CAMPUS 02 Fachhochschule der Wirtschaft
Studiengang Innovationsmanagement

MASTERARBEIT

Innovationsgetriebenes Kubernetes Design

Eine anforderungsbasierte Vorgehensweise für nachhaltiges Clusterdesign

Daniel Drack, BSc
PKZ: 1910318014

im Rahmen der Lehrveranstaltung
Technologiefrüherkennung und -assessment

Graz, Dezember 2020

betreut durch DI Dr. Manuela Reinisch
manuela.reinisch@lv.campus02.at

begutachtet durch FH-Prof. DI Dr. mont. Michael Terler
michael.terler@campus02.at

CAMPUS 02 Fachhochschule der Wirtschaft
Master Program Innovationsmanagement

MASTER THESIS

Innovation-Driven Kubernetes Design

A requirements-based approach to sustainable cluster design

Daniel Drack, BSc
PKZ: 1910318014

Early technology detection and assessment

Graz, December 2020

supervised and reviewed by DI Dr. Manuela Reinisch
manuela.reinisch@lv.campus02.at

supervised by FH-Prof. DI Dr. mont. Michael Terler
michael.terler@campus02.at

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am 12. Dezember 2020 Daniel Drack, BSc

GLEICHHEITSGRUNDSATZ

Aus Gründen der Lesbarkeit wird in dieser Arbeit darauf verzichtet, geschlechtsspezifische Formulierungen zu verwenden. Jedoch wird ausdrücklich festgehalten, dass die bei Personen verwendeten männlichen Formen für beide Geschlechter zu verstehen sind.

„Für Peter.

KURZFASSUNG

Kubernetes wird von mehr als 77% aller IT-basierten Unternehmen der Branche produktiv genutzt und der Platzhirsch auf dem Markt der Container-Orchestrierung. Die meisten Unternehmen berichten von großen Kosteneinsparungen und einer Steigerung der Entwicklungsgeschwindigkeit durch den Einsatz von Kubernetes. Das Ausmaß der Verbesserung hängt jedoch davon ab, wie gut das Kubernetes-System zum Unternehmen und seinen Anforderungen passt. Kubernetes besteht aus vielen Komponenten und kann auf vielfältige Weise genutzt werden. Für viele der Komponenten von Kubernetes gibt es diverse Produkte am Markt, daher wird eine Methode benötigt um die besten Produkte auszuwählen. Um den Entwurf der richtigen Kubernetes-Lösung für eine bestimmte Idee oder ein Konzept zu unterstützen, wurde ein Modell vorgeschlagen, das aus einem Designprozess und einem Anforderungskatalog besteht. Der Anforderungskatalog ist stark in den Prozess integriert. Dabei handelt es sich um einen Fragebogen der dabei hilft die notwendigen Anforderungen für den zukünftigen Cluster zu erfassen. Der Prozess ist in sieben Schritte unterteilt, beginnend mit der Notwendigkeit eine geeignete Plattform für eine innovative Idee zu finden. Die anderen Schritte konzentrieren sich auf die Ermittlung der Anforderungen, die Gestaltung des Clusters, die Suche nach geeigneten Produkten und die Integration des Kubernetes-Systems in das Unternehmen. Der Anforderungskatalog ist in drei Gruppen unterteilt, die insgesamt 65 Punkte umfassen. Das vorgeschlagene Modell ermöglicht es Unternehmen, ihre Kubernetes-Umgebung strukturiert, verständlich und transparent zu entwerfen. Zur Bewertung des ersten Entwurfs des Modells wurden zwei Expertenworkshops und sechs Interviews durchgeführt. Mittels qualitativer Inhaltsanalyse wurden die daraus resultierenden Protokolle analysiert und die Ergebnisse in das Modell aufgenommen.

ABSTRACT

Being productively used by more than 77% of all IT-based companies in the industry, Kubernetes is the biggest players in the container orchestration market. Most companies report enormous cost savings and an increase in development speed by embracing Kubernetes. The extend of improvement, however, depends on how good the Kubernetes system fits the company and its requirements. Kubernetes consists of many components and can be used or consumed in diverse ways. To assist with designing the proper Kubernetes solution for a given idea or concept, a model consisting of a design process and a requirement catalog was proposed. The process is heavily relying on the requirement catalog, a questionaire that assists in gathering necessary requirements for the future cluster. It is split into seven steps, starting at the urge to find a proper platform. The other steps are focused on identifying requirements, designing the cluster, finding proper products and incorporating the Kubernetes system in the company. The requirement catalog is split in three groups, specifying 65 points in total. The proposed model enables companies to design their Kubernetes environment in a structured, comprehensible, and transparent manner. Two expert workshops and six interviews were conducted to evaluate the first draft of the process and questionaire. Using qualitative content analysis the resulting protocols were analyzed and the results were included in the model.

INHALTSVERZEICHNIS

1 EINLEITUNG

Dieses Kapitel beschreibt die Ausgangssituation für die vorliegende Masterarbeit und zeigt die Ziele und Forschungsfragen auf. Die Problemstellung ergibt sich aus der vorherrschenden Ausgangssituation, sowie Beobachtungen und Erfahrungen aus der betrieblichen Praxis. Die Ziele wurden direkt von der Problemstellung abgeleitet. Die Forschungsfragen wiederum wurden direkt aus den Zielen und Nicht-Zielen heraus entwickelt.

1.1 Ausgangssituation und Problemstellung

Anstoß für die Entstehung dieser Arbeit ist das Thema Innovation in Unternehmen welche selbst IT Applikationen entwickeln oder betreiben. Eine Innovation kann entweder die Erschaffung neuer Applikationen und Lösungen, oder die Verbesserung von bestehenden Systemen sein. Viele Innovationen der vergangenen Jahre kommen zwar inhaltlich nicht aus dem IT Sektor, wurden jedoch basierend auf Informationstechnologien umgesetzt. Als Beispiel seien an dieser Stelle das Unternehmen AirBnB angeführt. Dieses Unternehmen hat die private Vermittlung von Übernachtungsmöglichkeiten, vor allem in Großstätten, revolutioniert. Das Modell basiert im Endeffekt auf einer App und einer Backend-Software, welche den Service ermöglicht. AirBnB ist nur eines von vielen Unternehmen welches in den letzten Jahren auf die Verwendung einer neuen Technologie namens Container, sowie aufbauend darauf, auf Kubernetes gesetzt hat. Laut einer Studie von 2019 verwenden bereits 78% der Unternehmen im Software und Technologie Sektor Kubernetes in produktiven Umgebungen.[1]
Grundlegend ist Kubernetes eine Plattform, für den Betrieb und die Entwicklung von Anwendungen, welche enorme Optimierungs-, Einsparungs- und Erweiterungspotenziale bietet. Technisch gesehen besteht Kubernetes aus diversen Einzelkomponenten, wobei die Herausforderung darin liegt, das System richtig zu designen. Es geht darum für die aktuell umzusetzende Innovation die richtige Kombination der Kubernetes Komponenten zu finden, um das optimal passende System zur Verfügung stellen zu können.

Da jede Komponente einen spezifischen Zweck erfüllt und die Schnittstellen definiert und frei zugänglich sind, gibt es eine sehr breite Landschaft an Herstellern und Open-Source Projekten. Die Cloud Native Computing Foundation (CNCF) Cloud Native Interactive Landscape, ersichtlich in Abbildung 1.1, ist eine Sammlung von Produkten und Technologien im Umfeld von Kubernetes. Sie soll die enorme Auswahl an Produkten verdeutlichen, welche das Design von Kubernetes Clustern zu einer Herausforderung macht. Das in der Praxis beobachtete Problem ist, dass ohne eine strukturierte Vorgehensweise, sowie konkrete Anforderungen an das zu entwerfende System, ein optimales Design nur schwer umsetzbar ist. Optimal bedeutet in diesem Zusammenhang, dass das Kubernetes System schon in erster Instanz auf die Anforderungen der Innovation zugeschnitten ist. Ist dies nicht der Fall, steigt die Wahrscheinlichkeit, dass Teile oder das gesamte System im Laufe der Zeit geändert oder ausgetauscht werden müssen, was zu Zeit- und Ressourcenverlusten führen wird. Auch eine generelle Verzögerung des Designstarts wurde ohne konkrete Vorgehensweise und Anforderungen beobachtet. Da speziell das Innovationsmanagement die Idee vom Ursprung bis zur Marktreife begleitet, ist es hier als Koordinator und Treiber des Designprozesses zu sehen. Die Verwendung von Kubernetes und angrenzenden Technologien, fällt in den strategischen Bereich des F&E Managements, welcher ebenfalls meist stark mit dem Innovationsmanagement zusammenhängt. Ein weiterer Grund für die Beteiligung des Innovationsmanagements am Designprozess ist außerdem, dass das vorliegende Problem weit über eine technische Fragestellung hinaus geht. Je gesamtheitlicher die Anforderungen an das Zielsystem betrachtet werden, desto erfolgversprechender ist der Designprozess. Die sowohl strategische, als auch disziplinübergreifende Natur der Problemstellung, sowie der frühe Berührungskontakt mit der Innovation an sich, machen diese Aufgabe zu einem Innovationsproblem.

[1] Vgl. McMahon (2020), Onlinequelle [06.12.2020].

Die vorliegende Herausforderung im Unternehmen ist also der strukturierte Entwurf einer Kubernetes Plattform für eine konkret vorliegende Idee, bzw. eine bereits umgesetzte Innovation. Abgeleitet davon geht es in dieser Arbeit darum, eine strukturierte Vorgehensweise vorzuschlagen, die Unternehmen dabei hilft Kubernetes Cluster schneller und fokussierter designen zu können. Als Basis dafür muss eine konkrete Idee oder fertige Innovation vorliegen, welche die Anforderungen an das zukünftige System vorgibt.

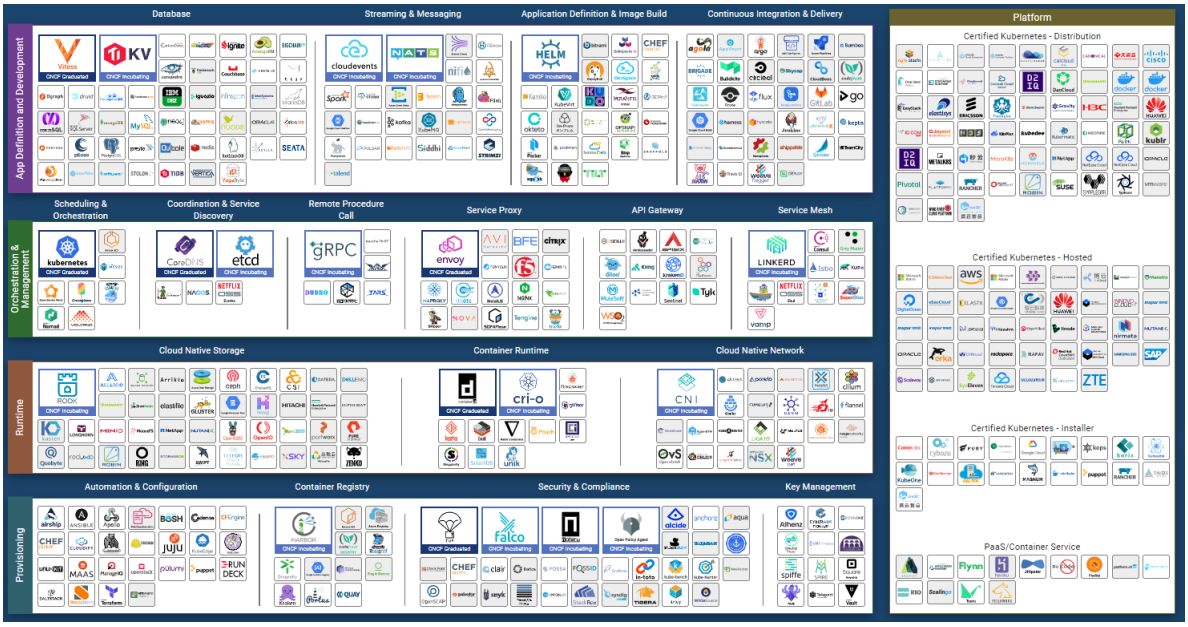


Abb. 1.1: **CNCF Container Landscape**,
Quelle: The Linux Foundation (2020i), Onlinequelle [02.03.2020]

Aufgabenstellung

- Theorie:
  - Erarbeiten einer strukturierten Vorgehensweise zum Design eines Kubernetes Clusters
  - Erhebung aller relevanten Komponenten eines Kubernetes Clusters
  - Erarbeitung eines Fragebogens zur Spezifikation von Anforderungen an ein neues Kubernetes System
- Praxis:
  - Validierung und Erweiterung des entworfenen Modells in einem Expertenworkshop
  - Validierung und Erweiterung des entworfenen Modells mittels Experteninterviews
  - Anwendung des Modells auf ein praktisches Beispiel

1.2 Zielsetzung und Forschungsfrage

Forschungsfragen

Basierend auf einer Innovation oder vorliegenden Idee, wie könnte eine strukturierte Vorgehensweise für das Design eines Kubernetes Clusters aussehen?

Welche Aspekte muss ein Fragebogen zur innovationsgetriebenen Spezifikation von Anforderungen an einen Kubernetes Cluster berücksichtigen?

Zielsetzung

- Ein Vorschlag für die strukturierte Vorgehensweise zum Design eines Kubernetes Clusters liegt vor.
- Ein Fragenkatalog zur Spezifikation von Anforderungen an ein Kubernetes System liegt vor.
- Der Anforderungskatalog ist durch einen Expertenworkshop, sowie durch Experteninterviews validiert und ggf. erweitert worden.
- Anforderungen an einen Cluster, basierend auf einer praktischen Problemstellung, sind mit dem Unternehmen formuliert.

Nicht-Ziele

- Eine vollständige Marktanalyse für Produkte pro Kubernetes Komponente wird durchgeführt.
- Ein konkreter Cluster wird entworfen und Produkte werden ausgewählt.
- Ein "Konfigurations-Tool"wird implementiert.

1.3 Aufbau der Arbeit

Die Arbeit ist grob in fünf Kapitel aufgeteilt. Das erste davon behandelt die Themen Container Technologie und Kubernetes von technischer Seite, zeigt die historische Entwicklung auf und gibt Auskunft über gängige Use-Cases von Kubernetes. Außerdem werden einige relevante Entwicklungen aufgezeigt, welche für das Verständnis des aktuellen Marktes interessant sind. Im zweiten Abschnitt wird konkret auf die Komponenten von Kubernetes eingegangen. Es wird beschrieben welche Aufgabe diese Bausteine haben und wie sie funktionieren. Das dritte Kapitel erklärt das vorgeschlagene Modell aus Designprozess und Anforderungsbogen. Es wird aufgezeigt wo das Thema im Kontext von Innovationsmanagement angesiedelt ist und in welchem Bezug es steht. Der entwickelte Prozess wird aufgezeigt und detailliert beschrieben. Außerdem wird der vorgeschlagene Anforderungsbogen vorgestellt und die Teile dessen genau beschrieben. Im vierten Kapitel geht es darum wie das initial erarbeitete Modell validiert und erweitert wurde. Es wurden Experteninterviews und Workshops durchgeführt und ausgewertet. In diesem Kapitel wird beschrieben welche Methodik verwendet wurde, wie die Validierungen durchgeführt wurden und welche Ergebnisse zutage gekommen sind. Abschließend wird die Arbeit im letzten Kapitel nochmal zusammengefasst, ein Fazit gezogen und die Forschungsfrage beantwortet.

1.4 Grafischer Bezugsrahmen

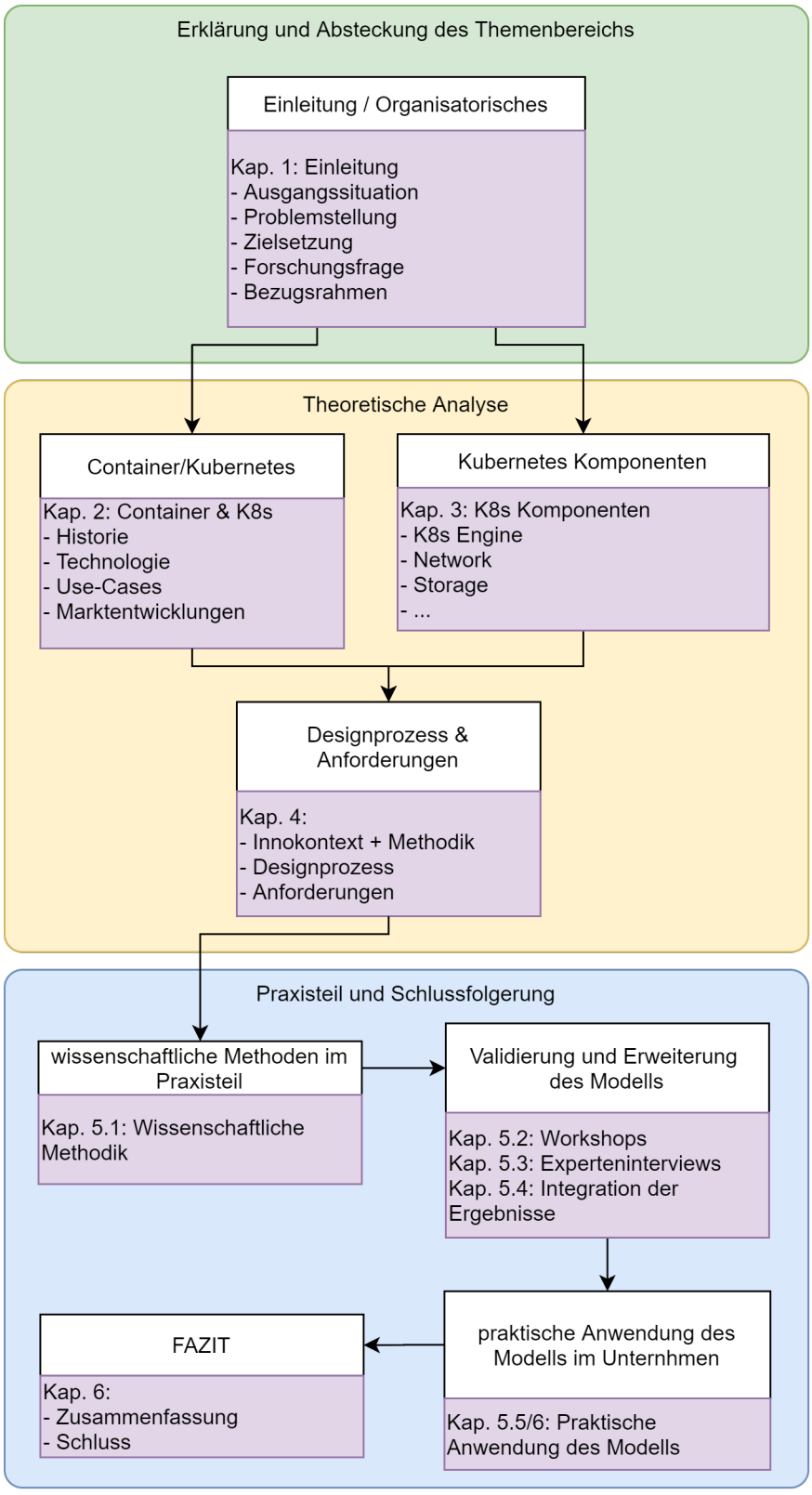


Abb. 1.2: **Grafischer Bezugsrahmen**,
Quelle: Eigene Darstellung

2 EINFÜHRUNG IN KUBERNETES UND DIE CONTAINER TECHNOLOGIE

Um verstehen zu können was Kubernetes ist, wie das System funktioniert und welche Hürden es beim Design eines optimalen Systems gibt, muss man erst verstehen welche Möglichkeiten Kubernetes bietet. Grundsätzlich versteht man unter Kubernetes ein Softwareprojekt, welches unter anderem von Google Inc., Red Hat Inc. und vielen weiteren Beitragenden entwickelt, und von der CNCF gehostet wird. Die offizielle Dokumentation beschreibt Kubernetes folgendermaßen: "Kubernetes (K8s) is an Open-Source system for automating deployment, scaling, and management of containerized applications".[2] Vereinfacht ausgedrückt bedeutet das, dass Kubernetes die Betreuung bzw. den Betrieb, die Skalierung sowie das Ausrollen von Applikationen ermöglicht. Mit Kubernetes können diverse unterschiedliche Applikationen auf dieselbe Art und auf derselben Hardware automatisiert betrieben werden. Die genaue Funktionsweise von Kubernetes, sowie dessen Vorteile und Anwendungsmöglichkeiten werden in den folgenden Kapiteln erklärt.

Als Beispiel dafür, wie Kubernetes Innovationen ermöglicht uns sich auf den Unternehmenserfolg auswirken kann, sei hier JD.com, Inc. angeführt. JD.com, Inc. ist eines der größten Online-Handelsunternehmen Chinas, welches 2019 einen Jahresumsatz von ca. 82 Mrd. Euro erwirtschaftete. Durch die Einführung von Kubernetes konnte JD.com die Zeit für das Ausrollen neuer Software von mehreren Stunden auf Sekunden verkürzen. Die IT-Kosten konnten bisher um 20-30% gekürzt werden, mit der Prognose für weitere Einsparungen im Bereich mehrerer hundert Millionen Euro in den kommenden Jahren. Im Jahr 2018 wurde eines der größten Shopping-Events des Unternehmens vollständig auf einer Kubernetes Umgebung abgewickelt. Alleine an diesem Event wurden über den Zeitraum von 11 Tagen Transaktionen im Wert von 23 Milliarden Euro durchgeführt.[3]
Kubernetes ist also eine Plattform, die basierend auf anderen Technologien, eine enorme Steigerung von Effizienz und Effektivität bewirken kann. Dabei sind sowohl der Betrieb, als auch die Entwicklung und Verbreitung von Applikationen und Systemen miteinbezogen.

2.1 Container Technologie

Das Kubernetes Projekt ist untrennbar an eine Technologie gekoppelt, die sich Container nennt. Sie sind die Basistechnologie hinter Kubernetes und der Grund dafür, dass man ein solches System überhaupt benötigt. Container bieten die Möglichkeit Software mit allen ihren Abhängigkeiten standardisiert zu paketieren, zu verteilen und auf beliebigen Systemen zu betreiben. Man kann Container auf den ersten Blick mit virtuellen Maschinen vergleichen. Diese haben das Ziel Hardware zu virtualisieren auf denen dann beliebige Betriebssysteme installiert werden können. Container hingegen sind ein Betriebssystem-Feature das es ermöglicht Applikationen isoliert voneinander zu betreiben.

[2] Vgl. The Linux Foundation (2020q), Onlinequelle [06.12.2020].
[3] Vgl. The Linux Foundation (2020e), Onlinequelle [06.12.2020].

Container bieten unter anderem in folgenden Bereichen große Möglichkeiten für Verbesserungen:

- Installation von Software
- Security
- Betrieb von Software unter konsistenten Bedingungen
- Optimale Ausnutzung der Hardware
- Entwicklung von Microservices
- Paketierung von Software, unabhängig von der Art der Software
- Realisierung des DevOps und CI/CD Paradigmas

Viele Probleme und Schwachstellen in der heutigen IT Welt können mit Hilfe von Containern behoben werden, angefangen bei der Installation von Software. Da Programme, vor allem große Enterprise Systeme, oft sehr viele Konfigurationsmöglichkeiten bieten und ein genau spezifiziertes Basissystem voraussetzen, ist die Installation oft sehr komplex. Außerdem treten häufig Probleme auf, wenn schon vorher Software auf einem System war und nicht sauber bereinigt wurde. Das jedoch größte Problem ist, dass die meiste Software ihre eigenen Abhängigkeiten zu Drittsoftware, sogenannten *Libraries*, hat. Möchte man nun zwei Programme gleichzeitig auf einem System betreiben, so muss sichergestellt sein, dass die Library für beide Programme in den korrekten Versionen vorliegt. Dies führt spätestens dann zu Konflikten, wenn beide Programme die gleiche Library in unterschiedlichen Versionen benötigen.[4] Mit Hilfe von Containern wird jeder Applikation eine gekapselte Umgebung zur Verfügung gestellt, in welcher die benötigten Libraries installiert werden können. Dies hat keinen Effekt auf andere Applikationen, bzw. Container, und ermöglicht somit den parallelen Betrieb auf einem Host.

Neben der Installation ist auch das Thema Security ein Problem das häufig auftritt und oft nur wenig beachtet oder schlecht behandelt wird. Für das große Gebiet Security sind hier zwei Beispiele angeführt, bei denen Container einen enormen Vorteil bieten: Isolation und Updates. Ist beispielsweise eine Applikation von Schadsoftware befallen oder wurde von Hackern infiltriert, so ist durch die gegenseitigen Isolation von Containern sichergestellt, dass das Problem eingedämmt wird. Wird bei Software eine Sicherheitslücke geschlossen, so muss bei konventionellen Programmen meist das aktuell installierte System verändert werden. Ist die Software hingegen in einem Container verpackt, so kann einfach eine neue Version des gesamten Containers erstellt, verteilt und installiert werden.[5]
Die Möglichkeit ganze Container, anstatt einzelner spezifischer Updates, zu verteilen ist ein enormer Vorteil gegenüber herkömmlich installierter Software. Mit dieser Technik ist es ein Leichtes fertig verwendbare Softwarepakete, wie zum Beispiel Webserver, Datenbanken oder Laufzeitumgebungen, zu verteilen. Hierzu werden einfach die benötigten Vorlagen für die Container, sogenannte *Images*, am Zielsystem heruntergeladen und gestartet.

Der Umstand, dass Container isoliert laufende Softwarepakete mit definiertem Inhalt sind, hat einen weiteren entscheidenden Vorteil: es liegt eine konsistente Umgebung für die jeweilige Applikation vor. Das bedeutet, dass egal auf welchem System oder Betriebssystem ein Container läuft, die Umgebung aus Sicht der Applikation im Container unverändert bleibt. Somit muss sich ein Entwickler keine Gedanken mehr um die zugrunde liegende Umgebung machen, sondern kann sich absolut auf die Applikation konzentrieren. Werden spezielle Komponenten oder Libraries benötigt, so können diese einfach in den Container gepackt werden und stehen immer zur Verfügung. Dieser Faktor ist zum Beispiel auch beim Onboarding neuer Mitarbeiter von Vorteil, wenn dem neuen Kollegen eine standardisierte Umgebung, basierend auf Containern, bereitgestellt wird. Es entfallen mühsame und fehleranfällige manuelle Installationen, und der neue Kollege kann sofort in einer identischen Umgebung arbeiten wie alle anderen Entwickler.[6]

[4] Vgl. Bhat (2018), S. 2f.
[5] Vgl. Nickoloff (2019), S. 11f.
[6] Vgl. Mouat (2015), S. 3f.

Da Container meist nur einzelne Applikationen beinhalten die voneinander isoliert laufen und die Ressourcen des Betriebssystems nutzen, erreichen sie eine höhere Auslastung der zugrunde liegenden Hardware als Virtuelle Maschinen. Um dieselbe Isolation von Applikationen zu erreichen müsste pro Applikation eine Virtuelle Maschine angelegt werden. Dies würde sowohl einen höheren Verbrauch an Hardware Ressourcen, als auch an administrativen Aufwand bedeuten. Aus diesem Grund werden Container meist bevorzugt, sofern eine Software die Verwendung von Containern auch zulässt.[7]

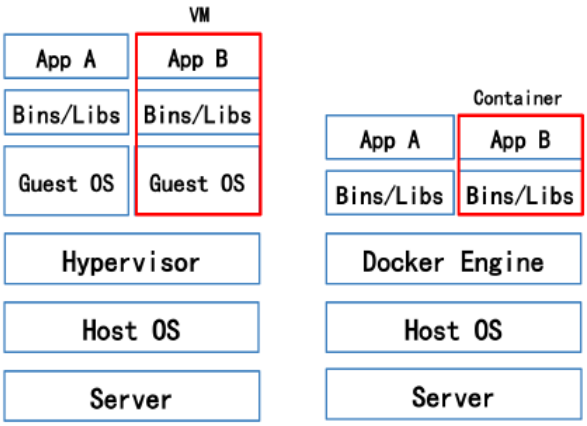


Abb. 2.1: **Architektur von Container vs. Virtuelle Maschine**
Quelle: Zhang (2018), Fig. 1.

Wie bereits erwähnt handelt es sich bei Containern um eine Art von paketierten Applikationen. Natürlich bieten diverse Plattformen und Programmiersprachen ihre eigenen Paketformate, diese haben jedoch den Nachteil, dass sie meist nur für eine spezielle Technologie verwendbar sind. Container sind im Gegensatz zu anderen Paketformaten komplett agnostisch im Bezug auf ihren Inhalt. Ein Container kann sowohl eine Datenbank, einen Applikationsserver oder ein einfaches Script enthalten. Die Sprache, beziehungsweise die Art und Weise wie ein Container entsteht ist unabhängig davon was sich darin befindet. Darum sind Container sehr flexibel und verändern die Art wie Software heute paketiert und verteilt wird.[8]

Einer der Hauptverwendungszwecke für Container ist die Entwicklung sogenannter Microservices. Dabei handelt es sich um meist kleine Programme die über eine definierte Schnittstelle einen Service zur Verfügung stellen. Microservices haben immer einen sehr spezifischen Zweck und werden daher so schlank wie möglich gehalten. Mittels Containern ist es einerseits möglich neue optimierte Microservices zu entwickeln, aber auch bestehende monolithische Applikationen zu betreiben. Bestenfalls ist es sogar möglich alte monolithische Applikationen in ihre Bestandteile aufzubrechen und durch mehrere Microservices zu ersetzen. Der Vorteil von Microservices, im Vergleich zu monolithischen Applikationen, ist, dass sie bei erhöhter Last mehrfach parallel ausgeführt werden können. Das bedeutet, dass zum Beispiel die Zugriffe auf eine Website auf mehrere identische Microservices verteilt werden können. Diese Microservices wiederum können auf verschiedenen Rechnern laufen, was man horizontale Skalierung nennt. Monolithischen Applikationen hingegen kann meist nur mehr Ressourcen pro Rechner zur Verfügung gestellt werden um mehr Last verarbeiten zu können. In diesem Fall spricht man von vertikaler Skalierung. Dies führt zu höheren Kosten und weniger Flexibilität als bei horizontaler Skalierung.[9]

[7] Vgl. Zhang (2018), Onlinequelle [06.12.2020].
[8] Vgl. Miell (2019), S. 3f.
[9] Vgl. Luksa (2018), S. 3ff.

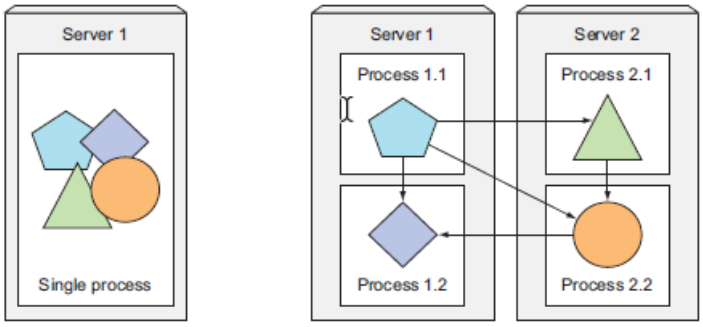


Abb. 2.2: **Architektur von monolithischer vs. Microservice Applikation**
Quelle: Luksa (2018), S. 3

Der letzte angeführte Vorteil von Containern ist, dass sie sich sehr gut in das DevOps und Continuous Integration/Continuous Delivery (CI/CD) Paradigma einfügen. Darunter versteht man einerseits die vollautomatische Erstellung und Testung (Continuous Integration) sowie die vollautomatische Auslieferung (Continuous Delivery) von Software. Der Entwickler muss seinen Code nur mehr in eine zentrale Code Datenbank, Repository genannt, geben. Anschließend wird ein Automatismus angestoßen an dessen Ende die Applikation am gewünschten Zielsystem installiert und gestartet wird. Zum Anderen geht es darum, dass sich Entwickler auch nach der eigentlichen Entwicklung um ihre Software kümmern müssen, sprich diese betreiben. Dabei spricht man von *DevOps*, einem Modewort, welches sich aus *Developer* und *Operations* zusammensetzt. Die einzelnen Teilbereiche des *DevOps* Paradigmas sind im sogenannten *DevOps Cycle* in Abbildung 2.3 dargestellt.

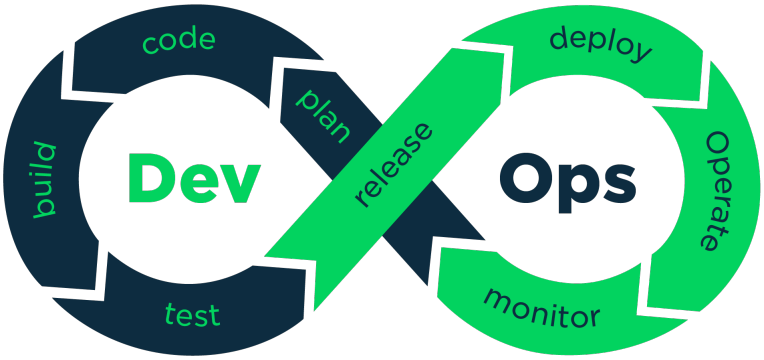


Abb. 2.3: **DevOps Cycle: Teilgebiete von DevOps**
Quelle: Timms (2018)

Container sind sowohl bei der vollautomatischen Erstellung, Testung und Auslieferung, als auch beim fortlaufenden Betrieb ein mächtiges Werkzeug. Durch die Isolation und das gleichbleibende Umfeld im Container lassen sich sowohl Automatisierung als auch Betrieb und Fehlersuche maßgeblich vereinfachen. Vor allem die Tatsache, dass sich Container immer und überall gleich verhalten, spielt bei Entwicklung und Betrieb eine große Rolle.[10]

Alle bisher genannten Punkte beziehen sich immer auf Container alleine. Dafür würde man Kubernetes noch nicht benötigen. Kubernetes kommt ins Spiel, wenn man Container verteilt auf diversen Rechnern betreiben und verwalten will. Dabei kommen zusätzliche Aufgaben wie Lastverteilung, Scheduling, Überwachung, Routing oder Security zu tragen. Alle diese Punkte werden im Kapitel 2.4 ausgiebig behandelt An diesem Punkt ist lediglich wichtig zu verstehen, dass Kubernetes Container in Rechnerverbänden, sogenannten *Clustern*, verwaltet und ihnen diverse Funktionen und Services zur Verfügung stellt.

[10] Vgl. Arundel (2019), S. 3ff.

2.2 Historie der Container-Technologie

Bevor weiter auf Kubernetes eingegangen wird, ist es nützlich einen Überblick über die historische Entwicklung der Container Technologie und deren Umfeld zu erhalten. Mit dem Wissen über die Entwicklung dieses Technologiezweiges lassen sich historische Probleme und etwaige Zusammenhänge besser verstehen und nachvollziehen. Dieses Kapitel gibt Auskunft darüber wie sich die Container-Technologie, wie wir sie heute kennen, über die vergangenen Jahrzehnte entwickelt hat. Abschließend wird beleuchtet, wie sich aus, und parallel zur Container-Technologie, die dazugehörigen Verwaltungssysteme entwickelt haben.

2.2.1 Container-Technologien

Wie einleitend schon erklärt sind Container ein Betriebssystem-Feature, genauer gesagt eine Funktionalität im Linux Kernel. Die Ursprünge von Containern, bzw. erste Implementierungen der Idee von Applikationsisolierung, gehen zurück bis in die 1970er Jahre. Erstmals im Jahr 1979 war es im UNIX V7 Betriebssystem möglich mit dem *chroot* Kommando eine gewisse Isolation von Programmen zu realisieren. Dieses Kommando legt den Ursprung des Dateisystems, wie ihn ein Prozess inklusive seiner Subprozesse sieht, auf einen bestimmten Pfad. Somit können zumindest auf Dateisystem Ebene Programme voneinander isoliert werden. Diese Technologie wurde 1982 in das BSD UNIX Derivat eingeführt.[11]

Für das FreeBSD Betriebssystem wurde im Jahr 2000 das sogenannte *Jails* Feature eingefügt. Dabei handelt es sich um eine funktionale Erweiterung von *chroot*, welche es ermöglicht nicht nur das Dateisystem mehrerer Prozesse zu isolieren. Mit *Jails* kann auch eine Isolation auf Netzwerk und Benutzer Ebene umgesetzt werden. Das bedeutet, dass jeder Jail seinen eigenen Netzwerk-Stack und eigene Benutzer hat.[12]

Erstmals im Jahr 2004 wurde der Begriff *Container* von Sun Microsystems für ein Feature des Solaris 10 Betriebssystems verwendet. Solaris Container ermöglichten es erstmals sowohl System Ressourcen aufzuteilen, als auch Isolation von Applikationen sicherzustellen. Zusätzlich waren noch Funktionalitäten wie Snapshots oder Duplikation von Containern möglich.[13,14]

OpenVZ ermöglichte ab 2005 ein ähnliches Set an Funktionalität wie Solaris Container. Allerdings war OpenVZ eine spezifische Erweiterung des Linux Kernels. OpenVZ ermöglichte es auf einer Hardware mehrere virtuelle Betriebssysteme zu betreiben, in diesem Kontext auch *Container* genannt. Es ist hervorzuheben, dass OpenVZ Container ein ganzes Betriebssystem virtualisieren, und sich darin von Containern aus heutiger Sicht unterscheiden. Aus heutiger Sicht versteht man unter Containern, die Kapselung einzelner Applikationen, nicht die Virtualisierung gesamter Betriebssysteme. Jeder Container hatte sein eigenes Dateisystem, eigene Benutzer, einen eigenen Prozessbaum und eine eigene Netzwerk Schnittstelle.[15]

Ursprünglich als *process containers* entstanden, hat Google 2007 das *Control Group (CGroup)*, Feature zum Linux Kernel beigesteuert. Dabei handelt es sich um eine Funktionalität die es erlaubt die Ressourcen von bestimmten Prozessgruppen einzuschränken. *CGroups* sind bis heute eine der Basistechnologien hinter aktuellen Container Technologien.[16]

Unmittelbar nachdem *CGroups* Teil des Linux Kernels wurden, entstand das LXC Projekt, welches direkt auf *CGroups* und sogenannten *LinuxNamespaces* beruht. Das Akronym LXC steht für*LinuX Containers*. Das besondere an LXC ist, dass es alle Features von Containern ohne jegliche Kernel Patches oder ähnliches liefert. LXC ist somit die erste Implementierung einer vollständigen Containertechnologie, zur Isolation von Applikationen, wie wir sie heute kennen.[17]

[11] Vgl. Osnat (2020), Onlinequelle [06.12.2020].
[12] Vgl. Gunaratne (2016), Onlinequelle [06.12.2020].
[13] Vgl. Osnat (2020), Onlinequelle [06.12.2020].
[14] Vgl. Kane (2018), S. 61.
[15] Vgl. Bhat (2018), S. 4f.
[16] Vgl. Bhat (2018), S. 5.
[17] Mouat (2015), S. 6.

Der wahrscheinlich entscheidende Schritt für den Erfolg von Containern war die Veröffentlichung der Docker Plattform von Docker Inc. im März 2013. Die Docker Plattform baute damals auf LXC als Container-Technologie auf, und erweiterte die reine Laufzeitumgebung um essenzielle Features. Die Möglichkeit Abbilder von Containern, sogenannte *Images*, zu erstellen und diese zu verteilen, eine einheitliche Benutzerschnittstelle, sowie ein zentrales Image-Repository namens *DockerHub* sind einige der Kernfeatures von Docker. Durch diese Sammlung von Container-Technologie und Supportfeatures, kombiniert mit dem Open-Source veröffentlichten Quellcode von Docker, wurden Container für einen großen Nutzerkreis interessant.[18]

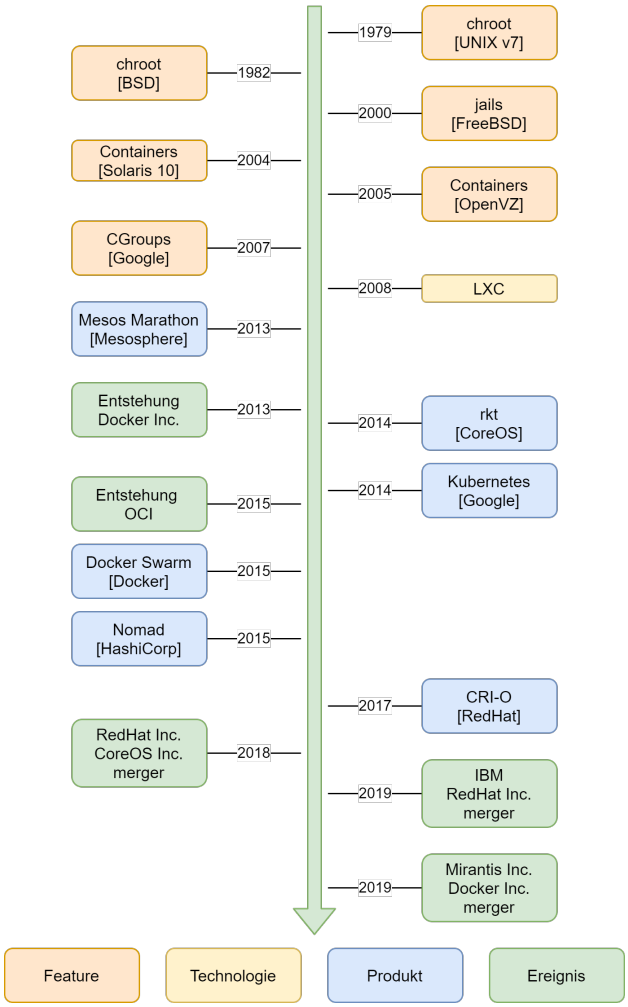


Abb. 2.4: **Zeitleiste wichtiger Ereignisse im Container Umfeld**
Quelle: Eigene Darstellung

Im November 2014 wurde vom Unternehmen CoreOS Inc. die Container-Plattform rkt (*"rock-it"*) vorgestellt. Diese Plattform verfolgt im Grunde dieselben Ziele wie Docker und ist auch imstande Docker Container zu betreiben, verwendet selbst jedoch ein anderes Container Image Format. Die unterschiedlichen Formate für*Images* haben sich jedoch schnell als ein Problem für die Container-Community herausgestellt, da man sich für eine Seite entscheiden musste. Aus diesem Problem ist 2015 die Open Container Initiative (OCI) hervorgegangen. Die OCI ist ein Projekt, gegründet unter anderem von Docker Inc. und CoreOS Inc., welches gewisse Standards im Containerumfeld hervorbringt.[19]

[18] Vgl. Kane (2018), S. 1ff.
[19] weitere Informationen unter: https://opencontainers.org/about/overview/

Bisher hat die OCI zwei Standards, nämlich für das Imageformat [20] sowie für die Containerlaufzeitumgebung [21] veröffentlicht.[22]
Am zweiten Januar 2017[23] wurde das Projekt *CRI-O* veröffentlicht. Dabei handelt es sich um ein von RedHat gesponsertes Open-Source Projekt, welches eine reine OCI und Container Runtime Interface (CRI) konforme Container Laufzeitumgebung (*Container-Runtime*), siehe Kapitel 2.3.5, entwickelt. Diese *Container-Runtime* ist speziell auf Kubernetes zugeschnitten und optimiert.[24]
Seit 2017 wurden viele der erwähnten Projekte und Technologien stark weiterentwickelt und verbessert. Es gab etliche Start-Up Gründungen und neuen Produkte welche sich um die einzelnen Komponenten einer Container Landschaft drehen. Technologisch sind seither jedoch keine signifikanten Sprünge mehr geschehen. Dies kann unter anderem an der vorherrschenden COVID-19 Pandemie, sowie an diversen M&A Deals im Container-Markt festgemacht werden.

2.2.2 Container-Orchestration

Die bisher erwähnten Projekte bzw. Produkte beschränken sich auf die Erstellung, Verbreitung und den Betrieb von Containern auf einzelnen Rechnern. Um jedoch eine Container-Landschaft im großen Stil zu ermöglichen, bedarf es eines sogenannten *Container-Orchestrators*. Dabei handelt es sich um ein Tool, welches den koordinierten Betrieb von Containern auf mehreren verbundenen Hosts ermöglicht, sowie einige wertvolle Features zur Verfügung stellt. Der Begriff Container-Orchestration bezeichnet das verwalten von Containern über mehrere Rechner hinweg. In Abbildung 2.5 ist schematisch der Unterschied zwischen einem einzelnen Container-Host, und mehreren Hosts, verwaltet durch einen *Container-Orchestrator* dargestellt. Der *Container-Orchestrator* würde beispielsweise das Verschieben eines Containers, in diesem Fall C9, veranlassen, falls Probleme am Host A auftreten. Ohne mehrere verbundene Container-Hosts und einen *Container-Orchestrator* wäre dies nicht möglich.

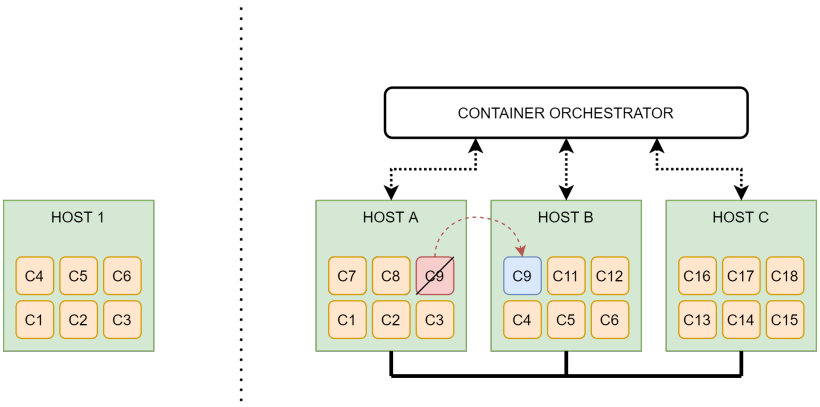


Abb. 2.5: **Schematische Darstellung: Container-Orchestrator**
Quelle: Eigene Darstellung

Eines der ersten Unternehmen welches sich intensiv mit dem Thema Container und Orchestration auseinander gesetzt hat, war Google. Nach einer Dekade der fortlaufenden Entwicklung ging 2014 Kubernetes aus seinen zwei Vorgängerprojekten BORG und Omega hervor. [25] Kubernetes war eines der ersten Projekte zum Thema Container-Orchestration und konnte sich auch auf lange Sicht als erfolgreichste Lösung etablieren.

[20] weitere Informationen unter: https://github.com/opencontainers/image-spec/releases
[21] weitere Informationen unter: https://github.com/opencontainers/runtime-spec/releases
[22] Vgl. Luksa (2018), S. 15f.
[23] weitere Informationen unter: https://github.com/cri-o/cri-o/releases/tag/v0.0.0
[24] Vgl. Luksa (2018), S. 555.
[25] Beda (2018), Onlinequelle [06.12.2020]; Burns (2016), Onlinequelle [06.12.2020].

Laut einer Umfrage der CNCF, wird Kubernetes in 78% der befragten Unternehmen bereits produktiv verwendet. Container generell werden von 84% der befragten Unternemhen verwendet.[26] Grundsätzlich gibt es viele Möglichkeiten Kubernetes zu verwenden. Die aufwändigste aber günstigste Variante ist *Upstream Kubernetes*, womit das reine Open-Source Kubernetes[27] gemeint ist. Der Begriff *Upstream* besagt, in diesem Kontext, dass es ein zentrales Projekt gibt, das eigentliche Kubernetes Projekt. Davon abgeleitet, bzw. darauf aufbauend, gibt es verschiedenste Kubernetes Distributionen.[28] Außerdem gibt es die Möglichkeit Kubernetes als Service von großen Public-Cloud-Providern zu beziehen. Hierzu würden beispielsweise Amazon ECS[29], Google GKE[30] oder Microsoft AKS[31] zählen. Wie in Abbildung 2.6 zu sehen ist, wird dieses Angebot auch sehr gut angenommen. Eine gerne

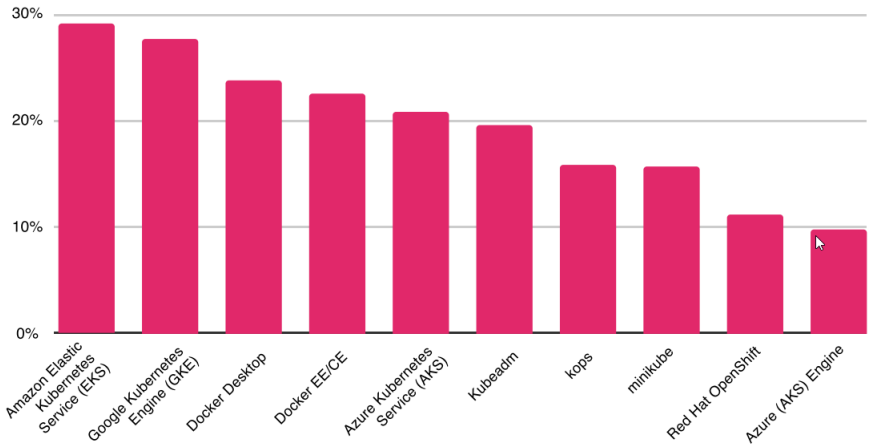


Abb. 2.6: **Verwendete Container Management Lösungen**
Quelle: McMahon (2020), Onlinequelle [25.03.2020]

gewählte Option bieten auch sogenannte *Managed-Services*, wobei der Cluster hier von einem Drittanbieter betrieben wird. Dieser kann dabei sowohl im Data Center des externen Partner, in einer Public-Cloud, oder auch im eigenen Data Center liegen. Die Variante, in der Software im eigenen Data Center läuft wird auch On-Premises genannt. Aktuell ist die Public-Cloud Variante, wie in Abbildung 2.7 zu sehen ist, noch die populärste, wobei sich der Trend hin zu hybriden Clouds entwickelt.

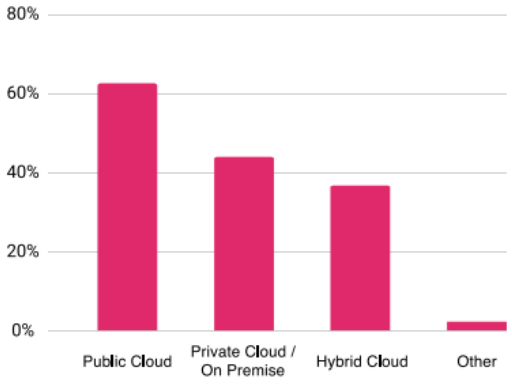


Abb. 2.7: **Verwendete Cloud Infrastrukturen**
Quelle: McMahon (2020), Onlinequelle [25.03.2020]

[26] Vgl. McMahon (2020), Onlinequelle [06.12.2020].
[27] weitere Informationen unter: https://github.com/kubernetes/kubernetes
[28] Vgl. Hombergs (2020), Onlinequelle [06.12.2020].
[29] weitere Informationen unter: https://aws.amazon.com/de/ecs/
[30] weitere Informationen unter: https://cloud.google.com/kubernetes-engine
[31] weitere Informationen unter: https://docs.microsoft.com/en-us/azure/aks/

Abschließend sei als Kubernetes Option auch noch explizit die *OpenShift* Distribution von RedHat Inc. erwähnt. Dabei handelt es sich um eine standardisierte Kubernetes Umgebung mit zusätzlichen integrierten Features, welche auf diversen Public-Cloud-Plattformen sowie On-Premises verwendet werden kann. OpenShift wird deshalb explizit erwähnt, weil es sich dabei um die einzige proprietäre Kubernetes Distribution handelt, welche dieselbe Umgebung, inklusive Support, für jede Installationsvariante bietet. Es ist also egal ob On-Premises, in einer zertifizierten Public-Cloud oder bei einem zertifizierten Managed-Service-Provider betrieben wird.
Neben Kubernetes haben sich jedoch auch noch andere Container-Orchestration-Tools entwickelt. In Abbildung 2.8 ist ersichtlich, dass jedoch schon 2018 Kubernetes den größten Marktanteil aller Container-Orchestration-Tools hatte.

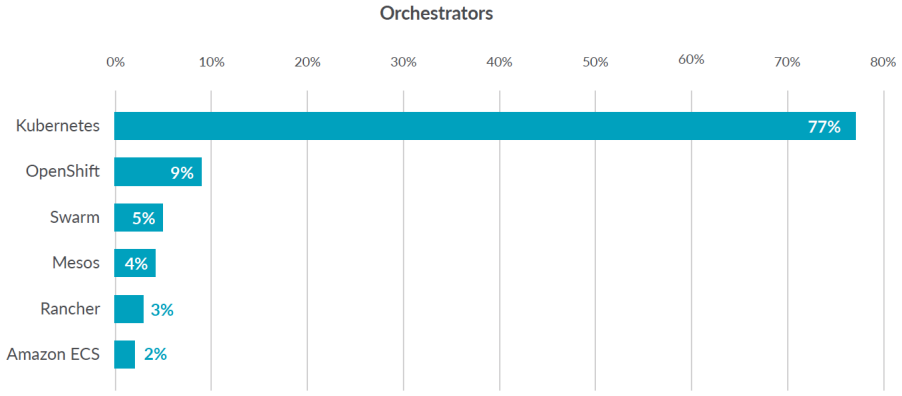


Abb. 2.8: **Container-Orchestration-Tool Marktverteilung, Oktober 2019**
Quelle: Carter (2019), Onlinequelle [10.12.2020]

Große Bekanntheit erlangte aufgrund der sehr einfachen Bedienbarkeit und Installation *Docker Swarm*, die größte Konkurrenz zu Kubernetes. Dabei handelt es sich um den Container-Orchestration-Tool von Docker Inc., welcher erstmals 2015 veröffentlicht wurde.[32] Seit der Version 1.12 ist Swarm fixer Bestandteil der Docker Engine.[33] Docker Swarm erlangte Anfangs große Beliebtheit, vor allem in der Entwicklungsphase von Applikationen, musste jedoch zunehmend Kubernetes weichen. Dies ist großteils auf die größere Community von Kubernetes zurückzuführen. Dennoch wird Docker Swarm auch noch weiterhin genutzt und weiterentwickelt[34].
Neben Kubernetes und Docker Swarm brachte auch HashiCorp Inc. ein Container-Orchestration-Tool namens *HashiCorp Nomad* auf den Markt. Nomad ist zwar nicht so verbreitet wie Kubernetes oder Docker Swarm, fügt sich jedoch perfekt in das breite und hochintegrierte Produktportfolio von HashiCorp Inc. ein. Neben Containern ist Nomad auch imstande VMs und bestehende Applikationen zu orchestrieren. HashiCorp Nomad wurde im September 2015 veröffentlicht und seither stetig weiterentwickelt.[35]

Die letzte seriöse Container Orchestartor Alternative zu Kubernetes stellt das *Apache Mesos* Projekt dar. Grundgedanke dieses Projektes ist es, ein gesamtes Datacenter nach außen als eine Ressource darzustellen und zu nutzen. Mesos setzt auf eine Agent-Master Architektur, wobei der Master sogenannte Frameworks zur Verfügung stellt, welche wiederum spezifische Applikations-Tasks auf den Agents ausführen können. Das Projekt Marathon[36] ist ein solches Framework, welches es ermöglicht mit Apache Mesos Container zu orchestrieren.[37] Apache Mesos ist für reine Container-Orchestration nicht die erste Wahl, auch rentiert sich der volle Funktionsumfang nur bei großen und heterogenen IT Landschaften.

[32] weitere Informationen unter: https://github.com/docker/swarm/releases?after=v0.2.0
[33] für Details siehe: https://docs.docker.com/engine/release-notes/prior-releases/swarm-mode-4
[34] Vgl. Bohn (2020), Onlinequelle [06.12.2020].
[35] weitere Informationen unter: https://github.com/hashicorp/nomad/releases/tag/v0.1.0
[36] weitere Informationen unter: https://github.com/mesosphere/marathon
[37] Vgl. Abdelrazik (2017), Onlinequelle [06.12.2020].

Daher ist der Marktanteil im Bereich Container-Orchestration vergleichsweise gering. Wie in Abbildung 2.9 ersichtlich, verwendeten bereits 2017 die meisten Umgebungen auf denen Container liefen auch Kubernetes.

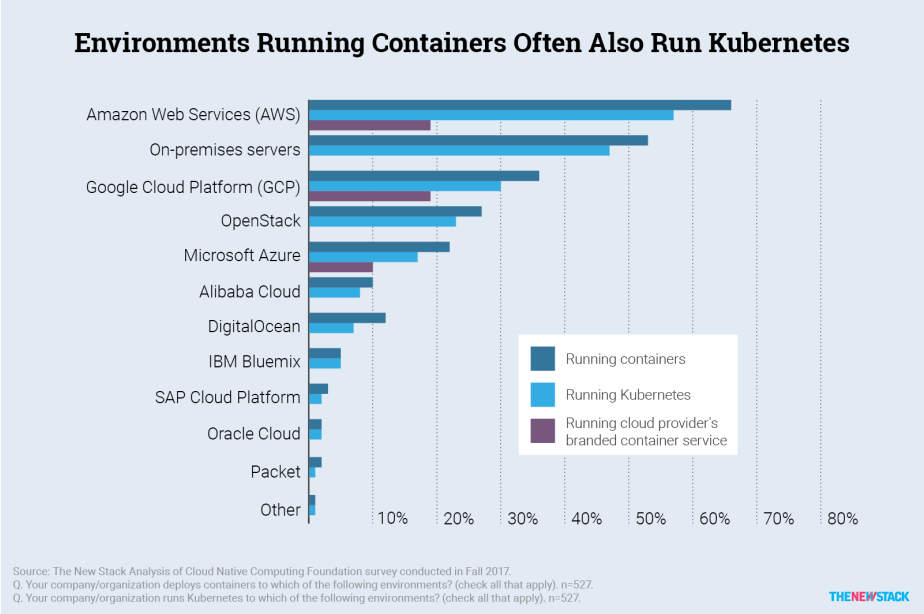


Abb. 2.9: **Container Umgebungen verwenden Kubernetes, 2017**
Quelle: Hecht (2018), Onlinequelle [06.08.2020]

In den vergangenen Jahren kam es am Container Markt noch zu einigen interessanten Firmenübernahmen, welche hier angeführt werden. Im Januar 2018 gab RedHat Inc. bekannt CoreOs Inc. zu kaufen[38], was folgende Auswirkungen hatte:

- Das *rkt* Projekt wurde eingestellt - letztes offizielles Release v1.3.0, am 16 April 2018[39]
- Das libpod Projekt, eine Alternative zu Docker und rkt, wurde am 12 Februar 2018 veröffentlicht.[40]
- libpod, bzw. dessen Aufbauprojekte Podman und buildah, ersetzen seither Docker auf den RedHat Betriebssystemen RHEL8 und CentOS [41]

Im Juli 2019 wurde RedHat Inc. von IBM für $34 Milliarden übernommen, was IBM schlagartig eine äußerst starke Position am Container Markt einbrachte. Die Kombination aus IBM Cloud und der Fachexpertise sowie den Produkten von RedHat, vereint mit dem gemeinsamen Kundenstamm der beiden Unternehmen, spricht für eine erfolgreiche Entwicklung.[42] Vier Monate später, im November 2019, wurde das Enterprise Segment von Docker Inc. von Mirantis Inc. gekauft. Mirantis Inc. ist ein seit 1999 agierender Provider von Softwarelösungen und Cloud Plattformen sowie einer der wichtigsten Befürworter der OpenStack Plattform. In den vergangenen Jahren entstanden auch unzählige Start-Ups auf dem Container Markt, welche jedoch von den bekannten Technologiekonzernen wie Google, IBM, Amazon und Microsoft überschattet werden.[43]

Dieses Kapitel befasste sich mit der Geschichte von Container, Container-Orchestration-Tools und Kubernetes im Speziellen. Für das Innovationsmanagement ist es interessant zu wissen woher Technologien kommen um deren Lebenszyklus abschätzen zu können.

[38] Vgl. MSV (2018), Onlinequelle [06.12.2020].
[39] weitere Informationen unter: https://github.com/rkt/rkt/releases/tag/v1.30.0
[40] weitere Informationen unter: https://github.com/containers/libpod/releases/tag/v0.2
[41] Red Hat Inc. (2019), Onlinequelle [06.12.2020]; Wallen (2020), Onlinequelle [06.12.2020].
[42] Armonk (2019), Onlinequelle [06.12.2020]; Lardinois (2019b), Onlinequelle [06.12.2020].
[43] Lardinois (2019a); Vaughan-Nichols (2019)

Außerdem ist es gut einen groben Überblick über den Markt zu haben, um etwaige Machtverhältnisse verstehen zu können. Damit lassen sich Produkt- bzw. Technologiestrategien sicherer und bedachter entwickeln und verfolgen.

2.3 Technische Funktionsweise von Containern

In diesem Kapitel geht es um die technische Funktionsweise von Containern. Das Wissen um die technische Funktionsweise von Containern ist wichtig, da spätestens bei der Formulierung von Anforderungen an einen Cluster auch ein gewisses Wissen über Container nötig ist. Des Weiteren ist es sehr nützlich für einen Innovationsmanager, speziell im IT Umfeld, zu verstehen wie die Container-Technologie grundsätzlich funktioniert. Besonders wenn es darum geht zu überlegen oder zu entscheiden, ob neue Innovationen mit den vorliegenden Technologien sinnvoll umgesetzt werden können oder nicht. Um Container verstehen zu können, müssen erst einige grundlegende Begriffe definiert und abgegrenzt werden.

Sehr kurz zusammengefasst sind *Container* gestartete Instanzen von *Images*. Ein *Image* kann man sich also als Template vorstellen, den *Container* als Objekt welches gemäß eines Templates erstellt wurde. Die Container laufen auf einem *Host* und benötigen eine *Container-Runtime* und ausgeführt werden zu können. Man könnte dies ungefähr mit der Verwendung einer mp3 Datei vergleichen. Man benötigt einen Mediaplayer (Host) um eine mp3 Datei (Image) abzuspielen und somit einen Song beliebig oft zu hören (Container). Die Verwaltung von *Container* Ressourcen wird mittels *CGroups* und *Namespaces* bewerkstelligt, welche wiederum von der *Container-Runtime* gesteuert werden.

2.3.1 Container Image und Layer

Unter einem *Image* versteht man die definierte Abfolge von Filesystem Operationen inklusive deren Parameter, gespeichert in einer Datei. Versteht man zum Beispiel ein leer aufgesetztes Betriebssystem (zB Ubuntu 18.04 LTS) als Basisimage, so würden alle Änderungen am Filesystem (also die neu hinzukommenden Dateien), bei der Installation eines Programms (zB eines Webservers), in Summe zu einem neuen *Image* führen. Dies ist schematisch in Abbildung A.1 dargestellt.
Jedes einzelne Kommando, welches eine Veränderung des Filesystems zur Folge hat, erzeugt in diesem Prozess einen neuen sogenannten *Layer*. Das resultierende *Image* ist somit ein inkrementeller Stapel von *Layern*. Wie ein gesamtes *Image* nach OCI Standard auszusehen hat, steht in der OCI *Image Format Specification* [44] beschrieben. Ebenso findet sich dort eine Beschreibung darüber, wie die einzelnen *Layer*[45] technisch auszusehen haben.

Die *Layer* eines Container-Images beziehen sich ausschließlich auf das Filesystem. An diesem Punkt geht es noch um keine Speicher-, Peripherie- oder Prozessorressourcen. Das Verhältnis von Layern untereinander ist in Richtung *Parent* eine strikte 1:1 Beziehung, in Richtung *Child* jedoch eine 1:n Beziehung. Das bedeutet, dass jeder Layer nur einen *Parent-Layer*, sprich einen Layer von dem er abgeleitet wird, haben kann. Von einem Layer können jedoch beliebig viele *Child-Layer* abgeleitet werden. Ein *Layer* entsteht nach jeder Veränderung des Filesystems. Technisch wird das so gelöst, dass der Layer dann gespeichert wird, wenn der Prozess welcher die Filesystem-Veränderung bewirkt, beendet ist. In einem Image sind alle Layer *readonly*, außer dem aktuell Letzten, dieser kann verändert werden. In Anhang A.8 ist ein Beispiel angeführt, welches den Aufbau von Layern eines Images verdeutlichen soll.

[44] weitere Informationen unter: https://github.com/opencontainers/image-spec/blob/master/spec.md
[45] weitere Informationen unter: https://github.com/opencontainers/image-spec/blob/master/layer.md

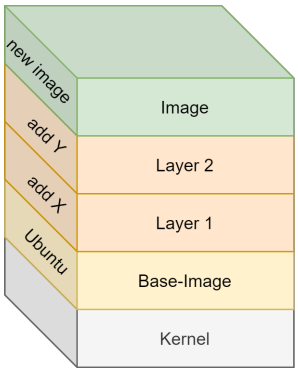


Abb. 2.10: **Schematische Darstellung von Image Layern**
Quelle: In Anlehnung an Pyasi (2020), Onlinequelle [06.08.2020]

Die Tatsache, dass sich Images bestimmte Layer teilen können, hat einen entscheidenden Vorteil. Dadurch, dass die Basislayer immer dieselben sind, müssen diese nur einmal pro Host vorhanden sein, und können dann von beliebigen Images verwendet werden. Wird nun im Container am Filesystem etwas verändert was sich in einem *readonly* Layer befindet, so wird dieser Speicherbereich in den aktuellsten Layer kopiert und dort verändert. Dadurch ändert sich der Zustand der Filesystems aus Sicht des ändernden Containers, der ursprüngliche Basislayer wird jedoch nicht manipuliert und sieht somit für andere Container unverändert aus. Dieses Verhalten nennt man *copy-on-write*. Dies wirkt sich sowohl auf die Performance, als auch auf den Platzverbrauch von Containern aus. Einerseits wird Kopierarbeit und damit Zeit und Rechenleistung eingespart, da alle laufenden Container die selben Basislayer verwenden. Andererseits wird Platz gespart, da nur Änderungen der Basislayer weiteren Speicherplatz benötigen.[46]

2.3.2 Control Groups

Unter einem Container versteht man ein Programm, bzw. einen Prozess oder Prozessbaum, der im Kontext eines Images läuft. Das bedeutet, dass zum Beispiel ein Webserver gestartet wird, dessen gesamte Daten sich im Image befinden. Das Programm wird als neuer Prozess am Host, sprich mit dessen Kernel gestartet. Das bedeutet wiederum, dass sich alle Container auf einem Host seinen Kernel und infolge dessen auch seine Ressourcen teilen. Um sauber abgrenzen zu können, welche Ressource zu welchem Container gehört, werden sogenannte *Namespaces* verwendet, dazu mehr unter Kapitel 2.3.3.
CGroups haben die Aufgabe Ressourcen für bestimmte Prozesse und deren Sub-Prozesse einzuschränken. So wird beispielsweise der Hauptprozess eines Containers, und damit implizit alle seine Sub-Prozesse, einer CGroup zugewiesen. Diese CGroup kann dann Ressourcen wie CPU, Arbeitsspeicher, Netzwerkbandbreite oder Input/Output Operations Per Second (IOPS) beschränken.[47] In Abbildung 2.11 wird die Beschränkung von CPU Ressourcen für die Prozesse bestimmter CGroups schematisch dargestellt. Technisch sind CGroups ein Linux Kernel-Feature und werden über Einträge in einem speziellen Filesystem verwaltet. Die Verwaltung der CGroups erfolgt heute durch sogenannte *Container Runtimes*[48], kann jedoch auch manuell oder gescriptet erfolgen.[49]
Im Appendix A.1 ist ein Beispiel für die Konfiguration von CGroups angeführt.

[46] Vgl. Miell (2019), S. 16ff.
[47] Vgl. Koutoupis (2018), Onlinequelle [06.12.2020].
[48] siehe Kapitel 2.3.5
[49] Vgl. Kane (2018), S. 156ff.

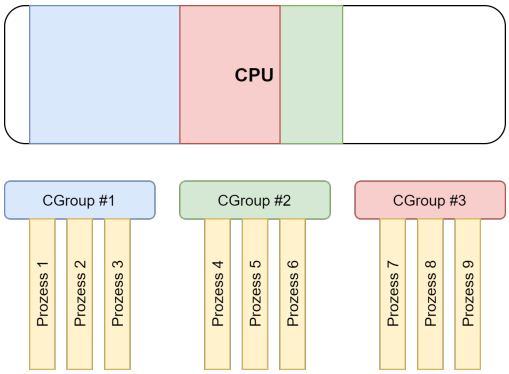


Abb. 2.11: **Schematische Darstellung der Ressourcenaufteilung mittels CGroups**
Quelle: Eigene Darstellung

2.3.3 Namespaces

Wie anfangs in Kapitel 2.3.2 erwähnt werden, um die Ressourcenabgrenzung zwischen Containern zu erreichen, sogenannte *Namespaces* verwendet. Dabei handelt es sich ebenfalls um ein Feature des Linux Kernels, welches es ermöglicht bestimmte Ressourcen, bzw. Ressourcentypen, für bestimmte Prozesse abzukapseln. Es existieren folgende Namespace-Typen[50]:

- *Mount Namespaces*: Diese stellen dem Container vermeintlich ein eigenes Filesystem zur Verfügung. Die Funktionsweise ist ähnlich dem *chroot* Befehl aus Kapitel 2.2, jedoch noch wesentlich tiefer in den Kernel integriert.
- *UTS Namespaces*: UTS steht für*UNIX Timesharing System*. Mit diesen Namespaces wird einem Container sein eigener Host- und Domainname gegeben.
- *IPC Namespaces*: IPC steht für*Inter-Process-Communication*. Diese Namespaces werden verwendet um die Kommunikation zwischen Prozessen einzuschränken. Damit soll sichergestellt werden, dass Prozesse in Containern, sprich im selben Namespace, nicht mit Prozesse außerhalb kommunizieren können.
- *PID Namespaces*: PID steht für*Process-Identifier*. Der PID ist eine eindeutige Nummer, mit der jeder Prozess im System eindeutig identifiziert und adressiert werden kann. Hiermit wird erreicht, dass Container, bzw. Prozesse eines Namespaces, auch nur Prozesse desselben Namespaces sehen. Der Startprozess eines Containers ist demzufolge der Prozess mit der PID 1 und somit der Anfang des Prozessbaums im Container. Das Hostsystem, außerhalb des Namespaces bzw. Containers, sieht alle Prozesse, auch jene in den Namespaces. Es hat für alle Prozesse im gesamten System eindeutige PIDs. Somit hat jeder Prozess, der einem Namespace zugeordnet ist mindestens zwei PIDs: die *globale* und die im Namespace *lokale* PID. Der Prozess im Namespace sieht allerdings immer nur seine *lokale* PID, und ebenso nur die *lokale* PID anderer Prozesse im Namespace.
- *Network Namespaces*: Mit *Network Namespaces* lassen sich Netzwerkgeräte explizit einzelnen Prozessen zuordnen, bzw. von anderen abschotten. Damit können beliebig erstellte *virtuelle Interfaces*, mit der Verwendung dedizierter Network Namespaces, explizit einzelnen Containern zugewiesen werden. Das Hostsystem muss sich dann jedoch mittels eigener *iptables* Konfigurationen darum kümmern, dass die Verbindungen zwischen den *virtuellen Interfaces* auch funktionieren.[51]

[50] Vgl. Kane (2018), S. 161ff.
[51] Vgl. Rosenhouse (2016), Onlinequelle [06.12.2020].

- *User Namespaces*: Mit *User Namespaces* lassen sich die User im und außerhalb des Namespaces trennen. Jedem Namespace wird ein *namespace first User Identifier (UID)* sowie ein *namespace first Group Identifier (GID)* im Hostsystem zugewiesen. Außerdem erhält ein Namespace eine bestimmte Anzahl an UIDs und GIDs. Diese *namespace first UID/GID* wird mit den UID/GIDs im Namespace verbunden. Somit hat zum Beispiel der User mit der UID 0 im Namespace, die UID 1234 am Host.
- *CGroup Namespaces:* Diese Art von Namespaces verbirgt die CGroup Zugehörigkeit des Containers vor ihm. Das bedeutet, dass der Container nicht weiß, dass er sich in einer CGroup befindet.

Namespaces im Kontext von Containern sind also ein Feature des Linux Kernels. Sie erlauben es zu definieren welche Prozesse welche Ressourcen sehen bzw. verwenden können. Somit können sie für die Isolation von Ressourcen für einzelne Container verwendet werden.

2.3.4 Volumes und Netzwerk

Die letzten beiden Bausteine, welche einem Container seinen vollen Funktionsumfang geben, sind Speicher und Netzwerk. Eine Netzwerkanbindung ist essentiell um mit der Container-Außenwelt kommunizieren zu können. Verschiedene Arten von dauerhaftem Speicher werden unter dem Begriff Volume zusammengefasst. Ein Volume könnte beispielsweise ein Speichermedium, ein geteilter Ordner des Hostsystems, oder ein Netzlaufwerk sein. Volumes sind nötig um einerseits Daten, zum Beispiel über ein gemeinsam verwendbares Filesystem, mit anderen Container austauschen zu können, und andererseits um Informationen auch über die Lebenszeit eines Containers hinaus zu speichern. Es ist ebenfalls gelebte Praxis diverse Initialdaten, Konfigurationen etc. via *Volumes* dem Container zugänglich zu machen. Sowohl die Netzwerkanbindung als auch Volumes sind grundsätzlich nur Ressourcen des Hostsystems, welche durch Namespaces abgekapselt dem Container zugewiesen werden.

Netzwerk: Vereinfacht dargestellt wird am Hostsystem für jeden Container ein Paar von virtuellen Netzwerkinterfaces angelegt. Auf der Hostseite, bzw. außerhalb des Container Network-Namespaces, wird ein Interface erstellt, welches mit einem Interface innerhalb des Container Netzwerk-Namespaces gekoppelt wird. Man kann sich dies vereinfacht wie ein Kabel mit zwei Anschlüssen vorstellen. Dieses *virtuelle Interface* stellt somit die externe Schnittstelle des Containers dar, welche er über sein internes Interface verwenden kann. Alle virtuellen Interfaces eines Hosts werden bei einer sogenannten Netzwerk-Bridge gesammelt. Diese kümmert sich um die Verbindung der Container am gleichen Host, sowie das Routing nach außerhalb des Hosts. Jeder Host besitzt zusätzlich noch ein Netzwerk-Interface, hier *Edge-Interface* genannt, welches die Verbindung zu anderen Hosts ermöglicht. Die Container können somit über ihr gekoppeltes virtuelles Interface und die Netzwerk-Bridge hostintern, und über das *Edge-Interface* mit Containern auf anderen Hosts, kommunizieren. [52]

Volumes: Ähnlich wie mit Netzwerkinterfaces verhält es sich mit Volumes. Unter Volumes können einerseits Geräte wie Festplatten, USB-Sticks und Netzwerklaufwerke, aber auch Ordner des Host-Filesystems gemeint sein. Grundsätzlich erhält jeder Container einen eigenen *mount Namespace*, in dem alle seine Volumes zugänglich gemacht werden. Der Dateipfad den eine Applikation im Container wahrnimmt ist nur ein Teil des Dateipfad am Hostsystem. Nämlich genau jener Teil, der dem *mount Namespace* zugänglich gemacht wird. Das bedeutet, dass die Umgebung die ein Container sieht in Wahrheit eine Sub-Umgebung des Hosts ist. Würde ein Volume beispielsweise unter einem anderen Pfad, welcher nicht im *mount Namespace* des Containers ist, zugänglich gemacht werden, würde der Container es nicht einmal wahrnehmen.[53]

[52] Matsiukevich (2018), Onlinequelle [06.12.2020]; Reddy (2018), Onlinequelle [06.12.2020].
[53] Vgl. Rilee (2017), Onlinequelle [06.12.2020].

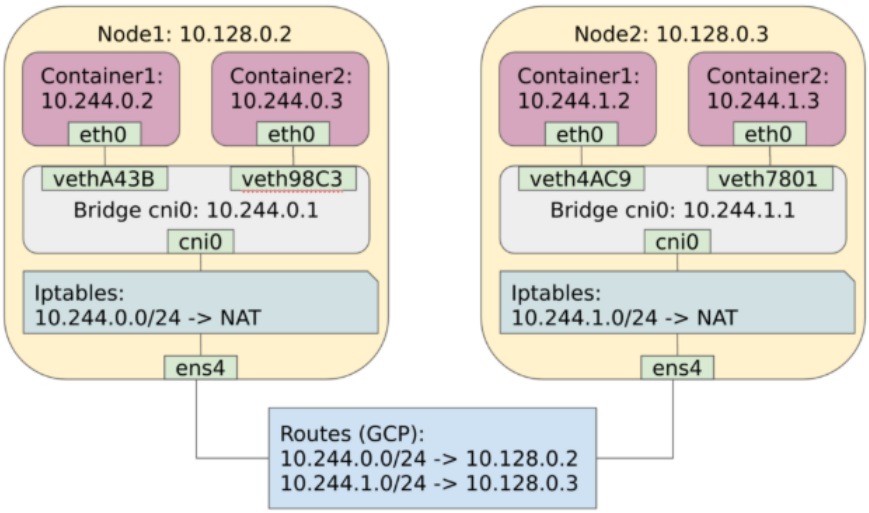


Abb. 2.12: **Container Netzwerkarchitektur**
Quelle: Matsiukevich (2018)

Mit sogenannten *bind mounts* ist es auch möglich bestimmte Ordnerstrukturen des Host-Filesystems im Container zu mounten. Somit könnten sich zum Beispiel mehrere Container ein gemeinsames lokales Verzeichnis am Host teilen.[54]

2.3.5 Container-Runtime & Container-Engine

Wie anfangs in Kapitel 2 bereits angemerkt können Container grundsätzlich auf jedem System laufen, welches eine sogenannte *Container-Runtime* hat. Voraussetzung ist lediglich, dass das Container und Host-Betriebssystem kompatibel sind. Die Container-Runtime übernimmt genau jene Tasks, welche in den vorhergehenden Kapiteln aufgezeigt wurden. Dazu gehören das Starten von Prozessen in Namespaces, die Verwaltung der dazugehörigen CGroups sowie das Verwalten von erweiterten Security Features.[55]
Da im Fachjargon zwei Begriffe oft fälschlicherweise synonym verwendet werden, wird hier auf die Unterscheidung von *Container-Runtime* und *Container-Engine* eingegangen. Die *Container-Runtime* ist das Kernstück, welches rein dafür da ist Container zu starten bzw. zu betreiben. Eine der gängigsten Container Runtimes ist zum heutigen Stand *runc*[56]. Dabei handelt es sich um eine von Docker Inc. entwickelte und der OCI zur Verfügung gestellte Open-Source Referenzimplementierung der OCI Runtime Specification [57]. Als Basiskomponente zur Ausführung von Containern wird *runc* von bekannten *Container Engines* wie *CRI-O* und *containerd* verwendet.[58]
Die *Container-Engine* ist somit eine logische Ebene über der *Container-Runtime* angesiedelt, und verwendet diese. Der Aufgabenbereich der *Container-Engine* umfasst[59]:

- das Bereitstellen eines Application Programming Interface (API) für Benutzer und Programme, beispielsweise Container-Orchestration-Tools wie Kubernetes
- das Herunterladen (pull) von Container Images von einem zentralen Speicherort, *Container Registry* genannt
- das Entpacken und Vorbereiten des Container Images für die Container-Runtime
- das Vorbereiten der Metadaten, welche der Container-Runtime beim Start übergeben werden
- das Starten bzw. Verwenden der Container-Runtime

[54] Vgl. Kuncoro (2018), Onlinequelle [06.12.2020].
[55] Vgl. McCarty (2018), Onlinequelle [06.12.2020].
[56] weitere Informationen unter: https://github.com/opencontainers/runc
[57] weitere Informationen unter: https://github.com/opencontainers/runtime-spec
[58] Vgl. Lewis (2017), Onlinequelle [06.12.2020].
[59] Vgl. Gracey (2019), Onlinequelle [06.12.2020].

Sämtliche Interaktion des Users und anderer Programme erfolgt somit über die Container-Engine. Diese wiederum bedient sich der Container-Runtime um das eigentliche Container Image als laufenden Container zu starten. In Abbildung 2.13 wird diese Architektur schematisch dargestellt.

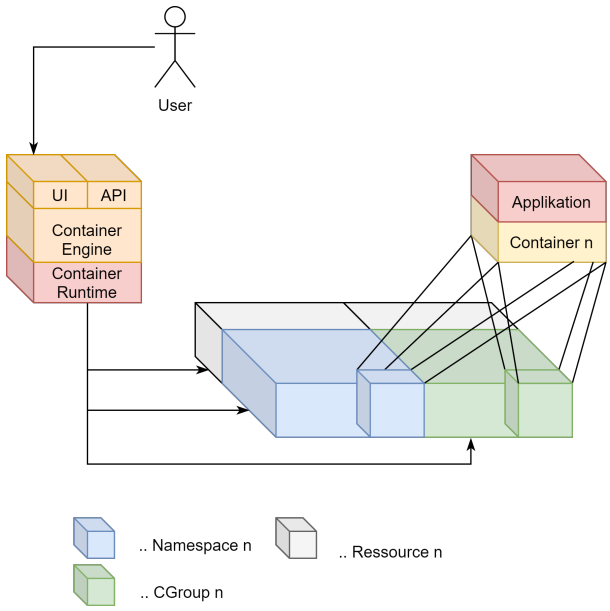


Abb. 2.13: **Schematische Darstellung von Container-Runtime und Engine**
Quelle: Eigene Darstellung

Dieses Kapitel hat sich mit der technischen Funktionsweise von Containern befasst. Das dadurch vermittelte Grundlagenwissen ist für das Innovationsmanagement interessant, um besser verstehen zu können welche Möglichkeiten Container bieten und warum sie das tun. Im folgenden Kapitel wird eine neue logische Ebene betrachtet, in der es darum geht Container nicht nur auf einem, sondern auf mehreren Rechnern zu betreiben.

2.4 Container-Orchestration mit Kubernetes

Welche grundlegenden Möglichkeiten der Technologiestack rund um Container bietet, wurde am Anfang von Kapitel 2 bereits erörtert. Dieses Kapitel widmet sich konkret dem Teil den Container-Orchestration mittels Kubernetes im Gesamtgefüge einer Containerlandschaft erfüllt. Es wird nochmal kurz darauf eingegangen welche grundlegenden Probleme Kubernetes löst. Außerdem wird aufgezeigt wie eine Kubernetes Umgebung aufgebaut ist und aus welchen grundlegenden Komponenten diese besteht. Wie in Kapitel 2.2.2 bereits angeschnitten, gibt es neben Kubernetes noch andere Produkte bzw. Projekte, die dieselben Probleme lösen. Fokus dieser Arbeit ist jedoch Kubernetes, daher werden andere Produkte wie HashiCorp Nomad, Apache Mesos (Marathon) oder Docker Swarm nicht näher behandelt.

2.4.1 Funktionsweise und Funktionalitäten von Kubernetes

Die grundlegende Funktion von Kubernetes ist es Applikationen in Containern auf beliebig vielen Rechnern zu betreiben und zu verwalten. Eine Kubernetes Umgebung, auch *Cluster* genannt, besteht aus zwei Ebenen, der *Control Plane* und der *Application Plane*. Die Rechner, fortführend *Nodes* genannt, der *Control Plane* sind das Herzstück von Kubernetes, sie übernehmen alle Kontrollaufgaben. Auf den Maschinen der *Application Plane*, im Weiteren *Worker* genannt, laufen die eigentlichen Container.

Kubernetes bringt sowohl für Operations-Teams, bzw. System Administratoren, als auch für Entwickler entscheidende Verbesserungen, welche auf den Features von Containern aufbauen. Viele Services, vor Allem im Enterprise Umfeld, müssen beispielsweise hoch verfügbar sein. Das bedeutet, dass auch wenn eine Komponente die einen Service zur Verfügung stellt ausfällt, der Service weiter bestehen bleiben muss. Dies kann zum Beispiel dadurch erreicht werden, dass mehrere Instanzen einer Applikation parallel über einen Zugangspunkt bereitgestellt werden. Man würde hierbei von einem *Load-Balancer* sprechen. Die manuelle Installation, Wartung und der Betrieb einer solchen Konstellation ist zeit- und ressourcenaufwändig. Kubernetes bietet unter anderem dieses Feature nativ an. Der parallele Betrieb von mehreren identischen Instanzen einer Applikation, beispielsweise eines Webservers, ist jedoch nicht nur ein Thema von Hochverfügbarkeit. Das Anpassen an Leistungsspitzen durch erhöhte Nachfrage eines Services, kann durch sogenanntes *Scaling* von Kubernetes übernommen werden. Entweder der Entwickler gibt selbst vor, wie breit ein Service skaliert, oder Kubernetes übernimmt die Skalierung automatisch, basierend auf gewünschten Lastmetriken.
Da Kubernetes die Worker Nodes immer überwacht, kann auch der jeweils beste Node für den nächsten zu startenden Container ausgewählt werden. Dadurch erreicht ein gesamter Cluster eine wesentlich bessere Hardware-Auslastung, als wenn die Applikationen manuell auf menschlich ausgewählten Nodes gestartet werden würden. Bei kleinen Clustern mag dies trivial erscheinen, bei großen bis riesigen Clustern können jedoch schon kleine Einsparungen gewaltige finanzielle Folgen haben.
Kubernetes bietet jedoch nicht nur Vorteile für Operations-Teams und System Administratoren, sondern auch für Entwickler und deren Management. Viele Funktionalitäten moderner verteilter Systeme, wie *Load-Balancing*, *Leader Election* oder *Service Discovery*, werden von Kubernetes übernommen und müssen somit nicht mehr von den Entwicklern implementiert werden. Dies erlaubt es wesentlich mehr Fokus auf die Entwicklung der Kernfunktionalität einer Applikation zu legen. Außerdem ist es von unschätzbarem Wert, dass das Ausrollen jeglicher Applikationen auf einem Kubernetes Cluster immer gleich funktioniert und immer wieder durchgeführt werden kann. Dies schafft Vertrauen ins System und ermöglicht automatisierte und kontinuierliche Auslieferung neuer Applikationsversionen, sogenanntes *Continuous Deployment* (CD). Durch das Wegfallen unnötiger Komplexität, welche der Kubernetes Cluster als Standard Features anbietet, wird nicht nur eine schnellere Entwicklung und Auslieferung, sondern auch eine bessere Planbarkeit erreicht.[60]

All diese Vorteile, bauen unter anderen auf folgenden Kubernetes Kernkonzepten auf: *Deklarative Konfiguration*, *Selbstheilung und Auto-Scaling* sowie *Abstraktion der Infrastruktur*.[61]
Um den Zustand eines Systems zu verändern gibt es grundsätzlich den *imperativen* und den *deklarativen* Ansatz. Beim *imperativen* Ansatz wird dem System explizit durch Befehle vorgegeben wie es sich verändern bzw. verhalten soll. Der *deklarative* Ansatz hingegen beschreibt ein Zielbild, den sogenannten *desired state*, welchen das System dann eigenständig zu erreichen versucht. Dieser Ansatz ist klar zu bevorzugen, denn er ist einerseits jederzeit reproduzierbar und andererseits technisch sehr leicht persistierbar. Das bedeutet, dass der gewünschte Zustand einer Applikation im Kubernetes Cluster sehr einfach, mittels sogenannter YAML Dateien, verschriftlicht werden kann. Diese Dateien sind leicht leserlich und können mit dem Source Code der Applikation versioniert und verwaltet werden. Man hat also den enormen Vorteil, sowohl den Code, als auch die beschriebene Umwelt bzw. Architektur der Applikation an einem Speicherort, beispielsweise in einem Source Code Repository, zu vereinen. Da Kubernetes Objekte und Container mit dem Grundsatz der *immutability* entwickelt werden, liefert die gleiche Kombination aus *desired state* und Container immer dasselbe Ergebnis. Der Begriff *immutability* besagt, dass ein Objekt, zum Beispiel ein Container, nicht mehr von außen verändert wird sobald es einmal erstellt wurde.

[60] Vgl. Luksa (2018), S. 16ff.
[61] Vgl. Burns (2019), S. 1ff.

Man würde also beispielsweise alle Systemupdates bei der Erstellung des Images installieren und das Image dann fertig ausliefern, anstatt die Updates beim Container Start zu installieren. Da beim Start andere Updates als beim Bau des Images vorliegen könnten, würde sich der Stand des Containers eventuell bei jedem Start verändern. Somit wäre keine *immutability* gegeben.[62]
Die Tatsache, dass Applikationen immer als Container gebaut und dann mittels desired state Konfiguration auf den Cluster ausgerollt werden, vereinfacht auch die Entwicklung ungemein. Da die Entwicklung nur mehr auf den Container an sich fokussiert werden muss und nicht mehr auf etwaige Infrastruktur oder Plattform, erspart sowohl Zeit als auch Ressourcen. Für den Container ist es im Endeffekt egal auf welchem Node er schlussendlich laufen wird, es müssen nur die richtigen und genügend Ressourcen vorhanden sein. Damit stellt Kubernetes nicht nur gewisse Services für den Entwickler bereit, sondern abstrahiert auch die gesamte Hardware bzw. Infrastruktur auf eine Ebene die für den Entwickler immer gleich aussieht und funktioniert.
Der *desired state* kann jedoch nicht nur besagen, dass zum Beispiel zu jedem Zeitpunkt *n* Instanzen eines Webservers laufen müssen, er kann auch einfach besagen, dass der Service verfügbar sein muss. Durch sogenanntes *self healing* und durch das bereits erwähnte *scaling*, übernimmt Kubernetes den gesamten Betrieb eines Services, inklusive etwaiger *health checks*, *Load-Balancing* und Auf- bzw. Abwärtsskalierung je nach Servicelast. Der Begriff *self healing* bedeutet in diesem Zusammenhang, dass Kubernetes erkennt wenn eine Instanz einer Applikation nicht mehr korrekt funktioniert und diese neu startet. Auch für diesen Mechanismus ist die *immutability* von Containern, bzw. Kubernetes Objekten, unabdingbar.[63]

Zusammenfassend kann also festgehalten werden, dass Kubernetes über eine sogenannte *desired state* Konfiguration mitgeteilt wird, wie die Architektur einer Applikation am Cluster aussehen soll. Dadurch, dass Container grundsätzlich *immutable* sind, wird das erzielte Ergebnis aus Container und erfolgreich erreichtem *desired state* immer gleich aussehen. Dies hat enorme Vorteile bei der Entwicklung, Fehlersuche und beim Betrieb von Anwendungen. Die *desired state* Konfiguration wird zusammen mit dem Source Code der Anwendung verwaltet, dadurch ergibt sich ein Gesamtbild aus Applikation und Infrastruktur. Des Weiteren bietet Kubernetes Entwicklern viele Features, welche bisher manuell implementiert werden mussten, was eine vertiefte Fokussierung auf die Kernthemen der Applikation erlaubt. Außerdem ist Kubernetes imstande Applikationen gemäß ihres Ressourcenverbrauchs auf unterschiedliche Nodes zu Verteilen, was eine optimale Ressourcenauslastung der Hardware garantiert.

2.4.2 Aufbau eines Kubernetes Clusters

Nachdem im vorherigen Kapitel die Funktionsweise und Funktionalitäten von Kubernetes aufgezeigt wurden, befasst sich dieses Kapitel mit dem grundlegenden Aufbau eines Kubernetes Clusters und gibt Auskunft über die Hauptbestandteile von Kubernetes. Diese Bestandteile sind in jedem Cluster gleich, was sie von den Komponenten in Kapitel 3 unterscheidet. Die Bausteine, welche in diesem Kapitel erklärt werden, bedienen sich der Möglichkeiten der Komponenten aus Kapitel 3, um die Funktionalitäten von Kubernetes zur Verfügung stellen zu können.

Der grundlegende Aufbau, bzw. die Architektur, eines Kubernetes Clusters ist immer gleich. Der Cluster besteht aus den erwähnten zwei logischen Ebenen, der *Control Plane* und der *Application Plane*, wobei jede der beiden aus ein bis *n* Nodes besteht. Bei den Nodes kann es sich jeweils um Hardware Rechner oder Virtuelle Maschinen handeln. In Abbildung 2.14 ist eine schematische Darstellung der zwei Ebenen ersichtlich. Die Control Plane stellt eine API zur Verfügung, welche wiederum Objekte definiert mit denen der desired state beschrieben wird. Welche Objekte die Kubernetes API definiert wird in Kapitel 2.4.3 näher beschrieben.

[62] Vgl. Burns (2019), S. 3f.
[63] Vgl. Luksa (2018), S. 21ff.

Vorab ist es nötig zu wissen, dass die kleinste Einheit einer Workload in Kubernetes kein Container, sonder ein sogenannter *Pod* ist. Pods bestehen aus einem bis mehreren Containern, welche zu einer Einheit zusammengefasst und immer zusammen auf einen Node platziert werden.[64]

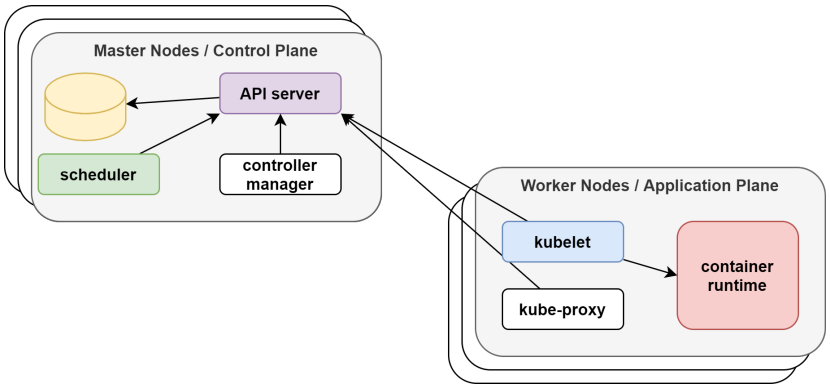


Abb. 2.14: **Kubernetes Architektur Überblick**
Quelle: angelehnt an Luksa (2018), S. 18, Fig. 1.9

Die Control Plane besteht aus folgenden Komponenten, welche sich jeweils auf den Master Nodes befinden[65]:

- *Controller-Manager*: Um den *desired state* einer Applikation im Cluster sicherstellen zu können, verwendet Kubernetes sogenannte *Controller*. Diese Controller überwachen Kubernetes Ereignisse und reagieren entsprechend auf diese. Es gibt folgende Controller, welche wiederum alle Teil der Controller-Managers sind:
  - *Node Controller*: überwacht den Status aller Nodes und reagiert sobald ein neuer Node hinzukommt oder ein Node nicht mehr reagiert.
  - *Replication Controller*: Stellt sicher, dass die richtige Anzahl der jeweiligen *Pods*, gemäß des *desired state*, läuft.
  - *Endpoint Controller*: Stellt Cluster-Service-Endpoints zur Verfügung und assoziiert diese mit den dazugehörigen Workloads.
  - *Service account und token Controller*: Initialisiert neue Kubernetes *Namespaces*[66] mit Standard Usern, sogenannten *service accounts* und dazugehörigen Zugriffsschlüsseln, *API access tokens* genannt.
- *API Server*: Der *kube-api-server* ist ein REST API Server der die Kubernetes API sowohl cluster-intern als auch extern zur Verfügung stellt.
- *Scheduler*: Der *kube-scheduler* verteilt die angeforderten Pods auf die verfügbaren Nodes.
- *etcd*: Dabei handelt es sich um eine spezielle Datenbank, einen sogenannten *key-value store*, welcher die gesamte Clusterkonfiguration speichert.

