



JAVACREAM

*Training
Consulting
Projectmanagement*

GIT

- Name
- Rolle im Unternehmen
- Themenbezogene Vorkenntnisse
- Aktuelle Problemstellung
- Konkrete individuelle Zielsetzung



Ausgangssituation

- Definition eines Standes eines Projekts bestehend aus Dateien
 - angereichert um Meta-Informationen: “Wer hat wann warum welche Änderungen gemacht?”
- Parallele Fortentwicklung von verschiedenen Ständen
- Konsistentes Zusammenführen (“Mergen”) von parallel entwickelten Ständen
- Zentrale Ablage der gesamten Informationen
 - Authentifizierung und Autorisierung
- Verfahren und Methoden zur Team-Zusammenarbeit
- Werkzeugunterstützung zum effizienten Arbeiten

- Definition eines Standes eines Projekts bestehend aus Dateien
 - angereichert um Meta-Informationen: “Wer hat wann warum welche Änderungen gemacht?”
- Parallele Fortentwicklung von verschiedenen Ständen
- Konsistentes Zusammenführen (“Mergen”) von parallel entwickelten Ständen
- Zentrale Ablage der gesamten Informationen
 - Authentifizierung und Autorisierung
- Verfahren und Methoden zur Team-Zusammenarbeit
- Werkzeugunterstützung zum effizienten Arbeiten

- Definition eines Standes eines Projekts bestehend aus Dateien
 - angereichert um Meta-Informationen: “Wer hat wann warum welche Änderungen gemacht?”
- Parallele Fortentwicklung von verschiedenen Ständen
- Konsistentes Zusammenführen (“Mergen”) von parallel entwickelten Ständen
- Zentrale Ablage der gesamten Informationen
 - Authentifizierung und Autorisierung
- Verfahren und Methoden zur Team-Zusammenarbeit
 - Git Flows mit Pull- bzw. Merge-Requests
- Werkzeugunterstützung zum effizienten Arbeiten
 - Web Frontend

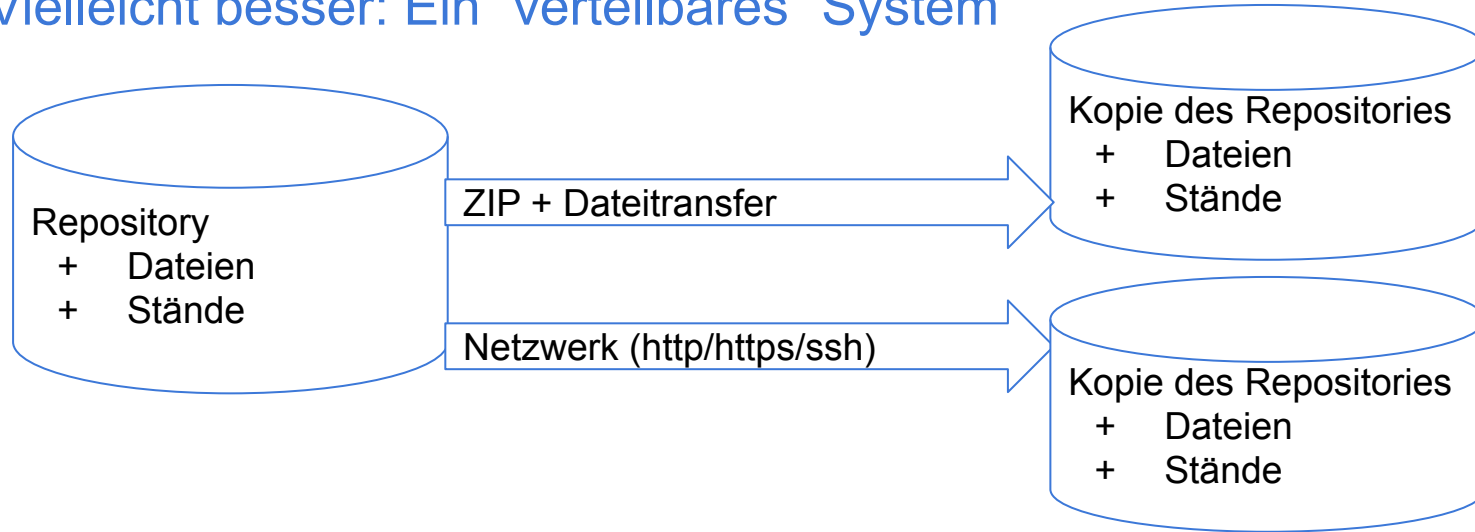
- BitBucket
 - Atlassian
- GitLab
 - GitLab.com
- **GitHub**
 - Microsoft

