Vorbereitungen

Auf Github einen Account erstellen: https://github.com/

SourceTree downloaden: https://www.sourcetreeapp.com/

SourceTree zurücksetzen (falls nötig)

Hierfür einfach folgende beiden Ordner löschen (falls vorhanden):

C:\Users\<windows-account-name>\AppData\Local\Atlassian

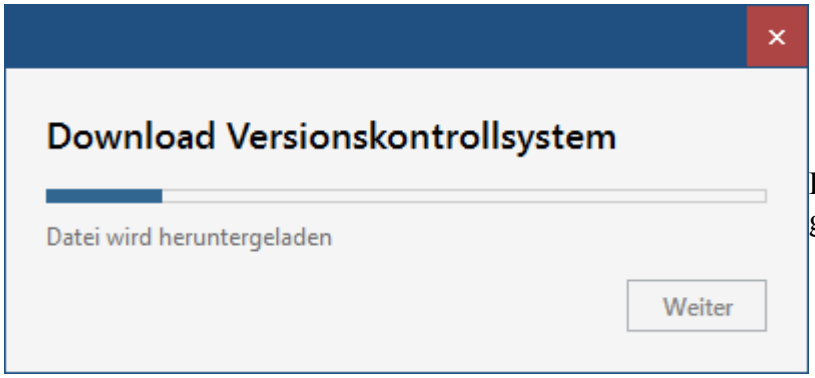
C:\Users\<windows-account-name>\Roaming\Local\Atlassian

SourceTree vorbereiten

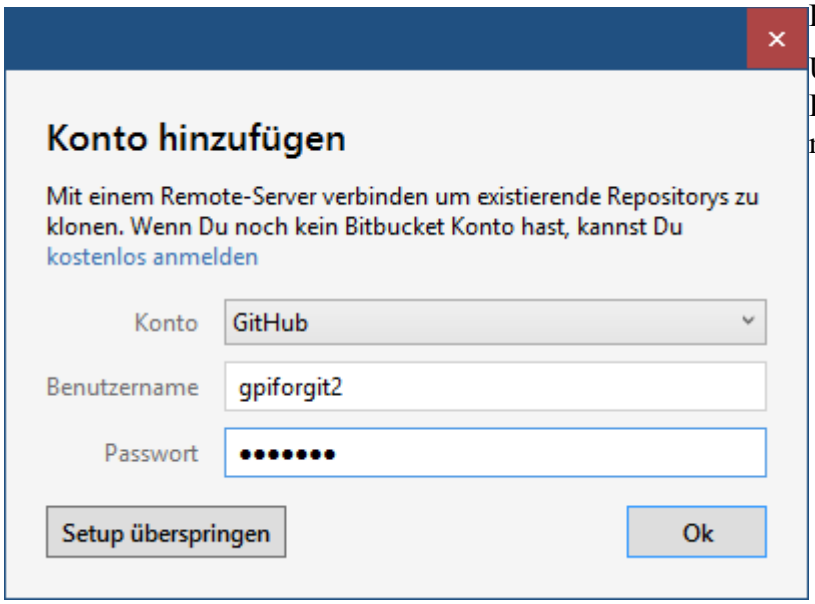
Die Installation sollte nicht weiter kompliziert sein, es ist eine typische WWF (weiter, weiter, fertig) Installation. In Anschluss CodeTree starten.

Wir werden wohl die Vereinbarung akzeptieren müssen.

Anschließend auf Fortfahren.

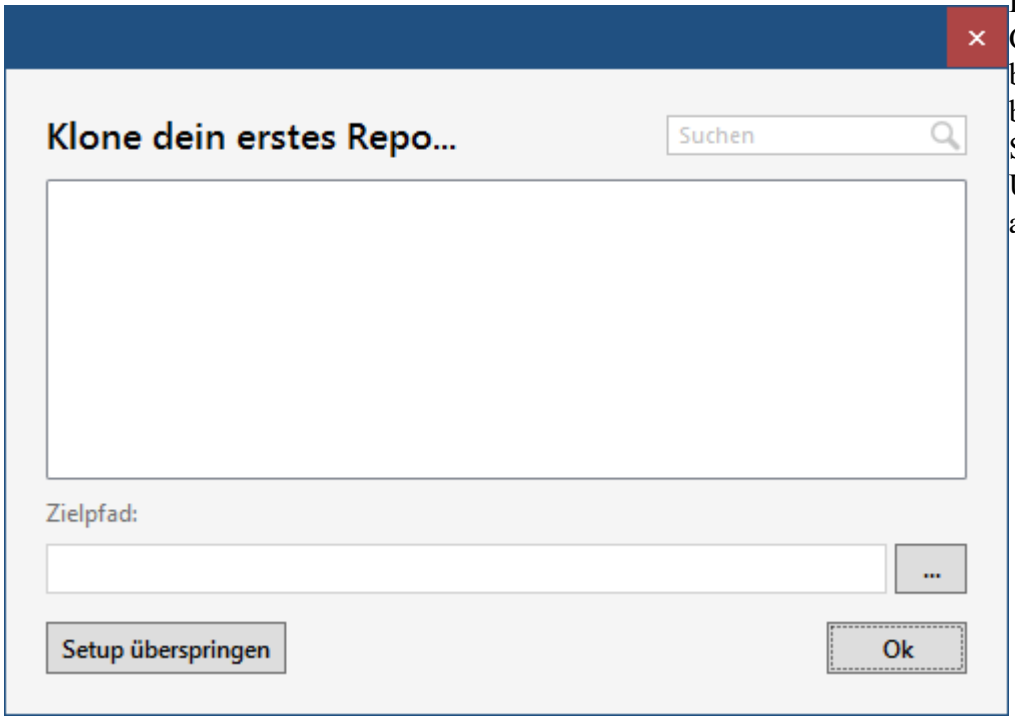


Die GIT-Engine wird automatisch gedownloaded



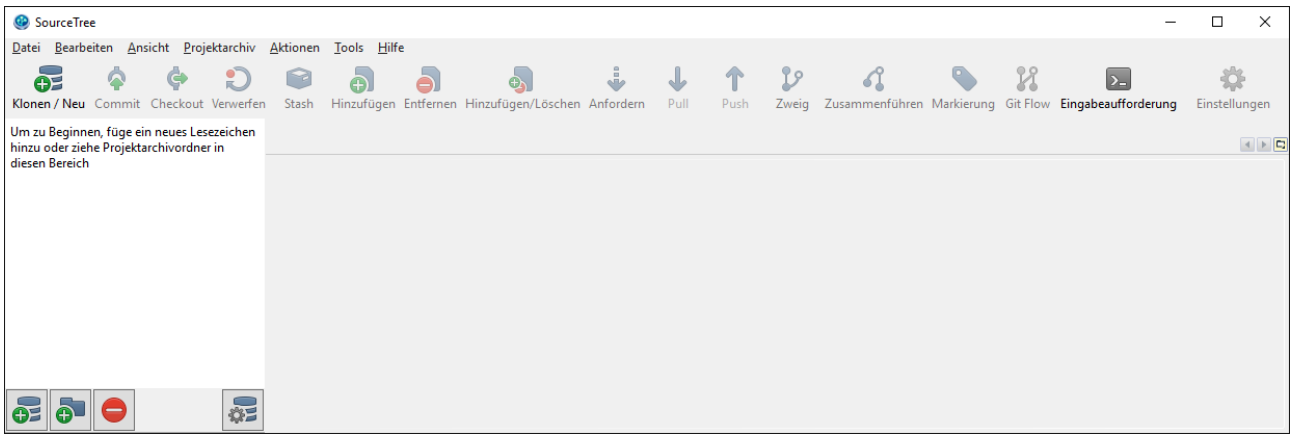
Konto GitHub auswählen

Und den Benutzer-Name (nicht Email) und Passwort eingeben und mit OK bestätigen.