Die Application Plane besteht aus folgenden Komponenten, welche sich auf jedem Worker Node befinden[67]:

- *kubelet*: Das Kubelet übernimmt die Kommunikation zum API Server und die Verwendung der am Node befindlichen Container-Engine.
- *Container-Runtime*: Die Container-Runtime wird vom *kubelet* verwendet um Container zu starten und zu betreuen.
- *kube-proxy*: Der kube-proxy ist für das Netzwerk zwischen den Nodes zuständig. Er ermöglicht es Pods Services zur Verfügung zu stellen und bietet Load-Balancing zwischen den Pods.

[64] Vgl. The Linux Foundation (2020p), Onlinequelle [06.12.2020].
[65] Vgl. Sayfan (2019), S. 9ff.
[66] Siehe Kapitel 2.4.3
[67] Vgl. Luksa (2018), S. 18.ff.

2.4.3 Kubernetes Objekte

Dieses Kapitel erklärt wie ein Kubernetes Cluster angesprochen und verwendet wird, sowie die Grundlagen der Kubernetes API. Es wird beschrieben welche Arten von Objekten es in einem Cluster gibt, was diese unterscheidet und wie sie beschrieben werden.

Das eigentliche Installieren von Applikationen auf einen Kubernetes Cluster, im Fachjargon Deployment genannt, erfolgt durch das Definieren einer *desired state* Konfiguration. Dies kann direkt über die Programmierschnittstelle (API), oder über das Kubernetes Kommandozeilenprogramm *kubectl* mit sogenannten YAML Dateien passieren. YAML ist die Notationssprache in der die Dokumente verfasst sein müssen: "YAML is a human friendly data serialization standard for all programming languages".[68] In Anhang A.7 findet sich ein Beispiel für einen API-Aufruf sowie eine äquivalente YAML Datei. Sobald der *desired state* am Cluster konfiguriert ist versucht der Cluster diesen, mit Hilfe der verschiedenen Controller auch zu erreichen. Wie ein *desired state* aussieht wird mit Objekten die in der Kubernetes API definiert sind beschrieben. Die Kubernetes API wird vom *kube-api-server* bereitgestellt. Die grundlegenden Objekte von Kubernetes sind unter anderen *Pods*, *Services*, *Volumes* und *Namespaces*. Aufbauend auf diesen, und teilweise unter Zuhilfenahme weiterer Controller, gibt es noch diverse weitere Objekte. Eine vollständige Auflistung ist per Definition nicht möglich, da Kubernetes es ermöglicht eigene Ressourcen bzw. Objekte zu definieren, sogenannte *Custom Resource Definitions*. Gängige Objekte, die auf den vorhin genannten Objekten aufbauen, sind beispielsweise *Deployments*, *Daemon Sets*, *Stateful Sets*, *Jobs* und *Persistent Volumes*.

Grundsätzlich wird ein Objekt immer durch mindestens vier Attribute, und deren Ausprägungen, beschrieben[69]:

- *apiVersion*: die Kubernetes API Version des Objektes
- *kind*: der Objekt Typ
- *metadata*: die Objekt Metadaten wie zum Beispiel Name oder Labels
- *spec*: die eigentliche Spezifikation bzw. der *desired state*

Pods: Eines der essentiellen Kernobjekte und das Objekt welches direkt die eigentlichen Container verwendet ist ein *Pod*. Ein *Pod* besteht aus einem oder mehreren Containern, die als eine *unit of deployment* behandelt werden. Das bedeutet, dass alle Container eines Pods dieselben LinuxNamespaces haben, untereinander kommunizieren können als wären sie am selben Host (*localhost*) und auch dem gleichen Lifecycle unterworfen sind. Ein Pod wird also immer als Gesamtes gestartet und gestoppt.

Labels: Um einem Kubernetes Objekt, wie zum Beispiel einem Pod, mehrere Identifikationsmerkmale bzw. Kennzeichnungen zu geben, werden sogenannte *labels* in der *metadata* Sektion des Objektes verwendet. Dabei handelt es sich um einfache key-value Paare, welche beispielsweise Auskunft über die Version, den Namen, den Business-Service etc. geben können. Diese *labels* werden auch technisch verwendet, und stellen somit nicht nur eine bloße Beschreibung dar. In Anhang A.2 findet sich ein Beispiel eines Pods mit Labels in YAML Notation.

configMap/Secret: Um einem Pod, bzw. dessen Containern, Konfigurationen von außen übergeben zu können, haben sich zwei Ansätze bewährt. Entweder die Werte werden als Umgebungsvariablen im Container gesetzt, oder es wird ein Volume mit den Konfigurationen als Konfigurationsdateien zur Verfügung gestellt. Beide Ansätze lassen sich mit sogenannten *configMaps*, einem weiteren Kubernetes nativem Objekttyp, bewerkstelligen. Eine *configMap* kann einzelne key-value Paare, oder auch ganze Konfigurationsdateien enthalten.

[68] weitere Informationen unter: https://yaml.org/
[69] Vgl. The Linux Foundation (2020t), Onlinequelle [06.12.2020].

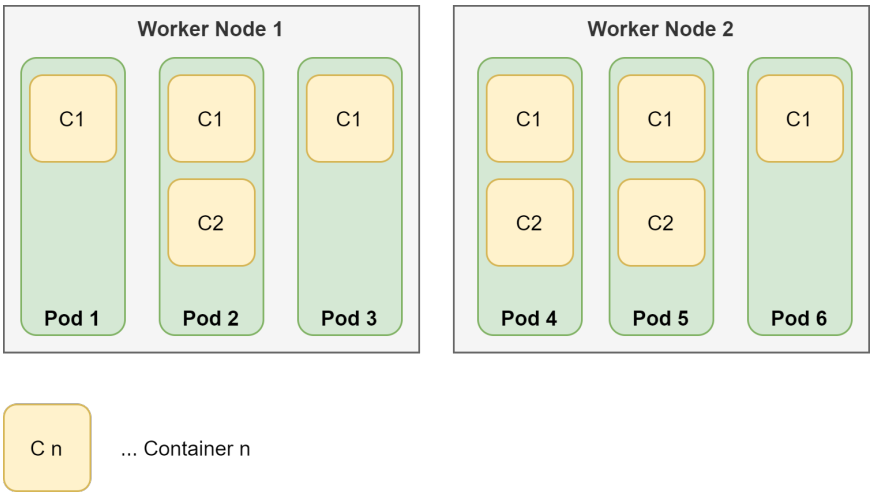


Abb. 2.15: **Schematische Darstellung von Pods**
Quelle: In Anlehnung an Luksa (2018), S. 43, Fig. 2.5

Das *configMap* Objekt, oder einzelne key-value Paare, können einem Container als Umgebungsvariablen oder als Volume zur Verfügung gestellt werden.[70] Ein Beispiel für die Verwendung von *configMaps* wären unterschiedliche Konfigurationen eines Webservers für verschiedene Anwendungsfälle.[71] Um neben normalen Konfigurationen auch sensible Daten wie Passwörter, Zertifikate oder Keys speichern und Containern zur Verfügung stellen zu können, gibt es in Kubernetes das *Secret* Objekt. Secrets sind, was die Handhabung betrifft, gleich zu verwenden wie *configMaps*, allerdings werden die Inhalte von *Secrets* am Node immer im Memory gehalten und in *etcd* verschlüsselt aufbewahrt.[72] In Anhang A.3 findet sich ein Beispiel einer ConfigMap in YAML Notation.

Namespaces: Nachdem *Pods* nun mit *labels* unterschieden und durch *configMaps* und *Secrets* konfiguriert werden können, ist es sinnvoll Pods voneinander zu trennen. Eine logische Gruppierung von Kubernetes Objekten erfolgt mit sogenannten *Namespaces*. Diese kann man sich wie Ordner eines Dateisystems vorstellen. Ressourcen in unterschiedlichen Namespaces können dieselben Namen haben, was es zum Beispiel ermöglicht identische Umgebungen parallel zu betreiben, wie es etwa bei verschiedenen Stages Sinn macht. Des weiteren können auf *Namespaces* Berechtigungen vergeben werden, was etwa ein Konzept von einem Namespace pro Entwicklerteam ermöglichen könnte. Es ist wichtig an dieser Stelle hervorzuheben, dass Kubernetes Namespaces nichts mit LinuxNamespaces, siehe Kapitel 2.3.3, zu tun haben. Welchem Namespace ein Kubernetes Objekt zugeordnet ist, ist dem *namespace* Attribut in der *metadata* Sektion zu entnehmen. Pods eines Namespaces können standardmäßig Verbindungen mit Pods anderer Namespaces herstellen. Um dies zu unterbinden benötigt man erweiterte Funktionalitäten.[73]
In Abbildung 2.16 ist eine schematische Darstellung von Namespaces angeführt.

Multi-Pod: Um mehrere identische *Pods* unter demselben Namen zu betreiben, beispielsweise um die Last auf einzelne Pods zu verteilen, gibt es drei gängige Möglichkeiten. Diese Möglichkeiten werden wiederum durch Kubernetes API Objekte beschrieben, welche jeweils auf das Pod Objekt aufbauen. Mit einem *Deployment* können beliebig viele *Pods* parallel gestartet werden. Diese werden auf zufälligen Nodes und mit einer zufällig generierten Zeichenfolge als Teil des Namens gestartet.[74] Sollte es nötig sein, dass die *Pods* immer einen gleichbleibenden, nummeriert aufsteigenden, Namen haben, so muss ein *Stateful Set* verwendet werden.

[70] Vgl. The Linux Foundation (2020j), Onlinequelle [06.12.2020].
[71] Vgl. Luksa (2018), S. 198ff.
[72] Vgl. The Linux Foundation (2020r), Onlinequelle [06.12.2020].
[73] Vgl. The Linux Foundation (2020o), Onlinequelle [06.12.2020].
[74] Vgl. Luksa (2018), S. 250ff.

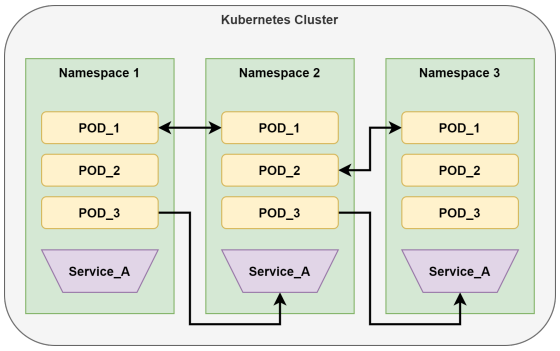


Abb. 2.16: **Schematische Darstellung von Namespaces**
Quelle: Eigene Darstellung

Außerdem assoziiert ein *Stateful Set* Ressourcen wie Volumes und IP Adressen fix mit dem Pod und stellt dem Pod dieselben Ressourcen zur Verfügung wenn er neu gestartet wird. Aus diesem Grund können mit einem *Stateful Set* Pods betrieben werden, die ihren Status über ihren Lebenszyklus hinaus behalten müssen.[75] Die dritte Option um parallel mehrere identische Pods zu betreiben sind *Daemon Sets*. Dieser Objekttyp startet einen Pod auf jedem Node, was sich ähnlich wie es ein LinuxDaemon verhält.[76] Die schematische Darstellung in Abbildung 2.17 soll die Unterschiede zwischen den beschriebenen Typen darstellen. In Anhang A.5 findet sich ein Beispiel eines Deployments in YAML Notation.

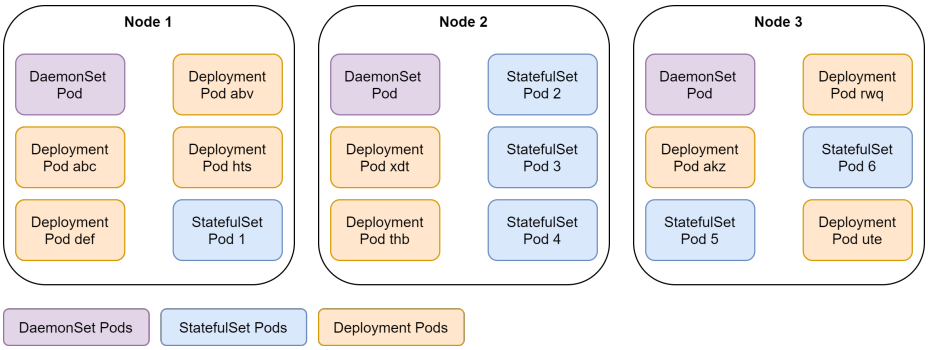


Abb. 2.17: **Schematische Darstellung von Deployments/StatefulSets/DaemonSets**
Quelle: Eigene Darstellung

Service: Um Funktionen die ein oder mehrere Pods zur Verfügung stellen auch anderen Pods sauber zugänglich zu machen, wird das *Service* Objekt verwendet. Grundlegend erstellt ein Service eine sogenannte *ClusterIP*, hinter der sich bestimmte Pods befinden. Welche Pods hinter dem Service stehen wird mit einem sogenannten *Selector* im Service Objekt festgelegt. An dieser Stelle könnten zum Beispiel die Labels von Pods als Selektionsmerkmal herangezogen werden.[77] In Anhang A.4 findet sich ein Beispiel eines Service in YAML Notation.

Ingress/LoadBalancer: Es ist auch möglich Services außerhalb des Clusters nutzbar zu machen. Hierfür gibt es zwei besondere Service Typen: *Ingress* und *LoadBalancer*. Ein *Ingress Service* ist im Prinzip eine Reverse Proxy Regel. Jeder Cluster hat im Normalfall einen Ingresscontroller, der eine externe IP Adresse verwendet und von außerhalb auf dieser erreichbar ist. An diesem Ingresscontroller können nun Regeln, die *Ingress Services*, definiert werden, wie Anfragen von außen nach innen geleitet werden sollen.

[75] Vgl. Luksa (2018), S. 280ff.
[76] Vgl. Luksa (2018), S. 108ff.
[77] Vgl. The Linux Foundation (2020s), Onlinequelle [06.12.2020].

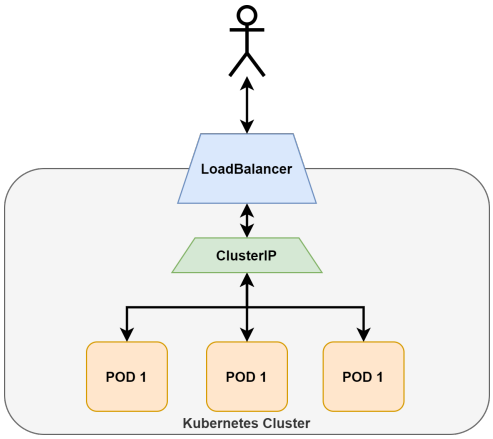


Abb. 2.18: **Schematische Darstellung eines LoadBalancer Services**
Quelle: Eigene Darstellung

Damit können zum Beispiel Anfragen an einen bestimmten Hostnamen auf ein internes Service geroutet werden. Ein *LoadBalancer Service* ermöglicht es einen internen Service direkt über eine externe IP Adresse freizugeben. In den Abbildungen 2.19 und 2.18 ist schematisch die Funktionsweise von LoadBalancer Service und Ingress aufgezeigt.

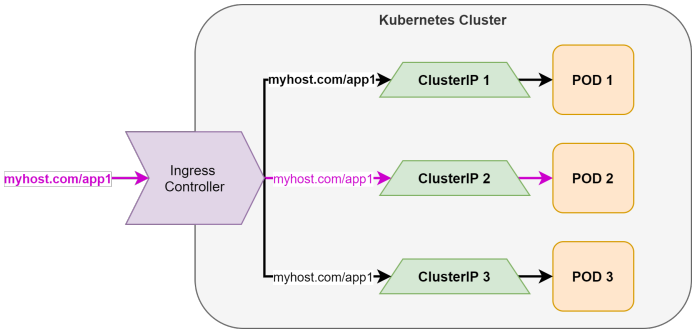


Abb. 2.19: **Schematische Darstellung einer Kubernetes Ingress Regel**
Quelle: Eigene Darstellung

Volumes: Um den Status von Pods zu sichern oder Daten auszutauschen benötigt man eine Möglichkeit Daten zu speichern. Hierfür werden wie schon bei Containern alleine sogenannte *Volumes* herangezogen. Im Kubernetes Umfeld gibt es drei Arten wie einem Container ein Volume zugänglich gemacht werden kann.

- Volumes können direkt über den jeweiligen Typ im *.spec.volumes* Attribut definiert und im Container eingebunden werden.
- Sogenannte Persistent Volume (PV) Objekte können vom Administrator als Ressource im Cluster angelegt werden. Diese PVs können dann mit dem Persistent Volume Claim (PVC) Typ anhand bestimmter Attribute wie Größe, Geschwindigkeit oder Zugriffs Modus ausgewählt werden. Der PVC wird wiederum im jeweiligen Container entsprechend eingebunden.
- Es ist auch möglich dem Cluster sogenannte *Storage Classes* zur Verfügung zu stellen. Damit kann der Cluster automatisch die passenden PV zu den angeforderten PVC erstellen und dem Container zur Verfügung stellen.

Es gibt noch viele andere Objekte in der Kubernetes API, welche den Dokumentationen der verschiedenen Versionen der API zu entnehmen sind.[78] Da diese Objekte für das unmittelbare Verständnis von Kubernetes jedoch nicht nötig sind, und den Umfang dieser Arbeit nur unnötig strapazieren würden, wird an dieser Stelle nicht weiter darauf eingegangen. Abschließend sei also festgehalten, dass der *desired state* im Kubernetes Cluster über Objekte der Kubernetes API definiert wird. Eines der Kernelemente ist der Pod, welcher von anderen Objekten verwendet wird und seinerseits andere Objekte für bestimmte Anwendungsfälle heranzieht. Alles was den Zustand eines Kubernetes Clusters beschreibt, ist mit Kubernetes Objekten definiert. Dies umfasst die Pods und deren Anzahl bzw. Verteilung, angelegte User, verfügbare Konfigurationen und Speichermöglichkeiten, verwendbare Verbindungen zur Außenwelt des Clusters und vieles mehr.

2.4.4 Möglichkeiten und Anwendungsgebiete von Kubernetes

Die vorhergehenden Kapitel haben aufgezeigt welche Funktionalitäten Kubernetes zur Verfügung stellt, sowie erklärt wie Kubernetes grundlegend funktioniert und aufgebaut ist. Es wurde beschrieben wie ein gewünschter Zielzustand am Cluster definiert wird und auf welche Art der Beschreibung dieser *desired state* aufbaut. Speziell im Innovationsmanagement ist es wichtig grundlegend zu wissen wie Technologien funktionieren, um abschätzen zu können, ob und warum man auf eine Technologie setzen sollte. In den vorhergehenden Kapiteln wurde verstärkt technisch erklärt wie die Technologien im Kubernetes Umfeld funktionieren. Dies soll dem Innovationsmanager dabei helfen verstehen zu können, ob Kubernetes für die Umsetzung einer konkreten Innovation sinnvoll ist. Dieses Kapitel verschafft einen Überblick über die Vorteile und Möglichkeiten von Kubernetes und zählt einige Anwendungsbeispiele aus der Praxis auf, in der Kubernetes schon erfolgreich verwendet wird. Es soll dem Innovationsmanager auf betrieblicher Seite aufzeigen, welche Vorteile die Verwendung von Kubernetes bringt.

Der erste enorme Vorteil von Kubernetes ist, dass dynamisch auf die aktuelle Systemauslastung reagiert werden kann. Dies bedeutet, dass beispielsweise bei Online-Shops die verfügbaren Ressourcen am Black Friday automatisch erweitert werden können. Geht die Anzahl der Einkäufe, sprich die Systemlast, wieder zurück, so werden auch die verwendeten Ressourcen wieder verringert.
Die Automatisierung von Ressourcen spielt auch im Bereich Speicher eine enorme Rolle. Mit Kubernetes muss ein Entwickler nicht mehr mühsam manuell Speicher anfordern, welcher individuell von einem Administrator bereitgestellt werden muss. Es genügt mittels *desired state* zu definieren wie der benötigte Speicher aussehen und wie er verwendet werden soll. Die Bereitstellung und Einbindung des Speichers an der richtige Stelle kann vollautomatisch von Kubernetes durchgeführt werden. Natürlich spricht auch nichts gegen die klassische manuelle Durchführung. Durch die Beschreibung des Sollzustandes am Cluster, im Gegensatz zur klassischen Abfolge von einzelnen Befehlen, entsteht implizit ein selbstheilendes System. Stürzt beispielsweise eine Applikation am Cluster ab, so nimmt Kubernetes dies wahr und startet die Applikation automatisch neu. Dies erspart manuelle Arbeit und stelle eine höhere Verfügbarkeit sicher. Viele der Funktionalitäten die der Cluster selbst verwendet stehen auch Applikationen im Cluster zur Verfügung. Ein Beispiel hierzu ist die Auswahl der primären oder führenden Instanz bei redundanten Applikationen. Technisch spricht man hier von Leader Election, einer Anforderung die ohne Kubernetes von der Applikation selbst durchgeführt werden müsste. Im Cluster selbst wird Leader Election unter anderem bei einigen Komponenten der Control Plane, sprich der Master Nodes, verwendet. Neben dem automatischen Skalieren von Ressourcen bietet Kubernetes auch einige Möglichkeiten um die Last am System gleichmäßig auf die verfügbaren Ressourcen zu verteilen. Aufbauend auf klassischer Lastverteilung ist es auch möglich komplexere Szenarien, zum Beispiel bei der Einführung einer neuen Applikationsversion, umzusetzen. Es ist beispielsweise möglich nur einen Teil der eingehenden Anfragen an eine neue Version der Applikation weiterzuleiten. Dies wird häufig bei großen Onlinesystemen angewendet, um das Ausmaß möglicher Fehler neuer Versionen anfangs gering zu halten.

[78] weitere Informationen unter: https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.18/

Auch komplexere Verteilung von Anfragen, basierend auf diversen Kriterien, ist mittels Kubernetes möglich. Abgesehen von den vielen Funktionen die Kubernetes zur Verfügung stellt, ist einer der größten technischen Vorteile von Kubernetes dessen Abstraktion der Infrastruktur. Aus Sicht der Applikation, bzw. des Entwicklers, ist jeder Kubernetes Cluster gleich zu verwenden. Das bedeutet, dass dieselbe Applikation auf jedem beliebigen Cluster lauffähig ist. Dabei ist es egal wo sich dieser Cluster befindet oder wie die Infrastruktur darunter aussieht. Da jede Applikation in einem eigenen Container läuft ist es auch egal welche Technologie die anderen Applikationen am Cluster verwenden. Aus Sicht des Entwicklers ist das Umfeld für seine Applikation immer gleich. Des Weiteren ermöglicht es die Kombination von Kubernetes und Containern die Ressourcenausnutzung des Clusters so optimal wie möglich zu gestalten. Der Cluster entscheidet selbst, wie die Applikationen, sprich Pods, am Besten auf die Nodes verteilt sind und kann diese Verteilung auch jederzeit anpassen. Wenn vom Cluster eine automatische Skalierung von Nodes unterstützt wird, kann sogar die Größe des Clusters selbst, basierend auf den aktuellen Ressourcenauslastungen, optimiert werden. Dies kann zu signifikanten finanziellen Einsparungen führen.

Die bisher genannten Vorteile beschreiben allesamt technische Aspekte, jedoch gibt es auch gute wirtschaftliche und betriebliche Argumente für Kubernetes. Neben Kubernetes selbst gibt es viele weitere Projekte und Produkte welche mit Kubernetes kompatibel sind, oder darauf aufbauen. Viele dieser Projekte sind Open-Source und frei verwendbar, einige davon werden ebenfalls von der CNCF verwaltet. Dies hat den enormen Vorteil, dass für viele klassische Problemstellungen schon fertige Lösungen vorhanden sind, welche oft sogar kostenlos sind. Die verstärkte und professionelle Open-Source Bewegung rund um Cloud Computing und Kubernetes stellt somit schon viele ausgereifte Lösungen zur Verfügung. Dies ermöglicht es Unternehmen schnell und professionell mit der Entwicklung neuer Applikationen starten zu können. Auch langfristig kann der Fokus besser auf die Entwicklung der Kernapplikation behalten werden. Sehr viele der Infrastrukturprobleme, welche im Lebenszyklus einer erfolgreichen Applikation auftreten, können schon von Anfang an mit Kubernetes und dessen Begleitprojekten gelöst werden.
Kubernetes ist nicht nur ein sehr erfolgreiches, sondern auch interessantes Projekt. Dies ist beispielsweise im Recruitingprozess ein gutes Argument um das Interesse potenzieller Bewerber zu erregen. In der StackOverflow Entwickler Studie von 2019 ist klar ersichtlich, dass Programmierer und IT Personal immer häufiger anhand des Technologiestacks ihren zukünftigen Arbeitgeber auswählen. Kubernetes ist, was die Attraktivität für Bewerber betrifft, im Spitzenfeld der beliebten Technologien.[79]
Viele Instrumente und Methoden des Innovationsmanagements, sowie der gesamte Bereich des Technologiemanagements, zielen darauf ab zukunftssichere Technologien zu finden und zu etablieren. Im IT Bereich zählt Kubernetes aktuell zu den zukunftssichersten Technologien was sich durch mehrere Aspekte begründen lässt. Zum Ersten stellt Kubernetes das Unternehmen vor keinen Vendor Lock-In. Das bedeutet, dass ein Wechsel jeglicher Komponente sowie des Providers oder der Distribution im Grunde kein Problem darstellt. Es kann natürlich zu kleineren Hürden kommen, jedoch sind diese wesentlich kleiner wie bisherigen proprietären Systemen. Des Weiteren ist Kubernetes bereits durch die enorme Verbreitung als de facto Standard etabliert. Alleine die Mitgliederliste der CNCF[80] lässt nahezu kein namhaftes Unternehmen vermissen. Diese Unternehmen tragen jedoch aktiv zu Kubernetes und anderen CNCF Projekten bei und nutzen diese nicht bloß. Die Dunkelziffer der Unternehmen welche Kubernetes nur nutzt, ohne aktiv an der Entwicklung mitzuhelfen ist noch erheblich größer. Die Verbreitung von Kubernetes lässt sich auch relativ einfach an der Anzahl der Beitragenden (Contributor) auf GitHub aufzeigen. Das GitHub Repository ist der Mittelpunkt der Entwicklung von Kubernetes, dort wird der Quellcode verwaltet. Laut GitHub[81] liegt Kubernetes mit ca. 6800 Contributern auf Platz 7 der Projekte mit den meisten Beitragenden.

[79] Vgl. Stack Exchange Inc. (2019), Onlinequelle [06.12.2020].
[80] weitere Informationen unter: https://www.cncf.io/about/members/
[81] Vgl. GitHub Inc. (2019), Onlinequelle [06.12.2020].

Weitere enorme Vorteile von Kubernetes, welche sich für das Unternehmen in Zeit- und Geldersparnis niederschlagen sind Stabilität und Entwicklungsgeschwindigkeit. Durch die Selbstheilungsfähigkeit von Kubernetes, kombiniert mit der Möglichkeit Applikationsversionen ohne Serviceunterbrechung zu tauschen, wird ein optimaler Betrieb sichergestellt. Außerdem ist das System ideal für die nahtlose Automatisierung der Auslieferung von Applikationen auf den Cluster geeignet. All dies ermöglicht es wesentlich kürzere Entwicklungszyklen zu realisieren. Dadurch können mehrere Releases, höhere Kundenzufriedenheit und im Endeffekt bessere Produkte erzielt werden. Die weitgehende Automatisierung, auch der umliegenden Infrastruktur, beseitigt außerdem menschliche Fehler und verbessert zusätzlich Stabilität und Gesamtperformance des Systems.
Neben den impliziten finanziellen Einsparungen durch Effizienzsteigerung, Fehlervermeidung und verbesserte Fokussierung auf die eigentliche Entwicklung, spielt natürlich die explizite finanzielle Einsparung auch eine große Rolle. Kubernetes an sich, zumindest Upstream Kubernetes, sowie die meisten anderen Projekte und Produkte im Kubernetes Umfeld sind kostenfrei verwendbar. Dadurch können hohe Lizenzkosten, welche für proprietäre Alternativprodukte anfallen würden gespart oder anderweitig verwendet werden.[82]

Die Popularität von Kubernetes und dem gesamten Container Technologiebereich zeigt sich auch in den Wachstumszahlen des Marktes. Im Jahr 2016 wurde der Container Markt noch mit einem Volumen von 700-760$ Mio. beziffert. Das Wachstum wurde bis 2020 mit 2,7$ Mrd. und bis 2025 mit 8,2$ Mrd. prognostiziert, was einer Jährlichen Wachstumsrate von 30-40% entspricht. Dies deutet nicht nur darauf hin, dass sich Container und Kubernetes als Technologien etabliert haben. Es zeigt auch neue Möglichkeiten auf, als Hersteller oder Provider einen enorm stark wachsenden Markt zu nutzen.[83]

Einer der Hauptaspekte dafür, dass sich der Container Markt so stark entwickelt, ist unter anderem das breite Anwendungsgebiet. Die folgenden Beispiele sollen aufzeigen, dass Kubernetes in diversen Branchen und Anwendungen einsetzbar ist.

- **Webservices:** In der heutigen Zeit sind Online Services einer der häufigsten Anwendungsfälle für Kubernetes. Repräsentativ dafür soll der Verbreitungsgrad bei Online-Shops und Marktplätzen aufzeigen, wie populär Kubernetes mittlerweile geworden ist. Der eCommerce Gigant JD.com, eines der größten Unternehmen Chinas, begann 2017 damit die bestehende Infrastruktur auf Kubernetes zu migrieren. Mit dieser Umstellung konnten 20-30% an IT Kosten gespart, sowie die Auslieferungszeit von Applikationen von Stunden auf Sekunden reduziert werden.[84]
  Der Online-Shop-Provider Shopify erkannte mit steigender Kundenzahl und Komplexität der Systeme ebenfalls den Bedarf für ein solides Orchestrierungstool. Zwischen 2011 und 2017 stieg die Anzahl der produktiven Services von Shopify um 2000%, die meisten davon basierten bereits auf der Container Technologie. Aus diesem Grund entschied sich Shopify dafür ihre gesamte Infrastruktur in die Google Cloud zu geben und Kubernetes als Herzstück der Plattform zu etablieren. Die Shopify Plattform betreut über 600.000 Händler und verarbeitet zu Spitzenzeiten mehr als 80.000 Anfragen pro Sekunde.[85]
  Analog zu Shopify migrierte auch Etsy, einer der größten Onlinehändler Amerikas, die gesamte Infrastruktur in die Google Cloud. Die Migration dauerte fast zwei Jahre, wobei ca. 5,5 Petabytes an Daten, welche über 65 Mio. Artikel repräsentieren, verschoben wurden. Durch die Automatisierungsmöglichkeiten, welche Cloud und Kubernetes bieten, konnte Etsy 15% der Entwicklerressourcen von Basisbetrieb hin zu kundenorientierter Entwicklung verschieben.[86]

[82] Thiry (2019), Onlinequelle [06.12.2020]; Ramanathan (2019), Onlinequelle [06.12.2020].
[83] Casey (2017), Onlinequelle [06.12.2020]; Correa (2020), Onlinequelle [06.12.2020].
[84] The Linux Foundation (2017); The Linux Foundation (2020e)
[85] Bryant (2018), Onlinequelle [06.12.2020]; Neufeld (2018), Onlinequelle [06.12.2020].
[86] Gibbs (2018); Carey (2020)

Bei eBay wird, aus Gründen der Performance, Kubernetes On-Premises auf Hardware Nodes verwendet. Damit schafft es eBay im Durchschnitt 300.000 Anfragen pro Sekunde, pro Datacenter, zu verarbeiten. Das gesamte System betreut mehr als 175 Mio. Benutzer und über 1.1 Mrd. Artikel. Dies äußert sich in ca. 300 Mrd. Anfragen pro Tag und einer Datenmenge von über 500 Petabytes. Die Performance um diese Last zu bewältigen wird durch die optimale Ressourcenauslastung von Kubernetes erzielt. Zusätzlich dazu ist eine Serviceverfügbarkeit von 99.999% gegeben.[87]

- **Machine Learning/Künstliche Intelligenz:** Neben klassischen Webapplikationen bietet sich Kubernetes auch für Machine Learning (ML) und Artificial Intelligence (AI) Anwendungen an. Aufgrund von weitreichendem Hardware Support, automatischer Skalierung und vollautomatischem Datenmanagement, erfüllt Kubernetes alle Anforderungen einer ML Basisplattform.[88] Außerdem ermöglichen es Tools wie KubeFlow[89], viele gängige ML Frameworks einfach auf Kubernetes verwenden zu können. Dies erleichtert sowohl die Erstellung neuer Anwendungen, als auch die Migration bestehender Systeme. Die Anwendungsgebiete von ML und AI sind wiederum sehr breit gefächert, die folgenden Beispiele sollen dies kurz verdeutlichen.
  Capital One, eine der größten Banken Amerikas, nützt Kubernetes für Betrugsentdeckung und die automatische Abwicklung von Kreditentscheidungen.[90] Babylon, ein amerikanisches Unternehmen welches digitale Gesundheitsservices anbietet, hat seine gesamte Infrastruktur auf Kubernetes aufgebaut. Die Migration der Services in die Cloud erhöhte einerseits die Entwicklungsgeschwindigkeit um ein vielfaches und löste auf der anderen Seite das aufkommende Problem mangelnder Hardware Ressourcen in den eigenen Datacentern. Zusätzlich konnte Babylon durch die steigende Akzeptanz der Cloud leichter in andere Länder expandieren.[91]
- **Streaming:** Spätestens seit den Erfolgszügen von Unternehmen wie Spotify oder Netflix sind Streamingdienste für Video und Audio vom Markt nicht mehr weg zu denken. Schon 2014 setzte Spotify stark auf die Verwendung von Containern, bis 2017 noch verwaltet durch ein eigenes Orchestrations Tool namens Helios. Seit der Migration von Helios auf Kubernetes konnte Spotify die Ausnutzung von Prozessorressourcen mindestens verdoppeln, sowie gleichzeitig die Erstellungszeit neuer Services von Stunden auf Sekunden reduzieren. Die Hauptgründe für den Technologiewechsel bei Spotify waren die viel größere Entwicklercommunity, sowie die Performance- und Skalierungsvorteile von Kubernetes. [92] Auch HBO erfuhr mit steigender Popularität der Serie *Game of Thrones* dieselben Probleme wie Spotify. Nämlich steigenden Ressourcenbedarf bei unzureichender Ausnützung, enorme Mengen an parallel konsumierenden Kunden und den Bedarf an eine schnell skalierende Plattform. Durch die Verwendung von Kubernetes, in Kombination mit der Amazon Cloud AWS, konnte die siebte Staffel von *Game of Thrones*, wie geplant und ohne größere Probleme, dem Publikum präsentiert werden. HBO erwähnt ebenfalls die Community rund um Kubernetes als eines der Hauptargumente für die Verwendung der Technologie.[93]
- **Gaming**: Das letzte Beispiel, welches die breite Verwendbarkeit von Kubernetes unterstreichen soll, kommt von Google und Ubisoft. Die beiden Softwareunternehmen veröffentlichten 2018 gemeinsam das *Agones* Projekt[94]. Dabei handelt es sich um eine Erweiterung von Kubernetes, womit Gameserver optimiert auf Kubernetes betrieben werden können. Ubisoft, sowie andere Gamingstudios, sind damit in der Lage Gameserver vollautomatisch, optimiert und zugeschnitten auf die jeweilige Last zu betreiben. Spieleserver haben, im Vergleich zu anderen Applikationen, andere Eigenschaften und Anforderungen.

[87] Jackson (2020), Onlinequelle [06.12.2020]; Kidd (2019), Onlinequelle [06.12.2020].
[88] Vgl. Platform9 Systems Inc. (2019), Onlinequelle [06.12.2020].
[89] weitere Informationen unter: https://www.kubeflow.org/
[90] Vgl. The Linux Foundation (2020c), Onlinequelle [06.12.2020].
[91] Vgl. The Linux Foundation (2020b), Onlinequelle [06.12.2020].
[92] The Linux Foundation (2020f); Hickey (2018)Williams (2020a)
[93] Vgl. Kerner (2017), Onlinequelle [06.12.2020].
[94] weitere Informationen unter: https://github.com/googleforgames/agones

> Zwei davon sind beispielsweise die verhältnismäßig kurze Lebenszeit der Gameserver, sowie die Verringerung der Latenzzeiten auf ein absolutes Minimum. Neben den Gameservern setzt Ubisoft auch für interne Services stark auf Kubernetes und bietet für die eigenen Entwickler sogar ein Ubisoft Kubernetes Service (UKS) an. [95]

Dieses Kapitel hat aufgezeigt was Kubernetes ist und tut, wie es funktioniert und wie ein Cluster grundlegend aufgebaut ist. Aus Sicht des Innovationsmanagements ist nun klar wie Kubernetes grundlegend funktioniert und wofür die Plattform verwendet werden kann. Es wurde außerdem erklärt welche Vorteile die Verwendung von Kubernetes mit sich bringt und in welchen Szenarien sich das System schon bewährt hat. Somit sollte ein Grundgefühl vorhanden sein, um einschätzen zu können für welche Innovationen Kubernetes als Technologie geeignet ist. Konkreter wird auf dieses Thema noch in Kapitel 4.2 eingegangen.

[95] Mandel (2018), Onlinequelle [06.12.2020]; Kvitka (2020), Onlinequelle [06.12.2020].

3 KUBERNETES KOMPONENTEN

Die vorhergehenden Kapitel haben das Thema der Container Technologie sowie die grundlegende Funktionsweise, Aufbau und Vorteile, sowie Anwendungsgebiete von Kubernetes behandelt. In Kapitel 2.4.2 wurden die Hauptbestandteile eines Kubernetes Clusters aufgezeigt. Jene Teile welche die Kernfunktionalität von Kubernetes realisieren und in jedem Cluster dieselben sind. Das folgende Kapitel widmet sich den konkreten Komponenten eines Kubernetes Clusters, sowie deren Eigenschaften und Unterscheidungsmerkmalen. Der Cluster bedient sich dieser Komponenten, um die Funktionalitäten mit Hilfe der Bestandteile aus 2.4.2 realisieren zu können.

Der erste Abschnitt befasst sich mit der Kubernetes Distribution, welche streng genommen nicht Teil des Clusters ist, sondern das gewählte Kubernetes Produkt selber. Jede Distribution hat in ihrem Kern das Open-Source Upstream Kubernetes Projekt, jedoch ergänzt um Zusatzkomponenten, Funktionen und Tools.

3.1 Kubernetes Distribution

Unter der Kubernetes Distribution, versteht man eine fertig gebaute, paketierte Software Plattform, welche auf Kubernetes basiert. Diese Distributionen kümmern sich einerseits darum die Komplexität für Installation und Wartung zu reduzieren, andererseits stellen sie teilweise weitere Features zur Verfügung. Man kann Kubernetes Distributionen beispielhaft mit Linux Distributionen vergleichen. Jedes Linux Betriebsystem, also jede Distribution, verwendet denselben Linux Kernel, baut jedoch unterschiedliche Komponenten und Funktionalitäten darauf auf. Die Entscheidung darüber welche Distribution gewählt wird, ist die erste und wichtigste im Designprozess eines Clusters. Welche Implementierung für jede weitere Komponente in Frage kommt, hängt essentiell von der gewählten Engine ab. Außerdem erübrigt sich die Entscheidung für gewisse Komponenten, da manche Kubernetes Engines gewisse Komponenten selbst mitliefern.

Grob lassen sich Kubernetes Distributionen in vier Gruppen einteilen:

- "pure" Distributionen: Diese stellen lediglich eine reine Upstream Kubernetes Umgebung zur Verfügung. Meist wird bei diesen Distributionen auf die Erleichterung der Installation Fokus gelegt. Die weitere Wahl der Implementierungen für die einzelnen Komponenten wird größtenteils dem Nutzer überlassen.
- "plus" Distributionen: Diese Distributionen stellen weitere Teile des Stacks, neben dem reinen Upstream Kubernetes zur Verfügung. Beispiele hierfür wären das Node Betriebssystem, die Container-Runtime oder Control Plane Erweiterungen.
- Kubernetes-as-a-Service (KaaS): Darunter fallen alle Optionen die dem Nutzer nur mehr die Kubernetes Schnittstelle zur Verfügung stellen. Dazu zählen Public-Cloud-Services wie AKS, EKS und GKE, sowie Kubernetes Managed-Service Angebote von anderen Unternehmen.
- Limited-Purpose Distributionen: Darunter versteht man Kubernetes Plattformen die einen speziellen Zweck erfüllen. Beispiele hierfür wären Kubernetes auf Entwickler-Desktops oder auf einzelnen Fahrzeug Bordcomputern.

Auch Kubernetes an sich, sprich das Upstream Projekt, ist für Kubernetes Nutzer eine legitime Option. Der Aufwand für Installation und Betrieb ist jedoch um ein Vielfaches höher, als bei einer bestehenden Distribution.[96]

[96] Vgl. Tozzi (2020), Onlinequelle [06.12.2020].

Eine spezielle Form von Kubernetes Systemen sind die sogenannten Hybrid-Cluster. Diese haben jeweils einen Teil ihrer Nodes in der Cloud und einen Teil On-Premises. Ein ähnliches Konstrukt stellen Multi-Cloud Cluster dar, deren Nodes sich über mehrere Clouds verteilen. Laut Gartner Inc. zeigt sich eine drastische Steigerung bei der Verwendung von Hybrid-Cloud sowie Public- und Multi-Cloud Clustern. Die zu berücksichtigenden Aspekte sind für Hybrid und Multi-Cloud Cluster dieselben wie für normale Cluster. Die folgende Aufzählung zeigt einige gängige Vertreter der verschiedenen Kubernetes Distributionen auf:[97]

- "pure" Distributionen:
  - Upstream Kubernetes [98]
  - Pharos Cluster[99]
- "plus" Distributionen:
  - Charmed Kubernetes[100]
  - RedHat OpenShift[101]/OKD[102]
  - Rancher Kubernetes Engine[103]
- Kubernetes-as-a-Service (KaaS):
  - Azure Kubernetes Service[104]
  - Amazon Elastic Kubernetes Service[105]
  - Google Kubernetes Engine[106]
- Limited-Purpose Distributionen:
  - MikroK8s[107]
  - K3s[108]

3.2 Container-Runtime

Kubernetes ist grundsätzlich im Stande jede beliebige Container-Runtime zu verwenden, welche das CRI implementiert. Die in diesem Kapitel verwendete Bezeichnung *Container-Runtime*, befindet sich streng genommen zwischen den in Kapitel 2.3.5 beschriebenen Bezeichnungen *Container-Engine* und *Container-Runtime*. Die Kubernetes Dokumentation verwendet den Begriff *Container-Runtime*, welcher streng genommen, im Container Technologie Jargon nicht zutreffend ist. Im Kubernetes Jargon übernimmt die *Container-Runtime* auch Aufgaben, wie das ziehen von Images aus Registries, was streng genommen Aufgabe einer *Container-Engine* ist. Zwar gibt es nicht viele seriöse Implementierungen für diese Komponente, doch die Entscheidung welche zu verwenden ist, muss trotzdem getroffen werden. Welche Eigenschaften zum Vergleich von Container Runtimes verwendet werden können ist Anhang A.10 zu entnehmen.
Gängige Vertreter für Container Runtimes sind Docker[109], CRI-o[110] und ContainerD[111].

3.3 Netzwerk

Die Kommunikation von einzelnen Containern innerhalb eines Pods erfolgt direkt, sprich über *localhost*, dementsprechend teilen sich die Container auch die verfügbaren Ports. Jeder Pod eines Kubernetes Clusters besitzt eine IP Adresse. Kubernetes lagert die Durchführung der Netzwerk-Aufgaben über eine Schnittstelle, das Container Network Interface (CNI) aus.

[97] Vgl. Goasduff (2019b), Onlinequelle [06.12.2020].
[98] weitere Informationen unter: https://github.com/kubernetes/kubernetes
[99] weitere Informationen unter: https://github.com/kontena/pharos-cluster
[100] https://ubuntu.com/kubernetes/docs/overview
[101] https://www.openshift.com/
[102] weitere Informationen unter: https://github.com/openshift/origin
[103] weitere Informationen unter: https://github.com/rancher/rke
[104] https://azure.microsoft.com/en-us/services/kubernetes-service/
[105] https://aws.amazon.com/de/eks/
[106] https://cloud.google.com/kubernetes-engine
[107] weitere Informationen unter: https://github.com/ubuntu/microk8s
[108] weitere Informationen unter: https://github.com/rancher/k3s
[109] weitere Informationen unter: https://github.com/docker/docker-ce
[110] weitere Informationen unter: https://github.com/cri-o/cri-o
[111] weitere Informationen unter: https://github.com/containerd/containerd

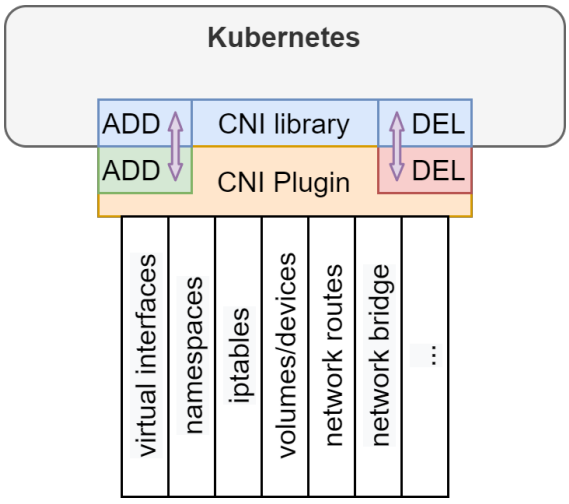


Abb. 3.1:
CNI Plugin im Container Stack
Quelle: Eigene Darstellung

Das Sicherstellen der Netzwerkverbindung zwischen den Pods wird somit über sogenannte CNI Plugins durchgeführt. Kubernetes hat eine CNI Library, welche auf die CNI Plugins zugreift und damit alle nötigen Einstellungen vornimmt, damit die Container eine funktionierende Netzwerkanbindung haben. In Abbildung 3.1 ist schematisch dargestellt wo sich das CNI im Container Stack befindet. Die Bezeichnungen "ADD" und "DEL" sollen schematisch die Schnittstelle zum Hinzufügen und Löschen von Containern, bzw. deren Netzwerkeinstellungen darstellen.

Die wichtigen drei Aspekte, die ein CNI Plugin erfüllen können muss sind[112]:

- Jeder Pod kann mit allen anderen Pods auf allen anderen Nodes ohne NAT[113] kommunizieren.
- Pods auf einer Node können mit allen Pods dieser Node kommunizieren.
- Pods im *Host Netzwerk* einer Node können mit allen anderen Pods aller anderen Nodes ohne Network Address Translation (NAT) kommunizieren.

Grundsätzlich erhält jeder Pod nur eine Netzwerkschnittstelle, es gibt jedoch auch CNI Plugins die mehrere Netzwerk Schnittstellen pro Pod zulassen[114]. Durch welche Eigenschaften CNI Plugins charakterisiert und unterschieden werden können ist in Anhang A.11 ersichtlich.
Gängige Vertreter von CNI Plugins sind Flannel[115], Calico[116], Weave Net[117] und Multus[118].

3.4 Ingress Controller

Unter *Ingress* versteht man ein Kubernetes API Objekt, das externen Zugriff auf Kubernetes interne Services ermöglicht. Typischerweise erfolgt der Zugriff über HTTP(S). Das *Ingress* Objekt selbst gibt lediglich vor wie der Zugriff auszusehen hat, technisch umgesetzt wird dies durch einen sogenannten *Ingress Controller*. Der Ingress Controller kann im IT Jargon auch als Reverse Proxy verstanden werden. Ein Ingress Objekt gibt an welche externen Requests auf welche internen Services weitergeleitet werden sollen. Diese Logik stützt sich hauptsächlich auf den angeforderten Hostnamen und Pfad der URL.

[112] Luksa (2018), vlg. p.335ff.
[113] technische Spezifikation: https://www.ietf.org/rfc/rfc2663.txt
[114] The Linux Foundation (2020h), Onlinequelle [06.12.2020]; Schmitt (2020), Onlinequelle [06.12.2020].
[115] weitere Informationen unter: https://github.com/coreos/flannel
[116] weitere Informationen unter: https://github.com/projectcalico/cni-plugin
[117] weitere Informationen unter: https://github.com/weaveworks/weave
[118] weitere Informationen unter: https://github.com/intel/multus-cni

Ingress Controller können auch den TLS Teil einer Verbindung übernehmen, sprich HTTPS. In Anhang A.12 finden sich Eigenschaften mit denen Ingress Controller charakterisiert und verglichen werden können, sowie ein Beispiel eines Ingress Objektes in YAML Notation in Anhang A.4.

Gängige Vertreter für Ingress Controller sind NGINX[119], Traefik[120], Contour[121], Ambassador[122] und Kong[123]. Auf Seite 27 in Abbildung 2.19 ist eine schematische Darstellung einer Ingress Regel ersichtlich.

3.5 Storage

Persistenter Speicher, auch Storage genannt, wird benötigt um Daten über die Laufzeit des Containers, bzw. Pods, hinaus zu speichern. Die CNCF Storage Special Interest Group (SIG) beschreibt in ihrem Storage Landscape Whitepaper[124] zwei Zugriffsarten für Storage, *Volumes* und *Application API*. Unter *Volumes* werden *Block-* und *Filesystem-Storage* verstanden, welche jeweils als *Volume* in einen Container eingebunden werden. Diese Möglichkeit Storage zu verwenden wurde bereits in Kapitel 2.3.4 erklärt. Unter *Application API* Storage wird Speicher verstanden, der über eine Programmierschnittstelle angesprochen wird. Dazu gehören *Object Storage*, *Key-Value Stores* sowie Datenbanken. In dieser Betrachtung wird auf *Key-Value Stores* und Datenbanken nicht eingegangen, da diese eigenständige Applikationen darstellen und nicht Teil eines Kubernetes Clusters sind.

Kubernetes hat grundsätzlich zwei Treiber-Arten um Container mit Storage Volumes zu verbinden:

- Sogenannte *in-tree* Storage Treiber, welche Teil des eigentlichen Kubernetes Source Codes sind.
- Sogenannte *Container Storage Interface (CSI)* Treiber, welche Plugins Darstellen die eine standardisierte Schnittstelle implementieren.

Seit Kubernetes Version v1.13 ist CSI ein ausgereiftes Feature, man spricht von General Availability (GA).[125] Mittlerweile werden keine neuen *in-tree* Storage Treiber mehr entwickelt und die bestehenden soweit als möglich auf CSI umgestellt. Aus diesem Grund liegt der Fokus der Betrachtung dieser Arbeit auf CSI Implementierungen.[126]

Volumes können Containern in einem Kubernetes Cluster auf zwei Arten zugänglich gemacht werden:

- Ein bestehendes Volume kann direkt über den jeweiligen Treiber (*in-tree* oder *CSI*) in den Container eingebunden werden.
- Ein sogenannter PVC kann eingebunden werden, welcher im Hintergrund auf ein PV verweist. PVCs und PVs wurde bereits in Kapitel 2.4.3 auf Seite 24 kurz erwähnt. Das PV kann entweder im Vorhinein dem Cluster zur Verfügung gestellt, oder dynamisch provisioniert werden. Auch das PVC/PV Konzept baut auf den *in-tree* oder *CSI* Storage Treibern auf, bzw. verwendet diese. Für weitere Informationen zur Funktionsweise von PVs und PVCs sei an dieser Stelle auf entsprechende Fachliteratur und Dokumentation verwiesen.[127]

Bei den eingebundenen Volumes kann es sich entweder um Block-Storage oder File-Storage handeln. Hierbei stellt File-Storage ein Filesystem zur Verfügung wie man es auch von Desktop Rechnern kennt. Das bedeutet man kann Verzeichnisse und Dateien verwenden, welche vom Filesystem wiederum in Block-Storage abgelegt werden.

[119] weitere Informationen unter: https://github.com/kubernetes/ingress-nginx
[120] weitere Informationen unter: https://github.com/containous/traefik
[121] weitere Informationen unter: https://github.com/projectcontour/contour
[122] weitere Informationen unter: https://github.com/datawire/ambassador
[123] weitere Informationen unter: https://github.com/Kong/kong
[124] Vgl. Chircop (2020), Onlinequelle [06.12.2020].
[125] Vgl. Ali (2019), Onlinequelle [06.12.2020].
[126] Vgl. Zhu (2019), Onlinequelle [06.12.2020].
[127] weitere Informationen unter: https://kubernetes.io/docs/concepts/storage/persistent-volumes/

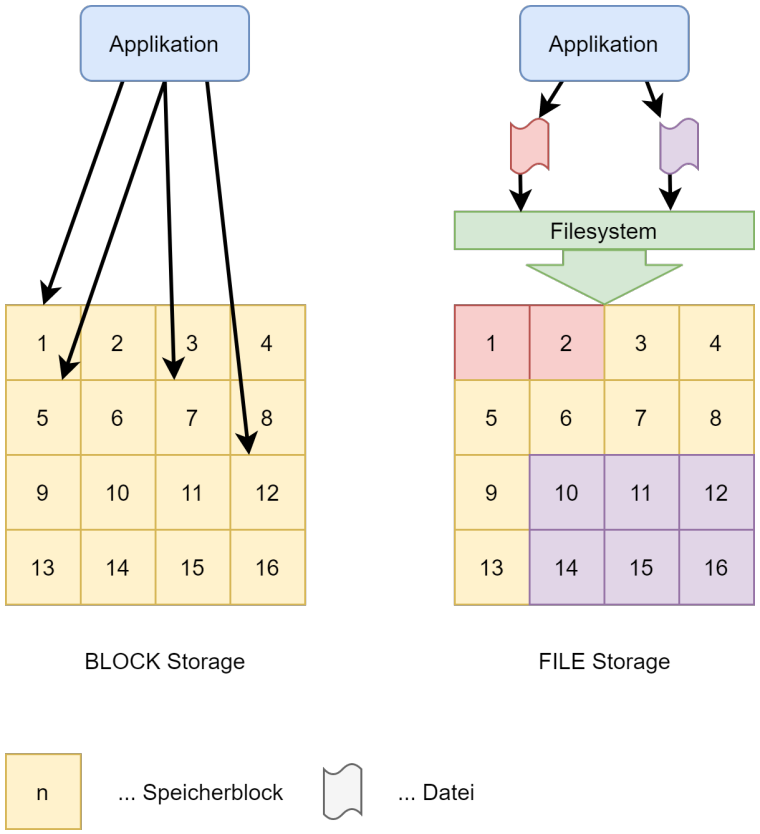


Abb. 3.2:
Schematische Darstellung von Block- und File-Storage
Quelle: Eigene Darstellung

Block-Storage bietet direkten Zugriff auf Speicherblöcke und stellt die Basis für Filesystems, sowie Applikationen die ihren Speicher selbst verwalten (zB Datenbanken), dar. In Abbildung 3.2 ist schematisch der Unterschied zwischen Block- und File-Storage dargestellt. Welche Speicherart benötigt wird ist in der Volume-Konfiguration im Pod oder PV angegeben.

Gängige Vertreter für Block- und File-Storage sind StorageOS (File-Storage)[128], Longhorn (Block-Storage)[129] oder Ceph (Block & File)[130].

Die Betrachtung von *Object Storage* ist nicht primär Teil dieser Arbeit, da dieser meist als Service bezogen wird. Es gibt erste Bestrebungen und Implementierungen von Filesystemen welche, anstatt von *Block Storage* auf *Object Storage* basieren. Diese könnten dementsprechend auch als CSI Plugin verwendet werden. Die klassische Verwendung von *Object Storge*, bezogen als Service über zB HTTP(S) ist nicht Teil der Betrachtung dieser Arbeit, das dies eine Überlegung auf Applikationsseite darstellt.

In Anhang A.13 sind Vergleichsmerkmale bzw. Eigenschaften für Storage Systeme angeführt.

[128] weitere Informationen unter: https://storageos.com/
[129] weitere Informationen unter: https://github.com/longhorn/longhorn
[130] weitere Informationen unter: https://github.com/ceph/ceph-csi

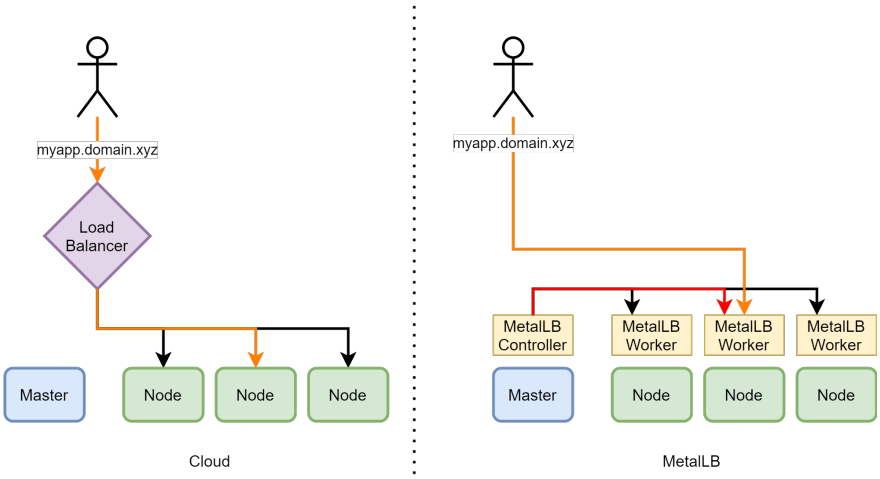


Abb. 3.3:
Schematische Darstellung eines Cloud- und MetalLB-Loadbalancers
Quelle: Eigene Darstellung, angelehnt an The Linux Foundation (2020a)

3.6 Load Balancer

Wie in Kapitel 3.4 angemerkt, können HTTP(S) basierte Services mit der Ingress Ressource von außerhalb des Clusters erreichbar gemacht werden. Services welche nicht auf HTTP(S) basieren, wie beispielsweise Datenbanken oder Messaging-Systeme, benötigen eine andere Möglichkeit um öffentlich zugänglich gemacht werden zu können. Dieses Kapitel betrifft konkret den Kubernetes Service vom Typ *LoadBalancer*, welcher hierfür verwendet werden kann. Dabei wird einem Service eine IP Adresse zugewiesen, welche von außerhalb des Clusters erreichbar ist. Bei gehosteten und Public-Cloud Kubernetes Distributionen kümmert sich der Provider um die Implementierung dieses Services. Aus Kundensicht ist es also egal, und auch nicht veränderbar, wie die Komponente implementiert wird. Bei On-Premises Installationen muss das Unternehmen diese Funktionalität selbst implementieren.

Grundsätzlich gibt noch andere Möglichkeiten wie Services von außerhalb des Clusters erreichbar gemacht werden können. Ein *NodePort* Service oder eine *ExternalIP* sind zwei Möglichkeiten um dies zu erreichen. Beide haben jedoch entscheidende Nachteile, wobei an dieser Stelle auf weiterführende Literatur verwiesen werden soll.[131]

Eine beliebte Lösung für die Implementierung der *LoadBalancer* Service Funktionalität auf On-Premises clustern, stellt das Projekt *MetalLB* dar. Dabei handelt es sich um eine Software welche die Anforderungen des *LoadBalancer* Services implementiert und somit die Verwendung von *LoadBalancer* Service Objekten ermöglicht. Bei dieser Variante kümmert sich einer der Worker Nodes darum, die Anfragen welche an die externe IP Adresse des Services gehen, zu beantworten. Welcher Node für welche externe IP zuständig ist wird von MetalLB gesteuert. In Abbildung 3.3 sind ein *LoadBalancer* in der Cloud, sowie On-Premises mit MetalLB dargestellt. Für eine detaillierte Erklärung der konkreten Funktionsweise von MetalLB sei an dieser Stelle auf die entsprechende Dokumentation verwiesen.[132] Im Gegensatz dazu würde bei gehosteten oder Public-Cloud Clustern ein vorgelagertes System die Requests annehmen und dem richtigen Node zustellen.

[131] Vgl. The Linux Foundation (2020a), Onlinequelle [06.12.2020].
[132] weitere Informationen unter: https://metallb.universe.tf/concepts/

3.7 Service Mesh

Um besser überwachen, kontrollieren und verwalten zu können welche Services in einem Cluster miteinander kommunizieren, hat sich die Verwendung sogenannter Service Meshes etabliert. Unter einem *Service Mesh* versteht man einen zusätzlichen Infrastruktur Layer, welcher sich um die Kommunikation zwischen Pods im Cluster kümmert. Damit kann kontrolliert werden welche Pods miteinander kommunizieren dürfen, Performance und Zugriffe können überwacht und Fehler leichter nachverfolgt werden. Einige der weiteren Funktionalitäten die Service Meshes liefern sind[133]:

- Service Discovery: Erleichtert/Ermöglicht das Erkennen neuer Services im Cluster.
- Load Balancing: Lastverteilung von Anfragen gemäß bestimmter Algorithmen auf andere Pods.
- Communication Resiliency: Darunter versteht man das Erhöhen der Belastbarkeit eines Service mithilfe verschiedener Pattern und Algorithmen.
- Security: Service Meshes erlauben die Verschlüsselung und Einschränkung von Verbindungen zwischen Pods.
- Observability: Überwachung wird durch das Sammeln von Metriken und Log-Daten ermöglicht.
- Routing Control: Dies beschreibt die Möglichkeit Datenströme gemäß bestimmter Regeln umleiten zu können.
- API: Service Meshes stellen oft eine Programmierschnittstelle für diverse Automatisierungen zur Verfügung.

Die meisten Service Meshes bestehen logisch gesehen aus einer Kombination aus *Data Plane* und *Control Plane*. Die *Data Plane* sind Programme welche als Container in alle Pods injiziert werden. Sie übernehmen die Netzwerkkommunikation des Pods, was für den Pod jedoch transparent ist. Weil die Container als eine Art Beiwagen im Pod fungieren spricht man von sogenannten Sidecar Container.[134] Diese Art Container in Pods zu injizieren wird auch für andere Tätigkeiten, wie das Sammeln von Log- oder Metrik-Daten verwendet. Diese Sidecar Container werden von der *Control Plane* gesteuert und überwacht. In Abbildung 3.4 wird diese Architektur schematisch dargestellt. Der Pod selbst weiß gar nicht, dass er Teil eines Meshes ist, für ihn ist die Kommunikation unverändert. Es ist wichtig an dieser Stelle anzumerken, dass Service Meshes kein nativer Teil eines Kubernetes Cluster sind, und auch keine Voraussetzung für den Betrieb eines Cluster. Seriöse produktive Cluster setzen heute jedoch nahezu ausschließlich ein Service Mesh voraus, da dieses ein großes Set an wichtigen Zusatzfeatures bringt.

Gängige Vertreter für Service Meshes sind Istio[135], linkerd[136], Maesh[137] oder Consul[138]. Analog zu Storage (CSI) und Netzwerk (CNI) ist auch für Service Meshes eine Schnittstelle vorgesehen, das Service Mesh Interface (SMI). Derzeit ist das SMI ein CNCF Sandbox Projekt[139] und somit noch im Frühstadion.

3.8 DNS

Ein Domain Name System (DNS) Server ermöglicht es Ressourcen, wie beispielsweise IP Adressen, in einem System einen sprechenden Namen zu geben. Damit können die IP Adressen von Pods und Services im Cluster mit Namen versehen und über diese auch angesprochen werden. Kubernetes stellt für das dynamische Erkennen von Pods und Services, sowie die Möglichkeit diese namentlich ansprechen zu können ein Cluster add-on für DNS zur Verfügung.

[133] Vgl. Buehrle (2019), Onlinequelle [06.12.2020].
[134] weitere Informationen unter: https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar
[135] weitere Informationen unter: https://github.com/istio/istio
[136] weitere Informationen unter: https://github.com/linkerd/linkerd2
[137] weitere Informationen unter: https://github.com/containous/maesh
[138] weitere Informationen unter: https://github.com/hashicorp/consul
[139] weitere Informationen unter: https://smi-spec.io/

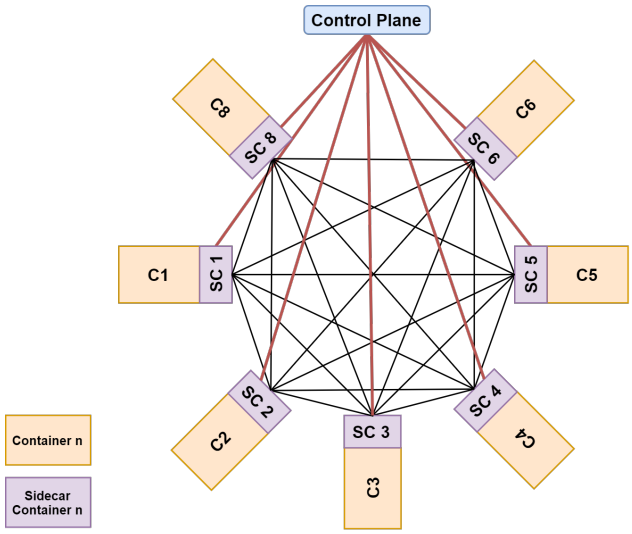


Abb. 3.4:
Schematische Darstellung eines Service Mesh
Quelle: Eigene Darstellung

In Kubernetes Version v1.11 hat das Produkt *CoreDNS* das bisher verwendete *kube-dns* plugin als Standard DNS Modul abgelöst. CoreDNS ist ein DNS Server, welcher modular aufgebaut ist und eine große Anzahl an konfigurierbaren Features mit sich bringt.
Es wäre grundsätzlich möglich entweder das ursprüngliche *kube-dns* Modul, oder einen eigenen DNS Server, anstatt CoreDNS zu verwenden. Es ist jedoch zu empfehlen sich an den Standard von *CoreDNS* zu halten, da dieser laufend weiterentwickelt und von der Community gewartet wird. Grundsätzlich kann es dem Benutzer egal sein, welche DNS Implementierung in Public-Cloud oder Managed-Service-Kubernetes Angeboten verwendet wird. Es sei an dieser Stelle jedoch angemerkt, dass auch einige der großen Public-Cloud-Provider auf CoreDNS setzen. DNS ist eine Komponente die bei jeder Kubernetes Distribution, sogar upstream Kubernetes, standardmäßig bereitgestellt wird. Die Auswahl an möglichen Implementierungen beschränkt sich grundsätzlich auf *kube-dns*[140] und *CoreDNS*[141], es könnten jedoch auch eigene DNS Server verwendet werden. Da nahezu jeder Faktor für die Verwendung von *CoreDNS*, bzw. des vom Kubernetes Service Provider verwendeten DNS Plugins, spricht, ist eine Betrachtung weiterer Optionen an dieser Stelle nicht sinnvoll[142].

3.9 Logging und Monitoring

Unter Logging wird in diesem Fall das Sammeln, Darstellen, Durchsuchen und Auswerten von Log Daten verstanden. Monitoring und darauf aufbauend Alerting, beschreibt das Sammeln, Verarbeiten, Aufbereiten, Darstellen und Verwerten von Metrik-Daten. Unter Metrik-Daten versteht man alle Daten, die den momentanen Zustand eines Systems darstellen. Typischerweise werden hierfür Ressourcenauslastungen, zeitliche Kennwerte wie Latenzen, Kennzahlen über Kubernetes Objekte sowie etwaige benutzerdefinierte Kennzahlen verwendet. Die beiden Punkte Logging und Monitoring sind streng genommen keine Komponenten eines Kubernetes Clusters, werden jedoch von einer seriösen Umgebung als vorausgesetzt betrachtet.

[140] weitere Informationen unter: https://github.com/kubernetes/dns
[141] weitere Informationen unter: https://github.com/coredns/coredns
[142] Vgl. Belamaric (2018), Onlinequelle [06.12.2020].

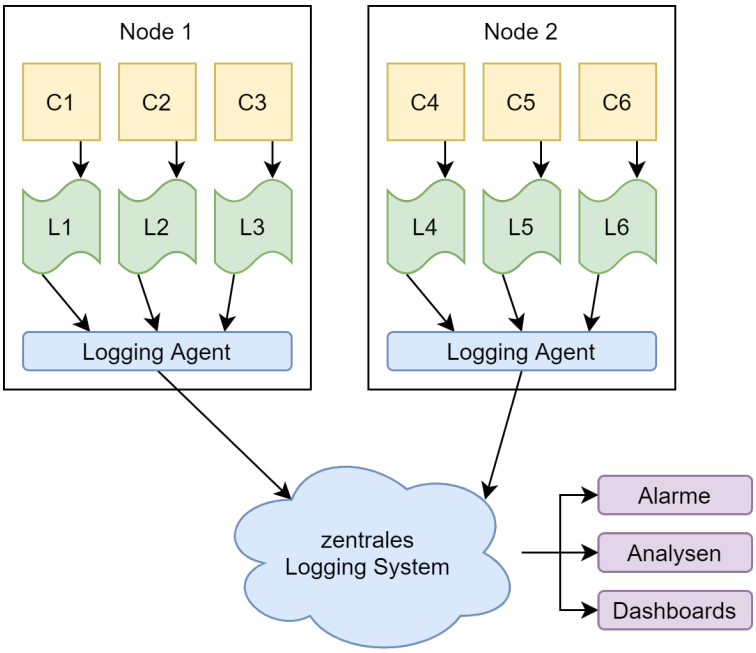


Abb. 3.5:
Schematische Darstellung der Logging-Architektur im Kubernetes Cluster
Quelle: Eigene Darstellung

Logging: Grundsätzlich gibt es in einem Kubernetes Cluster drei Entitäten welche Logs produzieren:

- Applikationen bzw. Container: Die Logdaten der eigentlichen Applikationen die am Cluster laufen.
- Kubernetes System-Komponenten: Die Cluster-Komponenten von Kubernetes selbst.
- Nodes: Das Betriebssystem sowie alle Applikationen die direkt am Node, nicht als Container, laufen.

Alle drei Gruppen schreiben ihre Log-Daten entweder in Files, oder verwenden eigene Logging-Systeme die die Log-Daten speichern. Die Log-Daten können manuell und teilweise mit Kubernetes und Container-Engine Bordmitteln angesehen und verarbeitet werden. Dies ist jedoch vor allem in großen Systemen und über mehrere Log-Dateien und Nodes hinweg sehr unvorteilhaft. Aus diesem Grund haben sich Systeme etabliert, welche alle erdenklichen Logs zusammentragen, zentral speichern und gesamtheitlich zur Verfügung stellen. Das Sammeln von Logs wird häufig mit kleinen Programmen erledigt, welche die Logs meist pro Node, sprich für alle Container auf diesem Node, verarbeiten. Diese Sammelcontainer, auch Agents genannt, verarbeiten die Log-Daten und senden sie anschließend weiter an ein zentrales Logging-System. In Abbildung 3.5 ist diese Architektur schematisch dargestellt. Da diese Systeme nicht ausschließlich auf Kubernetes ausgelegt sind, für den Betrieb eines Clusters nicht essenziell sind und streng genommen nur ein bequemes Add-On darstellen, wird das Thema an dieser Stelle nicht weiter vertieft.[143]

Gängige Vertreter für Logging-Systeme und Logging-Agents sind Elastic Stack(ELK)[144], Splunk[145], Loggly[146], fluentd [147] oder filebeat [148].

Monitoring: Analog dem Logging ist auch systematisches Monitoring kein verpflichtender Teil eines Kubernetes Clusters. Auch hier gilt jedoch, dass jede seriöse Kubernetes Umgebung ein umfangreiches Monitoring voraussetzt.

[143] Vgl. The Linux Foundation (2020n), Onlinequelle [06.12.2020].
[144] weitere Informationen unter: https://www.elastic.co/de/what-is/elk-stack
[145] weitere Informationen unter: https://www.splunk.com/
[146] weitere Informationen unter: https://www.loggly.com/
[147] weitere Informationen unter: https://github.com/fluent/fluentd
[148] weitere Informationen unter: https://github.com/elastic/beats/tree/master/filebeat

Unter Monitoring wird in diesem Fall nicht nur die Überwachung des aktuellen Zustandes des Clusters verstanden, sondern auch die langzeitige Aufbewahrung der Metrik-Daten. Kubernetes selbst hat zwei Komponenten um Metriken zur Verfügung zu stellen:

- Metrics-Server[149]: sammelt Ressourcen-Metriken im gesamten Cluster
- Kube-State-Metrics[150]: sammelt Metriken über Objekte der Kubernetes API

Beide Systeme speichern die aktuellen Metriken nur kurz oder gar nicht, sind also nicht für grafische Aufbereitung, Alerting oder langzeitige Aufzeichnungen verwendbar. Diese Aufgabe übernehmen Systeme wie beispielsweise *Prometheus*, das de-facto Standardprojekt für Monitoring und Alerting im Kubernetes Umfeld. Solche erweiterten Systeme bauen teilweise auf *Metrics-Server* und *Kube-State-Metrics* auf, ergänzen deren Daten jedoch noch um weitere Datenquellen. Auch hier kommen teilweise Sidecar Container zum Einsatz, um mehr oder bessere Metrik-Daten über die jeweiligen Pods zu erhalten. Viele Applikationen stellen jedoch auch nativ Metrik-Daten zur Verfügung, die von Systemen wie *Prometheus* nur mehr eingesammelt werden müssen. Aufbauend auf gesammelten und verarbeiteten Metriken können anschließend Graphen dargestellt (zB mit *Grafana*[151]) oder Alarme ausgelöst (zB mit *Alertmanager*[152]) werden.

Viele Kubernetes Distributionen liefern fertige Lösungen für Monitoring, Alerting und Logging. Welche Lösungen zur Verfügung stehen, bzw. wie diese beschrieben und unterschieden werden können ist nicht Thema dieser Arbeit.

Dieses Kapitel hat aufgezeigt aus welchen Komponenten ein Kubernetes Cluster technisch besteht. Die einzelnen Bestandteile wurden beschrieben, deren Zuständigkeiten aufgezeigt sowie einige gängige Vertreter pro Komponente angeführt. Auch für das Innovationsmanagement ist es interessant zu verstehen woraus ein Cluster besteht. Mit diesem Wissen lassen sich ggf. neue Möglichkeiten zur Erweiterung oder Optimierung von Services und Produkten finden. Außerdem unterstützt ein gutes Verständnis des technischen Aufbaus eines Clusters dabei, bei potenziellen Innovationen abschätzen zu können, welche Probleme wie gelöst werden könnten.

[149] weitere Informationen unter: https://github.com/kubernetes-sigs/metrics-server
[150] weitere Informationen unter: https://github.com/kubernetes/kube-state-metrics
[151] weitere Informationen unter: https://github.com/grafana/grafana
[152] weitere Informationen unter: https://github.com/prometheus/alertmanager

4 INNOVATIONSGETRIEBENES KUBERNETES CLUSTER DESIGN

In diesem Kapitel wird erklärt wie und warum das Innovationsmanagement in den Kubernetes Designprozess involviert ist. Es wird außerdem beschrieben mit welcher Methodik die Vorgehensweise zum Clusterdesign, sowie die dazu nötigen Anforderungen erarbeitet wurden. Des Weiteren werden diese Vorgehensweise, sowie die genannten Anforderungen beschrieben.

Das Innovationsmanagement begleitet eine Idee von ihrer Entstehung, über die Umsetzung bis zur Markteinführung. Parallel dazu entscheidet es in den Bereichen des strategischen Managements, Technologie- und F&E Managements über das Technologieportfolio des Unternehmens mit. Dabei werden Technologien und Wissen erfasst, gespeichert und verwertet, sowie kategorisiert, beobachtet und eingeschätzt. Zwei gängige Unterscheidungen werden nach Einsatzgebiet oder Lebenszyklus der Technologie getroffen.[153] Container und Kubernetes wären gemäß dieser Kategorisierung wahrscheinlich als Produkttechnologien und Schlüsseltechnologien einzuordnen.

Eine gängige Methode zur fortlaufenden Überwachung von Technologien stellt das Technologie-Radar dar. In seinem Technologieradar hat das Unternehmen ThoughtWorks Kubernetes 2016 noch im Versuchsstadium gesehen, bereits Ende 2017 wurde eine Verwendung stark empfohlen. Kubernetes schien als eigene Technologie Ende 2018 das letzte Mal in diesem Radar auf und wurde seither als de facto Standard gehandelt.[154] Der aktuelle Technologieradar der CNCF, ersichtlich in Abbildung 4.1, beschäftigt sich mit Technologien aus dem Bereich Continuous Delivery. Damit werden Automatisierung im Softwareentwicklungsprozess umgesetzt. Die zwei relevantesten Projekte laut diesem Radar sind Flux und Helm. Dabei handelt es sich um Projekte welche speziell für Kubernetes entwickelt wurden bzw. darauf aufbauen. Kubernetes ist somit eine der aktuell essentiellen Schlüsseltechnologien im Bereich der Applikationsplattformen und bereits die Basis für weitere Projekte.

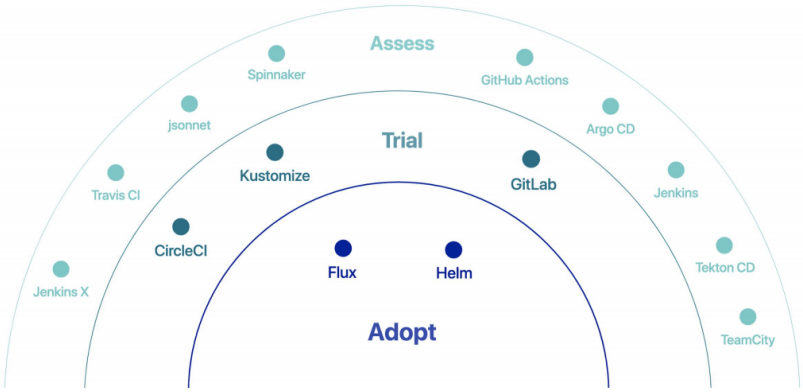


Abb. 4.1: **CNCF Technologie Radar**
Quelle: Hung (2020)

Das Innovationsmanagement ist bezüglich der Technologieauswahl an mindestens zwei Stellen beteiligt. Einerseits wird entschieden welche Technologie für die direkte Implementierung der Idee verwendet wird, andererseits wird auch über alle Umfeldtechnologien entschieden. Das bedeutet beispielsweise, dass das Technologiemanagement vorgibt, dass die Applikation in Java geschrieben werden muss und in Docker Containern laufen soll. Speziell die Wahl der Plattform ist für diese Arbeit ausschlaggebend, da dieser Bereich von Kubernetes beansprucht wird. Es sind für das Innovationsmanagement jedoch nicht nur technische Punkte interessant, auch wirtschaftliche oder organisatorische Punkte werden berücksichtigt.

[153] Vgl. Vahs (2015), S. 26f.
[154] ThoughtWorks Inc. (2020).

Das Innovationsmanagement ist ein essentieller Teil dauerhafter Verbesserung in allen Bereichen. Es geht also nicht darum nur einmal eine Entscheidung zu treffen, sondern sowohl die Produkte des Unternehmens, als auch dessen Infrastruktur und Plattform weiterzuentwickeln. Selbstverständlich gehören dazu auch alle beteiligten Prozesse und organisatorischen Einheiten. Das Innovationsmanagement ist also durch seine Beteiligung am Technologiemanagement eines Unternehmens auch bei der Gestaltung der Applikationsplattformen gefragt. In dieser Funktion ist es auch am Design der Kubernetes Umgebung beteiligt.

Wie am Anfang der Arbeit, in der Erörterung der Aufgabenstellung schon dargestellt, hat sich in der betrieblichen Praxis folgendes gezeigt: Es treten vermehrt Probleme auf wenn keine strukturierte Vorgehensweise definiert ist, nach der das Design der Kubernetes Plattform durchgeführt wird. Beispielsweise ist aufgefallen, dass bereits der Designstart, auf Grund fehlender Vorgehensweise, verzögert werden kann. Außerdem ist die Wahrscheinlichkeit hoch, dass im Laufe der Entwicklung und des Produktlebenszyklus, Anforderungen aufkommen die neue Clusterfunktionalitäten erfordern. Dies hat wiederum zur Folge, dass der gesamte Cluster oder Teile davon getauscht, verlagert oder umgebaut werden müssen. Je früher die Anforderungen an das System also vorliegen und berücksichtigt werden können, desto besser kann das System entworfen werden. Ziel ist es also eine strukturierte Vorgehensweise zu finden, die mit Fokus auf die Idee bzw. Innovation sowie deren Anforderungen, ein bestmögliches Kubernetes Cluster Design sicherstellt. Speziell dem Innovationsmanagement geht es hierbei jedoch nicht darum den Cluster selbst zu entwerfen oder explizit Produkte auszuwählen, dies ist Aufgabe einer anderen Partei. Es ist jedoch in der Verantwortung des Innovationsmanagements, die umsetzende Partei mit allen nötigen Informationen und Anforderungen zu versorgen.

Die Anforderungen an ein System stellt im klassischen Sinn meist ein Kunde, wobei der Kundenbegriff in diesem Fall zwei Parteien umfasst. Einerseits ist das Entwicklerteam welches die Innovation entwickelt der Kunde, da es technische Anforderungen an das System stellt. Andererseits ist der Innovationsverantwortliche, zum Beispiel ein Produkt- oder Projekt- oder Innovationsmanager, der Kunde. Dieser Kunde stellt im Vergleich zu den Entwicklern Anforderungen wirtschaftlicher, operativer und regulativer Natur. Der Endkunde, welcher schlussendlich die resultierende Innovation die am Cluster betrieben wird konsumiert, ist in dieser Betrachtung ausgenommen. Für den Endkunden ist der Kubernetes Cluster gar nicht als solcher ersichtlich. Er verlangt nur nach einem qualitativ hochwertigen und inhaltlich vollständigem Produkt bzw. Service. Die Anforderungen des Kunden werden in dieser Betrachtung durch den Innovationsverantwortlichen repräsentiert.
Im Zuge der Ausarbeitung der Vorgehensweise zum Clusterdesign, sowie der Erarbeitung des Anforderungsbogens, hat sich die Unterteilung der Anforderungen in die folgenden drei Gruppen bewährt.[155]

- Rahmenbedingungen: Rahmenbedingungen welche extern vorgegeben sind und nicht beeinflusst werden können.
- Funktionale Anforderungen: Anforderungen an die harten technischen Funktionalitäten des Clusters.
- Qualitätseigenschaften: Eigenschaften nach denen sich die Qualität des Clusters, bzw. einzelner Komponenten, unterscheiden lässt.

Bevor weiter auf die genannten drei Anforderungsgruppen eingegangen wird zeigt das folgende Kapitel die gesamte Vorgehensweise für das Cluster Design auf. Es wird erklärt welche Schritte durchlaufen werden und mit welcher Methode die Vorgehensweise entwickelt wurde.

[155] Vgl. Patig (2018), Onlinequelle [06.12.2020].

4.1 Strukturierte Vorgehensweise beim Kubernetes Design

Bei den Recherchen zur vorliegenden Arbeit hat sich schnell herausgestellt, dass Kubernetes zwar sehr viele Möglichkeiten und Vorteile bietet, jedoch nicht für jeden Anwendungsfall passend ist. Daher ist der erste logische Schritt beim Entwurf jedes Clusters die Beantwortung der Frage, ob überhaupt ein Cluster benötigt wird. Sollte sich das Unternehmen für Kubernetes entscheiden, dreht sich die unmittelbar nächste Frage um den Aufbau des Clusters. Es wird also gefragt wie der Cluster aussehen soll und was er können soll. Diese Frage wird im Weiteren durch die Spezifikation von Anforderungen, basierend auf der Innovation für welche der Cluster überhaupt entworfen wird, beantwortet. Der nächste Schritt ist die Auswahl konkreter Produkte und das Dimensionieren des Clusters, sowie dessen Installation und Übergabe. Abschließend muss das neue System noch sauber in die Ablauf- und Aufbauorganisation des Unternehmens integriert werden.

Bevor weiter auf die einzelnen Anforderungen eingegangen wird, ist es wichtig die verwendeten Begriffe eindeutig voneinander abzugrenzen. Unter *Anforderungen* werden die Erwartungen des Kunden an die Funktionalitäten und Eigenschaften des Clusters, sowie nicht verhandelbare externe Gegebenheiten, verstanden. Diese Erwartungen und Gegebenheiten müssen zwangsläufig immer vor der Definition von *Anforderungen* bekannt sein. In diesem Sinne stellen Anforderungen immer die Frage "Was muss der Cluster leisten können?". Unter *Komponenten* werden die konkreten technischen Einzelteile des Clusters verstanden, welche die *Anforderungen* erfüllen müssen. Jede *Komponente* besitzt wiederum ein Set an *Eigenschaften* welches beeinflusst ob und wie gut die *Anforderungen* erfüllt werden. Für jede *Komponente* gibt es verschiedenste Produkte/Projekte, die sogenannten *Implementierungen*. Nimmt man beispielsweise die *Komponente* "Storage Provider", so gibt es dafür diverse Produkte. Jedes dieser Produkte ist eine *Implementierung* der Komponente "Storage Provider". Anhand der jeweiligen *Eigenschaften* und der definierten *Anforderungen*, kann nun die optimale *Implementierung* ausgewählt werden.

Um die Frage zu klären, welche Anforderungen an den Cluster gestellt werden können, wurden die folgenden Schritte durchgeführt. Als Erstes wurden die Komponenten des Clusters, sowie alle möglichen Eigenschaften welche diese Komponente beschreiben, identifiziert. Diese Komponenten wurden bereits in Kapitel 3 beschrieben. Anhand der vorliegenden Komponenten und deren Eigenschaften konnten bestimmte individuelle Funktionalitäten gefunden werden, welche im weiteren Schritt als Anforderung formuliert werden könnten. Als Beispiel hierfür kann die Möglichkeit der Verschlüsselung von gespeicherten Daten dienen, die nur manche Storage Produkte bieten. Die auf diese Weise identifizierten Anforderungen sind größtenteils technischer Natur. Im zweiten Schritt wurden diverse Case-Studies und Erfahrungsberichte analysiert, welche Aufschluss darüber geben welche Anforderungen andere Unternehmen an ihre Cluster stellen. Speziell die Kubernetes Dokumentation[156] bietet einige sehr gut aufbereitete Case Studies mit deren Hilfe weitere, auch nicht technische Anforderungen identifiziert werden konnten. Abschließend wurden noch einige Anforderungen aus der betrieblichen Praxis und aus informellen Expertengesprächen gesammelt. Aus diesen drei Informationsquellen konnten die Anforderungen erarbeitet werden, welche in den folgenden Kapiteln näher beschrieben werden.

Um die erarbeiteten Anforderungen sinnvoll spezifizieren zu können muss natürlich bereits bekannt sein, welche Anforderungen die betrachtete Innovation überhaupt stellt. Es sei an dieser Stelle also angemerkt, dass davon ausgegangen wird, dass für die umzusetzende Innovation zumindest ein greifbares Konzept vorliegt. Natürlich wäre es von Vorteil, wenn schon ein lauffähiger Prototyp, oder gar eine fertige Applikation verfügbar wäre.

[156] weitere Informationen unter: https://kubernetes.io/case-studies/

Zumindest ein Konzept ist jedoch vonnöten, um die Anforderungen an die Kubernetes Plattform auch sinnvoll erfassen zu können. Des Weiteren wäre es auch vorteilhaft, wenn vor der Designphase des Clusters eine Analyse des zu erwartenden Marktes erfolgt. Dies ist sehr interessant um abschätzen zu können von welchem Kundenvolumen, bzw. welcher Rechenlast auszugehen ist. Auch die geografische Lage der erwarteten Endkunden ist relevant für das Design des Clusters.

Bevor man sich aber den spezifischen Anforderungen eines Kubernetes Systems widmet sollte sorgfältig abgeklärt werden, ob Kubernetes überhaupt die richtige Technologie für den vorliegenden Use-Case ist. Obwohl Kubernetes sehr viele Möglichkeiten bietet, müssen doch einige Punkte für eine sinnvolle Verwendung sichergestellt werden. Der erste Schritt beim Design einer Kubernetes Umgebung ist eine Evaluierung, ob Kubernetes überhaupt die richtige Wahl für den vorliegenden Use-Case ist. Wie dabei vorzugehen ist, bzw. welche Fragen Aufschluss darüber geben ob die Verwendung von Kubernetes sinnvoll ist, wird in Kapitel 4.2 beschrieben. Das Ergebnis dieser Überlegung ist eine Entscheidung für oder gegen die Verwendung von Kubernetes. Sobald die Frage über die grundlegende Verwendung von Kubernetes geklärt ist, kann damit begonnen werden die Anforderungen an das System zu spezifizieren.

Im ersten Schritt des Designprozesses werden die Rahmenbedingungen definiert, welche sich durch den restlichen Prozess ziehen. Diese liegen meist schon in Form von Unternehmensstrategien, Richtlinien, Gesetzgebungen oder anderen Vorgaben vor. Aufgabe des Innovationsmanagers ist es, diese Rahmenbedingungen so vollständig wie möglich zusammenzutragen und zu verschriftlichen. Somit kann sichergestellt werden, dass sich in den späteren Phasen des Designprozesses keine gröberen Fehlentscheidungen ergeben.

Der nächsten Schritt beim Entwurf des Clusters ist die Definition der funktionalen Anforderungen. Dabei geht es hauptsächlich um technische Vorgaben, die sich meist auf Grund der Funktionsweise der zu Entwickelnden Innovation ergeben. An dieser Stelle sei angemerkt, dass es gewisse funktionale Anforderungen gibt, welche vom Innovationsmanagement alleine höchstwahrscheinlich nicht definiert werden können. Dies liegt daran, dass das Innovationsmanagement meist nicht das nötige technische Know-How aufweist, um diese Anforderungen sinngemäß zu spezifizieren. Daher sollte bei Anforderungen dieser Natur technische Hilfe, zum Beispiel durch eine Entwicklungsabteilung, hinzugezogen werden. Die Anforderungen in den folgenden Sektionen sind jeweils mit [INNO] oder mit [TECH] gekennzeichnet. Anforderungen mit der Kennzeichnung [INNO] sollten vom Innovationsmanagement alleine beantwortet werden können. Für Fragen die mit [TECH] gekennzeichnet sind, empfiehlt es sich technische Unterstützung hinzuzuziehen. Der zweite Schritt ist also die Definition jener funktionalen Anforderungen, die noch vom Innovationsmanagement alleine spezifiziert werden können. Im dritten Schritt werden jene funktionalen Anforderungen definiert, die mehr technisches Know-How voraussetzen als vom Innovationsmanagement gefordert werden kann.

Der vierte Schritt des Entwurfsprozesses ist die Schnittstelle zwischen Innovationsmanagement und jener Partei die den Cluster später installieren und betreiben wird. An dieser Stelle werden die verschriftlichten Anforderungen an die umsetzende Partei übergeben, welche im letzten Schritt das eigentliche Design durchführen wird. Anhand des übergebenen Anforderungsbogens kann die umsetzende Partei nun strukturiert entscheiden, wie der Cluster auszusehen hat und welche Produkte jeweils zu wählen sind. Typischerweise wird der Cluster von der IT-Infrastruktur Abteilung des Unternehmens oder einem externen Dienstleister entworfen, installiert und betrieben. Bei der Erhebung der nötigen Informationen der jeweils in Frage kommenden Produkte kann das Innovationsmanagement natürlich soweit als möglich eingebunden werden. Speziell die erarbeiteten Qualitätseigenschaften sollen dabei helfen potenziell in Frage kommende Produkte miteinander vergleichen zu können. Ein Vorschlag dafür, wie die konkrete Auswahl von Produkten im letzten Schritt aussehen könnte ist in Anhang A.18 angeführt.

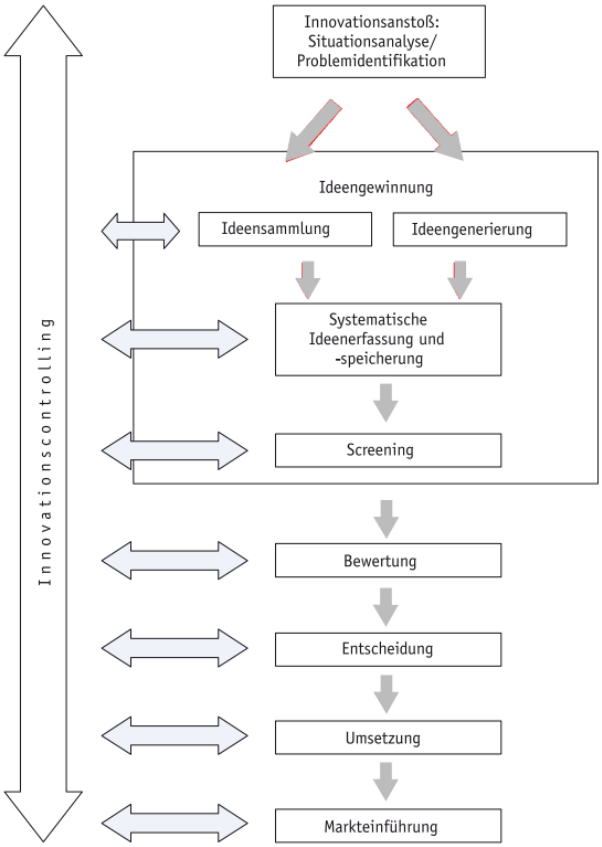


Abb. 4.2: **Innovationsprozess nach Vahs/Brem**
Quelle: Vahs (2015), S. 230

Dafür wurde jeder Anforderung eine eindeutige, ansteigenden Identifikationszahl mit einem definierten Präfix, der sogenannte Identifier, zugewiesen. Die *funktionalen Anforderungen* haben den Präfix *AF*, *Qualitätseigenschaften* den Präfix *AQ* und *Rahmenbedingungen* den Präfix *AR*.

Der fünfte und letzte Schritt im Designprozess ist die Integration des neuen Systems im Unternehmen. Damit ist zum Beispiel das Einfügen in eine bestehende Supportorganisation, oder das generelle Zuweisen von Zuständigkeiten gemeint. Auch die Aufnahme in etwaige Verwaltungssysteme und Dokumentationssysteme, wie sie beispielsweise gemäß ITIL Standard vorgeben werden, ist hier durchzuführen. Da dieser Punkt für jedes Unternehmen unterschiedlich ausfällt, wird an dieser Stelle nicht weiter darauf eingegangen. Es ist jedoch essentiell, dass das neue Kubernetes System aktiv in das Unternehmen integriert wird, um optimal genutzt und Reibungspunkte verhindern zu können.

