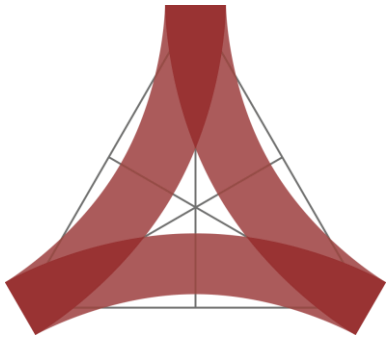
Git Schulung - Basics

academy consult
STUDENTISCHE UNTERNEHMENSBERATUNG



Yannick Gehring

Freitag, 03.04.2020

Agenda

- Versionsverwaltung und Git – Einführung
- Git – Interaktiv
- Git – Eine kleine Übung

Versionsverwaltung ermöglicht Nachvollziebarkeit, Autorenschaft und einfache Rollbacks

Versionsverwaltung und Git – Einführung

Was wurde geändert?

Wer hat diese Änderung gemacht?

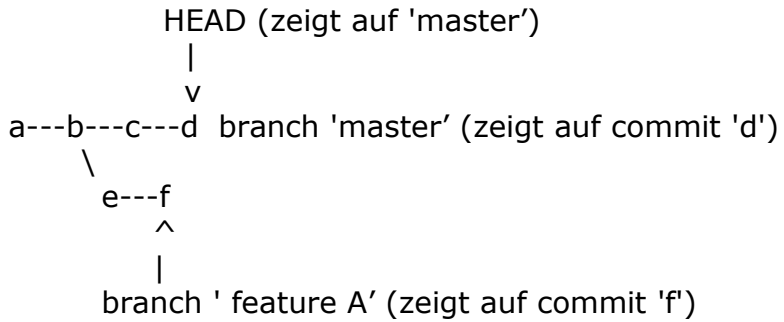
Wann hat diese Änderung stattgefunden?

Wieso wurde diese Änderung gemacht?

