

# Inhaltsverzeichnis

Vorbereitungen.....	2
SourceTree zurücksetzen (falls nötig).....	2
Windows:.....	2
SourceTree vorbereiten.....	2
Die CodeArchiv Repo holen.....	5
Checkout und Zweige.....	6
Unser erster Commit.....	8
Auf Github veröffentlichen.....	10
Konflikte lösen.....	12
SourceTree lizenzieren.....	15
Wie sollte man in CodeArchiv arbeiten.....	17
Das Clonen schlägt fehl (templates not found).....	18

## 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>\AppData\Roaming\Atlassian

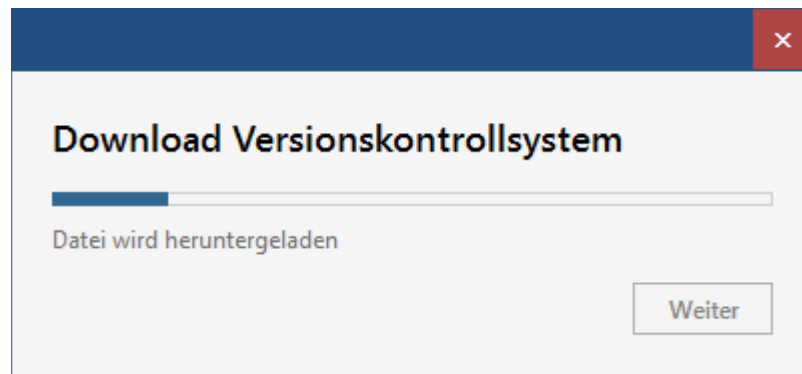
C:\ProgramData\Atlassian

## SourceTree vorbereiten

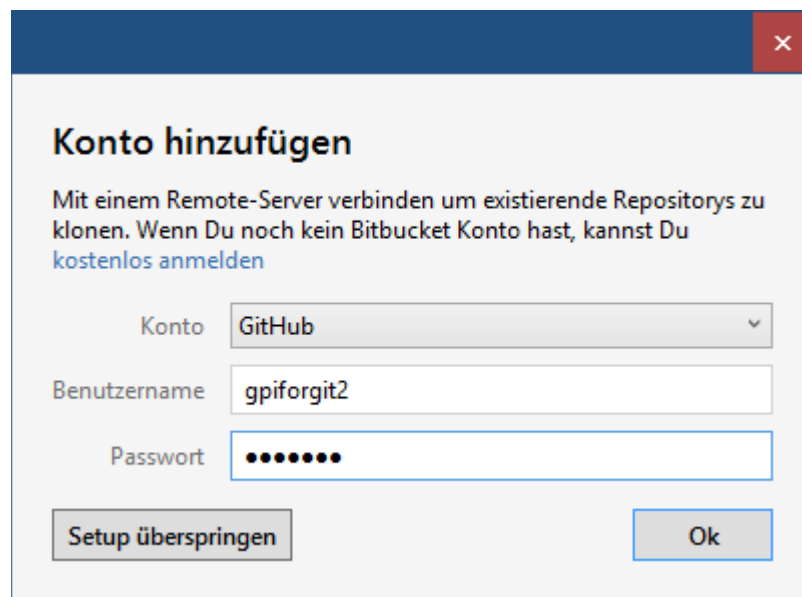


Die Installation sollte nicht weiter kompliziert sein, es ist eine typische WWF (weiter, weiter, fertig) Installation. In Anschluss CodeTree starten.

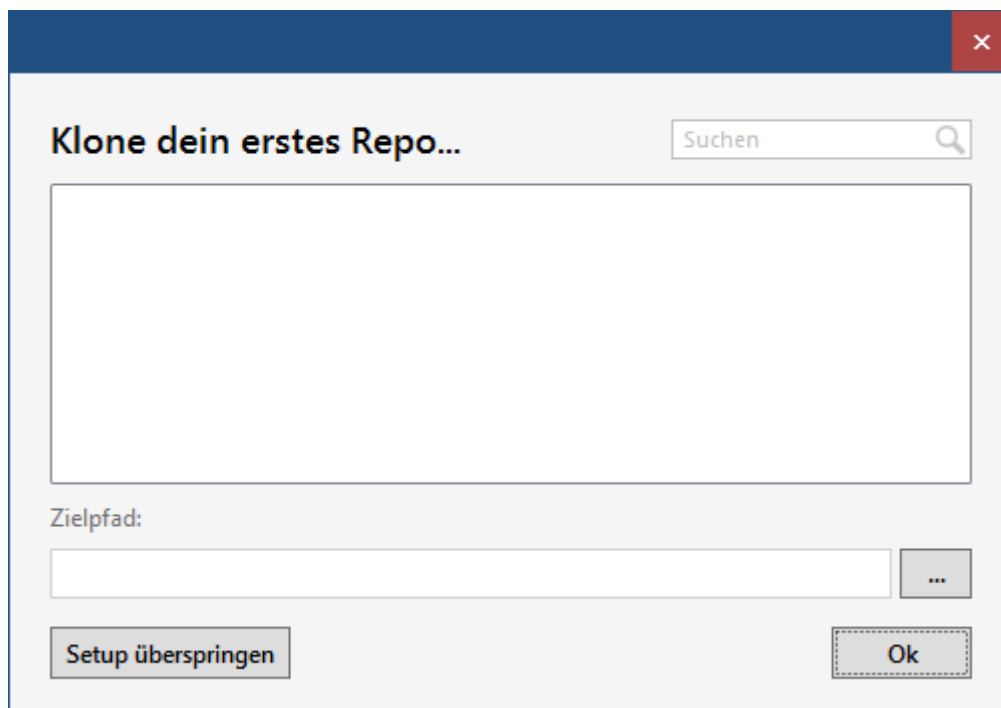
Wir werden wohl die Vereinbarung akzeptieren müssen. Anschließend auf Fortfahren.



