Git – Dezentrale Versionsverwaltung

Installation von Git für unsere OpenDCC/Fichtelbahn Entwickler

Stephan Bauer, Christoph Schörner, Andreas Kuhtz

05.09.2023

Installation von Git für unsere OpenDCC/Fichtelbahn Entwickler

Inhalt

1. Anmeldung bei GitHub

Für die Verwendung von GitHub benötigen Sie einen Benutzerzugang.
Dies passiert durch eine Anmeldung auf http://www.github.com.

1) Als Benutzername sollte der **OpenDCC-Forumsbenutzer-Name** verwendet werden. Falls dieser Name nicht verfügbar ist, sollte ein Namen gewählt werden, der dem realen Namen identisch ist.

2) Nach der erfolgreichen Anmeldung übermitteln Sie bitte an support@fichtelbahn.de den gewählten Benutzername.
Wir ermöglichen darauf einen Zugang in die BiDiB-Group.

2. Installation von Git

Sie benötigen für den Datenaustausch das **Tool Git**.

Diese Software kann auf http://git-scm.com/ heruntergeladen werden.

Führen Sie im Anschluss die Installation aus.

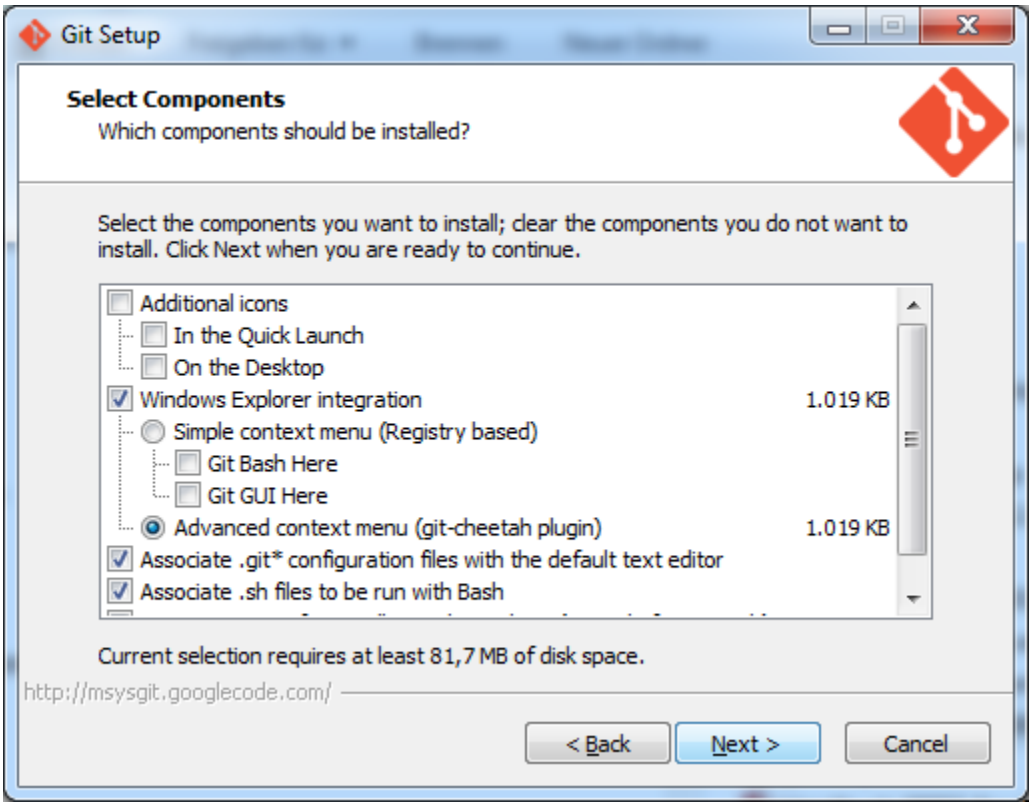

Abbildung 1: Git Setup Teil1

Der einzige Unterschied bei dieser Auswahl sind zusätzliche Einträge in PATH

Das kann auch nachträglich geändert werden.
(C:\Programme\Git\cmd; C:\Programme\Git\bin)

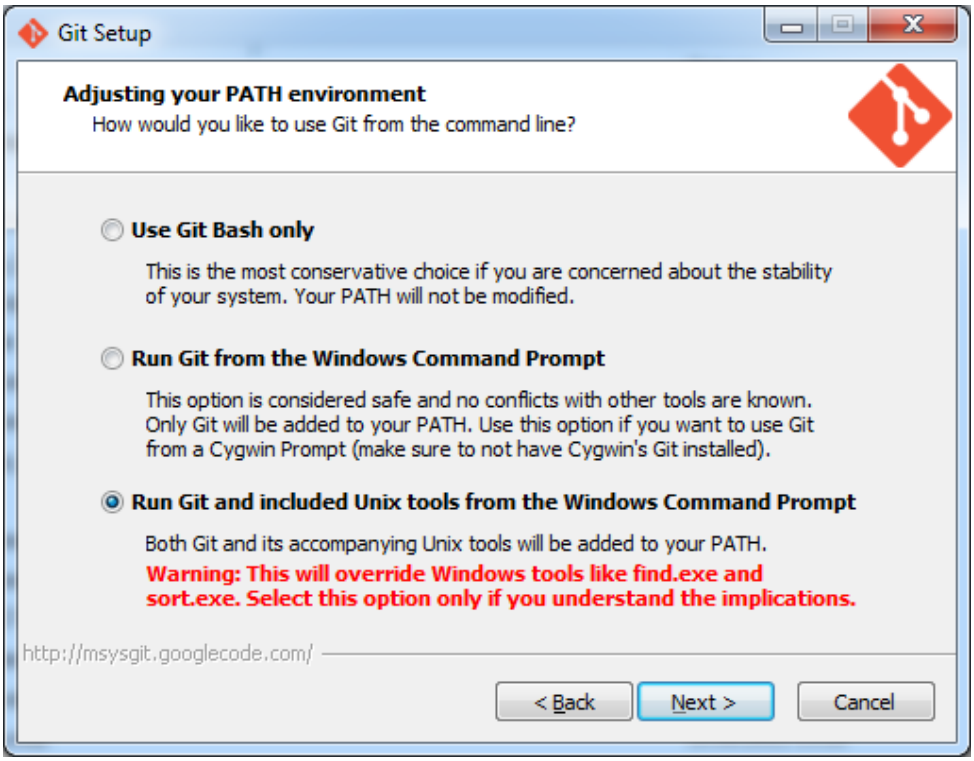

Abbildung 2: Git Setup Teil2

Diese Auswahl erscheint nicht immer:

