**Git für Fortgeschrittene**

Ab ins Lager - git stash

Stash bedeutet so viel wie „Lager“ oder „Versteck“ und meint nichts anderes, als einen Stack mit ungesicherten Dateien. Arbeitet man gerade in einem Branch und muss seine Arbeit wegen eines dringenden Fehlers unter­brechen, so fügt man alle geänderten Dateien zum Index hinzu und führt danach ein ***git stash***aus. Dadurch werden alle auf dem Index befindlichen Dateien in den Stash geschrieben und der Index geleert. Nun können der Fehler gefixt und die dafür erforderlichen Dateien über­geben werden. Anschließend werden die auf dem Stash befindlichen Änderungen per ***git stash apply*** oder ***git stash pop*** wieder in den Branch zurückgeholt und die Arbeit kann weitergehen.

Es können sogar mehrere Stapel angelegt und mit Namen versehen werden. Der Befehlt ***git stash list*** gibt z. B. eine Liste mit allen Stapeln aus. Mit ***git stash clear*** wird der Stapel gelöscht.

### Kommando zurück - git revert

Stellen wir uns vor, dass ein neues Feature entwickelt und anschließend übergeben wurde. Was, wenn sich in den finalen Tests herausstellt, dass alles doch nicht so funk­tioniert, wie es sollte? Für diesen Fall gibt es das Kommando ***git revert***.

Es lässt den aktuellen Branch auf einen älteren Stand zurückfallen und schreibt für den „neuen“ Stand einen eigenen Commit. So wird die Historie erweitert und nicht in ihr herumgelöscht. Ein Beispiel: Das Kommando ***git revert HEAD*** macht den letzten Commit rückgängig und kennzeichnet dies auch mit dem Schlüsselwort „Revert“ in der Commit-Message.

### Für immer gelöscht - git reset

Mit dem Befehl ***git reset*** lässt man einen Branch, ähnlich wie bei git revert, auf einen älteren Stand zurück­fallen. Allerdings mit dem Unterschied, dass hier alle übergebenen Dateien nach der ausgewählten Übergabe unwiderruflich gelöscht und aus der Historie entfernt werden. Ein Beispiel: Das Kommando***git reset HEAD~3***geht 3 Übergaben zurück und löscht alle nachfolgenden.

Außerdem können mit dem Befehl weitere sehr nützliche Aktionen durchgeführt werden. Mit ***git reset*** werden alle Dateien aus dem Index gelöscht, aber die Arbeitskopie bleibt unangetastet. ***git reset --hard*** setzt hingegen nicht nur den Index zurück, sondern auch den Workspace in den Status des letzten Commit.

Dieser Befehl sollte nur sehr sparsam und nur auf lokalen Branches angewendet werden. Schiebt man einen mit ***reset*** bearbeiteten Branch in das Remote Repository, so kann dies schon mal zu größeren Problemen bei anderen Entwicklern führen.

### Kirschenpflücken - git cherry-pick

Experimente oder neue Funktionalitäten bearbeitet man normalerweise auf einem eigenen Branch, einem sogenannten Feature Branch. Ist die Arbeit dort abgeschlossen, so kann ein einzelner Commit herausgepickt und in einen anderen Zweig, z. B. den Master Branch, übertragen werden. Der Befehl dafür lautet dann***git cherry-pick <Hashwert>***, wobei der jeweilige Hashwert des Commit, der in den aktuellen Branch überführt werden soll, mit angegeben werden muss. Nach solchen Aktionen kann ein Feature Branch normalerweise verworfen werden.

### Geschichte neu schreiben - git rebase

Um die Änderungen aus einem Branch in einen anderen zu übernehmen, gibt es neben dem Befehl***merge***eine zweite Variante, das sogenannte ***rebase***. Das Kommando ***rebase*** ist sehr mächtig und man könnte alleine damit einige Seiten dieser Zeitschrift füllen. Dennoch soll der Sachverhalt an einem Beispiel kurz dargestellt werden.

Ein Feature Branch wurde vom Master abgezweigt und auf diesem wurden einige Commits erzeugt. Anschließend sollen die Änderungen, die inzwischen auf dem Master durgeführt wurden, nachgezogen werden. Dies erfolgt mithilfe des Befehls***git rebase master***. Dieser nimmt die eigenen Änderungen vom Branch herunter, pflegt dann die neuen Punkte aus dem Master Branch ein und packt die eigenen Änderungen wieder obenauf.

Eventuelle Konflikte können in einem interaktiven Modus behoben werden. Nach dem Beheben von Konflikten kann der Befehl rebase mit ***git rebase --continue*** fort­geführt werden.