Orientation in Objects

Pipelines zeichnen ist nicht schwer, Pipelines bauen dagegen sehr

OOP 2016

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Version: 1.4

Gliederung

Orientation in Objects

- Einleitung
- Tool Time
- Zusammenfassung

Gliederung

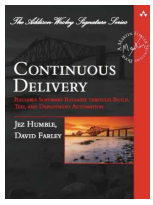
Orientation in Objects

- **Einleitung**
- Tool Time
- Zusammenfassung

The Good Book

Orientation in Objects

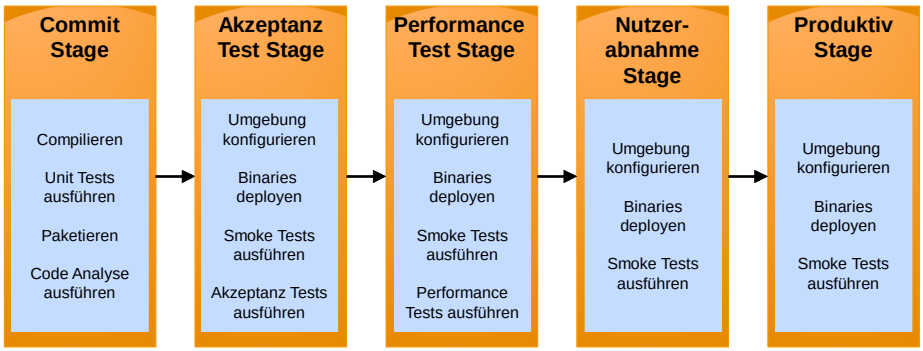
- Frühere Begriffsverwendung von Continous Delivery und Wurzeln
  - „Agile Manifesto" (2001)
  - „Deployment Pipeline" (2004 / 2005)
- Gleichnamiges Buch von Jez Humble & David Farley
  - Eigentliche Begriffsprägung (2010)
- Schwerpunktthemen „Automation" und „Collaboration"



Deployment Pipeline – Ein erster Blick

Orientation in Objects

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



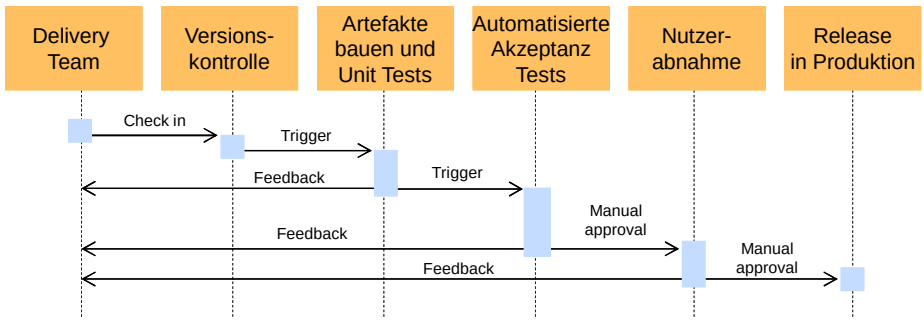
(Nach „Continuous Delivery“/J. Humble, D. Farley)