Die GIT-Engine wird automatisch gedownloaded

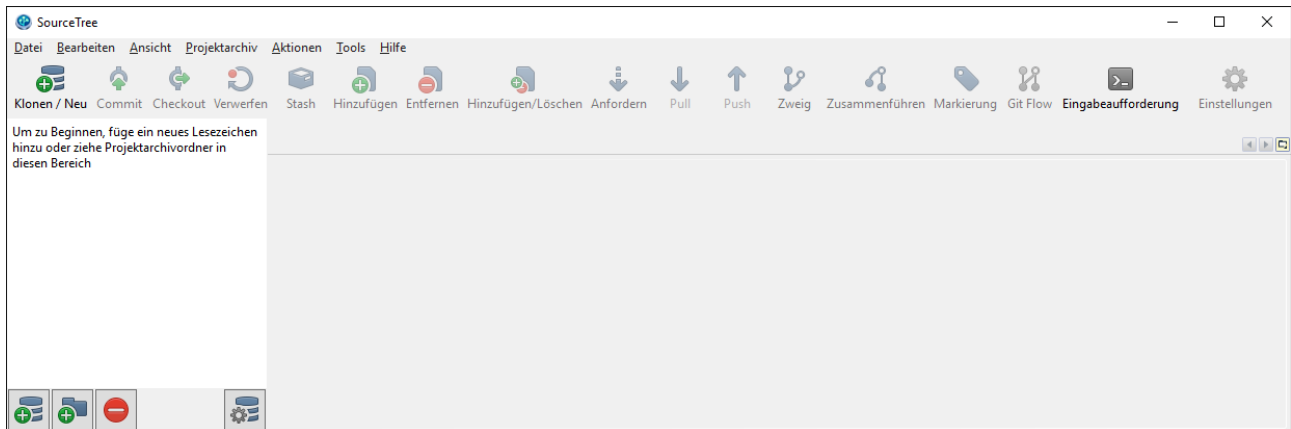


Konto GitHub auswählen, den Benutzernamen (nicht Email) und Passwort eingeben. Anschließend mit OK bestätigen.

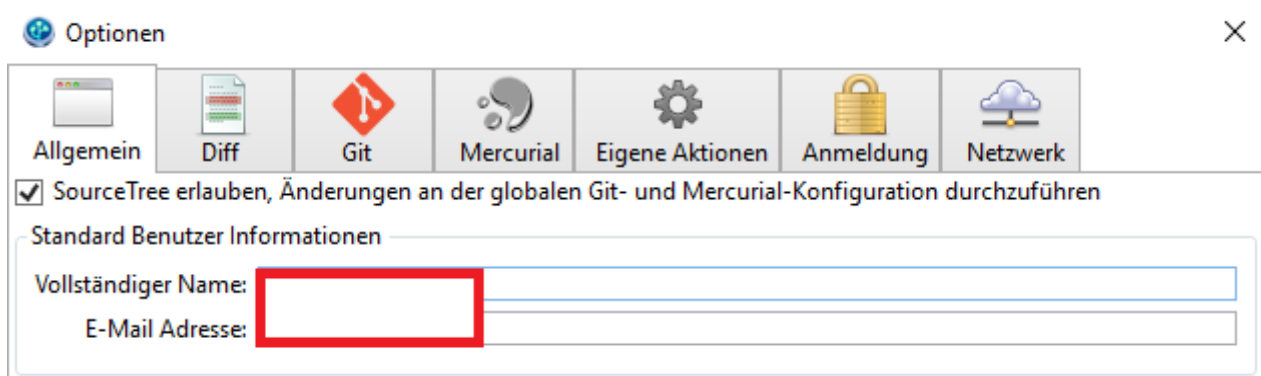


Da wir ja die CodeArchiv-Repo benutzen wollen, überspringen wir hier das Setup,

SourceTree sollten uns jetzt mit den Default-Bildschirm begrüßen:

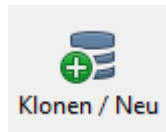


Als erstes sollten wir noch eine Einstellung vornehmen. Dazu in Menü oben Tools und Optionen auswählen.

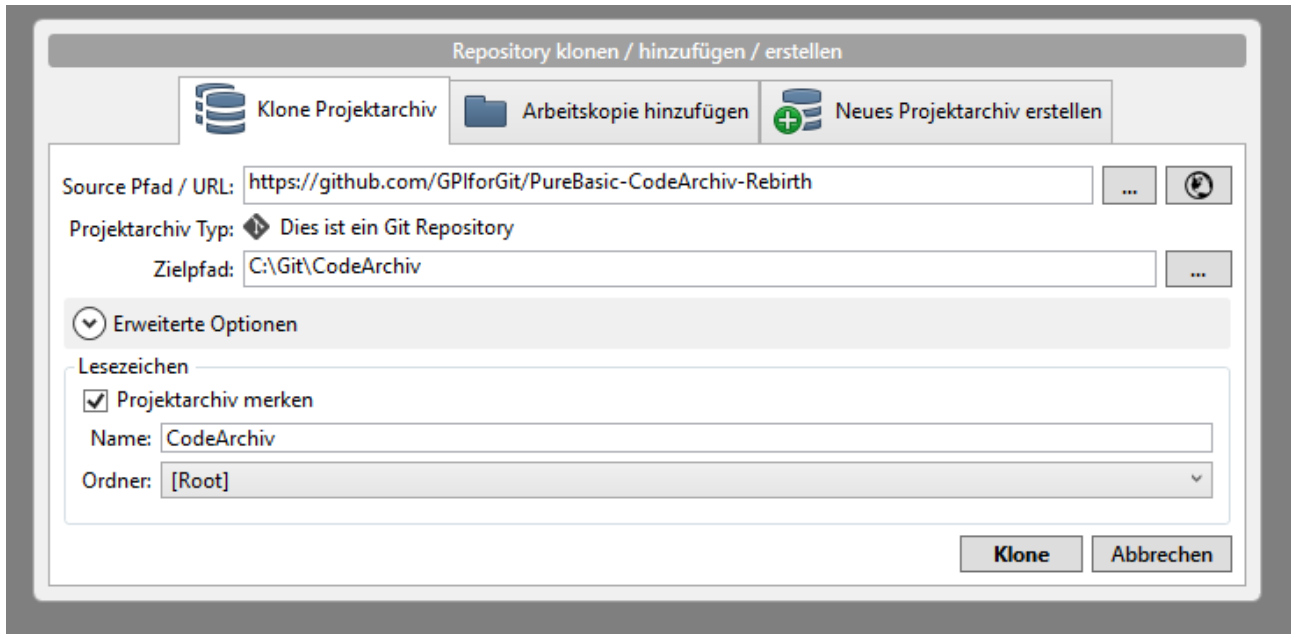


Hier müssen wir unbedingt unter Standard-Benutzer den GitHub-Benutzernamen und die E-Mail Adresse ergänzen.