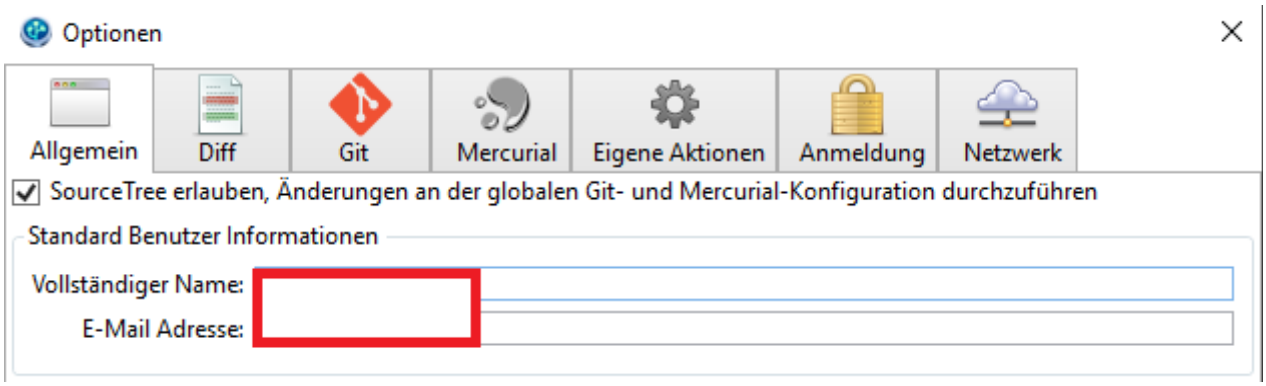


Da wir ja die CodeArchiv-Repo benutzen wollen, brechen wir das Setup mit „Setup Überspringen“ hier ab.

SoureTree sollten uns jetzt mit den Default-Bildschirm begrüßen:

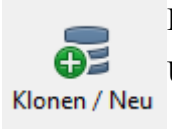


Als erstes sollten wir noch eine Einstellung vornehmen. Dazu in Menü oben Tools und Optionen auswählen.



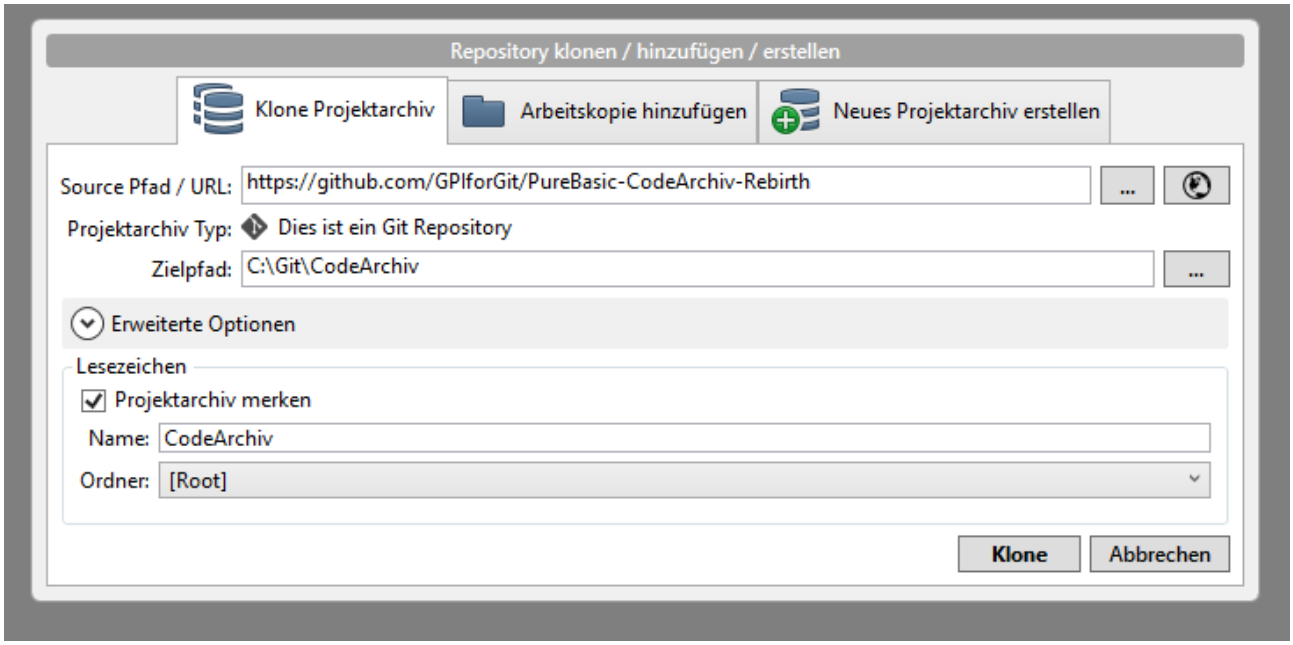
Hier müssen wir unbedingt unter Standard-Benutzer den GitHub-Benutzernamen und die E-Mail Adresse ergänzen.