Deployment Pipeline – Bestandteile

Orientation in Objects

- 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, …

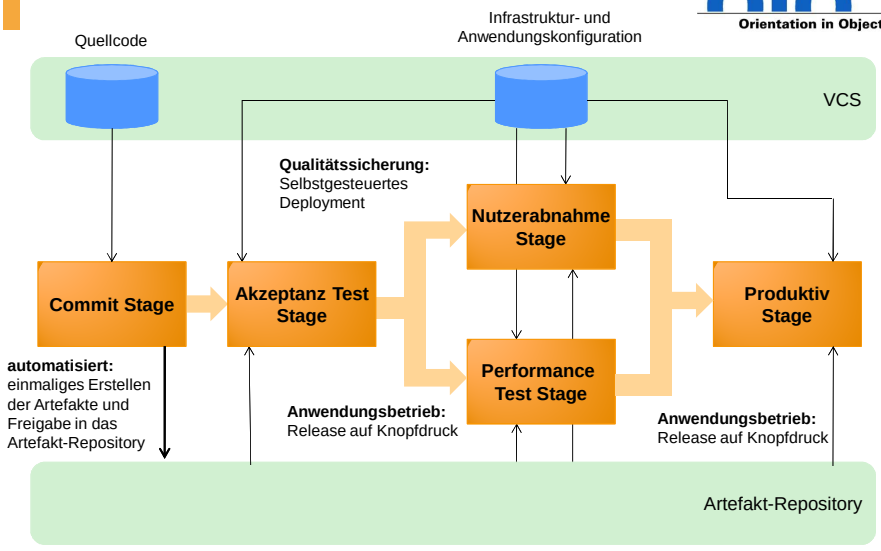
Deployment Pipeline – Sequenzdiagramm



- 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)

Deployment Pipeline – Ein zweiter Blick



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

Gliederung

Orientation in Objects

- Einleitung
- **Tool Time**
- Zusammenfassung

Alle Theorie ist grau

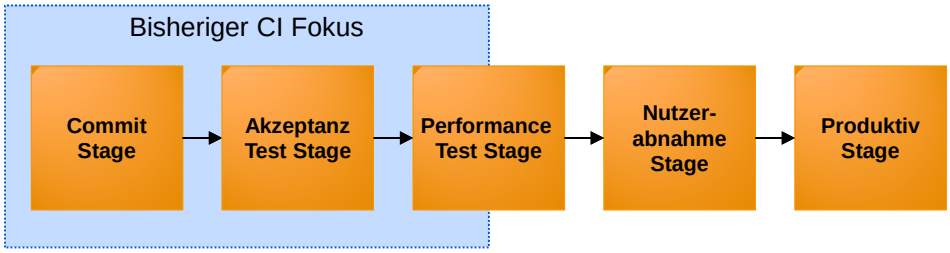
Orientation in Objects

- 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. Mobil vs. Rich Client, Programmiersprache, OS, usw.
  - Somit sind auch Continuous Delivery Umsetzungen verschieden
- Gibt es also gar kein Continuous Delivery Tool?

Continuous Delivery – Tooling (1)

Orientation in Objects

- *„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)

Orientation in Objects

- 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, Docker, …)
- 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