Git ist die meistgenutzte Versionsverwaltung unter Software Entwicklern

Versionsverwaltung und Git – Einführung

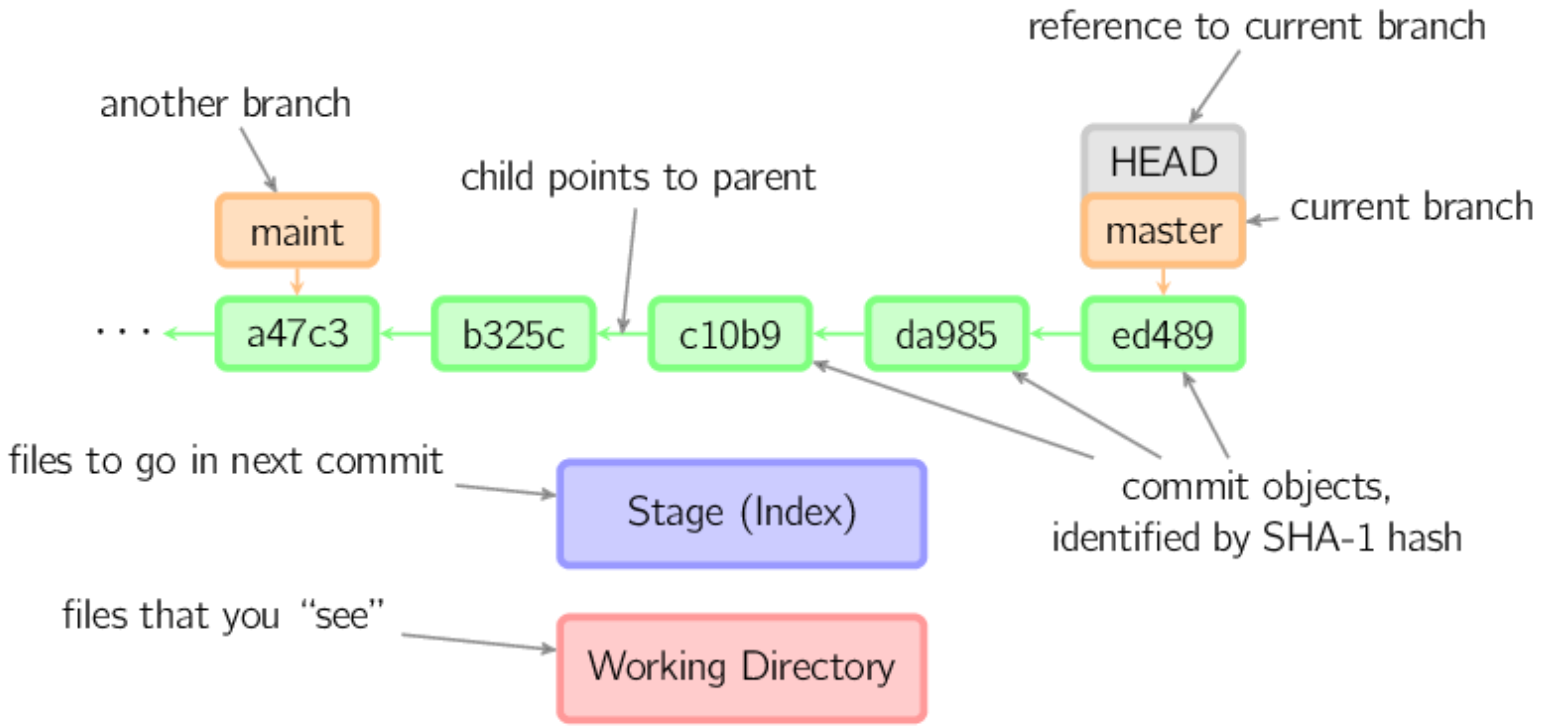
- 2005 von Linus Torvalds erfunden
- Git ist eine verteilte Versionsverwaltung ➔ kein zentraler Server, jeder besitzt eine lokale Kopie des gesamten Repositorys
- Auf Github können Git Repositories gehostet und damit mit anderen geteilt werden
  - Github stellt Möglichkeiten zur einfachen kollaborativen Zusammenarbeit, "soziale" Features und eine einfache Benutzeroberfläche bereit
  - Auf Github wird eine Vielzahl von open source Software gehostet
- **Repository**
  - Sammlung aller commits, branches, quasi ein Ordner mit Unterordnern und Dateien
- **Commit**
  - Sammlung an Änderungen an Dateien, (Unter)Ordnern etc.
  - Enthält Metadaten (Commit message, Autor, Zeitstempel)
  - Commits als inkrementelle Änderung (Commit A --> Commit B --> Commit C)
- **Branch**
  - Ist ein benannter Pointer auf einen commit
  - master ist der default branch
- **HEAD**
  - Der aktuell ausgecheckte commit
  - Normalerweise: der letzte commit im aktuellen branch
- **Remote**
  - Ein remote repository, mit dem ich meine Änderungen synchronisieren kann
  - Pull: Änderungen vom remote holen
  - Push: remote mit meinen Änderungen updaten



Ein repository hat mehrere "Komponenten"

