



Git, Github und Travis

Nina Stodolka und Johannes Struzek, 27.04.2017

Agenda

- Continuous Integration
- Versionsverwaltung
- Git und Github
- Workflows / Branching Modelle
- Build automation mit Travis CI

Continuous Integration

- Ursprung in agiler Softwareentwicklung
- CI = Softwareentwicklungsmethode, die durch hohe Integrationsfrequenz und angeschlossene Automatisierung die schnelle Auslieferung unterstützt
- Ziele:
 - Steigerung der Qualität der Software
 - Integrations-Probleme vermeiden
- Einfache Variante: nightly Build (Scheduled)

CI - Grundsätze

- Gemeinsame Codebasis
- Automatisierte Übersetzung
- Kontinuierliche Test-Entwicklung

CI - Grundsätze

- Häufige Integration
- Integration in den Hauptbranch
- Kurze Testzyklen

CI - Grundsätze

- Gespiegelte Produktionsumgebung
- Einfacher Zugriff
- Automatisiertes Reporting
- Automatisierte Verteilung

CI - Vorteile

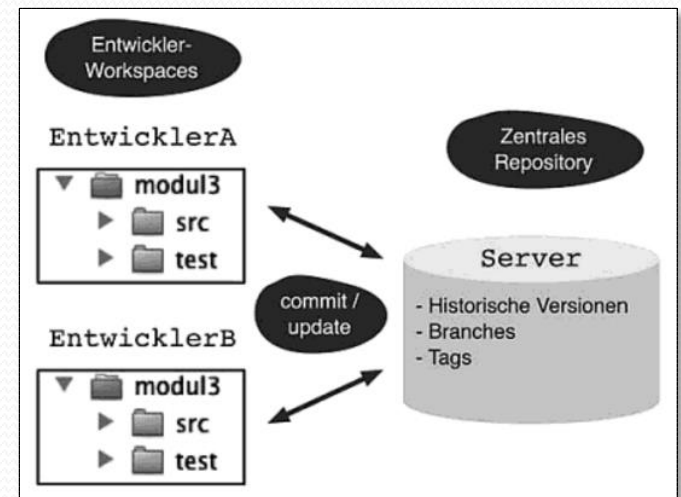
- Steigerung der Software-Qualität durch:
 - Frühzeitiges Erkennen von Integrations-Problemen
 - Frühzeitige Warnungen
 - Sofortige Unit Tests
 - Ständige Verfügbarkeit eines lauffähigen Standes
 - Verantwortlicherer Umgang

Versionsverwaltungssysteme

- SCM werden gebraucht um
 - Datenstände zu speichern und zu protokollieren
 - Ältere Datenstände wiederherstellen zu können
 - Dateien zu verwalten
 - Zugriffsrechte auf Dateien zu regeln
- Repository (eindeutig)
- Unterscheidung zwischen zentralen und dezentralen SCMs

Versionsverwaltungssysteme

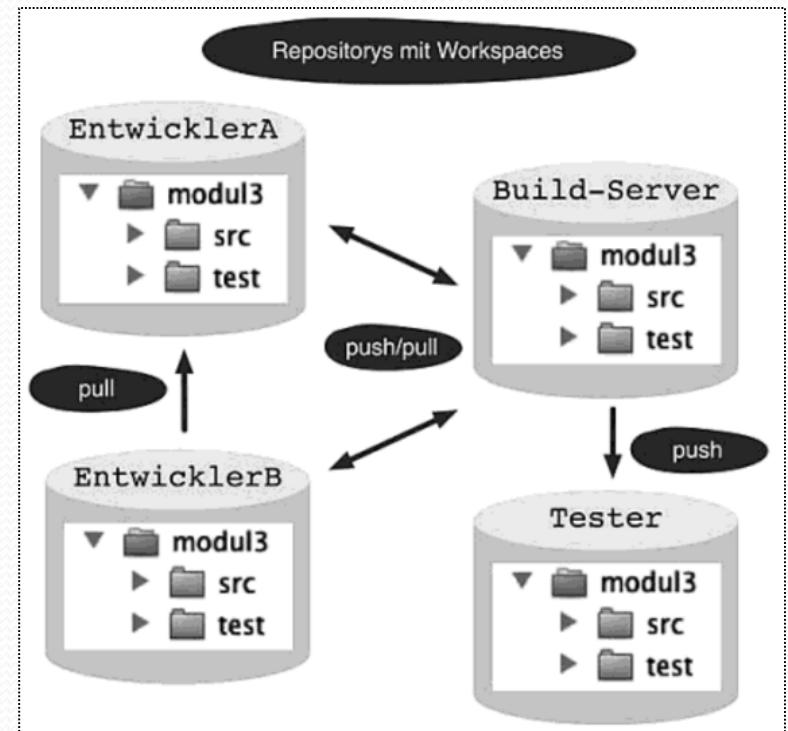
- Zentrale Versionsverwaltungssysteme
- Beispiele: CVS, SVN



