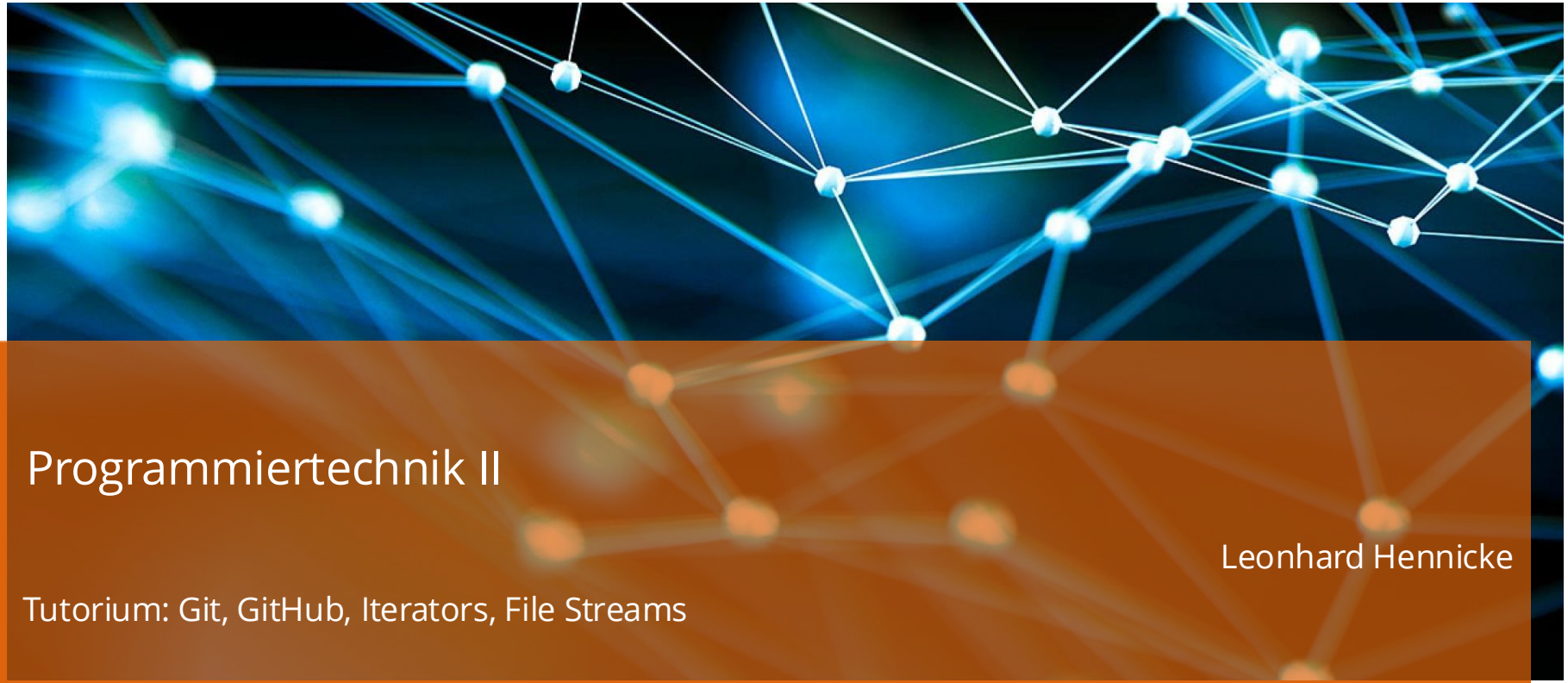


# Programmiertechnik II

Leonhard Hennicke

Tutorium: Git, GitHub, Iterators, File Streams

# Überblick

1. Git, GitHub, Classrooms
   - Install & Register
   - Basic Commands
   - Merging
   - `Pull Requests`
2. Iteratoren
3. File Streams

