



Features Branches mit CI?

Müssen sich Feature Branches und CI widersprechen?

Orientation in Objects GmbH

Weinheimer Str. 68
68309 Mannheim

www.oio.de
info@oio.de

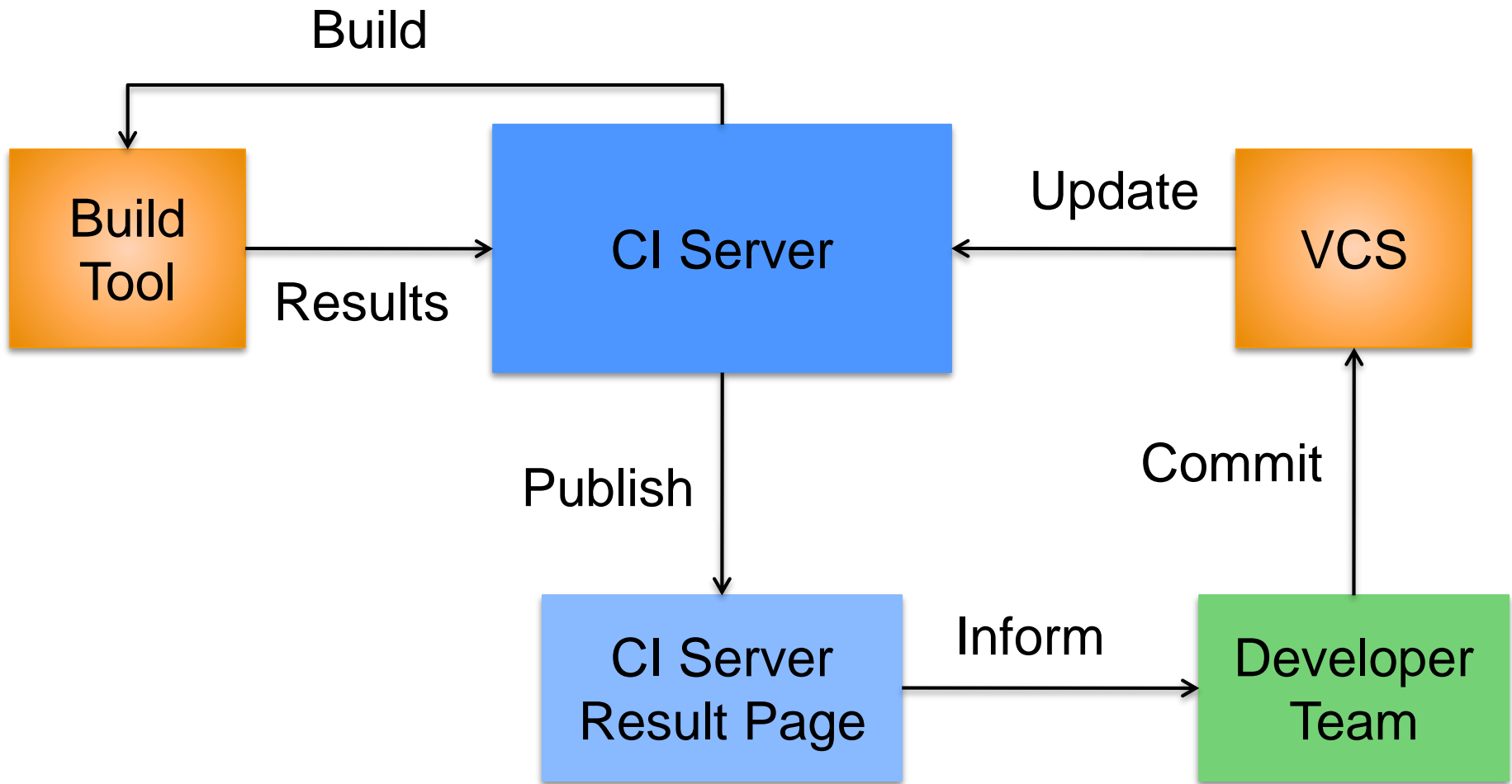


JAVAFORUM 20. Jubiläum
1997-2017 stuttgart

- That's just, like, your opinion, man
- This aggression will not stand, man
- Let's go bowling

