Handout: Git, GitHub / GitLab

# Zusammenfassung

* Git ist ein Versionskontrollsystem: mit Git lassen sich mehrere Versionen eines Projektes speichern und verwalten.
* Git führt einen Log aller Änderungen eines Projektes, in dem steht, wer wann was geändert hat.
* GitHub / GitLab sind Online-Plattformen, die Git-Repositories hosten

## Wichtige Befehle

|  |  |
| --- | --- |
| Befehl | Erklärung |
| git help | Gibt Hilfe (Befehlsübersicht) zu Git aus |
| git init | Initialisiert ein neues Git-Repository im Arbeitsverzeichnis |
| git add <dateiname> | Fügt die Datei zur Staging-Area hinzu. git add . fügt alle Änderungen hinzu |
| git commit -m “Commit-Nachricht” | Speichert die gestagten Änderungen dauerhaft mit einer Commit-Nachricht |
| git status | Zeigt den aktuellen Zustand des Arbeitsverzeichnisses (geänderte, gestagte Dateien etc.) |
| git log | Zeigt den Commit-Log |
| git diff | Zeigt Änderungen zwischen Arbeitsverzeichnis und letztem Commit |
| git branch | Listet alle Branches auf |
| git branch <name> | Erstellt einen neuen Branch |
| git checkout <branch> | Wechselt zu einem anderen Branch |
| git checkout <Commit-ID> | Wechselt zu einem anderen Commit |
| git merge <branch> | Führt den angegebenen Branch in den aktuellen Branch ein |
| git clone <URL> | Kopiert ein bestehendes Repository (z. B. von GitHub) auf den lokalen Rechner |
| git pull | Holt und mergt Änderungen vom Remote-Repository. |
| git fetch | Holt neue Änderungen vom Remote-Repository, ohne sie automatisch zu mergen |
| git push | Lädt lokale Commits ins Remote-Repository hoch. |
| git remote -v | Zeigt die konfigurierten Remote-Repositories. |
| git reset [--soft|--mixed|--hard] <commit> | Setzt den aktuellen Branch auf einen früheren Commit zurück:  --soft: behält Änderungen gestagt  --mixed: entfernt Staging, behält Dateiänderungen  --hard: verwirft alle Änderungen  Vorsicht: Nicht wiederherstellbarer Datenverlust |

Git-Einsteigerübung

**Aufgaben:**

1. Erstelle ein Verzeichnis und initialisierte Git in diesem Verzeichnis.
2. Erstelle eine Textdatei, schreibe einen Satz in diese Datei und commite diese Datei mit einer sinnvollen Nachricht.
3. Schreibe zusätzlich deinen Namen in die Datei und commite diese Datei erneut mit einer sinnvollen Nachricht
4. Stelle die erste Version der Datei, die deinen Namen nicht enthält, wiederher.
5. Wechsel zurück zur der Version, die deinen Namen enthält.
6. Erstelle einen neuen Branch, wechsel zu diesem Branch, erstelle eine neue Datei in diesem Branch und commite diese Datei mit einer sinnvollen Nachricht
7. Wechsel zu dem vorherigen Branch (vermutlich master oder main) und merge die im vorherigen Schritt erstellte Branch mit der aktuellen Branch.
8. Clone das folgende Repository von GitLab und lies die README.md im heruntergeladenen Ordner:

<https://gitlab.cvh-server.de/shoffmann/git-uebung.git>

# Musterlösung

# 1

mkdir git-uebung

cd git-uebung

git init

# 2

echo "Das ist mein erstes Git-Projekt." > info.txt

git add info.txt

git commit -m "Initial commit: info.txt hinzugefügt"

# 3

echo "Name: Sebastian Hoffmann" >> info.txt

git add info.txt

git commit -m "Name hinzugefügt"

#4

git log

git checkout <Commit-ID des ersten Commits>

more info.txt

*"Das ist mein erstes Git-Projekt."*

git checkout master

more info.txt

*"Das ist mein erstes Git-Projekt."*

*"Name: Sebastian Hoffmann"*

# 5

git checkout -b neuer-feature-branch

echo "Dies ist eine neue Funktion." > feature.txt

git add feature.txt

git commit -m "feature.txt hinzugefügt"

# 5

git checkout master

git merge neuer-feature-branch

# 6

git clone [**https://gitlab.cvh-server.de/shoffmann/git-uebung.git**](https://gitlab.cvh-server.de/shoffmann/git-uebung.git)

cd git-uebung

more README.md