Github Tutorial

Tobias Gerlach

16.12.16

Inhaltsverzeichnis

1 Einleitung **2**

2 Was ist GitHub **2**

3 Dein erstes Repository **2**
3.1 Nebenläufiges Arbeiten mit Branches 3
3.2 Änderungen Commiten 4
3.3 Ausführung eines Pull Request 5

4 Hochladen deines Projektes in ein Repository **9**

1 Einleitung

Schön, dass du deinen Weg zu uns in den RoboCub gefunden hast.
Mein Name ist Tobias Gerlach und ich habe Anfang 2016 bei den Lions angefangen.
Wie das hier so ist, wird man meistens sofort ins kalte Wasser geworfen.
Da auch ich diese Leidvolle Erfahrung machen musste habe ich beschlossen dieses Tutorial als Hilfe für alle Software Einsteiger zu schreiben.
Dieses Tutorial ist also wirklich so konzepiert, dass es hoffentlich jeder DAU versteht. :D
Ich bitte ausdrücklich darum meine seltsamen Anglizismen und meine Rechtschreibung zu ignorieren.
Ich freue mich dennoch jederzeit über Feedback.
Nun wünsche ich viel Spaß mit dem Tutorial.

2 Was ist GitHub

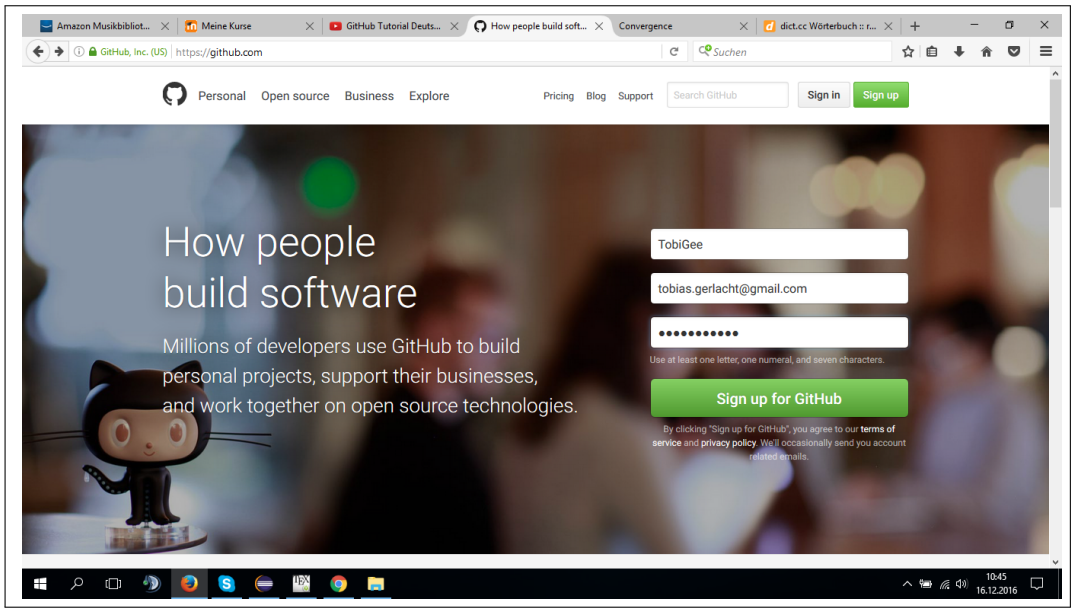
GitHub ist ein Programm, welches der Versionsverwaltung und Sicherung von Quellcode (oder anderem Kram) dient.
Wenn du gerade ein Laborprojekt oä. startest, dann wird dir der Sinn eines Solchen Programms vermutlich noch nicht klar sein. Aber bitte (BITTE!) glaube mir, die Notwendigkeit wird dich einholen.