Orientation in Objects

- 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

Gliederung

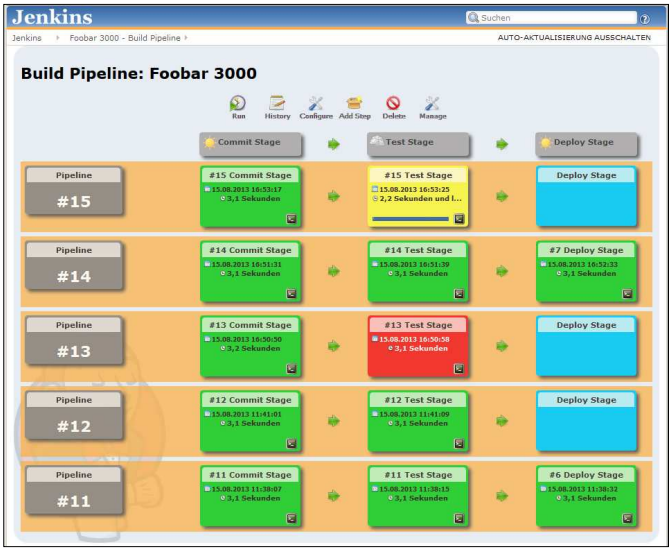
Orientation in Objects

- Einleitung
- **Tool Time**
- Zusammenfassung

- **Jenkins**

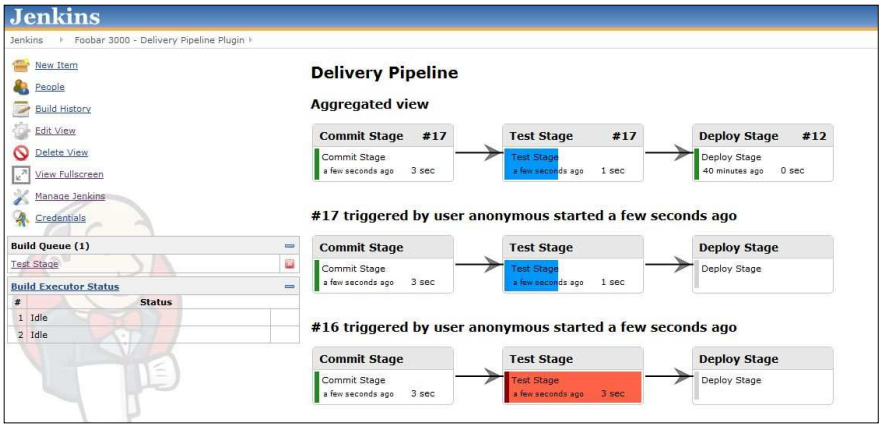
Build Pipeline Plugin in Jenkins

Orientation in Objects



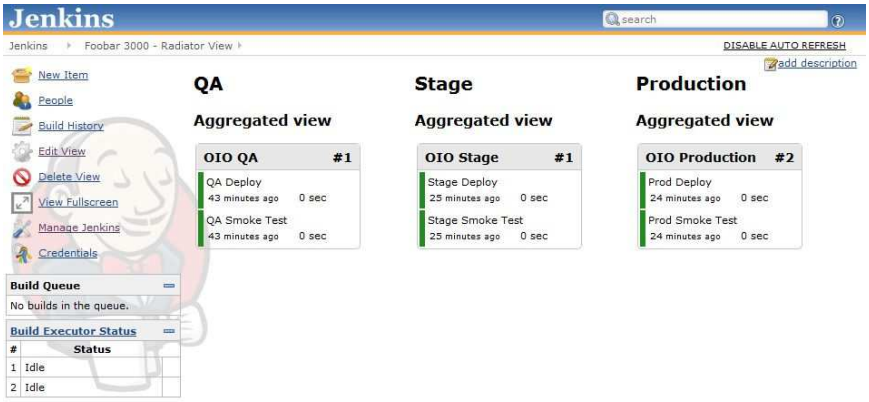
Delivery Pipeline Plugin in Jenkins (1)

Orientation in Objects



Delivery Pipeline Plugin in Jenkins (2)

Orientation in Objects



You Can Look But You Better Not Touch (1)

Orientation in Objects

- 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)

Orientation in Objects

Geht doch, oder?

Orientation in Objects

- Deployment Pipeline Erzeugung in Jenkins benötigt viele Plugins
  - Build Pipeline, Copy Artifact, Parameterized Trigger, Promoted Builds
- Plugins arbeiten nicht ideal zusammen
  - *"[The Build Pipeline and Delivery Pipeline plugin] fail to capture the link to the Deploy to Prod job, which is not an immediate downstream build, but triggered by the Promoted Builds plugin."*
- Build Jobs als höchste Abstraktionsebene, Environments fehlen
  - *"Creating one or multiple views per pipeline is an obvious approach, but it still leaves us with an incredibly large 'All jobs' view in Jenkins – not fun to navigate and manage."*
- Orchestrating Your Delivery Pipelines with Jenkins (Kawaguchi u.a.)

