

Git – der Spickzettel

Version vom 30. November 2010 <https://github.com/esc/git-cheatsheet-de>
© 2010 V. Hänel, J. Plenz Creative-Commons-Lizenz 3.0 (By.Nc.Sa)

Repositories klonen oder erstellen

```
git clone git://... Klonen (Git-Protokoll)  
git clone ssh://... Klonen (SSH)  
  
cd projekt  
git init  
git add .  
git commit
```

Neues Repository erstellen

Referenzen

HEAD	Letzter Commit
HEAD^	Vorletzter Commit
HEAD~n	n-ter letzter Commit
master	Branch-Name
v2.6.17	Tag-Name (z. B. für Versionierung)
a..b	Alle Commits zwischen a und b
master..	Commits, die noch nicht in master sind
44ac95d...	Objekt-Referenz (Commit, Tree, etc.)

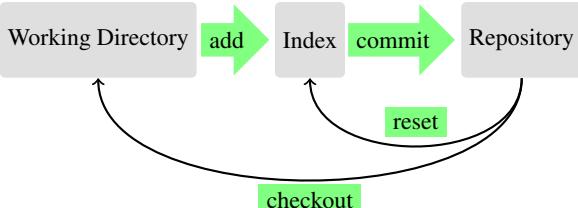
Änderungen untersuchen

Jedes der Kommandos akzeptiert optional als Argument eine **Referenz** auf einen oder mehrere Commits (siehe zweite Spalte).

git log	Zusammenfassung der Änderungen
git show	Detaillierte Ansicht eines Commits
git diff	Unterschiede mit diff anzeigen
git blame	Wer hat welche Zeile geändert?
gitk --all	Grafische Ansicht
tig	Curses-Frontend für Git

Nächsten Commit vorbereiten

git add	Datei hinzufügen
git add -p	Teile einer Datei hinzufügen
git rm	Datei löschen
git mv	Datei verschieben
git commit	Commit erstellen
... -m 'msg'	mit Beschreibung msg
... --amend	Letzten Commit verbessern



Index vs. Working Directory

git diff --cached	Änderung zwischen Index und Repository (HEAD)
git rm --cached	Datei nicht mehr beachten
git reset ref	Den Index auf ref setzen; Working Dir. unverändert
git checkout ... -b branch	Index und WD ändern Zu neuem branch wechseln

Branches

git branch name	Branch name erstellen
git checkout name	Branch name auschecken
git merge feature	feature integrieren
git rebase basis	Aktuellen Branch auf neue basis aufbauen
... --interactive	Commits ordnen und anpassen

Merge-Konflikte beheben

git diff --ours	Unterschied zu unserer
... --theirs	... und deren Version
gitk --merge	Relevante Commits untersuchen
git mergetool	Three-Way-Merge
git add, commit ...	Änderungen aufnehmen
git reset --hard	Merge abbrechen

Remote Repositories

git remote	Remote Repositories verwalten
git fetch ...	Neue Commits herunterladen
git pull und gleich mergen

Änderungen veröffentlichen

git push	Commits hochladen
git tag	Commit markieren (z. B. für eine neue Version)
git format-patch	Änderungen im Patch-Format exportieren, für E-Mail-Versand

Platz für eigene Anmerkungen!

Weitere praktische Kommandos

git grep expr	In allen Dateien nach expr suchen
git stash	Änderungen temporär in den Hintergrund schieben
git clean	Nicht von Git verwaltete Dateien löschen
git bisect	Commit finden, der ein bestimmtes Problem verursacht
git archive	Tar-Ball des Projektes erzeugen
git shortlog	Anzahl der Commits zählen (per -s -n)

Beispielhafte Programmaufrufe

Einen leeren Commit erstellen, praktisch z. B. für den *initial commit*:
\$ git commit --allow-empty -m 'leer'

Herauszufinden, durch welchen Commit eine **Datei gelöscht** wurde:
\$ git log --diff-filter=D datei

Einen Patch der Änderungen erstellen, die zwischen dem Tag v1.6 und dem Branch *neues-feature* bestehen:
\$ git diff v1.6..neues-feature > meine-aenderungen.patch

ASCII-Baumdiagramm aller Commits anzeigen:
\$ git log --graph --oneline --decorate --all

Alle unnötigen Dateien löschen, Änderungen verwerfen und mit einer „sauberen“ Version von *master* starten:

```
$ git checkout -f master  
$ git clean -dfx
```

Throw-Away-Integration: Testen, ob ein Merge funktionieren würde, ohne tatsächlich master zu ändern:

```
$ git checkout -b integrate master  
$ git merge feature
```

Den letzten Commit in zwei kleinere aufteilen (siehe auch man git-rebase):

```
$ git rebase -i HEAD^  
(pick durch edit ersetzen, speichern)  
$ git reset HEAD^  
$ git add -p  
$ git commit -m 'Erster Teil'  
$ git add -u  
$ git commit -m 'Zweiter Teil'
```