- Definition eines Standes eines Projekts bestehend aus Dateien
 - angereichert um Meta-Informationen: “Wer hat wann warum welche Änderungen gemacht?”
- Parallele Fortentwicklung von verschiedenen Ständen
- Konsistentes Zusammenführen (“Mergen”) von parallel entwickelten Ständen
- Zentrale Ablage der gesamten Informationen
 - Authentifizierung und Autorisierung
- Verfahren und Methoden zur Team-Zusammenarbeit
 - Git Flows mit Pull- bzw. Merge-Requests
- Werkzeugunterstützung zum effizienten Arbeiten
 - Web Frontend

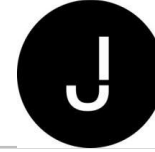
Im Seminar
+ Tag 1 + Tag 2
erste Session
+ Rest

Git ist ein “verteiltes Versionsverwaltungssystem”

- Vielleicht besser: Ein “verteilbares” System

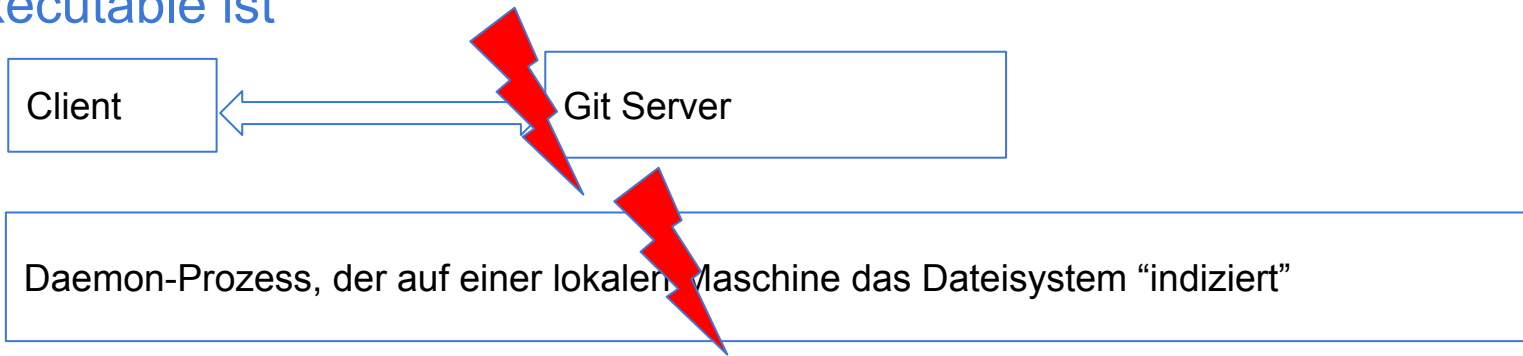


Unbedingt nötige Konsistenz = “Fälschungssicherheit” wird erreicht durch den Einsatz von Merkle-Trees (Jeder Stand bekommt einen Hashwert, und jeder Nachfolger enthält den Hashwert des Vorgängers) = Blockchain-Technologie




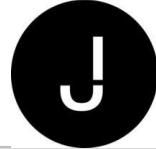
First Contact

- Git ist auf Ihren Maschinen installiert
 - `git --version` ist erfolgreich
- Die Git-Installation installiert ein `git-executable`
- `git-executable` ist



- Ein Kommando, das während der Ausführung eines Git-Kommandos die Funktionen eines Versionsverwaltungssystem bereitstellt

- 
- Einrichten eines Users auf dem Git Server
 - Lokal die Angabe der Server-URL
 - Lokale Konfiguration eines user.name und einer user.email
 - `git config --global user.name "Rainer Sawitzki"`
 - `git config --global user.email rainer.sawitzki@gmail.com`
 - `git config --help`
 - `git config --get user.name`



- Jetzt im Seminar total unüblich
 - Initialisieren eines neuen, lokalen Repositories mit `git init`
 - Didaktisch notwendig
- richtig wäre -> später
 - Repository wird auf GitHub eingerichtet
 - und auf die lokale Maschine gecloned



