git Kurzreferenz

Grundlagen

Hilfe erhalten

```
git help <command>
```

Konzepte

branch	Abgespaltener Entwicklungsstrang
revision	Eindeutiger Stand eines Branches
ref	Zeiger auf Revision, z.b HEAD
id	SHA1-Summe über Revision
master	Default Branch
	entspr. HEAD bei CVS / trunk bei SVN
origin	default name für push und pull Zielrepository
HEAD	ref auf letzte Revision aktueller Branch
HEAD^	ref auf Revision davor (parent von HEAD)

Konfigurieren

Globale Konfiguration liegt unter
`$HOME/.gitconfig` (`git config --help`)

Konfiguration pro Repository unter
`.git/config`

Dateien ignorieren

```
echo "<dateiname>" >> .gitignore
```

Benutzername global einstellen

```
git config --global user.name "<name>"
```

Email global einstellen

```
git config --global user.email <email>
```

Erstellen

Neues Repository anlegen

```
git init
git add .
```

Existierendes Repository klonen

```
git clone ssh://you@host.org/proj.git
```

verlinkt mit Repository als 'origin'

Mit einem anderen Repository verlinken

```
git remote add <name> <url>
```

Branchen

Auf Branch <branch> wechseln (~svn switch)

```
git checkout <branch>
```

nicht eingecheckte Änderungen bleiben erhalten!

Neuen Branch erstellen und auschecken

```
git checkout -b <new> [<old>]
```

basiert auf HEAD des Branch <old> bzw. aktuellem

Branch von remote holen (und lokal erstellen)

```
git checkout -t origin/<branch>
```

Branch auf remote einchecken (ggf. erstellen)

```
git push origin <branch>
```

Branch von remote folgen (für git pull)

```
git branch --set-upstream <b> origin/<b>
```

git pull geht nun auch in diesem Branch. (b=branch)

Branch <branch> löschen

```
git branch -d <branch>    lokal löschen
git push origin :<branch> remote löschen
```

Aktualisieren

Letzte Änderungen holen (z.B. für diff)

```
git fetch
```

verändert nicht den workspace

Workspace aktualisieren (fetch + merge)

```
git pull
```

merged die Änderungen direkt in den workspace

Patch ausführen

```
git am -3 patch.mbox
```

(im Konfliktfall lösen und dann `git am --resolved`

Veröffentlichen

Lokale Änderungen commiten (nach git add/rm)

```
git commit
```

Alle lokalen Änderungen commiten (impliziert add)

```
git commit -a
```

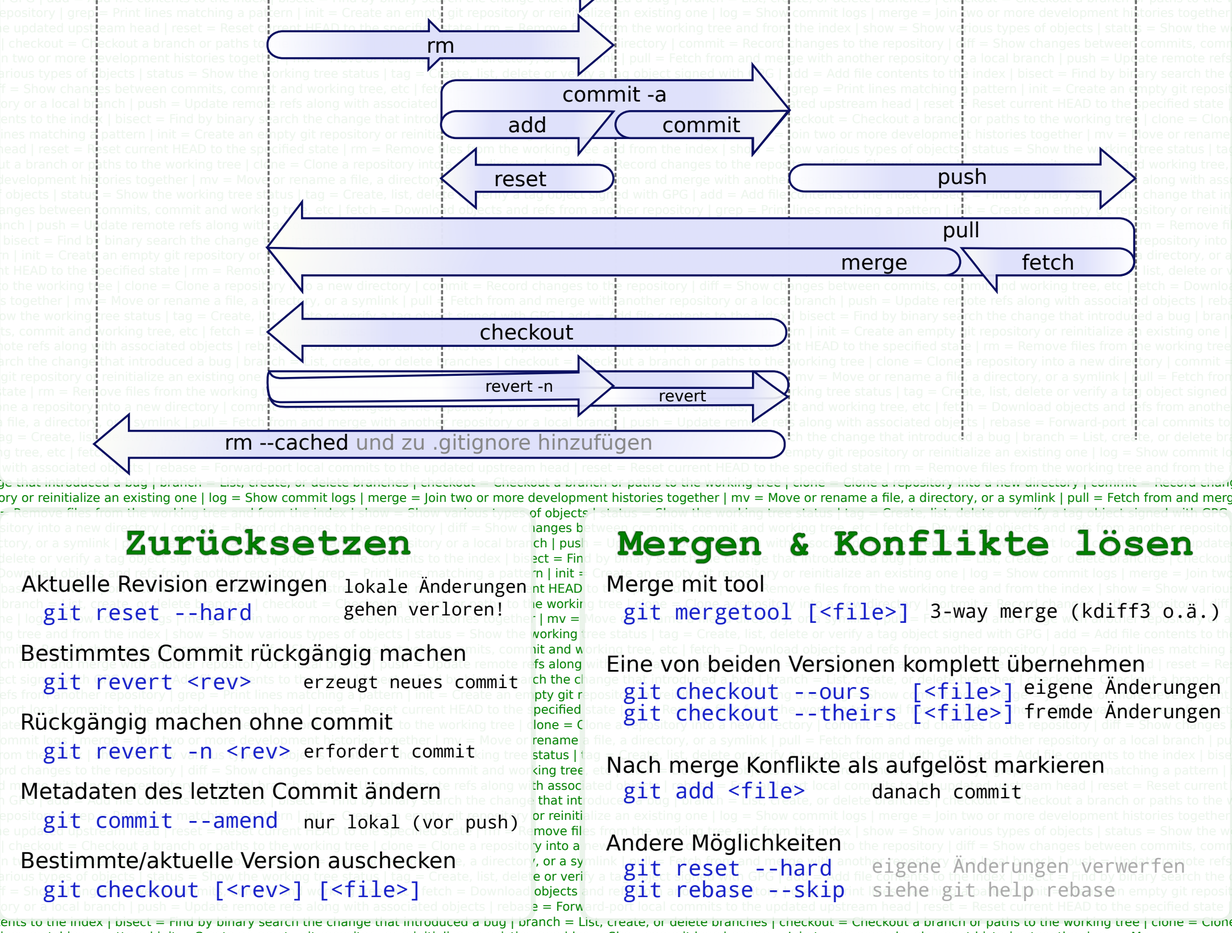
Änderungen von lokalem repo auf origin schieben

```
git push
```