Die CodeArchiv Repo holen



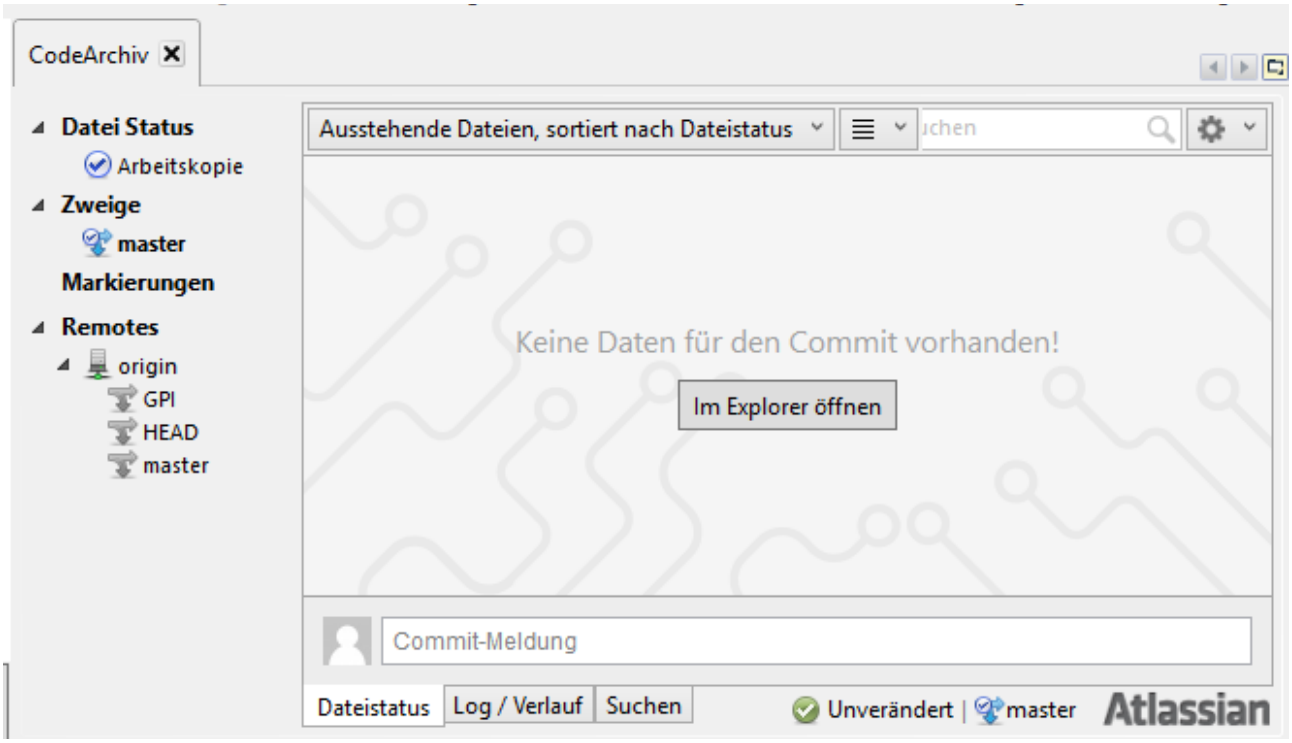
Dazu klicken wir auf das „Klonen / Neu“-Icon aus der Toolbar des Hauptfensters.

Und füllen folgendes Fenster entsprechend aus:



Als URL einfach die GitHub-Seite: https://github.com/GPIforGit/PureBasic-CodeArchiv-Rebirth

Ein Klick auf Klone und schon wird die Repro kopiert.



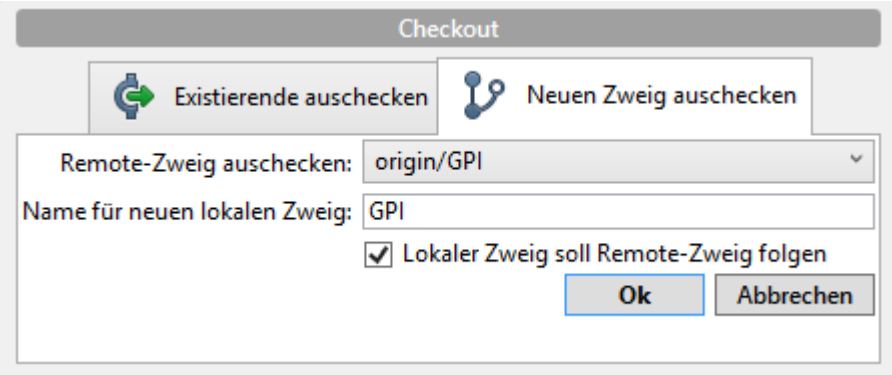
Das ganze sollte ungefähr so aussehen. Links sind die verschiedene Zweige. Aktuell ist der „master“-Zweig aktiv, erkennbar an den fetten Schriftzug. In der Remotes kann man die auf GitHub vorhandenen Zweige sehen. Interessant sind hier eigentlich nur GPI – das ist mein Zweig und master – das ist quasi der veröffentlichte Zweig.

Per doppelklick kann man hier in die verschiedenen Zweige reingehen und einen „Checkout“ machen – keine Ahnung, warum das so heißt, sinnvoller wäre wohl Checkin. Es wird dann der entsprechende Zweig aktiv. Die Daten auf der HDD (bei mir c:\git\CodeArchiv) werden automatisch so geändert, das sie den Zweig entsprechen.

Und rechts neben den „Atlassian“ Schriftzug sind noch zwei wichtige Infos. Einmal sieht man hier nochmals den aktiven Zweig und die Meldung „Unverändert“. Das bedeutet, das wir noch keine Dateien hinzugefügt, gelöscht oder geändert haben. Solange hier nicht Unverändert steht, sollte man nicht auf einen anderen Zweig klicken.

Wenn man einen Zweig einfach anklickt, wird der „Baum“ angezeigt und wo sich der Punkt gerade befindet. Die Beschreibungstexte sind etwas willkürlich. Das liegt daran, das ich da vermerke, wie weit ich gekommen bin und wo ich in Forum weitermachen sollte.

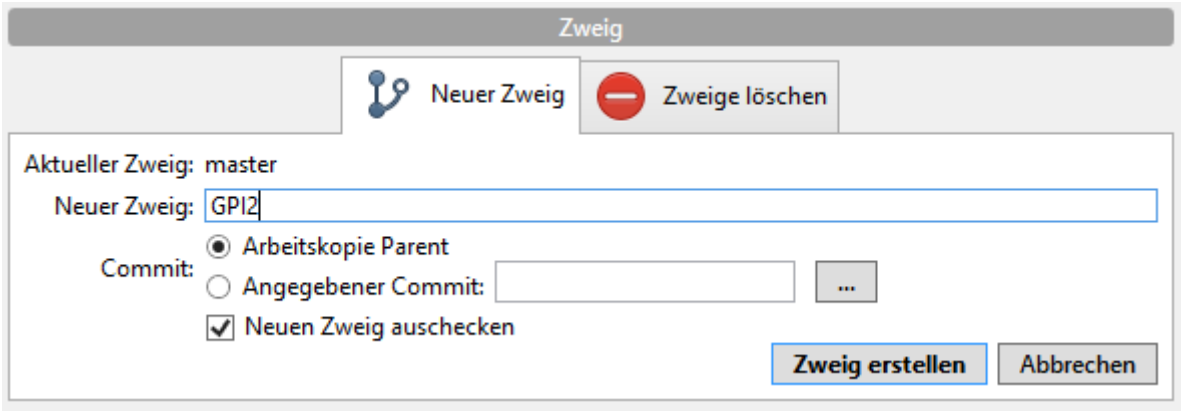
Nach einen doppelklick auf „origin/GPI“ wird mein Zweig von GitHub gedownloaded und als lokale Kopie eingefügt.



