Übungen zur Vorlesung Softwaretechnologie

- Wintersemester 2018/19 -

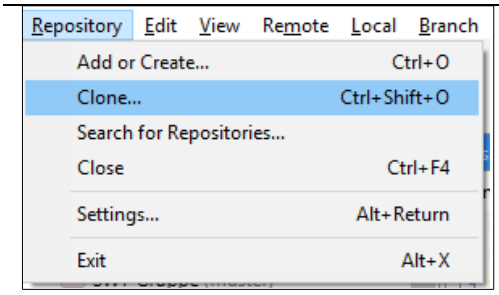
Dr. Günter Kniesel

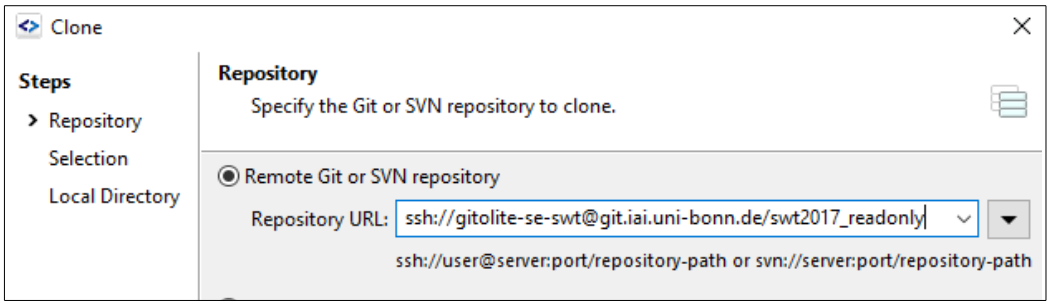
Übungsblatt 1 - Lösungen

Aufgabe 1. *Branch & merge sowie vollständig dezentralisiertes Arbeiten* (15 Punkte)

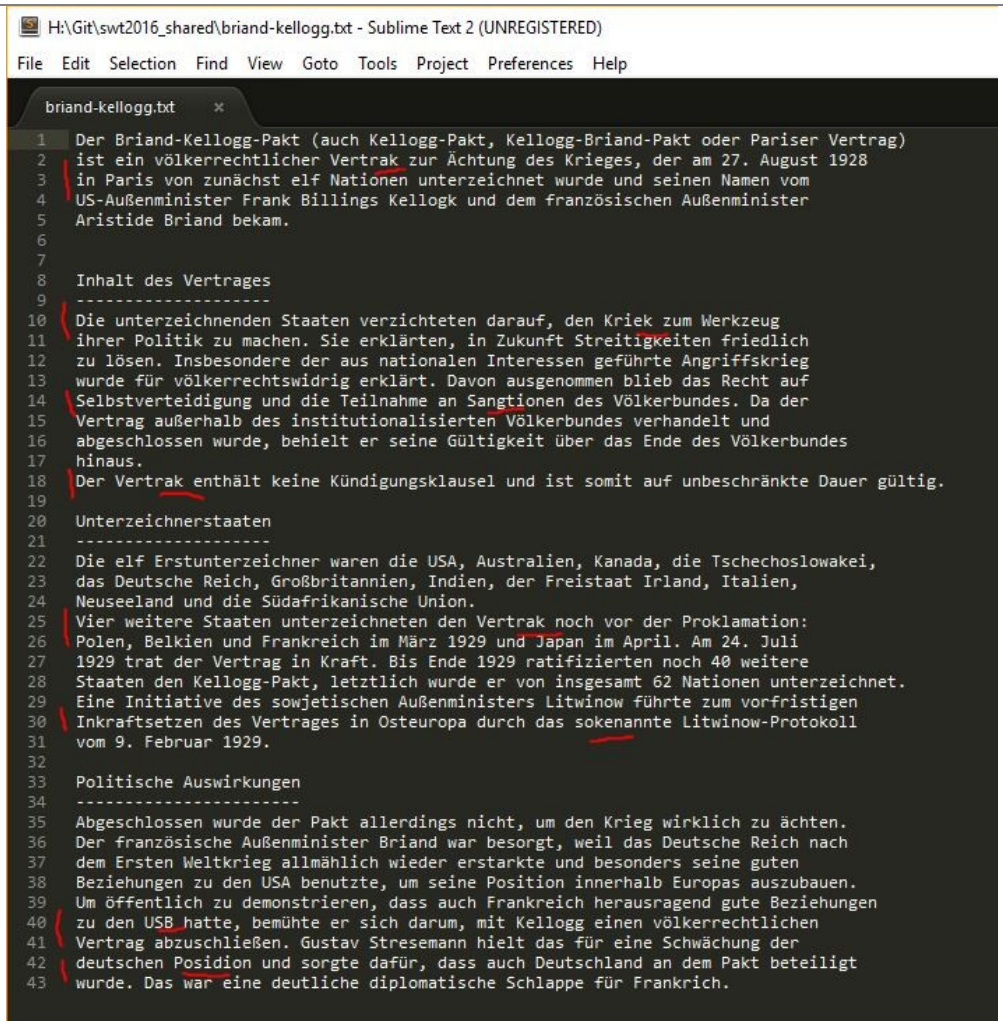
Lösungshilfe basierend auf SmartGit 8.0.2 & Windows 10