Quelle: Preißel, René / Stachmann, Bjørn (4. Auflage 2017): Git

Versionsverwaltungssysteme

- Dezentrale Versionsverwaltungssysteme
- Beispiele: Git, Monotone, BitKeeper



Quelle: Preißel, René / Stachmann, Bjørn (4. Auflage 2017): Git

Vergleich

	Zentral	Dezentral
Repository	Ein zentrales Repository, von dem Arbeitskopien erzeugt werden	Lokal vorliegende eigene Repositorys
History	Historische Versionen der Dateien liegen auf Server; Änderungen sind nur auf Server nachvollziehbar	Änderungen sind auch lokal ohne Verbindung zum Server nachvollziehbar
Netzwerkanbindung	Bei jedem Zugriff notwendig	Nur zur Synchronisation notwendig
Performance & Offline-Fähigkeit	Gering, da für Operationen Netzwerkanbindung erforderlich	Hoch, da fast alle Operationen lokal durchgeführt werden; Merge kann weitgehend automatisch erfolgen
Backup	Zentrale Datenspeicherung auf Server -> separates Backup notwendig	Serverausfall: Auf jedem Rechner lokale Kopien mit kompletter History
Flexibilität des Entwicklungsprozesses		Anlegen spezieller repositorys möglich. Änderungen via Push freigeben

Git



Quelle: <https://git-scm.com/>

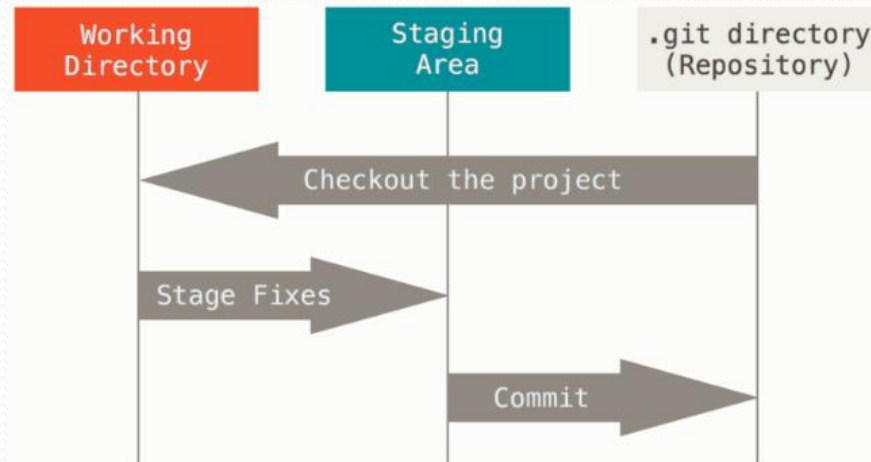
- Freie Software zur verteilten Versionsverwaltung
- Initiiert durch Linus Torvalds

Git - Besonderheiten

- Nicht-Lineare Entwicklung
- Datentransfer zwischen Repositorys (Protokolle, Patches, Review-Systeme, Push & Merge)
- Kryptographische Sicherheit
- Speichersystem und Dateiversionierung

Git - Besonderheiten

- Interoperabilität
- Web-Interface
- Staging Bereich



Quelle: <https://git-scm.com/book/en/v2/Getting-Started-Git-Basics>

Git - Nachteile

- Hohe Komplexität
- Komplizierter Umgang mit Submodulen
- Ressourcenverbrauch bei großen binären Dateien
- Repositorys können nur vollständig verwendet werden
- Mäßige grafische Werkzeuge für die Historienauswertung

