OPITZ CONSULTING

Überraschend mehr Möglichkeiten

Workshop: Continuous Delivery Kickstart

Mehr Verlässlichkeit im Release-Prozess mit modernen Entwicklungs- und Testmethoden!

Vom Commit einer Code-Änderung bis zum Release ist es meist ein weiter Weg: Änderungen werden getestet, Konfigurationen geändert, Patches eingespielt und Datenbanken migriert. Oft enthält der Ablauf viele manuelle Schritte, die nur von wenigen Personen beherrscht werden und fehleranfällig sind. Das Ergebnis sind lange Release-Zyklen und „Bauchschmerzen" vorm nächsten Release. Dabei könnte Ihre nächste Software-Lieferung nur einen Klick entfernt sein: Continuous Delivery bringt durch moderne Entwicklungs- und Testkonzepte sowie durch einen hohen Automatisierungsgrad mehr Sicherheit und Qualität in die Lieferstrecke. Einen leichten Einstieg bietet Ihnen jetzt unser Workshop „Continuous Delivery Kickstart".

Mit Continuous Delivery Risiken reduzieren

Mit Continuous Delivery ist die Entwicklungsabteilung zu jeder Zeit in der Lage, den aktuellen Stand einer Anwendung auszuliefern. Dazu wird der manuelle Aufwand zur Erstellung eines Releases auf ein Minimum reduziert.

Es werden schnellere Releasezyklen erreicht, die überschaubarere Feature-Inkremente hervorbringen. So können Sie Fehler schneller zuordnen und Kunden-Feedback früher einholen. Frühzeitige und regelmäßige Deployments in produktionsähnliche Umgebungen verhindern „Überraschungen" beim Release.

Doch nicht jedes Inkrement muss zu einem Deployment auf einer Produktionsumgebung führen. Der Release-Prozess auf die Produktionsumgebung erfolgt mit den gleichen Skripten und Werkzeugen wie auf die Abnahmeumgebung. Der komplette Releaseprozess wird reproduzierbar und vermeidet Fehlerquellen durch manuelle Schritte. So verringert Continuous Delivery die Risiken eines Releases in der Produktion.

Eine komplette Automatisierung ist notwendig, um später auch Continuous Deployment zu unterstützen und viel öfter als heute auf die Produktionsumgebung zu releasen. Firmen wie Flickr oder Facebook führen mehrmals täglich Produktionsreleases durch. Dadurch, dass die Änderungen klein sind, reduzieren Sie das Risiko einer Auslieferung.

Automatisierung schafft Verlässlichkeit

Um jederzeit lieferfähig zu sein, muss der Prozess zur Erstellung eines Releases weitestgehend automatisiert werden. Neben der Bereitstellung der Softwarepakete gehört dazu auch die Automatisierung von Infrastrukturänderungen wie Netzwerkkonfiguration, Patches und Datenmigration. Sogar das vollständige Aufsetzen einer Umgebung mit Betriebssystem, Services, Middleware und Anwendung lässt sich mit Techniken wie virtuellen Umgebungen und Provisionierungs-Tools automatisieren.

So kann mit wenig Aufwand jeder Systemstand zum Test oder im Rahmen eines Rollback wieder hergestellt werden. „Kopfwissen" wird in automatisierte und prüfbare Abläufe überführt, die verlässliche Ergebnisse liefern.

Mehr Sicherheit mithilfe einer Build Pipeline

Der automatisierte Ablauf manifestiert sich in einer Build Pipeline, die wie das Fließband in einer Fabrik, den Pfad visualisiert, den jedes Lieferartefakt durch den Erstellungsprozess nehmen muss. Bei der Definition der Build-Pipeline wird jeder Herstellungsschritt wie Kompilierung, Unit-Test, Doku-Generierung oder Release in eine Umgebung einer Stage zugeordnet. Das Erreichen einer neuen Stage stellt gleichzeitig ein Qualitätssiegel für die Anwendung dar.

Fact Sheet

So kann z. B. sichergestellt werden, dass nur Software in die Testumgebung deployt wird, die zuvor alle automatischen Tests und statischen Code-Analysen bestanden hat. Falls eine Stage nicht erfolgreich durchläuft, wird dies direkt zurückgemeldet und darauf aufbauende Stages werden nicht gestartet.

