



Quo vadis Continuous Delivery?



Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Steffen Schluff

Version: 1.0

Ihr Sprecher

Steffen Schluff

Trainer, Berater, Entwickler



Schwerpunkte
*Open Source Tooling
Continuous Delivery
Clean Code*

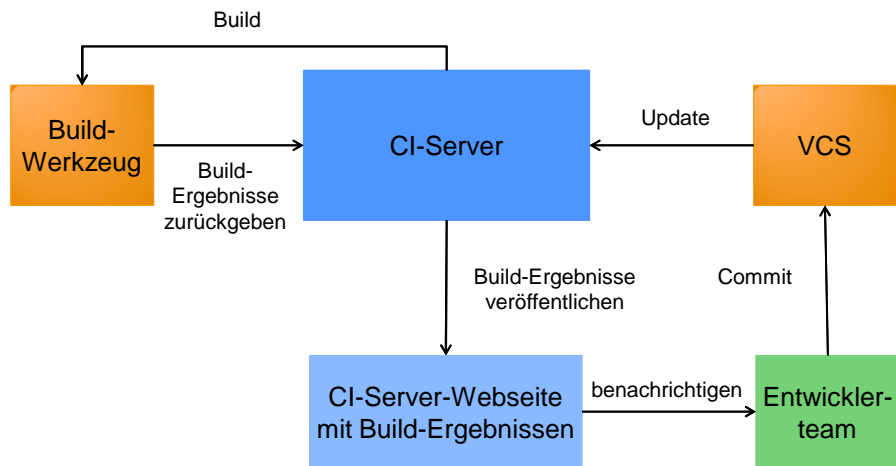
Gliederung

- Einleitung
- Continuous Delivery
- Tool Time
- “Continuous Integration was not designed for Continuous Delivery”
- Zusammenfassung

Gliederung

- **Einleitung**
- Continuous Delivery
- Tool Time
- “Continuous Integration was not designed for Continuous Delivery”
- Zusammenfassung

Been there, done that (1)

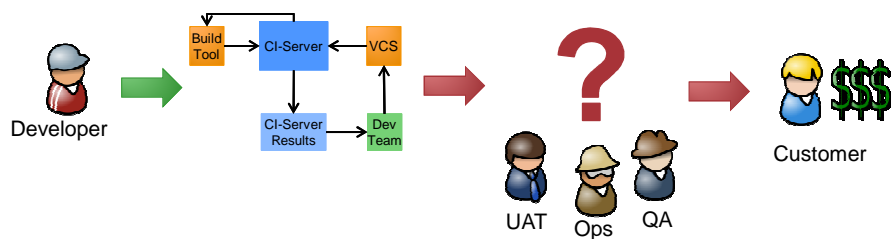


Been there, done that (2)

- „Daily Build and Smoke Tests“ sind schon ein alter Hut
 - Erste Veröffentlichung von Steve McConnell im Jahre 1996
 - Thema war bereits davor schon bekannt
- „Continuous Integration“ Artikel von Martin Fowler im Jahre 2000
 - Themenbereich bekam einen klingendem Namen
 - Definition „Key Practices“ (Automate the build, Make it self-testing, ...)
 - Erste Bereitstellung von „fertigen“ Tools (CruiseControl)
- „Continuous Integration“ gehört heute zum guten Ton
 - Wahlfreiheit zwischen diversen Servern (Jenkins, Hudson, Bamboo, ...)
 - Definition von Best Practices, Patterns und Anti-Patterns
 - Probleme der zweiten Generation: Testlaufzeiten, Virtualisierung, ...

