Fachhochschule Südwestfalen

Bereitstellung einer Fullstack Anwendung auf der Google Cloud Plattform

Lukas Wessel

18. Juli 2022

Zusammenfassung

Schriftliche Ausarbeitungen sind wissenschaftliche Texte, die in ihrem formalen Aufbau bestimmten Richtlinien entsprechen müssen. Dies gilt im Besonderen für Abschlussarbeiten (Bachelor- oder Masterarbeiten), aber prinzipiell auch für kürzere Aufsätze, Hausarbeiten und Projektberichte. In diesem Leitfaden soll es darum gehen, wie Sie Ihre Ausarbeitung strukturell aufbauen sollten und welche Qualitätskriterien für die äußere und sprachliche Form gelten. Bei einer typischen Projekt-, Bachelor- oder Masterarbeit macht die schriftliche Ausarbeitung nur einen Teil der Arbeitslast aus, ist aber gleichzeitig das wichtigste Kriterium für die Bewertung. Daher ist es ratsam, sich möglichst frühzeitig mit den inhaltlichen und formalen Anforderungen wissenschaftlicher Texte vertraut zu machen und diese bei der Anfertigung eigener Ausarbeitungen zu berücksichtigen.

Inhaltsverzeichnis

1 Glossar

DevOps Mandantentrennung - Personenbezogene Daten dürfen keinem fremden Kunden zugänglich werden. Aus diesem Grund müssen die Daten der Kunden voneinander getrennt werden, man spricht in diesem Fall von Mandantentrennung.

2 Einleitung

- Welche Kombination an Cloud-Services bietet sich zurzeit für eine Fullstackanwendung an? - Welche Alternativen Technologien können verwendet werden?

3 Stand der Technik

3.1 Verfügbare Technologien

(todo: Soll ich die Kapitelüberschrift streichen? Ist das doppelt mit Stand der Technik? Ich find das ist irgendwie doppelt.)

In diesem Kapitel werden die in Frage kommenden Technologien und Konzepte hinsichtlich des Betriebs, der Bereitstellung der Images und des CI/CD Prozesses beleuchtet.

3.1.1 Cloud SQL

Cloud SQL ist ein Service zum Bereitstellen von Datenbanken, die von Google in dessen Cloud Plattform bereitgestellt wird. Als Anwender kann hier eine PostGreSQL, MySQL oder eine MSSQL Datenbank instanziiert werden. Hierbei werden laut Ożarowska die Daten ßtored and processed in the cloud, on the infrastructure of a cloud service provider". **cloudsql**

3.1.2 Cloud Run

Venema definiert Cloud Run wie folgt: "Cloud Run is a platform on Google Cloud that lets you build scalable and reliable web-based applications. As a developer, you can get very close to being able to just write your code and push it, and then let the platform deploy, run, and scale your application for you."
Diese Applikationen lassen sich in Netzwerken vereinen oder voneinander trennen, mit Rechten und Rollen ausstatten (IAM) und mit verschiedensten Datenbankservices wie etwa Cloud SQL oder Firestore verbinden. Für Cloud Run sind Containertechnologien wie Docker vorgesehen, um deren Images mit einer "pre-built app"**cloud-run-johnson** zu "pullenünd zu instanziieren. Die Technologie lässt sich effizienter nutzen, indem sie mit

"Continuous Deployment"kombiniert wird (siehe Kapitel 3.1.10), da der administrative Aufwand von zukünftigen Bereitstellungen reduziert wird **whats-cd-lamm**.

3.1.3 Virtuelle Maschinen

Lackes und Siepermann definieren im "Gabler Wirtschaftslexikonëine virtuelle Maschine wie folgt: "Bei Mehrbenutzerbetrieb, v.a. bei Teilnehmerbetrieb, wird durch das Betriebssystem für jeden einzelnen Teilnehmer eine eigene Hardware-Umgebung simuliert, eine „virtuelle Maschine". Diese enthält z.B. einen eigenen Arbeitsspeicher, Magnetplattenspeicher und Drucker, individuell für den Teilnehmer. Prinzipiell kann jede virtuelle Maschine mit einem anderen Betriebssystem betrieben werden. Die interne Realisierung der virtuellen Maschine durch das Betriebssystem erfolgt natürlich auf den realen Geräten; z.B. wird der virtuelle Drucker eines Teilnehmers auf einem realen Drucker der Datenverarbeitungsanlage abgebildet."**gabler-vm**
Im Kontext einer Cloud spielen diese eine wichtige Rolle, da sie Mandantentrennung unter möglichst effizienter Nutzung der Hardwareressourcen des Rechenzentrums gewährleisten können.