Git - Befehle

ein neues Repository anlegen

→ **git init**

ein lokales Repository kopieren

→ **git clone /path/to/repository**

ein externes Repository kopieren

→ **git clone username@host:/path/to/repository**

hinzufügen von Änderungen ins Staging → **git add <filename>**

hinzufügen aller Änderungen ins Stag. → **git add ***

Änderungen committen

→ **git commit -m „Commit message“**

Message sollte kurz und prägnant sein

senden der Änderungen zum ext. Rep. → **git push origin master**

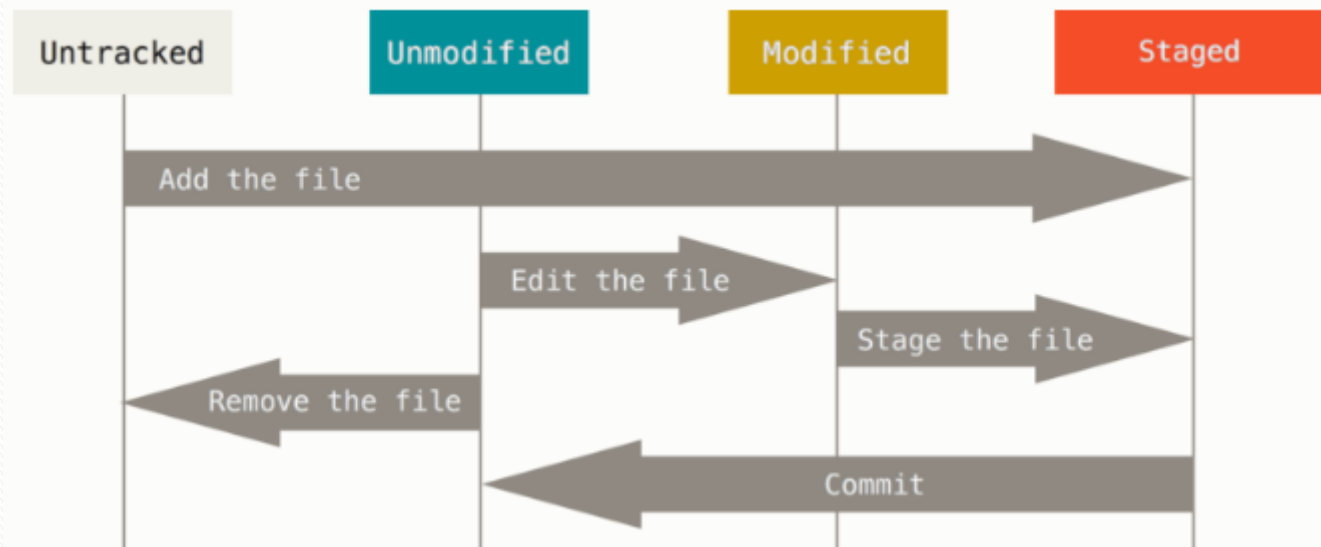
lokales Repository updaten

→ **git pull**

Git - Befehle

alle Änderungen seit letztem
commit anzeigen

→ **git status**



Quelle: <https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository>

Git - Befehle

neuen Branch anlegen & wechseln
zum Master zurückwechseln
Branch wieder löschen
Branch ins Repository hochladen
(verfügbar für andere machen)

→ **git checkout -b <branch>**
→ **git checkout master**
→ **git branch -d <branch>**

zusammenführen zweier Branches
Unterschiede anzeigen lassen

→ **git push origin <branch>**

→ **git merge <branch>**
→ **git diff <source_branch> <target_branch>**

taggen
Commit Ids erhalten

→ **git tag <tag> <commitID>**
→ **git log**

lokale Änderungen auf letzten Stand
des Repositorys zurücksetzen

→ **git checkout -- <filename>**



Kleine Demo Git


GitHub







GitHub



Quelle: <https://github.com/logos>



- Webbasierter Online-Dienst, der Software-Entwicklungsprojekte auf seinen Servern bereitstellt
- Unterstützung im Entwicklungsworkflow
- Code Hosting
- Seamless code review


 **Changes requested** [Hide all reviewers](#)
1 review requesting changes

  **github** — Build #5630178 [Details](#)

  **mdu** requested changes [See review](#)

  **yourfrienderin** was requested for review

  **stephbwillis** was requested for review

+8 -5      app/assets/stylesheets/head.scss	
1	1
2 - min-height: 40px;	2 + display: sticky;
3 - padding: 10px;	3 + top: 0;
4 - font-size: 16px;	4 + z-index: 29;
5 - left: -4px;	5 + width: 24px;
6 - width: 25px;	6 + min-height: 48px;
7	7 + padding: 15px;
8	8 + margin-top: 15px;
9	9 + font-size: 14px;
10	10

Quelle: <https://github.com/>

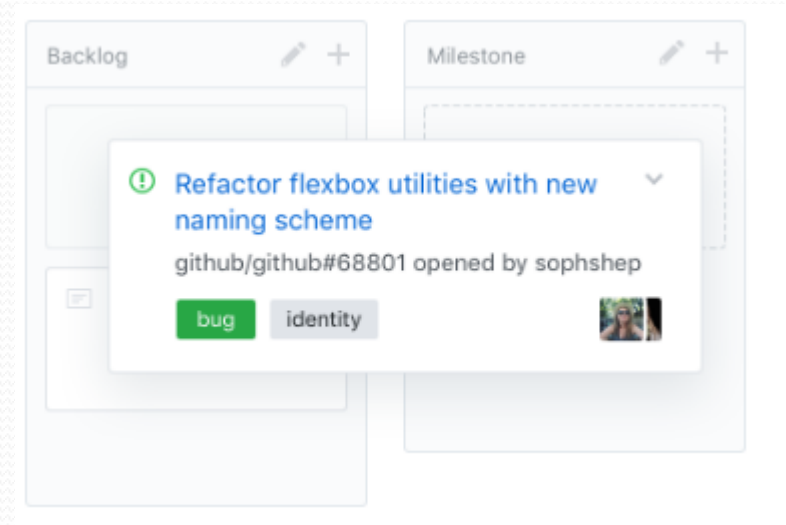
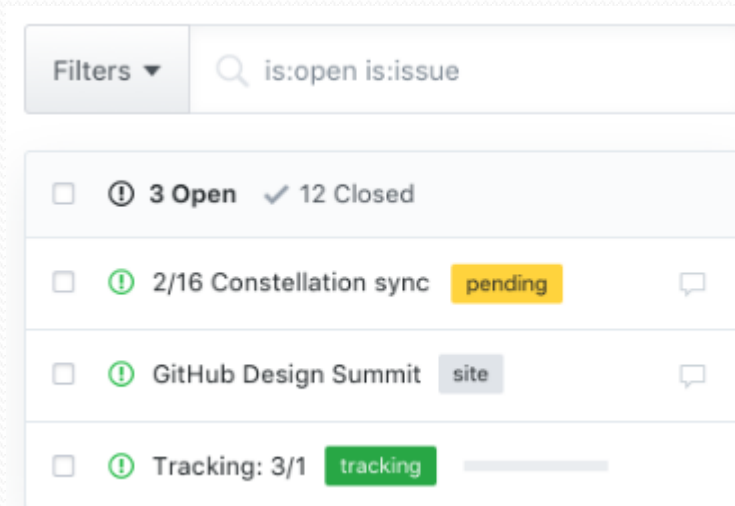
GitHub