3 Dein erstes Repository

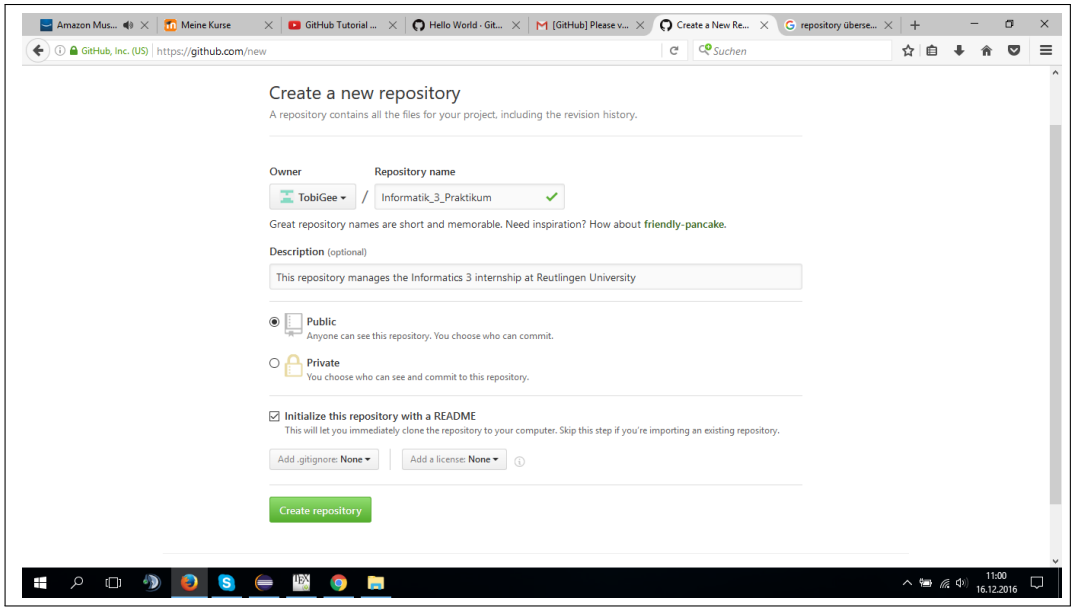
In diesem Kapitel werde ich an einem einfachen Beispiel die Grundlagen von GitHub erläutern.
Viel spaß dabei.

Ein Repository ist sozusagen ein Behälter für unseren Sourcecode.
Für jedes zu verwaltende Programm wird ein neues Repository erstellt.

Um ein Repository zu erstellen brauchst Du zunächst einen GitHub Account.
Diesen kannst du unter der Seite github.com anlegen.



Hast du das getan und dich angemeldet, dann wird dir ein Tutorial vorgeschlagen.
Dieses überspringen wir und klicken auf Start a Project.
Mein Beispielprojekt wird nun das Praktikum aus Informatik 3 sein.
Benutzt also bitte euer eigenes Gehirn, um die Eingabefelder richtig auszufüllen ;) .

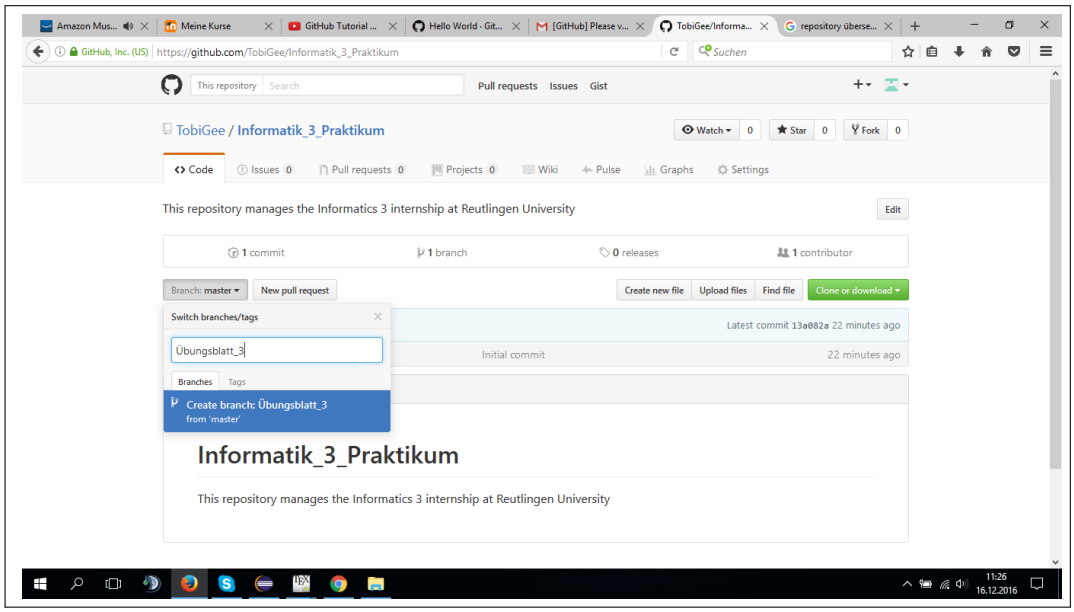


Wir möchten dieses Repository mit einer Readme Datei initialisieren, weswegen und wann wir diese Option nicht wählen, erkläre ich später.
Wir adden weder ein Gitignore, noch eine Lizenz. Dazu später mehr.

Ein Klick auf Create Repository und *zack* du hast dein erstes Repository erstellt.

