

Vorbereitungen

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

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

SourceTree zurücksetzen (falls nötig)

Windows:

Hierfür einfach folgende beiden Ordner löschen (falls vorhanden):

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

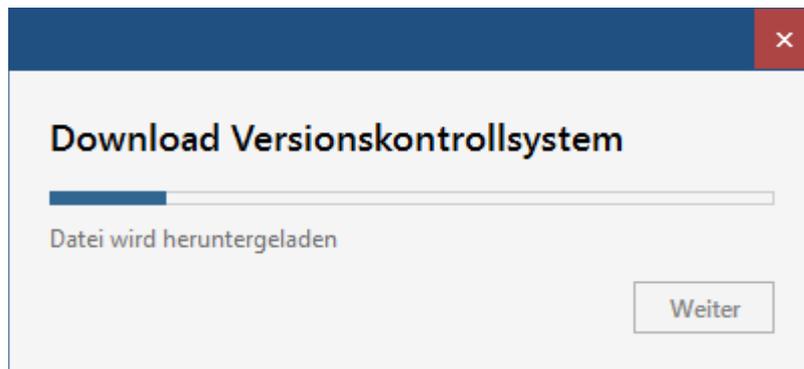
C:\Users\<<windows-account-name>\Roaming\Local\Atlassian

SourceTree vorbereiten

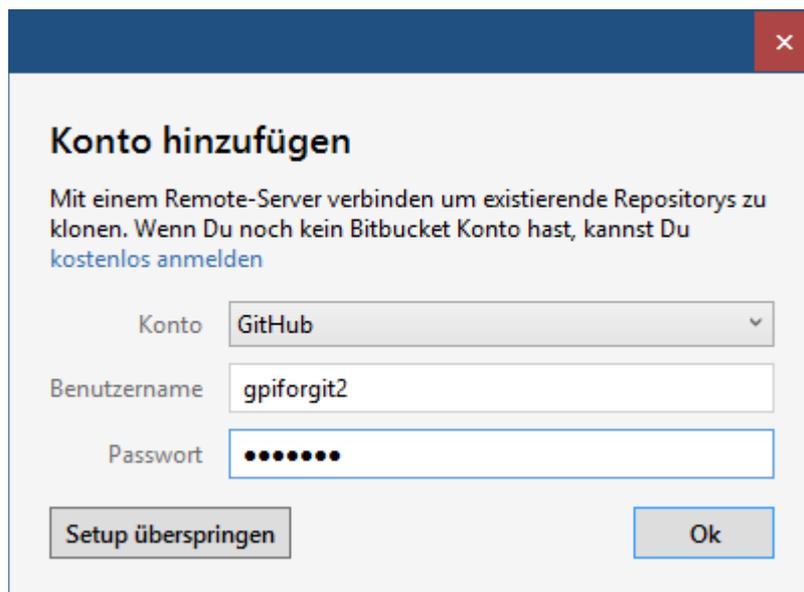
Die Installation sollte nicht weiter kompliziert sein, es ist eine typische WWF (weiter, weiter, fertig) Installation. In Anschluss CodeTree starten.



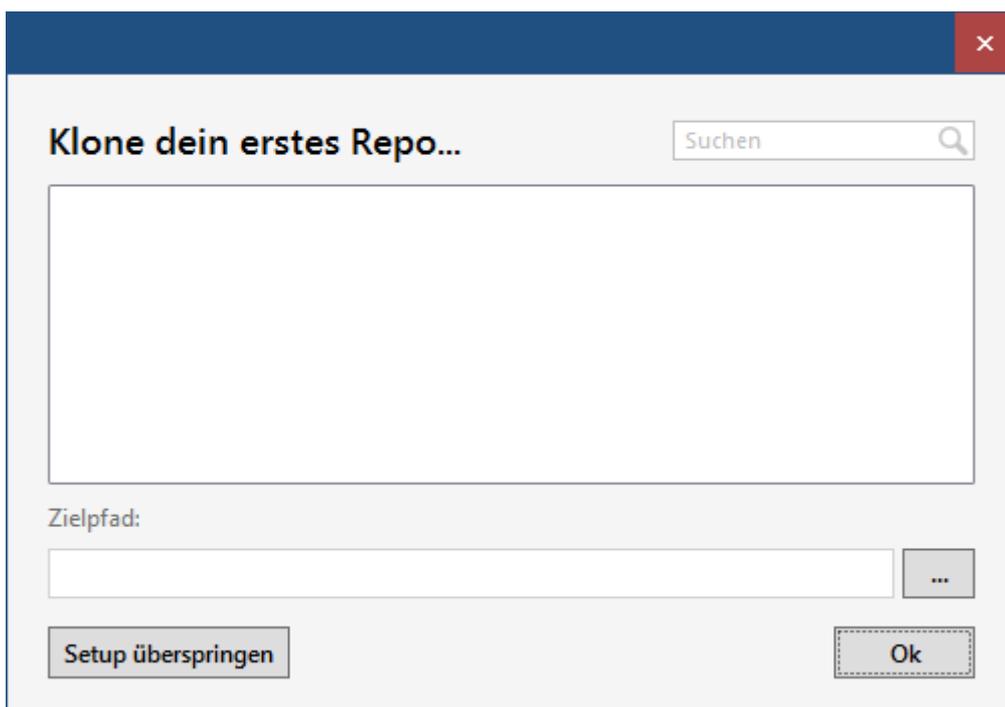
Wir werden wohl die Vereinbarung akzeptieren müssen. Anschließend auf Fortfahren.



Die GIT-Engine wird automatisch gedownloadet

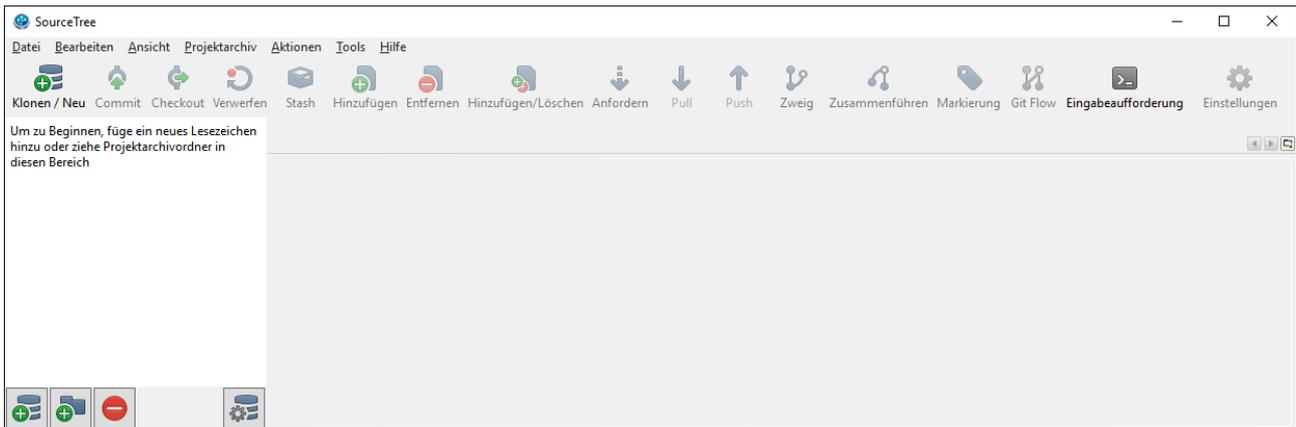


Konto GitHub auswählen, den Benutzernamen (nicht Email) und Passwort eingeben. Anschließend mit OK bestätigen.