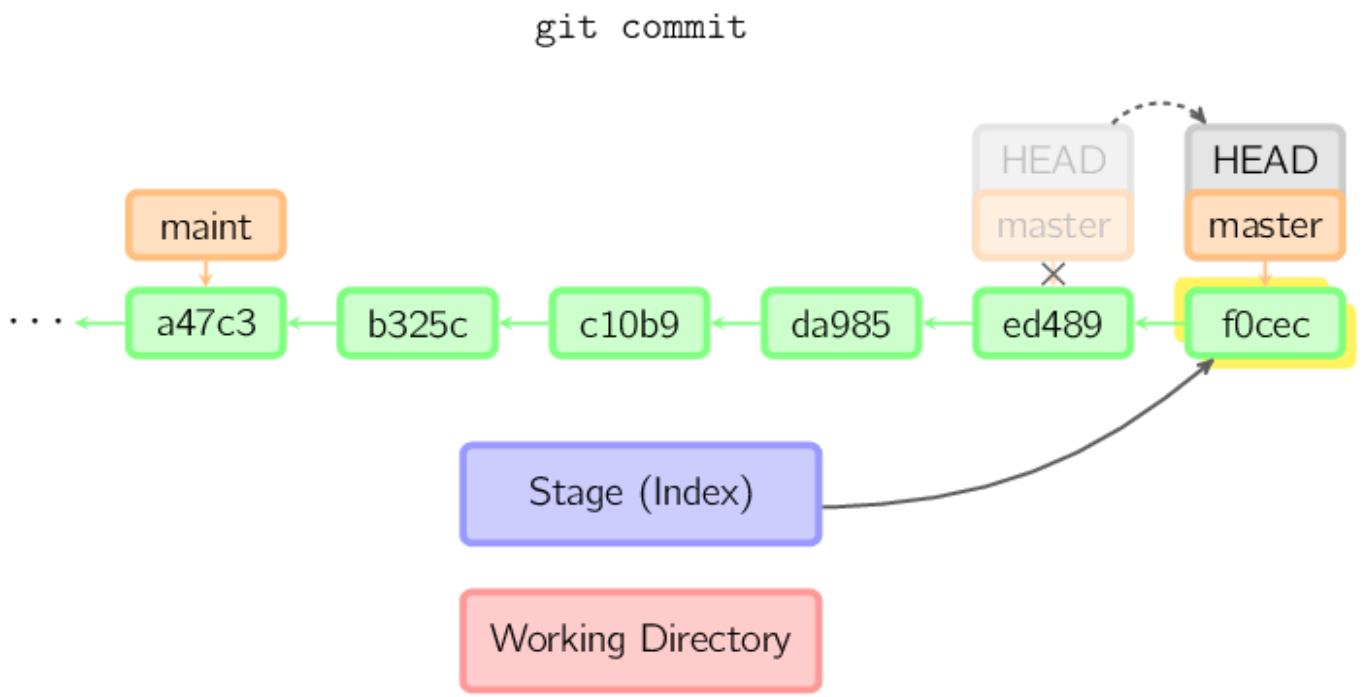
Git – Interaktiv

- Repositorys können von einem remote (z.B. Server/Github) gekloned werden
- Working directory: die "wirklichen" Dateien
- Index: staging area. Das soll commited werden
  - Mit git add <Datei> können Dateien zum index hinzugefügt werden



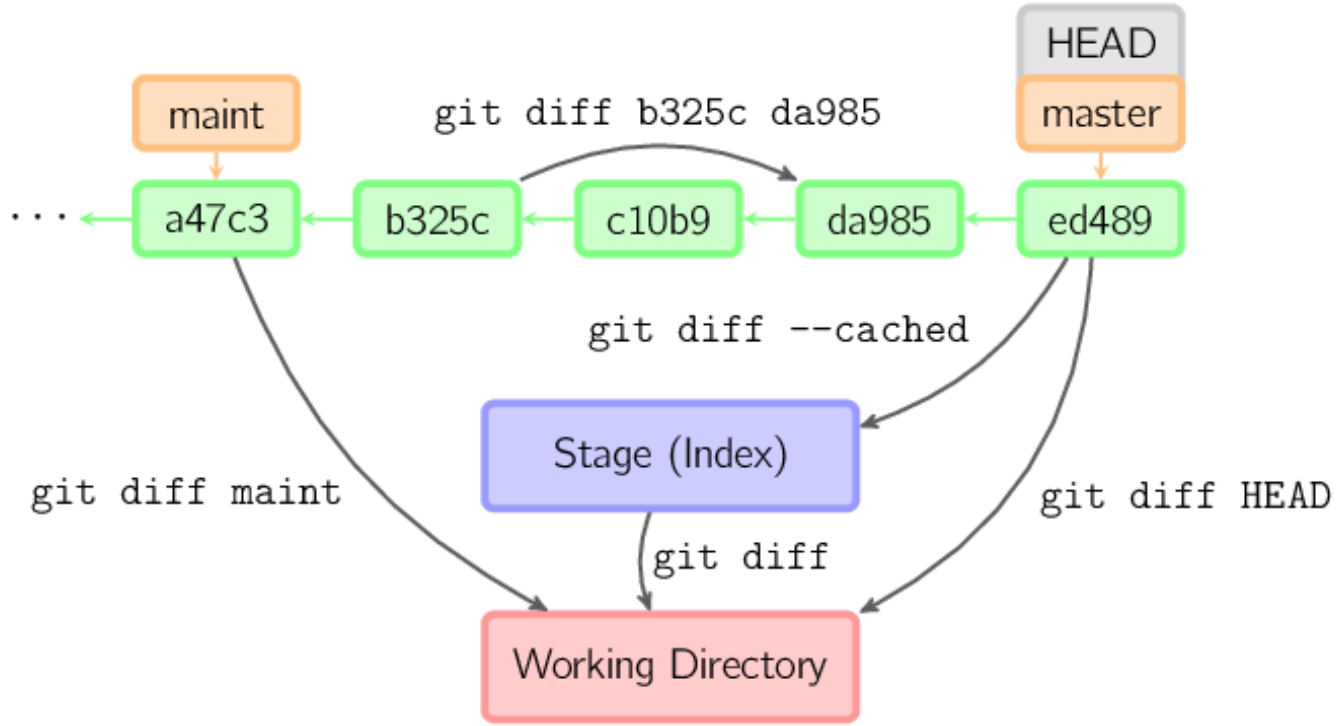
Ein neuer commit wird am Ende eines branches hinzugefügt

