Versionsverwaltung mit Git
Befehlsreferenz

Fachschaft Informatik TU Darmstadt

Fachschaft Informatik

TECHNISCHE UNIVERSITÄT DARMSTADT

Stand: 28. Oktober 2020

Alle Befehle müssen in einem Git-Repository ausgeführt werden!
`<name>` stellt einen Platzhalter dar. z. B. `git add <datei>` → `git add datei.txt`

Einrichtung

- Setze *Author*-Informationen:
  ```
  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"
  ```
- Setze einen alternativen anfängerfreundlichen Editor:
  - Linux: `git config --global core.editor nano`
  - macOS: `git config --global core.editor textedit`
  - Windows: `git config --global core.editor notepad`

Begriffe

- **Repository** ein Git-Verzeichnis
- **Commit** ein Paket abgeschlossener Änderungen in der Historie
- **ref** z. B. ein *Commit*

Allgemein

- `git init`: Anlegen eines Git-Repositories. **Nicht** erforderlich, wenn man ein Repository klont (siehe Zusammenarbeit).
- `git status`: Zeigt an, welche Änderungen im Repository für den Commit vorgemerkt sind.
- `git add <datei>`: Fügt Änderungen zu einem Commit hinzu.
- `git rm <datei>`: Löscht eine Datei sowohl aus dem Verzeichnis als auch vom Repository.
- `git commit`: Schreibt Änderungen in die Historie.
- `git commit --amend`: Schreibt den letzten Commit um (Beschreibung, Änderungen, etc.).
- `git log`: Zeigt die Historie/Commits an.

- `git show <ref>`: Zeigt Informationen zu einer bestimmten ref an (inkl. Änderungen).
- `git checkout <ref>`: Setzt das Git-Repository auf eine bestimmte ref.
- `git diff <ref>..<ref>`: Zeigt die Änderungen (pro Datei) von einer ref zu einer anderen ref an.
- `git revert <commit>`: Macht einen Commit rückgängig (erstellt einen neuen Commit, welcher den alten invertiert).

Zusammenarbeit

- `git clone <url>`: Klont das Git-Repository von einem Server
- `git push`: Lädt Commits auf den Server hoch
- `git fetch --all`: Lädt Änderungen aller Branches vom Server herunter
- `git pull`: Lädt Änderungen vom Server herunter und merged ggf. automatisch oder produziert Merge-Konflikte
- `git branch`: Listet alle Branches des Git-Repos auf
- `git branch <branch>`: Erstellt Branch branch; Parameter -d löscht ihn (-D auch wenn noch nicht gemergt)
- `git checkout <branch>`: Wechselt zum Branch branch; Parameter -b erstellt den Branch dabei
- `git merge <branch>`: Merged Branch branch in aktuellen Branch

.gitignore

Wenn man manche Dateien nicht versionieren möchte, kann man diese auf eine Ignorierliste setzen. Dafür wir eine `.gitignore`-Datei verwendet.
Achtung: Diese Datei hat keinen Dateinamen, nur eine Dateiendung. Unter Windows ist die Erstellung solch einer Datei im Windows-Explorer nicht möglich. Dafür muss die Datei direkt aus Notepad gespeichert werden, wähle dabei **keine** Dateiendung aus der Liste aus.
Die Datei enthält die Namen von zu ignorierenden Dateien, dabei können auch Verzeichnisnamen oder „Wildcards“ verwendet werden.

Beispielhafter Inhalt:

```
# kompilierte Java-Dateien ausschliessen
*.class

# Export-Daten nicht synchronisieren
exportiert/
# Datei, die doch synchronisiert werden soll
!exportiert/manuellErstellteDatei.tpt

# eine bestimmte ausgeschlossene Datei
zeugs/nichtFuerAndereRelevantes.tpt
```