Entwickler und Tester können sich jederzeit einen Überblick über laufende und vergangene Builds und deren Status in der Build Pipeline verschaffen, und bei Bedarf Builds, die z. B. alle Tests durchlaufen haben, per Button-Klick in eine eigene Umgebung deployen.

Nachvollziehbar und reproduzierbar

Die Überführung manueller Abläufe in technische Artefakte bewirkt, dass jede Änderung am Systemverhalten einer Änderung in der Build Pipeline zugeordnet werden kann. Dass die Elemente der Pipeline in einem Versionskontrollsystem abgelegt und alle Änderungen einem Pipeline-Lauf zugeordnet werden können, macht den Effekt jeder Änderung nachvollziehbar. Dazu müssen aber auch wirklich alle Bestandteile eines Builds versioniert vorliegen; dazu gehörten neben der Anwendung auch Setup-Scripte, Konfigurationen und Software-Basispakete.

Dank des automatischen Ablaufs und den versionierten Ständen der Pipeline kann ein Entwickler jedes Release inklusive Infrastruktur-Setup reproduzieren. Auch das Zurückrollen der Anwendung zu einem früheren Stand und das exakte Reproduzieren der Produktionsumgebung für die Nachstellung eines Fehlers sind so möglich.

Infrastruktur als Code

Beim Continuous Deployment wird Wert darauf gelegt, nicht nur das Software Deployment, sondern auch Setup und Konfiguration der Umgebungen zu automatisieren. Denn jede manuelle, nicht versionierte Änderung birgt die Gefahr eines nicht nachstellbaren Fehlers.

Infrastruktur-Management-Tools wie Puppet, Chef oder Ansible ermöglichen dem Software Developer, das Infrastruktur-Setup in verständlichen Skripten zu beschreiben, diese wie Anwendungscode zu behandeln und mit in die Build Pipeline aufzunehmen. Die entstehenden Skripte können dann einfach wiederverwendet und auf andere Systeme gespielt werden. Kleine Abweichungen, die bei einem manuellen Vorgehen leicht passieren, können so ausgeschlossen werden.

Problemen beim Produktiv-Deployment beugen die Entwickler vor, in dem sie die gleichen Setup- und Deploy-Skripte nutzen, die sie erfolgreich in einer Testumgebung eingesetzt haben.

Workshop: Continuous Delivery Kickstart

Sie möchten ein neues Projekt aufsetzen und darin über Continuous Integration hinausgehen? Sie sind sich noch nicht ganz sicher, ob Continuous Delivery das Richtige für Ihre Entwicklungsabteilung ist?

Unter Berücksichtigung Ihrer aktuellen Buildprozesse und CI-Infrastruktur zeigen wir Ihnen, wie eine Continuous Delivery Strategie bei Ihnen aussehen würde und mit welchen Schritten und Maßnahmen Sie von Continuous Integration in Richtung Continuous Delivery gehen können.

Vielleicht fragen sich Ihre Administratoren, welche Auswirkungen Continuous Delivery auf Ihre Arbeit haben wird? Wir zeigen Ihnen, wie sich Continuous Delivery und ITIL ergänzen können.

Buchen Sie unseren eintägigen Workshop mit Impuls- und Überblicksvorträgen zu wichtigen Aspekten von Continuous Delivery, und erarbeiten Sie mit uns die nächsten Schritte zur Einführung von Continuous Delivery!

Der Workshop auf einen Blick

Das erwartet Sie in unserem eintägigen Workshop:

- Wichtige Aspekte von Continuous Delivery
- Wesentliche Werkzeugbestandteile aus Architektensicht
- Mittels Vagrant, Docker und Ansible Entwicklungs- und Testumgebungen bereitstellen
- Geeignete Werkzeuge und ihr Einsatzgebiet im Applikationslebenszyklus
- Kosten und Nutzen einer Continuous Delivery Pipeline in Ihrem konkreten Anwendungsfall
- Erarbeitung nächster Schritte zur Einführung von Continuous Delivery

Sprechen Sie uns an:
Richard Attermeyer,
Senior Solution Architect

+49 201 892994-0
richard.attermeyer@opitz-consulting.com

Weitere Leistungen zu DevOps und Continuous Delivery:
www.opitz-consulting.com/devops