- That's just, like, your opinion, man
- This aggression will not stand, man
- Let's go bowling

Been there, done that

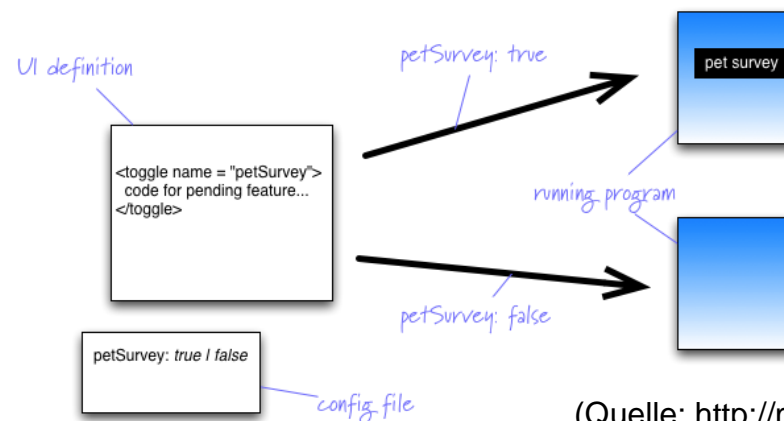


- „Continuous Integration“ Artikel von Martin Fowler im Jahr 2000
 - *„[...] focus [...] on the key practices that make up effective CI“*
- Key Practice Nr. 4 *„Everyone Commits To The Mainline Every Day“*
 - Falls möglich auch häufiger um Konflikte früh zu entdecken
 - Auch Ermunterung, Arbeit in Happen von wenigen Stunden aufzuteilen
- Begriff der „Mainline“ ist auch an andere Stellen sehr zentral
 - *„Pretty much everyone should work off this mainline most of the time.“*
- Branches werden als überbeanspruchtes VCS Feature betrachtet
 - *„Keep your use of branches to a minimum“*

- Technische Empfehlungen muss man im zeitlichen Kontext sehen
 - tl;dr : Das war vor Git und Branches mit SVN waren damals kein Spaß
- Eigener „FeatureBranch” Artikel von Martin Fowler im Jahr 2009
 - *„practice of [DVCS] feature branching and how it fits in with CI”*
- Ursprüngliche Kernaussage bleibt bestehen
 - *„So unless feature branches only last less than a day, **running a feature branch is a different animal to CI.** I've heard people say they are doing CI because they are running builds, perhaps using a CI server, on every branch with every commit. **That's continuous building,** and a Good Thing, but there's no integration, so **it's not CI.**”*
- Neben „Semantic Conflicts” Probleme bei “OpportunisticRefactoring”

Alternative „Feature Toggles“

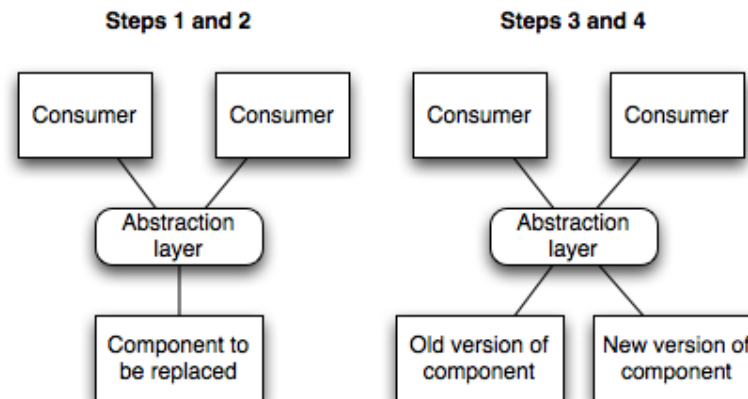
- „Feature Toggles“ als mögliche Alternative zu „Feature Branches“
 - „*mechanism for pending features taking longer than a [...] release cycle*“
- Erlaubt Feature Aktivierung bzw. Deaktivierung zur Laufzeit
 - Standardtechnik mit dedizierten Tools (Togglz, FF4J, ...)
- Hat wie jede Technik auch eine Menge bestimmter Nachteile
 - „*kind of technical debt*“, Code wird trotz Toggle in Produktion released