a): Klonen

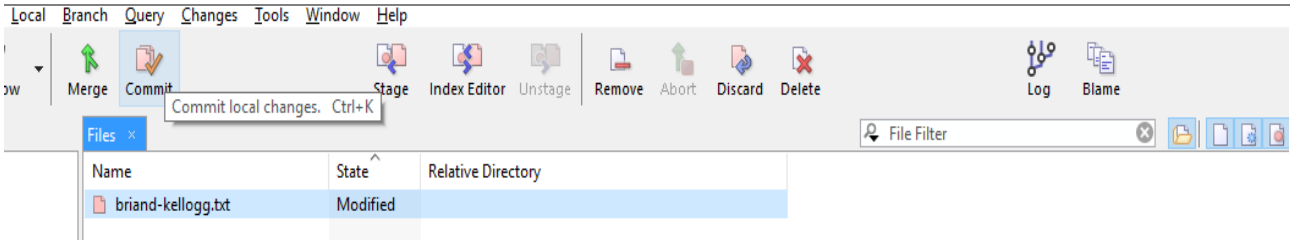




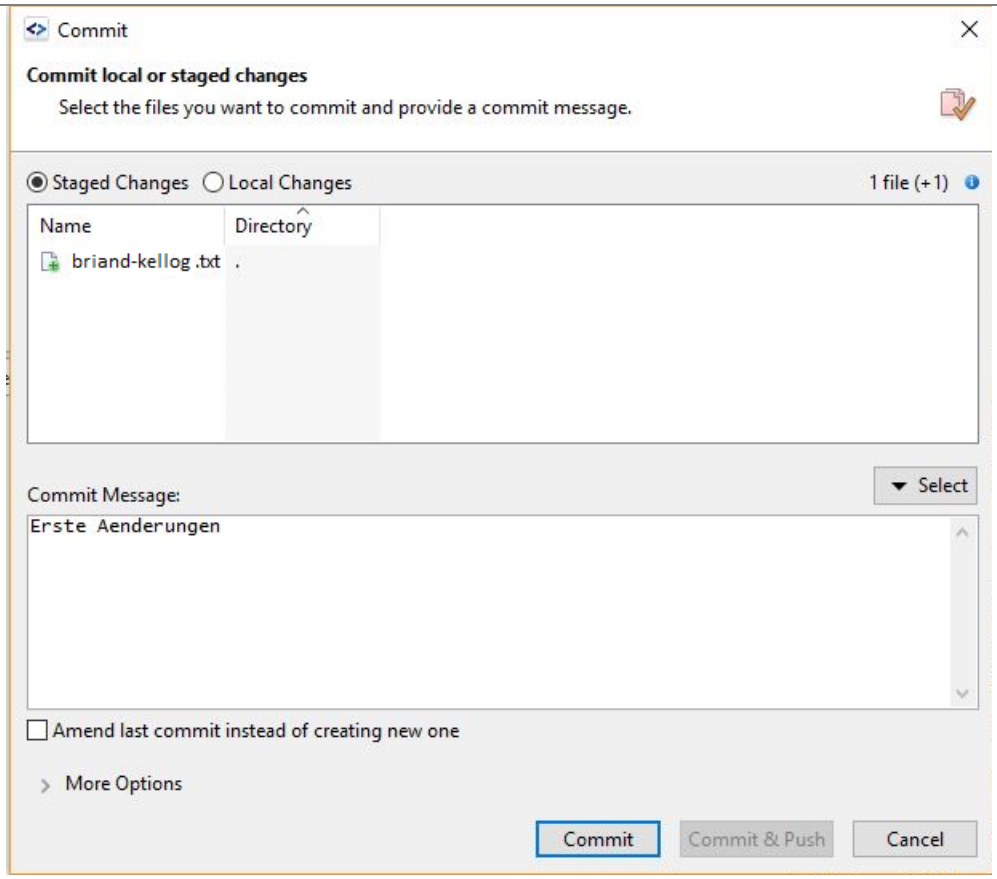
b1): Fehler in briand-kellogg.txt korrigieren (nur im „eigenen“ Abschnitt):