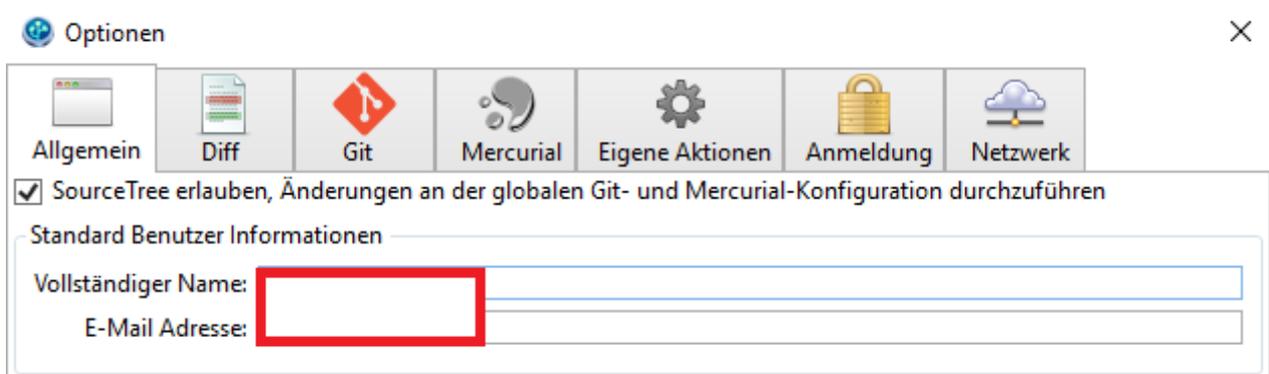


Da wir ja die CodeArchiv-Repo benutzen wollen, überspringen wir hier das Setup,

SourceTree sollten uns jetzt mit den Default-Bildschirm begrüßen:



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

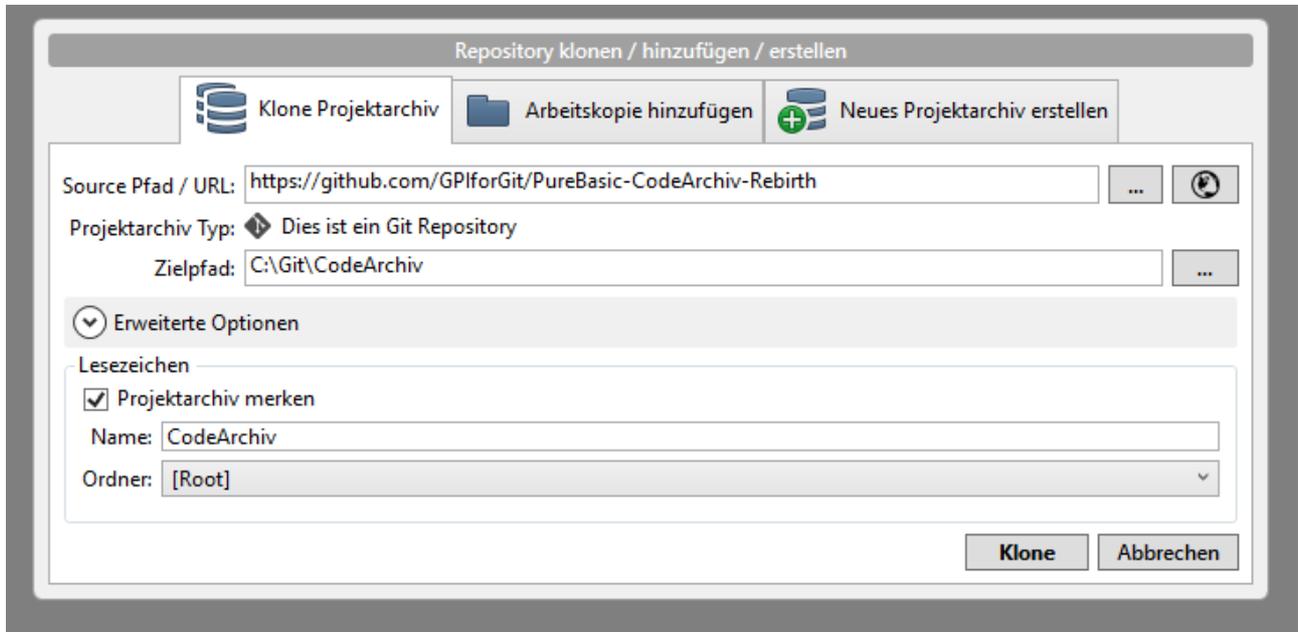


Hier müssen wir unbedingt unter Standard-Benutzer den GitHub-Benutzernamen und die E-Mail Adresse ergänzen.

Die CodeArchiv Repo holen



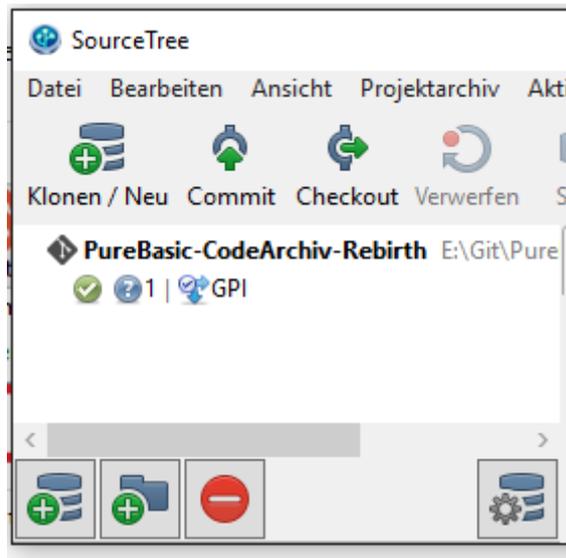
Dazu klicken wir auf das „Klonen / Neu“-Icon aus der Toolbar des Hauptfensters und füllen folgendes Fenster entsprechend aus:



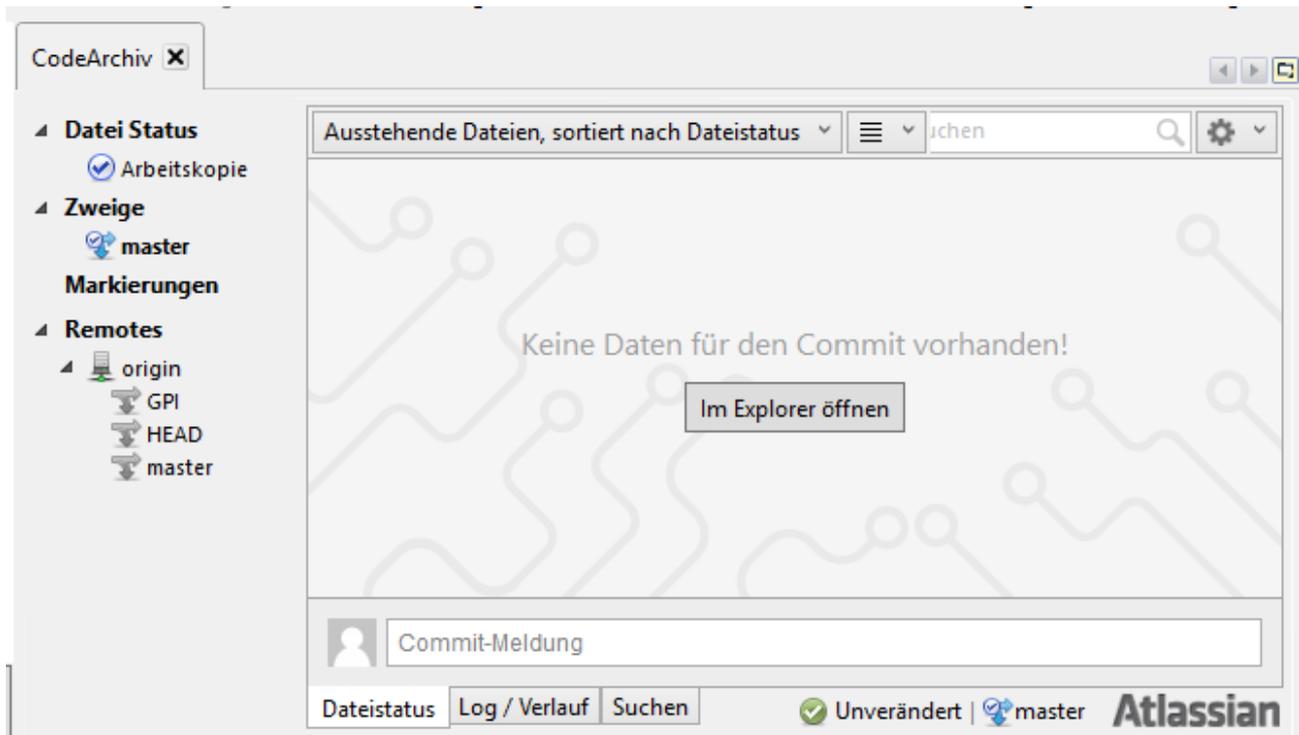
Als URL einfach die GitHub-Seite: <https://github.com/GPIforGit/PureBasic-CodeArchiv-Rebirth>

Ein Klick auf Klone und schon wird die Repo kopiert.

Auf der linken Seite des Fensters werden alle kopierten Repos aufgelistet:



Checkout und Zweige



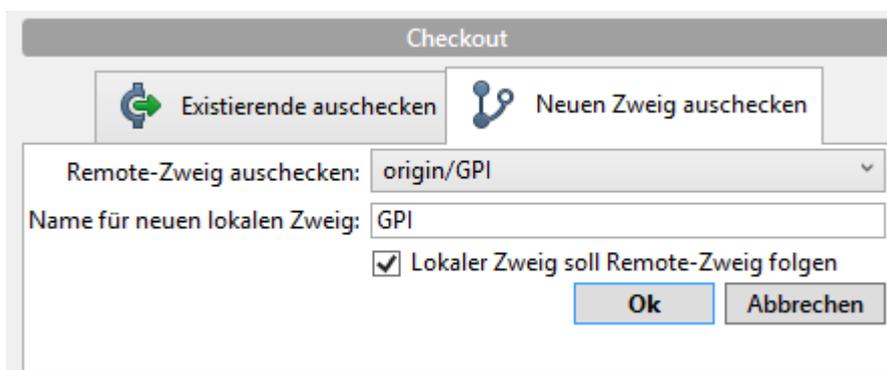
Links sind die verschiedenen Zweige. Aktuell ist der „master“-Zweig aktiv, erkennbar an den fetten Schriftzug. In Remotes kann man die auf GitHub vorhandenen Zweige sehen. Interessant sind hier eigentlich nur GPI – das ist mein Zweig und master – das ist quasi der veröffentlichte Zweig.

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 dem 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 wurden. Solange hier nicht Unverändert steht, sollte man nicht auf einen anderen Zweig klicken. In Zweifelsfall warnt aber einen das Programm.

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

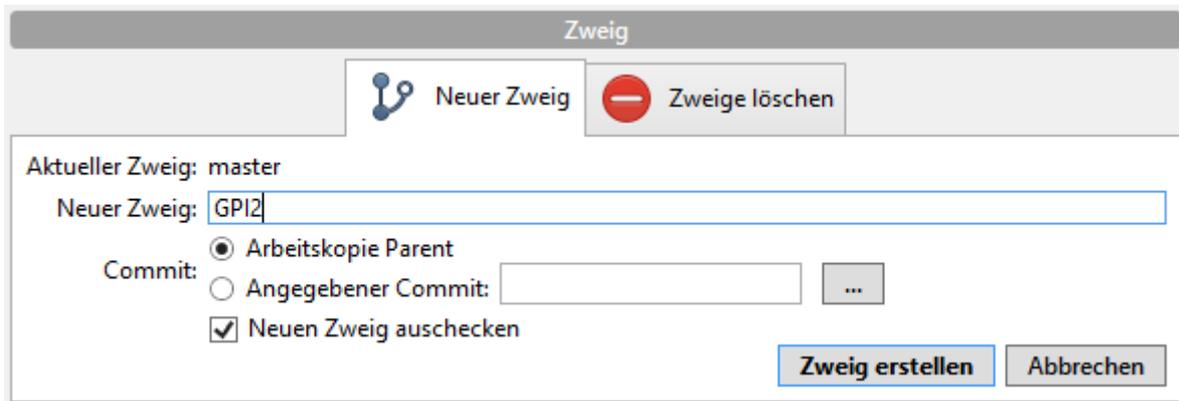
Nach einen doppelklick auf „origin/GPI“ wird mein Zweig von GitHub gedownload und als lokale Kopie eingefügt.