(Quelle: <http://martinfowler.com/bliki/FeatureToggle.html>)

Alternative „Branch by Abstraction“

- „Branch by Abstraction“ als mögliche Alternative zu Branches
 - „*alternative [...] when making large-scale changes to your system*“
- Betroffener Systemteil erhält eine eigene Abstraktionsschicht
 - Diese erledigt notwendige Delegation zwischen alten und neuem Code
- Auch dieser Ansatz kann Nachteile haben
 - Sockelkosten bei „*big ball of mud*“, Gefahr bei fehlender „*exit strategy*“



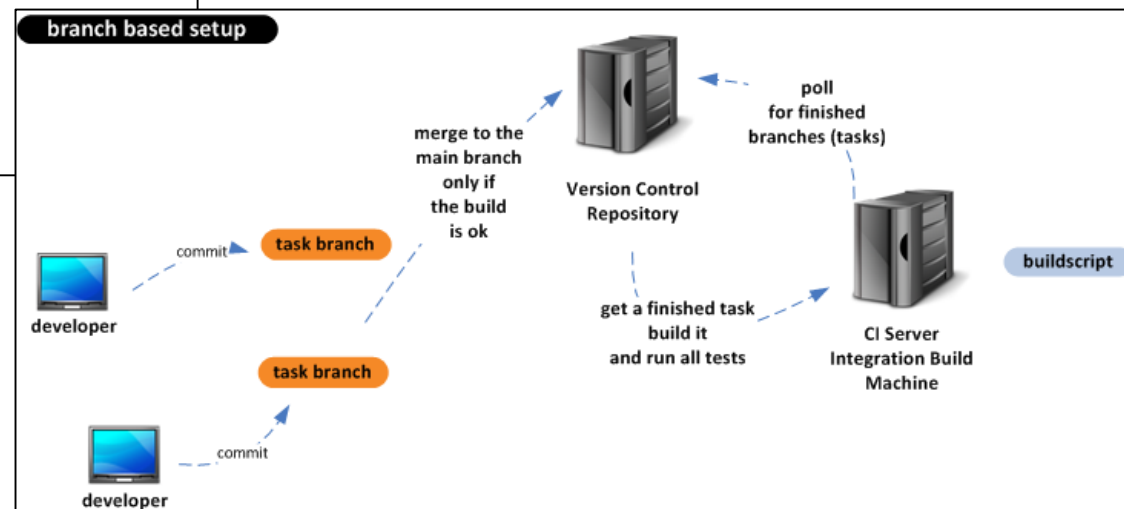
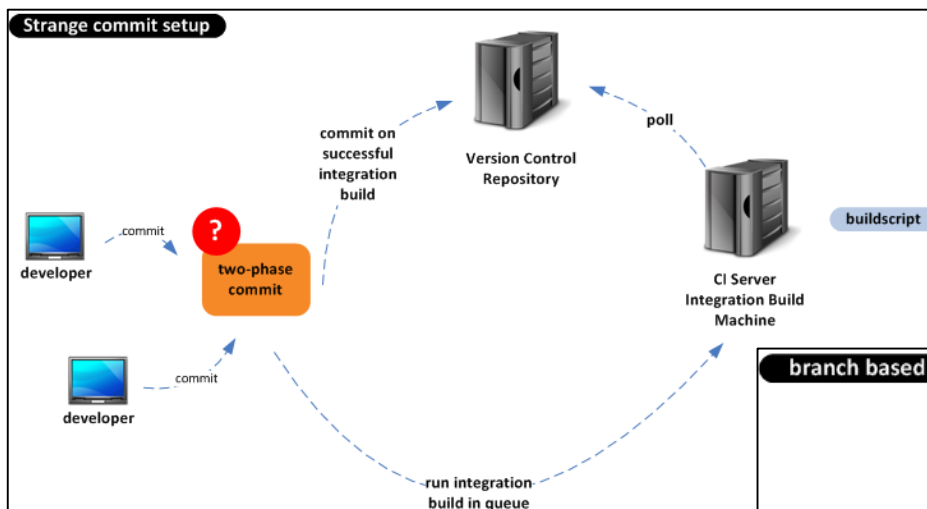
(Quelle: <https://continuousdelivery.com/2011/05/make-large-scale-changes-incrementally-with-branch-by-abstraction/>)