- Anlegen eines neuen Verzeichnisses

- mkdir training
- cd training

training ist ein ganz normales Verzeichnis

- Initialisieren des Repositories

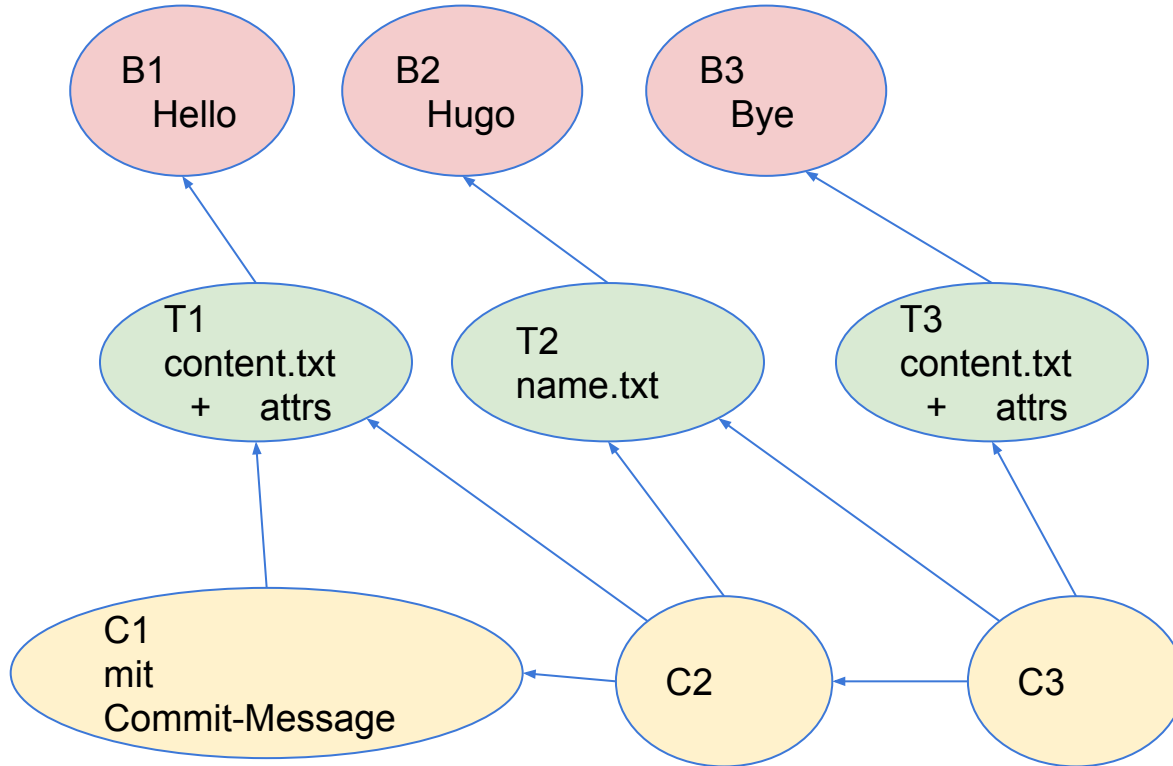
- git init
 - check: git status

training ist ein ganz normales Verzeichnis, aber nun genannt als Git Projekt-Verzeichnis

Das eigentliche Repository ist das Unterverzeichnis .git
Der Rest des Git-Projekts wird als Git-workspace bezeichnet

content.txt ist Bestandteil des Workspaces, aber dem Repository völlig unbekannt

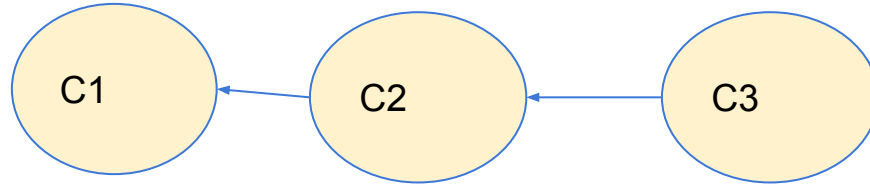
- Erstellen einer Datei
 - echo Hello > content.txt
 - check: git status mit einer “roten” Datei
- Bekanntmachen der Datei durch Hinzufügen zum Repository
 - genauer: Hinzufügen der Datei zur Staging-Area des Repositories
 - git add content.txt
 - check: git status mit einer “grünen” Datei
 - Hinweis: Das Hinzufügen zur Staging-Area ist keine Stand-Definition!
 - check: ls .git/objects/e9 mit der Datei `65047ad7c57865823c7d992b1d046ea66edf78`
- Definition des Standes mit git commit -m “commit message”
 - Vorsicht: Wenn Sie -m vergessen, öffnet sich ein Linux-Editor (vim, i -> Eingabemodus, ESC zurück zum Befehlsmodus, :wq zum schreiben und beenden)
 - git config --global core.editor <path_to_editor>
 - check: git status ist unauffällig, git log mit Ausgabe des Commits inklusive Commit-Hash



