Herausforderungen bei der Migration lokaler Dienste in die Cloud

Thomas Schuster

April 2025

1 Einleitung

Die Migration lokaler IT-Dienste in Cloud-Umgebungen (z. B. AWS, Azure, GCP) verspricht Unternehmen Vorteile wie **Skalierbarkeit**, **Agilität** und **mögliche Kosteneinsparungen**. Gleichzeitig ist sie eine komplexe, interdisziplinäre Aufgabe. Diese Einheit gibt einen anwendungsorientierten Einstieg und liefert einen Überblick über Strategien, typische Stolpersteine und ökonomische Bewertung.

Fahrplan dieser Einheit

1. Motivation & Ziele der Migration
2. Anwendungsfall *Lift & Shift*
3. Kostenfaktoren → TCO-Analyse
4. Alternativen & Handlungsempfehlungen

2 Motivation und Grundlagen der Cloud-Migration

Die Verlagerung von IT-Ressourcen in die Cloud stellt einen bedeutenden Schritt dar. In der Einheit besprechen wir grundlegende Konzepte, Motivationen und gängigen Strategien der Cloud-Migration.

- **Definition**: Verlagerung von Anwendungen, Daten & IT-Ressourcen aus einem lokalen Rechenzentrum (On-Premises) oder einer bestehenden Hosting-Umgebung in eine Cloud-Infrastruktur.
- **Typische Ziele**: Kostenreduktion (z. B. durch Pay-as-you-go), Erhöhung von Skalierbarkeit & Verfügbarkeit, schnellere Time-to-Market für neue Services, Innovationsförderung.
- **Migrationsstrategien** (**7 R's**, sortiert nach i. d. R. wachsendem Aufwand & Cloud-Nativität): Retain → Rehost → Replatform → Repurchase → Refactor → Rearchitect → Retire [1].
- **Lift & Shift** (Rehost): Gilt als schnellster Ansatz; Übertragung bestehender Workloads (Server, VMs) weitgehend unverändert in eine Infrastructure as a Service (IaaS) Umgebung.

3 Anwendungsfall: Lift & Shift einer Microservice-App

Als Beispiel für die Durchführung einer Lift & Shift Migration dient unsere bekannte App, die auf einer Quarkus Microservice-Architektur basiert. Zur Reproduzierbarkeit, streben wir in weiteren Schritten die Automatisierung durch Bereitstellung per Infrastructure as Code (IaC) – beispielsweise durch CloudFormation oder Terraform. In Abbildung 1 sehen wir das damit verbundene Zielbild. Lift & Shift charakterisiert sich für uns zunächst wie folgt:

Vorteile

- Einfacher Einstieg
- Geringe Code-Änderungen
- Schnell (Time-to-Cloud)

Nachteile

- Eingeschränkte Skalierung
- Cloud-native Features bleiben ungenutzt
- *Cloud Shock*: hohe Betriebskosten

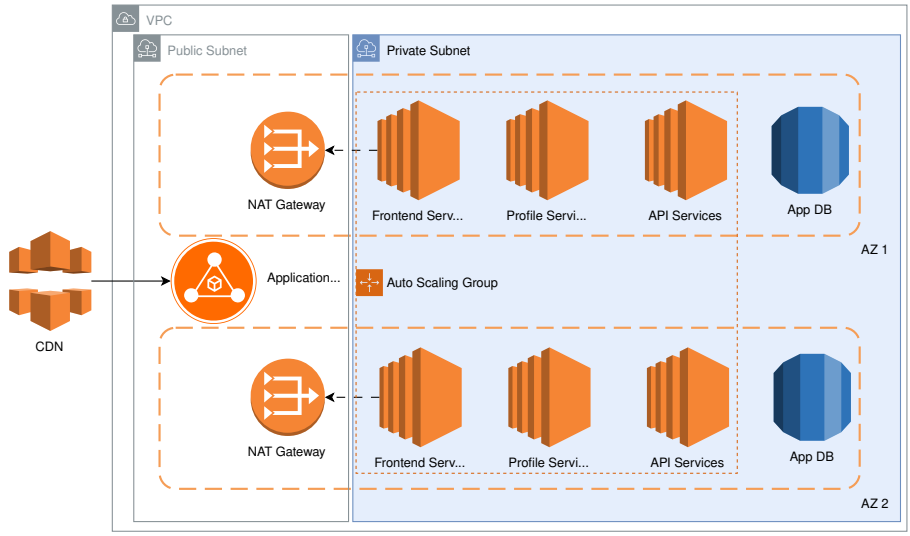

Abbildung 1: Zielarchitektur in AWS (EC2, RDS, S3, ALB)

4 Migrationsstrategien und wirtschaftliche Betrachtung

Nachdem eine Anwendung – wie in unserem Beispiel per Lift & Shift – in die Cloud gebracht wurde, beginnt die eigentliche Arbeit: die kontinuierliche Optimierung und strategische Weiterentwicklung. Ignoriert man dies, droht der sogenannte *Cloud Shock* durch unerwartet hohe Betriebskosten. Betrachten wir daher drei wichtige Aspekte:

4.1 Laufende Kostenoptimierung: Das A und O in der Cloud

Unabhängig von der ursprünglichen Migrationsstrategie sind bestimmte Praktiken essenziell, um die Cloud-Ausgaben im Griff zu behalten:

- **Rightsizing** (Passgenaue Dimensionierung): VM-Instanzen, Datenbanken und Speicher nicht zu groß wählen! Analysieren tatsächlicher Last und Anpassung der Ressourcen.
- **Rabattmodelle** nutzen: Für konstant laufende Systeme bieten Cloud-Provider erhebliche Rabatte bei längerer Bindung (1-3 Jahre), z.B. über *AWS Savings Plans* oder *Reserved Instances*.
- **Aufräumen & Überwachen**: Nicht mehr benötigte Ressourcen (Test-VMs, verwaiste Speicher-Snapshots) verursachen unnötige Kosten. Es ist daher wichtig Prozesse zum Abschalten und Löschen zu etablieren. *Tagging* und kontinuierliches *Monitoring* von Ressourcen helfen Kostentreiber zu erkennen.

4.2 Alternativen zu Lift & Shift: Cloud-Potenzial heben

Lift & Shift ist oft der schnellste Weg in die Cloud, nutzt aber deren spezifische Vorteile nur begrenzt. Daher folgen oft weitere, fortgeschrittenere „**R**“Schritte:

- **Replatforming** (Anpassen & Umbetten): Hierbei passt man die Anwendung leicht an, um verwaltete Cloud-Dienste (PaaS) zu nutzen. Statt VMs selbst zu managen, könnten die Microservices als Container in *AWS ECS* oder *EKS* laufen. Das reduziert den Verwaltungsaufwand.
- **Refactoring/Rearchitecting** (Modernisieren): Die Anwendung wird grundlegend umgebaut, um *Cloud-native* Architekturen zu etablieren. Beispiele sind die Zerlegung in kleine, ereignisgesteuerte Funktionen (*AWS Lambda*) oder asynchrone Kommunikation über Queues/Streams.

4.3 Total Cost of Ownership (TCO): Das Gesamtbild der Kosten

Die monatliche Cloud-Rechnung ist nur ein Teil der Wahrheit. Eine realistische Bewertung erfordert die Betrachtung der Gesamtkosten über den gesamten Lebenszyklus (*Total Cost of Ownership*). Diese umfasst typischerweise:

Phase	**Typische Kostenkomponenten**
Einführung (Adoption)	Schulungen, Proof-of-Concepts, externe Beratung
Migration	Daten-/System-Transfer, evtl. Parallelbetrieb, Code-Anpassungen
Betrieb (Operate)	Rechenleistung, Speicher, Netzwerk, Support, Monitoring-Tools
Ausstieg (Exit)	Datenrückführung, Vertragsauflösung, ggf. Re-Hosting On-Prem

Eine sorgfältige TCO-Analyse ist unerlässlich für Migrationsentscheidungen und eine realistische Budgetplanung. Sie hilft, böse Überraschungen zu vermeiden und kann dabei helfen die Migrationsstrategie anzupassen [2].

5 Fazit

Zusammenfassend lässt sich festhalten, dass die Migration lokaler Dienste in die Cloud ein komplexes Vorhaben ist, das über rein technische Aspekte hinausgeht. Für eine erfolgreiche Umsetzung sind strategische Überlegungen und eine kontinuierliche Kostenkontrolle entscheidend.

Take-aways

- Migration ist nicht nur technisch, sondern auch **strategisch**.
- Wähle die Strategie passend zu Ressourcen, Zeit & Use Case.
- Prüfe die **Kosten** kontinuierlich (vermeide einen Cloud Shock).

Reflexionsaufgaben

1) Vergleichen Sie die **TCO** zweier Optionen:
   (a) Lift & Shift auf EC2
   (b) Replatforming auf einen Managed-Container-Service
2) Ordnen Sie den vorgestellten Use Case in die *7 R's* ein und begründen Sie, welche nächste Evolutionsstufe (Replatform / Refactor) strategisch am meisten Mehrwert bringt.
3) Skizzieren Sie ein **Cloud-nativeres Deployment** unserer Quarkus-App:
   - Welche AWS-Services würden Sie einsetzen (EKS, Fargate, DynamoDB, …)?
   - Wie ändern sich Deploy-Pipeline und Betriebsverantwortung?

Literatur

[1] *AWS Migration Whitepaper: The 7 R's of Migration.* Amazon Web Services. 2023. URL: `https://aws.amazon.com/cloud-migration`.

[2] Simon Heinrich u. a. „A Total Cost of Ownership Model for Cloud Computing Infrastructure". In: (3. Jan. 2023), S. 5779–5788. URL: `https://hdl.handle.net/10125/103337` (besucht am 19. 04. 2025).