Git – Interaktiv



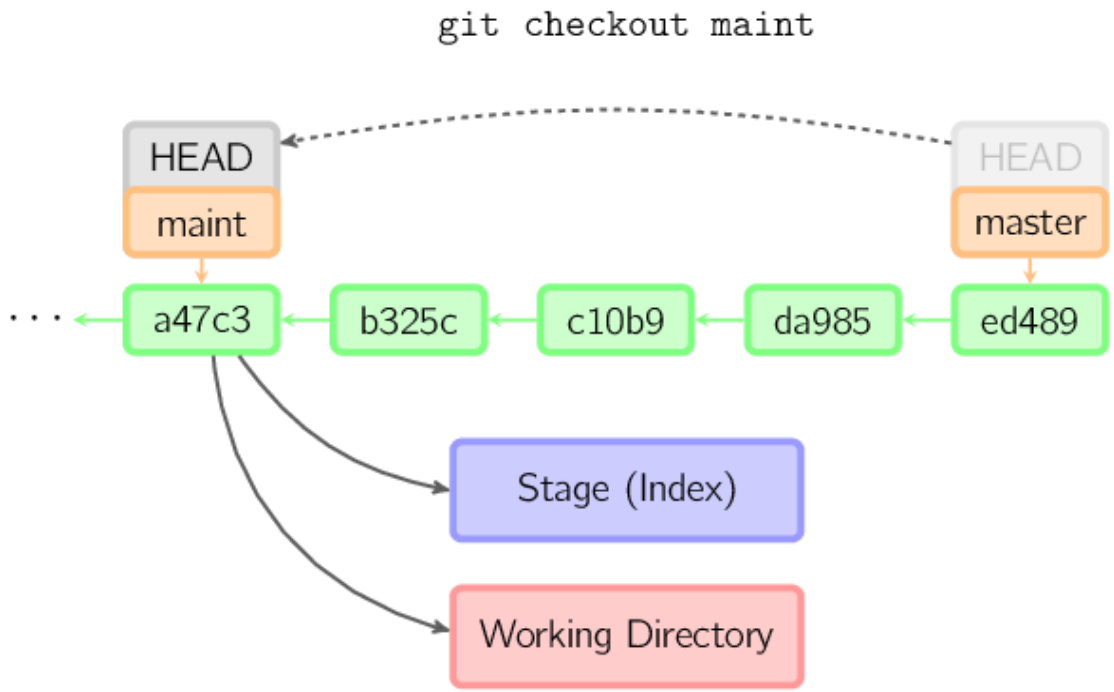
Vor dem commiten sollten die genauen Änderungen nochmals angeschaut werden

Git – Interaktiv



Mit einem Branchwechsel kann der aktuelle Stand eines anderen branches geladen werden