# Install & Register

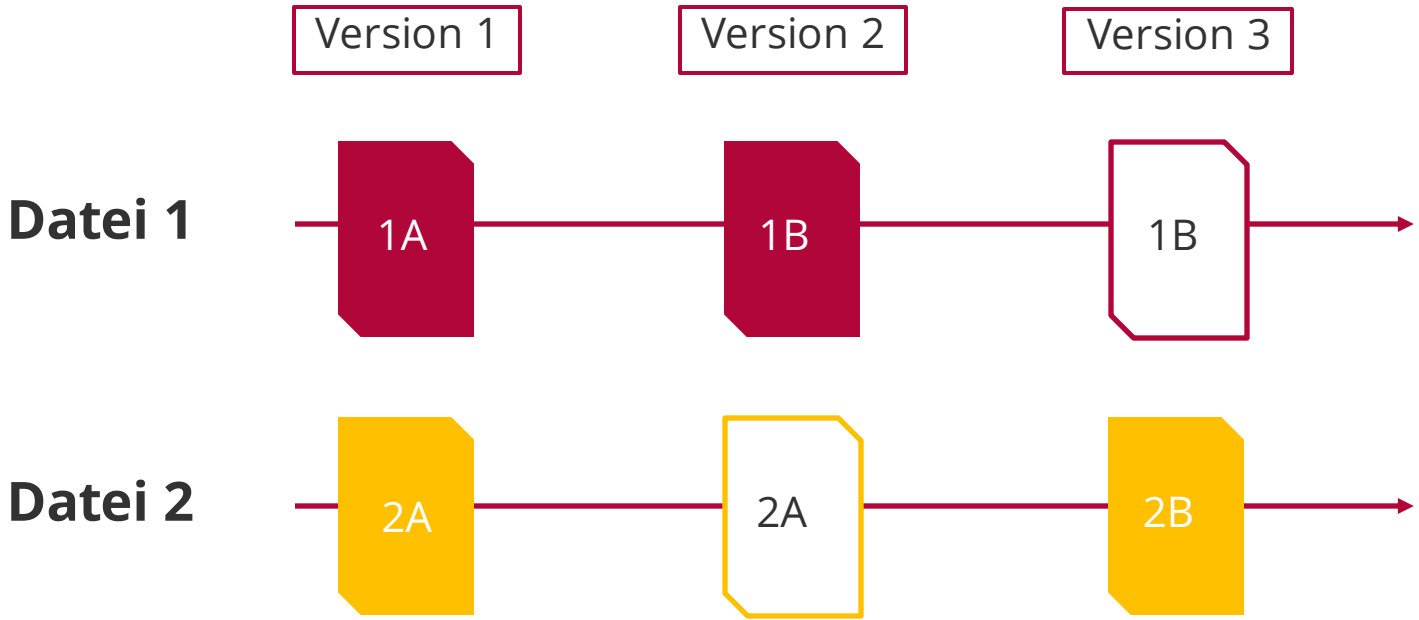
1. Git installieren
   - https://git-scm.com/downloads
2. GitHub Account erstellen
   - https://github.com
3. GitHub Classroom Assignment Link
   - https://classroom.github.com/a/6JGVYlbZ
4. Detailliertere Tutorials zu Git (optional)
   - https://www.atlassian.com/git

# Git vs. GitHub

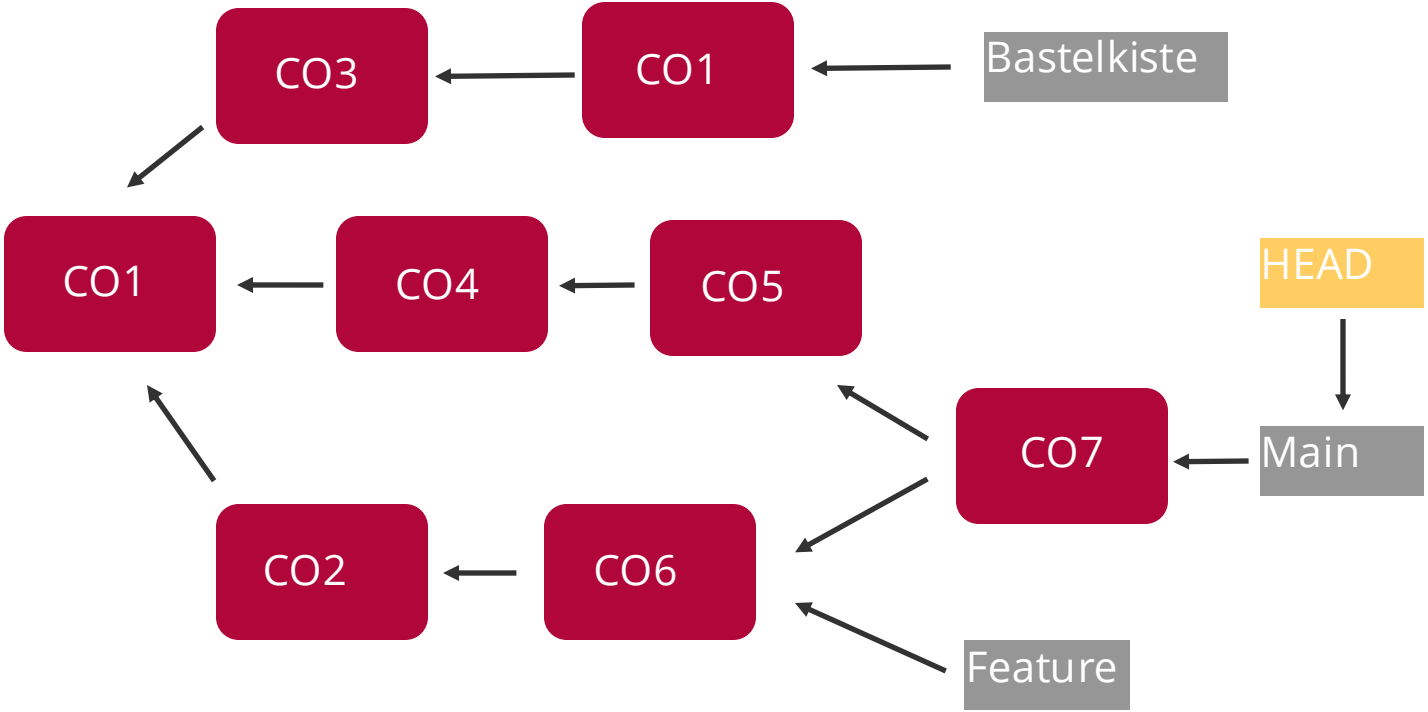
1. Git
   - System zur verteilten Versionsverwaltung
   - Open Source
   - Ursprünglich entwickelt von Linus Torvalds in 2005