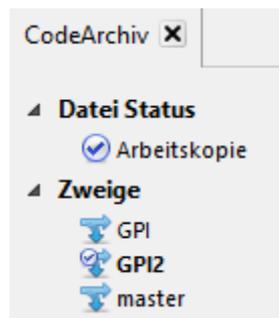
Links bei den Zweigen sind jetzt master und GPI – wobei GPI fett und damit aktiv ist. Die

geänderten Dateien werden automatisch in den angegebenen Verzeichnis abgebildet. Spring ruhig mal zwischen beiden Zweigen hin und her. Da „GPI“ schon gedownloadet wurde, nicht unter Remotes sondern unter Zweige doppelklicken.

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

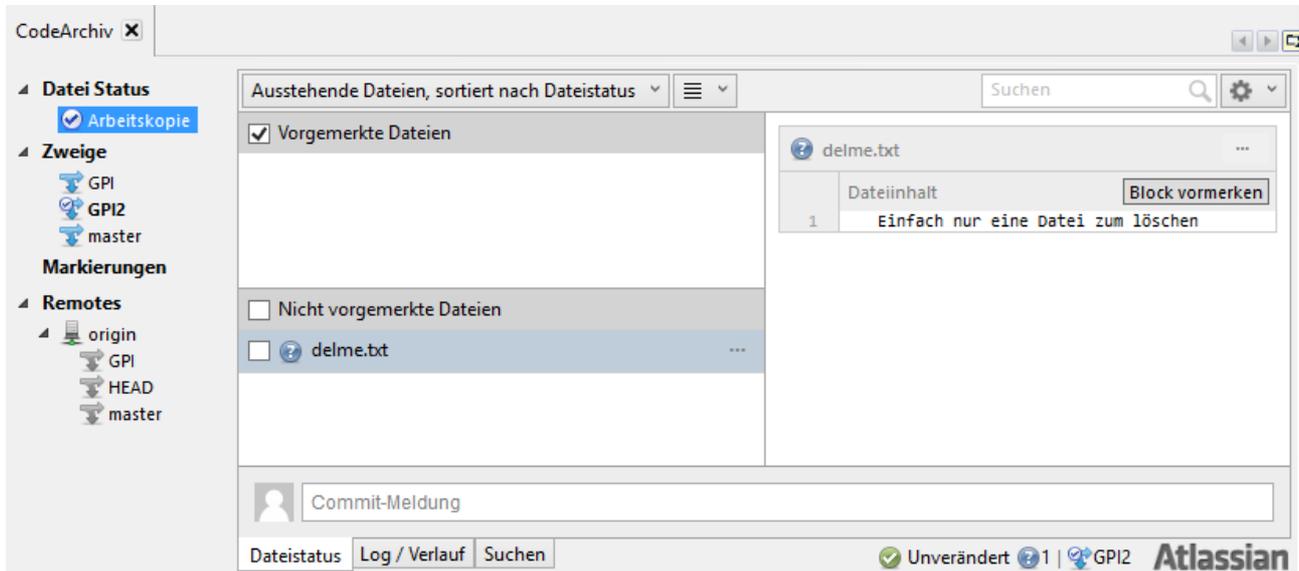


Als Zweigname bitte den Forumsnamen eingeben. Der neue Zweig sollte man gleich „auschecken“ d.h. aktivieren. Ungefähr so sollte nun die Seitenleiste aussehen.



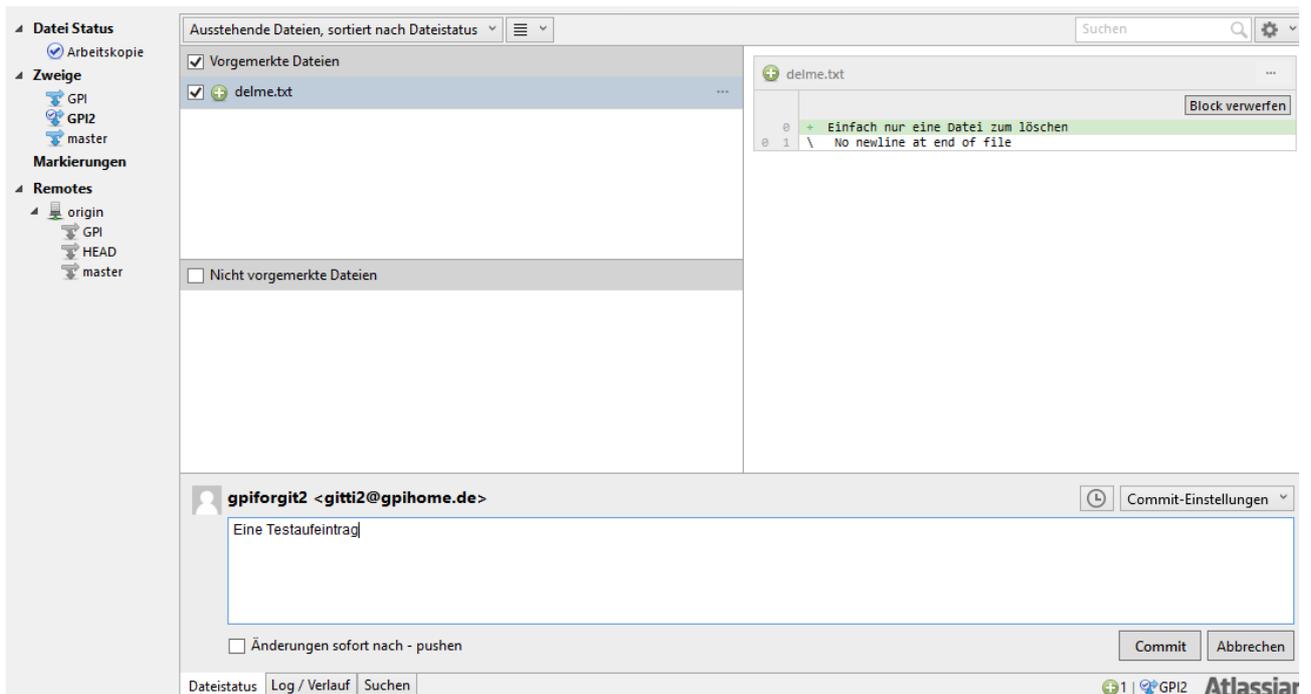
Unser erster Commit

Probieren wir doch mal aus, was passiert, wenn wir eine neue Datei in SourceArchiv-GitHub-Verzeichnis erstellen. Ich nenne sie der Einfachheit halber „delme.txt“ - dann weiß ich, dass 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.