Bis hierher sollst du kommen und nicht weiter

- Continuous Integration konzentriert sich auf Entwickler
- CI Server grün, Commit erfolgreich, Entwickler glücklich
- Aber kein Bereitstellen zum Testen und keine Produktivsetzung



(<http://www.public-domain-photos.com/free-cliparts>)

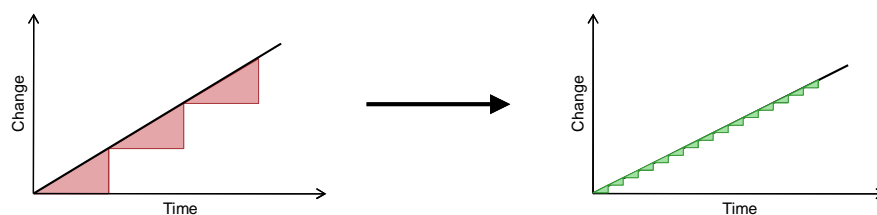
Denkt denn niemand an den Kunden?

- Kunde möchte Verfügbarkeit seiner Funktionalität
 - Kein Interesse, ob CI Server rot, grün, gelb oder blau ist
- Zwischen gutem CI Build und Kunden Verfügbarkeit liegt Release
 - Release Schritt oft nicht so gut beherrscht wie CI Ökosystem
- Release Modell „Big Bang“™ (alias der Klassiker)
 - Manuell, Zeitintensiv, Kompliziert, viele Beteiligte, Fehleranfällig

Don't do that then!

- Seltene dafür aber große Releases als Konsequenz
 - Organisatorische Vermeidungsreaktion
- Zugleich hoher Stressfaktor bei ungeplanten Releases
 - Wenig Übung führt zu hohen Fehlerrate bei Hotfixes
- Frustrierend für den Kunden
 - Auch einzelne Features brauchen scheinbar sehr lange
- Wir wollen den Kunden nicht frusten, hier muss sich etwas ändern!

Stay on target



"[...] the ability to rapidly, reliably and repeatedly push out enhancements and bug fixes to customers at low risk and with minimal manual overhead."

(http://en.wikipedia.org/wiki/Continuous_delivery)

Buzzword (1)



Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

First principle behind the Agile Manifesto
(<http://agilemanifesto.org/principles.html>)

Buzzword (2)



*Our highest priority is to satisfy the customer through early and **continuous delivery** of valuable software.*

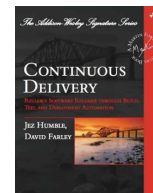
First principle behind the Agile Manifesto
(<http://agilemanifesto.org/principles.html>)

Gliederung

- Einleitung
- **Continuous Delivery**
- Tool Time
- “Continuous Integration was not designed for Continuous Delivery”
- Zusammenfassung

Continuous Delivery

- *„Continuous Delivery is not Continuous Integration. Continuous Delivery is being in the position to ship your product whenever you want, day or night.” (Neal Ford)*
- Frühere Begriffsverwendung und Wurzeln
 - „Agile Manifesto” (2001)
 - „Deployment Pipeline” (2004/2005)
- Gleichnamiges Buch von Jez Humble & David Farley
 - Eigentliche Begriffsprägung (2010)
- Schwerpunktthemen „Automation“ und „Collaboration“



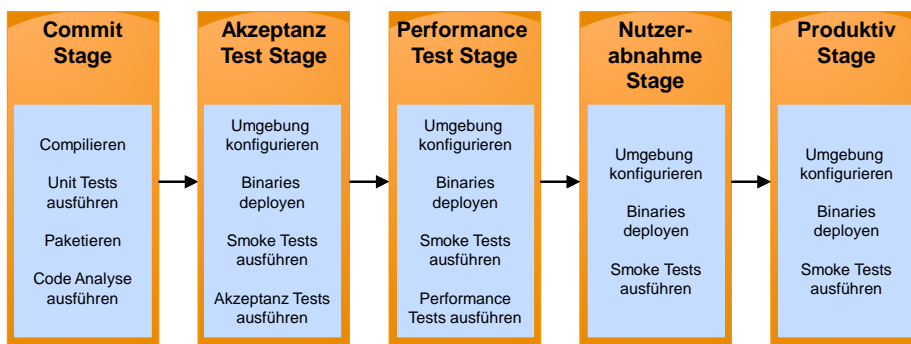
Continuous Delivery – Kerngedanken

- “Create a Repeatable, Reliable Process for Releasing Software”
 - “If It Hurts, Do It More Frequently, and Bring the Pain Forward”
 - “Automate Almost Everything”
 - “Keep Everything in Version Control”
- “Everybody Is Responsible for the Delivery Process”
- “Done Means Released”
- Und wie soll das alles umgesetzt werden?

