FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Fachhochschule Aachen, Campus Jülich

Fachbereich 09 - Medizintechnik und Technomathematik
Studiengang Angewandte Mathematik und Informatik

Seminararbeit

Vergleich verschiedener Cloud-Migrationsstrategien unter Berücksichtigung technischer und prozessualer Abhängigkeiten

Autor:
Fabian Sauer, 3571963

Betreuer:
Prof. Dr. rer. nat. Alexander Voß
Dr. rer. nat. Thomas Eifert

Aachen, 22. Dezember 2024

Eigenständigkeitserklärung

Ich, Fabian Sauer, erkläre hiermit, dass ich die vorliegende Seminararbeit mit dem Titel: Vergleich verschiedener Cloud-Migrationsstrategien unter Berücksichtigung technischer und prozessualer Abhängigkeiten selbstständig und ohne die unzulässige Hilfe Dritter verfasst habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Alle verwendeten Quellen und Hilfsmittel sind vollständig und genau angegeben.

Ich versichere weiterhin, dass diese Arbeit in gleicher oder ähnlicher Form noch nicht in einem anderen Prüfungsverfahren vorgelegt wurde.

Ich verpflichte mich, ein Exemplar der Seminararbeit fünf Jahre aufzubewahren und auf Verlangen dem Prüfungsamt des Fachbereiches Medizintechnik und Technomathematik auszuhändigen.

Ort, Datum: Aachen, 22.12.2024

Unterschrift: 

Zusammenfassung

Cloud-Computing revolutioniert die Art und Weise, wie Unternehmen digitale Ressourcen nutzen und verwalten. Durch die flexible Bereitstellung von Rechenleistung ermöglicht die Nutzung der Cloud-Technologien, eine neue Art und Weise Software zu betreiben.

Diese Seminararbeit untersucht unterschiedliche Strategien zur Migration lokal betriebener Services in die Cloud. Dabei werden die vier Strategien Rehosting, Replatforming, Refactoring und Replacing untersucht, welche ein Teil des 6-R-Modells sind. Im Rahmen der Untersuchung werden die verschiedenen Migrationsansätze anhand technischer Abhängigkeiten, wie Datenbanken, Identitätsanbieter und Netzwerkinfrastruktur, sowie prozessualer Abhängigkeiten, wie Monitoring, Incident-Management und Backupprozesse, verglichen. Dabei wird gezeigt, wie jeder Ansatz mit der Abhängigkeit umgeht und wie eine typische Umsetzung aussieht. Anhand dessen werden die Ansätze pro Strategie mittels unterschiedlicher Kriterien verglichen. Dies umfasst unter anderem die Skalierbarkeit, Verfügbarkeit und den Verwaltungsaufwand des in die Cloud migrierten Services sowie den Migrationsaufwand.

Diese Betrachtung wird anhand eines Beispiels, der Migration eines lokal betriebenen Exchange-Servers in die Cloud, für jeden Migrationsansatz exemplarisch durchgeführt. Dabei werden die jeweiligen Eigenschaften der Ansätze miteinander verglichen.

Die Seminararbeit liefert ein tiefgehendes Verständnis der verschiedenen Migrationsansätze, wie diese angewendet werden können und welche Auswirkungen sie auf die bestehenden Abhängigkeiten haben. Dadurch wird es möglich, bei einer Migration eine fundierte Entscheidung zu treffen, welche Strategie unter Berücksichtigung aller Abhängigkeiten am besten für das aktuelle Vorhaben und die gegebenen Rahmenbedingungen geeignet ist.

Inhaltsverzeichnis

1 Einleitung

1.1 Aktuelle Relevanz der Cloud

Die Relevanz von Cloud-Computing hat in den vergangenen Jahren massiv zugenommen [noak], da die Cloud Unternehmen ermöglicht, flexibel und kosteneffizient auf IT-Ressourcen zuzugreifen. Immer mehr Firmen wollen aus diesem Grund ihre Systeme von einer lokalen On-premise-Lösung in eine Cloud-Umgebung migrieren [KA24]. Die Migration ist dabei kein triviales Unterfangen und erfordert ein umfassendes Verständnis der Cloud-Technologien sowie des gesamten Migrationsprozesses.

1.2 Zielsetzung

Diese Seminararbeit beschäftigt sich mit den verschiedenen Arten der Cloud-Migration. Dabei werden vier Cloud-Migrationsstrategien gegenübergestellt und anhand von technischen und prozessualen Abhängigkeiten verglichen. Das ermöglicht eine bessere Beurteilung, welche Cloud-Migrationsstrategie am besten geeignet ist, die lokal betriebenen Services mit allen erforderlichen Abhängigkeiten sowohl in Bezug auf Hardware und Software als auch personelle Ressourcen erfolgreich zum gewünschten Ergebnis zu führen.

2 Cloud-Migration

2.1 Cloud-Computing-Modelle

Cloud-Computing ist definiert als „ein Modell für den allgegenwärtigen, bequemen und bedarfsgerechten Netzzugang zu einem gemeinsamen Pool konfigurierbarer Computerressourcen (z. B. Netze, Server, Speicheranwendungen und -dienste), die mit minimalem Verwaltungsaufwand oder minimaler Interaktion mit dem Dienstanbieter schnell bereitgestellt und freigegeben werden können.“ [MG]. Cloud-Computing wird dabei primär als Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS) oder Infrastructure-as-a-Service (IaaS) angeboten. Bei SaaS wird eine Softwarelösung meist über das Internet bereitgestellt und vom Anbieter betrieben und entwickelt. Ein Beispiel für SaaS ist Microsoft Office 365 oder Salesforce. PaaS bietet eine Ausführungsumgebung für eigene Software. Die komplette Infrastruktur und ein Großteil der nötigen Konfiguration werden dabei von dem Cloud-Provider übernommen. Der Nutzer muss nur die ausführbare Software, oftmals in Form eines Docker Images, bereitstellen. Ein Beispiel für PaaS ist unter anderem Elastic Container Service (ECS) von Amazon Web Services (AWS) oder auch Google Kubernetes Engine (GKE). IaaS stellt anders als bei den anderen beiden Varianten nur grundlegende IT-Infrastrukturen in Form von (virtuellen) Servern, Speichern und Netzwerken zur Verfügung. Der Nutzer hat die volle Kontrolle über diese Systeme, muss sie aber auch selbst verwalten. Der Cloud-Provider stellt in diesem Fall nur die unterliegende Hardware zur Verfügung und kümmert sich um diese. Zu den Beispielen für IaaS gehören Elastic Compute Cloud (EC2) von AWS oder auch die Google Compute Engine (GCE) [AG17] [MG].

2.2 Arten von Clouds

Es gibt drei verschiedene Arten an Cloud, und zwar Public, Private und Hybrid. Public Clouds werden von einem Cloud Service Provider (CSP) betrieben, welcher seine Ressourcen an mehrere Kunden untervermietet (Multi-Tenancy) und die Cloud-Infrastruktur verwaltet. Dabei zahlt jeder Kunde meinst nur genau das an Leistung, wie Rechenkapazität, Speicherplatz oder Datenübertragungsvolumen, welche dieser auch tatsächlich nutzt (pay as you go/pay-per-use) [KKA14] [AG17]. Die größten Public Clouds heutzutage sind Amazon Web Services mit 31%, Microsoft Azure mit 25% und Google Cloud Platform mit 10% Marktanteil im Q1 2024 [noao]. Private Clouds oder auch internal Clouds basieren auf den Techniken von Public Clouds. Anders als diese werden Private Clouds nur von einer Organisation benutzt (Singel-Tenancy). Diese bieten einen höheren Grad an Kontrolle, haben allerdings nicht so ausgedehnte Möglichkeiten zu skalieren wie die großen Public Clouds. Hybride Cloud-Modelle verbinden die beiden vorherigen Modelle und ermöglichen Unternehmen, Services sowohl in der Private Cloud als auch in verschiedenen Public Clouds laufen zu lassen. Bei besonders großer Auslastung können Services dann zum Beispiel aus der Private Cloud in die Public Cloud skalieren [MC22] [AG17]. In diesem Text wird der Begriff „Cloud“ spezifisch im Kontext der Public Cloud verwendet.

2.3 Definition Cloud-Migration

Cloud-Migration bezeichnet den Prozess der Überführung von Anwendungen, Daten und Infrastruktur in eine Cloud-Computing-Umgebung. Dabei werden meist Services von lokalen Systemen (On-Premise) in eine Cloud-Umgebung gebracht. Cloud-Migration kann auch den Prozess der Migration zwischen verschiedenen Cloud-Anbietern beschreiben (Cloud-Cloud Migration), oder die Migration aus der Cloud auf ein On-Premise-System (Reverse-Migration). Im Folgenden bezeichnet Cloud-Migration ausschließlich die Migration von On-Premise zu einem Cloud-Anbieter [ANH18].

2.4 Beweggründe für die Migration in die Cloud

Es gibt eine Reihe von Gründen, warum Unternehmen in die Cloud migrieren wollen. In dieser Arbeit werden diese in organisatorische und technologische Ziele aufgeteilt. Unter den technologischen Zielen sind die Skalierbarkeit und Flexibilität einer der Hauptpunkte für viele Firmen [ANH18]. Weitere Gründe sind aber auch eine Kostensenkung oder IT-Infrastruktur, eine verbesserte Verfügbarkeit, schnelle Bereitstellung von neuen Ressourcen und Zugriff auf neue Technologien [Geo24]. Bei organisatorischen Zielen der Cloud-Migration spielen die Konzentration auf das Kerngeschäft, Steigerung der Agilität und Innovationsfähigkeit sowie verbesserte Zusammenarbeit eine große Rolle [Geo24]. Gründe, warum Unternehmen sich dagegen entscheiden, einen Service in die Cloud zu migrieren, sind unter anderem der hohe initiale Aufwand, der Verlust an Kontrolle über die Daten und ein hoher Grad an Abhängigkeit gegenüber den Cloud-Providern (Vendor Lock-in).

3 Migrationsansätze

Es gibt zahlreiche Strategien und Ansätze, um Services in die Cloud zu migrieren. Eine der populärsten Herangehensweisen ist das sogenannte „6R Modell“. Das 6R Modell ist sowohl im akademischen als auch bei Unternehmen weitverbreitet [ANH18] [Ste16].

Dieses Modell baut ursprünglich auf dem 2011 von Gartner veröffentlichten „5R Modell“auf, einer ersten systematischen Einordnung von Migrationsansätzen, die Unternehmen berücksichtigen sollten [CR11]. Dieses Modell wurde 2016 von Stephen Orban, der zu diesem Zeitpunkt „Global Head of Enterprise Strategy“bei AWS war [noa24], um eine weitere Strategie erweitert, sodass das „6R's“Framework entstand [Ste16].

Orbans 6R-Modell nennt die folgenden Strategien: Rehosting, Replatforming, Repurchasing, Retire und Retain [Ste16]. Dabei werden Retire und Retain nur kurz in dieser Arbeit behandelt, da diese nicht zum Ergebnis führen, dass die Software in der Cloud läuft.

3.1 Rehosting

Bei Rehosting (auch bekannt als Lift-and-Shift) wird der Service ohne große Änderungen der Architektur in die Cloud gebracht. Dies wird gemacht, indem die On-Premise Hardwareinfrastruktur als IaaS beim Cloud-Provider gebucht und die Software dann nahezu identisch dort installiert wird.

3.2 Replatforming

Beim Replatforming, auch als Lift-Tinker-and-Shift bezeichnet, bleibt die grundlegende Architektur des Services unverändert, während kleinere Anpassungen vorgenommen werden, um den Service in die Cloud zu migrieren. Diese Anpassungen bestehen darin, einige Services in Cloud-Native Services zu migrieren. Bei diesen Cloud-Nativen Services handelt es sich in der Regel um PaaS oder SaaS, die vom Cloud-Provider betrieben und verwaltet werden und die Vorteile der Cloud voll ausnutzen [IKZ+22]. Diese Services können zum Beispiel die Datenbank sein, die bei einer Migration in AWS dann in den Relational Database Service (RDS) migriert werden [Ste16] [ANH18].

3.3 Refactoring / Rearchitecting

Beim Refactoring oder Rearchitecting wird die bestehende Architektur grundlegend überarbeitet, um sie so für die Cloud zu optimieren. Dieser Ansatz erlaubt die Nutzung Cloud-nativer Services wie Elastic Kubernetes Service oder Google Kubernetes Engine um containerisierte Microservices bereitzustellen. Zudem werden Datenbanksysteme in verwaltete Plattformdienste (PaaS) wie RDS oder Google Cloud SQL migriert [Ste16] [ANH18].

3.4 Replacing / Repurchasing

Der Replacing beziehungsweise Repurchasing Ansatz verfolgt eine ganz andere Strategie als die bisher behandelten Methoden. Bei dem Replacing Ansatz wird die Legacy-Software nicht in die Cloud migriert, sondern mit einem neuen, bereits Cloud-nativen, Service ersetzt. Oftmals wird auch zu einem SaaS gewechselt [Ste16] [ANH18]. Anders als bei den bisher beschriebenen Ansätzen wird bei dem Replacing nur der Service in die Cloud migriert und nicht die konkrete Software, aus dem der Service besteht. Ein Service ist „Eine Sammlung von zusammenhängenden IT-Komponenten, die zur Unterstützung eines oder mehrerer Geschäftsprozesse bereitgestellt werden.“ [Edi]. Da beim Replacing-Ansatz der Wechsel zu einem SaaS-Produkt am typischsten ist, wird dieser im Folgenden primär im Kontext des Replacing-Ansatzes behandelt [Ste16].

3.5 Retire

Bei der Retire Strategie geht es darum, bei der Migration großer Services herauszufinden, welche Dienste nicht mehr benötigt werden und infolgedessen nicht migriert werden müssen. Laut [Ste16] werden in großen Unternehmen mit vielen IT-Services ca. 10% aller betriebenen Services nicht mehr benötigt [Ste16].

Da diese Strategie nicht zur Folge hat, dass eine Software am Ende in der Cloud betrieben wird, wird sie in dieser Arbeit nicht weiter behandelt. Allerdings ist es sinnvoll, diese zu berücksichtigen, da sie den Migrationsprozess abkürzen und nicht benötigte Arbeit einsparen kann. Außerdem ist jeder abgeschaltete Dienst besonders in der Cloud mit direkten Einsparungen aufgrund des Pay-per-Use-Modells und potenziell weniger Angriffspunkten verbunden [KKA14].

3.6 Retain

Genau wie Retire hat auch der Retain Ansatz keine in der Cloud laufende Software zur Folge. Bei dem Retain Ansatz wird ein Service erst mal weiter On-Premise betrieben und zu einem späteren Zeitpunkt erneut bewertet. Dies kann verschiedene Gründe, wie beispielhaft eine zu komplexe Migration, oder Einschränkungen durch Compliance-Richtlinien haben [ANH18]. Auch dieser Ansatz wird aus den genannten Gründen nicht weiter in dieser Arbeit behandelt.

