softwarearchitektur, in: *Wehking*, *Karl-Heinz* (Hrsg.), Technisches Handbuch Logistik 2: Fördertechnik, Materialfluss, Intralogistik, o. O.: Springer Vieweg, 2020, S. 71–84

Becker, *Alexander* (2011): Nutzenpotenziale und Herausforderungen Service-orientierter Architekturen, o. O.: Gabler Verlag, 2011

Filho, *Messias*, *Pimentel*, *Eliaquim*, *Pereira*, *Wellington*, *Maia*, *P.*, *Cort'es*, *Mariela I.* (2021): Self-Adaptive Microservice-based Systems - Landscape and Research Opportunities, in: CoRR, abs/2103.08688 (2021)

Henneberger, *Matthias* (2016): Von „Cloud Enabling "zu „Cloud Native ": Wie Cloud Computing die Unternehmens-IT verändert, in: Wirtschaftsinformatik & Management, 8 (2016), S. 8–19

Knoll, *Matthias* (2018): IT-Architektur, in: HMD Praxis der Wirtschaftsinformatik, 55 (2018), S. 889–892

Kratzke, *Nane* (2021): Cloud-native Computing: Software Engineering von Diensten und Applikationen für die Cloud, o. O.: Carl Hanser Verlag GmbH Co KG, 2021

Laigner, *Rodrigo*, *Zhou*, *Yongluan*, *Salles*, *Marcos Antonio Vaz*, *Liu*, *Yijian*, *Kalinowski*, *Marcos* (2021): Data Management in Microservices: State of the Practice, Challenges, and Research Directions, in: CoRR, abs/2103.00170 (2021)

Wolff, *Eberhard* (2018): Das Microservices-Praxisbuch: Grundlagen, Konzepte und Rezepte, o. O.: dpunkt. verlag, 2018

Internetquellen

Kubernetes (2022): Was ist Kubernetes?, <https://kubernetes.io/de/docs/concepts/overview/what-is-kubernetes/> (2022) [Zugriff: 2023-09-03]

Linthicum, *David* (2023): Drei Nachteile: Was gegen Cloud-Native spricht, <https://www.computerwoche.de/a/was-gegen-cloud-native-spricht,3613754> (2023) [Zugriff: 2023-08-29]

McKinsey (2020): COVID-19 digital transformation & technology | McKinsey, <https://www.mckinsey.com/capabilities/strategy-and-corporate-finance/our-insights/how-covid-19-has-pushed-companies-over-the-technology-tipping-point-and-transformed-business-forever/> (2020) [Zugriff: 2023-08-22]

Priyanka Vergadia, *James Prompanya* (2021): How Pokémon GO scales to millions of requests?, <https://cloud.google.com/blog/topics/developers-practitioners/how-pok%C3%A9mon-go-scales-millions-requests?hl=en> (2021) [Zugriff: 2023-08-29]

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde/Prüfungsstelle vorgelegen hat. Ich erkläre mich damit einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hochgeladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Mörfelden-Walldorf, 6.9.2023 _____

(Ort, Datum) (Dominik Otte)