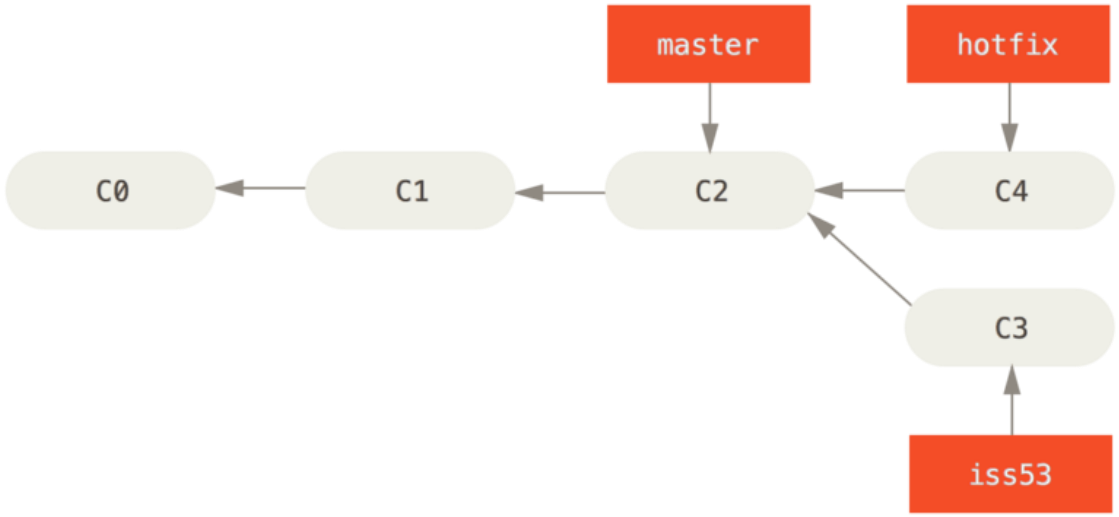
Git – Interaktiv



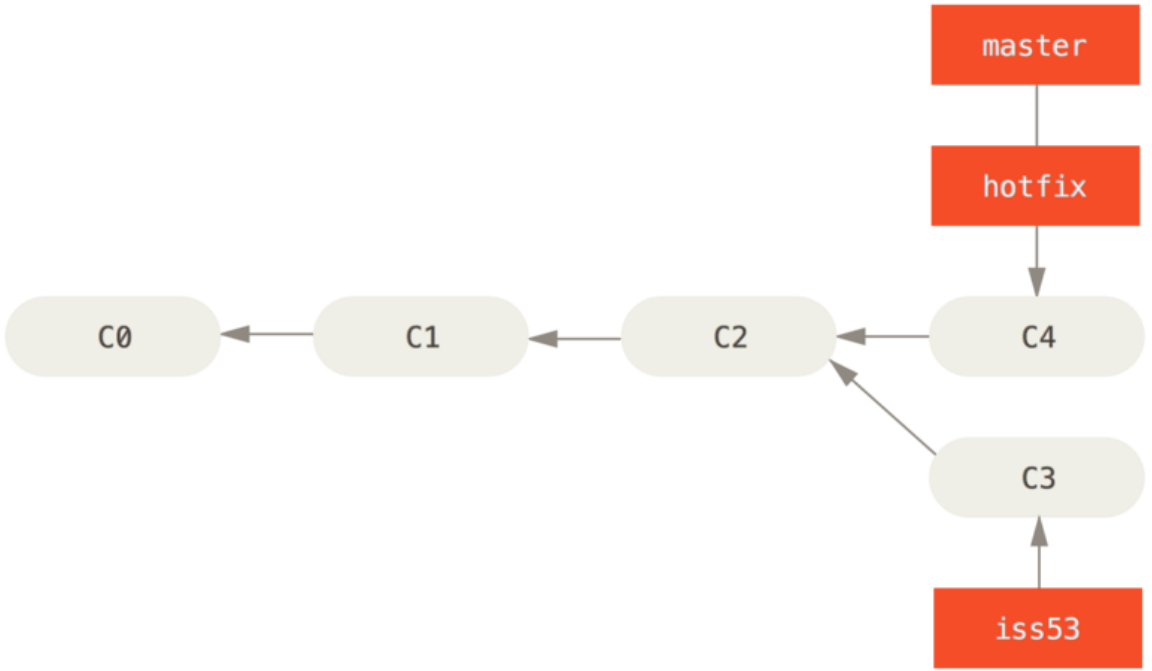
- Verwendung verschiedener branches:
  - Branch für Feature A, um Feature A unabhängig entwickeln zu können
  - Nach der Entwicklung können die Änderungen zusammengeführt (merge) werden
- Ein Branchwechsel kann Änderungen/Dateien im aktuellen Index bzw. Working Directory löschen
  - Man bekommt eine Fehlermeldung
  - Lösung 1: man commited seine Änderungen
  - Lösung 2: man stashed seine Änderungen, um sie später wieder anwenden zu können