An dieser Stelle soll noch angemerkt werden an welcher Stelle im Innovationsprozess der vorgestellte Prozess zum Einsatz kommt. In Abbildung 4.2 ist der Innovationsprozess nach Vahs/Brem ersichtlich. Dieser stellt den Entstehungsprozess einer Innovation von der Idee bis zur Markteinführung dar. Das Kubernetes Design, inklusive der Installation des Systems, sollte im Innovationsprozess bestenfalls schon vor dem Schritt der Umsetzung durchgeführt sein. Spätestens jedoch im Schritt der Umsetzung sollte eine Entscheidung bzgl. der Applikationsplattform getroffen werden. Das heißt hier wird diskutiert ob Kubernetes verwendet wird und wie das System aussehen soll, falls man sich dafür entscheidet. Damit kann man den größten Nutzen aus dem System ziehen, indem man es schon bei der Entwicklung bzw. Umsetzung der neuen Innovation miteinbezieht.

In Abbildung 4.3 ist der Prozessablauf des gesamten Clusterdesigns aufgezeigt. Die folgenden vier Kapitel behandeln die einzelnen Phasen bzw. Anforderungsgruppen näher und werden mit einem praktischen Beispiel abgeschlossen.

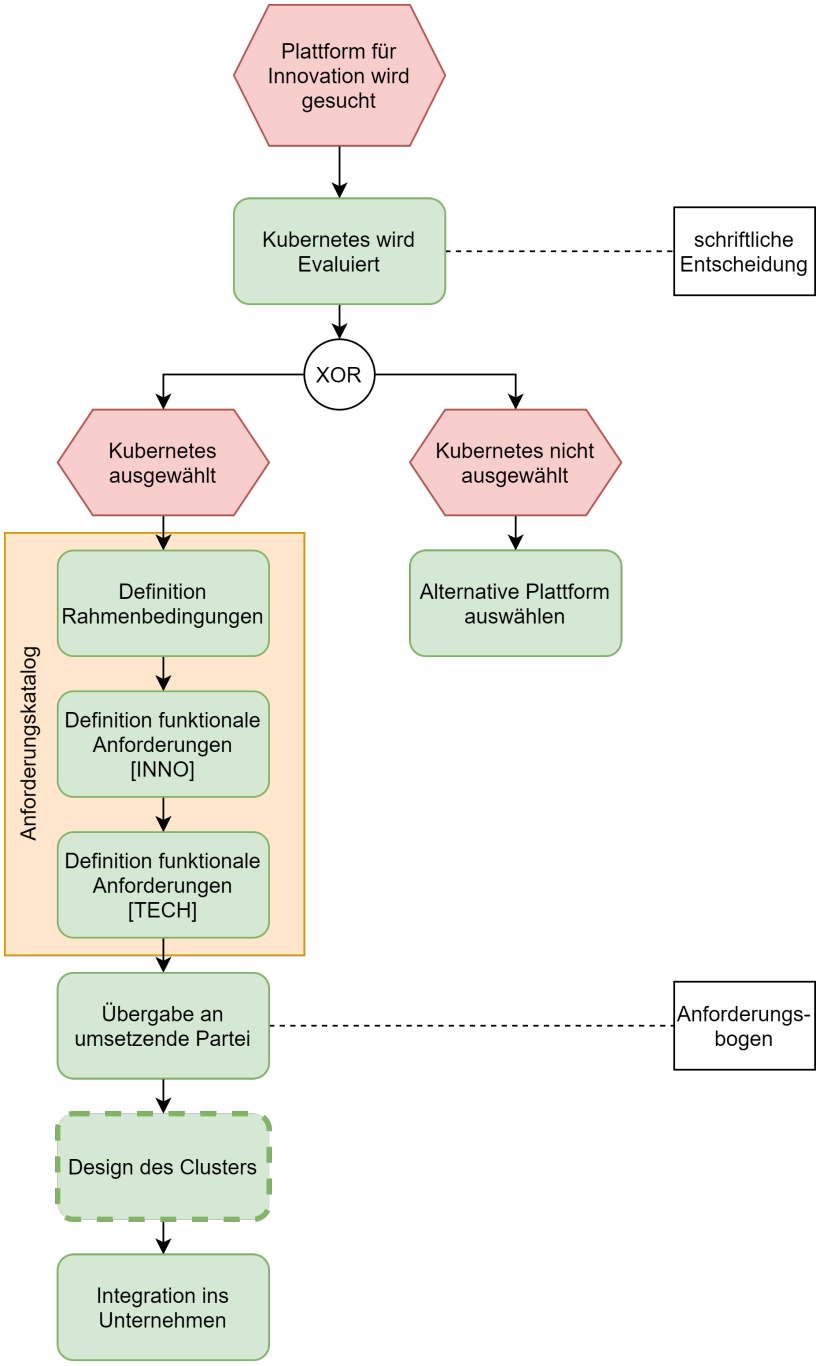


Abb. 4.3:
Designablauf eines Kubernetes Clusters
Quelle: Eigene Darstellung

4.2 Grundkriterien zur Verwendung von Kubernetes

Dieser Abschnitt beschäftigt sich mit der Frage, ob Kubernetes überhaupt die richtige technologische Wahl zur Umsetzung der vorliegenden Innovation ist. Zwar birgt Kubernetes enormes Potenzial und viele Möglichkeiten, kann diese jedoch nicht in jedem Use-Case voll entfalten. Vor allem für umfangreiche Anwendungen, die aus mehreren Komponenten oder Microservices bestehen, oder viel Last verarbeiten müssen, eignet sich Kubernetes sehr gut. Für kleine Applikationen ohne viele Komponenten und mit wenig erwarteter Last eignen sich andere Cloud-Technologien oder klassische Herangehensweisen wahrscheinlich besser. Die zusätzliche Komplexität und der Aufwand, sowie die Kosten des Clusters würden sich im Verhältnis zu den Anforderungen der kleinen Applikation höchstwahrscheinlich nicht rentieren. Vor allem wenn anfangs kein Know-How vorhanden ist, muss bei der Einführung von Kubernetes im Unternehmen, auf Grund der Lernkurve, mit Einbußen der Produktivität gerechnet werden. Sobald das Entwicklungs- und Betriebs-Team mit der Technologie vertraut sind zeigen sich meist gute Produktivitätssteigerungen. Es sei an dieser Stelle auch erwähnt, dass gerade die Migration bereits bestehender Applikationen manchmal problematisch sein kann. Werden Anwendungen mit Kubernetes als Zielplattform entwickelt, so kann auch das volle technologische Potenzial ausgeschöpft werden. Bestehende Applikationen, vor allem wenn diese schon älter sind, stellen ggf. spezielle Anforderungen an das Betriebssystem oder andere Komponenten. Bei solchen Anwendungen muss abgeschätzt werden, ob sich eine Migration auf Kubernetes auszahlt oder nicht.[157]

Die folgenden Punkte sollen dabei helfen zu entscheiden, ob Kubernetes für die angestrebte Innovation die richtige Plattform-Technologie ist oder nicht. Es wird vorsätzlich keine klassische Checkliste angeführt. Die aufgezeigten Punkte sollen vor der Verwendung von Kubernetes diskutiert und als Basis für eine strategische Entscheidung herangezogen werden.[158]

- Die Lernkurve bei der Einführung der Kubernetes Technologie ist anfangs relativ steil. Das bedeutet, dass es einiges an Aufwand und Zeit bedarf, bis Kubernetes für die vorliegende Anwendung so verwendet werden kann, dass die Vorteile bestmöglich ausgenutzt werden können. Es sollte berücksichtigt werden, ob schon Kubernetes Know-How im Team oder im Unternehmen vorhanden ist. Falls nicht muss entschieden werden, ob der Aufwand zur Aneignung des nötigen Know-Hows in Relation zur Größenordnung der angestrebten Innovation steht.[159]
- Je nachdem welche Art von Applikation vorliegt und mit wie viel Last gerechnet wird, stellt sich die Frage ob die Applikation skaliert können werden soll. Das bedeutet, dass beispielsweise erfolgreiche öffentliche Webservices bei häufiger Verwendung horizontal skalierbar sein sollten. In einem solchen Fall bietet sich die Verwendung von Kubernetes an. Kleinere firmeninterne Applikationen benötigen dieses Feature eher nicht. Man sollte sich also fragen, ob die zu entwickelnde Applikation zukünftig eine stark schwankende und große Verwendung haben wird oder nicht.
- Die Architektur der Anwendung spielt auch eine gewisse Rolle bei der Entscheidung ob sie auf Kubernetes betrieben werden sollte oder nicht. Speziell kleinere Applikationen und Mikroservices eignen sich sehr gut für den Betrieb auf Kubernetes. Große Applikationen, welche nicht parallel betrieben werden dürfen, können im Vergleich dazu nicht alle Vorteile von Kubernetes ausschöpfen. Grundsätzlich stellt sich auch die Frage, ob die Applikation schon mit Fokus auf Kubernetes entwickelt wird, oder ob sie ggf. sehr kompliziert migriert werden muss.[160]

[157] Vgl. Thiry (2019), Onlinequelle [06.12.2020].
[158] Vgl. Bakker (2020), Onlinequelle [06.12.2020].
[159] Vgl. Cerruti (2020), Onlinequelle [06.12.2020].
[160] Vgl. Samdan (2020), Onlinequelle [06.12.2020].

- Eines der Hauptargumente für die Verwendung von Kubernetes ist für viele Unternehmen das Vermeiden eines *Vendor Lock-In*. Das bedeutet, dass man sich nicht auf eine herstellerspezifische Technologie fixiert. Dieses Argument kann jedoch auch in die Gegenrichtung erfolgen. Verwendet ein Unternehmen beispielsweise sehr erfolgreich den Technologie-Stack eines bestimmten Herstellers, so muss ein Wechsel auf das Kubernetes Umfeld nicht zwangsläufig eine Verbesserung darstellen. Will sich das Unternehmen also nicht zu stark an einen Hersteller binden, ist Kubernetes eine gute Option. Wird schon erfolgreich ein bestehender Stack verwendet, so muss ein Wechsel gut durchdacht werden.[161]
- Je nachdem welche Distribution von Kubernetes in Betracht gezogen wird, muss auch die Größe des Teams berücksichtigt werden, welches den Cluster betreut. Bei Public-Cloud Distributionen fällt dieser Faktor weniger ins Gewicht, da der Cluster größtenteils vom Provider betreut wird. Sobald eine On-Premises Installation gewählt wurde, ist der Aufwand für Betrieb und Weiterentwicklung des Clusters keinesfalls zu vernachlässigen.[162]
- Da Kubernetes auf Container aufbaut stellt sich natürlich auch die Frage, ob Container generell schon im Unternehmen verwendet werden. Sollte dies der Fall sein, ist die Verwendung von Kubernetes ein naheliegender Schritt. Für den Fall, dass Container noch nicht verwendet werden, muss konkret für die Umsetzung der angestrebten Innovation entschieden werden, ob dieser Schritt getan werden soll. Hier sind wiederum die steile Lernkurve, sowie etwaige anfallende Kosten für den Aufbau von Know-How und die Anschaffung von Software zu berücksichtigen. Grundlegend ist jedoch festzuhalten, dass die Verwendung von Containern stark empfohlen wird.[163]
- Auch wenn die entwickelte Innovation auf Containern basiert, muss sich die Frage gestellt werden ob für den Betrieb eine Orchestrierungsplattform wie Kubernetes wirklich nötig ist. Zwar stellt Kubernetes viele Features zur Verfügung welche für große oder komplexe Softwarelösungen vorteilhaft sind, jedoch benötigt nicht jede Applikation diese auch wirklich. Sollte dies nicht der Fall sein, gibt es auch weitaus weniger komplexe Systeme die der Anwendung gerecht werden würden.[164]
- Ein Faktor der eine essenzielle Rolle spielt ist die Unternehmenskultur. Vor Allem ältere Unternehmen, welche erfolgreich einen etablierten Technologiestack verwenden, sind oft nicht offen gegenüber neuen Technologien wie Containern oder Kubernetes. Es stellt sich deshalb vorab die Frage ob das Unternehmen betreffend seiner Kultur bereit für eine moderne Kubernetes Infrastruktur, mitsamt des gesamten Ökosystems ist. Sollte dies nicht der Fall sein, so muss das Unternehmen zuvor eine grundsätzliche Neuausrichtung durchführen, bevor erfolgreich mit Kubernetes gearbeitet werden kann.[165]
- Sollte das Unternehmen schon eine ausgearbeitete Cloud Computing Strategie verfolgen, so stehen die Chance gut, dass auch mehrere Public-Cloud-Provider beteiligt sind. Die Verwendung eines Public-Cloud Services generell ist schon ein guter Schritt in Richtung Kubernetes. Auch die Verfügbarkeit von mehreren Cloud-Providern wäre für Kubernetes nur vorteilhaft. Die Verteilung eines Clusters auf mehrere Clouds kann Vorteile im Bereich Ausfallsicherheit, Performance und Verfügbarkeit bringen. Auch ist es oft einfacher in Kubernetes einzusteigen, wenn man mit einer Public-Cloud Distribution startet.
- Sollte das Unternehmen bisher sehr lange Entwicklungszyklen haben, oder lange bei der Auslieferung von Software auf den eigenen Systemen brauchen, birgt die Verwendung von Kubernetes einige Vorteile. Die erwähnten DevOps und CI/CD Paradigmen lassen sich sehr gut mit Kubernetes vereinen. Das Erlangen des nötigen Know-Hows erfordert zwar einen gewissen Aufwand, die Verbesserungen der Entwicklungs- und Auslieferungszeiten sind jedoch meist auch signifikant.

[161] Vgl. Turner-Trauring (2020), Onlinequelle [06.12.2020].
[162] Vgl. Bonizi (2020), Onlinequelle [06.12.2020].
[163] Vgl. McCarty (2019), Onlinequelle [06.12.2020].
[164] Vgl. Reznik (2018), Onlinequelle [06.12.2020].
[165] Vgl. ul Haq (2019), Onlinequelle [06.12.2020].

Sollte das Unternehmen mit anderen Technologien schon imstande sein schnell und einfach Software zu entwickeln und bereitzustellen, so muss die Einführung von Kubernetes natürlich diskutiert werden.

Die angeführten Punkte sollen dabei helfen, abschätzen zu können ob Kubernetes eine passende Wahl für die Plattform der vorliegenden Innovation ist. Es muss grundsätzlich für das vorliegende Projekt, sowie auch auf unternehmensweiter Ebene der Technologiestrategie entschieden werden, ob auf Kubernetes gesetzt wird. Kubernetes bietet enorme Vorteile für Systeme die die Funktionalitäten auch ausnutzen können, jedoch kann die Komplexität und der Lernaufwand für einfache Applikationen unverhältnismäßig hoch sein. Grundlegend sei an dieser Stelle festgehalten, dass die Verwendung von Containern enorm zunimmt. Zwischen 2016 und 2019 stieg laut der letzten CNCF Umfrage die Anzahl der Unternehmen, welche Container produktiv nutzen, von 23% auf 84%. Auch die produktive Verwendung von Kubernetes stieg von 2018 auf 2019 sprunghaft von 58% auf 78%.[166]

4.3 Rahmenbedingungen für Kubernetes Cluster

Dieser Abschnitt beschäftigt sich mit der ersten Gruppe von Anforderungen, den Rahmenbedingungen. Diese geben harte Grenzen aus technologischer, rechtlicher, organisatorischer oder ethischer Sichtweise vor, die bei der Bewertung von Produkten unumgänglich sind. Erfüllt ein Produkt eine Rahmenbedingung nicht, so ist diese nicht verwendbar. Beispiele für Rahmenbedingungen in diesem Kontext sind neben den hier aufgeführten, etwa Fragen aus den Bereichen Politik, Gesellschaft, Umwelt, Gesetzgebung, Ethik oder Unternehmensstrategie. Rahmenbedingungen werden meist vom Unternehmen selbst, der Gesellschaft, dem Gesetzgeber oder dem Markt vorgegeben.

In Anhang A.15 findet sich eine Sammlung von Rahmenbedingungen, welche beim Entwerfen eines Kubernetes Systems beachtet werden sollten. Diese Liste ist eine erste Aufstellung wichtiger Fragen, muss jedoch bei jedem Unternehmen nochmals überarbeitet und ggf. erweitert werden. Grundsätzlich befassen sich die Rahmenbedingungen meist mit Fragestellungen, die auf einer strategischen Ebene angesiedelt sind. Speziell strategische Themen wie die Verwendung gewisser Technologien, Partnerschaften mit Drittfirmen oder die geografische Lokation des Systems sind besonders auf der Management-Ebene relevant. Die Vorgaben dieser Ebene müssen klar definiert sein, um später grobe Designprobleme schon frühzeitig vermeiden zu können.

Im Designprozess eines neuen Clusters werden die Rahmenbedingungen herangezogen, um grobe architektonischen Fragen zu beantworten. Dazu gehören zum Beispiel ob Cloud-Dienste verwendet werden dürfen, wo sich der Cluster befinden soll, oder ob bestimmte Drittanbieter per se auszuschließen sind. Bei der späteren Auswahl von Produkten, aus welchen der Cluster zusammengesetzt werden soll, kann anhand der Rahmenbedingungen eine Vorselektion erfolgen. Produkte oder Hersteller welche in irgend einer Art die Rahmenbedingungen nicht erfüllen, müssen bei der weiteren Betrachtung gar nicht erst berücksichtigt werden.

Aus Sicht des Innovationsmanagements bewegt man sich bei den Rahmenbedingungen stark in den Bereichen F&E Management und strategisches Management. Die funktionalen Anforderungen im folgenden Kapitel hingegen beziehen sich konkret auf die technischen Anforderungen der angestrebten Innovation. Dort geht es darum, was ein Cluster können muss, um die geplante Innovation umsetzen zu können.
Da sich die Rahmenbedingungen stark am Unternehmensumfeld sowie auch an der Unternehmensstrategie und -kultur orientieren, wird es nötig sein diese teilweise auf das jeweilige Unternehmen anzupassen. Als Beispiel für eine konkrete Fragestellung aus dem Bereich der Rahmenbedingungen sei das Thema Open-Source und Lizensierung angeführt. Dieses Beispiel ist nochmals in der Sammlung von Rahmenbedingungen in Anhang A.15 angeführt.

[166] Vgl. McMahon (2020), Onlinequelle [06.12.2020].

Lizenz / Open-Source: Software wird immer unter einer bestimmten Lizenz verwendet. Besonders bei der Verwendung von sogenannter Open-Source Software ist darauf zu achten welche Lizenz vorliegt. Einige Open-Source Lizenzen erlauben zwar die Einsicht und Veränderung des Source Codes, jedoch unter Auferlegung gewisser Pflichten. Viele Unternehmen definieren strikt unter welcher Lizenz bezogen werden muss, um diese auch verwenden zu können. Bei proprietärer Software stellt sich diese Frage nicht, da hier nur das fertige Produkt bezogen wird, der Source Code jedoch nicht ersichtlich ist.

Ist die Verwendung von Open-Source Software möglich
(Falls NEIN kann auch proprietäre Software bezogen werden)

○ JA | ○ NEIN

Welche der folgenden Open-Source Lizenzen ist im Unternehmen zulässig?

□ GPL | □ LGPL | □ Apache 2.0 | □ BSD | □ CPL | □ BSD | □ MPL | □ EUPL | □ MIT

4.4 Funktionale Anforderungen an Kubernetes Cluster

Dieser Abschnitt beschäftigt sich mit der zweiten Gruppe von Anforderungen, den funktionalen Anforderungen. Diese beschreiben welche Szenarien der Cluster abdecken soll und daraus folgend, welche technischen Gegebenheiten erforderlich sind. Grundsätzlich können die funktionalen Anforderungen losgelöst von den Rahmenbedingungen definiert werden. Es ist jedoch kein Nachteil die bereits vorliegenden Rahmenbedingungen im Hinterkopf zu haben, um keine utopischen Abweichungen zwischen funktionalen Anforderungen und realistischen Möglichkeiten zu erhalten. Diese Anforderungen werden später herangezogen um schnell, passende Produkte anhand ihrer Eigenschaften zu identifizieren. Manche der folgenden Fragen erfordern ein gewisses technisches Know-How, sowie die Erfahrung einzuschätzen wie die Umsetzung der Innovation in etwa aussehen könnte. Daher ist es ratsam gegebenenfalls technische Unterstützung zu Rate zu ziehen. Fragen die eventuell technische Unterstützung erfordern, bzw. besser genauer besprochen werden sollten, sind mit [TECH] gekennzeichnet.
In Anhang A.16 findet sich eine Sammlung von Fragestellungen, die als Basis für die Formulierung der funktionalen Anforderungen herangezogen werden kann. Es sei an dieser Stelle darauf hingewiesen, dass es nötig sein kann diese Fragensammlung bei der Verwendung in unterschiedlichen Unternehmen zu ergänzen. Im Gegensatz zu den Fragestellungen der Rahmenbedingungen, welche sich noch eher auf Management-Ebene befinden, beschreiben funktionale Anforderungen technische Sachverhalte.

Beim Entwurf eines neuen Clusters werden die funktionalen Anforderungen dazu verwendet zu spezifizieren, welche technischen Eigenschaften die verschiedenen Cluster Komponenten erfüllen müssen. Diese Eigenschaften werden in weiterer Folge für die Auswahl der im Endeffekt verwendeten Produkte herangezogen. Aus Sicht des Innovationsmanagement befinden sich diese Anforderungen stark auf der Ebene der technischen Entwicklung der Innovation. Es muss also bekannt sein was die Innovation tun soll, wie sie funktionieren und implementiert werden soll. Nur wenn diese Punkte gegeben sind, kann auch auf einem entsprechenden technischen Level spezifiziert werden, welche funktionalen Anforderungen der Cluster erfüllen muss. Der Cluster ist im Endeffekt die Plattform auf der die Innovation betrieben und entwickelt wird. Ohne ausreichendes Wissen über die Innovation selbst, kann auch deren zugrunde liegende Plattform nicht korrekt entworfen werden. Die in dieser Sektion angeführten funktionalen Anforderungen definieren auf technischer Ebene was der zukünftige Cluster erfüllen muss.

Um das Verständnis für die Formulierung von funktionalen Anforderungen zu verbessern, sind an dieser zwei Beispiele angeführt. Das erste Beispiel kann vom Innovationsmanagement selbst beantwortet werden, für das zweite Beispiel empfiehlt sich Unterstützung mit technischem Know-How. Beide Beispiele finden sich auch im Fragenkatalog in Anhang A.16.

[INNO][AF03]**Storage Backups**: Werden Daten nur für Tests und Entwicklung benötigt, so wäre deren Verlust wahrscheinlich verkraftbar. In produktiven Umgebungen hingegen ist ein Datenverlust nicht in Kauf zu nehmen. Ein Möglichkeit um Datenverlust vorzubeugen sind Storage Backups.
Werden Storage Backups benötigt?

○ JA | ○ NEIN

[TECH][AF12]**Service-Protokolle**: Je nachdem welche Art von Applikationen auf dem Kubernetes Cluster betrieben werden, bzw. welche Services von außerhalb des Cluster erreichbar sein sollen, werden andere Zugriffsprotokolle verwendet. Grundsätzlich ist es wichtig zu wissen, ob alle Services am Cluster ausschließlich via HTTP(S) erreicht werden können müssen. Aufbauend darauf stellt sich die Frage ob auch HTTP der Version 2, oder gRPC, verwendet wird. Sollte dies nicht der Fall sein, wenn zum Beispiel Datenbanken oder Messaging-Systeme vorhanden sind, so muss noch unterschieden werden, ob diese Anwendungen Protokolle basierend auf UDP oder TCP verwenden.

Über welche Protokolle wird von außen auf Services am Cluster zugegriffen?

□ HTTP(S) | □ HTTP/2 / gRPC | □ TCP | □ UDP

4.5 Qualitätseigenschaften von Kubernetes Clustern

Dieser Abschnitt gibt Auskunft darüber, wie die Qualität den entstehenden Clusters, bzw. seiner Einzelteile, bemessen werden kann. Die Qualitätseigenschaften des Clusters beschreiben in welchem Ausmaß bestimmte Anforderungen erfüllt werden. Im Gegensatz zu den funktionalen Anforderungen geht es nicht mehr darum ob eine Anforderung erfüllt wird, sondern wie gut. Somit sind die Qualitätseigenschaften sowohl beim Design des Clusters, als auch bei dessen fortlaufendem Betrieb zu beachten. Sie helfen dabei zu bewerten welche Produkte im Vergleich zu bevorzugen sind und können auch zur Einschätzung des Gesamtsystems herangezogen werden. Es ist wichtig bei jeder Qualitätseigenschaft zu wissen wie diese definiert ist, wie sie bestimmt werden kann und ob ein großer oder kleiner Wert von Vorteil ist. Grundsätzlich werden Qualitätseigenschaften meist durch kontinuierliche Werte mit einer bestimmten Einheit angegeben. Im Gegensatz dazu werden funktionale Anforderungen meist durch diskrete Werte beschrieben.

Im Designprozess des Clusters werden die Qualitätseigenschaften herangezogen, wenn es darum geht die konkreten Produkte der einzelnen Komponenten auszuwählen. Wenn anhand von Rahmenbedingungen und funktionalen Anforderungen geeignete Produkte identifiziert wurden muss entschieden werden, welche Produkt final auszuwählen ist. Diese Auswahl wird durch das Vergleichen der Produkte, bezogen auf die angeführten Qualitätseigenschaften, erreicht. Auch nach dem Entstehungsprozess des Clusters können diese Eigenschaften verwendet werden, um laufend die Qualität des Clusters zu beurteilen. Aus diesem Grund eignen sich Qualitätseigenschaften sehr gut, um direkt oder indirekt daraus Key Performance Indicators (KPIs) ableiten zu können. Sie eignen sich außerdem zur Formulierung von Rahmenbedingungen. Zum Beispiel könnte eine bestimmte Mindestverfügbarkeit eine gegebene Rahmenbedingung darstellen.

Als Beispiel zur Verdeutlichung der Qualitätseigenschaften sei an dieser Stelle die Verfügbarkeit des Clusters, bzw. einer Komponente angeführt.

Verfügbarkeit: Unter der Verfügbarkeit wird in diesem Kontext die garantierte Uptime des Clusters an sich verstanden. Damit sind nicht die einzelnen Worker Nodes gemeint, sondern die durchgehende Erreichbarkeit der Control Plane, sprich der Kubernetes API. Diese wird meist durch ein Service Level Agreement abgesichert und als Prozentsatz in Verfügbarkeit pro Zeiteinheit angegeben, meist pro Monat. Kubernetes Distributionen die von Public-Cloud-Providern, sowie von Managed-Service-Providern angeboten werden, garantieren fixe Werte für die Verfügbarkeit. Bei On-Premises Installationen hingegen ist das Unternehmen selbst dafür verantwortlich die Verfügbarkeit zu gewährleisten. Je höher die prozentuale Verfügbarkeit pro Zeitintervall ist, desto besser.[167]

In Anhang A.17 findet sich eine Sammlung von Qualitätseigenschaften die zur Bewertung der Qualität eines Kubernetes Systems, bzw. dessen Komponenten, herangezogen werden kann.

Bisher wurde in diesem Kapitel erklärt wie der gesamte Designprozess eines neuen Kubernetes Clusters ablaufen könnte. Es wurde aufgezeigt welche Arten von Anforderungen an das System gestellt werden können und wie sich diese unterscheiden. Außerdem wurde argumentiert, dass die tatsächliche Auswahl von Produkten nicht mehr zum Kernthema dieser Arbeit gehört. Eine Idee dafür wie dieser Schritt aussehen könnte, findet sich in Anhang A.18. Dies ist jedoch mehr als Vorschlag, als als Forschungsergebnis zu sehen. Auch die Integration des Clusters in die bestehende Aufbau- und Ablauforganisation des Unternehmens wird an dieser Stelle nicht weiter vertieft. Dieser Punkt muss im Praxisfall individuell für jedes Unternehmen betrachtet werden. Das nächste große Kapitel beschreibt wie das bisher theoretisch erarbeitete Modell validiert und ergänzt wurde.

[167] Vgl. Berger (2020), Onlinequelle [06.12.2020].

5 VALIDIERUNG UND ERGÄNZUNG DES MODELLS

Dieses Kapitel und seine Abschnitte zeigen auf, wie das vorliegende Modell aus Designprozess und Anforderungskatalog auf seine praktische Anwendbarkeit geprüft und erweitert wurde. Bevor das Modell vollständig angewandt und dementsprechend validiert werden konnte, wurde versucht weiteres Feedback einzuholen und das Modell soweit als möglich abzurunden. Dafür wurden im ersten Schritt Workshops durchgeführt, bei denen Teilnehmer aus mehreren Bereichen eines Unternehmens vertreten waren. Im zweiten Schritt wurden einzelne Experteninterviews, mit Experten aus verschiedenen Branchen, durchgeführt. Die zugrundeliegende Idee war es, mit Hilfe der Workshops so viele Aspekte innerhalb eines Unternehmens zu beleuchten wie möglich. In den Interviews sollte anschließend noch versucht werden, Informationen und Anforderungen aus spezifischen Branchen einzuholen. Mit diesem Zugang wurde versucht das vorliegende Modell so gesamtheitlich wie möglich zu entwickeln, und eine größtmögliche Anzahl an verschiedenen Blickwinkeln zu berücksichtigen. Bevor auf die praktische Anwendung der Methoden eingegangen wird, soll der folgende Abschnitt noch die dafür zugrunde liegende Theorie erklären.

5.1 Wissenschaftliche Methodik

Um schlüssig erklären zu können, warum die angewandte Methodik ausgewählt wurde, sollen an dieser Stelle nochmals die Forschungsfragen angeführt werden.

Basierend auf einer Innovation oder vorliegenden Idee, wie könnte eine strukturierte Vorgehensweise für das Design eines Kubernetes Clusters aussehen?

Welche Aspekte muss ein Fragebogen zur innovationsgetriebenen Spezifikation von Anforderungen an einen Kubernetes Cluster berücksichtigen?

Es geht also darum einerseits eine Vorgehensweise bzw. einen Prozess vorzuschlagen, welcher beim Design neuer Kubernetes Cluster hilft. Andererseits geht es um die Erhebung eines Anforderungsbogen, welcher im vorgeschlagenen Prozess eingesetzt werden kann. Im Theorieteil der Arbeit wurden zu beiden Punkten erste Ausarbeitungen entworfen, welche ein ganzheitliches Designmodell für Kubernetes Cluster vorschlagen. Dieses Modell wird im praktischen Teil validiert, ergänzt und angewandt. Bei der Auswahl der wissenschaftlichen Methodik für Validierung und Ergänzung war grundlegend zwischen quantitativen und qualitativen Methoden zu unterscheiden. Quantitative Methoden zielen meist darauf ab bestimmte Beobachtungen anhand von Modellen oder Zusammenhängen zahlenmäßig zu beschreiben. Es geht darum ein bestmögliches Modell zu finden und zukünftige Ereignisse vorhersagbar zu machen. Dafür werden gewisse Ausprägungen oder Charakteristiken gemessen und quantifiziert sowie mit anderen Variablen kombiniert. Man erhofft sich am Ende ein Modell zu finden, welches von einer reduzierten Datenmenge ausgehend auf die Grundgesamtheit angewendet werden kann und dort immer noch zutrifft. Häufig wird hier auf statistische und mathematische Methoden zurück gegriffen sowie auf standardisierte und strukturierte Daten aufgebaut. Beispiele für quantitative Methoden wären etwa Single Choice Fragebogen oder die statistische Auswertung diverser Metriken.
Im Gegensatz dazu herrscht bei qualitativen Methoden eine wesentlich größere Flexibilität, es wird weitestgehend auf Standardisierung verzichtet. Qualitative Methoden sind sehr subjektiv orientiert, sie erzeugen bzw. analysieren offene Daten. So liegt beispielsweise einem Interview ein gewisser Leitfaden zugrunde, die Antworten der Interviewpartner sind jedoch völlig frei. Diese Methoden haben meist einen größeren Informationsgehalt, sind jedoch selten repräsentativ für eine Grundgesamtheit. Speziell für offene und explorative Fragestellungen eignen sich qualitative Methoden sehr gut. Quantitative Methoden hingegen haben ihre Stärken beim Identifizieren von Mustern und Zusammenhängen in bereits bekannten Modellen.

Stark vereinfacht ausgedrückt eignen sich qualitative Methoden gut für das Aufstellen von Modellen und Hypothesen (induktives Vorgehen), quantitative Methoden unter anderem für deren Überprüfung (deduktives Vorgehen).[168]

Da die Forschungsfragen eindeutig die Erarbeitung eines neuen Modells fordern, wurde die Methodenauswahl grundsätzlich in Richtung qualitativer Methoden getroffen. Um die Qualität quantitativer Forschung zu beurteilen gibt es derzeit noch kein standardisiertes Modell. Es wurden einige Ansätze in der Literatur vorgeschlagen, welche mehr oder weniger unterschiedliche Aspekte berücksichtigen. Punkte wie die Kohärenz, Relevanz, Regelgeleitetheit, Dokumentation, Validität und Triangulation werden von verschiedenen Autoren vorgeschlagen. Als praktisch verwendbarer Konsens der verschiedenen Ansätze hat sich die Betrachtung der folgenden drei Eigenschaften ergeben. Die Punkte Transparenz, Intersubjektivität, sowie eine gewisse Reichweite, müssen sichergestellt werden. Der Punkt Transparenz besagt, dass darauf Wert gelegt werden muss, dass der gesamte Forschungsprozess dokumentiert und nachvollziehbar ist. Auch alle Ergebnisse und Entscheidungen des Forschungsprozesses müssen vollständig einsehbar und Teil der Arbeit sein. Dieser Punkt wird durch das Anfügen aller generierten Daten im Anhang, sowie die Erklärung der Vorgehensweise in diesem Kapitel sichergestellt. Die Intersubjektivität bezeichnet das Konfrontieren der vorliegenden Ergebnisse durch mehrere Blickwinkel. Damit soll eine potenzielle Subjektivität weitestgehend ausgeschlossen werden. Dies betrifft sowohl die Erhebung, als auch die Auswertung von Daten. Diesem Punkt wird speziell durch die Anwendung verschiedener Datenerhebungsmethoden Rechnung getragen. Dadurch wird versucht eine einseitige Betrachtung zu verhindern. Bei der Auswertung der Daten wird auf eine strukturierte Methode gesetzt, welche Subjektivität per Definition bestmöglich ausschließen soll. Das letzte Gütekriterium der Reichweite beschreibt den Umfang des Themenbereichs für welchen die gewonnen Ergebnisse tatsächlich angewandt werden können. Durch die meist kleineren Stichproben, im Vergleich zu quantitativen Methoden, muss genau abgesteckt werden was der tatsächliche Geltungsbereich ist. Dieser Punkt wird berücksichtigt, indem die benötigten Daten möglichst umfassend gesammelt werden, die gewonnenen Erkenntnisse jedoch nur auf ein spezifisches Feld abzielen. Obwohl Informationen eingesammelt werden die für viele IT Systeme Geltung haben, bezieht sich das resultierende Modell lediglich auf Kubernetes Systeme.[169]

Die Berücksichtigung der letzten beiden Gütekriterien setzt voraus, dass eine ausreichende Menge an Daten bzw. Informationen erhoben wird, um eine seriöse Aussage treffen zu können. Um dies sicherstellen zu können wurde bei der Auswahl der Datenerhebungsmethoden auf einen sogenannten "Multi-Method" Ansatz zurückgegriffen. In diesem Kontext bedeutet dies, dass speziell zur Erhebung der Daten auf mehrere qualitative Methoden zurückgegriffen wurde. Konkret wurden Gruppeninterviews bzw. Workshops sowie Experteninterviews durchgeführt. In der Theorie findet sich die Unterscheidung in "Multi-Method" sowie "Mixed-Method" Ansatz, wenn mehrere Methoden herangezogen werden. Der Begriff "Multi-Method" bezeichnet hierbei das Verwenden mehrerer Methoden jeweils qualitativer oder quantitativer Natur. Hingegen wird bei 'Mixed-Method" Arbeiten eine Kombination aus qualitativen und quantitativen Methoden angewandt.[170]

In Abbildung 5.1 ist der Ablauf des praktischen Teils, sprich das Forschungsdesign, grafisch dargestellt. Die einzelnen Abschnitte werden im Folgenden ausführlich erklärt.

Gruppeninterviews/Workshops: Die Wahl zu Gunsten von Gruppeninterviews, im Folgenden Workshops genannt, als eine der beiden Datenerhebungsmethoden, wurde auf Grund ihrer vorteilhaften Charakteristiken getroffen. Im Gegensatz zu Gruppendiskussionen, welche die Eigenart haben, dass alle Teilnehmer miteinander diskutieren, wird ein Gruppeninterview stärker durch einen Moderator gelenkt.

[168] Vgl. Roebken (2019), S. 12ff; Winter (2000), Onlinequelle [06.12.2020]
[169] Mey (2020), Onlinequelle [06.12.2020]; Motschnig (2012), Onlinequelle [06.12.2020]
[170] Vgl. Collier (2008), S. 781f; Vgl. Schoonenboom (2017), S. 108f

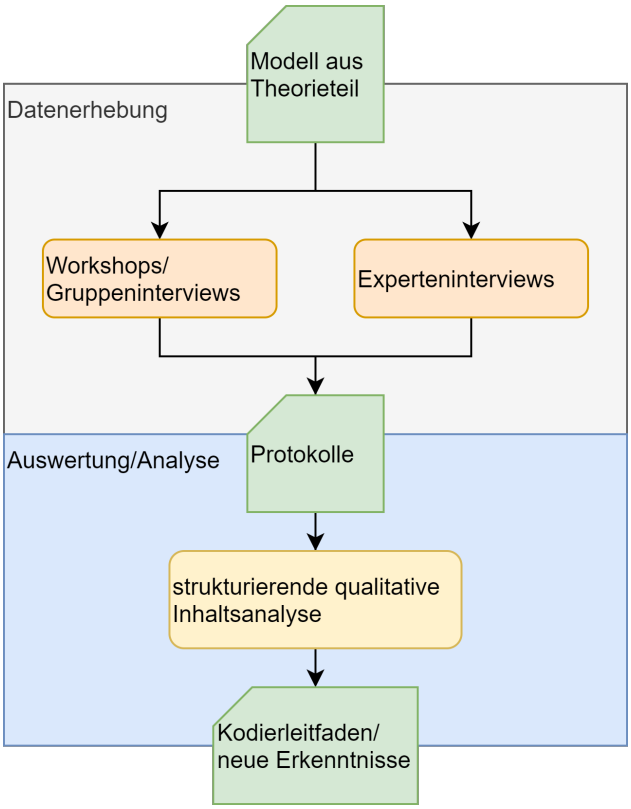


Abb. 5.1:
Forschungsdesign und Methoden
Quelle: eigene Darstellung

Auch ist die Interaktion zwischen den Teilnehmern weniger ausgeprägt. Wie der Name vermuten lässt, geht man eher wie bei einem Interview vor, jedoch werden alle Interviewpartner in einem Durchlauf befragt. Ob nun jeder Partner einzeln, vollständig und nacheinander (sequentiell), oder jedes Thema mit jedem Teilnehmer (parallel) behandelt wird ist eine Frage der Planung. Wichtig ist jedoch, dass immer alle Gruppenmitglieder anwesend sind, sich wahrnehmen können und auch die Möglichkeit haben zu diskutieren. Dadurch kann die kreative Anregung durch andere Gruppenmitglieder mit der strukturierten Vorgehensweise eines Interviews verbunden werden. Eine klassische Gruppendiskussion würde sich dagegen eher für das offene Diskutieren eines einzigen Themas sowie für Evaluierungen eignen. Es besteht die Gefahr, dass die Gruppe als Ganzes in eine bestimmte Richtung abdriftet.[171] Ein weiterer Gedanke bei der Planung der Workshops war es, die Befragung der Teilnehmer angelehnt an die Delphi Methode zu gestalten. Dies sollte den Teilnehmern dabei helfen andere Blickwinkel zu berücksichtigen und dadurch noch weitere Ergebnisse zu identifizieren. Die Delphi Methode besteht grob aus drei Runden. In der ersten Runde werden Experten geben, losgelöst voneinander und ohne vorherigen Austausch, ihre Meinung zu einem bestimmten Thema zu formulieren. Diese Ausarbeitungen werden in der zweiten Runde an alle Teilnehmer ausgegeben. Die letzte Runde bietet die Möglichkeit seine ursprüngliche Ausarbeitung abzuändern. Dieser Vorgang kann beliebig wiederholt werden. Mit dieser Herangehensweise soll, eventuell über mehrere Runden hinweg, ein Konsens unter den Experten zu einem Thema gefunden werden. In diesem Fall sollten die Teilnehmer schon vor dem Workshop erste Ideen und Vorschläge für Anforderungen einbringen, welche anschließend im Workshop präsentiert werden. In einer zweiten Runde während des Workshops sollten die Teilnehmer, inspiriert durch den Input ihrer Kollegen, versuchen weitere Anforderungen zu identifizieren. Dieser Ansatz musste jedoch auf Grund mangelnder Rückmeldung im ersten Schritt abgeändert werden.

[171] Vgl. Mäder (2017), S. 25.

Stattdessen wurden die beiden Befragungsrunden im Workshop selbst durchgeführt und die Teilnehmer wurden gebeten, sich zumindest bis zum Workshop Gedanken über potenzielle Anforderungen zu machen. Grundsätzlich waren die Fragerunden primär auf die Identifikation neuer Anforderungen für den Anforderungskatalog ausgelegt. Es wurden jedoch auch einige Inputs zum Designprozess selbst eingebracht.[172]

Experteninterviews: Die zweite Methode, welche gemäß des "Multi-Method" Ansatzes ausgewählt wurde, sind sogenannte Experteninterviews. Diese gehören der Gruppe der semi-strukturierten bzw. leitfadengestützten Interviews an, einer Gruppe von qualitativen Interviewmethoden. Im Gegensatz dazu gäbe es auch Interviews ohne Leitfaden, wie zum Beispiel das narrative oder das ethnografische Interview. Beim narrativen Interview wird der Partner aufgefordert monologisch ein Ereignis zu erzählen. Häufig handelt es sich dabei um Geschehnisse oder fantastische Erzählungen. Das ethnografische Interview dagegen ist ein spontanes und offenes Gespräch ohne vorgegebene Struktur. Leitfadengestützte Experteninterviews geben dem Interview einen groben Rahmen vor, lassen den Experten jedoch frei antworten. Diese Charakteristik erfordert, dass der Interviewpartner fundiertes Wissen im behandelten Bereich aufweist, daher auch der Name. Mit dieser Methode wird versucht, durch so spezifische Fragen wie nötig, so umfangreiche und informationsreiche Antworten wie möglich zu erhalten. Da aus der Theorie schon eine gewisse Grundstruktur für einen Interviewleitfaden vorhanden ist und das Thema spezielles Wissen erfordert, fiel die Wahl auf diese wissenschaftliche Methode. Außerdem war es wichtig ein Verfahren zu wählen, das es erlaubt ein Thema oder eine Richtung vorzugeben, dabei jedoch nichts von der Antwort vorweg zu nehmen. Der Interviewpartner durfte also durch die Fragestellung nicht beeinflusst werden, oder sollte dies zumindest so wenig als möglich. Durch die Möglichkeit die Interviewfragen zwar pro Themenbereich, aber trotzdem sehr offen zu formulieren, wurde dies sichergestellt. Der einfache Leitfaden, welcher für die Interviews verwendet wurde ist in Anhang A.21 ersichtlich.

Die Auswahl der Experten ist bei dieser Methode ausschlaggebend, da meist versucht wird durch wenige Interviews viele Informationen zu gewinnen. In diesem Fall wurde auf technische Experten aus dem Kubernetes und IT Plattformen Bereich zurückgegriffen. Die ausgewählten Personen haben durchwegs einen technischen Hintergrund und arbeiten selbst mit Kubernetes. Es ist nötig die durchgeführten Interviews vollständig zu transkribieren um die gesamte Information zu erhalten, und diese anschließend in der Analyse extrahieren zu können. In diesem Fall wurden die Interviews alle online via Videokonferenz durchgeführt und aufgezeichnet. Anschließend konnten die Mitschnitte transkribiert werden. [173]

Analyse: Für die Auswahl der Analysemethode gilt was schon am Anfang des Kapitels erklärt wurde. Da die Forschungsfrage auf die Erstellung eines neuen Modells abzielt, welches wiederum auf der Identifikation von Anforderungen beruht, wurde eine qualitative Methode gewählt. Es geht darum in den gesammelten Daten aus Workshops und Experteninterviews Informationen zu extrahieren, welche dabei helfen das Modell zu validieren und zu verbessern. Das zentrale Element der Analyse sind wiederum die Forschungsfragen. Dafür ist es wichtig neue inhaltliche Erkenntnisse, wie zusätzliche Prozessschritte oder neue Anforderungen, zu gewinnen. Quantitative Methoden wären dafür nicht geeignet. In Abbildung 5.2 ist der generelle Ablauf einer qualitativen Inhaltsanalyse ersichtlich. Diese Methode wurde als Analysewerkzeug herangezogen da sie sich für die Identifikation bestimmter Themen im Material sehr gut eignet. Im Gegensatz dazu wäre zum Beispiel die Grounded Theory Methode gänzlich ungeeignet. Diese hätte das Ziel ein neues Modell, auf Basis des vorliegenden Materials, zu erstellen, was nicht der vorliegenden Problemstellung entspricht.[174]

[172] Vgl. Adam (2012), S. 267f.
[173] Vgl. Helfferich (2019), S. 669ff; Vgl. Liebold (2009), S. 32f
[174] Vgl. Chun Tie (2019), S. 1f.

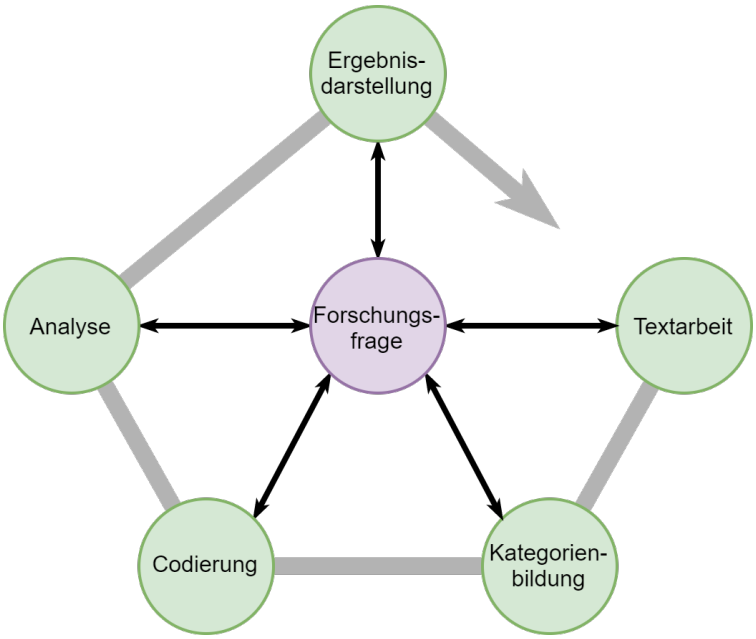


Abb. 5.2: **Ablauf einer qualitativen Inhaltsanalyse**
Quelle: in Anlehnung an Kuckartz (2016), S. 45

Im speziellen wurde auf die Methode der inhaltlich strukturierenden qualitativen Inhaltsanalyse zurückgegriffen. Die Schritte des Ablaufs dieser Methode sind in Abbildung 5.3 zu sehen. Dabei wird das gesamte Material analysiert und gemäß einer Sammlung sogenannter Codes kategorisiert.

Das bedeutet, dass einzelne Sätze oder gesamte Paragraphen jeweils einem oder mehreren Codes zugeordnet werden. Dadurch lässt sich in der nachfolgenden Auswertung gut aufzeigen worüber welcher Interviewpartner gesprochen hat und welche etwaigen Zusammenhänge es gibt. Eine Zusammenfassung oder Paraphrasierung wie sie bei der zusammenfassenden Inhaltsanalyse nach Mayring gemacht wird, wurde in diesem Fall ausgelassen. Die Informationsdichte in diversen Paragraphen war so hoch, dass eine Zusammenfassung nicht sinnvoll gewesen wäre. Weitere Sonderformen oder Unterkategorien der Inhaltsanalyse, wie etwa die explikative Inhaltsanalyse, waren ebenfalls nicht erforderlich. Es war beispielsweise nicht weiter nötig, etwaige unklare Passagen durch erweiterte Kontextinformationen zu verdeutlichen, wie dies in der explikativen Inhaltsanalyse der Fall ist.

Die inhaltlich strukturierende Inhaltsanalyse ist im engeren Sinne streng genommen eine deduktive Methode. Das bedeutet, dass die Codes an das Material herangetragen und angewendet werden. Daher müssen diese schon aus der Theorie bekannt sein. Da es jedoch darum geht Informationen aus dem Material zu gewinnen, hat sich ein Ansatz etabliert in dem das Material zur Gewinnung der Codes verwendet wird. Dies nennt man einen induktiven Ansatz. Dafür werden die Daten vorab gesichtet und anhand des Materials werden neue Codes definiert, welche abschließend auf das ganze Material angewendet werden. Man hat also einen teils induktiven, teils deduktiven Ansatz.[175]

[175] Vgl. Mayring (2014), S. 543ff.

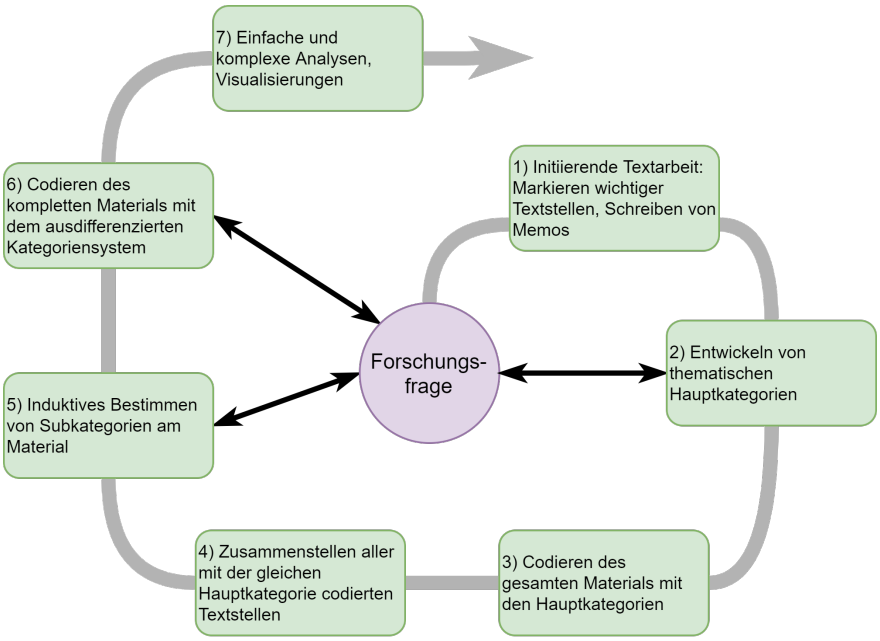


Abb. 5.3: **Ablauf einer strukturierenden qualitativen Inhaltsanalyse**
Quelle: in Anlehnung an Kuckartz (2016), S. 100

In diesem Fall wurde das Codesystem hierfür eigens entwickelt. Ausgangspunkt dafür waren die vier groben Kategorien "Rahmenbedingungen", "funktionale Anforderungen", "Qualitätseigenschaften" und "Designprozess". Mit diesen, durch den Literaturteil vorgegebenen Codes, wurde eine erste Kategorisierung des Materials vorgenommen. Dies entspricht den Schritten 2 - 4 in Abbildung 5.3. Anschließend wurden die kategorisierten Textstellen weiter betrachtet und gemäß ihrer Aussage in neu erstellte Subkategorien, bzw. Sub-Codes, unterteilt. Das bedeutet, dass anhand des vorliegenden Materials neue Codes erstellt wurden, die unter den vier Hauptcodes angesiedelt sind. Abschließend wurden die gewonnenen Codes mit den bestehenden Anforderungen des Anforderungskatalogs abgeglichen und, wo dies möglich war, vereint. Die finalen Codes bilden den sogenannten Kodierleitfaden welcher in Anhang A.28 ersichtlich ist. Mit diesem finalen Kodierleitfaden wurde nochmal das gesamte Material codiert bzw. ergänzt. Dies entspricht den Schritten 5 und 6 in Modell nach Abbildung 5.3.

Codes können grundsätzlich nominaler oder ordinaler Natur sein, oder anhand eines Intervalls definiert sein. Nominal bedeutet, dass es eine Gruppe von Ausprägungen gibt, die aber keine Ordnung haben. Beispiele hierfür wären Farben, die Geschlechter oder politische Parteien. Ordinale Codes hingegen haben zumindest eine Rangordnung bei der Möglichkeit ihrer Ausprägung. Darunter würde zum Beispiel eine Zufriedenheitsskala fallen. Intervalle wären konkrete Zahlenwerte, wie beispielsweise Temperaturen oder Geldbeträge.
Die Codes dieser Arbeit identifizieren nahezu immer nur das Vorhandensein einer Anforderung, waren also per se nominaler Natur. Als Definition für jeden Code wurde jeweils eine geschlossene Frage gewählt. Die codierten Textstellen stellen die Antworten auf diese Frage dar. Ob die Antwort, gemäß der codierte Textstelle, als ja oder nein interpretiert werden kann ist in diesem Fall irrelevant. Es geht lediglich darum, dass der Code per se verwendet wird. So wurde beispielsweise nicht unterschieden welche geografischen Standorte für den Interviewpartner interessant sind, sondern lediglich ob das Thema Standort an sich genannt wird. Dies wird dadurch gerechtfertigt, dass es gemäß der Forschungsfrage nicht relevant ist wie häufig gewisse Ausprägungen eines Aspekts vorkommen. Es ist jedoch ausschlaggebend welche Aspekte bzw. Themen genannt werden.
In diesem Sinne kann der Schritt 7, diverse einfache und komplexe Analysen, im Prozess gemäß Abbildung 5.3 vernachlässigt werden.

Dieser Schritt wäre unter anderem dafür gedacht Zusammenhänge zwischen codierten Segmenten zu erkennen und zu visualisieren. Da dies jedoch für die vorliegende Forschungsfrage irrelevant ist, wird darauf verzichtet. Der Fokus liegt auf der Identifikation und Abgrenzung von Codes, da diese im Endeffekt die Anforderungen bzw. Prozessschritte wiedergeben die es zu erheben gilt.[176]

Das Ergebnis der Analyse ist also eine Liste an Anforderungskategorien, sprich Codes, die sich teilweise mit den in der Theorie identifizierten Anforderungen decken. Es wurden im Zuge der Interviews und der daraus folgenden Analyse jedoch auch zahlreiche neue Anforderungen identifiziert.

5.2 Interdisziplinäre Workshops

Der erste Schritt des praktischen Teils dieser Arbeit beschäftigte sich mit der Planung und Durchführung von Workshops im Unternehmen. Dabei war der Fokus darauf gerichtet so viele Stakeholder wie möglich zu befragen, die auch nur entfernt mit Kubernetes in Berührung kommen könnten. Es ist an dieser Stelle wichtig zu erwähnen, dass es sich dabei nicht nur um Stakeholder mit technischer Ausbildung und Position drehte. Auch verschiedene Management Ebenen, sowie Innovationsmanagement und Marketing waren vertreten. Selbstverständlich waren viele der Teilnehmer auch in technischen Abteilungen angesiedelt. Die folgenden beiden Unterkapitel erklären wie die Workshops geplant und durchgeführt wurden, sowie deren Ergebnisse.

5.2.1 Planung und Durchführung der Workshops

Bei der Auswahl der Teilnehmer wurde versucht darauf zu achten, dass alle Stakeholder eines Kubernetes Systems weitestgehend vertreten sind. Es wurden präferiert Teilnehmer eingeladen, die schon eine längere Zeit im Unternehmen tätig sind und einen Senior Status haben. Mitarbeiter die noch nicht lange im Unternehmen oder Betätigungsfeld arbeiten, können wahrscheinlich weniger hilfreichen Input geben als erfahrene Mitarbeiter. Insgesamt wurden zwei Workshops abgehalten in denen 14 Fachbereiche vertreten waren.

Auf Grund der COVID-19 Pandemie und entsprechender Social Distancing Maßnahmen, mussten die Workshops online mittels Videokonferenz abgehalten werden. Dies hatte zur Folge, dass die Teilnehmer möglicherweise weniger aufmerksam oder engagiert am Workshop teilnahmen, als sie es in einem Workshop mit persönlicher Anwesenheit getan hätten. Ein enormer Vorteil der online Durchführung war jedoch das verhältnismäßig einfache Planen des Workshops. Da sich nahezu alle Teilnehmer in Homeoffice befanden musste kein Besprechungsraum gefunden werden der allen Teilnehmern zugänglich war. Die Teilnehmer sind über mehrere Standorte verteilt stationiert, was zu Problemen bzgl. der Anfahrt führen hätte können. Auch eine eventuelle Durchführung in hybridem Modus, sprich teilweise am Standort und teilweise via Videokonferenz, konnte somit umgangen werden. Das größte Hindernis war die Findung eines passenden Termins, da die meisten Teilnehmer in Management Positionen einen sehr vollen Terminkalender haben. Auch die Länge der Workshops, welche mit drei Stunden angesetzt wurden, erschwerte die Terminfindung.

Den Teilnehmern wurde in der Einladung vorab ein One-Pager zukommen gelassen, welcher die Problemstellung sowie die erklärten Gruppen von Anforderungen aufzeigte. Die Gruppen wurden jeweils durch Beispiele kurz erklärt. Somit sollten die Teilnehmer einen Einblick in das Thema bekommen und sich schon im Vorhinein Gedanken über Anforderungen aus ihrem Fachbereich machen können. Der One-Pager ist unter Anhang A.19 ersichtlich. Die folgende Aufzählung gibt eine Übersicht darüber welche Fachbereiche in den Workshops vertreten waren.

[176] Vgl. Schreier (2014), S. 1ff.

- DevOps / IT-Infrastructure Specialist
- Enterprise Architekt
- Developer
- Lead Developer
- Field-Engineer
- Innovationsmanager
- IT-Security Engineer
- Management Development
- Management Operations
- Testmanagement
- Marketing/Supportorganisation
- Product Manager
- Project Manager
- Process Manager

Um die Inhalte der vorliegenden Arbeit besser vermitteln und einen Bezug zum Unternehmen herstellen zu können, wurde folgender Aufbau für die Workshops gewählt. Es wurde im ersten Teil erklärt was Container und Kubernetes sind, wie sie grundlegend funktionieren und wofür man die Technologien verwenden kann. Anschließend wurde erklärt wie Kubernetes im Unternehmen bereits eingesetzt wird. Dafür wurden einige Kennzahlen vorgestellt, eine Demonstration eines einfachen Use-Cases vorgeführt, sowie einige Anwendungsfälle aufgezeigt für die Kubernetes bereits als Plattform verwendet wird. Nach dem firmenspezifischen Teil wurde die erste Befragungsrunde der Teilnehmer gestartet. Dabei wurde jeder Teilnehmer nacheinander aufgefordert Aspekte einzubringen die aus seinem Fachbereich zu berücksichtigen wären. In dieser Runde wurde noch weitestgehend auf Rückfragen durch den Workshopleiter verzichtet. Jedoch entstanden in beiden Workshops bereits hier einige Diskussionen. Anschließend an die erste Fragerunde wurden die bisherigen Ergebnisse der vorliegenden Arbeit kurz vorgestellt, sowie bereits eingebrachte Aussagen erneut hervorgehoben. Im letzten Abschnitt wurden die Teilnehmer erneut befragt, diesmal jedoch mit gezielten Fragestellungen zu bestimmten Bereichen ihres Fachgebietes. In beiden Workshops wurde in dieser Runde am meisten diskutiert. So wurde beispielsweise das Thema Lizenzen und Open-Source von mehreren Teilnehmern angesprochen und hervorgehoben, obwohl der eigentliche Lizenzverantwortliche, auf Grund terminlicher Überschneidungen, nicht am Workshop teilnahm.

Die Workshops dauerten zwei bzw. zweieinhalb Stunden, einer davon wurde vollständig aufgezeichnet. Diese Aufzeichnung wurde vom Unternehmen gefordert, um die Inhalte auch anderen Kollegen zugänglich machen zu können. Welche Teilnehmer an welchem Workshop teilgenommen haben ist in Tabelle 5.1 ersichtlich. Die Ergebnisse der Workshops wurden jeweils stichwortartig notiert und anschließend sauber aufbereitet. Es wurden alle gewonnen Erkenntnisse kategorisiert und wo nötig kurz erklärt. Die Präsentation, welche als Leitfaden der Workshops diente, ist in Anhang A.29 ersichtlich. Die Protokolle sind in Anhang A.30 und A.31 einsehbar.

Workshop 1	Workshop 2
IT-Security Engineer	Management Operations
Process Manager	Project Management
Management Development	Management Development
Innovationsmanager	Testmanagement
Product Manager	DevOps / IT-Infrastructure Specialist
Enterprise Architekt	
Field-Engineer	
Developer	
Marketing/Supportorganisation	

Tab. 5.1: Teilnehmerlisten der Workshops

5.2.2 Zusammenfassung der Ergebnisse der Workshops

Um die Ergebnisse der Workshops zu analysieren wurden die Protokolle, analog zu den Interviewprotokollen, mit einer qualitativen Inhaltsanalyse ausgewertet. Die Ergebnisse dieser Auswertung werden im Folgenden, zusammengefasst in die bekannten vier Gruppen, aufgezeigt. Dabei ist zu berücksichtigen, dass der Hauptfokus das Identifizieren neuer Anforderungen war. Dementsprechend wird vor allem beschrieben welche neuen Erkenntnisse aus den Workshops gewonnen werden konnten, und wo deren Ergebnisse sich schon mit der Theorie deckten.

Bei der Analyse des Materials wurde eine zusätzliche Oberkategorie eingeführt welche alle Punkte abdeckt, die nicht Teil der Betrachtung dieser Arbeit sind. Diese Kategorie wurde "Out-Of-Scope" benannt. Darin fielen im ersten Workshop beispielsweise Punkte wie das Scannen von Containern auf Viren und Schadsoftware. Auch die Überprüfung von Lizenzverletzungen der Software in den Containern wurde eingebracht. Diese Punkte können zwar von Drittherstellersoftware durchgeführt werden, haben jedoch mit dem eigentlichen Cluster nichts mehr zu tun. Des Weiteren wurden in dieser Kategorie noch Punkte bzgl. der Paketierung und Standardisierung, sowie der Auslieferung gesamter Softwarelösungen angebracht. Es wurde außerdem noch angemerkt, dass Multimandantenfähigkeit sowie eine standardisierte Möglichkeit zur textuellen Beschreibung von IT Infrastrukturen im Kubernetes Umfeld vonnöten sind. Diese letzten beiden Punkte sind irrelevant, da sie implizit von Kubernetes schon erfüllt werden, bzw. mit Bordmitteln erfüllt werden können. Im zweiten Workshop wurde zu dieser Kategorie noch das Testen von Images und ganzen Softwarelösungen hinzugefügt. Testing muss schon im Entwicklungsstadium passieren und hat in diesem Sinn nichts mit Kubernetes zu tun.

Zum Themenblock rund um den Designprozess haben sich im ersten Workshop drei wichtige Punkte ergeben. Zum Ersten die Hinterfragung, ob Kubernetes oder die Verwendung von Containern an sich die richtige Wahl sind. Dies unterstreicht die Aussage und Relevanz von Kapitel 4.2. Des Weiteren wurde eingebracht, dass die Etablierung von Rollen im Kubernetes Umfeld schon vor der Einführung des Systems erfolgen muss. Das heißt, dass definiert sein muss welche Mitarbeiter welche Tätigkeiten im Bereich des Kubernetes Systems erfüllen. Der dritte Punkt ist die Integration des neuen Systems in die bestehende Supportorganisation des Unternehmens. Das bedeutet, dass das Supportpersonal für etwaige Kubernetes Tätigkeiten geschult und vorbereitet werden muss. Außerdem muss geregelt werden welche Supportanfragen und Problemfälle von wem behandelt werden. Im zweiten Workshop wurde ebenfalls nochmals das Thema Support aufgebracht, diesmal jedoch mit Fokus auf Anfragen von etwaigen Endkunden. Diese Anforderung ist auch bei der Integration in die bestehende Supportorganisation mit zu betrachten.

Im Bereich der Rahmenbedingung wurden in den beiden Workshops einige neue Anforderungen identifiziert, sowie bestehende bestätigt. Die insgesamt fünf Anforderungen "[AR01] Lizenz/Open-Source", "[AR02] Availability", "[AR03] Support", "[AR05] Grafische Benutzerschnittstelle - Cluster" sowie "[AR10] Public-Cloud Strategie" wurden von den Workshopteilnehmern genannt und untermauern somit die ausgearbeitete Theorie. Zusätzlich zu den bereits bekannten, wurden fünf neue Anforderungen identifiziert, wovon vier auch in den Interviews nochmals eingebracht wurden. Diese drehen sich unter anderem um das Thema Long-Time-Support oder das Verhindern eines Vendor Lock-In. Speziell die Auslagerung von Support und Betrieb an einen Service-Provider ([AR13]) ist an dieser Stelle hervorzuheben, da dieser Punkt nur in den Workshops, und in keinem Interview, eingebracht wurde.

Was das Thema der funktionalen Anforderungen betrifft, so wurden auch hier insgesamt neun bestehende Anforderungen als solche bestätigt. Die Anforderungen mit den Nummern [AF02], [AF05], [AF07], [AF08], [AF09], [AF13], [AF14], [AF17] und [AF19] wurden von den Teilnehmern zu diesem Fragenbereich eingebracht. Zusätzlich dazu wurden noch einige neue Anforderungen identifiziert.

Insgesamt sieben neue Punkte zum Thema funktionale Anforderungen konnten aus den beiden Workshops gewonnen werden. Dabei handelt es sich um Themen wie Dokumentation, Integration in die bestehende Infrastruktur, Richtlinien, verschlüsselte Kommunikation und Backup Funktionalitäten. Zwei der sieben identifizierten Punkte wurden nicht in die Liste der funktionalen Anforderungen aufgenommen. Es wurde die Überprüfung der Workload auf gewisse Policies verlangt, was jedoch ein Applikationsthema darstellt und nichts mit dem Cluster zu tun hat. Außerdem wurde angemerkt, dass eventuell ein Tracing System vonnöten sein könnte. Auch dies stellt jedoch eine zusätzliche Applikation dar, welche nichts mit dem Cluster zu tun hat. Solche Tracing Systeme setzen jedoch unter Umständen ein Service Mesh, sowie ein Logging und Metrik-System voraus. Es ist an dieser Stelle festzuhalten, dass der Großteil der eingebrachten Punkte dieses Bereichs von den Teilnehmern aus technischen Positionen kam.

Beim letzten Themengebiet, den Qualitätseigenschaften, ist interessant zu beobachten, dass insgesamt 17 Anforderungen in den Workshops identifiziert wurden. Davon sind fünf bereits aus der Theorie bekannt, 12 davon wurden neu identifiziert. Im Vergleich zu den anderen Themenbereichen wurden hier also wesentlich mehr neue Anforderungen identifiziert als bestehende bestätigt. Von den bestehenden Anforderungen wurden jene mit den Nummern [AQ02], [AQ04], [AQ06], [AQ08] und [AQ09] im Workshop genannt. Bei den zusätzlich identifizierten Anforderungen geht es um Themen wie Dokumentationsqualität, Usability, Benchmarks, Administrationsaufwand und Schulungsaufwand. Von den 12 neuen Qualitätseigenschaften wurden acht ebenfalls in den Interviews nochmal erwähnt.

An dieser Stelle sei angemerkt, dass der Löwenanteil, nämlich 22 der 25 im Praxisteil identifizierten Anforderungen, von den Workshops stammt. Im ersten Workshop konnten bereits 18 neue Punkte gewonnen werden, im zweiten nochmals vier. Dieser enorme Anteil ist jedoch ausschließlich auf die chronologische Reihenfolge der Durchführung zurückzuführen. Lediglich vier Punkte (AQ18, AQ21, AQ22, AQ23) wurden alleine in den Workshops eingebracht, alle anderen Themen wurden in den Interviews ebenfalls behandelt.

Die Ergebnisse der Workshops wurden um die Ergebnisse der branchenspezifischen Experteninterviews erweitert, welche im folgenden Abschnitt angeführt sind. Wie die gesammelten Ergebnisse in das finale Modell eingearbeitet wurden, bzw. wie die neu gewonnen Erkenntnisse im Modell ersichtlich sind, wird in Kapitel 5.4 erklärt.

5.3 Branchenspezifische Experteninterviews

Die erwähnten Workshops hatten das Ziel, mehrere Bereiche eines Unternehmens bzgl. ihrer Anforderungen zu einem Kubernetes System zu befragen. Im Gegensatz dazu wurden mehrere Experteninterviews durchgeführt die das Ziel hatten, spezifische Anforderungen aus weiteren Branchen zu identifizieren. Dies sollte dabei helfen Anforderungen zu identifizieren die eventuell in anderen Unternehmen als solche gar nicht auftreten. Aus diesem Grund wurde versucht aus verschiedensten Branchen Unternehmen zu finden die bereit sind an einem Interview teilzunehmen. In Tabelle 5.2 ist ersichtlich wie viele Unternehmen in welchen Branchen kontaktiert wurden und mit wie vielen davon ein Interview durchgeführt wurde.

Insgesamt wurden 23 Unternehmen kontaktiert, wovon nur 6 dazu bereit waren sich für ein Interview zur Verfügung stellten. Es wurde bereits im Vorfeld davon ausgegangen, dass nicht jedes Unternehmen für ein Interview bereit sein würde, daher wurden teilweise pro Branche mehrere Anfragen ausgeschickt. Ziel war es, pro Branche ein Interview durchzuführen, die maximale Anzahl jedoch nicht überzustrapazieren.

Besonders interessant waren die Rückmeldungen aus dem Bereich Gesundheit, da hier drei von vier Unternehmen rückgemeldet haben, dass Kubernetes noch nicht im Einsatz ist. Dies ist spannend, da diese Unternehmen vor allem hohe Ansprüche an Qualität, Verfügbarkeit und Betriebsthemen haben. Gerade in diesen Bereichen kann Kubernetes seine Stärken gut ausspielen. Dass diese Unternehmen derzeit noch nicht auf Kubernetes gewechselt haben, lässt sich eventuell genau auf diese hohen Verfügbarkeitsanforderungen zurückführen. Ein Wechsel und die damit einhergehenden Probleme könnten einfach noch zu riskant sein.

Branche (durchgeführt/angefragt)	keine Antwort	keine Zeit	nicht in Verwendung	durchgeführt
App (0/3)	2	1	0	0
Finance (1/1)	0	0	0	1
Gaming (0/3)	2	0	1	0
Gesundheit (0/4)	1	0	3	0
Glücksspiel (0/1)	0	1	0	0
Industrie (1/3)	1	0	1	1
Infrastruktur (1/3)	0	1	1	1
IT (2/3)	0	1	0	2
Logistik (0/1)	0	0	1	0
Telekommunikation (1/1)	0	0	0	1

Tab. 5.2: Anfragen für Experteninterviews in verschiedenen Branchen

5.3.1 Planung und Durchführung der Experteninterviews

Analog zu den Workshops wurde auch den Interviewteilnehmern vorab ein One-Pager zugesendet. Dieser sollte den Teilnehmern dabei helfen sich auf ihr Interview vorzubereiten und einige Einblicke in das Thema geben. Der One-Pager enthielt die vier groben Bereiche des Interviews, nämlich den Designprozess sowie die drei Arten von Anforderungen. Zusätzlich wurden jeweils einige kleine Beispiele angeführt um die Fragestellungen zu verdeutlichen. Der an die Interviewpartner ausgesendete One-Pager ist in Anhang A.20 ersichtlich. Die Interviews dauerten im Durchschnitt ungefähr 30 Minuten, lediglich eines davon nahm knapp eine Stunde in Anspruch. Dieser Ausreißer ist auf eine ausführliche Erklärung der technischen Infrastruktur des Unternehmens zurückzuführen. Die folgenden Absätze beschreiben kurz die Interviewpartner und deren Tätigkeit, das Unternehmen sowie den Use-Case, welchen das jeweilige Unternehmen mit Kubernetes abdeckt. Diese Auflistung spiegelt auch die chronologische Abfolge der Interviews wieder. In Tabelle 5.3 ist angeführt wann und wie die Interviews durchgeführt wurden.

Unternehmen	Datum	Uhrzeit	Medium
A	DI 20.10.2020	15.30 Uhr -16.00 Uhr	Cisco Webex Teams
B	DI 21.10.2020	09.15 Uhr -09.45 Uhr	Skype
C	MI 21.10.2020	18.00 Uhr -19.00 Uhr	Google meet
D	FR 6.11.2020	10.00 Uhr -10.30 Uhr	Cisco Webex Teams
E	FR 6.11.2020	14.30 Uhr -15.00 Uhr	Cisco Webex Teams
F	MO 9.11.2020	17.00 Uhr -17.30 Uhr	Cisco Webex Teams

Tab. 5.3: Durchführungszeitpunkte der Experteninterviews

Unternehmen A ist ein österreichisches Unternehmen aus dem Finanzsektor. Das Unternehmen verwendet seit ungefähr zwei Jahren Kubernetes[177], vorrangig um damit externe Schnittstellen sowie interne Event-Streaming-Dienste zur Verfügung zu stellen[178]. Derzeit betreibt das Unternehmen seine Kubernetes Cluster ausschließlich On-Premises, spielt jedoch mit dem Gedanken Cloud Lösungen zu verwenden[179]. Es wird eine proprietäre Kubernetes Distribution verwendet, welche sich gut in die bestehende Infrastruktur integrieren ließ. Das Team in diesem Unternehmen umfasst mehrere Mitglieder, die nahezu ausschließlich die Kubernetes Umgebung betreiben und weiterentwickeln[180]. Der Vorschlag kubernetes im Unternehmen einzuführen kam eindeutig von Seiten der Techniker, genauer gesagt aus dem Team welches für Plattformen verantwortlich ist[181]. Als Motivation in Richtung von Kubernetes zu denken wurde angeführt, dass man versuchte "Flexibler, skalierbarer zu werden, und dann war eigentlich der Weg in Richtung Kubernetes eine logische Evolution von der alten Plattform. Und deswegen wussten wir immer schon, wo das Kubernetes Thema aufgekommen ist, dass wir dahin wollen. Weil es ist dann ein Industriestandard wo wir eine ausgebaute Automatisierung anwenden können. So sind wir zu dem Kubernetes Thema gekommen und haben im Unternehmen mit Kubernetes begonnen."[182] Die Entscheidung diesen Schritt tatsächlich zu gehen wurde im Mittelmanagement getroffen. Die Kubernetes Einführung wurde von einem Pilotprojekt begleitet, welches mittlerweile als erfolgreich betrachtet wird[183]. Der Interviewpartner aus Unternehmen A ist seit fast 20 Jahren im Unternehmen tätig. Seine Betätigungsfelder waren durchwegs im Bereich von Plattformen und deren Administration angesiedelt.

Das Geschäftsmodell von Unternehmen B ist die Auswertung von Bilddaten. Auch Unternehmen B verwendet seit ungefähr zwei Jahren Kubernetes[184]. In diesem Betrieb wird jedoch ausschließlich auf Cloud Lösungen gesetzt, was unter anderem an der Verfügbarkeit von essentiellen Daten in der Cloud liegt[185]. Die Workload ist in diesem Fall stark auf die Auswertung riesiger Datenmengen fokussiert. Hierbei kommen diverse Programmiersprachen und Frameworks zur Anwendung[186]. Die Cloud ist hier ebenfalls ein ausschlaggebendes Kriterium, da für den Einsatz von Kubernetes lediglich ein kleines Team zuständig sind. Somit spielten Punkte wie Usability und Administrationsaufwand eine entscheidende Rolle bei der Wahl der Distribution[187]. Auch in diesem Unternehmen kam die Idee Kubernetes auszuprobieren aus dem IT Team[188]. Die Motivation dafür war es eine Plattform zu finden, die es erlaubt sehr viele Berechnungen parallel durchzuführen. Ausschlaggebend war auch, dass es sich um eine Lösung handeln muss die schnell einsetzbar ist und mit Containern arbeiten kann. Das Unternehmen hat vor Kubernetes schon auf Container gesetzt[189]. Die Entscheidung Kubernetes tatsächlich zu verwenden wurde vom Top-Management an die IT-Abteilung übertragen. Es wurde ein bestimmtes Budget bereitgestellt und die IT-Abteilung hat sich für die Verwendung von Kubernetes entschieden[190]. Der Interviewpartner aus Unternehmen B ist seit ungefähr drei Jahren im Unternehmen und arbeitet dort seit zwei Jahren mit Kubernetes. Seine Fachbereiche sind vor allem das Paketieren und Ausrollen von Software, sowie die Betreuung der Plattformen[191].
Das Hauptbetätigungsfeld von Unternehmen C ist die Produktion von Metallverarbeitungsmaschinen. Auch dieses Unternehmen verwendet seit ungefähr zwei Jahren Kubernetes, derzeit noch ausschließlich On-Premises[192].

[177] Vgl. Anhang A.22, Paragraph 42
[178] Vgl. Anhang A.22, Paragraphen 18, 38
[179] Vgl. Anhang A.22, Paragraphen 30, 32
[180] Vgl. Anhang A.22, Paragraph 12
[181] Vgl. Anhang A.22, Paragraph 12
[182] Vgl. Anhang A.22, Paragraph 8
[183] Vgl. Anhang A.22, Paragraph 16
[184] Vgl. Anhang A.23, Paragraph 2
[185] Vgl. Anhang A.23, Paragraph 8
[186] Vgl. Anhang A.23, Paragraph 22
[187] Vgl. Anhang A.23, Paragraph 38
[188] Vgl. Anhang A.23, Paragraph 2
[189] Vgl. Anhang A.23, Paragraphen 2, 4
[190] Vgl. Anhang A.23, Paragraph 12
[191] Vgl. Anhang A.23, Paragraph 2
[192] Vgl. Anhang A.24, Paragraph 6

Die Workloads auf diesen Clustern sind einfache Webservices und Microservices, hauptsächlich jedoch Programme zur Verarbeitung von Telemetrie Daten. Das bedeutet, dass das Unternehmen Kubernetes nutzt um Messdaten ihrer Maschinen zu sammeln und auszuwerten[193]. Derzeit passiert dies noch im Unternehmen, es gibt jedoch schon Bestrebungen Cloud Services zu nutzen oder auch Cluster zum eigentlichen Endkunden zu liefern[194]. Der aktuelle Stack basiert auf Upstream Kubernetes, darauf werden Tools wie Apache Kafka und MongoDB betrieben[195]. In diesem Unternehmen wurde in den vergangenen zwei Jahren ein eigenes Team für die Betreuung und Weiterentwicklung der Kubernetes basierten Plattform geformt[196]. Die Idee Kubernetes zu verwenden kam auch hier vom IT Personal, in diesem Fall sogar vom Interviewpartner selbst. Dem Team wurde bei der Verwendung der Technologie freie Hand gelassen und man entschied sich für Kubernetes. Als Motivation wurde die Vorgabe des Vorstands genannt, das Unternehmen und seine Produkte mehr zu digitalisieren[197]. Der Interviewpartner dieses Unternehmens ist seit vier Jahren im Unternehmen und kann auf mehr als 30 Jahre Berufspraxis in der IT Branche verweisen[198]. Er war initial für die Etablierung der entstandenen Lösung verantwortlich und betreibt diese fortlaufend mit seinem Team, welches mittlerweile über zwei Dutzend Personen umfasst. Das Unternehmen ist Weltmarktführer auf seinem Gebiet, beschäftigt weltweit über 6000 Mitarbeiter und hat seinen Hauptsitz in Europa.

Unternehmen D ist im Bereich Infrastruktur tätig, bietet jedoch auch viele andere IT Services an. Unter anderem ist Unternehmen D auch Kubernetes-as-a-Service Provider, sowie Beratungsunternehmen für Kubernetes und Cloud Projekte. In diesem Unternehmen wird, analog zu Unternehmen A, ausschließlich auf eine proprietäre Kubernetes Distribution gesetzt. Das Unternehmen hat ein großes Team hochqualifizierter Mitarbeiter, die sich mit dieser Thematik befassen[199]. Die Infrastruktur wird nur On-Premises betrieben, unter anderem weil zum Zeitpunkt der Einführung die Public-Cloud Angebote noch nicht ausgereift genug waren[200]. Kubernetes ist hier ebenfalls seit etwas mehr als zwei Jahren produktiv im Einsatz[201]. Das System wird einerseits für interne Services verwendet, andererseits wird es auch an Businesskunden als Service verkauft. Die Workload ist dementsprechend breit gefächert, immer davon abhängig was der jeweilige Businesskunde benötigt[202]. Der Vorschlag Kubernetes zu verwenden wurde von den Technikern im Unternehmen eingebracht. Diese haben erkannt, dass die Anfrage am Markt für diese Technologie steigt, und es sich auch für interne Anwendungen anbieten würde[203]. Die finale Entscheidung wurde, bedingt durch das finanzielle Risiko, vom Management getroffen[204]. Der Interviewpartner aus Unternehmen D ist zertifizierter Linux und Kubernetes Administrator sowie Berater für Cloud- und Kubernetes-Themen. Er ist seit circa 1,5 Jahren im Unternehmen tätig, hat jedoch weitaus längere Erfahrung im Container und Kubernetes Bereich.

Unternehmen E ist im Telekommunikationswesen tätig. Die Besonderheit in diesem Konzern ist, dass das Unternehmen einerseits Kubernetes-as-a-Service Provider ist, andererseits aber auch Kubernetes verwendet. Eine Business Unit im Unternehmen stellt Endkunden Kubernetes, sowie andere Cloud Dienste zur Verfügung. Andere Business Units verwenden eine eigene Kubernetes Lösung für Services im Haus. Die Business Unit, welche als Provider auftritt, baut vollständig auf Upstream Kubernetes.

[193] Vgl. Anhang A.24, Paragraph 14
[194] Vgl. Anhang A.24, Paragraph 34
[195] Vgl. Anhang A.24, Paragraph 14
[196] Vgl. Anhang A.24, Paragraph 35
[197] Vgl. Anhang A.24, Paragraphen 16, 17-18
[198] Vgl. Anhang A.24, Paragraph 2
[199] Vgl. Anhang A.25, Paragraph 4
[200] Vgl. Anhang A.25, Paragraph 18
[201] Vgl. Anhang A.25, Paragraph 14
[202] Vgl. Anhang A.25, Paragraphen 4, 8, 10
[203] Vgl. Anhang A.25, Paragraph 16
[204] Vgl. Anhang A.25, Paragraph 38

Im Haus wird einerseits eine proprietäre Lösung verwendet, zusätzlich jedoch auch noch ein Upstream Cluster mit Rancher Management Plane[205]. Die Workload des Unternehmens ist sehr durchgemischt, da Kubernetes in vielen verschiedenen Bereichen eingesetzt wird. Der Vorschlag bzw. die Idee auf Kubernetes zu setzen kam in allen Teilen des Unternehmens von technischen Teams. Vor allem die Betriebsteams haben sich stark für die Verwendung ausgesprochen. Auch Themen wie schnellere Entwicklungszyklen und leichteres Deployment wurden als Motivationstreiber genannt[206]. Der Interviewpartner aus Unternehmen E blickt auf eine 20 Jahre lange Karriere im Konzern zurück, wovon er die letzten drei Jahre im Unternehmen E verbrachte. Er hat langjährige Erfahrung in der Administration von unixoiden Systemen und unterstützt derzeit Kunden bei der Umstellung auf Cloud und Container Technologien.

Das letzte Interview wurde mit Unternehmen F geführt, welches eine spezielle Software für Businesskunden entwickelt, vertreibt und zur Verfügung stellt. Daher ist die Workload dieses Unternehmens sehr homogen und beschränkt sich großteils auf Webseiten und die dazugehörigen Backend-Applikationen[207]. Als Kubernetes Distribution wurde EKS gewählt, das Kubernetes Angebot der Amazon AWS Cloud. Außerdem werden noch einzelne Cluster in lokalen Datacentern betrieben, dort aufgesetzt auf OpenStack[208]. Dieses Unternehmen ist schon sehr früh zur Verwendung von Public-Cloud-Service übergegangen, eine genaue zeitliche Angabe konnte nicht gemacht werden, da der Interviewpartner erst seit ungefähr einem Jahr im Betrieb arbeitet. Kubernetes ist auf alle Fälle schon länger im Einsatz als der Gesprächspartner im Unternehmen tätig ist[209]. Als Motivation für die Verwendung von Kubernetes wurde die Tatsache genannt, dass schon vorher auf Container gesetzt wurde. Daher ist Kubernetes als nächster Schritt naheliegend[210]. Das Unternehmen ist international tätig, hat über 4000 Mitarbeiter weltweit und betreibt auch Standorte in Österreich.

In Tabelle 5.4 sind nochmals die Charakteristiken aller befragten Unternehmen sowie der Interviewpartner zusammengefasst dargestellt. Es sind jeweils Alter, Position und Dienstzeit des Interviewpartners im Unternehmen ersichtlich, sowie die Anzahl der Mitarbeiter, der Sektor und die Kubernetes Merkmale des Unternehmens.

Unternehmen	Alter	Position	Dienstzeit Jahre	Anzahl Mitarbeiter	Sektor	Kubernetes Anwendung	Kubernetes Distribution	Kubernetes Verwendung
A	35-40	Platform Engineer	15-20	> 500	Finanz	In-House Anwendung für Web-Schnittstellen	On-Premises proprietär	~2 Jahre
B	30-35	Software Developer	3	> 50	Datenauswertung	Auswertung von Bilddaten	Public-Cloud AWS kops	~2 Jahre
C	50-55	Abteilungsleiter	4	>5000	Industrie	Verarbeitung von Telemetrie Daten	On-Premises Open-Source	2 Jahre
D	30-35	Consultant	1,5	>500	Infrastruktur	In-House Anwendungen Kundenprojekte	On-Premises proprietär	>2 Jahre
E	35-40	Cloud Solution Architect	20	> 50	Telekommunikation	In-House Anwendung Cloud Provider	On-Premises diverse	n.A.
F	25-30	Cloud Engineer	1	>4000	IT	Web-Anwendung	Public-Cloud Amazon EKS	>2 Jahre

Tab. 5.4: Zusammenfassung der befragten Unternehmen

Die Aufzählung der interviewten Unternehmen soll ein Gefühl vermitteln in welchen Branchen Kubernetes bereits eingesetzt wird und wofür. Im folgenden Kapitel wird detailliert auf die Analyse der einzelnen Experteninterviews eingegangen.

[205] Vgl. Anhang A.26, Paragraphen 2, 8, 14
[206] Vgl. Anhang A.26, Paragraphen 10, 12, 14
[207] Vgl. Anhang A.27, Paragraphen 38, 68
[208] Vgl. Anhang A.27, Paragraphen 10, 12, 84
[209] Vgl. Anhang A.27, Paragraph 22
[210] Vgl. Anhang A.27, Paragraph 22

5.3.2 Analyse der Interviews

Bei der Analyse der Interviews wurde, wie bereits in Abschnitt 5.1 erklärt, speziell darauf geachtet welche Codes gefunden werden können. Die identifizierten Codes konnten fast immer direkt Anforderungen zugeordnet werden. Es wurde nicht betrachtet welche eventuellen Ausprägungen ein Code haben kann, sonder ob sich eine Textstelle generell mit einem Thema befasst. Im Folgenden wird detailliert aufgeschlüsselt welche Informationen im jeweiligen Interview neu identifiziert wurden, wo das theoretische Modell untermauert wird und in welchen Paragraphen die Information zu finden ist.

Unternehmen A: Bei der Auswertung des Interviews mit Unternehmen A ist interessant zu sehen, dass es einige Segmente gibt die eine enormen Informationsgehalt haben. So wurden beispielsweise in den Paragraphen 24, 42 und 56 Inhalte gefunden, die auf mehrere Codes ansprechen. Dies ist darauf zurück zu führen, dass in manchen Sätzen einige Themen gleichzeitig angesprochen wurden. Wie schon in Kapitel 5.1 angemerkt, war es deshalb nicht sinnvoll möglich solche Paragraphen zu paraphrasieren. Davon abgesehen hätte dies für die Auswertung auch keinen Mehrwert gehabt. Die Position der codierte Segment gibt den Aufbau des Interview Leitfadens andeutungsweise wieder. Anforderungen des Bereichs Rahmenbedingungen befinden sich eher am Anfang, bis circa Paragraph 32. Die funktionalen Anforderungen befinden sich im Bereich von Paragraph 40-56. Lediglich die Qualitätsanforderungen sind auf alle Teile des Interviews verteilt. Insgesamt wurden in diesem Interview 15 Anforderungen aus der Theorie bestätigt. Vier Punkte, die schon in den Workshops genannt wurden, wurden auch hier aufgebracht. Es konnte eine neue Qualitätsanforderung (AQ11) identifiziert werden. Die Teile des Interviews die sich mit der Kubernetes Umgebung des Unternehmens und deren Entstehungsgeschichte befassen befinden sich ganz am Anfang, im Bereich von Paragraph 8-30. Das Interview hat insgesamt ziemlich genau 30 Minuten gedauert, das resultierende Protokoll umfasst 59 Paragraphen. In Tabelle 5.5 sind die Ergebnisse der Codierung des Interviews mit Unternehmen A ersichtlich. Das vollständige Protokoll befindet sich in Anhang A.22.

Anforderung	Erstmals identifiziert	Häufigkeit Erwähnungen	Paragraph(en)
AR01	Theorie	2	24, 26
AR04	Theorie	1	12
AR07	Theorie	2	20, 24
AR09	Theorie	1	12
AR10	Theorie	2	30, 32
AR16	Workshop 1	1	22
AF02	Theorie	1	42
AF07	Theorie	1	42
AF08	Theorie	1	22
AF09	Theorie	1	42
AF12	Theorie	1	46
AF13	Theorie	1	42
AF15	Theorie	1	42
AF21	Workshop 2	1	42
AF23	Workshop 1	2	12, 56
AQ02	Theorie	1	52
AQ03	Theorie	2	12, 14
AQ06	Theorie	1	24
AQ11	Unternehmen A	2	44,
AQ20	Workshop 1	1	24

Tab. 5.5: Ergebnisse: Interview Unternehmen A

Unternehmen B: Bei der Auswertung des Interviews mit Unternehmen B fällt auf, dass sich der Großteil der identifizierten Anforderungen im Bereich der Qualitätseigenschaften befindet. Es konnte eine Rahmenbedingung (AR11) in diesem Interview gefunden werden, die noch nicht bekannt war. Auf Grund der Anzahl der Erwähnungen sticht vor allem AQ08, der zeitliche Aufwand für die Installation des Systems, bei diesem Interview hervor. Der Interviewpartner hat mehrmals erwähnt, dass eine rasche Verfügbarkeit des Clusters für das Unternehmen essentiell ist. Distributionen die lange und kompliziert installiert werden müssen sind daher in diesem Fall undenkbar. Es ist hier sehr gut erkennbar, dass nur sehr wenige funktionale Anforderungen genannt wurden. Dies könnte unter anderem daran liegen, dass dieses Unternehmen einen sehr spezifischen Use-Case hat. Daher werden keine erweiterten funktionalen Anforderungen gestellt, Auch dieses Interview dauerte knapp 30 Minuten, das Transkript hat 39 Paragraphen. Die Anzahl der Absätze ist signifikant weniger als bei Unternehmen A, was daran liegt, dass der Interviewpartner längere Antworten gab. In diesem Interview wurden 10 Punkte aus dem Theorieteil ebenfalls genannt. Vier der Punkte, welche in den Workshops identifiziert werden konnten wurden auch hier eingebracht. Ein Punkt war schon aus dem Interview mit Unternehmen A bekannt und eine neue Rahmenbedingung (AR11) konnte identifiziert werden. In Tabelle 5.6 sind die Ergebnisse der Codierung des Interviews mit Unternehmen B ersichtlich. Das vollständige Protokoll befindet sich in Anhang A.23.

Anforderung	Erstmals identifiziert	Häufigkeit Erwähnungen	Paragraph(en)
AR02	Theorie	1	16
AR07	Theorie	1	18
AR10	Theorie	2	6, 10
AR11	Unternehmen B	2	8, 38
AR12	Workshop 1	2	16, 18
AF04	Theorie	2	30, 32
AF05	Theorie	1	24
AF15	Theorie	1	26
AQ03	Theorie	2	12, 32
AQ04	Theorie	2	20, 38
AQ08	Theorie	4	8, 12, 20, 34
AQ09	Theorie	1	38
AQ11	Unternehmen A	1	20
AQ14	Workshop 1	1	10
AQ19	Workshop 1	2	8, 38
AQ24	Workshop 1	1	10

Tab. 5.6: Ergebnisse: Interview Unternehmen B

Unternehmen C: Das Interview mit Unternehmen C dauerte etwas mehr als eine Stunde, was dem doppelten der restlichen Interviews entspricht. Trotzdem enthält das Transkript nur 49 Paragraphen. Dies ist darauf zurückzuführen, dass der Interviewpartner einerseits sehr ausschweifend erklärt hat. Andererseits wurden die erwähnten Themen auch relativ detailliert behandelt. Die Anzahl an Themen, sprich codierte Textelemente, ist im Vergleich dazu nicht doppelt so viel wie bei den anderen Interviews. Es wurden 16 Punkte besprochen die auch schon in der Theorie vorgekommen sind. Vier Punkte wurden schon in vorhergehenden Interviews eingebracht. Die ausschweifende Natur der Antworten kann zwar nicht an der Anzahl der verwendeten Codes, jedoch an der Anzahl codierter Elemente abgelesen werden. Viele Themen wurden drei, vier oder fünf mal genannt, was bei anderen Interviews selten der Fall war.

Der Interviewpartner hat in den Paragraphen 24, 30 und 32 mehrmals erwähnt, dass das Thema Kubernetes im Unternehmen durch einfaches probieren angegangen wurde. Es wurde unterstrichen, dass beim Design auf Grund von Unwissenheit einige Fehler gemacht wurden. Diese wurden im Nachhinein, mit der Hilfe von externen Partnern, behoben, was zusätzliche Aufwände zur Folge hatte. Durch diese Schilderung wird das Problem, welches diese Arbeit behandelt, sehr anschaulich erläutert.