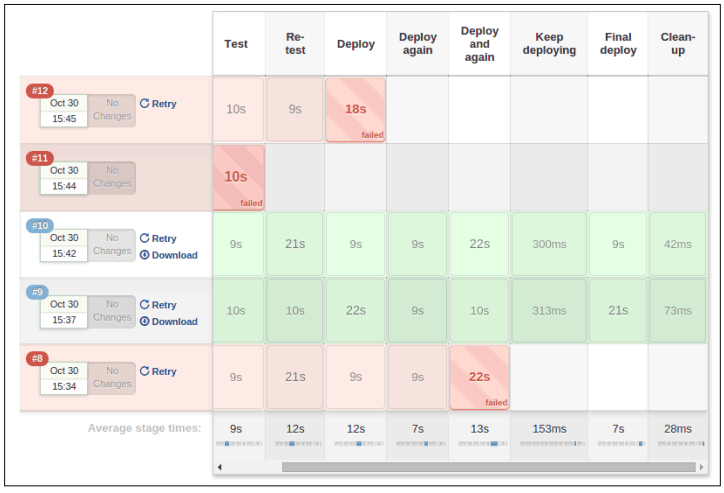
Pipeline Antipattern: Deployment Build

Orientation in Objects

- *“To say CI ‘has a’ CD capability is the wrong way around.”*
- *“When all you have is a CI system, everything looks like a build.”*
  - a.k.a. Maslov’s CI Hammer
- Jenkins ist nicht auf Pipeline Erstellung ausgelegt
  - Künftig Jenkins ~~Workflow Plugin~~ Pipeline Plugin als Lösung?
  - *“Your […] workflow is a single Groovy script using an embedded DSL”*
  - *“Workflows can be divided into sequential [Pipeline] stages”*
  - *https://github.com/jenkinsci/workflow-plugin*
- *“Deployment pipelines should be a first class concept in your CD tools to avoid headaches.”* (Martin Fowler)

Jenkins 2.0 Workflow Visualization?

Orientation in Objects



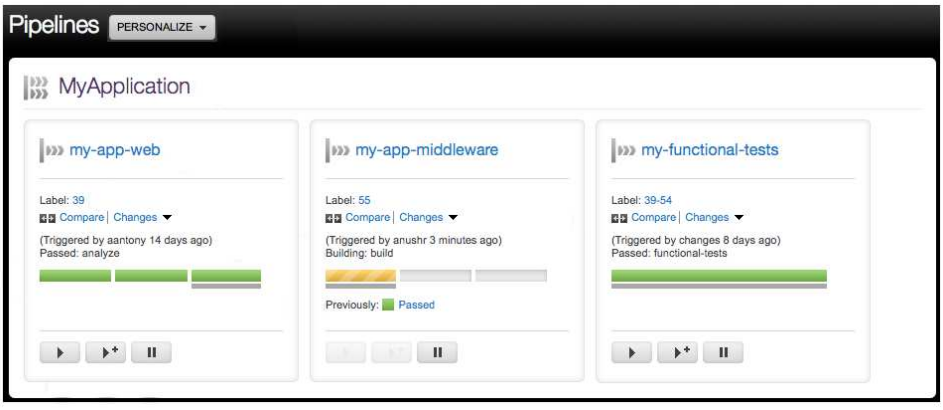
(Quelle: https://issues.jenkins-ci.org/browse/JENKINS-31154)

Gliederung

- Einleitung
- **Tool Time**
- Zusammenfassung

- Jenkins
- **Thoughtworks Go**

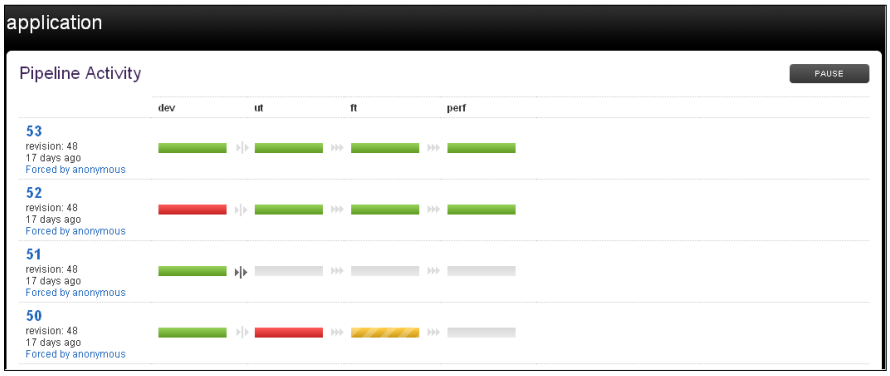
Pipelines Dashboard in Go



(Quelle: http://www.thoughtworks.com/products/docs/go/current/help/)

Pipeline Activity in Go

Orientation in Objects



(Quelle: http://www.thoughtworks.com/products/docs/go/current/help/)

Wenn man jemandem den kleinen Finger reicht, …

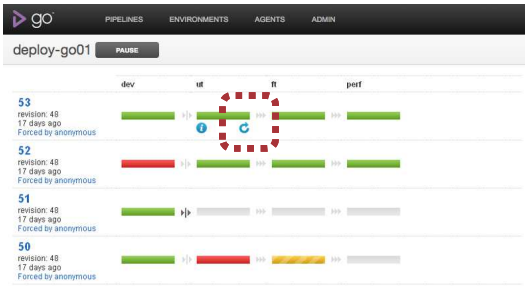
Orientation in Objects

- 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

Orientation in Objects

- 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?