## Die CodeArchiv Repo holen



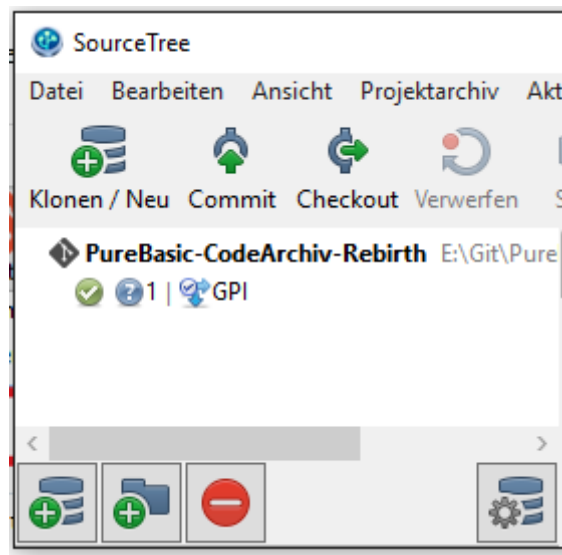
Dazu klicken wir auf das „Klonen / Neu“-Icon aus der Toolbar des Hauptfensters und füllen folgendes Fenster entsprechend aus:



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

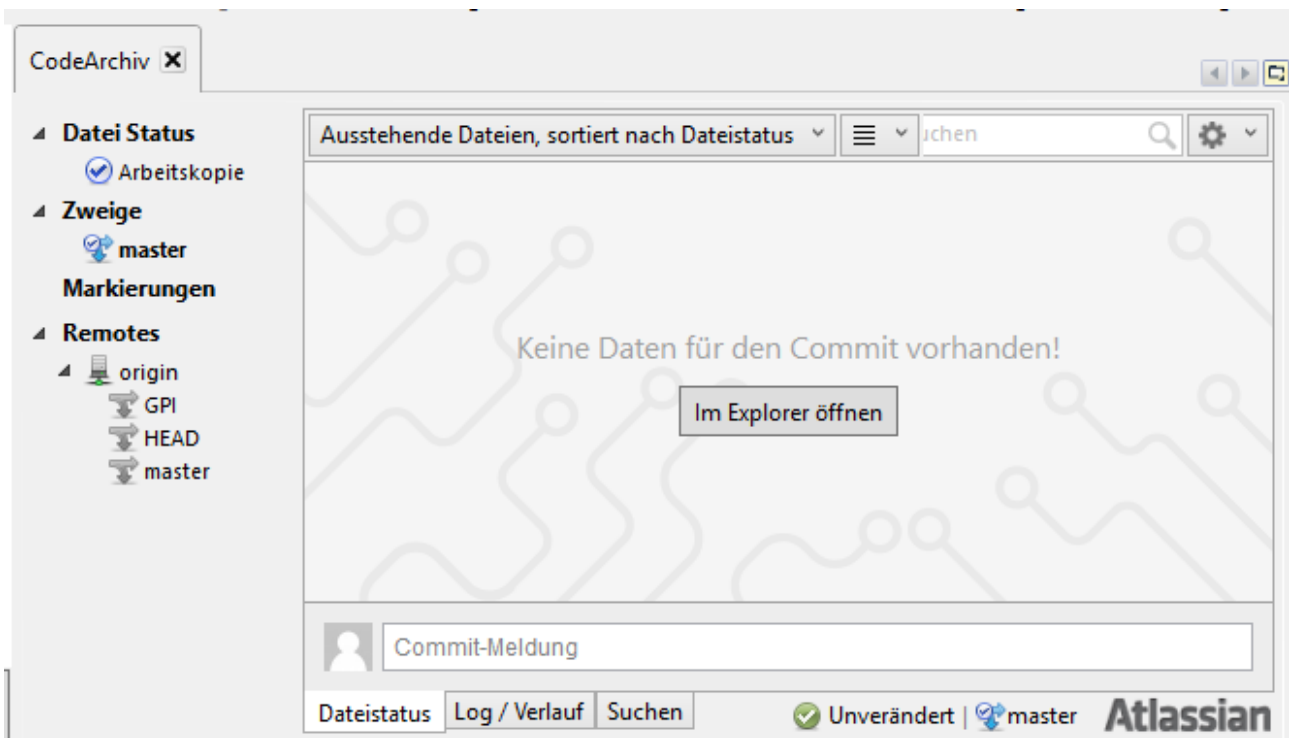
Ein Klick auf Klone und schon wird die Repo kopiert.

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



Falls ein Fehler auftritt das Template-Ordner nicht gefunden werden konnte, dann bitte SourceTree schließen, die eben erstellte lokale Repo löschen und weiter unten in separaten Kapitel nachlesen, wie man den Fehler beheben kann.

## Checkout und Zweige



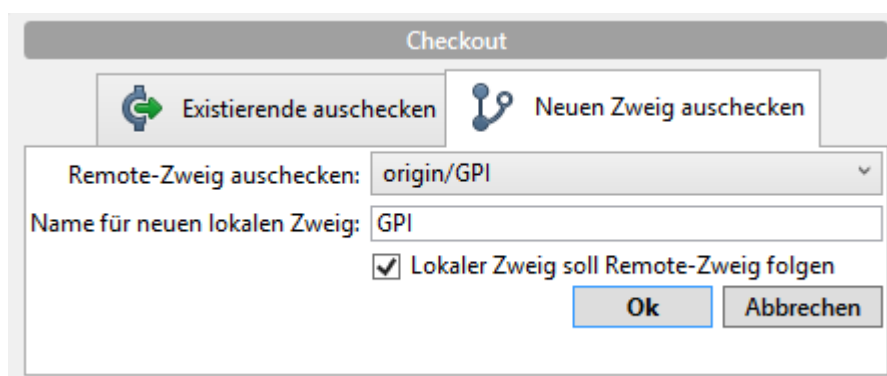
Links sind die verschiedenen Zweige. Aktuell ist der „master“-Zweig aktiv, erkennbar an den fetten Schriftzug. In Remotes kann man die auf GitHub vorhandenen Zweige sehen. Interessant sind hier eigentlich nur GPI – das ist mein Zweig und master – das ist quasi der veröffentlichte Zweig.