(Nach „Continuous Delivery“/J. Humble, D. Farley)

Deployment Pipeline – Ein erster Blick

- Zentrale Abstraktion „Deployment Pipeline“
 - Visualisierung aller Prozessteile für alle Beteiligten
 - Verbessertes Feedback während der Ausführung
 - Möglichkeit eines vollautomatischen Releases in alle Umgebungen

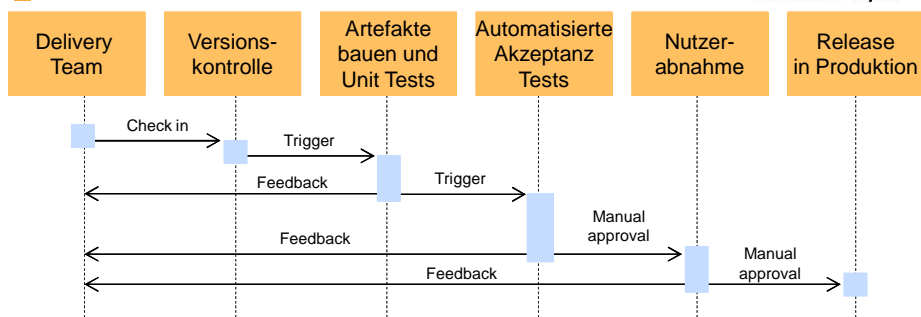


(Nach „Continuous Delivery“/J. Humble, D. Farley)

Deployment Pipeline – Bestandteile

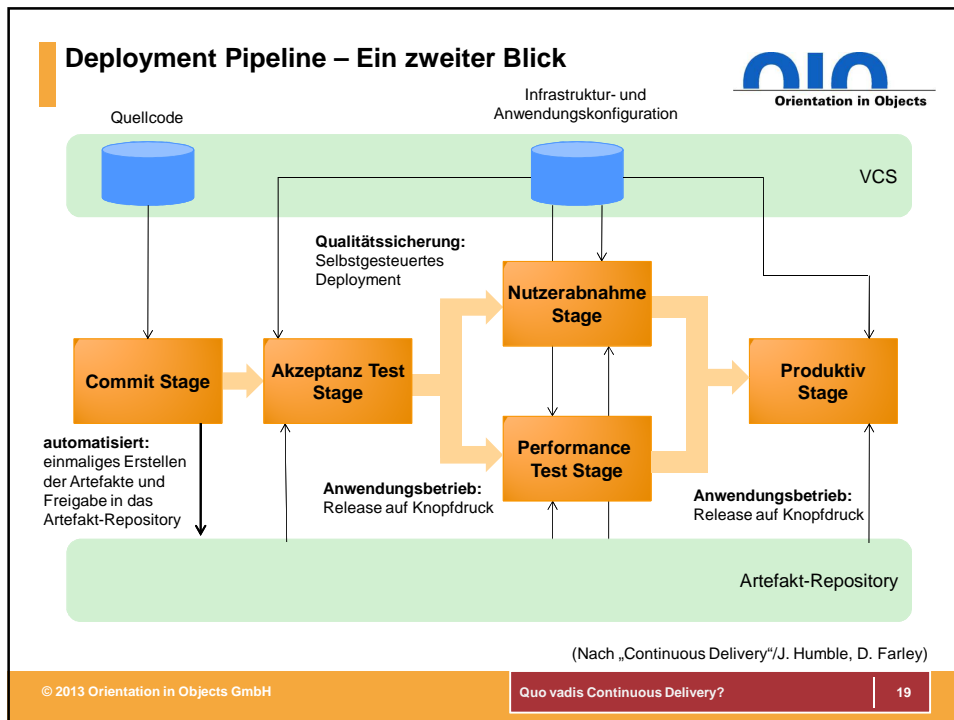
- Die **Deployment Pipeline**
 - Macht Status der Produktentwicklung sichtbar
 - Liefert Feedback zu jeder Änderung
 - Technisch-konzeptuelle Basis des Release Prozesses
- Die Pipeline besteht aus einer Folge von **Stages**
 - Commit Stage als zentrales Eingangs-Gate
 - Typische Stages: UAT, Performance Tests, Production Deployment
 - Stages verbunden durch Trigger (automatisch oder manuell)
- **Jobs** sind die Bausteine der Stages
 - „Unit of Work“
 - Bestehen aus Tasks wie Build, Deploy, Copy, Test, ...

