



JAVACREAM

*Training
Consulting
Projectmanagement*

GIT

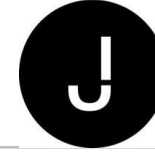
- Name
- Rolle im Unternehmen
- Themenbezogene Vorkenntnisse
- Aktuelle Problemstellung
- Konkrete individuelle Zielsetzung

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



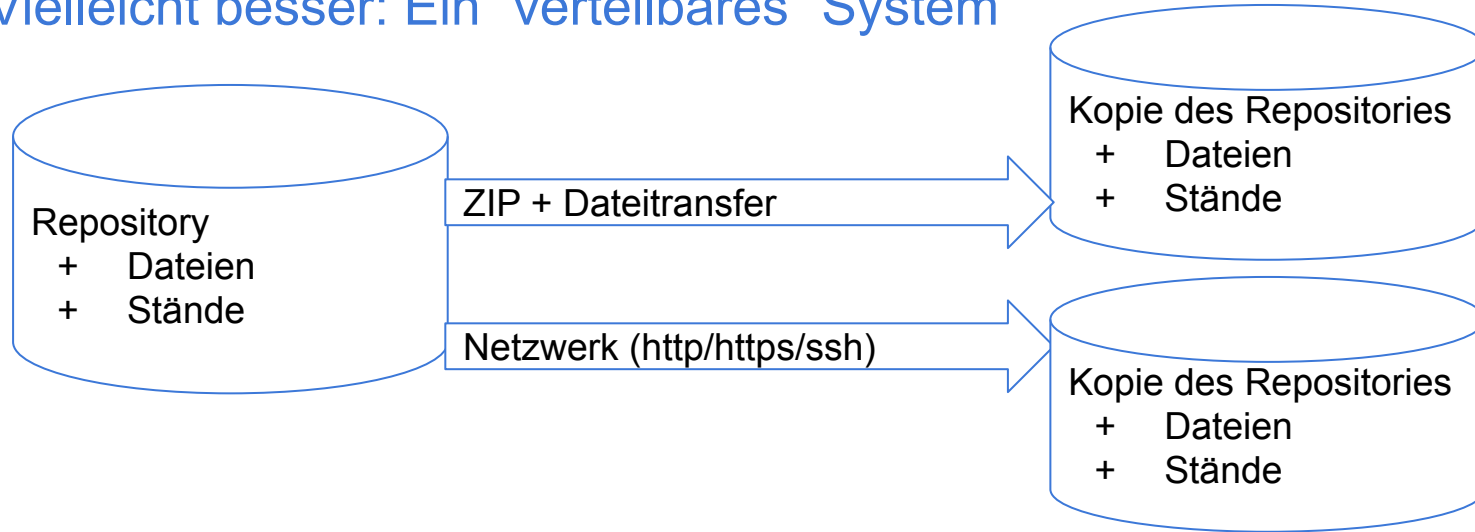
- BitBucket
 - Atlassian
- GitLab
 - GitLab.com
- **GitHub**
 - Microsoft

- 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

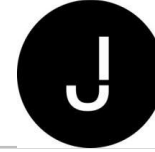
Im Seminar
+ Tag 1 + Tag 2
erste Session
+ Rest

Git ist ein “verteiltes Versionsverwaltungssystem”

- Vielleicht besser: Ein “verteilbares” System

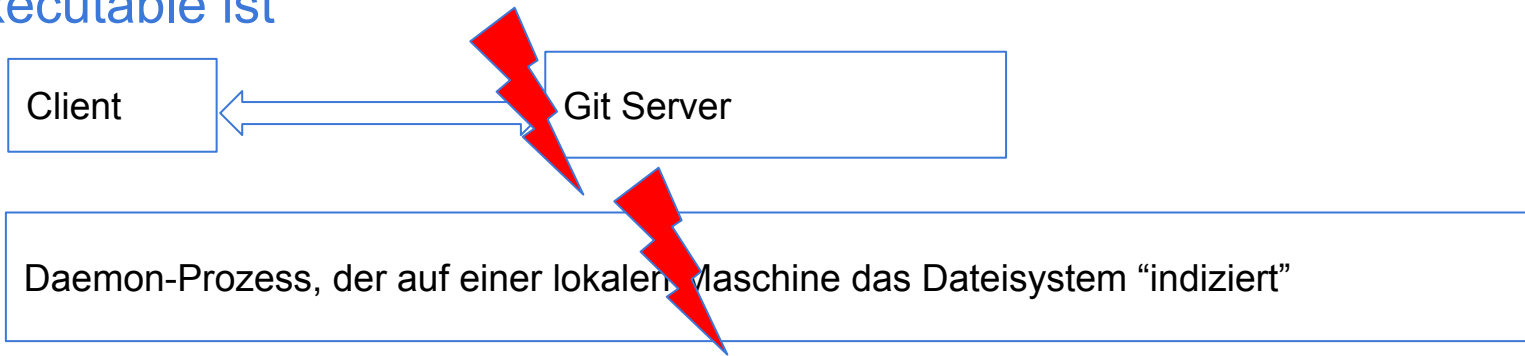


Unbedingt nötige Konsistenz = “Fälschungssicherheit” wird erreicht durch den Einsatz von Merkle-Trees (Jeder Stand bekommt einen Hashwert, und jeder Nachfolger enthält den Hashwert des Vorgängers) = Blockchain-Technologie