Content-Objects
oder
BLOBs

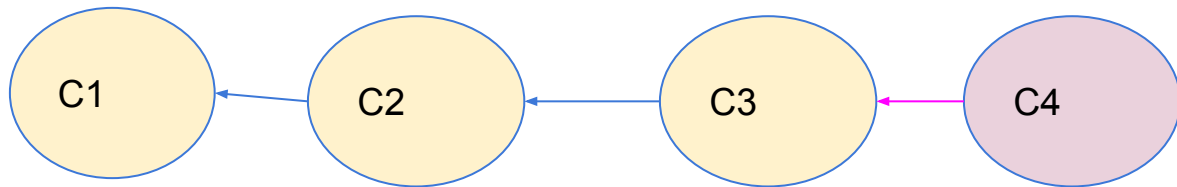
Tree-Objects

Commit-Objects



Commit-Objects

- Der alte Stand=Commit-Objekt + Staging-Area werden zu einem neuen Stand=Commit-Object zusammengeführt



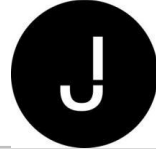
Bisher: `git log` -> Standard-Darstellung, relativ verbose

Jetzt: `git log --oneline --graph --all --decorate`

`git config --global alias.plog "log --oneline --graph --all --decorate"`



- `git checkout <hash>`
- Hinweise
 - Das Arbeiten mit dem hash-Wert ist natürlich sehr gewöhnungsbedürftig -> “Nerd-Modus”
 - In den meisten Fällen genügen bei der Angabe des Hash die ersten 7 Stellen
 - Dringende Empfehlung “Sawitzki”
 - checkout nur bei unauffälligem Status
 - Falls Status auffällig
 - `git add . + git commit -m`
 - `git add . + git stash (-> git.pdf bzw. Online-Dokumentation)`
 - Der Status nach dem checkout spricht von einem “detached HEAD” -> etwas später



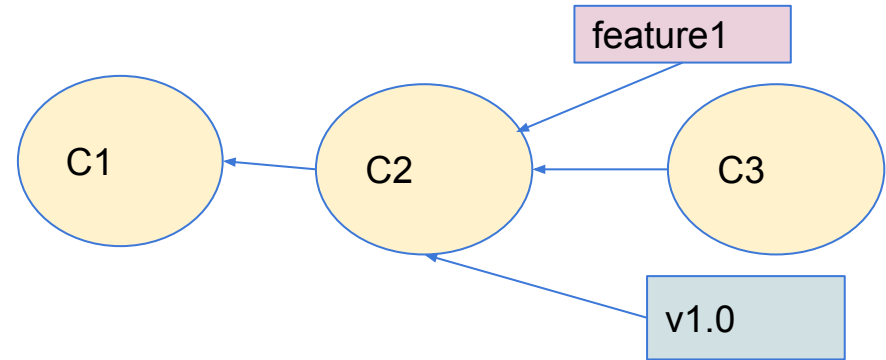
- `--global`
 - Für den angemeldeten Benutzer, `user.home .gitconfig`
- `--system`
 - für diese Git-Installation
- `--local`
 - gültig für das aktuelle Repository