3.1.4 Compute Engine

Virtuelle Maschinen werden mit der Compute Engine erzeugt. Hierbei entsteht ein großer administrativer Aufwand, da die Installation von Updates und Installationen wie Docker oder Datenbanktechnologien durch den Kunden vorgenommen werden. Auf der anderen Hand liegt eine hohe Kontrolle des Servers vor, da viele Einstellungen des Servers vom Kunden vorgenommen werden können. Er genießt somit große Freiheit in der Wahl seiner Software und Einstellungen, ist jedoch auch verantwortlich für die Pflege hierfür.

3.1.5 Docker

3.1.6 Cloud Build

3.1.7 Cloud

3.1.8 GitHub Actions

3.1.9 Continuous Integration

3.1.10 Continuous Deployment

4 Methodik

Sollte ich hier lieber eine Aufzählung machen oder eigene Kapitel vorsehen? Wie ausführlich sollte das sein, ich will mich nicht wiederholen. Bei der Umsetzung der praktischen Ausarbeitung wird einer strukturierten Vorgehensweise gefolgt, um das Ergebnis möglichst zeiteffizient zu erreichen. Diese erfolgt in sechs(todo: checken obs 6 sind) Schritten. Als Grundlage für diverse Bereitstellungen wird Docker herangezogen. Docker verfügt über die Möglichkeit, Software-Instanzen in einer relativ kleinen (bezüglich Arbeits- und physischem Speicher) Umgebung (Sandbox?) zu betreiben. Alternative Technologien hierfür sind (todo), aufgrund der Bekanntheit/Nutzerbasis und der umfangreichen Unterstützungsmöglichkeiten (Dokumentation) wird Docker an dieser Stelle als Containerisierungstechnologie für die vorliegende Arbeit festgelegt.

4.1 Bereitstellung des Repositories auf Github

Vor der eigentlichen Umsetzung wurde ein Softwareprojekt umgesetzt. Der Tech-Stack umfasst eine MSSQL Datenbank, ein ASP.NET Core Backend sowie ein Angular Frontend. Um es perspektivisch der Google Cloud zugänglich zu machen wurde der Code als privates GitHub Mono-Repository im Internet veröffentlicht. Das Projekt wird als gegeben / als Ausgangslage betrachtet und soll aus diesem Grund für die vorliegende Arbeit nicht näher betrachtet werden.

4.2 Erzeugen und lokales Ausführen der Docker Images

Es soll verifiziert werden, dass die Docker-Images der Datenbank, des Backends und des Frontend grundsätzlich lauffähig sind. Aus diesem Grund wird vorab ein Docker-Image mittels des "Docker Build"(todo: konkreter Befehl?) lokal erzeugt.

4.3 Relevante Artikel der Dokumentation lesen

Um zu erlernen, welche Google Services sich grundsätzlich für den Zweck eignen und wie diese miteinander verkettet werden können soll die Google Cloud Dokumentation hinsichtlich der genannten Cloud Technologien nachvollzogen werden.

4.4 Build der Anwendung und Publikation in einer Registry

Der Code soll vom GitHub Repository auf der Google Cloud Plattform gebaut werden und anschließend in einer Image Registry für den weiteren Prozess abgelegt werden. Hierfür sind potenziell die Google Cloud Docker Image Registry oder der Docker Hub geeignet, sie erlauben das "Pushingünd "Pulling"von Docker Images.

4.5 Deployment der Datenbank

Die vorhandene MSSQL Datenbank soll auf der Google Cloud Plattform bereitgestellt werden. Eine Verifizierung der Funktionalität erfolgt durch eine testweise Verbindung mittels eines Tools wie SQL Server Management Studio.

4.6 Deployment des Backends

Das ASP.NET 5 Backend wird in diesem Schritt verfügbar gemacht. Um zu überprüfen, ob dieses abgerufen werden kann, soll die bestehende API Dokumentation via Swagger getestet werden.

4.7 Deployment des Frontends

Im finalen Schritt soll das Angular Frontend bereitgestellt werden. Um die erfolgreiche Umsetzung zu verifizieren soll mit einem Browser auf die von Google deklarierte IP-Adresse bzw. URL navigiert werden.

