Architecture as Code für moderne Cloud-Anwendungen

Studiengang: MAS Information Technology

Traditionelle monolithische Architekturen weichen zunehmend verteilten Systemen, in denen Microservices als de-facto Standard gelten. Moderne Architekturen verteilter Systeme bringen jedoch auch neue Herausforderungen mit sich. Um diesen entgegenzuwirken, wurde in dieser Masterthesis das Paradigma Architecture as Code (AaC) evaluiert. Wie positioniert sich dieser Ansatz im Vergleich zu klassischen Microservices-Architekturen, und welche Herausforderungen kann es adressieren?

Ausgangslage

Moderne Softwarearchitekturen orientieren sich zunehmend an dem Prinzip, grosse monolithische Anwendungen in zahlreiche kleine, eigenständige Microservices zu zerlegen. Dieser Ansatz hat sich in der Systementwicklung etabliert und bietet viele Vorteile, darunter unabhängige Skalierung, entkoppelte Entwicklung von Systemkomponenten und schnellere Releasezyklen, was eine kürzere Time-to-Market ermöglicht. Allerdings bringt diese Flexibilität auch Herausforderungen mit sich. Dazu gehören ein komplexes Abhängigkeitsmanagement, Versions- und Interaktionsmanagement zwischen Komponenten sowie eine höhere Verwaltungskomplexität. Diese organisatorischen und technischen Hürden haben in der Branche eine Debatte darüber ausgelöst, ob Microservices langfristig die beste Architekturstrategie darstellen.

Architecture as Code Paradigma

Das Architecture as Code (AaC) Paradigma erweitert die Prinzipien von Infrastructure as Code (IaC), indem nicht nur Infrastruktur, sondern die gesamte Systemarchitektur – einschliesslich der Geschäftslogik – in einer gemeinsamen Codebasis abgebildet werden. Für Architecture as Code existiert keine einheitliche Definition in der Praxis. Der in dieser Arbeit evaluierte Ansatz orientiert sich an den Konzepten von Gregor Hohpe, einem einflussreichen Experten für Softwarearchitekturen. Zentral ist dabei der Einsatz von Cloud Development Kits (CDK), mit denen Architekturkomponenten als wiederverwendbare Bausteine definiert werden können. Dies ermöglicht eine explizite und transparente Modellierung verteilter Systeme.

Methodik

Um das Potenzial von Architecture as Code zu untersuchen, wurde ein Proof of Concept entwickelt, welcher das Paradigma praktisch umsetzt. Als Referenz diente eine klassische Microservices-Applikation, die im Rahmen des Proof of Concept unter Anwendung des Paradigmas in eine Serverless-Architektur transformiert wurde. Zur systematischen Bewertung wurde ein vergleichender Ansatz gewählt. Zunächst wurden die architektonischen Eigenschaften der ursprünglichen Microservices-Lösung analysiert und dokumentiert. Anschliessend wurde die gleiche Anwendung unter Berücksichtigung der Architecture as Code Prinzipien neu implementiert. Zur Evaluation wurden qualitative und quantitative Kriterien definiert, um zu untersuchen, inwiefern das Paradigma bestehende Herausforderungen klassischer Microservices-Architekturen adressieren kann.

Ergebnisse

Die Evaluation des Architecture as Code Paradigmas hat gezeigt, dass es besonders in einer Serverless-Architektur mehrere zentrale Vorteile entfaltet, da Infrastruktur und Geschäftslogik stärker verschmelzen. Durch die explizite Modellierung der Architektur in Code werden architektonische Entscheidungen nachvollziehbar dokumentiert und Abhängigkeiten zwischen Komponenten transparenter gestaltet. Zudem ermöglicht CDK durch das Construct Programming Model (CPM) die Definition einer modularen und wiederverwendbaren Architekturstruktur, wodurch sich Systeme flexibler gestalten lassen. Allerdings löst das Paradigma nicht alle Herausforderungen klassischer Microservices-Architekturen. Daher sollte es primär als ergänzendes Konzept betrachtet werden, das insbesondere im Kontext von Serverless-Architekturen Vorteile bietet, indem es typische Herausforderungen verteilter Systeme direkt adressiert. Die Ergebnisse legen nahe, dass sich Architecture as Code gerade in diesem Kontext als vielversprechender Ansatz erweist, um die Komplexität zu reduzieren und die Anwendungsarchitektur transparenter abzubilden.

Marc Touw
marc.touw@gmail.com