Projekt IPLocate

Seminararbeit

im Studiengang Informatik

an der Dualen Hochschule Baden-Württemberg

Stuttgart

von

Robert Orbach

1.04.2025

Bearbeitungszeitraum	März 2025 bis April 2025
Matrikelnummer, Kurs	8489365, INF22A
Bei Dozent	Niels Riekers

Inhaltsverzeichnis

1 Einleitung

Cloud Computing hat sich in den letzten Jahren zu einem zentralen Bestandteil moderner IT-Infrastrukturen entwickelt. Unternehmen nutzen Cloud-Dienste, um ihre Anwendungen effizienter zu betreiben, Kosten zu senken und die Skalierbarkeit zu verbessern.

Im Rahmen der Vorlesung haben wir das Themenfeld erkundet und uns mit den Möglichkeiten und Herausforderungen von Cloud Computing auseinandergesetzt. Um diese selbst auszuprobieren, haben wir die Chance, ein eigenes Projekt unter Anwendung der Möglichkeiten des Coud Computing zu realisieren.

In diesem Projekt wurde eine Webanwendung unter Nutzung von Cloud Computing Ansätzen entwickelt, die dem Nutzer erlaubt von IP Adressen auf die ungefähre geographische Lage zu schließen.

2 Theorie

2.1 Was ist Cloud Computing?

Klassische IT-Infrastrukturen basieren auf physischen Servern und Rechenzentren, die vor Ort betrieben werden. Da diese jedoch oft teuer zu betreiben und schwer zu skalieren sind, setzen viele Unternehmen auf Cloud Computing.

Cloud Computing bezeichnet die Bereitstellung von IT-Ressourcen wie Servern, Speicher, Datenbanken, Netzwerken und Software über das Internet. Es gibt drei Hauptmodelle[4, 6, 7]:

- **Infrastructure as a Service (IaaS):** Bereitstellung von virtuellen Maschinen, Netzwerken und Speicher. Der Nutzer verwaltet die Infrastruktur selbst. Beispiele: AWS EC2, Azure Virtual Machines.
- **Platform as a Service (PaaS):** Bereitstellung einer Plattform für die Entwicklung und Bereitstellung von Anwendungen. Der Anbieter verwaltet die Infrastruktur, während der Nutzer sich auf die Anwendung konzentriert. Beispiele: Azure App Service, Google App Engine.
- **Software as a Service (SaaS):** Bereitstellung von Software über das Internet. Der Nutzer verwendet die Software, ohne sich um die Infrastruktur oder Plattform kümmern zu müssen. Beispiele: Google Workspace, Microsoft Office 365.

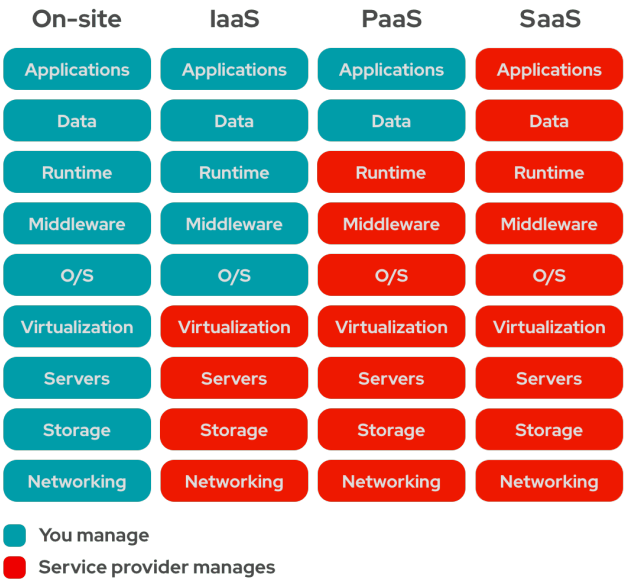

Abbildung 2.1: Cloud Computing Modelle

2.2 Warum Cloud Computing?

Die Cloud bietet zahlreiche Vorteile:

- **Skalierbarkeit:** Ressourcen können je nach Bedarf dynamisch angepasst werden.
- **Kostenersparnis:** Keine Investitionen in physische Hardware, Bezahlung nach Nutzung.
- **Flexibilität:** Zugriff auf Ressourcen von überall und jederzeit.
- **Automatisierung:** Tools wie Terraform und Ansible ermöglichen die einfache Verwaltung und Bereitstellung von Infrastruktur.