Deployment Pipeline – Sequenzdiagramm



- Jede Ressourcen Änderung startet neue Pipeline Instanz
- Erste Stage produziert alle Artefakte
- Durchlaufen aller Stages bis Fehlschlag („Stop the line“) oder ...
- Pipeline Ende erreicht ist (letzte Stage führt Deployment aus)

(Nach „Continuous Delivery“/J. Humble, D. Farley)



- ## Continuous Delivery – Prinzipien & Methoden (1)
- Fortlaufende Optimierung
 - In Verantwortung aller Beteiligten (Development, Operations, ...)
 - Artefakte
 - Werden **einmal** gebaut und in einem Artefakt Repository verwaltet...
 - und allen Stages zur Verfügung gestellt
 - Ziel ist identisches Deployment in allen Umgebungen
 - Umgebungsspezifika durch eigene Konfigurationen
 - Configuration-Management
 - Basis für einmalig erstellte Artefakte
 - Umfasst Software und Infrastruktur („Infrastructure as code“)
 - Konfigurationen werden versioniert
- © 2013 Orientation in Objects GmbH | Quo vadis Continuous Delivery? | 20

Continuous Delivery – Prinzipien & Methoden (2)



- Automatisierung
 - So umfangreich wie möglich
 - Umfasst auch alle Aspekte der Infrastruktur (inklusive OS)
 - Prägung durch Development und Operations
- Tests
 - Basis für Automatisierung und Pipeline Processing
 - Geben Sicherheit für erfolgreiche Änderungen
 - Smoke Tests speziell für Deployment
- Monitoring
 - Ermöglicht Feedback für Operating
 - Basis für fortlaufende Optimierung

Alle Theorie ist grau



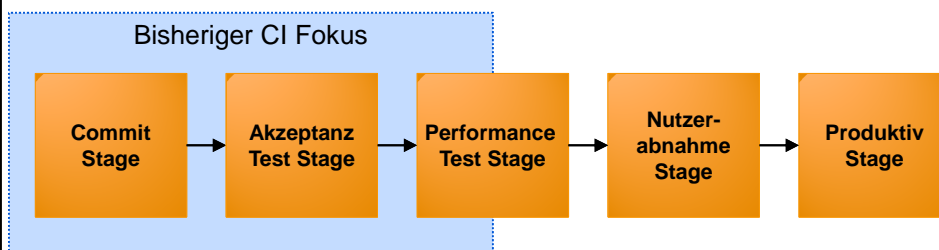
- Erfolgsfaktoren von Continuous Integration rückblickend waren
 - Eingängiger Name
 - Konkrete „Key Practices“
 - Einsetzbare Tools
- Zum Erfolg fehlt Continuous Delivery also noch ein gutes Tool
 - Erster Impuls oft selbstgemachte Lösungen („Home grown“), aber ...
 - häufig schnell veraltet bei schlechtem Kosten-Nutzen Verhältnis
- Jedes Projekt hat in der Praxis seine eigenen Spezialitäten
 - Web App vs. Mobile vs. Rich Client, Programmiersprache, OS, usw.
 - Somit sind auch Continuous Delivery Umsetzungen verschieden
- Gibt es also gar kein Continuous Delivery Tool?