Bei diesem Interview wurden außerdem noch zwei Themen angesprochen, welche nicht in den Anforderungskatalog aufgenommen wurden. Wie in Abschnitt 5.2.2 schon angemerkt, wurden auch diese beiden Punkte der Kategorie "Out-Of-Scope" zugeteilt. Einerseits wurde fünf mal, in den Paragraphen 8, 13, 32, 34 und 36, aufgezeigt, dass zukünftig auch beim Kunden Kubernetes Systeme laufen sollen. Ob diese der Kunde selbst betreibt, dies eine Drittfirma übernimmt oder das Unternehmen sich darum kümmern muss kann derzeit noch nicht gesagt werden. Außerdem ist dies eine eigene Designentscheidung und keine einzelne Anforderung. Der zweite Punkt, welcher zwar als wichtig kommuniziert wurde, jedoch keine Anforderung darstellt betrifft das Thema Skalierung. Es wurde vier mal angebracht, in den Paragraphen 8, 10, 26 und 34, dass die Programme am Cluster dynamisch skalierbar sein müssen. Dies ist keine eigene Anforderung, da diese Funktionalität standardmäßig von Kubernetes bereitgestellt wird. In Tabelle 5.7 sind die Ergebnisse der Codierung des Interviews mit Unternehmen C ersichtlich. Das vollständige Protokoll befindet sich in Anhang A.24.

Anforderung	Erstmals identifiziert	Häufigkeit Erwähnungen	Paragraph(en)
AR01	Theorie	3	8, 26, 34
AR02	Theorie	3	8, 10, 34
AR03	Theorie	3	8, 26, 34
AR07	Theorie	2	10, 30
AR09	Theorie	1	8
AR10	Theorie	4	6, 8, 34
AR11	Unternehmen B	1	30
AR16	Unternehmen A	2	8, 34
AF02	Theorie	1	34
AF04	Theorie	1	34
AF05	Theorie	1	34
AF07	Theorie	1	36
AF12	Theorie	2	14, 36
AQ02	Theorie	5	8, 24, 26, 34
AQ03	Theorie	3	26, 34, 36
AQ04	Theorie	2	28, 36
AQ06	Theorie	2	40, 42
AQ09	Theorie	3	2, 24, 36
AQ11	Unternehmen A	4	6, 24, 32, 36
AQ19	Unternehmen B	1	34

Tab. 5.7: Ergebnisse: Interview Unternehmen C

Unternehmen D: Im Interview mit Unternehmen D wurde in Absatz 32 nochmal explizit hervorgehoben warum es so wichtig ist, das Design von Kubernetes Systeme strukturiert und überlegt durchzuführen: "Ja es gibt immer wieder sehr spezielle Fälle, eben auch von Kundenseite. Aber das.. da kommt man dann in der Implementierung drauf, darum ist auch für mich das immer ganz wichtig das im Vorhinein abzuchecken. Wie schon erwähnt, irgendwelche Sonderlösungen dann im Nachgang zu implementieren das verursacht dann ein vielfaches an höheren Aufwand, im Vergleich zur initialen Installation."

Dieses Interview hat 13 der theoretisch schon bekannten Anforderungen bestätigt. Außerdem wurden noch zwei Qualitätseigenschaften genannt die aus einem Interview und einem Workshop schon bekannt waren. Die Antworten dieses Interviewpartners waren, im Vergleich zu Unternehmen C, sehr präzise und nicht ausschweifend. Trotz der Dauer von knapp 30 Minuten hat das Protokoll nur 39 Absätze, welche auch größtenteils nicht besonders lange sind. In Tabelle 5.8 sind die Ergebnisse der Codierung des Interviews mit Unternehmen D ersichtlich. Das vollständige Protokoll befindet sich in Anhang A.25.

Anforderung	Erstmals identifiziert	Häufigkeit Erwähnungen	Paragraph(en)
AR01	Theorie	1	22
AR02	Theorie	1	20
AR03	Theorie	2	14, 38
AR07	Theorie	1	20
AR09	Theorie	4	4, 6, 10, 38
AR10	Theorie	1	18
AF04	Theorie	1	28
AF05	Theorie	1	28
AF12	Theorie	2	28, 30
AF15	Theorie	1	34
AF19	Theorie	1	34
AQ02	Theorie	1	38
AQ03	Theorie	1	38
AQ11	Unternehmen A	1	4
AQ16	Workshop 1	1	14

Tab. 5.8: Ergebnisse: Interview Unternehmen D

Unternehmen E: Das interessante am Interview mit Unternehmen E ist die Konstellation in welcher Kubernetes dort verwendet wird. Beim Unternehmen handelt es sich um einen Konzern mit mehreren Tochterunternehmen. Eines dieser Tochterunternehmen stellt Public-Cloud-Services am Markt zur Verfügung und verwendet, bzw. betreibt, Kubernetes dafür. Ein zweiter Teil des Konzerns verwendet Kubernetes für interne Applikationen, baut dabei jedoch auf eine andere Distribution. Eine dritte Tochterfirma verwendet wiederum eine dritte Kubernetes Distribution für unternehmensinterne Applikationen. Diese Vielzahl an Use-Cases erklärt auch die verhältnismäßig hohe Anzahl an funktionalen Anforderungen und Qualitätseigenschaften die genannt wurden. Es ist auch interessant zu erwähnen, dass es trotz der Größe des Unternehmens nicht sonderlich viele Rahmenbedingungen gibt. In diesem Interview wurden keine neuen Punkte identifiziert, jedoch 23 bekannte Themen bestätigt. Vier davon wurden in Workshops eingebracht, die restlichen sind bereits aus der Theorie bekannt. In Tabelle 5.9 sind die Ergebnisse der Codierung des Interviews mit Unternehmen E ersichtlich. Das vollständige Protokoll befindet sich in Anhang A.26.

Unternehmen F: Dieses Interview war mit knapp 25 Minuten das kürzeste, hat jedoch 107 Absätze. Der Interviewpartner hat großteils sehr kurze Antworten gegeben, was die Anzahl der Paragraphen erklärt. Bei diesem Interview wurden bei weitem am wenigsten Themen behandelt. Es konnten lediglich neun Anforderungen identifiziert werden, wovon drei schon in Workshops eingebracht wurden und die restlichen aus der Theorie bekannt sind. Dies ist jedoch keinesfalls auf mangelnde Kompetenz des Interviewpartners zurückzuführen, sondern auf die verhältnismäßig einfache Natur des Use-Cases des Unternehmens. Es werden relativ wenige Rahmenbedingungen oder funktionale Anforderungen gestellt. Der gesamte Use-Case kann mit absoluten Kubernetes Basismitteln erfüllt werden. Die Anforderungen an die Qualität richten sich hauptsächlich an den Preis, sowie an den Betriebs- und Schulungsaufwand. Das Unternehmen unterhält seine Cluster mit EKS in der Amazon Cloud, darauf werden lediglich verhältnismäßig einfache Webanwendungen betrieben.

Anforderung	Erstmals identifiziert	Häufigkeit Erwähnungen	Paragraph(en)
AR01	Theorie	3	8, 16, 60
AR03	Theorie	3	16, 18, 60
AR05	Theorie	1	12
AR12	Workshop 1	2	20, 24
AF02	Theorie	1	27/28
AF04	Theorie	1	38
AF07	Theorie	1	36
AF08	Theorie	2	42, 46
AF12	Theorie	2	30, 32
AF14	Theorie	1	12
AF21	Workshop 2	2	27/28, 30
AF22	Workshop 1	2	12, 46
AF23	Workshop 1	2	32, 34
AQ02	Theorie	3	18, 36, 44
AQ03	Theorie	4	14, 16, 58
AQ04	Theorie	2	44, 48
AQ05	Theorie	1	44
AQ06	Theorie	1	50
AQ08	Theorie	1	44
AQ09	Theorie	1	46
AQ12	Theorie	1	22
AQ13	Theorie	1	44
AQ17	Theorie	1	16

Tab. 5.9: Ergebnisse: Interview Unternehmen E

Eine nennenswerte Anforderung dieses Unternehmens, welche jedoch nicht in den Anforderungskatalog aufgenommen wurde, ist der lokale Betrieb von Kubernetes. Damit ist gemeint, dass das Unternehmen für die Entwicklung auf lokale Cluster, auf den Notebooks der Entwickler, setzt. Dafür gibt es eigene Distributionen, welche jedoch in dieser Form nicht für den produktiven Betrieb ausgelegt sind. Dieses ist das einzige aller befragten Unternehmen, welches lokale Cluster für die Entwicklung verwendet.

Es sei an dieser Stelle angemerkt, dass der Interviewpartner, auf Grund seiner kurzen Betriebszugehörigkeit, eventuell noch nicht vollständig über die Unternehmensinfrastruktur Bescheid wusste. Die Antworten waren präzise, könnten jedoch auf Grund von Unwissenheit unvollständig sein. Beispielsweise wurde in Paragraph 24 angemerkt, dass der Interviewpartner nicht weiß wer am Entscheidungsprozess für die Verwendung von Kubernetes beteiligt war. In Tabelle 5.10 sind die Ergebnisse der Codierung des Interviews mit Unternehmen F ersichtlich. Das vollständige Protokoll befindet sich in Anhang A.27.

Insgesamt wurden sechs Interviews mit unterschiedlichen Unternehmen durchgeführt. Dieses Kapitel hat aufgezeigt welche Themen in welchen Interviews genannt wurden. Die Paragraphen in denen die Anforderungen im jeweiligen Interview wurden angeführt, ebenso wie deren erstmalige Erwähnung. Es wurden etwaige Besonderheiten in den Interviews hervorgehoben und wichtige Punkte nochmals unterstrichen. Der folgende Abschnitt wird diese feinen Ergebnisse auf einer höheren Ebene zusammenfassen und aufbereiten.

Anforderung	Erstmals identifiziert	Häufigkeit Erwähnungen	Paragraph(en)
AR02	Theorie	2	40, 52
AR10	Workshop 2	3	10, 72, 74
AR14	Workshop 2	2	12, 18
AF04	Theorie	1	58
AF05	Theorie	1	58
AF15	Theorie	2	54, 56
AQ03	Theorie	2	104, 106
AQ04	Theorie	1	76
AQ19	Workshop 1	1	106

Tab. 5.10: Ergebnisse: Interview Unternehmen F

5.3.3 Zusammenfassung der Ergebnisse der Experteninterviews

Die Ergebnisse der einzelnen Interviews wurden, analog zu den Workshop Ergebnissen, speziell mit dem Fokus auf neu zu identifizierende Anforderungen ausgewertet. Zwar wurden auch Fragen bzgl. der Einführung von Kubernetes in den jeweiligen Unternehmen gestellt, es konnte jedoch kein neuer Input für den entwickelten Designprozess gewonnen werden. Von Unternehmen A,C und D wurde erneut angeführt, wie wichtig eine strukturierte Vorgehensweise beim Design eines Kubernetes Systems und dessen Einführung im Unternehmen ist. Die Unternehmen A, B und C gaben außerdem an, sich dem Thema Kubernetes ganz pragmatisch durch Ausprobieren angenähert und Erfahrung durch Anwendung gesammelt zu haben. Eine Vorstellung der jeweiligen Unternehmen und ihrer Use-Cases erfolgte bereits im Abschnitt 5.3.1. Dieser Abschnitt konzentriert sich konkret auf die Zusammenfassung der Ergebnisse der sechs Interviews, bezogen auf die drei Gruppen von Anforderungen. Am Ende wird nochmals grafisch verdeutlicht und zusammengefasst woher welche Erkenntnisse gewonnen werden konnten.

Rahmenbedingungen: Von den aus der Theorie bekannten zehn Rahmenbedingungen wurden in den sechs Interviews insgesamt acht Rahmenbedingungen bestätigt. Lediglich [AR06], eine grafische Benutzerschnittstelle für eine Komponente, und [AR08], die Bevorzugung einer bestimmten Linux Distribution, wurden von keinem Interviewpartner genannt. Speziell die Punkte [AR10], Verwendung von Public-Cloud, [AR01], Open-Source und Lizenzen, sowie [AR03], Enterprise Support, wurden besonders oft thematisiert. Zusätzlich zu den bestehenden zehn Rahmenbedingungen konnten in den Interviews noch fünf weitere Punkte identifiziert werden. Diese überschneiden sich in drei Fällen mit bereits genannten Punkten aus den Workshops, nämlich [AR12], [AR14] und [AR16]. Einer dieser fünf Punkte wurde nicht in die Liste der Rahmenbedingungen aufgenommen. Es wurde angemerkt, dass Kubernetes immer häufiger bei oder für Kunden betrieben werden muss. Dies ist jedoch nicht als Anforderung zu verstehen, sondern an und für sich als neuer Designprozess mit spezifischen Anforderungen zu betrachten. Der Betrieb von Kubernetes für Kunden, ggf. auch beim Kunden vor Ort, stellt somit keine Rahmenbedingung dar. Der letzte der fünf Punkte, Anforderung [AR11], wurde lediglich in den Interviews mit Unternehmen B und C eingebracht. Insgesamt konnten die Rahmenbedingungen, mit Hilfe der durchgeführten Workshops und Interviews, von zehn auf 16 Anforderungen erweitert werden. Zwei dieser sechs neuen Rahmenbedingungen, nämlich [AR13] und [AR15], stammen ausschließlich aus den Workshops.

Funktionale Anforderungen: Im Bereich der funktionalen Anforderungen wurden von den bereits bekannten 20 Punkten, 12 in den Interviews ebenfalls erwähnt. Besonders häufig genannt wurden die Anforderungen [AF12], die verwendeten Protokolle, [AF07], Kommunikation zwischen Pods und [AF05], die benötigten Betriebssysteme der Container. Die Themen [AF09], Erwarten einer Logging Lösung, [AF13], Verwendung eines Service Mesh sowie [AF14], integriertes User Management, wurden nur von jeweils einem Interviewpartner angesprochen.

Neben der Bestätigung bestehender Punkte wurden noch zehn neue funktionale Anforderungen identifiziert. Sieben dieser zehn Punkte wurden außerdem in den Workshops erwähnt. Drei der Punkte, welche nur in den Interviews erwähnt wurden, konnten nicht in die finale Liste der funktionalen Anforderungen aufgenommen werden. Zum Ersten wurde angeführt, dass alte Verschlüsselungsalgorithmen (TLS Versionen) in gewissen Konstellation nötig sind, was jedoch eher eine technische Konfiguration und keine Anforderung ist. Zum Zweiten wurde angemerkt, dass Kubernetes auf den Rechnern von Entwicklern laufen muss. Dies ist eher eine Vorgabe für Entwicklungstools, als für ein zentrales Kubernetes System. Zum Dritten wurde eine Funktionalität benötigt, welche es erlaubt Pods am Cluster horizontal zu skalieren. Diese Anforderung wurde verworfen, da dies eine Standardfunktionalität von Kubernetes ist, und somit nicht explizit gefordert werden muss. Aufgrund der durchgeführten Workshops und Interviews, konnten somit zehn funktionale Anforderungen identifiziert werden, von denen fünf in die bestehende Liste aufgenommen wurden. Von diesen fünf neuen Punkten wurden zwei ([AF24]/[AF25]) lediglich in Workshops eingebracht. Insgesamt wurden also 25 funktionale Anforderungen identifiziert, die an einen Kubernetes Cluster gestellt werden können.

Qualitätseigenschaften: Von den aus der Theorie bekannten zehn Qualitätseigenschaften wurden sieben in den Interviews ebenfalls erwähnt. Lediglich der Aufwand für Upgrades([AQ01]), die Erfahrung eines Unternehmens ([AQ07]) sowie die Performance einer Komponente ([AQ10]) wurden in keinem Interview angeführt. Es ist vor allem interessant, dass das Thema Performance von keinem Interviewteilnehmer aufgeworfen wurde, obwohl dies vermeintlich eines der wichtigsten Gütekriterien eines IT Systems darstellt. In den durchgeführten sechs Interviews wurden insgesamt 17 Qualitätseigenschaften identifiziert, zehn davon waren aus der Theorie noch nicht bekannt. Acht von diesen zehn Punkten wurden auch schon in den Workshops genannt. Nur die Unternehmensgröße ([AQ12]) und die Anzahl bereits qualifizierten Personals ([AQ11]) wurden ausschließlich in den Interviews eingebracht. Insgesamt konnte die Anzahl der Qualitätseigenschaften, durch die Workshops und Interviews, von zehn auf 24 erhöht werden. Zwei der neuen Punkte kamen ausschließlich aus den Interviews, fünf davon nur aus den Workshops.

In den bisherigen Abschnitten wurde erwähnt, dass die Workshops und Interviews sowohl beim Designprozess, als auch beim Anforderungsbogen neue Erkenntnisse brachten. Die Abbildungen 5.4 und 5.5 sowie die Tabelle 5.11 zeigen nochmals auf wie viele Anforderungen im Praxisteil zum theoretischen Modell hinzugefügt werden konnten. Des Weiteren ist ersichtlich woher die jeweiligen Informationen gekommen sind.

Quelle	Identifizierte Anforderung(en)
Workshop 1	AR12, AR16, AF22, AF23, AF24, AF25, AQ13-AQ24
Workshop 2	AR13, AR14, AR15, AF21
Unternehmen A	AQ11
Unternehmen B	AR11
Unternehmen E	AQ12

Tab. 5.11: Herkunft der neu identifizierten Anforderungen

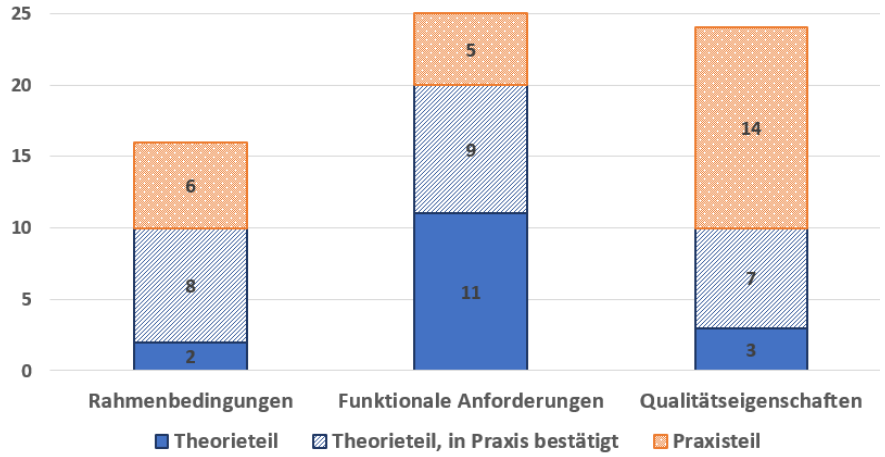


Abb. 5.4:
quantitative Darstellung aller identifizierten Anforderungen
Quelle: eigene Darstellung

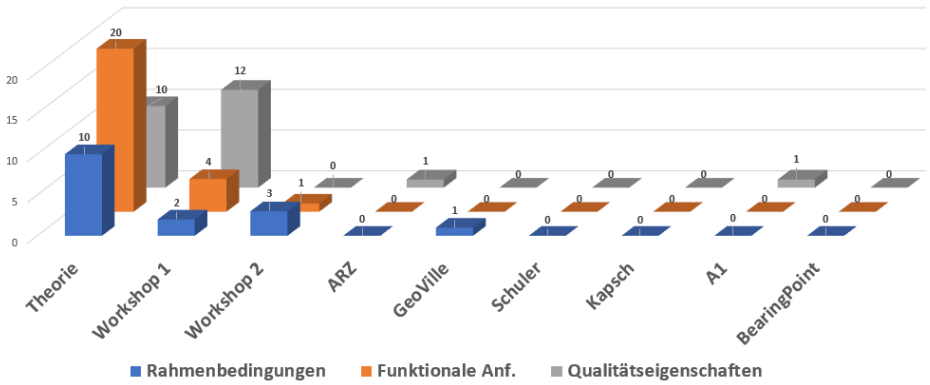


Abb. 5.5:
Anzahl neu identifizierter Anforderungen pro Quelle
Quelle: eigene Darstellung

5.4 Integration der Ergebnisse

Aus den Workshops kam die Erkenntnis, dass der letzte Schritt des Designprozesses eine Integration in die Organisation des Unternehmens sein sollte. Das heißt, dass das Kubernetes System in die vorhandene Prozesslandschaft integriert werden muss. Genau gesagt wurde die Integration in bestehende Supportstrukturen im ersten Workshop eingebracht. Es muss geregelt werden wer das System betreut und wer auf verschiedenen Ebenen technischen und administrativen Support zur Verfügung stellt. Salopp formuliert bedeutet dies, dass die entsprechenden Zuständigkeiten geregelt, verschriftlicht und in die Organisation eingebaut werden müssen. Dieser Schritt ist als Erinnerung für den durchführenden Innovationsmanager zu verstehen, dass das Design eines Kubernetes Systems nicht mit dessen technischer Installation beendet ist. Es bedarf auch einer vollständigen organisatorischen Integration ins Unternehmen, um das System bestmöglich nutzen und etwaige Reibungspunkte verhindern zu können. Dieser Punkt wurde nachträglich, nach der Durchführung der Workshops, in das theoretische Modell in Kapitel 4.1 aufgenommen. Damit sollte sichergestellt werden, dass der Leser in diesem Kapitel den finalen Prozess vorfindet und der Lesefluss nicht beeinträchtigt wird.

Die Ergebnisse der Workshops und Interviews, mit Bezug auf den Anforderungskatalog, wurden ebenfalls zur bestehenden Liste der Anforderungen hinzugefügt. Wie in Kapitel 4.1 auf Seite 47 bereits angemerkt hat jede Anforderung im Katalog einen Präfix und eine Nummer. Damit kann die Anforderung eindeutig identifiziert werden. Funktionale Anforderungen werden zusätzlich noch in [TECH] und [INNO] unterschieden. Damit wird signalisiert ob der durchführende Innovationsmanager technische Unterstützung für die Beantwortung dieser Frage benötigt oder nicht. Zusätzlich dazu wurden alle Anforderungen noch mit [T] oder [P] gekennzeichnet. Damit wird beschrieben, ob die Anforderungen in der theoretischen Recherche ([T]) oder bei der Durchführung der Interviews bzw. Workshops ([P]) identifiziert wurde. Die in der Praxis identifizierten Anforderungen wurden im Anforderungskatalog jeweils nach den theoretisch gefundenen eingefügt. Auf die Relevanz oder den Einfluss einer Anforderung hat die Reihenfolge bzw. deren Position im Katalog keine Auswirkung.

5.5 Praktische Anwendung des Modells

Um das System auf seine praktische Anwendbarkeit zu testen, wurde es zusammen mit einem Enterprise Architekten auf einen bestehenden Use-Case angewandt. Thematisch behandelte der Use-Case die Bereitstellung einer unternehmensinternen Kubernetes Infrastruktur auf der diverse Services betrieben werden sollten. Das Unternehmen hat den Mehrwert von Kubernetes im Bezug auf Flexibilität, Ressourcenauslastung und beschleunigte Entwicklung erkannt. Daher soll zunächst die Entwicklung interner Services mit Kubernetes als Plattform erfolgen. Zusätzlich dazu möchte das Unternehmen bestehende Tools und Services von Drittherstellern auf Kubernetes migrieren. Die dadurch entstandenen Anforderungen sind auf funktionaler Seite sehr breit gefächert. Es wurde aufgrund der Art und Menge bestehender Daten festgelegt, dass es sich um ein On-Premises System handeln soll. Eine detaillierte Auflistung der definierten Anforderungen ist an dieser Stelle nicht möglich, da es sich hierbei um unternehmensinterne Informationen handelt. Um die praktische Anwendung des Anforderungskatalogs dennoch demonstrieren zu können, wird im folgenden Kapitel ein Beispiel angeführt.

Die Anwendung des Modells wurde ähnlich einem Interview durchgeführt. Dabei wurde erst der Designprozess als Ganzes, anschließend jeder Punkt des Anforderungskatalogs besprochen. Der Interviewpartner hat auf alle Fragen, gemäß dem vorliegenden Use-Case geantwortet, wodurch der gesamte Katalog einmal ausgefüllt werden konnte. Bei einzelnen Anforderungen wurde noch explizit Feedback und Input gegeben. Als Interviewpartner wurde bewusst ein Enterprise Architekt gewählt, da er schon sehr lange im Unternehmen tätig ist.

Bedingt durch seine Stelle ist er auch stark mit vielen Personen und Abteilungen im Unternehmen vernetzt. Durch seine Funktion als Begleiter vieler Projekte hat er gute Einblicke in die Probleme und Anforderungen bei der Softwareentwicklung. Er ist einer der führenden technischen Experten im Unternehmen, ist jedoch auch für die Integration von Systemen auf architektonischer und organisatorischer Seite zuständig. Außerdem ist er auch im Bereich Container-Orchestration aktiv und betreut selbst die Initiative für den vorliegenden Use-Case.

Es wurde zum Designprozess auch hier nochmals explizit angemerkt, dass die Integration des Kubernetes Systems im Unternehmen essentiell ist. Damit ist nicht nur die technische Integration, sondern vor allem auch die Eingliederung in der Aufbau- und Ablauforganisation gemeint. Dies deckt sich mit dem Feedback aus den Workshops und wurde als letzter Schritt im Designprozess eingearbeitet. Zusätzlich wurde noch zu berücksichtigen gegeben, dass das neue System auch architektonisch in die technische Infrastruktur integriert werden muss. Das bedeutet, dass es einen konkreten Verwendungszweck geben muss, für welchen Kubernetes auch zukünftig zu verwenden ist. Es wurde außerdem noch folgendes Feedback zu Anhang A.18, der tatsächlichen Auswahl von Produkten, gegeben. Dieser Schritt sollte zwar sauber durchgeführt und dokumentiert werden, die Methode darf jedoch nicht zu strikt und bürokratisch gesehen werden. Dies würde ansonsten den Entscheidungsprozess eher hemmen als fördern. Es muss ein Mittelmaß aus strukturierter und recherchierter Entscheidung, sowie Erfahrung und Bauchgefühl gefunden werden.

Der Anforderungskatalog wurde insgesamt als gelungen angesehen. Es gab an drei Stellen noch ergänzendes Feedback, welches zwei kosmetische Änderungen zur Folge hatte. Bei [AR02] wurde angemerkt, dass die ursprüngliche Bezeichnung "Availability" irreführend sein kann. Diese wurde daher auf "Lokation/Availability Zone" geändert. Der Begriff "Availability Zone" spielt hierbei auf das Zonenkonzept an, welches von vielen Public-Clouds verwendet wird. Bei [AR04], der Voraussetzung von Preisnachlässen für Test- und Entwicklungssysteme wurde zusätzlich angemerkt, dass dies nicht nur auf On-Premises Installationen anwendbar ist. Es gibt mittlerweile auch Public-Cloud-Provider die Preisnachlässe für Entwicklungs- und Testsysteme gewähren. Dies wurde als Zusatzinformation gewertet und nicht aktiv in die Anforderungsformulierung aufgenommen. In [AQ11], der Berücksichtigung bereits qualifizierten Personals, hingegen wurde noch zusätzlich folgendes aufgenommen: Mitarbeiter können auch als qualifiziert gelten, wenn sie eine wesentlich flachere Lernkurve für ein neues Produkt aufweisen, wenn dieses auf bekannten Technologien aufbaut. So können beispielsweise Administratoren mit Public-Cloud Erfahrung meist schneller in ein anderes Cloud-Produkt einsteigen, als in ein On-Premises Produkt. Diese Anmerkung wurde in die Anforderungsformulierung aufgenommen. Das erarbeitete Modell aus Designprozess und Anforderungskatalog wurde vom Interviewpartner als gelungen und praktikabel eingeschätzt. Der Anforderungskatalog im Speziellen müsse jedoch für eine praktische Anwendbarkeit in ein anderes digitales Format gebracht werden. Die Fragen an sich bieten laut Interviewpartner eine solide Basis und müssen, falls in der Anwendung noch Lücken auftauchen, dahingehend erweitert werden. Der folgende Abschnitt zeigt auf wie der Anforderungskatalog anhand eines realen Beispiels angewendet werden könnte.

5.6 Beispiel: Anforderungen an die CERN Kubernetes Umgebung

Dieses Kapitel verdeutlicht die Verwendung des erarbeiteten Anforderungsbogens. Es wird pro Frage kurz erklärt warum die Antwort entsprechend gewählt wurde, um dem Leser die Zusammenhänge zu verdeutlichen. Als Beispiel wird die Europäische Organisation für Kernforschung (CERN) herangezogen, da sie einerseits einige sehr spezifische Anforderungen stellt, und andererseits einem breiten Publikum bekannt und somit eher anschaulich ist. Bei CERN verarbeiten derzeit ca. 3300 Benutzer mit einer Kubernetes Plattform von über 320.000 CPU Kernen einen Datenbestand von mehr als 500 Petabytes an Daten.

Dabei handelt es sich lediglich um die On-Premises Systeme, bzw. Daten von CERN. Das erweiterte dezentrale System umfasst über 200 Standorte mit mehr als 700.000 CPU Cores. Bei CERN werden über 80% der Ressourcen von ML und sogenannten Batch Jobs verbraucht. Ein Batch Job ist ein Programm welcher einmalig eine große Menge Daten verarbeitet, beispielsweise um etwas zu simulieren oder Kennzahlen zu berechnen. Kubernetes wurde bei CERN 2015 eingeführt, ist seit 2016 produktiv in Verwendung und wird seit 2018 als hybrides System betrieben. Durch hybride Cluster, sprich eine Kombination aus On-Premises und Cloud Ressourcen, können die häufig auftretenden Spitzenbelastungen besser bedient werden. Bei CERN hat man sich, bzgl. der Kubernetes Distribution, für Upstream Kubernetes entschieden. Außerdem werden die Projekte bzw. Produkte HELM[211] (Paketmanager), Prometheus[212] (Monitoring), coreDNS[213] (DNS), Kibana[214] (Logging/Dashboard), Ceph[215] (Storage), CernVM-FS[216] (Storage), Traefik[217] (Ingress) sowie NGINX[218] (Ingress) verwendet. Die Beantwortung der Fragen im folgenden Anforderungskatalog würde den Technologiestack, welchen CERN heute verwendet, ergeben.[219]

[INNO][AR01][T] Lizenz / Open-Source: Ist die Verwendung von Open-Source Software möglich?

⊗ JA | ○ NEIN

Welche der folgenden Open-Source Lizenzen ist im Unternehmen zulässig?
Erklärung: Die gewählten Produkte verwenden eine dieser Lizenzen.

⊠ GPL | ⊠ LGPL | ⊠ Apache 2.0 | □ BSD | □ CPL | □ BSD | □ MPL | □ EUPL | ⊠ MIT

[INNO][AR02][T] Lokation/Availability Zone: Ist es nötig, dass der Cluster sich in einem bestimmten Land, bzw. einer bestimmten Region befindet?
Erklärung: Die Hauptrechenzentren befinden sich in Genf, dies ist jedoch historisch bedingt und keine Vorgabe.

○ JA | ⊗ $NEIN$

[INNO][AR03][T] Support: Wird Enterprise Support für die Kubernetes Distribution benötigt?
Erklärung: Es werden keine Produkte mit Enterprise Support verwendet.

○ JA | ⊗ $NEIN$

Wird Enterprise Support für einzelne Kubernetes Komponenten benötigt?
Erklärung: Es werden keine Produkte mit Enterprise Support verwendet.

○ JA | ⊗ $NEIN$

[211] weitere Informationen unter: https://github.com/helm/helm
[212] weitere Informationen unter: https://github.com/prometheus/prometheus
[213] weitere Informationen unter: https://github.com/coredns/coredns
[214] weitere Informationen unter: https://github.com/elastic/kibana
[215] weitere Informationen unter: https://github.com/ceph/ceph
[216] weitere Informationen unter: https://cernvm.cern.ch/fs/
[217] weitere Informationen unter: https://github.com/traefik/traefik
[218] weitere Informationen unter: weitere Informationen unter: https://github.com/kubernetes/ingress-nginx
[219] The Linux Foundation (2020d), Onlinequelle [06.12.2020]; Rocha (2018), Onlinequelle [06.12.2020]; Bell (2019), Onlinequelle [06.12.2020]; Williams (2020b), Onlinequelle [06.12.2020].

[INNO][AR04][T] Test Cluster: Ist es eine Voraussetzung, dass kostenfreie oder vergünstigte Lösungen für Entwicklungs- bzw. Testumgebungen verfügbar sind?
Erklärung: Die Voraussetzung ist durch die Verwendung von freier Software implizit erfüllt, für ein Unternehmen dieser Größe jedoch grundsätzlich gegeben.

$\otimes$ JA | ○ NEIN

[INNO][AR05][T] Grafische Benutzerschnittstelle - Cluster: Ist eine grafische Benutzeroberfläche für Kubernetes notwendig?
Erklärung: Es wurde nirgends eine solche Anforderung erwähnt.

○ JA | $\otimes$ $NEIN$

[INNO][AR06][T] Grafische Benutzerschnittstelle - Komponenten: Ist es nötig, dass die einzelnen Komponenten jeweils über eine grafische Benutzerschnittstelle verfügen?
Erklärung: Es wurde nirgends eine solche Anforderung erwähnt.

○ JA | $\otimes$ $NEIN$

[INNO][AR07][T] GDPR/DSGVO: Sind die zu verarbeitenden Daten von der DSGVO/GDPR betroffen?
Erklärung: Die Daten sind frei zugänglich.

○ JA | $\otimes$ $NEIN$

[INNO][AR08] Linux Distribution: Wird eine bestimmte LinuxDistribution benötigt?
Erklärung: Es wurde nirgends eine solche Anforderung erwähnt.

□ Red Hat Enterprise Linux | □ Ubuntu | □ Suse LinuxEnterprise | □ openSUSE

□ Fedora CoreOS | □ Red Hat CoreOS | □ RancherOS | □ Photon OS | ⊠ keine

[INNO][AR09][T] Strategische Allianzen: Gibt es eine strategische Allianz oder Partnerschaft, welche Einfluss auf die Wahl einer Kubernetes Distribution oder Komponente haben könnte?
Erklärung: Es wurde beispielsweise explizit erwähnt, dass sowohl Google als auch Microsoft als Public-Cloud-Provider verwendet werden.

$\otimes$ JA | ○ NEIN

[INNO][AR10][T] Public-Cloud Strategie: Ist die Verwendung von Public-Cloud Angeboten möglich oder sogar erwünscht?
Erklärung: Public-Cloud Angebote werden verwendet.

$\otimes JA$ | ○ NEIN

Müssen bestehende On-Premises Systeme verwendet werden können?
Erklärung: Die bestehenden Data Center müssen verwendet werden.

$\otimes JA$ | ○ NEIN

[INNO][AR11][P] Verfügbarkeit der Daten: Gibt es Vorgaben oder Umstände welche die Wahl der Lokation des Clusters, basierend auf den zu verarbeitenden Daten, einschränken? Falls JA, welche?
Erklärung: Der Cluster sollte in Genf liegen, da dort auch die Messdaten von CERN liegen. Dabei kann es sich unter Umständen um enorme Datenmengen handeln, deren Übertragung nicht sinnvoll wäre.

$\otimes JA$ | ○ NEIN

[INNO][AR12][P] Politische Vorgaben/Herstellerstandort: Gibt es Vorgaben bzgl. des Herstellerstandortes, oder politischer Natur, die berücksichtigt werden müssen? Falls JA, welche?
Erklärung: Es gibt keine konkreten Vorgaben, obwohl grundsätzlich Open-Source Software bevorzugt wird.

○ JA | $\otimes NEIN$

[INNO][AR13][P] ausgelagerter Betrieb und Support: Ist es nötig den Betrieb inklusive Support des Clusters auszulagern?
Erklärung: Es gibt ein eigenes Team für den Betrieb der Infrastruktur, der Betrieb/Support muss nicht ausgelagert werden.

○ JA | $\otimes NEIN$

[TECH][AR14][P] Single-Node-Kubernetes: Wird eine "Single-Node-Kubernetes" Distribution benötigt?
Erklärung: Es muss ein enormer Cluster für große Berechnungen entworfen werden.

○ JA | $\otimes NEIN$

[INNO][AR15][P] Long-Term-Support: Wird LTS benötigt?
Erklärung: Auf Grund der großen Anzahl an Rechnern wird TLS Support bevorzugt, um nicht in zeitliche Probleme beim Upgrade der Hosts zu gelangen.

$\otimes JA$ | ○ NEIN

[INNO][AR16][P] Vendor Lock-In: Muss darauf geachtet werden, dass bei der Produktauswahl explizit kein Vendor Lock-In entsteht?
Erklärung: Auf Grund der Größenordnung des Clusters muss auf einen Vendor Lock-In aufgepasst werden. Ein solch großes und kritisches System darf nicht von einem Hersteller abhängig sein. Zum Beispiel könnte eine Preiserhöhung der Lizenzen gravierende Folgen haben.

⊗ JA | ○ NEIN

[INNO][AF01] **Redundanter Speicher**: Müssen die Daten redundant gespeichert werden?
Erklärung: Es wurde erwähnt, dass die Daten gesichert und archiviert werden.

⊗ $Redundanzbentigt$ | ○ Keine Redundanz benötigt

[INNO][AF02] **Verschlüsselter Speicher**: Müssen persistierte Daten verschlüsselt werden?
Erklärung: Die Daten sind nicht sensibel.

○ JA | ⊗ $NEIN$

[INNO][AF03] **Storage Backups**: Werden Storage Backups benötigt? **Erklärung:** Es wurde erwähnt, dass die Daten gesichert und archiviert werden.

⊗ JA | ○ NEIN

[INNO][AF04] **Grafikkarten Unterstützung**: Werden am Cluster AI oder ML Anwendungen betrieben?
Erklärung: Es gibt viele ML Anwendungen.

⊗ JA | ○ NEIN

[INNO][AF05] **Betriebssystem**: Container basierend auf welcher Art von Betriebssystem sollen unterstützt werden?
Erklärung: Bei Systemen solcher Größe kann davon ausgegangen werden, dass auch Windows Programme betrieben werden sollen.

⊠ Windows | ⊠ Linux

[INNO][AF06] **Costing**: Wird eine Funktionalität benötigt, die es erlaubt verbrauchte Ressourcen zu analysieren und monetär darzustellen?
Erklärung: Anhand der Größe und Benutzeranzahl kann davon ausgegangen werden.

⊗ JA | ○ NEIN

[INNO][AF07] **Inter-Pod Kommunikation**: Ist es nötig die Kommunikation zwischen bestimmten Pods zu unterbinden?
Erklärung: Da die Daten nicht sensibel sind wird nicht davon ausgegangen.

○ JA | ⊗ $NEIN$

[INNO][AF08] **Metrik-System**: Wird ein neues, cluster-internes System zur Sammlung von Metriken, sowie zur Verwaltung von Alarmen, benötigt?
Erklärung: Prometheus wird verwendet.

⊗ JA | ○ NEIN

[INNO][AF09] **Logging-System**: Wird ein neues, cluster-internes System zur Sammlung und Auswertung von Log Daten benötigt?
Erklärung: Kibana wird verwendet.

⊗ JA | ○ NEIN

[INNO][AF10] **Tracing-System**: Wird ein neues, cluster-internes System zum Tracing benötigt? **Erklärung:** Es wurde nirgends eine solche Anforderung erwähnt.

○ JA | ⊗ $NEIN$

[INNO][AF11] **Multiple Netzwerkschnittstellen**: Werden für gewisse Container im Cluster mehrere Netzwerkschnittstellen benötigt?
Erklärung: Die Verwendung von ML und Batch Anwendungen lässt diese Annahme zu.

⊗ JA | ○ NEIN

[TECH][AF12] **Service-Protokolle**: Über welche Protokolle wird von außen auf Services am Cluster zugegriffen?
Erklärung: Systeme dieser Größe lassen diese Annahme zu.

⊠ HTTP(S) | ⊠ HTTP/2 / gRPC | ⊠ TCP | ⊠ UDP

[TECH][AF13] **Service Mesh**: Wird ein Service Mesh benötigt?
Erklärung: Es wurde nirgends eine solche Anforderung erwähnt.

○ JA | ⊠ NEIN

[TECH][AF14] **User Management**: Wird eine Verknüpfung des zentralen User Management Systems mit Kubernetes benötigt?
Erklärung: Die Verwendung eines solchen Systems wurde erwähnt.

⊗ JA | ○ NEIN

[TECH][AF15] **Speicher**: Wird persistenter Speicher benötigt, oder reicht flüchtiger, sogenannter?
Erklärung: Die Daten müssen gespeichert und archiviert werden.

$\otimes\ persistenterSpeicherwirdbentigt$ | ○ persistenter Speicher wird nicht benötigt

[TECH][AF16] **Object Storage**: Wird *Object Storage* benötigt?
Erklärung: Es wurde nirgends eine solche Anforderung erwähnt.

○ JA | $\otimes\ NEIN$

[TECH][AF17] **File Storage**: Wird *File Storage* benötigt?
Erklärung: Die Natur der betriebenen Anwendungen lässt diese Annahme zu.

$\otimes\ JA$ | ○ NEIN

[TECH][AF18] **File Storage Zugriff**: Ist es nötig, dass sich mehrere Pods denselben File-Storage teilen?
Erklärung: Die Natur der betriebenen Anwendungen lässt diese Annahme zu.

$\otimes\ JA$ | ○ NEIN

[TECH][AF19] **Block Storage**: Wird *Block Storage* benötigt?
Erklärung: Die Natur der betriebenen Anwendungen lässt diese Annahme zu.

$\otimes\ JA$ | ○ NEIN

[TECH][AF20] **Cluster Scaling**: Wird eine Form der mehr oder weniger automatisierten Cluster Skalierung benötigt?
Erklärung: Es sind skalierende Cluster im Einsatz.

$\otimes\ JA$ | ○ NEIN

[TECH][AF21][P] verschlüsselte Kommunikation: Ist es nötig die Kommunikation zwischen Pods zu verschlüsseln?
Erklärung: Da es sich bei den Daten nur um Messdaten handelt muss nicht verschlüsselt werden.

○ JA | $\otimes\ NEIN$

[TECH][AF22][P] Backup/Restore: Ist es nötig, dass das Produkt bzw. die Kubernetes Distribution ein Backup/Restore System out-of-the-box mitliefert?
Erklärung: Es können die Upstream Kubernetes Standardmittel verwendet werden, das Team hat die dafür nötige Erfahrung und Expertise.

○ JA | $\otimes\ NEIN$

[TECH][AF23][P] Integration in bestehende Infrastruktur: Gibt es bestehende Infrastruktur in welche sich der neue Cluster bzw. das neue Produkt integrieren lassen muss? Falls JA, welche?
Erklärung: Upstream Kubernetes wurde in die bestehende OpenStack und DNS Umgebung integriert.

$\otimes\ JA$ | ○ NEIN

[INNO][AF24][P] Sprache der Dokumentation und Benutzerschnittstelle: Müssen andere Sprachen, außer Englisch, unterstützt werden? Falls JA, welche?
Erklärung: CERN ist eine internationale Forschungseinrichtung, Englisch wird vorausgesetzt.

○ JA | ⊗ $NEIN$

[TECH][AF25][P] WAF Funktionalität für Ingress: Werden zusätzliche Sicherheitsfunktionen am Ingresscontroller benötigt?
Erklärung: Die Cluster befinden sich in einem internen Netz, daher werden keine größeren Gefahren erwartet.

○ JA | ⊗ $NEIN$

6 FAZIT

Dieses Kapitel bildet den Schluss der vorliegenden Arbeit. Es wird anhand eines Beispiels zusammengefasst welche groben Probleme mit Kubernetes gelöst werden können. Außerdem wird aufgezeigt wo sich das entwickelte Modell in der Innovationsmanagement Theorie ansiedeln lässt. Es werden einige Querverweise zu bekannten Modellen vorgestellt, die den Einfluss von Kubernetes auf ein Unternehmen widerspiegeln können. Abschließend wird das erstellte Modell kompakt dargestellt, mögliche Erweiterungs- und Einsatzmöglichkeiten aufgezeigt und die Forschungsfragen beantwortet.

Man kann einige der Vorteile von Kubernetes sehr einfach durch ein anschauliches Gedankenexperiment vermitteln. Man stellt sich ein junges Start-Up Unternehmen vor, welches die Art Reiseziele zu finden innoviert hat. Hierfür wurde eine Handyapp entwickelt, welche einem Benutzer personalisierte Reiseziele, zugeschnitten auf dessen Vorlieben, präsentiert. Die Berechnung der Ziele erfolgt mit einem brandneuen Machine-Learning Algorithmus. Das Algorithmus-Programm läuft auf einem Server im Keller des Unternehmens. Alle Benutzer verwenden diesen einen Server für ihre Reisezielsuchen.
Nach kurzer Zeit wird klar, dass die Benutzer ihre Reisen auch in der App buchen und bezahlen wollen. Auch eine Bewertungsfunktionalität, sowie Social Media Anbindung werden vom Markt gefordert. Diese neuen Funktionalitäten werden vom Start-Up entwickelt. Dafür werden neue Programme und Technologien verwendet. Dies kostet einerseits Zeit, Ressourcen und Geld und andererseits müssen auch diese Programme betrieben werden, was den Server zusätzlich auslastet. Davon abgesehen verlangen die neuen Programme eventuell andere Plattformen oder Betriebssysteme.
Da die neuen Funktionalitäten sehr gut am Markt ankommen steigt die Nachfrage. Durch den Anstieg der Nachfrage steigt jedoch auch die Last am Server, der alle Benutzer bedienen muss. Ab einer bestimmten kritischen Anzahl von Benutzern wird es passieren, dass der Server keine zusätzlichen Anfragen mehr verarbeiten kann. Um diesem Szenario entgegen zu wirken werden mehrere Server hinzugefügt und die eingehenden Benutzer gleichmäßig auf alle Server verteilt. Man spricht hierbei von Lastverteilung, was wiederum Ressourcen, Zeit und Know-How benötigt.
Durch die steigende Popularität der App wird beschlossen, dass auf den europäischen oder internationalen Markt expandiert werden soll. Darum errichtet man Datacenter und Standorte in mehreren Ländern. Auf Grund diverser Faktoren kommt es häufig vor, dass sich die technischen Infrastrukturen in den jeweiligen Standorten unterscheiden. Dies wirkt sich negativ auf den Betrieb sowie die Möglichkeiten zur Weiterentwicklung der Software aus.

Jede der genannten Veränderungen kostet jeweils Zeit, Ressourcen und im Endeffekt Geld, da die Infrastruktur und Plattform jedes mal verändert wird. Durch die Verwendung von Kubernetes können alle diese Herausforderungen sehr elegant, schnell und meist auch vergleichsweise günstig gelöst werden. Kubernetes erlaubt es, durch die Verwendung der Container Technologie unterschiedlichste Programme, basierend auf diversen Technologien, auf derselben Plattform zu betreiben. So gibt es beispielsweise fertig vorbereitete Container Images für diverse Anwendungen, welche sofort am Kubernetes Cluster gestartet und verwendet werden können. Auch das Problem der Skalierung wird durch Kubernetes sehr einfach gelöst. Es müssen lediglich neue Rechner in den Cluster eingebunden werden. Der Rest wird von Kubernetes intern erledigt. Dies ist weniger arbeitsintensiv, als sich selbst um die Lastverteilung auf manuell betriebenen Rechnern zu kümmern. Und was den Betrieb der Software in mehreren Datacentern betrifft, so hilft Kubernetes auch hier.

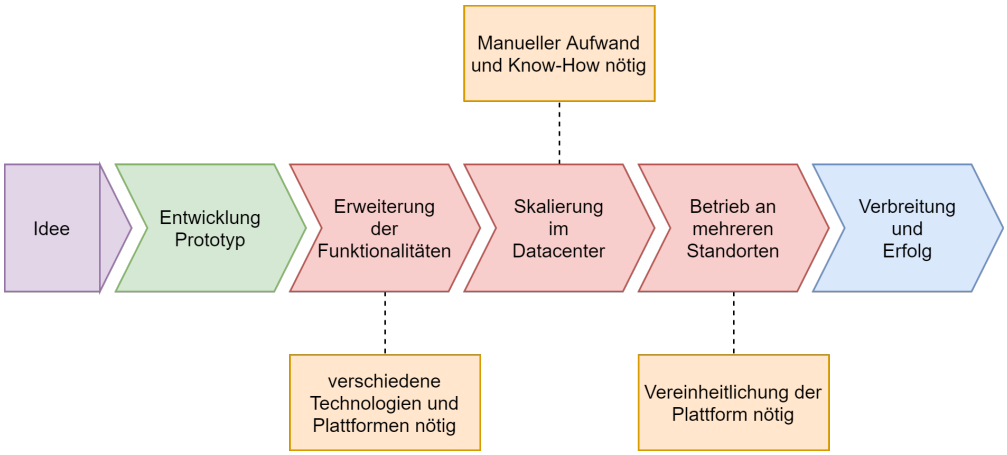


Abb. 6.1: **Ablauf des Gedankenexperiments**
Quelle: eigene Darstellung

Aus Sicht der Software ist es egal auf welcher Infrastruktur Kubernetes läuft, die Machine-Learning Anwendung beispielsweise würde immer gleich funktionieren. Es ist sogar möglich dieselbe Anwendung entweder auf Kubernetes On-Premises oder in einer Public-Cloud zu betreiben, was sogar noch mehr Möglichkeiten eröffnet.

Wie in Kapitel 2.2.2 bereits ausführlich beschrieben, ist Kubernetes die meist verbreitetste Technologie zur Verwaltung großer Container Systeme. Jedes Kubernetes System ist in seinem grundsätzlichen Aufbau identisch und besteht aus denselben Basiskomponenten. Der grundlegende Aufbau von Kubernetes wurde in Kapitel 2.4.2 bereits erklärt. Am Markt gibt es bereits verschiedenste Hersteller von Kubernetes Distributionen, die im Kern prinzipiell gleich oder ähnlich funktionieren, sich jedoch in ihrer Architektur, Umsetzung und ihren Möglichkeiten unterscheiden. Jedes Kubernetes System besteht darüber hinaus aus weiteren Komponenten, für welche es jedoch jeweils mehrere Produkte zur Auswahl gibt. Diese Komponenten wurden in Kapitel 3 ausführlich beschrieben. Man muss sich also im Endeffekt entscheiden, wie das Design des Clusters aussehen soll, also aus welcher Kombination von Kubernetes-Distribution und Produkten man sein System zusammensetzen will.

Es hat sich durch Beobachtungen aus der Praxis, sowie durch Feedback der in Kapitel 5.3 beschriebenen Interviews, gezeigt, dass die Entscheidung über das Design nicht einfach ist und enormen Einfluss hat. Um die beste Wahl aus allen möglichen Optionen treffen zu können, sollte eine Methode gewählt werden die strukturiert, transparent und nachvollziehbar ist. Ein Modell für eine solche Methode wurde in dieser Arbeit entwickelt. Es basiert auf einem Prozess der grob vorgibt in welchen Schritten das Design des Clusters ablaufen soll. Als Unterstützung für die Auswahl von passenden Produkten und architektonischen Entscheidungen liefert das Modell einen Katalog zur Spezifikation von Anforderungen.

In dieser Arbeit wird davon ausgegangen, dass Kubernetes als Plattform für einen bestimmten Anwendungsfall verwendet wird. Im einfachsten Fall liegt eine Idee vor, in Form eines fertigen Konzepts und basierend auf einem Computerprogramm, für welche Kubernetes als Plattform verwendet werden soll. In diesem Fall würde das entwickelte Modell sich im gesamten Innovationsprozess, siehe Abbildung 4.2, in den Punkten "Umsetzung" und "Markteinführung" eingliedern. Bei der Umsetzung muss bereits bekannt sein auf welcher Plattform die Entwicklung und der weitere Betrieb der fertigen Anwendung stattfinden soll. Daher sollte das Design des Kubernetes Systems bestenfalls schon vor oder während der Entwicklung der Applikation entschieden werden.

In der Phase der Markteinführung sollte das System bereits installiert und verfügbar sein, um alle Vorteile ausnützen und eventuelle zusätzliche Kosten vermeiden zu können. Außerdem erlaubt eine sauber entworfene Kubernetes Umgebung die Ausbreitung der Applikation am Markt technisch zu unterstützen. Schnelles globales Wachstum wäre damit beispielsweise deutlich einfacher und kosteneffizienter zu bewältigen als mit anderen Plattformen. Speziell das Thema kosten- und ressourceneffiziente Skalierung, in jede mögliche Richtung, spielt also bei der Markteinführung eine große Rolle. Man möchte nur jene Ressourcen verbrauchen die auch tatsächlich benötigt werden, gleichzeitig aber niemals die Stabilität des Service gefährden.

Der Designprozess für die Kubernetes Umgebung, ersichtlich in Abbildung 6.2, besteht aus sieben Schritten. In drei dieser Schritte wird der entwickelte Anforderungskatalog befüllt, welcher als Entscheidungsbasis für das konkrete Design herangezogen wird. Bis auf einen Schritt ist der gesamte Prozess vom Innovationsmanagement durchzuführen oder zumindest zu begleiten. Wer als Prozessverantwortlicher gilt ist im jeweiligen Unternehmen selbst zu entscheiden. Diese Entscheidung hat für den Aufbau des Modells keine Bedeutung, kann jedoch bei der Umsetzung durchaus relevant sein. Ein guter Prozessverantwortlicher stellt sicher, dass der Prozess eingehalten und sauber umgesetzt wird.

Schritt 1 - Kubernetes Evaluierung: Nachdem das Ereignis eintritt, dass eine Plattform für eine Anwendung gesucht wird, wird evaluiert ob Kubernetes dafür in Frage kommt. Die Punkte aus Kapitel 4.2 sollen dabei helfen diese Entscheidung wohl überlegt zu treffen. Das Ergebnis dieser Evaluierung muss eine schriftliche Entscheidung für oder gegen Kubernetes sein. Falls man sich gegen Kubernetes entscheidet, muss eine alternative Plattform gefunden werden.

Schritt 2 - Rahmenbedingungen: Wenn man sich für die Verwendung von Kubernetes entschieden hat muss abgesteckt werden in welchem Rahmen man sich beim Design des Systems bewegen darf. Bei dieser Definition sollen die Fragen aus dem Bereich Rahmenbedingungen des Anforderungskatalogs, siehe Anhang A.15, unterstützen. Dieser Schritt wird vom zuständigen Innovationsmanager durchgeführt, baut jedoch natürlich auf das Wissen aller beteiligten Personen auf.

Schritt 3 - Funktionale Anforderungen [INNO]: Sobald geklärt ist welche Rahmenbedingungen es beim Design gibt muss erhoben werden welche funktionalen Anforderungen an das Kubernetes System gestellt werden. Dabei handelt es sich großteils um mehr oder weniger technische Fragen. In diesem Schritt geht es darum jene Fragen zu klären, die der Innovationsmanager selbst noch beantworten könnten sollte. Als Unterstützung sollen die Ausarbeitungen in Anhang A.16 dienen. Das Label [INNO] kennzeichnet alle Fragen die in diesem Schritt behandelt werden sollen.

Schritt 4 - Funktionale Anforderungen [TECH]: Dieser Schritt behandelt jene funktionalen Anforderungen die ein größeres technisches Wissen voraussetzen als es vom durchführenden Innovationsmanager erwartet werden kann. Um die Fragen dieses Abschnitts behandeln zu können bedarf es außerdem eines tiefen technischen Verständnisses bzgl. der Umsetzung der geplanten Idee bzw. Applikation. Daher empfiehlt es sich in diesem Schritt einen technischen Experten, welcher mit dem Konzept der vorliegenden Idee vertraut ist, zu konsultieren. Gemeinsam mit diesem Experten sollen die verbleibenden funktionalen Anforderungen definiert werden. Die mit [TECH] gekennzeichneten Punkte in Anhang A.16 sollen dabei als Unterstützung dienen.

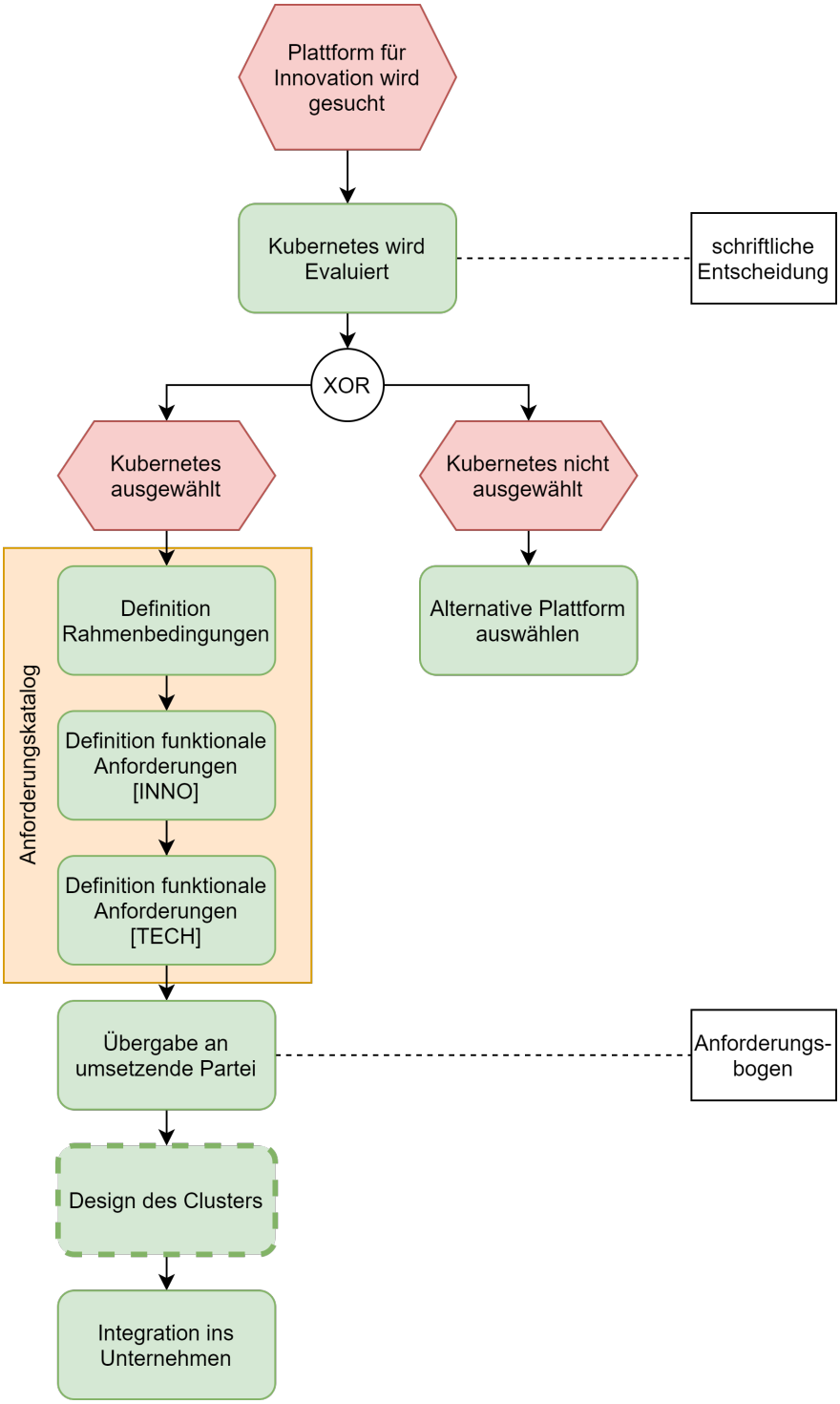


Abb. 6.2:
Designablauf eines Kubernetes Clusters
Quelle: Eigene Darstellung

Schritt 5 - Übergabe an umsetzende Partei: Die konkrete Auswahl einer Kubernetes Distribution, sowie der einzelnen Produkte für die jeweiligen Komponenten, setzt tiefes technisches Verständnis voraus. Dieser Schritt wird daher üblicherweise von jener Partei bzw. dem Team umgesetzt, welches das Kubernetes System später auch betreut. In diesem Schritt wird also die Sammlung aller erhobenen Anforderungen an das Team übergeben, das sowohl für die Installation als auch den Betrieb des Systems zuständig sein wird.

Schritt 6 - Design des Clusters: In diesem Schritt ist das Innovationsmanagement maximal unterstützend integriert. Hierbei wird das Kubernetes System vom technisch verantwortlichen Team entworfen und installiert. Es wird über den architektonischen Aufbau des Clusters entschieden, sowie über die verwendete Kubernetes Distribution. Des Weiteren werden Produkte für jede benötigte Komponente des Clusters ausgewählt. Die ausgearbeiteten und in Anhang A.17 aufgelisteten Qualitätseigenschaften sollen dabei helfen das jeweils beste Produkt zu finden. Am Ende dieses Schrittes ist das Kubernetes System entworfen und fertig installiert. Für die Auswahl der konkreten Produkte kann Kapitel A.18 als Hilfestellung dienen.

Schritt 7 - Integration ins Unternehmen: Nachdem das Kubernetes System installiert wurde muss es noch vollständig in das Unternehmen integriert werden. In diesem Schritt werden beispielsweise alle Zuständigkeiten betreffend des Clusters definiert. Etwaige Supportregelungen werden festgelegt und Rollen werden vergeben. Außerdem muss das System in alle nötigen Prozesse organisatorischer wie technischer Natur eingebunden werden. Dazu gehören zum Beispiel alle Automatisierungen im Entwicklungsprozess, deren Ziel oder Basis die Applikationsplattform ist. Das System muss in etwaige Drittsysteme wir Überwachungs- oder Verwaltungstools integriert werden. Idealerweise sollte man auch die strategische Bedeutung des neuen Kubernetes Systems kommunizieren. Man könnte also überlegen, ob bestehende oder neue Lösungen ebenfalls diese Plattform nutzen sollen und dies entsprechend kommunizieren, sowie in Strategien und Prozessen verankern.

Die vorliegende Arbeit schlägt nicht nur ein Modell für das Design von Kubernetes Systemen vor. Sie soll auch speziell einem Innovationsmanager dabei helfen in das Thema einzutauchen. In Kapitel 2 wird dem Leser näher gebracht welche Potenziale die Verwendung von Containern und Kubernetes birgt. Es wird behandelt wie sich die Technologien historisch entwickelt haben, aber auch welche nennenswerten Entwicklungen am Markt es in den letzten Jahren gab. Außerdem wird auf technischer Seite behandelt wie Container und Kubernetes aufgebaut sind und funktionieren. Abschließend erhält man einen Einblick in einige prominente Beispiele von Unternehmen die Kubernetes bereits erfolgreich verwenden. Der Leser erhält somit einen breiten Überblick über den Themenbereich, sowie das nötige technische Wissen um die weiteren Schritte verstehen zu können.

Vertieft wird in Kapitel 3 auf die einzelnen Komponenten von Kubernetes eingegangen. Es wird ausgeführt welchen Zweck diese Teile eines Clusters erfüllen und welche Eigenschaften sie beschreiben. Außerdem werden jeweils einige bekannte Vertreter jeder Komponente aufgelistet. Dieses Wissen hilft dem Leser dabei zu verstehen worin die Herausforderung beim Design eines Kubernetes Clusters liegt. Natürlich wird auch das technische Verständnis weiter vertieft und weitere Möglichkeiten aufgezeigt die Kubernetes technisch bietet.

In Kapitel 4 wird detailliert erläutert wie der Designprozess eines Clusters aufgebaut ist. Es wird das Modell dazu beschrieben und erklärt wie der Anforderungsbogen aufgebaut ist. Die drei Hauptgruppen des Anforderungsbogens werden erklärt und mit Beispielen dargestellt. Die Validierung und Ergänzung des Modells, sowie die dafür angewendeten wissenschaftlichen Methoden, werden in Kapitel 5 detailliert beschrieben.

Das entwickelte Modell soll somit einem Innovationsmanager dabei helfen die richtige Entscheidung bei der Wahl der Plattform für seine Innovation zu treffen. Der theoretische Teil der Arbeit soll als Einleitung in das Thema dienen und das nötige Basiswissen vermitteln.

Das erarbeitete Modell, bestehend aus Designprozess und Anforderungskatalog, wurde im praktischen Teil dieser Arbeit wissenschaftlich validiert und ergänzt. Es wird jedoch keinesfalls Anspruch auf Vollständigkeit erhoben, da das Modell mit Sicherheit noch Verbesserungspotenziale bietet. Ein Beispiel dafür sind die Schritte sechs und sieben im Prozess, welche sich mit der Produktauswahl und Integration im Unternehmen beschäftigen.

In Anhang A.18 befindet sich ein Vorschlag dafür, wie Produkte oder eine Kubernetes Distribution, basierend auf dem Anforderungskatalog, ausgewählt werden kann. Im Endeffekt geht es darum alle möglichen Optionen zu sammeln und die verfügbaren Informationen entsprechend aufzubereiten. Dann kann anhand der Rahmenbedingungen und funktionalen Anforderungen eine Vorselektion an passenden Produkten oder Distributionen erfolgen. Diese verbleibenden Optionen werden basierend auf den definierten Qualitätseigenschaften verglichen und gereiht. Somit könnte strukturiert und transparent entschieden werden welche Produkte gemäß welcher Anforderungen ausgewählt werden sollen. Diese Vorgehensweise ist jedoch nur ein erster Vorschlag und müsste ebenfalls verfeinert und in der Praxis validiert werden.

Es stellt sich natürlich auch die Frage wie oft ein Unternehmen die Designentscheidung für ein Kubernetes System treffen muss. Ist dies einmalig oder selten der Fall, wie im Gedankenexperiment am Anfang des Modells, so kann die analoge Version, wie sie hier vorliegt, verwendet werden. Steht diese Entscheidung jedoch öfters an, zum Beispiel in Beratungsunternehmen oder großen Konzernen, so würde sich eine digitale Variante des Modells anbieten. Dafür müssten der Prozess und der Anforderungsbogen digitalisiert werden, was zwar einen gewissen Aufwand darstellt, jedoch auch Vorteile bietet. Durch die vermehrte Durchführung könnten beispielsweise neue Erkenntnisse in Form einer Feedback-Schleife ins Modell eingearbeitet werden. Man hätte außerdem die Möglichkeit quantitativ und qualitativ die Antworten zu analysieren und somit zusätzlich Erkenntnisse zu gewinnen. Eine Digitalisierung würde wahrscheinlich auch die Handhabung und Übersichtlichkeit des Modells verbessern. Das bedeutet, dass das vorliegende Modell eine Basis darstellt, welche sowohl inhaltlich erweitert, als auch bzgl. der Anwendung und Darstellung weiterentwickelt werden kann.

Am Anfang der Arbeit, in Kapitel 1, wurden anhand der vorliegenden Problemstellung die Forschungsfragen und Ziele der Arbeit abgeleitet. Diese sollen nun reflektiert und beantwortet werden.

Basierend auf einer Innovation oder vorliegenden Idee, wie könnte eine strukturierte Vorgehensweise für das Design eines Kubernetes Clusters aussehen?

In Kapitel 4 wurde ein Prozess vorgeschlagen, welcher dabei unterstützt das Design von Kubernetes Systemen so strukturiert und transparent wie möglich zu gestalten. Der Prozess gliedert sich in sieben Schritte, beschreibt den Ablauf vom Bedarf einer Plattform bis zur Integration des Kubernetes Systems im Unternehmen und baut zu großen Teilen auf der Spezifikation von Anforderungen auf.

Welche Aspekte muss ein Fragebogen zur innovationsgetriebenen Spezifikation von Anforderungen an einen Kubernetes Cluster berücksichtigen?

In Kapitel 4 wurde ebenfalls der Aufbau eines Anforderungsbogens beschrieben, welcher Fragen aus den drei groben Kategorien "Rahmenbedingungen", "funktionale Anforderungen" und "Qualitätseigenschaften" beinhaltet. Der fertige Fragebogen ist in den Anhängen A.15, A.16 und A.17 ersichtlich. Er enthält insgesamt 65 Punkte die dabei helfen sollen die Anforderungen einer Innovation an ihre Kubernetes Plattform so genau wie möglich zu spezifizieren.

LITERATURVERZEICHNIS

Gedruckte Werke

Arundel, John; Domingus, Justin (2019): *Cloud Native DevOps with Kubernetes: Building, Deploying, and Scaling Modern Applications in the Cloud*, O'Reilly Media, Inc., o.O.

Bhat, Sathyajith (2018): *Practical Docker with Python: Build, Release and Distribute Your Python App with Docker*, Apress, o.O.

Buchanan, Steve; Rangama, Janaka; Bellavance Ned (2020): *Introducing Azure Kubernetes Service*, Apress, o.O.

Burns, Brendan; Grant, Brian; Oppenheimer David; Brewer Eric; Wilkes John (2016): *Borg, Omega, and Kubernetes*, in: *Communications of the ACM*, Heft 59/2016, S. 50–57

Burns, Brendan; Beda, Joe; Hightower Kelsey (2019): *Kubernetes: Up and Running*, 2. Auflage. O'Reilly Media, Inc., o.O.

Chun Tie, Ylona; Birks, Melanie; Francis Karen (2019): *Grounded theory research: A design framework for novice researchers*, in: *SAGE Open Med*, Ausgabe 7, online

Collier, David; Elman, Colin (2008): *Qualitative and Multi-Method Research: organizations, publication, and reflections on integration*, in: *Oxford University Press (Hrsg.): The Oxford Handbook of Political Methodology*, Oxford, S. 779 – 795

Helfferich, Cornelia (2019): *Leitfaden- und Experteninterviews*, in: *Handbuch Methoden der empirischen Sozialforschung, Springer VS*, Wiesbaden, S. 669–686

Kane, Sean; Matthias, Karl (2018): *Docker: Up & Running: Shipping Reliable Containers in Production*, O'Reilly Media, Inc., o.O.

Kuckartz, Udo (2016): *Qualitative Inhaltsanalyse. Methoden, Praxis, Computerunterstüzung*, 3. Auflage. Beltz Verlag, Basel

Liebold, Renate; Trinczek, Rainer (2009): *Experteninterview*, in: *Taffertshofer, Andreas (Hrsg.): Handbuch Methoden der Organisationsforschung: quantitative und qualitative Methoden, Wiesbaden*,, S. 32 – 57

Luksa, Marko (2018): *Kubernetes in Action*, Manning Publications, o.O.

Mayring, Philipp; Fenzl, Thomas (2014): *Qualitative Inhaltsanalyse*, in: *Handbuch Methoden der empirischen Sozialforschung, Springer VS*, Wiesbaden, S. 543–556

Mäder, Susanne (2017): *Die Gruppendiskussion als Evaluationsmethode - Entwicklungsgeschichte, Potenziale und Formen*, in: *Zeitschrift für Evaluation*, Heft 12/2017, S. 23 – 51

Miell, Ian; Sayers Hobson, Adrian (2019): *Docker in Practice, Second Edition*, 2. Auflage. Manning Publications, o.O.

Mouat, Adrian (2015): *Using Docker*, O'Reilly Media, Inc., o.O.

Nickoloff, Jeff; Kuenzli, Stephen (2019): *Docker in Action, Second Edition*, 2. Auflage. Manning Publications, o.O.

Roebken, Heinke; Wetzel, Kathrin (2019): *Qualitative und quantitative Forschungsmethoden*, 7. Auflage. Carl von Ossietzky Universität Oldenburg - Center für lebenslanges Lernen C3L, Oldenburg

Sayfan, Gigi (2017): *Mastering Kubernetes*, Packt Publishing, o.O.

Sayfan, Gigi (2019): *Hands-On Microservices with Kubernetes: Build, deploy, and manage scalable microservices on Kubernetes*, Packt Publishing, o.O.

Schoonenboom, Judith; Johnson, Burke (2017): *How to Construct a Mixed Methods Research Design*, in: *Kölner Zeitschrift für Soziologie und Sozialpsychologie*, Heft 69/2017, S. 107 – 131

Schreier, Margrit (2014): *Varianten qualitativer Inhaltsanalyse: Ein Wegweiser im Dickicht der Begrifflichkeiten*, in: *Forum Qualitative Sozialforschung*, Heft 15/2014, online

Vahs, Dieter; Brem, Alexander (2015): *Innovationsmanagement: Von der Idee zur erfolgreichen Vermarktung*, 5. Auflage. Schäffer-Poeschel Verlag, Stuttgart

Zhang, Qi; Liu, Ling; Pu Calton; Dou Qiwei; Wu Liren; Zhou Wei (2018): *A Comparative Study of Containers and Virtual Machines in Big Data Environment*, in: *IEEE Computer Society (Hrsg.): 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), Los Alamitos*, CA, S. 178–185

Onlinequellen

Abdelrazik, Amr (2017): *Docker vs. Kubernetes vs. Apache Mesos: Why What You Think You Know is Probably Wrong*,
https://d2iq.com/blog/docker-vs-kubernetes-vs-apache-mesos [Stand 25.03.2020]

Adam, Tobias (2012): *Die Bewertung von Innovationsideen: Eine empirische Analyse von Bewertungsdimensionen und sozialen Einflussfaktoren*,
https://publications.rwth-aachen.de/record/82789 [Stand 23.11.2020]

Ali, Saad (2019): *Container Storage Interface (CSI) for Kubernetes GA*,
https://kubernetes.io/blog/2019/01/15/container-storage-interface-ga/ [Stand 19.06.2020]

Armonk, N.Y.; Raleigh, N.C. (2019): *IBM Closes Landmark Acquisition of Red Hat for $34 Billion*,
https://www.redhat.com/en/about/press-releases/
ibm-closes-landmark-acquisition-red-hat-34-billion-defines-open-hybrid-cloud-future [Stand 06.08.2020]

Bakker, Paul (2020): *One year using Kubernetes in production: Lessons learned*,
https://techbeacon.com/devops/one-year-using-kubernetes-production-lessons-learned [Stand 06.08.2020]

Beda, Joe (2018): *4 Years of K8s*,
https://kubernetes.io/blog/2018/06/06/4-years-of-k8s/ [Stand 19.06.2020]

Belamaric, John (2018): *CoreDNS GA for Kubernetes Cluster DNS*,
https://kubernetes.io/blog/2018/07/10/coredns-ga-for-kubernetes-cluster-dns/ [Stand 19.06.2020]

Bell, Tim (2019): *Compute Federation at CERN*,
https://indico.cern.ch/event/830634/contributions/3507644/attachments/1888185/3113233/20190802_Compute_
Federation_v2.pdf [Stand 27.10.2020]

Berger, Alon (2020): *Kubernetes-as-a-Service: EKS vs. AKS vs. GKE*,
https://blog.alcide.io/kubernetes-as-a-service-eks-vs.-aks-vs.-gke [Stand 06.08.2020]

Bohn, Bjoern (2020): *Container: Mirantis verspricht Weiterentwicklung von Docker Swarm*, https://www.heise.de/developer/meldung/Container-Mirantis-verspricht-Weiterentwicklung-von-Docker-Swarm-4667834.html [Stand 19.06.2020]

Bonizi, Grazi (2020): *Why Not Use Kubernetes?* https://medium.com/better-programming/why-not-use-kubernetes-52a89ada5e22 [Stand 06.08.2020]

Bryant, Daniel (2018): *Shopify's Journey to Kubernetes and PaaS: Niko Kurtti at QCon NY*, https://www.infoq.com/news/2018/07/shopify-kubernetes-paas/ [Stand 06.08.2020]

Buehrle, Anita (2019): *Introduction to Service Meshes on Kubernetes and Progressive Delivery*, https://www.weave.works/blog/introduction-to-service-meshes-on-kubernetes-and-progressive-delivery [Stand 19.06.2020]

Carey, Scott (2020): *Etsy goes cloud native to better scale to seasonal demands*, https://www.infoworld.com/article/3530304/etsy-goes-cloud-native-to-better-scale-to-seasonal-demands.html [Stand 06.08.2020]

Carter, Eric (2019): *Sysdig 2019 Container Usage Report: New Kubernetes and security insights*, https://sysdig.com/blog/sysdig-2019-container-usage-report/ [Stand 10.12.2020]

Casey, Kevin (2017): *Kubernetes by the numbers: 10 compelling stats*, https://enterprisersproject.com/article/2017/11/kubernetes-numbers-10-compelling-stats [Stand 27.10.2020]

Cerruti, Martin (2020): *Do You Actually Need Kubernetes?* https://medium.com/better-programming/do-you-actually-need-kubernetes-a342b0a90fd8 [Stand 06.08.2020]

Chircop, Alex; Hoole, Quinton; Kitson Clinton; Li Xiang; Pabon Luis; Yang Xing (2020): *CNCF Storage Landscape*, https://bit.ly/cncf-storage-whitepaperV2 [Stand 19.06.2020]

Correa, David (2020): *Application Container Market is Expected to Reach $8.20 Billion by 2025*, https://www.globenewswire.com/news-release/2020/05/06/2028585/0/en/Application-Container-Market-is-Expected-to-Reach-8-20-Billion-by-2025-Says-Allied-Market-Research.html [Stand 25.08.2020]

Gibbs, Toria (2018): *Deploying to Google Kubernetes Engine*, https://codeascraft.com/2018/06/05/deploying-to-google-kubernetes-engine/ [Stand 06.08.2020]

GitHub Inc., (Hrsg) (2019): *The State of the OCTOVERSE*, https://octoverse.github.com/ [Stand 25.08.2020]

Goasduff, Laurence (2019b): *Navigate private cloud, public cloud and the edge for infrastructures of the future*. https://www.gartner.com/smarterwithgartner/modernize-it-infrastructure-in-a-hybrid-world/ [Stand 19.06.2020]

Gracey, Andrew (2019): *Technical Deep-Dive of Container Runtimes*, https://www.suse.com/c/technical-deep-dive-of-container-runtimes/ [Stand 25.03.2020]

Gunaratne, Imesh (2016): *The Evolution of Linux Containers and Their Future*, https://dzone.com/articles/evolution-of-linux-containers-future [Stand 25.03.2020]

Haq, Fahim ul (2019): *Why (and when) you should use Kubernetes*, https://hackernoon.com/why-and-when-you-should-use-kubernetes-8b50915d97d8 [Stand 06.08.2020]

Hecht, Lawrence E. (2018): *WHAT THE DATA SAYS ABOUT KUBERNETES DEPLOYMENT PATTERNS*, https://thenewstack.io/data-says-kubernetes-deployment-patterns/ [Stand 06.08.2020]

Hickey, Alex (2018): *How Spotify is migrating from an in-house Docker orchestration platform to Kubernetes*, https://www.ciodive.com/news/how-spotify-is-migrating-from-an-in-house-docker-orchestration-platform-to/525465/ [Stand 06.08.2020]

Hombergs, Tom (2020): *What is Upstream and Downstream in Software Development?* https://reflectoring.io/upstream-downstream/ [Stand 06.08.2020]

Hung, Cheryl (2020): *Introducing the CNCF Technology Radar*, https://www.cncf.io/blog/2020/06/12/introducing-the-cncf-technology-radar/ [Stand 07.12.2020]

Jackson, Joab (2020): *eBay Rolls Out Kubernetes for Performance-Sensitive Search Operations*, https://thenewstack.io/ebay-rolls-out-kubernetes-for-performance-sensitive-search-operations/ [Stand 06.08.2020]

Kerner, Sean Michael (2017): *Why HBO Chose Kubernetes To Help Stream Game Of Thrones*, https://www.silicon.co.uk/cloud/hbo-got-kubernetes-225897 [Stand 06.08.2020]

Kidd, Chrissy (2019): *How eBay is Reinventing Their IT with Kubernetes Replatforming Program*, https://www.bmc.com/blogs/ebay-kubernetes-replatforming/ [Stand 06.08.2020]

Klein, Evan (2019): *Kubernetes as a Service: GKE vs. AKS vs. EKS*, https://logz.io/blog/kubernetes-as-a-service-gke-aks-eks/ [Stand 06.08.2020]

Koutoupis, Petros (2018): *Everything You Need to Know about Linux Containers, Part I: Linux Control Groups and Process Isolation*, https://www.linuxjournal.com/content/ everything-you-need-know-about-linux-containers-part-i-linux-control-groups-and-process [Stand 25.03.2020]

Kuncoro, Giri (2018): *Containers, Shells, and Mounts, from Scratch*, https://blog.gojekengineering.com/containers-shells-and-mounts-from-scratch-18a2cc6cdd82 [Stand 25.03.2020]

Kvitka, Caroline (2020): *Creating Memorable Gaming Experiences with Kubernetes*, https://rancher.com/blog/2020/ubisoft [Stand 06.08.2020]

Lamba, Shruti (2017): *Key Considerations for Selecting Open Source Software*, https://www.tothenew.com/blog/key-considerations-for-selecting-open-source-software/ [Stand 27.10.2020]

Lardinois, Frederic (2019a): *Mirantis acquires Docker Enterprise*, https://techcrunch.com/2019/11/13/mirantis-acquires-docker-enterprise/ [Stand 06.08.2020]

Lardinois, Frederic (2019b): *With the acquisition closed, IBM goes all in on Red Hat*, https://techcrunch.com/2019/08/01/with-the-acquisition-closed-ibm-goes-all-in-on-red-hat/ [Stand 06.08.2020]

Lewis, Ian (2017): *Container Runtimes Part 1: An Introduction to Container Runtimes*, https://www.ianlewis.org/en/container-runtimes-part-1-introduction-container-r [Stand 25.03.2020]

Mandel, Mark (2018): *Introducing Agones: Open-source, multiplayer, dedicated game-server hosting built on Kubernetes*, https://cloud.google.com/blog/products/gcp/introducing-agones-\ \open-source-multiplayer-dedicated-game-server-hosting-built-on-kubernetes [Stand 06.08.2020]

Matsiukevich, Siarhei (2018): *Kubernetes Networking: How to Write Your Own CNI Plug-in with Bash*, https://www.altoros.com/blog/kubernetes-networking-writing-your-own-simple-cni-plug-in-with-bash/ [Stand 25.03.2020]

McCarty, Scott (2018): *A Practical Introduction to Container Terminology*, https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/ [Stand 25.03.2020]

McCarty, Scott (2019): *How to navigate the Kubernetes learning curve*, https://opensource.com/article/19/6/kubernetes-learning-curve [Stand 27.10.2020]

McMahon, Kim (2020): *2019 CNCF Survey results are here: Deployments are growing in size and speed as cloud native adoption becomes mainstream*, https://www.cncf.io/blog/2020/03/04/2019-cncf-survey-results-are-here-deployments-are-growing-in-size-and-speed-as-cloud-native-adoption-becomes-mainstream/ [Stand 25.03.2020]

Meeker, Heather (2017): *Open source licensing: What every technologist should know*, https://opensource.com/article/17/9/open-source-licensing [Stand 27.10.2020]

Mey, Günter; Vock, Rubina; Ruppel Paul (2020): *GÜTEKRITERIEN QUALITATIVER FORSCHUNG*, https://studi-lektor.de/tipps/qualitative-forschung/guetekriterien-qualitativer-forschung.html#src4 [Stand 23.11.2020]

Michael, Michael; Lang, Patrick (2019): *Kubernetes v1.14 delivers production-level support for Windows nodes and Windows containers*, https://kubernetes.io/blog/2019/04/01/kubernetes-v1.14-delivers-production-level-support-for-windows-nodes-and-windows-containers/ [Stand 19.06.2020]

Motschnig, Renate (2012): *Qualitative Forschung in der Medienpädagogik*, https://cewebs.cs.univie.ac.at/WA.AET.FM.UE/ws12/index.php?m=D&t=info&c=show&CEWebS_what=G~252~tekriterien~32~qualitativer~32~Forschung,~32~Zusammenfassung [Stand 23.11.2020]

MSV, Janakiram (2018): *Red Hat Acquires CoreOS For $250 Million*, https://www.forbes.com/sites/janakirammsv/2018/01/30/red-hat-acquires-coreos-for-250-million [Stand 25.03.2020]

Neufeld, Dale (2018): *Shopify's Infrastructure Collaboration with Google*, https://engineering.shopify.com/blogs/engineering/shopify-infrastructure-collaboration-with-google [Stand 06.08.2020]

Osnat, Rani (2020): *A Brief History of Containers From the 1970s Till Now*, https://blog.aquasec.com/a-brief-history-of-containers-from-1970s-chroot-to-docker-2016 [Stand 25.03.2020]

Patig, Susanne; Dibbern, Jens (2018): *Requirements Engineering*, https://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/Systementwicklung/Hauptaktivitaten-der-Systementwicklung/Problemanalyse-/Requirements-Engineering/index.html [Stand 19.06.2020]

Platform9 Systems Inc., (Hrsg) (2019): *Kubernetes for Machine Learning*, https://platform9.com/blog/kubernetes-for-machine-learning/ [Stand 06.08.2020]

Platform9 Systems Inc., (Hrsg) (2020): *10 Kubernetes Performance tips*, https://platform9.com/blog/10-kubernetes-performance-tips/ [Stand 27.10.2020]

Pyasi, Arun (2020): *How to Create Docker Container using Dockerfile*, https://linoxide.com/linux-how-to/dockerfile-create-docker-container/ [Stand 06.08.2020]

Radygin, Andrey; Shurupov, Dmitry (2019): *Comparing Ingress controllers for Kubernetes*, https://medium.com/flant-com/comparing-ingress-controllers-for-kubernetes-9b397483b46b [Stand 19.06.2020]

Ramanathan, Kalyan (2019): *5 business reasons why every CIO should consider Kubernetes*, https://www.sumologic.com/blog/why-use-kubernetes/ [Stand 25.08.2020]

Red Hat Inc., (Hrsg) (2019): *Is the docker package available for Red Hat Enterprise Linux 8?* https://access.redhat.com/solutions/3696691 [Stand 25.03.2020]

Reddy, Anil (2018): *Kubernetes with Flannel - Understanding the Networking - Part 2*, https://medium.com/@anilkreddyr/kubernetes-with-flannel-understanding-the-networking-part-2-78b53e5364c7 [Stand 25.03.2020]

Reznik, Pini (2018): *When is the WRONG time to use Kubernetes?* https://blog.container-solutions.com/when-is-the-wrong-time-to-use-kubernetes [Stand 06.08.2020]

Rilee, Kynan (2017): *Mount volumes into a running container*, https://medium.com/kokster/mount-volumes-into-a-running-container-65a967bee3b5 [Stand 25.03.2020]

Rimol, Meghan (2019): *Gartner Top 10 Trends Impacting Infrastructure Operations for 2020*, https://www.gartner.com/smarterwithgartner/gartner-top-10-trends-impacting-infrastructure-operations-for-2020 [Stand 25.03.2020]

Rocha, Ricardo (2018): *Multi-Cloud Federated Kubernetes at CERN*, https://openlab.cern/sites/openlab.web.cern.ch/files/2018-05/kubeconeurope2018-cern-180507122303.pdf [Stand 27.10.2020]

Rosenhouse, Gabriel (2016): *Don't mix goroutines and namespaces: Part 1*, http://engineering.pivotal.io/post/goroutines-and-namespaces-part-1/ [Stand 25.03.2020]

Samdan, Emrah (2020): *Do you really need Kubernetes?* https://blog.thundra.io/do-you-really-need-kubernetes [Stand 06.08.2020]

Schmitt, Mark (2020): *Demystifying Multus*, https://www.openshift.com/blog/demystifying-multus [Stand 19.06.2020]

Stack Exchange Inc., (Hrsg) (2019): *Developer Survey Results 2019*, https://insights.stackoverflow.com/survey/2019 [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2017): *Inside JD.com's Shift to Kubernetes from OpenStack*, https://kubernetes.io/blog/2017/02/inside-jd-com-shift-to-kubernetes-from-openstack/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020a): *Bare-metal considerations*, https://kubernetes.github.io/ingress-nginx/deploy/baremetal/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020b): *Case Study: Babylon*, https://kubernetes.io/case-studies/babylon [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020c): *Case Study: Capital One Bank*, https://kubernetes.io/case-studies/capital-one/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020d): *Case Study: CERN*, https://kubernetes.io/case-studies/cern/ [Stand 27.10.2020]

The Linux Foundation, (Hrsg) (2020e): *Case Study: JD.com*, https://kubernetes.io/case-studies/jd-com/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020f): *Case Study: Spotify*, https://kubernetes.io/case-studies/spotify/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020h): *Cluster Networking*, https://kubernetes.io/docs/concepts/cluster-administration/networking/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020i): *CNCF Cloud Native Interactive Landscape*, https://landscape.cncf.io/ [Stand 2.03.2020]

The Linux Foundation, (Hrsg) (2020j): *ConfigMaps*, https://kubernetes.io/docs/concepts/configuration/configmap/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020k): *CSI Volume Cloning*, https://kubernetes.io/docs/concepts/storage/volume-pvc-datasource/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020l): *Dynamic Volume Provisioning*, https://kubernetes.io/docs/concepts/storage/dynamic-provisioning/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020m): *Ingress*, https://kubernetes.io/docs/concepts/services-networking/ingress/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020n): *Logging Architecture*, https://kubernetes.io/docs/concepts/cluster-administration/logging/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020o): *Namespaces*, https://kubernetes.io/docs/tasks/administer-cluster/namespaces/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020p): *Pods*, https://kubernetes.io/docs/concepts/workloads/pods/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020q): *Production-Grade Container Orchestration*, https://kubernetes.io/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020r): *Secrets*, https://kubernetes.io/docs/concepts/configuration/secret/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020s): *Service*, https://kubernetes.io/docs/concepts/services-networking/service/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020t): *Understanding Kubernetes Objects*, https://kubernetes.io/docs/concepts/overview/working-with-objects/kubernetes-objects/ [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020v): *Volume Expansion*, https://kubernetes-csi.github.io/docs/volume-expansion.html [Stand 06.08.2020]

The Linux Foundation, (Hrsg) (2020w): *Volume Snapshots*, https://kubernetes.io/docs/concepts/storage/volume-snapshots/ [Stand 06.08.2020]

Thiry, Daniel (2019): *Kubernetes: Advantages and Disadvantages - The Business Perspective*, https://devspace.cloud/blog/2019/10/31/advantages-and-disadvantages-of-kubernetes [Stand 09.12.2020]

ThoughtWorks Inc., (Hrsg) (2020): *TECHNOLOGY RADAR Platforms: Kubernetes*, https://www.thoughtworks.com/radar/platforms/kubernetes [Stand 07.12.2020]

Timms, Simon (2018): *Devops and Microservices - Symbiotes*, https://westerndevs.com/microservices-devops/ [Stand 25.03.2020]

Tozzi, Christopher (2020): *Kubernetes Distribution: What It Is and What It Isn't*, https://containerjournal.com/topics/container-ecosystems/kubernetes-distribution-what-it-is-and-what-it-isnt/ [Stand 19.06.2020]

Turner-Trauring, Itamar (2020): *"Let's use Kubernetes!" Now you have 8 problems*, https://pythonspeed.com/articles/dont-need-kubernetes/ [Stand 06.08.2020]

Vaughan-Nichols, Steven (2019): *Mirantis acquires Docker Enterprise*, https://www.zdnet.com/article/mirantis-acquires-docker-enterprise/ [Stand 06.08.2020]

Vizard, Mike (2020): *Alcide Simplifies PCI, GDPR Compliance for Kubernetes*, https://containerjournal.com/topics/container-security/alcide-simplifies-pci-gdpr-compliance-for-kubernetes/ [Stand 27.10.2020]

Wallen, James (2020): *Check Out Podman, Red Hat's daemon-less Docker Alternative*, https://thenewstack.io/check-out-podman-red-hats-daemon-less-docker-alternative/ [Stand 25.03.2020]

Williams, Alex (2020a): *Spotify's Golden Path to Kubernetes Adoption Had Many Twist and Turns*, https://thenewstack.io/spotifys-golden-path-to-kubernetes-adoption-had-many-twist-and-turns/ [Stand 06.08.2020]

Williams, Alex; Gain, Cameron (2020b): *How CERN Accelerates with Kubernetes, Helm, Prometheus and CoreDNS*, https://thenewstack.io/how-cern-accelerates-with-kubernetes-helm-prometheus-and-coredns/ [Stand 27.10.2020]

Winter, Stefanie (2000): *Quantitative vs. Qualitative Methoden*, http://nosnos.synology.me/MethodenlisteUniKarlsruhe/imihome.imi.uni-karlsruhe.de/nquantitative_vs_qualitative_methoden_b.html [Stand 23.11.2020]

Zhu, David (2019): *Kubernetes 1.17 Feature: Kubernetes In-Tree to CSI Volume Migration Moves to Beta*, https://kubernetes.io/blog/2019/12/09/kubernetes-1-17-feature-csi-migration-beta/ [Stand 19.06.2020]

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

GLOSSAR

AI Artificial Intelligence. 31, 82

API Application Programming Interface. 19, 22–25, 28, 35, 36, 39, 42, 54, 112, 113, 122, 130, 135

CERN Europäische Organisation für Kernforschung. 78, 79

CGroup Control Group. 9, 15, 16, 18, 19, 107

CNCF Cloud Native Computing Foundation. 1, 2, 5, 12, 36, 39, 43, 101

CNI Container Network Interface. 34, 35, 102, 115, 116

Container-Orchestration Die Container-Orchestrierung automatisiert die Bereitstellung, Verwaltung, Skalierung und Vernetzung von Containern. [220]. 11, 13, 14, 19, 20, 101

CRI Container Runtime Interface. 11, 34, 114

CSI Container Storage Interface. 36, 37, 39, 118, 119

DDoS Distributed Denial of Service. 117

Deployment Im IT-Kontext wird damit der Einsatz aller Prozesse beschrieben, die mit der ordnungsgemäßen Inbetriebnahme neuer Software in ihrer Zielumgebung verbunden sind, einschließlich Installation, Konfiguration, Tests und Durchführung der erforderlichen Anderungen. 24

DevOps DevOps beschreibt einen Prozessverbesserungs-Ansatz aus den Bereichen der Softwareentwicklung und Systemadministration. DevOps ist ein Kunstwort aus den Begriffen Development (englisch für Entwicklung) und IT Operations (englisch für IT-Betrieb). DevOps soll durch gemeinsame Anreize, Prozesse und Software-Werkzeuge eine effektivere und effizientere Zusammenarbeit der Bereiche Dev, Ops und Qualitätssicherung ermöglichen. . 6, 8, 50, 101

DNS Domain Name System. 39, 40

DSGVO Datenschutz-Grundverordnung. 125

Filesystem Das Filesystem ist für die Organisation der auf Speichermedien abgelegten Dateien verantwortlich. Es ist ein Bestandteil eines Betriebssystems und kann von Speichermedium zu Speichermedium unterschiedlich sein. Das Filesystem definiert Dateinamenskonventionen, Dateiattribute oder die Zugriffskontrolle. 15–18, 36, 37, 119, 132

GA General Availability. 36

GDPR General Data Protection Regulation. 125

GID Group Identifier. 18

GUI Graphical User Interface. 117, 120, 124

Host Ein Host ist ein Rechner der eine Umgebung oder einen Service zur Verfügung stellt. 16, 18, 19, 24, 35

[220] weitere Informationen unter: https://www.redhat.com/en/topics/containers/what-is-container-orchestration

IOPS Input/Output Operations Per Second. 16

Kernel Der Linux-Kernel ist die Hauptkomponente eines Linux-Betriebssystems und stellt die zentrale Schnittstelle zwischen der Hardware eines Computers und seinen Prozessen dar. Er kommuniziert zwischen den beiden und verwaltet die Ressourcen so effizient wie möglich. Der Kernel wird so genannt, weil er - wie ein Keim innerhalb einer harten Schale - innerhalb des Betriebssystems existiert und alle wichtigen Funktionen der Hardware steuert, unabhängig davon, ob es sich um ein Telefon, einen Laptop, einen Server oder eine andere Art von Computer handelt. [221] . 9, 16–18, 33, 113

