Git-Crashkurs

im KBS SoSe 2021

Walter Stieben

17. Juni 2021

!!!Disclaimer!!!

Git ist riesig!

Wir haben wenig Zeit!

Fragen kostet nix!

Ich hab einen kaputten PC!

???

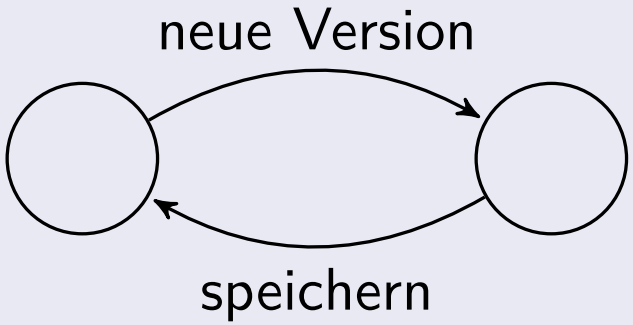
+ Profit!

Was ist Git?

Vorüberlegung

Typische Arbeit

- File erzeugen
- File speichern
- File ändern
- File wieder speichern



- Was hat sich verändert?
- Warum?
- Wer hat etwas geändert?

Version Control: Motivation

Aufgaben

- Versionshistorie ermöglichen
- Änderungen erkennen, darstellen und verarbeiten
- Synchronisation

Nutzen

- Dokumentation und Backup
- Zusammenarbeit
- Debugging
- Sandboxing / Branching und Merging

Version Control vor Git

- Zentralisierte Versionskontrolle mit z.B. SVN
- Proprietäres Bitkeeper als dezentrales VCS

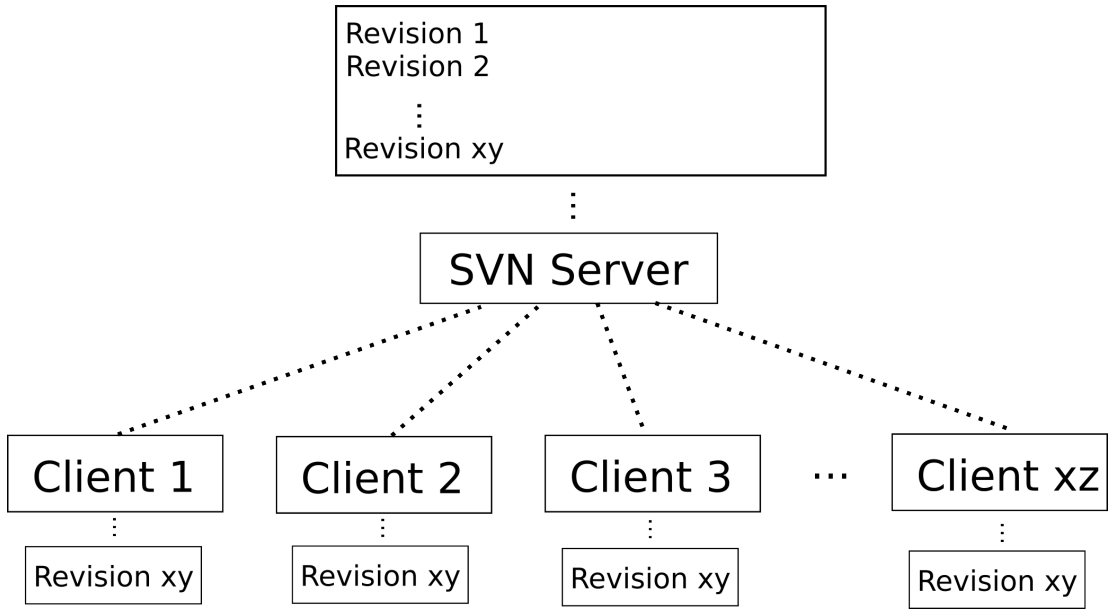


Abbildung: Quelle Q-0

Idee

Linus Torvalds

Abbildung: Quelle Q-1

- 2005: Linus Torvalds will Bitkeeper ersetzen
- Sollte den Anforderungen des Linux-Kernels genügen

Anforderungen

- Arbeit auch offline und verteilt
- Schnell, leicht und sparsam
- Ausfallsicherheit

Git: Funktionsweise

Komponenten

Trees, Blobs, Commits (& Tags)

- Jede Version einer Datei ist ein Blob
- Ein Dateiverzeichnis („Ordner“) ist ein Tree
- Ein Tree enthält weitere Trees und/oder Blobs
- Ein Tree mit einigen Zusatzinfos ist ein Commit
- Keine Änderungen (Deltas), sondern Zustände (Snapshots)
- Daten werden trotzdem komprimiert
- Jedes Objekt hat einen eindeutigen SHA-1-Hashwert