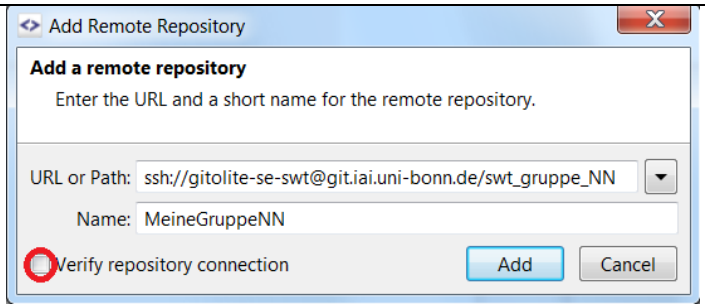
SG-Reaktion auf b1) Datei briand-kellogg.txt wird als „changed“ markiert



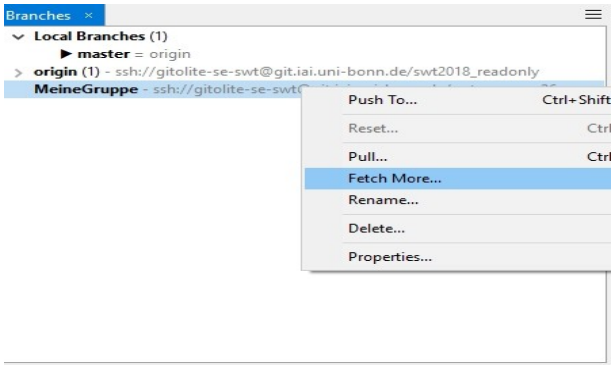
b2): Commit der Änderungen an briand-kellog.txt



