PROGRAMMIERPROJEKT 2018

VERSIONSVERWALTUNGSSYSTEME

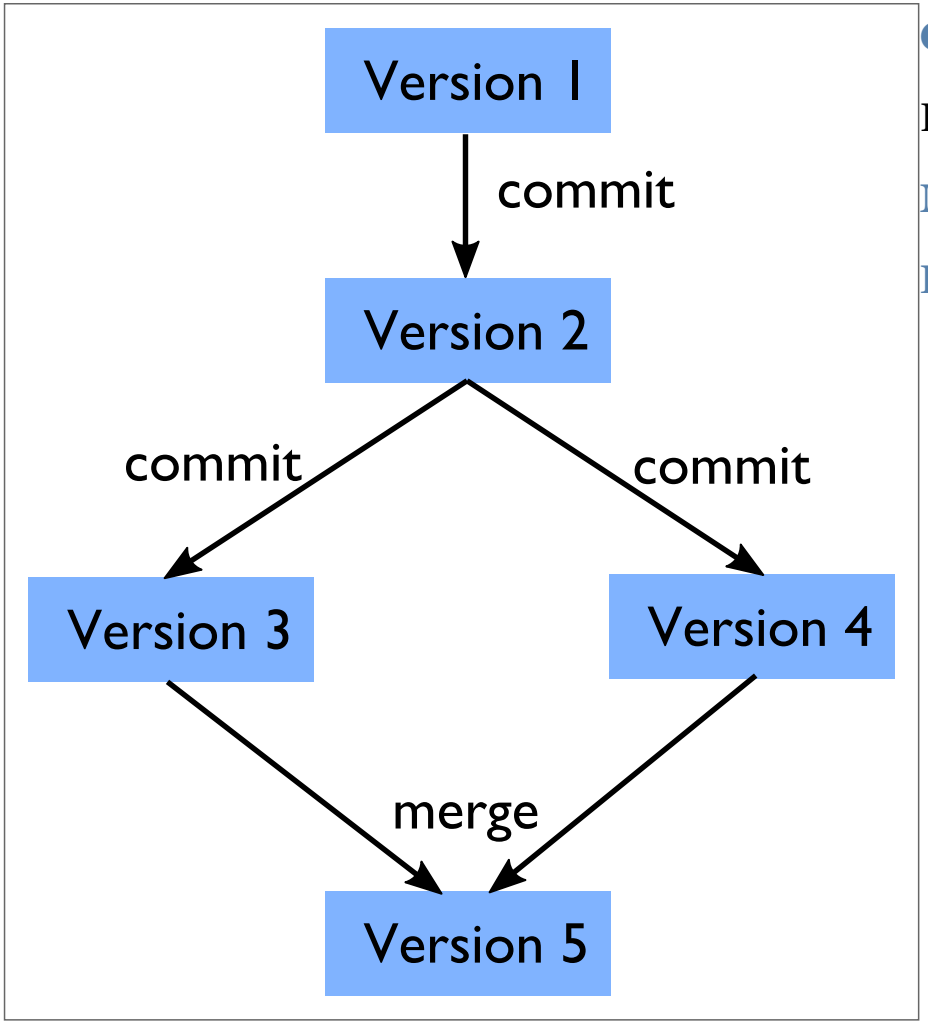
Dr. Annette Bieniusa, Mathias Weber

PROBLEMSTELLUNG

- Code und Änderungen müssen für alle Teammitglieder **zugänglich** sein
- Dateien müssen auf Server **gesichert** werden
- **Historie** der Änderungen sollte erhalten bleiben
- **Dokumentation**, welche Änderungen von welchem Teammitglied kommen

GIT

- Verteiltes Versionsverwaltungssystem (DVCS; distributed version control system)
- Versionen lokal verwaltet in Repository
- Mehrere Kopien des Repositories auf unterschiedlichen Rechnern