3.1 Nebenläufiges Arbeiten mit Branches

Im oberen Linken Eck siehst du, in welchem branch (=Ast) du dich gerade bewegst.
Der Wichtigste Ast ist logischerweise der Master Branch. Hier wird die finale Version deines Projekts verwaltet.
Möchtest du etwas an deinem Programm grundsätzlich verändern, wie z.B. Umstellung des Netzwerkprotokolls von UDP auf TCP (aktuelles Beispiel aus der Praxis), dann kann es passieren, dass dein code nicht mehr lauffähig ist.
Gleiches gilt für andere Änderungen und auch Features. Es ist also empfehlenwert im Master Branch immer nur lauffähigen, getesteten code zu verwalten.
Gearbeitet wird dann logischerweise auf anderen Branches. Beim Erzeugen eines neuen Branches aus dem Master Branch wird automatisch eine Kopie des Master Branches erstellt, sodass man direkt änderungen daran vornehmen kann.
So erstellst du einen extra Branch:

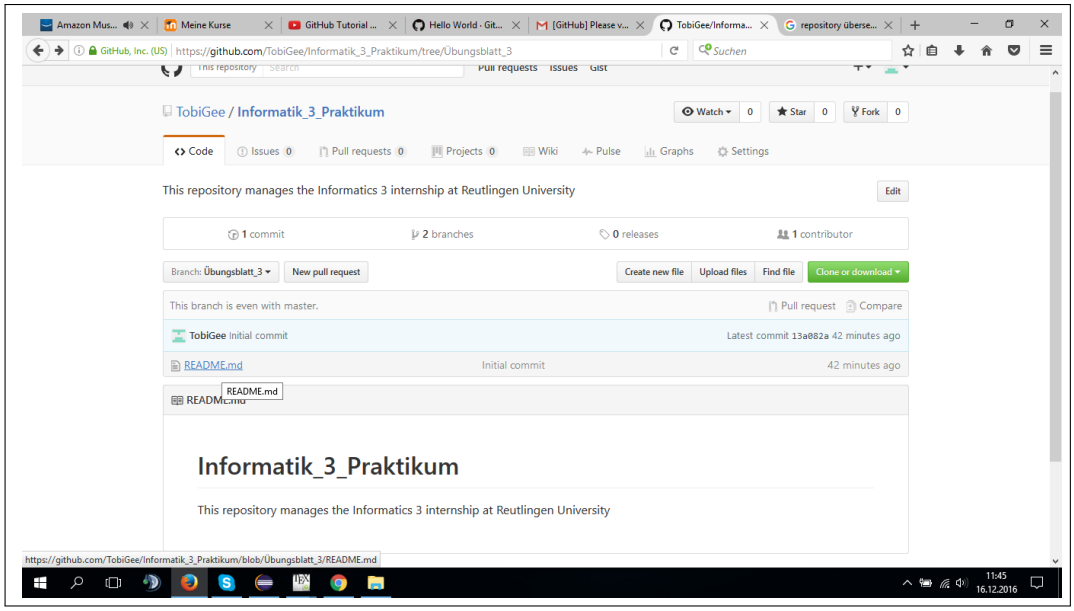


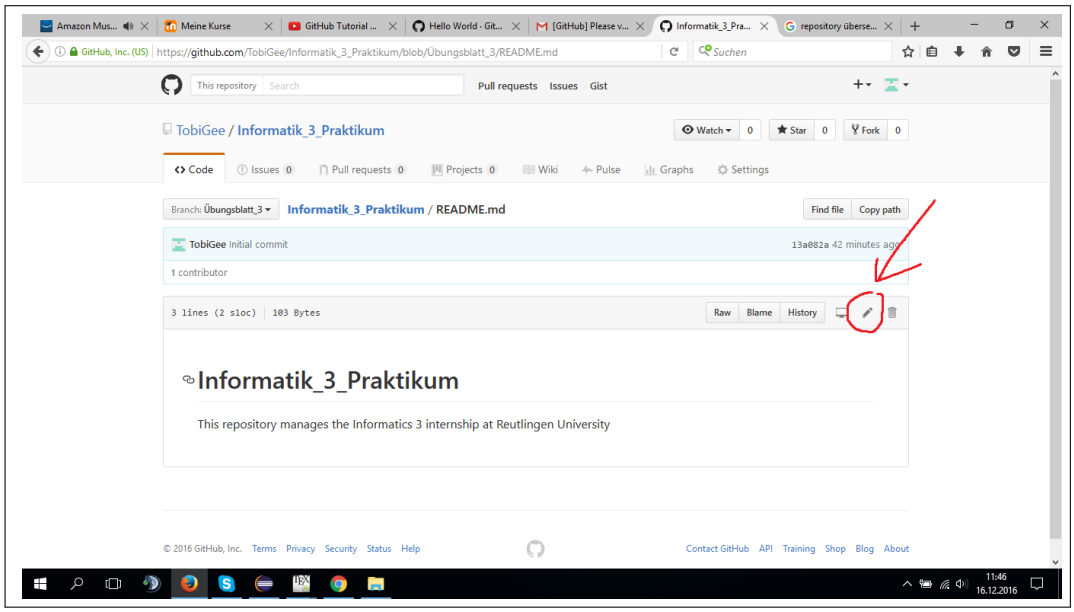
Gratulation du hast soeben deinen ersten Branch erstellt ;D