4 Vergleich nach Abhängigkeiten

Die Migration von Software in die Cloud stellt viele Firmen vor große Herausforderungen. Einen großen Anteil daran haben die vielfältigen technischen und prozessualen Abhängigkeiten, die mit komplexen Softwaresystemen kommen.

Im Folgenden wird für jede Abhängigkeit jeweils eine beispielhafte Umsetzung der betrachteten Migrationsansätze beschrieben. Dabei werden die unterschiedlichen Ansätze hinsichtlich ihrer Eigenschaften verglichen. Meist werden dabei die Amazon Web Services Services benannt, allerdings lassen sich diese auf Services der anderen Cloud-Anbieter übertragen [noas].

4.1 Technische Abhängigkeiten

Technische Abhängigkeiten sind alle externen Elemente einer Software, die maßgeblich für die implementierte Funktionalität benötigt werden. Die Software kann funktional von externen Elementen abhängig sein. Dazu gehören unter anderem Code Libraries, die genutzt werden um bestimmte Funktionalitäten zu integrieren, ohne sie komplett neu schreiben zu müssen, als auch ein Netzwerk, welches der Software ermöglicht, andere Dienste zu erreichen, sowie selbst erreicht zu werden. Außerdem kann die Software von externen Elementen abhängig sein, die Daten für den Betrieb zur Verfügung stellen, wie externe Identitätsanbieter, die Identitäten bereitstellen oder Datenbanken, die sämtliche Daten für eine Software bereitstellen können [CMRH09].

4.1.1 Datenbank

Datenbanken sind für viele Systeme ein zentraler Dienst und haben somit eine hohe Relevanz bei der Migration. Nach einer erfolgreichen Datenbankmigration sollten insbesondere folgende Eigenschaften gewährleistet sein und somit bei der Wahl der Strategie berücksichtigt werden: hohe Skalierbarkeit, verbesserte Verfügbarkeit, reduzierte Infrastrukturkosten, schnelle Bereitstellung einer Instanz sowie eine möglichst schnelle Migration [AV21].

Bei dem Rehosting-Ansatz würde die Datenbank ohne größere Anpassungen neu auf einer vom Cloud-Anbieter bereitgestellten virtuellen Maschine installiert werden (IaaS), die jedoch weiter eigenständig verwaltet werden muss. Dieser Ansatz ist somit ein zügiger und kostengünstiger Migrationsansatz für die Datenbank. Zudem ist eine maximale Flexibilität bei der Auswahl des Datenbankmanagementsystem (DBMS) gegeben. Allerdings optimiert dieses Verfahren kaum die Datenbank mit Hinsicht auf Erreichbarkeit und Performanz.

Anders ist es beim Replatforming-Ansatz. Dabei wird die Datenbank bereits in einen vom Cloud-Provider verwalteten Dienst (PaaS), wie RDS was ein Database-as-a-Service (DBaaS) ist, migriert. Dieser Ansatz bringt eine Reihe an Vorteilen mit sich: Die Datenbank muss nicht mehr selbst aufgesetzt und verwaltet werden, zudem sind solche Dienste für die Cloud optimiert, was unter anderem dazu führt, dass sie standardgemäß eine erhöhte Verfügbarkeit durch Redundanz vorweisen. Ein weiterer Vorteil liegt darin, dass sowohl Rechenleistung als auch Speicher meist automatisiert und bedarfsgerecht hinzubuchbar sind, sodass im Vergleich mit dem Rehosting-Ansatz die Datenbank mit einer höheren Flexibilität und geringeren Kosten betrieben werden kann. [CJP+] Die Migration in einen solchen Service erfordert allerdings einen höheren Aufwand und ist damit auch mit

höheren Kosten verbunden. Zudem kann es durch die Umstrukturierung Probleme mit dem später besprochenen Legacy-Systeme geben. Sofern dieses direkt von der Datenbank abhängig ist, würde es dann nicht mehr funktionieren.

Im Rahmen des Refactoring-Ansatzes wird die Struktur der Daten im Gegensatz zum Replatforming für die Cloud gezielt optimiert. Das kann zur Folge haben, dass die Daten beispielhaft von einer relationalen Datenbank in eine nicht-relationale Datenbank migriert werden. Diese Umstellung kann in einigen Fällen einen massiven Vorteil in der Cloud bringen, da nicht-relationale Systeme oft besser skalieren und somit die Vorteile der Cloud besser ausnutzen [HB17]. Das Refactoring bringt allerdings auch ein erhöhtes Risiko mit sich, da die Umstellung der Datenstruktur komplex und somit fehleranfälliger ist, was zu einem aufwendigen und damit teurem Migrationsprozess führen kann.

Beim Replacing spielt die Datenbank meist keine große Rolle, da sie von dem Anbieter der SaaS Anwendung betrieben und verwaltet wird. Allerdings ist hierbei der Migrationsaufwand der Daten aus der alten Datenbank in das neue System meist sehr hoch. Zudem wird die Kontrolle der kompletten Daten an den SaaS-Betreiber übergeben, was je nach Art der Daten als kritisch oder vorteilhaft betrachtet werden kann. [Bas24]

4.1.2 Netzwerkinfrastruktur

Netzwerke sind die Verbindungselemente und Zugangspunkte zu den meisten Services, die heute von Unternehmen betrieben werden [BASS11]. Dementsprechend haben sie eine hohe Relevanz bei der Migration in die Cloud. Der Cloud-Provider stellt dafür ein Network-as-a-service (NaaS) zur Verfügung. Dabei sollte ein Netzwerk in der Cloud skalierbar und flexibel sein, eine hohe Verfügbarkeit bieten, möglichst geringe Infrastrukturkosten verursachen und sich schnell bereitstellen lassen. Zudem sollte es sowohl in der Implementierung als auch im Betrieb wenig Wartungsaufwand erfordern. Des Weiteren ist zu beachten, dass eventuell Cloud-Provider-spezifische Bestandteile benutzt werden können oder müssen, was zu einem Vendor-Lock-in führen kann. Gleichzeitig teilen sich physische Netzwerke oft Ressourcen mit anderen Nutzern, weshalb ein angemessener Schutz der Daten von großer Bedeutung ist.

Der Rehosting-Ansatz zieht das Netzwerk besonders für die Erreichbarkeit der Anwendung in die Verantwortung und muss deswegen meist nur in dieser Hinsicht neu konzeptioniert werden. Dabei ist besonders die Änderung zu beachten, dass viele Services über Netzwerkgrenzen hinweg kommunizieren. Im Gegensatz zu einem On-Premises-Netzwerk, bei dem die interne Kommunikation typischerweise innerhalb des geschützten Firmen-Netzwerks bleibt, muss hier jede Kommunikation über geteilte Ressourcen hinweg abgesichert erfolgen. Andere Konfigurationen, wie Firewall-Regeln oder VLANs für verschiedene Dienste können meist übernommen werden, müssen aber auf die passenden Services des Cloud-Providers angepasst werden [BASS11]. Die Netzwerkstruktur des bestehenden Legacy-Systems kann dabei größtenteils beibehalten werden, muss jedoch implementierungstechnisch komplett an den Cloud-Provider angepasst werden [Pal23]. Das bringt viele Vorteile mit sich: Der Cloud-Provider sorgt durch Redundanz für eine erhöhte Verfügbarkeit und steigert die Flexibilität, da beispielsweise, um ein neues NAT-Gateway anzulegen, keine neue Hardware benötigt wird und NAT-Gateways in einer Standardkonfiguration in wenigen Sekunden bereitgestellt werden. Da in einem solchen Netzwerk meist nur grundlegende Netzwerkbestandteile benutzt werden, welche meist gratis sind, werden die Infrastrukturkosten um ein Vielfaches geringer. Sollte das Legacy-Netzwerk jedoch komplex sein und zahlreiche Komponenten wie Load-Balancer oder NAT-Gateways nutzen, können die Infrastrukturkosten bei einer direkten Übertragung in die Cloud erheblich steigen, da diese ohne eine optimierte Nutzung häufig kostenintensiv sind [noal].

Ähnlich ist der Replatforming-Ansatz, welcher ebenfalls große Teile der bestehenden Netzwerkstruktur übernimmt und anschließend mit den Services des Cloud-Providers implementiert. Allerdings ermöglicht der Replatforming-Ansatz, einzelne Teile der Netzwerkstruktur in Hinsicht auf Kosten zu optimieren, indem beispielsweise mehrere Load-Balancer zusammengefasst, oder NAT-Gateways

eingespart werden, was eine Kostenersparnis ermöglicht, ohne die Funktionalität oder Performance zu beeinträchtigen [noap]. Zusätzlich müssen die dadurch entstandenen Services, wie Datenbanken, im Netzwerk von den richtigen Servern erreicht werden können. Diese Anforderungen machen die Replatforming-Migration insgesamt aufwendiger als den Rehosting-Ansatz. Die Vorteile, wie eine hohe Flexibilität und Verfügbarkeit, bleiben jedoch ebenso erhalten. Da die Anbindung der neuen Services meist keine kostenintensiven Netzwerkkomponenten wie Load-Balancer oder NAT-Gateways erfordert und Kosten eingespart werden können, sind die Gesamtkosten im Vergleich zum Rehosting-Ansatz in der Regel niedriger [noal].

Grundsätzlich differenziert sich der Refactoring-Ansatz, bei dem die Netzwerkstruktur aufgrund der neu entworfenen Softwarearchitektur vollständig neu konzipiert werden muss. Das hat zur Folge, dass eine Migration des Netzwerks nicht möglich ist und somit ein erheblicher Aufwand in die Neukonzeption des Netzwerks fließen muss. Dabei gewinnt man gegenüber dem Replatforming-Ansatz auf der Netzwerkebene keine Vorteile. Eigenschaften wie Flexibilität und Verfügbarkeit sind identisch und auch die Infrastrukturkosten unterscheiden sich kaum, da auf teure Netzwerkelemente Rücksicht genommen wird und diese optimiert eingesetzt werden können.

Bei dem Replacing-Ansatz spielt eine Migration des Netzwerkes keine Rolle, da der SaaS-Betreiber für die Netzwerkinfrastruktur verantwortlich ist. Dennoch bleibt der Vorteil einer hohen Verfügbarkeit durch die Nutzung der Cloud gegeben. Die Flexibilität kann eingeschränkt sein, da durch den SaaS-Betreiber eventuell nicht auf alle Ressourcen voll zugegriffen werden kann. Zudem müssen bei dem Replacing-Ansatz die Mitarbeiter nicht für die Nutzung der Cloud Netzwerkkomponenten geschult werden, was bei den anderen Ansätzen nötig ist. Zwar ähneln die Grundprinzipien eines On-Premises-Netzwerks denen eines Cloud-Netzwerks, jedoch erfordert die Einrichtung eines sicheren Netzwerks bei einem Cloud-Provider detailliertes Wissen über spezifische Mechanismen [noad].

4.1.3 Legacy-Systeme

Bei einigen Migrationen wird nicht das gesamte System migriert, sondern einzelne Services bleiben weiterhin On-Premise. Dies geschieht häufig, wenn die Migration eines einzelnen Service zu aufwendig ist, besonders kritische Daten betroffen sind, die nicht in die Cloud ausgelagert werden sollen, oder wenn das System erst zu einem späteren Zeitpunkt migriert werden soll, wie es beim Retain-Ansatz der Fall ist. Wichtig ist jedoch, dass die migrierten Services in der Cloud weiterhin mit den On-Premises-Services zusammenarbeiten können. Hierbei muss sichergestellt werden, dass eine Netzwerkverbindung zwischen dem On-Premises-Service und den in der Cloud betriebenen Services besteht. Damit die in der Cloud laufenden Services möglichst effizient und ohne Einschränkungen funktionieren können [GPT20].

Da sich bei dem Rehosting-Ansatz ausschließlich der Standort des Services ändert, muss lediglich dafür gesorgt werden, dass die notwendige Netzwerkverbindung zwischen Cloud und On-Premises-Lösung vorhanden ist. Für diesen Zweck bieten Cloud-Anbieter fertige Lösungen an, wie die Einrichtung eines sicheren VPN-Tunnels, um die Kommunikation zwischen On-Premises-Netzwerk und Cloud-Netzwerk zu ermöglichen [Pal23] [Bak14]. Dabei ist es allerdings auch nötig, das On-Premise laufende System oder die On-Premise Netzwerkinfrastruktur anzupassen und eventuell einen neuen Service für die Kommunikation in dem Cloud-Netzwerk bereitzustellen. Der Aufwand vor dem Betrieb ist durch die fertige Lösung des Cloud-Providers meist überschaubar, allerdings entstehen langfristige Betriebskosten, da die zusätzlichen Services weiterhin betrieben werden müssen [noaq]. Diese sind jedoch in der Regel nicht besonders hoch, sodass die zusätzlichen Infrastrukturkosten des On-Premises-Betriebs im Vergleich zu den Migrationskosten oft gerechtfertigt werden können [noai]. Das Legacy-System schränkt die mit dem Rehosting-Ansatz migrierten Services in Bezug auf Flexibilität und Skalierbarkeit nicht ein, da sich lediglich der Kommunikationsweg verändert. In Hinsicht auf Performance gibt es lediglich bei der Übertragungsgeschwindigkeit zwischen den Systemen Einbußen. Die Verfügbarkeit des gesamten Systems ist weiter in eigener Verantwortung. Die Anzahl der Services, die On-Premise betrieben werden, hat sich verringert. Allerdings hat

sich die Verfügbarkeit des gesamten Systems nicht verbessert, sofern die in der Cloud laufenden Services voll abhängig von dem On-Premises-Service sind, im Vergleich zu einer Situation ohne Migration.

Der Replatforming-Ansatz verhält sich zunächst ähnlich wie der Rehosting-Ansatz. Es muss eine Verbindung zwischen den On-Premises- und den Cloud-Services geschaffen werden, was die bereits beschriebenen Nachteile, wie den zusätzlichen Verwaltungsaufwand und mögliche Performanceeinbußen, mit sich bringt. Genauso verhält es sich auch bei der Erreichbarkeit, wenn die migrierten Services voll von dem On-Premise Service abhängig sind, ist eine verbesserte Verfügbarkeit, nicht mehr, als bei der On-Premise-Service Lösung gegeben. Dementsprechend kann hier ebenfalls nicht von guter Verfügbarkeit des Cloud-Providers profitiert werden. Die Flexibilität bleibt ebenfalls weitgehend unbeeinträchtigt, solange eine stabile Netzwerkverbindung zwischen den Systemen besteht. Anders verhält es sich jedoch bei Skalierbarkeit und Performance, da bei dem Replatforming-Ansatz bereits einige Services in Cloud-native migriert wurden, können diese in der Regel besser skalieren [GPT20], was durch das Legacy-System eingeschränkt wird. Problematisch wird es, wenn diese Cloud-nativen Services direkt von nicht migrierten On-Premise-Services abhängig sind. In diesem Fall wird der On-Premise-Service zum Bottleneck, da dieser nicht in dem Maße und ohne weiteren Aufwand mit skalieren kann, was wiederum zu Performanzproblemen führt, oder die Skalierbarkeit des Cloud-nativen Services hinfällig macht.