KPI Key Performance Indicator. 53

Leader Election In der Informatik ist die Wahl des Leaders der Prozess, bei dem ein einzelner Prozess oder eine Programminstanz als Organisator einer auf mehrere Instanzen verteilten Aufgabe bestimmt wird. Vor Beginn der Aufgabe ist keiner Instanz, die in der Regel auf mehrere Knoten verteilt sind, entweder nicht bekannt welche Instanz als Leiter der Aufgabe fungieren wird, oder sie sind nicht in der Lage, mit dem aktuellen Leader zu kommunizieren. Nachdem ein Algorithmus zur Wahl des Leaders durchlaufen wurde, erkennt jedoch jede Instanz im gesamten Netzwerk eine bestimmte, einzigartige Instanz als den Leader an. 28

Library Eine Library, auch Programmbibliothek genannt, bezeichnet in der Softwareentwicklung eine Sammlung von Methoden oder Codefragmenten, die Lösungen für thematisch zusammengehörende Problemstellungen anbieten. Libraries sind im Unterschied zu Programmen keine eigenständig lauffähigen Einheiten, sondern sie enthalten Hilfsmodule, die von Programmen angefordert werden. 6, 35

Linux Linux ist eine Sammelbezeichnung für alle Betriebssysteme die auf dem Linux-Kernel basieren. 9, 16–18, 24–26, 33, 80, 113, 125, 129

Load-Balancer Systeme für Load Balancing dienen zur Lastverteilung von Netzwerkverkehr für Server. Dazu werden bereitgestellte Dienste, z. B. ein Webserver, auf mehrere Server verteilt. Das Ziel ist, die Last des Ansturms auf einen Dienst auf mehrere Server aufzuteilen und den Dienst bei einem Hardware-Ausfall vor dem Totalausfall zu schützen. 21, 105

Load-Balancing siehe Load-Balancer. 21–23, 117, 122

LTS Der Zusatz LTS bei der Versionsbezeichnung, z.B. 20.04 LTS, steht f?r Long Term Support (englisch für langfristige Unterstützung) und bedeutet, dass diese Versionen länger als andere gepflegt werden, d.h. mit Aktualisierungen und Fehlerkorrekturen ("updates") unterstützt werden. 15, 81, 127

LXC LXC ist eine Benutzerschnittstelle für Linux-Kernel Funktionen. Mit Hilfe einer API und einfachen Tools können Linux-Benutzer System- oder Anwendungscontainer erstellen und verwalten.. 9, 10

ML Machine Learning. 31, 79, 82, 83

NAT Network Address Translation. 35

Node Als Node wird ein Rechner in einem Cluster bezeichnet. 21–23, 25, 26, 33, 35, 38, 113, 114, 116, 121, 136

OCI Open Container Initiative. 10, 11, 15, 19, 115

[221] weitere Informationen unter: https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel

On-Premises On-Premises bezeichnet ein Nutzungs- und Lizenzmodell für serverbasierte Computerprogramme, welches den Betrieb von Software im lokalen Datacenter beschreibt. Seitdem die lokale Nutzung zunehmend von Software as a Service (SaaS) oder Cloud Computing verdrängt wird, ist der Begriff On-Premises entstanden.. 12, 13, 34, 38, 50, 54

Proof of Concept Ein Proof of Concept (POC) ist eine Übung, bei der sich die Arbeit darauf konzentriert festzustellen, ob eine Idee in die Realität umgesetzt werden kann. Ein Proof-of-Concept dient dazu, die Durchführbarkeit der Idee zu bestimmen oder zu verifizieren, dass die Idee wie vorgesehen funktionieren wird. [222]. 139

Proxy Ein Proxy-Server fungiert als Gateway zwischen einem Rechner und dem Internet. Es handelt sich um einen Zwischenserver, der Endbenutzer von den Websites trennt, die sie durchsuchen. Proxy-Server bieten unterschiedliche Funktionalitäts-, Sicherheits- und Datenschutzniveaus, je nach Anwendungsfall, Bedürfnissen oder Unternehmensrichtlinien. [223] . 120

PV Persistent Volume. 27, 36, 37, 119

PVC Persistent Volume Claim. 27, 36, 119

Reverse Proxy Der Reverse Proxy holt Ressourcen für einen externen Client von einem oder mehreren internen Servern. Die Umsetzung der Adresse ist atypisch und der Richtung des Aufrufes entgegengesetzt (deutsch 'umgekehrter Proxy'). Die wahre Adresse des internen Zielsystems bleibt dem externen Client verborgen. Das unterscheidet ihn vom typischen Proxy, der mehreren Clients eines internen (in sich abgeschlossenen) Netzes den Zugriff auf ein externes Netz gewährt. 26, 35

Service Level Agreement Ein Service-Level-Agreement definiert das Serviceniveau, das man von einem Anbieter erwarten kann, und legt die Metriken fest, an denen der Service gemessen wird. Außerdem regelt es Abhilfemaßnahmen oder Strafen für den Fall, dass vereinbarte Service-Levels nicht erreicht werden. [224]. 54, 135

SIG Special Interest Group. 36

SMI Service Mesh Interface. 39, 122

Stage Eine Stage bezeichnet eine Umgebung die, zum Entwickeln, Testen oder zum Betrieb von Applikationen verwendet wird. Übliche Stages bzw. Umgebungen sind Entwicklung (Development), Test, Vorproduktion(Staging) und Produktion. 25

UID User Identifier. 18

[222] weitere Informationen unter: https://searchcio.techtarget.com/definition/proof-of-concept-POC
[223] weitere Informationen unter: https://www.varonis.com/blog/what-is-a-proxy-server/
[224] weitere Informationen unter: https://www.cio.com/article/2438284/outsourcing-sla-definitions-and-solutions.html

ANHANG

A.1 Beispiel: CGroups Konfiguration im sys Filesystem

Das folgende Beispiel zeigt, dass der Container mit der ID *68450f882a56d4510ecc3cecc3a24bf4e42d09b3d9987efebeed5adba63c65af* zur gleichnamigen CGroup gehört. Diese CGroup verwaltet die Memory (Arbeitsspeicher) Ressourcen des Container. Einerseits legt die CGroup die Limits an maximal verwendbarem Arbeitsspeicher fest. Andererseits zeigt sie aber auch an, was derzeit vom Container verwendet wird. In diesem Beispiel ist erkennbar, dass der Container 52428800 Bytes (=50MB) Arbeitsspeicher benutzen darf, wovon er zurzeit 43315200 Bytes (86,63%) verwendet. Des Weiteren ist erkennbar, dass diese CGroup nur den Process mit der Prozess-ID (PID) 15207 verwaltet. Aus Gründen der Lesbarkeit wurde der absolute Pfad im Prompt bei den letzten Kommandos verkürzt dargestellt.

```
root@localhost:/sys/fs/cgroup/memory/kubepods/burstable/pode123/ \
68450f882a56d4510ecc3cecc3a24bf4e42d09b3d9987efebeed5adba63c65af# ls -l
total 0
-rw-r--r-- 1 root root 0 Apr  7 20:21 cgroup.clone_children
--w--w--w- 1 root root 0 Apr  7 20:21 cgroup.event_control
-rw-r--r-- 1 root root 0 Apr  7 05:30 cgroup.procs
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.failcnt
--w------- 1 root root 0 Apr  7 20:21 memory.force_empty
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.kmem.failcnt
-rw-r--r-- 1 root root 0 Apr  7 05:30 memory.kmem.limit_in_bytes
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.kmem.max_usage_in_bytes
-r--r--r-- 1 root root 0 Apr  7 20:21 memory.kmem.slabinfo
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.kmem.tcp.failcnt
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.kmem.tcp.limit_in_bytes
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.kmem.tcp.max_usage_in_bytes
-r--r--r-- 1 root root 0 Apr  7 20:21 memory.kmem.tcp.usage_in_bytes
-r--r--r-- 1 root root 0 Apr  7 20:21 memory.kmem.usage_in_bytes
-rw-r--r-- 1 root root 0 Apr  7 05:30 memory.limit_in_bytes
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.max_usage_in_bytes
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.move_charge_at_immigrate
-r--r--r-- 1 root root 0 Apr  7 20:21 memory.numa_stat
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.oom_control
---------- 1 root root 0 Apr  7 20:21 memory.pressure_level
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.soft_limit_in_bytes
-r--r--r-- 1 root root 0 Apr  7 20:21 memory.stat
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.swappiness
-r--r--r-- 1 root root 0 Apr  7 20:21 memory.usage_in_bytes
-rw-r--r-- 1 root root 0 Apr  7 20:21 memory.use_hierarchy
-rw-r--r-- 1 root root 0 Apr  7 20:21 notify_on_release
-rw-r--r-- 1 root root 0 Apr  7 20:21 tasks
root@localhost:/sys/.../68450f882a56d45...65af# cat memory.limit_in_bytes
52428800
root@localhost:/sys/.../68450f882a56d45...65af# cat memory.usage_in_byt
es
43315200
root@localhost:/sys/.../68450f882a56d45...65af# cat cgroup.procs
15207
```

Listing 1: CGroup Beispiel

A.2 Beispiel: YAML Darstellung eines einfachen Pods

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  namespace: my-namespace
  labels:
    app: myapp
spec:
  containers:
    - name: myapp-container
      image: busybox
      command: ['sh', '-c', 'echo Hello Kubernetes! && sleep 3600']
```

Listing 2: Beispiel YAML file eines Pods

A.3 Beispiel: YAML Darstellung einer ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  creationTimestamp: "2020-04-01T20:31:08Z"
  name: example-config-map
  namespace: my-namespace
data:
  dummy-data: "some data"
```

Listing 3: Beispiel YAML file einer ConfigMap

A.4 Beispiel: YAML Darstellung eines Service

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
  namespace: my-namespace
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Listing 4: Beispiel YAML file eines Service

A.5 Beispiel: YAML Darstellung eines Deployments

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Listing 5: Beispiel YAML file eines Deployments

A.6 Beispiel: YAML Darstellung eines Ingress Service

```yaml
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  namespace: my-namespace
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          serviceName: test
          servicePort: 80
```

Listing 6: Beispiel YAML file eines Ingress Service

A.7 Beispiel: Erstellung eines Pod Objektes

Die folgenden Codebeispiele zeigen auf wie ein Kubernetes-Objekt, in diesem Fall ein Pod, entweder via kubectl oder direkt via Kubernetes API erzeugt werden kann. Die Definition der Objekte findet sich in der Kubernetes API Referenz.[225]

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx1
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
    ports:
    - containerPort: 80