Speicherstruktur: Deltas

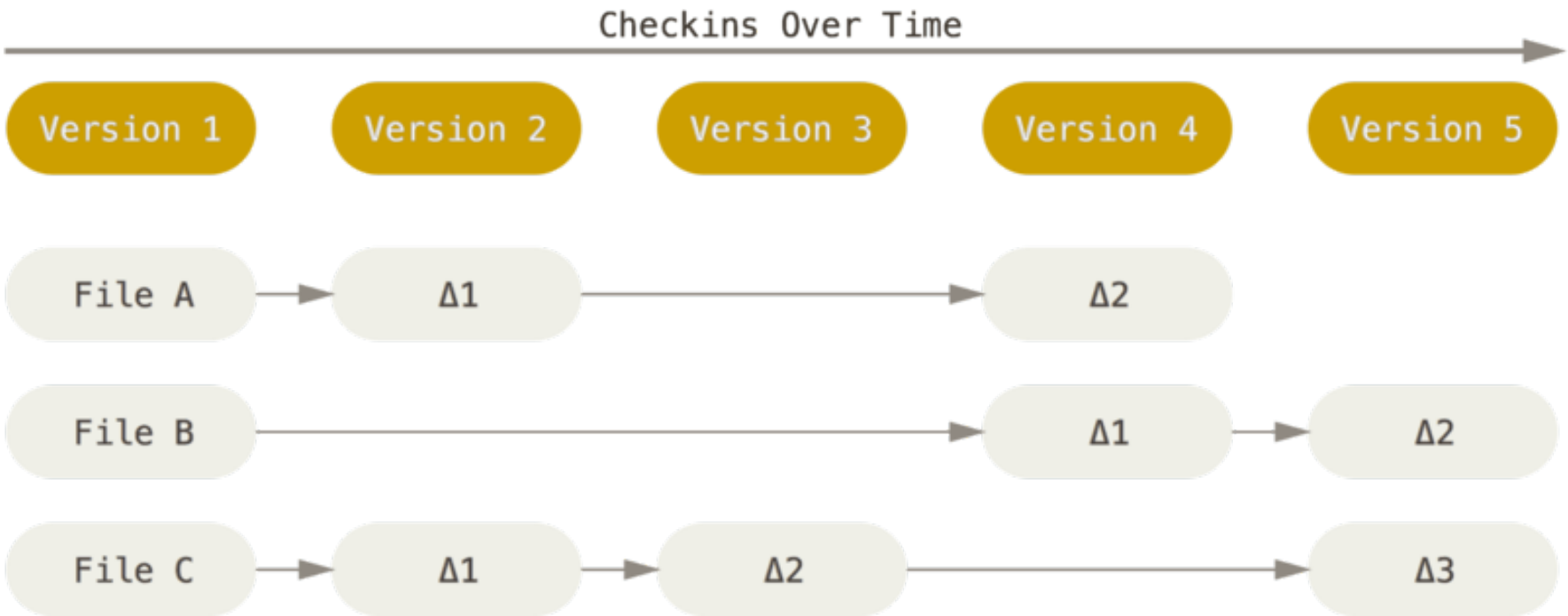


Abbildung: Quelle Q-2

Speicherstruktur: Snapshots

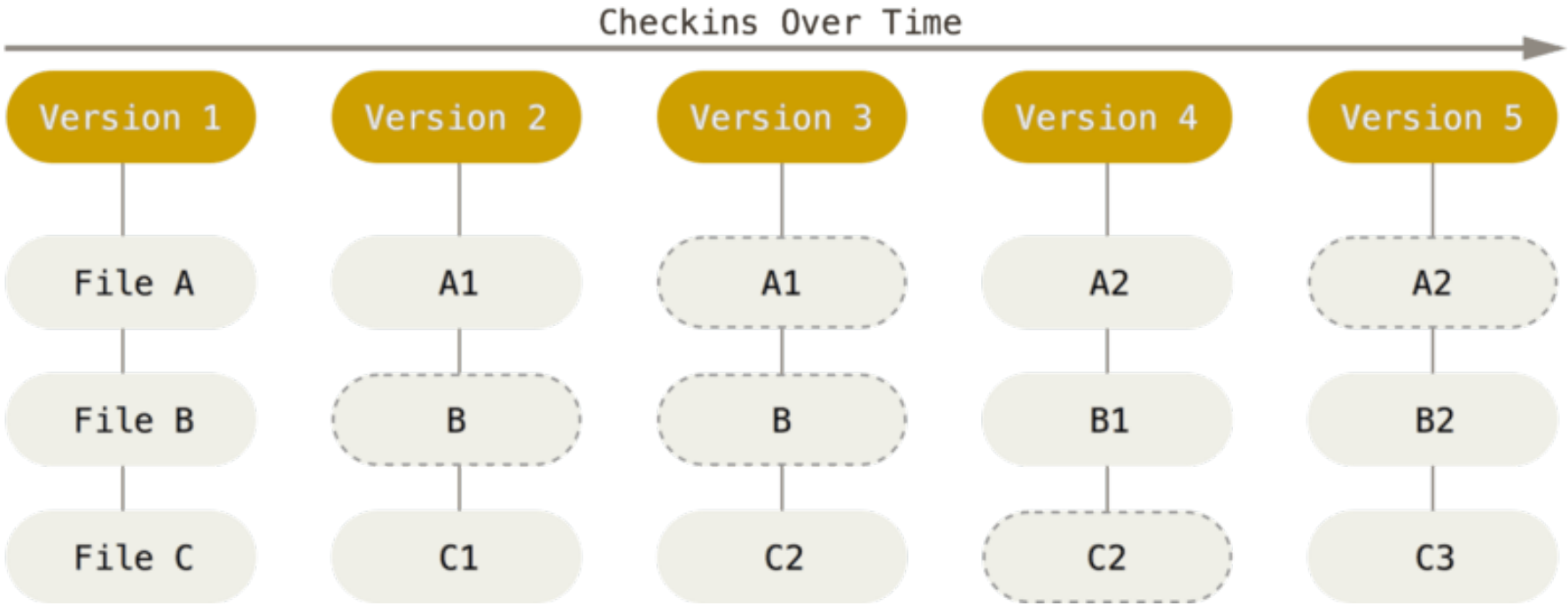


Abbildung: Quelle Q-3

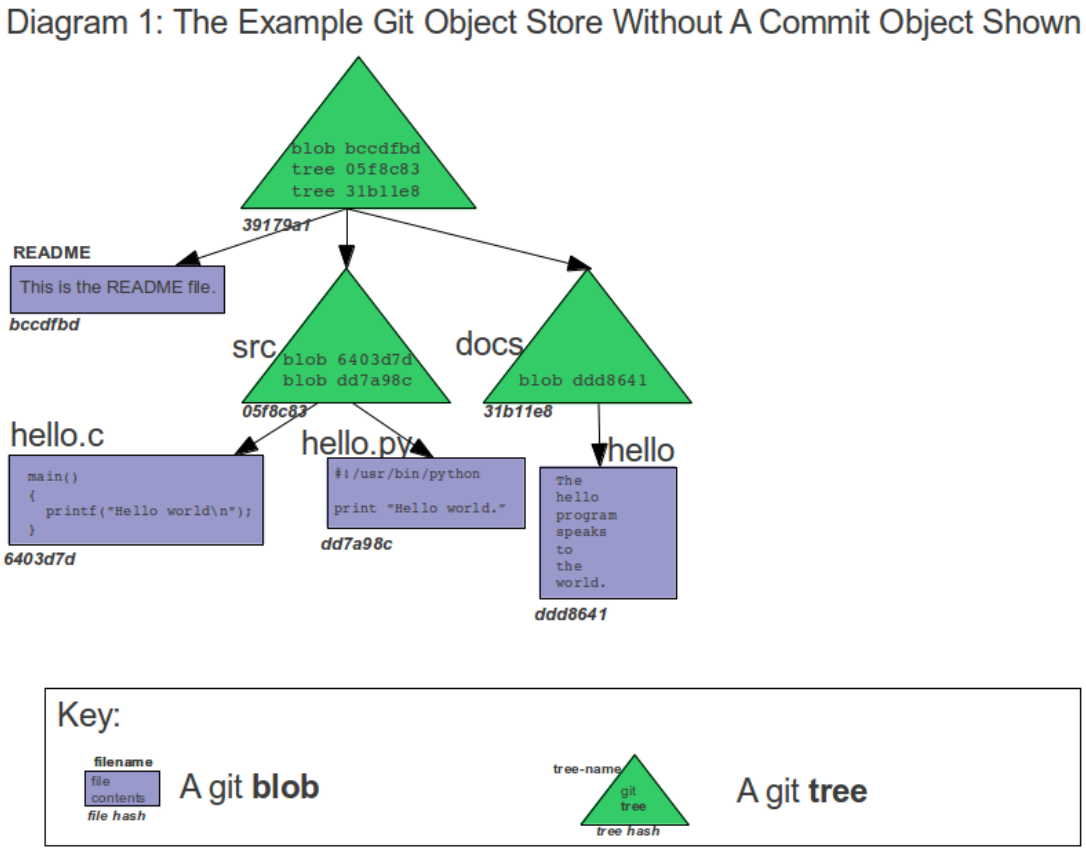


Abbildung: Quelle Q-4

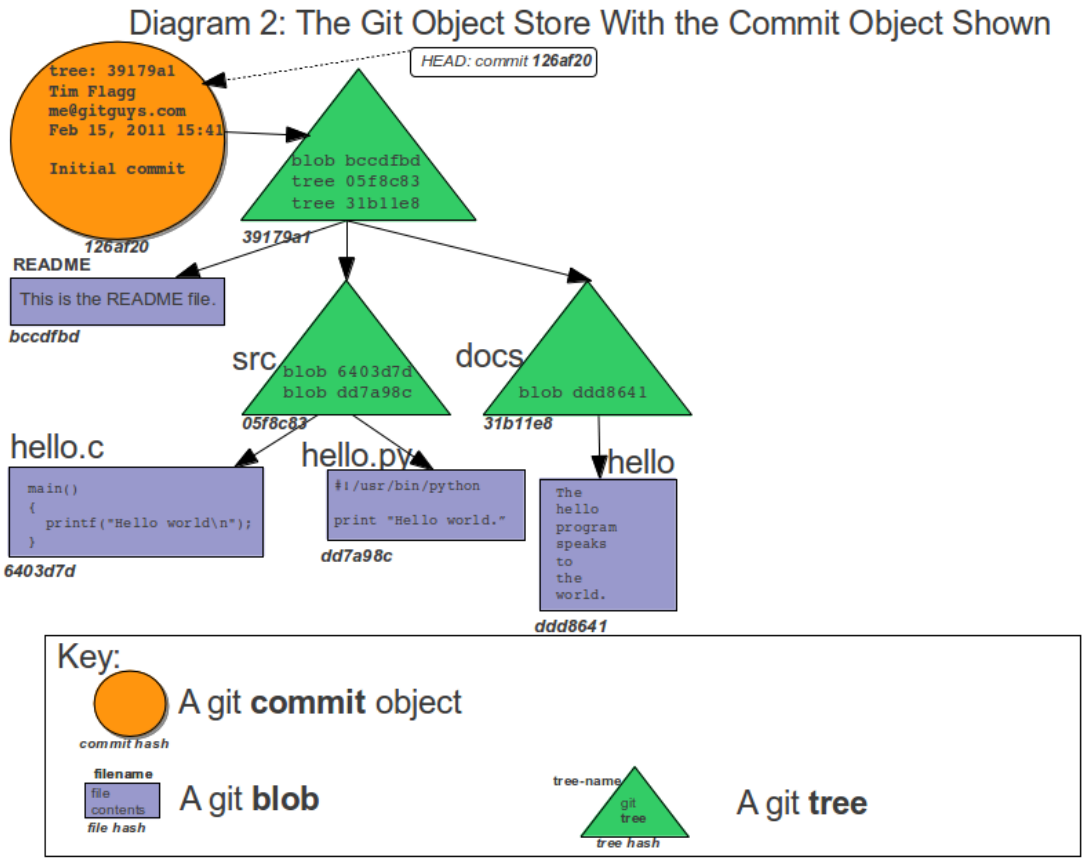


Abbildung: Quelle Q-5

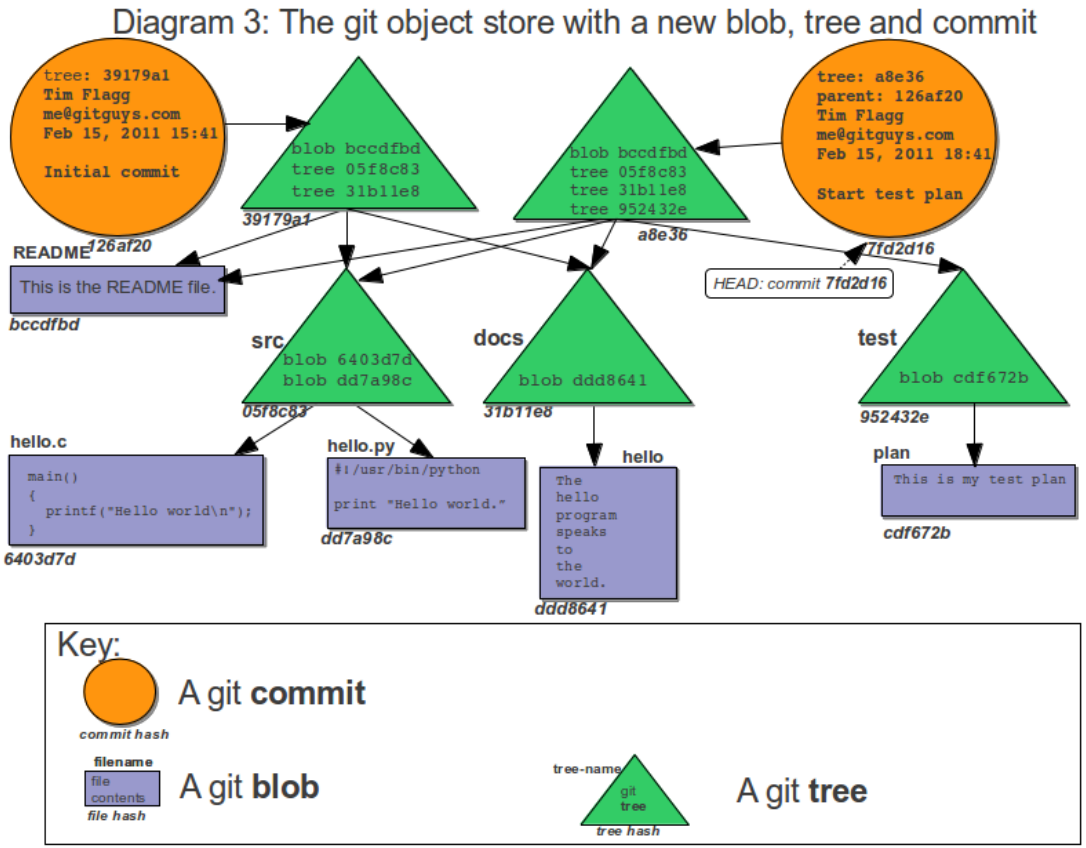


Abbildung: Quelle Q-6

Arbeiten mit Git: Basics

Arbeiten mit Git: Basics

- Installation
- Verbindung mit dem Remote Repository
- Commits und Stage
- Einfacher Workflow
- Branches
- Navigation
- Merge und Rebase
- Voller Workflow

Installation

Allgemeines

- Plattformübergreifend (Linux, Windows, Mac) verfügbar
- Zahlreiche GUIs vorhanden, aber nicht nötig
- Viele IDEs haben eingebaute VCS-Unterstützung
- Tools teilweise kostenpflichtig $\Rightarrow$ lieber Konsole beherrschen!