GitHub

Quelle: <https://github.com/logos>

- Projektmanagement Features



Quelle: <https://github.com/>

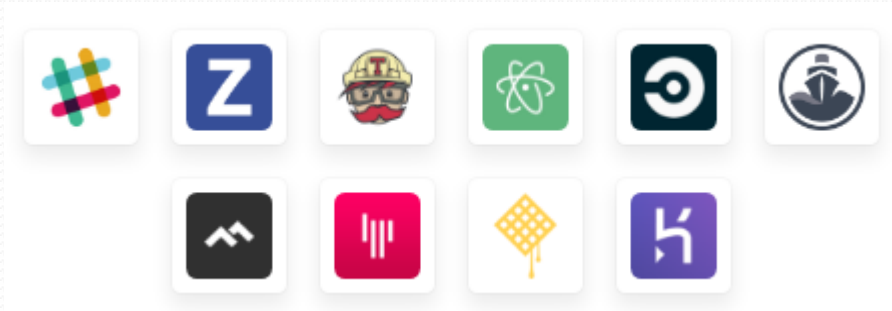
GitHub



GitHub

Quelle: <https://github.com/logos>

- Integrationen: z.B.:



Quelle: <https://github.com/>

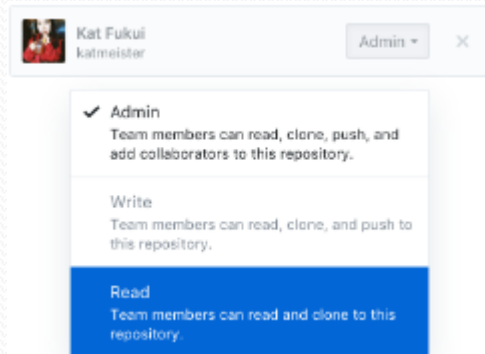
GitHub



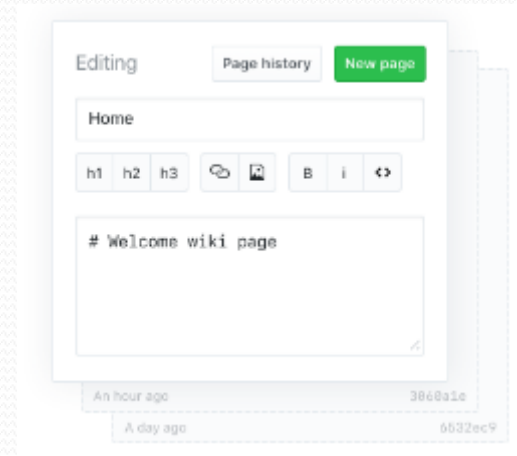
GitHub

Quelle: <https://github.com/logos>

- Großteil der Features von Git ohne Kommandozeile nutzbar
- Dokumentation und Wiki
- Community Management



Quelle: <https://github.com/>



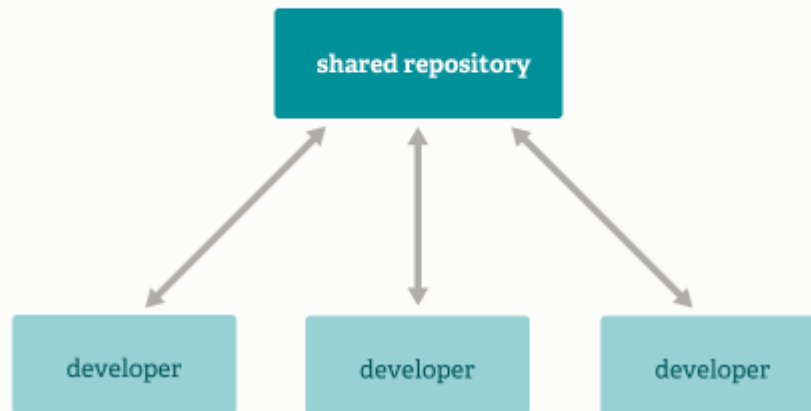
Workflows / Branching Modelle

- Gründe für Branches
 - Mehrere unabhängig arbeitende Entwickler
 - Bugfixes für ältere Versionen
 - Parallele Entwicklung mehrerer Features
 - Stabilisierungsphase für Release und Arbeiten an nächster Version