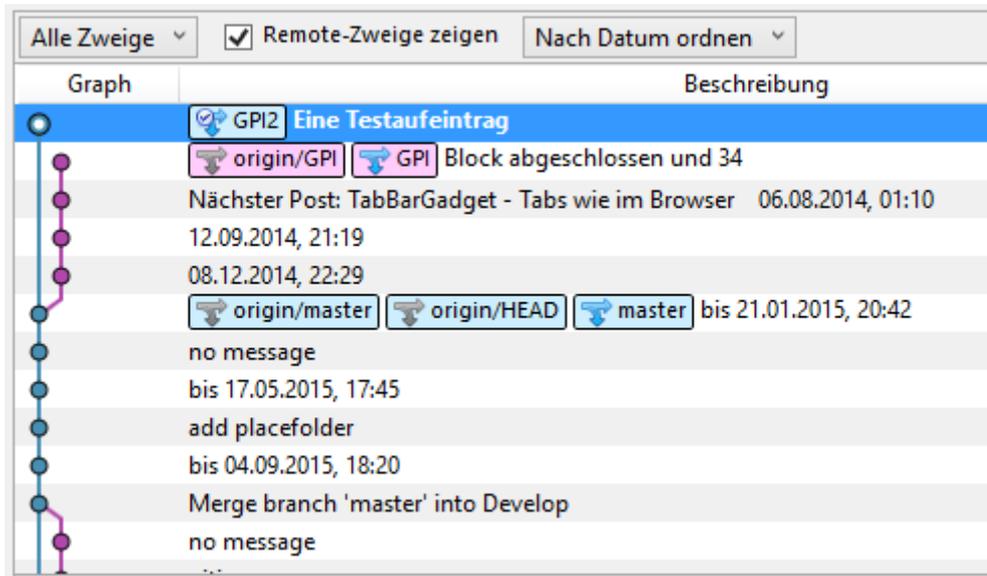
Es wird Zeit für den ersten Commit. Dazu einfach oben auf den Commit-Button klicken. Bei einem Commit wird der aktuelle Zustand in Git gesichert.



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 aufspalten. Per Doppelklick auf einem der Einträge kann man jederzeit zu einen bestimmten Zustand zurückkehren.

Es ist jederzeit möglich zwischen diesen Zuständen hin und her zu springen und auch hier neue Zweige von einen früheren Punkt zu aktivieren.

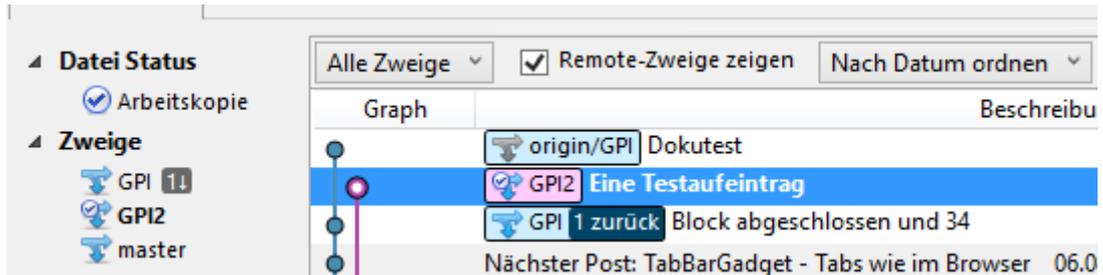
Auf Github veröffentlichen

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:

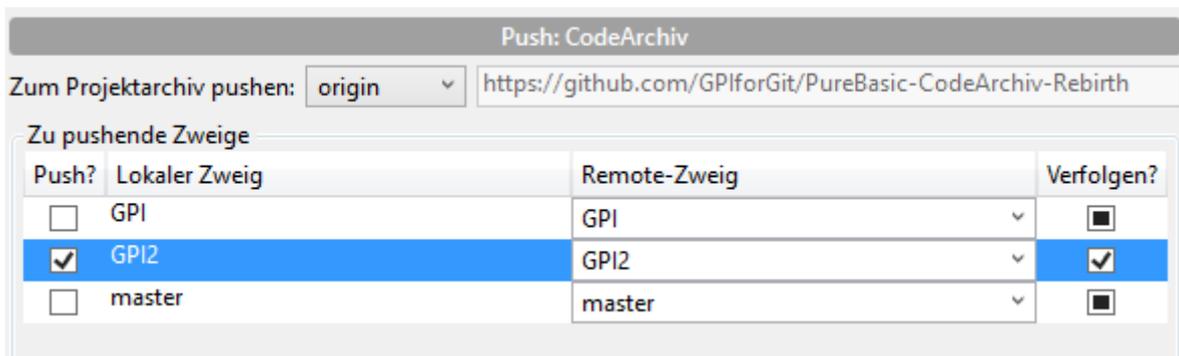


Hier sieht man, das man in Zweig GPI ein neuer Commit ist. Da wir ihn eh nicht nutzen, ignorieren wir ihn einfach.

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.

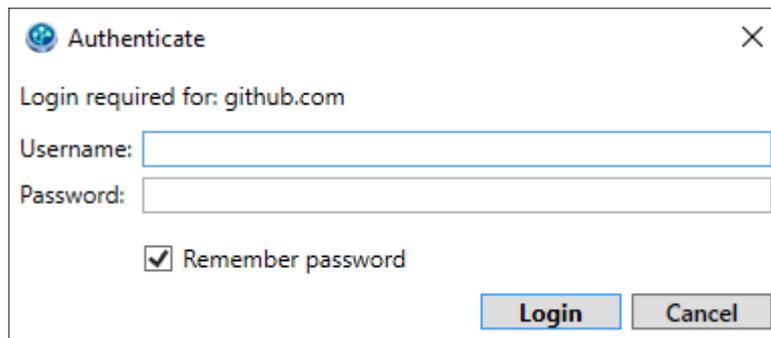
Um die eine Änderung zu holen, muss man den Zweig GPI aktivieren und auf Pull klicken. Den anschließenden Dialog einfach mit OK bestätigen.