- Damit sind durch CI alle Version Control Probleme gelöst, oder?
- Kann etwas passieren wenn alle täglich in „Mainline“ einchecken?
- Aus Artikel „SCM: Continuous vs. Controlled Integration“ von 2008
 - **Mainline instability:** *„All the changes directly hit the mainline, so making it unstable is relatively easy. Developers always work against the latest sources instead of against well-known releases“*
 - **Unnecessary bug spreading:** *„bug entering the mainline will be spread to all developers in a short time [and] several developers will end up fixing [it]“*
 - **No checkpoints:** *„developers can't enjoy the benefits of using a version control which helps them to keep track of small intermediate changes“*

- „Continuous Integration” Buch von 2007 (M. Fowler Signature Series)
 - *„two key complaints from those who have been practicing CI for a while.”*
 - *„How can I prevent broken builds? How can I get my builds to run faster?”*
- Problem *„[Continuous Integration] is still a rather reactionary practice.”*
 - Im Sinne von, dass man zum Testen nach Mainline einchecken muss
- Epilog *„The Future of CI”* nennt zwei Lösungsansätze
 - **Two-Phase Commit:** *„Before the repository accepts the code, it runs an integration build on a separate machine. Only if [...] successful will it commit the code [...].”*
 - **Personal Builds:** *„[...] capability for a developer to run an integration build using the integration build machine and his local changes along with any other changes committed to the version control repository.”*

Same difference?

- „*Obvious answer*” in Blog „Continuous integration future?” von 2008
 - Commit to branch, rebase onto master, run tests, merge up if tests pass



(Quelle: <http://blog.plasticscm.com/2008/03/continuous-integration-future.html>)

- „*Branching is the answer, isn't it?*”
- Linus Torvalds arbeitet bei der Linux Entwicklung auch mit Branches
 - „*The linux-next tree has something like 8600 commits, and I've merged about 5600 in the last few days.*”
 - „*Only 17 of the merges had any conflicts. And only a couple of those were annoying. The rest is all git doing all the work.*”
- Und die gezeigten Branching Alternativen haben auch Schwächen
 - Reaktivierung von altem Code via Toggles (Knight Capital 2012)
 - Verlust von 460 Millionen Dollar in 45 Minuten
- Also doch alle lieber mit Branches arbeiten, oder?

- Es gibt zahlreiche zu vermeidende „*Branching Anti-Patterns*”
 - Branches sind kein automatischer Erfolgsgarant
- **Merge Paranoia**
 - „*Merging is avoided at all cost, due to a fear of the consequences.*”
- **Merge Mania**
 - „*The team spends an inordinate amount of time merging software assets rather than developing them.*”
- **Big Bang Merge**
 - „*Merging has been deferred to the very end of the development effort and an attempt is made to merge all branches simultaneously.*”

- **Never Ending Merge**
 - *„Merge activity never seems to end; there's always more to merge.”*
- **Wrong Way Merge**
 - *„A software asset is merged with a previous version.”*
- **Branch Mania**
 - *„Branches are created often and for no apparent reason.”*
- **Cascading Branches**
 - *„Branches are never merged back to the main development line.”*
- **Mysterious Branches**
 - *„Nobody can tell you what the branches are for.”*