Strategien - Branching Modelle

Subversion-Style Workflow

A centralized workflow is very common, especially from people transitioning from a centralized system. Git will not allow you to push if someone has pushed since the last time you fetched, so a centralized model where all developers push to the same server works just fine.

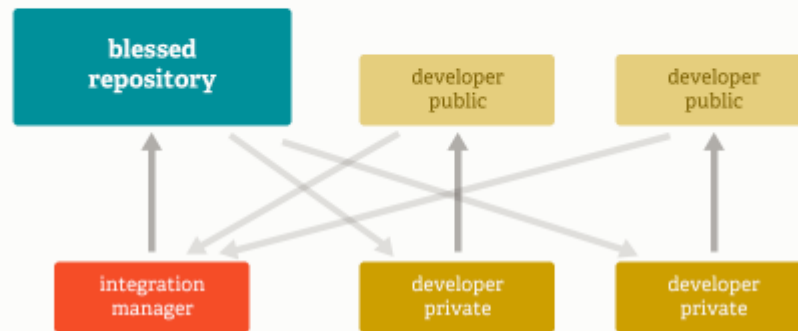


Quelle: <https://git-scm.com/about/distributed>

Strategien - Branching Modelle

Integration Manager Workflow

Another common Git workflow involves an integration manager — a single person who commits to the 'blessed' repository. A number of developers then clone from that repository, push to their own independent repositories, and ask the integrator to pull in their changes. This is the type of development model often seen with open source or GitHub repositories.