Download hier

`https://git-scm.com/download/`

Verbindung mit dem Remote Repository

Einrichtung des Clients

- Name und Emailadresse lokal hinterlegen
- Neues Repo erstellen oder vorhandenes klonen
- Verbindung zum Server über HTTPS (oder SSH...) einrichten

In unserem Fall

- MafiA hat Gogs (FBI hat Gitlab)
- Verbindung mit Mafiasi-Kennung (z.B. 17musterm)
- Nach Klonen des Remote-Repository sind Einstellungen für push und pull direkt richtig

Commits und Stage

Commits

- Momentaufnahme
- Veränderte Dateien gespeichert und referenziert
- Autor, Zeitstempel, Kommentar des Autors, Vorgänger

Stage

- eigentlich „Index“, oft aber „Staging Area“ oder „Stage“
- Zwischenschritt: Working Directory $\rightarrow$ Stage $\rightarrow$ Commit
- Nur zusammenhängende Änderungen in einem Commit!

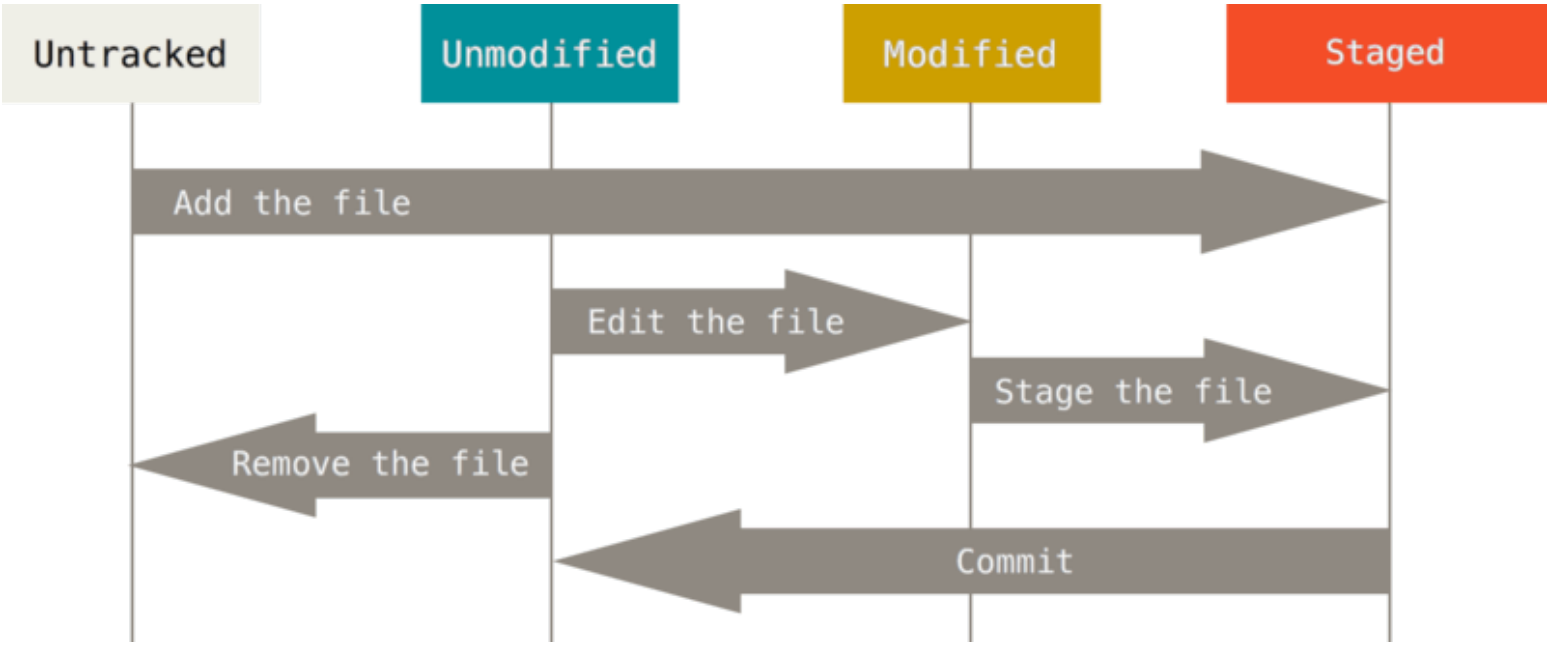


Abbildung: Quelle Q-7

Branches

- Branch = englisch für Zweig; hier: Entwicklungszweig
- Standardbranch: „master“
- Neuer Branch: isolierte Abspaltung von anderem Branch

Abbildung: Quelle Q-8

Navigation

Namen

- Branches = benannte Referenzen auf (aktuellste) Commits
- „HEAD“ = Referenz auf den Commit im Working Directory
- Tags können für alle Objekte vergeben werden

Checkout

- Springen zu jedem beliebigen Commit über Hashes
- Springen zu Branches über deren Namen
- Vorgänger mit „^“, z.B. „HEAD^“ oder „HEAD^^“
- X-ter Vorgänger mit „~X“
- Kann einzelne Dateien auswählen

History

log

- „git log“ zeigt die History an
- viele mögliche Parameter, um die Anzeige anzupassen
- z.B. Ausgabe für alle Branches, Ausgabe mit Graph
- Tools mit GUI verfügbar

Nutzen

- Übersicht!
- Commits suchen / Debugging, z.B. mit „git bisect“
- Ermöglicht Merges

Merge

- Verschiedene Versionen zusammenführen
- Branches vereinigen, z.B. nach parallelem Arbeiten
- Komplette Historie bleibt erhalten
- Bei Änderungen an derselben Stelle $\Rightarrow$ Konflikte

Rebase

- Branches auf den aktuellen Stand bringen
- Auch hier Konflikte möglich
- Verändert die Historie $\Rightarrow$ Vorsicht!

Bei Merge und Rebase sind der aktuelle Commit (HEAD) das Ziel und der als Parameter verlangte Commit der Ausgangspunkt.

