`git ls-files`
Listet alle Dateien auf, die im Stage liegen.
Das schließt auch Dateien ein, die bereits committet sind.
Durch `git rm --cached <datei>` wird sie aus dem Stage entfernt.
Liefert Infos der Art

```
100644 78981922613b2afb6025042ff6bd878ac1994e85 0 a.txt
```

`git rm --cached <datei>`
Löscht die Datei aus dem Stage. Sinnvoll, wenn die Datei in zukünftigen Commits nicht mehr auftauchen soll. Sie muss dafür in die `.gitignore` aufgenommen werden.

`git reset`
Entfernt eine Datei aus dem Stage, als wäre kein `.add` erfolgt. Ist quasi „einen Schritt zurück“. Änderungen nach dem `.add`, die an anderen Dateien erfolgt sind, bleiben erhalten!

`git reset --hard`
Dateien nach dem letzten `.add` bleiben unangetastet, der Inhalt des Stage wird verworfen.

`git cat-file -t <hash>`
Typ des Objekts wird geliefert (tree, blob, commit)

`git cat-file -p <hash>`
Liefert den Inhalt des Objekts. Bei *blob* ist es der Inhalt, bei *tree* sind die Dateien im Tree. Bei einem Commit erhält man die Message, Parent, ... und auch den Hash des trees!

Kernbefehle

`git restore <datei>`
Holt die Version aus dem Stage und überschreibt die Version im Workindir.

`git restore <datei> --staged`
Wenn im Workingdir und im Stage verschiedene Versionen existieren, wird die aus dem Stage einfach gelöscht.

Existiert aktuell keine Version mehr im Workingdir, so wird die Datei im Stage nicht gelöscht, sondern nur ins Workingdir verschoben.

`git restore --source=<HASH> <datei>`
Der Befehl holt eine Datei aus einem bestimmten Commit. Früher war das
`git checkout <HASH> -- <datei>`
Man kann hier auch noch andere Quellen angeben!

Werkzeuge

`git show <HASH>:datei.txt`