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

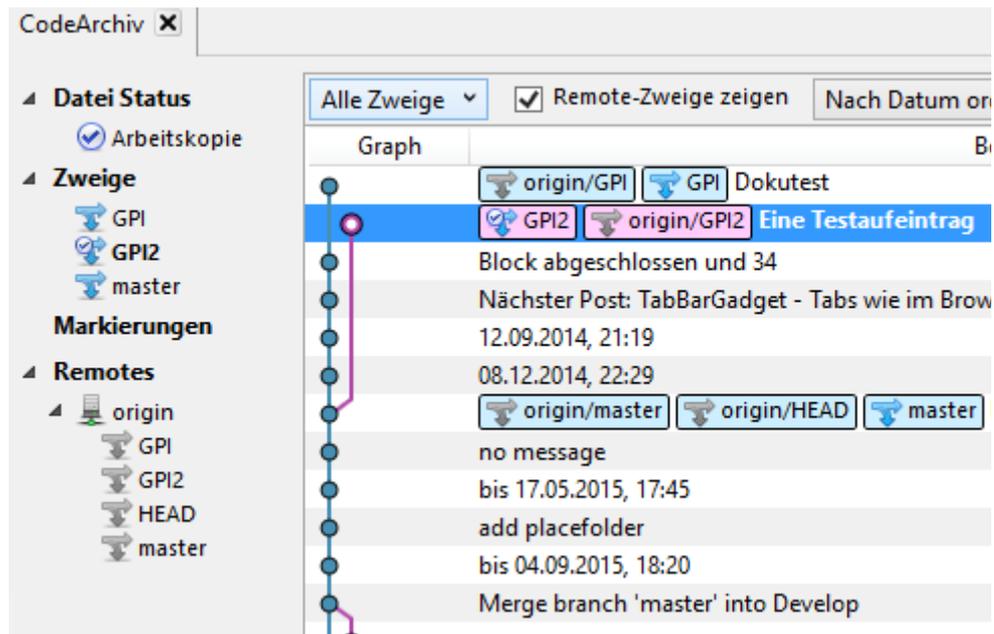


Einfach den Zweig, den wir hochladen wollen anklicken und bestätigen.

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



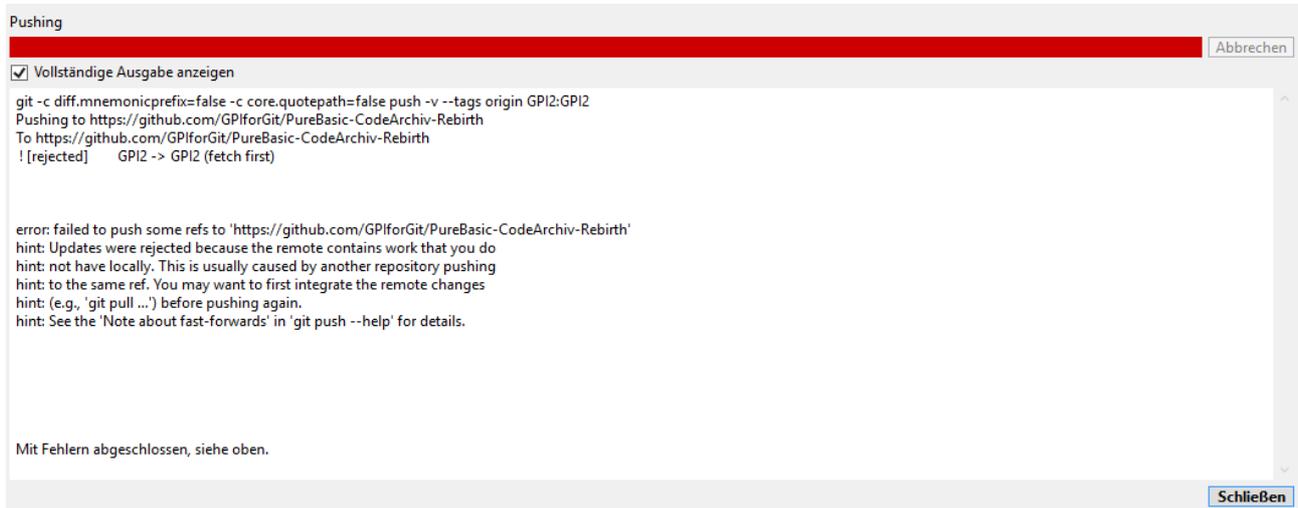
Und schon ist der neue Zweig auf dem Server:



Konflikte lösen

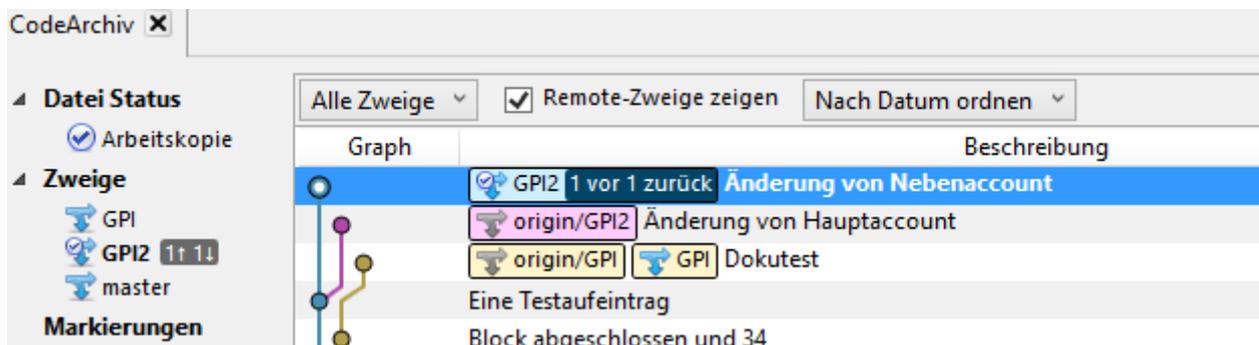
Jetzt mach ich mal was gemeines. Von meinen Haupt-Account aus wechsele ich in den Zweig GPI2, ändere die delme.txt, commit und pushe sie auf den Server. Davon bekommt der Nebenaccount nichts mit.

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, aber der Push schlug fehl, weil zwischenzeitlich ja der Hauptaccount ebenfalls diesen Zweig editiert und minimum eine gleiche Datei geändert 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.



Ein Commit auswählen zum Zusammenführen in den aktuellen Zweig

Alle Zweige Remote-Zweige zeigen Nach Datum ordnen Springe zu:

Graph	Beschreibung	Datum	Autor	Commit
	GPI2 1 vor 1 zurück Änderung von Nebenaccount	27 Nov 2015 17:36	gpiforgit2 <gitti2@	ebe299b
	origin/GPI2 Änderung von Hauptaccount	27 Nov 2015 17:36	gpiforgit <gitti@gj	e51c3e8
	origin/GPI Dokutest	27 Nov 2015 17:20	gpiforgit <gitti@gj	4a61ef5
	Eine Testaufeintrag	27 Nov 2015 17:14	gpiforgit2 <gitti2@	bf663b1
	Block abgeschlossen und 34	26 Nov 2015 16:50	gpiforgit <gitti@gj	3eb222f
	Nächster Post: TabBarGadget - Tabs wie im Browser 06.08.2014, 01:10	24 Nov 2015 16:43	gpiforgit <gitti@gj	2899718
	12.09.2014, 21:19	23 Nov 2015 18:00	gpiforgit <gitti@gj	8176747
	08.12.2014 22:29	22 Nov 2015 19:21	aniforait <nitti@n	e34db65