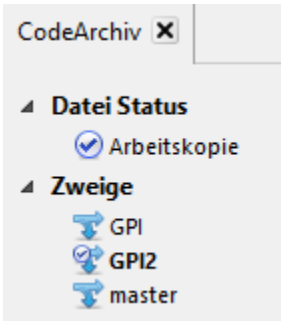
Links bei den Zweigen sind jetzt master und GPI – wobei GPI jetzt fett und damit aktiv sind. Die geänderten Dateien werden automatisch in den angegeben Verzeichnis abgebildet. Spring ruhig mal zwischen beiden Zweigen hin und her. Da „GPI“ schon gedownloaded wurde, nicht unter Remotes

sondern unter Zweige doppelklicken.

Aber kehren wir erstmal zur master zurück und erstellen einen neuen Zweig. Hierfür einfach in der Toolbar auf folgendes Icon klicken:

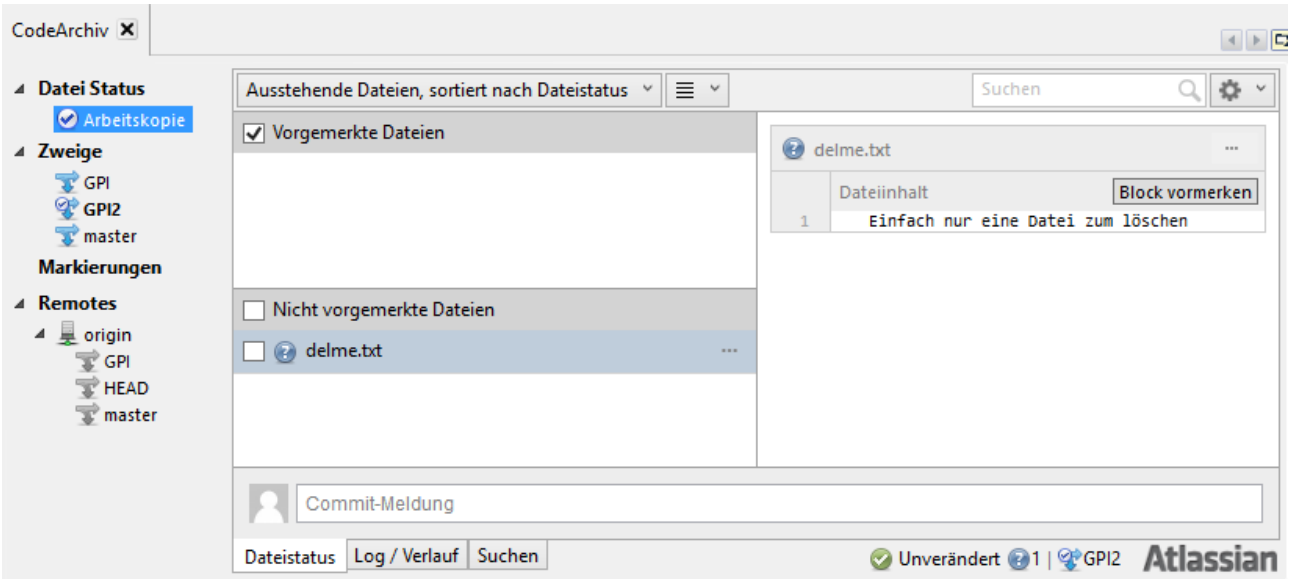


Als Zweigname bitte den Forumsnamen eingeben. Der neue Zweig sollte man gleich „auschecken“ d.h. aktivieren. Ungefähr so sollte nun die Seitenleiste aussehen.



Probieren wir doch mal aus, wenn wir eine neue Datei in GitHub-Verzeichnis erstellen. Ich nenn sie der einfachhaltshalber „delme.txt“ - dann weis ich, das es eine Nutzlose Testdatei ist.

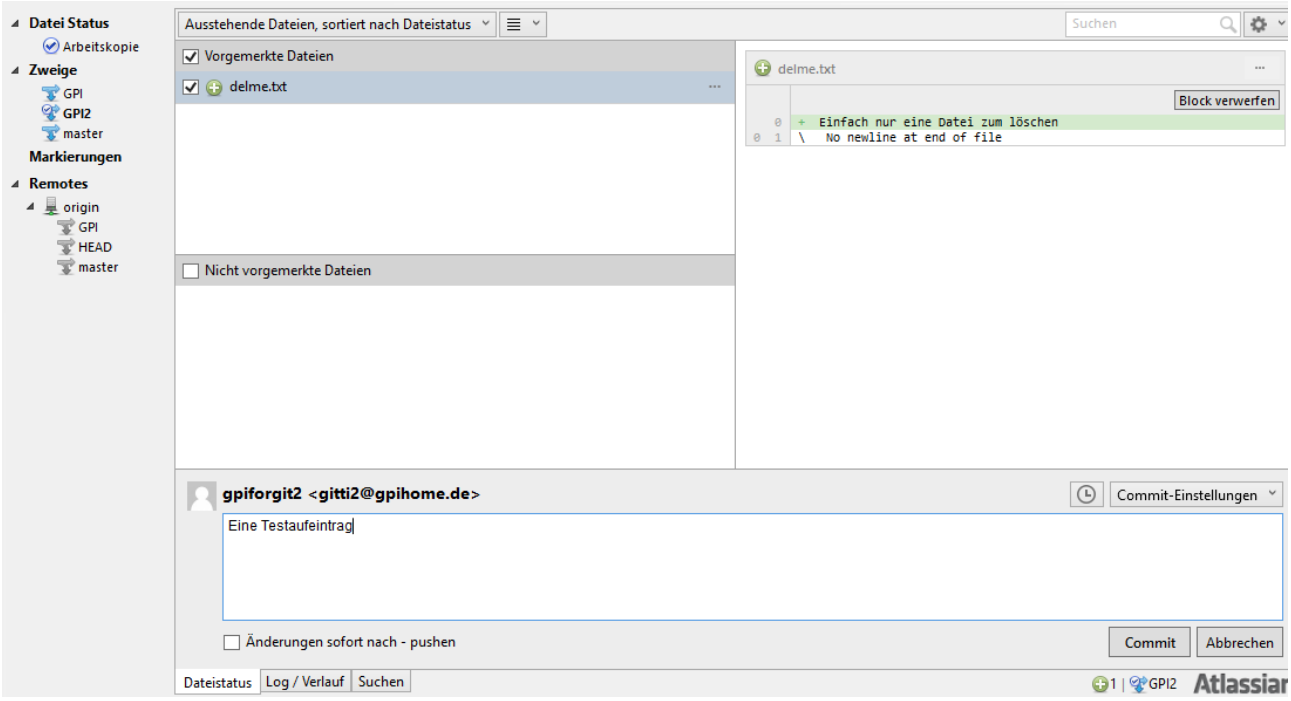
Bei SourceTree hat sich gleich was getan. Dazu gehen wir einfach bei den Zweigen auf den Eintrag aktuelle Arbeitskopie:



Hier sieht man was sich getan hat, welche Dateien hinzu gefügt wurden, welche gelöscht und welche verändert wurden. Außerdem kann man an der Statuszeile unten entnehmen das es eine neue

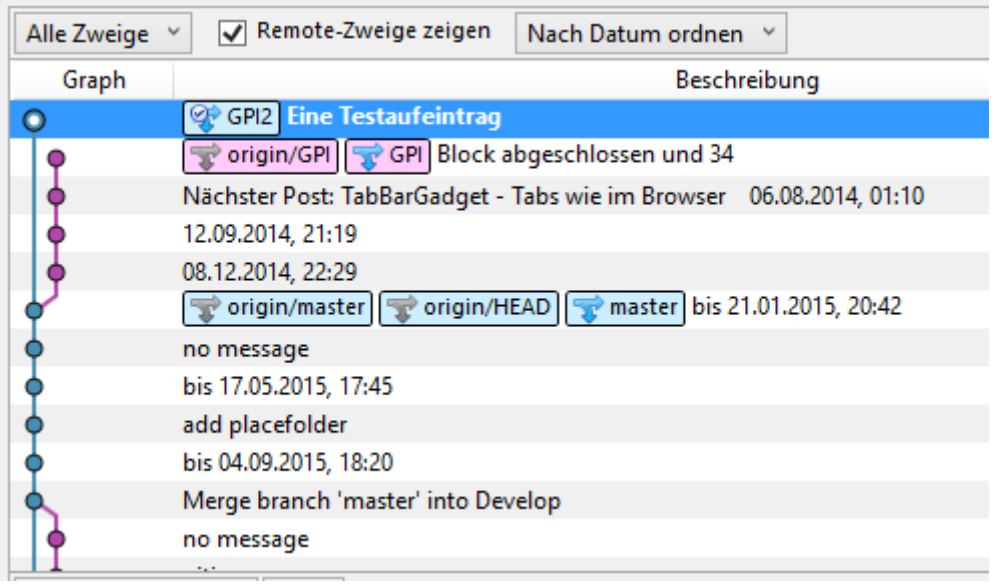
Datei gibt.

Es wird Zeit für den ersten Commit. Dazu einfach oben auf den Commit-Button klicken. Bei einen Commit wird der aktuelle Zustand in Git gesichert. Es ist jederzeit möglich zwischen diesen Zuständen hin und her zu springen und auch hier neue Zweige von einen früheren Punkt zu aktivieren. Also auf Commit oben gedürckt.



Die Dateien auswählen, so das sie oben stehen, unten in der Beschreibung reinschreiben, was man gerade gemacht hat (wie gesagt, ich vermerke da, wie weit ich in Forum gekommen bin) und dann auf Commit.

Und siehe da, der Eintrag ist jetzt im Baum:



Was man hier schön sieht:

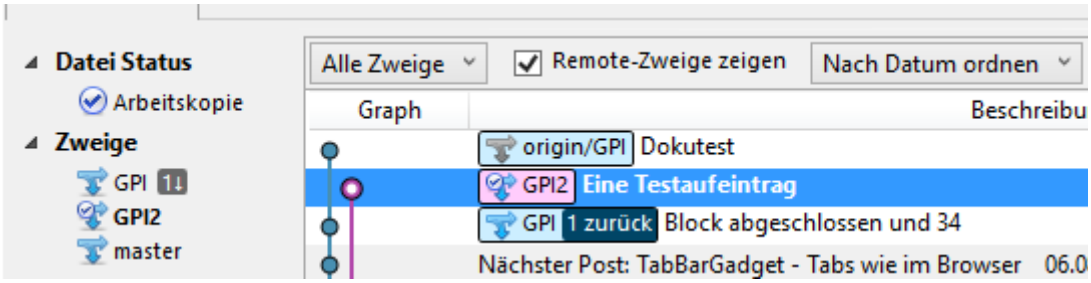
A) Ich sollte nicht schon zwei Sätze in voraus denken und Unsinn eintragen

B) Wo sich die einzelnen Zweige ansetzen. Per Doppelklick auf einen der Einträge kann man jederzeit zu einen bestimmten Zustand zurückkehren.

Bisher haben wir nur lokal gearbeitet, sprich auf Github ist nichts angekommen, das wollen wir jetzt ändern.

Erstmal sollten wir überprüfen ob wir noch aktuell sind. Das kann man mit „Anfordern“ und „Pull“ durchführen.

Anfordern heißt, das nur abgefragt wird, ob sich was geändert hat. Das sieht bspw. so aus:

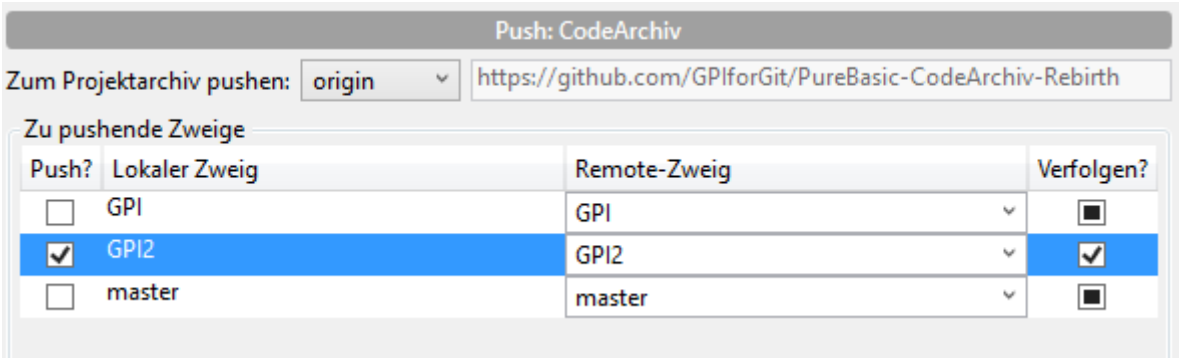


Hier sieht man, das man in Zweig GPI ein neuer Commit ist. Da wir ihn eh nicht nutzen, ignorieren wir ihn einfach.

Mit Pull aktualisieren wir aber die ganze Geschichte. Pull ist ein „Anforderung“ und anschließendes „Merge/Verbinden“ in einen Durchgang. Da wir eher selten die gleichen Dateien bearbeiten, sollte ein Pull ausreichend sein.

