Git-Spickzettel Nr. 1

Git konfigurieren

```
$ git config [--global] [option]
```

mit `--global`: wird es in `~/.gitconfig` gespeichert

Information über den Anwender

```
user.name NAME
user.email EMAIL
```

Farbige Ausgabe einschalten

```
color.ui auto
```

Verbesserung der interaktiven Bedienbarkeit

```
interactive.singlekey true
```

Erstellen und Klonen von Repositorien

Von existierenden Daten

```
$ cd  my_project_dir
$ git init
$ git add .
$ git commit -m 'initial commit'
```

Von existierendem Repo

```
$ git clone path/to/existing/repo path/to/new/repo
$ git clone you@host.de:dir/project.git
$ git clone http://[USER@]host.de/project.git
```

Informationen erhalten

Veränderte Dateien im Arbeitsverzeichnis

```
$ git status
```

Änderungen an überwachten Dateien

```
$ git diff
```

Änderungen zwischen ID1 und ID2

```
$ git diff <ID1> <ID2>
```

Änderungsverlauf

```
$ git log
$ gitk
```

Arbeiten mit dem Index und committen (dt.: beitragen) von Änderungen

Füge alle Änderungen in einer Datei oder Verzeichnis zum Index hinzu

```
$ git add path/to/add
```

Wähle interaktiv alle Änderungen zum Hinzufügen/Committen aus

```
$ git add -p [path/to/preselect/changes]
$ git commit -p
```

Committe alle hinzugefügten Änderungen zum Index

```
$ git commit
```

Füge alle lokalen Änderungen hinzu und committe sie

```
$ git commit -a
```

Committe Änderungen mit direkter Angabe einer Änderungsnachricht

```
$ git commit -m «message>"
```

Arbeiten mit Branches (dt.: Zweige)

Alle Branches auflisten

```
$ git branch
$ git branch -a
```

Listet auch entfernte* Branches auf

Zu einem Branch wechseln

```
$ git checkout <branch>
```

Branch B1 in B2 mergen (dt. Zusammenführen)

```
$ git checkout <B2>
$ git merge <B1>
```

oder

```
$ git merge --no-ff <B1>
```

Erstelle einen Commit auch wenn fast-forwarding möglich ist

Erstellen eines Branches basierend auf HEAD und wechselt zu diesem

```
$ git checkout -b <branch>
```

Löschen eines Branches

```
$ git branch -d <branch>
$ git branch -D <branch>
```

Lösche einen, noch nicht mit dem Standard-Branch gemergeten, Branch

Änderungen von entferntem Repos holen

Lade Änderungen von remote *(einem entfernten Repo) herunter

```
$ git fetch [remote]
```

`remote` ist voreingestellt auf: `origin`

Änderungen bekommen

```
$ git pull [remote] [refspec]
```

`remote` defaults to `origin`

Änderungen veröffentlichen

Änderungen nach remote pushen (also in ein entferntes Repo schieben)

```
$ git push [origin] [branch]
```

Einen tag (dt.: ???) erstellen

```
$ git tag [-s] <tag name>
```

mit `-s` den tag mit GPG signieren

Einen Patch vorbereiten

```
$ git format-patch origin
```

Änderungen rückgängig machen

Zum zuletzt committeten Zustand zurückkehren

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

Einen bestimmten Commit rückgängig machen

```
$ git revert <ID>
```

Dies ist sicher für bereits veröffentlichte Commits

Korrigieren/ Ändern des letzten Commits

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

Tue dies *niemals* mit bereits veröffentlichten Commits außer du weißt genau, was du tust

Verschiedenes

Dokumentation/ Hilfe

```
$ git help [command]
$ man git-[command]
```

Einen Branch löschen (lokal und remote)

```
$ git branch -d <branch>
$ git push <origin> :<branch>
```

Die unveröffentlichte Historie interaktiv aufhübschen

```
$ git rebase -i <ID>
```