Gliederung

- Einleitung
- Continuous Delivery
- **Tool Time**
- “Continuous Integration was not designed for Continuous Delivery”
- Zusammenfassung

Continuous Delivery – Tooling (1)

- *„The deployment pipeline has its foundation in the process of continuous integration and is in essence the principles of continuous integration taken to its logical conclusion.“ (J. Humble, D. Farley)*



Continuous Delivery – Tooling (2)



- CI Server werden bereits für alle möglichen Projekt Arten eingesetzt
 - Und integrieren dabei diverse Tool Arten (Build, Test, Lint, Coverage, ...)
- Tools für einzelnen Continuous Delivery Konzepte sind vorhanden
 - Artefakt Repositories (CI-Server eigene Repos, Maven, ...)
 - „Infrastructure as code“ (Puppet, Chef, ...)
- Continuous Integration wird Continuous Delivery Server durch ...
 - Integration der neuen CD spezifischen Tool Arten
 - Bereitstellung einer Deployment Pipeline (samt Stages, Jobs, Triggern)

Namen sind Schall und Rauch



- Jeder gängige CI Server bietet Pipeline Bausteine an
 - Und ist somit ein CD Server
- Konkrete Namen können variieren
 - Teils historische Gründe, teils Abgrenzung zur Konkurrenz
- Exemplarische Beispiele („*Your Mileage May Vary*“)
 - **Jenkins**: *Build Jobs, Build Steps, Post-build Actions, diverse Plugins*
 - **Go**: *Go Pipelines, Stages, Jobs, Tasks*
 - **Bamboo**: *Build Plans, Stages, Jobs, Tasks*
- Erster logischer Schritt ist Visualisierung der Pipeline
 - Übersicht aller Pipelines eines CD Servers
 - Bisherige Aktivitäten einer Pipeline, i.e. vergangene Pipeline Instanzen

Pipelines Dashboard in Go



Pipelines PERSONALIZE ▾

MyApplication

my-app-web

Label: 39
Compare | Changes ▾
(Triggered by antony 14 days ago)
Passed: analyze

▶ ▶+ ||

my-app-middleware

Label: 55
Compare | Changes ▾
(Triggered by anushr 3 minutes ago)
Building: build

Previously: ■ Passed

▶ ▶+ ||

my-functional-tests

Label: 39-54
Compare | Changes ▾
(Triggered by changes 8 days ago)
Passed: functional-tests

▶ ▶+ ||

(<http://www.thoughtworks.com/products/docs/go/13.3/help/>)

© 2013 Orientation in Objects GmbH

Quo vadis Continuous Delivery?

27

Pipeline Activity in Go



application

Pipeline Activity PAUSE

	dev	ut	ft	perf
53 revision: 48 17 days ago Forced by anonymous				
52 revision: 48 17 days ago Forced by anonymous				
51 revision: 48 17 days ago Forced by anonymous				
50 revision: 48 17 days ago Forced by anonymous				

(<http://www.thoughtworks.com/products/docs/go/13.3/help/>)

© 2013 Orientation in Objects GmbH

Quo vadis Continuous Delivery?

28

Build Pipeline Plugin in Jenkins



The screenshot shows the Jenkins interface for a pipeline named "Build Pipeline: Foobar 3000". The interface displays a grid of pipeline stages for five different pipeline instances, numbered #11 to #15. Each instance has three stages: Commit Stage, Test Stage, and Deploy Stage. The stages are color-coded: green for successful completion, yellow for warning, and red for failure. The #13 Test Stage is highlighted in red, indicating a failure. The interface also includes a search bar, a "Suchen" button, and a "Run" button. The top right corner shows "AUTO-AKTUALISIERUNG AUSSCHALTEN".

