

Git for gits

A. C. Hinrichs
April 30, 2019



Figure 1: Anleitung für diese Anleitung¹

Basis-Workflow

中文 中文 中文 中文 Ein generischer Git-workflow, ein **Repository** muss logischerweise nur ein mal angelegt werden,

Ein Repository klonen

```
$ git clone <REPO-URL>
```

Git erstellt nun eine **Arbeitskopie** des entfernten Repos. Diese liegt anschließend in einem Unterorder des aktiven Verzeichnisses. Existiert kein entferntes Repository kann man das aktive Verzeichniss mittels

```
$ git init
```

als Repository initialisieren.

Einen Branch erstellen & auf ihn wechseln

Die Entwicklung auf zwei verschiedenen Branches verläuft komplett unabhängig voneinander

```
$ git checkout -b <NEW-BRANCHNAME>
```

Wenn der Branch bereits existiert, kann die Option `-b` weggelassen werden. Alle (lokalen) Branches kann man sich mit dem Befehl

```
$ git branch --list
```

anzeigen lassen.

Im Repository Arbeiten

Wie auf einer nicht Versionierter Code-Base. Ist das Arbeitsverzeichnis im Vergleich zum letzten Commit verändert, so sagt man dass es **dirty** also Schmutzig ist.

Änderungen Comitten

Die geänderten Dateien kann man sich mit dem Befehl

```
$ git status
```

ausgeben lassen.

Um seine Änderungen zum **Index** hinzuzufügen (also dafür zu sorgen, dass Git sie sich merkt) nutzt man folgendes Kommando

```
$ git add <FILENAME>
```

Dies funktioniert auch für Verzeichnisse. Gibt man als Dateinamen `.` an, so legt Git alle Änderungen auf den Index.

Man erstellt nun einen **Commit** mit dem Befehl

```
$ git commit -m <COMMIT-NACHRICHT>
```

Pushen und Pullen

Um Änderungen aus dem entfernten Repository zu laden:

```
$ git pull
```

Um seine Commits auf das entfernte Repo zu laden:

```
$ git push
```

Mergen

Git ist sehr gut darin, Änderungen zusammenzuführen, benötigt aber manchmal dabei Hilfe (Git wird einen darauf hinweise). Diese **Merge Konflikte** lassen sich mittels dem tool

```
$ git mergetool
```

zusammenführen. Ich empfehle immer die Option `--tool=emerge` anzugeben².

Branche zusammenführen

Auch verschiedene Branches werden "gemerged":

```
$ git checkout <ZIEL BRANCH>
$ git merge <QUELL BRANCH>
```

Merged den Branch `<QUELL BRANCH>` in den Branch `<ZIEL BRANCH>`

¹Quelle: <https://xkcd.com/1597/>

²um seinen Merge mit dem besten Editor durchzuführen