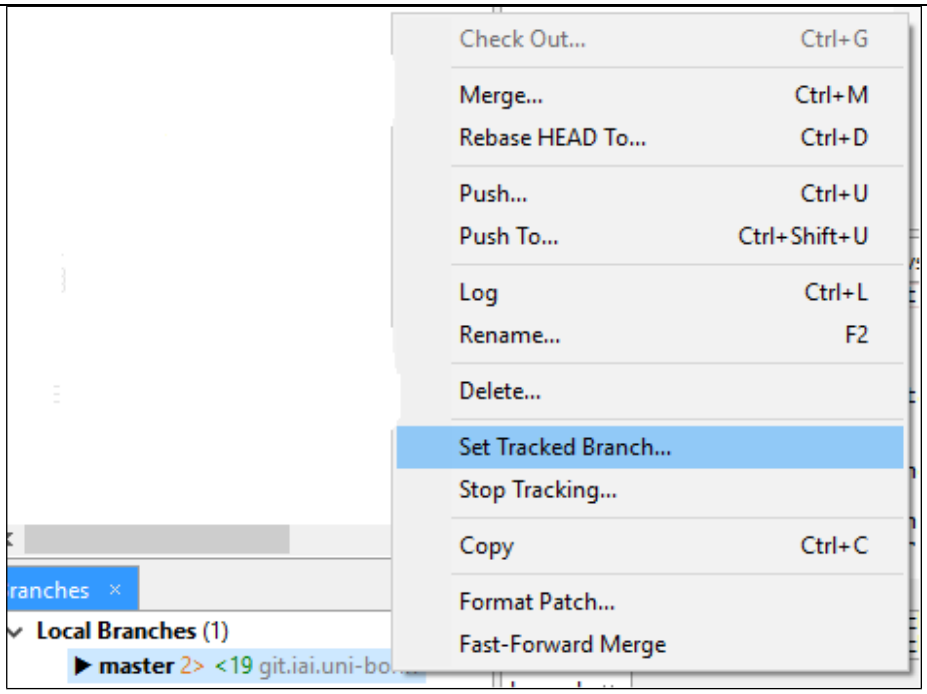
c) Remote Repository eintragen

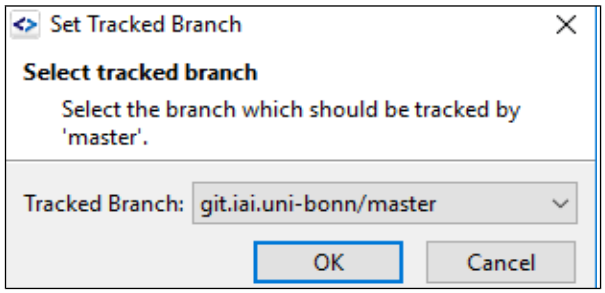


c2) Fetch aus Remote Repository

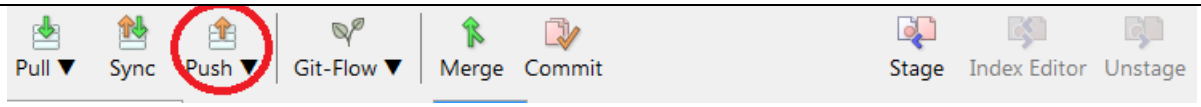


c3) Set Tracked Branch





d) Pushen

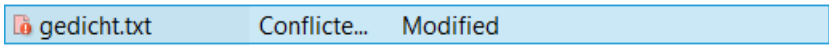


d2/3) Anmerkung zum Fehler „unrelated Histories“

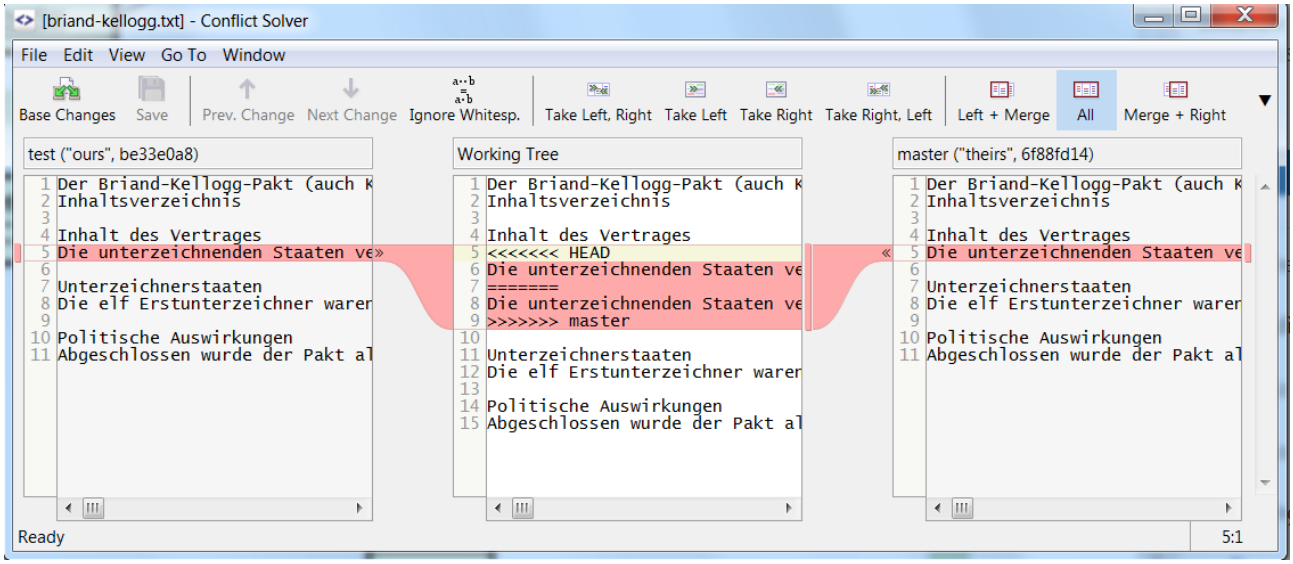
An dieser Stelle gab es oft die Fehlermeldung „`fatal: refusing to merge unrelated histories`“. Dies liegt daran, dass SmartGit nicht in der Lage ist zwei verschiedene Histories zu mergen, die keinen gemeinsamen Commit besitzen.
Um das Problem zu lösen, muss in der Git-Bash der Befehl **git pull –allow-unrelated-histories** angegeben werden.

```
Patrick ASUS@DESKTOP-815CTKR MINGW64 ~/Desktop/SWT-Tut/swt_gruppe_07 (master)
$ git pull --allow-unrelated-histories
```