Git merging – Fast-forward merge

Git – Interaktiv

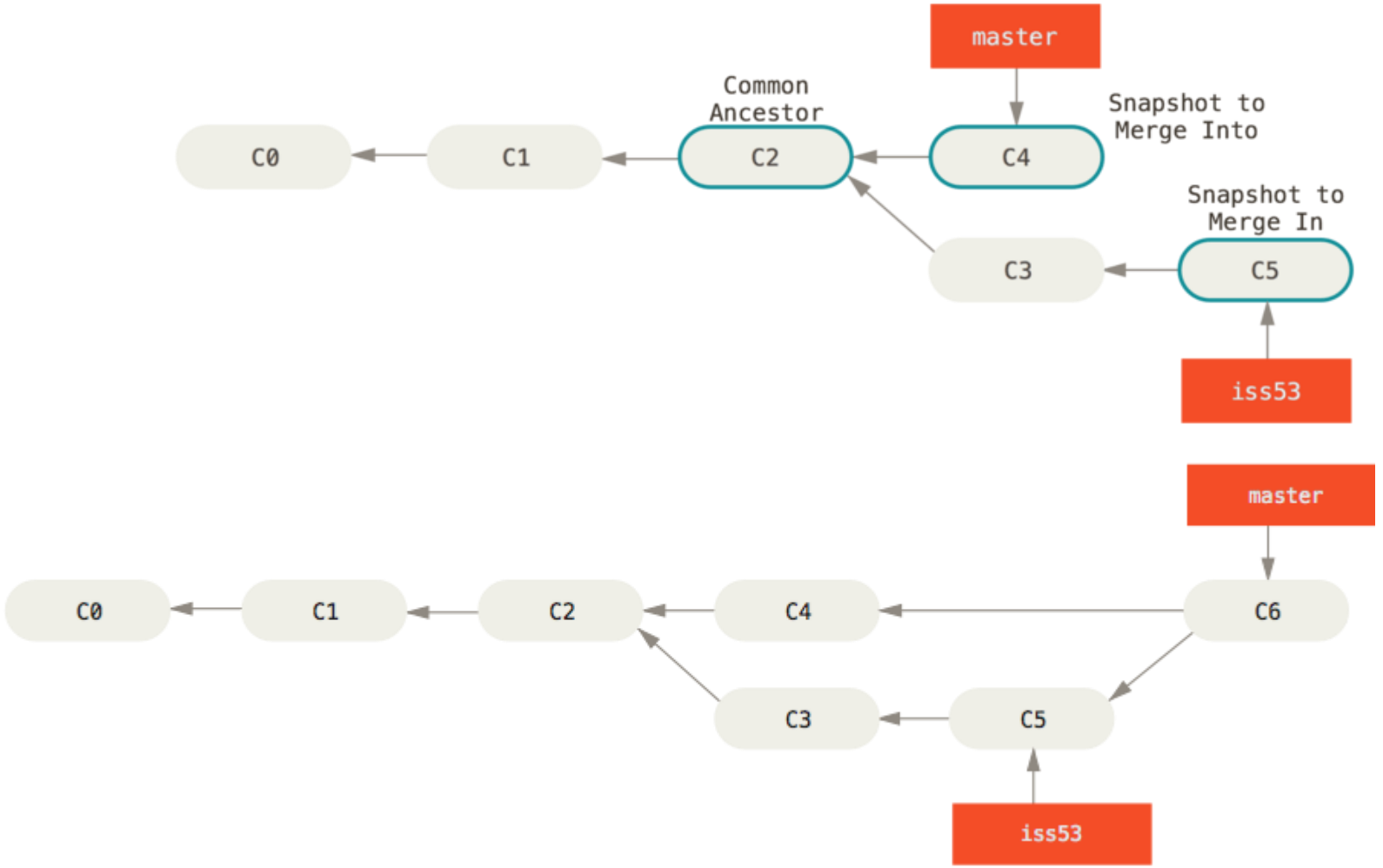



```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast-forward
 index.html | 2 ++
 1 file changed, 2 insertions(+)
```



Git merging – merge commit

Git – Interaktiv




```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by the 'recursive' strategy.
index.html |    1 +
1 file changed, 1 insertion(+)
```

Das lokale Repository kann mittels dem pull Befehl aktualisiert werden

Git – Interaktiv

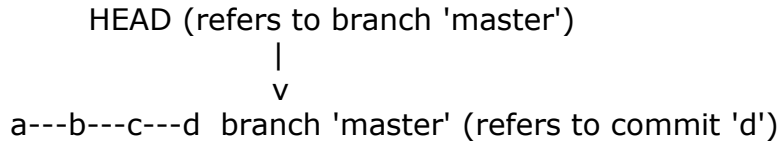
- Änderungen von anderen Leuten herunterladen:
  - git pull
  - Besteht aus git fetch und git merge
  - git fetch ➔ läd alle neuen commits herunter
  - git merge ➔ merged die commits in den lokalen branch

