Git: Advanced Commands

Hier sind einige fortgeschrittene Git-Aktionen und Befehle, die dir helfen können, komplexere Aufgaben in der Versionskontrolle zu meistern:

### **1. Interaktives Rebase (git rebase -i)**

* Mit git rebase -i HEAD~N (wobei N die Anzahl der letzten Commits angibt) kannst du mehrere Commits in der Geschichte ändern, zusammenführen, neu ordnen oder sogar löschen. Super nützlich, um die Commit-Historie vor dem Pushen aufzuräumen!

### **2. Stashes mit Nachrichten und Teilauswahl (git stash save "Nachricht")**

* Mit git stash push -m "Nachricht" speicherst du Änderungen im Stash mit einer Beschreibung. Mit git stash pop holst du sie zurück. Du kannst außerdem mit git stash -p nur bestimmte Teile der Änderungen stashen.

### **3. Reflog für verlorene Commits (git reflog)**

* Wenn du einen Commit verloren hast, hilft dir git reflog, um alle Aktionen (wie Checkouts, Resets usw.) aufzulisten. So kannst du sogar gelöschte oder zurückgesetzte Commits wiederherstellen.

### **4. Commit-Bereich ändern (git commit --amend)**

* Mit git commit --amend kannst du den letzten Commit anpassen, etwa um kleine Änderungen hinzuzufügen oder den Commit-Text zu korrigieren, ohne einen neuen Commit zu erstellen.

### **5. Cherry-Picking (git cherry-pick <Commit-ID>)**

* git cherry-pick <Commit-ID> erlaubt es dir, einen spezifischen Commit aus einer anderen Branch in die aktuelle Branch zu übernehmen. Das ist nützlich, um nur ausgewählte Änderungen zu übernehmen.

### **6. Submodules einrichten und verwalten (git submodule)**

* Wenn du ein Projekt hast, das ein anderes Repository als Teilprojekt benötigt, kannst du es mit git submodule add <URL> hinzufügen. Die Submodule werden dann als eigene Repositories behandelt, die du bei Bedarf aktualisieren oder festsetzen kannst.

### **7. Squash-Merges und Commits zusammenführen**

* Bei git rebase -i kannst du mehrere Commits zusammenfassen (squashen). Das ist nützlich, um die Historie kompakter zu machen, z.B. um viele kleine Commits in einem größeren zusammenzuführen.

### **8. Rebase von Branches (git rebase <Branch-Name>)**

* Anstatt zu mergen, kannst du git rebase <Branch-Name> verwenden, um die Änderungen eines anderen Branches aufzunehmen und die eigene Historie linear zu halten.

### **9. Revert eines gemergten Commits (git revert -m 1 <Merge-Commit-ID>)**

* Wenn du einen Merge-Commit rückgängig machen willst, kannst du das mit git revert -m 1 <Merge-Commit-ID> tun. Das erzeugt einen neuen Commit, der den gemergten Änderungen entgegenwirkt.

### **10. Custom Aliases für Git-Befehle (git config --global alias.<name> "<command>")**

* Du kannst benutzerdefinierte Aliase für häufig genutzte Befehle erstellen, z.B. git config --global alias.st status, sodass git st das gleiche wie git status ausführt.