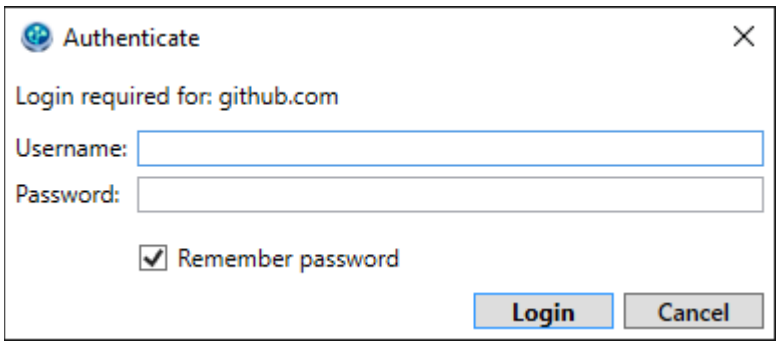
Um die eine Änderung zu holen, muss man den Zweig GPI aktivieren und auf Pull klicken. Den anschließenden Dialog einfach mit OK bestätigen.

Aber zurück zu unseren Zweig, den wollen wir ja hochladen. Hierfür einfach auf „Push“ klicken

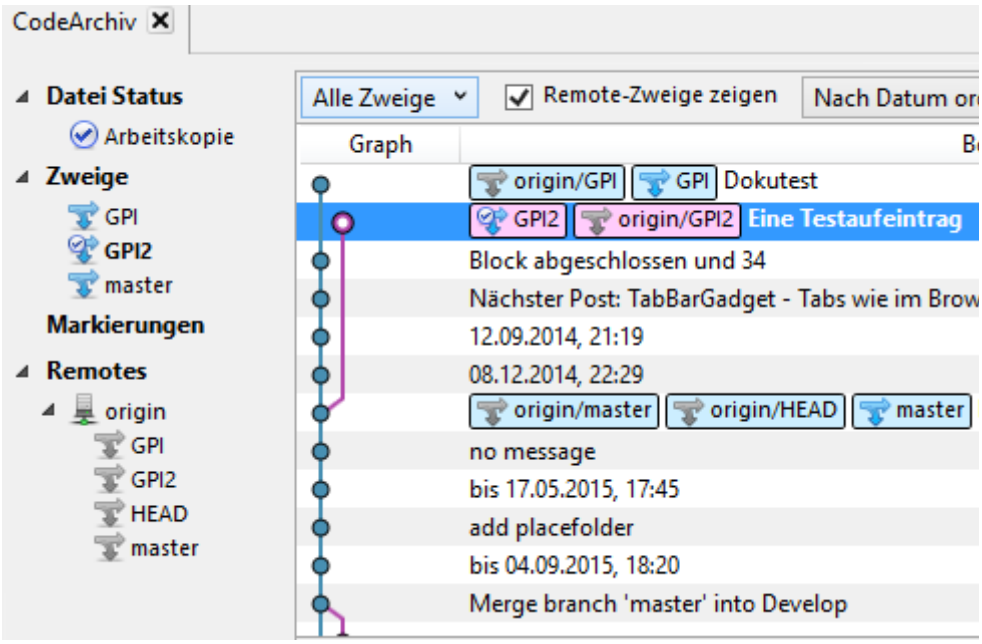


Einfach den Zweig, den wir hochladen wollen anklicken und bestätigen.

Beim ersten mal will er noch den Namen und das Passwort:

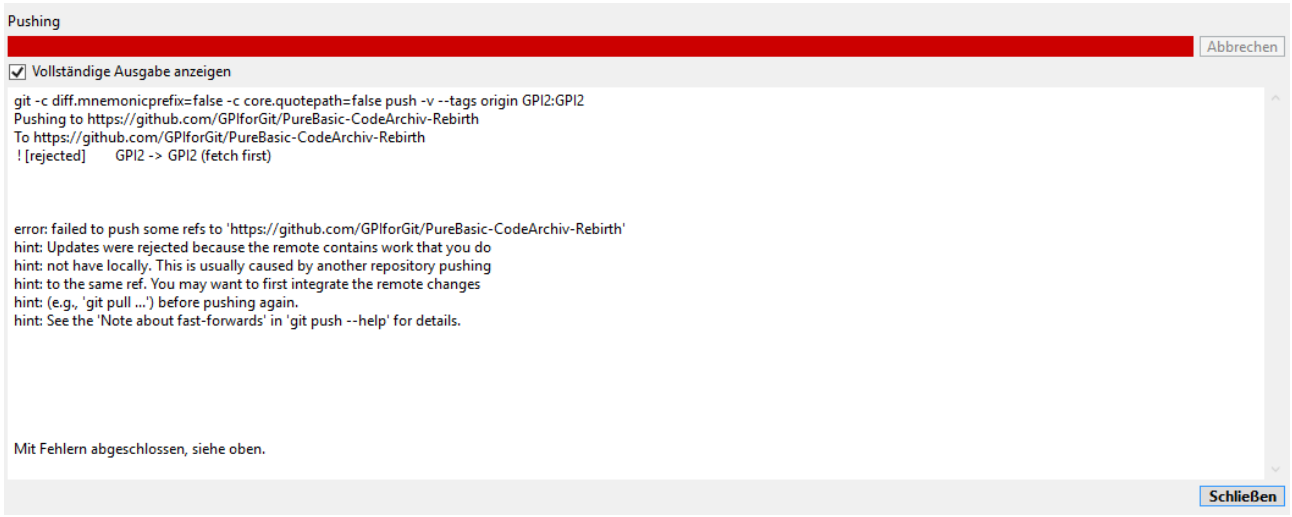


Und schon ist der neue Zweig auf den Server:



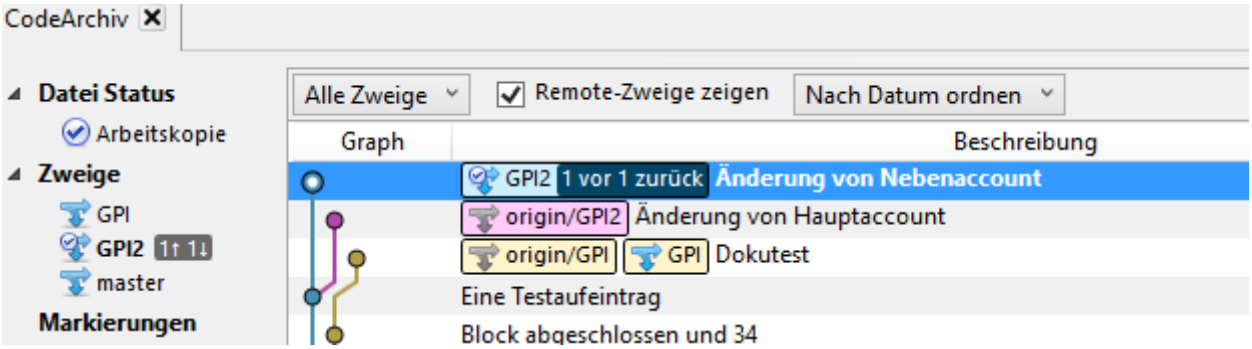
Jetzt mach ich mal was gemeines. Von meinen Haupt-Account aus wechsle ich in den Zweig GPI2, ändere die Delme.txt, commit und pushe sie auf den Server. Davon bekommt der Nebenaccount nichts mit.

