

# Git

Git ist eine freie Software zur verteilten Versionsverwaltung von Dateien.

- [Git Cheat-Sheet](#)

# Git Cheat-Sheet

## Terminologie

|                 |   |
|-----------------|---|
| repo/repository | hier wird der Code und die vergangenen Versionen gespeichert                        |
| clone           | eine lokale Kopie eines Repositories erzeugen, in der gearbeitet werden kann        |
| fork            | (GitHub) ein fremdes Repository in den eigenen Account kopieren                     |
| commit          | den aktuellen Zustand deiner Arbeit zu einem bestimmten Zeitpunkt festhalten        |
| branch          | ein anderer Entwicklungsweig. Ein branch ist ein Zeiger auf einen bestimmten commit |
| master          | der Standardentwicklungsweig  |
| origin          | (Github) das Repository, aus dem dein Fork stammt                                   |
| HEAD            | der aktuelle Entwicklungsweig   |
| remote          | ein Repository auf einem anderen Computer   |

## Ein Git-Repository erzeugen

|  |  |
|--|--|
| git init   | ein neues Repository starten   |
| git clone <repo url>   | ein vorhandenes Repository auf deine Workstation klonen (das Verzeichnis hat den Namen des Repos)  |
| git clone <repo url> <folder name>                               | ein Repository in ein bestimmtes Verzeichnis klonen  |
| git clone <repo url> .   | ein Repository in das aktuelle Verzeichnis klonen (sollte leer sein)   |
| git remote add origin http://github.com/username/<repo name>.git | Create a remote repo named origin pointing at your Github repo (after you've already created the repo on Github) (used if you git init since the repo you created locally isn't linked to a remote repo yet) |
| git remote add origin git@github.com:username/<repo name>.git    | Create a remote repo named origin pointing at your Github repo (using SSH url instead of HTTP url)   |
| git remote   | zeige die verbundenen entfernten Repos   |
| git remote -v  | zeige zusätzlich die URLs dieser Repos   |
| git remote set-url origin <neue URL>                             | eine neue URL für ein entferntes Repository setzen   |
| git remote rm <remote name>                                      | ein verbundenes Repo entfernen   |

## Autoreninformationen setzen

|                                |  |
|--------------------------------|--|
| git config --global user.name  | globale Einstellung anzeigen (aus ~/.gitconfig)                    |
| git config --global user.email | globale Einstellung anzeigen (aus ~/.gitconfig)                    |
| git config --local user.name   | lokale Einstellung anzeigen (gilt nur für das aktuelle Repository) |

|                               |  |
|-------------------------------|--|
| git config --local user.email | lokale Einstellung anzeigen (gilt nur für das aktuelle Repository) |
|-------------------------------|--|

## Änderungen anzeigen/rückgängig machen

|                                     |  |
|-------------------------------------|--|
| git status                          | zeige die geänderten Dateien   |
| git diff                            | zeige die Änderungen seit dem letzten Commit                           |
| git diff <filename>                 | zeige die Änderungen an einer bestimmten Datei seit dem letzten Commit |
| git diff <commit id > <commit id 2> | Zeige die Änderungen zwischen zwei Commits                             |
| git log                             | zeige die vergangen Änderungen, Commitinfos und -IDs                   |
| git blame <filename>                | zeige wer welche Zeilen einer bestimmten Datei geändert hat und wann   |
| git reset --hard                    | stelle zurück auf den Stand des letzten Commits                        |
| git revert HEAD                     | letzten Commit rückgängig machen und einen neuen Commit erzeugen       |
| git revert <commit id>              | einen bestimmten Commit rückgängig machen und einen neuen erzeugen     |

## Staging / Verladezone

|                           |  |
|---------------------------|--|
| git add -A                | stellt alle Dateien (neu, geändert, gelöscht) in die Stagingarea   |
| git add .                 | neue und geänderte Dateien in die Stagingarea (keine Gelöschten)   |
| git add -u                | geänderte und gelöschte Files in die Stagingarea (keine Neuen)   |
| git rm <filename>         | eine Datei löschen und aus dem Repository entfernen  |
| git rm <filename> -cached | eine Datei aus dem Repository nehmen, die Datei wird nicht gelöscht. Üblicherweise wird diese Datei nun in .gitignore eingefügt. |