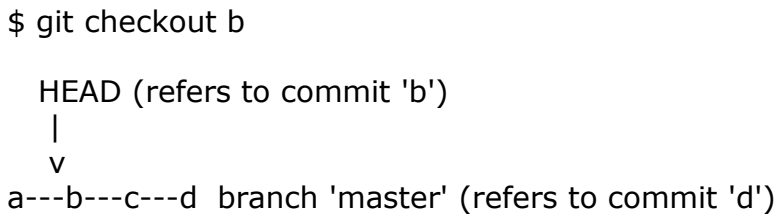
- Eigenen Änderungen "hochladen"
  - git push remote branch
  - git clone setzt remote automatisch
  - Falls nicht, kann das mit 'git remote add remote_name <server>' gemacht werden

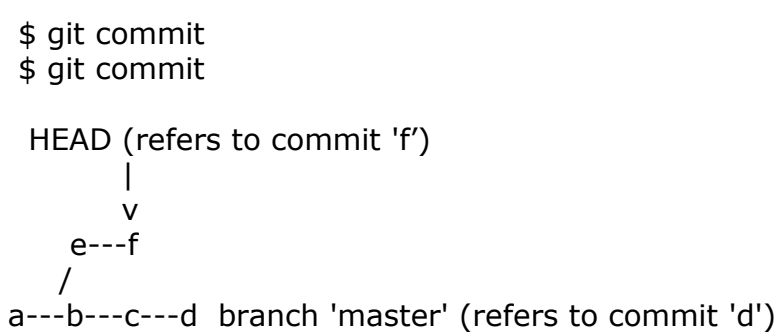
Detached head state – Was tun?

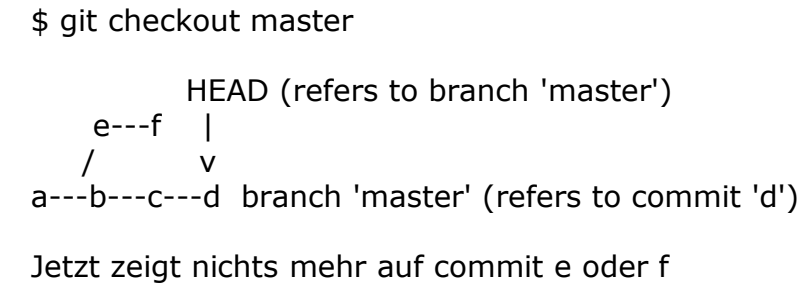
Git – Interaktiv

- Wenn wir direkt einen spezifischen commit auschecken, und nicht einen branch, landen wir im detached head state