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 gedownloaded 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 gedownloaded 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:



**Zweig**

Neuer Zweig   Zweige löschen

Aktueller Zweig: master

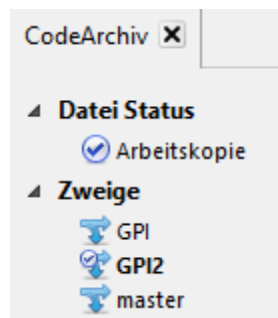
Neuer Zweig:

Commit: ☒ Arbeitskopie Parent  
☐ Angegebener Commit:  ...

☒ Neuen Zweig auschecken

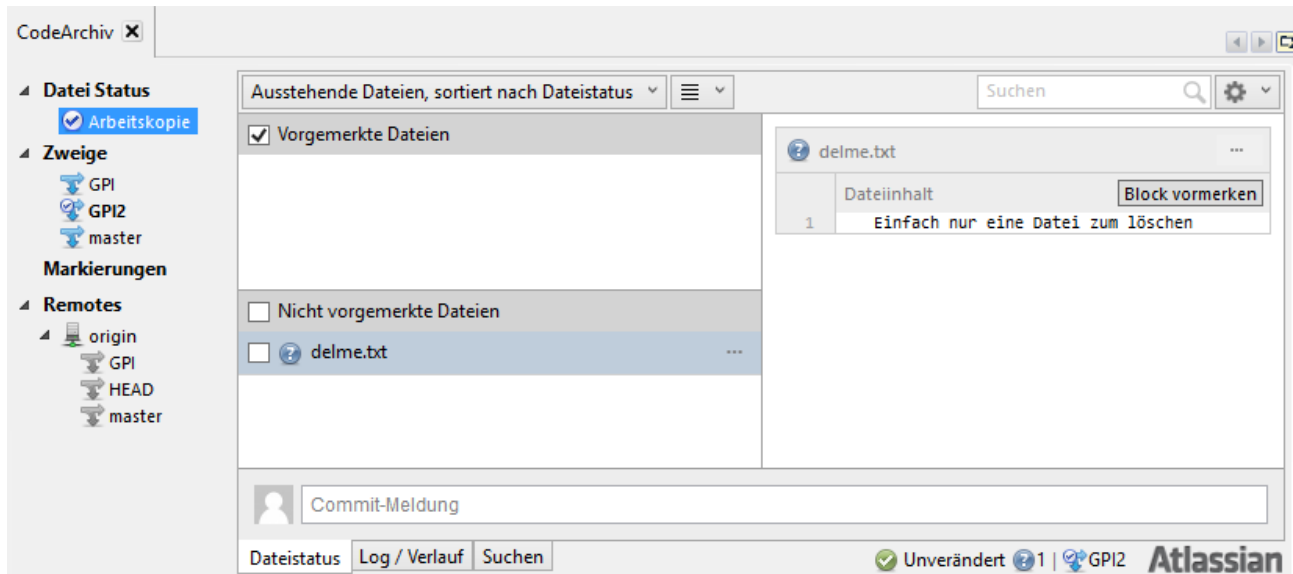
**Zweig erstellen** **Abbrechen**

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



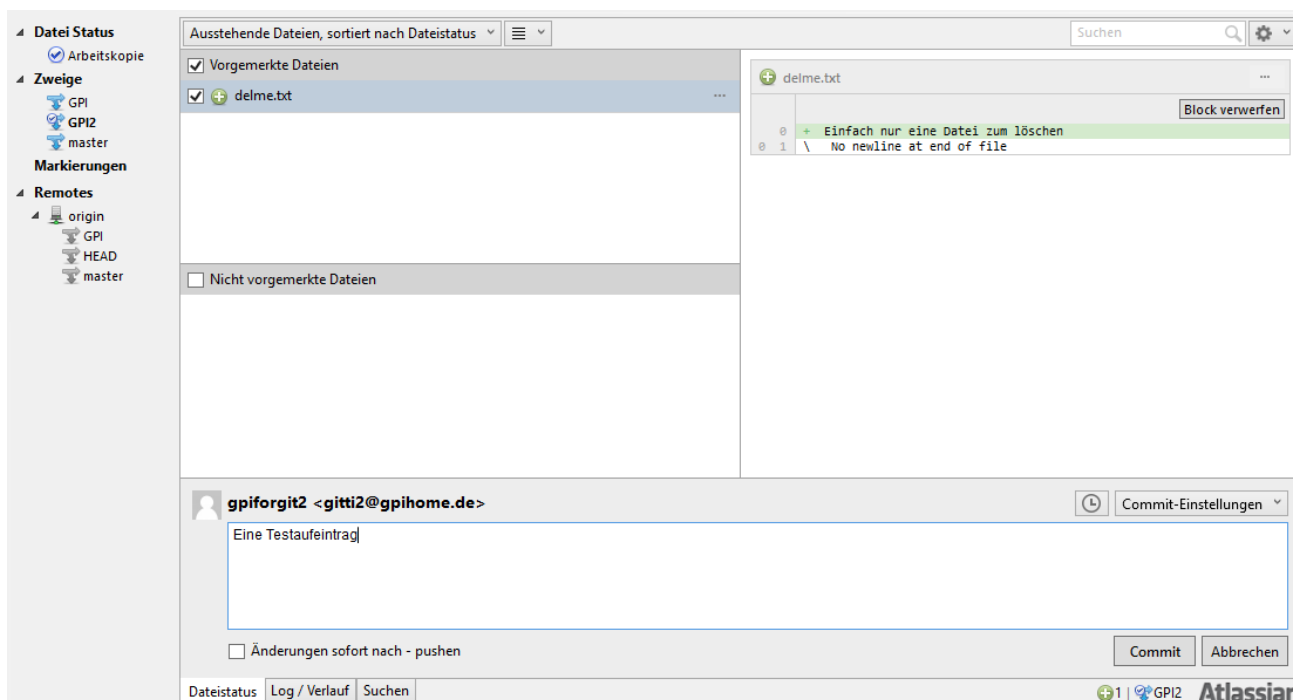
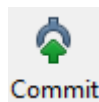
# Unser erster Commit

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.