# Installation am Cluster: kubectl apply -f pod-definition.yaml
```

Listing 7: Beispiel: Pod YAML Definition

```
curl -k -X POST -d @- \
-H "Authorization: Bearer $TOKEN" \
-H 'Accept: application/json' \
-H 'Content-Type: application/json' \
https://{kubernetes-url}/api/v1/pods <<'EOF'
{
  "apiVersion": "v1",
    "kind": "Pod",
    "metadata": {
        "name": "nginx1"
    },
    "spec": {
        "containers": [
            {
                "name": "nginx",
                "image": "nginx:1.7.9",
                "ports": [
                    {
                        "containerPort": 80
                    }
                ]
            }
        ]
    }
}
EOF
```

Listing 8: Beispiel: Erstellung eines Pod via API call

[225] https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.19/

A.8 Beispiel: Funktionsweise von Image Layer

In Abbildung A.1 ist die Layer-Hierarchie der beiden Images *dockerinaction/ch3_ myapp* und *dockerinaction/ch3_ myotherapp* dargestellt. Beide Images bestehen aus den beiden Basislayern *debian:buster-slim* (ID: 83ed3c583403) und *openjdk:11.0.4-jdk-slim* (ID: 4820fdebc4fb). Vom Layer *openjdk:11.0.4-jdk-slim* werden dann die beiden finalen Layer *dockerinaction/ch3_ myapp* (ID: 0858f7736a46) und *dockerinaction/ch3_ myotherapp* (ID: c0a16f5f469c) abgeleitet.[226]

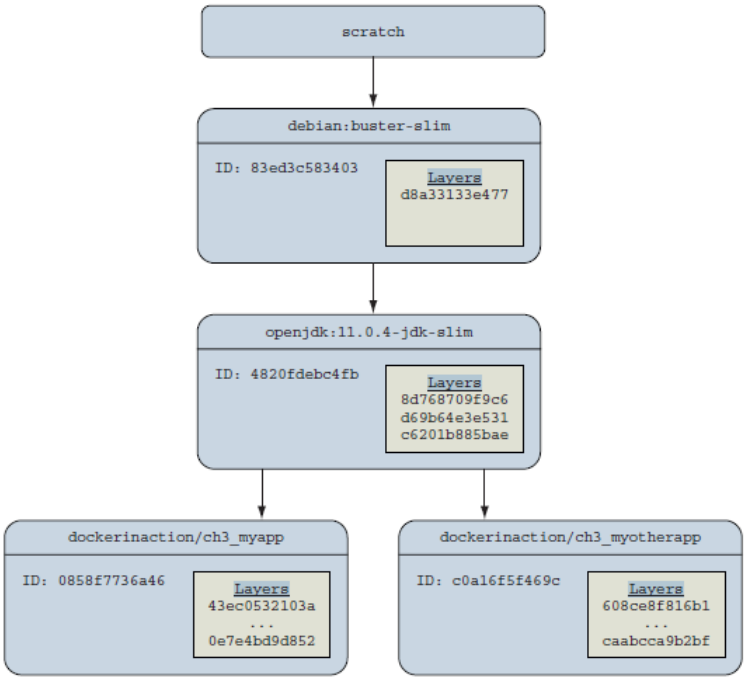


Abb. A.1: **Layerhierarchie zweier Images**
Quelle: Nickoloff (2019), Fig. 3.7, pp. 59

[226] Vgl. Nickoloff (2019), S. 57ff.

A.9 Eigenschaften: Kubernetes Distribution

Die folgenden Eigenschaften eignen sich dafür Kubernetes Distributionen zu beschreiben und zu unterscheiden. Aspekte und Eigenschaften welche eine Distribution zwar beschreiben, für alle Distributionen jedoch gleich oder bereits ein Basismerkmal sind, werden an dieser Stelle nicht erwähnt.

[ED01] Aktuell unterstützte Kubernetes Version(en): Dabei stellt sich die Frage, welche Version(en) von Upstream Kubernetes werden in der aktuellen Version der Distribution unterstützt bzw. verwendet. Im Normalfall gilt "je neuer desto besser", zumindest aber sollten eine der letzten drei *minor relases*, nach Semantic Versioning[227], unterstützt werden. Ältere Versionen werden von der Community nicht mehr gewartet[228].

[ED02] Alter der Distribution: Das Alter kann etwas über den Erfahrungs- und Reifegrad der Distribution aussagen. Im Normalfall kann davon ausgegangen werden, dass ältere Distributionen robuster und reicher an Features sind.

[ED03] CNCF konforme Kubernetes Version: Die CNCF ermöglicht es Herstellern von Kubernetes Distributionen, im speziellen Kubernetes-as-a-Service-Providern, ihr Produkt für die jeweilige Kubernetes Version zu zertifizieren. Dabei wird sichergestellt, dass alle Vorgaben der entsprechenden Kubernetes API in der gewählten Version eingehalten werden[229].

[ED04] Master Upgrade Prozess: Diese Metrik beschreibt wie aufwändig und kompliziert es ist, die Kubernetes Version der Master Nodes auf eine höhere Version zu heben. Da es hierfür keine klar bestimmte Kennzahl gibt, soll der Aufwand von Experten auf einer Skala von 1 bis 5 eingeschätzt werden. Dabei steht 1 für "vollständig automatisiert", 2 bis 4 sind Abstufungen und 5 bedeutet "vollständig manuell".

[ED05] Worker Upgrade Prozess: Diese Metrik beschreibt, wie aufwändig und kompliziert es ist, die Kubernetes Version der Worker Nodes auf eine höhere Version zu heben. Da es hierfür keine klar bestimmte Kennzahl gibt, soll der Aufwand von Experten auf einer Skala von 1 bis 5 eingeschätzt werden. Dabei steht 1 für "vollständig automatisiert", 2 bis 4 sind Abstufungen und 5 bedeutet "vollständig manuell".

[ED06] Container-Runtime: Gewisse Distributionen liefern eigene Container Runtimes mit oder sind nur mit bestimmten Runtimes kompatibel. Sollte das Unternehmen eine gewisse Strategie bzgl. einer Container-Runtime verfolgen, so kann diese Eigenschaft ausschlaggebend werden.

[ED07] Control Plane SLA: Diese Metrik ist vor allem für Kubernetes-As-A-Service Distributionen relevant. Sie gibt an zu wie viel Prozent die Control Plane, sprich die Kubernetes API, in einem gewissen Zeitraum zur Verfügung steht. Der SLA Zeitraum bezieht sich meist auf ein Monat, Beispiele hierfür sind den Fußnoten zu entnehmen [230] [231] [232].

[ED08] SLA finanziell gestützt: Sollte eine SLA verletzt werden, wird dafür eine Entschädigung bezahlt.

[ED09] GPU Unterstützung: Ist die Verwendung von GPUs, sprich Grafikkarten, standardmäßig von der Distribution vorgesehen. Eine manuelle Bereitstellung, vorbei an der Distribution, direkt im enthaltenen Upstream Kubernetes zählt an dieser Stelle nicht.

[227] weitere Informationen unter: https://semver.org/
[228] https://kubernetes.io/docs/setup/release/version-skew-policy/
[229] weitere Informationen unter: https://github.com/cncf/k8s-conformance
[230] weitere Informationen unter: https://aws.amazon.com/de/eks/sla/
[231] weitere Informationen unter: https://cloud.google.com/kubernetes-engine/sla
[232] weitere Informationen unter: https://azure.microsoft.com/en-us/support/legal/sla/kubernetes-service/v1$_1$/

[ED10] Master Node Kosten: Es stellt sich die Frage, ob für die Master Nodes Kosten anfallen und falls ja, welche. Bei non-KaaS Distributionen werden nur Lizenzkosten betrachtet, bei Kaas Distributionen alle anfallenden Kosten.

[ED11] Verteilte Master Nodes: Hierbei ist die Frage, ob die Distribution die Verteilung von Master Nodes über mehrere Data Center oder Regionen aktiv unterstützt. Kubernetes selbst ist dafür ausgelegt, dass die Master Nodes redundant über mehrere Standorte verteilt ausgeführt sind. Diese Eigenschaft besagt jedoch, ob auch die aufbauende Distribution diese Anforderung nativ unterstützt. Für non-KaaS Distributionen ist dies irrelevant, da die zugrundeliegende Infrastruktur vom Unternehmen gestellt wird und sich die Nodes somit auch verteilt befinden könnten.

[ED12] Maximale Anzahl an Worker Nodes pro Cluster: Diese Eigenschaft sagt aus, wie viele Worker Nodes ein Cluster der vorliegenden Distribution maximal unterstützt.

[ED13] Maximale Anzahl an Pods pro Worker Node: Diese Kennzahl sagt aus, wie viele Pods maximal pro Worker Node unterstützt werden können.

[ED14] Network Plugin: Diese Eigenschaft sagt aus, welche(s) Network Plugin von der Distribution nativ unterstützt wird, bzw. werden. Dies kann, abhängig von den Unternehmensvorgaben oder technischen Anforderungen ein starkes Entscheidungskriterium darstellen.

[ED15] Network Policy Unterstützung: Hierbei ist die Frage, ob die Distribution Network Policies nativ unterstützt, oder zumindest die Installation eines Network Plugins das dies ermöglicht.

[ED16] Applikationskatalog: Es ist zwar verhältnismäßig einfach Applikationen auf ein Kubernetes Cluster zu installieren, doch die Methoden unterscheiden sich. Neben klassischen Kubernetes YAML Dateien und sogenannten HELM Charts, gibt es auch Distributionen die einen Self-Service Applikationskatalog zur Verfügung stellen. Diese Kataloge bauen oft auf HELM Charts auf, bieten jedoch eine grafische Oberfläche und teilweise erweiterte Bedienungsfeatures für den Anwender. Diese Eigenschaft beschreibt, ob die betrachtete Distribution nativ einen Applikationskatalog bereitstellt.

[ED17] Windows Worker Nodes: Kubernetes basiert auf der Container-Technologie, welche ihrerseits sehr stark mit dem Linux Kernel verbunden ist. Seit März 2019, bzw. seit Kubernetes Version v1.14[233] und Windows Server 2019 ist Kubernetes auch auf Windows verfügbar. Aus diesem Grund wird eine Kubernetes Distribution durch ihre native Unterstützung von Windows Nodes charakterisiert[234].

[ED18] CLI Zugriff: Die Hauptschnittstelle von Kubernetes ist dessen REST API. Der Zugriff darauf wird durch das Kommandozeilenprogramm *kubectl* erleichtert. Nicht alle Distributionen erlauben den Zugriff auf den Cluster via *kubectl*. Bei manchen Distributionen werden auch andere Tools für den Zugriff zur Verfügung gestellt. Diese Eigenschaft beschreibt ob, und wie nativ (*kubectl*) die Distribution den Zugriff auf den Cluster zulässt.

[ED19] Resource Monitoring: Das Monitoring von Ressourcen ist mittlerweile nahezu eine Voraussetzung für den seriösen Betrieb einer Applikationsplattform. Dabei geht es sowohl um das Monitoring der Cluster-, als auch der Applikationsressourcen. Da die gängigen Monitoringsysteme im Kern den gleichen Funktionsumfang bieten, geht es in erster Linie nicht darum welches, sondern ob generell ein Monitoringsystem nativ in der Distribution inkludiert ist. Diese Eigenschaft gibt an, ob die Distribution das Monitoring von Ressourcen nativ unterstützt. Es geht darum, dass nichts manuell installiert werden muss, nicht darum welche Lösung bereitgestellt wird.

[233] weitere Informationen unter: https://github.com/kubernetes/kubernetes/releases/tag/v1.14.0
[234] Vgl. Michael (2019), Onlinequelle [06.12.2020].

[ED20] Logging: Analog zum Ressourcen Monitoring ist es nahezu Standard, dass zeitgemäße Applikationsplattformen eine Möglichkeit bieten Applikationslogs zu sammeln, und zu verwerten. Auch hierfür sind die Lösungen meist relativ ähnlich, was den Fokus eher auf das grundlegende Vorhandensein, als auf die konkrete Implementierung des Loggingsystems legt. Diese Eigenschaft sagt aus, ob eine Distribution nativ eine Möglichkeit vorsieht mit Logs umzugehen. Es muss noch unterschieden werden, ob die Distribution imstande ist Logs zu sammeln oder diese auch noch zu speichern und zu verwerten.

[ED21] Cluster Setup Zeit/Aufwand: Wie lange es dauert einen Cluster einer bestimmten Distribution zu installieren hängt von zwei Faktoren ab:

- der technischen Durchlaufzeit der reinen Installation
- der Zeit die benötigt wird um das nötige Wissen aufzubauen

Da die zeitliche Messung dieser Eigenschaft sehr umständlich ist, basiert die Bewertung auf einer Skala von 1 bis 5. Bewertet wird vom Expertenteam welche sich für eine Distribution entscheiden muss. Dabei steht 1 für "in Sekunden", 2 bis 4 sind Abstufungen und 5 bedeutet "in Wochen".

[ED22] Cluster (Auto-)Scaling: Um Cluster sinnvoll skalieren zu können bedarf es der Fähigkeit zu erkennen, ob das bestehende Sizeing optimiert werden kann, sowie der technischen Möglichkeit dies auch zu tun. Skaliert kann grundsätzlich vertikal, durch Erhöhung oder Verringerung der Ressourcen einer Node, oder horizontal, durch das Erhöhen oder Verringern der Node Anzahl, werden. Diese Eigenschaft sagt einerseits aus, ob eine Distribution nativ imstande ist den Bedarf nach Optimierung festzustellen und andererseits, ob sie diesen auch automatisch durchführen kann.

[ED23] Worker Node Pricing: Diese Eigenschaft gibt an, wie viel der Betrieb einer Worker Node mit definierter Größe pro Jahr kostet. Die Einschätzung der Kosten für On-Premises Installationen müssen individuell von Unternehmen definiert werden.

[ED24] Authentication[235]: Grundlegend ist Kubernetes imstande nativ verschiedene externe Systeme zur Authentifizierung von Benutzern anzubinden. Hierzu gehören beispielsweise Systeme welche Authentifizierung über LDAP, SAML, OIDC oder andere Authentifizierungsprotokolle ermöglichen. Die Konfiguration zur Anbindung externer Authentifizierungssysteme ist teilweise nicht trivial[236]. Diese Eigenschaft sagt aus, ob eine Kubernetes Distribution nativ eine Möglichkeit bietet, die Einbindung eines Authentication Systems zu erleichtern.

A.10 Eigenschaften: Container-Runtime

Anhand der folgenden Eigenschaften lässt sich eine Container-Engine beschreiben und Argumente für die Wahl einer passenden Implementierung festmachen.

[ER01] Low-Level Container-Runtime: Die Container Runtimes, welche das CRI implementieren, basieren auf sogenannten *low-level Container-Runtimes*. Was in diesem Kapitel als *low-level Container-Runtime* bezeichnet wird, entspricht der Definition einer *Container-Runtime* in Kapitel 2.3.5. Diese *low-level Container-Runtimes* sind für das eigentliche Ausführen der Container zuständig. Die CRI implementierende *Container-Runtime* ist das Bindeglied zwischen *low-level Container-Runtime* und Kubernetes. Diese Eigenschaft gibt an, welche *low-level Container-Runtimes* verwendet werden können. Implizit bestimmen die *low-level Container-Runtimes* auch darüber, welche Arten von Containern verwendet werden können.

[235] weitere Informationen unter: https://kubernetes.io/docs/reference/access-authn-authz/authentication/
[236] Sayfan (2017), vlg. S. 101f.

Mittlerweile gibt es mehrere Arten von Containern, welche zwar alle die OCI Spezifikation erfüllen, technisch jedoch anders funktionieren. Beispiele hierfür sind gVisor[237], Nabla[238] oder Kata[239] Container.

[ER02] Community: Da die meisten Container Runtimes mittlerweile Open-Source Projekte darstellen, ist die Größe und Aktivität der Community eine äußerst relevante Größe. Die Anzahl an *Stars* und *Forks* auf Github, sowie die Frequenz von Releases sind gute Metriken um auf die Stärke der Community rückschließen zu können.

A.11 Eigenschaften: Kubernetes Netzwerk Plugins

Durch folgende Eigenschaften lassen sich die verschiedenen CNI Implementierungen, auch Plugins genannt, charakterisieren und unterscheiden:

[EN01] Speicherung von Konfiguration: Diese Eigenschaft gibt an wie und wo die Konfiguration, Status und andere Daten des CNI Plugins gespeichert werden. Es kann dem Nutzer, sofern kein Mehraufwand entsteht, grundsätzlich egal sein. Jedoch sei erwähnt, dass sich Plugins die ihre Daten im ectd des Clusters selbst speichern, als praktikabel erwiesen haben.

[EN02] Setup Zeit und Komplexität: Diese Eigenschaft gibt an wie komplex es ist das Plugin zu installieren und wie lange es dauert. Da es hierfür keine konkrete Maßzahl gibt, bietet es sich an die vorliegenden Alternativen zu vergleichen und gegenseitig zu bewerten. Das Ergebnis sollte eine Reihung, bestenfalls ohne mehrfach besetzte Plätze sein.

[EN03] Performance: Die Performance lässt sich am besten anhand der zu erreichenden Bandbreite, sowie zum Beispiel der Anzahl erreichter Requests pro Sekunde darstellen. Für die Bewertung der Performance wird auf Benchmarks unter genormten Bedingungen zurück gegriffen.

[EN04] Network-Policy: Das Unterstützen von *NetworkPolicy* Objekten, sprich Regeln die die Kommunikationsfähigkeit zwischen Pods einschränken können, ist ein kritisches Unterscheidungsmerkmal. Es muss noch weiter unterschieden werden ob sogenannte *Ingress*, *Egress* oder beide Regeln möglich sind. Die *Ingress* Regeln beschreiben zu einem Pod eingehenden Traffic, *Egress* Regeln beschreiben ausgehenden Traffic.

[EN05] Support: Speziell in produktiven Umgebungen im Enterprise Umfeld kann es nötig sein, kommerziellen Support beziehen zu müssen. Da viele CNI Plugins Open-Source Projekte sind, ist es nicht selbstverständlich, dass für jede Lösung kommerzieller Enterprise Support verfügbar ist. Diese Eigenschaft gibt an welche Support Möglichkeiten bestehen.

[EN06] Encryption: Die Verschlüsselung von Traffic zwischen Pods über Nodes hinweg, ist ein Feature welches nicht alle CNI Plugins unterstützen. Es wirkt sich sehr stark auf die Performance aus, kann jedoch in gewissen Umgebungen nötig sein. Diese Eigenschaft beschreibt, ob ein CNI Plugin nativ imstande ist seinen Traffic zu verschlüsseln.

[237] weitere Informationen unter: https://github.com/google/gvisor
[238] weitere Informationen unter: https://github.com/nabla-containers
[239] weitere Informationen unter: https://github.com/kata-containers

[EN07] Multi-Interface Support: Gewisse Pods können die Anforderung nach mehreren Netzwerk Interfaces haben. Standardmäßig hat ein Pod ein Interface mit einer IP Adresse. Um einem Pod mehrere Interfaces zuweisen zu können, wird ein sogenanntes Meta-Plugin benötigt, manchmal auch CNI Multiplexer genannt. Diese Plugins bedienen sich wiederum anderer Plugins, über welche die eigentlichen Interfaces für den Pod angelegt werden[240].

[EN08] Ressourcen Verbrauch: Da CNI Plugins ganz normale Programme sind welche auf jedem Node im Cluster laufen, ist deren Ressourcen Verbrauch von Interesse. Hierbei sind vor allem der Verbrauch an CPU und RAM Ressourcen zu betrachten. Da keine definitive Aussage über den genauen Verbrauch von Ressourcen gemacht werden kann, soll hierzu auf Benchmarks zurückgegriffen werden.

A.12 Eigenschaften: Ingress Controller

Dieser Abschnitt beschäftigt sich damit, die relevanten Eigenschaften und Unterscheidungsmerkmale von *Ingress Controllern* aufzuzeigen[241].

[EI01] Protokolle: Laut offizieller Kubernetes Dokumentation sind Ingress Controller typischerweise für HTTP(S) ausgelegt.[242] Moderne Systeme haben jedoch auch die Anforderung andere Protokolle wie HTTP/2, gRPC, TCP oder UDP zu verwenden. Die Protokolle, welche von den einzelnen Ingress Controllern unterstützt werden, unterscheiden sich teilweise noch grob. Aus diesem Grund stellt diese Eigenschaft ein wichtiges Unterscheidungsmerkmal dar.

[EI02] Basistechnologie: Viele Ingress Controller bauen auf denselben Basistechnologien bzw. Produkten auf und erweitern oder automatisieren diese. Obwohl die zugrunde liegende Technologie nur sekundär ist, kann es doch von Vorteil sein, zu wissen worauf die vorliegende Implementierung beruht. Vor allem im Bezug auf Weiterentwicklung, Patching und Community der Basistechnologie ist dies nicht gänzlich unerheblich.

[EI03] Routing: Die grundlegende Funktionalität eines Ingress Objektes ist es, Requests die auf einen bestimmten Pfad abzielen, dem richtigen Kubernetes Service zuzuordnen. Der Hostname des Requests, Port des Kubernetes Services oder der Umgang mit SSL/TLS sind weitere Konfigurationskriterien. Welche Möglichkeiten beim Routing von Hostname + Pfad auf Kubernetes Service geboten werden, hängt vom Ingress Controller ab.

[EI04] Verhalten mit Namespace: Die meisten modernen Ingress Controller können Requests auf alle Services im Cluster verteilen, unabhängig vom Namespace. Dies kann jedoch auch als Nachteil betrachtet werden, man könnte mit mangelnder Isolation argumentieren. Es gibt auch Ingress Controller, welche Requests nur auf Services in ihrem eigenen Namespace routen können. Dies wiederum kann als Mangel an Flexibilität und generell als unpraktikabel und nicht ressourcenschonend betrachtet werden. Es gibt auch Ingress Controller, welche sowohl global, als auch an Namespaces gebunden agieren können. Diese Eigenschaft gibt Auskunft über den sogenannten *Scope* des Ingress Controllers.

[EI05] Upstream Probes: Die sogenannten *Upstream Probes* sind eine Möglichkeit um zu überprüfen, ob der Endpunkt an den ein Request weitergeleitet werden soll, auch tatsächlich gesund und imstande ist den Request zu akzeptieren.

[240] Vgl. Schmitt (2020), Onlinequelle [06.12.2020].
[241] Vgl. Radygin/Shurupov (2019), Onlinequelle [06.12.2020].
[242] Vgl. The Linux Foundation (2020m), Onlinequelle [06.12.2020].

Dies könnte aus Sicht des Ingress Controllers als unnötig angesehen werden, da dieselbe Funktionalität für Pods durch *Liveness Probes* bzw. *Readiness Probes* erreicht werden kann. Nichtsdestotrotz ist es von Vorteil den Service vor den Pods unter realen Bedingungen überwachen, und ggf. entsprechend agieren zu können. Diese Eigenschaft sagt aus, ob und über welche eigene Upstream Probes ein Ingress Controller verfügt.

[EI06] Load-Balancing Algorithmen: Standardmäßig werden Requests einfach per *Round Robin* Verfahren an die Pods eines Services verteilt. Ist es jedoch beispielsweise nötig, dass alle Requests der Session eines Clients beim selben Pod landen, so benötigt man einen anderen Load-Balancing Algorithmus. Diese Eigenschaft sagt aus welche alternativen Load-Balancing Algorithmen der Ingress Controller unterstützt, und ob überhaupt.

[EI07] Authentication: Um den Zugriff auf die veröffentlichten Services einzuschränken, bzw. sicherzustellen, dass nur authentifizierte User Zugriff erhalten, müssen Ingress Controller Authentication Features bereitstellen. Diese Eigenschaft sagt aus, ob und welche Authentication Methoden vom Ingress Controller unterstützt werden.

[EI08] Traffic Distribution: Unter *Traffic Distribution* versteht man ein Feature, welches es ermöglicht Datenströme nach gewissen Kriterien umzulenken. Dies wird beispielsweise bei der Einführung neuer Versionen von Services genutzt, um anfangs nur wenige oder bestimmte Requests auf die neue Version umzuleiten. Diese Eigenschaft beschreibt ob und welche *Traffic Distribution* Methoden ein Ingress Controller unterstützt.

[EI09] Community: Analog zu anderen Komponenten ist auch bei der Auswahl des passenden Ingress Controllers die Community zu berücksichtigen. Die Größe der Community lässt sowohl auf eine gewisse Expertise, als auch auf die Geschwindigkeit von Entwicklung und Bugfixing schließen. Diese Eigenschaft beschreibt die Stärke der Community, ausgedrückt durch GitHub Stars, Commits und Contributors.

[EI10] Support: Analog zu anderen Komponenten ist auch für Ingress Controller die Option für kommerziellen Enterprise Support manchmal unerlässlich. Nicht alle Ingress Controller Implementierungen bieten solche Support Modelle an, weshalb dies als Unterscheidungsmerkmal gilt. Diese Eigenschaft beschreibt ob und in welchem Ausmaß Enterprise Support bezogen werden kann.

[EI11] Grafische Benutzerschnittstelle: Ähnlich dem Kubernetes Cluster selbst, ist die Konfiguration bzw. die Bedienung von Ingress Controllern in den meisten Fällen über Dateien oder per Kommandozeile möglich. Es ist jedoch auch nicht unüblich, dass gewisse Ingress Controller ein Graphical User Interface (GUI) bieten. Diese Eigenschaft sagt aus, ob ein Ingress Controller nativ eine grafische Benutzeroberfläche bietet.

[EI12] DDoS Protection: Unter Distributed Denial of Service (DDoS) Attacken wird eine Methode verstanden, mit der Systeme durch das häufig wiederholte Anfragen einer Ressource, ohne das Abwarten der Antwort, überlastet werden sollen. Manche Ingress Controller bieten Features um solche Attacken frühzeitig zu erkennen und bösartige Anfragen sofort zu blockieren. Diese Eigenschaft besagt, ob ein Ingress Controller über DDos Protection Features verfügt oder nicht.

[EI13] Preis: Der Preis eines Ingress Controllers kann durch unterschiedliche Verrechnungsmodelle bestimmt werden. Vor allem im Enterprise Segment gibt es meist keine Listenpreise mehr, daher ist diese Eigenschaft im direkten Vergleich vom bewertenden Unternehmen zu vergleichen. Es empfiehlt sich den Preis eines Ingress Controllers auf einen vergleichbaren Bezugszeitraum zu berechnen, zum Beispiel ein Kalenderjahr.

[EI14] Performance: Die Performance eines Ingress Controllers kann auf viele Arten bewertet werden. Die Anzahl von bearbeiteten Requests pro Sekunde, sowie die Latenz mit der Requests weitergeleitet werden sind zwei gängige Metriken zur Bewertung von Performance. Diese Eigenschaft gibt an wie performant ein Ingress Controller, im Hinblick auf die beiden genannten Parameter, ist. Zur Bewertung sollte auf seriöse Benchmarks zurückgegriffen werden.

A.13 Eigenschaften: Storage

Die folgenden Eigenschaften beschreiben CSI Plugins und können für einen Vergleich dieser herangezogen werden.

[ES01] Support: Speziell im Enterprise Segment und für produktive Systeme stellt Support ein wichtiges Unterscheidungsmerkmal dar. Diese Eigenschaft beschreibt ob und welche Supportmodelle verfügbar sind.

[ES02] Community: Vor allem im Bereich von Open-Source Projekten stellen die Größe und Aktivität der Community, welche das Projekt entwickelt, ausschlaggebende Kriterien dar. Die Anzahl an *Stars* und *Forks* auf GitHub, sowie die Frequenz neuer Releases sind gute Metriken um auf die Stärke der Community zurückschließen zu können.

[ES03] Performance: Performance ist bei Speicher eine wichtiges Qualitätskriterium. Die Bestimmung der Performance ist jedoch kein triviales Unterfangen. Für die Beschreibung der Performance können verschiedene Werte wie IOPS und Durchsatz verwendet werden. Zur Ermittlung dieser Werte kann auf bestehende Benchmarks zurückgegriffen werden. Es empfiehlt sich jedoch verschiedene Storage Systeme für den jeweiligen Anwendungsfall selbst zu testen, um einen realen Vergleich zu bekommen.

[ES04] Setup Zeit/Komplexität: Die Komplexität eines Systems und die Zeit zur Installation dessen, bzw. der Aufwand zur Wartung sind zwei Größen, die sich quantitativ schwer festmachen lassen. Da sie jedoch nicht außer Acht gelassen werden dürfen, empfiehlt es sich auf eine qualitative Einschätzung von Experten zurück zu greifen. Die Bewertung findet auf einer Skala von 1 ("sehr einfach") bis 5 ("höchst komplex") statt.

[ES05] Lizenz/Open-Source: Der Lizenztyp und die Verfügbarkeit des Quellcodes sind auch bei Storage Systemen ein Kriterium welches für viele Unternehmen relevant ist. Diese Eigenschaft gibt Auskunft über die Verfügbarkeit des Quellcodes und unter welcher Lizenz dieser ggf. verfügbar ist.

[ES06] Backend System: Da CSI Plugins streng genommen keine Speicher Systeme sind, sondern diese lediglich bedienen bzw. verwalten, ist es interessant zu wissen was hinter dem CSI Plugin steckt. Dieser Aspekt ist vor allem dann interessant, wenn Unternehmen schon bestehende Storage Systeme besitzen und CSI Plugins suchen welche diese Systeme verwenden können. Somit beschreibt diese Eigenschaft welches Backend System hinter dem Plugin steht oder damit verwendet werden kann.

[ES07] Storage Topology: Unter *Storage Topology* versteht man die Möglichkeit zu definieren wie die Topologie des Clusters, bzw. der Nodes und des verfügbaren Storage aussieht[243]. Damit könnte man zum Beispiel definieren, welche Nodes welche Storage Klassen verwenden können. Die Beschreibung einer Topologie von Cluster und Storage sowie Implikationen dieser Topologie ist Teil des CSI, wird jedoch noch nicht von allen Plugins

[243] weitere Informationen unter: https://kubernetes-csi.github.io/docs/topology.html

unterstützt. Diese Eigenschaft gibt an, ob das CSI Topologie Feature unterstützt wird oder nicht.

[ES08] Access Mode: Der *Access Mode* gibt an in welchem Modus und in wie vielen Pods ein Volume gemountet werden kann. Es gibt drei Möglichkeiten:

- RWO - Read Write Once: Lese- und Schreibzugriff für einen Pod
- ROX - Read Only Many: Lesezugriff für mehrere Pods
- RWX - Read Write Many: Lese- und Schreibzugriff für mehrere Pods

Diese Eigenschaft beschreibt welche *Access Modes* ein CSI Plugin unterstützt.

[ES09] Raw Block Support: Wird in der Volume Definition nicht explizit angegeben ob es sich um ein Filesystem oder Block Volume handelt, so wird per default ein Filesystem Volume angenommen.[244] Diese Eigenschaft beschreibt, ob ein CSI Plugin auch imstande ist Block Volumes bereit zu stellen und diese als Raw Block Devices zu mounten.

[ES10] Snapshot Support: Unter *Volume Snapshots* wird ein Feature verstanden, welches es ermöglicht eine Momentaufnahme eines Volumes zu erstellen. Dieses Feature funktioniert ähnlich dem Konzept von PVs und PVCs. Das bedeutet, dass diverse Snapshots erstellt und zu einem späteren Zeitpunkt von Pods angefordert und gemountet werden können.[245] Diese Eigenschaft sagt aus, ob ein CSI Plugin das Snapshot Feature unterstützt oder nicht.

[ES11] Dynamic Provisioning: Unter *Dynamic Provisioning* wird die Fähigkeit eines CSI Plugins verstanden, Volumes dynamisch auf die in einem PVC angeforderten Kriterien zugeschnitten zu erstellen. Diese Eigenschaft besagt, ob ein CSI Plugin imstande ist PVs dynamisch zu erstellen oder nicht.[246]

[ES12] Expansion: Unter *Expansion* versteht man das dynamische Vergrößern eines bestehenden PVs, sollte sich der dazugehörige PVC ändern. Diese Eigenschaft sagt aus, ob ein CSI Plugin imstande ist bestehende PVs zu vergrößern, wenn die Größe im dazugehörigen PVC erhöht wird.[247]

[ES13] Cloning: Das *Cloning* ist die Fähigkeit einen bestehenden PVC, bzw. das daran gebundene PV, zu klonen und zur Verfügung zu stellen. Diese Eingenschaft besagt ob ein CSI Plugin *Cloning* unterstützt.[248]

[ES14] CSI Version: Diese Eigenschaft gibt an, welche CSI Version vom Plugin implementiert bzw. unterstützt wird.

A.14 Eigenschaften: Service Mesh

Die folgenden Eigenschaften können als Kriterien für die Unterscheidung von Service Meshes herangezogen werden.

[EM01] Performance: Analog zu anderen Kubernetes Komponenten gilt, dass für den Vergleich von Service Mesh Performance ein Benchmark herangezogen, oder im Idealfall selbst durchgeführt werden sollte. Als Messgröße bieten sich analog dem Ingress Controller die verarbeiteten Requests pro Sekunde (RPS) sowie die auftretende Latenz an. Es ist vor allem darauf zu achten, die unterschiedlichen Perzentile bei Latenz und RPS zu betrachten.

[244] Vgl. The Linux Foundation (2020l), Onlinequelle [06.12.2020].
[245] Vgl. The Linux Foundation (2020w), Onlinequelle [06.12.2020].
[246] Vgl. The Linux Foundation (2020l), Onlinequelle [06.12.2020].
[247] Vgl. The Linux Foundation (2020v), Onlinequelle [06.12.2020].
[248] Vgl. The Linux Foundation (2020k), Onlinequelle [06.12.2020].

Eine reine Betrachtung des Durchschnitts oder Medians ist speziell mit Fokus auf User Experience nicht realistisch.

[EM02] Resource Consumption: Der Ressourcenverbrauch bezieht sich einerseits auf die *Data Plane* und andererseits auf die *Control Plane*. Auch hier gilt es auf Benchmarks zurück zu greifen, oder diese bestenfalls selbst für den konkreten Anwendungsfall durchzuführen. Die relevanten Ressourcen sind der summierte Verbrauch an RAM und CPU der *Data Plane* sowie der *Control Plane*.

[EM03] Multi Cluster: Manche Service Meshes ermöglichen es, die Services mehrerer Cluster unter einem Service Mesh zu verbinden. Dies ist beispielsweise praktisch wenn dieselben Services in mehreren Lokationen zur Verfügung stehen. Mittels Service Mesh könnten dann der Traffic intelligenter geroutet, oder eine höhere Ausfallsicherheit erzielt werden. Diese Eigenschaft sagt aus, ob ein Mesh über mehrere Cluster hinweg gespannt werden kann.

[EM04] Mesh Expansion: Es gibt Service Meshes, welche Services von außerhalb des Clusters in das Cluster Mesh integrieren können, man spricht hierbei von *Mesh Exansion*. Diese Services könnten beispielsweise auf eigenen Hosts oder einzelnen Containern betrieben werden. Diese Eigenschaft beschreibt, ob ein Mesh um externe Services ergänzt werden kann.

[EM05] Encryption/mTLS: Unter *mutual TLS*(mTLS) versteht man das Ausstellen und Verwenden eines eigenen Zertifikates für jeden Sidecar Proxy des Meshs. Damit kann die Identität eines Pods im Mesh sichergestellt werden. Des Weiteren kann mit den verteilten Zertifikaten der Datenverkehr zwischen den Pods/Services verschlüsselt werden. Besonders hierbei ist, dass beide Enden der Kommunikation ein valides Zertifikat vorweisen müssen, daher *mutual*(engl. "gegenseitig") TLS. Diese Eigenschaft besagt ob ein Mesh mTLS unterstützt.

[EM06] Externe CA: Die für mTLS verwendeten Zertifikate können entweder vom Mesh selbst verwaltet, oder von einer extern Certificate Authority (CA) bezogen werden. Diese Eigenschaft sagt aus, ob das Service Mesh die Verwendung einer externen CA für die mTLS Zertifikate unterstützt.

[EM07] Authorization Rules: Manche Meshes ermöglichen die Definition von Zugriffsregeln auf Services. Diese Regeln besagen, ob ein Service X Zugriff auf einen Service Y erhält, unter welchen Umständen und unter welchen Bedingungen. Eine Voraussetzung für eine solche Funktionalität kann mTLS sein, um die Identität des anfragenden Services sicherzustellen. Diese Eigenschaft beschreibt, ob ein Service Mesh imstande ist den Zugriff von Services untereinander zu reglementieren.
[EM08] Dashboard/GUI: Diese Eigenschaft sagt aus, ob ein Service Mesh nativ über eine GUI verfügt oder nicht.
[EM09] Tracing Integration: Unter *Tracing* oder *Distributed Tracing* versteht man das Kennzeichnen und Nachverfolgen von Requests. Dieses Feature muss vom Mesh, bzw. von den Sidecar Proxies des Service Mesh, unterstützt werden. Diese Eigenschaft besagt ob und nach welchem Standard Tracing vom Mesh unterstützt wird.

[EM10] Prometheus/Grafana: Diese Eigenschaft besagt, ob das Service Mesh nativ eine Prometheus und/oder Grafana Instanz zur Speicherung und Visualisierung von Metriken mitliefert.
[EM11] Support: Diese Eigenschaft sagt aus, ob es für das Service Mesh kommerziellen Enterprise Support gibt, oder nicht.

[EM12] License/Open-Source: Diese Eigenschaft sagt aus ob der Code für das Service Mesh Open-Source ist, bzw. mit welcher Lizenz das Mesh verwendet werden kann.

[EM13] Ingress Controller: Manche Service Meshes liefern einen eigenen Ingress Controller mit, oder sind mit bestimmten Ingress Controllern besser integriert. Diese Eigenschaft sagt aus, mit welchen Ingress Controllern das Mesh kompatibel ist und ob es mit bestimmten Ingress Controllern verbesserte Integration oder erweiterte Features gibt.

[EM14] Distribution Lock-In: Unter dieser Eigenschaft versteht man, ob das betrachtete Service Mesh nur für eine Distribution verwendbar ist. Dies ist beispielsweise bei gewissen Public-Cloud-Provider Meshes der Fall. Diese Eigenschaft ist unter anderem für den Aufbau von Wissen oder die Verwendung bestimmter Features interessant, welche nur für eine Distribution verwendbar wären.

[EM15] Erweiterbarkeit: Diese Eigenschaft sagt aus, ob es möglich ist für das Service Mesh, bzw. für die Sidecar Proxies selbst Erweiterungen und neue Funktionen zur Verfügung zu stellen.

[EM16] Setup Zeit/Komplexität: Die Komplexität eines Systems und die Zeit zur Installation dessen, bzw. der Aufwand zur Wartung, sind zwei Größen die sich quantitativ schwer festmachen lassen. Da sie jedoch nicht außer Acht gelassen werden dürfen, empfiehlt es sich auf eine qualitative Einschätzung von Experten zurück zu greifen. Die Bewertung findet auf einer Skala von 1 ("sehr einfach") bis 5 ("höchst komplex") statt.

[EM17] Maturity: Die sogenannte *Maturity* sagt aus, wie Ausgereift ein bestimmtes Produkt oder Projekt bereits ist. Nicht alle verfügbaren Service Meshes sind bereits über die *alpha* oder *beta* Phase hinaus gewachsen. Speziell für produktive Cluster ist es wichtig, dass ein Produkt eine gewisse Reife aufweist. Diese Eigenschaft zeigt die aktuellste, sowie die aktuellste stabile Version auf.

[EM18] Header/Path based Routing: Diese Eigenschaft sagt aus, ob ein Service Mesh imstande ist Routing Entscheidungen basierend auf Request Header Attributen oder dem angeforderten Request Pfad zu treffen.

[EM19] Circuit Breaker: Unter einem *Circuit Breaker* versteht man einen Mechanismus, welcher Services überwacht und bei gewissen Schwellwerten weitere Verbindungen auf das Service einstellt. Dies kann beispielsweise bedeuten, dass pro Node nur maximal n parallele Verbindungen geöffnet werden können. Eine andere Alternative wäre die Überwachung der Anzahl fehlgeschlagener Requests pro Zeiteinheit. Wird diese überschritten reagiert der Circuit Breaker, zum Beispiel mit re-routing der Requests. Diese Eigenschaft besagt, ob ein Service Mesh über *Circuit Breaker* Features verfügt oder nicht.

[EM20] Fault Injection: Unter *Fault Injection* versteht man das absichtliche Retournieren von Fehlern auf einen definierten Teil der Requests eines Service. Welche Requests einen Fehler erhalten kann je nach Service Mesh Funktionalität durch eine einfache Prozentangabe, oder durch komplexere Routingdefinitionen festgelegt werden. Diese Eigenschaft beschreibt, ob ein Service Mesh generell imstande ist definierte Fehler in einen Teil des Datenverkehrs zu injizieren.

[EM21] SMI Compatibility: Das Service Mesh Interface beschreibt zurzeit vier APIs, welche ein Service Mesh implementieren kann:

- Traffic Access Control
- Traffic Split
- Traffic Specs
- Traffic Telemetry

Für Details über die verschiedenen APIs sowie deren konkreten Inhalt sei an dieser Stelle auf die Spezifikation selbst verwiesen[249]. Diese Eigenschaft beschreibt, ob und welche API vom Service Mesh implementiert wird.
Unabhängig von den bereits genannten Eigenschaften, gibt es noch ein Set an Kriterien, welches auf Grund breiter Verbreitung als gegeben betrachtet wird. Service Meshes welche diese Aspekte nicht erfüllen, werden für eine realistische Betrachtung nicht in Erwägung gezogen. Folgende Features werden als vorausgesetzt betrachtet:

- Load-Balancing von TCP traffic
- Load-Balancing von Websocket traffic
- Load-Balancing von HTTP und HTTP/2 traffic
- Logging von Zugriffen auf Services
- Retry / Timeout Definition für Requests

[249] weitere Informationen unter: https://github.com/servicemeshinterface/smi-spec/blob/master/SPEC$_LATEST_STABLE.md$

A.15 Anforderungen: Rahmenbedingungen

Fragen deren Optionen mit einem "○" gekennzeichnet sind, lassen nur eine wählbare Option zu.
Fragen deren Optionen mit einem "□" gekennzeichnet sind, lassen mehrere auswählbare Optionen zu.

[INNO][AR01][T] Lizenz / Open-Source[250]: Software wird immer unter einer bestimmten Lizenz verwendet. Besonders bei der Verwendung von sogenannter Open-Source Software ist darauf zu achten welche Lizenz vorliegt. Einige Open-Source Lizenzen erlauben zwar die Einsicht und Veränderung des Source Codes, jedoch unter Auferlegung gewisser Pflichten. Viele Unternehmen definieren strikt unter welcher Lizenz bezogen werden muss, um diese auch verwenden zu können. Bei proprietärer Software stellt sich diese Frage nicht, da hier nur das fertige Produkt bezogen wird, der Source Code jedoch nicht ersichtlich ist.[251]

Ist die Verwendung von Open-Source Software möglich?

○ JA | ○ NEIN

Welche der folgenden Open-Source Lizenzen ist im Unternehmen zulässig? Falls nicht aufgelistet, bitte anführen.

□ GPL | □ LGPL | □ Apache 2.0 | □ BSD | □ CPL | □ BSD | □ MPL | □ EUPL | □ MIT

Lizenz: _____

[INNO][AR02][T] Lokation/Availability Zone[252]: Im Gegensatz zur Verfügbarkeit, welche angibt wie stabil ein System ist, beschreibt die Availability Zone wo das System gehostet wird. Hier wird tatsächlich die physische Lokation des Clusters gemeint. Dieser Aspekt kann entweder aus Sicht der Performance, oder aus rechtlicher Sicht interessant sein. Es ist beispielsweise ratsam den Cluster so nahe wie möglich an den Kunden zu installieren, um Latenzen zu minimieren. Bei On-Premises oder Managed-Service Clustern ist die genaue Lokation des Datacenters meist bekannt. Public-Cloud-Provider hingegen geben meist nur grobe geographische Zonen oder Länder an, in welchen die Datacenter sich befinden. Ein Beispiel für Public-Cloud-Services in einem dedizierten Land ist Azure Germany[253], der Microsoft Azure Dienst zugeschnitten auf die rechtlichen Bedürfnisse von Unternehmen am deutschen Markt.[254]

Ist es nötig, dass der Cluster sich in einem bestimmten Land, bzw. einer bestimmten Region befindet? Falls JA, spezifizieren Sie Land/Region bitte genauer.

○ JA | ○ NEIN

Erklärung: _____

[250] inspiriert durch: Kane (2018), S. 1.
[251] Vgl. Meeker (2017), Onlinequelle [06.12.2020].
[252] inspiriert durch: Buchanan (2020), S. 147f.
[253] weitere Informationen unter: https://docs.microsoft.com/en-us/azure/germany/
[254] Vgl. Platform9 Systems Inc. (2020), Onlinequelle [06.12.2020].

[INNO][AR03][T] Support[255]: Abhängig davon wie kritisch die Applikationen sind, die am Kubernetes Cluster betrieben werden, entstehen unterschiedliche Anforderungen an den verfügbaren Support. Public-Cloud-Provider, proprietäre Lösungen und Open-Source Lösungen stellen alle unterschiedliche Level von Support zur Verfügung. Die entscheidende Frage für viele Unternehmen ist, ob es grundsätzlich eine Form von professionellem, dedizierten Support gibt. Falls verfügbar sind das Ausmaß und die genauen Spezifikationen in den meisten Fällen Verhandlungssache und vom Preis bestimmt. Es ist gängige Praxis, dass vor allem große Unternehmen nur Software verwenden, für welche auch professioneller und dedizierter Support mitgekauft werden kann. Man spricht dabei von *Enterprise Support*.[256]

Wird Enterprise Support für die Kubernetes Distribution benötigt?

○ JA | ○ NEIN

Wird Enterprise Support für einzelne Kubernetes Komponenten benötigt? Fall JA, für welche?

○ JA | ○ NEIN

Erklärung: _____

[INNO][AR04][T] Test Cluster[257]: Um sinnvoll Applikationen entwickeln zu können ist es nötig diese ausgiebig zu testen. Hierfür ist es gängige Praxis, dass dedizierte Cluster für die Entwicklung und für Tests aufgesetzt werden, welche der produktiven Umgebung so ähnlich als möglich sein sollen. Da Entwicklungs- und Testcluster jedoch keinen direkten Profit erzielen, sollten die Kosten dafür umso niedriger gehalten werden. Einige Softwarehersteller bieten große Rabatte für die Verwendung von solchen nicht-produktiven Clustern, meist unter der Bedingung, dass keine produktiven Applikationen darauf betrieben werden. Durch strategische Partnerschaften mit Softwareherstellern ergeben sich oft große Rabatte oder Freilizenzen. Es durchaus aus Budgetsicht Voraussetzung sein, dass die nicht-produktiven Systeme kostenfrei oder vergünstigt sind.

Ist es eine Voraussetzung, dass kostenfreie oder vergünstigte Lösungen für Entwicklungs- bzw. Testumgebungen verfügbar sind?

○ JA | ○ NEIN

[INNO][AR05][T] Grafische Benutzerschnittstelle - Cluster[258]: Jede Kubernetes Distribution kann über das Kommandozeilenprogramm *kubectl* angesprochen werden. Viele Distributionen bieten jedoch auch die Möglichkeit einer GUI an. Aus rein technischer Sicht ist eine solche grafische Schnittstelle nicht nötig, sie erleichtert die Bedienung der Kubernetes Umgebung jedoch ungemein. Auch das Onboarding neuer Entwickler, sowie die Startphase mit der neuen Umgebung im Allgemeinen, werden durch einen GUI erleichtert.

Ist eine grafische Benutzeroberfläche für Kubernetes notwendig?

○ JA | ○ NEIN

[255] inspiriert durch: Miell (2019), S. 360.
[256] Vgl. Lamba (2017), Onlinequelle [06.12.2020].
[257] inspiriert durch betriebliche Beobachtung
[258] inspiriert durch: Luksa (2018), S. 52f.

[INNO][AR06][T] Grafische Benutzerschnittstelle - Komponenten[259]: Analog zu Kubernetes selbst, gibt es auch für viele Kubernetes Komponenten, bzw. deren Implementierungen, grafische Benutzerschnittstellen.

Ist es nötig, dass einzelne Komponenten jeweils über eine grafische Benutzerschnittstelle verfügen? Falls JA, welche Komponenten?

○ JA | ○ NEIN

Erklärung: _____

[INNO][AR07][T] rechtliche Vorgaben[260]: Gartner Inc. prognostiziert, dass bis 2022 über 50% der Enterprise Daten außerhalb des Data Centers produziert und verarbeitet werden[261]. Besonders die Verarbeitung personenbezogener Daten unterliegen strengen Vorschriften durch das europäische General Data Protection Regulation (GDPR) bzw. die österreichische Datenschutz-Grundverordnung (DSGVO). Die DSGVO/GDPR ist jedoch nur ein prominentes Beispiel für rechtliche Vorgaben im Allgemeinen.[262]

Sind die zu verarbeitenden Daten von der DSGVO/GDPR betroffen oder sind andere rechtliche Vorgaben zu berücksichtigen? Falls JA, welche?

○ JA | ○ NEIN

Erklärung: _____

[INNO][AR08][T] Linux Distribution[263]: Unterschiedliche LinuxDistributionen haben unterschiedliche Eigenschaften, Support-Modelle, Features und Tools. Je nach Unternehmen gibt es auch einen bestehenden Standard oder gewisse Vorlieben für bestimmte Distributionen.

Wird eine, oder mehrere, bestimmte Linux Distribution(en) benötigt bzw. bevorzugt? Falls JA, welche?

○ JA | ○ NEIN

Distribution: _____

[259] inspiriert durch: Luksa (2018), S. 52f.
[260] inspiriert durch betriebliche Beobachtung
[261] Vgl. Rimol (2019), Onlinequelle [06.12.2020].
[262] Vgl. Vizard (2020), Onlinequelle [06.12.2020].
[263] inspiriert durch: Kane (2018), S. 10.

[INNO][AR09][T] Strategische Allianzen[264]: Es ist sehr üblich, dass Unternehmen entweder gezielt strategische Allianzen eingehen, oder sich zumindest auf die Verwendung der Produkte eines Herstellers oder Providers fokussiert.

Gibt es eine strategische Allianz oder Partnerschaft, welche Einfluss auf die Wahl einer Kubernetes Distribution oder Komponente haben könnte? Falls JA, muss diese bei der Auswahl berücksichtigt werden.

○ JA | ○ NEIN

Erklärung: _____

[INNO][AR10][T] Public-Cloud Strategie[265]: Die Entscheidung ob die Verwendung von Public-Cloud Angeboten im Unternehmen zulässig ist, oder sogar forciert wird, spielt eine entscheidende Rolle. Es kann hierbei jedoch mehrere Gesichtspunkte geben, welche berücksichtigt werden müssen. Beispielsweise ist die Verwendung von bestehenden On-Premises Systemen möglicherweise zu bevorzugen, auch wenn die Strategie grundsätzlich Richtung Public-Cloud geht.

Ist die Verwendung von Public-Cloud Angeboten möglich oder sogar erwünscht?

○ JA | ○ NEIN

Müssen bestehende On-Premises Systeme verwendet werden können? Falls JA, welche?

○ JA | ○ NEIN

Erklärung: _____

[INNO][AR11][P] Verfügbarkeit der Daten: Je nachdem für welchen Use-Case ein Cluster verwendet wird, kann es kritische Auswirkungen haben wo sich die zu verarbeitenden Daten befinden. Dies kann einerseits an der puren Menge der Daten liegen, welche einen ökonomischen Transfer schlichtweg ausschließen würden. Andererseits kann es jedoch auch gewisse rechtliche oder strategische Vorgaben geben, welche bestimmen wo die Daten verarbeitet werden müssen.

Gibt es Vorgaben oder Umstände welche die Wahl der Lokation des Clusters, basierend auf den zu verarbeitenden Daten, einschränken? Falls JA, welche?

○ JA | ○ NEIN

Erklärung: _____

[264] inspiriert durch strategische Allianzen im eigenen Unternehmen
[265] inspiriert durch das steigenden Public-Cloud Angebot

[INNO][AR12][P] Politische Vorgaben/Herstellerstandort: Der Standort eines Herstellers kann politisch ein Problem darstellen. Speziell bei Unternehmen die mit Behörden oder Staaten zusammenarbeiten ist dieser Aspekt vermehrt zu berücksichtigen. Es kann jedoch auch strategische Vorgabe sein, Unternehmen bestimmter Staaten oder Regionen zu bevorzugen. Diese Vorgaben müssen nicht immer rational sinnvoll sein, jedoch unbedingt berücksichtigt werden.

Gibt es Vorgaben bzgl. des Herstellerstandortes, oder politischer Natur, die berücksichtigt werden müssen? Falls JA, welche?

○ JA | ○ NEIN

Erklärung: _____

[INNO][AR13][P] ausgelagerter Betrieb und Support: Der Betrieb eines Kubernetes Systems beschränkt sich nicht nur auf den technischen Aspekt Es wird im Normalfall auch gefordert, eine gewisse Form von Support für das System zur Verfügung zu stellen. Dabei ist tatsächlich der Anwendersupport, zum Beispiel in Richtung der Entwickler, gemeint. Vor allem Managed-Service-Anbieter stellen dafür teilweise Modelle zur Verfügung, welche es erlauben Support auszulagern. Es gibt auch spezialisierte Unternehmen, welche ausschließlich Support, meist zugeschnitten auf bestimmte Produkte und Kubernetes Distributionen, bieten.

Ist es nötig den Betrieb inklusive Support des Clusters auszulagern?

○ JA | ○ NEIN

[TECH][AR14][P] Single-Node-Kubernetes: Manche Aufgabenstellungen erfordern es, dass ein gesamter Kubernetes Cluster auf nur einem Host läuft. Man spricht in solchen Fällen von sogenannten "Single-Node-Kubernetes" Distributionen. Diese könnten beispielsweise auf einzelnen Maschinen in entlegeneren Lokationen, oder lokal auf Entwicklermaschinen benötigt werden. Sollte eine solche Distribution benötigt werden, engt dies das Feld an verfügbaren Optionen drastische ein.

Wird eine "Single-Node-Kubernetes" Distribution benötigt?

○ JA | ○ NEIN

[INNO][AR15][P] Long-Term-Support: Im Gegensatz zu [AR03] geht es bei dieser Anforderung nicht um die ausschließliche Verfügbarkeit von Enterprise Support. Es wird betrachtet ob für die vorliegende Software, in einer bestimmten Version, langfristig Support verfügbar ist. Typischerweise spricht man hier von sogenannten LTS Versionen, welche meist mehrere Jahre lang Updates und Support erhalten. Unter anderem um nicht mehrmals pro Jahr updaten zu müssen, um im unterstützten Supportrahmen zu bleiben, setzen manche Unternehmen LTS voraus.

Wird LTS benötigt?

○ JA | ○ NEIN

[INNO][AR16][P] Vendor Lock-In: Wenn die Verwendung einer Software, oder mehrere Softwareprodukte eines Herstellers, ein Unternehmen an einen Hersteller binden, spricht man von Vendor Lock-In. Dieser Lock-In kann auf Grund technische Hürden beim Wechsel zu anderen Technologien, bestehender Verträge, strategischer Allianzen oder anderen Ursachen entstehen. Wichtig ist bei diesem Punkt zu beachten, ob die Verwendung einer betrachteten Software zu einem potenziellen Vendor Lock-In führen kann. Sollte dies der Fall sein, muss entschieden werden ob dies ein Problem darstellt. Entscheidet man sich beispielsweise für eine langfristige strategische Kooperation mit eine Hersteller ist dagegen nichts einzuwenden. Achtet man darauf, dass ein solcher Lock-In nicht entsteht, nimmt dies natürlich die potenziellen Vorteile einer engen geschäftlichen Bindung. Andererseits hält man sich die Option offen auf andere Produkte, oder zu anderen Herstellern, zu wechseln.

Muss darauf geachtet werden, dass bei der Produktauswahl explizit kein Vendor Lock-In entsteht?

○ JA | ○ NEIN

A.16 Anforderungen: Funktionale Anforderungen

[INNO][AF01][T] Redundanter Speicher[266]: Redundante Datenspeicherung ermöglicht es Daten wiederherzustellen, sollte ein Speicherelement beschädigt werden. Für redundante Datenspeicherung fallen höhere Kosten als für nicht redundante Speicherung an. Speziell auf Entwicklungssystemen wird oft auf Redundanz verzichtet.

Müssen die Daten redundant gespeichert werden?

○ Redundanz benötigt | ○ Keine Redundanz benötigt

[INNO][AF02][T] Verschlüsselter Speicher[267]: Aus Gründen der Datensicherheit sowie rechtlicher Bestimmungen kann es nötig sein, gespeicherte Daten zu verschlüsseln. Man spricht hier von *data at rest encryption*.

Müssen persistierte Daten verschlüsselt werden?

○ JA | ○ NEIN

[INNO][AF03][T] Storage Backups[268]: Werden Daten nur für Tests und Entwicklung benötigt, so wäre deren Verlust wahrscheinlich verkraftbar. In produktiven Umgebungen hingegen ist ein Datenverlust meist nicht in kauf zu nehmen. Ein Möglichkeit um Datenverlust vorzubeugen sind Storage Backups.

Werden Storage Backups benötigt?

○ JA | ○ NEIN

[266] inspiriert durch: Buchanan (2020), S. 146.
[267] inspiriert durch: Buchanan (2020), S. 129.
[268] inspiriert durch eigene betriebliche Praxis

[INNO][AF04][T] Grafikkarten Unterstützung[269]: Speziell für *AI* oder *Machine Learning* Anwendungen wird besondere Hardware benötigt. Konkret werden Grafikkarten verwendet, da diese Berechnungsaufgaben, wie sie in *Machine Learning* Anwendungen zu finden sind, sehr performant lösen können. Da Grafikkarten in Servern jedoch eher unüblich sind, stellen diese eine besondere Anforderung dar.

Werden am Cluster *AI* oder *Machine Learning* Anwendungen betrieben?

○ JA | ○ NEIN

[INNO][AF05][T] Betriebssystem[270]: Container müssen immer mit dem Betriebssystem ihres Hosts zusammenpassen. Daher muss entschieden werden ob Linuxund/oder Windows Container verwendet werden können sollen. Diese Entscheidung ist essentiell, da Unternehmen sehr oft eine strategische Ausrichtung bezüglich ihres Technologie-Portfolios haben. Außerdem sei an dieser Stelle angemerkt, dass Windows Container, sowie das gesamte Ökosystem darum, noch weit nicht so ausgereift sind wie LinuxContainer.

Container basierend auf welcher Art von Betriebssystem sollen unterstützt werden?

□ Windows | □ Linux

[INNO][AF06][T] Costing[271]: Um die verbrauchten Ressourcen eines Clusters sauber verrechnen, oder zumindest eine kostenmäßige Verteilung erheben zu können, wird ein Costing Feature benötigt. Mit dieser Funktionalität ist es möglich detailliert zu erheben welche Applikation, und somit welche organisatorische Kostenstelle oder welcher Kunde, wie viele Ressourcen konsumiert hat.

Wird eine Funktionalität benötigt, die es erlaubt verbrauchte Ressourcen zu analysieren und monetär darzustellen?

○ JA | ○ NEIN

[INNO][AF07][T] Inter-Pod Kommunikation[272]: Um Anforderungen wie erhöhter Security oder *Multi Tenancy* gerecht werden zu können, muss es möglich sein die Kommunikation zwischen Pods zu reglementieren. Diese Anforderung wird durch kein natives Kubernetes Feature erfüllt. Es gibt zwei Ansätze welche eine Regulierung bzw. Unterbindung der Kommunikation zwischen Pods erlauben, diese stellen jedoch einen gewissen Mehraufwand dar. Ein Beispiel für eine solche Notwendigkeit wäre es, mehreren unterschiedlichen Kunden Zugriff auf unterschiedliche Namespaces eines Clusters zu geben. In diesem Fall sollten auch die Pods der jeweiligen Namespaces nicht miteinander Kommunizieren können.

Ist es nötig die Kommunikation zwischen bestimmten Pods zu unterbinden?

○ JA | ○ NEIN

[269] inspiriert durch: Luksa (2018), S. 73f.
[270] inspiriert durch: Buchanan (2020), S. 38.
[271] inspiriert durch eigene betriebliche Praxis
[272] inspiriert durch: Sayfan (2019), S. 167ff.

[INNO][AF08][T] Metrik-System[273]: Metriken sind eine sehr gute Möglichkeit um den aktuellen Status des Clusters, sowie der darin laufenden Applikationen, zu erheben. In vielen Unternehmen haben sich schon vor entstehen der ersten Kubernetes Cluster zentrale Metrik Systeme etabliert. Aufbauend auf Metriken lassen sich auch Alarme auslösen, sollte etwas mit dem Cluster nicht stimmen.

Wird ein neues, cluster-internes System zur Sammlung von Metriken, sowie zur Verwaltung von Alarmen, benötigt?

○ JA | ○ NEIN

[INNO][AF09][T] Logging-System[274]: Das Sammeln und Auswerten von Log Daten ist ein wichtiger Punkt für den Betrieb, das Troubleshooting, sowie Security und rechtliche Belange. Für ältere Systeme gibt es in vielen Unternehmen schon zentrale Systeme die Logs sammeln und aufbereiten bzw. analysieren.

Wird ein neues, cluster-internes System zur Sammlung und Auswertung von Log Daten benötigt?

○ JA | ○ NEIN

[INNO][AF10][T] Alerting-System[275]: Aufbauend auf Metriken und Log Daten gibt es sogenannte *Tracing Systeme*. Damit lassen sich Fehler, bzw. deren Entstehen und Ausbreitung, durch mehrere Applikationen hindurch verfolgen. Eine fehlerhafte Anfrage zurückzuverfolgen und deren Ursprung herauszufinden wäre ein Beispiel für den Anwendungsfall eines Tracing Systems. Auch hier gilt, dass es bereits schon vor der Entstehung des Clusters solche Systeme geben kann.

Wird ein neues, cluster-internes System zum Tracing benötigt?

○ JA | ○ NEIN

[TECH][AF11][T] Multiple Netzwerkschnittstellen[276]: Gewisse Applikationen, welche in Containern im Cluster laufen, könnten ggf. zwei oder mehrere Netzwerkschnittstellen benötigen. Etwa für eine dem Cluster zugewandte API, sowie zum Zugriff auf ein System mit hohem Netzwerkdurchsatz. Hierfür könnten einem Container zwei oder mehrere Netzwerkinterfaces gegeben werden, welche eventuell sogar 1:1 mit physikalischen Interfaces verbunden sind. Man könnte beispielsweise einen Datenbank-Container betreiben der an einem Interface seinen Service im Cluster zur Verfügung stellt. Das zweite Interface wäre direkt und exklusiv mit einem physikalischen Interface verbunden, welches Zugriff auf ein Netzwerk-Storagesystem hat. Somit hätte der Container eine sehr schnelle Storage-Anbindung für seine Daten, und ein normales Interface für seinen Service in Richtung Cluster.

Werden für gewisse Container im Cluster mehrere Netzwerkschnittstellen benötigt?

○ JA | ○ NEIN

[273] inspiriert durch: Buchanan (2020), S. 136.
[274] inspiriert durch: Buchanan (2020), S. 138ff.
[275] inspiriert durch eigene betriebliche Praxis
[276] inspiriert durch eigene betriebliche Praxis

[TECH][AF12][T] Service-Protokolle[277]: Je nachdem welche Art von Applikationen auf dem Kubernetes Cluster betrieben werden, bzw. welche Services von außerhalb des Cluster erreichbar sein sollen, werden andere Zugriffsprotokolle verwendet. Grundsätzlich ist es wichtig zu wissen, ob alle Services am Cluster via HTTP(S) erreicht werden können. Aufbauend darauf stellt sich die Frage ob auch HTTP der Version 2, oder gRPC, verwendet wird. Sollte dies nicht der Fall sein, wenn zB Datenbanken oder Messaging-Systeme vorhanden sind, so muss noch unterschieden werden ob diese Anwendungen Protokolle basierend auf UDP oder TCP verwenden.

Über welche Protokolle wird von außen auf Services am Cluster zugegriffen?

□ HTTP(S) | □ HTTP/2 / gRPC | □ TCP | □ UDP

Ergänzung: _____

[TECH][AF13][T] Service Mesh[278]: Speziell wenn viele Mikroservices auf einem Cluster betrieben werden, ist es oft nötig diese besonders zu überwachen. Außerdem kann es nötig sein den Datenfluss zwischen den Pods zu visualisieren, umzuleiten oder zu verschlüsseln. All diese Funktionalitäten werden von einem Service Mesh zur Verfügung gestellt. Ein solches Service Mesh stellt jedoch auch einen gewissen betrieblichen und Installationsaufwand dar.

Wird ein Service Mesh benötigt?

○ JA | ○ NEIN

[TECH][AF14][T] User Management[279]: Die meisten Unternehmen verwenden für die Verwaltung ihrer IT User ein zentrales System wie beispielsweise Microsoft Active Directory. Diese Systeme werden entweder direkt abgefragt, oder von Drittsystemen als backend System genutzt. Speziell für User login (Authentication und Authorization) wird in den meisten Fällen auf ein zentralen System zur Userverwaltung zurückgegriffen. Kubernetes verwaltet seine User intern, jedoch gibt es auch Möglichkeiten Kubernetes mit einem zentralen User Management System zu koppeln. Grundsätzlich bietet es sich immer an ein zentrales User Management System anzubinden, sobald man viele verschiedene User auf einem Cluster hat. Vor Allem auf größeren Entwicklungssystemen oder produktiven Systemen ist eine solche Verbindung sinnvoll.

Wird eine Verknüpfung des zentralen User Management Systems mit Kubernetes benötigt?

○ JA | ○ NEIN

[TECH][AF15][T] Speicher[280]: Nicht-Flüchtiger bzw. persistenter Speicher wird für die Sicherung von Daten über die Lebenszeit eines Containers hinaus benötigt. Flüchtiger Speicher wird wieder freigegeben nachdem der Container beendet wurde. Anwendungen, wie zB Datenbanken, welche Daten auch über einen eventuellen Container-Crash hinaus speichern sollen, brauchen beispielsweise solchen Speicher. Sogenannte *stateless* Anwendungen, beispielsweise statische Webseiten, benötigen diesen Speicher nicht. Jedem Container steht flüchtiger, sogenannter *ephemeral* Speicher zur Verfügung, welcher jedoch mit dem Ende des Containers verschwindet.

[277] inspiriert durch: Luksa (2018), S. 134ff.
[278] inspiriert durch: Sayfan (2019), S. 412ff.
[279] inspiriert durch: Buchanan (2020), S. 122f.
[280] inspiriert durch betriebliche Beobachtung

Wird persistenter Speicher benötigt, oder reicht flüchtiger Speicher?

○ persistenter Speicher wird benötigt | ○ persistenter Speicher wird nicht benötigt

[TECH][AF16][T] Object Storage[281]: Aufbaufrage zu [AF01]
Sogenannter *Object Storage* wird meist als Webservice bezogen und eignet sich besonders für Daten die nicht häufig geändert, jedoch oft gelesen werden. Die Verwendung wäre zum Beispiel bei statischen Webseiten, Backups, Multimediadateien etc. sinnvoll. Anwendungen die viele Speicherzugriffe haben sollten von dieser Speicherart eher absehen. Object Storage hat den Vorteil, dass er verhältnismäßig günstig und nahezu beliebig erweiterbar ist. Es wurde in Kapitel 3.5 bereits angemerkt, dass Object Storage in dieser Arbeit nicht näher betrachtet wird. Allerdings kann die Verwendung von Object Storage, eines beliebigen Anbieters, in der richtigen Applikationsarchitektur enorme Vorteile bringen.

Wird *Object Storage* benötigt?

○ JA | ○ NEIN

[TECH][AF17][T] File Storage[282]: Aufbaufrage zu [AF01]
Diese Speicherart ist allgemein von Desktop Rechnern bekannt und stellt gewissermaßen den Standard-Speicher in Kubernetes Systemen dar. Sogenannter *File Storage* wird verwendet um Daten strukturiert in einem Filesystem abzulegen. Das bedeutet, dass Features wie Ordnerstrukturen, "soft delete" (sprich ein Papierkorb) und "concurrent operations" (paralleles zugreifen auf dieselben Dateien) verwendet werden können. Dieser Speicher wird verwendet, wenn Applikationen regelmäßige Speicherzugriffe tätigen, die ein Filesystem voraussetzen. Sofern eine Applikation nicht imstande ist ihren Speicher selbst zu organisieren, sprich Block Storage zu verwenden, verwendet sie im Normalfall File Storage.

Wird *File Storage* benötigt?

○ JA | ○ NEIN

[TECH][AF18][T] File Storage Zugriff[283]: Aufbaufrage zu [AF05]
Sollen mehrere Pods gleichzeitig Zugriff auf denselben File-Storage haben, so muss dies das Storage-System auch unterstützen. Dies kann beispielsweise nützlich sein, um Dateien einfach unter den Pods auszutauschen.

Ist es nötig, dass sich mehrere Pods denselben File-Storage teilen?

○ JA | ○ NEIN

[281] inspiriert durch eigene betriebliche Praxis
[282] inspiriert durch: Luksa (2018), S. 171ff.
[283] inspiriert durch: Sayfan (2017), S. 161.

[TECH][AF19][T] Block Storage[284]: Aufbaufrage zu [AF01]
Sogenannter Block Storage stellt die Basis für die Filesysteme des File Storage und Object Storage dar. Diese Speicherart kann verwendet werden um darauf diverse Filesysteme zu installieren. Außerdem wird Block Storage von Applikationen, wie beispielsweise manchen Datenbanken, direkt verwendet, da diese Speicherart erheblich schneller ist als File- oder Object Storage. Applikationen müssen jedoch imstande sein Block Storage direkt zu verwenden, um den Geschwindigkeitsvorteil auch ausnutzen zu können.

Wird Block Storage benötigt?

○ JA | ○ NEIN

[TECH][AF20][T] Cluster Scaling[285]: Um mit maximaler Flexibilität auf sich ändernde Ressourcenanforderungen eingehen zu können, kann es möglich sein nicht nur die Pods, sondern den gesamten Cluster zu skalieren. Das bedeutet, dass der Cluster imstande ist, weitere Worker Nodes hinzuzufügen, diese zu entfernen oder ggf. deren Ressourcen zu ändern. Man spricht in diesem Fall von *Cluster Autoscaling*. Dieses Feature erlaubt es ebenfalls die Ausnutzung von Ressourcen auf den Worker Nodes zu optimieren. Dazu werden Workloads auf Nodes gezogen die noch freie Ressourcen haben, leere Nodes werden entfernt und überdimensionierte Nodes werden verkleinert. Diese Funktionalität wird von manchen Systemen standardmäßig zur Verfügung gestellt, kann bei anderen Systemen nachinstalliert werden und ist manchmal gar nicht verfügbar. Auch der Grad der Automatisierung unterscheidet sich. Bei manchen Lösungen kann ein gewisser manueller Aufwand nötig sein, um den Cluster zu Skalieren. Diese Funktionalität ist speziell bei Anwendungen mit sehr volatilem Lastaufkommen, oder sehr großen Clustern von Vorteil.

Wird eine Form der mehr oder weniger automatisierten Cluster Skalierung benötigt?

○ JA | ○ NEIN

Erklärung: _____

[TECH][AF21][P] verschlüsselte Kommunikation: Es besteht grundsätzlich die Möglichkeit die Kommunikation zwischen Pods auf einem Cluster zu verschlüsseln. Dies kann beispielsweise nötig sein um rechtliche Vorschriften zu erfüllen. Auch der Betrieb eines Cluster der mehrere Mandanten bedient würde eine Solche Anforderung mit sich bringen. Verschlüsselung kommt jedoch meist mit einem erheblichen Verlust von Performance bzw. Bandbreite.

Ist es nötig die Kommunikation zwischen Pods zu verschlüsseln?

○ JA | ○ NEIN

[284] inspiriert durch: Sayfan (2017), S. 170f.
[285] inspiriert durch: Buchanan (2020), S. 105f.

[TECH][AF22][P] Backup/Restore: Backups sind, speziell in produktiven Systemen, ein essentieller Teil einer seriösen Plattform. Sowohl Kubernetes selbst, als auch dessen Komponenten müssen periodisch gesichert werden. Nur so kann gewährleistet werden, dass im Fall eines Ausfalls das System auf einen funktionierenden Stand zurückgesetzt werden kann. In diesem Fall spricht man von einem "Restore". Es ist grundsätzlich möglich sich selbst um die Sicherung aller relevanten Komponenten zu kümmern, jedoch gibt es auch Produkte oder Distributionen die solche Funktionalitäten out-of-the-box anbieten. Speziell die Bedienbarkeit ist bei solchen Lösungen um ein vielfaches vereinfacht, da sie gerne auch mit einer grafischen Benutzeroberfläche konfiguriert werden können.

Ist es nötig, dass das Produkt bzw. die Kubernetes Distribution ein Backup/Restore System out-of-the-box mitliefert?

○ JA | ○ NEIN

[TECH][AF23][P] Integration in bestehende Infrastruktur: Häufig herrscht im Unternehmen schon eine gewisse Systemlandschaft, in welche der neue Cluster bzw. das betrachtete Produkt integriert werden soll. Dies könnten beispielsweise bestehende Netzwerk- oder Storagesysteme, Virtualisierungslösungen oder Verwaltungstools sein. Auch Logging-, Alerting- und Metriksysteme sind häufig schon in Verwendung und sollen an die neue Software angebunden werden.

Gibt es bestehende Infrastruktur, in welche sich der neue Cluster bzw. das neue Produkt integrieren lassen muss? Falls JA, welche?

○ JA | ○ NEIN

Erklärung: _____

[INNO][AF24][P] Sprache der Dokumentation und Benutzerschnittstelle: Je nachdem in welche Land sich das eigene Unternehmen und das Herstellerunternehmen befinden, gibt es oft sprachliche Unterschiede. Zwar ist Englisch mittlerweile als Standardsprache weltweit etabliert, in manchen Situationen kann es jedoch immer noch hilfreich oder nötig sein weiteren Sprachen zu unterstützen. Sowohl die vorliegende Dokumentation, als auch die Benutzerschnittstelle (grafisch oder textbasiert), könne Anforderungen an die Sprache stellen.

Müssen andere Sprachen, außer Englisch, unterstützt werden? Falls JA, welche?

○ JA | ○ NEIN

Erklärung: _____

[TECH][AF25][P] WAF Funktionalität für Ingress: Manche Ingress Controller bieten erweiterte Sicherheitsfunktionen. Dabei spricht man von sogenannten Web Application Firewalls, kurz WAF. Damit kann der Datenverkehr auf effektiv auf Gefahren untersucht, und entsprechend reagiert werden. Dies ist vor allem sinnvoll, wenn der Cluster öffentlich zugänglich ist und man nicht weiß wer auf die Cluster-Services zugreift. Solche Funktionalitäten können gegebenenfalls die Performance einschränken und Kosten verursachen.

Werden zusätzliche Sicherheitsfunktionen am Ingresscontroller benötigt?

○ JA | ○ NEIN

A.17 Anforderungen: Qualitätseigenschaften

[AQ01][T] Upgrade Aufwand[286]: Kubernetes Cluster, bzw. deren Bestandteile, müssen von Zeit zu Zeit auf die aktuellste, oder zumindest eine zeitgemäße, Version aufgerüstet werden. Je nach Distribution, Komponente und Implementierung erfordert dies mehr oder weniger manuelle Arbeit und dementsprechend Aufwand. Auch der Grad an Komplexität, beim Durchführen des Upgrades, ist für manche Cluster Setups nicht zu vernachlässigen. Hauptunterscheidungsmerkmal ist bei diesem Punkt die Einfachheit und der manuelle Aufwand für ein Upgrade. Die Zeit zur Vorbereitung und Durchführung des Upgrades ist eine geeignete Maßgröße, je weniger Zeit benötigt wird desto besser.[287]

[AQ02][T] Verfügbarkeit[288]: Unter der Verfügbarkeit wird in diesem Kontext die garantierte Uptime des Clusters an sich verstanden. Damit sind nicht die einzelnen Worker Nodes gemeint, sondern die durchgehende Erreichbarkeit der Control Plane, sprich der Kubernetes API. Diese wird meist durch ein Service Level Agreement abgesichert und als Prozentsatz in Verfügbarkeit pro Zeiteinheit angegeben, meist pro Monat. Kubernetes Distributionen die von Public-Cloud-Providern, sowie von Managed-Service-Providern angeboten werden garantieren fixe Werte für die Verfügbarkeit. Bei On Premises Installationen hingegen ist das Unternehmen selbst dafür verantwortlich die Verfügbarkeit zu gewährleisten. Je höher die prozentuale Verfügbarkeit pro Zeitintervall ist, desto besser.[289]

[286] inspiriert durch: Buchanan (2020), S. 103.
[287] Vgl. Klein (2019), Onlinequelle [06.12.2020].
[288] gängige betriebliche Metrik
[289] Vgl. Berger (2020), Onlinequelle [06.12.2020].

[AQ03][T] Kosten: Die Kosten für einen Kubernetes Cluster lassen sich grundsätzlich in drei grobe Gruppen unterteilen:

- Control Plane Kosten
- Worker Node Kosten
- Kosten pro Komponente

Da die Kosten eine generelle Qualitätseigenschaft darstellen, werden die drei Kostenarten unter einen Überpunkt zusammengefasst.
Grundsätzlich gilt, dass die Kosten in Geldeinheiten pro Zeiteinheit, meist ein Jahr, angegeben werden. Je weniger ein Cluster oder Produkt kostet desto besser.

Kosten - Control Plane: Dass die Worker Nodes, auf denen die eigentliche Workload des Clusters läuft, bezahlt werden müssen ist selbstverständlich. Dabei geht es nicht nur um die verwendete Hardware, sondern auch um etwaige Lizenzen. Anders verhält es sich jedoch bei der Control Plane, sprich bei den Master Nodes. Abhängig von der gewählten Distribution können für die Control Plane unterschiedlich hohe, oder auch keine, Gebühren anfallen. Bei manchen Public-Cloud oder Managed-Service-Providern entfallen die Control Plane Gebühren beispielsweise gänzlich. Allerdings sind bei On-Premises Installationen wiederum auch sämtliche Infrastrukturkosten der Master Nodes mit einzuberechnen.

Kosten - Worker Nodes: Die Kosten für Kubernetes Worker Nodes setzen sich je nach betrachteter Distribution anders zusammen. Vor allem Public-Cloud Anbieter verrechnen meist einfach die verbrauchten Ressourcen, ohne weitere Lizenzaufschläge oder dergleichen. Bei proprietären On-Premises, oder auch Cloud, Installationen kommen zur verwendeten Hardware oft noch Lizenzkosten für die Kubernetes Distribution selbst hinzu. Es können außerdem noch weitere Kosten für das verwendete Betriebssystem, etwaige Virtualisierungen oder andere Größen wie zum Beispiel die Anzahl der User am System hinzukommen. Vernachlässigt dürfen an dieser Stelle auch erweiterte Infrastrukturkosten wie Storage, Backups oder Netzwerk nicht werden. Es bietet sich für Worker wie für Master Nodes an, die Kosten detailliert mit Unterstützung der IT oder des Controllings zu berechnen.

Kosten - Cluster Komponenten: Neben den Kosten für die Worker und Master Nodes, können auch für die diversen anderen Komponenten des Clusters Kosten anfallen. Wichtige Faktoren sind hierbei meist die Lizenzkosten und etwaige Supportkosten. Da es bei dieser Kategorie keine gängige Regel gibt, sei an dieser Stelle auf die Angaben der jeweiligen Produkte und Hersteller verwiesen.

[AQ04][T] Personeller Aufwand[290]: Je nachdem welche Kubernetes Distribution oder Komponente betrachtet wird, fällt mehr oder weniger personeller Aufwand für Wartung, Betrieb und Weiterentwicklung an. Es ist an dieser Stelle nahezu unmöglich das genaue Ausmaß an anfallenden Aufwänden abzuschätzen. Jedoch kann festgehalten werden, dass Kubernetes On Premises Installationen erheblich mehr betrieblichen Aufwand verursachen als Distributionen welche als Service bezogen werden. Es gibt auch Mischformen von Clustern, wie beispielsweise die manuelle Installation auf Cloud Infrastruktur, bei welcher zumindest der Infrastrukturanteil vom Public-Cloud-Provider übernommen wird. Wie viel Aufwand tatsächlich anfällt ist fallspezifisch zu beurteilen, die Tatsache an sich muss jedoch mitberücksichtigt werden. Als Messgröße empfiehlt sich bei diesem Kriterium die Anzahl an Mitarbeitern die vollständig mit dem Produkt oder der Distribution beschäftigt sind.

[290] inspiriert durch betriebliche Beobachtung

[AQ05][T] Marktanteil[291]: Wie groß der derzeitige Marktanteil einer Kubernetes Distribution, oder der Implementierung einer Komponente, ist, hat eine nicht zu vernachlässigende Relevanz. Zwar sagt der Marktanteil nicht zwangsläufig etwas über die Qualität aus, hat jedoch sehr wohl Auswirkungen auf den weiteren wirtschaftlichen Erfolg. Speziell für KMUs stellt der Marktanteil ihres Produktes eine essentielle Größe dar. Dies gilt natürlich ebenfalls für Open-Source Projekte, denn auch deren Erfolg ist direkt an ihren Verbreitungsgrad gekoppelt. Der Marktanteil kann als Prozentwert aus Marktforschungen oder Studien bezogen werden, je größer desto besser.

[AQ06][T] Communityinspiriert durch: [0]: Neben dem Marktanteil stellt auch die Community hinter einem Produkt oder Projekt ein grundlegendes Entscheidungskriterium dar. Im speziellen Open-Source Projekte leben von der Community welche diese stetig weiterentwickelt und optimiert. Proprietäre Produkte haben das Problem, dass sie nur eine begrenzte Anzahl an aktiven Entwicklern haben können. Jedoch gilt auch hier, dass eine große Community um das Produkt vorteilhaft ist, auch wenn diese hierbei nur eine Nutzer- und Feedbackfunktion hat. Die Größe der Community muss nicht zwangsläufig mit dem Marktanteil korrelieren. Gemessen wird die Größe der Community gerne an der Anzahl aktiver Nutzer in Foren oder anhand der Aktivität auf Open-Source Plattformen wie GitHub.

[AQ07][T] Enterprise Experience[292]: Je mehr Erfahrung ein Unternehmen in seiner Geschichte gesammelt hat, und je besser es diese einsetzen kann, desto bessere Produkte wird es anbieten können. Dies wirkt auf den ersten Blick selbstverständlich, muss jedoch bei der Bewertung von Produkten aktiv berücksichtigt werden. Da Kubernetes selbst noch eine verhältnismäßig neue Technologie darstellt, sind auch die Unternehmen auf diesem Markt oft noch sehr jung. Dies muss nicht zwangsläufig schlecht sein, es muss jedoch berücksichtigt werden auf welchen Erfahrungsschatz ein Unternehmen zurückgreifen kann. So können beispielsweise junge Unternehmen, welche aus erfahrenen Mitarbeitern bestehen, bessere Ergebnisse erzielen als alte Unternehmen die sich unbeholfen in einem neuen Markt versuchen wollen. Zum Vergleich muss bei diesem Kriterium auf ein subjektives Abschätzen der betrachteten Unternehmen vertraut werden. Telefonate und Meetings sowie Kontakt mit Referenz- und Bestandskunden können die Einblicke ins Unternehmen verbessern.

[AQ08][T] Time-to-get-Started[293]: Unter der "time to get started" versteht man einerseits die Zeit die es dauert, bis ein Produkt genutzt werden darf, andererseits die Zeit bis es verwendet werden kann. Das bedeutet, dass sowohl die Zeit für Kauf, Anmeldung, Lizenzbereitstellung etc. eine Rolle spielt. Auch jedoch die Zeit vom Start der Installation bis zum fertig einsetzbaren Produkt. Dieser Punkt lässt sich nicht konkret quantifizieren, doch gilt auch hier, dass er berücksichtigt werden muss. Beispielsweise sind Open-Source Produkte schneller zu beziehen, Public-Cloud Lösungen meist jedoch wesentlich schneller einsatzbereit wenn erst einmal ein Zugang vorliegt. Der direkte Vergleich für diese Eigenschaft lässt sich nur durch Schätzungen, Erfahrungswerte oder bereits durchgeführte Installationen durchführen.

[AQ09][T] benötigtes Know-How[294]: Der Punkt Know-How ist ein kritischer Faktor für die Auswahl einer zukunftsträchtigen Lösung. Hierbei stellt sich die Frage wie viel Know-How benötigt wird um das Produkt initial nutzen zu können, aber auch wie viel Know-How für einen seriösen Betrieb nötig ist. Für Know-How gibt es keine Maßgröße, doch es ist für die zuständigen Ingenieure grundsätzlich möglich abzuschätzen ob eine Lösung mehr oder weniger Know-How bedarf als andere. Speziell bei den Managed-Service-Angeboten ist das nicht benötigte Know-How ein starkes Kaufargument. Bei diesem Kriterium sollte man sich auf eine einfach Skala einigen um diverse Produkte bzw. Projekte einordnen zu können. Eine genaue Abschätzung des benötigten Know-Hows ist nahezu unmöglich, jedoch schon eine ungefähre Unterscheidung kann wertvoll sein.

[291] McMahon (2020), inspiriert durch:.
[0] Github2019
[292] inspiriert durch betriebliche Beobachtung
[293] inspiriert durch eigene betriebliche Praxis
[294] gängiges Kriterium

[AQ10][T] Performance[295]: Die Performance einer Kubernetes Umgebung ist ein Qualitätsmerkmal welches sich aus der Performance jeder einzelnen Komponente zusammensetzt. Das bedeutet, dass nicht nur die vermeintlich klassischen Merkmale wie CPU und RAM, sonder auch Storage, Netzwerk und die Implementierungen jeder einzelnen Komponente erheblichen Einfluss auf die Gesamtperformance haben. Die Messung und Bewertung der jeweiligen Performance eines Teilsystems muss individuell erfolgen und kann auch nicht über Komponenten hinweg verglichen werden. Daher ist diese Qualitätseigenschaft nur sinnvoll im direkten Vergleich zweier oder mehrerer Produkte. Es empfiehlt sich die betrachteten Komponenten anhand einiger weniger aussagekräftiger Performance-Indikatoren zu vergleichen.

[AQ11][P] Qualifiziertes Personal: Viele Technologien im Kubernetes oder Cloud-Native Umfeld haben schon eine längere Historie. Daher ist es gut möglich, dass es im Unternehmen schon Mitarbeiter gibt die entsprechendes Know-How aufweisen. Dieser Punkt soll aufzeigen, dass sich manche Produkte bzw. Projekte eventuell leichter ins Unternehmen integrieren lassen, da es möglicherweise schon Wissende gibt. Es gilt auch zu berücksichtigen, ob es möglicherweise ähnliche Produkte gibt mit denen Mitarbeiter bereits Erfahrung haben. Auch die verwendeten Basistechnologien eines Produktes, wie beispielsweise die verwendete Programmiersprache, können bei vorhandenem Know-How den Einstieg erleichtern. Bzgl. dieser Eigenschaft werden Produkte anhand der bereits qualifizierten Personen im Unternehmen verglichen.

[AQ12][P] Unternehmensgröße: Die Größe eines Unternehmens ist definitiv kein aussagekräftiges Kriterium für die Güte der Software. Jedoch laufen kleinere Unternehmen eher Gefahr insolvent zu gehen oder gekauft zu werden also große Unternehmen. Dieser Punkt ist also eher im Sinne eines wirtschaftlichen Risikos zu sehen, welches sich natürlich direkt auf das bezogene Produkt des Unternehmens auswirken würde. Gemäß dieser Eigenschaft werden Produkte oder Projekte anhand der Größe des dahinterstehenden Unternehmens verglichen. Unter Größe wird die Anzahl an Mitarbeitern sowie die Marktkapitalisierung verstanden. Open-Source Projekte sind dabei von der Betrachtung ausgeschlossen, bei diesen sollte der Fokus auf der Community ([AQ06]) liegen.

[AQ13][P] Benchmarks: Eine sehr bequeme, wenn auch teilweise teure, Möglichkeit Produkte eines gewissen Sektors miteinander zu vergleichen sind Industriebenchmarks. Solche Benchmarks werden von spezialisierten Unternehmen durchgeführt, betrachten im Normalfall viele Aspekte und sind gut aufbereitet. Benchmarks bieten sich an, sofern vorhanden, um schnell eine Gegenüberstellung potenzieller Produkte zu erhalten.

[AQ14][P] State-of-the-Art Technologie: Die Technologie hinter einem Produkt oder einem Projekt muss zwar nicht zwangsläufig etwas über die Güte der Software aussagen, jedoch sollte darauf geachtet werden auf moderne und zukunftsträchtige Technologien zu setzen. Speziell wenn das Unternehmen selbst in die Entwicklung eines Projektes/Produktes involviert ist, oder dies plant, ist darauf zu achten für welche Technologie man sich entscheidet. Für diese Eigenschaft empfiehlt es sich eine einfache Skala zu definieren, anhand welcher die Produkte verglichen werden können. Welche Technologie als wie modern und verheißungsvoll bewertet wird ist durch Recherche und subjektive Einschätzung zu definieren.

[AQ15][P] Dokumentation: Die Qualität von Dokumentation bezieht sich auf Faktoren wie Vollständigkeit, Verständlichkeit, Aktualität und Bedienbarkeit. Diese Eigenschaft soll verdeutlichen, dass es große Unterschiede in der Qualität von Dokumentation gibt, und dass diese durchaus relevant sind. Es empfiehlt sich hierbei pragmatisch eine einfache Skala zu definieren, anhand derer die Dokumentation eines Produktes/Projektes bewertet wird. Anhand der vergebenen Bewertungen können anschließend die Kandidaten verglichen werden.

[AQ16][P] Support Qualität: Dieser Punkt sagt aus, dass es, obwohl man Enterprise Support bezieht, durchaus Unterschiede in der Qualität des Support gibt. Kriterien wie die Antwortdauer auf Anfragen, das Level an Expertise des Support Mitarbeiters oder die verfügbaren Sprachen sind wichtige Punkte. Um die Qualität von

[295] gängige betriebliche Metrik

Supportleistungen einschätzen zu können sollte man sich auf die Erfahrungen bestehender Kunden, sowie auf auf das angebotene Supportpaket berufen. Es bietet sich an die Qualität anhand einer einfachen Skala festzumachen und anhand derer die Angebote zu vergleichen.

[AQ17][P] Ausgereiftheit: Diese Eigenschaft beschreibt wie ausgereift eine Software bereits ist. Grundsätzlich bietet es sich an, Software zu verwenden die schon erprobt ist und einen hohen Reifegrad besitzt. Speziell in der schnelllebigen cloud-native Umgebung ist es wichtig einzuschätzen wie stabil und ausgereift ein Produkt bereits ist. Es bietet sich an zu betrachten wie viele Releases es bisher schon gab, wie viele offene Bugs/Issues es gibt und wie die Meinung der Community zum Produkt ist. Zur Bewertung sollte auf eine einfache Skala zurückgegriffen werden, anhand derer die Optionen bewertet und verglichen werden.

[AQ18][P] Nutzerkreis/bestehende Kunden: Referenzkunden sind ein aussagekräftiges Indiz dafür, wie weit eine Software schon in den Markt vorgedrungen ist. Speziell große und namhafte Unternehmen haben oft hohe Anforderungen und haben daher ein gewisses Gewicht als Referenzkunde. Es kann auch interessant sein zu wissen, welche Software Partner- oder Konkurrenzunternehmen mit ähnlichen Anforderungen verwenden. Diese Eigenschaft zeigt auf, wie gut das Portfolio an bestehenden Nutzern einer Software ist. Als Messgröße sollte eine einfache Skala definiert werden, welche die Anzahl, Größe und Art der relevanten Referenzunternehmen berücksichtigt.

[AQ19][P] Benutzerfreundlichkeit: Es gibt keine harten Kriterien an denen sich die Benutzerfreundlichkeit festmachen lässt, jedoch ist sie essentiell für jede Software. Je benutzerfreundlicher, intuitiver und leichter verwendbar ein Produkt ist, desto mehr Akzeptanz wird es erhalten. Auch die benötigte Zeit um mit dem Produkt zu arbeiten, ist bei benutzerfreundlicher Software meist wesentlich geringer. Um die Benutzerfreundlichkeit zu quantifizieren bietet sich eine Skala an, anhand derer die Optionen bewertet und anschließend verglichen werden. Dafür ist entweder auf bestehende Erfahrungswerte zurückzugreifen, oder pro Option ein Proof of Concept durchzuführen, um ein Gespür für die Software zu bekommen.

[AQ20][P] Versionsmanagement und Security/Hardening: Das Produkt oder die Kubernetes Distribution selbst sowie die darunterliegenden Hosts müssen regelmäßig aktualisiert und gemäß aktuellen Sicherheitsstandards konfiguriert werden. Das Anwenden von best-practices um potenzielle Fehlkonfigurationen auszuschließen und Sicherheitslücken zu schließen nennt man im Fachjargon "hardening". Dies, sowie das aktive Management der verwendeten Softwareversionen, ist meist zeitaufwändig und komplex. Diese Eigenschaft beschreibt wie viel Aufwand für das Management der Softwareversionen und hardening pro Zeiteinheit anfällt. Es empfiehlt sich als Messgröße die Personentage pro Quartal zu verwenden.

[AQ21][P] Häufigkeit von Releases/Updates: Die Häufigkeit in der neuen Releases oder Updates erscheinen sagt wenig über die endgültige Qualität einer Software aus. Sie ist jedoch ein Indiz dafür wie agil das Entwicklerteam hinter dem Produkt auf mögliche Fehler oder Feature-Requests reagiert. Speziell bei der Behebung von etwaigen Fehlern (Bugs) kann es daher vorteilhaft sein, wenn das Produkt häufiger Updates bereitstellt. Als Messgröße für diese Eigenschaft empfiehlt sich die Anzahl an Releases/Updates pro Quartal. Auch die durchschnittliche Wartezeit auf einen Fix, gemessen ab dem Einmelden eines Fehlers, kann herangezogen werden.

[AQ22][P] mögliche Granularität der Berechtigungen: Diese Eigenschaft sagt aus, wie fein die Berechtigungen beim Zugriff auf ein Produkt eingestellt werden können. Manche Produkte bieten beispielsweise die Verschachtelung von Rollen und Gruppen, was in der Praxis sehr nützlich ist. Andere Produkte bieten dieses Feature nicht an. Auch unterscheidet sich die Anzahl/Granularität an verschiedenen Rechten, die einer Rolle/Gruppe oder einem Benutzer zugeordnet werden können. Im Normalfall ist es vorteilhafter, ein Produkt zu wählen welches imstande ist komplexere bzw. feinere Berechtigungen zu verwalten.

[AQ23][P] Auswertungs-/Überwachungsmöglichkeiten: Diese Eigenschaft beschreibt das Ausmaß der Möglichkeiten, welche ein Produkt bietet um dessen aktuellen Status auszuwerten. Dies könnte beispielsweise das Anbieten eines Endpunktes zum Auslesen von Performancedaten sein. Manche System bieten auch eigene Dashboards und grafische Darstellungen, oder die Integration in andere Softwaretools an. Es empfiehlt sich für diese Eigenschaft eine Skala zu definieren, auf der aufgetragen wird wir ausgereift die Möglichkeiten eines Tools sind, die dieses zu seiner Überwachung bietet.

[AQ24][P] Ressourcenverbrauch: Diese Eigenschaft beschreibt die benötigten Ressourcen eines Produkts. Grundsätzlich gilt, je weniger Ressourcen ein System benötigt, desto besser. Vor allem wenn zwei Produkte verglichen werden, die genau die selbe Aufgabe erfüllen sollen, kommt es auch auf deren Ressourcenverbrauch an. Metriken wie die benötigte CPU, Arbeitsspeicher oder Festplattenspeicher sind sinnvolle Vergleichsmerkmale.

Die Qualitätseigenschaften aus dieser Sektion helfen dabei die Güte eines Clusters und dessen Komponenten einzuordnen. Außerdem können diese Eigenschaften als Vergleichsgrößen bei der Auswahl der jeweiligen Produkte im Designprozess herangezogen werden. Wie aus den definierten Rahmenbedingungen, funktionalen Anforderungen und Qualitätseigenschaften ein Kubernetes System entworfen werden kann, wird in der nächsten Sektion beschrieben.

A.18 Konkrete Auswahl von Implementierungen

Dieser Abschnitt stellt einen Vorschlag für den sechsten Schritt des Designprozesses dar, welcher sich um die konkrete Auswahl von Produkten im Zuge des Designprozesses dreht. Die hier beschriebene Methode wird im praktischen Teil der Arbeit jedoch nicht mehr umgesetzt. Das Innovationsmanagement beschäftigt sich primär mit der Vorbereitung und Begleitung des Entwicklungsprozesses der Innovation. Die konkrete Implementierung bzw. Entwicklung der Innovation sowie der zugrundeliegenden Kubernetes Plattform ist jedoch Aufgabe einer anderen Partei. Aus diesem Grund wird an dieser Stelle nur mehr eine mögliche Variante für den sechsten Prozessschritt aufgezeigt, dieser jedoch nicht mehr weitergehend vertieft.

Um in diesem Prozessschritt von den Anforderungen auf konkrete Produkte schließen zu können, muss für jede Cluster Komponente ein Set von Eigenschaften vorliegen. Auf diese Eigenschaften wurde in Kapitel 3 bei jeder Cluster Komponente verwiesen, siehe Anhang A.9 und Folgende. Analog zu den Anforderungen haben auch sämtliche Eigenschaften einen Identifier, der pro Komponente ansteigend und mit einem Präfix versehen ist. Nur wenn ein Produkt Eigenschaften aufweist die auch zu den geforderten Anforderungen passen kommt es für den Cluster in Frage. Um beim Entwerfen des Clusters von den definierten Anforderungen auf die jeweils relevanten Eigenschaften rückschließen zu können, bedarf es einer Verknüpfung dieser Informationen. Wie diese Verknüpfung konkret aussieht muss vom Team definiert werden, welches diesen Schritt durchführt. Im Weiteren wird erklärt wie weiter vorgegangen wird, wenn man eine Verknüpfung zwischen Anforderungen und Komponenten-Eigenschaften erarbeitet hat.

Als erstes muss für jede relevante Komponente erhoben werden welche Produkte bzw. Optionen es am Markt gibt. Pro Option müssen so viele Informationen wie möglich, sprich deren Eigenschaften, eingeholt und vergleichbar gespeichert werden. Bei der Auswahl der Produkte pro Komponente geben die Anforderungen an, welche Eigenschaften ein Produkt grundlegend erfüllen muss um in Betracht gezogen zu werden. Man filtert die Produkte also im nächsten Schritt einmal anhand ihrer Eigenschaften, sodass nur mehr die Produkte übrig bleiben, die grundlegend in Frage kommen. Man könnte diesen Schritt an folgendem Beispiel beschreiben: Man sucht einen Ingress Controller und hat die Anforderung, dass es ein Open-Source Produkt sein muss. Es sind jedoch nur drei bestimmte Lizenzen erlaubt. Im ersten Schritt erhebt man eine Liste aller am Markt verfügbaren Ingress Controller inklusive aller verfügbaren Informationen. Dann werden alle Produkte ausgeschieden welche keiner der drei zulässigen Open-Source Lizenzen unterliegen.

Für den nächsten Schritt sind die erarbeiteten Qualitätsanforderungen, ersichtlich in Anhang A.17, erforderlich. Gemäß dieser Qualitätsanforderungen werden die verbleibenden Produkte bewertet und gereiht. Sind für eine Komponente mehrere Qualitätsanforderungen interessant, so muss ebenfalls individuell definiert werden, welche Eigenschaft wie stark in die Entscheidung einfließt.

Der sechste Schritt des Designprozesses gliedert sich also nochmals in folgende Teilabschnitte:

- Aufstellen einer Referenzliste an verfügbaren Implementierungen pro Komponente
- Filtern der Produkte anhand benötigter Eigenschaften, vorgegeben durch die verknüpften Anforderungen
- Sortierung der verbleibenden Produkte gemäß definierter Eigenschaften

Die Abbildung A.2 soll den Zusammenhang zwischen Anforderungen, Implementierungen und deren Eigenschaften nochmals verdeutlichen. In der Grafik stehen für eine Komponente x sechs mögliche Implementierungen zur Auswahl. Auf Grund deren Eigenschaften, sowie der formulierten Anforderungen, können die Implementierungen

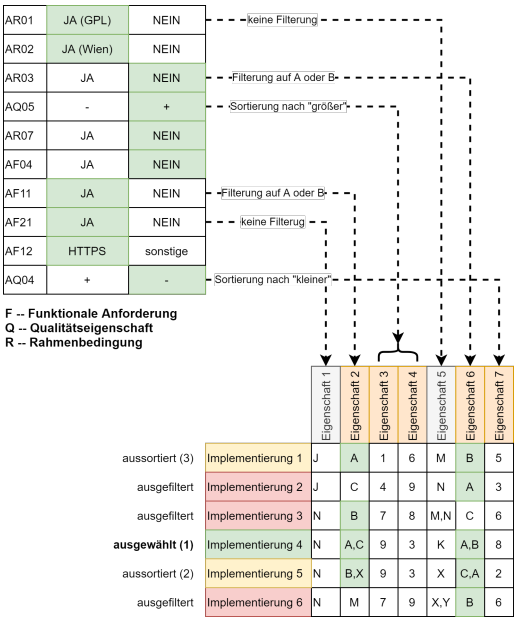


Abb. A.2: Zusammenhang zwischen Anforderungen, Implementiereungen und Eigenschaften

2,3 und 6 im ersten Schritt aussortiert werden. Welche Anforderung durch welche Ausprägung einer Eigenschaft erfüllt wird muss individuell beurteilt werden. In diesem Beispiel werden fiktive Werte verwendet. Gemäß der Qualitätseigenschaften, und der Eigenschaften welche diese technisch beeinflussen, werden die verbleibenden Implementierungen sortiert. Die Sortierreihenfolge ist hierbei willkürlich festgelegt worden. Es wurde erst nach den Eigenschaften 3+4 sowie anschließend nach Eigenschaft 7 sortiert. In diesem Fall fällt die Auswahl auf Implementierung 4.

Anmerkung zur Auswahl von Implementierungen: Es sei an dieser Stelle erwähnt, dass bei den jeweiligen Implementierungen darauf geachtet werden muss, dass diese auch zu anderen gewählten Implementierungen und zur Kubernetes Engine selbst kompatibel sind. Des Weiteren sei angemerkt, dass es empfehlenswert ist mit der Auswahl der Kubernetes Distribution selbst zu starten. Diese Entscheidung hat die größten Implikationen auf das gesamte System und wirkt sich auch auf die Auswahl verfügbarer Produkte für die restlichen Komponenten aus.

A.19 Workshop: One-Pager

Lieber Teilnehmer/Liebe Teilnehmerin!

Mein Name ist Daniel Drack, ich schreibe derzeit meine Masterarbeit im Fachgebiet Innovationsmanagement am Campus02. Die Arbeit dreht sich um das Design von Kubernetes Clustern, sowie um die dafür notwendigen Anforderungen an ein solches System. Dieser One-Pager dient als kleine Vorbereitung bzw. Erklärung für unseren anstehenden Workshop. Ich habe bisher im Theorieteil meiner Arbeit eine strukturierte Vorgehensweise zum Designen von Kubernetes Clustern herausgearbeitet. Kernstück dieser Vorgehensweise ist ein Anforderungskatalog der so vollständig wie möglich sein sollte. Mit Hilfe von Anforderungen kann gezielt nach den richtigen Produkten und Komponenten für den benötigten Cluster gesucht werden. Ich habe versucht anhand von Literaturrecherche und bekannten Case-Studies so viele sinnvolle Anforderungen wie möglich abzuleiten. Um jedoch noch etwaige branchenspezifische, und/oder aus praktischer Erfahrung gewonnene Anforderungen aufnehmen zu können brauche ich die Hilfe von Experten - wie dir. Grundlegend werden Anforderungen in drei grobe Kategorien eingeteilt:

- Rahmenbedingungen - nicht verhandelbare Grundgegebenheiten
- funktionale Anforderungen - Vorgaben an den Kubernetes Cluster
- Qualitätseigenschaften - Gütekriterien für den Cluster oder dessen Komponenten

Ich würde dich bitten schon vor unserem Workshop etwas über diese drei Kategorien nachzudenken und dir Anforderungen zu notieren, die aus deiner Erfahrung in diese Gruppen fallen können. Um die Kategorien etwas zu verdeutlichen habe ich hier noch einige Beispiele angeführt:

- Rahmenbedingungen
  - Open Source Software: Ist der Einsatz von Open-Source-Software möglich?Welche Open-Source-Lizenzen sind erlaubt?
  - Verfügbarkeit: Ist es notwendig, dass sich der Cluster in einem bestimmten Land oder einer bestimmten Region befindet?
  - Support: Ist Enterprise Support für die ausgewählte Kubernetes-Distribution notwendig?
  - Public Cloud: Ist die Nutzung von Public-Cloud-Diensten erlaubt oder sogar erwünscht?
- funktionale Anforderungen
  - Wird GPU Unterstützung benötigt?
  - Müssen gespeicherte Daten verschlüsselt werden?
  - Welche Betriebssysteme für Container werden benötigt? (Windows/Linux)
  - Muss die Kommunikation zwischen Pods regulierbar sein?
  - Über welche Protokolle wird von außerhalb des Clusters auf Services zugegriffen?

- Qualitätseigenschaften
  - Verfügbarkeit/Uptime [%]
  - Installation/Upgrade Aufwand [time]
  - Kosten [?]
  - Personalbedarf [FTE]
  - Marktanteil [%]
  - Größe/Qualität der Community

Bitte überleg dir schon vor dem Workshop ob dir Anforderungen aus deiner praktischen Erfahrung für diese Kategorien einfallen. Jeder Stolperstein und Lessons-Learned aus deiner Erfahrung mit Applikationsentwicklungen und Softwareprojekten ist wertvoll. Vor Allem die Themen die sich um die verwendeten Plattformen - wie Kubernetes - drehen sind von Interesse. Den betrachteten Aspekten sind hierbei keinerlei Grenzen gesetzt - von administrativen/organisatorischen Punkten bis tief technischen Fragen ist alles relevant.

Danke und LG Daniel Drack

A.20 Interviews: One-Pager

Lieber Teilnehmer/Liebe Teilnehmerin!

Mein Name ist Daniel Drack, ich schreibe gerade meine Masterarbeit im Fachgebiet Innovationsmanagement am Campus02. Die Arbeit dreht sich um das Design von Kubernetes Clustern, sowie um die dafür notwendigen Anforderungen an ein solches System. Dieser One-Pager dient als kleine Vorbereitung bzw. Erklärung für unser anstehendes Interview. Ich habe bisher im Theorieteil meiner Arbeit eine strukturierte Vorgehensweise zum Designen von Kubernetes Clustern herausgearbeitet. Kernstück dieser Vorgehensweise ist ein Anforderungskatalog der so vollständig wie möglich sein sollte. Mit Hilfe von Anforderungen kann gezielt nach den richtigen Produkten und Komponenten für den benötigten Cluster gesucht werden. Ich habe versucht anhand von Literaturrecherche und bekannten Case-Studies so viele sinnvolle Anforderungen wie möglich abzuleiten. Um jedoch noch etwaige branchenspezifische, und/oder aus praktischer Erfahrung gewonnene Anforderungen aufnehmen zu können brauche ich die Hilfe von Experten - wie Ihnen. Grundlegend werden Anforderungen in drei grobe Kategorien eingeteilt:

- Rahmenbedingungen - nicht verhandelbare Grundgegebenheiten
- funktionale Anforderungen - Vorgaben an den Kubernetes Cluster
- Qualitätseigenschaften - Gütekriterien für den Cluster oder dessen Komponenten

Ich würde Sie bitten schon vor unserem Interview etwas über diese drei Kategorien nachzudenken und dir Anforderungen zu notieren, die aus deiner Erfahrung in diese Gruppen fallen können. Um die Kategorien etwas zu verdeutlichen habe ich hier noch einige Beispiele angeführt:

- Rahmenbedingungen
  - Open Source Software: Ist der Einsatz von Open-Source-Software möglich? Welche Open-Source-Lizenzen sind erlaubt?
  - Verfügbarkeit: Ist es notwendig, dass sich der Cluster in einem bestimmten Land oder einer bestimmten Region befindet?
  - Support: Ist Enterprise Support für die ausgewählte Kubernetes-Distribution notwendig?
  - Public Cloud: Ist die Nutzung von Public-Cloud-Diensten erlaubt oder sogar erwünscht?
- funktionale Anforderungen
  - Wird GPU Unterstützung benötigt?
  - Müssen gespeicherte Daten verschlüsselt werden?
  - Welche Betriebssysteme für Container werden benötigt? (Windows/Linux)
  - Muss die Kommunikation zwischen Pods regulierbar sein?
  - Über welche Protokolle wird von außerhalb des Clusters auf Services zugegriffen?

- Qualitätseigenschaften
  - Verfügbarkeit/Uptime [%]
  - Installation/Upgrade Aufwand [time]
  - Kosten [?]
  - Personalbedarf [FTE]
  - Marktanteil [%]
  - Größe/Qualität der Community

Bitte überlegen Sie sich schon vor dem Interview ob Ihnen Anforderungen aus deiner praktischen Erfahrung für diese Kategorien einfallen. Dies erlaubt es uns das Interview kompakter und effizienter zu gestalten - danke.

LG Daniel Drack

A.21 Interviews: Leitfaden

- Eröffnung und Vorstellung des Interviewers
  - Ablauf des Interviews
    * Intro
    * Vorstellung Experte/Unternehmen
    * Designprozess
    * Rahmenbedingungen
    * Funktionale Anforderungen
    * Qualitätseigenschaften
  - Wer bin ich und was studiere ich?
  - Warum brauche ich diese Informationen?
  - Wie wird mit den gesammelten Daten vorgegangen?
  - Darf Interview aufgezeichnet werden?
  - Darf Unternehmen/Interviewee namentlich erwähnt werden?
  - Modus des Interviews - Online/Telefon/Face-2-Face?

- Vorstellung des Experten
  - Name des Experten
  - Aktueller Arbeitgeber und Position innerhalb des Unternehmens
  - Warum dieses Unternehmen
  - Grobe Erklärung der Kubernetes Umgebung im Unternehmen

- Strukturierte Vorgehensweise / Designprozess
  - Wer hat entschieden, dass Kubernetes verwendet wird?
  - Wie wurde die Kubernetes Umgebung entworfen?
  - Welche Stellen/Teams waren beim Designprozess beteiligt?

- Rahmenbedingungen
  - Was waren die definitiven Grenzen beim Designprozess?
    * strategische, rechtliche, technische, organisatorische Vorgaben?

- Funktionale Anforderungen
  - Welche Funktionen muss der Cluster erfüllen?
  - Gibt es spezifische Eigenheiten gegenüber "normalen"Clustern?
  - Welche Workload wird auf dem Cluster betrieben?

- Qualitätseigenschaften
  - Gibt es KPIs?
  - Wie wurde sich für die gewählten Produkte entschieden? Wie wurden diese unterschieden?
- Rückblick und Ausblick
  - Bedankung bei Interviewee
  - Einverständniserklärung
  - Next Steps und Ergebnisse der Arbeit

A.22 Interview: Protokoll Unternehmen A

1 **I:** Grundsätzlich hätte ich gedacht, wird das Ganze folgendermaßen ausschauen: Ein kurzes Intro, Vorstellungen von dir und vom Unternehmen, was grundsätzlich deine Positioning ist, was ihr schon macht. Dann würde ich gerne darüber sprechen, wie ihr Kubernetes grundsätzlich das System entwickelt habt. Also wie ihr so auf die Infrastruktur gekommen seid die ihr jetzt habt, sprich Designprozess unter Anführungszeichen. Und dann gerne über die drei groben Gruppen von Anforderungen, die ich in der Einladung schon mitgeschickt habe. Das ein bisschen abklopfen, was was was Unternehmen da so eventuell branchenspezifische Sachen hat, das wir nicht haben. Also grundsätzlich kennen wir uns schon. Also "wer bin ich?" fällt weg. Ich studiere mittlerweile Innovationsmanagement am Campus02 in Graz im Master und sollte jetzt hoffentlich im Februar fertig werden. Ich brauche diese Info weil ich mich dafür entschieden habe in der Masterarbeit den Designprozess von Kubernetes ein bisschen strukturiert zu betrachten. Also wie könnte man es wirklich strukturiert angehen, und nicht nur: "Schauen wir mal, dann sehen wir schon", sondern wirklich strukturiert das Ganze angehen. Und das Herzstück dessen ist im Prinzip ein Anforderungskatalog, mit dem man spezifizieren kann, was mein Unternehmen eigentlich braucht, damit die dann das dementsprechendes System wirklich so gut wie möglich zusammenbauen kann. Auch die verschiedenen Komponenten und nicht ungetrieben dahin stolpere. Ich hab das jetzt in der Theorie schon gemacht, das erhoben und mache dann in meinem Unternehmen bereichsübergreifend einen Workshop zum Thema, wo ich versuche die verschiedenen Disziplinen abzuklopfen. Lizenzmanagement, Prozessmanagement, Projektleiter, Operations Manager so verschiedene Disziplinen und ergänzend dazu hätte ich gerne Experteninterviews, damit ich auch die Blickwinkel verschiedener Branchen sehe. Und da hätte man jetzt dein Unternehmen aus Finanzsicht, damit man da eventuell zusatzlich noch ein paar Anforderungen herausfindet, die wir z.B. in der Automotive so eventuell nicht haben.

2 **B:** Also du bist in der Automotive, studierst und arbeitest nebenbei noch?

3 **I:** Ja, sicher, ich arbeite Vollzeit. Also gleich wie im letzten Unternehmen.

4 **B:** Also habt ihr auch schon Kubernetes Cluster?

5 **I:** Ja, also wir haben seit fast zwei Jahren, also genau mein Team, also ich bin jetzt in mit einem DevOps Team in der IT und wir haben mittlerweile 7 produktive Cluster, davon 4 DMZ Cluster und zwar Spielwiesen Cluster auf drei Kontinenten. Also schon schon einiges, aber wir haben ja genau des Problem. Wir wir sind halt so reingestolpert und ich hab mir gedacht, es wäre eigentlich sinnvoll, vor allem für große und professionelle Unternehmen das ganze ein bisschen strukturierter anzugehen, und sich im Vorhinein zu überlegen, was was man will. Anstatt im Nachhinein da drauf zu kommen, was es nicht war.

Rechtfertigung der Masterarbeit

6 **B:** Das passiert öfters.

7 **I:** Aber anscheinend gibt es ja Möglichkeiten, das ein bisschen besser machen zu können. Darf ich dich bitten? Mal ganz kurz was. Also wer bist du? Machst du? Und was macht ihr?

8 **B:** Mein Name ist Person A. Ich bin seit ungefähr 20 Jahren im Unternehmen und immer im Bereich Plattformen tätig gewesen. Ich glaube den Begriff Plattformen gibt es auch schon seit ewigen Zeit. Seit man Applikationen und Infrastruktur trennen wollte und einen stabilen Unterbau machen. War früher im Server Umfeld. Irgendwann wurde daraus mal eine eigene Unternehmens Plattform die ABC hieß mit Chef Provisionierung, immer mit dem Background Infrastructure as Code und DevOps Prinzip so gut es geht umzusetzen. Flexibler, skalierbarer zu werden, und dann war eigentlich der Weg in Richtung Kubernetes eine logische Evolution von der alten Platform. Und deswegen wussten wir immer schon, wo das Kubernetes Thema aufgekommen ist, dass wir dahin wollen. Weil es ist dann ein Industriestandard wo wir eine ausgebaute Automatisierung anwenden können. So sind wir zu dem Kubernetes Thema gekommen und haben im Unternehmen mit Kubernetes begonnen.

..Stellenbeschreibung

..Motivation für Kubernetes

..Vorschlag für Kubernetes

9 **I:** Darf ich fragen wie eure Infrasturktur grob aussieht? Welche Basiskomponenten verwendet ihr? Welche Cluster habt ihr, in welcher Größenordnung ungefähr? Dass man es ein wenig einordnen kann.

10 **B:** M Wir haben derzeit drei Cluster, also einiges weniger als ihr und haben diese nach einem Staging Konzept aufgebaut. Also einen Incubator Cluster auf dem Technologien und Applikationen einfach mal getestet werden können im Sinne von komplette Experimentierfreudigkeit. Da ist alles erlaubt, der Cluster darf auch keine Egress Connections machen. Dann haben wir einen Non-Prod Cluster, wo die Applikation in ihrem normalen Release Management bis in die Vorproduktion gehen. Also von Integrationstests bis in die Vorproduktion alles dabei. Und dann haben wir einen eigenen produktiven Cluster. Derzeit unterscheiden wir aber noch nicht nach Teams oder nach unterschiedlichen Applikationen, sondern einfach nur nach Stages.

..Kubernetes Architektur

11 **I:** Grundsätzlich, wie seid ihr das Thema angegangen? Habt ihr gesagt wir nehmen Upstream Kubernetes und schauen dann mal wies funktioniert? Oder habt ihr wirklich davor schon versucht das strukturiert ein bisschen heranzugehen, bisschen abgeklopft welche needs ihr so habt? Wie habt ihr das ungefähr angegangen?

12 **B:** Ähm..Also wo .. also damals sind wir noch getrennt gewesen, wir waren in der Systemarchitektur und die ganze Infrastruktur wo auch Linux Server betrieben werden und so weiter, war ein eigener Bereich. Und wir aus der Systemarchitktur haben gesagt wir möchten einen Evolutionsschritt machen von der alten Applikationsplattform hin zu Kubernetes und haben dann die Infrastruktur gefragt welche Kubernetes Distribution würde euch am besten gefallen. Wir waren immer so in Richtung OpenShift, weil wir das halt überall gehört haben und die Infrastruktur Abteilung hat sich aber aus ihrem VMWare Background, weil vSphere ESX eben im Unternehmen hauptsächlich verwendet wird.

..Vorschlag für Kubernetes

..[AR09] Strategesche Partne

..Entscheidung für Kubernete

..Technologiestack

..Integration in bestehende

..[AR09] Strategesche Partner
..Entscheidung für Kubernetes
..Technologiestack
..Integration in bestehen
..[AQ03] Preis/Lizenzkost
..[AR04] Testcluster Raba

Haben sie gesagt es würden sich starke Synergieeffekte ergeben wenn wir auch bei der Kubernetes Distribution auf VMWare setzten würden. Auch mit NSX Netzwerkintegration und so weiter. Das war also mal der erste Schritt, dass man sich für diese Distribution entschieden hat und hier aber auch von der Vorgehensweise her klein anfangen. Wir haben das auch als Experiment gestarten und haben zu diesem Zeitpunkt noch nicht gewusst ob das im Unternehmen ein Erfolg werden kann. Dort ist uns auch die VMWare preislich entgegen gekommen und können ein Attraktives Angebot machen, um einiges besser als OpenShift , wo wir von Anfang an schon sehr viel zahlen hätten müssen. Das war eben der Weg. Wir wollten wirklich ganz klein Anfangen und nochmal schauen, wird da was daraus.

13 **I:** Darf ich gleich was dazwischen fragen? War das für euch ein KO Kriterium, die kommen euch da preislich etwas entgegen? Oder habt ihr gesagt... für die Entscheidung Kubernets per se ist es uns egal, aber für dieses Produkt ist es uns wichtig, dass sie uns entgegen kommen, sonst sind sie draußen. Also für die die es in dem Fall..

..[AQ03] Preis/Lizenzkosten
..Entscheidung für Kubernetes

14 B: Also für das Produkt war es wichtig ob wir uns dann, wenn insgesamt Kubernetes zehn mal so teuer wäre wenn wir uns für Kubernetes entschieden hätten, dann hätte sicher die Entscheidungsphase länger gebraucht. Und die Überzeugungsarbeit weil, das kann ich jetzt schon sagen. Die Notwendigkeit einer Kubernetes Plattform bottom-up gekommen ist, also von den Technikern als, dass jemand vom Management gesagt hätte wir brauchen Kubernetes. Also ich glaube die wissen gar nicht..

15 I: Same here. Gleich meine Folgefrage: Wer hat denn wirklich unter Anführungszeichen entschieden, dass ihr das machen dürfts, sollt oder wollt?

..Entscheidung für Kubernetes

16 B: Die Bereichsleiter haben sich entschieden, dass wir das im Zuge eines Pilotprojektes.. da ist dann eine EU Richtlinie herausgekommen. Das hat geheißen BSD2 Richtlinie. Dass jedes Rechenzentrum, Bankenrechenzentrum und jede Bank API zur Verfügung stellen muss fürs Internet-Banking, und da hat man schon gewusst, dass dieses ganze API Thema noch mehr an Fahrt aufnehmen wird. Dafür hat man sich dann entschieden, das Ganze dann auf der Kubernetes Plattform zu machen. Wir haben dann vom Bereichsleiter der Entwicklung das OK bekommen, BSD2 machen wir auf Kubernetes. Von der Architektur auch, dass wir das angehen mit Kubernetes, und auch von der Infrastruktur. Wir sind gleich mit einem Kundenprojekt gestartet, zwar mit einem das damals nicht so riskant war und noch nicht so ein Zeitdruck da war, aber prinzipiell nicht mir irgend einem "hello world" Programm, sondern gleich vollgas Kunden API.

17 I: Habt ihr dann wirklich mehr oder weniger von einem green-field Projekt gestartet, oder habt ihr eine bestehende.. weil du gesagt hast WebSphere zum Beispiel.. bestehende Installation und Programme versucht zu portieren auf Kubernetes? Oder habts ihr wirklich gesagt ihr habts jetzt die Plattform und könnt darauf bauen wie ihr wollt?

..Art der Workload	18	B: Grundsätzlich war es green-field weil die Applikation.. wir haben dann auch, open source nicht von einem Fremdhersteller, eine API hergenommen. Die haben auch gesagt sie sind Kubernetes tauglich und haben auch HELM charts dafür gehabt und so weiter. Und mit dem sind wir gestarten. Alle anderen legacy Applikationen haben wir auf die legacy Plattformen belassen, derweil für den Anfang. Da fängt jetzt ein Portierung an, aber prinzipiell kann man grundsätzlich sagen das war green-field. Green-field mit dem Wissen, dass man schon legacy Systeme anbinden müssen.
	19	I: Vielleicht ein kleiner Vorgriff auf funktionale Anforderungen. Gibts wirklich etwas wo ihr sagt das sind exotische Anforderungen an so einen Cluster, die er unterstützen muss? Also Beispiel wir zum Beispiel wir haben ziemlich hardcore build Maschinen auf Windows. Also wir haben einen Hybrid-Cluster auch, was es definitiv nicht so hundert prozentig Standard ist. Noch was nicht so ganz trivial ist, aber das ist eine unserer Anforderungen weil wir dot net Anwendungen haben und dot net core Anwendungen haben die gebaut werden wollen. Gibts da bei euch auch irgendwas wo ihr sagt ihr habt exotischere Anforderugen?
..[AR07] rechtliche Vorgaben	20	B: Noch nicht. Also das was wir sicher am meisten zu kämpfen haben sind im Banksektor die ganzen Governance Richtlinien, Security Richtlinien, das ist etwas wo wir in der letzten Zeit am meisten gemacht haben. Aber das haben wir zu Beginn, also beim Pilotprojekt noch teilweise ignoriert, oder hats es die harten Anforderungen noch nicht gegeben. Das müssen wir jetzt alles nachholen, aber ansonsten sag ich jetzt mal technologisch war das alles am Anfang recht klassisch - normal.
	21	I: Vielleicht noch eine abschließende Frage zum Designprozess. Ich nehme mal mit bottom-up wie du gesagt hast, das war also eine Entwickleridee, gestartet mit einem Pilotprojekt, dann abgesegnet. Ihr habt euch dann für die Distribution von VMWare entschieden, für. Oder fragen wir anders herum: Bietet diese Distribution schon alles was Kubernetes kann, auch Richtung Logging Interfaces, Metriken Alerting und so weiter wie OpenShift das tut? Oder habts ihr euch da noch eigene Sachen angeschaut? Also wir zum Beispiel haben gerade fluentD evaluiert als log Agent, Prometheus für Metriken natürlich. Aber da gibts ja verschiedenste..
..Vendor Lock-In [AR16]	22	B: Nein wir haben uns sehr viel trotzdem zusätzlich selbst angeschaut, zum Einen weil wir eigentlich noch herstellerneutral sein wollten. Wir wollten kein komplettes VMWare vendor lock-in haben, und wir haben uns OpenShift angeschaut und uns kommt immer noch vor die all-in-one Packages sind ganz gut zum Starten. Ja man hat ein bisschen CICD pipeline, man hat ein bisschen ein Monitoring, aber wenn man es dann wirklich professioneller verwenden will kommt man ziemlich schnell darauf man muss doch best-of-breed gehen. Wir haben zum Beispiel Dynatrace zum Monitoring, also wirklich ein recht teures Produkt sogar. Vor allem weil wir es schon im Haus gehabt haben und müssen es ja intergieren in unsere bestehende Systemlandschaft. Wir haben Splunk, wir nehmen auch fluentD Agent und halt Splunk connect for Kubernetes um die ganzen Daten in Splunk zu bringen. Und wir
..Technologiestack ..[AF09] Logging ..[AF08] Metriken		

..Technologiestack
..[AF09] Logging
..[AF08] Metriken

haben uns eine eigene CD pipeline mit gitlab CI und Argo CD als GitOps Agent.

23 I: Ja des haben wir auch. Gut also Vorgehensweise glaube ich bin ich soweit mal happy. Grundsätzlich Rahmenbedingungen: gibts irgendwas wo ihr gesagt habts das war indiskutabel? Wo ihr gesagt habts ihr könnts euch was anschauen, aber bestimmte Produkte, Distributionen fallen einfach raus auf Grund von x? Du hast schon gesagt ihr habts so eine Art vendor lock-in mit VMWare, dass ihr gesagt habts es ist naheliegend, dass ihr da mal nachfragt. Oder eben mit die Regulatorien da hat es mit Security was gegeben. Gibts so bestimmte Rahmenbedingugnen wo du sagst die... die definitiv da sind?

..Versionsmanagement + Host/K8s Ha
..[AR07] rechtliche Vorgaben
..Antivirus
..[AQ06] Communitygröße und -qu
..[AR01] Open-Source Lizenzen

24 B: Ja also was wir mittlerweile haben, das hat man am Anfang nich so sehr gesehen, sind eben diese ganzen Security Richtlinien. Also wenns es gerade um Schwachstellenmanagement geht, um hardening... dann müssen wir auf das sehr sehr stark schauen. Wobei ich auch sagen muss... weil du auch als ersten Punkt open-source gehabt hast... das hat open-source nicht ausgeschlossen. Also zum Beispiel Argo CD wenn die Community groß genug ist, wenn die irgendwie zeigen können, dass sie sich um Security kümmern und wir intern über unsere Image Prüfungen da auch sehen dass das passt, dann funktioniert sowas auch. Und wenn die Lizenz passt. Also das kann ich vielleicht auch vorweg nehmen, wir hatten dadurch dass dies Apache 2.0 Lizenz war hier auch nie Probleme mit unsren Legal Leuten. Und sonst eigentlich auch keine Geschichten bei denen ich sage es geht gar nicht.

25 I: Das heißt ihr habt schon gewisse open-source Linzenzen die ihr ablehnen würdet? Also Apache 2.0 oder MIT und so sind ja relativ gütig, aber es gibt auch ein paar die gerade mit kommerzieller Nutzung oder Weiterverbreitung nicht so flauschig sind.

..[AR01] Open-Source Lizenzen

26 B: Wir haben es jetzt nicht... Also ich kenne jetzt keine offizielle Richtlinie die jetzt welche ausschließen würde, aber ich weiß halt umgekehrt, dass uns gesagt wurde.. wie du jetzt sagst.. MIT und Apache 2.0 sind sehr günstig für kommerzielle Produkte.

27 I: OK. Technisch in Richtung Rahmenbedingungen.. Verwendet ihr nur virtuelle Maschinen oder macht ihr auch bare-metal Nodes?

..Kubernetes Architektur
..Technologiestack

28 B: Wir verwenden nur virtuelle Maschinen.

29 I: OK. Ja Rahmenbedingungen sonst... Ich glaube den Großteil haben wir eh gesagt. Rein aus Interesse..

..[AR10] Verwendung der public cloud

30 B: Was man auch noch sagen kann.. Wir verwenden noch einen on-premises Cluster, wir sind nicht in der public cloud.

31 I: OK das hätte ich implizit schon mitgenommen, aber es ist eine gute Info.

..[AR10] Verwendung der public cloud

32 B: Aber die Diskussion kommt jetzt, es ist nicht so, dass wir jetzt ewig bleiben werden. Es kommt aus einer anderen Ecke... Schon ein Diskussion warum gehen wir nicht in die public cloud. Also uns

..[AR10] Verwendung der public cloud

drängen auch ein paar.. weil es einfach.. also externe Berater interessanterweise. Weil sie sagen es ist kostentechnisch.. macht es das besser. Wir sind skalierbarer. Vielleicht haben die auch noch nicht ganz begriffen, dass wir eh manageded Kubernetes verwenden und nicht vanilla upstream. Aber trotzdem müssen wir uns viel mehr der Diskussion stellen und möglicherweise... Man fangt vielleicht auch an ein cloud Transformation Projekt zu machen wo man ein Assessment macht was vom Unternehmen könnten wir in die cloud bringen. Also es könnte sein, dass wir in einem Jahr hier sitzen und sagen "Ja, tatsächlich, wir haben einiges in der cloud"

33 I: Ja, also soviel auch von unserer Seite. Wir haben Azure und sind sehr Microsoft getrieben, aber bei uns ist es auch.. die Transistion in Richtung Cloud, O365, Azure AD ganz ganz stark. Vor allem jetzt mit Corona. Und gerade in Richtung Kubernetes wäre interessant federated Cluster... oder AKS hybride.. solche Sachen überlegen wir. Und vor Allem für Affiliate Standorte.

34 B: Was habt ihr für eine Distribution.

35 I: Upstream.. wir fahren Upstream.

36 B: Cool

37 I: Also wir haben überlegt OpenShift aber da sind wir mit RedHat noch nicht so auf einen Nenner gekommen.. auch preislich. Aber.. grundsätzlich Upstream. Den Grund erzähle ich dir im Nachhinein, wenn ich nicht mehr abtippen muss... also wir können dann gerne weiter reden, aber ich muss das alles noch transkribieren. Ich glaube wir haben schon mal angefangen.. also Rahmenbedingungen soweit für mich ok. Thema funktionale Anforderungen.. ich habe mitgenommen ihr macht die Schnittstelle also BSD2. Das heißt im Prinzip, der Cluster hosted Microservices? Macht der sonst noch was in Richtung Stichwort Machine Learning, AI, Windows Linux Hybrid Applikationen, Datenbanken, stateful Applikationen.. sowas?

..Art der Workload

38 B: Machen wir noch nicht, nein. Also wir haben Apache Kafka noch zusätzlich oben.. Also Event streaming. Und Datenbanken teilweise integriert für so Sandboxes, so PostgreSQL aber nicht... aber.. wir persistieren die Daten jetzt nicht so, dass man sagt man könnte sie jederzeit wiederherstellen. Also wenn die weg sind sind sie weg.

39 I: OK also es heißt ihr verwendent schon den "treat your application as cattle" Ansatz?

40 B: Ja, genau.

41 I: OK, gut das Thema Eigenheiten oder irgendwelche exotischen Anforderungen haben wir eh schon gehabt. Workload haben wir jetzt im Prinzip auch gehabt, da seid ihr noch relativ konservativ. Gut ich glaube mit funktionale Anforderungen, sollte dir dazu nicht noch was einfallen, glaube ich wärs das dann eigentlich schon. Also.. naja wobei.. hab ihr in Richtung Storage oder in Richtung Netzwerk irgendwelche spezifischen Anforderungen? Also gerade

in Richtung network policy zum Beispiel. Verwendet ihr Meshes, oder habt ihr irgendwelche speziellen fancy Storage Systeme? Mit, keine Ahnung, high performance oder Verschlüsselungsanforderungen oder solche Sachen?

..[AF15] Persistenter Speicher
..[AF02] data-at-rest Encryption
..[AF07] Pod2Pod Kommunikati
..[AF13] Service Mesh
..verschlüsselte Kommunikation
..Dauer Kubernetes Verwendun

42 B: Nein haben wir nicht. Mit Storage fangen wir jetzt erst richtig an uns das genauer anzuschauen, weil wir bis jetzt immer gesagt haben, offiziell unterstützt die Plattform noch keine persistenten Daten, die sind alle volatil. Da fangen wir in den nächsten Monaten an, weil es Anforderungen gibt... Verschlüsselung weiß ich ehrlich gesagt keine speziellen Anforderungen. Richtung Netzwerk oder Mesh, ja Vorgabe, ist jeder Namespace hat default mäßig eine deny all policy, Kann also nicht mit anderen Pods kommunizieren. Wenn man das braucht muss man es explizit freigeben.. also wirklich Microsegmentierung auf Netzwerk Ebene und jede Kommunikation muss extra freigegeben werden. Service Mesh ist auch in der Evaluierung weil... also die BSD2 APIs das war der Anfang, damit haben wir ungefähr vor zwei Jahren angefangen. Mittlerweile sind auch noch andere APIs oben und man will das Produkt X neu... das sind genau so Microservices.. auf Kubernetes bringen. Und da überlegt man sich mit Hilfe von Istio... resilience beziehungsweise Verschlüsselung, Kommunikationsverschlüsselung auszulagern in einen Service Mesh.

43 I: Rein aus Interesse: Istio weil erste Wahl oder Istio weil begründet evaluiert?

..Qualifiziertes Personal [AQ11]

44 B: Ja evaluiert hat das eigentlich das Projekt X auch zusammen mit einem Externen der... also ich glaube der bringt da glaube ich einfach für Istio ziemlich viel Erfahrung mit und der sagt.. das ist nicht die schlechteste Wahl.

45 I: Gut.. also wie gesagt das war rein aus Interesse. OK gut funktional sonst, wenns derlgeichen nichts mehr gibt..

..[AF12] Layer 4 / Load Balancer

46 B: Protokolle.. weil du das gesagt hast. Wir greifen hauptsächlich von außen mit HTTPS zu, nur für Apache Kafka gibt es noch so ein proprietäres protokoll.

47 I: Ja der hat ein proprietäres. Den Spaß hatte ich auch schon. Selber ein HELM chart schreiben für Kafka.. nie wieder.

48 B: Nein, wir nehmen hier StreamC.. die StreamC Kafka Distribution.

49 I: Da glaube gibt es sogar eine mit Operator mittlerweile, der nicht nur das Chart macht sondern den Operator deployed.

50 B: Genau, wir verwenden den Operator. Aber .. na gut.

51 I: Gut, vorletzter Punkt: Qualitätseigenschaften. Habt ihr grundsätzlich Eigenschaften oder KPIs wo ihr sagt ihr könnt unterscheiden welche Produkte ihr eher verwenden wollt, im Vergleich zu anderen. Oder wie ihr die Qualität von eurem Cluster messen könnt. Also wie gesagt als Beispiel hätte ich mal Kosten natürlich, aber Community hast du eh schon gesagt. Also das wäre ein Punkt bei uns gewesen, also wir entscheiden uns für Produkte

anhand von ihrer Communitygröße.. GitHub Sterne.. irgend sowas in die Richtung. Aber gibt es auch intern etwas wo ihr sagt ihr habt eigene KPIs die die Güte des Clusters auch qualifizieren?

..[AQ02] Verfügbarkeit in %

52 B: Ja wir müssen im Zuge von unserem SLA Management KPIs angeben, so etwas wie Verfügbarkeit oder maximale Ausfallszeit oder Anzahl der Ausfälle pro Monat. Da haben wir was wenn es in die Richtung geht. Also wir sagen zum Beispiel wir haben uns mal an die Verfügbarkeitsanforderungen unserer wichtigsten Applikation.. also das ist das BSD2.. gehängt und da haben wir eine Verfügbarkeitsanforderung von 99 Prozent. Das heißt 7.3 Stunden Ausfalls insgesamt pro Monat ungefähr, maximale Ausfallszeit pro Einzelfall sieben Stunde und maximale Anzahl Ausfälle pro Monat zwei. Das sind die Verfügbarkeits KPIs die wir momentan haben, die wir erfüllen müssen.

53 I: Gut, rein aus Interesse: habt ihr euch auch mal Gedanken gemacht über solche Sachen zum Beispiel wie lange brauche ich um ein Produkt installieren zu können, oder wie schaut die Lernkurve aus? Um vergleichen zu können für was ihr euch entscheidet.

54 B: Meinst du jetzt um den Kubernetes Cluster zu installieren, oder um Applikationen auf dem Kubernetes Cluster?

55 I: Die Kubernetes Distribution oder Komponenten dessen. Also zum Beispiel ein CNI Plugin oder Storage Plugin, oder keine Ahnung... die Logging Lösung oder so.

..Qualifiziertes Personal [AQ11]

..Integration in bestehende Infrastrukt

56 B: Nein das waren jetzt keine ausschlaggebenden Kriterien. Also da war es eben eher so wie bei VMWare, "kennen wir das Produkt schon und können wir es leicht integrieren?". Wir sind von dem ausgegangen, dass man deswegen auch die Installation und Upgrade.. leichter geht und das bestätigt sich jetzt eigentlich. Da haben wir.. sehen wir keine Schwierigkeiten. Und auch bei den anderen Produkten, ich mein die waren jetzt nicht... Sowie Dynatrace und Splunk die waren eher gesetzt weil wir sie schon im Haus haben und weil wir dann eigentlich auch angenommen haben wir tun uns leichter mit der Lernkurve und dem Installieren. Das hat sich auch bewarheitet. Bei Argo CD das war einfach die... dass wir relativ schnell darauf gekommen sind, dass der GitOps Ansatz für uns perfekt war.

57 I: Gut. Ja passt dann hätte ich gesagt den offiziellen Teil hätte ich durch. Vielen vielen Dank auf jeden Fall. Sehr sehr wertvoller Input und ich muss sagen es schlagt auch Gott sei Dank ziemlich in die Kerbe in die ich schon erhofft hätte. Ich schicke dir noch die Einverständniserklärung, muss nur noch deinen Namen darauf schreiben. Grundsätzlich geht es jetzt so weiter, ich werde das in den nächsten paar Tagen abtippen, das Interview dann auswerten einfach.. qualitativ.. schauen was gesagt worden ist und dann quantitativ wie oft und wohin. Natürlich werde ich dein Unternehmen und deinen Namen schwärzen und alles anonymisieren. Wenn die Arbeit fertig ist, ich kann dir das gerne auch schicken.. das Transkript ist im Anhang, also wenn ihr das braucht und wollt bitte. Es ist öffentlich.

58 B: Coole Sache. Also die Arbeit selber ist auch öffentlich. Sind dann die Transkripte anderer auch öffentlich, oder sind die Teil der Arbeit, oder..?

59 I: Sie sind im Anhang ja, aber sie werden alle anonymisiert.. also ja. Ich mache es auch gar nicht anders, ich nehmen niemanden namentlich auf, denn es bringt nichts. Also Interviews werden es.. ich schätze mal 5-7 in der Größenordnung werden, weil der Hauptfokus eigentlich der Workshop ist und die Interviews eine Zusatzinformation sind, aber die sind grundsätzlich alle im Anhang ja.. muss ich.

A.23 Interview: Protokoll Unternehmen B

1 I: Ja, das wird schon funktionieren so. Gut. Ja also mein Name ist Daniel Drack, das hatten wir schon. Ich habe die letzten... also ich wohne jetzt seit zwei Jahren in Graz und hab davor meine ich fünf Jahre mit Person W zusammengearbeitet, der war zum Schluss auch mein Team Lead, so halb offiziell. Daher kennen wir uns und da bin ich auch auf ihn gekommen weil er bei euch jetzt eben soweit ich weiß auch IT Softwareentwicklungsleiter Teamleiter Position hat. Und weil ihr eigentlich für meine Arbeit relativ gut rein passt. Ich mache gerade meinen Master in Innovationsmanagement am Campus02 in Graz. Und ich habe mir gedacht da Kubernetes, vor allem in den letzten fünf Jahren, so als die neue evolutionäre innovative Technologie gehyped ist, bietet es sich an darüber zu schreiben. Außerdem arbeite ich bei der AVL vollzeit im DevOps Team, ich bin auch einer der Hauptverantwortlichen für unsere Kubernetes Umgebung. Deswegen habe ich da schon einiges an Vorwissen. Vom Modus her ist es so, ich mache für meine Arbeit einmal einen Workshop in der AVL wo ich versuche für meine Forschungsfrage interdisziplinär mehrere Bereiche aus der AVL abzuklopfen. Und zusätzlich dazu noch einige Experteninterviews, für das, wo es darum geht aus verschiedenen Sektoren und Branchen noch ein bisschen verschiedene Anforderungen an Kubernetes abzuklopfen, dass ich eine so vollständige Fragestellung wie möglich abdecken kann. Grundsätzlich geht es in meiner Arbeit darum, dass ich mir anschaue, wie kann man ein Kubernetes System sinnvoll designen. Also einerseits mal einen Prozess, beziehungsweise eine Vorgehensweise zu entwicklen, wo ich sage ich kann das strukturiert machen. Und dann nochmal wirklich basierend auf einem Anforderungskatalog, über verschiedene Fragestellungen hinweg, dies als Fundament zu nehmen um wirklich dann sinnvolle Distribution und die verschiedenen Kubernetes Teile oder Produkte auswählen zu können. Also zum Beispiel sagen zu können, basierend auf meiner Anforderung X muss ich mich jetzt für Storage System entscheiden, das Performance Faktoren erfüllt oder Encryption beherrscht oder dergleichen. Das man halt in Zukunft nicht mehr das Problem hat und so rein stolpert, sondern schon basierend auf einem guten Anforderungskatalog sagen kann, "mein System muss das erfüllen und deshalb brauche ich das und das Produkt oder die und die Distribution". Um das gehts im groben und ganzen. Ja, darf ich dich mal bitten, dass du dich mal vorstellst und mir mal erklärst was du eigentlich so machst oder was ihr als Unternehmen so macht. Und wie eure Umgebung so aussieht im groben und für was ihr Kubernetes so verwendet.

2 B: Ok... ja.. der Vollständigkeit halber, ich heiße Person A und arbeite bei Unternehmen A, jetzt seit ungefähr drei jahren. Person W ist jetzt seit ungefähr 1.5 Jahren dabei und ist unser Team Lead. Vorher hatten wir keinen Team Lead, da war etwas Chaos. Und was für dich relevant ist.. also Unternehmen A arbeitet ja im earth observation Bereich, das heißt wir arbeiten eben mit Satellitendaten. Wobei ich mittlerweile.. ich habe früher angefangen in der Methodenentwicklung, das heißt ich habe so Algorithmen implemenentiert.. ich würde nicht sagen Machine Learning.. aber in die Nähe von Machine Learning, aber einfach so Bildauswertungen. Habe dann nach ungefähr einem Jahr gewechselt in die Softwareabteilung. Zu dem Zeitpunkt gab es noch

Unternehmen

..Stellenbeschreibung

..Stellenbeschreibung

..Dauer Kubernetes Verwendung

..Motivation für Kubernetes

..Vorschlag für Kubernetes

..Technologiestack

keine Softwareabteilung aber ich habe mich eben etwas umpositioniert und mache jetzt eigentlich obwohl wir mit Satellitendaten arbeiten bin ich sehr weit weg von den Algorithmen und mache eher Backend und Frontend. Ganz klassische Softwarentwicklungsarbeiten. Vielleicht was für dich interessant ist... wir haben Kubernetes verwendet für ein ganz ein konkretes Projekt. Dabei ging es darum, dass die Firma 20 Jahresfeier gehabt hat. Dafür wollte unser Chef so eine Produkt vorstellen. Und zwar gehts da darum, das nennt sich Seasonality nenn ich es jetzt mal, das ist einfach eine Karte von Europa bei der man sieht wie die Oberfläche klassifiziert ist sag ich jetzt mal. Da sieht man ob da zum Beispiel grassland ist, also Gras, ob da Wasser ist oder ob da versiegelter Boden oder Wüste ist. Aber ich glaube des ist für dich jetzt gar nicht relevant. Und das haben wir für Europa gerechnet und das hat ein recht schönes Ergebnis geliefert. Er hat sich das dann angeschaut und wir haben das dann eben erstmals in der Amazon Cloud gerechnet, aber noch ganz simpel über Script das einfach vier Rechner startet und diese Tiles nennt sich das. Also jedes.. Also die Erde ist unterteilt in Tiles, das sind so Kacheln und das sind in Europa ungefähr 1000. Und das haben wir gemacht auf Amazon und das war für uns schon recht ein Fortschritt, weil wir es vorher immer auf einer Infrastruktur in Wien gerechnet haben die... da wo alles manuell war. Das war das erste mal der Schritt, dass wir in die Cloud gehen und das hat erstaunlich gut funktioniert. Dann war das Ergebnis von Europa da.. 1000 Kacheln und jetzt hat er irgendwie Luste bekommen und gefragt, wie würd das aussehen wenn man das für die ganze Welt rechnet. Die ganze Welt das sind 12000 Kacheln, also das ist wirklich ein großer Unterschied. Dann haben wir gesagt das schaffen wir nicht in der kurzen Zeit indem man einfach das System wie wir es jetzt haben hochskalieren, weil wir es eben händisch aufgesetzt haben. Dann haben wir einfach etwas herumgeschaut, also ich habe da wirklich... ich habe das Wort Kubernetes bis dahin noch gar nicht gekannt.. Und habe einfach mal ins Blaue gegoogled was man da machen könnte, bin da dann auf Kubernetes gestoßen. Also als Tool wie man.. mehrere Container abarbeiten könnte. Das habe ich jetzt vergessen, wir haben das in Containern gerechnet. Also jedes Tile war ein Docker Container.

3 I: Das heißt ihr habt vorher schon Container verwendet und...

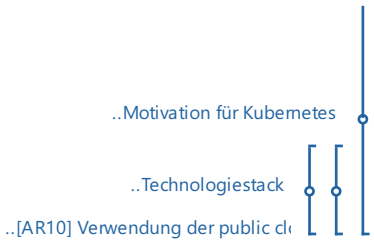
..Motivation für Kubernetes

4 B: Genau. Die Container Technologie die haben wir.. die war schon in Verwendung. Aber Kubernetes haben wir noch nie verwendet und ich sage es ganz offen, haben wir auch noch nie gehört. Vor Person W waren wir auch nicht wirklich ein IT Team, es waren nur ich und noch jemand. Wir haben so vor uns hin gearbeitet, ohne dass es jemand in eine Richtung geleitet hat. Wäre es für dich.. Möchtest du sehen was wir hier gemacht haben, damit du dir was darunter vorstellen kannst? Oder ist das für dich jetzt nicht wichtig?

5 I: Im nachhinein gerne, jetzt für die Arbeit..

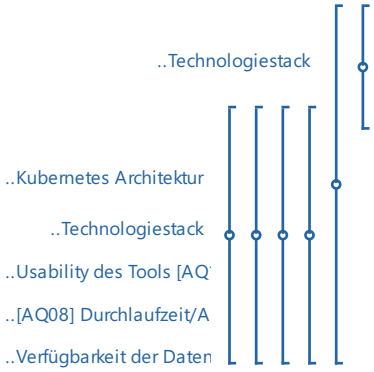
..Motivation für Kubernetes

6 B: Ich schicks dir einfach, damit du das sehen kannst. OK, wie gesagt ich hab dann da einfach auf eigene Hand Research betrieben was es hier für Möglichkeiten gibt und bin dann auf Kubernetes gestoßen. Und im Speziellen auf das Job Pattern von Kubernetes. Das heißt



wir haben Kubernetes in dem Sinn nicht verwendet wie man es klassische kenn, sondern ganz speziell dieses Job Pattern. Auf der Job Pattern Seite von Kubernetes gibt es mehrere Use-Case Szenarien und wir haben uns für das fine coarse pattern entschieden. Ja so sind wir eigentlich zu dem gekommen und vereinfacht gesagt.. wir haben einen Kubernetes Cluster in der Amazon Cloud aufgesetzt mit ich glaube das waren 40 Rechner.

7 I: EKS oder selbst aufgesetzt?



8 B: Nein mit kops haben wir das gemacht. Also wir haben zuerst einen Prototyp mit Minicube gemacht, weil wir haben wirklich null Erfahrung gehabt und haben gesagt wir probieren einfach etwas hands-on. Dann haben wir etwas herumgespielt und dann war für uns recht klar, dass wir das in der Amazon Cloud rechnen müssen, weil wir die Daten in der Amazon Cloud verfügbar haben. Das war so ein bisschen auch der Grund warum AWS überhaupt, weil dort die Daten liegen. Und da das alles schnell gehen müssen hat, haben wir gesagt, schauen wir uns mal dieses kops tool an. Mit dem haben wir dann eigentlich sehr zügig einen Cluster erstellen können, mit ein paar Command Lines. Also das kops kennst du wahrscheinlich?

9 I: Das kenne ich ja.

10 B: Zu dem Zeitpunkt hat es den EKS Service auch noch nicht gegeben in der Amazon Cloud. Sonst hätten wir den verwendet, der war da gerade irgendwo in der Ankündigung. Genau, dann haben wir mit kops einen Cluster erstellt und haben dann über dann über dieses Job Pattern unsere 18000 Berechnungen heruntergerechnet. Da dauert eine Berechnung ungefähr zwei Stunden. Und das haben wir über drei Tage geschafft, was natürlich für uns total super war, und was extrem cool war, weil es war das erste mal, dass wir das verwenden. Dass einfach von ich glaub von den 18000 Containern sind glaub ich nur zwei oder so fehlgeschlagen, was aber überhaupt kein Problem war weil das mit der restar policy super funktioniert hat. Und wir haben eigentlich Kubernetes verwenden um uns.. so haben wir es im Nachhinein öfters verwendet, wenn wir einfach immer dann wenn wir.. wir brauchen eine große Berechnung in kurzer Zeit, da haben wir es verwendet. Das heißt der Cluster der liegt bei uns, der ist ausgeschaltet und wenn wir ihn brauchen schalten wir ihn an. Da fahren wir dann diese 40 EC2 Instanzen hoch, die sind recht groß.. die haben 64 Kerne.. rechnen dann unsere Docker Container parallel so viele sich halt ausgehen. Das tun wir immer so ein bisschen händisch, das heißt da haben wir kein autoscaling sonder wir schauen uns an wie viel RAM braucht ein Container und wie viel können wir parallel ausführen. Dann rechnen wir das durch und schalten ihn manuell wieder aus. Das ist unser Use-Case.

11 I: Jetzt kurz zur Frage, du hast gesagt am Prozess beteiligt war auf jeden Fall mal ihr als Entwicklungsteam und dann euer Chef, also euer Geschäftsführer in dem Fall nehme ich an. Der wird sich dann auch dafür entschieden haben, dass ihr das verwendet. Oder habt ihr das selbst... habt ihr selbst die Entscheidungsgewalt dass ihr sagt..

..Entscheidung für Kubernetes

..[AQ08] Durchlaufzeit/Aufwand fü

..[AQ03] Preis/Lizenzkosten

12 B: Wir haben das selber. Muss man vielleicht dazu sagen, dass unser Chef vom technischen sehr weit weg ist. Ich kann mich noch gut erinnern, weil ich ein bisschen Bauchweh gehabt habe mit dem ganzen. Er hat gesagt er kann sich eh nicht vorstellen was da funktioniert, ich soll ihm nur sagen was es kostet. Er hat zu mir gesagt das darf nicht mehr als x€ kosten. Alles was darunter ist, passt. Und.. also der hat jetzt da nicht wirklich gesagt, "verwenden wir das weil es so eine coole Technologie ist". Sondern seine Vorgabe war, er möchte diese 18000 Kacheln.. also diese 18000 Rechnungen rechnen, innerhalb von vier Tagen weil das muss schnell da sein jetzt. Und wir haben x€ Budget, das war alles an Vorgabe. Dann haben wir das eben in der AWS aufgesetzt, das hat dann y€ gekostet.. wir waren also leicht darunter. Das war also das erste Mal wie wir diesen Kubernetes Cluster verwendet haben. Wir haben den seit dem vier, fünf mal verwendet.. da immer für das Gleiche eigentlich. Immer.. nicht um eine Web App zu skalieren oder wie man es so klassisch kennt, oder... ich bin ja kein Experte. Ich war auf einer Kubernetes Schulung und habe gemerkt, dass die meisten die da sind die verwenden es ganz klassisch als... einfach um.. Skalierbarkeit zu schaffen bei einer Web App zum Beispiel. Wir haben aber immer ganz speziell dieses Job Pattern verwendet.

13 I: OK.

14 B: Und seit dem drei vier fünf mal glaube ich jetzt.. und es hat immer ausgezeichnet funktioniert. Ja, seit dem, dass wir da hands-on haben wäre jetzt der Plan dass wir... wir bieten ja Services an in unserem Unternehmen. Seit dem Person W da ist hat sich das alles entwickelt. Ganz einfach du möchtest wissen... irgend ein Service als Kunde, zum Beispiel.. wir haben zum Beispiel einen Service der schaut wie oft ist eine Wiese im Jahr gemäht worden. Das ist ein leichtes Beispiel. Das heißt ein Bauer könnte sagen, er möchte wissen.. möchte wissen wie oft hat der Bauer neben mir das Feld gemäht. Weil da gibts ja Subventionen von der EU wenn man das nicht öfter als drei mal macht. Dann könnte er diesen Service bestellen. Und da brauchen wir noch kein Kubernetes, denn das sind alles so kleine Services die man in-house rechnen kann. Aber die Idee wäre jetzt in Folge, dass wir auch diesen Cluster anbinden an unser.. an unseren Servicekatalog. Und wenn jetzt jemand sagt er möchte eine große Berechnung bei uns bestellen... also groß im Sinne von großer Fläche. Dann wäre die Idee, dass wir wieder auf den Kubernetes Cluster zurückgreifen und dort rechnen. Und da sind wir aber noch ein bisschen in der Planungsphase ob wir das wieder über das Jobpattern machen, ob wir vielleicht.. ob man vielleicht eine andere Möglichkeit findet. Aber bei uns ist das immer so eine on-demand Sache. Das heißt der rennt nicht immer, sondern wir wissen wir brauchen ihn für Berechnungen und schalten ihn an. Skalieren den sogar händisch, rechnen einfach durch wieviele Rechner brauchen wir, rechnen die Docker Container ab und schalten ihn wieder aus. Nur, dass du mal eine Vorstellung hast wie unser use-case ist, so grob.

..Motivation für Kubernetes

..Art der Workload

15 I: Hat es grundsätzlich noch andere Rahmenbedingungen gegeben, außer dass ihr sagt ihr habt die Daten in AWS und das Budget von x€? Also gerade in Richtung.. gibt es bei euch rechtliche Vorschriften oder.. keine Ahnung, wo du sagst das ist indiskutabel?

..politische Vorgaben/Herstellerstand
..[AR02] Availability/Lokation

16 B: Nein, es ist vielleicht immer ein bisschen weil wir im EU Sektor arbeiten. Also das war jetzt wirklich ein privates Projekt, wirklich für den Showcase, zum Herzeigen.. "Schaut euch an, das ist was wir gemacht haben". Sonst bei den Projekten haben wir oft mal.. das kann ich nicht.. da habe ich nicht so die Einblicke, ich bekomme es am Rande mit, dass es oft EU Projekte sind. Und wir diesen Cluster in der AWS betreiben, da ist halt of... dass wenn man ein Proposal schreibt und... sollte man jetzt vielleicht nicht rein scheiben man verwendet AWS Technologie, weil die das nicht so gerne haben, wenn es um ein EU Projekt geht. Die wollen immer, dass man auf europäischen Clouds arbeitet, aber dort haben wir noch nie so einen Cluster aufgesetzt.

17 I: Ist das also ein Problem klassisch nach DSGVO, wo du sagst die Daten müssen in Europa liegen, oder einfach weil es ein amerikanisches Unternehmen ist?

..politische Vorgaben/Herstellerstand
..[AR07] rechtliche Vorgaben

18 B: Nein, das ist wirklich nur politisch, dass die.. dass die einfach wollen, dass man Sachen verwendet die sie schon.. Es gibt zum Beispiel... Mundi nennt sich das, das sind so in der Telekom Cloud.. so wie ich das versteh... auf der Telekom cloud darauf ein Portal in dem diese Erdbeobachtungsdaten zur Verfügung gestellt werden. So wie ich das verstehe, wenn man eben in ein Proposal rein schreibt, man verwendet jetzt zum Beispiel Mundi, das kommt eben aus Deutschland und von einem europäischen Konsortium. Das ist jetzt nur so wie ich das wahrnehme... dann mögen die das einfach lieber, weil es einfach so gesehen in Europa bleiben soll. Und da.. Google Cloud oder Amazon Cloud eher ein bisschen als Konkurrenz wahrgenommen wird. Das ist aber... Das ist in manchen Projekten wirklich eine Vorgabe, was ich so mitbekommen habe, da darf man das nicht verwenden. In manchen halt, wo man schaut... ja.. schreibt man es jetzt nicht rein, dass wir das verwenden wollen, weil es wahrscheinlich nicht so positiv aufgefasst wird. So verstehe ich das jetzt, aber das ist wirklich eher politischer Natur und nicht Datenschutz.

19 I: Perfekt, genau die Info die ich.. ich will nicht sagen die ich wollte, aber die definitiv brauche.

..[AQ04] Betriebspersonal Aufwan
..[AQ08] Durchlaufzeit/Aufwand fü
..Qualifiziertes Personal [AQ11]

20 B: Ich habe dann immer die Diskussionen geführt: "Können wir das nicht in der T-Mobile Cloud aufsetzen?". Da ist halt immer meine Sache: "Ja können wir vielleicht". Man kann ja so einen Cluster wahrscheinlich manuell aufsetzen, ich hab das noch nie gemacht. Aber es ist halt oft bei uns so.. Wir sind ein kleines Team, es muss schnell gehen und da ist halt einfach die Amazon Cloud besser. Da geht es einfach zack zack und man hat das einfach in kürzester Zeit aufgesetzt. Da ist halt immer ein bisschen die Diskussion: "Wieso können wir das nicht überal machen?". Wir haben zum Beispiel auch ein Rechenzentrum in Wien, das von uns mitbetrieben wird und da wollten sie immer wieder mal dort was rechnen über Kubernetes. Da haben wir lange gebraucht, dass verstanden wird, dass dieser Cluster nicht so einfach betrieben werden kann. Also wenn man ihn manuell aufsetzt. Dass wir deshalb so sehr an die Amazon Cloud gebunden sind.. im Moment. Es liegt also an uns, nicht am technischen.

21 I: Zu den funktionalen Anforderungen. Du hast schon gesagt ihr rechnet.. ihr habt im Prinzip nur Berechnungen also Container die gleich funkionieren. Ich glaube du hast gesagt ihr habt da Python?

22 B: Genau. Also es ist verschieden.. also es läuft meistens so ab, dass ich vom Methodenteam Code bekomme, den ich in einen Container packe. Das ist of Python, das ist manchmal R. Für diese große Berechnung war es C++. Also es ist bei jedem Produkt das wir generieren unterschiedlich, aber die Vorgehensweise ist immer gleich. Ich bekommen den Code, ich packe den in einen Container, schau dass dieser lauffähig ist. Die Datenanbindung habe ich bisher immer über S3 gemacht, das heißt... weil die Daten die wir brauchen als Input die liegen auf S3 auf der Amazon Cloud. Das sind Satellitenbilder. Was natürlich nochmal für die Amazon Cloud spricht, weil du die Daten natürlich extrem schnell da hast. An dem Punkt wo ich den Container habe, da war es mir dann immer egal was die Technologie ist von der Methode. Also Python, C++ und R, die drei sind so Merkmale. Manchmal ist es auch Julia, aber eher selten. Und wir packen das in einen Container und den lassen wir dann auf Kubernetes laufen. Die Daten kommen immer über S3 rein und die Ergebnisse speichern wir auch wieder auf ein eigenes S3 Bucket. Also das ist Daten Input und Output.

..Technologiestack

..Art der Workload

23 I: Noch zwei technisch Fragen: Grundsätzlich nehme ich an Linux Container, also keine Windows Container?

24 B: Ja, das waren immer Linux Container.

..[AF05] Betriebssystem für Container

25 I: Und bei S3, mounted ihr das wirklich als Filesystem, oder ist das etwas das sich die Applikation einfach als REST Endpoint konsumiert.

26 B: Ich habe es so gemacht, dass ich in diese Container immer die Amazon CLI installiert habe. Und da einfach über die S3.. da habe ich ein paar kleine Bash Scripte, oder Python, ist verschieden. Die mir einfach wirklich von S3 herunterladen, also wir haben hier nichts hineingemounted. Also ganz einfach gesagt, der Container macht ein AWS S3 copy und kopiert die Satellitendaten runter und dort werden sie verwendet. Wenn man es nicht verschieden macht, manchmal müssen auch Daten gelöscht werden, oder wenn eine Berechnung [..] dann sind die Daten sowieso weg, weil der Container stirbt. Bevor ich es vergesse, das hat uns auch vor.. das war eines der größten Probleme anfangs. Weil in der.. Wenn man diesen Cluster default aufsetzt mit kops, dann hat man einen shared Speicher von 120GB, soweit ich das im Kopf habe. Was bei uns oftmals zu wenig war, weil dieser Input für eine Berechnung um die 200 GB Daten sind, das hat dann nicht funktioniert.

..[AF15] Persistenter Speicher

27 I: Pro Container oder insgesamt?

28 B: Nein, pro Container.. also pro Kachel, von diesen 18000 Kacheln. Also das sind wirklich oft so Satellitendaten, zum Beispiel Innsbruck ist ein Tile kann man so sagen, mehr oder weniger. Das sind 10000 mal 10000 Meter und das haben wir zum Beispiel einmal gerechnet, Satellitendaten über drei Jahre circa. Das waren einfach mehr als

120GB pro Container eben, deswegen haben wir da Probleme gehabt. Wir haben die dann so gelöst, dass wir die Datenmenge reduziert haben. Weil, vielleicht.. ich weiß nicht ob es für dich relevant ist, wir haben jetzt bei Amazon angeschaut, dort gibt es die.. kann man sich Volumes dazu mounten. Da hatten wir aber das Problem, da kann man glaube ich nur ein Volume pro... die Details kenne ich nicht mehr, aber der Speicher vom Cluster, also wirklich der Speicher an Daten die man hat, der war oftmals für uns ein Problem. Wo wir jetzt auch noch keine Lösung gefunden haben, falls dies mal... ich meine größer werden wir nicht mehr werden wie das. Das war jetzt schon das worst-case Szenario, aber wenn wir jetzt eine Anforderung hätten über fünf Jahre so eine Zeitserie zu rechnen, dann wüssten wir jetzt nicht.. dann müssten wir uns was überlegen wie wir hier mehr Speicher rein bekommen. Also ich... Es gibt da schon Möglichkeiten, aber es ist halt immer eine Geldfrage.

29 I: Ja klar. Rechnet ihr nur auf CPUs, oder habt ihr auch irgendwie GPU Support?

..[AF04] GPU Support

30 B: Bei unseren Kubernetes Geschichten haben wir bisher immer auf der CPU gerechnet.

31 I: Wäre das etwas wo ihr sagt das könnte oder das wolltet ihr nutzen wenn es gehen würde? Also es geht natürlich.

..[AQ03] Preis/Lizenzkosten

..[AF04] GPU Support

32 B: Ja, wir haben Leute die auch Algorithmen auf der GPU rechnen. Das ist sicher interessant, nur so im Hinterkopf bei mir wäre da jetzt, dass es wahrscheinlich zu teuer ist. So was ich weiß von Amazon sind die GPU Stunden wahrscheinlich teuer. Deswegen glaube ich, werden.. Aber ja, kann durchaus mal vorkommen, dass das dann auch benötigt wird. Bis jetzt noch nicht. Das war bis jetzt noch kein use-case.

33 I: OK, dann kenn ich mich soweit aus. Letzter Punkt: Qualitätseigenschaften. Habt ihr irgendwie.. du hast ja mal gesagt ihr habt euch mehr oder weniger so durch Vorgaben gedrungen schon für AWS entschieden, und das funktioniert auch gut. Habt ihr irgendwelche bestimmten KPIs wo ihr sagt da könntet ihr unterscheiden ob jetzt... AWS oder Azure zum Beispiel sinnvoller wäre. Ich meine, Geld natürlich ja, aber gibt es sonst auch noch irgendetwas wo ihr sagt ihr habt irgendwelche Qualitätseigenschaften wo ihr die verschiedenen Kubernetes Komponenten unterscheiden könntet, oder auch die Distributionen? Also als Denkanstoß vielleicht eben Geld, oder ich habe es schon von anderen gehört, dass die die Community Size, also "wie groß ist die Community für ein Produkt?", essenziell ist. Oder wie du schon gesagt hast, wie lange es dauert bis man zum Fliegen kommt mit einem Produkt, also Zeit in dem Fall von "Time-To-Get-Started". Solche Eigenschaften, irgendwas wo ihr sagt es ist für euch essenziell gewesen, oder könnte essenziell sein?

..[AQ08] Durchlaufzeit/Aufwand für Insta

34 B: Also in der Vergangenheit war immer der Zeitfaktor essenziell. Das heißt da hat es schnell gehen müssen, also "wie kommen wir am schnellsten zum Ergebnis?". Für das was wir in Zukunft möchten, dass wir das in unseren Servicekatalog integrieren möchten, da.. so richtig Vorgaben haben. Vielleicht, kannst du bitte nochmal die

Frage wiederholen? Ich glaube ich habe das jetzt nicht ganz verstanden.

35 I: Es geht darum, wenn du mehrere Kubernetes Distributionen hättest, oder pro Distribution dann..

36 B: Was verstehst du unter Distribution?

37 I: Naja du könntest sagen du nimmst Upstream oder du nimmst OpenShift, oder ich macht mit kops auf Amazon oder ihr nehmt EKS oder so. Also wie ihr unterscheiden würdet für welche ihr euch unterscheidet? Und dann auch in den jeweiligen Komponenten, weil du hast ja verschiedene Storage Systeme, verschiedene Netzwerkkomponennten, Service Meshes etc. . Wie ihr euch vorstellen könntet euch für eine.. für eines davon zu entscheiden?

..Verfügbarkeit der Daten [AR11]

..Usability des Tools [AQ19]

..[AQ09] Schulung Admins

..[AQ04] Betriebspersonal Aufwan

38 B: OK. Also ich glaube bei uns ist immer ein ganz... wenn wir uns entscheiden für eine Berechnung auf Kubernetes, ist das erste und wichtigste, oder eines der wichtigsten Dinge die wir berücksichtigen ist immer die Datenverfügbarkeit. Daten sind oft nur auf Amazon verfügbar, oder sind oft auf Amazon verfügbar.. gibt es auch anderswo. Das heißt da schaut man immer als erstes "wo sind die Daten?". Weil wenn wir zum Beispiel einen Cluster aufsetzen würden in Wien in unserem Rechenzentrum, dann.. und die Daten dort nicht haben, müssten wir erst mal die ganzen Daten von AWS herunterladen, was viel zu teuer ist mit dem outbound traffic. Das heißt da sind wir oft schon eingeschränkt, und sonst würde ich sagen, da wir kein ganz klassisches IT Unternehmen sind und eine relativ kleine IT Abteilung haben, sind wir immer interessiert an Lösungen die einfach sind. Deswegen haben wir jetzt zum Beispiel für die.. der Plan geht jetzt mal dahin, dass wir den Kubernetes Cluster in unser Serviceportfolio integrieren, dass wir hier EKS von Amazon verwenden. Und der Gedanke ist einfach der, dass wir uns erhoffen, dass das am wenigsten overhead in der IT verursacht. So nach dem Motto: "Das ist ein Service von Amazon, der wird schon funktionieren". Währenddessen wenn wir das aufsetzen würden mit.. wie du gemeint hast, mit anderen Technologien, was vielleicht mehr Input von uns braucht, einfach, dass der Cluster funktioniert, dann glaube ich, dass das für uns nicht so interessant ist. Weil wir wollen hier nicht fünf Leute haben, die sich mit Kubernetes auskennen. Im Gegenteil, wir haben da jetzt mich, und ich kenne mich nicht aus, sondern ich habe es einfach mal etwas verwendet. Das heißt.. ich war zum Beispiel auf einer Fortbildung und das war interessant, weil da jemand war vom Verbund. Der hat mir erzählt die haben drei, vier Leute die nichts anderes als Kubernetes machen den ganzen Tag. Das ist dann natürlich ganz etwas anderes, weil dann kannst du natürlich so einen Cluster sicher selber betreiben. Mit allen Vor- und Nachteilen, aber ich glaube bei uns ist immer einfach auch zu berücksichtigen, dass wir ein kleines Team sind und hier nicht die Ressourcen haben, dass jetzt da einer drei Wochen mit diesem Cluster herumspielt. Weil das einfach zu viel kosten würde.

39 I: Das macht absolut Sinn. OK, dann von offizieller Seite wäres das mal. Grundsätzlich vielen Dank, dass du dir die Zeit nimmst. Echt super! Sehr guter Input, also vor Allem auch aus eurer Sparte als

kleines Unternehmen. Wie gesagt, ihr seid jetzt kein Standardunternehmen, echt lässig. Auch, dass ich zum Fliegen gekommen seid. Ich würde dir dann bitte noch eine Einverständniserklärung schicken, einfach in gegenseitigem Interesse. Da steht drinnen, dass ich die Daten anonymisieren werde, also das transkribieren. Den Mitschnitt werde ich dann löschen nach der Masterarbeit. Ich nehme deinen Namen und den Unternehmensnamen raus, das mache ich bei allen Interviews, damit hier alle gleichbehandelt sind. Ja, ich werde das transkribieren, werde es auswerten. Die Arbeit ist dann öffentlich, also wenn es euch interessiert oder wenn ihr Fragen habt, jederzeit bitte melden.

A.24 Interview: Protokoll Unternehmen C

1

I: Wie gesagt Studium berufsbegleitend, in meiner Arbeit geht es darum, dass ich mir angeschaut habe, was bzw. wie könnte man eigentlich den Design Prozess von so einem Kubernetes Cluster und so einer Umgebung etwas strukturierter angehen, weil man immer wieder so das Gefühl hat man probiert halt man und man stolpert mal und dann purzelt hinten irgendwann einmal das System raus nach ein oder zwei Jahren, das man dann halt verwendet. Aber um Zeit und Ressourcen zu sparen wäre meine Herangehensweise zumindest eine strukturierte Vorgehensweise zu finden. In der Theorie bin ich dann drauf gekommen, dass das dann eigentlich relativ stark auf den Anforderungen basiert die man an das System hat. Deshalb ist eigentlich der Hauptfokus der Arbeit die Erhebung von so einem Anforderungsbogen bei dem ich mittlerweile an die 40 Fragen habe. Damit man so genau wie möglich spezifizieren kann, was brauche ich eigentlich damit ich den Cluster dementsprechend designen und schneiden kann. Ich mache dann dazu in der AVL noch einen Workshop um noch verschiedenste Perspektiven reinzubringen: also Lizenzmanager, Projektmanager, Software Entwickler etc. um das wirklich holistisch zu betrachten, und ihre Meinung und die Experten Interviews bräuchte ich dann um verschiedene Branchen abzudecken. Die AVL ist relativ automotiv getrieben und teilweise ziemlich hart in der Softwareentwicklung, vor allem Simulationssoftware, und jetzt wäre es natürlich interessant was andere Sektoren für Anforderungen haben an Kubernetes, um das dann auch gesamtheitlicher betrachten zu können. So viel von meiner Seite.

2

..Stellenbeschreibung

..Motivation für Kubernetes

..[AQ09] Schulung Admins

B: OK, dann stell ich mich kurz vor - meinen Namen kennen sie schon. Ich bin die letzten Jahre Leiter für den Bereich Smart Technology bei der Firma X. Firma X ist ein Tochter Unternehmen von der Unternehmen Y Gruppe, deswegen sind wir bei euch zu dem Kontakt gekommen. Gut ich war damals, ich habe dort die letzten vier Jahre eigentlich den ganzen IT Sec aufgebaut und Team, ich war der erste bei uns im Unternehmen im Bereich Digitalisierung. Mittlerweile waren wir ungefähr 35 Mitarbeiter. Es hat leider das Unternehmen etwas darunter gelitten durch die ganze Corona Krise, es sind leider ein paar Leute in der letzten Zeit entlassen worden. Nichts desto trotz ist die Digitalisierung bei uns trotzdem noch wichtig. Vielleicht ein Satz zu dem was ich vorher gesagt habe mit dieser Server Geschichte oder so, ich muss ehrlich zugeben, wo ich drüber nachgedacht hab, das war bei uns am Anfang auch etwa so, weil es einfach ein komplett neues Thema war, wo wir wirklich bei null angefangen haben und ich bin heute aber auch froh darüber, dass wir viele Sachen auf Kubernetes Ebene oder auch jetzt mit anderen Technologien, Docker Container und solchen Sachen so lernen konnten, weil es hat uns innerhalb kürzester Zeit ein wesentlich besseres Verständnis für alles gegeben. Es war zwar ein harter und stressiger Weg, aber wir haben, also wie gesagt, wir sind leider am Anfang den Weg auch so gegangen. Ich denke wir können ja bei den Punkten, ich habe mir Punkte so durchgegangen die sie mir so geschickt haben, ich habe mir so ein paar Sachen so dazu notiert, was kann ich zu uns sagen. Unternehmen X ist der größte X Hersteller weltweit mit mehreren Tausend Mitarbeitern, mir ist auch ihr Unternehmen bekannt, weil ich glaub, ihr CEO oder was, ist glaub, ich bei uns im Aufsichtsrat, wenn ich es richtig weiß.

3 I: Möglich ja.

4 B: Ja ich war auch überrascht wo ich gesehen hab von welcher Firma sie kommen. Da war ich dann etwas überrascht. Irgend jemand, ich glaube irgendein Herr Y.

5 I: Ja das ist unser Eigentümer.

6 B: Ja insofern wir sind eigentlich, wir sind eigentlich komplett bei null gestartet. Wir haben uns in den letzten zwei Jahr sehr viel mit Unternehmen Y zusammengesetzt, weil es denen ja genauso auch ging und haben dann immer regelmäßige Workshops gehabt die aber mehrere Tage gingen und unser ganzes Know How auf die Schnelle auszutauschen oder Erfahrung auszutauschen und in die gleiche Richtung zu ziehen. Wir haben auch viel Learning by Doing gemacht. Es war nicht so, dass da irgendwelche großen Vorkenntnisse da waren. Zu unserer eigenen Thematik, wir sind auch überwiegend autmobilgetrieben, allerdings auch unter anderem der größte X Hersteller, also 99% der Y aller Welt werden mit Unternehmen X gemacht. Ansonsten, wie gesagt, die ganzen großen Automobiler sind alles Kunden von uns und die ganzen Zulieferbetriebe, und das ist auch etwas was unsere ganze Architektur und das ganze was wir aufgebaut haben mit beeinflusst hat. Das eine ist also wir haben uns zum Beispiel am Anfang schon entschlossen, dass wir eine Private Cloud machen bei uns im Haus im Rechenzentrum. Sie unterbrechen mich wenn irgendwas ist wo sie Fragen haben.

..Qualifiziertes Personal [AQ11]
..[AR10] Verwendung der public cloud
..Kubernetes Architektur

7 I: Also wie gesagt, es wäre interessant wie ihre Umgebung generell ungefähr ausschaut, natürlich abhängig davon wie viel sie mir sagen können und wollen.

