ERSTELLEN

Titel

```
$ git Kommando
```

Nähere Details (optional)

Repo erstellen

```
$ git init
```

Neues lokales Git Repository erstellen

Repo klonen

```
$ git clone *URL*
```

Ein vorhandenes Git Repository klonen (URL ist im TFS unter "Code > Clone" zu finden)

LOKALE ÄNDERUNGEN

```
$ git status
```

Alle lokalen Änderungen im aktuellen Repository anzeigen (wie *Pending Changes* im TFVC)

```
$ git diff
```

Anzeigen aller Änderungen bei bereits getrackten Dateien (Dateiinhalt)

```
$ difftool --dir-diff
```

Wie git diff, aber mit grafischen Tool

```
$ git add *Pfad/zu/Datei*
```

Datei zum Committen markieren (stagen, in den Index verschieben)

```
$ git add .
```

Alle Dateien ab dem aktuellen Verzeichnis (inkl. aller Unterordner) zum Committen markieren (stagen, in den Index verschieben)

```
$ git reset
```

Dateien vom Status "Bereit für Commit" entfernen (Aber Änderungen bleiben erhalten, wird nur vom *Index* entfernt)

```
$ git commit [-m *Kommentar*]
```

Alle zum Committen markierte Dateien Committen **Ohne** den Parameter **-m** geht der konfigurierte Editor auf, der den Kommentar entgegennimmt und nach dem Schließen des Editors wird committed. **Mit** dem Parameter **-m** gibt man den Kommentar direkt auf der Kommandozeile an (bei vorhandenen Leerzeichen mit Anführungsstrichen umschließen)

```
$ git commit --amend
```

Den letzten Commit zurücksetzen und deren Änderungen wieder als lokale Änderungen einspielen, um so Dateien oder den Commit-Kommentar ändern zu können

NIEMALS BEREITS VERÖFFENTLICHTE COMMITS AMENDEN

HISTORY ANZEIGEN

```
$ git log
```

History des gesamten Repository anzeigen

```
$ git log *Pfad/zu/Datei*
```

History der Datei anzeigen

```
$ git blame *Pfad/zu/Datei*
```

Alle Änderungen in einer Datei anzeigen, dabei wird pro Zeile angezeigt, welche Änderung von wem committed wurde (wie *Annotate* im TFVC)

BRANCHES & TAGS

```
$ git branch *Branchname*
```

Branch mit dem Namen *Branchname* anlegen

```
$ git checkout *Branchname*
```

Zum Branch *Branchname* wechseln

```
$ git tag [-a] *Name*
```

Tag mit Namen und Beschreibung (Editor) im aktuellem Branch anlegen

UPDATE & PUBLISH

```
$ git fetch
```

Alle Änderungen des Remote Repository in das lokale Repository laden, dabei wird aber das Arbeitsverzeichnis nicht geändert

```
$ git pull
```

Alle Änderungen des Remote Repository in das lokale Repository laden und gleichzeitig die Arbeitskopie aktualisieren (Erzeugt ggf. eine Merge-Commit)

MÖGLICHST NICHT VERWENDEN. BITTE DEN BEVORZUGTEN WORKFLOW VERWENDEN

```
$ git push
```

Alle lokalen Commits zum Remote Repository schicken

```
$ git push --tags
```

Alle Tags im aktuellen Branch zum Remote Repository schicken

MERGE & REBASE

```
$ git merge *Branch*
```

Den angegebenen *Branch* in den aktuellen branch mergen, dabei entscheidet Git selbst, ob es rekursiv oder mittels fast-forward geschieht

```
$ git merge --ff-only
```

Wie merge, aber nur fast-forward zulassen. Ist das

nicht möglich, bricht der merge ab.

```
$ git merge --no-ff
```

Wie merge, aber immer rekursiv, selbst, wenn ein fast-forward möglich wäre.

```
$ git mergetool [--tool=<tool>]
```

Öffnet das konfigurierte Standard Mergetool, um die Merge Konflikte aufzulösen. Wird der Parameter für die Toolauswahl nicht gesetzt, nutzt git mergetool die Variable merge.tool aus der git Konfiguration. Falls diese nicht gesetzt ist, wird ein passendes Default-Programm geöffnet.

```
$ git rebase
```

Commit History neu schreiben, v.a. hilfreich, wenn beim fetch & merge eines Remote Branches kein Fast Forward möglich ist

UNDO

```
$ git reset --hard
```

Alle Änderungen an bereits getrackten Dateien rückgängig machen (betrifft keine neu erstellten Dateien)

```
$ git checkout .
```

Alle Änderungen an den Dateien rückgängig machen, die **nicht gestaged** sind (betrifft keine neu erstellten Dateien)

WORKFLOWS

Änderungen vom zentralen Repo integrieren

KEIN PULL VERWENDEN

Erst fetchen:

git fetch

Rebase ausführen:

git rebase

Bei Merge-Konflikten:

git mergetool

Anschließend das Rebasen abschließen:

git rebase --continue

WEITERFÜHRENDE LINKS

https://www.ralfebert.de/git/
http://tklepzig.github.io/git-branching-pres/