- **Temporary Branches**

- „*The purpose of a branch keeps changing; it effectively serves as a permanent ‘temporary’ workspace.*”

- **Volatile Branches**

- „*An unstable branch is shared by other branches or merged into another branch.*”

- **Development Freeze**

- „*All development activities are stopped during branching, merging and building new baseline.*”

- **Berlin Wall**

- „*Branches are used to divide the development team members, rather than divide the work they are performing.*”

- That's just, like, your opinion, man
- **This aggression will not stand, man**
- Let's go bowling

- CI und FBs sind per Definition widersprüchlich
 - „By definition [...] CI and FB [...] are mutually exclusive. There **is** in fact a dichotomy. **You can do neither, one or the other, but not both at the same time.**”
 - „By definition, if you have code sitting on a branch, it isn't integrated.”
- Aber Schutz der „Mainline“ ist in beiden Ansätzen ein Kernaspekt
- Lediglich **WIE** geschützt wird ist unterschiedlich
 - Bei CI durch kontinuierliches Integrieren, Bauen und Testen
 - Bei FBs dadurch, dass erst abgeschlossener Code in Master einfließt

- CI alleine verhindert keine „*bad checkins*”
 - „*The **more developers** you have on a project, the **higher the chances** are that one of those developers will check something really bad into source control and **disrupt everyone else's work**. It's simple statistics.*”
- Branches erzwingen Merges bei falschem Gefühl von Sicherheit
 - „*Branches may introduce a **false sense of safety**, as changes made in different branches will eventually be merged together (either manually or automatically), and **bugs may arise if these changes are [...]** incompatible.*”

- Gibt es vielleicht einen pragmatischen Kompromiss?
- Aus „Using Git and feature branches effectively“ von 2011
 - „*You can’t do decentralized versioning unless you also **decentralize** your **testing and integration.***“
 - „***effort** [for] testing and integration **scales exponentially** rather than linearly with amount of change.*“
 - „*decentraliz[ing] testing and integration **solve[s most] problems** before change is **pushed upstream.***“
 - „*By **decentralizing** you have many **people working on smaller sets of changes** that are much easier to deal with: the **collaborative effort on testing decreases** and **when it all comes together you have a much smaller set of problems to deal with.***“
- Tests bereits auf Branch Ebene **VOR** dem Merge zurück nach Master

- Dezentralisiertes Testen löst nicht Problem der „late integration“
- Disziplin im Entwicklerteam notwendig (wie bisher halt auch)
 - *„**Rebase against main frequently** – If your feature branch is way behind, you have done something very wrong“*
 - *„**Push as early as you can** – Feature branches are not about hiding change but about isolating change“*
 - *„**Communicate** – Ask people to push their changes before [pushing] big changes“*
 - *„**Pull change rather than pushing it** – Push only works in small teams. Pull forces people to communicate“*