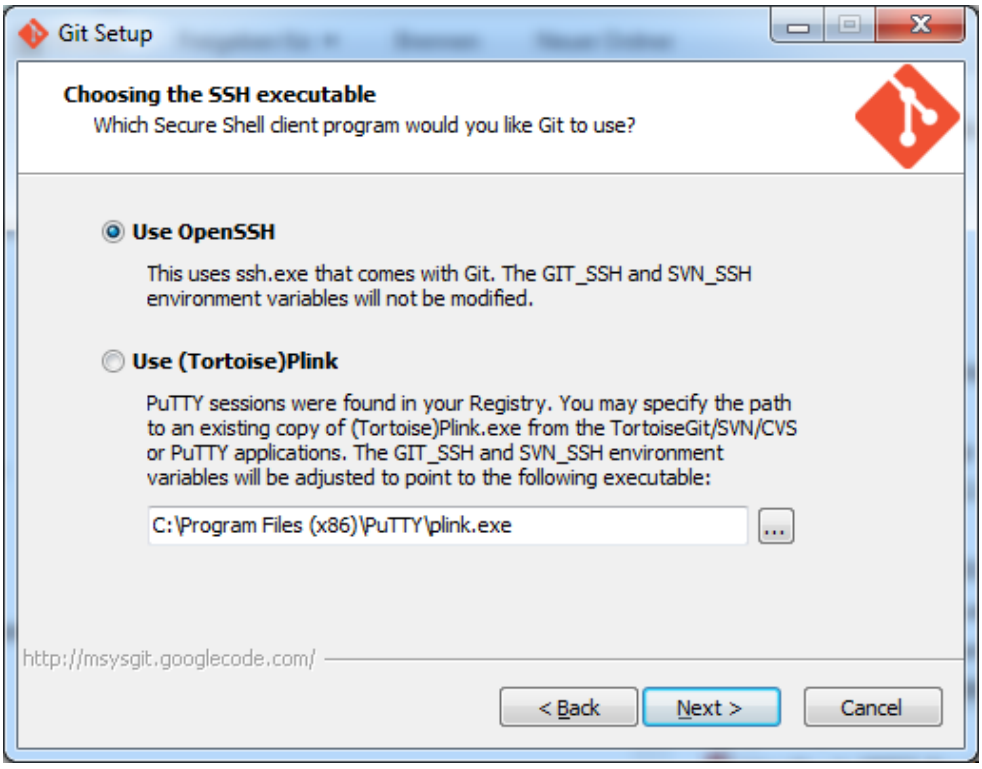

Abbildung 3: Git Setup Teil3

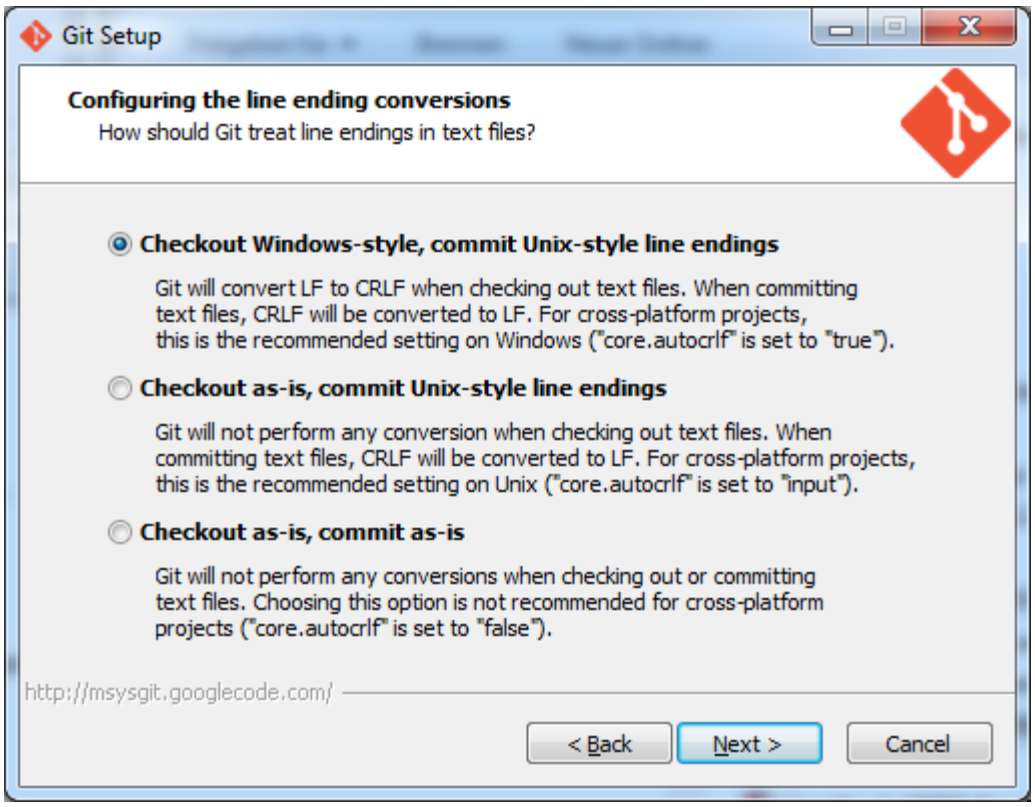

Abbildung 4: Git Setup Teil4

Danach ist die Installation von Git abgeschlossen.

3. SSH Key erstellen

CMD über 'Start->Ausführen' öffnen. Es muss das Home-Verzeichnis ausgewählt sein.

Folgenden Befehl eingeben:

ssh-keygen -t rsa -C "*your_email@example.com*" (Email-Adresse der GitHub-Anmeldung).

Die nachfolgenden Fragen einfach mit „Return" und „Yes" bestätigen:

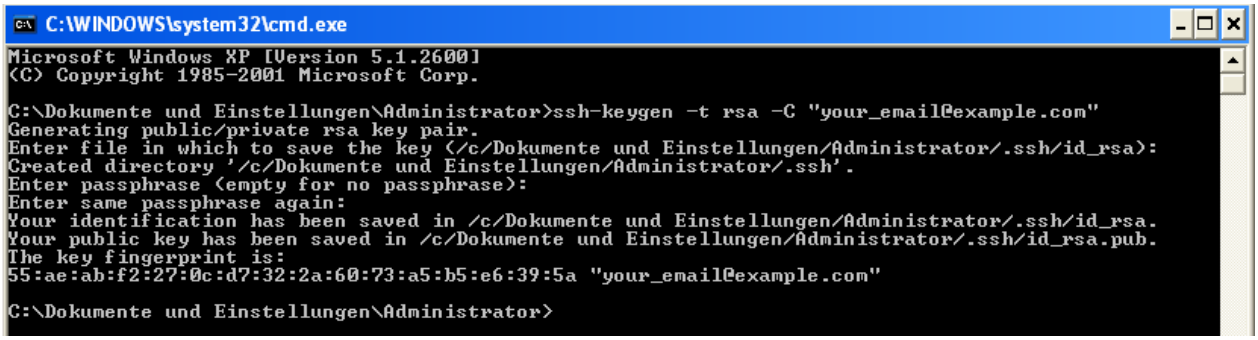

Abbildung 5: SSH Key Teil1

Im Home-Verzeichnis gibt es jetzt einen Ordner '.ssh'. Hier liegen jetzt 2 Dateien, 'id_rsa' und 'id_rsa.pub'.

Die Datei 'id_rsa.pub' wird im Editor geöffnet und der gesamte Inhalt in die Zwischenablage kopiert und bei GitHub in den 'Profile Settings' unter 'SSH Keys' eingetragen.

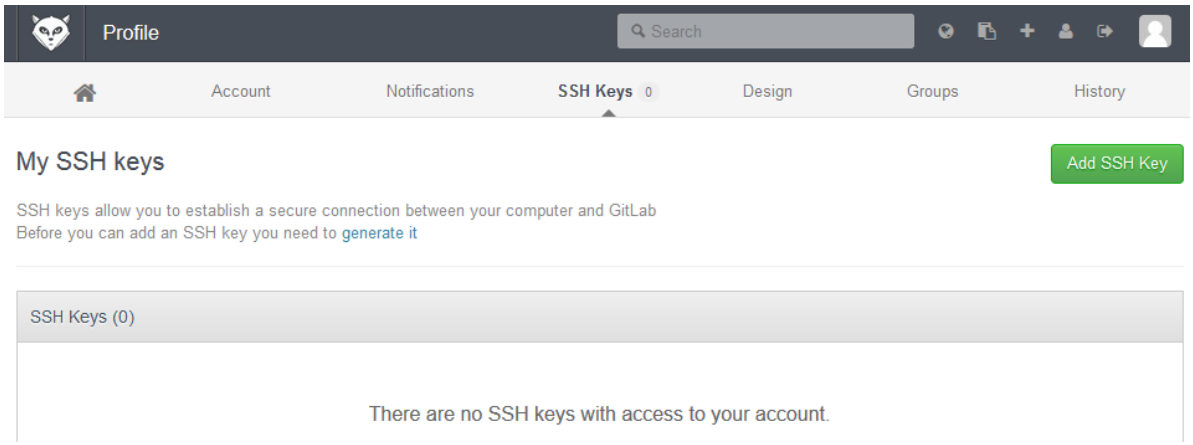

Abbildung 6: SSH Key Teil2

Im Feld 'Title' muss nichts eingetragen werden. Hier wird automatisch die E-Mail-Adresse aus dem Key verwendet. Jetzt kann ein neues Projekt in GitHub angelegt werden und nach der Anleitung initialisiert werden.