(Quelle: http://www.thoughtworks.com/products/docs/go/current/help/)

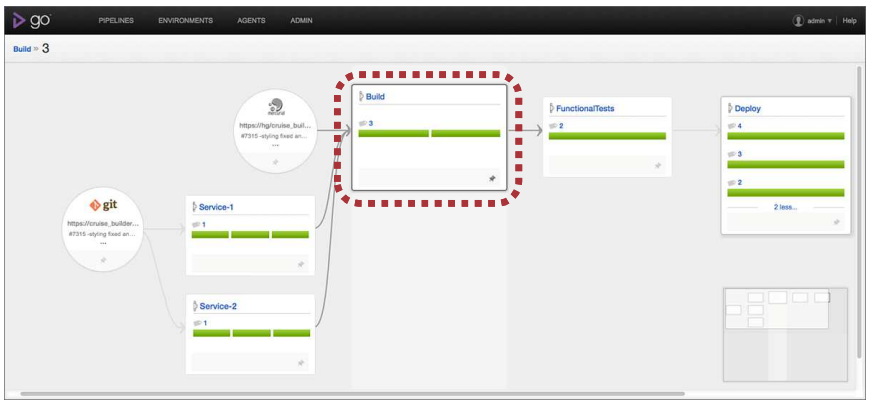
Go Pipelines vs Deployment Pipelines

Orientation in Objects

- *„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

Orientation in Objects



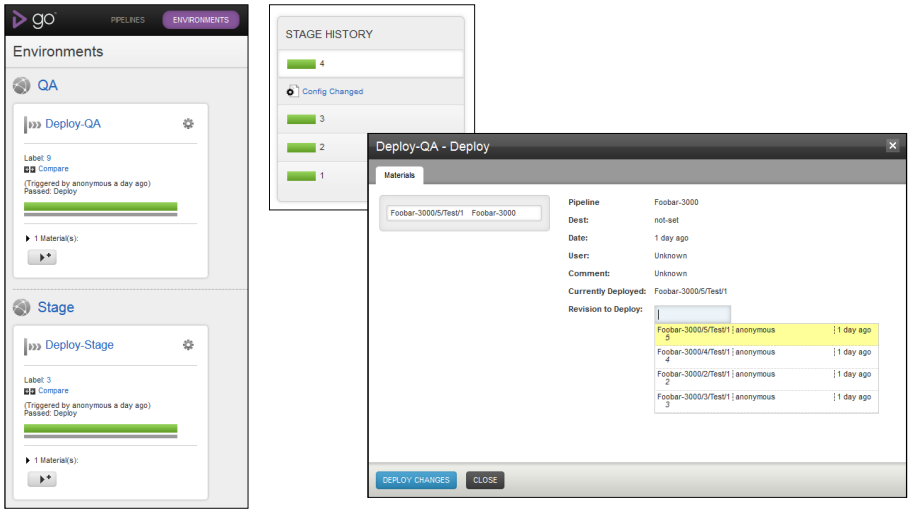
(Quelle: http://www.thoughtworks.com/products/docs/go/current/help/)

Environments in Go (1)

Orientation in Objects

- Go erlaubt explizites Anlegen und Verwalten von Environments
- Jede Go Pipeline gehört zu maximal einer Environment
- Und kapselt so die notwendigen Tätigkeiten für diese Environment
  - Deployment 3-Schicht App in UAT Env. mit 6 Servern und Smoke Test
- Environments beantworten schnell Fragen folgender Art
  - Was läuft gerade in Produktion?
  - Wie relase ich nach Produktion?
  - Wie deploye ich einen bestimmten Versionsstand nach UAT?
  - Wie führe ich ein Rollback durch?