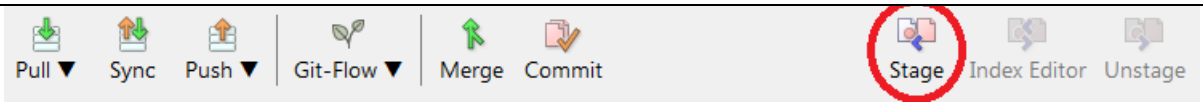
d4) Konflikt entsteht



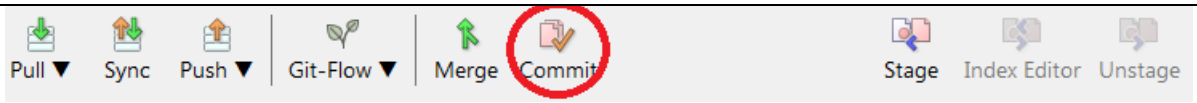
d4) Konflikte lösen



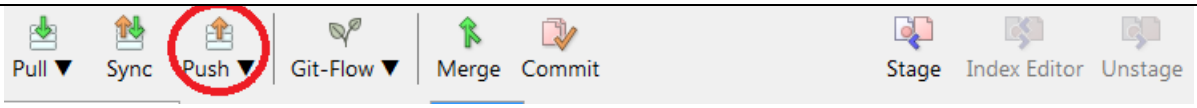
d6) Stage der Änderungen



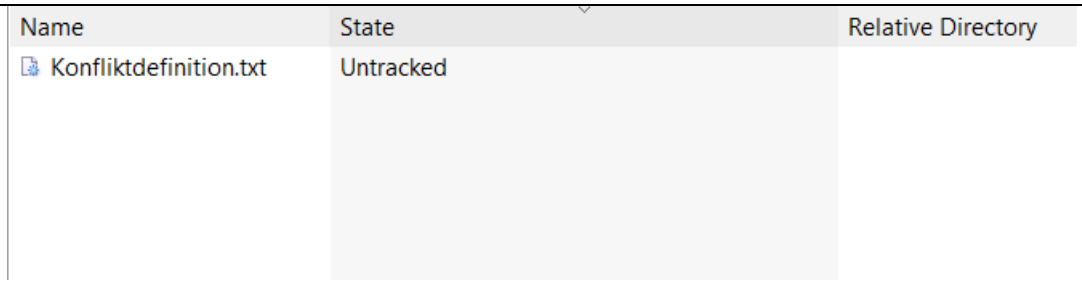
d7) Commit der Änderungen in der Stage



d8) Push des Repository



e1) Konfliktdatei erstellen



e2) Definition aufschreiben

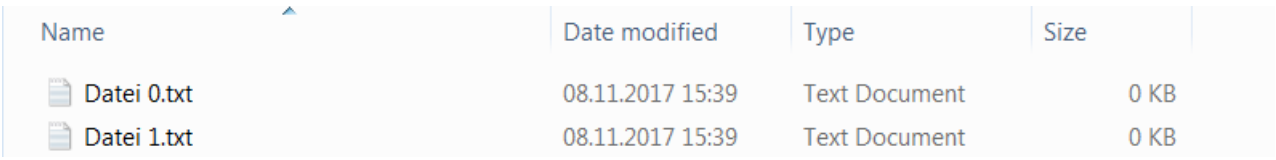
Definition III ist korrekt:
Ein Konflikt entsteht erst wenn in derselben Zeile unterschiedliche Änderungen gemacht werden

e3) Konflikte beseitigen und Ergebnisse hochladen

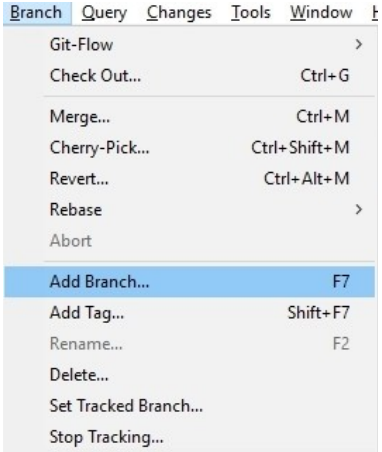
Siehe Aufgabenteil **d)** zur Beschreibung des Commit/Pushvorgangs. Darauf ist zu achten dass jeder seine eigene Definition aufschreibt und nicht die der anderen überschreibt.

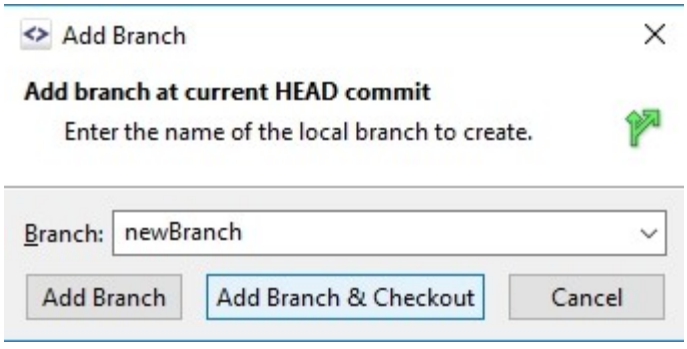
Aufgabe 2. *Dateioperationen mit Git* (11 Punkte)