## Veröffentlichen

|                            |   |
|----------------------------|---|
| git commit -m „message“    | Änderungen am Stagingarea festschreiben (committen?)  |
| git commit -am „message“   | Geänderte und gelöschte Dateien (nicht Neue) stagen und gleich committen.   |
| git stash                  | den aktuellen Stand an nicht festgeschriebenen (committeten) Änderungen (also geänderte Dateien und Änderungen, die bereits in der Stagingarea warten) speichern. |
| git stash list             | Liste der vorhandenen Zwischenspeicher anzeigen   |
| git stash apply            | Reapply the latest stashed contents   |
| git stash apply <stash id> | Reapply a specific stash. (stash id = stash@{2})  |
| git stash drop <stash id>  | Einen bestimmten Stash löschen  |
| git push                   | Push your changes to the origin   |

|                                     |  |
|-------------------------------------|--|
| git push origin <local branch name> | Push a branch to the origin                          |
| git tag <tag name>                  | Tag a version (ie v1.0). Useful for Github releases. |

## Code holen und aktualisieren

|  |   |
|--|---|
| git fetch  | Get the latest changes from origin (don't merge)  |
| git pull   | Get the latest changes from origin AND merge  |
| git checkout -b <new branch name> origin/<branch name> | Get a remote branch from origin into a local branch (naming the branch and switching to it) |

## Branching

|  |   |
|--|---|
| git branch   | Show all branches (local)   |
| git branch -a  | Show all branches (local and remote)  |
| git branch <branch name>                               | Create a branch from HEAD   |
| git checkout -b <branch name>                          | Create a new branch and switch to it  |
| git checkout <branch name>                             | Switch to an already created branch   |
| git push origin <branch name>                          | Push a branch up to the origin (Github)   |
| git checkout -b <new branch name> origin/<branch name> | Get a remote branch from origin into a local branch (naming the branch and switching to it) |
| git push origin -delete <branch name>                  | Delete a branch locally and remotely  |

## Branching (Fortgeschritten)

|                             |  |
|-----------------------------|--|
| git checkout master         |  |
| git merge <branch name>     | Merge a specific branch into the master branch.  |
| git rebase <branch name>    | Take all the changes in one branch and replay them on another. Usually used in a feature branch. Rebase the master to the feature branch so you are testing your feature on the latest main code base. Then merge to the master. |
| git cherry-pick <commit id> | Merge just one specific commit from another branch to your current branch.   |

## Tipps und Tricks

### Letzte, noch nicht commitete, Änderungen verwerfen

```
git stash save --keep-index --include-untracked
git stash drop
```

### Sensible Informationen aus dem Git-Repository entfernen

In diesem Beispiel soll der Ordner python/python3 vollständig, inklusive der History, aus dem Repository entfernt werden:

```
git filter-branch --tree-filter 'rm -rf python/python3' --prune-empty HEAD
git for-each-ref --format="% (refname)" refs/original/ | xargs -n 1 git update-ref -d
echo python/python3/ >> .gitignore
git add .gitignore
git commit -m 'Removing python/python3 from git history'
git gc
git push origin master --force
```

## Author und Email-Adresse in Git-History ändern

Mit einem `git push -force` funktioniert das auch mit bereits gepushten commits. In Gitlab muss für `-force` zuerst der Schutz der Branch aufgehoben werden (Repository Settings ? Protected Branches).

```
git filter-branch --env-filter 'if [ "$GIT_AUTHOR_EMAIL" = "falsche@adresse.de ]; then
GIT_AUTHOR_EMAIL=info@ovtec.it;
GIT_AUTHOR_NAME="Oliver Völker";
GIT_COMMITTER_EMAIL=$GIT_AUTHOR_EMAIL;
GIT_COMMITTER_NAME="$GIT_AUTHOR_NAME"; fi' -- --all
```

Danach noch `git push -force`. Jaja, unschön, ich weiß.