3.2 Änderungen Commiten

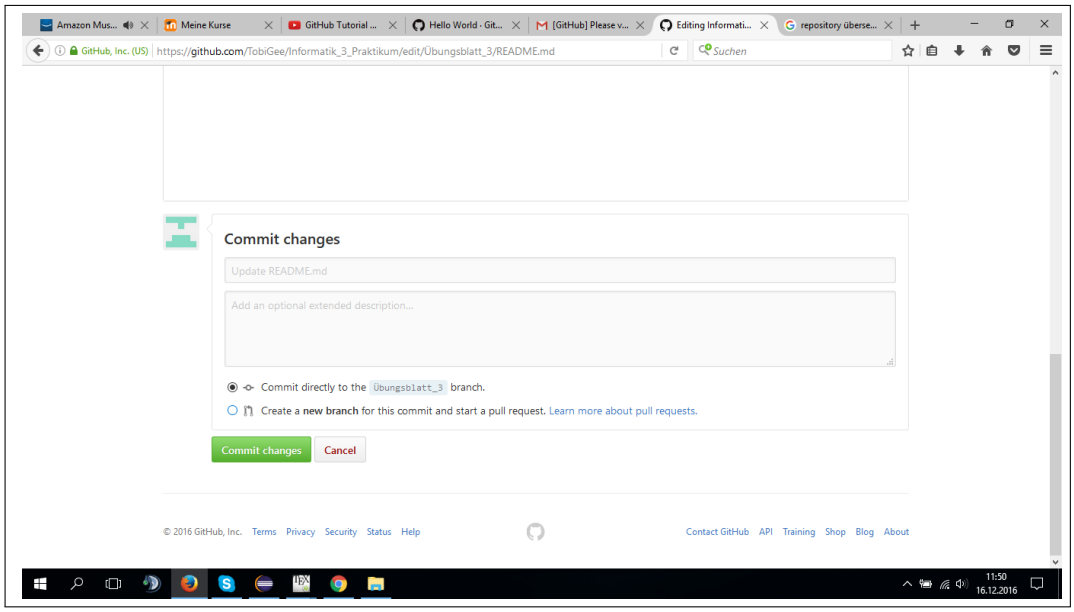
Commiten (De: übergeben) ist im Prinzip nur ein anderes Wort für Speichern im Kontext von GitHub. (Achtung formell ist das falsch, aber es hilft erstmal dem Verständnis)

Wir wollen zunächst eine Änderung in unserem Branch vornehmen, um diese Commiten zu können. Wir klicken in unserem Branch auf die Readme.md Datei und schreiben irgendwas rein.





Wir scrollen etwas nach unten und können(sollten) noch eine Beschreibung der Änderungen hinzufügen.

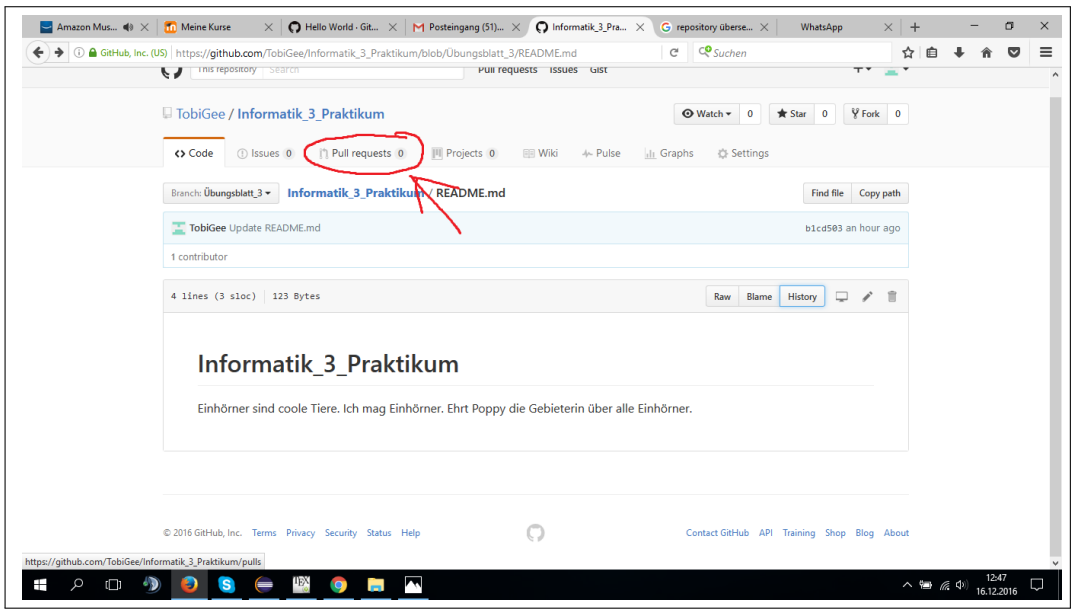


Ein Klick auf Commit Changes und wir haben unseren ersten Commit geschafft.
Ein Klick auf history ermöglicht uns nun unseren Commitverlauf zu betrachten.