a) Erzeugen eines Ordners Dateien_Vorname mit Datei0 und Datei 1

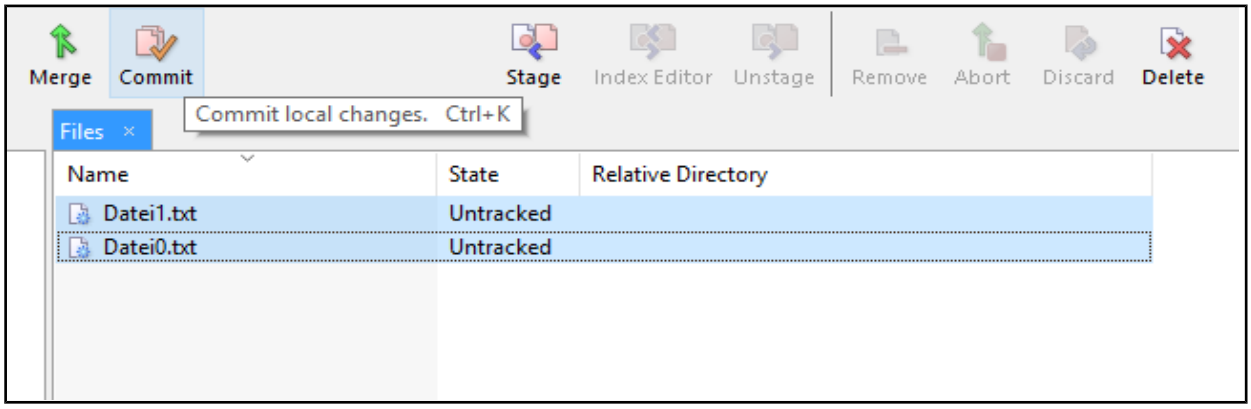


b) Erzeugen eines Branches

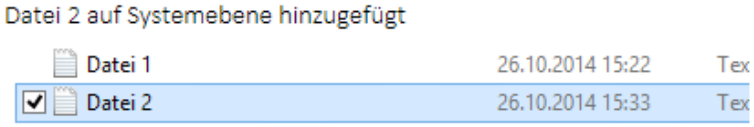




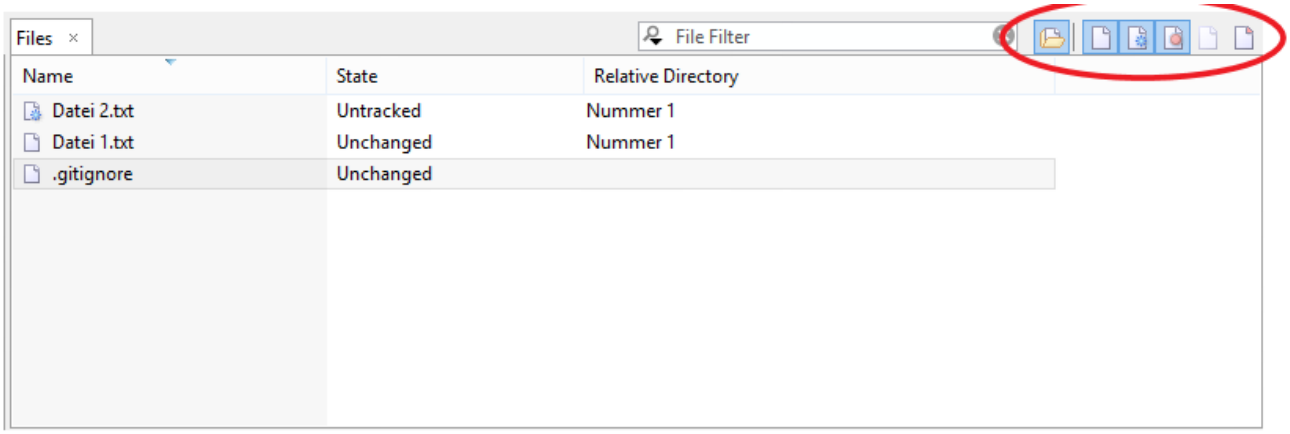
c) Commit der beiden erstellten Dateien

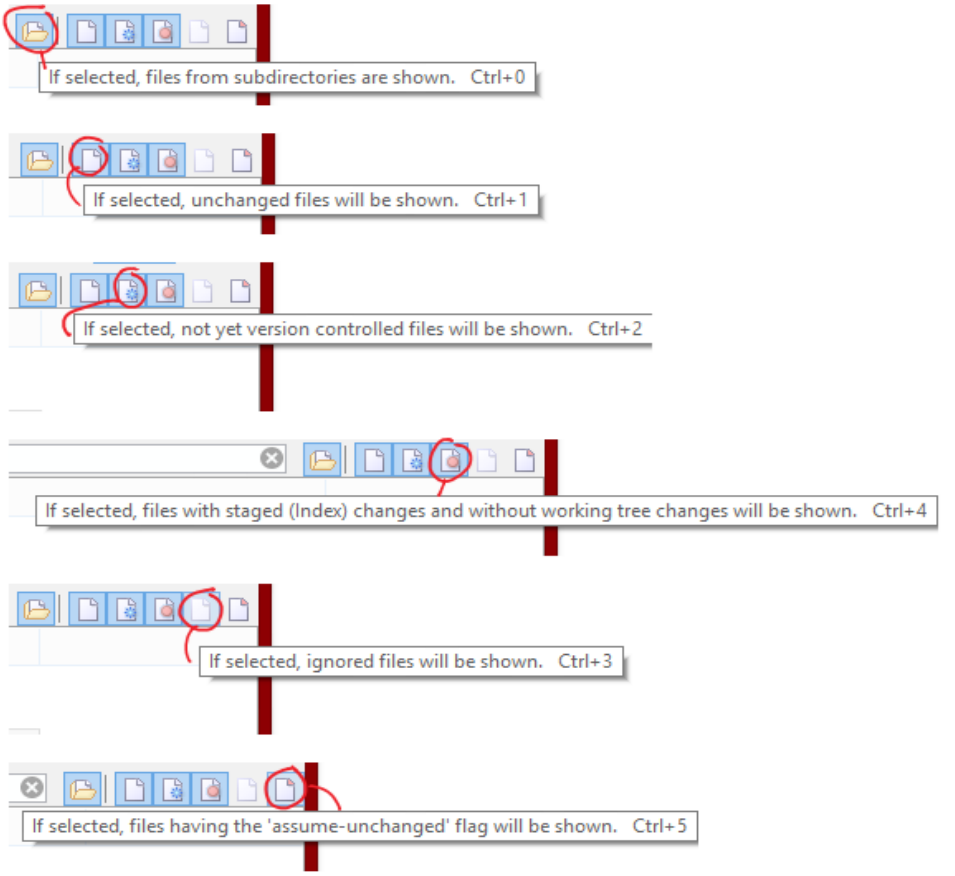


d1) Datei 2 erstellt



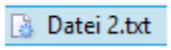
d2) Steuerung der Ansicht





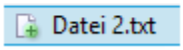