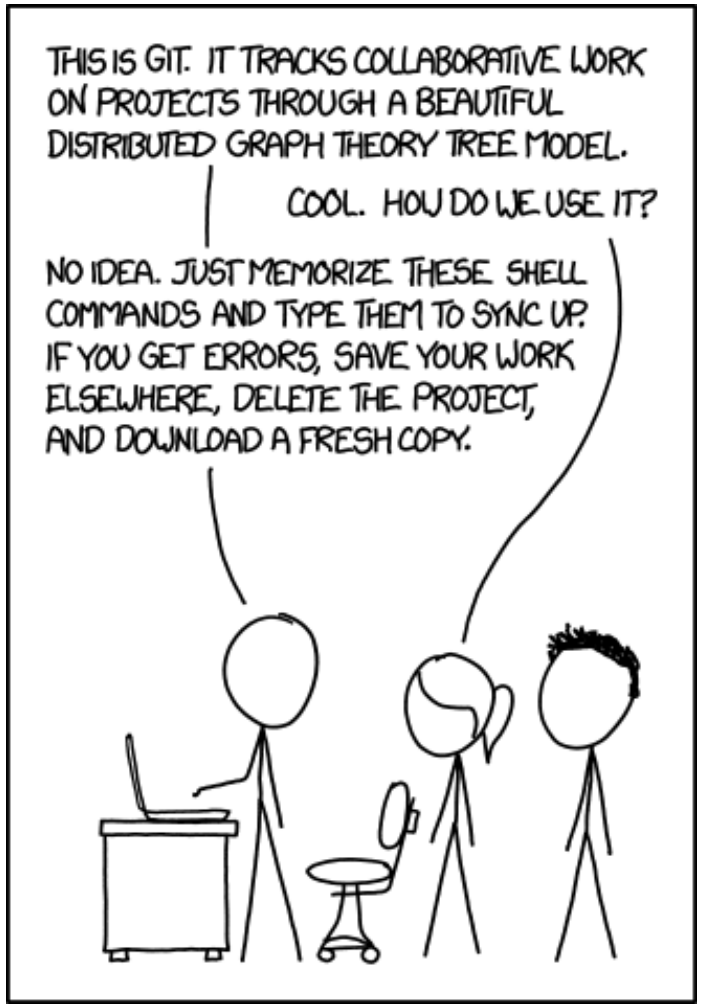






Anfängliche Schwierigkeiten mit git sind normal :-)

Git – Nähere Betrachtung



Die Website eurer JE wird überarbeitet und eure Expertise ist gefragt!

Git – Eine kleine Übung

- Repository mittels *git clone ssh://git@git.bdsu.it/repos/website.git* klonen
  - Password: bdsuit<3
- Wir haben festgestellt, dass "Keine Konzepte aus der Schublade" nicht gut bei Unternehmen ankommt.
  - Kannst Du hier einen besseren Grund auf unsere neue Website schreiben und die Änderung anschließend committen? Denke auch dran deine JE Mail als Autor zu verwenden, sowie eine aussagekräftige commit message zu schreiben.
    - Hinweis: einfach mit einem normalen Texteditor bearbeiten
  - Wer möchte kann auch die Hintergrundfarbe seiner JE anpassen.

- Um noch mehr coole Projekte an Land ziehen zu können haben wir das interne Projekt "Gute Gründe für unsere JE" gestartet.
  - Erstelle bitte deinen eigenen feature branch.
  - Anschließend kannst du neue Gründe für deine JE hinzufügen und committen.
  - Das interne Projekt hat ergeben, dass ein Beispielprojekt die conversion rate signifikant erhöht.
  - Füge ein Beispielprojekt in einem Stichpunkt hinzu und committe deine Änderungen.
  - Nun ist die neue Webiste fertig – gibt deinem letzten commit deswegen den Tag v2.0.
  - Pushe deinen neuen branch

- Bonus: Dein IT-Teamleiter hat anscheinend schlecht geschlafen und verlangt von dir, dass du die Reihenfolge der commits änderst. Leider stellt sich der Vorstand hinter den Teamleiter, weswegen dir keine andere Möglichkeit bleibt, als diesem Wunsch nachzukommen.
  - Ändere die Reihenfolge der commits
  - Kann so eine Änderung zu Problemen führen?
  - Hinweis: Benutze hierfür rebase im interaktiven Modus.

Übersicht – Befehle

Git – Eine kleine Übung

- **Hilfe zu einem Befehl**
  - git help <command>
- **Repository klonen:**
  - git clone <url>
- **Status des Repos abrufen:**
  - git status
- **Dateien zum Index hinzufügen:**
  - git add file1 file2
- **Commiten**
  - git commit
- **Letze commits anschauen**
  - git log
- **Neuen branch erstellen und hin wechseln:**
  - git checkout –b new_branch_name
- **Branch wechseln:**
  - git checkout branch_name
- **Neuen branch pushen:**
  - git push origin <branch>
- **Änderung im working directory anzeigen**
  - git diff
- **Dateien zurück setzen**
  - git checkout -- <filename>
    - Setzt die Datei auf den letzten Stand im aktuellen branch zurück
    - Änderungen, die schon zum Index hinzugefügt wurden sind nicht betroffen
- **Lokale Änderungen rückgängig machen:**
  - Auf Stand eines branchs
    - git reset --soft origin/branch_name
  - Auf Stand eines commits
    - git reset --soft <commit_hash>
  - --soft kann mit --hard ausgetauscht werden
    - --hard löscht alle Dateien aus dem Index und working directory
    - --soft behält die Datein im working directory

Vielen Dank für eure Aufmerksamkeit!

academy consult
STUDENTISCHE UNTERNEHMENSBERATUNG

Yannick Gehring
Schulungsleiter

yannick.gehring@academyconsult.de

0174 9546837