4. Initialisierung von GIT

Die Initialisierung muss nur einmalig global erfolgen:

git config --global user.name "Vorname Nachname"

git config --global user.email "*your_email@example.com*"

Beim ersten push oder clone wird ein RSA key im System hinzugefügt (mit 'yes' bestätigen):

```
C:\Users\Stephan\Web\Projekte\Privat\Fichtelbahn\git\OpenDecoder2_V12>git push -u origin master
The authenticity of host 'gitlab.com (54.243.197.170)' can't be established.
RSA key fingerprint is b6:03:0e:39:97:9e:d0:e7:24:ce:a3:77:3e:01:42:09.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'gitlab.com,54.243.197.170' (RSA) to the list of known hosts.
Counting objects: 48, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (48/48), done.
Writing objects: 100% (48/48), 145.62 KiB | 0 bytes/s, done.
Total 48 (delta 10), reused 0 (delta 0)
To git@gitlab.com:bidib/opendecoder2.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

Abbildung 7: Initialisierung Teil1

Achtung:

bei 'git commit -m „first commit“' unter Windows unbedingt die doppelten Anführungsstriche verwenden. Es gibt sonst eine Fehlermeldung.

Neue Projekte in einer Gruppe kann nur der 'Owner' anlegen.

5. GIT mit Comand Line

Bestehendes Projekt klonen:

git clone git@github.com:BiDiBorg/*projektname*.git

z.B: git clone git@github.com:BiDiBorg/opendecoder2.git

Jetzt können Änderungen gemacht werden.

Danach mit git add * die Änderungen im lokalen Index hinzufügen und mit

git commit -m "Commit-Nachricht" dem lokalen Repository hinzufügen.

Mit git push origin master werden die Änderungen zu GitHub übertragen.

Änderungen können in der Web-Oberfläche angesehen werden.

Anleitungen:

http://rogerdudler.github.io/git-guide/index.de.html

http://git-scm.com/book/de

6. GIT mit Windows GUI

Das GIT GUI für Windows wird bei der Installation mitgeliefert.

Abbildung 8: Git GUI Teil1

Starten Sie das **Git GUI** und klicken Sie auf den Punkt „**Projektarchiv klonen**“.

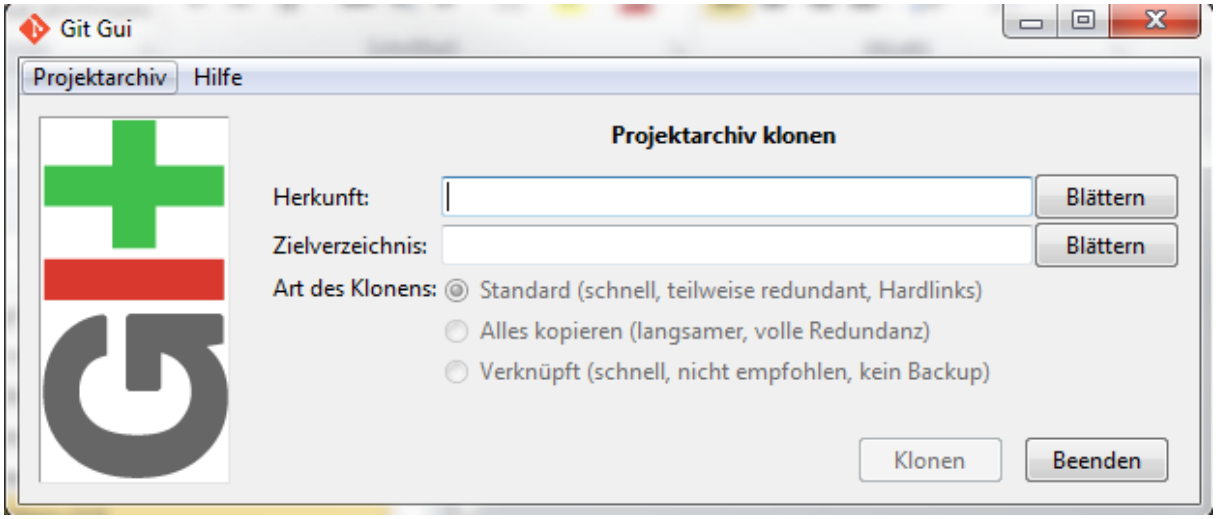

Abbildung 9: Git GUI Teil2

In das Feld **Herkunft** wird der Pfad von dem jeweiligen Projekt aus der GitHub Web-Oberfläche kopiert.

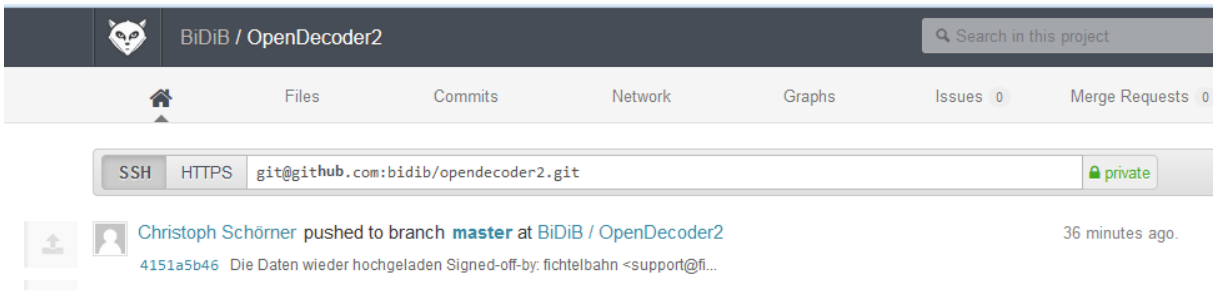

Abbildung 10: Git GUI Teil2

In das Feld **Zielverzeichnis** wird der Speicherort auf dem PC eingetragen.

Der Zielordner (Projektordner) darf nicht vorhanden sein, sondern wird vom Tool angelegt.

Im Anschluss wird mehrmals nach dem Passwort gefragt und die Daten vom Server synchronisiert.