Sortiert nach Pfad Suchen

Commit: e51c3e852a7a5aed1591673f749851e38468a442 [e51c3e8]
Parents: bf663b1efa
Verfasser: gpiforgit <gitti@gpishome.de>
Datum: Freitag, 27. November 2015 17:36:11
Labels: origin/GPI2

delme.txt

```

1 - Einfach nur eine Datei zum löschen
2 \ No newline at end of file
3 + Einfach nur eine Datei zum löschen
4 + Eingefügt von Hauptaccount
5 \ No newline at end of file

```

Optionen
 Sofort Commit für Zusammenführen-Befehl ausführen (wenn keine Konflikte auftreten)
 Kommentare von Commits für die Zusammenführung im Zusammenführen-Commit einschließen
 Einen neuen Commit erstellen, selbst wenn fast-forward möglich ist
 Rebase statt Merge (WARNUNG: Sicherstellen, dass die Änderungen nicht hochgeladen ('push') worden sind)
 Erkenne Umbenennungen mit Ähnlichkeit von: %

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. Bei unseren Beispiel ja leider nicht, aber klicken wir erstmal auf ok

Konflikte zusammenführen Nicht nochmal fragen

You now have merge conflicts in your working copy that need to be resolved before continuing.

You can do this by selecting the conflicted files and using the options under the 'Resolve Conflicts' menu.

Die erwartete Fehlermeldung. Bei den Zweigen wählen wir „unveränderte Änderungen“:

Arbeitskopie

Zweige: GPI, GPI2 (11:11), master

Markierungen

Remotes: origin, GPI, GPI2, HEAD, master

Alle Zweige Remote-Zweige zeigen Nach Datum ordnen Springe zu:

Graph	Beschreibung	Datum	Autor	Commit
	Unveränderte Änderungen	27 Nov 2015 17:50	*	*
	GPI2 1 vor 1 zurück Änderung von Nebenaccount	27 Nov 2015 17:36	gpiforgit2 <gitti2@	ebe299b
	origin/GPI2 Änderung von Hauptaccount	27 Nov 2015 17:36	gpiforgit <gitti@gj	e51c3e8
	origin/GPI Dokutest	27 Nov 2015 17:20	gpiforgit <gitti@gj	4a61ef5
	Eine Testaufeintrag	27 Nov 2015 17:14	gpiforgit2 <gitti2@	bf663b1
	Block abgeschlossen und 34	26 Nov 2015 16:50	gpiforgit <gitti@gj	3eb222f
	Nächster Post: TabBarGadget - Tabs wie im Browser 06.08.2014, 01:10	24 Nov 2015 16:43	gpiforgit <gitti@gj	2899718
	12.09.2014, 21:19	23 Nov 2015 18:00	gpiforgit <gitti@gj	8176747
	08.12.2014, 22:29	22 Nov 2015 19:21	gpiforgit <gitti@gj	e34db65
	origin/master origin/HEAD master bis 21.01.2015, 20:42	21 Nov 2015 13:21	gpiforgit <gitti@gj	a72d8b1
	no message	21 Nov 2015 12:03	gpiforgit <gitti@gj	2faefbc
	bis 17.05.2015, 17:45	15 Nov 2015 21:47	gpiforgit <gitti@gj	8ad7751

Ausstehende Dateien, sortiert nach Dateistatus

Vorgemerkte Dateien

- delme.txt

Nicht vorgemerkte Dateien

- delme.txt

delme.txt

```

1 Einfach nur eine Datei zum löschen
2 - Änderung von Zweitaccount
3 \ No newline at end of file
4 <<<<<< HEAD
5 + Änderung von Zweitaccount
6 =====
7 + Eingefügt von Hauptaccount
8 >>>>>> origin/GPI2

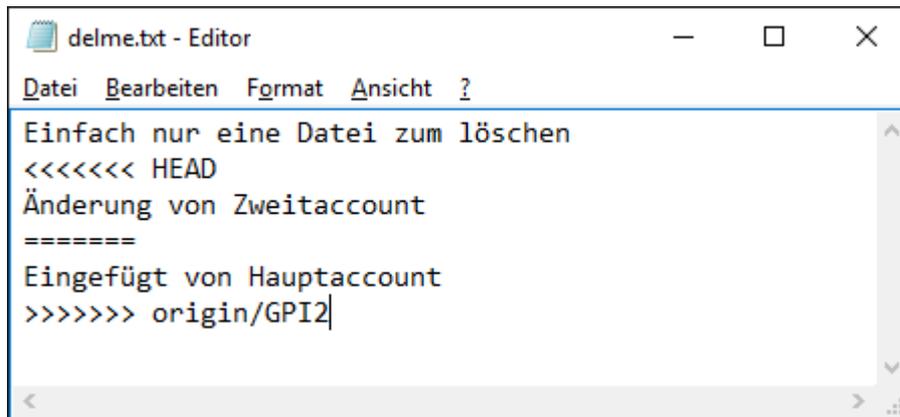
```

Dateistatus Log / Verlauf Suchen

Merge: 1 | | 1 1 **Atlassian**

Man sieht gleich die doppelte „delme.txt“ (einmal unter „Vorgemerkte Dateien“ und einmal „Nicht vorgemerkte Dateien“).

Wenn wir diese mal in Editor anschauen:

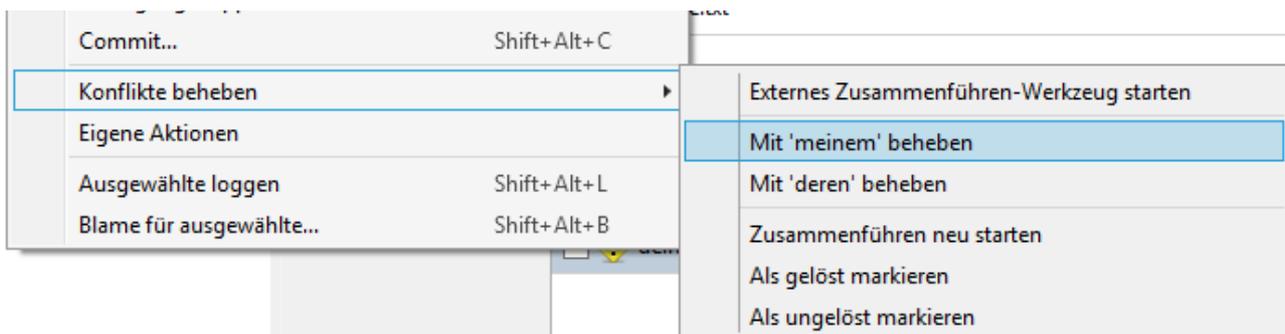


```
delme.txt - Editor
Datei Bearbeiten Format Ansicht ?
Einfach nur eine Datei zum löschen
<<<<<< HEAD
Änderung von Zweitaccount
=====  
Eingefügt von Hauptaccount
>>>>>> origin/GPI2
```

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 einem Editor, so das es passt, wir übernehmen die lokale Datei, wir übernehmen die origin-Datei.

