Zur Person

Jan Speckamp

- Student MSc Informatik @WWU
- Vorher BSc Geoinformatik @WWU
- speckij auf Github
- Werksstudent bei 52°North GmbH
  - Seit April 2017
  - Fokus: Backendprogrammierung mit Java/Maven/Spring

- Email: j_spec05@wwu.de
- Mattermost: @j_spec05

Einführung in Git

Jan Speckamp
GeoSoPra Tutor

Inhalt

- Problem/Motivation
- Kernkonzepte von git
- Wie benutze ich git?
- Alternativen zu git
- Interaktive Demo

Problem:

Kollaborative Softwareentwicklung ist schwierig

Situation:

- Viele Entwickler
- Wenige (dafür große) Projekte (im Vergleich zur Anzahl von Entwicklern)
- Code muss archiviert/versioniert/geteilt werden
- Zeitgleiche Entwicklung an verschiedenen Stellen
- Zeitgleiche Entwicklung an gleichen Stellen

Lösung: Versionierungssystem

git

- Konzipiert von Linus Torvalds als Verwaltung für den Linux Kernel Sourcecode
- Offen unter GPLv2 lizensiert (https://git.kernel.org/pub/scm/git/git.git/)
- Schnell & Effizient (im Vergleich mit anderen Systemen)
- Besondere Features:
  - Kein zentraler Server
  - Branching (nicht-lineare Entwicklung)
  - Projektgeschichte final und nicht modifizierbar
  - Protokollunabhängig (http, git, file, etc.) & Plattformunabhängig

→ Sehr weit verbreitet (70%+ Marktanteil bei OS-Repositories, steigend)
→ Sollte auf jedem Lebenslauf stehen (können)

- Öffentlicher Host für git Repositories
- Eigentümer Microsoft (seit 2018, Kaufpreis 7,5 Milliarden Dollar)
- Kostenfreie & Kostenpflichtige Abonnements
- Zahlreiche Zusatzfunktionen (ggü. git)
  - Continuous Integration
  - Wiki
  - Projekt Boards (Kanban)
  - PR-Bots

- Öffentlicher Host & Hosting Software
- MIT Lizenz (Community Edition)
- Zahlreiche Zusatzfunktionen (ggü. git)
  - Continuous Integration
  - Wiki
  - Projekt Boards (Kanban)
  - PR-Bots
  - Metrics
  - etc.

Weitere: Gitea, BitBucket, Gerrit, Gogs, ...

Kernkonzepte von git

- **Grundbausteine**
  - Repository
  - Commits
  - Branches

- **Aktivitäten**
  - Pull
  - Push
  - Merge

Hinweis:

Git lernt man nicht von Folien!

Repository

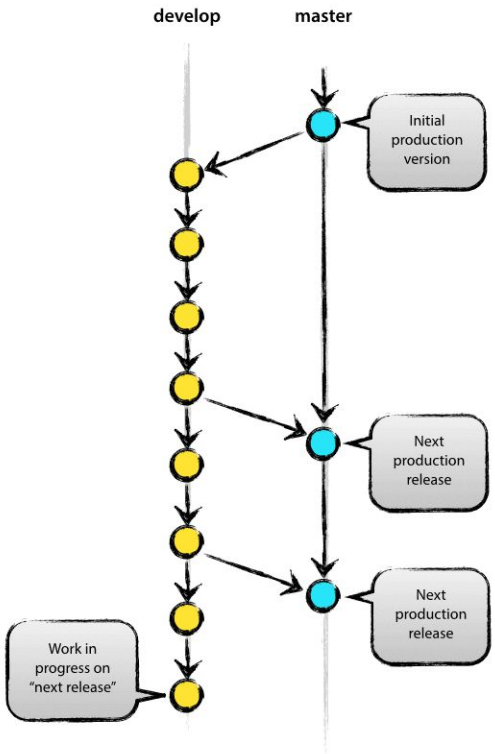
- Ordner auf dem Computer
- Enthält alle Daten
  - .git
  - Sourcecode
  - Doku
  - Lizenzen
  - etc.

Commit

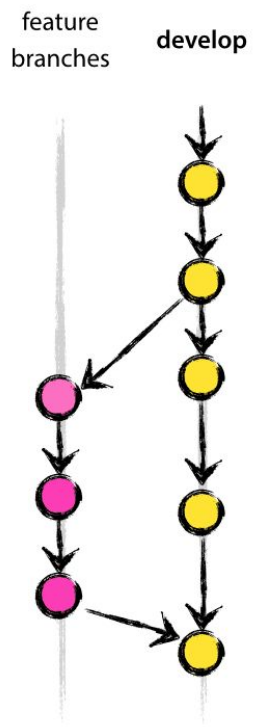
- “Änderung am Repository”
- Eindeutig durch Hash identifiziert (Hash basiert auf vorangegangen Commits)
- Wird durch den Entwickler manuell definiert → Änderungen an Dateien sind erst wirksam wenn sie in einen Commit sind
- **Commit History:** Liste der vorausgegangenen Commits

Branch

“Liste von Commits”