Der Refactoring-Ansatz ist nahezu identisch zu dem Replatforming-Ansatz, nur dass bei dem Refactoring-Ansatz die Problematik nicht skalierender Legacy-Services und die daraus entstehenden Herausforderungen deutlich stärker ausfällt [GPT20]. Das liegt daran, dass beim Refactoring alle Systembestandteile gezielt auf Skalierbarkeit, Flexibilität und Effizienz optimiert werden. Allerdings besteht bei dem Refactoring-Ansatz durch die fundamentale Umstrukturierung der Software die Möglichkeit, dass die Abhängigkeit zu dem Legacy-Service verringert wird, sodass die Einschränkungen durch diesen nicht so groß sind.

Bei dem Repurchasing-Ansatz hat das Legacy-System nur dann eine Relevanz, wenn es weiter benötigt wird. Dies kann der Fall sein, weil es etwa Funktionalität beinhaltet, die die neue SaaS-Anwendung nicht bereitstellt, oder Daten verwaltet, die nicht an den SaaS-Betreiber weitergegeben werden sollen. In diesem Fall entstehen die gleichen Probleme des Refactoring-Ansatzes. Der Legacy-Service stellt in Hinsicht auf Skalierbarkeit und Verfügbarkeit der Bottleneck dar. Zudem ist bei dem Repurchasing-Ansatz die Integration des Legacy-Systems auf SaaS-Seite häufig problematisch und meist mit hohen Kosten verbunden, da der SaaS-Betreiber extra Ressourcen aufbringen muss.

Meist unabhängig von dem Ansatz gibt es zudem auch Probleme, wenn das Legacy-System von einem oder mehr Cloud-Services abhängig ist. Dabei ist besonders die Datenbank eine große Herausforderung. Hier gibt es die Option eine Verbindung zur entfernten Datenbank herzustellen oder eine zweite Instanz laufen zu lassen, welche dann synchronisiert wird. Allerdings ist besonders die Synchronisierung in Form von Replicas nicht einfach einzurichten und hat meist eine schlechtere Performance [MPB20].

4.1.4 Bibliotheken und Betriebssystem

Bibliotheken sind ein unverzichtbarer Bestandteil jeder Software. Sie ermöglichen eine effiziente Softwareentwicklung und fördern die Wiederverwendung bestehender Funktionalitäten [HNZ24]. In vielen Projekten ist jedoch eine exakte Version der Bibliothek erforderlich, um die korrekte Funktionsweise der Software sicherzustellen. Ebenso wird auch eine spezifische Betriebssystemart sowie eine bestimmte Version benötigt, damit die Software richtig funktionieren kann. Diese Anforderungen bringen bei der Migration verschiedene Herausforderungen mit sich, da sie berücksichtigt werden müssen. Gleichzeitig eröffnet ein Migrationsprozess jedoch die Möglichkeit, Bibliotheken und Betriebssysteme zu aktualisieren. Dadurch können neue Sicherheitsupdates integriert [noat] oder zusätzliche Funktionalitäten genutzt werden.

Bei dem Rehosting-Ansatz treten bezüglich Bibliotheken keinerlei Probleme auf, da die gesamte Struktur auf einer Virtual Machine (VM) installiert wird, in der alle Freiheiten, was genau installiert werden kann, bestehen bleiben. Allerdings bietet dieser Ansatz auch keine Gelegenheit, Bibliotheken auf den neuesten Stand zu bringen. Die Installation des passenden Betriebssystems stellt kein Problem dar, sofern es von Cloud-Providern in der richtigen Version zur Verfügung gestellt wird. Sollte es sich allerdings um eine ältere Version oder um eine Version handeln, die der Cloud Provider nicht zur Verfügung stellt, muss ein custom Image genutzt werden [noau]. Damit ist nicht nur mehr Aufwand verbunden, sondern sind Custom Images auch nicht für die Cloud-Umgebung optimiert und es gibt meist keinen offiziellen Support des Cloud-Anbieters bei Problem oder Fragen [noag].

Der Replatforming-Ansatz unterscheidet sich deutlich vom Rehosting-Ansatz, da er die Möglichkeit bietet, einzelne Teile der Anwendung in Cloud-native Services zu migrieren. Beispielsweise können bestimmte Codeblöcke ohne größeren Aufwand in serverlose Function-as-a-Service (FaaS) Umgebungen übertragen werden. Bei diesen stellt der Cloud-Provider die komplette Infrastruktur zur Ausführung bereit und der Kunde muss nur den Code einfügen, der zu vom Kunden definierten Events ausgeführt wird [noaf]. Ein wesentlicher Vorteil dieses Ansatzes ist die automatische Skalierbarkeit der Funktionen, ohne dass zusätzlicher administrativer Aufwand entsteht. Allerdings kann ein Problem auftreten, wenn spezifische Versionen benötigt werden, die in der serverlosen Umgebung nicht standardmäßig verfügbar sind. Das würde mehr Aufwand für die Code-Umstrukturierung erfordern, bietet aber die Chance, auf eine aktuelle Bibliotheksversion umzusteigen.

Beim Refactoring-Ansatz ist es essenziell, Bibliotheken zu aktualisieren oder neu auszuwählen, damit sie nahtlos in den von Cloud-Anbietern bereitgestellten Umgebungen funktionieren. Da die gesamte Struktur überarbeitet wird, ist dies ohne großen Mehraufwand möglich. Zudem sollte in diesem Schritt darauf geachtet werden, die Abhängigkeit von Windows Servern für den Betrieb zu minimieren, sofern keine Microsoft eigene Software eingesetzt werden soll, da ein performanter und kostengünstiger Betrieb unter Linux in den meisten Cloud-Umgebungen, insbesondere in containerisierten Umgebungen einfacher realisierbar ist [PH18].

Bei dem Repurchasing-Ansatz spielen sowohl die Bibliotheken als auch das Betriebssystem des Legacy-Systems keine Rolle mehr, da der Code selbst nicht migriert wird, sondern primär die Daten in das neue SaaS-System migriert werden.

4.1.5 Identitätsanbieter

Identitätsanbieter (engl.: Identity Provider (IdP)) ist ein Service, der die Verwaltung von Identitäten und die Authentifizierung von Benutzern übernimmt. Sie spielen eine zentrale Rolle, da sie es ermöglichen, eine zentral gespeicherte Identität in verschiedenen Services zu nutzen. Der bekannteste IdP ist Microsoft Active Directory, welcher oftmals On-Premise, aber auch in der Cloud betrieben wird [noar]. Für einen IdP ist es besonders wichtig, zuverlässig und sicher alle nötigen Daten für den Service zur Verfügung zu stellen. Zudem ist eine möglichst kurze Antwortzeit essenziell, um eine optimale Nutzererfahrung zu gewährleisten. [GST12].

Beim Rehosting-Ansatz wird in der Regel eine Verbindung zu einem lokalen IdP hergestellt, um die Migration ohne Änderungen am Code durchführen zu können. Diese Verbindung erfolgt typischerweise über einen VPN-Tunnel, ähnlich wie bei dem nicht migrierten Legacy-System. Ein großer Vorteil dieses Ansatzes ist, dass die sensiblen Daten weiterhin lokal gespeichert werden und andere lokale Dienste weiter darauf zugreifen können, allerdings muss dieser lokal betrieben und somit verwaltet werden. Zudem können aufgrund der räumlichen Entfernung und der zusätzlichen Verbindungsschicht Probleme bei der Performance und Verfügbarkeit auftreten. Eine weitere Option ist es, den IdP neu auf einer VM des Cloud-Providers zu installieren. Diese Lösung bietet eine größere Nähe zu den migrierten Services, ist jedoch mit hohen Betriebskosten verbunden, da eine redundante Infrastruktur erforderlich ist. Ebenso ist die Netzwerkkonfiguration für eine lokale Erreichbarkeit komplex und erfordert zusätzlichen Arbeitsaufwand.

Bei dem Replatforming-Ansatz bietet es sich an, Cloud-native Dienste wie Azure AD, AWS Cognito oder AWS AD zu nutzen und diese mit dem lokalen IdP zu synchronisieren oder, falls der lokale Dienst nicht mehr benötigt wird, vollständig in die Cloud zu migrieren. Dabei sollte ein möglichst ähnliches Produkt mit vergleichbaren Protokollschnittstellen wie bei dem bisherigen On-Premises Legacy-System ausgewählt werden. Beispielsweise sollte, wenn zuvor Microsoft Active Directory genutzt wurde, die Wahl auf Azure AD oder AWS AD fallen, da diese eine einfache Integration in bestehende Software ermöglichen und in der Regel eine Datensynchronisation unterstützen [Jus]. Dieser Ansatz hat erhebliche Vorteile gegenüber dem Rehosting-Ansatz: Zum einen skalieren Cloud-native IdP automatisch mit dem Service, was eine Senkung der Infrastrukturkosten ermöglicht. Außerdem haben sie eine erhöhte Verfügbarkeit, was wichtig bei IdP ist, da viele Services ohne diese nicht genutzt werden können [shl23]. Zudem profitieren Nutzer von Performancevorteilen, da die Verbindungen innerhalb der Infrastruktur des Cloud-Anbieters bleiben. Allerdings erfordert dieser Ansatz auch Vertrauen in den Cloud-Anbieter, da alle Daten bei diesem gespeichert werden.

Der Refactoring-Ansatz ähnelt in seiner Ausrichtung dem Replatforming-Ansatz, legt jedoch einen stärkeren Fokus auf die tiefgreifende Integration von Cloud-nativen Services wie AWS Cognito in die Anwendung. Hierbei sollte gezielt auf modernere Technologien gesetzt werden, die optimal in der Cloud-Umgebung funktionieren, wie zum Beispiel die Umstellung von LDAP auf SAML, da dies anders als LDAP nicht primär für lokale Netze konzeptioniert wurde [noa22] [Dha21]. Besonders bei einer Microservice-Architektur, die häufig aufgrund ihrer guten Integration mit zahlreichen Cloud-Funktionen gewählt wird, können durch die Aufteilung in viele einzelne Services Performanceprobleme entstehen. Diese Performanceprobleme resultieren oft aus der notwendigen Authentifizierung bei jedem einzelnen Request und dem damit verbundenen Overhead der verwendeten Protokolle für die Kommunikation zum IdP. Diese sind nötig, um sicherzustellen, dass die eventuell benötigten Berechtigungen gegeben sind. Diese Implementierung nimmt mehr Zeit als bei den anderen Ansätzen in Anspruch, ermöglicht jedoch in Kombination mit Technologien, die in Cloud-nativen Services genutzt werden, einen Authentifizierungsprozess zu schaffen, der sowohl sicher ist, als auch mit den Services skaliert [JTMH23].

Bei dem Replacing-Ansatz besteht oft die Möglichkeit, den aktuellen IdP weiter bei der SaaS-Anwendung zu nutzen [noaw]. Dies hat den Vorteil, dass der Aufwand der Migration minimal ist und die Daten weiter On-Premise liegen. Allerdings bleibt dadurch die Verfügbarkeit abhängig vom lokalen System, welches weiterhin betrieben werden muss. Eine alternative Option ist die Migration des lokalen IdPs zu einem Cloud-nativen Service. Dies ermöglicht es, die Daten weiterhin selbstständig verwalten zu können und sie für andere Anwendungen nutzbar zu machen, während man gleichzeitig von den Vorteilen eines Cloud-nativen Dienstes profitiert. Der Migrationsaufwand hängt dabei von den Systemen und den bereitgestellten Tools ab. Beispielhaft kann die Migration eines On-Premises Microsoft AD zu Azure AD durch vorhandene Microsoft-Tools vergleichsweise einfach und ressourcenschonend durchgeführt werden [Jus]. Die dritte Option ist es, das komplette Identity-Management an den SaaS-Betreiber abzugeben. Das spart den Aufwand einen IdP betreiben zu müssen, schränkt jedoch die Zugänglichkeit der Daten für andere Anwendungen stark ein. Die letzten beiden Optionen bieten im Hinblick auf Verfügbarkeit und Flexibilität ähnliche Vorteile, da sie von den optimierten und skalierbaren Infrastrukturen der Cloud-Anbieter profitieren.

4.1.6 Hardware

Hardware ist vermutlich die offensichtlichste Abhängigkeit einer Software, da ohne diese keine Software laufen kann. Bei On-Premise-Systemen sind die Anschaffungskosten für Hardware mit den erforderlichen Spezifikationen hoch und der Beschaffungsprozess oft langwierig. Darüber hinaus entstehen laufende Kosten wie Strom, Miete und Internetverbindung, die für den Betrieb erforderlich sind. Zudem wird Personal benötigt, um die Hardware zu warten und im Falle von Ausfällen oder Defekten schnell reagieren zu können [Gai23]. On-Premise-Lösungen erfordern auch eine Reserve an Hardware, um im Fall eines Fehlers Ersatz bereitstellen zu können. Dies erhöht die initialen Kosten und bindet Kapital. Unabhängig der Migrationsstrategie entfällt dieser

Beschaffungsprozess in der Cloud komplett. In der Cloud ist beliebig viel Rechenleistung innerhalb von Sekunden bereitgestellt, ohne dass Initialkosten entstehen [Gai23]. Diese schnelle und flexible Bereitstellung ist ein wesentlicher Vorteil der Cloud. In der Cloud erfolgt die Abrechnung nach einem nutzungsbasierten Modell, bei dem die Kosten beispielsweise pro Stunde oder pro genutzter Ressource anfallen [AG17]. Das hat zur Folge, dass die Initialkosten zwar nicht vorhanden sind, allerdings die laufenden Kosten im Vergleich zu On-Premise-Lösungen höher sein können. Was sich bei den verschiedenen Migrationsstrategien ändert, ist das Verhältnis zur Hardware und die Abschätzbarkeit dieser Kosten.

Durch die ausschließliche Nutzung von VMs bei dem Rehosting-Ansatz besteht noch eine genauso enge Bindung an die Hardware wie zuvor. Das heißt, dass jede Software, die isoliert läuft, genau die Rechenleistung zur Verfügung hat, die die darunter liegende VM zur Verfügung stellt. Allerdings ist es bereits bei diesem Ansatz möglich, diese zur Verfügung stehende Hardware hoch- und runterzuskalieren, allerdings nicht ohne Downtime oder redundante Instanzen hinter einem Loadbalancer [Bra]. Ein Vorteil des Rehosting-Ansatzes ist die einfache Kostenabschätzung. Die bestehende Hardware-Infrastruktur wird in passende VM-Konfigurationen des Cloud-Anbieters übersetzt, wodurch die monatlichen Kosten berechenbar sind. Wenn Autoscaling genutzt wird, dienen die kalkulierten Kosten als Worst-Case-Szenario.