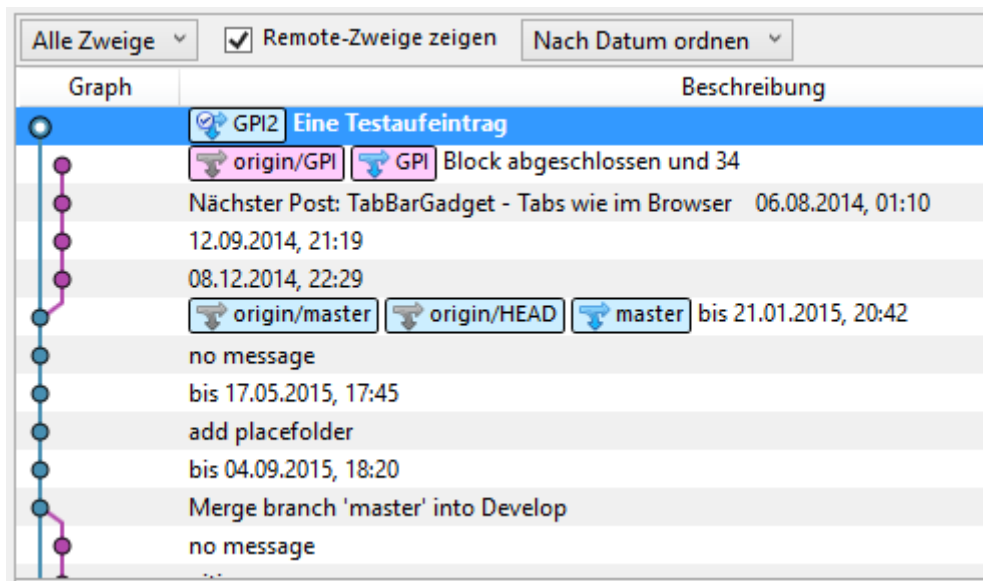


Die Dateien auswählen, so das sie oben stehen, unten in der Beschreibung reinschreiben, was man



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.

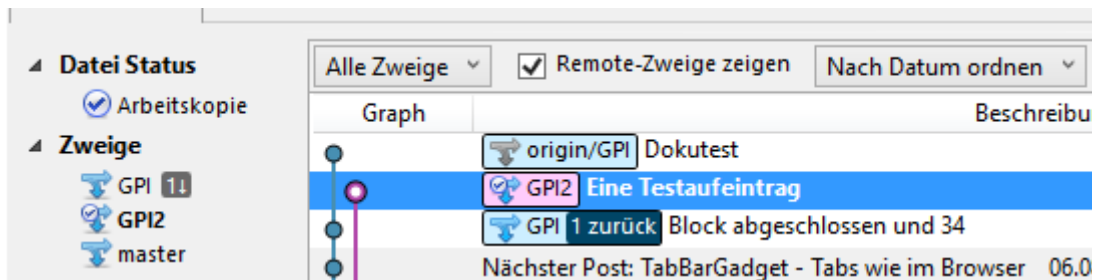
## Auf Github veröffentlichen

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:

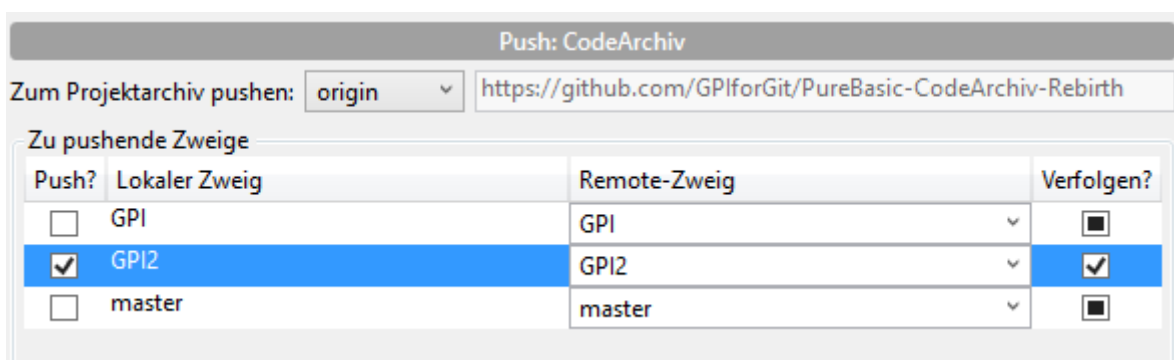
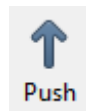


Hier sieht man, das man in Zweig GPI ein neuer Commit ist. Da wir ihn eh nicht nutzen, ignorieren wir ihn einfach.