e1) Symbol einer neuen Datei

Symbol der neuen Datei:



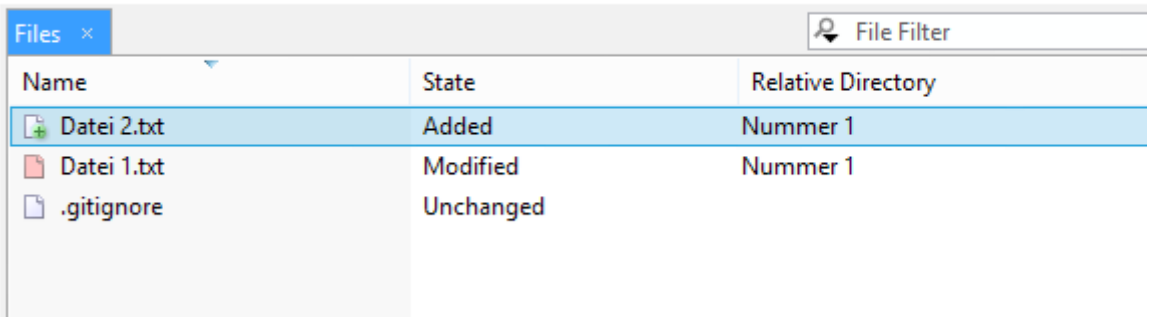
e2) Symbol nach Stage

Dateisymbol nach "Stage"



f) Symbol nach ändern von Datei 1 (bzw. Datei 0)

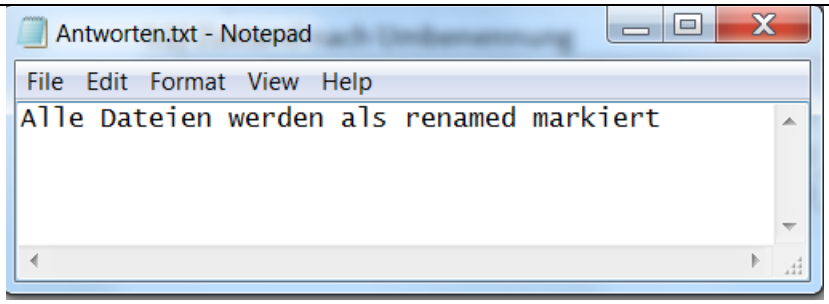
Dateisymbol nach Speichern



h1) Zustand nach Umbenennung

Name	State	Relative Directory
Null.txt	Renamed (untracked)	
Eins.txt	Renamed (untracked)	
Antworten.txt	Renamed (untracked)	