Beim Replatforming-Ansatz verhält sich der Teil der Infrastruktur, der nicht in Cloud-native Services migriert wird, genauso wie beim Rehosting. Für die in Cloud-native Services überführten Komponenten ist die Bindung an spezifische Hardware jedoch stark reduziert. Cloud-native Lösungen wie beispielhaft AWS der Service ECS mit Fargate erlauben den Betrieb von Container in einer Serverless Umgebung. Hier wird ausschließlich für den tatsächlichen Verbrauch von CPU und RAM bezahlt [HBS21]. Diese Flexibilität erschwert die Kostenabschätzung im Vergleich zum Rehosting-Ansatz, allerdings gibt es Tools wie den AWS Pricing Calculator oder den Google Cloud-Preisrechner, um diese Abschätzung zu vereinfachen. Nach der Migration können außerdem Vorhersagen über zukünftige Kosten genutzt werden, die der Cloud-Provider zur Verfügung stellt [noan].

Beim Refactoring-Ansatz wird das gesamte System auf Cloud-native Services umgestellt, was die Kopplung an spezifische Hardware nahezu vollständig auflöst. Bei Services wie dem Elastic Kubernetes Service (EKS) bleibt nur eine stark abstrahierte Hardware-Kopplung bestehen. Bei der Nutzung von serverless Computing wie AWS Lambda oder Google Cloud Functions ist die Kopplung zu Hardware nicht mehr vorhanden und somit spielt ein tatsächlicher Server keine Rolle mehr. Das hat abgesehen von der Flexibilität den Vorteil, dass sich bei einem Deployment keine Gedanken über Leistung oder Anzahl der Server gemacht werden müssen. Diese maximale Flexibilität hat jedoch ihren Preis: serverless-Dienste sind bei hoher Last oft teurer als andere Cloud-Optionen [FJG20]. Viele Cloud-Anbieter bieten an, Rechenleistung zu einem günstigeren Preis und für einen festen Zeitraum zu reservieren. Die Variante schränkt die Flexibilität ein, da die reservierte Kapazität immer bezahlt werden muss, senkt dafür aber bei richtiger Kalkulation die Infrastrukturkosten [noam]. Die Kostenabschätzung bei serverless Computing bleibt jedoch schwierig, da die tatsächlichen Kosten stark von der Nutzung abhängen. Durch das Reservieren von Kapazität ist eine Kostenabschätzung einfach und zuverlässiger möglich.

Bei dem Replacing-Ansatz liegt die Hardware komplett in der Verantwortung des SaaS-Betriebs. Die Kosten für die Hardware sind meist in den allgemeinen Lizenzkosten für die Software mit inbegriffen und nicht extra aufgeführt. Das vereinfacht die Kostenplanung bei dem Replacing-Ansatz in Hinsicht auf Infrastrukturkosten erheblich.

4.2 Prozessuale Abhängigkeiten

Prozessuale Abhängigkeiten umfassen alle notwendigen Prozesse, um das Softwaresystem zuverlässig und effizient zu betreiben. Darunter fallen insbesondere Prozesse, die direkt den Betrieb der Software sicherstellen, wie Monitoring oder Incident-Management-Prozesse. Ohne solche Prozesse, die potenzielle Störungen frühzeitig erkennen und schnell darauf reagieren, könnten Ausfälle unbemerkt bleiben, was im schlimmsten Fall den Geschäftsbetrieb beeinträchtigen würde. Neben den Prozessen, die direkt am Betrieb beteiligt sind, gibt es auch Prozesse, die nur indirekt für den aktuellen Betrieb verantwortlich sind, wie das Change Management oder Risikomanagement. Diese Prozesse arbeiten meist den aktiv am Betrieb beteiligten Prozessen zu und sorgen für eine langlebige Software [Llo13] [Gro10].

4.2.1 Backup- und Wiederherstellungsprozesse

Ein robuster Backup- und Wiederherstellungsprozess ist essenziell, um im Falle eines Datenverlusts die Daten und damit den Betrieb einer Anwendung möglichst schnell und verlustfrei wiederherzustellen. Datenverlust kann durch verschiedene Ursachen wie Hackerangriffe, menschliches Versagen oder technische Defekte entstehen. Insbesondere die Zunahme von Ransomware-Angriffen in den vergangenen Jahren unterstreicht die Notwendigkeit einer sicheren Backupstrategie [ste23] [noav]. Zudem kann es auch gesetzliche- oder Kundenvorgaben bezüglich der Häufigkeit und Aufbewahrungsdauer von Backups geben, wie bei Finanzberichten oder Steuerunterlagen [noaa].

Da bei dem Rehosting-Ansatz die gesamte Infrastruktur übernommen wird, kann auch der Backupprozess übernommen werden. Allerdings ergeben sich hier nur minimale Vorteile gegenüber einer On-Premise-Lösung. Zwar können die Flexibilität und das Pay-as-you-go-Modell des Cloud-Speichers genutzt werden, allerdings ist der Aufwand der Backup-Erstellung identisch zum On-Premise System [LDL+16]. Da alle Services in VMs migriert wurden, sind die Backup- und Wiederherstellungsprozesse nur in bestimmten Fällen vereinfacht. Von den VMs lässt sich in der Regel ohne großen Aufwand ein vollständiges Backup des Dateisystems erstellen. Die Wiederherstellung mithilfe dieser Methode ist ebenfalls äußerst unkompliziert und erfordert nur minimalen Aufwand [noac]. Allerdings ist diese Art des Backups per Snapshots nicht für große Datenmengen geeignet, da diese sehr langsam sein können [noax]. Insbesondere bei Datenbanken ist diese Art des Backups ungeeignet, da ein Backup über die spezifischen Mechanismen der Datenbank deutlich effizienter ist und wesentlich kleinere Datenmengen erzeugt [noa23b] [noa21a]. Daher ist der Migrationsprozess insbesondere für die relevanten Services kaum optimiert.

Beim Replatforming-Ansatz können Backup- und Wiederherstellungsprozesse durch die teilweise Nutzung Cloud-nativer Dienste erheblich optimiert werden. Diese Optimierung wird ermöglicht, da viele Cloud-native Dienste bereits integrierte Mechanismen für die Erstellung und Verwaltung von Backups bieten [noay]. Da bei dem Replatforming-Ansatz oft die Datenbank in einen Cloud-nativen Service migriert, entfällt das Problem des Datenbankbackups gegenüber dem Rehosting-Ansatz. Die Herausforderung hierbei ist, alle Backupprozesse der verschiedenen Services zu verwalten und bei einem Problem möglichst schnell eine Wiederherstellung einleiten zu können.

Beim Refactoring-Ansatz wird der Backupprozess durch den Einsatz zahlreicher zustandsloser Komponenten, etwa in einem Kubernetes-Cluster oder durch die Nutzung von FaaS erheblich vereinfacht, da die Daten an weniger zentralisierten Orten gespeichert sind. Die enge Cloud-Integration ermöglicht es zudem, auf fertige Backup-Lösungen des Cloudanbieters zurückzugreifen, wie beispielsweise AWS Backup. Hierbei müssen lediglich die relevanten Services ausgewählt sowie, Zeitpläne und Speicherfristen definiert werden. Den Rest übernimmt der Cloud-Provider, was eine einmalige Einrichtung ohne weiteren Verwaltungsaufwand erlaubt [noay]. Das ermöglicht die einmalige Einrichtung des Prozesses, ohne diesen weiter verwalten zu müssen.

Beim Replacing-Ansatz liegt die Verantwortung für Backups primär beim Anbieter der SaaS. Dadurch entfällt die Notwendigkeit, einen eigenen Backup-Prozess zu haben. Allerdings kann es sinnvoll sein, insbesondere bei rechtlichen Vorgaben, eine eigene Sicherung der Daten vorzunehmen. Der Aufwand hängt von der Software ab, kann aber, sofern die Daten zugänglich sind, meist mit den gleichen Methoden wie bei dem Refactoring-Ansatz automatisiert werden.

4.2.2 Monitoring

Eine robuste Monitoring-Infrastruktur ist für große Systeme von entscheidender Bedeutung, um Fehler, Ausfälle und Performanceprobleme frühzeitig zu erkennen. Das ist in der Cloud nicht anders als bei einem On-Premise Betrieb, vielmehr gewinnt Monitoring zusätzlich an Bedeutung, da auch die Kostenüberwachung eine zentrale Rolle spielt, unter anderem um kostenintensive Services zu identifizieren und deren Kosten zu reduzieren [WB14]. Dabei ist es essenziell, dass die Monitoring-Systeme möglichst ressourcenschonend und kostengünstig arbeiten, aber trotzdem eine hohe Erreichbarkeit haben. Gleichzeitig sollten sie flexibel sein und eine breite Palette an Services überwachen können, um sämtliche relevanten Daten zentralisiert zu erfassen [ABDDP13].

Beim Rehosting-Ansatz wird das bestehende Monitoring-System weiterverwendet, da die Infrastruktur unverändert bleibt. Diese Herangehensweise hat jedoch einige Nachteile. Beispielsweise müssen alle zu überwachenden Server und VMs manuell zu dem alten System hinzugefügt werden. Zwar ermöglicht das Legacy-Tool eine einfache Überwachung der Komponenten des alten Systems, allerdings müssen die neuen Bestandteile, die bei dem Rehosting-Ansatz hinzukommen, sofern möglich, eingebunden oder in einem anderen Tool überwacht werden. Dazu zählen beispielsweise die laufenden Kosten für Hardware oder neue Netzwerkelemente. Für die Überwachung der vom Cloud-Provider verwalteten Ressourcen bietet der Provider in der Regel eigene Monitoring-Tools an [noae]. Diese erlauben es, ohne großen Mehraufwand wichtige Kennzahlen wie den Ressourcenverbrauch von VMs, den Netzwerkverkehr und andere kritische Daten zu überwachen. Die Verwendung von zwei Monitoring-Systemen beschleunigt zwar den Migrationsprozess, hat aber den Nachteil, dass das Legacy-System weiter betrieben werden muss, was aufgrund der hohen Anforderungen an Verfügbarkeit sowohl aufwendig als auch kostspielig ist. Darüber hinaus erfordert die parallele Nutzung zweier Systeme zusätzliche Arbeit, da Informationen aus beiden Quellen zusammengeführt werden müssen, um ein vollständiges Bild zu erhalten.

Bei dem Replatforming-Ansatz verhält es sich ähnlich wie bei dem Rehosting-Ansatz. Da nur einige Teile in Cloud-native Services migriert werden, müssen die verbleibenden Legacy-Komponenten weiterhin über das alte Monitoring-System überwacht werden. Gleichzeitig können jedoch zentrale Elemente wie eine Datenbank und andere Metriken in einem Cloud-nativen Dienst überwacht werden [noae]. Dadurch verliert das Legacy-System an Bedeutung, bleibt aber beispielsweise für die Log-Überwachung weiterhin unverzichtbar. Falls die Log-Überwachung nicht besonders komplex ist, könnte sie ebenfalls in einen Cloud-nativen Service wie AWS CloudWatch Logs migriert werden [noaz]. Ist dies nicht möglich bleibt das Problem bestehen, dass zwei Monitoring-Systeme parallel betrieben werden müssen. Dies erhöht den Aufwand und verursacht zusätzliche Kosten. Allerdings vereinfacht sich die Migration, da das vom Cloud-Provider bereitgestellte Monitoring-System die relevanten Daten automatisch erfasst und das Legacy-System wird nur noch für wenige Aufgaben benötigt. Dadurch reduziert sich der Arbeitsaufwand bei der Migration erheblich.

Beim Refactoring-Ansatz wird das Legacy-Monitoring-System vollständig auf das System des Cloud-Anbieters umgestellt. Der Cloud-Anbieter übernimmt dabei die Bereitstellung aller relevanten Metriken, wie Informationen zur Systemauslastung oder zur Anzahl der Verbindungen über einen Loadbalancer [noah]. Dies reduziert den Arbeitsaufwand im Vergleich zu herkömmlichen Monitoring-Tools erheblich. Im Rahmen der Migration müssen lediglich Schwellenwerte definiert werden, bei deren Erreichen automatisch Aktionen ausgelöst werden, wie etwa das Versenden von Benachrichtigungen oder das Hochfahren neuer Ersatzrechner. Neu erstellte Services werden meist automatisch oder mit minimalem Aufwand in das Monitoring-System integriert, wodurch der

Betriebsaufwand für das Monitoring-System minimal bleibt und sich somit auf das Kerngeschäft konzentriert werden kann. Zudem ist es durch Nutzung standardisierter Containerumgebungen und Serverless Functions möglich, Logs automatisiert zu sammeln und auszuwerten. Für diesen Zweck stellt der Cloud-Anbieter spezifische Funktionen bereit. Der Einsatz zahlreicher kostenloser Tools sowie die Bezahlung zusätzlicher Funktionen nach dem „Pay-as-you-go“-Modell machen diesen Ansatz nicht nur kostengünstig, sondern auch äußerst flexibel [noab].

Beim Replacing-Ansatz übernimmt der SaaS-Anbieter die vollständige Verantwortung für das Monitoring der Anwendung. Bei selbst gehosteten Anwendungen wird häufig ein eigenes Monitoring-Tool mitgeliefert oder ein Cloud-natives Tool genutzt, das die zuvor beschriebenen Vorteile bietet.

4.2.3 Incident-Management

Incident-Management bezeichnet die Vorbereitung auf Probleme, die während des Betriebs auftreten können, sowie die Maßnahmen, um diese so schnell wie möglich zu beheben. Das National Institute of Standards and Technology (NIST) unterteilt das Incident-Management in die folgenden Schritte: Vorbereitung, Erkennung und Analyse, Eindämmung, Beseitigung und Wiederherstellung sowie Nachbereitung [CMGS12]. Die Migration in die Cloud kann insbesondere bei der Erkennung und Analyse sowie bei der Eindämmung, Beseitigung und Wiederherstellung Vorteile bieten, insbesondere in Bezug auf Automatisierung und die Möglichkeit eines zentralisierten und gemeinsamen Arbeitens [PHGK21].

Beim Rehosting-Ansatz kann das Incident Management nur minimal optimiert werden, da weiter das Legacy Montierung System genutzt wird. Diese ermöglicht zwar teilweise die Weiterverwendung der bestehenden Incident-Erkennung, was den Migrationsprozess vereinfacht, jedoch weder wesentliche Verbesserungen noch Einsparungen an Zeit und Kosten bringt. Bezüglich der Beseitigung von Incidents müssen die bisherigen Pläne überarbeitet werden. Dennoch bietet selbst diese grundlegende Nutzung der Cloud Vorteile bei der Beseitigung und Wiederherstellung, da alle Ressourcen zentral über das Web-Interface des Cloud-Providers eingesehen und verwaltet werden können.

