# Versionsverwaltung: Einführung in GIT

[n/a: Keine Treffer]

## Aufgabe 1: Ein erstes Repo

Einfaches Beispiel.

```
$ mkdir webseite
$ cd webseite
$ git init
Leeres Git-Repository in /home/frank/Downloads/webseite/.git/ initialisiert
```

Jetzt stelle das Verzeichnis `webseite` ein lokales Git-Repository dar. Wenn man sich den Inhalt des Verzeichnisses genau ansieht, stellt man fest, dass es dort ein Verzeichnis `.git` gibt:

```
$ ls -la
insgesamt 132
drwxr-xr-x  3 frank frank   4096 24. Okt 13:32 .
drwxr-xr-x 21 frank frank 122880 24. Okt 13:32 ..
drwxr-xr-x  7 frank frank   4096 24. Okt 13:32 .git
```

git status Auf Branch master

Noch keine Commits

Damit git geschmeidig funktioniert, sollte man für spätere Änderungen und dergleichen noch festlgen, wer man eigentlich ist.

```
$ git config --global user.name "Mein Name"
$ git config --global user.email nix@example.org
```

## Aufgabe 2: Ein erster Commit

Den aktuellen Status eines Repositorys kann man sich durch den Befehl `git status` anzeigen lassen:

```
$ git status
Auf Branch master

Noch keine Commits

nichts zu committen (erstellen/kopieren Sie Dateien und benutzen
Sie "git add" zum Versionieren)
```

Lege nun eine `index.html` Datei ein sowie zwei Verzeichnisse - `css` und `img`:

```
sbel@r107-ws15:~/git$ touch index.html
sbel@r107-ws15:~/git$ mkdir css
sbel@r107-ws15:~/git$ mkdir img
sbel@r107-ws15:~/git$ ls
css  img  index.html
```

- Neue Dateien befinden sich zunächst im Arbeitsverzeichnis und werden von git ignoriert. Teste das mit `git status`.
- Mit dem Befehl `git add` wird eine Datei in gits „Staging Area“ verschoben - das kann man sich vorstellen wie ein Einkaufswagen, in dem neue Dateien und Änderungen gesammelt werden, bis man zu einem Punkt kommt, den man sich „merken“ möchte. Dann macht man einen „Commit“. Füge die Datei `index.html` deiner Staging Area hinzu und kontrolliere das Ergebnis mit `git status`
- Führe den Befehl `git commit` aus, gib eine Commit-Message an.
- Überprüfe den Zustand von Arbeitsverzeichnis und Staging Area mit `git status`. Schau dir die Liste deiner Commits mit `git log` an.

# Aufgabe 3: Workflow

Der Workflow sieht jetzt einfach so aus, dass man Dateien ändert oder hinzufügt und die Änderungen in der Staging Area vorhält bis zum nächsten Commit, dann führt man `git commit` aus um sich die „nächste Version“ zu merken.

- Füge im Verzeichnis `css` eine Datei `style.css` ein, die einige Informationen zur Formatierung von Überschriften enthält.
- Versehe die Datei `index.html` mit einem HTML Grundgerüst, das die `style.css` Datei einbindet.[1)]
- Betrachte mit `git status` die Änderungen im Arbeitsverzeichnis. Füge fehlende Dateien zur Staging Area hinzu. Mache einen Commit.

# Tags als Lesezeichen

Man kann sich den Zustand es Repos merken als „Tag“, das ist wie ein Lesezeichen:

```
 git tag -l // zeigt tags
 git tag v1 // legt das tag v1 an
```

Um zu einem solchen „Lesezeichen“ zurückzukehren, kann man den Tag ins lokale Repo „auschecken“:

```
git checkout v1
```

Task: Zurückgehen in der Zeit

Aktuellen Zustand merken:

```
git tag v2
```

Zielcommit raussuchen:

```
git log
```

Zielcommit auschecken:

```
git checkout <commit-id>
```

Umsehen - ist es das was man sich merken will? wenn ja: Tag anlegen mit `git tag v0`.

Nun kann man zwischen den Tags hin und her wechseln, wie man möchte.

[1)]
Infos zu HTML und CSS findest du im alten Wiki:
https://scotty.qg-moessingen.de/itg/doku.php?id=kurs:kursstufe:html_css:start