Geklonte Repositorys

Zunächst ist ein Git-Repo wie wir gesehen haben, eine **vollkommen lokale Angelegenheit** - alle wichtigen Informationen und die Snapshots werden im .git-Verzeichnis gespeichert.

Um besser zusammenarbeiten zu können ist es möglich, ein Repo über entsprechende Mechanismen anderen Menschen zur Verfügung zu stellen, z.B. via Mail, SSH oder https. Diese Weitergabe muss nicht unbedingt öffentlich geschehen, sondern kann durchaus z.B. mit einem Passwort geschützt werden, so dass nur Mitglieder einer bestimmten Gruppe Zugriff auf das Repo haben. Außerdem kann zwischen lesendem Zugriff und schreibendem Zugriff unterschieden werden und vieles mehr.

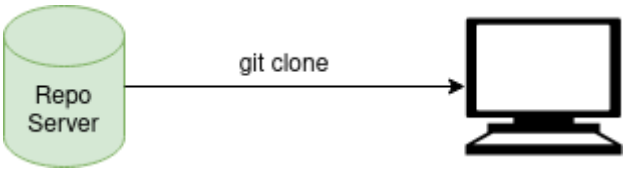
Sehr beliebt sind für solche Austauschzwecke grafische Webanwendungen, die bekannteste dürfte die Microsoft Plattform "Github" sein, in Deutschland ist eine bekannte Plattform "Codeberg"[1)] auf der sich auch viele Codebeispiele aus dem Wiki befinden. Man kann eine solche Austauschplattform aber auch selbst betreiben, wenn man möchte.

Klonen eines Repositorys

Ein so veröffentlichtes Repo kann man "klonen". Der Befehl lautet z.B.

```
git clone https://codeberg.org/info-bw-wiki/git-kurs-tagebuch.git
```

Die Adresse, von der das Repo geklont wird, heißt `origin`.




```
max@pc:git-clone$ git clone https://codeberg.org/info-bw-wiki/git-kurs-
tagebuch.git
Klone nach 'git-kurs-tagebuch'...
remote: Enumerating objects: 12, done.
remote: Counting objects: 100% (12/12), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 12 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Empfange Objekte: 100% (12/12), fertig.
Löse Unterschiede auf: 100% (1/1), fertig.
max@pc:git-clone$
```

Der Begriff des Klonens ist hier wörtlich zu nehmen - jetzt existiert eine vollständige Kopie des Repos auf dem lokalen Rechner, die alle Commits des ursprünglichen Repos nachverfolgbar enthält. Bevor man mit git arbeiten kann, muss man allerdings in das Verzeichnis hineinwechseln, das beim Klonen erstellt wurde - hier `git-kurs-tagebuch`, das geht mit dem Befehl `cd git-kurs-tagebuch`.

```
max@pc:git-kurs-tagebuch$ git lg --all
* d7aaac4 - (HEAD -> main, origin/main, origin/HEAD) Mittagessen hinzugefügt (vor 5 Tagen)
* adc15c2 - fruehstueck.txt geändert (vor 5 Tagen)
* f9353d6 - Frühstück in Datei 'fruehstueck.txt' hinzugefügt (vor 5 Tagen)
* 28ec5a7 - Erster Commit: aufstehen.txt angelegt (vor 5 Tagen)
max@pc:git-kurs-tagebuch$
```

Änderung auf den Server zurückkopieren

Nun kann man mit dem Repo lokal ganz normal arbeiten, der wesentliche Unterschied zum ausschließlich "lokalen" Repository ist, dass dieses Repository "weiss", woher es kommt – das ermöglicht es, Änderungen auch wieder auf den entfernten Server zurück zu übertragen. Der dazu verwendete Befehl lautet `git push`.

Zunächst bearbeitet man lokal Dateien im Repo und erzeugt einen (oder mehrere Commits) :

```
max@pc:git-kurs-tagebuch$ nano nachmittags.txt
max@pc:git-kurs-tagebuch$ ls -la
insgesamt 28
drwxr-xr-x 3 max max 4096  7. Okt 19:34 .
drwxr-xr-x 3 max max 4096  7. Okt 19:30 ..
-rw-r--r-- 1 max max   55  7. Okt 19:30 aufstehen.txt
-rw-r--r-- 1 max max   30  7. Okt 19:30 fruehstueck.txt
drwxr-xr-x 8 max max 4096  7. Okt 19:31 .git
-rw-r--r-- 1 max max   25  7. Okt 19:30 mittagessen.txt
-rw-r--r-- 1 max max   16  7. Okt 19:34 nachmittags.txt
max@pc:git-kurs-tagebuch$ git add nachmittags.txt
max@pc:git-kurs-tagebuch$ git commit -m "Nachmittagsbeschäftigung ergänzt"
[main 354c303] Nachmittagsbeschäftigung ergänzt
 1 file changed, 2 insertions(+)
 create mode 100644 nachmittags.txt
max@pc:git-kurs-tagebuch$
```

Mit dem Befehl `git push` kann man die Änderungen nun zum "origin" zurück übertragen:

```
max@pc:git-kurs-tagebuch$ git status
Auf Branch main
Ihr Branch ist 1 Commit vor 'origin/main'.
  (benutzen Sie "git push", um lokale Commits zu publizieren)

nichts zu committen, Arbeitsverzeichnis unverändert
max@pc:git-kurs-tagebuch$ git push
Username for 'https://codeberg.org': geheim
Password for 'https://geheim@codeberg.org':
Objekte aufzählen: 4, fertig.
Zähle Objekte: 100% (4/4), fertig.
Delta-Kompression verwendet bis zu 4 Threads.
Komprimiere Objekte: 100% (2/2), fertig.
Schreibe Objekte: 100% (3/3), 312 Bytes | 312.00 KiB/s, fertig.
```

```
Gesamt 3 (Delta 1), Wiederverwendet 0 (Delta 0), Pack wiederverwendet 0
To https://codeberg.org/info-bw-wiki/git-kurs-tagebuch.git
   d7aaac4..354c303  main -> main
```

Damit landen die Änderungen auf dem Server, von dem das Repo zuvor geklont wurde. Es darf aber natürlich nicht jeder auf jedes im Internet zugänglich Repository Änderung zurückspielen, sondern nur diejenigen, die dazu berechtigt sind. Wenn man den dargestellten Ablauf genau studiert, erkennt man, dass es notwendig war, Benutzername und Passwort anzugeben.

Wenn die Seite des Repositorys – wie im Beispiel des Tagebuchs – öffentlich zugänglich ist, kann aber **jeder** die Änderungen betrachten und das Repository zu sich klonen!

(A1)

- Öffne das Repo auf Codeberg: https://codeberg.org/info-bw-wiki/git-kurs-tagebuch Erkunde dort in der Weboberfläche die Änderungen. Findest du den Branch mit den Blaubeeren zum Frühstück?

Aktuell bleiben mit "pull"

Um Probleme beim "pushen" von Änderungen zu vermeiden - besonders wenn man in einem Team an einem Repo arbeitet, oder wenn man das Repo auf mehreren Rechnern gleichzeitig bearbeitet, muss man seine lokale Arbeitskopie aktuell halten. Das geschieht mit dem Befehl `git pull`.

Man sollte vor Arbeitsbeginn pullen und bevor man versucht zu pushen. wenn Konflikte auftreten, weil zwischenzeitlich Änderungen auf das zentrale Repository gepusht wurden, kann man diese zunächst lösen und dann das Ergebnis pushen. Wie man Konflikte löst betrachten wir gesondert.

[1)]
https://codeberg.org/