Beim Replatforming-Ansatz können durch die erweiterte Nutzung Cloud-nativer Dienste nur minimale Verbesserungen gegenüber dem Rehosting-Ansatz, insbesondere bei der Erkennung, erzielt werden. Zwar wird beim Replatforming-Ansatz bereits verstärkt auf Cloud-native Monitoring-Tools gesetzt, jedoch wird in vielen Fällen weiterhin das Legacy-Monitoring-System verwendet. Dies führt dazu, dass kein System ein vollständiges Bild liefert, was jedoch für eine effektive Incident-Erkennung essenziell ist. Im Allgemeinen ist daher in Bezug auf die Erkennung eher ein Rückschritt im Vergleich zu einem Rehosting-Ansatz zu verzeichnen. Hinsichtlich der Beseitigung und Wiederherstellung verhält es sich bei dem Replatforming-Ansatz ähnlich zum Rehosting-Ansatz. Allerdings können geringfügige Vorteile durch die verbesserten Backup- und Wiederherstellungsprozesse des Replatforming-Ansatzes erzielt werden, die für eine schnelle Wiederbereitstellung wichtig sind.

Eine deutliche Verbesserung kann durch den Refactoring-Ansatz erreicht werden. Durch die ausschließliche Nutzung Cloud-nativer Dienste kann ein speziell auf die Umgebung abgestimmtes Cloud-natives Monitoring-Tool eingesetzt werden [BK11]. Zudem stehen Tools zur Verfügung, um die gesamte Umgebung, einschließlich aller Netzwerkelemente, zu visualisieren und somit den Verlauf eines Incidents präzise nachzuverfolgen [Wiz]. Diese Visualisierung kann vollständig automatisiert generiert werden, da alle verwendeten Komponenten standardisierte Cloud-native Services sind. Zwar ist eine solche automatisierte Visualisierung auch bei den beiden vorherigen Ansätzen möglich, jedoch kann ihr volles Potenzial erst bei der Nutzung zahlreicher Cloud-nativer Services, wie bei dem Refactoring-Ansatz, ausgeschöpft werden. Zudem können auf alle Ressourcen über standardisierte Schnittstellen des Cloud-Providers zugegriffen werden, was die Erstellung einfacher Skripte ermöglicht, die im Falle eines Incidents standardisierte Aufgaben schnell ausführen können [noa23a]. Allerdings sind diese Tools zur Erkennung notwendig, da die Fehlersuche in einer Cloud-Umgebung aufgrund vieler Abhängigkeiten weiterhin komplex sein kann. Dementsprechend ist eine Schulung der Mitarbeiter bei einer solchen Nutzung der Cloud wichtig.

Beim Replacing-Ansatz ist meist der SaaS-Betreiber für das Incident Management verantwortlich. In diesem Fall ist es wichtig, eine gute Kommunikation mit dem Betreiber zu pflegen, um effektiv mit ihm zusammenzuarbeiten oder rechtzeitig zu erfahren, wann das System wieder erreichbar ist bzw. welche Einschränkungen bestehen. Sollte das Incident-Management nicht in der Verantwortung des SaaS-Betreibers sein, ist die Bereitstellung aller nötigen Metriken wichtig, um diese in die eigenen Überwachungstools zu integrieren [GW09].

4.2.4 Change Management

Anders als in den Kapiteln zuvor, bei denen die Ergebnisse und die dadurch hergeführten Vor- und Nachteile der Ergebnisse der verschiedenen Migrationsansätze verglichen wurden, werden in diesem Abschnitt die nötigen Schritte des Change Managements beschrieben, die nötig sind, um eine weitreichende Akzeptanz nach der Migration zu erhalten und mit welchem Aufwand diese verbunden sind. Bei dem Change Management gibt es diverse Aspekte zu betrachten. Da laut [LC06] menschliche Faktoren mit 57% der häufigste Grund sind, dass Änderungen fehlschlagen, werden im Weiteren nur diese betrachtet. Dabei gibt es laut [LS08] folgende acht Gründe, warum Änderungen nicht gewollt sind und dadurch fehlschlagen können:

1. Mangelndes Vertrauen
2. Überzeugung, dass Veränderungen unnötig sind
3. Überzeugung, dass Veränderungen nicht durchführbar sind
4. Wirtschaftliche Bedrohungen
5. Relativ hohe Kosten
6. Furcht vor persönlichem Versagen
7. Verlust von Status und Macht
8. Bedrohung von Werten und Idealen
9. Unmut über Einmischung

Um den Umfang dieses Kapitels übersichtlich zu halten, wird im Weiteren nur auf 4 Wirtschaftliche Bedrohungen und 7 Verlust von Status und Macht eingegangen. Diese Gründe sind oft mit den Ängsten der Mitarbeitenden verbunden und führen häufig zu Widerstand gegenüber Veränderungen [LS08]. Dabei ist die Angst vor dem Verlust von Status und Macht primär bei dem Personal, welches im Betrieb und in der Architektur beschäftigt ist, zu finden.

Beim Rehosting-Ansatz kann die Angst vor wirtschaftlichen Bedrohungen besonders gut eingegrenzt werden, da sich die zukünftigen Betriebsausgaben für die nächsten Jahre leicht berechnen lassen. Zudem ist der Migrationsaufwand abschätzbar, da nur wenige Änderungen vorgenommen werden. Diese Angst kann somit durch eine klare und leicht nachzuvollziehende Kostenrechnung weitestgehend genommen werden. Auch die Angst vor dem Verlust von Status und Macht lässt sich bei diesem Ansatz relativ einfach reduzieren, da ein Großteil der Verantwortung im Unternehmen bleibt [LSA17]. Zudem kann die Migration meist ohne externes Personal durchgeführt werden, da das nötige Wissen meist schon vorhanden ist oder durch kurze Schulungen gewonnen werden kann.

Beim Replatforming-Ansatz verhält es sich in Bezug auf die Angst vor wirtschaftlichen Bedrohungen ähnlich wie beim Rehosting-Ansatz. Allerdings kann die Angst vor Verlust von Status und Macht, beispielsweise bei den Verantwortlichen für die Datenbank, stärker ausgeprägt sein. Dies liegt daran, dass bei dem Replatforming-Ansatz Datenbanken in Cloud-nativen Dienste migriert werden, was in diesem Bereich zu erheblichen Veränderungen führt. Diese Angst kann jedoch durch gezielte Schulungen eingegrenzt werden. Da jedoch viele der alten Systeme weiterhin bestehen bleiben, ist

die Angst vor Status- und Machtverlust zwar ausgeprägter als beim Rehosting-Ansatz, kann aber auch durch geeignete Maßnahmen effektiv eingegrenzt werden.

Beim Refactoring-Ansatz sind sowohl die Ängste vor wirtschaftlichen Bedrohungen als auch vor dem Verlust von Status und Macht am stärksten ausgeprägt und zugleich am schwierigsten einzugrenzen. Da es sich um eine extrem große Veränderung handelt, ist die Kostenkalkulation schwieriger, wodurch wirtschaftliche Bedrohungen nur schwer durch einfache Fakten entkräftet werden können. Umfangreiche Schulungen können die Angst vor dem Verlust von Status und Macht reduzieren, da diese die Kompetenz der Mitarbeitenden während und nach der Migration stärken. Allerdings erfordert eine große Migration oft Unterstützung durch externe Dienstleister oder die Einstellung neues Personals für den Cloud-Betrieb. Dies kann interne Konkurrenzkämpfe auslösen und die Stimmung gegen die Cloud verstärken.

Beim Replacing-Ansatz ist die Angst vor einer wirtschaftlichen Bedrohung in der Regel weniger ausgeprägt oder kann relativ einfach ausgeräumt werden. Dies liegt daran, dass die Kosten über mehrere Jahre meist fest und somit einfach zu kalkulieren sind. Je nach Vertrag können diese Kosten sogar flexibel reduziert werden, zum Beispiel durch Anpassung der Anzahl benötigter Lizenzen. Die Angst vor dem Verlust von Status und Macht ist hingegen schwerer zu mindern, da ein Großteil der Arbeit aus dem Unternehmen ausgelagert wird, was Arbeitsplätze gefährden kann. Zudem handelt es sich häufig um ein vollständig neues System, in das sich alle Mitarbeitenden erst einarbeiten müssen. Besonders Experten für die alten Systeme könnten befürchten, ihre bisherige Stellung zu verlieren.

4.2.5 Risikomanagement und Verantwortung

Das Risikomanagement dient dazu, potenzielle Risiken, die die Geschäftstätigkeit beeinträchtigen könnten, zu identifizieren, zu bewerten und zu vermeiden [BGL08]. Es ist nicht nur im On-Premise-Betrieb essenziell, sondern auch bei einem Betrieb in der Cloud unverzichtbar. Das grundlegende Vorgehen im Risikomanagement unterscheidet sich zwischen On-Premise und Cloud-Betrieb nicht wesentlich, da viele Risiken beider Systeme ähnlich sind [DB15]. Allerdings verschiebt sich die Verantwortung über bestimmte Teile des Systems, sodass der Cloud-Provider für einige Bestandteile die Verantwortung übernimmt und somit auch Teile des Risikomanagements an den Cloud-Provider abwandern [LSA17]. Diese Verlagerung kann die Migration des bestehenden Risikomanagements erleichtern und unterstützt dabei, es an das neue, dynamische Umfeld der Cloud anzupassen. Im Folgenden werden die verschiedenen Stufen der Verantwortungsübernahme näher betrachtet.

Bei dem Rehosting-Ansatz wird die Software meist auf IaaS-Systemen migriert, darunter zählt zum Beispiel EC2 von AWS oder GCE. Dabei übernimmt der Cloud-Provider die Verantwortung für die zugrunde liegende Hardware, die grundlegende Bereitstellung der Netzwerkinfrastruktur sowie die Verwaltung und den Betrieb der Rechenzentren, in denen die Hardware untergebracht ist [LSA17]. Allein durch die Abgabe der Hardwareverantwortung wird einiges an Risiko genommen, da 36% aller Systemausfälle auf Hardwareausfälle zurückzuführen sind [noaj]. Dadurch verringert sich auch der Aufwand für das Management verbleibender Risiken.

Beim Replatforming-Ansatz bleibt für alle Komponenten, welche nicht in Cloud-native Dienste migriert werden, die Risikoverteilung identisch zum Rehosting-Ansatz. Allerdings werden bei diesem Ansatz oft kritische Teile des IT-Systems, wie Datenbanken, in Cloud-native Dienste migriert. Dabei sind diese essenziell, da sie nicht nur für den Betrieb unverzichtbar sind, sondern auch sensible und unverzichtbare Daten verwalten. Die Migration einer Datenbank in ein PaaS-System führt zu weiterer Verlagerung der Verantwortung an den Cloud-Provider [LSA17]. Allerdings wird im Gegensatz zu einem IaaS-System primär nur noch die Verantwortung über die Installation und Konfiguration des Betriebssystems und zum Teil der Software genommen. Da eine fehlerhafte Konfiguration für 64% aller Ausfälle verantwortlich ist, verbessert dies das Risikomanagement in diesen Teilen. [noaj]. Zusammengefasst vereinfacht der Replatforming-Ansatz nicht signifikant oder nur in bestimmten Teilen das Risikomanagement gegenüber dem Rehosting-Ansatz.

Beim Refactoring-Ansatz wird nahezu das gesamte System in IaaS-Services betrieben, was bereits die zuvor beschriebenen Vorteile mit sich bringt. Da das System vollständig neu strukturiert wurde und auf einem flexibel konfigurierbaren IaaS-Services läuft, ergibt sich zudem die Möglichkeit, Infrastructure as Code (IaC) zu nutzen. IaC ermöglicht die komplette Konfiguration des Systems über Code oder Config-Dateien, welche mittels eines Befehls auf die Cloud angewandt werden [noa21b]. Das hat den großen Vorteil, dass fehlerhafte Konfigurationen automatisiert erkannt werden können, bevor sie in der Cloud umgesetzt werden. Zudem können Konfigurationsfehler schneller behoben werden, was das nötige Risikomanagement für eine Änderung erheblich vereinfacht.

Beim Replacing-Ansatz erfolgt die Migration meist zu einem SaaS-Anbieter, wobei ein Großteil der Verantwortung an den Betreiber der SaaS-Software übertragen wird. Dabei ist es wichtig, dass der SaaS-Anbieter ein angemessenes Risikomanagement transparent darlegen kann, insbesondere wenn es sich bei der Anwendung um einen kritischen Dienst handelt. Diese Anforderungen werden in der Regel in einem Service-Level-Agreement (SLA) festgehalten, um die Einhaltung von Standards und Verantwortlichkeiten zu gewährleisten [noa21b]. Dies ermöglicht es, diese Anwendung sehr einfach in das bestehende Risikomanagement zu übernehmen.

5 Vergleich der Migrationsansätze

5.1 Skalierbarkeit und Verfügbarkeit

Unabhängig von der gewählten Methode profitiert insbesondere die Hardware von der verbesserten Skalierbarkeit in der Cloud. Die höchste Skalierbarkeit wird jedoch bei dem Refactoring-Ansatz und dem Replacing-Ansatz sowohl bei der Datenbank als auch bei der Hardware erreicht, gefolgt von dem Rehosting Ansatz. Mit steigender Skalierbarkeit nehmen jedoch auch die Einschränkungen durch Legacy-Systeme zu. Wenn ein Legacy-System weiterhin benötigt wird, treten die größten Einschränkungen beim Refactoring- und Replacing-Ansatz auf.

Ähnlich verhält es sich auch mit der Verfügbarkeit. Verbesserte Verfügbarkeit kann bei der Netzwerkinfrastruktur unabhängig vom Migrationsansatz erreicht werden. Auch bei der Datenbank und der Hardware ist eine erhöhte Verfügbarkeit möglich. Allerdings lässt sich diese bei Verwendung des Refactoring-Ansatzes einfacher umsetzen und bleibt stabiler als bei Rehosting oder Replatforming.

5.2 Migrationsaufwand

Der Migrationsaufwand ist beim Rehosting-Ansatz am geringsten. Dies liegt vor allem daran, dass viele bestehende Strukturen und Systeme direkt übernommen werden können. Beispielsweise lassen sich auf prozessualer Ebene das Monitoring sowie das Incident-Management weitgehend unverändert übernehmen. Auf technischer Seite können sowohl die Datenbank als auch ein Großteil der Netzwerkinfrastruktur beibehalten werden. Zudem ist die Wahrscheinlichkeit von Kompatibilitätsproblemen, etwa bei Bibliotheken und Betriebssystem, geringer. Auch die Integration von Identitätsanbietern und Legacy-Systemen gestaltet sich einfacher.

Beim Replatforming-Ansatz steigt der Migrationsaufwand bei den technischen Abhängigkeiten nur gering und bleibt bei den prozessualen Abhängigkeiten, ähnlich wie bei dem Rehosting-Ansatz. Die primäre Erhöhung des Migrationsaufwands kommt von der Migration der Datenbank und die damit verbundenen Umstellungen der prozessualen Abhängigkeiten, wie bei den Backup- und Wiederherstellungsprozessen gezeigt.