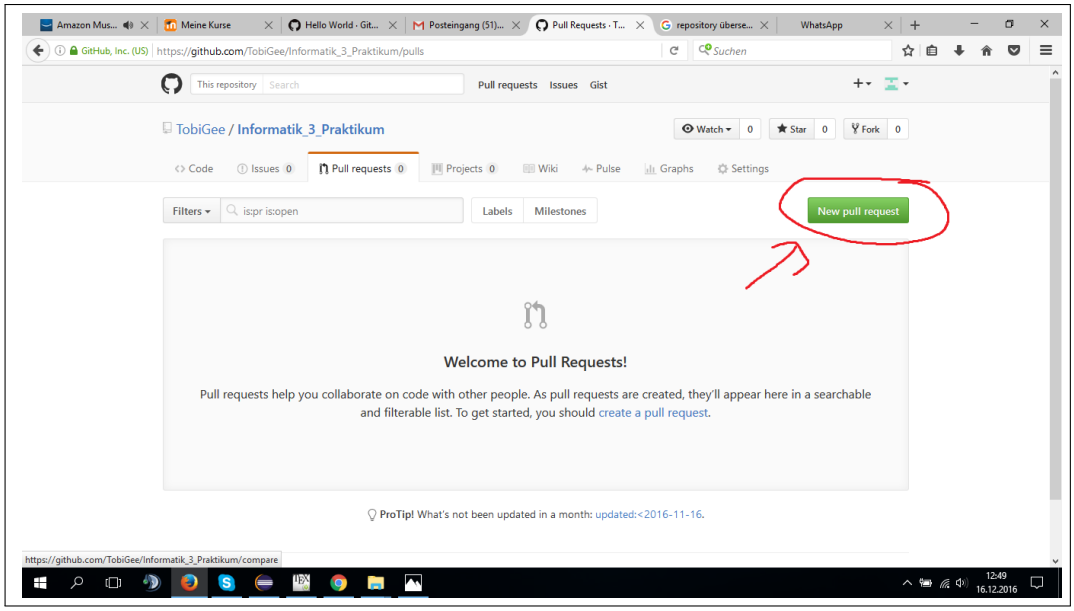
3.3 Ausführung eines Pull Request

Die pull requests sind das Kernstück der Idee von GitHub, wie du gleich sehen wirst.
Da sich unser extra branch nun von dem master branch unterscheidet, ist es nun möglich die Änderungen durch das extra branch an dem master branch zu übernehmen.
Dies geschieht mithilfe von pull requests. Wir werden nun exemplarisch einen pull request durchfüh-

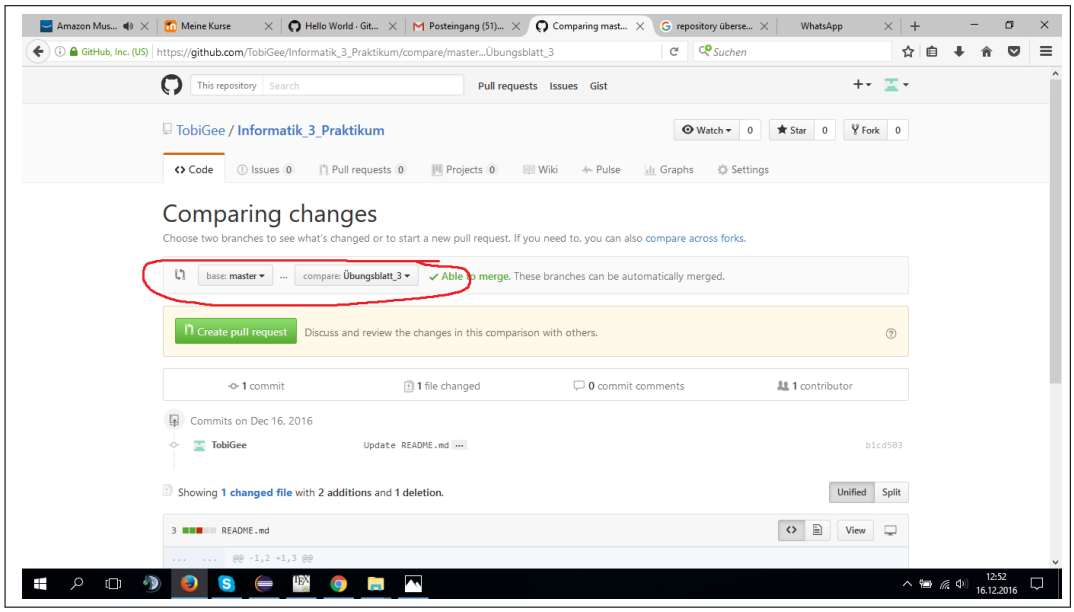
ren. In der Praxis erschließt sich der Sinn meiner Meinung nach am Besten.
Wir gehen hierführ zunächst auf den pull requests Reiter.