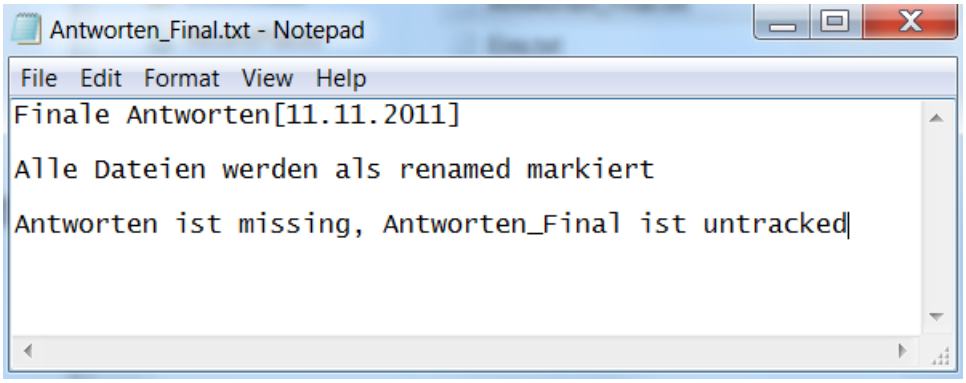
h2) Antworten in Datei



i1) Zustand nach Umbenennung und Inhaltsänderung

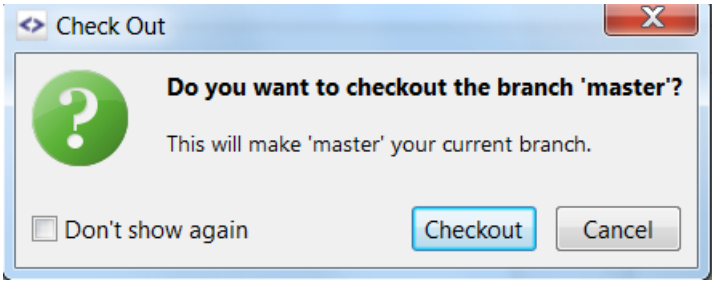
Name	State	Relative Directory
Antworten_Final.txt	Untracked	
Antworten.txt	Missing	

i2) Antworten in Datei

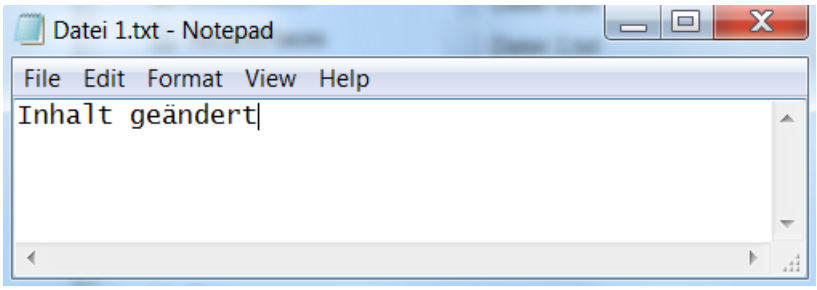


j1) Ändern des Branches

Doppelklick auf den Zielbranch, dann :

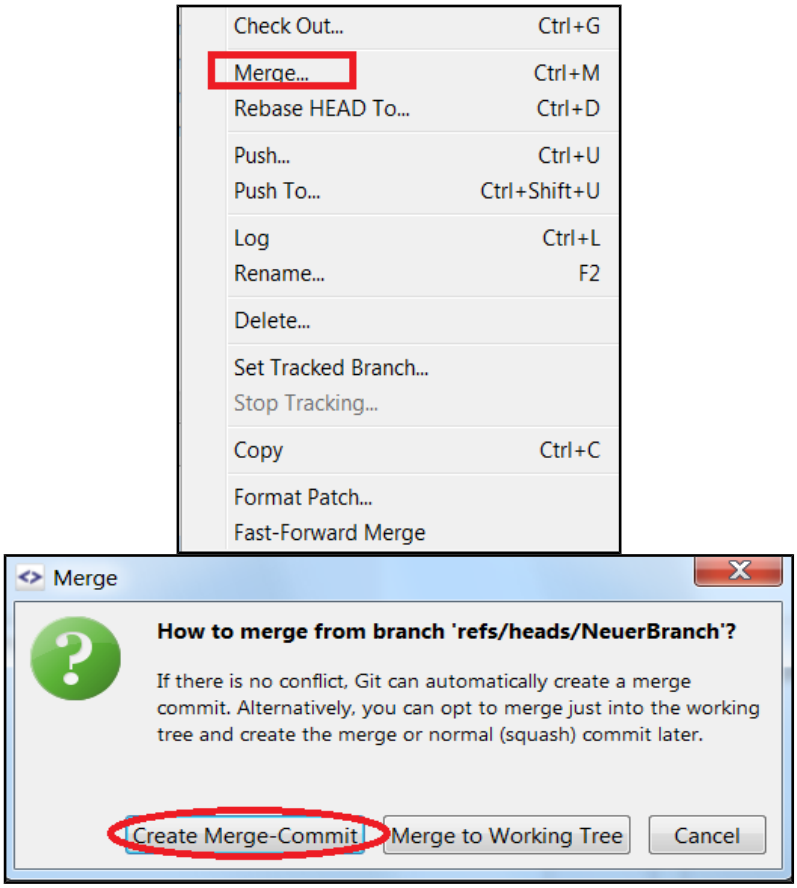


j2) Ändern von Datei 1



k) Merge der branches

Rechtsklick auf den Branch der hereingemerged werden soll


Check Out... Ctrl+G
Merge... Ctrl+M
Rebase HEAD To... Ctrl+D
Push... Ctrl+U
Push To... Ctrl+Shift+U
Log Ctrl+L
Rename... F2
Delete...
Set Tracked Branch...
Stop Tracking...
Copy Ctrl+C
Format Patch...
Fast-Forward Merge
Merge
How to merge from branch 'refs/heads/NeuerBranch'?
If there is no conflict, Git can automatically create a merge commit. Alternatively, you can opt to merge just into the working tree and create the merge or normal (squash) commit later.
Create Merge-Commit
Merge to Working Tree
Cancel