You Can Look But You Better Not Touch (1)

- Reine Visualisierung für „passive“ Pipelines ausreichend
 - Neue Pipeline Instanzen entstehen durch Ressourcen Änderungen
- Aber Benutzerinteraktion ist Teil der Pipeline Idee („*Manual Trigger*“)
 - „*Manual Approval*“ oder „*Push Button Releases*“ als Variationen
- Allgemeine Bedienelemente ebenfalls nötig
 - Pipeline Instanz ohne Ressourcen Änderung „von Hand“ erzeugen
 - Bestehende Pipeline pausieren
- Pipeline Visualisierung wird zur Pipeline GUI

You Can Look But You Better Not Touch (2)

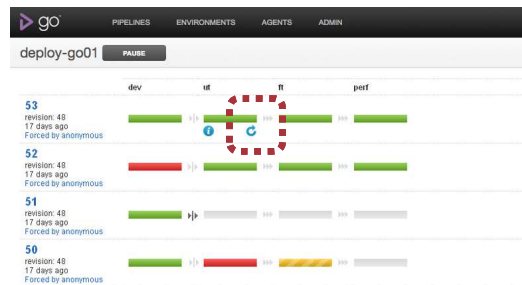


Wenn man jemandem den kleinen Finger reicht, ...

- Visualisierung der Deployment Pipeline macht Lust auf mehr
 - und schafft dadurch neue GUI Probleme
- Die üblichen Verdächtigen sind oft gleich ...
- Selektives „Überspringen“ von Stages
 - Deployen auf Produktion aber nicht auf Test Umgebung (z.B. bei Hotfix)
- Erneutes Ausführen einzelner Stages
 - Wiederholen automatisierter Tests (z.B. gleicher Code, neuer Agent)
 - Rollback oder Re-deploying (z.B. nach Produktions Crash)
- Zentralisierte Rechteverwaltung für manuelle Trigger
 - Bestimmte Gruppen verwalten bestimmte System (z.B. Prod Admins)

... nimmt er gleich die ganze Hand

- GUI der Pipeline Aktivität um entsprechende Buttons ergänzen
 - Jede Pipeline Stage (auch vergangene) wahlfrei ausführbar machen
- Aber erzeugt dies implizit dann eine neue Pipeline Instanz?
 - Und wie visualisiere ich das eigentlich?

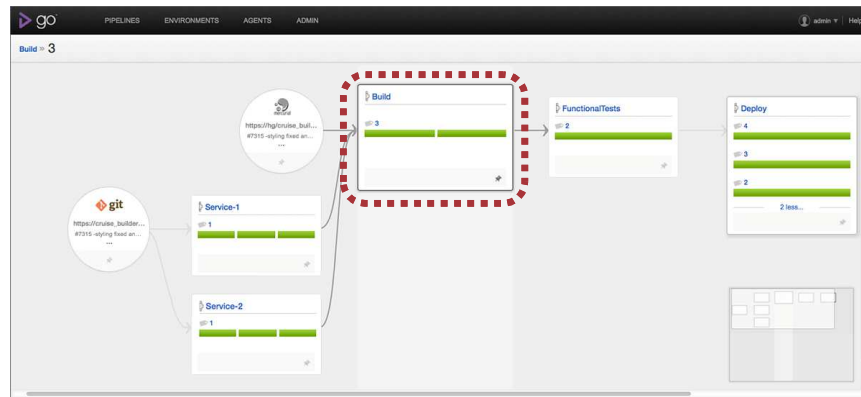


(<http://www.thoughtworks.com/products/docs/go/13.3/help/>)