Environments in Go (2)



(Quelle: http://www.thoughtworks.com/products/docs/go/current/help/)

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)*
- *„The tool you use to model and control **your deployment pipeline becomes a system of record for builds**, recording which version they came from in version control, who deployed them to which environments when, and what the results were.“ (J. Humble, D. Farley)*
- Was will uns der Autor damit sagen?

Gliederung

- Einleitung
- **Tool Time**
- Zusammenfassung

- Jenkins
- Thoughtworks Go
- **Atlassian Bamboo**

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)

Orientation in Objects

- 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 Relases 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)

Orientation in Objects

- 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 Revsion 42“
- „*Give deployments the first-class treatment*“
  - Klare Schnittstelle zwischen Dev und Ops

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

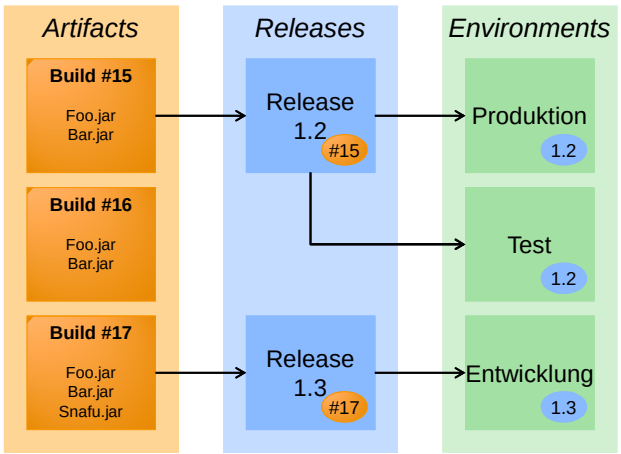
Deployment Projects in Bamboo (1)

Orientation in Objects

- 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)

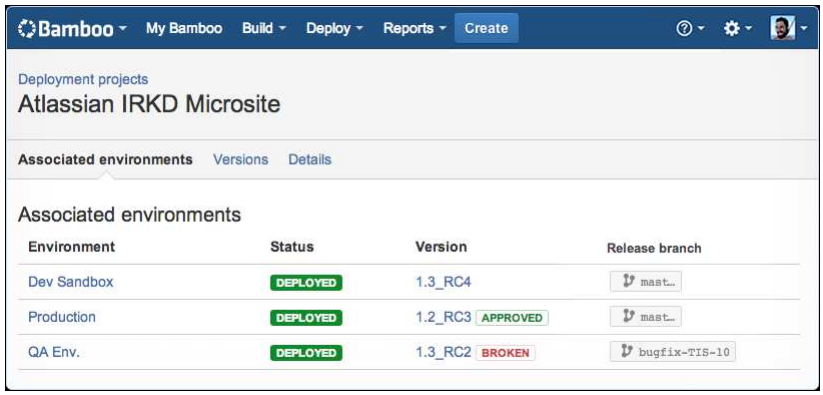
Deployment Project in Bamboo (2)

Orientation in Objects



Environments in Bamboo

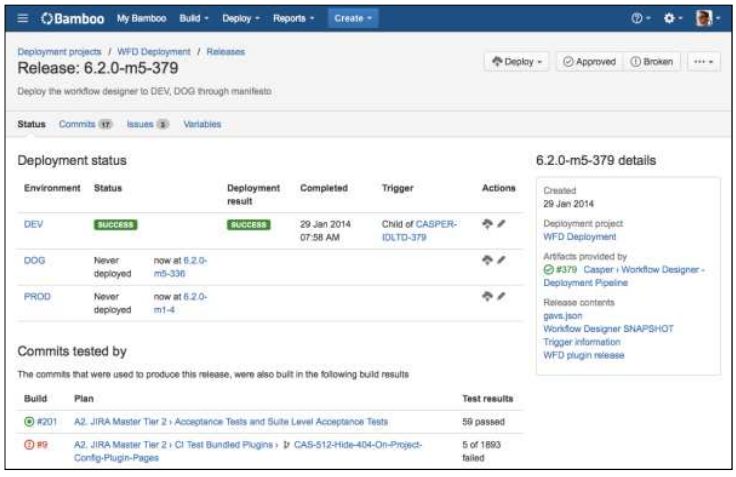
Orientation in Objects



(Quelle: https://confluence.atlassian.com/x/hwI_EQ/)