Die weitere Verwaltung kann mit dem GUI erfolgen:

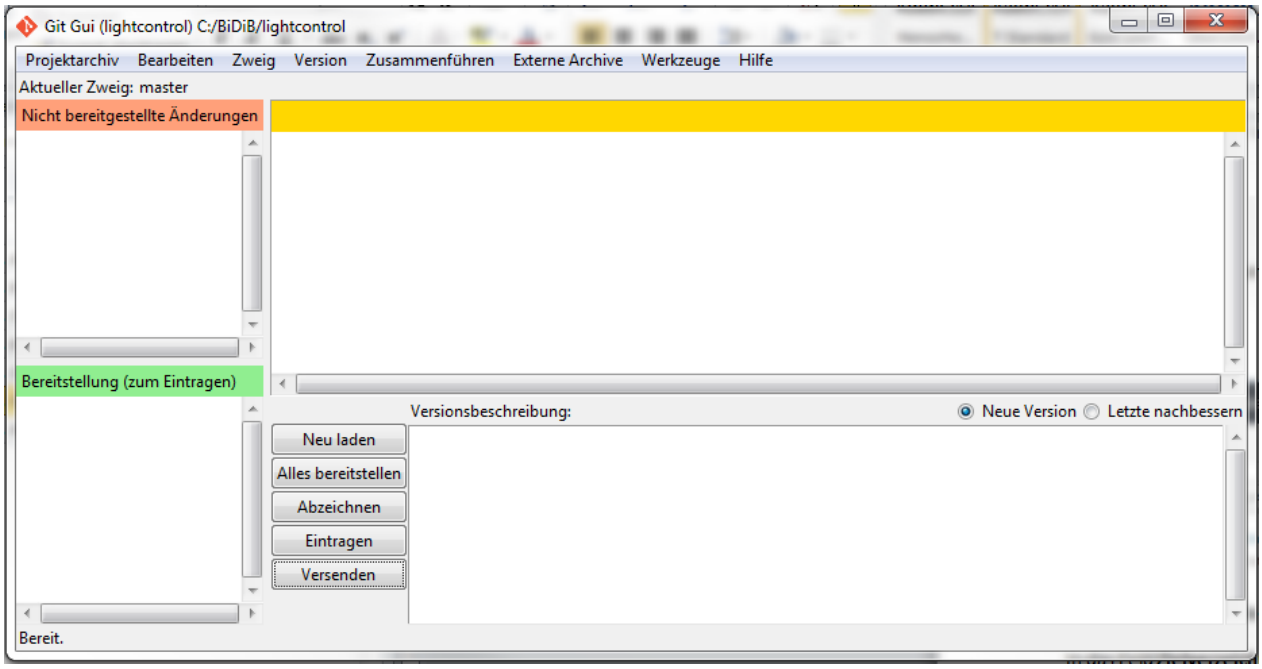

Abbildung 11: Git GUI Teil3

7. GIT mit Windows GUI „SourceTree“

1. SourceTree downloaden und installieren

http://sourcetreeapp.com/download/

Das Tool SourceTree installieren über den Installer (das aktuelle .NET Framework wird installiert, falls nicht vorhanden). Während der Installation kommt eine Anfrage nach Mercurial. Wir benötigen aktuell kein Mercurial.

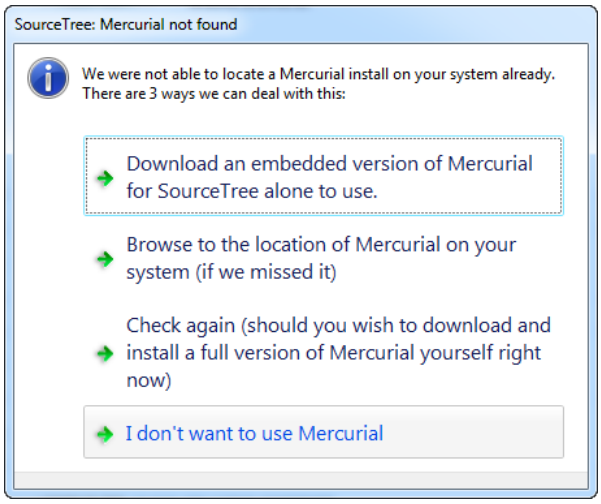

Abbildung 12: Git GUI SourceTree Teil1

Abbildung 13: Git GUI SourceTree Teil1

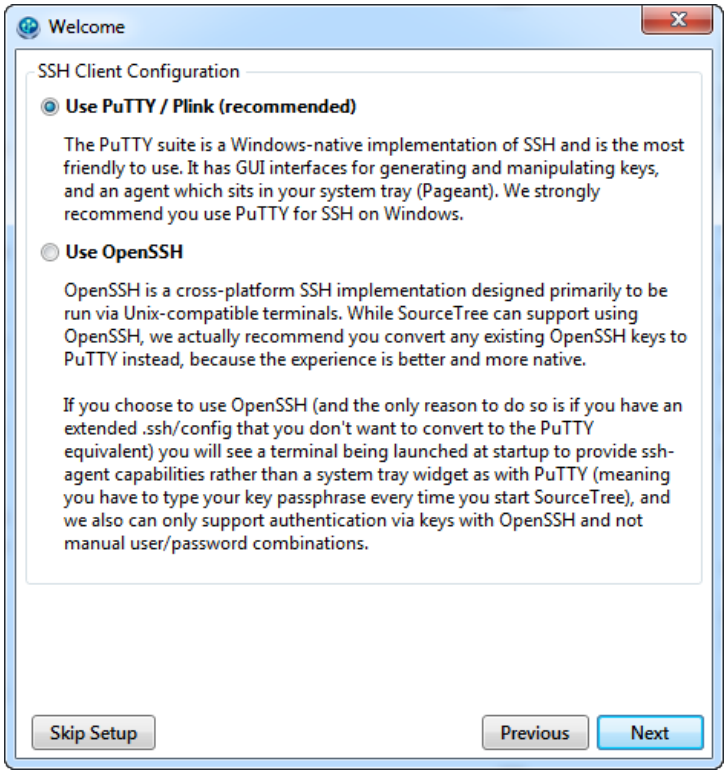

Abbildung 14: Git GUI SourceTree Teil2

Hier kann man den SSH-Key auswählen den man schon im GitHub Account eingetragen hat.

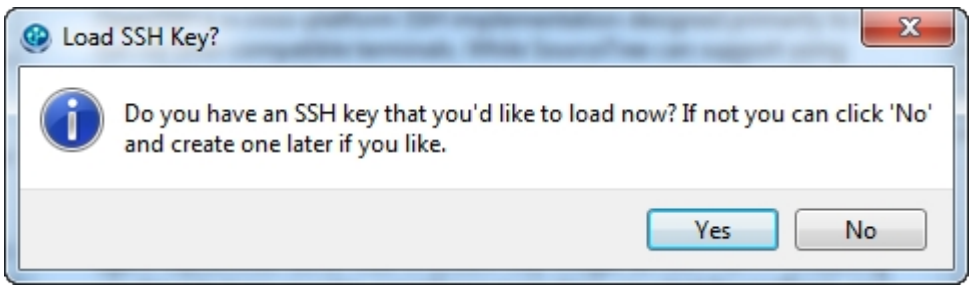

Abbildung 15: Git GUI SourceTree Teil3

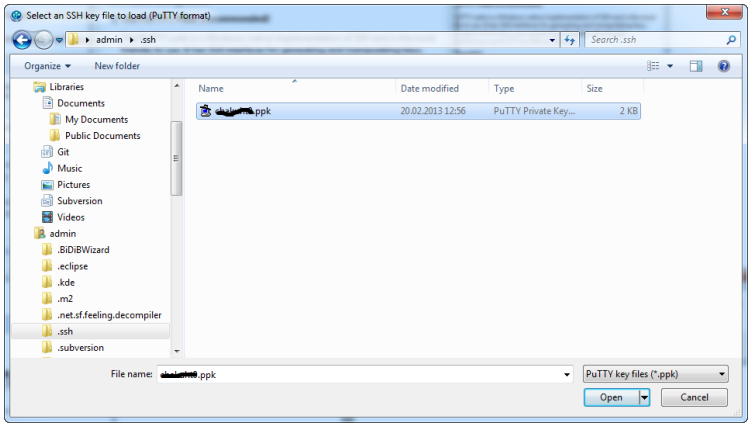

Abbildung 16: Git GUI SourceTree Teil4

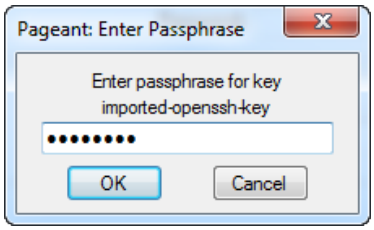

Abbildung 17: Git GUI SourceTree Teil5

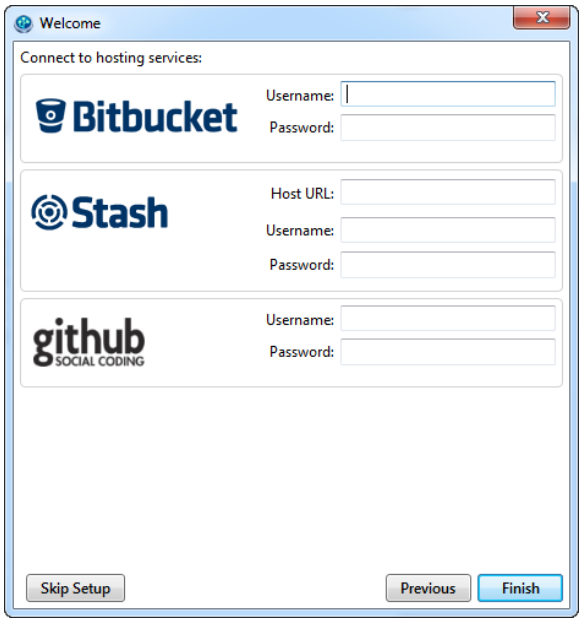

Abbildung 18: Git GUI SourceTree Teil6

Sie benötigen keine weiteren Einstellungen, deshalb genügt ein Klick auf „Finish“. Im Anschluss versucht das Tool noch Connections zu testen, da kann man Skip Test wählen.