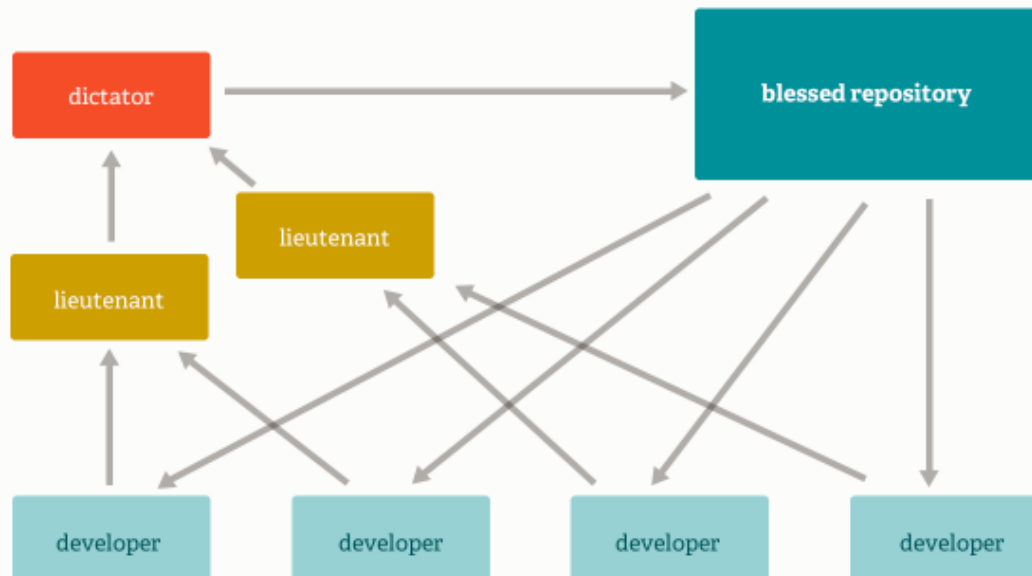


Quelle: <https://git-scm.com/about/distributed>

Strategien - Branching Modelle

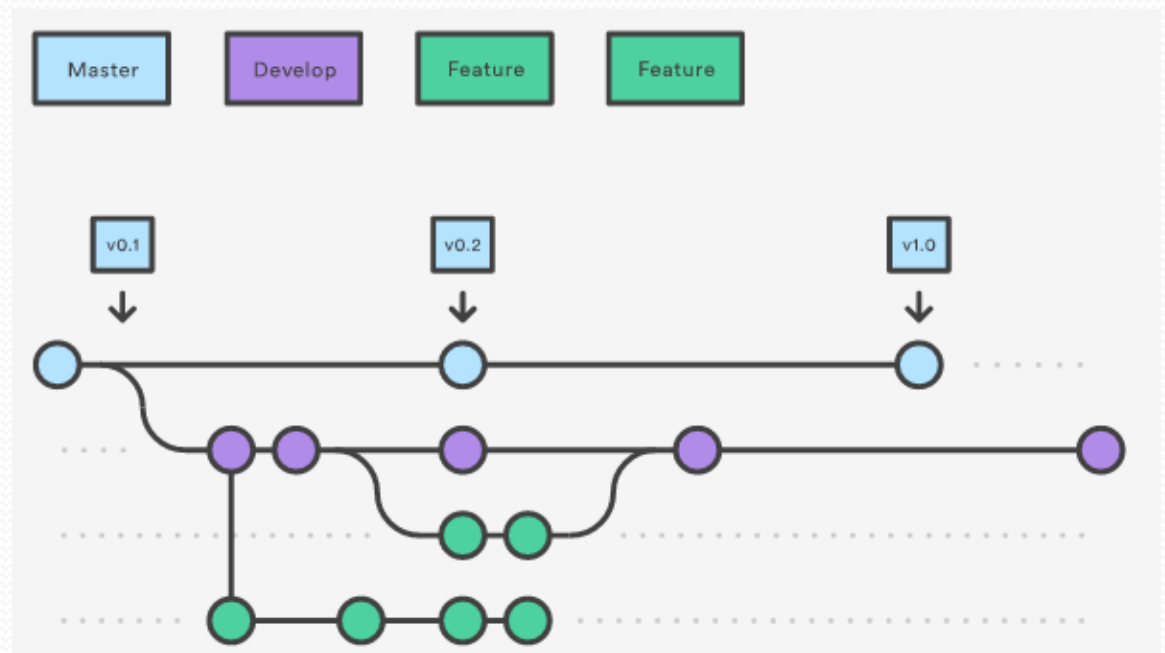
Dictator and Lieutenants Workflow

For more massive projects, a development workflow like that of the Linux kernel is often effective. In this model, some people ('lieutenants') are in charge of a specific subsystem of the project and they merge in all changes related to that subsystem. Another integrator (the 'dictator') can pull changes from only his/her lieutenants and then push to the 'blessed' repository that everyone then clones from again.



Workflows / Branching Modelle

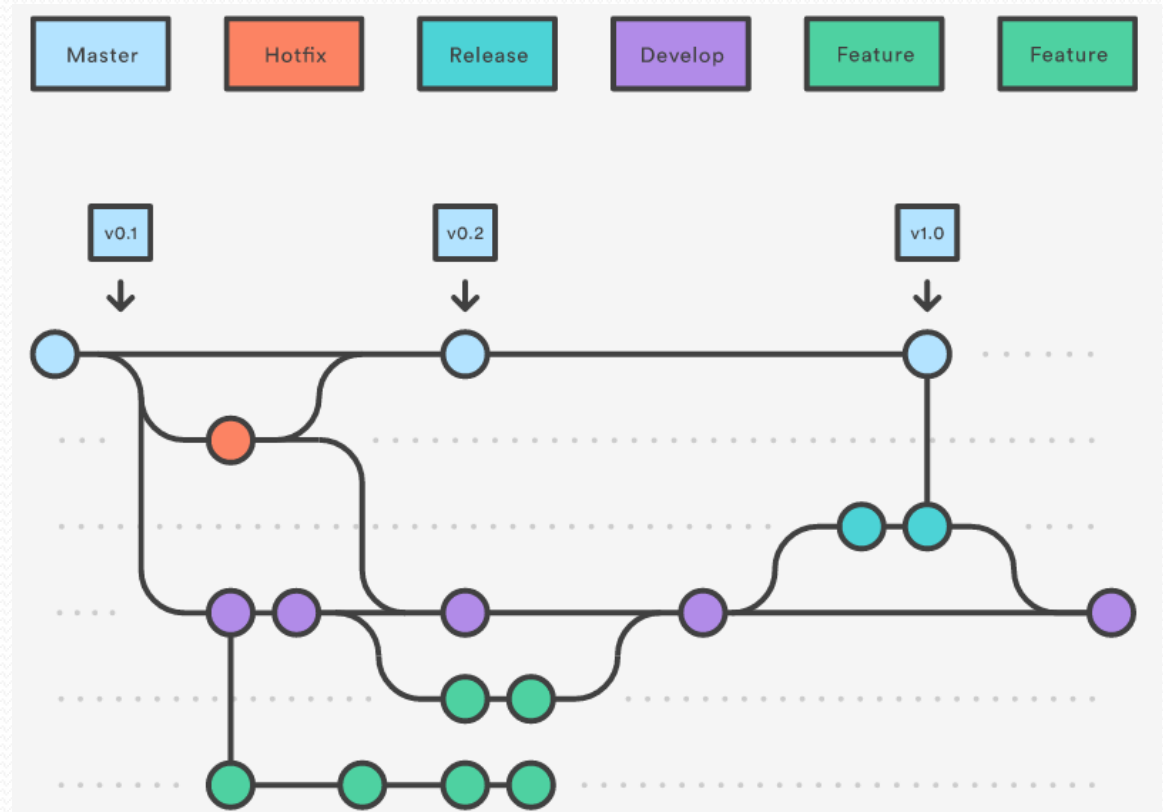
- Mit Feature-Branches entwickeln



Quelle:
<https://de.atlassian.com/git/tutorials/comparing-workflows#feature-branch-workflow>