[5]

Jedoch gibt es auch Nachteile bei der Nutzung von Cloud Computing:

- **Abhängigkeit:** Durch die Nutzung von Cloud-Diensten machen sich Unternehmen von den Anbietern abhängig.
- **Sicherheit:** Schutz sensibler Daten und Zugriffskontrolle sind kritisch und müssen extra genau betrachtet werden da die Daten auf Drittservern gespeichert werden.
- **Kosten:** Fehlende Kontrolle über die Ressourcennutzung kann zu hohen Kosten führen. Außerdem sind die Kosten bei einem konstanten Bedarf höher als bei einer eigenen Infrastruktur.

[1]

2.3 Terraform

Terraform ist ein Open-Source-Tool zur Infrastrukturautomatisierung. Es ermöglicht die deklarative Beschreibung von Infrastruktur als Code (IaC). Mit Terraform können Ressourcen wie virtuelle Maschinen, Netzwerke und Datenbanken in verschiedenen Cloud-Anbietern wie AWS, Azure oder Google Cloud bereitgestellt werden[2].

Vorteile von Terraform:

- Plattformunabhängigkeit.
- Wiederholbare und konsistente Bereitstellung.
- Versionskontrolle der Infrastruktur.

2.4 Ansible

Ansible ist ein Open-Source-Tool zur Konfigurationsverwaltung und Automatisierung. Es wird verwendet, um Software zu installieren, Konfigurationen zu ändern und Anwendungen bereitzustellen[3].

Folgende Features machen Ansible attraktiv:

- Agentenlos: Es benötigt keine zusätzliche Software auf den Zielsystemen.

- Einfachheit: Konfigurationen werden in YAML geschrieben.
- Skalierbarkeit: Kann große Umgebungen effizient verwalten.

3 Umsetzung

3.1 Architektur

Die Architektur des Projekts umfasst folgende Komponenten:

1. **Virtuelles Netzwerk:** Ein Azure Virtual Network mit Subnetzen für die Anwendung und die Datenbank.

2. **Datenbank:** Eine PostgreSQL Flexible Server Instanz, die als PaaS-Dienst bereitgestellt wird.

3. **Webanwendung:** Mehrere virtuelle Maschinen (VMs) mit einer Load Balancer-Konfiguration, die eine Next.js-Anwendung hosten.

4. **Monitoring:** Eine Monitoring Instanz über die mittels pgAdmin, Prometheus und Graphana die Überwachung der Anwendung erfolgt.

5. **Automatisierung:** Terraform wird verwendet, um die Infrastruktur zu erstellen, und Ansible, um die VMs zu konfigurieren und die Anwendung bereitzustellen.