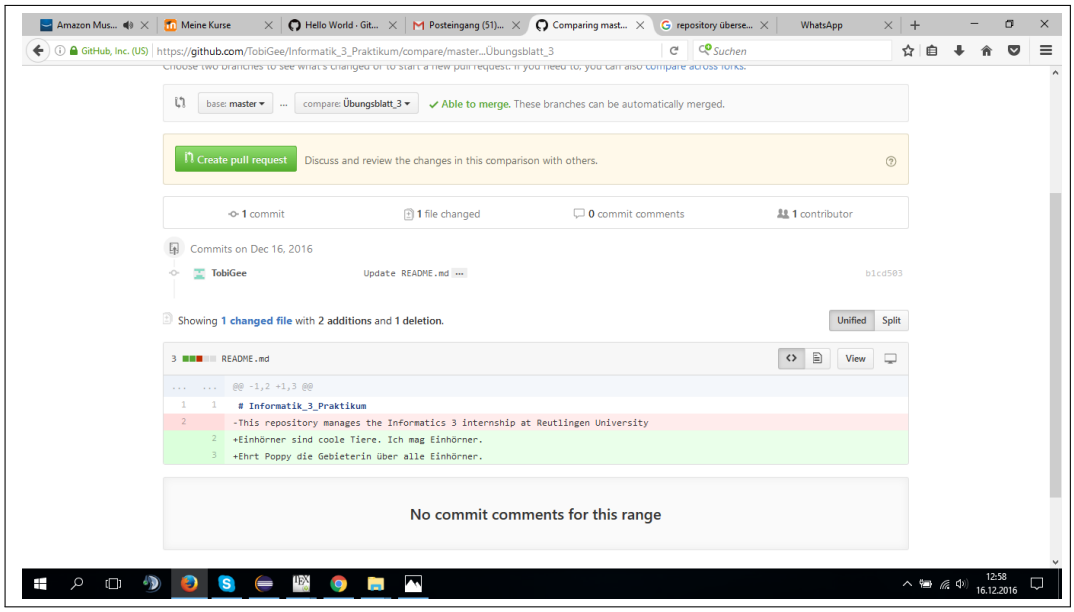
Gefolgt von einem Klick auf new pull request.



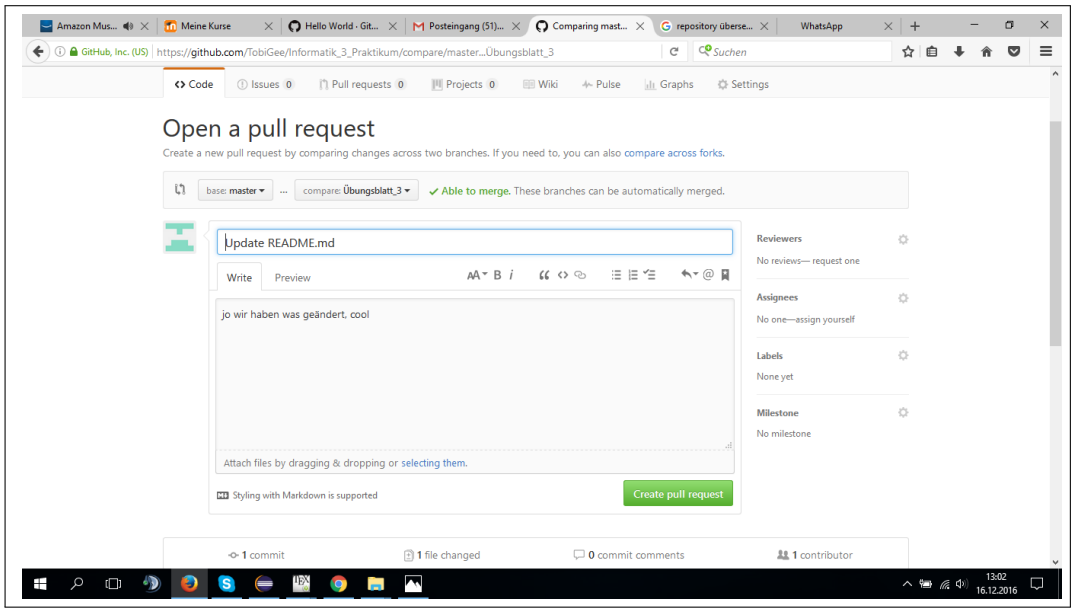
Nun fragt das Programm ab, welche Branches wir vergleichen wollen. Das zu Überschreibende kommt immer an die erste Stelle.



