Git Tutorial

Git installieren

Link zu Git Homepage

- installieren von git für Windows
- macOS und Linux sollten bereits mit vorinstalliertem Git kommen

Git einrichten

- git config-Datei konfigurieren
- überprüfen ob config eingerichtet ist

```
git config --list
```

- mindestens E-mail und User-Name muss eingerichtet werden

```
# User Name erstellen
git config --global user.name "<Your Name>"

# E-Mail erstellen
git config --global user.email "<your email address>"
```

- optional direkt immer `main` als Haupt-Branch festlegen

```
git config --global init.defaultBranch <name>
```

Git Repository initialisieren

- dieser Befehl initialisiert das Git Repo mit dem Haupt-Branch 'master'.

```
git init
```

- um den Namen nachträglich zu 'main' zu ändern

```
git branch -m main
```

- um direkt bei der Initialisierung den Hauptbranch `main` zu nennen:

```
git init -b main
```

- wenn Git Repo initialisiert kann der status abgefragt werden

```
git status
```

Dateien tracken bzw. zu Staging hinzufügen

- Datei im Arbeitsverzeichnis anlegen und mittels Git command zu Staging hinzufügen

```
git add <datei.endung_der_datei>

# oder alle Dateien auf einmal
git add .
```

- Datei ggf. wieder unstagen

```
git rm --cached <file_name>
```

Der erste Commit

- mit einem Commit werden alle Dateien, welche sich gerade im Staging befinden in das lokale Repository (Storage), mit dem aktuellen Staging Zustand, aufgenommen (Snapshot des aktuellen Zustands)

```
git commit -m "<Deine Commit Message>"
```

Der zweite Commit

- lege eine neue Datei Deiner Wahl an und füge diese zum Staging hinzu mit `git add` und commite anschließend diesenneuen Stand Deines Arbeitsverzeichnisses

`git log`

- git log gibt eine Übersicht über alle commits mit den dazugehörigen Daten
- optional nimmt `git log` die `--oneline` Flag um die Übersicht kürzer anzuzeigen

```
git log --oneline
```

Reise in die Vergangenheit

- um einen vergangen Snapshot bzw. den Zustand des Git-Projektes anzuschauen:

```
git checkout <hash/versionsnummer>
```

- dort kann herumexperimentiert werden und verschiedene Dinge getestet werden ohne das Hauptprojekt zu verändern
- um wieder in das Hauptprojekt bzw. die Gegenwart zurückzukehren

```
git checkout main
# oder
git checkout -
```

ACHTUNG

- manchmal möchtest Du Änderungen verwerfen und zu einem früheren Zeitpunkt Deines Projetes zurückkehren, nicht nur um zu schauen wie es zu dem Zeitpunkt aussah, sondern um von dem Zeitpunkt aus neu zu starten

```
git reset --hard <commit-hash>
```

- wenn die Änderungen, die danach gemacht wurden nicht gelöscht werden sollen, dann wird stattdessen aus dem vergangen commit ein neuer Branch abgeleitet und nicht resetet

Branching

durch Branching wird eine Kopie des aktuellen Zustands des Haupt-Branch (hier: main) erstellt, z.B. feature-branch. In diese Feature-Branch wird nun eine neue Funktionalität des Projektes entwickelt

1. neuen branch erstellen:

   ```
   git branch <branch_name>
   ```

2. in den neuen branch wechseln

   ```
   git checkout <branch_name>
   ```

   oder besser und neuer

   ```
   git switch <branch_name>
   ```

oder

Branch anlegen und direkt in den neuen Branch wechseln

```
git switch –c <new_branch_name>

# oder mit checkout
git checkout –b <new_branch_name>
```