Releases in Bamboo

Orientation in Objects



(Quelle: https://confluence.atlassian.com/x/hwI_EQ/)

Geht doch, oder? / Wünsch Dir was (Again)

Orientation in Objects

- Deploy Project nicht in Build Plans nutzbar („Wo ist die Pipeline?“)
  - „Represent deployment triggers as stages within a plan“ (BAM-13347)
  - „Smoke testing reports in deployment environments“ (BAM-13276)
  - „Deployment workflows“ (BAM-13356)
- Wunsch: „Multi-Application Continous Delivery“
  - Verwalten und Auswerten von Abhängigkeiten zwischen Anwendungen
- Wunsch: Unterstützung von Release Planung nicht nur Ausführung
  - Wenn es sein muss, sogar manuelle Aufgaben am „Release Day“

Gliederung

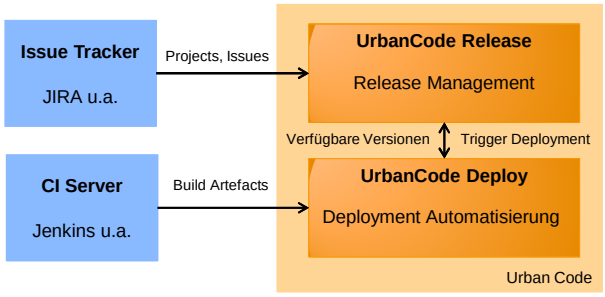
Orientation in Objects

- Einleitung
- **Tool Time**
- Zusammenfassung

- Jenkins
- Thoughtworks Go
- Atlassian Bamboo
- **IBM UrbanCode**

IBM UrbanCode

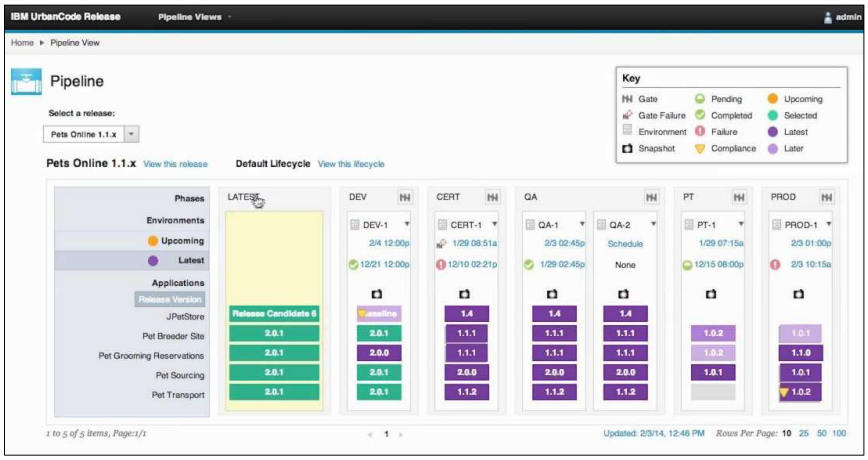
- Übernahme von UrbanCode durch IBM im April 2013
  - Vormals Entwickler von AnthillPro
- Unterteilung in UrbanCode Deploy und UrbanCode Release
  - *“Do you have large monthly or quarterly releases that take hours/days and require dozens or that hundreds of people to get on a call?”*



IBM UrbanCode Release Auszug Kernbegriffe

- Release
  - Kapselt Weg mehrerer Applikationen von Entwicklung zu Produktion
  - Bezieht sich auf einen bestimmten Versionsstand (Snapshot)
  - Enthält Deployment Plan der entsprechenden Applikationen
  - Unterteilt in mehrere Phasen (basierend auf einem Lifecycle Template)
- Deployment Plan
  - Liste manueller und automatisierter Task um Applikationen zu deployen
- Phases
  - Abfolge bestimmter „Stationen“ (zum Beispiel DEV, QA, PT, PROD)
  - Aufhänger für „Pipeline“ und „Train“ Metaphern
  - Jede Phase kann Quality Gates definieren
  - Phasen enthalten Verweise auf Environments zwecks Deployment