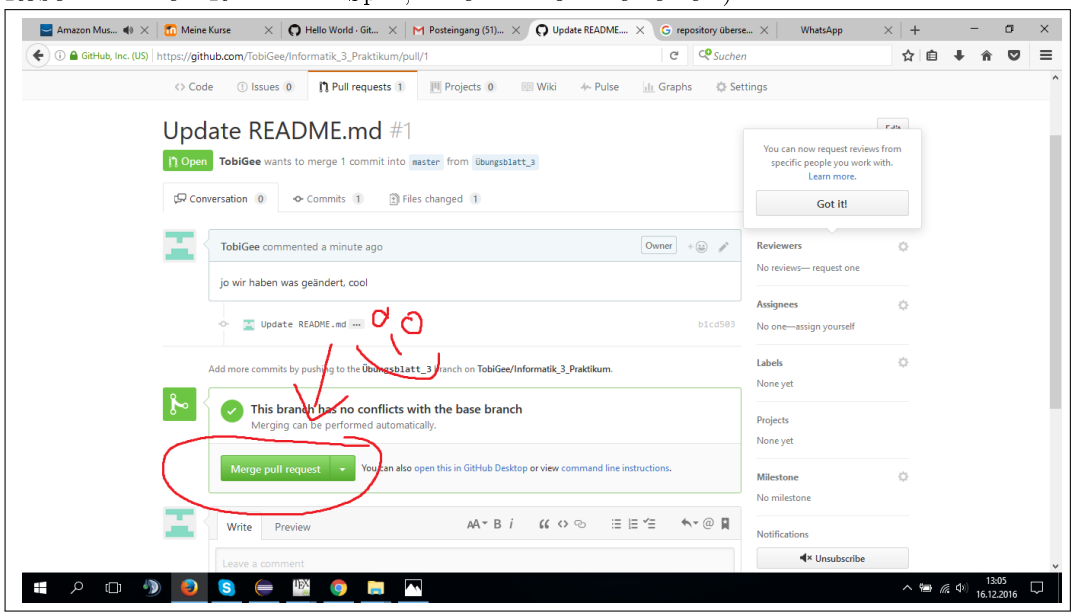
GitHub zeigt uns nun an, was genau am Quellcode modifiziert wurde.
Das Minus steht für eine gelöschte Zeile, das Plus für eine Neue.



Sind wir mit den Änderungen einverstanden, dann drücken wir auf create pull request und kommentieren mit unseren Änderungen.



Nun haben wir einen vollständigen request formuliert. Jeder der des Englischen mächtig ist, wird erkennen, dass man diesen request aber erst ausführen muss, damit seine Änderungen wirksam werden.
Diese geschieht durch einen einfachen klick auf merge pull request.
Der Sinn davon ist nebenbei bemerkt, dass man eine Änderung beim Chef (Herr. Rätsch oder Robert) beantragen kann. Der schaut dann nochmal drüber und kann die Änderung dann mergen. (Das mit Robert und Herr Rätsch war Spaß, außer ihr wollt sie nerven.)