Workflows / Branching Modelle

- Periodisch Releases durchführen



Quelle:
<https://de.atlassian.com/git/tutorials/comparing-workflows#feature-branch-workflow>

Workflows / Branching Modelle

- Einfacher Workflow für Git:



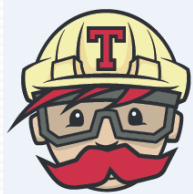
Quelle: Eigene
Darstellung

- Hinweise zum Committen
 - Logisch zusammenhängende Änderungen gemeinsam einchecken
 - Projekt muss nach jedem Commit kompilierbar sein
 - Projekt sollte nach jedem Commit lauffähig sein

Build Automation

- Automatisierte Erstellung von Builds
- Lokal oder Server basiert
- Typen:
 - On-demand automation
 - Scheduled automation
 - Triggered automation

Build automation mit Travis CI



Travis CI

Quelle: <https://travis-ci.com/logo>

- Freie Open-Source-Software für CI
- Zum Testen und Erstellen von Projekten, die auf GitHub, Bitbucket und Ähnlichem veröffentlicht wurden
- Builds erstellen und testen
- Lokal (Docker) oder cloud-basiert (Standard)

Build automation mit Travis CI

- Features:
 - Schnelles Setup
 - weitere Anbindungen
 - Testüberwachung im laufenden Test möglich
 - „clean“ VM für jeden Build
 - Vorinstallierte Testumgebung
 - Parallele Tests möglich
 - Linux, Mac und iOS Unterstützung
- Beta-Features

Build automation mit Travis CI

- Funktionsweise
 - „Triggered automation“ oder bei Cron Job „Scheduled automation“
 - Konfiguration über ein .travis.yml File, das in das Root Verzeichnis des Repositorys gelegt wird



Kleine Demo GitHub & Travis

Erfahrungen aus der Praxis - CI

- Vernachlässigung Testprozesse
- Spiegelung Produktionsumgebung
- Automatisierte Verteilung / einfacher Zugriff
- Herausforderung Verantwortungsübernahme für Build-Qualität
- Automatisierung von Builds: Freigabe von Builds in einzelnen Schritten (für Testsystem, Produktionsumgebung)
- Abstimmung & Kontrolle Termine
 - Wer darf wann wo etwas einchecken (SVN)
 - Herausforderung viele Umgebungen & ungleiche Zyklen -> Ungenauigkeiten Check-In



Danke für eure Aufmerksamkeit!
Fragen?

Quellenverzeichnis

- 1 & 1 Digital Guide: Git vs. SVN – Von verteilter und zentralisierter Versionsverwaltung, URL: <https://hosting.1und1.de/digitalguide/websites/web-entwicklung/git-vs-svn-versionsverwaltung-im-vergleich/> vom 15.04.2017
- Atlassian: Comparing Workflows, URL: <https://de.atlassian.com/git/tutorials/comparing-workflows> vom 15.04.2017
- Chacon, Scott / Straub, Ben (2. Auflage 2014): Pro Git, Apress, URL: <https://git-scm.com/book/en/v2> vom 15.04.2017
- Dudler, Roger: git – Der einfach Einstieg, URL: <https://rogerdudler.github.io/git-guide/index.de.html> vom 15.04.2017
- Fowler, Martin: Continuous Integration, URL: <https://www.martinfowler.com/articles/continuousIntegration.html> vom 15.04.2017
- Git: Homepage, URL: <https://git-scm.com> vom 15.04.2017
- GitHub Inc.: Homepage, URL: <https://github.com> vom 15.04.2017
- Preißel, René / Stachmann, Bjørn (4. Auflage 2017): Git – Dezentrale Versionsverwaltung im Team – Grundlagen und Workflows, Heidelberg
- Software & Support Media GmbH: CI-Server im Vergleich: Jenkins vs. CruiseControl vs. Travis <https://jaxenter.de/ci-server-im-vergleich-jenkins-vs-cruisecontrol-vs-travis-38081> vom 17.04.2017
- Stückler, Moritz: Was ist eigentlich dieses GitHub? , URL: <http://t3n.de/news/eigentlich-github-472886/> vom 15.04.2017
- Travis CI GmbH: Homepage, URL: <https://travis-ci.com/> vom 15.04.2017
- Von dem Berge, Jana (2009): Auswirkungen der Benutzung von zentralen und dezentralen Versionsverwaltungssystemen In Open Source Projekten, URL: <http://www.inf.fu-berlin.de/inst/ag-se/theses/Berge09-versionsverwaltung-OSS.pdf> vom 15.04.2017
- Wikimedia Foundation Inc.: Build automation, URL: https://en.wikipedia.org/wiki/Build_automation vom 17.04.2017
- Wikimedia Foundation Inc.: Continuous integration, URL: https://en.wikipedia.org/wiki/Continuous_integration vom 17.04.2017
- Wikimedia Foundation Inc.: Git, URL: <https://de.wikipedia.org/wiki/Git> vom 11.04.2017
- Wikimedia Foundation Inc.: GitHub, URL: <https://de.wikipedia.org/wiki/GitHub> vom 15.04.2017
- Wikimedia Foundation Inc.: GNU General Public License, URL: https://de.wikipedia.org/wiki/GNU_General_Public_License vom 15.04.2017
- Wikimedia Foundation Inc.: Kontinuierliche Integration, URL: https://de.wikipedia.org/wiki/Kontinuierliche_Integration vom 15.04.2017
- Wikimedia Foundation Inc.: Travis CI, URL: https://de.wikipedia.org/wiki/Travis_CI vom 17.04.2017
- Wikimedia Foundation Inc.: Versionsverwaltung, URL: <https://de.wikipedia.org/wiki/Versionsverwaltung> vom 17.04.2017