Im Gegensatz dazu ist der Aufwand beim Refactoring deutlich höher. Alle technischen sowie einige prozessualen Abhängigkeiten benötigen erheblich mehr Aufwand als bei Datenbanken, Bibliotheken und Betriebssystem, Legacy-Systemen, Change Management, Netzwerkinfrastruktur gezeigt. Dabei stellt die Migration der Daten den größten Aufwand dar. Daher hängt der Migrationsaufwand im Wesentlichen davon ab, welche fertigen Tools zur Verfügung stehen und wie viel Arbeit damit die Migration ist. Zudem ist beim Refactoring-Ansatz mit einem höheren Arbeitsaufwand im Change Management zu rechnen, da die Veränderungen sehr umfangreich sind. Darüber hinaus kann es notwendig sein, das Incident-Management für den Service neu zu gestalten, was ebenfalls einen erheblichen Zeitaufwand mit sich bringen kann.

5.3 Flexibilität

Die Flexibilität verhält sich beim Refactoring- und Replacing-Ansatz ähnlich: Sowohl bei Datenbank, Hardware, Netzwerkinfrastruktur als auch beim Monitoring kann die Flexibilität erheblich gesteigert werden. Besonders bei der Hardware ist der Unterschied zum Refactoring-Ansatz minimal, sodass der zusätzliche Aufwand beim Refactoring in Bezug auf die Flexibilität von Hardware kaum einen nennenswerten Vorteil bietet. Besonders der Backup- und Wiederherstellungsprozesse können durch die Flexibilität beim verfügbaren Speicherverbrauch bereits beim Refactoring-Ansatz profitieren. Allerdings wird das volle Potenzial der Cloud erst beim Replatforming-Ansatz ausgeschöpft.

Allerdings ist Flexibilität beim Rehosting- und Replatforming-Ansatz durch die verwendeten Bibliotheken und Betriebssystem eingeschränkt. Es sollte stets geprüft werden, ob die gesteigerte Flexibilität im täglichen Geschäft tatsächlich einen Mehrwert bietet und den zusätzlichen Migrationsaufwand rechtfertigt.

5.4 Verwaltungsaufwand und Infrastrukturkosten

Der Verwaltungsaufwand ist proportional zum Grad der übernommenen Verantwortung, wie unter Incident-Management beschrieben. Dies wird auch durch das Vorgehen bei der Netzwerkinfrastruktur und der Datenbank untermauert, bei denen der Migrationsaufwand relativ hoch ist. Sobald die Systeme einmal bei dem Cloud-Provider laufen, müssen sie meist nur noch minimal verwaltet werden.Besonders bei der Hardware wird ein Großteil des Verwaltungsaufwands an den Cloud-Provider abgegeben. Zudem können die Tools, die unter Incident-Management, Backup- und Wiederherstellungsprozesse und Monitoring aufgeführt sind, dazu beitragen, die Komplexität der neuen Umgebung zu verringern und nach einer Einarbeitungsphase den Verwaltungsaufwand zu senken.

Infrastrukturkosten sind, wie unter Change Management beschrieben, oft schwer vorherzusagen. Allerdings fallen sie beim Refactoring-Ansatz kurzfristig häufig niedriger aus, wie in Hardware beschrieben. Generell führt die verstärkte Nutzung Cloud-nativer Dienste dazu, dass die Kosten des Cloud-Anbieters im Vordergrund stehen, während prozessuale Betriebskosten durch automatisierte Prozesse eingespart werden können, wie bei Backup- und Wiederherstellungsprozesse, Incident-Management und Monitoring gezeigt. Allerdings erfordert die Migration in diesem Szenario mehr Arbeitsaufwand. Daher ist es essenziell, eine möglichst präzise Kostenabschätzung über mehrere Jahre hinweg vorzunehmen, um fundiert entscheiden zu können, welche Methode am besten geeignet ist.

6 Praxisbeispiele

Im folgenden Abschnitt wird exemplarisch dargestellt, wie die Migration eines Microsoft Exchange Servers von einer On-Premise-Installation in die Cloud aussehen könnte und wie verschiedene Abhängigkeiten das Endergebnis beeinflussen können. Microsoft Exchange Server ist ein E-Mail-Transportserver mit integrierter Groupware-Funktionalität. Er dient zum Senden und Speichern von E-Mails sowie zur Verwaltung von Terminen und Kontakten [noa23c]. Dabei wird Microsoft Active Directory (AD) für die Verwaltung und Authentifizierung der Nutzer eingesetzt.

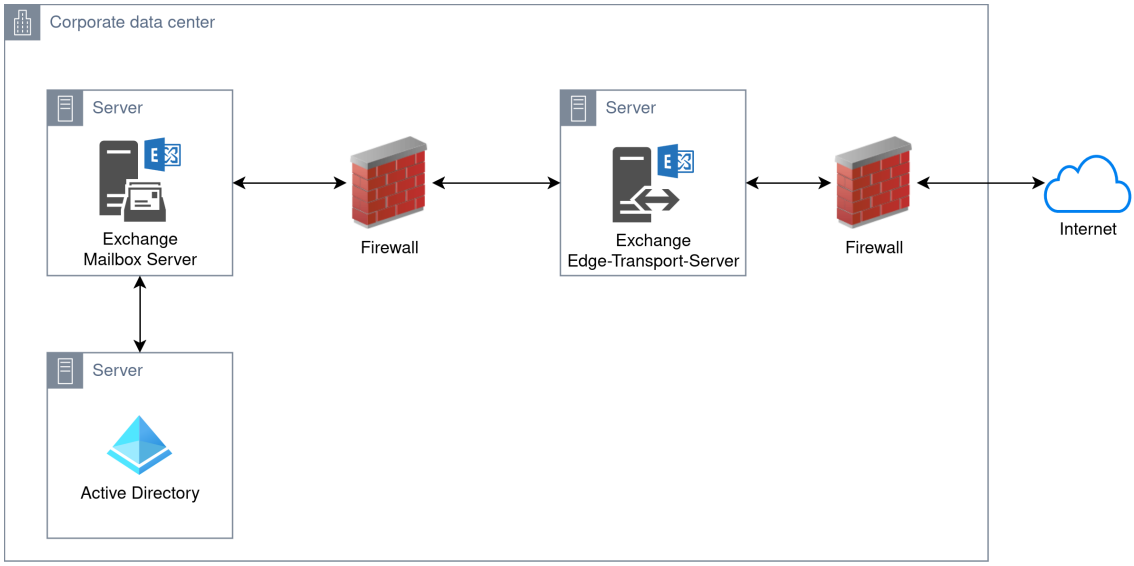

Abbildung 6.1: Exchange On-Premis Architecture

Das On-Premise-System besteht aus einem Loadbalancer, der aus dem Internet erreichbar ist. Dieser fungiert als Firewall, nimmt alle eingehenden Verbindungen an und leitet sie weiter. Hinter dem Loadbalancer befindet sich ein Microsoft Exchange Edge-Transport-Server, der alle eingehenden E-Mails empfängt und an redundante Exchange-Server weiterleitet. Die Exchange-Server beinhalten auch die Datenbank, in der sämtliche E-Mails gespeichert werden. Die Verwaltung von Identitäten und die Authentifizierung für die verschiedenen Schnittstellen des Exchange-Servers (z. B. Web-UI, SMTP) erfolgen über ein Microsoft Active Directory, das darüber hinaus für die zentrale Verwaltung aller lokalen Rechner genutzt wird.

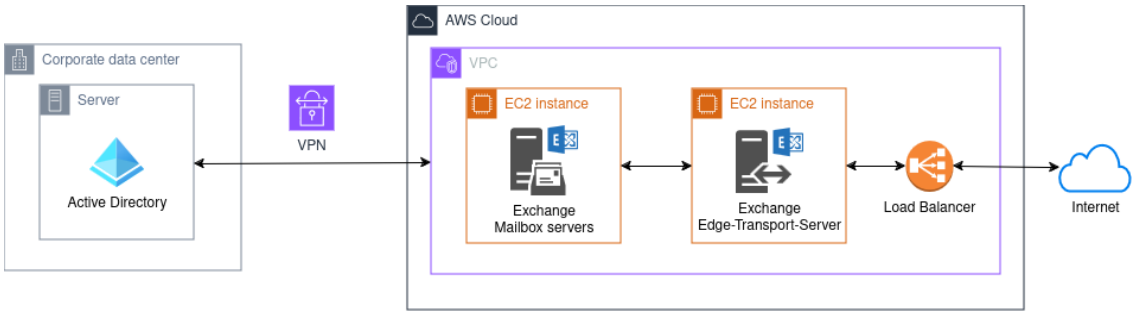

Abbildung 6.2: Exchange Rehosting Architecture

Beim Rehosting-Ansatz würden sowohl die beiden Exchange-Server als auch der Edge-Transport-Server in der Cloud auf neuen EC2 Instanzen installiert werden. Dadurch wird die vollständige Verantwortung für die Hardware an den Cloud-Provider übertragen. Allerdings bietet dieser Ansatz aufgrund der Nutzung von EC2-Instanzen nur begrenzte Vorteile in Bezug auf die Skalierbarkeit der Cloud. Die Daten aus den lokalen Installationen können mithilfe der von Microsoft bereitgestellten Tools problemlos auf das neue System übertragen werden, wodurch der Migrationsaufwand für diese Services minimal bleibt. Der Loadbalancer wird durch einen Classic Load Balancer ersetzt, welcher von AWS verwaltet wird und somit eine hohe Verfügbarkeit hat und kaum Verwaltungsaufwand benötigt. Für die Nutzung des AD wird eine VPN-Verbindung zum lokalen Netzwerk aufgebaut. Dies ermöglicht eine einfache Weiterverwendung der bestehenden lokalen Verwaltung und gleichzeitig einen sicheren Zugriff zwischen dem lokalen Netzwerk und den Exchange-Servern. Allerdings hat dieses Set-up zur Folge, dass das lokale AD weiterhin betrieben werden muss. Performance-Probleme aufgrund der VPN-Verbindung sind bei diesem Szenario jedoch kaum zu erwarten, da mit einem relativ geringen Datenaufkommen zu rechnen ist. Das bestehende Monitoring kann in der Regel weiterhin genutzt werden. Allerdings kann es erforderlich sein, für die Überwachung des Load Balancers und der Kosten auf die nativen AWS-Tools zurückzugreifen. Dies hat zur Folge, dass die Überwachung erschwert wird, da nicht alle Daten zentralisiert erfasst werden können. Dadurch kann die Erkennung von Incidents komplizierter werden. Die Kosten sind aufgrund der Nutzung einfacher EC2-Instanzen leicht zu bestimmen. Da nahezu die gleichen Services nur an einem anderen Standort laufen, müssen die Mitarbeiter auch nicht umgeschult werden.

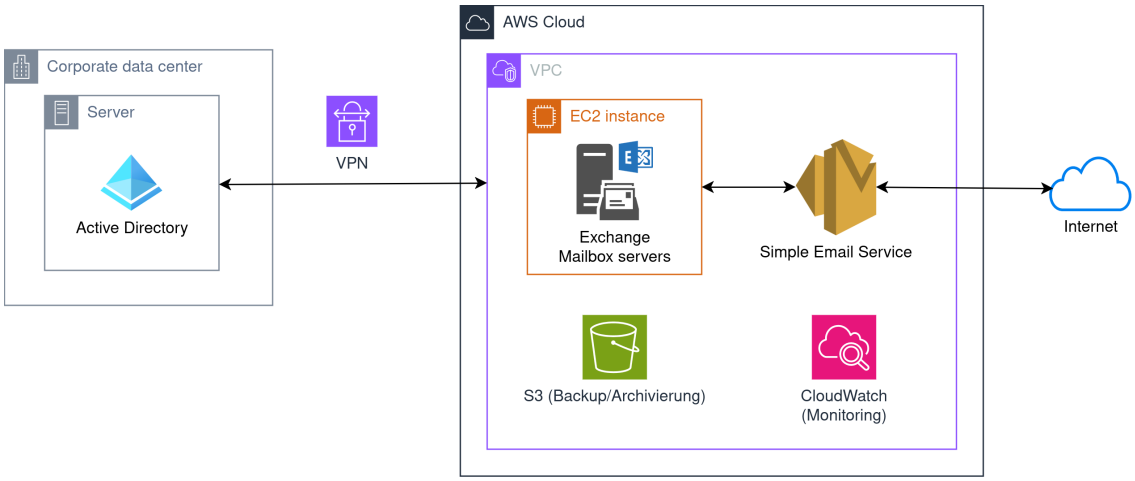

Abbildung 6.3: Exchange Replatforming Architecture

Beim Replatforming-Ansatz verhält es sich bezüglich der Exchange-Instanzen und der VPN-Verbindung zum lokalen AD identisch zum Rehosting-Ansatz. Allerdings werden der Edge-Transport-Server und der Loadbalancer durch den PaaS-Dienst Simple Email Service (SES) ersetzt. Dieser Cloud-native Dienst von AWS übernimmt die entsprechenden Aufgaben und muss lediglich korrekt konfiguriert werden. Da diese Konfiguration wenig aufwändig ist, ist der Migrationsaufwand geringer als beim Rehosting-Ansatz. Außerdem ist keine umfangreiche Schulung der Mitarbeitenden erforderlich, da die Bedienung des Dienstes unkompliziert ist. Ein weiterer Vorteil besteht darin, dass ein Großteil der Verantwortung für diesen kritischen Dienst an den Cloud-Provider übertragen wird, der sowohl eine hohe Verfügbarkeit sicherstellt als auch einfache Skalierungsmöglichkeiten bietet. Dennoch muss das SES überwacht werden, um im Falle von Zwischenfällen (Incidents) schnell reagieren zu können. Falls es nicht möglich ist, die Überwachung von SES in das bestehende Monitoring zu integrieren, müssten beide Systeme parallel genutzt werden. Da SES kostengünstig ist, sind die Infrastrukturkosten im Vergleich zum Rehosting-Ansatz ebenfalls geringer. Da die primären Daten weiter in den EC2-Instanzen liegen, ist eine Optimierung des Backup-Prozesses nur in sofern möglich, skalierbaren Cloudspeicher wie in Form eines S3 Bucktes zu nutzen. Zusammengefasst ist der Replatforming-Ansatz im Vergleich zum Rehosting-Ansatz in nahezu allen Aspekten die bessere Wahl.

Im Rahmen des Replatforming-Ansatzes kann außerdem das AD in die Cloud migriert werden. Dabei gibt es einmal die Option das AD in einer EC2 Instanz zu betreiben, allerdings gibt es wesentlich bessere Alternativen, sodass das keine reale Option ist. Eine sinnvolle Option wäre, das AD in einen Cloud-Nativen Dienst wie AWS Managed Microsoft AD zu migrieren. Dieses Vorgehen wird im Refactoring-Ansatz ausführlich beschrieben.