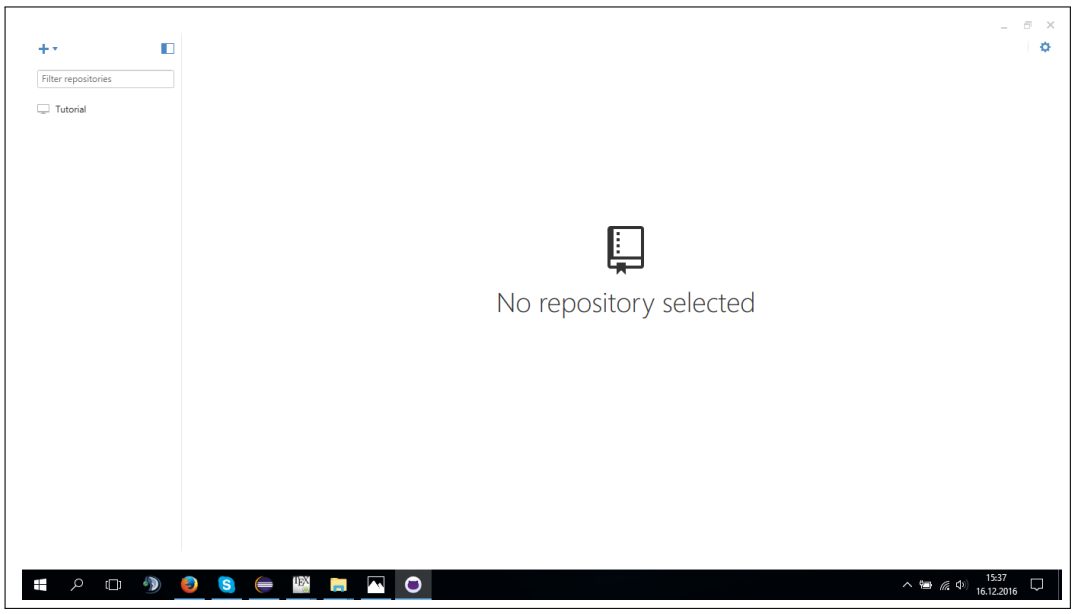


Ein weiteres mal kommentieren und bestätigen führt zudem Ergebnis:
Du hast soeben deinen ersten merge erfolgreich durchgeführt.
Das Programm fragt nun, ob du deinen extra branch löschen möchtst.
In aller Regel wollen wir das nun, die Änderungen sind wirksam auf die master branch übertragen worden.

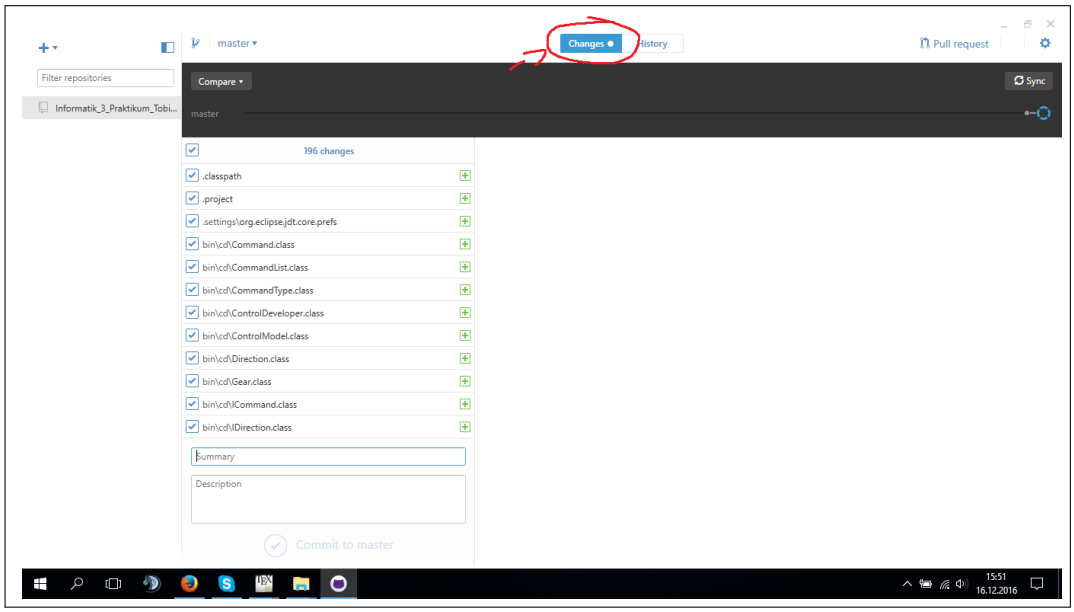
4 Hochladen deines Projektes in ein Repository

Hier gibt es generell den schweren Weg und den leichten.
Leider wird bei uns oftmals der schwere betrieben, dh. die GitHub Verwaltung über die Konsole.
Da das aber nicht gerade anfängerfreundlich ist, empfehle ich folgende Software:
`https://desktop.github.com/`

Sobald du Desktop heruntergeladen hast, musst du dich mit deiner E-Mail und Passwort anmelden.
Wenn du folgendes Menü siehst, hast du alles richtig gemacht.



Nun kannst du deine Projektdatei eifach per Drag and Drop in das Fenster ziehen.
Ist dein Projektordner bereits ein Repository, dann wird es hier nun eingefügt.
Falls nicht Poppt ein Fenster auf, dass den Ordner zu einem Repository machen möchte. Das akzeptieren wir. Der Projektordner wird dabei nicht verändert, es werden nur ein paar Dateien hinzugefügt.
Klicke nun auf Changes, um deine First Version zu committen. (Anmerkung: lokales committen verändert dein repository in der cloud nicht, erst das synchronisieren)



Gebe für den Commit einen Namen und Kommentar ein und schon kannst du deine Version speichern (committen).
Dein Programm kannst du jetzt jederzeit, von jedem Rechner mit deiner Cloud (deinem Repository in GitHub) synchronisieren, indem zu auf sync (rechts oben) klickst. Achte hierbei darauf, dass deine Commits dabei auch gepushed werden.

Die Alternative wäre es hier das Repository mit clone zu clonen und damit eine Kopie zu erstellen. (Tendentiell eher nicht empfohlen)