2. SourceTree einrichten

Falls die SSH Keys schon vorher importiert wurden, kann dieser Schritt entfallen.

Im Menü unter **Tools** > **Create or Import SSH Keys**

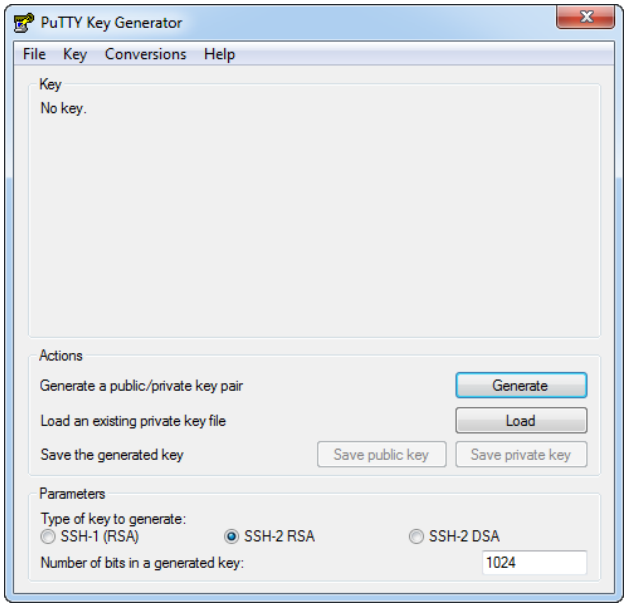

Abbildung 19: Git GUI SourceTree Teil7

Mit Load den SSH Key selektieren.

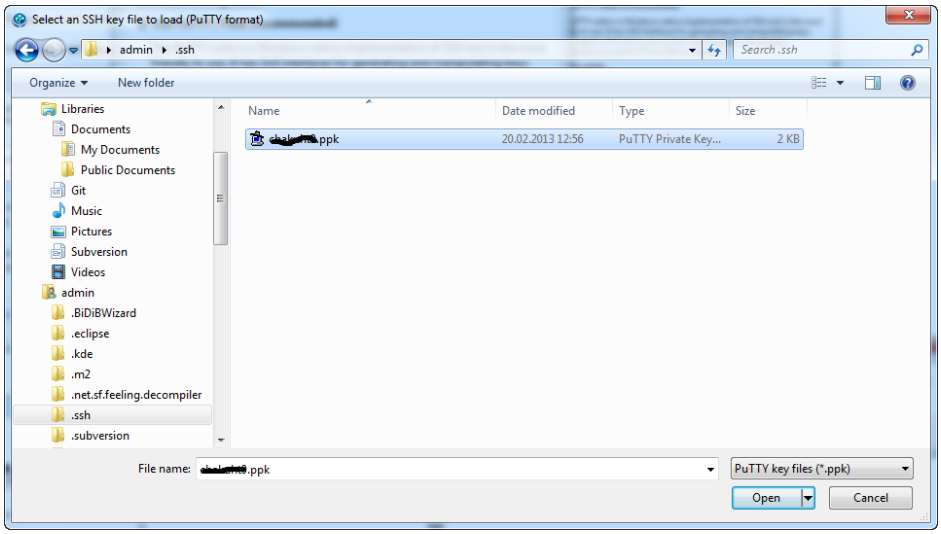

Abbildung 20: Git GUI SourceTree Teil8

3. Repo clonen

Im nächsten Schritt wird das Repo (https://gitlhub.com/bidib/test.git) gecloned, d.h. eine lokale Kopie von Repo erstellt.

Über den Toobar-Button ***Clone / New*** wird der Dialog ***Clone/Add/Create Repository*** geöffnet. In ***Source Path / URL*** wird die URL vom Repo eingegeben (hier: https://github.com/bidib/test.git).

Der Destination Path ist das lokale Repo wohin der Clone (Subversion Syntax: Checkout) gemacht wird.

☝ Das Fenster zur Eingabe der Passphrase kann im Hintergrund geöffnet werden oder bis das Authenticate Fenster kommt, kann auch etwas Zeit vergehen.

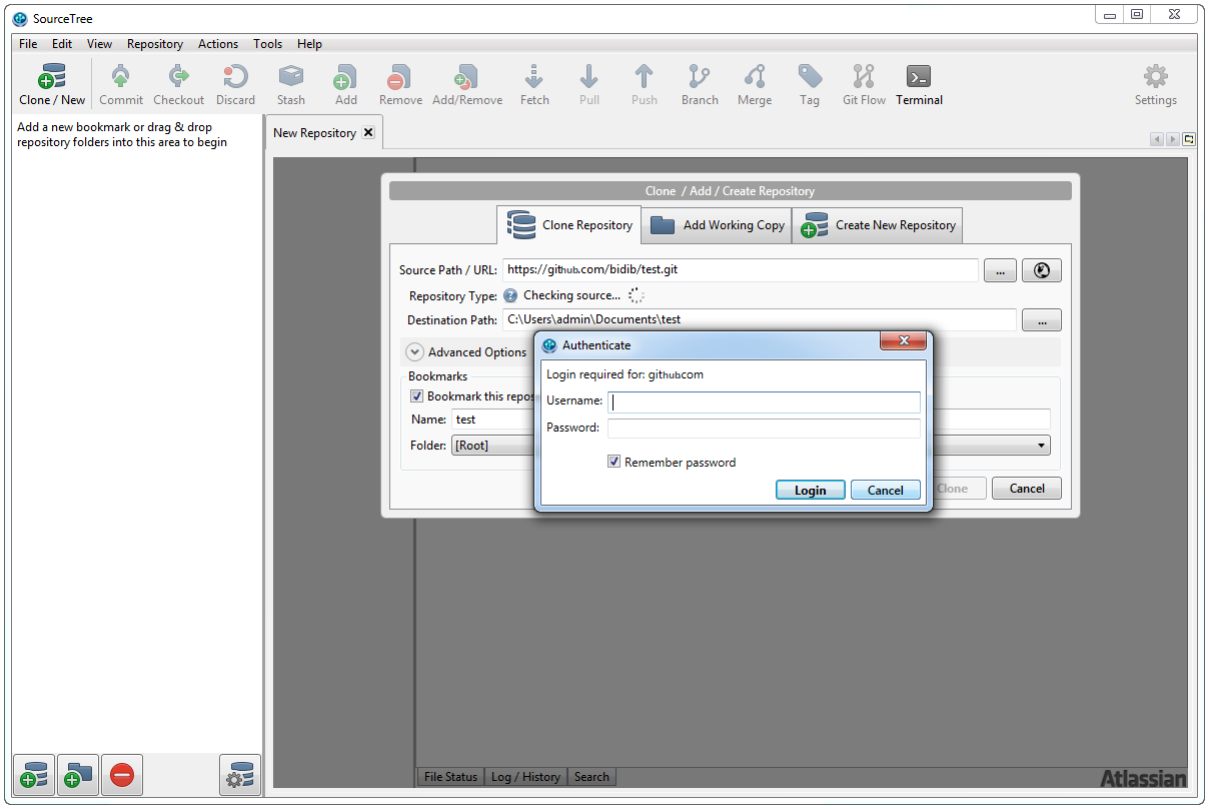

Abbildung 21: Git GUI SourceTree Teil9

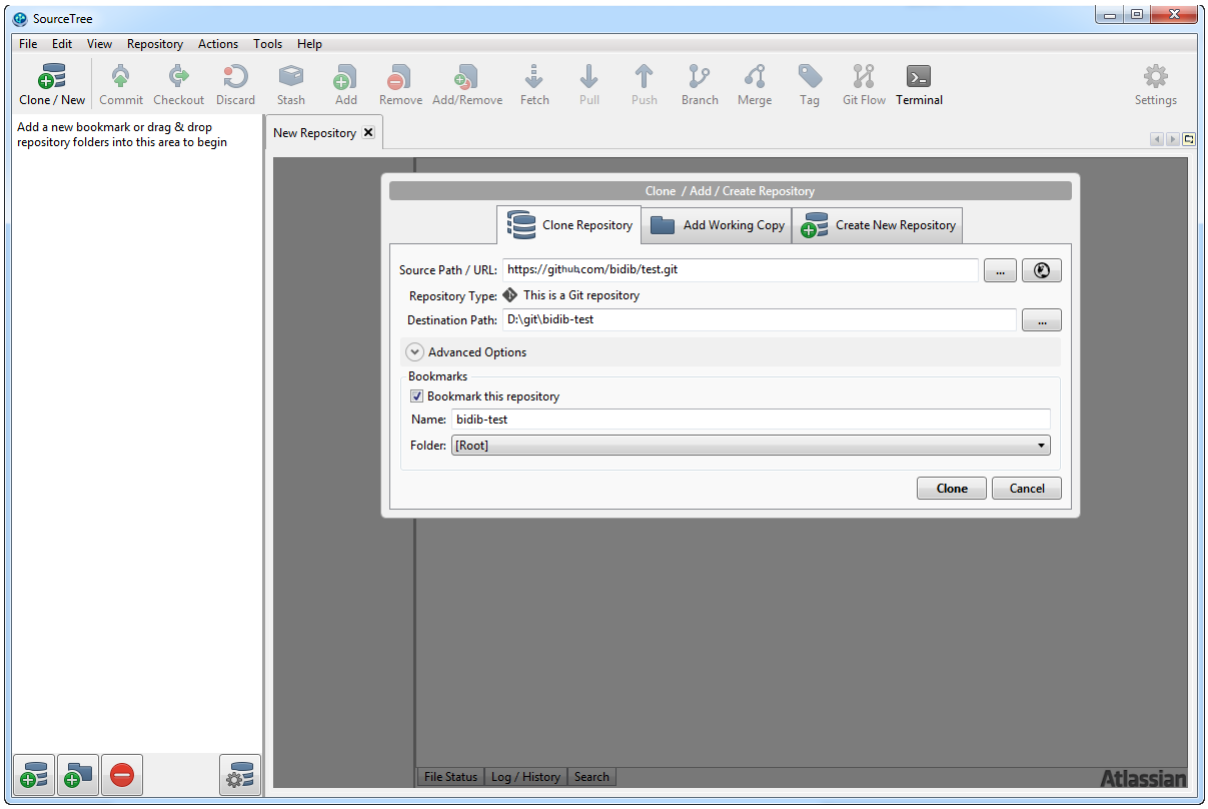

Abbildung 22: Git GUI SourceTree Teil10

Ein Klick auf Clone schließt den Dialog.

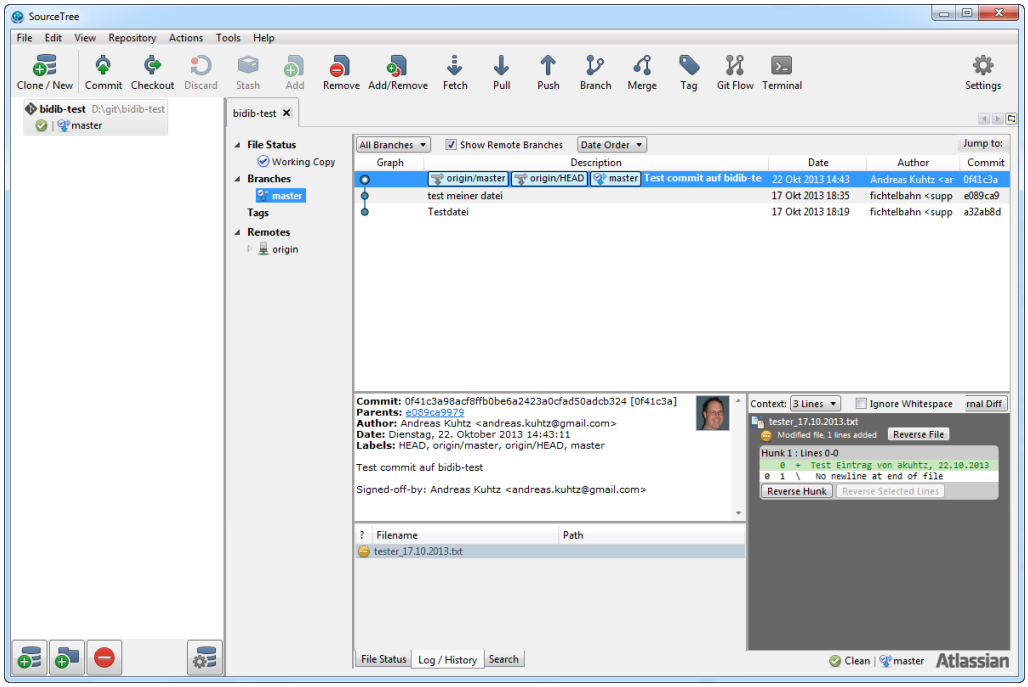

Abbildung 23: Git GUI SourceTree Teil11