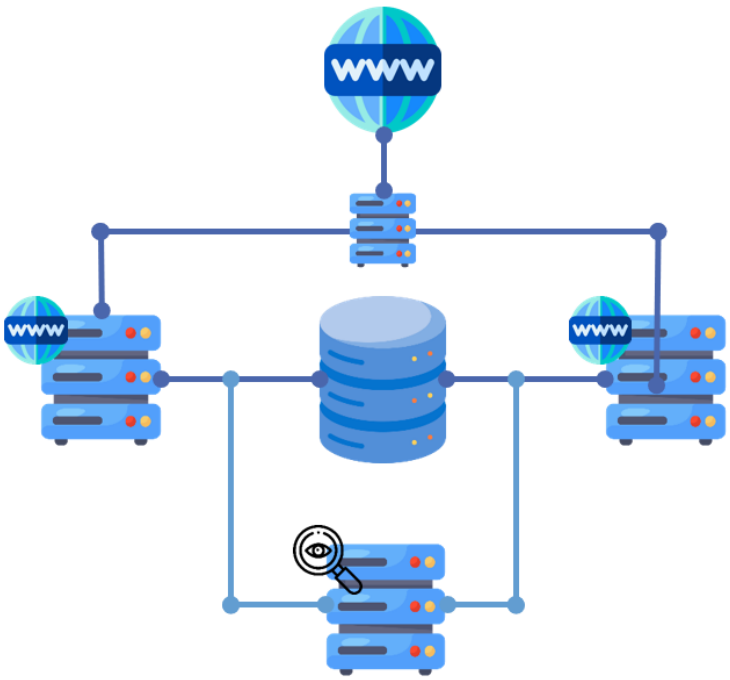

Abbildung 3.1: Architektur des Projekts

3.2 Aufbauablauf

Schritt 1: Infrastruktur mit Terraform

Zunächst wird die Infrastruktur mit Terraform bereitgestellt. Dazu wird eine Azure Resource Group erstellt, die als Container für alle Ressourcen dient. Anschließend wird ein virtuelles Netzwerk mit Subnetzen für die Anwendung und die Datenbank konfiguriert. Eine PostgreSQL Flexible Server Instanz wird mit privatem Netzwerkzugriff eingerichtet. Danach werden zwei virtuelle Maschinen für die Webanwendung erstellt und mit einem Load Balancer verbunden. Abschließend wird ein Monitoring-Server bereitgestellt, um die Überwachung der Anwendung zu ermöglichen.

Schritt 2: Konfiguration mit Ansible

Konfiguration der App VMs

- Installation von Node.js und PostgreSQL-Client auf den VMs.

- Bereitstellung der Next.js-Anwendung aus dem Git-Repository.
- Konfiguration der Umgebungsvariablen für die Datenbankverbindung.
- Erstellung eines Systemd-Dienstes, um die Anwendung automatisch zu starten.

Konfiguration des Monitoring Servers

- Installation von pgAdmin, Prometheus und Grafana auf dem Monitoring Server.
- Konfiguration von pgAdmin zur Verwaltung der Datenbank.
- Konfiguration von Prometheus und Grafana zur Überwachung der Anwendung.
- Einrichtung von Dashboards in Grafana zur Visualisierung der Metriken.

4 Fazit

Die im Rahmen dieses Projekts entwickelte Cloud-Infrastrukturlösung demonstriert die modernen Möglichkeiten, um Software effektiv mittels DevOps-Praktiken in die Cloud zu bringen und in dieser zu Managen. Die Implementierung einer vollständig automatisierten Bereitstellungspipeline mittels Terraform und Ansible schafft eine robuste Grundlage für skalierbare Webanwendungen.

4.1 Anwendungsszenarien und Eignung

Das Projekt hat gezeigt, dass die Lösung besonders wertvoll für mittelgroße bis große Anwendungen ist, die folgende Anforderungen erfüllen:

- Schwankender Nutzerlast wodurch von flexibler Skalierung profitiert werden kann
- Hohe Verfügbarkeit durch Redundanz der Anwendungsserver benötigt wird
- Nachhaltige Entwicklungsprozesse mit wiederholbaren Bereitstellungsschritten

Vor allem für Teams, die agil arbeiten und häufige Deployments durchführen, bietet dieser Ansatz erhebliche Vorteile durch die Standardisierung der Infrastruktur als Code und der möglichkeiten zur automatischen Überführung von Änderungen in die Produktion.

4.2 Aufwand und Ressourcenbedarf

Die Implementierung der Lösung erfordert eine differenzierte Betrachtung des Aufwands:

- **Initiale Einrichtung:** Der Aufbau der Terraform- und Ansible-Konfigurationen verlangt Fachwissen und eine präzise Planung der Infrastrukturkomponenten und ist relativ aufwendig.
- **Wartung und Weiterentwicklung:** Nach der initialen Einrichtung reduziert sich der Wartungsaufwand erheblich. Infrastrukturänderungen können meist durch einfache Anpassungen der Konfigurationsdateien versioniert und getestet werden.

- **Betriebsaufwand:** Sobald die Infrastruktur steht, ist der Betriebsaufwand gering. Die Überwachung und das Management der Ressourcen erfolgt automatisiert und kann durch Dashboards wie Grafana visualisiert werden.