Patch erstellen

```
git format-patch origin
```

Lebenszyklus der Dateien in git



Zurücksetzen

Aktuelle Revision erzwingen `lokale Änderungen`

```
git reset --hard
```

`gehen verloren!`

Bestimmtes Commit rückgängig machen

```
git revert <rev>
```

`erzeugt neues commit`

Rückgängig machen ohne commit

```
git revert -n <rev>
```

`erfordert commit`

Metadaten des letzten Commit ändern

```
git commit --amend
```

`nur lokal (vor push)`

Bestimmte/aktuelle Version auschecken

```
git checkout [<rev>] [<file>]
```

Mergen & Konflikte lösen

Merge mit tool

```
git mergetool [<file>]
```

`3-way merge (kdiff3 o.ä.)`

Eine von beiden Versionen komplett übernehmen

```
git checkout --ours   [<file>]
git checkout --theirs [<file>]
```

`eigene Änderungen`
`fremde Änderungen`

Nach merge Konflikte als aufgelöst markieren

```
git add <file>
```

`danach commit`

Andere Möglichkeiten

```
git reset --hard
git rebase --skip
```

`eigene Änderungen verwerfen`
`siehe git help rebase`

Anzeigen

Historie der Änderungen

```
git log
```

Geänderte Dateien anzeigen

```
git status
```

Unterschied workspace zu lokalem Repository

```
git diff <file>
```

Graphisches tool verwenden (z.B. kdiff3)

```
git difftool <file>
```

Änderungen zwischen Revisionen

```
git diff <rev1> <rev2>
```

Änderungen in <parent> seit Brancherzeugung

```
git diff <branch>...<parent>
```

Letzte Änderungen einer Datei

```
git log -p -1 <file>
```

Zeige Commit-Messages und Benutzer

```
git blame <file>
```

Zeige Änderungen einer Revision (log+diff)

```
git show <rev>
```

Inhalt einer bestimmten Datei anzeigen (~cat)

```
git show <rev>:<file>
```

Alle (lokalen) Branches anzeigen

```
git branch
```

Der Stern '*' markiert den aktuellen Branch
-r für remote Branches / -a für alle

Sonstiges

Repository überprüfen

```
git fsck
```

Repository aufräumen

```
git gc
```

Suchen in Dateien

```
git grep "<suchbegriff>"
```

Suchen in Änderungen (Zeilen +/- Suchbegriff)

```
git diff -S"<suchbegriff>" ..
```

Markieren einer Version

```
git tag <version>
```

Markieren einer Revision als gut/schlecht

```
git bisect good/bad <rev>
```

Legende

<rev>

Revisionsbezeichner
siehe git help revisions

<branch>

beliebiger Branchname

<file>

beliebige Datei(en)