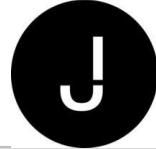
First Contact

- 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

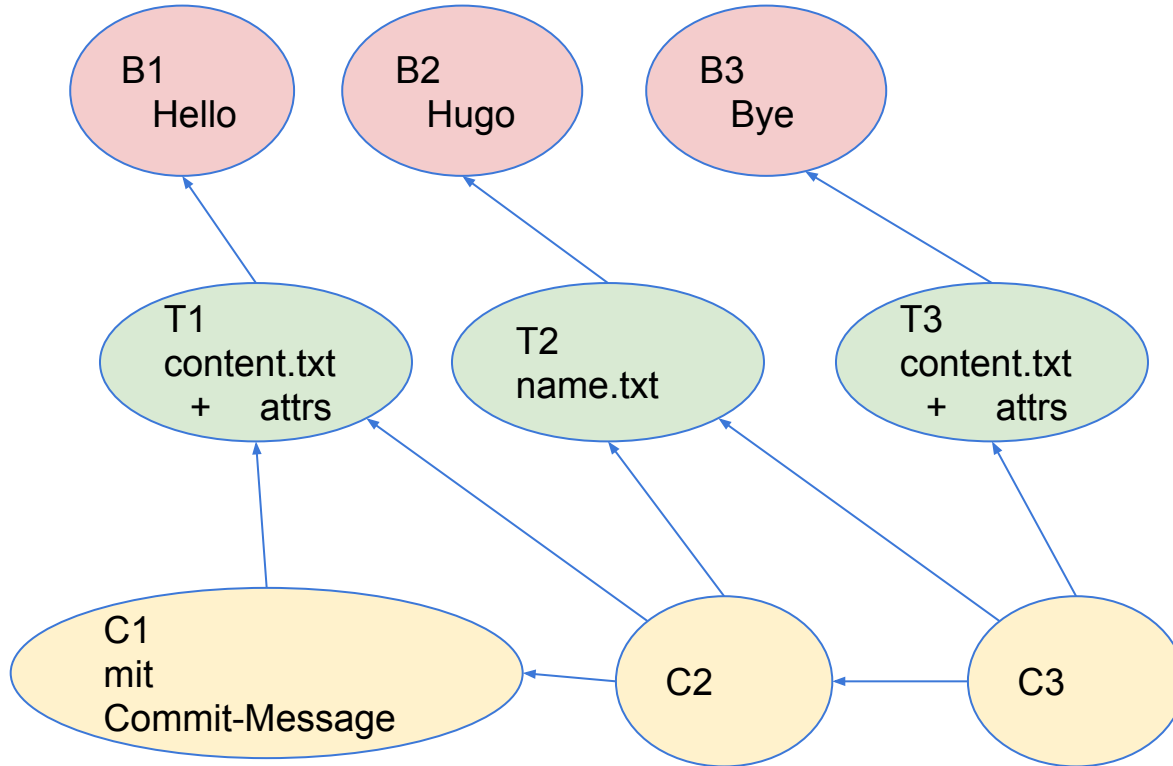
- Initialisieren des Repositories

- 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

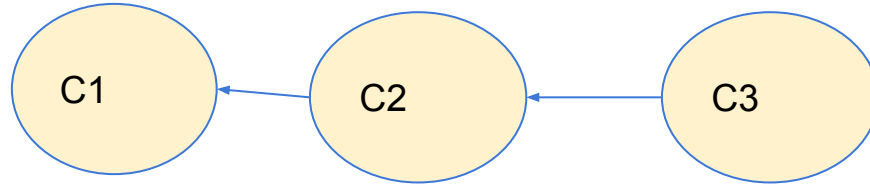
- Erstellen einer Datei
 - echo Hello > content.txt
 - check: git status mit einer “roten” Datei
- Bekanntmachen der Datei durch Hinzufügen zum Repository
 - genauer: Hinzufügen der Datei zur Staging-Area des Repositories
 - git add content.txt
 - check: git status mit einer “grünen” Datei
 - Hinweis: 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