Go Pipelines vs Deployment Pipelines

- *„A Go Pipeline does not necessarily map one-to-one with what is referred to as **the** automated deployment pipeline in continuous delivery literature. The automated deployment pipeline is essentially the end-to-end CD value stream. This end to end value stream is often better modeled using multiple Go Pipelines.“*
(<http://www.thoughtworks.com/insights/blog/how-do-i-do-cd-go-part-2-pipelines-and-value-streams>)
- Mehrere Go Pipelines bilden „Continuous Delivery Value Stream“
 - Und dieser entspricht der Deployment Pipeline
 - (Go) Pipelines sind plötzlich also auch Bausteine
- Go bietet „Value Stream Maps“ zur Visualisierung an
 - Zeigt Status *einer* bestimmten Go Pipeline Instanz ...
 - sowie alle dazu beitragenden Upstream Abhängigkeiten ...
 - und alle daraus entstandenen Downstream Abhängigkeiten

Value Stream Maps in Go



(<http://www.thoughtworks.com/products/docs/go/13.3/help/>)

Environments in Go

- Environments zuvor nur implizit vorausgesetzt (UAT, Produktion, ...)
- Go erlaubt explizites Anlegen und Verwalten von Environments
- Jede Pipeline gehört zu maximal einer Environment
- Erlaubt Darstellung in welcher Umgebung was läuft

Gliederung

- Einleitung
- Continuous Delivery
- Tool Time
- **“Continuous Integration was not designed for Continuous Delivery”**
- Zusammenfassung

WAT

- *„The deployment pipeline has its foundation in the process of continuous integration and is in essence the principles of continuous integration taken to its logical conclusion.“ (J. Humble, D. Farley)*
- *„Continuous Integration was **not** designed for Continuous Delivery. Continuous Integration is designed to keep developers informed about the state of the latest code changes.“ (Atlassian Bamboo Doc)*
(<https://confluence.atlassian.com/x/LgQrF>)
- Was will uns der Autor damit sagen?

He french fried when he should have pizza'd

- Deployment meist nicht durch Development sondern Operations
 - Unterschiedliche Personen mit unterschiedlichen Aufgaben
- CI Server Tooling ist auf Development ausgerichtet
- „Aber wir haben doch jetzt auch Pipelines im CI Server...“
 - Eine nachträgliche Abstraktion perfekt passend für...
 - „Der letzte erfolgreiche Build wird zeitnah deployt“



(Quelle: <http://dev2ops.org/2010/02/what-is-devops/>)

Wünsch Dir was (1)

- Ziel: Stabiler Betrieb aller Umgebungen und der deployten Software
 - Deployment ist nur Ops Teilaspekt, zentrale Fragen sind abstrakter
- Welche Releases existieren? Welche User Stories sind enthalten?
 - Welche Tests sind für das Release gelaufen? Haben User getestet?
 - Bei Bedarf: Welche konkreten VCS Commits sind enthalten?
- Was sind die Unterschiede zwischen zwei bestimmten Releases?
 - Nein zu „grep VCS Logs“ und „Klicken in Pipeline Activity Graphiken“
- Welche Umgebungen existieren und mit welchen Rechten?
 - Wie kann ich diese Rechte zentral rollenbasiert verwalten?

Wünsch Dir was (2)

- Welche Releases laufen in einer Umgebung?
 - Wie ist Historie der Releases in einer Umgebung?
 - Wer hat wann welches Release deployt? Wer hat es freigegeben?
- Wie führe ich ein Release Rollback in Umgebung durch?
 - Nein zu „Wiederhole erste Stage aus Pipeline rel2prod mit Revision 42“
- „Give deployments the first-class treatment“
 - Klare Schnittstelle zwischen Dev und Ops

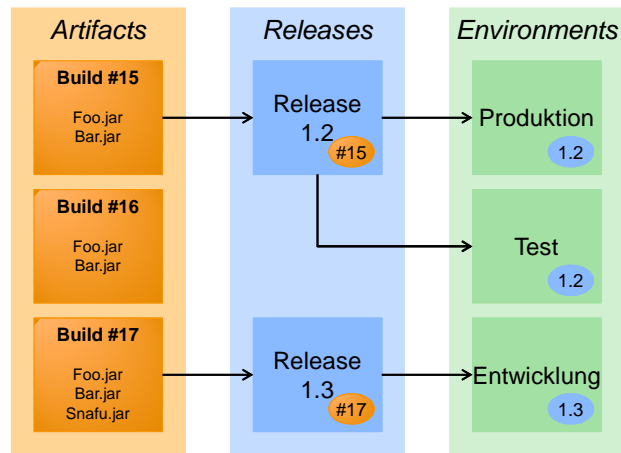


(Quelle: <http://dev2ops.org/2010/02/what-is-devops/>)