Nachdem die Daten heruntergeladen wurden, wird das Repo angezeigt. Für den Screenshot oben hab ich auf Branches > master geklickt. Dadurch wird die Historie angezeigt.

Wenn im lokalen Repo eine Anpassung gemacht wird, erscheint dies unter working copy.
Im Fall der Abbildung unten habe ich ein neues File erzeugt und gespeichert.
Es wird automatisch von SourceTree erkannt.

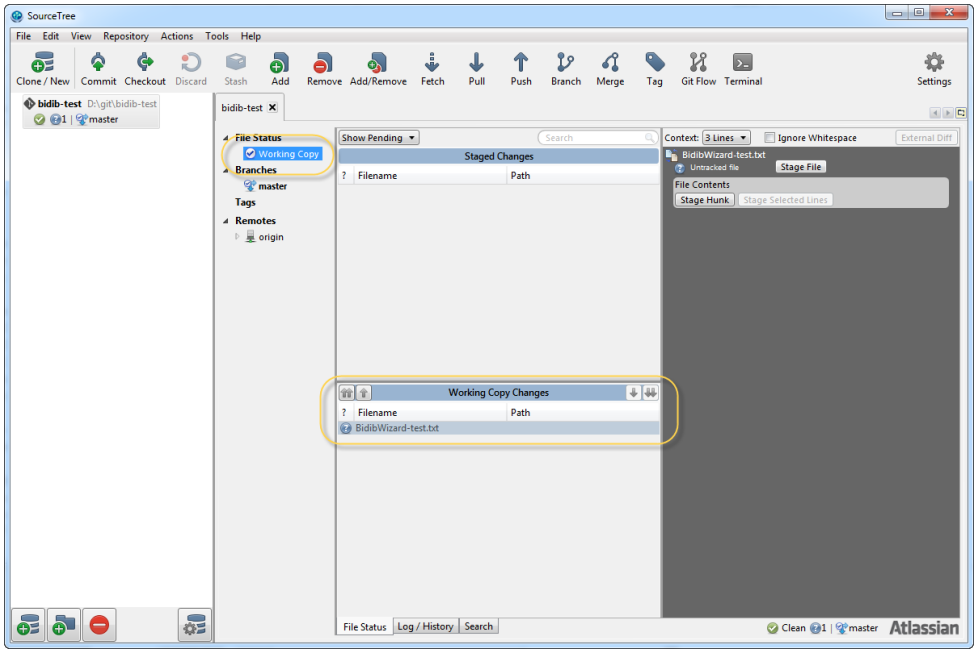

Abbildung 24: Git GUI SourceTree Teil12

Um das File ins Repo hinzuzufügen, geht's über das Context-Menü Add to index.

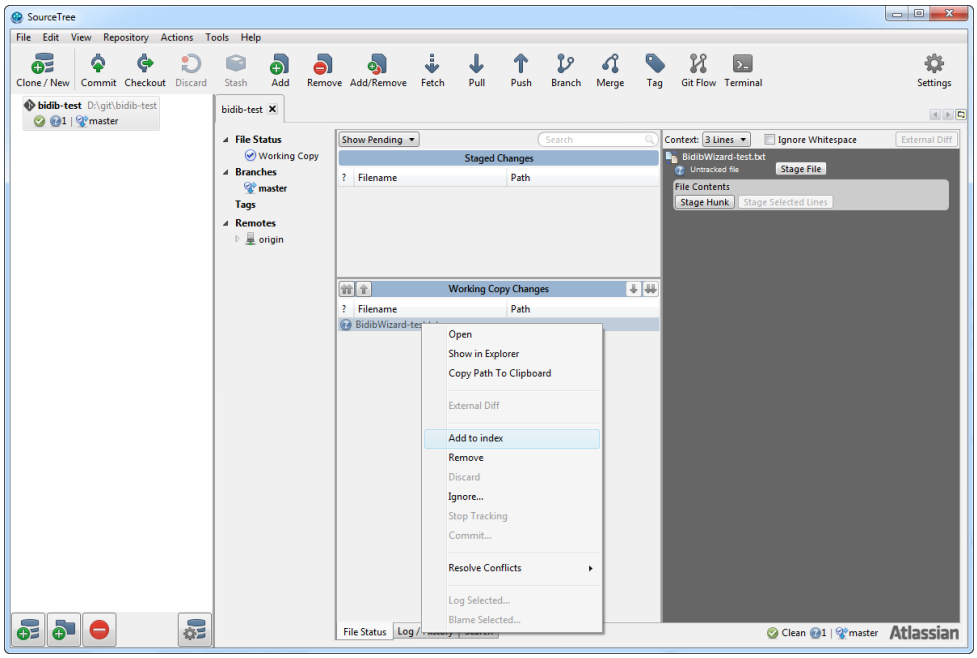

Abbildung 25: Git GUI SourceTree Teil13

Anschließend wird das File unter Staged Changes dargestellt.

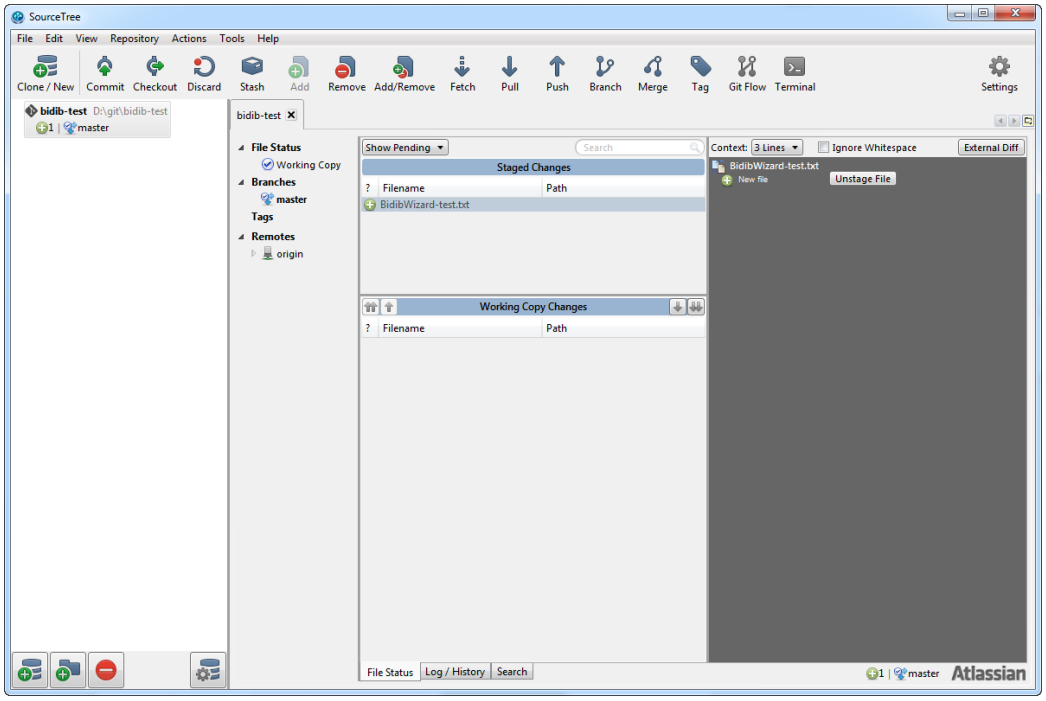

Abbildung 26: Git GUI SourceTree Teil14

Um das File ins Repo zu commiten, wird auf der Working Copy ein Commit gemacht.
Damit sind die Änderungen lokal gesichert!
Das Remote Repo auf dem GitHub Server weiß davon noch nichts.

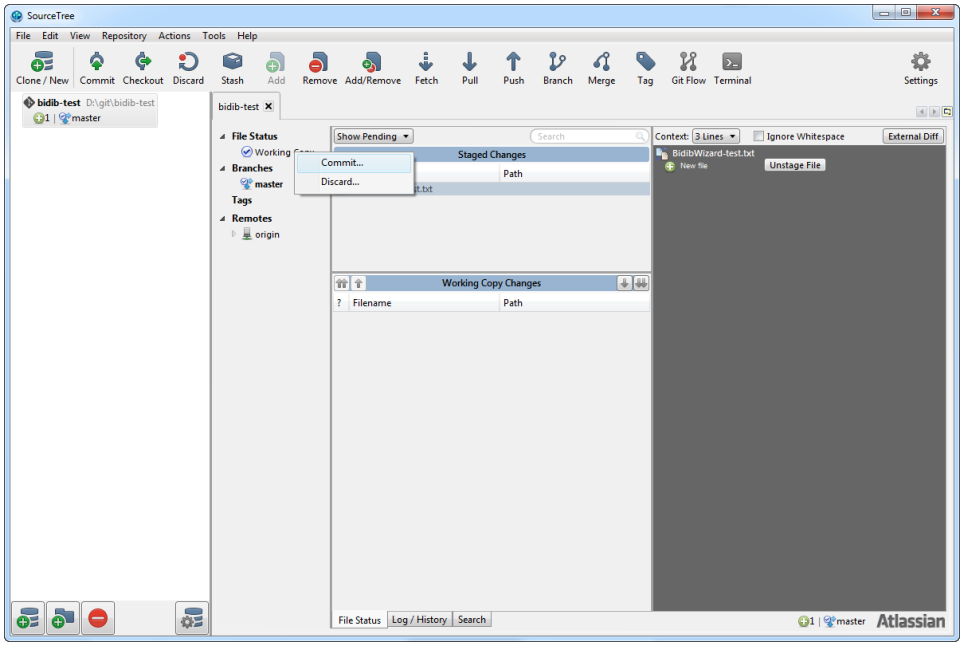

Abbildung 27: Git GUI SourceTree Teil15

Dadurch wird dieser Dialog geöffnet.

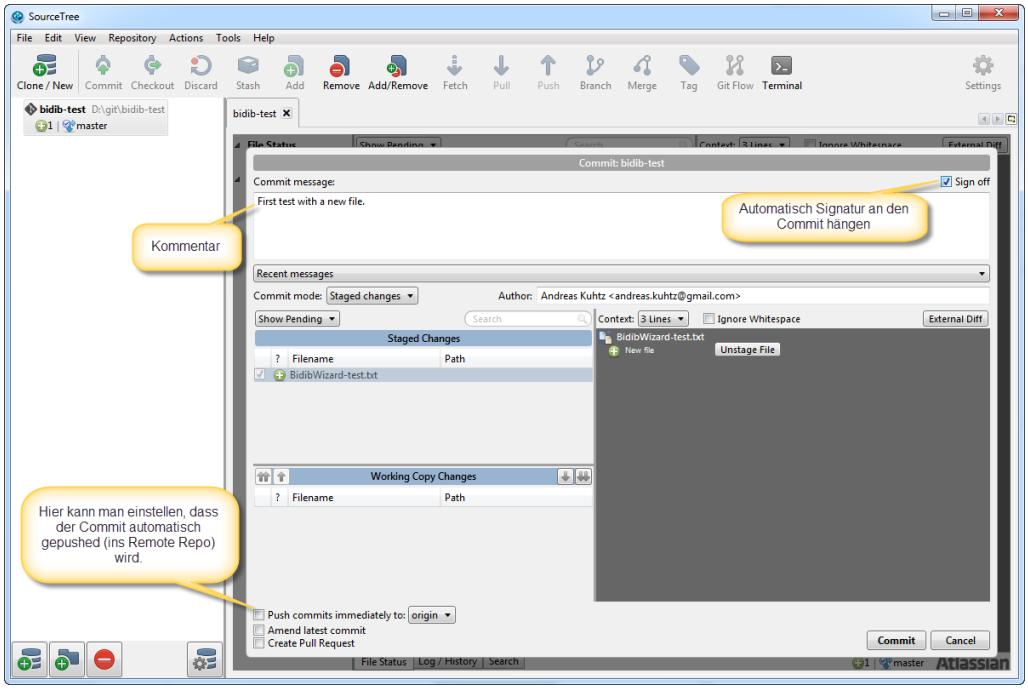

Abbildung 28: Git GUI SourceTree Teil16

Hier kann man die Commit Message angeben.