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:



Authenticate

Login required for: github.com

Username:

Password:

☒ Remember password

Login Cancel

Und schon ist der neue Zweig auf dem Server:

**Datei Status**

- Arbeitskopie

**Zweige**

- GPI
- GPI2
- master

**Markierungen**

**Remotes**

- origin
  - GPI
  - GPI2
  - HEAD
  - master

**Commit History**

Alle Zweige Remote-Zweige zeigen Nach Datum sortieren

Graph	Commit Message
[icon] origin/GPI [icon] GPI Dokutest	
[icon] GPI2 [icon] origin/GPI2 Eine Testaufeintrag	
	Block abgeschlossen und 34
	Nächster Post: TabBarGadget - Tabs wie im Brow
	12.09.2014, 21:19
	08.12.2014, 22:29
[icon] origin/master [icon] origin/HEAD [icon] master	
	no message
	bis 17.05.2015, 17:45
	add placefolder
	bis 04.09.2015, 18:20
	Merge branch 'master' into Develop

## 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!

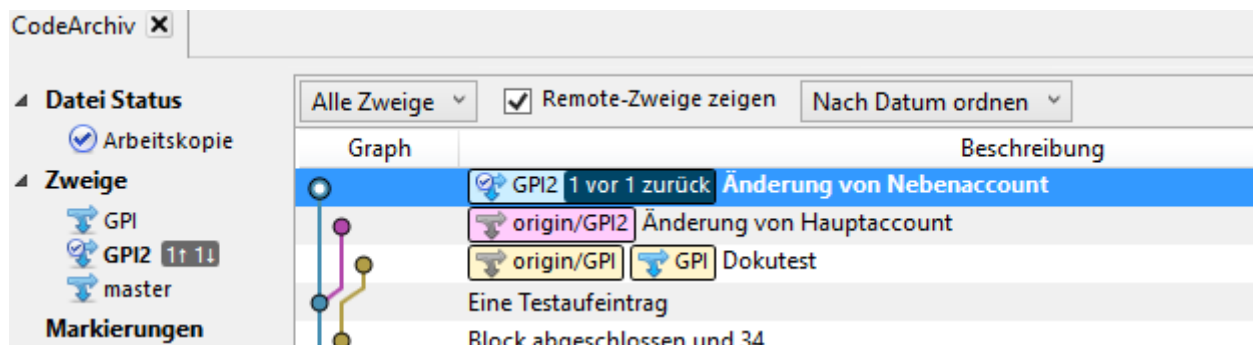
```
Pushing
[Abbrechen]
[ ] Vollständige Ausgabe anzeigen
git -c diff.mnemonicprefix=false -c core.quotePath=false push -v --tags origin GPI2:GPI2
Pushing to https://github.com/GPIforGit/PureBasic-CodeArchiv-Rebirth
To https://github.com/GPIforGit/PureBasic-CodeArchiv-Rebirth
! [rejected] GPI2 -> GPI2 (fetch first)

error: failed to push some refs to 'https://github.com/GPIforGit/PureBasic-CodeArchiv-Rebirth'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Mit Fehlern abgeschlossen, siehe oben.
[Schließen]
```

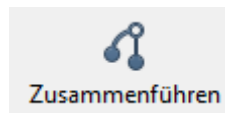
Es konnte ein Commit erzeugt werden, aber der Push schlug fehl, weil zwischenzeitlich ja der Hauptaccount ebenfalls diesen Zweig editiert und minimum eine gleiche Datei geändert hat.

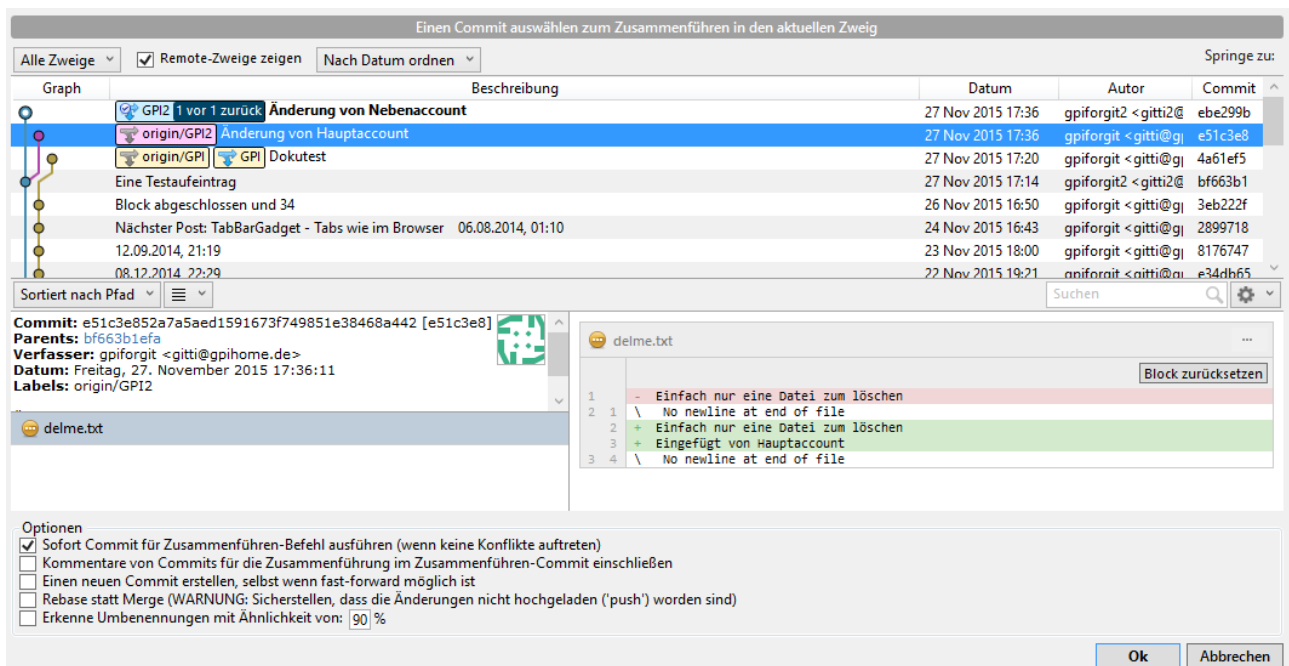
Klicken wir also auf Anfordern (fetch in Englischen) und schauen mal, was passiert:



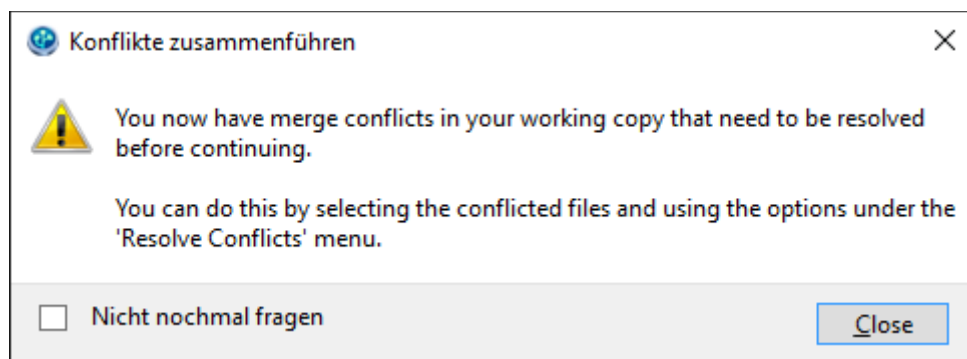
Man sieht schön, das der Zweig GPI2 seit „Eine Testaufeintrag“ sich spaltet. Einmal unsere lokale Kopie und einmal die entfernte origin/GPI2 Zweig (der liegt ja auf Github).

