## Kurzbeschreibung:

GIT ist ein Versionierungstool bzw ein DVCS (Distributed Version Control System)

Das heißt, dass jeder Anwender eines Repositorys eine komplette Kopie des ganzen Produkts hat. Dadurch wird sichergestellt, das keine Version verloren geht. Denn sollte der Server mit dem Repository abstürzen, kann man den Stand durch den der User wiederherstellen.

Somit eignet sich GIT insbesondere für Teams oder ganze Unternehmen, kann aber auch als einzelne Person von Vorteil sein. GIT hat seine größte Stärke somit in Textdokumenten wie Code oder ganz nomalen Word Dateien.

## GIT einrichten mit GITHUB etc:

1. Melde dich bei GITHUB an. Das ist eine Internetseite, die es ermöglicht öffentlich zugängliche Repositorys anzulegen und zu verwalten. Benötigt wird dafür ein Username und eine Emailadresse.
2. Installiere GIT und öffne GITBASH (einfach unten in Suchleiste eingeben)
3. Nun Username und email (zwingend die gleichen wie in GITHUB angeben)
   1. git config --global user.name "Dein Name"
   2. git config --global user.email [deine@email.de](mailto:deine@email.de)
4. Prüfen ob alles passt
   1. git config –list
5. Nun verknüpfe dein lokales GIT mit GITHUB und ssh
   1. ssh-keygen -t ed25519 -C [deine@email.de](mailto:deine@email.de)
   2. enter drücken
   3. enter drücken
   4. cat ~/.ssh/id\_ed25519.pub
   5. alles inklusive mail kopieren
6. In GITHUB auf Profil > Einstellungen > SSH.. > add new SSH > Text einfügen und speichern
7. In GITBASH prüfen
   1. ssh -T [git@github.com](mailto:git@github.com)
   2. „yes“ eingeben

## Erstes Repo bauen und in GITHUB anzeigen lassen

Im allgemeinen möchte man mit GIT nur bestimmte Dateien tracken. Deshalb benötigen wir dafür einen Ordner in dem alle Dateien liegen. Darin werden wir nun unser Repo also einen .git Ordner legen. Das heißt auch, dass nur Dateien die in dem Ordner liegen getrackt werden können.

1. Optionen:
   1. Ziehe den Ordner per Drag and Drop in GITBASH
   2. cd dateipfad
2. git init
3. Im Ordner oben bei Ansicht ausgeblendete Elemente anzeigen und prüfen ob ein .git Ordner erscheint
4. git add . // „.“ Fügt alles aus dem Ordner zu GIT hinzu
5. git commit -m "Notiz" // erster Commit fügt den Ordner nun lokal zur Datenbank
6. Prüfe ob alles hinzugefügt wurde
   1. cd dateipfad
   2. ls -a //list files all zeigt also alle Dateien aus dem Repo

Jetzt erstellen wir das Repo in GITHUB

1. In GITHUB oben rechts auf „+“ dann „New repository“
2. Namen des Projektes angeben. Evtl selben wie den Ordnernamen
3. Privat oder public
4. Das README-Zeug nicht ändern
5. Create Repository
6. In GITBASH:
   1. git remote add origin [git@github.com:deinBenutzername/dein-repo-name.git](mailto:git@github.com:deinBenutzername/dein-repo-name.git)

Erster PUSH

1. git branch -M main
2. git push -u origin main // das -u verlinkt nun den lokalen Branch mit GITHUB

// In Zukunft reicht es git push zu schreiben

## Wichtige Begriffe:

Repository: Gemeinsamer Ordner mit dem Fortschritt des Projekts

Modified: Dateien die geändert, aber noch nicht in lokaler Datenbank eingecheckt wurden.

Staged: geänderte Datei, die für den Commit vorgemerkt wurde

Comitted: Daten in lokaler Datenbank

Push: Dateien auf Remote Repository (GITHUB) geschoben

## Arbeitsablauf?

1. Ändere eine Datei (im Verzeichnisbaum)
2. Stelle selektiv eine Änderung in Staging Area um sie später zu comitten
3. Commit: Snapshot von Datei in Staging Area wird in GIT-Verzeichnis aufgenommen

Das heißt: wenn eine Datei verändert wurde, ist sie erst modified.

1. git pull //Nur nötig um Dateien vom Remote zu ziehen
2. git add . //bzw ein anderer add Befehl
3. git status //zeigt: neue/geaänderte Dateien, offene Commits
4. git commit -m „sinnvolle Notiz“ //staged
5. git push //pushed die Dateien auf das GITHUB-Repo
6. git log //zeigt commit-history

Sollte git in einer Umgebung verwendet werden, die GIT bereits integriert, findet man die einzelnen Befehle idR als Symbole recht schnell über das git-Symbol.

## GIT-Befehle im Überblick:

|  |  |
| --- | --- |
| Befehl | Beschreibung |
| git init | Erstellt ein neues Git-Repository im aktuellen Verzeichnis |
| git clone <url> | Klont ein entferntes Repository (z. B. von GitHub) lokal |
| git status | Zeigt den aktuellen Status der Arbeitskopie (z. B. Änderungen, Staging-Bereich) |
| git add <datei> | Fügt eine Datei dem Staging-Bereich hinzu |
| git add . | Fügt alle Änderungen im Verzeichnis dem Staging-Bereich hinzu |
| git commit -m "Nachricht" | Speichert die Änderungen mit einer Commit-Nachricht |
| git log | Zeigt eine Historie aller Commits |
| git diff | Zeigt die Unterschiede zwischen Dateien/Commits |
| git branch | Zeigt alle lokalen Branches |
| git branch <name> | Erstellt einen neuen Branch |
| git checkout <name> | Wechselt zu einem anderen Branch |
| git checkout -b <name> | Erstellt und wechselt zu einem neuen Branch |
| git merge <branch> | Führt den angegebenen Branch in den aktuellen zusammen |
| git pull | Holt neue Änderungen vom Remote-Repository und merged sie |
| git push | Schickt lokale Commits ins Remote-Repository |
| git remote -v | Zeigt die verbundenen Remote-Repositories |
| git reset --hard <commit> | Setzt das Projekt hart auf einen früheren Commit zurück |
| git stash | Speichert ungesicherte Änderungen zwischen |
| git stash pop | Stellt gespeicherte Änderungen wieder her |
| git rm <datei> | Entfernt eine Datei aus dem Arbeitsverzeichnis und dem Staging-Bereich |
| git config --global user.name "Name" | Setzt den globalen Git-Benutzernamen |
| git config --global user.email "E-Mail" | Setzt die globale Git-E-Mail-Adresse |