8 B: Also bei dieser Private Cloud das hat einfach auch den Grund gehabt, dass viel von unseren großen Kunden, sie wollen nicht, dass ihre Daten dann irgendwo bei irgendeinem Cloud Provider sind und dann ist zum Teil natürlich auch die Problematik, dass dann auch zum Beispiel Firmen wie A oder sowas, die haben natürlich auch irgendwo bei Azure oder AWS ihre Clouds. Andererseits haben wir aber auch kleinere mittelständische Zulieferbetriebe die gar keine eigene Cloud aufbauen können, weil sie einfach das fachliche Know How gar nicht haben. Insofern waren wir eigentlich von zwei verschieden Anforderungen getrieben, das eine, dass wir natürlich global aufgestellt sind wir sind überall auf der Welt verbreitet, da kommt dann nachher auch in ihrer Fragenliste noch ein Punkt, weshalb ich das gerade nochmal erwähne, und das andere eben auch vor Ort bei uns mit eigenen Kräften den kleinen mittelständischen Unternehmen auch was anbieten zu können. Wir sind damals hergegangen und haben uns ein paar Dinge auf die Fahnen geschrieben die wir machen wollten, und zwar, dass wir das Ganze hochverfügbar machen, dass wir das Ganze modular aufbauen, modular mit dem Gedanken, dass wenn wir später mit dieser Lösung die wir uns aufbauen zu AWS oder zu Azure gehen wollen, dass wir das da auch überall nehmen können und das drittte war, dass wir versucht haben Open Source einzusetzen wo es nur geht. Es gab dann nachher auch so zwei Software Bereiche wo

..Kubernetes Architektur
..[AR10] Verwendung der public cloud
..[AR02] Availability/Lokation
..Kubernetes beim Kunden
..[AQ02] Verfügbarkeit in %
..Vendor Lock-In [AR16]
..[AR01] Open-Source Lizen:
..[AR03] Enterprise Support

..[AR03] Enterprise Support

..Technologiestack

..[AR09] Strategesche Partner

..[AQ02] Verfügbarkeit in %

..HPA/Skalierung

ich gesagt habe, ok wenn Open Source, ich meine bei Open Source kann man ja trotzdem auch Support bekommen, das lassen die sich dann halt auch entsprechend zahlen. Es gab zwei Dinge, das waren zum Beispiel die Datenbanken wo wir das gemacht haben und ein weiteres Produkt wo es um unsere Streaming Analytics geht. Grundsätzlich haben wir angefangen mit Rancher und Kuberentes das ganze aufzusetzen auf SUSE Linux Servern, die sind einfach bei uns Voraussetzung gewesen, wenn wir einen Vertrag mit der IBM zum Beispiel haben der über mehrere Jahre geht. Dann war eben das Thema mit der Skalierbarkeit, Modularität und andere Punkte und zum Beispiel auch Zero Down Time war so ein Thema also, dass wir gesagt haben mit Docker Containern und Kuberentes können wir solche Sachen gut machen und auch relativ schnell. Dann kam natürlich das Thema auf, wo kriegen wir die Daten her, das heißt, wir brauchten irgendwo einen Broker bei uns zum einem mal und zum anderen haben wir an den Maschinen, wir können theoretisch bis zu 2000 Sensorwerte abfragen, wenn wir wollen.

9 I: kurze Zwischenfrage: Sie haben hauptsächlich Applikationen die Messdaten abholen und verarbeiten? Oder was kann ich mir da vorstellen?

10 B: Im Moment sind das Daten die wir irgendwelche Motortemperatur oder irgendwelche Stössel Einstellungen oder eigentlich alles was wir aus der Steuerung rausholen können, also wir sind wirklich hergegangen, wir nehmen wirklich alles weil wir am Anfang natürlich das Problem hatten, dass wir gar nicht wussten, was können wir überhaupt mit den Daten machen. Aber durch das alles war natürlich auch ein Punkt der mir am Anfang schon immer aufgefallen ist. Ich habe früher sehr viel mit großen Datenbanken gearbeitet und mit der Performance gab es dann manchmal ein Problem, sprich eine Abfrage konnte bis zu vier Stunden laufen, deshalb habe ich von Anfang an drauf hingewiesen, dass wir auch mit der Speicherung von den Daten uns etwas vernünftiges überlegen müssen, dass wir auch skalierbar sind, dann natürlich auch, da wir internationale Kunden haben, haben wir die Voraussetzung, dass wir Daten auch von Kunden in den USA auch in USA abspeichern müssen, in Europa in Europa und von Asien eben in Asien. Deshalb auch das ganze, dass wir von vornherein darauf achten mussten, dass wir zu anderen Providern wechseln konnten.

..[AR02] Availability/Lokation

..HPA/Skalierung

..[AR07] rechtliche Vorgaben

11 I: Sie haben dann wirklich auch bei jedem Kunden dann auch einen Cluster stehen, verstehe ich das richtig?

12 B: Nein sie meinen, dass wir für jeden Kunden bei uns auf dem Cluster?

13 I: Verstehe ich das richtig, dass sie auch bei Kunden, On Premise, dann Cluster hinstellen? Also in deren Werk.

..Kubernetes beim Kunden

14 B: Momentan noch nicht, aber ich denke das wird kommen. Wir machen es wie gesagt ungefähr seit zwei Jahren, vor zwei Jahren haben wir mit der Klar Thematik angefangen, also haben wir erstmal geschaut wie können wir die ganzen Daten bekommen und wie können wir die ganzen Daten transportieren. Das sind bei uns

..Dauer Kubernetes Verwendung

..[AF12] Layer 4 / Load Balance

..[AF12] Layer 4 / Load Balancer
..alte TLS Versionen
..Art der Workload
..Technologiestack

zu dem Zeitpunkt ganz klar Themen gewesen wie entweder MQTT als Protokoll, deswegen auch vorher das mit dem Broker und eben HTTPS für irgendwelche API's im ersten Schritt mal, und wir haben auch zum Beispiel, dass ich jetzt nicht zu arg abdrifte, wir haben zum Beispiel festgestellt, dass wenn wir eine Maschinenlinie haben wo wir die Sensordaten abfragen, da sind meistens sechs Maschinen drinnen und verschieden andere Sachen wie Tranfers und solche Dinge, dann kann es eben sein, dass wir bis zu 70 MB an Daten pro Sekunde haben und dadurch war natürlich auch der Punkt, dass wir hergegangen sind und gesagt haben für jede einzelne Maschine haben wir einen Broker gemacht, wir haben seinerzeit dann auch einen zentralen Sammelbroker gemacht, wo wir die Daten auch schon On Edge beim Kunden im Prinzip abfragen können und analysieren können und wir wussten aber es werden halt sehr viele Daten und das war dann auch was, was uns dann in dem Cloud Bereich beeinfluss hat, deshalb haben wir gesagt wir schauen uns Kafka an, deswegen sind wir dann eigentlich auf Kafka noch. Als Datenbank, wie gesagt, ich habe früher schon mit fast allen großen Datenbanken gearbeitet im relationalen Bereich und habe da auch schon bestimmte Probleme gesehen mit der Skalierbarkeit und allem und deshalb haben wir uns dann für Mongo DB als SQL Datenbank entschieden und gut dann natürlich noch zusätzliche mit Docker Containern die ganzen Micro Services und so Sachen wo wir dann auch unsere Front End deployen. Wir haben auch im Augenblick zwei, drei Anwendungsthemen, das eine ist in unserem IOT Bundle wo wir da eben diese Maschinendaten unseren Kunden zur Verfügung stellen, das andere wäre dann zum Beispiel ein Visual die Protection, wo wir zum Beispiel mit KI unsere Werkzeuge beobachten, ob zum Beispiel was rein fällt, oder ein Blech drinnen hängt und solche Sachen, des Dritte wäre dann eigentlich Augmented Reality Thema wo wir eben aber noch am Anfang sind.

15 I: Darf ich zwischenfragen? Wer hat grundsätzlich mit der Idee angefangen, oder wo ist das hergekommen das Thema Kubernetes.

..Vorschlag für Kubernetes

16 B: Von mir alles. Das war ein Thema das direkt vom Vorstand kam wo man gesagt hat man möchte digitalisieren. Ich bin, also Software entwickle ich schon über 30 Jahre, in der WEB Welt bin seit ca. 96, also am Angang vom Internet war ich schon dabei. Ich war damals schon einer der ersten Netscape Partner die damals eigentlich den Browser auf den Markt gebracht haben, hab die ganzen Web Technologien kennengelernt und hab unheimlich viel WEB und Mobile Entwicklung gemacht seit 96 auch Consulting, über 15 Jahre, wo ich selbständig war und ja aufgrund von diesen ganzen Erfahrungen die ich da gemacht habe, habe ich eigentlich verschiedene Ideen gehabt und man hat mir dann damals auch die Möglichkeit gegeben, das dann auch mal umzusetzen und zu machen.

..Vorschlag für Kubernetes

17 I: Das heißt, man kann also plakativ sagen es kommt definitiv aus der Entwickler Riege und nicht vom Management, ganz salopp formuliert.

18 B: Ja auf jeden Fall, also ganz ehrlich wir haben heute noch den Punkt, dass es der Großteil der Mitarbeiter im Unternehmen mit

..Vorschlag für Kubernetes

Digitalisierung nichts anfangen können und sagen, wozu brauchen wird das. An unserem Standort haben wir ca. 1300 Leute und selbst unser CEO hat gesagt, wenn es 200 Leute sind die sagen, OK ja brauchen wir, der Rest ist einfach noch gar nicht so weit. Entweder weil sie es nicht kennen, weil sie nicht wissen was sie damit machen können. Also ich war sehr viel sozusagen regelrecht als Evangelist unterwegs und musste immer sehr viel Überzeugungsarbeit leisten und hab dann einfach auch mal Dinge probiert. Das ging bei ganz banalen Sachen los, zum Beispiel wie binde ich eine Steuerung an, wo es damals geheißen hat – selbst von Siemens Seite – also Siemens Steuerung das geht nicht und mit einem Kollegen zusammen haben wir das dann trotzdem gemacht und es ging dann und zwar war damals Ziel war OPCUA Daten abzufragen, haben wir eben gemerkt, dass das von der Geschwindigkeit nicht so ist wie wir das brauchen, also hätten wir haben maximal 50 Millisekunden hinbekommen, wir haben aber manche Daten wo wir damals gesagt haben die könnten auch im zwei Millisekunden Takt anfallen und das war auch ein Grund warum wir so viele Broker eingesetzt haben, weil wir von Anfang an geschaut haben, dass wir die Daten entsprechend verteilen und asynchron laufen lassen können und ja und eben dann auch die ganze Kafka Geschichte das direkt analysieren können.

19 I: Also mit dem Durchsatz gibt es eh fast keine andere Lösung. Alles andere wäre falsch. Sie haben gesagt, sie haben dann einmal angefangen mit dem Design von Kubernetes und von Ihrer Umgebung. Haben Sie dann da ihr Team gehabt, oder waren da auch noch andere Parteien vom Unternehmen beteiligt?

..Dauer Kubernetes Verwendung

20 B: Wie gesagt ich war ganz an Anfang komplett alleine. Ich hab dann vor zwei Jahren das erste mal noch Mitarbeiter dazu bekommen und wir sind dann praktisch so in diesen – ja jetzt November zwei Jahre ungefähr – sind wir dann ein Team geworden mit 35 Mitarbeitern. Damals war ich eigentlich der einzige Software Entwickler in dem Sinne beim Unternehmen, also es gab Steuerungsentwickler, aber sonst keinen Software Entwickler und die Kollegen waren alles bloß Maschinenbauingenieure. Mittlerweile haben wir glaub ich noch zwei Maschinenbauingenieure bei uns im Bereich, der Rest sind glaub ich alles Entwickler, oder Solution Architekt oder alles Mögliche in der Richtung, aber eben alles digitalisiert. Man hat auch den damaligen, wir waren ein CTO Bereich und haben die ganze Forschung und Entwicklung für die Maschinen gemacht , der wurde aufgelöst und man hat den Digitalisierungsbereich daraus gemacht. Also es hat eine unheimliche Dynamik gehabt. Man hat dann auch versucht ein IOT Product Home aufzusetzen und zwar wo wir das Thema agile Software Entwicklung das überhaut erste mal ins Unternehmen gebracht und das praktisch dann auch mit mehreren agilen Teams auch über verschiedene Bereiche zu platzieren. Also sprich das, es waren dann zum Beispiel in unsrem Automotive Bereich, die haben noch nie etwas mit agilen Sachen zu tun gehabt. Da hat man gesagt, wenn ihr was macht, dann agile Projekte und haben uns da entsprechend beraten lassen und unterstützen lassen von Externen am Anfang. Natürlich auch Product Owner und über was weiß ich was alles und die musste man natürlich schulen und alles Mögliche.

21 I: Das heißt bei Ihnen war das nicht nur ein Teil des Kubnernetes, sondern wirklich in den letzten Jahren eine komplette Transformation in Richtung agilen Micro Service von der Software bis zum Prozess?

..Motivation für Kubernetes

22 B: Komplett ja. Der erste große Meilenstein hat so etwa eineinhalb Jahre gedauert. Also so im November vor zwei Jahren, kam das Thema von einer Messe zurück, dass die Kunden gesagt haben, wir wollen eigentlich mobil die Information haben. Davor hatten wir gedacht man könnte das am Desktop lösen irgendwo. Dann hatten wir auch eine Lösung von Unternehmen Y seinerzeit und dann hat man aber gesagt, nein wir brauchen es am Tablet oder Smart Phone. Dann habe ich glaub ich innerhalb von sechs Wochen den ersten Stack einmal aufgesetzt und haben wir mal einen POC gemacht, wo wir zeigen konnten, dass wir die Maschinendaten hoch kriegen auch mit entsprechende Geschwindigkeit. Dann kam eigentlich das OK, dass wir gesagt haben, dass wir dann gesagt haben OK jetzt weiter machen wir den ersten POC mit einem Kunden. Haben dann live eine Maschine hier aus Deutschland angebunden und haben dann die ersten Daten dann übertragen. So hat es sich bei uns Stück für Stück entwickelt. Das waren dann von dem ersten Stack bis zu dem Proof of Concept wo wir das erste mal dort war, waren ungefähr fünf bis sechs Monate.

23 I: Wie sind Sie das dann aus Kubernetes Sicht angegangen, also wie haben sie sich für Distribution entschieden und gesagt unsere Architektur schaut so aus? War das einfach mal Try and Error oder wie sind sie das angegangen.

..Entscheidung für Kubernetes

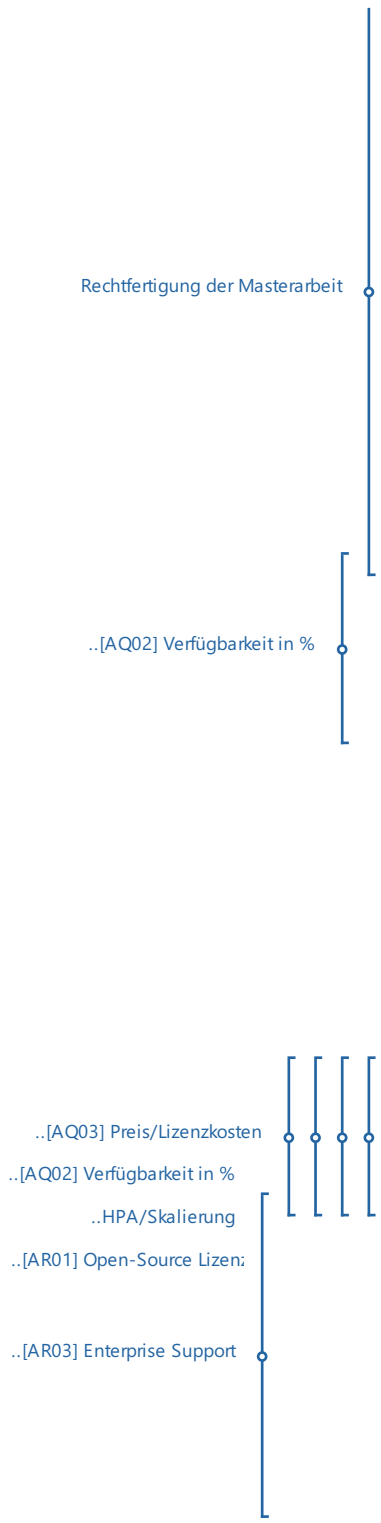
..Qualifiziertes Personal [AQ11]

24 B: Das eine war, dass, ich war früher wie gesagt in der Android Entwicklung unterwegs und mit Google Themen relativ vertraut insofern war mir Kubernetes schon mal ein Begriff. Ja ich hab sehr viel mir einfach angelesen, ich hab mir massenhaft Bücher in den letzten zwei Jahren zugelegt. Ich schätze mal mindesten 70 Stück oder so und habe halt eben, wie gesagt ich hatte ja zu diesem Zeitpunkt noch nicht einmal ein richtiges Team ich hab mir parallel zu dem was wir da gemacht haben, eigentlich wie der Proof of Concept anfing kam das erste Team als wir dann noch parallel Mitarbeiter angestellt haben, die komplett neu bei uns angefangen haben. Also die hatten dann alle mit Maschinen nichts zu tun, also mit Maschinen, die kamen mehr oder weniger aus der Software Entwicklung oder direkt von der FH und das Gute war dann aber auch, da haben Kollegen damals schon bei mir Bachelor und Masterarbeit gemacht, also von der FH aus und die habe ich dann auch bei mir übernommen und die haben dann auch Kenntnisse gehabt. Also insofern haben wir dann unser Know How mehr oder weniger zusammen geschmissen, aber was auch so eine Erkenntnis für mich war, also wie gesagt am Anfang haben wir es erst einmal mit probieren und mit den Basis Kenntnissen die wir hatten versucht. Wir haben dann im Nachhinein festgestellt, dass wir zum Beispiel unseren Rancher den wir aufgesetzt hatten, dass das nicht gut gelöst war so wie wir es gemacht haben und zwar kam das Ganze zum Vorschein, dass wir irgendwann das Thema Security angegangen sind. Das heißt, wir haben uns einen externen Partner gesucht und haben dann gesagt wie sieht denn jetzt unser ganzer IT

..Qualifiziertes Personal [AQ11]

..[AQ09] Schulung Admins

Rechtfertigung der Masterarbeit



Stack jetzt aus, passt das so oder gibt es irgendwelche Security Themen und wir haben von denen eigentlich dann so zwölf Empfehlungen bekommen, die irgendwo noch offen waren. Wir waren schon ganz gut unterwegs, aber so ein paar Punkte wie zum Beispiel, dass wir separate Name Spaces machen oder von mir aus im Docker Container kein root drinnen sein sollte. Wir hatten am Anfang wirklich für jedes Ding das wir gemacht haben eine eigene VM hochgezogen und das war dann so diese Erkenntnis wo dann diese Kollegen die mir dann die Masterarbeit gemacht hatten, die hatte ich dann ja übernommen und alles, und wo man dann gesagt haben komm jetzt habt ihr die Zeit, oder auch schon in der Masterarbeit, setzt euch dahin, schaut wie ihr zum Beispiel die Daten analysieren könnt, wie ihr mit dem Kubernetes, dem Docker und so klarkommt und auch da haben wir dann festgestellt, dass so wie wir es ursprünglich aufgesetzt haben, dass es nicht ganz glücklich war und dass man das viel geschickter machen kann und gerader um so eine Zero Down Time hinzukriegen. Weil bei uns wenn da was läuft, oder wenn Updates sind, wir können es denen nicht unterm Hintern wegziehen, wenn da so eine Maschine läuft und die bleibt stehen das kostet am Tag ungefähr € 70.000,- das heißt, wir waren immer getrieben, dass die Sache eigentlich ständig laufen müsste.

25 I: Können wir hier vielleicht gleich anknüpfen an den Punkt, weil Sie sagen das war so einer der Punkte der indiskutabel war. Was waren denn sonst noch so Rahmenbedingungen wo sie sagen das muss das System erfüllen, um das kommen wir nicht herum. Also wie sie auch gesagt haben die Daten müssen beim Kunden, oder zumindest im gleichen Land liegen, das glaube ich würde da so reinfallen.

26 B: Die Skalierbarkeit, Modularität, Hochverfügbarkeit, das wären so eigentlich die drei, vier Punkte hatten wir wo wir gesagt haben das muss auf jeden Fall sein und womöglich eben der Einsatz von Open Source Software, weil letzten Endes eben auch die Kosten ein Thema natürlich waren. Es konnte am Anfang keiner richtig einschätzen was da auf uns zukommt, weil als Beispiel, wir haben die Datenbank, da habe ich gesagt da möchte ich auf jeden Fall Support haben, und dass wir eine Enterprise Lizenz haben und dann waren wir schon bei der Basis bei so ca. € X ungefähr. Dann hatten wir beim Kafka, das haben wir komplett selbst aufgesetzt als Open Source haben uns dann aber eine Software besorgt, die heißt „Lances“ mit der kann man Windwos für Kafka, also eine Bedieneroberfläche wo man relativ einfach mit SQL und allem möglichen das verwenden kann.

27 I: Ja das haben wir auch im Einsatz gehabt. Das kenne ich, das funktioniert relativ gut.

28 B: Also ich komme ursprünglich aus der JAVA Entwicklung und eben mit Spring Framework und alles uns so Sachen angeschaut, aber wie soll ich sagen, da gab es zum Beispiel ein Cloud Data Flow heißt es glaub ich, wo man dann zum Beispiel wenn man Kafka konnektieren wollte mit zum Beispiel vom Brocker Daten abholen und dann in die Mongo DB schreiben, das ähm jetzt habe ich irgendwie den Faden verloren. Ja genau, mit dieser Oberfläche hätte man sage ich mal

auch visuell darstellen können. Ich weiß nicht ob Sie den Node Read kennen und – sagt Ihnen nichts? Node Read basiert auf Node das Ganze und es gibt einem die Möglichkeit mit simplen Drag and Drop, mit sag ich mal Dinge zu konfigurieren zum Beispiel um von der Steuerung Daten abzuholen. Da gibt es einfach Plugins wo man sagen kann man holt beim [..] oder OPCOA holt man die Daten ab und das ist so eine Drag and Drop Oberfläche und da kann man sich die Sachen zusammen ziehen und sieht nachher den kompletten Data Flow im Prinzip und das hätte ich mir eigentlich immer gewünscht nur ist mit Redhat das Problem, dass es zu langsam ist für höhere Frequenzen und deswegen haben wir das Data Flow angeschaut und das war eigentlich die ideale Lösung und so sind wir eigentlich zu Lances gekommen. Weil man das da eben auch sehen konnte und relativ einfach machen kann. Wobei jetzt die Kollegen im Moment gerade dabei sind, wir haben mittlerweile, ich weiß es nicht ich glaube 140 oder 160 Konnektoren da laufen, aus ihrer Sicht wird es Ihnen da langsam zu unübersichtlich und wir sind vom Team her, also für das IT Stack Team sind wir momentan so sieben, acht Leute, wir holen uns manchmal noch jemanden dazu aus einem anderen Team, aber das Warten, gleichzeitig Anwendungen Entwickeln und all diese Dinge ist doch relativ viel in letzter Zeit. Wie gesagt investieren die Unternehmen im Moment nicht gerade in Personal durch die ganze Krise.

..[AQ04] Betriebspersonal Aufwand

29 I: Das Problem kennen wir auch irgendwoher. Weil sie gesagt haben die Hauptanforderungen sind auf jeden Fall die Themen Hochverfügbarkeit und Broker auch, sie haben auch gesagt natürlich, dass der Speicherdurchsatz ein Thema ist. Gibt es sonst noch irgendwelche spezifischen Eigenarten, funktionale Anforderungen die ihre Cluster erfüllen müssen. Also ich gehe jetzt einmal davon aus, Netzwerk Konnektivität wird wahrscheinlich auch ein Punkt sein, aber haben sie dann irgend etwas, das jetzt spezifisch auf ihre Branche oder atypisch?

30 B: Also spezifisch sind halt, würde ich schon sagen, diese Datenmengen, vielleicht ein ganz kleiner Ausblick was jetzt natürlich auch immer mehr angedacht wird, dass man künftig die Daten mit KI und so analysieren und dann brauch ich eben auch eine entsprechende Daten und so große Datenmenge und so eine Maschine ist eben auch unglaublich komplex. Wir könnten theoretisch 2000 Sensordaten an der Maschine abfragen und manche davon sind im Millisekunden Bereich. Also insofern die Geschwindigkeit ist dann natürlich ein Thema und ich hatte vorher gesagt an der Maschinenlinie haben wir 70 MB pro Sekunde, wobei das aber auch ein Grund war, also das ist ein Grund warum wir es momentan noch nicht bis zu uns in die Cloud schicken. Also der eine Punkt ist natürlich von den Kunden her, dass die sagen, wenn dann maximal bei uns und nicht irgendwo anders, das ist vertraglich auch so geregelt und das andere natürlich, dass einfach auch 70 MB pro Sekunde die kann ich nicht einfach kurz um den Erdball schicken. Das war aber auch ein Punkt zum Beispiel warum wir mit den Brokern, wieso ich das entsprechend auch zum Teil vor den Kafka noch, also damals hatten wir es ehrlich gesagt noch nicht so gemacht. Es sind viele hergegangen und haben es direkt ins Kafka geschoben, was im Prinzip ja auch wie ein Broker ist, aber der Punkt war, ich habe damals geschaut wie viele Daten kann man da auch

..Verfügbarkeit der Daten [AR11]

..[AR07] rechtliche Vorgaben

drüber schicken, ich kann zum Beispiel auf unserem Broker, den wir jetzt nehmen – ist ein Active MQ zum Beispiel, kann ich bis zu 50.000 Topics drauf machen und es läuft eigentlich noch. Also ich habe mich da auch zum Teil einfach eingelesen bei den Herstellern und alles Mögliche was da machbar ist und dann war natürlich auch der Gedanke was ist wenn irgendwo etwas ausfällt, zum Beispiel der Konnektor oder irgendeine Verbindung weg ist und deshalb hab ich den Broker als Puffer mehr oder weniger davor gesetzt und zu sagen, selbst wenn die Verbindung zum Kafka weg wäre, dann sind die Daten da gespeichert bis wir eben, und das ist der Vorteil vom Broker gewesen, über zum Beispiel einen Durable Client, der auf jeden Fall die Daten holt, werden die dann auf jeden Fall vorgehalten, bis er sie geholt hat und das sind eigentlich so mehr die Hauptpunkte gewesen. Wir haben jetzt so, sag ich mal, keine festen Werte gehabt und es konnte keiner sagen wieviel Daten kommen denn da. Also wie gesagt, ich habe das wirklich ausprobieren müssen. Ich habe mit den Jungs aus unserem Automotiv Bereich im Prinzip hergesetzt und habe das aufgesetzt um zu schauen kriegen wir die Daten durch und was kommt denn da überhaupt zusammen. Wenn uns einer gefragt hat wie viel Festplattenplatz brauchen wir denn für die Datenbank, konnte uns auch eigentlich keiner sagen. Insofern mussten wir, das ging sicher über ein paar Monate, die ersten Erfahrungswerte sammeln, wir haben dann auch festgestellt, es ging ja dann das andere Thema war ja was wollen wir analysieren und ich glaube da stehen halt die meisten anderen Unternehmen auch davor, und das ist halt ein Problem, dass viele sagen sie wollen alles Mögliche ausprobieren, die wollen am besten alle Daten haben, aber sie wissen gar nicht richtig was und dass mal ein paar Use Cases definiert werden konnten, die uns eigentlich gesagt hätten, ok ihr müsst das und das umsetzten, oder das und das brauchen wir, also ganz ehrlich an diesen Punkt sind wir erst jetzt in diesem Jahre mehr oder weniger gekommen.

Rechtfertigung der Masterarbeit

31 I: Gibt es bei ihnen grundsätzlich so KPI's oder so etwas wo sie sagen sie können die Qualität von ihrem Cluster oder von den Anwendungen die darauf laufen irgendwie monitoren oder so, dass sie sagen das ist die Haupt KPI die für uns zählt. Also ich habe einmal mitgenommen Verfügbarkeit ganz hoch im Kurs, aber haben sie da auch irgendwelche Performance Faktoren wo sie sagen das ist relevant für uns?

32 B: Nein ich glaube auch einfach aufgrund von der Anwendung die wir jetzt im Moment bei unseren Kunden laufen haben, spielt es noch gar nicht so die große Rolle. Also die Kunden sind halt auch zum Teil sehr zurückhaltend. Es reden zwar alle von Digitalisierung, aber das wirklich auch überall hinzubringen, oder dass die Leute da investieren, also das will nicht jeder oder zum Beispiel ist auch ein Punkt zum Beispiel, wenn wir Produkt Z machen, die Fertigung bei uns die sitzt in Karlsruhe, die wollen gar keine Server bei uns haben, die wollen sie wenn dann bei sich im Haus behalten. Also da ist noch eine unheimliche, wie soll ich sagen, Vorsicht und Skepsis da bei den Firmen, selbst bei der Firma D oder so, selbst die sind ja noch vorsichtig im Prinzip. Wir haben mit denen einen Proof of Concept gemacht zum Beispiel, und es geht jetzt glaub ich schon seit eineinhalb Jahren. Also ich glaub das Thema ist einfach sehr

..Kubernetes beim Kunden

Rechtfertigung der Masterarbeit

Rechtfertigung der Masterarbeit

..Qualifiziertes Personal [AQ11]

komplex für viele und viele wissen nicht wirklich wie sie es angehen können und wenn ich mir mal so den Stellenmarkt anschaue, das Thema Solution Architekt das scheint bei so vielen so hoch im Kurs zu sein, das ist unglaublich was im Bereich Architektur Leute gesucht werden. Und die sind einfach nicht da, geschweige denn die anderen die es dann mit einem aktuellen Wissenstand umsetzten können. Auch bei uns an der FH hatten wir das Thema, dass die eigentlich den Standort bei uns zumachen wollten hier bei uns in Stadt X und zwar hatten die gar keine Professoren mehr, die das den Leuten auf dem aktuellen Stand vermitteln können.

33 I: das hätte ich mir nicht gedacht, dass es soweit ist. Aber man hört es auch bei uns gute Softwareleute sind so rar aber so gesucht natürlich, aber natürlich nicht nur – ich sage es jetzt mal ganz böse – Code Monkeys – sondern auch die Leute die dann auch architektonisch ein bisschen Ahnung haben, das ist schon sehr rar.

34 B: Ich weiß nicht ob es für sie in Ordnung ist, wenn wir vielleicht auch mal die Fragen die sie da auf der Liste hatten, der Reihe nach durchgehen. Ist der Einsatz von Open Source Software möglich – ja also wie gesagt auf jeden Fall wo möglich und wo es auch sinnvoll ist und wir haben es im Moment einfach auch, weil viele Sachen auch natürlich viel Geld kosten, also wenn ich zum Beispiel einen Rancher über das Kubernetes übersetzen will, oder irgendwas anderes dann bin ich gleich einmal bei € 50.000,- oder € 100.00 dabei, also zum Beispiel wir wollten ein Gateway davor machen und wenn ich dann mit Engine X dann zum Beispiel, da gibt es so eine Plus Version, wenn ich da ein paar Lizenzen brauch dann bin ich da bei € 50.000,- bis € 60.000,- dabei, insofern nützen wir Open Source Software wo es nur geht. Von den Lizenzen haben wir jetzt auch eigentlich oft darauf geschaut, es gibt ja mittlerweile diese Apache 2.0 Lizenz, dass man uns an der orientieren. Aber ich hab zum Beispiel von einem großen Automobilunternehmen, da hatten wir das Thema mit dem Lance das was wir vorher angesprochen hatten, da weiß ich vom Herstellen von Lances selber, dass die 120 Seiten haben wollten was alles in diesem Lances drinnen ist. Also sämtliche Libraries, welche Lizenzen und sofort. Also ein richtiger Papierkrieg steht dahinter, also ich glaube drei Monate bis die 120 Seiten zusammengestellt hatten ,so dass Unternehmen D dann zum Beispiel sagt sie nehmen das Lances. Also insofern, das ist noch ein heikles Thema. Wir haben zwar zum Beispiel auch eine Legal Abteilung, die aber selber in diesem ganzen Softwarebereich eigentlich total unerfahren ist, denn wenn, dann gibt man es vielleicht eher noch an externe Spezialisten, aber ja also wie soll ich sagen, ich kann eigentlich nur auf die eigene Erfahrung ein Stück weit zurückgreifen, dass ich sage also die und die Lizenzen müssten passen. Dann war da noch die Verfügbarkeit, da hatten wir auch schon gesagt diese Zero Down Time, das schöne ist halt wenn ich dann bei Mongo DB einen Cluster hab und mehrere Replika Sets, da hatten wir jetzt zum Beispiel ein Update diese Woche, dann kann ich halt eines runterfahren, das andere hoch und die anderen laufen weiter und dann fahr ich das wieder hoch wenn es upgedatet ist und fahr die anderen runter und das ist halt etwas was zum ersten mal die Leute gar nicht gemerkt haben, dass ein Update durchgeführt wurde. Also es war technologisch denen vorher gar nicht bewusst, was es eigentlich bedeutet und warum

..[AR01] Open-Source Lizenzen

..[AQ03] Preis/Lizenzkosten

..Technologiestack

..[AQ02] Verfügbarkeit in %

..[AR02] Availability/Lokation

..Vendor Lock-In [AR16]

..[AR10] Verwendung der publi

..HPA/Skalierung

..[AR03] Enterprise Support

..Usability des Tools [AQ19]

..[AR10] Verwendung der public cloud

..Kubernetes beim Kunden

..[AF04] GPU Support

wir das alles machen mit diesen Replika Sets. Mit dem Cluster in einem bestimmten Land, ja das habe ich ja vorher gesagt, dass ich eben die Daten entsprechend abspeichern muss und momentan machen wir es noch in Europa, da sind wir aber gerade dabei dass wir es für Amerika eben auch machen wollen. Vielleicht in dem Zusammenhang wir haben jetzt auch eine kleine Kehrtwende gemacht, dass wir nicht nur Private Cloud machen, sondern auch zu Azure gehen zum Beispiel und haben was die Mongo DB betrifft, die haben auch ein eigenes Atlas wo sie dann überall verteilen kann egal ob bei Azure, AWS oder Google. Das war auch so ein Punkt dass wir gesagt haben wir wollen nicht unbedingt von Microsoft allein abhängig sein, weil ich hatte ja vorher gesagt diese Modularität sollte ja auch, oder die Skalierbarkeit, die Möglichkeit geben, dass wir auch wo anders hingehen können, ohne dass wir jetzt was Neues einsetzten müssen. Das ist ja auch so ein Ding, zum Beispiel Kafka ist bei allen großen Providern auch im Einsatz bzw. mit unter der Haube, aber es wird immer ein anderer Name darüber gestellt. Bei Google ist es glaube ich Data Flow, bei Azure ist es im [..] mit drin und solche Sachen und auch da haben wir uns zum Beispiel gesagt Kafka, die wird auch mittlerweile von Confluence direkt angeboten, dass man es bei denen laufen lässt und sich dann praktisch aussuchen kann wo man es jetzt braucht. Das ist noch so ein Punkt wo auch die Skalierbarkeit oder ich sage mal das nicht verlieren der Flexibilität unheimlich wichtig ist. Weil einfach der Markt zu schnelllebig ist. Was hatten wir noch – Enterprise Support. Also wie gesagt für Kubernetes haben wir es eigentlich nicht gebraucht. Für unser Team von der Größe her ist es gerade angenehm, wenn wir zum Beispiel zu Azure gehen, dass sie sich jetzt nicht mehr alles um jedes Detail im Hintergrund kümmern müssen und jeden Befehl unbedingt auf der Kommandozeile ausführen müssen, aber es hat halt unheimlich geholfen die ersten Erfahrungen, weil wenn was ist kann man halt trotzdem mal geschwind Hand anlegen und weiß wenigstens was man tut. Nutzung von Public Cloud Diensten erlaubt oder sogar erwünscht – wie gesagt das ist Kundenabhängig bei uns, der eine möchte es, der andere möchte es nicht, was natürlich auch bei großen Kunden kommen wird, wie gerade Automobilherstellern, die haben schon irgendwo eine Cloud Anbindung. Das heißt. wir müssen dann eben auch auf denen ihre API's zugreifen, oder ihnen entsprechend in ihrer Anwendung Daten liefern. Das ist was was jetzt glaube ich auch in der nächsten Zeit mehr kommen wird, also bei den ersten zumindest einmal. GPU – Unterstützung ja – sag ich jetzt einmal von mir her noch, es ist glaub ich im Unternehmen noch nicht so wirklich bewusst was man eigentlich da noch machen kann. Wir haben wie gesagt unsere KI Lösung nur im [..] Floor laufen beim Kunden, aber ich beschäftige mich eigentlich sehr viel damit und habe mir schon GPU Datenbanken angeschaut und für die KI Themen wird es sicher ein Thema werden, aber KI ist für viele auch noch auch noch, sag ich mal, so ein Schlagwort und viele können es noch nicht richtig greifen, was sie damit machen können. Also es gibt da ein paar Lieferanten, wir haben zum Beispiel wie gesagt bei dieser Werkzeug Geschichte was ich vorher gesagt habe, da haben wir einen Kamera Lieferanten der hat ein riesen Team an Entwicklern, der hat dann natürlich gesagt, Mensch wir können das auch für euch machen und dann hatten wir eben auch geschaut, OK können wir das Auslagern oder solche Sachen also durch diesen

..[AF02] data-at-rest Encryption
..Mandantenfähigkeit
..[AF05] Betriebssystem für Contain
..Kubernetes beim Kunden
..[AF07] Pod2Pod Kommunikation
..[AF12] Layer 4 / Load Balancer

Ressourcen Mangel an Personal guckt man natürlich auch schon, ob man nicht irgendwelche Themen auslagern kann, das ist auch etwas was sich in den letzten Monaten unheimlich bemerkbar macht eigentlich. Müssen gespeicherte Daten verschlüsselt werden – das ist bei den Daten die wir jetzt haben kein so ein riesen Punkt. Was wir gemacht haben bei dem Security Check damals mit diesem externen Anbieter ist, dass wir gesagt haben OK die Kunden Daten werden grundsätzlich natürlich in separaten Datenbanken und alles getrennt abgespeichert. Das war so eigentlich die Hauptherausforderung, weil wir eigentlich eine mandantenfähige Lösung, und dass natürlich keiner auf den anderen seine Daten zugreifen kann. Ansonsten wäre so etwas vielleicht irgendwo, bei Kreditkarten Daten könnte man es vielleicht verschlüsseln oder so was aber, wir speichern jetzt auch momentan nichts irgendwas was personenbezogen ist da drinnen, das sind eigentlich alles Maschinen, also ist das nicht so tragisch bei uns. Betriebssystem - hatte ich glaube ich vorher schon gesagt, LINUX setzen wir hier bei uns eine. Ich merke, dass hier der Trend auch immer mehr zu LINUX hingeht, selbst Microsoft hat sich ja mittlerweile unheimlich gegenüber LINUX geöffnet. Ich habe früher wie gesagt ja auch jahrelang Hosting für Kunden und so gemacht, und da waren auch einfach die LINUX Server von der Stabilität her wesentlich besser, die konnte man manchmal monatelang laufen lassen, ohne dass man irgendwo eingreifen musste, was bei Windows halt damals nicht immer so glücklich war und so haben wir eigentlich von vornherein immer LUNUX bei uns eingesetzt gehabt. Kommunikation zwischen POD's regulierbar – das war mir jetzt nicht ganz klar inwiefern sie das gemeint haben?

35 I: Es gibt die Möglichkeit, dass man über verschiedene Netzwerke, also CNI Plugins, Network Policies auf Kubernetes fährt, dass man zum Beispiel sagt, dieser Pod darf nicht mit diesem sprechen, so ingress und egress Regeln auf POD Level. Ist jetzt die Frage, brauchen sie so etwas?

36 B: Da hat sich ein Kollege daran versucht und der ist schier daran verzweifelt, aber das ganze kam eigentlich ursprünglich aus dem Edge Bereich, weil wir haben natürlich, wir sind gerade dabei, dass wir diese Technologien immer mehr zum Kunden im Production Floor transportieren, das heißt. früher hat es um die Maschine herum da hat es keine Cluster von Kubernetes, oder Docker Container oder irgend so etwas gegeben, und das kommt halt auch immer mehr. Insofern hat sich da ein Kollege einmal damit beschäftigt, wie können wir da was absichern und alles und der ist an machen Sachen hier, sag ich mal, wirklich aus dem Internet gezogen hat verzweifelt, weil die auch Kuberentes im Hintergrund einfach wieder geändert hat und insofern ist da momentan nicht wirklich viel passiert, aber ich glaube bei uns ist da auch schon relativ viel Sicherheit schon drinnen, insofern ist da gerade im Moment nicht so das große Thema. Vielleicht wird es noch einmal eines, wenn mir zu Azure gehen, weiß ich nicht, aber kann ich heute noch nicht abschätzen. Protokolle – haben wir auch schon darüber gesprochen. Bei uns ist eigentlich überwiegend MQTT, HTTPS. Vielleicht dann, wenn man es so Art managen oder so von manchen Sachen, dann halt mal SSH noch aber ansonsten da eigentlich nichts. Qualitätseigenschaften – hatte ich gesagt

einerseits diese Zero Down Time was ich mir aber noch dazugeschrieben habe, was vielleicht auch noch kommen könnte, wären erstmal Serverless Function Huts, da habe ich dann halt eigentlich in dem Sinn ja das dann in dem Sinn, dass das da halt nicht dauernd läuft, weil das halt auch sonst entsprechend Kosten produziert. Das heißt ich rufe meine Functions eigentlich nur auf, wenn ich sie wirklich brauche, das wäre eigentlich das wo es ein bisschen halt in die andere Richtung geht aber da halten wir uns gerade auch noch sehr zurück. Installation – Upgrade – Aufwand, wenn man es vernünftig automatisiert, würde ich mal sagen, ist der Aufwand gering, aber muss halt gut geplant sein. Momentan ist es etwas das war bisher immer so im Argen bei uns, das ist etwas was ich gerade auch versuche zu ändern, dass wir die ganzen CI CD Themen noch einmal massiv angehen, weil da laufen ein paar Dinge noch nicht so ideal, also wie gesagt, wir sehen es einfach auch an anderen Sachen wie zum Beispiel an der Mongo DB wie ich vorher gesagt habe, da sieht man wie super es ist, wenn das Ding nicht runtergefahren werden muss, wenn das einfach in laufenden Betrieb gemacht werden kann, aber das war bisher bei uns einfach momentan bei uns aufgrund von Kapazitäten und zum Teil auch, ich hatte ja gesagt, die Kollegen die sind zum Teil frisch von der FH oder sonst wo gekommen, die hatten diese Erfahrung mit großer Software Entwicklung hatten die so noch gar nicht gemacht. Also ich war zum Beispiel bei meinem früheren Arbeitgeber, waren wir ein Entwickler Team von 50 Leuten, wir haben glaube ich in Deutschland 18.000 Kunden gehabt und wir haben wirklich fast für die ganze Automobil Industrie die Daten bei uns gespeichert. Da war das ein ganz anderes Denken, das ist jetzt so in einem relativ kleinen Team wo jetzt die Leute Schritt für Schritt dazukommen und integriert werden müssen, einfach noch nicht so der Mittelpunkt. Aber das ist auch etwas was ich wirklich auch viel bei Mitbewerbern oder Geschäftspartnern oder so sehe, dass es denen auch überall ja ähnlich geht. Kosten - oder ja Nutzung für Updates – sie hatten bloß Kosten, ich kann es jetzt wie gesagt nur für zwei Sachen sagen, ansonsten haben wir die Kosten in dem Sinn noch nicht so richtig festgemacht. Wie gesagt die Mongo DB wo wir Enterprise Support haben und Consulting nutzen und für das Lances, alles andere, es ist kein unerheblicher Betrag würde ich mal sagen, also das ist auch ein Grund warum wir uns mit Azure gerade massiv beschäftigen, weil wir einfach sagen wir können das aufgrund der ganzen Leute so nicht abdecken. Also wenn ich jetzt mal überlege wir haben momentan was Personal betrifft, in diesem Team für unsere Architecture haben wir jetzt fünf Leute grob die wo fest drinnen sind, aber dazu noch zwei, drei dazu und da können sie fast davon ausgehen, dass die fast ihre ganze Zeit damit verbringen.

..[AQ09] Schulung Admins

..Qualifiziertes Personal [AQ11]

..[AQ04] Betriebspersonal Aufwand

..[AQ03] Preis/Lizenzkosten

37 I: Also ist das dann definitiv schon auch ein großer Punkt?

38 B: Ja auf jeden Fall, also das ist jetzt in der Vergangenheit in Euro nie so richtig festgehalten worden, aber uns ist das schon bewusst, dass das schon ein größerer Brocken ist. Also wir hatten neulich mal über Werte gesprochen von so einer viertel Million im Jahr also auf jeden Fall das kann gut sein.

39 I: Dann abschließend noch, legen sie Wert auf den Marktanteil oder die Community Größe bei Produkten die sie sich anschauen, oder?

..[AQ06] Communitygröße und -qualität

40 B: Also bei Größe der Community habe ich eigentlich extra, das war auch ein Punkt, stimmt, wo wir damals diese Regeln festgelegt hatten, weil wir gesagt haben Community ist uns sehr wichtig, weil wie gesagt, die Sachen entwickeln sich ja so unheimlich schnell, ich sage mal früher konnte man hergehen und konnte ein Buch durchlesen und hatte dann vielleicht zwei, drei Jahre davon profitiert, das geht heute nicht mehr. Bei den Dingen wo es dringend ist, wenn eine Datenbank weg wäre, da sag ich halt dann OK externer Support. Aber bei allen anderen Sachen war es mir schon wichtig, dass wir Community mäßig, dass es da was Gutes gibt. Ich weiß nicht ob ihnen, wie heißt sie, ich glaub Cloud Native Foundation?

41 I: ja die CNCF

..[AQ06] Communitygröße und -qualität

42 B: CNCF genau, da gibt es halt zum Beispiel eine Landscape wo halt welche Produkte wo eingesetzt werden und das ist halt auch was wo mir uns auch halt daran orientiert haben. Sieht man ja sind es kleine Unternehmen sind es große Unternehmen, das sieht man da ja auch, mitarbeitermäßig, umsatzmäßig, was hat sich wo etabliert, aber wie gesagt ich glaube die Community ist heute wichtiger als es noch vor ein paar Jahren, vor allem in diesem schnell wachsenden Umfeld grad.

43 I: ja da stimme ich absolut überein. OK ich denk also mit dem hätten wir eigentlich Qualitätseigenschaften abgeschlossen. Also von meiner Seite aus, super Input, vielen vielen Dank, echt lässig, vor allem den Einblick einmal aus einer anderen Sparte, einmal ganz was Neues. Echt cool.

44 B: [..] um den heißen Brei irgendwo herumgeredet, aber wenn was ist, sie können sich ja gerne noch einmal melden oder so, überhaupt kein Thema.

45 I: Ja also wie gesagt, auf jeden Fall viel, aber sehr sinnvoll und Gott sei Dank auch schon in die Richtung wo ich sage da geht meine Arbeit und mein Theorie Teil hin, also das deckt sich Gott sei Dank schon recht gut, aber auf jeden Fall ein paar neue Einblicke. Also den ganzen Industrie Sektor hätte ich so jetzt nicht am Plan gehabt, aber echt wertvoll also vielen Dank, dass sie sich hier die Zeit nehmen und vor allem auch um diese Zeit. Ich würde ihnen dann heute oder morgen noch die Einverständnis Erklärung schicken, dass wir das dann eventuell zurückschreiben, wenn sie das brauchen. Wie gesagt, das Interview wird anonymisiert, ich muss es abtippen und es wird transkripiert, das fließt dann so in die Arbeit ein. Ich werde eine Auswertung von dem Ganzen machen. Die Interviews an sich dann alle zusammenfassen und wenn es sie interessiert, die Arbeit ist ja öffentlich, ich kann ihnen das dann auch gern nochmal schicken, also als PDF soweit ich weiß. Also falls sie das interessiert, gerne.

46 B: Die Anonymisierung wäre mir auch wichtig, weil wie gesagt es ist halt in dem Bereich sind sehr viele unterwegs und das andere ist natürlich – ich weiß nicht wer es bei ihnen zu sehen bekommt, aber wie ich ja schon gesagt habe, der Herr A ist bei uns im Aufsichtsrat,

nicht dass dann direkt Unternehmen X steht und er sieht das irgendwo, könnte ich mir vorstellen, dass er dann neugierig wird.

47 I: nein also ich schreibe nicht in Kooperation mit der Firma N, muss ich auch dazusagen, es ist nur meine akademische Betreuerin, die sieht das und mein Studiengangsleiter, also die Korrektur ja, aber ich werde sie einfach als Maschinenbau Unternehmen führen und ich denke das ist dann genug anonymisiert.

48 B: Es war viel auch neben der Spur, aber ich hoffe es hat ihnen trotzdem ein Stück weit geholfen.

49 I: ja auf jeden Fall, also das hat mir sehr weitergeholfen.

A.25 Interview: Protokoll Unternehmen D

1 I: Ja gut also, vielen Dank einmal. Mein Name ist Daniel Drack, der Herr K. hat mich an Sie verwiesen und hat mir Ihre Kontaktdaten gegeben. Super, dass das so schnell hingehaut hat, das ist echt wichtig für mich. Wie gesagt, ich studiere am Campus02 Innovationsmanagement... ich glaube ich schalte mal mein Video aus, das packt meine Verbindung nicht ganz. Hoffentlich gehts besser. Ja, wie gesagt ich studiere am Campus02 Innovationsmanagement, mache da gerade meinen Master und schreibe über das Thema Design... also innovationsbasiertes Design von Kubernetes Clustern meine Masterarbeit. Ich arbeite nebenbei beim Unternehmen X vollzeit im DevOps Team, also ich mach das auch beruflich. Deswegen auch die Idee über das Thema zu schreiben. Grundsätzlich bin ich bei meiner Arbeit mittlerweile so weit, dass ich sagen kann, ich habe eine Art Prozess entwickelt wie man sowas angehen könnte. Das Herzstück dessen ist eben ein Katalog an Anforderungen, basierend auf dem man dann Sachen wie die Kubernetes Distribution, die verschiedenen Produkte, CNI Plugins etc. auswählt. Dieser Anforderugnskatalog, das habe ich eh in der Einladung schon mitegschrieben, teilt sich laut Literatur in drei Kategorien. In sogenannte Rahmenbedingungen, funktionale Anforderungen und Qualitätseigenschaften. Und die ganzen Anforderungen werden in die drei Kategorien eingeteilt. Das ist so das Ergebnis von meiner theoretischen Forschung.. unter Anführungszeichen. Ich habe schon in meinem Unternehmen zwei Workshops gemacht zu dem Thema, um das einfach noch ein bisschen interdisziplinär abzuklopfen, also verschiedene... nicht nur aus Softwareperspektive sonder wirklich auch aus anderen Unternehmensbereiche wie Management, Licensecompliance, IT-Security etc. Dass man da ein bisschen ein holistisches Bild bekommt zu dem Thema. Und jetzt mach ich eben noch Interviews um verschiedene Branchen abzuklopfen, damit ich nicht nur aus der puren Softwareentwicklung oder Automotive Ecke bei uns habe, sondern wirklich auch andere Branchen wie Finance, wie Gesundheitswesen, wie Infrastrukturprovider etc. Ja, deswegen haben wir dieses Interview und bin ich auf Sie gekommen. Ja, ich würde das jetzt so gestalten. Also meine Vorstellung haben wir, wir wissen ungefähr um was es geht. Ich würde Sie dann bitten sich vorzustellen, kurz zu sagen wer Sie sind, was Sie so machen und dann einfach mal erklären wie Ihre Umgebung ausschaut. Dann würde ich gerne die drei Kategorien von Fragen durchgehen, das ein bisschen diskutieren was sie da für Erfahrungen gemacht haben. Ja, dann wärs dann im Prinzip auch schon. Derweil noch eine Kleinigkeit, ich werde Sie anonymisieren und auch das Unternehmen, das mache ich bei allen Interviews so. Und das Unternehmen einfach als Infrastrukturprovider oder dergleichen einfach bezeichnen. Und Sie auch namentlich heraushalten.

2 B: Ja, okay. Mein Name ist Y, bin beim Unternehmen X IT-Systems Engineer. Bin dort hauptsächlich zuständig für die Produkte aus dem RedHat Portfolio, also RHEL, mit dem Satellite als Management. Aber mein Hauptfokus aktuell liegt eben beim Design und beim Betrieb von Container Orchestrierungsplatformen. Und alles andere was sich so um das Thema containierisierung und DevOps handelt. Bevor ich voriges Jahr im Mai 2019 beim Unternehmen X

..Stellenbeschreibung

..Technologiestack

..Stellenbeschreibung

angefangen habe war ich neun Jahre beim Unternehmen Y im Informatikcenter in Graz tätig. Habe dort eben das gleiche gemacht, dort aber angefangen mit den klassischen Linux LAMP Stack. Also Linux, Apache, PHP und MySQL und so Sachen. Und seit 2017 mache ich halt hauptsächlich Systeme im Kontext mit Container und Container Orchestrierung. Dort mit dem Produkt OpenShift. Bei uns im Unternehmen X habe ich heuer im Jänner bis März ein Projekt umgesetzt, da ist es darum gegangen Docker basierte Container Hosts und containerisierte Softwarelösungen die in-house entwickelt worden sind auf eine Container Orchestrierungsplattform zu migrieren. Das heißt wir haben diese bestehende Applikation die auf Linux Server mit Docker betrieben wurde, und davor eben mit einem LoadBalancer gebalanced wurde, auf die Container Plattform von RedHat, also OpenShift, migriert. Und aktuell bin ich dabei für diverse andere Kunden aus dem.. aus diverseset Sektoren und Bereiche eben auch Container Plattform Lösungen zu designen und in Betrieb zu nehmen.

3 I: Darf ich vielleicht gleich einhaken bei dem Punkt? Ist das Thema OpenShift bzw. RedHat von vorne herein schon gesetzt gewesen wo Sie ins Unternehmen gekommen sind? Oder war das eine bewusste Entscheidung, also man will OpenShift? Und wenn, warum?

..Kubernetes Architektur

..[AR09] Strategesche Partner

..Qualifiziertes Personal [AQ11]

4 B: Also.. nachdem ich ja aus dem Finanzbereich komme und jetzt beim Unternehmen X sehr viel mit diesem Umfeld zu tun habe. Also Banken, Energieversorger usw. also große Unternehmen, also bewegen wir uns im Enterprise Umfeld. Und für das Enterprise Umfeld ist die Anzahl an Container Orchestrierungsplattformen sehr überschaubar. Und zusätzlich kommt noch dazu, dass wir als Unternehmen X RedHat Certified Cloud Service Provider sind und mit der RedHat eine Partnerschaft haben und in diesem Bereich eben sehr eng zusammenarbeiten. Und ich persönlich bin auch RedHat Certified System Administrator Engineer, Certified Professional in Container for Kubernetes und Specialist in OpenShift Administration. Ich habe also ein sehr abgerundetes Zertifizierungsportfolio im Bereich OpenShift.

5 I: Das klingt so ja. Aber grundsätzlich war die Entscheidung für OpenShift schon da bevor Sie ins Unternehmen gekommen sind, oder? Habe ich das richtig verstanden?

..[AR09] Strategesche Partner

6 B: Das kann man so sagen ja. Eben auf Grund der Partnerschaft mit RedHat.

7 I: OK. Können oder dürfen Sie mir kurz sagen was Sie ungefähr für eine workload betreiben? Also um was es sich bei den Applikationen handelt? Und sonst halt nur auf technischer Ebene.

..Art der Workload

8 B: Also.. hauptsächlich geht es dabei um Eigenentwicklungen in .net mit .net-core Containern.

9 I: Perfekt.

..Art der Workload

..[AR09] Strategesche Partner

10 B: Also das ist interessanterweise ein sehr großer Anteil bei uns an Technologien aus der Microsoft Ecke. Auch dem geschuldet, dass wir eben auch ein Microsoft Partner sind. Die Softwareentwicklung

..Art der Workload
..[AR09] Strategesche Partner

ist eben hauptsächlich C# basierend was ich so mitbekommen habe, daher diese Workload sehr .net lastig.

11 I: Habe ich das richtig verstanden, dass Sie jetzt so wohl ihre in-house Lösungen auf Kubernetes betreiben, als auch Kubernetes für Kunden betreiben oder hosten.

..Deployment zum Kunden

12 B: Genau, bei uns gibt es eine Abteilung die nennt sich managed service. Das heißt wir implementieren für den Kunden Lösungen. Also im Bereich Kubernetes und Orchestrierungsplattformen und OpenShift, die wir beim Kunden on-premises implementiern, oder eben bei uns im DataCenter. Und eben für diesen dann auch die Betriebsführung anbieten und dann auch durchführen.

13 I: Dann ist natürlich OpenShift nochmal naheliegender. Gut grundsätzlich, ich meine der Vollständigkeit halber... kann man ungefähr sagen von wem die Entscheidung gekommen ist, dass man jetzt wirklich auf A Kubernetes trifft: "wir machen jetzt Container Orchestrierung", und B: "Wir machen das ganze mit OpenShift". Also war das eher so eine bottom-up Geschichte, oder sagt man das ist schon vom Management strategische in diese Richtung gedrückt oder entschieden worden?

..Entscheidung für Kubernetes
..Dauer Kubernetes Verwendun
..[AR03] Enterprise Support
..Support-Responsezeit auf Feh

14 B: Ja also da kann man schon sagen, dass ist sehr.. also von beiden Seiten eigentlich. Aber die Entscheidung war relativ einfach zu finden. Nachdem ja diese Enterprise Lösungen, wie vorhin schon gesagt, sehr überschaubar sind am Markt und es auch vor zwei drei Jahren noch sehr überschaubar war. Und eben auf Grund des Herstellersupports mit den SLAs für Supportcases. Also... da haben wir dann eben uns dafür entscheiden müssen, dass wir eine kommerzielle.. also sagen wir so.. einen Hersteller einer Container Orchestrierungsplattform finden die uns dann in diesem Bereich dann auch diesen sieben mal 24 Support dann bieten.

15 I: Vielleicht.. kann man ungefähr was sagen, wer die Entscheidung wirklich getroffen hat.. also wer die Entscheidung getroffen hat, dass es dann OpenShift wird? Oder wer daran beteiligt war? Was das etwas wo man sagt das hat man nur auf Management Ebene entschieden, und es war wirklich der Support ausschlaggebend? Oder hat man da auch technisch mit Entwickler, Administratoren, Architekten das abgewägt was es da für Alternativen gäbe? Also wer war da beiteiligt von Teams, oder von Stellen im Unternehmen?

..Motivation für Kubernetes
..Entscheidung für Kubernetes
..Technologiestack
..Vorschlag für Kubernetes

16 B: Also bei uns im Unternehmen X heraus getrieben wurde das Thema eben aus der Technik heraus, eben dass das in Technik erkannt worden ist, dass der Bedarf an Container Orchestrierungsplattformen am Markt sehr steigt. Und wir haben eben aus der Technik heraus diesen Vorschlag schon unterbreitet, eben durch das das wir ein tiefes Wissen im Bereich RedHat haben und diese Partnerschaft eben strategisch verfolgen, dass wir eben aus der Technik heraus schon die Empfehlung abgegeben haben, das soll OpenShift werden.

17 I: OK, gut dann hätte ich mal das erste Kapitel vonwegen Vorgehensweise, Designprozess, Entscheidungsprozess etc. mal als

solches abgeschlossen, von meiner Seite aus. Der nächste Punkte wären diese Rahmenbedingungen, die ich in der Einladung schon mal drinnen gehabt habe. Was waren denn von Ihrer Seite aus, von Seiten des Unternehmen X aus so definitive Grenzen wo Sie gesagt haben: "OK, das können wir nicht überschreiten, wenn wir uns für eine Plattform oder ein Produkt entscheiden müssen"? Also ich habe schon mitgenommen diese strategische Richtung, wo Sie gesagt haben Sie sind certified RedHat Partner. Das ist sicher mal ein großer Faktor, dass man sagt OpenShift hab schon mal einen gewissen Vorteil. Aber hat es sonst noch Rahmenbedingungen gegeben wo Sie gesagt haben, das sind sachen die können wir.. die sind nicht verhandelbar. Umstände die müssen von einer Distribution oder einem Produkt erfüllt werden, dass das in die nähere Auswahl kommt?

18 B: Also unsere Anforderungen waren jetzt an und für sich in diesem Bereich sehr überschaubar. Man hat eben.. also wir haben eben dazu die Entscheidung gehabt, dass wir das on-premises betreiben müssen. Zum damaligen Zeitpunkt war eben das Thema public-cloud eben außen vor. Also das war nicht das Thema, dass man jetzt sagt man geht in ein Azure oder ein AWS oder GPC oder was auch immer. Also damals war eben die Entscheidung schon gefällt, dass das on-premises bei uns im eigenen Data Center betrieben wird. Das heißt also, dass diese.. diesen Mandanten den wir bei uns im.. in dieser private cloud für diesen.. dieses Produkt haben. Das war eben dafür schon gesetzt. Also da gibt es mehrere Mandanten in dieser Infrastructure as a Service Lösung, und damit war eben diese Rahmenbedingung schon gegeben. Dass die Container Orchestrierungsplattform on-premises, in diesem Mandanten zu integrieren sein wird.

..Kubernetes Architektur

..[AR10] Verwendung der public cl

..Mandantenfähigkeit

19 I: Gibt es vielleicht von Ihren Kunden oder von Ihnen selber in ihrem Business Segment irgendwelche rechtlichen Vorschriften die Sie auf technischer Ebene dahingehend treffen. Wo Sie sagen, wenn ich ein neues Produkt... Hausnummer wir evaluieren einen neuen Ingress Controller, der muss bestimmte Rahmenbedingungen erfüllen, sonst fällt er einfach schon mal durchs Raster.

20 B: Ja also wir haben aktuell schon bei einem Kunden aus dem Bankenumfeld dieses Thema PCI DSS. Wobei des... dieses PCI DSS Thema das betrifft sowieso jede Bank in gewisser Art und Weise. Und da haben wir eben den Vorteil schon, dass wir mit RedHat OpenShift Container Lösung den PCI DSS Standard out-of-the-box erfüllen können. Und wenn wir sagen man müsste jetzt andere Distributionen evaluieren... da müsste man dann schon eben schauen, sind diese auch für diese rechtlichen Anforderungen geeignet. Diese Anforderungen haben wir bei uns im konkreten Fall für diese Lösung nicht gehabt. Aber wenn man das Thema Compliance.. und ... legal betrachtet, da muss man dann schon eben solche Aspekte dann mitbetrachtet. Da kommt man dann auch wieder auf Themen zurück wenn man in die Cloud geht. Also wechel Lokation verwendet man, also da kann es schon sein, dass die USA... schon mal rausfällt.

..[AR07] rechtliche Vorgaben

..[AR02] Availability/Lokation

21 I: OK..

..[AR01] Open-Source Lizenzen

..Lizenzcheck im Container

22 B: Weiteres ist dann noch zu betrachten das Lizenzthema. Also.. das eine Thema ist, wie betreibt man die Container Orchestrierungsplattform, wie wird die subskribiert, im Fall von RedHat, oder lizensiert? Was dann auch noch zu betrachten ist, ist wie ist die Lizenzthematik der containerisierten Anwendungen? Also, dass man.. das.. das muss mann dann auch noch zusätzlich mitbetrachten. Welche.. Lizenzen sind dann dafür notwendig?

23 I: Vielleicht eine Frage rein aus Interesse. Liefert OpenShift eine Möglichkeit, dass ich die Software im Container überprüfen kann? So wie WhiteSource oder so das machen würde?

24 B: Nein.

25 I: Nicht?

..Antivirus

26 B: Es gibt weder.. Es gibt jetzt keine vulnarability Scanner out-of-the-box dabei, und auch keine.. License Scanning Tools. Also alles was den Bereich Scanning angeht ist jetzt im Bereich OpenShift nicht integriert. Also es gibt einen vulnarability Check auf Container Image Eben der eben das Container Image gegen bekannte vulnarabilieties und CVEs prüft. Das ist aber ein reiner Check.. auf Metadatenbasis. Also da werden jetzt keine aktiven Prüfungen in den Container Images durchgeführt. Aber solche Prüfungen sind jetzt in OpenShift nicht out-of-the-box integriert, nein.

27 I: Gut, ich denke dann hätten wir das Thema Rahmenbedingungen auch schon. Der nächste Punkt schlägt in die Kerbe wo es um das Thema funktionale Anforderungen geht. Sie haben gesagt, dass bei Ihnen in dem Fall.. Sie haben hauptsächlich .net core Anwendungen, was jetzt wahrscheinlich nicht irgendwelche exotischen Anforderungen hat. Gibt es irgendwas, wo Sie sagen Ihre Kunden oder Sie selbst im Unternehmen X haben irgendwie funktionale Anforderungen, die vielleicht über das klassische Kubernetes Portfolio hinaus gehen. Zum Beispiel, dass Sie Support für GPU brauchen, oder hybride Cluster mit Windows. Irgendwas in diese Richtung?

..[AF12] Layer 4 / Load Balancer

..[AF04] GPU Support

..[AF05] Betriebssystem für Container

28 B: Naja also aktuell haben wir diese Anforderungen noch nicht. Einige Pitfalls hatten wir Letztens bei der Integration von diversen IoT Geräten. Layer vier basiertes Routing ist ja im.. Layer sieben Routing ist ja im OpenShift integriert, aber da muss man dann schauen wie bindet man Lösungen an die zum Beispiel UDP verwenden. MQTT Protokolle oder reine TCP Connections, diese Anforderungen sind schon auch zu prüfen. Welche Anforderungen bestehen auch von der Client Seite. Also es werden unterstützt WebSockets und Layer vier (sieben) basierte Routings mit SNI und TLS, das ist sehr überschaubar, macht aber meines Erachtens mehr als 90% des traffics aus. Sollte man jetzt irgendwie nach Pareto noch diese 20% integrieren müssen, macht das eben diesen 80%igen Aufwand aus, für solche Speziallösungen dann irgendwelche Integrationen zu bauen. GPU haben wir noch nicht die Anforderung gehabt. Glaube wird sich bei uns auch in Grenzen halten momentan noch. Was wir aber schon an Anforderungen haben, das kommt aus der Ecke von Microsoft. Durch sehr starke Integrationen in ein

..[AF05] Betriebssystem für Container

Microsoft Umfeld, in ein sehr heterogene Microsoft Landschaft. So Stichwortartig.. das Thema Service Accounts.. da muss man da irgendwie diese Container über group managed Service Accounts dann mit einer Microsoft Infrastruktur verheiraten. Mittlerweile gibt es auch im Bereich OpenShift schon einiges an Lösungen, auch Microsoft Server.. also Container Hosts zu betreiben. Aber das sind auch sehr spezielle Anforderungen die es unter Umständen auch geben kann. Dass man sagt, Container müssen gut in die Microsoft Infrastruktur integriert werden.

29 I: OK. Jetzt wieder rein aus Interesse, wir hatten nämlich das gleiche Problem, auch mit Layer vier Geschichten. MQTT und Kafka in unserem Fall. Darf ich fragen wie Sie das gelöst haben? Also wir haben, als Beispiel... wir haben uns für MetalLB einfach als LoadBalancer entschieden. Also einfach MetalLB als LoadBalancing Objekt im Cluster, und das funktioniert eigentlich relativ gut. Wobei man sagen muss, wir haben jetzt nicht so brutales Durchsatzaufkommen.

..[AF12] Layer 4 / Load Balancer

30 B: Wir haben das damals mit NodePorts gelöst. Das heißt, dass man eben wirklich dann auf Layer vier Basis dann auf die NodePorts durchroutet. Also wenn man in der.. in der RedHat OpenShift Version 4.6 schaut, ist das Feature dazu gekommen, dass NodePorts auch über eine gewisse damalige Grenze hinaus freigeschalten werden. Das ist dem Unternehmen X geschuldet, dass dieses Feature integriert worden ist in der neuen OpenShift Version. Und wir haben auch das Thema gehabt... Es ist eh nicht zu verheimlichen, dass eben diese IoT Geräte ja sehr lange im Feld draußen sind. Nachdem ja heuer im Frühjahr TLS 1.0 und 1.1 abgedreht worden sind.. und eben die Mozilla Foundation eben dann nach Ende des Corona Lockdown eins diese... das auch durchgesetzt hat, haben dann einige Geräte draußen das Problem gehabt mit TLS und Cipher Suites. Wir haben dann eben für diese Anwendung dann so eine Art legacy Proxy auf unserem externen Load Balancer integriert. Der eben noch für diese alten TLS Version Unterstützung liefert. In OpenShift ist denn eben dieser TLS Support mitte des Jahres auch heraus gefallen. Da werden nur mehr die aktuellste TLS Versionen 1.2 und 1.3 supportet.

..alte TLS Versionen

31 I: OK. Das ist guter Input. OK gut, gibt es sonst noch funktionale Anforderungen wo Sie sagen die würden in diese Kerbe schlagen? Oder haben wir damit alle?

Rechtfertigung der Masterarbeit

32 B: Ja es gibt immer wieder sehr spezielle Fälle, eben auch von Kundenseite. Aber das.. da kommt man dann in der Implementierung drauf, darum ist auch für mich das immer ganz wichtig das im vorhinein abzuchecken. Wie schon erwähnt, irgendwelche Sonderlösungen dann im Nachgang zu implementieren das verursacht dann ein vielfaches an höheren Aufwand, im Vergleich zur initialen Installation.

33 I: Genau deswegen tue ich mir den Aufwand mit dieser Arbeit auch an. Die Erfahrung haben wir nämlich auch gemacht. Gerade aus Interesse würde mich noch interessieren, was haben Sie an Storage für Anforderungen? Also haben Sie einfach ganz klassisch normal.. normale file basierte Storages, oder brauchen sie Objekt Storage

Gescheichten, S3 oder sowas? Oder ist das ganz klassisch einfach..

34 B: Also bei den jetzigen Applikationen die wir da onboarden haben wir eigentlich überhaupt keine Storage Anforderungen. Das ist dem geschuldet, dass eben diese .net Applikationen.. also wie es jetzt bei uns in-house betrieben wird.. rein mit einer MSSQL Datenbank arbeiten.. und... einem Caching Server Produkt. Also das heißt, diese Anforderung an persistente Daten ist jetzt bei uns intern noch nicht gegeben. Und ich habe es auch ganz selten bei Kundenprojekten, dass irgendwelche Object oder Blockstorages, oder auch filebasierte Storages, von der Applikation benötigt werden. Diese Anfroderung kommt eben.. auf Grund von Design aus der Plattform selbst heraus. Dass eben diese Metrikdaten und Loggingdaten mit dem EFK Stack.. jetzt wenn man redet von OpenShift.. einen Blockstorage brauchen. Also das ist aktuell die einzige Anforderung die wir an Storage haben, von der Plattform selbst. Von Kundenseite aktuell eher weniger die Anforderung persistent storage verwenden zu müssen.

..Technologiestack

..[AF15] Persistenter Speicher

..[AF19] Block Storage

35 I: OK, sehr interessant. Bei uns geht es genau in die andere Richtung.

36 B: Ja aber da haben wir gerade ganz.. den Vorteil, dass eben die Entwicklung sehr stateless passiert. Also die Container sind wirklich sehr stateless und das einzige das dann eben persistent gehalten wird ist in einer MSSQL Datenbank, bzw in einem Cache drinnen. Wobei der darf angeblich flüchtig sein, da gehts dann wirklich nur darum, dass der shared ist.

37 I: OK, sehr gut. Dann wären wir eigentlich schon beim letzten Punkt wo es geht um Qualitätseigenschaften. Damit werden wirklich Eigenschaften bezeichnet anhand deren man vor Allem Produkte, wenn man jetzt mehrere Produkte zur Evaluierung hat, unterscheiden könnte. Die zum größten Teil feature-gleich sind, aber dann zum Beispiel sagen kann, Preis ist zum Beispiel eine Eigenschaft. Nach dem Motto billiger ist besser. Oder zum Beispiel time to get startet, wo ich sagen kann, eine Applikation oder ein Produkt das ich schneller installieren kann, ist für mich eher vorteilhafter. Also gibt es gewisse Eigenschaften, wo Sie im Unternehmen X sagen da legen Sie viel Wert drauf, wenn Sie sich für ein Produkt entscheiden müssten? Aus mehreren Produkten heraus.

38 B: Ja das ist jetzt wieder so ein Äpfel und Birnen vergleich. Wir haben jetzt bei den Kundengesprächen sehr oft erkannt, dass eben aktuell in diesem Enterprise Umfeld die VMWare sehr in den Markt herein drückt. Da wird VMWare Tanzu in der neuen Lizensierungsvariante das eben... sehr günstig eben an bestehende VMWare Kunden mitliefert. Die RedHat hat in dem Bereich monetäres nachgezogen und so etwas wie OpenShift light entwickelt. Das ist jetzt meiner Meinung nach preislich im sehr überschaubaren Bereich was das... im Enterprise Umfeld bewegt. Ich habe aber auch schon andere Kundenanforderungen gehabt, die haben gesagt sie wollen eigentlich nur normales Kubernetes betreiben das... sie in dem Sinn gratis aus dem Internet downloaden wollen. Denen habe ich dann schon immer wieder ins

..[AR09] Strategesche Partner

..[AQ03] Preis/Lizenzkosten

..[AR03] Enterprise Support

Gewissen geredet in dem Bereich. Ihr müsst euch dann schon vorstellen, dass ich dann keinen Herstellersupport habt, weil ihr eben so eine frei verfügbare Softwarvariante verwendet. Wie könnt ihr dann einen möglichen Ausfall dieser Plattform dann rechtfertigen, wenn es dann.. Ich kann mir jetzt hierzu keinen kommerziellen Support mehr hinzukaufen. Ich muss dann also hierfür die Verantwortung zu 100% selber übernehmen, und auch die Folgen tragen. Also.. das ist schon eben auch ein Punkt wo man sagt, da sollte man nicht sparen in dem Bereich. Weil wir haben selber schon den Fall gehabt, wenn man irgendwelche Probleme hat, da sind wir schon auch auf den Support angewiesen, und der kann einem dann auch helfen. Da redet man dann teilweise schon von betriebsrelevanten Ausfällen. Die Verfügbarkeit der Plattform an sich ist ja sehr sehr hoch. Also man geht ja den Ansatz, dass man sowieso immer eine hochverfügbare Installation aufsetzt. Aber es kann halt immer wieder passieren, dass irgendwelche Fehlfunktionen eintreten und irgendwelche Ausfälle produzieren. Also monetäre Entscheidung obliegt da sowieso meist dem Management. Da haben wir dann als Techniker in dem Sinn wenig Mitspracherecht. Aber bei den meisten Kunden kann man sagen, sie tendieren schon zu kommerziellen Enterprise Lösungen. Zumindest sind wir hier sehr stark vertreten in diesem Bereich. Man bekommt es von eher kleineren Firmen auch mit, die liebäugeln... weil sie schon Erfahrung haben mit irgendwelchen Cloud Varianten wie Azure oder AWS, aber auch die lassen sie sich dann auch was kosten.

39 I: OK. Gut, dann denke ich hat das meine Fragen soweit relativ gut beantwortet. Vielen Dank!

A.26 Interview: Protokoll Unternehmen E

1 I: Ja gut, also wie gesagt vielen Dank, dass Sie sich die Zeit nehmen. Kurz zum Anfang, mein Name ist Daniel Drack, ich habe Ihre Kontaktdaten von Herrn P bekommen. Ich bin auch im Unternehmen A, in der IT Infrastruktur tätig. Nebenbei studiere ich Innovationsmanagement am Campus02, mache dort gerade meinen Master. Im Zuge meiner Masterarbeit beschäftige ich mich mit dem Thema des Designs von Kubernetes Clustern. Erstens weil es eine sehr innovative Technologie ist und zweitens weil wir im Unternehmen A mittlerweile auch einge Cluster haben. Also ich bin im DevOps Team und mache das auch beruflich. Wir haben halt gesehen, dass es hier einen need gibt das Design solcher Systeme etwas strukturierter anzugehen. Grundsätzlich würde ich das Ganze jetzt so gestalten... ich würde Sie bitten sich kurz vorzustellen, kurz zu sagen wer Sie sind und was Sie so machen. Welche Kubernetes Umgebung Sie im Einsatz haben und was das in Ihrer Firma so an... oder wie das verwendet wird. Danach würde ich gerne diese drei bzw. vier Themen durchgehen. Also einmal den Designprozess, dann diese Rahmenbedingungen, funktionale Anforderungen und Qualitätseigenschaften. Ja, dann darf ich Sie bitten sich vorzustellen und ein bisschen zu erzählen für was Sie im Unternehmen X Kubernetes verwenden.

2 B: OK. Sagen wir so, ich bin Unternehmen Y, wir sind ein.. sagen wir so.. wir sind eine Schwester zu Unternehmen X. Da gibt es die Unternehmensgruppe, darunter hängen die Holdings, also die Länderfirmen, und das Unternehmen Y ist eine davon. Ich habe aber eine Historie im Unternehmen X mit 17 Jahren in der Infrastruktur. Ich war dort für den Betrieb von Rechenzentrumsinfrastruktur zuständig. Also klassische Unix Umgebungen, HP-X, AIX und ähnliches halt. In den letzten Jahren natürlich sehr viel VMWare, Linux und sonstige x86 Umgebung. Ich habe vor knapp drei Jahren ins Unternehmen Y gewechselt und das Unternehmen Y hat 2017 die Firma Z gekauft. Die Firma Z ist ein schweizer Serviceprovider. Die sind.. nennen wir es höflich, ein kleiner Konkurrent zu AWS. Was das IaaS Portfolio angeht, und wo auch das PaaS Portfolio langsam wächst. Und, die Firma Z selbst... ich habe zwei Seiten. Ich arbeite einerseits für das Unternehmen Y, dort bin ich zuständig als Firma Z pre-sales Consultant. Ich unterstütze Kunden bei der Migration in die Cloud, oder..[..]. Andererseits bin ich zwiegespalten, ich trete hin und wieder auch als Firma Z Mitarbeiter auf, weil die Firma Z... ich will mich nicht festnageln aber so 97 bis 98% mittlerweile dem Unternehmen Y gehört, und unsere Cloudanbindung ist. Also einerseits bin ich User der Firma Z, das heißt das Unternehmen Y verwendet die Firma Z selbst als Cloud Plattform. Firma Z bieter aber auch relativ vielen Kunden ihre Services an. Andererseits benutzt die Firma Z auch sich selbst und ihre eigenen Angebote, die wir auf den Markt bringen. Das also mal zu meiner Historie. Wo setzen wir Kubernetes ein? Einerseits setzten wir, also Unternehmen Y, Kubernetes für unsere Car sharing Plattform ein. Das ist ein Rancher geführtes Kubernetes. Dann für interne Zwecke, gerade was Entwicklung angeht, ist Kubernetes based. Und unsere Kollegen aus der IT-Security Abteilung, die setzten auch Kubernetes ein für irgendwelche Penetrationstests. Wenn man die Firma Z betrachtet, also diese ist eine eigene

..Kubernetes Architektur
..Stellenbeschreibung
..Vorschlag für Kubernetes
..Technologiestack
..Art der Workload

Business Entität, also eine eigene Firma die parallel geführt wird. Die Firma Z hat den Ansatz, das was sie entwickeln.. sie verfolgen den Ansatz, dass das was sie entwickeln das soll so sein, dass sie es auch selbst benutzen können. Also es macht wenig Sinn wenn ich etwas entwickle das ich selbst für mich nicht benutzen kann, weil dann ist die Wahrscheinlichkeit, dass am Markt vorbei entwickelt wird ziemlich groß. Und das heißt die Firma Z hat selbst angefangen ihre Sachen die vorher natürlich nicht Kubernetes based, oder sagen wir nicht Container based waren, mittlerweile doch.. Container based umzuschreiben und zu entwickeln. Und hat da auch einige Sachen in Richtung Kubernetes migriert und umgewandelt. Also vom Portal, API und sonstige Sachen die halt die üblichen Cloud Provider auch zur Verfügung stellen. Das Billing System wurde mittlerweile auch migriert.

..Art der Workload

3 I: Darf ich fragen was sie da ungefähr für einen Technologiestack verwenden? Sowohl für Kubernetes als auch für die Applikationen die darauf laufen.

4 B: Das ist sehr unterschiedlich. Die Firma Z schreibt viel auf Cloujure, also die Firma Z ist Cloujure based, als Programmiersprache. Die verwenden großteils plain Kubernetes, da ist nicht viel anderes dabei. In der Firma Y ist eher.. ist unterschiedlich. Da sind wir nicht wirklich die Software Entwickler, da sind wir mehr der Operations Teil. Das Unternehmen Y hat selbst keine Entwicklung, das heißt wir suchen immer Partner mit denen wir zusammen entwickeln. Aber dort ist eher mehr Rancher basierte Umgebung mit normalem Kubernetes darunter.

..Technologiestack
..Vorschlag für Kubernetes

..Technologiestack
..Vorschlag für Kubernetes

5 I: Also Rancher als Management Plane, und die Cluster upstream.

6 B: Genau richtig. Weil dann hat man.. der Vorteil von Rancher ist eben zentrales Management. Das ist halt bei der Firma Z eben alles.. nachdem die eine Entwicklungsfirma sind, haben die sich sehr viel selbst entwicklelt. Die machen de facto alles automatisiert, also Config Management, User Management und sonstiges. Also das ist dann wahrscheinlich in einer Firma mit unterschiedlichen Applikationen dann nicht mehr so ganz leicht.

7 I: OK, das heißt die Firma Z verwendet nicht Rancher.

..Technologiestack
..[AR01] Open-Source Lizenzen
..Technologiestack

8 B: Die Firma Z verwendet nicht Rancher, die Firma Z hat so eine Philosophie... die Firma Z sagt sie verwendet kein fremdes IP. Also verwendet sie schon aber zahlt keine Lizenzen. Das heißt die verwenden wirklich plain Kubernetes, derzeit Version 1.18 oder 1.19. Plus, das was man auch sagen kann. Die Firma Z bringt jetzt gerade ihr eigenes Kubernetes raus, als Produkt am Markt. Also, dass der Endkunde sich bei der Firma Z sich seinen Kubernetes Cluster bestellen kann. Und bei dem wird es so sein, dass der Kubernetes den der Endkunde bekommt... also der Kunde bekommt dann seine managed Masters. Die rennen bei der Firma Z im Kubernetes verteilt. Das heißt der etcd rennt in Wirklichkeit in einem Container der in einem Kubernetes rennt. Die schauen halt wirklich, dass sie das Produkt das sie entwickeln, oder das sie intern verwenden, auch wirklich so verwenden, dass sie es dann auch für den Kunden verfügbar machen können. Und die ganzen Vorteile

..Art der Workload

..Art der Workload

von Kubernetes halt ausnützen können, die das so bringt.

9

I: Darf ich hier vielleicht gleich ins nächste Thema einsteigen? Sowohl beim Unternehmen Y als auch bei der Firma Z kann man da sagen, wer da wirklich die Verwendung von Kubernetes entschieden oder vorgeschlagen hat. Also, dass Container Technologie und Container Orchestrierung verwendet wird?

10

..Entscheidung für Kubernetes

..Technologiestack

..Motivation für Kubernetes

B: Also bei der Firma Z.. sagen wir so.. die Firma Z... um es zu vergleichen die Firma Z ist ein Unternehmen mit 50 Leuten. Davon sind 25 Entwickler, so ungefähr, und 25 sind... sagen wir so, 20 sind Entwickler, 20 sind Ops und der Rest ist Overhead wie CTO, CEO, Finance usw. Und dort ist der need halt von den Entwicklern gekommen und von den Ops Leuten. Also definitiv weil die einfacher zu deployen, einfach zu entwickeln sind und was die Release Zyklen angeht... weil die leben halt wirklich, also die Firma Z lebt halt wirklich.. was den approval Prozess angeht, die leben halt wirklich im GitHub, unter Anführungszeichen. Also dort wird halt wirklich im GitHub die Approvals durchgemacht, wenn es soweit ist gibt es das Release, gibst das Deployment da rein, wird automatisch die CICD pipeline... wird dort getestet und wenn halt der Test in Ordnung ist geht es in Produktion mit dem nächsten Rollout. Im Unternehmen Y oder im Unternehmen X generell würde ich sagen, dass es über Lieferanten hereingetragen wurde, über eine große Firma. Weil es natürlich irgenwelche Buzzzwords sind bzw. Schlagwörter sind, und weil es für die Zukunft einfacher ist, kleinere Services zu pflegen als so riesige monolithische Systeme wo man nicht weiß, wenn man etwas angreift, wie sich das auswirkt auf das ganze Drumherum.

11

I: Dann vielleicht, ja eigentlich auch Richtung beide Firmen gerichtet. Können Sie mir sagen wer oder welche Stellen wirklich am Designprozess der Umgebungen beteiligt waren? Also wer hat gesagt, dass Sie Rancher verwenden, oder dass Sie nur plain upstream fahren?

12

..Entscheidung für Kubernetes

..[AR05] GUI

..[AF14] RBAC Integration

..backup/restore out-of-the-box

B: Also bei Firma Z kommt das sicher sehr stark aus der Ops und aus der... dort ist sicher die Development Ecke sehr stark und das Ops Team. Und das... dass auch der Fokus.. sagen wir so, dass die da wenig fremde Sachen einsetzen. Also keine Management Tools oben drüber. Im Unternehmen Y war das eher eine Entscheidung aus dem IT Ops Team. Also da gibt es ein Operations Department, die den Betrieb von allen möglichen Applikationen im Unernehmen machen, und die auch Services erbringen wenn es an Endkunden geht. Und die haben halt den need, dass es für sie halt einfacher geht. Deshalb Rancher, also gerade was Userverwaltung angeht, [..], backup und restore. Oder wenn wirklich die Instanzen wachsen müssen, dann ist das dort drin halt leichter zu machen, als wenn man sich das wirklich von scratch an lernen muss.

13

..Entscheidung für Kubernetes

I: War dort auch irgend eine Form von Management beteiligt, oder noch andere Abteilungen? Sowas wie IT-Security, Projektmanagement, Prozessmanagement oder sowas in diese Richtung?

14

B: Glaub ich gar nicht, nein. Also im Unternehmen Y sicher nicht. Im

..Entscheidung für Kubernetes
..[AQ03] Preis/Lizenzkosten
..OpenShift kompatibel
..Technologiestack
..[AQ03] Preis/Lizenzkosten
..Entscheidung für Kubernete

Unternehmen X ist dort sicher IT sicher involviert, Operations Team sehr involviert. Also die setzen dort keinen Rancher ein, die seitzen OpenShift ein im Unternehmen X. Oder wollen zumindest OpenShift einsetzen, daher ist es auch ein gewisser Kostenpunkt der diskutiert werden muss. Also in den 17 Jahren in denen ich im Unternehmen X war habe ich gelernt, dass es solange es.. unter Anführungszeichen von den Kosten her egal ist, dann ist die Entscheidung meist bei denen die es wirklich betreiben oder machen müssen. Das Problem ist, umso teurer ein Produkt wird, bzw. umso mehr die Kosten werden, umso mehr sprechen auch mit was das ganze angeht. Also man wird halt schwer dann selbst argumentieren können wenn man.. keine Ahnung, ich muss jetzt zwei Millionen Euro in die Hand nehmen, für Subscriptions für OpenShift zum Beispiel, dann muss man schon großes Standing haben damit man die zwei Millionen Euro bekommt für sowas.

15 I: Das bringt mich gleich nahtlos zum nächsten Punkt, und zwar Rahmenbedingungen. Ich nehme jetzt mal mit, dass eine bestimmte Kostenobergrenze so eine Rahmenbedingung wäre. Aber wenn Sie sich jetzt als Unternehmen, egal welches, für eine Distribution oder auch für ein Produkt, zB ein Ingress Controller, ein Storage System etc. entscheiden müssten. Was sind so Rahmenbedingungen wo Sie sagen würden, da würde ein Herstelle von vorne herein rausfallen?

..Ausgereiftheit [AQ17]
..[AQ03] Preis/Lizenzkosten
..[AR03] Enterprise Support
..[AR01] Open-Source Lizenzen

16 B: Also einerseits sollte meiner Meinung nach das Produkt schon eine gewisse Roadmap hinter sich haben. Also Version 1.0 ist eventuell nicht immer der optimal Zeitpunkt um irgendwo einzusteigen. Die Kosten sind langfristig ein Thema. Also Kosten und Lizenzmodell sind ein gewisses Thema. Also.. wobei gerade in der open-source Welt.. bei RedHat gibt es kein Lizenzmodell sondern ein Subscription Modell was auch den Support angeht. Das ist ein Thema, das kann sehr böse sein oder kann böse enden wenn man hier nicht aufpasst, was die Variante angeht. Ja dann kommt es halt darauf an wie sehr möchte ich mich absichern, wenn es wirklich zu Fehlern kommt gegenüber dem Hersteller. Also einfaches Beispiel, man muss sich nur Debian und Ubuntu anschauen. Ich würde sagen die zwei sind per se zu wahrscheinlich 99% das gleiche Produkt. Würde vielleicht nicht jeder zu 100% unterschreiben, aber bei Ubuntu habe ich halt im worst case jemanden dahinter den ich sieben mal 24 anrufen kann, wenn ich irgend ein Problem mit meinem Produkt habe. Bei einem Debian, ich kann es in ein Supportforum rein schreiben, kann hoffen, dass sich vielleicht jemand meldet und den Fehler findet. Ja, das sind halt so die Sachen.

17 I: Das heißt das würde dann so in die Kerbe Enterprise Support schlagen?

..[AR03] Enterprise Support
..[AQ02] Verfügbarkeit in %

18 B: Es reicht glaube ich nicht Enterprise Support und wie viel also... also wir hatten im Unternehmen X früher einen Abteilungsleiter der hat gesagt: "Wie viele Hosenträger möchte ich bei gewissen Sachen an haben, um sicher zu sein?". Wie viel Risiko traue ich mich auch zu nehmen. Wenn das vielleicht ein... um es in der Unternehmenssprache zu sagen ein TelKo System ist, bei dem sehr viel Revenue drüber rennt und ich einen Ausfall habe von einer

..[AR03] Enterprise Support
..[AQ02] Verfügbarkeit in %

Stunde. Dann kostet mich das vielleicht Geld, dann ist das direkter Schaden den ich verursache. Wenn ein internes System nicht wenn eine interne Website nicht verfügbar ist, dann wird das wahrscheinlich eher egal sein.

19 I: Vielleicht noch in eine andere Richtung gedacht. Haben Sie jetzt, vor allem das Unternehmen X als Infrastrukturprovider, auch gewisse gesetzliche Vorgaben, oder gewisse gesellschaftliche Vorgaben, dass Sie gewisse Hersteller verwenden können oder gewisse nicht verwenden dürfen?

..politische Vorgaben/Herstellerstandort |

20 B: Ich kann nur aus meiner alten Zeit im Unternehmen X sprechen und da hat es keine Vorgaben gegeben. Nicht, dass ich wüsste, dass es Vorgaben gegeben hat. Also so wie jetzt die Diskussion die es mit Huawei giebt, ob man die zum Ausbau von 5G will. Würde mich nicht erinnern, dass es solche Diskussionen gegeben hat.

21 I: OK, also das heißt man ist auf der Seite relativ frei in der Herstellerwahl.

..Unternehmensgröße [AQ12]

22 B: Also.. relativ frei.. wobei.. ehrlich gesagt, es gibt nicht so viele große Hersteller. Also wenn man sich die Hersteller anschaut die es gibt, es gibt.. RedHat, es gibt SUSE, die mittlerweile nicht mehr so groß ist, zumindest in Europa kommt mir das so vor. Dann gibt es doch.. die amerikanischen Firmen die den Markt recht gut in der Hand haben, wenn es um Software geht. Die HP, oder die HPE wie es früher war, die hat sich am Softwaremarkt selbst vernichtet in den letzten zehn Jahren. IBM hatte ich wenig Kontakt, das gebe ich offen zu, als Software Lieferant, hat sich jetzt aber natürlich die RedHat einverleibt. Dann gibt es natürlich noch einige Hersteller aus Israel. Wobei die nicht so auf der Software Seite.. also wenn man sich anschaut [..] zum Beispiel. Also gerade was Sicherheitsprodukte, Securityprodukte angeht, kommt viel aus Israel. Hat Vor- und Nachteile gegenüber.. das eine sind amerikanisch geführte Unternehmen wo man nicht ganz weiß was mit den Daten passiert, das andere sind israelische Firmen die nicht gerade so unbekannt sind, dass sie gerne herumschnüffeln. Auf der dritten Seite hat man chinesische Firmen wo man sich jetzt auch überlegen kann wie sehr die staatlich gelenkt sind.

23 I: Ist das etwas was bei Ihnen als.. ich will nicht sagen staatliches Unternehmen.. aber als zumindest österreichweites, vorherrschendes Unternehmen, politisch wichtig ist wo ein Unternehmen herkommt? Ist das irgendwie ausschlaggebend, oder ist das egal?

..politische Vorgaben/Herstellerstandort |

24 B: Also auf meiner Ebene hätte ich das nicht mitbekommen, muss ich ganz ehrlich sagen. Also ich kann es nur sagen von innerhalb des Unternehmen X, wir haben doch.. gerade auf der Produkt X Seite mit allen die am Markt vertreten sind irgendwas zu tun. Also wir hatten Nokia, Ericsson, wir haben jetzt Huawei, wir hatten früher Nortel im Haus, also das ist alles sehr unterschiedlich. Aber ich hätte jetzt nicht mitbekommen dass politisch irgendjemand Druck gemacht hätte.

25 I: OK, dann vielleicht der nächste Fragenkomplex der da angrenzt.

Wie schaut es denn aus mit funktionalen Anforderungen? Also vor allem auf technischer Ebene. Ich meine jetzt nicht standard Kubernetes Features, davon gehen wir aus. Gibt es irgendwas das Sie im Unternehmen sagen, das ist eine spezielle funktionale Anforderung die Sie haben, an den Cluster? Stichwort: hybride Cluster, Windows Betriebssystem, GPU Support oder irgendwas in diese Richtung?

26 B: Naja also sagen wir so, innerhalb der Firma Z nein, da sind wir relativ plain was das angeht. Innerhalb der Firma Y würde ich auch nicht, sagen, dass wir große Anforderungen haben. Aber wenn wir in Richtung Kundenmarkt betrachtet.. also nachdem ich ja auch pre-sales mache, wenn etwas herein kommt. Also Verschlüsselung ist definitiv ein Thema, das kommt immer wieder.

27 I: Data-at-rest oder Netzwerk Verschlüsselung?

..verschlüsselte Kommunikation [AF21

..[AF02] data-at-rest Encryption

28 B: Beides. Sagen wir so, Netzwerkverschlüsselung ist mittlerweile Standard. Also, dass die Daten zwischen den Servern verschlüsselt herumgehen müssen ist fast Standard.

29 I: Sie denken auch in Richtung Service Meshes, oder in Richtung CNI Plugins die Verschlüsseln?

30 B: Genau richtig, also solche Sachen. Also gerade Istio ist ist natürlich ein Schlagwort in die Richtung auf der RedHat Seite, was einiges macht. Google Anthos geht auch in diese Richtung. Also das ist de facto.. Sagen wir so.. Also wenn man Kubernetes betreibt, oder wenn man diese Service Architekturen betreibt hat man im Endeffekt [..], dann ist auf jeden Fall alles verschlüsselt. Und ich möchte mich als Entwickler jetzt nicht darum kümmern Verschlüsselung in meine Applikation einzubauen, oder mich vielleicht sogar noch um das SSL Management zu kümmern, das wäre wirklich zu mühsam. Das heißt das wird mittlerweile als Standard betrachtet. Oder auch Load Balancer oder sonstige Sachen sind hier halt out-of-the-box drinnen, dass man die benutzen kann. Ohne mich jetzt darum zu kümmern wie viele Pods dahinter sind oder sonstiges.

..verschlüsselte Kommunikation [AF21]

..[AF12] Layer 4 / Load Balancer

31 I: Das heißt, Sie bieten auch sowas wie zum Beispiel MetalLB oder, ich glaube OpenShift hat eine eigene Implementierung. Also die wirklich auch Layer vier connectivity in den Cluster anbietet. Also, dass LoadBalancer als Service Objekt in diesem Sinn verwendet werden kann.

32 B: Genau, also was wir getestet haben.. Wir haben von F5 die Integration getestet. Also die machen wirklich, dass da Hardware Load Balancer über Pods kommunizieren mit Kubernetes, und damit den Load Balancer zur Verfügung stellt und automatisch deployed, also die Rules. Solche Sachen, plus in der Firma Z haben wir einen selbst entwickelten Load Balancer der das auf IG Basis macht und damit integriert. Im Unternehmen X, also das Unternehmen ist ja ein großer Cisco Kunde, die machen.. wir wollen Cisco ACI einsetzen für das. Wo es dann eben ein OpenShift CNI Plugin gibt.

..[AF12] Layer 4 / Load Balancer

..Technologiestack

..Integration in bestehende Infrasti

33 I: Darf ich da kurz dazwischen fragen: Verwenden Sie das schon erfolgreich? Oder ist das auf der Road Map?

34 B: Also ich weiß, dass es auf der Road Map ist von denen. Ich weiß, dass sie Cisco ACI einsetzen, also nicht auf der OpenShift Seite, sondern Cisco ACI auf der... mit vCloud Director gemeinesam um Kundenumgebungen zu deployen und zu erstellen. Also das einfach zu machen. Für OpenShift ist es auf der Road Map von ihnen, dass sie das machen. Und damit Durchgängigkeit haben was das LAN angeht. Wobei, persönlich bin ich nicht ganz davon überzeugt, dass es sinnvoll ist, bis vielleicht auf den Load Balancer. Weil im.. die Firma X hat ein relativ starres altes Sicherheitskonzept, oder Zonenkonzept wie sie das nennt. Das heißt die Firma hat sechs Zonen, also Security Layers. Eigentlich hat sie sieben, weil die Zone 0 ist halt das Internet. Und dann hat sie sechs interne Zonen, die halt festlegen wie business kritisch oder unternehmenskritisch etwas ist. In Zone eins sind zum Beispiel Webserver, und das geht dann halt.. in Zone zwei sind Kundenanbindungen, in Zone drei sind typischerweise Desktop PCs von Mitarbeitern. Zone vier sind interne Services die gebraucht werden. Zone fünf sind Datenbanksysteme und Zone sechs sind hochkritische Sachen wie jegliches Equipment, ein Billing System usw., wo man halt nicht will, dass der Kunde direkt hinkommt.

..Integration in bestehende Infrastruktur [

35 I: Das wird dann alles über ACI gemanaged?

36 B: Genau, das soll über ACI gemanaged werden. Denn historisch bedingt bedient das Unternehmen ja mehrere Rechenzentren die nicht alle den selben Stand haben. Also die haben da einiges an Investment, wenn man das gerade zieht. Also es ist ja nicht so, dass das so einfach geht. Meistens sind auch irgendwelche downtimes und Wartungsarbeiten beinhaltet, das ich auch nicht so ungefährlich. Und wenn ich jetzt aber an OpenShift denke, oder wenn ich generell an Kubernetes denke, dann habe ich in Kubernetes ja innerhalb des Systems mein abstrahiertes Netzwerk. Welches jetzt über network policies voneinander abgeschottet werden kann, oder über Namespace.. also IP Spaces, Namespaces, sonstiges, werden die Sachen zueinander abgeschottet. Und meistens spreche ich dann von außerhalb über meinen Load Balancer oder meinen Ingress Controller mein Service an. Wenn ich das aber jetzt auf unterschiedliche Zonen noch runter brechen muss. Also im worst case muss ich vier Zonen machen wo ich meine Server stehen habe. Da kann das schon eine rechte Challenge werden, denn das heißt ich muss auch meine ganzen Container in diesen Policies abbilden. Und denen alle möglichen Labels mitgeben, damit der Developer nicht unabsichtlich das Ding in die falsche Zone deployed. Plus ich muss Sorge tragen, dass die Container dann nur in der Zone miteinander reden dürfen. Also ist die Frage nochmal, ob dieses Kubernetes Konzept nicht langfristig [..] wird. Wo es ja doch sehr einfach und sehr applikationsnahe verwalten können, im Gegensatz zu alten Systemen wo ich eine Firewall habe die das LAN based macht, oder IP based, je nachdem wie man es betrachtet. Das eben in ein Deployment rein zu kriegen ist natürlich schwer. Also jetz mal als.. das wirklich über Labels oder die Objekte im Kubernetes mitverwalten kann. Aber das ist derzeit ihr Plan, also in dem Fall OpenShift mit ACI Plugin zum Laufen

..[AQ02] Verfügbarkeit in %

..[AF07] Pod2Pod Kommunikation

kriegen.

37 I: Spannend.

38 B: GPU Support ja, also GPU Support kommt immer wieder. Es gibt aber unterschiedliche Auffassungen ob GPUs gut oder schlecht sind. Also ja, es gibt use-cases wo es definitiv gut ist, gerade Machine Learning oder AI. Es gibt aber auch sehr wohl Kunden die sagen sie rechnen lieber auf einer CPU, oder auf mehreren CPUs weil es für sie schneller und günstiger ist. Also da muss man immer sehr genau wissen, seinen business case dahinter.

..[AF04] GPU Support

39 I: Aber grundsätzlich kommt die Anforderung auch vor?

40 B: Kommt.. grundsätzlich kommt die Anforderung vor, und ist jetzt nichts ungewöhnliches.

41 I: Gibt es sonst noch irgendwas technisches wo Sie sagen das ist vielleicht eine Anforderung die ein normaler Kubernetes Kunde oder Benutzer in dem Sinn jetzt nicht haben würde? Oder haben wir mit dem jetzt eigentlich alles?

42 B: Nein also... das was alle Kunden haben wollen, die wollen halt intern ein Reporting bzw. Monitoring dazu haben. Das ist meiner Meinung nach die Grundlage davon, damit ich sowas überhaupt betreuen kann. Ahm.. dann geht es halt meistens schon ins spezifische rein, also manche Kunden machen dann wirklich fertige CICD Pipelines in so ein Produkt rein. Also die holen wirklich aus dem Github oder BitBucket... also wo man die Approval Prozesse machen kann.. oder auch einen Jenkins anbinden kann an solche sachen. Das ist dann aber.. da gehts wirklich sehr ins spezifische rein. Das sind dann Sachen die nicht von meinen Kunden oder anderen Kunden repliziert werden können. Also die auch nicht komplett übernommen werden können.

..[AF08] Metriken

..Technologiestack

43 I: Ich verstehe. Okay, dann wären wir eigentlich schon bei der letzten Gruppe und zwar Qualitätseigenschaften. Das beschreibt eigentlich Eigenschaften. Wenn ich jetzt zum Beispiel zwei Produkte habe die ich miteinander vergleiche, zum Beispiel zwei Ingress Controller. Die sind funktional gleichwertig, anhand welcher Eigenschaften würden Sie sich dann für einen... würden Sie versuchen diese Produkte dann zu unterscheiden? Zum Beispiel habe ich von anderen Interviewpartnern schon gehört, der Preis natürlich, time to get started und solche Sachen. Verfügbarkeit zum Beispiel, also Eigenschaften mit der ich die Qualität eines Produktes oder auch eine Clusters im weiteren Sinne bestimmen könnte. Man könnte auch sagen KPIs.

44 B: Sagen wir so.. Also am Ende des Tages.. einige Punkte wie Installation, Aufwand etc. mündet am Ende des Tages wieder in Kosten bzw. Personalbedarf. Das ist meistens in vielen Firmen ein.. also Personalbedarf.. das ist in vielen Firmen ein kritischer Punkt. Personal sind immer Kosten. Das heißt da ist eigentlich.. naja Verfügbarkeit bzw. Uptime, also ich glaube nicht, dass man sich für ein Produkt entscheiden würde das schlechte Uptime liefert. Also ich würde da schon eher nach Produkten gehen wo ich weiß, dass

..[AQ04] Betriebspersonal Aufwand

..[AQ08] Durchlaufzeit/Aufwand fü

..[AQ02] Verfügbarkeit in %

..[AQ02] Verfügbarkeit in %
..[AQ05] Marktanteil
..Benchmarks [AQ13]

sie das erfüllen oder erfüllen können. Marktanteil.. ja also ich glaube am Ende wird es bei vielen auf den Kosten hängen bleiben. Das was ich gelernt habe.. bitte nicht immer dem Gartner Quadrant vertrauen. Also auch wenn Produkte hoch geranked sind müssen sie nicht gut sein. Haben wir selbst beim backup tool am eigenen Leibe erfahren. Dass dies einfach ein bisschen Lobby Arbeit ist, was im Gartner Quadranten ist. Und das was man wirklich oft hinterfragen muss, ist.. wie viel von dem Produkt das ich kaufe brauche ich davon? Und wie viel ist nice to have? Also was sind wirklich meine must haves, die ich wirklich erfüllen muss? Was ist vielleicht nett wenn ich es habe, und was weiß ich dass ich eh nie verwende davon?

45 I: Das heißt das wäre auf jeden fall irgendwo auch interessant eine gewisse Skalierbarkeit bei einem Produkt zu haben. Dass man sagen kann: "Das brauche ich und das möchte ich bezahlen, aber ich brauche nicht alles".

..[AF08] Metriken
..backup/restore out-of-the-box [AF22

46 B: Genau richtig. Also wie gesagt, mir ist wichtig ich habe ein Monitoring, ich habe ein Backup, ich habe ein funktionierendes Recovery. Das sind meine Grundpunkte die ich unbedingt brauche. Aber mir ist jetzt nicht wichtig, dass ich GPU Support habe.. es gibt keinen Use-Case wo ich das brauche. Dann ist es vielleicht der Punkt bei dem ich Abstriche machen kann und auch den Preis runter drücke. Oder auch die.. das was man auch nicht unterschätzen darf bei dem ganzen Personalbedarf sind auch Schulungen. Wenn ich mir zum Beispiel ein OpenShift anschaue, wenn ich das wirklich gut betreuen will oder auch regelmäßig betreuen will, brauche ich ein paar Leute die das, meiner Meinung nach, Tag ein Tag aus machen. Die sich damit auskennen! Generell braucht man, wenn man sowas sieben mal 24 anbietet, dann braucht man mal mindestens vier Leute damit man das machen kann, gesetzlich in Österreich.

..[AQ09] Schulung Admins

47 I: Also gerade zwecks Bereitschaftsrad usw?

..[AQ04] Betriebspersonal Aufwand

48 B: Genau, genau zwecks Bereitschaftsrad. Und vier Leute ist jetzt aber das Minimum. Das heißt.. Davon wir einer krank oder einer will auf Urlaub gehen, dann sind wir schon darunter. Das heißt höchstwahrscheinlich sechs bis acht wäre da sinnvoller, dass man das machen kann. Und da ist wahrscheinlich gut, wenn man sich auf ein zwei Produkte fokusiert, die wirklich der Kernfokus sind die man macht. Ich persönlich glaube nicht, dass einer zum Beispiel im Produkt OpenShift zu 100% und nebenher vielleicht noch VMWare vCloud Director in allen Ausprägungen so nebenbei betreuen kann, dass er auch um drei in der früh wenn er Pech hat und angerufen wird, performen kann.

49 I: Das sehe ich gleich, ja.

..[AQ06] Communitygröße und -qualität

50 B: Größe und Qualität der Community.. ja das ist sicherlich nicht zu unterschätzen, würde ich sagen. Ich glaube es gibt gute Software die jetzt nicht so ein große Community hat. Wenn ich mir zum Beispiel Proxmox anschaue, als österreichische Lösung. Dann ist das wahrscheinlich Software die gut ist und viel kann, die aber wahrscheinlich international gar nicht so bekannt ist.

51 I: Okay.

52 B: Also ich weiß nicht ob Sie Proxmox kennen?

53 I: Nein. Sagt mir glaube ich nichts.

54 B: Das ist eine KVM basierte Virtualisierung. Die sind eine österreichische Firma, die das komplett gut gemacht hat. Also die komplette Virtualisierungslösung. Die sind ein bisschen ein VMWare Gegenstück, sage ich mal so.

55 I: Das klingt interessant.

56 B: Natürlich um Ecken günstiger, wenn man es herunterrechnet, als eine VMWare. Plus die Jungs, die Proxmox, sind auch aktive Debian Entwickler zum Beispiel. Das heißt, da weiß ich wenn ich Probleme habe, die können mir auch helfen wahrscheinlich.

57 I: Das habe ich bisher nicht gekannt, aber das ist guter Input.

..[AQ03] Preis/Lizenzkosten

58 B: [..] Meiner Meinung nach werden am Ende nur die Kosten entscheidend sein. Also wie viel kostet es mich über X Jahre das zu betreiben und damit auszukommen.

59 I: Okay. Gut gibt's sonst noch eventuell etwas das Sie eventuell mitgeschrieben haben? Oder vorbereitet hätten?

..[AR01] Open-Source Lizenzen

..[AR03] Enterprise Support

60 B: Ist der Einsatz von Open-Source Software möglich? Also meiner Meinung nach gibt es wenig das nicht mehr open source ist. Also ausgenommen die Windows Welt, aber sonst glaube ich, ist viel open source oder beruht darauf. Verfügbarkeit.. ja also ich glaube, dass Kubernetes die Verfügbarkeit.. also eine Verfügbarkeit zu gewährleisten einfacher macht als manche anderen Methoden, sage ich mal. Mit dem ganzen Design rundherum. Und es ist vom Grund auf schon mal so geschrieben und designed, dass man etwas hoch verfügbar machen kann out-of-the-box. Sofern die Software auch dafür gedacht ist. Support, Enterprise Support.. ja ist wichtig aber das haben wir eh schon besprochen.

61 I: Gut, dann müssen wir es nicht weiter stapazieren. Vielen Danke auf jeden Fall für Ihre Zeit.

A.27 Interview: Protokoll Unternehmen F

1 I: Ja gut, also herzlich willkommen, vielen Danke. Wie ich es in der Einladung schon geschrieben habe, mein Name ist Daniel Drack, ich habe Ihren Kontakt von Person A. Ich mache gerade meinen Master in Innovationsmanagement am Campus02 und arbeite vollzeit beim Unternehmen A im DevOps Team. Also ich bin selbst Kubernetes Administrator und betreue das System bei uns. Deswegen habe ich mir das Thema auch ausgesucht. Ob es die richtige Wahl ist sei dahingestellt, aber zumindest eine interessante.

2 B: Ich habe gerade am Freitag eine Bachelorarbeit zum Thema Kubernetes im weitesten Sinn abgegeben.

3 I: Ja optimal, dann werden wir hier sicher super zusammenfinden. Grundsätzlich geht es bei mir darum, dass ich mir angeschaut habe wie man das Design von so einem Cluster ein bisschen strukturierter angehen könnte. Weil man halt oft das Problem sieht, dass Unternehmen da so rein stolpern und mal probieren Und wir gesehen haben, dass und das Zeit und Geld kostet wenn wir das nicht systematisch macht. Deshalb habe ich versucht einen Prozess oder eine Vorgehensweise zu entwickeln.. Prozess ist vielleicht sogar übertrieben. Aber eine Vorgehensweise zu entwickeln mit der man das etwas systematischer angehen könnte. Das Herzstück dessen ist ein Anforderungskatalog. Das heißt man muss ungefähr wissen was man auf dem Cluster betreiben will, oder auch nicht. Dieser Katalog soll dabei helfen so gesamtheitlich wie möglich zu betrachten welche Anforderungen man an das zukünftige System hat. Um dann im weiteren Schritt die richtige Distribution oder die passenden Produkte für die verschiedenen Cluster Komponenten auswählen zu können. Ich habe diesbezüglich schon zwei Workshops gemacht im Unternehmen A, um verschiedene Disziplinen im Unternehmen abzugreifen, und habe jetzt auch ein paar Interviews gemacht. Also Sie sind das letzte Interview jetzt. Um verschiedene Branchen abzuklappern, da vielleicht verschiedene Branchen, Finance zum Beispiel, andere Anforderungen hat wie wir in der Software Entwicklung. Genau.. Grundätzlich würde ich das Gespräch gerne so gestalten, dass Sie sich vielleicht mal vorstellen und dann würde ich auf diese Gruppen eingehen, die ich schon mitgeschickt habe. Also einmal den Designprozess selbst, Rahmenbedingungen, funktionale Anforderungen und Qualitätseigenschaften. Dann ein kleines Summary und dann wären wir eigentlich schon fertig. Ich denke das sollte sich in dieser halben Stunde schön ausgehen.

4 B: Probieren wir es. Wie stellen wir das an?

5 I: Dann würde ich einfach bitten, stelle Sie sich bitte kurz vor. Vielleicht noch vorab, ich werde Sie namentlich nicht erwähnen, also das aus dem Transkribt heraus nehmen. Also auch Ihr Unternehmen nicht namentlich erwähnen, das wird alles anonymisiert. Ich werde Ihr Unternehmen als Softwareunternehmen und Serviceprovider anführen. Das habe ich bei allen Unternehmen so gemacht.

6 B: Ich habe eh keine genauen Vorgaben wie es ist, aber

		anonymisieren ist sicherlich eine gute Idee. Okay, und vorstellen in dem Sinn was ich mache, oder?
	7	I: Genau, was Sie machen und dann auch bitte mal was Sie... also für was Sie Kubernetes im Einsatz haben und wie Ihre Umgebung und Workloads ungefähr ausschaut. Was Sie mir halt sagen dürfen.
..Technologiestack ..Stellenbeschreibung ..Technologiestack ..Technologiestack	8	B: Ja. Also grundsätzlich, ich bin als Cloud Engineer angestellt, das heißt so irgendwo zwischen Kubernetes und AWS. Ich habe da vor allem viel mit HELM und Terraform und so gemacht, in letzter Zeit. Und unsere Workload an sich sind Java Spring Applikationen, eine ordentliche Menge. Das ist glaube ich das Gröbste.
	9	I: Da Sie.. wenn ich gleich so einsteigen darf. Sie verwenden Kubernetes auch in der public cloud und das um Ihre Webservices zu hosten, die auf Spring basieren.
..Technologiestack ..Vorschlag für Kubernetes ..[AR10] Verwendung der public cloud	10	B: Genau. Also Spring und nodeJS ist das im Großen und Ganzen. Ähm.. nicht nodeJS, Spring und Angular 1 ist es im Großen und Ganzen. Und wir verwenden Kubernetes auf OpenStack, auf AWS und lokal.
	11	I: Lokal auch auf OpenStack?
..Technologiestack ..Single Node Kubernetes [AR14]	12	B: Also OpenStack.. nein. Lokal.. nein also OpenStack bei uns on-premises und lokal MicroK8s und Docker Desktop.
	13	I: OK, also wirklich lokal auf Ihrer Workstation?
	14	B: Genau. Ganz ganz lokal.
	15	I: Cool, so weit sind wir noch nicht. Okay..
	16	B: Bei uns ist es notwendig, da die Entwicklungsumgebung auch auf Kubernetes läuft.
	17	I: Das heißt Sie verwenden da für single node Kubernetes MicroK8s oder K3s, oder was für eine Distribution verwenden Sie da zum Entwickeln?
..Single Node Kubernetes [AR14]	18	B: MicroK8s auf Linux und Docker Desktop auf Windows.
	19	I: Okay. Haben Sie viele Linux Desktops?
	20	B: Ja schon, aber bald nicht mehr.
	21	I: Ja gut, vielleicht gleich in die erste Kategorie. Können Sie was dazu sagen wie das zustande gekommen ist, dass Sie Kubernetes verwenden? Wessen Idee das war und wie das so entstanden ist?
..Dauer Kubernetes Verwendung ..Vorschlag für Kubernetes ..Motivation für Kubernetes	22	B: Tatsächlich nein, weil ich erst danach dazugekommen bin. Ich bin schon, wie ich dazugekommen bin, vor einem Jahr ungefähr, war Kubernetes schon längst im Einsatz. Aber die Motivation war sicher einfach weil vorher das auch schon in Containern gelaufen ist. Und die Orchestrierung halt mit Kubernetes einfacher ist, wenn es mal rennt.

23 I: Wissen Sie wer da eventuell beteiligt war, bei der Entscheidung, dass dieses System verwendet wird? Und, dass dann wirklich auch dieser Stack, also in dem Fall OpenStack auf AWS [falsch verstanden] verwendet wird?

..Entscheidung für Kubernetes

24 B: Nein ich kann es nur zur lokalen Entwicklung sagen, weil da war ich es. Was davor passiert ist weiß ich in dem Detail nicht. Und weil es für mich jetzt nicht wirklich eine Bedeutung gehabt hat.

25 I: Okay, kein Problem. Gibt es bestimmte, also vielleicht in Richtung der single node Lösung, gibt es bestimmte Rahmenbedingungen in Ihrem Unternehmen wo Sie sagen das ist von Gott gegeben. Sprich das ist nicht verhandelbar. Also wenn Sie sich neue Produkte anschauen, Bedingungen was den Rahmen vorgeben?

26 B: Neue Produkte im Sinne von einer Kubernetes Distribution oder?

27 I: Sowohl als auch. Gerne natürlich auch Kubernetes im Vordergrund. Aber auch sonstige Infrastrukturkomponenten oder so. Also beispielsweise, wenn Sie jetzt einen neuen Ingress Controller evaluieren müssten, an was müssten Sie sich halten, auf ganz höchster Ebene? Also wo Sie sagen, da kann man nicht daran rütteln.

28 B: Okay, nein. Wir sind da sehr, da wir die Abteilung sind die dafür zuständig ist, sind wir da auch sehr frei zu entscheiden. Wie zum Beispiel beim Ingress Controller, was wir kürzlich gemacht haben.

29 I: Darf ich fragen was Sie da verwenden?

30 B: Kong jetzt.

31 I: Kong? Okay. Und, gute Erfahrungen damit gemacht?

..Technologiestack

32 B: Zwei Sachen eigentlich, Kong und den NGINX Ingress. Ja, passt schon. Der hat einige Funktionalitäten die der reine NGINX Ingress nicht hat, und Traefik haben wir auch teilweise im Einsatz. Grundsätzlich wenn wir nichts, aus meiner Erfahrung bisher, wenn wir nichts von den Spezialfunktionen die die Ingress Controller haben verwenden, funktionieren die alle sehr gut.

33 I: Darf ich fragen, für... war Ihr Produkt eigentlich macht? Also welche Kunden Sie bedienen? Um dann gleich in die Folgefrage zu gehen, ob es vielleicht aus Ihrer Kundensicht Anforderungen gibt? Regulatorischer Natur, rechtliche Dinge die sie erfüllen müssen, solche Sachen, die Sie im Hinterkopf behalten sollten?

34 B: Also was war jetzt die erste Frage? Entschuldigung, ich bin heute schon etwas müde.

35 I: Was wirklich Ihr Workload ist, also Ihr Produkt, und welche Kunden Sie bedienen?

36 B: Das Hauptprodukt ist eine Software mit der man komplex Verrechnung machen kann würde ich sagen. Wobei ich die

		erdenklich schlechteste Person bin um genau zu sagen was unsere Software macht. Ich habe mit der Software ganz ganz wenig zu tun.
	37	I: Okay.
..Art der Workload	38	B: Also so eine Art Serviceportl für Kunden, Verrechnungen und viel für Telekommunikationsunternehmen.
	39	I: Aber die Software läuft grundsätzlich bei Ihnen und wird grunsätzlich als Software as a Service vom Kunden bezogen?
..[AR02] Availability/Lokation	40	B: Sowohl als auch. Das gibt es als SaaS Angebot oder auch on-premises beim Kunden.
	41	I: Wenn Sie es zum Kunden liefern, liefern Sie es dann auch auf Kubernetes Stack oder wie gehen Sie damit um?
	42	B: Ab der neuesten Version ja.
	43	I: Das heißt Sie setzen eine Umgebung voraus, oder stellen Sie dem Kunden wirklich den Cluster hin?
..Kubernetes beim Kunden	44	B: Nein wir setzen es voraus. Wir liefern aber durchaus auch die Anleitung wie man es selber auf AWS machen kann.
	45	I: Okay. Darf ich fragen wie Sie das formulieren? Ist das irgend ein OpenShift wo Sie sagen das muss ein zertifizierter Cluster sein, oder einfach eine bestimmte Upstream version?
..Kubernetes beim Kunden	46	B: Das muss nur Kubernetes Cluster sein, zwischen 1.14 und 1.19.
	47	I: Also wenn ich soweit mitnehmen, andere Rahmenbedingungen in dem Sinne gibt es jetzt nicht. Weil Sie gesagt haben, Sie haben viele Kummunikationsunternehmen als Kunden, gibt es da vielleicht bzgl Lokation oder Standort irgendwelche hot Topics? Wo Sie sagen müssen ein Cloud provider zum Beispiel muss in diesem Land verfügbar sein. Oder ist das auch relativierbar?
	48	B: Nicht, dass ich wüsste.
	49	I: Okay.
	50	B: Aber ich muss auch sagen, ich bin wirklich jetzt Gott sei Dank, ganz auf technischer Seite. Das heißt vielleicht weiß ich manches auch einfach nicht. Aber nicht, dass ich wüsste, nein.
	51	I: Ja gut, dann sind wir schon beim nächsten Block der funktionalen Anforderungen.
..[AR02] Availability/Lokation	52	B: Wenn jemand ganz unzufrieden wäre, mit der Lokation die wir haben in unserem AWS SaaS, dann kann er sich ja selber wo auch immer er will seinen Service aufsetzen.
	53	I: Richtung funktionale Anforderungen. Weil Sie gesagt haben, es ist nur eine Version zwischen 1.14 und 1.19 nötig, das ist ja doch recht breit. Nehme ich das richtig an, dass Sie keinene großartigen

funktionalen Anforderungen haben. Zum Beispiel in Richtung Persistenz oder irgendwelche hybriden.. also mehrere Betriebssysteme, GPU Support. Also irgendwelche extravaganten technischen Anforderungen.

..[AF15] Persistenter Speicher

54 B: Gar nicht, nein. Und auch keine Persistenz. Persistenz brauchen wir nur in Testumgebungen, weil wir in den live Umgebungen eigentlich die AWS mit ihren ganzen Datebanken verwenden.

55 I: Wirklich Datenbanken oder auch so sachen wie File Storage, Block storage in der AWS

..[AF15] Persistenter Speicher

56 B: Das brauchen wir nicht, weil in der AWS.. also in Testumgebungen ja, weil in Testumgebungen die Datenbanken Cache usw. auch als Pod gestartet wird, oder halt als Container. Aber in der live Umgebung, in Produktivumgebungen wird einfach der managed Service für Datenbanken verwendet. Daher wir keine Persistenz außer in Testumgebungen benötigt.

57 I: Okay, das heißt Sie haben, gehe ich mal davon aus, alles auf Linux Containern laufen? Also auch kein OS Thema?

..[AF05] Betriebssystem für Container
..[AF04] GPU Support

58 B: Nein, Windows Container gibt es gar nicht. Und GPU Beschleunigung oder so, das sind Dinge die wir nicht brauchen.

59 I: Wie schaut es aus in Richtung Skalierung? Verwenden Sie einen horizontal Pod Autoscaler oder sowas in die Richtung? Sollte mehr Last anfallen.

60 B: Ja, nein.. tatsächlich noch nicht.

61 I: Das heißt man macht das dann schlimmstenfalls einfach händisch.

..HPA/Skalierung

62 B: Ja genau. Das wäre kein Problem, wir können es jederzeit hoch oder runter skalieren, aber der HPA wie er in den älteren Versionen war hat sich mit unseren Containern nicht gut vertragen. Weil die recht lange startup Phase haben bei der sie viel CPU benötigen. Das heißt, dass wenn man das vor den HPA hinsetzt, dann startet und startet und startet.

63 I: Das Problem kenne wir.

64 B: Auch Spring oder?

65 I: Nein Spring, wir haben eine Machine Learning Anforderung gehabt einmal und da war auch das Problem, dass das anfangs noch ziemlich holprig war. Also vor einem Jahr haben wir das mal probiert, das hat leider überhaupt nicht funktioniert.

66 B: Aber es gibt mittlerweile bessere optionen mit neuere Kubernetes Versionen. Da kann man ihm schon mehr Sachen lernen. Also, dass er einfach auch eine längere wartephase hat bevor er weiter skaliert, damit lässt es sich besiegen. Also theoretisch ginge es, aber wir brauchen es im Moment nicht dringend, weil unsere Last relativ konstant ist.

67 I: Das heißt Sie haben wirklich jetzt nur diese Java Anwendung, oder gibt es da noch mehrere Microservices oder mehrere Teile dieser Applikation die man aus funktionaler Sicht noch betrachten sollte?

..Art der Workload

68 B: Also es sind Angular Webfrontend, aber das liefert ja eigentlich nur ein statisches aus. Und natürlich die Java Spring APIs.

69 I: Das klingt nahezu angenehm.

70 B: Ja woll. Von dem her ist es recht homogen.. das ist vom Verwalten her ganz angenehm.

71 I: Darf ich grundsätzlich noch fragen, wissen Sie warum AWS als cloud gesetzt wurde?

..[AR10] Verwendung der public cloud

72 B: Weil es einfach schon lange dort läuft und... also schon lange in der cloud generell ist, und angefangen wurde mit der Fujitsu cloud und das hat gar nicht gut funktioniert. Und dann ist halt zum Zeitpunkt wo AWS die stärkste cloud wurde zu AWS gewechselt worden.

73 I: OK, das heißt Sie als Softwareprovider haben von vorne herein gesagt: "Wir fahren cloud!"

..[AR10] Verwendung der public cloud

74 B: Also es war recht früh die Motivation in die cloud zu gehen, aber das erzähle ich jetzt wirklich aus zweiter Hand, da es vor mir war.

75 I: Ja gut, das ist auf jeden Fall ein Statement. Also wie gesagt, in den letzten Interviews die ich gehabt habe war halt oft die Aussage, sehr skeptisch gegen die Cloud oder eben nur on-premises.

..[AQ04] Betriebspersonal Aufwand

..OpenShift kompatibel

76 B: Also ich sage auch, wenn man nicht ein irrsinnig gutes Adminteam hat, macht es aus meiner Sicht Sinn die Cloud zu verwenden. Weil man braucht schon recht viel know how, damit das gescheid läuft. Wobei wir auch imstande dazu sind, weil wir durchaus Kubernetes auch im OpenStack haben, aber... wir versuchen auch die Testumgebungen langsam nach AWS zu bringen. Weil es einfach komfortabler ist. Ach ja, und was mir von funktionalen Anforderungen.. das habe ich vergessen vorher.. mittlerweile.. wir müssen nicht OpenShift verwenden, aber es muss auf OpenShift funktionieren. Haben Sie schon mal mit OpenShift gearbeitet? Weil das hat einige Einschränkungen, die uns einige Zeit gekostet haben.

77 I: Ja, also wir hatten das Problem mit den relativ energischen Security Einschränkungen.

78 B: Genau, das gleiche war auch bei uns.

79 I: Jetzt muss ich nur nochmal kurz nachfragen, wie oder welches tool verwenden Sie um dann auf AWS zu provisionieren? Also kops zum Beispiel, oder machen Sie das von Hand?

80 B: Terraform.

	81	I: Für die Installation vom Cluster, oder für die Provisionierung der Nodes im OpenStack?
	82	B: Sowohl als auch. Also nicht die Installation der Software am Cluster, aber der Kubernetes selbst.
	83	I: Das heißt es gibt wirklich Terraform Module, wo ich zum Beispiel das Kubelet installieren kann? Die ganze Docker Engine etc.
..Technologiestack	84	B: Das machen wir nicht, wir verwenden EKS, das ist das Elastic Kubernetes Service. Das ist managed Kubernetes. Also eine managed control plane. Und da des eine der Standardmodule im Terraform ist, kann man relativ leicht einen EKS Cluster provisionieren und dann braucht man nur noch einen Worker anstarten, der auf AWS... auf AWS das Amazon Linux hat schon ein Script drinnen, mit dem man einen EKS Cluster als Worker joinen kann.
	85	I: Aha, das heißt Sie provisionieren sich die control plane und dann bauen Sie die Nodes mehr oder weniger mit dem Basisimage im OpenStack und joinen die dann zu EKS cluster?
	86	B: Nein nicht OpenStack. EC2 alles im AWS
	87	I: Achso, okay
..Technologiestack	88	B: Das ist einfach, im AWS werden EC2 Intanzen gestartet und die joinen als Worker die EKS control plane.
	89	I: Dann habe ich das jetzt falsch verstanden. Dann haben Sie OpenStack on-premises?
..Technologiestack	90	B: Genau, das ist on-premises. Das ist unsere Testumgebung, die nicht am eigenen Laptop laufen sondern am OpenStack. Die Installation der Software dann auf dem Kubernetes Cluster ist mit HELM.
	91	I: Ja gut, das macht Sinn.
..Technologiestack	92	B: Und wird jetzt wahrscheinlich dann mit ArgoCD laufen.
	93	I: Das ist zu empfehlen ja, also wir haben das schon einmal evaluiert und probieren das ausrollen, das funktioniert sehr sehr gut. Das ist ein cooles Projekt.
	94	B: Ja also ich verwende es, also ich baue gerade einen Prototypen dafür. Verwenden Sie schon ArgoCD?
	95	I: Teilweise ja. Also grundsätzlich sind wir sehr Microsoft getrieben, also wir verwenden Azure DevOps Server on-premises für unsere ganzen pipelines. Aber wir haben schon mal probiert mit ArgoCD und da vor allem aber die Webhook Funktionalitäten. Also wir provisionieren zum Beispiel Build Agenten dynamisch, und das machen wir mit ArgoCD Hooks. Das ist einfach ein listener und der startet dann eine mini pipeline die dann diesen Build Agenten zur Verfügung stellt und zum Cluster joined und dann dem Azure

DevOps Server zur Verfügung stellt.

96 B: Dass der eine build Agenten im lokalen Cluster hat?

97 I: Genau. Und dann wenn der build vorbei ist wird er wieder niedergefahren und gibt die Ressourcen frei.

98 B: Das könnte man mit Tekton auch ganz gut machen schätze ich.

99 I: Genau, das wäre im Prinzip das gleiche in grün. Also wir verwenden halt kein Tekton pipeline, sondern halt die Azure DevOps pipeline.

100 B: Und funktioniert das gut?

101 I: Ja, das funktioniert super, also das ist schon echt ein stabiler Stack.

102 B: Coole Sache. Also ich habe Azure DevOps nur ein bisschen ausprobiert. Ich habe zwei Bachelorarbeiten schreiben müssen. Für die erste habe ich es ein bisschen verwendet, aber nicht mit Kubernetes.

103 I: Nein das funktioniert super. Machen wir bitte noch schnell das letzte Kapitel, und zwar Qualitätseigenschaften. Da geht es darum, wenn Sie jetzt sagen Sie müssten zwei single node Distributionen, K3s und MicroK8s, vergleichen.. auf welche Eigenschaften würden Sie persönlich, aber auch das Unternehmen, Wert legen? Also sowas wie natürlich Preis, ganz klar...

..[AQ03] Preis/Lizenzkosten

104 B: Also Preis ist fast irrelevant, weil eh fast alles open-source und gratis ist.

105 I: Ja aber generell, also auch in Richtung andere Produkte gedacht. Also welche Qualitätseigenschaften würden Sie für den Vergleich von Produkten heranziehen?

..Usability des Tools [AQ19]

..[AQ03] Preis/Lizenzkosten

..WSL kompatibel auf Windows

106 B: Also mir wäre extrem wichtig, dass war auch sicherlich wo wir überlegt haben was die Sachen war, warum wir dann wirklich Docker Desktop und MicroK8s verwenden. Also ich würde sagen auf jeden Fall, dass es einfach simple zu verwenden ist, für den Entwickler. Also da geht es jetzt wirklich um die lokalen Installationen für die Entwickler. Aber es muss einfach zu verwenden sein, es muss so stabil wie möglich sein. Also wenn ein drittel der Entwickler damit beschäftigt ist, dass die Entwicklungsumgebung nicht geht, dann macht man sich keine Freunde. Ich meine Preis.. es sollte schon gratis sein, das ist in dem Zusammenhang aber meist auch gar nicht so schwer. Was war denn noch so wirklich wichtig? Was auf Windows sicherlich ein größeres Thema war, was, dass das Ganze WSL kompatibel ist. Damit man nicht extra noch eine Virtualbox oder so installieren muss. Was war denn sonst noch so Diskussionspunkt? Tatsächlich für die lokale Umgebung ein großes Thema ist VPN und lokales Kubernetes. Weil in dem Moment wo das lokale Kubernetes in einer extra VM läuft, hast du ein Problem mit VPN Verbindungen. Ja ich glaube das waren so die wesentlichsten Dinge die wir gehabt haben.

107 I: Gut, das ist eigentlich relativ vollständig hätte ich gesagt. Also grundsätzlich wenn Ihnen zu den vorherigen Punkte nichts mehr einfällt, würde ich mich damit bedanken. Vielen Dank für Ihre Zeit.

A.28 Kodierleitfaden

Liste der Codes	Memo	Häufigkeit
Codesystem		378
Rechtfertigung der Masterarbeit	Unterstreicht die Aussage, dass eine strukturierte Vorgehensweise nötig ist. --- Bsp: Ja es gibt immer wieder sehr spezielle Fälle, eben auch von Kundenseite. Aber das.. da kommt man dann in der Implementierung drauf, darum ist auch für mich das immer ganz wichtig das im vorhinein abzuchecken. Wie schon erwähnt, irgendwelche Sonderlösungen dann im Nachgang zu implementieren das verursacht dann ein vielfaches an höheren Aufwand, im Vergleich zur initialen Installation.	5
Unternehmen		97
Vorschlag für Kubernetes	Woher kam der initiale Vorschlag Kubernetes zu verwenden? --- Bsp: I: Das heißt, man kann also plakativ sagen es kommt definitiv aus der Entwickler Riege und nicht vom Management, ganz salopp formuliert. B: Ja auf jeden Fall, also ganz ehrlich wir haben heute noch den Punkt, dass es der Großteil der Mitarbeiter im Unternehmen mit Digitalisierung nichts anfangen können und sagen, wozu brauchen wird das.	11
Kubernetes Architektur	Wo wird K8s betrieben? Wie ist die Architektur? --- Bsp: Das gibt es als SaaS Angebot oder auch on-premises beim Kunden.	9
Art der Workload	Was wird am Kubernetes betrieben? --- Bsp: Also es sind Angular Webfrontend, aber das liefert ja eigentlich nur ein statisches aus. Und natürlich die Java Spring APIs	12
Entscheidung für Kubernetes	Wer hat entschieden, dass Kubernetes verwendet wird? --- Bsp: Also monetäre Entscheidung obliegt da sowieso meist dem Management. Da haben wir dann als Techniker in dem Sinn wenig Mitspracherecht.	13
Motivation für Kubernetes	Warum wurde Kubernetes angedacht? Was war die Motivation? --- Bsp: Aber die Motivation war sicher einfach weil vorher das auch schon in Containern gelaufen ist. Und die Orchestrierung halt mit Kubernetes einfacher ist, wenn es mal rennt.	10

Liste der Codes	Memo	Häufigkeit
Dauer Kubernetes Verwendung	Wie lange wird Kubernetes schon verwendet? --- Bsp: Service Mesh ist auch in der Evaluierung weil... also die BSD2 APIs das war der Anfang, damit haben wir ungefähr vor zwei Jahren angefangen.	6
Technologiestack	Welche Technologien werden im Kubernetes Umfeld verwendent? --- Bsp: Genau. Also Spring und nodeJS ist das im Großen und Ganzen. Ähm.. nicht nodeJS, Spring und Angular 1 ist es im Großen und Ganzen. Und wir verwenden Kubernetes auf OpenStack, auf AWS und lokal.	35
Interviewpartner		6
Stellenbeschreibung	Wofür ist der Interviewpartner zuständig? --- Bsp: Mein Name ist Y, bin beim Unternehmen X IT-Systems Engineer. Bin dort hauptsächlich zuständig für die Produkte aus dem RedHat Portfolio, also RHEL, mit dem Satellite als Management. Aber mein Hauptfokus aktuell liegt eben beim Design und beim Betrieb von Container Orchestrierungsplatformen. Und alles andere was sich so um das Thema containierisierung und DevOps handelt.	6
Out-of-Scope		19
OpenShift kompatibel		2
Solution neu aufbauen wenn Tests fehlschlagen		1
Testing von Images		1
Mehrwert begreifbar machen		1
Mandantenfähigkeit		3
Out-of-the-Box Funktionalität		1
technische Beschreibung von Infrastruktur		1
Softwarepacketierung		1
Deployment zum Kunden		2
Lizenzcheck im Container		2
Antivirus		4
Qualitätseigenschaften		0

Liste der Codes	Memo	Häufigkeit
[AQ02] Verfügbarkeit in %	Ist die Verfügbarkeit, sprich Uptime, eine relevante Kenngröße? --- Bsp: Ja wir müssen im Zuge von unserem SLA Management KPIs angeben, so etwas wie Verfügbarkeit oder maximale Ausfallszeit oder Anzahl der Ausfälle pro Monat. Da haben wir was wenn es in die Richtung geht. Also wir sagen zum Beispiel wir haben uns mal an die Verfügbarkeitsanforderungen unserer wichtigsten Applikation.. also das ist das BSD2.. gehängt und da haben wir eine Verfügbarkeitsanforderung von 99 Prozent. Das heißt 7.3 Stunden Ausfalls insgesamt pro Monat ungefähr, maximale Ausfallszeit pro Einzelfall sieben Stunde und maximale Anzahl Ausfälle pro Monat zwei. Das sind die Verfügbarkeits KPIs die wir momentan haben, die wir erfüllen müssen.	12
[AQ03] Preis/Lizenzkosten	Sind die Kosten für das Produkt an sich relevant? --- Bsp: Also die setzen dort keinen Rancher ein, die seitzen OpenShift ein im Unternehmen X. Oder wollen zumindest OpenShift einsetzen, daher ist es auch ein gewisser Kostenpunkt der diskutiert werden muss.	14
[AQ04] Betriebspersonal Aufwand	Ist es wichtig wie viele MA für den Betrieb beschäftigt sind? Gibt es schon eine verfügbare Betriebsmannschaft/Ops Team? --- Bsp: Sagen wir so.. Also am Ende des Tages.. einige Punkte wie Installation, Aufwand etc. mündet am Ende des Tages wieder in Kosten bzw. Personalbedarf. Das ist meistens in vielen Firmen ein.. also Personalbedarf.. das ist in vielen Firmen ein kritischer Punkt. Personal sind immer Kosten.	9
[AQ05] Marktanteil	Spielt der Marktanteil einer Distribution/eines Produktes eine Rolle? --- Bsp: Marktanteil.. ja also ich glaube am Ende wird es bei vielen auf den Kosten hängen bleiben. Das was ich gelernt habe.. bitte nicht immer dem Gartner Quadrant vertrauen.	1
[AQ06] Communitygröße und -qualität	Sind Größe und Qualität der Community ein wichtiger Faktor? --- Bsp: Größe und Qualität der Community.. ja das ist sicherlich nicht zu unterschätzen, würde ich sagen. Ich glaube es gibt gute Software die jetzt nicht so ein große Community hat. Wenn ich mir zum Beispiel Proxmox anschaue, als österreichische Lösung. Dann ist das wahrscheinlich Software die gut ist und viel kann, die aber wahrscheinlich international gar nicht so bekannt ist.	5

Liste der Codes	Memo	Häufigkeit
[AQ08] Durchlaufzeit/Aufwand für Installation	Ist die Zeit, die für die Installation des Clusters/ Produktes benötigt wird relevant? --- Bsp: Also in der Vergangenheit war immer der Zeitfaktor essenziell. Das heißt da hat es schnell gehen müssen, also "wie kommen wir am schnellsten zum Ergebnis?". Für das was wir in Zukunft möchten, dass wir das in unseren Servicekatalog integrieren möchten, da.. so richtig Vorgaben haben	7
[AQ09] Schulung Admins	Ist der Schulungsaufwand für das Admin-/Ops-Team ein relevanter Faktor? --- Bsp: Oder auch die.. das was man auch nicht unterschätzen darf bei dem ganzen Personalbedarf sind auch Schulungen. Wenn ich mir zum Beispiel ein OpenShift anschaue, wenn ich das wirklich gut betreuen will oder auch regelmäßig betreuen will, brauche ich ein paar Leute die das, meiner Meinung nach, Tag ein Tag aus machen. Die sich damit auskennen! Generell braucht man, wenn man sowas sieben mal 24 anbietet, dann braucht man mal mindestens vier Leute damit man das machen kann, gesetzlich in Österreich.	9
Qualifiziertes Personal [AQ11]	Wie viele MA habe ich schon, die qualifiziert sind? Habe ich schon Mitarbeiter die sich mit der Technologie auskennen? --- Bsp: Ja evaluiert hat das eigentlich das Projekt X auch zusammen mit einem Externen der... also ich glaube der bringt da glaube ich einfach für Istio ziemlich viel Erfahrung mit und der sagt.. das ist nicht die schlechteste Wahl.	9
Unternehmensgröße [AQ12]	Spielt die Größe eines Unternehmens eine Rolle? --- Bsp: Also wenn man sich die Hersteller anschaut die es gibt, es gibt.. RedHat, es gibt SUSE, die mittlerweile nicht mehr so groß ist, zumindest in Europa kommt mir das so vor. Dann gibt es doch.. die amerikanischen Firmen die den Markt recht gut in der Hand haben, wenn es um Software geht.	1
Benchmarks [AQ13]	Sind offizielle Benchmarks als Qualitätsmerkmal interessant? --- Bsp: Das was ich gelernt habe.. bitte nicht immer dem Gartner Quadrant vertrauen. Also auch wenn Produkte hoch geranked sind müssen sie nicht gut sein. Haben wir selbst beim backup tool am eigenen Leibe erfahren. Dass dies einfach ein bisschen Lobby Arbeit ist, was im Gartner Quadranten ist.	2

Liste der Codes	Memo	Häufigkeit
State of the Art [AQ14]	Ist die "Aktualität" einer Technologie wichtig? Würde man sich heute für eine neuere Technologie entscheiden? --- Bsp: Zu dem Zeitpunkt hat es den EKS Service auch noch nicht gegeben in der Amazon Cloud. Sonst hätten wir den verwendet, der war da gerade irgendwo in der Ankündigung.	2
Dokumentation [AQ15]	Ist die Qualität der Dokumentation ausschalggebend? --- Bsp: Messgröße: 1-5; selbst definieren; in PoC herausfinden (Qualität, Umfang, Aktualität, Anwendbarkeit,..)	2
Support-Responsezeit auf Fehler [AQ16]	Ist es wichtig wie schnell vom Support auf Tickets reagiert wird? --- Bsp: Support-Responsezeit auf Fehler Messgröße: Antwortzeit des Supportteams in Stunden	2
Ausgereiftheit [AQ17]	Ist die bisherige Lebenszeit/Ausgereiftheit des Produktes relevant? --- Bsp: Also einerseits sollte meiner Meinung nach das Produkt schon eine gewisse Roadmap hinter sich haben. Also Version 1.0 ist eventuell nicht immer der optimal Zeitpunkt um irgendwo einzusteigen.	2
Nutzerkreis/bestehende kunden [AQ18]	Ist es wichtig welche Unternehmen eine bestimmte Software bereits nutzen? --- Bsp.:	1
Usability des Tools [AQ19]	Ist die Bedienbarkeit eines Tools relevant? --- Bsp: Also mir wäre extrem wichtig, dass war auch sicherlich wo wir überlegt haben was die Sachen war, warum wir dann wirklich Docker Desktop und MicroK8s verwenden. Also ich würde sagen auf jeden Fall, dass es einfach simple zu verwenden ist, für den Entwickler. Also da geht es jetzt wirklich um die lokalen Installationen für die Entwickler. Aber es muss einfach zu verwenden sein, es muss so stabil wie möglich sein. Also wenn ein drittel der Entwickler damit beschäftigt ist, dass die Entwicklungsumgebung nicht geht, dann macht man sich keine Freunde.	5

Liste der Codes	Memo	Häufigkeit
Versionsmanagement + Host/K8s Hardening [AQ20]	Muss das Hardening der Kubernetes Hosts als Faktor berücksichtigt werden? --- Bsp: Ja also was wir mittlerweile haben, das hat man am Anfang nich so sehr gesehen, sind eben diese ganzen Security Richtlinien. Also wenns es gerade um Schwachstellenmanagement geht, um hardening… dann müssen wir auf das sehr sehr stark schauen	5
Häufigkeit von Updates/neuen Releases [AQ21]	Ist es wichtig, dass regelmäßig und oft neue Releases erscheinen? --- Bsp:	1
mögliche Granularität der Berechtigungen [AQ22]	Ist es relevant zu beachten, wie ganular Berechtigungen in einer Software vergeben werden können? --- Bsp.:	1
Auswertungs-/ Überwachungsmöglichkeiten [AQ23]	Ist es relevant wie viele Möglichkeiten eine Software bietet um Systeminformationen aufbereitet darzustellen? --- Bsp.:	1
Ressourcenverbrauch [AQ24]	Ist es wichtig wie gut die bestehende Infrastruktur ausgelastet wird? --- Bsp: Da fahren wir dann diese 40 EC2 Instanzen hoch, die sind recht groß.. die haben 64 Kerne.. rechnen dann unsere Docker Container parallel so viele sich halt ausgehen. Das tun wir immer so ein bisschen händisch, das heißt da haben wir kein autoscaling sonder wir schauen uns an wie viel RAM braucht ein Container und wie viel können wir parallel ausführen. Dann rechnen wir das durch und schalten ihn manuell wieder aus. Das ist unser Use-Case.	2
Funktionale Anforderungen		73
[AF04] GPU Support	Wird GPU Unterstützung benötigt? --- Bsp: GPU Support ja, also GPU Support kommt immer wieder. Es gibt aber unterschiedliche Auffassungen ob GPUs gut oder schlecht sind. Also ja, es gibt use-cases wo es definitiv gut ist, gerade Machine Learning oder AI. Es gibt aber auch sehr wohl Kunden die sagen sie rechnen lieber auf einer CPU, oder auf mehreren CPUs weil es für sie schneller und günstiger ist. Also da muss man immer sehr genau wissen, seinen business case dahinter.	6
[AF05] Betriebssystem für Container	Welche Betriebssysteme werden für die Applikation im Container benötigt? --- Bsp: Ja, das waren immer Linux Container.	6

Liste der Codes	Memo	Häufigkeit
[AF07] Pod2Pod Kommunikation	Muss die Kommunikation zwischen den Pods eingeschränkt werden können? --- Bsp: Richtung Netzwerk oder Mesh, ja Vorgabe, ist jeder Namespace hat default mäßig eine deny all policy, Kann also nicht mit anderen Pods kommunizieren. Wenn man das braucht muss man es explizit freigeben.. also wirklich Microsegmentierung auf Netzwerk Ebene und jede Kommunikation muss extra freigegeben werden.	6
[AF08] Metriken	Wird ein out-of-the-box Metrik-System benötigt? Bsp: Nein also... das was alle Kunden haben wollen, die wollen halt intern ein Reporting bzw. Monitoring dazu haben. Das ist meiner Meinung nach die Grundlage davon, damit ich sowas überhaupt betreuen kann.	4
[AF09] Logging	Wird ein out-of-the-box Logging System benötigt? Bsp: Wir haben Splunk, wir nehmen auch fluentD Agent und halt Splunk connect for Kubernetes um die ganzen Daten in Splunk zu bringen. Und wir haben uns eine eigene CD pipeline mit gitlab CI und Argo CD als GitOps Agent.	2
[AF12] Layer 4 / Load Balancer	Werden Layer4, bzw. nicht-Layer-7, Zugriffsmöglichkeiten benötigt? --- Bsp: Layer vier basiertes Routing ist ja im.. Layer sieben Routing ist ja im OpenShift integriert, aber da muss man dann schauen wie bindet man Lösungen an die zum Beispiel UDP verwenden. MQTT Protokolle oder reine TCP Connections, diese Anforderungen sind schon auch zu prüfen. Welche Anforderungen bestehen auch von der Client Seite. Also es werden unterstützt WebSockets und Layer vier (sieben) basierte Routings mit SNI und TLS, das ist sehr überschaubar, macht aber meines Erachtens mehr als 90% des traffics aus. Sollte man jetzt irgendwie nach Pareto noch diese 20% integrieren müssen, macht das eben diesen 80%igen Aufwand aus, für solche Speziallösungen dann irgendwelche Integrationen zu bauen.	7

Liste der Codes	Memo	Häufigkeit
[AF13] Service Mesh	Gibt es spezielle Anforderungen in Richtung LoadBalancing oder Resilienz? --- Bsp: Richtung Netzwerk oder Mesh, ja Vorgabe, ist jeder Namespace hat default mäßig eine deny all policy, Kann also nicht mit anderen Pods kommunizieren. Wenn man das braucht muss man es explizit freigeben.. also wirklich Microsegmentierung auf Netzwerk Ebene und jede Kommunikation muss extra freigegeben werden. Service Mesh ist auch in der Evaluierung weil... also die BSD2 APIs das war der Anfang, damit haben wir ungefähr vor zwei Jahren angefangen. Mittlerweile sind auch noch andere APIs oben und man will das Produkt X neu... das sind genau so Microservices.. auf Kubernetes bringen. Und da überlegt man sich mit Hilfe von Istio... resilience beziehungsweise Verschlüsselung, Kommunikationsverschlüsselung auszulagern in einen Service Mesh.	2
[AF14] RBAC Integration	Wird eine out-of-the-box Lösung zur Verbindung mit einem Usermanagement System benötigt, um RBAC zu ermöglichen? --- Bsp: Und die haben halt den need, dass es für sie halt einfacher geht. Deshalb Rancher, also gerade was Userverwaltung angeht, [..], backup und restore. Oder wenn wirklich die Instanzen wachsen müssen, dann ist das dort drin halt leichter zu machen, als wenn man sich das wirklich von scratch an lernen muss.	2
[AF15] Persistenter Speicher	Wird vom Cluster verlangt persistenten Speicher zur Verfügung zu stellen? --- Bsp: Diese Anfroderung kommt eben.. auf Grund von Design aus der Plattform selbst heraus. Dass eben diese Metrikdaten und Loggingdaten mit dem EFK Stack.. jetzt wenn man redet von OpenShift.. einen Blockstorage brauchen. Also das ist aktuell die einzige Anforderung die wir an Storage haben, von der Plattform selbst. Von Kundenseite aktuell eher weniger die Anforderung persistent storage verwenden zu müssen	13
[AF02] data-at-rest Encryption	Müssen persistente Daten verschlüsselt werden? --- Bsp: I: Data-at-rest oder Netzwerk Verschlüsselung? B: Beides. Sagen wir so, Netzwerkverschlüsselung ist mittlerweile Standard. Also, dass die Daten zwischen den Servern verschlüsselt herumgehen müssen ist fast Standard.	4
[AF17] File Storage	Wird explizit File Storage benötigt? --- Bsp.:	1

Liste der Codes	Memo	Häufigkeit
[AF19] Block Storage	Wird explizit Block Storage benötigt? --- Bsp.: Diese Anfroderung kommt eben.. auf Grund von Design aus der Plattform selbst heraus. Dass eben diese Metrikdaten und Loggingdaten mit dem EFK Stack.. jetzt wenn man redet von OpenShift.. einen Blockstorage brauchen. Also das ist aktuell die einzige Anforderung die wir an Storage haben, von der Plattform selbst. Von Kundenseite aktuell eher weniger die Anforderung persistent storage verwenden zu müssen	2
shared Filesystem	Muss der Cluster Filesysteme zur Verfügung stellen können, die von mehreren Pods gleichzeitig genutzt werden können? --- Bsp: Shared Filesystem - dynamische Bereitstellung von gemeinsam verwendbaren shares	1
verschlüsselte Kommunikation [AF21]	Muss die Kommunikation im Cluster verschlüsselt werden? --- Bsp: Genau richtig, also solche Sachen. Also gerade Istio ist ist natürlich ein Schlagwort in die Richtung auf der RedHat Seite, was einiges macht. Google Anthos geht auch in diese Richtung. Also das ist de facto.. Sagen wir so.. Also wenn man Kubernetes betreibt, oder wenn man diese Service Architekturen betreibt hat man im Endeffekt [..], dann ist auf jeden Fall alles verschlüsselt.	4
backup/restore out-of-the-box [AF22]	Liegt eine backup/restore Funktionalität out-of-the-box vor? --- Bsp: Und die haben halt den need, dass es für sie halt einfacher geht. Deshalb Rancher, also gerade was Userverwaltung angeht, [..], backup und restore. Oder wenn wirklich die Instanzen wachsen müssen, dann ist das dort drin halt leichter zu machen, als wenn man sich das wirklich von scratch an lernen muss.	3
Integration in bestehende Infrastruktur [AF23]	Muss das neue System in eine bestehende Infrastruktur integrierbar sein? --- Bsp: Im Unternehmen X, also das Unternehmen ist ja ein großer Cisco Kunde, die machen.. wir wollen Cisco ACI einsetzen für das. Wo es dann eben ein OpenShift CNI Plugin gibt.	6
Sprache der Dokumentation [AF24]	Muss die Dokumentation in einer bestimmten Sprache vorliegen?	1
Ingress WAV [AF25]	Muss der Ingress Controller tiefergehende WAV Funktionalitäten bieten? --- Bsp: WAV Features für Ingress - erweiterte Security Features für Ingress Controller	1

Liste der Codes	Memo	Häufigkeit
alte TLS Versionen	Gibt es Anforderungen alte TLS Versionen zu unterstützen? --- Bsp.: Und wir haben auch das Thema gehabt... Es ist eh nicht zu verheimlichen, dass eben diese IoT Geräte ja sehr lange im Feld draußen sind. Nachdem ja heuer im Frühjahr TLS 1.0 und 1.1 abgedreht worden sind.. und eben die Mozilla Foundation eben dann nach Ende des Corona Lockdown eins diese... das auch durchgesetzt hat, haben dann einige Geräte draußen das Problem gehabt mit TLS und Cipher Suites. Wir haben dann eben für diese Anwendung dann so eine Art legacy Proxy auf unserem externen Load Balancer integriert. Der eben noch für diese alten TLS Version Unterstützung liefert. In OpenShift ist denn eben dieser TLS Support mitte des Jahres auch heraus gefallen. Da werden nur mehr die aktuellste TLS Versionen 1.2 und 1.3 supportet.	2
HPA/Skalierung	Müssen Workloads am Cluster automatisch skaliert werden? --- Bsp.: Ja genau. Das wäre kein Problem, wir können es jederzeit hoch oder runter skalieren, aber der HPA wie er in den älteren Versionen war hat sich mit unseren Containern nicht gut vertragen. Weil die recht lange startup Phase haben bei der sie viel CPU benötigen. Das heißt, dass wenn man das vor den HPA hinsetzt, dann startet und startet und startet	5
K8s Policies	Braucht es eine out-of-the-box Lösung um policies verwenden zu können?	1
Tracing	Sind Tracing Funktionalitäten am Cluster nötig? --- Bsp.:	1
WSL kompatibel auf Windows		1
Rahmenbedingungen		81
[AR01] Open-Source Lizenzen	Ist open-source verwendbar? Gibt es gewisse Lizenzen die nicht verwendet werden dürfen? --- Bsp: Wobei ich auch sagen muss... weil du auch als ersten Punkt open-source gehabt hast... das hat open-source nicht ausgeschlossen. Also zum Beispiel Argo CD wenn die Community groß genug ist, wenn die irgendwie zeigen können, dass sie sich um Security kümmern und wir intern über unsere Image Prüfungen da auch sehen dass das passt, dann funktioniert sowas auch. Und wenn die Lizenz passt. Also das kann ich vielleicht auch vorweg nehmen, wir hatten dadurch dass dies Apache 2.0 Lizenz war hier auch nie Probleme mit unsren Legal Leuten. Und sonst eigentlich auch keine Geschichten bei denen ich sage es geht gar nicht.	11

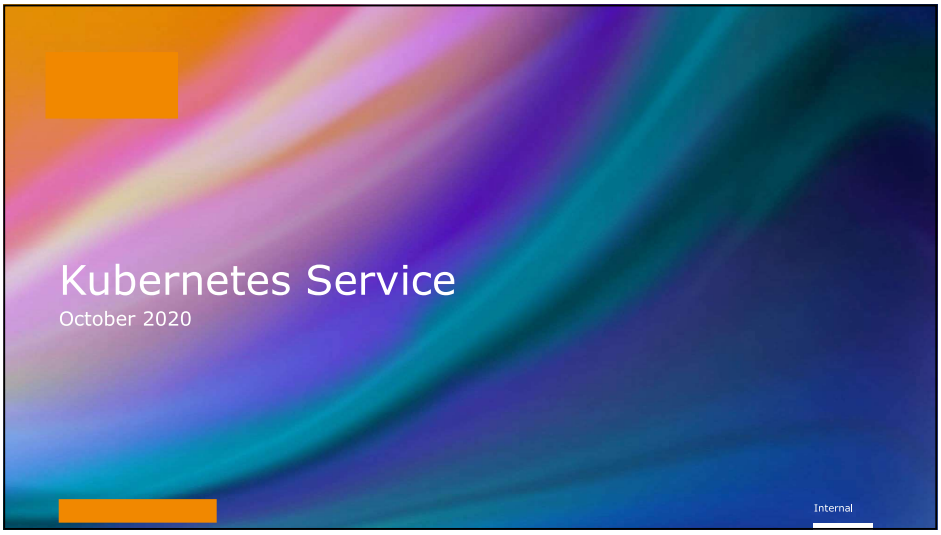
Liste der Codes	Memo	Häufigkeit
[AR02] Availability/Lokation	Ist die geografische Lokation des Systems ausschlaggebend? --- Bsp: Aber wenn man das Thema Compliance.. und … legal betrachtet, da muss man dann schon eben solche Aspekte dann mitbetrachtet. Da kommt man dann auch wieder auf Themen zurück wenn man in die Cloud geht. Also wechel Lokation verwendet man, also da kann es schon sein, dass die USA… schon mal rausfällt.	8
[AR03] Enterprise Support	Wird für das Produkt Enterprise Support benötigt? --- Bsp: Es konnte am Anfang keiner richtig einschätzen was da auf uns zukommt, weil als Beispiel, wir haben die Datenbank, da habe ich gesagt da möchte ich auf jeden Fall Support haben, und dass wir eine Enterprise Lizenz haben und dann waren wir schon bei der Basis bei so ca. € X ungefähr.	9
[AR04] Testcluster Rabatt	Ist es Voraussetzung, dass Testsysteme weniger kosten als produktive Systeme? --- Bsp: Wir haben das auch als Experiment gestarten und haben zu diesem Zeitpunkt noch nicht gewusst ob das im Unternehmen ein Erfolg werden kann. Dort ist uns auch die VMWare preislich entgegen gekommen und können ein Attraktives Angebot machen, um einiges besser als OpenShift , wo wir von Anfang an schon sehr viel zahlen hätten müssen. Das war eben der Weg. Wir wollten wirklich ganz klein Anfangen und nochmal schauen, wird da was daraus.	1
[AR05] GUI	Wird eine grafische Benutzeroberfläche für die Kubernetes Distribution benötigt? Bsp: Und die haben halt den need, dass es für sie halt einfacher geht. Deshalb Rancher, also gerade was Userverwaltung angeht, [..], backup und restore. Oder wenn wirklich die Instanzen wachsen müssen, dann ist das dort drin halt leichter zu machen, als wenn man sich das wirklich von scratch an lernen muss.	3
[AR07] rechtliche Vorgaben	Sind rechtliche Vorgaben bei der Wahl des Produktes relevant? --- Bsp: Ja also wir haben aktuell schon bei einem Kunden aus dem Bankenumfeld dieses Thema PCI DSS. Wobei des… dieses PCI DSS Thema das betrifft sowieso jede Bank in gewisser Art und Weise. Und da haben wir eben den Vorteil schon, dass wir mit RedHat OpenShift Container Lösung den PCI DSS Standard out-of-the-box erfüllen können. Und wenn wir sagen man müsste jetzt andere Distributionen evaluieren… da müsste man dann schon eben schauen, sind diese auch für diese rechtlichen Anforderungen geeignet.	6

Liste der Codes	Memo	Häufigkeit
[AR09] Strategesche Partner	Spielen bestehende Geschäftsbeziehungen oder Partnerschaften bei der Wahl des Produktes eine Rolle? --- Bsp: Und zusätzlich kommt noch dazu, dass wir als Unternehmen X RedHat Certified Cloud Service Provider sind und mit der RedHat eine Partnerschaft haben und in diesem Bereich eben sehr eng zusammenarbeiten.	6
[AR10] Verwendung der public cloud	Ist die Verwendung von public-cloud Diensten möglich/ erwünscht/nötig? --- Bsp: Also unsere Anforderungen waren jetzt an und für sich in diesem Bereich sehr überschaubar. Man hat eben.. also wir haben eben dazu die Entscheidung gehabt, dass wir das on-premises betreiben müssen. Zum damaligen Zeitpunkt war eben das Thema public-cloud eben außen vor. Also das war nicht das Thema, dass man jetzt sagt man geht in ein Azure oder ein AWS oder GPC oder was auch immer. Also damals war eben die Entscheidung schon gefällt, dass das on-premises bei uns im eigenen Data Center betrieben wird.	13
Verfügbarkeit der Daten [AR11]	Ist für die Wahl der Distribution ausschlaggebend wo sich die zu verarbeitenden Daten befinden? --- Bsp: Dann haben wir etwas herumgespielt und dann war für uns recht klar, dass wir das in der Amazon Cloud rechnen müssen, weil wir die Daten in der Amazon Cloud verfügbar haben. Das war so ein bisschen auch der Grund warum AWS überhaupt, weil dort die Daten liegen.	3
politische Vorgaben/ Herstellerstandort [AR12]	Gibt es politische Punkte, die bei der Wahl eines Herstellers zu berücksichtigen sind? --- Bsp: Sonst bei den Projekten haben wir oft mal.. das kann ich nicht.. da habe ich nicht so die Einblicke, ich bekomme es am Rande mit, dass es oft EU Projekte sind. Und wir diesen Cluster in der AWS betreiben, da ist halt of... dass wenn man ein Proposal schreibt und... sollte man jetzt vielleicht nicht rein scheiben man verwendet AWS Technologie, weil die das nicht so gerne haben, wenn es um ein EU Projekt geht. Die wollen immer, dass man auf europäischen Clouds arbeitet, aber dort haben wir noch nie so einen Cluster aufgesetzt.	5
Support auslagern [AR13]	Muss es die Möglichkeit geben den Betrieb inkl. Support des Clusters auszulagern?	1
Single Node Kubernetes [AR14]	Muss die Kubernetes Distribution auf einem Node laufen? (Single-Node/Edge K8s) --- Bsp: Edge-Lösungen für eine Maschine - "Single-Node-Kubernetes"	3
Long Time Support [AR15]	Wird Long-Time-Support für das Produkt benötigt?	1

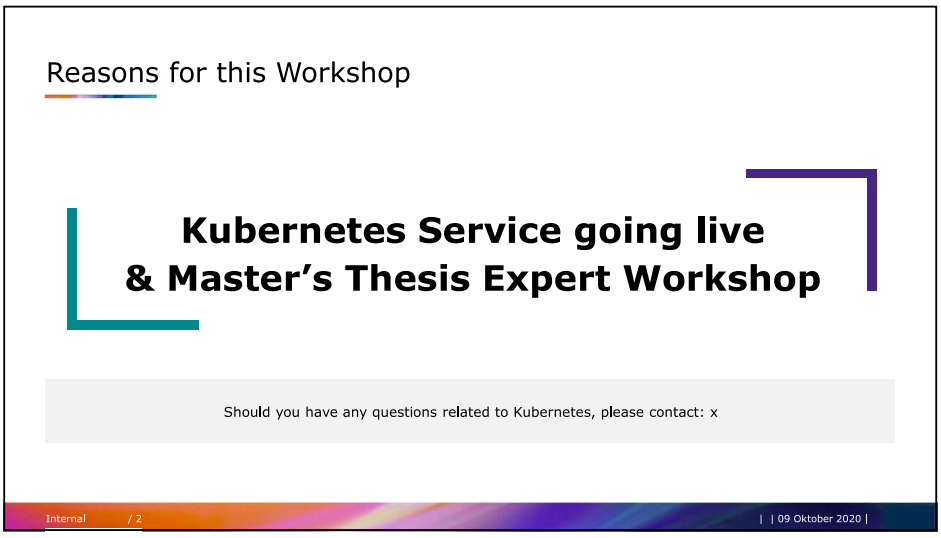
Liste der Codes	Memo	Häufigkeit
Vendor Lock-In [AR16]	Haben Hersteller/Produkte Nachteile auf Grund von vendor-lock-in Bedenken? --- Bsp: Nein wir haben uns sehr viel trotzdem zusätzlich selbst angeschaut, zum Einen weil wir eigentlich noch herstellerneutral sein wollten. Wir wollten kein komplettes VMWare vendor lock-in haben, und wir haben uns OpenShift angeschaut und uns kommt immer noch vor die all-in-one Packages sind ganz gut zum Starten.	4
Kubernetes beim Kunden	Muss Kubernetes auch beim Kunden laufen bzw. dort betrieben werden? --- Bsp: I: Verstehe ich das richtig, dass sie auch bei Kunden, On Premise, dann Cluster hinstellen? Also in deren Werk. B: Momentan noch nicht, aber ich denke das wird kommen.	7
Designprozess		4
Verwendung sinnvoll?	Ist die Verwendung von Kubernetes an sich zielführend? --- Bsp.:	1
Supportorganisation verankern	Wie wird das Thema Support für Kubernetes im Unternehmen behandelt? --- Bsp.:	2
RBAC Standardrollen vorprovisionieren	Gibt es Standardrollen (technisch(nicht-technisch) für Benutzer die mit Kubernetes arbeiten? --- Bsp.:	1

A.29 Workshop: Präsentation

18.11.2020



1



1

2

Goals of this Workshop

Provide Overview of Kubernetes Service

Common Understanding of what's available

How to get started?

Gather Requirements towards Kubernetes

Interdisciplinary & wide spread

For K8s Cluster Design & Master's Thesis

Agenda – Part 1

Intro Kubernetes Technology

Containers

Kubernetes Components

Kubernetes in Company

Numbers & Facts

Use-Cases & Examples

Kubernetes Infrastructure

Components & Functionalities

Next Steps

Agenda – Part 2

- **Overview Master's Thesis**
- **Participant's Requirements towards Kubernetes**
- **Current Thesis Results**
- **Brainstorming & Summary**

Part 1:

Intro Kubernetes Technology

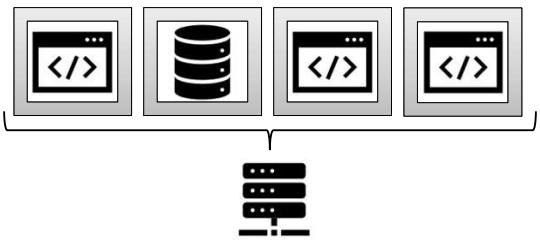
Containers

HW/VM based solution

- Different library versions
- Isolation
- Poor HW utilization
- Administrative overhead
- Custom Environments
- …

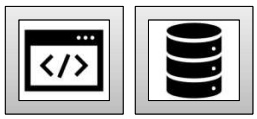
Containers

Container based solution



- Individual Containers
- Multiple Containers per Host
- Good HW utilization
- No Administrative overhead
- Standardized Environment
- Only **One** Host

Images



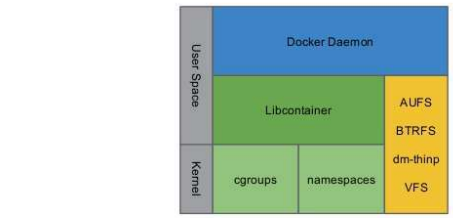
- Blueprint for Containers
  - Image = Class
  - Container = Object
- Can be shared
- Similar to .zip file
- Standardized (OCI)
- Contain all necessary files!
  - Libraries
  - Application
  - Config

Containers - Technical

Containers are a combination of OS features, filesystems and wrapper software.

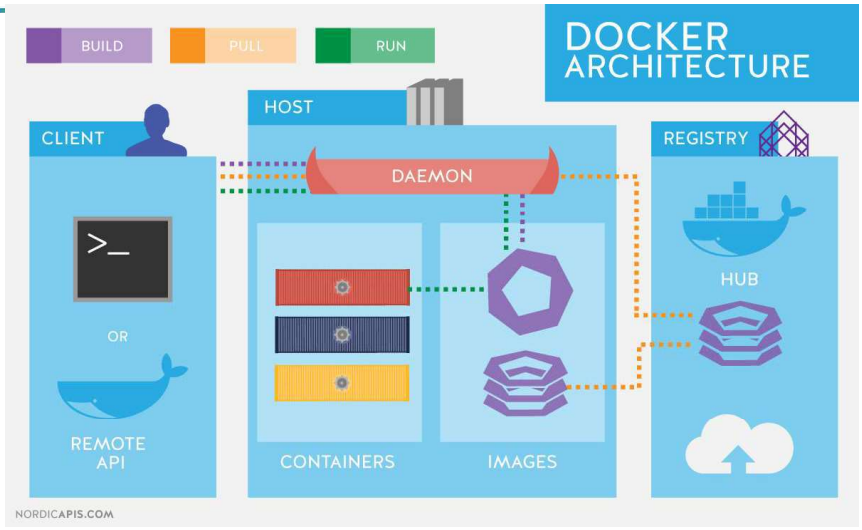
- OS features
  - CGroups
  - Namespaces
  - Virtual Interfaces
  - IPtables/IPVS
- Filesystems
  - Layered Filesystem
  - Copy-On-Write
  - AUFS
- Wrapper Software
  - Container Engine/Runtime
    - Docker
    - containerD
    - podman

Docker Components



Source: https://www.slideshare.net/RohitJnagal/docker-internals

Containers - Technical



Source: https://nordicapis.com/api-driven-devops-spotlight-on-docker/

11

Containers - in the Field

root~# docker image ls

root~# docker ps

6

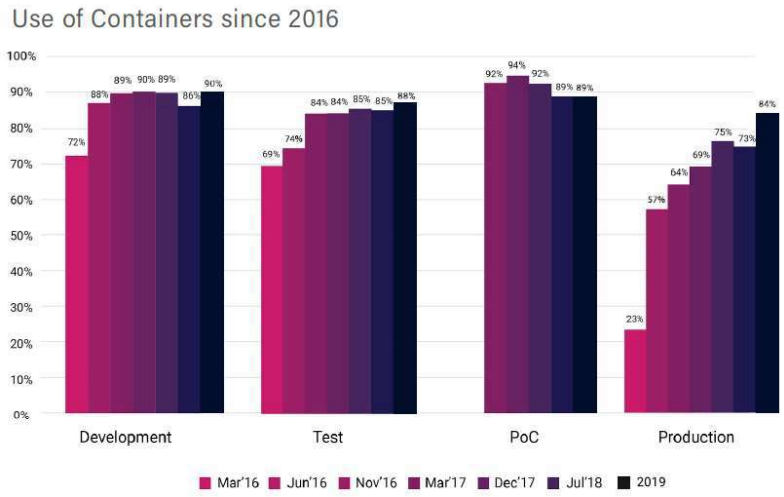
12

Containers

- 03/2016: 23%

...

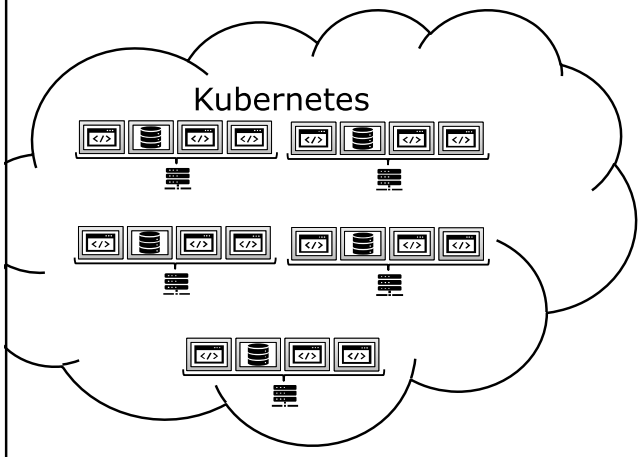
- 10/2019: 84%



Source: https://www.cncf.io/wp-content/uploads/2020/08/CNCF_Survey_Report.pdf

13

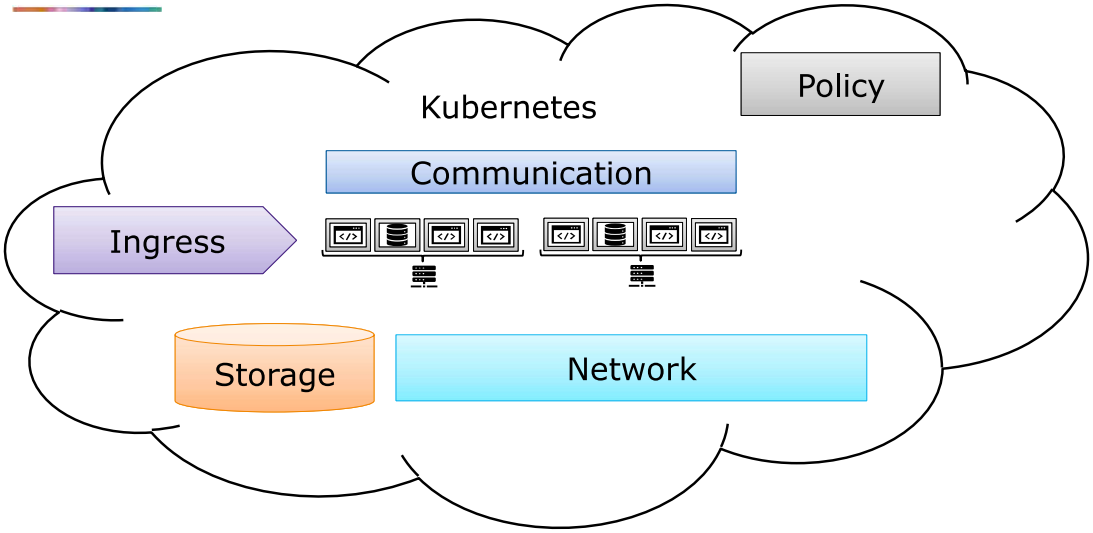
Kubernetes –Containers on multiple Nodes



- Desired State
- Container Orchestration
- Load Balancing
- Storage Provisioning
- Network Connectivity
- ...

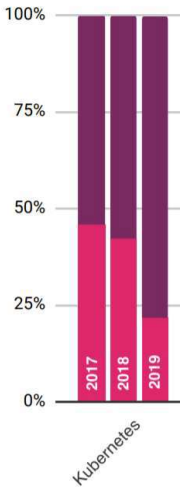
14

Kubernetes Components



15

Kubernetes Key Facts



[..] 78% of respondents are using Kubernetes in production [..]

[..] Load time for the e-commerce site was reduced by half. Releases went from every 4-6 weeks to 3-4 times a day.
- ADIDAS (https://kubernetes.io/case-studies/adidas/)

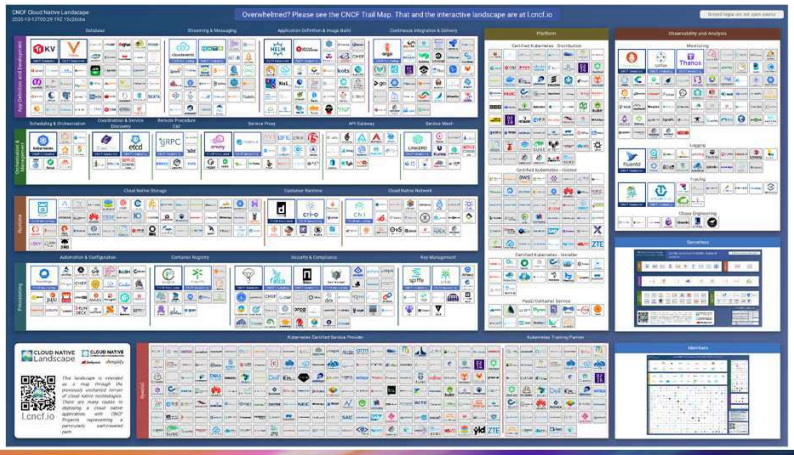
Initially, virtualization gave 20% overhead, but with tuning this was reduced to ~5%. Moving to Kubernetes on bare metal would get this to 0%. Not having to host virtual machines is expected to also get 10% of memory capacity back.
- CERN (https://kubernetes.io/case-studies/cern/)

Deployment time went from several hours to tens of seconds. Efficiency has improved by 20-30%, measured in IT costs.
- JD.com (https://kubernetes.io/case-studies/jd-com/)

16

Kubernetes Key Facts

You are viewing 1,485 cards with a total of 2,440,750 stars, market cap of $19.83T and funding of $65.3B . - https://landscape.cncf.io/



Demo

Part 2:

Overview Master's Thesis

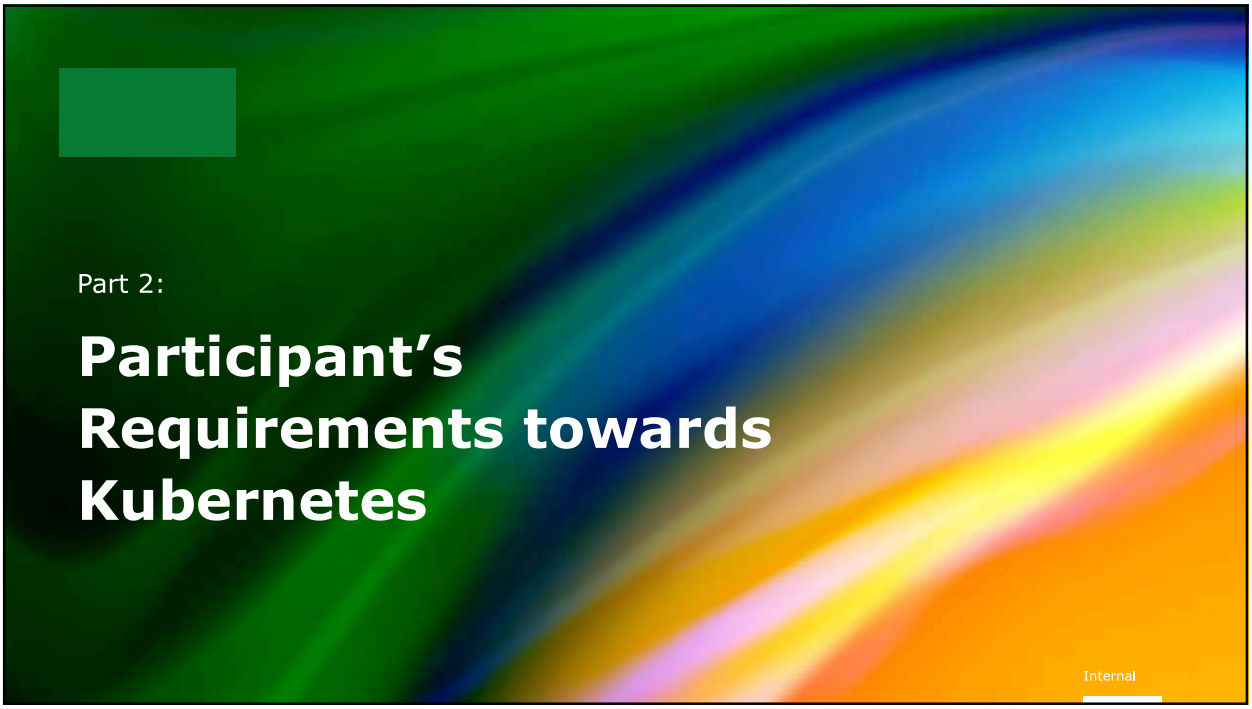
Internal

37

Innovation Driven Kubernetes Design

- Development of a structured approach for the design of a Kubernetes cluster
- Identifying all relevant components of a Kubernetes cluster
- Development of a questionnaire to specify requirements for a new Kubernetes system

38



39

Participant’s Requirements towards Kubernetes

- Prozessmanagement
- IT-Security
- Entwicklung
- IT-Architektur
- Operations
- Management
  - Personal
  - Strategie
  - Technologie

40

Part 2:

Current Thesis Results

Internal

41

Current Results

- Cluster Components: 9
  - Properties: 83
- Requirements
  - Framework Conditions: 8
  - Functional Requirements: 22
  - Quality Characteristics: 10

42

Framework Conditions

- Open Source Software:
  - Is the use of open-source software possible?
  - Which open-source licenses are allowed?
- Availability
  - Is it necessary, that the cluster is located in a specific country or region?
- Support
  - Is Enterprise Support necessary for the selected Kubernetes Distribution?
- Public Cloud
  - Is the use of public-cloud services allowed or even welcomed?

Functional Requirements

- GPU Support
- Encrypted data at rest
- OS Support for Containers (Windows/Linux)
- Restricted inter-Pod communication
- Published service protocols

Quality Characteristics

- Availability/Uptime [%]
- Installation/Upgrade [time]
- Cost [€]
- Staff [FTE]
- Market share [%]
- Community

45

Part 2:

Brainstorming & Summary

46

Participant’s Requirements towards Kubernetes

- Prozessmanagement
- IT-Security
- Entwicklung
- IT-Architektur
- Operations
- Management
  - Personal
  - Strategie
  - Technologie

A.30 Workshop 1: Protokoll

..Supportorganisation verankern
44 Supportorganisation verankern - für Service Roll-Out interessant -> als letzten Prozessschritt verankern
45 {Designproz.}
46 Marketing- & Trainingsvorlagen vorhanden - Trainings/Marketing Videos für interne Verbreitung
47 {funkt. Anf.}

48 [Produktmanagement]
49 Lizenzthema
..[AR01] Open-Source Lizenzen
50 Open-Source Lizenzen genau beachten
51 {Rahmenb.}
..Lizenzcheck im Container
52 Auch container überprüfen -> Lizenzen der SW in Containern
53 {funkt. Anf.}
..Versionsmanagement + Host/K8s Hardc
54 Patching/Wartung -> siehe oben
..Deployment zum Kunden
55 Lieferung [von Produkten] in einem produktiven industrielles Umfeld?
56 Wie wird zum Kunden ausgeliefert?
57 Prozess, technische Auswirkungen, personelle Folgen
58 {out-of-scope}
59 Frage dreht sich um Applikation, nicht um Cluster
..Softwarepacketierung
60 Standardisierung von Software Solutions - eine Ebene über Container
61 Standard Templates für Solutions/Produkte
62 {out-of-scope}
63 Frage dreht sich um Applikation, nicht um Cluster
64 Antwort: HELM
..technische Beschreibung von Infrastrukt
65 Standard der Infrastruktur-Beschreibung?
66 Kundenumbegung und eigene Umgebung beschreiben
67 {out-of-scope}
68 Implizite Eigenart von Kubernetes | "Dafür hat man Kubernetes"
69 Terraform
70 {out-of-scope}
71 Infrastruktur Automatisierungstool
..Sprache der Dokumentation [AF24]
72 Sprache der Dokumentation
73 {funkt. Anf.}
74 Dokumentation muss in bestimmten Sprachen verfügbar sein (zB Chinesisch/Deutsch)
..Vendor Lock-In [AR16]
75 Vendor Lock-In
76 {Rahmenb.}
77 Hersteller kann entweder vorgegeben oder explizit ausgeschlossen werden

78 [Field-Engineer]
..Antivirus
79 Security Scanning für Container und Nodes
80 {funkt. Anf.}
..Out-of-the-Box Funktionalität
81 Permissions/Restrictions auf POD Ebene
82 Container auf Container Zugriff
83 {out-of-scope}
84 Das macht Kubernetes implizit | Eigenart von Containern per-se
..shared Filesystem
85 Shared Filesystem - dynamische Bereitstellung von gemeinsam verwendbaren shares
86 {funkt. Anf.}
..[AF09] Logging
..Tracing
87 Nachverfolgung/Tracing
88 Logging und Nachverfolgung der Aktivitäten im Pod und

..[AF09] Logging
..Tracing
..[AF08] Metriken
..politische Vorgaben/Herstellerstandort [

zwischen den Pods
{funkt. Anf.}
Logging + Metriken + Tracing
Herstellerstandort
{Rahmenb.}
Beispiel: Chinesische Cloud-Provider oder Produkte in US derzeit schwierig

..Integration in bestehende Infrastruktur [

[Entwickler]
Kompatibilität zu bestehenden Systemen (analoges Beispiel: Browserkompatibilität)
{funkt. Anf.}
Anmerkung dreht sich um die Integration des Cluster in bestehende Kundeninfrastruktur
zB. Verwendung bestehender Storage- oder Netzwerksysteme

..backup/restore out-of-the-box [AF22]

backup/restore Funktionalitäten
{funkt. Anf.}
Anforderung beschreibt das Vorhandensein einer out-of-the-box Lösung für Backup/Restore

..[AQ09] Schulung Admins

[Enterprise Architekt]
Skills - aufbauen von Skills der MA
Wie viel Lernkurve muss man haben - K8s DEV || ADMIN Zertifizierung
{Qualitätseig.}
Messgröße: Schulungsaufwand in Zeit/Geld pro Entwickler/ Administrator

..Mandantenfähigkeit

Multimandantenfähigkeit
Policies, RBAC, Namespaces, eigene Cluster pro Mandant
{out-of-scope}

..[AF05] Betriebssystem für Container

[Entwicklungsleiter]
Verwendung bestehender Frameworks und Programmiersprachen
Betriebssystemfrage - Windows und/oder Linux
{funkt. Anf.}

..Verwendung sinnvoll?

Ist container verwendung sinnvoll?
technisch entscheiden

..[AR03] Enterprise Support

{Designproz.}
verfügbare Supportmodelle
{Rahmenb.}

..[AQ06] Communitygröße und -qualität

[alle] Qualitätseigenschaften
Communitygröße und -qualität
Messgröße: GitHub Sterne, Responsedauer, Forks

..Häufigkeit von Updates/neuen Release

Häufigkeit von Updates/neuen Releases
Messgröße: Verfügbare Updates/Releases pro Quartal

..Usability des Tools [AQ19]

Usability des Tools
Messgröße: 1-5; selbst definieren; in PoC herausfinden

..mögliche Granularität der Berechtigung

mögliche Granularität der Berechtigung out-of-the-box
Messgröße: 1-5; selbst definieren; in PoC herausfinden

..Auswertungs-/Überwachungsmöglichke

Reportingmöglichkeiten out-of-the-box
Messgröße: 1-5; selbst definieren; in PoC herausfinden

..Nutzerkreis/bestehende kunden [AQ18]

Nutzerkreis/bestehende kunden - Wer nutzt das Produkt schon? (Referenzkunden)
Messgröße: 1-5; selbst definieren

..Ausgereiftheit [AQ17]

132 bisherige Lebenszeit des Produktes/Ausgereiftheit
133 Messgröße: Alter des Produktes in Jahren; älter im Normalfall besser

..[AQ08] Durchlaufzeit/Aufwand für Insta

134 Durchlaufzeit/Aufwand für Installation
135 Use-Case suchen und Durchlaufzeit/Aufwand pro Tool vergleichen
136 Messgröße: Zeit in Stunden für erfolgreiche Umsetzung eines PoC

..Integration in bestehende Infrastruktur [

137 Connectivität zu anderen Tools
138 Messgröße: 1-5; selbst definieren; in PoC herausfinden

..Ressourcenverbrauch [AQ24]

139 Ressourcenauslastung
140 Messgröße: CPU|GPU|RAM|Netzwerk|Speicherauslastung in % pro Zeitintervall

..Support-Responsezeit auf Fehler [AQ16

141 Support-Responsezeit auf Fehler
142 Messgröße: Antwortzeit des Supportteams in Stunden

..Dokumentation [AQ15]

143 Dokumentation
144 {Qualitätseig.}
145 Messgröße: 1-5; selbst definieren; in PoC herausfinden
146 (Qualität, Umfang, Aktualität, Anwendbarkeit,..)

..[AQ04] Betriebspersonal Aufwand

147 Betriebsmannschaft
148 Größe des Teams bei 24/7 beachten (Bereitschaft)
149 {Qualitätseig.}
150 Messgröße: benötigte FTEs für Betrieb des vorliegenden Cluster

..State of the Art [AQ14]

151 State of the Art
152 Entwicklungsstand der Applikation/Distribution
153 {Qualitätseig.}
154 Messgröße: 1-5; selbst definieren; in PoC herausfinden
155 (Programmiersprache, Frameworks, Library-Versionen,..)

..Benchmarks [AQ13]

156 Industrie Benchmarks
157 heranziehen bestehender Benchmarks/Vergleiche
158 Messgröße: Benchmark-Score
159 (Gartner, Performance Benchmarks,..)

A.31 Workshop 2: Protokoll


