Matthias Mruzek-Vering, M.Sc.

Hochschule Karlsruhe
University of
Applied Sciences

Git Cheat Sheet

Git ist ein verteiltes Versionskontrollsystem. Dieser Spickzettel enthält die wichtigsten und am häufigsten verwendeten Git-Befehle zum einfachen Nachschlagen.

Setup

Die ersten beiden Zeilen setzten alle nötigen Informationen über euch Global auf eurem System. Damit könnt ihr alle eure Repositories problemlos erreichen. Ebenfalls kann (wie Beispielsweise Zeile 3) allgemeinere UI Einstellungen vorgenommen werden, hier eine Farbige Historie.

```
$ git config --global user.name „ThisIsMyUserName“
$ git config --global user.email „ThisIsMyEmail@email.com"
$ git config --global color.ui auto
```

Init

Initialisiere und Klone eine Repository

```
$ git init
$ git clone https://github.com/hka-mmv/dscb230-tutorial
```

Update

Zeigte alle veränderten Dateien an.

```
$ git status
```

Löscht eine Datei aus dem Git Repository

```
$ git rm file-1.py
```

```
$ git rm file-1.py file-2.py
```

Füge eine oder mehrere Dateien zum Commit hinzu.

```
$ git add file-1.txt
```

```
$ git add file-1.txt file-2.py file-3.json
```

Tim Ehmann
Fabian Juny
Raphaele Salvatore Licciardo, B.Sc.

Zeige alle Änderungen einer Datei

```
$ git diff file-1.txt
```

Füge alle Unversionierte Dateien einem Commit hinzu.

```
$ git add .
```

Benenne den erstellten Commit.

```
$ git commit -m „This is my Message“
```

Branch & Merge

Zeige alle Entwicklungszweige / Branches die auf der lokalen Maschine sind

```
$ git branch
```

Zeige alle Entwicklungszweige / Branches die es in dem Git Repo. gibt.

```
$ git branch -a
```

Erstelle einen neuen Entwicklungszweig

```
$ git branch Feature-1
```

Wechsel den Branch

```
$ git checkout Feature-1
```

Erstelle einen Branch und wechsel in diesen

```
$ git checkout -b Feature-2
```

Merge einen Branch dem aktuellen Branch hinzu

```
$ git merge main
```

Lösche einen Branch

```
$ git branch -d -f Feature-1
$ git push origin --delete Feature-1
```

Historie umschreiben

Schiebe einen Branch / Commits zu einem neuen Ursprung.

```
$ git rebase Branch-1
```

Macht Änderungen rückgängig.

```
$ git reset --hard [commit-message]
```

Vergleichen

Zeige alle Commits in dem aktuellen Branch

```
$ git log
```

```
$ git log --graph
```

```
$ git log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset
%s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit
```

Zeige alle Änderungen einer Datei, auch wenn diese unbenannt wurde.

```
$ git log --follow file-1.py
```

Zeige den aktuellen Entwicklungsstand in Farbe mit Branches

```
$ git log --graph
```

Zeige alle Commits von Branch-1 die nicht in Branch-2 sind

```
$ git log Branch-1 Branch-2
```

Ignorieren & Entfernen

Dateien können auch ignoriert werden. Erstelle eine .gitignore Datei und füge deine Regulären Ausdrücken ein. Zum Beispiel: Ignoriere den logs Ordner, die note Markdown Dateien und jegliche swp Dateien.

```
$ touch .gitignore
$ echo „logs/„ >> .gitignore
$ echo „note_*.md„ >> .gitignore
$ echo „.*.swp„ >> .gitignore
```

Remote

Erhalte alle Änderungen aus einem Remote Repository

```
$ git pull
```

```
$ git pull <remote>
```

Hole alle Commits, Dateien und Verweise aus einem Remote Repository in das lokale Repository.

```
$ git fetch <remote> <branch>
```

Pushe alle Commits in Git.

```
$ git push
```

```
$ git push <remote> <branch>
```