Merge - Beispiel

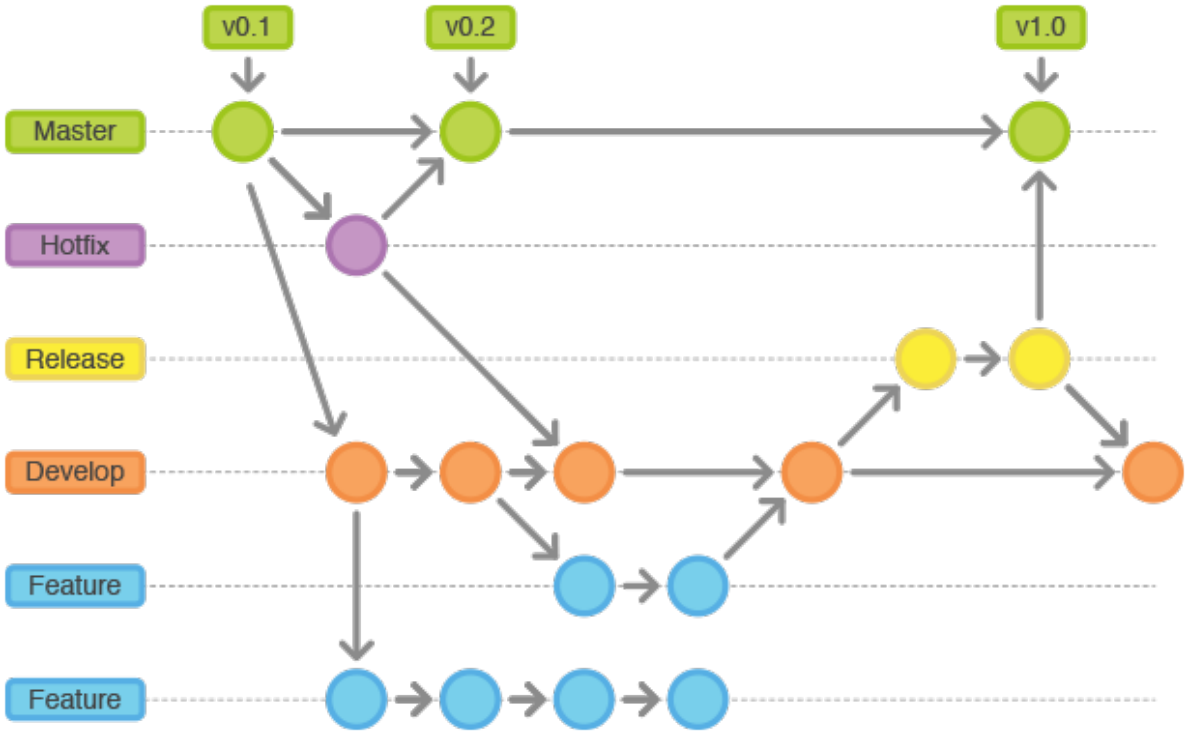


Abbildung: Quelle Q-9

Merge und Rebase - Unterschied I

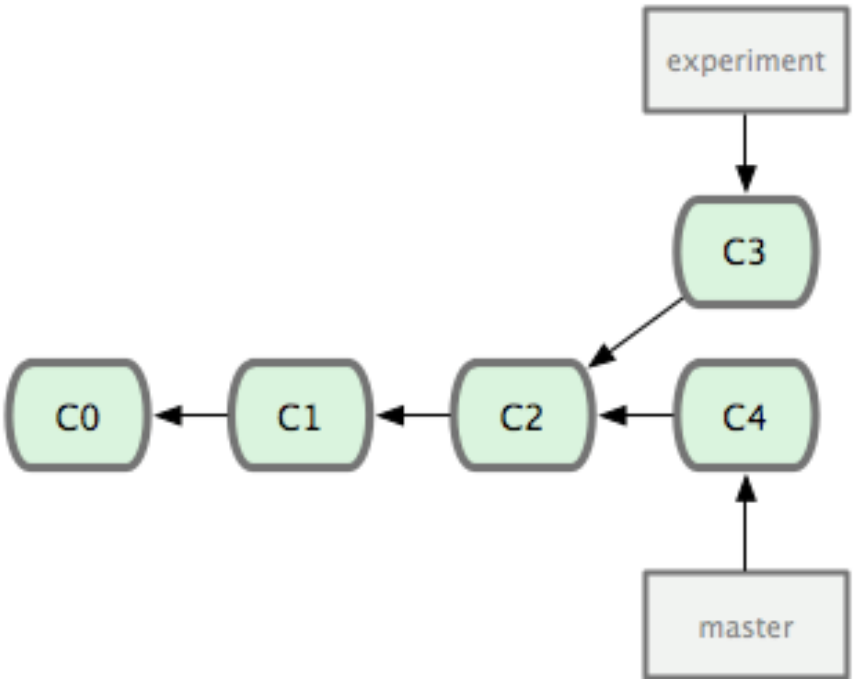


Abbildung: Quelle Q-10

Merge und Rebase - Unterschied II

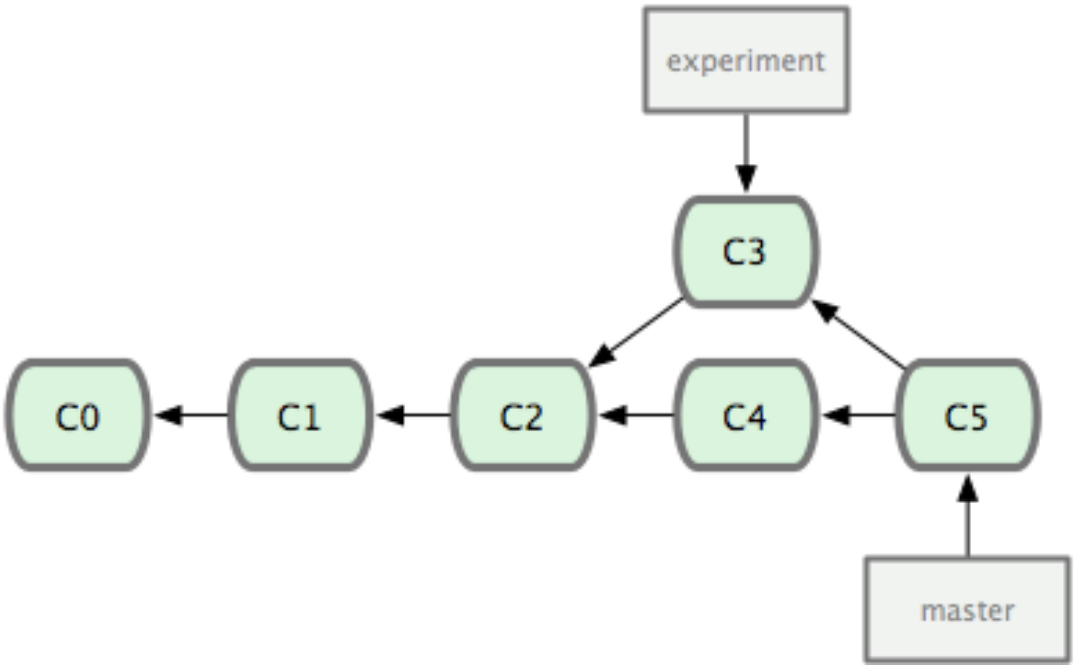


Abbildung: Quelle Q-11

Merge und Rebase - Unterschied III

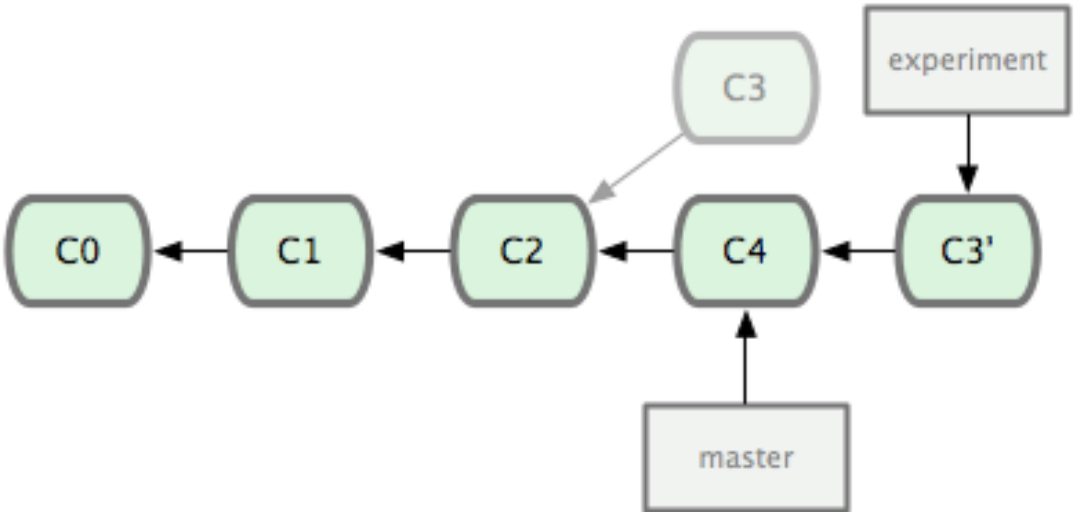


Abbildung: Quelle Q-12

Merge und Rebase - Unterschied IV

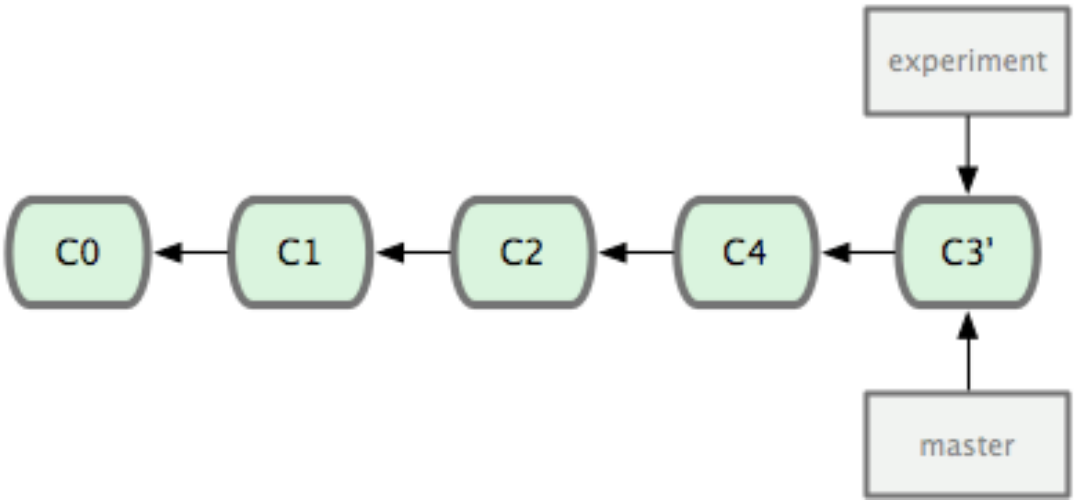


Abbildung: Quelle Q-13

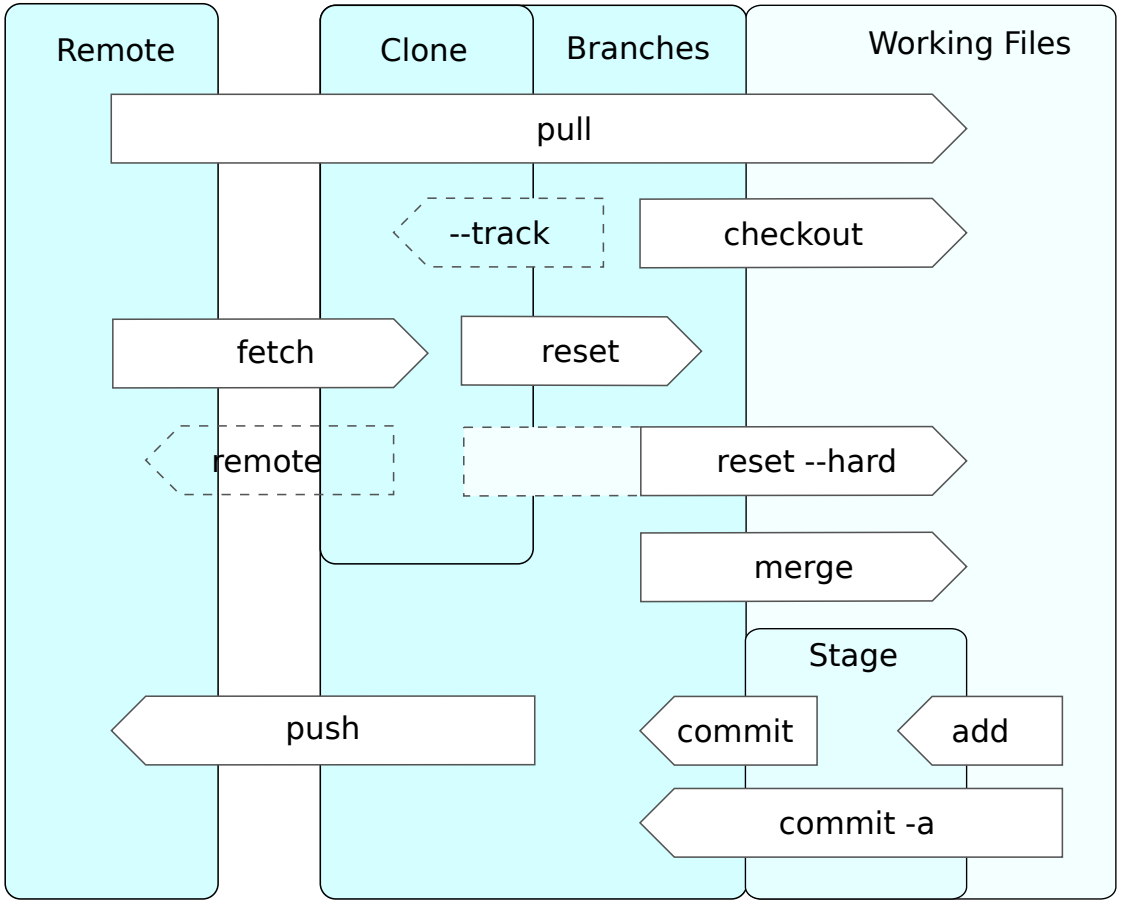


Abbildung: Quelle Q-14

Livedemo

Livedemo

- Was wollt ihr sehen?
- LearnGitBranching

!!!Livedemo!!!

Was wollt ihr sehen?

LearnGitBranching

Warum LearnGitBranching?

- Interaktive Beispiele
- Anschaulich dargestellt
- Schrittweise erklärt

Hier gehts zur Website!

Arbeiten mit Git: Advanced