2. GitHub
   - Proprietäre Softwareentwicklingsplatform
   - Betrieben von Microsoft
   - Basierend auf Git

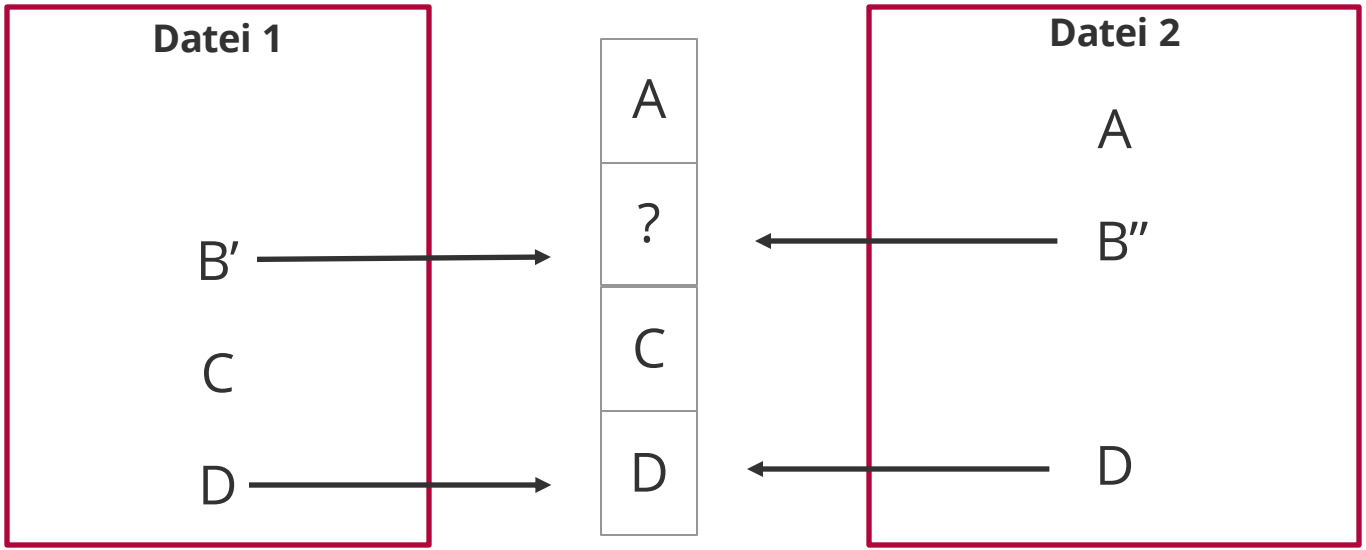
# Git



# Git: Branches



# Merge Konflikte



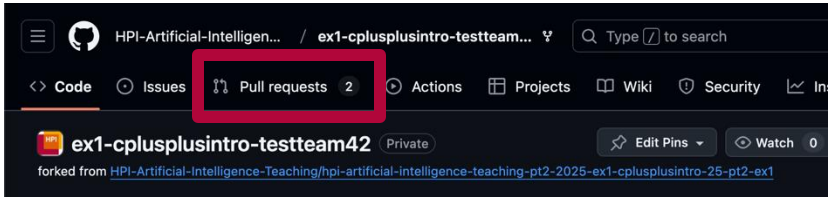

Original A B D

# Git: Befehle

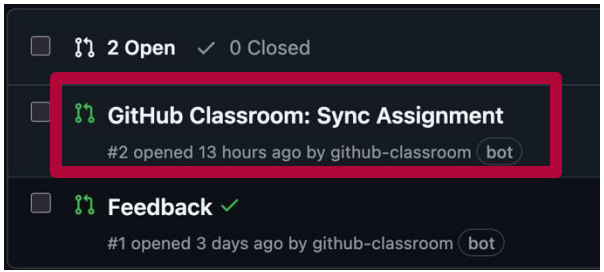
- git init
- git help, git help <befehl>
- git clone <repo link>
- git status, git log, git diff
- git pull
- git add <file-Pfad>, git add -A
- git commit -m “<commit message>”
- git push
- git checkout -b <new branch name>
- git checkout <branch name>
- git merge <branch name>

# Pull Requests im Übungsrepo

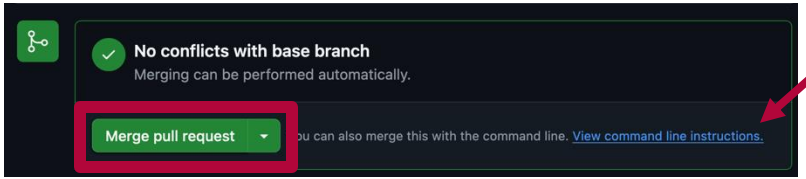
1.




2.



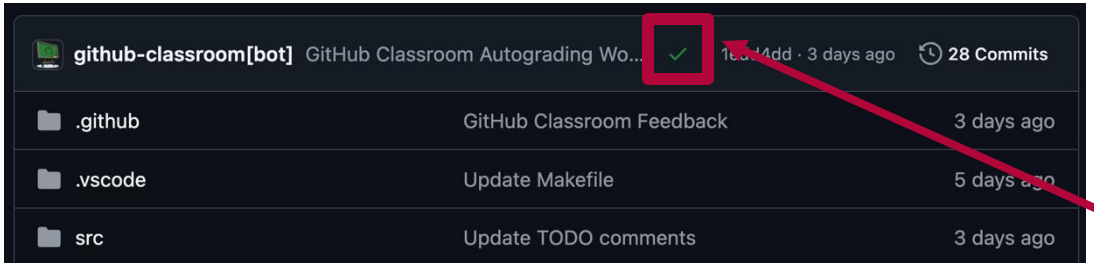

3.



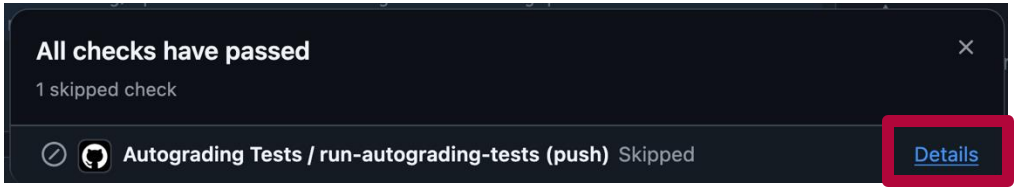

Lokal mergen ist auch eine Option

# GitHub Classrooms: Autograding

1.




2.




Automatisierte Tests via GitHub Classrooms nach jedem Push

# Iterator

- **Problem:** Selbst definiert Klasse soll einfach iterierbar sein
- **Lösung:** Eine zweite selbst definierte Klasse, die sich verhält wie Iteratoren anderer Container-Typen (z.B. vector::iterator), definiert Logik, wie iteriert werden soll
- **Vorteile:**
  - Range-basierte for-Loops: *for (auto it : MyContainer) {}*
  - STL Algorithmen für ander Container funktionieren auf der neuen Klasse
- **Minimales Interface:** Operatoren ++, *, ==, !=, Definition von value_type, sowie von begin() und end()

**array_iterator.h, array_iterator.cpp**

# File Stream (std::ifstream)

- Standard C++ class um Daten aus Dateien zu lesen
- Öffnet eine Datei und behandelt sie als Input Stream aus Characters/Bytes
- Binärdaten können mit .read() gelesen werden
- Um andere Datentypen zu lesen, sagen wir der .read() Method, dass die so tun soll als wären die Variablen, in die eingelesen wird char Pointer, ohne etwas an den eigentlichen Daten zu ändern
- Um Daten in Variable x einzulesen:
  - my_filestream.read(**reinterpret_cast<char*>(&x)**, sizeof(x));

**filestreamin.cpp**

Hasso Plattner Institut
Digital Engineering · Universität Potsdam

# Viel Spaß bis zur nächsten Vorlesung!