```
1  Management Operations
2  Projektmanagement
3  Management Entwicklung
4  Managing Testing
5  DevOps / Authentifizierung

6  ###### Tue Oct 27 12:58:51 CET 2020
7  ###### Tue Oct 27 15:05:48 CET 2020

8  [Management Entwicklung]
9    Lizenzthema
10     open source ok, aber schwierig bei großen Unternehmen
11     populäre Produkte und Distributionen bevorzugt - leichter mit
lizenzen
12     {Rahmenb.}
13   Kommunikation zwischen public-cloud und on-premises muss erlaubt
sein
14     (Hybrid-Cluster | Verwendung der public-cloud)
15     {Rahmenb.}
16   Hochverfügbarkeit unabdingbar
17     {Qualitätseig.}
18     Messgröße: Verfügbarkeit in %; Downtime in Minuten; Ausfälle pro
Monat
19   Long Time Support für Distribution und Einzelkomponenten
20     {Rahmenb.}
21     Verfügbarkeit von Long-Time und Extended Support
22   persistenter Speicher wird benötigt (PVC)
23     dynamische Provisionierung von File und Block storage
24       {funkt. Anf.}
25     Lernkurve für Admin-Team beachten
26       {Qualitätseig.}
27       Messgröße: Zeit für Einschulung der K8s Admins
28   data-at-rest Encryption ist nötig für Kundendaten
29     {funkt. Anf.}
30   verschlüsselte Kommunikation (innerhalb und außerhalb d. Clusters)
31     {funkt. Anf.}
32   Windows und Linux benötigt
33     bestehende Windows Applikationen werden auf K8s migriert
34     Neuentwicklungen basierend auf Linux
35     {funkt. Anf.}
36   inter-Pod-Kommunikation muss eingeschränkt werden können
37     {funkt. Anf.}
38   Edge-Lösungen für eine Maschine - "Single-Node-Kubernetes"
39     muss mit "Hauptcluster" kompatibel sein
40     {Rahmenb.}
41     "Handelt es sich bei der Distribution um eine 'Single-Node-
Distribution'?"
42   LoadBalancing Algorithmen - Lastverteilung basierend auf definierten
Kriterien
43     {funkt. Anf.}
44   GUI beim Einstieg wichtig
45     {funkt. Anf.}

46 [Projektmanagement]
47   Dauer für PoC und Installation (auch ggf. beim Kunden)
48     {Qualitätseig.}
```

..[AQ08] Durchlaufzeit/Aufwand f

49 Messgröße: Umsetzungsdauer für PoC bzw. Installation in Stunden

50 [Management Operations]

..[AQ02] Verfügbarkeit in %

51 Verfügbarkeit -> siehe oben

..[AR02] Availability/Lokation

52 Availability - verfügbare Standorte | Wo kann Distribution betrieben werden?

53 {Rahmenb.}

54 Anforderung vom Unternehmen festzulegen

..[AQ09] Schulung Admins

55 Umschulung von (alt gedienten) Personal

56 Lernkurve für Umschulung

57 {Qualitätseig.}

58 Messgröße: Schulungsaufwand in Zeit/Geld pro Entwickler/ Administrator

..Mehrwert begreifbar machen

59 Mehrwert von K8s begreifbar machen

60 {out-of-scope}

61 zu diesem Zeitpunkt muss Mehrwert schon bekannt sein - kein Problem der Distribution

..Dokumentation [AQ15]

62 Dokumentation

63 {Qualitätseig.}

64 Messgröße: 1-5; selbst definieren; in PoC herausfinden

65 (Qualität, Umfang, Aktualität, Anwendbarkeit,..)

..[AQ04] Betriebspersonal Aufwa

66 Betriebsmannschaft

67 Größe des Teams bei 24/7 beachten (Bereitschaft)

68 {Qualitätseig.}

69 Messgröße: benötigte FTEs für Betrieb des vorliegenden Cluster

..Supportorganisation verankern

70 Supportstrukturen für Cluster beim Kunden

71 Support in Fachteams

72 {out-of-scope}

73 organisatorische Maßnahme, keine Anforderung an Cluster

..Support auslagern [AR13]

74 Betrieb inkl. Basissupport auslagern

75 {Rahmenb.}

76 "Muss Betrieb ausgelagert werden?" (zB in public-cloud)

77 [Management Testing]

..Testing von Images

78 Testing von Images

79 {out-of-scope}

80 Entwicklungsmaßnahme, nicht für Cluster relevant

..Solution neu aufbauen wenn Te

81 Solution neu aufbauen wenn Tests fehlschlagen

82 {out-of-scope}

83 Entwicklungsmaßnahme, nicht für Cluster relevant

Abbildungsverzeichnis Vertraulicher Inhalt

Tabellenverzeichnis Vertraulicher Inhalt