Git: Basic Commands

Hier sind die grundlegenden Git-Befehle, die dir helfen, eine solide Basis im Umgang mit Git zu bekommen:

### **1. Repository erstellen oder klonen**

* **git init**: Erstellt ein neues lokales Git-Repository im aktuellen Ordner.
* **git clone <URL>**: Klont ein bestehendes Repository von einer URL (z.B. von GitHub) in einen lokalen Ordner.

### **2. Änderungen nachverfolgen**

* **git status**: Zeigt den Status von Dateien im Arbeitsverzeichnis an – welche Dateien geändert wurden, bereit zum Commit sind oder noch nicht.
* **git add <Datei/Ordner>**: Fügt Dateien dem Staging-Bereich hinzu, um sie für den nächsten Commit vorzubereiten. git add . fügt alle Änderungen hinzu.
* **git commit -m "Nachricht"**: Speichert die Änderungen im Repository mit einer beschreibenden Nachricht.

### **3. Commit-Historie anzeigen**

* **git log**: Zeigt eine Liste der letzten Commits im aktuellen Branch an, zusammen mit Details wie Commit-Nachricht, Autor und Datum.
* **git log --oneline**: Zeigt die Commit-Historie in einer kürzeren Übersicht an – praktisch für schnelle Überblicke.

### **4. Branching und Branches wechseln**

* **git branch**: Listet alle Branches auf und zeigt an, auf welchem du dich gerade befindest.
* **git branch <Name>**: Erstellt einen neuen Branch.
* **git checkout <Branch>**: Wechselt zu einem anderen Branch.
* **git checkout -b <Name>**: Erstellt einen neuen Branch und wechselt direkt darauf.

### **5. Zusammenführen und Mergen von Branches**

* **git merge <Branch>**: Führt den angegebenen Branch mit dem aktuellen Branch zusammen.
* **git rebase <Branch>**: Übernimmt die Commits des anderen Branches und fügt sie an die aktuelle Historie an. (Wird oft verwendet, um eine lineare Historie zu erhalten.)

### **6. Änderungen rückgängig machen**

* **git reset**: Macht Änderungen im Staging-Bereich rückgängig oder setzt Commits zurück (Achtung: kann Daten verlieren).
* **git revert <Commit-ID>**: Erstellt einen neuen Commit, der einen früheren Commit rückgängig macht. Sicherer als reset.

### **7. Stashes für ungespeicherte Änderungen**

* **git stash**: Speichert aktuelle, aber noch nicht committete Änderungen temporär, um mit einem sauberen Arbeitsbereich zu arbeiten.
* **git stash pop**: Ruft die letzten gestashten Änderungen wieder auf und entfernt sie aus dem Stash.

### **8. Remote-Repositories**

* **git remote add <Name> <URL>**: Verbindet das lokale Repository mit einem Remote-Repository.
* **git fetch <Remote>**: Lädt Änderungen aus dem Remote-Repository, ohne sie zu mergen.
* **git pull <Remote>**: Holt Änderungen vom Remote und merged sie direkt in den aktuellen Branch.
* **git push <Remote> <Branch>**: Überträgt die Commits des lokalen Branches in das Remote-Repository.

### **9. Dateien aus der Versionskontrolle entfernen**

* **git rm <Datei>**: Entfernt eine Datei aus der Versionskontrolle und dem Arbeitsverzeichnis.
* **git rm --cached <Datei>**: Entfernt eine Datei nur aus der Versionskontrolle (nicht aus dem Arbeitsverzeichnis), was nützlich ist, wenn man die Datei weiterhin lokal behalten möchte, sie aber nicht versionieren will.

Diese Befehle decken die meisten grundlegenden Git-Aktionen ab und sind eine gute Grundlage, um Git effektiv zu nutzen!