4.3 Stärken der Architektur

Die gewählte Architektur zeichnet sich durch mehrere Stärken aus:

- **Hohe Ausfallsicherheit** durch redundante Anwendungsserver und Load Balancing
- **Sicherheit durch Design** mit privaten Subnetzen und klar definierten Netzwerkgrenzen
- **Konsistente Umgebungen** durch Infrastructure as Code, die Umgebungsdrift verhindert
- **Umfassende Überwachbarkeit** durch integrierte Monitoring-Lösungen mit Prometheus und Grafana
- **Wartbarkeit** durch klare Trennung von Infrastruktur und Konfiguration

4.4 Herausforderungen und Einschränkungen

Trotz der Vorteile bestehen einige Herausforderungen:

- **Cloud-Anbieter-Bindung:** Die Lösung nutzt spezifische Azure-Dienste, was eine potenzielle Abhängigkeit schafft. Eine Multi-Cloud-Strategie würde zusätzlichen Aufwand erfordern. Dies ist mit PaaS Services nur schwer erreichbar.
- **Lernkurve:** Teams ohne DevOps-Erfahrung müssen sich in Tools wie Terraform und Ansible einarbeiten was einen erheblichen Schulungsaufwand bedeutet.
- **Kostenmanagement:** Die flexible Skalierung erfordert proaktives Kostenmonitoring, um Ressourcenverschwendung zu vermeiden.
- **Komplexität:** Die verteilte Architektur erhöht die Komplexität bei der Fehlerdiagnose, was ein umfassendes Monitoring unerlässlich macht.

4.5 Zusammenfassung

Das Projekt demonstriert erfolgreich den Einsatz moderner DevOps-Methoden zur Bereitstellung einer skalierbaren Web-Infrastruktur in der Azure-Cloud. Durch die strategische Kombination von Infrastructure as a Service (IaaS) für Anwendungsserver und Platform as a Service (PaaS) für die Datenbank wurde eine ausgewogene Balance zwischen Kontrolle und Verwaltungsaufwand erreicht.

Die vollständige Automatisierung des Bereitstellungsprozesses durch Terraform und Ansible eliminiert manuelle Konfigurationsschritte und stellt sicher, dass die Infrastruktur jederzeit reproduzierbar bleibt. Das integrierte Monitoring-System ermöglicht eine kontinuierliche Überwachung aller Komponenten und schafft die Grundlage für proaktives Ressourcenmanagement.

Diese Architektur bietet einen zukunftssicheren Ansatz für moderne Webanwendungen, der mit den Anforderungen wachsen kann und gleichzeitig einen strukturierten Rahmen für kontinuierliche Verbesserungen der Infrastruktur bietet.

Literaturverzeichnis

[1] J. Günther und CP Praeg. „Bedeutung und Management von Cloud Computing, Multi-Cloud und Cloud Brokerage in Unternehmen“. In: *Cloud Computing Journal* (2023). DOI: 10.1365/s40702-023-00991-z.

[2] HashiCorp. *Terraform Documentation*. o.J. URL: https://developer.hashicorp.com/terraform/docs (besucht am 28. 03. 2025).

[3] Red Hat. *Ansible Documentation*. o.J. URL: https://docs.ansible.com/ansible/latest/getting_started/index.html (besucht am 28. 03. 2025).

[4] Judith Hurwitz u. a. *Cloud Computing For Dummies*. John Wiley & Sons, 2010. ISBN: 978-0-470-59001-9.

[5] IBM. *Was sind die Vorteile von Cloud-Computing?* o.J. URL: https://www.ibm.com/de-de/topics/cloud-computing-benefits (besucht am 28. 03. 2025).

[6] Peter Mell und Timothy Grance. „The NIST Definition of Cloud Computing“. In: NIST Special Publication 800-145 (Sep. 2011). DOI: 10.6028/NIST.SP.800-145. URL: https://doi.org/10.6028/NIST.SP.800-145.

[7] Qinghua Zhang, Lu Cheng und Raouf Boutaba. „Cloud computing: state-of-the-art and research challenges“. In: *Journal of Internet Services and Applications* 1.1 (2010), S. 7–18. DOI: 10.1007/s13174-010-0001-x.