Deployment Projects in Bamboo (1)

- Build Plan erzeugt und testet Build Artefakte
 - „Klassische“ Continuous Integration (Dev Sicht)
 - Kapselt Build Prozess Details, fungiert nach außen als Artefakt Quelle
- Deployment Project abstrahiert zu deployende Software (Ops Sicht)
 - Deployment Project ist fest mit einem Build Plan verknüpft
- Deployment Projects definieren Environments
 - Laufzeitumgebungen mit Berechtigungen und Artefakt Deploy Skripten
- Für Deployment von Artefakten muss Release erzeugt werden
 - Bündelt Artefakte eines konkreten Builds zur deploybaren Einheit
 - Verbindung Deployment und Build Prozess (Dev Ops Brücke)

Deployment Project in Bamboo (2)



Demonstration

Deployment projects / Hello Webapp Deployment / Releases

Release: 1.0.5 **APPROVED**

Status Commits Issues Variables

Environment	Status	Deployment result	Completed	Trigger	Actions
Staging	SUCCESS	SUCCESS	22 Oktober 2013 02:59 PM	Manual run by Administrator	<ul style="list-style-type: none"> Created 5 hours ago Created by Administrator Reviewed by Administrator APPROVED A moment ago Deployment project: Hello Webapp Deployment Artifacts provided by #17 Hello Webapp > Core Release contents: Hello-War
Production	Never deployed	now at 1.0.2			

Commits tested by

The commits that were used to produce this release, were also built in the following build results

Build	Plan	Test results
#17	Hello Webapp > Core	2 passed



Gliederung

- Einleitung
- Continuous Delivery
- Tool Time
- “Continuous Integration was not designed for Continuous Delivery”
- **Zusammenfassung**

Zusammenfassung

- Kundenzufriedenheit erfordert Auslieferung von Software
 - Klassische Release Modelle sind zu unflexibel für moderne Entwicklung
- Continuous Delivery soll helfen dieses Problem zu lösen
 - Technische Abstraktion Deployment Pipelines als Hilfsmittel
- Deployment Pipeline in aktuellen CI Servern verfügbar
 - „Infrastructure As Code“ Werkzeuge ebenfalls erhältlich
- Ende der Fahnenstange im Bereich CD Tooling noch nicht erreicht
 - CI Server Hersteller konkurrieren mit neuen Features und Ideen

If you remember one thing



“In order for you to keep up with customer demand, you need to create a deployment pipeline. You need to get everything in version control. You need to automate the entire environment creation process. You need a deployment pipeline where you can create test and production environments, and then deploy code into them, entirely on demand.”

(“The Phoenix Project”)

Links



- Continuous Delivery
 - <http://continuousdelivery.com/>
- Jenkins Build Pipeline Plugin
 - <https://wiki.jenkins-ci.org/x/IgJDAw>
- Go Concepts
 - http://www.thoughtworks.com/products/docs/go/13.3/help/concepts_in_go.html
- Bamboo Deployment Projects
 - <https://confluence.atlassian.com/x/LgQrF>

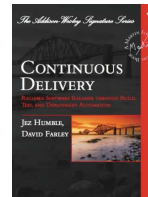
Links

- **The Phoenix Project**
A Novel About IT, DevOps, and Helping Your Business Win

Gene Kim, Kevin Behr, George Spafford
ISBN 0988262592

- **Continuous Delivery**
Reliable Software Releases through Build, Test, and Deployment Automation

Jez Humble, David Farley
ISBN 0321601912



Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de



**Vielen Dank für ihre
Aufmerksamkeit !**

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de