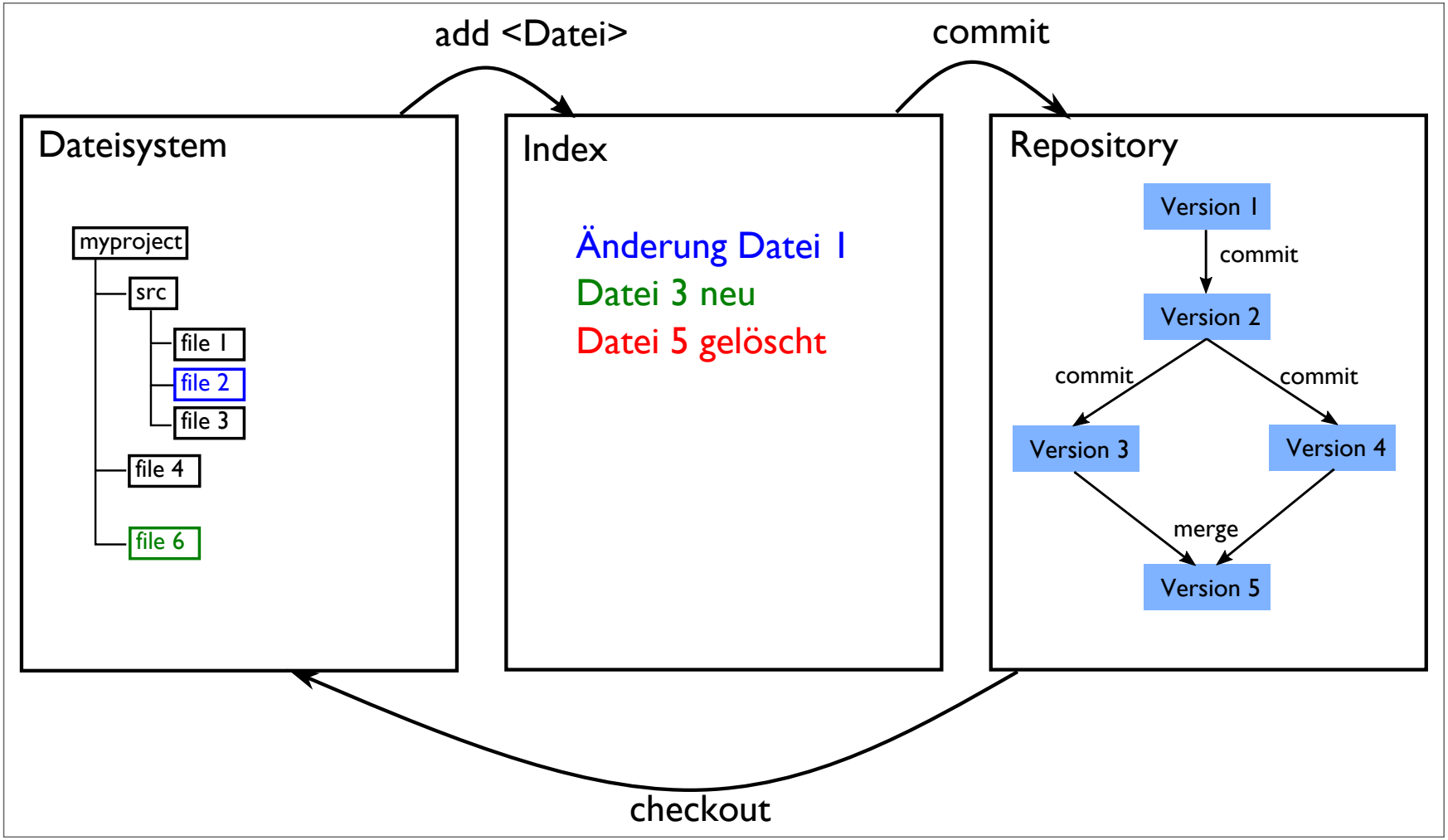


Commit von Änderungen erzeugt neue Version

Parallele Änderungen → mehrere Kindversionen

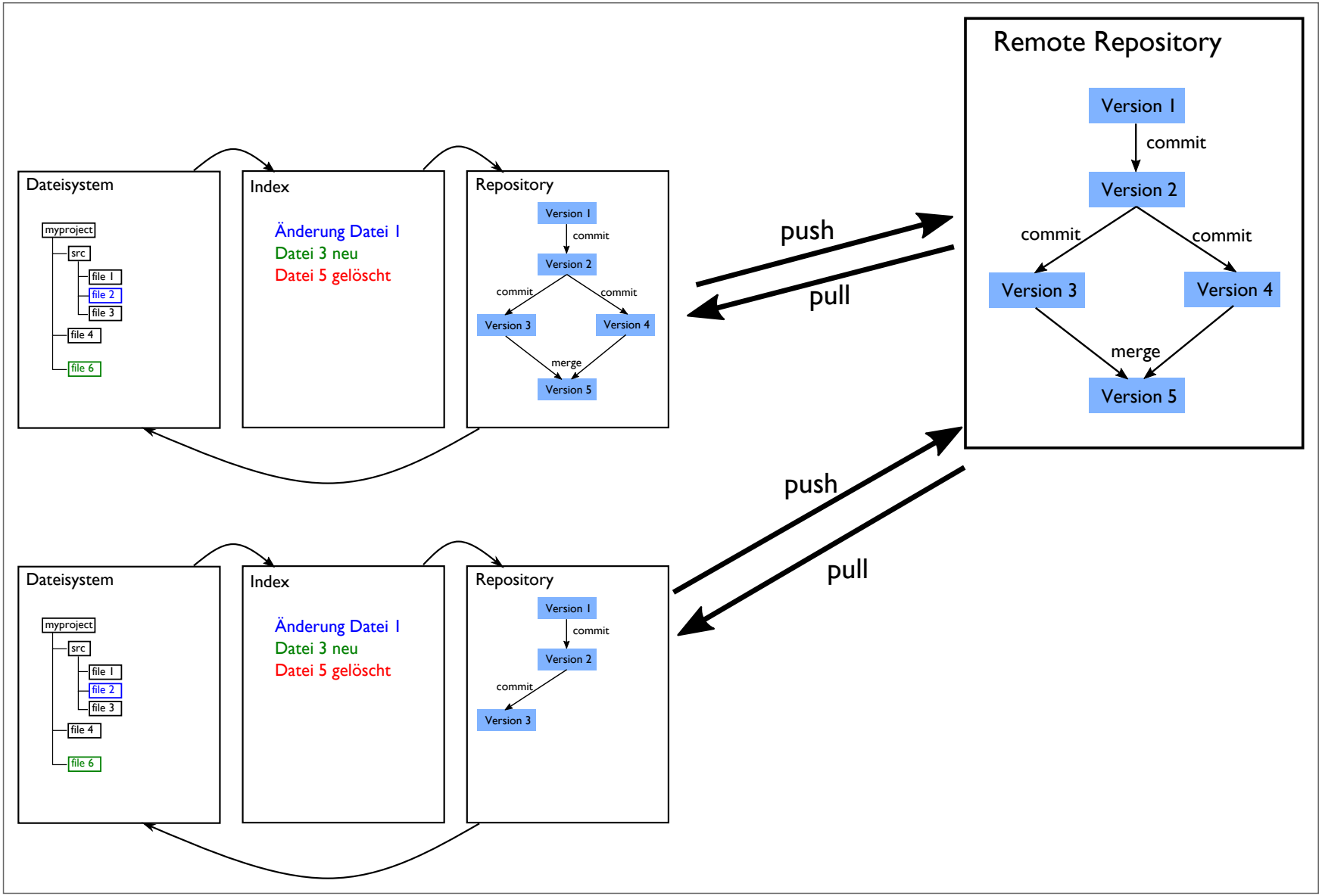
Merge führt zwei Versionen zusammen

Log zeigt Historie


add <Datei>
commit
Dateisystem
myproject
src
file 1
file 2
file 3
file 4
file 6
Index
Änderung Datei 1
Datei 3 neu
Datei 5 gelöscht
Repository
Version 1
commit
Version 2
commit
commit
Version 3
Version 4
merge
Version 5
checkout

