Applikationsarchitektonischer Einfluss eines Umstiegs zur Continuous Deployment Methodik

Studiengang: MAS Information Technology

Hauptbestandteil der Arbeit ist die Analyse von Auswirkungen eines zukünftigen Umstiegs zur Continuous Deployment Methodik innerhalb einer bestehend Applikationslandschaft. Der Fokus der gesamten Arbeit liegt dabei auf Applikationsarchitekturthemen, wobei die erarbeitete Zielarchitektur nun als Grundlage für die bevorstehende Umrüstung dient.

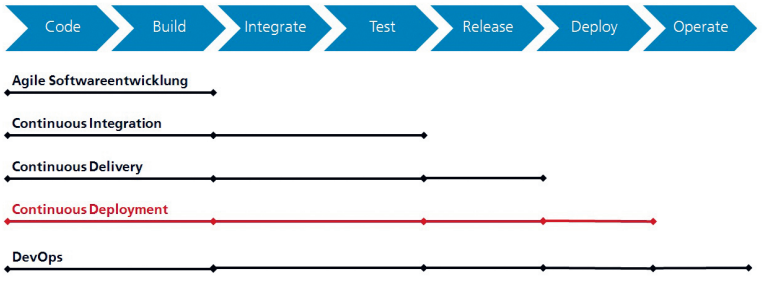
Peter K. Mäder

Im Zuge der ersten Auseinandersetzung mit der Bedeutung von Continuous Deployment, scheint zu aller erst keine offensichtliche Verbindung zu einer Applikationsarchitektur zu bestehen. Schliesslich wird primär das Ziel verfolgt, Software möglichst effizient und effektiv in die Produktion zu bringen, was zunächst nur nach einer werkzeugtechnischen Frage aussieht. Doch die Leitlinie der Methodik lautet Automatisierung der ganzen Auslieferungskette. Die Auslieferung soll kontinuierlich, schnell und qualitativ hochwertig erfolgen. Im Gegensatz zum Continuous Delivery Ansatz, wird nicht zu einem gewählten Zeitpunkt, sondern nach jeder, im Gefäss des Versionsverwaltungswerkzeugs platzierten, Softwareänderung vollautomatisch in die Produktion ausgeliefert. Unter der Bedingung, dass zuvor alle definierten Qualitätsprüfungen erfolgreich passiert wurden.

Damit die Methodik umgesetzt und eine unterbrechungsfreie, ausfallsichere Auslieferung ermöglicht werden kann, sind mehrere applikationsarchitektonische Vorkehrungen nötig. So beginnt der Einfluss bereits bei der Testbarkeit. Durch den Anspruch, die Qualitätssicherung komplett zu automatisieren, müssen gut testbare, modulare Applikationen entworfen werden. Um ein schnelle Auslieferung erreichen zu können, müssen die entsprechenden Einheiten, sowohl relativ klein, als auch unabhängig test-, auslief- und ausführbar sein. Eine Ausrichtung der Applikationsarchitektur in Richtung cloud-nativen Anwendungen und die Umsetzung der bewährten 12-Faktor-App-Regeln, ermöglicht ausserdem horizontale Skalierung, Canary-Releasing/-Testing und Entwicklung von resilientem Verhalten.

Die zielführendste Methode zur Aufteilung einer Applikation in unabhängige Komponenten, bietet der Einsatz von Domain-driven Design. Gerade dessen Ausrichtung auf die Fachlichkeit verspricht bei neu umzusetzenden fachlichen Anforderungen, eine möglichst geringe Anzahl an zu verändernden Komponenten, die unabhängig voneinander ausgeliefert und ausgeführt werden können. Beim Entwurf der Komponenten sollte ausserdem eine Lose Kopplung und ein hochgradige Geschlossenheit gegenüber Änderungen im Vordergrund stehen, um den Softwarenpassungsaufwand so klein als möglich zu halten.

Um ideale Voraussetzungen für die bestehenden Applikationen zu schaffen, sind ausserdem unter anderem Konzepte wie Feature-Toggling, Consumer-driven Contracts, Expand/Contract-Pattern, Circuit Breaker Pattern und ausgebautes fachliches und technisches Monitoring zu berücksichtigen.



Begriffseinordnung in der Auslieferungskette