

### Vokabeln

Git	Software zur Versionsverwaltung von textuellen Dateien (zB. Code)
GitHub, GitLab, etc.	Plattformen die Git Server, nützliche Tools und grafische Oberflächen zur Verfügung stellen um gemeinsames Arbeiten zu vereinfachen
Repository	Beinhaltet alle zu einem (Unter-)Projekt gehörenden Ordner und Dateien die verwaltet werden sollen
Branch	Eine "Abzweigung" bzw. Kopie von einem existierenden Branch (zB. master/main) an der man arbeiten kann ohne das Original zu verändern
Commit	Ein einzelner "Snapshot" der zuletzt gemachten Änderungen. Mit Commits speichern wir die Änderungen in der Historie ab

### GitHub Repository klonen (downloaden)

- 1) Navigiere mit der **Git Bash** zum Ordner in den das Projekt gedownloadet werden soll
  - Entweder **Rechtsklick** -> **Open Git Bash here** auf gewünschten Ordner
  - Oder in offener Git Bash per **cd PFAD/ZUM/ORDNER** zum Ordner wechseln
- 2) In GitHub im gewünschten Repository den **HTTPS Link** kopieren der unter dem **grünen Code Button** zu finden ist
- 3) Gib folgenden Befehl in der Git Bash ein:
 

```
git clone [HTTPS _LINK]
```

### Allgemeiner Workflow

<code>git fetch</code>	<b>Lädt Informationen</b> zu eventuell neuen Updates aus dem GitHub Repository, <b>wendet sie aber noch nicht an</b>
<code>git status</code>	<b>Zeigt Informationen</b> zu aktuellem Status an. Dazu zählen <b>vorliegende Updates</b> die angewendet werden können, sowie deine <b>lokalen Änderungen</b> die noch nicht committed wurden
<code>git pull</code>	<b>Wendet Updates an</b> die im GitHub Repository vorlagen
<code>git log</code>	<b>Zeigt Historie der Commits</b> an mit einigen Infos
<code>git branch</code>	<b>Zeigt lokal existierende Branches</b>
<code>git checkout [BRANCH]</code>	<b>Aktiviert und wechselt zu angegebene Branch</b>

### Änderungen und Beiträge machen

<code>git checkout -b [BRANCH_NAME]</code>	<b>Erstellt neuen Branch</b> vom aktuell aktiven Branch (meist master/main) und <b>wechselt zu diesem</b>
<code>git add [PFAD_ ZUR _DATEI]</code>	<b>Neue/Geänderte Datei in den "Warteraum"</b> für deinen nächsten Commit schicken
<code>git commit -m [BESCHREIBUNG]</code>	<b>Speichert Änderungen</b> an den geaddeten Files in der Historie und <b>hängt Beschreibung</b> an diesen Snapshot an

### Fehler gemacht? Nicht schlimm

<code>git restore [--staged] [DATEI _PFAD]</code>	<b>Reverted angegebene Datei</b> zu dem Stand vom letzten Commit. Mit der <b>-staged flag</b> können Dateien aus dem "Warteraum" entfernt werden
<code>git reset --soft HEAD~1</code>	<b>Löscht den letzten Commit</b> und schickt die dort geänderten Files zurück in den "Warteraum". Zahl kann angepasst werden um mehr als einen Commit zu resetten

### Lokale Änderungen und Beiträge hochladen

1) Sicherstellen dass alle lokalen Änderungen entweder committed, oder reverted sind (git status sollte keine Files mehr zeigen die geadded werden müssen)

2) Nutze den Befehl: `git push`

- Kann sein dass er sagt dass dein Branch noch nicht im Repository existiert und dir folgenden Befehl als Lösung vorschlägt

```
git push --set-upstream origin [BRANCH_NAME]
```

3) Sobald du die Änderungen deines Branches wieder in den Hauptbranch bringen willst, erstelle auf GitHub einen Pull Request dafür

- Im GitHub Repository oberhalb des Codes auf **Branches** klicken

- Bei gewünschtem Branch **New pull request** klicken und nötige Infos angeben



By **testtesttest123**

[cheatography.com/testtesttest123/](https://cheatography.com/testtesttest123/)

Not published yet.

Last updated 24th July, 2023.

Page 1 of 2.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>