Grundbefehle	
git clone <url>	Erzeugt eine lokale Kopie eines Git Repositories (es werden keine Schreibrechte benötigt)
git status	gibt den aktuellen Zustand der Working Copy aus
git commit -a	Fügt alle lokalen Änderungen dem Repository hinzu (ohne Übertragung an den Remote Host)
git reset --hard origin/<branch-name>	Setzt local auf den HEAD Stand zurück. Lokale commits werden verworfen.
git clean -f -d	Löscht alle temporären Dateien und Folder z.B. Files die nicht unter git Kontroller stehen
git push	Veröffentlichung der lokalen Commits auf dem Remote Repository
git branch	Liste alle lokal bekannten Branches auf
git branch <branch-name>	Erstellt lokal einen neuen Branch, bleibt aber auf dem Alten
git branch -D <branch-name>	Löscht einen Branch unabhängig davon ob er im upstream existiert
git push origin --delete <branch>	Löscht auch den Remote Branch erfolgreich
git checkout <branch-name>	Wechselt zu einem anderen Branch
git tag --delete <tagname>	Löscht den Tag lokal
git push --delete origin <tagname>	Löscht den Tag im Remote Repo
git stash	Lokale Änderungen als Backup auf einen internen Stack legen.
git stash pop	Lokale Änderungen aus dem Stack wiederholen
git remote add upstream <url>	Alias *upstream* zum master eines Remote Repos definieren
git remote add –track <branch-name> upstream <url>	Alias *upstream* zum Branch mit Repo URL definieren
git fetch upstream	Aktuellen Stand vom Alias *upstream* herunterladen
git merge <branch> - -no-commit - -no-ff	<branch> wird in den aktuell ausgecheckten branch gemerged (ohne commit)
git merge upstream/master	Merge der Änderungen vom Alias *upstream* branch "master" in den lokalen branch wobei für konfliktfreie Änderungen ein Autocommit erfolgt.
git merge upstream/master - -no-commit - -no-ff	Merge der Änderungen vom Alias *upstream* branch "master" in den lokalen branch wobei für konfliktfreie Änderungen KEIN Autocommit erfolgt.
git clone --mirror <repourl>	Spiegelt ein Repository und kann zur Erstellung eines lokalen Backups verwendet werden.
git remote update	In einem Backup die remote Updates einspielen

By **Huluvu424242**
(FunThomas424242)

cheatography.com/funthomas424242/
github.com/Huluvu424242

Published 6th April, 2015.
Last updated 29th April, 2019.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Grundbefehle (cont)

git push <remotename> <commit SHA>:<remotebranchname>	Push eines ausgewählten lokalen commits. z.B. git push origin 712acff81033eddc90bb2-b45e1e4cd031fefc50f:master
git branch -m new-name	Aktuellen lokalen Branch in new-name umbenennen
git branch -m old-name new-name	Einen lokalen Branch old-name in new-name umbenennen
git push origin :old-name new-name	Erst Prüfen!: Alten remote Branch löschen und den neuen lokalen Branch pushen
git push origin -u new-name	Aktuellen Branch im Remote umbenennen

Alle lokalen Änderungen müssen um im Remote Repository sichtbar zu werden über den Mechanismus commit und push in das Remote Repository eingetragen werden (auch neu angelegte Branches).

Github - Pull Requests aktuell halten

git clone <url>	Lokale Arbeitskopie erstellen
git checkout <branch-name>	Auf den Pull Request wechseln
git remote add upstream git://github.com/owner/projectname.git	Alias *upstream* für den Zugriff auf den Master definieren
git fetch upstream	Die Änderungen vom *master* herunterladen
git merge upstream/master	Den *master* in den lokalen Arbeitsstand mergen
git commit -a	Alles commiten wenn keine Konflikte mehr bestehen
git push	Nach dem erfolgreichen lokalen bauen, den aktuellen Stand veröffentlichen.

Generell sollte für jedes neue zu realisierende Feature ein eigener Feature Branch vom *master* gezogen werden.
Hat man das Feature erfolgreich implementiert kann man auf github.com einen pull Request stellen.
Wenn dieser Pull Request vom Owner nicht gemerged werden kann weil es Änderungen auf dem *master* gab dann hilft das oben beschriebene Vorgehen.

Neuen Branch mit lokalen Änderungen erstellen

git status	Gibt den aktuellen Branch aus und zeigt welche lokalen Änderungen vorliegen
git stash	Lokale Änderungen auf den Stack legen
git branch <neuer-branch-name>	Neuen Branch anlegen, Abspaltung vom Stand des aktuellen Branches
git checkout <neuer-branch-name>	Wechseln auf den neuen Branch
git stash pop	Lokale Änderungen vom Stack in den neuen Branch laden
git commit -a	Alle Änderungen auf den neuen Branch commiten
git push	Nach erfolgreichem lokalen Bauen die Änderungen und den neuen Branch an das Remote Repository publizieren.

Es gibt Tage das will man schnell mal was ausprobieren und dann funktioniert es gleich - unglaublich. Aber natürlich ist man mit den lokalen Änderungen gerade auf dem frisch ausgecheckten *master* und will die Änderungen aber als neuen Feature Branch commiten - was tun? Das was oben steht ;)

By **Huluvu424242**
(FunThomas424242)

cheatography.com/funthomas424242/
github.com/Huluvu424242

Published 6th April, 2015.
Last updated 29th April, 2019.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Git für Präsentationen

Wer in einer Präsentation zum nächsten oder vorherigen Stand des Programmes springen möchte ohne das Risiko von Schreibfehlern zu riskieren der sollte sich die Git Aliase prev und next anlegen. Damit lässt sich zum vorhergehenden oder nächsten Commit wechseln.
(Quelle: https://coderwall.com/p/ok-iyg/git-prev-next)
Die Aliase werden in der ~/.gitconfig eingetragen

```
[alias]
prev = checkout HEAD^1
next = "!sh -c 'git log --reverse --pretty=%H master | awk \"/$(git rev-parse HEAD)/{getline;print}\" | xargs git checkout'"
```

By **Huluvu424242**
(FunThomas424242)

cheatography.com/funthomas424242/
github.com/Huluvu424242

Published 6th April, 2015.
Last updated 29th April, 2019.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com