Arbeiten mit Git: Advanced

- Konfiguration
- Index kontrollieren
- Reset
- Revert
- Stash
- Arbeiten mit Commits
- Editoren
- Log Advanced

Konfiguration

.gitconfig

- Einstellungen im Scope System, Nutzer oder Repository
- Eigene Befehle definieren ⇒ Shortcuts
- Commit-Templates, Standard-Editor, Autocorrect etc.

.gitignore

- Outputfiles enthalten keine eigene Information
- ständiges Kopieren ⇒ hoher Speicherverbrauch
- ausgewählte Dateien können „ignoriert“ werden
- Spezifikation mit regulären Ausdrücken, z.B. *.class
- Oft Templates vorhanden, z.B. für LaTeX

Index kontrollieren

git add --patch

- Genaue Kontrolle darüber, was auf den Index soll
- Blöcke lassen sich auch aufsplitten

Filesystem-Operationen

- „git mv“ zum Bewegen (= move) von Files.
- „git rm“ zum Entfernen (= remove) von Files.
- notwendiger Schritt, um Git die Veränderung klarzumachen

Verschiebt HEAD auf einen vorherigen Commit, ändert Historie

git reset --soft

- Index und Working Directory bleiben erhalten
- Index kann vom Target aus commited werden

git reset --mixed

- Index geht verloren, Working Directory bleibt erhalten
- Schnelles löschen des Index, außerdem Default für den Befehl

git reset --hard

- Index und Working Directory gehen verloren
- Praktisch zum schnellen Zurücksetzen des Working Directory

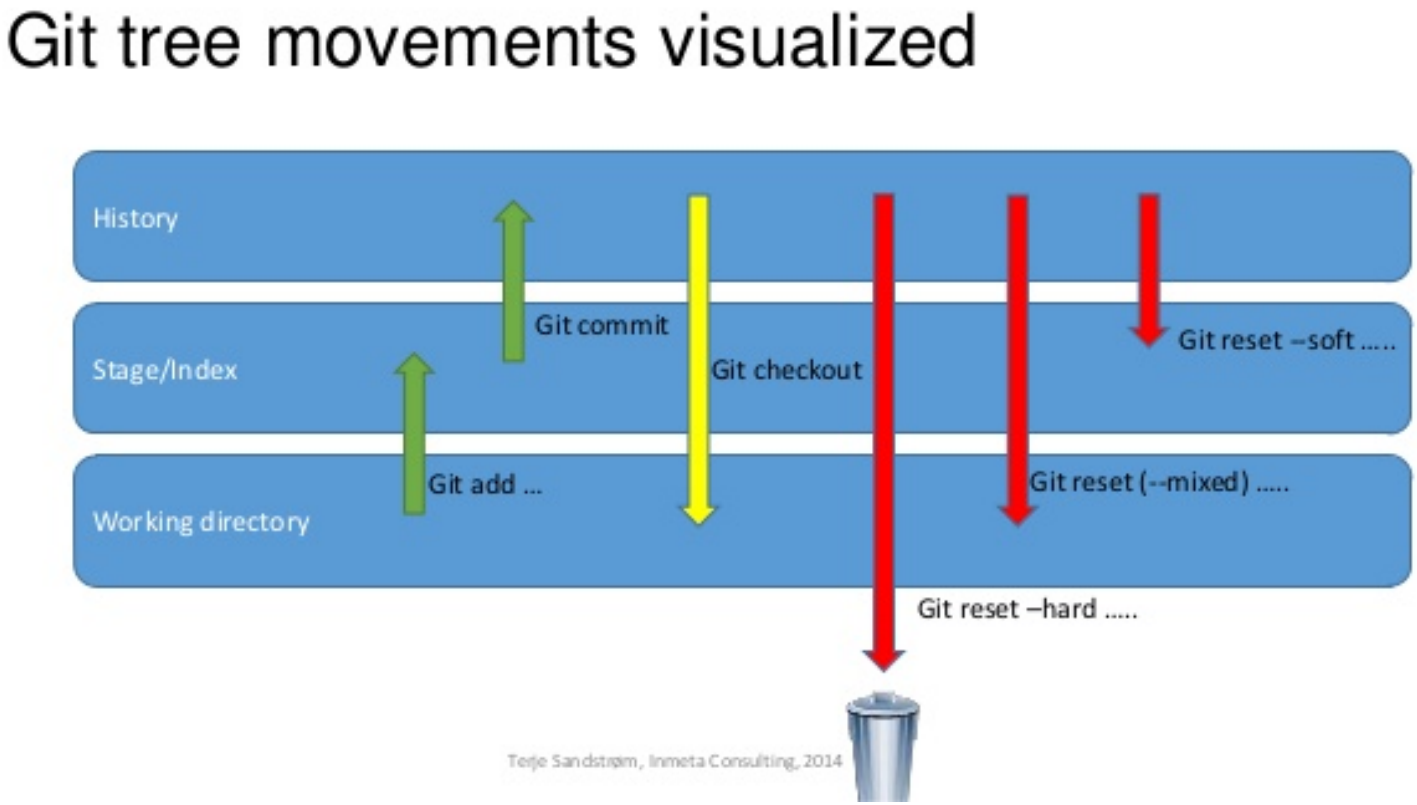


Abbildung: Quelle Q-15

Problem mit Reset

- Reset manipuliert die Historie
- „Die Geschichte neu schreiben“
- Öffentliche Trees müssen erhalten bleiben

git revert

- Macht stattgefundene Änderungen explizit rückgängig
- Historie bleibt vollständig erhalten
- Neue Commits werden hinten angehängt