IBM UrbanCode Release Pipeline



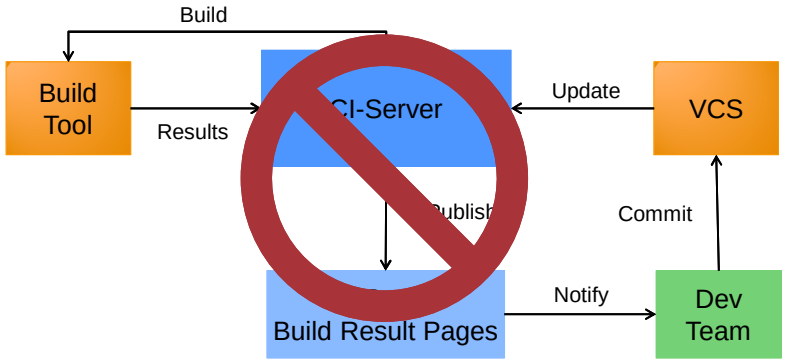
(Quelle: https://www.ibmdw.net/urbancode/products/urbancode-release/)

Gliederung

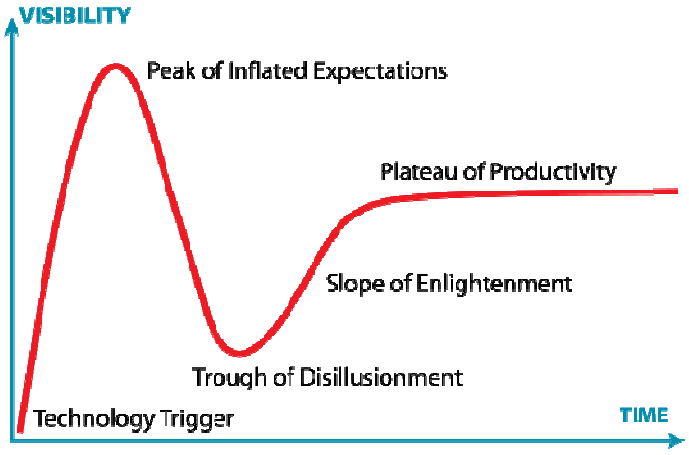
- Einleitung
- Tool Time
- **Zusammenfassung**

Continuous Delivery != Continuous Integration

- Continuous Delivery ist die logische Fortsetzung von Continuous Integration
- CI Tools werden dadurch aber nicht automatisch zu CD Tools



Continous Delivery Tooling wird Fahrt aufnehmen



(Quelle: http://en.wikipedia.org/wiki/Hype_cycle)

If you remember one thing

Orientation in Objects

“There is no one-size-fits-all solution to the complex problem of implementing a deployment pipeline.”

(“Continuous Delivery”, J. Humble, D. Farley)

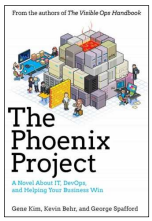
Links

Orientation in Objects

- Orchestrating Your Delivery Pipelines with Jenkins
  - http://www.infoq.com/articles/orch-pipelines-jenkins
- Pipeline Antipattern: Deployment Build
  - http://www.alwaysagileconsulting.com/pipeline-antipattern-deployment-build/
- Open-Sourcing ThoughtWorks Go
  - http://martinfowler.com/articles/go-interview.html
- How do I do CD with Go?: Part 2: Pipelines and Value Streams
  - http://www.thoughtworks.com/insights/blog/how-do-i-do-cd-go-part-2-pipelines-and-value-streams

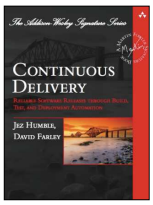
Literaturhinweise

Orientation in Objects



- 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

Fragen ?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

Orientation in Objects

Vielen Dank für ihre Aufmerksamkeit !

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de