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

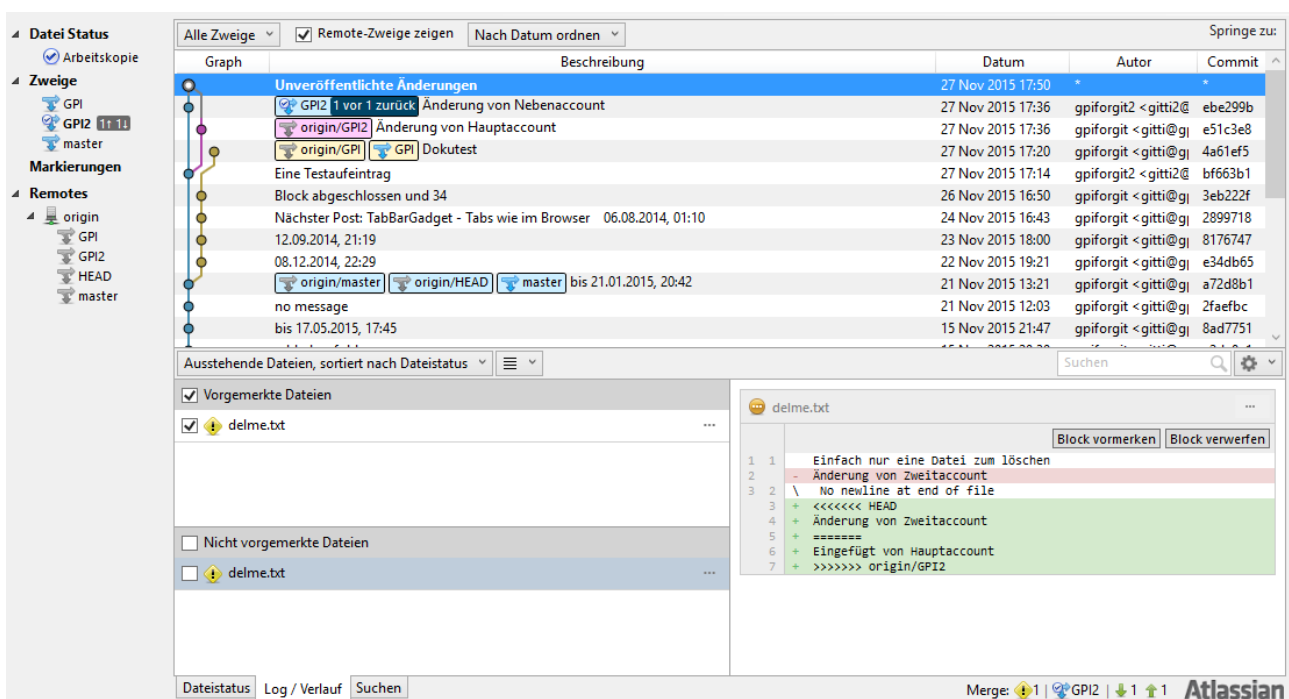




Das sieht jetzt komplizierter aus als es ist. Wir wählen oben den Zweig, den wir mit den aktuellen verbinden wollen und klicken auf OK. Wenn es keine Konflikte gibt, wird automatisch ein neuer Commit erzeugt und alles ist gut. Bei unseren Beispiel ja leider nicht, aber klicken wir erstmal auf ok

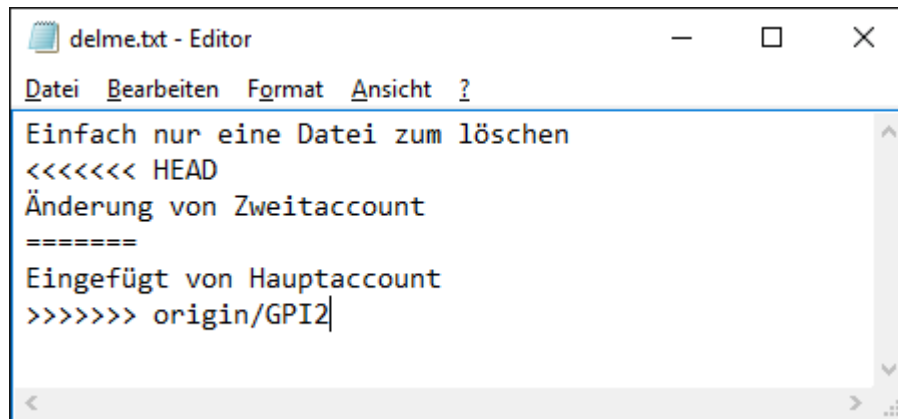


Die erwartete Fehlermeldung. Bei den Zweigen wählen wir „unveränderte Änderungen“:



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

Wenn wir diese mal in Editor anschauen:

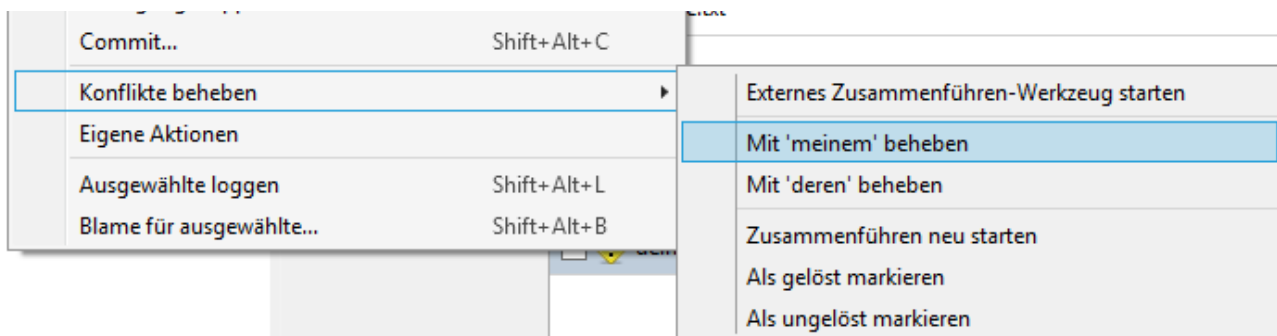


```
delme.txt - Editor
Datei Bearbeiten Format Ansicht ?
Einfach nur eine Datei zum löschen
<<<<<< HEAD
Änderung von Zweitaccount
=====
Eingefügt von Hauptaccount
>>>>>> origin/GPI2
```

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 dass es passt, wir übernehmen die lokale Datei, wir übernehmen die origin-Datei.

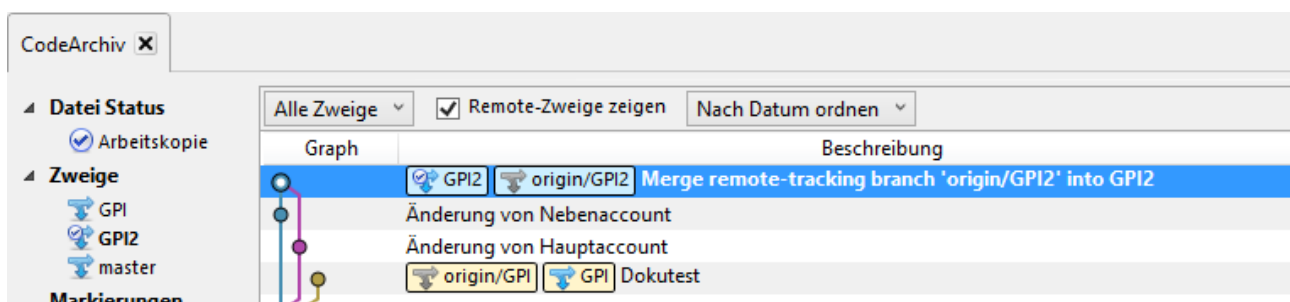
Wir rechtsklicken auf die untere delme.txt.



Hier kann man sagen mit „meinem“ beheben, dass man die lokale haben will, mit deren beheben, dass 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

A few words you'll find easy to remember



Ich bin kein Roboter.



reCAPTCHA

[Datenschutzerklärung](#) - [Nutzungsbedingungen](#)

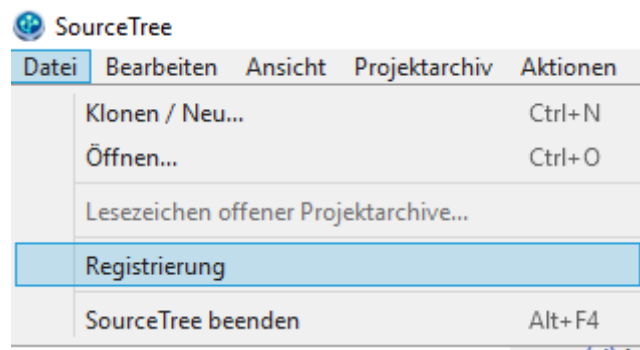
Sign up

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

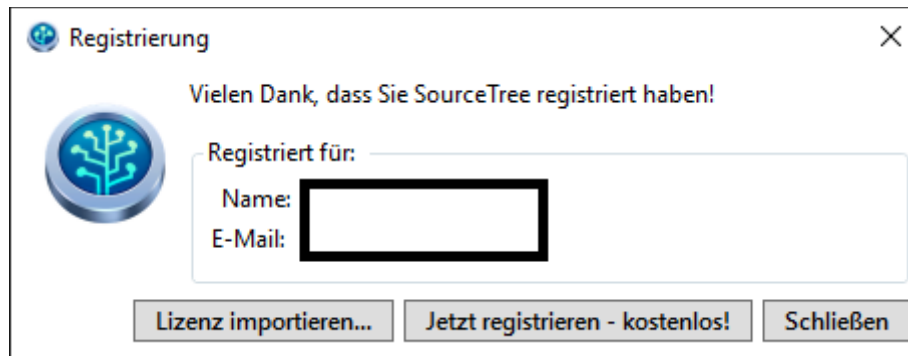
Im Anschluss sollte man eine E-Mail bekommen. Den Link in der E-Mail folgen und man kommt auf diese Seite:

SEN	Product	Name	Support Expires	Support
▼ SEN-6227529	SourceTree: Free License	<div></div>	28 Nov 2015	
<div><div><div>Name <div></div></div><div>SEN <div></div></div><div>Technical Contact <div></div><div>contact@example.com</div><div>Add</div></div><div>Billing Contact <div></div><div>contact@example.com</div><div>Add</div></div></div><div><div>Licensed To <div></div></div><div>License Key <a href="#">Download your SourceTree license.</a> Within SourceTree select from the menu bar "SourceTree &gt; Registration", click "Import License" to import the downloaded the license file.</div><div>Actions <a href="#">Archive</a></div></div></div>				

Der große rote Pfeil zeigt auf den Downloadlink. Sobald diese Datei gedownloadet 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 folgende 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.

## Das Clonen schlägt fehl (templates not found)

Es gibt wohl einen recht komischen Bug. Wenn man eine Repo clonen will, kann es passieren, dass es fehlschlägt. Wenn man jetzt weitere Infos anschaut, dann findet man gleich zu Beginn in etwa das „warning: templates not found /usr/local/git/share/git-core/templates“.

Bei Windows sollte man in

`c:\users\<account>\AppData\Local\Atlassian\SourceTree\git_local\mingw32`

wechseln und den kompletten Inhalt eine Ebene tiefer kopieren – also nach

`c:\users\<account>\AppData\Local\Atlassian\SourceTree\git_local\`

Wichtig: Bestehende Dateien nicht überschreiben!