Wenn die Checkbox „Push commit immediately to origin“ aktiviert ist, dann erfolgt nach dem (lokalen) Commit (Kommando gibt es nicht im Subversion Syntax) automatisch ein Push (Subversion Syntax: Commit) auf das Remote Repo. Wenn diese Checkbox nicht aktiviert ist, muss man das anschließend in einem weiteren Schritt machen.

☝ Man kann auch mehrere Commits gemeinsam pushen …

Für dieses Beispiel habe ich die Checkbox „Push commit immediately to origin“ aktiviert, daher wurde nach dem Commit direkt ein Push gemacht.

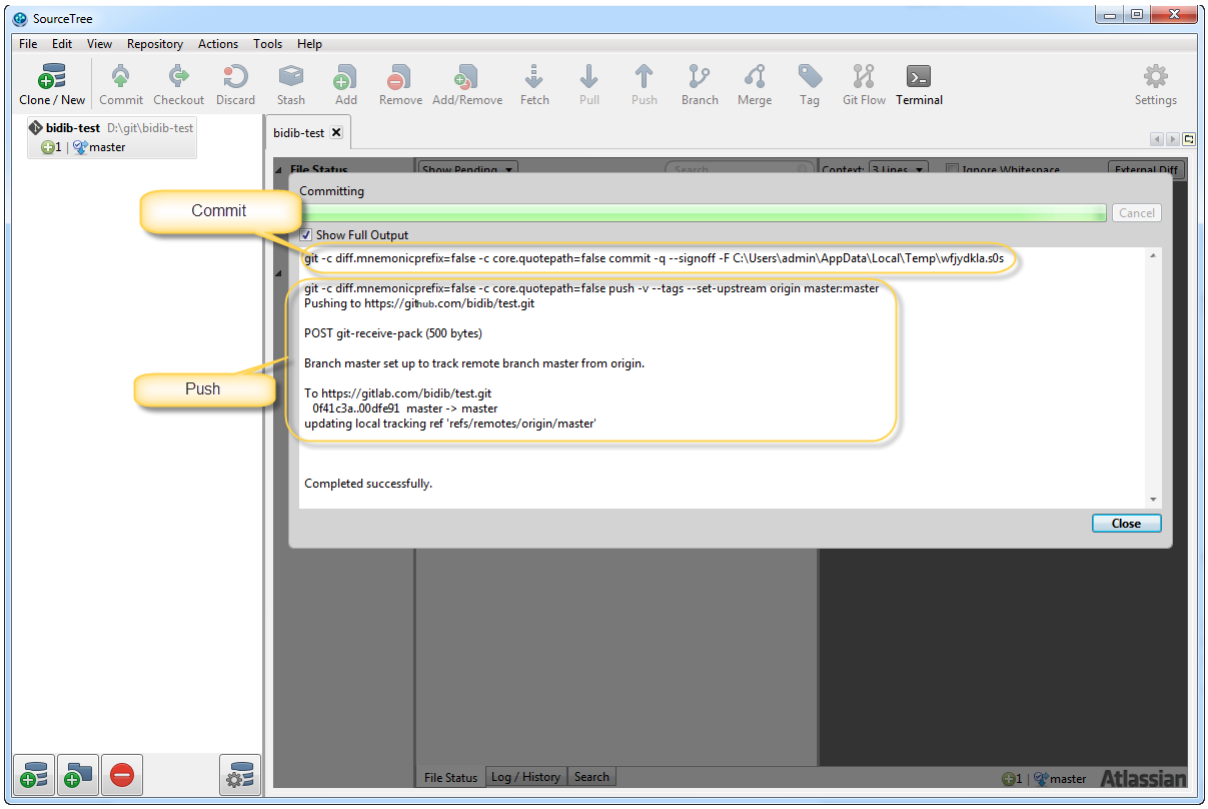

Abbildung 29: Git GUI SourceTree Teil17

Anschließend sind die Änderungen auf dem Remote Repo sichtbar.

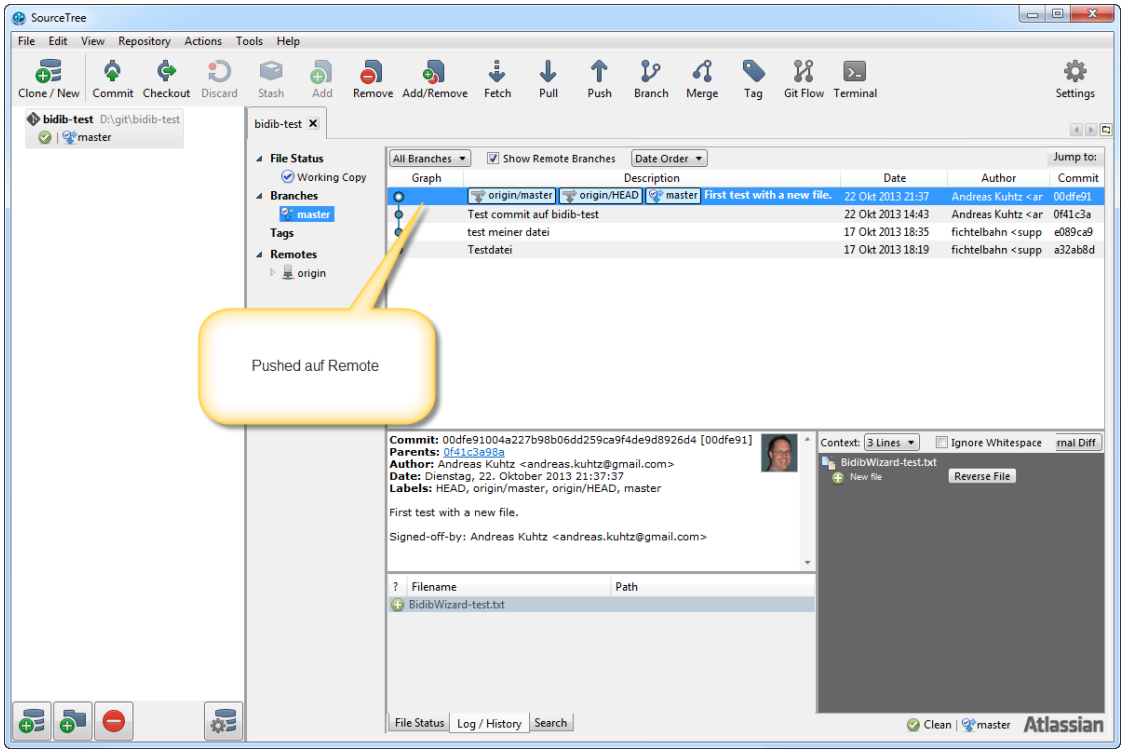

Abbildung 30: Git GUI SourceTree Teil18

Im Browser sieht man die Änderungen auch:

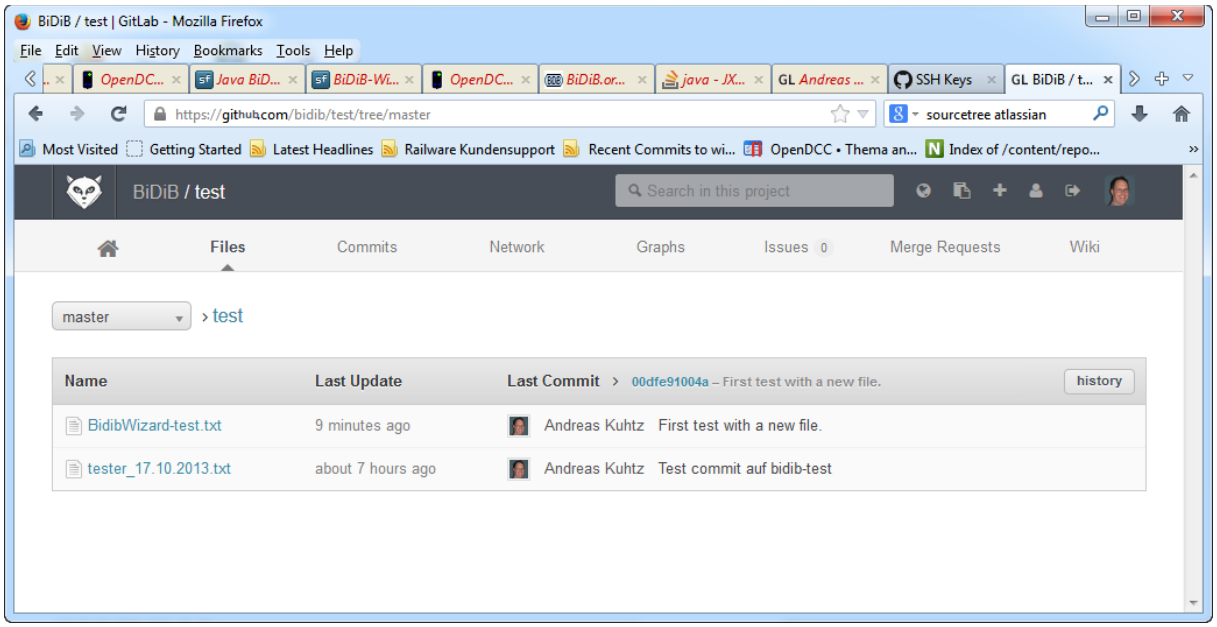

Abbildung 31: Git GUI SourceTree Teil19

8. Wiki im GIT

In der GUI WEB-Oberfläche gibt es zu jedem Projekt ein eigenes Wiki.

In diesem Wiki können Entwicklungsinformationen an andere Entwickler abgelegt werden.

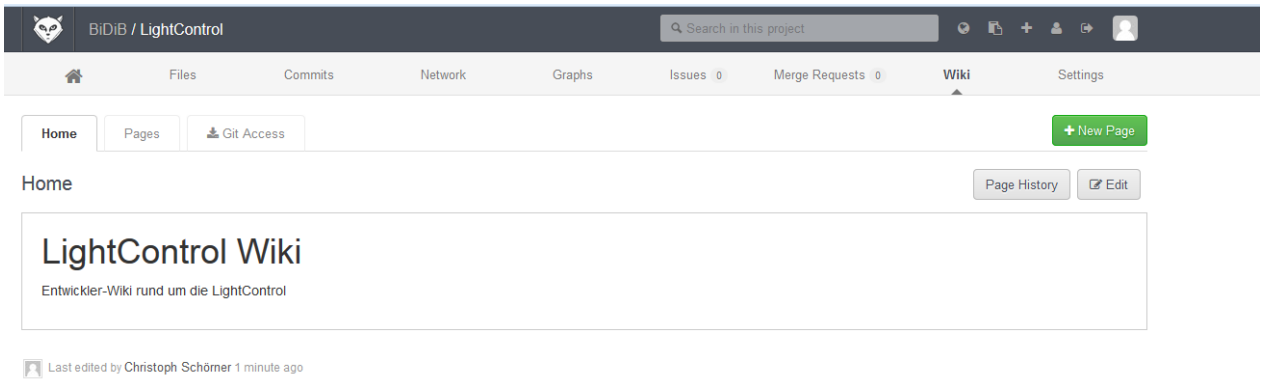

Abbildung 32: Git Wiki

9. Ablauf für eine erfolgreiche Zusammenarbeit

Hier sollten wir noch das Verfahren für die Verwaltung der Strukturen und Dateien festlegen. Ich freue mich auf Vorschläge ☺