LOKALE ÄNDERUNGEN

- Git verwaltet **Änderungen**, keine Dateien
- Änderungen werden lokal im Dateisystem gemacht
- Durch `git add` zum **Index** hinzugefügt (engl. stage)
- Befehl `git commit` übernimmt Änderungen im Index zu einer **Version**
- Befehl `git checkout` aktualisiert das Dateisystem auf Zustand einer Version

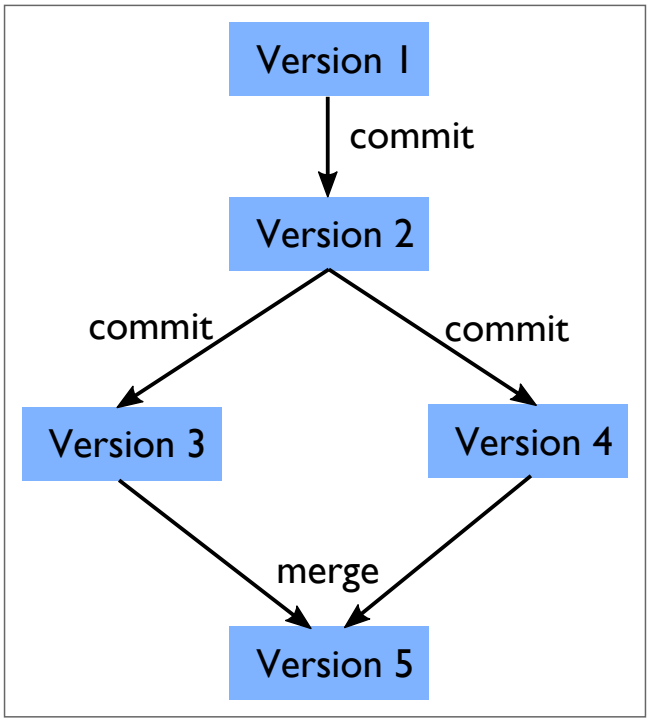

Remote Repository
Version 1
commit
Version 2
commit
commit
Version 3
Version 4
merge
Version 5
push
pull
Dateisystem
myproject
src
file 1
file 2
file 3
file 4
file 6
Index
Änderung Datei 1
Datei 3 neu
Datei 5 gelöscht
Repository
Version 1
commit
Version 2
commit
commit
Version 3
Version 4
merge
Version 5
push
pull
Dateisystem
myproject
src
file 1
file 2
file 3
file 4
file 6
Index
Änderung Datei 1
Datei 3 neu
Datei 5 gelöscht
Repository
Version 1
commit
Version 2
commit
Version 3

REMOTE REPOSITORIES

- Repository auf einem Server ermöglicht **Austausch von Änderungen**
- Jedes Teammitglied hat lokales Repository
- Mit `git push` werden **lokale** Versionen **auf Server** geschoben
- Mit `git pull` werden Versionen anderer Teammitglieder **vom Server geholt**
- Ein `git pull` macht gleichzeitig ein `checkout` (Aktualisierung des Dateisystems)

KONFLIKTE

- Versionen können Änderungen an **gleichen Stellen der gleichen Datei** enthalten
- Basieren beide Versionen auf der **selben Grundversion** ist die ein **Konflikt**



- Konflikte treten beim mergen auf
- Manuelle Lösung erforderlich
- Absprachen helfen, Konflikte zu vermeiden

WICHTIGE BEFEHLE

- `git init` - Neues Repository anlegen in aktuellem Ordner
- `git status` - Übersicht Zustand des Dateisystems/Index, Vergleich zu letzter Version
- `git diff` - Änderungen anzeigen
- `git add` - Änderungen in Index aufnehmen (stagen)
- `git commit` - Ändernungen im Index zu neuer Version
- `git checkout` - Dateisystem Zustand auf Zustand einer Version setzen
- `git push` - Lokale Versionen zum Server senden
- `git pull` - Versionen vom Server holen und Dateisystem aktualisieren

GITLAB

GitLab

https://softech-git.informatik.uni-kl.de/

- Verwaltung von Git Repositories
- Issue Tracking
- Wiki
- Code Snippets

DEMO