Der ändert ebenfalls die gleiche Datei, commit ebenfalls und will gleich die Datei hochpushen. Jetzt gibt es einen Konflikt!



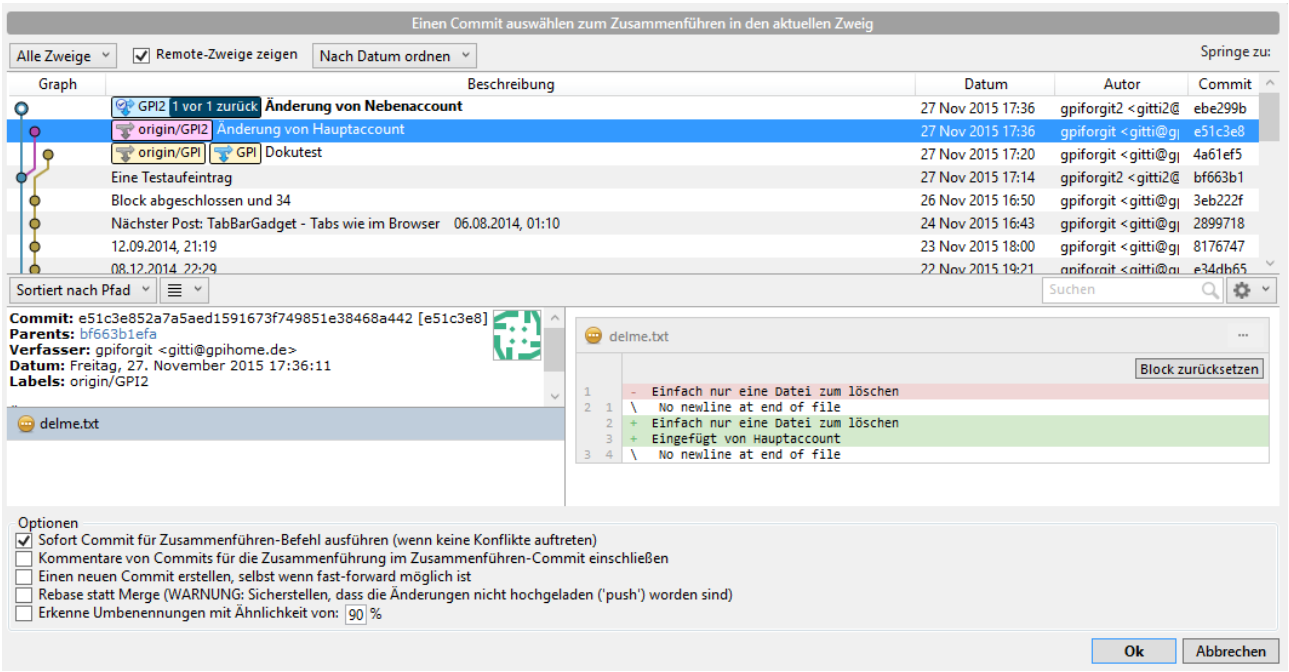
Es konnte ein Commit erzeugt werden, der Push schlug aber fehl, weil zwischenzeitlich ja der Hauptaccount ebenfalls diesen Zweig editiert hat.

Klicken wir also auf Anfordern (fetch in Englischen) und schauen mal, was passiert:

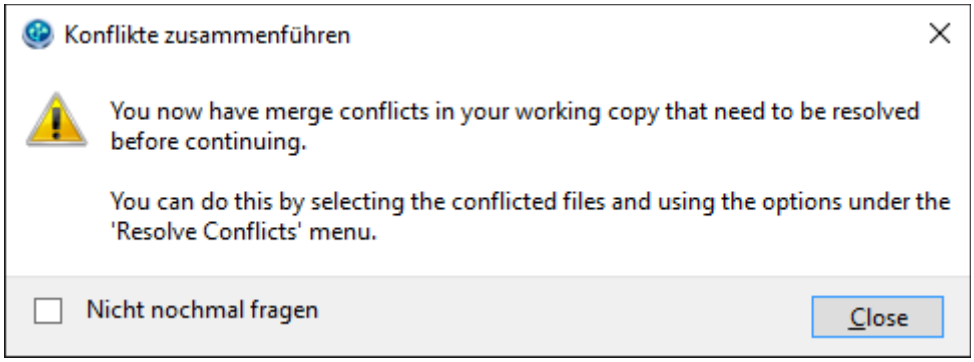


Man sieht schön, das der Zweig GPI2 seit „Eine Testaufeintrag“ sich spaltet. Einmal unsere lokale Kopie und einmal die entfernte origin/GPI2 Zweig (der liegt ja auf Github).

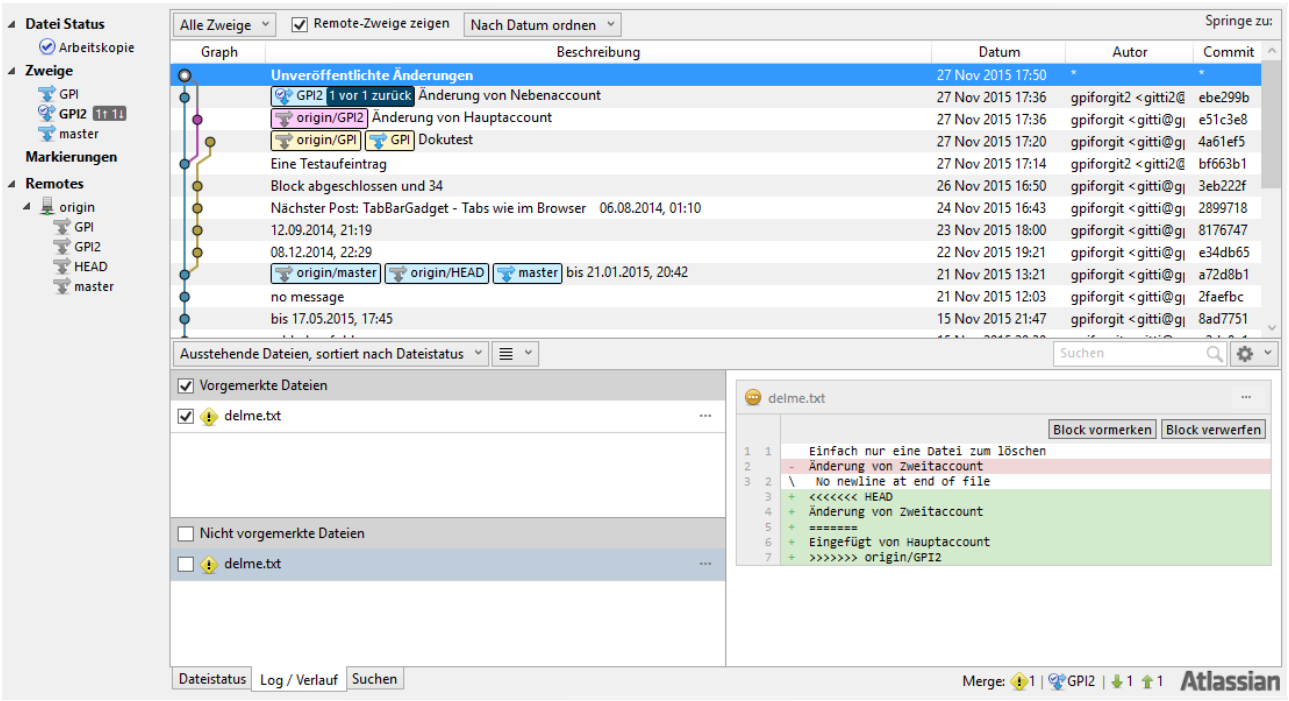
Wir wollen mal beide Subzweige zu einen zusammenführen. Dazu klicken wir auf Zusammenführen.