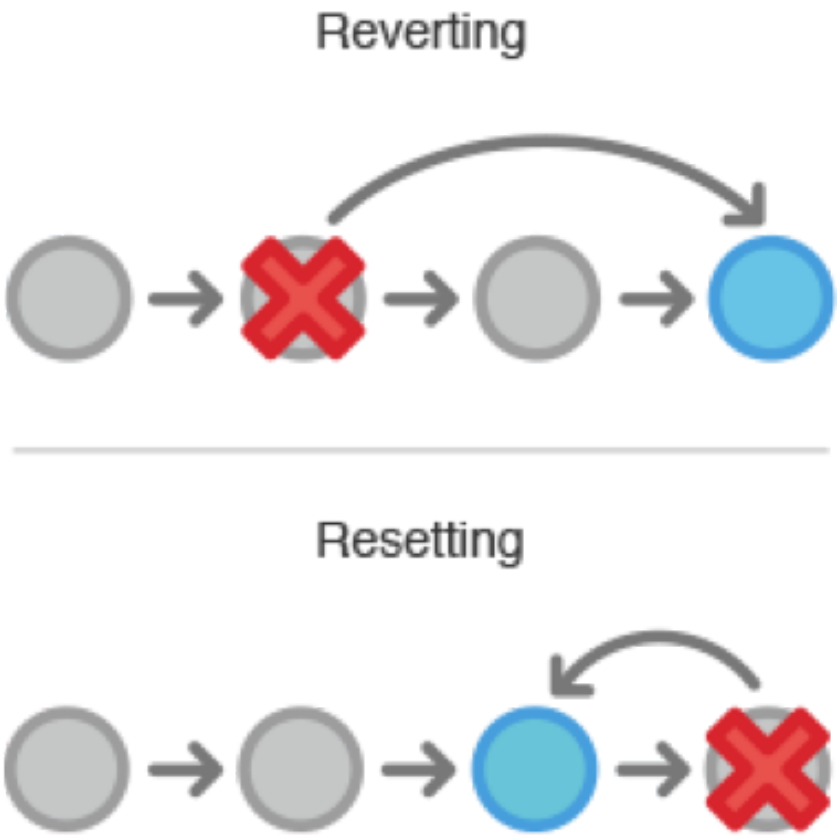


Abbildung: Quelle Q-16

Stash

- Zwischenspeicher für Änderungen
- Kurzfristige Lösung
- Funktioniert als Stack (LIFO)
- Befehle: „git stash“ (= push) & „git pop“

Einsatzgebiete

Bei nicht gespeicherten Änderungen

- zum Benutzen von „pull“ vor eventuellen Mergekonflikten
- zum Wechseln des Working Directory mit „checkout“

git cherry-pick

- Ein beliebiges Set von Commits heraussuchen und anwenden
- Ideal um die Historie frei von Debug-Commits zu halten

git rebase -i

- Interaktiv Reihenfolge von Commits ändern
- Kann Commits auch aus der Historie entfernen
- Manipuliert die Historie $\Rightarrow$ Vorsicht!

git commit --amend

- Schreibt den aktuellen Commit um
- Manipuliert die Historie $\Rightarrow$ Vorsicht!

Vim entkommen

- Standard-Editor, für unerfahrene User nicht bedienbar
- :q zum Verlassen
- :wq zum Schreiben und Verlassen

Editor ändern

- Andere Editoren lassen sich einstellen
- git config --global core.editor [editor-name]

log --graph

- Grafische Übersicht über Branches

Ausgabe anpassen

- log --pretty=oneline
- log --abbrev-commit

Shortcuts

- Alias definieren
- git config --global alias.lg "log --color --graph [etc.]

- Viele weitere Ideen hier und hier
- „q“ zum Verlassen der Übersicht

Best Practices und Diskussion

Best Practices

- **Saubere Historie!**
- Kleine Commits, inhaltliche Einheit
- Gute Commit-Nachrichten brauchen Zeit
- „Branch early & branch often“
- Rebases können die Historie sauberer halten
- Historie nur in privaten Trees manipulieren
- Merges nur mit gutem Grund
- Nur getesteten Code pushen
- Möglichst keine großen Files & solche nicht ändern
- Verteiltes Arbeiten braucht klare Aufgabenteilung

Alles Übungssache!

Quellen

Quellen

- https://git.informatik.uni-hamburg.de/help
- https://git-scm.com/book/en/v2
- https://git-scm.com/docs
- http://gitimmersion.com/lab_01.html
- https://rogerdudler.github.io/git-guide/

Bildquellen I

Bildquellen [Q-]

0. https://upload.wikimedia.org/wikipedia/commons/7/73/SVNvsGITServer_1.png
1. https://upload.wikimedia.org/wikipedia/commons/6/69/Linus_Torvalds.jpeg
2. https://git-scm.com/book/en/v2/images/deltas.png
3. https://git-scm.com/book/en/v2/images/snapshots.png
4. http://www.gitguys.com/wordpress/wp-content/uploads/2011/05/03/img02.png

Bildquellen II

5. http://www.gitguys.com/wordpress/wp-content/uploads/2011/05/03/img12.png
6. http://www.gitguys.com/wordpress/wp-content/uploads/2011/05/03/img22.png
7. https://git-scm.com/book/en/v2/book/02-git-basics/images/lifecycle.png
8. https://blog.seibert-media.net/wp-content/uploads/2014/03/Gitflow-Workflow-1.png
9. https://blog.seibert-media.net/wp-content/uploads/2014/03/Gitflow-Workflow-4.png
10. https://git-scm.com/figures/18333fig0327-tn.png

Bildquellen III

11. https://git-scm.com/figures/18333fig0328-tn.png
12. https://git-scm.com/figures/18333fig0329-tn.png
13. https://git-scm.com/figures/18333fig0330-tn.png
14. https://upload.wikimedia.org/wikipedia/commons/d/d8/Git_operations.svg
15. https://stackoverflow.com/questions/3528245/whats-the-difference-between-git-reset-mixed-soft-and-
16. https://alexdiliberto.com/talks/all-things-git/img/revert_reset.png