Main branches



Feature branches

Quelle: https://nvie.com/posts/a-successful-git-branching-model/

Pull

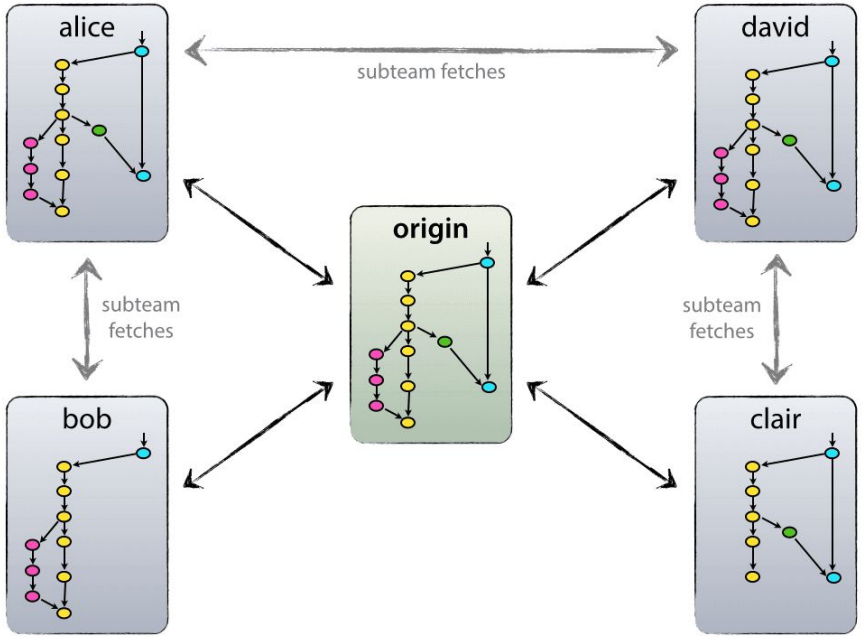
- Herunterladen & Integrieren von fremden Änderungen

Push

- Hochladen eigener Änderungen

Fetch/Merge

- Herunterladen von Änderungen
- Integrieren von Änderungen

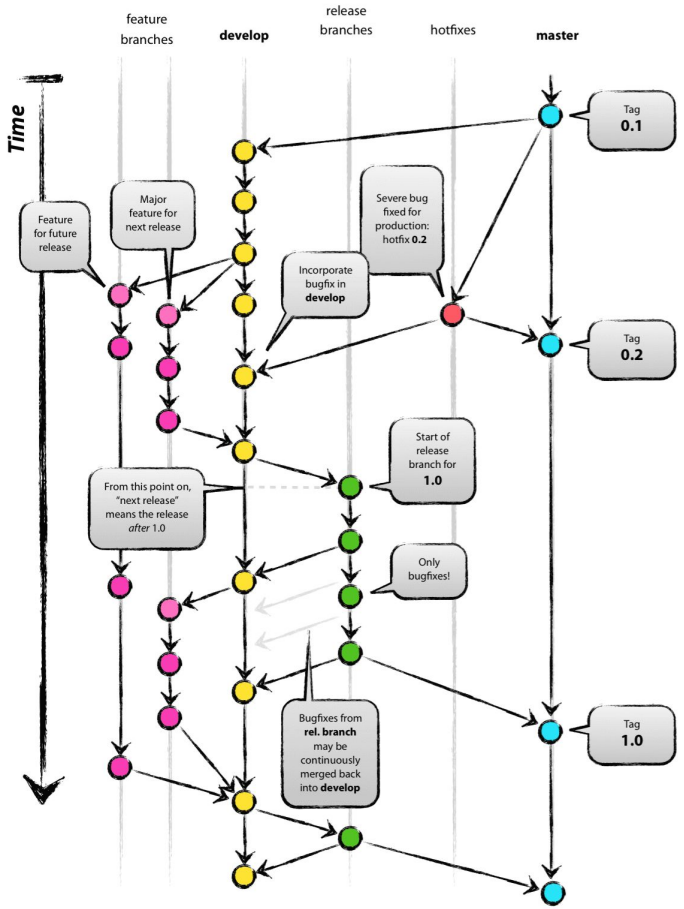


Wie benutze ich git?

- CLI
- GUI:
  - Git - GUI Clients
  - SourceTree
  - GitKraken
  - Git-Cola
  - Gitk
- Git Cheatsheets
  - git pretty: Git pretty
  - git flight rules: k88hudson/git-flight-rules: Flight rules for git

Realität

- Merge Konflikte
  - Mehrere Leute haben gleiche Codezeilen geändert
- Commits “mal eben schnell” ausserhalb von branches
- Versehentliche Commits müssten zurückgerollt werden



Alternativen

Interaktive Demo

1. Installation von git auf dem eigenen Computer
2. Anlegen eines zivgitlab/github Accounts
3. [Optional] Erstellen einer Organisation
4. Erstellen eines Basis-Repositories (pro Team)
5. Erstellen eines develop-branches
6. Klonen des Repositories auf den eigenen Computer
7. Initialer commit