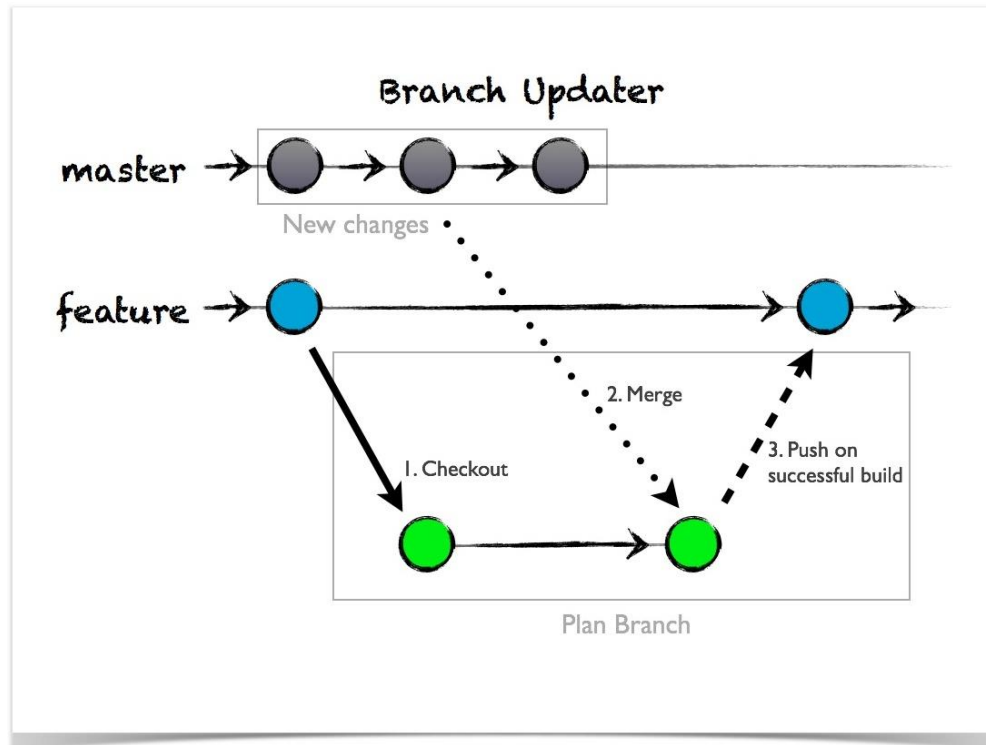
- Lebensdauer der Branches beachten
 - Tasks (Features) innerhalb Branch überschaubar dimensionieren
 - Agile Methoden wie Scrum können hierbei helfen
 - „Feature Toggles“ und „Branch by Abstraction“ nutzen wenn es hilft

- Werkzeuge kennen und Konventionen für Umgang festlegen
 - Git Lernkurve nicht unterschätzen
 - Merging vs. Rebasing, Fast-Forward Merges vs. Merge Commits
 - „*[Don't] pick random [...] points for development to begin with.*“
 - Zusammenarbeit durch „Pull Requests“ und „Code Reviews“ fördern

- That's just, like, your opinion, man
- This aggression will not stand, man
- **Let's go bowling**

- Feature Branch erzeugen
- Continuous Integration für Feature Branch bereit stellen
- Im Feature Branch arbeiten
- Master Änderungen regelmäßig (!) in Feature Branch übernehmen
- Pull Request erstellen jedes Mal wenn „Up Merge“ erfolgen soll
- „Up Merge“ nach Master durchführen und danach deployen

- Und das soll ich jetzt alles von Hand machen?
 - Dezentrales Testen für jeden Branch einrichten
 - Änderungen aus Master nach erfolgreichem Test in Branch übernehmen
 - Pull Requests erstellen und verwalten
- Unterstützung durch Tools ist bereits vorhanden
 - Viele CI Server unterstützen Branch Erkennung
 - Für erkannte Branches dann automatisiert CI Prozesse erzeugbar
 - Automatisches Up und Down Merging üblicherweise auch möglich
 - Git Entwicklungsplattformen unterstützen bei Pull Request Verwaltung
 - Issue Tracker können oft direkt Feature Branches zu erzeugen



(Quelle: <https://confluence.atlassian.com/bamboo/using-plan-branches-289276872.html>)