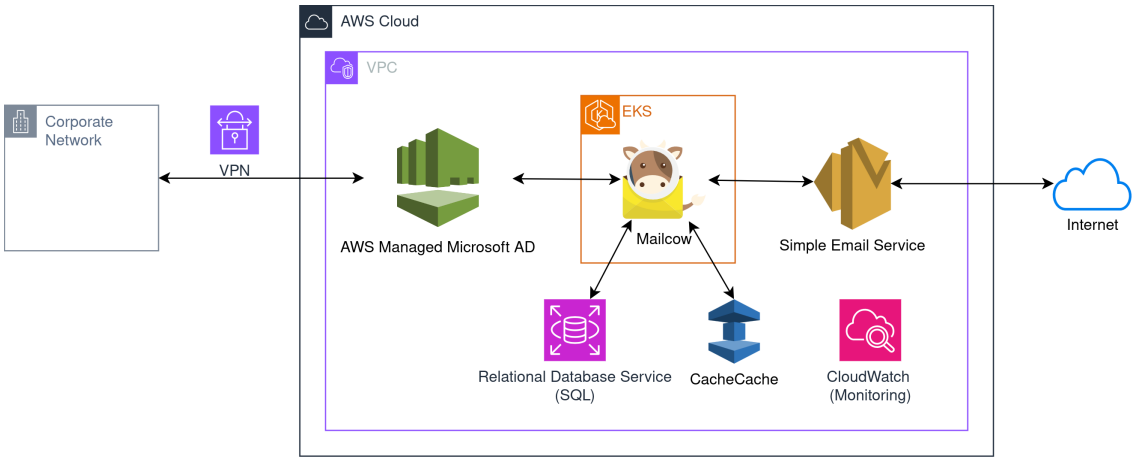

Abbildung 6.4: Exchange Refactoring Architecture

Beim Refactoring-Ansatz wird der Microsoft Exchange Server durch einen Mailcow-Server ersetzt, welcher in Docker-Containern läuft. Diese Container werden in einem AWS EKS (Elastic Kubernetes Service) Cluster bereitgestellt. Dabei entfällt die komplette Verantwortung und Arbeit über das Betriebssystem und die Hardware. Da die Container unabhängig von der Hardware laufen, ist dieser Ansatz sehr flexibel und skalierbar. Ein wesentlicher Vorteil von Mailcow gegenüber Exchange ist die Möglichkeit, die SQL-Datenbank in Amazon RDS zu migrieren. Dies bringt die im Kapitel Backup- und Wiederherstellungsprozesse beschriebenen Vorteile mit sich. Ebenso kann der von Mailcow genutzte Cache in den Cloud-nativen und vollständig von AWS verwalteten ElastiCache-Service migriert werden. Der SES kann weiterhin, wie bereits im Replatforming-Ansatz beschrieben, genutzt werden. Eine weitere große Änderung ist die Migration des lokalen AD in den von AWS verwalteten AD-Service. Dies ermöglicht die Abgabe eines Großteils der Verantwortung und damit auch des Arbeitsaufwands. Zudem skaliert dieser Dienst ohne große Mehrkosten auf eine hohe Nutzeranzahl und bietet somit eine zukunftssichere Lösung. Ein weiter großer Vorteil ist, dass das Monitoring und alle davon abhängigen Prozesse durch die Nutzung der Cloud-nativen Services vereinfacht werden können. Allerdings ist der Migrationsaufwand bei dem Refactoring-Ansatz um ein Vielfaches höher im Vergleich zu den anderen Strategien. Während die Migration des AD vergleichsweise unkompliziert ist, da AWS AD ein ähnliches Interface wie das lokale AD bietet, ist der Aufwand für die Migration des Exchange-Servers groß. Das liegt hauptsächlich daran, dass alle Daten in ein neues Format gebracht werden müssen. Zudem muss das Personal sich mit den neuen Techniken und der neuen Software vertraut machen. Durch die große Veränderung kann auch mit einem hohen Widerstand der Mitarbeiter gerechnet werden. Nach Abschluss des hohen Migrationsaufwands ist jedoch mit einem sehr geringen Betriebsaufwand zu rechnen, da viele Dienste von AWS verwaltet werden. Zudem kann von der hohen Verfügbarkeit dieser Services erheblich profitiert werden.

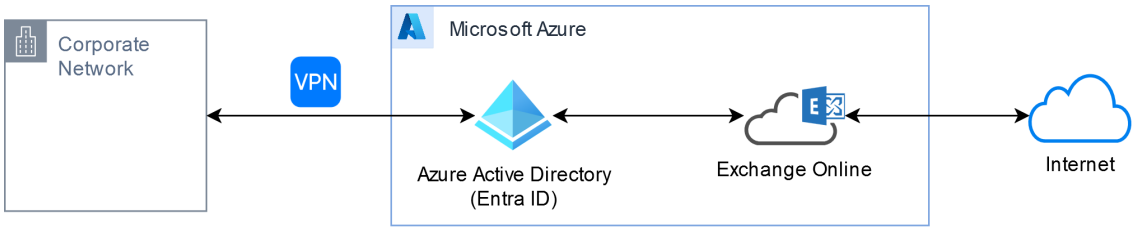

Abbildung 6.5: Exchange Replacing Architecture

Beim Replacing-Ansatz wird auf ein SaaS-Produkt gewechselt. Da Microsoft mit Azure Active Directory und Exchange Online fast genau die benötigten Services als SaaS bereitstellt, bietet sich diese Lösung ideal an. Der Migrationsaufwand ist hierbei am geringsten, da Microsoft dafür bereits einsatzbereite Tools zur Verfügung stellt. Zudem hat diese Variante den Vorteil, dass fast alle Funktionen erhalten bleiben und lediglich die Mitarbeitenden, die für den Betrieb verantwortlich sind, im Umgang mit den neuen Azure-Tools geschult werden müssen. Die Nutzung eines SaaS-Produkts reduziert zudem den Betriebsaufwand erheblich, da Microsoft Azure die Verwaltung aller relevanten Komponenten übernimmt. Um allerdings die komplette Funktionalität beizubehalten, ist es besonders bei einer großen Anzahl lokal betriebener Hardware weiter nötig, einen lokalen AD zu betreiben, wie in [RJ] beschrieben.

Zusammengefasst ermöglicht dieser Ansatz bei gleichbleibenden (oder sogar erweiterten) Funktionen sowohl eine Migration als auch einen Betrieb mit minimalem Arbeitsaufwand. Zudem bietet er eine Umgebung mit hoher Verfügbarkeit und Skalierbarkeit. Allerdings geht mit diesem Ansatz die vollständige Kontrolle über die Systeme verloren, was je nach Anwendungsfall auch als Nachteil gewertet werden kann.

7 Fazit

Wie in dieser Arbeit gezeigt, ist die Migration eines On-Premise betriebenen Services in die Cloud kein triviales Unterfangen und erfordert eine sorgfältige Planung. Dabei kann die Analyse der verschiedenen Abhängigkeiten helfen, fundierte Entscheidungen zu treffen. Es ist wichtig, dabei sowohl die technischen als auch die prozessualen Abhängigkeiten im spezifischen Kontext des jeweiligen Projekts zu bewerten, um alle relevanten Faktoren in die Entscheidung einzubeziehen. Jede Migration-Strategie hat diverse Auswirkungen auf die unterschiedlichen Abhängigkeiten, welche unter divergenten Kontexten anders bewertet werden können. Dies kann dazu führen, dass unter veränderten Rahmenbedingungen für die Migration desselben Services eine andere Migrationsstrategie besser geeignet ist, da die unterschiedlichen Ansätze in der Regel bestimmte Eigenschaften wie hohe Verfügbarkeit oder geringe Migrationskosten unterschiedlich stark unterstützen. Aufgrund dessen ist eine allgemeingültige Aussage über die beste Methode unmöglich zu treffen.

Wie in dem Beispiel gezeigt ist, ein intensives Auseinandersetzen mit der Cloud-Architektur und den Eigenschaften der verschiedenen Services essenziell, um mögliche Strategien zu entwickeln und diese anschließend miteinander zu vergleichen. Dabei dürfen jedoch die bestehenden Prozesse und Mitarbeiter nicht außer Acht gelassen werden, da sie wie gezeigt, ein essenzieller Teil der Migration sind und deswegen auch in dem Migration-Prozess berücksichtigt werden müssen.

A Abkürzungsverzeichnis

AD Active Directory
SES Simple Email Service
CSP Cloud Service Provider
IaC Infrastructure as Code
DBMS Datenbankmanagementsystem
SLA Service-Level-Agreement
SAML Security Assertion Markup Language
LDAP Lightweight Directory Access Protocol
IdP Identity Provider
AWS Amazon Web Services
ECS Elastic Container Service
EKS Elastic Kubernetes Service
EC2 Elastic Compute Cloud
RDS Relational Database Service
GKE Google Kubernetes Engine
GCE Google Compute Engine
SaaS Software-as-a-Service
IaaS Infrastructure-as-a-Service
PaaS Platform-as-a-Service
FaaS Function-as-a-Service
DBaaS Database-as-a-Service

B Glossar

Container Ein Container ist eine leichtgewichtige und isolierte Laufzeitumgebung, die eine Anwendung und ihre Abhängigkeiten verpackt und auf verschiedenen Systemen zuverlässig ausführbar macht

Serverless Serverless ist ein Cloud-Computing-Modell, bei dem der Kunde weder Hardware noch Software der Ausführungsumgebung betreiben muss

Firewall Eine Firewall ist eine Sicherheitsvorrichtung im Netzwerk, welche den Netzwerktraffic nach definierten Regeln überwacht und filtert

VLAN Ein VLAN (Virtual Local Area Network) ist eine logische Unterteilung eines physischen Netzwerks, um Geräte in isolierten Gruppen zu betreiben

Legacy-System Ein Legacy-System ist ein veraltetes Computersystem oder eine Softwareanwendung, das weiterhin benötigt wird

NAT-Gateway Ein NAT-Gateway (Network Address Translation Gateway) ist ein Netzwerkgerät oder Service, welcher viele IP-Adressen in eine geringe Anzahl von IP-Adressen übersetzen kann

Load-Balancer Ein Load-Balancer ist ein Netzwerkgerät oder Service, der den eingehenden Datenverkehr auf mehrere Server verteilt

Bottleneck Ein Bottleneck (Flaschenhals) ist ein Engpass in einem System, welcher die gesamte Leistung oder Effizienz eines Systems einschränkt

Microservice Microservices ist eine Architektur, bei der eine Anwendung in kleine und unabhängige Bestandteile aufgeteilt wird, die eine spezielle Aufgabe übernehmen und über eine klar definierte Schnittstelle kommunizieren

containerisierte Anwendung oder Service der in einem Container ausgeführt wird

VM Eine Virtual Machine (VM) ist eine Software zur Simulation eines physischen Computers, die auf einem Host-System läuft. Dabei können mehrere VM auf einem Host-System isoliert voneinander laufen

Image Ein Image enthällt alle erforderlichen Dateien, Konfigurationen, Abhängigkeiten für ein Betriebssystem um in einer VM, einem Container oder einem physisches System ausgeführt zu werden

C Literaturverzeichnis

[ABDDP13] Giuseppe Aceto, Alessio Botta, Walter De Donato, and Antonio Pescapè. Cloud monitoring: A survey. Computer Networks, 57(9):2093–2115, June 2013. doi:10.1016/j.comnet.2013.04.001.

[AG17] Nick Antonopoulos and Lee Gillam, editors. Cloud Computing: Principles, Systems and Applications. Computer Communications and Networks. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-54645-2.

[ANH18] Naim Ahmad, Quadri Noorulhasan Naveed, and Najmul Hoda. Strategy and procedures for Migration to the Cloud Computing. In 2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS), pages 1–5, Bangkok, Thailand, November 2018. IEEE. doi:10.1109/ICETAS.2018.8629101.

[AV21] Ruhul Amin and Siddhartha Vadlamudi. Opportunities and Challenges of Data Migration in Cloud. Engineering International, 9(1):41–50, April 2021. doi:10.18034/ei.v9i1.529.

[Bak14] Kapil Bakshi. Secure hybrid cloud computing: Approaches and use cases. In 2014 IEEE Aerospace Conference, pages 1–8, March 2014. ISSN: 1095-323X. doi:10.1109/AERO.2014.6836198.

[Bas24] Tom Bassett. Salesforce Data Migration Guide for Admins, August 2024. URL: https://www.salesforceben.com/salesforce-data-migration-guide-for-admins/, Zugriff am 13. November 2024.

[BASS11] Theophilus Benson, Aditya Akella, Anees Shaikh, and Sambit Sahu. CloudNaaS: a cloud networking platform for enterprise applications. In Proceedings of the 2nd ACM Symposium on Cloud Computing, pages 1–13, Cascais Portugal, October 2011. ACM. doi:10.1145/2038916.2038924.

[BGL08] Lawrence D. Bodin, Lawrence A. Gordon, and Martin P. Loeb. Information security and risk management. Communications of the ACM, 51(4):64–68, April 2008. doi:10.1145/1330311.1330325.

[BK11] Aashish Bhardwaj and Vikas Kumar. Cloud security assessment and identity management. In 14th International Conference on Computer and Information Technology (ICCIT 2011), pages 387–392, Dhaka, Bangladesh, December 2011. IEEE. doi:10.1109/ICCITechn.2011.6164819.

[Bra] Henry Bravo. Change the EC2 instance type :: FPGA workshop with Amazon EC2 F1. URL: https://fpga-development-on-ec2.workshop.aws/en/5-ec2-instance-references/change-the-ec2-instance-type.html, Zugriff am 24. November 2024.

[CJP+] Carlo Curino, Evan P C Jones, Raluca Ada Popa, Nirmesh Malviya, Eugene Wu, Sam Madden, Hari Balakrishnan, and Nickolai Zeldovich. Relational Cloud: A Database-as-a-Service for the Cloud.

[CMGS12] Paul Cichonski, Tom Millar, Tim Grance, and Karen Scarfone. Computer Security Incident Handling Guide : Recommendations of the National Institute of Standards and Technology. Technical Report NIST SP 800-61r2, National Institute of Standards and Technology, August 2012. doi:10.6028/NIST.SP.800-61r2.

[CMRH09] M. Cataldo, A. Mockus, J.A. Roberts, and J.D. Herbsleb. Software Dependencies, Work Dependencies, and Their Impact on Failures. IEEE Transactions on Software Engineering, 35(6):864–878, November 2009. doi:10.1109/TSE.2009.42.

[CR11] Christy Pettey and Rob van der Meulen. Gartner Identifies Five Ways to Migrate Applications to the Cloud, May 2011. URL: https://web.archive.org/web/20181028163840/https://www.gartner.com/newsroom/id/1684114, Zugriff am 2. November 2024.

[DB15] Temesgen Kitaw Damenu and Chitra Balakrishna. Cloud Security Risk Management: A Critical Review. In 2015 9th International Conference on Next Generation Mobile Applications, Services and Technologies, pages 370–375, September 2015. doi:10.1109/NGMAST.2015.25.