I like to see the world burn.



SHA1-Prüfsummen

- Prüfsummen Funktion
- Aus beliebigen Datenblöcken Prüfwerte bildbar (bei Git auf Basis aller Daten – Inhalten der Dateien, Autor und Zeitpunkt)
- Durch den Vergleich von Prüfwerten bei Sender und Empfänger wird sichergestellt, dass Nachrichten unverändert ankommen
- ABER: es gibt mehrere Ausgangszahlen die die gleiche Prüfzahl ergeben -> ist an sich geknackt

GNU GPLv2 Lizenz

- GNU GPL:
 - Erlaubt Software auszuführen, zu studieren, zu ändern und zu verbreiten
 - -> Freie Software
 - Falls Software einem Copyleft unterliegt, müssen diese Rechte bei Weitergabe beibehalten werden
 - „Liberty or Death“-Klausel: Diese besagt, wenn es nicht möglich ist, einige Bedingungen der GNU GPL einzuhalten – beispielsweise wegen eines Gerichtsurteils – es untersagt ist, diese Lizenz nur bestmöglich zu erfüllen. In diesem Fall ist es also überhaupt nicht mehr möglich, die Software zu verbreiten.
 - Paragraph 8: Gültigkeit für einzelne Länder abschließbar

Versionsverwaltungssysteme

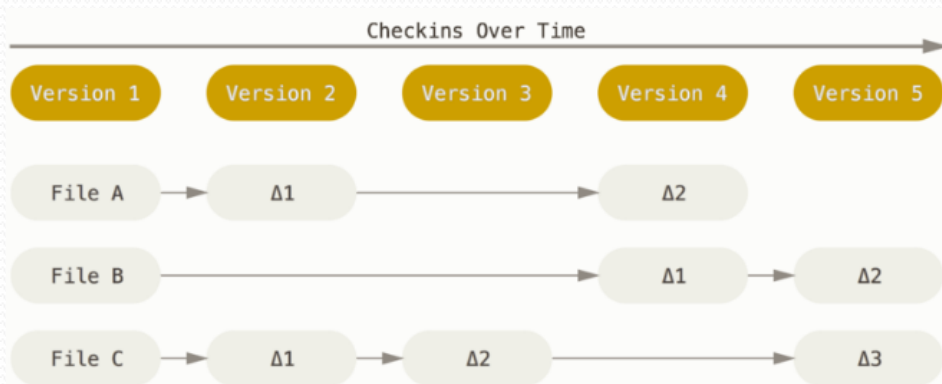
- Jede Revision enthält Metadaten
 - Name des Autors
 - Zeitstempel
 - Commit Message
- Zugriff auf das Repository entweder lesend (vor allem bei OpenSource-Projekten) oder so stark wie möglich eingeschränkt
- Schreibender Zugriff kann direkt gewährt werden (Commit-based Collaboration) oder indirekt (Patch-based Collaboration)
- Unterscheidung zwischen zentralen und dezentralen SCMs

Versionsverwaltungssysteme

- Zusammenarbeit (Beispiel A möchte bei B mitentwickeln)
 - B macht Repository öffentlich (Hosting)
 - Repository von B klonen -> A hat eigenes unabhängiges Repository
 - Um auf dem gleichen Stand zu bleiben Remote Update:
Regelmäßiges Pull von dessen Repository
 - Änderungen von A an B übertragen:
 - Veröffentlichung eigenes Repository und bittet um Pull ODER
 - Remote Commit: Rechte von B bekommen, direkt in dessen Repository zu pushen ODER
 - Versand von Patches

Git

- Art der Datenhaltung (SVN,...)



- Art der Datenhaltung (Git)