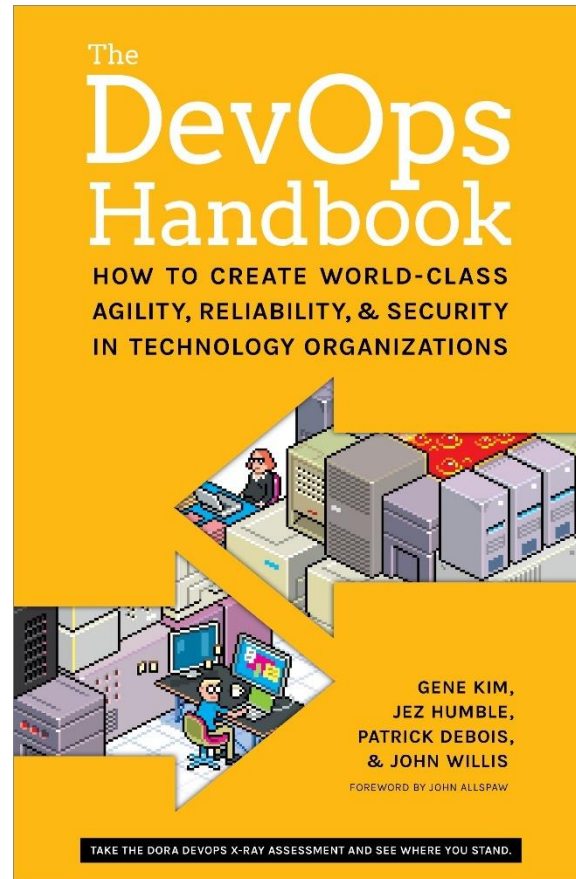
- Entwickler können durch DVCS und Branches häufiger committen
- Dezentrale Tests und Integration je Branch sind guter Master Schutz
- Pull Requests verbessern Kommunikation und Code Qualität weiter
- Feature Branches in Kombination mit CI bieten einen Weg den Master fehlerfrei zu halten, ohne auf die Vorteile automatisierter Testpraktiken zu verzichten
- CI Puristen sind vielleicht anderer Meinung 😊

If you remember one thing

„Practicality beats purity”

(The Zen of Python, Tim Peters)

Literaturhinweise (And Now for Something Completely Different)



(Quelle: <http://itrevolution.com/devops-handbook>)

- Martin Fowler Artikel
 - <http://martinfowler.com/articles/continuousIntegration.html>
 - <http://martinfowler.com/bliki/FeatureBranch.html>
 - <http://martinfowler.com/bliki/OpportunisticRefactoring.html>
 - <http://martinfowler.com/bliki/FeatureToggle.html>
 - <http://martinfowler.com/bliki/BranchByAbstraction.html>
- Feature Toggles are one of the Worst kinds of Technical Debt
 - <https://dzone.com/articles/feature-toggles-are-one-worst>
- Make Large Scale Changes Incrementally with Branch By Abstraction
 - <https://continuousdelivery.com/2011/05/make-large-scale-changes-incrementally-with-branch-by-abstraction/>

- SCM: Continuous vs. Controlled Integration
 - <http://www.drdobbs.com/architecture-and-design/scm-continuous-vs-controlled-integration/205917960>
- Continuous integration future?
 - <http://blog.plasticscm.com/2008/03/continuous-integration-future.html>
- Linus Torvalds' comment on merging
 - <https://plus.google.com/+LinusTorvalds/posts/fDENcrCqHmD>
- Nightmare: A DevOps Cautionary Tale
 - <https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/>

- Software Branching and Parallel Universes
 - <https://blog.codinghorror.com/software-branching-and-parallel-universes/>
- Branch per task workflow explained
 - <http://blog.plasticscm.com/2010/08/branch-per-task-workflow-explained.html>
- Help me, because I think Martin Fowler has a Merge Paranoia
 - <https://arialdomartini.wordpress.com/2011/11/02/help-me-because-i-think-martin-fowler-has-a-merge-paranoia/>
- The Effect of Branching Strategies on Software Quality
 - <https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/shihab-esem-2012.pdf>

- Linus Torvalds auf Linux Kernel Mailing Liste am 28.09.2010
 - <https://lkml.org/lkml/2010/9/28/362>
- Using Git and feature branches effectively
 - <http://www.jillesvangurp.com/2011/07/16/using-git-and-feature-branches-effectively/>
- Imitating pure continuous integration in branching workflows
 - <https://www.atlassian.com/continuous-delivery/continuous-integration-workflows-for-feature-branching>



Orientation in Objects



Fragen ?

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