Alias-Namen auf vorhandene Commit-Objekte

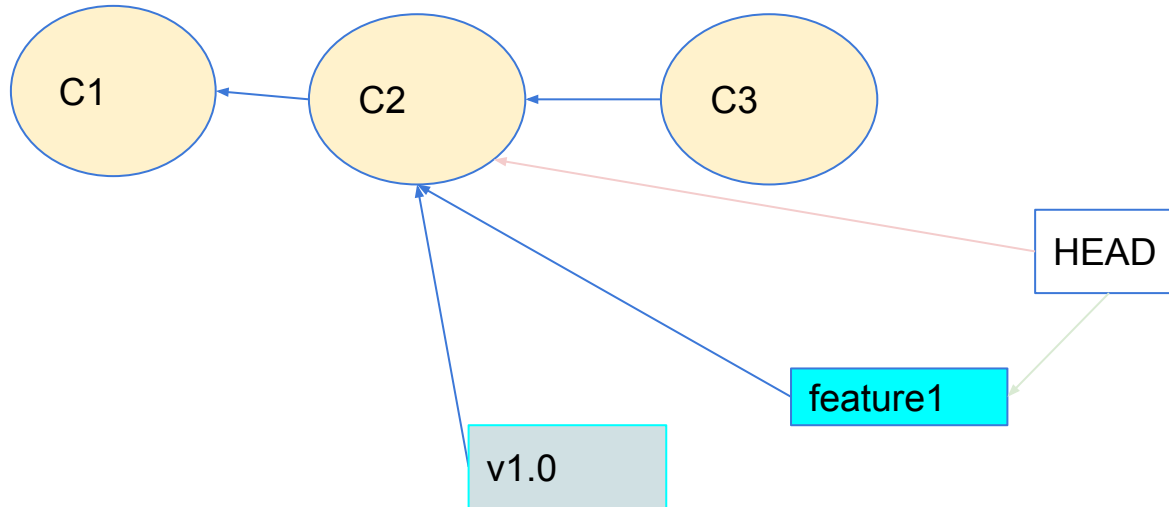
- Statt Nerd-Modus benutzen wir sprechende Namen
- 2 Einsatzbereiche im Kontext Versionsverwaltung
 - Definition eines fixen Standes
 - Versionsnummer, Release, Milestone
 - v1.0
 - Savepoint1
 - Heute Morgen
 - Definition einer aktuell fortschreitenden Entwicklung
 - implement/feature1
 - jira-issue-4711
 - experiment
 -

- Fixe Stände sind Tags
 - `git tag <tag_name>`
 - `git tag --list`
 - `git tag -d <tag_name>`
- Weiterentwicklung
 - `git branch <branch_name>`
 - `git branch --list`
 - `git branch -d <branch_name>`



```
git checkout C2
git tag v1.0
git branch feature1
```

HEAD, ein intern gepflegter
Alias-Name auf den aktuell
benutzten Stand



git checkout C2

Detached HEAD

git checkout v1.0

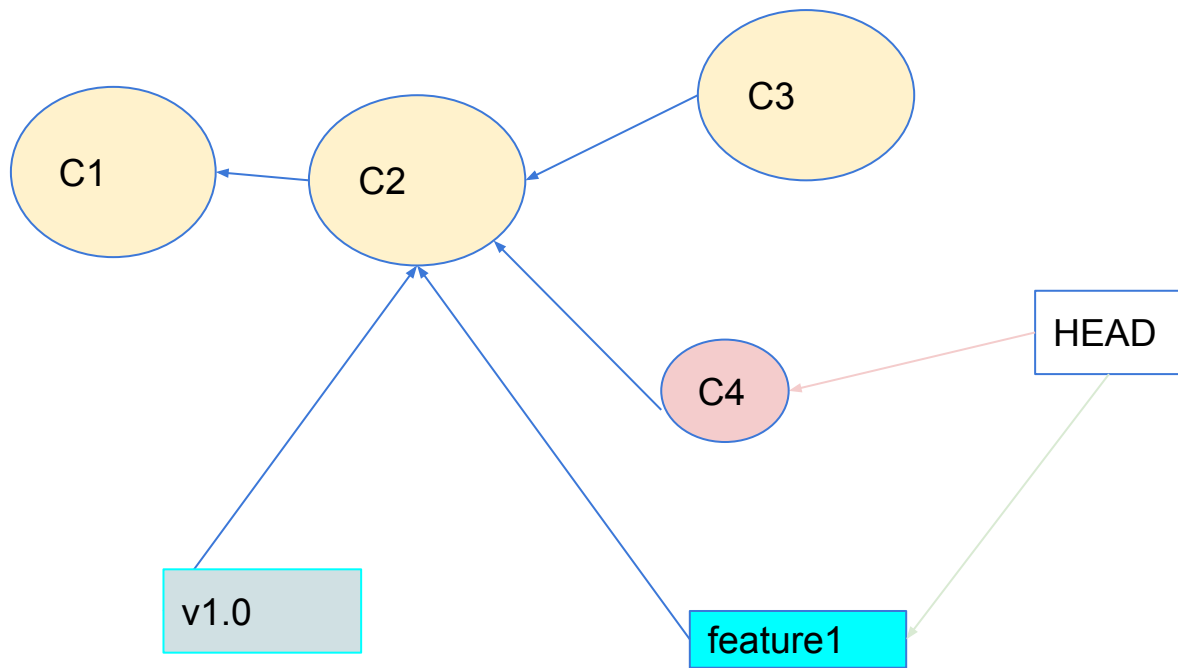
git checkout feature1

Attached HEAD

git commit revisited

Detached HEAD

Attached HEAD



```
git checkout C2  
echo ... > ...  
git add .  
git commit -m
```

```
git checkout feature1
```