Das sieht jetzt komplizierter aus als es ist. Wir wählen oben den Zweig, den wir mit den aktuellen Verbinden wollen und klicken auf OK. Wenn es keine Konflikte gibt, wird automatisch ein neuer Commit erzeugt und alles ist gut. Dumm nur, das wir hier einen Konflikt haben – in beiden Zweigen wurde die gleiche Datei geändert und Git weis nicht, was zu tun ist. Aber klicken wir erstmal auf OK.

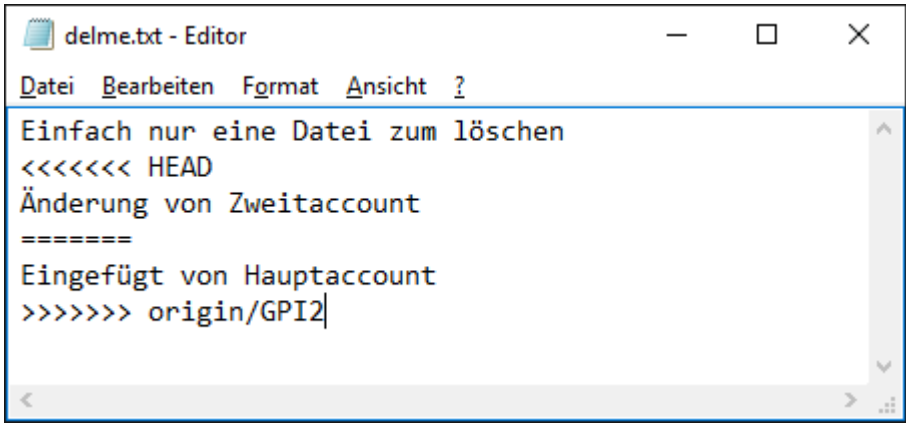


Wie erwartet gibt’s Probleme.



Wenn man unveröffentliche Änderungen auswählt, sieht man gleich, die „delme.txt“ ist doppelt vorhanden.

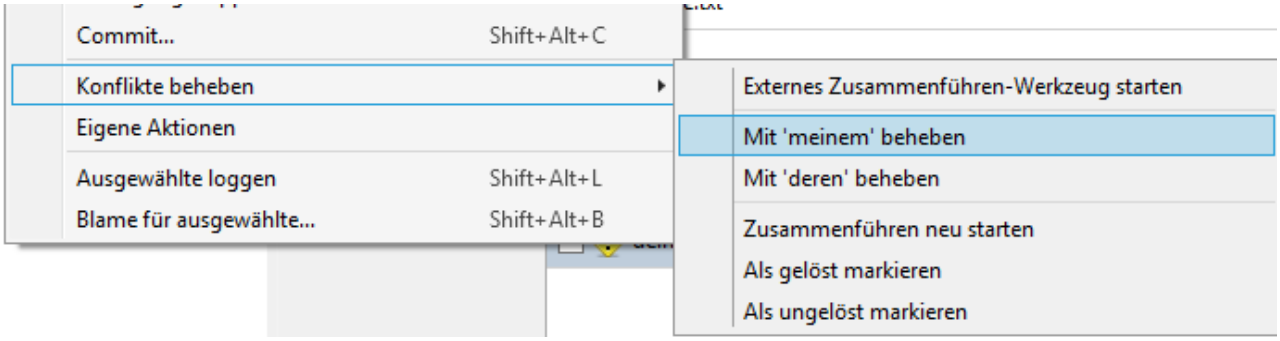
Wenn wir diese mal in Editor anschauen:



Hier sieht man schön was los ist. Der HEAD zeigt an, was bei uns lokal steht (bis zu den =======) und danach was auf origin/GPI2 (also auf GITHUB) wäre.

Wir haben drei Möglichkeiten: Wir ändern die delme.txt mit einen Editor, so das es passt, wir übernehmen die lokale Datei, wir übernehmen die origin-Datei.

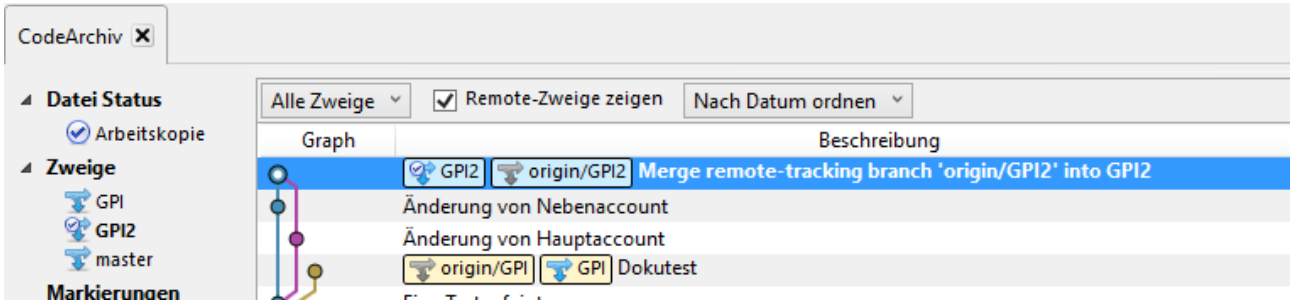
Wir rechtsklicken auf die untere Delme.txt



Wenn man die Datei manuell verändert, hier kann man sagen mit „meinem“ beheben, das man die lokale haben will, mit deren beheben, das man die remote will. Oder man editiert die Datei und wählt „als gelöst markieren“ aus.

Welche Lösung man auch immer nimmt, anschließend ist das Problem weg.

Wir können jetzt einen neuen Commit erstellen und das Ergebnis hochladen:



Pull macht übrigens das ganze etwas automatischer. Es ist eigentlich nichts weiter wie ein „Anfordern“ mit einen „Zusammenführen“ auf den Remotezweig. Man hat nur weniger Kontrolle, das sollte aber für uns eher egal sein.

Eine Bitte hätte ich dann doch, bitte rührt den Masterzweig nicht an, den möchte ich gerne erstmal selbst verwalten.

Bei Fragen etc. bitte mich in deutschen Forum anschreiben.

Danke für die Hilfe.