1. Bereitstellung des
2. Age

5 Umsetzung

Um abschätzen zu können, welche Technologien für ein Deployment in der Google Cloud in Frage kommen wurde die Dokumentation von Google herangezogen. Es wurde erkannt, dass derzeit die "Cloud RunÄPI eine für den Zweck passende Alternative darstellt. Begründen lässt sich dies darin, dass es einerseits eine aktuelle Cloud-Technologie ist und andererseits von Google und mehreren Autoren in aktuellen Lektüren für diesen Anwendungszweck vorgesehen wird.(todo ZITATE) Darüber hinaus existiert ein begrenztes Gratis Kontingent an Ressourcen für diesen Service. So sind beispielsweise die ersten 180.000 vCPU Sekunden im Monat kostenfrei. (https://cloud.google.com/run/pricing) Weiterhin (Fachwort möglich lol) eignet sich grundsätzlich auch die Compute Engine für den Betrieb der Applikationen, da hierdurch virtuelle Maschinen und folglich Docker-Installationen ermöglicht werden.

5.1 Cloud Build

Der Google Cloud Build wurde verwendet, um den Code von GitHub in Docker Images zu transformieren und diese anschließend in einer Container Registry abzulegen. Da Docker verwendet werden soll, wird während des Erstellprozesses "Dockeräls Format selektiert und als Quelle wurde "GitHub (Cloud Build-GitHub-Anwendung)"verwendet, um den Build Prozess mit dem GitHub Repository zu verbinden. Im Anschluss muss eine Authentifizierung für dieses Repository durchgeführt werden. Hierbei fordert Google die erforderlichen Berechtigungen an, sodass keine weitere Maßnahmen auf GitHub erforderlich sind. Es werden diverse Events wie "Push to Branchöder "Pull Requestßum initiieren des Builds angeboten. Die Entscheidung hierfür sollte auf den jeweiligen Team- beziehungsweise Unternehmensrichtlinien basieren, da dies eine kollektiv entschiedene Konvention sein sollte. Abschließend muss noch eine Konfiguration angegeben, in der Ausarbeitung wurde hierfür ein Dockerfile verwendet.
Sofern man große Projekte oder Software mit umfangreichen Abhängigkeiten bauen möchte kann es vorkommen, dass Zeitüberschreitungen auftreten. Im weiteren Verlauf der Arbeit wird ersichtlich, dass man für ein Continuous Deployment einen speziellen Build Prozess erzeugen muss. (todo: Validieren)

(todo: Verschieben in anderes Kapitel) Es ist vonnöten, dass man eine YAML Datei schreibt und diese dem Build Prozess zur Verfügung anfügt. Optimalerweise wird diese auch dem Repository hinzugefügt, da sie genau wie reguläre Code Dateien auch einen Veränderungsprozess durchlaufen. (todo: Quelle)

5.2 Docker Registry

Es wurde während des Lesens der Dokumentation erkannt, dass es eine aktuellere Technologie gibt, die Google für Docker Registries vorsieht. Obgleich der Name "Google

Container Registryßuggeriert, dass dieser Dienst für das Ablegen von Images verwendet werden sollte, empfiehlt die Google Dokumentation die Verwendung der aktuelleren Ärtifact Registry". Martinez formuliert diesbezüglich: "[The Google Cloud Platform Artifact Registry] will [..] replace our current [Google Cloud Registry] in the future [...]"**artifactRegistryReplacesGCP** (todo: Sollte ich die Artifact Registry auch noch in den Werkzeugen ausformulieren? Ja oder?)

Grundsätzlich kamen zunächst

5.3 Bereitstellung der Anwendungen

- Ausgangslage: Es liegt eine ASP.NET 5

5.4 Backend

Grundsätzlich: Betrieb von Frontend + Backend im Docker Container, Datenbank entweder Cloud Run