[Dha21] Dhaval Gogri. SAML vs. LDAP: Best Practices and Use Cases for Cloud-Based Enterprise Authentication and Authorization, 2021. doi:10.13140/RG.2.2.19894.10563.

[Edi] CSRC Content Editor. service - Glossary | CSRC. URL: https://csrc.nist.gov/glossary/term/service, Zugriff am 13. Dezember 2024.

[FJG20] Chen-Fu Fan, Anshul Jindal, and Michael Gerndt. Microservices vs Serverless: A Performance Comparison on a Cloud-native Web Application:. In Proceedings of the 10th International Conference on Cloud Computing and Services Science, pages 204–215, Prague, Czech Republic, 2020. SCITEPRESS - Science and Technology Publications. doi:10.5220/0009792702040215.

[Gai23] Mihail Gaianu. On Premise Data Center vs CLOUD. In 2023 International Conference on Computational Science and Computational Intelligence (CSCI), pages 1068–1071, December 2023. ISSN: 2769-5654. doi:10.1109/CSCI62032.2023.00176.

[Geo24] Dr. A.Shaji George. Consequences of Enterprise Cloud Migration on Institutional Information Technology Knowledge. April 2024. Publisher: PU Publications. doi:10.5281/ZENODO.10938874.

[GPT20] Srinivasa Rao Gundu, Charan Arur Panem, and Anuradha Thimmapuram. Hybrid IT and Multi Cloud an Emerging Trend and Improved Performance in Cloud Computing. SN Computer Science, 1(5):256, September 2020. doi:10.1007/s42979-020-00277-x.

[Gro10] Großbritannien, editor. Service operation: ITIL. TSO, The Stationery Office, London, 3. imp edition, 2010.

[GST12] Kumar Gunjan, G Sahoo, and R K Tiwari. Identity Management in Cloud Computing –A Review. International Journal of Engineering Research, 1(4), 2012.

[GW09] Wei Guo and Ying Wang. An Incident Management Model for SaaS Application in the IT Organization. In 2009 International Conference on Research Challenges in Computer Science, pages 137–140, December 2009. doi:10.1109/ICRCCS.2009.42.

[HB17] Harrison John Bhatti and Babak Bashari Rad. Databases in Cloud Computing: A Literature Review. International Journal of Information Technology and Computer Science, 9(4):9–17, April 2017. doi:10.5815/ijitcs.2017.04.02.

[HBS21] Hassan B. Hassan, Saman A. Barakat, and Qusay I. Sarhan. Survey on serverless computing. Journal of Cloud Computing, 10(1):39, July 2021. doi:10.1186/s13677-021-00253-7.

[HNZ24] Manuel Hoffmann, Frank Nagle, and Yanuo Zhou. The Value of Open Source Software. SSRN Electronic Journal, 2024. doi:10.2139/ssrn.4693148.

[IKZ+22] Muhammad Iqbal, Muhammad Ijaz Khan, Asia Zaman, Muhammad Shahjahan, Muhammad Shahjahan, Rizwan Ullah, Muhammad Mustafa, and Asim Khalil. Challenges in Multi-Cloud and Benefits from Leveraging Cloud Native Strategy to Digital Transformation of Business. 14, 2022.

[JTMH23] Chang Hoong Jack, See Kwee Teck, Lim Tong Ming, and Ding Ying Hong. An Overview Analysis of Authentication Mechanism in Microservices-Based Software Architecture: A Discussion Paper. In 2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS), pages 1–6, Penang, Malaysia, August 2023. IEEE. doi:10.1109/ICSECS58457.2023.10256409.

[Jus] Justinha. Integrate on-premises AD domains with Microsoft Entra ID - Azure Architecture Center. URL: https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/identity/azure-ad, Zugriff am 23. November 2024.

[KA24] Phani Durga Nanda Kishore Kommisetty and Nareddy Abhireddy. Cloud Migration Strategies: Ensuring Seamless Integration and Scalability in Dynamic Business Environments. International Journal of Engineering and Computer Science, 13(04):26146–26156, April 2024. doi:10.18535/ijecs/v13i04.4812.

[KKA14] Issa Khalil, Abdallah Khreishah, and Muhammad Azeem. Cloud Computing Security: A Survey. Computers, 3(1):1–35, February 2014. doi:10.3390/computers3010001.

[LC06] Paul Legris and Pierre Collerette. A Roadmap for it Project Implementation: Integrating Stakeholders and Change Management Issues. Project Management Journal, 37(5):64–75, December 2006. doi:10.1177/875697280603700507.

[LDL+16] Xu Liu, Xiaoqiang Di, Jinqing Li, Hui Qi, Huamin Yang, and Ligang Cong. Optimal backup strategy in cloud disaster tolerance. In 2016 International Conference on Computer, Information and Telecommunication Systems (CITS), pages 1–5, July 2016. doi:10.1109/CITS.2016.7546444.

[Llo13] Vernon Lloyd, editor. ITIL continual service improvement. Number 5 in ITIL lifecycle publication suite. TSO, Norwich, 2013.

[LS08] Suzanna Long and David G. Spurlock. Motivation and Stakeholder Acceptance in Technology-driven Change Management: Implications for the Engineering Manager. Engineering Management Journal, 20(2):30–36, June 2008. doi:10.1080/10429247.2008.11431764.

[LSA17] Michael Lane, Anup Shrestha, and Omar Ali. Managing the Risks of Data Security and Privacy in the Cloud: A Shared Responsibility between the Cloud Service Provider and the Client Organisation. 2017.

[MC22] Vanderlei Munhoz and Márcio Castro. HPC@Cloud: A Provider-Agnostic Software Framework for Enabling HPC in Public Cloud Platforms. In Anais do XXIII Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD 2022), pages 157–168, Brasil, October 2022. Sociedade Brasileira de Computação. doi:10.5753/wscad.2022.226528.

[MG] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing.

[MPB20] Yaser Mansouri, Victor Prokhorenko, and M. Ali Babar. An automated implementation of hybrid cloud for performance evaluation of distributed databases. Journal of Network and Computer Applications, 167:102740, October 2020. doi:10.1016/j.jnca.2020.102740.

[noaa] 14b UStG. URL: https://www.gesetze-im-internet.de/ustg_1980/__14b.html, Zugriff am 24. November 2024.

[noab] Amazon CloudWatch Pricing – Amazon Web Services (AWS). URL: https://aws.amazon.com/cloudwatch/pricing/, Zugriff am 25. November 2024.

[noac] Amazon EBS snapshots - Amazon EBS. URL: https://docs.aws.amazon.com/ebs/latest/userguide/ebs-snapshots.html, Zugriff am 24. November 2024.

[noad] Amazon EC2 security groups for your EC2 instances - Amazon Elastic Compute Cloud. URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html, Zugriff am 22. November 2024.

[noae] Amazon Überwachung und Beobachtbarkeit – Amazon CloudWatch – AWS. URL: https://aws.amazon.com/de/cloudwatch/, Zugriff am 24. November 2024.

[noaf] Ausführungsumgebung von Cloud Functions-Funktionen | Cloud Functions Documentation | Google Cloud. URL: https://cloud.google.com/functions/docs/concepts/execution-environment?hl=de, Zugriff am 23. November 2024.

[noag] AWS | Amazon Linux AMI. URL: https://aws.amazon.com/de/amazon-linux-ami/, Zugriff am 23. November 2024.

[noah] AWS services that publish CloudWatch metrics - Amazon CloudWatch. URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/aws-services-cloudwatch-metrics.html, Zugriff am 25. November 2024.

[noai] AWS VPN | Pricing | Amazon Web Services (AWS). URL: https://aws.amazon.com/vpn/pricing/, Zugriff am 22. November 2024.

[noaj] Causes of IT system software outages global 2023. URL: https://www.statista.com/statistics/1482105/it-system-software-related-outages-root-cause/, Zugriff am 28. November 2024.

[noak] Cloud Computing - Umsatz weltweit bis 2025. URL: https://de.statista.com/statistik/daten/studie/195760/umfrage/umsatz-mit-cloud-computing-weltweit/, Zugriff am 1. November 2024.

[noal] Create estimate: Configure Amazon Virtual Private Cloud (VPC). URL: https://calculator.aws/#/createCalculator/VPC, Zugriff am 22. November 2024.

[noam] EC2 Reserved Instances – Preise – Amazon Web Services (AWS). URL: https://aws.amazon.com/de/ec2/pricing/reserved-instances/, Zugriff am 24. November 2024.

[noan] Forecasting with Cost Explorer - AWS Cost Management. URL: https://docs.aws.amazon.com/cost-management/latest/userguide/ce-forecast.html, Zugriff am 24. November 2024.

[noao] Global cloud infrastructure market share 2024. URL: https://www.statista.com/statistics/967365/worldwide-cloud-infrastructure-services-market-share-vendor/, Zugriff am 1. November 2024.

[noap] How to Optimize AWS Virtual Private Cloud (VPC) Costs. URL: https://www.astuto.ai/blogs/aws-virtual-private-cloud-vpc-strategies-to-reduce-cost, Zugriff am 22. November 2024.

[noaq] Introduction - Amazon Virtual Private Cloud Connectivity Options. URL: https://docs.aws.amazon.com/whitepapers/latest/aws-vpc-connectivity-options/introduction.html, Zugriff am 22. November 2024.

[noar] Microsoft Azure Active Directory - Market Share, Competitor Insights in Identity And Access Management. URL: https://www.6sense.com/tech/identity-and-access-management/microsoft-azure-active-directory-market-share, Zugriff am 23. November 2024.

[noas] milanm/Cloud-Product-Mapping: All major services between AWS, Azure, and GCP are mapped with links pointing to product home pages. URL: https://github.com/milanm/Cloud-Product-Mapping?tab=readme-ov-file, Zugriff am 5. Dezember 2024.

[noat] OWASP Secure Coding Practices - Quick Reference Guide | Secure Coding Practices | OWASP Foundation. URL: https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/02-checklist/05-checklist.html, Zugriff am 23. November 2024.

[noau] Supported operating systems and machine types - AWS Systems Manager. URL: https://docs.aws.amazon.com/systems-manager/latest/userguide/operating-systems-and-machine-types.html, Zugriff am 23. November 2024.

[noav] Top 10 Ransomware-Maßnahmen. URL: https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Cyber-Sicherheitslage/Analysen-und-Prognosen/Ransomware-Angriffe/Top-10-Ransomware-Massnahmen/top-10-ransomware-massnahmen.html?nn=1064216, Zugriff am 24. November 2024.

[noaw] Top 8 IAM Challenges with your SaaS Apps | Okta. URL: https://www.okta.com/resources/whitepaper/top-8-iam-challenges-with-your-saas-apps/, Zugriff am 23. November 2024.

[noax] Troubleshoot why the creation of EC2 AMI or EBS snapshot is slow. URL: https://repost.aws/knowledge-center/ebs-snapshot-ec2-ami-creation-slow, Zugriff am 24. November 2024.

[noay] Was ist AWS Backup? - AWS Backup. URL: https://docs.aws.amazon.com/de_de/aws-backup/latest/devguide/whatisbackup.html, Zugriff am 24. November 2024.

[noaz] What is Amazon CloudWatch Logs? - Amazon CloudWatch Logs. URL: https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html, Zugriff am 24. November 2024.

[noa21a] Improve native backup and restore performance in Amazon RDS for SQL Server AWS Database Blog, 2021. Section: RDS for SQL Server. URL: https://aws.amazon.com/blogs/database/improve-native-backup-and-restore-performance-in-amazon-rds-for-sql-server/, Zugriff am 19. Dezember 2024.

[noa21b] Was ist Terraform? | IBM, October 2021. URL: https://www.ibm.com/de-de/topics/terraform, Zugriff am 5. Dezember 2024.

[noa22] Active Directory vs. LDAP. Wofür wird LDAP verwendet? | Hideez, August 2022. URL: https://hideez.com/de-de/blogs/news/active-directory-vs-ldap, Zugriff am 23. November 2024.

[noa23a] How to improve your security incident response processes with Jupyter notebooks | AWS Security Blog, November 2023. Section: Amazon SageMaker. URL: https://aws.amazon.com/blogs/security/how-to-improve-your-security-incid

ent-response-processes-with-jupyter-notebooks/, Zugriff am 29. November 2024.

[noa23b] Performance of EBS Snapshot Backup and Restore, 2023. URL: https://docs.pingcap.com/tidb-in-kubernetes/stable/backup-restore-snapshot-perf, Zugriff am 19. Dezember 2024.

[noa23c] Was ist Microsoft Exchange Server?, March 2023. URL: https://www.ionos.de/digitalguide/e-mail/e-mail-technik/exchange-server-microsofts-groupware-im-ueberblick/, Zugriff am 16. Dezember 2024.

[noa24] Stephen Orban Linkedin, February 2024. URL: https://www.linkedin.com/in/stephen-orban-7086471/.

[Pal23] Melissa Palmer. Cloud Networking vs On Premises: Differences and Similarities, March 2023. Section: Building the cloud. URL: https://deploy.equinix.com/blog/how-cloud-networking-is-and-isnt-different-from-on-prem/, Zugriff am 22. November 2024.

[PH18] Sebastian Pfaller and Thomas Hartley. Comparison of Windows and Linux as Docker Hosts. 2018.

[PHGK21] Arnak Poghosyan, Ashot Harutyunyan, Naira Grigoryan, and Nicholas Kushmerick. Incident Management for Explainable and Automated Root Cause Analysis in Cloud Data Centers   . JUCS - Journal of Universal Computer Science, 27(11):1152–1173, November 2021. doi:10.3897/jucs.76608.

[RJ] Dr Michael Rodriguez and Sarah Johnson. Unlocking the Power of Azure AD Best Practices for Enterprise Identity Control.

[shl23] shlipsey3. SLA-Leistung Azure Active Directory - Microsoft Entra, May 2023. URL: https://learn.microsoft.com/de-de/azure/active-directory/reports-monitoring/reference-azure-ad-sla-performance, Zugriff am 23. November 2024.

[Ste16] Stephen Orban. 6 Strategies for Migrating Applications to the Cloud | AWS Cloud Enterprise Strategy Blog, November 2016. Section: Adoption. URL: https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/, Zugriff am 18. Oktober 2024.

[ste23] stefanie.loy@extern.euroforum.com. Die Eskalation der Ransomware-Angriffe, July 2023. URL: https://live.handelsblatt.com/die-eskalation-der-ransomware-angriffe/, Zugriff am 24. November 2024.

[WB14] Jonathan Stuart Ward and Adam Barker. Observing the clouds: a survey and taxonomy of cloud monitoring. Journal of Cloud Computing, 3(1):24, December 2014. doi:10.1186/s13677-014-0024-2.

[Wiz] Wiz Cloud: Verwalten der Sicherheitslage. URL: https://www.wiz.io/de-de/platform/wiz-cloud, Zugriff am 29. November 2024.

D Abbildungsverzeichnis