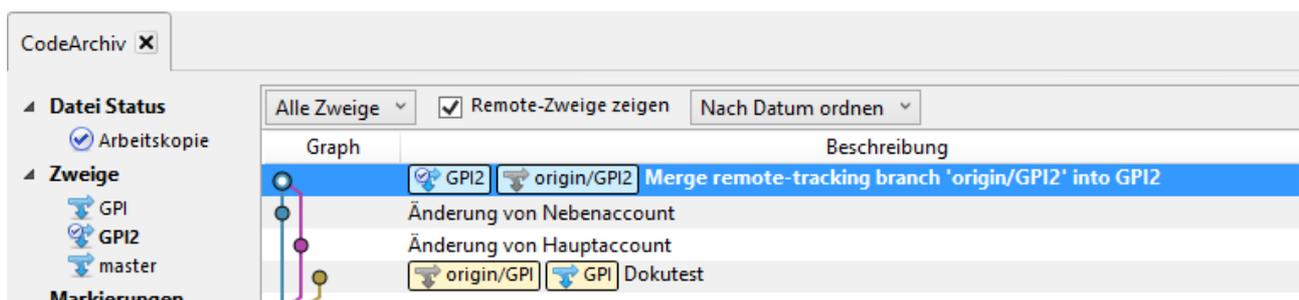
Wir rechtsklicken auf die untere delme.txt.



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 einem „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.

SourceTree lizenzieren

SourceTree ist zwar kostenlos, leider muss man sich trotzdem registrieren. Irgendwie ist die Programmfunktion defekt, aber man kann sich auf der Seite auch manuell anmelden.

<https://id.atlassian.com/signup>



Welcome to Atlassian!

Email address

Full name

Choose a password Show password

Ich bin kein Roboter. 
Datenschutzerklärung - Nutzungsbedingungen

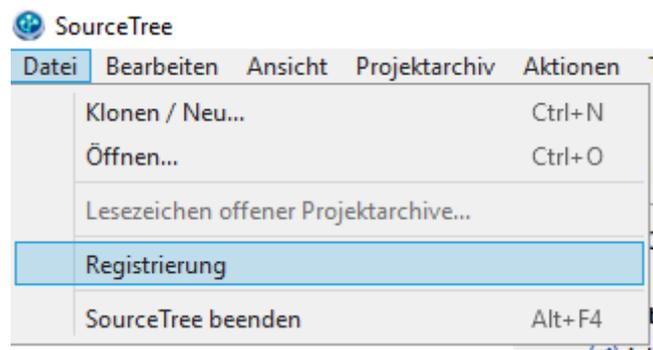
Sign up

[Log in using your existing Atlassian account](#)

Im Anschluss sollte man eine E-Mail bekommen. Den Link in der E-Mail folgen und man kommt auf diese Seite:

SEN	Product	Name	Support Expires	Support
▼ SEN-6227529	SourceTree: Free License	<input type="text"/>	28 Nov 2015	
Name <input type="text"/>		Licensed To <input type="text"/>	Download your SourceTree license.  Within SourceTree select from the menu bar "SourceTree > Registration", click "Import License" to import the downloaded the license file.	
SEN ? <input type="text"/>	License Key			
Technical Contact ? ▼ <input type="text"/> <input type="text" value="contact@example.com"/> <input type="button" value="Add"/>	Actions Archive			
Billing Contact ? ▼ <input type="text"/> <input type="text" value="contact@example.com"/> <input type="button" value="Add"/>				

Der große rote Pfeil zeigt auf den Downloadlink. Sobald diese Datei gedownloaded ist, kann man sie in SourceTree importieren.



Den entsprechenden Menüpunkt findet man unter Datei.



Hier auf „Lizenz importieren“ klicken und die Lizenz-Datei auswählen. Fertig.

Wie sollte man in CodeArchiv arbeiten

In Hauptverzeichnis gibt es eine Template.pb, da ist dann der nötige Header gleich mit drin. Nach Möglichkeit den Code mit einer 64Bit-Version und Unicode testen. Klar geht nicht wenn man Windows nutzt und der Code nur für Mac/Linux ist. Genauso wenn Zusatzsoftware nötig ist. Eventuell auch nur testen, ob ein Compilen überhaupt möglich ist (eine dummy.exe erzeugen).

Wenn ein Code eine Beispieldatei benötigt, dann diese wie die PB-Datei benennen. Siehe Graphics\2d\Dot*

Übrigens, in der .gitignore sind folge Endungen ausgenommen:

```
# User-specific files
*.exe
*.pre.pb
*.temp.*
*.bat
```

Also falls man irgendeine Info-Datei anlegen will, nenn sie bspw. MeineInfos.temp.txt - das wird dann automatisch ignoriert.

Es geht übrigens nicht darum, jeden Code aufzunehmen. Genauso sollten für eine Lösung nicht 10 Codes auftauchen (wie bspw. in alten CodeArchiv für SendKey).

In den alten Threads würde ich keinen Post der Marke "Aufgenommen in CodeArchiv" posten und so die alten Threads hochpushen. Das würde ich später nur bei neuen Threads machen.

Achja: Die placeholder.txt sind nur dazu da, das die Ordner synchronisiert werden. Git ignoriert ansonsten leere Ordner. Wenn du in einen solchen Order eine PB oder PBI erstellst, kann man die Placerholder.txt einfach löschen.

Da es aktuell darum geht, die alten Threads abzuarbeiten, ist eine Absprache nötig. Der einfachhaltshalber würde ich Blöcke verteilen. Bspw. von 04.05.2015, 22:14 bis 27.01.2015, 21:17. Gemeint ist hier der „Letzter Beitrag“-Zeitpunkt von Forum. Das würde aktuell folgende Threads von Seite Zwei beinhalten: „Module: Silbentrennung“ bis „Duplikate in List() löschen (alle OS, Anfänger-Tip)“. Falls mal ein älterer Thread eine Antwort erhält, macht das nichts, er wird ja so nach oben durchsortiert und wir wissen Bescheid, das wir ihn noch aufnehmen müssen.

Falls ein neuer Thread bearbeitet wird, sollte man ihn trotzdem erstmal ruhen lassen, meist entwickeln sich in den ersten Tagen noch eine Diskussion und Verbesserungen, die man erstmal abwarten sollte. Wenn so ein aktueller Thread aufgenommen wird, sollte man ein kurzes „Wird in CodeArchiv unter <Pfad&Name> aufgenommen“ reinposten. Damit wissen alle Bescheid. Wie gesagt nur bei neuen Threads und nicht bei Monate/Jahre alten.