5.4.1 CORS Anpassungen

5.4.2 Anpassungen der Konfigurationsdateien

5.5 Datenbank mittels Compute Engine

Die Bereitstellung der Datenbank erfolgte zunächst auf einer Compute Engine Instanz. Die erstellte virtuelle Maschine wurde mit einer Debian und einer Docker Installation versehen. Da im Softwareprojekt die Datenbank mit einer MSSQL Datenbank umgesetzt wurde, sollte auch der Betrieb mit einer MSSQL Datenbank stattfinden. Um dies zu realisieren wurde das offizielle Microsoft SQL Server Docker Image vom Docker Hub instanziiert. Dies lies sich mit dem Befehl bewerkstelligen. Damit man sich anschließend mit der Datenbank durch ein Client Tool wie SQL Server Management Studio verbinden konnte musste noch der Port 3306 in der Google Cloud Firewall für die erstellte virtuelle Maschine freigegeben werden. Sodann war es möglich, sich mit den festgelegten Benutzerdaten und der externen IP Adresse sowie dem Port gegen die Datenbank zu verbinden. Die Notation der Adresse erfolgt hierbei grundsätzlich im Format ïp,port". Sowohl das Schema als auch die Daten ließen sich nun via Insert Skript einspielen. //

Je nach Technologie Stack kommen auch andere Datenbanken in Frage. Sofern es ein geeignetes Docker Image hierfür gibt, kann diese Datenbank ebenfalls hiermit betrieben werden. (todo: Zeitform?) Da in dem zugrunde liegenden Softwareprojekt bereits eine MSSQL Datenbank vorlag, wurde auch ebenjene für diese Ausarbeitung verwendet. Als Alternativen würden sich für das vorliegende Szenario entweder andere relationale Datenbanken anbieten. Sofern die Anwendung eine höhere Skalierbarkeit bieten soll, k Den

grundsätzlichen Betrieb einer Datenbank in einem Container gilt es jedoch zu hinterfragen. Laut Heddings ist Docker nicht für ßtateful services"wie Datenbanken vorgesehen. Sämtliche Daten sind laut ihm flüchtig und werden vernichtet, sobald der Container gelöscht wird. Dieses Problem lässt sich jedoch gemäß Heddings in gewissem Maße umgehen, indem Docker Volumes für die Persistierung der Daten verwendet werden./cite(heddings-database-docker)

Die bisherige Umsetzung erfolgte auf einer MSSQL Datenbank. Diese liegt als Docker Image im Docker Hub vor. MySQL oder MariaDB, weil es hierzu Docker Images gibt. Als Alternativen können sich hier auch NoSQL Datenbanken anbieten. Für das Caching gleichen Anfragen und Sessions kann sich eine Redis DB anbieten, da bei jeder Anfrage, die eine gültige Session erfordert, eine Datenbankabfrage vonnöten ist. Für das Speichern der Buchungsdaten lohnt sich vermutlich eine dokumentenbasierte Datenbank. Der Vorteil von dokumentenbasierten Datenbanken wäre eine zu erwartende Umgehung von DB-Locks bei den Schreibintensiven Operationen durch die Import Funktionalität. Im vorliegenden Fall wurde aufgrund der Einheitlichkeit eine relationale Datenbank gewählt.

Erkenntnis: Das ist grundsätzlich nicht vorgesehen, es gibt eine native Lösung: Das Cloud SQL. Es wurde zunächst versucht mit einer kostenlosen VM bereituzustellen. Hierbei konnte nach umfangreichen Anstrengungen und Recherchen kein Ergebnis erzielt werden. Es sei an dieser Stelle jedoch kritisch zu vermerken, dass die Expertise des Autors im Netzwerkbereich nicht umfangreich genug waren, um eine korrekte Lösung zu implementieren. Bei ausreichenden Kenntnissen besteht die Möglichkeit, dass hier dennoch eine korrekte Umsetzung erfolgen kann. Google sieht vor, dass man die Cloud-native SQL Lösung "Cloud SQL"verwendet.(todo: belegen, wenn das nicht geht dann sollte das irgendwie umformuliert werden oder raus.) Die Verbindung eines Cloud Run Service mit einer Cloud SQL Datenbank liegt umfangreich von Google dokumentiert vor.

5.5.1 Netzwerkprobleme

5.5.2 CloudSQL

5.5.3 Alternative Möglichkeiten

5.6 Frontend

(ggf. zusammen mit Backend?)

(

5.7 Aufgetretene Probleme

) vermutlich eher beiläufig

6 Evaluation und Ergebnisse

Stärken und Schwächen herausstellen Verbesserungsfähige Aspekte benennen - App.Config auslagern in Environment Variables Lösungsansätze aufzeigen

Es wurden keine Sicherheitsaspekte betrachtet. In Unternehmenslösungen sollten beispielsweise Rechte- und Rollenkonzepte erarbeitet und angewandt werden.

7 Zusammenfassung

8 Ausblick